Customer Relationship Manager de clientes

Memoria de Proyecto Final de Grado/Máster

Máster Universitario en Ingeniería Informática

M1.324 TFM-Desarrollo de aplicaciones web

Autor: Héctor Aguado García

Consultor: Ignasi Lorente Puchades

Profesor: César Pablo Córcoles Briongos

11 de Enero de 2016

"Gracias a mi mujer Nieves por su apoyo y su paciencia, y a mis hijos Víctor y Mario por no haber podido prestarlos toda la atención que requerían."

"*Primero tienes que aprender las reglas del juego, y después jugar mejor que nadie.*"

Albert Einstein.

Abstract

Desarrollo y entrega de un Customer Relationship Manager de clientes para ofrecer información de productos, sitios o eventos que satisfagan los intereses de los usuarios.

Como metodología de trabajo el autor ha pensado en utilizar Kanban, porque es la mejor opción que se puede adaptar a su ritmo de trabajo para poder ir obteniendo pequeñas iteraciones de trabajo y prepararse frente a posibles cambios de alcance y nuevas mejoras.

El proyecto será una aplicación web realizada con Node.js y Express en la parte backend, implementando un API Rest para acceder a los diferentes servicios. En la parte frontend se utilizará Polymer (con material design y responsive).

Para el almacenamiento de datos se utilizará una base de datos noSQL como MongoDB.

Para los tests unitarios (métodos y funciones) y de integración (para realizar las pruebas de llamadas a la API Rest) se utilizarán Mocha y Chai.

El alojamiento de la web será sobre Heroku, incluyendo un balanceador para recuperarse ante la caída de cualquiera de los servidores Node.js.

Palabras clave: **CRM, Node.js, Express, MongoDB, Polymer, Heroku y Kanban.**

Development and delivery a Customer Relationship Manager of clients to provide product information, places or events that satisfy the interests of users.

As work methodology the author has thought of using Kanban, because it's the best option that can be adapted to his workflow to be getting smaller iterations of work and to prepare for possible scope changes and enhancements.

The project is a web application develop with Node.js and Express in the backend, implementing an API REST to access different services. In the frontend Polymer (with material design and responsive) will be used.

For data storage a noSQL database as MongoDB will be used.

For unit (methods and functions) and integration (for testing of Rest API calls) test Mocha and Chai will be used.

The web hosting is over Heroku, including a balancer to recover from the fall of any of the Node.js servers.

Keywords: **CRM, Node.js, Express, MongoDB, Polymer, Heroku, and Kanban**.

Notaciones y Convenciones

Notaciones y Convenciones: Uso de tipografías en negrita para destacar algunas palabras del resto y de cursivas para nombres y expresiones. Además para las partes de código se utiliza la tipografía Calibri para diferenciar del resto.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción/Prefacio

Figura 1: Logotipo CRM

1.1 Contexto y justificación del Trabajo

Actualmente, y dada la cantidad de ofertas y planes de ocio existentes en el mercado, surge la necesidad de ofrecer a los clientes desde una misma aplicación, toda la información que se ofrece tanto en su ciudad, como en otras a las que vaya de vacaciones, trabajo, etc…, y si además se les puede ofrecer ofertas y cupones de descuento, pues mayor satisfacción obtendrán.

En este contexto es donde entra en juego el producto, es decir, la aplicación web a presentar, un Customer Relationship Manager de clientes, con capacidad de ofrecer todas las cuestiones descritas anteriormente y dar solución a la cantidad de información distribuida en diferentes webs. Todo esto será necesario llegando a acuerdos con diferentes proveedores de servicios y compañías distribuidos en Internet, para que alojen sus propuestas en nuestra base de datos.

De cara al futuro se podría aumentar la funcionalidad de los dispositivos móviles, y así poder aprovechar los servicios de geolocalización en base al lugar en que nos encontremos.

1.2 Objetivos del Trabajo

- Desarrollo de una aplicación web en Producción.
- Producto que satisfaga expectativas y necesidades de clientes.
- Aprendizaje de nuevas tecnologías.
- Utilización de metodología ágil (Kanban) para realizar un trabajo con entregas pequeñas.
- Aprender a realizar una documentación que aporte valor al producto.

1.3 Enfoque y método seguido

Se va a desarrollar un nuevo producto desde cero, en base a unos criterios que se han obtenido previamente mediante un estudio de viabilidad comercial, y que se adapta a los gustos y necesidades de una amplia muestra de clientes.

Se ha decidido esta estrategia, porque es la que mejor se adapta a las necesidades de los clientes y del autor, debido a que los productos actuales en el mercado no cumplían exactamente con la idea de aplicación que tiene el autor en mente, y alguno que pudiera ser extendido, no cumplía con los criterios de calidad exigidos.

1.4 Planificación del Trabajo

La planificación se realizará siguiendo una metodología de trabajo ágil, y más en concreto, la metodología Kanban, debido a que un producto de estas características necesita de pequeñas entregas de funcionalidades operativas y que sean capaces de aportar valor al futuro producto a mostrar.

1.5 Breve sumario de productos obtenidos

- Esqueleto de la aplicación, junto a los recursos necesarios para que la aplicación funcione correctamente, realizado con Node.js y Express.

- Base datos MongoDB, con datos de clientes precargados.
- Driver de conexión para la base de datos.
- Realización de API REST para los diferentes servicios.
- Testing de los diferentes servicios.
- Realización de la capa front con los diferentes componentes web.
- Pruebas E2E sobre la aplicación web.
- Diferentes diseños técnicos, funcionales, de pruebas, etc…
- Entrega de la memoria final con toda la información del producto desarrollado.

2. Descripción/Definición/Hipótesis

El autor plantea el Trabajo de Final de Master en base a una propuesta de realización de un CRM de clientes, es decir, un sistema que sea capaz de ofrecer a los clientes planes de ocio y restauración, así como ofertas y descuentos sobre dichos productos.
Además este trabajo permite plantar la semilla de creación del producto a largo plazo, y poder seguir creciendo y ampliando la aplicación para que pueda ser lo más robusta y atractiva que demande el mercado actual y futuro de las aplicaciones web. También se tiene en mente su expansión a los dispositivos móviles, ya que este nicho hoy en día es el que mayor crecimiento y necesidad tiene por parte del usuario.

La aplicación se plantea como un desarrollo web, con una arquitectura de capas, donde se pueden ver la parte front-end, formada por pequeños componentes web y que está desarrollada con Polymer, y la parte back-end que se encuentra dividida en cuatro capas, la capa de rutas, la capa de controladores, la capa de servicios y los objetos de acceso a la Base de Datos, y que están desarrollados con Node.js y Express, además de otras librerías, y finalmente, tenemos la Base de datos, para el almacenamiento de información de productos y clientes (MongoDB).

El planteamiento del TFM es poder conseguir un producto lo suficientemente potente, para que pueda ser utilizado por una buena cantidad de clientes y que resulte interesante y fácil de utilizar, y si bien no será una versión definitiva, sino que se pretende que sea un producto en constante evolución, siendo capaz de atraer al mayor número de usuarios en sus primeras versiones.

3. Objetivos

3.1 Principales

- Realizar el desarrollo de una aplicación web en Producción. Para un perfil técnico esto siempre es un reto y una motivación para ir mejorando en el trabajo y alcanzar nuevos conocimientos en base al trabajo e investigación realizados.

- Ofertar a los clientes un producto que satisfaga sus expectativas y pueda cumplir con sus necesidades. En base a un estudio previo realizado se ha verificado que la demanda de planes de ocio y restauración acompañados de ofertas es un mercado en auge, y gracias a este objetivo se podrá alcanzar.

- Aprender nuevas tecnologías y mejorar las capacidades de desarrollo y administración con las herramientas con las que se va a realizar la aplicación. Un buen producto será aquel que sea atractivo y usable para un gran volumen de usuarios y aquel que resulte motivador para el que lo realiza.

- Utilizar una metodología ágil (Kanban) para realizar un trabajo acorde a las capacidades del autor y al tiempo del que dispone, pudiendo hacer entregas en base a pequeñas funcionalidades realizadas, intentando que todas ellas sean “working software”.

- Aprender a realizar una documentación que aporte valor al producto realizado, y que sea consecuente, en cuanto a calidad se refiere, con el producto a entregar. Perfecto para aprender a futuro como ser capaz de realizar informes para vender un nuevo producto o servicio desarrollado o diseñado.

3.2 Secundarios

- Alojar la aplicación en un Cloud como puede ser Heroku. Una manera de tener alojada la aplicación y disponer de recursos necesarios para el perfecto funcionamiento al menor coste posible, tanto económico como de mantenimiento.

- Ser capaz de realizar un buen diseño web que pueda servir tanto para web como para dispositivos móviles y tablets. El futuro radica hoy en día en los dispositivos móviles, por lo que una aplicación de este tipo se convierte en un marco imprescindible para llegar al mayor número de usuarios.

4. Marco teórico/Escenario

El TFM se enmarca dentro de los productos destinados a llegar a una gran demanda de clientes cuyos perfiles son aquellos que les gusta disfrutar del ocio y la restauración y están enmarcados dentro de la denominada clase “media”, es decir, aquellas personas con un nivel adquisitivo medio y que les gusta disfrutar de diferentes planes generalmente de jueves a domingo, y que no están dispuestos a gastar una excesiva cantidad dinero en dichos planes.

- Antecedentes. Puede que en el mercado puedan existir productos con características similares a las del TFM, pero lo que se pretende es ofrecer un nuevo producto atractivo e interesante que puede dar cabida a una cantidad importante de usuarios.

- Estado del arte/Escenario. El producto se enmarca en la utilización las últimas tecnologías en cuanto a desarrollo de software se refiere, llevando a cabo la utilización de test de todo tipo para verificar que el código desarrollado es de calidad y cumple con las características que garantizan un software de calidad hoy en día. El escenario a su vez, se desarrolla en plataformas de última tecnología que garantizan el correcto funcionamiento y una rápida recuperación ante posibles incontinencias que pudieran ocurrir.

- Bases teóricas de referencia. La base sobre todo a nivel desarrollo es una arquitectura y estructura básica en un proyecto Node.js de tipo vertical (o modelo de capas) donde se distinguen diferentes apartados en el código para que sea más legible, funcional y pueda ser adaptado a cambios nuevos sin que el funcionamiento de lo anterior se vea prácticamente afectado.

- Otros proyectos/estudios/productos similares o relacionados. La idea básica parte de un producto de software que el autor utiliza en su trabajo diario, pero cuyo diseño, desarrollo y fin, nada tienen que ver con el producto que finalmente se presente por parte del autor.

5. Contenidos

El TFM es un proyecto de gran envergadura y por la tanto se ha decido dividir en diferentes partes más pequeñas para que la estructura del mismo sea más fácil de abordar:

- El core de la aplicación, pieza troncal de la aplicación, sin este contenido la aplicación no podría funcionar ni poder satisfacer toda la funcionalidad de la que se compone. Incorpora además el API Rest y los drivers de acceso a las bases de datos.

- El frontal de la aplicación, pieza básica de usabilidad y presentación, gracias a ello se produce la relación digital entre el usuario y la aplicación, es la parte que ve y por la cual se producen las diferentes acciones que puede llevar a cabo con la misma.

- La base de datos, sin ella no podría ser posible la interacción con los clientes ya que no se tendrán guardados los datos de los mismos ni sus preferencias y selecciones, así como la oferta de productos.

- La memoria. Pieza imprescindible del TFM, ya que es el exponente de todo el trabajo realizado y la primera piedra y toma de contacto del mismo con el cliente que ha “comprado” la aplicación. En ella se refleja todo el esfuerzo y dedicación que no se puede ver únicamente con la aplicación.

Figura 2: Contenido

6. Metodología

6.1. Tipo de metodología

El autor ha decidido decantarse para la utilización como metodología de trabajo ágil, como lo es Kanban.

Kanban es un método no para gestionar el trabajo intelectual, con énfasis en la entrega justo a tiempo, mientras no se sobrecargan a los miembros del equipo. En este enfoque, el proceso, desde la definición de una tarea hasta su entrega al cliente, se muestra para que los participantes lo vean y los miembros del equipo tomen el trabajo de una cola.

Un primer panel con las primeras tareas identificadas sería como el siguiente:

Figura 3: Tablero Kanban

6.2. Justificación

El autor seleccionó esta metodología porque se adaptaba muy bien a sus tiempos de trabajo, a la capacidad de equipo, en este caso una solo persona, por lo que no tenía sentido utilizar una metodología ágil como Scrum y porque es capaz de abordar diferentes cambios de alcance en el

desarrollo o en las tecnologías a utilizar, asunto que con una metodología de trabajo no ágil no sería capaz de atajar. Además el autor se considera un defensor de las metodologías de trabajo ágiles y son las que utiliza en su trabajo del día a día.

En este caso en particular el proceso será el siguiente: las tareas de pequeñas piezas irán siendo generadas con antelación suficiente para llenar un backlog de tareas y serán abordadas de una en una (cuando se finalice una se afrontará la siguiente). El autor será todos los perfiles que participan en el proceso e irá adecuando las tareas y tiempos a su disponibilidad.

Las iteraciones serán empezar por el backend, continuar con los tests del backend, empezar el front, testear el front (alternando las tareas con la realización de la memoria, en la que se cambia de perfil de desarrollador/arquitecto a gestor).

7. Arquitectura de la aplicación/sistema/servicio

La aplicación web está desarrollada en el lenguaje Javascript, y funciona en la parte servidora sobre Node.js y se apoya en la parte cliente sobre Polymer para distribuir y mostrar los diferentes componentes web que contiene la aplicación.

- En el lado servidor tenemos un modelo de arquitectura de tres capas, donde tenemos la siguiente estructura:

  - **Routes**, contienen el registro de todas las llamadas url al API Rest de cada uno de los componentes involucrados en la aplicación, así como las acciones a ejecutar una vez son llamados por el frontal de la aplicación, devolviendo el resultado en formato JSON. Esta capa hace uso de la capa de controllers.
  - **Controllers**, contienen las diferentes funcionalidades o servicios de los que hará uso la aplicación para obtener la información que será devuelta a la capa anterior. Comunica la capa de routes con la capa de models que accede a los datos.
  - **Models**, contiene las llamadas a la Base de datos mediante la librería driver de acceso a la misma, para realizar las operaciones CRUD (Create, Read, Update, Delete) de cada una de las colecciones.

- Como Base de datos tenemos MongoDB, una noSQL líder en el mercado de los grandes volúmenes de datos (también conocido como Big Data), que almacena los diferentes datos en estructuras de datos BSON (Binary JSON), un formato similar al JSON, que al ser Javascript trabaja de manera muy rápida con la aplicación Nodejs. Las colecciones que almacenarán los diferentes datos serán las siguientes:

  - **CLIENTS**: colección principal que almacenará los datos de los clientes, así como el listado de ofertas, lugares o eventos a los que se suscriba.

- **OFFERS**: colección con las diferentes ofertas de la app que se podrán asociar a la cuenta de los clientes.
- **PLACES**: colección con los diferentes lugares de la app que se podrán asociar a la cuenta de los clientes.
- **EVENTS**: colección con los diferentes eventos de la app que se podrán asociar a la cuenta de los clientes.

- En el lado Cliente tenemos un front-end con un modelo de aplicación Single Page Interface (SPI) con componentes web realizados con Polymer para cada una de las operativas, utilizando además javascript para las funciones de comunicación con el lado servidor.

- En cuanto a la parte de testing se utiliza Mocha como plataforma de ejecución (librería npm de node.js), y Chai como librería para realizar los test unitarios y de integración, para probar los diferentes servicios que contiene la aplicación.

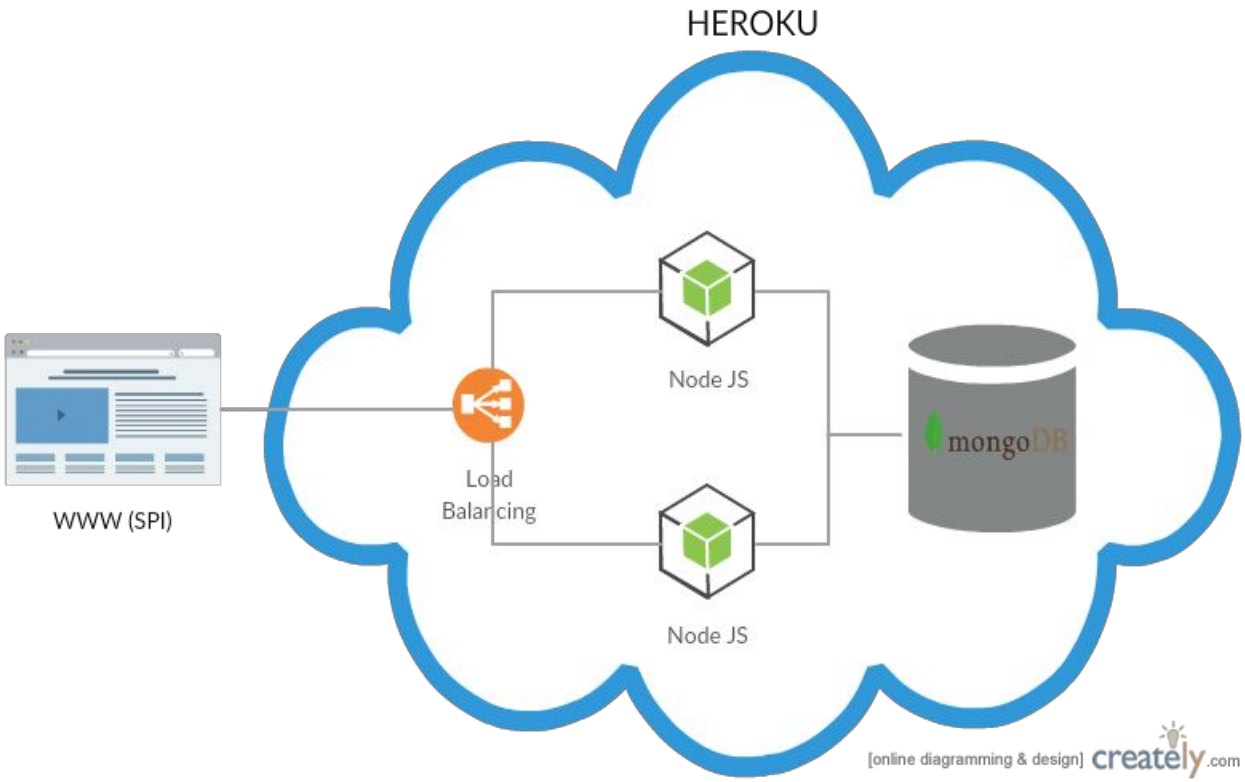


Figura 4: Arquitectura Sistema

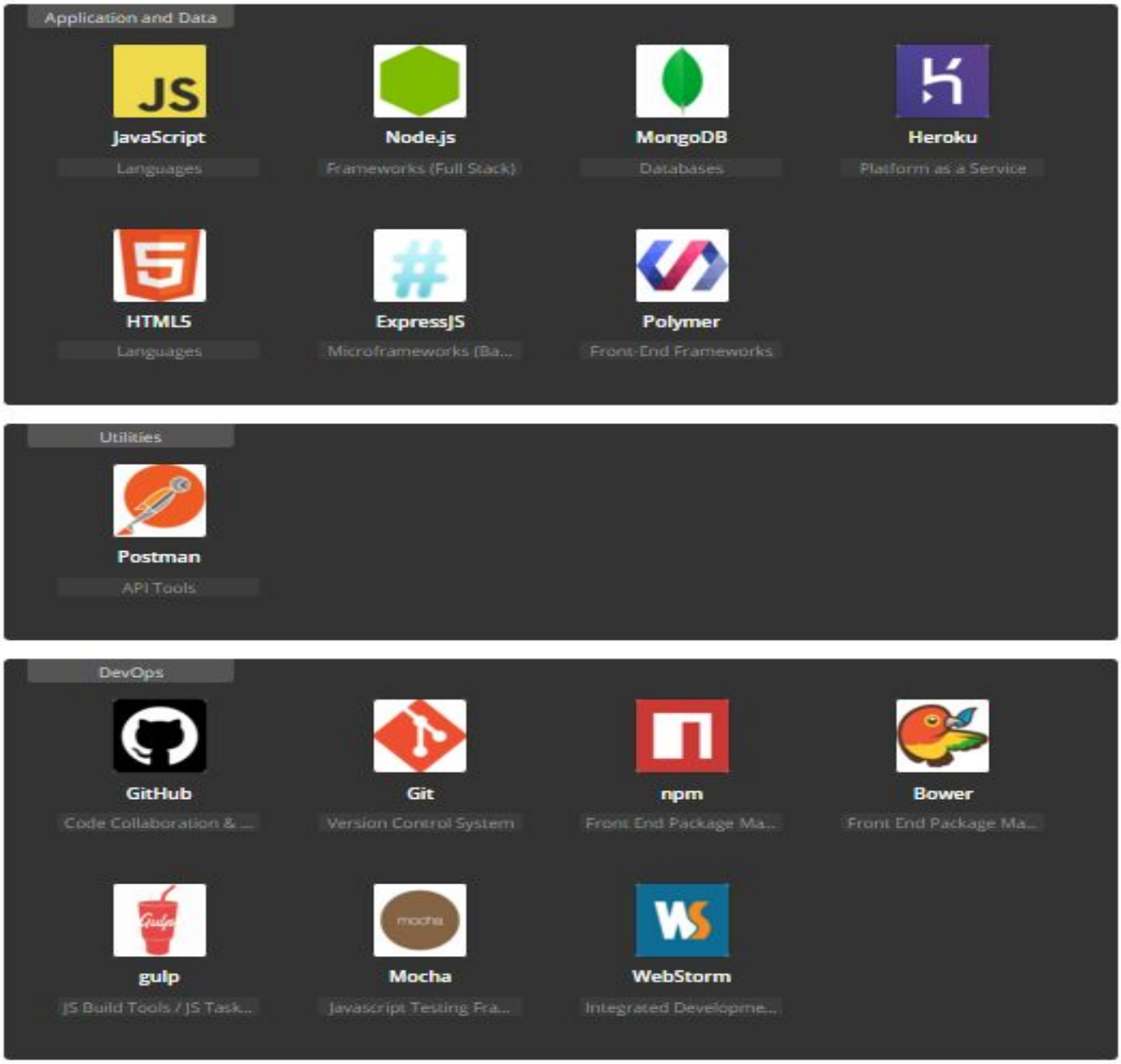


Figura 5: Stack tecnológico

En este enlace se puede comprobar en detalle todo el stack tecnológico de la aplicación registrado en la web stackhare.io

Y en este otro enlace, la aplicación en el cloud Heroku desplegada.

		Front-end			
		Clients	Offers	Places	Events
Back-end	Routes	clients.router	offers.router	places.route	events.route
	Controllers	clients.controller	offers.controller	places.controller	events.controller
	Models	clients.model	offers.model	places.model	events.model
	BBDD	CLIENTS	OFFERS	PLACES	EVENTS

Tabla 1: Arquitectura aplicación/sistema/servicio

8. Plataforma de desarrollo

Para el desarrollo de la aplicación se han utilizado los siguientes recursos, que serán organizados por las siguientes categorías:

- Software:
    - IDE de desarrollo Webstorm (de Jetbrains).
    - Entorno en tiempo de ejecución multiplataforma Node.js
    - Lenguaje de desarrollo Javascript.
    - Librerías de terceros (desarrollo, testing, etc) de NPM.
    - Librerías de terceros (frontend) de Bower.
    - Control de versiones Git.
    - Repositorio en la nube Github.
    - Microsoft Word para la memoria.
    - Heroku para el despliegue y puesta en Producción.
    - Base de Datos MongoDB.
    - Polymer como capa front de componentes web.
    - Google Chrome como navegador web.
    - Adobe Reader para lecturas pdf.
    - Notepad++ para las diferentes notas.
- Hardware:
    - Portátil HP, Intel Core i5-4300M y CPU 2.60GHz, 8 GB de RAM y Sistema Operativo de 64 bits (Ubuntu y Windows).
- Otros:
    - Panel para tablero Kanban.
    - Post-it para poner las diferentes tareas identificadas.

- Rotuladores para escribir en panel y en los post-it.

		Software	Hardware	Otros
Aplicación	1	Webstorm	HP, Intel Core i5-4300M y CPU 2.60GHz, 8 GB de RAM y Sistema Operativo de 64 bits	Panel Kanban Post-it tareas Rotuladores
	2	Node.js		
	3	Javascript		
	4	NPM		
	5	Bower		
	6	Git		
	7	Github		
	8	Heroku		
	9	MongoDB		
	10	Polymer		
	11	Chrome		
	12	Notepad++		
Memoria	1	Adobe Reader		
	2	Word		

Tabla 2: Plataforma desarrollo

9. Planificación

La planificación se adaptará a los tiempos marcados en el campus virtual y a las diferentes fechas de entrega de las PEC y de la entrega del trabajo final. Además con la utilización de la metodología ágil de trabajo el autor entregará pequeñas piezas de software operativas y funcionales.

Una información más en detalle sobre la planificación a realizar es la siguiente:

- Entrega de la PEC 1 el día 30/09/15, con la primera versión del documento y el plasmado de las ideas sobre el trabajo en papel.

- Entrega de la PEC 2 el día 28/10/15, con la continuación de la documentación y con parte del desarrollado realizado.

- Entrega de la PEC 3 el día 15/12/15, con más documentación y con una parte importante del desarrollo realizado.

- Entrega final del TFM el día 11/01/16, con la documentación completa y con el producto desarrollado y testado.

- Debate virtual para exponer el TFM el día 22/01/16.

- Los diferentes hitos serán entregas parciales cada dos semanas

- Metas:

    - Montaje de base de datos
    - Generación de esqueleto de backend
    - Generación de las capas de arquitectura backend (clientes, eventos, ofertas y lugares)
    - Creación API Rest
    - Test unitarios

- Desarrollo fontend
- Test usabilidad
- Despliegue app en cloud

- Funcionalidades:

  - Operativas testeadas en backend
  - Operativas testeadas en frontend
  - Funcionalidad completa en local
  - Subida de app a cloud Heroku.

- Diagrama Kanban de trabajo:

Figura 6: Diagrama Kanban

10. Proceso de trabajo/desarrollo

Las diferentes fases por las que pasará el TFM serán las siguientes:

- Investigación sobre las diferentes tecnologías de Base de Datos.
- Instalación de la Base de Datos MongoDB.
- Identificación de colecciones y creación de las mismas en MongoDB.
- Documentar el TFM.
- Creación del esqueleto de la parte Backend de la aplicación.
- Creación del API Rest de la aplicación para los diferentes modelos identificados.
- Continuación de la documentación.
- Testing de los procesos de las diferentes capas del API.
- Creación de las pantallas de la aplicación en la parte Front
- Continuación de la memoria.
- Testing de la parte front de la aplicación.
- Subida de la aplicación al cloud Heroku.
- Pruebas de la aplicación en Heroku.
- Finalización de la memoria.
- Entrega de la aplicación y de la memoria.

11. APIs utilizadas

Debido a la versatilidad proporcionada por la plataforma de desarrollo Node.js, existen multitud de librerías ya desarrolladas y que añaden un plus tanto de calidad, como de seguridad y facilidad en el desarrollo del software, es posible la utilización de las mismas en la aplicación. A nivel general, las más importantes al entender del autor son las siguientes:

- async, es un módulo de utilidad que proporciona funciones sencillas y de gran alcance, para trabajar con JavaScript de forma asíncrona.

- express, es una infraestructura de aplicaciones web node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Con miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla. Además proporciona una delgada capa de características de aplicación web básicas, que no ocultan las características de Node.js que tanto ama y conoce.

- helmet, librería que ayuda a proteger las aplicaciones express programando varias cabeceras HTTP para evitar recibir ataques.

- mongoDB, es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto. Forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (mongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

- chai, es una biblioteca de asserts BDD / TDD para node.js y el navegador, que puede ser fácilmente emparejada con cualquier framework de pruebas javascript.

- gulp, es una herramienta de construcción de streaming rápida e intuitiva desarrollada sobre Node.js.

- mocha, es un framework de pruebas de JavaScript rico en funciones que se ejecutan en Node.js y el navegador, haciendo las pruebas asíncronas sencillas y divertidas. Los tests se ejecutan en serie, lo que permite la presentación de informes flexible y precisa, mientras que mapea las excepciones no capturadas para corregir los casos de prueba correctamente.

- polymer, librería que está diseñada para hacer más fácil y más rápido para los desarrolladores crear grandes componentes reutilizables para las webs de hoy en día (reusables, *responsive design*, *material design*, etc…). El autor hace uso sobre todo de dos elementos de dicha librería:

    - iron-elements, son un conjunto de elementos de utilidades visuales y no visuales. Incluye elementos para trabajar con layout, user input, selection y scaffolding apps.

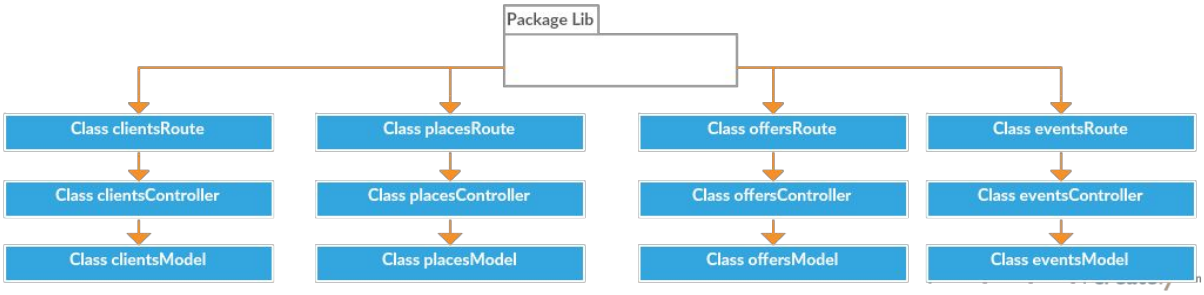
    - paper-elements, son un conjunto de elementos visuales que implementan el diseño *material design* de Google.

Es posible que en un futuro se pueda utilizar también la integración con las redes sociales más conocidas (como Facebook, Twitter, LinkedIn, etc…), así como el login con cuentas de correo como Gmail.

12. Diagramas UML

Los diferentes diagramas UML y casos de uso identificados para el diseño de la aplicación son los siguientes:

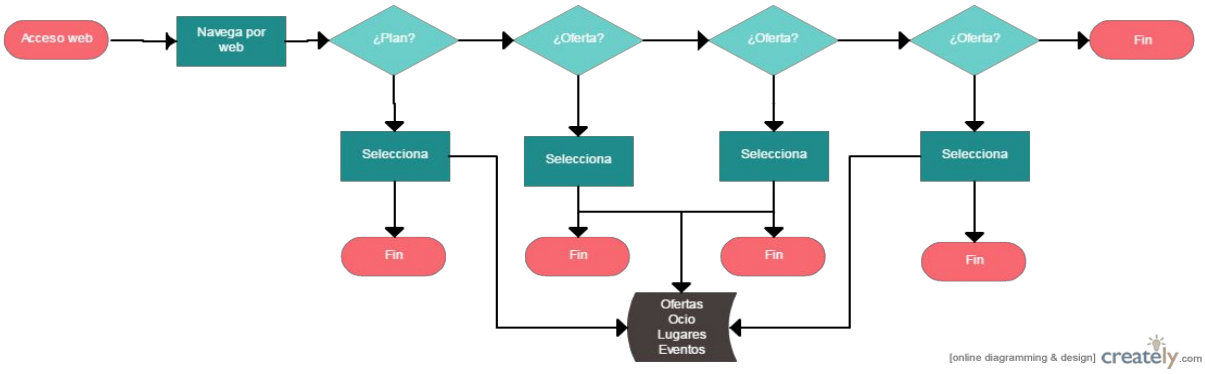
- Diagrama de clases:



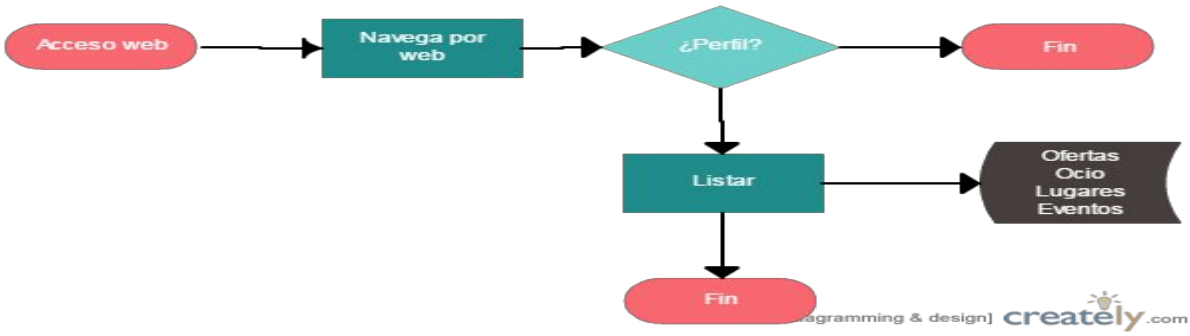
- Diagramas de interacción:

  - Acceso a web sin logado y creación de usuario:

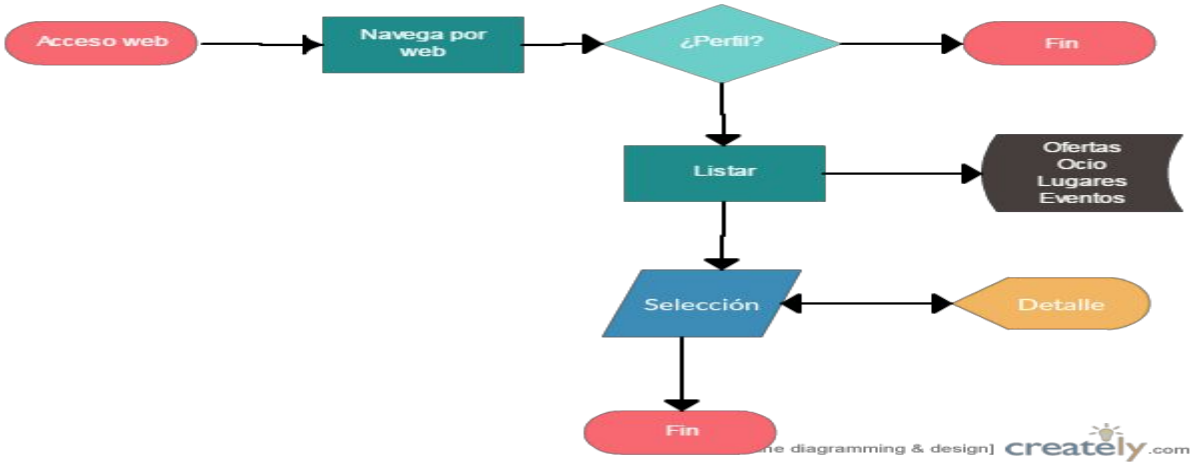
  - Usuario busca plan/oferta/evento/lugar y selecciona:



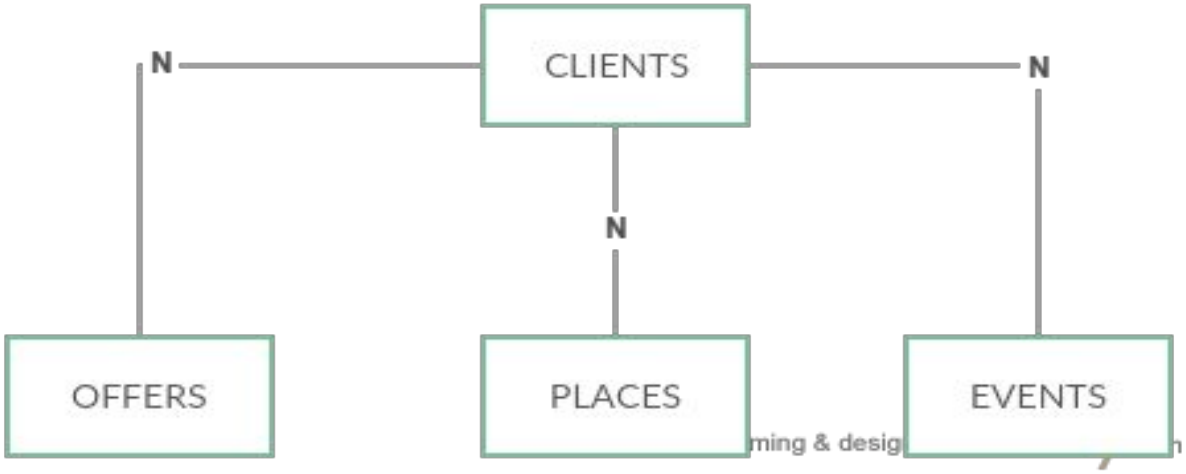
- Usuario entra en su perfil y ve lista de información:



- Usuario entra en detalle en cada opción del listado (no implementado en esta versión de la aplicación):



- Diagrama de base de datos (noSQL):



- Casos de uso:

Identificador	CU-001
Nombre	Alta usuario
Prioridad	Alta

Descripción	El usuario accede a la aplicación y desea registrarse, por lo que accederá al formulario de alta y se registrará en la app.
Actores	Nuevo usuario que accede a la aplicación
Pre-Condiciones	Aplicación web sin datos de usuario
Iniciado por	Botón de registro
Flujo	Usuario pulsa botón registro, introduce sus datos en formulario y se registra correctamente en la web y navega por la aplicación
Puesto-Condiciones	El usuario se queda logado en la web
Notas	

Identificador	**CU-002**
Nombre	Búsqueda evento/oferta/lugar
Prioridad	Alta
Descripción	Usuario accede a la aplicación (logado) e intenta buscar un plan de evento/lugar/oferta y selecciona uno que le gusta.
Actores	Usuario
Pre-Condiciones	Eventos/Planes/ofertas previamente cargados en la Base de Datos
Iniciado por	Acceso de usuario a las diferentes búsquedas
Flujo	Usuario accede a la app, selecciona Ofertas/Eventos/Lugares, hace una búsqueda y selecciona el que desee.
Puesto-Condiciones	El usuario tiene asignada un evento/oferta/lugar a su perfil
Notas	

Identificador	**CU-003**
Nombre	Listado de productos
Prioridad	Normal
Descripción	El usuario puede acceder a su perfil para ver la relación de productos que ha seleccionado para hacer uso de ellos.
Actores	Usuario
Pre-Condiciones	Usuario tiene que haber seleccionado algún producto
Iniciado por	Usuario al pinchar el link del perfil
Flujo	Usuario accede a la app, pulsa en perfil y comprueba la lista de productos que tiene.
Puesto-Condiciones	Ninguna
Notas	

Identificador	**CU-004**
Nombre	Detalle oferta/lugar/evento
Prioridad	Normal
Descripción	Usuario pulsa en una opción del listado para ver el detalle del producto seleccionado
Actores	Usuario
Pre-Condiciones	Los productos tiene que estar cargados en Base de Datos
Iniciado por	Usuario al pulsar en opción de listado (por cualquiera de los diferentes listados de la aplicación)
Flujo	• Usuario entra en perfil y ve el listado de productos y pulsa en el que le interese para ver su detalle.

	● Usuario hace una búsqueda en oferta/lugar/evento y pulsa en el resultado que desee para ver su detalle.
Puesto-Condiciones	Ninguna
Notas	

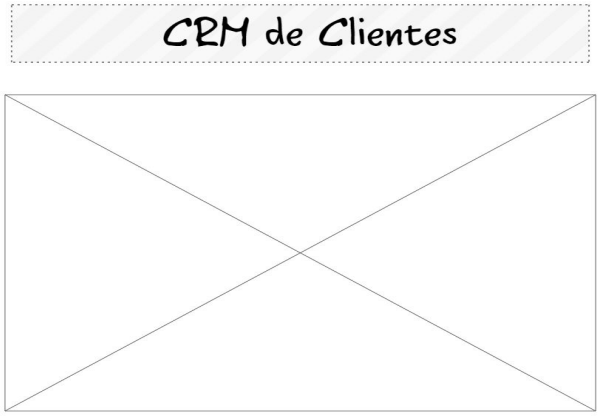
13. Prototipos

A la hora de diseñar la aplicación, y siguiendo los diseños de los flujos de interacción del apartado 16. Diseño conceptual, se han identificado los siguientes prototipos:

13.1 Lo-Fi

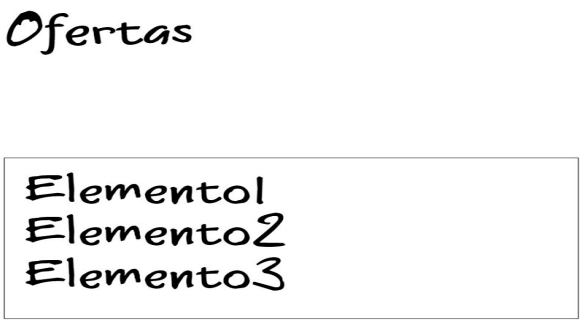
- Wireframes, representaciones esquemáticas de las pantallas de la aplicación web sin elementos gráficos que muestran contenido y comportamiento de las páginas. Tenemos los siguientes:

  - Pantalla de inicio:

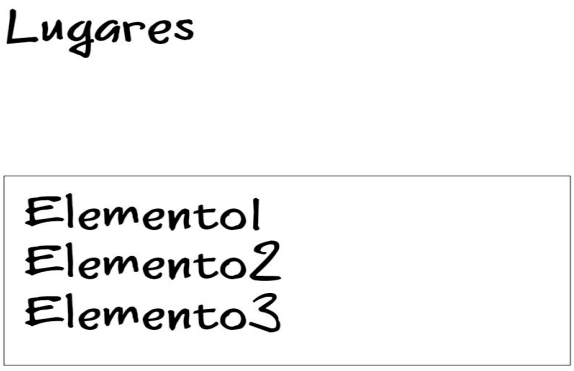


  - Pantalla de formulario de alta:

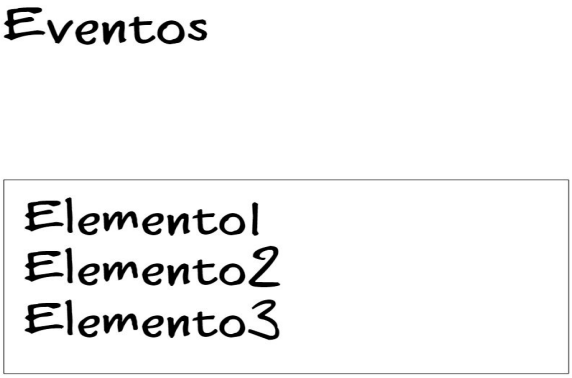
- Pantalla de ofertas:



- Pantalla de lugares:



- Pantalla de eventos:



- Pantalla de detalle de productos (lugares, ofertas y eventos):

Detalle <producto>

Info1
Info2
Info3
Info4

- Pantalla de perfil Cliente:

Cliente

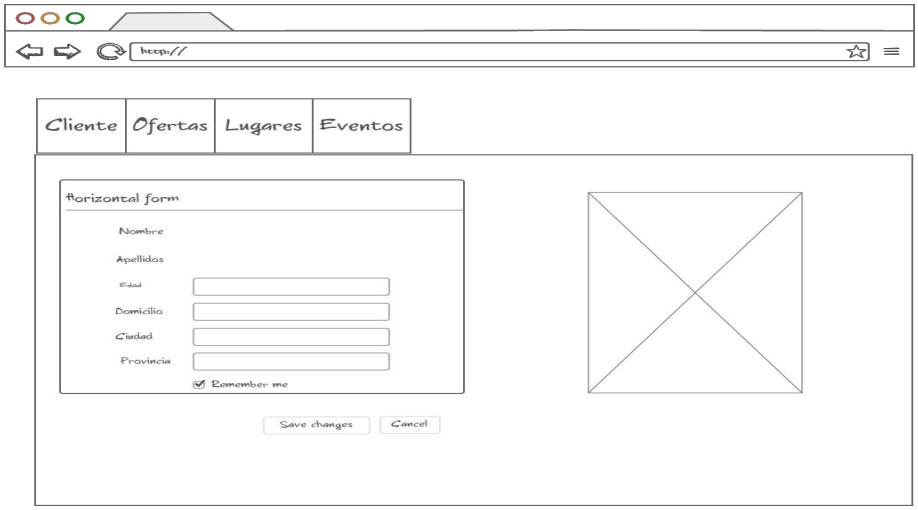
Oferta1
Evento1
Lugar1

13.2 Hi-Fi

- Mockups, son utilizados por los diseñadores principalmente para la adquisición de comentarios por parte de los usuarios. Tenemos los siguientes:

  - Pantalla de inicio:

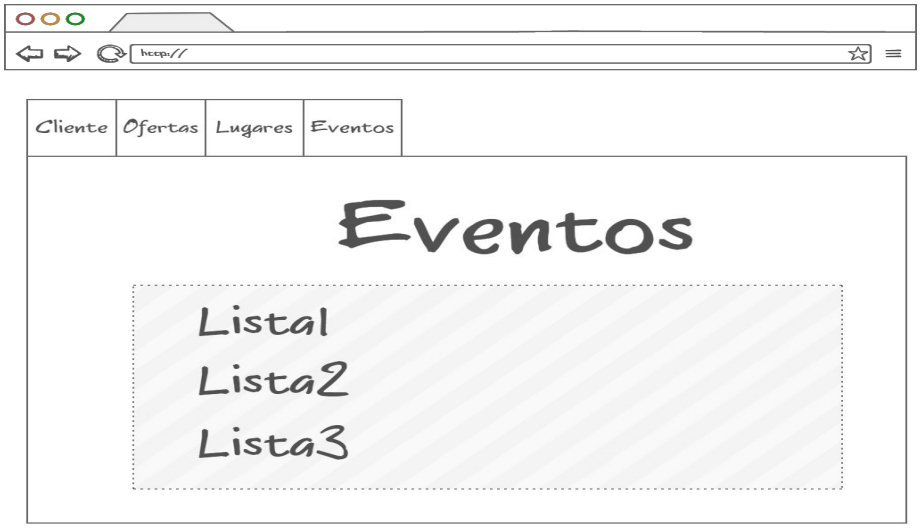
- Pantalla de formulario de alta:



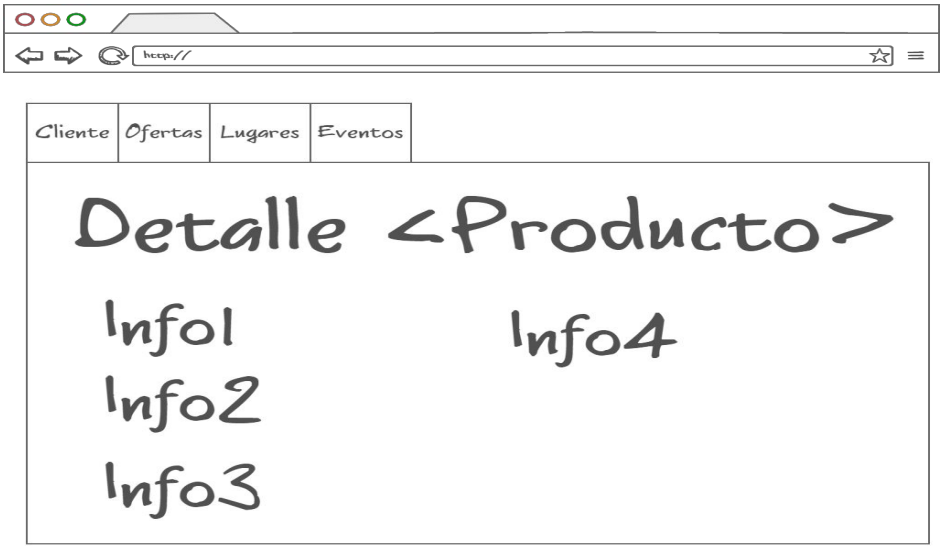
- Pantalla de ofertas:

- Pantalla de lugares:

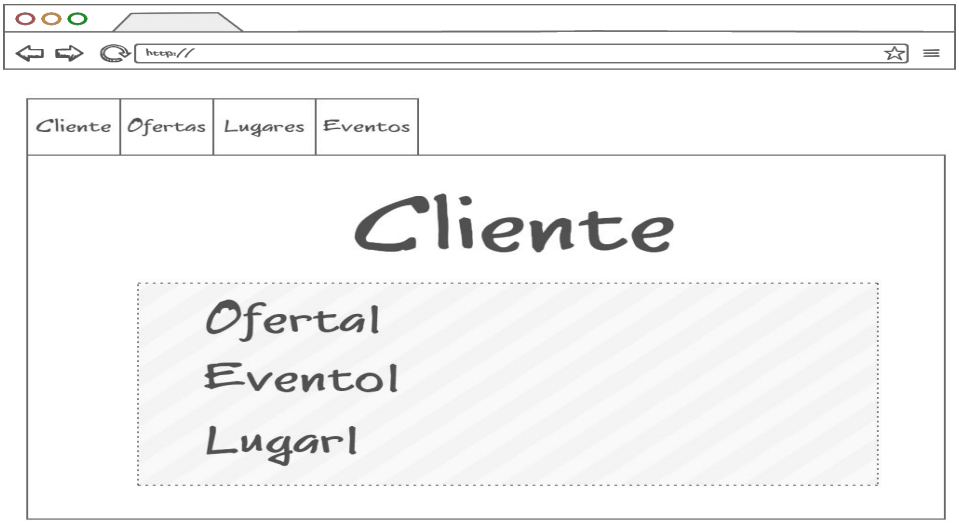
- Pantalla de eventos:



- Pantalla de detalle de productos (lugares, ofertas y eventos):



- Pantalla de perfil Cliente:


http://
Cliente
Ofertas
Lugares
Eventos
Cliente
Ofertal
Eventol
Lugarl

14. Evaluación

En el momento de realizar la evaluación para el DCU se han identificado los apartados siguientes:

- El conjunto de preguntas de información sobre el usuario que realizaría el test:
    - ¿Tiene interés por la aplicación?
    - ¿Proporciona una experiencia amigable?
    - ¿Le resulta útil?
    - ¿Los temas tratados son importantes para usted?
    - ¿Recomendaría la aplicación a familiares y amigos?
- Las tareas que los usuarios debería realizar:
    - Darse de alta como cliente
    - Buscar evento y seleccionar uno
    - Buscar lugar y seleccionar uno
    - Buscar oferta y seleccionar una
    - Entrar en perfil cliente y ver el listado de eventos, ofertas y lugares
    - Seleccionar un producto para ver su detalle en perfil de usuario
    - Seleccionar un producto para ver su detalle en listado de ofertas, lugares y eventos
- Las preguntas referentes a las tareas:
    - ¿Los datos para el alta son suficientes?
    - ¿Resultan útiles los diferentes buscadores?
    - ¿El listado de elementos buscado es excesivo?
    - ¿El detalle del producto proporciona suficiente información?
    - ¿El detalle de perfil le resulta útil?
    - ¿Los diferentes productos seleccionados son amigables en el perfil?

15. Usuarios y contexto de uso

A la hora de seleccionar los perfiles de usuario el autor se ha decantado por la técnicas de indagación “encuestas” y “análisis competitivo”, esto es así porque ambas son compatibles para encontrar los perfiles que se desean que utilicen la aplicación. Por un lado las encuestas nos dan un gran volumen de gustos, particularidades y demandas que plantea el usuario a la hora de elegir o buscar un plan de ocio por Internet, pero aunque el volumen sea grande, se deberían de sesgar bastantes resultados, porque en la encuesta en un cierto porcentaje, puede que existan respuestas que no sean reales.

La manera de compensar estas desviaciones en los resultados reales a obtener es realizar la segunda técnica y obtener feedback de los posibles usuarios recogiendo información de por dónde navegan o como se las ingenian para buscar y obtener los resultados que desean; con esto conseguimos un gran volumen de información para poder utilizarla y generar los diferentes perfiles que pueden hacer uso de la aplicación.

Con esta primera aproximación se abarcaría la parte de indagación en la selección de los perfiles y se comenzaría a realizar los dos procesos durante un periodo de tres días, primero pasando la encuesta a los usuarios seleccionados y a continuación dejando que naveguen por Internet planteándoles una serie de ejercicios para que sean capaces de buscar y obtener unos resultados para los problemas que se plantean (lo que se pretende evaluar son los pros y contras de diferentes productos similares en la red).

Una vez realizadas las dos técnicas se obtienen unos resultados bastantes acentuados para poder dirimir cuales son los perfiles de usuario que pueden hacer uso de la aplicación y son los siguientes:

- Persona con pareja y trabajador a jornada completa.

Características	Colectivo que tiene un nivel adquisitivo alto, independientes, edad comprendida entre 28 a 45 años y que se manejan

	perfectamente con los pagos, descuentos y reservas por Internet.
Contextos de uso	Generalmente uso de la aplicación de jueves a domingo realizando búsquedas en casa, el trabajo y con el móvil (será necesario sacar la aplicación móvil cuanto antes o generar la web de tipo responsive).
Análisis de tareas	Este tipo de usuario busca realizar las búsquedas y las gestiones con la web lo más rápido y sencillo posible pero siempre buscando descuentos y ofertas o en la propia web o en webs de terceros.

- Soltero y trabajador a jornada completa.

Características	Colectivo de personas con nivel adquisitivo medio/bajo, no independientes, y que gastan su dinero en ocio sin preocupaciones de facturas. Edad entre 18 a 30 años, con un gran dominio de la web y conocedores de los mejores planes de ocio.
Contextos de uso	Generalmente cualquier día de la semana es válido para usar la aplicación siendo los días de jueves a domingo los que más se utilizaría.
Análisis de tareas	Buscan encontrar un resultado rápidamente en lugares típicos y céntricos de la ciudad donde viven o donde han ido de vacaciones sin importarles mucho el precio del sitio.

- Universitario

Características	Colectivo de personas con nivel adquisitivo bajo, y que gastan el dinero que reciben de sus padres en ocio. Edad entre 18 a 25

	años, con un gran dominio de la web y conocedores de los mejores planes de ocio con los precios más ajustados.
Contextos de uso	Generalmente cualquier día de la semana es válido para usar la aplicación.
Análisis de tareas	Buscan encontrar un resultado rápidamente en lugares típicos y céntricos de la ciudad donde viven o cerca de algún campus universitario.

- Desempleados

Características	Colectivo de personas con nivel adquisitivo bajo, y que gastan poco dinero en ocio. Edad entre 35 a 60 años, con un bajo dominio de la web y que buscan raramente planes de ocio por Internet.
Contextos de uso	Generalmente una vez cada 3 meses
Análisis de tareas	Buscan encontrar algo muy barato y a poder ser con ofertas que acompañen al producto. Hay que generar una experiencia atractiva para que este tipo de usuario repita con mayor asiduidad.

- Personas con hijos

Características	Colectivo que tiene un nivel adquisitivo alto, independientes, edad comprendida entre 28 a 45 años y que se manejan perfectamente con los pagos, descuentos y reservas por Internet y además necesitan de lugares que sean para ir con niños.
Contextos de uso	Generalmente uso de la aplicación de sábado y domingo realizando búsquedas en casa, el trabajo y con el móvil (será

	necesario sacar la aplicación móvil cuanto antes o generar la web de tipo responsive).
Análisis de tareas	Este tipo de usuario busca realizar las búsquedas y las gestiones con la web lo más rápido y sencillo posible pero siempre buscando descuentos y ofertas o en la propia web o en webs de terceros y poniendo como requisito que sean aptas para ir con niños de todas las edades, siendo un plus algún descuento u oferta.

16. Diseño conceptual

A partir de la información recopilada en el apartado anterior, donde se identificaron los perfiles "*Persona con pareja y trabajador a jornada completa*", "*Soltero y trabajador a jornada completa*", "*Universitario*", "*Desempleados*" y "*Personas con hijos*" se han diferenciado los siguientes escenarios de uso:

- Escenario 1:

Perfil de usuario	"Persona con pareja y trabajador a jornada completa"
Contexto	Jueves a domingo, búsquedas en casa, el trabajo y con el móvil, buscando planes de ocio para divertirse.
Objetivos	Disfrutar de una amplia oferta de ocio y restauración sin importar mucho el precio.
Tareas para alcanzar objetivos	● Buscar en la app la mejor relación de productos. ● Ver el listado de encontrados. ● Pulsar los que desee para ver en detalle el que desee. ● Seleccionar la oferta o producto que desee para almacenarla a su perfil.
Necesidades de información	Interfaz sencilla para que el usuario se sienta cómodo, los propios menús deberían ser suficiente información para obtener un resultado satisfactorio.
Funcionalidades que necesita	Buscador de productos y ofertas y listado de búsquedas encontradas junto a un link a cada producto encontrado para verlo de manera individual en detalle.
Desarrollo de tareas	● Buscador (básico y/o avanzado) ● Lista de productos encontrados ● Detalle de producto seleccionado

- Escenario 2:

Perfil de usuario	"Soltero y trabajador a jornada completa"
Contexto	Cualquier día de la semana, siendo de jueves a domingo los que más se utilizaría, buscando diferentes opciones de ocio.
Objetivos	Encontrar una oferta de manera rápida y sencilla.
Tareas para alcanzar objetivos	● Buscar en la app la mejor relación de productos.

	● Ver el listado de encontrados. ● Pulsar los que desee para ver en detalle el que desee. ● Seleccionar la oferta o producto que desee para almacenarla a su perfil.
Necesidades de información	Interfaz sencilla para que el usuario se sienta cómodo, los propios menús deberían ser suficiente información para obtener un resultado satisfactorio.
Funcionalidades que necesita	Buscador de productos y ofertas y listado de búsquedas encontradas junto a un link a cada producto encontrado para verlo de manera individual en detalle.
Desarrollo de tareas	● Buscador (básico y/o avanzado) ● Lista de productos encontrados ● Detalle de producto seleccionado

- Escenario 3:

Perfil de usuario	"Universitario”
Contexto	Generalmente cualquier día de la semana es válido para usar la aplicación, buscando alternativas de ocio económicas.
Objetivos	Encontrar una oferta lo más económica posible y si es con descuento mejor.
Tareas para alcanzar objetivos	● Buscar en la app la mejor relación de productos. ● Ver el listado de encontrados. ● Pulsar los que desee para ver en detalle el que desee. ● Seleccionar la oferta o producto que desee para almacenarla a su perfil, previamente introduciendo algún código de oferta.
Necesidades de información	No aplica, generalmente un perfil de este tipo tienen un dominio muy alto de las aplicaciones web.
Funcionalidades que necesita	Buscador de productos y ofertas y listado de búsquedas encontradas junto a un link a cada producto encontrado para verlo de manera individual en detalle.
Desarrollo de tareas	● Buscador (básico y/o avanzado) ● Lista de productos encontrados ● Detalle de producto seleccionado ● Posibilidad de ofrecer código de oferta

- Escenario 4:

Perfil de usuario	"Desempleados”
Contexto	Generalmente una vez cada 3 meses, pudiendo elegir algún plan lo más barato posible.
Objetivos	Encontrar una oferta de manera rápida y sencilla, y que sea muy económica
Tareas para alcanzar objetivos	● Buscar en la app la mejor relación de productos ● Posibilidad de ordenar por precio. ● Ver el listado de encontrados. ● Pulsar los que desee para ver en detalle el que desee. ● Seleccionar la oferta o producto que desee para almacenarla a su perfil.
Necesidades de información	Interfaz amigable y atractiva para que el usuario se sienta cómodo y pueda volver a repetir, los propios menús deberían ser suficiente información para obtener un resultado satisfactorio, aun así puede ser interesante algún tipo de “*tooltip”* sobre algunos apartados para facilitar aún más el entendimiento de la operativa.
Funcionalidades que necesita	Buscador de productos y ofertas y listado de búsquedas encontradas junto a un link a cada producto encontrado para verlo de manera individual en detalle.
Desarrollo de tareas	● Buscador (básico y/o avanzado) ● Lista de productos encontrados ● Detalle de producto seleccionado ● Posibilidad de ofrecer código de oferta y/o descuento por primera selección o similares

- Escenario 5:

Perfil de usuario	"Personas con hijos”
Contexto	Sábado y domingo realizando búsquedas en casa, el trabajo y con el móvil, teniendo una amplia variedad de oferta para familias
Objetivos	Encontrar una oferta de manera que se adapte a las familias.
Tareas para alcanzar objetivos	● Buscar en la app la mejor relación de productos. ● Ver el listado de encontrados. ● Pulsar los que desee para ver en detalle el que desee. ● Seleccionar la oferta o producto que desee para almacenarla a su perfil.

Necesidades de información	Interfaz sencilla para que el usuario se sienta cómodo, los propios menús deberían ser suficiente información para obtener un resultado satisfactorio.
Funcionalidades que necesita	Buscador de productos y ofertas y listado de búsquedas encontradas junto a un link a cada producto encontrado para verlo de manera individual en detalle.
Desarrollo de tareas	● Buscador (básico y/o avanzado) ● Lista de productos encontrados ● Detalle de producto seleccionado ● Indicador de actividad o plan especialmente indicado para familias.

17. Seguridad

La aplicación hace uso de una librería de seguridad para Express llamada **Helmet**, que añade una serie de cabeceras HTTP que securizan las diferentes llamadas realizadas a la aplicación.

Helmet es en realidad una colección de 10 funciones de middleware más pequeñas que establecen las siguientes cabeceras HTTP:

- *contentSecurityPolicy*: para ajustar contenido Política de Seguridad
- *dnsPrefetchControl*: Control de precarga de DNS del navegador
- *frameguard*: para evitar clickjacking
- *hidePoweredBy*: para eliminar la cebecera By-X-Powered
- *hpkp*: para “pinning” el HTTP Public Key
- *hsts*: para la seguridad estricta en el transporte HTTP
- *ieNoOpen*: establece X-Download-Options para navegadores superiores a IE8
- *nocache*: para desactivar la caché del cliente
- *noSniff*: para mantener los clientes de sniffing del MIME type.
- *xssFilter*: añade algunas pequeñas protecciones XSS

Ejecutar app.use (helmet()) incluye 6 de los 10 descritos, dejando de lado *contentSecurityPolicy*, *dnsPrefetchControl*, *hpkp*, y *noCache*. También se puede utilizar cada módulo por separado, tal como se indica en la web de Helmet: https://github.com/helmetjs/helmet

A su vez se cuenta con todos los criterios y funcionalidades proporcionadas por Heroku dentro del despliegue de la aplicación en la Cloud (tolerancia a fallos, caída del servicio o de la base datos).

18. Tests

Para las diferentes pruebas a realizar sobre la aplicación se han identificados los siguientes tipos de tests:

- **Unitarios**: pruebas que se van a realizar sobre cada una de las funciones de los diferentes ficheros que componen la parte backend. Con esto se consigue que cuando haya que añadir nueva funcionalidad o cambiar algún existente se verifique que no se ha roto nada y que todo sigue funcionando como debería porque se pasarán estos tests y se verificará que todo está bien. Se realizarán dentro de la aplicación como paso previo a Producción con Mocha y Chai. En concreto se han testeado todas las funciones de acceso a base de datos de la capa model (alba, baja, modificación y consulta) y todas las de los diferentes controllers que comunican la capa de routes con la de models.
  Algunos ejemplos son los siguientes:

  - Fichero client.model.test:

    - Si el fichero existe:

```
it('clientsModel must exists', function () {
  var result = clientsModel;
  expect(result).to.be.an('object');
  expect(result).to.include.keys(['mongo']);
});
```

    - Obtener clientes:

```
it('getClients', function (done) {
  clientsModel.mongo.getClients(app.db, function (err, data) {
    expect(err).to.be.null;
    expect(data).to.be.string;
    done();
  });
});
```

- Insertar clientes desde un fichero mock de datos:

```
for(key in data){
  it('putClient data '+key, function (done) {
    client = {
      id : data[key]
    };
    clientsModel.mongo.putClient(app.db, client, function (err, data) {
      expect(err).to.be.null;
      expect(data).to.be.string;
      done();
    });
  });
}
```

- Fichero client.controller.test:
  - Actualizar un cliente:

```
for(key in data){
  it('postClient data '+key, function(done) {
    client = {
      id : data[key]
    };
    clientsController.postClient(client, function(err, data) {
      expect(err).to.be.null;
      expect(data).to.be.string;
      done();
    });
  });
}
```

- Borrar un cliente:

```
for(key in data){
  it('deleteClient data '+key, function(done) {
    client = {
      id : data[key]
    };
    clientsController.deleteClient(client, function(err, data) {
      expect(err).to.be.null;
      expect(data).to.be.string;
      done();
    });
  });
}
```

- **Integración**: pruebas que se van a realizar sobre cada una de las funciones de los diferentes ficheros que componen la parte de routes del backend, es decir, se probarán cada una de las urls expuestas como API Rest de la aplicación. Con esto se consigue que cuando haya que añadir nueva funcionalidad o cambiar algún existente se verifique que no se ha roto nada y que todo sigue funcionando como debería porque se pasarán estos tests y se verificará que todo está bien. Se realizarán dentro de la aplicación como paso previo a Producción con Mocha y Chai. Se han testeado las diferentes llamadas al API Rest de cada entidad (cliente, oferta, evento, lugar), para todas las CRUD operations.
  Un ejemplo sería el siguiente:

  - Fichero clients.route.test:

    - Obtener clientes:

```
it('Get clients', function (done) {
  server
    .get('/clients')
    .expect("Content-type", /json/)
    .end(function (err, res) {
      expect(err).to.be.null;
      expect(res).to.be.string;
      assert.equal(res.status, 200);
      done();
    });
});
```

- **Usabilidad**: estudio de comportamiento realizado sobre los usuarios identificados anteriormente para predecir el comportamiento que tendrán al utilizar la aplicación y que condicionan el desarrollo a una mejor experiencia de usuario.

- **Usuario (funcionales)**: pruebas para verificar que la capa front funciona correctamente y en base a los diferentes flujos de interacción, que todo funciona correctamente. Se realizarán de manera manual probando cada una de las navegaciones entre pantallas, así como la obtención de datos entre la parte frontend y la parte backend.

- **Seguridad**: pruebas para verificar que la aplicación responde correctamente sobre un entorno cloud. Se realizarán en la plataforma Heroku.

19. Versiones de la aplicación/servicio

Al estar trabajando en una metodología ágil las diferentes entregas de la aplicación se agrupan en features que conformarán una versión utilizable de la aplicación.

- Beta: se considera como la primera versión entregable de la aplicación. Contiene API de acceso a los datos almacenados en la base de datos, parte importante del frontend de la aplicación y todos los servicios que comunican el frontend con el backend.

- 1.0.0 (y sucesivas): Diferentes versiones producidas de manera iterativa, siendo la posición (major, medium o minor) la que subirá en base a un cambio pequeño o más importante

20. Requisitos de instalación/implantación/uso

Los requisitos necesarios previos a la instalación de la aplicación son los siguientes:

- Software:
  - Sistema operativo Windows, Linux o Mac.
  - Disponer de la plataforma Node.js instalada, instrucciones de instalación en su web: https://nodejs.org/en/download/
  - Disponer la aplicación (base de datos) MongoDB, instrucciones de instalación en su web: https://www.mongodb.org/downloads
  - Arrancar MongoDB si el sistema Operativo no lo arranca automáticamente.
- Hardware:
  - No se requiere un entorno específico para ejecución, pero se recomienda un equipo con al menos 4 GB de memoria RAM.
- Formación/Conocimientos:
  - No es necesaria una formación específica para poder utilizar la web.

21. Instrucciones de instalación/implantación

Para poder instalar la aplicación en entorno local es necesario seguir los siguientes pasos:

1. Importación de los ficheros JSON para la base de datos MongoDB mediante la ejecución de los siguientes comandos en la consola:
    a. *mongoimport -d crmclients -c PLACES --file places.json --jsonArray*
    b. *mongoimport -d crmclients -c OFFERS --file offers.json --jsonArray*
    c. *mongoimport -d crmclients -c EVENTS --file events.json --jsonArray*
    d. *mongoimport -d crmclients -c CLIENTS --file clients.json --jsonArray*

2. Copia de la carpeta del proyecto en la ruta deseada.

3. Acceso a la carpeta para poder ejecutar los siguientes comandos.

4. Ejecución del comando *"npm install"*

5. Ejecución del comando *"npm install –g bower"*

6. Ejecución del comando *"bower install"*

7. Ejecución del comando *"npm instal –g gulp"*

8. Ejecución del comando *"npm run backend"*

9. Ejecución del comando *"npm run frontend"*

22. Instrucciones de uso

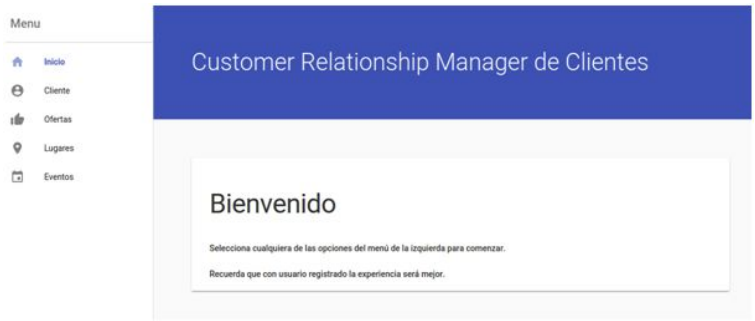
Una vez se han seguido los pasos de instalación, la web automáticamente se abrirá en el navegador predeterminado con la pantalla de Inicio.

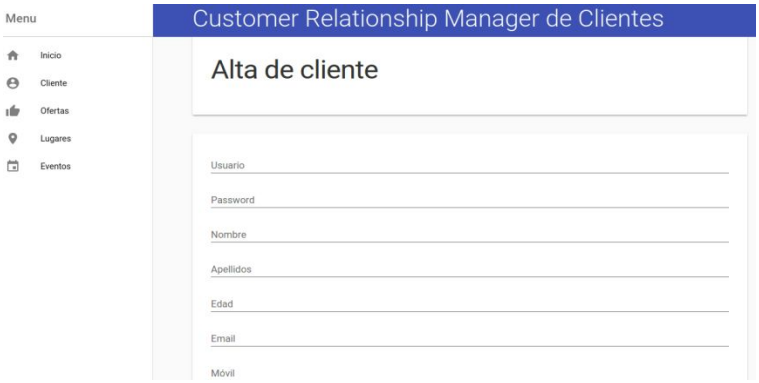
Para acceder a los diferentes apartados de la web bastará con hacer click en los enlaces de la izquierda, pudiendo seleccionar cualquiera de las opciones deseadas.

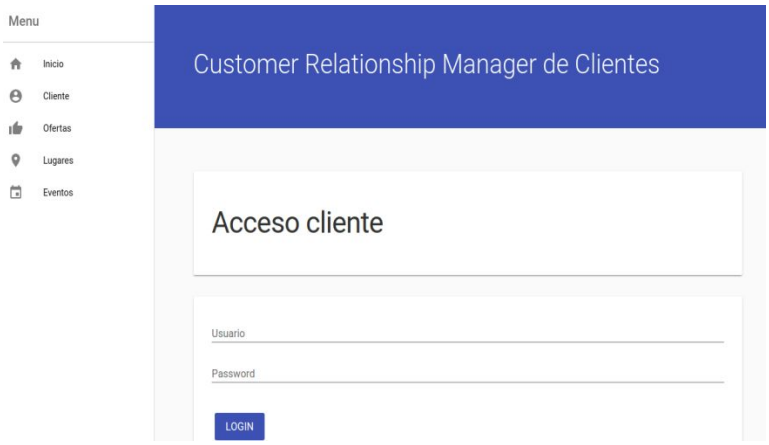
Siguiendo los diagramas de interacción indicados en el apartado 12 del documento, nos encontramos con los siguientes flujos:

- *Acceso a web sin logado y creación de usuario*:

  El usuario pulsa en el menú de Cliente, aparece pantalla de login, donde el usuario pulsa en el link "*Si desea crear un usuario nuevo pulse el link*", posteriormente aparecerá la pantalla de registro de usuario, donde se introduce todos los datos y se pulsa en botón "Registrar", con lo que se dará de alta en el sistema el usuario y se volverá a la pantalla de inicio:



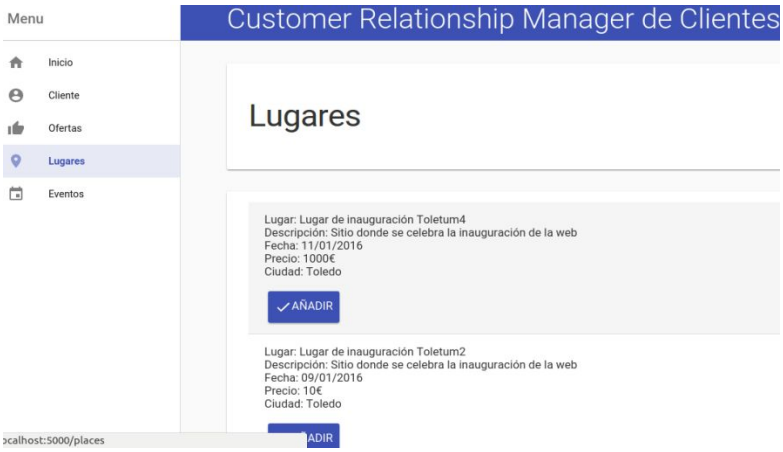


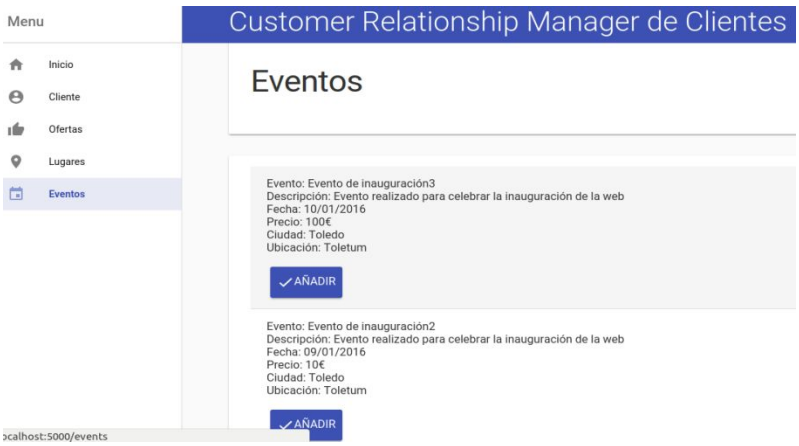


- *Usuario busca plan/oferta/evento/lugar y selecciona*:

  Usuario pulsa en alguno de los botones de menú, Ofertas, Eventos o Lugares, una vez hecho esto aparecerán las pantallas respectivas de cada producto con los últimos contenidos publicados:

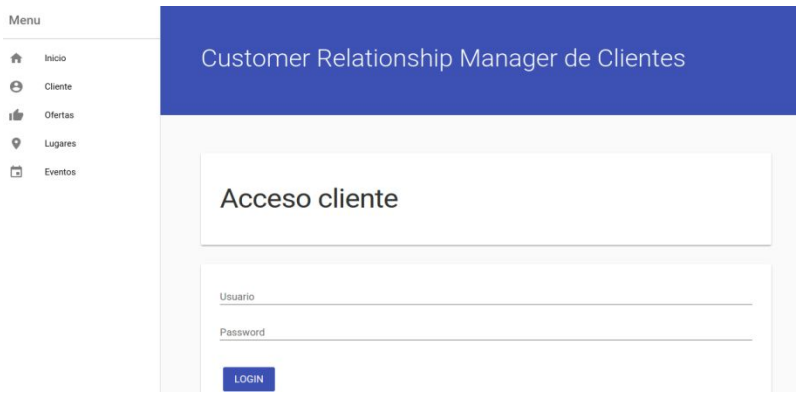

Menu
Customer Relationship Manager de Clientes
Inicio
Cliente
Ofertas
Lugares
Eventos
Ofertas
Oferta: Oferta de inauguración2
Descripción: Oferta realizada para celebrar la inauguración de la web
Fecha: 09/01/2016
Precio: 10€
Ciudad: Toledo
Ubicación: Toletum
AÑADIR
Oferta: Oferta de inauguración4
Descripción: Oferta realizada para celebrar la inauguración de la web
Fecha: 11/01/2016
Precio: 1000€
Ciudad: Toledo
Ubicación: Toletum
AÑADIR

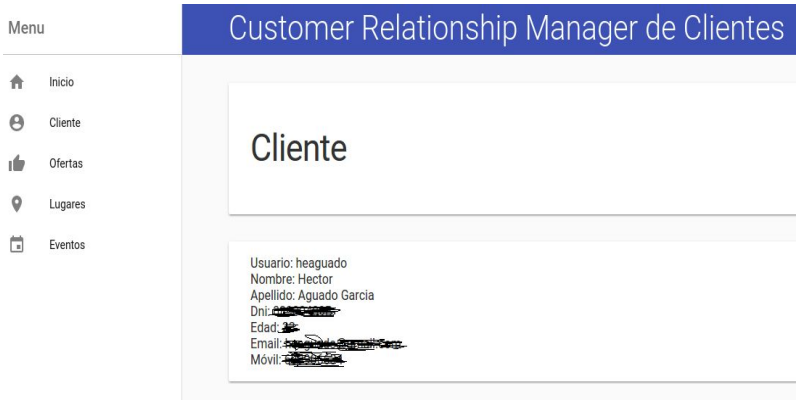

Menu
Customer Relationship Manager de Clientes
Inicio
Cliente
Ofertas
Lugares
Eventos
Lugares
Lugar: Lugar de inauguración Toletum4
Descripción: Sitio donde se celebra la inauguración de la web
Fecha: 11/01/2016
Precio: 1000€
Ciudad: Toledo
AÑADIR
Lugar: Lugar de inauguración Toletum2
Descripción: Sitio donde se celebra la inauguración de la web
Fecha: 09/01/2016
Precio: 10€
Ciudad: Toledo
calhost:5000/places


Menu
Customer Relationship Manager de Clientes
Inicio
Cliente
Ofertas
Lugares
Eventos
Eventos
Evento: Evento de inauguración3
Descripción: Evento realizado para celebrar la inauguración de la web
Fecha: 10/01/2016
Precio: 100€
Ciudad: Toledo
Ubicación: Toletum
AÑADIR
Evento: Evento de inauguración2
Descripción: Evento realizado para celebrar la inauguración de la web
Fecha: 09/01/2016
Precio: 10€
Ciudad: Toledo
Ubicación: Toletum
AÑADIR
calhost:5000/events

- *Usuario entra en su perfil y ve lista de información:*

  Usuario pulsa en el botón Cliente, le aparecerá la pantalla de login, donde introducirá su usuario y su password y pulsará en el botón Login, una vez hecho esto, le aparecerá la pantalla de cliente con su información:





23. Bugs

- El **login** no funciona correctamente:

  Al ir a ejecutar la aplicación no funcionaba el login, en el momento actual solo se ha corregido la verificación de que el usuario existe en la base de datos, esto es así porque debido a la falta de tiempo no ha podido ser implementado en esta versión. La solución que se tiene en mente es utilizar JWT, y poder logar a través del uso de un token generado a partir del usuario y la password, pero es algo que se implementará en una futura versión.

- El botón de **añadir** eventos, ofertas y lugares a clientes no funciona:

  Se ha decidido quitarlo temporalmente hasta que el problema del login esté solucionado y el usuario se pueda mantener “vivo” en la sesión para poder añadirle la información de cada apartado.

- El **usuario** no se mantiene logado:

  Relacionado con el primero y no solucionado, la idea es que el usuario una vez esté logado se pueda mantener conectado mediante el uso de un token identificativo para así poder obtener una mejor experiencia de usuario.

Todos ellos fueron identificados al comenzar a añadir funcionalidad mayor al frontend de la aplicación.

24. Presupuesto

Los costes detallados del TFM son los siguientes:

- Equipo humano: al ser un único miembro el que realiza las funciones y tareas, el cálculo de tiempo se calcula por persona/hombre, siendo además dicha persona trabajador a jornada completa (y con responsabilidad dentro de la empresa fuera de horario laboral) y padre de familia, el tiempo empleado en el TFM no ha podido ser el deseado, siendo este de una dedicación media de 2 horas por día de lunes a viernes (a veces incluso menos), y de ¾ horas los fines de semana.

- Equipamiento técnico: el TFM se ha realizado sobre un portátil del autor y con software libre o con licencia de estudiante, con lo que el gasto económico técnico es cero.

- Otros recursos: un panel kanban, post-it y rotuladores, siendo el gasto total de unos 10 euros.

25. Análisis de mercado

El proceso para realizar un análisis de mercado se puede dividir en tres partes:

- Entender las condiciones del mercado, esto te da información básica acerca del mercado completo, el tamaño, la competencia, los clientes, etc. Una vez se verifica la necesidad potencial de ofertar el producto, el momento parece satisfactorio para ofertar la aplicación y el volumen de clientes parece bastante aceptable en una primera versión de la aplicación.

- Identificar las oportunidades de mercado, esto te da una información más específica acerca de los problemas potenciales u oportunidades en un mercado objetivo, esto incluye información sobre crecimiento, tendencias actuales y futuras, factores externos y más información sobre sus competidores. El mercado actual es bastante propicio porque cada vez más gente hace uso de la web (y los móviles) para buscar todo tipo de cosas, y esto incluye los productos que les ofrece la aplicación.

- Desarrollar estrategias dirigidas a un mercado, aquí es en donde la investigación de mercado marca el camino. Ayudará a encontrar las oportunidades de crecimiento para el negocio. Entendiendo el mercado y conociendo las oportunidades que se encuentran disponibles, se puede crear una estrategia que te posiciona por encima de otros competidores. Aquí el ajuste de precios y las ofertas pueden ser un factor determinante para estar un nivel por encima del resto.

26. Viabilidad

Un posible estudio de viabilidad del producto sería uno en el cual se tuviera una respuesta positiva y concreta sobre las siguientes preguntas:

- **¿Existe un nicho de mercado?**
  Sí, porque no existe una oferta similar de estas características y las que existen parecidas no cumplen con los objetivos y necesidades que plantea esta..

- **¿Existe un hueco de mercado?**
  Sí, porque no hay oferta del producto y servicio que pretendemos ofrecer para cubrir la demanda. Es más a futuro, se podrían ampliar las opciones a ofrecer dentro de la aplicación, siendo posible un reescalado de mercado y el acceso a otros nichos del mismo.

- **¿Qué volumen potencial tiene el mercado?**
  Es necesario conocer a los clientes potenciales, pero una vez se realizó el estudio con los usuarios se prevé una gran acogida y un diverso tipo de clientes potenciales que hagan uso del producto.

- **¿Cómo reaccionan nuestros competidores?**
  Debe preverse cómo reaccionarán nuestros competidores. Debe procederse a identificarlos, a los actuales y a los futuros, conocer sus resultados, su cuota de mercado, su posible comportamiento en el futuro y sus puntos fuertes y débiles.

- **¿Es un mercado que crece o decrece?**
  Está en desarrollo puesto que es un mercado novedoso y que pueden dar lugar a la generación de beneficios en futuras nuevas versionas cuando se lleguen a acuerdos con más empresas que se ofrezcan en la web.

- **¿Cuánto estarían dispuestos a pagar por el producto o servicio los clientes potenciales?**

Esta pregunta es fundamental para elaborar el plan financiero pero pensando en el futuro, porque las primeras versiones saldrán bajo el paraguas de una aplicación gratuita, siendo a posteriori cuando la acogida sea buena, obtener los ingresos de diversas fuentes, siendo el cliente principal una de ellas pero no la única.

27. Proyección a futuro y conclusión/-es

El proyecto tiene una ambición de crecimiento que se pretende abordar con el paso del tiempo para poder mejorar y poder ser relevante a medio plazo.

En futuras entregas se quiere introducir mejoras visuales en cada una de las pantallas para ir introduciendo, por ejemplo, imágenes, y una mejor experiencia de usuario. Además también sería necesario realizar un frontend interno para la introducción de datos mediante una interfaz que se ofertara a los proveedores y así poder introducir ellos mismos sus productos. Otro punto interesante es poder almacenar más información de clientes mediante “taggeos” introducidos en la web para ver los comportamientos de la gente de cara a ofertarles productos que se asemejen a sus gustos, en fin, que las posibilidades son muy amplias y todo dependerá del tiempo dedicado a la mejora de la aplicación.

Otro punto fuerte también es la utilización de funciones propias de móvil para ser más accesible por los smartphones de los clientes.

El autor en este punto quiere manifestar su absoluta satisfacción con el trabajo realizado, y si bien, no ha podido conseguir los objetivos planteados en primera instancia, sobre todo debido al poco tiempo dispuesto para realizar el trabajo, se podido obtener un producto en una primera versión aceptable y válido.

La experiencia que ha obtenido realizando tanto el desarrollo de la aplicación como de la memoria ha sido muy interesante y lucrativa, tanto por el software como por la memoria, siendo esta, algo que el autor nunca había abordado con tanta profundidad y complejidad.

Como parte final del Máster, este trabajo ha supuesto un cierre muy interesante para todo el progreso realizado desde los comienzos hasta este punto, debido a que todo lo aprendido es algo que en un futuro a corto plazo le será muy útil en sus planes de futuro tanto personales como profesionales.

Anexo 1. Entregables del proyecto

La lista de archivos es la siguiente:

- **Aplicación**: Directorio crm-clients con todo el contenido de la aplicación, a nivel general, los directorios principales de la aplicación son los siguientes:

  - *Directorio raíz*, donde se encuentran todos los ficheros de arranque de la aplicación y los de configuración para utilizar las librerías de terceros.
  - *app*, incluye toda la parte frontend de la aplicación.
  - *config*, incluye las configuraciones para los diferentes entornos de ejecución.
  - *lib*, incluye toda la estructura de la arquitectura backend.
  - *tests*, incluye la estructura de ficheros necesaria para los tests unitarios.

- Ficheros **JSON** con los datos para la base de datos **mongoDB**:

  - *clients.json*: fichero de clientes
  - *events.json*: fichero de eventos
  - *offers.json*: fichero de ofertas
  - *places.json*: fichero de lugares

- **Memoria**: el documento presente con toda la información de la aplicación.

- **Presentaciones** tanto en **vídeo** como en **diapositiva (PPT)**: para mostrar de manera concisa y resumida las partes más importantes del trabajo realizado.

Anexo 2. Código fuente (extractos)

Como partes de ficheros más importantes de la aplicación tenemos las siguientes:

- *app.js*, fichero principal de la parte backend, sobre él se hace uso de todas las demás funciones incluidas en el resto de ficheros:

```
function configureExpress(callback) {
  app.set('port', port);
  app.use(morgan('dev'));
  app.use(favicon(__dirname + '/../app/favicon.ico'));
  app.use(serveStatic(path.join(__dirname, '/../app')));
  app.use(methodOverride());
  app.use(bodyParser.json());      // to support JSON-encoded bodies
  app.use(bodyParser.urlencoded({extended:true}));   // to support URL-encoded bodies
  app.use(cors()); //to support cross-domain
  app.use(helmet());
  //app.use('/client', expressJwt({secret: config.secretKey}));
  // Express in Development Mode
  if ('dev' != app.get('prod')) {
    app.use(errorHandler());
  }
  console.log("Express has been configured.");
  callback();
}
function configureMiddlewares(callback) {
  mongo.connect(config, function(err,db) {
    if(err) {
      console.log('ERROR initializing MongoDB: '+err);
      callback(err);
```

```
        } else {
            app.db = db;
        }
    });
    console.log("Middlewares has been configured.");
    callback();
}
function configureRoutes(callback) {
    fs.readdir('./lib/routes', function (err, files) {
        if(err){
            console.error("No routes folder found ",err);
            process.exit(-1);
        }
        else{
            var isRoute = /(route)(\.js)$/;
            async.eachSeries(files, verifyFile, callback);
        }
        function verifyFile(file, callback){
            if(isRoute.test(file)){
                var route = require('./routes/'+file);
                route.loadRoutes();
                if(callback) {
                    console.log("Routes has been configured.");
                    callback();
                }
            }
            else{
                console.log("Routes has been configured.");
                callback();
            }
        }
    });
```

```
}
```

- *index.html*, fichero principal de la parte frontend de la aplicación, en él se importan el resto de componentes web de la aplicación para hacer uso de ellos, generándose la web de tipo SPI:

```
    <!-- Main Area -->
  <paper-scroll-header-panel main condenses keep-condensed-header>
   <!-- Main Toolbar -->
   <paper-toolbar id="mainToolbar" class="tall">
    <!-- Application name -->
    <div class="middle middle-container center horizontal layout">
     <div class="app-name">Customer Relationship Manager de Clientes</div>
    </div>
   </paper-toolbar>
   <!-- Main Content -->
   <div class="content">
    <iron-pages attr-for-selected="data-route" selected="{{route}}">
     <section data-route="home">
      <paper-material elevation="1">
       <my-greeting></my-greeting>
      </paper-material>
     </section>
     <section data-route="client-login">
      <paper-material elevation="1">
       <h2 class="page-title"><span>Acceso cliente</span></h2>
      </paper-material>
      <paper-material elevation="1">
       <client-login style="position:relative" user-id="{{userId}}"></client-login>
      </paper-material>
     </section>
```

```
<section data-route="client-form">
  <paper-material elevation="1">
    <h2 class="page-title"><span>Alta de cliente</span></h2>
  </paper-material>
  <paper-material elevation="1">
    <client-form style="position:relative" user-id="{{userId}}"></client-form>
  </paper-material>
</section>
<section data-route="my-client">
  <paper-material elevation="1">
    <h2 class="page-title"><span>Cliente</span></h2>
  </paper-material>
  <paper-material elevation="1">
    <my-client style="position:relative" user-id="{{userId}}"></my-client>
  </paper-material>
</section>
<section data-route="offers">
  <paper-material elevation="1">
    <h2 class="page-title"><span>Ofertas</span></h2>
  </paper-material>
  <paper-material elevation="1">
    <my-offers style="position:relative"></my-offers>
  </paper-material>
</section>
<section data-route="places">
  <paper-material elevation="1">
    <h2 class="page-title"><span>Lugares</span></h2>
  </paper-material>
  <paper-material elevation="1">
    <my-places style="position:relative"></my-places>
  </paper-material>
</section>
```

```
        <section data-route="events">
          <paper-material elevation="1">
            <h2 class="page-title"><span>Eventos</span></h2>
          </paper-material>
          <paper-material elevation="1">
            <my-events style="position:relative"></my-events>
          </paper-material>
        </section>
      </iron-pages>
    </div>
  </paper-scroll-header-panel>
```

- *clients.route.js*, es uno de los ficheros que generan las rutas para el API Rest y las operaciones CRUD relacionadas con la entidad Cliente:

```
/*
 * Module Dependencies
 */
var jwt    = require('jsonwebtoken'); // used to create, sign, and verify tokens

var app = require('../app');
var config = require('../../config/env');
var clientsController = require('../controllers/clients.controller');

module.exports = {
   loadRoutes : loadRoutes
};

function loadRoutes() {
   var url;
```

```
getAll();
getClient();
client();

function getAll() {
  url = '/clients';
  app.get(url, function(req, res) {
    clientsController.getClients(done);

    function done(err,data) {
      if(err) {
        res.status(500).send({error: err});
      } else {
        /*var token = jwt.sign(data, config.secretKey, {
          expiresIn: 1440 // expires in 24 hours
        });
        data.token=token;*/
        res.status(200).send(data);
      }
    }
  });
  console.log('   - Registered URL '+url+' and METHOD GET ');
}

function getClient() {
  url = '/client/:id';
  app.get(url, function(req, res) {
    clientsController.getClient(req.params.id,done);

    function done(err,data) {
      if(err) {
```

```
          res.status(500).send({error: err});
        } else {
          res.status(200).send(data);
        }
      }
    });
    console.log('    - Registered URL ' + url + ' and METHOD GET ');
}

function client() {
    url = '/client';
    app.route(url).put(function(req,res) {
      clientsController.putClient(req.body,done);
        function done(err,data) {
          if(err) {
            res.status(500).send({error: err});
          } else {
            res.status(200).send(data);
          }
        }
    }).post(function(req,res) {
        clientsController.postClient(req.body,done);
        function done(err,data) {
          if(err) {
            res.status(500).send({error: err});
          } else {
            res.status(200).send(data);
          }
        }
    }).delete(function(req,res) {
        clientsController.deleteClient(req.body,done);
        function done(err,data) {
```

```
            if(err) {
              res.status(500).send({error: err});
            } else {
              res.status(200).send(data);
            }
          }
        });
        console.log('    - Registered URL ' + url + ' and METHODS PUT, POST AND DELETE');
    }
}
```

Anexo 3. Librerías/Código externo utilizado

Debido a que node.js es un plataforma que dispone de muchas librerías externas, es lógico que se haga uso de las mismas para mejorar la funcionalidad, la seguridad y las buenas prácticas en el desarrollo de software.

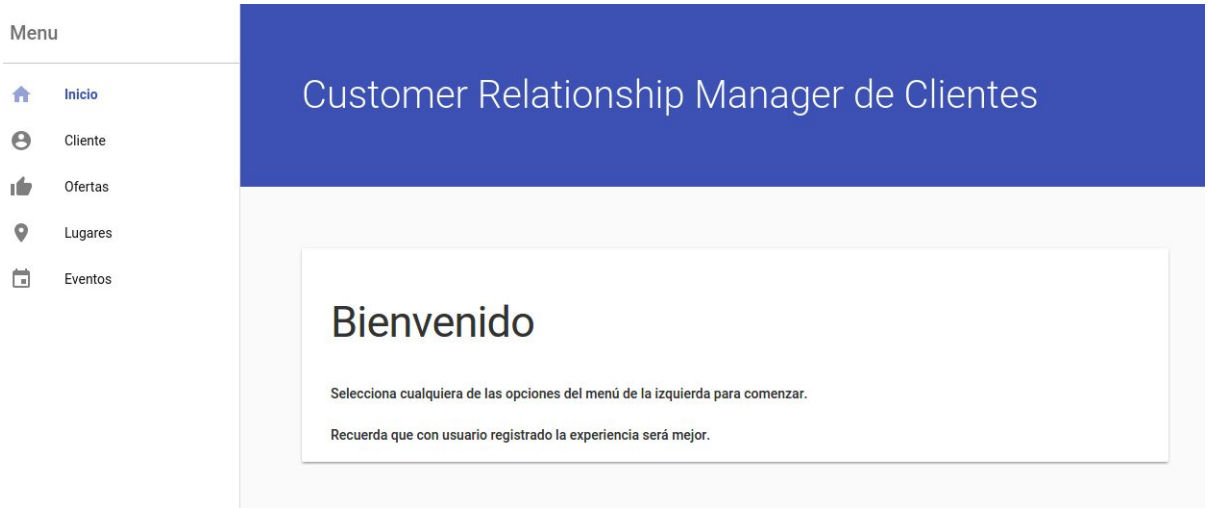
Entre las más importantes utilizadas se encuentran las siguientes (fueron detalladas por separado en el apartado 11):

- **async**
- **express**
- **helmet**
- **mongodb**
- **chai**
- **gulp**
- **mocha**
- **polymer**

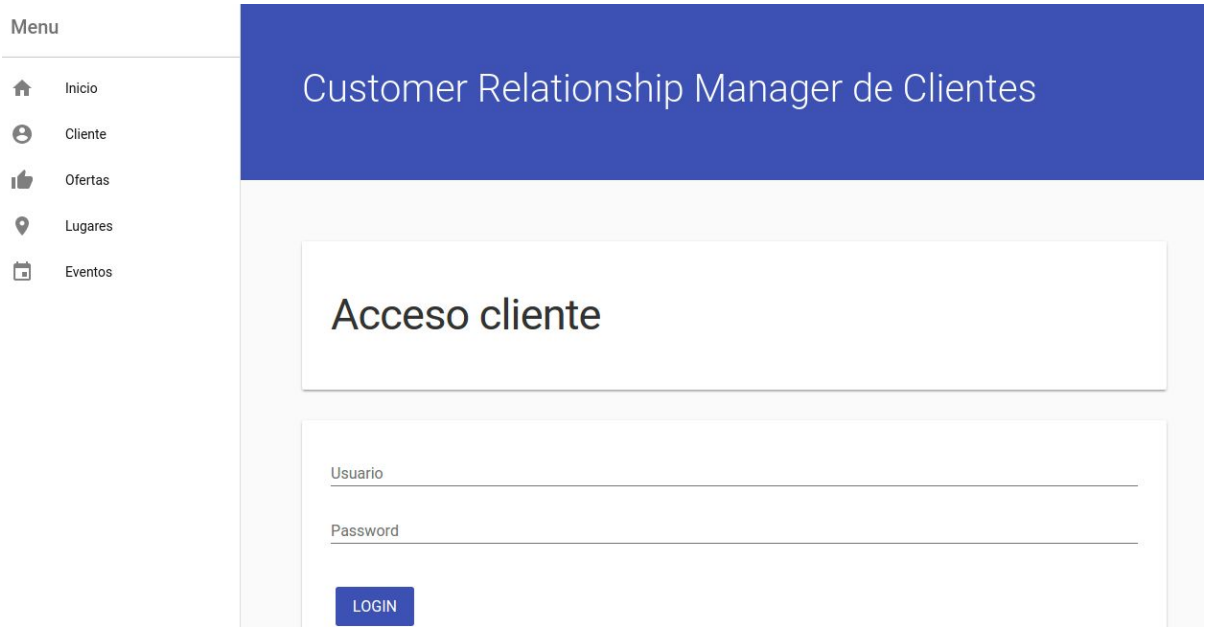
Anexo 4. Capturas de pantalla

En la aplicación encontramos las siguientes:

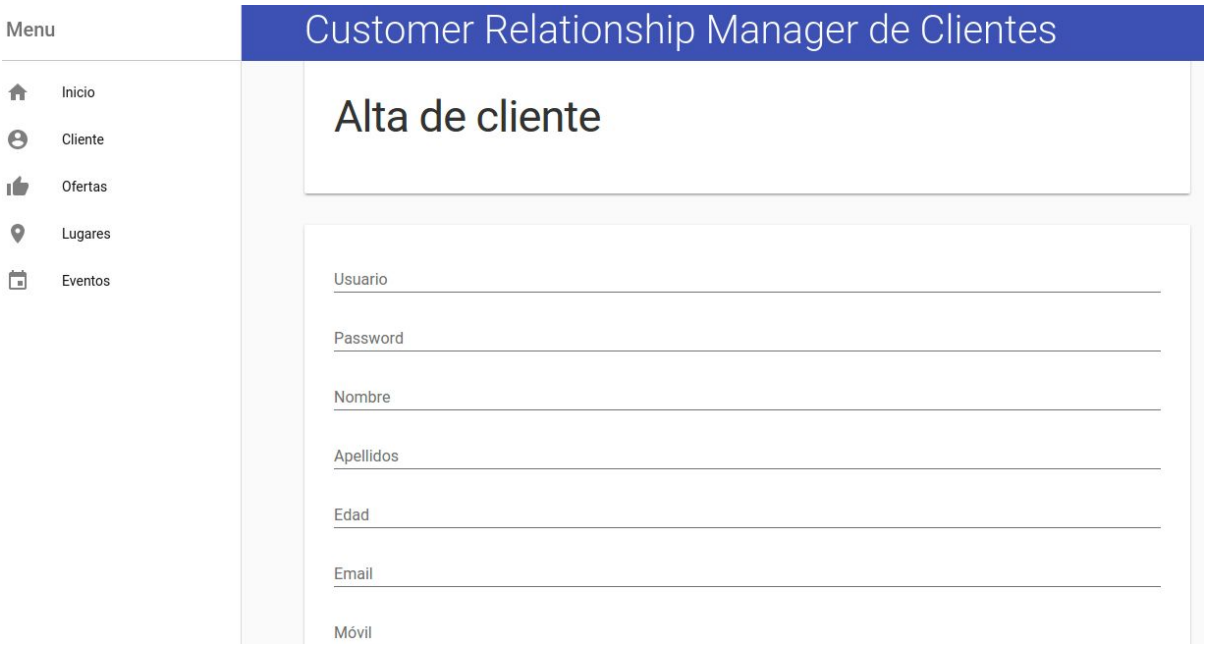
- Pantalla inicio:



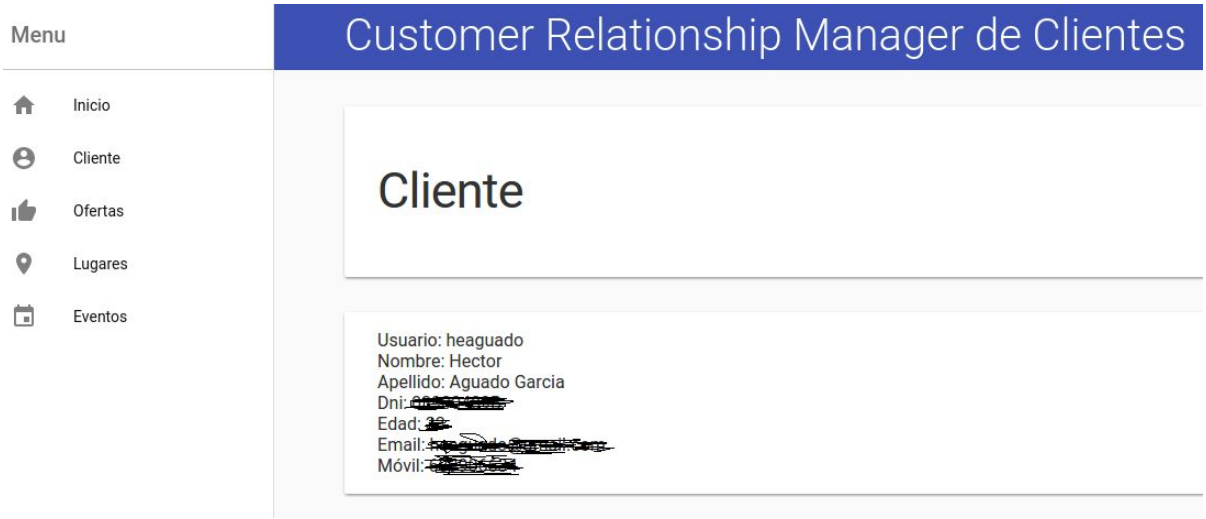
- Pantalla Login cliente:



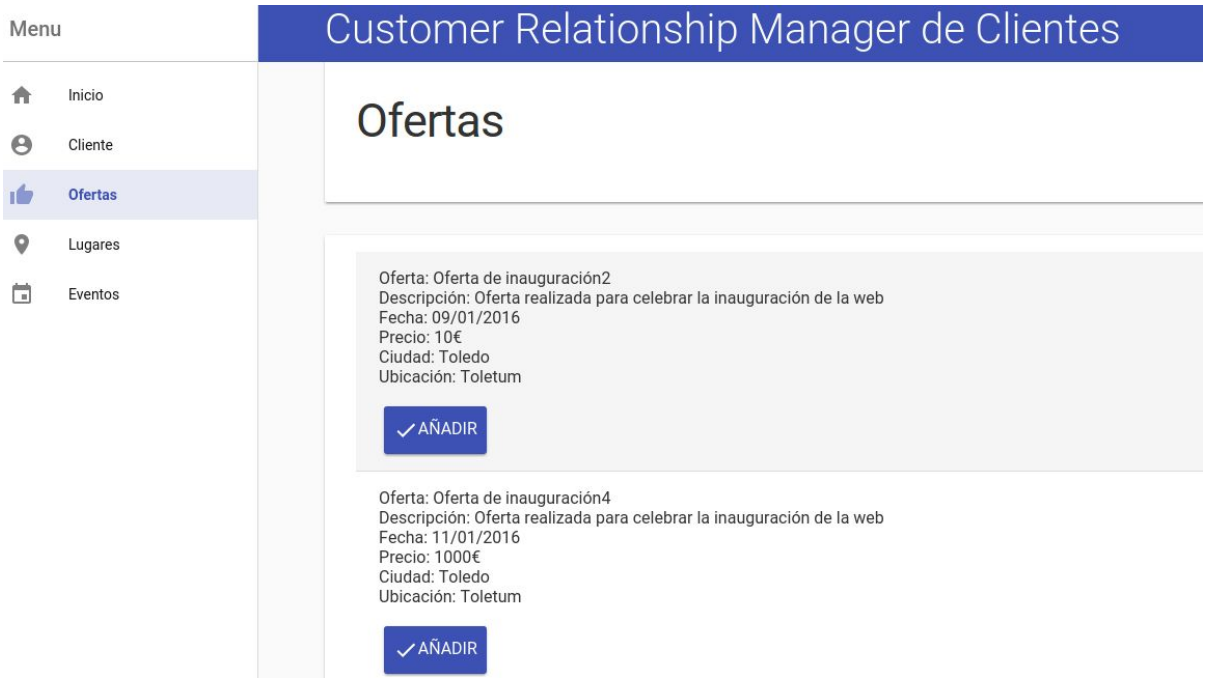
- Pantalla Registro cliente:



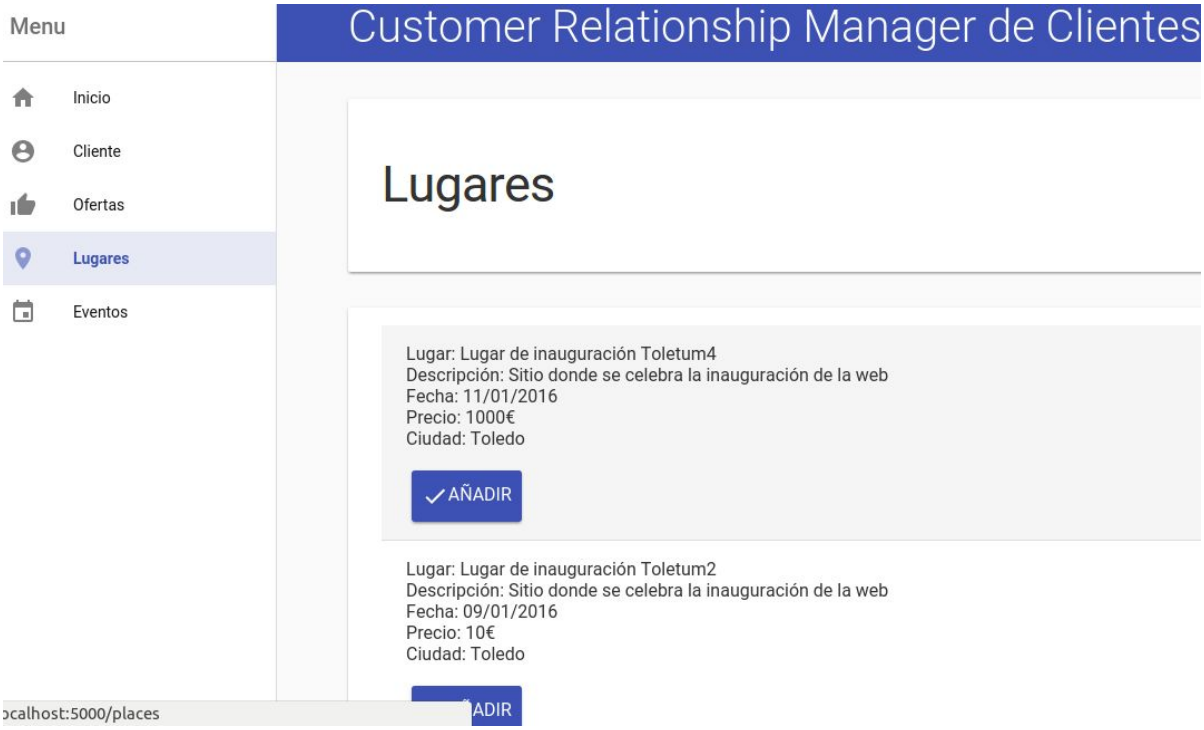
- Pantalla Cliente:



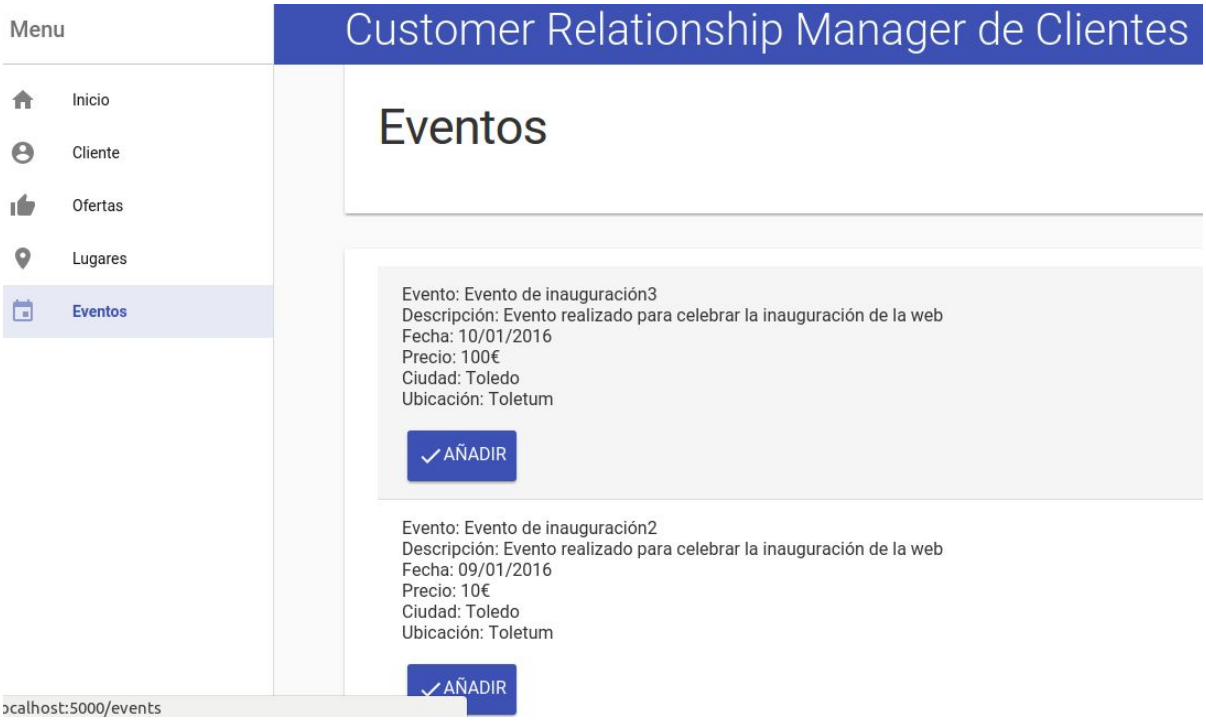
- Pantalla Ofertas:



- Pantalla Lugares:



- Pantalla Eventos:


Menu
Customer Relationship Manager de Clientes
Inicio
Cliente
Ofertas
Lugares
Eventos
Eventos
Evento: Evento de inauguración3
Descripción: Evento realizado para celebrar la inauguración de la web
Fecha: 10/01/2016
Precio: 100€
Ciudad: Toledo
Ubicación: Toletum
AÑADIR
Evento: Evento de inauguración2
Descripción: Evento realizado para celebrar la inauguración de la web
Fecha: 09/01/2016
Precio: 10€
Ciudad: Toledo
Ubicación: Toletum
AÑADIR
localhost:5000/events

Anexo 5. Guía de usuario

Versión	Autor
1.0 (Inicial)	Héctor Aguado García

El presente documento explica cómo utilizar la aplicación correctamente.

La aplicación en esta primera versión es bastante fácil de utilizar, debido a que no contiene funcionalidad más allá de una serie de consultas, el alta de cliente y el login de cliente.

- Pantalla Inicial: El usuario accede a una pantalla en primera instancia con descripción de la web.

- Menú/pantalla cliente: el usuario pulsa en la opción de menú cliente y le aparece la pantalla de login, donde puede llevar a cabo dos opciones:

    - Login de cliente: una vez introducidos los datos del cliente dado de alta anteriormente correctamente se accede a la pantalla de información del cliente.

    - Alta de cliente: una vez pulsado el link se accede a la pantalla de registro de clientes.

- Pantalla de registro de cliente: el usuario introduce sus datos y pula en el botón de registrar para darse de alta, o en el botón de limpiar para borrar todos los datos si se introdujeron mal.

- Pantalla de información de cliente: el usuario comprueba todos los datos que introdujo en la pantalla de registro y que se recuperan de la base de datos.

- Menú/pantalla ofertas: el usuario puede observar los datos de ofertas que ofrece la web.

- Menú/pantalla eventos: el usuario puede observar los datos de eventos que ofrece la web.

- Menú/pantalla lugares: el usuario puede observar los datos de lugares que ofrece la web.

Anexo 6. One-page business plan/Resumen ejecutivo

El micro-plan de empresa del proyecto es el siguiente:

- **Nombre comercial:** Customer Relationship Management de Clientes
- **Resumen comercial:** Aplicación web utilizada para la gestión por parte de los clientes, de diferentes planes de ocio, ofertas y eventos que pudieran ser de su interés. Se pretende llegar al mayor número posible de clientes, llegando a acuerdos con la mayor cantidad posible de proveedores para que ofrezcan sus productos.
- **Modelo de negocio:** modelo de fidelización, por el cual se pretende que un cliente realice una gestión y que a partir de esta continúe realizando más, siendo una serie de obtenciones de productos el objetivo a medio plazo.
- **Expertise:** no se cuenta con ninguna experiencia en este ámbito ni en ninguno relacionado, siendo esta la primera aproximación al medio.
- **Productos y servicios:** dentro de la web se ofrecen actualmente tres tipos de productos y servicios; las ofertas, los eventos y los lugares, siendo algunos de ellos complementarios entre sí.
- **Mercado:** la aplicación está destinada a llegar a una amplia variedad de clientes digitales, que hacen uso de aplicaciones web para conseguir oportunidades, descuentos y ofertas sobre una gran variedad de productos.
- **Competencia:** actualmente no hay aplicaciones ni sistemas que ofrezcan la misma funcionalidad ni tipo de servicios, existiendo solamente algunas que ofrecen contenidos parciales respecto a lo ofertado en el producto.
- **Plan de marketing:** se pretende realizar una gran campaña de aceptación entre clientes a través de medios digitales, además del boca a boca entre los clientes para abarcar el mayor número de usuarios y de proveedores de productos.
- **Inversión inicial y costes a corto y medio plazo:** la inversión inicial es prácticamente cero, siendo posible que si la aplicación y el desarrollo a futuro lo permiten se incrementen para poder abastecer todos los cambios propuestos.

- **Proyección económica corto y medio plazo y ROI:** la proyección ecómica que se pretende a corto plazo es baja, siendo a medio plazo donde se quieren obtener beneficios, por lo que al ser la inversión muy poca poder generar rentabilidad en el medio plazo para poder crecer sobre la aplicación y poder generar nuevas funcionalidades y objetivos.
- **DAFO:** en base a dicha matriz, se pueden obtener las siguientes conclusiones: hay que potenciar lo bueno que se ha generado, cuando surjan oportunidades de negocio interesantes hay que aprovecharlas, los puntos negativos hay que mejorarlos y las amenazas deber ser tratadas con coherencia y a poder ser solventadas de la mejor manera posible.

Anexo 7. Glosario/Índice analítico

- **TFM**: Trabajo de fin de Master.

- **CRM**: Customer Relationship Manager. Es el nombre del proyecto así como unas palabras inglesas muy utilizadas hoy en día y cuya traducción es Gestor de relaciones con el cliente.

- **API**: Application Programming Interface. Es el conjunto de métodos y funciones ofrecidas por una librería para poder ser utilizados por otras aplicaciones como una capa de abstracción.

- **REST**: Representational Estate Transfer. Describe cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, sus aspectos clave son: Un protocolo cliente/servidor sin estado, un conjunto de operaciones bien definidas que se aplican a todos los recursos de información (los más importantes GET, POST, PUT y DELETE), una sintaxis universal para identificar los recursos y el uso de hipermedias.

- **JSON**: Javascript Object Notation. Es el formato más utilizado hoy en día para intercambio de datos.

- **SPI**: Single Page Interface. Formato de web muy utilizado en las webs moderna y que deja el control de la navegación a la parte frontend de la misma.

- **IDE**: Integrated Development Environment. Es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

- **BDD**: Behaviour Driven Development. Desarrollo orientado por comportamiento, combina las técnicas generales y los principios del desarrollo guiado por pruebas, junto con ideas del diseño guiado por el dominio y el análisis y diseño orientado a objetos para proveer al desarrollo de

software y a los equipos de administración, con herramientas compartidas y un proceso compartido de colaboración en el desarrollo de software.

- **TDD**: Test Driven Development. Es una práctica de ingeniería de software que involucra otras dos prácticas: escribir las pruebas primero (Test First Development) y refactorización (Refactoring).
- **DCU**: Diseño centrado en usuario. Es una filosofía de diseño que tiene por objeto la creación de productos que resuelvan necesidades concretas de sus usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo de su parte.

Anexo 8. Bibliografía

https://github.com/caolan/async Página 24

http://expressjs.com/ Página 24

https://github.com/helmetjs/helmet Página 24

https://www.mongodb.org/ Página 24

http://chaijs.com/ Página 24

http://gulpjs.com/ Página 24

https://mochajs.org/ Página 24

https://www.polymer-project.org/1.0/ Página 24

https://elements.polymer-project.org/browse?package=iron-elements Página 25

https://elements.polymer-project.org/browse?package=paper-elements Página 25

https://github.com/helmetjs/helmet Página 44

https://nodejs.org/en/download Página 50

https://www.mongodb.org/downloads Página 50

https://jwt.io/ Página 55

Anexo 9. Vita

Héctor Aguado García, reside en la actualidad en Toledo. Desde su juventud siempre ha sido un amante de la tecnología, siendo todo lo relacionado con la informática su pasión. Estudió Ingeniería Informática en la Universidad Pontificia de Salamanca en Madrid. Sus comienzos en el mundo del desarrollo del software fueron en el 2007 como Programador Junior, avanzando posteriormente su carrera pasando por diferentes categorías profesionales, siendo actualmente Arquitecto de Software, y participando en proyectos relacionados con el Big Data y el Cloud.