Diseño e implementación de una aplicación multidispositivo en un entorno HTML5

Alvaro Martínez Rodríguez

Resumen—*mobCar* es una aplicación móvil pensada para cubrir la necesidad de muchos alumnos universitarios de compartir coche para acudir a la universidad. Actualmente ya está implementada para las plataformas nativas Android e iOS y está en funcionamiento con alumnos de la Universidad Autónoma de Barcelona. Este proyecto se basa en la migración de dicha aplicación a un entorno HTML5 para que pueda ser accesible desde cualquier dispositivo, ya sea móvil, tablet, ordenador portátil u ordenador de sobremesa y se implementará usando el framework AngularJS para la lógica y el framework Bootstrap para el diseño.

Palabras clave— Compartir vehículo, aplicación multidispositivo, migración a entorno HTML5, AngularJS, diseño adaptativo

Abstract— *mobCar* is a mobile application thought to cover the need of many university pupils of share car to come to the university. Nowadays it is already implemented for the native platforms Android and iOS and it is working with pupils of the Autonomous University of Barcelona. This project is based on the migration of the above mentioned application to an HTML5 environment in order that it could be accessible from any device, mobile, tablet, laptop or desktop and it will be implemented with the framework AngularJS for the lógic and with the framework Bootstrap for the design.

Index Terms— Share vehicle, multidevice application, migration to HTML5 environment, AngularJS, responsive design

1 Introducción

Cualquier estudiante que se haya tenido que desplazar a la universidad en transporte público sabe lo costoso que es tanto en tiempo como en dinero y seguro que en algún momento se le ha pasado por la mente la idea de compartir coche.

A su vez, cualquier estudiante que haya dispuesto de coche propio para acudir a la universidad y se haya encontrado caravanas quilométricas en las carreteras seguro que se le ha pasado por la mente la idea de que, si todos compartieran coche, se podría reducir esta caravana considerablemente.

Mi experiencia personal es una mezcla de ambos casos: durante un año acudí a la universidad en transporte público y, a pesar de estar a tan solo 25km de distancia, el trayecto superaba la hora y media, y si además se le añaden los más de 150€ que costaba el desplazamiento cada tres meses, resultaba una experiencia poco agradable. Los tres años restantes de universidad tuve la suerte de compartir coche con tres compañeros de universidad, la duración del trayecto bajó a un rango de entre 20 y 30 minutos, el precio descendió a 30€ mensuales (90€ trimestrales) y la compañía de mis compañeros compensaba con creces la posible caravana que pudiéramos encontrar.

Ese primer año, pensé varias veces en que sería fantástico que existiera una solución que me permitiera ponerme en contacto con estudiantes que hicieran el mismo trayecto que yo y poder compartir medios y transporte, pero por desgracia no poseía los conocimientos necesarios para crear dicha solución.

Una vez en el cuarto curso, realicé las prácticas de empresa en la empresa Sigma Gestión Universitaria [1]. En esta empresa, los compañeros del equipo al cual fui asignado, me comentaron que estaban desarrollando una aplicación que permitía a los estudiantes universitarios crear y buscar viajes con el fin de compartir vehículo en el trayecto hacia o desde la universidad y que necesitaban migrar dicha aplicación móvil para Android e iOS a un entorno web bajo HTML5. No dudé en aceptar la propuesta, ya que veía la creciente necesidad de que esta aplicación pudiera ser accesible desde cualquier dispositivo y de esta forma pudiera utilizarse por el máximo número de alumnos posibles, y así tuvieran la posibilidad de compartir vehículo para ahorrar tiempo y dinero.

Para realizar la migración correctamente se ha usado CSS y Javascript. Como framework para la implementación, después de investigar y probar varias alternativas, se optó por AngularJS [2],[3].

2 Objetivos

El objetivo principal del proyecto es hacer una migración de una aplicación, existente en las plataformas nativas Android e iOS, a un entorno HTML5 y que esta sea multidispositivo.

Ya que hay que realizar una migración y no implementar la aplicación de cero, hay varias restricciones impuestas desde el principio:

- No se pueden modificar ni el servidor ni la base de

- *E-mail de contacto: alvaromr91@gmail.com*

- *Mención realizada: Computación*
- *Trabajo tutorizado por: Ramon Baldrich*
- *Curso 2014/15*

datos existente, por lo tanto la aplicación se debe adaptar al sistema y al formato que estos utilicen.
- Las funcionalidades ya están especificadas y la aplicación no podrá tener ni más ni menos de estas funcionalidades.
- El diseño debe parecerse lo máximo posible al original para mayor comodidad del usuario.

Teniendo en cuenta estas restricciones y el objetivo principal se extrajeron unos requisitos que había que cumplir, estos se agrupan en funcionales y no funcionales:

Requisitos funcionales:

La aplicación debería tener:

1. Sistema de login que permita a un usuario registrado acceder a la aplicación introduciendo su NIU y su contraseña.
2. Pantalla con un formulario que permita al usuario ver y modificar su información de perfil.
3. Formulario que permita al usuario crear un viaje con toda la información necesaria.
4. Formulario que permita al usuario buscar un viaje y que, si existieran viajes con los parámetros deseados, se mostrasen en una lista que permitiera escoger entre estos, ver toda la información relacionada, escoger el número de asientos y reservar dicho viaje.
5. Pantalla que permitiera al usuario ver todos sus viajes activos, siendo pasajero o conductor, junto con todas las peticiones de viaje.
6. Dar la posibilidad a un conductor de aceptar o rechazar una petición de viaje.

Requisitos no funcionales:

7. Implementar correctamente el sistema de geolocalización con la API de GoogleMaps [4] para poder gestionar correctamente el origen del viaje. El destino vendrá predefinido por la universidad.
8. Gestionar correctamente las conexiones al servidor para poder intercambiar la información necesaria cuando se precise.
9. Controlar la autentificación del usuario asegurando siempre la privacidad de sus credenciales.
10. La aplicación deberá adaptarse correctamente a cualquier tamaño de pantalla de dispositivo móvil, Tablet u ordenador de sobremesa.
11. Aprender a usar el framework AngularJS [3].

3 Estado del arte

La plataforma online de referencia en nuestro ámbito para compartir vehículo es BlaBlaCar [5], la cual es una red social creada para compartir tus viajes, ya seas conductor o pasajero, para trayectos cortos o largos. Esta plataforma online es una de las pioneras en el ámbito de compartir el transporte, convirtiéndose en una alternativa al transporte público real para millones de usuarios.

Buscando alternativas, se encontró en un artículo de una web llamada madrideducacion.es [6] una referencia a una iniciativa de *BlaBlaCar* llamada Campus [7] que, el día que se consultó (mostrado en la referencia), parecía no estar disponible. Según se comentaba en dicho artículo, era una plataforma propuesta por *BlaBlaCar* para compartir coche en las principales universidades madrileñas, que a fecha del 21 de marzo de 2012 era utilizada por cientos de estudiantes de dichas universidades.

Existe otra plataforma llamada *Amovens* [8], que como la anterior, también te permite buscar y crear viajes con el fin de compartir el trayecto.

Como las anteriores, hay varias alternativas más como por ejemplo: *carpooling* [9], *compartococheˌcom* [10], *viajamosjuntos.com* [11], con las que también puedes buscar un acompañante para tu viaje o un conductor, pero todas estas están pensadas para viajes de ciudad a ciudad y no están enfocadas para viajes dentro de una misma localidad.

Sin embargo, otras sí que te dan esta posibilidad como por ejemplo *carpling* [12], que dispone de un apartado donde, una vez registrado, te permite buscar y crear viajes con origen o destino tu universidad [13] o *xarxaeco* [14], un proyecto nacido de la voluntad de fomentar un uso más racional del coche entre la población, que también te permite buscar viajes desde cualquier dirección hasta una amplia cantidad de campus de diferentes universidades de Barcelona.

El problema reside en que, como ya se ha comentado, ninguna de estas plataformas o aplicaciones se centra en viajes dentro de una misma ciudad, sino en viajes de ciudad a ciudad, además, las que disponen de esta opción, no son muy accesibles ni conocidas, por lo que *mobCar* se presenta como una solución a esta necesidad.

4 Metodologia

Para el desarrollo de *mobCar* se ha seguido un modelo de metodología evolutivo, el cual se divide en las fases de: Análisis de requerimientos, diseño, codificación y prueba. En este tipo de modelo de proyecto, una vez finalizadas las cuatro fases se entiende que se ha completado una iteración y se vuelve a empezar para arreglar una implementación errónea o para seguir con otra parte del proyecto.

Como resultado se van creando versiones de la aplicación caracterizadas por las funcionalidades que se van añadiendo en cada una de estas.

Este apartado se va a dividir en cuatro bloques, análisis de requerimientos, diseño, codificación y pruebas. En cada uno de los bloques se explicará la metodología que se siguió para completar cada una de las fases del proyecto.

4.1 Análisis de requerimientos

Debido a que antes de empezar este proyecto ya existía una versión acabada de la aplicación *mobCar*, esta fase ha sido ligeramente diferente a la de un proyecto normal. Esta fase consistió en investigar las funcionalidades de la aplicación móvil ya existente y en una entrevista con sus creadores para recabar posibles requerimientos que no se pudieran encontrar explorando la aplicación.

En esta fase se creó un diagrama de Gantt que distribuía las tareas necesarias para cumplir cada objetivo en el tiempo y que permitió una correcta planificación y organización a la hora de implementar la aplicación.

4.2 Diseño

En este subapartado se va a comentar el diseño visual de la

aplicación y la metodología seguida para llegar al resultado actual.

Debido a que uno de los objetivos principales del proyecto era que la aplicación fuera multidispositivo, esta fase ganó una gran importancia. Al basarse en HMTL5 y por tanto en el uso de *web browsers* el problema se resuelve diseñando la aplicación de manera *responsive [16]* y evitando el uso de herramientas no estándares (*Flash*, …)

La aplicación debía adaptarse a cualquier tamaño de pantalla, para cumplir este objetivo se crearon tres diseños diferentes: móvil, tablet y ordenador de sobremesa. Para ello se usó el framework para diseño *Bootstrap* [15], el mismo que usa la red social Twitter, que permitió diseñar la aplicación de manera *responsive*[16], es decir, que respondiera a los tamaños de pantalla y se adaptara en consecuencia, creando así una aplicación web con CSS y Javascript dinámica.

Este framework pone a nuestra disposición una serie de herramientas para poder crear el diseño responsive que buscamos, en concreto, utiliza la inserción de clases para lograrlo.

Estas clases podrían dividirse en dos tipos diferentes: las própias del *Grid system* [17] que ofrece la posibilidad de dividir la pantalla en doce columnas generadas automáticamente, y las correspondientes a las *Responsive utilities*[18] que permiten mostrar un ocultar cualquier elemento de la vista en el tamaño de pantalla que se desee.

Las primeras corresponden a las clases *.col*. Estas clases se añadirían a la sección del código HMTL5 de la vista que se desee que responda al tamaño de pantalla. Las clases *.col* tienen como parámetros el tamaño de pantalla en el que se quiere aplicar el cambio y el número de columnas que medirá. Como ejemplo se podría crear una sección que en pantallas de dispositivos móviles ocupase cuatro columnas, esta quedaría de la siguiente forma: *.col-xs-4*. En la Tabla 1 se pueden observar los cuatro prefijos que componen el sistema de grid, así como información sobre los tamaños de pantalla en los que se realizarán los cambios.

	Extra small devices Phones (<768px)	**Small devices** Tablets (≥768px)	**Medium devices** Desktops (≥992px)	**Large devices** Desktops (≥1200px)
Container width	None (auto)	750px	970px	1170px
Class prefix	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
Column width	Auto	~62px	~81px	~97px

Tabla 1. En esta tabla se muestran los tamaños de pantalla en los que actuan cada uno de los prefijos y el tamaño de los contenedores y columnas que tendrían en cada ancho de pantalla.

El segundo tipo, las "Responsive utilities" usan diferentes prefijos que se pueden utilizar para mostrar o no una sección en un determinado ancho de pantalla, detallados en la Tabla 2.

Estas clases se usarían en secciones del código que se desearan ocultar o mostrar en los diferentes tamaños de pantalla.

	Phones (<768px)	Tablets (≥768px)	Desktops (≥992px)	Desktops (≥1200px)
`.visible-xs`	Visible	Hidden	Hidden	Hidden
`.visible-sm`	Hidden	Visible	Hidden	Hidden
`.visible-md`	Hidden	Hidden	Visible	Hidden
`.visible-lg`	Hidden	Hidden	Hidden	Visible
`.hidden-xs`	Hidden	Visible	Visible	Visible
`.hidden-sm`	Visible	Hidden	Visible	Visible
`.hidden-md`	Visible	Visible	Hidden	Visible
`.hidden-lg`	Visible	Visible	Visible	Hidden

Tabla 2. En esta tabla se pueden observar los prefijos que se deberían de usar en caso de querer mostrar u ocultar una sección del código HTML en los diferentes tamaños de pantalla.

Una vez vistas estas dos posibilidades que nos da Bootstrap, se podría aplicar una metodología muy sencilla para diseñar la aplicación de manera responsive de la forma siguiente:

1. Crear una sección, o como se denominan en HTML5 , la cual contendría todo el trozo de código que se quisiera adaptar. A esta sección se le añadiría una clase llamada *.row* que podría contener en su interior doce columnas como máximo.
2. Crear dentro de la sección, tantas divisiones como se necesitara, que serían las que contendrían la información de la vista.
3. Añadir una de las clases vistas anteriormente, ya sea del sistema de grid, de las responsive utilities o ambas, a la división deseada, para asignarle un tamaño en el que se mostrara y las columnas que ocupará. Se pueden añadir tantas clases como sea necesarias para obtener el resultado deseado.

A continuación se mostrará un pequeño ejemplo siguiendo los tres pasos para ver el resultado conseguido al seguir este método. A este ejemplo se le añadirá estilo para poder explicar mejor el resultado

1. Se añade la clase *.row*:
```
<div class="row"> </div>
```
2. Se crean las divisiones que se deseen:
```
<div class="row">
    <div></div>
    <div></div>
    <div></div>
</div>
```
3. Se añaden las clases para formatear las divisiones:
```
<div class="row">
    <div class="hidden-xs col-lg-3 col-md-5 sec-
    cionIzquierda">
```

```
        Sección oculta en tamaño móvil, en tamaño
        mediano ocupa 5 columnas y en tamaño grande
        3 columnas
    </div>
    <div class="col-lg-7 col-md-7 col-xs-12 sec-
    cionCentral">
        Sección común en los tres tamaños, en tamaño
        móvil ocupa el total de columnas y en tamaño
        mediano y grande ocupa 7 columnas
    </div>
    <div class="visible-lg col-lg-2 seccionDere-
    cha">
        Sección solo visible en tamaños de pantalla
        grandes en las que ocupa solo 2 columnas
    </div>
</div>
```

Estilo:

```
. seccionIzquierda {
            background-color: blue;
            height: 300px;
}
. seccionCentral {
            background-color: yellow;
            height: 300px;
}
. seccionDerecha {
            background-color: green;
            height: 300px;
}
```

El resultado en una pantalla de más de 1200 píxeles serían tres secciones, tal y como se puede observar en la Figura 1. La sección izquierda tendría un ancho de 3 columnas (291p), 300 píxeles de alto y color de fondo azul. La sección central estaría formada por 7 columnas (679 píxeles de ancho), sería de 300 píxeles de altura y de color amarillo y finalmente la sección derecha, que solo se vería con este ancho de pantalla, tendría 2 columnas (194 de ancho), 300 píxeles de alto y fondo verde.

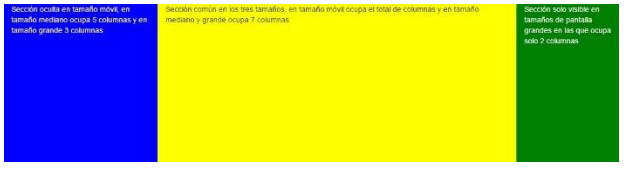

Figura 1. Resultado del código de ejemplo extraído de una pantalla de más de 1200 píxeles de ancho, se pueden observar las tres secciones tal como se había pretendido en el código.

El resultado del ejemplo en una pantalla mediana, es decir, de igual o mayor de 992 píxeles serían la sección izquierda y la central, tal y como se puede comprobar en la Figura 2. La sección izquierda esta vez tendría un ancho de 5 columnas (405 píxeles), el color y la altura no varían. Tal y como se pretendía la sección izquierda no aparece en este ancho de pantalla debido a la clase `visible-lg`, que solo permite que se vea esa sección en pantallas de más de 1200 píxeles.

Figura 2. Resultado del código de ejemplo en una pantalla de entre 992 y 1199 píxeles. Solo se observan dos secciones y una de ellas se ha ampliado para adaptarse a la pantalla.

Por último, el resultado en una pantalla de menos de 768 píxeles sería únicamente la sección central, tal y como se muestra en la Figura 3. En esta resolución de pantalla se observaría solamente la sección central, tal y como se pretendía con las clases `hidden-xs` de la sección izquierda, que oculta esta división cuando se renderiza la vista en una pantalla pequeña y la clase `visible-lg` de la sección derecha, que solo se muestra en pantallas grandes.

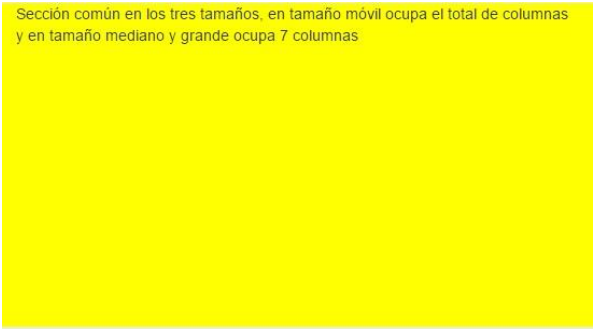

Figura 3. Resultado del código de ejemplo extraído de una pantalla e menos de 768 píxeles d ancho, se puede observar que, tal u como se pretendía, solo se muestra la sección central.

Repitiendo estos sencillos pasos llegaríamos a obtener una aplicación totalmente responsive que se adaptase tal y como quisiéramos a cualquier tamaño de pantalla, adaptando los tamaños con las clases `col-*-*` o mostrando u ocultando los elementos pertinentes con las clases `visible-* y hidden-*`.

De esta forma se creó la aplicación *mobCar* y para la pantalla principal se obtuvieron los siguientes resultados:

- En tamaño "lg" (≥ 1200 píxeles) se obtuvo el resultado de la Figura 4, en esta figura se puede ver que la aplicación está dividida en dos secciones, la sección principal y el perfil, que serán las que irán cambiando dependiendo de cada tamaño de cada resolución de pantalla.
- Para pantallas "md" y "sm" (≥ 768 píxeles y ≥ 992 píxeles) se obtuvo el resultado de la Figura 5, en la que se puede observar que el perfil ha desaparecido de la parte derecha de la pantalla y se ha creado una nueva pestaña en la barra de pestañas.
- Y para tamaño "xs" (< 768 píxeles) se obtuvo el resultado de la Figura 6, en la que se observa un cambio significativo en el diseño debido a que esta se adaptó a la aplicación ya existente de *mobCar*, concretamente a su versión Android.

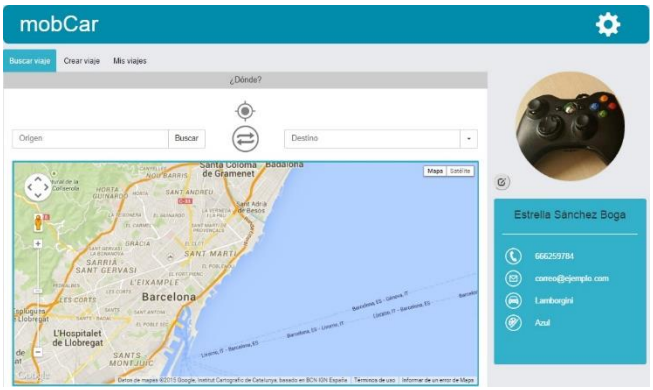

Figura 4. Pantalla principal de la aplicación en su resolución máxima.

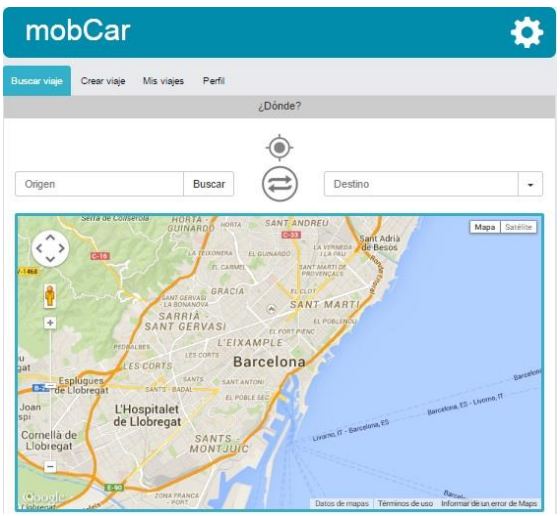

Figura 5. Aplicación vista desde una pantalla media (md) o pequeña (xs). Se puede observar la desaparición del perfil a la derecha y la aparición de una pestaña en la barra de pestañas llamada Perfil.

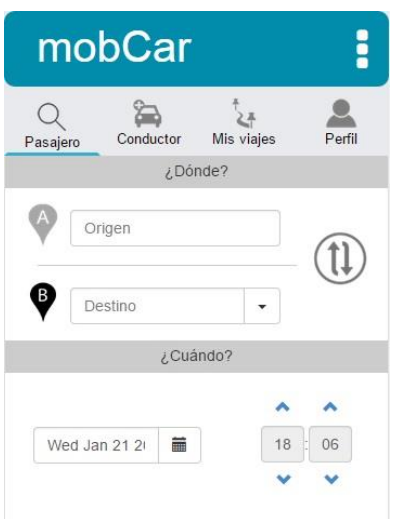

Figura 6. Pantalla principal vista desde un dispositivo con una pantalla de menos de 768 píxeles de ancho. Se puede observar un gran cambio con respecto a las anteriores debido al diseño especial para dispositivos móviles.

4.3 Codificación

Para la implementación de la parte lógica de la aplicación se ha usado prioritariamente el framwork AngularJS junto con código JavaScript.

Se escogió este framework debido a las buenas recomendaciones que se encontraron a la hora de buscar frameworks y a su facilidad de aprendizaje y uso, además proporciona una gran cantidad de utilidades que hacen mucho más fácil y rápido programar aplicaciones web.

AngularJS es un framework de JavaScript de código abierto y mantenido por Google que esta especialmente pensado para crear aplicaciones de una sola página. Este framework sigue el patrón Modelo-Vista-Controlador por lo que apuesta por la separación entre la presentación, los datos y los componentes lógicos utilizando el "two-way data binding" que no es más que el intercambio de datos en vivo entre la vista y el controlador [19].

AngularJS te permite crear controladores [20] que serán los encargados de gestionar la lógica de la presentación, en ellos podremos tener el código necesario para inicializar una aplicación, gestionar los eventos y mostrar los resultados en la vista. Estos controladores podrán agregarse al DOM (vista) mediante la directiva *ngController* añadiendo de la siguiente forma en una etiqueta HTML: `<div ng-controller="miControlador">`. De esta forma tendremos disponible dentro del ámbito de esa etiqueta todo el código de `miControlador`. AngularJS también te permite crear *factorías* [21] que serán las encargadas de gestionar los datos entre los controladores.

La aplicación *mobCar* se creó usando básicamente estas dos herramientas que nos proporciona AngularJS y se hizo siguiendo la siguiente metodología:

Usando las fases de un proyecto evolutivo, a la hora de implementar, se cogía un objetivo, se dividía en tantas tareas como fueran necesarias para completarlo, se creaban los controladores necesarios para poder completar estas tareas y finalmente se realizaban las pruebas pertinentes para comprobar el funcionamiento de dichos controladores. Los pasos serían los siguientes:

1. Requisitos: seleccionar un objetivo a completar y dividirlo en subobjetivos o tareas.
2. Diseño: diseñar uno o más controladores o/y factorías que cumplan con una o más de estas tareas.
3. Codificación: implementar dicho controlador, controladores o factorías usando AngularJS y Javascript
4. Testeo: probar si el controlador, los controladores o factorías completan los objetivos anteriormente seleccionados.

Vamos a poner como ejemplo cómo se completó uno de los objetivos siguiendo la metodología comentada anteriormente.

1. Se escogió el objetivo *Implementar un formulario que permita al usuario crear un viaje con toda la información necesaria.* Se extrajo una tarea que consistía en la creación de una función que recogiera los datos introducidos por el usuario y creara una llamada para enviar al servidor de Sigma y así crear un viaje.
2. Se diseñó un controlador llamado *crearViajeCtrl*, que se encargaría de recoger de la vista el origen del viaje, el destino, el día, la hora, las preferencias y los

márgenes. También se creo una factoría llamada *FactoriaDatos* que es la encargada de guardar todos los datos importantes de la aplicación para poder usarlos en cualquier controlador o vista.

3. Se implemento este controlador creando funciones internas como *crearViaje( ) o invertirViaje( )*, encargadas, por ese orden, de juntar los datos en una llamada y enviarla al servidor con el método *$http* [22] de AngularJS para crear un viaje y de invertir los datos de origen y destino al pulsar el botón adecuado. También se creó la factoría con una variable llamada *data* que es la que se usaría más adelante para guardar y cargar datos.
4. Se realizaron pruebas para asegurar el correcto funcionamiento del controlador y sus funciones internas y que este cumpliera los objetivos para los que había sido creado. Se realizaron a la vez pruebas para cerciorarse de que la factoría cargara y guardara datos correctamente.

Usando esta metodología se crearon todos y cada uno de los controladores de la aplicación así como sus funciones interiores y se probó unitariamente que cada una de las partes de la aplicación funcionara correctamente y posteriormente se probó el conjunto de la aplicación para asegurar el correcto funcionamiento una vez integradas todas las partes.

4.3.1 Problemas a destacar en la codificación

En este apartado se van a comentar los dos problemas más importantes que se encontraron durante la implementación.

El primer problema está relacionado con la búsqueda de viajes y fue costoso de identificar. Consistía en que los viajes que se creaban desde la aplicación web no se podían encontrar, ni buscándolos desde la misma aplicación, ni buscándolos desde las aplicaciones móviles de Android e iOS, pero si existían en la base de datos como viajes. Investigando y con la ayuda de los compañeros de Sigma se pudo encontrar la solución.

El problema que impedía la correcta búsqueda de viajes estaba relacionado con el *timestamp* que se enviaba para determinar el día del viaje. Existen varios formatos de *timestamp* pero un ejemplo podría ser el siguiente: Oct 29, 2010 5:40:23, que transformado en array de números quedaría así de la siguiente forma: 1288323623006. El error era que cuando se seleccionaba la fecha para crear un viaje se insertaba el timestamp con la hora a la que se había creado incluida y esto se transformaba en array para enviarlo al servidor, lo mismo ocurría al buscar un viaje, se buscaba con el timestamp junto con la hora a la que se había creado y por lo tanto al transformarlo en array de números estas fechas no coincidían nunca, ya que las horas eran diferentes

Una vez detectado se solucionó en el acto y por lo tanto ya se podían crear viajes con el formato correcto y encontrarlos en una búsqueda.

El segundo problema que obligó a modificar uno de los objetivos marcados.

El objetivo 7 decía que el sistema de geolocalicación se debía implementar con la API de GoogleMaps, extendiendo el objetivo se entiende que todo lo relacionado con la geolocalización debía implementarse con esta API, incluyendo también la geocodificación, que consiste en transformar coordenadas en direcciones reales.

El problema reside en que en este proyecto se usó la versión gratuita de esta API, la cual sólo permitía geolocalizar o geocodificar diez direcciones o coordenadas por segundo, al principio no supuso un problema, pero cuando la cantidad de viajes aumentó empezó a devolver errores que no se podían controlar ya que se superaba esa restricción.

En ese momento se tuvo tomar la decisión de usar otro servicio de localización. Se siguió usando Google Maps para la búsqueda de direcciones y la geolocalización, pero para la geocodificación se usó el la herramienta *Nominatim* de OpenStreetMaps [23],[24]. Este servicio de localización gratuito consiguió solucionar el problema ya que no tenía ninguna restricción en cuanto a peticiones por segundo, pero como inconvenientes no era tan preciso ni rápido como Google Maps lo cual provoca unos segundos de retraso a la hora de mostrar los viajes y sus direcciones.

4.4 Pruebas

Debido a la baja complejidad del código no se crearon test especiales para cada uno de los casos de la aplicación, si no que se probó su funcionamiento por el método de ensayo y error, es decir, se probaba un código, si funcionaba se daba por correcto pero si fallaba se probaba con una versión alternativa. Si se hubiera necesitado, se podría haber usado las herramientas de testing [25] que proporciona AngularJS. Este método permitió solucionar todos los errores conocidos actualmente de la aplicación en un tiempo comprendido entre 10 minutos y dos días, siendo dos días el máximo tiempo que se empleó a la hora de solucionar un error encontrado, principalmente debido a un error en la compatibilidad con el formato de datos que usa la empresa.

Aparte de los test realizados por el programador, se realizaron test con voluntarios, estos test consistieron en presentar la aplicación a 10 voluntarios y analizar su reacción.

Los test se hicieron a personas de entre 18 y 62 años, aunque los Stakeholders del proyecto serían personas de entre 18 y 26 años, edad media en la que se empieza y se termina una carrera universitaria, se escogieron algunos voluntarios de más avanzada edad y de menos experiencia con las nuevas tecnologías para probar la facilidad de uso de la aplicación.

A continuación se mostrará la metodología seguida para realizar estos test y los resultados obtenidos.

1. Se explicó al voluntario, que no había visto anteriormente la aplicación, en qué consistía la aplicación pero sin dar detalles de su funcionamiento.
2. Se proporcionó al voluntario unos orígenes y unos destino ya predefinidos y se le pidió lo siguiente:
    a. Que buscara un viaje y se uniera como pasajero.
    b. Que creara un viaje simple y otro periódico.
    c. Que aceptara y denegara peticiones de pasajeros de un viaje ya creado.
    d. Que anulara una petición suya a un viaje.

e. Que eliminara un viaje.
f. Que modificara el perfil de usuario.
g. Que cambiara el idioma de la aplicación.
h. Que saliera de la aplicación.

Durante este paso se fueron tomando apuntes de la respuesta del usuario ante las herramientas de la aplicación y de los errores que pudieran detectarse, pero sin comunicarse con el voluntario.

3. En este último paso se pidió al voluntario una opinión sobre la aplicación, se le pregunto sobre la facilidad de uso, sobre si en algún momento le había acostado entender algo o realizar alguna de las peticiones y sobre si había detectado algún error de funcionamiento o de diseño.

A continuación se van a mostrar algunos de los resultados obtenidos más importantes de estos test, la información que se muestra es: número de voluntario, edad, sexo, valoración del manejo de las nuevas tecnologías, la opinión que tuvo de la aplicación, los problemas que encontró en el diseño y finalmente los problemas que encontró en la implementación.

Voluntario 1

- Edad: 25
- Sexo: hombre
- Nuevas tecnologías: tiene título de Ingeniería informática

  Opinión: le pareció una aplicación fácil de usar aunque le costó ver algunos botones y encontró un error relacionado con la búsqueda de viajes.

  Diseño: se tuvo que ampliar el tamaño de los botones de buscar y crear viaje.

  Implementación: se encontró un error que no permitía encontrar viajes creados por la aplicación pero sí creados por la aplicación de Sigma en la que se basa el proyecto, se tuvieron que emplear dos días para solucionar el problema.

Voluntario 2

- Edad: 18
- Sexo: hombre
- Nuevas tecnologías: se defiende bastante bien con cualquier aplicación web o móvil

  Opinión: le da la impresión de que está diseñada para móvil pero le parece muy sencilla de usar, encontró un error en un viaje encontrado que se había creado anteriormente.

  Diseño: ningún cambio

  Implementación: al crear un viaje con hora 09:35, esta se guardaba como 935, y al devolverla el servidor como viaje buscado la devolvía como 935 y se mostraba como 93:50, se tuvo que añadir un 0 delante antes de mostrarla en la vista. No supuso cambios en la planificación y se solucionó en escasas horas.

Voluntario 3

- Edad: 62
- Sexo: mujer
- Nuevas tecnologías: poca habilidad con aplicaciones web y normal con aplicaciones móviles

  Opinión: le gustó el diseño ya que se muestra todo ordenado y con un tamaño fácil de ver, aun así le costó encontrar el botón para editar el perfil y le costó mostrar los viajes pasados.

  Diseño: se añadió color al botón de cambio de perfil y en cuanto al botón para ver los viajes pasados no se pudo hacer nada debido a las restricciones de diseño que se tienen al crear esta aplicación ya que se debe parecer a las originales.

  Implementación: ningún cambio.

Voluntario 4

- Edad: 59
- Sexo: hombre
- Nuevas tecnologías: se defiende en aplicaciones móviles y web

  Opinión: de nuevo, como al anterior voluntario, le gusto el tamaño de los elementos de la vista así como el diseño en general. Problemas para ver también el botón de editar perfil.

  Diseño: se añadió color al botón de cambio de perfil y en cuanto al botón para ver los viajes pasados no se pudo hacer nada debido a las restricciones de diseño que se tienen al crear esta aplicación ya que se debe parecer a las originales.

  Implementación: ningún cambio.

Voluntario 5

- Edad: 22
- Sexo: mujer
- Nuevas tecnologías: habilidad alta en el manejo de cualquier aplicación web o móvil

  Opinión: en el modo móvil veía confusa la disposición de la información al ver los viajes del usuario y encontró un problema al buscar una dirección.

  Diseño: se separó un poco la información que la voluntaria veía confusa y se enmarcó en rectángulos azules el día y la hora del viaje.

  Implementación: se tuvo que debugar la aplicación para encontrar el error que surgía cuando se buscaba viaje sin haber geo localizado antes, se solucionó de manera satisfactoria en un día de investigación.

La información proporcionada por estos test sirvió para solventar varios errores de diseño y de código, así como para mejorar el diseño de la aplicación.

5 LA APLICACIÓN

En este apartado se van a mostrar a modo de ejemplo alguna las pantallas más importantes e interesantes de la aplicación y se van a explicar las acciones que el usuario podría realizar en cada una de ellas. Se verán las pantallas en el tamaño de resolución de más de 1200 píxeles.

5.1 Pantalla principal/pantalla de búsqueda

La pantalla principal que se muestra nada más hacer login es la pantalla de buscar viaje, tal y como se puede ver en la Figura 7**.** En esta pantalla el usuario puede crear una búsqueda y enviarla al servidor. El origen o destino se puede insertar de dos formas diferentes, geo localizando o introduciendo una dirección y pulsando buscar. Se puede escoger como origen cualquier dirección y como destino una de las direcciones predefinidas de la universidad o viceversa si se pulsa el botón invertir viaje, mostrado con dos flechas que se apuntan entre si.

Una vez escogido el origen y el destino se deberán escoger la fecha deseada del viaje con el calendario desplegable y la hora deseada.

Finalmente se escogerán dos parámetros que corresponden a los márgenes, el primero será el margen de distancia máxima que podrá haber entre el origen del usuario y el origen del viaje creado por el conductor y el segundo, será el margen máximo que podrá haber entre la hora escogida por el usuario y la escogida por el creador del viaje.

Una vez rellenados todos los campos se deberá pulsar el botón *Buscar* y se enviará la búsqueda al servidor.

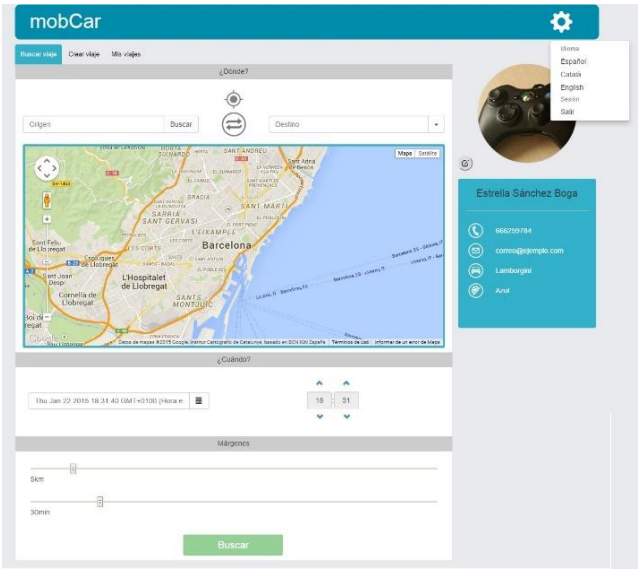

Figura 7. Pantalla que aparecerá al acceder a la aplicación, también se muestra en la esquina superior derecha el contenido del menú desplegable.

5.2 Pantalla resultado de la búsqueda

En esta pantalla, mostrada en la Figura 8, se mostrará el resultado de la búsqueda, en el caso de que no se encuentre ninguna coincidencia se mostrará un mensaje y se dará la posibilidad de realizar otra búsqueda.

Si la búsqueda es fructuosa, se mostrarán todos los viajes listados y el usuario podrá escoger el que desee para ver su información detallada. Una vez escogido un viaje, se deberá indicar cuantas plazas se desean y pulsar el botón *Confirmado*, de esta forma se enviará una petición con tantas plazas de un viaje como se hayan escogido. En la Figura 8 se puede ver un caso en el que se ha desplegado la información de un viaje y se han escogido 3 plazas.

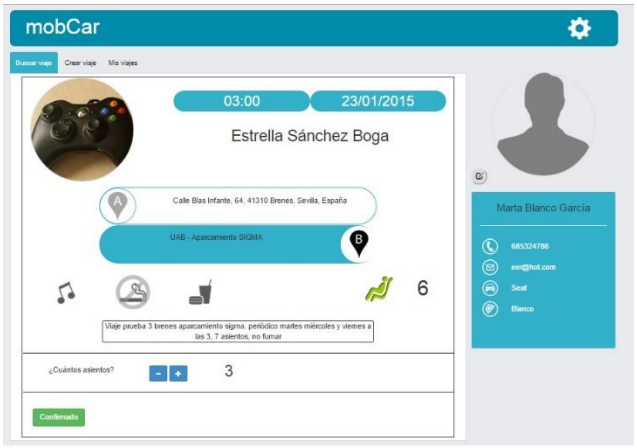

Figura 8. Respuesta del servidor donde se muestra un viaje resultado de una búsqueda, desplegado para ver toda su información.

5.3 Mis viajes

En esta pantalla se mostrarán todos los viajes del usuario ya sean viajes en los que es pasajero o conductor, también se mostrarán los viajes periódicos creados por el usuario y si se pulsa el texto "*Visualiza todos los viajes pasados...*" situado en la parte inferior de la pantalla, se podrá ver el historial de viajes del usuario.

En los viajes creados por el usuario se podrán ver las peticiones aceptadas, pendientes y rechazadas de los pasajeros, se podrá ver la información de cada uno de ellos y en el caso de que exista una petición pendiente se podrá aceptar o rechazar mediante dos botones. El usuario también podría eliminar el viaje, lo cual implicaría eliminar todas las peticiones de los pasajeros de dicho viaje. En el caso de que fueras pasajero y eliminaras el viaje solamente eliminarías tu petición a dicho viaje.

En la Figura 9 se muestra un viaje simple extendido con varios usuarios pasajeros.

En la resolución de pantalla de los ejemplos, el usuario podría editar el perfil en cualquiera de las pantallas anteriores, simplemente tendría que pulsar el botón que hay encima de la información, cambiar la información deseada y pulsar el botón *Guardar* que aparecería bajo los datos.

El perfil en resoluciones media, pequeña y muy pequeña aparecería en una pestaña a la derecha de "Mis viajes".

En cualquier pantalla el usuario también podrá acceder al menú pulsando el botón en forma de engranaje situado en la derecha de la cabecera tal y como se ve en la Figura 7, con este menú podrá salir de la aplicación o cambiar de idioma al castellano, catalán o inglés.

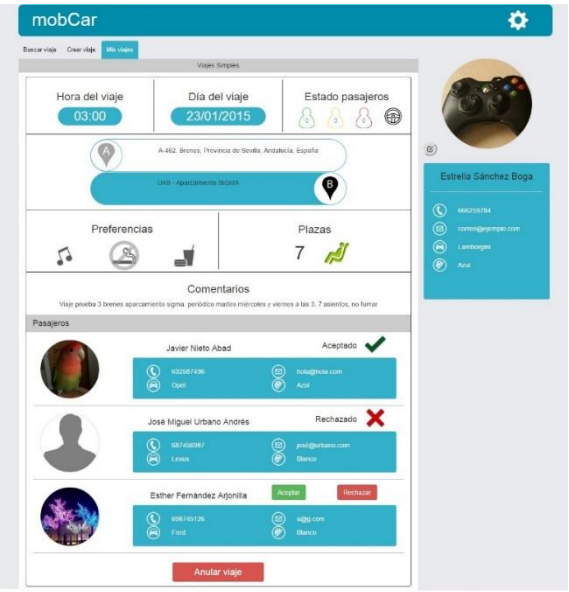

Figura 9. Viaje en el que el usuario es conductor y en el que se pueden ver tres pasajeros, uno aceptado, otro rechazado

6 Resultado

El resultado del proyecto ha sido una aplicación web programada en HTML5, CSS, Javascript y el framework AngularJS que te permite crear o buscar viajes cuyo origen o destino sea cualquier punto de la ciudad hasta la universidad en la que estudies. Está implementada de tal forma que te permite acceder desde cualquier dispositivo sin tener que preocuparte por la interfaz, ya que esta se adapta a cualquier tamaño de pantalla cambiando su diseño para sacarle el máximo rendimiento. Gracias a *angular-translate* [26] esta aplicación está totalmente traducida al catalán y al inglés.

La aplicación actualmente se encuentra en su versión 1.0, lo que significa que dispone de todas las funcionalidades necesarias para que el usuario pueda realizar todas las acciones que se programaron al inicio del proyecto. Esto significa que se han conseguido alcanzar todos los objetivos marcados al inicio del proyecto, gracias a la correcta planificación que se realizó y a las metodologías utilizadas para realizar las tareas.

Por lo tanto, la aplicación ya estaría lista para su lanzamiento al público, para así poder empezar con la fase de testeo con usuarios reales que permitiese una retroalimentación para la mejora de la aplicación.

7 Ampliaciones y mejoras

Como a cualquier aplicación web del mercado, a *mobCar* se le podrían añadir muchas más funcionalidades, estas funcionalidades podrían facilitar la comunicación conductor-pasajero, mejorar la experiencia de uso de la aplicación o mejorar su eficiencia.

Este apartado se dividirá en dos subapartados, mejoras y nuevas implementaciones, en los que se comentarán algunos posibles añadidos que mejorarían substancialmente la aplicación.

7.1 Mejoras

Creación de una sesión para almacenar los datos de la aplicación en aras de que no se deban descargar cada vez que se abre la aplicación: Actualmente la aplicación no guarda ni un solo dato en el dispositivo donde se ejecuta, es decir, no existen sesiones, además si se refresca la página te retorna a la pantalla del login.

Se pensó de esta forma con el objetivo de minimizar el tamaño de la aplicación en el dispositivo, concretamente si se ejecuta en un dispositivo móvil, pero tal vez para mayor comodidad y mayor velocidad de carga se podría mejorar creando una sesión en la que los datos se descargaran sólo una vez y se eliminaran solo al cerrar sesión.

Poder modificar todos los datos del perfil desde la aplicación web: La aplicación, a día de hoy solo te permite modificar ciertos datos del perfil. Se intentó introducir la posibilidad de modificar la imagen de perfil pero por problemas de compatibilidad de formatos con el servidor no fue viable.

7.2 Nuevas implementaciones

Implementación de notificaciones push para alertar a los usuarios de cualquier cambio en uno de sus viajes (nueva petición, baja de petición, cancelación de viaje, etc): La implementación de notificaciones push se descartó en la planificación del proyecto debido a que se necesita información de la parte back-end de la aplicación de Sigma para poder crearlas, además, ya que la aplicación se ejecuta en navegador, se podría abrir desde cualquier dispositivo móvil y por el momento no se sabría controlar el sistema de notificaciones de cada sistema operativo.

Una gran mejora de la aplicación sería precisamente implementar este tipo de notificaciones, para ello habría que acceder al código interno del servidor de la aplicación y una vez teniendo dicha información, estudiar los sistemas de notificaciones de los dispositivos más usados para así poder implementar las notificaciones para el mayor número de dispositivos posible. También se podrían implementar alarmas, que escogida la anterioridad por el usuario, avisara de que se acerca el día de un viaje.

Chat interno: Se podría implementar un chat interno que, una vez creado un viaje, permitiera a todos los pasajeros aceptados intercambiar información o simplemente conocerse antes del día del viaje. Esta nueva funcionalidad se crearía una vez acabada la implementación de las notificaciones push, ya que requeriría de las mismas para su funcionamiento.

Sistema de reputación mediante la puntuación del conductor y los pasajeros: Sería muy interesante poder reflejar la experiencia de los viajes realizados y poder ver la opinión que puedan tener los conductores de sus pasajeros o viceversa para poder decidir a la hora de escoger viaje. Por lo tanto se podría implementar un sistema de votaciones que se mostraría una vez finalizado un viaje, que permitiría a los pasajeros dar al conductor una puntuación del 1 al 10 en uno o diferentes aspectos (conversación, seguridad en la conducción, comodidad, puntualidad, etc) y a la vez que un conductor pudiera puntuar a los pasajeros de su viaje. Esto permitiría tener algo de información a priori

a la hora de escoger un viaje ya que tendrías una puntuación real de cada conductor y cada pasajero que te ayudaría a escoger de manera más segura un trayecto.

8 Conclusiones

Una vez finalizado el proyecto, se podría comprobar que se han cumplido con todos y cada uno de los objetivos que se marcaron al principio, se ha conseguido seguir la planificación inicial sin tener que modificar sustancialmente la duración de las tareas y hemos llegado a obtener una aplicación que cumple con todos los requisitos deseados.

También es cierto que se podrían haber obtenido mejores resultados aplicando algunos de los puntos del anterior apartado o utilizando una metodología más eficiente, pero al principio del proyecto se definieron unos objetivos acorde a los recursos de los que se disponía, tanto materiales como de tiempo, para que se pudiera obtener un proyecto acabado con un número de funcionalidades suficiente para poder tener una aplicación en su versión 1.0. Por lo tanto, una vez obtenida dicha aplicación con todas las funcionalidades principales disponibles ya se podría empezar a mejorar e ir añadiendo todas las ampliaciones que fuesen necesarias.

A simple vista, podría decirse que diseñar e implementar una aplicación web es algo sencillo, pero nada más lejos de la realidad, es un trabajo largo y que requiere mucha dedicación, además, si un solo programador se debe de encargar de todas las partes del proyecto, la dificultad se multiplica. No obstante, este proyecto me ha permitido realizar un estudio e investigación sobre los frameworks AngularJS y Bootstrap y de esta manera he ampliado mis conocimientos en el desarrollo de aplicaciones front-end. He aprendido a diseñar e implementar una aplicación totalmente adaptable a cualquier dispositivo y que esta funcione de una manera transparente al usuario, rápida y eficientemente

A su vez, he aprendido la importancia de tener unos objetivos claros y definidos que te permitan diseñar una planificación estable para saber en todo momento las tareas que debes realizar, las dependencias y los resultados que se deben obtener, asó como la necesidad de establecer canales de comunicación entre los diferentes actores de los proyectos.

Y quizás lo más importante es que este proyecto me ha dado la oportunidad de acceder a una empresa y a trabajar en un equipo de desarrollo de software real, lo cual me ha permitido dar un pequeño vistazo al mundo laboral que me aguarda en el futuro. Es cierto que todo el trabajo se ha realizado en solitario, pero los compañeros de equipo me ayudaron con su amplia experiencia a aprender más rápidamente y a solucionar algunos errores del proyecto.

Agradecimientos

Me gustaría agradecer a la familia y amigos por haberme distraído y alimentado para que no muriera de hambre durante el desarrollo del proyecto y a su apoyo incondicional.

A los diez voluntarios que se ofrecieron a perder una tarde para ayudarme a mejorar la aplicación y me dieron su opinión objetiva sobre ella.

A Sergi Almazán del equipo Sigma mobile por ayudarme en todo lo que pudo, a Joan Lasierra, Joan Busquiel y Teresa Gómez por ofrecerme realizar este proyecto y hacerme un hueco en su empresa.

Y finalmente a Jordi Pons y a mi tutor Ramon Baldrich por ayudarme a poder hacer este proyecto y a guiarme en la elaboración de los documentos.

Bibliografia

[1] Sigma 2014, accessed 17 October 2014, <http://www.gestionuniversitariasigma.com/index.php/es/>

[2] Google 2014, *AngularJS API Docs,* AngularJS, accessed 17 October 2014, <https://angularjs.org/>

[3] Ari Lerner 2012, *ng-book The Complete Book on AngularJS,* ISBN 978-0-9913446-0-4

[4] Google, 2014, *API de Google Maps,* Google Developers, accessed 17 October 2014, < https://developers.google.com/maps/?hl=es>

[5] BlaBlaCar, 2014, accessed 30 October 2014, <http://www.blablacar.es/>

[6] Marta Benayas 2012, *Compartir coche para ir a la Universidad en Madrid* , Madrideducacion.es, accessed 6 December 2014, <http://madrideducacion.es/blog/2012/03/compartir-coche-para-ir-a-la-universidad-en-madrid/>

[7] BlaBlaCar, *Campus,* accessed 5 February 2015, <http://campus.blablacar.es/>

[8] Amovens 2014, accessed 5 November 2014, <https://www.amovens.com/es/>

[9] Carpooling 2014, accessed 6 November 2014, <http://www.carpooling.es/>

[10] Compartocoche.com, accessed 6 November 2014, <http://www.compartocoche.com/rutas/buscar>

[11] Comunidad para compartir coche en España 2014, ¿viajamosjuntos.com?, accessed 6 November 2014, <http://www.viajamosjuntos.com/>

[12] Carpling 2014, accessed 9 November 2014, <https://www.carpling.com/es/car/compartir-coche>

[13] Universidades: Compartir coche con otr@s estudiantes o profesores de tu universidad 2014, Carpling, accessed 10 November 2014, <https://www.carpling.com/es/info/universities>

[14] Ir a la universidad 2014, xarxaeco.compartir.org, accessed 2 November 2014, <http://xarxaeco.compartir.org/viajes/universidades/>

[15] Bootstrap 2014, accessed 20 October 2014, <http://getbootstrap.com/>

[16] Ari Lerner, *Mobile Apps*, e-book, ISBN 978-0-9913446-0-4, ng-book pp. 458-481 (2012)

[17] Bootstrap 2014, *Grid system*, CSS, accessed 4 October 2014 <http://getbootstrap.com/css/#responsive-utilities>

[18] Bootstrap, 2014, Responsive utilities, CSS, accessed 4 October 2014 <http://getbootstrap.com/css/#responsive-utilities>

[19] Ari Lerner, *Data Binding and Your First AngularJS Web Application*, e-book, ng-book pp. 10-17 (2012)

[20] Ari Lerner, *Controllers*, e-book, ISBN 978-0-9913446-0-4, ng-book pp. 25-30 (2012)

[21] Ari Lerner, *factory()*, e-book, ISBN 978-0-9913446-0-4, ng-book pp. 164 (2012)

[22] Ari Lerner, *XHR in Practice*, e-book, ISBN 978-0-9913446-0-4, ng-book pp. 211-225 (2012)

[23] OpenStreetMap 2014, accessed 29 December 2014, <https://nominatim.openstreetmap.org/>

[24] Wiki, *Nominatim,* OpenStreetMap, accessed 29 Dedember 2014, <http://wiki.openstreetmap.org/wiki/Nominatim>

[25] Ari Lerner, *Testing*, e-book, ISBN 978-0-9913446-0-4, ng-book pp. 295-372 (2012)

[26] Ari Lerner, *Localization*, e-book, ISBN 978-0-9913446-0-4, ng-book pp. 482-497 (2012)