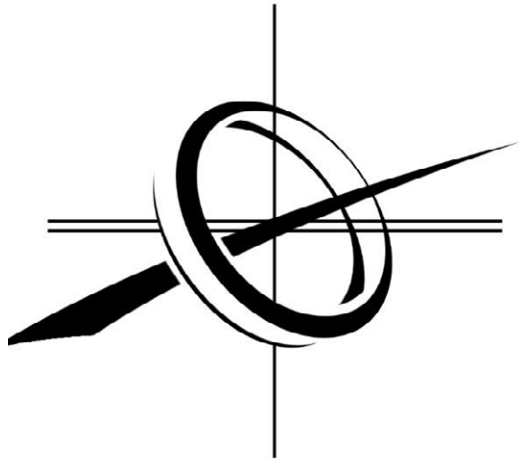


**UNIVERSIDAD POLITÉCNICA DE MADRID**

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
SISTEMAS INFORMÁTICOS  
GRADO EN INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

**Aplicación Web de Gestión Financiera Personal**  
**Desarrollada con MEAN.JS**



**Autor: Ignacio Aldarabi Carrillo**

**Curso Académico: 2015/2016**

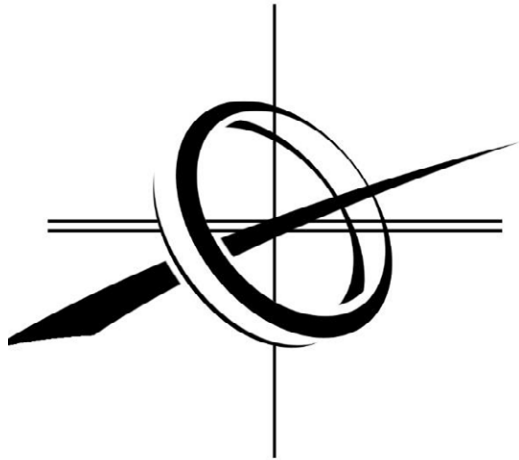


**UNIVERSIDAD POLITÉCNICA DE MADRID**

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
SISTEMAS INFORMÁTICOS  
GRADO EN INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

**Aplicación Web de Gestión Financiera Personal**  
**Desarrollada con MEAN.JS**



**Autor: Ignacio Aldarabi Carrillo**

**Tutor: Fernando Ortega Requena**

**Curso Académico: 2015/2016**



Agradezco a mis padres todo el esfuerzo realizado,  
sin ellos nunca habría sido posible llegar hasta aquí.  
Tampoco me olvido de mis personas más cercanas, mis amigos  
y pareja los cuales me hacen tan ameno el día a día.



## RESUMEN

En este trabajo de fin de grado se llevará a cabo la elaboración de una aplicación web de gestión de gastos personales desde sus inicios, hasta su completo funcionamiento.

Estas aplicaciones poseen un crecimiento emergente en el mercado, lo cual implica que la competencia entre ellas es muy elevada. Por ello el diseño de la aplicación que se va a desarrollar en este trabajo ha sido delicadamente cuidado. Se trata de un proceso minucioso el cual aportará a cada una de las partes de las que va a constar la aplicación características únicas que se plasmarán en funcionalidades para el usuario, como son: añadir sus propios gastos e ingresos mensuales, confeccionar gráficos de sus principales gastos, obtención de consejos de una fuente externa, etc...

Estas funcionalidades de carácter único junto con otras más generalistas, como son el diseño gráfico en una amplia gama de colores, harán su manejo más fácil e intuitivo. Hay que destacar que para optimizar su uso, la aplicación tendrá la característica de ser responsive, es decir, será capaz de modificar su interfaz según el tamaño de la pantalla del dispositivo desde el que se acceda.

Para su desarrollo, se va a utilizar una de las tecnologías más novedosas del mercado y siendo una de las más revolucionarias del momento, MEAN.JS. Con esta innovadora tecnología se creará la aplicación de gestión económica de gastos personales.

Gracias al carácter innovador de aplicar esta tecnología novedosa, los retos que plantea este proyecto son muy variados, desde cómo estructurar las carpetas del proyecto y toda la parte de backend hasta como realizar el diseño de la parte de frontend.

Además una vez finalizado su desarrollo y puesta en marcha se analizarán posibles mejoras para poder perfeccionarla en su totalidad.

## ABSTRACT

In this final degree project will take out the development of a web application from its inception, until its full performance management.

These applications have an emerging market growth, implying that competition between them is very high. Therefore the design of the application that will be developed in this work has been delicately care. It's a painstaking process which will provide each of the parties which will contain the application unique features that were translated into functionality for the user, such as: add their own expenses and monthly income, make graphs of your major expenses, obtaining advice from an external source, etc...

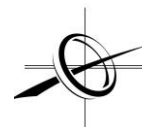
These features of unique character together with other more general, such as graphic design in a wide range of colors, will make more easy and intuitive handling. It should be noted that to optimize its use, the application will have the characteristic of being responsive, will be able to modify your interface according to the size of the screen of the device from which are accessed.

For its development, it is to use one of the newest technologies on the market and being one of the most revolutionary moment, MEAN. JS. The economic management of personal expenses application will be created with this innovative technology.

Thanks to the innovative nature of applying this new technology, the challenges posed by this project are varied, from how to structure the folders of the project and all the backend part up to how to perform the part of frontend design.

In addition once finished its development and commissioning possible improvements will analyze to be able to perfect it in its entirety.





## ÍNDICE

RESUMEN .....	7
ABSTRACT .....	8
1. INTRODUCCIÓN .....	3
2. OBJETIVOS .....	9
2.1. MOTIVACION .....	9
2.2. OBJETIVOS ESPECÍFICOS .....	10
2.3. ESTRUCTURA DEL DOCUMENTO .....	11
3. APLICACIONES DE GESTIÓN DE GASTOS PERSONALES .....	13
3.1 FINTONIC .....	14
3.2 MONEFY .....	18
4. ENGINSAVE .....	21
4.1 REQUISITOS DE ENGINSAVE .....	21
4.2 RESULTADO FINAL DE LA APLICACIÓN .....	24
4.1 ACCESO A LA APLICACIÓN .....	25
4.2 INICIO .....	28
4.3 MOVIMIENTO .....	29
4.4 CALENDARIO .....	31
4.5 GRÁFICOS .....	32
4.6 PERFIL .....	34
4.7 CONSEJOS .....	35
5. TECNOLOGÍAS UTILIZADAS .....	37
5.1 MEAN.JS .....	37
5.2 NODE.JS .....	37
5.3 EXPRESS.JS .....	43
5.4 ANGULAR.JS .....	44
5.5 MONGO DB .....	48
6. ESTRUCTURA DE LA APLICACIÓN .....	51
7. PUESTA EN PRODUCCIÓN DE LA APLICACIÓN .....	59



---

7.1 ¿QUÉ ES OPENSIFT?.....	59
7.2. CONFIGURACIÓN DE UNA APLICACIÓN DE NODE CON OPENSIFT .....	59
8. FUTURAS MEJORAS DE LA APLICACIÓN .....	67
9. CONCLUSIONES .....	69
BIBLIOGRAFÍA .....	71

## 1. INTRODUCCIÓN

En un principio internet era sencillamente una colección de páginas estáticas y documentos para consultar o descargar. El paso inmediatamente posterior en su evolución fue la inclusión de un método para elaborar páginas dinámicas que permitieran que lo mostrado tuviese carácter dinámico (es decir, generado a partir de los datos de la petición).

La gran velocidad de avance del crecimiento de internet hizo que se desarrollan diferentes herramientas para optimizar su uso, como son las aplicaciones web y las aplicaciones nativas.

Las aplicaciones nativas son aquellas que están diseñadas y desarrolladas para un sistema operativo específico, estas deben ser descargadas e instaladas en los dispositivos.

Las aplicaciones web, son herramientas que no necesitan ser instaladas en los dispositivos, ya que es en los propios navegadores desde donde se ejecutan, estas son multiplataforma y no es necesaria su descarga ni instalación en ningún dispositivo.

Este trabajo se centrará en el desarrollo de una aplicación web.

Las ventajas de las aplicaciones web son muy variadas:

- Se ahorra tiempo a la hora de usarlas, ya que la aplicación web no hay que instalarla sino que se ejecuta al momento desde el navegador.
- No ocupan casi espacio en el dispositivo del usuario final ya que no es necesaria la instalación ni descarga de la herramienta.
- Están permanentemente actualizadas, ya que solo existe la versión que esté subida a internet.
- Son multiplataforma, pueden ser utilizadas en cualquier sistema operativo que tenga un navegador.

Sus inconvenientes más destacados son que requieren una conexión a internet para poder ser utilizadas y que suelen ofrecer un menor número de funcionalidades que las aplicaciones convencionales.

En cuanto a la arquitectura que siguen, están compuesta por dos partes, la parte del cliente y la parte del servidor, ambas unidas por internet. En la figura que a continuación se muestra (Figura 1) se puede observar el esquema de una aplicación web básica.

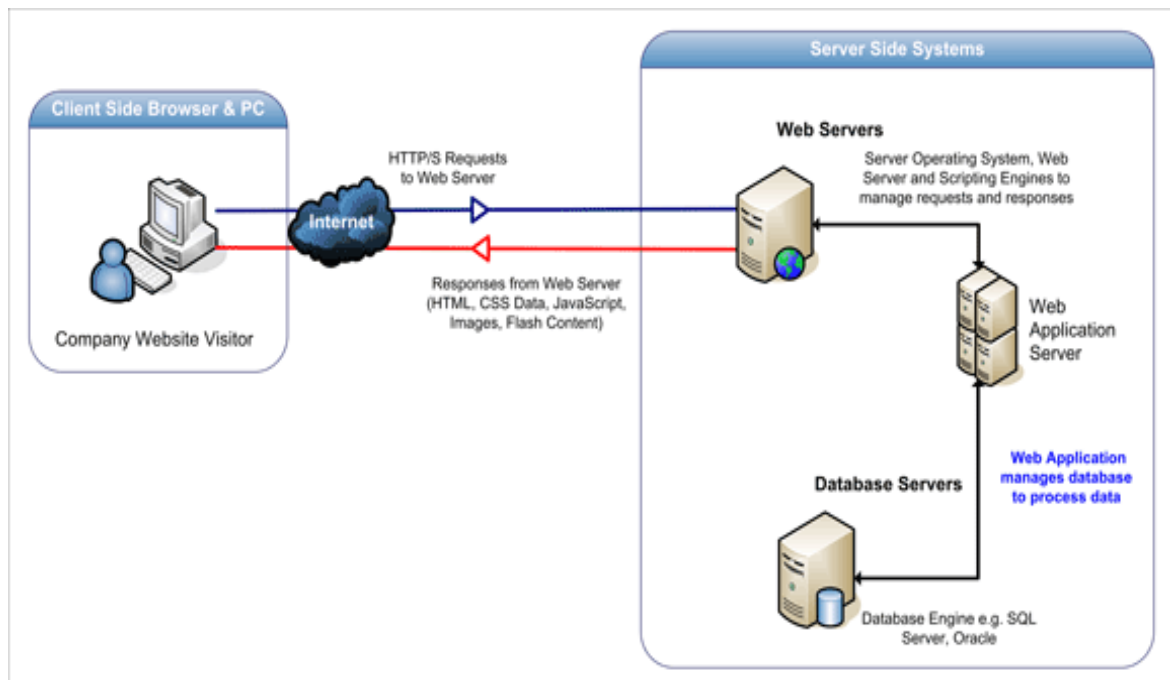


Figura 1, Esquema del funcionamiento de una aplicación web

Tal y como puede visualizarse en el esquema de la anterior figura el funcionamiento de una aplicación web sigue unos determinados pasos: un usuario realiza una petición HTTP a un servidor en el cual está alojada una aplicación web, en este se maneja las peticiones y se carga el contenido de la aplicación combinándolo con los datos obtenidos a través de su base de datos, posteriormente se manda la respuesta al usuario enviándole todo el contenido de la herramienta.

Estas aplicaciones web están desarrolladas con un lenguaje de programación propio, los más relevantes usados en backend se detallan a continuación:

- **PHP:** es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que

procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante.

- **Java:** es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.
- **JavaScript** (el cual va a ser utilizado en el presente trabajo): es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado, dinámico y orientado a eventos. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.
- **Perl:** es un lenguaje de propósito general originalmente desarrollado para la manipulación de texto y que ahora es utilizado para un amplio rango de tareas incluyendo administración de sistemas, desarrollo web, programación en red, desarrollo de GUI y más.
- **Ruby:** es un lenguaje con un balance cuidado. Su creador, Yukihiro "Matz" Matsumoto, mezcló partes de sus lenguajes favoritos (Perl, Smalltalk, Eiffel, Ada y Lisp) para formar un nuevo lenguaje que incorporara tanto la programación funcional como la imperativa.

- **Python:** es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

En cuanto a la parte de frontend, los lenguajes más utilizados son JavaScript, HTML, el lenguaje que se encarga de la parte de maquetado de la página y CSS, el lenguaje que dota a la estructura HTML creada de características visuales a través de hojas de estilo.

En la siguiente gráfica (Figura 2 [9]) se puede observar la tendencia de uso de los lenguajes de programación durante los últimos años.

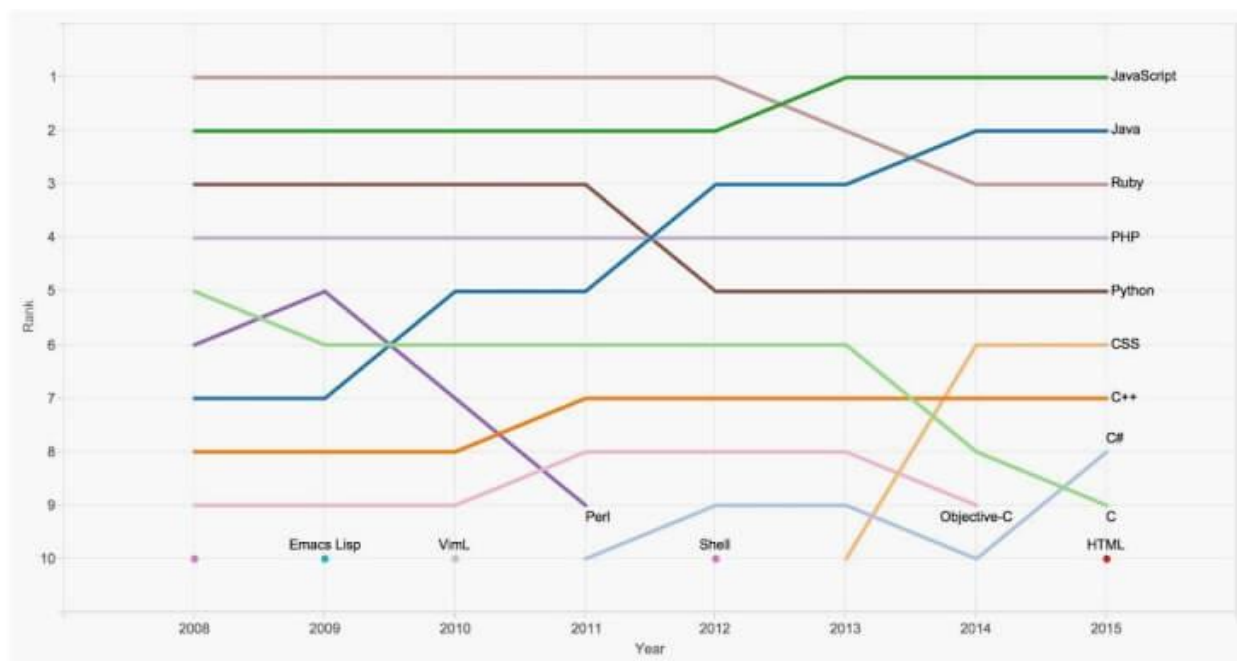
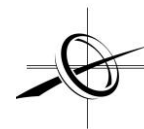


Figura 2, Tendencia de uso de los lenguajes de programación

Como se puede observar en la gráfica, JavaScript lidera el ranking de los lenguajes de programación y java le sigue con una progresión constante en los años. Según esta representación los lenguajes que lideran el ranking son aquellos que están orientados a la programación web.



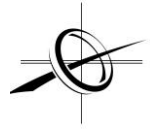
Gracias a esta gran versatilidad de las aplicaciones web, se han desarrollado muchas y muy variadas, tanto en temática a la que van dirigidas, como en los aspectos técnicos utilizados para desarrollarlas.

Dentro de esta enorme variedad de temáticas, en este trabajo se ha explorado el campo de las aplicaciones financieras y más concretamente las aplicaciones de gestión de ahorros personales que tienen un gran campo de actuación debido a la alta demanda por parte del cliente. Esta alta demanda ha surgido principalmente debido a la situación general de la economía. En un lenguaje generalista se puede afirmar que los desarrolladores han aprovechado y han implementado multitud de herramientas con este fin, entre ellas aplicaciones de ahorro en compras por internet, ahorro energético, ahorro en los gastos de un vehículo o la gestión financiera.

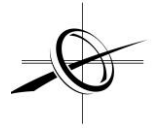
De la importancia de la gestión de la economía personal para tener un mayor control de los gastos ha surgido la necesidad de crear una aplicación web que sea capaz de convertir unos simples datos numéricos proporcionados por el usuario, en recursos visuales y gráficos que hagan más entendible la evolución de sus gastos e ingresos. Pudiendo así mejorar la gestión propia de los bienes.

Por ello, en este trabajo se va a desarrollar una aplicación web de gestión financiera personal, que ayude a sus usuarios a mantener un control en la economía diaria.

Como se ha mencionado anteriormente, el mercado de las aplicaciones de gestión financiera es variado y competitivo. Al haber mucha demanda de usuarios, la creación de una aplicación óptima para el uso al que va destinada se hace más difícil. Esta competitividad también será un reto que se afronta al diseñar el trabajo fin de grado presente, se aunarán esfuerzos para crear una aplicación que sea ventajosa y competitiva en el mercado, que con vistas al futuro pueda destacar como una de las aplicaciones más novedosas y útiles para el amplio público objetivo.







## 2. OBJETIVOS

El objetivo principal de este trabajo es el aprendizaje sobre el funcionamiento y los métodos de construcción de aplicaciones web, a través de la profundización en el conocimiento y la familiarización de la tecnología full-stack MEAN.JS para la creación de una aplicación web que gestione los gastos e ingresos de uno o varios usuarios.

### 2.1. MOTIVACION

Al afrontar este trabajo fin de grado las temáticas a abordar eran múltiples y variadas, la principal motivación personal por la cual decidí realizar el desarrollo de una aplicación de gestión financiera fue el interés por la tecnología emergente MEAN.JS y el ámbito de programación web. El desconocimiento sobre este ámbito es un dato que ayuda a alimentar mi interés sobre la misma, ya que en los años de Grado en Ingeniería de Software no he cursado ninguna asignatura que impartiera estas temáticas.

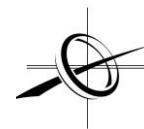
La creación de una aplicación web con un mismo lenguaje para todas sus partes me llamó mucho la atención ya que ese concepto todavía no lo había observado en ninguna otra tecnología, lo cual no significa que sea trivial, sino todo lo contrario, he tenido que investigar mucho y dedicar mucho tiempo en el desarrollo de la aplicación.

Por todo esto y gracias a mi tutor, Fernando Ortega, que me dio la oportunidad para realizar este proyecto, me decidí a crear la aplicación web sobre la que se habla en este documento.

## 2.2. OBJETIVOS ESPECÍFICOS

Los objetivos específicos que pretende cubrir el desarrollo de este trabajo son los siguientes:

1. Conocimiento y uso de la base de datos MongoDB. El uso de una base de datos no relacional plantea nuevos puntos de vista, MongoDB es una base de datos orientada a documentos, lo que supone un cambio en la forma de trabajo y amplía las posibilidades de uso.
2. Conocimiento y uso de Node.js. Esta tecnología, al igual que MongoDB, requiere un cambio de visión, ya que su funcionamiento es diferente a las tecnologías actuales del lado del servidor, un ejemplo de ello es su asincronismo, que veremos en detalle más adelante.
3. Conocimiento y uso de Express.js. El conocimiento de esta tecnología va muy de la mano del conocimiento de Node.js, ya que Express.js es un framework web para Node.js que proporciona un conjunto de características para aplicaciones web y móviles.
4. Conocimiento y uso del framework JavaScript Angular.js. Este objetivo incluye también el aprendizaje de HTML, ya que este framework extiende HTML para facilitar la creación de aplicaciones basadas en el patrón MVC (Modelo, Vista, Controlador), también se aprenderán conceptos como los de single page y directivas entre otros.
5. Conocimiento de la estructura y funcionamiento de una aplicación web. Las aplicaciones web, al igual que las aplicaciones convencionales, siguen una estructura de carpetas y una organización específica para que se realice un correcto desarrollo y sea más sencillo añadir o modificar funcionalidades de la aplicación en un futuro. Para desarrollar una aplicación web también se debe conocer el funcionamiento básico de las páginas web.
6. Desarrollo de una aplicación web de gastos personales utilizando MEAN.JS. Con el uso de los conocimientos adquiridos en los objetivos anteriores, se estará preparado para afrontar la aplicación que atañe a este proyecto, llamada ENGINSAVE.



## 2.3. ESTRUCTURA DEL DOCUMENTO

El presente documento se estructura en sus contenidos de acuerdo a los siguientes puntos:

En la sección 3 (Aplicaciones de gestión de gastos personales) se hace un breve repaso sobre la evolución de las aplicaciones financieras y se realizará un análisis de las aplicaciones más significativas que están actualmente a disposición de los usuarios.

En la sección 4 (ENGINSAVE) se describirá en detalle la aplicación desarrollada en este trabajo.

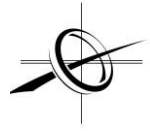
En la sección 5 (Tecnologías utilizadas) se nombrarán y explicarán todas las tecnologías aplicadas en este proyecto.

En la sección 6 (Estructura de la aplicación) se verán aspectos de diseño de la aplicación, los archivos más importantes.

En la sección 7 (Puesta en producción de la aplicación) se mostrará cómo se ha hecho posible la puesta en producción de la aplicación web.

En la sección 8 (Futuras mejoras) se verán los posibles aspectos a mejorar o la nuevas funcionalidades que se podrían implementar para mejorar la aplicación.

En la sección 9 (Conclusiones) se hará una reflexión global sobre lo que personalmente me ha aportado el proyecto y las conclusiones oportunas sobre las tecnologías utilizadas.



### 3. APLICACIONES DE GESTIÓN DE GASTOS PERSONALES

Como se ha comentado anteriormente en este documento, la categoría de aplicaciones financieras es un campo que ha crecido notablemente en los últimos tiempos. Debido a esta gran importancia, este trabajo se centra en las aplicaciones de gastos personales, que son aplicaciones cuyo foco está centrado en el control y administración de los gastos de usuario, de manera que a través del registro de los movimientos, compras y otros gastos, el usuario pueda controlar su economía de forma más eficiente.

Gracias a la gran inclusión en el mercado de las aplicaciones financieras, estas están disponibles tanto de tipo web como de tipo nativo. Para ejemplarizar la importancia que ha adquirido este ámbito de las aplicaciones financieras se muestra a continuación un ejemplo representativo de varios rankings de aplicaciones.

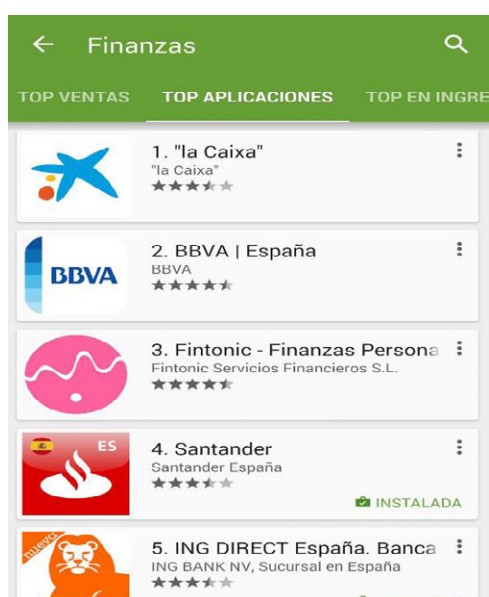


Figura 3, Ranking general de aplicaciones financieras en Play Store

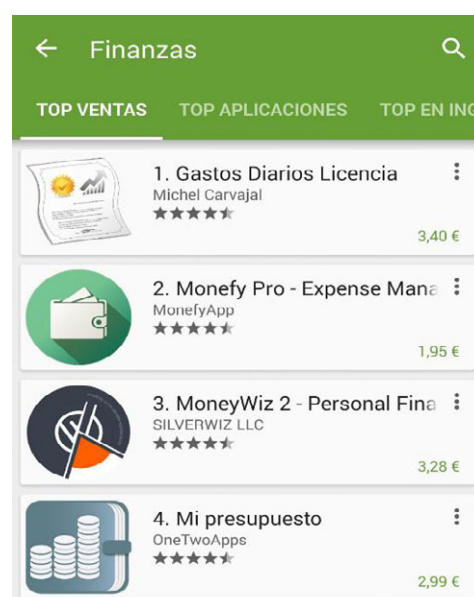


Figura 4, Ranking de aplicaciones de pago financieras en Play Store

Como se puede observar la aplicación Fintonic, que es tanto web como nativa, se sitúa la tercera en el ranking general de aplicaciones financieras (Figura 3), hay que destacar que tanto la primera como la segunda posición están ocupadas por aplicaciones que son propiedad de bancos. Por tanto Fintonic es la que lidera el top de las aplicaciones de gestión de gastos personales. En cuanto al ranking de ventas (Figura 4), el estudio se centra en Monefy Pro porque a pesar de ser la segunda posicionada es la aplicación que más se asemeja a la que se va a desarrollar en este trabajo.

En los siguientes puntos se realizará un análisis de ambas para profundizar en el mercado de las aplicaciones de gestión de gastos personales.

### 3.1 FINTONIC

Fintonic [7] [12] es una herramienta gratuita bastante novedosa, tiene gran éxito debido a su gran control y detalle de los gastos e interfaz muy intuitiva, actualmente está experimentando un gran crecimiento e incluso es promocionada a nivel mundial a través de sus spots publicitarios en televisión.

Fintonic es una aplicación de gestión de gastos personales que consta tanto de aplicación web, como de una aplicación nativa, estos dos hechos convierten a Fintonic en una aplicación multiplataforma, disponible para móvil, tablet y sobremesa.

Algunas de las principales características de Fintonic son las siguientes:

- Centralización de la información de todas las cuentas y tarjetas en un mismo lugar.
- Información simple y organizada.
- Claridad en los movimientos.
- Categorización de los gastos.
- Envíos de alertas y notificaciones sobre movimientos o cargos en las cuentas.

En cuanto a la seguridad:

- Es una aplicación exclusivamente de consulta y puramente informativa, no permite transacciones ni movimientos entre cuentas.
- Para poder descargar los movimientos de cuentas y tarjetas, pide las claves de lectura de acceso a la entidad bancaria del usuario (claves web).

Una de las desventajas que presenta esta aplicación es la necesidad de introducir un gran número de datos sensibles, tales como números de cuenta y números de tarjetas, para poder utilizarla.

En las siguientes ilustraciones se muestra parte de la aplicación anteriormente descrita.

La Figura 5, pertenece a la aplicación desarrollada para móviles, muestra el balance general de un usuario, los detalles de un movimiento y la categorización de un movimiento.

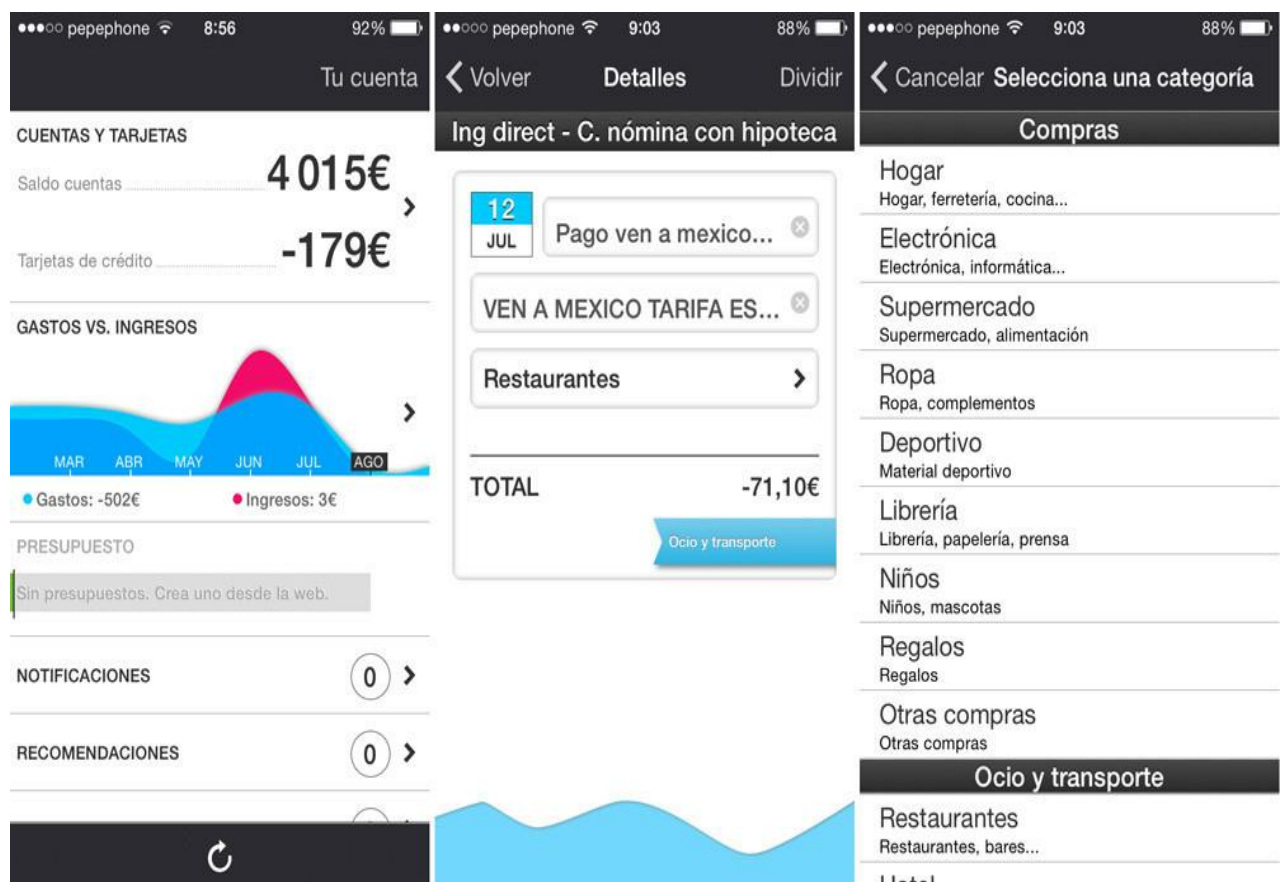


Figura 5, Imágenes de la aplicación FINTONIC en dispositivos móviles.

La Figura 6 **Error! Reference source not found.** muestra los balances y situación general de las cuentas de un usuario desde la aplicación web con un ordenador.



Figura 6, Imagen de la aplicación FINTONIC en su versión de ordenador



En la Figura 7 se puede observar los gastos realizados por categoría en la aplicación web desde un ordenador. Aquí puede extraerse la conclusión de que posee unas interfaces visualmente llamativas, lo cual hace más intuitivo su uso.



Figura 7, Imagen de la aplicación FINTONIC en su versión de ordenador

### 3.2 MONEFY

Monefy [13] [6] es otra aplicación de control de gastos personales, se encuentra en el puesto número 2 del ranking de ventas de aplicaciones financieras en Play Store [17] según datos recogidos el día 08/09/2015, y en este caso es una aplicación nativa en Android y tiene dos modalidades, una de pago y una gratuita

En este caso el usuario no tendrá que introducir ningún tipo de dato sensible, ya que sencillamente el usuario sólo deberá ir introduciendo sus gastos manualmente.

Algunas de sus principales características son las siguientes:

- Es muy sencilla de usar.
- Las categorías de ingresos y gastos se pueden editar (solo en el caso de la versión premium).
- Tiene protección por contraseña.
- Su interfaz es muy sencilla y sigue un diseño flat.
- Incluye sincronización a través de Dropbox.

En las siguientes figuras se muestra una visión general del funcionamiento de la aplicación nativa Monefy. De estas figuras puede extrapolarse la idea de realizar un interfaces intuitivo que ayuda a entender mejor el funcionamiento de la aplicación y poder así optimizar sus recursos. Tal y como se observa en las figuras que siguen, puede verse que el principal objetivo de Monefy al ser diseñada era llegar al cliente de forma clara y basada en inputs de imágenes y colores.

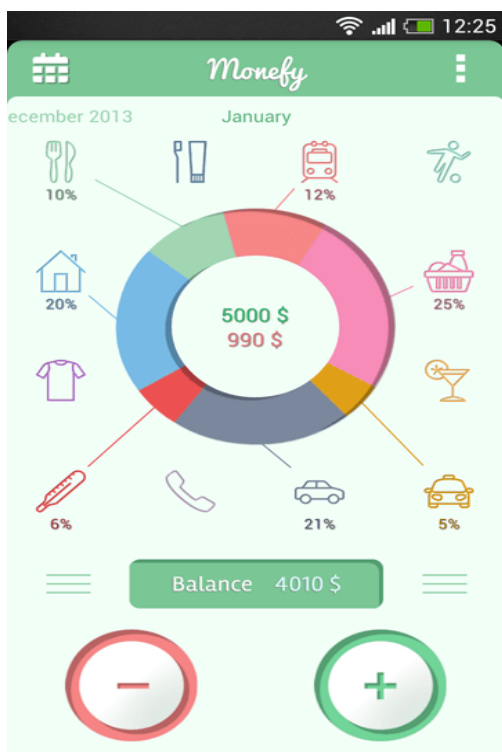


Figura 11, Gráfico de gastos por categoría de la aplicación Monefy

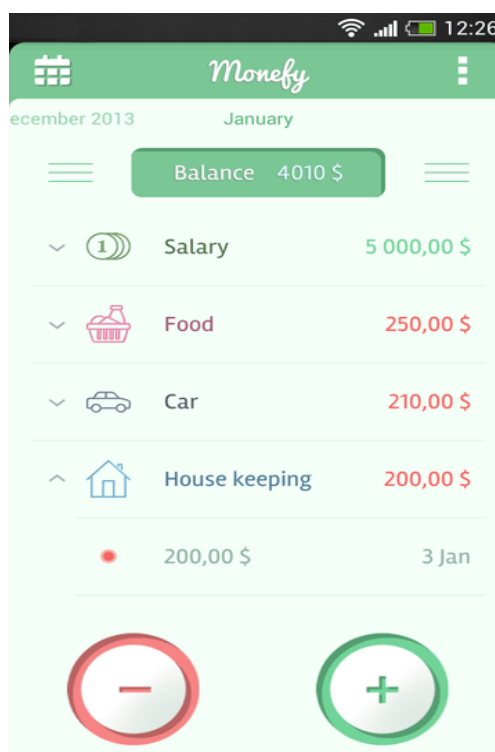


Figura 10, Agrupación de gastos por categoría de la aplicación Monefy

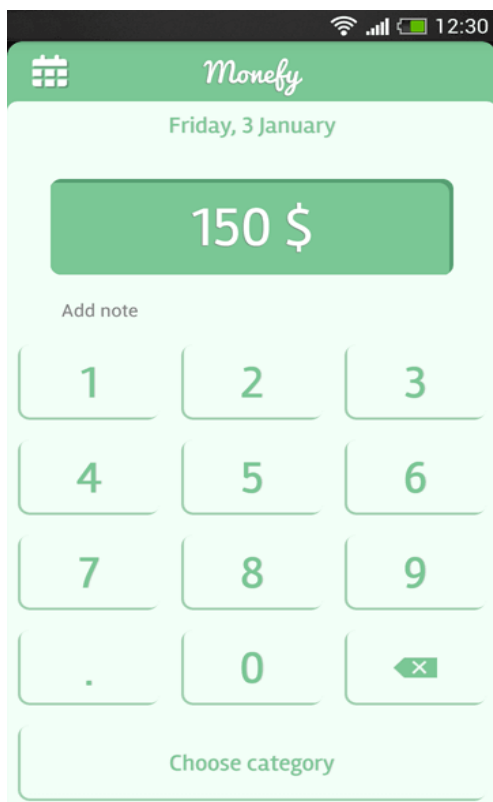
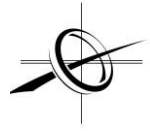


Figura 9, Introducción de un gasto en la aplicación Monefy



Figura 8, Categorización de un gasto introducido



## 4. ENGINSAVE

ENGINSAVE es la aplicación web de gestión financiera personal que se va a desarrollar para este proyecto, dicha aplicación recoge pinceladas de las distintas aplicaciones analizadas anteriormente.

Después de realizar un estudio sobre las aplicaciones ya existentes en el mercado, fueron surgiendo las ideas que serían la base del desarrollo de las funcionalidades de la aplicación que se va a llevar a cabo en el presente proyecto. Hay que destacar que se han añadido iniciativas personales para el diseño de la gestión de movimientos (página principal, página de gráficos...).

### 4.1 REQUISITOS DE ENGINSAVE

En este apartado se presentan los requisitos funcionales que deberán ser satisfechos por la aplicación. Todos los requisitos aquí expuestos son esenciales, es decir, no sería aceptable una aplicación que no satisfaga alguno de ellos.

#### REQUISITOS FUNCIONALES

- **GESTION DE USUARIOS**

RF001. Los usuarios podrán registrarse en la aplicación:

Este requisito hace referencia a la posibilidad de dar de alta un nuevo usuario en la aplicación. Mediante la interfaz gráfica se pedirán los datos necesarios para el registro. Una completado este paso, el usuario apretará el botón de registro para realizar el alta de usuario, en este paso se comprobarán si todos los campos están rellenos y si son correctos.

RF002. Los usuarios deben poder autenticarse en la aplicación:

Este requisito hace referencia a la autenticación del usuario en la aplicación, ya que para acceder a cualquier funcionalidad el usuario debe estar registrado. Una vez rellenos los campos de usuario y contraseña, el usuario debe pulsar el botón de login, en ese momento se comprobaran los credenciales y se dará o no acceso a la aplicación.

RF003. Los usuarios podrán modificar sus datos de perfil:

Este requisito hace referencia al cambio de datos personales de un usuario en la aplicación. Mediante la interfaz gráfica el usuario tendrá acceso a un apartado donde aparecerán todos los datos personales. Una vez cambiados los datos el usuario deberá aceptar los cambios pulsando sobre el botón de Guardar.

- **GESTIÓN DE MOVIMIENTOS**

RF004. Los usuarios podrán añadir sus movimientos diarios:

Este requisito hace referencia a la adición de un nuevo movimiento de usuario en la aplicación. Mediante la interfaz el usuario rellenará todos los campos necesarios para crear el nuevo movimiento (importe, tipo de movimiento, categoría, descripción, fecha de movimiento). Una vez rellenos todos los campos el usuario deberá añadir el movimiento pulsando sobre el botón Enviar.

RF005. Los usuarios podrán borrar sus movimientos diarios:

Este requisito hace referencia a la eliminación de un movimiento de usuario en la aplicación. Mediante la interfaz el usuario pulsará sobre el botón de borrado de un movimiento. Una vez pulsado el usuario deberá confirmar el borrado del movimiento pulsando sobre el botón Aceptar.

RF006. Los usuarios podrán actualizar sus movimientos diarios:

Este requisito hace referencia a la modificación de un movimiento de usuario en la aplicación. Mediante la interfaz el usuario modificará todos los campos deseados para actualizar el movimiento (importe, tipo de movimiento, categoría, descripción, fecha de movimiento). Una vez modificados el usuario deberá guardar el movimiento pulsando sobre el botón Enviar.

- **CONSULTA DE CONSEJOS DE AHORRO**

RF007. Los usuarios podrán consultar un listado de consejos:

Este requisito hace referencia al listado de consejos de ahorro que proporciona la aplicación. El usuario deberá acceder a la pestaña dedicada a este fin en la aplicación y observar los distintos consejos obtenidos a través de una consulta en la red social Twitter.

RF008. Los usuarios podrán actualizar la lista de consejos pulsando sobre el botón actualizar:

Este requisito hace referencia a la actualización de la lista de consejo que puede ser consultada. El usuario, ya situado en la pestaña dedicada en la aplicación para consejos de ahorro pulsara sobre el botón actualizar para que se refresque la lista de consejos obtenida a través de la red social Twitter.

- **CONSULTA DE INFORMACIÓN GRÁFICA**

RF009. Los usuarios podrán consultar la información de sus movimientos de forma gráfica:

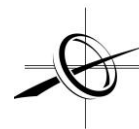
Este requisito hace referencia a la visualización de los movimientos de los usuarios de forma gráfica. El usuario podrá obtener información valiosa de forma visual en la pantalla principal y en la pantalla dedicada especialmente a este fin, denominada Gráficos

## **REQUISITOS DE LA INTERFACES**

- **INTERFAZ DE USUARIO**

La representación de los datos de forma visual a través de diagramas, calendarios y colores en las categorías e importes y la sencillez de la página, debe otorgar un manejo sencillo e intuitivo de la aplicación para el usuario.

La aplicación será multiplataforma, por lo que el diseño de la aplicación debe ser responsive (adaptable según el dispositivo con el que se acceda).



## REQUISITOS DE SEGURIDAD

La seguridad es un elemento muy importante, y sobre todo en este tipo de aplicaciones en las cuales se maneja una gran cantidad de datos con información sensible de carácter personal, por ello nunca deben ser usadas las contraseñas tal cual las introduce el usuario. Las contraseñas que el usuario escriba en la aplicación deberán ser tratadas mediante el método de encriptación SALT, que añade una palabra secreta a la contraseña y luego cifra con el método base64.

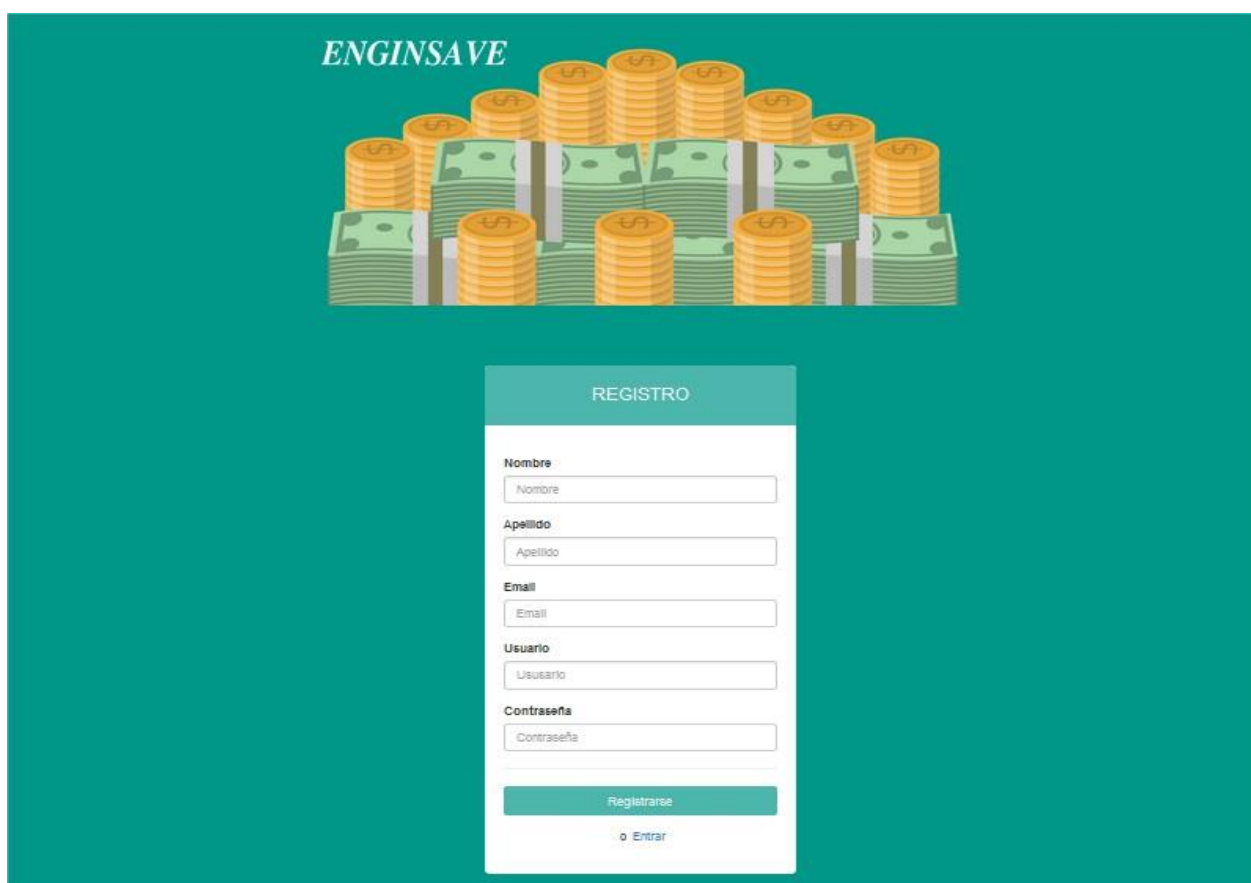
## 4.2 RESULTADO FINAL DE LA APLICACIÓN

Una vez especificados los requisitos de la aplicación, se ha desarrollado la aplicación al completo satisfaciendo todos y cada uno de ellos. En las posteriores páginas se mostrará el resultado final y como se ha implementado cada uno de los requisitos.



## 4.1 ACCESO A LA APLICACIÓN

Para acceder a la aplicación el usuario debe estar registrado (requisito RF001), lo cual se hace accediendo a la pantalla de registro, la cual se muestra en la Figura 12. Puede observarse que el registro requiere datos personales básicos que pedirían en cualquier página para poder crear un perfil personalizado.



ENGINSAVE

REGISTRO

Nombre

Apellido

Email

Usuario

Contraseña

Figura 12, Pantalla de registro de la aplicación web ENGINSAVE

Una vez realizado el registro el usuario estará preparado para acceder a la aplicación (requisito RF002) con su nombre de usuario y contraseña desde la pantalla de login, que se muestra a continuación (Figura 13).

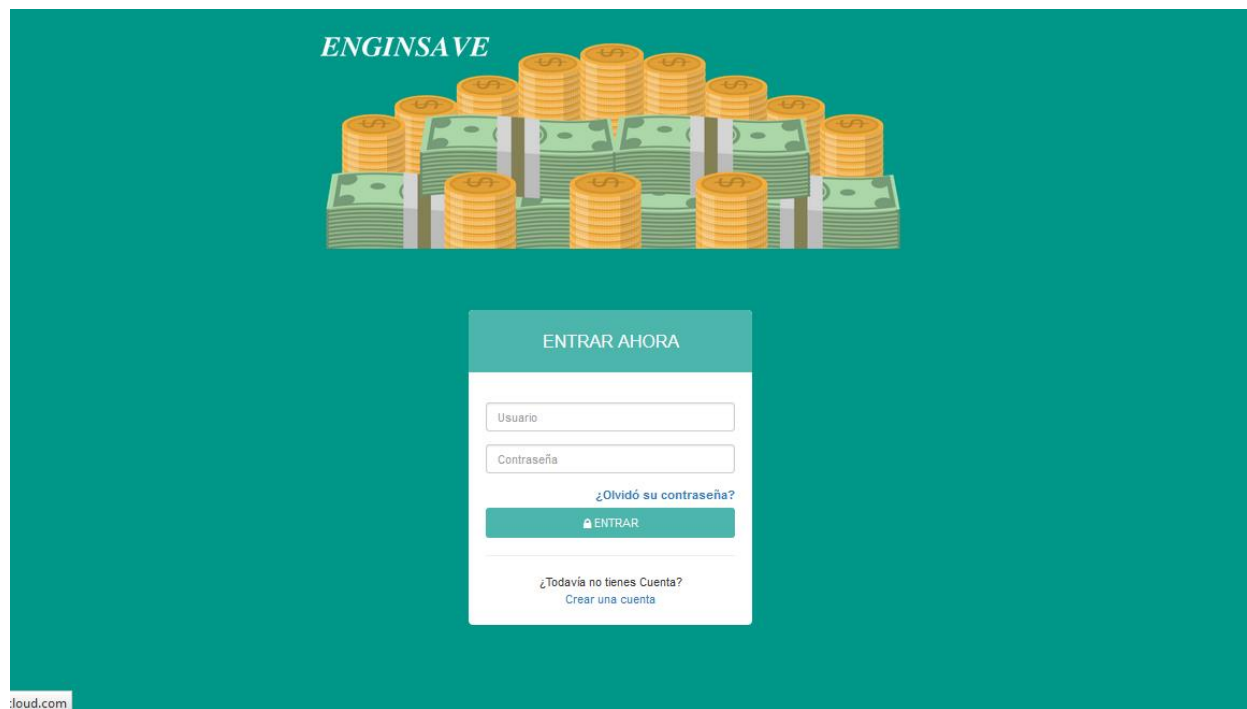


Figura 13, Pantalla de inicio de sesión de la aplicación web ENGINSAVE

La aplicación también incluye la función de recuperación de la contraseña, esta es una funcionalidad básica para cualquier aplicación que contiene un perfil personalizado. Al utilizar esta opción, se debe introducir el correo electrónico en el cuadro de dialogo que se muestra en la Figura 14 y finalizar la acción enviando la petición. Esta petición enviará un mail a la dirección de correo del usuario, tal y como se observa en la Figura 15, este mensaje contiene un enlace para poder restablecer la contraseña.

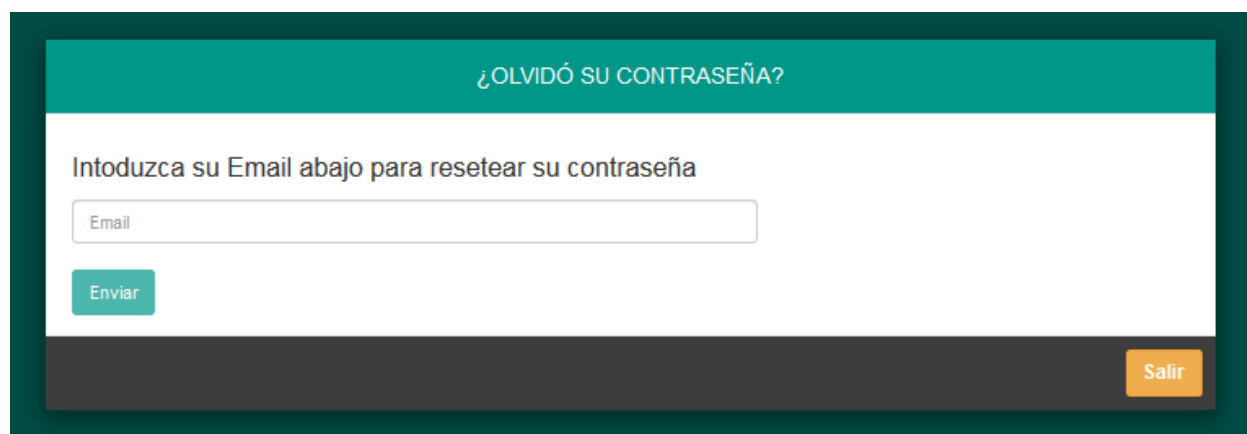
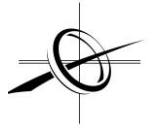


Figura 14, Ventana emergente de introducción de correo electrónico.



Esta recibiendo este correo porque usted (o alguien) ha solicitado el reinicio de la contraseña para su cuenta.

Por favor haga click en el siguiente enlace, o péguelo en el navegador para completar el proceso:

<http://nodejs-enginsave.rhcloud.com/reset/48d9be56fbb4f6840269ef8c3ffdf363df1f7bd3>

Si no hizo esta petición, por favor ignore este mensaje y su contraseña continuará como hasta ahora.

Figura 15, Correo de reinicio de contraseña de usuario.

Siguiendo la preocupación por la seguridad desde la que se ha diseñado esta aplicación, si pasada media hora desde que se envió el correo no se ha reseteado la contraseña accediendo al enlace proporcionado, este quedará invalidado y será necesario el envío de un nuevo correo con un nuevo enlace.

## 4.2 INICIO

En la Figura 16 se muestra la pantalla principal a la se accede una vez iniciada la sesión. Como se puede observar se obtiene un resumen de la situación general de los movimientos del usuario en el último mes, y para hacer que el uso y entendimiento de la aplicación sea fácil e intuitivo la información se muestra utilizando colores y gráficas (requisito RF009). En el supuesto de que el usuario no haya realizado ningún movimiento en el mes se mostraría un mensaje indicándolo.

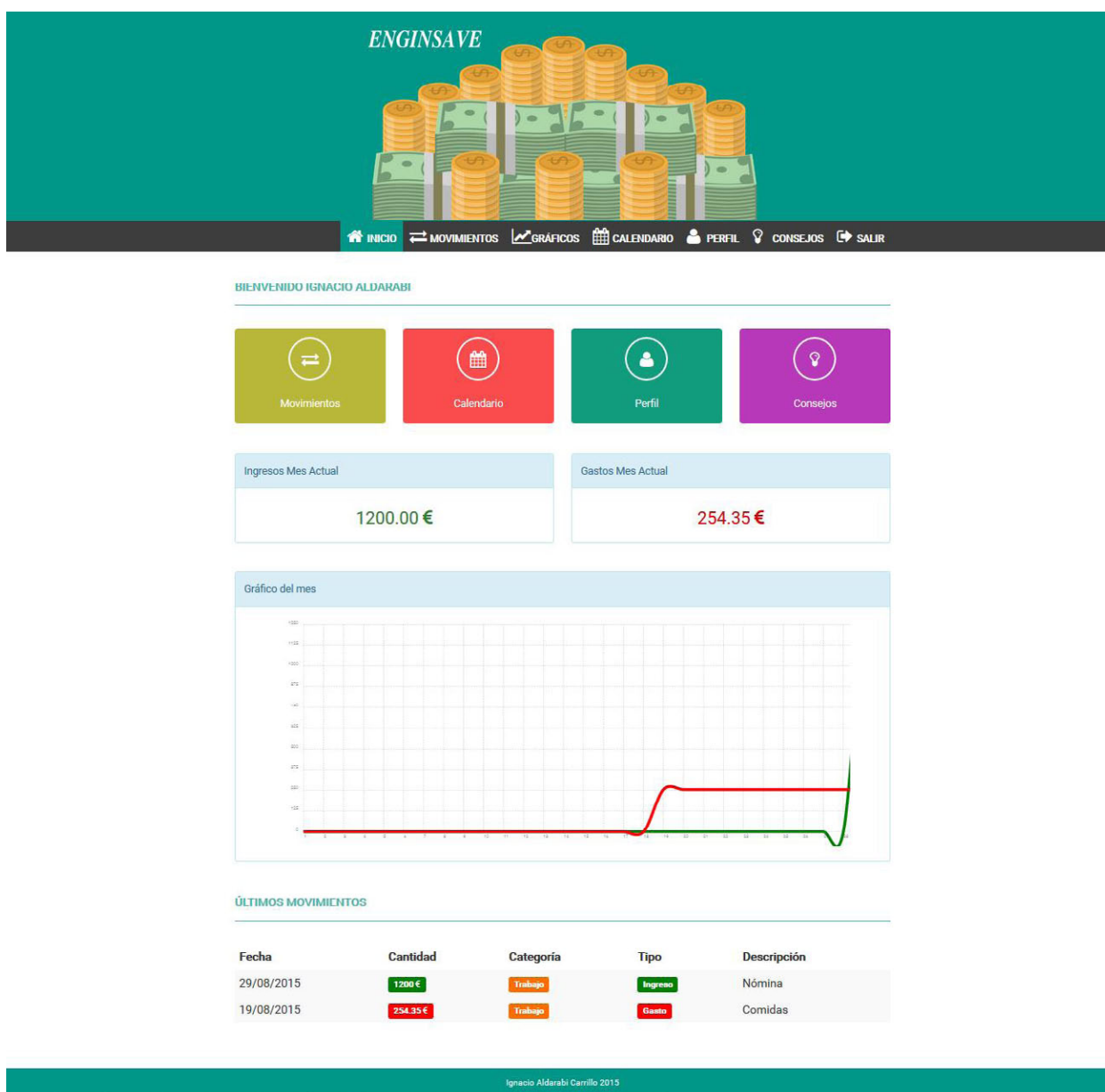
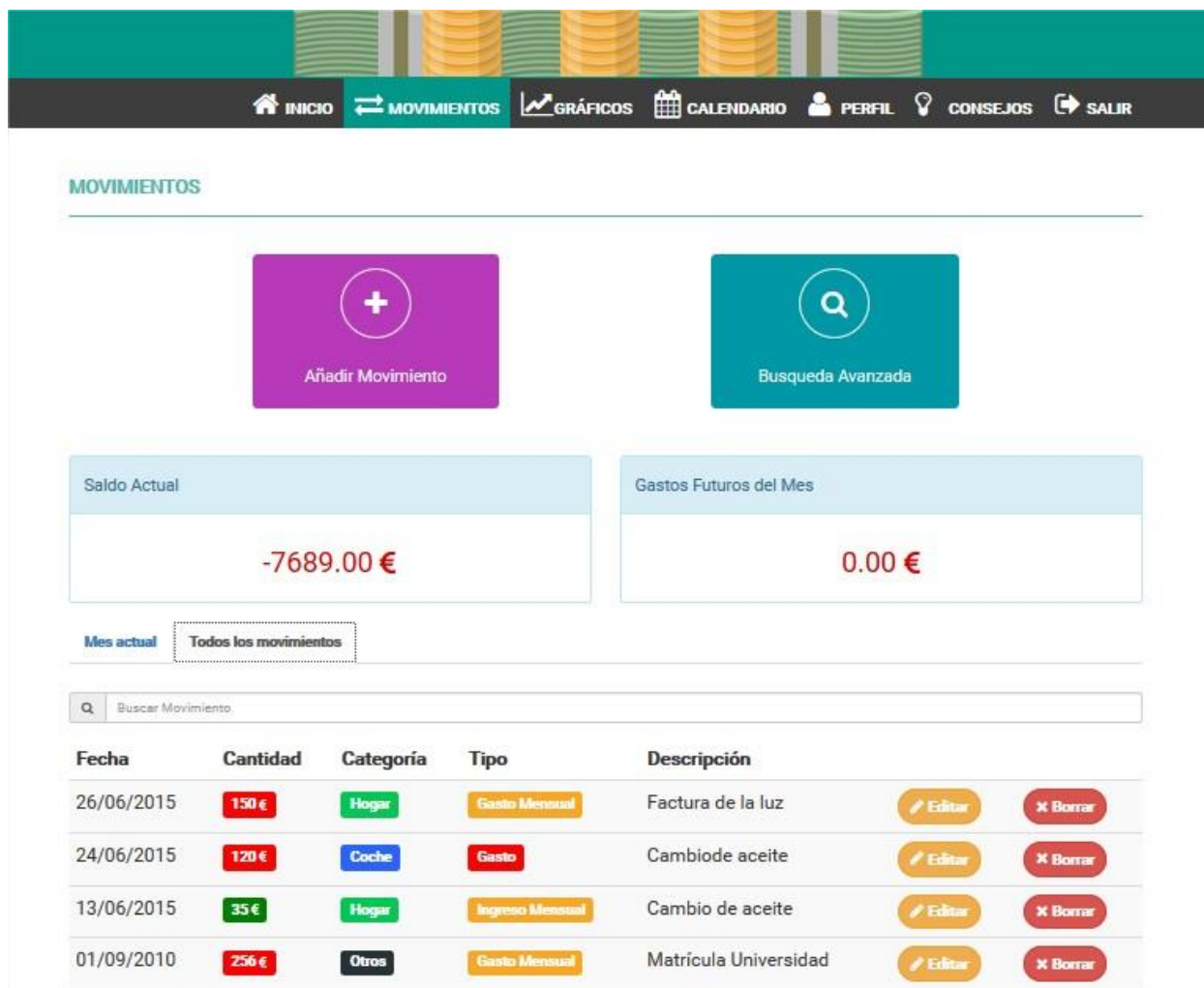


Figura 16, Pantalla principal de la aplicación web ENGINSAVE

Desde la pantalla principal también se puede acceder a las diferentes funcionalidades de la aplicación a través de los botones situados bajo el logo.

## 4.3 MOVIMIENTO

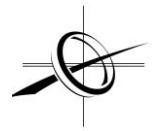
En la Figura 17 se muestra la pantalla de movimientos, en la cual podemos consultar, editar o borrar todos los movimientos hechos en el mes actual (requisitos RF004, RF005 y RF006) y también un histórico de todos los registrados en la aplicación.



Fecha	Cantidad	Categoría	Tipo	Descripción	Editar	Borrar
26/06/2015	150 €	Hogar	Gasto Mensual	Factura de la luz	Editar	Borrar
24/06/2015	120 €	Coche	Gasto	Cambiode aceite	Editar	Borrar
13/06/2015	35 €	Hogar	Ingreso Mensual	Cambio de aceite	Editar	Borrar
01/09/2010	256 €	Otros	Gasto Mensual	Matrícula Universidad	Editar	Borrar

Figura 17, Pantalla de gestión de los movimientos de la aplicación web ENGINSAVE

Se puede observar que hay dos opciones, por un lado la opción de añadir un movimiento y por otro lado la búsqueda avanzada (esta opción no está desarrollada, se detallará en el apartado de futuras mejoras). La opción de añadir movimiento abre una ventana en la que se tendrá que configurar las características del nuevo movimiento, entre ellas el importe, el tipo de movimiento que es, a que categoría pertenece...



Debajo de estas opciones se puede ver la posición global del mes y los gastos programados restantes del mes actual, esto es de gran ayuda para gestionar el dinero mensual y evitar sobresaltos en la economía.

Para finalizar esta parte, al final de la ventana se puede visualizar todos los movimientos que registrados que se podrán tanto editar como eliminar según las necesidades del usuario.

La opción de búsqueda no está desarrollada, pero con vistas a futuro ayudaría a realizar una búsqueda avanzada de movimientos pudiendo filtrar por varios campos a la vez, como por ejemplo importe, nombre, tipo...

## 4.4 CALENDARIO

En la Figura 18 se puede observar un calendario que proporciona la facilidad de visualizar los movimientos realizados mensualmente y hacer un repaso rápido de los gastos.

El calendario usa un código de colores para representar los gastos (color rojo) e ingresos (color verde) que junto con la descripción de los movimientos al situar el ratón sobre estos proporciona una mayor agilidad de uso.

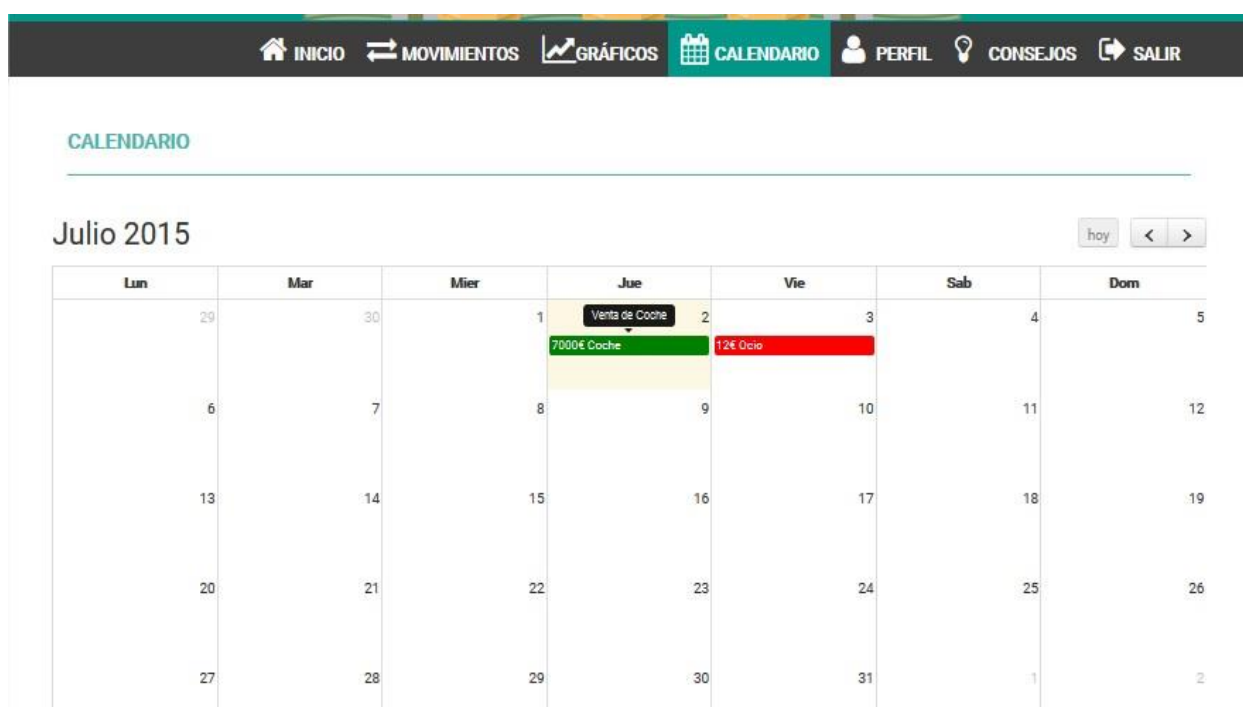


Figura 18, Calendario que muestra visualmente las fechas en las que se producen los gastos en la aplicación web ENGINSAVE.

## 4.5 GRÁFICOS

Como ya se ha mencionado con anterioridad, la aplicación está constituida por una gran representación de los datos a través de gráficos, los cuales hacen mejor el entendimiento de la misma además de ayudar a su uso y poder así abrirse a un amplio público objetivo (requisito RF009).

La Figura 19 muestra un resumen grafico de los gastos, en dos escalas de detalle, por un lado se puede observar el detalle mensual de gastos y por otro lado el detalle del año actual completo. De este tipo de gráficos se puede extraer información de valor como por ejemplo, en que se gasta más, cuál es el balance anual y mensual... Estos datos proporcionan una idea general de la evolución de gastos que ayudará al usuario a tomar decisiones sobre la gestión de su economía personal.



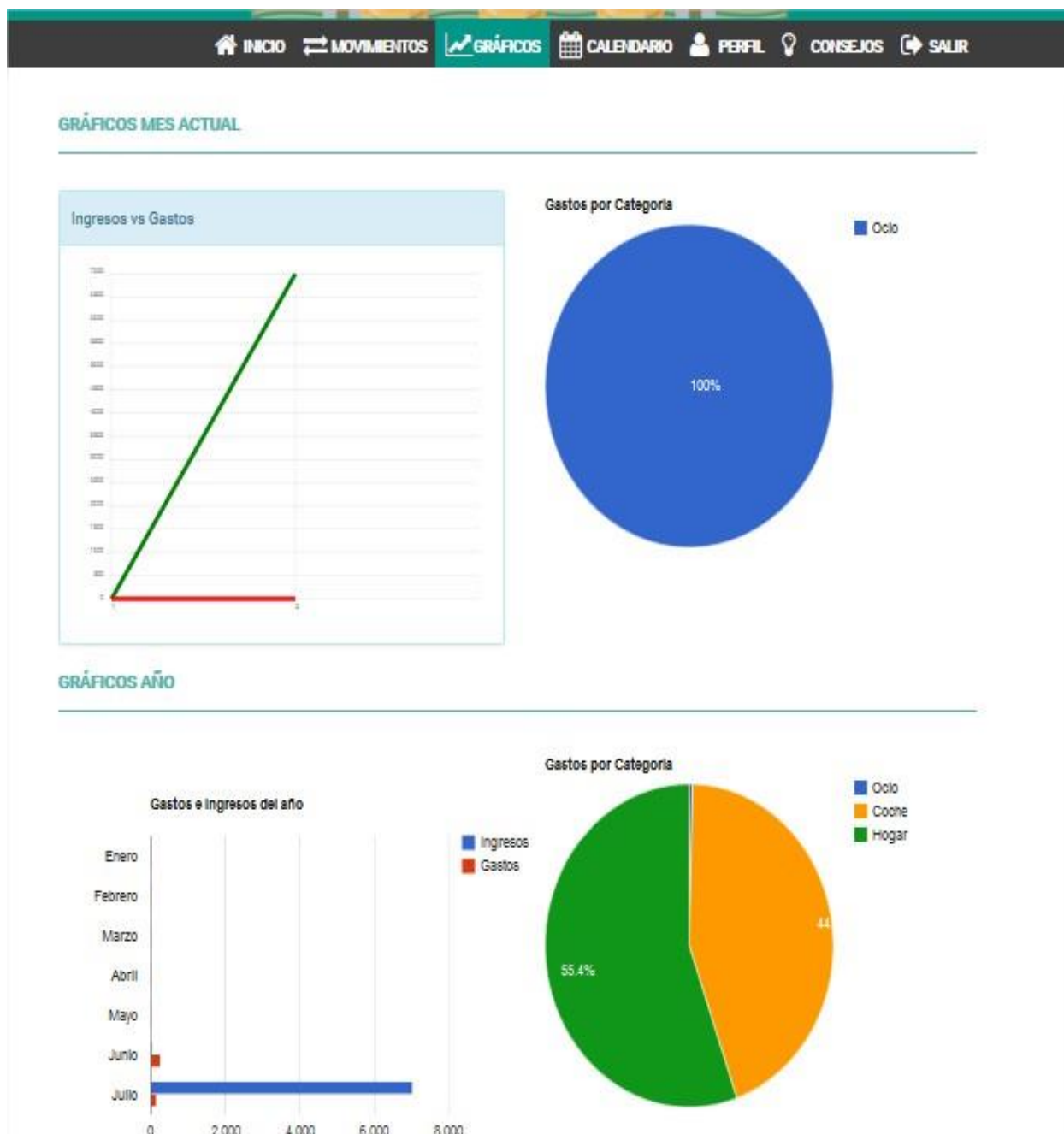
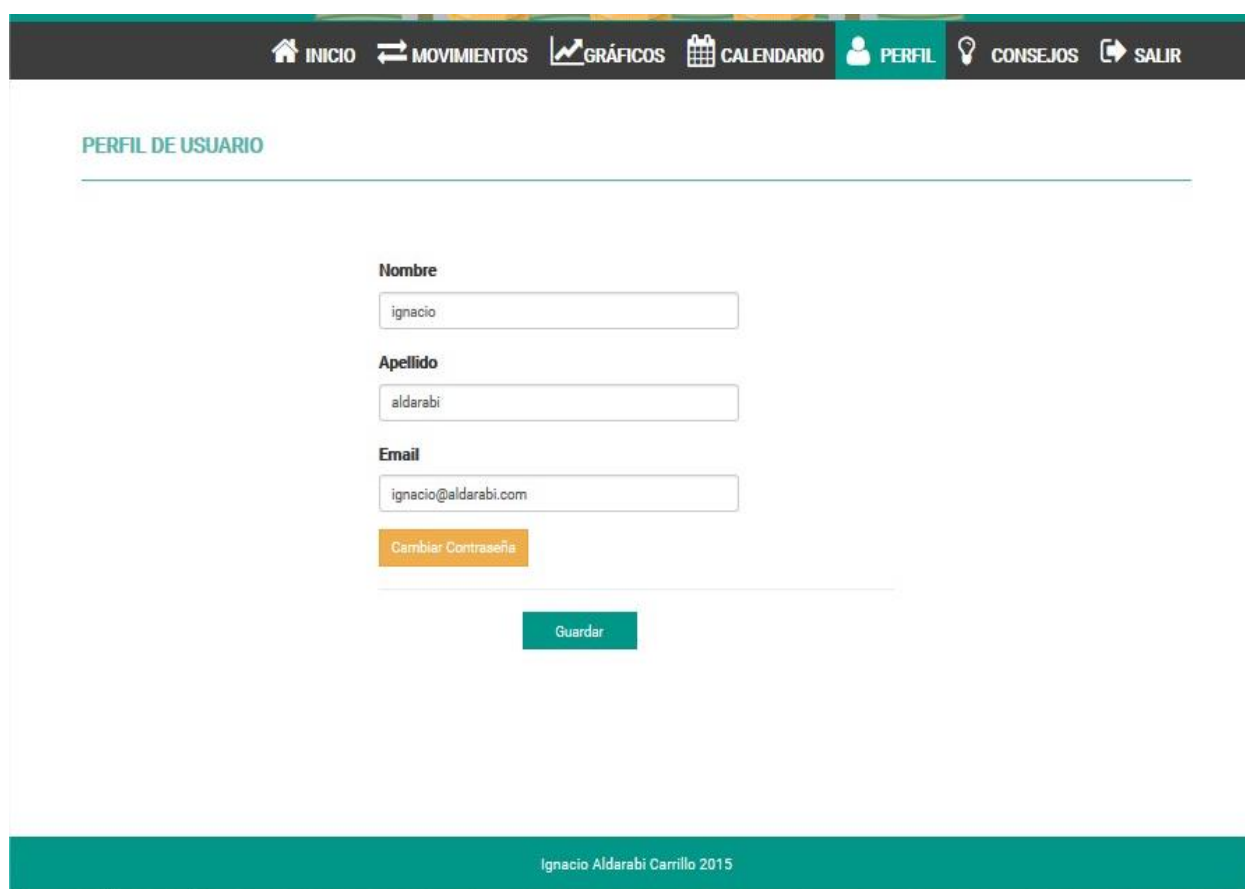


Figura 19, Pantalla de gráficos de la aplicación web ENGINSAVE.

## 4.6 PERFIL

A lo largo de este documento se ha visto que la aplicación ENGINSAVE maneja datos de usuario personales, por ello es importante dotar a la aplicación de una funcionalidad donde se puedan editar dichos datos (requisito RF003).

En esta funcionalidad se puede editar los campos básicos de información personal (Nombre, Apellido y Correo electrónico) (Véase Figura 20)



The screenshot shows the 'PERFIL DE USUARIO' (User Profile) page. At the top, there is a navigation bar with icons and labels for 'INICIO', 'MOVIMIENTOS', 'GRÁFICOS', 'CALENDARIO', 'PERFIL' (active), 'CONSEJOS', and 'SALIR'. Below the navigation bar, the page title 'PERFIL DE USUARIO' is displayed. The main content area contains three input fields for 'Nombre' (containing 'ignacio'), 'Apellido' (containing 'aldarabi'), and 'Email' (containing 'ignacio@aldarabi.com'). Below these fields is an orange button labeled 'Cambiar Contraseña'. At the bottom of the form is a green button labeled 'Guardar'. The footer of the page shows 'Ignacio Aldarabi Carrillo 2015'.

Figura 20, Pantalla para edición de datos de perfil de usuario de la aplicación web ENGINSAVE

## 4.7 CONSEJOS

La gestión de la economía ocupa un lugar importante entre las preocupaciones de la vida cotidiana de cualquier individuo, por ello a la hora de diseñar las funcionalidades de la aplicación ENGINSAVE se ha añadido un apartado de consejos para captar ideas que mejoren la economía de forma global (requisito RF007).

La Figura 21 muestra la pantalla de consejos incluida en ENGINSAVE. Los consejos están en constante actualización, ya que se extraen de la red social Twitter (requisito RF008).

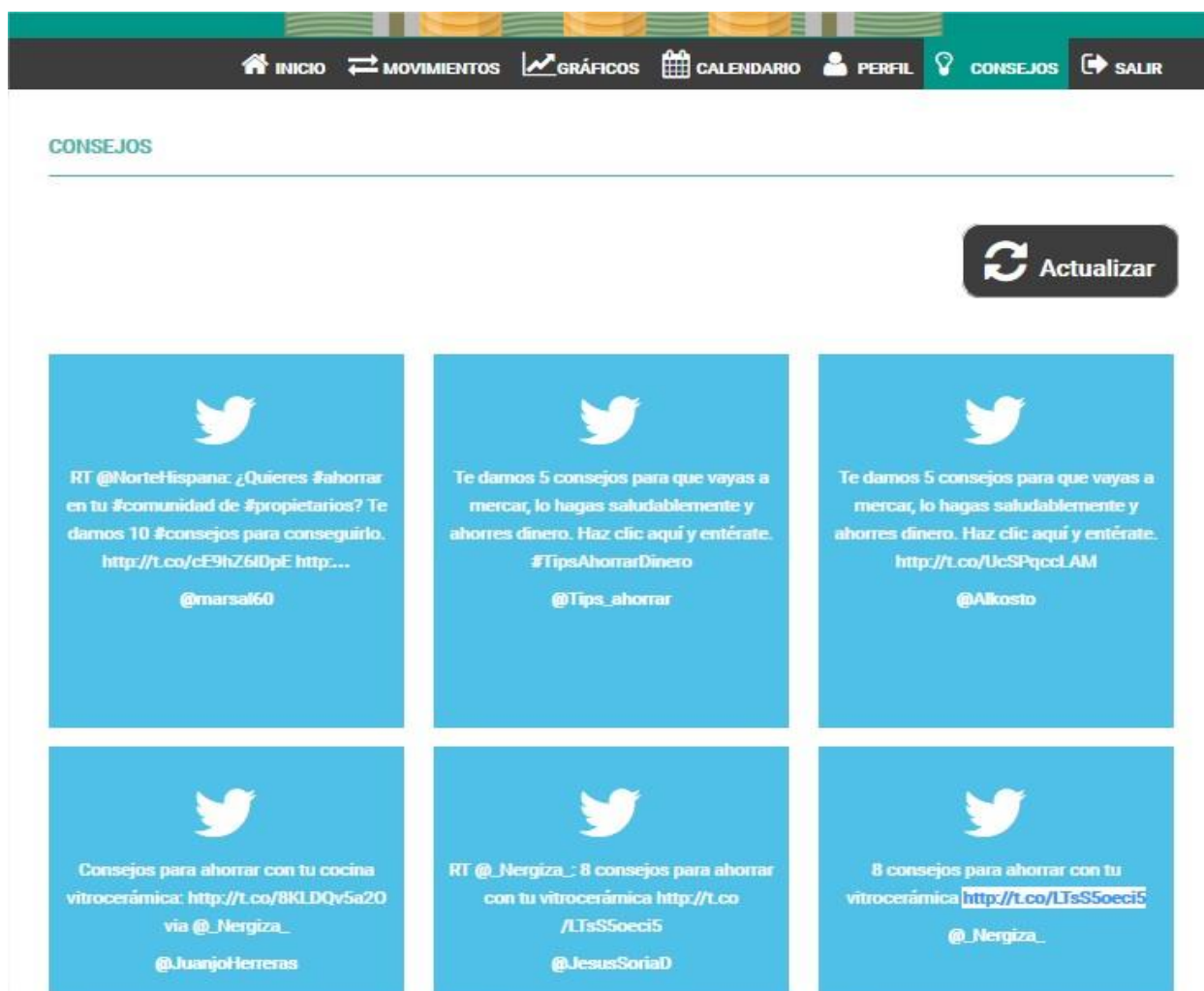
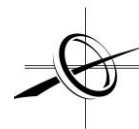


Figura 21, Pantalla de consejos de la aplicación web ENGINSAVE



A modo de resumen, destacar que ENGINSAVE es una aplicación de gestión económica que utiliza datos personales y movimientos económicos para crear gráficos y resúmenes que ayuden al usuario a mejorar su economía personal. Esta aplicación está diseñada siguiendo los criterios de seguridad necesarios al tratarse de una aplicación informática que maneja información sensible de alto valor personal. Además se ha diseñado siempre pensando que el público objetivo potencialmente utilitario de la misma sea lo más amplio posible, dotándola de gráficos resumen que facilitan el uso de la misma.

ENGINSAVE constituye una herramienta fácil de utilizar que proporciona información sencilla de interpretar por el usuario, lo cual ayuda a optimizar la evolución económica de sus gastos e ingresos.

## 5. TECNOLOGÍAS UTILIZADAS

Para desarrollar este trabajo, ENGINSAVE se ha basado en el uso de la tecnología full-stack MEAN.JS [15], la cual vamos a proceder a definir detalladamente.

### 5.1 MEAN.JS

MEAN es una abreviación para MongoDB, Express.js, Angular.js y Node.js. Todas estas tecnologías, que juntas cubren los diferentes aspectos de una aplicación (desarrollo end-to-end), tienen en común el uso de JavaScript.

JavaScript es un lenguaje ligero e interpretado, orientado a objetos que se usa tanto en entornos con navegador como en entornos sin navegador (Node.js). Como se puede observar es un lenguaje muy versátil, ya que se puede usar tanto del lado del servidor como del lado del cliente.

Las ventajas que se obtienen al utilizar estas tecnologías son numerosas, tales como:

- El uso de un único lenguaje en toda la aplicación, JavaScript.
- Todas las partes de la aplicación soportan, y a menudo refuerzan, el uso de la arquitectura MVC.
- La serialización y deserialización de estructuras de datos ya no es necesaria porque el cálculo de referencias de datos se hace utilizando objetos JSON.

Para entender cómo funciona el desarrollo con estas tecnologías a continuación se detallarán cada una de ellas.

### 5.2 NODE.JS

Node.js [2] es un framework que se encarga de ejecutar código JavaScript en el lado del servidor. Node.js por sí solo es un servidor, pero la gestión de peticiones HTTP por el mismo es muy complicada, por ello hay que dotarle de Express.js, que facilita enormemente la gestión de peticiones.

La principal diferencia entre Node y los servidores web tradicionales (IIS o Apache) es que Node trabaja con un único hilo de ejecución. Una de las preocupaciones más comunes de este tipo de ejecución es el bloqueo del hilo por una tarea que conlleve mucho tiempo. Para solventar este problema lo que hace Node es gestionar todas sus tareas de forma asíncrona, lo cual es un concepto revolucionario e innovador, ya que la gestión de las ejecuciones se realiza de manera mucho más eficiente debido a que todo el código se ejecuta en paralelo, lo cual significa que en vez de bloquear el flujo de ejecución por la espera de la respuesta de una instrucción, Node sigue ejecutando otras instrucciones hasta que el proceso anterior acaba y devuelve una respuesta.

El procesamiento asíncrono se realiza de la siguiente manera, el servidor web recibe una petición, entonces se registra su callback y se ejecuta el proceso pedido (consulta de BBDD, procesado de archivo...). Mientras esta ejecución se lleva a cabo Node sigue ejecutando el resto de instrucciones en paralelo, y si se realiza otra petición se sigue el mismo procedimiento que para la petición anterior.

Cuando alguno de los procesos termina se avisa al bucle de evento, el cual es el encargado de enviar el callback a nuestra aplicación para que continúe con la ejecución anterior. La siguiente figura (Figura 22) es una representación gráfica de este concepto.



Figura 22, Representación de funcionamiento asíncrono de Node.js

Para intentar reforzar este concepto, se verá un ejemplo de una ejecución real en la cual la aplicación realiza dos procesos (Figura 23), una recuperación de los seguidores de un usuario en Twitter, y la segunda, la creación de un thumbnail.

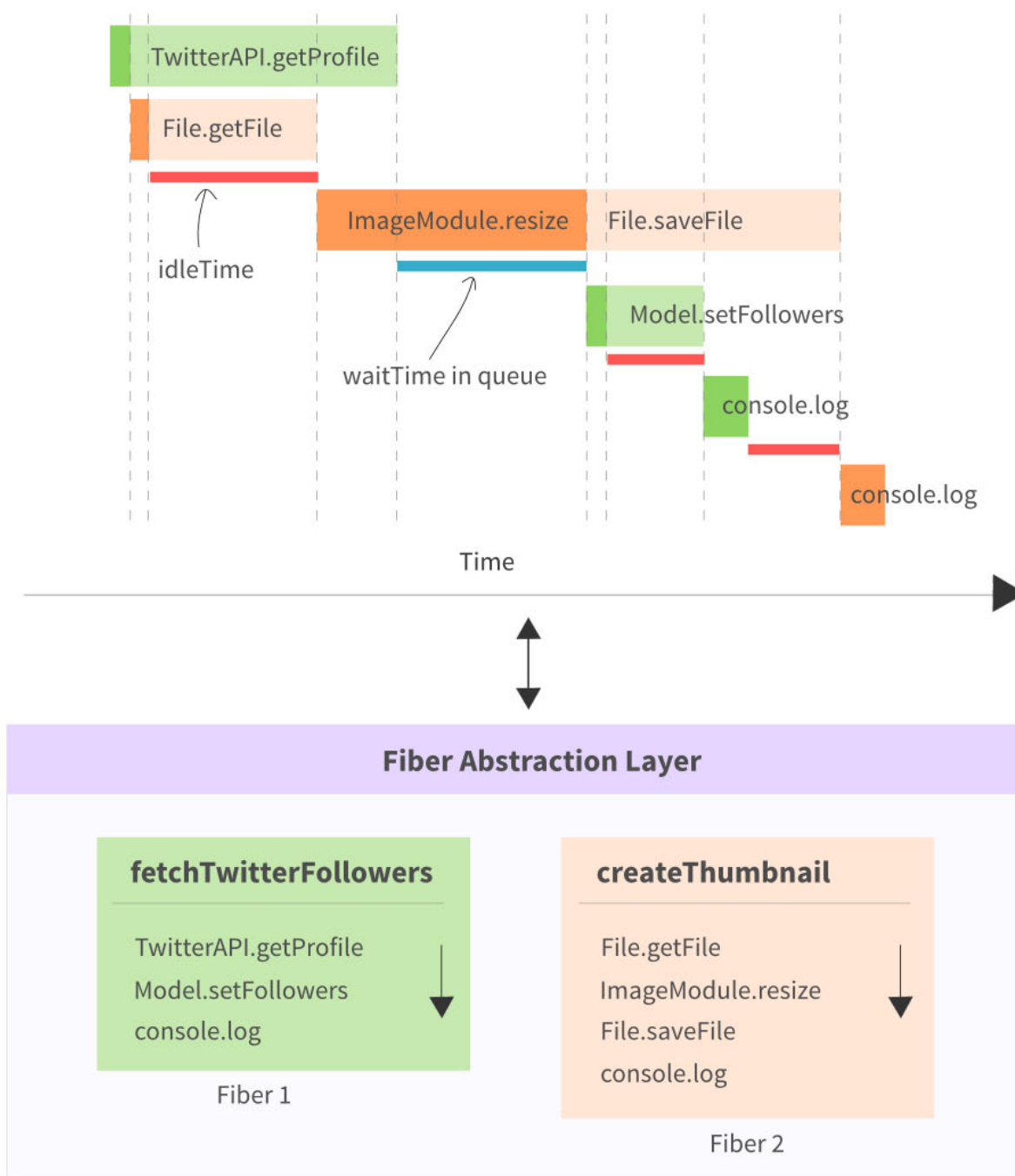


Figura 23, Ejecución en Node.js de dos peticiones.

Como se puede observar primero se gestiona la primera petición del primer proceso, la de recuperación del perfil de Twitter, mientras esta petición termina, Node ejecuta la primera petición del segundo proceso, la recuperación del archivo a convertir en thumbnail. Cuando acaba la recuperación del archivo, la recuperación del perfil de Twitter todavía no ha acabado, por lo que se ejecuta la siguiente instrucción de la creación del thumbnail, la redimensión de la imagen.

Durante el proceso de redimensión de la imagen, la recuperación del perfil de Twitter termina, y la siguiente instrucción de ese proceso se encola hasta que el proceso actual termina. Cuando se ejecuta la siguiente instrucción del proceso de creación de thumbnail, salvar el archivo creado, se desencola la petición del proceso anterior, y se ejecuta, que en este caso es la asignación de los seguidores del perfil. Cuando este proceso acaba el de salvar el thumbnail todavía está en ejecución, por lo que se ejecuta la última instrucción de la recuperación de los seguidores de Twitter, la impresión por pantalla.

Finalmente cuando termina la ejecución de salvado de la imagen se ejecuta la última petición del proceso de creación de thumbnail, que este caso también en una impresión por pantalla.

Como puede observarse único hilo de ejecución simplifica enormemente desarrollo de aplicaciones web, pero se plantean las siguientes cuestiones, la aplicación que se desarrolla en este trabajo se ejecuta sencillamente con un comando, pero ¿qué ocurre si se cae el proceso en producción?, si se necesita el rendimiento de una aplicación multiproceso, ¿no se pueden aprovechar los otros núcleos de microprocesador? Para cubrir estas necesidades existe un módulo para Node.js llamado PM2 [16].



PM2 es un gestor de procesos que proporciona la opción de poder crear balanceadores de carga a nuestra aplicación, entre sus principales características destacan las siguientes:

- Capacidad de manejar un gran número de procesos.
- Capacidad de monitoreo de memoria y cpus de nuestros procesos.
- Manejo de logs.
- Iniciar tus aplicaciones una vez el servidor se inicia.
- Levantar el proceso si se cae debido a un error.

Por todas estas características es muy recomendable el uso de este módulo tanto para la fase de desarrollo, como para la productiva.

En cuanto a la manera en que se escriben las aplicaciones, las aplicaciones Node tienen más en común con PHP o Ruby que con aplicaciones .NET o Java. El motor de JavaScript que Node utiliza (V8 de Google) compila JavaScript a código máquina (al igual que C o C++), y lo hace transparente, así que desde la perspectiva del usuario, se comporta como un lenguaje puramente interpretado. No tener un paso de compilación separado reduce mantenimiento y molestias de despliegue: todo lo que se debe hacer es actualizar un archivo de JavaScript, y los cambios automáticamente estarán disponibles.

Otra ventaja muy destacable de Node es la gran cantidad de módulos adicionales que pueden ser integrados. En las posteriores secciones mostraremos tanto el funcionamiento de la ejecución de hilo único de Node como la gestión de los módulos de Node.

Los módulos de Node extienden sus funcionalidades permitiéndole por ejemplo conectarse a una base de datos o utilizar la API de una aplicación. Suelen instalarse a través de un gestor de paquetes llamado Node Package Manager (NPM) [11]. El registro NPM es una colección pública de paquetes de código abierto para Node.js, aplicaciones web front-end, aplicaciones móviles, robots, routers y otras innumerables necesidades de la comunidad de JavaScript.

Parte integral del ecosistema de NPM es un archivo JSON simple llamado package.json. Este archivo tiene un significado especial para NPM y es de vital importancia tener configurado

correctamente cuando se desea compartir un módulo con la comunidad, pero es igualmente útil si se están consumiendo módulos de otras personas. Dentro de este módulo se encuentra información de la aplicación tales como su nombre, una descripción, y la dependencia de módulos de la aplicación. Una de las grandes ventajas que reporta este archivo es la configuración rápida del proyecto con un solo comando y sin la necesidad de realizar un backup de los paquetes que se hayan instalado.

Un ejemplo de package.json es el siguiente, el cual corresponde al archivo usado en la aplicación que nos ocupa, ENGINSAVE.

```
//Esta parte del archivo recoge los datos identificativos de la aplicación//
{
  "name": "enginsave",
  "version": "1.0.0",
  "description": "App para control de gastos personales",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Ignacio",
  "license": "ISC",
  //Esta parte del archivo es donde se rocen todas las dependencias de la
  //aplicación y la versión con la que se trabaja de esos módulos//
  "dependencies": {
    "add-flash": "0.0.3",
    "body-parser": "^1.12.4",
    "bower": "^1.4.1",
    "connect-mongo": "^0.8.1",
    "cookie-parser": "^1.3.5",
    "express": "^4.12.4",
    "express-flash": "0.0.2",
    "express-session": "^1.11.2",
    "method-override": "^2.3.2",
    "mongodb": "^2.0.33",
    "mongoose": "^4.0.5",
    "morgan": "^1.5.2",
    "nodemailer": "^1.3.4",
    "passport": "^0.2.2",
    "passport-local": "^1.0.0",
    "swig": "^1.4.2",
    "twitter": "^1.2.5"
  }
}
```

## 5.3 EXPRESS.JS

Express es el componente de Node.js que se encarga de la gestión de conexiones HTTP, lo que proporciona un carácter de servidor web a Node.

Según la página oficial de Express [10], *“Express es un framework web minimalista y flexible para Node.js que proporciona un conjunto robusto de características para aplicaciones web y móviles.”*.

El hecho de ser minimalista es uno de los aspectos más atractivos de Express. Generalmente, los desarrolladores olvidan la idea de que "menos es más", sin embargo, la filosofía de Express es proporcionar la capa mínima entre el desarrollador y el servidor. Eso no quiere decir que no sea robusta, o que no tenga características suficientemente útiles, lo que significa es que se interponen menos cosas en el camino, lo cual permite plena expresión de ideas por parte de los desarrolladores.

Otro aspecto clave de la filosofía de Express es su extensibilidad. Express ofrece un framework muy minimalista al que se le puede agregar diferentes funcionalidades según sea necesario, o cambiar lo que no cumple las expectativas del desarrollador. Esto es un soplo de aire fresco, ya que muchos frameworks incluyen muchas funcionalidades por defecto, confeccionando proyectos sobredimensionados, misteriosos y complejos antes incluso de que se haya escrito una sola línea de código. Muy a menudo, la primera tarea es perder tiempo quitando todas las funcionalidades innecesarias, o sustituyendo las funcionalidades que no cumple con los requisitos.

En cuanto a otro de los conceptos más revolucionarios de Express.js es la creación de aplicaciones web single-page (de una única página). Este tipo de aplicaciones en lugar de requerir un sitio web que requiere una solicitud de red cada vez que el usuario navega a una página diferente, en una aplicación single-page de una página descargas todo el sitio (o una buena parte de él) en el navegador del cliente. Después de la descarga inicial, la navegación es más rápida porque hay poca o ninguna comunicación con el servidor. El desarrollo de aplicaciones web de una página se facilita por el uso de frameworks populares como Angular o Ember, que Express integra perfectamente. Esta es una forma innovadora de desarrollar aplicaciones web.

Para la gestión de las conexiones HTTP Express tiene diversos comandos, a continuación se mostrarán los más significativos, que son aquellos que componen el interfaz CRUD (escritura, lectura, actualización y borrado). Dichas instrucciones son las siguientes:

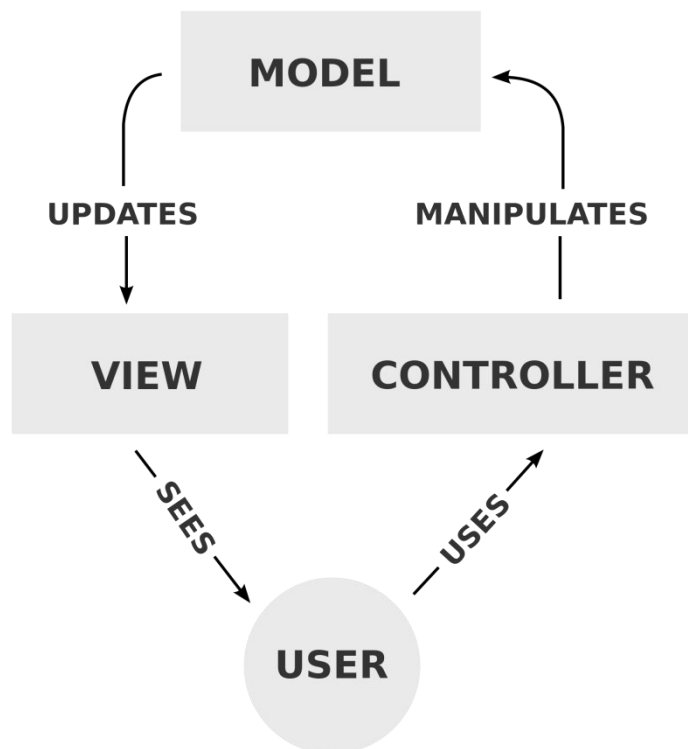
- `app.get(path, callback [, callback ...])`: método que se ejecuta cuando se recibe una petición de tipo GET con un path determinado, se usa en peticiones de consultas.
- `app.post(path, callback [, callback ...])`: método que se ejecuta cuando se recibe una petición de tipo POST con un path determinado, se usa en peticiones de creación o inserción de registros o datos.
- `app.put(path, callback [, callback ...])`: método que se ejecuta cuando se recibe una petición de tipo PUT con un path determinado, se usa en peticiones para actualizar registros.
- `app.delete(path, callback [, callback ...])`: método que se ejecuta cuando se recibe una petición de tipo DELETE con un path determinado, se usa en peticiones para la eliminación de registros.

Todas las peticiones contienen paths o rutas que pueden estar compuestas por expresiones regulares para ser definidas, también pueden tener una función de callback asociada. El orden en el que se definen las diferentes llamadas en el código tiene importancia, ya que si una petición coincide con varias definiciones de petición, esta entrará por la primera que haya sido definida.

## 5.4 ANGULAR.JS

Angular.js [3] es un framework en JavaScript muy potente, y es el encargado de gestionar el frontend de la aplicación.

Angular.js soporta la estructura MVC [3], que fue introducida en los 70, y desde entonces tuvo una gran acogida entre diferentes tecnologías. La siguiente figura (Figura 24) muestra la lógica de esta estructura.



*Figura 24, Esquema de arquitectura MVC*

La idea principal del MVC es la clara separación en el código del manejo de datos (los que son el modelo), la lógica de la aplicación (el controlador), y la presentación de los datos al usuario (la vista).

Seguir la estructura MVC es muy útil, ya que proporciona una idea mental de dónde situar cada elemento evitando así la tarea de investigación cada vez que se quiera ampliar el contenido de la aplicación. Otro gran motivo para llevar a cabo esta arquitectura es que facilita mucho el entendimiento del código para terceras personas que quieran saber la función de cada elemento y donde está implementado.

Angular.js, al igual que Node.js, tiene un gestor de paquetes que se denomina BOWER. Funciona de la misma forma que NPM, existe un archivo llamado bower.json en el cual se incluyen todas las dependencias de módulos que se usen en el proyecto con Angular.js, y a través de sencillos comandos se actualizan o instalan nuevos módulos.

El archivo bower.json que se ha usado para el desarrollo del presente trabajo es el siguiente.

```
//Esta parte del archivo recoge los datos identificativos de la aplicación//
{
  "name": "enginsave",
  "version": "1.0.0",
  "description": "Fullstack website with MongoDB, Express,
AngularJS, and Node.js.",
//Esta parte del archivo es donde se rocogen todas las dependencias de la
aplicación y la versión con la que se trabaja de esos módulos//

  "dependencies": {
    "angular-bootstrap": "~0.12.0",
    "angular-resource": "~1.2",
    "angular-ui-utils": "~0.1.1",
    "angular-ui-router": "~0.2.11",
    "chartist": "~0.9.0",
    "angular-ui-calendar": "~1.0.0",
    "fullcalendar": "~2.3.2"
  }
}
```

Otra de las características más populares de Angular.js es su flujo de datos bidireccional, que permite a una aplicación tener siempre sincronizados los datos con la vista y viceversa. La Figura 25 muestra un esquema de su funcionamiento en una aplicación real.

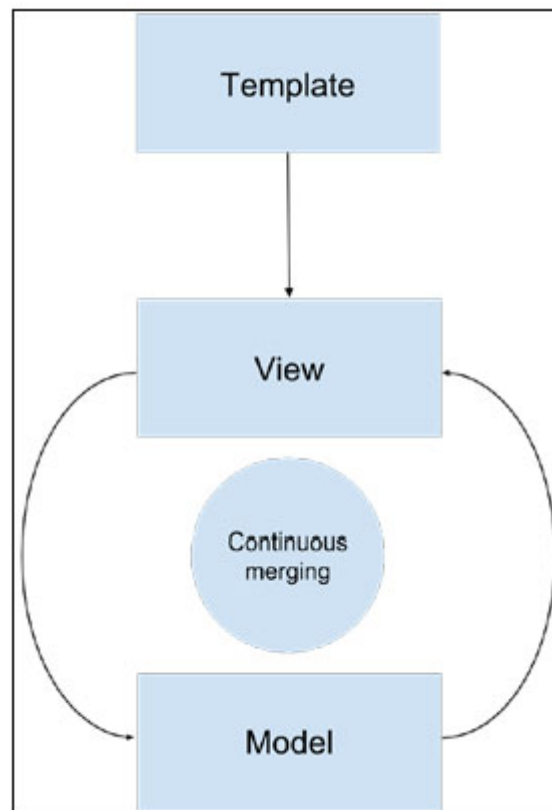


Figura 25. Funcionamiento del flujo de datos bidireccional en AngularJS

La idea es sencilla, la vista se carga a través de la plantilla, la cual se enriquece con los datos que se extraen del modelo, y una vez realizada la carga, esos datos pueden ser cambiados tanto en el modelo como en la vista. Ambos al estar en continua sincronización tendrán los datos siempre actualizados de forma instantánea.

## 5.5 MONGO DB

MongoDB [4] es un nuevo concepto de base de datos, las denominadas bases de datos NoSQL. Surgió debido a la necesidad de romper con el esquema fijo de tablas que ofrece SQL e introducir dinamismo dentro de las bases de datos para poder almacenar información no estructurada.

Esto es un gran avance, ya que la cantidad de datos que se deben almacenar desde hace unos años hasta el día de hoy han evolucionado exponencialmente. Actualmente se generan 2,5 exabytes de datos (2,5 millones de gigabytes), de los cuales cerca del 75% de los datos no son estructurados y provienen de redes sociales, mensajería instantánea y otros [8].

Este concepto abre nuevas puertas de negocio a través de la explotación de datos no estructurados, ya que una vez almacenados a través del uso de técnicas de Big Data o minería de datos, se podría extraer información clave para el éxito de proyectos tales como el emplazamiento de un nuevo bar de copas, o la apertura de un nuevo banco. Todavía es un concepto que está en plena expansión, pero está claro que va a ser transcendental.

Algunas de las principales características técnicas de MongoDB se enumeran a continuación:

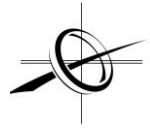
- Es capaz de actualizarse sin dejar de dar servicio.
- No tiene los cuellos de botella que se producen en las bases de datos relacionales (RDBMS) al procesar grandes cantidades de información.
- Utiliza objetos JSON para guardar y transmitir la información. JSON es el estándar web hoy en día, por lo que supone una gran ventaja que web y base de datos hablen el mismo lenguaje.
- Permite utilizar Map-Reduce para el procesado de la información a través de funciones JavaScript que se ejecutan en los servidores.
- Puede almacenar y ejecutar funciones JavaScript en el servidor.
- Tiene una potente herramienta web que nos permite monitorizar todo lo que pasa en nuestras bases de datos y en nuestras máquinas, MongoDB Management Service (MMS):



La sintaxis de MongoDB cambia con respecto a SQL, la gran diferencia entre es que el primero no es una simple sentencia, este no requiere de espacios en blanco entre palabras, o comas o paréntesis, o caracteres entrecomillados. MongoDB utiliza una "cascada" de las estructuras del operador/operando, normalmente en pares clave: valor. La equivalencia de queries entre SQL y Mongo DB con respecto al interfaz CRUD (Consulta, Inserción, Borrado y Actualización) se muestran en la siguiente tabla (Figura 26).

SQL Statements		MongoDB Statements
<pre>SELECT * FROM users</pre>	➡	<code>db.users.find()</code>
<pre>INSERT INTO users(user_id,                   age,                   status) VALUES ("bcd001",         45,         "A")</pre>	➡	<pre>db.users.insert(   { user_id: "bcd001", age: 45, status: "A" } )</pre>
<pre>UPDATE users SET status = "C" WHERE age &gt; 25</pre>	➡	<pre>db.users.update(   { age: { \$gt: 25 } },   { \$set: { status: "C" } },   { multi: true } )</pre>
<pre>DELETE FROM users WHERE status = "D"</pre>	➡	<code>db.users.remove( { status: "D" } )</code>

Figura 26, Tabla de equivalencias entre SQL y MongoDB



## 6. ESTRUCTURA DE LA APLICACIÓN

En este apartado se definirán los componentes técnicos más importantes adoptados en la aplicación.

Referente a la seguridad de la aplicación, no se usa nunca la clave de usuarios en claro, ya que se usa cifrado SALT. Durante este proceso a la clave introducida por el usuario se le añade una clave secreta, que en ningún momento es conocida por nadie, para que la contraseña sea resistente a ataques realizados mediante diccionario, y además se cifra con el método base64. La última medida de seguridad adoptada se pone en marcha en cuanto el usuario ha iniciado sesión, en ese momento se elimina la clave cifrada de las transmisiones para evitar la posible interceptación del mensaje.

En cuanto al almacenamiento de los movimientos de BBDD, para evitar inconsistencias se ha desarrollado un sistema de clave única en el cual se combina el nombre de usuario registrado en la aplicación con un secuencial automático que se autoincrementa cada vez que se añade un nuevo movimiento.

La arquitectura aplicada en este proyecto es MVC, que ya se definió previamente. Para implementar esta arquitectura se ha utilizado un esquema de carpetas horizontal, en el cual se juntan los archivos del mismo tipo en carpetas (controladores con controladores, modelos con modelos...), debido a que no es una aplicación excesivamente grande. Si en un futuro se plantease ampliarla cambiaríamos al esquema de carpetas vertical (archivos agrupados por funcionalidades).

Este esquema se ha empleado tanto en frontend, como en backend, y para exponer la arquitectura se mostrarán y definirán las carpetas y archivos más importantes de la aplicación:

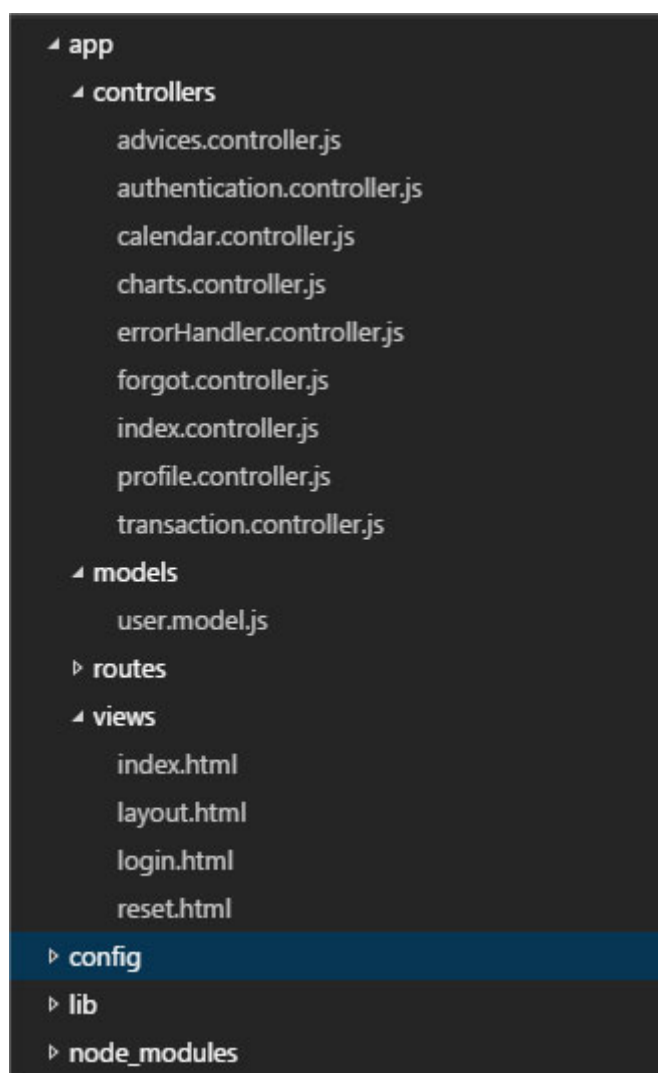


Figura 27. Esquema de carpetas horizontal backend

Como se observa en la Figura 27, la carpeta app es la carpeta que contiene toda la parte de backend de la aplicación. En su interior se encuentran las siguientes carpetas:

- Controllers, contiene los controladores de la parte de backend de la aplicación, la carpeta.
- Models, en la cual se encuentran los esquemas de datos con los que se van a trabajar.
- Routes, contiene los archivos de configuración de los paths de nuestra aplicación.
- Views, contiene las plantillas en HTML que serán utilizadas más tarde por la parte de frontend para mostrar el contenido de las páginas web.

En la Figura 27 también se pueden visualizar las carpetas Config, en la cual se almacenan todas las configuraciones referentes al backend de la aplicación, y en el caso de la aplicación, la carpeta contiene los datos de conexión a BBDD, los datos de configuración del evento, las estrategias de seguridad seguida y la configuración del servidor express. En la carpeta Lib se

almacenan las librerías de aplicaciones que han sido creadas por el desarrollador, el conector con la fuente externa Twitter y los métodos creados para el manejo de la BBDD MongoDB. La carpeta restante, denominada `node_modules`, contiene todos los módulos incluidos en las dependencias de nuestro fichero `package.json`.

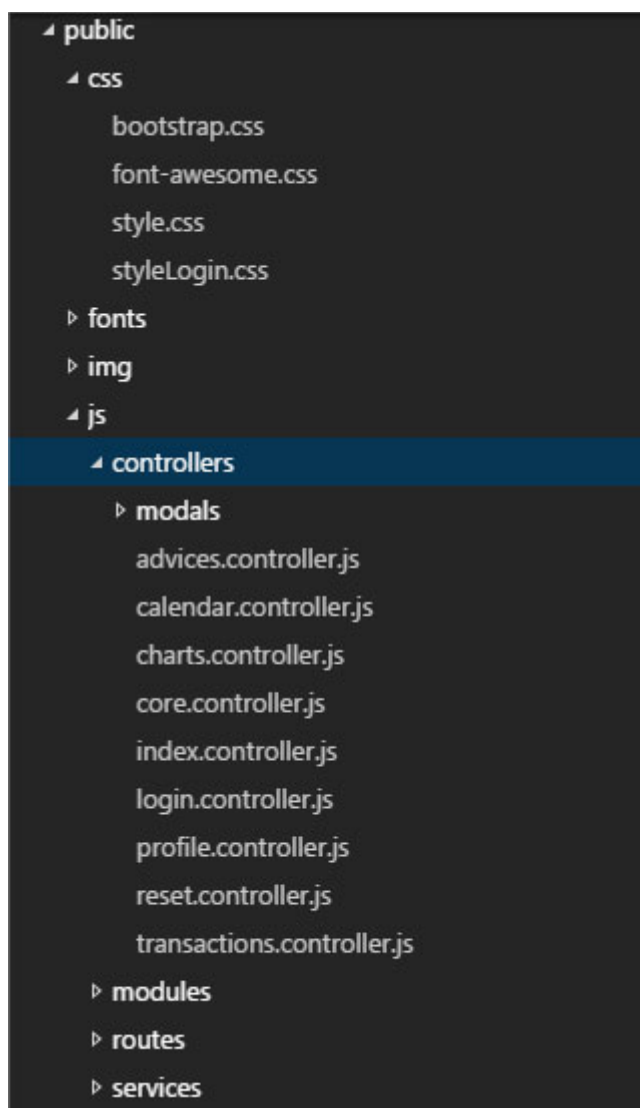


Figura 28. Esquema de carpetas horizontal en frontend

En la Figura 28, observamos la estructura de carpetas utilizadas en la parte de frontend. La carpeta que contiene todos los archivos de la parte de frontend es la carpeta `public`, la carpeta es denominada de esta forma debido a que es de acceso público para poder mostrar en la web objetos como por ejemplo imágenes. Estos permisos de acceso proporcionan desde el archivo de configuración de servidor `express`.

Dentro de la carpeta raíz de la parte de frontend (la carpeta public) se encuentran las siguientes carpetas:

- Css, carpeta que contiene todos los archivos de hojas de estilo utilizados por la aplicación.
- Fonts, carpeta que incluye todas las fuentes tipográficas que se utilizan en la aplicación.
- Img, carpeta que contiene todas las imágenes utilizadas en la aplicación.
- Lib, carpeta que contiene todos los módulos que enriquecen Angular.js y que vienen incluidos como módulos de dependencia en el archivo bower.json. Contiene otros módulos como por ejemplo chartist o bootstrap.
- Views, carpeta que contiene el código html enriquecido con las variables de Angular.js para la creación de una aplicación web dinámica.

Para el último lugar se ha dejado la carpeta JS, ya que en ella se entrará más en detalle puesto que contiene una gran parte del grueso de frontend. Esta carpeta contiene los controladores (carpeta controllers) de Angular.js, los modales de las ventanas emergentes (carpeta modals dentro de controllers), los módulos de los controladores (carpeta models), los servicios (carpeta services) creados que utiliza Angular.js, los archivos de enrutamiento de la aplicación (carpeta routes), que es la parte que define qué se muestra o qué se debe cargar al introducir una dirección en el navegador.

En la parte de backend existen varios archivos fundamentales para el funcionamiento del servidor web, y uno de ellos es el archivo express.js, que se encuentra en la carpeta config, este archivo es el responsable de la carga de todas las dependencias necesarias para que el servidor funcione de la forma deseada. A continuación se muestran las partes más significativas del archivo express.js:

```
//Zona de carga de módulos, en esta zona se cargan todas las dependencias//
```

```
var express = require('express'),  
    mongoose = require('mongoose'),  
    morgan = require('morgan'),  
    flash = require('express-flash'),  
    session = require('express-session'),  
    cookieParser = require('cookie-parser'),  
    database = require('./database.js'),  
    bodyParser = require('body-parser'),  
    add_flash = require('add-flash'),  
    swig = require('swig'),  
    passport = require('passport'),  
    mongoStore = require('connect-mongo')({  
        session: session  
    }),
```

```
//Carga de las configuraciones del entorno//
```

```
config = require('./env/all.js'),  
methodOverride = require('method-override');
```

```
//Conexion con la BBDD y carga del modelo de datos de usuario//
```

```
db = mongoose.connect(database.url);  
require('../app/models/user.model.js');
```

```
//Definicion de la ruta publica de los archivos estáticos//
```

```
app.use(express.static('./public'));
```

```
//Configuracion del motor de plantillas swig//
```

```
app.engine('html', swig.renderFile);  
app.set('view engine', 'html');  
app.set('views', './app/views');  
swig.setDefaults({varControls:['[', ']]']});
```

```
//Almacenamiento de las sesiones en MongoDB//
```

```
app.use(session({  
    saveUninitialized: true,  
    resave: true,  
    secret: config.sessionSecret,  
    store: new mongoStore({  
        url: database.url,  
        collection: config.sessionCollection  
    }),  
    cookie: config.sessionCookie,  
    saveUninitialized: true,  
    name: config.sessionName  
}));
```

```
//Carga de passport para el manejo de Login//
```

```
app.use(passport.initialize());  
app.use(passport.session());
```

Otro archivo importante es el archivo llamado `server.js`, que está en la raíz de la aplicación y se encarga de levantar una estancia de nuestra aplicación en un puerto determinado. A continuación se muestran las partes más importantes de este fichero.

```
//Como se observa existen dos variables de entornos OPENSIFT, estas  
variables sirven en la puesta en producción de la aplicación aporrandos los  
datos de conexión//
```

```
var server_port = process.env.OPENSIFT_NODEJS_PORT || 8080;  
var server_ip_address = process.env.OPENSIFT_NODEJS_IP || '127.0.0.1';
```

```
//Ponemos a escuchar nuestro servidor arrancado y ya estará preparado para  
recibir peticiones//
```

```
app.listen(server_port, server_ip_address, function(){
```

Para arrancar la aplicación basta con colocarnos con la consola en la carpeta de nuestra aplicación e introducir el comando `node server.js`

En cuanto a la parte de frontend nos encontramos también con dos archivos fundamentales, uno de ellos es el archivo `application.js`, que se encuentra en el path `public/js`, este archivo es el encargado de configurar el módulo principal y las dependencias de los módulos, del cual se muestran varios fragmentos importantes a continuación.

```
//Llamada para crear la aplicación Angular.js compuesta por el modulo  
principal de Angular.js y el resto de submódulos//
```

```
angular.module(ApplicationConfiguration.applicationModuleName,  
ApplicationConfiguration.applicationModuleVendorDependencies);
```

```
//Configuración de modo de enrutamiento de HTML5, es decir el protocolo que  
va a seguir los path//
```

```
angular.module(ApplicationConfiguration.applicationModuleName).config(['$locationProvider',  
function($locationProvider) {  
    $locationProvider.html5Mode({  
        enabled: true,  
        requireBase: false
```



El otro archivo de gran importancia en la parte de frontend, es el archivo `config.js` también ubicado en el path `public\js`. A continuación se muestra el contenido más relevante de este archivo:

```
//Configuracion de las opciones del modulo principal//

var applicationModuleName = 'enginsave';
var applicationModuleVendorDependencies = ['ngResource', 'ui.router',
'ui.bootstrap', 'ui.utils'];

//Funcion de creacion de nuevos modulos en Angular//

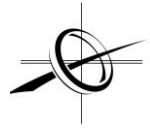
var registerModule = function(moduleName, dependencies) {

    angular.module(moduleName, dependencies || []);
    angular.module(applicationModuleName).requires.push(moduleName);
};

//Devolucion del modulo principal con los nuevos modulos incluidos//

return {
    applicationModuleName: applicationModuleName,
    applicationModuleVendorDependencies:
applicationModuleVendorDependencies,
    registerModule: registerModule
}
```

Los archivos definidos eran los principales, pero la aplicación se compone de muchos más como por ejemplo controladores, modelos, servicios, plantillas, archivos estáticos...



## 7. PUESTA EN PRODUCCIÓN DE LA APLICACIÓN

Para la puesta en producción de la aplicación web se buscó un hosting que se ajustase a las necesidades del trabajo, ya que no todos están preparados o soportan Node.

Se barajó la opción de 1&1 y estuvo a prueba un par de días pero su configuración era difícil ya que el hosting no incluye Node.js y da muchos problemas, debido a que 1&1 no soporta la ejecución continua de un proceso. Seguidamente se probó la capa gratuita de Amazon Web Services, Amazon EC2, la cual sí que es compatible pero tenía sus limitaciones en cuanto a horas de ejecución, accesos, ancho de banda...

Finalmente se hayo la solución más apropiada, Openshift, esta solución soporta Node perfectamente y también proporciona la BBDD MongoDB. A continuación se explicará qué es Openshift y cómo se configura una aplicación web con MEAN.js.

### 7.1 ¿QUÉ ES OPENSIFT?

OpenShift [14] Online es una nube pública de desarrollo de aplicaciones de Red Hat y una plataforma de alojamiento que automatiza el aprovisionamiento, la gestión y la ampliación de las aplicaciones. Con Openshift se puede tener hasta 3 aplicaciones de forma gratuita y con soporte para BBDD.

### 7.2. CONFIGURACIÓN DE UNA APLICACIÓN DE NODE CON OPENSIFT

Para configurar una aplicación en Openshift lo primero que se debe hacer es añadir una aplicación como se indica en las siguientes figuras.

En esta figura (Figura 29) se observa la pantalla principal de aplicaciones de Openshift, donde se muestran las aplicaciones que hay creada, como se muestra ya hay una aplicación creada que es la correspondiente a la creada para este trabajo. Para ver en detalle el proceso de creación se añadirá una nueva aplicación.

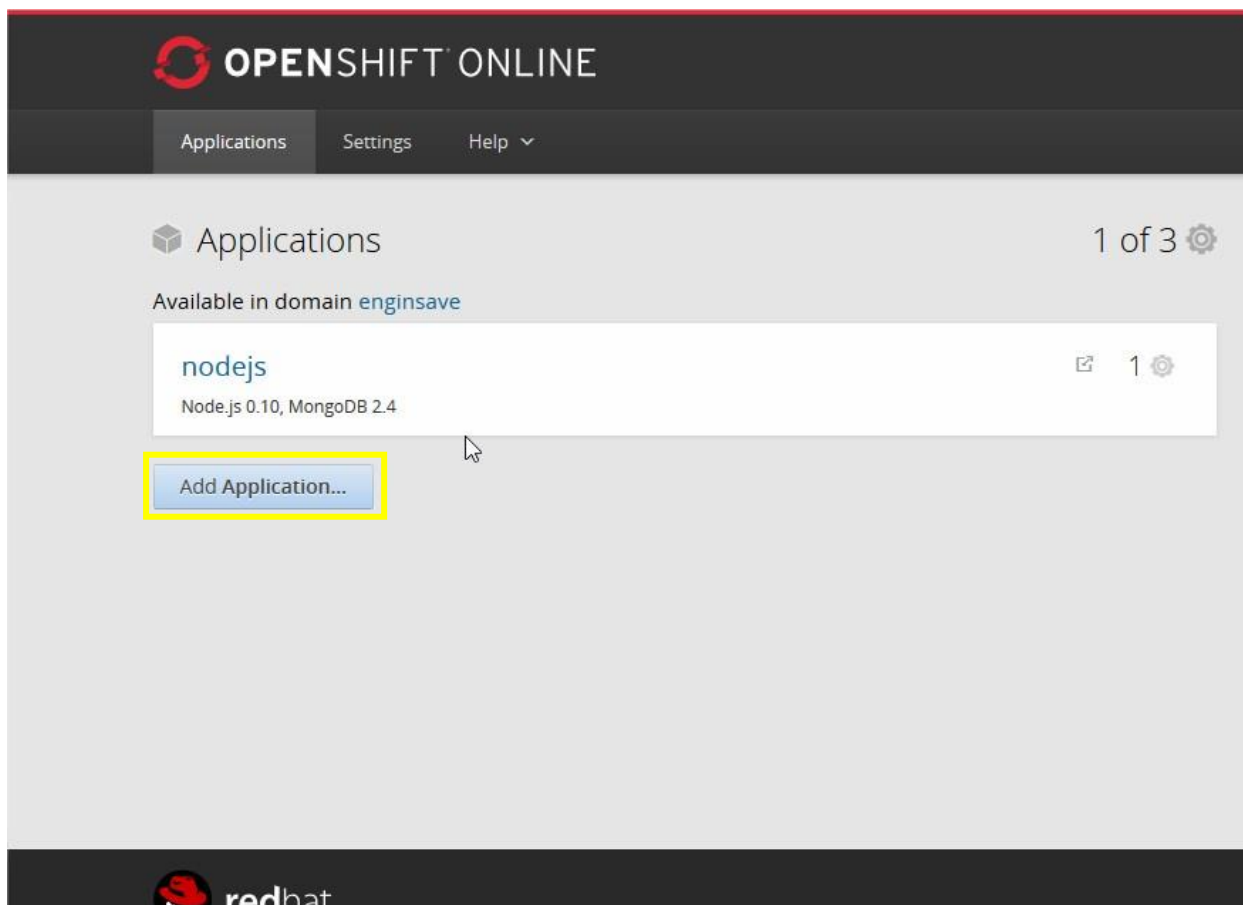


Figura 29. Como añadir una aplicación en Openshift

La siguiente parte (Figura 30) es la de selección de la tecnología que se quiere utilizar la aplicación, que en el caso del trabajo es Node, pero Openshift soporta muchas más.

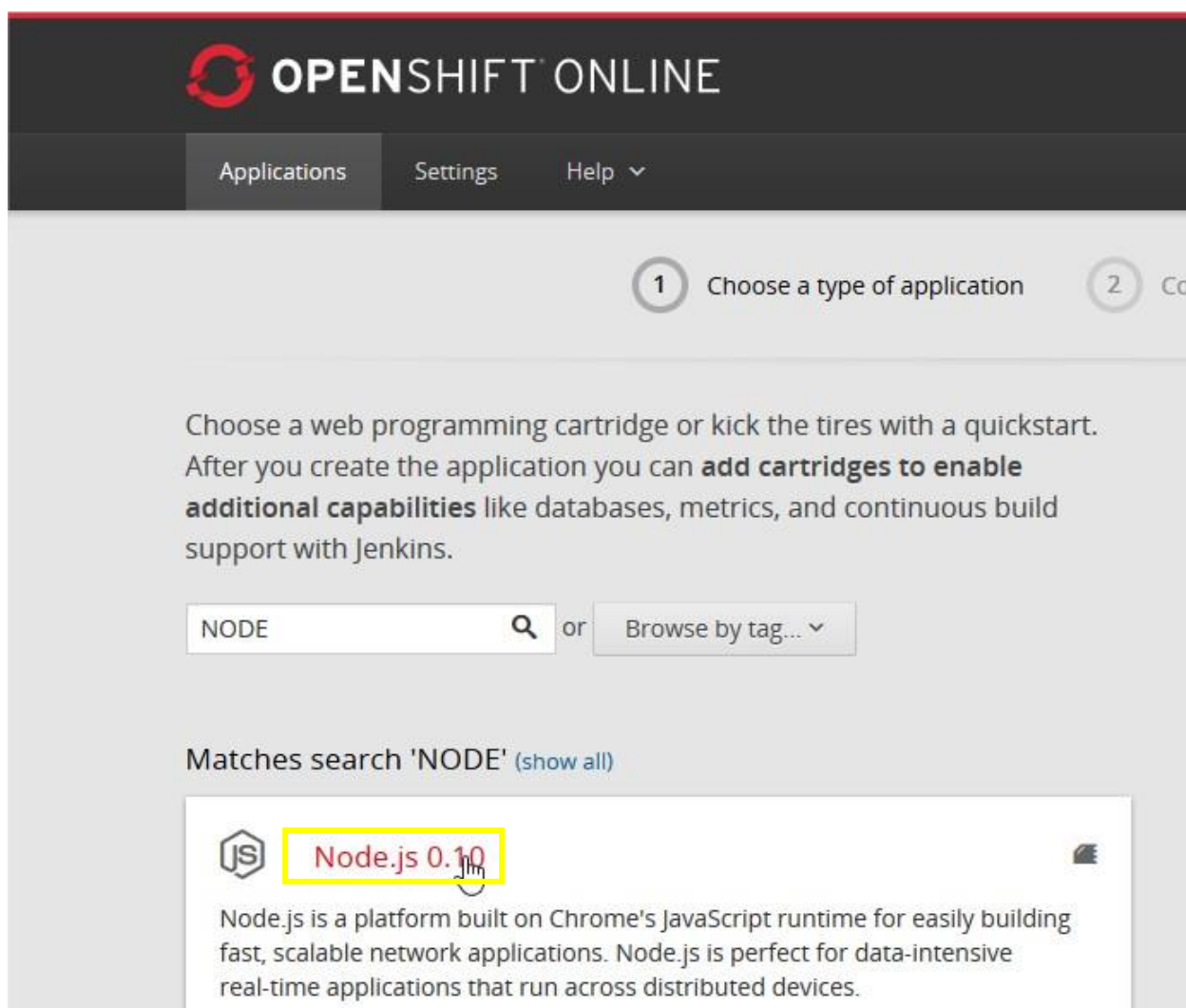


Figura 30. Pantalla de selección de la tecnología deseada en Openshift

Después de elegir Node se debe elegir un nombre de dominio Openshift para la aplicación. Una vez hecho esto ya está preparada la aplicación para poder trabajar, lo único que faltaría sería descargar el proyecto con GIT [18], y se podrá comenzar a editar, lo único que necesitaremos para desplegar la aplicación será un sencillo PUSH (Orden de GIT) del proyecto.

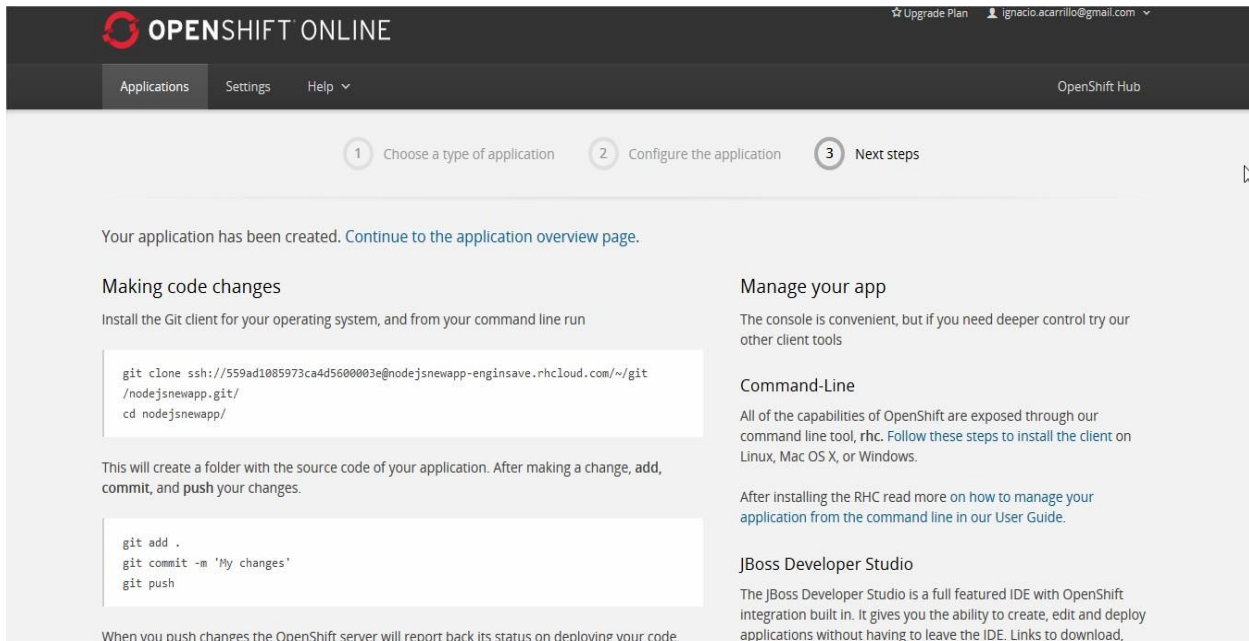


Figura 31. URL de GIT para poder descargarnos y editar nuestro proyecto de Openshift

Para poder conectarse remotamente al servidor de la aplican se deben generar unas claves para que el servidor pueda autenticar la máquina del administrador de la aplicación y para evitar ataques externo. Estas claves se generan a través de la consola de LINUX con el comando **ssh-keygen -t dsa** de la herramienta SSH. La clave pública generada debe añadirse a las claves de Openshift y la privada debe conservarse en la máquina del administrador, una vez hecho esto ya se podrá establecer una conexión a través de Secure Shell (SSH) con el servidor.

Llegado a este punto sólo faltaría agregar una BBDD para la aplicación, lo cual también es bastante sencillo. Lo único que se debe hacer es seleccionar la que se prefiera en la pantalla principal del gestor de la aplicación creada, esto generara unos datos de conexión para una BBDD.

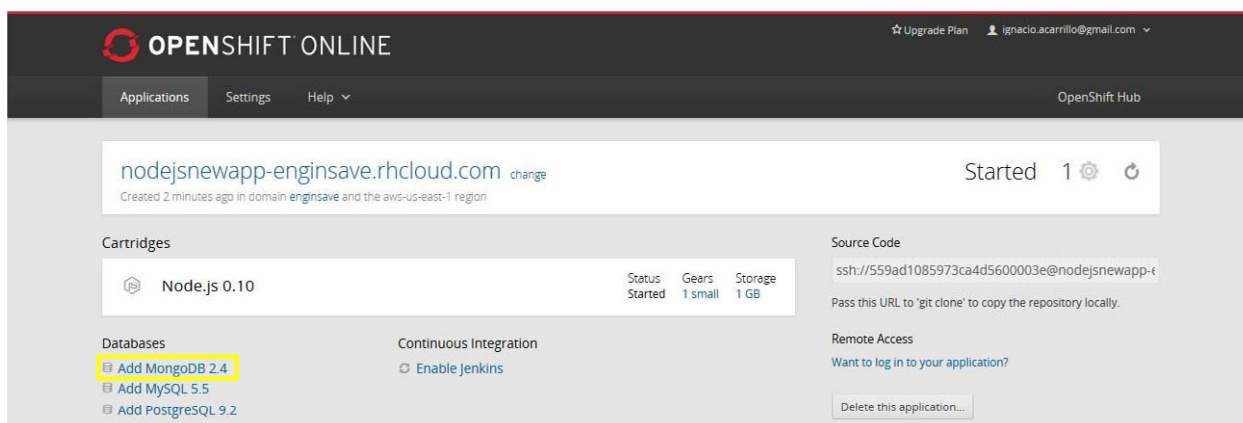


Figura 32. Añadiendo la BBDD a nuestra aplicación en OpenShift

Una completado todo se debe configurar los parámetros de entorno de la aplicación, que son variables que toman un valor determinado dependiendo del ámbito en el que se ejecuten, es decir, una variable que contiene la contraseña de acceso a la base de datos, si es ejecutada en el entorno de desarrollo tomará el valor de la contraseña de la BBDD de desarrollo, pero si es ejecutada en producción, tomará el valor de la contraseña de la BBDD de producción. Una vez realizada esta explicación, debemos configurar todas las variables de conexión a OpenShift (producción) para que la aplicación funcione correctamente. En las siguientes figuras se muestra la configuración que se adopta para las variables de entorno de las BBDD y el servidor web.

```
//Configuraciones de conexion a la BBDD MongoDB

if(process.env.OPENSIFT_MONGODB_DB_PASSWORD){
  url = "mongodb://" + process.env.OPENSIFT_MONGODB_DB_USERNAME + ":" +
  process.env.OPENSIFT_MONGODB_DB_PASSWORD + "@" +
  process.env.OPENSIFT_MONGODB_DB_HOST + ':' +
  process.env.OPENSIFT_MONGODB_DB_PORT + '/' +
  process.env.OPENSIFT_APP_NAME;
}
else{
  url = 'mongodb://127.0.0.1:27017/prueba';
}

module.exports = {
  url:url
}
```

Figura 33. Variables globales de configuración para la BBDD

En la Figura 33, se muestran las variables globales de configuración de BBDD. A continuación se definirán estas variables:

- `process.env.OPENSIFT_MONGODB_DB_USERNAME`: esta variable contiene el nombre de usuario para acceder a la BBDD MongoDB del hosting Openshift.
- `process.env.OPENSIFT_MONGODB_DB_PASSWORD`: esta variable contiene la contraseña para acceder a la BBDD MongoDB del hosting Openshift.
- `process.env.OPENSIFT_MONGODB_DB_HOST`: esta variable contiene la dirección ip para acceder a la BBDD MongoDB del hosting Openshift.
- `process.env.OPENSIFT_MONGODB_DB_PORT`: esta variable contiene el puerto para acceder a la BBDD MongoDB del hosting Openshift.
- `process.env.OPENSIFT_APP_NAME`: esta variable contiene el nombre de la aplicación, que será el nombre de la BBDD MongoDB del hosting Openshift.

Uniando todas estas variables se conformará la url de conexión a la BBDD MongoDB de Openshift.

```
//Archivo de arranque del servidor

var express = require('./config/express')

var app = express();

require('./config/passport')();

//Configuracion de la direccion y puerto a usar
var server_port = process.env.OPENSIFT_NODEJS_PORT || 8080;
var server_ip_address = process.env.OPENSIFT_NODEJS_IP || '127.0.0.1';

app.listen(server_port, server_ip_address, function(){
  console.log("Listening on " + server_ip_address + ", server_port " + server_port)
});
```

Figura 34. Variables globales de configuración para el servidor



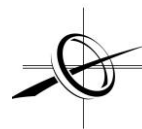
En la Figura 34, se muestran las variables globales de configuración del servidor web MEAN.js. A continuación se definirán estas variables:

- `process.env.OPENSIFT_NODEJS_PORT`: esta variable contiene el puerto de nuestra aplicación web en hosting Openshift.
- `process.env.OPENSIFT_NODEJS_IP`: esta variable contiene la dirección ip de nuestra aplicación web en hosting Openshift.

Si las variables de entorno de Openshift estuviesen vacías o no existieran, se supondrá que la ejecución de la aplicación no es en el entorno productivo y se cargarían las configuraciones de las variables de entorno de desarrollo.

Un aspecto muy importante en Openshift es que se debe tener bien actualizado el archivo `package.json`. Como se ha comentado anteriormente en este trabajo, el archivo `package.json` es el archivo encargado de indicar a NPM las dependencias de la aplicación web creada, en este caso se puede observar de forma práctica la importancia de este archivo, ya que cada vez que se hace una subida a producción o se despliega una versión en el servidor, se reinstalan todos los módulos de las dependencias según se indica en el archivo, y si faltara algún módulo o la versión del módulo es incorrecta, la aplicación podría dejar de funcionar o no funcionar de forma adecuada.

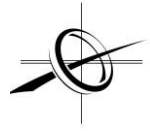
Para finalizar este apartado es recomendable establecer la conexión a través de SSH con el servidor ya que así se podrá acceder a los archivos de LOG de la aplicación, lo cual facilitará la detección de errores o malfuncionamientos. Los LOGS más importantes son los denominados `mongodb.log`, en el cual aparecen todas las trazas que tienen que ver con la base de datos, y `nodejs.log` que tiene las trazas del servidor web.



## 8. FUTURAS MEJORAS DE LA APLICACIÓN

De cara al futuro y a seguir mejorando y desarrollando la aplicación, estas son alguna de las mejoras o nuevas funcionalidades que se podrían añadir a la aplicación:

- Implantación de la búsqueda avanzada: en la aplicación ya aparece esta funcionalidad pero todavía no está implementada, esta funcionalidad permitiría al usuario filtrar los movimientos por fecha, cantidad gastada o ingresada y categoría.
- Mejora de la parte de calendario: el calendario que hay en la aplicación muestra los movimientos por fecha gráficamente, pero no se puede acceder a ellos desde el mismo, la mejora que se realizaría sería enlazar directamente esos movimientos para poder editarlos de forma directa.
- Enlace con cuentas bancarias: aunque con ese aspecto no se estaba muy de acuerdo, para muchos usuarios podría ser mucho más cómodo, ya que no se precisaría añadir manualmente los movimientos, esta funcionalidad se implementaría de forma totalmente segura teniendo solo funciones de lectura para evitar robos y movimientos en las cuentas. Aunque esta funcionalidad fuese implantada en la aplicación se conservaría el método manual para aquellos usuarios que prefieran no dar sus datos bancarios.
- Envío automático de avisos: esta función automatizaría un servicio de mailing para avisar a los usuarios sobre nuevos movimientos (en el caso de las sincronizaciones con cuentas), y también avisaría de los gastos mensuales programados.
- Planificador para compra: esta es una funcionalidad interesante pero que debería ser bien planificarla para que sea útil para los usuarios. La funcionalidad pretende decir al usuario dónde puede reducir gastos para que pueda planificar la compra de algún artículo o servicio. A esta funcionalidad se le debería introducir el importe de lo que se quiera adquirir y una fecha, y esta hará los cálculos de lo que el usuario debería ahorrar mensualmente de cada categoría para que pueda llegar a adquirirlo en la fecha deseada.



- Enriquecimiento del perfil de usuario: este apartado es uno de los que en un futuro sería necesario mejorar, por ejemplo añadiendo avatares o imágenes al usuario, una descripción o un apartado donde se pueda introducir un sueldo mensual y que se calcule automáticamente en la aplicación.

## 9. CONCLUSIONES

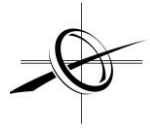
Aunque el uso de las tecnologías que componen MEAN.js pueda parecer trivial, en realidad es una idea equivocada. La utilización de MEAN requiere unos amplios conocimientos de JavaScript por descontado, y aparte un estudio de los frameworks por separado, ya que cada uno tiene su peculiaridad, véase el asincronismo en Node.js, la forma de trabajo basada en documentos de MongoDB, el enrutamiento de Express.js, las plantillas, enrutamiento, y trabajo con modelos en Angular.js...

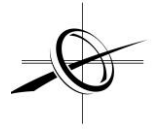
Todo esto me ha llevado mucho tiempo y esfuerzo, ya que ha requerido de meses de estudio previo antes de lanzarse a montar el rompecabezas completo. Aunque ya tenía algo de experiencia en HTML y PHP esto ha sido completamente nuevo y diferente, la forma de trabajo, establecimiento de un orden adecuado para la estructura del proyecto y la configuración adecuada para que se mostrase en todos los dispositivos adecuadamente.

De las características que ofrece MEAN.js lo que más me ha gustado es la de poder crear una web single-page y su actualización de datos en tiempo real sin necesidad de tener que recargar la página, esto último me parece realmente novedoso y útil a la hora de trabajar con una aplicación web.

En mi opinión MEAN.js es muy potente y me ha introducido en un mundo sobre el que no había aprendido nada en el grado, lo cual me abre muchas puertas profesionalmente ya que es una tecnología muy novedosa, potente y que ahora mismo está en auge. El proyecto no ha sido fácil pero me ha enseñado y aportado muchas cosas, tanto de backend como de frontend, he aprendido como funciona y se implementa una aplicación a todos los niveles y desde cero, lo que me da una visión más amplia de cómo funcionan las cosas.

Estoy satisfecho con el trabajo desarrollado y me alegro de haber escogido una tecnología tan novedosa para desarrollarlo.





## BIBLIOGRAFÍA

### *Libros*

- [1] BROWN, Ethan. *Web Development with Node and Express*; ed: O'REALLY.
- [2] ALI SYED, Babar. *Beginning Node.js*; ed: APRESS.
- [3] GREEN, Brad; SESHADRI, Shyam. *AngularJS*; ed: O'REALLY.
- [4] HOWS, David; MEMBREY, Peter; PLUGGE, Eelco. *MongoDB Basics*; ed: APRESS.
- [5] HAVIV, Amos Q. *MEAN Web Development*; ed: Packt Publishing.

### *Artículos*

- [6]<http://www.elandroidelibre.com/2014/03/monefy-anota-y-controla-de-forma-sencilla-tus-gastos-e-ingresos.html>
- [7]<http://www.xatakamovil.com/aplicaciones/fintonic-se-actualiza-segundo-asalto-al-mundo-de-la-economia-personal>
- [8] [http://www.bbc.com/mundo/noticias/2014/03/140304\\_big\\_data\\_grandes\\_datos\\_rg](http://www.bbc.com/mundo/noticias/2014/03/140304_big_data_grandes_datos_rg)
- [9]<http://venturebeat.com/2015/08/19/here-are-the-top-10-programming-languages-used-on-github/>

### *Direcciones Web.*

- [10]<http://expressjs.com/es/>
- [11]<https://docs.npmjs.com/>
- [12]<https://www.fintonic.com>
- [13]<http://www.monefy.me/>
- [14]<https://developers.openshift.com/en/overview-what-is-openshift.html>
- [15]<http://meanjs.org/docs.html>
- [16]<https://github.com/Unitech/pm2>
- [17]<https://play.google.com/store>
- [18]<https://git-scm.com/>