PARTE II

ANEXOS

ANEXO A - Manual de usuario de la aplicación Android *MyCity4Me*

En este anexo se recoge el manual de instrucciones de la aplicación Android *MyCity4Me* desarrollada como parte del sistema implementado en este PFC.

A.1 ¿Qué es *MyCity4Me*?

MyCity4Me es una aplicación para dispositivos móviles Android que sirve como herramienta de difusión de las ofertas publicadas por pequeños comercios de Zaragoza, en un sistema implementado sobre en una infraestructura de datos en la nube.

MyCity4Me muestra en tiempo real las ofertas activas que hay en Zaragoza en función de la posición del usuario del dispositivo móvil. Dicha representación de ofertas se lleva a cabo de dos formas, a) mediante una lista donde aparecen las ofertas en función del perfil del usuario y de su posición geográfica y b) mediante un mapa en el que aparecen todos los comercios con ofertas activas.

MyCity4Me permite acceso mediante el registro del usuario en el sistema, junto con un perfil en el que se solicitan tres datos: género, fecha de nacimiento y si es usuario de la *Tarjeta Ciudadana* o no. Si el usuario dispone de la tarjeta ciudadana, este tendrá acceso a ofertas mejores.

Si al usuario no le interesa registrarse, esta aplicación también cuenta con un modo *Invitado*, con acceso a menos ofertas.

A continuación se describe el funcionamiento de la aplicación y cómo usarla, con el apoyo de capturas de pantalla de la aplicación en uso.

A.2 Inicio de la aplicación y acceso a ella.

Cuando el usuario inicia la aplicación, la ventana que aparece es la de la Figura A.1.

Figura A.1. Pantalla de inicio, selección de modo de acceso.

En esta pantalla se pregunta cómo quiere el usuario acceder a la aplicación, es decir si quiere acceder como invitado o si prefiere crear un perfil personal. Si pulsa sobre *Entrar como invitado*, la aplicación accede directamente a la ventana principal del modo invitado (Figura A.2), asignándole un identificador de usuario por defecto. Si por el contrario decide pulsar *Entrar como usuario registrado*, la aplicación muestra la ventana de *Login* (Figura A.4)

Figura A.2. Ventana principal en modo invitado.

Si el invitado decidiese crear una cuenta de usuario personal, este debe hacer *click* en *crear cuenta*, y entonces accede a la ventana de creación de usuarios, representada en la Figura A.3

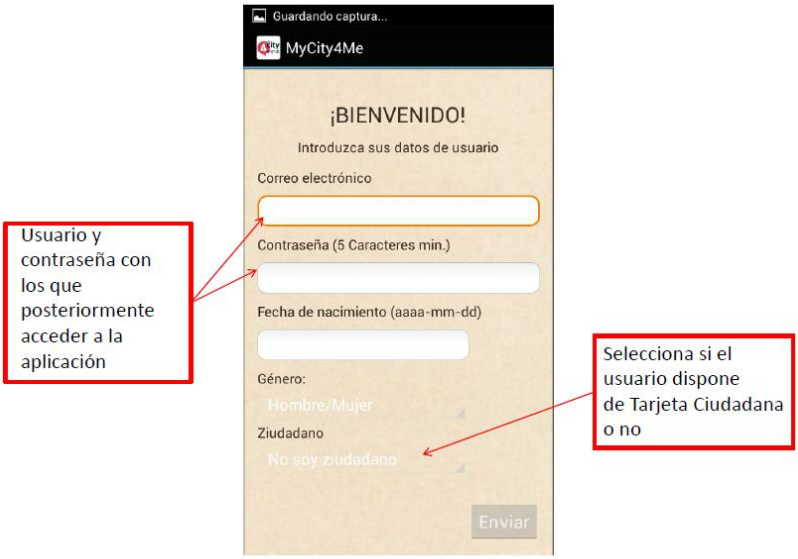

Figura A.3. Ventana de registro.

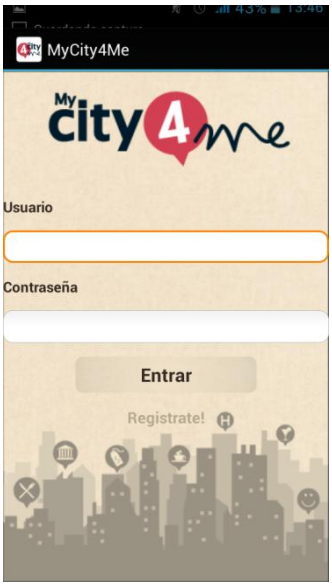

Figura A.4. Ventana de login.

Si el usuario ya se ha registrado antes, para acceder basta con introducir sus datos de acceso para acceder a la ventana principal de la aplicación, Figura A.5. Si por el contrario aún no se ha registrado, debe hacer *click* sobre *regístrate!*, y lanza la ventana de registro donde puede proceder a la creación de su usuario personal, tal y como hemos visto en la Figura A.3.

Figura A.5. Ventana principal en modo usuario registrado.

Una vez dentro de la aplicación, a excepción de la ventana principal, el resto de ventanas son comunes a ambos modos de acceso y se describen en la siguiente sección.

Mientras el usuario no cierre la sesión, o en el caso de los invitados, no se creen una cuenta propia, la siguientes veces que accedan a la aplicación, la sesión se iniciará directamente, accediendo de esta manera a la ventana principal.

A.3 Consulta y solicitud de las ofertas activas en la ciudad

Una vez se ha accedido a la aplicación, ya sea como invitado o como usuario registrado, se puede acceder a las ventanas que muestran las ofertas de la ciudad.

Partiendo de la ventana principal mostrada en la Figura A.5, el usuario puede realizar tres funciones: Ver su perfil, acceder al menú de ofertas o acceder al mapa de ofertas.

En primer lugar la ventana del perfil muestra, como su nombre indica, el perfil del usuario. Además incluye tres datos: el número de solicitudes realizadas, el número de ofertas compradas y un porcentaje que compara los dos datos anteriores. Un ejemplo de esta ventana se puede ver en la Figura A.6.

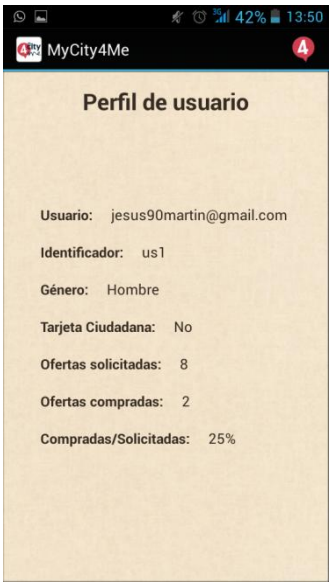

Figura A.6. Perfil de usuario.

Desde esta ventana también es posible acceder directamente al mapa de ofertas, gracias al botón incorporado en la barra de acciones.

En segundo lugar, si se pulsa sobre *Menú de ofertas*, se accede a la ventana representada en la Figura A.7. Esta ventana permite filtrar las ofertas según el tipo de comercio que las ha publicado, o si el usuario lo prefiere puede acceder a todas las ofertas, sin importar su origen, pulsando el botón *Todas las ofertas*.

Figura A.7. Filtrado de ofertas en función del tipo de comercio.

Sea cual sea la opción seleccionada se accede a una ventana con el aspecto de la Figura A.8, en la que aparece una lista con las ofertas en función del tipo de comercio elegido,

el perfil del usuario, y la posición geográfica de este. Además la lista esta ordenada según la hora de finalización de la oferta, dando prioridad a las ofertas a las que menos tiempo de vida les queda.

Figura A.8. Aspecto de la lista de ofertas después de pulsar el botón Todas las ofertas.

En esta ventana, el usuario es capaz de ver información acerca de ofertas, así como llamar directamente al comercio o enviar una solicitud de oferta. Tras el envío de la solicitud aparecerá un mensaje diciendo que su solicitud ha sido enviada correctamente y mostrando el identificador de usuario, el cual debe usar posteriormente cuando vaya a adquirir la oferta en el comercio (disponible también en el perfil de usuario). En el caso de haber solicitado ya esa oferta y no haberla comprado, si se vuelve a intentar solicitar, la aplicación devuelve un mensaje diciendo que el límite de solicitudes ha sido alcanzado.

Por último, después de seleccionar una de las ofertas, se puede obtener una ruta desde la posición actual del usuario hasta el comercio que ha publicado dicha oferta, lo cual facilita al usuario una forma de saber cómo llegar hasta el comercio ofertante.

En tercer y último lugar, si se pulsa sobre *Mapa de ofertas*, se accede a un mapa que se encarga de mostrar todos los comercios que tienen ofertas activas. Su aspecto es el representado en la Figura A.9.

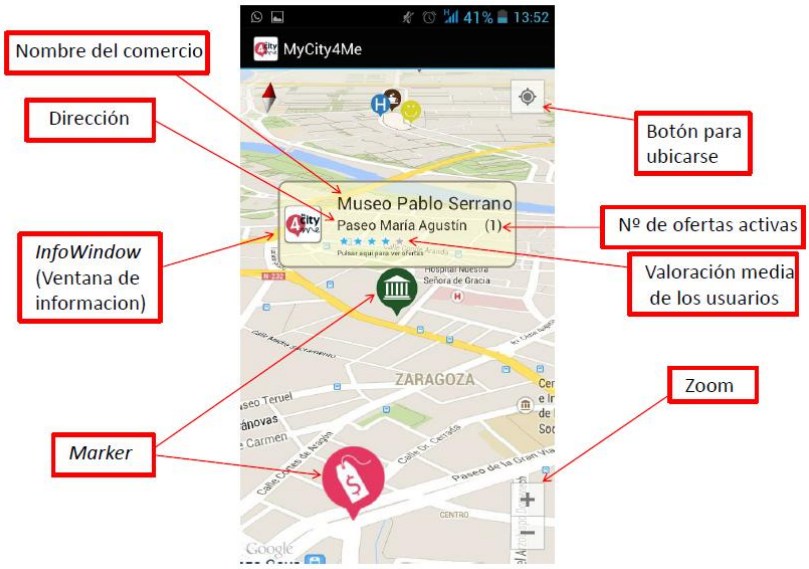

Figura A.9. Mapa de ofertas.

Desde este mapa el usuario puede visualizar rápidamente cuáles son los comercios que tienen ofertas publicadas en ese momento. Si se pulsa sobre un *marker*, aparece una ventana de información que muestra el nombre del comercio, su dirección, el número de ofertas activas y una valoración media que los usuarios han realizado sobre el comercio. A su vez, esta ventana de información también se puede pulsar, de manera que al pulsarla se accede a una lista de ofertas, como la de la Figura A.10, que se asemeja a la del menú de ofertas, sólo que ahora las ofertas que aparecen son únicamente las del comercio seleccionado. Las funciones disponibles en esta ventana son las mismas que la de la Figura A.8, a excepción que desde esta ventana se puede enviar una valoración del comercio.

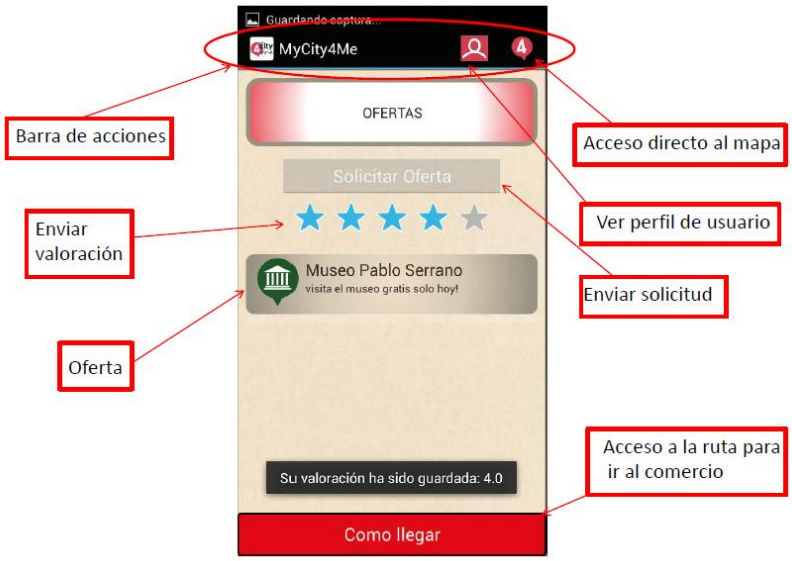

Figura A.10. Lista de ofertas del comercio seleccionado en el mapa.

ANEXO B - Manual de usuario de la aplicación web

En este anexo se recogen los manuales de instrucciones de la aplicación web modo *administrador* y modo *comercio* del sistema desarrollado en este PFC.

B.1 Modo *administrador*

La aplicación para el administrador consiste en una web que permite administrar el sistema de una manera sencilla. El administrador se encarga de crear las credenciales de acceso de los comercios, y éstos deben facilitar sus datos de nombre, dirección, teléfono, etc. Posteriormente el comercio tendrá la posibilidad de modificar sus datos desde la aplicación en modo *comercio*. También se pueden crear usuarios de la aplicación Android, aunque ellos mismos puedan realizar dicha labor desde la propia aplicación móvil. Además el administrador puede ver un informe con todos los datos de los comercios y los usuarios del sistema, pudiendo borrarlos o modificarlos.

A continuación se describe cada función paso a paso con la ayuda de imágenes de la aplicación web.

La pantalla de acceso al sistema tiene el aspecto de la Figura B.1. Una vez introducidos los datos de acceso del administrador se accede a la ventana de inicio de la aplicación, desde la cual se puede navegar al resto de ventanas pulsando sobre los botones que así lo indican, tal y como se aprecia en la Figura B.2

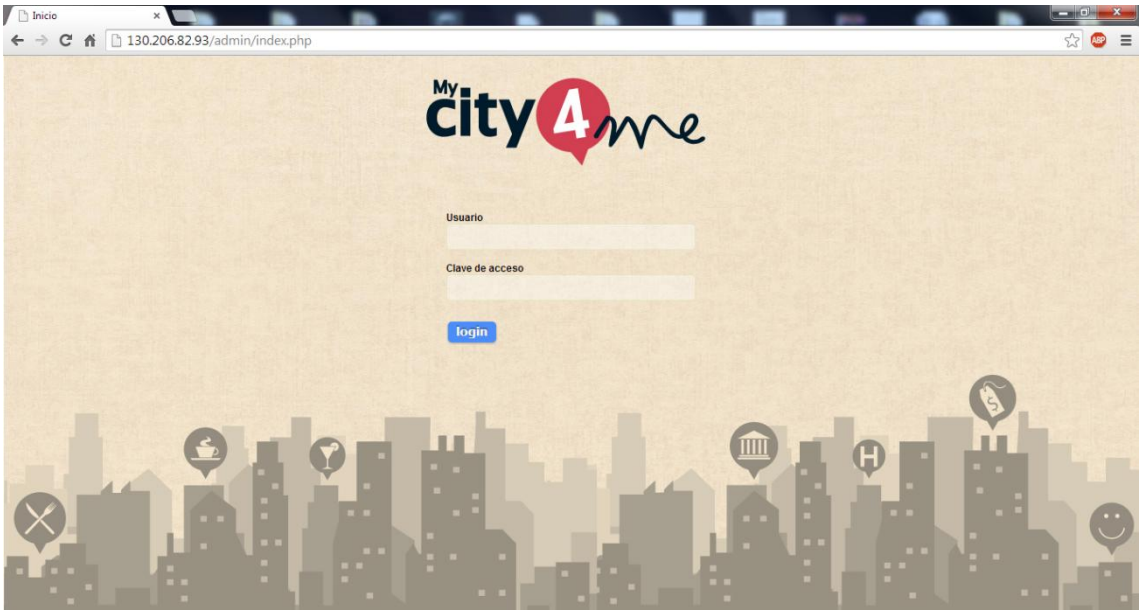

Figura B.1. Ventana de acceso al sistema del administrador.

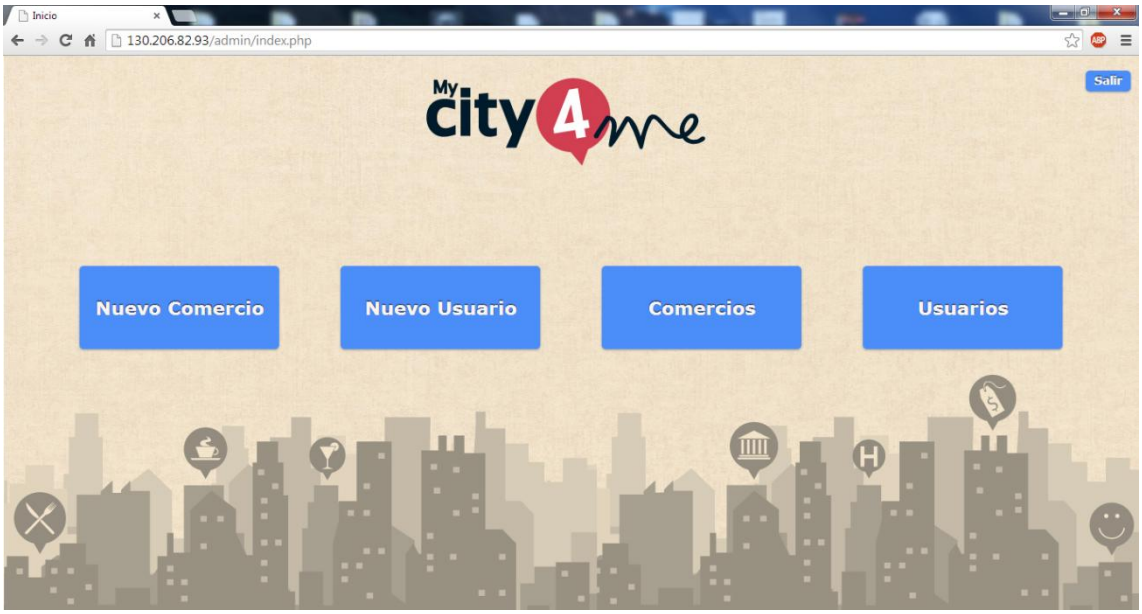

Figura B.2. Pantalla de inicio.

B.1.1 Nuevo Comercio

Pulsando el botón *Nuevo Comercio* se accede al formulario que permite introducir los datos del nuevo comercio a registrar. Esta ventana tiene el aspecto mostrado en la Figura B.3

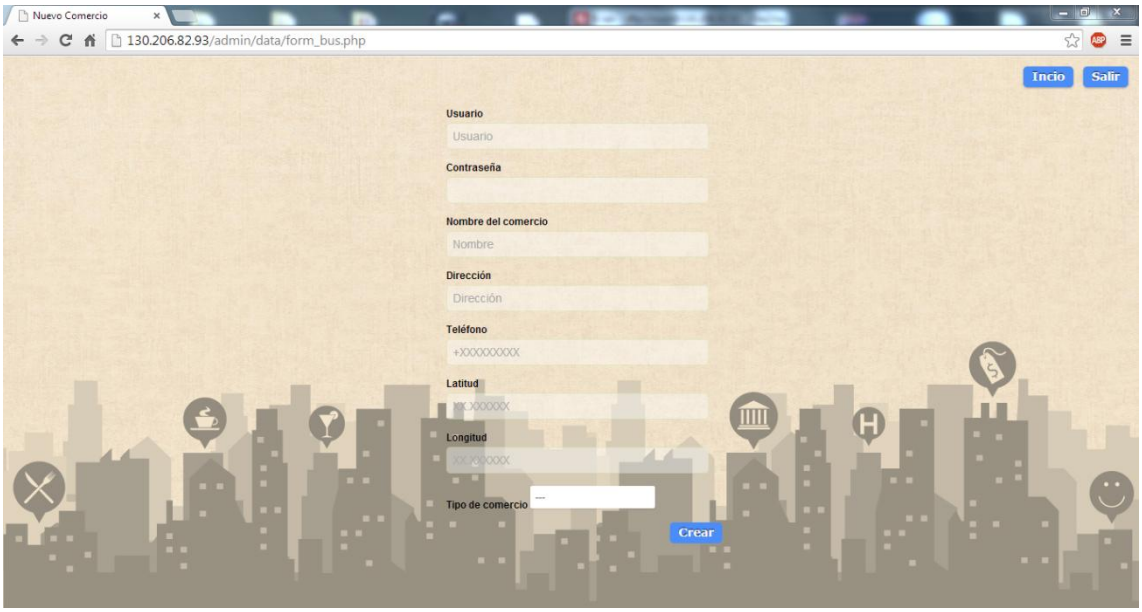

Figura B.3. Ventana donde se registran nuevos comercios.

B.1.2 Nuevo Usuario

Pulsando el botón *Nuevo Usuario*, se accede a un formulario que permite registrar los datos del perfil de un nuevo usuario, tal y como aparece en la Figura B.4

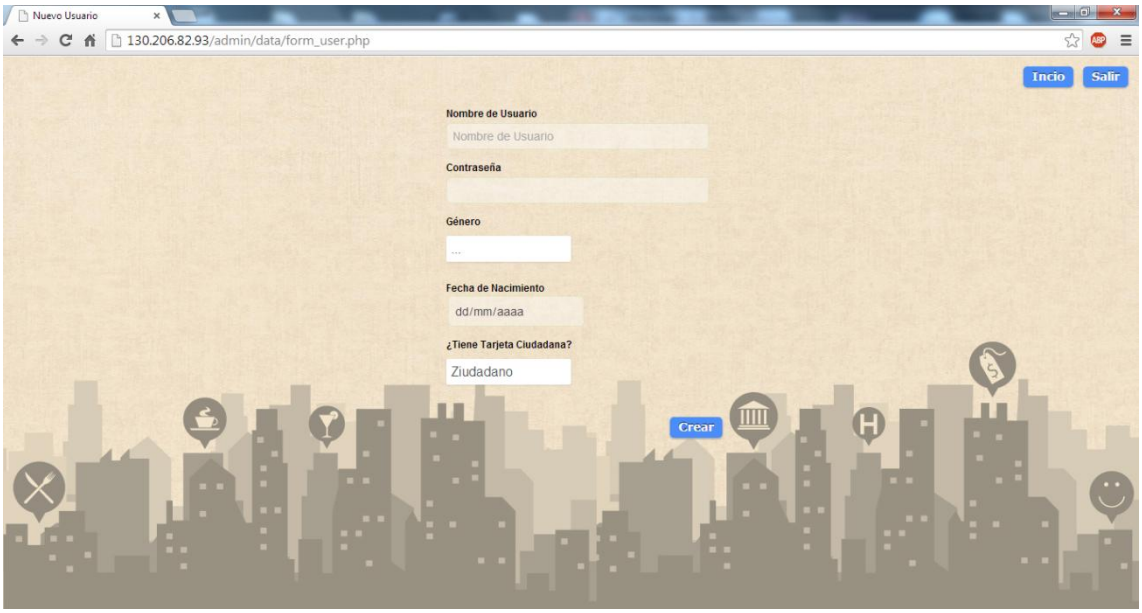

Figura B.4. Ventana donde se registran nuevos usuarios.

B.1.3 Informe de comercios

Pulsando el botón *Comercios* se accede a la ventana representada en la Figura B.5, en la cual se puede observar una tabla donde figuran todos los datos de los comercios registrados en el sistema. Debajo de esta tabla hay dos botones que introduciendo el identificador del comercio, permiten borrarlo o modificar sus datos. En el caso de pulsar *Borrar comercio*, la aplicación pregunta si realmente quieres eliminar dicho comercio antes de realizar ninguna acción. En el caso de pulsar *Modificar comercio*, se accede a una nueva ventana con un formulario en el que se pueden apreciar los datos actuales del comercio, pudiendo modificarlos y guardarlos. Esta ventana se muestra en la Figura B.6.

Figura B.5. Informe de comercios.

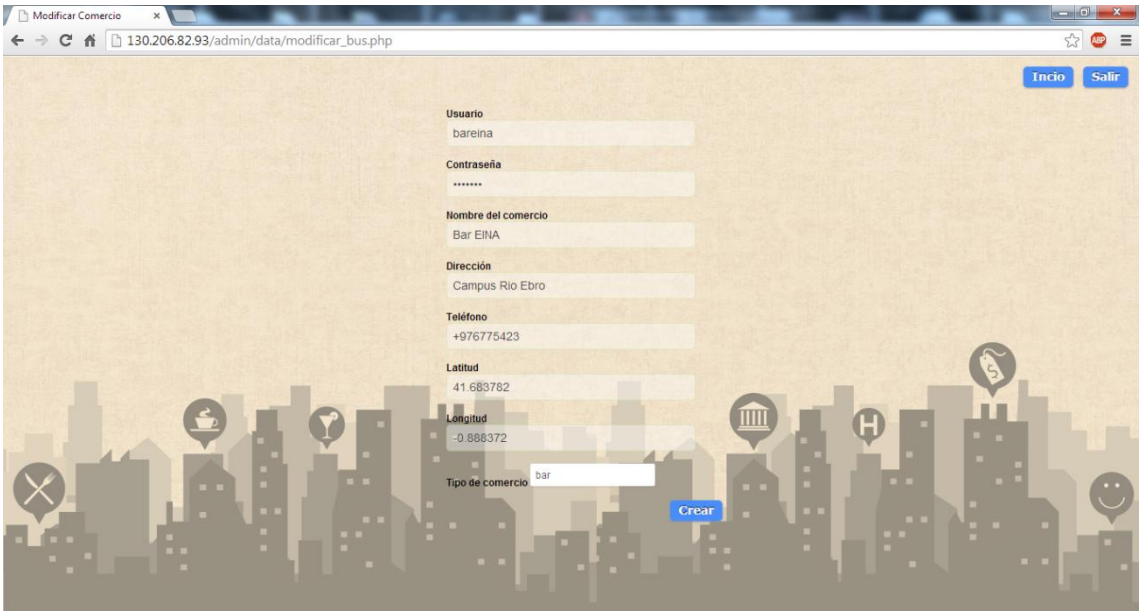

Figura B.6. Ventana que permite modificar el comercio seleccionado.

B.1.4 Informe de usuarios

Si se pulsa el botón *Usuarios*, se pasa a la ventana que contiene la tabla con todos los datos de los diferentes usuarios dados de alta en el sistema (ver Figura B.7), los cuales tienen acceso a la aplicación móvil *MyCity4Me*. Igual que en el informe de los comercios, existe la opción de borrar usuarios o modificar sus datos, pulsando los botones de borrar usuario y modificar usuario respectivamente. En el caso de pulsar *modificar usuario* se accede a la ventana representada en la Figura B.8

Figura B.7. Informe de usuarios.

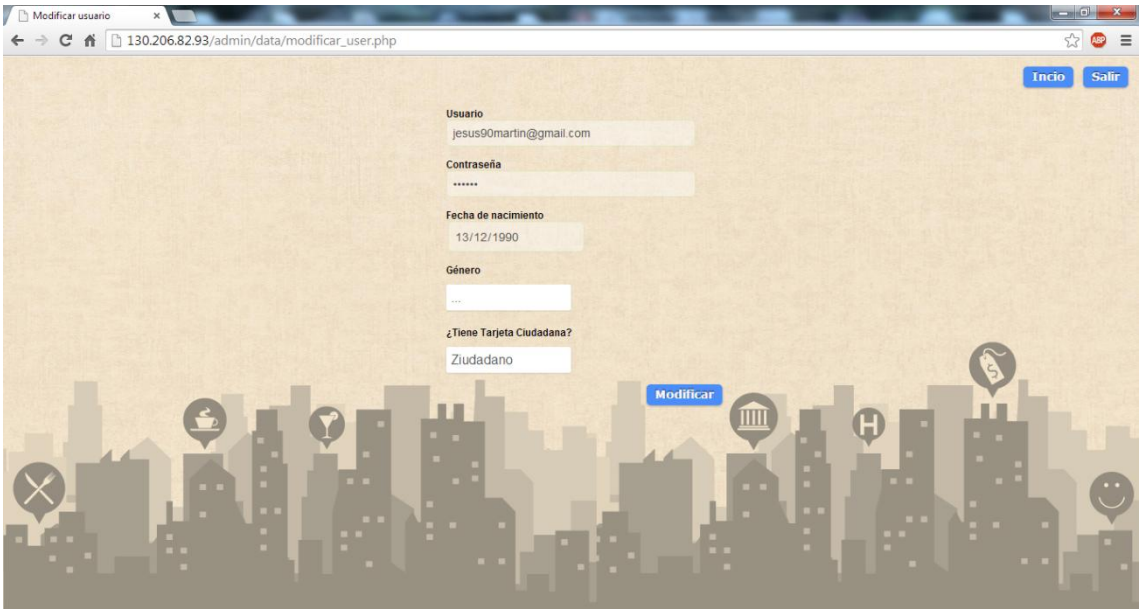

Figura B.8. Ventana que permite modificar el usuario seleccionado.

Desde cualquier ventana se puede salir del sistema pulsando el botón *Salir*.

B.2 Modo *comercio*

La aplicación para comercios, consiste en una web que permite a los comercios registrados en el sistema publicar ofertas y realizar un seguimiento de éstas. El comercio publica ofertas con unas características, y estas ofertas se publican en la aplicación Android *MyCiy4Me*, donde los usuarios podrán verlas y en caso de interesarles podrán

solicitarlas. Esta solicitud es como una reserva, pero sin cargo alguno, simplemente simboliza que un usuario le ha gustado la oferta e intentará ir a consumirla, Desde esta aplicación web, el comercio podrá ver cuántas solicitudes hay de una determinada oferta y que usuarios las han enviado, de manera que cuando un usuario la consuma, el comercio podrá registrar en el sistema que ese usuario ha comprado la oferta. Gracias a esto, el sistema genera una serie de datos muy valiosos para los comercios, por ejemplo, pueden saber cuáles son las ofertas más exitosas, a qué hora del día se consumen más ofertas, etc.

A continuación se describen las ventanas que forman la aplicación web en cuestión, junto con todas sus funcionalidades.

La primera ventana que aparece al acceder a la web es la ventana de *Login*, en la cual los comercios introducen sus credenciales de acceso para acceder al sistema. Su aspecto se puede observar en la Figura B.9.

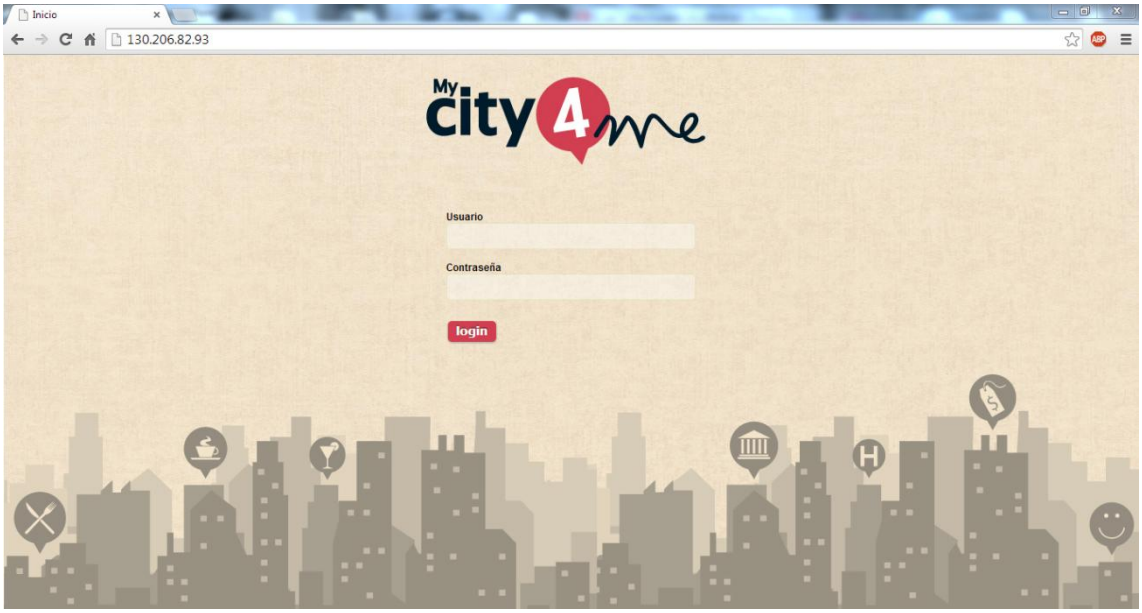

Figura B.9. Pantalla de Login.

Una vez dentro del sistema aparece la ventana principal, representada en la Figura B.10. Desde esta ventana se puede acceder al resto de las ventanas que forman la aplicación. Existen cuatro posibles funciones accesibles desde esta ventana: ver el perfil o modificarlo, insertar una nueva oferta, ver las ofertas activas o ver el historial de ofertas. Para acceder a dichas funciones basta con pulsar el botón que así lo indica.

Figura B.10. Pantalla de inicio.

B.2.1 Perfil del comercio

Pulsando sobre la imagen típica de perfil de usuario que aparece en la pantalla de inicio se accede a un formulario que contiene el perfil del comercio, desde donde existe la posibilidad de realizar cualquier modificación sobre sus datos. Este formulario aparece en la Figura B.11.

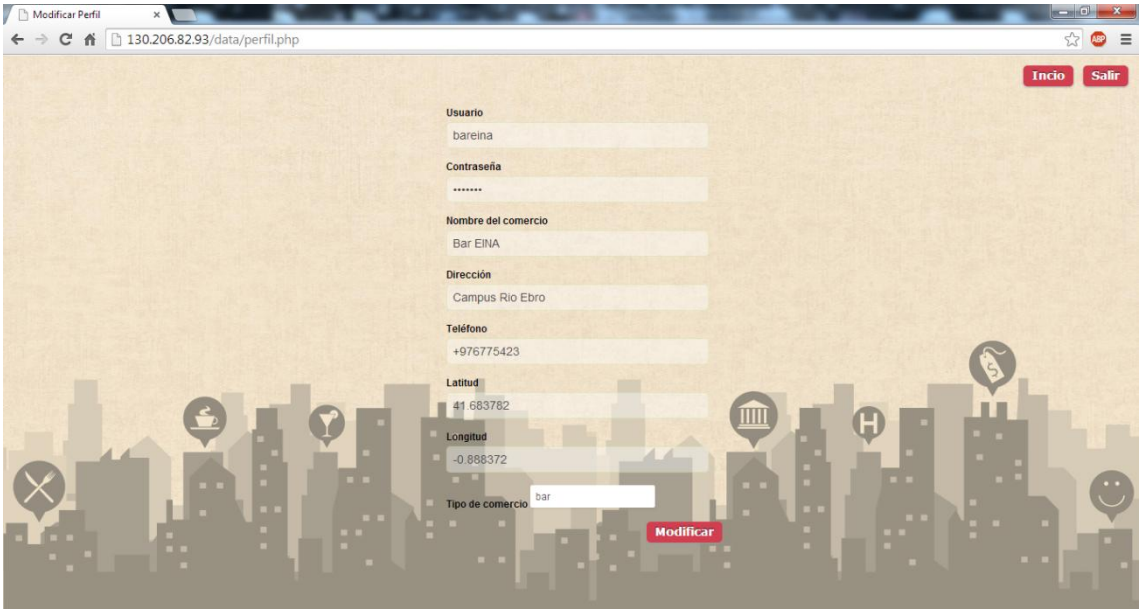

Figura B.11. Formulario del perfil de comercio.

B.2.2 Insertar Oferta

Si se pulsa el botón *Insertar Oferta*, se accede a otro formulario desde el cual se pueden introducir todos los datos de una oferta, es decir, descripción, horario de validez, género y rango de edad al que va dirigida, y si va dirigida a usuarios de la *Tarjeta Ciudadana* de Zaragoza o no. Esta ventana se presenta en la Figura B.12.

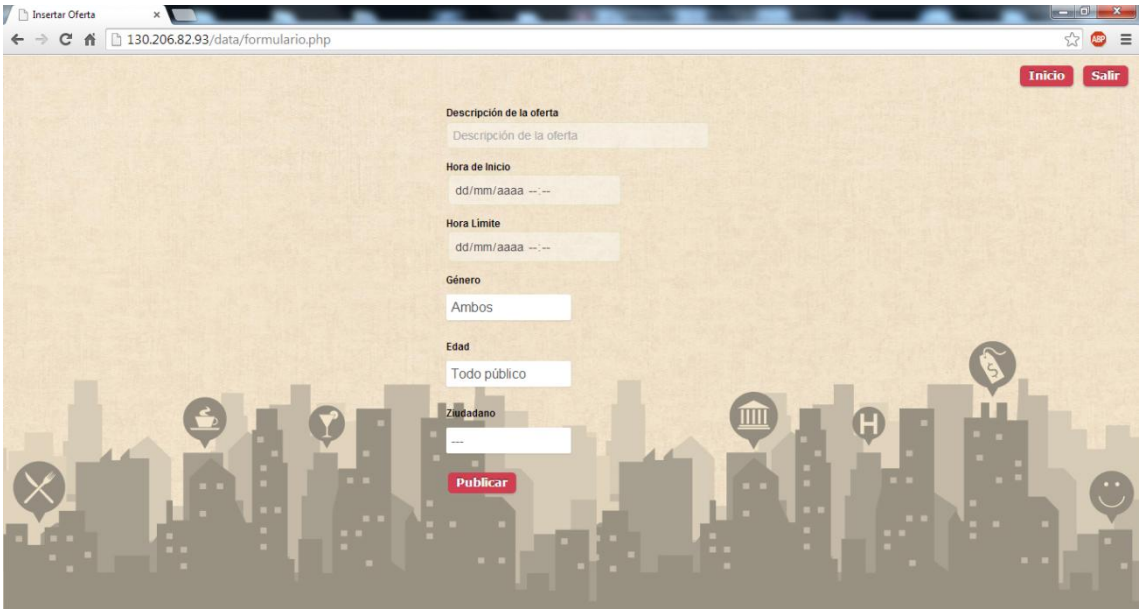

Figura B.12. Formulario de publicación de ofertas.

B.2.3 Ofertas Activas

Pulsando sobre el botón *Ofertas Activas*, se accede a una ventana que presenta una tabla con toda la información acerca de las ofertas publicadas que no han llegado a su hora límite de validez. Junto a esta tabla aparecen tres funciones: *Borrar oferta*, *Modificar oferta* y *Ver solicitudes*. Todo lo descrito se puede apreciar en la Figura B.13

Figura B.13. Ventana de Ofertas Activas.

Si el comercio desea borrar una de las ofertas, selecciona su identificador, que aparece en la tabla junto con sus datos, y pulsa el botón *Borrar Oferta*. La aplicación confirma si realmente se desea eliminar la oferta antes de realizar ninguna acción.

Si el comercio desea modificar una oferta, selecciona su identificador y pulsa el botón *Modificar Oferta*, el cual le traslada a una nueva ventana, apreciable en la Figura B.14, que presenta un formulario que permite la modificación de los datos de la oferta seleccionada.

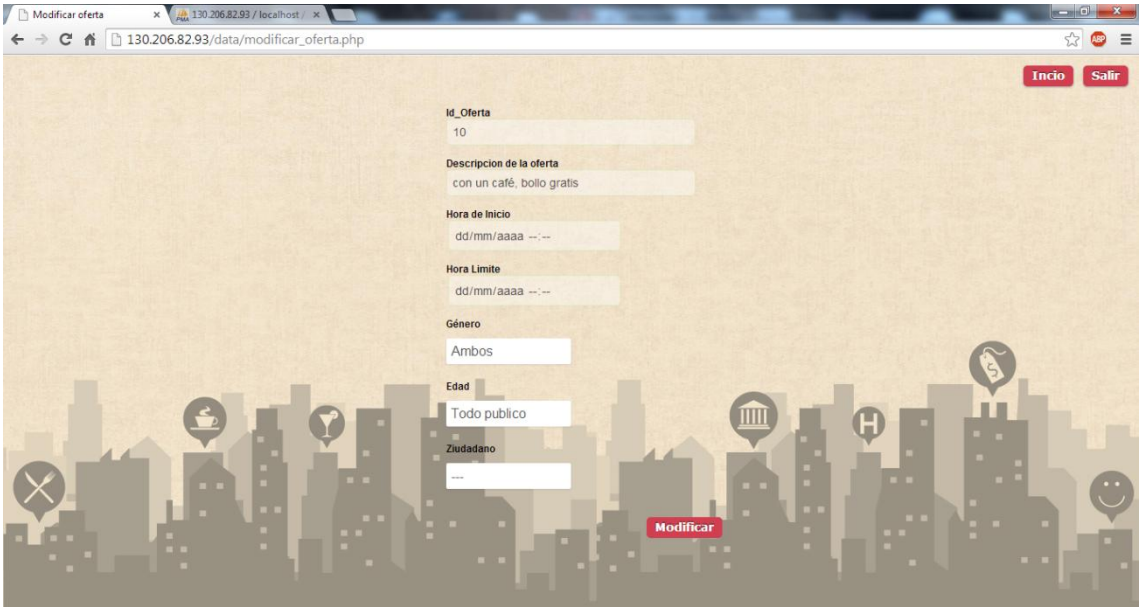

Figura B.14. Ventana de modificación de la oferta seleccionada.

Por último, si el comercio quiere ver las solicitudes que los usuarios de la aplicación móvil *MyCity4Me* han enviado, puede seleccionar el identificador de la oferta y pulsar el botón *Ver solicitudes*. De esta manera, la aplicación avanza hasta la ventana de solicitudes, que se puede ver en la Figura B.15, donde se muestran los usuarios que han enviado solicitudes de la oferta seleccionada, junto con la fecha exacta de dicha solicitud. Desde esta nueva ventana se puede confirmar si el usuario ha comprado la oferta. Para ello basta con seleccionar su id de usuario y pulsar el botón *Oferta Comprada*.

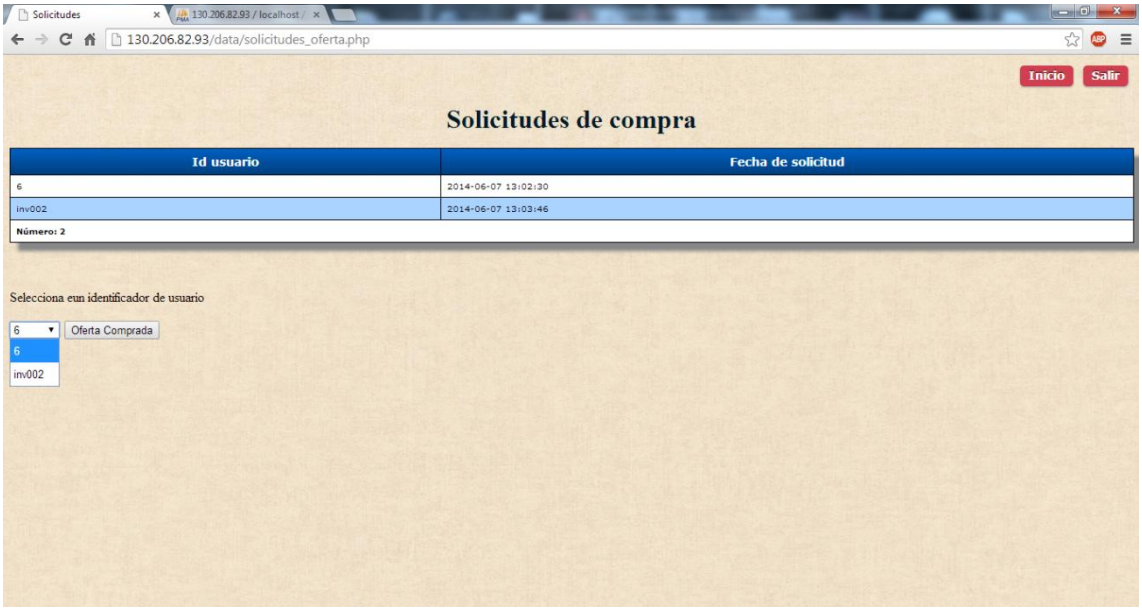

Figura B.15. Ventana con las solicitudes de la oferta seleccionada.

B.2.4 Historial de ofertas

Por último, si el comercio quiere ver un historial con los datos y algunas estadísticas de todas sus ofertas, puede pulsar el botón *Historial de Ofertas* y acceder a la ventana que se muestra en la Figura B.16.

Figura B.16. Historial de ofertas.

Desde todas las ventanas existe la posibilidad de volver a la pantalla de *Inicio*, o de *Salir* de la aplicación web.

ANEXO C - Fi-Lab, entorno de desarrollo de FI-WARE

C.1 ¿Qué es FI-Lab?

El entorno de trabajo de FI-WARE se conoce como FI-Lab [5]. Se trata de una instancia de FI-WARE a la que se puede acceder para realizar pruebas y experimentos que requieran recursos de la nube. Por su carácter experimental, hasta el momento se puede utilizar de forma gratuita. Además, tiene la finalidad de ser un medio de comunicación entre desarrolladores de aplicaciones y posibles empresas o inversores interesados en realizar desarrollos sobre FI-WARE. Dentro de FI-Lab podemos distinguir varias funciones, como podemos ver en la Figura C.1:

- *Cloud*: Esta utilidad permite la creación de instancias en una máquina virtual en la nube. Es la función utilizada para crear las máquinas virtuales sobre las que implementar el sistema descrito en este documento.
- *Store*: Permite publicar ofertas de servicios desarrollados, así como adquirir aquellos que resulten de interés para tus aplicaciones.
- *Mashup*: Sirve para crear *mashup*[1] de aplicaciones web con cierto valor añadido, incorporando componentes como datos, lógica e interfaces de usuario *(widgets*[2]*)*.
- *Account*: Lugar donde registrar aplicaciones en FI-WARE, asociándoles permisos y roles.
- *Help&Info*: En este apartado se dispone de información acerca del entorno de trabajo, como por ejemplo video-tutoriales.

[1] *Mashup*: Se define como una aplicación que usa y combina contenido de más de una fuente, para crear un nuevo servicio, visualizado en una única interfaz gráfica.

[2] *Widget*: Se define como una pequeña aplicación, generalmente presentada en archivos o ficheros pequeños que son ejecutados por un motor de widget *(Widget Engine)*, cuya finalidad es dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

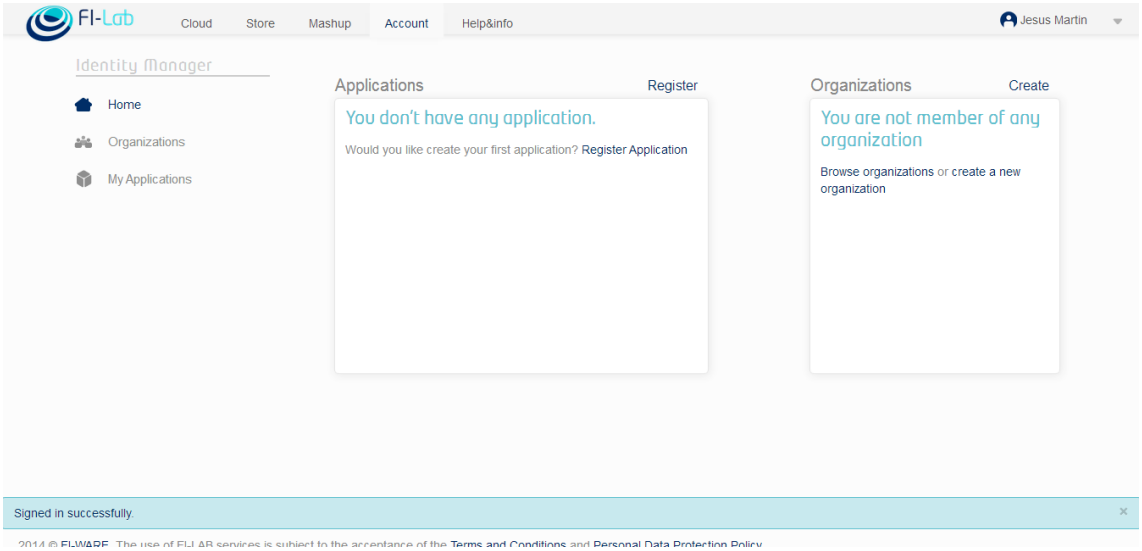

Figura C.1. Pantalla principal FI-Lab.

C.2 Su uso en el presente proyecto: creación de instancias o máquinas virtuales.

Para proceder a la creación de un servidor virtual, primero hay que acceder a la pestaña *cloud*. Dentro de este apartado aparecen una serie de opciones a la izquierda de la pantalla (Figura C.2), que serán explicadas según vayan siendo requeridas.

En primer lugar se deben definir una serie reglas de seguridad, y para ello se accede a la página *Security*. Esta página está organizada en tres apartados, *Floating IPs*, *Security Groups* y *Keypairs*.

En el primer apartado aparecen las direcciones IP públicas asociadas a tus servidores virtuales, para poder acceder a ellos remotamente (Figura C.2). El segundo consiste en un conjunto de reglas que pueden ser definidas en tu servidor virtual, para permitir o denegar el acceso a ciertos servicios o puertos. En este sistema se definen las reglas mostradas en la Figura C.3. El último apartado sirve para crear, importar o eliminar archivos de claves públicas o privadas, con los que posteriormente se accederá vía *ssh* al servidor virtual, ya que el acceso mediante usuario y contraseña esta desactivado por defecto (Figura C.4).

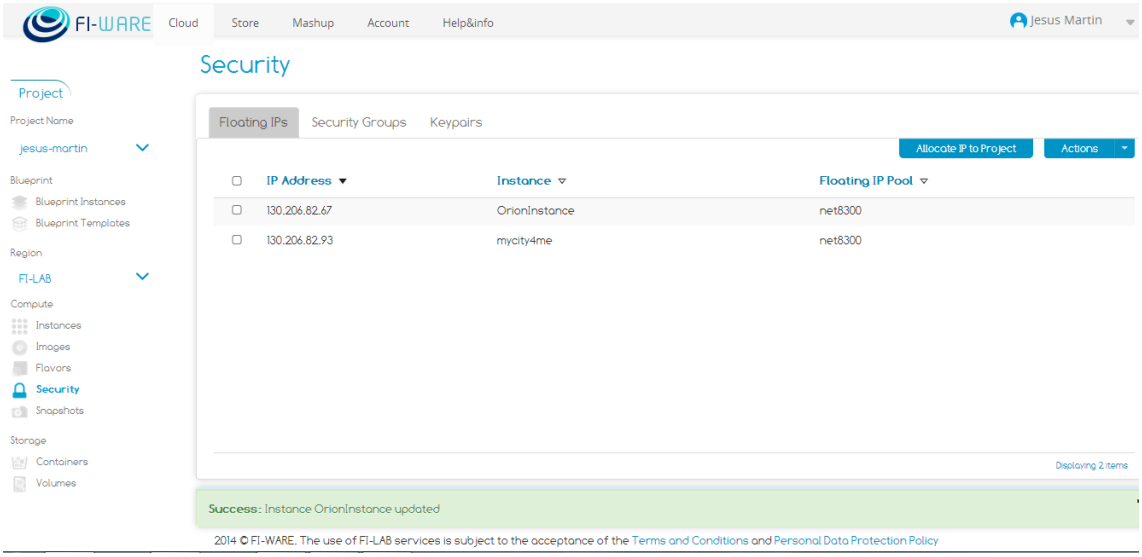

Figura C.2. Sección de Floating IPs.

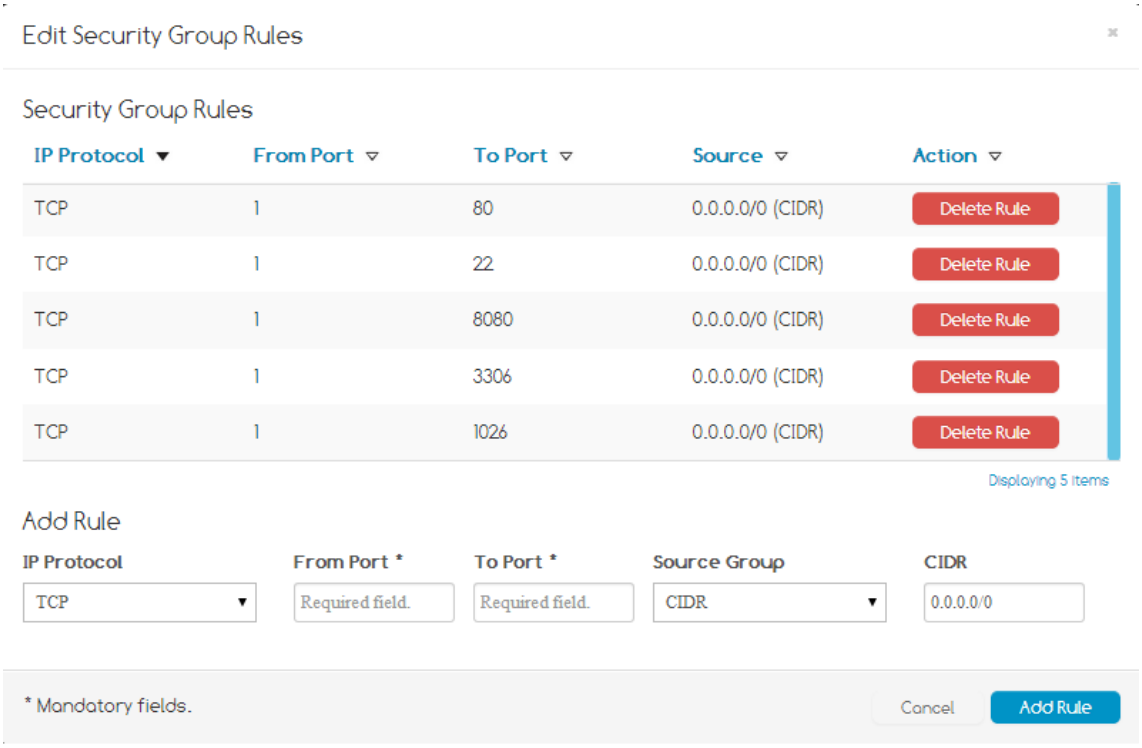

Figura C.3. Reglas de acceso de ciertos servicios o puertos.

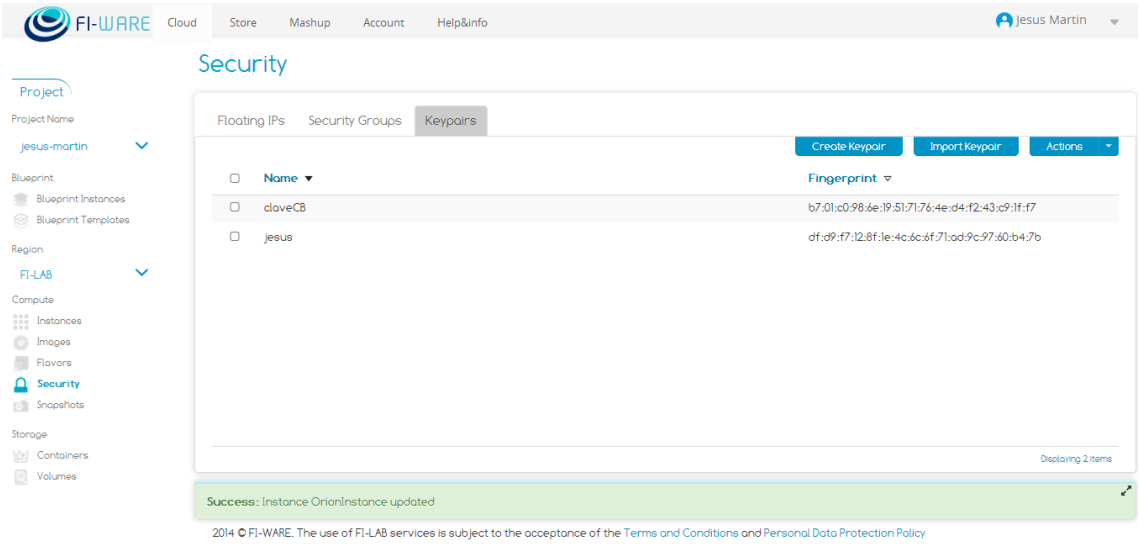

Figura C.4. Sección de Keypairs.

Ahora que se han definido las reglas de seguridad podemos proceder al lanzamiento del servidor virtual. Para ello accedemos a la pestaña denominada *Images*, y aparecen una serie de posibles imágenes que lanzar. Elegimos la imagen llamada *Ubuntu12.04-server-x86_64* y pulsamos *Launch.* Aparece una pantalla en la que se definen las características del servidor. En primer lugar elegimos nombre, características técnicas, numero de instancias (Figura C.5) y pasamos al siguiente apartado, donde le asociamos la *keypair* y el *security group* definido en el apartado anterior (Figura C.6). Por último nos muestra un resumen de las características de la instancia y se lanza (Figura C.7). Tras unos minutos, la instancia estará lanzada y en funcionamiento.

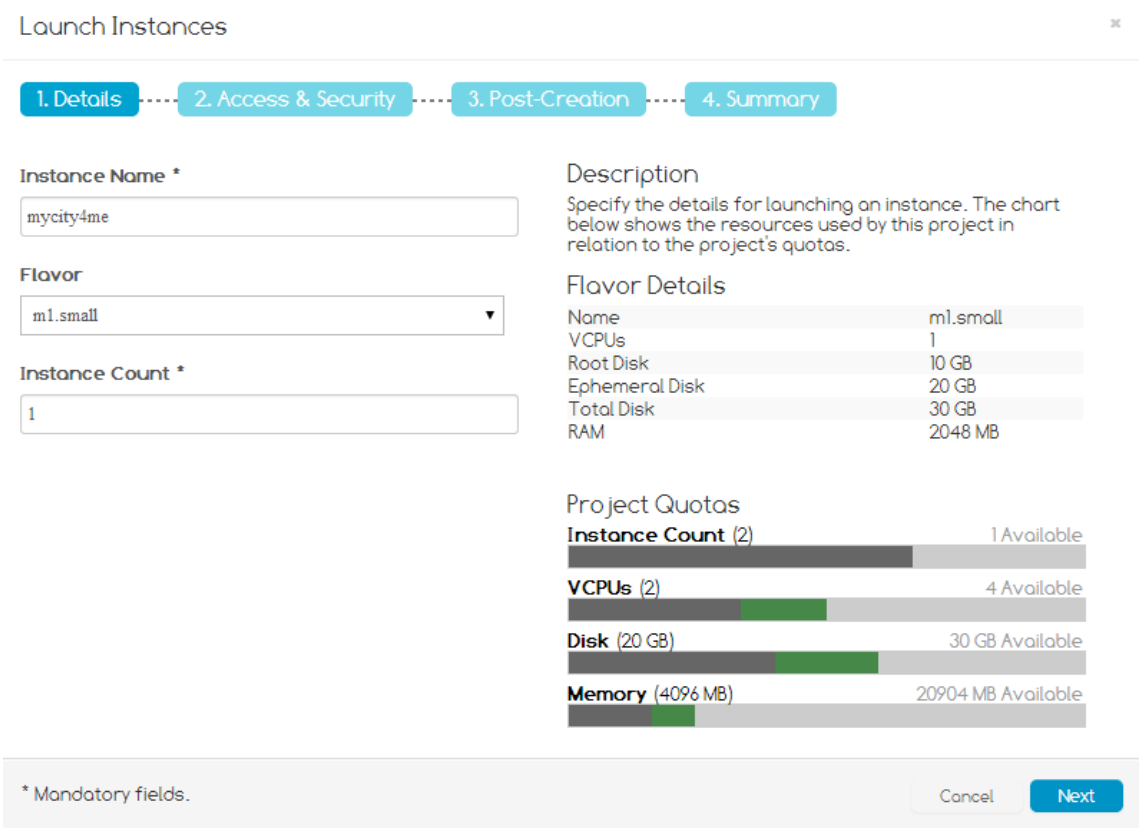

Figura C.5. Detalles técnicos de la instancia a lanzar.

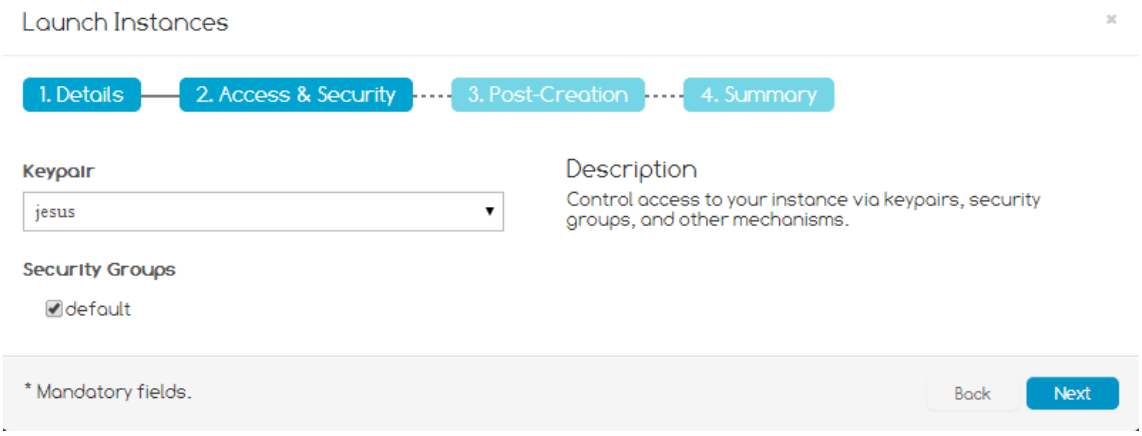

Figura C.6. Asociación de keypair y security groups.

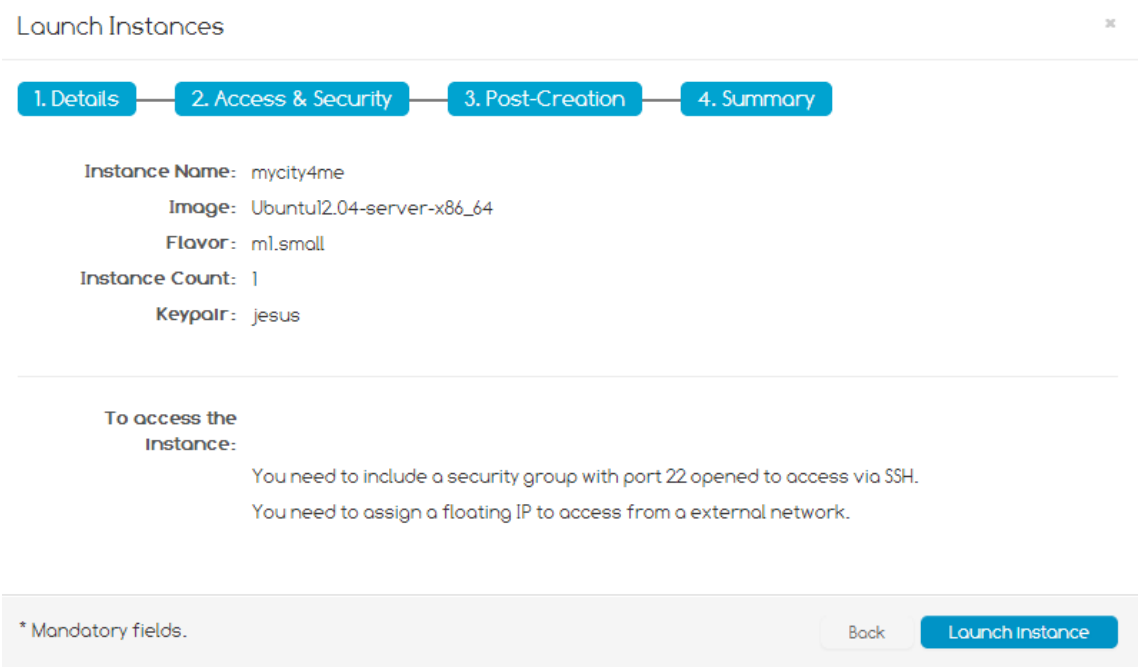

Figura C.7. Resumen de las características de la instancia.

Finalmente, para poder acceder al servidor virtual vía *ssh* debemos asociarle una dirección IP pública en el apartado *Security-Floating IPs*.

Para llevar a cabo la instalación de componentes en las máquinas virtuales, en este caso el paquete de instalación de Apache, PHP y MySQL, existen infinidad de tutoriales en la red que ayudan a realizarla[3].

[3] http://drupalalsur.org/videos/instalar-y-configurar-un-servidor-apache-en-ubuntu

ANEXO D - Puesta en marcha del sistema *Orion-Cygnus-Cosmos*

Para poder hacer uso de *Orion*, existen dos opciones, la primera consiste en crear una instancia, como se ha explicado en el Anexo C y posteriormente instalar una imagen de dicho GE; y la segunda consiste en desplegar una instancia que tiene instalado *Orion* por defecto (imagen: *orion-psb-image-R3.3*). Se ha optado por esta última opción debido a que todas las actualizaciones que se llevan a cabo por parte de los técnicos de FI-WARE sobre este GE se instalan fácilmente dada su correcta configuración por defecto.

Orion gestiona tanto información de contexto, mediante el interface *NGSI10*, como la disponibilidad de la información de contexto, mediante el interface *NGSI9*. En este proyecto sólo se contempla el primer caso, es decir, se gestiona la información de contexto utilizando la API NGSI10[4], concretamente información de entidades, como podría ser la temperatura de un coche.

En primer lugar se describen las distintas operaciones que este GE permite realizar, tal y como se explica en [26], que son cinco:

- *updateContext*: Sirve para actualizar la información de contexto.
- *queryContext*: Sirve para realizar consultas de la información guardada.
- *subscribeContext*: Sirve para realizar suscripciones a determinados datos, de manera que, cuando ocurre algo (en un intervalo periódico de tiempo, o cuando la información sufre un cambio) se envía una notificación.
- *updateContextSubscription*: Sirve para actualizar una suscripción ya creada.
- *unsubscribeContext*: Sirve para dar de baja una suscripción.

Para realizar estas operaciones, el GE permite hacer uso de lenguaje XML o JSON. En este documento se representan ejemplos con ambos formatos.

En este proyecto, el objetivo es capturar la posición geográfica de los usuarios de la aplicación móvil cada vez que se conecten. Para este fin, utilizando la operación *updateContext* (Figura D.1), se crea a entidad, llamada "*Usuario*" formada por un atributo, llamado "*posición*", que contiene las coordenadas generadas por el dispositivo

[4] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/NGSI-9/NGSI-10_information_model

móvil del usuario en el momento de entrar a la aplicación. El campo *updateAction* puede tener 3 valores, APPEND, UPDATE o DELETE:

- APPEND: Añade la nueva entidad con sus atributes o, si ya existe, la actualiza.
- UPDATE: Actualiza los atributos de una entidad existente.
- DELETE: Borra la entidad indicada.

Figura D.1. Código XML y JSON utilizado para crear una nueva entidad, con su posición.

Una vez recibida la solicitud, *Orion* crea la entidad en su base de datos interna, actualiza el valor de sus atributos y devuelve la respuesta de la Figura D.2.

Figura D.2. Respuesta de Orion al recibir una operación de tipo updateContext.

Una vez que la entidad se ha creado, podemos simular la recepción de coordenadas. Para ello realizamos operaciones del tipo *updateContext*, de esta manera se puede ver si el sistema captura nuevos usuarios y sus coordenadas o bien, si el usuario ya es miembro de la entidad, se puede ver si se actualiza su información o no.

Para verificarlo se pueden hacer consultas, mediante operaciones *queryContext* como la de la Figura D.3, de manera que se puede comprobar cuáles son las ultimas coordenadas almacenadas de la entidad indicada, tal y como se aprecia en la Figura D.4.

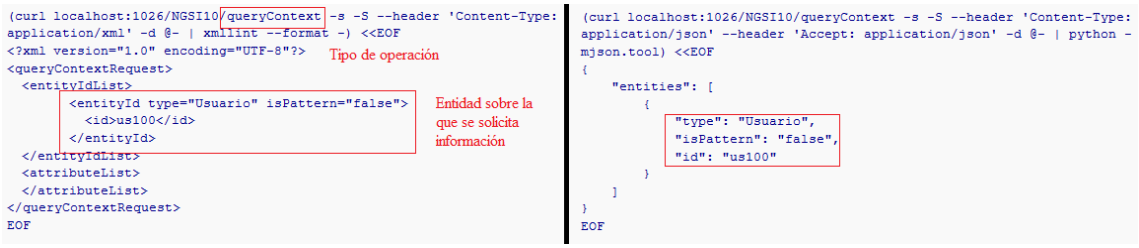

Figura D.3. Ejemplo de operación queryContext.

Figura D.4. Respuesta a la operación queryContext.

Además de la consulta anterior, *Orion* permite realizar consultas mucho más útiles cuando trabajamos con coordenadas. Por ejemplo, se puede realizar una consulta como la de la Figura D.5, solicitando todas las entidades que cumplan que sus coordenadas están dentro de un radio, definido por un punto central y un radio de expansión en metros. Esto podría servir para saber cómo se distribuyen los usuarios de la aplicación sobre un mapa de la ciudad, información que podría utilizarse para mejorar los comercios, pues podrían poner publicidad en las zonas más concurridas.

Figura D.5. Ejemplo de queryContext especial.

Orion por defecto no es capaz de almacenar un historial con todas las actualizaciones de datos que se realizan a lo largo del tiempo, y es por eso por lo que necesitamos el uso de *Cosmos*

Esto se realiza por medio del conector *Cygnus*, tal y como se describe a continuación.

Para trabajar sobre *Cosmos* no es necesario crear otra máquina virtual, ya que existe una instancia global correctamente configurada y actualizada. Para acceder a ella es necesario solicitar unas credenciales de acceso. Una vez obtenidas se puede comenzar a trabajar sobre dicho GE.

El acceso a Cosmos se realiza mediante el comando:

```
ssh mycityforme@130.206.80.46
```

donde "*mycityforme*" es el nombre de usuario y "130.206.80.46" es la dirección IP en la que está desplegado *Cosmos*.

Llegados a este punto, por un lado, se dispone de una máquina virtual con *Orion* y por otro, de una instancia global con *Cosmos*, y debemos establecer una conexión entre ambas, que se lleva a cabo mediante el conector *Cygnus*.

Para ello, se accede vía *ssh* a la máquina virtual de *Orion* y por medio de comandos se configura el uso combinado de estos tres elementos, explicado paso a paso a continuación:

- Se instala y se configura *Cygnus*, típicamente en la misma máquina que *Orion*, siguiendo los pasos descritos en [19]
- Se crea un nuevo directorio en *Cosmos (user/mycityforme/data*), donde se almacenan los datos enviados por *Orion* y que se llamará igual que el directorio

de almacenamiento incluido en el archivo de configuración de *Cygnus* como lugar donde almacenar los datos que le llegan.

- Se ejecuta *Cygnus* mediante el comando:

```
Nohup APACHE_FLUME_HOME/bin/flume-ng agent --conf
APACHE_FLUME_HOME/conf -f APACHE_FLUME_HOME/conf/cygnus.conf
-n orionagent -Dflume.root.logger=INFO, LOGFILE &
```

- Se realiza una suscripción manual de *Cygnus* en *Orion*, especificando la URL donde *Cygnus* estará escuchando notificaciones (definida en el archivo de configuración de *Cygnus*), las entidades y atributos que se quieren guardar en cosmos, la duración de la suscripción y las condiciones necesarias para que se escuchen las notificaciones. Dicha suscripción se muestra en la Figura D.6.
- Por último, se comprueba que los datos se están guardando correctamente en el directorio creado en *Cosmos*. Para ello se accede a *Cosmos* y se escribe el comando `hadoop fs -ls /user/mycityforme/data`, el cual muestra los archivos guardados en ese directorio, tal y como aparecen en la Figura D.7. También se puede ver el contenido de los archivos guardados (Figura D.8).
- Además, el sistema crea automáticamente una tabla, que contiene los datos almacenados en *Cosmos*, preparada para realizar consultas SQL sobre ella, mediante un *Hive*[5] *CLI* local. También permite que el usuario cree una tabla de acuerdo a sus necesidades.

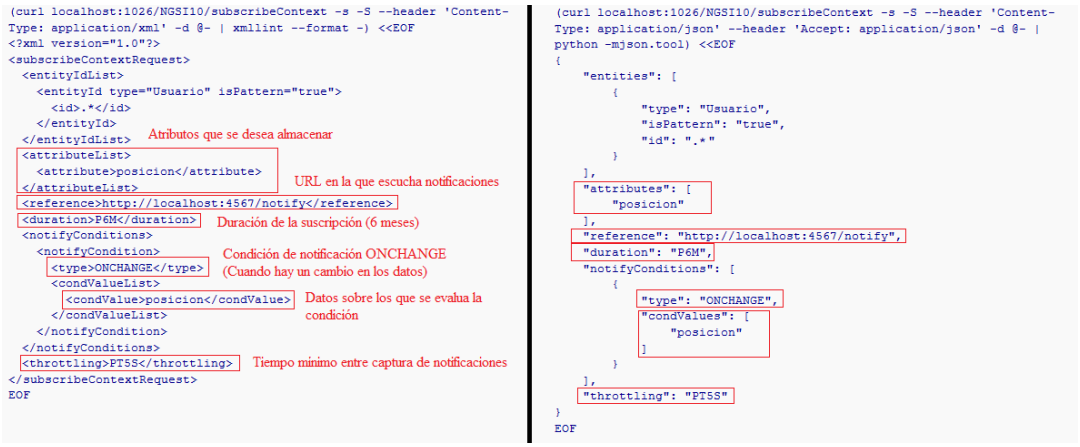

Figura D.6. Código XML y JSON de la suscripción de Cygnus en Orion.

[5] http://archive.cloudera.com/cdh4/cdh/4/hive/index.html

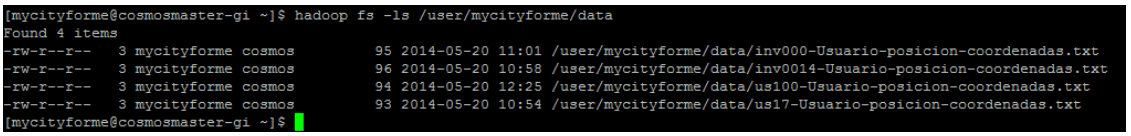

Figura D.7. Ejemplo notificación guardada en Cosmos.

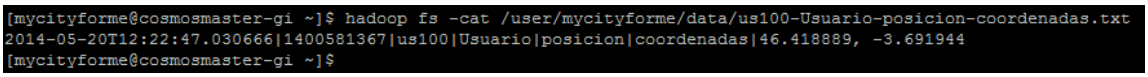

Figura D.8. Ejemplo de datos de la notificación guardados en Cosmos.

ANEXO E - Implementación del servicio web PHP

Un servicio web sirve principalmente para intercambiar datos entre aplicaciones. Distintas aplicaciones de software, desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos.

En este proyecto, se ha desarrollado un servicio web sencillo, basado en un conjunto de archivos PHP, cuya función va a ser intercambiar datos entre la aplicación Android y la base de datos principal.

Este conjunto de archivos PHP está formado por:

- Un archivo de configuración (*config.php*). Archivo utilizado para definir las variables de configuración de la base de datos.
- Un archivo de conexión (*connectbd.php*). Este archivo se encarga de establecer la conexión con la base de datos.
- Un archivo de funciones (*funciones_bd.php*). En este archivo se programan todas las funciones que se encargan de realizar las consultas a la base de datos, ya sean de tipo SELECT, UPDATE o INSERT. Una consulta de tipo SELECT sirve para consultar registros de la base de datos que satisfagan un determinado criterio. Una consulta UPDATE sirve para modificar los valores de los campos y registros especificados. Una consulta tipo INSERT se encarga de introducir datos en la base de datos con una única operación.
- Un archivo PHP para cada tipo de consulta que se quiera realizar desde Android. Estos archivos reciben los parámetros enviados desde Android vía HTTP y llaman a una de las funciones programadas en el archivo *funciones_bd.php*, que devuelve los datos solicitados. Estos datos se codifican en formato JSON y se devuelven a Android donde son decodificados y utilizados.

No obstante, para entender mejor cómo funciona, se plantea el siguiente ejemplo de uso, explicado paso a paso.

1) La aplicación Android solicita conocer cuáles son las ofertas activas de un comercio concreto.
2) Este envía la petición POST a la URL donde se encuentra el servicio web

encargado de ello (*http://130.206.82.93/android/businessdeals.php*), pasando como parámetro el nombre del comercio.

3) *Businessdeals.php* llama a "*takebusinessdealsmenu($nombre_comercio)*", función guardada en *funciones_bd.php*. Este archivo requiere el archivo *connectdb.php* que a su vez requiere el archivo *config.php*, y de esta manera se establece la conexión con la base de datos.
4) Una vez la conexión a la base de datos ha sido establecida, la función realiza una consulta SELECT solicitando las ofertas activas del comercio que se ha pasado como parámetro.
5) Por último el resultado devuelto por la función se codifica en JSON escribiendo "*echo json_encode($resultado);"* dentro del archivo *businessdeals.ph*p y se devuelve a Android, donde se decodifica.

ANEXO F - Elementos típicos de una aplicación Android

Para entender mejor algunos conceptos utilizados en la redacción de la parte del desarrollo de la aplicación Android es oportuno incluir una breve definición de estos.

Una aplicación Android está formada por una serie de componentes esenciales. Cada componente es un punto de unión entre el sistema operativo y la aplicación en sí. Para este proyecto, únicamente vamos a comentar algunos de los más importantes:

- *Activity*: Componente de una aplicación que proporciona una pantalla con la que los usuarios pueden interactuar. Cada *Activity* es una pantalla distinta con su propia interfaz.
- *Intent*: Objeto de mensajería que se usa para solicitar la acción de otro componente de la aplicación y facilitan la comunicación entre componentes de diversas maneras. Se suelen utilizar para:
    - Inicializar un *Activity.* Un *Activity* es una pantalla particular de la aplicación, y se puede lanzar mediante un *Intent* escribiendo "*startActivity();*".
    - Inicializar un servicio. Un servicio es un componente que realiza operaciones en segundo plano, sin una interfaz de usuario, escribiendo "*startService();*".
    - Difundir un *broadcast.* Un *broadcast* es un mensaje que puede recibir cualquier aplicación. Este mensaje se lanza escribiendo, por ejemplo, "*sendBroadcast();*".
- *Marker*: Icono que indica un punto de interés en el mapa.
- *InfoWindow*: Proporciona información adicional a los Marker.
- *Aplication Framework*: proporciona los bloques de construcción que se utilizan para crear aplicaciones. Los más destacados son, *Activity Manager*, *Content Provider*, *Resource Manager*, *Location Manager*, *Notification Manager.*

Además de todos estos componentes, existe un documento esencial en todo proyecto Android llamado *AndroidManifest.xml*. Se trata de un archivo XML encargado de la configuración de la aplicación. Algunas de las principales funciones de dicho archivo se enumeran a continuación:

- Identificar los permisos de usuario que la aplicación requiere, tales como acceso

a Internet, o a la localización, etc.

- Declarar la versión de la API *(Application Programming Interface)* de Android que se va a utilizar. En nuestro proyecto la aplicación parte de la versión 4.0, lo que equivale a la API 14.
- Declarar las características del software y hardware utilizado o requerido por la aplicación, tales como el servicio de GPS *(Global Positioning System)*.
- Definir la configuración de cada una de los *Activity* que van a formar parte de la aplicación como el uso de sensor de giro, o la aparición de la entrada de texto automática.

También dispone de una serie de directorios o carpetas donde se almacenan los archivos de *Layout*, las imágenes utilizadas en los fondos de pantalla o los iconos, los archivos XML de diseño de botones, los menús de la barra de acción, etc.

A continuación se explica brevemente cuáles son estas carpetas y para qué se utilizan en *MyCity4Me*.

- **Layout**: Guarda los *Layout* correspondientes a los *Activity* que únicamente disponen de vista vertical, o que la vista horizontal la tienen por defecto.
- **Layout-port**: Almacena los *layout* verticales.
- **Layout-land**: Almacena los *layout* horizontales.
- **Drawable-xxpi**: Sirve para guardar las imágenes que se utilizan en la aplicación, iconos, fondos, *Markers*, etc. También sirve para guardar XML de diseño de botones.
- **Menu**: En este directorio se guarda la configuración de las barras de acción de los distintos *Activity.*
- **Values**: En esta carpeta se guardan los *Strings* que contienen todo el texto predefinido de la aplicación en español. También se guarda el formato de algunos elementos como los S*pinner,* conocidos como listas desplegables.
- **Values-en**: En esta carpeta se guardan los Strings que contienen todo el texto predefinido de la aplicación en inglés.

ANEXO G - Diseño y navegación de la aplicación Android

Como ya se ha comentado en la memoria, el diseño de la aplicación se ha realizado al mismo tiempo que su implementación. De esta manera los diseños de interfaz realizados en primera instancia se han ido actualizando a lo largo del proyecto.

El diseño global de la aplicación se ha llevado a cabo en dos pasos, el primero consiste en un prototipado inicial de las ventanas, y el segundo consiste en la creación de la navegación entre dichas ventanas

Prototipado de las ventanas de navegación.

Como ya se dispone de los requisitos funcionales de la aplicación, la tarea llevada a cabo en esta fase se ha reducido a buscar la mejor forma de representar esos requisitos, de manera que el usuario quede satisfecho con el uso de la aplicación. Es bien sabido que un diseño atractivo puede marcar una gran diferencia en cuanto al éxito de una aplicación.

Para llevar a cabo el prototipado se ha utilizado una herramienta propia del SDK de Android, que mediante archivos XML permite crear el aspecto de la ventana y además dispone de un editor gráfico que muestra al instante cómo va quedando dicha ventana.

En la Figura G.1 se puede observar un ejemplo de dicho editor.

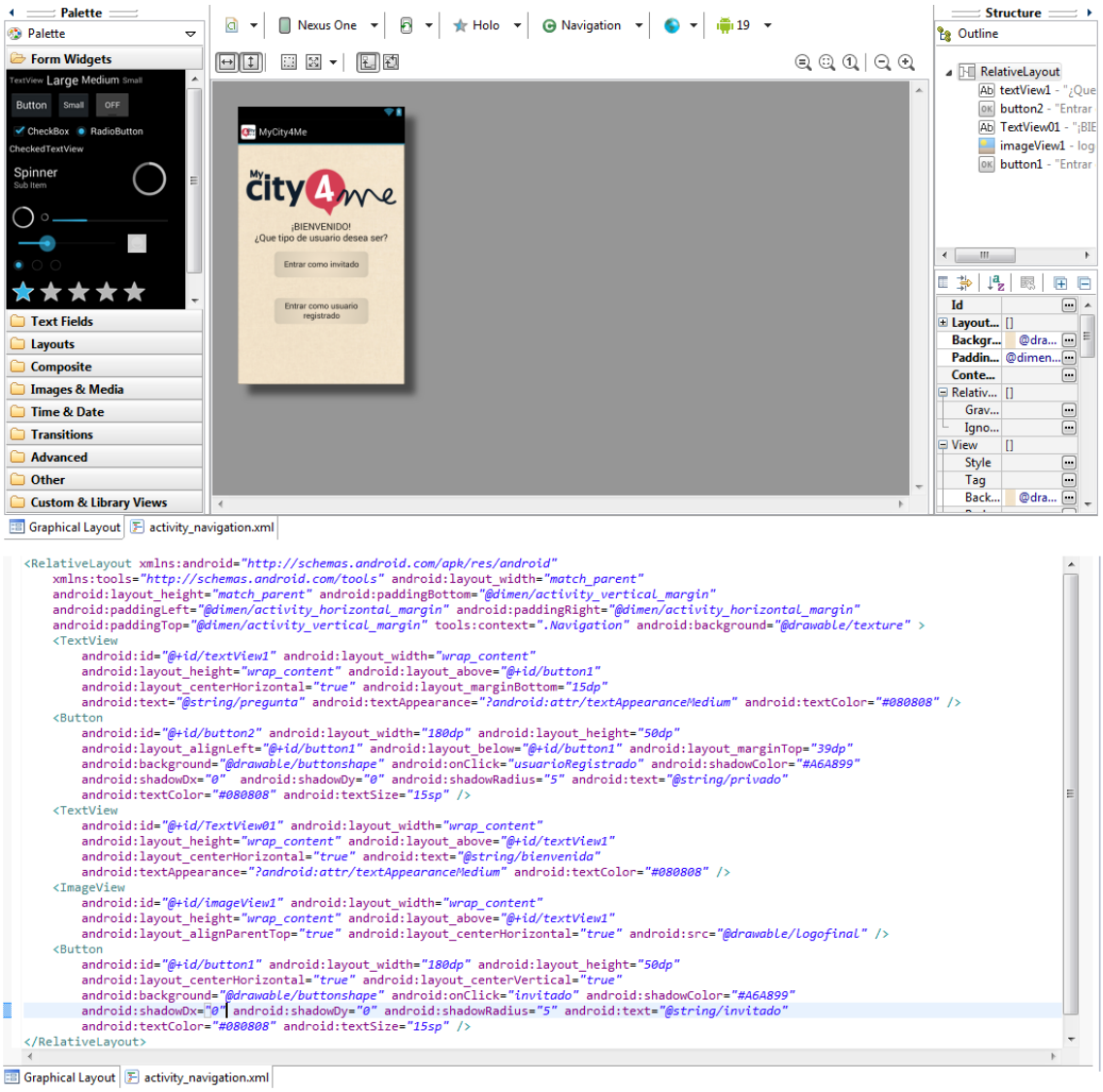

Figura G.1. Editor XML del entorno de programación utilizado para el diseño de la interfaz de usuario de MyCity4Me.

Navegación de la aplicación

Tras la creación de los prototipos de la pantalla, se ha realizado un análisis de cómo debería ser la navegación entre ellos. No obstante, no ha sido hasta la fase final del proyecto cuando se ha definido por completo como debe ser la navegación entre las ventanas.

En la Figura 2.18 (apartado 2.3.6) se puede observar un esquema que describe la navegación entre las ventanas.

ANEXO H - Activación de los mapas de Google en *MyCity4Me*

Para activar los mapas es necesario incluir la librería *google_play_services_lib.jar en el* proyecto Android y obtener una *API key*, que se puede adquirir en repositorio de herramientas de Google[6]. Esta *API key* hay que incluirla en el *androidManisfest.xml* junto con otras líneas de código, es decir, se debe incluir el siguiente código XML:

```
<uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

<meta-data          <!--API Key-->

        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyBbeGtpeAU08Wh6wiDEwW7d-7NPYI5r21c" />

<meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
```

Posteriormente, en el *activity* donde se va a activar el mapa se debe extender el *activity* a otro *activity* llamado "`android.support.v4.app.FragmentActivity`" y por último, se adjunta el siguiente código al *activity*:

```
private GoogleMap map;

//Lanzamos el mapa
map =((SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.map))
.getMap();
//boton geolocalizar
map.setMyLocationEnabled(true);
//mi localizacion
yo=initlocation();
// posicion inicial de la cámara
CameraPosition cameraPosition = new CameraPosition.Builder()
            .target(yo)                    // centro de la camara
            .zoom(16)                      // zoom
            .tilt(80)                      // inclinacion
            .build();                      // Crea la  posicion desde el builder


map.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
```

Al finalizar este proceso, los mapas están listos para usarse.

[6] https://code.google.com/apis/console/b/0/?noredirect#project:137687004243:services

ANEXO I - Metodología de desarrollo de la aplicación Android

La metodología de desarrollo consiste en la reiteración de las siguientes fases:

- Diseño gráfico de la ventana mediante la edición del archivo XML correspondiente, con la ayuda del asistente gráfico del entorno de programación.
- Análisis de la información necesaria en la ventana de navegación correspondiente, así como la obtención y presentación de la misma.
- Dotación de interacción con el usuario a todos los elementos presentes en la ventana (botones, cuadros de texto, etc.)
- Análisis "prueba y error" de cada componente.

I.1 Diseño gráfico de la ventana

En una aplicación Android, la interfaz de usuario se construye a partir de objetos *View* o *ViewGroup*, grupo de varias *View*, que son las unidades básicas que conforman la interfaz de usuario en la plataforma Android.

La clase *View* sirve como base para subclases llamadas *widgets* u objetos de la interfaz de usuario, como botones o campos de texto. Mientras que la clase *ViewGroup* sirve como base para subclases llamadas *layout*, que ofrecen diferentes tipos de estructura al diseño de la interfaz. Existen diferentes tipos de *layout* como lineales, listas, tabulares o relativos. En la Figura I.1 se pueden observar algunos ejemplos.

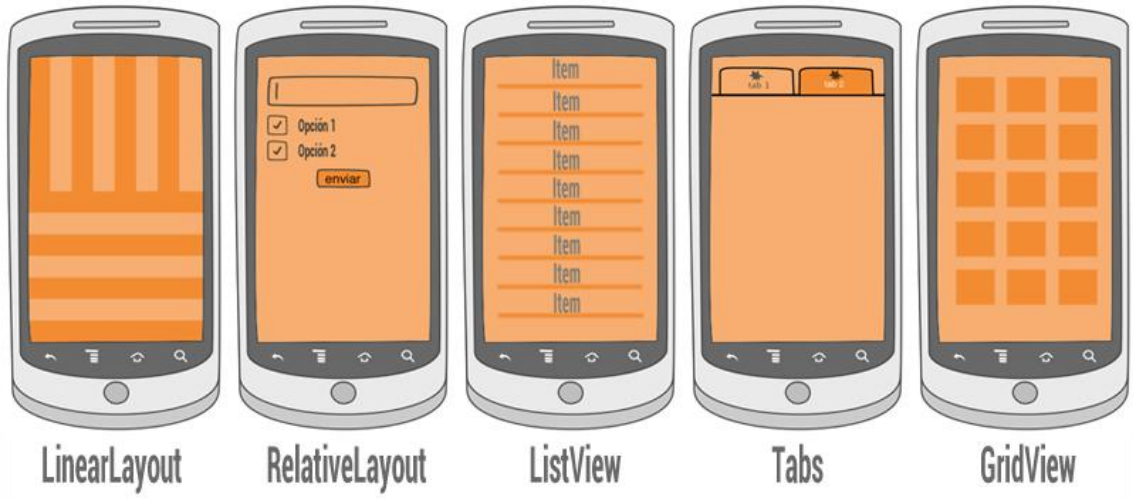

Figura I.1. Ejemplos de tipos de layout en Android.

Cada *View* tiene propiedades que guardan parámetros de diseño y contenido para un área rectangular específica de la pantalla. Algunos de estos parámetros pueden ser las dimensiones del objeto, el diseño, el estilo, posicionamiento respecto a otros *View*, etc. Además, cada una de estas *View* hereda algunos parámetros del *ViewGroup* que lo contiene.

Un objeto *View* es también un punto de interacción para el usuario y un receptor de eventos de interacción. En este anexo, donde se habla de la dotación de la interacción, se describen de manera general los eventos táctiles detectables y los correspondientes métodos de activación que se han utilizado para crear la interacción deseada entre el usuario y la aplicación Android.

El método más común para definir el diseño gráfico es mediante el uso de archivos XML o *layout.* Estos archivos XML definen el diseño de las ventanas de navegación, conocidas como *activities*, obtenidas durante la fase de diseño.

El entorno de programación utilizado en este proyecto, y que se nombra en el Anexo K "Herramientas utilizadas", incluye un editor de interfaz de ventanas para los archivos XML, el cual puede dar una idea previa de cómo va a quedar la ventana sin necesidad de tener que ejecutar la aplicación en el móvil.

En el Anexo G "Diseño y navegación de la aplicación Android" ya se ha visto el aspecto de dicho editor.

I.2 Análisis, obtención y presentación de la información

El siguiente paso, después de la elaboración de la interfaz gráfica de una ventana concreta, y teniendo en cuenta el modelo de datos, consiste en realizar un análisis de la información necesaria, para después dotar de la funcionalidad necesaria a los *widgets* o elementos que forman la ventana.

Tal y como se ha comentado anteriormente, la forma de adquirir dicha información es a través de fuentes de datos, ya sean internas o externas.

Fuentes de información internas: Este término se refiere a las fuentes de datos accesibles en el mismo dispositivo móvil, sin necesidad de ninguna conexión.

En este proyecto se han utilizado las siguientes:

- Fichero *sharedPreferences*: Se trata de una clase de Android utilizada para guardar sobre todo datos de configuración. Estos datos se almacenan en un

fichero XML en forma de clave-valor y se guardan en el sistema de archivos, de la siguiente forma:

/data/data/com.nombre_de_la_app/shared_prefs/nombre_fichero.xml

En este PFC se utiliza para guardar los datos de acceso de los usuarios, de manera que si no cierran la sesión, la siguiente vez que entren a la aplicación no necesitan volver a introducir sus credenciales de acceso.

- Software del móvil: Para obtener información a partir del uso de aplicaciones nativas de Android, se utilizan *Intent* predefinidos en el SDK.

  Estos *Intent* llevan información adicional que es interpretada por Android y este devuelve la información requerida. Algunos ejemplos utilizados en *MyCity4Me*, para extraer información acerca de GPS:

  -locManager.isProviderEnabled(LocationManager.GPS_PROVIDER)

  Se informa al sistema de que se quiere saber si el proveedor GPS está activado o no. De esta forma, en función de su estado, se puede mostrar o no un mensaje para activarlo.

  -Intent myIntent = new Intent(Settings.ACTION_SETTINGS);

  Se informa al sistema de que se quiere acceder a los ajustes del dispositivo.

- Hardware del móvil (GPS): La API de Android ofrece librerías que contactan el hardware del dispositivo para el que la aplicación tenga permiso de uso. Estos permisos se incluyen en el archivo de configuración *AndroidManifest.xml*.

  Ejemplos de permisos incluidos en dicho archivo:

  ```
  -<uses-permission android:name="android.permission.INTERNET" />
  -<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  ```

  El permiso de uso de hardware es muy importante, ya que utilizado de modo incorrecto plantearía inconvenientes al usuario. Es por esto que cuando se va a descargar la aplicación, primero se pregunta al usuario sobre el uso permitido de los componentes hardware utilizados en la aplicación.

Fuentes de información externas: En este caso sí es necesaria una conexión a Internet. Existen diversos medios para comunicarse con este tipo de fuentes, como pueden ser Bluetooth, Wi-Fi, red de datos, etc. En este caso únicamente se utiliza el acceso a la red

de datos. Para tener acceso a Internet desde la aplicación es necesario solicitar permisos al usuario antes de la instalación de la misma.

Las fuentes externas que se han usado son:

- Servicios web: la aplicación móvil puede tener acceso a la información de bases de datos externas, aunque no de forma directa. Para ello se crean los llamados *servicios web*, que actúan de intermediarios entre la aplicación y las bases de datos.

  En el caso de *MyCity4Me*, como la base de datos donde se encuentra la información que se quiere mostrar en la aplicación se ha creado también en este proyecto, y se encuentra en una máquina virtual a la que se tiene acceso, se ha tenido que desarrollar un servicio web propio. Este servicio web está basado en un conjunto de archivos PHP que son invocados desde Android, de manera que, pasando los parámetros necesarios, devuelven la información requerida. El esquema del servicio web se muestra en la Figura I.2.

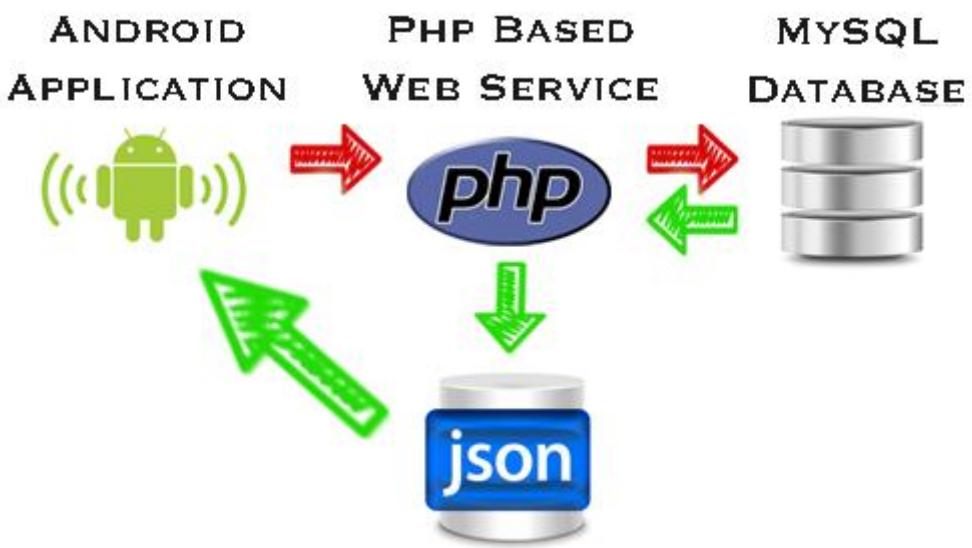

Figura I.2. Sistema de comunicación entre Android y la base de datos.

- Servidor de Google: La descarga y visualización de los mapas se lleva a cabo a través del servidor de Google. Para ello se ha incluido la librería *google_play_services_lib.jar,* y de esta manera la aplicación se conecta con un servidor para realizar la descarga del mapa y su posterior visualización.

Además la aplicación Android también hace uso del navegador de Google Maps. Cuando un usuario selecciona la oferta y pulsa el botón de *"Cómo llegar"*, la aplicación llama al servicio de Google Maps, junto con las coordenadas origen (usuario) y destino (comercio), y éste representa la ruta a seguir para llegar hasta el establecimiento.

I.3 Dotación de interacción

Una vez que se ha realizado el interfaz y se ha obtenido la información, se procede a dotar de funcionalidad a los distintos elementos que forman la ventana, para que el usuario interaccione con ella.

En Android existen diversas formas para capturar la interacción entre el usuario y la aplicación. Nos referimos a capturar eventos originados a partir de un objeto *View*, como por ejemplo la pulsación de un objeto, un texto o una imagen.

Para capturar dichos eventos, Android posee unas interfaces llamadas *Listener*. Un *Listener* es una interfaz dedicada a detectar cualquier tipo de contacto sobre un *widget* o elemento en el que ha sido registrado dicho *Listener*. Para cada *Listener* existen distintos métodos de activación. Estos métodos son los que permiten realizar una acción u otra en función del tipo de evento capturado.

A continuación se presenta una breve descripción de los métodos más comunes en Android.

- **Método *onClick***

  Este se trata del método más común. Se activa al capturar un evento de pulsación simple (Figura I.3-(a)) con un solo dedo sobre uno de los elementos Android.

- **Método *onLongClick***

  Este método captura eventos de pulsación larga, cuando el elemento permanece pulsado más de un segundo.

- **Método *onTouch***

  Este método captura eventos de cualquier tipo, pulsación, liberación o cualquier gesto de movimiento sobre la zona donde está el *widget* programado con este método.

- **Método *onFling***

  Este método está encargado de capturar un evento de pulsación simple seguido

de un desplazamiento horizontal (Figura I.3-(b)) con un solo dedo.

- **Método *onScroll***

  Este método captura eventos de pulsación simple seguida de movimiento vertical (Figura I.3-(c)) con un solo dedo.

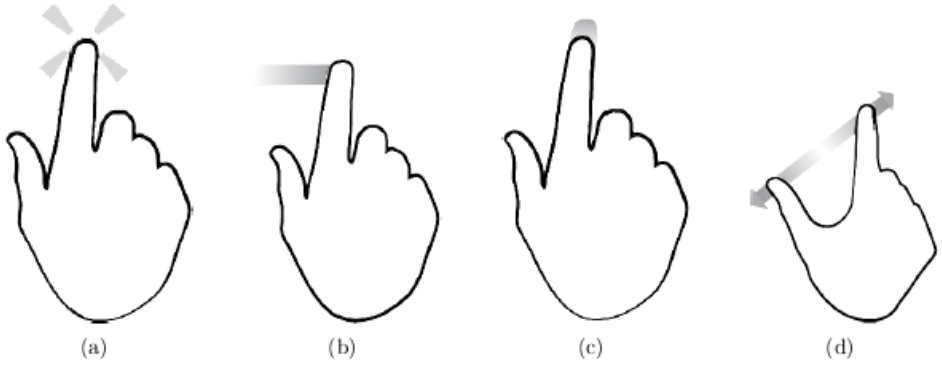

Figura I.3. Métodos de interacción más comunes de Android. (a) Pulsación simple (b) Desplazamiento horizontal (c) Desplazamiento vertical (d) Zoom.

En *MyCity4Me* únicamente se van a utilizar el método *onClick* en el caso de botones, texto, etc. y el método *onScroll* en la lista de ofertas. El mapa, al ser una librería externa, tiene sus propios métodos programados, como pulsación doble, desplazamientos verticales y horizontales, zoom (Figura I.3-(d)), etc.

I.4 Pruebas de integración en la aplicación

Una vez realizadas las fases anteriores, queda un último paso, que consiste en realizar una serie de pruebas para verificar el correcto funcionamiento de los distintos elementos que forman las ventanas.

Estas pruebas son básicamente tres:

- **Prueba sobre cada elemento independientemente**. Se comprueba que la acción a realizar por dicho elemento se lleva a cabo correctamente.
- **Prueba de cada elemento, teniendo en cuenta el funcionamiento del resto**. Se comprueba que la acción desempeñada por el elemento no afecta negativamente al comportamiento del resto de elementos.
- **Prueba de integración de la ventana completa con el resto.** Se comprueba que al pasar de una ventana a otra a través del uso de *Intent*, se envía correctamente la información deseada.

En el caso de que en el transcurso de alguna de estas pruebas se detecte un funcionamiento incorrecto, el entorno utilizado para el desarrollo de la aplicación proporciona una serie de herramientas que facilitan la labor de localización y las causas de los errores. A través de la herramienta de depuración se puede ejecutar el código instrucción por instrucción, comprobando el cambio en el estado de los registros, el valor de las variables en uso, etc., localizando así el origen del fallo.

Además, Android ofrece una herramienta denominada *LogCat*, que proporciona al desarrollador un mecanismo para visualizar los mensajes de salida del sistema de depuración. De esta manera, cuando ocurre un error, se puede consultar dicha herramienta y buscar cuál ha sido el fallo y en qué lugar del código se encuentra.

ANEXO J - Metodología de desarrollo de la aplicación web

El método empleado en el desarrollo de la aplicación web se explica en los siguientes apartados:

J.1 Diseño gráfico de la ventana

En el desarrollo web hay infinidad de opciones a la hora de pensar en el diseño. En este proyecto, desde el principio se ha pretendido que el diseño de las ventanas del navegador, en ambos modos de uso, sea lo más sencillo e intuitivo posible, facilitando la labor a los usuarios. En el desarrollo se ha utilizado HTML 5 para que se represente perfectamente en los dispositivos móviles y se ha utilizado CSS 3 para el diseño de tablas y botones.

Para llevar a cabo la labor de diseño se ha utilizado el entorno de trabajo descrito en el Anexo K "Herramientas utilizadas". Este entorno permite programar el diseño al mismo tiempo que se observa cómo va quedando, tal y como se puede observar en la Figura J.1. De esta manera, se ha podido realizar la tarea de diseño al mismo tiempo que la implementación del resto de la aplicación web, lo cual facilita bastante la labor.

Para cumplir el requisito de que sea una aplicación sencilla e intuitiva se ha decidido incluir una botonería grande con nombres que permiten saber rápidamente cual es la función a la que se accede.

Para representar la información hay muchas opciones. No obstante, por simplicidad y eficiencia se ha optado por utilizar tablas, ya que, al fin y al cabo, la información que se va a mostrar como mejor se entiende es clasificándola en filas y columnas.

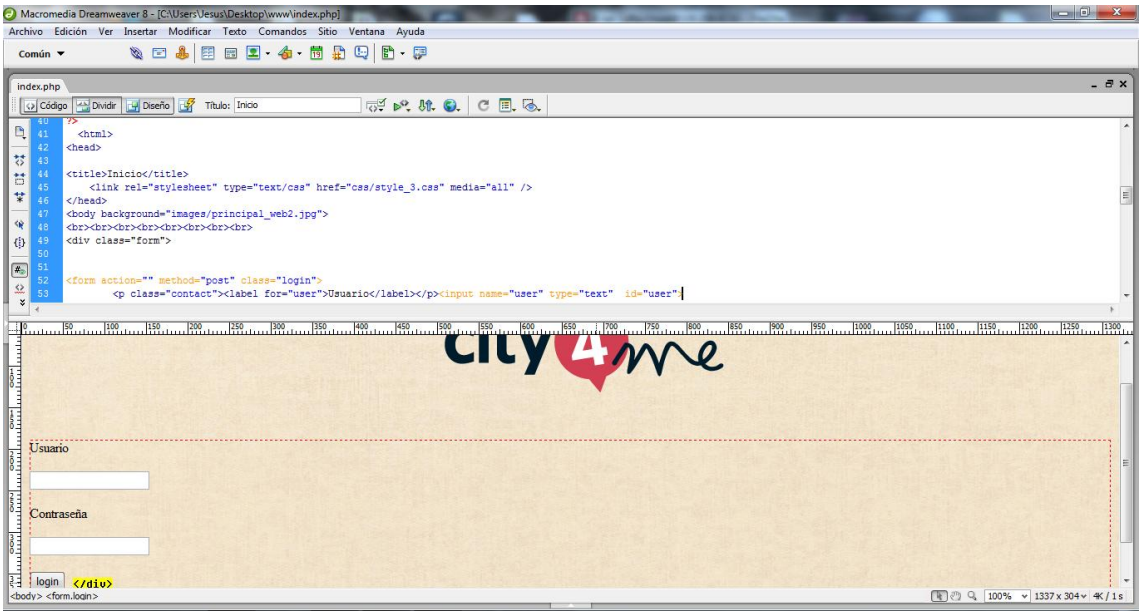

Figura J.1. Editor gráfico de DreamWeaver.

Los formularios que permiten la introducción de datos, ya sean sobre nuevos usuarios, desde el modo administrador, o sobre ofertas desde el modo comercio, son unos formularios básicos con cajas de introducción de texto asociados a un sobrenombre que indica que es lo que se debe escribir en dichas cajas. En aquellos casos en los que las opciones a escribir son limitadas se han utilizado desplegables que muestran las distintas posibilidades.

J.2 Análisis, obtención y presentación de la información

Una vez se ha optado por un modelo de datos y el diseño a utilizar en la aplicación web, se realiza un análisis de cómo obtener la información que se quiere representar en la aplicación web y de cómo almacenar la información obtenida de las ofertas publicadas por los comercios, o en el caso del administrador, información acerca de cuentas de usuario.

Mediante programación PHP se obtiene la información a mostrar procedente de la base de datos principal y se lleva a cabo el almacenamiento de la información, introducida en la aplicación web, en dicha base de datos.

En el archivo de programación de cada ventana se establece una conexión con la base de datos del sistema, de manera que mediante PHP se pueden realizar consultas MySQL sobre la base de datos, ya sea para obtener datos, como para introducirlos o actualizarlos.

La información que se introduce en los formularios (publicación de ofertas o creación de usuarios) se almacena directamente en la base de datos y la información obtenida mediante consultas a esa base de datos se muestra conforme al diseño adoptado en el apartado anterior.

J.3 Dotación de interacción

Una vez hemos determinado el diseño de la ventana y se obtenido la información que se quiere representar, se procede a la dotación de interacción para que el usuario pueda interactuar con ella.

La interacción en la aplicación web es muy sencilla, ya que era uno de los requisitos iniciales, únicamente se basa en la introducción o selección de texto y la pulsación de botones. Para dar la función necesaria al hacer *click* en uno de los botones después de rellenar un cuadro de texto o seleccionar una de las opciones que aparezcan, basta con poner la referencia y enviar por POST los parámetros introducidos o seleccionado, al archivo PHP que se encarga de realizar dicha función. Un ejemplo de código utilizado en este PFC es:

```
<form  name="form3" method="post" id="form3">
<input id="modificar" name="modificar" placeholder="modificar"  type="text">
<input class="buttom" value="Modificar usuario" type="submit" name="button"
id="button" onClick="this.form.action = 'modificar_user.php'" />
</form>
```

La primera línea define el estilo y el método de envío, la segunda identifica la caja de texto, identificador que será la clave utilizada para recibir la petición post. También se define el texto a mostrar en de fondo. La tercera línea define el botón, el nombre que aparece sobre él, y la acción que debe realizar al pulsarlo (*onClick*). El resultado es el mostrado en la Figura J.2.

Figura J.2. Ejemplo de caja de texto y botón.

En el archivo *modificar_user.php* se recibe el post escribiendo:

```
$id_user = $_POST['modificar'];
```

De esta manera, en la variable *$id_user* tendremos el dato introducido en la caja de texto después de pulsar el botón.

J.4 Pruebas

Por último, se realiza una fase de pruebas sobre la ventana en desarrollo, que se divide en tres partes:

- **Prueba sobre cada elemento independientemente**. Se comprueba que la acción a realizar por dicho elemento se lleva a cabo correctamente, sin errores.
- **Prueba de cada elemento, teniendo en cuenta el funcionamiento del resto de elementos.** Se comprueba que la acción desempeñada por el elemento no afecta negativamente al comportamiento del resto de elementos.
- **Prueba de integración de la ventana completa con el resto de ventanas.** Se comprueba que al pasar de una ventana a otra la información se muestra correctamente.

En el caso de que en el transcurso de alguna de estas pruebas se detecte un funcionamiento incorrecto, se procede a su resolución. Dependiendo de dónde venga el error se puede localizar fácilmente o no. Por ejemplo, si el error viene de una consulta SQL, el sistema detecta en qué parte de la consulta se encuentra el error, mientras que si el error está en el código HTML o PHP, el error hay que localizarlo repasando el código línea a línea ya que el entorno de trabajo utilizado no tiene un sistema de detección de errores.

La aplicación web resultante tras la iteración de este proceso de desarrollo se puede contemplar en el Anexo B “Manual de usuario de la aplicación web”.

ANEXO K - Herramientas utilizadas.

Para la creación de la estructura del sistema en la nube de FI-WARE se ha hecho uso de FI-Lab, entorno web en fase experimental que permite realizar pruebas de desarrollo, así como crear máquinas virtuales sobre las que trabajar, que requieran recursos de la nube.

Para el acceso a las máquinas virtuales vía ssh, se ha utilizado el terminal de Linux, cuando se ha trabajado sobre Linux y Putty[7] o Filezilla[8] cuando se ha trabajado sobre Windows. Filezilla se ha utilizado principalmente para la gestión de archivos de la aplicación web y el servicio web PHP, mientras que Putty se ha utilizado principalmente para la gestión de los Generic Enabler (GE) de FI-WARE

La aplicación web ha sido programada mediante lenguaje PHP y HTML 5, diseñando los estilos con CSS 3. Para ello se ha utilizado DreamWeaver[9], entorno que permite programar aplicaciones web estándar, además de disponer de una herramienta de diseño CSS, lo cual te permite programar al mismo tiempo que se observa el resultado de la interfaz visual.

La aplicación Android ha sido programada principalmente en lenguaje Java, en el que está basado Android, utilizando Eclipse[10] como principal herramienta de trabajo. Eclipse es una plataforma de desarrollo Open Source y multiplataforma basada en Java. La característica más destacable de Eclipse es la extensibilidad, ya que es una gran estructura formada por un núcleo y múltiples *plugin* que van conformando la funcionalidad final.

En el desarrollo del servidor web PHP, se ha utilizado también DreamWeaver por comodidad en el desarrollo de código PHP, ya que este detecta los distintos elementos del programa utilizando distintos colores lo que facilita el reconocimiento de dichos elementos en el programa global.

Para la creación y configuración de la base de datos en la web se ha utilizado PhpMyAdmin[11] que es una herramienta de código libre creada para administrar bases de datos MySQL.

[7] http://www.putty.org/
[8] https://filezilla-project.org/
[9] http://www.adobe.com/es/products/dreamweaver.html
[10] http://www.eclipse.org/
[11] http://www.phpmyadmin.net/home_page/index.php

Para la edición de los diagramas de bloques de este documento se ha utilizado LibreOffice Impress, Microsoft Office Acces y Microsoft PowerPoint.