E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Implementación de una web Single Page App para proyecto de emprendedores TeamSurfing.com

Grado en Ingeniería Informática

Trabajo Fin de Grado

Oliver Sanz Catalán
Oscar Ardaiz Villanueva
Pamplona, 27 de junio de 2014

upna Universidad Pública de Navarra
Nafarroako

AGRADECIMIENTOS

Quiero dedicar esta página entera para Aitor, Alba, Alicia, Ana, Bambu, Carlos, Catia, Ciri, César, David, Eyad, Galo, García, Huici, Ion, Iñaki, Iñigo, Javier, Kaperu, Killian, Leyre, María, Michael, Musku, Olaia, Rakel, Raúl, Rodry, Sara, Turri, Unai, Vicente, Villar, Whils y Xavi (espero no dejarme a nadie).

Gracias por escucharme, tratar de entenderme (incluso cuando hablo en chino) y animarme a seguir.

RESUMEN

Implementación de un minumum viable product para un grupo del Laboratorio Universitario de Creación de Empresas. Este mpv consistirá en una herramienta para facilitar la formación de nuevos equipos de trabajo enfocada desde el factor humano, permitiendo a los candidatos preseleccionar entre sus contactos de Linkedin con quien desearía trabajar. Además, ofrecerá a las empresas la posibilidad de publicar una oferta de trabajo mostrando el equipo para el cual se ofrece la vacante de manera que ambas parte (candidatos y empresas) puedan ver la compatibilidad de los equipos.

La aplicación será de tipo web con el fin de poder llegar a un mayor numero de usuarios y se desarrollará con HTML, CSS y Javascript utilizando varias librerías conocidas para facilitar el mismo como son jQuery, Backbone y Bootstrap. Además utilizaremos Parse, un servidor que permite almacenar gratuitamente datos en la nube y que también nos provee de una librería para facilitarnos su uso.

INDICE

AGRADECIMIENTOS **3**

RESUMEN **4**

CAPÍTULO 1. INTRODUCCIÓN **7**

1.1 Visión general 7

1.2 Objetivos 7

CAPÍTULO 2. CONTEXTO TECNOLÓGICO **11**

2.1 Aplicaciones web 11

2.2 JavaScript 12

2.3 API de LinkedIn 14

2.4 Parse 14

2.5 Backbone 15

2.5.1 Clases de Backbone *16*

2.6 Bootstrap 18

CAPÍTULO 3. TEAMSURFING **19**

3.1 Metodología 19

3.2 Diseño 20

3.3 Implementación 20

3.3.1 LinkedIn *20*

3.3.2 Parse *23*

3.3.3 Backbone *27*

3.4 Resultados y líneas futuras 33

3.4.1 Modificaciones de la aplicación *33*

3.4.2 Cambios en la metodología *34*

3.4.3 Líneas futuras *34*

3.5 Conclusiones 35

CAPÍTULO 4. BIBLIOGRAFÍA **36**

CAPÍTULO 5. ANEXOS **37**

5.1 Aplicación en funcionamiento 37

5.2 Ficheros JavaScript 41

5.2.1 Parte usuario normal *41*

5.2.2 Parte empresas *49*

CAPÍTULO 1. INTRODUCCIÓN

1.1 Visión general

Actualmente hay una gran variedad de buscadores de empleo en internet en los que las empresas evalúan a sus candidatos en función de sus currículos, en los cuales valoran sus estudios, experiencia laboral e idiomas, entre otras cosas. Existen también otros buscadores que además te permiten completar tu currículo añadiendo actividades extracurriculares como pueden ser danza o teatro para diferenciarse de los anteriores y destacar.

Este tipo de buscadores se centran más en lo que buscan las empresas, tratan de encajar el perfil de una persona que cumpla unos requisitos para suplir un puesto de trabajo sin tener en cuenta el ambiente de trabajo que se formará ni como puede afectar este a nuevo grupo de trabajo que se formara, esto puede desembocar en problemas dentro este grupo lo cual puede dar lugar a una menor productividad.

Esta aplicación surge como respuesta a este problema ya que planteamos a los usuarios la opción de escoger con quién querrían trabajar, dándoles la oportunidad de valorar a sus conocidos e informándoles de con quien compartirán su futuro puesto de trabajo en el caso de ser seleccionados.

La decisión de realizar este proyecto como una aplicación web fue tomada con el fin de hacerlo llegar a un público más amplio, haciendo que esté disponible para cualquier dispositivo que pueda acceder a una web y tenga habilitado JavaScript.

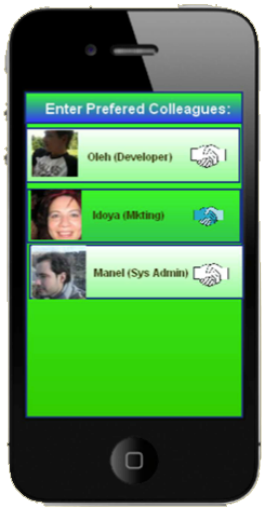
1.2 Objetivos

Este trabajo fin de grado se centra en el desarrollo de un prototipo inicial de aplicación que ofrezca la posibilidad a los usuarios de buscar trabajo y elegir con quien trabajar entre sus contactos de LinkedIn pudiendo resaltar tanto sus cualidades buenas como las cualidades que debería mejorar. Por otra parte permitirá a las empresas publicar ofertas de trabajo y

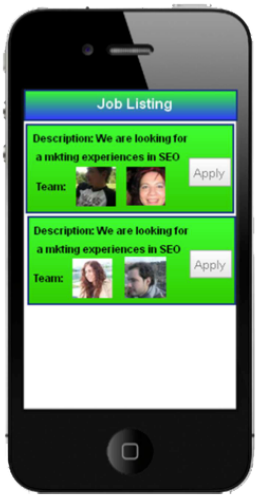
consultar información acerca de lo que comentan los usuarios sobre sus contactos de LinkedIn. Esta aplicación se implementará siguiendo el patrón modelo vista controlador utilizando para ello frameworks como Backbone.js y Bootstrap.js.

Cabe destacar que debido a la naturaleza de este proyecto, los objetivos propuestos para el mismo nacen de una serie de decisiones acerca de la funcionalidad y diseño del mismo. Las cuales han sido tomadas por los componentes del grupo de emprendedores del Laboratorio Universitario de Creación de Empresas. Y, si bien estas decisiones han ido cambiando a lo largo de la implementación de este prototipo, se ha mantenido la idea final del producto.

Partimos de una serie de bocetos que pretender funcionar como guía del resultado final que esperamos conseguir, los cuales se presentan a continuación:



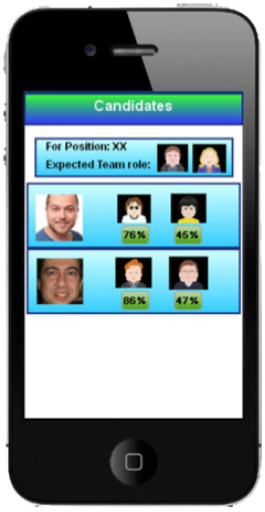
En este boceto vemos una de las pantallas de la parte de la aplicación que interactúa con el usuario en ella éste dispone de la posibilidad de añadir o borrar gente de "su equipo".



En este boceto vemos otra de las pantallas de la parte de la aplicación que interactúa con el usuario en ella pude ver las ofertas publicadas por las empresas donde, a su vez, se proporciona una pequeña descripción y la foto de quienes serán sus futuros compañeros de trabajo.

En este boceto vemos una de las pantallas de la parte de la aplicación con la que interactúan el usuario. En ella el usuario debe contestar a una serie de preguntas planteadas por la empresa para medir su compatibilidad con el puesto de trabajo que se ha ofertado.

En este boceto vemos una de las pantallas de la parte de la aplicación con la que interactúan las empresas en ella la empresa puede publicar una oferta de trabajo especificando que tipo de candidato busca para cubrir el puesto vacante.



En este boceto vemos una de las pantallas de la parte de la aplicación con la que interactúan las empresas en ella la empresa ve los candidatos que se han apuntado a una oferta de trabajo y su compatibilidad con el perfil que se buscaba en para ocupar el puesto vacante.

En este boceto vemos una de las pantallas de la parte de aplicación la con la que interactúan las empresas en ella una empresa puede consultar información acerca de los usuarios como las valoraciones que ha recibido o su equipo favorito. Como podemos apreciar, esta parte de la web ha sido pensada para ser visualizada desde la pantalla de un ordenador, por lo que no es indispensable realizar el diseño de esta vista de manera que se adapte tanto a tablets como a teléfonos móviles.

CAPÍTULO 2. CONTEXTO TECNOLÓGICO

En este capítulo hablaremos de las diferentes tecnologías que hemos utilizado para desarrollar nuestra aplicación, así como de las razones por las que hemos decido utilizar estas tecnologías y no otras.

2.1 Aplicaciones web

La web, tal y como la conocemos hoy en día, permite un flujo de comunicación global. Personas separadas físicamente en tiempo y espacio pueden usar la web para intercambiar pensamientos o ideas. La información web puede ser buscada de manera más fácil y eficiente que en cualquier otro medio físico gracias a su carácter virtual, y mucho más rápido de lo que una persona podría encontrar por sí misma a través de cualquier otro medio de comunicación.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales.

Una ventaja significativa es que las aplicaciones web deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, GNU/Linux, Mac OS, etc., una misma aplicación web se ejecuta igual en todas partes.

Otras ventajas destacables de las aplicaciones web es que su disponibilidad suele ser alta, son portables, consumen pocos recursos, no ocupan espacio en nuestro disco duro y siempre están actualizadas.

2.2 JavaScript

A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web comenzaban a incluir formularios complejos.

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

Brendan Eich, un programador que trabajaba en Netscape, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje LiveScript.

Posteriormente, Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento Netscape decidió cambiar el nombre por el de JavaScript. La razón del cambio de nombre fue exclusivamente por marketing, ya que Java era la palabra de moda en el mundo informático y de Internet de la época.

La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.

Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa". El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript.

Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 a través de su comisión IEC, dando lugar al estándar ISO/IEC-16262. [1]

2.3 API de LinkedIn

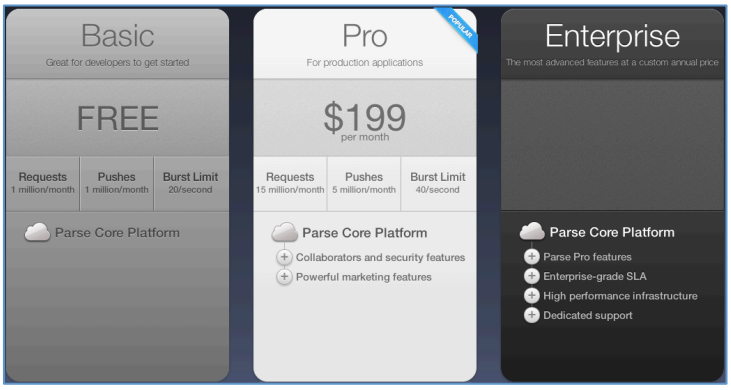
Para la parte del proyecto en la que tenemos que interactuar con LinkedIn hemos decido utilizar el API que nos ofrece la página de desarrolladores de esta red, ya que nos provee de métodos con los cuales hacer consultas sobre información de los diferentes usuarios. Además nos permite autenticar a los usuarios de nuestra aplicación mediante su cuenta en esta red de manera simple y fiable.

2.4 Parse

Parse es un producto de Facebook que nos provee de la implementación de un modelo Backend as a Service, el cual proporciona a los desarrolladores una forma de vincular aplicaciones al almacenamiento en la nube.

Nos permite olvidarnos del mantenimiento del servidor y nos proporciona una manera sencilla de utilizar diferentes características como pueden ser notificaciones push y conexión de usuarios mediante inicios de sesión tradicionales o mediante redes sociales.

Además, cabe destacar que este servicio es gratuito hasta superar un límite de peticiones al servidor, tras superarlo permite pagar para aumentar ese número de peticiones, así como añadir nuevos servicios.

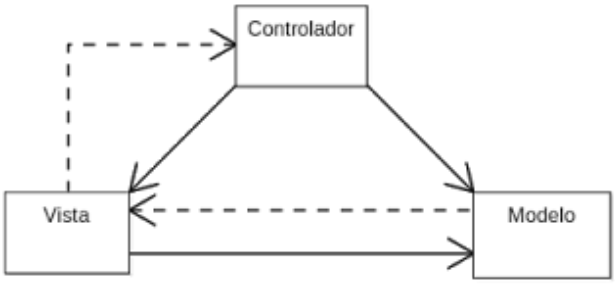


Todo esto, junto con el hecho de que ya habíamos trabajado previamente con este servicio, ha hecho que decidamos utilizarlo para el almacenamiento de los datos de nuestra aplicación frente a otras posibilidades.

2.5 Backbone

Backbone es un framework que facilita la creación de single page application, aunque también sirve para crear sitios web de más de una página, donde se encuentran interfaces de usuario avanzadas. Este framework facilita la ordenación del código y la sincronización de las vistas con los datos de la aplicación. Todo esto nos permite realizar una programación mas modularizada y con mayor separación entre las partes de la aplicación.

Backbone facilita estructurar las aplicaciones web basándose en el paradigma de programación MVC, modelo, vista, controlador. Para aplicaciones enriquecidas del lado del cliente, un enfoque estructurado como el que ofrece este framework, es muy útil y beneficioso.



Backbone además proporciona una forma para trabajar con servicios REST tanto para recuperar los datos de los modelos como para actualizarlos, guardarlos y eliminarlos en el servidor.

2.5.1 Clases de Backbone

Backbone pone a nuestra disposición una serie de clases para poder organizar mejor el código de la aplicación:

- Model: contiene la información de un objeto de datos, así como gran parte de la lógica que los rodea (conversiones, validaciones, propiedades calculadas, y control de acceso).

```
var Sidebar = Backbone.Model.extend({
  promptColor: function() {
    var cssColor = prompt("Please enter a CSS color:");
    this.set({color: cssColor});
  }
});
```

Figura 7

- Collection: conjuntos ordenados de modelos.

```
var Library = Backbone.Collection.extend({
  model: Book
});
```

- Router: se encarga de las transiciones entre las vistas, proporciona métodos para redirigir páginas del lado cliente de la aplicación y unirlos a acciones y eventos.

```
var Workspace = Backbone.Router.extend({

  routes: {
    "help":                 "help",    // #help
    "search/:query":        "search",  // #search/kiwis
    "search/:query/p:page": "search"   // #search/kiwis/p7
  },

  help: function() {
    ...
  },

  search: function(query, page) {
    ...
  }

});
```

- View: se encarga de gestionar los objetos del DOM y son la parte visible de la aplicación.

```
var DocumentRow = Backbone.View.extend({

  tagName: "li",

  className: "document-row",

  events: {
    "click .icon":          "open",
    "click .button.edit":   "openEditDialog",
    "click .button.delete": "destroy"
  },

  initialize: function() {
    this.listenTo(this.model, "change", this.render);
  },

  render: function() {
    ...
  }

});
```

- History: guarda el historial de navegación del usuario dentro de la single page app.

```
Backbone.history.start({pushState: true});
```

Como se puede observar, este framework nos ofrece todo lo necesario para poder realizar nuestra aplicación. Además, tiene la ventaja de que es compatible con Parse, el servicio de almacenamiento en la nube descrito anteriormente, por lo que lo hemos elegido como base para implementar esta aplicación.

2.6 Bootstrap

Bootstrap es un framework utilizado por Twitter (entre otros) que permite crear interfaces web con CSS y JavaScript que adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, es decir, automáticamente se adapta al tamaño de un ordenador o de una Tablet sin que el usuario tenga que hacer nada, esto se denomina diseño adaptable o Responsive Design.

Los diseños creados con Bootstrap son simples, limpios e intuitivos, esto les da agilidad a la hora de cargar y al adaptarse a otros dispositivos. Este framework trae varios elementos con estilos predefinidos fáciles de configurar: Botones, Menus desplegables, Formularios incluyendo todos sus elementos e integración jQuery para ofrecer ventanas y tooltips dinámicos.

Decidimos utilizar este framework, ya que nos facilita la tarea de construir una aplicación adaptable a dispositivos con diferentes tamaños de pantalla. En la web hemos encontrado diferentes editores que ayudan en la creación de diferentes plantillas utilizando este framework como Bootstrap Form Builder (http://minikomi.github.io/Bootstrap-Form-Builder/).

CAPÍTULO 3. TEAMSURFING

En este capítulo hablaremos sobre cómo se han utilizado las tecnologías que hemos elegido para la implementación de este proyecto, descritas en el capitulo anterior, además de la metodología que se ha llevado a cabo para su desarrollo.

3.1 Metodología

Para el desarrollo de este proyecto hemos seguido una metodología Scrum con ligeras modificaciones, la cual se basa en el principio ágil de desarrollo iterativo e incremental.

Al periodo de trabajo para desarrollar un incremento de producto se le denomina "sprint", y se recomiendan duraciones entre una y cuatro semanas (en este caso la duración de los sprint varían entre una y dos semanas).

Establece una reunión al inicio de cada sprint y para determinar el trabajo que se va a realizar, otra al final para evaluar el resultado, y revisiones diarias que realiza el equipo para su autogestión.

Sprint 0 BackLog

- Login Linkedin OK ,Como hacer login simultanea de linkein y parse: http://blog.parse.com/2014/01/14/adding-third-party-authentication-to-your-web-app/
- Traer Contactos Linkedin OK
- Traer email Linkedin ->crear usuario Parse si no existe y guardar su email, # contactos,…
- (si primer login: pedir localización, años experiencia,….)
- Mostrar Contactos Linkedin OK
- Añadir Contacto en Tabla Parse "Favoritos"
- Mostrar Tabla Parse "Favoritos"
- (podría invitar anónimamente a sus contacto-> pedir email, crear usuario parse)

Sprint 1.0 BackLog

- Login Empresas: usar linkedin de jefe RRHH;
- Empresa busca candidatos:
  - (equipo: linkedin)
  - (pedir email equipo e invitar no anónimamente,..)
  - (equipo son nuevos usuarios)-> se calcula localización, años experiencia, ….
- Empresa ve lista de "Candidatos" para busqueda.

(si búsqueda no es satisfactorio, se le propone que publique oferta??)

- Empresa añade oferta en Parse "Ofertas": campos…
- Usuarios ven lista de "Ofertas"

Primeros sprints realizados durante la implementación de este proyecto.

3.2 Diseño

Con el fin de facilitar la organización de la aplicación y tener una estructura más clara de la misma, hemos decidido separar las dos partes de la aplicación en dos sub-proyectos diferentes, de manera que cada uno es una implementación de MVC independiente del otro y solo comparten recursos como librerías e imágenes. Explicaremos más detalladamente este diseño en el apartado de implementación de Backbone a continuación.

3.3 Implementación

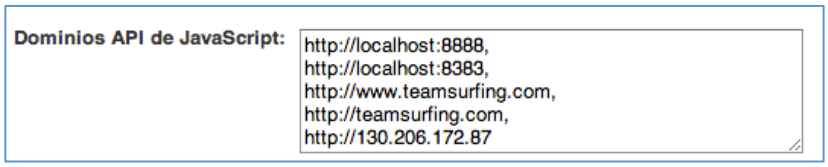
Como ya se ha comentado en otros apartados el lenguaje de programación utilizado para el desarrollo de este proyecto es JavaScript con la ayuda de diferentes frameworks como Bootstrap y Backbone, y utilizando las APIs de Linkedin y Parse para comunicarnos con estos servicios.

Este apartado pretende explicar cómo se han utilizado las diferentes herramientas para poder implementar esta aplicación además de mostrar por separado las diferentes funciones que cumplen cada uno de los componentes de este proyecto.

3.3.1 LinkedIn

Uno de los requisitos para que los usuarios (tanto de las personas que buscan trabajo como las empresas) de la aplicación puedan usarla es que previamente tengan creada una cuenta en LinkedIn, ya que, utilizamos la información que el usuario proporciona en esta red (contactos, email, foto de perfil) para el funcionamiento de esta aplicación, además usamos la propia cuenta de LinkedIn para acceder a la aplicación.

Para poder acceder a la información que el usuario tiene en su cuenta y permitirle iniciar sesión con ella hemos tenido que registrar previamente nuestra aplicación en la página de desarrolladores de LinkedIn indicando los dominios en los que será utilizada.



En la misma pantalla encontramos también la clave de la API de nuestra aplicación, una clave alfanumérica que nos permite comenzar a utilizar todas las funciones de la API de LinkedIn. Esta clave la introducimos, como se muestra a continuación, en la cabecera del fichero HTML desde el cual vamos a utilizar las funciones del API.

```
<script type="text/javascript" src="https://platform.linkedin.com/in.js">
    api_key: CLAVE_API_LINKEDIN
    authorize: true
</script>
```

Una vez tenemos acceso a LinkedIn a través de la clave obtenida, para poder comenzar a utilizarla solo debemos realizar las invocaciones a las funciones que nos ofrece el API. Para permitir a los usuarios acceder a la página con su cuenta de LinkedIn solo es necesario realizar una llamada a la función authorize.

```
function logINUsuario() {
    IN.User.authorize(function(){
        window.location ="usuarios";
    });
}
```

La llamada que vemos en el código anterior se muestra al usuario la siguiente pantalla de LinkedIn para que introduzca sus credenciales.

Para obtener información acerca del usuario y sus contactos realizamos diferentes llamadas a la API indicando la información que esperamos obtener.

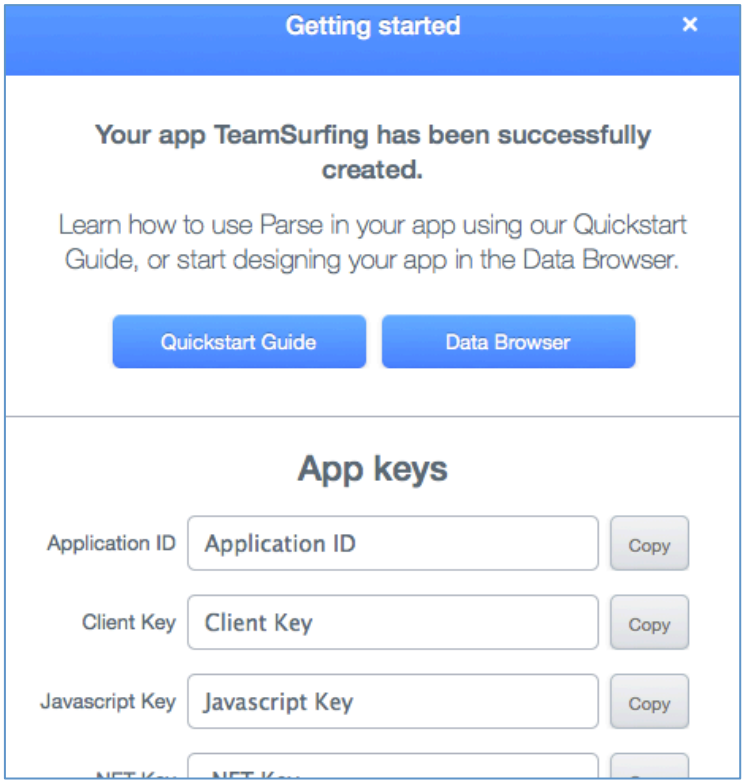
```
IN.API.Profile("me")
        .fields(["id", "email-address","threeCurrentPositions"])
        .result(function(result, metadata){...53 lines });


IN.API.Connections("me")
    .fields("id", "firstName", "lastName", "pictureUrl")
    .result(function(result, metadata) {...27 lines });
```

Estas llamadas nos devuelven información sobre el usuario, que previamente se ha autenticado con LinkedIn, como su identificador único de usuario de la red, así como su email de contacto e información de donde trabaja (en la primera llamada al API del código anterior) y el identificador único de usuario, nombre, apellido e imagen de perfil (en la segunda llamada al API). Esta información la utilizaremos para autenticar posteriormente al usuario en Parse, y mostrarle sus contactos a la hora de elegir su "equipo ideal" de trabajo.

3.3.2 Parse

Toda la información que utilizamos para el correcto funcionamiento de la aplicación la almacenamos en Parse, puesto que nos permite usarlo sin ningún coste hasta superar un límite de peticiones a este servicio, como ya explicamos en un apartado anterior.

Para poder comenzar a utilizar este servicio primero debemos crear una cuenta en su página web (https://parse.com/) y, además, crear una aplicación como vemos a continuación.



Tras crear la aplicación, Parse nos muestra una serie de claves para poder acceder al su API mediante diferentes lenguajes de programación (en este caso solo necesitamos la de JavaScript), así como el identificador de la aplicación. Estas claves las utilizaremos en la parte de nuestro proyecto que se comunica con Parse para obtener y guardar los datos de los usuarios de nuestra aplicación como vemos a continuación.

```
Parse.initialize("Application ID", "Javascript Key");
```

Una vez tenemos acceso a Parse a través de la clave y el id de aplicación obtenidos, podemos comenzar a utilizarla para realizar las invocaciones a las funciones que nos ofrece el API. Para comenzar autenticamos al usuario de la aplicación como usuario de Parse (en el caso de no existir se le daría de alta) de la siguiente manera.

```
user = new Parse.User();
user.set("username", USUARIO);
user.set("password", CONTRASEÑA);
user.set("email", EMAIL);


user.signUp(null,{
    success: function(user){
        // Nada
    },
    error: function(user, error){
        if (error.code === Parse.Error.USERNAME_TAKEN){
            Parse.User.logIn( DATOS_DEL_USUARIO , {
                success: function(user) {...20 lines },
                error: function(user, error) {
                    window.location.href = '../';

                }
            });
        }
    }
});
```

Una vez el usuario esta autenticado en Parse podemos empezar a trabajar con los datos de este usuario. Cuando éste cree su "equipo ideal" o se inscriba en una oferta de trabajo la información quedará almacenada en Parse con el identificador de este usuario en las diferentes tablas creadas para ese fin, las cuales mostramos a continuación.

objectId String	password String	email String	company String	empresa Boolean
laTOteibUK	(hidden)	oscar.ardaiz@unavar...	Public University of N...	true
3KM9MGMRdl	(hidden)	aitor.mendivil@icloud...		false
MEINvS9Ik6	(hidden)	inigoezcurdia@gmail...	(undefined)	false
umVEAG9xxq	(hidden)	oliversanz1991@gm...	Joe Chest	true

En esta tabla almacenamos la información referente al usuario como su identificador, su email, su empresa (si actualmente trabaja en alguna), y si tiene acceso a la parte de empresas de la aplicación.

name String	description String	team Array
Graduado Artes Visuales o Inte...	Trabajaras con especialistas en dise...	["es.linkedin.com/in/perevilaro"]
Graduado ADE para marketing ...	Trabajaras con especialistas en dise...	["https://www.linkedin.com/in/oliversan...
Ingeniero Software de Aplicaci...	Trabajarás con especialistas en dise...	[""]
Graduado Económicas para de...	Trabajaras con especialistas en dise...	[""]

En esta tabla almacenamos la información referente a las ofertas creadas por usuarios con perfil de empresa en la que guardamos información de la oferta como su nombre, una breve descripción y los perfiles públicos de LinkedIn de las personas que ya forman parte del equipo para el cual hay una vacante.

INId String	firstName String	lastName String	createdBy String
Jv2gvbB345	Iñigo	Ezcurdia Aguirre	3KM9MGMRdl
XzsmkWlUN8	Oliver	Sanz Catalán	3KM9MGMRdl
F5BlJzYcKY	David	Gonzalez	3KM9MGMRdl
k0CAj_FpDM	Xabier	Napal Roncal	umVEAG9xxq
yRz9K7NLml	María	Barea	umVEAG9xxq
skCtBR4RXv	David	García	umVEAG9xxq
X6lWvdnvED	Michael	Coloma Calva	umVEAG9xxq
gBZ1xXxLNr	Jon	Legarrea	3KM9MGMRdl
38MMXXxrSb	Jose Enrique	Armendariz-Inigo	3KM9MGMRdl
PPlh-MDOzC	Iker	Belloso Sancet	3KM9MGMRdl

En esta tabla almacenamos los integrantes de los "equipos ideales" de nuestros usuarios, de los cuales guardamos su nombre completo, su identificador de LinkedIn, quien los ha añadido a su equipo ideal y una pequeña valoración tanto

positiva como negativa acerca los mismos. Estas valoraciones solo podrán ser leídas por usuarios con el perfil de empresa, aunque no podrán saber quién las ha escrito.

idUsuario String	idOferta String
umVEAG9xxq	EGOD2RwtZa
umVEAG9xxq	EGOD2RwtZa
umVEAG9xxq	c62NlKkcu0
umVEAG9xxq	EriNOYb2uS
umVEAG9xxq	c62NlKkcu0
umVEAG9xxq	yCT0CPeyyc

En esta tabla almacenamos qué usuario se ha inscrito a qué oferta.

objectId String	comentario String	email String
40jfhMAMPK	¿Tenéis algún email de conta…	maria_12f@g…
8hMM2E2x5t	¿Cómo entrar en la parte de …	oliversanz199…
EJFxbPriwl	¿Cuando estará la web en fu…	javiersanz17@…

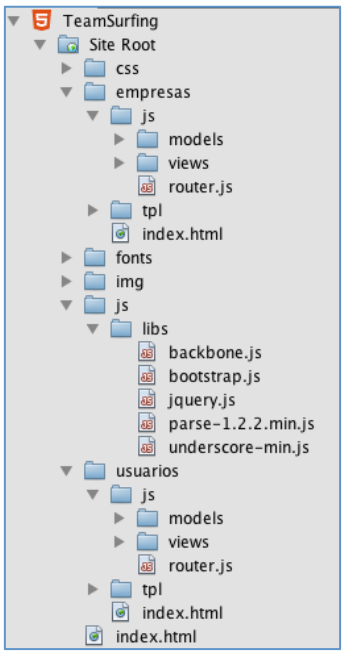
Esta tabla contiene comentarios acerca de la web por parte de usuarios que pueden no estar registrados, ya que damos opción a opinar sobre la web antes de acceder mediante LinkedIn.

Las operaciones de inserción, borrado, actualización y lectura realizadas sobre estas tablas las explicaremos en el apartado de Backbone, ya que todas ellas se realizan desde la parte del modelo de esta aplicación.

Cabe destacar que las tablas que acabamos de mostrar tienen otros campos, los cuales no mostramos aquí, como son el nombre de usuario, la fecha de creación y modificación de los objetos, verificación de email, etc. ya que no tienen importancia para la compresión de este apartado.

3.3.3 Backbone

Para implementar el MVC hemos utilizado Backbone ya que este framework facilita la organización del código separando cada parte del resto. Backbone permite una estructuración libre del proyecto, aunque da algunas directrices sobre cómo estructurar los diferentes componentes del mismo. Para este proyecto hemos decidido seguir la estructura que vemos en la siguiente imagen.



De esta manera, las librerías necesarias para el funcionamiento de toda la aplicación (jQuery.js, Backbone.js, Bootstrap.js, Parse.js y Underscore.js) se encuentran en raíz del proyecto junto con los glyphicons, 180 iconos creados mediante una fuente especial llamada Glyphicon Halflings, las imágenes que utilizaremos en toda la aplicación y las hojas de estilos, comunes también a toda la aplicación.

Vemos también que, como hemos mencionado con anterioridad, las dos partes de la aplicación han sido separadas en dos directorios, los cuales siguen el mismo esquema (índice, plantillas y ficheros JavaScript).

Ahora explicaremos cómo hemos implementado el patrón MVC con la ayuda de Backbone en cada proyecto.

Usuarios

Esta parte de aplicación es la que cumplirá la función de permitir a las personas que buscan trabajo ver las ofertas de las diferentes empresas y valorar a sus contactos de Linkedin tanto positivamente como negativamente.

La función del controlador es llevada a cabo por el router. Este es el encargado de manejar las URLs para dirigir las peticiones o las acciones de los usuarios a los lugares concretos donde está el código que tiene que realizar lo que se solicita. En él, hemos definido los siguientes métodos y atributos:

- Routes: en este atributo definimos las URL que enrutará las diferentes peticiones de los usuarios y a donde las enrutará.
- Initialize: método encargado de indicar cuál será el contenido de la web que modificaremos en base a las diferentes peticiones de los usuarios.
- Home: método encargado de cargar la vista inicial de la página que hará las veces de página principal para esta parte de la aplicación.
- Contactos: método encargado de cargar la vista en la que se encontrarán los contactos de LinkedIn del usuario actual que no formen parte de su "equipo ideal".
- Favoritos: método encargado de cargar la vista en la que se encontrarán los contactos de LinkedIn de usuario actual que hallan sido seleccionados como parte de su "equipo ideal".
- Ofertas: método encargado de cargar la vista en la que se encontrarán las diferentes ofertas de trabajo ofrecidas por las empresas.
- CargaInicial: método encargado de obtener los datos a través del API de LinkedIn, de autenticar el usuario en Parse y obtener los datos de su "equipo ideal". Este método se ejecuta tras autenticar al usuario mediante su cuenta de LinkedIn al acceder a la página.

El modelo de esta parte de la aplicación lo hemos separado en tres archivos diferentes (una por cada modelo de datos que necesitamos), de manera que tenemos uno que extiende el modelo de Backbone encargado de almacenar los

datos acerca de los contactos de LinkedIn de nuestro usuario y dos modelos que extienden del modelo de Parse, encargados de almacenar los datos acerca del "equipo ideal" de nuestro usuario y las ofertas de trabajo. Estos dos últimos modelos son también los encargados de realizar las operaciones de creación, lectura, actualización y borrado sobre los datos almacenados en Parse, es por ello que extienden del modelo de Parse y no del modelo de Backbone. Esto ocurre también con las colecciones que utilizamos. Ambas clases siguen una misma estructura, la cual explicaremos a continuación junto con algunos de sus métodos:

- Defaults: conjunto de atributos y valores que tomará el modelo al inicializarse en el caso de que no se le indiquen valores diferentes para esos atributos.
- IdAttribute: atributo que representará el objeto dentro de una colección. Este atributo se utiliza para forzar a la colección a utilizar como índice un parámetro en concreto, lo cual nos facilita después el trabajo con éstas.
- ClassName: atributo que representa el nombre de la tabla de Parse en la que será almacenado este un objeto creado con este modelo.
- Comparator: método utilizado para ordenar los modelos dentro de una colección en base a un criterio concreto. En nuestro caso, hemos utilizado como criterio la primera letra del nombre de los contactos.
- Fetch: método que se encarga de pedir a Parse el conjunto de modelos almacenados en la tabla correspondiente y los guarda en la colección cuando los recibe.
- CargarEquipo: método que pide al API de LinkedIn la información publica acerca de los integrantes del equipo de una oferta de trabajo. Tras recibirla la guarda en el modelo correspondiente.

La vista de esta parte de la aplicación la hemos separado en cuatro archivos diferentes (uno por cada sección de la página). Todas extienden de la vista de Backbone y comparten una misma estructura que explicamos a continuación:

- Initialize: en este método asignamos a los eventos de la vista el conjunto de actividades que deben realizar según los eventos que tengan lugar mientras esa vista esté activa.
- Render: en este método cargamos los datos del modelo correspondiente asignándoselos a la plantilla de la vista con la que estamos trabajando de manera que nos devuelve la plantilla con los datos ya introducidos.

- AddUser: este método esta asignado a las vistas de contactos de LinkedIn como la que vemos en la imagen anterior. En él preguntamos al usuario información acerca del contacto que acaba de añadir a su "equipo ideal" y, tras recoger esa información la guardamos en Parse. Además este método también se encarga de eliminar la vista que representaba a este usuario.
- RemoveUser: este método esta asignado a las vistas del "equipo ideal" similares a la anterior. En él borramos el contacto seleccionado de nuestro equipo actualizando esta información en Parse.
- Aplicar: este método actualiza la tabla de candidatos añadiendo a este usuario a la lista de personas que se han apuntado a la oferta.

Empresas

Esta parte de aplicación es la que cumplirá la función de permitir a las empresas publicar ofertas de trabajo, consultar las ofertas que habían publicado previamente y ver las recomendaciones que han recibido los diferentes usuarios.

La función del controlador es llevada a cabo por el router, al igual que en la parte del usuario. En él, hemos definido los siguientes métodos:

- Routes: lo mismo que en la parte del usuario.
- Initialize: lo mismo que en la parte del usuario.
- Home: : lo mismo que en la parte del usuario.
- Publicar: método encargado de cargar una vista en la que se mostrara un pequeño formulario en el que introducir datos para crear una oferta de trabajo.
- MisOfertas: método encargado de cargar una vista en la que se mostrará una tabla con las diferentes ofertas de empleo publicadas por la empresa que ha accedido a la aplicación.
- FichaUsuario: método encargado de mostrar una vista con información acerca del usuario con los comentarios positivos y negativos que han hecho sobre él.
- CargaInicial: : lo mismo que en la parte del usuario, salvo que en esta parte de la aplicación no lee de Parse el equipo ideal del usuario. En esta parte coge la información acerca de las recomendaciones de los usuarios para mostrar a las empresas tanto las cualidades como los defectos que puede tener un candidato según sus contactos de LinkedIn.

El modelo de esta parte de la aplicación también lo hemos separado en tres archivos diferentes, de manera que tenemos uno que extiende el modelo de Backbone encargado de almacenar los datos acerca de un usuario concreto y dos modelos que extienden del modelo de Parse, encargados de almacenar los datos acerca de las ofertas de trabajo creadas por la empresa y las valoraciones que han

sido realizadas por los usuarios que no tienen perfil de empresa acerca de sus contactos de LinkedIn. Al igual que en la parte del usuario, estos dos últimos modelos son los encargados de realizar las operaciones de creación, lectura, actualización y borrado sobre los datos almacenados en Parse. Esto ocurre, también, con las colecciones que utilizamos. Todos los modelos siguen la misma estructura que los explicados anteriormente, a continuación explicaremos sus métodos y atributos:

- Defaults: lo mismo que en la parte del usuario.
- Initialize: lo mismo que en la parte del usuario, salvo que en esta parte de la aplicación lo utilizamos para pedir los datos al API de LinkedIn del modelo que no se comunica con Parse.
- ClassName: lo mismo que en la parte del usuario.
- Fetch: lo mismo que en la parte del usuario.
- FindByName: este método recibe una cadena de caracteres como parámetro y busca, en la colección de valoraciones que tenemos, aquellas valoraciones que hagan referencia a un usuario cuyo nombre o apellidos contengan esa secuencia de caracteres.

La vista de esta parte de la aplicación la hemos separado en seis archivos diferentes, uno por cada sección de la página, uno para el menú superior y una para el buscador de usuarios. Todas extienden de la vista de Backbone y, al igual que en la otra parte de la aplicación, comparten una misma estructura que explicamos a continuación:

- Initialize: lo mismo que en la parte del usuario.
- Render: lo mismo que en la parte del usuario.
- CrearOferta: este método recoge la información del formulario de publicación de ofertas en su vista y crea una nueva entrada en la tabla de ofertas de trabajo.
- Search: este método recoge lo que el usuario introduce en el cuadro de búsqueda y se lo pasa como parámetro a la función FindByName del modelo.

Todos los ficheros JavaScript (vistas, controladores y routers) de los que hemos hablado en esta sección pueden ser consultados en el segundo anexo de este trabajo.

3.4 Resultados y líneas futuras

En este apartado hablaremos sobre las decisiones que hemos tomado a la hora de abordar ciertas partes del proyecto, así como de las discusiones acerca de la metodología seguida durante la implementación del mismo. En este apartado realizaremos también una comparativa entre el prototipo actual de la aplicación y los diseños iniciales que se propusieron al inicio del mismo. Al final de este apartado hablaremos sobre las ideas hacia las que se puede enfocar este proyecto en un futuro.

3.4.1 Modificaciones de la aplicación

A lo largo del proyecto ha habido varios cambios respecto de las ideas iniciales, muchos de ellos debido a cambios impuestos por el grupo de emprendedores del Laboratorio Universitario de Creación de Empresas, y otros en base nuestro criterio.

En un inicio se pensó la aplicación como un modelo simple de trabajo en equipo "yo quiero trabajar con" en el cual los usuarios simplemente creaban su equipo únicamente eligiendo sus componentes. Tras varias reuniones con empresas y asistencias a charlas de grupos de emprendedores se tomó la decisión de que un cambio importante sería, con el fin de evitar un efecto arrastre (los usuarios buscan ser incluidos en los equipos de desconocidos incluyendo a estos desconocidos en sus propios equipos) pedir a los usuarios dos pequeñas valoraciones, una positiva otra negativa sobre las personas que quisieran añadir a sus equipos.

Otro de las decisiones que se tomaron a lo largo de la implementación de este prototipo fue eliminar, de este prototipo inicial, el breve cuestionario creado por la empresa para medir la compatibilidad de los candidatos con la vacante de la oferta de trabajo a la que se inscriben. Esta decisión fue tomada debido a que, tras una charla en la Asociación Navarra de Empresas Laborales (anel), descubrimos que se

necesitaba mucha más información de la que disponíamos para realizar un test que cumpliera los objetivos para los que se había propuesto su creación, por lo que decidimos que sería implementado en futuras versiones de la aplicación.

Decidimos también no utilizar un índice de popularidad como factor a la hora de valorar un candidato con el fin de evitar el efecto arrastre del que hemos hablado anteriormente. Aunque, también es cierto que, cuantas mas personas añadan a una persona a su equipo, más probable es que sea valorado de manera más positiva por parte de las empresas ya que dispondrán de más información acerca de él.

3.4.2 Cambios en la metodología

Aunque la mayor parte del desarrollo de este proyecto ha seguido la metodología Scrum, esta se ha seguido en una versión más reducida debido a factores como, por ejemplo, que el equipo de desarrolladores constaba de un único integrante el cuál se encargaba también de realizar otras tareas como el seguimiento o las estimaciones, por lo que se opto por omitir ciertas partes de esta metodología como son las gráficas Burn-down o gráfico de avance, las gráficas Burn-up o gráficas de producto, las reuniones de equipo y la estimación de póker (entre otras cosas).

3.4.3 Líneas futuras

Aunque esta versión de la aplicación cumple con bastantes de los objetivos que marcamos al inicio del proyecto no deja de ser un prototipo sin acabar, de cara a futuras versiones del producto, una serie de mejoras a implementar serían:

- Permitir a las empresas publicar test de compatibilidad en cada oferta de trabajo de manera que pudieran obtener un indicador más a la hora de valorar a los candidatos inscritos para esa vacante.
- Mejorar el aspecto visual de la aplicación, hacerla mas atractiva y sencilla de manejar a los usuarios.

- Incluir información como contactos, trabajos, estudios y valoraciones desde otras redes de trabajo como, por ejemplo, infojobs.

3.5 Conclusiones

Tras finalizar este proyecto (o esta iteración de la aplicación) consideramos que debemos realizar una pequeña retrospectiva de lo que hemos aprendido con este proyecto.

Hemos aprendido a utilizar el API de LinkedIn para permitir a nuestros usuarios iniciar sesión en nuestra aplicación con su cuenta y a obtener información relevante acerca de los mismos.

Hemos aprendido a utilizar el API de Parse para realizar una gestión de usuarios e información en base al su sistema de tablas y permisos.

Hemos aprendido a utilizar frameworks que nos facilitan la creación de simple page applications (Backbone) y que nos permiten realizar un diseño adaptable de una manera más cómoda y sencilla (Bootstrap).

Hemos aprendido varias cosas acerca del mundo de la contratación de personal a nivel tanto de grande como de pequeñas y medianas empresas. Un nuevo punto de vista muy interesante de cara a la naturaleza de este proyecto.

Todo esto nos ha permitido crear este prototipo y, si bien existen ciertos puntos en los que se podría mejorar, el balance final del mismo es muy positivo ya que hemos conseguido implementar una aplicación que nos ofrecerá un nuevo punto de vista a la hora de contratar personal para nuestros grupos de trabajo o bien nos mostrará una nueva manera de valorar las ofertas de trabajo en las empresas.

CAPÍTULO 4. BIBLIOGRAFÍA

1. Breve historia (Introducción a JavaScript), LibrosWeb: http://librosweb.es/javascript/capitulo_1/breve_historia.html
2. Juan Palacio, Claudia Ruata (2011): *Scrum Manager Gestión de Proyectos*
3. Página oficial del framework Backbone: http://backbonejs.org/
4. Ejemplo de aplicación creada con Backbone: http://dsthode.info/2012/09/crear-una-aplicacion-sencilla-con-backbone-js/
5. Otro ejemplo de aplicación creada con Backbone: http://dsthode.info/2012/09/crear-una-aplicacion-sencilla-con-backbone-js/http://elblogdepicodev.blogspot.com.es/2013/04/introduccion-y-ejemplo-de-backbonejs.html
6. Definición del framework Bootstrap: http://openwebcms.es/2013/que-es-bootstrap/
7. Página oficial del framework Bootstrap: http://getbootstrap.com/
8. Página oficial del framework Bootstrap en castellano: http://www.oneskyapp.com/es/docs/bootstrap/
9. Aplicación final: http://www.teamsurfing.com/
10. Página de desarrolladores de LinkedIn: https://developer.linkedin.com/
11. Página oficial de LinkedIn: https://www.linkedin.com/
12. Página de desarrolladores de Facebook, apartado de Parse: https://developers.facebook.com/products/parse/

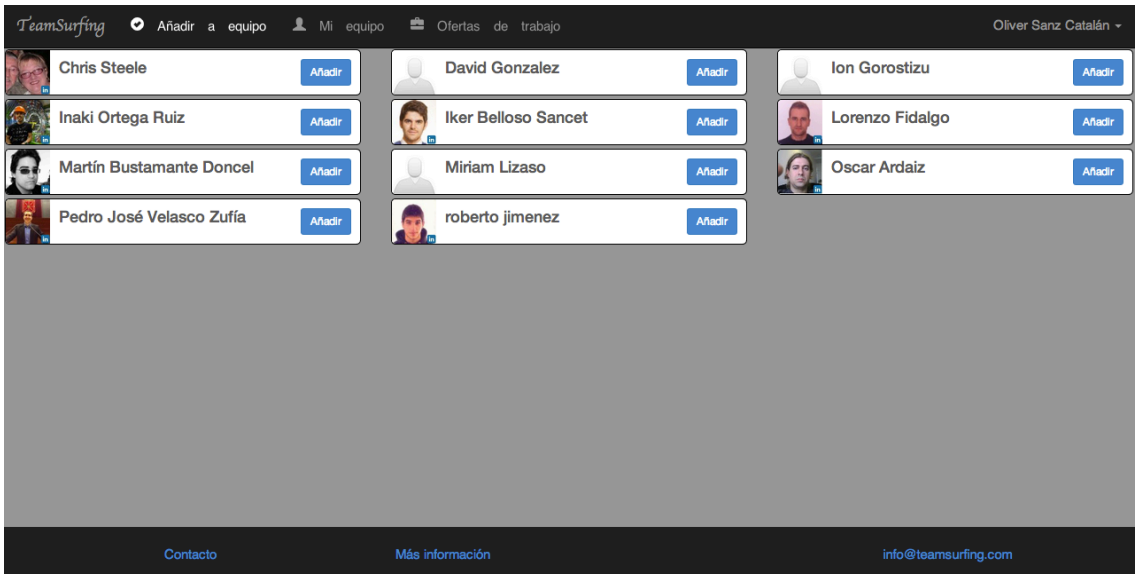
CAPÍTULO 5. ANEXOS

5.1 Aplicación en funcionamiento

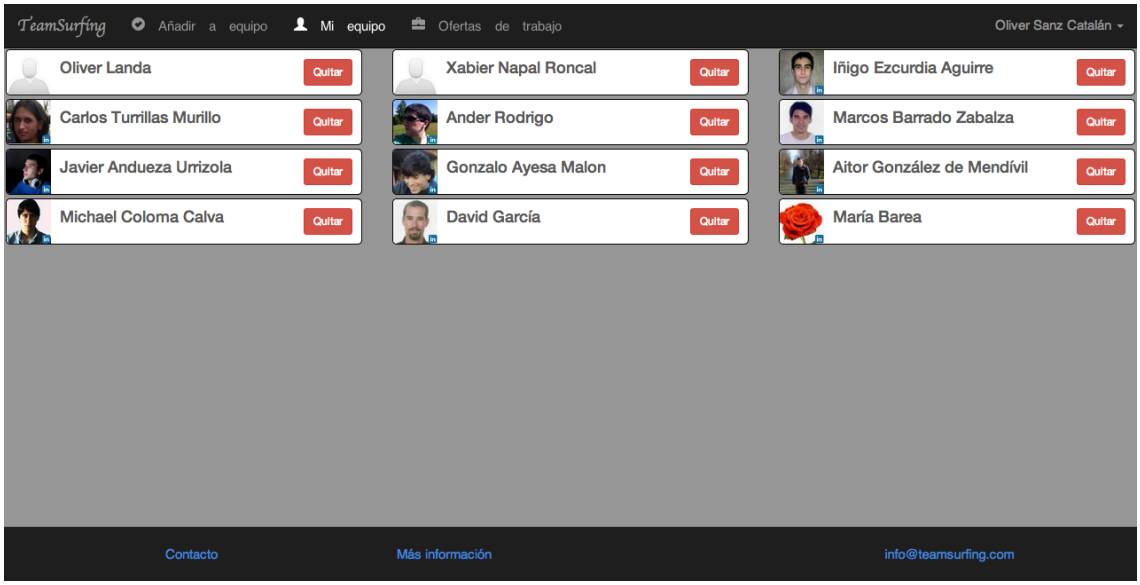
A continuación mostraremos unas imágenes del prototipo en funcionamiento:

- Página principal: esta es la página que veremos al acceder a la aplicación en http://www.teamsurfing.com

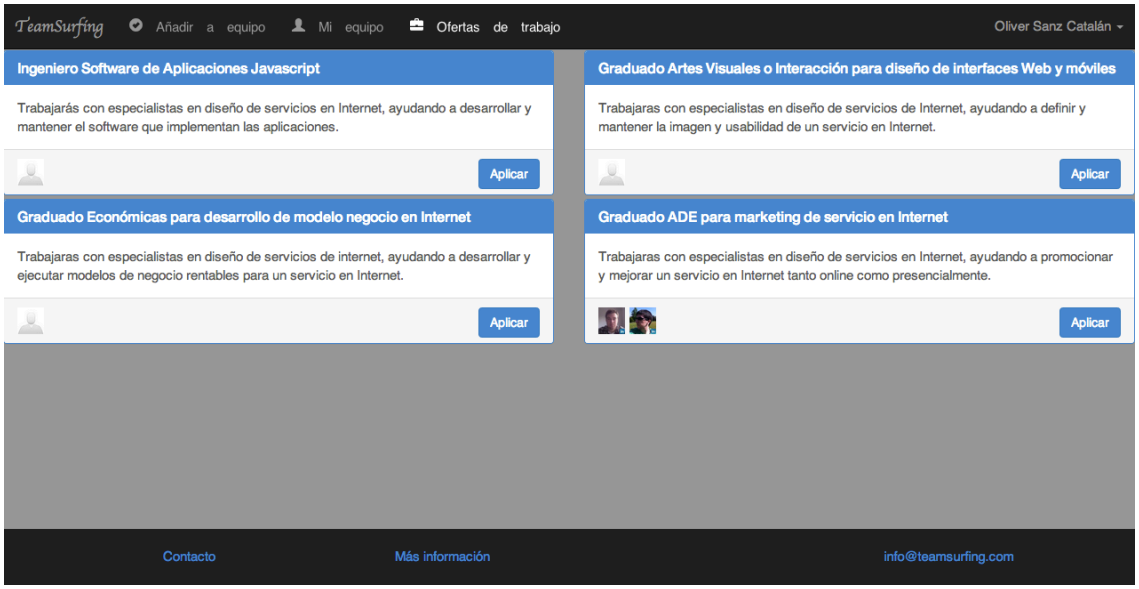
- Añadir a equipo (usuario normal): tras acceder como usuario normal de la aplicación podremos acceder a esta pantalla, la cual nos muestra nuestros contactos de LinkedIn



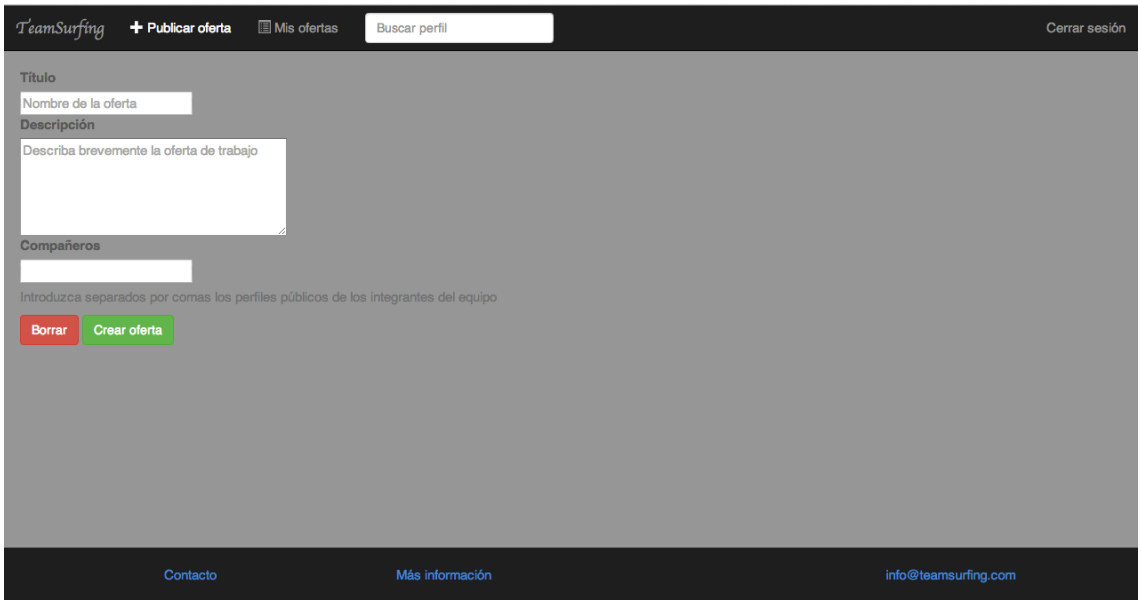
- Mi equipo (usuario normal): esta pantalla es similar a la anterior. En ella podemos consultar nuestro "equipo ideal" y eliminar gente del mismo



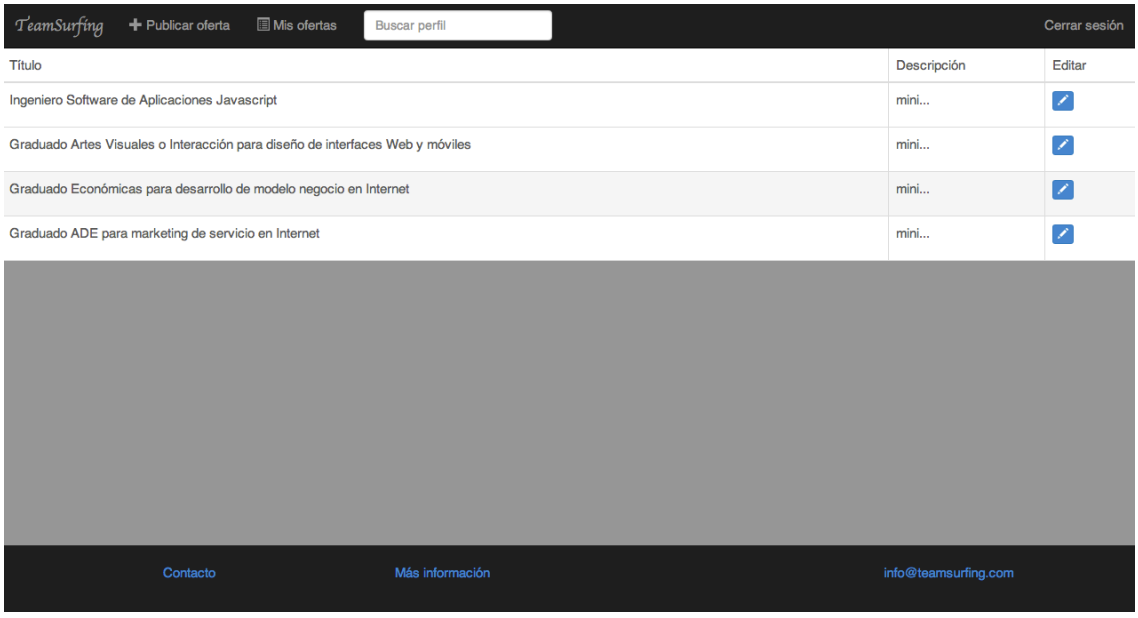
- Ofertas de trabajo(usuario normal): en esta pantalla veremos las diferentes ofertas publicadas por las empresas y podremos inscribirnos en ellas



- Publicar oferta (parte empresas): en esta pantalla vemos el formulario para publicar ofertas de trabajo

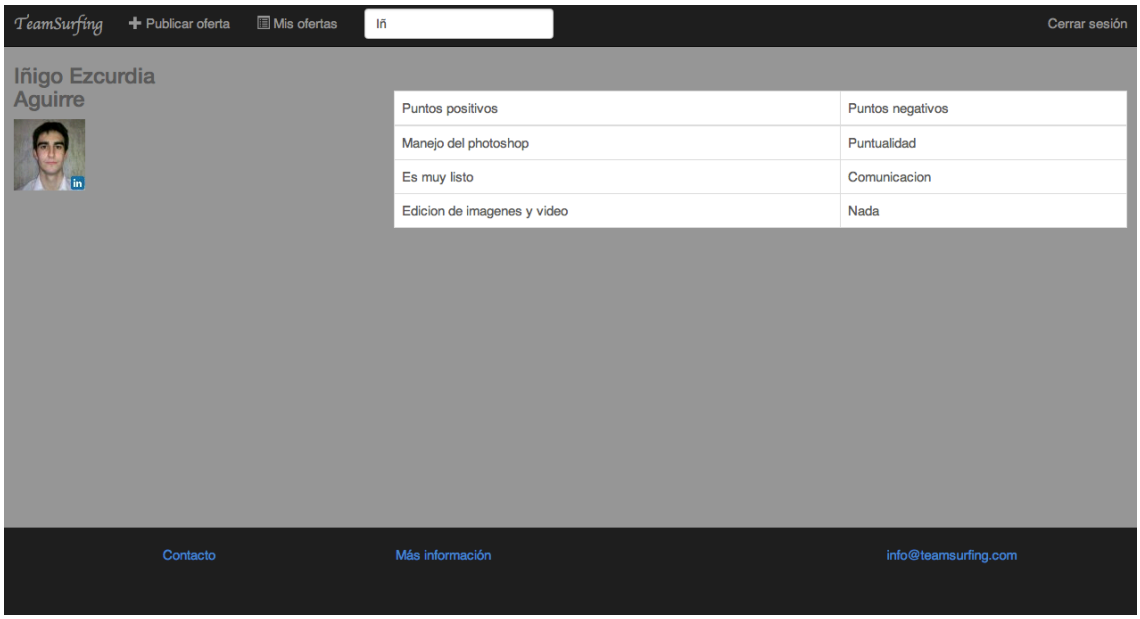


- Mis ofertas (parte empresas): aquí veremos un listado de las ofertas publicadas por el usuario actual



- Buscador (parte empresas): en todo momento tendremos en el menú superior un formulario de búsqueda en el que podremos consultar las personas sobre las que se han escrito valoraciones

- Perfil de usuarios (parte empresas): si nuestra búsqueda nos devuelve resultados (como vemos en la pantalla anterior) podemos acceder a la siguiente pantalla en la que vemos información mas detallada acerca de los usuarios.



5.2 Ficheros JavaScript

En este apartado adjuntamos los ficheros JavaScript utilizados para implementar el patrón modelo vista controlador con la ayuda de las librerías anteriormente comentadas.

5.2.1 Parte usuario normal

router.js (controlador)

```
var user;
var aplicacion = {
    views: {},
    models: {},
    loadTemplates: function(views, callback) {
        var deferreds = [];
        $.each(views, function(index, view) {
            if (aplicacion[view]) {
                deferreds.push($.get('usuarios/tpl/' + view + '.html', function(data) {
                    aplicacion[view].prototype.template = _.template(data);
                }, 'html'));
            } else {
                alert(view + " not found");
            }
        });
        $.when.apply(null, deferreds).done(callback);
    }
};

aplicacion.Router = Backbone.Router.extend({
    routes: {
        "": "home",
        "contactos": "contactos",
        "equipo": "favoritos",
        "ofertas" : "ofertas"
    },
    initialize: function() {
        this.$content = $(content);
    },
    home: function() {
        this.$content.html(new aplicacion.HomeView().render().el);
        $('.navbar-collapse').collapse('hide');
    },
    contactos: function() {
        _.each(aplicacion.misFavoritos.models, function (favorito) {
            aplicacion.linkedinContactCollection.remove(favorito.get('INId'));
        });
        this.$content.html(new aplicacion.ContactListView({model: aplicacion.linkedinContactCollection}).render().el);
        $('.navbar-collapse').collapse('hide');
    },
    favoritos: function() {
        aplicacion.misFavoritos.fetch({reset: true, data: {}});
        this.$content.html(new aplicacion.FavoriteListView({model: aplicacion.misFavoritos}).render().el);
        $('.navbar-collapse').collapse('hide');
    },
    ofertas: function() {
```

```
        this.$content.html(new aplicacion.OfertListView({model: aplicacion.listaOfertas}).render().el);
        $('.navbar-collapse').collapse('hide');
    }
});
function cargaInicial() {
    Parse.initialize("k4FMOdHOqNGrGDcpHsJoaA9mtOjFcNYfet1A7361", "B0MHq5nxjPb7vqZwk5thkc5CXAodVkxv1pNac9NS");
    aplicacion.loadTemplates(["HomeView", "ContactListItemView", "FavoriteListItemView", "OfertListItemView"],
    function () {
        aplicacion.router = new aplicacion.Router();
        Backbone.history.start();
    });

    IN.API.Profile("me")
            .fields(["id", "email-address","threeCurrentPositions"])
            .result(function(result, metadata){

        var companyname = '';

        user = new Parse.User();
        user.set("username", result.values[0].emailAddress);
        user.set("password", result.values[0].emailAddress);
        user.set("email", result.values[0].emailAddress);

        // Bucle para obtener el nombre de la empresa donde trabaja el usuario.
        _.each(result.values[0].threeCurrentPositions.values, function(position){
            if (position.isCurrent){
                companyname = position.company.name;
            }
        });

        user.signUp(null,{
            success: function(user){
                // Nada
            },
            error: function(user, error){
                if (error.code === Parse.Error.USERNAME_TAKEN){
                    Parse.User.logIn(result.values[0].emailAddress, result.values[0].emailAddress, {
                        success: function(user) {
                            // Nada

                            user = Parse.User.current();
                            user.set('company', companyname);
                            user.save(
                                {company: companyname},
                                {success: function(x) {
                                },
                                error: function(x, error) {
                                    console.log(error);
                                }
                            });

                            aplicacion.misFavoritos = new aplicacion.TeamCollection();
                            aplicacion.misFavoritos.fetch({reset: true, data: {}});
                            aplicacion.listaOfertas = new aplicacion.OfertCollection();
                            aplicacion.listaOfertas.fetch({reset: true, data: {}});

                        },
                        error: function(user, error) {
                            window.location.href = '../';

                        }
                    });
                }
            }
        });
```

```
    });


    IN.API.Connections("me")
        .fields("id", "firstName", "lastName", "pictureUrl")
        .result(function(result, metadata) {
            var lnk = [];

            for (var index in result.values) {

                profile = result.values[index];
                if (profile.firstName != "private") {
                    var fullName = profile.firstName + ' ' + profile.lastName;

                    if (!profile.pictureUrl) { profile.pictureUrl = "img/no-photo.png"; }
                    if (fullName.length > 50) { fullName = fullName.substring(0,50) + '...'};
                    var pos = {
                        profileID: profile.id,
                        firstName: profile.firstName,
                        lastName: profile.lastName,
                        fullName: fullName,
                        pictureUrl:  profile.pictureUrl,
                    };
                    lnk.push(pos);
                }
            }
            aplicacion.linkedinContactCollection = new aplicacion.INContactCollection(lnk);
            aplicacion.linkedinContactCollection.sort();
        });
}
```

Home.js (vista)

```
aplicacion.HomeView = Backbone.View.extend({
    initialize: function() {
        var self = this;
    },
    render: function() {
        this.$el.html(this.template());
        return this;
    }
});
```

ContactList.js (vista)

```
aplicacion.ContactListView = Backbone.View.extend({
    className:'row',
    initialize:function () {
        var self = this;
        this.model.on("reset", this.render, this);
        this.model.on("add", function (contacto) {
            self.$el.append(new aplicacion.ContactListItemView({model:contacto}).render().el);
        });
    },

    render:function () {
        this.$el.empty();
        _.each(this.model.models, function (contacto) {
            this.$el.append(new aplicacion.ContactListItemView({model:contacto}).render().el);
        }, this);
        return this;
```

```
    }
});



aplicacion.ContactListItemView = Backbone.View.extend({
    className: 'col-md-4 col-sm-6',
    events: {
        'click .addUser' : 'addUser'
    },
    initialize:function () {
        this.model.on("change", this.render, this);
        this.model.on("destroy", this.close, this);
    },

    render:function () {
        var data = _.clone(this.model.attributes);
        data.id = this.model.id;
        this.$el.html(this.template(data));
        return this;
    },

    addUser : function () {
           var $celda = this.$el;
           var usuario = this.model.attributes;

           $('.aceptar').on('click', function (e) {
           $celda.remove();
           $('#myModalAddUser').modal('hide');
            var userFavorite = new aplicacion.TeamProfile();
            userFavorite.set('INId', usuario.profileID);
            userFavorite.set('firstName', usuario.firstName);
            userFavorite.set('lastName', usuario.lastName);
            userFavorite.set('pictureUrl', usuario.pictureUrl);
            userFavorite.set('createdBy', Parse.User.current().id);
            userFavorite.set('puntosFuertes', $('#buenas').val());
            userFavorite.set('puntosDebiles', $('#malas').val());
            var acl = new Parse.ACL(Parse.User.current());
            acl.setPublicReadAccess(true);
            userFavorite.setACL(acl);
            userFavorite.save();
            aplicacion.misFavoritos.add(userFavorite);
            aplicacion.linkedinContactCollection.remove(userFavorite);
            $('#buenas').val('');
            $('#malas').val('');
           });
           $("#myModalAddUser").modal('toggle');
    }
});
```

FavoriteList.js (vista)

```
aplicacion.FavoriteListView = Backbone.View.extend({

    className:'row',

    initialize:function () {

        var self = this;
        this.model.on("reset", this.render, this);
        this.model.on("add", function (contacto) {
            self.$el.append(new aplicacion.FavoriteListItemView({model:contacto}).render().el);
        });
```

```
    },

    render:function () {
        this.$el.empty();
        _.each(this.model.models, function (contacto) {
            this.$el.append(new aplicacion.FavoriteListItemView({model:contacto}).render().el);
        }, this);
        return this;
    }

});

aplicacion.FavoriteListItemView = Backbone.View.extend({


    className: 'col-md-4 col-sm-6',

    events: {
        "click .removeUser" : "removeUser"
    },

    initialize:function () {
        this.model.on("change", this.render, this);
        this.model.on("destroy", this.close, this);
    },

    render:function () {
        var data = _.clone(this.model.attributes);
        data.id = this.model.id;

        this.$el.html(this.template(data));
        return this;
    },

    removeUser : function () {
           var $usuario = this.$el;

        var exFavorito =  aplicacion.misFavoritos.get(this.model.id);

        var linkedinUser = new aplicacion.INContact();
        linkedinUser.set("firstName", exFavorito.get("firstName"));
        linkedinUser.set("lastName", exFavorito.get("lastName"));
        linkedinUser.set("fullName", exFavorito.get("firstName") + " " + exFavorito.get("lastName"));
        linkedinUser.set("pictureUrl", exFavorito.get("pictureUrl"));
        linkedinUser.set("profileID", exFavorito.get("INId"));

        exFavorito.destroy({
            success: function (x) {
                aplicacion.linkedinContactCollection.add(linkedinUser);
                aplicacion.misFavoritos.remove(exFavorito);
            },
            error: function (myObject, error) {
                alert("Error");
            }
        });


           $('#myModalDelUser').on('hidden.bs.modal', function (e) {
            $usuario.remove();
           });

           $("#myModalDelUser").modal('toggle');
    }

});
```

OfertList.js (vista)

```
aplicacion.OfertListView = Backbone.View.extend({

    className:'row',

    initialize:function () {

        var self = this;
        this.model.on("reset", this.render, this);
        this.model.on("add", function (oferta) {
            self.$el.append(new aplicacion.OfertListItemView({model:oferta}).render().el);
        });

    },

    render:function () {
        this.$el.empty();
        _.each(this.model.models, function (oferta) {
            this.$el.append(new aplicacion.OfertListItemView({model:oferta}).render().el);
        }, this);
        return this;
    }

});

aplicacion.OfertListItemView = Backbone.View.extend({

    className: 'col-sm-6',

    events: {
        'click .aplicar' : 'aplicar'
    },


    fotos: {},

    initialize:function () {
        this.model.cargarEquipo();

        this.model.on("change", this.render, this);
        this.model.on("destroy", this.close, this);
    },



    render:function () {
        var data = _.clone(this.model.attributes);
        data.id = this.model.id;

        this.$el.html(this.template(data));

        for (var empleado in data.team){
            if (data.fotos){
                this.$('.panel-footer').append('<a href="' + data.team[empleado] + '">' +
                        '<img style="height: 30px; width: 30px; margin-right:5px;" src="' +
                        data.fotos[empleado] + '"></a>');
            }
```

```
            else{
                this.$('.panel-footer').append('<a href="' + data.team[empleado] + '">' +
                        '<img style="height: 30px; width: 30px; margin-right:5px;" ' +
                        ' src="img/no-photo.png"></a>');
            }
        }
        return this;
    },

    aplicar: function(){

        var Aplicar = Parse.Object.extend("candidatos");
        var candidato = new Aplicar();

        candidato.set('idUsuario', Parse.User.current().id);
        candidato.set('idOferta', this.model.id);
        candidato.setACL(new Parse.ACL(Parse.User.current()));


        candidato.save(null, {
            success: function(comentario) {
                $("#mensajeOK").modal('toggle');
            },
            error: function(comentario, error) {
                alert("Ocurrió un error, pruebe más tarde");
            }
        });

    }

});
```

model-Linkedin.js (modelo)

```
aplicacion.INContact = Backbone.Model.extend({
    idAttribute: 'profileID'
});


aplicacion.INContactCollection = Backbone.Collection.extend({
    model: aplicacion.INContact,

    initialize : function () {
           this.sort_key = 'firstName';
    },

    comparator: function(a, b) {
           a = a.get(this.sort_key).charAt(0);
           b = b.get(this.sort_key).charAt(0);
           return  a > b ?  1
            :   a < b ? -1
               :           0;
           }

});
```

model-Ofertas.js (modelo)

```
aplicacion.Ofert = Parse.Object.extend({

    className: 'oferta',

    defaults: {
        name: '',
        minidescription: '',
        description: '',
        company: '',
        fotos: []
    },

    cargarEquipo: function() {
        self = this;

        var equipo = [];

        for (var index in this.attributes.team) {
            var x = this.attributes.team[index];
            equipo.push('url=http://'.concat(x.substring(x.search('www'), x.length)));
        }

        IN.API.Profile(equipo).fields('pictureUrl').result(function(resultado){
            self.attributes.fotos = [];
            for (var empleado in resultado.values){
                if (resultado.values[empleado].pictureUrl){
                    self.attributes.fotos.push(resultado.values[empleado].pictureUrl);
                }
                else{
                    self.attributes.fotos.push('img/no-photo.png');
                }
            }
            self.trigger('change');
        });
    }


});

aplicacion.OfertCollection = Parse.Collection.extend({
    model: aplicacion.Ofert
});
```

model-Parse.js (modelo)

```
aplicacion.TeamProfile = Parse.Object.extend({
    className: "Favoritos",
    defaults: {
      INId: "",
      firstName: "",
      lastName: "",
      pictureUrl: ""
    }
});
aplicacion.TeamCollection = Parse.Collection.extend({
    model: aplicacion.TeamProfile,
    fetch: function(options) {
        this.query = new Parse.Query(aplicacion.TeamProfile);
        this.query.equalTo("createdBy", Parse.User.current().id);
        Parse.Collection.prototype.fetch.apply(this, arguments);
    }
});
```

5.2.2 Parte empresas

router.js (controlador)

```
var user;
var aplicacion = {
    views: {},
    models: {},
    loadTemplates: function(views, callback) {
        var deferreds = [];
        $.each(views, function(index, view) {
            if (aplicacion[view]) {
                deferreds.push($.get('empresas/tpl/' + view + '.html', function(data) {
                    aplicacion[view].prototype.template = _.template(data);
                }, 'html'));
            } else {
                alert(view + " not found");
            }
        });
        $.when.apply(null, deferreds).done(callback);
    }
};

aplicacion.Router = Backbone.Router.extend({
    routes: {
        '': 'home',
        'publicar' : 'publicar',
        'ofertas' : 'misofertas',
        'usuarios/:id' : 'fichausuario'
    },
    initialize: function() {
        aplicacion.headerView = new aplicacion.HeaderView();
        $('.header').html(aplicacion.headerView.render().el);
        $('.navbar-collapse').collapse('hide');
        this.$content = $('#content');
    },
    home: function() {
        $('.navbar-search').removeClass('open');
        this.$content.html(new aplicacion.HomeView().render().el);
        $('.navbar-collapse').collapse('hide');
    },
    publicar: function () {
        $('.navbar-search').removeClass('open');
        this.$content.html(new aplicacion.NewOfertView({model: new aplicacion.OfertCollection()}).render().el);
        $('.navbar-collapse').collapse('hide');
    },
    misofertas: function() {
        $('.navbar-search').removeClass('open');
        aplicacion.listaOfertas = new aplicacion.OfertCollection();
        aplicacion.listaOfertas.fetch({reset: true, data: {}});
        this.$content.html(new aplicacion.OfertListView({model: aplicacion.listaOfertas}).render().el);
        $('.navbar-collapse').collapse('hide');
    },
    fichausuario: function(id) {
        $('.navbar-search').removeClass('open');
        aplicacion.usuario = new aplicacion.User({'INId' : id});
        this.$content.html(new aplicacion.UserView({model: aplicacion.usuario}).render().el);
        $('.navbar-collapse').collapse('hide');
    }
});
```

```
function cargaInicial() {
    Parse.initialize("k4FMOdHOqNGrGDcpHsJoaA9mtOjFcNYfet1A7361", "B0MHq5nxjPb7vqZwk5thkc5CXAodVkxv1pNac9NS");
    aplicacion.loadTemplates(["HomeView", "NewOfertView", "OfertListView", "HeaderView", "UserListItemView", "UserView"],
    function () {
        aplicacion.router = new aplicacion.Router();
        Backbone.history.start();
    });
    IN.API.Profile("me").fields(["id", "email-address","threeCurrentPositions"]).result(function(result, metadata){
        var companyname = '';
        user = new Parse.User();
        user.set("username", result.values[0].emailAddress);
        user.set("password", result.values[0].emailAddress);
        user.set("email", result.values[0].emailAddress);
        _.each(result.values[0].threeCurrentPositions.values, function(position){
            if (position.isCurrent){
                companyname = position.company.name;
            }
        });

        user.signUp(null,{
            success: function(user){
                if (user.attributes.empresa){
                    aplicacion.usuarios = new aplicacion.TeamCollection();
                    aplicacion.usuarios.fetch({reset: true, data: {}});
                }
                else {
                    window.location.href = '../';
                }
            },
            error: function(user, error){
                if (error.code === Parse.Error.USERNAME_TAKEN){
                    Parse.User.logIn(result.values[0].emailAddress, result.values[0].emailAddress, {
                        success: function(user) {
                            if (user.attributes.empresa){
                                aplicacion.usuarios = new aplicacion.TeamCollection();
                                aplicacion.usuarios.fetch({reset: true, data: {}});
                            }
                            else {
                                window.location.href = '../';
                            }
                            user = Parse.User.current();
                            user.set('company', companyname);
                            user.save(
                                {company: companyname},
                                {success: function(x) {
                                    aplicacion.usuarios = new aplicacion.TeamCollection();
                                    aplicacion.usuarios.fetch({reset: true, data: {}});
                                },
                                error: function(x, error) {
                                    console.log(error);
                                }
                            });
                        },
                        error: function(user, error) {
                            window.location.href = '../';
                        }
                    });
                }
            }
        });
    });
}
```

headerview.js (vista)

```
aplicacion.HeaderView = Backbone.View.extend({

    initialize: function () {
        aplicacion.searchResults = new aplicacion.TeamCollection();
        aplicacion.searchresultsView = new aplicacion.UserListView(
           {model: aplicacion.searchResults, className: 'dropdown-menu'}
        );
    },

    render: function () {
        $(this.el).html(this.template());
        $('.navbar-search', this.el).append(aplicacion.searchresultsView.render().el);
        return this;
    },

    events: {
        "keyup .search-query": "search",
        "keypress .search-query": "onkeypress"
    },

    search: function () {
        var key = $('#busqueda').val();

        // Limito los resultados a 5.
        aplicacion.searchResults.reset(aplicacion.usuarios.findByName(key).slice(0, 5));

        setTimeout(function () {
            if (key.length == 0 || aplicacion.searchResults.length == 0){
                $('.navbar-search').removeClass('open');
            }
            else{
                $('.navbar-search').addClass('open');
            }
        });
    },

    onkeypress: function (event) {
        if (event.keyCode == 13) {
            event.preventDefault();
        }
    }
});
```

home.js (vista)

```
aplicacion.HomeView = Backbone.View.extend({
    initialize: function() {
        var self = this;
    },
    render: function() {
        this.$el.html(this.template());
        return this;
    }
});
```

newOfert.js (vista)

```
aplicacion.NewOfertView = Backbone.View.extend({
    events: {
        'click #crear': 'crearOferta',
        'click #borrar': 'limpiar',
        'keypress #linkedins': 'crearOferta'
    },

    initialize: function() {
        //var self = this;
    },

    render: function() {
        this.$el.html(this.template());
           return this;
    },

    limpiar: function(e) {
        this.$('#titulo').val('');
        this.$('#descripcion').val('');
        this.$('#linkedins').val('');
    },

    crearOferta: function(e) {

        // Si hemos pulsado el enter o hecho click sobre el boton Crear Oferta.
        if (e.keyCode === 13 || e.type === 'click'){
            if (this.$('#titulo').val() === '' || this.$('#descripcion').val() === '') return;

            var currentUser = Parse.User.current();

            if (currentUser){
                this.model.create({
                    name: this.$('#titulo').val(),
                    description: this.$('#descripcion').val(),
                    minidescription: this.$('#descripcion').val().substring(0, 50),
                    team: this.$('#linkedins').val().split(','),
                    company: currentUser.attributes.company,
                    createdBy: Parse.User.current().id,
                    ACL: new Parse.ACL({"*":{'read':true}})
                });
                aplicacion.router.misofertas();
            }
            else {
                window.location.href = '../';
            }
        }
    }
});
```

ofertList.js (vista)

```
aplicacion.OfertListView = Backbone.View.extend({
    initialize:function () {
        this.model.on("reset", this.render, this);
        this.model.on("destroy", this.close, this);
    },

    render:function () {
        this.$el.html(this.template());
        _.each(this.model.models, function (ofert) {
            if (ofert.attributes.createdBy === Parse.User.current().id){
                $('#tabla').append(new aplicacion.OfertListItemView({model:ofert}).render().el);
            }
        }, this);
       return this;
    }
});

aplicacion.OfertListItemView = Backbone.View.extend({
    tagName: 'tr',
    initialize:function () {
        this.model.on("change", this.render, this);
        this.model.on("destroy", this.close, this);
    },

    render: function() {
        // Clear existing row data if needed
        this.$el.empty();
        this.$el.append('<td>' + this.model.get('name') + '</td>');
        this.$el.append('<td>' + this.model.get('minidescription') + '...' + '</td>');
        this.$el.append('<td><p><button class="btn btn-primary btn-xs" data-title="Edit" data-toggle="modal"' +
            'data-target="#edit" data-placement="top" rel="tooltip">' +
            '<span class="glyphicon glyphicon-pencil"></span></button></p></td>');
        return this;
    }

});
```

userList.js (vista)

```
aplicacion.UserListView = Backbone.View.extend({
    tagName:'ul',
    className:'nav nav-list',
    initialize:function () {
        var self = this;
        this.model.bind("reset", this.render, this);
        this.model.bind("add", function (user) {
            $(self.el).append(new aplicacion.UserListItemView({model:user}).render().el);
        });
    },
    render:function () {
        $(this.el).empty();
        _.each(this.model.models, function (user) {
            $(this.el).append(new aplicacion.UserListItemView({model:user}).render().el);
        }, this);
        return this;
    }
});
```

```
aplicacion.UserListItemView = Backbone.View.extend({
    tagName:"li",
    initialize:function () {
        this.model.bind("change", this.render, this);
        this.model.bind("destroy", this.close, this);
    },
    render:function () {
        $(this.el).html(this.template(this.model.toJSON()));
        return this;
    }
});
```

userView.js (vista)

```
aplicacion.UserView = Backbone.View.extend({
    initialize: function() {
        var self = this;
        this.model.on("change", this.render, this);
    },
    render: function() {
        var self = this;

        this.$el.html(this.template(self.model.attributes));

        $('.comentarios').empty();
        _.each(self.model.attributes.comentarios, function (comentario) {

            $('.comentarios').append(
                    '<tr><td>' + comentario.bueno + '</td>' +
                    '<td>' + comentario.malo + '</td></tr>'
            );
        }, this);
        return this;
    }
});
```

model-Parse.js (modelo)

```
aplicacion.TeamProfile = Parse.Object.extend({
    className: "Favoritos",
    defaults: {
      INId: "",
      firstName: "",
      lastName: "",
      pictureUrl: ""
    }
});

aplicacion.TeamCollection = Parse.Collection.extend(({

    model: aplicacion.TeamProfile,
    fetch: function(options) {
        this.query = new Parse.Query(aplicacion.TeamProfile);
        Parse.Collection.prototype.fetch.apply(this, arguments);
    },
```

```
    findByName:function (key) {
        var lista = [];
        if (key.length > 0){
            $.each(aplicacion.usuarios.toJSON(), function(i, usuario) {
                      if ((usuario.firstName.toLowerCase().indexOf(key.toLowerCase()) > -1) ||
                      (usuario.lastName.toLowerCase().indexOf(key.toLowerCase()) > -1) || ((usuario.firstName + ' ' +
                      usuario.lastName).toLowerCase().indexOf(key.toLowerCase())> -1)){
                      var usr = {firstName : usuario.firstName,
                               lastName : usuario.lastName,
                               pictureUrl : usuario.pictureUrl,
                               INId : usuario.INId};

                      var encontrado = false;
                      for (var usuario in lista){
                        if (lista[usuario].INId == usr.INId){
                            encontrado = true;
                            break;
                        }
                    }
                    if (!encontrado) lista.push(usr);
                }
            });
        }

        lista.sort(function(a, b) {
            if (a.firstName < b.firstName) return -1;
            if (a.firstName > b.firstName) return 1;
            return 0;
        });

        return lista;
    }

}));
```

model-User.js (modelo)

```
aplicacion.User = Backbone.Model.extend({

    defaults : {
        INId: "",
        firstName: "Prueba",
        lastName: "Apellido",
        pictureUrl: "img/no-photo.png",
        comentarios: []
    },

    initialize : function() {
        var self = this;

        IN.API.Profile(self.attributes.INId)
           .fields(["id", "firstName","lastName", "pictureUrl"])
           .result(function(result, metadata){
            self.attributes.firstName = result.values[0].firstName;
            self.attributes.lastName = result.values[0].lastName;

            if (!result.values[0].pictureUrl){
                self.attributes.pictureUrl = 'img/no-photo.png';
            }
            else{
                self.attributes.pictureUrl = result.values[0].pictureUrl;
```

```
            }

            self.trigger('change');
        });

        self.attributes.comentarios = [];

        _.each(aplicacion.usuarios.toJSON(), function(usuario) {
           if (usuario.INId == self.attributes.INId){
               self.attributes.comentarios.push({bueno: usuario.puntosFuertes, malo : usuario.puntosDebiles});
           }
        });

    }
});
```

ofertas.js (modelo)

```
aplicacion.Ofert = Parse.Object.extend({
    className: 'oferta',
    defaults: {
        name: '',
        minidescription: '',
        description: '',
        team: ''
    }
});

aplicacion.OfertCollection = Parse.Collection.extend({
    model: aplicacion.Ofert,
    fetch: function(options) {
        this.query = new Parse.Query(aplicacion.Ofert);
        Parse.Collection.prototype.fetch.apply(this, arguments);
    }
});
```