Universitat Oberta de Catalunya

# Control doméstico por voz desde Android

## Desarrollo de aplicaciones de software libre


Nombre del Autor : Manuel Alejandro Moscoso Dominguéz
Nombre del Consultor : Gregorio Robles Martinez
Nombre Tutor Externo : Oriol Palenzuela

Fecha

Este documento se publica bajo la Licencia GNU Free Documentation License (GFDL).
Léanse los términos de uso para más información.

Copyright (C)  2014 Manuel Alejandro Moscoso Domínguez.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "Licencia".

# Resumen del proyecto

El proyecto consiste en el desarrollo de un producto de Software para el sistema operativo Android que permite la gestión de equipos ODControl mediante comandos de Voz. El producto de Software debe comunicarse con equipos ODControl por lo que es fundamental entender el funcionamiento de estos equipos lo que permite definir la mejor alternativa para establecer la comunicación con los dispositivos móviles que tendrán instalada la aplicación.

Para el desarrollo del proyecto se planifico que los trabajos realizados fueran parte de iteraciones, las cuales generaron prototipos funcionales capaces de ser probados y validados para corroborar el cumplimiento de los requerimientos establecidos para el Sofware.

El proyecto esta compuesto por los proceso de análisis, diseño y desarrollo. Para cada uno de esto procesos se definieron tareas especificar que entregan como resultados artefactos UML (Diagramas) capaces de explicar cada una de las ideas desarrollas. Todos los artefactos tiene como objetivo poder encaminar el trabajo para poder observar como resultado final un producto de Software totalmente funcional que satisface los requerimientos definidos por la empresa OpenDomo.

Un aspecto importante a señalar son los conocimientos transversales adquiridos durante el desarrollo que tiene como principal característica la utilización de herramientas (IDE, Framework, Sistema de control de versiones), las cuales ayudan a la codificación, gestión y administración del proyecto.

# Tabla de contenido

# Introducción

Actualmente la tecnología se ha convertido en un recurso imprescindible para las personas, permitiendo realizar de manera más amigable diversas tareas en el transcurso de los días. Estos dispositivos electrónicos son los ejecutantes de asuntos tan cotidianos como recordar la hora de despertar, reuniones, además de la facilitación de diversas tareas existentes en nuestro día laboral o en momentos de diversión. En este contexto de nuestras vidas, podemos hablar sobre un concepto que ya hace varios años vió la luz; el cual corresponde a Ubicuidad.

La computación Ubicua se puede entender en palabras simples como la integración de distintos tipos de dispositivos informáticos a nuestro entorno, cuya principal característica es que las personas dejan de percibirlos como instrumentos en sí. En este punto no hablamos de un Notebook en particular, con procesadores de última generación, ni de un Automóvil último modelo; sino más bien de como estos motores, interruptores, microcontroladores desaparecen de la percepción del usuario y se comienzan a utilizar sin razonar pudiendo establecer nuevos objetivos.[1]

Parte de la computación Ubicua o entornos ubicuos es el área de la Domótica; la cual corresponde a la integración de tecnología al diseño y gestión de espacios específicos como una oficina, casa, habitación, etc. [2] La Domótica se puede clasificar en cinco aspectos principales; ahorro energético, seguridad, comunicaciones, accesibilidad y comodidad & confort. Estos aspectos no son excluyentes, por lo que una solución de Domótica puede involucrar a más de uno de ellos como por ejemplo, la gestión remota del encendido o apagado de luces, cierre de ventas o activación de riego; donde convergen aspectos básicos como comunicación, confort y ahorro energético. A través de equipos especiales de la Domótica se puede controlar gran cantidad de dispositivos dentro de un recinto en particular, por medio de diversas interfaces de integración, las cuales se encuentran disponibles para pc, tablets y smartphones.

Un producto de Software que pueda gestionar de manera central cada uno de estos equipos, capaces de controlar diversas acciones sobre un recinto, entrega la posibilidad a las personas de poder trabajar y operar con cada uno de ellos de manera eficiente. Otra característica del Software, debe ser la integración de instrucciones mediante el reconocimiento de Voz, permitiendo una comunicación más fluida a través de interfaces ubicuas.

Para este propósito se pretende utilizar como plataforma de desarrollo Android, la cual entrega herramientas para la creación de aplicación junto con un framework que permita además emplear componentes fundamentales para el reconocimiento de voz.

## *Objetivo General*

Desarrollar un producto de Software para equipos con el sistema operativo Android, que permita el control de dispositivos ODControl, mediante la ejecución de instrucciones reconocidas a través de la herramienta de reconocimiento de Voz que provee el Sistema operativo.

## Objetivos Específicos

Los objetivos específicos que se establecen para el proyecto son:

- Definir y almacenar información de los dispositivos ODControl para el acceso y ejecución de instrucciones. En este caso es fundamental la dirección IP del dispositivo junto con las credenciales de acceso.
- Definir y almacenar la información de los puertos de cada dispositivo ODControl con el fin de poder relacionar cada uno con los periféricos que tienen conectados.
- Definir y almacenar las acciones que se puede ejecutar sobre cada puerto de los dispositivos ODControl asociando cada una de esta a operaciones de encendido o apagado.
- Desarrollo de una interfaz que permita el reconocimiento de acciones sobre puertos mediante comandos de voz que se traduzcan en instrucciones de encendido o apagado.

## *Tecnologías asociadas*

A continuación hablaremos de las tecnología involucradas durante el desarrollo del proyecto.

## Android

Android es un sistema operativo para dispositivos móviles que pertenece a Google y se encuentra respaldado por la Open Handset Alliance[4], un consorcio de más de 50 compañías de Software, Hardware y Telecomunicaciones comprometidas en la creación de un sistema capaz de estandarizar el desarrollo en dispositivos móviles. Es importante señalar que una de las principales características de Android es que se encuentra basado en el Kernel de Linux entregando una capa de abstracción del SO operativo con los componentes de Hardware.

Tanto para usuarios y desarrolladores es fundamental conocer la arquitectura del sistema operativo para de esta manera entender mejor su funcionamiento. La arquitectura[5] de Android se puede separar en 5 pisos, siendo dos los más importantes para el desarrollo; Aplicaciones y el Framework de Aplicaciones.

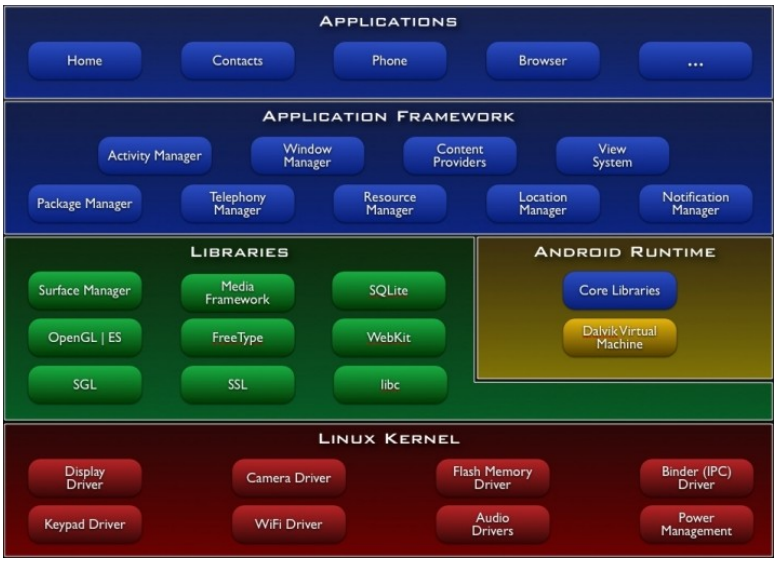


*Ilustración 1: Arquitectura del Sistema operativo Android*

En la capa de aplicaciones, encontramos todos aquellos productos de Software instalados en el dispositivo y es donde cualquier desarrollo realizado se posicionará. Esta capa es importante para entender donde se ubican los desarrollos y como podemos también utilizar recursos de otras aplicaciones en la nuestra.

El desarrollador de aplicación interactúa principalmente con la capa del Framework, la cual entrega un conjunto de paquetes (API) que permiten por ejemplo:

- Utilización de Sensores dependiendo de cuales tenga disponible el Equipo. Ejemplos de Sensores son: Orientación, GPS, proximidad, Temperatura, etc.
- Utilizar los dispositivos de conectividad de que tiene el equipo: Bluetooth, Wifi, IR, etc.
- Almacenar información para la aplicación: Datastore en Equipo o SDCard, Preferencias, Base de Datos SQLite.
- Interactuar con el usuario a través de Interfaces gráficas como: Notificaciones, recordatorios, alertas, etc.
- Interactuar con el usuario a través de Reconocimiento de Voz, imágenes, patrones, etc.

Android es un Sistema operativo abierto que podemos encontrar en un gran número de Smartphones y Tablets en el mercado. Esta característica particular permite a los desarrolladores tener acceso a un gran número de posibles usuarios en distintos escenarios; permitiendo una distribución del desarrollo bastante amplia y acceso sin ningún costo. Sumado además a las diversas herramientas de desarrollo disponibles para los Desarrolladores. Gracias a las grandes ventajas que presenta Android frente a otras alternativas en el mercado, es por lo cual se prefiere como plataforma para el desarrollo del producto de Software.

## ODControl

ODControl[3] es un producto que permite controlar a través de la red distintos tipos de dispositivos. Una gran ventaja es que no necesita de una gran programación para su instalación, sino que solo requiere su configuración. Durante este proceso, se deben definir o determinar que dispositivos están conectados a los puertos digitales y análogos que presenta.

Su principal característica, es el acceso a ODControl a través de la red (mediante su debida configuración), permitiendo acceder a través de distintos dispositivos como son el PC, Tablet u otros, generando un rápido acceso e incluso logrando realizar esta tarea remotamente fuera del perímetro donde se encuentren estos equipos (Vinculado por ejemplo a direcciones de Red públicas).

Para realizar la configuración inicial, existe una herramienta visual, donde se ejecutan las tareas de definición para cada “puerta” manualmente. En caso de no querer realizar estas actividades también existe la posibilidad de poder descargar una plantilla con configuraciones predefinidas que se adapten a las necesidades de cada usuario, como por ejemplo:

Control de iluminación, encender/ apagar luces.

- Riego, encender o apagar riego.
- Persianas.
- Etc.

Una vez finalizado el proceso de configuración correspondiente a los nombres de las puertas y que permita identificar en que lugar se encuentra conectado, se puede acceder a la interfaz de control, permitiendo su “encendido” o “apagado”.

## *Estado del Arte*

Día tras día existe mayor interés por la integración de las tecnologías que nos rodean en todos los entornos que nos desenvolvemos; es por esto que existe un gran número de empresa desarrollando productos de domótica con control ubicuo sobre todo. En este contexto y tomando en cuenta que el objetivo es el desarrollo de un producto de Software, capaz de interactuar con estos dispositivos mediante el reconocimiento de instrucciones a través de la comandos de Voz, es que podemos encontrar en el mercado soluciones como por ejemplo:

- **Ubi, The Ubiquitous computer** [6]
- **Tasker, Total Automation for Android** [7]
- **SmartThings Mobile** [8]

Es importante señalar que cada uno de estas soluciones tiene características relacionadas con la domótica y la automatización de ciertas tareas. Teniendo claro Tasker y SmartThings, corresponden a aplicaciones disponibles para su instalación en Android.  Algunos aspectos

sobre estos productos que tiene vital relevancia son:

- **Ubi,** es un dispositivo que se debe conectar a la toma de corriente junto con un acceso a conexión Wifi para su correcto funcionamiento. Lo interesante, es que se centra en el reconocimiento de instrucción de Voz, a través de los micrófonos que tiene incorporados El paradigma que plantea es una interacción directa con el dispositivo por medio de instrucciones de Voz y Preguntas; logrando de esta manera poder gestionar distintos tipos de equipos en el hogar: calefacción, alarmas, etc. **Este producto aún está en desarrollo y se encuentra en el período de pruebas Beta.**
- **Tasker**, Es una aplicación en Android y junto con la integración de un Plugin llamado AutoVoice y otros componentes externos (Dispositivos para el control de luces, interruptores, etc), permite la ejecución a través de instrucciones o comando de Voz. Las instrucciones no solamente tiene relación con operar interruptores, luces, calefacción, etc sino que involucra todo lo que se encuentre instalado y disponible en el Equipo con Android. **La solución por si sola, no entrega la posibilidad de reconocer instrucciones de Voz y menos operar equipos de domótica conectados en el hogar. Debido a que toda la integración de una solución como esa, implica hacer todo por separado y tener los conocimiento de cada uno de los componentes.**
- **SmartThings,** es una aplicación móvil que permite la gestión y ejecución de distintas tareas desde un dispositivo móvil. Esta solución esta compuesta por varios dispositivos periféricos que permite entregar seguridad, comodidad y ahorro a los usuarios, mediante su incorporación en sectores específicos dentro del hogar u oficina. Estos dispositivos pueden Encender luces, activar alarmas, detectar cierre/apertura de puertas y la gestión de cada uno de estos, a través del Dispositivo móvil. Por si sola, esta aplicación no permite gestionar de manera inteligente el control sobre los equipos en el hogar y no tiene incorporado el reconocimiento de instrucciones a través de la Voz.

El siguiente cuadro comparativo muestra las principales características a evaluar en los productos que se encuentran en el Mercado y agrega el producto de Software que resulta de este proyecto.

| | **Ubi** | **Tasker + Otros** | **SmartThing** | **Producto de Software** |
|---|---|---|---|---|
| Instrucción Por Voz | Si | Si | No | Si |
| Aplicación Móvil | No | Si | Si | Si |
| Periféricos de Control | S/I | No | Si | Si, Equipos ODControl |

- S/I: Sin información.

## *Estructura del Documento*

El documento tiene como objetivo entregar la información relacionada con el desarrollo de un producto de Software el cual se divide en la siguientes secciones:

- **Portada y Licencia del Documento.**
- **Resumen**
- **Tabla de Contenido**
- **Introducción:** En esta sección encontramos una instrucción al contexto del proyecto junto con los objetivos, estado del arte o soluciones del mercado, junto con las principales tecnologías asociadas.
- **Implementación:** En esta sección encontramos la información relacionada con el desarrollo del producto de Software que puede ser dividida en las siguiente tres secciones:
  - **Diseño**
  - **Desarrollo**
  - **Pruebas**
- **Conclusiones:** En esta sección encontramos las principales conclusiones del proyecto, junto con las metas a futuro en torno al producto.
- **Anexos:** En esta sección se encuentra información relevante y materiales complementarios a la sección de Implementación.

# Implementación

A continuación hablaremos de los procesos involucrados en el desarrollo del producto de Software.

## *Análisis*

En esta sección se describe todo el proceso de análisis realizado para el desarrollo del producto de Software; en el cual se identifican los requerimientos, funcionalidades, alcances y limitaciones que tendrá el sistema.

### Requerimientos

En base a los objetivos específicos del proyecto se desglosan las necesidades que debe suplir el sistema como producto de Software, las cuales son:

- El sistema debe almacenar la información de los Equipos ODControl como por ejemplo; dirección IP, nombre, credenciales de acceso, etc. Esto es con la finalidad de poder identificar cada uno de los equipos ODControl que se pretende controlar.
- El sistema debe almacenar la información de los puertos pertenecientes a los equipos ODControl como por ejemplo; nombre del puerto, etiqueta de este, acciones sobre el puerto. Esto es con la finalidad de poder identificar cada puerto utilizado dentro de un equipo ODControl y poder relacionarlo con un aparato o periférico conectado (Lámparas, luces, riego, persianas, etc).
- El sistema debe brindar la facilidad al usuario de poder conectarse directamente a un equipo ODControl; con la finalidad de poder establecer una conexión segura con el equipo para luego poder ejecutar instrucciones sobre sus puertos.
- El sistema debe ejecutar instrucciones de encendido o apagado sobre los puertos de un Equipo ODControl mediante el reconocimiento de estas a través de comandos de Voz; con el propósito de poder interactuar mediante la Voz con los dispositivos ODControl.

### Especificación de Requerimientos

Los requerimientos presentado en la sección anterior puede ser clasificados en tres categorías correspondientes a:

- Registrar información de los Equipos
- Registrar información de los Puertos de los Equipos
- Ejecutar instrucciones

La siguiente tabla identifica los requerimientos específicos y las funciones para suplirlos.

| ID | Requerimiento | Función | Categoría | Detalle y Limitaciones |
|---|---|---|---|---|
| R1 | Registrar información de los Equipos | | | |
| 1.1 | | Crear un equipo | Evidente | Debe ingresar la información del equipo |
| 1.2 | | Guardar información | Oculta | Guardar información de manera persistente |
| 1.3 | | Recuperar información | Oculta | Recuperar información del medio de almacenamiento persistente. |
| 1.4 | | Actualizar un equipo | Evidente | Debe ingresar la nueva información del equipo |
| 1.5 | | Actualizar información | Oculta | Actualizar la información en el medio de almacenamiento persistente. |
| 1.6 | | Eliminar un equipo | Oculta | Eliminar información del medio de almacenamiento persistente. |
| R 2 | Registrar información de los Puertos de los Equipo | | | |
| 2.1 | | Crear un puerto | Evidente | Debe ingresar la información del puerto perteneciente a un equipo |
| 2.2 | | Guardar información | Oculta | Guardar información de manera persistente |
| 2.3 | | Recuperar información | Oculta | Recuperar información del medio de almacenamiento persistente. |
| 2.4 | | Actualizar un puerto | Evidente | Debe ingresar la nueva información de puerto |
| 2.5 | | Actualizar información | Oculta | Actualizar la información en el medio de almacenamiento persistente. |
| 2.6 | | Eliminar puerto | Oculta | Eliminar información del medio de almacenamiento persistente. |
| R 3 | Ejecutar instrucciones | | | |
| 3.1 | | Reconocer comando de Voz | Evidente | Especificar un comando de voz al dispositivo. |
| 3.2 | | Identificar etiqueta | Oculta | Identificar si el comando de voz tiene una de las etiquetas que representan los puerto de un equipo. |
| 3.3 | | Identificar acción | Oculta | Identificar si el comando de voz tiene alguna acción de las que se encuentra definidas para el puerto. |
| 3.4 | | Ejecutar instrucción | Oculta | Realizar una conexión segura al equipo para luego ejecutar una instrucción **set on u off** sobre un puerto en particular. |
| 3.5 | | Visualizar resultado | Evidente | Actualizar la interfaz según el resultado del reconocimiento |

## Atributos Generales

El producto de Software debe presentar algunos atributos generales mediante los que se pretende garantizar su calidad y rendimiento. Los atributos generales pueden ser relacionados a requerimientos y/o funciones específicas definidas para el sistema.

La siguiente tabla muestra los atributos generales para el producto de Software y su relación con los requerimientos determinados en las secciones anteriores.

| ID | Atributo | Detalle |
|---|---|---|
| AT1 | Facilidad de uso | El sistema debe tener como característica principal la claridad para manejar todas sus opciones. La idea principal es poder navegar a través de la aplicación de manera simple y fluida entregando opciones claras y ayuda para ser guiado a través de este . |
| Requerimientos/Funciones | | 1.1, 1.4, 2.1, 2.4, 3.1, 3.5 |
| AT2 | Tiempo de Respuesta | El sistema debe presentar un corto tiempo de respuesta, siendo el principal potencial el acceso rápido a la información y ejecución de instrucciones remotamente en equipos. |
| Requerimientos/Funciones | | 1.2, 1.3, 1.5, 2.2, 2.3, 2.5, 3.5 |
| AT3 | Seguridad | El sistema almacena de acceso a dispositivos ODControl por lo que que la exposición indebida no esta permitida. |
| Requerimientos/Funciones | | 1.2, 1.3, 2.2, 2,3 |

## Casos de usos

Corresponden a escenarios puntuales donde los actores interactúan con el sistema, con la finalidad de suplir cada uno de los requerimientos mencionados anteriormente. Cada uno de los escenarios planteados en los casos de uso, deben suplir uno o más requerimientos establecidos para el producto de Software. A través de UML se puede realizar una representación gráfica de todos los casos de uso definidos para el proyecto.

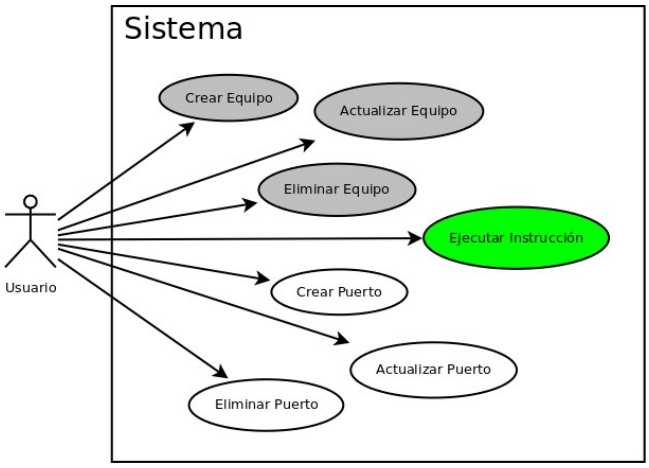


*Ilustración 2: Diagrama de Casos de Usos.*

## Alcances y limitaciones

En esta sección se detallan los alcances y limitaciones que se deben tener presentes para el desarrollo del producto de Software y lograr su funcionamiento de manera exitosa.

- **Operatividad de dispositivos ODControl:** La ejecución remota de instrucciones a través del reconocimiento de esta mediante comandos de voz y su éxito, tiene como **requerimiento principal que los equipos ODControl deben estar disponibles y operativos**. Esto quiere decir que los equipos deben estar completamente funcionales, debido a que mediante la aplicación no se puede acceder a su configuración para verificar su estado y posible corrección de algún inconveniente que este pueda tener.
- **Conectividad a la Red y/o Internet :** El acceso a los equipos ODControl es mediante la Red, por lo que, de esta misma forma se ejecutan las instrucciones a través de la aplicación. **Es de carácter fundamental que tanto los equipos ODControl y el dispositivo móvil con la aplicación tengan conexión a la Red** (Privada o Pública) y que entre estos equipos se puedan “Ver” a través de ella.

## *Diseño*

En esta sección se describe todo el proceso de diseño ejecutado para el desarrollo del producto de Software, en el cual se identifican los diagramas de solución propuesta, de clases, base de datos, arquitectura y como interactúan los paquetes.

## Solución propuesta

En el siguiente diagrama, se puede apreciar las principales características que debe presentar el software para cumplir con los objetivos establecidos. Es importante resalta, que el producto de Software, debe permitir almacenar información de los dispositivos, reconocer las instrucciones de Voz, junto con poder ejecutar estas sobre los equipos ODControl a través de una petición GET del protocolo HTTP.

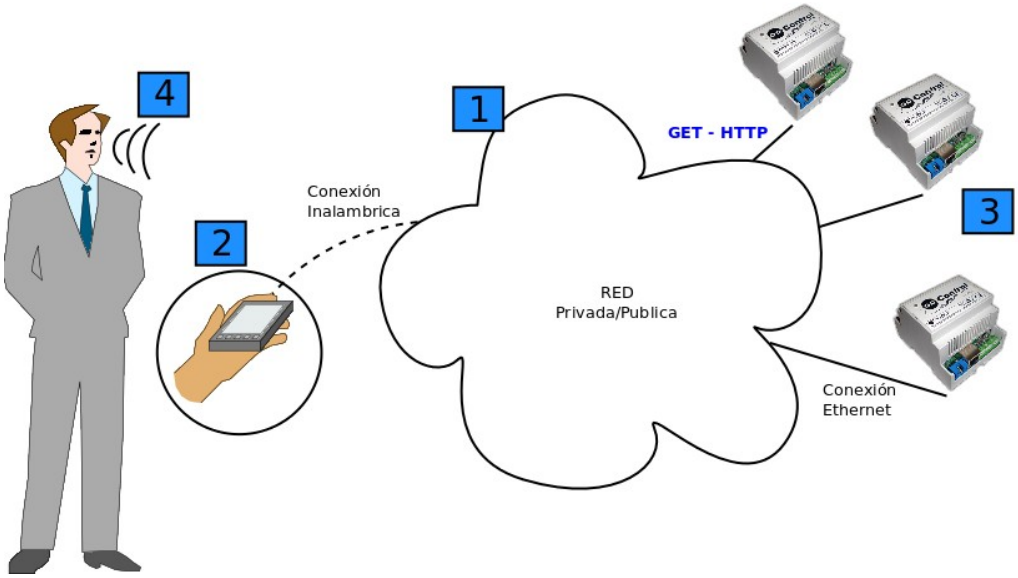


*Ilustración 3: Diagrama de Solución propuesta*

1. **RED Privada/Publica:** Un aspecto importante es que tanto los equipos ODControl y el dispositivo con la aplicación en Android estén conectados a una RED.
2. **Aplicación Android:** La aplicación para poder interactuar con los equipos ODControl debe poder almacenar de manera persistente datos de estos equipos. Además es fundamental tener en cuenta que la aplicación debe poder guardar la información de más de un equipo; con el objetivo de poder operar a través de instrucciones de voz a cada uno de estos, los cuales puede estar ubicados físicamente en distintos lugares.
3. **Equipos ODControl:** Los equipos ODControl, se encuentran conectados mediante conexión Ethernet y a través de puerta digitales o análogas a los otros equipos que se puede manipular. La configuración de estos equipos debe estar completa antes de poder utilizar la aplicación, debido a que la aplicación no realiza tareas de configuración de equipos ODControl.

4. **Comandos por Voz:** El reconocimiento de voz se realiza mediante la utilización de la herramienta provista por Android de manera nativa. Las acciones que se pueden realizar sobre cada puerta del Odcontrol, también son configuradas a través del reconocimiento de voz, siendo esta una manera de asegurar que el usuario defina correctamente mediante que acción ejecuta una instrucción determinada. Las acciones se clasifican como ON o OFF.

Estos aspectos son fundamenta para entender cada uno de los pasos que fueron realizados durante el proceso de diseño e implementación del producto de Software.

## Arquitectura de la Solución

El desarrollo de la aplicación para Android fue diseñado pensado en una arquitectura de tres capas. Esta arquitectura tiene la particular de separar en capas y definir las responsabilidad de cada una de ellas dentro del funcionamiento del producto de Software.

Las capas que se pretende implantar son:

- **Presentación:** Esta capa corresponde a la que el usuario ve y con la cual interactúa directamente.
- **Negocio:** Esta capa contiene todas las reglas del negocio, gestiona todas las funcionalidades y responde a la capa de presentación.
- **Datos:** Esta capa se encarga de gestionar y almacenar la información relacionada con el producto de Software.

El siguiente diagrama muestra la arquitectura de tres capas y los componentes de la solución propuesta que encontraremos en cada uno de ellos.

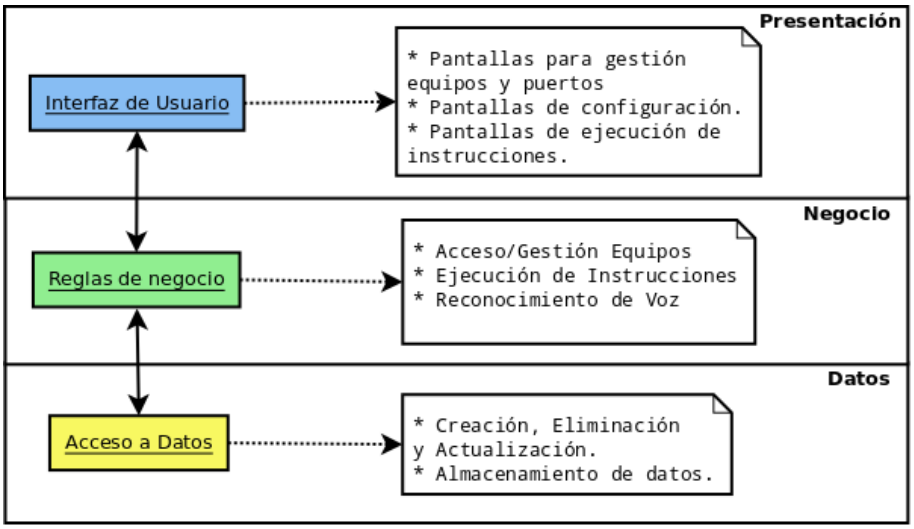


*Ilustración 4: Arquitectura de solución de tres capas*

## Diagrama de Clases

Este diagrama describe gráficamente las especificaciones de las clases del software donde encontraremos:

- Clases; atributos y métodos.
- Información sobre los tipos de atributos.
- Visibilidad.
- Dependencias.

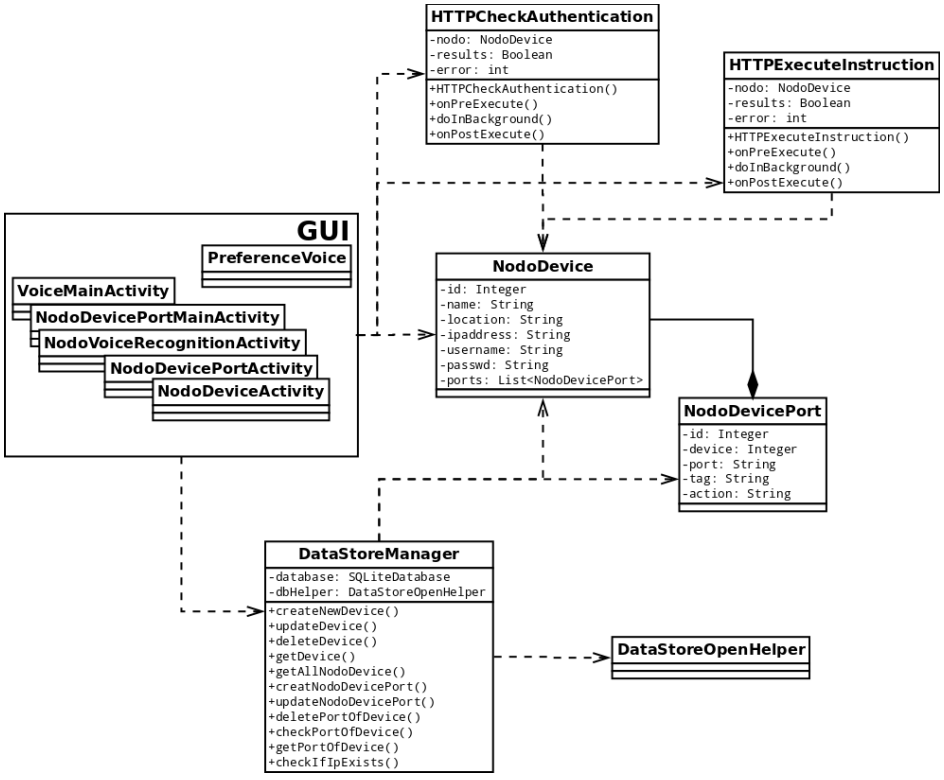


*Ilustración 5: Diagrama de clases de diseño*

En el diagrama de clases, se puede apreciar un conjunto de clases relacionadas con la interfaz gráfica de usuario (***Graphic user interface – GUI***), la cual corresponde a clases heredadas de la clase ***Activity*** de Android. Estas clases serán las encargadas de recibir las acciones del usuario y con las que se interactúa directamente; siendo esta la razón del porque no se especifican los atributos de cada clases, debido a que depende de los ***widgets*** definidos en los archivos XML de Android.

Algunos ejemplo de widgets con los que interactúa el usuario son: botones - ***Button***, campos de texto - ***TextView***, campos de texto editables ***EditText***, etc.

## Diagrama de Paquetes

El diagrama de paquetes, es un artefacto que se puede crear mediante la utilización de UML, tiene como objetivo mostrar como un producto de Software o sistema divido en agrupaciones lógicas y la relación funcional existente entre cada una de ellas. Los paquetes, por lo general, están compuestos por una o más clases, identificadas en el diagrama de clases.

A continuación se presenta un diagrama con los paquetes que existen en el producto de Software junto con situarlos en las capa de la arquitectura que corresponde. En la ilustración 33, se encuentra el diagrama de paquetes, con la agrupación lógica de las clases.

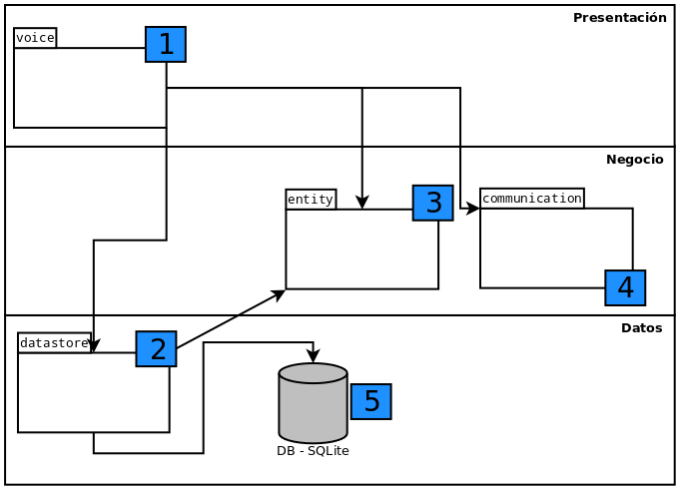


*Ilustración 6: Diagrama de paquetes distribuido por capas*

A continuación se describen los cinco componentes del diagrama:

1. **Paquete voice:** En este paquete se encuentran todas las Clases que corresponde a Activities con las cuales el Usuario interactúa con la aplicación.
2. **Paquete datastore:** Todas las clases que gestionan la creación, actualización, obtención y eliminación de la información de la aplicación relacionada con los equipos ODControl.
3. **Paquete entity:** Agrupa todas las clases que permiten tener instancias de los equipos y los puertos. Los puertos corresponden a las posibles conexiones que tiene un ODControl y que se puedan utilizar.
4. **Paquete communication:** Es la clase encargada de realizar la comunicación entre la aplicación y los equipos ODControl a través de la RED.
5. **Componente DB – SQLite:** A través de la librería SQLite y la creación de una base de datos se almacenará la información de la aplicación.

## Diagrama de componentes

El diagrama de componentes es un artefacto que se puede crear mediante la utilización de UML que tiene como objetivo mostrar un sistema o producto de Software divido en componentes, con el fin de representar las relaciones o dependencias que existe entre ellos.

El siguiente diagrama muestra los componentes identificados en la solución de Software, junto con resaltar los principales componentes del framework de Android que son utilizados para el desarrollo de la implementación.

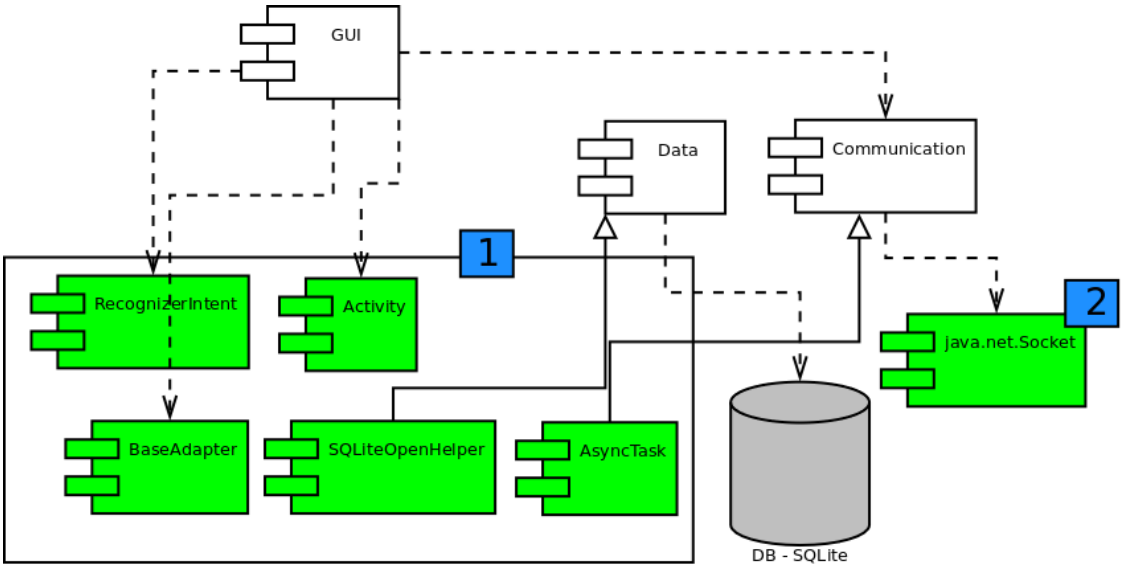


*Ilustración 7: Diagrama de componentes*

A continuación se detallan la importancia de las dos secciones identificadas con números en el diagrama.

1. **Clases de Android:** Existen varios componentes de Android que son utilizados en la mayoría de las clases que permiten por ejemplo: trabajar con recursos definidos en archivos xml, cargar layout xml, crear mensajes de alerta, ventanas emergentes, etc, pero a continuación se detallan las más relevantes para el funcionamiento general de la aplicación.
    1. **Activity**: Permite crear instancias en las que interactúa con el usuario. Cada una de las actividades (Activities) que se desarrollan corresponde a una “pantalla” con la cual interactúa el usuario.
    2. **BaseAdapter**: Creación de listas personalizadas. De esta manera se define el contenido de cada item dentro de un ListView.
    3. **SQLiteOpenHelper**: Gestiona el acceso a la base de datos, creación de esta, y la actualización.

4. **RecognizerIntent**: Permite el reconocimiento de Voz dentro de la aplicación.
5. **AsyncTask**: Ejecución de tareas en segundo plano, con la posibilidad de poder ejecutar actualizaciones de la interfaz de antes, durante y después de una tarea específica. En este caso utiliza para el manejo de Sockets para la conexión con otros equipos.

2. **Clases de Java**: Para la implantación del funcionamiento de funcionamiento de todos las clases, se usan variadas clases de Java como por ejemplo: List, ArrayList, Arrays, HashMap, etc. En este contexto es importante señalar que una clase fundamental utilizada para el funcionamiento de la comunicación con los equipos ODControl corresponde a java.net.Sockets.

## Diagrama de la Base de datos

El diagrama físico de una base de datos corresponde a una representación gráfica de las tablas, sus campos y la relación que existe entre ellos. El siguiente diagrama muestra las dos tablas que permiten almacenar información sobre los equipos y los puertos que pertenecen a cada uno de ellos.

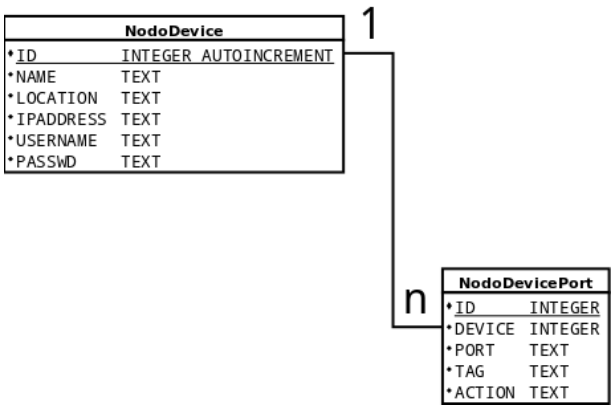


*Ilustración 8: Diagrama/Esquema físico de la base de datos*

**En el diagrama se puede apreciar la relación 1 es a n entre las tablas que representa, pudiendo ser interpretado como *"un equipo puede tener n puertos configurados"***

## *Herramienta para el desarrollo*

A continuación veremos la información relevante relacionada con las herramientas utilizada para el desarrollo del producto de Software.

## Android SDK

Android ofrece a la comunidad herramientas que permite desarrollar aplicaciones, y dentro de esta se encuentra el **ADT - Android Developer Tools**. Esta herramienta esta disponible para la comunidad bajo una licencia open-source y se puede utilizar en distintos Sistemas operativos.

El SDK de Android es el componente que entrega un **conjunto de librerías que conforman la API** junto con las herramientas de desarrollo necesarias. Dentro de estas herramientas se encuentra el **ADT** el cual corresponde a un ***plugin*** o complemento que se agrega al IDE Eclipse. Este complemente entrega lo necesario para tener el entorno de desarrollo de aplicaciones.

Dentro de sus principales características, además de la posibilidad de crear aplicaciones, es tener las herramientas para depurar, crear paquetes y hacer pruebas de las aplicaciones desarrolladas. Por otra parte el ambiente de desarrollo presenta algunas característica importantes tales como:

- Herramientas gráficas para la creación de Interfaces de Usuario.
- Opciones de desarrollo como; depuración sobre USB y visualización de estado de CPU/GPU del equipo entre otras cosas.
- Desarrollo en Hardware Real y en Equipos virtuales (AVD).
- Herramientas de depuración.
- Desarrollo nativo en C y C++.

Para agilizar el desarrollo, Android entrega desde su sitio web oficial una versión de Eclipse con el plugin- ADT ya integrado.

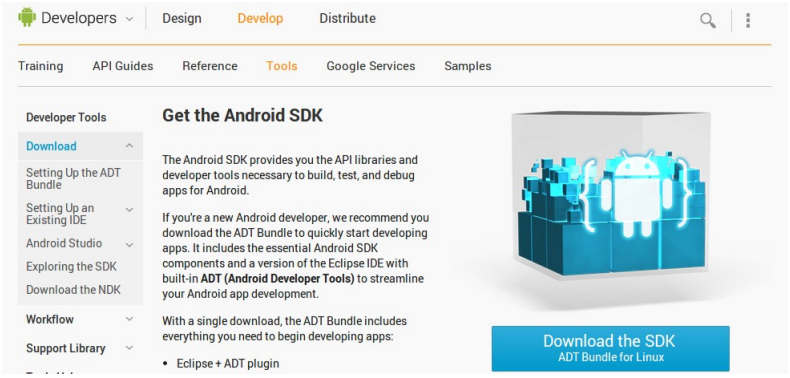


*Ilustración 9: Android Developers Web Site - Descarga de ADT para comenzar el desarrollo.*

## Github – Git como herramienta de Control de Versiones

Los sistemas de control de versiones son herramientas que permiten gestionar los trabajos sobre el código fuente de un producto de Software. Son utilizada para el manejo de contribuciones, la gestión de la mezcla de estas y junto con la creación de nuevos archivos

Dentro de las principales funcionalidades que entregan estas herramientas son:

- Almacenar distintas versiones de los archivos.
- Permite la generación de ramas. Estas son utilizadas por los desarrolladores para crear versiones nuevas.
- Automatización del proceso de unión. Esta tarea es realizada cuando se quiere unir el trabajo de varios desarrolladores sobre por ejemplo el mismo archivo.

La versatilidad de funcionalidades de los sistemas de control va de la mano con la herramienta utilizada y como gestiona el control sobre las versiones. Es por esta razón que existe CVS, SVN y GIT los cuales ofrecen distintas caracterizas para el cumplimiento de las tres funcionalidades mínimas que debe entrega este tipo de herramientas.

Hoy podemos encontrar sitios que entregan la herramienta de control de versiones junto con otros servicios que permiten facilitar y potenciar el proyecto. La utilización de estos sitios permite no destinar recursos ni tiempo para la configuración de esto servicios, logrando dedicar una mayor cantidad de tiempo al desarrollo del Software. El sitio **Github**[9] corresponde al sitio que aloja el proyecto que ofrece la herramienta **Git** como control de versiones. Además incluye Wiki, herramientas para obtener estadísticas y una plataforma para el control de fallos.

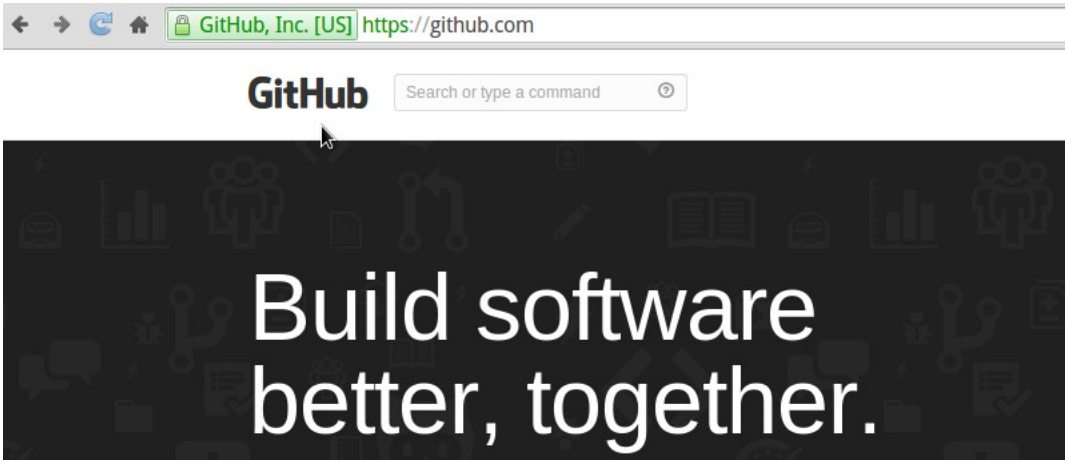


*Ilustración 10: GitHub - Git como herramienta para el Control de Versiones - Usuario odbelix.*

## *Previo al Desarrollo*

A continuación se entrega la información relacionada con las consideraciones tomadas en cuenta para comenzar con el desarrollo del producto de Software.

## Producto de Software en Android

El desarrollo de un producto de software en la plataforma Android genera como resultado, una Aplicación capaz de ser operada en un dispositivo móvil la cual se define como un grupo de una o mas ***Activities*** libres o relacionadas entre sí agrupadas en un solo archivo de empaquetado con el sufijo .apk. Los principales componentes presentes en una aplicación son:

1. **Interfaz gráfica de Usuario:** Android provee la clase Activity como componente de la API capaz de implementar interfaces de usuario. Es importante señalar sobre las ***Activities*** que:
    1. Todos los componentes visuales corresponde a objetos derivados de la clase *View* de Androd. La definición de estos componentes y su orden dentro de una “pantalla” pueden ser definidos a través de código fuente en archivo XML, también llamados ***layouts***.
    2. Todas las Activities tiene un ciclo de vida que define los distintos estado por los que puede pasar. La gestión de ***Activities*** es realizada mediante pilas y cuando se lanza una nueva ***Activity***, esta queda en la cima y las ***Activities*** previas, que estaban en ejecución, solo pueden volver a subir en la pila cuando la nueva ***Activity*** deje de existir.
    3. La comunicación entre ***Activities*** es realizadas mediante objetos de la clase Bundle.
2. **Persistencia de la Aplicación:** Para el manejo de la persistencia de la aplicación se utilizaron dos componentes de la plataforma de Android:
    1. **Preferences**: Se utilizaron las clases relacionadas con la gestión de preferencias para poder almacenar las acciones que presentan ON y OFF.
    2. **SGBD SQLite:** Para almacenar la información de los equipos y los puertos que tiene estos configurados.
3. **Comunicación con equipos:** La ejecución de instrucciones es a través del protocolo HTTP mediante el método GET. Para realizar esta tarea se utilizo la Clase Socket de Java.

## Versiones de Android Soportadas

Para comenzar el desarrollo del producto de Software es fundamental determinar que versiones de Android podrán soporta la aplicación. Para definir la versión mínima que sera soportada es importante conocer información sobre que versiones tiene los actuales equipos en el mercado, la cual se encuentra en el sitio web de Android. El sitio proporciona datos estadísticos sobre las versiones de Android de los equipos que descargan aplicaciones a través del ***Google Play***.

Los datos son tomados por un periodo de siete días y son actualizados periódicamente siendo el ultimo registro de información el día 1 de Mayo de 2014. La siguiente imagen muestra un gráfico con la distribución de los equipos por cada versión.

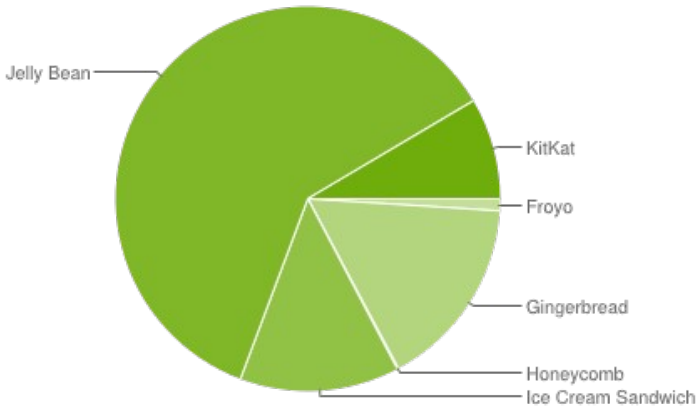


*Ilustración 11: Gráfico de Distribución de versiones de Android con fecha 1 de Mayo de 2014.*

La siguiente tabla muestra el resumen de la información que se muestra en la imagen donde se puede apreciar:

- No existe registro de las versiones anteriores a la 2.2. Eso es debido a que no tiene soporte para la aplicación Google Play.
- El Nivel de la API en algunos casos no es continuo debido a las variaciones en versiones de Android.
- Más del 50% de los dispositivos con Android se encuentra en niveles iguales o sobre el API 16. Este valor define como candidato a valor mínimo de la API a ser definido en la aplicación.

| Versión | Nombre clave | Nivel API | Porcentaje |
|---|---|---|---|
| 2.2 | Froyo | 8 | 1.0 % |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 16.2% |
| 3.2 | Honeycomb | 13 | 0.1% % |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 13.4 % |
| **4.1.x** | **Jelly Bean** | **16** | **33.5 %** |

| | | | |
|---|---|---|---|
| **4.2.x** | | **17** | **18.8 %** |
| **4.3** | | **18** | **8.5 %** |
| **4.4** | **KitKat** | **19** | **8.5%** |
| | | **Total** | **100%** |

## Hardware utilizado

Para el desarrollo y ejecución de pruebas fue necesario tener dos componentes de Hardware. Por un lado es un Smartphone, Galaxy S4 para instalar las distintas versiones del Software durante el proceso de Desarrollo y pruebas junto con una placa Arduino Mega (más un Shield Arduino Ethernet) para simular las condiciones de un dispositivo ODControl instalado en el Hogar.

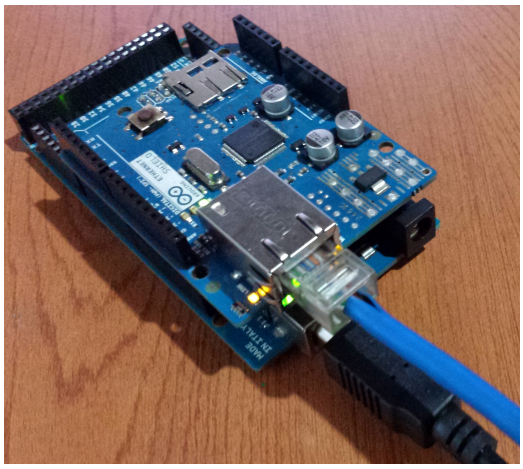

*Ilustración 12: Arduino Mega + Shield Arduino Ethernet.*

Para poder trabajar con Arduino y poder simular un dispositivo similar a ODControl se debe utilizar Domino OSE. Este producto de Software es un firmware que se instala en la Placa para hacerla compatible con dispositivos ODControl lo que permite simular la ejecución de instrucciones SET ON/OFF sobre el Dispositivo.

Para la habilitación de la Placa, mediante la instalación de Domino OSE, se realizaron los siguientes pasos:

1. Descarga de código de Domino OSE.
2. Se instalaron los siguientes paquetes en Elementary OS – Luna/Release 0.2 con el objetivo de poder instalar el código de Domino OSE sobre la placa de Arduino Mega.
    1. Arduino [11]
    2. arduino-core
3. Se edita el código definiendo la configuración de Red que tendrá el equipo.
4. Luego se Carga en la Placa.
5. Se conecta el equipo a la Red mediante un cable Ethernet.

Las siguientes imágenes muestran el IDE mediante el cual se realizar la edición y carga del código a la placa, junto con la interfaz de usuario que presenta Domino OSE. Un aspecto importante a tener en cuenta es que algunos recursos de la interfaz son obtenidos mediante acceso a internet (Archivos css y js). En caso de no tener este tipo de conexión es posible no visualizar la misma GUI.

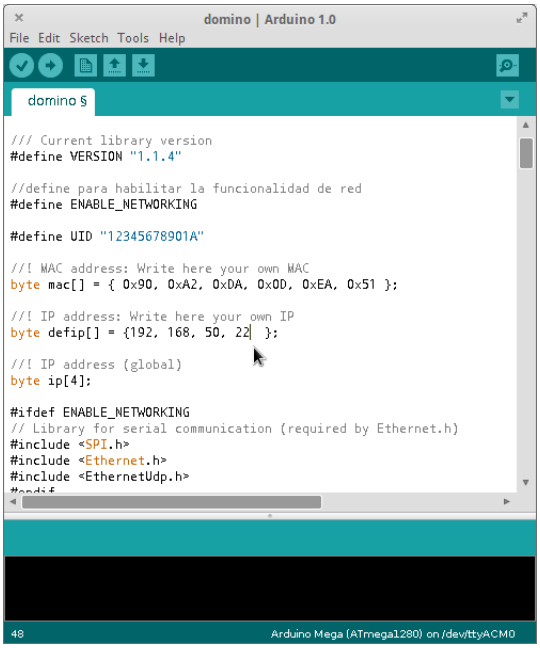


*Ilustración 13: Arduino IDE y cambio de configuración de Red para equipo.*

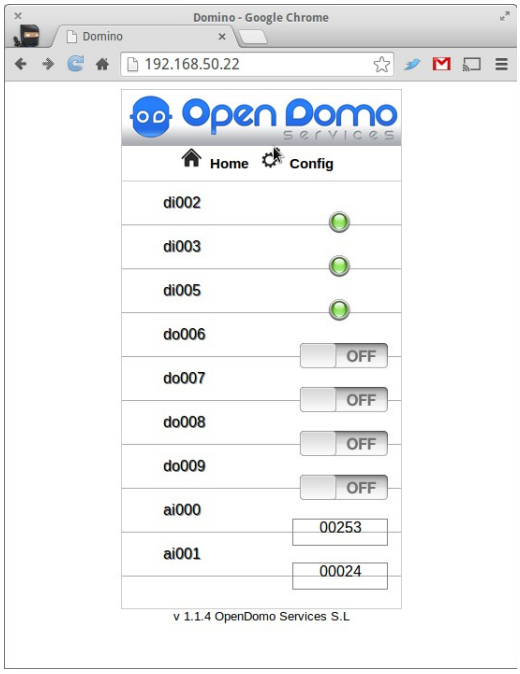


*Ilustración 14: Interfaz gráfica de Usuario de Domino OSE sobre un Arduino Mega + Shield Ethernet conectado a la Red con la dirección IP 192.168.50.22*

## *Desarrollo de Software*

A continuación se entrega la información relevante sobre el proceso de desarrollo y pruebas del producto de Software.

## Almacenamiento de Información SQLite

El desarrollo de la capacidad de almacenar información de manera persistente fue realizado mediante la utilización de una alternativa que ofrece la API de Android; correspondiente a ***SQLite***[12]. Para la creación de la base de datos y gestión de esta se tomaron en cuenta los siguientes consideraciones:

1. Se puede crear una o más bases de datos dentro de una aplicación y tiene la característica que todas las clases pueden acceder a ella sin tener que definir permisos de accesos.
2. Para la creación de una base de datos se utiliza una subclases heredada de la clase SQLiteOpenHelper de Android. A través de esta clase se reescribir el método onCreate para la creación de la base de datos.
3. Al tener una instancia de la subclase definida, se utiliza los métodos getWritableDatabase () y getReadableDatabase () para escribir y leer en la base de datos. Ambos métodos entregan un objeto SQLiteDatabase que representa la base de datos de la aplicación.

   Se define el paquete **cl.mamd.datastore** el cual contiene las clases:

   - **DataStoreOpenHelper:** Clase heredada de **SQLiteOpenHelper** para la creación de la base de datos.
   - **DataStoreManager:** Clase que implementa la capa de persistencia de la aplicación. Esta clases es la encargada de la gestión de la base de datos junto con la creación, actualización y eliminación de la información de equipos y puertos.

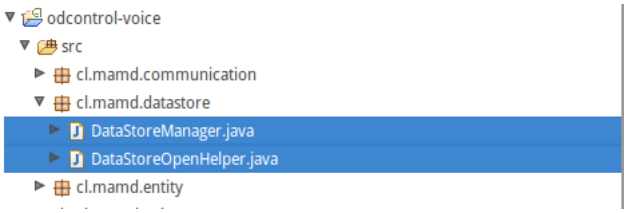


*Ilustración 15: Contenido de paquete cl.mamd.datastore*

A continuación se detalla el contenido con mayor relevancia en ambas clases que tiene relación con la creación y gestión de la base de datos.

```
private static final String TABLE_CREATE =
                "CREATE TABLE " + TABLE_NAME_NODODEVICE + " (" +
                "ID" + " integer primary key autoincrement, " +
                "NAME"  + " TEXT, " +
                "LOCATION" + " TEXT, " +
                "IPADDRESS" + " TEXT, " +
                "PASSWD" + " TEXT, " +
                "USERNAME" + " TEXT);";
```

Variable de la clase **DataStoreOpenHelper** que contiene la sentencia SQL que permite la creación de la tabla para almacenar información de equipos

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(TABLE_CREATE);
    db.execSQL(TABLE_PORT_CREATE);
}
```

Sobrescribir (@Override) el método **onCreate** de la clase **DataStoreOpenHelper** para crear las tablas en caso que no existan.

```
public DataStoreManager(Context context) {
    dbHelper = new DataStoreOpenHelper(context);
}
```

Constructor de la clase **DataStoreManager** en la cual se instancia la clase **DataStoreOpenHelper**

```
public void openDataBase() throws SQLException {
    database = dbHelper.getWritableDatabase();
}
```

Método de la clase **DataStoreManager** que abre la base de datos. En este punto se ejecuta el método **getWritableDatabase()** el cual entrega una instancia de la base de datos para poder escribir en ella (INSERT, UPDATE, etc.)

```
public boolean updateDevice(ContentValues values,String ipaddress){
       long result = this.database.update("nododevice",
          values,"IPADDRESS='"+ipaddress+"'",null);
       if ( result == -1 ){
              return false;
       }
       else {
              return true;
       }
}
```

Método de la clase **DataStoreManager** que permite actualizar la información de un equipo en particular. Para esto, a través de los parámetros, se entrega los valores a actualizar junto con la dirección IP del equipo para identificarlo en la base de datos.

La clase **DataStoreManager** es la que se incluye en todas las clases de la aplicación que necesitan obtener o gestionar información de la base de datos.

El siguiente código muestra los pasos a seguir para obtener la información desde la base de datos correspondiente a todos los equipos registrados.

```
private DataStoreManager dsm;

/**
* Codigo
*
*/

dsm = new DataStoreManager(this);
dsm.openDataBase();
this.values = dsm.getAllNodoDevice();
```

```
public List<NodoDevice> getAllNodoDevice() {
    List<NodoDevice> nodos = new ArrayList<NodoDevice>();
    Cursor cursor = database.query("nododevice",
                this.allColumnsDevice, null, null,
                 null,null, null);

    cursor.moveToFirst();

     while (!cursor.isAfterLast()) {
        NodoDevice device = this.cursorToNodo(cursor);
        nodos.add(device);
        cursor.moveToNext();
    }
    cursor.close();
     return nodos;
}
```

Además la clase **DataStoreManager** permite realizar acciones de UPDATE, INSERT, SELECT y DELETE a través de métodos implementados para la gestión de equipos y puertos.

## Almacenamiento de información con Preferences

La aplicación también utiliza las preferencias o ***Preferences***[13] para guardar información relacionada con el reconocimiento de Voz.

Para realizar el reconocimiento de voz se definieron “acciones”, que debe ser mencionadas en el comando de voz, las cuales son reconocidas e interpretadas por la aplicación para luego ser traducidas en instrucciones ON u OFF.

Para la implementación de este método de persistencia se tomaron en cuenta las siguientes consideraciones:

1. El concepto de Preferencias y/o Configuraciones generalmente son incluidas en las aplicaciones con el objetivo de entregar al usuario la posibilidad de modificar comportamiento y/o apariencia de la aplicación.
2. La apariencia y valores de las preferencia puede ser definidas en archivos XML de manera similar a los ***Layouts***.
3. Una ***Preference*** corresponde a la definición de una sola configuración posible. Esta configuración almacena información a través de un método key-value y son guardadas en el archivo SharedPreferences por defecto de cada aplicación.
4. Las ***Preference*** puede almacenar valores primitivos tales como; Boolean, String,

Float, String set, etc.

5. A partir de la versión 3.0 de Android se reemplaza la ***PreferenceActivity*** por un ***PreferenceFragment*** el cual puede ser "llamado" desde una ***Activity*** cualquiera.

Las preferencias, al tener un componte gráfico, se incluyeron en el paquete principal de la aplicación **cl.mamd.voice** y son implementadas mediante las siguientes clases:

- **VoicePreferenceFragment:** Clase que corresponde a una subclase de ***PreferenceFragment.*** Esta clase es responsable de cargar la vista de las preferencias mediante el archivo XML **voice_main_preference.xml**
- **PreferenceVoice:** Clase que corresponde a una subclase de ***Activity***. Esta clase instancia una clase **VoicePreferenceFragment** para la visualización de las preferencias de la aplicación.

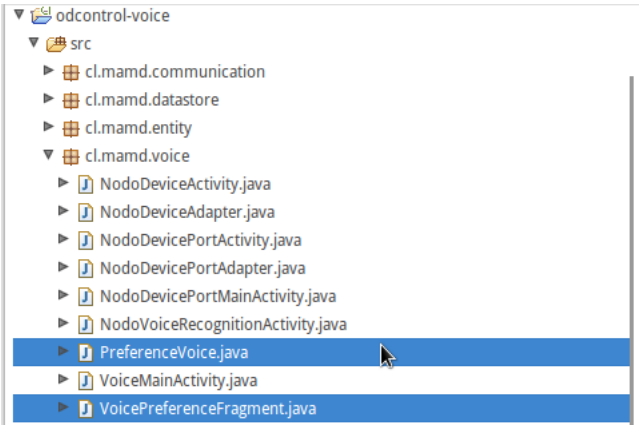


*Ilustración 16: Clases del paquete voice que tiene relación con las preferencias definidas en la aplicación.*

A continuación se detalla el contenido con mayor relevancia en ambas clases que tiene relación con las preferencias de la aplicación.

```
public class VoicePreferenceFragment extends PreferenceFragment {
      @Override
       public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.voice_main_preference);
       }
}
```

Definición de la clase ***VoicePreferenceFragment*** la cual es una subclase de ***PreferenceFragment***. En este punto se carga desde el archivo XML las preferencias definidas para la aplicación.

```
public class PreferenceVoice extends Activity {

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		this.setTitle(getResources().getString(R.string.app_name));


		VoicePreferenceFragment prefFrag = new VoicePreferenceFragment();
		FragmentManager fragmentManager = getFragmentManager();
		FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
		fragmentTransaction.replace(android.R.id.content, prefFrag);
		fragmentTransaction.commit();

	}
}
```

Definición de la clase ***PreferenceVoice***. Esta clase genera una instancia de la ***VoicePreferenceFragment*** para mostrar en pantalla la interfaz gráfica de usuario correspondiente a las Preferencia que puede configurar el usuario.

```
Intent intentSetPref = new Intent(getApplicationContext(),PreferenceVoice.class);
startActivityForResult(intentSetPref,PREFERENCE);
```

Definición del ***Intent intentSetPref*** para el acceso a las preferencia. Se inicia a través de método ***startActivityForResult.***.

```
SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(this);
/*
* Codigo
*/

Set<String> on_list = sharedPref.getStringSet("on_options_list",onSet);
```

En el método ***onActivityResult*** se accede a las preferencias de la aplicación con el fin de obtener nuevamente los datos definidos en las preferencias (pueden sufrir cambios después que un usuario ingrese a la pantalla de configuración). En este caso se obtiene el ***StringSet*** correspondiente a la lista de acciones ON que pueden ser reconocidas.

Uno de los aspectos fundamentales que se debe tener en cuenta, al momento de crear preferencias, es que las herramientas de Android entrega un entorno gráfica que facilita el proceso de construcción de las interfaces de usuarios. La imágenes a continuación muestran los dos escenarios para la edición de un archivo XML perteneciente a las preferencias.

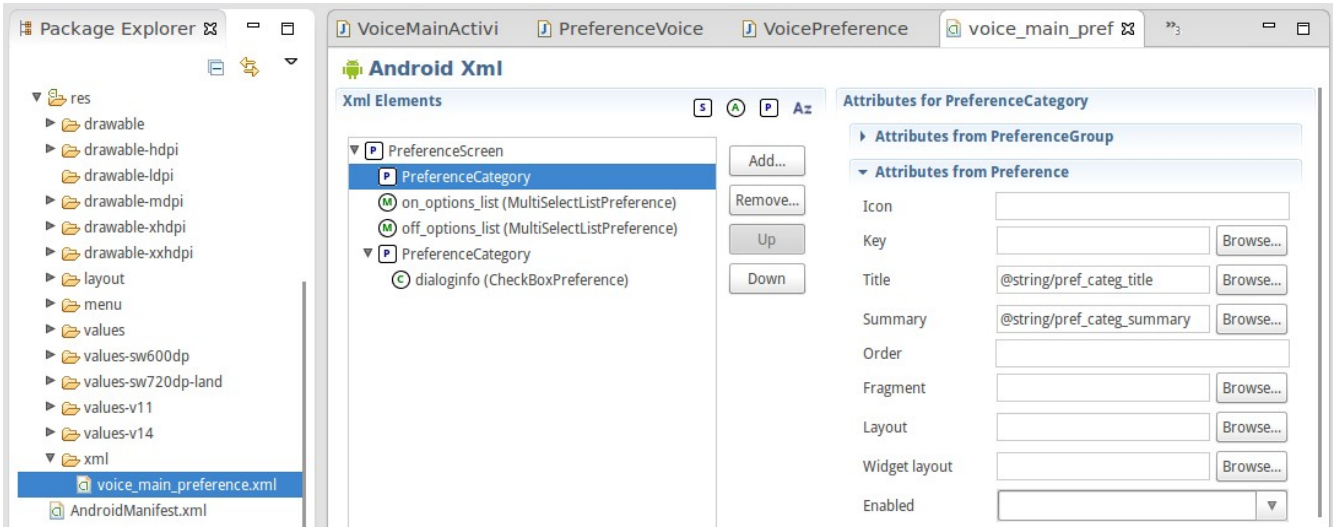


*Ilustración 17: Edición de Archivo XML de preferencia mediante la opción Estructura o Structure.*

En este esquema se puede agregar componente y agruparlos jerárquicamente. La otra opción, es la que veremos a continuación, la cual muestra el código XML del archivo que puede ser editado directamente.

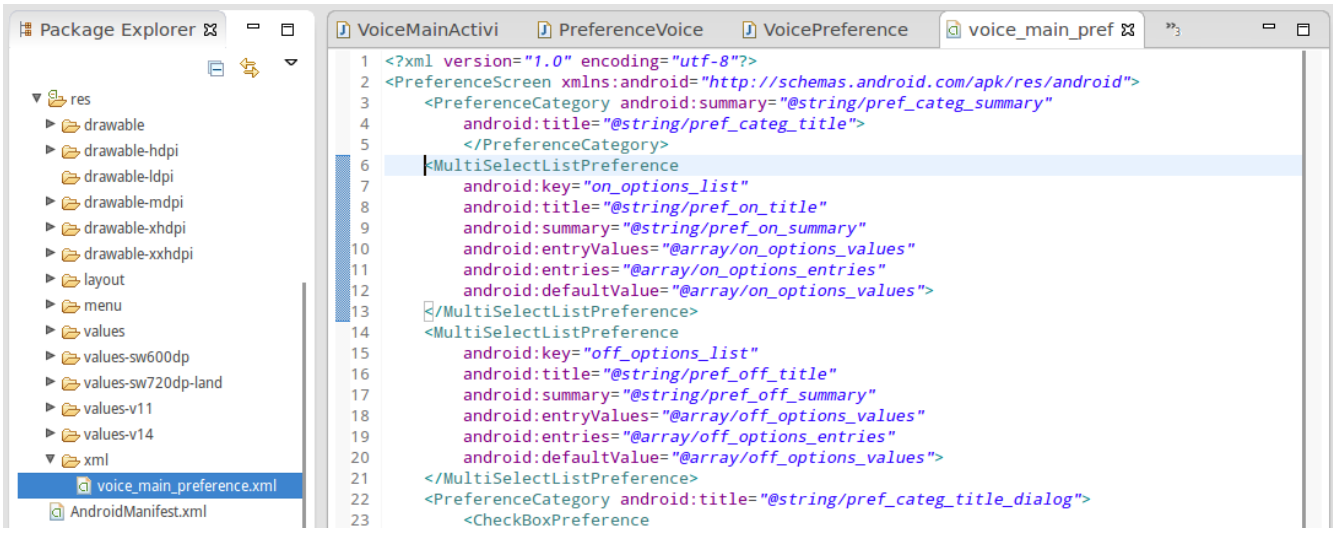


*Ilustración 18: Edición de Archivo XML de preferencia mediante la opción xml (código directo).*

Esta herramienta es una alternativa que reúne dos aspectos importantes; la facilidad de poder crear una interfaz gráfica de usuario junto con la posibilidad de guardar valores/variables los cuales forman parte de la aplicación almacenados de manera persistente.

## Reconocimiento de Voz

El reconocimiento de comando de voz es la principal característica que diferencia el producto de Software con otras alternativas en el mercado. El reconocimiento de voz es utilizado en dos escenario dentro de la aplicación:

- **Reconocimiento de voz para Acciones:** Cuando se quiere guardar la configuración de un puerto (perteneciente a un equipo), se deben definir las acciones tendrá asociadas (las cuales se traducen en una instrucción ON u OFF). Las acciones permitidas se encuentra definidas en la configuración de la aplicación y son palabras que, una vez reconocidas a través de la API de Voz, se traduce en instrucciones ON u OFF.
  - **Acciones - Palabras ON:** Encender, Abrir, Subir, Levantar, Activar, Prender e Iluminar.
  - **Acciones – Palabras OFF:** Apagar,Cerrar, Bajar, Descender, Desactivar, Soltar y Oscurecer.
- **Reconocimiento de Comandos de Voz:** Cuando se accede a un equipo y se quiere ejecutar una instrucción, esta se debe realizar a través de un comando de Voz. El Comando a ejecutar esta compuesto por la etiqueta de un puerto junto con una de las acciones definidas para el. Tenemos el puerto do001 que tiene la etiqueta “Luz comedor” y se le agregaron las acciones “enceder” y “apagar” correspondientes a instrucciones ON y OFF respectivamente. Si el usuario dice el comando “encender luz comedor” o “luz comedor encender” esto se traduce en **set+do001+on.**

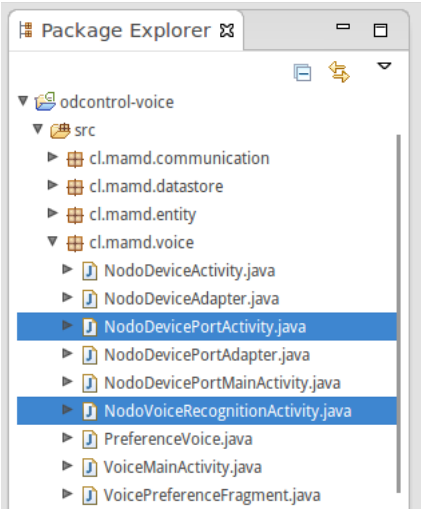


*Ilustración 19: Clases del paquete voice que tiene relación con Reconocimiento de Voz.*

A continuación se detalla el contenido con mayor relevancia que tiene relación con el proceso de reconocimiento de Voz.

```
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS,5);
startActivityForResult(intent, RESULT_SPEECH);
```

Este código es común en las dos clases y permite iniciar un Intent con la acción **ACTION_RECOGNIZE_SPEECH.** Esta acción se traduce en una Activity capaz de reconocer la información dicha por el usuario y entregar un arreglo con su resultado.

Se definen dos los valores para el Intent; la variable EXTRA_LANGUAGE_MODEL[14] con un valor LANGUAGE_MODEL_FREE_FROM para que el reconocimiento lo realice de manera libre (la Activity interpreta los valores como mejor le parezca) y la variable EXTRA_MAX_RESULTS con un valor de 5 para definir el máximo de resultados a retornar por cada reconocimiento.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case RESULT_SPEECH:
            if (resultCode == RESULT_OK && null != data) {
                final ArrayList<String> text = data
                        .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                int i = 0;
                while ( i < text.size()){
                    //text.get(i) has results of recognition
                    /*
                    * Codigo
                    */
                }
            }
        /*
         * Codigo
         */
    }
}
```

Cuando finaliza la ***Activity*** que realiza el reconocimiento de Voz se debe trabajar con el resultado de esta. Este procedimiento se realiza en el método sobrescrito ***onActivityResult*** el cual se ejecuta una vez finalizado el reconocimiento.

El resultado se entrega en un ***ArrayList*** de ***String donde su*** **longitud máxima es 5**. Lo interesante que el resultado se trata como un texto por lo se realiza el procedimiento de “comparar el string con la etiqueta y acción de un puerto”.

Las siguientes imágenes muestras el proceso de “agregar acción a puerto”. En este proceso se lanza una ***Activity*** para realizar el reconocimiento y luego se comparan los resultados con el arreglo de acciones ON u OFF para ver la posibilidad de agregar la palabra mencionada.

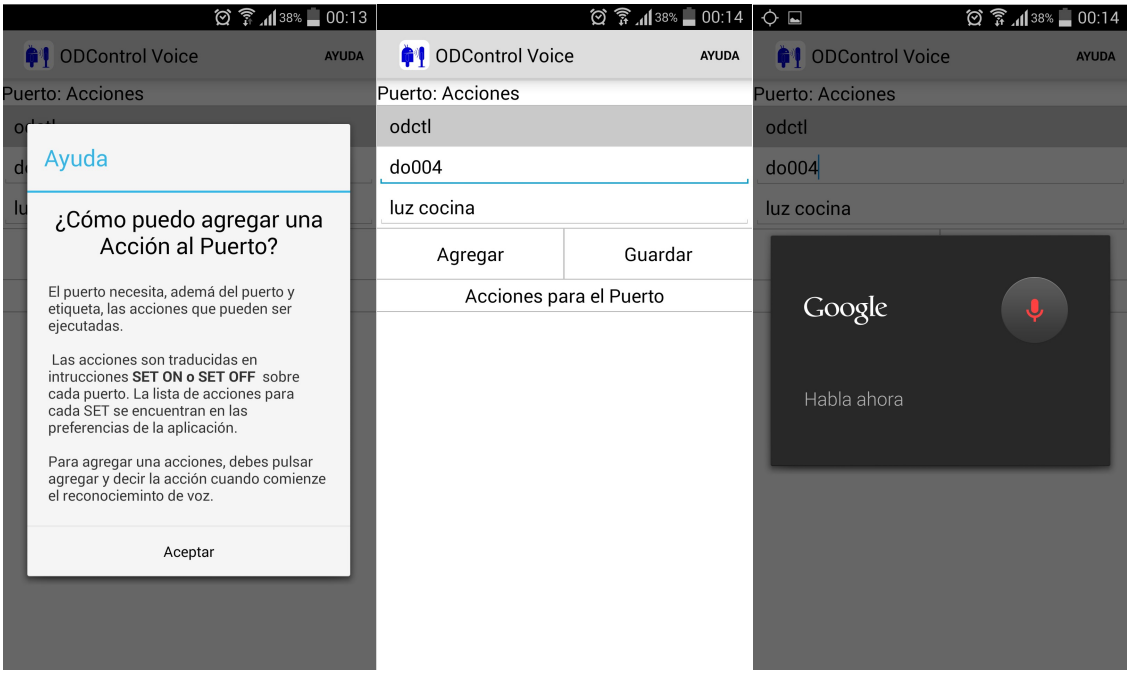


*Ilustración 20: Iniciando el reconocimiento de Voz a través de pulsar el botón "Agregar".*

El resultado del reconocimiento, al ser un arreglo de Strings, se recorre y compara con las palabras ON y OFF configuradas. Existe tres escenarios posibles una vez finalizada la ***Activity*** de reconocimiento:

1. **No existe resultado:** No se entrega resultado o existen problemas para realizar el reconocimiento. Los problemas comunes son poca conectividad junto con el ruido ambiental al momento de utilizar el micrófono del equipo.
2. **Reconoce el comando y es una palabra configurada:** El reconocimiento es exitoso junto la palabra reconocida se encuentra en la lista de acciones ON u OFF configuradas para la aplicación.
3. **Reconoce el comando y no es una palabra configurada:** El reconocimiento es exitoso pero la palabra no pertenece a la lista de acciones ON u OFF configuradas para la aplicación.

```
final ArrayList<String> text = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
//Check if exists on Optiosn
boolean result = false;
int i;
for ( i = 0 ; i < this.on_option.length ; i++ ){
    if ( this.on_option[i].equals(text.get(0)) ){
        result = true;
    }
}
```

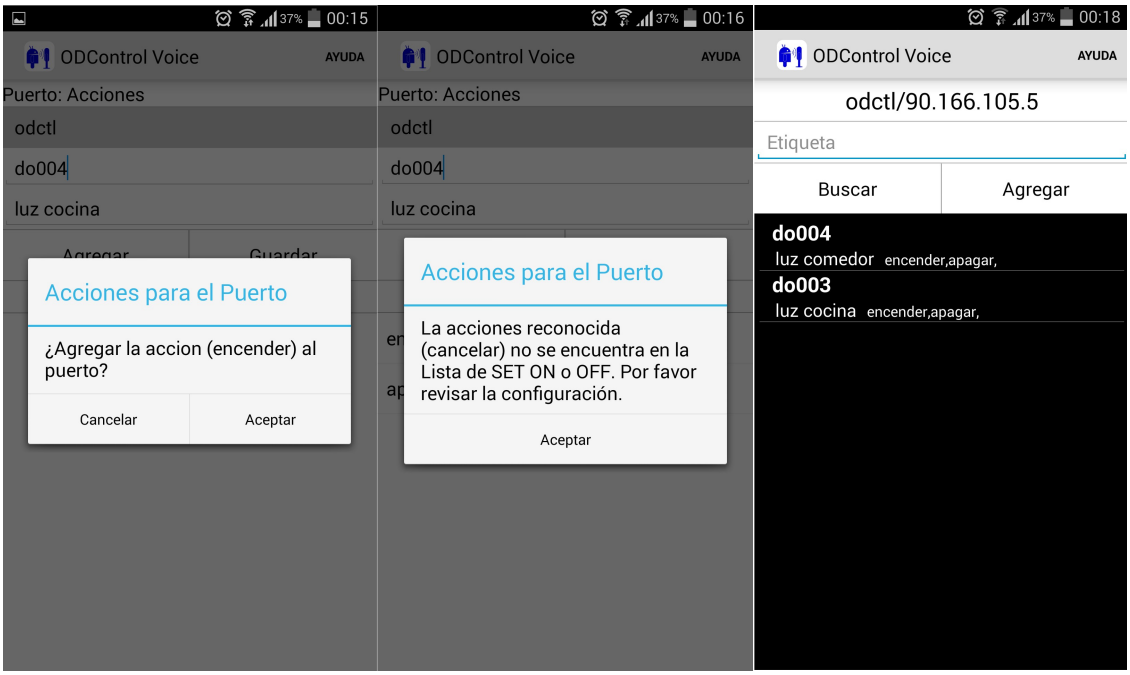


*Ilustración 21: Muestra dos escenarios donde reconoce la acción "enceder" y rechaza la palabra "cancelar" para ser agregada. La palabra "encender" esta en la lista de Palabras ON de la configuración.*

## Ejecución de Instrucción

Una vez finalizado el reconocimiento de un comando de voz se procede a la tarea de ***"traducir esto en una instrucción sobre un dispositivo ODControl"***. Para la ejecución de la instrucción se contemplan que:

1. Se debe utilizar el protocolo **HTTP** y el método **GET** para ejecutar una instrucción en un equipo ODControl.
2. El acceso a los equipos ODControl se realizar con un usuario y contraseña a través de **HTTP Basic access authentication**[15] por lo que se debe contemplar enviar esta información en la ejecución.
3. La respuesta de los equipos ODControl no es **HTTP Response** por lo que para realizar la comunicación se utilizo la clase **Socket** de Java para la implementación sin problemas de respuesta.
4. **Gestionar la interfaz de Usuario:** El procedimiento se debe ejecutar sin afectar la "sensación" del usuario. Para realizar una actualización de la interfaz gráfica posterior a la ejecución de una tarea, se utilizo la clase ***AsyncTask*** de Android.

La aplicación realiza dos tareas en "background" en las que se ejecutan instrucciones sobre un equipo ODControl. Estas tareas se codificaron en dos clases pertenecientes al paquete **cl.mamd.communication:**

- **HTTPCheckAuthentication:** Ejecución de instrucción para verificar si las credenciales están correctas.
- **HTTPExecuteInstruction:** Ejecución de instrucciones **set+puerto+on** o **set+puerto+off** de algún puerto perteneciente a un equipo en particular.

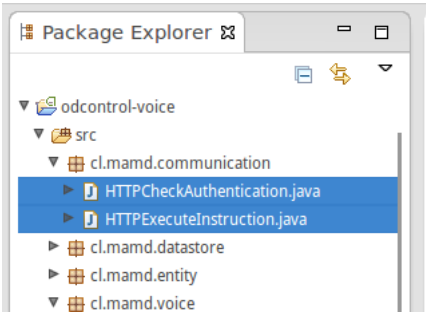


*Ilustración 22: Clases que ejecutan instrucciones sobre equipos ODControl pertenecientes al paquete communication.*

La diferencia entre ambas clases es la ruta que es solicitada mediante el método HTTP GET. Una ruta es “/” para la comprobación de las credenciales y “/set+puerto+opcion” para la ejecución de una instrucción sobre el equipo. Esta ejecución se encuentra en el método doInBackground en cada clase. A continuación se detallas el contenido de mayor relevancia que tiene relación con la ejecución de instrucciones.

```
public class HTTPExecuteInstruction extends AsyncTask<String, Void, String> {
```

Definición de la clase AsyncTask de Android.

```
public HTTPExecuteInstruction(Context context,NodoDevice nodoexe,String
instruction,EditText text){
```

Constructor de la clase **HTTPExecuteInstruction**. Lo fundamental para realizar cambios e interactuar con la interfaz gráfica, es la inclusión del **Context** en los parámetros. De esta manera se obtiene el contexto de la aplicación y la posibilidad de realizar cambios en la interfaz.

```
@Override
protected void onPreExecute() {
```

Este método se llama previo a la ejecución de la tarea a realizar en backgroud. Aquí se pueden realizar cambios y actualizaciones de la interfaz grafía previa ejecución por lo que se utiliza bastante para enviar mensajes o iniciar “loading dialogs”.

```
@Override
protected String doInBackground(String... params) {
```

Método que ejecuta la tarea en background.

```
Socket socket = new Socket();
socket.setSoTimeout(10000);
socket.connect(new InetSocketAddress(addr, port), 10000);

BufferedWriter wr = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(), "UTF8"));
wr.write("GET "+path+" HTTP/1.0rn");
wr.write("Content-Type: text/plain");
wr.write("Authorization: Basic "+author_String);
wr.write("rn");

wr.flush();

BufferedReader rd = new BufferedReader(
```

```
new InputStreamReader(socket.getInputStream()));

String line;
int cont = 0;

while ((line = rd.readLine()) != null) {
```

Se realizar la ejecución de una solicitud a un equipo ODControl a través del método ***HTTP Get***. Para esto se utiliza la clase de Socket en Java. Se define un máximo de tiempo de 10 segundos para obtener una respuesta.

```
@Override
protected void onPostExecute(String unused) {
```

Este método es llamado una vez finalizadas las tareas realizadas en el método ***doInBackground***. Aquí se realizan cambios y actualizaciones de la interfaz grafíca.

En el caso de ejecutar la instrucción correctamente, se pintar de color verde el campo que muestra el reconocimiento realizado.

```
new HTTPCheckAuthentication(VoiceMainActivity.this,nodocred)
.execute();
```

Se llama a la clase ***HTTPCheckAuthentication*** con los parámetros en su construcción y luego se llama al método ***execute***() lo cual desencadena la ejecución de los métodos ya explicado en la sección anterior.

Los resultados entregan un mensaje al usuario mediante la clase ***Toast de Android***. Los posibles mensajes tiene relación con el tipo de respuesta que es recibido por parte del equipo ODControl. Estos resultados y mensajes son:

| Error | Razón | Mensaje a Usuario |
|---|---|---|
| ECONNREFUSED | Al intentar realizar una conexión esta es rechaza. | Conexión rechazada por el equipo, por favor verificar dirección ip del equipo |
| EHOSTUNREACH | El equipo esta inalcanzable. | No se puede alcanzar el equipo,por favor verificar dirección ip y tu conexión |
| ENETUNREACH | Red inalcanzable. | Red inalcanzable. Verificar el estado de la conexión. |
| ETIMEDOUT | Tiempo de espera agotado. | Se cumplió el tiempo limite |

| | | |
|---|---|---|
| | | para la conexión Verificar estado conexión. |
| Unauthorized | No tiene acceso con las credenciales. | Las credenciales de acceso se encuentra incorrectas, no podrá ejecutar instrucciones sobre el equipo |

La siguiente secuencia de imágenes muestra como se despliega un mensaje de error.

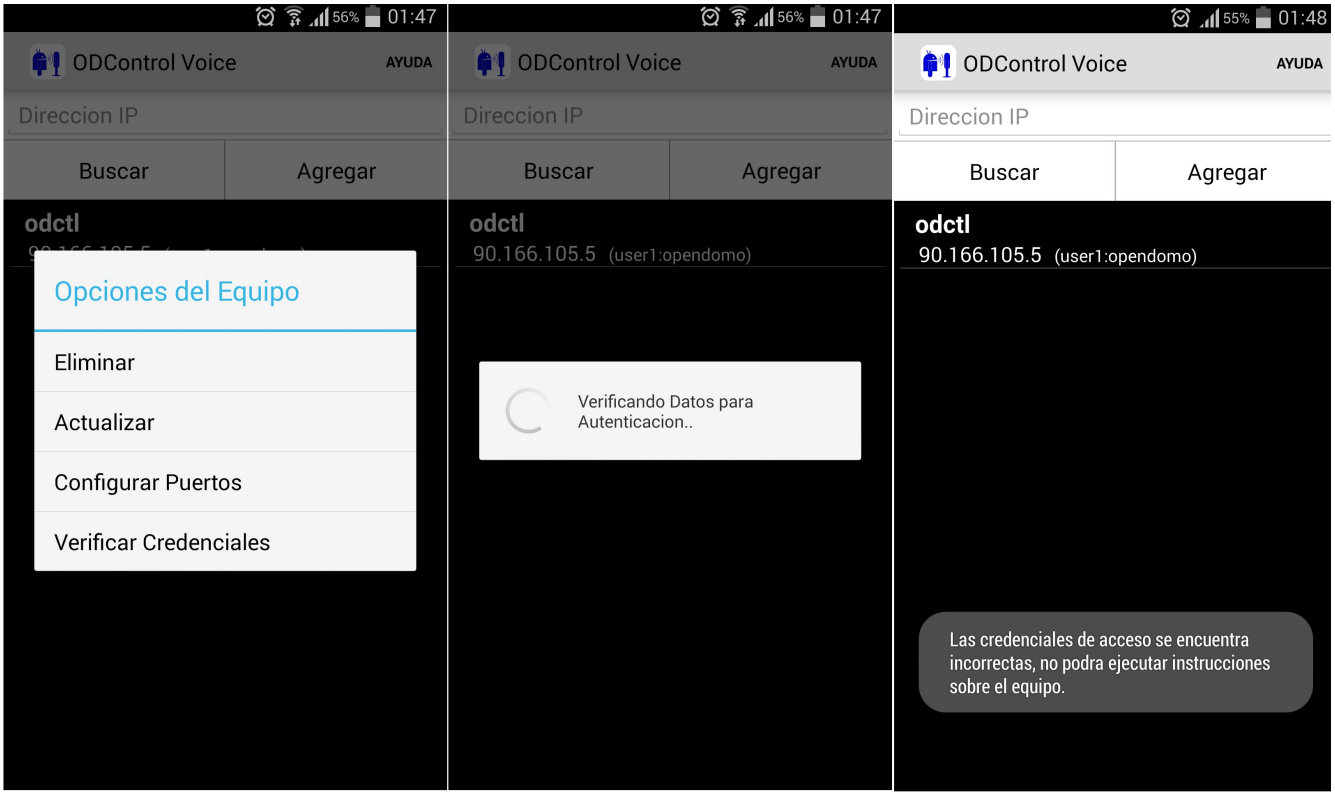


*Ilustración 23: Verificación de credenciales entrega un mensaje que estas se encuentra incorrectas. Esto es debido que el usuario es user y no user1.*

## Interfaz gráfica de Usuario

La gráfica de la aplicación contribuye directamente a la facilidad de uso, fácil acceso a la información. Cada pantalla o ***GUI***, con la cual el usuario interactúa, esta desarrollada mediante la utilización de la clase de Android ***Activity*** y la definición del contenido a través de archivos ***XML***.

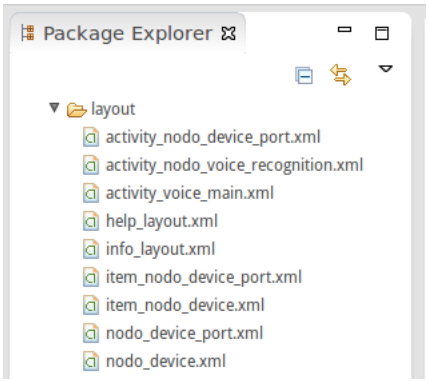


*Ilustración 24: Archivos XML en el directorio layout del proyecto.*

La definición de los ***layouts*** esta compuesta por la inclusión ***View*** y se puede hacer a través de una interfaz gráfica (en el IDE) o mediante la edición directamente en el código. La siguiente imagen muestra el ***Layout,*** de la primera interfaz de usuarios que muestra la aplicación, En el panel “Palette” se pueden apreciar los componentes que pueden ser agregados. En la ilustración 34 se puede apreciar la alternativa de edición directamente en el código del archivo XML.

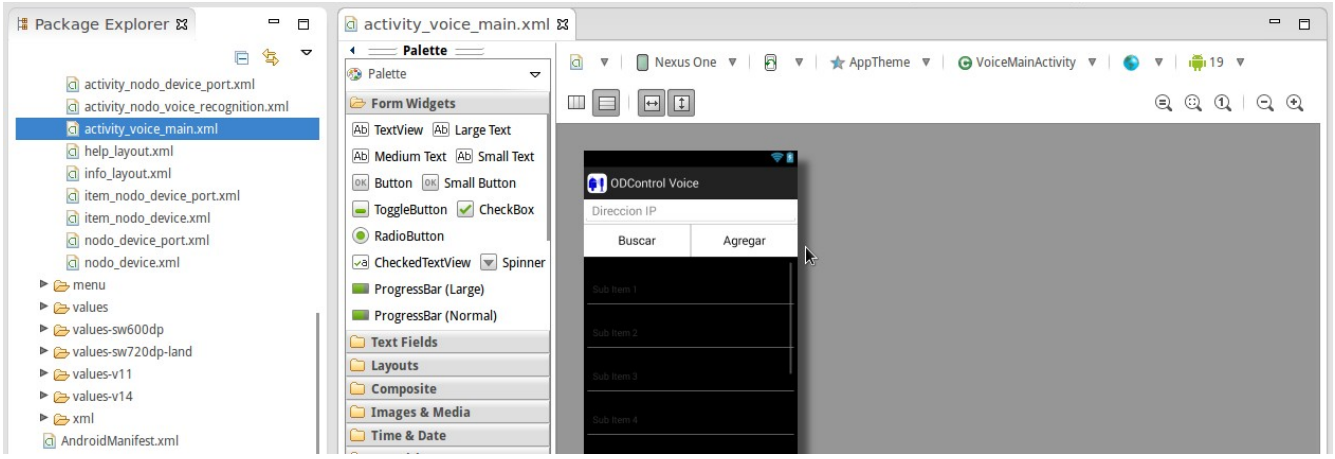


*Ilustración 25: Definición de archivo XML a través de la Interfaz grafica. La principal característica es que cualquier edición muestra de inmediato como se vería el layour dentro de la aplicación.*

Las interfaces de usuarios son construidas mediante la utilización de objetos del tipo ***View***[16], y ***ViewGroup***. Los componentes ***View*** son organizados jerárquicamente mediante ***ViewGroup*** los que permite definir el armazón o esqueleto de cada layouts.

En los layouts de la aplicación podemos encontrar:

| Elemento View/ViewGroup | Descripción |
|---|---|
| <LinearLayout> | Permiten ordenar todos los componentes en su interior en una sola columna o fila. [17] |
| <TextView> | Elemento que permite mostrar texto en la Interfaz |
| <EditText> | Elemento que permite manejar un campo de texto editable en el cual el usuario puede escribir. |
| <Button> | Elemento que permite la utilización de un botón en la Interfaz. |
| <ListView> | Permite mostrar una lista de elementos los cuales se desplazan automáticamente. Cada elemento de se inserta a través de la gestión de Adapter. [17] |
| <CheckBox> | Elemento que permite la utilización de un checkbox en la interfaz. |

Luego de definir la interfaz, en un archivo XML, esta debe ser cargado a través de una ***Activity***. El siguiente código muestra el método ***onCreate*** sobrescrito de la clase ***VoiceMainActivity*** (La que se muestra cuando se inicia la aplicación) donde se define el contenido de la ***Activity*** proveniente del archivo ***activity_voice_main.xml.***

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*
         * Disabled screen orientation changes and remove title
         */
        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        this.setTitle(getResources().getString(R.string.app_name));
        this.setContentView(R.layout.activity_voice_main);
```

La manera de trabajar con los ***View*** dentro del código ( por ejemplo recuperar la dirección IP ingresada por el usuario) es creando variables del tipo de ***View*** que se quiere acceder. Un ejemplo de esto se muestra a continuación; donde se define un atributo el tipo ***EditText*** en el archivo XML, para que el usuario pueda ingresar una dirección IP, y luego con esta poder buscar o agregar un nuevo equipo a la aplicación.

```
<EditText
    android:id="@+id/edittext_ipaddress"
    android:hint="@string/hint_ipdevice"
    android:layout_width="0dip"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:textColor="@color/black"
    android:ems="1" >
</EditText>
```

*Ilustración 26: Código XML del elemento EditText.*

En el archivo ***XML*** se define el atributo ***android:id*** correspondiente a la identificación única del elemento dentro de la aplicación. Este ***id*** es utilizado en la ***Activity*** para asociar el elemento del archivo ***XML*** con un atributo o variable de la clase del tipo ***android.widget.EditText.***

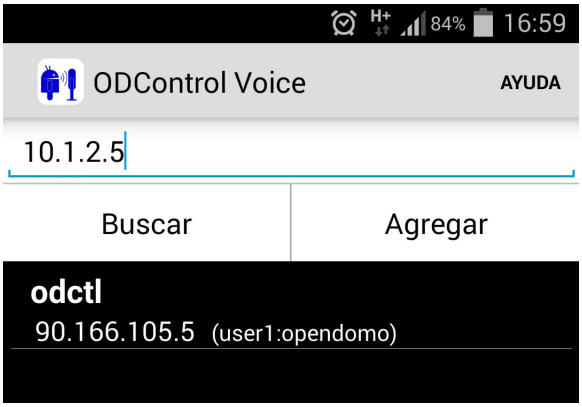


*Ilustración 27: Ingreso del valor "10.1.2.5" por el Usuario.*

En el código de la aplicación se define el atributo ***ipaddress_for_add*** tipo ***EditText*** para trabajar con la información ingresada por el usuario.

```
private EditText ipaddress_for_add;
```

Definición de Atributo en Clase ***VoiceMainActivity***

Posterior a cargar el contenido de la aplicación desde el archivo ***XML*** se vincula el atributo con el elemento ***EditText*** definido en el archivo ***XML***.

```
this.ipaddress_for_add = (EditText)findViewById(R.id.edittext_ipaddress);
```

Asignación de Elemento con id edittext_ipaddress al atributo ipaddress_for_add de la clase ***VoiceMainActivity***

Con la asignación ya definida, se puede trabajar y acceder al contenido que el usuario ingresa en el elemento de la interfaz Gráfica. El siguiente código muestra el método, de la clase ***VoiceMainActivity,*** que es llamado al pulsar el botón Buscar. Aquí recupera el texto ingresado por el usuario mediante el método **getText()** del atributo ipaddress_for_add y lo almacena en una variable del tipo ***String***.

```
public boolean searchDeviceButton(View view){
      String ipforsearch = this.ipaddress_for_add.getText().toString();
```

Existe otro componente de la interfaz gráfica que tiene especial relevancia en la aplicación en cual corresponde a las ***ListView***. Este elemento gráfico se utiliza para visualizar las listas de; equipos, puertos y acciones. El objetivo de la lista es utilizar este tipo de ***ViewGroup*** para mostrar una cantidad de N elementos y asociar acciones al seleccionar uno de ellos. La asociacion de acciones a cada selección se realiza a través de los siguientes ***Listener:***

- **OnItemClickListener:** Interfaz que es invocada cuando un elemento o item de la lista es seleccionado o clickeado.[19]
- **OnItemLongClickListener:** Interfaz que es invocada cuando un elemento o item de la lista es seleccionado y se mantiene pulsado.[20]

El siguiente código muestra como se agregan elementos al ***ListView*** utilizado en la clases ***VoiceMainActivity.*** Los elementos corresponden a la lista de todos los equipos almacenados en la base de datos ***SQLite***.

```
/*
* Getting device information from SQLite Database
*/
dsm = new DataStoreManager(this);
dsm.openDataBase();
this.values = dsm.getAllNodoDevice();

/*
* Loading all devices in list view
*/
this.adapter = new NodoDeviceAdapter(this,values);
this.listView.setAdapter(adapter);
```

Recuperación de información desde la base de datos, creación de un ***Adapter***[18] definido para mostrar la información de los equipos y definir el ***Adapter*** a utilizar en el ***ListView***.

El siguiente código muestra como se implementa en la clase ***VoiceMainActivity*** las interfaces mencionadas anteriormente. Un punto importante es que requiere sobrescribir de métodos para gestionar las acciones frente a un click o un “click prolongado”. Estos métodos son: ***onItemClick*** y ***onItemLongClick***.

```
public class VoiceMainActivity extends Activity implements
OnItemClickListener,OnItemLongClickListener {
```

Implementación de interfaces para el trabajo con elementos de ListView

El siguiente código muestra las tareas que se ejecuta una vez clickeado un Equipo de la Lista (que tiene puertos configurados). En este caso lanza la **Activity** que permite luego hacer el reconocimiento de comandos de Voz para realizar instrucciones.

```
@Override
public void onItemClick(AdapterView<?> arg0, View view, int arg2, long arg3) {

/*
* Access to Device
*/

Intent nodovoicerecognition = new Intent(VoiceMainActivity.this,NodoVoiceRecognitionActivity.class);
TextView ipvalue = (TextView)view.findViewById(R.id.textViewIpAddress);
dsm.openDataBase();
final NodoDevice nodo = dsm.getDevice(ipvalue.getText().toString());

/*
* Check port configuration
*/
dsm.openDataBase();
if ( dsm.checkPortOfDevice(nodo.getId()) ){
    /*
     * Codigo
     */
     startActivityForResult(nodovoicerecognition,NODOVOICE_REQUEST);
     dsm.closeDataBase();
}
```

El valor del parámetro ***view*** corresponde al elemento de la lista “clickeado”. Se recupera el valor del ***TextView*** que tiene la dirección IP del equipo y con este se realiza la verificación de puertos.  Si el equipo tiene puertos se inicia la ***Intent nodovoicerecognition***.

Los aspectos mencionados durante esta sección permite tener una perspectiva clara de como construyen las interfaces de usuario y como esta interactuar en el código para la implantación de las reglas de negocio de la aplicación.

## *Pruebas*

Una de las principales etapas del desarrollo de un producto de software corresponde a las pruebas o t***esting.*** Las pruebas permiten evaluar la calidad, validaciones o tolerancia a errores de un producto de software o de un módulo en particular.

La metodología utilizada durante la implementación, corresponde a la generación de prototipos funcionales y al finalizar cada iteraciones se realizan pruebas. Las pruebas realizadas durante el proceso de desarrollo se dividen en tres tipos:

- **Pruebas unitarias de clases implementadas:** Luego del desarrollo de cada clase se realizan pruebas unitarias de funcionamiento.

- **Pruebas de funcionamiento de procesos críticos:** Los procesos críticos están relacionadas con gestión de la base de datos, gestión de preferencias, reconocimiento de Voz y ejecución de instrucciones.

- **Pruebas de la manipulación y gestión de respuesta de la interfaz gráfica para la gestión de información:** Esta pruebas están enfocadas solamente a la manipulación de las interfaces gráficas creadas y las respuestas que están tiene frente a la manipulación del usuario. Además de como se comportan en los distintos estados del ciclo de vida de cada ***Activity***.

El tipo de pruebas mencionado anteriormente se realizan durante el proceso de desarrollo y entre cada iteración por lo que la evaluación de esta va de la mano con las funcionalidades particulares que se están implantando. Al finalizar la prueba existe dos casos posibles:

- **Prueba sin observaciones:** El funcionamiento es correcto y no presenta observaciones para ser retocado el código.
- **Prueba con observaciones:** El funcionamiento presenta observaciones las cuales se pueden traducen en cambios o correcciones. Es importante señalar que no todas las observaciones corresponde a errores por lo que también puede ser distintas formas de atacar el problema.

## Proceso de pruebas finales

Para las pruebas finales del producto de Software se utilizo la metodología de la caja blanca y la caja negra. Las **pruebas de *caja blanca*** se centran en el conocimiento del funcionamiento interno de la aplicación por lo que van de la mano con la revisión del código fuente y las secciones relacionadas con el flujo de trabajo o de datos que se pretende probar. Las **pruebas de *caja negra*** se centran en la evaluación de los datos de entradas y las salidas que el producto de Software entrega dejando de lado el conocimiento del funcionamiento interno de la aplicación.

Las siguientes imágenes muestra una representación gráfica de como interpretar el funcionamiento de los dos tipos de pruebas aplicados en el procedimiento de testing del producto de Software.

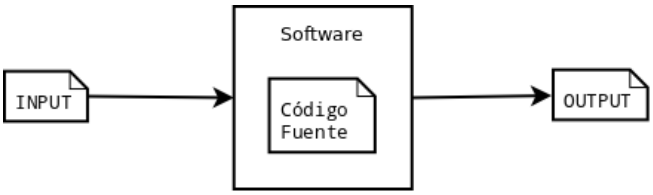


*Ilustración 28: Pruebas Caja Blaca*

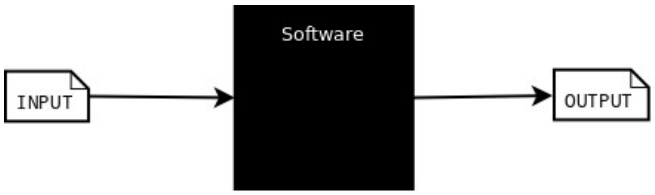


*Ilustración 29: Pruebas Caja Negra*

El espíritu de las pruebas de caja blanca son las que se manejan durante el desarrollo continuo del producto de Software.

## Herramienta – Logcat y Clase Log.

La herramienta utilizada para obtener información,durante las pruebas, respecto a las caídas y/o errores presentes en la aplicación es Logcat[21]. Logcat permite revisar y filtrar todo los mensajes generados por errores o la utilización de la Clase Log. [22] Esta clases tiene la particularidad de poder enviar salidas log para conocer los problemas y el comportamiento de la aplicación.

A través de la clase Log se puede clasificar los mensajes enviados por la aplicación en:

- **Log.i – INFO**: Mensaje de información.
- **Log.d – DEBUG**: Mensaje de depuración
- **Log.e – ERROR**: Mensaje de error.
- **Log.w – WARNNIG**: Mensaje de alerta.

La siguiente imagen muestra la Interfaz de LogCat donde se pueden apreciar los Log generados por la aplicación. En este caso el

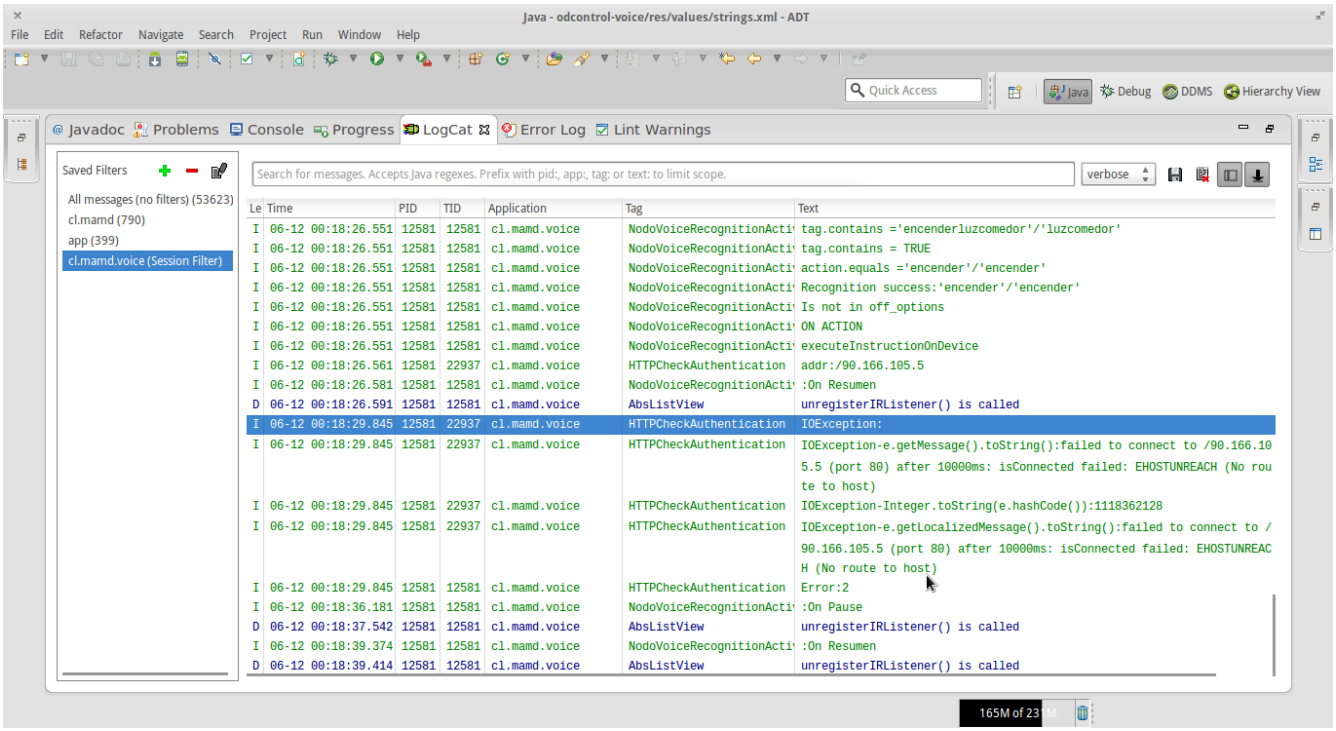


*Ilustración 30: Log.i Equipo inalcanzable - EHOSTUNREACH*

## Pruebas realizadas

**Las pruebas de flujo de trabajo** tiene como principal objetivo evaluar de manera global el funcionamiento y cumplimiento de los escenarios planteados en los casa caso de uso. Esta pruebas son de Caja negra y pretenden evaluar solamente como usuario la aplicación.

Otro caso distinto son las pruebas de flujo de datos. Para estas pruebas el procedimiento consisten en evaluar todas las posibles entradas que tiene la aplicación y el comportamiento de esta traducido en Salidas o mensajes. Estas pruebas son las que se traducen en la gestión de los posibles errores y las respuesta de estos por parte de la aplicación.

Para las pruebas de flujo de trabajo se tomar en cuenta los casos de uso:

- Crear Equipo
- Actualizar Equipo
- Eliminar Equipo
- Crear Puerto
- Actualizar Puerto
- Eliminar Puerto
- Ejecutar Instrucción.

La siguiente tabla muestra las pruebas de flujo de trabajo realizadas y el comportamiento esperado por la aplicación. Este comportamiento fue establecido luego de reiterados intentos de pruebas y posterior a las correcciones junto con validaciones en el código.

| # | Nombre | Comportamiento |
|---|---|---|
| T 1 | Creación de Equipo | Existe campos obligatorios para la creación. Al completar todos los campos se agrega el equipo y aparece en la Lista. |
| T 2 | Creación de Puerto | Existe campos obligatorios para la creación. Al completar todos los campos se agrega el puerto y aparece en la Lista. |
| T 3 | Actualización de Equipo | Se actualiza la información del Equipo. |
| T 4 | Actualización de Puerto | Se actualiza la información del Puerto. |
| T 5 | Eliminación de Equipo | Se elimina la información del Equipo. Esto elimina la información de los puertos del Equipo. |
| T 6 | Eliminación de Puerto | Se elimina la información del Puerto. Esto no elimina la información del Equipo |
| T 7 | Acceder al Equipo | Accede al equipo para reconocer comandos de voz. |
| T 8 | Ejecutar Instrucción | Se reconoce la instrucción de Voz. Para esto se debe tener conectividad. |

La siguiente Tabla muestra las pruebas de flujo de datos realizadas. Además se incluyen los mensajes (gestionados por la aplicación) correspondiente a errores en las reglas de negocios e integridad de la información.

| # | Nombre | Entrada | Salida Error |
|---|---|---|---|
| D 1 | Agregar equipo | | |
| D 1.1 | Agregar equipo | IP | Dirección IP muy larga |
| D 1.2 | Agregar equipo con misma IP | IP ya utilizada | La dirección IP ya se encuentra ocupada por otro equipo |
| D 2 | Agregar información Equipo | | |
| D 2.1 | Agregar información Equipo | Falta nombre | Se debe agregar nombre del equipo |
| D 2.2 | Agregar información Equipo | Falta username | Se debe agregar el nombre de usuario del equipo |
| D 2.3 | Agregar información Equipo | Falta clave | Se debe agregar la clave del equipo |
| D 3 | Agregar información puerto | | |
| D 3.1 | Agregar información puerto | Falta puerto | Se debe completar el campo y debe ser inferior a 6 caracteres |
| D 3.2 | Agregar información puerto | Falta acción | Se debe agregar como mínimo una acción para el Puerto |
| D 3.3 | Agregar información puerto | Falta etiqueta | Se debe agregar una Etiqueta al Puerto. Como Luz, Ventana, etc |
| D 3.4 | Agregar información puerto | Etiqueta duplicada | No se puede agregar la etiqueta debido a que ya esta asociada a un puert |
| D 3.5 | Agregar información puerto | Puerto duplicado | El nombre del puerto ya se encuentra agregado |
| D 4 | Verificar credenciales | | |
| D 4.1 | Verificar credenciales | Problemas de conexión | ECONNREFUSED: Conexión rechazada por el equipo, por favor verificar dirección ip del equipo. |
| D 4.2 | Verificar credenciales | Problemas de conexión | EHOSTUNREACH:No se puede alcanzar el equipo,por favor verificar dirección ip y tu conexión. |
| D 4.3 | Verificar credenciales | Sin conexión a Red | ETIMEDOUT:Se cumplio el tiempo limite para la conexión. Verificar estado conexión. |
| D 4.4 | Verificar credenciales | Problemas de conexión | ENETUNREACH:Red inalcanzable. Verificar el estado de la conexión. |
| D 5 | Pulsar atrás | | |
| D 5.1 | Pulsa atrás en agregar equipo | No pulsar Agregar | Atención: Los datos ingresados no fueron guardados |
| D 5.2 | Pulsa atrás en agregar equipo | No pulsar Actualizar | Atención: Los datos ingresados no fueron actualizados |
| D 5.3 | Pulsa fuera del dialog en agregar equipo | Pulsa atrás en agregar equipo | Pulsa atrás en agregar equipo |
| D 5.4 | Pulsa fuera del dialog en agregar equipo | No pulsar Actualizar | Atención: Los datos ingresados no fueron actualizados |
| D 5.5 | Pulsa atrás en agregar puerto | NO pulsar Agregar | Atención: Los datos ingresados no fueron guardados |
| D 5.6 | Pulsa atrás en agregar puerto | No pulsar Actualizar | Atención: Los datos ingresados no fueron actualizados |

## Caso de prueba

Parte del desarrollo de pruebas es la evaluación del caso de éxito de la aplicación. Este es un escenario controlado donde se evaluar el funcionamiento óptimo. Para el proyecto el funcionamiento óptimo consiste en conseguir que la aplicación pueda ejecutar una instrucción a través del reconocimiento de Voz.

Para realizar esto se deben tener las siguientes condiciones:

1. Debe existir a lo menos un equipo Registrado.
2. El equipo debe tener asociado a lo menos un puerto.
3. El puerto perteneciente al equipo debe tener a lo menos una acción configurada.
4. Las credenciales de acceso deben estar correctas y verificadas.
5. Debe existir conectividad de Red
6. Debe estar poder alcanzar el equipo ODControl que se pretende controlar.

Para cualquier incumplimiento de las condiciones mencionadas anteriormente, existe una alerta o restricción de la aplicación indicando al usuario porque no se consigue el objetivo establecido. Una vez que existe todas las condiciones se procede a realizar las pruebas del caso óptimo. Estas pruebas fueron realizadas mediante la utilización de un smartphone Galaxy S4 y un equipo ODControl con la dirección ip 90.166.105.5. Las siguientes imágenes muestran el caso de éxito de la ejecución de un comando de Voz.

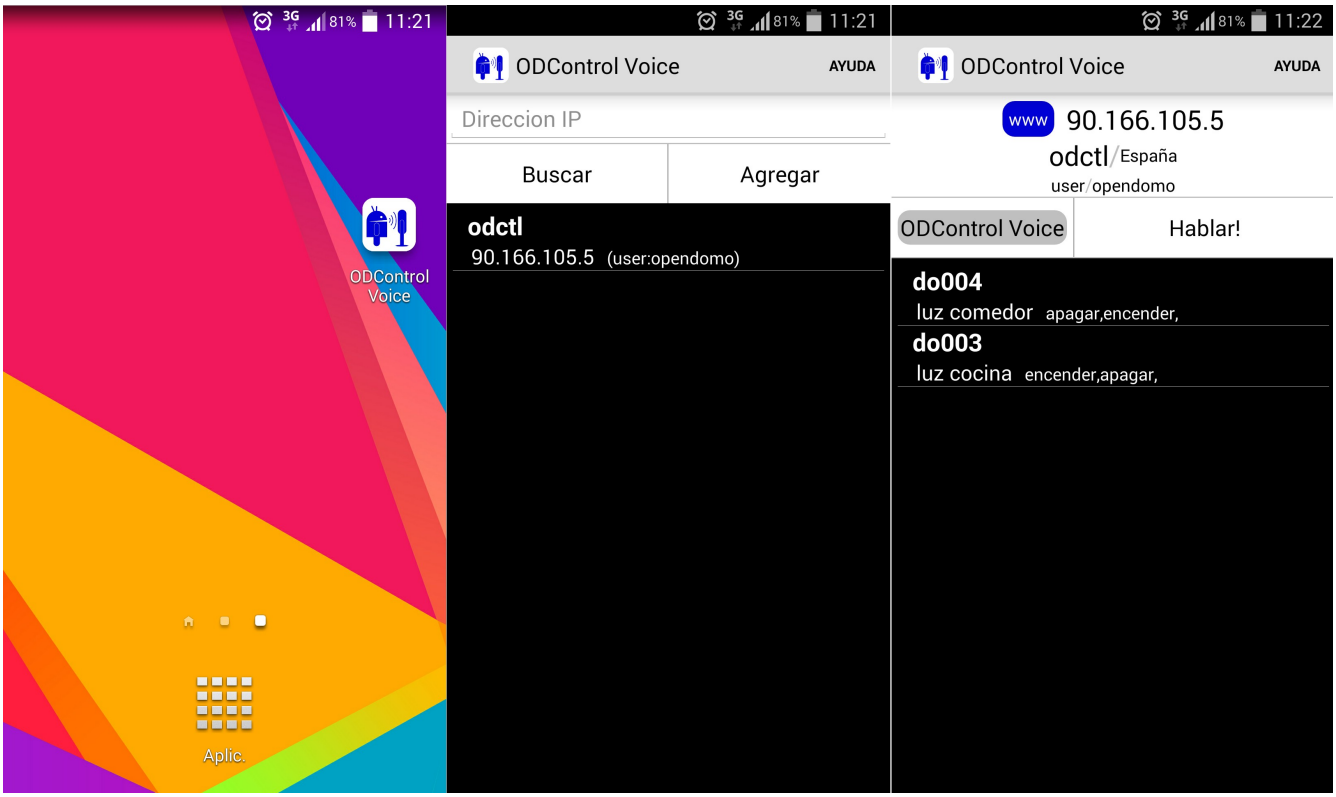


*Ilustración 31: Acceso a la Aplicación y a un equipo para ejecutar comandos de voz.*

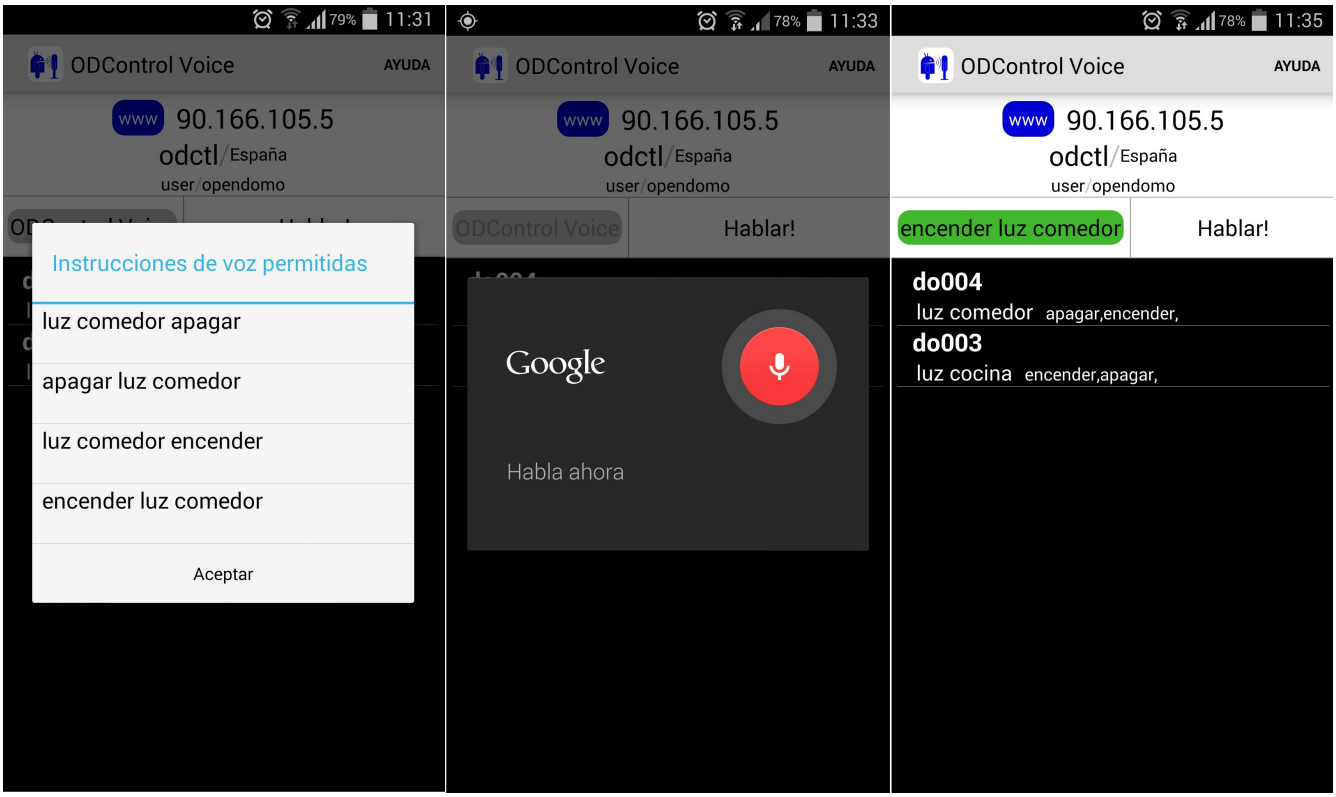


*Ilustración 32: Ver instrucciones permitas para un puerto e iniciar el reconocimiento de Voz.*

A través de las imágenes se puede ver como el usuario:

1. Accede a la Aplicación
2. Selección un Equipo
3. Ver instrucciones permitidas sobre un puerto.
4. Pulsa el botón “Hablar” para comenzar el reconocimiento
5. El reconocimiento es exitoso.
6. Se ejecuta la instrucción sobre el equipo satisfactoriamente.

# Conclusión

En esta sección se presentan las principales conclusiones obtenidas durante el desarrollo del producto de Software en el marco de la práctica correspondiente al Proyecto Final del Máster.

La posibilidad de realizar un producto de Software, el cual forma parte de un proyecto mayor como es ODControl, permite participar en un escenario en el cual se aterrizan conocimientos vistos manera teórica y en prácticas no formales en un producto de Software terminado.

Además el compromiso que existe por parte de los tutores, tanto externos como de la UOC; de apoyo durante todo el proceso de desarrollo, permite encaminar de mejor manera los esfuerzo estimados para cada una de las tareas. Esto entrega la posibilidad de comparación con la realidad cotidiana y como nos desenvolvemos en nuestro actual ambiente laboral.

Parte de las conclusiones son los principales aportes y experiencias experimentadas en el antes, durante y posterior a la práctica.

- El desarrollo de una aplicación nativa en Android, para dispositivos ODControl que permita ejecutar instrucciones a través de comando de voz, abre la posibilidad de poder generar más productos de Software que puedan solucionar distintos tipos de necesidades de OpenDomo.
- Existen distintos tipos de componentes que deben trabajar en conjunto para que el producto de Software pueda suplir los requerimientos establecidos en este proyecto. La posibilidad de poder integrar y plasmar en un producto de Software conocimientos de Redes, Desarrollo de Software y Microcontroladores, permite tener una visión amplia de como distintos tipos de tecnologías funcionan entre sí y se integran en soluciones completamente funcionales.
- Otro aspecto importante, que va de la mano con el desarrollo de Software, es entender como participar en un equipo de trabajo no presencial y la utilización de herramientas para apoyar el desarrollo (IDE, Sistema de control de Versiones, Sistema de reporte de errores y Correo/Listas de correos). Estos conocimientos son transversales a los que se adquieren durante el desarrollo y entregan un valor agregado a todo tipo de trabajo o escenario laboral donde puedan ser aplicados.
- La experiencia de poder participar con OpenDomo mediante el desarrollo de un producto de Software capaz de suplir los requerimientos planteados, entrega indicadores que permiten conocer las principales fortalezas y debilidades frente a un escenario profesional real.

## *Trabajos futuros*

Uno de los aspectos fundamentales de la participación en el desarrollo de un producto de Software es la posibilidad de ver proyectar trabajo a futuro. En este contexto es importante señalar que:

- **Mayor integración con ODControl:** Trabajar para crear una mayor integración con los dispositivos ODControl. Esto es pensado en escenario como:
  - Si el ODControl ya esta funcionando, se podría “migrar a la aplicación” los puertos que ya se tiene configurados para evitar realizar estos pasos en la aplicación en Android. Hoy en día se debe agregar un equipo y agregar cada puerto que se quiere operar a este de manera “manual”.
  - Si se realizan cambios mediante otra interfaz a un ODControl en particular, no existe una publicación del estado o un registro respecto a los cambios sobre cada puerto digital o análogo. Este problema no permite saber si una interfaz esta ON u OFF y solamente se comprueba viendo su interfaz Web.
- **Desarrollo de una Interfaz nativa en Android: E**l objetivo es entregar una alternativa a la interfaz web que tiene los dispositivos ODControl. Si bien tiene una interfaz web a la cual se puede acceder a través de un Browser, la incompatibilidad de algunos componentes hace que se dificulte la operabilidad sobre el cliente web de los dispositivos móviles.
- **Registro de Actividades en la Aplicación para configurar parámetros de conductas:** La posibilidad de registrar, cuando y que se ejecuta a través de la aplicación permitiría que esta misma le sugiera a los usuarios que hacer a cierta hora del día. Esta sugerencia sería en base a las instrucciones ejecutadas a través de la aplicación. Un ejemplo de esto, es que si entre el horario 19:00 y 20:30 horas el usuario siempre “enciende luz comedor”, que la aplicación le pueda recordar a las 18:50 esta tarea mediante alguna notificación o alerta.
- **Inclusión de Temporizador o Timer:** En un contexto en el cual se ejecute por ejemplo “encender riego jardín” poder definir Temporizadores/Timers para que posterior a cierta cantidad de Tiempo pueda “apagar riego jardín” o recordarle al usuario “Riego esta activado, ¿Desea apagar?”.

Estos son algunos proyectos que se pueden desglosar del trabajo realizado durante el desarrollo de este producto de Software, que pueden tener un gran impacto en el mercado y en el valor agregado que se puede entregar, mediante la inclusión de una Aplicación en Android, a los productos ODControl comercializados por OpenDomo.

# Anexo

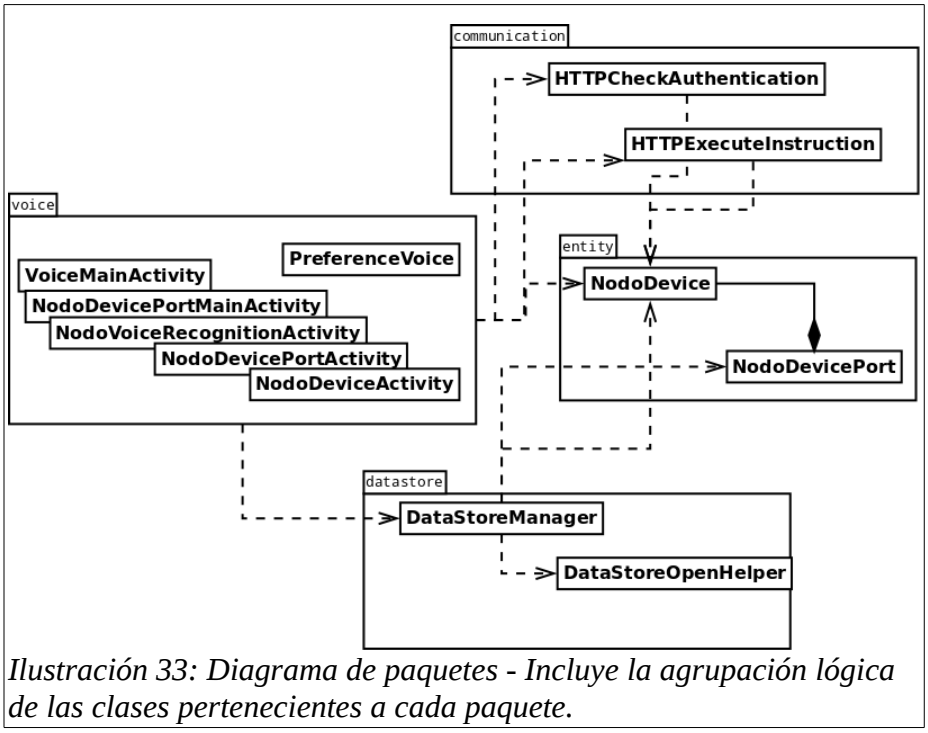


*Ilustración 33: Diagrama de paquetes - Incluye la agrupación lógica de las clases pertenecientes a cada paquete.*

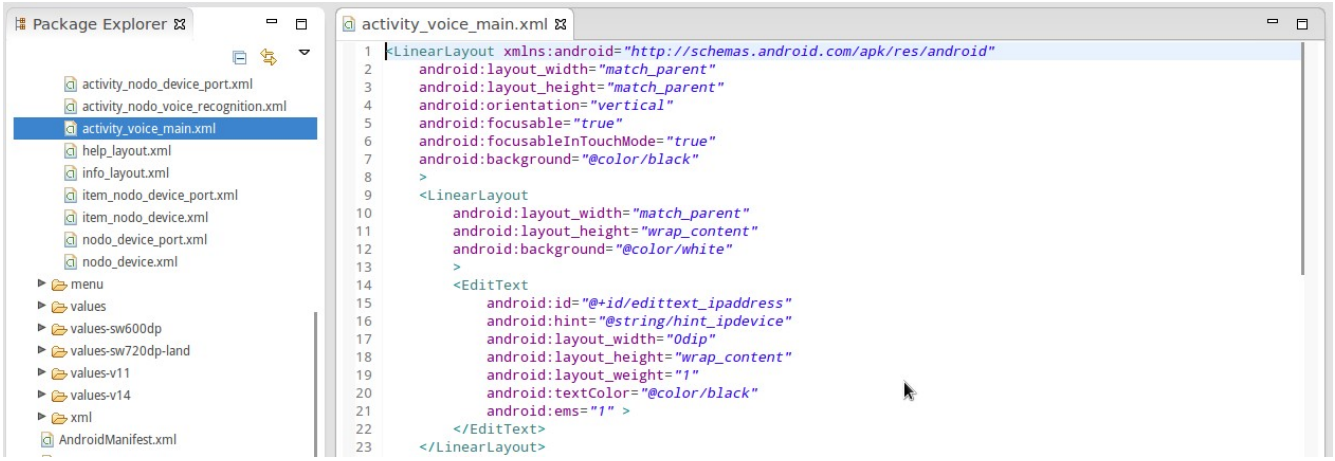


*Ilustración 34: Edición de layout directamente en su código.*

# Referencias


1. The Computer for the 21st Century , Mark Weiser , 1991.
2. Domotica, http://es.wikipedia.org/wiki/Domotica
3. ODControl, http://www.opendomo.es/products/ODControl
4. OpenHandSet Alliance, http://www.openhandsetalliance.com
5. Android Arquitectura, http://elinux.org/Android_Architecture
6. Ubi, http://www.theubi.com
7. Tasker + Plugin, http://tasker.dinglisch.net/
8. SmartThings, http://www.smartthings.com
9. Github, https://github.com/
10. Domino OSE, https://github.com/opalenzuela/dominoOSE
11. Arduino, http://arduino.cc/en/Main/Software
12. Using Databases, http://developer.android.com/guide/topics/data/data-storage.html#db
13. Preferences, http://developer.android.com/guide/topics/ui/settings.html
14. RecognizerIntent: EXTRA_LANGUAGE_MODEL ,http://developer.android.com/reference/android/speech/RecognizerIntent.html#EXTRA_LANGUAGE_MODEL
15. Basic access Authentication, http://en.wikipedia.org/wiki/Basic_access_authentication
16. UI Overview, http://developer.android.com/guide/topics/ui/overview.html
17. Android Development Lecture 3, Android Graphical User Interface 1, http://www.ce.unipr.it/~picone/mobdev/Unipr_AD_3.pdf
18. Abstrac Class BaseAdapter, http://developer.android.com/reference/android/widget/BaseAdapter.html
19. Interface AdapterView.OnItemClickListener, http://developer.android.com/reference/android/widget/AdapterView.OnItemClickListener.html
20. Interface AdapterView.OnItemLongClickListener, http://developer.android.com/reference/android/widget/AdapterView.OnItemLongClickListener.html
21. Reading and Writing Logs, http://developer.android.com/tools/debugging/debugging-log.html
22. Final Class Log, http://developer.android.com/reference/android/util/Log.html

# Licencia

**GNU Free Documentation License**

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**0. PREAMBLE**

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

**1. APPLICABILITY AND DEFINITIONS**

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

**2. VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to

the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

**3. COPYING IN QUANTITY**

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

**4. MODIFICATIONS**

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

> A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
>
> B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors

of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

**5. COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

**6. COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

**7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does

not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

**8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

**9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

**10. FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

**11. RELICENSING**

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.