Planificador colaborativo de viajes

Plan de trabajo

Autor: Fernando Albar Pérez
Dirigido por: Ignasi Lorente Puchades

Trabajo fin de grado
Grado Multimedia
Ingeniería Web
Curso 2015-2016

Índice

Índice de figuras

1. Descripción del proyecto

1.1 Descripción

En la actualidad, la utilización de medios en linea para recabar información y adquisición de viajes se ha vuelto mayoritaria en nuestro entorno[1], es por ello que son numerosas las aplicaciones informáticas desarrolladas para estas tareas. Pese a esto, creemos que aún es posible el desarrollo y explotación de nuevas herramientas enfocadas a facilitar todavía más la preparación de viajes.

El proyecto que se presenta es, pues, un intento de ayudar en el proceso de selección y planificación de excursiones y viajes. Para ello, haciendo uso de las herramientas cartográficas que proporciona Google se crea una aplicación web en la que los usuarios pueden, valiéndose de las opiniones de otros, planificar los destinos que quieren visitar seleccionando el destino y el tiempo que permanecerán en él.

La herramienta hace uso de la API (*Application Programming Interface*, Interfaz de programación de aplicaciones) de Google Maps para representar la información cartográfica de los destinos, una vez seleccionado éste y el periodo de tiempo que durará la visita, la aplicación se encargará de ofrecer una planificación de los destinos a visitar según la duración y valoración de los mismos efectuada por otros usuarios. Esto hace de ella una herramienta social y colaborativa, en la que los utilizadores pueden valorar y comentar los destinos turísticos.

De este modo, se le brinda a los usuarios la opción de votar y clasificar los destinos y lugares por los que ha pasado, obtener información sobre otros que han sido comentados por los demás y, en última instancia – y éste es el elemento diferenciador del software - disponer de un recorrido automatizado de los mejores sitios que visitar durante su estancia.

1.2 Objetivos

1.2.1 Principales

- Desarrollar una aplicación web completamente funcional para la planificación colaborativa de viajes que pueda ser utilizada en distintas plataformas. Ésta ha de posibilitar elección de viajes y excursiones entre grupos de usuarios y, a su vez, establecer las rutas óptimas para

1 VVAA (2012) *Usos, actitudes y tendencias del consumidor digital en la compra y consumo de viajes* Observatorio Digital IAB Spain [en línea] http://www.segittur.es/opencms/export/sites/segitur/.content/galerias/descargas/documentos/Hot_Topic_Viajes_IAB_abril_20122.pdf

las visitas seleccionadas según los votos de los participantes.

- Conocer cómo se emplea una plataforma de desarrollo basada en la nube para la ejecución de aplicaciones web y analizar funcionalidades, ventajas e inconvenientes de las PaaS

1.2.2 Secundarios

- Aprender a gestionar un proyecto TIC.
- Conocer el funcionamiento y posibilidades del lenguaje JavaScript del lado del servidor y los *frameworks* que existen para tal fin.
- Conocer el entorno de desarrollo MEAN.
- Adquirir conocimientos sobre el funcionamiento e implementación de bases de datos NoSQL.
- Comprender el funcionamiento de la API de Google Maps y sus herramientas de cartografía digital.
- Aplicar los estándares de marcado HTML5 y CSS3.
- Conocer los distintos algoritmos de búsqueda de información.
- Analizar el uso y funciones de los metadatos.
- Comprender el funcionamiento de las redes sociales y demás proyectos colaborativos.

1.3 Alcance del proyecto

- Representación cartográfica de una base de datos.
- Desarrollo de un algoritmo de selección y distribución de lugares adecuado a las necesidades del proyecto.
- Implementación eficiente de algoritmos de búsqueda.
- Creación de una aplicación social colaborativa y de las estructuras necesarias y medios para su funcionamiento.

- Diseño y desarrollo de una base de datos no relacional.
- Adaptación y uso de las herramientas proporcionadas por la API de Google Maps.
- Aplicación de las tecnologías basadas en el uso de JavaScript en el lado del servidor.
- Redacción de una adecuada Memoria de proyecto.
- Relación de una presentación virtual.
- Participación del debate virtual.

2. Plataforma de desarrollo

Para la creación del proyecto se recurrirá principalmente al siguiente equipo y herramientas informáticas:

2.1 Hardware

Apple iMac 27” 3,06 GHz Core 2 Duo
Memoria RAM 12 GB 1067 MHz DDR3
Tarjeta gráfica ATI Radeon HD 4670 256 MB
Conexión internet de fibra óptica 300/30 Mb

2.2 Software

Sistema operativo Apple OS X 10.10.5
Suite ofimática LibreOffice Vanilla 5.1.0.2
Entorno de desarrollo Visual Studio Code 0.8.0
Editor de texto Textwrangler 4.5.12
Framework node.js 0.10.32
Bases de datos MongoDB 2.6
Gestor de bases de datos MongoHub 2.3.2
Gestor de bases de datos Robomongo 0.8.4
Framework Express 4.9.0
API Google Maps 3
Herramienta para la elaboración de diagramas de Gantt GanttProject 2.6.6.
Retoque fotográfico Adobe Photoshop CS6
Software de dibujo vectorial Adobe Illustrator CS6
Montaje de vídeo Adobe After Effects CS6
Creación de wireframes Justinmind Prototyper Pro Edition Ver. 6.2.0
Diagramas UML MagicDraw 17.0.5
Grabación de pantalla y audio QuickTime Player 10.4

Además, con el fin de realizar pruebas sobre el correcto funcionamiento de la aplicación web, se utilizarán otros equipos de prueba.

3 Planificación

3.1 Actividades

Listado de actividades según el calendario propuesto para la confección del proyecto. Se divide en cinco bloques principales: 3 PACs (*prova d'avaluació contínua*), una entrega final y un debate virtual.

3.1.1 PAC1

Creación Plan trabajo			
Inicio: 17/09/15	**Fin:** 30/09/15	**Horas:** 36	**Hojas:**
Elaboración del documento que organiza y define el proyecto.			

Entrega PAC1			
Inicio: 30/09/15	**Fin:** 30/09/15	**Horas:**	**Hojas:**
Entrega del Plan de trabajo.			

3.1.2 PAC2

Prototipos iniciales			
Inicio: 01/10/15	**Fin:** 02/10/14	**Horas:** 10	**Hojas:** 1/2
Creación de los primeros prototipos que definen la estructura visual y organización de la aplicación.			

***Wireframes* bajo nivel**			
Inicio: 03/10/15	**Fin:** 05/10/15	**Horas:** 15	**Hojas:** 1/2
Definición y estructuración clara del proyecto.			

***Wireframes* detallados**			
Inicio: 06/10/15	**Fin:** 07/10/15	**Horas:** 10	**Hojas:** 2
Evolución de los *wireframe*s anteriores: se definen todos los elementos visuales.			

Identidad gráfica			
Inicio: 08/10/15	**Fin:** 09/10/15	**Horas:** 10	**Hojas:** 1
Estudio y creación del diseño gráfico asociado al proyecto.			

Estudio usabilidad			
Inicio: 10/10/15	**Fin:** 11/10/15	**Horas:** 10	**Hojas:** 1
Análisis de la usabilidad, experiencia de usuario y accesibilidad.			

Prototipo final			
Inicio: 12/10/15	**Fin:** 13/10/15	**Horas:** 10	**Hojas:** 2
Creación de un prototipo final que reflejará el aspecto definitivo del sitio.			

Diseño UML y diagramas de uso			
Inicio: 14/10/15	**Fin:** 15/10/15	**Horas:** 10	**Hojas:** 2
Representación del dominio del proyecto mediante el lenguaje de modelado UML. Elaboración de los diagramas de uso.			

Estudio bases de datos NoSQL			
Inicio: 16/10/15	**Fin:** 17/10/15	**Horas:** 12	**Hojas:**
Análisis sobre la posibilidades de uso de las bases de datos NoSQL.			

Modelo de datos			
Inicio: 18/10/15	**Fin:** 19/10/15	**Horas:** 6	**Hojas:** 1
Creación de los modelos para la base de datos y estudio de los diferentes formatos de archivo.			

Entrega borrador PAC2			
Inicio: 21/10/15	**Fin:** 21/10/15	**Horas:**	**Hojas:**
Entrega de la versión preliminar de la PAC2.			

Correcciones PAC2			
Inicio: 22/10/14	**Fin:** 28/10/15	**Horas:** 35	**Hojas:**
Correcciones y añadidos según la retroalimentación del consultor.			

Entrega PAC2			
Inicio: 28/10/15	**Fin:** 28/10/15	**Horas:**	**Hojas:**
Entrega final de la PAC2.			

3.1.3 PAC3

Estudio MEAN stack			
Inicio: 27/10/15	**Fin:** 21/11/15	**Horas:** 72	**Hojas:**
Estudio y análisis del funcionamiento, estructura y posibilidades del *MEAN stack*: angularJS, express, mongoDB y node.js.			

Instalación node.js			
Inicio: 30/10/15	**Fin:** 30/10/15	**Horas:** 2	**Hojas:**
Instalación y puesta en funcionamiento del *framework* node.js.			

Estudio API Google Maps			
Inicio: 30/10/15	**Fin:** 31/10/15	**Horas:** 8	**Hojas:** 1
Análisis del funcionamiento y posibilidades de la API de Google Maps.			

Estudio algoritmos y motores de búsqueda			
Inicio: 01/11/14	**Fin:** 02/11/14	**Horas:** 10	**Hojas:** 2
Estudio de los distintos algoritmos y métodos de búsqueda que mejor se adapten a las necesidades del proyecto.			

Estudio MongoDB			
Inicio: 03/11/15	**Fin:** 05/11/15	**Horas:** 16	**Hojas:** 1
Análisis y estudio del gestor de bases de datos MongoDB y las posibilidades que brinda para el desarrollo de la aplicación.			

Instalación MongoDB			
Inicio: 05/11/15	**Fin:** 05/11/15	**Horas:** 2	**Hojas:**
Instalación y puesta en funcionamiento del estor de bases de datos MongoDB.			

Desarrollo web html			
Inicio: 06/11/15	**Fin:** 08/11/15	**Horas:** 15	**Hojas:** 2
Elaboración del sitio web siguiendo los estándares HTML5 y CSS3.			

Creación base de datos			
Inicio: 09/11/15	**Fin:** 09/11/15	**Horas:** 4	**Hojas:** 1
Creación definitiva de la base de datos según los modelos contemplados con anterioridad.			

Inserción de datos			
Inicio: 09/11/15	**Fin:** 09/11/15	**Horas:** 2	**Hojas:**
Inserción de los datos necesarios para las versiones preliminares de la aplicación.			

Desarrollo node.js			
Inicio: 10/11/15	**Fin:** 14/12/15	**Horas:** 125	**Hojas:** 4
Desarrollo de la aplicación web mediante el uso del *framework* node.js.			

Entrega borrador PAC3			
Inicio: 08/12/15	**Fin:** 08/12/15	**Horas:**	**Hojas:**
Entrega de la versión preliminar de la PAC3.			

Correcciones PAC3			
Inicio: 09/12/15	**Fin:** 15/12/15	**Horas:** 36	**Hojas:**
Correcciones y añadidos según la retroalimentación del consultor.			

Entrega PAC3			
Inicio: 15/12/15	**Fin:** 15/12/15	**Horas:**	**Hojas:**
Entrega final de la PAC2.			

3.1.4 Entrega final

Desarrollo HTML II			
Inicio: 01/12/15	**Fin:** 05/12/14	**Horas:** 25	**Hojas:**
Continuación del desarrollo del sitio web.			

Desarrollo MongoDB			
Inicio: 06/12/15	**Fin:** 09/12/14	**Horas: 1**5	**Hojas:**
Continuación y perfeccionamiento de la base de datos.			

Desarrollo node.js II			
Inicio: 07/12/15	**Fin:** 02/01/15	**Horas:** 125	**Hojas:**
Mejoras y correcciones en el desarrollo del motor de la aplicación web.			

Elaboración de la presentación virtual			
Inicio: 18/12/15	**Fin:** 23/12/15	**Horas:** 25	**Hojas:**
Creación del guión y grabación de la presentación visual del proyecto.			

Revisión, actualización y finalización de la Memoria			
Inicio: 23/12/16	**Fin:** 07/12/16	**Horas:** 42	**Hojas:**
Elaboración final de la Memoria del proyecto.			

Entrega borrador Memoria			
Inicio: 02/01/16	**Fin:** 02/01/16	**Horas:**	**Hojas:**
Entrega de la versión preliminar de la Memoria del Proyecto.			

Correcciones de la Memoria			
Inicio: 03/01/16	**Fin:** 08/01/16	**Horas:** 36	**Hojas:**
Correcciones y añadidos a la Memoria del proyecto según la retroalimentación última del consultor.			

Entrega de la Memoria			
Inicio: 08/01/16	**Fin:** 08/01/16	**Horas:**	**Hojas:**
Entrega de la versión definitiva de la Memoria del Proyecto.			

3.1.5 Debate virtual

Desarrollo del debate virtual			
Inicio: 18/01/16	**Fin:** 22/01/16	**Horas:** 30	**Hojas:**
Respuesta a las preguntas que serán formuladas por el Tribunal encargado de evaluar el proyecto.			

3.2 Hitos principales

Listado de las principales entregas en el transcurso de la elaboración del proyecto.

	Fecha
Entrega PAC 1	30/09/2015
Entrega borrador PAC 2	21/10/2015
Entrega PAC2	28/10/2015
Entrega borrador PAC3	08/12/2015
Entrega PAC3	15/12/2015
Entrega borrador Memoria	02/01/2016
Entrega Memoria	**08/01/2016**
Debate virtual	**22/01/2016**

3.3 Diagrama de Gantt

Propuesta de planificación del proyecto siguiendo la metodología del diagrama Gantt.

Se establece una media de cinco horas de trabajo para cada día, de lunes a sábado. En inicio, los domingos y días festivos no se asigna trabajo alguno.

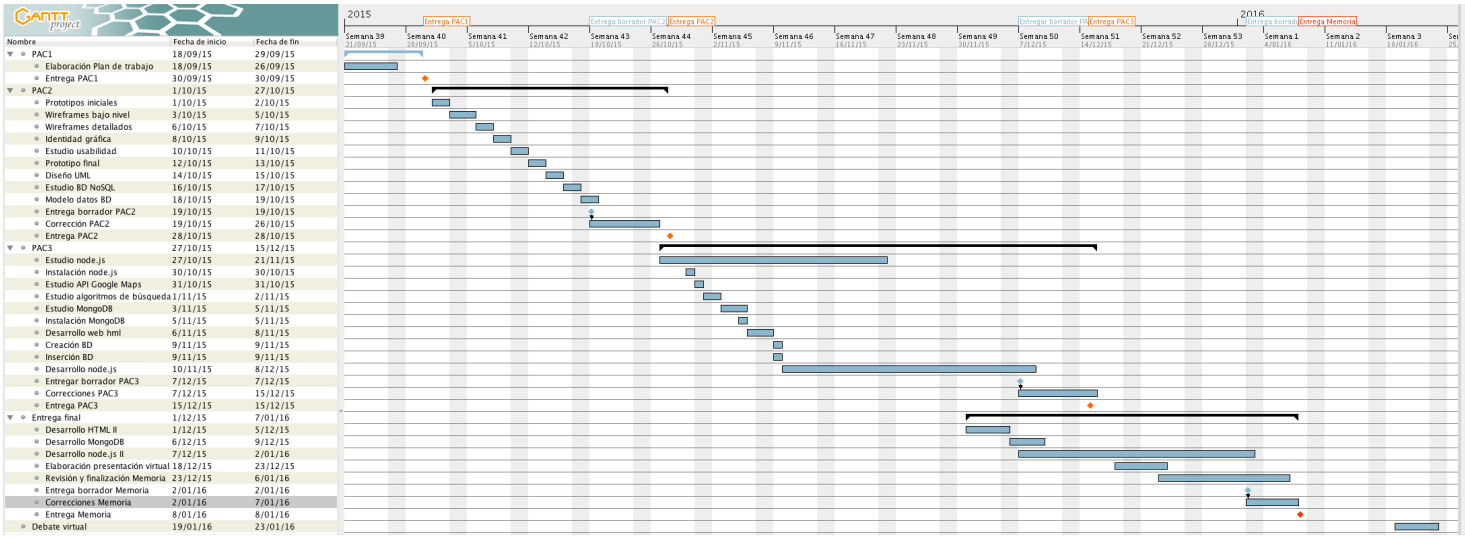

Figura 1. Diagrama de Gantt

4. Análisis de riesgos

Falta experiencia en el desarrollo de una aplicación web	
Riesgo: Alto	**Impacto:** Alto
Descripción: el desarrollo de una aplicación web de estas características es un proceso laborioso que, con toda seguridad, ocasionará problemas al estudiante.	
Mitigación: la elaboración de la planificación del proyecto ha tenido en cuenta este factor, dando la posibilidad de aumentar los plazos ante posibles contingencias. Además, puede recurrirse a tiempo que no estaría dedicado a él.	

Incorrecta planificación del proyecto	
Riesgo: Alto	**Impacto:** Medio
Descripción: debido a la poca experiencia en la planificación de un proyecto de esta envergadura, es posible que no se tengan en cuenta todos los elementos necesarios para su correcta elaboración.	
Mitigación: la estructura de tiempos es lo suficiente flexible para absorber los errores de planificación. También es posible disponer de más horas que, en un principio, no fueron reflejadas.	

Tecnologías poco maduras	
Riesgo: Medio	**Impacto:** Alto
Descripción: ante lo novedoso de algunas de las tecnologías propuestas, pueden suceder problemas derivados de su poco recorrido y/o implantación.	
Mitigación: realizar un correcto estudio y análisis de las mismas antes de su puesta en funcionamiento y recurrir a la ayuda de expertos en etapas lo más tempranas posibles.	

Enfermedad o incidencias familiares	
Riesgo: Medio	**Impacto:** Medio/Alto
Descripción: debido a la duración del proyecto no es posible anticipar este tipo de incidencias inevitables.	
Mitigación: si los acontecimientos no son de gravedad, es posible subsanarlos recurriendo a tiempo para asuntos que no estarían relacionados con el proyecto.	

Cambios en el API de Google	
Riesgo: Bajo	**Impacto:** Alto
Descripción: que la compañía que ha creado la API decida cerrarla o imponer condiciones que hagan imposible su uso.	
Mitigación: este sería un golpe muy importante pero, dado la pequeña envergadura de la aplicación y características propias del fabricante, es muy poco probable que esto sucediese. Aun así, existe la posibilidad de utilizar otras herramientas como OpenStreetMaps.	

4. Diagramas UML

4.1 Casos de uso

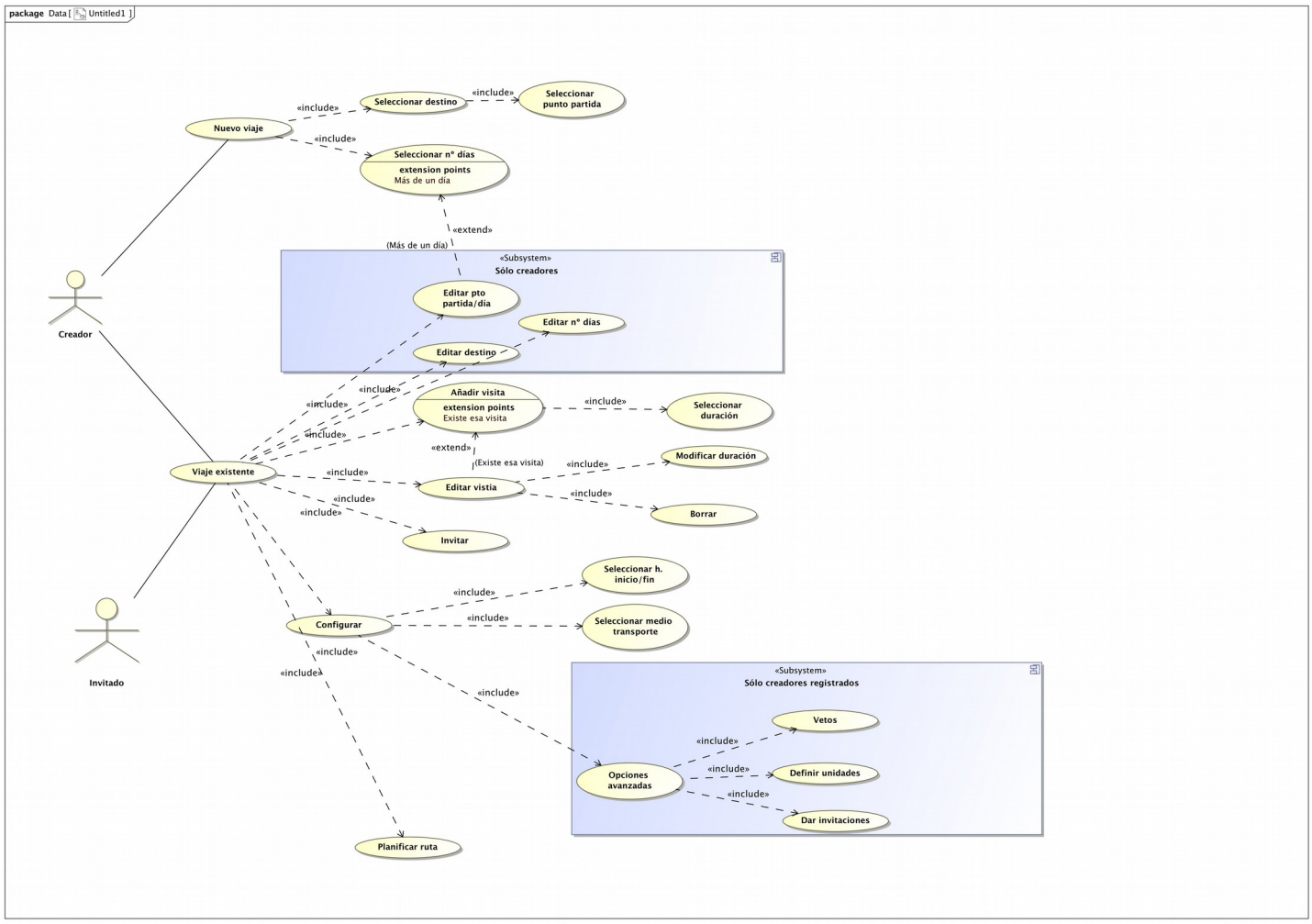

Figura 2. Casos de uso

Caso de uso: Crear un nuevo viaje.
Actor principal: usuario.
Ámbito: aplicación.
Nivel de objetivo: usuario.
***Stakeholders* e intereses**:
Usuario: quiere crear el viaje
Precondiciones: -
Garantías mínimas: -

Garantías en caso de éxito: el sistema mostrará al usuario una nueva página con los datos de su viaje.

Escenario principal de éxito:

1. El usuario introduce su destino.
2. El sistema valida el lugar mediante la Google Maps
3. El usuario introduce el número de días
4. El sistema crea una nueva entrada en la base de datos con la información de ese viaje.
5. El sistema muestra una nueva página con los datos del viaje y las opciones para compartir y modificar.

Extensiones:

3A. El usuario introduce un rango de fechas.

5A. El usuario desea modificar el destino y/o número de días

5A1. El usuario introduce un nuevo destino.

5A2. El usuario introduce un nuevo número de días o rango de fechas.

5A3. El sistema modifica la entrada de la base de datos.

5A4. El sistema modifica la página del viaje.

Caso de uso: Añadir nueva visita.

Actor principal: usuario.

Ámbito: aplicación.

Nivel de objetivo: usuario.

***Stakeholders* e intereses**:

Usuario principal: quiere crear una nueva visita.

Usuarios viaje: cuentan con una nueva opción a visitar

Precondiciones: Hay un viaje creado.

Garantías mínimas: El sistema guarda el intento de creación

Garantías en caso de éxito: Una nuevo lugar a visitar es añadido en el viaje

Escenario principal de éxito:

1. El usuario introduce un lugar a visitar.
2. El sistema valida el lugar mediante la Google Maps
3. El usuario introduce el tiempo de duración de la visita.
4. El sistema actualiza el registro de ese lugar.
5 El sistema actualiza la base de datos del viaje.
6. El sistema muestra la nueva visita en la página del viaje.

Extensiones:

3A. El usuario no introduce la duración de la visita.

3A1 El sistema selecciona la duración mediana en ese lugar o la establecida por defecto.

6A. El usuario desea modificar la duración o borrar la visita.

6A1 El usuario establece una nueva duración o elimina la visita.

6A2 El sistema actualiza la base de datos del viaje.

6A3 El sistema muestra la nueva visita en la página del viaje.

Caso de uso: Usuario registrado invita nuevo participante.

Actor principal: usuario registrado.

Ámbito: aplicación.

Nivel de objetivo: usuario.

***Stakeholders* e intereses**:

Usuario: quiere crear el mensaje.

Invitado: tiene la posibilidad de intervenir en el viaje.

Precondiciones: Hay un viaje creado por un usuario registrado.

Garantías mínimas: El sistema guarda el intento de invitación.

Garantías en caso de éxito: Un potencial usuario recibe una invitación para unirse al viaje.

Escenario principal de éxito:

1. El usuario introduce el correo electrónico o cuenta en una red social del invitado.
2. El sistema guarda un registro con los datos de las invitaciones.
3. El sistema actualiza la base de datos del viaje.
4. El sistema envía un correo electrónico con los datos del anfitrión, del viaje y el modo de unirse a él.

Extensiones:

5. Arquitectura de la web

La aplicación web está basada en lo que se ha denomido como *MEAN stack*, un conjunto de soluciones informáticas de software libre que comprende las siguientes aplicaciones: MongoDB (base de datos no relacional), ExpressJS (*framework* de servidor para Node.JS), AngularJS (framework MVC) y node.js (entorno de ejecución de Javascript del lado del servidor).

A continuación se hace un breve repaso de la arquitectura que da forma a la web:

5.1 Cliente

Entendemos por clientes los diversos usuarios que acceden al servicio. Esto se podrá realizar desde la inmensa mayoría de navegadores web actuales, tanto para ordenadores como en dispositivos portátiles. Sólo habrán de tener activado **JavaScrip**t, circunstancia que ocurre en la práctica totalidad de los casos.

5.2 Servidor

Para esta tarea se ha hecho uso de la plataforma **Node.JS** que permite utilizar el lenguaje de programación JavaScript en el lado del servidor. Con él podemos ocuparnos de todas las tareas del servidor, tales como enrutamiento o creación de páginas dinámicas.

Para agilizar y llevar a cabo las tareas de enrutamiento de un modo más eficiente y completo se ha recurrido al previamente citado *framework* **Express**.

En cuanto a la plataforma de hardware que albergue el sistema, debe analizarse según la previsión de clientes estimada. En un primer momento, y dado que no va a ser exigente se puede optar por alquilar espacio en un servidor o por alternativas en la nube como Amazon S3 o IBM Bluemix . En fases posteriores se habrá de acometer un estudio en profundidad sobre qué servidores utilizar.

Debido a que la infraestructura que pone la Universidad a disposición de sus alumnos para el alojamiento del proyecto no comprendía un caso tan poco usual como el estándar MEAN se ha optado por hacer uso de la **PaaS de IBM Bluemix**. Este servicio basado en la nube (*cloud computing*) brinda la posibilidad de construir, gestionar y ejecutar aplicaciones de diversos tipos y en diferentes entornos y lenguajes de programación.

5.3 Base de datos

En este apartado se ha optado por la utilización del modelo de bases de datos **NoSQL** del tipo orientado a documentos mediante el gestor **MongoDB** que crea ficheros de datos persistentes en formato JSON. El uso de estas tecnologías se decide por los siguientes criterios:

- Su capacidad para manejar altos volúmenes de datos y a facilidad de escalabilidad. Esta es una aplicación que se basa fundamentalmente en la utilización de grandes cantidades de datos que podrían llegar a lastrar su funcionamiento, por lo que es de vital importancia que su crecimiento sea sostenible.
- Las bases de datos pueden ser controladas mediante funciones Javascript, haciéndolo especialmente adecuado para una plataforma basada en node.JS.
- Está enfocado en la indexabilidad de los datos y cuenta, a diferencia de otros sistemas, con indices geoespaciales, lo que se manifiesta como una clara ventaja en un programa que hace un importante uso de datos.
- Su alta velocidad de lectura y escritura y la posibilidad de realizarlas en tiempo real, lo que concuerda con la filosofía de node.JS.

5.4 Arquitectura MVC

Para la confección del servicio web se ha empleado el patrón de arquitectura de software Modelo-Vista-Controlador mediante el *framework* desarrollado por Google **AngularJS**. El uso de este patrón se basa en seguir las mejores prácticas de la ingeniería del software: la reutilización del código y la separación de conceptos

Se ha optado por esta solución no sólo porque forme parte del estándar MEAN, si no por la versatilidad y comodidad que brinda este patrón a la hora de crear una aplicación de estas características. AngularJS está pensado para la elaboración de lo que se conoce aplicaciones de una sola página (*single-page applications*). Y, pese a que la presente no cumple esta característica en sentido estricto, ya que se crea una página con una url diferente para compartir con los contactos, creemos que esta pequeña variación no modifica en modo alguno la utilidad del patrón y el *framework*.

Podemos ver en nuestro desarrollo este patrón del siguiente modo:

- **Modelo**: la estructura de datos de la base de datos no relacional y los datos en formato JSON. Estos datos se extraen de los formularios cumplimentados por los usuarios.
- **Controlador:** este módulo principal se encarga de gestionar los datos de los viajes para crear los mapas, rutas y llamadas AJAX.
- **Vista**: se encarga del *front-end*, manejar los datos que se mostrarán a los usuarios

5.5 Anatomía de la aplicación

A continuación mostramos un breve esqueleto del sistema de archivos para intentar dar a conocer cómo se organiza la aplicación y los archivos que la conforman.

TFG-Falabar

- - server.js // Servidor de node.js

- - package.json // Dependencias para ejecución de npm

- - manifest.yml // Manifiesto para IBM Bluemix

Backend

- - - app

- - - - model.js // Modelo base de datos

- - - - routes.js // Enrutador

Front-end

- - - public

- - - - index.html // Indice de la web

- - - - travel.html // Página de viajes

- - - - js

- - - - - app.js // Archivo principal

- - - - - controller.js // Controlador

- - - - - mapService.js // Se encarga de los mapas

- - - - - page.js // Ordenamiento y paginación de datos

- - - - - routes.js // Crea las rutas de los viajes

- - - - - style.css

- - - - - style-page.css

- - - - images

Además de esto, contamos con la estructura de ficheros creados por los diferentes frameworks (en el directorio *node_modules*) y otros componentes necesarios como jQuery, Mongoose, Morgan (en el directorio *bower-components*).

5.6 Otro software presente

- **NPM:** Instalador de paquetes para entorno node.JS. Es el encargado de instalar los diferentes *frameworks* y programas afines
- **Bower**: Sistema de manejo de paquetes.
- **GIT:** sistema de control de versiones.
- **CF**: interfaz en línea para subir código a IBM Bluemix
- **Bootstrap**: framework para diseño de sitios web.
- **jQuery**: biblioteca de JavaScript
- **Mongoose**: cliente MongoDB para node.JS
- **Body-parser**: *middleware* para *parseo*.
- **Modernizr**: librería para homogeneizar HTML5 y CSS3 en distintos navegadores.
- **Angular-JS-geolocation**: permite geolocalización de usuarios para AngularJS

Ejemplo del archivo package.json que muestra las dependencias (software creado por terceros que añade funcionalidades) que son necesarias para la aplicación

```
{
    "name": "TFG_Falbar",
    "version": "0.4.5",
    "description": "TFG",
    "author": "Fernando Albar",
    "main": "server.js",
    "dependencies" : {
        "express"    : "~4.8",
        "mongoose"   : "~4.3.5",
        "morgan"     : "~1.6.1",
        "body-parser": "~1.5.2",
        "jsonwebtoken": "^5.5.0",
        "method-override": "~2.3.5"
    }
}
```

6. APIs utilizadas

En primer lugar hay que destacar que en el desarrollo de la aplicación se ha llevado a cabo de tal modo que se dote a la misma de su propia API. Esta será la encargada de gestionar los datos introducidos por los usuarios. Así, por ejemplo, con la ruta `/places` añadida a la url del viaje podemos, con el método GET, visualizar los lugares que se han añadido o, mediante POST, añadir un nuevo; podremos, también, con `/deletetravel` borrar el viaje.

La creación de esta API pretende facilitar la escalabilidad del proyecto y dotarlo de un lógica interna mejor definida. En un futuro habrá que dotarla de un sistema de autenticación más potente para evitar usos indebidos.

A continuación se explican algunas de las principales APIs externas empleadas en el desarrollo del proyecto.

6.1 Google Maps

La aplicación está basada en la **Google Maps API** en su versión 3. Con ella es posible hacer uso de toda la tecnología cartográfica creada por Google sin tener que ser nosotros quienes confeccionemos los mapas.

Facilita una serie de funcionalidades para el trabajo con mapas. A continuación se ennumeran las más utilizas:

- Geolocalización de direcciones. Tanto directa, introduciendo una dirección, como inversa, introduciendo la latitud y longitud.
- Presentación de mapas con varias capas de visualización diferentes.
- Cálculo de distancias entre puntos.

En el siguiente ejemplo podemos ver como se hace una llama al método de geolocalización que introduciendo el nombre de un lugar nos brinda las coordenadas de un punto.

```
function geocoding() {

var geocoder = new google.maps.Geocoder();

geocoder.geocode( { "address":  document.getElementById("destination").value },
```

```
function(results, status) {

    if (status == google.maps.GeocoderStatus.OK && results.length > 0) {

        var location = results[0].geometry.location,

            lat      = location.lat(),

            lng      = location.lng();

      document.getElementById("latitude").value=lat;

      document.getElementById("longitude").value=lng;}
```

6.2 Express

Esta API. en su versión 4.13.3 cuenta con innumerables métodos y *middleware* para el enrutamiento http/https bajo node.JS. Esto permite la elaboración de proyectos de un modo más simple y poderoso que si se desarrolla con los métodos nativos del lenguaje que complementa, llegándose a convertir en el estándar como *framework* de servidor.

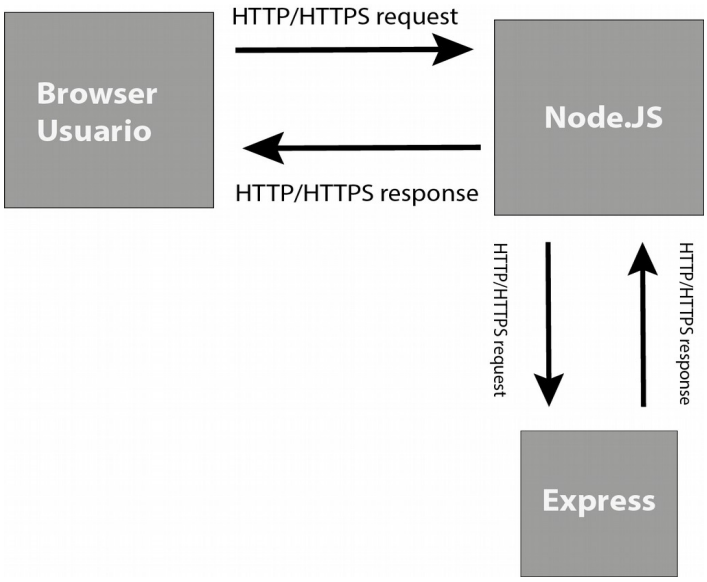

Figura 3. Flujo API de Express

Permite, además, crear la API de la aplicación basada en JSON y ofrecer capas para vistas de la web qua actúan a modo de plantilla cuando se redirecciona a determinados recursos. Veamos un par de ejemplos de uso simple:

```
app.get('/conocenos', function(request, response) {
```

```
response.sendFile('public/who.html');

});
```

Se redirige de un modo al archivo “who.html” a quienes acceden a la url /conocenos. De esta manera se separa enrutamiento y presentación.

```
$http.get(urlR + '/places').success(function(response){

            locations = convertToMapPoints(response)

            // Then initialize the map.

            initialize(latitude, longitude);

        }).error(function(){});

    };
```

De este modo la API de la aplicación obtiene mediante Express los datos JSON de cada viaje.

7. Prototipos

Para la elaboración del diseño de la aplicación se han creado, en una fase inicial, un conjunto de *wireframes* en baja fidalidad para ir definiendo la estructura y organización de contenidos. Posteriormente, en un proceso de iteración se han desarrollado en alta fidelidad para mostrar el resultado final del sitio.

7.1 *Wireframes* de baja fidelidad

Página inicial

Página principal del sistema a la que se accede desde la url principal. Muestra el contenido básico para crear un nuevo viaje, para registrarse en el sistema o darse de alta si ya se trata de un usuario registrado. Tanto el menú principal (bloque superior) como el *footer* permanacen iguales en todas las páginas para asegurar la coherencia visual y guiar al usuario. Siguiendo las convenciones de la web, el logotipo (zona superior izquierda) actúa también como enlace a esta página principal.

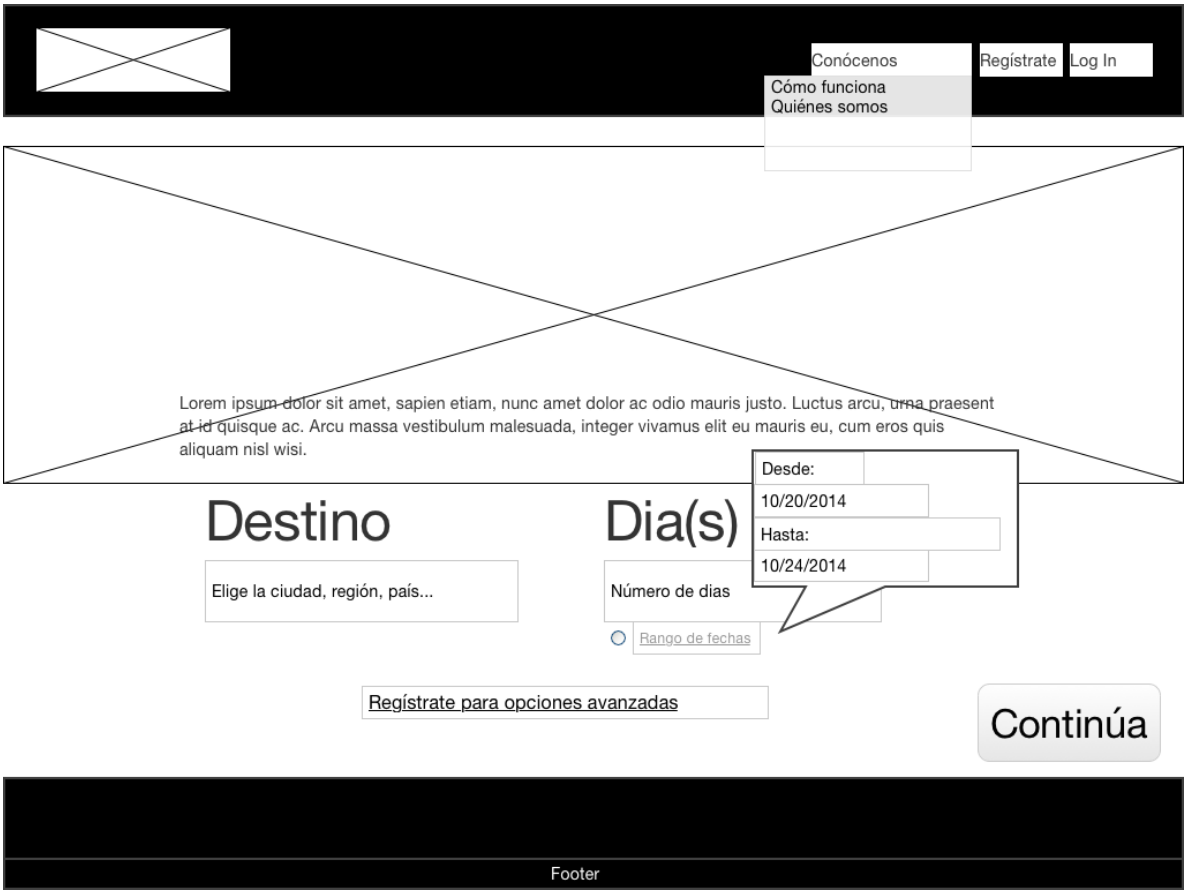

Figura 3. Prototipo baja fidelidad página inicial

Página de planificación para usuarios no registrados

Tras la creación de un nuevo viaje por usuarios que no están dados de alta, el sistema crea esta página para todo el proceso de planificación del viaje e invitaciones. Esta es, además, la página que reciben los invitados.

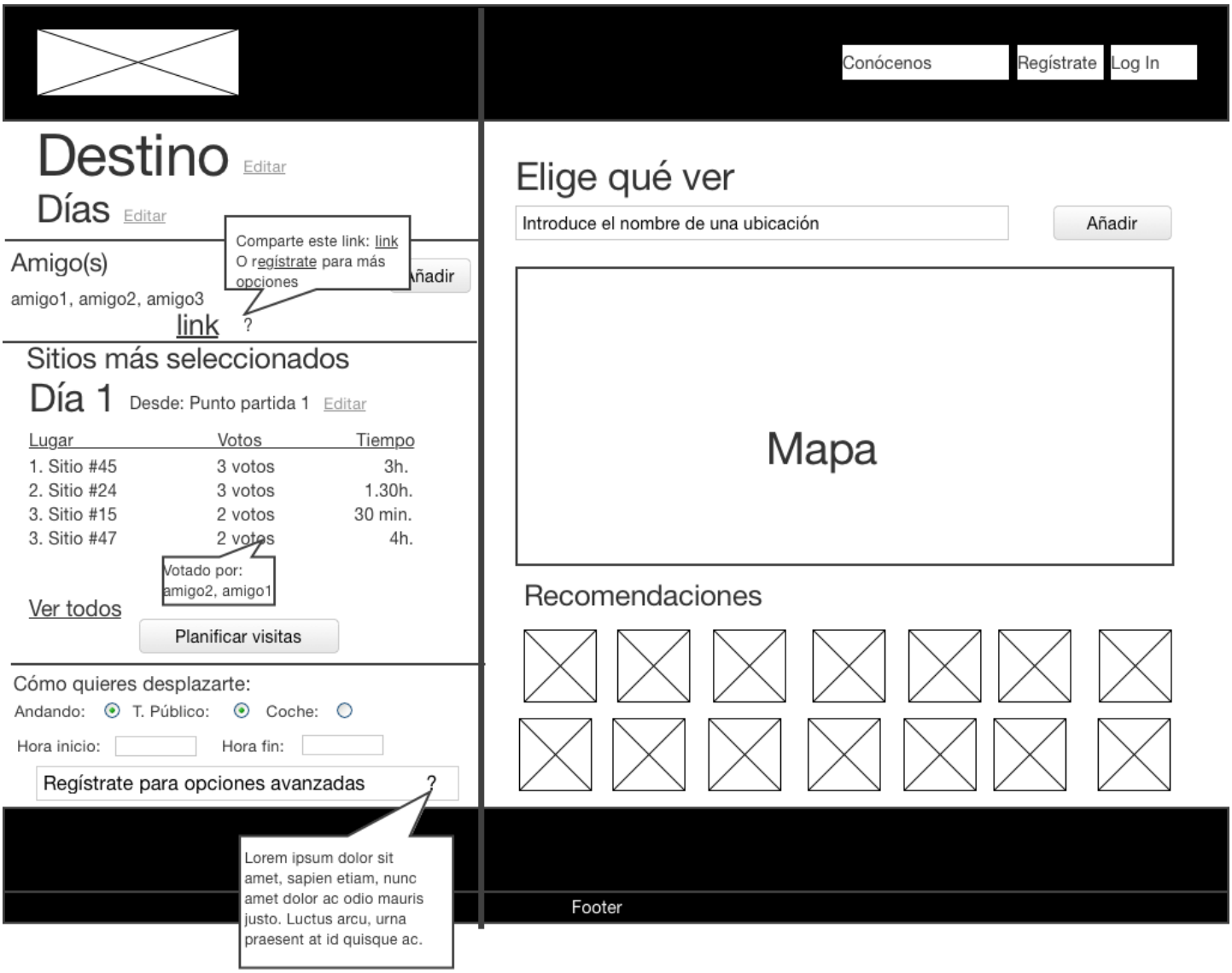

Figura 4. Prototipo baja fidelidad de página de planificación para usuarios no registrados

Página de planificación para usuarios registrados

Versión de la anterior pero para usuarios dados de alta. El número de opciones es más elevado y permite, por consiguiente, un uso más avanzado y completo.

Figura 6. Prototipo baja fidelidad de apartado de opciones

7.2 *Wireframes* de alta fidelidad

Figura 7. Prototipo alta fidelidad de la página inicial

Figura 8. Prototipo alta fidelidad de página de planificación para usuarios registrados

8. Perfiles de usuario

Los perfiles de usuario son una estrategia encaminada a conocer como los diferentes utilizadores interactuarán con el software, cuáles son las funcionalidades que demandan y como se habrán de implementar éstas para un uso óptimo. Para la elaboración de los mismos, se procedió a realizar entrevistas a potencial usuarios, de ellas se han intentado encontrar los patrones comunes a sus respuestas para, de este modo, crear una serie de personajes arquetípicos que ayuden a discernir lo que se espera del uso de la aplicación. Así pues, se ha llegado a las siguientes conclusiones:

En primer lugar, se han de destacar los dos tipos de usuarios que existen en la propia aplicación: *creadores de viajes* e i*nvitados*. Los creadores son los que deciden la aplicación a usar y ejecutan el proceso de confeccionar un nuevo viaje y administrar sus características. Es por ello, que son quienes desempeñan un papel más importante (y, en muchos casos, con un mayor perfil de conocimiento tecnológico) que los invitados. Estos adoptan un papel pasivo en la elección del producto, que se puede trasladar a su utilización; por ejemplo, sólo votando las propuestas sin aportar ninguna.

El nicho principal de usuarios lo componen **grupos de viajeros** que les unen lazos familiares o de amistad (aunque no habría que despreciar otros tipos como grupos virtuales que deseen organizar excursiones conjuntas) que están planificando acometer un viaje conjunto y son ellos mismo quienes desean planificar los procesos de un modo colaborativo.

Aunque no siendo su finalidad inicial, hay que considerar también los **viajeros individuales** (o que son los únicos encargados de la planificación) que desean organizar un viaje o excursión valiéndose de la capacidad del software para proporcionar sugerencias de lugares o rutas en base a la información proporcionada por las decisiones de otros utilizadores.

Por último, hay que tener en cuenta aquellos que se valen de la herramienta para analizar cuales son las **rutas más eficientes** entre varios destinos. Objetivo que puede ir más allá de la planificación de viajes turísticos y alcanzar otros sectores profesionales, como, por ejemplo, viajantes de comercio.

Por todo ello, se ha optado por plantear una aplicación enfocada en el uso por parte de grupos en los que media una relación personal, puesto que componen previsiblemente el sector mayoritario del público potencial. Pese a esto, en la medida de lo posible, y siempre que no dificulte el uso de estos, se intentará integrar a los otros perfiles analizados.

9. Escenarios de uso

9.1 *Scenario* 1

Breixo (28 años, estudios universitarios, conocimientos informáticos medios) después de su jornada laboral como profesor de bachillerato quiere planificar el viaje anual que lleva a cabo con sus amigos. Es conocedor de la zona a la que quieren desplazarse y de los días que disponen.

Su paso por la página podría definirse como accidental (le hablaron de ella de pasada en una charla sobre las vacaciones). Decide probarla; introduce los datos requeridos, sin registro, y va probando algunas de las visitas sugeridas por el sistema. Decide enviar el link a sus amigos mediante Whatsapp.

Días después, sigue utilizando el sitio y da el salto a registrarse con su cuenta de Facebook. Sus compañeros lo adoptan como "guía" informal para que sea quien adopte un papel más activo. A partir de aquí se muestra más interesado en conocer las diferentes opciones con que cuenta la página.

9.2 *Scenario* 2

Antía (37 años, nivel socioeconómico medio bajo, conocimientos tecnológicos bajos) recibe una invitación mediante la red social Facebook de un viaje que está organizando su amiga Helena. Lo acepta y empieza a informarse sobre qué lugares puede visitar, este proceso lo realiza sin saber muy bien cómo funciona la aplicación ni importarle demasiado. Sólo quiere que sus sitios preferidos sean votados por el mayor número de compañeros.

9.3 *Scenario* 3

Xavier (40 años, nivel socioeconómico medio , conocimientos tecnológicos medio-altos) trabaja como comercial en una empresa de neumáticos, por este motivo debe desplazarse por toda la Península para conocer distintos talleres. Aprovechando que tiene que pasar dos días en San Sebastián quiere conocer su gastronomía local y, especialmente, sus aclamados *pintxos*.

Haciendo uso de la aplicación busca algún recorrido de *pintxos* ya creado por un usuario y compartido públicamente. El trayecto que encuentra consta de 12 visitas a otros tantos bares y restaurantes que se llevan a cabo en un único día, por lo que Xavier decide modificarlo para

dividirlo en dos etapas y reducir el número de lugares a visitar en 3 el primer día y 4 el segundo. Posteriormente decide publicar su nuevo recorrido de manera pública para que otros usuarios puedan consultarlo.

9.4 Scenario 4

Carmo (31 años, nivel socioeconómico medio, conocimientos tecnológicos medios) es visitadora médica, por su trabajo de asistir a varios centros de salud al día y, en ocasiones, celebrar comidas de negocios con los facultativos.

Para intentar ahorrar tiempo, planificando las distintas siguiendo la ruta más corta posible, cada domingo por la tarde revisa su agenda y va apuntando en la página aquellos lugares que ha de visitar. Con la ayuda visual del mapa se hace una primera idea de dónde ir cada día, cuando tiene una primera aproximación ejecuta la función de crear una ruta para obtener, de este modo, una planificación semanal de su trabajo.

10. Usabilidad

Para la confección de los procesos interacción diseño de interfaces, interacción persona-ordenador, usabilidad y experiencia de usuario se ha seguido la metodología de desarrollo denominada ***Diseño centrado en el usuario*** **(DCU).** En ella se toma al usuario como centro en todos los procesos; la concepción del producto, su evaluación y desarrollo.

En este proceso de análisis se ha hecho uso de algunas de las herramientas utilizadas por el DCU. Principalmente el uso de prototipos (en alta y baja resolución) con el fin de identificar las reacciones y el modo de interactuar con el sistema por parte de un grupo de usuarios de evaluación. Además se han llevado a cabo entrevistas con ellos para evaluar qué esperan de una aplicación de estas características y si esto se ve reflejado en las pruebas a las que han sido sometidos. Este proceso no debe ser entendido como una fase inicial del proyecto, sino como un método que lo va guiando de principio a fin.

Por otra banda, se ha efectuado un proceso de investigación de aplicaciones similares, para esta tarea éstas se han analizado en profundidad para intentar discernir cuales eran sus fortalezas y sus debilidades y que elementos eran susceptibles de ser trasladados a este software y cuales requerían una reconfiguración.

Tanto las pruebas efectuadas con usuarios como el análisis de otras herramientas han arrojado que uno de los elementos más demandados y valorados son la sencillez de uso como la posibilidad de utilización lo más inmediata posible. Para cumplir los requisitos dictaminados en la fase de análisis se ha evitado que los usuarios tengan la obligación de registrarte en ella antes de poder operar. Además también se han prescindido de farragosos procesos de aprendizaje previos; sólo es necesario cumplimentar dos campos para comenzar.

De este modo, se intenta conseguir que una aplicación como ésta, que propone concepciones diferentes y, por lo tanto, susceptibles de una mayor complejidad, destaque por la simpleza de uso. Por esto, además de lo expuesto con anterioridad, se ha puesto especial atención en la utilización de terminología lo más común posible; un diseño visual poco recargado, que se presente al servicio del contenido y la funcionalidad, e inclusión de elementos que guíen a los usuarios en todo momento.

En una herramienta de este tipo, también se hace necesario incluir opciones específicas para usuarios más avanzado. Este hecho se ha intentado desarrollar de tal modo que no suponga un impedimento a los menos duchos, para ello se han confinando a lugares específicos que permiten que se pueda desenvolver una experiencia completa sin recurrir a e estas opciones pero, a su vez,

puedan ser utilizadas sin suponer un proceso de aprendizaje arduo.

11. Presupuesto

Para la confección del presupuesto se han englobado las tres áreas principales en la creación del producto (análisis, diseño y desarrollo) y se ha estipulado un precio por hora que se le asignará al único trabajador del proyecto.

El total bruto del mismo sería de 11.450 €. Resultado de calcular las jornadas de trabajo a 25€/h. e incrementar una tasa del 10% para cubrir posibles imprevistos.

A esto habría que añadir otros gastos como costos de dominio y alojamiento web o, si se diera el caso, el traslado a otras plataformas como teléfonos inteligentes.

Concepto	Precio
Análisis	**1.250€**
Diseño	**3350€**
• Usabilidad	1870€
• Diseño gráfico	1480€
Programación y desarrollo	**6.650€**
• Desarrollo web	3800€
• Programación de la aplicación	1890€
• Base de datos	960€
Alquiler servidor y dominio	95€/año
Total	11.540€/año

Figura 9. Presupuesto

12. Análisis de mercado

La aplicación propuesta se inscribe en el grupo de las dedicadas a la **organización y planificación de viajes turísticos**. En este sector se ha venido notando de manera clara la irrupción de las TICs en el modo en que planifican, organizan y compran los viajes y actividades relacionadas. Esta presencia de las nuevas tecnologías, que aún no ha llegado a su punto culminante, ha supuesto un cambio radical en la industria, hasta el punto de ir remplazando a las agencias de viaje como punto central para la organización y venta de viajes mediante herramientas telemáticas.

Como se puede deducir, éste es un sector maduro con una competencia establecida, aunque por las particularidades del nuevo mercado convergente, no deja de estar en un continuo proceso de cambio y transformación.

12.1 Principales competidores

Debido a la propia naturaleza de la aplicación, debemos hacer una distinción entre dos tipos claramente definidos de competidores: por un lado, aquellos destinado a la **planificación y confección de viajes y excursiones** y, por otro, aquellas herramientas enfocadas a la **votación y toma de decisiones en grupo**. Como se verá, son dos categorías diferenciadas que convergen en esta aplicación. Un análisis de ambas permiten arrojar luz sobre el campo en el que se ubica el proyecto.

El campo destinado a la planificación de viajes es el más amplio y con participantes destacados. Aquí encontramos toda una variedad de herramientas con concepciones y fines diversos: mapas y herramientas cartográficas, buscadores de lugares y/o establecimientos hosteleros, organizadores de viajes.

12.1.1 Mapas

- Google Maps: aunque es la herramientas de mapas y buscador en el que se basa la aplicación presentada, debemos considerarla también como una de los principales competidores
- www.openstreetmap.org
- www.guiarepsol.com
- www.viamichelin.es

- Bing Maps

Aunque con una concepción inicial diferente, también hay que señalar que los sistemas de GPS (tanto a nivel de software como en dispositivos físicos) son utilizados, en mayor o menor medida, para estas tareas.

12.1.2 Buscadores de lugares y/o establecimientos hosteleros

- www.touristeeye.com
- www.tripadvisor.es
- www.minube.com

12.1.3 Organizadores de viajes

- www.Tripbox.com
- www.evaneos.es
- www.way-away.es
- www.tripit.com

Hay que destacar que la definición de las categorías se ha llevado a cabo para facilitar el proceso de análisis pero muchas herramientas presentan características que permite englobarlas en diferentes áreas.

Por otra banda, hay una serie de herramientas concebidas para facilitar la toma de decisiones en grupo, mediante las votaciones de los usuarios que participan en ella. Estas son compatibles con la planificación de viajes en grupo; creando una lista de destinos y sometiéndolos al escrutinio de los miembros del grupo.

En esta labor destaca principalmente doodle.com, aunque hay competidores menores como Do´zz o Dudle, Polls o, incluso, el uso de herramientas de sondeo u hojas de cálculo.

	Fortalezas	Debilidades
Google Maps	• Posición destacada en mapas • Ecosistema Google	• Soluciones avanzadas a través de terceros
Tripadvisor	• Implantación en el mercado. • Cantidad de críticas	• Sólo servicios hosteleros • Ausencia planificación colaborativa
minube	• Interfaz y uso	• Uso colaborativo limitado
doodle	• Referencia fundamental en decisiones de grupos	• Demasiado genérico
tripbox	• Simplicidad • Uso coloborativo	• Sólo iOS • Escasa implantación

Figura 10. Análisis principales competidores

Visto lo cual, la presente aplicación debe ser integrarse en el territorio de convergencia entre los sistemas de soluciones de cartografía, las de planificación e información sobre viajes y las de toma de decisiones en grupo. De esta manera, es posible ofrecer respuestas en un campo todavía no cubierto de manera óptima por los competidores; que cubra el proceso integro de planificación de viajes.

14. Desarrollo

Con el fin de explicar el proceso de desarrollo de la aplicación se adjuntan extractos del código de la aplicación de aquellas partes que pueden ser de interés para el lector.

Routes.js: enrutador

Esta clase se encarga del enrutamiento, de la creación de páginas, tanto dinámicas (viajes) como estáticas (páginas de información, etc.) y de los métodos para obtener y modificar registros de la base de datos.

Las páginas de viajes se crean dinámicamente mediante el uso de la plantilla `travel.html` y los datos obtenidos de la base de datos por el controlador. Estos datos se presentan tanto en el mapa como en texto mediante el controlador `page.js`

```
// Dependencies
var mongoose        = require('mongoose');
var User            = require('./model.js');
var path            = require('path');
var url             = require('url');
var http            = require('http');

// Opens App Routes
module.exports = function(app) {

    // GET Routes
    // --------------------------------------------------------



/* Renders page for travels */
     app.get('/unTravel/:id', function(request, response) {
    response.sendFile('travel.html');

});

/* Renders page for help page */
     app.get('/ayuda', function(request, response) {
    response.sendFile('public/help.html');
```

```
});

/* Renders page for information page */
     app.get('/ayuda', function(request, response) {
    response.sendFile('public/quienes-somos.html');

});

/* Retrieves places content in DB  */

    app.get('/unTravel/:id/places', function(req, res){
        // Uses Mongoose schema to searching
        var url= req.params.id;
        var query = Travel.find({ travelID:  url}); //
        query.exec(function(err, users){
            if(err)
                res.send(err);

            // If no errors are found, it responds with a JSON of all places
in current travel
            res.json(users);
        });
    });



    // POST Routes
    // --------------------------------------------------------
    // Method for saving new places in DB
    app.post('/unTravel/:id/users', function(req, res){

        // Creates a new User based on the Mongoose schema and the post body
        var newplace = new Place(req.body);

        // New Place is saved in the db.
        newplace.save(function(err){
```

```
            if(err)
                res.send(err);

            // If no errors are found, it responds with a JSON of the new
place
            res.json(req.body);
        });
    });

    app.post('/unTravel/:id/users/deletetravel', function(req, res){

        // Removes current travel
        var url= req.params.id;
        var remove = Travel.remove({ travelID:  url}); //
        remove.exec(function(err, users){
            if(err)
                res.send(err);

        });
    });









};
```

Server.js: clase principal

Es la encargada de indicar que otros programas requiere la aplicación, conectarse con la base de datos, añadir métodos y configurar el puerto del servidor.

```
// Dependencies
// -----------------------------------------------------
var express         = require('express');
var mongoose        = require('mongoose');
var port            = process.env.PORT || 3000;
var morgan          = require('morgan');
var bodyParser      = require('body-parser');
var methodOverride  = require('method-override');
var app             = express();
var path            = require('path');

// Express Configuration
// -----------------------------------------------------
// Sets the connection to MongoDB
mongoose.connect("mongodb://localhost/pcvDB");

// Logging and Parsing
app.use(express.static(__dirname + '/public'));                 // sets the
static files location to public
app.use('/bower_components',  express.static(__dirname +
'/bower_components')); // Uses BowerComponents
app.use(morgan('dev'));                                         // Morgan
app.use(bodyParser.json());                                     // parsing
application/json
app.use(bodyParser.urlencoded({extended: true}));               // parsing
app.use(bodyParser.text());                                     // allows
bodyParser to look at raw text
app.use(bodyParser.json({ type: 'application/vnd.api+json'}));  // parse
application/vnd.api+json as json
app.use(methodOverride());

// Routes
// -----------------------------------------------------
require('./app/routes.js')(app);

// Listen
```

```
// ------------------------------------------------------
app.listen(port);
console.log('App listening on port ' + port);
```

model.js: Modelo de datos

Modelo de datos de MongoDB de los distintos sitios a visitar. Proporciona, además, las fechas de creación y modificación de los datos. Usa formato JSON.

```
// Dependencies
var mongoose    = require('mongoose');
var Schema      = mongoose.Schema;

// Creates a place Schema.
var placeSchema = new Schema({
    travelID: {type: String, required: true},
    username: {type: String},
    age: {type: Number},
    places: {
        stPoint: {type: Boolean, required: true},
        name: {type: String, required: true},
        location: {type: [Number], required: true}, // [Long, Lat]
        creator: {type: String, required: true},
        voted: {type: [String], required: true},
        votes: {type: Number, required: true},
    },

    created_at: {type: Date, default: Date.now},
    updated_at: {type: Date, default: Date.now}
});

// Sets the created_at parameter equal to the current time
PlaceSchema.pre('save', function(next){
    now = new Date();
    this.updated_at = now;
```

```
    if(!this.created_at) {
        this.created_at = now
    }
    next();
});

// Indexes this schema in 2dsphere format
UserSchema.index({location: '2dsphere'});

// Exports the UserSchema
module.exports = mongoose.model('travels', UserSchema);
```

Controller.js: controlador principal

Siguiendo el modelo MVC, es el controlador principal. Se muestran algunos extractos que ejemplarizan algunas de sus funciones.

```
 // Functions
    //
----------------------------------------------------------------------------
    // Get coordinates based on mouse click.
    $rootScope.$on("clicked", function(){

        // Run the gservice functions associated with identifying
coordinates
        $scope.$apply(function(){
            $scope.formData.latitude =
parseFloat(gservice.clickLat).toFixed(4);
            $scope.formData.longitude =
parseFloat(gservice.clickLong).toFixed(4);
```

```
        });
    });

    // Creates a new user based on the form fields
    $scope.createTravel = function() {

        // Grabs all of the text box fields
     var userData = {
            travelID: travelID,
            username: $scope.formData.username,
            age: $scope.formData.age,
            favlang: $scope.formData.favlang,
            places:  {
                stPoint: true,
                name: $scope.formData.favlang,
                location: [$scope.formData.longitude,
$scope.formData.latitude],
                creator: $scope.formData.username,
                voted: $scope.formData.username,
                votes: 1},

        };

        // Saves the travel data in DB
        $http.post('/unTravel/' + travelID+ '/users', userData)
            .success(function (data) {
```

```
            })
            .error(function (data) {
                console.log('Error: ' + data);
            });
    };

        $scope.createTravel = function() {

        // Grabs all of the text box fields
        var userData = {
            travelID: urlID,
            username: $scope.formData.username,
            age: $scope.formData.age,
            favlang: $scope.formData.favlang,
            places:  {
                stPoint: true,
                name: $scope.formData.favlang,
                location: [$scope.formData.longitude,
$scope.formData.latitude],
                creator: $scope.formData.username,
                voted: $scope.formData.username,
                votes: 1},

        };

      // Saves the user data to the db
```

```
        $http.post('/unTravel/:id/users', userData)
            .success(function (data) {


            })
            .error(function (data) {
                console.log('Error: ' + data);
            });
            window.location="/unTravel/"+urlID;
    };
```

15. Bugs

Conforme se iba avanzado en el desarrollo del sitio, han sido descubiertos fallos que comprometían el correcto funcionamiento. Indicamos algunos a modo de ejemplo.

Bug: Los nuevos destino no se actualizaban en tiempo real.

Cada vez que un nuevo destino era añadido a un mapa no se mostraba hasta que la página no era recargada manualmente por el usuario.

Se ha podido subsanar valiéndose de los métodos de carga AngularJS

```
gservice.refresh($scope.formData.latitude, $scope.formData.longitude);
```

Bug: Era posible acceder a viajes no creados.

Un usuario podía acceder a una url todavía no creada, apareciendo, de este modo, un viaje sin destino.

Para solucionarlo se ha procedido a dos medidas: crear las url mediante expresiones regulares más complejas y realizar una búsqueda en la base de datos, de tal manera que cuando el viaje está vacío redirige al usuario a una página para crear un viaje nuevo, informándole de que la URL introducida no es válida.

15. Proyección a futuro

El desarrollo de la aplicación se encuentra en una fase embrionaria, donde sólo ha sido creada la base de lo que puede ser un proyecto mayor. Este estadio inicial, donde es posible compartir ideas de viaje entre usuarios sin necesidad de registro para que voten y acuerden qué ver, esta ideada para que la web se empiece a distribuir de una manera sencilla, pues, como se ha visto en el proceso de investigación, el hecho de funcionar sin tener que darse de alta beneficiaría su expansión.

Una vez superada esta etapa primera, se pasa a otra que ya abarca todos los aspectos recogidos en el presente trabajo: posibilidad de registro, itinerarios creados por otros usuarios o recomendados desde la plataforma, más opciones de visualización…

Además de todo esto hay una serie de características que se pueden ir añadiendo en un proceso de expansión:

Descripción e información de los lugares más allá de la proporcionada por Google. Esto puede construirse con reseñas de los usuarios o creándose desde la aplicación; reseñas por humanos o extrayendo la información de recursos informáticos.

Mejora del sistema de planificación de rutas. Este es un tema complicado, pues la resolución del denominado Problema del viajante es una tarea aún en proceso de desarrollo óptimo en las matemáticas. Pero, sin duda será posible encontrar soluciones eficientes con la potencia computacional de la que se disponga.

La expansión en la utilización de la aplicación traerá un importante beneficio con el aumento del volumen de información obtenida de los usuarios: patrones de viajes, valoración de los lugares, tiempo de estancia, datos demográficos… Esta utilización del Big Data será clave para la creación de nuevos algoritmos de recomendación y personalización de la experiencia y, más allá de esto, permite crear nuevos oportunidades de negocio que aún no han sido concebidas.

Llegar a acuerdos con otros actores en el campo de los viajes para lograr beneficios mutuos: venta de billetes adaptados al viaje en cuestión, hoteles, conductores, ofertas especiales en las cercanías,

entradas, etcétera. Este es un sector altamente propicio para tejer sinergías y colaboraciones que aporten valor a todos los *stakeholders* involucrados.

16. Conclusiones

Lo primero que habría de destacar es el reto que ha supuesto introducirse en un campo como el de las tecnologías MEAN. Si bien no ha estado exento de dificultades, las estudios realizados durante el grado (Programación web, Diseño de patrones, Bases de Datos, Ingeniería del software, etcétera.) han permitido que una tarea, en principio tan ardua, haya sido posible – en mayor o menor medida.

El *MEAN Stack* es todavía una tecnología incipiente y en proceso de expansión. Como consecuenia, se presentan una serie de inconvenientes: información menor que en otros sistemas, cambios importantes en los programas y en las empresas o grupos que los crean (notar como ejemplo las constantes tensiones surgidas en el entorno de Node.JS) Estas pequeñas trabas se ven compensadas con creces con lo aprendido y por el hecho de participar en las etapas primeras de unas herramientas de las que se espera un gran porvenir.

En ningún momento se ha seleccionado esta tecnología con el fin de “estar a la última”, en una suerte de esnobismo tecnológico. Se ha procurado una tecnología incipiente pero con el desarrollo suficiente para que su aprendizaje pueda constituir una ayuda en el entorno laboral y, además, pueda aprovecharse de un modo más eficiente todo lo aprendido en el grado.

El desarrollo del proyecto lo podemos dividir en dos etapas fundamentales: elaboración de la idea y análisis del mercado y, por otra parte, desarrollo y programación. Ambas han tenido una evolución similar; se parte de un proceso de investigación y cuando se ha llegado a una conclusión se intenta concretar en un desarrollo. Esta segunda parte del desarrollo concreto es, seguramente, la que más se ha resentido, sin duda por la falta de experiencia que - espero - se haya reducido gracias a la elaboración de este TFG

Personalmente, e independientemente del resultado final, creo haber conseguido el reto que me marqué en un inicio: conseguir enfrentarme a un proyecto de cierta envergadura en una tecnología completamente desconocida por mí. La intención de esto era prepararme para un campo donde los cambios tecnológicos son constantes y permanecer anclado a viejos paradigmas es un riesgo con consecuencias que pueden ser terribles. Un proyecto académico puede hacer que se sienten las bases para perder el miedo a afrontar nuevos escenarios en la vida laboral.

Glosario

API *f Application Programming Interface*, Interfaz de programación de aplicaciones. Es un grupo

de funciones o rutinas que permite la comunicación entre dos o más programas informáticos.

Framework *m* Conjunto de clases o estructuras estandarizadas que sirven para el desarrollo de aplicaciones informáticas.

GPS *m Global Positioning System*, Sistema de posicionamiento global. Permite, mediante un conjunto de satélites, determinar la posición de un objeto.

JSON *m JavaScript Object Notation*, Notación de objetos JavaScript. Es un formato para el intercambio de datos.

Metadatos *m* Conjunto de datos que brindan información sobre otros datos.

Middleware *m* software que asiste a una aplicación para interactuar o comunicarse con otras, con redes o con otro hardware.

MVC *m Modelo Vista Controlador* Patrón de ingeniería del software que separa los datos y la lógica de negocio de la interfaz de usuario y del módulo encargado de gestionar los eventos.

NoSQL *f* Sistema de gestión de bases de datos no realacionales (no basadas en relaciones entre tablas).

PaaS *m Platform as a service (PaaS) Plataforma como servicio* Paradigma informático que permite obtener servicios de computación a través de una red de comunicaciones.

TIC *f* Tecnologías de la información y computación.

UML *m Unified Modeling Language*, Lenguaje unificado de modelado. Es un lenguaje para representar y documentar sistemas informáticos.

Wireframe *m* Esquema gráfico que muestra información sobre los contenidos de, por ejemplo, una aplicación o sitio web.

Bibliografía

Documentos

Ornbo G., (2013) *Node.js* Madrid: Anaya Multimedia

López Quintero I. (2015) Node.js *JavaScript en el lado del servidor* Tarragona: Altaria

Wilson M., (2012) *Building Node Applications with MongoDB and Backbone* Sebstopol, Ca: O'Reilly

Haviv A, Q., (2014) MEAN Web Developement Apress

Alper D., Uraz B. (2013) *Google Maps JavaScript API Cookbook Birminghan: Packet P*ublishing

Muñoz de la Torre Monzón, A., (2013) *Introducción a Node.JS a través de Koans*

Gackenheimer C., (2013) *Node.js Recipes* Apress

Branas R., (2014) *AngularJS Essentials* Apress

(2012) “Maps Example With Google Maps And Nodejs” En: Prash's Blog

http://prazjain.wordpress.com/2012/04/19/maps-example-with-google-maps-and-nodejs/

(2015) “10 Habits of a Happy Node Hacker (2016)” En: Heroku Blog

https://blog.heroku.com/archives/2015/11/10/node-habits-2016

Sáenz Higueras, N., Vidal Oltra, R., *Redacción de textos científico-técnicos* Barcelona: Universitat Oberta de Catalunya

Jacobson L., (2015)"Solving the Traveling Salesman Problem Using Google Maps and Genetic Algorithms”

http://www.theprojectspot.com/tutorial-post/solving-traveling-salesman-problem-using-google-maps-and-genetic-algorithms/9

Hahque A., (2015) “Create a MEAN Stack Google Map App (Part I)”

https://scotch.io/tutorials/making-mean-apps-with-google-maps-part-i

De Fuenmayor López, D., González Sancho, M., *Ciclo de desarrollo de una aplicación Rich Media* Barcelona: Universitat Oberta de Catalunya

Morville, P., Rosefeld, L., (2010) *Arquitectura de la información para la World Wide Web* Barcelona: Universitat Oberta de Catalunya

(2008) *SIG per a la gestió de la base de vèrtex geodèsics de Catalunya amb Geomedia*

Garreta Domingo, M., Mor Pera, E. *Diseño centrado en el usuario* Universitat Oberta de Catalunya

Monjo Palau, T. *Diseño centrado en el usuario* Universitat Oberta de Catalunya

Gauchat, J.D. *El gran libro de HTML5, CSS3 & Javascript* Marcombo

Dunn Z. (2010) "Maps In Modern Web Design: Showcase and Examples" Smashing Magazine http://www.smashingmagazine.com/2010/04/06/maps-in-modern-web-design/

Kenny, T. (2010) "Progress Trackers in Web Design: Examples and Best Practices" http://www.smashingmagazine.com/2010/01/15/progress-trackers-in-web-design-examples-and-best-design-practices/

Goltz S. (2014) "A Closer Look At Personas: What They Are And How They Work (Part 1)"

http://www.smashingmagazine.com/2014/08/06/a-closer-look-at-personas-part-1/

VV.AA (2014) *Travel Distribution 2018: A Look Ahead By Bob Offutt and the PhoCusWright* Analyst Team

http://www.ithotelero.com/wp-content/uploads/2014/01/Analysis_TravelDistribution2018.pdf

Bank C., Cao J. (2014) *The Guide to UX Design Process & Documentation* UXPin

Sánchez, J., (2011). "En busca del Diseño Centrado en el Usuario (DCU): definiciones, técnicas y una propuesta." En: No Solo Usabilidad, nº 10, 2011. <nosolousabilidad.com>. ISSN 1886-8592 -

(2014) Hows D., Membrey P., Plugge E, *MongoDB Basics* Apress

Recursos en línea

API Google Maps: https://developers.google.com/maps/?hl=es

MongoDB: http://www.mongodb.org/

node.js: http://nodejs.org/

Stackoverflow: http://stackoverflow.com

Express: http://expressjs.com

Angular: http://www.angularjs.org

NPM: http://www.npmsjs.com

Bootstrap: http://getbootstrap.com/

IBM Bluemix: http://www.ibm.com/cloud-computing/bluemix/