FriendsMap

Bernat Bombí Fernández
Grau en Multimèdia

Enero 2012
César Tardáguila Moro

1. Resumen

En esta memoria se explica detalladamente el proceso de elaboración de una aplicación para iPhone y iPad.

Nombrada **FriendsMap**, ésta aplicación permite a los usuarios localizar a sus amigos gracias a la geolocalización.

Así pues, el lector de la memoria podrá conocer cada una de las fases necesarias para llevar a cabo el desarrollo de esta aplicación, ya que a lo largo del documento se detallan los procesos a seguir.

2. Índice

1. Resumen 2
2. Índice 3
3. Introducción 4
4. Objetivos 5
5. Planificación del proyecto 6
5.1. Presupuesto 6
5.2. Resumen de los pasos a seguir para el correcto desarrollo 7
6. Estructura de la información 9
6.1. Nombre de la aplicación 11
7. Diseño 13
7.1. Wireframes 13
7.2. Diseño gráfico 15
7.2.1. Diseño del icono 16
7.3. Diseño de la base de datos 17
8. Programación 18
8.1. Aplicación web 18
8.2. Aplicación iPhone y iPad 20
8.2.1. Notas generales 24
9. Test en dispositivos y usuarios 26
10. Resultados finales y conclusiones 27
10.1. Conclusiones 27
10.1.1. Mejoras futuras de la aplicación 28
11. Bibliografía 29

3. Introducción

En los últimos años los miembros de la sociedad han tendido cada vez más a interactuar entre ellos. El boom de las redes sociales como Facebook o Twitter ha potenciado esta interactuación, consiguiendo que invirtamos una gran parte de nuestro tiempo en las relaciones sociales enfocadas desde un punto de vista digital.

Más recientemente se ha puesto de moda el término "geolocalización". Foursquare, por ejemplo, -una red social con 15 millones de usuarios que se ubican en un mapa a base de hacer check-in's (anunciar en qué lugar están)- está cada día incrementando su número de usuarios. También otras redes sociales como Facebook o Twitter se han apuntado a esta moda: el primero creando Facebook Places y dando la opción de compartir la ubicación, y el segundo permitiendo indicar desde dónde se hace un tweet.

Vista esta tendencia, me planteé qué otra utilidad podía tener el hecho de estar localizado en un mapa. Entonces recordé aquella típica situación en la que estás esperando a tus amigos, impaciente, porqué no sabes dónde están y hace rato que deberían haber llegado.

La clave para resolver el problema es una aplicación para iPhone que me permita saber dónde están los amigos y que ellos puedan saber dónde estoy yo.

4. Objetivos

El objetivo principal de este proyecto es diseñar y desarrollar una herramienta en forma de aplicación que permita a los usuarios de dispositivos iOS (iPhone, iPad y iPod Touch) saber dónde están sus amigos.

Para conseguir este objetivo general se plantean otros secundarios, necesarios a su vez:

- Desarrollar una aplicación web en PHP, conectada a una base de datos, que permita hacer funcionar la aplicación iOS.

- Desarrollar la aplicación usando el lenguaje Objective-C, el nativo para las aplicaciones iOS y MAC OS, consiguiendo así un mayor rendimiento. Además, de esta forma, se acata la obligatoriedad impuesta por Apple de utilizar sus aplicaciones para el desarrollo, obteniendo así la posibilidad de distribuir la aplicación.

- Diseñar la aplicación siguiendo la GUI (Graphical User Interface) predefinida por Apple, adaptándose a los estándares.

    - Adaptar el diseño o resolución de la aplicación a cada dispositivo, es decir, con un diseño diferente para iPhone que para iPad, o iPhone y iPhone con Retina Display (éste último dispone de mejor pantalla con el doble de resolución).

Finalmente, a nivel personal, me planteo dos objetivos:

- Mejorar mis conocimientos de programación en Objective-C.

- Desarrollar una aplicación desde cero para conocer bien cómo funcionaría un proceso real.

5. Planificación del proyecto

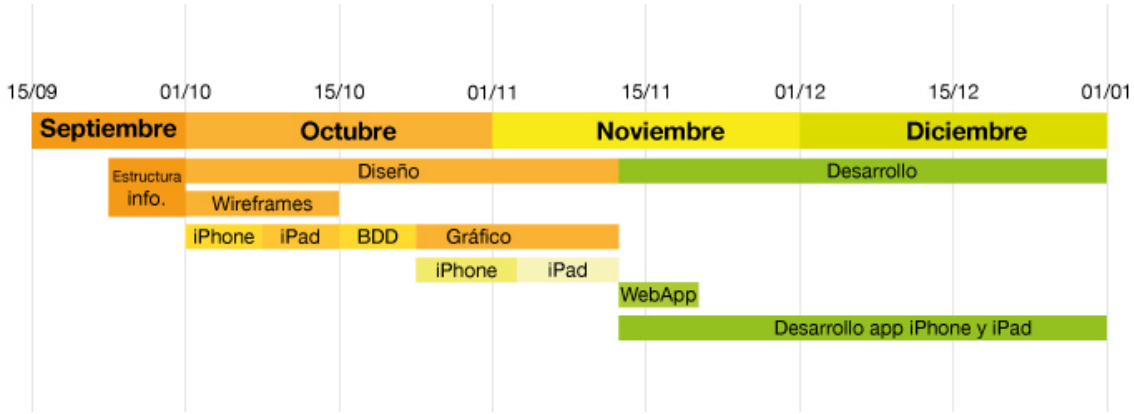
El correcto desarrollo de la aplicación cuenta con distintas fases. A continuación se listan y en el punto 5.2. se detallan.

Estructura de la información. Organización de los contenidos.

Diseño. Diseño wireframes, diseño de la base de datos y diseño gráfico de la aplicación.

Desarrollo. Desarrollo del programa web y de la aplicación.

El calendario se organiza de la siguiente manera:



Como se puede apreciar en el gráfico, en los últimos días de septiembre se programará organizar la información.

Ya en octubre, se diseñarán los wireframes y se empezará con el diseño gráfico y el de la base de datos. La fase durará hasta mediados de noviembre.

Una vez acabada la fase de diseño se empezará, de forma paralela, con el desarrollo de la aplicación y del programa web, que durará hasta finales de diciembre.

5.1. Presupuesto

A continuación se presenta el presupuesto para un cliente que encargase la creación de la aplicación.

Concepto	Precio
Estructura de la información	**300 €**
Diseño • **Diseño Wireframes** • **Diseño gráfico**	**800 €**
Desarrollo • **Diseño y desarrollo BDD** • **Desarrollo programa web** • **Desarrollo aplicación**	**2.000 €** 300 € 600 € 1.100 €
Alquilar servidor web + dominio	**80 €/año**
Registro iOS Developer Program	**79 €/año**
Total	**3.259 €**

El presupuesto está desglosado por fases, haciendo visible el coste de cada una de ellas.

Como se puede ver, el apartado más costoso es el de desarrollo (2.000€), puesto que es el apartado que más recursos temporales requiere.

Se ha incorporado el precio del alquiler del servidor donde alojar el programa web y el precio del registro en el iOS Developer Program.

Es importante comentar que, en el caso real -durante el desarrollo del Trabajo de Final de Grado-, los costes han sido nulos, ya que las tres primeras fases sólo requieren un coste temporal puesto que se dispone de un servidor web y un dominio, y de un registro en el iOS Developer Program.

5.2. Resumen de los pasos a seguir para el correcto desarrollo

A continuación se resumen las distintas fases a seguir para el correcto desarrollo de la aplicación.

Estructura de la información

La primera fase del proyecto radica en la organización de la información que debe contener la aplicación. Es importante tener en cuenta las necesidades del usuario.

También es necesario organizar las secciones de la aplicación de forma adecuada.

Finalmente, en esta fase también se decide un nombre para la aplicación.

Diseño

Antes de empezar con el diseño gráfico conviene trazar, mediante wireframes, la estructura básica de la aplicación para saber como será.

En esta fase, inicialmente, se diseñan wireframes para iPhone y iPad.

Una vez acabados y revisados los wireframes se procede al diseño gráfico de la aplicación, con todos los detalles necesarios a nivel visual.

Finalmente se diseña la base de datos de la aplicación web.

Desarrollo

De forma paralela se desarrollan la aplicación web que permite a la aplicación funcionar y la propia aplicación.

El programa web se crea en PHP (Hypertext Preprocessor), lenguaje de programación web y la base de datos se construye en SQL.

Por otra parte la aplicación para iPhone y iPad se desarrolla en Objective-C y con el programa Xcode.

6. Estructura de la información

Como se ha comentado previamente, en esta fase del proyecto se analiza la forma en que la información está organizada en la aplicación.

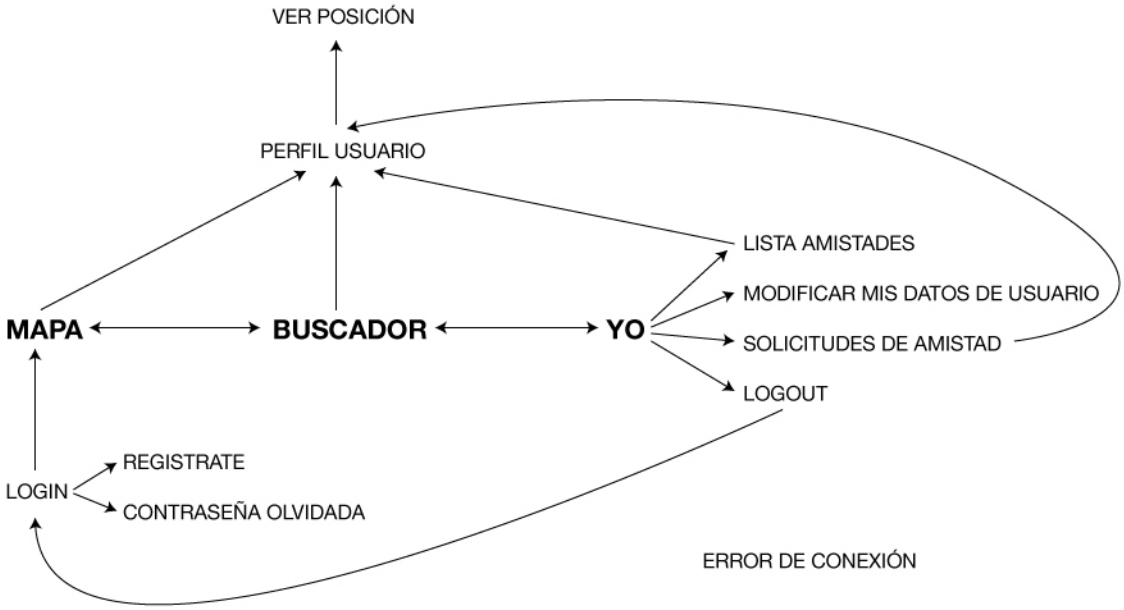
Así pues, para poder saber qué información encontrar al acceder a la aplicación se analizaron los apartados más necesarios.

- **Mapa**. La sección más importante contiene un mapa dónde ver la posición de los amigos.

- **Buscador.** Para poder buscar personas que están registradas en la aplicación.

- **Yo**. Para encontrar la información del propio usuario.

Una vez establecidas las secciones principales se desarrolló el árbol de contenidos de la aplicación. También se tuvo en cuenta que para poder acceder a la aplicación es necesario estar registrado.



A continuación se explica qué nos permite hacer cada uno de los apartados:

Login

El usuario puede acceder a la aplicación introduciendo su correo electrónico y su contraseña.

Si no está registrado puede ir a la página de registro. Si ha perdido la contraseña puede generar una nueva.

Regístrate

Formulario que permite al usuario crear una nueva cuenta. Debe introducir sus datos personales y una contraseña.

Contraseña olvidada

Formulario que permite al usuario generar una nueva contraseña introduciendo su e-mail. Una vez generada se guarda la nueva en la base de datos y se envía vía e-mail.

Mapa

Mapa donde aparecen indicadas las posiciones de los amigos y la del propio usuario. Al pulsar sobre un pin (que indica la posición de un amigo) se puede acceder al perfil del amigo en cuestión.

Buscador

Permite buscar a personas que están registradas en la aplicación. Una vez encontrados los resultados se puede acceder a su perfil de usuario.

Yo

Muestra la información del usuario de la aplicación, así como su lista de amistades, la posibilidad de modificar sus datos de usuario, ver las solicitudes de amistad o cerrar sesión.

Lista amistades

Proporciona una lista de las amistades del usuario de la aplicación. Al pulsar sobre el nombre de un usuario se puede acceder a su perfil.

Perfil usuario

Es uno de los apartados más importantes. Permite ver información de un usuario. Dependiendo de la relación entre el usuario de la aplicación y éste, se mostrarán unos datos u otros. En caso que no sean amigos, se podrá ver su nombre y ciudad.

También se muestra un botón que permite: solicitar amistad, aceptar amistad o nada (si ya se ha enviado la solicitud).

En caso que sean amigos, se mostrarán la dirección de correo del usuario, el nombre, ciudad y un botón para ver su posición en un mapa.

También aparece un botón que permitirá eliminar la amistad.

Ver posición

Mapa donde se muestra la posición de un usuario en concreto.

Modificar mis datos de usuario

Permite al usuario de la aplicación modificar sus datos de usuario (nombre, ciudad, e-mail y contraseña).

Solicitudes de amistad

Muestra las solicitudes de amistad que tiene el usuario de la aplicación.

Error de conexión

Al ser necesario Internet para el correcto funcionamiento de la aplicación, en caso de no haber conexión se le mostrará al usuario una ventana indicándolo.

6.1. Nombre de la aplicación

En la fase de organización también se buscó un nombre apropiado para la aplicación.

Después de un brainstorming, los conceptos más repetidos hacían referencia a los amigos y a la localización.

De las mejores propuestas, la más adecuada hacía hincapié en los conceptos fundamentales, así que se decidió nombrar a la aplicación **FriendsMap**.

7. Diseño

La fase de diseño se divide en 3 subapartados: wireframes, diseño gráfico y diseño de la base de datos.

7.1. Wireframes

Un wireframe es una guía visual que representa el esqueleto de una aplicación.

Es importante crear wireframes para ver la representación gráfica de cómo será la aplicación.

Así pues, teniendo en cuenta la organización de la aplicación creada en la anterior fase, se crearon wireframes de las principales páginas de la aplicación.

También se tuvo presente la GUI (Graphical User Interface) predefinida por Apple para el diseño de aplicaciones.

Para facilitar la navegación a los usuarios se decidió usar un Tab Bar, uno de los elementos más populares para navegar en las aplicaciones.

Según las Human Interface Guidelines de Apple, los Tab Bar deben ser usados para dar acceso a distintas tareas relacionadas con el total de la aplicación. Así pues, Apple indica que es una buena forma de organizar la información de forma jerárquica y proveer a los usuarios de varias categorías de información o modos a la vez.

Los Tab Bar se encuentran en la parte inferior de la pantalla y deben ser accesibles desde cualquier parte de la aplicación. Cada unos de los tabs que forman un Tab Bar muestran texto, un icono y tienen la misma anchura.

Como se puede ver en la imagen del ejemplo que se muestra a continuación, cuando accedemos a una sección el tab correspondiente muestra un fondo iluminado, indicando que es el seleccionado en aquel momento.

Ejemplo de Tab Bar con el tab "Top 25" seleccionado:

En cada una de las tres principales secciones se usaron Views, TableViews, NavigationControllers y MapViews.

View: gestiona una área rectangular en la ventana de la aplicación. En ella se puede escribir contenido, controlar movimientos táctiles y manejar la interfaz de subviews. También se pueden añadir imágenes, texto y formas dibujadas, entre otros.

Se puede pensar en un view como un bloque para la construcción de la interfaz. Normalmente se usa más de un view, construyendo así una jerarquía de views, cada uno adaptado al tipo de contenido a mostrar.

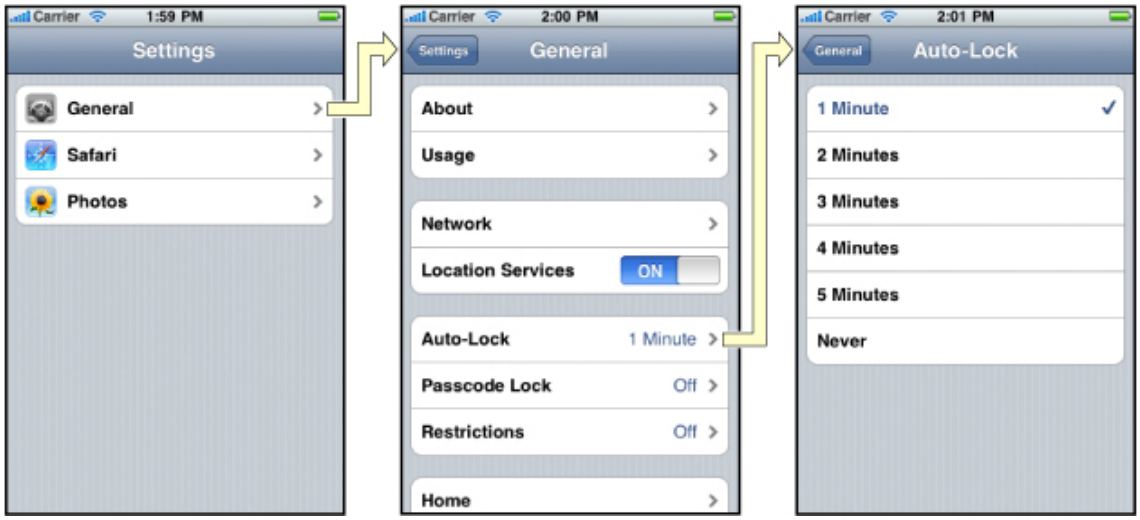
TableView: presenta datos en una lista con múltiples filas. Estos datos pueden estar divididos en secciones.

Sus propósitos generales son:

- Permitir a los usuarios navegar por una estructura jerárquica de datos.
- Mostrar una lista indexada de ítems.
- Visualizar información detallada y controles en grupos visualmente distintos.
- Presentar una lista seleccionable de opciones.

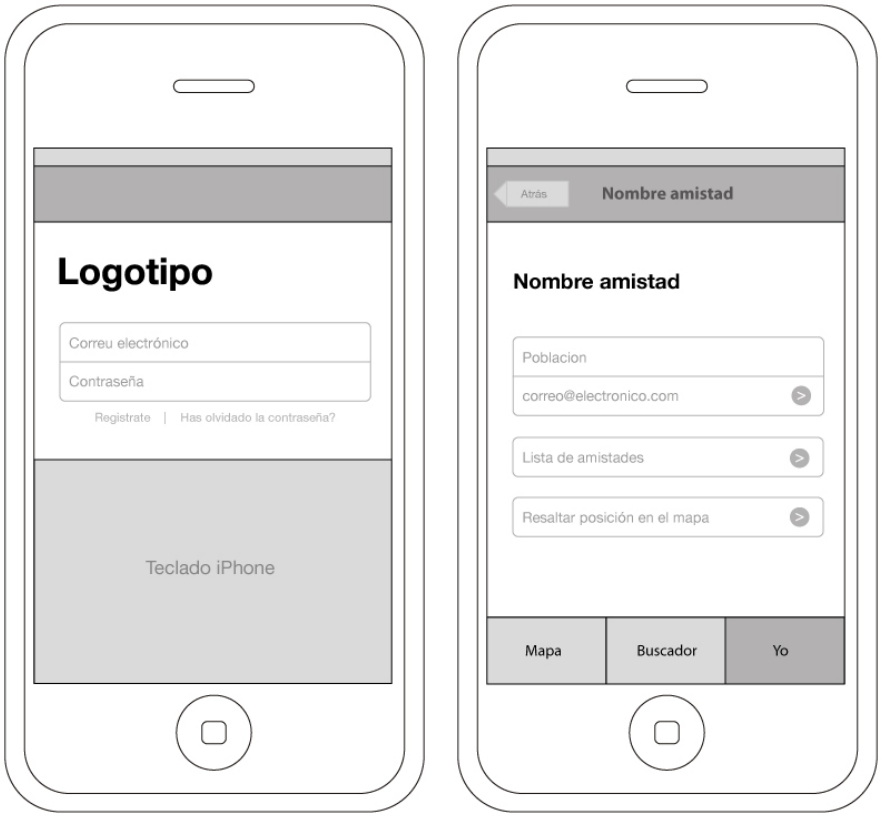
NavigationController: implementa un view especializado que maneja la navegación jerárquica de contenido. Se utiliza de forma que la interfaz refleja la naturaleza jerárquica del contenido que se presenta. Junto al NavigationController se suele presentar una barra de navegación que permite al usuario volver al View anterior.

Ejemplo de interfaz con NavigationController:



MapView: proporciona una interfaz con un mapa, similar al que aparece en la aplicación Mapas. El mapa se puede centrar en la posición elegida, se pueden modificar sus proporciones y se puede anotar distinta información.

A continuación se presenta una muestra de los wireframes creados, en este caso para iPhone.

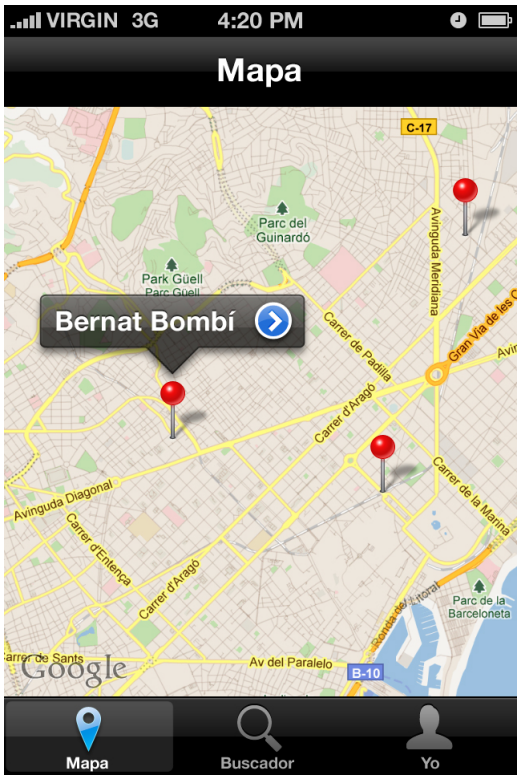


(se pueden encontrar el resto de los wireframes, tanto los de iPhone como los de iPad, en el Anexo 1)

7.2. Diseño gráfico

Usando los elementos gráficos de la GUI se procedió al diseño gráfico, dotando a la aplicación de una estética adaptada a esa interfaz.

A continuación se presenta una muestra de los diseños creados que, como se puede observar, siguieron las estructuras y elementos usados en los wireframes.



(Se pueden encontrar el resto de los diseños, tanto los de iPhone como los de iPad, en el Anexo 2)

7.2.1. Diseño del icono

Durante la fase de diseño también se creó el icono de la aplicación.

Con el objetivo de destacar, se buscaron colores apropiados para crear un icono llamativo.

7.3. Diseño de la base de datos

Uno de los aspectos más importantes para el buen funcionamiento de la aplicación es diseñar una base de datos funcional y escalable, que permita almacenar los datos que se generan al navegar por la aplicación.

Después de analizar las necesidades de almacenamiento de datos, la estructura de la base de datos fue la siguiente:

user
id
mail
psw
name
surname
city
sign_up_date
date

locations
id
id_user
long
lat
date

relations
id
id_user1
id_user2
status

En la tabla **user** se guardan los datos de usuario así como la fecha de la última vez que ha sido localizado por la aplicación. Esto nos permite mostrar en el mapa las últimas amistades ubicadas.

En la tabla **locations** se almacenan las ubicaciones de los usuarios. Cada vez que un usuario es localizado se crea una fila nueva, indicando la id del usuario, la longitud, la latitud y la fecha y hora.

La base de datos podría desarrollarse de tal forma que cada localización de un usuario sobrescriba la anterior del mismo, pero se considera interesante guardar la información de las distintas localizaciones.

En la tercera tabla, **relations**, se guardan las relaciones de amistad entre los usuarios. En **id_user1** guardamos la id del usuario que envía la solicitud de amistad. En **id_user2** la id del usuario que recibe la solicitud de amistad. Finalmente, si **status=0** quiere decir que el usuario no ha aceptado aún la solicitud. Si **status=1** los usuarios son amigos.

Si no son amigos, la fila no existe y, si ha existido previamente, será borrada.

8. Programación

Durante la fase de programación se desarrollan de forma paralela la aplicación web que nos servirá para hacer funcionar la aplicación y la aplicación para iPhone y iPad.

8.1. Aplicación web

La mayoría de secciones de la aplicación requieren recibir datos desde un servidor. Para que esto sea posible se desarrollaron una serie de ficheros que permiten a la aplicación recoger dichos datos.

Antes de empezar con la programación en PHP se creó la base de datos, diseñada previamente, y se subió a un servidor web.

(Se puede encontrar el SQL correspondiente a la creación de la base de datos en el Anexo 3)

Una vez creada la base de datos se desarrollaron 3 formularios: uno para registrarse en la aplicación, otro para recordar la contraseña y un tercero para cambiar los datos de usuario.

Estos formularios están enlazados a sus correspondientes archivos PHP que guardan la información introducida en la base de datos. Para hacer operaciones simples con la base de datos de forma fácil y rápida se decidió usar ezSQL, una herramienta ideal para estos casos.

sign_up.php
Muestra un formulario donde introducir los datos personales del usuario.

do_sign_up.php
Guarda los datos provenientes de **sign_up.php** en la base de datos.

remember.php
Muestra un formulario donde introducir el e-mail del usuario.

do_remember.php
Comprueba el e-mail introducido en **remember.php** y, si existe, genera una nueva contraseña que es enviada por e-mail a la correspondiente dirección.

change_data.php
Muestra un formulario con los datos del usuario de la aplicación. Permite modificarlos y enviar el formulario.

do_change_data.php
Recibe los datos de **change_data.php** y, después de comprobar que son correctos, procede a modificar el registro de la base de datos.

También se crearon otros archivos:

do_location.php
Guarda la posición de un usuario en la base de datos.

do_log_in.php
Comprueba el e-mail y la contraseña de un usuario, dejándolo acceder o no a la aplicación.

do_relations.php
Cambia el tipo de relación entre dos usuarios.

También se programaron un tipo de archivos cuya función es devolver datos.

Algunos de ellos devuelven un array, que es una colección ordenada de objetos. Se ha utilizado en distintos puntos de la aplicación dada su facilidad de uso.

En otras partes de la aplicación, como en los mapas, se ha usado un dictionary, que maneja una asociación entre claves y valores. Es útil para recibir listas de información no ordenada.

buscador.php
Devuelve un array con los resultados de una búsqueda.

lista_amigos.php
Devuelve un array con la lista de amigos de un usuario.

lista_lastPos.php
Devuelve un dictionary con los datos y la posición de un usuario en concreto.

lista_loc.php
Devuelve un dictionary con los datos y la posición de los amigos del usuario de la aplicación.

lista_solicitudes.php
Devuelve un array con la lista de solicitudes de amistad que tiene el usuario de la aplicación.

yo.php
Devuelve un array con los datos del usuario que accede a la aplicación.

you.php
Devuelve un array con los datos de un usuario de la aplicación. Por ejemplo, al acceder al perfil de un amigo.

Todos estos archivos fueron colgados en

http://friendsmap.bombi.cat

y la aplicación, como se comenta en el próximo apartado, los usa constantemente.

(Se pueden encontrar todos los archivos en el Anexo 4)

8.2. Aplicación iPhone y iPad

La estructura de clases de la aplicación se corresponde con la estructura diseñada en la fase "Estructura de la información".

A continuación se detalla qué función tiene cada una de las clases creadas:

LoginViewController

Presenta 2 UITextField, que son objetos que muestran texto editable, donde se introduce el e-mail y la contraseña, y un botón enlazado a un IBAction, nombre que reciben los métodos que son activados a partir de una acción especificada en la interfaz (como por ejemplo, pulsar un botón).

Preguntado el servidor mediante el archivo **do_log_in.php**, se comprueba si los datos son correctos.

En caso que no lo sean no se deja acceder a la aplicación.

Por otra parte, si el e-mail y la contraseña son correctos, mediante NSUserDefaults (interfaz que nos deja guardar las preferencias de usuario) se

guardan los datos del usuario (id, nombre, ciudad y e-mail), hecho que es equivalente al inicio de sesión.

Una vez guardados se cierra el LoginViewController y se muestra **MapViewController**.

LoginViewController también incorpora 2 botones (para registrarse en la aplicación o recordar la contraseña) que abren Safari con una de las páginas correspondientes (friendsmap.bombi.cat/sign_up.php o friendsmap.bombi.cat/remember.php).

MapNavController

Es un UINavigationController que nos permite una navegación de contenido jerárquica en el tab correspondiente al Mapa.

MapViewController

En primer lugar comprueba que un usuario haya iniciado sesión. Si no es así, presenta **LoginViewController**.

Si se ha iniciado sesión, muestra un mapa con los pins correspondientes a la posición de los amigos del usuario de la aplicación.

Los pins son recibidos del servidor mediante el archivo **lista_loc.php**, pasando una variable $_GET con la Id del usuario de la aplicación.

Al pulsar sobre un pin nos permite identificar el nombre del amigo y navegar hasta su perfil (**YouViewController**).

También dispone de un botón para recargar los datos.

SearchNavController

Es un UINavigationController que nos permite una navegación de contenido jerárquica en el tab correspondiente al Buscador.

SearchTableViewController

Es un TableView que contiene los datos de la búsqueda que se realiza. Para hacer la búsqueda se dispone de un UISearchBar, que permite introducir texto.

Al pulsar el botón "Buscar" del teclado se hace una petición al servidor mediante el archivo **buscador.php**, pasando una variable $_GET con el texto a buscar en formato UTF-8.

MeNavController

Es un UINavigationController que nos permite una navegación de contenido jerárquica en el tab correspondiente a Yo.

MeTableViewController

Muestra un tableView de tipo agrupado con los datos del usuario de la aplicación. Los datos los recoge de NSUserDefaults.

También muestra filas que permiten visualizar la lista de amistades (**FriendsListViewController**), modificar los datos de usuario (**WebViewController**) y ver solicitudes de amistad (**FriendsRequestsViewController**).

Finalmente añade un botón para cerrar sesión.

YouViewController

Muestra un tableView de tipo agrupado con los datos de un usuario, generalmente un amigo del usuario de la aplicación. Estos datos son recogidos mediante el archivo **you.php**, pasando dos variables: la id del usuario de la aplicación y la id del usuario del que queremos saber la información.

Dependiendo de la relación que haya entre los dos nos mostrará unos datos u otros.

Si no son amigos, mostrará el nombre y la ciudad. Además, dependiendo de la relación, se visualizará un botón que permite: solicitar amistad, aceptar solicitud de amistad o indicar que ya se ha enviado la solicitud.

Si ya son amigos, aparte del nombre y ciudad, también mostrará el correo electrónico y la posibilidad de abrir **DetailMapViewController**, que nos muestra la posición del usuario. En este caso, el botón comentado anteriormente permitirá eliminar la amistad.

DetailMapViewController

Muestra un mapa que nos indica la posición de un usuario. Se hace recogiendo datos del fichero **lista_lastPos.php**.

FriendsListViewController

Presenta un tableView con la lista de amigos del usuario de la aplicación. Recoge los datos mediante el archivo **lista_amigos.php**.

Al pulsar sobre una fila se dirige a **YouViewController**, cargando los datos del usuario deseado.

FriendsRequestsViewController

Muestra un tableView con la lista de solicitudes de amistad que tiene el usuario de la aplicación. Recoge los datos a través del archivo **lista_solicitudes.php**.

Al pulsar sobre una fila se dirige a **YouViewController**, cargando los datos del usuario elegido y permitiendo, gracias al botón inferior, aceptar la solicitud de amistad.

WebViewController

Muestra un webView con una página web. La página web se carga mediante un NSURL, permitiendo que el webView sea usado para cargar distintas URLs.

Reachability

Es una clase que permite conocer el estado de la conexión a Internet.

ErrorViewController

Es un View que aparece cuando no hay conexión a Internet. Incorpora un botón que al ser pulsado comprueba si hay Internet o no. Si hay conexión desaparece él mismo.

AppDelegate

El AppDelegate declara métodos que son implementados por la aplicación. Estos métodos proporcionan información sobre eventos clave durante la ejecución de la aplicación, como por ejemplo: cuándo acaba de iniciarse,

cuándo termina, cuándo no hay suficiente memoria disponible o cuándo suceden cambios importantes. Además, en este caso, se utiliza para otras funciones:

Por una parte controla todo lo relacionado con la localización del usuario:

Al iniciar la aplicación envía al servidor la posición del usuario. Posteriormente, cuando el usuario se mueve 500 metros, comprueba el tiempo transcurrido desde la última actualización en el servidor. Si hace más de 120 segundos actualiza la base de datos con la nueva ubicación.

Los parámetros de actualización de la posición fueron optimizados para que el dispositivo no consumiese la batería en exceso. Se han elegido 500 metros considerando que es una medida suficiente para conocer una ubicación y 120 segundos para situaciones donde el terminal se encuentre dentro de un vehículo en movimiento.

Por otra parte, se crea un método BOOL que nos permite saber si hay o no conexión a Internet, preguntando a la clase **Reachability**.

8.2.1. Notas generales

En todos los controladores se comprueba, antes de realizar cualquier otra operación, que haya conexión a Internet. Si no la hay, presenta **ErrorViewController.**

Todas las solicitudes al servidor se hacen de forma asíncrona, lo que hace que la aplicación no quede bloqueada mientras se cargan datos.

Las imágenes que se usan en la aplicación están adaptadas a Retina Display, nombre con el que se apoda la pantalla LED de que disponen los iPhone 4 y iPhone 4S, que cuenta con el doble de resolución que la de las versiones previas de iPhone.

La resolución de la Retina Display es de 960x640 píxeles, mientras que en la no-Retina Display es de 480x320 píxeles.

La aplicación funciona en modo vertical en iPhone y iPod Touch, mientras que lo hace en modo horizontal en iPad.

Se usan dos storyboards -archivos que permiten diseñar el flujo de trabajo de la aplicación de forma visual-, uno para iPhone y otro para iPad. Dependiendo del dispositivo en uso se selecciona uno u otro para la aplicación.

Los iconos y splashes -imágenes que aparecen al iniciar la aplicación- están adaptados a cada dispositivo.

9. Test en dispositivos y usuarios

Una vez la aplicación estuvo programada y funcional, se instaló en distintos dispositivos. Concretamente se instaló en un iPhone 4S, dos iPhone 4 y un iPad 2.

Estos dispositivos fueron entregados a sus respectivos dueños con el objetivo de aportar mejoras para la aplicación.

Al cabo de una semana se organizó una reunión con ellos para que cada uno expusiese los fallos que había encontrado o propusiera mejoras.

Después de anotar los comentarios hechos por los usuarios se analizaron y algunos de ellos fueron aplicados.

Ésta es la lista de cambios propuestos por los usuarios que probaron la aplicación y fueron implementados:

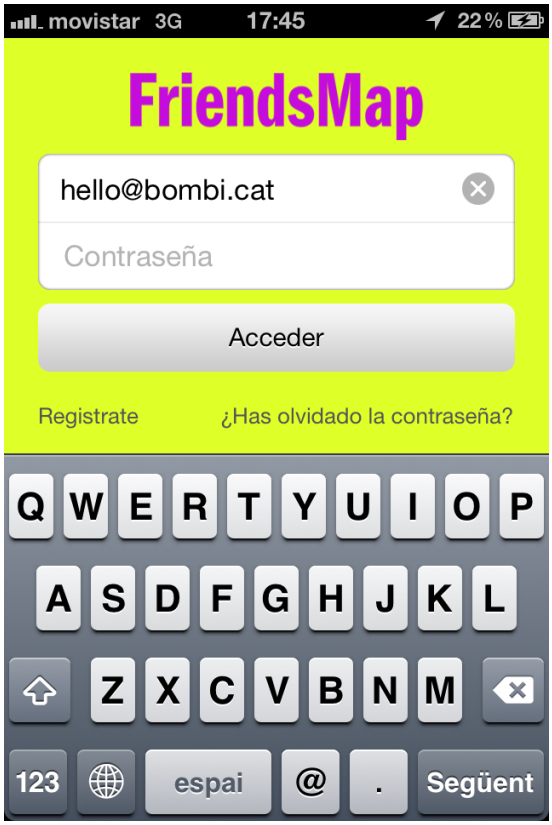
- En el MapViewController se implementó un botón en la barra de navegación para recargar los pins.

- Se cambió la estética de dos botones: el de cerrar sesión y el que se muestra en la parte inferior de YouViewController. El cambio ha supuesto una simplificación de los botones, eliminando una sombra interior.

- Todos los tableView en el iPad pasaron a tener una anchura proporcional a la anchura del iPad.

Así pues, la aplicación final incorpora estos cambios aplicados a la interfaz y funcionalidad original.

10. Resultados finales y conclusiones



Analizando los resultados finales, podemos ver que la aplicación ya acabada es plenamente funcional.

Está adaptada a todos los dispositivos deseados y realiza las funciones propuestas en el inicio del proyecto.

Los datos generados son guardados de forma correcta en la base de datos y posteriormente recogidos sin ningún error.

La interfaz diseñada durante la creación de los wireframes y posteriormente durante la fase de diseño gráfico ha sido aplicada de forma eficiente, añadiendo sólo algunos pequeños cambios como se comenta en el apartado anterior.

(Se pueden encontrar capturas de pantalla de iPhone y iPad con los resultados finales en el Anexo 5)

10.1. Conclusiones

Vistos los resultados finales, las conclusiones son positivas:

Por una parte se ha desarrollado la aplicación web en PHP conectada a la base de datos y es plenamente funcional. El objetivo se ha cumplido.

Por otra, la aplicación ha sido diseñada siguiendo la Graphical User Interface de Apple y adaptando la interfaz a los distintos dispositivos y pantallas. El objetivo se ha cumplido.

La aplicación ha sido programada en Objective-C, el lenguaje nativo de las aplicaciones de Apple. El objetivo se ha cumplido.

En cuanto a los objetivos de carácter personal:

He conseguido mejorar mucho mis conocimientos de Objective-C. Una experiencia real es la clave para saber cómo resolver distintas situaciones que se plantean a nivel de programación.

También estoy muy satisfecho por el hecho de haber vivido el proceso de elaboración de una aplicación desde su inicio hasta su final, pasando por todas las fases.

Así pues, la experiencia durante la realización de este Proyecto de Final de Grado ha sido enriquecedora: he podido aplicar los conocimientos adquiridos a lo largo del Grado en un producto real, permitiéndome experimentar desde dentro cómo se trabaja en el sector de las aplicaciones.

10.1.1. Mejoras futuras de la aplicación

En un futuro, y siempre que sea posible, se plantea mejorar la aplicación para hacerla más funcional.

A continuación se listan algunos de las mejoras a realizar:

- Crear un perfil de usuario más completo, con número de teléfono, fotografía. Añadir la posibilidad de guardar el perfil de usuario como contacto.

- Añadir la funcionalidad de compartir la posición en redes sociales.

- Mejorar pins del mapa, haciendo que en vez de aparecer un pin de color rojo aparezca la fotografía del usuario.

- Organizar amigos por grupos, permitiendo asignar horarios personalizados de localización. Añadir la posibilidad de bloquear usuarios.

- Permitir al usuario elegir cada cuántos metros ser rastreado.

11. Bibliografía

Mark, D.; LaMarche, J. (2009). Beginning iPhone Development: Exploring the iPhone SDK. New York: Springer-Verlag New York, Inc.

Apple Inc. (2012). iOS Developer Library. *Apple Developer* [documentación en línea]. [Fecha de consulta: noviembre - diciembre 2011].

<http://developer.apple.com/library/ios>

Apple Inc. (2012). iOS App Programming. *Apple Developer* [documentación en línea]. [Fecha de consulta: noviembre - diciembre 2011].

<http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/>

Apple Inc. (2012). iOS Human Interface Guidelines. *Apple Developer* [documentación en línea]. [Fecha de consulta: noviembre - diciembre 2011].

<http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006556-CH1-SW1>

FriendsMap