



TRABAJO DE FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA

Observatorio Remoto: un cliente INDI para Android

Autor

Jaime Torres Benavente

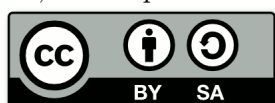
Tutor

Dr. Sergio Alonso Burgos



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, 8 de septiembre de 2015



Observatorio Remoto: un cliente INDI para Android

Jaime Torres Benavente

Yo, **Jaime Torres Benavente**, alumno de la titulación **Grado en Ingeniería Informática** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado (*Observatorio Remoto: un cliente INDI para Android*) en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Además, este mismo trabajo es realizado bajo licencia **Creative Commons Attribution-ShareAlike 4.0** (<https://creativecommons.org/licenses/by-sa/4.0/>), dando permiso para copiarlo y redistribuirlo en cualquier medio o formato, también de adaptarlo de la forma que se quiera, pero todo esto siempre y cuando se reconozca la autoría y se distribuya con la misma licencia que el trabajo original. El documento en formato **LaTeX** así como el código del proyecto se puede encontrar en el siguiente repositorio de **GitHub**: <https://github.com/torresj/indi-android-ui>.

Fdo: Jaime Torres Benavente

Granada, a 8 de septiembre de 2015

D. Dr. Sergio Alonso Burgos, profesor del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *Observatorio Remoto: un cliente INDI para Android*, ha sido realizado bajo su supervisión por **Jaime Torres Benavente**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 8 de septiembre de 2015.

El tutor:

Dr. Sergio Alonso Burgos

Agradecimientos

A mi familia porque gracias a ellos soy como soy.

A mis amigos porque sin ellos no habría podido llegar hasta aquí

A mi tutor **Dr. Sergio Alonso Burgos**, por aguantarme durante tantos meses con una paciencia infinita.

Índice general

1. Introducción	1
1.1. La Astronomía	1
1.2. Instrumental Astronómico	3
1.2.1. Telescopios	3
1.2.2. Cámaras CCD	4
1.2.3. Monturas	5
1.2.4. Rueda portafiltro	6
1.2.5. Enfocadores	6
1.2.6. Cúpulas	6
1.2.7. Ópticas adaptativas	6
1.2.8. Estaciones meteorológicas	7
1.3. Control de dispositivos astronómicos	8
1.4. INDI	9
1.4.1. INDI for Java	9
1.5. Dispositivos Móviles	10
1.5.1. iOS	11
1.5.2. Android	12
2. Objetivos	15
2.1. Alcance de los objetivos	17
2.2. Interdependencia de los objetivos	17
3. Planificación	19
3.1. Fases	19
3.2. Estimación de tiempos	20
3.3. Recursos humanos	21
3.4. Presupuesto	22
3.5. SCRUM	22
3.6. Temporización	24
4. Análisis	27
4.1. Análisis de requisitos	27
4.1.1. Requisitos funcionales	27

4.1.2. Requisitos no funcionales	28
4.2. Casos de uso	28
4.2.1. Descripción de actores	28
4.2.2. Descripción casos de uso	29
4.3. Diagrama de paquetes	44
4.4. Diagramas de casos de uso	44
4.5. Diagramas de actividad	48
4.6. Diagrama conceptual	57
4.7. Otros diagramas o interfaces	57

Índice de figuras

1.1. ASCOM Standard (http://ascom-standards.org/)	8
1.2. INDI Client (http://www.indilib.org/)	10
1.3. INDI Server (http://www.indilib.org/)	10
3.1. Proceso de desarrollo Scrum (http://www.qasoluciones.es/metodologia/agile)	23
3.2. Diagrama de Grantt inicial	24
3.3. Diagrama de Grantt final	25
4.1. Diagrama de paquetes	44
4.2. Diagrama de casos de uso: paquete Administración portal	45
4.3. Diagrama de casos de uso: paquete Acceso información	45
4.4. Diagrama de casos de uso: paquete Pruebas de software	47
4.5. Diagrama de casos de uso: paquete Configuración automática	48
4.6. Diagrama de actividad CU-1. Inicio automático del servidor del portal	49
4.7. Diagrama de actividad CU-2. Consultar información de Administración	50
4.8. Diagrama de actividad CU-3. Consultar información de Docencia	51
4.9. Diagrama de actividad CU-4. Consultar información de Gestión e Investigación	52
4.10. Diagrama de actividad CU-5. Consultar información de Normativa Legal	53
4.11. Diagrama de actividad CU-6. Realizar tests unitarios	54
4.12. Diagrama de actividad CU-7. Realizar test de cobertura	55
4.13. Diagrama de actividad CU-8. Usar integración continua	56
4.14. Diagrama de actividad CU-9. Usar despliegue automático	57
4.15. Diagrama de actividad CU-10. Usar provisionamiento	58
4.16. Diagrama conceptual	59

Índice de tablas

4.1. CU-1. Añadir nueva conexión.	29
4.2. CU-2. Editar una conexión.	30
4.3. CU-3. Borrar conexiones.	31
4.4. CU-4. Editar los ajustes.	32
4.5. CU-4. Conterase a un servidor.	33
4.6. Curso alternativo de CU-5. Conectarse a un servidor.	33
4.7. CU-6. Desconectarse de un servidor.	34
4.8. CU-7. Salir de la aplicación.	35
4.9. CU-8. Mostrar dispositivo.	36
4.10. CU-9. Cambiar vista de dispositivo.	37
4.11. CU-10. Editar una propiedad <i>text</i>	38
4.12. Curso alternativo de CU-10. Editar una propiedad <i>text</i>	38
4.13. CU-11 Editar una propiedad <i>number</i>	39
4.14. Curso alternativo de CU-11. Editar una propiedad <i>number</i>	39
4.15. CU-12 Editar una propiedad <i>switch</i>	40
4.16. Curso alternativo de CU-12. Editar una propiedad <i>switch</i>	40
4.17. CU-13 Editar una propiedad <i>blob</i>	41
4.18. Curso alternativo de CU-13. Editar una propiedad <i>blob</i>	41
4.19. CU-14 Editar una propiedad <i>connection</i>	42
4.20. CU-15 Editar una propiedad <i>abort</i>	43

Índice de fragmentos de código

Capítulo 1

Introducción

1.1. La Astronomía

Desde el principio de los tiempos, el ser humano a mirado al cielo con incertidumbre, viendolo como una fuente inagotable de interrogantes sin resolver. En casi todas las religiones antiguas existía la “**cosmogonía**” que intentaba explicar el origen del universo, ligando este a los elementos mitológicos, dando paso esta a la “**astronomía**”:

“Ciencia que se ocupa del estudio de los cuerpos celestes del universo, incluidos los planetas y sus satélites, los cometas y meteoritos, las estrellas y la materia interestelar, los sistemas de materia oscura, estrellas, gas y polvo llamados galaxias y los cúmulos de galaxias; por lo que estudia sus movimientos y fenómenos ligados a ellos.”

(<https://es.wikipedia.org/wiki/Astronom%C3%ADa>)

La Astronomía es probablemente la más antigua de las ciencias naturales originándose en la antigüedad en casi todas las culturas humanas. Sus orígenes se pierden en prácticas religiosas de la prehistoria cuyos vestigios se encuentran en numerosos sitios arqueológicos (como Stonehenge) e incorporados todavía en la astrología una disciplina entrelazada con la astronomía y no separada de ella completamente hasta el siglo XVIII en el mundo occidental. La astronomía antigua constituyó las bases del calendario y la medida de periodos temporales como la semana el mes o el año. Los astrónomos antiguos eran capaces de distinguir entre estrellas y planetas dado que las primeras permanecen fijas en sus posiciones relativas mientras que los planetas se mueven una cantidad apreciable de espacio a lo largo de periodos relativamente cortos (Saturno el más lento de los planetas conocidos en la antigüedad describe un periodo orbital en 29 años). La Astronomía antigua culmina con el desarrollo ordenado del modelo heliocéntrico expuesto

en las obras de *Ptolomeo*. Previamente *Aristarco de Samos* había medido las distancias de la Tierra a la Luna y al Sol afirmando como consecuencia de éstas que el Sol era el centro del Universo alrededor del cual giraban los demás planetas incluyendo la Tierra. Otros logros destacados de la época clásica de la astronomía fueron los conseguidos por *Hiparco* quien realizó el primer catálogo estelar y propuso un sistema de clasificación estelar en 6 magnitudes basado en la luminosidad aparente de las diferentes estrellas. La Astronomía en la Europa medieval se produce un oscurantismo en todos los campos del conocimiento incluyendo la astronomía. Ésta permanece preservada en escasas copias de tratados antiguos de la astronomía griega y romana. La astronomía observacional tan sólo se conserva en el mundo árabe.

Tycho Brahe (1546-1601) introdujo la idea de la precisión de la medida en astronomía e inventó y produjo una gran cantidad de instrumental astronómico previo al telescopio. *Galileo Galilei* (1564-1642) construyó su propio telescopio a partir de un invento holandés y lo utilizó inmediatamente en el estudio astronómico descubriendo los cráteres de la Luna, las lunas de Júpiter y las manchas solares. Sus observaciones tan sólo eran compatibles con el modelo **copernicano**. Paralelamente *Johannes Kepler* expuso sus famosas **leyes de Kepler** para el movimiento de los planetas basando su trabajo en las detalladas observaciones de *Tycho Brahe*.

Una generación más tarde Isaac Newton fue el primer científico que unió la Física con la Astronomía proponiendo que las mismas fuerzas que hacían caer los cuerpos sobre la Tierra causaban el movimiento de los planetas y la Luna. Utilizando su Ley de la gravedad las leyes de Kepler resultan inmediatamente explicadas. Newton también descubrió que la Luz blanca del Sol está descompuesta en diferentes colors, un hecho importantísimo para el futuro desarrollo de la astronomía.

La **astronomía** es una de las pocas ciencias en las que los aficionados aún pueden desempeñar un papel activo, especialmente en el descubrimientos y seguimiento de fenómenos. Es por ello que existe una gran variedad de **herramientas** e **instrumental astronómico** que permiten a cualquier persona observar el universo.

1.2. Instrumental Astronómico

Existe una gran variedad de **instrumental astronómico** en la actualidad. A continuación se describen las familias más importantes.

1.2.1. Telescopios

“El telescopio es un instrumento óptico que permite ver objetos lejanos con mucho más detalle que a simple vista al captar radiación electromagnética, tal como la luz.”

(<https://es.wikipedia.org/wiki/Telescopio>)

Hace cuatro siglos nació un invento que habría de redefinir nuestro lugar en el universo. Tachado en su momento como el instrumento más diabólico de la historia, el telescopio sacudió la sociedad hasta las raíces. Al alzar los ojos al cielo nos convencimos de que éramos el centro de la creación, y había razones para ello: desde nuestra perspectiva, todo parece girar en torno a la Tierra

Los fabricantes de vidrio sabían desde la antigüedad que una esfera de vidrio podía aumentar imágenes, pero tuvieron que pasar siglos antes de que alguien ensamblara dos lentes en un tubo y mirara a través de ellas. Señalar la fecha, lugar y autor exactos de su invención es controvertido. Los holandeses se inclinan por el 2 de octubre de 1608, el día en que *Hans Lippershey* patentó un instrumento llamado **kijker**, que significa mirador. Un moledor de vidrio holandés aseguraba haber inventado un aparato similar, pero el primero en patentarlo fue *Lippershey*. Como era alemán, vivía en Holanda y registró la patente en Bélgica, más de un país ha pugnado por el honor de su autoría. Sin embargo, como dijo *Darwin*:

“en la ciencia el crédito es del que convence al mundo y no del primero en tener la idea”

(Charles Darwin)

Por eso la gloria se la llevó Italia, ya que fueron las mejoras que introdujo *Galileo* las que permitieron usar el aparato como instrumento astronómico. El diseño de *Galileo* consistía en una lente convexa para el objetivo y otra cóncava en el ocular. En 1611 el alemán *Johannes Kepler* fue el primero en usar dos lentes convexas que enfocaban los rayos en un mismo punto. La configuración de *Kepler* aún se usa en binoculares y cámaras fotográficas modernas y es la base del telescopio refractor.

Tras la muerte de *Galileo*, fue *Isaac Newton* quien nos dio una nueva imagen del universo que sobrevivió 250 años hasta la llegada de *Albert Einstein*.

“Si he logrado ver más lejos ha sido porque me he subido a hombros de gigantes”

(Isaac Newton)

Y así, sobre la herencia de *Galileo*, *Newton* inventó el **telescopio reflector**, que es la base de los actuales. La innovación consistía en usar espejos en lugar de lentes para enfocar la luz y formar imágenes. Entonces el universo se nos abrió en todo su esplendor.

1.2.2. Cámaras CCD

Un dispositivo de carga acoplada (en inglés **Charge-Coupled Device**, conocido también como **CCD**), es un circuito integrado que contiene un número determinado de condensadores enlazados o acoplados. Bajo el control de un circuito interno, cada condensador puede transferir su carga eléctrica a uno o a varios de los condensadores que estén a su lado en el circuito impreso.

El **CCD** se inventó a finales de los 60 por investigadores de **Bell Laboratories**. Originalmente se concibió como un nuevo tipo de memoria de ordenador pero pronto se observó que tenía muchas más aplicaciones potenciales tales como el proceso de señales y sobre todo la captación de imagen, esto último debido a la sensibilidad a la luz que presenta el silicio.

El sensor **CCD** de una cámara digital es como el motor de un coche, es la pieza principal. En su forma más elemental, el **CCD** es como un ojo electrónico que recoge la luz y la convierte en una señal eléctrica. Tienen dos diferencias básicas con los fotomultiplicadores:

Los sensores **CCD** son de menor tamaño y están contruidos de semiconductores lo que permite la integración de millones de dispositivos sensibles en un solo chip. La eficiencia cuántica de los **CCD** (sensibilidad) es mayor para los rojos. Los fotomultiplicadores son más sensibles a los azules.

Físicamente, un **CCD** es una malla muy empaquetada de electrodos de polisilicio colocados sobre la superficie de un chip. Al impactar los fotones sobre el silicio se generan electrones generados que pueden guardarse temporalmente. Periódicamente se lee el contenido de cada pixel haciendo que los electrones se desplacen físicamente desde la posición donde se originaron (en la superficie del chip), hacia el amplificador de señal con lo que se genera una corriente eléctrica que será proporcional al número de fotones que llegaron al pixel. Para coordinar los periodos de almacenamiento (tiempo de exposición) y vaciado del pixel (lectura del pixel) debe existir una fuente eléctrica externa que marque el ritmo de almacenamiento-lectura: el reloj

del sistema. La forma y amplitud de reloj son críticas en la operación de lectura del contenido de los píxeles.

Al tratarse el **CCD** de un dispositivo semiconductor, técnicamente es posible implementar en él todas las funciones electrónicas de un sistema de captación de imagen, pero esto no es rentable económicamente y por tanto se implementa en otros chips externos al **CCD**: la mayoría de **CCD** de cámaras tienen varios chips (de tres a ocho).

La necesidad de usar chips distintos implica dos desventajas importantes; la necesidad de voltajes múltiples de abastecimiento de los chips y un gran consumo de potencia de todo el sistema electrónico.

1.2.3. Monturas

La montura de un telescopio es la parte mecánica que une el trípode o base al instrumento óptico. Existen varios tipos de monturas, algunas muy simples, otras mas complejas, incluso con correctores electrónicos y dispositivos de localización y seguimiento muy sofisticados (sistemas **GOTO**)

La montura tiene como objetivo proveer de movimiento controlado al telescopio. Es muy importante la firmeza y suavidad de los movimientos, para que la observación sea confortable y las astrofotografías perfectas. Las monturas se clasifican en dos grandes grupos, según los planos de referencia que utilicen (coordenadas).

La más simple es la montura altacimutal, que realiza movimientos horizontales y verticales (acimut y altura, respectivamente). Este tipo de diseño lo traen incorporados los telescopios pequeños, por lo general telescopios refractores de uso terrestre, dado que su uso es simple, y también varios modelos de equipos automatizados (sistemas **GOTO**)

Le sigue la montura ecuatorial, que utiliza como plano fundamental el ecuador celeste (proyección del ecuador terrestre). Este diseño usa las coordenadas ecuatoriales, ascensión recta (A.R. o R.A.) y declinación (Dec.), que son proyecciones de las coordenadas terrestres longitud y latitud, respectivamente, sobre la esfera celeste.

Existen varios tipos de monturas basados en los dos diseños fundamentales anteriores. La montura **Dobson** por ejemplo (suelen llamarse telescopios *dobsonianos* a los que la poseen), es un modelo basado en la altacimutal, sin trípode y un telescopio de diseño newtoniano como instrumento de observación. Es muy utilizado por los que desean una gran apertura en reflectores, por ejemplo los que se construyen su propio espejo y no quieren tener grandes gastos en monturas sofisticadas.

1.2.4. Rueda portafiltro

La rueda porta-filtros consiste en un cuerpo, generalmente de aluminio, que en su interior puede alojar varios filtros, normalmente de 1,25" de diámetro. Lo aconsejable es que tenga, al menos, 4 huecos para filtros si queremos hacer astrofotografía con cámaras CCD blanco y negro, puesto que vamos a necesitar el azul, rojo y verde (RGB) y, posiblemente, un filtro para infrarrojos.

1.2.5. Enfocadores

El **enfocador** es una pieza fundamental del telescopio. Nos permitirá ver las imágenes formadas tras la reflexión de la luz en el espejo primario y su desviación por el espejo secundario. Para verlas necesitaremos un juego de oculares. La longitud focal de los oculares combinada con la longitud focal de nuestro telescopio nos dará el número de aumentos total del sistema. Dichos oculares están montados en el **enfocador**, un dispositivo móvil que permitirá mover la posición vertical del ocular para enfocar adecuadamente la imagen.

Un ejemplo de enfocador son los de tipo **Crayford** y los **rack and pinion**.

1.2.6. Cúpulas

Las **cúpulas** son recintos cerrados mas o menos grandes que nos permiten albergar y proteger el instrumental astronómico. De esta forma, las **cúpulas** pueden ser abiertas o cerradas para exponer los instrumentos en el momento de las observaciones.

1.2.7. Ópticas adaptativas

La **óptica adaptativa** es una técnica que permite corregir las perturbaciones más importantes que sufren las imágenes astronómicas debido a la atmósfera terrestre. Con este sistema es posible obtener imágenes más nítidas o de mejor resolución espacial. La diferencia que introduce esta técnica es comparable a la que existe entre mirar un objeto situado en el fondo de una piscina con agua o sin agua.

Las posibilidades que la óptica adaptativa ofrece a la astronomía son espectaculares. Eliminar las perturbaciones producidas por la atmósfera equivale esencialmente a observar desde el espacio. Las perturbaciones atmosféricas causan una pérdida en nitidez o resolución espacial. Esta pérdida se traduce, por un lado, en una disminuida capacidad para resolver objetos, es decir, para realizar estudios detallados de su morfología. Por otro lado,

influye también en la capacidad de detectar objetos débiles, dado que la imagen se dispersa en puntos de luz mayores.

La mejora que introduce la óptica adaptativa se puede cuantificar utilizando la relación entre el tamaño del telescopio y el tamaño de la mejor imagen que puede obtener. El poder de detección de un telescopio aumenta con el diámetro de su espejo primario y disminuye con el tamaño de la imagen que forma de un objeto puntual (de aquí la importancia de la calidad de imagen en un telescopio). Por tanto, la diferencia con un mismo espejo de 10 metros, entre conseguir enfocar imágenes de 0.4 segundos de arco (lo posible en una noche de visibilidad excelente) y una imagen de 0.04 segundos de arco, que debe ser posible con un sistema de óptica adaptativa, equivaldría a tener un espejo primario de 100 metros

1.2.8. Estaciones meteorológicas

Las **estaciones meteorológicas** son sistemas compuestos por un “*data logger*” y un conjunto de sensores que nos proporcionan datos de las distintas magnitudes meteorológicas, tales como la temperatura, humedad, presión barométrica, etc... permitiéndonos generar modelos a partir de los cuales conocer la situación climática y su posible evolución.

Gracias a los datos aportados por las **estaciones meteorológicas**, podemos conocer la climatología en el momento de realizar observaciones astronómicas. De esta forma podemos decidir si las condiciones son óptimas, o incluso decidir si debemos cerrar la cúpula para evitar daños en los instrumentos por lluvias o similar.

1.3. Control de dispositivos astronómicos

Actualmente existen diversas formas de controlar los dispositivos astronómicos pero la mayoría presenta los mismos inconvenientes:

- Normalmente se controlan los dispositivos directamente.
- Se conecta el dispositivo a un PC y se trabaja desde él.
- Se utilizan herramientas para el control remoto como el “escritorio remoto”.

Por otro lado, existen estándares como el de **ASCOM** para instrumental astronómico. Con él, se intenta crear una capa entre los programas para controlar dispositivos astronómicos y los propios dispositivos. La principal desventaja es que solo puede utilizarse en sistemas *Microsoft Windows*

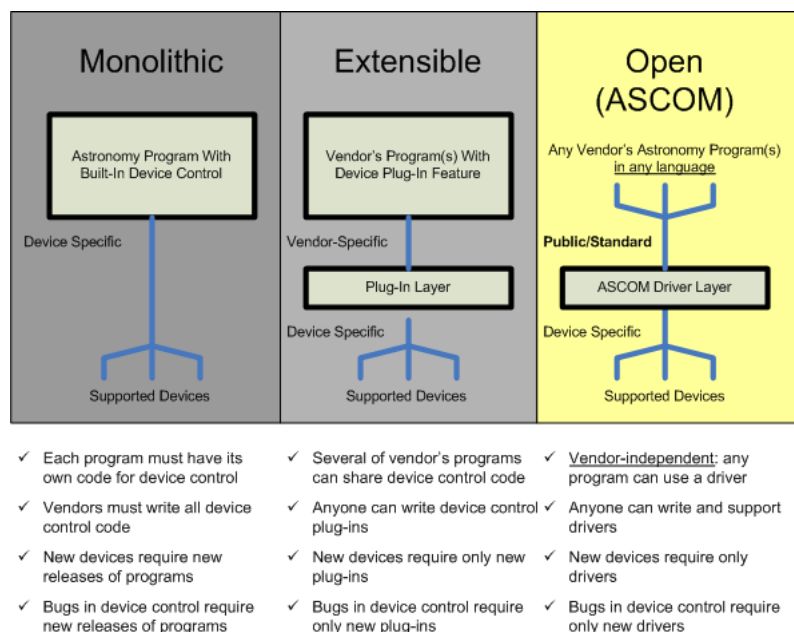


Figura 1.1: ASCOM Standard (<http://ascom-standards.org/>)

1.4. INDI

“The Instrument Neutral Distributed Interface (INDI) Library is a cross-platform software designed for automation control of astronomical instruments. It supports a wide variety of telescopes, CCDs, focusers, filter wheels..etc, and it has the capability to support virtually any device. INDI is small, flexible, easy to parse, and scalable. It supports common DCS functions such as remote control, data acquisition, monitoring, and a lot more. With INDI, you have a total transparent control over your instruments so you can get more science with less time.”
(<http://indilib.org/about.html>)

El protocolo **INDI** es una plataforma software diseñada para el control de instrumental astronómico. La biblioteca **INDI** permite controlar cualquier dispositivo con un driver **INDI** mediante el paso de archivos XML. Sus principales ventajas frente a otras soluciones para el control de dispositivos son:

- Es una biblioteca ligera, flexible y escalable.
- Es de código abierto por lo que cualquiera puede ver su código y mejorarlo o crear drivers para cualquier dispositivo
- El intercambio de información es mínimo.
- Es multiplataforma.
- Separa el cliente del servidor.
- Los fabricantes comienzan a desarrollar drivers para sus dispositivos o liberan las especificaciones para que la comunidad pueda desarrollarlos.
- Existen numerosos clientes INDI como <https://edu.kde.org/kstars/>

1.4.1. INDI for Java

La biblioteca **INDI** está escrita en lenguaje **C**, pero existe una implementación realizada en Java y que se encuentra en desarrollo. En la página oficial de **INDI** <http://indilib.org/develop/indiforjava.html> podemos encontrar toda la información sobre nuevas versiones y la documentación para poder utilizarla. La principal ventaja de poder usar Java es que podemos implementar drivers y clientes con la potencia de un lenguaje Orientado a Objetos y combinarlo con otras tecnologías como los dispositivos móviles basados en la plataforma **Android**

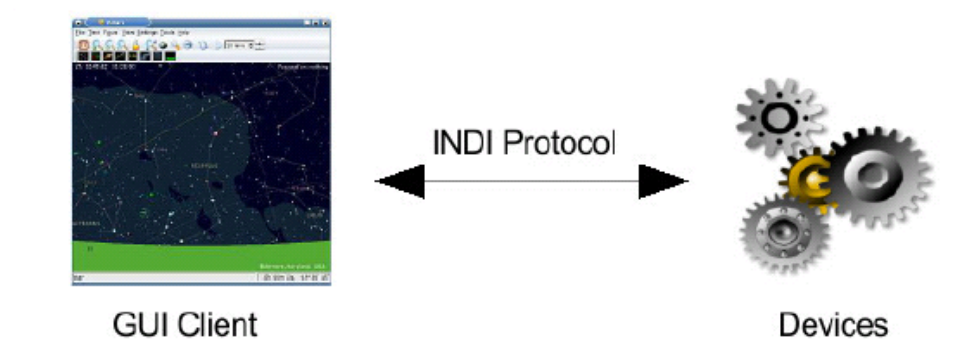


Figura 1.2: INDI Client (<http://www.indilib.org/>)

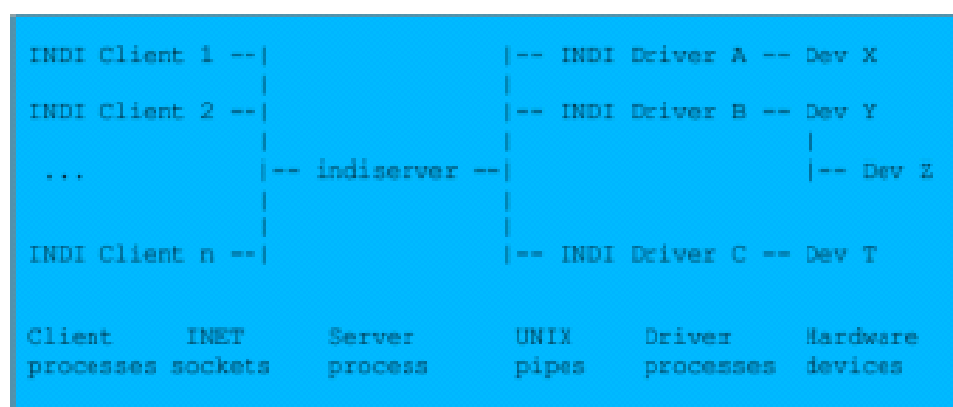


Figura 1.3: INDI Server (<http://www.indilib.org/>)

1.5. Dispositivos Móviles

Un **dispositivo móvil** es un tipo de computadora de tamaño pequeño, con capacidad de procesamiento, con conexión a internet, con memoria, diseñados específicamente para una función pero que pueden llevar a cabo otras funciones más generales.

Los **dispositivos móviles** hoy en día están integrados en la mayoría de tareas cotidianas de una persona. La tendencia de la sociedad actual nos empuja hacia un mundo cada vez más móvil donde necesitamos estar conectado e interactuar con otros sistemas. Es por ello que la mayoría de soluciones tecnológicas, hayan sido pensadas o no para el sector de los dispositivos móviles, siempre acaba teniendo una versión para éstos.

Paralelamente a la expansión de los **dispositivos móviles**, se han creado un gran número de sistemas operativos para estos dispositivos entre los

que se encuentra:

- Android.
- iOS.
- BlackBerry OS.
- Palm OS.
- Windows Mobile/Phone.
- Symbian.

Actualmente **Android** y **iOS** copan el 96.3 % del mercado¹ . Por lo que nos centraremos principalmente en estos S.O.² y los **dispositivos móviles** compatibles con ellos.

1.5.1. iOS

iOS es un S.O. móvil de la compañía *Apple Inc* originalmente desarrollado para el *iPhone*³ y posteriormente introducido en otros **dispositivos móviles** de la compañía como el iPod touch⁴ y el iPad⁵. **iOS** no puede ser instalado en hardware de terceros.

Actualmente tiene una cuota de mercado aproximadamente del 19.7 %, siendo el segundo S.O. más utilizado.

iOS es un sistema muy estable, diseñado para un hardware muy concreto y por tanto, muy eficiente y depurado. Pero de cara a elegirlo como una opción a la hora de desarrollar una nueva aplicación para **dispositivos móviles** se debe tener en cuenta los siguientes aspectos:

- Hay que pagar una cuota anual de 99\$ para poder publicar aplicaciones en el *Apple Store*⁶. Además si esta licencia, no podremos desarrollar aplicación y cargarla en nuestros dispositivos Apple.
- Necesitamos un MAC⁷ ya que las herramientas para el desarrollador solo pueden utilizarse en sus equipos.

¹Fuente:<http://www.idc.com/getdoc.jsp?containerId=prUS25450615>.

²Sistema Operativo.

³Smartphone de la compañía Apple Inc.

⁴Dispositivo móvil para reproducir multimedia de la compañía Apple Inc

⁵Tablet de la compañía Apple Inc.

⁶Tienda de aplicaciones de la compañía Apple Inc.

⁷Computadoras personales de la compañía Apple Inc.

- Necesitaremos conocer el lenguaje de programación **Objective-C**
- **iOS** es un sistema de código cerrado que va en contra de la filosofía del **Software Libre** y el código abierto y reutilizable.

Aunque **iOS** es un sistema muy extendido y con un gran número de usuarios, creemos que no es la mejor opción para orientar una aplicación móvil basada en **Software Libre** además de la inversión anual requerida para poder publicar una aplicación que pretendamos sea gratuita, libre y accesible a cualquier usuario.

1.5.2. Android

Android es un Sistema Operativo basado en un **núcleo Linux**⁸. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, relojes inteligentes, televisiones inteligentes y automóviles. Inicialmente pertenecía a la compañía **Android Inc.** que posteriormente sería adquirida por **Google**. Actualmente posee la mayor cuota de mercado de aproximadamente el 76.6 %.

Los principales componentes del sistema operativo **Android** son:

- **Aplicaciones:** Todas las aplicaciones están escritas en lenguaje de programación **Java**.
- **Framework**⁹: Los desarrolladores tienen acceso completo a las mismas APIs¹⁰ que utiliza el sistema. La arquitectura está diseñada para simplificar la reutilización de componentes.
- **Runtime de Android:** Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje **Java**. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual *Dalvik*¹¹.
- **Núcleo Linux:** Android depende de Linux para los servicios base del sistema como la seguridad, la gestión de memoria, la gestión de procesos, etc. El núcleo Linux también sirve como capa de abstracción entre el hardware y el software.

⁸Sistema operativo basado en Unix

⁹Marco de trabajo para los desarrolladores

¹⁰Interfaz de programación de aplicaciones (Application Programming Interface)

¹¹Máquina virtual que utiliza la plataforma Android para ejecutar aplicaciones Java.

Android no tiene restricciones de uso por lo que puede utilizarse en número muy extenso de dispositivos móviles. Además es un sistema parcialmente de código abierto. Está basado en Linux y la mayoría del código es abierto aunque no todo el sistema lo es.

De cara al desarrollo de aplicaciones móviles, **Android** es una opción muy recomendable por las siguientes razones:

- La arquitectura del sistema (basada en Linux, lenguaje de programación Java,...)
- La mayoría de los dispositivos móviles del mundo tienen como sistema operativo a **Android** por lo que la difusión será mayor que con otros sistemas.
- Las herramientas para desarrollar en Android son multiplataforma y gratuitas. Para poder crear y probar una aplicación solo necesitas un ordenador con cualquier sistema operativo, un dispositivo móvil con android y descargar la herramienta para desarrolladores.
- Para poder publicar aplicaciones en *Google Play*¹² hay que pagar 25\$ sin tener renovarlo anualmente y sin ninguna limitación.

¹²Tienda de aplicaciones para dispositivos Android

Capítulo 2

Objetivos

El objetivo de este proyecto es desarrollar una aplicación para la plataforma **Android** que implemente un cliente utilizando la biblioteca “**INDI for Java**” basado en el **Software Libre** y que sea fácilmente extensible.

A continuación se describen los objetivos principales a alcanzar:

- **OBJ-1.** Conseguir un cliente funcional capaz de controlar cualquier dispositivo **INDI**.
- **OBJ-2.** Poder gestionar múltiples conexiones con múltiples dispositivos simultáneamente.
- **OBJ-3.** Facilmente extensible, permitiendo añadir vistas para propiedades y dispositivos por parte de desarrolladores ajenos al proyecto.
- **OBJ-4.** Desarrollar la aplicación bajo una licencia de código abierto fomentando la filosofía del **Software Libre** y la publicación de todo el código.

Además de los objetivos principales, se persigue alcanzar los siguientes objetivos:

- **OBJ-S-1.** Desarrollar la aplicación siguiendo los estándares actuales y las recomendaciones para la plataforma **Android**.
- **OBJ-S-2.** Facilitar la usabilidad mediante un diseño adecuado de las interfaces, adaptándola a los distintos tamaños de pantalla y personalizándolas a las propiedades estándares de **INDI**.
- **OBJ-S-3.** Desarrollar para incluir un correcto funcionamiento en el mayor número posible de versiones de **Android**, maximizando el número de dispositivos compatibles.

- **OBJ-S-4.** Añadir una versión estable en *Google Play* y publicar el *APK*¹ para poder descargarlo a través de internet.

Para la realización de los objetivos se pondrán en practica los conocimientos alcanzados en;

- **Ingeniería del software** para el análisis del proyecto.
- **Programación orientada a objetos** para la estructura y la organización del código **Java**.
- **Programación concurrente y sistemas operativos** para la gestión de las distintas hebras y la comunicación entre ellas.
- **Programación de sistemas multimedia** para poder implementar las interfaces de usuario en **Android** y poder tratar y mostrar imágenes enviadas por los dispositivos.
- **Infraestructura virtual** para poder gestionar los sistemas para realización de test y simulaciones.
- **Transmisión de datos y redes de computadores** para comprender el comportamiento del protocolo **INDI** y configurar correctamente las redes para las pruebas.
- **Diseño de Aplicaciones para Internet** para añadir código html a las interfaces de **Android** y para el desarrollo de un portal web que de difusión e información sobre la aplicación.

Por otro lado, han sido necesarios alcanzar conocimientos en otras áreas:

- **Astronomía y equipos astronómicos** para entender a los usuarios potenciales y poder acomodar la aplicación a sus necesidades.
- **Android** para conocer las herramientas que ofrece la plataforma y usar las mas adecuadas según las necesidades concretas.
- **Raspberry Pi**² para montar un servidor permanente de pruebas o acceso público para probar la aplicación
- **Latex**³ para la realización del presente documento y la ampliación de conocimientos para futuros textos científicos.
- **Git** para la gestión de versiones y la publicación de código abierto que permita a otros desarrolladores participar.

¹Paquete para el sistema operativo Android (Application Package File)

²Ordenador de placa reducida y única de bajo coste.

³Sistema de composición de textos.

2.1. Alcance de los objetivos

La aplicación móvil desarrollada debe cumplir los objetivos principales para cubrir una necesidad existente. Actualmente no existe ninguna aplicación móvil basada en **INDI** para controlar dispositivos astronómicos. Con la realización del proyecto se pretende cubrir dicha necesidad, obteniendo una aplicación estable y que será mantenida y mejorada más allá de la finalización del Proyecto Fin de Grado. Se trata de un proyecto vivo y extensible en el tiempo.

La consecución de alcanzar también los objetivos secundarios tendrá un efecto directo en la difusión de la aplicación y en la satisfacción directa de los usuarios de la misma. Por ello, se comprará una licencia de desarrollador para *Google Play* y se publicará y dará difusión en distintos canales de comunicación como la página oficial **INDI** y a través de foros y páginas web.

2.2. Interdependencia de los objetivos

El principal objetivo que debe cumplir la aplicación es el *OBJ-1*, aunque todos los objetivos son independientes excepto los objetivos secundarios *OBJ-S-1*, *OBJ-S-2* y *OBJ-S-3*. Seguir los estándares y recomendaciones de la plataforma **Android** derivará en una mayor compatibilidad con versiones antiguas del sistema operativo y un diseño de la interfaz de usuario más amigable y fácil de usar.

Capítulo 3

Planificación

3.1. Fases

El proyecto se divide en una sucesión de fases previamente establecidas que nos ayudará a estructurar, temporizar y evaluar los costes tanto económicos como humanos. Dado que el propio planteamiento del proyecto implica el uso de una serie de tecnologías como Android e **INDI**, y la necesidad de conocer el campo de la **astronomía**, se propuso el proyecto con bastante antelación ya que se preveía tener que realizar una fase de familiarización con las tecnologías y campos implicados. Esta fase es bastante extensa ya que se parte de 0.

- **Fase 0:** Planteamiento del problema.
- **Fase 1:** Familiarización con las tecnologías implicadas.
- **Fase 2:** Especificaciones del proyecto.
- **Fase 3:** Análisis y diseño.
- **Fase 4:** Implementación.
- **Fase 5:** Pruebas.
- **Fase 6:** Documentación.

3.2. Estimación de tiempos

A continuación se muestran las fases con sus actividades principales y la estimación inicial de tiempos.

■ **Planteamiento del problema:**

- Priemra reunión con el cliente.
- Descripción de los objetivos que se persiguen.
- Planteamiento de las posibles tecnologías.
- Estimación: 4 horas.

■ **Familiarización con las tecnologías implicadas:**

- Android: Generación de aplicaciones.
- Android: Generación de interfaces de usuario.
- Android: Posibles entornos para el desarrollador
- INDI: Comprensión del protocolo.
- INDI: Familiarización con la blilbioteca “*INDI for Java*”
- Familiarización con el campo de la astronomía.
- Realización de pruebas simples para estudiar la viabilidad técnica del proyecto
- Estimación: 80 horas.

■ **Especificación del proyecto:**

- Tecnologías elegidas. Entornos de trabajo.
- Recursos humanos.
- Presupuesto.
- Temporización.
- Estimación: 18 horas.

■ **Análisis y diseño:**

- Análisis de requisitos.
- Diagramas.
- Metodología de desarrollo.
- Estimación: 36 horas.

■ **Implementación:**

- Herramientas seleccionadas.

- Creación de una aplicación en Android para abrir y cerrar conexiones de red.
- Creación de una aplicación en Android para conectarse con un driver INDI.
- Creación de una aplicación en Android para poder listar todas las propiedades y dispositivos de una conexión INDI.
- Creación de una aplicación en Android para poder interactuar con las propiedades de los dispositivos de una conexión INDI.
- Creación de una aplicación en Android para poder gestionar varias conexiones INDI simultáneamente.
- Creación de interfaces de usuario específicas para propiedades concretas y dispositivos concretos.
- *Estimación:* 180 horas.

■ **Pruebas:**

- Pruebas de la aplicación en entornos simulados.
- Pruebas de la aplicación en entornos reales.
- *Estimación:* 30 horas.

■ **Documentación:**

- Documentación de la aplicación.
- Manual de usuario.
- Documentación del proyecto.
- Manual del desarrollador.
- *Estimación:* 30 horas.

3.3. Recursos humanos

Dado que el objetivo es demostrar las capacidades y competencias del alumno a la hora de afrontar un proyecto, el equipo de recursos humanos solo lo formará él, teniendo que afrontar todas las etapas del desarrollo del proyecto.

3.4. Presupuesto

Para el presente proyecto se tendrán en cuenta los siguientes costes:

- **Costes por hora de equipo humano:** En este caso son las horas dedicadas al proyecto por parte del alumno. Podemos ver que el total de horas dedicadas son 298 horas. si cuantificamos el precio de desarrollo por hora. Si estimamos el precio por hora en 25€ tenemos un coste de 7450€.
- **Costes asociados a licencias necesarias para publicar o desarrollar el software:** Dado que hemos elegido la plataforma **Android** y que basamos el proyecto en **Software Libre** no será necesario realizar ninguna inversión previa. Unicamente debemos tener en cuenta que para poder publicar la aplicación en *Google Play* debemos pagar 25€.
- **Costes asociados a los entornos de prueba simulados:** Para poder realizar las pruebas ha sido necesario comprar una *Raspberry Pi B+* que tiene un coste asociado de 35€. En ella se alojan los simuladores necesarios para testear las diferentes funciones del software.
- **Costes asociados a los entornos de prueba con equipos:** Los entornos de prueba simulados son limitados, por lo que para poder probar de forma completa el software es necesario adquirir instrumental astronómico. Por ello ha adquirido una montura, un telescopio básico y una cámara básica por 400€.
- **Costes asociados a la publicación y difusión a través de internet:** Para dar difusión y permitir descargar la aplicación sin tener que usar *Google Play* se ha desarrollado una página web cuyo coste anual de dominio y hosting asciende a 30€ al año.

Como puede observarse, el coste inicial del proyecto es de **7940€**

3.5. SCRUM

Hasta ahora hemos basado la planificación en una metodología de desarrollo clásica o *en cascada*. Esta metodología se basa en un conocimiento alto de los requisitos del sistema por parte del cliente y una estructura fija y previamente establecida.

Para el proyecto actual no podemos utilizar este tipo de metodología ya que el cliente no sabe con exactitud lo que quiere, dado que hay una parte de investigación asociada a la consecución del proyecto, lo cual implica la

revisión de los requerimientos a lo largo del proceso de desarrollo. Es por ello que se considera más idóneo el uso de una **metodología ágil** basada en iteraciones incrementales como **Scrum**.

En **Scrum** se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, **Scrum** está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

En **Scrum** un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

Debida a las características descritas, se puede aplicar al proyecto dado que en cada iteración el cliente tendrá una aplicación funcional que podrá probar y comprobar si cumple con los objetivos y que nuevos requerimientos son necesarios para, de esta forma, retroalimentar el proceso de desarrollo produciendo una nueva iteración.

En el siguiente diagrama podemos ver el proceso de desarrollo por iteraciones incrementales en **Scrum**.



Figura 3.1: Proceso de desarrollo Scrum (<http://www.qasoluciones.es/metodologia/agile>)

El cambio de metodología redefine las fases en tanto en cuanto ahora debemos repetir n veces las fases de análisis, diseño, implementación, pruebas y documentación. Las horas estimadas son las mismas ya que se repartirían entre las iteraciones. De esta forma el coste del proyecto no se ve incrementado, solo la organización de las tareas y fases de cara al desarrollo del software.

3.6. Temporización


A continuación se mostrará un diagrama de **Grantt**¹ para ilustrar la temporización de las tareas basandonos en la planificación inicial.



Figura 3.2: Diagrama de Grantt inicial

Aunque inicialmente se plantea una temporización basada en un número fijo de iteraciones, el proceso de desarrollo hace necesario modificar los planteamientos iniciales de horas, costes y temporización. Por ello, a continuación puede observarse una diagrama de **Grantt** con la temporización real.

¹Herramienta gráfica para mostrar la temporización de una serie de tareas



../images/diagrama_grantt_final.png

Figura 3.3: Diagrama de Grantt final

Capítulo 4

Análisis

4.1. Análisis de requisitos

El primer paso en el análisis de un desarrollo software es identificar los requisitos funcionales y no funcionales. Estos requisitos son los que deberá garantizar el producto final y son generados a partir de las entrevistas con el cliente y los objetivos marcados para el software.

Nuestra metodología es ágil basada en iteraciones incrementales por lo que los requisitos son analizados en cada iteración, pudiendo ser modificados según las necesidades.

4.1.1. Requisitos funcionales

Los requisitos funcionales son las características que debe satisfacer el sistema, es decir, todas aquellas funciones que debe cumplir el producto final:

- **RF-1.** Conectarse con un servidor INDI.
- **RF-2.** Gestionar conexiones (crear, editar y borrar).
- **RF-3.** Listar todos los dispositivos de una conexión INDI.
- **RF-4.** Listar todas las propiedades de un dispositivo INDI.
- **RF-5.** Tener más de una conexión INDI simultáneamente.
- **RF-6.** Mostrar un log para cada conexión.
- **RF-7.** Agrupar las propiedades por grupos.
- **RF-8.** Editar las propiedades INDI:
 - **RF-8.1.** Propiedad Blob.

- **RF-8.2.** Propiedad Switch.
- **RF-8.3.** Propiedad Number.
- **RF-8.4.** Propiedad Text.
- **RF-8.5.** Propiedad Light.

4.1.2. Requisitos no funcionales

Los requerimientos no funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste:

- **RN-1.** Las interfaces deben seguir las recomendaciones de diseño establecidas por Android.
- **RN-2.** Se deben usar las clases y elementos de interfaz recomendados para la última versión de Android y usar las bibliotecas de compatibilidad.
- **RN-3.** Controlar la hebra principal para no sobre cargarla, creando nuevas hebras en paralelo mejorando así el rendimiento.
- **RN-4.** Crear interfaces específicas para las propiedades genéricas de INDI y para dispositivos conocidos.
- **RN-5.** Utilizar licencias libres para publicar el proyecto como Software libre
- **RN-6.** Adaptar la aplicación a distintos tamaños de pantalla.
- **RN-7.** Diseñar el software para facilitar la extensibilidad de las vistas de dispositivos y propiedades.
- **RN-8.** Internacionalización de la aplicación: Mínimo inglés y castellano.

4.2. Casos de uso

Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores

4.2.1. Descripción de actores

- **Ac-1.** Usuario.
 - Descripción: Persona que utilizará la aplicación.

- Características: Es el usuario estándar de una aplicación.
- Relaciones: Ninguna.
- Atributos: Ninguno.
- Comentarios: El usuario no tiene ningún conocimiento previo.

4.2.2. Descripción casos de uso

■ **CU-1.** Añadir una conexión.

- Actores: Usuario.
- Tipo: Primario, esencial.
- Referencias:
- Precondición:
- Postcondición: La nueva conexión será añadida a la lista y guardada.
- Autor: Jaime Torres Benavente.
- Versión: 1.0.
- Propósito: Añadir una nueva conexión.
- Resumen: El usuario rellenará una serie de campos y marcará unas opciones para añadir una nueva conexión a la lista.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para añadir una nueva conexión.		
		2	El sistema muestra el formulario para añadir nuevas conexiones.
3	Usuario: Rellena los campos del formulario, marca las opciones y pulsa en el botón de añadir.		
		4	El sistema almacena la conexión y la añade a la lista de conexiones.

Tabla 4.1: CU-1. Añadir nueva conexión.

- **CU-2.** Editar una conexión.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: La conexión debe existir y estar en estado “desconectada”.
 - Postcondición: La conexión será editada y guardada.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar una conexión existente.
 - Resumen: El usuario rellenará una serie de campos y marcará unas opciones para editar la conexión.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para desplegar el menú lateral.		
		2	El sistema muestra el menú lateral con las conexiones, su estado y sus dispositivos.
3	Usuario: Pulsa el botón editar para una conexión concreta.		
		4	El sistema muestra el formulario para la edición de una conexión.
5	Usuario: edita los campos del formulario, marca las opciones y pulsa en el botón de editar.		
		6	El sistema almacena la conexión

Tabla 4.2: CU-2. Editar una conexión.

■ **CU-3.** Borrar conexiones.

- Actores: Usuario.
- Tipo: Primario, esencial.
- Referencias:
- Precondición: Las conexiones deben existir.
- Postcondición: Las conexiones serán borradas.
- Autor: Jaime Torres Benavente.
- Versión: 1.0.
- Propósito: Borrar conexiones.
- Resumen: El usuario seleccionará de entre las conexiones disponibles, una selección para que sean borradas.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para desplegar el menú superior derecho.		
		2	El sistema muestra el menú superior.
3	Usuario: Pulsa el botón para borrar conexiones.		
		4	El sistema muestra el formulario con una lista de todas las conexiones
5	Usuario: selecciona aquellas conexiones que desee borrar.		
		6	El sistema Borra las conexiones seleccionadas

Tabla 4.3: CU-3. Borrar conexiones.

- **CU-4.** Editar los ajustes.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición:
 - Postcondición: Los ajustes serán guardados.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar los ajustes.
 - Resumen: El usuario establecerá las distintas configuraciones.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para desplegar el menú superior derecho.		
		2	El sistema muestra el menú superior.
3	Usuario: Pulsa el botón para ver los ajustes.		
		4	El sistema muestra la pantalla con una lista de ajustes y su estado.
5	Usuario: edita los ajustes que considere.		
		6	El sistema guarda el estado de cada configuración.

Tabla 4.4: CU-4. Editar los ajustes.

■ **CU-5.** Conectarse a un servidor.

- Actores: Usuario.
- Tipo: Primario, esencial.
- Referencias:
- Precondición: La conexión debe haber sido añadida previamente.
La conexión debe estar desconectada.
- Postcondición: Se añaden los dispositivos de la conexión a la lista (si los hubiese).
- Autor: Jaime Torres Benavente.
- Versión: 1.0.
- Propósito: Conectarse a un servidor.
- Resumen: El usuario se conectará a un servidor.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para desplegar el menú lateral izquierdo.		
		2	El sistema muestra el menú lateral izquierdo con la lista de conexiones y dispositivos.
3	Usuario: Pulsa el botón para conetarse.		
		4a	El sistema esconde el menú lateral y realiza la conexión. A partir de ahora la conexión se mantiene en segundo plano para refrescar los dispositivos añadidos o borrados que serán listados al desplegar el menú lateral izquierdo.

Tabla 4.5: CU-4. Conterase a un servidor.

Curso alterno	
4b	Si el servidor no responde, o los datos de la conexión no son correctos, el sistema muestra una alerta para informar al usuario.

Tabla 4.6: Curso alterno de CU-5. Conectarse a un servidor.

- **CU-6.** Desconectarse de un servidor.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: La conexión debe haber sido añadida previamente. La conexión debe estar conectada.
 - Postcondición: Se borran de la lista los dispositivos (si los hubiera).
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Desconectarse de un servidor.
 - Resumen: El usuario se desconecta de un servidor.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para desplegar el menú lateral izquierdo.		
		2	El sistema muestra el menú lateral izquierdo con la lista de conexiones y dispositivos.
3	Usuario: Pulsa el botón para desconectarse.		
		4	El sistema esconde el menú lateral y realiza la desconexión. .

Tabla 4.7: CU-6. Desconectarse de un servidor.

- **CU-7.** Salir de la aplicación.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: La aplicación debe estar iniciada.
 - Postcondición: Se cierran todas las conexiones, hebras y procesos liberando todos los recursos de la aplicación.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Salir de la aplicación.
 - Resumen: El usuario cierra la aplicación explícitamente.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para desplegar el menú superior derecho.		
		2	El sistema muestra el menú superior.
3	Usuario: Pulsa el botón salir.		
		4	El sistema comprueba cada conexión y se desconecta de todas, cerrando todas las hebras. Después cierra la aplicación.

Tabla 4.8: CU-7. Salir de la aplicación.

- **CU-8. Mostrar dispositivo.**
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: La conexión debe haber sido añadida previamente. La conexión debe estar conectada.
 - Postcondición: Se listan todas las propiedades del dispositivo. Cualquier cambio en las propiedades será mostrado en la lista en tiempo real.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Mostrar las propiedades de un dispositivo.
 - Resumen: El usuario selecciona un dispositivo para mostrar la lista de sus propiedades.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa el botón para desplegar el menú lateral izquierdo.		
		2	El sistema muestra el menú lateral izquierdo con la lista de conexiones y dispositivos.
3	Usuario: Pulsa sobre el dispositivo deseado.		
		4	El sistema esconde el menú lateral y muestra una pantalla tabulada con todas las vistas especiales que tenga el dispositivo (si las tiene) más la vista por defecto con la lista de propiedades y la ayuda general de la aplicación.

Tabla 4.9: CU-8. Mostrar dispositivo.

- **CU-9.** Cambiar vista de dispositivo.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: El usuario debe haber seleccionado un dispositivo (CU-8).
 - Postcondición:
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Cambiar entre las vistas de un dispositivo.
 - Resumen: El usuario cambia de vista de un dispositivo entre las disponibles.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa sobre el nombre de la pestaña correspondiente o desliza el dedo por la pantalla.		
		2	El sistema muestra la vista correspondiente.

Tabla 4.10: CU-9. Cambiar vista de dispositivo.

- **CU-10.** Editar propiedad *text*.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: El usuario debe haber seleccionado un dispositivo (CU-8).
 - Postcondición: La propiedad es editada y enviada al servidor.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar una propiedad *text*.
 - Resumen: El usuario pulsará sobre una propiedad *text* para editarla.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa sobre una propiedad de tipo <i>text</i> .		
		2a	El sistema muestra la vista para la edición de las propiedades <i>text</i> con todos los elementos de la propiedad concreta.
3	Usuario: edita los elementos que desee y pulsa el botón de actualizar		
		4	El sistema cierra la vista de edición y actualiza la vista de la propiedad.

Tabla 4.11: CU-10. Editar una propiedad *text*.

Curso alterno	
2b	Si la propiedad es de solo lectura, el sistema muestra una alerta.

Tabla 4.12: Curso alterno de CU-10. Editar una propiedad *text*.

- **CU-11.** Editar propiedad *number*.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: El usuario debe haber seleccionado un dispositivo (CU-8).
 - Postcondición: La propiedad es editada y enviada al servidor.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar una propiedad *number*.
 - Resumen: El usuario pulsará sobre una propiedad *number* para editarla.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa sobre una propiedad de tipo <i>number</i> .		
		2a	El sistema muestra la vista para la edición de las propiedades <i>number</i> con todos los elementos de la propiedad concreta.
3	Usuario: edita los elementos que desee y pulsa el botón de actualizar		
		4a	El sistema cierra la vista de edición y actualiza la vista de la propiedad.

Tabla 4.13: CU-11 Editar una propiedad *number*.

Curso alterno	
2b	Si la propiedad es de solo lectura, el sistema muestra una alerta.
4b	Si algún valor de algún elemento editado está fuera de rango o tiene un formato erróneo, el sistema mostrará una alerta.

Tabla 4.14: Curso alterno de CU-11. Editar una propiedad *number*.

- **CU-12.** Editar propiedad *switch*.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: El usuario debe haber seleccionado un dispositivo (CU-8).
 - Postcondición: La propiedad es editada y enviada al servidor.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar una propiedad *switch*.
 - Resumen: El usuario pulsará sobre una propiedad *switch* para editarla.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa sobre una propiedad de tipo <i>switch</i> .		
		2a	El sistema muestra la vista para la edición de las propiedades <i>switch</i> con todos los elementos de la propiedad concreta.
3	Usuario: edita los elementos que desee y pulsa el botón de actualizar		
		4	El sistema cierra la vista de edición y actualiza la vista de la propiedad.

Tabla 4.15: CU-12 Editar una propiedad *switch*.

Curso alterno	
2b	Si la propiedad es de solo lectura, el sistema muestra una alerta.

Tabla 4.16: Curso alterno de CU-12. Editar una propiedad *switch*.

- **CU-13.** Editar propiedad *blob*.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: El usuario debe haber seleccionado un dispositivo (CU-8).
 - Postcondición: La propiedad es editada y enviada al servidor.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar una propiedad *blob*.
 - Resumen: El usuario pulsará sobre una propiedad *blob* para editarla.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa sobre una propiedad de tipo <i>blob</i> .		
		2a	El sistema muestra la vista para la edición de las propiedades <i>blob</i> con todos los elementos de la propiedad concreta.
3	Usuario: edita los elementos que desee y pulsa el botón de actualizar		
		4	El sistema cierra la vista de edición y actualiza la vista de la propiedad.

Tabla 4.17: CU-13 Editar una propiedad *blob*.

Curso alterno	
2b	Si la propiedad es de solo lectura, el sistema muestra una alerta.

Tabla 4.18: Curso alterno de CU-13. Editar una propiedad *blob*.

- **CU-14.** Editar propiedad *connection*.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: El usuario debe haber seleccionado un dispositivo (CU-8).
 - Postcondición: La propiedad es editada y enviada al servidor.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar una propiedad *connection*.
 - Resumen: El usuario pulsará sobre una propiedad *connection* para editarla.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa sobre una propiedad de tipo <i>connection</i> .		
		2	El sistema muestra la vista para la edición de las propiedades <i>connection</i> con un <i>switch</i> para conectar o desconectar la propiedad.
3	Usuario: pulsa sobre el <i>switch</i> para conectar o desconectar la propiedad y después pulsa en el botón actualizar.		
		4	El sistema cierra la vista de edición y actualiza la vista de la propiedad.

Tabla 4.19: CU-14 Editar una propiedad *connection*.

- **CU-15.** Editar propiedad *abort*.
 - Actores: Usuario.
 - Tipo: Primario, esencial.
 - Referencias:
 - Precondición: El usuario debe haber seleccionado un dispositivo (CU-8).
 - Postcondición: La propiedad es editada y enviada al servidor.
 - Autor: Jaime Torres Benavente.
 - Versión: 1.0.
 - Propósito: Editar una propiedad *abort*.
 - Resumen: El usuario pulsará sobre una propiedad *abort* para editarla.

Curso normal			
Actor		Sistema	
1	Usuario: Pulsa sobre una propiedad de tipo <i>abort</i> .		
		2	El sistema muestra la vista para la edición de las propiedades <i>abort</i> con un botón para abortar.
3	Usuario: pulsa sobre el botón para abortar.		
		4	El sistema cierra la vista de edición y actualiza la vista de la propiedad.

Tabla 4.20: CU-15 Editar una propiedad *abort*.

4.3. Diagrama de paquetes

Este diagrama representa la estructura lógica del sistema basado en las dependencias existentes entre sí. El paquete de **Configuración automática** depende del paquete **Administración portal** porque necesita de tareas de administración como es iniciar el servidor.



Figura 4.1: Diagrama de paquetes

4.4. Diagramas de casos de uso

Los diagramas de casos de uso representan como los diferentes actores se relacionan con el sistema para usar sus funciones. Por ejemplo, en el primer caso de uso vemos como el **desarrollador** se relaciona con el sistema para iniciar el servidor del portal, mientras, en el segundo vemos como el **usuario** se relaciona con el sistema para consultar la información de las diferentes secciones del portal.

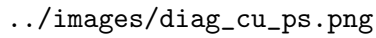


Figura 4.2: Diagrama de casos de uso: paquete Administración portal



Figura 4.3: Diagrama de casos de uso: paquete Acceso información

En el tercer diagrama vemos como el **desarrollador** se relaciona con los procesos relacionadas con las pruebas (que a su vez son dependientes entre sí); y en el cuarto vemos como también el **desarrollador** se relaciona con el sistema para realizar las tareas de configuración automática, como son el despliegue automático y el provisionamiento.



../images/diag_cu_ps.png

Figura 4.4: Diagrama de casos de uso: paquete Pruebas de software



Figura 4.5: Diagrama de casos de uso: paquete Configuración automática

4.5. Diagramas de actividad

Los diagramas de actividad sirven para representar la descomposición de un proceso en las diferentes acciones de las que está compuesto. Las actividades de consultar algún tipo de información son procedimiento secuenciales en los que la ejecución es bastante simple; pero la actividad de iniciar el servidor del portal, realizar los test unitarios, realizar los test de cobertura, usar la integración continua tienen situaciones condicionales que son los que dan lugar a cursos alternos de la ejecución. Las actividades de despliegue automático y provisionamiento además tienen también puntos de sincronización que harán que el proceso siga el mismo cauce en su ejecución.



Figura 4.6: Diagrama de actividad CU-1. Inicio automático del servidor del portal



Figura 4.7: Diagrama de actividad CU-2. Consultar información de Administración



Figura 4.8: Diagrama de actividad CU-3. Consultar información de Docencia



Figura 4.9: Diagrama de actividad CU-4. Consultar información de Gestión e Investigación



Figura 4.10: Diagrama de actividad CU-5. Consultar información de Normativa Legal



Figura 4.11: Diagrama de actividad CU-6. Realizar tests unitarios



Figura 4.12: Diagrama de actividad CU-7. Realizar test de cobertura



Figura 4.13: Diagrama de actividad CU-8. Usar integración continua



Figura 4.14: Diagrama de actividad CU-9. Usar despliegue automático

4.6. Diagrama conceptual

En el diagrama conceptual podemos ver una representación de la estructura de la implementación. A excepción de la clase **test**, todas las clases son parte de la aplicación principal (**app**) por lo que tienen una relación de composición con la misma y no tienen sentido sin esta. Las clases de **test** y **app** tienen una relación de agrupación, porque el módulo test puede realizar las pruebas sobre cualquier módulo de aplicación que sea recibido.

4.7. Otros diagramas o interfaces

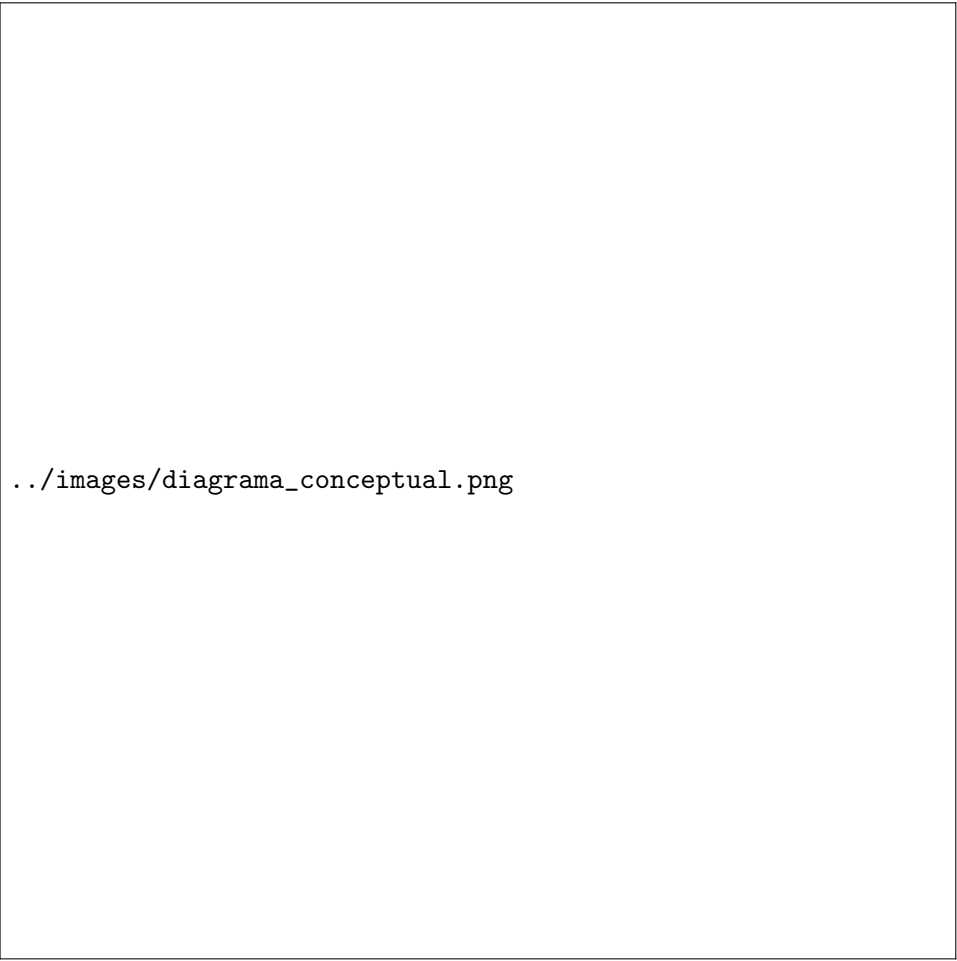
Gran parte de la implementación no va a ser de la aplicación principal en si misma, sino herramientas que se le van a agregar para obtener dife-



Figura 4.15: Diagrama de actividad CU-10. Usar provisionamiento

rentes funcionalidades que se usarán durante su desarrollo, de ahí que no se considere necesario el realizar diagramas de comunicación ni diagramas de secuencia.

En cuanto a la interfaz gráfica, no será necesaria porque todas las herramientas solo tienen modo de funcionamiento a través de terminal.

A large rectangular box containing a file path, serving as a placeholder for a conceptual diagram.

`../images/diagrama_conceptual.png`

Figura 4.16: Diagrama conceptual

