



Trabajo Final de Grado – Ingeniería de Sistemas

PlaceOn

Red social georreferencial genérica

Alumnos: Jorge Bolpe - Fernando Vitale

Director: Mg. Oscar Nigro

Codirectora: Ing. Sandra González Císaro

Agradecimientos

Durante estos años de carrera fue fundamental el apoyo incondicional que hemos tenido por parte de quienes queremos mencionar como pequeña muestra de agradecimiento y cariño.

Gracias:

A mi familia por el apoyo brindado. En especial a mis viejos Jorge y Juliana, y mi abuela Perla; por ser un ejemplo de vida a seguir para mí; y por darme tantas veces darme las palabras de aliento y el empuje que he necesitado, para continuar con el estudio y “meterle” cuando las ganas eran esquivas; y demostrarme siempre que el sacrificio vale la pena.

Jorge

--

Mario mi “viejo” y Beatriz, “mami”; más que nadie ustedes saben los obstáculos que juntos tuvimos que superar para llegar a este momento. Ya llegamos, gracias por siempre.

San, mi hermana y “madre tandilense”. Mi cuñado Andrés, y sobrinos Juanse, Emilia y Andresito, que me reciben como el hijo/hermano mayor cada vez.

An, mi hermano, con quien comparto charlas técnicas y a quien admiré siempre, y Leonela mi sobrina intelectual, su fiel reflejo, ¿no querés venir a estudiar Sistemas acá?

Eva, mi amor, quien aguantó fines de semana de estudio y tesis. Listo, terminé lo dije.

Fernando

--

Por último, ambos queremos agradecer también a nuestros familiares, amigos y compañeros que nos acompañaron desde el primer día en la facultad, a los profesores que día a día eligen trabajar para que nosotros consigamos llegar a este momento que nos toca vivir.

A Oscar y Sandra nuestros directores, gracias por la ayuda, la confianza depositada, y por la paciencia.

Y una dedicatoria muy especial para “el Adrián” nuestro amigo y referente técnico, que se fue pero siempre está, nos vemos a la vuelta viejo, te extrañamos.

Contenido

Agradecimientos	1
Índice de Figuras	4
Capítulo 1 Introducción.....	5
Capítulo 2 Marco teórico	7
Red geosocial.....	7
Realidad aumentada	8
Estado actual de redes populares	8
Google Latitude	9
Google+	10
Glympse.....	10
I'm Here	10
Life360 – Family Locator	11
Family by Sygic	11
Foursquare	11
Estado “local” de aplicaciones georreferenciales	11
Capítulo 3 Análisis funcional y de Requerimientos de “PlaceOn”	13
Requerimientos funcionales	13
Requerimientos no funcionales	13
Capítulo 4 Diseño arquitectónico de “PlaceOn”	15
Primer análisis arquitectónico.....	15
Refinamiento arquitectónico	15
MVC Características generales.....	21
Elección de Framework	21
¿“App nativa” o “Web app”?	22
Web app	22
App Nativa	22
Yii Framework.....	23
Capítulo 5 Diseño detallado	28
Modelo de datos	28
Descripción de las tablas (Fig. 5)	30
Descripción de los modelos.....	30

Descripción de la Vista	33
Descripción del Controlador.....	33
Capítulo 6 PlaceOn en funcionamiento	36
Diagramas de secuencia.....	56
Login	56
Configurar filtro de alerta.....	57
Crear estado sharedUrl	57
Capítulo 7 Posibles Mejoras y Trabajos futuros.....	59
Capítulo 8 Conclusiones	60
Bibliografía	61

Índice de Figuras

Fig. 1 Descomposición en Cliente-Servidor	17
Fig. 2 Descomposición en MVC	19
Fig. 3 Descomposición en el Módulo de Información	20
Fig. 5 Diagrama del Modelo de Datos	29
Fig. 6 Diagrama de Clases de Modelos.....	31
Fig. 8 Pantalla inicial	36
Fig. 9/10: Pantalla de registro de nuevo usuario	37
Fig. 11/12 Pantalla de inicio de sesión de usuarios.....	38
Fig. 13/14 Pantalla inicial de usuario simple y usuario lugar	39
Fig. 15/16 Pantalla de edición de posición fija de lugar y	40
Fig. 17 Menú de navegación	41
Fig. 18/19 Pantalla de perfil propio.....	42
Fig. 20/21 Pantalla de lista de usuarios y pantalla de perfil de un usuario no amigo	43
Fig. 22/23 Pantalla de listado de Amigo, y pantalla de perfil de un amigo.....	44
Fig. 24 Pantalla de filtro de alertas para un amigo	45
Fig. 25/26 Pantalla principal con notificación y panel de notificaciones desplegado.....	46
Fig. 27 Pantalla de Estado Simple.....	47
Fig. 28/29 Nueva notificación en pantalla principal y vista del Estado Anuncio	48
Fig. 30/31 Pantalla de Perfil de usuario tipo lugar, y filtro de alerta para el usuario lugar amigo ...	49
Fig. 32 Pantalla principal con lista de notificaciones leídas	50
Fig. 33 Menú de creación de estados.....	51
Fig. 34/35 Pantalla de creación de estado simple y pantalla del estado simple creado	52
Fig. 36/37 Pantalla de creación de estado anuncio y pantalla del estado anuncio creado	53
Fig. 38/39 Pantalla de creación de estado Url y pantalla del estado Url creado	54
Fig. 40 Pantalla de perfil propio con el listado de estados creados.....	55
Fig. 41/42 Pantalla principal con notificaciones de transporte público y pantalla de perfil del usuario transporte público con lista de estados	56
Fig. 43 Diagrama de secuencias para el acceso al sistema o login.....	57
Fig. 44 Diagrama de secuencias para la configuración del filtro de alertas para un amigo.....	57
Fig. 45 Diagrama de secuencias para la creación de un estado Url	58

Capítulo 1 Introducción

En la actualidad, resulta evidente el advenimiento de los dispositivos móviles con avanzada capacidad de procesamiento, los cuales cobran importancia por hacerse cada vez más comunes entre los usuarios. Este cambio en el modo de acceso a la conectividad web ha generado una creciente en el volumen de información que es intercambiada entre personas a través de las aplicaciones denominadas “redes sociales”. Es sabido también que tales redes han comenzado hace algún tiempo a hacer uso de las nuevas tecnologías que brindan un dato adicional clave, la georreferenciación.

Dada la diversidad de productos existentes, y los variados enfoques con los que se los confecciona, hemos podido apreciar que los usuarios se ven muy a menudo, utilizando muchas de esas aplicaciones simultáneamente, con el fin de satisfacer todas las necesidades que pudiesen tener.

Habiendo observado y analizado la situación antes introducida, nos hallamos interesados en estudiar la factibilidad de la existencia de una red, capaz de abstraer el mayor conjunto de comportamientos, actividades, interacciones, estados y funcionalidad en general, valiéndonos de los conocimientos adquiridos en la carrera.

Definimos como objetivo principal del presente trabajo, diseñar y realizar una aplicación web, con cualidades comunes a las denominadas “redes sociales”, buscando antes que nada lograr que sea altamente adaptable a cualquier necesidad o uso, que abarque mediante la abstracción de elementos comunes la mayor parte de las prestaciones, y centralmente focalizada en la generación de información de interés a los usuarios, basándose en la tecnología de “georreferenciación”. Este último, es sin dudas, el eje central de la idea y del trabajo desarrollado, ya que toda la información que “PlaceOn” provee a sus usuarios, debe estar por definición, asociada a un punto en el tiempo y en el espacio; de manera que la “georreferenciación”, tanto sea en los aspectos tecnológicos como en su concepción física como cualidad de los eventos que suceden en el mundo, es un tópico sobre el cual nos extenderemos a lo largo de los análisis que el presente informe describe.

Para concretar el fin propuesto ha sido necesario un trabajo que abarcó diversas etapas tales como el relevamiento de productos existentes, su análisis desde lo técnico y desde las prestaciones que brindan; generación de requerimientos necesarios para la aplicación; evaluaciones arquitectónicas y tecnológicas para el desarrollo; establecimiento de pautas metodológicas para ejecutar en equipo el desarrollo del sistema; pruebas de uso y adaptabilidad del sistema obtenido; y también por último, la recopilación de la documentación formal de cada etapa, con el fin de poder confeccionar el presente informe que da cuenta de todo el proceso. Queda claro entonces, que el mencionado informe y el software diseñado, son ambos el resultado esperado y conseguido de este Trabajo Final de Grado.

Finalmente, podemos afirmar que, como esperábamos a priori, una vez concluido el proyecto en el que nos hemos embarcado, hemos logrado satisfactoriamente hacer uso de una pluralidad de

conocimientos adquiridos a lo largo de nuestro camino universitario, facilitando la toma de decisiones que se han presentado en cada una de las etapas mencionadas.

Capítulo 2 Marco teórico

Resulta bastante obvio para quienes estamos constantemente inmersos en el desarrollo web, pensar en un sistema informático, al escuchar las palabras “red social”, sin embargo, a pesar de que no nos solemos detener a pensar en ello, es sabido que dicho concepto de “red social” abarca un ámbito, una ciencia social, un estudio antropológico e histórico, los cuales exceden la preparación académica que un Ingeniero de Sistemas puede tener, o por lo menos; reconocemos con humildad, que nosotros tenemos. Gracias a las elecciones tomadas a la hora de recorrer nuestro camino a lo largo del ciclo de especialización, hemos podido sin embargo, adquirir conceptos que nos han permitido comprender los fenómenos sociales que derivan del análisis formal de las relaciones humanas. Analizando dichas relaciones mediante la construcción de grafos, con nodos y conexiones, evaluaciones y ponderaciones que permiten clasificarlos, aumentando su complejidad a través de superposiciones con distintos tipos de relaciones, es posible representar diversas vetas acotadas de la realidad, que facilitan su análisis (Wasserman & Faust, 1994) (Watts, 2006) (Cangrejo Aljure, 2011).

Volviendo a la primera idea a la que nuestra mente entrenada en las ciencias duras, en la programación, en las tecnologías web asocia el concepto de “red social”, seguramente estaremos de acuerdo en que hace ya varios años, la popularidad de las así denominadas comúnmente, ha crecido exponencialmente a lo largo de la Internet; y estando al tanto de diversos trabajos realizados que han abarcado su estudio y el de sus usuarios y su comportamiento, su análisis como fuente de datos para generar información a través del estudio de nodos, lazos, y el resultado de efectuar un proceso de minería de datos (Tang & Liu, 2010), etc., consideramos que resultaría redundante y repetitivo enunciar sus conceptos básicos generales. De modo que partiendo de la suposición de que el lector tiene una idea más o menos precisa acerca de lo que estamos referenciando mediante esa definición, es que brindaremos otros detalles teóricos, como así también el estado actual de varios sistemas existentes, que son los que definen el estado del arte en el ámbito en el que nos estamos adentrando. Aclaremos que de aquí en adelante, llamaremos “red social” o “red geosocial” al concepto que abarca a la aplicación web completa, a menos que explícitamente hagamos referencia al concepto social referido anteriormente.

Red geosocial

El primero concepto que se desprende del amplio espectro de las redes sociales, es el de red geosocial.

Una red geosocial es un tipo de red social que incluye funcionalidades relacionadas con la georreferenciación, tales como la geocodificación o la geoetiquetación. Ellas permiten a sus usuarios una dinámica social adicional a la que existe en otras redes sociales, como la interacción basada en el lugar donde se encuentran.

La georreferenciación se puede dar en las redes sociales gracias a localización de la dirección IP, la trilateración de un hotspot (zona de cobertura WiFi), la localización del teléfono móvil o incluso la información enviada por el propio usuario al respecto.

La historia de las redes geosociales es breve, dado que siendo una tecnología emergente todavía está penetrando en el mercado (Sui & Goodchild, 2011).

Su evolución comienza con la implicación social de las Interfaces de programación de aplicaciones por parte de las empresas de Internet a principios del año 2000. El desarrollo de aplicaciones basados en esta tecnología luego se verían altamente influenciados por la salida al mercado de los smartphones, con los cuales se abre la posibilidad de compartir información en tiempo real y en el exacto lugar donde ocurren los acontecimientos.

Con este tipo de aplicaciones, todo está basado en los lugares y la interacción con las personas. Descubrir cosas interesantes cerca de la locación de uno mismo, compartir lo que uno descubra o le interese con el resto de los usuarios. De manera que, como se ve, con este tipo de aplicaciones el foco suele estar tanto en lo social como en lo georreferencial. Por ejemplo compartir contenidos georreferenciados con nuestros amigos.

Realidad aumentada

Un concepto adicional en el ámbito en el que nos estamos adentrando, es el de “realidad aumentada”. Como primera definición podemos decir, que “realidad aumentada” refiere a una combinación entre lo virtual y lo real, que generan una apreciación del entorno con detalles agregados. Es decir, mediante la inclusión de elementos virtuales en tiempo real, el entorno físico del mundo real se ve “aumentado”. Esto se produce mediante el uso de dispositivos, por ejemplo un Smartphone, y aplicaciones que brinden información que sea pertinente (Cadavieco, Pascual Sevillano, & Madeira Ferreira, 2012).

Vemos entonces, la clara diferencia con la denominada “realidad virtual”, que lo que intenta es suplantar e imitar al mundo real.

Lo esperado es que a través de la mencionada “realidad aumentada”, se obtenga una nueva imagen de nuestro entorno, mucho más enriquecida en información, que propicia una mejor percepción y mayor entendimiento del espacio en el que nos encontramos.

Aunque la Realidad Aumentada está considerada como una tecnología revolucionaria y muy novedosa, en realidad su origen data de la década de los 90, pero no es hasta el año 2009, y gracias a la implantación de los smartphones, cuando su uso se ve verdaderamente potenciado hasta convertirse en una tecnología accesible al público en general.

Este concepto es tomado por nosotros como clave a la hora de definir un requerimiento en nuestra aplicación. La idea clave que se desprende de esta definición es que es un requerimiento, que la aplicación brinde información en tiempo real al usuario a medida que vaya recorriendo ciertos puntos de la ciudad los cuales haya marcado como de interés.

Estado actual de redes populares

Resulta ser un primer paso intuitivo, buscar y analizar experiencias previas con redes sociales existentes. De hecho lo ha sido a tal punto que a la hora de observar con detenimiento estas

redes, ni siquiera estábamos del todo seguros que tales apreciaciones serían con el paso del tiempo, parte de un desarrollo que pudiera convertirse en un proyecto de Trabajo Final.

Las primeras discusiones y charlas al respecto rondaban buena parte del tiempo, en comparaciones sobre los productos y sistemas de los que éramos -y aún somos- usuarios. Frases que resultaban ser cotidianas, poseían la siguiente estructura: “pude ver la red A, capaz de hacer X e Y, la red B que permite al usuario una experiencia con Z, e implementa X; o a la red C que mejora Y incorpora Y’ pero no es buena en X”. Estas primeras charlas, nos condujeron a observar que era posible tomar cuestiones y usos comunes de cada ejemplo visto, con el fin de crear un sistema adaptable y genérico.

Por otro lado, se dieron otras dos circunstancias que juntas llevaron a establecer el segundo aspecto clave del sistema. La primera de ellas, resultó del mismo análisis descripto. Pudimos ver la tendencia de vincular más estrechamente los eventos que los usuarios experimentaban en la vida real -información compartida, suceso vivido, acontecimiento presenciado, tarea realizada, etc.- a los estados que los representan en la red, lo cual hacía cada vez más frecuente y necesaria la inclusión de los atributos espaciotemporales que los describiesen. La segunda, relativa al aspecto técnico, fue la posibilidad que casi por fortuna tuvimos ambos, de trabajar paralelamente en proyectos laborales que nos permitieron conocer herramientas capaces de brindar información geográfica desde cualquier dispositivo con acceso a Internet, y usar dichos datos para crear aplicaciones interactivas usando mapas. Como se puede suponer, haber visto lo importante desde lo social y lo factible desde lo tecnológico que era resolver los problemas que pudimos ver, nos llevó a decidírnos a poner manos a la obra en el proyecto.

Daremos cuenta a continuación, de las redes sociales más relevantes que hemos estudiado con el fin de avanzar en el desarrollo de nuestra aplicación. Cabe destacar que haremos especial hincapié en las que consideramos efectivamente como “redes sociales georreferenciales”, tal vez dejando de lado a algunos “híbridos” (redes sociales que si bien usan algún tipo de dato de locación, no se focalizan en tal aspecto) ya que se tratan de las que han atrapado nuestra mayor atención por lo explicado hasta aquí.

Google Latitude

Tal vez sea la aplicación que más ha influenciado el desarrollo de los requerimientos que hemos definido. Cabe destacar que su desarrollo y soporte, ha sido discontinuado, y no se encuentra disponible desde el pasado Agosto de 2013.

Las principales características de esta aplicación estaban siempre centradas en la poderosa herramienta de Google Maps. Estos mapas en línea, además de poseer un buen desempeño, y más que aceptable fidelidad (precisión en las mediciones y ubicaciones); proveen una interfaz de programación muy completa y bien documentada, disponible para los desarrolladores que necesiten hacer uso de la misma.

Por otro lado, Google Latitude permitía a sus usuarios, compartir sus posiciones actuales, con el fin de facilitar encuentros entre amigos. Mediante actualizaciones de estado, era posible

comunicarse entre los amigos, aunque también permitía el envío correos electrónicos, y de llamadas audio y video.

Una acertada mirada que también hemos rescatado, fue la decisión de apuntar la compatibilidad del servicio con los dispositivos móviles, lo cual es una natural evolución dada la particular importancia que cobra poder desplazarse con la aplicación en funcionamiento.

Google+

Si bien en su concepción, Google+ no es una red puramente georreferencial, no dejamos de mencionarla, ya que son sus propios creadores quienes han recomendado a sus usuarios, a utilizar sus prestaciones para conocer la ubicación de sus amigos, en el boletín de anuncio de cierre de Latitude.

De esta aplicación, no hemos podido sustraer innovaciones significativas, aunque claramente hay puntos en común desde lo tecnológico, con su predecesor.

Glympse

Herramienta que también apunta a favorecer los encuentros entre amigos. Apuntada a dispositivos móviles, en muchos casos es destacada con el mote de “verdadero sucesor de Latitude” debido al parecido de las prestaciones brindadas.

Los puntos que pueden catalogarse como innovadores, apuntan al modo de compartir la ubicación, que intenta generar una sensación de control de privacidad superior. Incluye un “reloj de Glympse” que permite determinar durante cuánto tiempo queremos que nuestra posición sea compartida. A nuestro juicio, si bien no vimos en la solución que esta aplicación brinda, una funcionalidad del todo apropiada, es destacable el hecho de haber traído a la luz el problema de privacidad que supone el uso y transmisión de un dato tan sensible como la posición en la que una persona se halla.

I'm Here

Si bien su uso se ha estado multiplicando a lo largo de los usuarios de dispositivos móviles, está completamente centrada en la actividad de ocio, y la comunicación entre amigos.

Entre sus características principales se encuentra la de crear puntos de reunión virtuales. Permite a los usuarios compartir su posición en tiempo real. Los usuarios se relacionan por medio de vínculos de amistad y posee mensajería instantánea para comunicarse.

En cuanto a la privacidad todos los datos se borran automáticamente cuando la aplicación se cierra.

Específicamente en lo referido a prestaciones innovadoras o características especiales, no ha logrado captar nuestra atención.

En lo técnico, especialmente en los estilos visuales, provee una interfaz amigable y simple, ideal para ser utilizada desde un teléfono.

Life360 – Family Locator

Apunta como bien lo indica su nombre, a facilitar y brindar constantemente, información de ubicación con nuestra familia. Queda limitada por esa característica, ya que utiliza una jerarquía entre el grupo de usuarios, que busca simular una estructura familiar. Así, un “padre” puede siempre saber dónde está ubicado su “hijo menor”, por citar un ejemplo.

Adicionalmente, brinda el servicio de “botón de pánico” que emite avisos ante su activación.

Family by Sygic

Comparte la esencia de su concepción con la red anterior, sin brindar algo puntualmente innovador en esa comparación.

Foursquare

Esta afamada aplicación, provee el servicio típicamente llamado “check-in”, es decir, permite decir mediante un registro de actividad, dónde hemos ido y qué nos pareció la experiencia vivida allí. Así, generalmente, su uso refiere a relevamientos, evaluaciones, puntuaciones y opiniones, de lugares comerciales, como puede ser un restaurante, un museo, es decir un punto turístico en general.

Un usuario comienza a ganar reputación mediante el uso frecuente de la aplicación, y así sus opiniones progresan en su valor. De esa manera, y usando motores de análisis de los datos, la red social ha evolucionado hacia la forma de “motor de recomendaciones”.

Como conclusión del análisis que fuimos realizando, quedaron pautas claras que se convirtieron en requerimientos formales a la hora del desarrollo propiamente dicho. Las bases sentadas definieron las siguientes cualidades que consideramos necesarias:

- Toda la información generada por un usuario debe ser georreferenciada.
- El tipo de información que se comparte, debe ser adaptable a las necesidades.
- Proveer un mecanismo para ser notificado de las novedades, que también se base en la georreferenciación.

Estado “local” de aplicaciones georreferenciales

Haciendo análisis del actual uso que se le está dando a las capacidades georreferenciales en un ámbito nacional, pudimos notar que hoy en día, existen diversas aplicaciones con funciones demasiado acotadas, que las convierten en cuasi similares, faltas de abstracciones que permitan un uso más amplio, lo que termina generando que los usuarios deban tener instaladas cada una en su celular para poder satisfacer las necesidades.

Para tener una mejor idea de lo que estamos expresando, daremos ejemplos de algunas de ellas.

El Ministerio del Interior y Transporte presentó la aplicación “Trenes en Vivo”, según el organismo brinda información precisa, actualizada y en tiempo real, para que los usuarios de la Línea

Sarmiento conozcan los próximos arribos de las formaciones, en las distintas estaciones del recorrido.

El gobierno de la Ciudad de Buenos Aires lanzó dos aplicaciones para smartphones que proveen de información y guías para trasladarse a través de la ciudad en distintos medios de transporte así como también conocer el estado del tránsito.

“Ba Móvil” provee estado del tránsito vehicular en tiempo real ya que proporciona información a partir de la Geolocalización del usuario, incluidos cortes por manifestaciones o accidentes.

“¿Cómo Llego?” proporciona información sobre la mejor forma de realizar un viaje utilizando los distintos medios de transportes disponibles en la ciudad: subte, colectivo, tren, auto o a pie.

Ahora, pensemos que si un usuario necesitara saber los horarios de los trenes junto con el estado del tránsito para decidir cómo trasladarse. Nos surge naturalmente la pregunta ¿es realmente necesario utilizar dos aplicaciones diferentes? Una hipótesis válida es pensar que al estar tan relacionadas, podrían coexistir en una sola aplicación más completa.

De hecho, consideramos que la fusión de ellas aumentaría la probabilidad de éxito de la misma, pudiendo llegar a un amplio espectro de usuarios que serían atraídos por cada tipo de uso de la aplicación. También, esto permitiría crear una base de datos de usuarios de más envergadura en lugar de tener varias más pequeñas. Se podría hacer uso de esta mayor cantidad de información para hacer estudios más amplios sobre los usuarios, acciones y preferencias, en incluso sobre logística de distribución de tránsito en una ciudad.

En general, cada aplicación diferente que lograra ser abstraída por conceptos que las lograra unir en una sola, aprovecharía la popularidad que proveería la plataforma integrada debido a los usuarios de cada una de ellas por separado.

Concluimos expresando, que por las situaciones observadas, hemos pensado en proveer una solución lo suficientemente abstracta que pueda englobar distintas aplicaciones basadas en Geolocalización y preferencias del usuario; y habiendo extraído esos valiosos lineamientos, procedimos a dar los primeros pasos en el desarrollo, siendo esa etapa descripta y analizada en los siguientes capítulos.

Capítulo 3 Análisis funcional y de Requerimientos de “PlaceOn”

Con el fin de tener una referencia clara sobre los requerimientos con los que luego lleváramos a cabo el desarrollo, los mismos fueron documentados y explicitados. Esa información además de haber guiado el proceso que nosotros hemos seguido, al ser puesta al alcance del lector, brinda un claro panorama general del funcionamiento del resultado obtenido.

A continuación, se detallan los mencionados requerimientos.

Requerimientos funcionales

- El sistema debe proporcionar el manejo y registro de usuarios, es decir, cada persona que desee hacer uso de la aplicación, debe poder crear un perfil de usuario.
- El sistema debe proporcionar autenticación de usuarios, o sea que un usuario deberá proporcionar información de nombre y contraseña para poder ser identificado, iniciar su sesión y así hacer uso de la aplicación.
- El sistema debe ser capaz de detectar la posición geográfica exacta del usuario que ha dado inicio a su sesión.
- El sistema debe permitir a un usuario, buscar otros usuarios con el fin de establecer el canal de comunicación entre ellos.
- El sistema debe permitir a los usuarios, crear información georreferenciada para ser emitida al resto de los usuarios interesados.
- El sistema debe soportar un mecanismo que defina la información entrante para un usuario, basado en la georreferenciación.
- El sistema debe mostrar la información entrante al usuario receptor, en forma de alerta visual y georreferenciada.

Requerimientos no funcionales

- El sistema debe ser una aplicación web, es decir debe ser accesible desde un navegador sin necesidad de instalación de ningún otro software en los clientes.
- Se busca apuntar a la compatibilidad del producto, que es un atributo de calidad fundamental para poder abarcar mayor número de usuarios.
- El sistema debe proveer una interfaz de usuario capaz de ser utilizada en dispositivos móviles.
- En este caso también creímos vital, poder brindar una experiencia de usabilidad adaptada a las pantallas reducidas de un teléfono.
- El sistema debe ser fácilmente adaptable en lo referido a la estructura que representa la “información compartida” con el fin de poder soportar diversos tipos de datos – multimedia, enlaces web, documentos, evaluaciones, recomendaciones-.

Por último, el resultado buscado apunta a poder redefinir, agregar o quitar de modo sencillo, definiciones de “Objetos compartidos” para poder adaptar así el sistema a las necesidades puntuales con el fin de garantizar la realización del concepto inicial de abstraer comportamientos.

Por lo demás, no fue sino hasta el momento de estar ya desarrollando el sistema, que fuimos tomando decisiones tan importantes como las que aquí quedaron definidas. Tecnologías utilizadas, decisiones arquitectónicas y funcionalidad extra añadida, fueron fruto del uso y prueba de los resultados parciales que fuimos obteniendo. Tales cuestiones son abordadas en otros apartados.

Capítulo 4 Diseño arquitectónico de “PlaceOn”

A la hora de comenzar con el diseño de la aplicación, se tomaron varias decisiones a nivel arquitectónico que dieron forma al resultado final.

Primer análisis arquitectónico

Antes que nada, buscando satisfacer de forma primordial la accesibilidad web hacia el sistema, por parte de los más diversos terminales, consideramos que el estilo de “cliente-servidor” era casi mandatorio. Es obviamente un estilo usado en una enorme mayoría de casos, con éxito y sustento empírico, además de que nosotros contamos con experiencia en su uso.

Algunas de las ventajas que obtenemos por tomar este estilo como parte de nuestra solución son:

- Capacidad de atender desde el servidor, a múltiples clientes comunicándose de manera concurrente
- Capacidad de servicio escalable, al poder añadir potencia o incluso cantidad de servidores atendiendo a los clientes
- Modificabilidad alta, al centralizar el código en el que corre la aplicación, en los servidores mismos

Refinamiento arquitectónico

Para lograr un diseño arquitectónico que pudiese satisfacer la calidad de producto esperada, hicimos uso del método ADD (Attribute-Driven Design). De hecho, con dicho método, hemos podido generar una arquitectura de software, y luego un sistema, que cumple satisfactoriamente con lo estipulado.

El método como ya sabemos, propone recursivamente, descomponer el sistema tratando de asegurar en cada iteración, los atributos de calidad de mayor prioridad entre los que aún no fueron atendidos. Al final, a través de la elección de tácticas y patrones, todas las cualidades deberán haber sido satisfechas.

De este modo, comenzamos evaluando los requerimientos de calidad, requerimientos funcionales, descriptos antes. También se tuvieron en cuenta algunas restricciones de diseño que nos impusimos. El análisis necesario generó un listado priorizado de los mismo, necesarios como entrada para comenzar el proceso recursivo.

Entre las restricciones mencionadas, elegimos como lenguaje de programación PHP versión 5 debido a que contamos con experiencia suficiente para poder sacar provecho del mismo. Adicionalmente, dado que hemos optado por hacer uso de una API de Georreferencia y localización provista por HTML5, entonces es necesario que dicha tecnología sea compatible con las decisiones que se tomen.

Comenzando con el proceso de priorización propiamente dicho, consideramos que el atributo más importante a tener en cuenta, es el de portabilidad. Es prioridad primera de PlaceOn, poder ser accedido desde la mayor parte de los dispositivos disponibles hoy en día. La forma de expresar

estos atributos y lo que se necesita para su cumplimiento, es definiendo seis factores para el denominado escenario de calidad. Para el requerimiento mencionado como prioridad, el escenario de calidad será:

<u>Fuente</u>	Usuario final
<u>Estímulo</u>	Desea acceder al sistema desde cualquier dispositivo
<u>Artefacto</u>	Sistema
<u>Entorno</u>	Runtime
<u>Respuesta</u>	El sistema es completamente accesible
<u>Medida de respuesta</u>	Acceso 100% funcional desde cualquier dispositivo que cuente con un browser compatible con html5 y API de Geolocalización

Acordamos tomar como segundo en el orden de prioridades, lo referido a la usabilidad. Buscamos apuntar a una experiencia de usuario adaptada a los dispositivos móviles, los cuales poseen espacio de pantalla acotado, y generalmente sensores táctiles. Entonces el escenario que deseamos satisfacer es:

<u>Fuente</u>	Usuario final
<u>Estímulo</u>	Desea utilizar eficientemente el sistema desde un dispositivo móvil
<u>Artefacto</u>	Interfaz de usuario del sistema
<u>Entorno</u>	Runtime
<u>Respuesta</u>	El sistema provee una interfaz cómoda para un dispositivo móvil
<u>Medida de respuesta</u>	La información del sistema debe ser completamente visible y accesible en pantallas de 4" o más

Como tercer atributo de calidad deseable, añadimos la necesidad de poseer un sistema modificable, especialmente en el punto clave de lo que ahora denominaremos "información compartida", que no es más que la unidad de información que deseamos que la red PlaceOn, permita hacer circular entre sus usuarios. Como objetivo de este requerimiento, se halla el deseo de poder fácilmente, añadir o modificar tipos de información. Sólo por citar ejemplos, actualmente nuestro sistema permite compartir textos, enlaces con vista previa e imágenes, pero bien podríamos desear en alguna otra instancia, proveer soporte para videos, mensajes con múltiples destinos, etc.

Definimos nuestro escenario:

<u>Fuente</u>	Desarrollador
<u>Estímulo</u>	Desea añadir una nueva clase de "información compartida"
<u>Artefacto</u>	Sistema
<u>Entorno</u>	Tiempo de desarrollo
<u>Respuesta</u>	Se realiza el cambio sin afectar otras funcionalidades
<u>Medida de respuesta</u>	Se consigue el resultado final luego de 1 día de trabajo

Estos tres requerimientos de calidad, son según el criterio propio, nuestros tres Architectural Drivers. Esto es, del conjunto entero de requerimientos funcionales y de calidad, observamos en estos la posibilidad más alta de tener impacto en el diseño de la arquitectura.

Entonces nos hallamos ante la primera decisión arquitectónica a ser tomada. A priori, es necesario hacer una subdivisión de lo que podemos llamar "Sistema", que no es más que la idea menos detallada de una arquitectura.

El primer requerimiento en el orden de prioridades, indica que deseamos que la aplicación sea accedida desde múltiples dispositivos. Es bastante lógico pensar que una aplicación web, que necesita disponer de datos centralizados, información compartida entre usuarios, y que deseamos se comporte como una aplicación web, nos lleve a decidir como primer medida arquitectónica, que será necesario poseer componentes del tipo Cliente y Servidor.

Las razones y los razonamientos que nos hacen optar por este tipo de arquitectura, están basados sobre todo en la experiencia tanto personal, como ajena, ya que está comprobada la efectividad de sistemas de este tipo, a lo largo del tiempo y el espacio, especialmente en aplicaciones que son accedidas a través de la Internet.

Lo que esperamos entonces a través de esta estructura, es poder garantizar el acceso al Servidor, desde múltiples clientes. Cabe destacar aquí, que también al apegarnos a las restricciones impuestas, también dejamos sentado el hecho de utilizar protocolos HTTP/HTTPS como esquema de comunicación, favoreciendo así el acceso al cliente masivamente, necesitando para ello solamente de un navegador o browser compatible con HTML5 (prácticamente cualquier navegador conocido en sus últimas versiones).

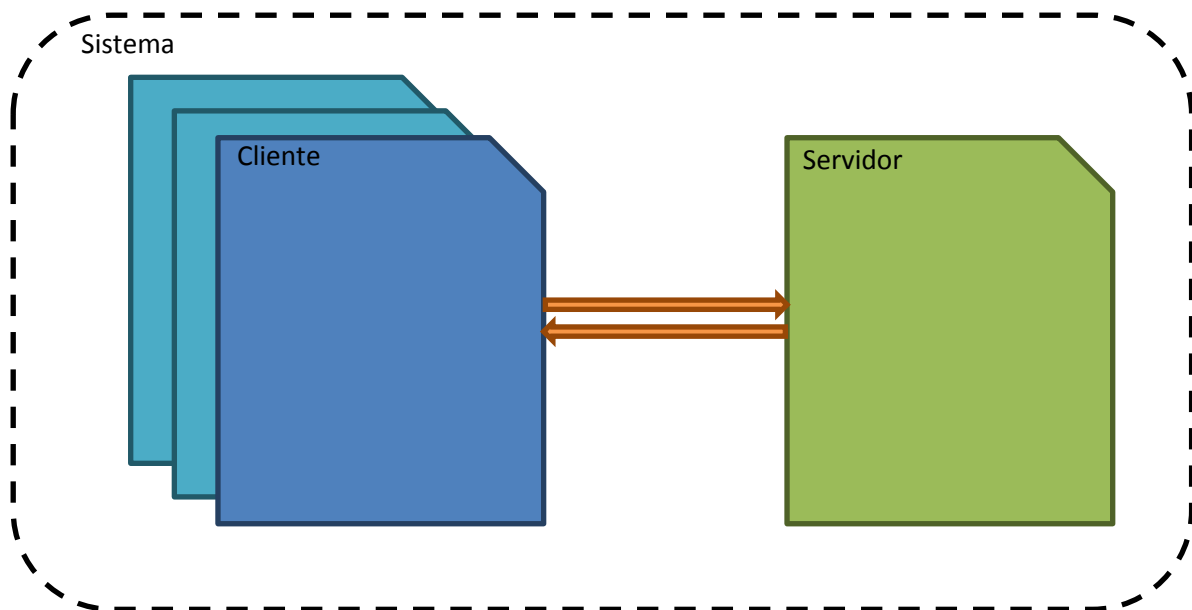


Fig. 1 Descomposición en Cliente-Servidor

Los mecanismos e interfaces de comunicación entre estos dos componentes, como ya hemos mencionado, se apegan al protocolo HTTP/HTTPS con el fin de garantizar la compatibilidad con navegadores web, siendo ese un punto clave en nuestro sistema.

En este punto, consideramos cubierto el primer Architectural Driver y de esta manera, pudimos concluir una primera descomposición que resultó en una arquitectura general establecida (Fig. 1), nos hallamos en condiciones aptas para seguir adelante en el refinamiento y descomposición subsiguiente.

Buscando alcanzar el objetivo de brindar una experiencia de usuario adaptable a los diversos potenciales Clientes (haciendo referencia al dispositivo físico propiamente dicho, desde el cual se accede al sistema, volviéndolo una instancia del artefacto arquitectónico "Cliente") hemos analizado un par de alternativas de las que daremos cuenta a continuación.

Aquí las dos alternativas que consideramos, eran por un lado, el diseño de aplicaciones Cliente específicas para cada plataforma que deseábamos alcanzar, por el otro una estructura arquitectónica genérica que permitiera un desacople entre la lógica que se utiliza para alcanzar los requerimientos funcionales, y la forma en que esa información es presentada a los usuarios. En otro apartado del presente informe, haremos una comparación más detallada entre las denominadas App Nativas y las Web Apps. Más allá de las cuestiones que han hecho inclinar la balanza para uno de los lados, es importante no perder de vista un par de consideraciones. Primero, la restricción impuesta del lenguaje PHP, que vuelve complejo sino imposible, el hecho de programar aplicaciones nativas (que suelen estar atadas a un lenguaje específico según en que dispositivo o sistema operativo se busque ejecutar la aplicación), y segundo, la decisión tomada en el paso anterior, de apuntar a generar un cliente capaz de correr sobre navegadores.

Con estos hechos a la vista, llegamos a concluir que será beneficioso el empleo de una arquitectura Model-View-Controller (MVC), que nos facilite el desarrollo por separado de los componentes lógicos pertinentes al modelo de negocio, y de la presentación de la información a los usuarios (Fig. 2). También en un apartado posterior, se dará cuenta de un framework MVC hecho en PHP, elegido para el desarrollo del sistema.

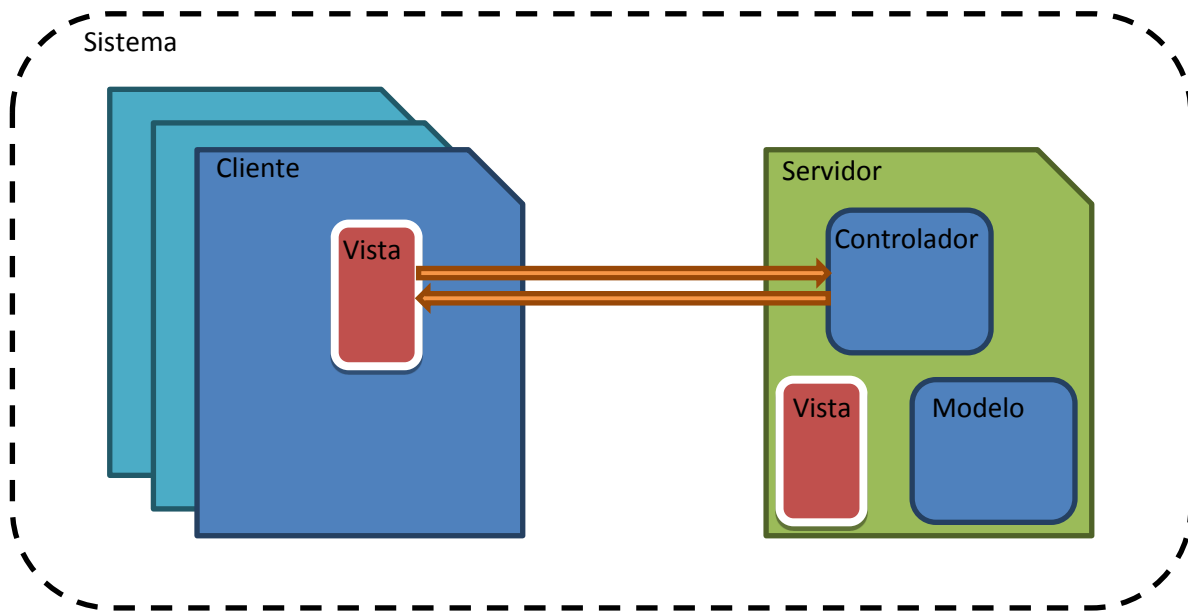


Fig. 2 Descomposición en MVC

Aquí cabe destacar los mecanismos de comunicación entre los componentes. Dentro del artefacto Modelo, tenemos que reflejar todas responsabilidades pertinentes a la lógica de negocio del sistema, control de integridad de datos, mapeo hacia la base de datos, y obtención y guardado de información. El Controlador claro, el quien en efecto deberá recibir, responder y atender las peticiones propiamente dichas, que llegarán a través de las interfaces dispuestas por el protocolo HTTP, desde los clientes. También se mantiene desde el Controlador, comunicación bidireccional con el Modelo, ya que enviará los pedidos tanto de guardado como de búsqueda de datos.

La denominada Vista, como se ve, se encuentra dividida tanto en el Cliente como en el Servidor. Esto es debido a que parte de la información que será mostrada a los clientes, será enviada de regreso ante las peticiones HTTP realizadas. Es decir, el Servidor generará buena parte de las vistas, y las retornará para ser mostradas por los navegadores webs que corren en los clientes. Por otro lado, considerando como parte de la Vista, el código Javascript y HTML que están en el navegador, es que dividimos dicho componente de este modo. El cliente, al interactuar con el navegador, y ejecutar peticiones a través del mismo, vía AJAX y usando el mismo protocolo HTTP, se comunica con el Controlador que estará atento a dichos mensajes entrantes.

Luego de haber cubierto el segundo Architectural Driver, con el resultado arquitectónico mostrado, procedimos al análisis del considerado como tercer y último requerimiento.

Con el fin de favorecer la facilidad a la hora de modificar lo que abstractamente denominamos "información compartida", es que decidimos valernos de varias tácticas que son pertinentes en este caso.

Ya aquí estamos un paso adelante. Sabemos que éste, es un punto crítico en el cual es muy probable que sean necesarias modificaciones, agregados, etc. Es una de las tácticas más valiosas

para prevenir efectos en cadena (cambios que conllevan otros, y otros, aún en sitios "lejanos" arquitectónicamente hablando).

En ese mismo sentido, separar la pequeña sección de modelo que refleja dicha información en un módulo abstracto más genérico, supone también que los puntos de conflicto se hallen localizados, es decir se espera así poder acotar los lugares que deban ser cambiados ante la eventual necesidad.

Estas tácticas que esperan favorecer en buena medida la modificabilidad del sistema en el punto clave, nos hacen pensar en un módulo que se desprenda del modelo general (Fig. 3).

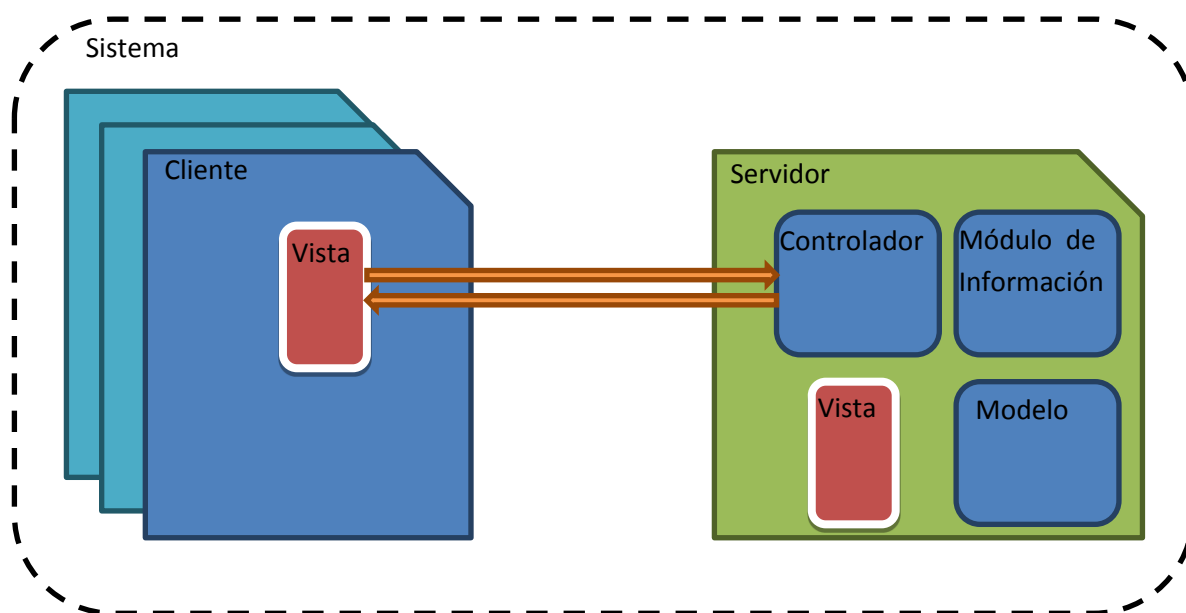


Fig. 3 Descomposición en el Módulo de Información

En cuanto al modo de interacción con el nuevo módulo, deseamos proveer similares interfaces a las que existen entre Controlador y Modelo que hemos mencionado antes.

Habiendo provisto al sistema de características que solucionan y resuelven las necesidades planteadas como prioridad, definidas como Architectural Drivers, hemos podido iterativamente, definir una arquitectura que se ajusta a la medida.

Todo este proceso que hemos atravesado, nos permitió no solamente obtener la arquitectura previa al desarrollo, sino que nos ayudó enormemente para poder observar los problemas, las diversas soluciones alternativas, como así también constituye una experiencia invaluable personal y profesional para nosotros. No es del todo cotidiano, a pesar de ser ambos desarrolladores avanzados, tener esta posibilidad de tener injerencia directa en las decisiones arquitectónicas de los proyectos en los cuales nos hemos involucrado en nuestra carrera profesional, de manera que esa es sino la más rica de las experiencias obtenidas del presente trabajo.

MVC Características generales

A grandes rasgos, este patrón realiza y define una separación de los datos, la interfaz de usuario, y la denominada “lógica del negocio” en tres componentes distintos. Como mencionamos se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el sistema de Base de Datos y la “lógica de negocio”, y el controlador es el componente de software responsable de recibir los eventos de entrada desde la vista.

El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la “vista” aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al “modelo” a través del “controlador”.

El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al “modelo” cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su “vista” asociada si se solicita un cambio en la forma en que se presenta de 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el “controlador” hace de intermediario entre la “vista” y el “modelo”.

La Vista: Presenta el “modelo” (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho “modelo” la información que debe representar como salida.

Elección de Framework

Con las necesidades y decisiones tomadas hasta este punto, y con la experiencia en el desarrollo de sistemas web que poseemos, fue necesario llevar a cabo la elección de las tecnologías con nombre propio. Entonces, fue necesario elegir el camino que sería seguido para adaptar nuestra aplicación a los diversos dispositivos móviles. Se llevó a cabo una evaluación entre las denominadas “App nativas” y las “Web apps”, como primera instancia. También se necesitó elegir un framework que se adaptase al estilo MVC, con un servidor web, con una estructura para definir vistas adaptables y que sean eficaces a la hora de ser usadas a través de Internet. Aquí definitivamente tuvo un peso determinante, el conocimiento previo con el que contamos. El lenguaje elegido para el desarrollo de la aplicación del servidor fue PHP, potenciándolo con el uso del “Yii framework”, un potente y adaptable framework MVC que nos provee de la estructura básica para partir y avanzar a buen ritmo (Yii Software LLC, 2012).

Brindaremos a continuación, detalles sobre estas nuevas resoluciones adoptadas.

¿“App nativa” o “Web app”?

Otra de las decisiones tempranas que debimos tomar en el desarrollo de la aplicación fue elegir si debíamos enfocarnos en crear una Web App o una App Nativa.

La diferencia entre ambas es importante, y brindaremos los detalles más destacados de cada enfoque, y las pautas que tuvimos en cuenta para la elección final (Lionbridge, 2012).

Web app

Una web app es una aplicación web optimizada mediante HTML5, CSS3 y Javascript para la correcta visualización en los smartphones. Aunque no es una app propiamente dicha, pues no se instala en el dispositivo, consigue tener una apariencia cercana a una aplicación nativa.

Uno de los puntos fuertes -ciertamente, el más- de las aplicaciones web móviles es su funcionamiento en todas las plataformas (solo se debe tener en cuenta la compatibilidad con el motor de browser o navegador). Una aplicación web funcionará en casi todos los smartphones, de manera que es intrínsecamente “multiplataforma”, ya que con mínimos ajustes pueden funcionar en móviles y tabletas, ya sean iOS, Android, Windows Phone, BlackBerry etc.

Por otro lado, es destacable otra de las grandes ventajas de las aplicaciones web, y es que requieren una menor inversión inicial, debido a que la mayor parte del desarrollo no se debe repetir para cada sistema operativo apuntado. Esto acarrea otra ventaja clara, el menor tiempo de desarrollo que sus equivalentes nativas, lo que puede ser crucial en el caso de que exista una ventana de oportunidad para la venta de la aplicación, si bien no es nuestro caso.

Como ya explicamos, las Web Apps o Aplicaciones Web se ejecutan siempre mediante un navegador web. Esto quiere decir que utilizan HTML, CSS y JavaScript, las mismas tecnologías que cualquier página web, salvo que hacen uso de las últimas especificaciones disponibles para dispositivos móviles. Al contrario de lo que pudiera parecer, estas aplicaciones pueden asemejarse visualmente mucho a las aplicaciones nativas, hasta el punto de ocultar cualquier aspecto de la interfaz del navegador en que se ejecutan. Incluso pueden instalarse con su propio ícono junto al resto de apps y guardar datos localmente para funcionar incluso sin conexión a Internet.

Además, al estar alojadas en un servidor centralizado, se garantiza que todos los dispositivos cuentan con la versión más actualizada de la app, lo que redundará en menores problemas de soporte al cliente final.

App Nativa

Una app nativa es una aplicación implementada en el lenguaje nativo de cada terminal. Estas apps podrán acceder a los sensores internos del móvil para aprovecharse de funcionalidades típicas de estos dispositivos.

Como hemos destacado, buscar un alcance multiplataforma sin requerir dividir el desarrollo en varios frentes, fue lo que nos hizo decidir entre ambos esquemas. Como supondrá el lector, en el caso de estas app nativas, se debe tener en cuenta que no todos los smartphones funcionan bajo la misma plataforma. En el caso de optar por desarrollar aplicaciones nativas, se debería hacer un

desarrollo distinto para cada sistema operativo, lo que supone un incremento del coste de desarrollo de la aplicación, división en equipos especializados en diversas tecnologías, aumento del tiempo invertido necesario.

Claro que hay una gran cantidad de ventajas. Uno muy importante es el acceso directo a mayor funcionalidad del dispositivo, y la performance del mismo. Programando directamente con las tecnologías propias del aparato en cuestión, se puede hacer uso de los accesorios integrados dentro del propio teléfono como el geoposicionamiento, brújula, acelerómetro, cámara, etc. Esto las hace también más rápidas en tareas tales como animaciones o juegos, dándoles la posibilidad de tener una experiencia de usuario más sofisticada. Una app nativa, tiene muchas más chances de brindar a los usuarios una experiencia que iguala al uso del teléfono (o el dispositivo que fuese) en cuanto a fluidez, debido a que están escritas en el propio lenguaje de la plataforma, cuentan con todos los widgets y estilos pre programados.

Esta diferencia a favor del desarrollo nativo, no obstante, tiende a reducirse con cada nueva versión de los sistemas operativos y navegadores.

Respecto al modelo de publicidad y marketing, a diferencia de las aplicaciones web, aquí entran en juego los market places, los cuales se han convertido en la perfecta plataforma de promoción para llegar a millones de usuarios. Nadie duda ya del poder de sitios como el App Store de Apple o el Google Play de Android. Evidentemente sólo las apps nativas se benefician de las ventajas que proporcionan estos mencionados market places, ya que son repositorios de aplicaciones instalables en los smartphones, y como hemos dicho antes, las web apps son básicamente páginas web que no requieren instalación.

Sin embargo debemos recordar que para acceder a cada uno de los market places, existe un proceso de validación, que puede volverse muy engorroso. Se busca que todas las aplicaciones disponibles para los usuarios de los market places, cumplan todos los requisitos especificados en las guías para desarrolladores específicas. Este proceso se repite con cualquier actualización de la app que se quiera publicar.

Como conclusión de esta comparación, hemos tomado la decisión de profundizar el análisis del desarrollo de una web app, ya que provee grandes ventajas en cuanto a tiempos, complejidad, y el alcance multiplataforma. Dado que esta solución presenta varias desventajas relacionadas al acceso de características integradas de los dispositivos (principalmente el GPS), y a la experiencia de usuario, es que hemos adoptado en conjunto, varias herramientas y tecnologías que allanan el camino en este sentido, las cuales son de nuestro conocimiento y serán presentadas en otras secciones del presente capítulo.

Yii Framework

Yii es un framework MVC para PHP de alto rendimiento basado en componentes web para desarrollar aplicaciones de gran escala. Permite una máxima reusabilidad en la programación web y puede acelerar significativamente el proceso de desarrollo.

Yii es un framework genérico para programar Webs que puede ser utilizado para desarrollar virtualmente cualquier tipo de aplicaciones web. Ya que es liviano y está equipado con las soluciones más sofisticadas, está especialmente diseñado para trabajar con aplicaciones web de tráfico alto, como portales, foros, CMS, comercios electrónicos, etc.

Yii sobresale sobre los otros frameworks PHP por su eficiencia y su rica librería de funcionalidades así como también su clara documentación, puesto que es el resultado de la experiencia que tienen sus autores en el desarrollo de aplicaciones web ricas en funcionalidad y la investigación y reflejo de los frameworks y aplicaciones más populares para programar Webs.

Para crear aplicaciones utilizando Yii, debe seguir un simple proceso compuesto de 3 pasos:

- Crear la base de datos
- Generar el código PHP de la base, utilizando Yii
- Modificar el código para adecuarlo a sus necesidades

Ejemplo de funcionamiento del framework Yii (Fig. 4):

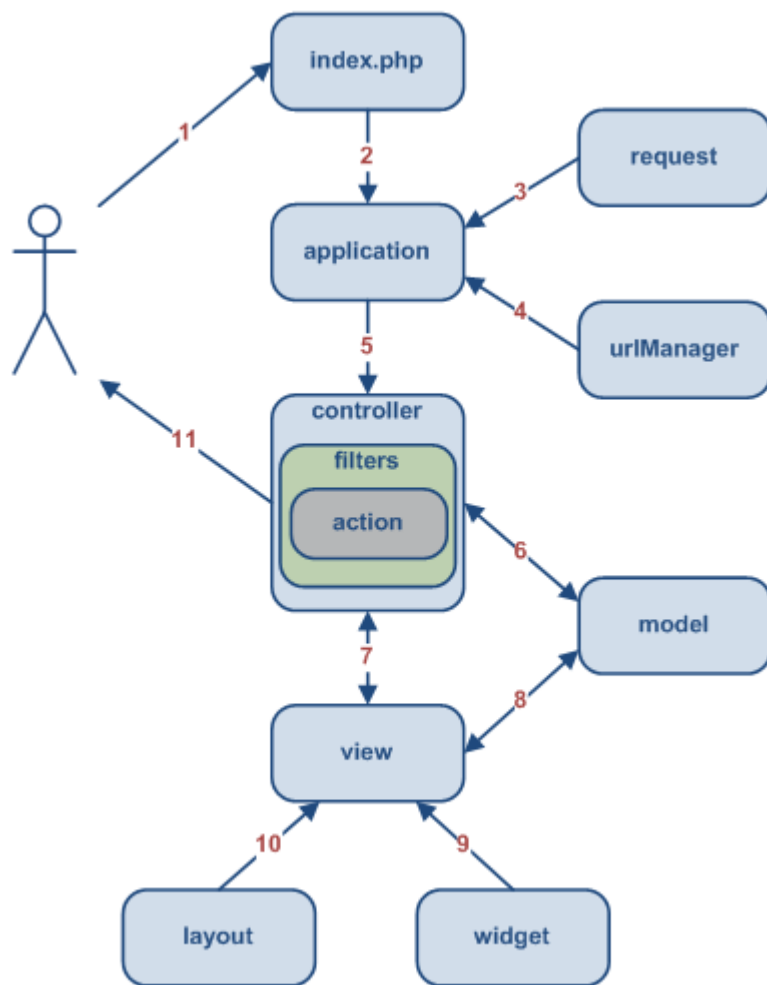


Fig. 4 Yii en funcionamiento

1. Un usuario hace una petición en la siguiente URL: <http://www.example.com/index.php?r=post/show&id=1> Y el servidor web maneja la petición ejecutando el script index.php.
2. El script crea una instancia de la aplicación y la ejecuta.
3. La aplicación obtiene la información detallada de la petición de los usuarios desde un componente de la aplicación llamado request (petición).
4. La aplicación determina la petición del controlador y actúa con la ayuda de un componente de la aplicación llamado urlManager (manejador de url's). Para este ejemplo, el controlador es Post que hace referencia a la clase PostController; y la acción es show(mostrar) el cuál su significado es determinado por el controlador
5. La aplicación crea una instancia de la petición del controlador para más adelante manejar la petición del usuario. El controlador determina que la acción show (mostrar) se refiere a un método llamado actionShow en la clase del controlador. Entonces crea y ejecuta filtros (por ej.: controles de acceso) asociados a esta acción. La acción es ejecutada si los filtros lo permiten.
6. La acción lee un modelo Post el cuál si ID es 1 en la base de datos.
7. La acción asocia una vista llamada show (mostrar) con el modelo Post.
8. La vista lee y muestra los atributos del modelo Post.
9. La vista ejecuta algunos atilugios.
10. Los resultados de la vista son embebidos en un diseño (layout)
11. La acción completa la prestación de la vista y muestra el resultado al usuario.

HTML 5

HTML son las siglas de HyperText Markup Language (“lenguaje de marcas de hipertexto”).

Es un estándar definido por la W3C (World Wide Web Consortium, comunidad internacional que trabaja en el desarrollo de los estándares de casi todas las tecnologías ligadas a la Web), que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc.

Han sido liberadas varias versiones a lo largo del tiempo, siendo actualmente la número 5 la que se está utilizando como más reciente.

Hay una gran cantidad de cambios y agregados en HTML5, muchos de ellos apuntan al manejo de multimedia (audio, videos), y a la mejora del modo de definir el código en sí mismo. Para nosotros, si bien todas esas nuevas características fueron usadas a la hora de escribir el HTML de nuestro sistema, el punto clave introducido fue la definición de la API de Geolocalización.

API de Geolocalización

Como es de suponerse, una de las necesidades técnicas prioritarias en este proyecto, ha sido poder comprender y utilizar las tecnologías disponibles para obtener información geográfica en tiempo real. La herramienta básica que resolvimos utilizar, es la denominada "Geolocation API"

(HTML5 specification, s.f.), o sea una interfaz de programación para la georreferenciación. Explicaremos en este capítulo, sus características principales, funcionamiento y otros detalles.

Técnicamente, no es parte de la descripción de la especificación de HTML5, ya que fue desarrollada y publicada como una especificación de la W3C, por separado; pero en general, por ser introducida por la misma entidad y en paralelo, se suele mencionar como una de sus funcionalidades.

Si bien diversas herramientas heterogéneas y diversas han sido utilizadas durante varios años, no fue sino hasta 2012 que la W3C lanzó una especificación formal de su funcionamiento, la cual se halla en constante modificación. Debido a que el modo de implementación queda a ser definido por cada cliente (es decir, cada fabricante de navegador puede modificar el funcionamiento interno de la interfaz) ha sido fundamental el paso dado, de definir tal estándar. De tal manera, aun siendo diferentes las implementaciones que usa por ejemplo Firefox comparadas con las de Internet Explorer, para el programador no deben suponer un problema, suponiendo que ambos navegadores se apegan al estándar.

Entre los puntos más destacables de la definición del estándar, se remarcán:

“La API de Georreferencia define una interfaz de alto nivel para la información de ubicación asociada solamente con el dispositivo cliente de la implementación, tales como latitud y longitud”

Como se ve, la posibilidad que se posee al utilizar esta API, es la de poder requerir al cliente web, su posición. La respuesta esperada serán los valores de latitud y longitud del lugar donde se halle tal cliente. A partir de esos dos valores numéricos, es que se construye todo el resto de la información inferida, como la ciudad en la que se halla, los amigos que están o no cerca, la hora actual local, y una innumerable cantidad de cuestiones que surgen de esos dos pequeños números.

“La API en sí misma, es agnóstica respecto de las fuentes de información de ubicación. Fuentes comunes de información de ubicación incluyen Sistema de Posicionamiento Global (GPS) y ubicación inferida desde señales de red tales como dirección IP, RFID, direcciones MAC de WiFi y Bluetooth, e identificación celular por GSM/CDMA; como así también por datos introducidos por el usuario”

Aquí observamos la forma de obtención de esa información. En la mayoría de las implementaciones de la API, el modo de consulta parte por tratar de identificar la presencia de la fuente más precisa, el GPS. Desde allí, comienza a buscar los demás emisores como WiFi o GSM menos precisos por naturaleza, hasta dar con uno que se encuentre disponible en el cliente utilizado. De este modo, si estamos accediendo a la API desde un celular con GPS habilitado, será esa la información que obtengamos como respuesta.

“La API es utilizada para obtener la posición geográfica de un dispositivo cliente. En casi todos los casos, esta información revela la ubicación del usuario, y en consecuencia, potencialmente se compromete su privacidad. Una implementación aceptable de esta especificación debe proveer un

mecanismo que proteja la privacidad del usuario, y ese mecanismo debe asegurarse que ninguna información de ubicación estará disponible a través de esta API sin el consentimiento y permiso del usuario”

Se destaca aquí, y se pone especial atención en la privacidad de los usuarios. Es obligación de los desarrolladores que implementen esta API, requerir permiso expreso del usuario antes de realizar envíos de información geográfica. Sabiendo que los navegadores soportados por nuestra aplicación, hacen un debido uso e implementación de la API de Georreferencia, podemos dar por hecho que nuestra aplicación no violará estos principios éticos de privacidad.

Por último, luego de haber visto que esta API era más que apta en su funcionamiento, como para incluirla en la aplicación que desarrollamos, fue necesario hacer un análisis del espectro de dispositivos que están actualmente soportándola. Afortunadamente, ya se halla bastante extendida, y la mayoría de los navegadores ya sean de computadoras de escritorio, como de dispositivos móviles, ya la han implementado. Este es un pequeño listado de los navegadores y dispositivos que aceptan el uso de la API:

Navegadores:

- Firefox 3.5+
- Chrome 5.0+
- Safari 5.0+
- Opera 10.60+
- Internet Explorer 9.0+

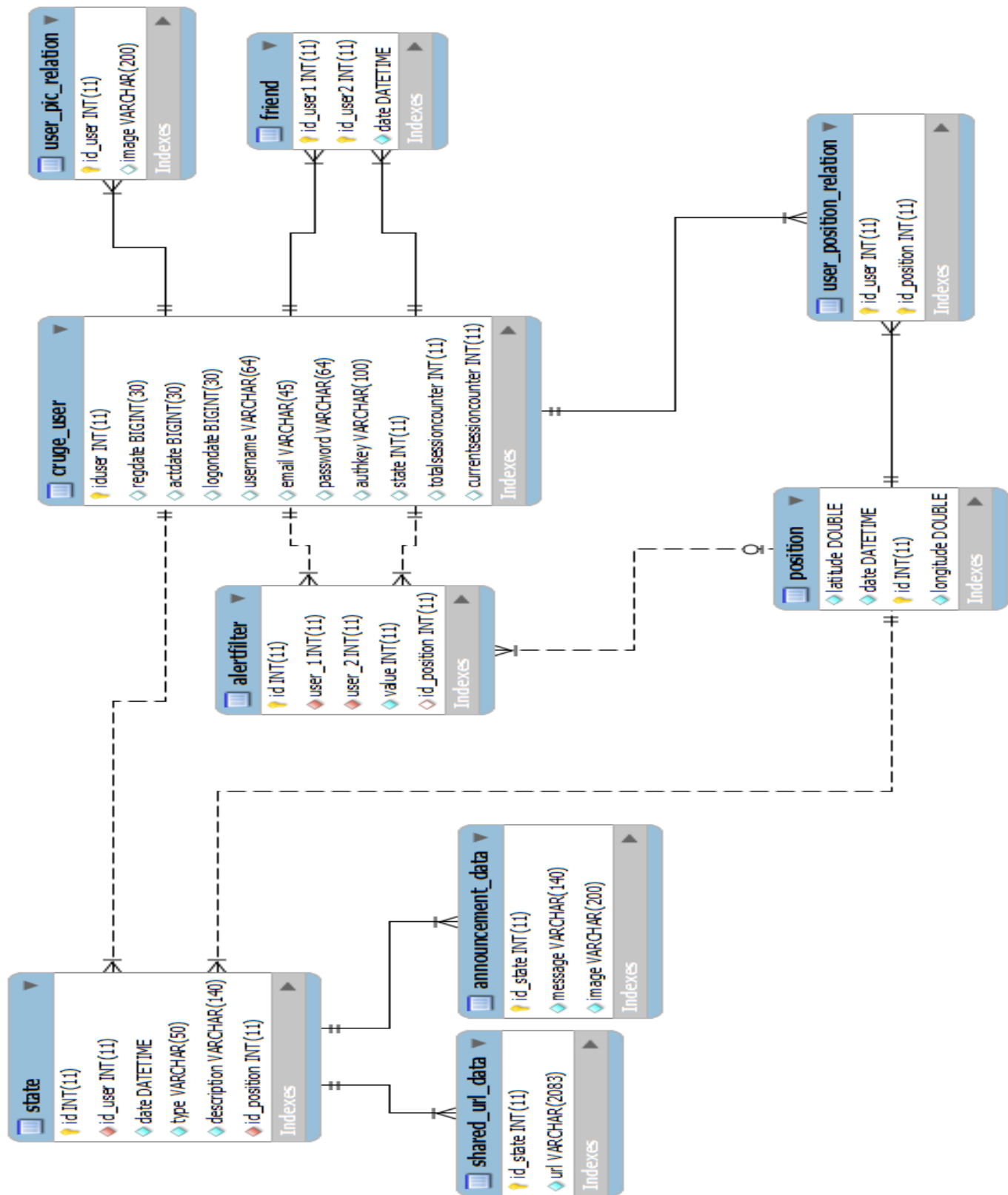
Móviles:

- Android 2.0+
- iPhone 3.0+
- Opera Mobile 10.1+
- Symbian (S60 3ra. y 5ta. generación)
- Blackberry OS 6

Capítulo 5 Diseño detallado

A continuación definimos el diseño detallado partiendo de la arquitectura MVC, expandiendo cada componente de la arquitectura para proveer un mayor entendimiento de la aplicación.

Modelo de datos



Descripción de las tablas (Fig. 5)

State: Almacena la información común a cualquier actualización de estado de un usuario de la red. Es una representación del estado más simple que solo posee una descripción.

Announcement_data: Estado particular que extiende de State y agrega campos como una imagen y un mensaje.

Shared_url_data: Estado particular que extiende de la tabla State y posee una url que permite compartir vínculos a otras páginas.

Alertfilter: Almacena la distancia que define un usuario en base a su posición con respecto a la posición de otro usuario. Este valor será usado como filtro para recibir las actualizaciones de estado del segundo usuario.

Position: Define posiciones mediante los valores de latitud y longitud. Adicionalmente posee una fecha en la cual se almaceno la información.

Cruge_user: Almacena la información de cada usuario de la aplicación. Posee información adicional relacionada con el módulo Cruge para el framework Yii para el manejo de usuarios y sesiones.

Friend: Define la relación de Amistad entre dos usuarios registrados en la table cruge_user. Adicionalmente posee una fecha en la cual se almaceno la información.

User_position_relation: Tabla que representa la relación entre un usuario y las posiciones almacenadas.

User_pic_relation: Tabla que almacena las imágenes de perfil que puede tener un usuario.

Como se podrá ver a continuación, los modelos de la aplicación están directamente derivados de cada tabla de la base de datos.

Descripción de los modelos

Los modelos son utilizados para mantener los datos y sus reglas de negocio relevantes.

Un modelo representa un solo objeto de datos. El mismo hace referencia a una tabla de la base de datos.

Cada campo del objeto de datos está representado por un atributo en el modelo. El atributo tiene una etiqueta y esta se puede validar contra un juego de reglas.

A continuación se observa un diagrama que lista todos los modelos de la aplicación y la relación entre ellos (Fig. 6):

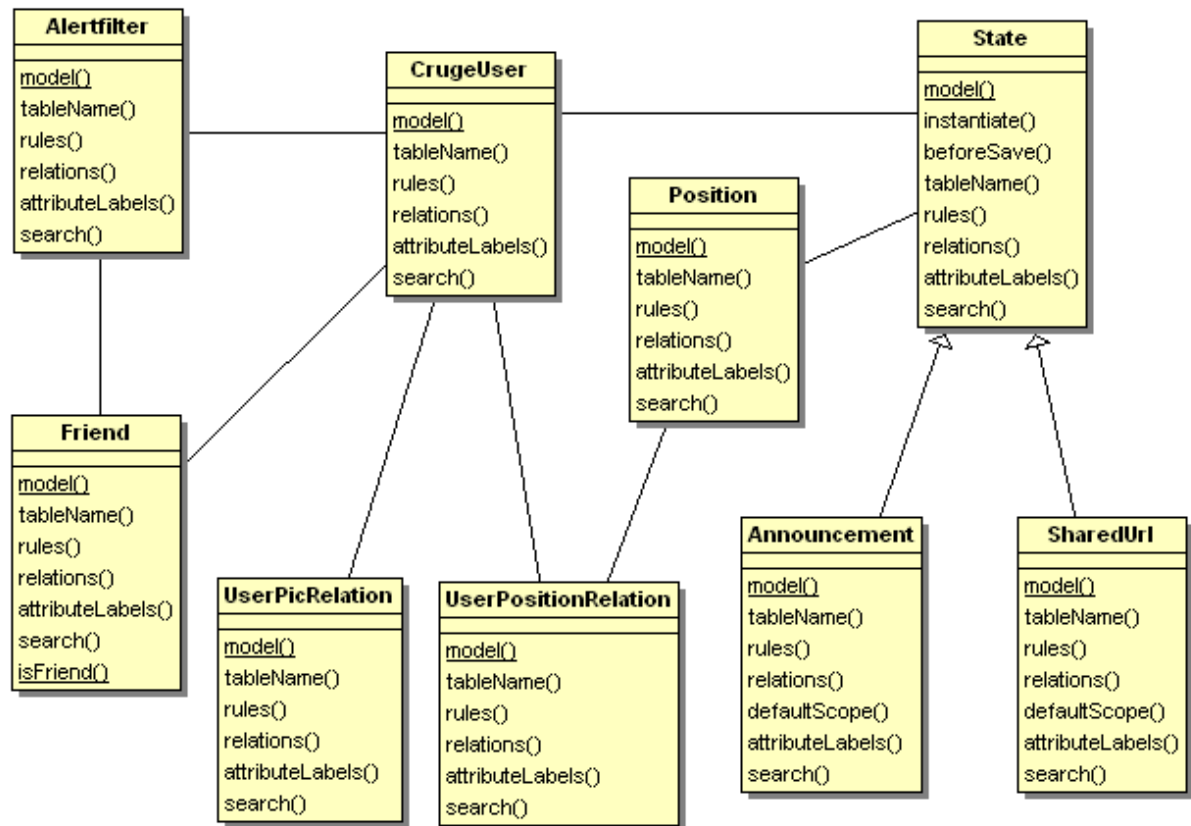


Fig. 6 Diagrama de Clases de Modelos

Referencia breve de los métodos

model: Constructor. Método que todos los modelos deben sobrescribir.

tableName: Devuelve el nombre de la tabla de la base de datos a la que representa el modelo.

Ejemplo del modelo Position:

```

public function tableName()
{
    return 'position';
}
  
```

rules: Devuelve un arreglo con reglas de validación definidas por el usuario. Estas se evalúan cada vez que se quiere utilizar un registro y es requerido que se cumplan para completar la acción.

Ejemplo del modelo State:

```

public function rules()
{
  
```



```

return array(
    array('id_user', 'date', 'description', 'required'),
    array('id_user', 'id_position', 'numerical', 'integerOnly'=>true),
    array('type', 'length', 'max'=>50),
    array('description', 'length', 'max'=>140),
    // The following rule is used by search().
    array('id', 'id_user', 'date', 'type', 'description', 'id_position', 'safe',
'on'=>'search'),
);
}

```

relations: Devuelve un arreglo donde están definido como se relaciona dicho modelo con otros.

Ejemplo del modelo State:

```

public function relations()
{
    return array(
        'announcementData' => array(self::HAS_ONE, 'AnnouncementData',
'id_state'),
        'position'=> array(self::BELONGS_TO, 'Position', 'id_position'),
        'idUser' => array(self::BELONGS_TO, 'CrugeUser', 'id_user'),
    );
}

```

attributeLabels : Cuando se diseña un form, generalmente se necesita mostrar una etiqueta para cada campo de entrada. La etiqueta le indica al usuario que clase de información se espera que cargue en el campo. Éste método devuelve un arreglo que mapea cada atributo del modelo con una etiqueta definida por el usuario.

Ejemplo del modelo Position:

```

public function attributeLabels ()
{
    return array(
        'latitude' => 'Latitude',
        'date' => 'Date',
        'id' => 'ID',
        'longitude' => 'Longitude',
    );
}

```

search: Define los atributos que serán utilizados en los filtros de búsqueda.

Métodos heredados

beforeSave: Método que es invocado antes de guardar un modelo. Generalmente utilizado con fines de validación. En este caso se usó para definir el atributo type del modelo State en la herencia.

instantiate: Método que crea una nueva instancia del modelo. Se utilizó para crear instancias del modelo State, cada vez que se creaba una instancia de uno de sus hijos.

defaultScope: Define el ámbito del modelo, el cual será utilizado en las búsquedas de la base de datos. Por ejemplo en nuestro caso: El defaultScope de announcement o sharedUrl es State.

Descripción de la Vista

Una vista es un script PHP que consiste principalmente en los elementos de la interfaz de usuario. Básicamente la vista es la página HTML y el código que provee de datos dinámicos a la página.

Las vistas son las responsables de presentar los modelos en el formato que el usuario final desea. Las vistas de la aplicación están divididas en dos grupos:

- Vistas complejas: son las vistas que se componen de otras vistas y a su vez utilizan más de un controlador y modelo (login, search, map).
- Vistas simples: de ABM (Alta Baja Modificación). Son las vistas que están asociadas a un modelo y un controlador y se utilizan para Obtener, Actualizar y Borrar el modelo asociado.

Descripción del Controlador

El controlador es el responsable de recibir los eventos de entrada desde la vista. Cuando un controlador es invocado, ejecuta una acción que utiliza los modelos necesarios y muestra la información a través de la vista apropiada.

Una acción, en su forma más simple, es un método de la clase controlador cuyo nombre comienza con action.

Los controladores y acciones están definidas por IDs. El ID del controlador se encuentra en la forma de path/to/xyz el cual es interpretado como el archivo de clase controlador protected/controllers/path/to/XYZController.php, donde xyz debe ser remplazada por el nombre de su controlador (ejemplo: post corresponde a protected/controllers/PostController.php). El ID de acción es el nombre del método sin el prefijo action. Por ejemplo si el controlador contiene el método actionEdit el ID de la acción correspondiente será edit.

Los usuarios realizan pedidos por un controlador y acción en términos de ruta. Una ruta se encuentra formada por la concatenación de un ID de controlador y un ID de acción separados por una barra. Por ejemplo la ruta post/edit se refiere a PostController y a su acción edit.

A continuación, un diagrama en el que se listan los controladores de la aplicación junto con sus acciones (Fig. 7):

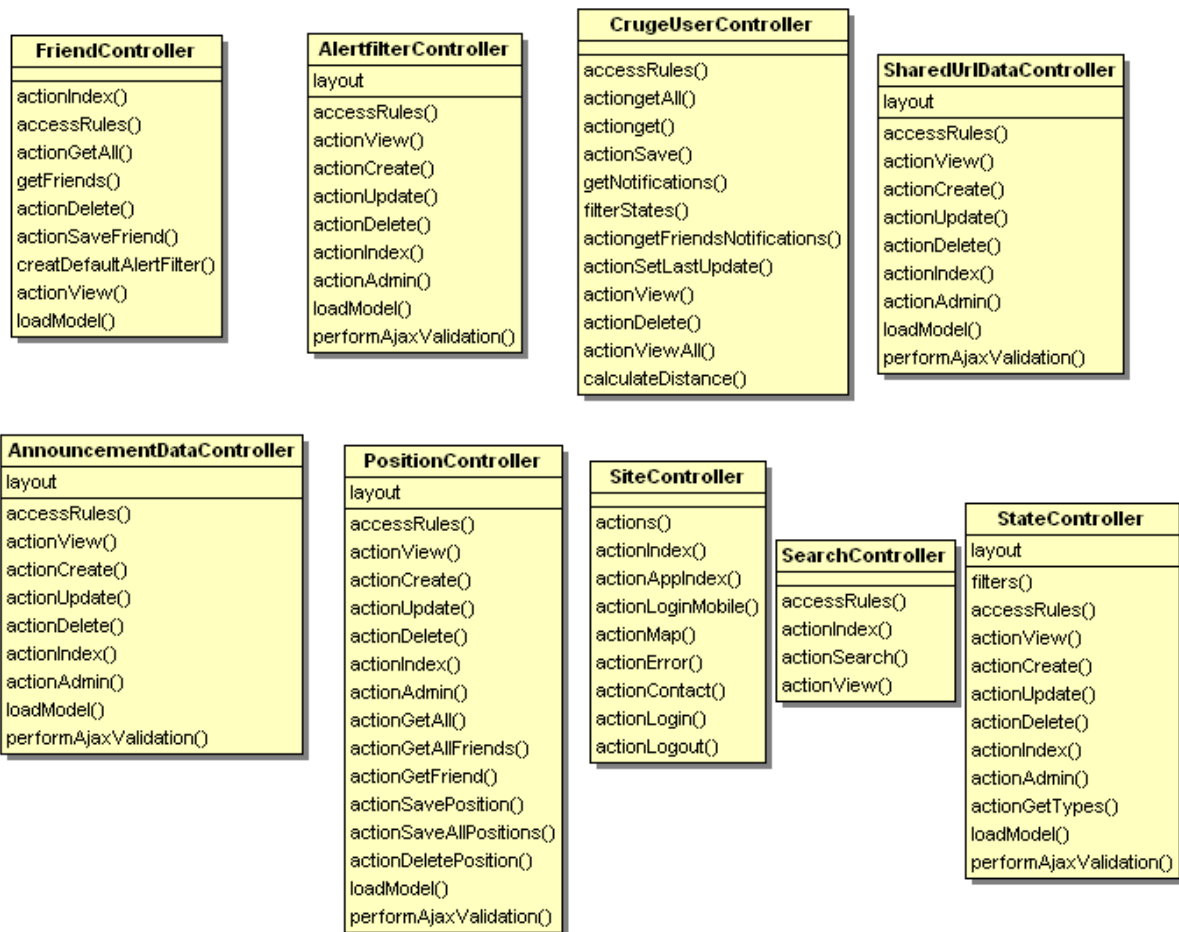


Fig. 7 Controllers

Referencia breve de los métodos:

Actions get, update, delete, save: Como se ve detallado, cada controlador define el CRUD (acrónimo de Crear, Obtener, Actualizar y Borrar del original en inglés: Create, Read, Update and Delete) de cada modelo. Se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software. Además le agregamos algunas funcionalidades como obtener o guardar todos los registros dados de un modelo (getAll, saveAll).

actionIndex: Se utiliza para listar todos los elementos de un modelo particular.

actionAdmin: Método que se utiliza para definir un comportamiento particular si el usuario con privilegios de Administrador esta logueado.

loadModel: Se utiliza para cargar el modelo asociado al controlador.

performAjaxValidation: Posibilita realizar validación AJAX de un form sobre un modelo dado.

accessRules: Define los permisos de cada tipo de usuario a acciones del controlador.

Ejemplo del controlador site:

```
public function accessRules()
{
    return array(
        // allow all users to perform login action
        array('allow',
            'actions'=>array('login'),
            'users'=>array('*'),
        ),
        // allow authenticated user to perform map action
        array('allow',
            'actions'=>array('map'),
            'roles'=>array('@'),
        ),
        // allow admin user to perform 'admin' action
        array('allow',
            'actions'=>array('admin'),
            'users'=>array('admin'),
        ),
    );
}
```

Capítulo 6 PlaceOn en funcionamiento

En este apartado, haremos un análisis de la aplicación que hemos desarrollado, a nivel funcional desde la perspectiva del usuario. Es decir, se procederá a describir las diferentes opciones, configuraciones y vistas que un usuario obtendrá al momento de utilizar la red PlaceOn desde un dispositivo compatible.

Para facilitar las explicaciones, podrán verse diversas capturas de pantalla que ilustrarán las situaciones que enunciaremos.

Haremos la descripción tratando de simular un recorrido normal que se haría de la aplicación en funcionamiento, desde un teléfono móvil. De este modo, como es de esperarse, al acceder a la raíz de la url donde se aloja PlaceOn, se observa una clásica pantalla que nos provee dos alternativas (Fig. 8):

- Iniciar sesión, en el caso de ser un usuario registrado en el sistema, introduciendo los campos de nombre de usuario (email) y contraseña.
- Registrar un usuario, en el caso de estar accediendo por primera vez al sistema.

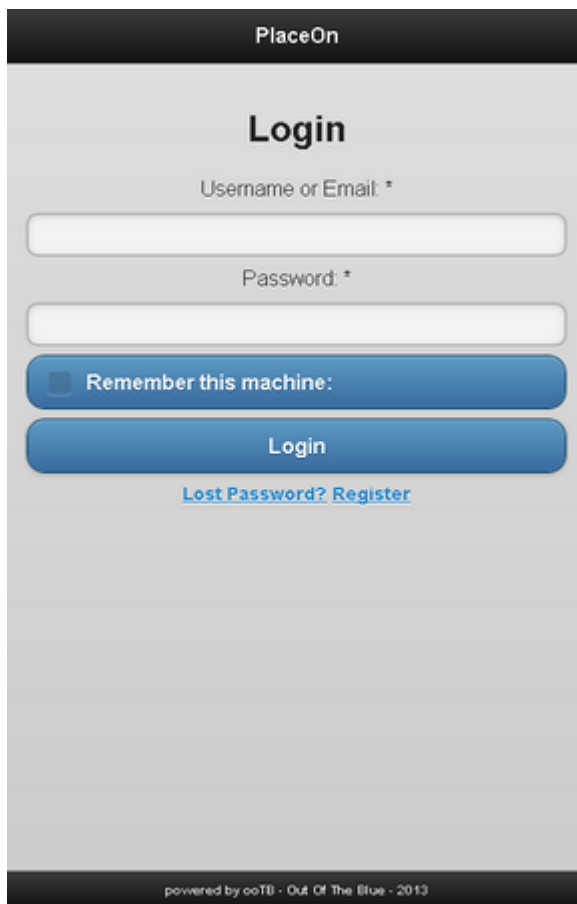


Fig. 8 Pantalla inicial

Suponiendo que somos nuevos en PlaceOn, ingresamos haciendo click en Register a la pantalla de registro de nuevo usuario.

Como vemos en las siguientes dos imágenes (Fig. 9/10), será presentado al usuario, un formulario donde se pedirá el ingreso de información necesaria para el registro. Un aspecto destacable del formulario que se presenta, es que el mismo se basa en los atributos que el modelo de usuario posee, de manera que pueda hacerse una rápida adaptación ante algún requerimiento, por ejemplo agregar un campo requerido más, para lo cual no sería necesario modificar las presentes vistas.

The image displays two side-by-side screenshots of the 'PlaceOn Register' form. Both screens have a black header with 'PlaceOn' in white. The title 'Register' is centered in bold black text. Below the title is a light gray box containing the registration fields. The left screenshot shows a form for a user named 'Test1' with email 'tes1@domain.com'. The right screenshot shows a form for a place named 'Place1' with email 'place1@domain.com'. Both forms have fields for Username, Email, Password, and a 'Generate a new password' button. Below these is a 'Profile' section with a checkbox 'Is Place' (unchecked on the left, checked on the right) and a 'Register' button at the bottom. The footer of both screens reads 'powered by ootB - Out Of The Blue - 2013'.

Fig. 9/10: Pantalla de registro de nuevo usuario

Aquí creamos dos usuarios, uno que representa a una persona "Test1", y otro que representa a un lugar "Place1". Hay algunos puntos en los que se diferencian estos dos tipos de usuario, los cuales serán explicados en el presente recorrido.

Volviendo a la pantalla inicial, será necesario iniciar nuestra sesión. Desde la pantalla principal, ya sea un usuario o un lugar, se podrá ingresar a la aplicación colocando los datos correctos (Fig. 11/12).

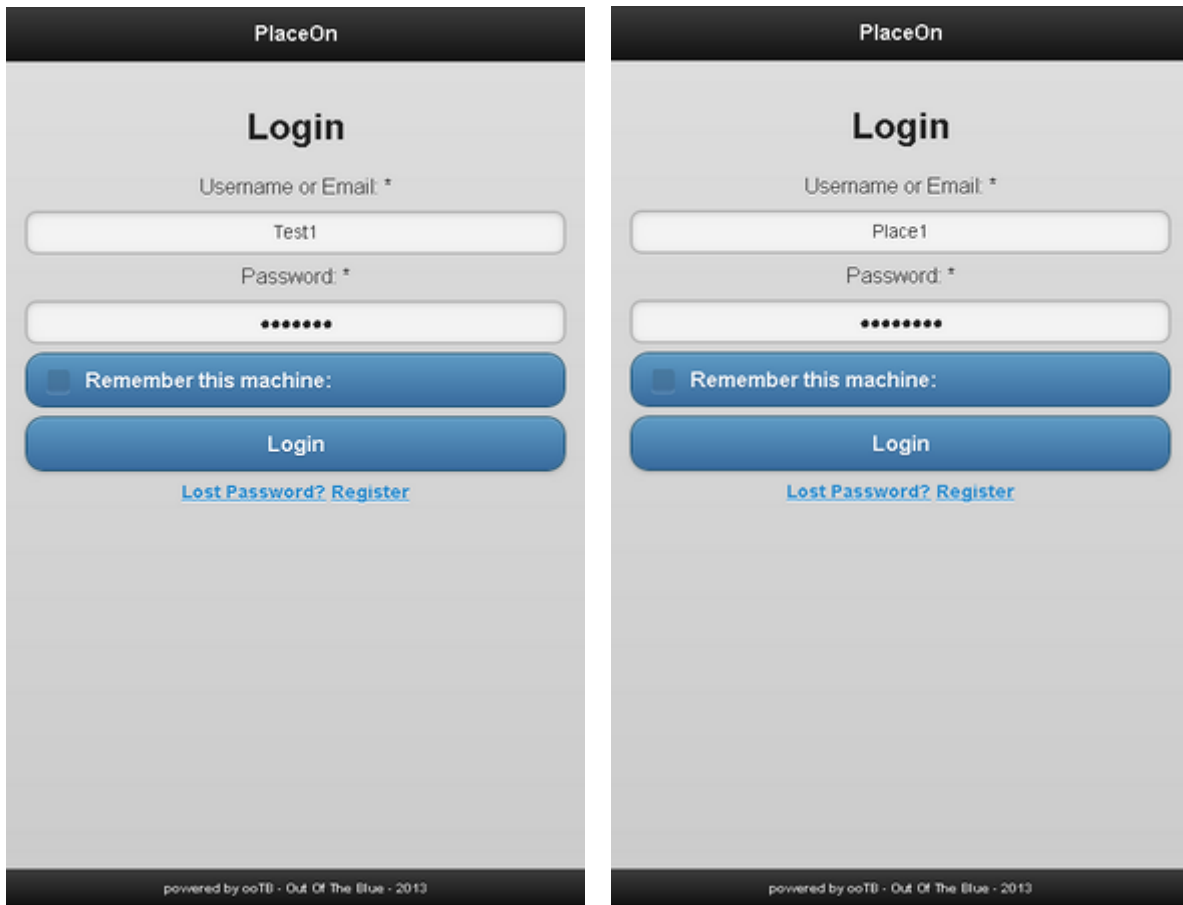


Fig. 11/12 Pantalla de inicio de sesión de usuarios

Habiendo accedido al sistema, podrá verse una pantalla principal que presenta un mapa.

En caso de tratarse de un usuario, se verá su ubicación actual, la cual se irá actualizando en tiempo real. Como es de suponerse, el pequeño "hombrecito", nos representa y ubica en el mapa.

Aquí tenemos una primera diferencia con respecto al usuario de tipo "lugar" (Fig. 13/14), ya que ante nuestro primer acceso, el sistema pedirá que indiquemos la ubicación física de dicho lugar, la cual permanecerá estática independientemente del movimiento que el dispositivo que inició la sesión realice. Esto es de esperarse, suponiendo que nuestro usuario representa, por ejemplo en nuestro caso, a un supermercado.

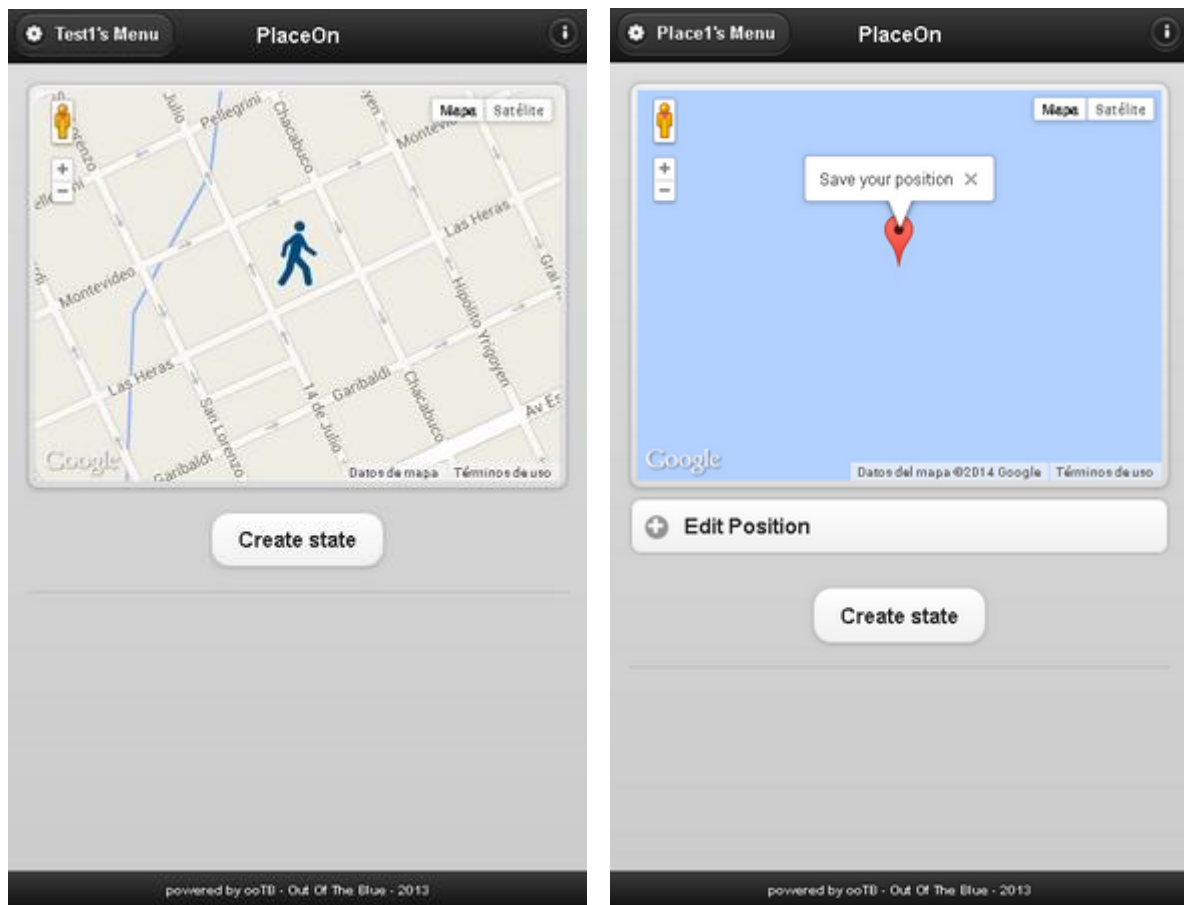


Fig. 13/14 Pantalla inicial de usuario simple y usuario lugar

Como vemos, la ubicación del lugar, puede hacerse manualmente "arrastrando" mediante "drag n' drop" del indicador rojo, o bien colocando un texto que identifique una dirección postal, o bien pidiendo al sistema que indique mi posición actual como la posición fija del lugar de aquí en más (Fig. 15).

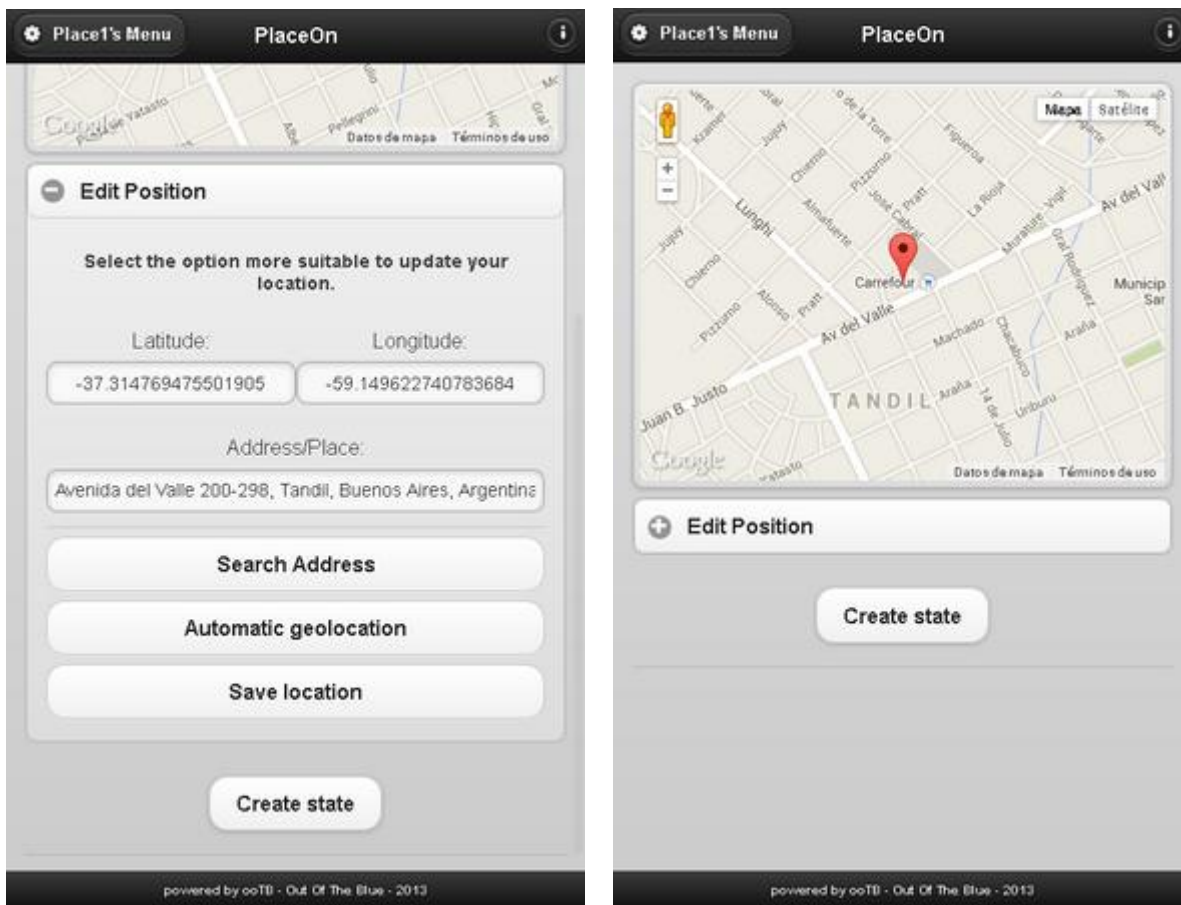


Fig. 15/16 Pantalla de edición de posición fija de lugar y

Una vez guardada la posición del lugar, se podrá ver la siguiente pantalla (Fig. 16), donde además de ser necesario, se pueden hacer ajustes de la posición fijada.

Si se accede al botón superior izquierdo (Fig. 17), se despliega el menú de navegación toda la aplicación. La opción Map, nos ubica en la pantalla que ya vimos del mapa. Las otras opciones Search, Friends y Profile, son parte del recorrido y serán descriptas en breve. Obviamente, la opción Logout, permite poder terminar la sesión activa.

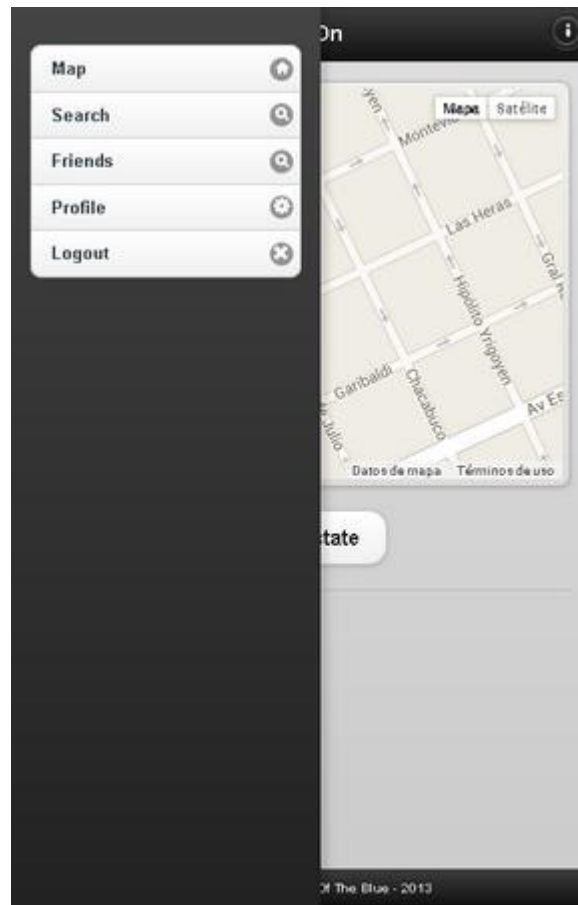


Fig. 17 Menú de navegación

Como siguiente acción, describiremos la pantalla de perfil propio, a la que se accede desde la opción Profile (Fig. 18/19). Aquí por ejemplo, podemos ver que es posible editar la información asociada a nuestro usuario, además de establecer imagen de perfil. Para ambos casos, usuario o lugar, la presentación es similar. También es posible ver la lista de estados creados por nuestro usuario a lo largo del tiempo, aunque por ahora no hemos creado ninguno.

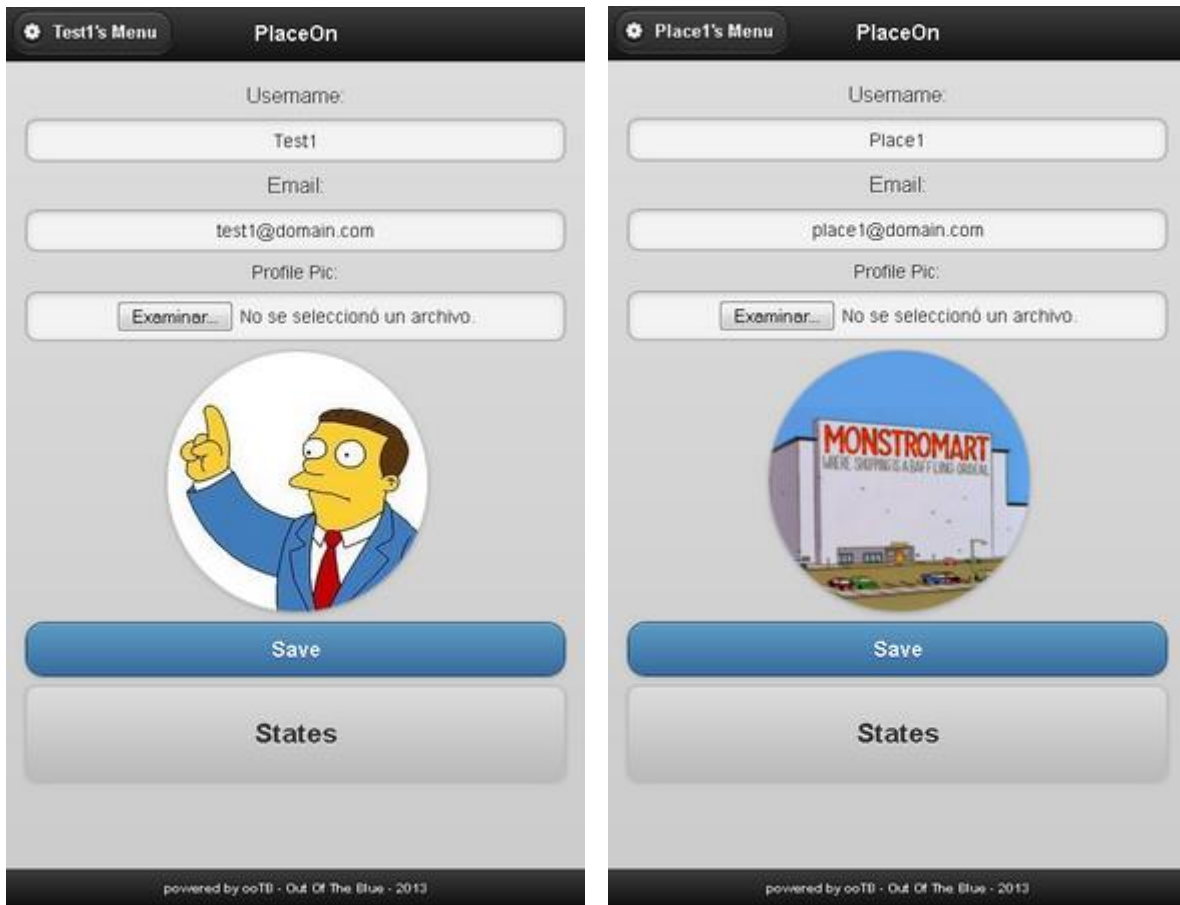


Fig. 18/19 Pantalla de perfil propio

Desde la opción Search, se puede obtener un listado actual de usuarios registrados en el sistema (Fig. 20), con el fin de analizar la posibilidad de añadirlos como amigos. En este caso accedemos al perfil de Test2 (Fig. 21) y lo añadimos a nuestros amigos. De este modo, a partir de ahora, si accedemos a la opción Friends (Fig. 22), podremos verlo.

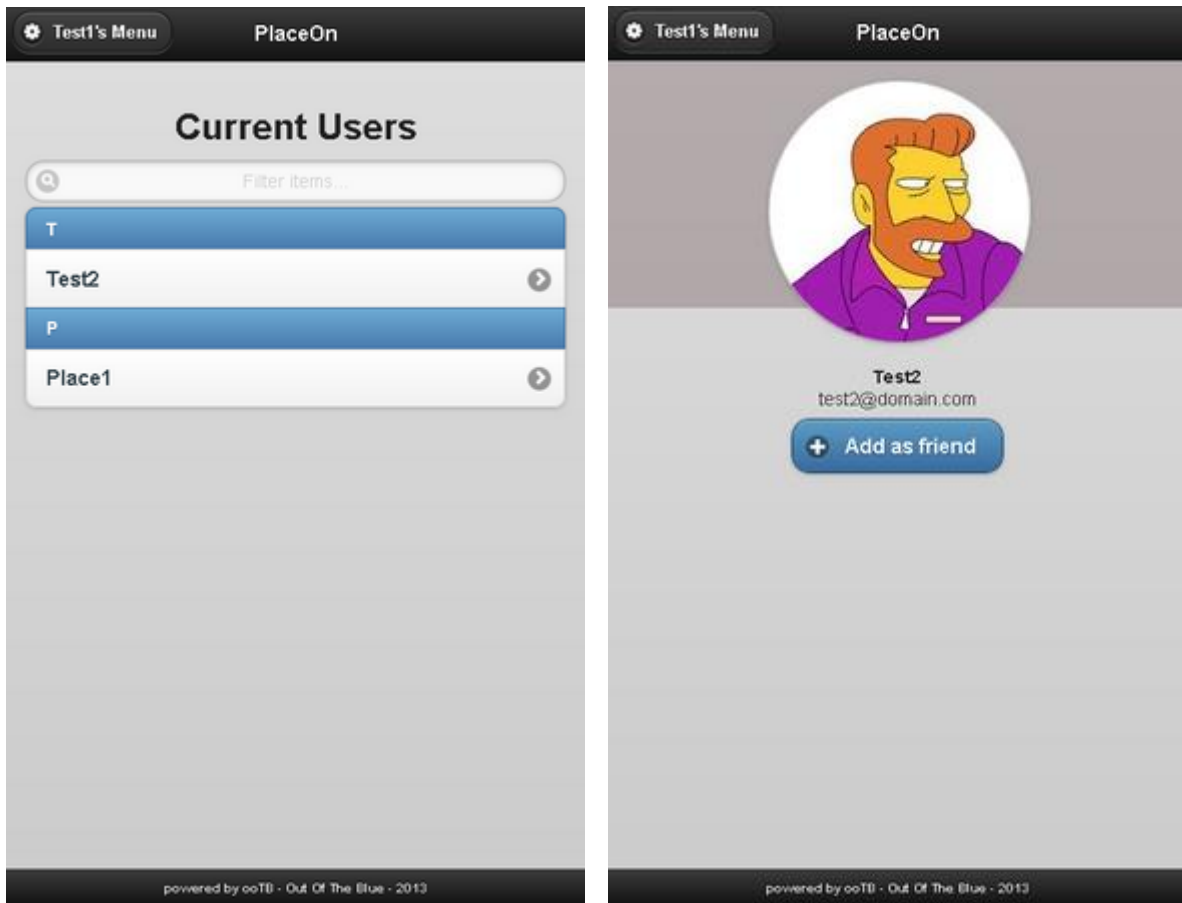


Fig. 20/21 Pantalla de lista de usuarios y pantalla de perfil de un usuario no amigo

El perfil de un usuario que es amigo nuestro (Fig. 23), ahora posee algunas nuevas funcionalidades. Además de poder eliminarlo de nuestra lista de amistades, podemos ver los estados que nuestro amigo ha publicado, además de poder configurar el "filtro de alertas".

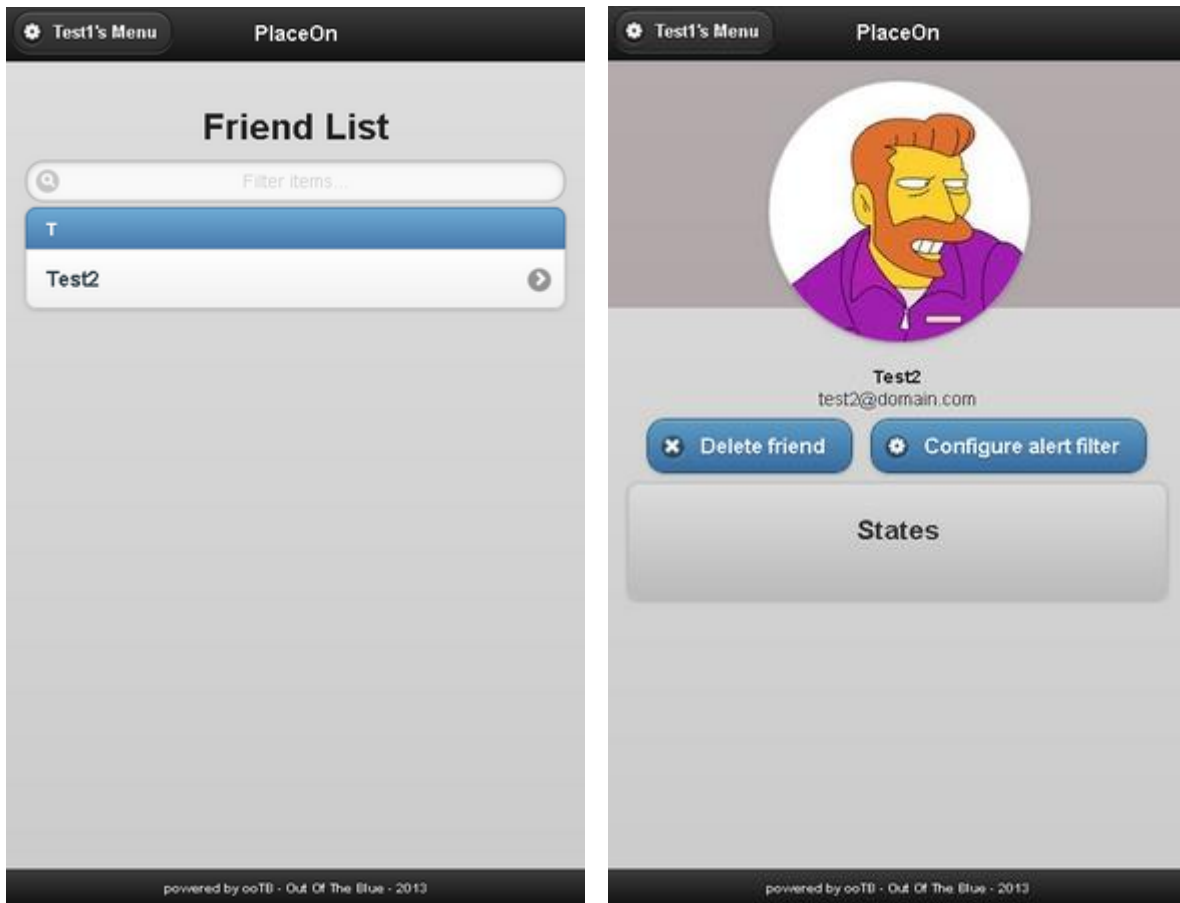


Fig. 22/23 Pantalla de listado de Amigo, y pantalla de perfil de un amigo

El filtro de alertas (Fig. 24), permite al usuario establecer las notificaciones que desea recibir de un amigo en particular. Como se ve en esta pantalla, es posible decidir el radio de interés de los estados, de forma que solamente, por ejemplo, tengamos notificaciones de "Test2" si son estados generados a menos de 999 metros de distancia con respecto a nuestra posición actual. También, si se escoge la opción "Use customized fix position to filter" se puede fijar un punto de referencia en lugar de nuestra posición real. Por ejemplo, esto es útil si me interesa tener las notificaciones de noticias cercanas a mi casa.

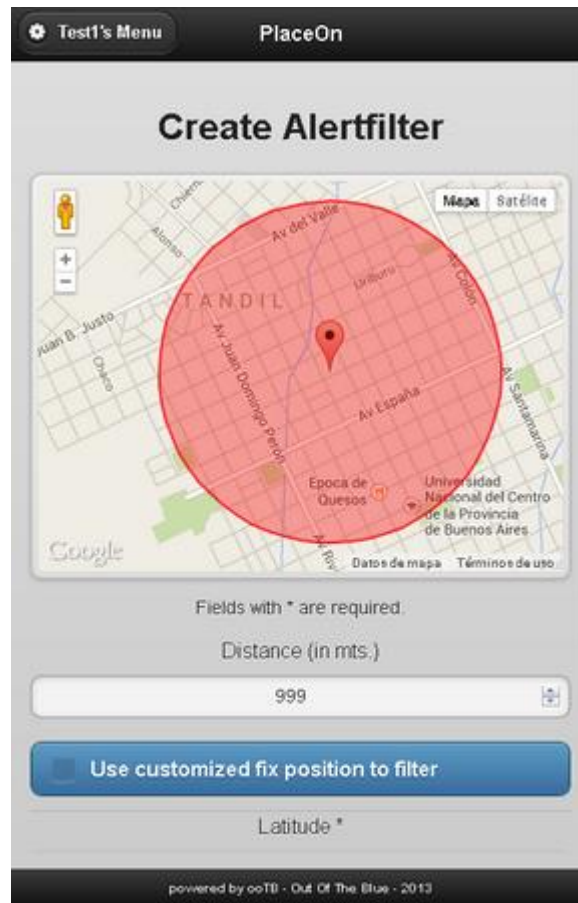


Fig. 24 Pantalla de filtro de alertas para un amigo

Estuvimos describiendo filtros de alertas de notificaciones, de manera que ahora describiremos que sucede en PlaceOn cuando un nuevo estado de un amigo, que está dentro del rango de interés, es creado.

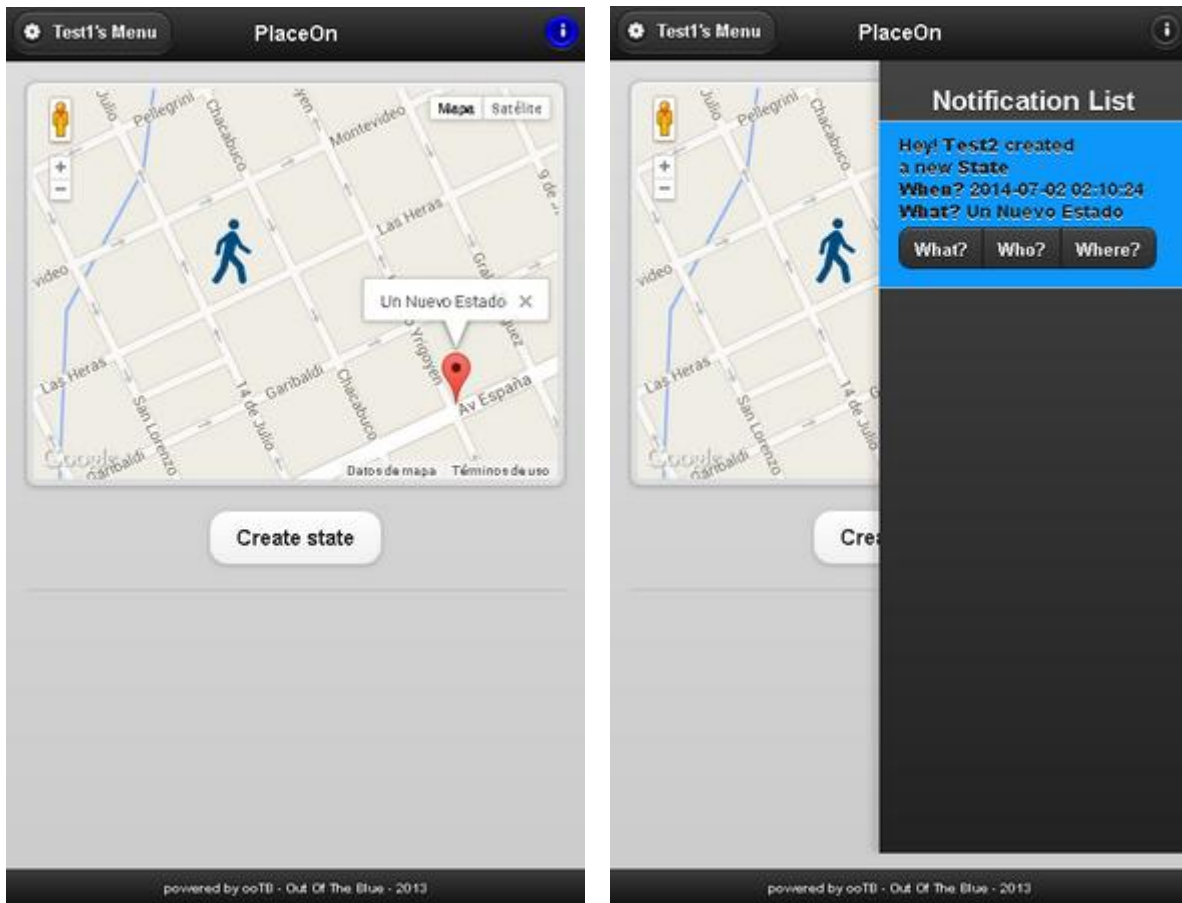


Fig. 25/26 Pantalla principal con notificación y panel de notificaciones desplegado

Como vemos aquí (Fig. 25), el mapa se ve modificado por la creación del estado nuevo de nuestro amigo, además de que un destello azul ilumina el icono superior derecho. El pin rojo que indica la posición del estado, al ser señalado, agrega un pequeño texto donde vemos el nombre que se le dio a dicho estado.

Por otro lado, si se accede al icono azul (Fig. 26), se despliega la lista de todas las notificaciones, las cuales figuran en celeste solamente si aún no fueron leídas. Los tres botones What? Who? y Where?, permiten ver detalles del estado, acceder al perfil de amigo que lo ha creado, y centrar el mapa en el punto, respectivamente.

Aquí (Fig. 27) se puede ver la pantalla de detalles de estado a la que accedimos haciendo click en What?

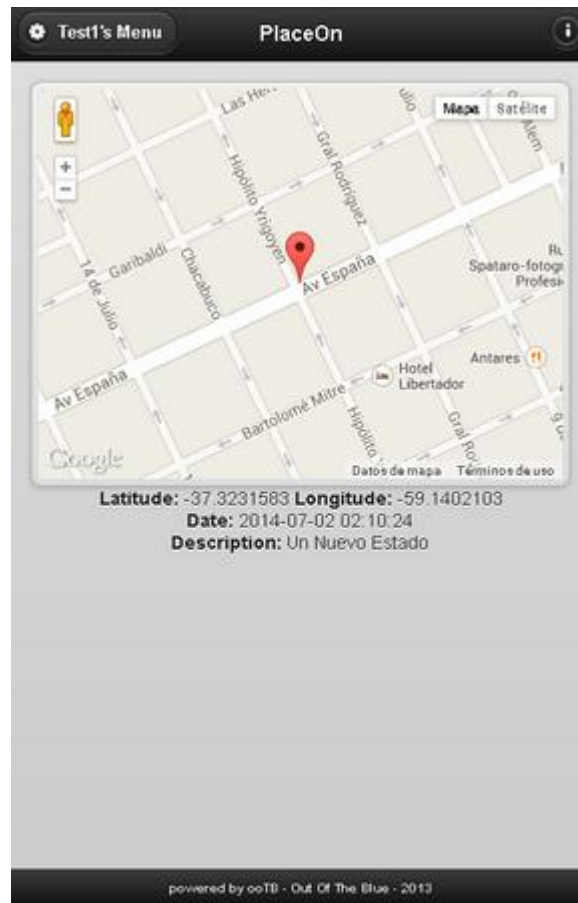


Fig. 27 Pantalla de Estado Simple

Al tratarse de un Estado simple, la información brindada es el mapa que nos sitúa en el punto, las coordenadas del mismo, la fecha de creación y la descripción.

Volviendo a la vista principal, observamos nuevamente una notificación emergente "Oferta" (Fig. 28).

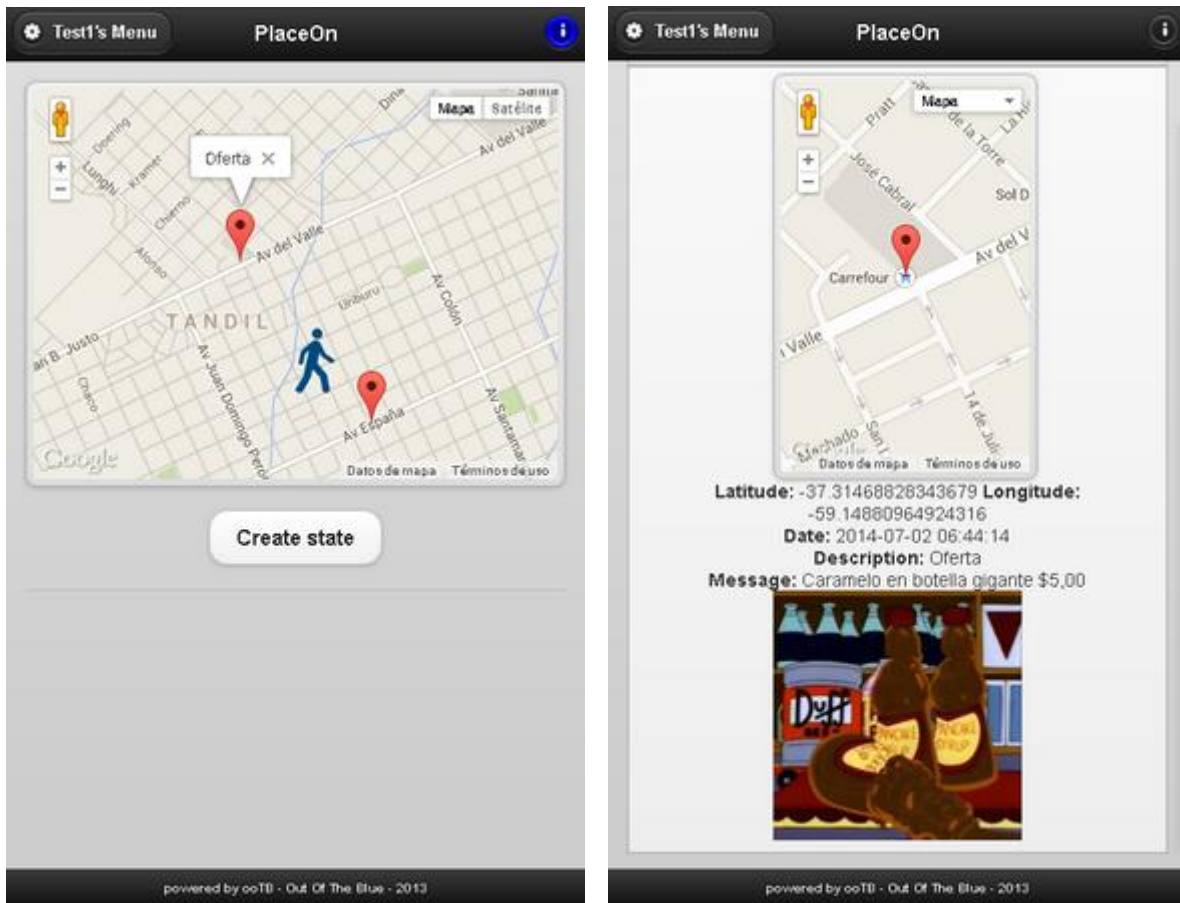


Fig. 28/29 Nueva notificación en pantalla principal y vista del Estado Anuncio

Al acceder a sus detalles, podemos ver que se trata de un Estado de tipo Anuncio (Fig. 29), el cual cuenta con mapa, fecha, descripción, un nuevo campo Message y una imagen asociada. Esto es debido a que se trata de un Estado especial de anuncio, el cual extiende las características del Estado simple con los atributos mencionados.

Si accedemos al perfil del usuario que ha creado este Estado (Fig. 30), podemos ver que se ha agregado a su lista de Estados, el que acabamos de ver. Si vamos a su configuración de filtro (Fig. 31), vemos que el estado que hemos visto, fue mostrado porque efectivamente, se encontraba dentro del rango de interés.

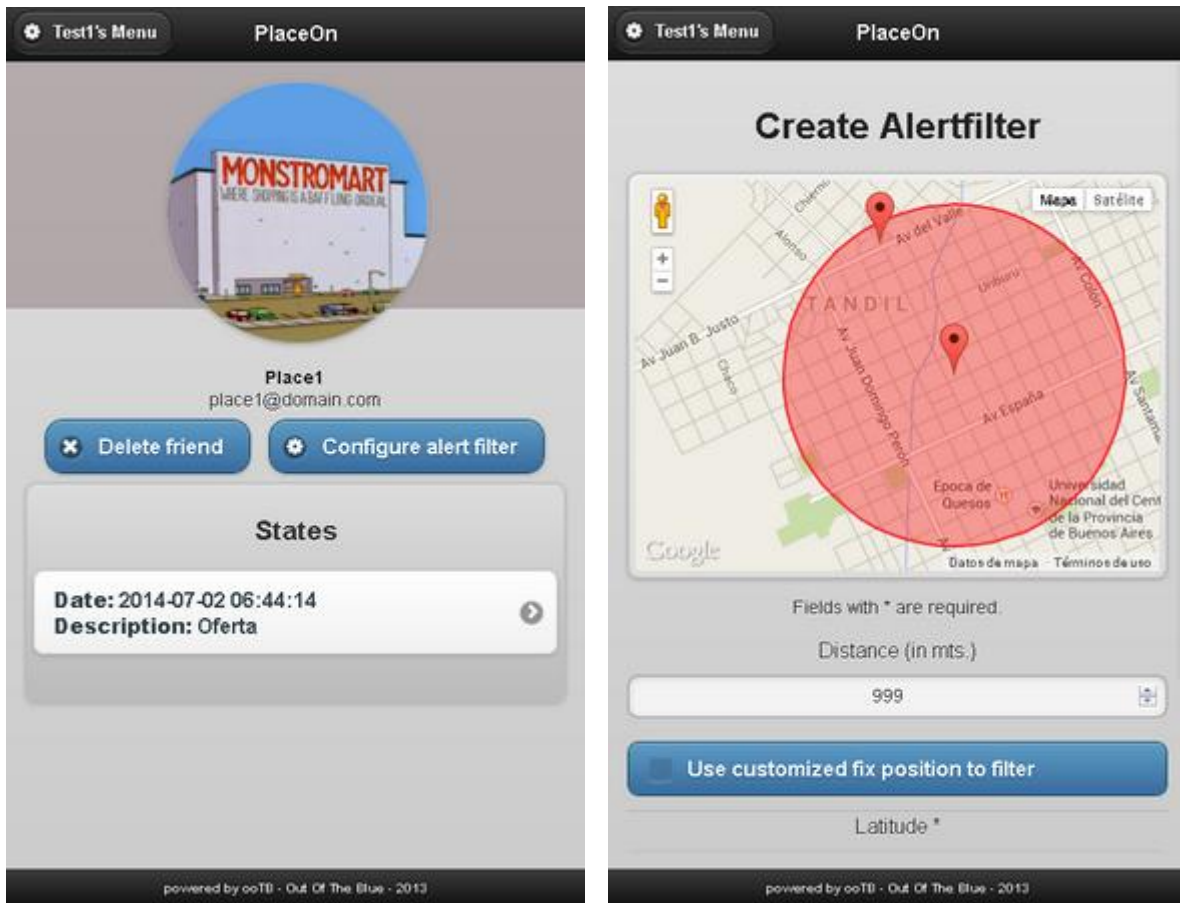


Fig. 30/31 Pantalla de Perfil de usuario tipo lugar, y filtro de alerta para el usuario lugar amigo

Volviendo a la vista inicial de mapas, podemos ver que ahora la lista de notificaciones muestra a las dos que hemos descrito, pero ya no con el azul característico de las "no leídas" (Fig. 32).

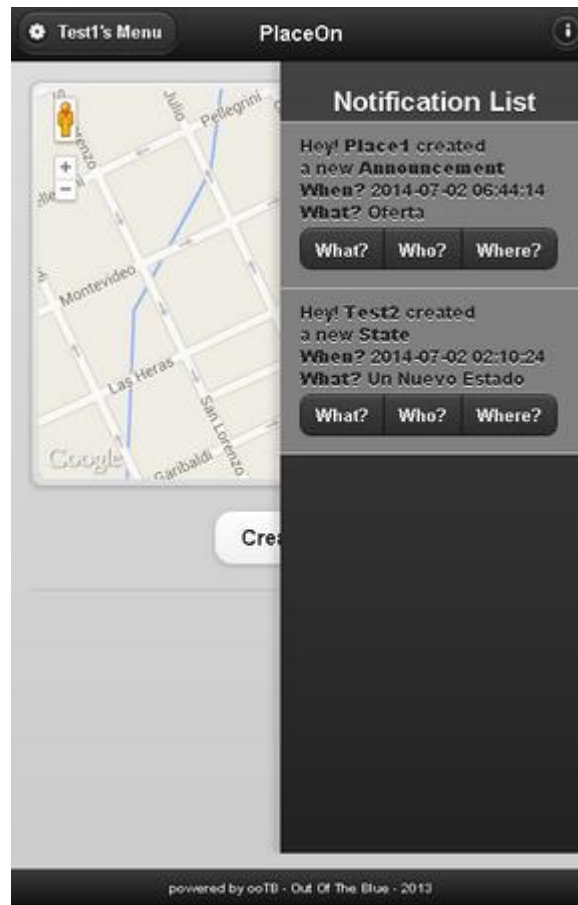


Fig. 32 Pantalla principal con lista de notificaciones leídas

Ahora para finalizar, haremos un recorrido por las diferentes opciones para la creación de estados.

Cuando se accede al menú de creación de estado (Fig. 33), vemos las tres opciones existentes actualmente, State, Announcement y Shared link. Mostraremos un ejemplo de cada uno de ellos.

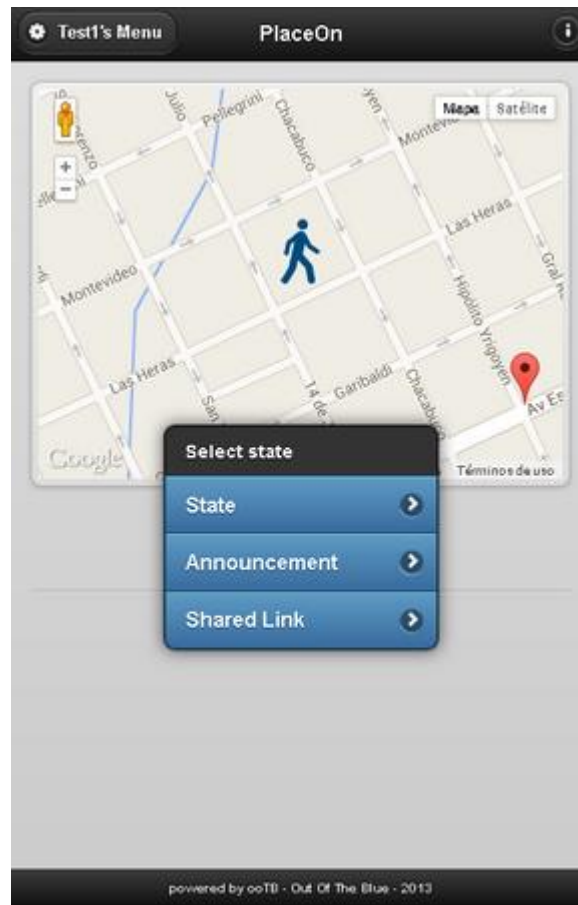


Fig. 33 Menú de creación de estados

Un Estado simple, o State (Fig. 34), cuenta con una descripción y fecha de creación, además claro de la posición, que por defecto está situada en nuestra ubicación actual. Una vez creado, tendremos una vista final (Fig. 35), que muestra el estado tal cual será visto por nuestros amigos.

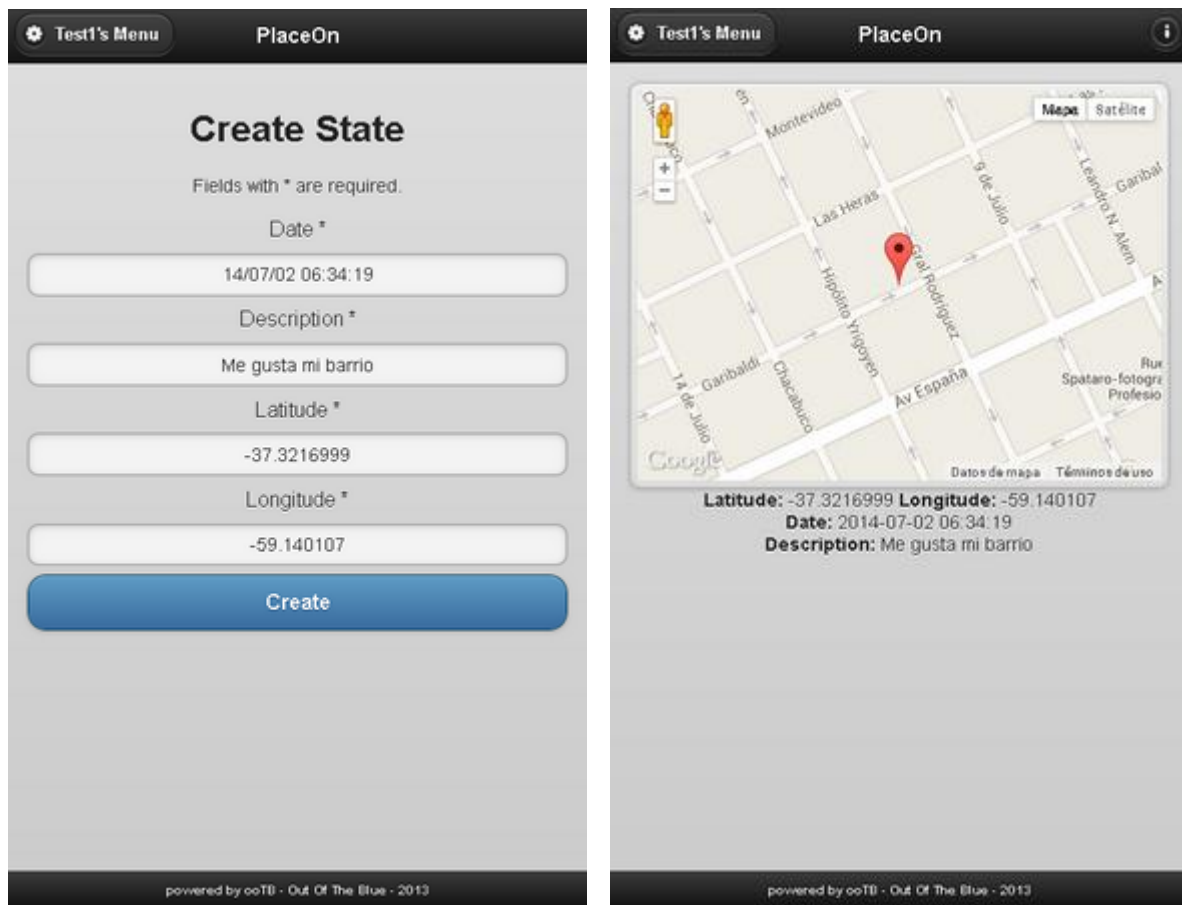


Fig. 34/35 Pantalla de creación de estado simple y pantalla del estado simple creado

Un Anuncio, o Announcement, añade a los atributos de Estado simple, un campo extra de Message (Fig. 36), además de permitirnos añadir una imagen asociada. Una vez más, cuando el estado se encuentra creado, podemos ver cómo será visualizado (Fig. 37).

Create AnnouncementData

Fields with * are required.

Description

Hecho delictivo

Message

La policía local detuvo al ladrón

Latitude *

-37.3220753

Longitude *

-59.1381115

Image

Examinar... fakeNews.png

Create

powered by ootB - Out Of The Blue - 2013

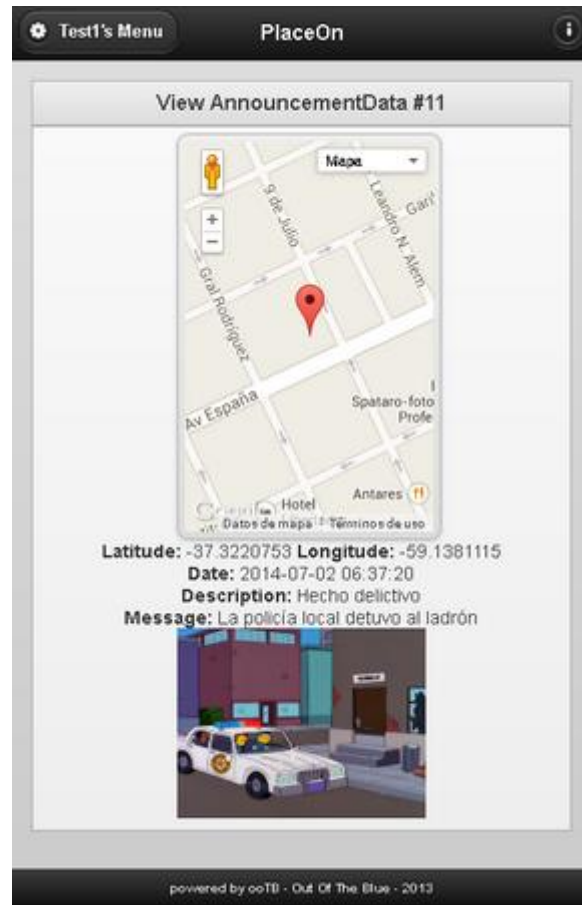


Fig. 36/37 Pantalla de creación de estado anuncio y pantalla del estado anuncio creado

Por último, hemos definido un estado que permite compartir enlaces dinámicamente, al que denominamos Shared URL. Aquí además de una descripción, tenemos disponible un campo para agregar la URL (Fig. 38). Una vez guardada, podremos tener una vista reducida de su contenido, la cual podrá verse cuando dicho estado sea accedido (Fig. 39).

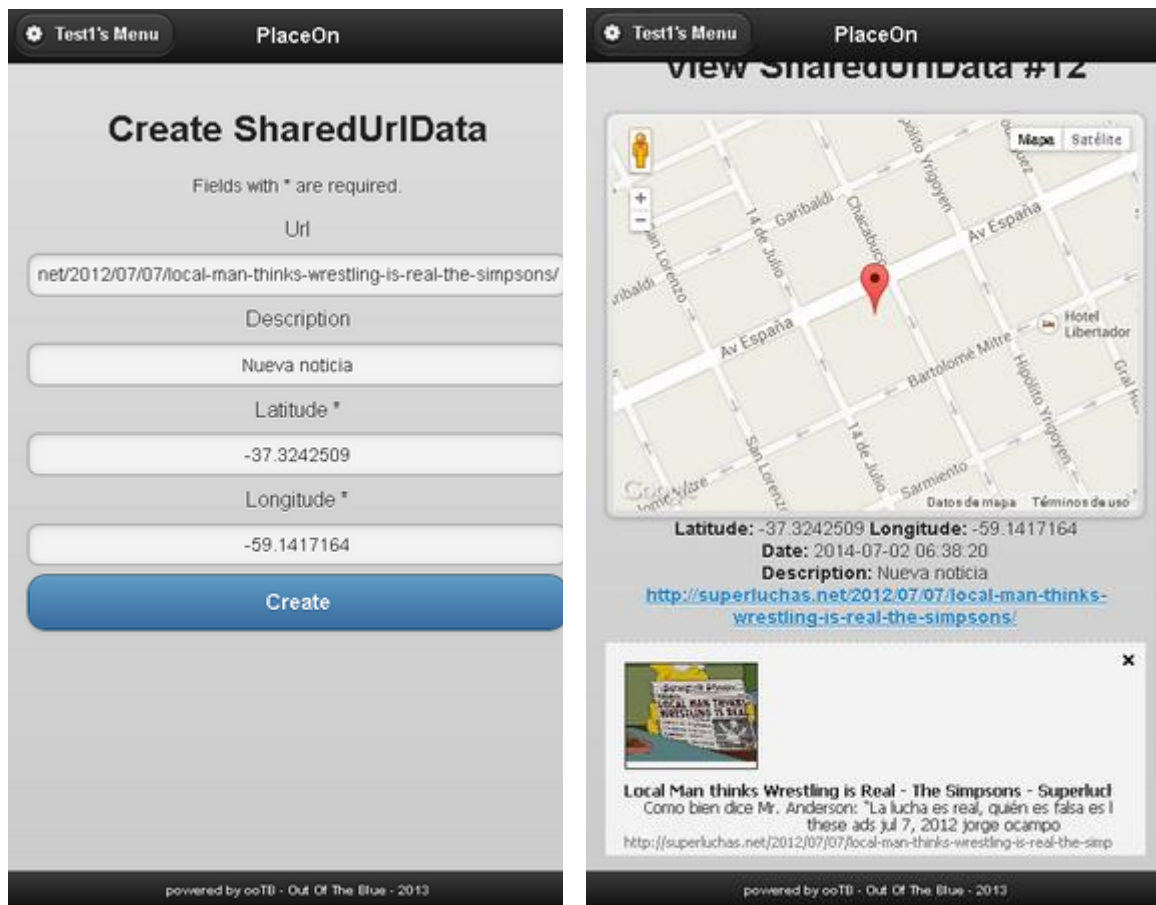


Fig. 38/39 Pantalla de creación de estado Url y pantalla del estado Url creado

Para finalizar, nos dirigimos a la página de nuestro perfil personal (Fig. 40), para mostrar cómo se han listado los tres estados que acabamos de crear.



Fig. 40 Pantalla de perfil propio con el listado de estados creados

Como caso de uso alternativo, mostraremos otra posible implementación del sistema. En este escenario, un usuario se encuentra recibiendo las actualizaciones de estado de un coche perteneciente a una línea del transporte público. Por otro lado, el móvil envía vía 3G y con el GPS funcionando, estados periódicos, anunciando su posición. De esta manera le permite al usuario saber en qué momento el coche entra en el radio definido como de interés, pudiendo anticiparse a su llegada a la parada deseada. La visualización de estos estados que anuncian la llegada inminente del autobús, se puede observar en las siguientes capturas (Fig. 41/42).

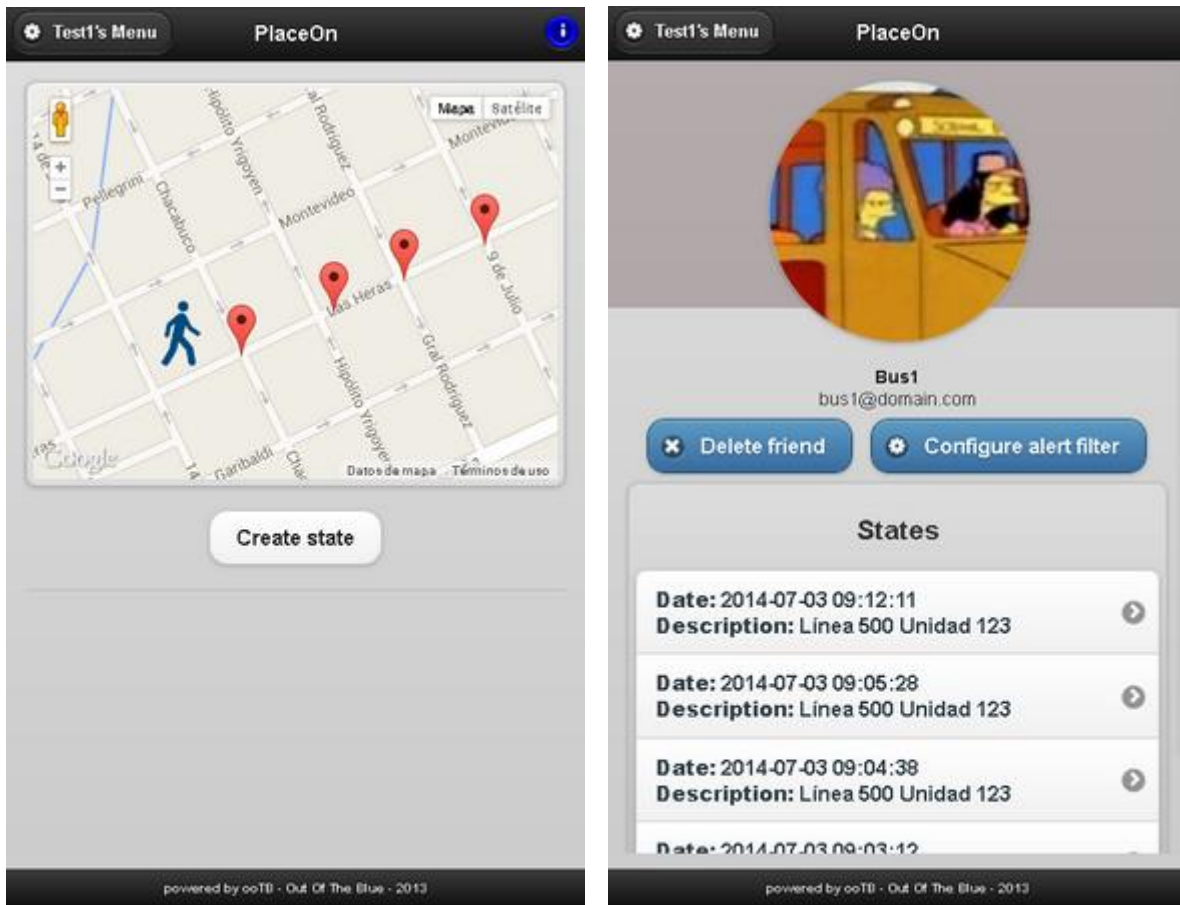


Fig. 41/42 Pantalla principal con notificaciones de transporte público y pantalla de perfil del usuario transporte público con lista de estados

Diagramas de secuencia

A continuación modelamos la interacción entre los distintos objetos para explicar ciertos escenarios de funcionamiento de la aplicación:

Login

El usuario accede a la aplicación y se encuentra con la página de login. Ingresa su usuario y contraseña y el sistema verifica los datos. Siendo estos correctos lo redirige a la página principal de aplicación que es la vista del mapa (Fig. 43).

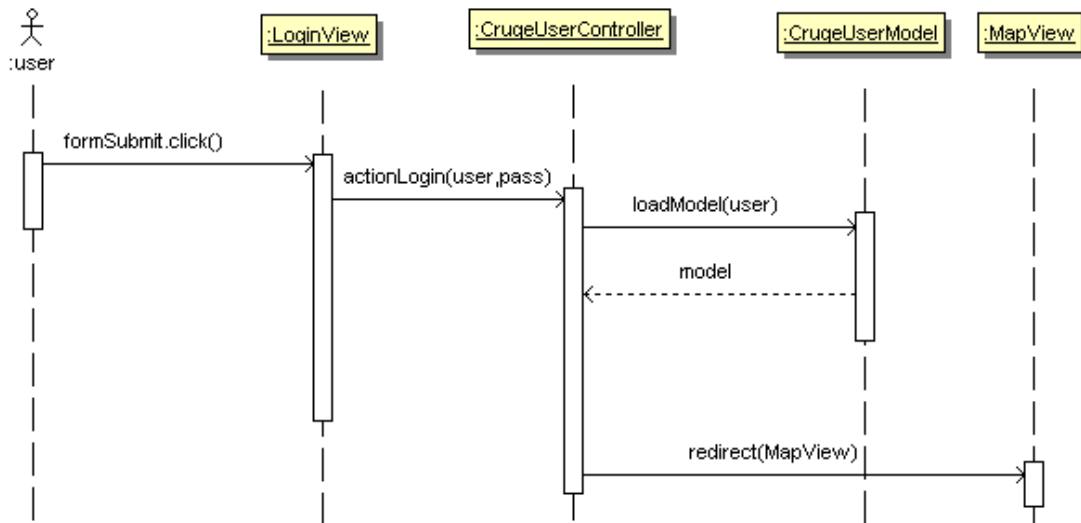


Fig. 43 Diagrama de secuencias para el acceso al sistema o login

Configurar filtro de alerta

Un usuario logueado a la aplicación entra al perfil de un amigo para modificar la distancia del filtro de alertas. Cabe destacar que cada vez que una amistad es creada, se crea un filtro de alerta por defecto de una distancia predeterminada (Fig. 44).

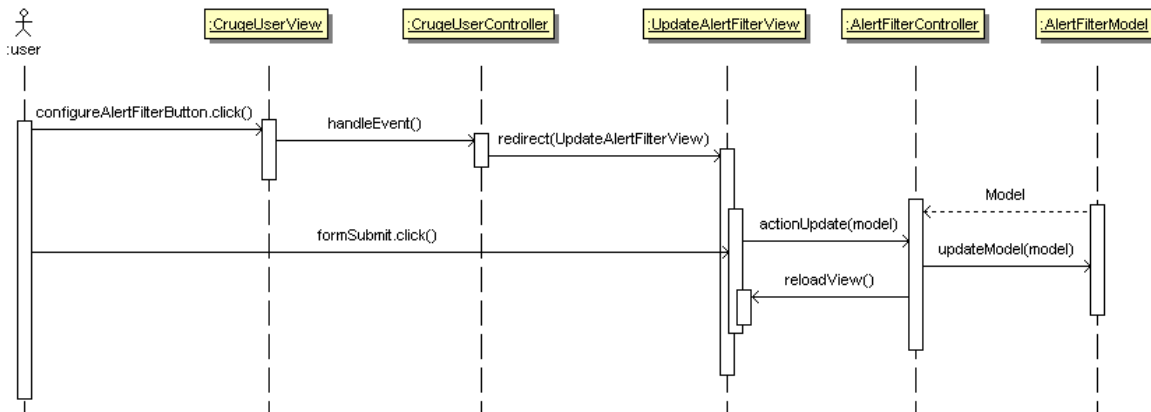


Fig. 44 Diagrama de secuencias para la configuración del filtro de alertas para un amigo

Crear estado sharedUrl

Un usuario logueado crea un estado para compartir un link. El sistema toma su posición actual del mapa y guarda la información del estado (Fig. 45).

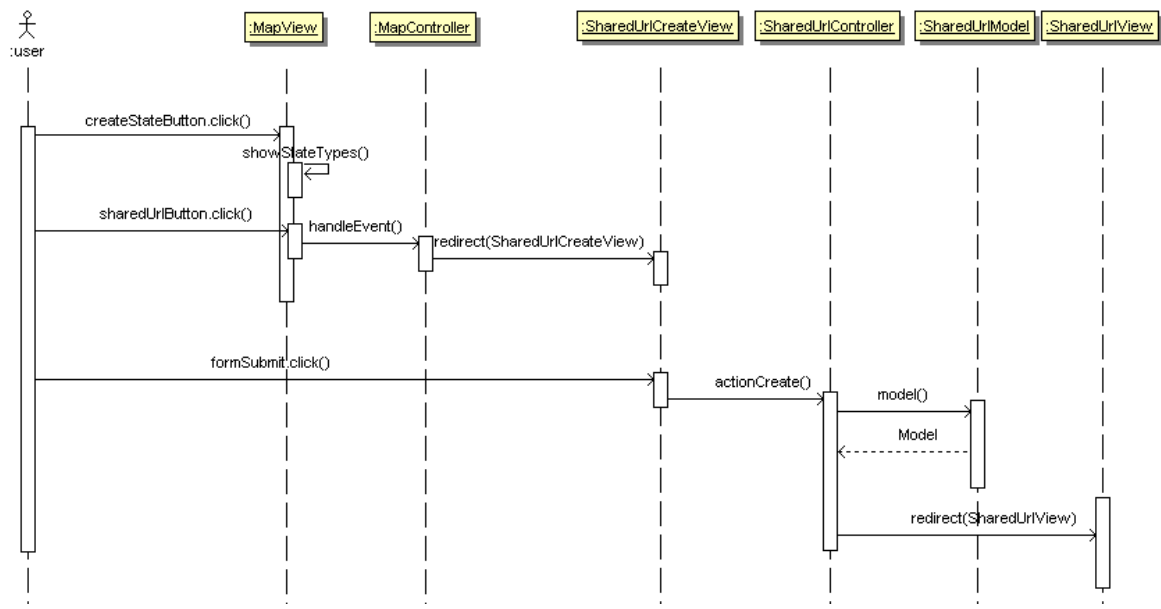


Fig. 45 Diagrama de secuencias para la creación de un estado Url

Capítulo 7 Posibles Mejoras y Trabajos futuros

Una vez obtenido el sistema resultante, analizado su desempeño, y evaluada su correcta cobertura de los objetivos planteados en un principio, se vuelve ineludible el análisis siguiente que nos permite vislumbrar alguna de las muchas posibles mejoras que podrían ser llevadas a cabo de aquí en más.

Las cualidades y funcionalidades que mencionaremos escapan al actual alcance del trabajo de grado, aun así nos parece apropiado dejar sentado lo que bien puede ser el puntapié inicial de algún otro trabajo, y aún también, como listado de “To-Do” en el caso de que en el futuro cercano, seamos nosotros mismos los que retomemos la implementación de PlaceOn.

Algunas características a ser tenidas en cuenta:

Crear más instancias de estados: Esto es, aprovechar la versatilidad y modificabilidad en la que hemos puesto hincapié, creando diversas nuevas instancias, por ejemplo permitiendo más formatos multimedia, o proveyendo estados con destinatarios prefijados, etc.

Crear categorías para los estados: Brindaría la oportunidad de clasificar que estados se quieren ver no solo por distancia, sino también por la categoría del mismo. De la mano con esta mejora, podrían definirse distintos tipos de alerta, permitiendo diferenciarlas.

Sugerencias de amigos: Se podría agregar funcionalidad de que la aplicación brinde sugerencias a los usuarios de a que personas o lugares podría seguir. Esto podría ser basado en cantidad de amigos en común o lugares que suele frecuentar una persona por ejemplo. Aquí entran en juego toneladas de ideas, teorías e implementaciones asociadas con el tratamiento de los datos que se acumulan, que constituyen el punto que nombraremos a continuación.

Uso de la información de la aplicación para minería de datos: Desde nuestro trabajo queremos hacer énfasis en la utilidad que tendría la minería de datos. Mediante el uso de dichas técnicas, sería posible lograr una identificación de patrones sobre los datos, los cuales se convierten en la herramienta fundamental para la realización de proyecciones, deducción de patrones de comportamiento de las personas en base a sus movimientos, permitiendo realizar predicciones y sugerencias de todo tipo. Usar los datos geográficos asociados con el resto de la información almacenada en la base de datos para la detección de diversos tipos de relaciones espaciales.

Definir una arquitectura RESTful: Actualmente la aplicación tiene una arquitectura soap. Adaptarla para proveer una interfaz de programación (API) RESTful, permitiría utilizar los mismos servicios para desarrollar aplicaciones nativas. Además, facilitaría la comunicación con aplicaciones paralelas (no clientes para el uso de PlaceOn) por ejemplo para analizar los datos de la base de datos, lo que allanaría el camino por ejemplo para integrar sistemas que provean las características que mencionamos en el punto anterior.

Capítulo 8 Conclusiones

A la hora de observar en retrospectiva el trabajo realizado, podemos obtener un panorama general de los resultados en varios puntos clave del proceso.

Como primera observación que destacaremos queremos nombrar el mero hecho de haber podido identificar lo que consideramos un problema o necesidad relacionada de alguna manera a un problema de software propiamente dicho, o de un problema cuya solución total o parcial puede ser acercada a través de un sistema informático. A partir de esa inquietud, y de manera informal, fue que devinieron las diversas funcionalidades que considerábamos importantes para dicha hipotética solución. Casi sin haber tomado conciencia cierta, y ayudados por las diversas herramientas y experiencias recabadas durante nuestro recorrido académico, nos vimos embarcados en el proceso de análisis funcional e incubación de un sistema.

Una vez llegado al punto de haber descrito en lenguaje natural un sistema y sus capacidades, llegamos a la segunda etapa que consideramos destacable. Con un panorama más o menos claro de lo que queríamos realizar, procedimos a formalizar las características deseadas en requerimientos, y buscamos alternativas arquitectónicas logrando con éxito una solución factible.

Como tercer y último escalón, fue necesario poner manos a la obra en la actividad de programación. Diversos problemas técnicos y desafíos fueron presentándose como es de esperar, los cuales podemos decir con satisfacción, que los hemos podido sortear valiéndonos de estrategias, investigación, programación conjunta en pares, etc.

Como conclusión general, el trabajo ha sido capaz de hacernos recorrer un amplio espectro de técnicas, herramientas, metodologías, teorías y recursos tecnológicos, y nos ha llenado de alegría haber podido cumplir nuestras propias expectativas, no sólo en los límites de PlaceOn, sino como futuros Ingenieros de Sistemas.

Bibliografía

- Cadavieco, J., Pascual Sevillano, M., & Madeira Ferreira, M. F. (2012). Realidad Aumentada, una evolución de las aplicaciones de los dispositivos móvil. *Revista de Medios y Educación*.
- Cangrejo Aljure, D. &. (2011). Minería de datos espaciales. *Revista Avances en Sistemas e Informática*, 8(3).
- CSS reference. (s.f.). Obtenido de <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
- Google Maps Javascript API V3 Reference. (s.f.). Obtenido de <https://developers.google.com/maps/documentation/javascript/reference?hl=es>
- HTML5 specification. (s.f.). Obtenido de <http://www.w3.org/TR/html5/>
- Javascript reference. (s.f.). Obtenido de <https://developer.mozilla.org/es/docs/Web/JavaScript>
- JQuery API Documentation. (s.f.). Obtenido de <http://api.jquery.com/>
- JQuery Mobile Documentation. (s.f.). Obtenido de <http://api.jquerymobile.com/>
- Lionbridge. (2012). *Mobile Web Apps vs. Mobile Native Apps: How to Make the Right Choice*. Obtenido de Lionbridge: http://www.lionbridge.com/files/2012/11/Lionbridge-WP_MobileApps2.pdf
- Manual de PHP. (s.f.). Obtenido de <http://php.net/manual/es/>
- Sui, D., & Goodchild, M. (2011). The convergence of GIS and social media: challenges for GIScience. *International Journal of Geographical Information Science*.
- Tang, L., & Liu, H. (2010). *Community detection and mining in social media*. San Rafael, California: Morgan & Claypool.
- Wasserman, S., & Faust, K. (1994). *Social network analysis methods and applications*. Cambridge: Cambridge University Press.
- Watts, D. (2006). *Seis grados de separación: La ciencia de las redes en la era del acceso*. Barcelona: Paidós.
- Yii Software LLC. (2012). *Yii Framework en Español*. Obtenido de <http://yiiframeworkenespanol.com/wiki/index.php>