ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

"NUEVA RED SOCIAL CON INTEGRACION DE SERVICIOS BASADOS EN LA LOCALIZACION"

Javier Marquina Salinas

Jesús Villadangos

Pamplona, 15-4-2014

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

INDICE

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1.- INTRODUCCION

Mi proyecto se basa en una nueva Red social con integración de servicios basados en la localización. La idea original era un mashup que obtiene información de diferentes redes sociales pero mi proyecto se va a centrar en la información obtenida de la red social Facebook.

Este proyecto es una idea original de David Azcona y Niels McEvoy. En la cual pretendían unir diferentes redes sociales con acceso a información con localización y colocar esa información en un mapa de Google Maps.

En desarrollo Web, una **mashup** es una aplicación que usa y combina contenido de más de una fuente, para crear un nuevo servicio simple, visualizado en una única interfaz gráfica.

El contenido usado en mashups es típicamente obtenido de otra fuente vía una interfaz pública o API.

Mi proyecto al final se concretó en un mashup que integra la información de Facebook en el mapa de Google Maps , dejando para más adelante la implementación de otras redes sociales como Twitter, LinkedIn , FourSquare, etc…

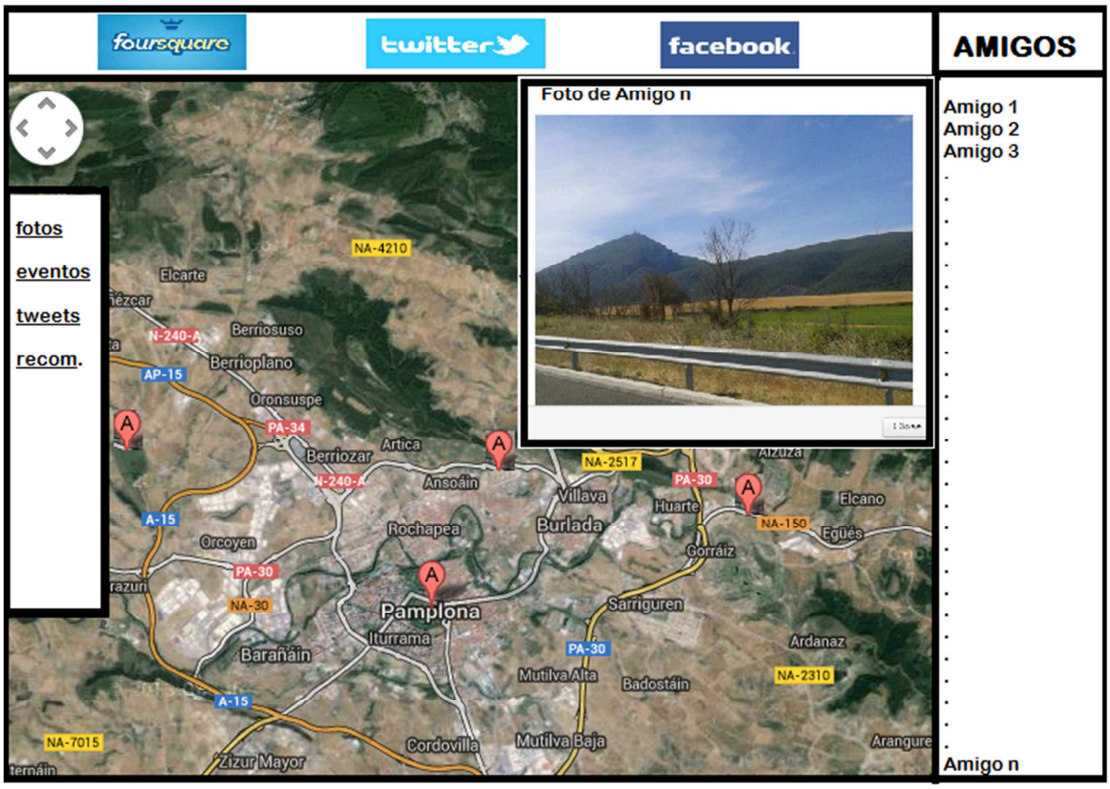


Figura 1 – Idea del interfaz del proyecto

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Sistemas similares

A continuación voy a comentar algunos sitios Web con puntos en común o similares a mi proyecto.

- **Tagtagcity**

TagtagCity es un sitio web en el cual recomiendan los lugares más importantes de una ciudad con marcadores en un mapa .La idea general es parecida a la de mi proyecto pero yo voy a sacar la información del API de Facebook y en Tagtagcity obtienen la información de una base de datos.

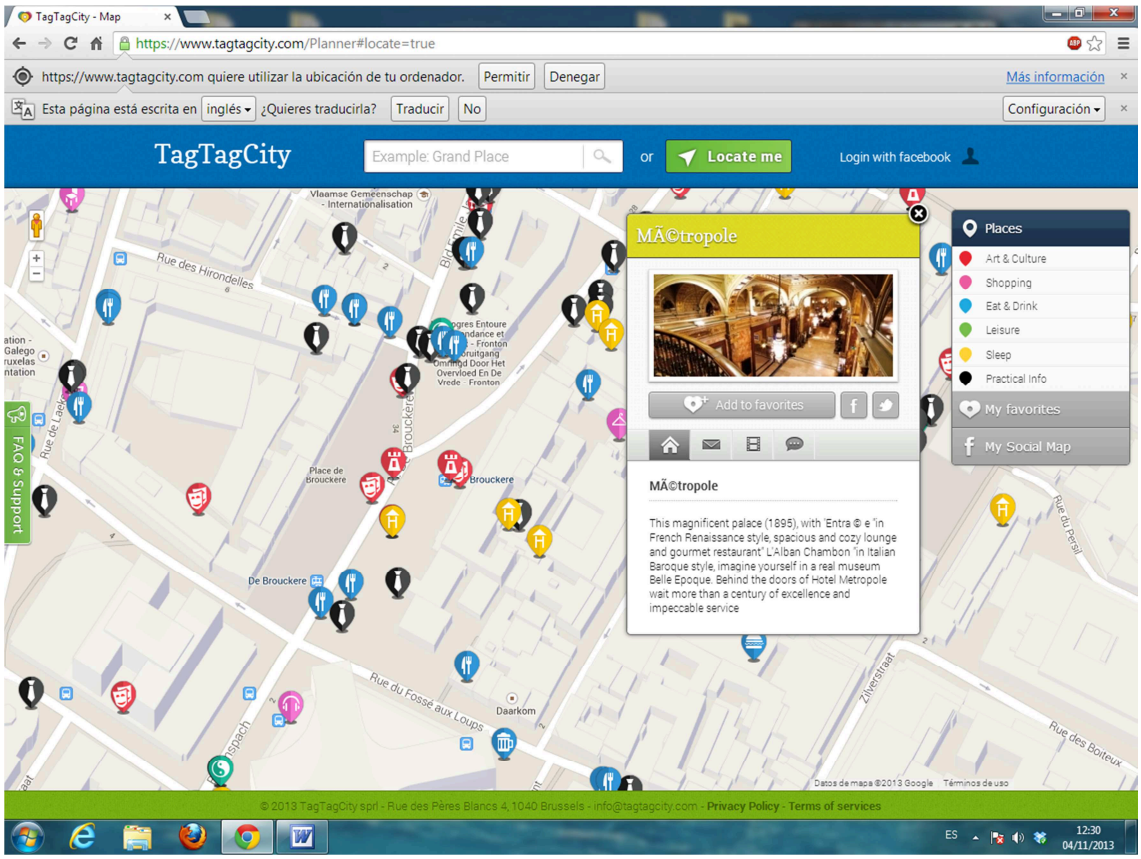


Figura 2 – Imagen de la web TagtagCity

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- **Facebook Nearby**

Nearby es un servicio de recomendaciones personalizadas de acuerdo con tu ubicación, una forma de crear un contexto personal de acuerdo con las coordenadas de los usuarios, basándose además en recomendaciones y consejos, entre otras cosas desde el lado de los usuarios.

Básicamente, se trata de una aplicación para descubrimiento de negocios locales, que podemos encontrar en las aplicaciones móviles de Android e iOS.

Esta funcionalidad ayuda a los usuarios a descubrir bares, centros de Shopping, restaurantes y más, basándose en los Likes, check-ins, y recomendaciones que hayan hecho sus amigos.

Se puede buscar por determinadas categorías, como bares, hoteles, etc, pero además ver un consolidado, y hacer una búsqueda completamente independiente.

Cada una de las entradas Nearby muestra el nombre del local, su categoría, una foto de perfil, la dirección, la distancia desde donde estamos, un *rating* hecho por los usuarios, y los mejores resultados de acuerdo con el área donde estemos.

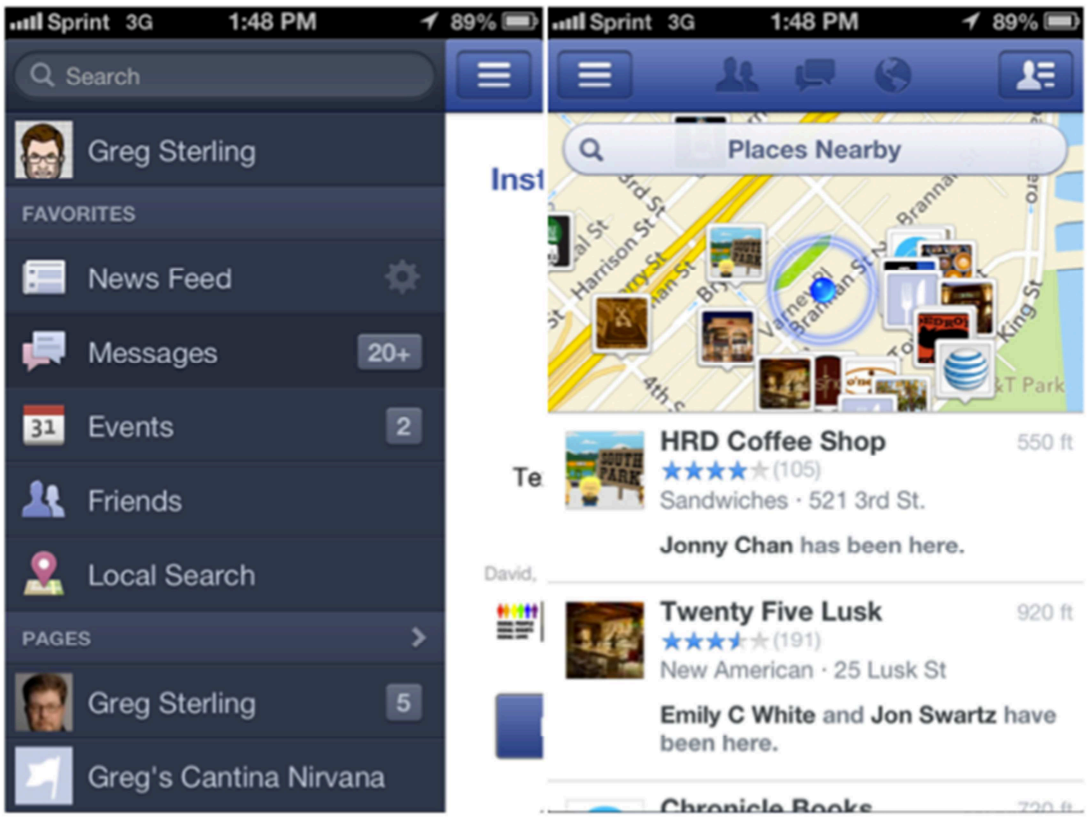


Figura 3 – Imagen de la aplicación de Facebook Nearby

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- **Yelp**

Yelp es una guía local formada por las opiniones de los usuarios. La diferencia con otros sistemas similares es que está orientado a los locales y las valoraciones que hacen los usuarios de ellos.

Se centra en negocios locales que quieren darse a conocer de una manera gratuita.

En ella podemos encontrar los lugares cercanos en el mapa y directamente hacer check-in de recomendaciones.

Figura 4 – Imagen de la web Yelp

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- **Foursquare**

Se trata de un servicio basado en localización Web aplicada a las redes sociales. La geolocalización permite localizar un dispositivo fijo o móvil en una ubicación geográfica.

La idea principal de la red es marcar (*check-in*) lugares específicos donde uno se encuentra e ir ganando puntos. A partir de la información que los usuarios han ido introduciendo, el servicio ha ido evolucionando hacia un motor de recomendaciones que sugiere lugares interesantes de manera inteligente.

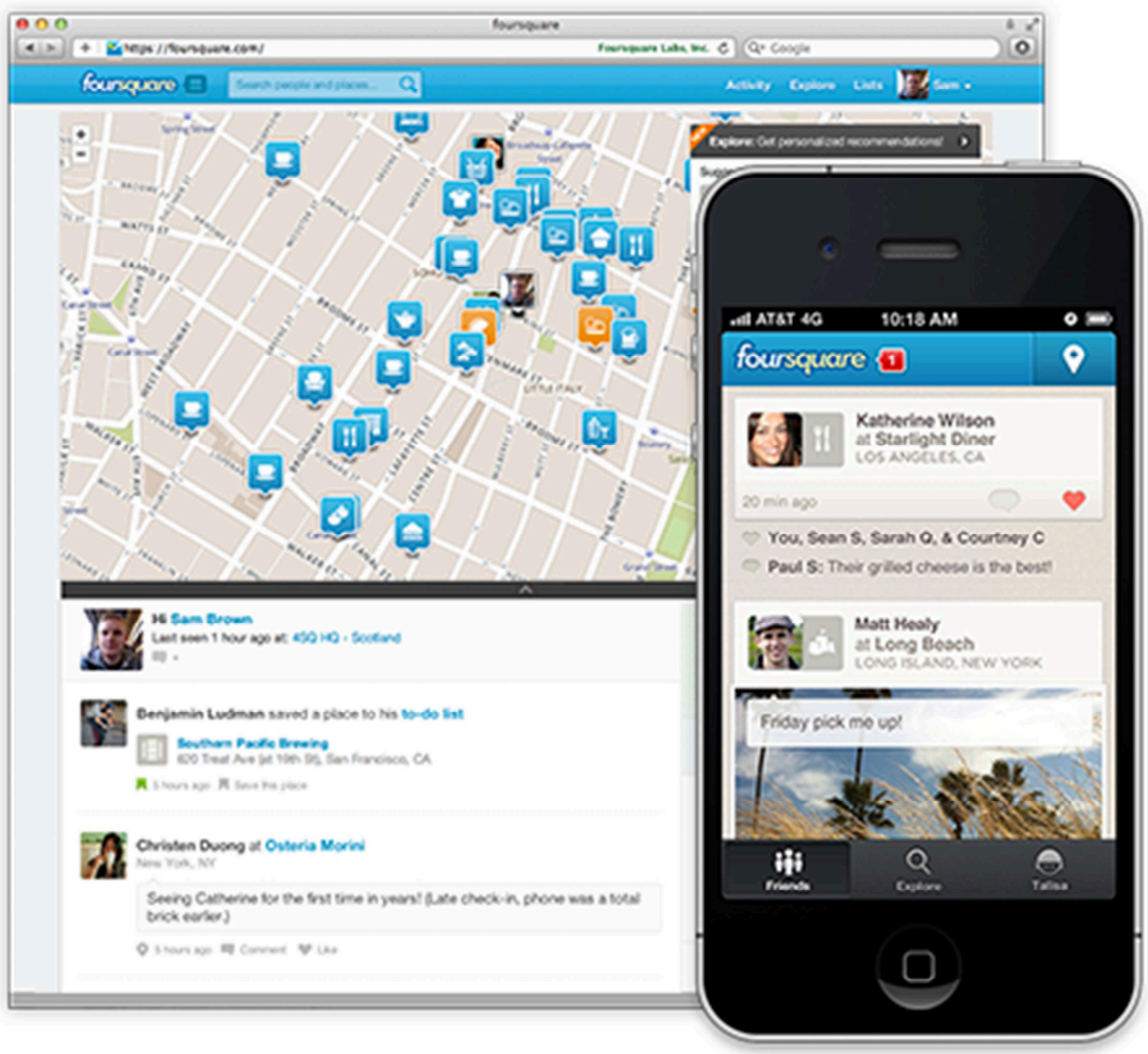


Figura 5 – Imagen de la web Foursquare

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Necesidades que se quieren cubrir

La necesidad fundamental que se quiere cubrir con este proyecto es recoger información de Facebook con geo-localización y colocarla en un mapa de nuestro sitio web.

La idea del proyecto es una versión inicial que podría ser implementada con otras redes sociales y reunir la información de varias redes sociales en una sola página web. Siempre aprovechando la información con localización.

Como se cubren desde otras aplicaciones

Facebook Nearby

Facebook Nearby es un servicio de recomendaciones personalizadas de acuerdo con nuestra ubicación pero se centra más en el marketing con el fin de descubrir negocios locales.

Tienen la ventaja que poseen la información más valiosa y que la gente pueda usar una herramienta que conoce (Facebook) a otra nueva.

Tagtagcity

TagTagCity proporciona a las empresas un sitio móvil basado en la localización. Con esto TagTagCity ofrece a tiendas una promoción de sí mismos.

En esta aplicación no trabajan con una red social. La información se extrae de una base de datos propia y esa información la convierten en marcadores.

Foursquare y Yelp

Funcionamiento similar que se basa en las opiniones/recomendaciones de la gente. Se centra en empresas y bares o lugares que se quieren dar a conocer de una forma gratuita.

Que necesidades quedan sin cubrir con otras herramientas

El tema de la localización es el punto que está sin desarrollar en otras herramientas o sitios Web. Esta necesidad es la que cubrimos en este trabajo ya que está sin explotar.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Objetivos a conseguir con el proyecto

El objetivo principal del proyecto es crear una nueva red social basada en la localización e implementar en un mapa de Google Maps las fotos que posean localización con ayuda del API de Facebook.

La idea es acceder a las fotos de Facebook que posean localización, obtener su latitud y longitud y después colocar un marcador en un mapa de Google Maps en el lugar donde se realizaron.

Mostrar toda la información de cada foto con localización en la posición adecuada con un marcador en el mapa.

Acceder a toda la información de Facebook tanto de nuestros amigos como nuestra propia .En el mapa se mostraran los objetos tanto propios como de nuestros amigos de la red social en la que nos autentifiquemos.

Otro objetivo es conseguir un dominio de las APIs que vamos a trabajar, y coger soltura con el manejo tanto del API de Google Maps como con el de Facebook

También pretendemos familiarizarnos con el desarrollo de mashups y coger destreza en la unión de información de diferentes APIs.

Se va a trabajar tanto con HTML como con Javascript durante todo el proyecto, lenguajes en los cuales se pretende obtener destreza.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.- ARQUITECTURA DEL SISTEMA

Este proyecto es un mashup debido a que usa y combina contenido de más de una fuente, para crear un nuevo servicio simple, visualizado en una única interfaz gráfica.

La arquitectura de los mashups está siempre compuesta de tres partes:

- El proveedor de contenidos: fuente de los datos. Los datos están disponibles vía una API y diferentes protocolos web como RSS, REST y Web Service.
- El sitio mashup: es la nueva aplicación web que provee un nuevo servicio utilizando diferente información y de la que no es dueña.
- El web browser cliente: es la interface de usuario del mashup. En una aplicación web, el contenido puede ser mezclado por los web browser clientes usando lenguaje web del lado del cliente. Por ejemplo Javascript.

En mi proyecto los componentes de la arquitectura del mashup son los siguientes:

- El proveedor de contenidos: Los datos los obtenemos de los APIs de Facebook y Google Maps
- El sitio mashup: es nuestra aplicación web que está depositada en nuestro servidor y en la cual se maneja la información obtenida de los APIs.
- El web browser cliente: Se trata de los navegadores de los clientes que acceden a nuestro sitio web.

Tipos de Mashups

- Mashup de presentación

Se trata de plataformas que muestran una apariencia similar al escritorio de un sistema operativo, en las que el usuario incluye la información que desea de forma visual, esta información suele proceder de diversas fuentes, y la distribuye libremente por el área de trabajo.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

De esta inclusión obtenemos una combinación de datos que aunque no llegan a mezclarse literalmente, sí se consigue un enriquecimiento de la experiencia de usuario, ya que puede reunir en una única vista la información que desee. Un ejemplo de esto puede ser iGoogle.

- Mashup de datos

Se utilizan, como su nombre indica, para combinar datos, Concretamente se mezclan datos procedentes de más de una fuente distinta y se obtienen como resultado datos combinados y con un valor añadido con el que no contarían por separado. En definitiva, estas plataformas se han categorizado de este modo, atendiendo únicamente a su forma de manejar los datos. Un ejemplo sencillo sería su utilización en sindicación de contenidos (RSS).

Otra forma de categorizarlos puede ser:

mashups de consumidores, mashups de datos y mashups empresariales.

El tipo más conocido es el de mashup de consumidores, que está muy bien ejemplificado por muchas aplicaciones que utilizan Google Maps. Los mashups de este tipo combinan datos de fuentes varias, escondiendo ello tras una interfaz gráfica simple.

Un mashup de datos mezcla datos de tipo similar proveniente de diferentes fuentes. Por ejemplo, combinando los datos de múltiples feeds RSS en un solo feed con nuevo un front-end gráfico.

Un mashup empresarial integra usualmente datos de fuentes externas e internas. Por ejemplo, podría crear un informe sobre la cuota de mercado de un negocio combinando la lista externa de todas las casas vendidas la semana anterior con datos internos de las casas vendidas por una sola agencia.

Mashup de consumidores: Los mashups de este tipo combinan datos de fuentes variadas, escondiendo ello tras una interface gráfica simple.

Mashup de datos: Mezcla datos de tipo similar proveniente de diferentes fuentes. Por ejemplo, combinando los datos de múltiples feeds RSS en un solo feed con nuevo un front-end gráfico.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Mashup empresarial: Integra usualmente datos de fuentes externas e internas. Por ejemplo, podría crear un reporte sobre la cuota de mercado de un negocio combinando la lista externa de todas las casas vendidas la semana anterior con datos internos de las casas vendidas por una sola agencia.

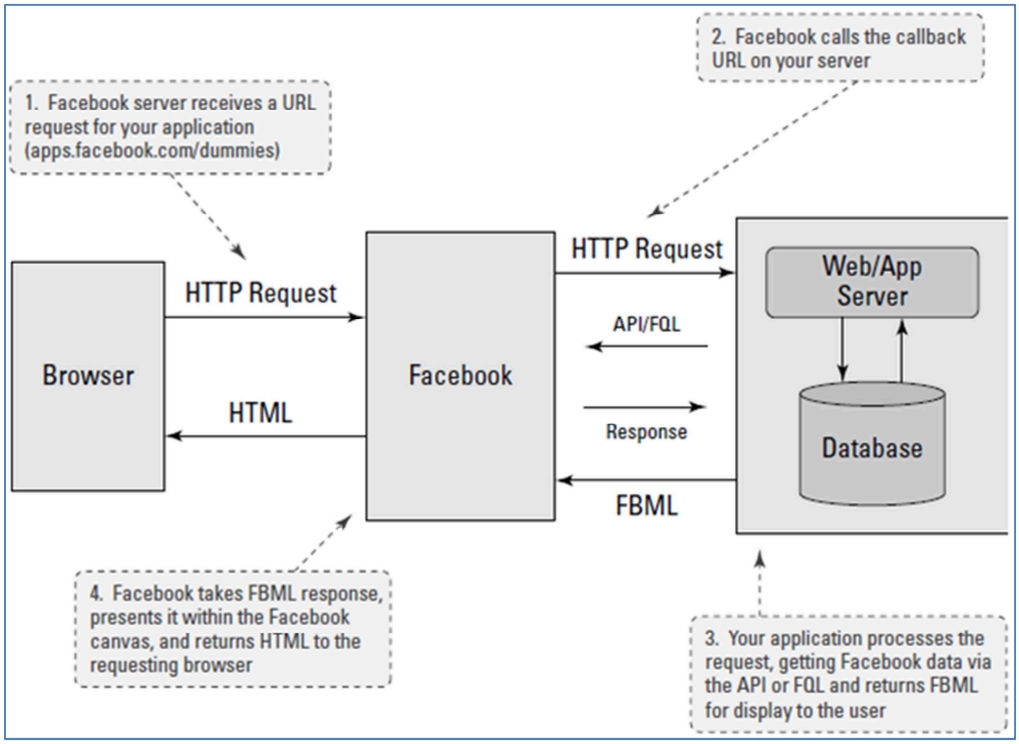


Figura 6 – Esquema de la arquitectura del sistema.

Mi tipo de Mashup

Según el objetivo del mashup, Mi proyecto es un mashup de consumidores puesto que esta enfocados al usuario final.

Otra razón por la que se concluye que es un mashup de consumidores es porque combina información de Facebook con un mapa de Google Maps, detrás de una única interfaz gráfica simple.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

3.- HERRAMIENTAS Y COMPONENTES EXTERNOS

Las APIs tienen un papel importante en el desarrollo de mashups.
El mashup accede a los APIs de Google Maps y Facebook, y para ello he seguido los siguientes pasos:

API de Facebook

Crear App Facebook

Las aplicaciones de Facebook tienen muchas funcionalidades, una de las utilidades que más nos pueden interesar es el poder utilizar las aplicaciones de Facebook para personalizar nuestra *landing tab*, o pestaña de aterrizaje en Facebook.

Para configurar una aplicación para Facebook, nos tenemos que dirigir a la dirección http://Facebook.com/developers que nos mostrará el listado de aplicaciones que hemos creado, junto con un botón de Create New App. Haciendo clic en éste comenzamos el proceso de creación de nuestra aplicación para Facebook con los dos primeros datos:

App Display Name: Es el nombre que queremos darle a la aplicación
App Namespace: Este es el identificador (único) de la aplicación que formará parte de la url de ésta: http://apps.Facebook.com/tuAppNamespace.

Y una vez introduzcamos estos datos, junto con el captcha que nos pedirá Facebook, pasaremos a configurar el resto de parámetros de la aplicación. Si se va a tratar de una aplicación para móviles nos pedirá unos datos, o si se trata de una aplicación web estándar nos pedirá otros, etc.

Lo interesante es que independientemente del tipo de aplicación que sea, Facebook nos otorgará un identificador de aplicación (App Id) y una clave secreta (App Secret) para poder utilizar esta aplicación desde el API de Facebook. Ambas claves son necesarias para los pasos necesarios de cara a configurar nuestra aplicación en Facebook.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

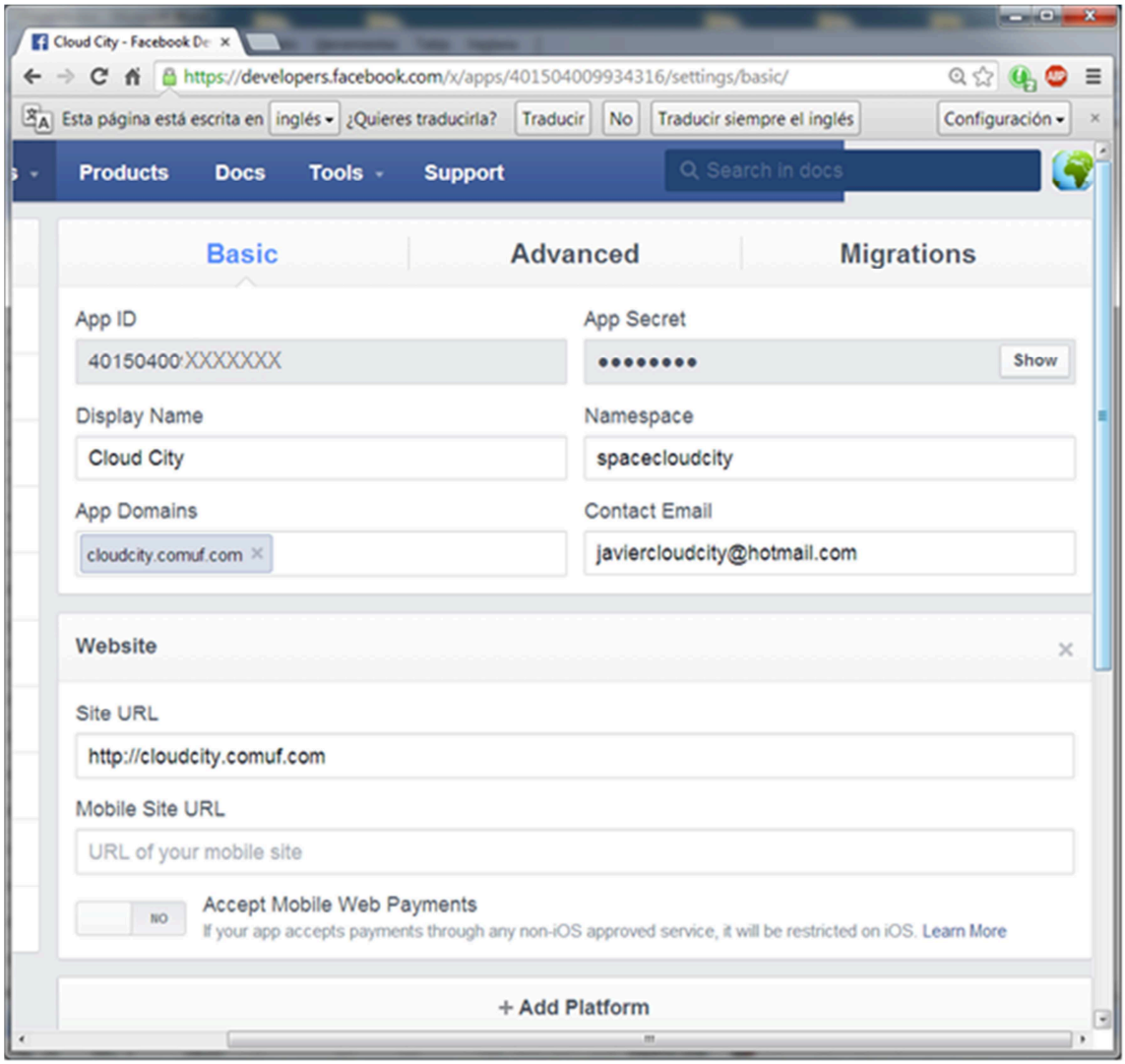


Figura 7 – Configuración de la App de Facebook

Información de la aplicación de Facebook:

El namespace es spacecloudcity
El id de la aplicación de Facebook es '40150400XXXXXXX'
El contacto de Email es javiercloudcity@hotmail.com
El dominio de la App es cloudcity.comuf.com

El id de la aplicación lo pondremos en nuestro código para poder autentificarnos y tener acceso a Facebook.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Autenticación con Facebook SDK para Javascript

El objetivo es dar la opción a los usuarios de iniciar sesión utilizando su cuenta de Facebook. Permitiendo esto, se consigue evitar al usuario tomarse tiempo para crear una nueva cuenta en el sitio y tener que recordar un usuario y contraseña más.

Para los desarrolladores, la tentación de utilizar estos métodos que yo llamo "Externos" es demasiada, ya que permite evitarte el módulo de autenticación que no es cualquier cosa, además ya no tendrás en tus bases de datos nombres de usuarios y contraseñas, información sumamente valiosa y que para algunos puede ser un alivio no tener que gestionar esta información.

Describiremos el proceso de autenticación mediante Facebook utilizando el SDK para Javascript, éste funciona mediante el estándar OAuth que también podría ser implementado desde algún lenguaje de servidor si así se necesitara.

Requisitos:

Un *App Id*, este identificador de aplicación se generará automáticamente al crear una aplicación en Facebook Developers , como he indicado en el paso anterior.

Un *lugar donde hospedar tus archivos HTML*, es necesario que tus archivos puedan ser accesibles desde una url, por lo que tendrás que utilizar un servidor web. Comencemos a programar:

Paso 1: Agregar/Cargar el SDK de Facebook

```
<head>
<script src="//connect.Facebook.net/en_US/all.js"></script>
<head>
```

Paso 2: Agregar referencia a JQuery API

```
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script >
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Paso 3: Inicializar SDK

Utilizamos la función ready de JQuery para inicializar el SDK de Facebook, utilizando la función FB.init con los parámetros appId, status y xfbml.

```
<script>
$("document").ready(function () {
  FB.init({
    appId: '111111111111111', // App ID
    status: true,// revisa el estatus del  login
    xfbml: true //  muestra el botón login
  });
});
<script/ >
```

Paso 4: Agregar botón de login

Dentro del <body>, agregar los siguientes divs.

```
<div id="fb-root"></div>
<fb:login-button show-faces="false"></fb:login-button>
```

Estos div son tomados por el SDK para construir el botón de Login. Quedando algo así:

Paso 5: Adjuntar un manejador para el evento auth.statusChange

Este manejador estará escuchando el evento statusChange, el cual se dispara cuando exista un cambio en el status del usuario, las tres condiciones que hacen que este evento se disparare son:

- En el inicio de Sesión (Login)
- En el cierre de Sesión (Logout)
- En el refresco de la Sesión

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Para adjuntar este evento tenemos que utilizar el método FB.Event.subscribe, el cual implementa un Callback donde me devuelve la información del estatus actual del usuario. Esta acción tendrá que estar dentro del método ready de JQuery.

```
FB.Event.subscribe('auth.statusChange', OnLogin);
```

Aquí podemos tener tres posibles valores de estatus:
Conectado (connected), si el usuario esta logueado a Facebook y autorizó tu aplicación
No Autorizado (not_authorized), si el usuario esta logueado a Facebook pero no autorizó tu aplicación.
Desconocido (Unknow), si no se sabe si el usuario está logueado a Facebook.

```
function OnLogin(response) {
//Si el estatus es conectado.
     if (response.status === 'connected') {
//Si el estatus es no autorizado
     } else if (response.status === 'not_authorized') {
//Si es desconocido
     } else {
}
```

Hasta aquí, Facebook solamente nos da la información del estatus, es decir, sólo podemos saber si el usuario se auténtico, autorizó o no está conectado, en este momento no tenemos acceso a ninguna información del perfil del usuario.

Para lograr acceder a la información del perfil, el usuario debe tener un estatus de *connected*, sólo entonces podemos acceder a su información usando el método FB.API. Este método implementa un callback en el cual Facebook devuelve la información del perfil dentro de un objeto de tipo Usuario.

```
if (response.status === 'connected') {
  FB.API('/me', function(response){
    $("#nombre").html(response.first_name + " "+response.last_name);
    $("#correo").html(response.email);
    $("#miPagina").html(response.link);
    var perfil="https://graph.Facebook.com/"+response.username+"/picture?type=large&#821;
    $("#imagen").attr('src', perfil);
  });
}
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Sacar lista de Amigos de Facebook

Una FriendList es un objeto que hace referencia a un grupo de amigos creado por alguien o generados automáticamente para alguien (como los "amigos cercanos" o listas de "conocidos"). Puedes ver todas las listas de amigos de una persona utilizando el user-id de Facebook.

Para poder obtener la lista de amigos de alguien a través del API debes pedirle permiso una vez se autentifica.

En este caso ,se requiere un *token* de acceso de usuario con permiso read_friendlists para ver cualquiera de las listas de amigos de esa persona.

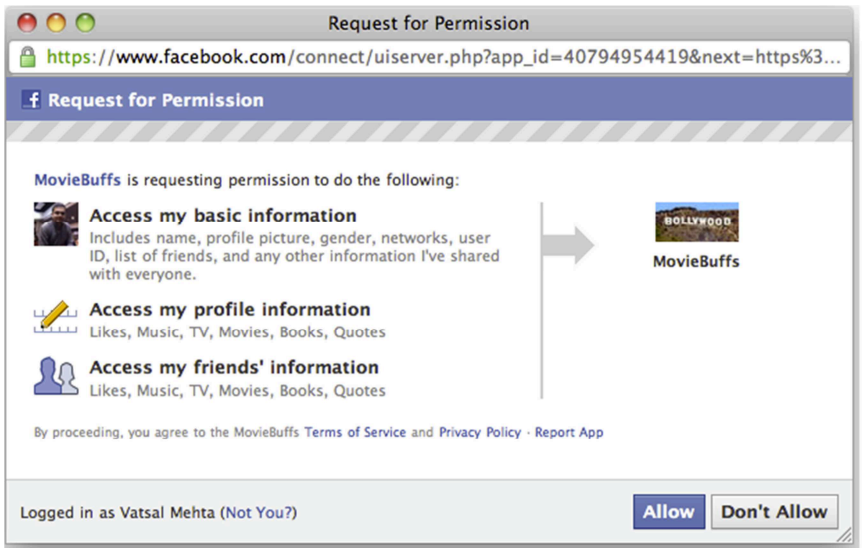


Figura 8 – Imagen de los permisos de Facebook

La forma de acceder a la lista de amigos con el SDK de Javascript es la siguiente:

```
FB.API('me/friends?fields=id,name,picture,birthday,gender',
function(friendsQuery) {
        var friends = friendsQuery["data"];
        for(var j=0; j < friends.length ; j++) {
                window.CloudCity.db.friends.insert({
                        id: friends[j]["id"],
                        network: 'Facebook',
                        name: friends[j]["name"],
                        avatar: friends[j]["picture"]["data"]["url"],
                        birthday: friends[j]["birthday"],
                        gender: friends[j]["gender"],
                        extra: friends[j]
                });
        }
});
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Acceder a las fotos de los amigos

Las fotos de los usuarios de Facebook se almacenan dentro de álbumes.
Un álbum de fotos como se representa en la Graph API.

Para leer un álbum se requiere de varios permisos

- El permiso user_photos si pertenece a un usuario.
- El permiso friends_photos si pertenece a un amigo de un usuario.

Si una aplicación permite al usuario elegir un álbum al cargar las fotos, la aplicación debe comprobar la opción can_upload para asegurarse de que no se permite a la aplicacións añadir nuevas fotos al álbum.

Para acceder a las fotos seguimos los siguientes pasos:

Primero accedemos al API y consultamos los amigos.
Después recorremos los amigos y accedemos a todas las fotos de cada amigo.
Por último recorremos las fotos de cada amigo y creamos un marcador por foto y le asignamos la localización y la información de la foto.

```
FB.API('me/friends?fields=id,name,picture,birthday,gender',
function(friendsQuery) {

   var friends = friendsQuery["data"];

   for(var j=0; j < friends.length ; j++){

      FB.API('/' + friends[j]["id"]+'/? fields=id,picture,name,locations.fields
( id,type,place,created_time,from)', function(locationsQuery){

         for(var i=0; i < locationInfo.length ; i++) {

            var marker = {
               id: locationInfo[i]["id"],
               friend_id: id,
               friend_network: 'Facebook',
               lat: locationInfo[i]["place"]["location"]["latitude"],
               lng: locationInfo[i]["place"]["location"]["longitude"],
               time: new Date(locationInfo[i]["created_time"]),
               link: "www.Facebook.com/" + locationInfo[i]["id"],
               extra: locationInfo[i],
               icon: Logo
            }
         }
      });
   }
});
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

API de Google Maps

Obtener una clave de API

Todas las aplicaciones del API de Google Maps deben cargar el API de Google Maps mediante una clave de API.

Utilizar una clave de API te permite controlar cómo utiliza tu aplicación el API de Google Maps y asegurarte de que Google puede ponerse en contacto contigo con respecto a tu aplicación si fuese necesario. Si el uso que hace tu aplicación del API Google Maps supera los límites de uso, debes cargar el API de Google Maps mediante una clave de API para adquirir un límite adicional.

Los desarrolladores del API de Google Maps for Business no deben incluir una clave en sus solicitudes. Consulta la sección Cómo cargar el API de Javascript de Google Maps para obtener instrucciones específicas para las empresas.

Para crear tu clave de API:

- Accede a la página de la consola de las API
- Inicia sesión con tu cuenta de Google.
- Haz clic en el enlace de servicios.
- Activa el servicio de la versión 3 del API de Google Maps.
- Haz clic en el enlace de acceso al API . Tu clave de API está disponible desde la página de acceso al API, en la sección de acceso al API sencilla. Las aplicaciones del API de Google Maps utilizan la clave para aplicaciones del navegador.

De forma predeterminada, una clave se puede utilizar en cualquier sitio. Google recomienda que se restrinja el uso de la clave a los dominios que administres para evitar que se utilice en sitios no autorizados. Para especificar los dominios que pueden utilizar la clave de API, haz clic en el enlace para editar referencias permitidas.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

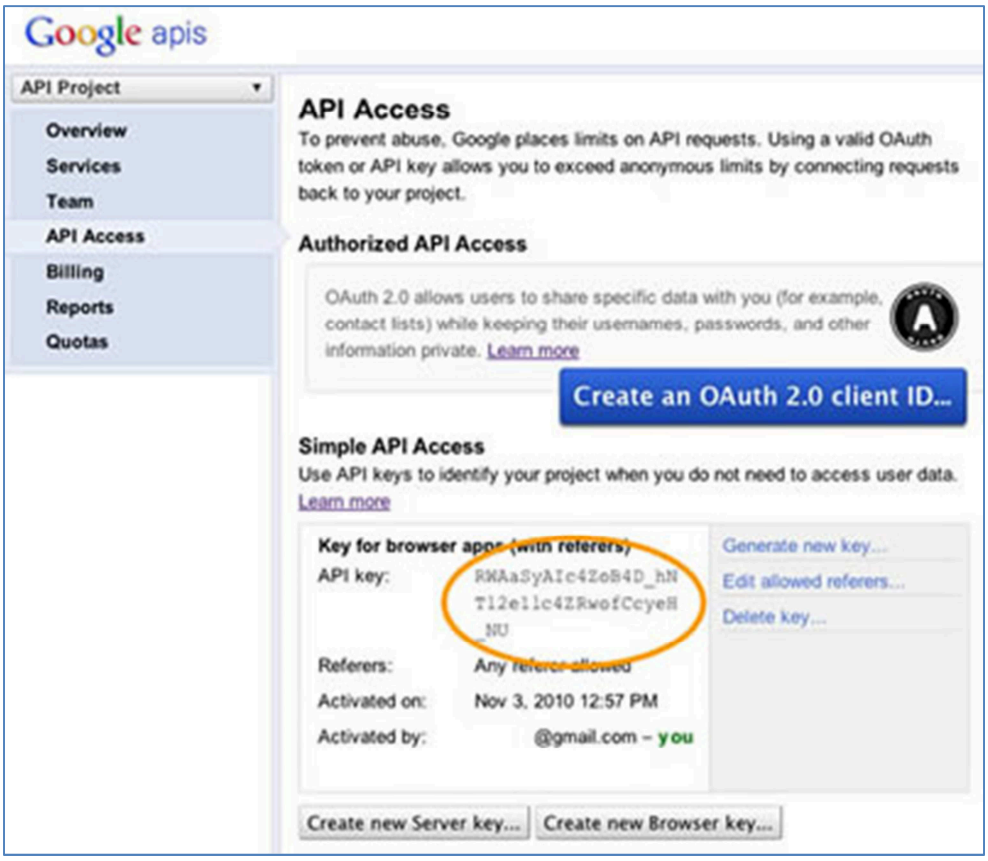


Figura 9 – Imagen de la clave de acceso al API de Google Maps

Cargar el API de Google Maps

```
<html>
    <head>
        <script type="text/javascript"
src="http://Maps.GoogleAPIs.com/Maps/API/js?key=YOUR_API_KEY&sensor=SET_
TO_TRUE_OR_FALSE">
      </script>
```

La URL incluida en la etiqueta script indica la ubicación de un archivo Javascript que carga todos los símbolos y las definiciones que necesitas para utilizar el API de Google Maps. La etiqueta script es obligatoria.

El parámetro *key* incluye la clave de API de tu aplicación. Para obtener más información, consulta la sección Cómo obtener una clave de API. Ten en cuenta que esta clave no es la misma clave que se utiliza en la versión 2 del API y que se debe generar a partir de la consola de las API.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

El parámetro sensor de la URL es obligatorio e indica si esta aplicación utiliza un sensor (por ejemplo, un localizador GPS) para determinar la ubicación del usuario. En este ejemplo hemos dejado el parámetro como la variable *set_to_true_or_false* para hacer hincapié en que se debe definir este valor en true o false de forma explícita.

Añadir mapa en nuestra página web

La manera más fácil de implementar un mapa del API de Google Maps es observar un sencillo ejemplo. La siguiente página web muestra un mapa centrado en Sídney, Nueva Gales del Sur, Australia:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
    <style type="text/css">
      html { height: 100% }
      body { height: 100%; margin: 0; padding: 0 }
      #map_canvas { height: 100% }
    </style>
    <script type="text/javascript"

src="http://Maps.GoogleAPIs.com/Maps/API/js?key=YOUR_API_KEY&sensor=SET
_TO_TRUE_OR_FALSE">
    </script>
    <script type="text/javascript">
      function initialize() {
        var mapOptions = {
          center: new Google.Maps.LatLng(-34.397, 150.644),
          zoom: 8,
          mapTypeId: Google.Maps.MapTypeId.ROADMAP
        };
        var map = new Google.Maps.Map(document.getElementById("map_canvas"),
            mapOptions);
      }
    </script>
  </head>
  <body onload="initialize()">
    <div id="map_canvas" style="width:100%; height:100%"></div>
  </body>
</html>
```

Incluso en este sencillo ejemplo, hay algunas cosas que se deben tener en cuenta:

Declaramos la aplicación como HTML5 mediante la declaración <!DOCTYPE HTML>.

Incluimos el código Javascript del API de Google Maps mediante la etiqueta script.Creamos un elemento div denominado "map_canvas" que aloja el mapa.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Creamos un objeto Javascript literal para alojar una serie de propiedades de mapa.

Escribimos una función Javascript para crear un objeto de mapa.
Inicializamos el objeto de mapa desde el evento *onload* de la etiqueta body.
A continuación, se explican estos pasos.

Esta declaración de CSS indica que el contenedor del mapa (denominado map_canvas) debe ocupar el 100% de la altura del cuerpo de HTML. Ten en cuenta que debemos declarar de forma específica estos porcentajes tanto para <body> como para <HTML>.

```
<style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0; padding: 0 }
  #map_canvas { height: 100% }
</style>
```

Cargar el API de forma asíncrona

A continuación se muestra un fragmento de código que permite que la aplicación cargue el API de Google Maps una vez que se haya cargado por completo la página (mediante window.onload) y que se incluya el API de Javascript de Google Maps en una etiqueta <script> dentro de la página. También se puede hacer que el API no ejecute la función *initialize*() hasta que se haya cargado por completo.

```
function initialize() {
  var mapOptions = {
    zoom: 8,
    center: new Google.Maps.LatLng(-34.397, 150.644),
    mapTypeId: Google.Maps.MapTypeId.ROADMAP
  }
  var map = new Google.Maps.Map(document.getElementById("map_canvas"),
mapOptions);
}

function loadScript() {
  var script = document.createElement("script");
  script.type = "text/javascript";
  script.src =
"http://Maps.GoogleAPIs.com/Maps/API/js?key=YOUR_API_KEY&sensor=TRUE_O
R_FALSE&callback=initialize";
  document.body.appendChild(script);
}

window.onload = loadScript;
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Elementos DOM de mapas

```
<div id="map_canvas" style="width: 100%; height: 100%"></div>
```

Para que el mapa aparezca en una página web, se debe reservar un lugar para él.

Normalmente, lo hacemos mediante la creación de un elemento div con nombre y la obtención de una referencia a este elemento en el modelo de objetos de documento (DOM) del navegador.

En el ejemplo anterior, definimos un objeto denominado "map_canvas" y establecemos su tamaño mediante atributos de estilo. Ten en cuenta que este tamaño se establece en "100%", lo que amplía el mapa hasta que ocupa toda la pantalla del dispositivo móvil. Es posible que sea necesario ajustar estos valores según el desplazamiento y el tamaño de la pantalla del navegador. Ten en cuenta que el mapa siempre adoptará el tamaño del elemento en el que esté contenido, por lo que siempre debes establecer un tamaño para el elemento de forma explícita.

Opciones de mapas

```
var mapOptions = {
  center: new Google.Maps.LatLng(-34.397, 150.644),
  zoom: 8,
  mapTypeId: Google.Maps.MapTypeId.ROADMAP
};
```

Antes de inicializar un mapa, debemos crear un objeto Map options que contenga las variables de inicialización correspondientes. Este objeto no se construye, sino que se crea como un objeto literal.

```
var mapOptions = {};
```

Latitudes y longitudes

Como queremos center el mapa en un punto específico, creamos un objeto LatLng para mantener esta ubicación especificando las coordenadas de ubicación en el orden {latitud, longitud}:

```
center = new Google.Maps.LatLng(-34.397, 150.644)
```

El proceso de convertir una dirección en un punto geográfico se conoce como codificación geográfica. Esta versión del API de Google Maps incluye funciones de codificación geográfica.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Niveles de zoom

La propiedad zoom especifica la resolución inicial con la que se mostrará un mapa, donde un zoom de 0 se corresponde con un mapa de la Tierra totalmente alejado y los niveles de zoom acercan el mapa con una resolución más elevada.

zoom: 8

Para ofrecer un mapa de todo el planeta como una única imagen, es necesario un mapa muy grande o un mapa pequeño con una resolución muy baja. Por consiguiente, las imágenes de mapa en Google Maps y el API de Google Maps se dividen en "mosaicos" de mapas y "niveles de zoom". A niveles bajos de zoom, un conjunto pequeño de mosaicos de mapas cubre una superficie amplia; a niveles de zoom más elevados, los mosaicos tienen una resolución mayor y cubren una superficie más pequeña.

Tipos de mapas

También debes establecer expresamente un tipo de mapa inicial en este momento.

mapTypeId: Google.Maps.MapTypeId.ROADMAP

Se admiten los siguientes tipos de mapas:

- ROADMAP, que muestra los mosaicos normales en 2D predeterminados de Google Maps.
- SATELLITE muestra imágenes de satélite.
- HYBRID muestra una mezcla de mosaicos fotográficos y una capa de mosaicos para los elementos del mapa más destacados (carreteras, nombres de ciudades, etc.).
- TERRAIN muestra mosaicos de relieve físico para indicar las elevaciones del terreno y las fuentes de agua (montañas, ríos, etc.).

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

El objeto "Map"

```
var map = new Google.Maps.Map(document.getElementById("map_canvas"),
    mapOptions);
```

La clase de Javascript que representa a los mapas es Map. Cada objeto de esta clase define un único mapa en una página. (Puedes crear más de una instancia de esta clase; cada objeto definirá un mapa independiente en la página). Creamos una nueva instancia de esta clase mediante el operador new de Javascript.

Al crear una nueva instancia de mapa, se especifica un elemento HTML en la página como contenedor para el mapa. Los nodos HTML son elementos secundarios del objeto *document* de Javascript. Se obtiene una referencia a este elemento mediante el método document.getElementById().

Este código permite definir una variable (denominada *map*) y asignar dicha variable a un nuevo objeto *Map*, además de transmitir opciones definidas en el objeto mapOptions literal. Estas opciones se utilizarán para inicializar las propiedades del mapa.

Cargar el mapa

```
<body onload="initialize()">
```

Mientras se procesa una página HTML, se crea el modelo de objetos de documentos (DOM) y las imágenes y secuencias de comandos externas se reciben y se incorporan al objeto *document*. Para garantizar que nuestro mapa se añada a la página cuando se cargue por completo, solo ejecutamos la función que crea el objeto *Map* cuando el elemento <body> de la página HTML ha recibido un evento *onload*.

De este modo, evitamos un comportamiento impredecible y obtenemos más control acerca del modo y del momento en que se dibuja el mapa.

El atributo *onload* de la etiqueta *body* es un ejemplo de un controlador de eventos. El API de Javascript de Google Maps también proporciona varios eventos que se pueden controlar para determinar cambios de estado. Para obtener más información, consulta la sección Eventos de mapa.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Insertar marcadores con localización de las fotos

Marcadores

Los marcadores identifican ubicaciones en el mapa. De manera predeterminada, utilizan un icono estándar, aunque puedes establecer un icono personalizado dentro del constructor del marcador o mediante la ejecución de *setIcon()* en el marcador.

El constructor Google Maps Marker toma un único objeto literal Marker options que especifica las propiedades iniciales del marcador. A continuación se indican algunos campos especialmente importantes que se suelen definir al crear un marcador.

- Position (obligatorio) especifica un valor de *LatLng* que identifica la ubicación inicial del marcador.

- Map (opcional) especifica el objeto *Map* en el que se sitúa el marcador.

Los marcadores están diseñados para ser interactivos. De forma predeterminada, reciben eventos y se suelen utilizar dentro de detectores de eventos para abrir ventanas de información.

En el siguiente ejemplo, se añade un marcador simple a un mapa :

```
var myLatlng = new Google.Maps.LatLng(-25.363882,131.044922);
  var mapOptions = {
    zoom: 4,
    center: myLatlng,
    mapTypeId: Google.Maps.MapTypeId.ROADMAP
  }
  var map = new Google.Maps.Map(document.getElementById("map_canvas"),
mapOptions);

  var marker = new Google.Maps.Marker({
      position: myLatlng,
      map: map,
      title:"Hello World!"
  });
```

Este título de Marker se mostrará como información sobre la herramienta.

Si no quieres incluir ninguna opción del marcador (*Marker options*) en el constructor del marcador, incluye un objeto vacío en el último argumento del constructor.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Añadir información de las fotos a los marcadores

Ventanas de información

Las ventanas de información (InfoWindow) muestran contenido en una ventana flotante situada encima del mapa. La ventana de información tiene un aspecto ligeramente parecido al de los bocadillos de los cómics.

Tiene un área de contenido y un pico afilado, cuyo extremo se encuentra en una ubicación especificada en el mapa. Puedes ver cómo funcionan las ventanas de información si haces clic en los marcadores de negocios de Google Maps.

El constructor *InfoWindow* utiliza un objeto *InfoWindow* options , que especifica un conjunto de parámetros iniciales para la visualización de la ventana de información.

La ventana de información no se añade al mapa cuando se crea. Para mostrar la ventana de información, necesitas ejecutar el método *open()* en el objeto *InfoWindow*, transmitir el mapa (*Map*) en el que se va a abrir y, opcionalmente, el marcador (*Marker*) en el que se va a anclar

El objeto *InfoWindow options* es un objeto literal que contiene los siguientes campos:

- Content contiene una cadena de texto o un nodo DOM que se mostrará cuando se abra la ventana de información.

- Position contiene el objeto *LatLng* en el que se ancla esta ventana de información. Ten en cuenta que este valor se actualiza automáticamente con una nueva posición cuando se abre la ventana de información en un marcador.

- MaxWidth especifica la anchura máxima en píxeles de la ventana de información. De forma predeterminada, las ventanas de información se amplían para ajustarse a su contenido y ajustan el texto automáticamente si la ventana de información se amplía hasta cubrir el mapa.

El contenido de *InfoWindow* puede incluir una cadena de texto, un fragmento HTML o un elemento DOM. Para especificar este contenido, puedes trasmitirlo al constructor *InfoWindow options* o ejecutar *setContent()* explícitamente en el objeto *InfoWindow*. Si quieres especificar explícitamente el tamaño del contenido, puedes utilizar elementos y, si quieres, habilitar el desplazamiento.

Para cambiar la ubicación de la ventana de información, puedes ejecutar *setPosition()* explícitamente en la ventana de información o bien adjuntarla a un nuevo marcador mediante el método *InfoWindow.open()*. Ten en cuenta que si ejecutas open ()

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

sin transmitir un marcador, la ventana de información (*InfoWindow*) utilizará la posición especificada en la construcción mediante el objeto *InfoWindow options*.

El fragmento de código que aparece a continuación permite mostrar un marcador en el centro de Australia. Al hacer clic en el marcador, aparece la ventana de información.

```
var myLatlng = new Google.Maps.LatLng(-25.363882,131.044922);
var mapOptions = {
  zoom: 4,
  center: myLatlng,
  mapTypeId: Google.Maps.MapTypeId.ROADMAP
}

var map = new Google.Maps.Map(document.getElementById("map_canvas"),
mapOptions);

var contentString = '<div id="content">'+
    '<div id="siteNotice">'+
    '</div>'+
    '<h2 id="firstHeading" class="firstHeading">Uluru</h2>'+
    '<div id="bodyContent">'+
    '<p><b>Uluru</b>, also referred to as <b>Ayers Rock</b>, is a large ' +
    'sandstone rock formation in the southern part of the '+
    'Northern Territory, central Australia. It lies 335 km (208 mi) '+
    'south west of the nearest large town, Alice Springs; 450 km '+
    '(280 mi) by road. Kata Tjuta and Uluru are the two major '+
    'Heritage Site.</p>'+
    '<p>Attribution: Uluru, <a
href="http://en.wikipedia.org/w/index.php?title=Uluru&oldid=297882194">'+
    'http://en.wikipedia.org/w/index.php?title=Uluru</a> (last visited June 22,
2009).</p>'+
    '</div>'+
    '</div>';

var infowindow = new Google.Maps.InfoWindow({
    content: contentString
});

var marker = new Google.Maps.Marker({
    position: myLatlng,
    map: map,
    title:"Uluru (Ayers Rock)"
});

Google.Maps.event.addListener(marker, 'click', function() {
  infowindow.open(map,marker);
});
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

4.- REQUISITOS DEL SISTEMA A DESARROLLAR

Hosting

El sitio web necesita estar alojado en un servidor para tener acceso desde cualquier ordenador. Para ello debemos contratar un dominio en un servidor.
Debido a que se trata de un proyecto de fin de curso y a que no era importante el nombre del dominio decidimos coger un subdominio gratuito en 000webhost.com.

El subdominio que nos proporcionaron es Cloudcity.comuf.com y este es el formulario de solicitud.

- **Nombre del subdominio**: Será el nombre que se desea utilizar para que los visitantes puedan acceder al sitio web que se está creando. La extensión del dominio que se ofrece va variando.
- **Nombre de usuario**: Para identificarse al acceder en otra ocasión al servidor y realizar modificaciones.
- **Dirección de correo electrónico**: Donde se recibirá información detallada sobre el servicio.
- **Contraseña**: Para asegurar que ninguna otra persona puede acceder a la configuración del sitio web que se va a crear.
  Se debe volver a escribir la misma contraseña por seguridad.
- **Número de seguridad:** Este número lo utiliza el servidor para evitar el acceso de forma automática a este servicio.

Figura 10 – Formulario de solicitud del hosting

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Una vez que ya éramos los poseedores del subdominio accedimos a nuestra cuenta para poder transferir archivos al servidor. Para ello introducimos email y contraseña que anteriormente hemos proporcionado.

Figura 11 – Sistema de acceso al servicio de hosting

Una vez que ya hemos accedido al panel de control del dominio, nos dirigimos a 'File Manager' para gestionar los archivos.

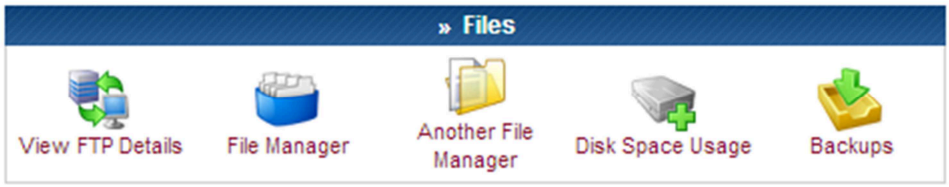


Figura 12 – Menú del panel de control

Para acceder a 'File Manager' nos vuelven a pedir introducir la contraseña de la cuenta.

Una vez dentro ya, tenemos acceso al sistema gestor de archivos del dominio en el cual vamos a poder subir archivos desde nuestro ordenador, descargar archivos, editarlos o eliminarlos. También aparece información de los ficheros alojados como el tipo de archivo, el tamaño, el propietario, el grupo, los permisos y la fecha de modificación.

Existe la opción de poder editar los ficheros sin la necesidad de descargarlo accediendo directamente al fichero desde el enlace Edit que se encuentra a la derecha de cada fichero.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

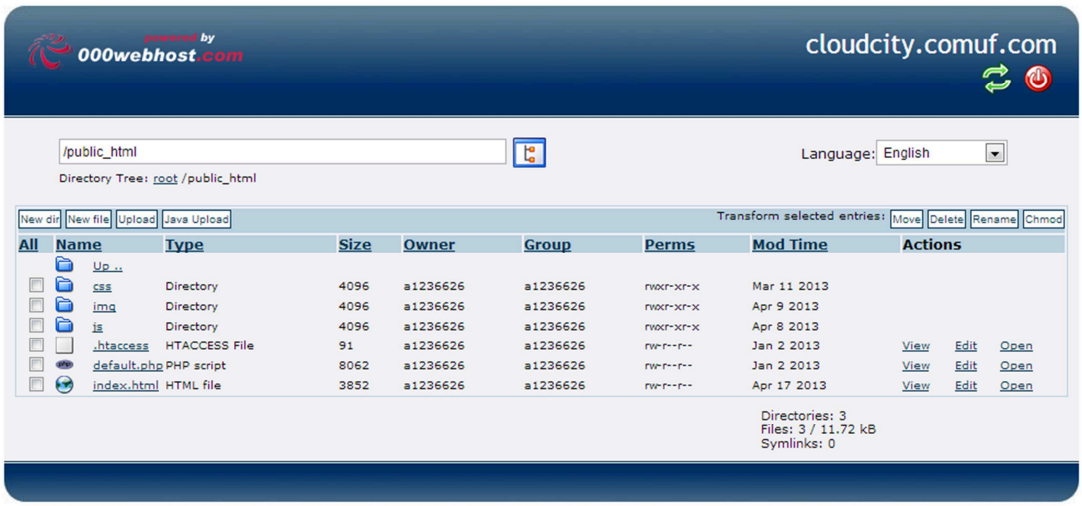


Figura 13 – Sistema de archivos

Facebook

El proyecto se centra en la red social Facebook y en su información, para poder utilizar esos datos tenemos que realizar diversas funciones.

Los requisitos que tenemos en el proyecto para Facebook son los siguientes:

CONEXIÓN CON FACEBOOK

Lo primero que debemos llevar a cabo es identificarnos con Facebook con el *id* de la aplicación que hemos creado en la red social.

AUTENTIFICACION

Comprueba si el usuario se ha logueado y si es así accede al API.

ACCESO AL API

El primer paso es consultar el API en busca de información de amigos.

ACCESO A LAS FOTOS

Una vez hemos obtenido del API a la información de los amigos accedemos a sus fotos con otra consulta al API en la cual ponemos como requisito que la foto tenga localización.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Base de Datos

En nuestro proyecto vamos a necesitar crear dos colecciones para almacenar la información de Amigos y Marcadores.

También vamos a necesitar gestionar la base de datos insertando o filtrando registros según nos interese. También vamos a utilizar un patrón (*Template*) para añadir valores a la base de datos.

Para ello vamos a utilizar *Taffy DB* que no es otra cosa que una biblioteca *open source* que reúne las características de base de datos en las aplicaciones de Javascript.Para poder utilizarla nos tenemos que descargar la librería y alojarla en el Hosting.

Taffy DB

Una forma de introducir en los navegadores bases de datos locales es *TaffyDB*.

TaffyDB es una librería Javascript *open source* que actúa como una delgada capa para aplicaciones web 2.0 y AJAX.

Vamos a mostrar una pequeña demostración de lo que se puede hacer con *TaffyDB*:

CREAR COLECCIONES

Para crear una colección *Taffy* hay que pasar un *array* de objetos similares como el que devuelve un servicio web JSON.

También acepta una cadena de texto JSON por evaluar. Siempre devolverá una colección de métodos para trabajar sobre el conjunto recibido.

BUSCAR

Para realizar una consulta se debe llamar al método *find* con un objeto como parámetro para indicar las condiciones de filtrado a aplicar. Es similar a la cláusula *where* de SQL. Devuelve un *array* de índices a los objetos que cumplan las condiciones.

ACTUALIZAR

El método *update* recibe un objeto con los datos que hay que cambiar y otro objeto (cláusula WHERE) con las condiciones de filtrado para encontrar los objetos

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

sobre los que hay que aplicar los cambios. También es posible indicar el índice del objeto a modificar o directamente el resultado de una búsqueda.

INSERTAR

El método *insert* acepta un objeto a insertar o un *array* de objetos.

ELIMINAR

El método para eliminar registros se llama *remove* dado que *delete* es una palabra reservada. Acepta como parámetro un objeto con las condiciones de filtrado para seleccionar los objetos a borrar.

ORDENAR

Para ordenar hay que indicar un *array* con los campos de ordenación, indicando el nombre del campo si es ascendente o con la expresión {"campo":"desc"} si es descendente. Por ejemplo, para ordenar por edad ascendente y por nombre alfabético pero descendente.

TEMPLATES

Se pueden usar patrones para añadir valores por defecto a una colección y minimizar el código escrito.

Un patrón es un objeto que será usado como base para nuevas inserciones. A menos que el nuevo registro sobrescriba los valores indicados en el patrón, *TaffyDB* usara los valores del patrón para rellenar el registro.

Interfaz Grafico

Para mostrar el interfaz gráfico del sitio web vamos a utilizar diferentes métodos para poder cargar todos los objetos que utilizamos a lo largo del proyecto.

METODO INIT

El método *init* se encarga de cargar las librerías utilizadas y también añade al menú de amigos de Facebook todos los amigos que están en la base de datos.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

MENU AMIGOS

Cargar el bloque en el cual aparece el menú de amigos de la red social.

MENU FILTROS

Cargar el bloque del menú de filtros para gestionar los marcadores.

VENTANA MODAL

Función que recibe como parámetros título y cuerpo del mensaje y los incrusta en la estructura HTML del bloque de la ventana modal y lo muestra en pantalla.

Mapa

CARGAR MAPA

Este método se encarga de cargar el mapa y poner sus opciones por defecto latitud, longitud, zoom, botones, tipo de mapa.

FILTRO DE MARCADORES

Recibe como parámetro el tipo de marcador a mostrar y los que no cumplen la condición, los oculta.

MOSTRAR MARCADORES

Este método se encarga de mostrar todos los marcadores se utiliza al autentificarse con Facebook.

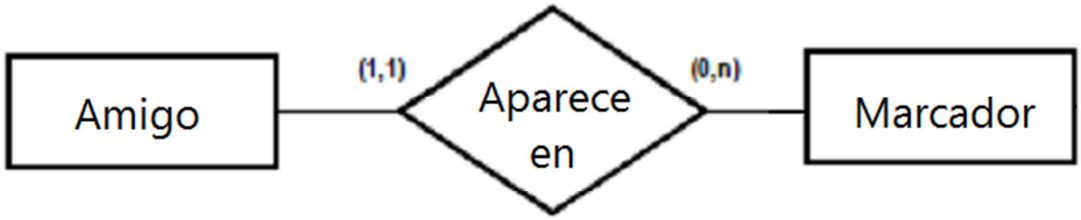
OCULTAR MARCADORES

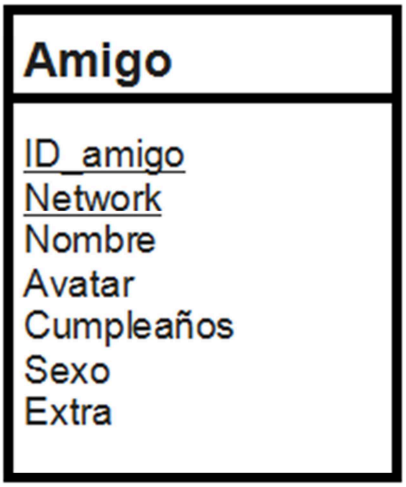
Este método oculta los marcadores del tipo que recibe como parámetro.

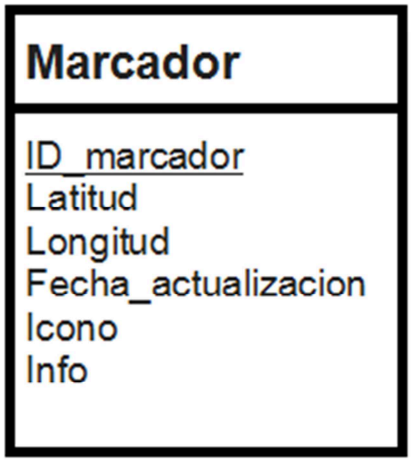
upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

5.- ANALISIS Y DISEÑO DEL SISTEMA

Diagrama de entidad-relación de bbdd.







1. Escribe los nombres de las entidades (rectángulo).En nuestro caso Amigo y Marcador.

2. Dibuja una línea entre las entidades que tienen una relación entre ellas. Y escribe la naturaleza de la relación dentro de él (rombo).

3. Añade los atributos a las entidades. En nuestro caso las he puesto en tablas.

4. Añade la cardinalidad o el número de elementos de cada entidad. En este caso la cardinalidad es 1:N

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

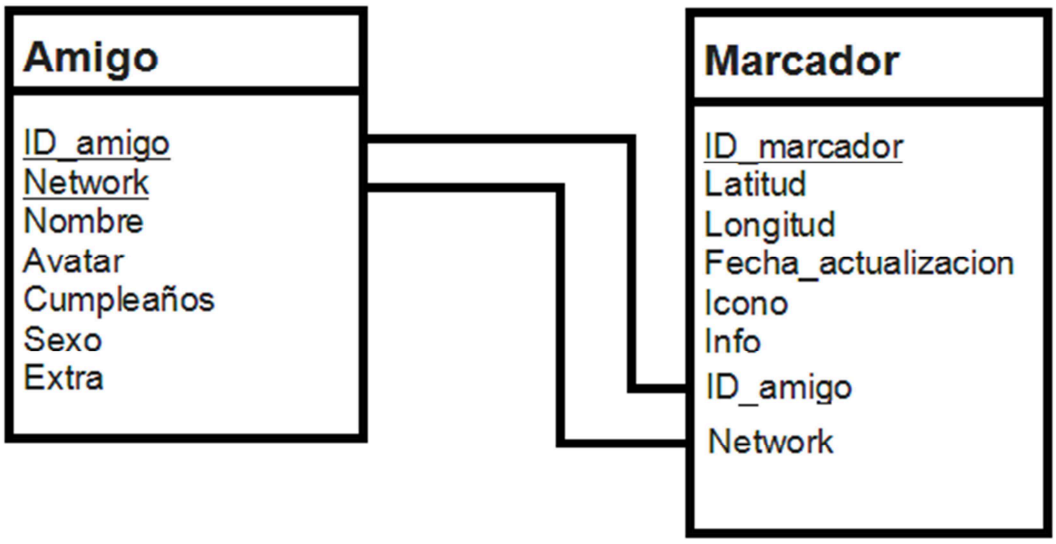
Amigo (ID_amigo, Network, Nombre , Avatar , Cumpleaños , Sexo , Extra)
Marcador (ID_marcador, Latitud , Longitud , Fecha actualización , Icono , Info)
Aparece en (ID_amigo , Network , ID_marcador) 1:N

Modelo relacional

Existe una relación entre la entidad Amigo y la entidad Marcador, que se llama "Aparece en" la cardinalidad es de uno a muchos, un solo Amigo aparece en cero o muchos Marcadores, por lo que , vamos a tomar la clave primaria del extremo "uno" (ID_Amigo, Network) e insertarlo en la tabla de extremo muchos (Marcador).

Amigo (ID_amigo, Network, Nombre , Avatar , Cumpleaños , Sexo , Extra)

Marcador (ID_marcador, Latitud, Longitud, Fecha_actualización , Icono , Info , ID_AMIGO , NETWORK)



upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Acceso a Facebook

Aquí voy a analizar las acciones que hay que desarrollar con la red social Facebook en nuestro proyecto. Entre las que se encuentran conectar con Facebook, insertar la opción de autentificación del usuario con la red social, acceso a la API y consulta de datos.

CONEXIÓN CON FACEBOOK

Aquí debemos inicializar y configurar el SDK. Es posible personalizar los parámetros utilizados. Todos las demás llamadas al SDK deben ser realizadas después de esto, porque no van a existir hasta que lo haga.

Name	Type	Description	Default
`appId`	`string`	*required* The Facebook application ID of the initializing app.	
`cookie`	`bool`	Sets a cookie which your server-side code can use to validate a user's Facebook session	`true`, contrary to the default in the Javascript SDK
`logging`	`bool`	*effective in Web Player only* If `true`, outputs a verbose log to the Javascript console to facilitate debugging.	`true`
`status`	`bool`	If `true`, attempts to initialize the Facebook object with valid session data.*	`true`
`xfbml`	`bool`	*effective in Web Player only* If `true`, Facebook will immediately parse any XFBML elements on the Facebook Canvas page hosting the app	`false`
`frictionlessRequests`	`bool`	Use frictionless app requests, as described in their own documentation.	`true`

Figura 14 –Parámetros de conexión con Facebook

Uno de los parámetros que necesitamos para inicializar el SDK de Facebook es el id de la aplicación que hemos creado anteriormente. Otros parámetros importantes son status, cookie, xfbml.

Una vez hecho esto se añade el botón de autenticación de Facebook.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

AUTENTIFICACION

Al hacer 'clic' en el botón de autenticación de la red social pide al usuario que autorice la aplicación utilizando el cuadro de diálogo de sesión a la plataforma. Si el usuario ya está conectado y ha autorizado su solicitud, comprueba si se han concedido todos los permisos requeridos, y si no, se le solicitan al usuario.

Por lo general, se le llama una vez para preguntar al usuario para la autenticación, luego otra vez para solicitar permisos adicionales según sea necesario.

Hay que verificar los permisos que el usuario haya concedido porque es posible que no haya aceptado todos los permisos que se han solicitado, ya que el usuario puede haber autorizado la aplicación para algunos pero no a otros. Los usuarios también pueden revocar los permisos a través de su configuración de privacidad, fuera de la aplicación.

Si tenemos la sesión abierta y volvemos a hacer 'clic' en el botón de autenticación cerraremos la sesión con el SDK de la red social. También anulará cualquier señal de acceso que usted tiene para el usuario que se emitió antes del cierre de sesión.

ACCESO AL API

Hacemos una llamada a la API de Facebook para obtener datos en nombre de un usuario. Esto se puede utilizar una vez que un usuario ha iniciado sesión y ha concedido los permisos codificados por el *token* de acceso que determinan qué llamadas al API estarán disponibles.

Dentro del API debemos realizar diferente acciones:

- CARGAR AMIGOS

El primer paso es hacer la consulta al API en la cual pedimos el id del amigo, la foto de perfil del amigo, su fecha de cumpleaños, y su sexo.

Guardamos cada registro de la consulta en la tabla Friends de nuestra base de datos. También mostramos la lista de amigos en pantalla.

- ACCEDER A LAS FOTOS

Una vez hemos obtenido la información de los amigos accedemos a sus fotos con otra consulta al API en la cual solo nos devolverá las foto que tengan localización.

La información que nos interesa obtener de las fotos y que pedimos en la consulta es el id de la foto, la *url* de la foto, el nombre de la foto y la localización.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Toda esta información se almacenara en nuestra base de datos para poder filtrarla con facilidad.

- RECORRER FOTOS Y ALMACENAR INFO DE MARCADORES

Recorremos las fotos de cada amigo y guardamos por cada foto un registro en la tabla marcador. Almacenamos el ID de la foto, el ID del amigo , la red social del objeto, la latitud , la longitud , la fecha de actualización , el link de la foto.

- CAMBIAR EL COLOR DEL MARCADOR

Dependiendo de qué tipo sea el objeto guardamos si es foto o estatus y dependiendo del tipo tendrá un marcador de un color, también le añadiremos el título al marcador.

- GUARDAR INFO DE MARCADORES

Almacenamos la estructura de tipo marcador que habíamos creado en la tabla marcadores de nuestra base de datos. También los mostramos en pantalla al cargar al autentificarse el usuario.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Interfaces del sistema

Interfaz inicial al entrar a la página.

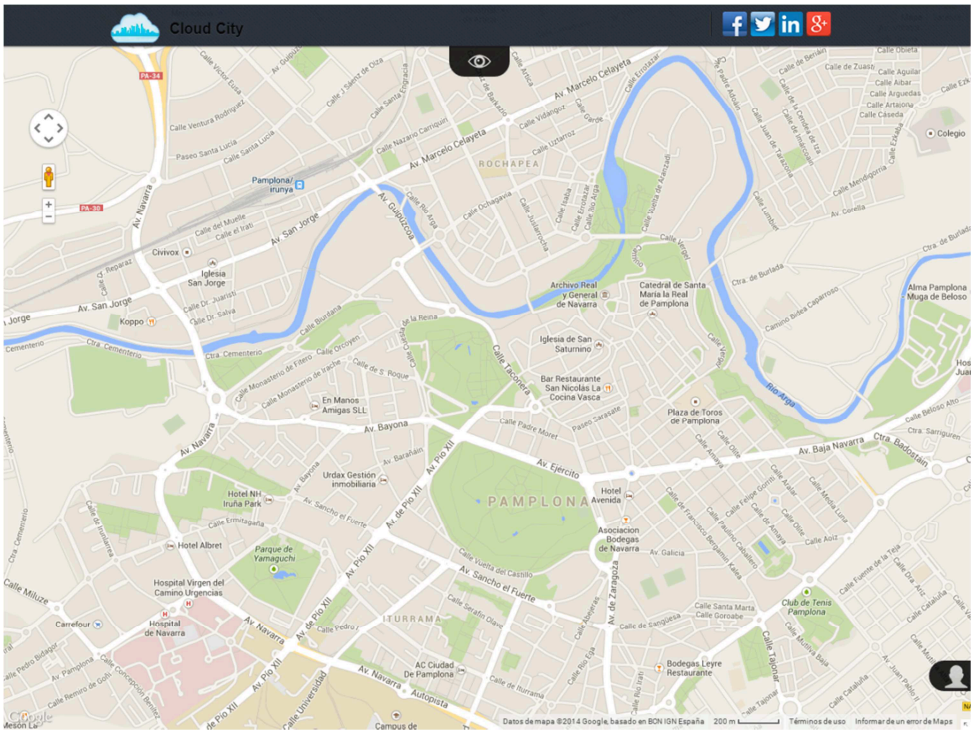


Figura 15 – Interfaz inicial

Interfaz de autenticación de Facebook.

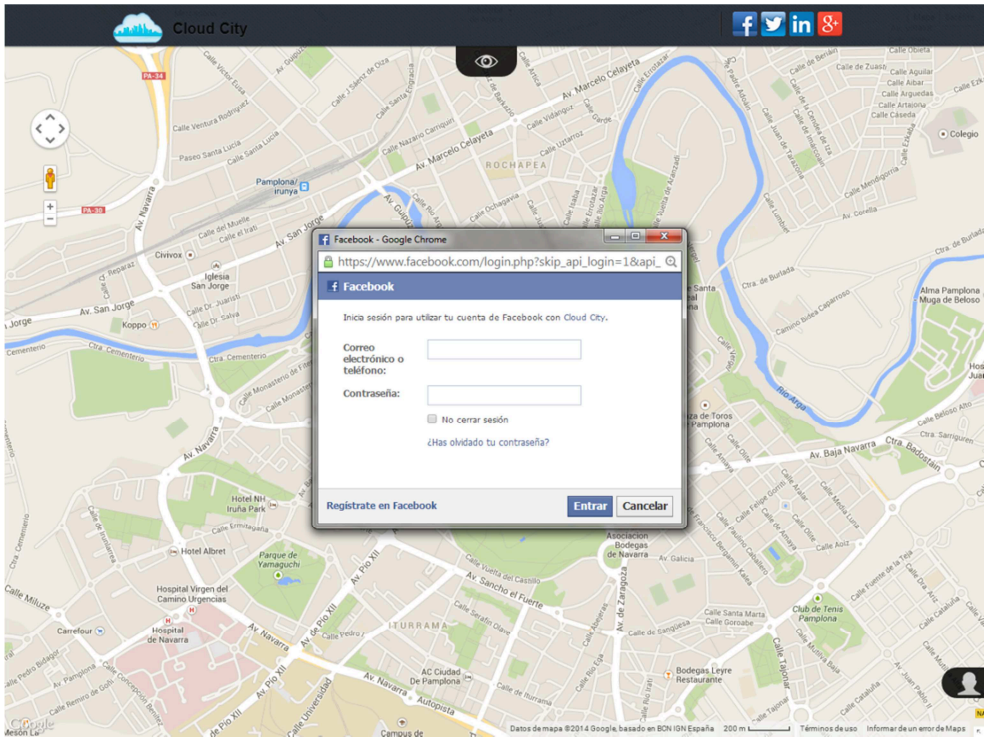


Figura 16 – Interfaz de autenticación

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Menú de filtros desplegado.

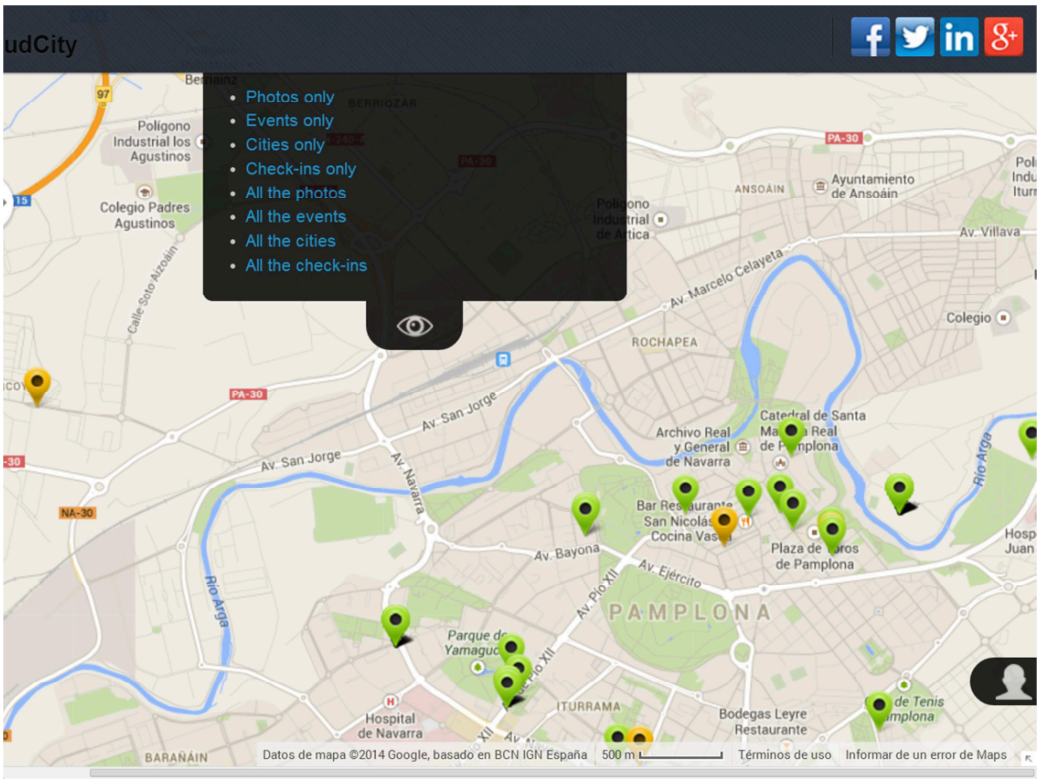


Figura 17 – Menú de filtros

Menú de amigos desplegado.

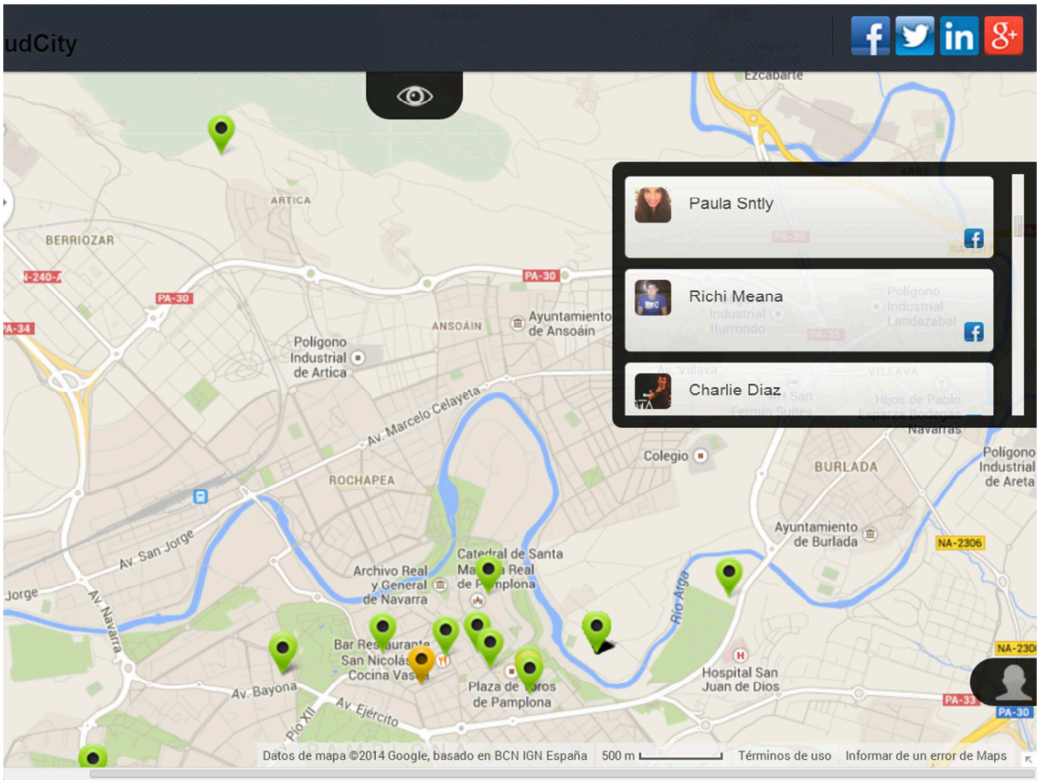


Figura 18 – Menú de amigos

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Ventana modal al seleccionar una foto.

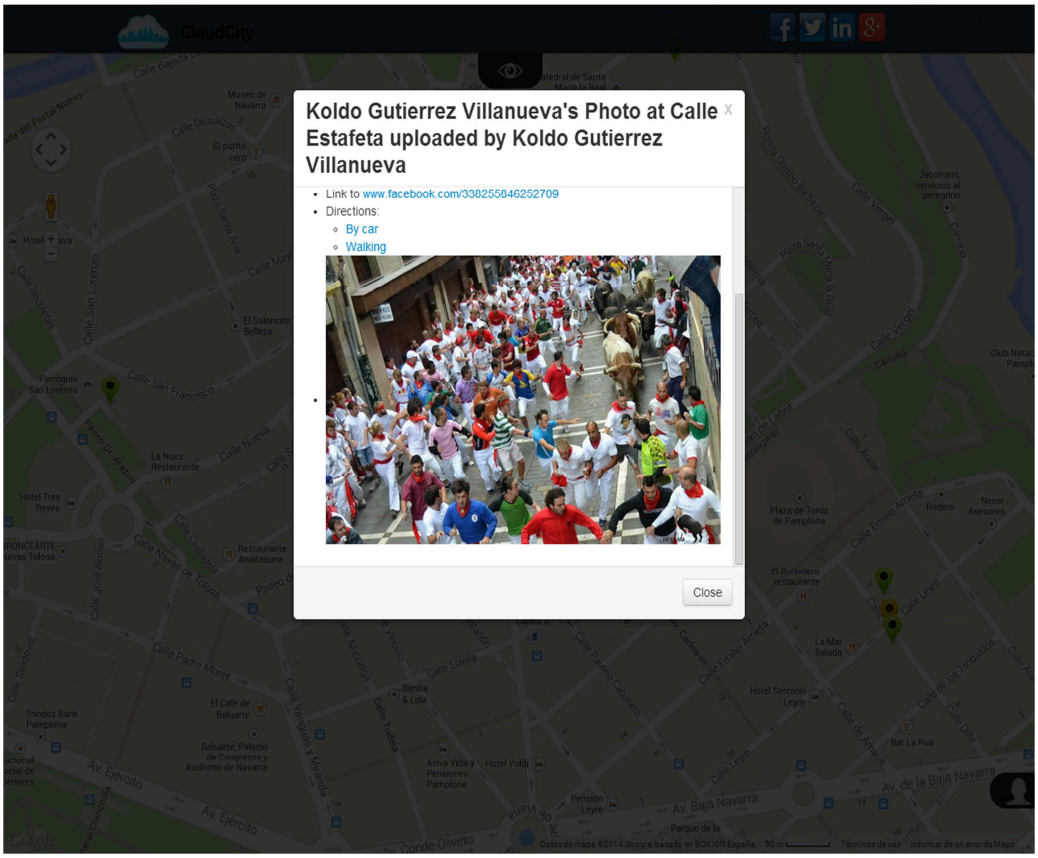


Figura 19 – Ventana Modal

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

6.- IMPLEMENTACION

Acceso a Facebook

Aquí voy a comentar la parte del código que me ha parecido más interesante del proyecto, que es el acceso al API de Facebook para obtener las fotos de los amigos y su almacenamiento y paso a marcadores.

CONEXIÓN CON FACEBOOK

En el método *init* recibe como parámetro la variable *options* en la cual está el id de la aplicación de Facebook necesario para la autentificación de la red social.se inicializa las variables de Facebook y se inicia la autentificación.
También se añade el botón de identificación de Facebook.

```
this.init = function (options) {
        $.getScript('http://connect.Facebook.net/en_US/all.js', function () {
                var defaults = {
                        appId: '',
                        status: true,
                        cookie: true,
                        xfbml: true
                };
                var opts = $.extend(defaults, options);
                FB.init(opts);
                FB.getLoginStatus(function (response) {
                        if (response.status == 'connected') {
                                loadFriends();
                        }
                });
        });

        $('.social-logins ul').append('<li class="Facebook-login"><a href=""
        onclick="Facebook.login(); return false;" title="Sign-in with Facebook"><img
        src="img/signin-buttons/Facebook_32.png" alt="Sign-in with Facebook" data-
        toggle="tooltip" /></a></li>');
};
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

AUTENTICACION

Comprueba si el usuario se ha logueado y si es así carga el método *LoadFriends(),*También están en *scope* los permisos que se necesitan.

```
this.login = function () {
        try {
                FB.login(function(response) {
                        if (response.authResponse) {
                                loadFriends();
                        }
                }, {scope: 'user_status,friends_status,user_checkins,
                friends_checkins,user_events,user_hometown,user_location,user_photos
                ,friends_hometown,friends_location,friends_photos,friends_events'}
                );
        } catch (e) {
                console.log(e);
        }
};
```

ACCESO AL API

```
FB.API('me/friends?fields=id,name,picture,birthday,gender', function(friendsQuery)
{

        var friends = friendsQuery["data"];

        for(var j=0; j < friends.length ; j++) {
                window.CloudCity.db.friends.insert({
                        id: friends[j]["id"],
                        network: 'Facebook',
                        name: friends[j]["name"],
                        avatar: friends[j]["picture"]["data"]["url"],
                        birthday: friends[j]["birthday"],
                        gender: friends[j]["gender"],
                        extra: friends[j]
                        });
```

El primer paso es hacer la consulta al API en la cual pedimos el id del amigo, la foto de perfil del amigo, su fecha de cumpleaños, y su sexo.

Guardamos cada registro de la consulta en Friends y lo recorremos con un *for* hasta la longitud de Friends. Dentro del bucle almacenamos todos los campos que

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

hemos obtenido en la consulta de cada amigo en la tabla *Friends* de nuestra base de datos.

ACCESO A LAS FOTOS

```
FB.API('/' + friends[j]["id"]+'/?fields=id,picture,name,locations.fields(id, type, place,
created_time,from)', function(locationsQuery) {

        var id = locationsQuery["id"];
        var name = locationsQuery["name"];
        var foto = locationsQuery["picture"];
        var locationInfo = locationsQuery["locations"]["data"];
```

Una vez dentro de un amigo accedemos a sus fotos con otra consulta al API en la cual obtenemos el id de la foto, la *url* de la foto, el nombre de la foto y la localización.

ALMACENAR INFO DE MARCADORES

```
For (var i=0; i < locationInfo.length ; i++) {

        var marker = {
                id: locationInfo[i]["id"],
                friend_id: id,
                friend_network: 'Facebook',
                lat: locationInfo[i]["place"]["location"]["latitude"],
                lng: locationInfo[i]["place"]["location"]["longitude"],
                time: new Date(locationInfo[i]["created_time"]),
                link: "www.Facebook.com/" + locationInfo[i]["id"],

        }
```

Creamos un bucle para recorrer las fotos de cada amigo y guardamos por cada foto una estructura de tipo marcador. Almacenamos el Id de la foto, el Id del amigo, la red social del objeto, la latitud, la longitud, la fecha de actualización y el link de la foto.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

CAMBIAR PARAMETROS DEL MARCADOR

```
if (locationInfo[i]["type"] == 'photo' ){
            marker.type = CloudMap.markerType.PHOTO;
            marker.title = CloudCity.__t(name + "'s Photo at " +
            locationInfo[i]["place"]["name"] + " uploaded by " +
            locationInfo[i]["from"]["name"]);
}

else if (locationInfo[i]["type"] == 'status' ){
          marker.type = CloudMap.markerType.CHECKIN;
          marker.title = CloudCity.__t(name + "'s Status at " +
          locationInfo[i]["place"]["name"] + " uploaded by " +
          locationInfo[i]["from"]["name"]);
}

else if (locationInfo[i]["type"] == 'post' ){
           marker.type = CloudMap.markerType.CHECKIN;
           marker.title = CloudCity.__t(name + "'s Post at " +
            locationInfo[i]["place"]["name"] + " uploaded by " +
            locationInfo[i]["from"]["name"]);
}
```

Dependiendo de qué tipo sea el objeto guardamos su tipo y dependiendo del tipo tendrá un marcador de un color y le ponemos el título al marcador.

GUARDAR Y MOSTRAR EL MARCADOR EN EL MAPA

```
CloudCity.db.markers.insert(marker);
CloudMap.showMarker(marker);
```

Almacenamos la estructura de tipo marcador que habíamos creado en la tabla marcadores de nuestra base de datos. También llamamos al método *showMarker* pasándole como parámetro el marcador para que lo inserte al mapa.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

VENTANA MODAL FOTO

```
this.PHOTO = function (marker) {
        var c = '';
        c += '<ul>';
        c += '<li>' + marker.title + '</li>';
        c += '<li>Id: ' + marker.id + '</li>';
        c += '<li>friend_id: ' + marker.friend_id + '</li>';
        c += '<li>friend_network: ' + marker.friend_network + '</li>';
        c += '<li>time (timestamp): ' + marker.time.toDateString() + '</li>'
        c += '<li>type: ' + marker.type + '</li>';
        c += '<li>Link to <a href="http://' + marker.link + '" target="_blank"> '  +
        marker.link + '</a></li>';


        FB.API('/' + marker.id + '?fields=source', function(imageQuery) {
                c += '<li><img src=' + imageQuery["source"] +'> </li>';
                c += '</ul>';
                CloudUI.modal(marker.title, c);
        });
};
```

Si el objeto es una foto se almacena en una variable de texto un código HTML con los datos de la foto y se accede al API para obtener la imagen. Por último se llama al método modal que muestra la ventana modal que muestra el código HTML anteriormente guardado.

VENTANA MODAL STATUS

```
this.CHECKIN = function (marker) {
        var c = '';
        c += '<ul>';
        c += '<li>' + marker.title + '</li>';
        c += '<li>Id: ' + marker.id + '</li>';
        c += '<li>friend_id: ' + marker.friend_id + '</li>';
        c += '<li>friend_network: ' + marker.friend_network + '</li>';
        c += '<li>time (timestamp): ' + marker.time.toDateString() + '</li>';
        c += '<li>type: ' + marker.type + '</li>';
        c += '<li>Link to <a href="http://' + marker.link + '" target="_blank">'
        + marker.link + '</a></li>';
        c += '</ul>';

        CloudUI.modal(marker.title, c);
};
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

BBDD

Utilizamos la base de datos en Javascript: *TaffyDB*

Lo primero es crear las tablas.

Nuestra tabla que almacena los amigos de Facebook se llamara *friends* y la tabla de los marcadores será *markers*.

```
CloudCity.db.friends = TAFFY();
CloudCity.db.markers = TAFFY();
```

Después creamos unas plantillas con valores por defecto de las dos tablas, que nos servirá a la hora de añadir un registro nuevo. Si es necesario se puede sobrescribir algún dato.

```
CloudCity.db.friends.settings({
  template: {
        id: 0,
        network: '',
        name: '',
        gender: '',
        birthday: '',
        age: 0,
        email: '',
        avatar: 'img/avatar-male.png',
        address: '',
        url: 'javascript: return false;',
        extra: {},
        deployed: false
  }
});
```

```
CloudCity.db.markers.settings({
  template: {
        id: 0,
        friend_id: 0,
        friend_network: '',
        time: 0, // timestamp!
        type: 0,
        animation: 0,
        clickable: true,
        cursor: 'pointer',
        draggable: false,
        flat: false,
        icon: '',
        optimized: false,
        lat: 0,
        lng: 0,
        raiseOnDrag: true,
        shadow: null,
        title: '',
        visible: true,
        zIndex: 99,
        link: '',
        extra: {}
  }
});
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Por ultimo insertamos registros a las tablas con los datos del API de Facebook.

```
window.CloudCity.db.friends.insert({
        id: friends[j]["id"],
        network: 'Facebook',
        name: friends[j]["name"],
        avatar: friends[j]["picture"]["data"]["url"],
        birthday: friends[j]["birthday"],
        gender: friends[j]["gender"],
        extra: friends[j]
});
```

```
var marker = {
        id: locationInfo[i]["id"],
        friend_id: id,
        friend_network: 'Facebook',
        lat: locationInfo[i]["place"]["location"]["latitude"],
        lng: locationInfo[i]["place"]["location"]["longitude"],
        time: new Date(locationInfo[i]["created_time"]), //updated_time
        link: "www.Facebook.com/" + locationInfo[i]["id"],
        extra: locationInfo[i],
        icon: Logo
}


CloudCity.db.markers.insert(marker);
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Mapa

En el método *init* que inicializan los parámetros del mapa por defecto como son el bloque en el que va a ir el mapa, la latitud, la longitud, el zoom, el tipo de mapa, y los controles del mapa.

Después creamos un objeto de tipo *GMaps* de Google Maps en el cual recibe los parámetros que habíamos designado anteriormente. Y por último se optimizan las opciones de Street view que ofrece Google Maps para mostrar imágenes del mapa.

```
init: function(options) {
  var self = this;
  var defaults = {
    el: 'theMap',
    lat: -12.043333,
    lng: -77.028333,
    zoom: 15,
    mapType: 'Roadmap',
    streetViewControl: true,
    zoomControl: true,
    zoomControlOpt: {
      style: 'SMALL',
      position: 'TOP_LEFT'
    }
  };
  self.options = $.extend(defaults, options);
  self.GMaps = new GMaps(self.options);

  self.GMaps.map.getStreetView().set('enableCloseButton', false);
  Google.Maps.event.addListener(self.GMaps.map.getStreetView(), 'visible_changed',
  self.streetViewStarted);
},
```

A la hora de cargar la página web llamamos al método *init* de CloudMap y sobrescribimos los valores del nombre del bloque donde se inserta el mapa, la latitud y la longitud.

```
CloudMap.init({
                el: '#map_canvas',
                lat: 42.81943978990297,
                lng: -1.6511842649597352
        });
```

Este es bloque en el que se añade el mapa.

```
<div id="map_canvas"> </div>
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

7.- PRUEBAS DEL SISTEMA

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Es una actividad más en el proceso de control de calidad.

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.

Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo.

Pruebas de unidad

Se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software. Las pruebas de unidad se concentran en la lógica del procesamiento interno.
Este tipo de pruebas pueden realizarse en paralelo a varios componentes.

Pruebas de integración

La prueba de integración es una técnica sistemática para construir la arquitectura del software. El objetivo es tomar componentes a los que se les aplico una prueba de unidad y construir una estructura de programa que determine el diseño.

Pruebas de validación

La validación se define de una forma simple en que se alcanza cuando el software funciona de tal manera que satisface las expectativas razonables del cliente.

Pruebas de sistema

Al final del desarrollo el software se incorpora a otros elementos del sistema (hardware, personas, información) y se realiza una serie de pruebas de integración del sistema y de validación.

Estas pruebas están más allá del alcance del proceso del software y ni las realiza únicamente los ingenieros del software.

Sin embargo, los pasos dados durante el diseño y la prueba del software mejoraran en gran medida la probabilidad del tener éxito en la integración del software del sistema mayor.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Las pruebas de sistema se dividen en tres tipos:

- Pruebas de Seguridad

  La prueba de seguridad comprueba que los mecanismos de protección integrados en el sistema realmente protejan de irrupciones inapropiadas.

  Durante la prueba de seguridad quien la aplica desempeña el papel del individuo que desea entrar en el sistema.

- Pruebas de Resistencia

  Las pruebas de resistencia están diseñadas para confrontar los programas con situaciones anormales. Ejecuta un sistema de tal manera que requiera un volumen anormal de recursos. Hay que intentar sobrecargar el sistema.

- Pruebas de Desempeño

  Están diseñadas para probar el desempeño del software en tiempo de ejecución dentro del contexto de un sistema.

  La prueba de desempeño se aplica en todos los pasos del proceso de la prueba, incluso al nivel de la unidad, el desempeño de un módulo individual debe evaluarse mientras se realizan las pruebas.

Pruebas de aceptación

Finalmente el usuario comprueba si en su propio entorno de explotación si lo acepta como esta o no.

PRUEBAS DEL SISTEMAS QUE HEMOS REALIZADO

Hemos realizado las siguientes pruebas del sistema mientras se desarrollaba el sistema para comprobar su funcionamiento y su rendimiento.

- Pruebas de Seguridad

  Es difícil acceder al nuestro sistema porque la única forma de entrar datos es la autenticación de Facebook que tiene su propio sistema de seguridad.

  Así que no es sencillo que haya una irrupción inapropiada en el sistema.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- Pruebas de Resistencia

  Se han realizado pruebas de autentificaciones de cuentas en Facebook con muchos objetos (fotos, status, etc…) a mostrar en pantalla y el sistema no se sobrecarga.

- Pruebas de Desempeño

  Las pruebas de desempeño quizás sean en las que más se han profundizado. Se han realizado pruebas con diferentes usuarios para ver si el sistema es funcional y sencillo y el interfaz es amigable.

  También se han realizado pruebas desde diferentes navegadores y desde diferentes host para comprobar que la página web no se descuadraba ni había ningún problema con algún navegador específico.

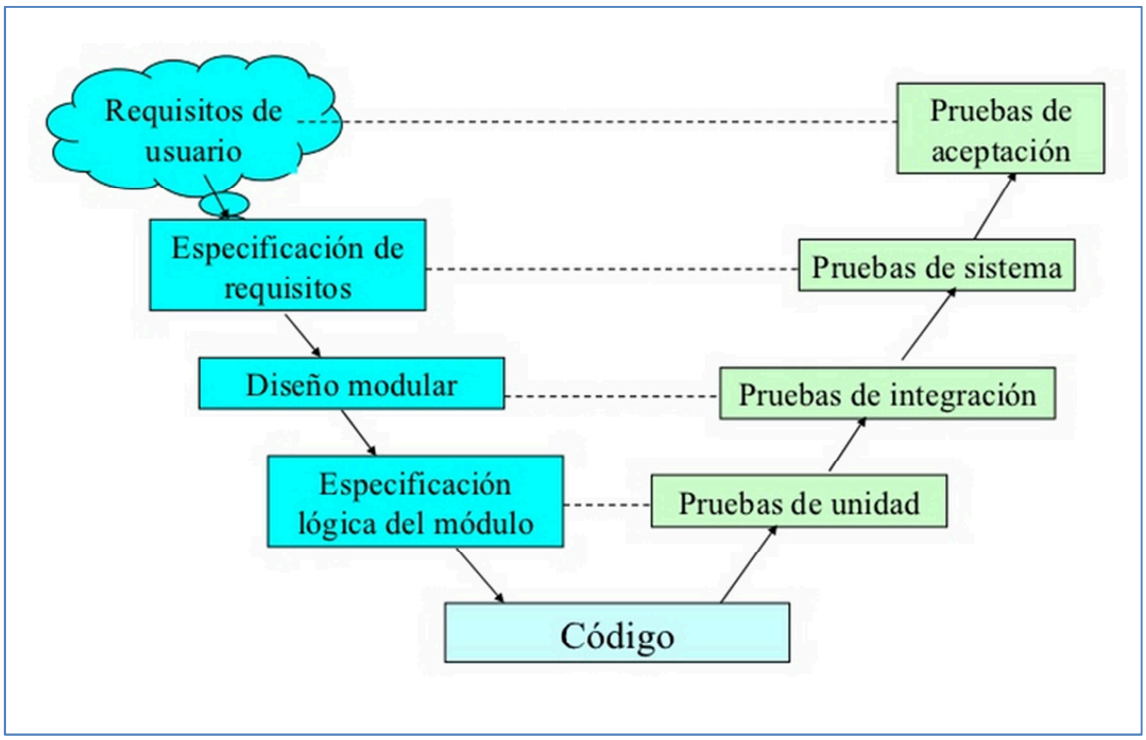


Figura 20 – Esquema de las pruebas

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

8.- DESPLIEGUE DEL SISTEMA

Primero actuamos en servidor local

Para desarrollar la página web en un comienzo trabajamos en un entorno local para no tener que contratar ningún hosting. El primer paso fue instalar un servidor local en nuestro PC para trabajar en la web .Estos fueron los pasos que seguimos:

- APPSERV

  Appserv es una herramienta *Open Source* para Windows con Apache, MySQL, PHP, estas aplicaciones se configuran de forma automática, lo que permite ejecutar un servidor web completo.
  Como extra incorpora phpMyAdmin para el manejo de MySQL

  Instalamos Appserv debido a que creíamos que trabajaríamos con PHP pero al final no utilizamos este lenguaje.

Figura 20 – Imagen de AppServ

- GOOGLE MAPS

  En local probamos y configuramos todo lo relacionado con Google Maps como seleccionar las librerías necesarias, mostrar un mapa , saber trabajar con los marcadores: insertar, eliminar, ocultar, mostrar información, etc…

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- HTML

  También en local trabajamos la estructuración de la página web con HTML en más tarde se modificaría.

- FACEBOOK APP

  Al introducirnos con Facebook nos requiere una APP para conectarnos con la red social. La configuración de la app necesita que le demos un dominio desde el cual se va a conectar a Facebook por lo cual ya no podemos trabajar desde local. Necesitamos un Hosting.

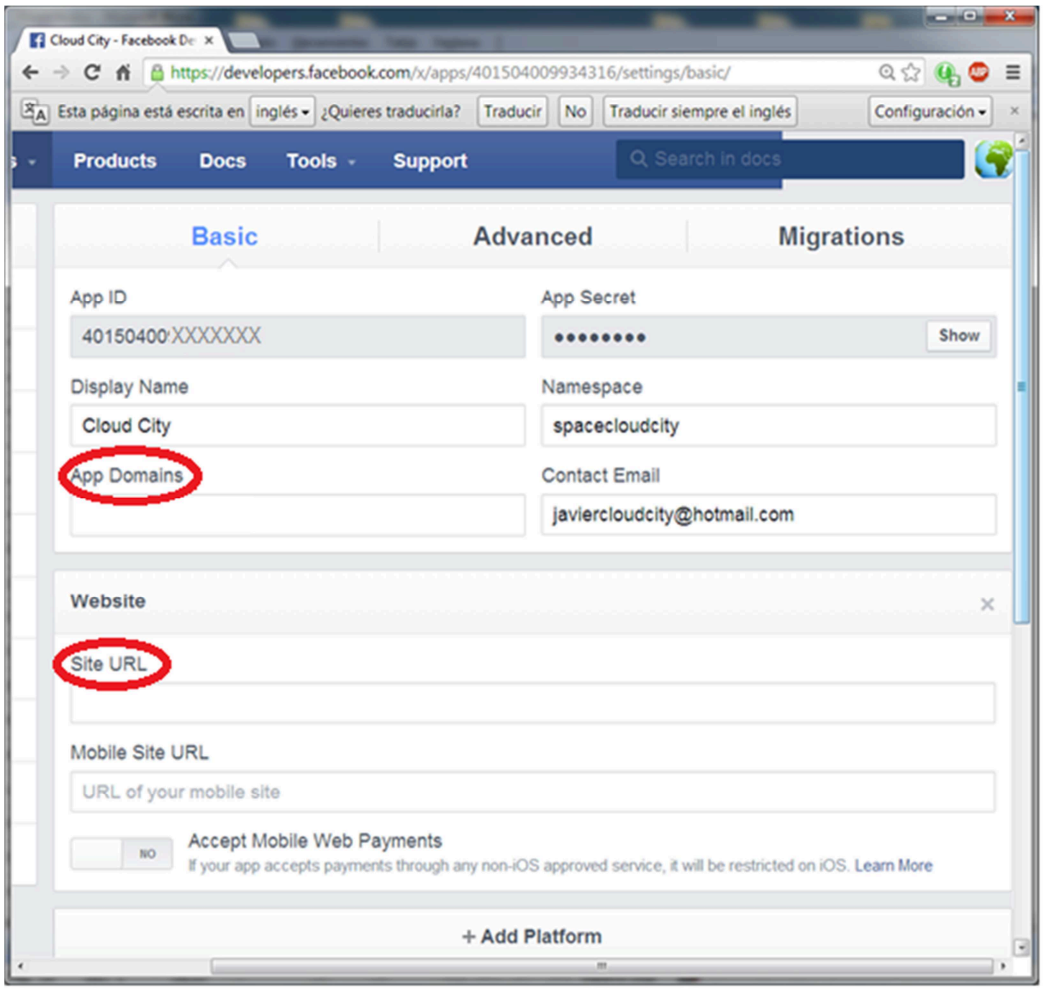


Figura 21 – Configuración de App de Facebook

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Pruebas con heroku

Para solucionar el problema con la APP de Facebook intentamos trabajar con Heroku.

Heroku es un servicio de hosting en la nube. Los clientes no tiene que contar con infraestructura, el tiempo de procesamiento y almacenamiento se le renta a un tercero.

Se trata de un sistema gratuito hasta los 5 MB y su implementación se hace a través de Git.

Al intentar poner como hosting el de Heroku surgieron problemas , al parecer Facebook lo detectaba como un sistema de correo basura y no permitía poner esa *url*.

Por este problema y por ser un sistema con el que no estaba muy familiarizado decidimos trabajar con un hosting gratuito.

Al final hosting gratuito

Después de los problemas surgidos con Heroku buscamos información sobre hosting gratuitos y encontramos el servicio de hosting 000webhost.com.

Figura 22 – Logo de hosting gratuito

Rellenamos un formulario de solicitud para un subdominio gratuito y nos proporcionar el siguiente: Cloudcity.comuf.com

El servicio de host nos da el subdominio y un gestor de ficheros para poder trabajar el subdominio.

Con este hosting se solucionan los problemas con la App de Facebook y la página web comienza a funcionar bien.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

9.- METODOLOGIA DE GESTION DEL PROYECTO

La idea original del proyecto es de David Azcona y Niels McEvoy que se pusieron en contacto con la universidad en busca de un alumno que querría realizar el proyecto de fin de curso con ellos.

David se pone en contacto conmigo y decidimos cual va a ser mi trabajo y delimitamos mi proyecto de fin de curso.

Debido a que David y Niels se encuentran en Irlanda las reuniones han tenido que ser por Skype y la comunicación por correo electrónico. También compartimos archivos importantes del proyecto en la nube de Gmail, que podíamos actualizar cualquiera de los tres.

El diseño de la página fue a cargo de Chris, alumno de grado.

Para la realización del proyecto seguí los pasos que me marcaba David mediante correo electrónico. Estos fueron mis pasos a lo largo del proyecto:

- CREAR ESTRUCTURA HTML

  Diseño de una estructura en bloques de la página Web.
  Estructuración de la página en diferentes bloques con una página de estilos.
  La mayor parte de la página muestra el mapa de Google Maps y otro bloque contendrá los amigos de Facebook.

- PRIMEROS CONTACTOS CON LOS API

  Documentación, lectura y pruebas con los APIs de Google Maps y Facebook.

- CREARSE CUENTA DE PROGRAMADOR PARA PODER CONECTAR CON APIS

  En Google Maps requiere una cuenta de desarrollador y en Facebook crear una aplicación en Facebook.

- INSERTAR MAPA DE GOOGLE MAPS

  Inclusión de un mapa de Google Maps.

- AÑADIR Y MODIFICAR MARCADORES

  Realización de pruebas para añadir marcadores en el mapa. Modificando la localización, tipo de marcador, eventos, etc…

- AUTENTIFICACION DE FACEBOOK

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Surgen problemas al hacer la autentificación con Facebook debido al uso del servidor Heroku. Facebook lo detecta como spam.

- CONSEGUIR DOMINIO GRATUITO

  Conseguimos un nuevo hosting en un proveedor gratuito y al alojar nuestra página en él se resuelven los problemas con la autentificación de Facebook. El dominio de nuestra página es http://www.cloudcity.net63.net/.

- OBTENER LISTADO DE AMIGOS FACEBOOK

  He obtenido el listado de amigos de la API de Facebook y los he añadido en la *sidebar* en la parte derecha de la página web.

- MOSTRAR MARCADORES DE AMIGOS DE UNO EN UNO

  Mostramos los marcadores de uno en uno al pinchar en la lista de amigos Cuando clicamos en uno de ellos en la *sidebar* de amigos vemos la cuidad actual en la que se encuentra.

- MOSTRAR NUEVAS MARCAS EN EL MAPA

  Las nuevas marcas que pondremos en el mapa serán: ciudad de nacimiento, fotos, eventos, *checkins* del amigo/s en cuestión que seleccionemos.

- VENTANA MODAL CON INFORMACION DE MARCADOR

  Mostramos la información del marcador en la ventana de información de Google Maps. Pero al final decidimos crear nuestra propia ventana modal.

  Para visualizar la información del punto seleccionado aparecerá una ventana en la parte derecha de la pantalla.

- MOSTRAR TODOS LOS MARCADORES DE AMIGOS A LA VEZ

  Decidimos que en vez de seleccionar el amigo en el *sidebar* para mostrar su información es mejor cargar la de todos los amigos al loguearse.

  Para acceder de una forma rápida a la información de los amigos y debido a que se accede a mucha información y a la importancia de que debe funcionar rápido modificamos la consulta al API de Facebook y en ella desechamos los eventos para que la consulta sea más eficaz y la página tenga un rendimiento mayor.

- FILTRADO DE MARCADORES

  Creamos un *sidebar* para poder filtrar los marcadores por fotos, localidades y *checkins*.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

10.- CONCLUSIONES

El objetivo general de este proyecto fue realizar una red social basada en la localización y eso se ha cumplido.

Entre los puntos que se vieron para cumplir con el objetivo general se puede concluir que:

- Realizando el proyecto me he familiarizado con el trabajo con los APIs muy especialmente con el de Facebook
- También he conseguido mejorar aptitudes con lenguajes como HTML, CSS, Javascript, etc…
- Al hacer el proyecto para una empresa externa y no directamente para la universidad aprendí a trabajar de una forma más real y cercana con la empresa y obtener una gran experiencia enfocada a salir al mundo laboral.
- He obtenido y gestionado un Hosting para alojar el sitio WEB.
- Adquirir conocimientos para introducir en los navegadores bases de datos locales con *TaffyDB*.

En conclusión queda la confianza de haber realizado un proyecto bien hecho y confiable y que el mundo de los mashups tiene una gran tesitura de posibilidades. Con un poco de trabajo con los APIs puedes construir un sitio web con mucha información y muchas posibilidades.

Con una buena idea y uniendo información de dos o más focos de información se puede desarrollar un sitio web con mucha utilidad.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

11.- LINEAS FUTURAS

La idea original de la empresa es mayor de la que había en mi proyecto, en la idea original se conectaban datos de varias redes sociales y esa es el siguiente objetivo a cumplir en el proyecto.

Los principales objetivos futuros son los siguientes:

- Implementar el proyecto con más redes sociales como puede ser Twitter Foursquare, Linkedin.
- Añadir como llegar desde tu localización hasta un marcador seleccionado.
- Incorporar una línea del tiempo para poder seleccionar la fecha desde la cual se quieren mostrar los marcadores .Desde hace 2 días hasta hace 6 meses.
- Poder añadir localización a fotos que no la tengan directamente arrastrando la foto hasta el mapa.
- Adaptar la aplicación para dispositivos móviles como pueden ser iPhones, iPads, *tablets*, sistemas androids, etc…
- Adaptar la accesibilidad del sistema para permitir su uso a personas con limitaciones físicas, psíquicas o sensoriales.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

12.- BIBLIOGRAFIA

Libros

1. Luc Van Lancker. *Los API Javascript de HTML5.Ediciones Eni. 978-2-7460-8291-5*

Sitios web

2. Versión 3 del API de Javascript de Google Maps.

   Disponible en: https://developers.Google.com/Maps/documentation/Javascript/tutorial?hl=es

3. Facebook for Developers

   Disponible en: https://developers.Facebook.com/

4. Stackoverflow

   Disponible en: http://stackoverflow.com/questions/11840555/Facebook-API-login-logging

5. Digitta

   Disponible en: http://digitta.com/2009/05/taffydb.HTML

Blogs

6. Iñaki Huerta. *Ikhuerta* [blog]. Consultado en: Abril 2013

Disponible en: http://blog.ikhuerta.com/usando-Facebook-graph-API-con-Javascript-sdk-pero-aun-mas-sencillo

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Nueva red social con integración de servicios basados en la localización

Javier Marquina Salinas
Proyecto Fin de carrera

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

¿Quiénes somos?

- Empresa innovadora
- Creada por David Azcona y Niels McEvoy
- Nueva red social

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Idea original

- Pagina web que une información de varias redes sociales.
- La información debe tener localización.
- Todo se muestra en un mapa.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Mi proyecto

- No abarca toda la idea original.
- Posee la estructura general.
- Implementado para la Facebook.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Sistema similares

- FourSquare
- Facebook Nearby
- Yelp
- TagTagCity

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Arquitectura del sistema

- Mashup
- Proveedor de servicios : API
- Mashup de consumidor : enfocado al usuario final.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Manejo de APIs

- Acceso a Información
- Datos Facebook
- Utilidades Google Maps
- APIs de Facebook y Google Maps

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

API Facebook

- Crear Aplicación
- Autenticación
- Realizar consultas

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

API Google Maps

- Clave de API
- Mapa y configuración
- Marcadores
- Información en los marcadores

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Requisitos del sistema

- Hosting
- Facebook
- Base de Datos
- Interfaz grafica
- Mapa

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Hosting

- Servidor
- Nombre del dominio
- Subdominio gratuito
- Cloudcity.comuf.com

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Facebook

- Conexión con Facebook
- Autentificación
- API
- Fotos,Status

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Base de Datos

- Almacenar dos colecciones.
- Amigos y Marcadores.
- Taffy DB

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Interfaz Grafico

- Cargar las librerías
- Menú amigos
- Menú filtros
- Ventana modal

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Mapa

- Cargar mapa
- Filtro de marcadores
- Mostrar marcadores
- Ocultar marcadores

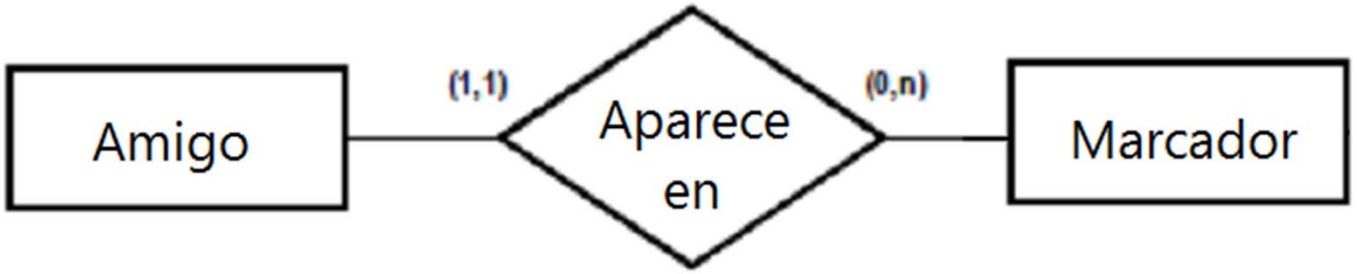
upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Análisis del sistema

- Diagrama Entidad-Relación
- Modelo Relacional

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Diagrama Entidad-Relación



Amigo

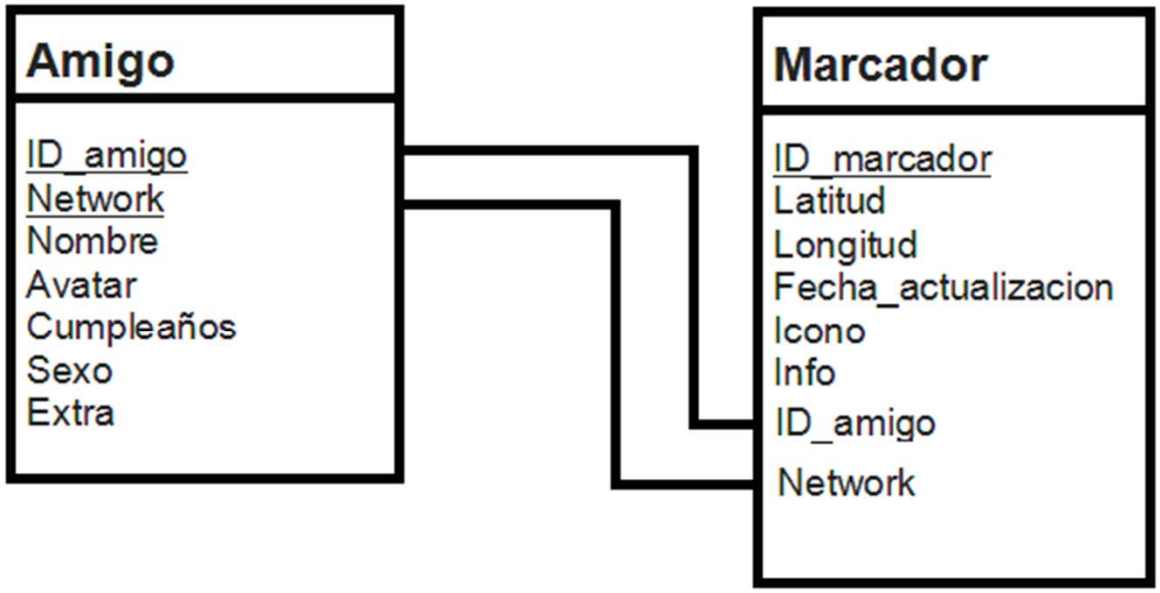
ID_amigo
Network
Nombre
Avatar
Cumpleaños
Sexo
Extra

Marcador

ID_marcador
Latitud
Longitud
Fecha_actualizacion
Icono
Info

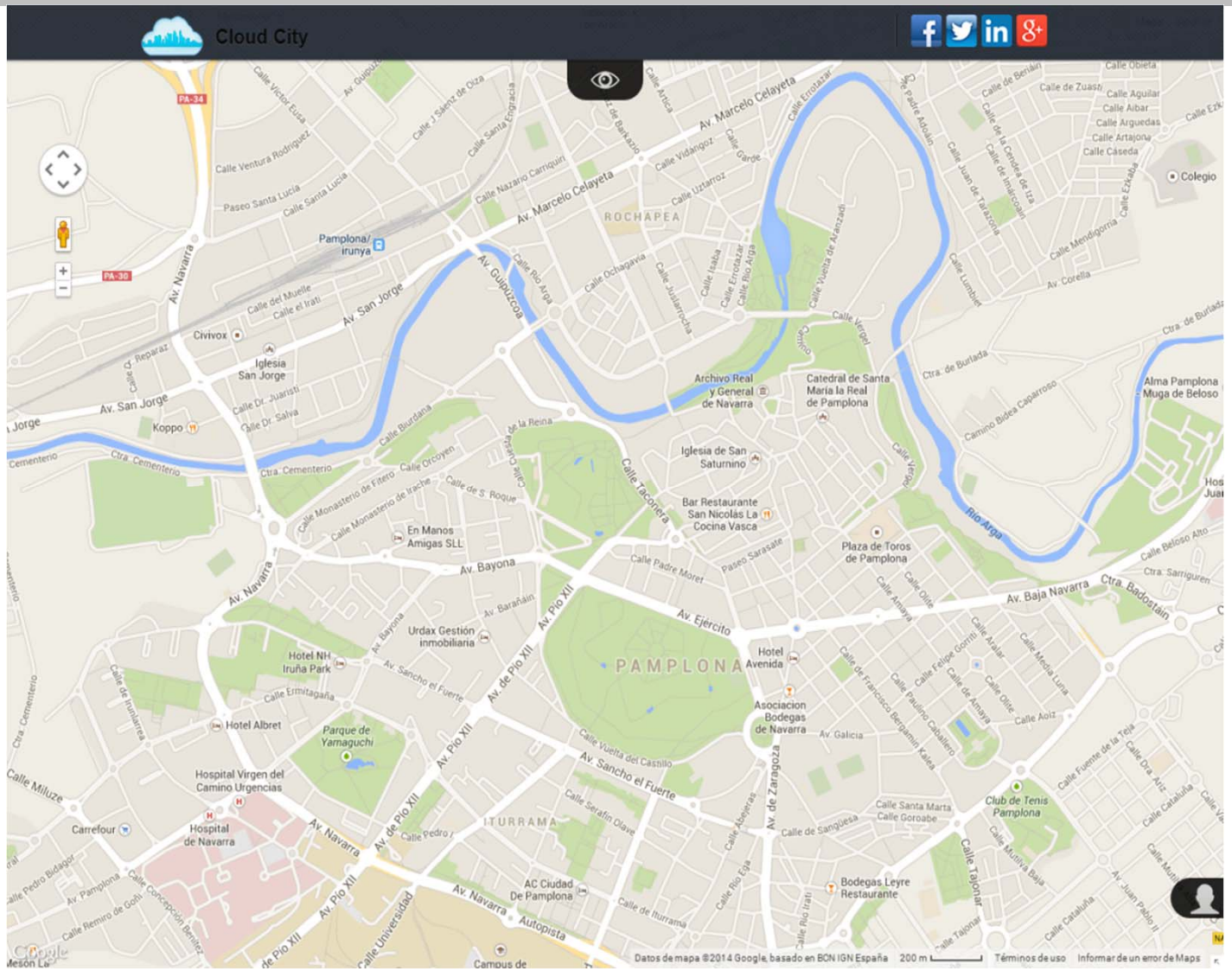
upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Modelo Relacional



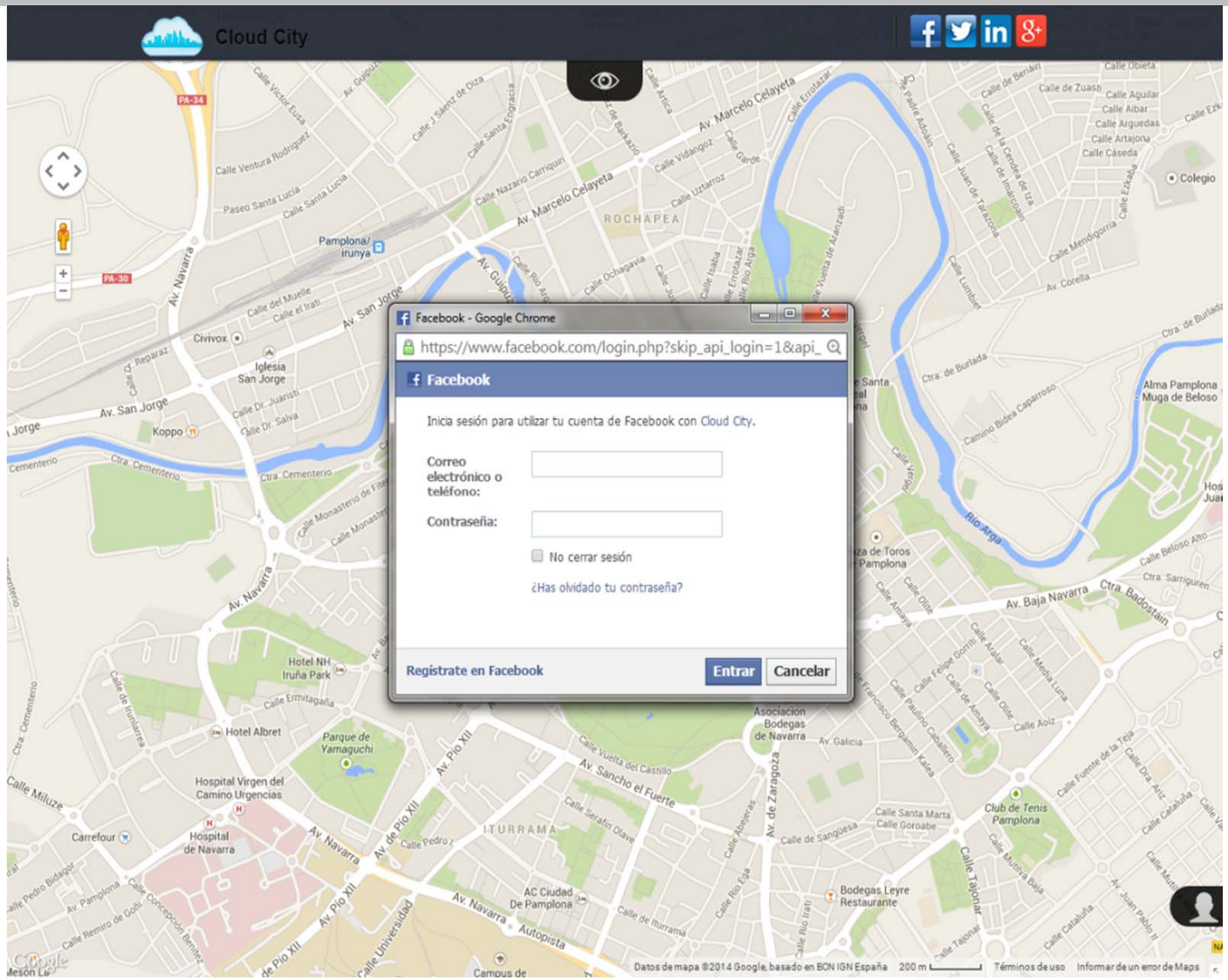
upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Interfaces del Sistema



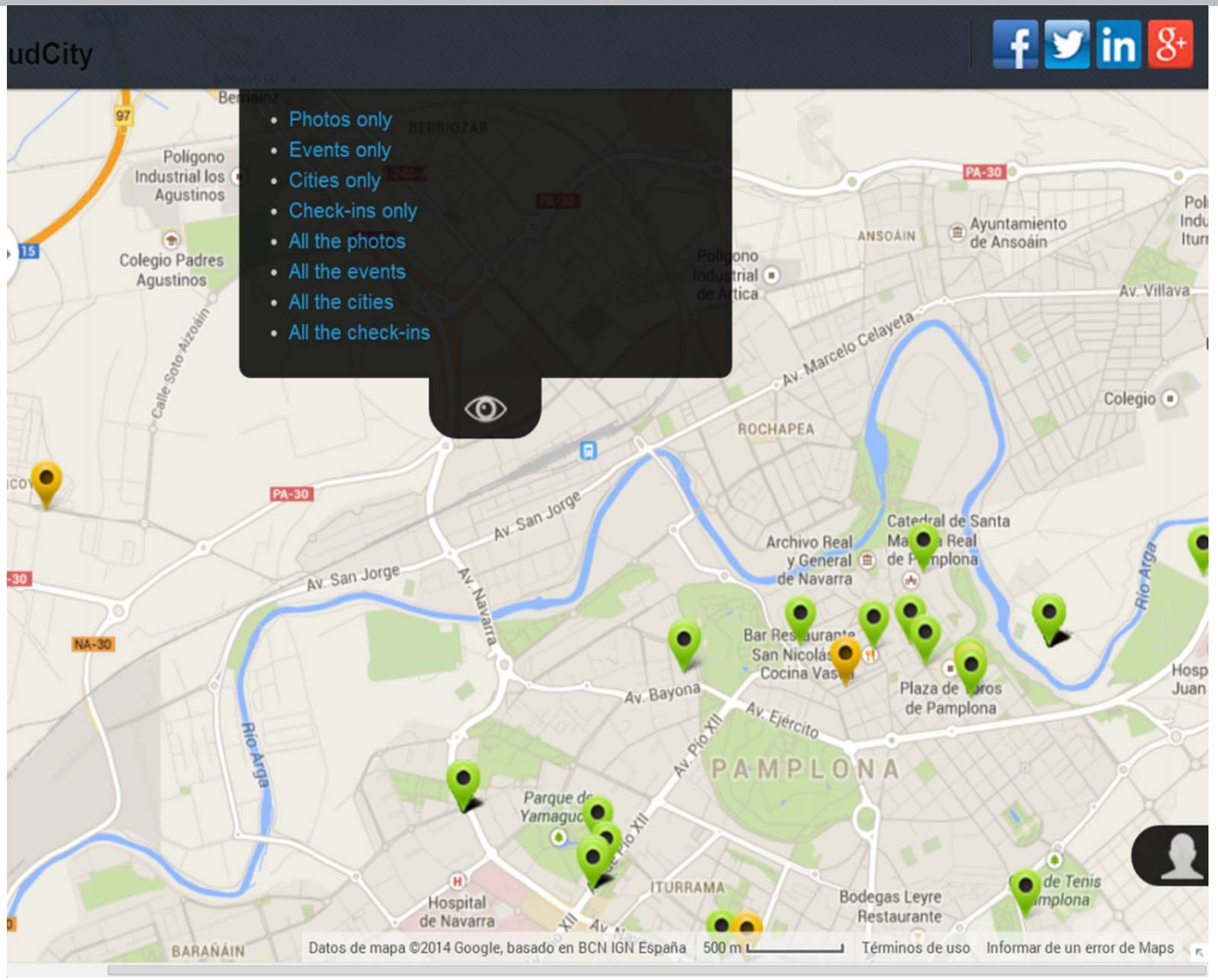
upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Interfaces del Sistema



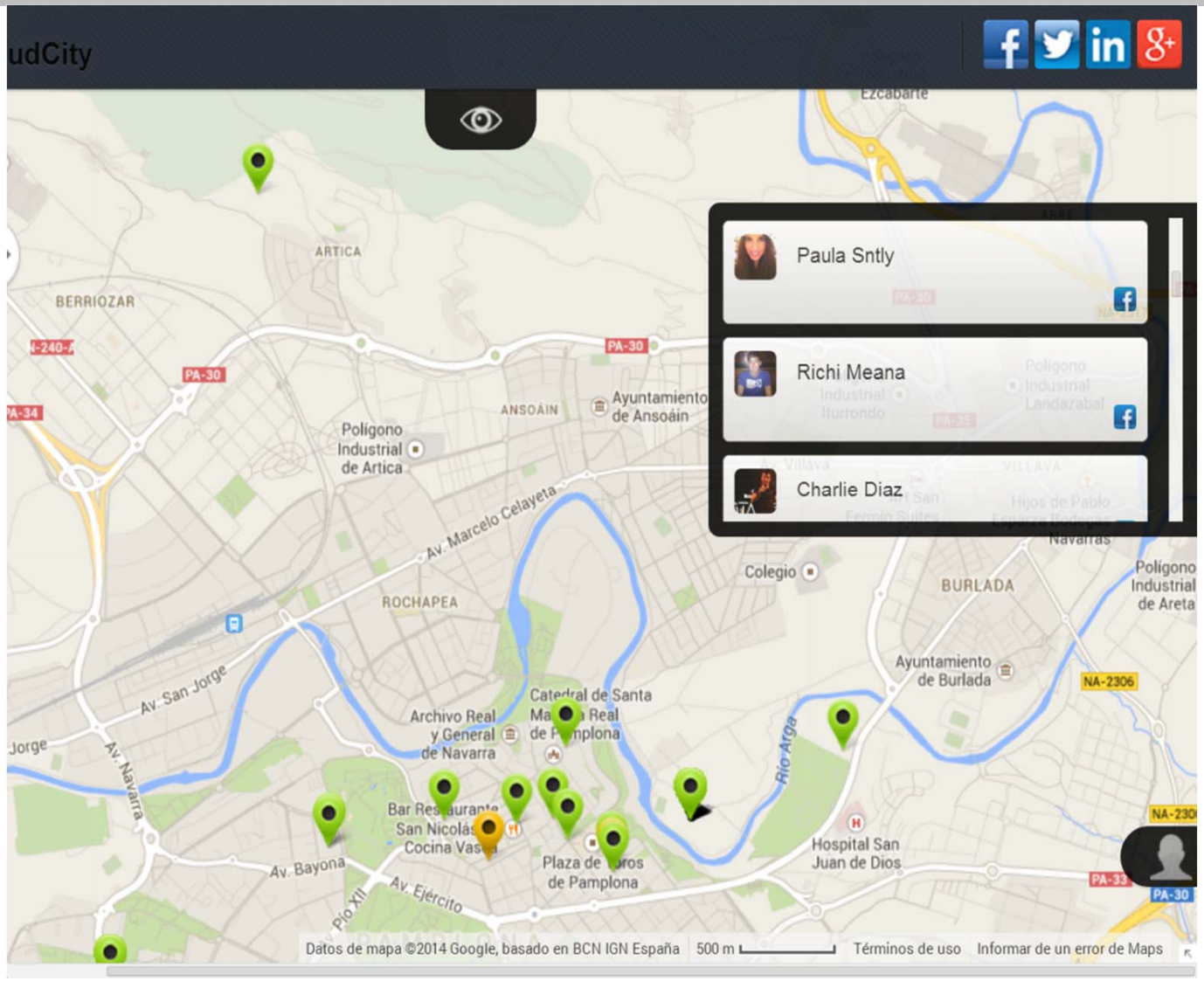
upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Interfaces del Sistema



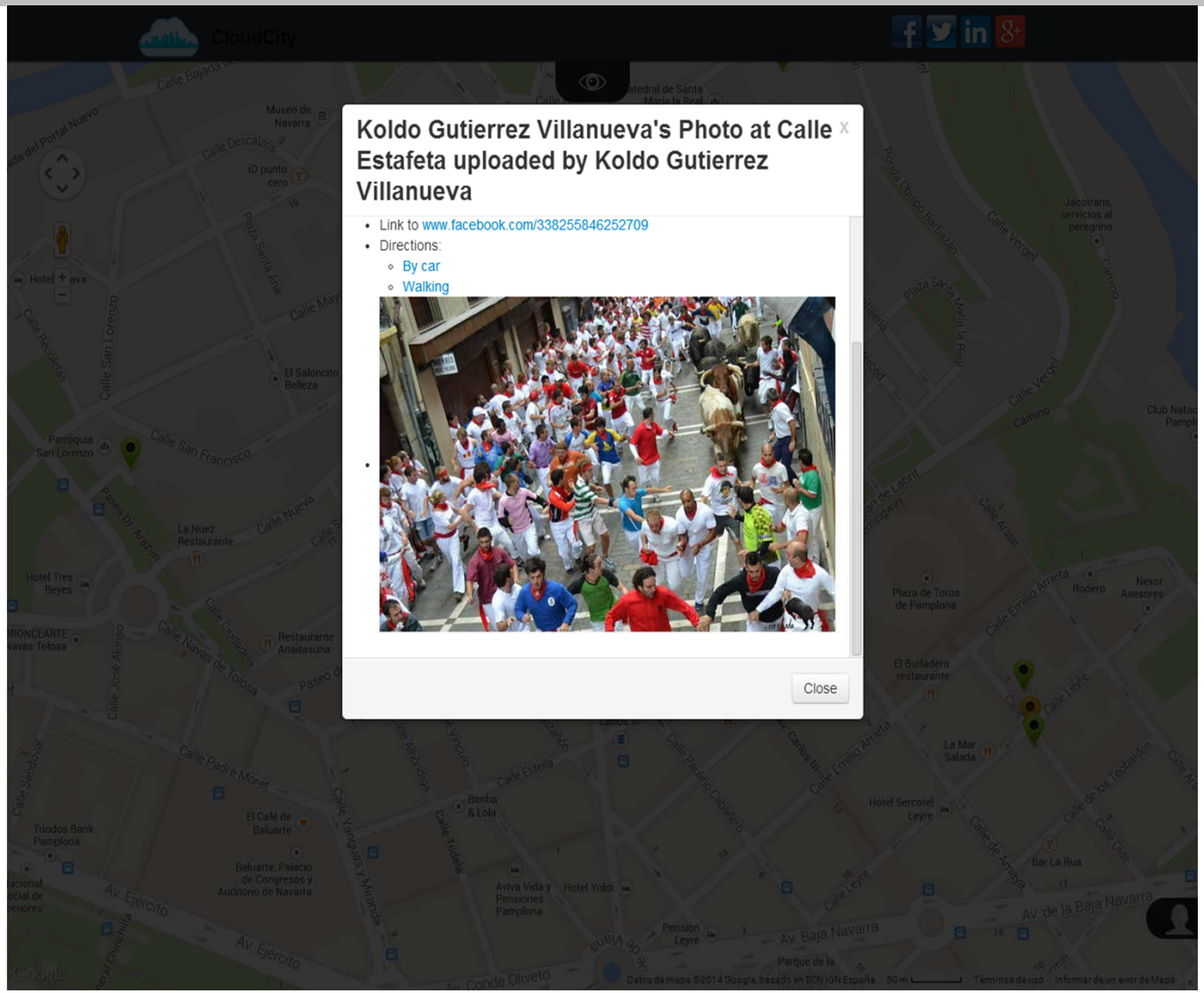
upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Interfaces del Sistema



upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Interfaces del Sistema



upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Pruebas del sistema

El objetivo es proporcionar información sobre la calidad del producto.

- Pruebas de Seguridad
- Pruebas de Resistencia
- Pruebas de Desempeño

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Despliegue del sistema

- AppServ . Servidor Local
  - Google Maps
  - HTML
  - Facebook App
- Heroku. Plataforma en la nube
- 000webhost.com. Hosting gratuito

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Metodología

La idea original del proyecto es de David Azcona y Niels McEvoy que se encuentran en Irlanda.

- Reuniones por Skype .
- Comunicación por correo electrónico.
- Compartir archivos en la nube.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Metodología

- Estructura HTML
- Cuenta de programador API
- Insertar mapa de Google Maps
- Autentificacion de Facebook
- Adquirir Dominio

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Metodología

- Listado de amigos Facebook
- Mostrar marcadores
- Mostrar nuevas marcas
- Ventana modal
- Filtrado de marcadores

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Conclusiones

- Mashups
- APIs
- Lenguajes HTML, CSS, Javascript, etc…

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Conclusiones

- Empresa externa.
- Gran experiencia.
- Gestionar Hosting para sitio WEB.
- Base de datos local en navegadores.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Líneas futuras

- Más redes sociales.
- Como llegar.
- Eventos.
- Línea del tiempo.
- Añadir localización a fotos.
- Dispositivos móviles.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Muchas Gracias

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira