Facultad 4

Centro de Informática Industrial

Tema: Aplicación multiplataforma para el control de luminarias en un sistema domótico

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Miguel Alejandro Ulloa Borges

Tutor: Ing. Ignais La Paz Trujillo

Co-Tutor: Ing. Rosabel Laches Hernández

Curso docente: 2016-2017.

Declaración de Autoría

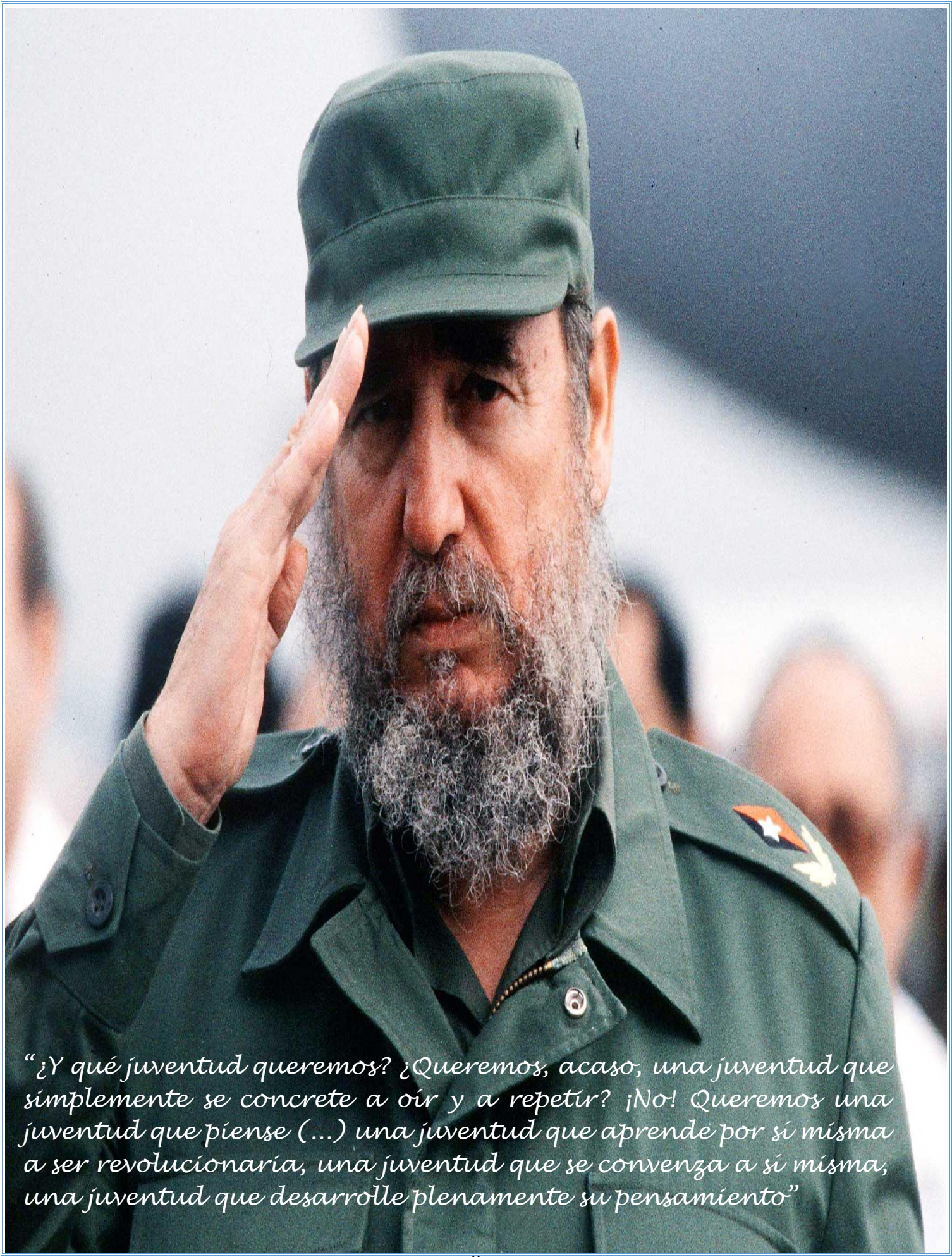
Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los _____ días del mes de _____ del año _____.

Firma del Autor: Miguel Alejandro Ulloa Borges

Firma del Tutor: Ing. Ignais La Paz Trujillo

Firma del Cotutor: Ing. Rosabel Laches Hernández


"¿Y qué juventud queremos? ¿Queremos, acaso, una juventud que simplemente se concrete a oír y a repetir? ¡No! Queremos una juventud que piense (...) una juventud que aprende por sí misma a ser revolucionaria, una juventud que se convenza a sí misma, una juventud que desarrolle plenamente su pensamiento"

Agradecimientos

Agradecer a mi mama por ser mi fuerza y mi apoyo, por darme el impulso para seguir adelante, por no dejarme caer, por escucharme y ser la persona que guarda mis sueños, alegrías, pesares y esperanzas.

A mi padre por creer en mi cuando nadie más lo hizo, por ser más que mi padre, ser mi mejor amigo, por ayudarme a comprender que mis limites los pongo yo.

A mi tía Nancy por toda la ayuda brindada, ayuda que aunque parezca poca significó mucho en mi formación.

Al resto de mi familia por estar siempre en los momentos difíciles, y contribuir con cada acción o palabra a ser la persona que hoy soy.

A mis tutores por su apoyo y comprensión, sé que no he sido el mejor estudiantes, gracias.

A mis amigos Oscar, Yosvani, Richard, Jorgito, Jander, Rogelio, Keiger, Almirola, Jose, Addiel, Ever y demás que fueron acumulándose a lo largo de estos largos, pero estupendos 5 años, no hubiese escogido mejor compañía que la de ustedes para transitar este camino.

Y en especial para el profesor Enrique Carreras Paz.

A todos muchas gracias.

Dedicatoria

Dedico mi trabajo de 5 estupendos años a mis padres, mis hermanos, sobrinos y en especial a mis abuelos. Todos piezas fundamentales en mi formación como profesional y como persona.

Resumen

A nivel mundial, los sistemas domóticos, son una muestra evidente de desarrollo y avance, haciéndose presente en la mayoría de los países desarrollados. Su implementación conlleva a un ahorro significativo de energía, de ahí que la gestión de luminarias sea uno de los módulos principales en este tipo de sistemas. La presente investigación tiene como objetivo implementar una aplicación multiplataforma que permita el control de luminarias en un sistema domótico. Para obtener un resultado satisfactorio se realiza una investigación sobre el campo de la domótica, evaluándose los principales conceptos y estándares sobre dichos sistemas. De esta forma se logra un mayor conocimiento sobre el tema, permitiendo desarrollar la solución propuesta.

Palabras Claves: Domótica, Hiletgo, iOS, microcontroladores, sistema domótico.

Contenido

Índice de Ilustraciones

Índice de Tablas

Introducción

La **automatización** de tareas en el hogar es un tema que gana fuerza en la actualidad, una vez que se desarrolla y utiliza, permite a los usuarios una mayor comodidad y seguridad gracias a la autonomía del sistema implantado, dado por la interacción entre usuario y la interfaz. Todo esto sería posible gracias a la **domótica**. Este término, que se introdujo en España a través de los Pirineos en la década de los noventa, procede del latín domus (casa y hogar) y del griego automática (que actúa de forma autónoma)(1). Se puede decir que domótica es un conjunto de sistemas capaces de automatizar una vivienda o edificio, mediante servicios de gestión de energía, seguridad, bienestar y comunicación; los cuales pueden estar integrados por redes cableadas o inalámbricas; siendo este, sin dudas, un campo en descubrimiento y estudio, que lo hace estar en constante evolución.

A nivel mundial, los sistemas domóticos, son una muestra evidente de desarrollo y avance, haciéndose presente en la mayoría de los países desarrollados, esto es posible gracias a la tecnología de punta con la que cuentan estos. Debido al alto costo de adquisición de estos sistemas y la importancia que representa esta tecnología para el desarrollo del país se hace necesario desarrollar una solución a dicho problema, de ahí que la **Universidad de las Ciencias Informáticas (UCI)**, sea un baluarte en el trabajo científico necesario para obtener resultados relevantes en este aspecto. El **Centro de Informática Industrial (CEDIN)** cuenta con una línea que se dedica al desarrollo de **Sistemas Embebidos**, el cual realiza trabajos relacionados con dispositivos interconectados o controlados a distancia. Actualmente se trabaja en una aplicación móvil que pueda gestionar y controlar un sistema de iluminación mediante una conexión inalámbrica, dicha aplicación se está desarrollando para el sistema operativo **Android**, lo cual limita el uso de esta solo para usuarios de este sistema operativo. Dado que esta solución no llegaría a todos los usuarios de dispositivos móviles, se propone el desarrollo de una aplicación multiplataforma para la gestión de sistemas de iluminación, diversificando de esta manera el alcance de la aplicación hasta todos los usuarios de dispositivos móviles.

Dada la situación problemática expuesta anteriormente se llega al siguiente **problema de investigación**: ¿Cómo lograr el control de luminarias en un sistema domótico basado dispositivos móviles?

Definiéndose como **objeto de estudio**: El control de luminarias en sistemas domóticos.

Con la intención de dar solución a este problema de investigación se define como **objetivo general**: Desarrollar una aplicación multiplataforma para dispositivos móviles capaz de realizar el control de luminarias en un sistema domótico.

Enmarcándose el **campo de acción** en: Aplicaciones móviles para el control de luminarias en sistemas domóticos.

Las siguientes **tareas de investigación** se proponen para dar cumplimiento a la solución propuesta:

1. Identificación del funcionamiento del proceso de negocios.
2. Identificación de los requisitos funcionales para el desarrollo de la solución propuesta a través de las entrevistas realizadas.
3. Generación de los artefactos relacionados con el análisis y diseño de la solución propuesta.
4. Propuesta de la arquitectura del sistema cumpliendo con las políticas de software libre.
5. Propuesta de un mecanismo de seguridad para la obtención de comandos generada mediante la comunicación entre microcontroladores y dispositivos móviles.

Para dar cumplimiento a las tareas de investigación descritas anteriormente, se utilizarán los siguientes métodos de investigación:

Métodos de investigación:

- **Análisis-síntesis:** se aplica en el estudio de los fundamentos y teorías relacionadas con la domótica y con los sistemas de medición de datos.
- **Observación participante**: utilizado en el seguimiento del desarrollo de aplicaciones en el área de la domótica haciendo uso de dispositivos móviles inteligentes.
- **Análisis de documentos**: relacionado con la consulta de literatura especializada relacionada con la teoría de la automatización de procesos.
- **Histórico-lógico**: se aplica en el estudio de la evolución de diferentes sistemas domóticos.

- **Investigación-acción:** es utilizado para la constante realización de pruebas de concepto y prototipos tanto no funcionales como funcionales.

La estructura de la tesis estará compuesta por la introducción, tres capítulos, conclusiones y recomendaciones. Los capítulos se distribuirán de la siguiente forma:

- **Capítulo 1. Fundamentación teórica**. Este capítulo abarcará conceptos y teorías asociadas al desarrollo de aplicaciones móviles, y su interacción con microcontroladores para su uso en sistemas domóticos.

- **Capítulo 2. Análisis y diseño**. En este capítulo se aborda la propuesta para dar solución al problema planteado. También se puntualizan los procesos asociados al análisis y diseño de la aplicación.

- **Capítulo 3. Implementación y pruebas.** En este capítulo se expone el proceso de desarrollo de la solución, se realizarán pruebas para garantizar el correcto funcionamiento de la aplicación, revisando si cumple con lo deseado, además se exponen los resultados de las pruebas.

Capítulo 1 Fundamentación teórica

Introducción

Este capítulo hará referencia a los conceptos fundamentales relacionados con la domótica, los dispositivos interconectados y las aplicaciones móviles, de forma que se logre un mayor entendimiento del tema. También se seleccionarán las herramientas y tecnologías a utilizar para desarrollar la solución propuesta.

1.1 Domótica

Existen muchas definiciones de domótica, todas con una perspectiva diferente. Según la Asociación Española de Domótica, esta sería “La incorporación al equipamiento de nuestras viviendas y edificios de una sencilla tecnología que permita gestionar de forma energéticamente eficiente, segura y confortable para el usuario los distintos aparatos e instalaciones domésticas tradicionales que conforman una vivienda.”(2). Por otra parte, Antonio J. Jara Varela propone que el término domótica nace del neologismo francés ‘domotique’, el cual procede de la palabra latina domus (casa) y del francés telematique (telecomunicación-informática), refiriéndose además como “conjunto de servicios de la vivienda garantizado por sistemas que realizan varias funciones, los cuales pueden estar conectados entre sí, y a redes interiores y exteriores de comunicación” o “instalación e integración de varias redes y dispositivos electrónicos en el hogar, que permite automatizar actividades cotidianas de forma local o remota, de la vivienda o edificio”(3).

1.1.1 Componentes

Este nuevo mundo tecnológico ha cobrado fuerza, y esto se debe a una serie de características muy propias de estos sistemas, las cuales se muestran en las ilustraciones 1 y 2, tales como:

- La interfaz debe ser sencilla y agradable, permitiéndole al usuario una mejor y mayor adaptación, así como una mejor comunicación con el sistema.
- La flexibilidad de los sistemas, dado que estos deben dar la mayor comodidad posible, deben permitir ampliar las necesidades existentes y futuras, dado que una vez implementados pueden ser expandidos sin que se deban rediseñar por completo.

- Su implementación se debe efectuar por medio de módulos independientes, lo que genera mayor fiabilidad en los sistemas, ya que algún fallo en uno de los módulos no va a afectar a otro, y la implementación de módulos nuevos, no perjudicara los anteriores.
- La posibilidad de controlar los servicios por medio de un control remoto o una aplicación móvil, dando la facilidad de acceder y gestionar de forma eficiente los procesos del hogar desde cualquier parte del mismo.

Ilustración 1 Interfaz del sistema domótico

Ilustración 2 Flexibilidad del sistema

Aunque se piense que estos sistemas podrían ser solo un lujo o una manera de atraer a nuevos usuarios a comprar casas más costosas, se puede afirmar que esta nueva tecnología ayuda al ahorro de energía eléctrica, dado que controla las luces, los equipos electrodomésticos y sistemas de seguridad. La domótica está concebida para el ahorro de energía, el incremento en los niveles de seguridad, mayor y mejor control centralizado sobre todas las áreas o habitaciones, mejor comunicación, optimización de recursos, automatización, ahorro de tiempo, calidad de vida y sobre todo un alto grado de confort (3). La domótica posee una amplia gama de objetos que se pueden automatizar, desde un sistema de luminarias hasta las puertas y ventanas de una habitación, se puede controlar los equipos electrodomésticos, su encendido y apagado, regaderas de agua en jardines o la climatización del recinto; prácticamente todo en el hogar se puede hacer parte de un sistema domótico. Para la implementación de un edificio domótico se debe desplegar una serie de dispositivos encargados de hacer que todo el engranaje funcione y se comunique.

Entre ellos se encuentran los actuadores, los sensores y los controladores, los cuales se muestran en las ilustraciones 3, 4, 5, 6, 7, 8:

- Los actuadores son los encargados de recibir la orden desde el controlador y realizar las acciones requeridas, activando desde una válvula hasta un motor o rotor.

Ilustración 3 Actuador #1

Ilustración 4 Actuador #1

- Los sensores son los dispositivos capaces de detectar señales eléctricas, físicas o químicas para ser interpretadas y procesadas, ejemplos de sensores serían los detectores de humo o de medición de temperatura.

Ilustración 5 Sensor#2

Ilustración 6 Sensor#1

- Los controladores serían los encargados de interactuar con el usuario de manera tal que se complete así la red de comunicación usuario-sistema, permitiéndole realizar las funciones de forma remota o presencial.

Ilustración 7 Controlador#1

Ilustración 8 Controlador#2

1.1.2 Aplicaciones móviles para domótica

Es muy común escuchar el termino *smartphones*, pero realmente pocos saben porque se les denomina así. Emparejado al desarrollo tecnológico, la telefonía móvil ha ido avanzando, y dado el auge de usuarios de dispositivos móviles, se abría así el campo para que estos también evolucionaran. Los *smartphones* son teléfonos inteligentes, según su traducción al español, pero lo que los hace inteligentes es la posibilidad de instalar aplicaciones y su capacidad de reconocer patrones y detectar gestos, son híbridos entre un celular y una computadora, no tienen la potencia de una PC, pero tampoco son simples teléfonos, además poseen la mayoría de las principales tecnologías de comunicación. Al tener la habilidad de conectarse entre ellos, o a otros dispositivos mediante una conexión inalámbrica, los hace merecedores de un lugar en la carrera por dominar el campo de la domótica, de ahí que muchos de los proyectos de este tipo hacen referencia o incluyen a los *smartphones* como un factor de suma importancia.

Algunas de las aplicaciones móviles para uso en la domótica que se han desarrollado son:

- ***Houseinhand knx*:** Esta aplicación para dispositivos iOs de *Apple* o *Android* te permite controlar tu casa de una forma rápida e intuitiva. Con ella podrás manejar a distancia y en tiempo real dispositivos knx (luces, persianas, climatización), audiovisuales (televisión, dispositivos de audio, DVD), videoporteros y cámaras ip (axis y *mobotix*) estés donde estés(4)

***.PhilipsHue*:** App que permite controlar en remoto los productos de iluminación Hue de la marca Philips que tengas en casa desde el iPhone o el iPad. Permite configurar la iluminación idónea para cada ocasión. También disponible para Android (instalado en la casa), para comunicarse con hardware compatible con *Insteon* y x10(4).

- ***Nexho*:** App para *iOs* y *Android* destinada al control domótico de la vivienda, con diferentes menús que permiten gestionar elementos como la climatización, iluminación, persianas y cerramientos, equipos eléctricos genéricos, alarmas de inundación, de incendio, de intrusión(4).
- ***WeMo Light Switch*:** A través de la *Wifi*, permite encender y apagar las luces de las habitaciones automatizadas desde cualquier lugar. El sistema reemplaza a los interruptores de luz estándar en un recinto cerrado y puede ser controlado de forma remota con un dispositivo móvil inteligente. WeMo trabaja con una red Wifi y en cualquier lugar usando un teléfono 20 inteligente o tableta tiene una conexión a

internet (3G o 4G LTE). El sistema es compatible con dispositivos de la compañía Apple que usan sistema operativo iOs 5 o versiones más actualizadas(5).

Todas estas aplicaciones son realmente útiles, además son altamente valoradas en el mundo de la domótica, pero la mayoría son de carácter propietario, ya que se debe pagar una licencia para su uso, a lo que se le agrega que están diseñados para la comunicación con sensores y hardware muy costosos y difíciles de conseguir en Cuba, como son ***Nexho*** y ***WeMo Light Switch***. Por su parte aquellas aplicaciones antes mencionadas que se pueden adquirir de forma gratuita cuentan con el impedimento de que son implementadas para sistemas a la medida o sea sistemas generalizados con un hardware específico y para un sistema domótico ya estructurado, en este caso se encuentran ***PhilipsHue***. De ahí que se decida realizar una aplicación capaz de solventar los problemas de compatibilidad de hardware y sistema, permitiendo su uso en todos los sistemas domóticos a implementar.

1.2 Sistemas operativos para dispositivos móviles

Un sistema operativo móvil o SO móvil permite controlar un dispositivo móvil al igual que las computadoras utilizan Windows, Linux o Mac OS. Sin embargo, los SO móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos(6)-

Entre los más populares se encuentran *Android*, iOS, *Black Berry* y *Windows Phone,* en la ilustración 9 se ve una comparación de su uso a nivel mundial hasta principios de 2017, según datos *Android* es el SO móvil más utilizado, seguido iOS teniendo el control total del mercado con 66,71% y 29,55% respectivamente, de estos datos arrojados en dicha comparativa se decide profundizar en iOS y *Android* durante la investigación(7).

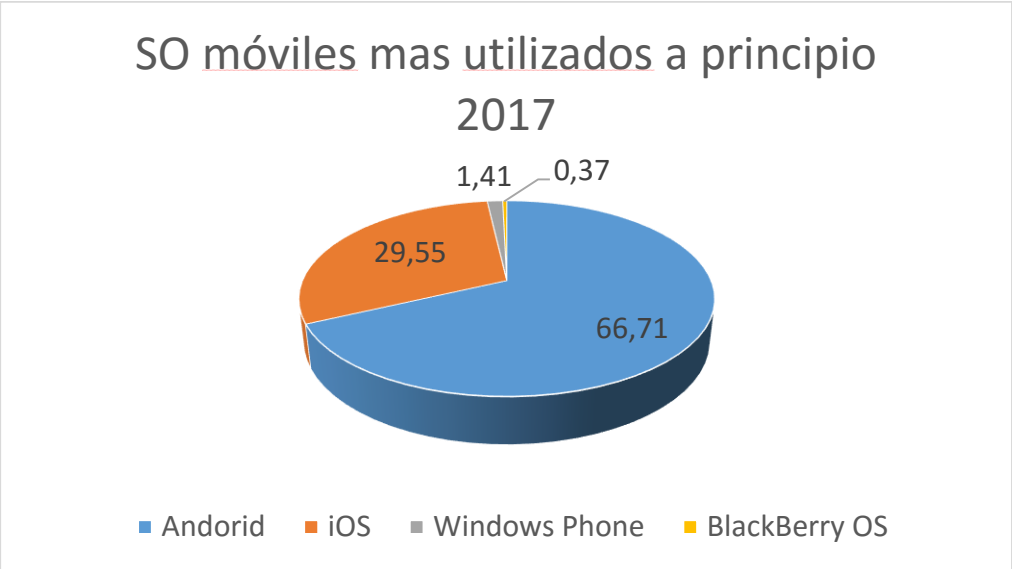


Ilustración 9 Comparativa SO móviles más utilizados

IOS:

IOS (anteriormente denominado *iPhone OS*) es un sistema operativo móvil de Apple. Originalmente desarrollado para iPhone, siendo después usado en dispositivos como *iPod Touch, iPad* y el *Apple TV*. *Apple, Inc*. no permite la instalación de iOS en hardware de terceros. Tenía el 26% de cuota de mercado de sistemas operativos móviles vendidos en el último cuatrimestre de 2010, detrás de *Google Android* y *NokiaSymbian*. En mayo de 2010 en los Estados Unidos, tenía el 59% de consumo de datos móviles (incluyendo el *iPod Touch y el iPad*). La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal). IOS se deriva de Mac OS X, que a su vez está basado en *Darwin BSD*, y por lo tanto es un sistema operativo Unix. IOS cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios" y la capa de "Cocoa Touch". La versión actual del sistema operativo (iOS 6.0) ocupa más o menos 770 megabytes, variando por modelo(8).

Android:

Android es un sistema operativo basado en Linux, desarrollado por la empresa *Android Inc*., creado para teléfonos móviles inteligentes y el cual era prácticamente desconocido hasta que en 2005 Google lo compró. Hasta noviembre de 2007 sólo hubo rumores, pero en esa fecha se lanzó la *Open Handset Alliance*, que agrupaba a muchos fabricantes de teléfonos móviles, chipsets y Google y se proporcionó la primera versión de Android una plataforma para dispositivos móviles construida sobre la versión 2.6 de *Linux*, junto con el SDK para que los programadores empezaran a crear sus aplicaciones para este sistema.

Aunque los inicios fueran un poco lentos, debido a que se lanzó antes el sistema operativo que el primer móvil, rápidamente se ha colocado como el sistema operativo de móviles más vendido del mundo, situación que se alcanzó en el último trimestre de 2010.

Android ha visto numerosas actualizaciones desde su liberación inicial. Estas actualizaciones al sistema operativo base típicamente arreglan bugs y agregan nuevas funciones(9).

Ventajas:

- El código de Android es abierto: Google liberó Android bajo licencia Apache. Cualquier persona puede realizar una aplicación para Android.
- Hoy día hay más de 650.000 aplicaciones disponibles para teléfonos Android, aproximadamente 2/3 son gratis. Además la libertad de código permite adaptar Android a bastantes otros dispositivos además de teléfonos celulares. Está implantado en Tablets, GPS, relojes, microondas… incluso hay por internet una versión de Android para PC.
- El sistema Android es capaz de hacer funcionar a la vez varias aplicaciones y además se encarga de gestionarlas, dejarlas en modo suspensión si no se utilizan e incluso cerrarlas si llevan un periodo determinado de inactividad. De esta manera se evita un consumo excesivo de batería. Esta es una de sus mayores ventajas por la rapidez con la que carga una aplicación abierta previamente(10).

Desventajas:

- A pesar de ser una ventaja el ser un sistema multitarea: El hecho de tener varias aplicaciones abiertas hacen que el consumo de la batería aumente y como no todas las aplicaciones Android las cierra hay que instalar una aplicación para que las cierre. En la Market de Android hay una buena cantidad de aplicaciones para este fin, así que el problema es solucionable pero debería venir pre instalado de fábrica.
- Duración de la batería: la batería en un celular Android se agota muy rápido. Utilizando las aplicaciones de manera moderada la batería puede llegar a durar más, pero para un usuario que usa mucho sus aplicaciones la batería no tiende a durar, lo que se puede solucionar con algunas aplicaciones, pero volvemos a lo mismo no viene pre instalada de fábrica, hace falta una aplicación externa al sistema Android para optimizar mejor la batería.
- Poco intuitivo: Para la mayoría el sistema operativo es muy complicado. Por ejemplo se vuelve complicado configurar el teléfono, esto te puede llevar mucho tiempo, y esto es generado por la interfaz de Android. Hay aplicaciones que ayudan en tareas que deberían ser sencillas como desinstalar otras aplicaciones pero, otra vez, volvemos a lo mismo se hace necesario instalar aplicaciones para solucionar el problema(10).

1.3 Microcontroladores

Un microcontrolador es un circuito integrado digital que puede ser usado para muy diversos propósitos debido a que es programable. Está compuesto por una unidad central de proceso (CPU), memorias (ROM y RAM) y líneas de entrada y salida (periféricos), tiene los mismos bloques de funcionamiento básicos de una computadora lo que nos permite tratarlo como un pequeño dispositivo de cómputo.

Un microcontrolador puede usarse para muchas aplicaciones algunas de ellas son: manejo de sensores, controladores, juegos, calculadoras, agendas, avisos lumínicos, secuenciador de luces, cerrojos electrónicos, control de motores, relojes, alarmas, robots, entre otros. Como el *hardware* ya viene integrado en un solo chip, para usarse se debe especificar su funcionamiento por *software* a través de programas que indiquen las instrucciones que el microcontrolador debe realizar. Los lenguajes de programación típicos que se usan para este fin son ensamblador y C pero antes de grabar un programa al microcontrolador hay que compilarlo a hexadecimal que es el formato con el que funciona el microcontrolador(11).

Para diseñar programas es necesario conocer los bloques funcionales básicos del microcontrolador, estos bloques son:

- **CPU** (Unidad central de proceso)
- **Memoria ROM** (Memoria de solo lectura)
- **Memoria RAM** (Memoria de acceso aleatorio)
- **Líneas de entrada y salida** (Periféricos)

La forma en la que interactúan estos bloques dependerá de su arquitectura

La CPU posee, de manera independiente, una memoria de acceso rápido para almacenar datos, esta memoria se denomina registro, si estos registros son de 8 bits se dice que el microcontrolador es de 8 bits(11).

1.4 Protocolo de comunicación

Existen varios métodos de comunicación inalámbrica que permiten la transferencia de datos entre dispositivos y facilitan su conexión. Entre ellos se encuentra el Infrarrojo, el *Bluethooth* o la *Wifi*.

1.4.1 Infrarrojo

En el caso del **Infrarrojo** se trata de una tecnología de transmisión inalámbrica por medio de ondas de calor a corta distancia, capaces de traspasar cristales. Tiene una velocidad promedio de transmisión de datos hasta de 115 Kbps (Kilobits por segundo), no utiliza ningún tipo de antena, sino un diodo emisor semejante al de los controles remoto para televisión. Funciona solamente en línea recta, debiendo tener acceso frontal el emisor y el receptor ya que no es capaz de traspasar obstáculos opacos.

1.4.2 *Bluetooth*

Bluetooth es una tecnología de transmisión inalámbrica por medio de ondas de radio de corto alcance (1, 20 y 100 m a la redonda dependiendo la versión). Las ondas pueden incluso ser capaces de cruzar cierto tipo de materiales, incluyendo muros. Para la transmisión de datos no es necesario el uso de antenas externas visibles, sino que pueden estar integradas dentro del mismo dispositivo. Este tipo de transmisión se encuentra estandarizado de manera independiente y permite una velocidad de transmisión de hasta 1 Mbps(12) .

1.4.3 IEEE 802.11 (Wifi)

Wifi es una tecnología de comunicación inalámbrica que permite conectar a internet equipos electrónicos, como computadoras, tablets, smartphones o celulares, mediante el uso de radiofrecuencias o infrarrojos para la trasmisión de la información. Wifi es originalmente una abreviación de la marca comercial *Wireless Fidelity*, que en inglés significa 'fidelidad sin cables o inalámbrica'. En este sentido, la tecnología wifi es una solución informática que comprende un conjunto de estándares para redes inalámbricas basados en las especificaciones IEEE 802.11, lo cual asegura la compatibilidad e interoperabilidad en los equipos certificados bajo esta denominación(13). Dentro de los estándares dados por la IEEE 802.11 se encuentran IEEE 802.11a, IEEE 802.11b, IEEE 802.11g, IEEE 802.11n e IEEE 802.11ac; recientemente se lanzó al mercado IEEE 802.11ad que permite mejor conexión además de una mayor transferencia de datos. Actualmente casi todos los dispositivos móviles y computadoras permiten utilizar esta tecnología de comunicación, siendo capaces de conectarse a un enrutador, a otro dispositivo móvil o siendo ellos mismos los puntos de acceso.

De esta manera se decide seleccionar como método de comunicación para su uso en la solución propuesta el protocolo **Wifi**, dada que su compatibilidad con los sistemas domóticos y puesto que representa un protocolo de comunicación confiable y seguro, gracias a que permite conexiones a largas distancias y una alta transferencia de datos.

1.5 Herramientas y tecnologías

Para la realización de este trabajo fue necesario apoyarse en disímiles herramientas y tecnologías, de las cuales serán expuestas a continuación una breve descripción.

1.5.1 Entorno de Desarrollo Integrado (IDE)

1.5.1.1 Apache Cordova

Apache Cordova es un marco de desarrollo móvil de código abierto. Permite utilizar las tecnologías estándar web como HTML5, CSS3 y JavaScript para desarrollo multiplataforma, evitando el lenguaje de desarrollo nativo para cada plataforma móvil. Las aplicaciones se ejecutan dentro de envolturas para cada plataforma y dependen de enlaces estándares API (*Application Programming Interface*) para acceder a cada dispositivo, sensores, datos y estado de la red. Cordova soporta diversas plataformas como *Android, Windows Phone*, *Blackberry* 10, *Amazon Fire OS*, *Firefox OS*, *Ubuntu*. Posee como

funciones principales: permitir crear una aplicación para varias plataformas, sin tener que volver a implementarlo con el lenguaje y herramienta de cada una. La implementación de una aplicación web que se convierte o envasa en una aplicación móvil. Se mezclan los componentes de la aplicación nativa con una *vista Web* (navegador) que puede tener acceso a las API de nivel de dispositivo, o si quiere desarrollar una interfaz *plugin* entre componentes *WebView* y nativos. Las aplicaciones se basan en un archivo común que proporciona información acerca de la aplicación y especifica los parámetros que afectan a cómo funciona. Este archivo se adhiere a la especificación de Empaquetado de la aplicación. La misma aplicación se implementa como una página web, que hace referencia a cualquier CSS, JavaScript, imágenes, archivos multimedia, u otros recursos son necesarios para que se ejecute. La aplicación se ejecuta como un *WebView* dentro de la envoltura de la aplicación nativa, que distribuye a tiendas de aplicaciones. De la aplicación web interactuar con varias características del dispositivo hacer las aplicaciones de forma nativas(14).

1.5.1.2 Sublime Text 3

Sublime Text es un editor de código multiplataforma, ligero y con pocas concesiones a los adornos. Es una herramienta concebida para programar sin distracciones. Su interfaz de color oscuro y la riqueza de coloreado de la sintaxis, centra la atención completamente. El programa dispone de auto-guardado, muchas opciones de personalización, cuenta con un buen número de herramientas para la edición del código y automatización de tareas. Soporta *macros*, *Snippets* y auto completar, entre otras funcionalidades. Algunas de sus características son ampliables mediante *plugins*(15).

1.5.2 Lenguaje de Programación

1.5.2.1 *HTML5*

HTML5 es la última versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance(16). HTML, que significa Lenguaje de Marcado para Hipertextos (*HyperText Markup Language*) es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad. HTML le da "valor

añadido" a un texto estándar en español. *HiperText* se refiere a enlaces que conectan una página Web con otra, ya sea dentro de una página web o entre diferentes sitios web(17).

1.5.2.2 CCS 3

CSS3 es la última evolución del lenguaje de las Hojas de Estilo en Cascada, y pretende ampliar la versión CSS2.1. Trae consigo muchas novedades altamente esperadas, como las esquinas redondeadas, sombras, gradientes, transiciones o animaciones, y nuevos *layouts* como multi-columnas, cajas flexibles o maquetas de diseño en cuadrícula(18). Hojas de Estilo en Cascada (*Cascading Style Sheets*) es el lenguaje utilizado para describir la presentación de documentos HTML o XML, esto incluye varios lenguajes basados en XML como son XHTML o SVG. CSS describe como debe ser renderizado el elemento estructurado en pantalla, en papel, hablado o en otros medios(19).

1.5.2.3 *JavaScript*

JavaScript es el lenguaje interpretado orientado a objetos desarrollado por Netscape que se utiliza en millones de páginas web y aplicaciones de servidor en todo el mundo. Es un lenguaje de programación dinámico que soporta construcción de objetos basado en prototipos. La sintaxis básica es similar a Java y C++ con la intención de reducir el número de nuevos conceptos necesarios para aprender el lenguaje. Las capacidades dinámicas de *JavaScript* incluyen construcción de objetos en tiempo de ejecución, listas variables de parámetros, variables que pueden contener funciones, creación de scripts dinámicos(20).

1.5.3 *Framework*

1.5.3.1 *Bootstrap*

Bootstrap es un conjunto de conceptos, prácticas y criterios (*framework*) desarrollado por Mark Otto y Jacoob Thornton dentro de *Twitter* con la intención de estandarizar las herramientas que utilizaban los programadores de *front-end*. Bootstrap es el *framework* más popular para desarrollar proyectos móviles de primera respuesta en la Web utilizando *HTML*, *CSS* y JavaScript. Bootstrap hace que el desarrollo web *front-end* sea más rápido y más fácil. Está hecho para desarrolladores de todos los niveles de habilidad y proyectos de todos los tamaños. Posee una extensa documentación para elementos HTML comunes, docenas de componentes HTML y CSS personalizados, y complementos jQuery(21).

1.5.3.2 JSON

JSON (*JavaScript Object Notation* - Notación de Objetos de *JavaScript*) es un formato ligero de intercambio de datos. Está basado en un subconjunto del Lenguaje de Programación JavaScript, es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, *Java*, *JavaScript*, *Perl*, *Python*, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. JSON está constituido por dos estructuras: Una colección de pares de nombre/valor y una lista ordenada de valores. Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras. (22)

1.5.3.3 *AngularJS*

AngularJS es un *framework* cliente desarrollado por Google, al ser un *framework* con un gran auge en el mundo *web* cuenta con un buen set de herramientas y *plugins* ya desarrollados, así como también con buenos manuales y recomendaciones de buenas prácticas, los cuales pueden ayudar a crear un proyecto sostenible. *AngularJS* permite extender el vocabulario HTML con directivas y atributos, manteniendo la semántica y sin necesidad de emplear librerías externas como *jQuery* o *Underscore.js* para que funcione. Cuenta con un set de utilidades, opciones y configuraciones medianamente amplio. *Angular* está desarrollado en base a módulos y cuenta con su propio sistema de inyección de dependencias. (23)

Angular promueve y usa patrones de diseño de software como Modelo-Vista-Controlador. Básicamente estos patrones nos marcan la separación del código de los programas dependiendo de su responsabilidad. Posee mejoras del *HTML*, viene cargado con todas las herramientas que los creadores ofrecen para que los desarrolladores sean capaces de crear ese *HTML* enriquecido. La palabra clave que permite ese *HTML* declarativo en *AngularJS* es "directiva", que no es otra cosa que código *JavaScript* que mejora el HTML. (24)

1.5.4 Herramienta CASE

CASE (*Cumputer Aided Software Engineering*) es la aplicación de las tecnologías informáticas a las actividades, técnicas y metodologías propias del desarrollo de sistemas,

en términos generales una Herramienta CASE es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, ya sea parcial o completamente, se puede ver al CASE como el agrupamiento de las herramientas de software y las metodologías de desarrollo de software formales(25) .**Visual Paradigm** es una potente herramienta de diseño y gestión de plataforma multiplataforma para sistemas informáticos. **Visual Paradigm** proporciona a los desarrolladores de software una plataforma de desarrollo de vanguardia para crear aplicaciones de calidad. Facilita una excelente interoperabilidad con otras herramientas CASE y con la mayoría de los principales IDEs que superan todo el proceso de desarrollo de *Model-Code-Deploy*(26).

1.5.5 Lenguaje de Modelado

Un lenguaje de modelado puede estar compuesto de pseudocódigo, código real, imágenes, diagramas o largos pasajes de descripción, de hecho, es prácticamente cualquier cosa que pueda ayudar a describir un sistema(27). Los lenguajes de modelado poseen como objetivo principal visualizar de manera gráfica el sistema que se desarrollara. Estos pueden utilizarse a la hora de la comunicación con el o los clientes, grupo de desarrollo y otros factores que intervengan en este proceso. Durante el desarrollo de este trabajo se utilizará el Lenguaje Unificado de Modelado (UML por sus siglas en ingles), es uno de los lenguajes de modelado de sistemas de software más conocido y utilizado en la actualidad respaldado por el Grupo de Administración de Objetos (OMG, por sus siglas en ingles). Proporciona ventajas en la representación del ciclo de vida de un software y de los artefactos específicos del Proceso Unificado de Desarrollo del Software(28).

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones en:

- **Visualizar**: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar**: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir**: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar**: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión(29).

1.5.5 Metodología

Diferentes proyectos necesitan diferentes procesos. Los factores típicos nos indican la necesidad de procesos formales o agiles, como el tamaño del equipo y donde van a estar, la complejidad de la arquitectura, las innovaciones tecnológicas o los estándares de conformidad.

AUP –UCI

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. (30)

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes fases de AUP en una sola, la que lleva el nombre de Ejecución y se agrega la fase de Cierre. (31), como se muestra en la Tabla 1.

Tabla 1 Comparación entre las fases de AUP y AUP-UCI.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Dado lo antes expuesto se puede apreciar que la metodología AUP-UCI es la más adecuada para aplicar en este proyecto, esta metodología al ser ágil se complementa perfectamente con la solución a desarrollar ya que esta soporta cambios durante el desarrollo de la propuesta, su utilización es adecuada para pequeños grupos de trabajo y proyectos de poca duración, además de ser la utilizada para el proceso de desarrollo de *software* en la UCI.

Conclusiones parciales

- Los conceptos expuestos permitieron comprender el campo relacionado con la domótica y su uso en aplicaciones móviles para el hogar.

- Se realizó un estudio de las principales herramientas y metodologías a utilizar, presentándose sus características y ventajas, las cuales apoyan el correcto desarrollo de la solución.

Capítulo 2 Análisis y diseño

Introducción

Este capítulo tiene como objetivo describir los procesos de análisis y diseño. Se presentan los requisitos funcionales y no funcionales por los que se regirá el desarrollo de la solución propuesta, se determinan las historias de usuarios, se muestran los diagramas que describen el funcionamiento del sistema, se diseña y se describe una arquitectura y se exponen los patrones utilizados para el desarrollo de la solución propuesta.

2.1 Modelo de Dominio

El Modelo de Dominio es un artefacto de la disciplina de análisis que puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de *software* o de otro tipo. En la imagen 13 se muestra la relación entre las principales entidades del sistema,

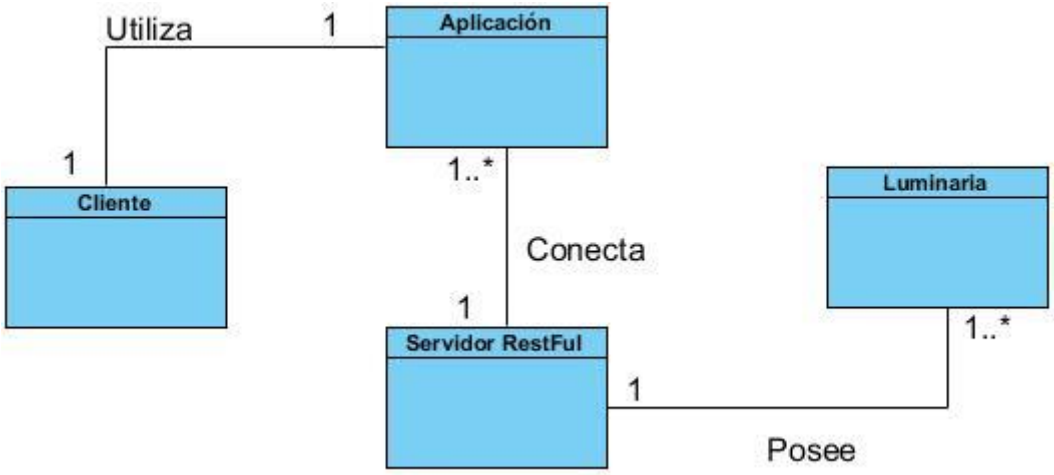


Ilustración 13 Modelo de dominio.

2.2 Requisitos del sistema

Entre los artefactos que define la metodología seleccionada se encuentran las historias de usuario, las cuales representan una breve descripción del comportamiento del sistema, emplean terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. (28)

2.2.1 Requisitos funcionales

RF1.El sistema debe permitir agregar luminarias al sistema.

RF2. El sistema debe permitir eliminar luminarias del sistema.

RF3. El sistema debe permitir encender las luminarias del sistema.

RF4. El sistema debe permitir apagar las luminarias del sistema.

RF5. El sistema debe permitir encender todas las luminarias del sistema.

RF6. El sistema debe permitir apagar todas las luminarias del sistema.

RF7. El sistema debe permitir la conexión con el servidor RestFul.

2.2.2 Requisitos no funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

Requisitos no funcionales

Software

RNF.1: La aplicación requiere que el móvil cuente con SO iOs 6 en caso *iPhone*, *Android* 4 o superiores respectivamente.

Hardware

RNF.2: El dispositivo debe permitir la conexión Wifi.

RNF.3: La aplicación requiere de una capacidad disponible de 8 MB en la memoria de almacenamiento del dispositivo para la instalación y ejecución de la misma.

2.3 Historias de usuario

Las historias de usuarios son la propuesta de las metodologías ágiles para la especificación de los requisitos de cliente. Se escriben desde el punto de vista del usuario del sistema y usando su vocabulario. (32)

Tabla 2 HU #1

Historia de Usuario	
Número: 1	**Nombre de Historia:** Implementar la funcionalidad Adicionar Luminaria
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimados:** 6 días
Nivel de Complejidad: Media	**Tiempo real:** 1 semana
Descripción: La aplicación permitirá al usuario agregar una luminaria al sistema mediante el botón Configuración y seleccionándola de la lista de las luminarias en existencia en el sistema.	

Tabla 3 HU #2

Historia de Usuario	
Número: 2	**Nombre de Historia:** Implementar la funcionalidad Eliminar Luminaria
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimados:** 6 días
Nivel de Complejidad: Media	**Tiempo real:** 1 semana
Descripción: La aplicación permitirá al usuario eliminar una luminaria del sistema mediante el botón Configuración y seleccionándola de la lista de las luminarias en existencia en el sistema.	

Tabla 4 HU#3

Historia de Usuario

Número: 3	**Nombre de Historia:** Implementar la funcionalidad Encender Luminaria
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimados:** 8 días
Nivel de Complejidad: Alta	**Tiempo real:** 2 semana
Descripción: La aplicación permitirá al usuario encender las luminarias del sistema mediante un botón Switch, realizando una petición al servidor Rest.	

Tabla 5 HU#4

Historia de Usuario	
Número: 4	**Nombre de Historia:** Implementar la funcionalidad Apagar Luminaria.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimados:** 8 días
Nivel de Complejidad: Alta	**Tiempo real:** 2 semana
Descripción: La aplicación permitirá al usuario apagar las luminarias del sistema mediante un botón Switch, realizando una petición al servidor Rest.	

Tabla 6 HU#5

Historia de Usuario

Número: 5	**Nombre de Historia:** Implementar la funcionalidad Encender todas las Luminarias.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimados:** 8 días
Nivel de Complejidad: Alta	**Tiempo real:** 2 semana
Descripción: La aplicación permitirá al usuario encender las luminarias del sistema mediante un botón Switch global, realizando una petición al servidor Rest.	

Tabla 7 HU#6

Historia de Usuario	
Número: 6	**Nombre de Historia:** Implementar la funcionalidad Apagar todas las Luminarias.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimados:** 8 días
Nivel de Complejidad: Alta	**Tiempo real:** 2 semana
Descripción: La aplicación permitirá al usuario apagar las luminarias del sistema mediante un botón Switch global, realizando una petición al servidor Rest.	

Tabla 8 HU#7

Historia de Usuario

Número: 7	**Nombre de Historia:** Implementar la conexión al servidor RestFul.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimados:** 12 días
Nivel de Complejidad: Alta	**Tiempo real:** 2 semana
Descripción: La aplicación deberá conectarse al servidor Rest, de forma inalámbrica mediante el protocolo IEEE 802.11 (WIFI), para el consumo de servicios.	

2.4 Propuesta de sistema

Realizado el estudio de las tecnologías y una vez seleccionadas, se dispone a diseñar la propuesta de solución.

2.4.1 Construcción del proyecto

La herramienta de línea de comandos de Cordova se distribuye como un paquete de npm en un formato listo para usar. No es necesario compilarlo desde su código fuente(33).

Para instalar la herramienta de línea de comandos de cordova, se siguen los siguientes pasos:

- Descargar e instalar *Node.js*.
- Descargar e instalar un cliente de *git*, aunque no utilices *git* manualmente, la CLI usa de fondo para descargar algunos archivos cuando se crea un nuevo proyecto.
- Instalar el módulo *cordova* utilizando el manejador de paquetes de *Node.js npm*. El módulo *cordova* será descargado automáticamente por *npm*.
- Se ejecuta el comando: $ sudo npm install -g cordova.

2.4.2 Crear la aplicación

Ubicándose en la dirección donde aparecerá el código fuente de la aplicación se ejecuta el siguiente código: $ cordova *create hello com.example.hello HelloWorld*

Esta línea genera un proyecto con las carpetas *HTML, CCS y JS* donde se guardara la configuración de la aplicación.

2.4.3 Añadir plataformas

El desarrollo de la aplicación en Apache Cordova proporciona la alternativa de ser multiplataforma, pero si se desea compilar para un dispositivo específico se siguen los siguientes pasos(34):

- Ejecutar el comando siguiente para acceder a la ruta del proyecto: $ cd hello
- Se agregan los SDK para la plataforma deseada como se muestra a continuación para iOS: $ cordova platform add ios, y para *Android* se instala de forma manual.

2.4.4 Construir la aplicación

En la fase final del proceso se pasa a construir la aplicación para el dispositivo.

Ejecutar el siguiente comando para crear el proyecto:

$ cordova build

Esto genera un código específico de plataforma dentro del subdirectorio del proyecto *platforms*. Opcionalmente puede limitar el alcance de cada *build* a plataformas específicas:

$ cordova build ios

$ cordova build android

2.4.5 Instalación en el dispositivo móvil

En el caso de *Android* se copia la aplicación al dispositivo móvil y se instala desde el gestor de archivos.

En el caso de iOS es necesario un Certificado de Desarrollador de Apple para poder instalar la aplicación en cualquier dispositivo móvil *iPhone*, el cual debe ser pagado mediante tarjeta de crédito. De aquí que se muestre la solución desde el simulador de Xcode.

Para las aplicaciones iOs se utiliza el IDE Xcode, entorno creado por Apple para el desarrollo de sus programas y aplicaciones, permite realizar trabajos para iPhone, iPad y demás dispositivos de la propia marca. Xcode es un entorno de desarrollo que en su versión 7 ya es reconocido internacionalmente por ser confiable y potente, con grandes prestaciones en la implementación, realización de pruebas y depuración, permitiendo detectar el daño de memoria antes de que suceda. Con Xcode 7 se permite la programación en Objetive-C y Swift. Este último es un lenguaje de programación relativamente nuevo y que según la propia marca Apple será el lenguaje universal para aplicaciones móviles(35).

Como se expuso en el capítulo anterior se desarrollara una aplicación móvil multiplataforma para el control de luminarias en un sistema domótico. Las aplicaciones domóticas poseen una interfaz sencilla y amigable, mediante la cual los usuarios puedan interactuar con el sistema de forma rápida, por esto la solución a desarrollar deberá contar con una vista principal y una vista de configuración:

- **Vista principal:** Esta vista es la encargada de mostrar la lista de las luminarias del sistema, de las cuales se visualizarán un nombre, un botón Switch de encendido/apagado y un botón de Encender todo. En la parte inferior de la pantalla aparecerá el botón Configuración.

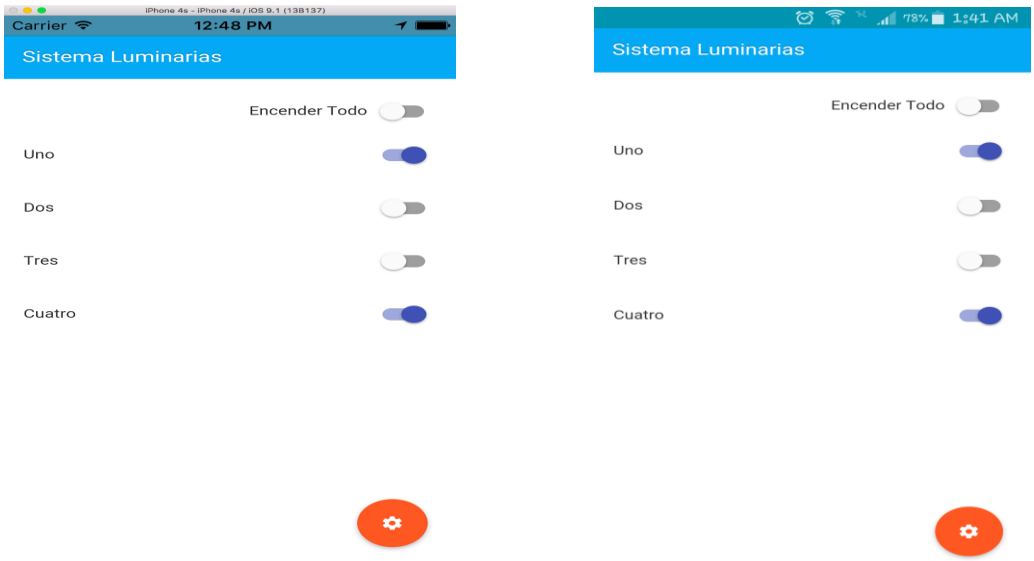


Ilustración 9 Vista Principal

- **Vista Configuración:** En esta vista presentará una modal con las opciones editar el número de la luminarias que se mostraran en el sistema en base a las que contenga el servidor en su base de datos.

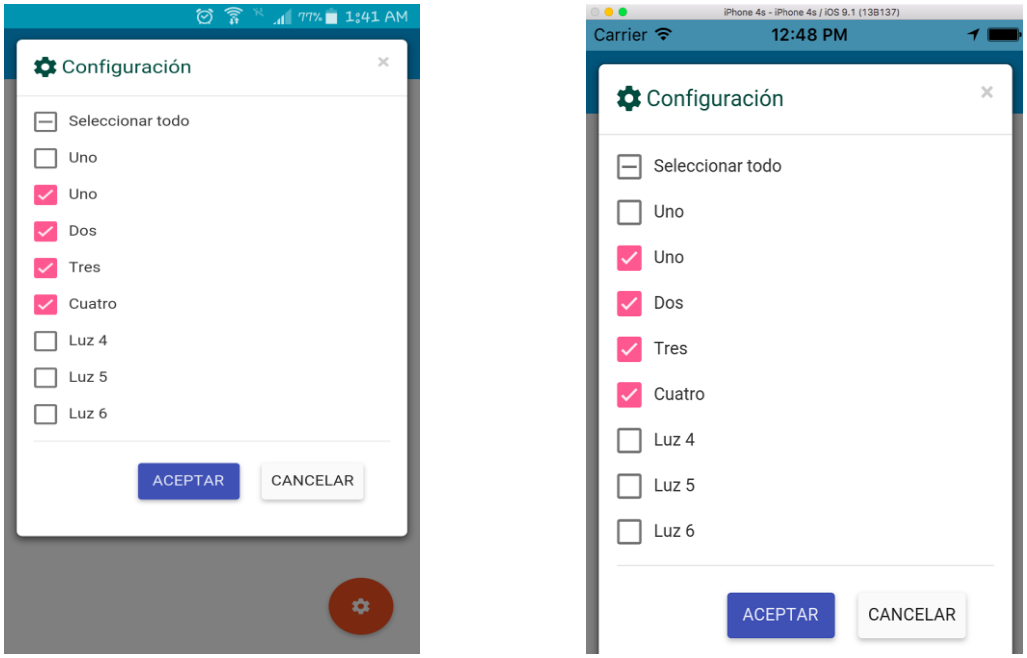


Ilustración 10 Vista Configuración

Una vez conectada la aplicación con el servidor el resultado esperado es un sistema de control de luminarias, como se muestra en la Ilustración 11.

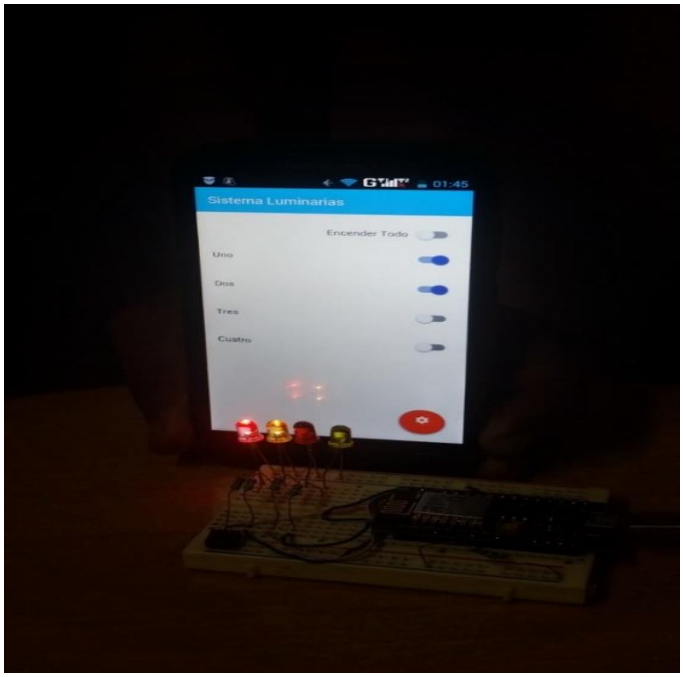


Ilustración 11 Despliegue de aplicación

Como se muestra en la Ilustración 12, el dispositivo móvil cuenta con la aplicación desarrollada, se conecta al servidor por medio de la conexión **IEEE 802.11 (Wifi)** para obtener la configuración del sistema, una vez obtenida la configuración se mostrará en la pantalla principal las luminarias que el servidor cuenta como habilitadas, dicho servidor se encuentra instalado en la tarjeta **Hiletgo**, el usuario puede agregar o eliminar luminarias de la vista principal, mostrando solo las deseadas, luego de consumir dichas peticiones desde el dispositivo móvil, el servidor enviará las órdenes del usuario a las luminarias desplegadas.

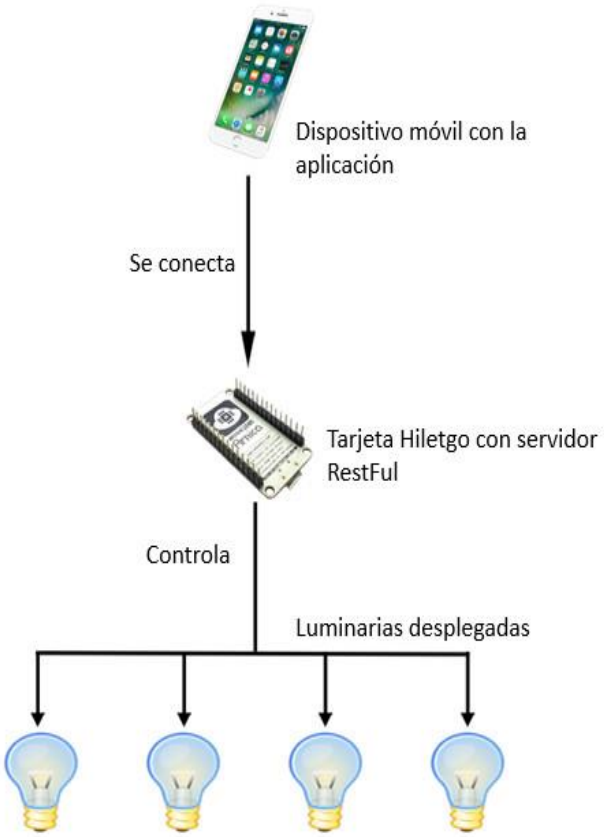


Ilustración 11 Propuesta de solución

2.5 Diagrama de Paquetes

En los Diagramas de Paquetes, un paquete es un mecanismo utilizado para agrupar elementos de UML. Un paquete es una parte de un modelo, contiene elementos del modelo al más alto nivel, tales como clases y sus relaciones, máquinas de estado, diagramas de casos de uso, interacciones y colaboraciones: cualquier elemento que no esté contenido en

otro. Los paquetes pueden contener otros paquetes. Las dependencias entre paquetes resumen dependencias entre los elementos internos a ellos, es decir, las dependencias del paquete se derivan a partir de las dependencias entre los elementos individuales(36).

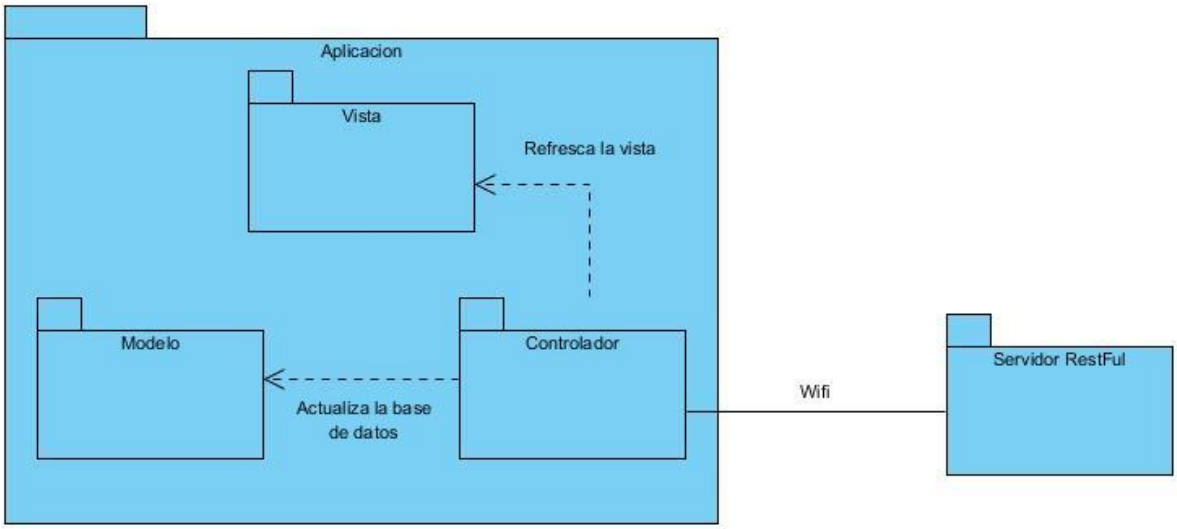


Ilustración 12 Diagrama de Paquete

Como se muestra en la imagen, existen dos paquetes principales, el paquete Aplicación y el paquete Servidor, en el caso de aplicación cuenta con otros 3 paquetes: Modelo, Vista, Controlador, este último realiza las peticiones al Servidor RestFul y luego actualiza en el modelo los cambios realizados, posteriormente refresca la vista para que el usuario verifique que se hayan realizado las operaciones seleccionadas.

2.6 Arquitectura del sistema

La arquitectura de software es la organización fundamental de un sistema, encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución(37). Es una vista estructural de alto nivel que define un estilo o una combinación de estilos para una solución. Un estilo arquitectónico proporciona una plantilla de trabajo que unifica la manera en que todos los miembros del equipo ven el sistema e impone una transformación al diseño del mismo(38).

El sistema está diseñado utilizando el modelo arquitectónico Modelo-Vista-Controlador, como se aprecia en la Ilustración 15, el cual basa su fundamento en la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores:

Ilustración 16 Modelo-Vista-Controlador

- **Modelo:** Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas(39).
- **Vista:** Las vistas, como su nombre nos hace entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que nos permitirá renderizar los estados de nuestra aplicación. En las vistas nada más tenemos los códigos que nos permite mostrar la salida(39).

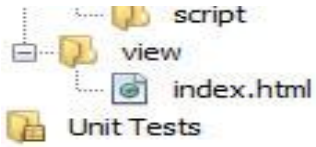


Ilustración 13 Vista

- **Controlador:** Contiene el código necesario para responder a las acciones que se solicitan en la aplicación. Es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de la aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo(39).

Ilustración 14 Controlador

Como se mostró en las ilustraciones 17 y 18, la capa Vista se representa en la clase *index.html*, conteniendo esta, el código para mostrar los datos al usuario, la capa Controlador se evidencia en las clases *app.js* y *ctrl.js*, siendo estas las encargadas de implementar las funcionalidades para que la aplicación consuma los servicios en base a la acción realizada por el cliente, y en el caso de la capa Modelo no se representa en ninguna clase dado que los datos se obtienen del servidor, cargándose la configuración que este contiene.

2.7 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces(40). **Existen tres tipos de patrones de diseño:**

- **Patrones creacionales**: Utilizados para instanciar objetos, y así separar la implementación del cliente de la de los objetos que se utilizan. Con ellos intentamos separar la lógica de creación de objetos y encapsularla(40).
- **Patrones de comportamiento**: Se utilizan a la hora de definir como las clases y objetos interaccionan entre ellos(40).
- **Patrones estructurales**: Utilizados para crear clases u objetos que incluidos dentro de estructuras más complejas(40).

2.7.1 Patrones de diseño GOF

***Singleton*:** Es un patrón específico que utiliza *AngularJS*, se pone de manifiesto en la página principal *index.html*, esta encapsula la vista principal, por lo que se puede describir como una instancia única y que posee un acceso global a ella.

Ilustración 19 Patrón Singleton

***Observer*:** La entidad ctrl.js está observando los cambios realizados en la base de datos para actualizarlos luego en la interfaz. En acciones como la gestión de luminarias, los cambios se guarda en el servidor, y a su vez en la interfaz index.html, como estos cambios se deben mostrar, el script ctrl.js se encarga de observar si hay algún cambio en este servidor y pedir los nuevos datos para actualizarlos en la interfaz asignada.

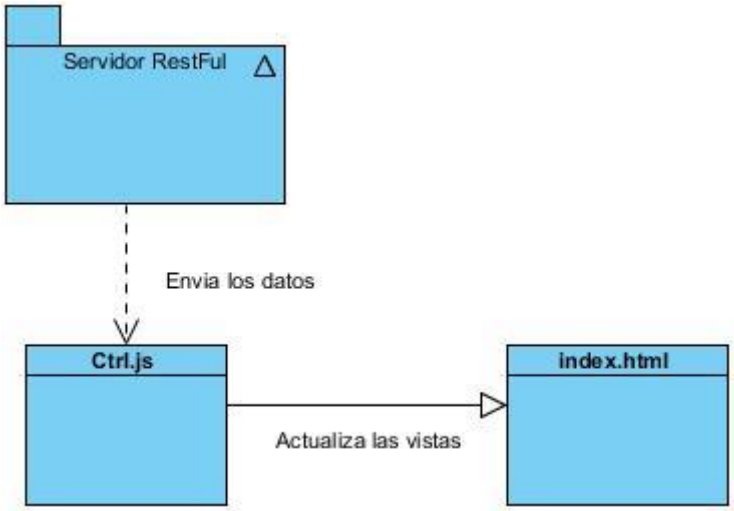


Ilustración 20 Patrón Observer

2.8.2 Patrones de diseño GRASP

Controlador: Este patrón se pone de manifiesto en el módulo ctrl.js, siendo este el encargado de manejar los eventos relacionados al funcionamiento de la aplicación, se evidencia este patrón de manera más clara en la arquitectura MVC utilizada para el desarrollo de software.

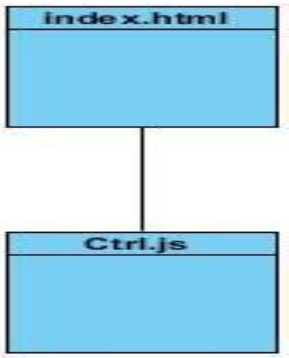


Ilustración 19 Patrón Controlador

Conclusiones parciales

- Se definieron los requisitos funcionales y no funcionales, necesarios para el correcto desarrollo de la solución.
- La realizaron historias de usuario y el modelado de dominio permitió un mayor esclarecimiento de las funcionalidades del sistema.
- La selección del modelo arquitectónico y los patrones de diseño a utilizar en el proyecto, ayudaron a una correcta planificación de la implementación de la solución.

Capítulo 3 Implementación y pruebas

Introducción

Terminado el proceso de diseño, este capítulo abordará el proceso de implementación y pruebas. Se realizaran las pruebas necesarias para determinar que la solución cumple los requisitos. Se expondrá el estándar de codificación utilizado para el desarrollo de la solución.

3.1 Estándar de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, establezca un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente(41).

A continuación, se muestra las técnicas utilizadas en este sistema:

- Ninguna función debe tener más de 200 líneas.
- Los nombres de las variables son cortos y significativos.

  var temp = angular.copy(luminaria);

- Las variables utilizan la nomenclatura *CamelCase*, comienzan con letra minúscula y cada palabra consecutiva en el nombre comienza con letra mayúscula.
- Las funciones utilizan la nomenclatura *lowerCamelCase* (con letra mayúscula y cada palabra consecutiva en el nombre comienza con letra mayúscula).

  onOffAll: function ()

- Las carpetas y archivos del sistema están escritos en inglés.

El código de un programa lo leen muchas personas, bien para repararlo, bien para ampliarlo o, simplemente, para evaluarlo. Para estos lectores es fundamental que el programa esté

bien redactado, con estilo, para que su significado sea claro y diáfano. Aun reconociendo que redactar con buen estilo es más un arte que una ciencia, sí existen una serie de reglas básicas que ayudan a conseguir un texto satisfactorio. Al igual que cuando aprendemos redacción literaria, una forma excelente de adquirir un buen estilo es leer textos ajenos. Que un programa funcione o no es en buena medida independiente de que esté bien o mal escrito(42).

Un programador dispone de cuatro mecanismos básicos para comunicarse con sus lectores:

- El estilo.
- Los comentarios.
- Los nombres de las variables, constantes y métodos.
- Los espacios en blanco (el sangrado).

Reglas del estilo de escritura:

- Documentación (javadoc) de la clase o interfaz.
- *Class* o *interface.*
- Variables de clase (*static*).
- Variables de instancia u objeto.
- Constructores.
- Demás métodos.

Reglas sobre las variables:

- Las variables de clase u objeto jamás deberían ser públicas.
- Las variables deben ver reducida su existencia al mínimo tiempo posible.
- Las variables deben inicializarse inmediatamente.
- Jamás use la misma variable para retener dos cosas diferentes.
- Las variables de iteración deben tener nombres muy cortos: i, j, k, etc.

Reglas sobre las expresiones:

- Evitar expresiones complejas, considere la creación de variables auxiliares o métodos auxiliares.
- En una construcción condicional, ponga el caso normal en la parte *if* y el caso excepcional en la parte *else*.

- Evite comparar con TRUE o FALSE para tomar decisiones.

3.2 Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. Un Nodo es un elemento de hardware o software. Esto se muestra con la forma de una caja en tres dimensiones, como a continuación(43).

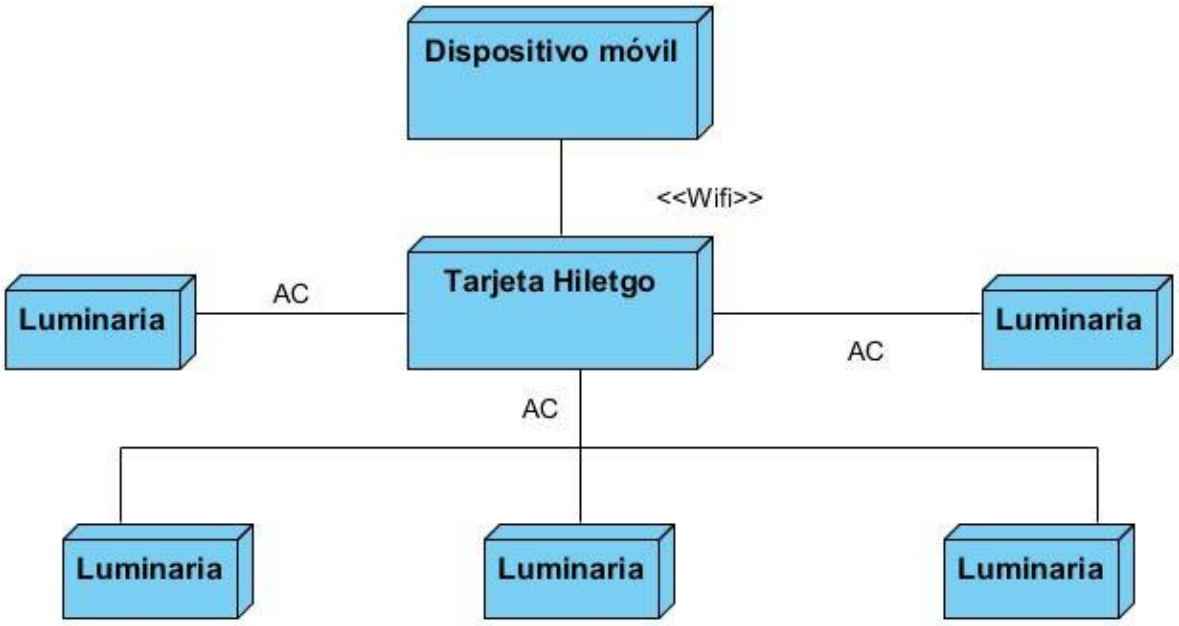


Ilustración 21 Diagrama de despliegue

- **Dispositivo móvil:** Este nodo hace referencia al dispositivo móvil que tiene instalado la aplicación, este realiza una conexión al servidor RestFul mediante el protocolo IEEE 802.11 (Wifi) para el consumo de servicios.
- **Tarjeta Hiletgo:** El nodo servidor representa el servidor RestFul al cual se conecta de forma inalámbrica mediante el protocolo IEEE 802.11 (Wifi), el dispositivo móvil, y mediante conexión eléctrica a las luminarias.
- **Luminaria:** Los nodos en cuestión hace referencia a los LEDs o luminarias desplegadas en el recinto, las cuales realizarán la acción final. Estas se conectan por conexión eléctrica al servidor RestFul.

3.3 Pruebas de software

Las pruebas de sistema tienen como objetivo examinar profundamente el producto, comprobándolo de forma global. Esto posibilita alcanzar una visión similar a su comportamiento en el entorno de producción.

3.3.1 Tipos de prueba

Pruebas de estrés

Pruebas para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones. Son pruebas de carga o rendimiento, pero superando los límites esperados en el ambiente de producción o determinados en las pruebas. Ejemplo: encontrar la cantidad de usuarios simultáneos, en que la aplicación deja de responder (cuelgue o time out) en forma correcta a todas las peticiones(44).

Se realizaron acciones de encendido y apagado simultáneos de las luminarias en el sistema, en algunos casos se utilizó la funcionalidad Encender Todos, se detectaron problemas en el encendido y que algunas luminarias se quedaban encendidas luego de haberse apagado. Todas las no conformidades se solucionaron.

Pruebas de carga

Pruebas para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios o transacciones que se espera en el ambiente de producción. Ejemplo: verificar la correcta respuesta de la aplicación ante el alta de 100 usuarios en forma simultánea. Se compara con el volumen esperado(44).

Se ejecutaron peticiones al servidor, agregando y eliminando luminarias del sistema en pocos intervalos de tiempo, generando esto que la aplicación se cerrara, a dicha no conformidad se le dio solución.

Una vez descritas las pruebas anteriores se llega a la conclusión que las pruebas de aceptación son las indicadas para evaluar si el software cumple los requisitos de aceptación marcados por el cliente.

3.4 Ambientes de pruebas

Las pruebas se realizaron sobre los dispositivos que se listan a continuación:

Para iOS

- Una PC con el IDE Xcode y Apache Cordova.
- Una tarjeta *Hiletgo* con el servidor RestFul.

Para *Android*

- Una PC con el IDE Visual Estudio y Apache Cordova.
- Una tarjeta *Hiletgo* con el servidor RestFul.
- Un dispositivo móvil con sistema operativo Android y conexión Wifi.

3.5 Diseño de casos de pruebas

Para la realización de los casos de pruebas se utilizó el ambiente anteriormente descrito, haciéndose hincapié en la conexión entre las aplicaciones instaladas en la PC con el simulador del IDE *Xcode* y el dispositivo móvil *iPhone* 4s con la tarjeta Hiletgo donde se encuentra el servidor RestFul.

Tabla 9 CP#1

Caso de Prueba Aceptación	
Numero: 1	**Historia de usuario: 1**
Nombre: Implementar la funcionalidad Adicionar Luminaria.	
Descripción: La aplicación permitirá al usuario adicionar una luminaria en el sistema mediante un formulario.	
Condiciones de Ejecución: El usuario deberá comprobar que se cumple la funcionalidad Adicionar Luminaria.	
Entradas/ Pasos de Ejecución: - Se seleccionará el botón Configuración en la parte inferior derecha de la pantalla. - Luego aparecerá un formulario para seleccionar las luminarias que se desean agregar. - Se seleccionará la opción “Aceptar”.	

Resultado esperado: La luminaria ha sido añadida al sistema.
Evaluación de la prueba: Una vez realizada la primera iteración se detectan varias no conformidades, a las cuales se les da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.

Tabla 10 CP#2

Caso de Prueba Aceptación	
Número: 2	**Historia de usuario: 2**
Nombre: Implementar la funcionalidad Eliminar Luminaria	
Descripción: La aplicación permita al usuario eliminar una luminaria del sistema mediante un formulario.	
Condiciones de Ejecución: El usuario debe comprobar que se cumple la funcionalidad eliminar luminaria.	
Entradas/ Pasos de Ejecución: - Se seleccionará el botón Configuración en la parte inferior derecha de la pantalla. - Luego aparecerá un formulario para seleccionar las luminarias que se desean eliminar. - Se seleccionará la opción “Aceptar”.	
Resultado esperado: La funcionalidad eliminar luminaria es correcta.	
Evaluación de la prueba: Una vez realizada la primera iteración se detectan varias no conformidades, a las cuales se les da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.	

Tabla 11 CP#3

Caso de Prueba Aceptación

Número: 3	**Historia de usuario: 3**
Nombre: Implementar la funcionalidad Encender Luminaria	
Descripción: La aplicación permitirá al usuario encender la luminaria deseada mediante el botón encendido/apagado.	
Condiciones de Ejecución: El usuario deberá comprobar que se cumple la funcionalidad Encender Luminaria.	
Entradas/ Pasos de Ejecución: - Se seleccionará la vista principal que contiene la lista de las luminarias - Se deslizará hacia la derecha el botón encendido/apagado de la luminaria que se desea encender.	
Resultado esperado: La funcionalidad Encender Luminaria se realizó satisfactoriamente.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 12 CP#4

Caso de Prueba Aceptación	
Número: 4	**Historia de usuario: 4**
Nombre: Implementar la funcionalidad Apagar Luminaria	
Descripción: La aplicación permitirá al usuario apagar la luminaria deseada mediante el botón encendido/apagado.	
Condiciones de Ejecución: El usuario deberá comprobar que se cumple la funcionalidad Apagar Luminaria.	
Entradas/ Pasos de Ejecución: - Se seleccionará la vista principal que contiene la lista de las luminarias	

- Se deslizará hacia la izquierda el botón de encendido/apagado de la luminaria que se desea encender.
Resultado esperado: La funcionalidad Apagar Luminaria se realizó satisfactoriamente.
Evaluación de la prueba: Prueba Satisfactoria.

Tabla 13 CP#5

Caso de Prueba Aceptación	
Número: 5	**Historia de usuario: 5**
Nombre: Implementar la funcionalidad Encender todas las Luminarias.	
Descripción: La aplicación permitirá al usuario encender todas las luminarias deseadas mediante el botón encendido/apagado global.	
Condiciones de Ejecución: El usuario deberá comprobar que se cumple la funcionalidad Encender todas las Luminarias.	
Entradas/ Pasos de Ejecución: - Se seleccionará la vista principal que contiene la lista de las luminarias - Se deslizará hacia la derecha el botón encendido/apagado global en la parte superior derecha de la pantalla.	
Resultado esperado: La funcionalidad Encender todas las Luminarias se realizó satisfactoriamente.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 14 CP#6

Caso de Prueba Aceptación	
Número: 6	**Historia de usuario: 6**

Nombre: Implementar la funcionalidad Apagar las Luminaria
Descripción: La aplicación permitirá al usuario apagar la luminaria deseada mediante el botón encendido/apagado global.
Condiciones de Ejecución: El usuario deberá comprobar que se cumple la funcionalidad Apagar todas las Luminarias.
Entradas/ Pasos de Ejecución: - Se seleccionará la vista principal que contiene la lista de las luminarias - Se deslizará hacia la izquierda el botón encendido/apagado global en la parte superior derecha de la pantalla.
Resultado esperado: La funcionalidad Apagar todas las Luminarias se realizó satisfactoriamente.
Evaluación de la prueba: Prueba Satisfactoria.

Tabla 15 CP#7

Caso de Prueba Aceptación	
Número: 7	**Historia de usuario: 7**
Nombre: Implementar la conexión con el servidor Restful	
Descripción: Comprobar que la aplicación se conecta al servidor Restful.	
Condiciones de Ejecución: El usuario deberá comprobar la conexión con el servidor.	
Entradas/ Pasos de Ejecución: - Se encenderá la conexión wifi del dispositivo. - Seleccionará la conexión enviada por el servidor. - La aplicación se conectará automáticamente al servidor Restful.	
Resultado esperado: La aplicación se conecta satisfactoriamente al servidor.	

Evaluación de la prueba: Una vez realizada la primera iteración detectaron varias no conformidades en la conexión con el servidor, a las cuales se les da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.

3.6 Ejecución de los casos de pruebas de aceptación

Las pruebas de aceptación fueron realizadas en un conjunto de iteraciones; cuando las pruebas arrojan no conformidades se corrigen los errores obtenidos. Después de corregidos, se vuelve a realizar la prueba de aceptación en una segunda iteración de pruebas a estas funcionalidades que presentaron no conformidades. El resultado de estas pruebas por iteración se muestra en el siguiente gráfico:

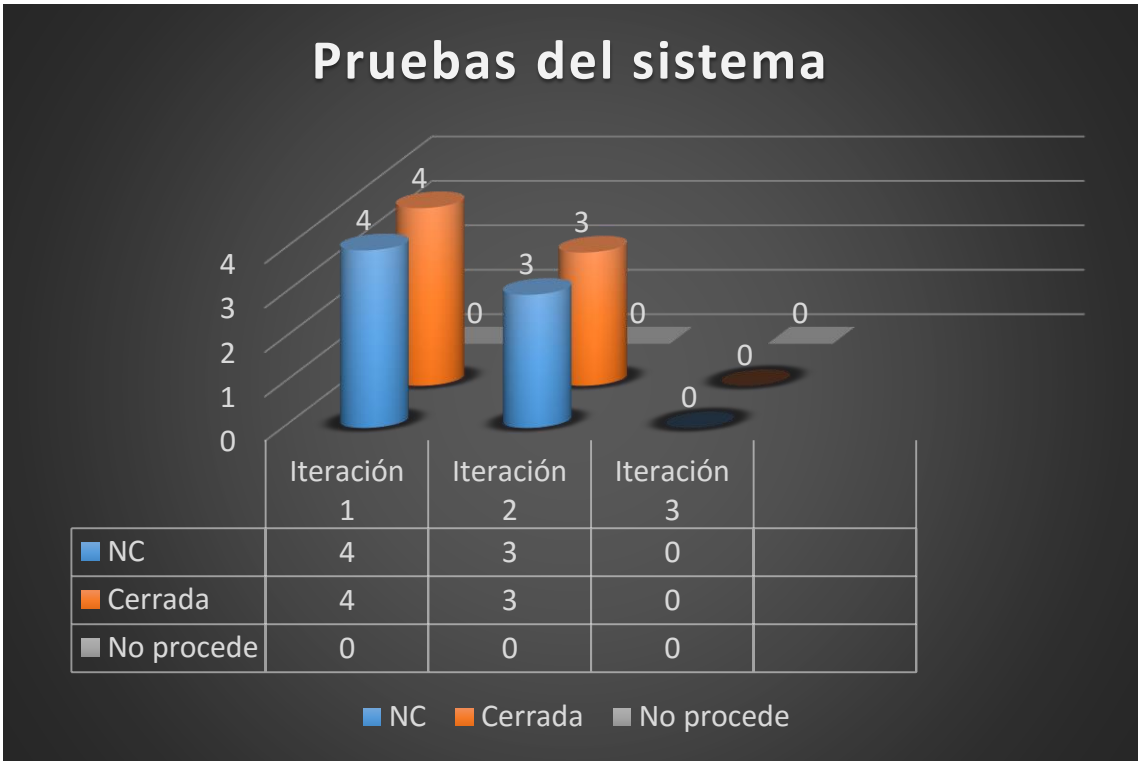


Ilustración 22 Gráfica de pruebas de aceptación

Conclusiones Parciales

- La definición de un estándar de codificación permitió una mejor gestión del código utilizado en la solución.
- Mediante las pruebas ejecutadas al sistema, se lograron encontrar errores funcionales en el sistema.

Conclusiones generales

Con la realización de la presente investigación se desarrolló una aplicación móvil multiplataforma para el control luminarias en un sistema domótico. De esta forma se da cumplimiento al objetivo planteado al inicio de la misma, además:

- El estudio de los principales conceptos relacionados con el tema de la domótica, aportó conocimientos necesarios para el desarrollo de la aplicación propuesta.
- La correcta selección de la metodología y herramientas, permitió identificar las funcionalidades de la aplicación y desarrollarlas.
- Se realizaron las pruebas necesarias, las cuales permitieron.
- Con la implementación de la solución se pudo eliminar los problemas de compatibilidad con todos los dispositivos móviles.

Recomendaciones

Luego de realizada la investigación y analizado los resultados, se recomienda:

- Apoyándose del marco de trabajo Apache Cordova, compilar la solución para los SO *BlackBerry OS*, *Windows Phone* y iOS, este último utilizando el Certificado de Desarrollador de Apple.
- Ampliar el marco de acción de la aplicación agregándole servicios de climatización, seguridad o comunicación.

Bibliografía

1. *libro domótica final_libro domótica final.qxd - libro_domotica.pdf* [online]. [Accessed 3 June 2017]. Available from: http://lsi.vc.ehu.es/pablogn/investig/dom%C3%B3tica/libro_domotica.pdf

2. J.ANTONIO. *"Diseño tecnológico. Electrónica y ocio. Domótica e Inmótica."* 2009.

3. ANTONIO J. JARA VARELA. *"Introducción de la vivienda. Viviendas inteligentes. ."* 2000.

4. 7 apps para controlar tu casa con un dedo desde tu smartphone o tablet. *idealista/news* [online]. [Accessed 15 November 2016]. Available from: https://www.idealista.com/news/inmobiliario/vivienda/2014/08/29/730815-7-apps-para-controlar-tu-casa-con-un-dedo-desde-tu-smartphone-o-tabletDesde que en 1955 el estadounidense eugene polley inventase el primer mando a distancia para la televisión, el ser humano ha soñado con controlar todo lo que tiene a su alrededor simplemente con un dedo. Pues bien, hoy es posible.

5. WeMo® Light Switch. *Belkin* [online]. [Accessed 15 November 2016]. Available from: http://www.belkin.com/us/p/P-F7C030The WeMo® Wi-Fi enabled Light Switch

6. GABRIEL OSMAR PEDROZO PETRAZZINI. *"Sistemas Operativos en Dispositivos Móviles."* 2012.

7. Operating system market share. [online]. [Accessed 19 June 2017]. Available from: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomc=&qpcd=130&qpcustomb=&qpcustomd=1

8. GABRIEL OSMAR PEDROZO PETRAZZINI. *Licenciatura en Sistemas de Información*. UNIVERSIDAD NACIONAL DEL NORDESTE, 2012.

9. Historia. *Sistema Android* [online]. 13 November 2012. [Accessed 27 June 2017]. Available from: https://scoello12.wordpress.com/historia/HISTORIA: Android es un sistema operativo basado en Linux, desarrollado por la empresa Android Inc., creado para teléfonos móviles inteligentes y el cual era prácticamente desconocido hasta que en …

10. Ventajas y Desventajas. *Sistema Android* [online]. 13 November 2012. [Accessed 20 June 2017]. Available from: https://scoello12.wordpress.com/ventajas-y-desventajas/VENTAJAS DEL SISTEMA ANDROID 1. El código de Android es abierto: Google liberó Android bajo licencia Apache. Cualquier persona puede realizar una aplicación para Android. 2. Hoy día hay más de 650.…

11. 1. ¿Qué es un microcontrolador? | Sherlin.xBot.es. [online]. [Accessed 15 November 2016]. Available from: http://sherlin.xbot.es/microcontroladores/introduccion-a-los-microcontroladores/que-es-un-microcontrolador

12. Redes inalámbricas , tipos y características .:: www.informaticamoderna.com ::. [online]. [Accessed 28 November 2016]. Available from: http://www.informaticamoderna.com/Redes_inalam.htm

13. Significado de Wifi. *Significados* [online]. [Accessed 17 November 2016]. Available from: http://www.significados.com/wifi/Qué es Wifi. Concepto y Significado de Wifi: Wifi es una tecnología de comunicación inalámbrica que permite conectar a internet...

14. Perspectiva general - Apache Cordova. [online]. [Accessed 25 January 2017]. Available from: https://cordova.apache.org/docs/es/3.1.0/guide/overview/

15. Sublime Text, un sofisticado editor de código multiplataforma - Genbeta. [online]. [Accessed 25 January 2017]. Available from: http://www.genbeta.com/herramientas/sublime-text-un-sofisticado-editor-de-codigo-multiplataforma

16. HTML5. Mozilla Developer Network. [online]. Available from: https://developer.mozilla.org/es/docs/HTML/HTML5

17. HTML. Mozilla Developer Network. [online]. Available from: https://developer.mozilla.org/es/docs/Web/HTML

18. CSS3. Mozilla Developer Network. [online]. Available from: https://developer.mozilla.org/es/docs/Web/CSS/CSS3

19. CSS. Mozilla Developer Network. [online]. Available from: https://developer.mozilla.org/es/docs/Web/CSS

20. Acerca de JavaScript. Mozilla Developer Network. [online]. Available from: https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript

21. Bootstrap • The world's most popular mobile-first and responsive front-end framework. [online]. Available from: http://getbootstrap.com/

22. JSON. [online]. [Accessed 25 January 2017]. Available from: http://www.json.org/json-es.html

23. ¿Qué es AngularJS? Introducción a este framework MVW. [online]. [Accessed 25 January 2017]. Available from: http://blog.escuelaweb.net/que-es-angularjs-introduccion-a-este-framework-mvw/

24. Qué es AngularJS. [online]. [Accessed 25 January 2017]. Available from: http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html

25. Herramientas case. [online]. [Accessed 15 November 2016]. Available from: http://es.slideshare.net/00menni/herramientas-case-63470520

26. Visual Paradigm Product Overview. [online]. [Accessed 15 November 2016]. Available from: https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html

27. RUSS MILES and KIM HAMILTON. *´´Learning UML 2.0´´*. 2006.

28. YOENIA LAPIDO RAMIREZ. *´´HMI para el sistema Arex sobre dispositivos mo´viles con sistema operativo Android´´*. La Habana : Universidad de Ciencias Informaticas, 2015.

29. ENRIQUE HERNANDEZ ORALLO. *´´El Lenguaje Unificado de Modelado(UML)´´*. [no date].

30. AGUILAR, Yunior Duque and ANGEL ROLANDO MAURE MOREJÓN. *Gestión de tipos documentales del Sistema para la obtención de documentos digitales con valor legal.*

31. TAMARA RODRÍGUEZ SÁNCHEZ. *Metodología de desarrollo para la Actividad productiva de la UCI*. [no date].

32. Requisitos para Sistemas de Información - get.php. [online]. [Accessed 2 June 2017]. Available from: http://www.lsi.us.es/docencia/get.php?id=6890

33. La interfaz de linea de comandos - Apache Cordova. [online]. [Accessed 6 June 2017]. Available from: https://cordova.apache.org/docs/es/latest/guide/cli/

34. Guía de la plataforma Android - Apache Cordova. [online]. [Accessed 13 June 2017]. Available from: https://cordova.apache.org/docs/es/latest/guide/platforms/android/

35. New Features in Xcode 7. [online]. [Accessed 18 June 2017]. Available from: https://developer.apple.com/library/content/documentation/Xcode/Conceptual/WhatsNewXcode-Archive/Articles/xcode_7_0.html

36. MOISES CRUZ. Diagramas de paquetes. [online]. 01:51:05 UTC. [Accessed 4 June 2017]. Available from: https://es.slideshare.net/moyscruz/diagramas-de-paquetes

37. *IEEE Recommended practice for architectural description of software-intensive systems - IEEE Std 1471-2000 - ieee1471.pdf* [online]. [Accessed 4 June 2017]. Available from: http://cabibbo.dia.uniroma3.it/ids/altrui/ieee1471.pdf

38. *ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF* [online]. [Accessed 4 June 2017]. Available from: http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

39. DESARROLLOWEB.COM. Qué es MVC. *DesarrolloWeb.com* [online]. [Accessed 4 June 2017]. Available from: http://www.desarrolloweb.com/articulos/que-es-mvc.html

40. Qué son y para qué sirven los patrones de diseño. *platzi.com* [online]. [Accessed 4 June 2017]. Available from: https://platzi.com/blog/patrones-de-diseno/Cuando empezamos a desarrollar software, es común que cada quien utilice su propia lógica, conocimientos y experiencia para crear código. Y esto muchas vec

41. Revisiones de código y estándares de codificación. [online]. [Accessed 4 June 2017]. Available from: https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx

42. Estilo de Programación. [online]. [Accessed 4 June 2017]. Available from: http://www.lab.dit.upm.es/~lprg/material/apuntes/doc/estilo.htm

43. Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue. [online]. [Accessed 4 June 2017]. Available from: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html

44. JUAN.-. Un poco de Sistemas: Pruebas de estrés, carga y rendimiento. *Un poco de Sistemas* [online]. 25 March 2010. [Accessed 4 June 2017]. Available from: http://juanbevilacqua.blogspot.com/2010/03/pruebas-de-estres-carga-y-rendimiento.html