Trabajo Fin de Máster - Máster Universitario en Software Libre

Desarrollo de aplicaciones de Software Libre

Control doméstico visual desde Android *VisualDomo*

Autor: Alfredo Oltra Orengo
Consultor: Gregorio Robles Martínez
Tutor externo: Oriol Palenzuela i Rosés
Empresa asociada: OpenDomo Services S.L

Enero 2015

Publicación

Versión: 0.99.150103
Fecha: 3 de enero de 2015

Documentación generada con:

- Sistema operativo: MacOS Mavericks
- Suite ofimática: LibreOffice 3.6 – es.libreoffice.org
- Edición de imágenes: GIMP – gimp.org
- Planificación de proyectos: Gantt Project 2.6
- Diseño diagramas UML: ArgoUML
- Screencast: Screen Recorder

Copyright

Licencia

Resumen

Abstract

El presente proyecto tiene como objetivo el diseño y la implementación de *VisualDomo*, una aplicación para dispositivos móviles que permite el control de manera visual de dispositivos *ODControl*.

ODControl es un controlador con posibilidad de conexión IP desarrollado por la empresa *OpenDomo Services S.L..* Este controlador permite actuar sobre diferentes dispositivos electrónicos gracias a varios puertos, tanto de tipo analógico como digital, lo que lo hace muy adecuado para el control de instalaciones domóticas.

Con el proyecto se pretende facilitar al usuario el manejo de su instalación domótica, permitiendo definir su localización y controlar cada uno de los aparatos conectados a los puertos de manera sencilla.

Las propia idiosincracia del proyecto implica la posibilidad de abarcar el mayor número posible de dispositivos móviles existentes, no únicamente por configuración física (tabletas o móviles) sino también por la variedad de plataformas existentes. Además, los productos *OpenDomo* están en constante evolución, lo que hace complicado el mantenimiento de varias aplicaciones. Por todo ello se utiliza para el desarrollo de *VisualDomo* el framework *PhoneGap*/*Cordova* que permite, mediante tecnologías web (HTML5+JS+CSS3), crear un único código fácilmente compilable en distintas plataformas.

Por último, el proyecto también tiene carácter académico y educativo ya que se presenta como *Trabajo Final del Master Universitario de Software Libre*, impartido por la *Universitat Oberta de Catalunya (UOC)*.

Palabras clave

OpenDomo, Domótica, Software Libre, Cordova, PhoneGap, Android, iOS

Índice

Índice de figuras

1. Introducción

1.1. OpenDomo

Este *Trabajo Fin de Master* se realiza en colaboración con *OpenDomo Services S.L.* una empresa dedicada a la investigación, desarrollo, instalación y mantenimiento de productos especialmente aplicados al control de instalaciones domóticas y gestión energética avanzada.

Una de sus actuaciones consiste en dar apoyo a una comunidad de desarrolladores que tiene el objetivo de crear una tecnología abierta para el control de instalaciones. Esta comunidad desarrolla varios proyectos tanto hardware como software, siendo su base el *ODControl* del que hablaremos en el siguiente punto.

1.2. Contexto

Como en cualquier otro proyecto, el primer paso a realizar es conocer bien el problema al que nos enfrentamos y definir correctamente dónde queremos llegar. Obviamente en nuestro contexto el protagonista principal es *ODControl*.

Controlador IP

De manera sencilla, un *ODControl* es un controlador IP. Un controlador es un sistema electrónico que, como su nombre indica, permite controlar un conjunto de dispositivos eléctricos que pueden ser conectados a él, para lo cual dispone de una serie de puertos que por una parte pueden ser de entrada o salida y por otra, analógicos o digitales.

El hecho de ser IP lo convierte además, en un dispositivo que puede ser conectado a un red IP, posibilitando el acceso desde otro dispositivos de la red y mediante un navegador web, una consola *ssh* o cualquier otro método, acceder a las funciones que disponen sus entradas y salidas.

Por ver un ejemplo gráfico, el esquema de una supuesta instalación completa (y básica) para controlar un sistema de iluminación podría ser:

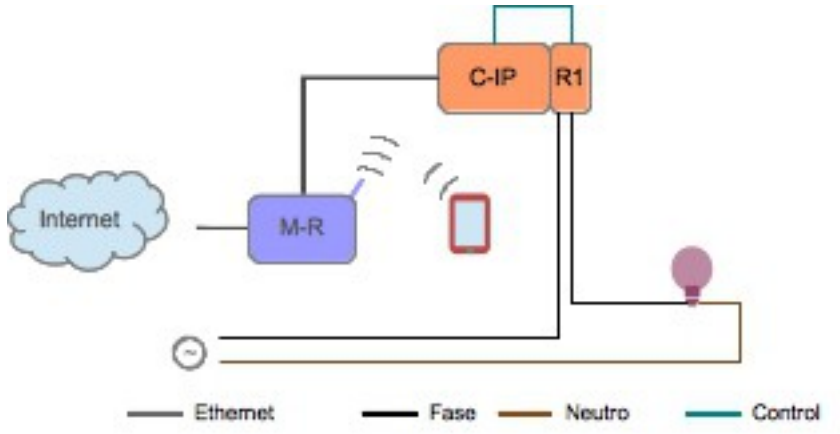


Figura 1: Esquema ejemplo controlador IP

El controlador IP (C-IP) está conectado mediante Ethernet a un modem-router que, por una parte, da salida/entrada a internet y por otra funciona como punto de acceso WiFi. Además el C-IP conecta una de sus salidas a un relé (R1) que hace las funciones de interruptor y que a su vez estará conectado a la fase y al punto de luz a controlar. Dicho control (apagado/encendido) se podrá realizar desde el dispositivo móvil.

ODControl

ODControl es el controlador IP que comercializa *OpenDomo Services S.L.*. En el momento de desarrollo del proyecto, *OpenDomo Services S.L.* tiene disponibles dos versiones (la 1 y la 2).

Figura 2: ODControl

Para nuestro objetivo, no es necesario entrar en detalle en varias de las especificaciones técnicas de cada una, si bien es interesante saber las funcionalidades de ambos productos, especialmente en la versión 2 que es en la que nos centraremos en un primer momento.

ODControl 1

Parámetro	Valor
Número de salidas digitales	8
Número de entradas digitales	8
Número de entradas analógicas	8
Número de salidas analógicas	0
Bus IP (RJ45)	1
Protocolos	UDP, HTTP
Puertos virtuales	48

ODControl 2

Parámetro	Valor
Número de salidas digitales	8
Número de entradas digitales	8
Número de entradas analógicas	2
Número de salidas analógicas	2
Bus IP (RJ45)	1
Protocolos	UDP, HTTP
Puertos virtuales	61

Obviamente, el número de puertos es limitado, luego, según las necesidades de la instalación, puede ser necesario utilizar para cada una de las localizaciones más de un *ODControl*.

La configuración de cada *ODControl* se realiza accediendo mediante un navegador web a la dirección IP asignada al controlador (por defecto 169.254.0.15), lo que da acceso a la aplicación *Configurator*. Desde ella es posible visualizar el estado de los puertos digitales de salida y de entrada, pero también podremos configurar los puertos, ejecutar comandos o realizar tareas de mantenimiento (cambio de IP's, contraseñas, nombre, etc)

Para simplificar la configuración, *ODControl* permite el uso de plantillas predefinidas. El usuario puede crear las que desee y posteriormente replicarlas en otros de sus *ODControl* o descargar desde el sitio web de *OpenDomo* aquellas que le resulten interesantes y que otros usuarios han compartido. Así, por ejemplo, si deseamos una instalación para el control de 8 puntos de luz (plantilla que podemos encontrar en la web) la configuración de los puertos (asignar nombres a los puertos de entrada, activarlos...) simplemente se reduce a cargar la plantilla.

Además de los puertos físicos *ODControl* también posee puertos virtuales. Estos puertos permiten el almacenamiento de manera temporal o persistente de valores para ser utilizados posteriormente. El valor que guardan puede venir calculado por alguna operación booleana o aritmética entre otros dos puertos, por la finalización de un disparador (día, fecha o cuenta atrás) o el resultado de una operación de comparación donde el puerto virtual se activará (1) en caso de que la condición sea cierta.

Por último, el *ODControl* posee una gran cantidad de comandos [1] que pueden ser ejecutados desde el propio *Configurator* o mediante llamadas externas al puerto 81 utilizando el protocolo HTTP. Este tipo de comunicación va autenticada pero no cifrada, por lo que únicamente resulta conveniente su uso en entornos controlados[1]. Como se verá a lo largo del desarrollo este va a ser el principal mecanismo de comunicación entre *VisualDomo* y *ODControl*.

1.3. Estructura de la memoria

Esta memoria se encuentra organizada de la siguiente manera:

- Objetivos: se describen los objetivos que se pretenden alcanzar con la realización del proyecto, tanto a nivel funcional y técnico, como a nivel académico y personal.
- Planificación: esquema de la planificación definida para la consecución de los objetivos planteados.
- Diseño: se detallan los aspectos tenidos en cuenta para el desarrollo de la aplicación y se presentan los resultados obtenidos tras los análisis previos.
- Herramientas: se detallan las tecnologías y herramientas utilizadas.
- Implementación: describe el desarrollo realizado, tanto metodológicamente como a nivel técnico centrándose en los principales problemas encontrados y las soluciones adoptadas en cada uno de ellos.

1- Si se desea realizar el control de manera remota, es decir, a través de internet es muy recomendable utilizar *OpenDomo Lightweight Encryption Protocol (ODLEP) [2]*

- Control de calidad y pruebas: muestra las diferentes pruebas que se han llevado a cabo durante el proceso de creación para obtener un producto dentro de un rango admisible de calidad.

- Resultados y conclusiones: se muestran una serie de tablas con mediciones objetivas sobre el resultado obtenido, así como una valoración de personal y una perspectiva de futuro.

- Vocabulario: explicación de algunos de los de conceptos comentados a lo largo de la memoria.

- Anexos: manual de usuario de la aplicación y licencia bajo la que está protegido.

- Bibliografía: listado de las referencias bibliográficas y páginas de la red mencionadas en el documento o utilizadas como documentación.

2. Objetivos

2.1. Objetivos

El objetivo principal es la creación de una aplicación de código abierto para el control visual de una localización domotizada, controlada por uno o varios *ODControl*. Dicha aplicación debe ser compatible con la gran mayoría de dispositivos móviles, independientemente de su plataforma o modelo. Específicamente la aplicación debe:

- Localizar o permitir configurar los posibles *ODControl* existentes.
- Permitir cargar planos de la localización a controlar.
- Ubicar en los planos los distintos dispositivos a controlar.
- Guardar las configuraciones para simplificar futuros accesos.
- Visualizar el estado de los puertos de entrada, ya sean digitales o analógicos.
- Modificar el estado de los puertos de salida.

En otro ámbito más transversal:

- Es de desear que la aplicación pueda trabajar en varios idiomas (al menos español e inglés).
- Diseño responsable. Dentro de las limitaciones de diseño indicadas por los fabricantes de los sistemas, se pretende que el diseño de la aplicación se adapte al entorno hardware en el que está siendo ejecutado (móvil, tableta o, en un futuro, navegador en equipo de sobremesa).
- Se tendrá especial cuidado en conseguir alta *usabilidad*.

Existen otros objetivos relacionados con la visibilidad del proyecto:

- El código del proyecto será publicado en alguna comunidad virtual, en este caso *GitHub*.
- Documentar el código de tal manera que simplifique la continuidad del trabajo por parte de otros usuarios. Para facilitar la internacionalización de esta labor dicha documentación se hará en inglés.
- Se pretende realizar un seguimiento del proyecto mediante el desarrollo de un blog[2] que muestre tanto el proceso de diseño como el de programación. Dado el carácter abierto de *VisualDomo* se espera que, junto con la publicación del código, el blog facilite la labor de hacer crecer el desarrollo una vez el TFM sea presentado.
- Publicación en las principales tiendas electrónicas de aplicaciones (*Google Play, App Store...*)

A nivel personal, con el desarrollo de *VisualDomo* se busca:

- Adquirir conocimientos sobre dispositivos domóticos.
- Adquirir conocimientos técnicos sobre la creación de aplicaciones multiplataforma para dispositivos móviles.
- Mejorar los conocimientos sobre el uso de bibliotecas *javascript* de apoyo al desarrollo web y de librerías de apoyo al diseño de interfaces web.
- Poner en práctica todos los conocimientos relacionados con los métodos de ingeniería del Software tanto para las fases de diseño como en las de implementación, control de versiones y documentación.

2- www.uhurulabs.com/tag/visualdomo/

2.2. Planificación

El desarrollo del Trabajo Fin de Máster se ha divido en seis fases no necesariamente secuenciales. Las dos primeras han ocupado el primer semestre y tienen un claro carácter de aprendizaje y análisis. Las cuatro restantes tienen un carácter más práctico y conforman el grueso de la implementación y pruebas.

La duración del proyecto ha sido un año académico (Febrero de 2014 – Enero 2015).

Fase 1. Estudio preliminar.

Definición de la idea/objetivos del proyecto.
Estudio de los sistemas *ODControl* de *OpenDomo*.
Estudio del framework de desarrollo *Cordova/PhoneGap*.
Análisis de requisitos y limitaciones.
Definición de recursos necesarios.

Fase2. Diseño de la aplicación.

Diseño funcional.
Diseño lógico.
Creación de *mockups*.

Fase 3. Implementación de la aplicación.

Elección de la metodología de trabajo a aplicar.
Planificación de las fases de implementación.
Instalación y configuración de las herramientas de desarrollo.
Test diarios.

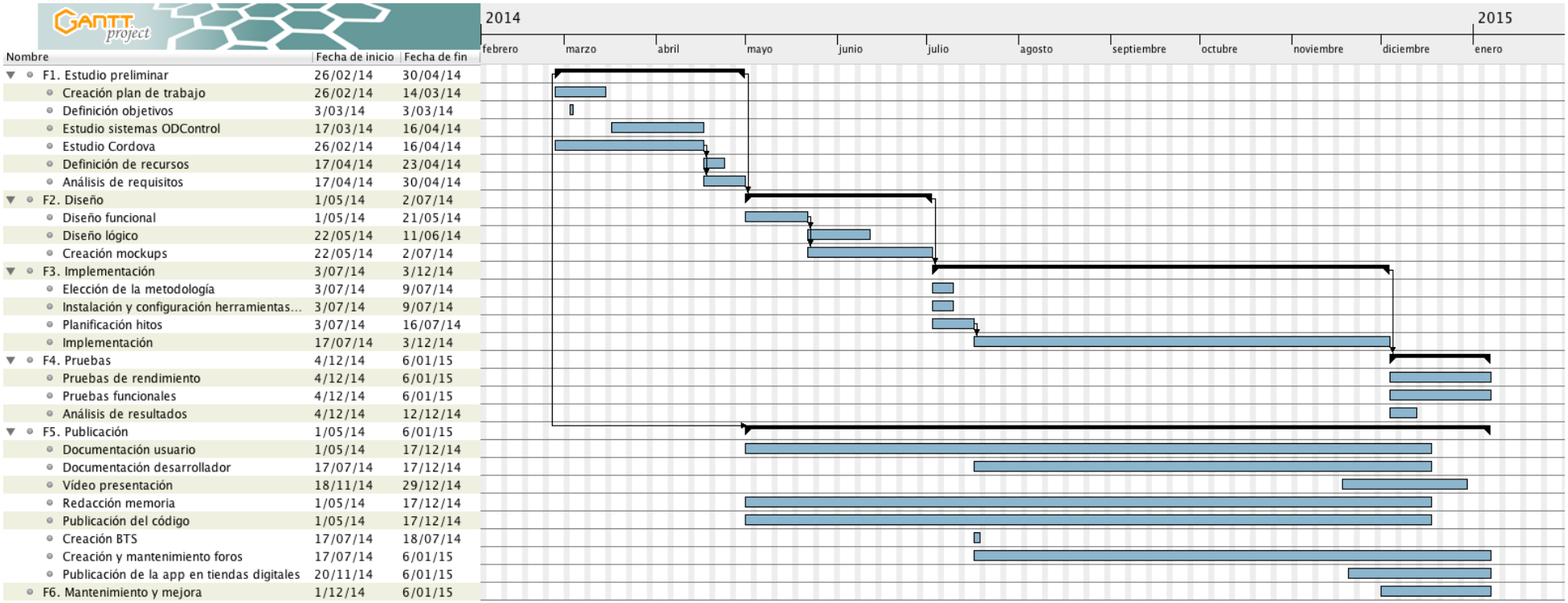
Fase 4. Pruebas en fase beta.

Realización de pruebas de rendimiento en distintas plataformas físicas/lógicas.
Realización de pruebas funcionales.
Análisis de resultados.

Fase 5. Publicación.

Creación del vídeo de presentación de la aplicación.
Redacción memoria.
Publicación del código para uso de la comunidad.
Documentación usuario (ayuda).
Documentación desarrollador.
Creación de un sistema de control de errores (aviso/redacción)
Creación foros de comunicación entre usuarios.
Publicación de la app en tiendas digitales.

Fase 6. Mantenimiento y mejora.



Nombre	Fecha de inicio	Fecha de fin
F1. Estudio preliminar	26/02/14	30/04/14
Creación plan de trabajo	26/02/14	14/03/14
Definición objetivos	3/03/14	3/03/14
Estudio sistemas ODControl	17/03/14	16/04/14
Estudio Cordova	26/02/14	16/04/14
Definición de recursos	17/04/14	23/04/14
Análisis de requisitos	17/04/14	30/04/14
F2. Diseño	1/05/14	2/07/14
Diseño funcional	1/05/14	21/05/14
Diseño lógico	22/05/14	11/06/14
Creación mockups	22/05/14	2/07/14
F3. Implementación	3/07/14	3/12/14
Elección de la metodología	3/07/14	9/07/14
Instalación y configuración herramientas...	3/07/14	9/07/14
Planificación hitos	3/07/14	16/07/14
Implementación	17/07/14	3/12/14
F4. Pruebas	4/12/14	6/01/15
Pruebas de rendimiento	4/12/14	6/01/15
Pruebas funcionales	4/12/14	6/01/15
Análisis de resultados	4/12/14	12/12/14
F5. Publicación	1/05/14	6/01/15
Documentación usuario	1/05/14	17/12/14
Documentación desarrollador	17/07/14	17/12/14
Vídeo presentación	18/11/14	29/12/14
Redacción memoria	1/05/14	17/12/14
Publicación del código	1/05/14	17/12/14
Creación BTS	17/07/14	18/07/14
Creación y mantenimiento foros	17/07/14	6/01/15
Publicación de la app en tiendas digitales	20/11/14	6/01/15
F6. Mantenimiento y mejora	1/12/14	6/01/15

Figura 3: Diagrama Gantt planificación

3. Diseño

3.1. Requisitos

Mínimos

- No será posible acceder a un *ODControl* que no esté en la red a la que se encuentre conectado *VisualDomo.* Dicho de otro modo, no es posible controlar dispositivos externos a la red local. Aún así se preverá un posible acceso futuro a dicha funcionalidad.
- Posibilidad de trabajar con varios *ODControl* de una misma red (mínimo 10).
- El protocolo de comunicación entre *VisualDomo* y los *ODControl* será *HTTP* autenticado (no seguro).
- En una primera fase*, VisualDomo* debe correr en plataformas *Android*. Los cambios entre plataformas deben ser mínimos.
- Se debe permitir la configuración de localizaciones con varias plantas.
- La definición de los planos se hará mediante la carga de algún formato gráfico (png, jpg, etc...).
- La configuración de la localización podrá realizarse *off-line.*
- *VisualDomo* debe guardar las configuraciones de cada localización (puertos, planos, ubicaciones,...) para facilitar una posterior conexión.
- Identificación sencilla por iconos del tipo y estado de cada puerto.
- Control de errores con notificaciones en caso de problemas en los *ODControl*.
- Los estados serán actualizados de manera automática a los *ODControl*, aunque también se podrán solicitar de manera manual.
- Acceso directo o remoto de manual de usuario.
- Licenciado bajo GPL v3.0

Opcionales

- La aplicación permitirá trabajar con varias localizaciones si bien al mismo tiempo únicamente con una de ellas. Aplicación *SDI (Single Document Interface).*
- Soporte para localización, que en este caso se centrará únicamente en el idioma. Al menos dar soporte para dos lenguas (castellano e inglés).

3.2. Actores de la aplicación

Los *ODControl* permite el acceso de dos actores: *user* y *admin*.

admin se encarga de las configuraciones avanzadas del controlador, alejadas de las funcionalidades descritas para *VisualDomo*, por lo que para nuestro contexto únicamente utilizaremos el primero de ellos (*user*). En principio no se considera la creación desde *ODControl* de otro actor diferente a los existentes.

Esta especificación implica que el acceso desde *VisualDomo* a *ODControl* se deberá dar una vez estos últimos se encuentren instalados y configurados correctamente en la localización.

3.3. Casos de uso

A partir de los actores y de los objetivos podemos crear el diagrama de casos de uso.

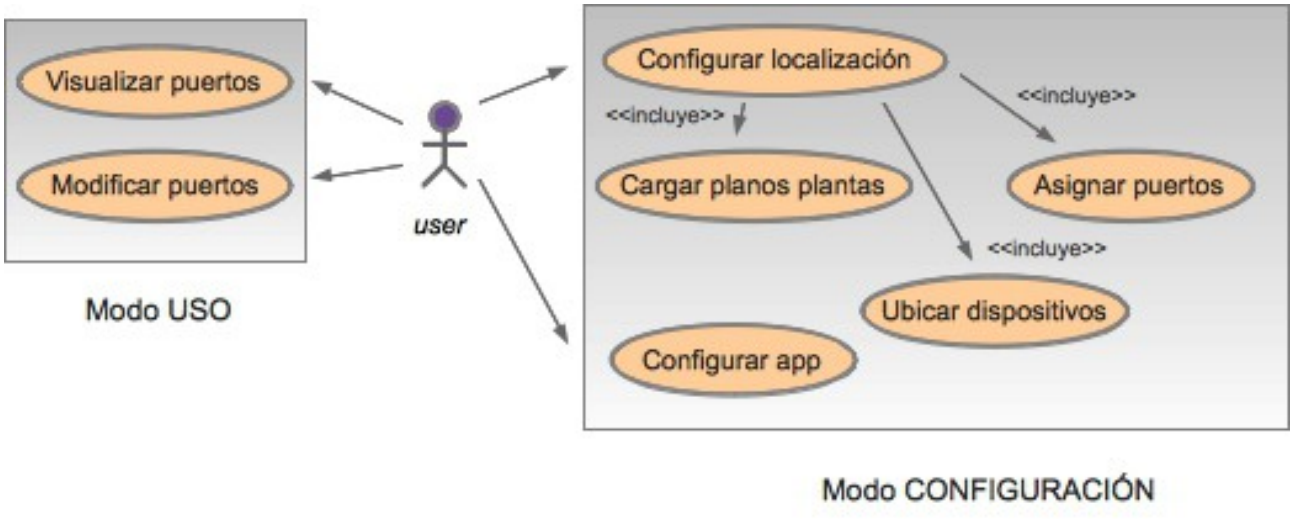


Figura 4: Diagrama de casos de uso

Nuestro único actor (*user*) puede trabajar en dos modos:

- Modo *Visualización.* Será el modo de trabajo habitual. Desde él se podrá visualizar el estado de los dispositivos y, en aquellos que lo permitan, modificarlo. Para poder utilizarse será necesario disponer de conexión a la red de los *ODControl*.

- Modo *Edición.* Permite la configuración de V*isualDomo.* No requiere necesariamente la conexión a la red de los *ODControl,* pero si es recomendable para evitar posibles futuros errores. Su funciones son definir (cargar) los planos de cada planta de la localización, ubicar los dispositivos en ella y, por último, asociar dichos dispositivos a los puertos de los ODControl disponibles.

3.4. Diseño funcional

A la hora de realizar el diseño funcional estudiamos cuatro posibles estados de la aplicación (inicialización, visualizar puertos, modificar puertos y configurar aplicación) mostrando para cada uno de ellos su diagrama de actividades.

Inicialización

Como se comenta en los requisitos mínimos, *VisualDomo* debe trabajar conectado a la red donde se encuentran los *ODControl*, por lo que el primer paso será obtener el *SSID* de la red y, en caso de ser una red conocida cargar la configuración asociada. Si la red es desconocida o no hay configuración asociada se pasará al modo configuración o se descarta la ejecución. En el modo configuración se podrán definir las plantas, cargar los planos, ubicar dispositivos y enlazarlos con los puertos correspondientes de cada *ODControl*. Una vez la configuración se haya realizado y haya conexión a la red, se procederá a obtener las credenciales y actualizar el estado de los dispositivos.

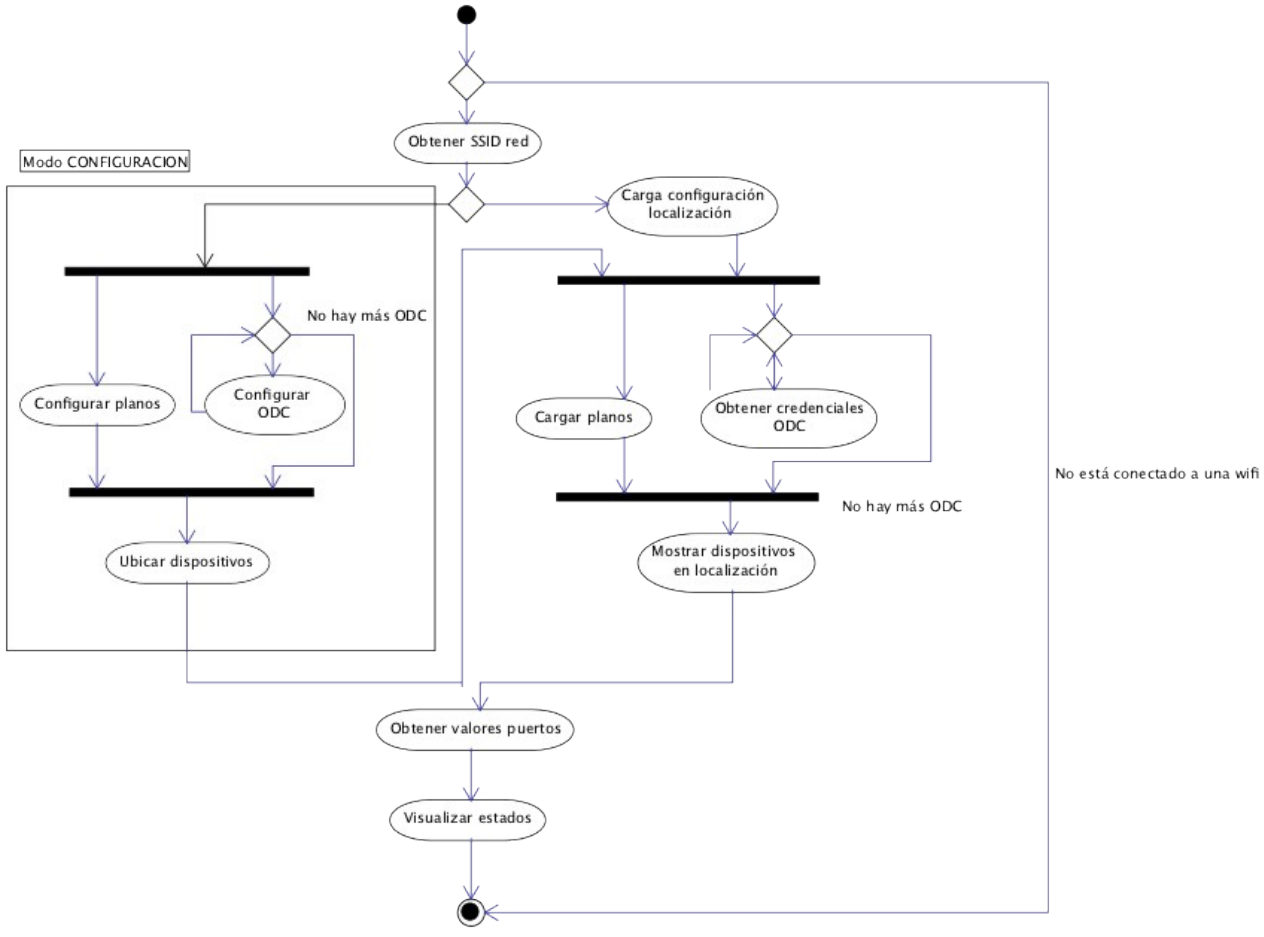


Figura 5: Diagrama de actividades. Inicialización

Visualizar puertos

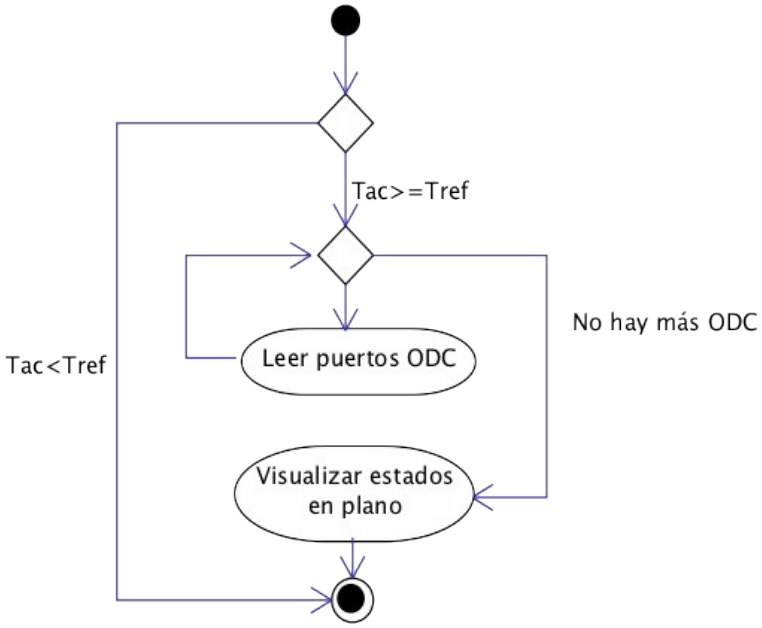


Figura 6: Diagrama de actividades. Visualizar puertos

Los puertos del *ODControl* no mandan ningún tipo de señal (al estilo de una interrupción) para indicar cuando su estado ha cambiado, por lo que debe ser *VisualDomo* el que los interrogue cada cierto tiempo (*Tref*) para evaluar su estado. En caso de que *Tref* sea mayor o igual al tiempo pasado desde la última encuesta (*Tac*) la información volverá a ser solicitada. Por otra parte, el proceso de encuesta podrá ser lanzado de manera manual siempre que el usuario lo desee.

Modificar puertos

Como puede verse en la *Figura 7*, el proceso de modificación de puertos resulta muy sencillo. Simplemente una vez modificado por parte del usuario el puerto *VisualDomo* lo notificará al *ODControl* correspondiente y esperará confirmación, momento en el que actualizará el estado en la pantalla.

Configurar aplicación

La configuración de la aplicación funciona de manera muy similar a la de modificación de los puertos, si bien en este caso no es necesario el acceso a ningún *ODControl*.

Es importante resaltar que *VisualDomo* no permite la configuración de ningún aspecto del *ODControl*. Ese tipo de acciones (como cambio de contraseñas, IP, etc) se harán desde un navegador web accediendo a la aplicación *Configurator* del *ODControl.*

Figura 7: Diagrama de actividades. Modificación de puertos

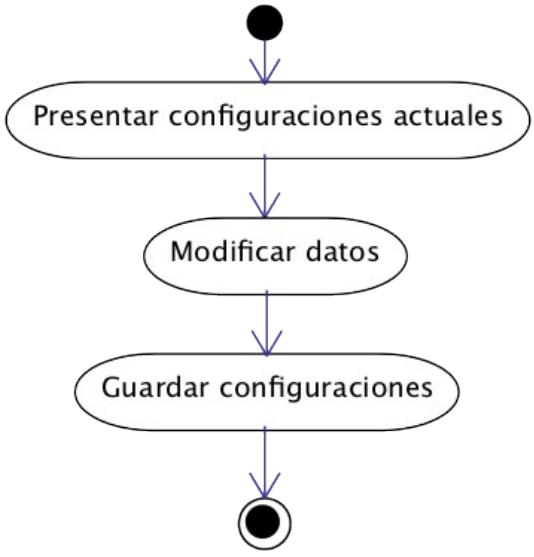


Figura 8: Diagrama de actividades. Configurar aplicación

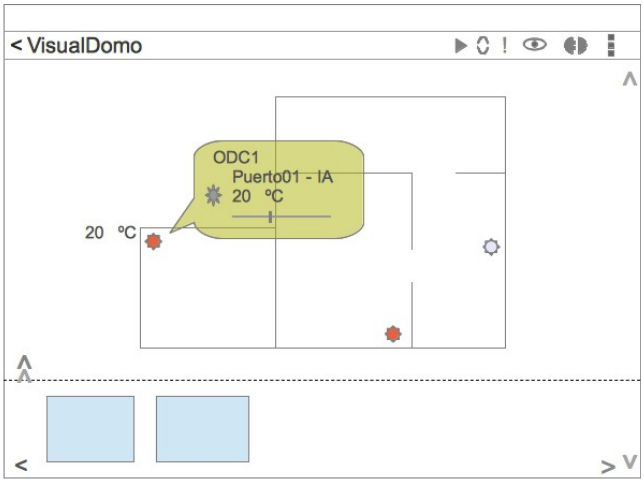
3.5. Mockups

Antes de empezar a trabajar, resulta muy importante tener una imagen de cual es nuestro objetivo en lo que respecta a la interfaz. De manera muy esquemática indicamos cuáles son los controles (indicando el objetivo de cada uno de ellos) que deberán existir en cada una de las partes principales de *VisualDomo*. Es muy probable que esta representación vaya evolucionando a lo largo de la implementación y difiera con el resultado final (ver capítulo *Resultados y conclusiones*), pero hace funciones de guía tanto para el diseñador/programador como para el cliente, que puede hacerse una idea de las pantallas, proponer nuevas ideas y así evitar sorpresas finales.

Visualización/modificación de puertos

Esta pantalla engloba el funcionamiento del *Modo Uso*. El objetivo principal es la visualización de la planta con la ubicación de los dispositivos. Entre sus controles tenemos:

- Zona de visualización de planos.
- Cambiar de modo uso a modo configuración.
- Cambiar modo visualización: una planta / dos plantas / tres plantas / cuatro plantas.
- Si sólo se visualiza una planta, dejarla fija o cambiar de planta cada cierto tiempo.
- Cambiar planta.
- Cambiar estado de puerto.
- Obtener información del puerto.
- Forzar actualización.
- Cambiar localización.
- Acceso a configuración app.



Al pulsar sobre el dispositivo éste cambiará su estado automáticamente en caso de ser digital. En caso de ser analógico, se mostrará el control que permita regular su valor. La pulsación continuada mostrará más información.

Configuración localización

Esta pantalla engloba el funcionamiento del *Modo Configuración*. El objetivo principal es la carga de los planos de cada una de las plantas así como la ubicación y configuración de los dispositivos. Entre sus controles tenemos:

- Zona de visualización del plano.
- Cargar plano/planta.
- Miniaturas planos cargados.
- Eliminar planta.
- Listado de puertos disponibles.
- Eliminar asignación.
- Guardar/cancelar configuración.
- Asignar datos generales de configuración de la localización: nombre...
- Notificaciones: errores, estado (offline/online).

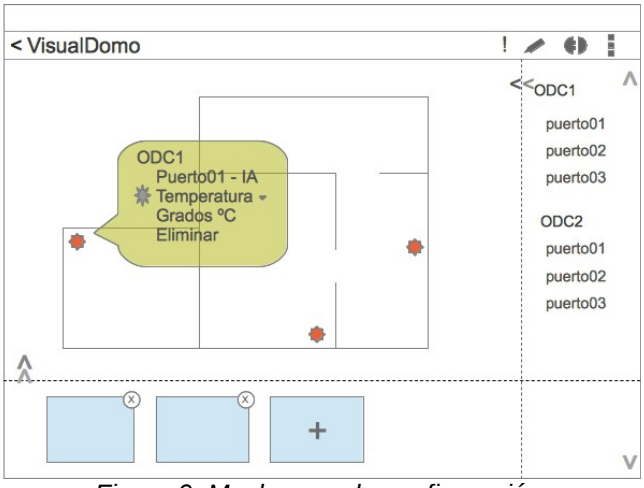


Figura 9: Mockup modo configuración

El usuario arrastrará el puerto a la posición que desee. Una vez ubicado deberá introducir el tipo de dispositivo y si fuera necesario las unidades de medición. En caso de que el tipo no esté asignado, el icono de dispositivo indicará error.

Configuración aplicación

Esta pantalla permite la configuración de los datos de la aplicación. Su esquema es muy sencillo. Entre sus controles tenemos:

- Listado variables a configurar.
- Aceptar/Cancelar configuración.

Figura 10: Mockup pantalla de configuración

Enlace/selección de localizaciones

La pantalla ha de mostrar la lista de todas las localizaciones creadas y permitir enlazar con la red wifi actual o cargarla en la ventana de configuración para su modificación. Al ser pantallas muy similares se modelará como una única y solamente se cambiarán ciertos detalles. Su esquema también tiene que ser muy sencillo.

- Listado de localizaciones existente.
- Información sobre la localización mostrada.
- Botón para asignar/desasignar wifi (modo enlace).
- Botón para seleccionar localización (modo selección).

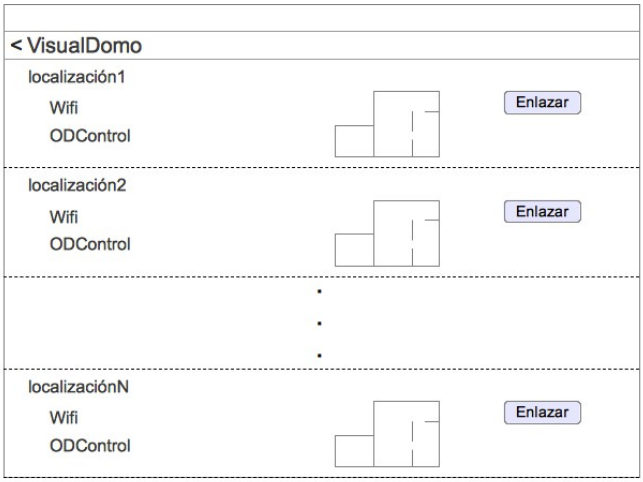


Figura 11: Mockup pantalla selección/enlace

4. Herramientas

4.1. PhoneGap

De manera resumida, *PhoneGap* es un *framework* para la creación de aplicaciones móviles multiplataforma. Es decir, su objetivo es crear un único código y que éste pueda funcionar en distintas plataformas móviles como *iOS, Android, Blackberry, FirefoxOS, Windows Phone*...

En general, para conseguir un objetivo así, la idea es crear aplicaciones web con *HTML5, CSS3* y *javascript* que corran sobre un navegador. Evidentemente este método proporciona multiplataforma, pero por contra hace las aplicaciones más lentas (necesitan al navegador entre ellas y el SO) y por obvios motivos de seguridad limita el acceso a varios recursos como la cámara, el acelerómetro, etc.

Para solucionar en parte este problema *PhoneGap* permite la creación de aplicaciones híbridas, es decir, añadir ciertos componentes (que llama *plugins*) que conectan de manera nativa con esos recursos. Lógicamente esto hace que la aplicación pierda la posibilidad de ser multiplataforma, pero la manera de migrar a otras plataformas queda únicamente reducida a crear los plugins en ellas. Estos accesos se desarrollan en *javascript*, que si permite la llamada a código nativo (ya sea *Java*, *C, C*++...). Aunque *PhoneGap* ya proporciona varios de estos plugins por defecto, el usuario puede crear aquellos que necesite para su proyecto.

De esta manera una aplicación desarrollada con *PhoneGap* no es una simple página web, sino que es una aplicación nativa (un *apk* si hablamos en sistemas *Android*) que en su interior lleva un componente visualizador *Web* (*WebView*), las páginas web creadas y los *plugins* de acceso a los recursos.

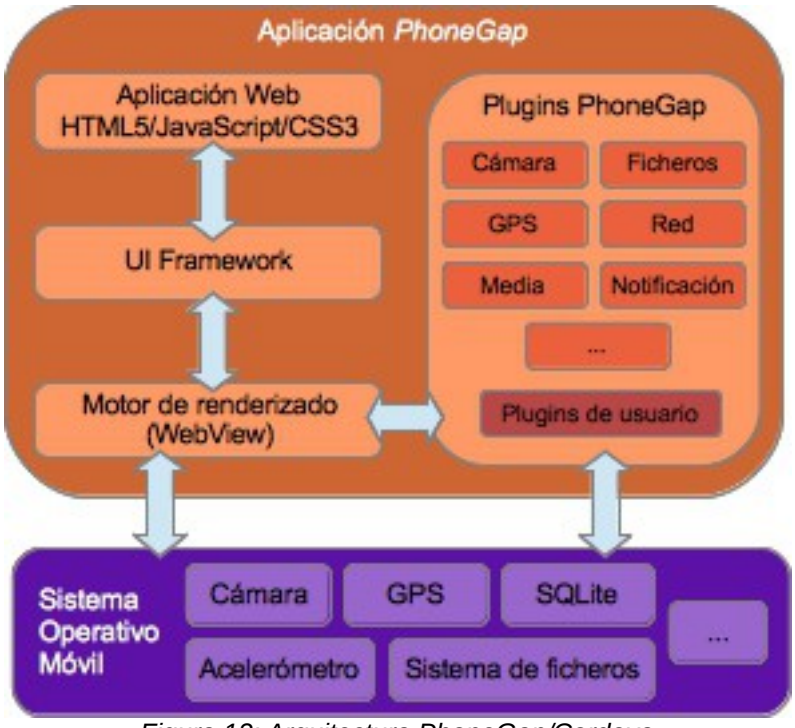


Figura 12: Arquitectura PhoneGap/Cordova

Por otra parte, para entender el concepto de *framework* podemos relacionarlo con *Java* o *.NET*. Este concepto incluye herramientas que ayudan al proceso de compilación, API's, etc. y que generalmente se ejecutan sobre una máquina virtual. En este caso podríamos hacer la relación de máquina virtual/*webView*. Para entenderlo mejor podemos ver en la página anterior el esquema de bloques de una aplicación *PhoneGap.*

El bloque *UI framework* no es necesario. Evidentemente, podemos desarrollar las aplicaciones desde cero con *CSS3* y *Javascript*, pero es obvio que si ya hay diferencias entre distintos navegadores más las puede haber ahora que entra además el factor plataforma. Es por ello que resulta muy interesante el uso de algún tipo de librería como *JQuery Mobile* que ayude a la creación sencilla del interfaz.

Por último queda otro pequeña confusión a aclarar: ¿qué es *Apache Cordova* y qué tiene que ver con *PhoneGap*? Hemos estado hablando de *PhoneGap*, pero realmente tendríamos que haberlo hecho de *Apache Cordova*. *PhoneGap* fue desarrollado por *Nitobi*, empresa que al poco tiempo fue comprada por *Adobe*. *Adobe* ha cedido la licencia a la *Fundación Apache* para liberar el código y asegurar que ese tipo de licencia se mantenga. De esta manera el producto pasó a denominarse *Apache Cordova* y *PhoneGap* a ser una distribución de él. A día de hoy la única diferencia es el servicio de compilación en la nube que *PhoneGap* ofrece.

4.2. JQuery Mobile

Como acabamos de comentar, la posibilidad de disponer de una librería que nos permita simplificar la implementación, tanto del interfaz gráfico como la integración de la página con el dispositivo móvil, es una idea que más que ser meramente práctica se antoja casi fundamental.

De la misma manera que surgió *JQuery* para simplificar el desarrollo web y homogeneizar el trabajo con varios navegadores, con la aparición de los dispositivos móviles y las nuevas necesidades en la representación e interacción con las páginas web desde ellos, se hace necesaria la aparición de nuevas librerías que contemplen estas funcionalidades. Estas librerías no sólo nos permiten adaptarnos a dichos escenarios, si no que poseen widgets específicos que simulan los controles habituales de los dispositivos compatibles.

A la hora de elegir la librería a utilizar existen varias posibilidades en el mercado. La elección resulta, casi con total seguridad una de las más complicadas, ya que muchas de ellas están muy bien integradas con *Cordova* y poseen características muy interesantes. Aun así, posiblemente los dos más utilizadas sean *Sencha Touch 2* y *JQuery Mobile*.

Se realizaron algunas pruebas en las primeras fases del desarrollo de las que se obtuvieron las siguientes conclusiones:

JQM
Respuesta a las acciones aceptable en sistemas *Android* y buena en *iOS*.
Complejidad a la hora de simular los *widgets* nativos de Android sin librerías de terceros.
Desde el punto de vista personal, facilidad de aprendizaje ya que está basada en *Jquery.*

Sencha Touch
Widgets mucho más adaptados a cada uno de los entornos.
Respuesta a las acciones buena en sistemas *Android* y buena en *iOS*.
Personalmente más compleja de aprender ya que se parte de cero.

Dado que los rendimientos obtenidos era aceptables en ambos sistemas, que se pretendía diseñar el interfaz desde cero sin tener en cuenta los elementos nativos y que el tiempo era un factor muy importante, la decisión se centró en la facilidad de aprendizaje, por lo que se optó por el uso de *JQuery Mobile.* Entre sus característica tenemos:

- Temas personalizados. El *framework* permite el uso de temas ya creados y da la posibilidad de crear nuevos temas y trabajar con ellos.
- Tamaño reducido. Toda la librería comprimida pesa menos de 12K.
- Facilidad de uso.
- Múltiples plataformas: iOS, Android, Blackberry, Palm WebOS, Symbian, Windows Mobile, etc.
- Soporte HTML5.
- Open source (bajo licencia MIT)

4.3. Entorno de trabajo

Además de la elección de la tecnología de apoyo para la creación de proyecto, resulta vital la elección de un buen entorno de trabajo, entendido como tal el conjunto de herramientas software que nos van permitir desarrollar nuestro trabajo. Esta elección es un factor clave a la hora de obtener un resultado satisfactorio en el desarrollo del proyecto ya que posibilitará o no un trabajo rápido, con ayudas, de fácil acceso, con herramientas de depuración adecuadas, etc. Además el programador ha de sentirse cómodo con él, por lo cual parte de la elección puede deberse a experiencias anteriores, conocimientos previos, o simplemente gustos personales.

OSX Mavericks

El primer escalón en el entorno de trabajo es el sistema operativo. El equipo en el que se ha centralizado el desarrollo ha sido un portátil *MacBook* con el sistema operativo *OSX Mavericks* instalado. Dentro de todo el entramado de nuestro entorno es la única pieza que no dispone de licencia abierta.

Aunque durante las primeras fases de pruebas y desarrollo se trabajó sobre un *Linux Mint 15*, la elección definitiva del sistema *Mavericks* se debió a la posibilidad de generar la aplicación tanto para plataformas *Android* como *iOS* (posibilidad que finalmente no se ha podido implementar por falta de tiempo) ya que el paquete de desarrollo de aplicaciones para esta última sólo se encuentra disponible en *MacOS*.

Brackets

La mayor parte del código generado en un aplicación *PhoneGap/Cordova* es *HTML, Javascript* y *CSS* por lo que parece indicado la elección de una herramienta de codificación pensada para ese entorno web. De entre las existente se ha elegido *Brackets* por varias razones:

- Desarrollo enfocado a tecnologías web.
- Existente en varias plataformas (*MacOS, Linux, Windows*).

- Integración dinámica con el navegador (*Chrome*) de tal manera que las modificaciones realizadas en el código son visualizadas en tiempo real.
- Desarrollo *inline*, de manera que se permite saltar directamente desde el código javascript o HTML de un elemento a su representación en CSS.
- Aplicación open source.

Por otra parte se encuentra en constante evolución, con incorporación de nuevas características y corrección de errores cada dos o tres semanas.

Aunque muchas de las recomendaciones existentes a la hora de elegir IDE van dirigidas al uso de *Eclipse* (ver siguiente apartado), como apunte personal se considera que ha sido un buena elección ya que posiblemente sea el editor de tecnologías web con mayor apoyo visual gracias a su integración dinámica con el navegador. Esa característica ha permitido equilibrar el ritmo de trabajo, ya que de partida los conocimientos en CSS eran menores al resto de tecnologías usadas.

Eclipse

Ya que la plataforma de destino inicial va a ser *Android*, la instalación de *Eclipse* resulta muy adecuada por dos motivos. Por una parte nos va a permitir, con el uso de los *plugins* correspondientes, hacer el mantenimiento de los paquetes de necesarios para la compilación sobre Android. Por otra nos sirve de base para la creación de *plugins* para específicos de Android para *PhoneGap/Cordova*. Realmente estos *plugins* se pueden compilar directamente el compilador de *PhoneGap/Cordova*, pero la detección de errores resulta compleja (más cuanto más complicado es el *plugin*) por lo que la creación de una pequeña aplicación de test para *Android* puede con las funcionalidades buscadas puede facilitar la tarea.

Emulador/dispositivo Android

Por motivos de eficiencia, la mejor manera de realizar las pruebas de la aplicación es mediante un dispositivo físico. En este caso se ha usado un móvil *Galaxy Nexus (Android 4.2)* y una tableta *Nexus 7 (Android 4.4,* pero migrada en las últimas fases de desarrollo a *Android 5.0)*

Aún así, como se comenta en el apartado de *Implementación,* se hace necesario el uso de software adicional para poder depurar la aplicación de la mejor manera posible. En este caso se ha utilizado *Ripple Emulator*.

Chrome

El uso de un navegador a la hora de realizar una aplicación web resulta, obviamente, indispensable. En general la elección podría ser cualquiera, pero se ha elegido *Chrome* ya que además de soportar *HTML5* de una manera muy completa, nos proporciona integración con *Brackets* para la visualización de la web en tiempo de redacción y la depuración remota de páginas web situadas en dispositivos externos. Esta opción resulta fundamental a la hora de trabajar en un proyecto de este tipo donde muchas de las opciones son generadas dinámicamente a través de las librerías.

Birdfont

Una de las técnicas más habituales para facilitar la creación de aplicaciones *Responsive* consiste en el uso de fuentes tipográficas de iconos de manera que su tamaño pueda ser cambiado con facilidad. Eso ha llevado a la creación de una fuente denominada *VisualDomo q*ue incluye los elementos gráficos utilizados en la aplicación. Para su creación se ha utilizado el programa

Birdfont. Entre sus características más importantes:

- Importación de svg.
- Facilidad de uso.
- Permite la exportación de fuentes a varios formatos.
- Es libre.

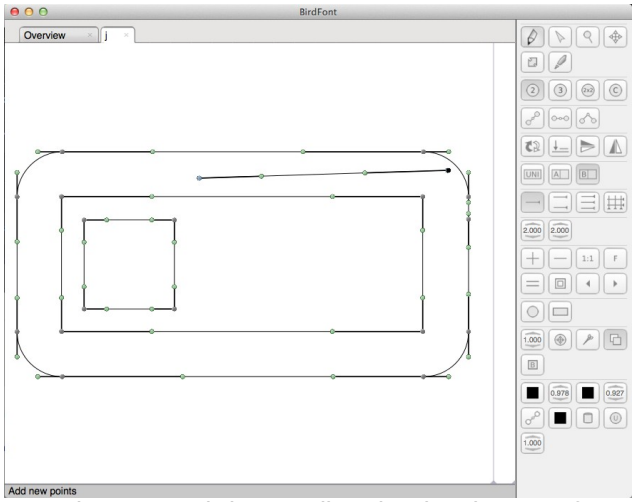


Figura 13: BirdFont editando VisualDomo.ttf

Git/GitHub

Si el uso de un sistema de control de versiones es fundamental hoy día en cualquier tipo de proyecto software, en el caso de desarrollos libres es casi obligatorio. Un sistema de control de versiones nos va permitir entre otras muchas cosas recuperar código perdido, mantener varias versiones, experimentar con nuevas características sin interferir con el código de producción o facilitar el trabajo en equipo.

Existen en el mercado multitud de CVS, pero sin duda el referente libre actual está siendo *Git*. Surge por la necesidad de mantener el código del kernel de *Linux* de una manera más estable, un proyecto de grandes dimensiones que no sólo maneja una gran cantidad de archivos, si no que además implica a muchos usuarios con desarrollos en paralelo.

Resulta complicado saber por qué hoy en día *Git* ha llegado ser un estándar de facto ya que al final los factores que intervienen son muchos, pero podríamos incluir:

- Existen muchos interfaces gráficos para el manejo de *Git.*
- Cada vez es más usado por la comunidad (especialmente la *open source*).
- Su escalabilidad es excelente. Permite trabajar igual de bien con proyectos pequeños como grandes.
- Es libre.
- Es distribuido. Los proyectos cada vez son más grandes y más modularizados. La distribución permite que varios equipos trabajen con sus propias ramas y revisiones sin interferir en el trabajo diario del resto de grupos.

Por otra parte *GitHub* es un alojamiento web para repositorios *Git* que ha cogido mucho peso en los últimos años, en parte por la gran cantidad de proyectos libres que alberga por su pequeño pero potente sistema de trabajo en grupo, que permite la creación de documentación, la creación de hitos (*milestones*), de cuestiones a solucionar (*issues*), etc

La elección de ambos sistemas (obviamente uno es dependiente del otro) se realiza por recomendación de *OpenDomo Services SL*, que aloja de esa manera todos sus desarrollos abiertos.

LibreOffice/Gimp

En las primeras fases de desarrollo se creó un trabajo de arte que conducía a la creación del logotipo de la aplicación. El diseño vectorial se realizó con el apoyo del módulo de diseño gráfico que se incluye en el paquete ofimático *LibreOffice*. Para la parte de retoque fotográfico se utilizó *Gimp.*

5. Implementación

5.1. Metodología

Se ha optado por el uso de una metodología mixta. Durante las primeras fases se ha utilizado un ciclo en cascada, es decir, en primer lugar analizar, posteriormente diseñar, y a continuación implementar. Sin embargo en esta última fase de implementación y en la posterior de pruebas su utiliza una metodología SCRUM.

Aprovechando las facilidades que *GitHub* ofrece para la gestión del proyecto se crearon una serie de pequeñas tareas agrupadas en hitos que a su vez se etiquetan en versiones menores. La definición de los hitos se realiza de manera progresiva en función de las nuevas funcionalidades a añadir, a mejorar, a incorporar o errores a corregir.

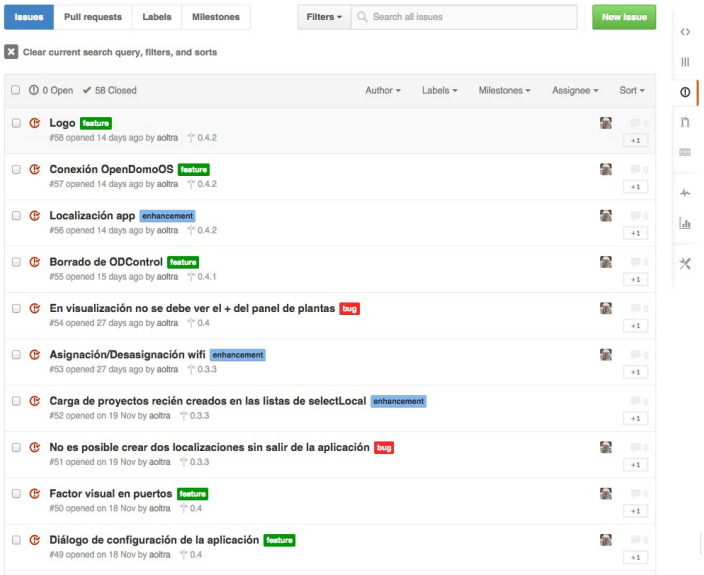


Figura 14: Tareas (Issues) VisualDomo en GitHub

Esta elección de la metodología se basa en que proporciona diferentes velocidades de trabajo según en la fase en la que nos encontremos. El primer problema con el que nos encontramos es el desconocimiento del producto (*ODControl*) y en gran medida de la tecnología a aplicar (*Cordova*). Es por ello necesario dedicar una gran cantidad de tiempo a centrar adecuadamente el problema. Posteriormente el problema principal pasa a ser la creación de una correcta planificación que permita alcanzar los objetivos adaptados a la disponibilidad existente así como entrar ya en el detalle de los posibles problemas técnicos que puedan surgir. En esta situación es más conveniente una metodología *SCRUM* que ha permitido:

- Reaccionar con rapidez en caso de aparición de errores no esperados o de planteamientos demasiado ambiciosos.

- No atascar el desarrollo en momentos de poca disponibilidad al permitir crear hitos de pocas tareas o de tareas más sencillas.

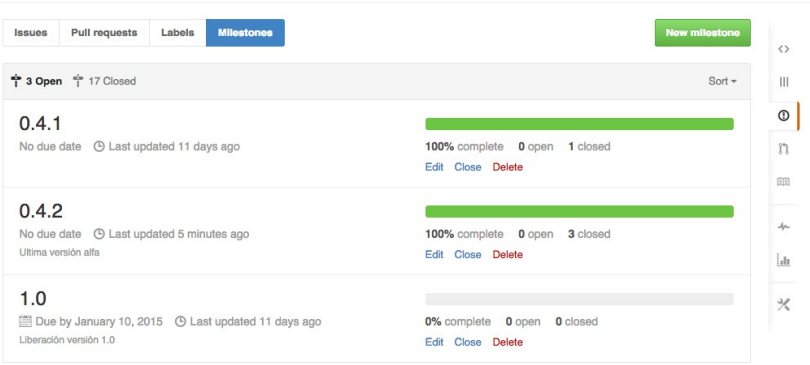


Figura 15: Hitos (milestones) VisualDomo en GitHub

5.2. WifiInformation

Una de las funcionalidades de *VisualDomo* es la asignación de la localización a una determinada red Wifi. Eso permite evitar confusiones al usuario pero además, cargar directamente la localización indicada en función de la red wifi en la que nos encontremos.

Para implementar esta característica es necesario obtener el identificador unívoco de la wifi. Esta funcionalidad de más bajo nivel requiere el acceso a hardware por lo que escapa de la API de *Cordova* y es necesario implementarlo mediante *plugins*.

La primera opción es obtener dicha funcionalidad de los plugins ya existentes en el mercado, pero todos aquellos relativos al acceso a red no llegan a proporcionar dichos datos, por lo que la única opción era crear nuestro propio *plugin*.

La creación de un *plugin* supone la perdida de multiplataforma, en la medida en que no desarrollemos versiones específicas del mismo para cada una de las plataformas. En este caso y por motivos de tiempo, únicamente se ha desarrollado el destinado a *Android*, pero sí se realizó un estudio referente a otras plataformas como *Windows Phone* o *iOS* y se comprobó que la implementación no resulta demasiado compleja.

En líneas generales la implementación consiste en la creación dos ficheros:

- Fichero .js (javascript) que define el objeto que será utilizado desde *Cordova* para llamar a las funciones del *plugin*. En este caso el objeto creado es *wifiinfo* y posee dos funciones, *getBSSID* y *getSSID*. Ambas dos funcionan de manera asíncrona y pueden recibir funciones de retorno en caso de éxito o fracaso de la ejecución.

- Fichero .java que implementa la funcionalidad directamente. Este fichero contiene dos funciones, una encargada de crear el contexto de la aplicación a partir del Interface con *Cordova* y una segunda que llama o implementa directamente cada una de las funcionalidades.

Al igual que *VisualDomo*, el proyecto se licencia mediante GPL 3.0 y todo su código está disponible en *GitHub*[3].

3 *https://github.com/aoltra/WifiInformation*

5.3. Estructura

El propio *framework* obliga a definir una estructura de archivos determinada a la hora de trabajar, de tal manera que se almacenan por separado los *plugins*, el código de específico de cada plataforma, elementos comunes... y el código principal desarrollado separando por una parte *Javascript* y por otra CSS.

El código se estructura en varios ficheros que podríamos diferenciar en tres tipos:

- Definición de objetos. Encapsulan la funcionalidad de los objetos básicos de la aplicación como son la *Localización*, el *ODControl*, los *Puertos* o la *Planta*.

- Definición de pantallas. Encapsulan la funcionalidad de las pantallas (no cuadros de diálogo) de la aplicación, en este caso la ventana principal, la de selección de localizaciones (o enlaces) y visualización.

- *Helpers*. Encapsulan funcionalidades de ayuda en distintos aspectos como el dibujo o el acceso al sistema de ficheros. Se encuentran en el carpeta *www/js/helpers.*

5.4. Acceso al sistema de ficheros

Cordova, a través de la API *file access* de *HTML5,* permite el acceso al sistema de ficheros del dispositivo de manera que podemos crear, editar y/o borrar elementos. Las primitivas proporcionadas por esta API son relativamente básicas y en caso de accesos más o menos complejos es necesario construir funciones a partir de ellas.

La librería permite el acceso a los ficheros de manera síncrona y asíncrona. El modo síncrono es, a priori, más sencillo de implementar y de entender pero en la práctica resulta muy confuso. En un escenario y con unas tecnologías claramente pensadas para el entorno asíncrono introducir una funcionalidad síncrona resulta peligroso si no se tiene muy bien acotada la zona de actuación. Es habitual encontrar respuestas asíncronas en modelos supuestamente síncronos, debido a que las llamadas a estos últimos se realizan desde entornos asíncronos (que es lo más habitual). Tanto por esta razón como por dar mayor sensación de fluidez a una aplicación que por sus tecnologías y dispositivos puede resultar lenta, la implementación utilizada es la asíncrona.

Con el objetivo de mejorar el encapsulamiento y la reutilización, se crea un fichero *FileHelper.js* que implementa el objeto *helpFile* con funciones para la creación *recursiva* de directorios, borrado de directorios, lectura de entradas de directorios, etc

Modelo de datos

A la hora de almacenar la información de cada localización se utiliza un fichero en formato JSON, de fácil lectura y escritura con las herramientas del *framework*. El problema surgió con el almacenamiento de las imágenes de cada una de las plantas de la misma.

La primera solución adoptada fue la de almacenar enlaces a las imágenes existentes. El objetivo era reducir el uso de espacio en dispositivos que en general cuentan con poca capacidad de almacenamiento y evitar de paso el proceso de estructuración y copia de los ficheros. Este tipo de formato almacenaba correctamente pero, por motivos de permisos, no permitía leer si la imagen se encontraba fuera del ámbito de acceso la aplicación (lo que es la situación más habitual).

Al final se optó por ampliar el formato de almacenamiento a un fichero más un directorio, donde se almacenan copias de las imágenes originales. Este desarrollo ha implicado una mayor dedicación pero permite dar más independencia al sistema.

Acceso a galería

Independientemente de la forma de almacenamiento también es necesario implementar la manera de acceder al sistema de ficheros para obtener tanto las localizaciones como las imágenes.

Pensando en las imágenes, en un principio se optó por crear un diálogo de acceso al sistema, pero posteriormente se prefirió optar por utilizar el *plugin camera,* que además de permitir el control de la cámara del dispositivo, permite el acceso directo a toda la galería del sistema, lo que permite disponer no sólo de imágenes obtenidas por la cámara, sino de las generadas por otro tipo de aplicaciones.

Almacenamiento de la configuración de la aplicación

A la hora de almacenar los datos de configuración de *VisualDomo* se opta por la definición un objeto *Settings* con los elementos a guardar y el almacenamiento directo de manera persistente mediante *localStorge [3] la tabla VDSettings.*

5.5. GUI

JQuery Mobile proporciona una colección de funciones para la creación del interfaz gráfico de la aplicación, incluyendo los *widgets* habituales como botones, listas, etc. Su aplicación resulta sencilla, pero no está exenta de problemas.

Por ejemplo, la creación de los cuadros de diálogo para solicitar a los usuarios cierto tipo de información, resultó muy problemática al entrar en conflicto varios de los valores del diseño propio con los heredados desde CSS. Hubo que realizar varias depuraciones para conseguir conciliar ambos funcionamientos.

Otro de los problemas surge al querer simular llamadas a cuadros de diálogo desde los menús de la parte superior. Tanto los *menús* como los cuadros de diálogo se implementan con el widget *popup y Jquery Mobile* impide la llamada desde un *popup* a otro. Para solucionarlo se ha tenido que capturar el evento de cierre del *popup*, esperar cierto tiempo y lanzar manualmente el cuadro de diálogo.

```
popupafterclose: function () {
      setTimeout(function () {
            $('#popup-conf-location').popup('open');
      }, 100);

      $('#config-menu').off('popupafterclose');
}
```

Responsive Design

El motivo principal de utilizar *Cordova* para el desarrollo de la aplicación es, claramente, la posibilidad de trasladar a muchas plataformas la aplicación con poco esfuerzo. Pero, aunque generalmente nos estamos refiriendo a plataformas software, está claro que la fragmentación existente en cuanto a los dispositivos hardware dentro de una misma plataforma, afecta también (y mucho) a la hora de poder ejecutar en condiciones la aplicación.

Este tipo de aplicación está pensada, principalmente, para el trabajo con dispositivos móviles de una tamaño de pantalla mínimo de entre 6 ó 7 pulgadas, pero aún así, se intenta que pueda ser ejecutada en dispositivos más pequeños con cierto grado de funcionalidad.

Para ello en los CSS se ha optado por crear en algunas pantallas *media queries* que ajustan de la mejor manera posible las características de la aplicación a la del dispositivo.

5.6. Drag & drop

Otro de los retos era la implementación de la técnica de *Drag&Drop* para ubicar los puertos en cada una de las plantas. Gracias a *JQuery Mobile* una primera implementación funcional no fue difícil de conseguir, pero su rendimiento dejaba bastante que desear.

Para la mejora del mismo se ha optado por estudiar cada uno de los posibles casos y eliminar las máximas redundancias. Así, por ejemplo, se pasó de dibujar directamente sobre el *canvas* cada uno de los puertos, a crear una fuente basada en iconos que facilitaba el redibujado en tiempo de arrastre. La fuentes, denominada *VisualDomo*, se basa en otra fuente libre denominada *entypo*[4] *[4].*

Otras posible mejora a incorporar (no se han podido llevar a cabo por falta de tiempo) consistiría en la aplicación de un doble *canvas* a modo de capas, una para mostrar únicamente el plano de la planta y otro para realizar las tareas de *Drag&Drop*. Esto permitiría aligerar todo la parte del proceso de cálculo, pero crearía un nuevo problema con la sincronización entre *canvas,* sobre todo pensando en un futuro en el que se incorporasen funcionalidades como zoom's o visualización de varias plantas de manera simultánea.

5.7. Localización: jquery.auderoTextChanger

La localización de la aplicación (únicamente en lo referente al idioma) es muy sencilla en lo que respecta al código *Javascript*. Únicamente se declara un objeto *Translation* con varias entradas en función del idioma. Posteriormente a la hora de mostrar un determinado texto se accede a él por idioma.

El problema surge en la modificación del código HTML. Para su traducción se optó por el uso de una librería desarrollada por Aurelio De Rosa [5] y liberada bajo MIT/GPL 3.0, que en función del tipo de etiqueta y el ID cambia el texto. Lamentablemente la librería no soporta todas las características necesitadas (sobre todo con la traducción de varios tipos de botones) por lo que tuvo que ser modificada.

5.8. Depuración

Tal vez uno de los puntos débiles del desarrollo con *PhoneGap/Cordova* es el sistema de depuración. Por una parte no es una aplicación nativa que se comunique directamente con el entorno de trabajo y por otra el código que queremos depurar corre en un navegador que no está en la máquina de desarrollo.

De entre las posibles soluciones existentes [6], las aplicadas en el desarrollo pasan por:

- Diseñar realizando las pruebas con un navegador Web de escritorio con motor *WebKit* (*Chrome, Safari*) similar al utilizado en los componentes de navegación web de los sistemas destinos (*Android* e *iOS*).

4 *VisualDomo.otf* es licenciada bajo *CC BY-SA 3.0*

- Probar funcionalidades básicas con el emulador *Ripple*. Este emulador da soporte sobre un navegador web de escritorio de la infraestructura *Cordova,* permitiendo además adaptarse especialmente en el apartado de visualización a diferentes dispositivos (diferentes tamaños de pantalla, resoluciones...) .

- Depurar remotamente la aplicación que corre en el dispositivo mediante *Chrome Remote Debugging* [7]. Esta utilidad permite analizar con las herramientas de desarrollador incluidas en *Chrome* el estado de la aplicación instalada remotamente: consola Javascript, *DOM*, análisis de *DIV*.

- Depuración con *Android Debug Bridge*. *ADB* permite la actuación sobre el dispositivo remoto mediante el acceso directo, ya sea por consola o mediante una aplicación gráfica. Este acceso posibilita estudiar el sistema de ficheros, así como analizar los diferentes mensajes que aparecen en la consola de errores/avisos.

5.9. Art work

Durante las primeras fases del desarrollo se realizó un trabajo de arte basado en la obtención del nombre y la creación de la imagen del producto esta última principalmente enfocada en la creación de un logotipo.

Figura 16: VisualDomo logo

Figura 17: VisualDomo logo y texto

El logotipo juega con el logotipo de *OpenDomo* cambiando el color y girando parte de su elementos interiores para hacerlos similares a las letras V y D representativas de *VisualDomo*.

Aunque en todas las versiones alfa y las primeras beta se utilizó todo este material, fue descartado en la versión final por motivos relativos al registro del mismo, por lo que se optó por utilizar el genérico de la empresa. Aun así, el material queda disponible en *GitHub* para posibles usos posteriores.

6. Control de calidad y pruebas

Para realizar el control de calidad tanto del producto como del proceso de desarrollo se han utilizado los siguientes procedimientos:

6.1. Pruebas unitarias.

Aunque existen librerías para la realización de pruebas unitarias en *Javascript*, la interoperabilidad con el entorno de desarrollo resulta compleja. Es por ello que se ha optado por una solución más sencilla, basada en utilizar la propia consola del sistema (*adb logcat*) para mostrar mensajes que indiquen la comprobación satisfactoria o no de los puntos a observar.

6.2. Pruebas de funcionalidad.

Para poder contrastar que la funcionalidad que se iba adquiriendo era la deseada, se ha ido creando una hoja de ruta que iba añadiendo las especificaciones al mismo tiempo en el que se iban implementando. Al final de cada nueva implementación se realizaba el proceso completo indicado por la hoja, para comprobar no sólo el correcto funcionamiento de cada una de ellas sino también las posibles interacciones.

Por otra parte, a partir de la primera versión alfa (0.3) se ha ido suministrando periódicamente distribuibles a *OpenDomo,* para ir comprobando la correcta implementación de cada funcionalidad de forma más real.

6.3. Pruebas de eficiencia.

Este tipo de pruebas no han podido ser realizadas de una manera correcta, aunque las simulaciones realizadas son bastantes satisfactorias.

El escenario ideal para probar este tipo de sistema consiste en la creación de un entorno local con uno o varios *ODControl,* a los que se acceda de manera habitual. Desafortunadamente, durante el periodo de realización del proyecto se ha producido un *upgrade* de los dispositivos físicos y por motivos de fabricación ha sido imposible disponer de ellos. Como escenario alternativo se ha trabajado contra un *ODControl* de acceso abierto por Internet disponible en la sede de *OpenDomo*. Aun así, teniendo en cuenta este entorno, los tiempos de respuesta obtenidos tanto en la lectura de datos como en la escritura de los mismos, resultan sumamente aceptables.

Posiblemente los peores puntos en cuanto a eficiencia se encuentren en la naturaleza en sí de la aplicación. No hay que olvidar que *VisualDomo* no es ni más ni menos que una página web corriendo sobre una navegador embebido en una aplicación para móviles. Todo ese trasiego de información genera una serie de retardos que se notan sobre todo en:

- reacción a la pulsación de eventos.
- arrastre de puertos por el plano.

El primer punto se ha intentado minimizar mostrando información del estado de la aplicación, de tal manera que el usuario tenga la menor sensación de bloqueo, mientras que el segundo se podría optimizar creando distintos *canvas* superpuestos: uno para la representación estática y otro para la dinámica (no se ha implementado por falta de tiempo).

Otro factor que entra en juego en este tipo de pruebas es la plataforma en la que corre, y más teniendo en cuenta la variedad de dispositivos existentes, no ya dentro de una plataforma en concreto, sino de varias. En este caso nos veíamos limitados a la disponibilidad de dispositivos (2) aunque como puede leerse en el apartado de pruebas de usabilidad, se aprovechó las encuestas para disponer de más datos de dispositivos distintos.

6.4. Pruebas estáticas.

Para mantener un alto grado de calidad, tanto de ejecución como de formato código fuente, se ha procesado todo el código *javascript* mediante el analizador estático *JSLint*. Este sistema proporciona (tal vez de manera un tanto estricta) una lista de posibles zonas de código que pueden resultar problemáticas, así como elementos del mismo que no siguen un formato adecuado para facilitar su legibilidad.

El código pasa el test con un 98% de acierto. El 2% restante hace referencia a zonas donde se han utilizado algún truco controlado para aligerar el código.

6.5. Pruebas de usabilidad.

A la hora de realizar pruebas de usabilidad, se ha utilizado un método de evaluación de pruebas de usuarios. Este tipo de pruebas involucran al usuario y permiten identificar problemas de usabilidad que se presentan durante el uso del sistema.

Existen varios procedimientos, casi todos ellos basados en la elaboración de un cuestionario (generalmente con las opciones *sí, no, no sabe*), que el usuario va respondiendo a medida que va interaccionando con el sistema. Desgraciadamente, este proceso requiere de mucha planificación y mucho volumen de información para que el resultado sea representativo, requerimientos que son complicados de alcanzar, aunque aun así se ha podido detectar algún error que ha podido ser modificado.

La encuesta se pasó con la primera versión beta (0.4.2) a tres personas que previamente fueron instruidas sobre la funcionalidad de los *ODControl*. A todas ellas se le solicitó la siguiente información:

Tipo de dispositivo móvil	
Sistema Operativo	
Tamaño de pantalla	
Fluidez del programa general (0-5):	
Carga	
Visualización	
Edición.	
Conocimiento de que se está haciendo en cada momento (0-5)	
Facilidad para hacer lo que se desea en cada momento (0-5)	
Control de posibles errores (el sistema impide que se realicen tareas que provocarían error)(0-5)	

Facilidad y eficiencia de uso (0-5)	
Estética (0-5)	
Documentación y ayuda (0-5)	
Impresión general:	
Mejoras sugeridas:	

Obviamente, con la muestra analizada los resultados obtenidos no poseen una gran fiabilidad, pero al menos sí permitían obtener alguna que otra conclusión. Por ejemplo, todos los usuarios se quejaron de la forma de actuar del botón de apertura del panel de plantas. En los sistemas Android se confundía con el botón de acceso a la pantalla de inicio o incluso a *Google Now*. Se optó realizar por una pequeña modificación que lo ubicaba a la izquierda.

7. Resultados y conclusiones

7.1. Resultados

Posiblemente la mejor manera de valorar el estado final del proyecto sea su contrastación con los objetivos indicados en el capítulo 2.

✔ Realizado al 100%
✗ No realizado
∿ No realizado al 100%

En cuanto a la implementación de la aplicación:

✔ Localizar o permitir configurar los posibles ODControl existentes.
Aunque el objetivo mínimo especificaba una de las dos opciones, la localización automática (realizar un *broadcast* a todo la red local) no se ha implementado.
✔ Permitir cargar planos de la localización a controlar.
✔ Ubicar en los planos los distintos dispositivos a controlar.
✔ Guardar las configuraciones para simplificar futuros accesos.
✔ Visualizar el estado de los puertos de entrada, ya sean digitales o analógicos.
✔ Modificar el estado de los puertos de salida.

En el ámbito transversal:

✔ Localización español e inglés
∿ Diseño responsable.
Se ha aplicado diseño responsable en algunas de las pantallas, pero el objeto en sí de la aplicación hace bastante complejo el aplicarlo en todas ellas. Posiblemente, más que un diseño responsable debería ser una funcionalidad “responsable”, de tal manera que en la versión para dispositivos de pantalla más pequeña ciertas funcionalidades no existieran.
✔ Se tendrá especial cuidado en conseguir alta usabilidad.
Como se comenta en el apartado de *Control de calidad y pruebas*, sería necesario un volumen de encuestas mayor para poder asegurar una correcta usabilidad.

Otros objetivos:

✔ Publicación del código en *GitHub*.
✔ Documentar el código en inglés y de tal manera que simplifique la continuidad del trabajo por parte de otros usuarios.
∿ Seguimiento del proyecto mediante el desarrollo de un blog.
Posiblemente se hayan podido hacer más post de los existentes, especialmente para tratar temas técnicos.

✗ Publicación en las principales tiendas electrónicas de aplicaciones.
Se descarta esta opción ya que la aplicación requiere de una mayor madurez para poder ser promocionada a producción.

A nivel personal, los objetivos quedan claramente satisfechos.

✔ Adquirir conocimientos sobre dispositivos domóticos.
✔ Adquirir conocimientos técnicos sobre la creación de aplicaciones multiplataforma para dispositivos móviles.
✔ Mejorar los conocimientos sobre el uso de bibliotecas javascript de apoyo al desarrollo web y de librerías de apoyo al diseño de interfaces web.
✔ Poner en práctica todos los conocimientos relacionados con los métodos de ingeniería del Software tanto para las fases de diseño como en las de implementación, control de versiones y documentación.

Evolución de los mockups

Resulta interesante echar la vista atrás y comparar lo *mockups* iniciales de cada una de las pantallas con los resultados finales. Se puede observar que aunque no son del todo fieles debido a las propias evolución que ocurren en el día a día, si que tiene siguen manteniendo la misma idea.

Pantalla de configuración (I)

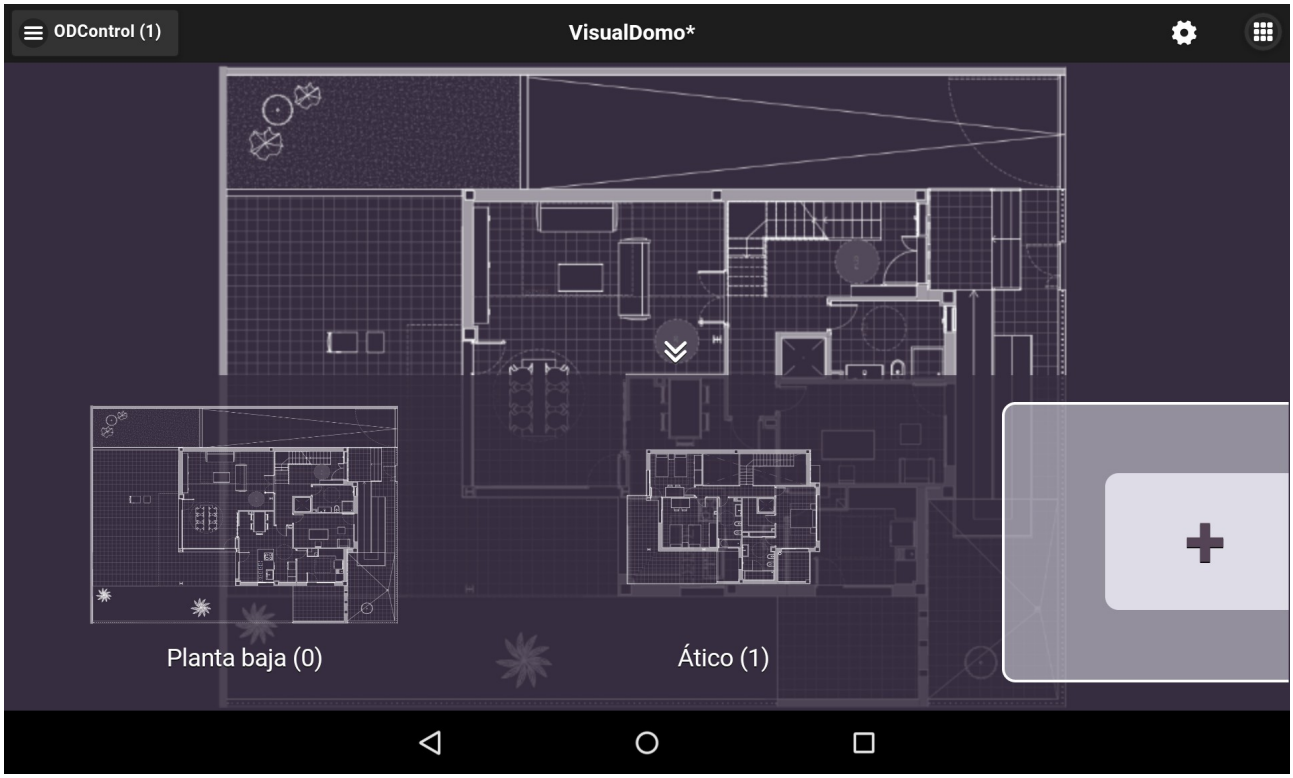


Figura 18: Pantalla de configuración (I)

Pantalla de configuración (II)

Figura 19: Pantalla de configuración (II)

Pantalla de enlace/selección

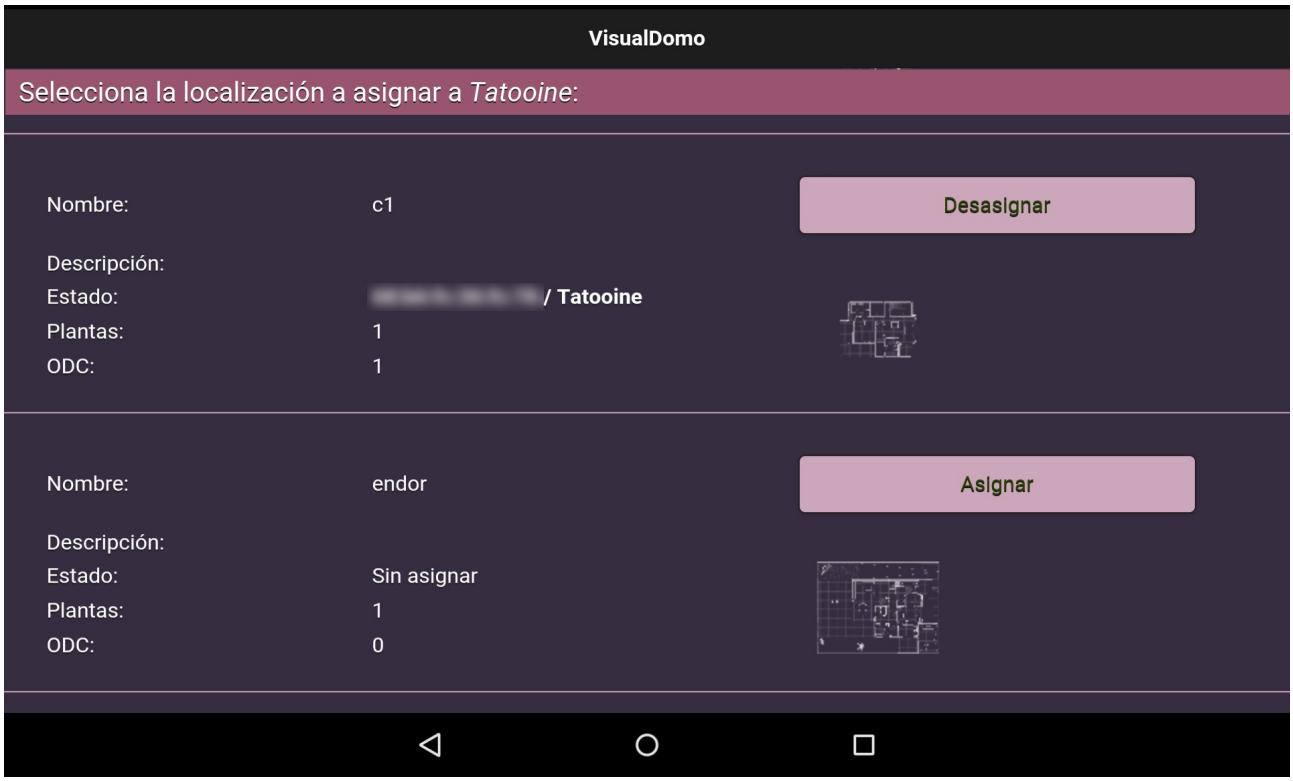


Figura 20: Pantalla de enlace/selección

Pantalla de visualización

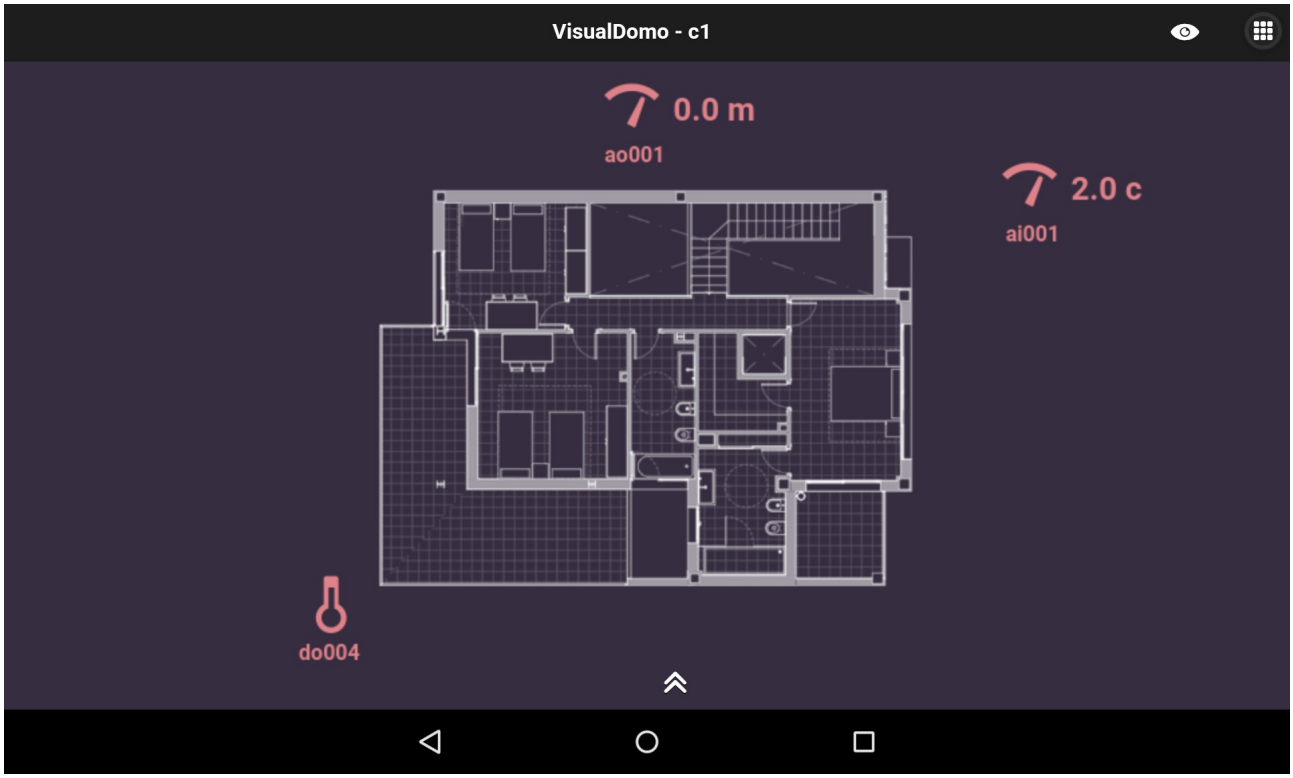


Figura 21: Pantalla de visualización

7.2. Métrica

Se ofrece a continuación una tabla con información obtenida sobre el código generado:

Concepto	**Valor**
Ficheros de código (javascript) / Líneas	16 / 3586
Ficheros de código (CSS) / Líneas	4 / 753
Ficheros de código (html) / Líneas	1 / 423
Ficheros de código (xml) / Líneas	1 / 15
Ficheros de código WifiInformation (Java) / Líneas	1 / 83
Ficheros documentación / Líneas	7 / 298
Ficheros totales / Líneas	28 / 5158
Tamaño código Javascript	136.167 bytes
Tamaño código CSS	15.258 bytes
Tamaño código HTML	26.610 bytes
Tamaño código XML	574 bytes

Tamaño código Java	2.607 bytes
Tamaño documentación	17.847 bytes
Tamaño total	199.063
Número de funciones Javascript	86

7.3. Dedicación

Se ha realizado una medición de las horas dedicadas al desarrollo del proyecto a lo largo de las fases del mismo[5]. Como se puede observar la dedicación ha sido irregular, aunque con bastante trabajo en los meses de verano, donde se crearon las líneas generales de la implementación y en los meses finales.

Fase	Dedicación (h)
Fase 1. Estudio preliminar.	55
Fase2. Diseño de la aplicación.	15
Fase 3. Implementación de la aplicación.	313
Fase 4. Pruebas en fase beta.	18
Fase 5. Publicación.	25
Totales	426

Si nos centramos únicamente en la implementación, un seguimiento más concreto puede hacerse observando las gráficas de uso que *GitHub* proporciona. En ellas se puede observar el periodo de codificación con los pico en época de máxima disponibilidad[6].

7.4. Valoración personal

Conocimientos aplicados

En el desarrollo del proyecto se han aplicado varios de los conocimientos adquiridos en el Máster, especialmente las de aquellas asignaturas de áreas relacionadas con la programación. Entre ellas:

Desarrollo de aplicaciones WEB

Es obvio, que aunque la aplicación está dirigida a dispositivos móviles, las tecnologías aplicadas son de entorno web, especialmente en el lado del cliente. Cabe destacar las enseñanzas relativas a *HTML5* en general y a su componente *canvas* en particular, sin olvidar el uso de CSS.

5 No se computa la fase seis ya que es una fase que debe empezar a la finalización del proyecto.

6 Sólo se muestran los gráficos referentes a VisualDomo. Los datos referente a *WifiInformation* puede consultarse desde *https://github.com/aoltra/WifiInformation/graphs/contributors*

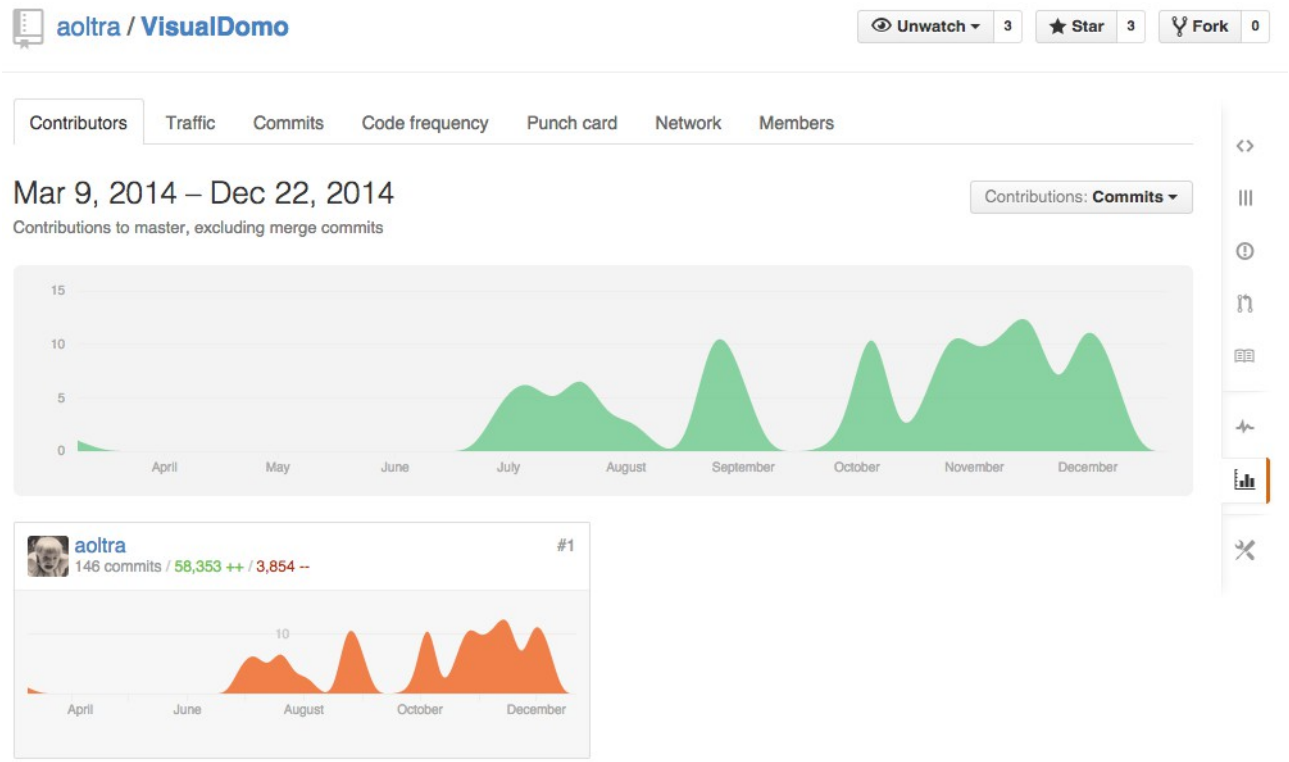


Figura 22: Gráfico de contribuciones en GitHub

Ingeniería de Software en entornos de Software Libre

Posiblemente la asignatura que más me ha hecho cambiar la manera de trabajar. De ella se han aplicado todos los conocimientos relativos al diseño de software, utilización de diagramas UML, metodologías, realización de pruebas y control de versiones, en este caso aplicado a *Git*.

Desarrollo de Software

La asignatura está centrada en el desarrollo de software para plataforma (escritorio, dispositivos móviles..), pero explica conceptos como el manejo de librerías o creación de ficheros de compilación aplicados en el TFM.

Implantación de sistemas de Software Libre

Aunque la asignatura puede parecer alejada a la temática del proyecto, se han aplicado ciertas técnicas como el estudio de la vida del proyecto, su planificación temporal, la valoración de los recurso, etc. Además se han utilizado técnicas que, aunque estudiadas también en otras asignaturas, fueron puestas en práctica en ésta, como metodologías *SCRUM*.

Introducción al Software Libre

Tal vez es la asignatura base del *Máster*. Gracias a ella se pueden entender bien cada una de las posibles licencias libres bajo las que se puede distribuir un software y las compatibilidades entre ellas, conocimiento fundamental para poder saber qué software y qué librerías podían utilizarse para la creación de *VisualDomo*.

Conocimientos extras alcanzados

También se han adquirido conocimientos no directamente tratados en el Máster, pero de aplicación en el TFM, especialmente en lo referente a tecnologías más específicas.

- Conocimientos sobre dispositivos de control domótico.
- Conocimientos sobre framework *Cordova*.
- Utilización de librerías *Javascript* como *JQuery* y *JQuery Mobile*.
- Creación de documentación en formato *markdown.*
- Creación sencilla de fuentes tipográficas*.*
- Conocimientos sobre la filosofía *Responsive Design.*

Valoración

A la hora de valorar la experiencia tengo un sabor agridulce. Por una parte, académicamente me encuentro muy satisfecho. Los conocimientos adquiridos, especialmente en el campo de la programación web han sido tremendos y ni que decir tiene del uso de un *framework* como *Cordova.*

Pero, sin embargo, veo que el uso de un aplicación *HTML5* sobre móviles aún no está del todo afinada. Creo que es una tecnología que va madurando a pasos agigantados y que ya tiene un nivel de madurez alto para la creación de cierto tipo de aplicaciones sencillas, pero para aquellas que requieren ciertos componentes pesados, aún dependen mucho del *hardware* de la máquina*.* Además, me resulta antinatural el desarrollo de aplicaciones plataforma de manera asíncrona (posiblemente porque provengo de ese mundo). La gestión asíncrona para una aplicación meramente local, resulta confusa tanto de entender como de implementar.

Por último, mi mayor enemigo: el tiempo. Quedan ideas en el tintero, ideas que considero interesantes y que el tiempo no me ha dejado implementar y que probablemente me harían replantear ciertas decisiones tomadas.

Todo esto me lleva a hacerme una pregunta ¿qué haría de otra manera?. Posiblemente desarrollaría en código nativo (J*ava, Objective-C, C#* …), pero si la especificación de uso de *Cordova* se mantuviera, me centraría en una reestructuración del código, posiblemente con una estructura mucho más orientada a objetos y la aplicación de algún *framework* MVC como *Angular.js*

7.5. Futuro

Es evidente que aunque el proyecto es completamente funcional, tanto el carácter académico como el tiempo de implementación y las horas de pruebas a las que ha sido testeado, hacen que el estado del proyecto únicamente pueda considerarse como un prototipo. Y, como todo prototipo, en el propio proceso de desarrollo surgen nuevas ideas que podrían ser implementadas en posteriores versiones.

Las posibilidades, entre otras, son:

- Aprovechar la tecnología para exportarlo a otras plataformas móviles (principalmente iOS).
- Crear una extensión para navegadores de escritorio.
- Acceso remoto a localizaciones externas a la red local (mediante conexión 3G/4G).
- Posibilidad de hacer copias de seguridad de las localizaciones.
- Aplicación de diferentes estilos.
- Soporte de formatos gráficos vectoriales para facilitar en muchos casos, la carga desde formatos más habituales en este tipo de representación y para posibilitar un implementación más correcta de herramientas como zoom.
- Dotarlo de accesibilidad para permitir el manejo a personas con deficiencias visuales o de otro tipo.
- Controlar simultáneamente varias localizaciones.
- Implementación de avisos.
- Modo multivista en tabletas que permitan visualizar varias plantas de manera simultánea.

8. Vocabulario

Actores: personas o entidades externas a la aplicación que interactúan con ella, realizando un intercambio de información.

CASE (Herramienta CASE): cada uno de los instrumentos o herramientas software de apoyo al desarrollo.

IDE: aplicación destinada a la creación de aplicaciones que integra varias herramientas CASE, especialmente las destinadas a la fase de codificación.

Mockup: prototipo, generalmente a escala real, utilizado para la demostración, evaluación del diseño, etc.

Usabilidad: grado de eficacia, eficiencia y satisfacción con la que usuarios específicos pueden lograr objetivos específicos, en contextos de uso específicos.

Responsive Design (RWD): filosofía de diseño que hacen que el creador adapte su desarrollo a cada una de los posibles medios en los que puede ser ejecutado/visualizado. Su principal escenario es el desarrollo de aplicaciones Web.

SCRUM: Estrategia de desarrollo incremental que se centra principalmente en basar la calidad del resultado, más en la calidad que determinan las personas conocedoras del proyecto que en la calidad de los procesos empleados, y en solapar las diferentes fases del desarrollo.

MVC: patrón de diseño que separa de manera clara los datos, la capa de negocio (que gestiona esos datos) y la capa de interfaz gráfica.

9. Anexos

9.1. Anexo I. Manual de usuario

Introducción

El objetivo de *VisualDomo* es la creación de lo que denominamos *localizaciones* y de su posterior visualización/interacción de manera gráfica.

Una *localización* es una ubicación espacial de una o más plantas en la que existen uno o varios *ODControl* conectados que disponen, para al menos uno de sus puertos, de elementos distribuidos a lo largo de los diferentes espacios. Sobre estos puertos se permite interaccionar, ya sea para obtener información de ellos o para modificarlos. Una vivienda particular, una vivienda estacional, las oficinas de una empresa o un colegio pueden ser considerados ejemplos de posibles localizaciones.

Modos de funcionamiento

VisualDomo posee dos modos de funcionamiento:

- *Modo edición*. Permite la creación y edición de las localizaciones. Añadir plantas, configurar ODControls, ubicar puertos en cada planta, definir el tipo de información de cada puerto...

- *Modo visualización*. Es el modo de trabajo habitual. Permite visualizar el estado de los puertos ubicados en las plantas, así como (si el puerto es de salida) modificar su valor.

Estados de VisualDomo. Operatividad

En función de la conectividad del dispositivo donde está funcionando, *VisualDomo* puede encontrarse en tres posibles estados:

- *Sin conexión a red.* El dispositivo móvil no tiene conexión a ningún tipo de red. En este estado únicamente es posible acceder a la creación de nuevas localizaciones o a la configuración de las ya existentes. En este último caso no será posible añadir nuevos *ODControl*, pero si ubicar puertos (o reubicar los ya colocados) y/o cambiar su configuración.

- *Conexión a red 3G/4G*. El dispositivo únicamente se encuentra conectado a una red de datos. En este caso además de las opciones anteriores tendrá acceso a comunicarse con localizaciones remotas (opción no soportada en esta versión).

- *Conexión Wifi*. El dispositivo se encuentra conectado a un red Wifi. En este caso todas las opciones se encuentran disponibles de partida, aunque la opción de visualización sólo lo estará en el caso de que alguna de las localizaciones esté asignada a la red Wifi a la que se encuentra conectado el dispositivo.

Estado de una localización

Teniendo en cuenta lo anterior, una *localización* puede encontrarse en tres posibles estados:

- *Desasignada.* La *localización* está creada pero no está enlazada a ninguna red.
- *Asignada.* La *localización* está creada y configurada, estando enlazada a una red wifi determinada.
- *Conectada.* La *localización* está asignada a la red wifi a la que el dispositivo móvil se encuentra conectado. En esta situación es posible acceder al modo de visualización.

Creación de localizaciones

La creación de localizaciones permite la asignación de plantas, la asignación de *ODControls*, la ubicación de los puertos en las diferentes plantas y la personalización de esos puertos.

Asignación de plantas

La gestión de las plantas se realiza en su totalidad desde el panel inferior de la ventana de creación. Este panel puede obtenerse mediante la pulsación del símbolo ^ ubicado en la parte inferior de la ventana.

- Añadir planta

  Cada una de las plantas se apoya en una imagen que representa el plano de esa planta. Para añadirla es necesario pulsar sobre el + del último ítem del panel. Este botón abre una ventana que permite la configuración de la planta.

  Nivel. Nivel de la planta comenzando en 0. En caso de que no se defina, se asignará el nivel de la última planta definida más 1.

  *Nombre. N*ombre de la planta. Por ejemplo: sótano, planta baja, altillo...

  Descripción. Descripción de la planta.

  Invertir colores. Para una mejor visualización, invierte automáticamente los colores de la imagen.

  Selecciona imagen. Permite al carga de una de las imágenes almacenadas en la librería del dispositivo.

  > Se aceptan varios formatos a la hora de seleccionar la imagen de la planta: jpg, bmp, gif, png... Para una mejor visualización es recomendable el uso de un formato con transparencias como PNG.

- Edición de plantas.

  La edición de plantas se realiza desde la barra de herramientas de plantas que se puede obtener pulsando durante cierto tiempo sobre la planta en el panel inferior. Esta barra permite:

  Borrar la planta

  Configurar planta. Muestra el cuadro de diálogo de configuración (ver apartado

Añadir planta)

Cambiar de nivel. Desplazando la planta a la derecha (bajar nivel) o izquierda (subir nivel).

- Selección de planta.

  Para ubicar la planta en el panel central únicamente es necesario pulsar sobre ella.

Asignación de ODControl

La gestión de los *ODControl* se realiza desde el panel izquierdo de la ventana de creación. Este panel puede obtenerse mediante la pulsación del botón *ODControl (n)* situado a la izquierda de la cabecera. En el botón, *(n)* indica el número de *ODControl* añadidos a la localización.

- Añadir *ODControl*.

  Para añadir un *ODControl* hay pulsar sobre el botón *Añadir ODC* situado en el panel izquierdo. Este botón abre una ventana que permite la configuración del *ODControl*.

  IP. Dirección IP del ODControl

  Usuario. Usuario con permisos de acceso al *ODControl*

  Contraseña. Contraseña del usuario

  Nombre. Nombre con el que será identificado el *ODControl*

  Una vez localizado, aparecerá un ítem en el panel izquierdo con el nombre del *ODControl* que dará acceso a un panel desplegable con la lista de todos los puertos disponibles.

  > Para la carga de los *ODControl* es necesario que *VisualDomo* esté en el estado *Conexión Wifi*.

  > El añadido de los *ODControl* no implica la asignación de la localización a la red Wifi de la que se está tomando datos.

- Edición del *ODControl*

  La edición de los *ODControl* se realiza desde la barra de herramientas de *ODControl,* que se puede obtener pulsando sobre el texto del nombre de *ODControl* en el panel izquierdo. Esta barra permite:

  Borrar el ODControl.

  Configurar el ODControl. Presenta el cuadro de diálogo de configuración (ver *Añadir ODControl*) pero únicamente da acceso a cambiar el nombre.

  Recargar ODControl. Permite la actualización de los puertos, añadiendo nuevos y eliminado los deshabilitados (opción no soportada en esta versión).

Ubicación de puertos

La ubicación de puertos se realiza desde el panel izquierdo. En su interior se encuentra la lista de *ODControls* configurados. Cada uno de estos *ODControl* permite abrir un panel desplegable con una lista de todos aquellos puertos que el *ODControl* permite visualizar.

> Si desea que algún puerto desaparezca o aparezca en el listado debe realizar la correspondiente configuración desde el interfaz gráfico de configuración del *ODControl*.

Cada uno de los ítems de la lista define un puerto mediante el formato:

/ (*analógico/digital*) / (*entrada/salida*) nombre (*ubicado/no ubicado*) / / / / *(función)*

- Ubicación de puertos

  Para ubicar un puerto en la planta actual, únicamente hay que pulsar sobre el nombre del puerto. Esta acción colocará el puerto en el centro de la planta. En ese instante el icono aparecerá en la entrada del puerto en la lista del panel izquierdo.

  > la representación de un puerto es un icono especificando su funcionalidad. Ver apartado *Configuración de puertos*.

  Para ubicar el puerto en la posición deseada simplemente hay que pulsar sobre él y desplazarlo suavemente sobre la imagen de la planta.

- Eliminación de puertos.

  Para eliminar un puerto de su ubicación en planta simplemente pulse sobre el icono desde la lista de puertos en el panel izquierdo.

- Configuración de puertos

  La configuración de puertos permite la definición de la funcionalidad del puerto, así como del texto que acompaña a la representación visual. Se realiza desde un cuadro de diálogo al que se accede pulsando sobre el icono de funcionalidad (último icono) del puerto en la lista de puertos. Si no se ha definido ninguna funcionalidad, se muestra el icono por defecto en función del tipo de puerto (analógico o digital).

  > No es necesario que el puerto esté ubicado para proceder a su configuración.

  El cuadro de diálogo consta de:

  Unidades. En el caso de puerto analógico permite definir un texto que se situará detrás del valor del puerto a modo de unidades.

  Factor. Factor por el que se multiplica el valor del puerto a la hora de representarlo y/o modificar su valor (ver *Visualización*)

  Funcionalidad. Identifica mediante un icono la posible funcionalidad del puerto. En esta versión se encuentran disponibles iconos para representar *Luces, Persianas, Temperatura y Audio.*

Configuración de la localización

La configuración de la *localización* permite la definición del nombre y la descripción de la misma. Se puede acceder a ella desde el menú situado en la parte derecha de la cabecera.

El cuadro de diálogo de configuración consta de:

Nombre.Nombre de la localización. Este nombre no puede ser vacío. Además, no permite la inclusión de espacios.

Descripción. Descripción de la localización.

Grabación de la localización

La grabación de la *localización* se realiza desde el menú situado en la parte derecha de la cabecera.

> En caso de que el usuario no haya definido un nombre de la localización a la hora de grabar, *VisualDomo* presentará automáticamente el cuadro de diálogo de configuración de la localización.

Las localizaciones se almacenan en el directorio *VisualDomo/locations*. Para cada *localización* se crea un fichero con extensión *.vdlt* o *.vdl* (en función de si la *localización* está asignada o no) y una carpeta que contiene una copia de las imágenes de las plantas.

Configuración.

La opción de *Configuración* permite carga una de las localizaciones guardadas y trabajar con ella de la misma manera que se puede realizar desde la opción *Nueva localización*.

Enlazar

La opción *Enlazar* del menú principal permite asignar a una *localización* la red wifi actual o desasignar una localización de la red wifi que tenga definida.

En caso de ser asignada *VisualDomo* pasa automáticamente al modo *Visualización* .

> La red wifi se identifica mediante su BSSID.

Externa

Permite la conexión a localizaciones asignadas a redes diferentes a la red actual. Esta opción no esta soportada en esta versión.

Visualización

La visualización es el modo habitual de trabajo de *VisualDomo*. Se puede acceder a él mediante la opción *Visualización* del menú principal, al enlazar una localización o directamente cuando el sistema detecta que existe una localización para la wifi actual.

La ventana de visualización es muy similar a la ventana de creación de localizaciones, pero sin el panel izquierdo de gestión de los *ODControl*.

La ventana principal muestra cada una de las plantas con los puertos ubicados y su valor actual. En caso de que el puerto sea de salida será posible modificar su valor (en caso de que sea analógico) o su estado (en caso de que sea digital). Para ello, en ambos casos bastará con pulsar

sobre el puerto para cambiar su estado o mostrar la ventana de configuración de los valores, desde donde podemos modificar el valor.

> ✎ Utilice valores dentro del rango admisible. Tenga en cuenta que estos valores se encuentran afectados del factor utilizado en la configuración del puerto.

Configuración de la aplicación

Desde el menú situado a la derecha de la cabecera es posible acceder a la configuración de general de *VisualDomo*. Dicha configuración permite:

OpenDomoOS. URL (sin protocolo) de acceso a *OpenDomoOS.*

Tiempo entre actualizaciones de estado. Define (en segundos) el tiempo entre conexión a los *ODControl* para actualizar en pantalla el estado de los puertos.

Idioma. Lenguaje en el que aparecerán los textos de la aplicación.

9.2. Anexo II. Licencia

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used

to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.
All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.
No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.
You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and

give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.
You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.
A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.
You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the

Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.
A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require

their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.
All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

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, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you

must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public 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.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have

the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

10. Referencias y bibliografía

1: Palenzuela, Oriol; Lerch, Daniel, Comandos ODControl, 2014, http://www.opendomo.com/wiki/index.php/Comandos_de_ODControl2

2: Lerch, Daniel, ODLEP, 2013, http://www.opendomo.com/wiki/index.php/ODLEP

3: Ashfaq Hossain, Almacenamiento HTML5, 2014, https://developer.mozilla.org/es/docs/DOM/Almacenamiento

4: Daniel Bruce , entypo font, , http://www.entypo.com/

5: Aurelio De Rosa, JQuery Translation, 2013, https://github.com/AurelioDeRosa/Currency-Converter/blob/e656a1aa44253db8eb8c39bd201547bc5d16f659/js/jquery.auderoTextChanger.min.js

6: JOK101, Debugging in PhoneGap, 2014, https://github.com/phonegap/phonegap/wiki/Debugging-in-PhoneGap

7: Google, Chrome Remote debugging, 2014, https://developer.chrome.com/devtools/docs/remote-debugging#debugging-webviews

8: Richard York, Beginning JavaScript® and CSS Development with jQuery, 2009, Wiley Publishing, Inc.,ISBN: 978-0-470-22779-4

9: Raymond Camden, Andy Matthews, JQuery Mobile Web Development Essentials, 2012, Packt Publishing Ltd., ISBN 978-1-84951-726-3

10: Jack Franklin, Beginning JQuery, Apress, 2013, Apress, ISBN-13 (pbk): 978-1-4302-4932-0

11: Rohit Ghatol, Yogesh Patel, Beginning PhoneGap, 2012, Apress, ISBN-13 (pbk): 978-1-4302-3903-1

12: Thomas Myer, Beginning PhoneGap, 2012, John Wiley & Sons, Inc., ISBN: 978-1-118-15665-0