UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Diseño de un sistema de gestión segura de eventos de calendario en la nube

Rodrigo Xavier Pico Paredes
Tutor: David Arroyo Guardeño

Julio 2015

AGRADECIMIENTOS

Quisiera agradecer y mencionar aquí a:

- Mi familia por soportarme y apoyarme durante toda mi vida, pero querría hacer una especial mención a mi madre, que es la que mayormente me ha apoyado en los malos momentos y facilitándome las tareas para que yo dedicara mi tiempo en acabar la carrera con éxito.

- Mis tíos y primos que han tenido que irse, ellos saben que no hablo mucho con ellos, pero que siempre los tengo presentes. Ya que ellos me cuidaron cuando no podían mis padres durante mi infancia más cercana.

- Mis otros tíos que están en el lugar donde nací. Ellos me mimaron durante mi niñez y me siguen queriendo demasiado, más de lo que me merezco.

- Mi otra gran familia del norte que me hacen disfrutar cuando voy allí de vacaciones.

- Mis amigos del colegio, instituto y universidad que me han ayudado a llegar hasta donde he llegado. Pero todavía queda un largo camino que recorrer.

- Y finalmente, y no por ello menos importante, al gran tutor que he tenido, David Arroyo, que me ha ayudado a llevar este proyecto a buen término, y al cuál considero como uno de los mejores profesores que he tenido durante mi vida. Esto puede ser debido a que como profesor es bueno, pero como persona lo es aún más.

A todos vosotros muchísimas gracias.

RESUMEN

En la actualidad cada vez son más las empresas que ofrecen servicios en la nube facilitando a los usuarios el acceso a las mismas. Un gran conjunto de personas utilizan estos servicios, tanto es sus momentos de ocio como para realizar sus trabajos, sin preocuparse de qué hacen con sus datos o con qué finalidad los utilizan los proveedores de servicios. Es por ello que la seguridad se ha vuelto un aspecto importante cuando se manejan los servicios *cloud*, para asegurar la confidencialidad y privacidad de la información que se ingresa en estos entornos.

Los gestores de eventos de calendario en la nube no son muy populares entre la sociedad, es decir, no es que los utilicen un gran número de usuario, pero al tratarse también de servicios *cloud*, no están exentos de los problemas antes mencionados. Debido a esto se crea la necesidad de construir una aplicación que realice estas tareas, pero sin comprometer los datos de los usuarios. Por ello, se abordó el desarrollo de este proyecto. Crear un gestor de eventos de calendario de forma segura en la nube, para intentar ofrecer a los usuarios, una herramienta que tenga como objetivo principal otorgar la máxima seguridad y privacidad, sin olvidarse también de la funcionalidad, y no la recolección de datos personales para lucrarse de los mismos. Además, si se ofrece un sistema libre, son los usuarios y/o los desarrolladores los que tienen el máximo control sobre sus datos y/o el servicio.

En este documento se expone las fases por las que pasó la creación de la herramienta software. Haciendo un estudio inicial del mercado para observar que servicios parecidos se ofrecen, para obtener las funcionalidades que requerirán los usuarios por el sistema. Después de concretar este aspecto, se procedió a analizar cómo y con qué mecanismos de seguridad y herramientas tecnológicas, a poder ser libres, se podría desarrollar la aplicación. Una vez, se realizó el análisis y el diseño del cliente software se procedió a crear el mismo. Tras finalizar esta fase, se procedió a la validación del cliente web en distintos entornos para que el servicio se adecúe a estos, y así brindar un servicio satisfactorio a los usuarios que lo utilicen. Con todo ello, logrando los objetivos de seguridad, funcionalidad y libertad, sin anteponer ninguno contra otro.

PALABRAS CLAVE

JavaScript, jQuery, CSS, HTML, Google Drive, Google Plus, GPG, OpenPGP, Servidor de Claves PGP, Código Libre, Criptografía, Cliente Web, Nube, iCalendar, Aplicación de una sola página, Servicio en la nube, Gestor de eventos de calendario.

ABSTRACT

Today there are more and more companies supplying cloud services and providing users access to them. A big group of people uses these services, either to carry out personal and private activities in their spare time or to perform their work without worrying about how their data is handled and what service providers intend to do with their information assets. This is the reason that makes very important to assess cloud services when they are selected as preferential option to manage our information. In this regard, it is necessary to confirm that cloud service providers guarantee not only the confidentiality of our information, but also privacy must be properly respected.

One example of cloud services is given by cloud-based manager of calendar events. Although this kind of cloud services is not very popular, it endorses a very interesting use of the cloud in order to handle calendar events from multiple and very different locations. Nevertheless, it is difficult to find implementations guaranteeing the control of calendar events without eroding users' privacy. Therefore, in this project we face this need to create a software product to handle calendar events in a secure and private respectful way. In addition, we have considered that security should not imply an obstacle when using such a product. In other words, usability is a major concern in this project. Finally, we have adopted the OpenSource methodology as bottom line of both the design and implementation of the final solution associated to this bachelor project. Certainly, OpenSource software enables users and/or developers evaluation, and thus can be considered as an income to further conclude the security of software products.

This document describes the different stages of the software design, development, and validation. An initial study of the market is performed, to observe that similar services are offered, to determine the system functionalities that users will require. After completion of this aspect, it was analyzed the set of OpenSource APIs, tools and mechanisms necessary to develop our application. Then the analysis and design of the software client was done and the corresponding implementation was carried out. After completing this phase, web client validation was realized in different environments in order to confirm that user with different devices can access to the service satisfactorily. All in all, the objectives of security, functionality and OpenSource implementation were achieved in a coherent way fostering synergy.

KEY WORDS

JavaScript, jQuery, CSS, HTML, Google Drive, Google Plus, GPG, OpenPGP, PGP Key Server, OpenSource, Cryptography, Web Client, Cloud, iCalendar, Single Page Application, Cloud Service, Calendar events manager.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En este trabajo se afronta la necesidad de crear un **sistema para gestionar los eventos de calendario de forma segura en la nube**. Esta aplicación permite organizar la información del calendario mediante una interfaz fácil e intuitiva, esto hace que su utilización sea sencilla para los usuarios. Además, la privacidad de los datos se realiza de forma interna, para que los usuarios sólo tengan que utilizar su clave una única vez, logrando que la gestión de la seguridad siga un proceso simple, y de ese modo no sean necesarios tener grandes conocimientos de criptografía. Así se logra que toda la información se guarde confidencialmente en el servidor, y que la misma sea legible dentro del cliente web.

Por otro lado, en este apartado se tratan las razones por las cuales se ha llevado a cabo la elaboración del proyecto. Además, se especifican los objetivos que se buscan durante el desarrollo del mismo. También, se explican los estadios por los que atraviesa el proceso del producto software. Por último, se elabora una reseña sobre la estructura de la cual consta esta memoria.

1.1. MOTIVACIÓN

La seguridad de la información para algunos de los servicios *cloud* que hay en el mercado no es un aspecto fundamental, permitiendo que los datos de sus usuarios puedan sufrir posibles ataques de terceros. Igualmente, al usar estos servicios no se tiene la certeza total de cómo se administra la información que se utiliza en ellos, ya que estos sistemas suelen pertenecer a empresas privadas. Como la aplicación se basa en la gestión de eventos de calendario, se pretende tratar la necesidad de la protección de datos en este ámbito en concreto, para asegurar que los eventos de los usuarios sean confidenciales, y no se haga un uso inadecuado y no autorizado de los mismos. Suministrando una herramienta útil y segura para gestionar el calendario personal.

También, se intentan analizar las **herramientas de programación** necesarias para crear una aplicación de este tipo, y así **ampliar los conocimientos** que se tenían antes de comenzar con el proyecto. Además, de aprender cómo hay que orientar el desarrollo de un sistema donde la seguridad es un punto principal y no algo que se puede incluir después de acabar el desarrollo del software.

Otro aspecto que impulsa este trabajo es facilitar una **aplicación *OpenSource*** frente a la competencia privada, para que los usuarios tengan la seguridad y la confianza del procesamiento de su información, y para que la comunidad de desarrolladores pueda aportar mejoras al sistema o consultar toda la estructura del mismo para verificar su funcionamiento.

1.2. OBJETIVOS

Con los conocimientos adquiridos durante la formación académica, se persigue afrontar el proceso que conlleva **crear una herramienta software durante todas las fases de su desarrollo**, logrando de ese modo poner en práctica dichos conocimientos y aumentar el aprendizaje para la posterior etapa laboral. En concreto se profundizarán en los **conocimientos y competencias relativas al desarrollo de aplicaciones web** haciendo especial énfasis en la **tecnología JavaScript.** Dicho objetivo se llevará a término mediante la consideración de un caso concreto de implementación: el **diseño y desarrollo de un sistema para la gestión segura de eventos de calendario en la nube.**

En el presente proyecto se considera que el **servidor de almacenamiento en la nube no es confiable**, y por ello el usuario cifrará sus calendarios antes de guardarlos en dicho servidor. Además, se quiere que el **sistema sea lo más versátil posible y se adapte a diversos dispositivos** (ordenadores de sobremesa, portátiles, *smartphones* y *tablets*), para poder dar al usuario distintas vías para administrar sus datos personales.

Igualmente, el mecanismo para **garantizar la seguridad (confidencialidad y privacidad)** de la información en la aplicación tiene que ser sencillo y poco tedioso para el usuario. En efecto, la **usabilidad debe ser un requisito esencial** de los sistemas de gestión de seguridad. Se ha de conseguir que la experiencia del usuario con el sistema sea agradable, de modo que, la seguridad de los datos se consiga sin provocar que los usuarios piensen que el sistema desarrollado es difícil de utilizar. Así, se podrá abarcar un mayor conjunto de personas que pueden usar estos sistemas de seguridad, en este caso, la herramienta para la gestión segura de calendarios ubicados en la nube. A lo largo del diseño y desarrollo de la aplicación se establecerán los mecanismos necesarios para conseguir que la intervención del usuario en los aspectos y objetivos de seguridad sea mínima, de forma que lo único que hay que garantizar es que el usuario debidamente autenticado tiene acceso a los eventos de su calendario. Asimismo, **se quiere que los usuarios que estén acostumbrados a otros sistemas parecidos no tengan dificultades para poder usar la aplicación**, y puedan seguir realizando las funciones que hacían en ellas.

Finalmente, el software desarrollado debe tener mecanismos para controlar todas las funciones relacionadas con los eventos del calendario. **Los eventos tienen que poder ser compartidos con otros usuarios**, sin faltar a la premisa de la seguridad en todo momento. Para otorgar mayor **versatilidad y libertad** al sistema se usarán estructuras de fichero de calendario estándar, y librerías, algoritmos y herramientas que no estén sujetas al pago de ninguna licencia para su utilización. En efecto, la **adopción de estándares y soluciones OpenSource** es un criterio no solo económico, sino que también es una opción para conseguir un mayor nivel de aceptación y de seguridad para las aplicaciones [1].

1.3. PLAN DE TRABAJO

A continuación, se explican las fases seguidas durante la elaboración del proyecto. El trabajo se realizó durante los cuatro meses correspondientes a la duración de la asignatura durante el curso académico, con una dedicación diaria variable en función a la disponibilidad.

Las etapas específicas por las que ha pasado la ejecución del proyecto son:

- **Documentación y estudio del estado del arte**

En este primer estadio se analizaron los lenguajes de programación JavaScript, HTML y CSS, estos eran necesarios para crear el cliente web del proyecto. Con el lenguaje de programación JavaScript, además se tuvo que estudiar algunos *frameworks* para conocer las ventajas y desventajas de cada uno de ellos, y evaluar cuál era el que mejor se adaptaba a las necesidades del sistema. Igualmente, se consideraron distintos servicios *cloud* de almacenamiento gratuitos, así como las distintas técnicas de control de versiones de datos cifrados, y de verificación de usuarios, en este último aspecto se consideró la API de Google Plus. Asimismo, se investigaron algoritmos criptográficos para utilizar en la parte de seguridad de la aplicación. Por último, se examinaron diversos tipos y estructuras de ficheros para guardar los datos del calendario.

- **Análisis y diseño**

Una vez estudiada la teoría base para la ejecución del proyecto, se procedió a definir el catálogo de requisitos funcionales y no funcionales del sistema. También, se especificó el diagrama de flujo y los casos de uso del software. Además, se seleccionaron los mecanismos y las herramientas adecuadas para el desarrollo. Igualmente, en esta fase se definió el diseño del conjunto de módulos y la estructura del cliente web, teniendo en cuenta la escalabilidad de la aplicación.

- **Implementación**

En esta fase se desarrollaron las especificaciones de diseño que se concretaron en la etapa anterior. Como se implementaba una aplicación web no fue necesario tener un gran equipo hardware para la creación de la misma.

- **Pruebas y experimentación**

En esta etapa se definió un plan de pruebas para validar el producto software. Además, se probó el funcionamiento de la aplicación en diversos entornos disponibles (dispositivos de escritorio y móviles), y navegadores para comprobar la adaptabilidad y usabilidad del sistema. Por último, se procedió a corregir los fallos encontrados durante estas pruebas.

- **Memoria**

Tras realizar todas las fases anteriores, se continuó con la redacción de la memoria final del trabajo con la explicación de los resultados obtenidos y el software desarrollado. Finalmente, se procedió a evaluar dicha memoria para su posterior entrega.

1.4. ESTRUCTURA DE LA MEMORIA

Este documento está formado por una estructura lineal, en la cual se va explicando las fases por las que se ha pasado durante la elaboración de la herramienta software. Las partes específicamente son las siguientes:

- **Capítulo 1:** En este capítulo se expone las causas y los objetivos por los que se realizó el presente proyecto. También, se describe el plan de trabajo seguido y la estructura del documento final del TFG.

- **Capítulo 2:** En este capítulo se describe el concepto de computación en la nube. También, se detallan los aspectos de seguridad y los entornos utilizados en los servicios en la nube.

- **Capítulo 3:** En este capítulo se analiza los problemas que se quieren abordar durante la elaboración del proyecto. También, se detalla la finalidad y la funcionalidad de la aplicación a realizar. Además, se especifican las herramientas y tecnologías que se utilizarán en el desarrollo del sistema.

- **Capítulo 4:** En este capítulo se define el diseño del sistema. También, se describe la secuencia entre los diferentes estados de la aplicación y las estructuras que se deben emplear para la realización de la misma.

- **Capítulo 5:** En este capítulo se detalla el proceso de la implementación del sistema, explicando el entorno de desarrollo empleado, el código generado y el proceso de documentación de la aplicación.

- **Capítulo 6:** En este capítulo se concreta el plan de pruebas a seguir para la validación del software.

- **Capítulo 7:** En este capítulo se muestra el resultado final de la aplicación desarrollada.

- **Capítulo 8:** En este capítulo se exponen las conclusiones que se obtienen y los aspectos relevantes después de finalizar el proyecto. Así como, algunas posibles ideas que se podrían abordar en un trabajo futuro.

2. ESTADO DEL ARTE

En este capítulo se hace una breve introducción de los servicios en la nube, explicando su evolución a lo largo del tiempo. Además, se trata el tema de la seguridad en estos servicios y como hay que afrontarla para intentar conseguir la confidencialidad de la información que se deposita en ellos. También, se expone la expansión de estas tecnologías en la nube en la actualidad, detallando las características favorables y los inconvenientes que presentan la utilización de estos sistemas. Finalmente, se habla sobre el tipo de herramientas necesarias para poder hacer uso del *cloud computing*.

2.1. INTRODUCCIÓN AL CLOUD COMPUTING

En el mundo tecnológico actual la computación en la nube ha tomado gran relevancia, pero sus inicios fueron bastante inciertos allá por finales del siglo XX, cuando algunas compañías empezaron a prestar servicios *cloud*. No obstante, en esa época no terminó de funcionar ya que se vio que eran necesarios recursos más avanzados para ofrecer buenos servicios a los clientes [2].

La computación en la nube se creó como una estructura en la cual se utilizaba la capacidad de cálculo de los servidores para ofrecer servicios a los usuarios, y que estos ya no se preocuparan de tener que mantener y gestionar las aplicaciones que utilizaban [3], [4]. El término ***cloud computing*** fue relacionado con los investigadores **John McCarthy y J.C.R. Licklider**, en cuyos pensamientos se vislumbraba la concepción de Internet y de los servicios *cloud*. McCarthy no definió en sí el concepto, si no que propuso que los sistemas de "tiempo compartido" [5] ayudarían a repartir la potencia de cálculo de los ordenadores/servidores, o a crear aplicaciones que pudieran prestar servicios *online*. McCarthy planteó que "la computación algún día podría ser estructurada para ofrecerse como un servicio público" [6]. Por su parte Licklider expuso que sería posible crear una red ("Red Computacional Intergaláctica") que fuera capaz de conectar a los usuarios en distintos lugares, para que estos pudieran colaborar entre sí o tener acceso a su información personal y a aplicaciones [7].

Ya casi entrando en el siglo XXI las compañías se lanzaron con más confianza a elaborar servicios *cloud*. Algunas de estas primeras empresas fueron SalesForce, Amazon y Google. Esta última marcó un **punto de inflexión en la computación en la nube** gracias a su aplicación ***Google Docs***, la cual fue lanzada en el año 2007, y que provocó que la sociedad empezara a tener conocimiento de este tipo de servicios [8]. Esto dio lugar a la expansión del *cloud computing*, creándose distintos servicios en las diferentes capas que lo estructuraban. Estas capas eran de aplicaciones (SaaS), de plataformas (PaaS) y de infraestructura (IaaS) [9]; y algunos de los tipos de servicios

que se construyeron eran de comunicación, de búsqueda de contenidos, de administración de correo, de almacenamiento de datos, de redes sociales, de distribución de contenidos multimedia y de gestión de calendarios (Ver Figura 1).

Figura 1: Ilustración de servicios cloud computing en función del tipo.
Fuente: http://southworks.com/blog/2008/08/19/cloud-computing-taxonomy-map/

Tal fue el auge de la creación de este tipo de herramientas en la nube, que Google desarrolló, su propio sistema operativo en la nube, **Chrome OS** [10]. Este sistema se basó en la dependencia absoluta de los servicios *cloud* para realizar cualquier operación dentro del mismo. Aunque esta aplicación para controlar los equipos de los usuarios aún no tiene un gran acogimiento por parte del público en general, puede ser el principio de que los próximos dispositivos requieran la utilización de estos entornos en la nube para poder realizar los trabajos cotidianos a los cuales muchos usuarios ya se están acostumbrado.

Hoy en día, ya existen muchas empresas que ofrecen servicios *cloud* con diversas funcionalidades para el usuario, creando así un gran panorama muy variado donde elegir. Tal es la cantidad de sistemas que se han desarrollado, abarcando muchas de las tareas que se realizan habitualmente, que han permitido a sus clientes ahorrar costes en sus equipos personales. Además, con el uso de estas tecnologías se permite a los usuarios almacenar de forma deslocalizada sus recursos, lo cual hace factible su compartición entre distintos terminales de trabajo.

Pero la rápida propagación de este tipo de sistemas ha tenido también sus inconvenientes. Así, se han encontrado debilidades en los servidores donde se administran los datos, y se ha conseguido entrar en la infraestructura de los mismos, permitiendo a terceros sustraer la información de los usuarios [11]. Esto ha provocado que un conjunto de la sociedad vea con reticencia la fiabilidad de estos servicios. Además, los usuarios no tienen la certeza de cómo estos sistemas gestionan sus datos, poniendo en duda la privacidad de su información personal [12].

2.2. SEGURIDAD EN LOS SERVICIOS CLOUD

Uno de los principales retos de las tecnologías de la información es **la seguridad de las aplicaciones**. Este reto es especialmente significativo en el caso de las tecnologías *cloud* [13], [14]. La fácil adopción de estos productos en la mayor parte de los casos **no está respaldada por una custodia adecuada de los recursos de los usuarios**. Así, existen múltiples casos de pérdida de credenciales o acceso a información privada por fallos de seguridad en los servidores *cloud*. Esta circunstancia se puede apreciar recientemente en el caso del *hackeo* de los servidores del servicio de almacenamiento de datos de Apple, iCloud [15].

Por esta razón son cada vez más las propuestas que abogan por incluir al usuario como elemento activo en el contexto *cloud*. De esta forma se consigue concienciar al mismo de que es necesario implicarse en la confidencialidad de su información para que no se vean vulneradas sus libertades. En efecto, cuando se hace uso de servidores *cloud* no se tienen garantías sobre cómo se almacenan y tratan los datos, dejando a elección de terceros la forma de gestionar la información privada [16], [17]. Estos problemas de seguridad afectan no sólo a recursos como documentos, imágenes y correos electrónicos, sino que también incluyen sistemas de gestión de calendarios en la nube.

Por ello es necesaria la utilización de mecanismos criptográficos para salvaguardar la privacidad de los recursos propios [18]. Se debe cifrar la información personal en el dispositivo local que se esté usando antes de introducir o guardar los datos dentro de los servicios *cloud* no confiables, utilizando así estas aplicaciones únicamente como herramientas de transporte o de almacenamiento [19]. Esto asegura que los responsables de las aplicaciones o que terceras personas, mediante acceso no autorizado a los servidores, no puedan obtener el contenido de los datos. De esta manera se logra que solo el usuario al que le pertenece la información pueda conseguir la misma de forma legible, haciendo que sea solo él, el único dueño y al único que le debe importar sus contenidos privados. Como resultado se consigue la transición desde un paradigma centrado en la confidencialidad, a un modelo que incluye la preservación de la privacidad en base a un esquema de confianza nula en el servidor externo de almacenamiento [20].

No obstante, siempre que se usan **herramientas de seguridad** dentro de las aplicaciones en las cuales tiene que intervenir el usuario, se debe intentar lograr una **implementación de fácil manejo**. Esto no quiere decir que los algoritmos que se empleen para asegurar la confidencialidad de la información sean fáciles de vulnerar, sino que en los sitios del proceso que sea necesaria la inclusión del agente humano, las herramientas o los mecanismos que tengan que utilizar no sean excesivamente complicados [21]. Pues las partes complejas del algoritmo tienen que ser manejadas adecuadamente por el sistema, y no se puede asumir que todo usuario tenga grandes conocimientos criptográficos para poder cifrar sus datos.

2.3. SERVICIOS CLOUD EN LA ACTUALIDAD

Uno de los grandes avances tecnológicos de la última década es la implantación de los sistemas de almacenamiento y procesamiento de información en la nube. Cada vez son más los usuarios que utilizan estas tecnologías, pues evitan la onerosa tarea de instalar y mantener actualizado el software requerido. Además de permitir centralizar sus recursos en estas aplicaciones, y así poder acceder a los mismos desde diversos sitios y dispositivos de trabajo, sin tener que preocuparse de llevar consigo ningún aparato más, de la posibilidad de perder la información o de replicar los datos para mantenerlos actualizados. El gran auge de los sistemas *cloud* en la actualidad se beneficia de la proliferación del uso de los sistemas portátiles (*smartphones* y *tablets*) [22], que facilitan el acceso a estas aplicaciones en la nube. Así se ha conseguido ir cambiando la mentalidad del uso de sistemas de escritorio en favor de servicios web, haciendo que la computación en la nube tenga una mayor importancia en la sociedad.

Además, los entornos *cloud* ofrecen funcionalidades de tratamiento de información que relajan las exigencias de cómputo en los terminales finales del usuario, ya que la mayor parte de la carga de los procesos se desplaza a la infraestructura que se monta para suministrar el servicio [23]. Esto hace que el usuario tenga que adquirir menos hardware adicional o reducir la instalación del número de aplicaciones en su ordenador de trabajo, solo teniendo que utilizar un navegador o un único software específico para acceder a los servicios *cloud*, logrando así ahorrar costes en la construcción, la gestión y el mantenimiento del equipo. En el caso particular de las pequeñas y las medianas empresas (PYMEs), y en el caso general de los negocios, el uso de estos sistemas en la nube también aportan ventajas que les permiten alcanzar sus objetivos profesionales con un menor coste. Esta circunstancia es de gran relevancia tal y como se pone de manifiesto en el plan de Agenda Digital de España [24].

Sin embargo, pese a la gran facilidad que ofrecen estos servicios a los usuarios para manejar sus datos y usar sus sistemas, una de las dificultades encontradas es la necesidad permanente de la utilización de Internet, ya que es un requisito indispensable para poder beneficiarse de estos sistemas, porque todos estos se encuentran en un entorno *cloud*. Si bien las posibilidades de tener Internet han aumentado considerablemente, no toda la gente puede contratar dicho servicio, ya sea por problemas de cobertura o por tener que asumir los costes desproporcionados que tienen en algunos lugares [25].

Igualmente, dentro de las necesidades del uso de Internet, también hay que tener en cuenta la calidad del mismo. Este es un factor clave, ya que los servicios *cloud* se ofrecen en distintos dispositivos, ordenadores de sobremesa, portátiles, *smartphones* y *tablets*, entre otros, y tienen que suministrar un mínimo de calidad a los distintos equipos. Es por esto que aunque la calidad que se puede obtener en un ordenador, ya sea de sobremesa o portátil, mediante el uso de cable para conseguir Internet sea buena en la gran mayoría, los dispositivos portátiles, *smartphones*, *tablets* y algunas veces los mismos portátiles, no pueden utilizar o acceder al mismo método, y tienen que hacer uso de redes inalámbricas (redes Wi-Fi o redes de datos móviles), cuya calidad de servicio puede llegar a ser buena, pero muchas veces es bastantes baja e intermitente. Por esta razón es necesaria una velocidad de Internet mínima para poder obtener las características que brindan estos entornos *cloud*.

Otro problema que se añade a la utilización de las tecnologías *cloud*, es la aceptación de posibles obligaciones interpuestas en cualquier momento por parte de las compañías que prestan los servicios, para manejar los recursos propios del usuario. Esto hace que se dependa de la voluntad del proveedor del sistema para poder acceder a los datos que se han depositado en las aplicaciones públicas [26]. Además, al tener la información administrada en servidores externos se crea la necesidad de confiar en que la infraestructura del servicio en la nube esté correctamente organizada, ya que, si surge algún contratiempo típico, el sistema puede interrumpir su funcionamiento, provocando que los clientes no consigan gestionar la aplicación.

Por otro lado, los propietarios de las aplicaciones deben comprometerse a optimizar sus sistemas para que un gran conjunto de usuarios pueda utilizar sus servicios de forma adecuada. Al mismo tiempo tienen que tener en cuenta el tipo de cliente al que se orienta la herramienta, pues aunque la complejidad de los temas que se pueden presentar en estos servicios es muy variada, la interfaz de gestión del recurso *cloud* tiene que ser sencilla, intuitiva y funcional, consiguiendo que los usuarios queden satisfechos con su usabilidad. Con todo ello se persigue que el desarrollo de las tecnologías en la nube siga con su vertiginosa expansión (Ver Figura 2) sin ocasionar dificultades añadidas para el acogimiento de las mismas.

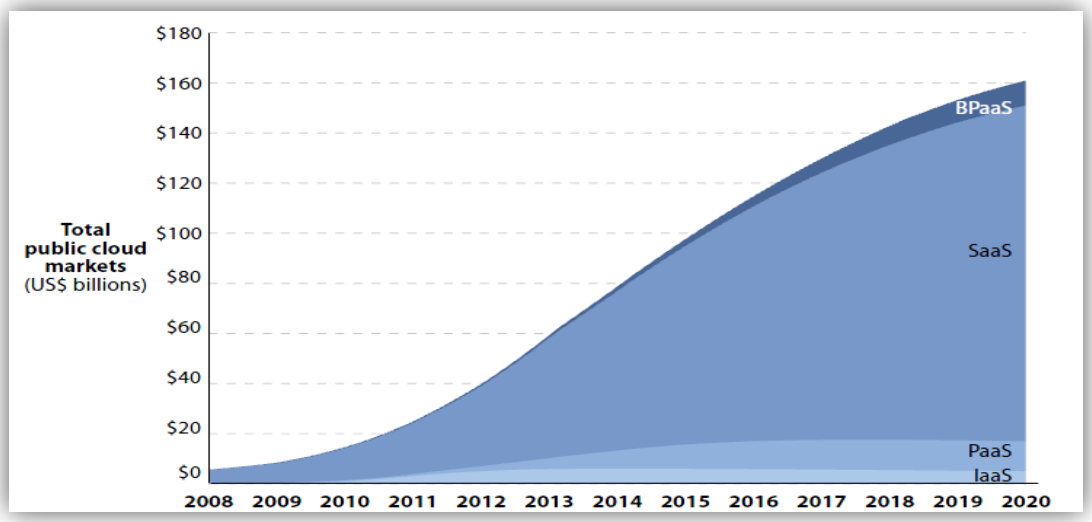


Figura 2: Estimación del tamaño del mercado de los servicios cloud públicos globales.
Fuente: http://www.forbes.com/sites/louiscolumbus/2015/01/24/roundup-of-cloud-computing-forecasts-and-market-estimates-2015/

2.4. NAVEGADORES WEB EN LOS SERVICIOS CLOUD

Una de las formas de **administrar, gestionar y usar los sistemas en la nube** es **mediante navegadores web**. Estos son herramientas software específicas que sirven de vehículo conductor de acceso a los entornos *cloud* mediante la conexión a Internet del equipo del usuario. En el mercado existen varios tipos de navegadores [27], los cuales se diferencian en función del motor de *renderizado* que utilizan para mostrar el contenido de las páginas web o de la compañía propietaria a la que pertenecen. Actualmente, los navegadores más importantes son Google Chrome, Internet Explorer, Mozilla Firefox, Safari y Opera (Ver Figura 3). Como se puede apreciar en el gráfico, la caída del manejo del navegador Internet Explorer está siendo bastante pronunciada en los últimos años, debido a que presenta algunos inconvenientes de diseño y funcionalidad para los desarrolladores [28]. Por esto los creadores de servicios *cloud* se suelen centrar en validar sus aplicaciones en los navegadores Google Chrome y Mozilla Firefox, que son más estables, sin olvidarse de verificar sus sistemas en los otros navegadores también.

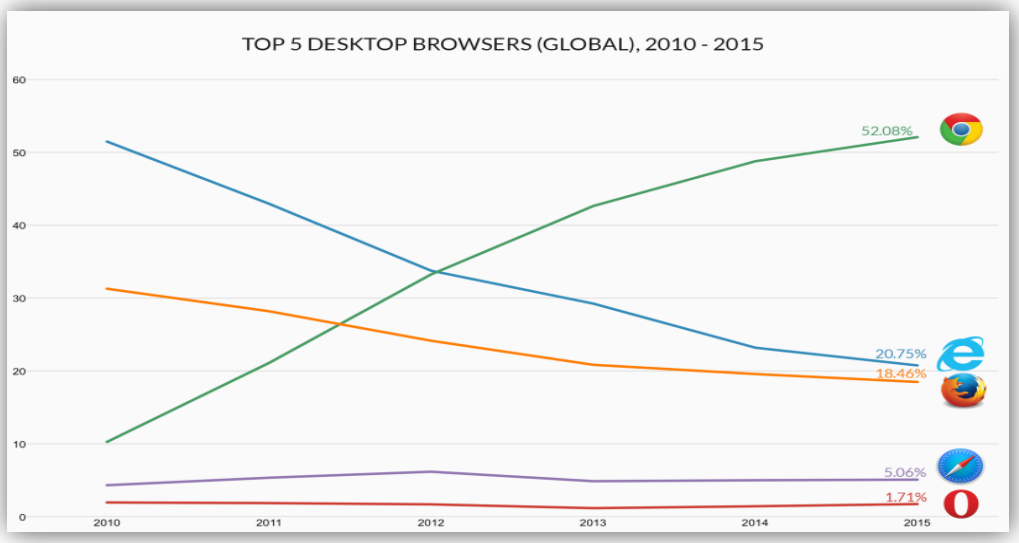


Figura 3: Cuota de mercado de los navegadores de escritorio.
Fuente: http://dadaviz.com/i/3622

Cada navegador tiene sus propias características de funcionamiento, pues son desarrollados por diferentes empresas, pero éstas intentan seguir unos estándares específicos comunes para facilitar la unificación en la implementación de sistemas tecnológicos [29], [30]. Pese al **esfuerzo por satisfacer tales estándares**, existen **diferencias en la implementación** de los diversos navegadores. Tales diferencias provocan que las páginas cargadas se visualicen de una forma distinta en cada uno de ellos. Por este motivo, **los proveedores de servicios en Internet deben tener en cuenta el tipo de navegador que usan los usuarios para entrar en los sistemas** [31], ya que la aplicación *cloud* tiene que **adaptarse a las dificultades que se encuentre en cada uno de los entornos**, para lograr mostrar la interfaz del cliente adecuadamente [32].

Además, no hay que olvidar que se tiene **otro punto de acceso disponible, los dispositivos móviles**. Estos dispositivos se utilizan cada vez más para acceder a servicios de Internet a través de sus propios navegadores (Ver Figura 4). Por ello habría que **diseñar una interfaz optimizada para cada tipo de dispositivo móvil** de cara a solventar los problemas de resolución de pantalla, conexión a Internet, y de selección de navegador. Lo más adecuado es la **creación de aplicaciones nativas** para estos dispositivos, puesto que se solventan la mayoría de estas dificultades. Ahora bien, **no todas las compañías tienen la posibilidad de desarrollar sistemas específicos**, ya que esto conlleva unos gastos adicionales. Esto determina la **preferencia por crear aplicaciones web *universales***, lo cual **no es una tarea trivial** al existir muchos factores importantes que aumentan la complejidad del proceso [33], debido a que se agrupan todas las plataformas de acceso a los servicios *cloud* en una única aplicación de desarrollo, y esta tiene que ser funcional en cada uno de los equipos.

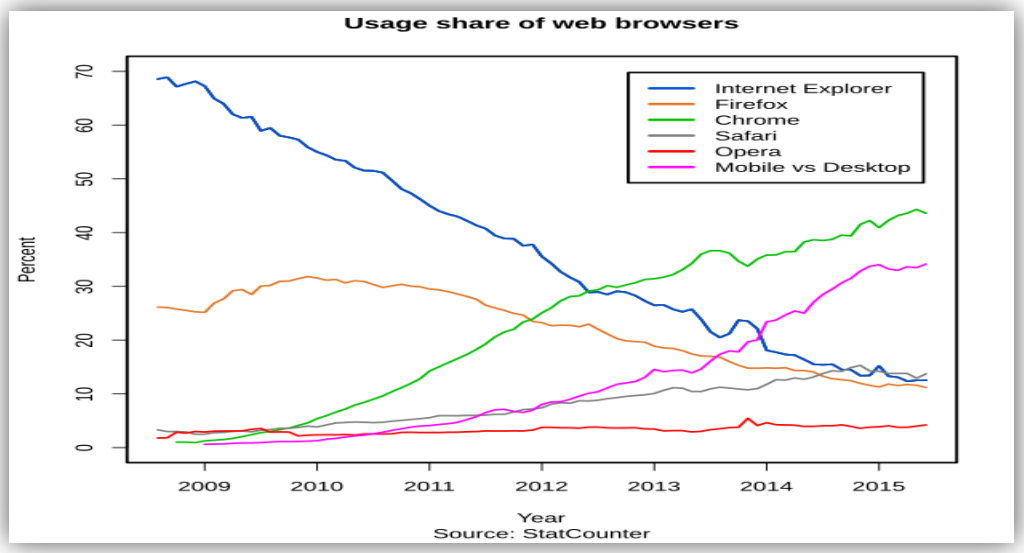


Figura 4: Cuota de uso de los navegadores web en dispositivos de escritorio y móviles.
Fuente: https://es.wikipedia.org/wiki/Navegador_web

3. ANÁLISIS

En este apartado se pone de manifiesto el problema que se quiere abordar y se plasma la definición del proyecto, a través del conjunto de requisitos funcionales y no funcionales que se deben cumplir antes de diseñar el software. Igualmente, se definen los casos de uso del sistema.

Una vez se ha especificado la aplicación, y después de concretar su funcionalidad, se van a estudiar los diversos *frameworks* de JavaScript que hay disponibles, para comprender cuál de estos se adapta mejor al desarrollo del entorno web. También, se examinan algunos de los servidores *cloud* gratuitos para evaluar cuál es más adecuado para su incorporación en la herramienta software. Además, se comparan los métodos de cifrado y los tipos de estructuras de ficheros de calendario que se pueden emplear.

3.1. INTRODUCCIÓN AL PROBLEMA

En este proyecto se pretende desarrollar una aplicación web multiplataforma para la gestión segura del calendario almacenado en la nube. Esta idea surge tras analizar los sistemas de calendario más conocidos que hay disponibles en el mercado (**Calendar de Google, Calendar de Microsoft y Calendar de Apple**), y ver que estos generan una serie de problemas. Los inconvenientes que presentan son los característicos de todos los proveedores de servicios en Internet:

1. No ofrecen la suficiente transparencia a sus usuarios respecto a cómo tratan toda la información que se deposita en ellos (si se usa de forma inadecuada o si es segura).

2. Es preciso un equipo específico para poder acceder a sus servicios.

3. Se necesita aceptar un contrato, que otorga al proveedor de servicios el permiso de usar los datos del usuario, para poder utilizar el sistema.

4. Estas aplicaciones pertenecen a grandes empresas y pueden hacer con sus herramientas lo que ellas quieran, sin que el cliente pueda hacer algo en su contra.

Debido a estas dificultades, y considerando el caso concreto de la gestión privada de calendarios, se siguió buscando otras aplicaciones de este tipo y se encontró **Flock** [34]. Esta es una aplicación Android que gestiona de forma segura los contactos y los calendarios del usuario. El problema que se encontró en este sistema es que está vinculado a un tipo de plataforma concreta, Android.

Como consecuencia de todo este análisis, dado que no se halló herramienta alguna que cubra los objetivos de una aplicación multiplataforma para la protección de calendarios en servicios *cloud* no confiables, se procedió a crear un entorno que proporcione confidencialidad y protección de la privacidad, que sea OpenSource y multiplataforma, para dar todo el poder de gestión del sistema a las personas que la utilicen y a los potenciales desarrolladores que deseen evaluarla y mejorarla.

3.2. DESCRIPCIÓN DEL PROYECTO

El sistema tiene un **mecanismo de acceso basado en la verificación de la cuenta del usuario**, si éste no dispusiera de una cuenta debería crear una antes de poder entrar en la aplicación web. Una vez se cumple esta primera fase de verificación, es necesaria **conseguir la clave privada del usuario** con la que se manejan los datos de los eventos dentro del cliente, aumentando así la seguridad de la aplicación. Si no se ingresa esta clave no es posible continuar con la gestión de los datos del calendario. Si el usuario no tiene aún una clave propia o su clave se ha visto comprometida, **el cliente web puede generar un par de claves pública y privada** para cifrar e intercambiar las claves de sesión que protegen los datos, pero al utilizar este mecanismo los datos guardados anteriormente en el servidor ya no podrán ser accesibles para el sistema.

Después de superar correctamente la verificación del usuario, se muestra el estado principal del sistema, en el cual se puede ver el calendario donde se dispone de la información de los eventos. En esta pantalla **se puede visualizar y gestionar los eventos**, realizar búsquedas de eventos y visualizar las notificaciones del sistema para informar al usuario sobre los eventos compartidos. En el apartado de la gestión de los eventos del calendario se pueden crear, actualizar y eliminar eventos, así como cargar eventos mediante fichero o URL. También se tiene varias formas de visualizar los eventos en el calendario, pudiendo verse en una vista mensual, semanal o diaria.

Igualmente, hay algunos **mecanismos de navegación dentro del calendario, para poder estructurar de forma ordenada todos los eventos**. Además, se comunica al usuario de cambios en el calendario a través de la barra de tareas del entorno local. Por último, si ya no se quiere utilizar la aplicación se da la posibilidad de borrar toda la información de los eventos del calendario personal del servidor *cloud*.

Por otro lado, **la información depositada en el sistema es confidencial**, y por ello se usan algoritmos criptográficos para satisfacer esta premisa. Además, a la aplicación se puede acceder desde distintos dispositivos, ya sean de escritorio o portátiles, y se adapta a las características que presentan cada una de estas estaciones de trabajo. El software implementado es **OpenSource**, para que cualquiera pueda ver o modificar la herramienta software, y así asegurarse de su funcionamiento o adaptarlo a sus necesidades propias.

3.3. CATÁLOGO DE REQUISITOS

En esta sección se detallan los requisitos que tiene la aplicación, los cuales son funcionalidades expuestas que debe cumplir el producto intrínsecamente. Estos requerimientos se dividen en requisitos funcionales y no funcionales. Los requisitos funcionales son las características del sistema que tienen un comportamiento específico dentro de la aplicación. Mientras que los requisitos no funcionales son criterios añadidos que se deben satisfacer para terminar de dar completitud al entorno, aportando más calidad al desarrollo del producto software.

3.3.1. Requisitos funcionales

A continuación, se describen los requisitos funcionales de la aplicación, los cuales se nombran mediante el uso del término RF. Se clasifican siguiendo una organización temática, según el comportamiento que realicen en el software.

- **Acceso**
  - **RF1: Acceder a la aplicación.**
    Acceder al sistema mediante el uso de la cuenta personal del usuario.

  - **RF2: Salir de la aplicación.**
    Salir de la aplicación limpiando todos los datos privados del usuario.

- **Seguridad**
  - **RF3: Mostrar credenciales de acceso.**
    Mostrar los datos del usuario que ha entrado en la aplicación.

  - **RF4: Crear clave privada del usuario.**
    Generar la clave secreta de los nuevos usuarios que usan el software.

  - **RF5: Seleccionar archivo con la clave privada del usuario.**
    Permitir seleccionar el fichero donde el usuario guarda la clave personal.

- **RF6: Introducir clave privada del usuario.**
  Registrar la clave personal del usuario para gestionar los eventos del calendario.

- **RF7: Validar clave privada del usuario.**
  Comprobar si la clave introducida es correcta para poder acceder a la aplicación.

- **RF8: Mostrar clave privada del usuario.**
  Permitir ver la clave personal del usuario ingresada.

- **RF9: Ocultar clave privada del usuario.**
  Permitir esconder el contenido de la clave privada.

- **Gestión**
  - **RF10: Visualizar eventos del calendario.**
    Mostrar los eventos que se encuentran en el calendario personal del usuario.

  - **RF11: Crear eventos del calendario.**
    Introducir nuevos eventos en el calendario del usuario. Los datos de los eventos son: título, fecha de inicio, fecha de fin, hora de inicio, hora de fin, tipo (diario u horario), estado (disponible u ocupado), color, lugar, URL, descripción y asistentes.

  - **RF12: Actualizar eventos del calendario.**
    Modificar los datos del evento seleccionado del calendario del usuario.

  - **RF13: Eliminar eventos del calendario.**
    Borrar el evento seleccionado del calendario del usuario.

  - **RF14: Arrastrar eventos del calendario.**
    Permitir modificar los datos del evento del calendario mediante el desplazamiento del mismo.

  - **RF15: Cancelar operación sobre el evento del calendario.**
    Permitir anular la operación que se realiza sobre el evento del calendario seleccionado antes de finalizar y guardar la ejecución de la misma.

  - **RF16: Cargar eventos de calendario externos mediante fichero.**
    Guardar eventos de calendario en la aplicación usando un fichero que contiene eventos externos.

- **RF17: Cargar eventos de calendario externos mediante URL.**
  Guardar eventos de calendario en la aplicación usando una dirección URL donde se obtienen los eventos externos.

- **RF18: Cancelar operación de carga de eventos del calendario.**
  Permitir anular la operación de carga de eventos externos antes de finalizar y guardar la ejecución de la misma.

- **RF19: Eliminar calendario.**
  Borrar todos los eventos del calendario personal del usuario.

- **Visualización**
  - **RF20: Gestionar visualización del calendario.**
    Mostar los eventos del calendario de distintas maneras. Las formas de visualizar el calendario son: mensual, semanal y diaria.

  - **RF21: Mostrar información de los eventos del calendario.**
    Presentar al usuario el contenido del evento del calendario seleccionado.

- **Navegación**
  - **RF22: Desplazar calendario de forma anual.**
    Mover la pantalla del calendario un año hacia delante o hacia atrás respecto a la colocación actual.

  - **RF23: Desplazar calendario de forma unitaria.**
    Mover la pantalla del calendario una unidad hacia delante o hacia atrás respecto a la colocación actual. La unidad de desplazamiento varía respecto a la visualización actual del calendario.

  - **RF24: Desplazar calendario al día actual.**
    Mover la pantalla del calendario a la posición del día actual en ese momento.

- **Búsqueda**
  - **RF25: Buscar eventos en el calendario.**
    Encontrar eventos que contengan en su título la búsqueda realizada.

  - **RF26: Recargar búsqueda de eventos.**
    Actualizar la búsqueda de eventos realizada cuando se modifique el contenido del calendario.

  - **RF27: Finalizar búsqueda de eventos.**
    Borrar la búsqueda realizada por el usuario.

- **Información**
  - **RF28: Informar de los eventos del sistema.**
    Generar diálogos de aviso de los eventos que ocurren en la aplicación.

  - **RF29: Notificar el estado de la compartición de eventos.**
    Mostrar la información del estado final de la compartición de eventos con otros usuarios.

  - **RF30: Notificar el estado de la carga de eventos compartidos.**
    Mostrar al usuario la carga de eventos compartidos en la aplicación. Asegurar que la notificación sea visible.

  - **RF31: Mostrar notificaciones de estado de eventos compartidos.**
    Visualizar las notificaciones del sistema sobre los eventos compartidos.

  - **RF32: Ocultar notificaciones de estado de eventos compartidos.**
    Esconder las notificaciones del sistema sobre los eventos compartidos.

3.3.2. Requisitos no funcionales

Después de concretar los requisitos funcionales de la aplicación, se procede a detallar los requisitos no funcionales, estos se referencian mediante la utilización del término RNF.

- **RNF1: Mantenibilidad.**
  El mantenimiento es un aspecto fundamental en toda herramienta software. Además, en este caso al tratarse de una aplicación OpenSource, el sistema tiene que brindar una serie de características útiles para los desarrolladores, como son la traza de información y de errores, un código bien documentado, y una estructura modular adecuada para poder incluir nuevas funcionalidades de manera sencilla.

- **RNF2: Usabilidad.**
  La aplicación es multiplataforma, y por ello se puede utilizar en diferentes dispositivos, como ordenadores de sobremesa, portátiles, smartphones y tablets. Esto hace que el sistema tenga que ser funcional y adaptable en los distintos equipos.

- **RNF3: Interfaz.**
  Las pantallas de la aplicación tienen ser fáciles de utilizar, sencillas en su aspecto y no entorpecer la utilización del sistema con demasiada información.

- **RNF4: Portabilidad.**
  El entorno puede ser usado en cualquier lugar y dispositivo, funcionando de modo correcto independientemente del sistema operativo que se tenga en el equipo.

- **RNF5: Escalabilidad.**
  La herramienta software tiene que tener una estructura adecuada para poder añadir nuevas características de forma sencilla.

- **RNF6: Robustez.**
  El sistema tiene que hacer un correcto manejo de los errores que se produzcan para no dificultar su uso por parte del usuario. Además, de no dejar información sensible al alcance de terceros.

- **RNF7: Rendimiento.**
  La aplicación tiene que ser los más ligera posible para funcionar de manera rápida y fluida. La administración de los datos del servidor ha de ser eficaz para no ocasionar pérdidas de tiempo e insatisfacción en los usuarios. Aunque esto depende de la calidad de la conexión a Internet del dispositivo.

- **RNF8: Interoperabilidad.**
  La aplicación tiene que poder gestionar datos sobre eventos de calendario de otros sistemas.

- **RNF9: Coste.**
  Los costes de desarrollo necesarios no tienen que ser elevados para mantener la viabilidad del sistema.

- **RNF10: Seguridad.**
  La aplicación tiene que tratar los datos de los usuarios correctamente para que no se cuestione su confidencialidad e integridad.

3.4. DIAGRAMA DE CASOS DE USO

El diagrama de casos de uso (Ver Figura 5) ilustra las funciones que el usuario puede realizar cuando utiliza el cliente web. Los casos de uso representados, se obtienen a partir de los requisitos funcionales especificados para el sistema. También, se explica el comportamiento de la aplicación en algunos de los casos de uso.

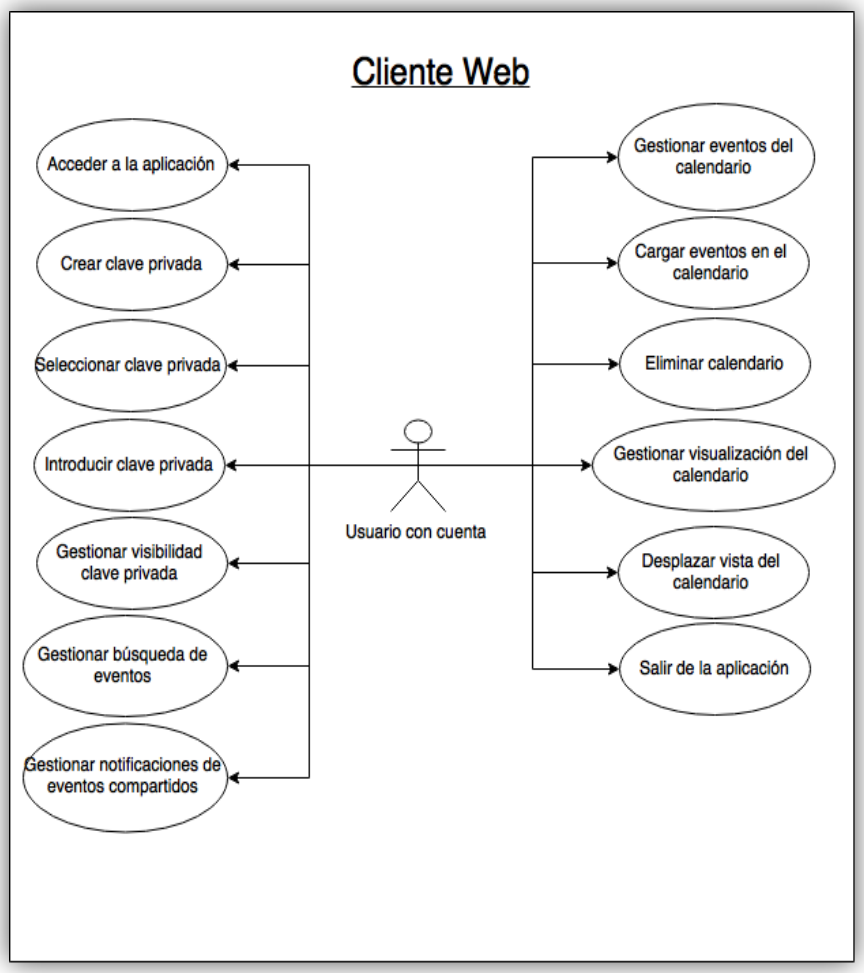


Figura 5: Diagrama de casos de uso

A continuación, se detallan algunos de los casos de uso más importantes dentro del cliente web.

- **Acceder a la aplicación**
  Actor: Usuario con cuenta.
  Descripción: El usuario ingresa su cuenta personal para entrar en el sistema.
  Precondiciones: El usuario tiene una cuenta para acceder al cliente web.
  Postcondiciones: El usuario ve la pantalla para introducir su clave privada.

- **Crear clave privada**
  Actor: Usuario con cuenta.
  Descripción: El usuario selecciona la opción para que la aplicación le cree una clave privada.
  Precondiciones: El usuario ha iniciado sesión en el cliente web.
  Postcondiciones: El usuario ve en la interfaz del sistema la clave privada generada.

- **Introducir clave privada**
  Actor: Usuario con cuenta.
  Descripción: El usuario ingresa su clave privada para poder gestionar sus eventos de calendario.
  Precondiciones: El usuario ha iniciado sesión en el cliente web.
  Postcondiciones: El usuario ve la pantalla principal de la aplicación. En esta se muestra el calendario personal.

- **Gestionar búsqueda de eventos**
  Actor: Usuario con cuenta.
  Descripción: El usuario introduce una búsqueda a realizar entre los eventos de su calendario. Puede borrar la búsqueda después de haberla realizado.
  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.
  Postcondiciones: El usuario visualiza los eventos encontrados con la búsqueda realizada.

- **Gestionar eventos del calendario**
  Actor: Usuario con cuenta.
  Descripción: El usuario puede ver, crear, actualizar y eliminar un evento del calendario.
  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.
  Postcondiciones: El usuario ve el resultado de la operación realizada sobre el evento.

- **Cargar eventos en el calendario**
  Actor: Usuario con cuenta.
  Descripción: El usuario introduce una fuente con eventos para cargar en la aplicación.
  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.
  Postcondiciones: El usuario visualiza el calendario con los eventos cargados.

- **Eliminar calendario**
  Actor: Usuario con cuenta.
  Descripción: El usuario seleccionar borrar toda la información del calendario.
  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.
  Postcondiciones: El usuario visualiza el calendario completamente vacío.

- **Salir de la aplicación**
  Actor: Usuario con cuenta.
  Descripción: El usuario decide cerrar su sesión actual del sistema.
  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.
  Postcondiciones: El usuario sale de la aplicación y visualiza la pantalla de acceso al cliente web.

3.5. TECNOLOGÍAS A EMPLEAR

Para crear la aplicación con la finalidad que proporcione confidencialidad, y sea OpenSource y multiplataforma, se estudiaron los problemas de seguridad y la forma de tratar los datos, para poder dotar a la aplicación cliente de un mecanismo de verificación de la integridad y preservación de la privacidad. Tal y como aparece reflejado en [35], el control de datos administrados por terceras partes no confiables es uno de los grandes retos del actual contexto tecnológico. Dicho reto puede ser abordado, por ejemplo, mediante la consideración de **tecnologías de protección de datos en el cliente y el almacenamiento de la información en un servidor *cloud***.

Además, se tuvo como objetivo la usabilidad del producto final (otro de los grandes retos tecnológicos señalados en [35]), por ello se analizaron las herramientas que permiten crear entornos flexibles para adaptar la aplicación a las características específicas de los dispositivos desde donde se accede al servicio. También, se priorizó la utilización de *software libre* para desarrollar un sistema OpenSource, y así beneficiar a la comunidad de una herramienta de fácil acceso.

3.5.1. Cliente web

Para el propósito que se pretende con la aplicación, se utilizó el concepto de un cliente web [36]. Este tipo de sistemas actúan como un servicio remoto, al cual se puede acceder desde cualquier lugar mediante el uso de un navegador e Internet, o solo con el navegador dentro de entornos locales. Estos servicios han alcanzado bastante popularidad, debido a que permiten implementar herramientas software de forma ágil y sencilla. Para el desarrollo de estos clientes es común emplear los lenguajes de programación HTML, CSS y JavaScript. Estos lenguajes tienen unas características propias que facilitan el diseño de las aplicaciones web. Además, cada lenguaje tiene su entorno de actuación en los clientes:

- HTML en la estructuración de la interfaz de las páginas.
- CSS en la creación del diseño de las pantallas.
- JavaScript en el control del funcionamiento del sistema.

Estos sistemas ofrecen algunas ventajas, las cuales son:

- Acceso deslocalizado a la aplicación mediante el uso del navegador, puesto que el servicio reside en los servidores y no en los dispositivos.

- Reducción de las tareas de gestión (instalación y actualización) de la aplicación para los usuarios, permitiéndoles así ahorrar tiempo y utilizar las últimas funcionalidades de la herramienta siempre que se utiliza el servicio, ya que el mantenimiento es una labor del proveedor del sistema.

- Creación de clientes ligeros, ya que se puede desplazar la carga computacional de la aplicación a los servidores.

No obstante, también existen desventajas en la adopción de estos entornos, las cuales son:

- Gran conjunto de dispositivos de acceso, a los cuales hay que adaptar la aplicación para que sea funcional. Esta tarea de adaptación se facilita con los clientes web, mediante el uso adecuado de la hoja de estilos CSS, pero no es una cuestión trivial.

- Conexión a Internet obligatoria para su correcta ejecución, ya que se deben descargar los ficheros de la aplicación necesarios para que el navegador los muestre en la pantalla del equipo del usuario.

Por otro lado, con la creación de un cliente web se consiguieron alcanzar varias de las características buscadas por el software, una de ellas era ser una aplicación multiplataforma. Esta se logró mediante la utilización de un navegador para acceder al contenido del servicio, abstrayendo al entorno del conocimiento del sistema operativo del equipo de acceso. Del mismo modo, los sistemas web se pueden desarrollar utilizando tecnologías libres, lo que ayudó a la concepción de un sistema OpenSource y de bajo coste. Finalmente, con el uso de los entornos web se pueden crear clientes adaptativos, que modifiquen la interfaz en función del dispositivo utilizado, logrando con ello que la experiencia con el servicio sea adecuada y satisfactoria para el usuario.

Tras concretar el concepto de software con el que se va a diseñar la aplicación, se procedió al estudio más detallado de cada uno de los lenguajes de programación que se van a emplear. Los lenguajes analizados fueron HTML5, CSS3 y JavaScript [37], [38]. Además, se hizo especial hincapié en el lenguaje JavaScript [39], [40], ya que era el que más se iba a usar en la implementación del sistema, y por ello era adecuado tener bastantes conocimientos sobre el mismo.

En este proceso de estudio de las tecnologías a utilizar, se encontró que en HTML5 existe una API de notificaciones integrada [41], la cual se utilizó para crear las alertas locales con la información de la aplicación en el dispositivo del usuario.

3.5.2. Frameworks de JavaScript

Una de las consecuencias de la popularización de los servicios *cloud* ha sido el crecimiento acelerado de aplicaciones web cliente en los últimos tiempos. Esto ha hecho que la comunidad de desarrolladores cree varios *frameworks* para JavaScript (Ver Figura 6), facilitando el diseño de este tipo de sistemas a los nuevos programadores de aplicaciones web.

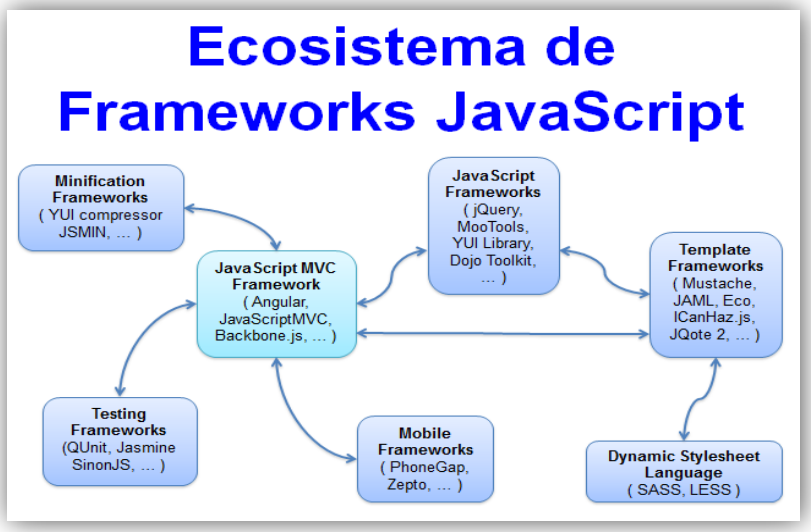


Figura 6: Ecosistema de frameworks JavaScript.

Dentro de esta gran diversidad de *frameworks* que se puede encontrar para el desarrollo web [42], se eligieron para analizar más profundamente los siguientes: Angular, Ember, Backbone, jQuery y Bootstrap. Después de evaluar las ventajas y desventajas de cada uno de ellos, **la elección final fue el *framework* de jQuery para la implementación del código del sistema, y el *framework* de jQuery UI** para usar en la interfaz del cliente. Las diferencias entre los distintos frameworks estudiados eran pequeñas, y la elección fue motivada por la **curva de aprendizaje necesaria**, **la documentación, la relevancia, los ejemplos prácticos en la web, el aporte de la comunidad, el acceso a *plugins* y la compatibilidad con otros *frameworks***.

En la actualidad la diferencia dentro de la comunidad de desarrolladores entre Angular y jQuery es baja (Ver Figura 7), dominando Angular y habría que tenerlo en cuenta en un futuro cercano, ya que está teniendo una fuerte adopción. No obstante, jQuery tiene más bagaje ya que se lanzó antes que Angular, y por ello tiene una gran comunidad de desarrolladores detrás, dando mayor confianza a los desarrolladores noveles, ya que sí surgen problemas se pueden buscar soluciones rápidamente. Además, jQuery tiene un gran potencial gracias a la elevada cantidad de *plugins* que existen asociados a dicho paquete. Estos *plugins* facilitan bastante el desarrollo de este tipo de sistemas software.

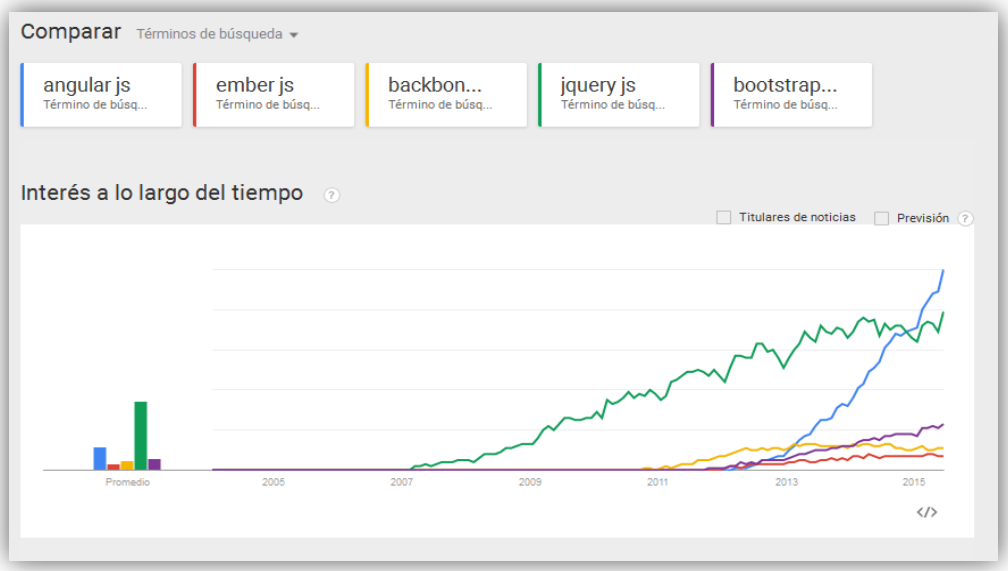


Figura 7: Análisis de uso de frameworks JavaScript de acuerdo con las estadísticas de Google Trends.

3.5.3. Servidores de almacenamiento cloud

Al igual que ocurría en el apartado anterior, en este también se tienen varios elementos que analizar, en concreto **diversos servicios de almacenamiento de datos en la nube** [43]. En la actualidad los servicios de este tipo más importantes o reconocidos son: Google Drive, Dropbox, OneDrive e iCloud. Se estudiaron cada uno de estos servicios para tomar la decisión de cuál de ellos se podría integrar en la aplicación, utilizándolo como servidor donde guardar los datos de los eventos del calendario y asumiendo que es un servidor no confiable a efectos de preservar la privacidad de nuestros datos.

Para **elegir uno** de los servicios de almacenamiento se tuvo en cuenta la **documentación de su API**, la **cantidad de espacio disponible gratuitamente**, el **método de verificación de usuarios**, el **tamaño máximo de archivo que puede gestionar**, la **compatibilidad con otros servicios** y su **uso por parte de los usuarios** (Ver Figura 8). Además, también fue importante la aportación que podrían ofrecer otros servicios relacionados con el servidor *cloud*. Después de evaluar cada uno de estos aspectos, se concluyó que **el servidor a utilizar** fuera **Google Drive**, ya que cumplía con un grueso significativo de las características buscadas y, además, tiene un **servicio complementario** que es **Google Plus**, el cual ofrece herramientas para **validar las credenciales de acceso del usuario**. También, se decidió usar estos servicios por motivos de unificación de datos bajo una sola cuenta de usuario, ya que pertenecen a la misma empresa. El servicio de Google Plus tiene varios métodos para verificar la identidad del usuario, pero se eligió el protocolo OAuth como mecanismo de autorización para acceder al sistema. Esto se decidió así, porque es un protocolo

estándar y abierto [44], que también utilizan otros servicios de almacenamiento, y todo lo que se aprendió con este podría ser útil en una posible integración de otros servicios en el cliente web.

Figura 8: Análisis de uso de los servicios de almacenamiento en la nube de acuerdo con las estadísticas de Google Trends.

3.5.4. Métodos de cifrado

Para asegurar la privacidad de los datos del calendario del usuario se utilizaron algoritmos de cifrado en el lado del cliente. Antes de tomar una decisión concreta, se procedió a investigar los distintos métodos criptográficos existentes, para evaluar cuál era más eficiente y adecuado para este aspecto.

De modo general se distinguen dos tipos de cifrado [45]:

- **Cifrado simétrico.**
  Es un tipo de cifrado sencillo y rápido. Utiliza una misma clave tanto para cifrar como para descifrar los datos. Por ello, es necesario el intercambio de la misma para poder obtener la información de los datos cifrados. El mecanismo para intercambiar las claves es un inconveniente en este método. Además el nivel de seguridad de este sistema se cuantifica en función del tamaño de su clave. Mientras más grande es, más tiempo se tiene que emplear para romper el cifrado.

- **Cifrado asimétrico.**
  Es un tipo de cifrado avanzado y robusto. Utiliza un par de claves (clave pública y clave privada) para el cifrado y el descifrado de los datos. Además, permite firmar la información con la clave privada. En este sistema existe un par de claves para realizar el cifrado, por lo que ya no es necesario el intercambio de claves. La clave que se distribuye es la clave pública, lo que no genera ningún problema. El gran inconveniente de este cifrado es lo costosa que resulta su ejecución, pues precisa de claves de mayor tamaño que el cifrado simétrico para garantizar la completitud del algoritmo y generar datos cifrados más grandes que el contenido original.

Después de estudiar los métodos generales de cifrado se optó por el uso de un cifrado híbrido, que utiliza lo mejor de cada algoritmo en partes distintas de la composición del mensaje cifrado. Concretamente, el cifrado híbrido utiliza el cifrado asimétrico para intercambiar la clave que se genera para el cifrado simétrico, y el cifrado simétrico se emplea para restringir el acceso a la información. A continuación, se va a explicar más detalladamente el funcionamiento de este proceso de cifrado.

- **Criptografía híbrida**

Este sistema (Ver Figura 9) **genera una clave de sesión de forma aleatoria**, y esta es **cifrada con la clave pública del destinatario del mensaje**. Esta misma **clave de sesión**, se utiliza como **clave simétrica para cifrar el mensaje a enviar**. Después, se construye un único paquete con las dos partes antes generadas. Cuando el paquete es recibido, se separa la cabecera del cuerpo del paquete. De la cabecera se obtiene, después de usar la clave privada del receptor, la clave simétrica para descifrar el cuerpo del paquete. Además, el contenido del mensaje se utiliza comprimido para aumentar la complejidad de rotura del sistema.

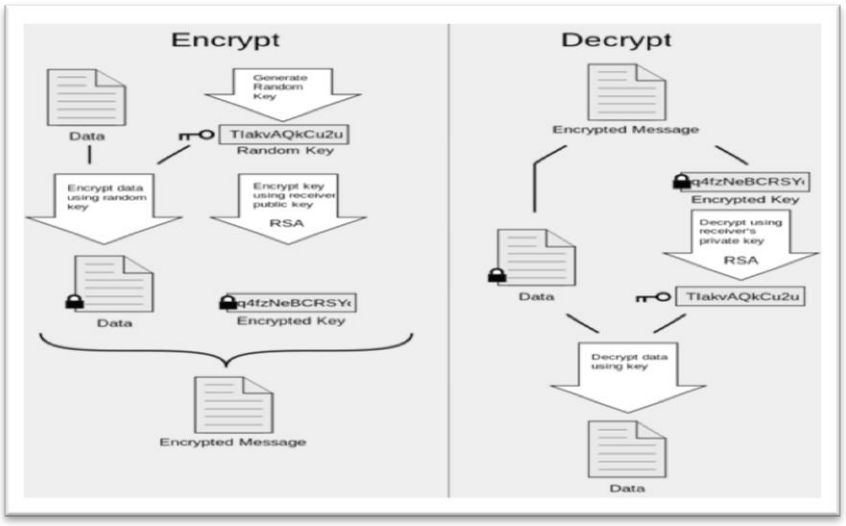


Figura 9: Ilustración del algoritmo de criptografía híbrida.
Fuente: https://en.wikipedia.org/wiki/Pretty_Good_Privacy

Este tipo de cifrado es robusto y eficaz, porque utiliza el cifrado asimétrico para cifrar la clave de sesión, que es más pequeña comparada con el mensaje, esto hace que la carga computacional necesaria no sea elevada, y usa el cifrado simétrico para cifrar la parte más extensa que es el mensaje. Además, soluciona el problema del intercambio de la clave del cifrado simétrico. Esta se envía junto con el mensaje cifrado, pero va cifrada con la clave pública del cifrado asimétrico, lo que aporta seguridad a la transferencia de la clave. Incluso si se logra romper el sistema obteniendo la clave simétrica, solo se puede descifrar un mensaje, ya que se genera una clave de sesión nueva para cada envío de un mensaje nuevo. Es decir, el esquema de cifrado híbrido proporciona una protección adicional para posibles ataques por fuerza bruta.

En el desarrollo de la aplicación se utilizó el **cifrado híbrido** para asegurar la **confidencialidad de la información del usuario**. Para formar los paquetes de cifrado se estudió la estructura estándar **OpenPGP** [46], que cumple con las características del sistema de cifrado híbrido.

Además, este tipo de cifrado está muy extendido dentro de la comunidad de usuarios que quieren proteger su información. Por ello existen **servidores de claves gratuitos, donde almacenar las claves públicas**. Particularmente, en el sistema se utilizó un servidor de claves que pertenece al MIT[1], este se usó para guardar las claves públicas de los usuarios. Estas claves públicas fueron necesarias para poder compartir los eventos del calendario con otros usuarios manteniendo la seguridad de los mismos.

[1] https://pgp.mit.edu/

3.5.5. Tipo de fichero para el calendario

Para almacenar los eventos del calendario en el servidor público, se estudió las posibles estructuras que se podían seguir para formar los ficheros. Por ello se contemplaron los siguientes modos de organización:

- **Estructuras propias:** Estructuras creadas específicamente para organizar los eventos del sistema mediante el uso de XML, HTML o JSON.

- **Estructuras estándares:** Estructuras que ya existían para manejar eventos de calendario. Aquí se analizó los ficheros iCalendar y vCalendar.

Después de hacer el análisis, se decidió usar el tipo de **fichero estándar iCalendar** [47]. Esta elección se realizó, porque se trata de una estructura estándar y libre que se utiliza bastante en la web por otros sistemas de gestión de calendarios. Esto permitió dotar al cliente web de retrocompatibilidad con otros servicios. Además, es una estructura que permite añadir nuevos componentes de manera sencilla.

3.6. HERRAMIENTAS DE TRABAJO Y SOPORTE

Para la elaboración del proyecto se utilizó un conjunto de herramientas de trabajo, que facilitaron su correcto desarrollo. Cabe destacar, que estos materiales son **OpenSource** o que se pueden obtener por medios oficiales, gracias a la universidad. Lo que conlleva a no tener que aumentar los costes de producción del sistema. Estas herramientas se detallan más concretamente a continuación:

- **JetBrains PhpStorm:** Este programa es un entorno de desarrollo integrado (IDE) que permite la creación y codificación de clientes web. Con esta herramienta se manejaron los ficheros del sistema, los cuales eran los archivos HTML, CSS y JavaScript.

- **XAMPP:** Este programa permite crear servidores web en entornos locales para ejecutar las aplicaciones en el equipo de trabajo. Con esta herramienta se creó un servidor Apache local para utilizar la aplicación y así comprobar su implementación.

- **Mendeley:** Este es un administrador de referencias gratuito, que se utilizó para crear la bibliografía de la memoria.

- **Grunt:** Este es un programa que sirve para automatizar la ejecución de tareas JavaScript. Esta herramienta se utilizó para validar el código de la aplicación desarrollada.

- **JSDoc:** Esta es una aplicación que permite generar documentación técnica de código JavaScript. En el sistema se utilizó para documentar los ficheros JavaScript que se desarrollaron.

- **QUnit:** Es un framework que permite crear test unitarios para funciones JavaScript. Esta herramienta se utilizó para validar las funciones JavaScript de los ficheros del sistema.

- **Consola del navegador web:** Esta herramienta permite gestionar la ejecución de las páginas web y renderizar en tiempo real cambios en las mismas. Esta se utilizó para facilitar el desarrollo de la implementación del sistema con los archivos HTML y CSS, y para depurar el código de los ficheros JavaScript.

- **Microsoft Office 2010:** Este programa permite generar documentos de textos, entre otras funciones. Esta herramienta se utilizó para crear la memoria del proyecto.

- **Adobe Reader:** Este programa permite visualizar archivos pdf. Esta herramienta se utilizó para leer la documentación necesaria para el desarrollo del proyecto.

- **Buscador de Google:** Este servicio permite buscar información acerca de cualquier tema. Esta aplicación se utilizó para obtener información durante todo el proceso del trabajo, pero sirvió especialmente en la fase de implementación, ayudando a buscar soluciones dentro de la comunidad de desarrolladores cuando surgían problemas.

4. DISEÑO

En este apartado se especifica el diseño del cliente web, el flujo de navegación del sistema, y la estructura de la aplicación y de los datos, para cumplir con los requisitos y tecnologías detallados en la fase de análisis del proyecto.

4.1. DEFINICIÓN DEL DISEÑO

Para poder realizar un desarrollo adecuado de la herramienta software, se tiene que pensar el diseño con la **finalidad** de poder facilitar la **escalabilidad del sistema y reutilización del código** generado. Para crear un sistema escalable se tiene que seguir una **estructura modular**, separando correctamente las funciones de la aplicación. Siguiendo estas prácticas en el desarrollo del software, se cumple con la necesidad de tener la **máxima cohesión y el mínimo acoplamiento** posible. Además, todo esto también **ayudará al mantenimiento** del cliente web.

La aplicación es un sistema de gestión de eventos de calendario en la nube. Por ello, la interfaz del sistema debe tener una serie de características concretas para que el usuario pueda disfrutar de todas las funcionalidades que ofrece la herramienta software, sin que tenga dificultades para encontrar las mismas o vea deteriorada su experiencia personal.

Las especificaciones que debe cumplir la interfaz del cliente web son las siguientes:

- **Estructurada:** La interfaz debe presentar una organización adecuada en los distintos entornos de funcionamiento.

- **Consistente:** Los elementos de la interfaz tienen que seguir una misma línea visual y de diseño para que la estructuración de la misma resulte coherente.

- **Tiempo de respuesta bajo:** La navegación entre las funciones que realiza el sistema tiene que ser fluida.

- **Informativa:** Se deben mostrar mensajes de información de las operaciones realizadas mediante el uso de textos claros y concisos. Pero, la cantidad de estos mensajes tiene que ser adecuada para que no resulten molestos para el usuario.

- **Sencillez:** La forma de manejar la aplicación no tiene que ser compleja, para que el sistema se pueda adaptar a las necesidades de un mayor conjunto de usuarios, sin importar los conocimientos que éstos tengan. Esta característica hay que tenerla en cuenta sobretodo en el aspecto de la seguridad, ya que hay que usar mecanismos de cifrado. Por ello, el sistema tiene que presentar estas herramientas con un proceso de fácil utilización. Además, se debería minimizar la intervención del usuario en las mismas.

- **Intuitiva:** Los componentes de la interfaz deben mostrar las funcionalidades del sistema de manera directa y precisa.

Para garantizar algunas de estas características en el **cliente web**, éste se diseñó siguiendo una **estructura SPA** (Single Page Application o aplicación de una sola página) [48]. Este tipo de organización está teniendo un gran acogimiento por parte de los desarrolladores en la actualidad [49]. Esta estructura se utiliza para lograr que la navegación entre los elementos de la interfaz sea dinámica, haciendo que la experiencia de usuario sea satisfactoria.

Las estructuras SPA tienen una serie de características propias. A continuación, se comentan algunas de estas:

- Permiten tener toda la interfaz de la aplicación en un **sólo fichero HTML**. No obstante, esto provoca que su organización tenga que ser adecuada para que la implementación del cliente web no se vuelva un problema.

- Intentan simular que el cliente web funciona como una aplicación de escritorio.

- Los clientes web SPA se cargan solamente una vez durante todo su funcionamiento (Ver Figura 10). Debido a esto se hace necesaria la utilización de una tecnología dinámica, como lo es AJAX. Ésta permite obtener los nuevos componentes de la aplicación sin tener que recargar la página web. Al suprimir esta necesidad, se consiguen eliminar los tiempos de espera. Todo esto hace que la navegación dentro de la aplicación sea fluida y agradable.

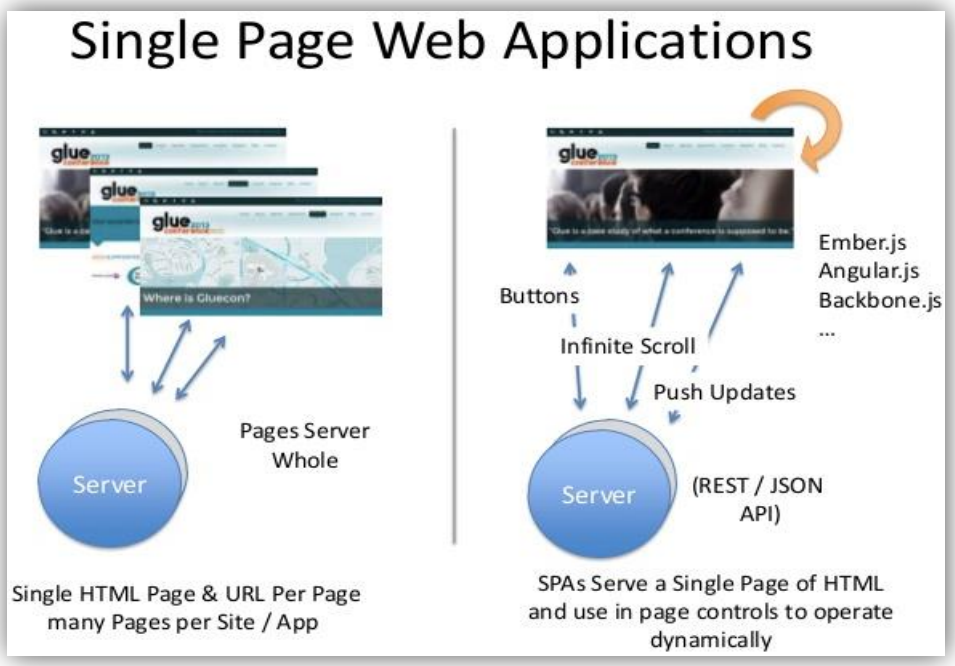


Figura 10: Ilustración de páginas web SPA frente a páginas web tradicionales.
Fuente: http://apievangelist.com/2013/05/23/ember-angular-backbone-single-page-applications-and-apis/

Por otro lado, dentro del diseño también se detalló cómo se iba a tratar la información del calendario del usuario. Se adoptó como criterio para la seguridad, que el **cliente web** fuera el encargado de cifrar los **recursos del calendario** mediante un **cifrado híbrido**, antes de almacenar los ficheros en el servidor *cloud* público y gratuito de Google Drive. Esto se decidió así para hacer que la **información sea segura y confidencial**.

Además, esta elección de diseño cumplía con dos aspectos fundamentales en la seguridad web.

- **Client-side encryption** (seguridad desde el cliente web) [35]: Es una característica importante asegurar los datos antes de transferirlos por Internet. Por ello es necesaria la utilización de mecanismos criptográficos en el lado del cliente. Se realiza de esta forma para que solo se pueda descifrar la información en el cliente web.

- **Zero-knowledge server** (servidor con cero conocimiento) [19]: Es importante no confiarse de los servidores ni de los proveedores de servicios utilizados. Debido a esto, se cifra la información antes de almacenarla, logrando así que ni los servidores ni terceros logren leer los datos guardados.

4.2. FLUJO DE NAVEGACIÓN DEL CLIENTE WEB

A continuación, se muestra la navegación entre los estados de la interfaz del cliente web durante su utilización por parte del usuario (Ver Figura 11).

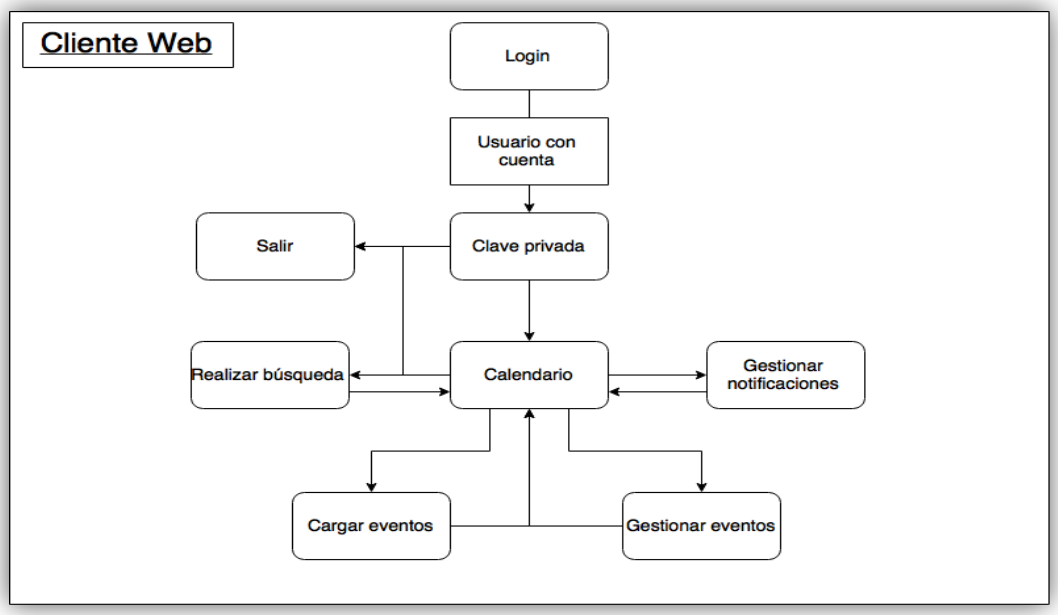


Figura 11: Diagrama del flujo de navegación del cliente web.

4.3. ESTRUCTURA DE LA APLICACIÓN

Para realizar posteriormente la implementación del cliente web se definió una organización según los tipos de ficheros, y una estructura modular para separar los aspectos básicos del sistema. A nivel organizativo el proyecto consta de tres partes:

- **HTML:** Tiene un único fichero HTML, ya que se utilizó un diseño SPA para el cliente web. Este archivo se sitúa en la carpeta principal del proyecto. Este fichero se llama **index.html**, debido a que se utiliza la nomenclatura estándar para el archivo principal de una página web.

- **CSS:** Consta de una carpeta llamada css. Ésta tiene un sólo fichero CSS, donde se especifican las características visuales y de organización de los componentes de la interfaz del cliente web. Este fichero se llama **style.css**.

- **JavaScript:** Consta de una carpeta llamada lib. Ésta tiene cuatro ficheros JavaScript, en los cuales se desarrollarán las funciones del sistema. Estos archivos se crearon en función del módulo de la aplicación que iban a tratar. Por ello, los módulos del cliente web son:

    - Calendario: Tiene un único fichero llamado **calendar.js**. En este archivo se desarrollarán las funciones principales para gestionar el calendario del sistema.

    - Datos: Tiene un único fichero llamado **ical.js**. En este fichero se implementarán las funciones para manejar los ficheros iCalendar. En estos archivos se guardarán los eventos del calendario.

    - Servicios: Tiene un único fichero llamado **gapi.js**. En este fichero se desarrollarán las funciones relacionadas con los servicios de Google. Los servicios a utilizar fueron Google Drive y Google Plus. Estos servicios se usarán para el almacenamiento de los datos y para la verificación del usuario de la aplicación.

    - Seguridad: Tiene un único fichero llamado **crypto.js**. En este fichero se implementarán las funciones criptográficas del cliente web. Estas funciones se encargarán de la parte del cifrado y descifrado del contenido de los ficheros de la herramienta software.

4.4. ESTRUCTURA DE DATOS

Para manejar los eventos del calendario en la parte lógica del cliente web, se concretó la estructura específica que debían seguir los objetos de JavaScript para tal finalidad. La estructura de datos tiene los siguientes campos: identificador, título, indicador de evento diario u horario, fecha de inicio, fecha de fin, URL, lugar, descripción, indicador de evento disponible u ocupado, fecha de creación, fecha de última modificación, secuencia, color, organizador del evento y asistentes al evento.

Por otro lado, al haber decidido utilizar el tipo de fichero iCalendar, para almacenar la información de los eventos del calendario en el servidor *cloud*, éste ya tiene definida una estructura de datos propia para los eventos de calendario. Además, esta estructura de datos es fácil de personalizar, lo que permitió añadir el campo de color del evento mediante el uso del elemento "X-PARAM" (X-COLOR), ya que este aspecto no viene contemplado dentro de los componentes para formar el fichero iCalendar. Todos los otros campos del evento si vienen reflejados dentro de los componentes del archivo.

5. IMPLEMENTACIÓN

En este apartado se explica el proceso que se siguió y las herramientas que se utilizaron para la realización del desarrollo del cliente web. Se describen los entornos de desarrollo, los ficheros de código generados y el proceso de documentación.

5.1. ENTORNO DE DESARROLLO

A continuación, se explica detalladamente el equipo hardware y el entorno software que se utilizó durante la implementación del cliente web. Los dispositivos hardware fueron un ordenador portátil, un *smartphone*, y una pantalla externa que se conectó al portátil. Cabe destacar, que en estos dispositivos serán donde se realizarán las pruebas del sistema. Los comentarios y las características técnicas de cada una de estas herramientas son:

- **Ordenador portátil:** La implementación se ejecutó totalmente en este dispositivo, gracias a la facilitad de movilidad que ofrecía el equipo. Este ordenador fue el entorno principal donde se siguió el desarrollo del cliente web. Debido a esto, la aplicación está optimizada para un equipo de las mismas características. Las especificaciones del mismo son: ordenador portátil Asus A52JC con procesador Intel i5 540M, sistema operativo Windows 7 Home Basic, memoria RAM de 4GB, pantalla de 15.6 pulgadas con relación de aspecto 16:9 y una resolución de 1366x768 píxeles, tarjeta gráfica NVIDIA GeForce 310M con 1GB de memoria dedicada, almacenamiento de 500GB y una tarjeta de red integrada 802.11 b/g/n.

- **Dispositivo móvil:** En la parte final del desarrollo de la aplicación se adaptó la interfaz a las especificaciones de este dispositivo. La parte lógica no tuvo que ser adaptada ya que se utiliza el navegador de los dispositivos para funcionar, abstrayendo la implementación, del sistema operativo que se ejecuta en el equipo. La adaptación se hizo para cumplir los requisitos de portabilidad y usabilidad de la aplicación. Las especificaciones del mismo son: Smartphone Galaxy S4 con pantalla de 5 pulgadas Full HD (resolución 1920x1080 píxeles), memoria interna de 16 GB, conexión móvil 4G y conexión Wi-Fi 802.11 a/b/g/n/ac, sistema operativo Android 5.0 Lollipop, procesador Exynos 5 Octa 5410, tarjeta gráfica PowerVR SGX544MP3 y soporte para navegadores HTML5.

- **Pantalla externa:** También se adaptó el cliente web a otra pantalla que se conectaba por cable VGA al portátil de trabajo. Esto se hizo de nuevo por satisfacer el requisito de usabilidad del sistema. Las especificaciones de esta pantalla son: monitor LG L194WS con una pantalla de 19 pulgadas y una resolución de pantalla de 1440x900 píxeles.

Además, de los equipos hardware utilizados también se tuvo que tener en consideración los navegadores web que se manejaban en cada uno de ellos. Porque los distintos navegadores gratuitos disponibles no se comportan igual ni entre ellos ni entre los dispositivos donde se ejecutan. Por ello, el navegador web principal donde se realizó la implementación fue Mozilla Firefox en la versión 38. Pero después se adaptó el diseño de la interfaz del cliente web al navegador Google Chrome en la versión 43. Igualmente, este proceso de adaptación también se tuvo que aplicar al navegador del dispositivo móvil. El navegador que se utilizó en este entorno fue Google Chrome en la última versión disponible. Como se pone de manifiesto en este párrafo, la creación de un cliente web universal no fue una tarea trivial, ya que cada equipo y navegador tiene sus propias características.

5.2. CODIFICACIÓN

El cliente web se desarrolló siguiendo las directrices de la fase de análisis y diseño. Por ello los conceptos y ficheros generados en este proceso son:

- **Estructural y de composición**

En este concepto se enmarca el fichero index.html (Ver Figura 12). En el archivo se crearon los componentes de la interfaz y se organizaron de forma que favorecieran la usabilidad. La estructura que se realizó es la siguiente: Hay un componente principal que es la aplicación. En ella se distinguen tres grandes elementes, la cabecera, la barra lateral y el contenido principal. En la cabecera se pueden observar los componentes para gestionar la visibilidad de la barra lateral, un separador de contenidos, las opciones de búsqueda y las opciones del usuario. En la barra lateral se pueden ver los elementos para la navegación en el calendario y las opciones para gestionar el calendario. En la sección del contenido principal se pueden ver los componentes de información de progreso de obtención de eventos propios, de notificación de eventos compartidos y el calendario principal del cliente web. Además, en la aplicación se tiene el elemento de información del sistema, y los marcos de trabajo de login, clave de usuario, búsqueda, notificaciones, creación/actualización de eventos, información de eventos y carga de eventos externos.

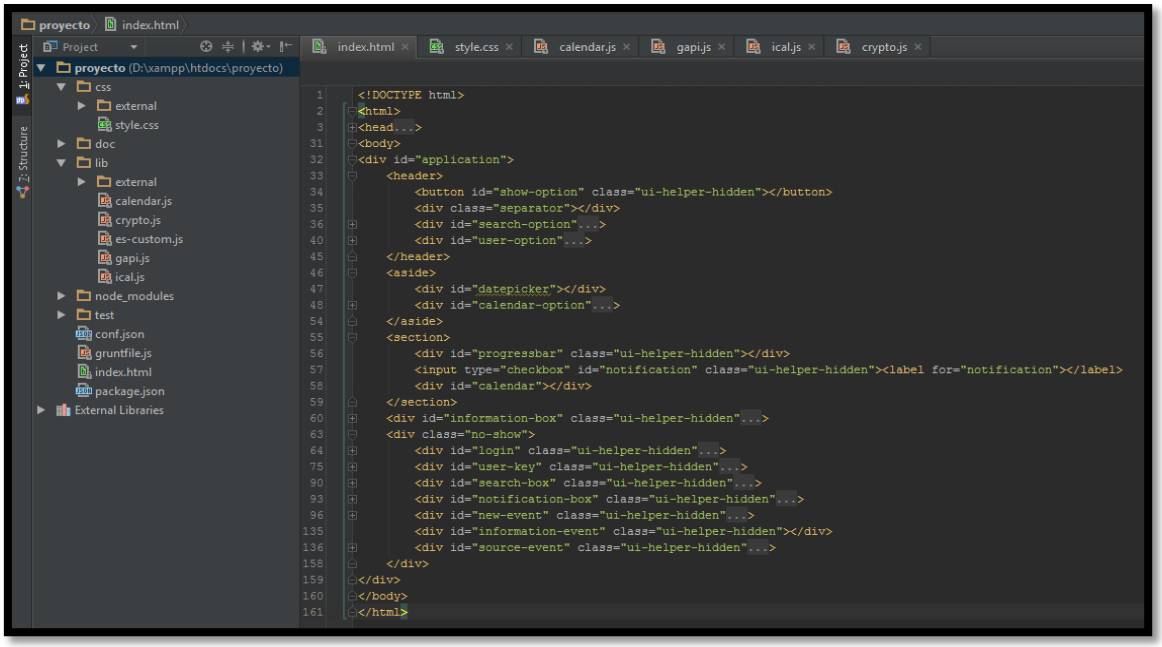


Figura 12: Fichero index.html

- **De disposición y de diseño**

En este concepto se enmarca el fichero style.css (Ver Figura 13). En el archivo se definieron las características visuales y de posicionamiento de los componentes de la interfaz del cliente web. Además, se declararon las propiedades específicas para que la pantalla de la aplicación se adapte a los distintos dispositivos y navegadores de funcionamiento.

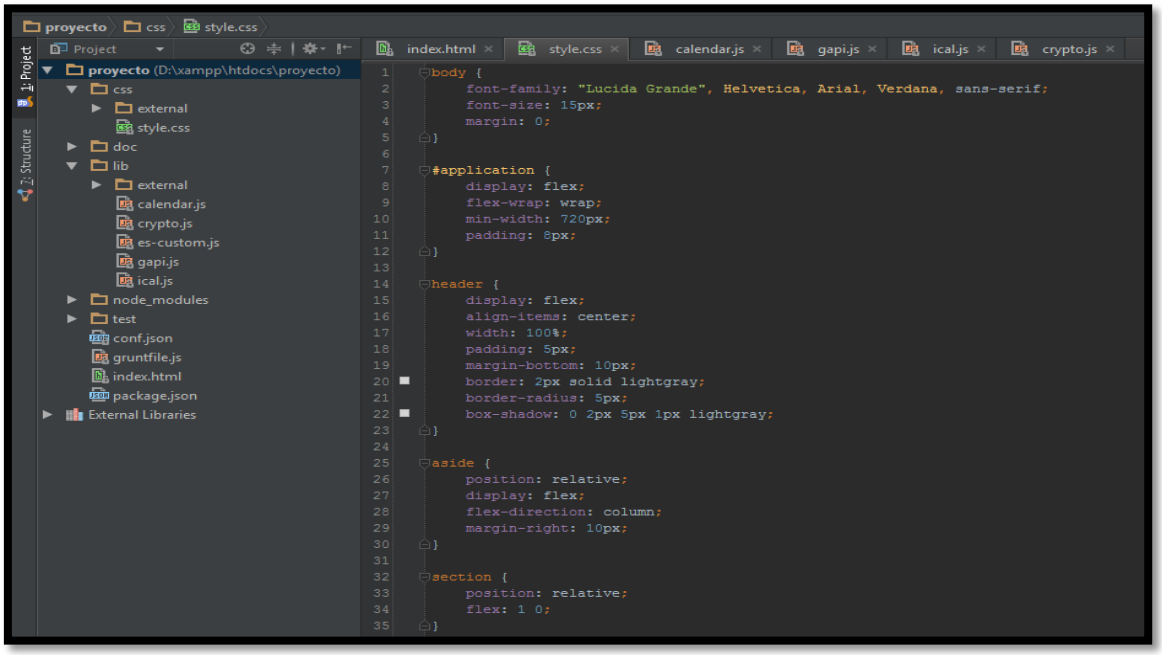


Figura 13: Fragmento del fichero style.css

- **De funcionalidad**

En este concepto se enmarcan cuatro ficheros. Éstos se dividen según el módulo que manejen dentro del cliente web. Los módulos se detallan a continuación:

- Calendario

  En este módulo se encuentra el fichero calendar.js (Ver Figura 14). En el archivo se codificaron las funciones para gestionar todo lo relativo al calendario. Entre estas funciones se pueden encontrar mecanismos de: gestión de usuarios y eventos del calendario, validación, normalización y obtención de datos de formularios de eventos, visualización y navegación del calendario, información de operaciones y de errores, búsqueda de eventos, adaptación funcional, notificación de eventos, compartición de eventos, carga de datos del usuario, carga de eventos externos, confirmación de operaciones críticas y limpieza de los datos del usuario en este módulo.

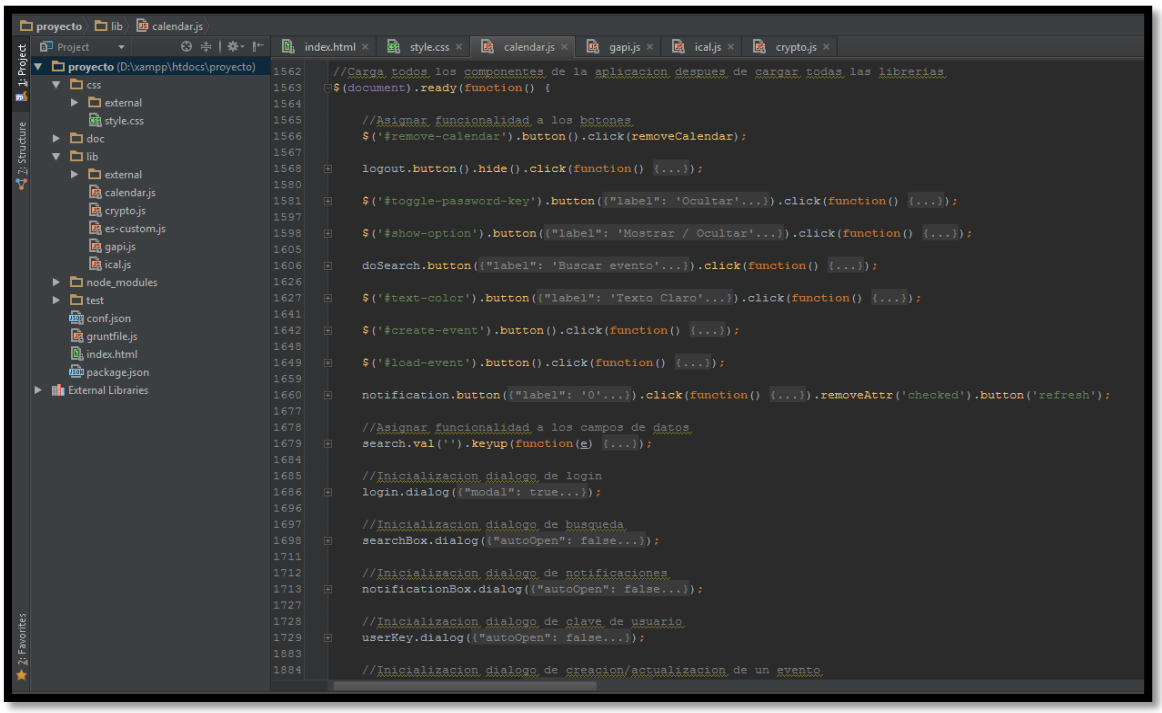


Figura 14: Fragmento del fichero calendar.js

- Datos

  En este módulo se encuentra el fichero ical.js (Ver Figura 15). En el archivo se codificaron las funciones para: manejar los caracteres no soportados por la estructura iCalendar, normalizar la longitud de los componentes iCalendar, generar el contenido de los ficheros iCalendar, convertir el contenido de una estructura iCalendar en un objeto y obtener ficheros iCalendar mediante una dirección URL.

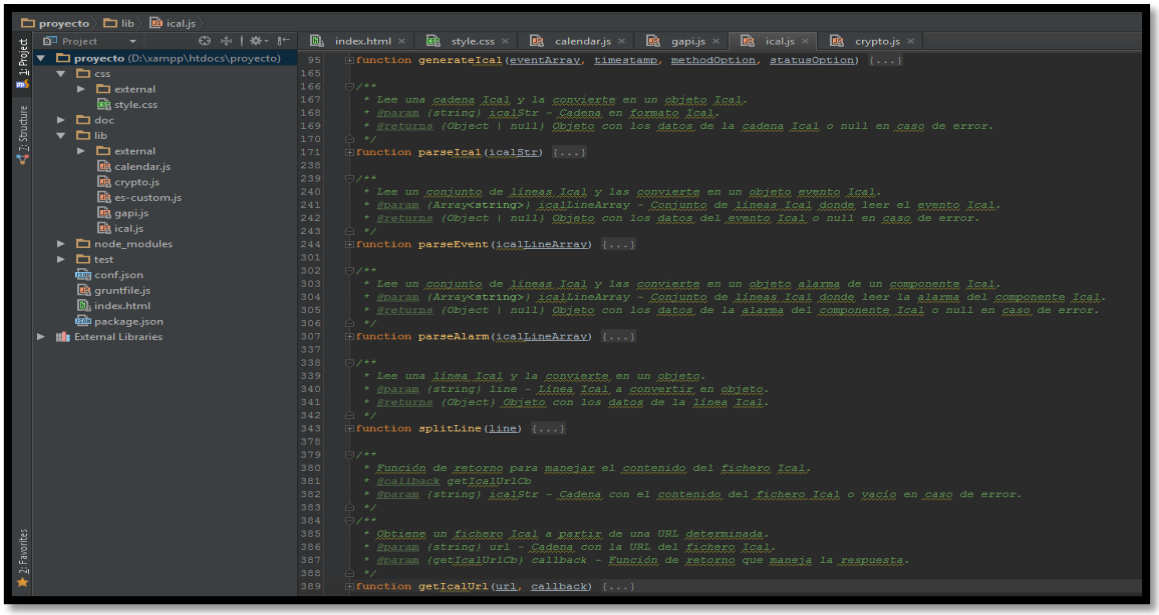


Figura 15: Fragmento del fichero ical.js

- Servicios

  En este módulo se encuentra el fichero gapi.js (Ver Figura 16). En el archivo se codificaron las funciones para: iniciar y cerrar sesión, manejar los errores de los servicios de Google Drive y Google Plus, crear, obtener, actualizar, guardar, eliminar y compartir ficheros en el servidor de almacenamiento, crear la estructura de los ficheros soportados por el servidor y limpiar los datos del usuario en este módulo.

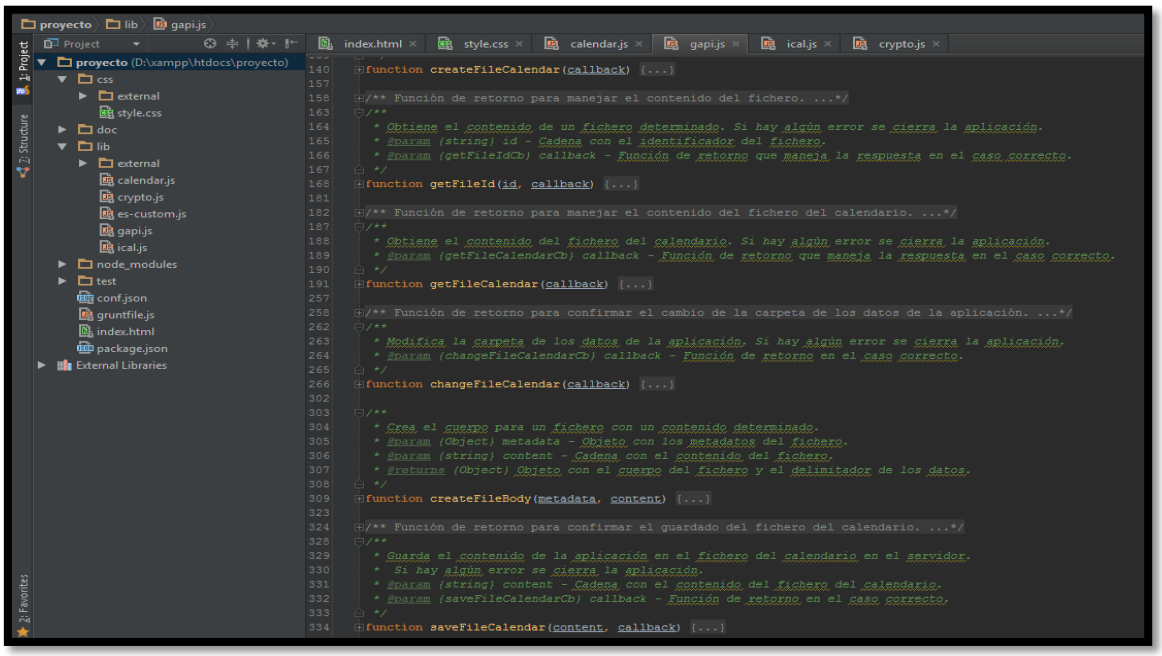


Figura 16: Fragmento del fichero gapi.js

- Seguridad

  En este módulo se encuentra el fichero crypto.js (Ver Figura 17). En el archivo se codificaron las funciones para: generar el par de claves (clave pública y clave privada) del usuario, validar la clave privada del usuario, cifrar y descifrar los ficheros del calendario, obtener la clave pública de un usuario para compartir un evento, añadir la clave pública del usuario en el servidor de claves y limpiar los datos del usuario en este módulo.

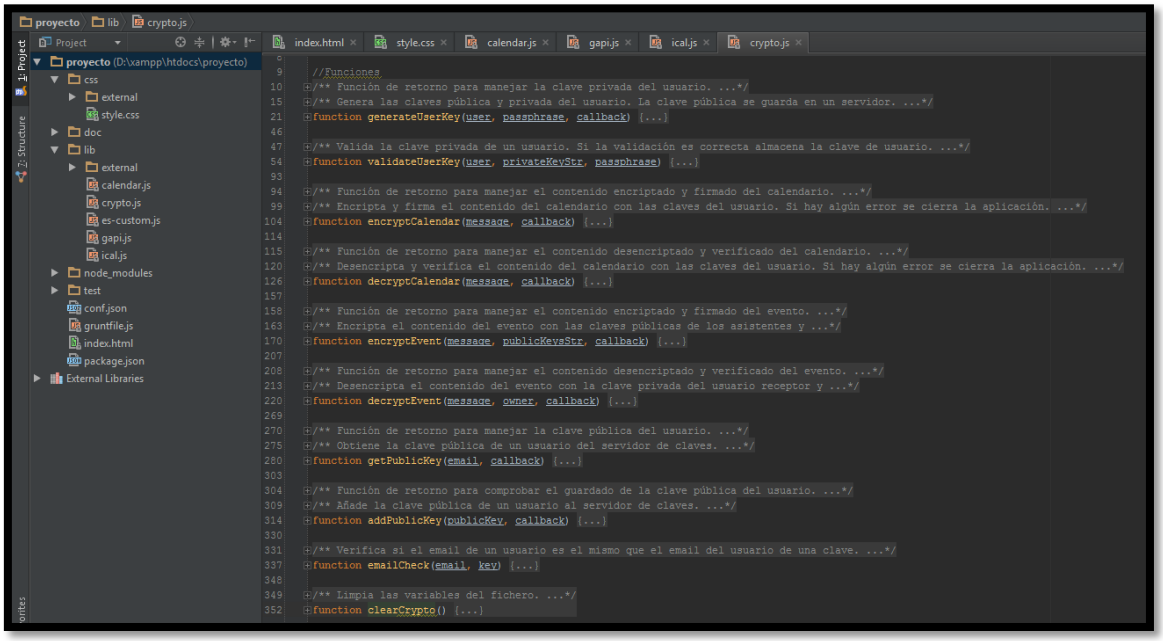


Figura 17: Fichero crypto.js

5.3. DOCUMENTACIÓN

Para acabar con la implementación del cliente web, se procedió a crear la documentación técnica de la aplicación mediante la herramienta **JSDoc**. Se realizó este proceso para que el sistema pueda ser **mantenido fácilmente**. Además, como la aplicación es OpenSource, con esta documentación se consigue explicar el procedimiento específico de cada función codificada. Asimismo, durante toda la fase de implementación se fue comentando en las líneas de código, los aspectos más relevantes de cada método. Esto permite que el **código del sistema** se pueda **seguir con fluidez**, y que cualquier desarrollar externo pueda comprobar el funcionamiento del cliente web sin demasiadas complicaciones. A continuación, se muestra la página de inicio de la documentación del sistema desarrollado (Ver Figura 18):

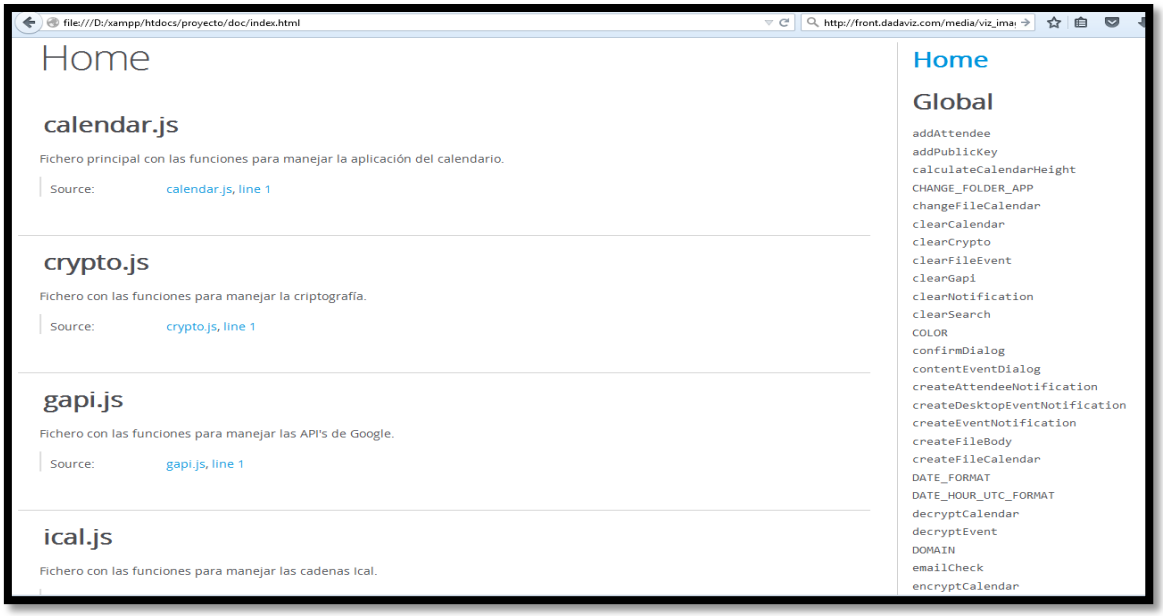


Figura 18: Página de inicio de la documentación.

6. PRUEBAS

En este apartado se muestra y describe el resultado de las pruebas que se realizaron sobre el cliente web.

Se efectuaron tres tipos de pruebas:

- **Pruebas unitarias:** Con estas pruebas se quiere comprobar que las funciones implementadas se ejecutan correctamente.

- **Pruebas de integración:** Con estas pruebas se quiere verificar que los elementos y los módulos desarrollados se compenetran adecuadamente para realizar sus funciones específicas. Además, con estas pruebas se quiere comprobar el correcto funcionamiento de toda la aplicación en su conjunto.

- **Pruebas de sistema:** Con estas pruebas se quiere probar la integración del cliente web con distintos entornos (diferentes navegadores y dispositivos).

Las **pruebas unitarias** de la aplicación se realizaron mediante el uso de la herramienta **QUnit**. Se construyó un módulo de pruebas por cada módulo de la aplicación. A continuación, se puede ver el resultado de estas pruebas.

Las **pruebas de integración** del cliente web se realizaron mediante la experimentación. Se ejecutaron así, ya que era necesario un token OAuth válido para acceder a la aplicación, y no se puede conseguir uno durante las pruebas, porque es obligatoria la verificación de una cuenta de correo del usuario en un servicio externo (Google Plus).

Se eligió a algunos agentes humanos para que testearan la herramienta software, persiguiendo como objetivo encontrar la mayor cantidad posible de fallos. Tras comprobar el correcto funcionamiento del cliente en un entono, se unieron las pruebas de integración con las **pruebas del sistema**.

Estas pruebas se realizaron en dos equipos diferentes, un portátil y un *smartphone* (Samsung Galaxy S4). Además, también se emplearon dos navegadores distintos, en el ordenador portátil se utilizaron Google Chrome y Mozilla Firefox, y en el dispositivo móvil se manejó Google Chrome.

En el navegador **Google Chrome** existe una **herramienta de desarrollo** que permite **simular la pantalla de algunos dispositivos móviles**. Esta utilidad se empleó para continuar con el plan de pruebas de la aplicación. Las pruebas de integración y de sistema que se realizaron en los entornos simulados fueron satisfactorias.

Por otro lado para validar el código de los ficheros JavaScript se utilizó la herramienta **Grunt**, que permite automatizar esta tarea de verificación de los archivos.

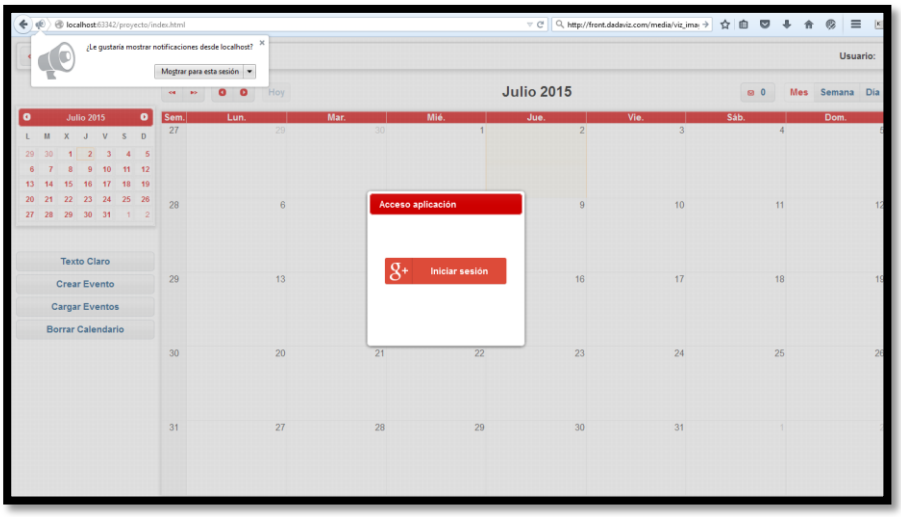
7. RESULTADO

En este apartado se muestra el funcionamiento del cliente web desarrollado. También se comentan las acciones que el usuario de la aplicación puede llevar a cabo en las pantallas de la misma.

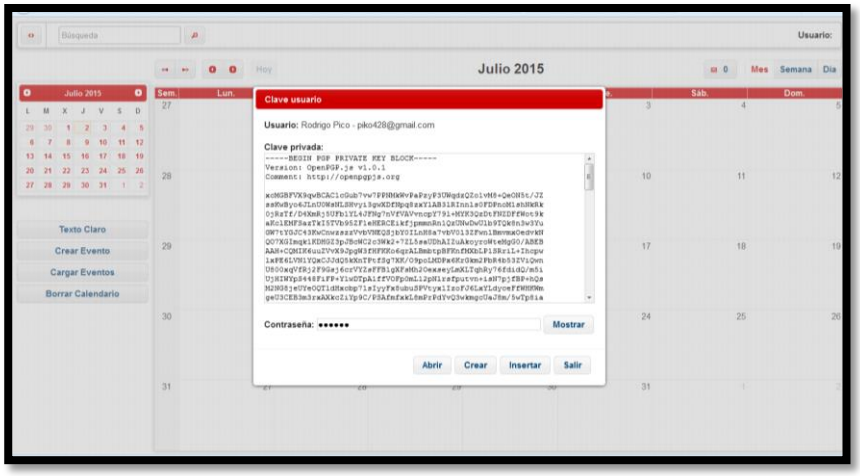
7.1. INTERFAZ DEL CLIENTE WEB

A continuación se pueden ver las distintas pantallas que el usuario se encuentra cuando utiliza el sistema.

- **Pantalla de login**
  En esta pantalla se puede ver el diálogo con el botón de acceso para iniciar sesión y la petición de confirmación de las notificaciones de escritorio. Si el usuario se autentica correctamente pasa a la pantalla de clave de usuario.

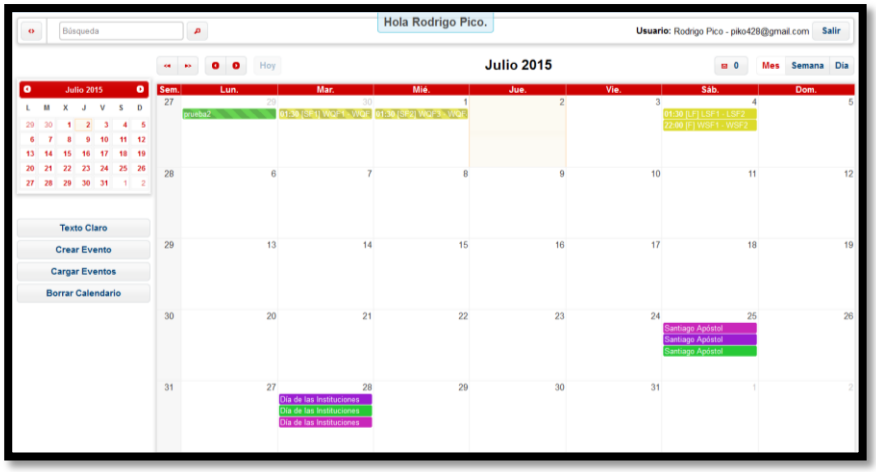


- **Pantalla de clave de usuario**
  En esta pantalla se puede ver un ejemplo de clave privada. También se muestra las credenciales del usuario que ha iniciado sesión. Además, existen cuatro acciones que se pueden realizar: abrir un fichero con la clave privada, crear una clave privada, insertar una clave privada en el sistema y salir de la aplicación. Si la clave privada es válida se muestra la pantalla del calendario.



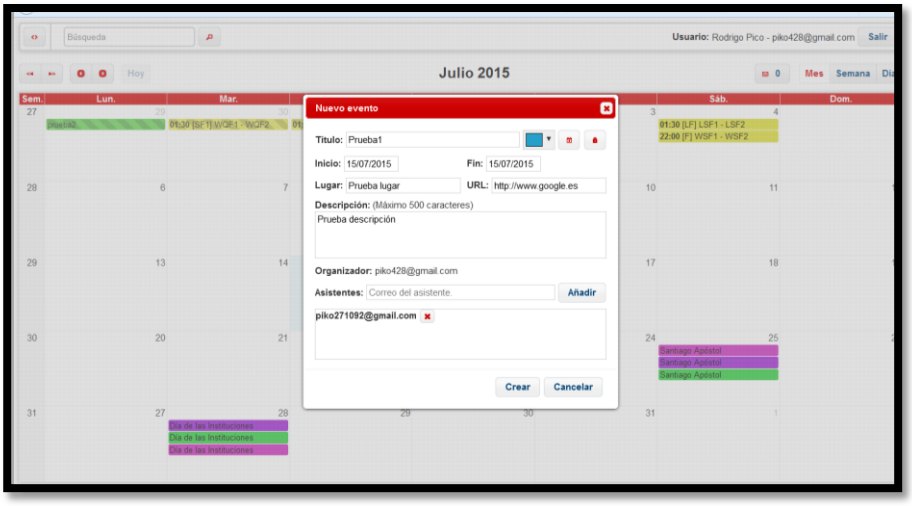
- **Pantalla del calendario**

  En esta pantalla se muestra el mensaje de bienvenida a la aplicación y los eventos del usuario. En esta pantalla se pueden realizar búsquedas, navegar por el calendario, gestionar los eventos del calendario, borrar el calendario y salir de la aplicación. Estas acciones se detallan en otras pantallas.



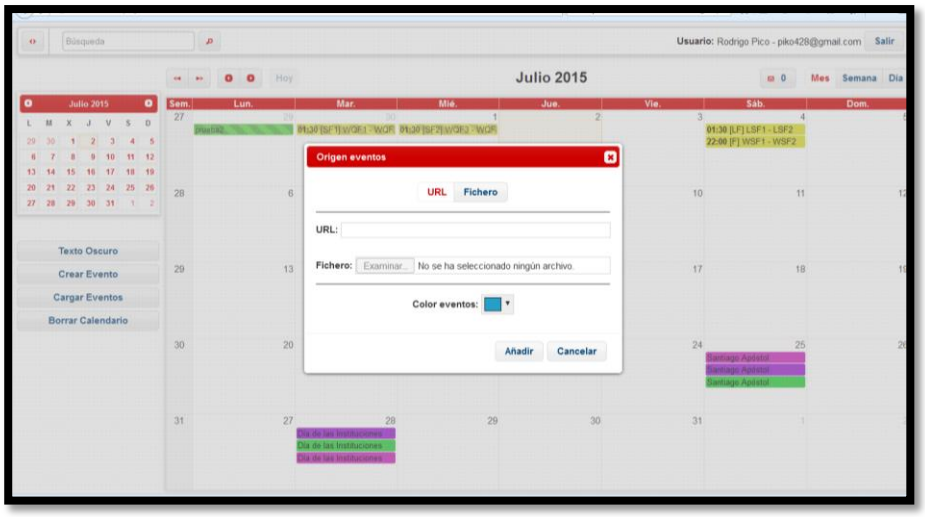
- **Pantalla de creación de evento**

  En esta pantalla se muestra el formulario que hay que rellenar para crear un evento. El evento consta de título, color, indicador de diario u horario, indicador de disponible u ocupado, fecha de inicio, fecha de fin, lugar, URL, descripción y asistentes. EL organizzador del evento es el usario con sesión. Además se pueden realizar dos acciones, confirmar la creación del evento o cancerlar la operación. Con las dos acciones se vuelve a la pantallla de calendario.



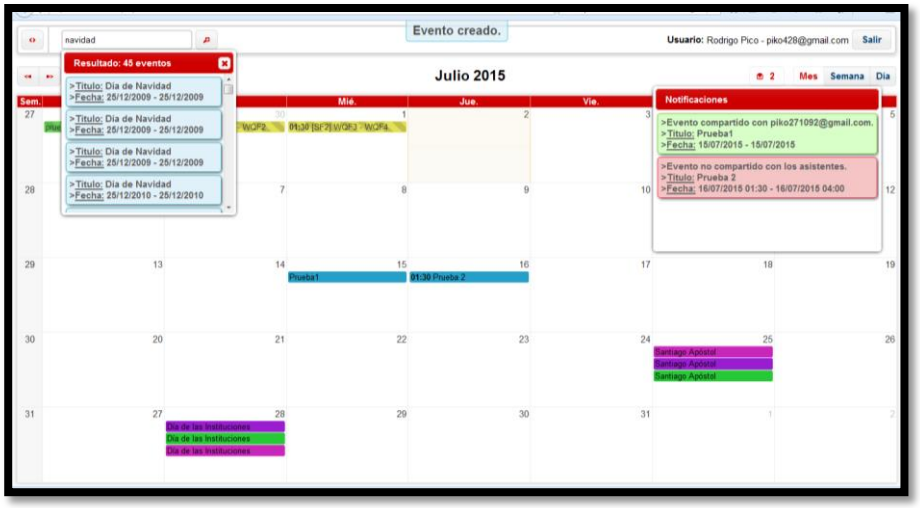
- **Pantalla de carga de eventos**

  En esta pantalla se muestra la opciones disponibles para cargar eventos externos. Hay dos formas mediante URL o por fichero. Además se pueden realizar dos acciones confirmar la carga de eventos o cancelar la operación. Con las dos acciones se vuelve a la pantallla de calendario.

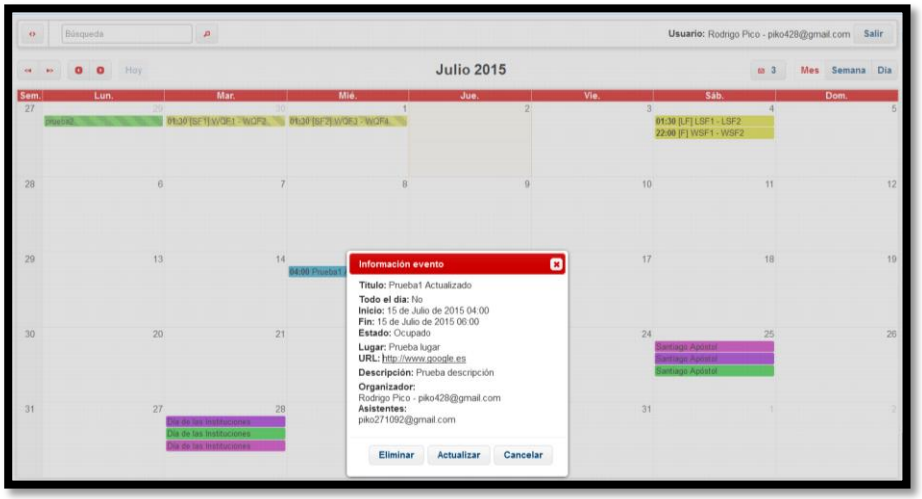


- **Pantalla de búsqueda y notificaciones de eventos compartidos**

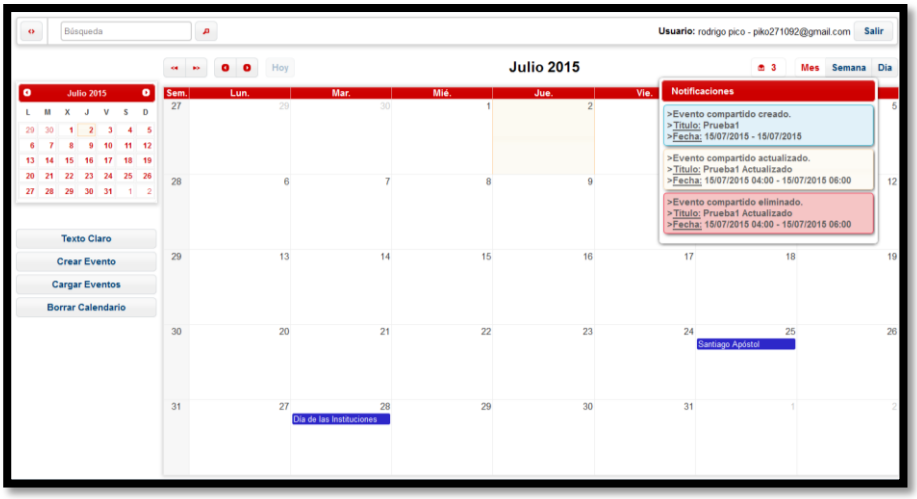
  En esta pantalla se muestra los eventos encontrados cuando se realiza una búsqueda (parte superior izquierda) y los mensajes de estado (correcto o incorrecto) cuando se comaprte un evento (parte superior derecha).



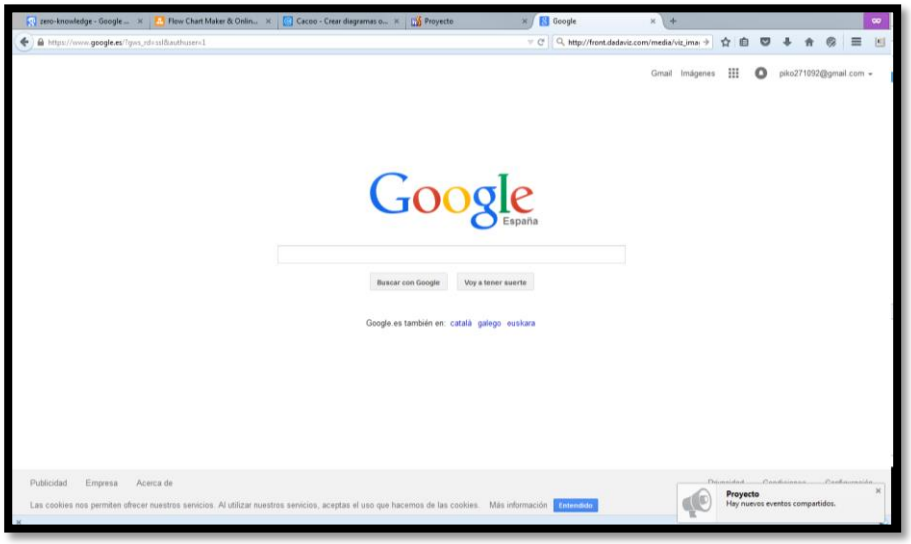
- **Pantalla de información de evento**
  En esta pantalla se muestra la información de un evento seleccionado. Aquí se pueden realizar tres acciones, actualizar un evento, eliminar un evento o cancelar la visualización de la información. Cuando se eligen las operaciones de eliminiar y cancelar se vuelve a la pantalla del calendario. Si se selecciona actualizar se pasa a la pantalla de actualización de evento, esta pantalla es parecida a la de creación.



- **Pantalla de notificaciones de carga de eventos compartidos**
  En esta pantalla se muestra el estado de la carga de tres eventos compartidos (parte superio derecha). El primero muestra la creación de un evento compartido, el segundo la actualización de un evento compartido y el tercero la eliminación de un evento compartido.



- **Pantalla de notificaciones de escritorio**
  Si el usuario no tiene activa la aplicación, se muestra una notificación de escritorio (parte inferior derecha) cuando se han cargado nuevos eventos compartidos.



8. CONCLUSIONES

Con la realización de este Trabajo Fin de Grado, se quiso poner en práctica los conocimientos adquiridos durante los estudios del Grado de Ingeniería Informática. También se pretendió aumentar los conocimientos que se tenían sobre los campos de la seguridad y de la creación de aplicaciones, ya que se tiene especial interés en analizar las últimas herramientas, tecnologías y procedimientos que se utilizan para conseguir tales finalidades. Además, estos aspectos han cobrado gran relevancia en los últimos años, aumentando las posibilidades que después se consiga acabar trabajando en estas temáticas. Igualmente, se perseguía poder gestionar todas las fases que integran la creación de una herramienta software, para saber si se era capaz de dirigir todo el desarrollo del proceso de elaboración correctamente.

Por otro lado, durante el proyecto se realizó el diseño de una aplicación en un entorno *cloud*, teniendo como principales objetivos los fundamentos conceptuales denominados *security-by-design* y *privacy-by-design* [50]. Concretamente, se efectuó el diseño y la creación de una aplicación para la gestión de eventos de calendario en la nube de forma segura, garantizando así la confidencialidad y la privacidad de la información del usuario almacenada en el servidor. Para lograr este objetivo se partió de los conocimientos básicos que se tenían sobre el desarrollo seguro de aplicaciones web.

Además, una parte importante del trabajo vino determinada por las consideraciones de usabilidad de la aplicación resultante. Ésta debía ser funcional tanto en ordenadores como en dispositivos móviles. Por ello se tuvieron que analizar más profundamente los lenguajes de programación del lado del cliente para intentar adaptar la aplicación a la mayoría de entornos. Esto ayudó a ganar experiencia con estos lenguajes, además de llegar a la conclusión que no es una tarea trivial, desarrollar un único sistema para diferentes dispositivos.

Asimismo, la obtención de los requisitos de seguridad y privacidad exigieron la utilización de tecnologías criptográficas que aún no se conocían, como el cifrado híbrido y el protocolo de autorización federada OAuth. Estas herramientas permiten manejar de forma flexible la generación y distribución de claves, así como la gestión de acceso a recursos en sistemas complejos. Sobre el tema de la seguridad, cabe destacar que este tipo de soluciones son muy importantes y necesarias en el ámbito de plataformas *cloud*, pero también lo son en el escenario global del actual estado de desarrollo TIC [35]. Además, el proceso completo de la creación de aplicaciones con la seguridad en mente, demanda un mecanismo exhaustivo de evaluación, para garantizar el cumplimiento de las características de diseño iniciales.

Finalmente, se pudo concluir satisfactoriamente el proyecto con una aplicación que cumplía con las especificaciones de eficiencia y eficacia en la gestión de eventos de calendario propuesto. No obstante, la aplicación puede ser mejorada de acuerdo con los siguientes aspectos de trabajo futuro:

- En el sistema desarrollado la plataforma de almacenamiento que se utilizó fue la del servicio Google Drive únicamente. En futuras versiones se podrían incluir otros servicios de almacenamiento *cloud* gratuitos, que cuenten con una API de desarrollo, como Dropbox, Mega, OneDrive, etc.

- El sistema se puede hacer más flexible combinando plataformas *cloud* públicas y privadas, es decir, se podrían utilizar sistemas híbridos de almacenamiento en la nube.

- La incorporación de nuevas funcionalidades dentro del gestor del calendario, es otra vía de mejora a tratar, después de la creación de este primer prototipo.

- El protocolo OAuth en este proyecto sólo se empleó para obtener un token de acceso que permitía la utilización del cliente web para administrar el calendario. Sin embargo, el protocolo OAuth es un protocolo de gran complejidad que permitiría crear un conjunto más amplio de esquemas de autorización. Esto es de gran interés para los entornos híbridos de almacenamiento en la nube. En relación a este aspecto se puede tomar como punto de partida, para mejorar la herramienta software creada, la información elaborada en este TFG [51].

GLOSARIO

Angular: Tipo de framework JavaScript que se utiliza para crear aplicaciones MVC (Modelo-Vista-Controlador).

Cloud: Se utiliza como sinónimo de Internet.

Cloud computing: Computación en la nube. Se utiliza para ofrecer servicios.

CSS: (Cascading Style Sheets) Hoja de estilos en cascada.

Framework: Un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problema particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

GPG: (GNU Privacy Guard, GnuPG) Es una herramienta de cifrado y firmas digitales, que viene a ser un reemplazo de PGP (Pretty Good Privacy), pero con la principal diferencia que es software libre licenciado bajo la GPL.

Hackear: Acción de explorar y buscar las limitantes de un código o de una máquina.

HTML: (HyperText Markup Language) Lenguaje de marcas de hipertexto.

IaaS: (Infrastructure as a Service) Infraestructura como servicio.

iCalendar: Es un estándar para el intercambio de información de calendarios**.**

JavaScript: Es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

jQuery: Es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Login: Iniciar sesión.

Logout: Cerrar sesión.

OAuth: Es un protocolo que permite flujos simples de autorización para sitios web o aplicaciones informáticas.

OpenSource: Recursos o herramientas que son libres, es decir, no hay que pagar por su utilización.

PaaS: (Platform as a Service) Plataforma como servicio.

PGP: (Pretty Good Privacy, privacidad bastante buena) Es un programa cuya finalidad es proteger la información distribuida a través de Internet mediante el uso de criptografía de clave pública, así como facilitar la autenticación de documentos gracias a firmas digitales.

Plugin: Es un componente software que añade una característica específica a otra aplicación.

Renderizar: Proceso de generar una imagen o vídeo.

SaaS: (Software as a Service) Software como servicio.

Smartphone: Teléfono móvil inteligente de menos de 6 pulgadas de pantalla.

Software: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

Tablet: Dispositivo móvil inteligente de más de 8 pulgadas de pantalla.

URL: (Uniform Resource Locator) Un localizador de recursos uniforme es un identificador de recursos uniforme (URI) cuyos recursos referidos pueden cambiar.

Wi-Fi: Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

BIBLIOGRAFÍA

[1] J. Hoepman and B. Jacobs, “Software Security Through Open Source,” pp. 1–15, 2005.

[2] Wikipedia, “Computación en la nube,” 2015. [Online]. Available: https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube. [Accessed: 27-Jun-2015].

[3] T. Velte, A. Velte, and R. Elsenpeter, *Cloud computing, a practical approach*. McGraw-Hill, Inc., 2009.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and others, “A view of cloud computing,” *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] Wikipedia, “Time-sharing,” 2015. [Online]. Available: https://en.wikipedia.org/wiki/Time-sharing. [Accessed: 27-Jun-2015].

[6] S. Garfinkel, “The Cloud Imperative,” 2015. [Online]. Available: http://www.technologyreview.com/news/425623/the-cloud-imperative/. [Accessed: 27-Jun-2015].

[7] S. W. Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, “Brief History of the Internet,” 2015. [Online]. Available: http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet. [Accessed: 27-Jun-2015].

[8] CloudTweaks, “Cloud Computing and Google Docs,” 2015. [Online]. Available: http://cloudtweaks.com/2010/12/cloud-computing-and-google-docs/. [Accessed: 27-Jun-2015].

[9] L. Qian, Z. Luo, Y. Du, and L. Guo, “Cloud computing: an overview,” in *Cloud Computing*, Springer, 2009, pp. 626–631.

[10] Wikipedia, “Chrome OS,” 2015. [Online]. Available: https://es.wikipedia.org/wiki/Chrome_OS. [Accessed: 27-Jun-2015].

[11] Data Center Knowledge, “Security Breaches, Data Loss, Outages: The Bad Side of Cloud | Data Center Knowledge.” [Online]. Available: http://www.datacenterknowledge.com/archives/2015/03/16/security-breaches-data-loss-outages-the-bad-side-of-cloud/. [Accessed: 01-Jul-2015].

[12] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, “Security and privacy in cloud computing: A survey,” in *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, 2010, pp. 105–112.

[13] L. M. Kaufman, “Data security in the world of cloud computing,” *Secur. Privacy, IEEE*, vol. 7, no. 4, pp. 61–64, 2009.

[14] S. Subashini and V. Kavitha, “A survey on security issues in service delivery models of cloud computing,” *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.

[15] N. Statesman, “The iCloud leak: weak security isn’t only a problem for Apple's backup service,” 2015. [Online]. Available: http://www.newstatesman.com/sci-tech/2014/09/icloud-leak-weak-security-isnt-only-problem-apples-backup-service. [Accessed: 27-Jun-2015].
[16] TechRadar, “Has Google been listening to us via Chrome?,” 2015. [Online]. Available: http://www.techradar.com/news/software/applications/has-google-been-listening-to-us-via-chrome--1297637?src=rss&attr=all. [Accessed: 27-Jun-2015].
[17] G. O. S. Blog, “Google’s Updated Privacy Policy,” 2015. [Online]. Available: http://googlesystem.blogspot.com.es/2015/06/googles-updated-privacy-policy.html. [Accessed: 27-Jun-2015].
[18] INTECO, “Seguridad y privacidad del cloud computing,” 2015. [Online]. Available: https://www.incibe.es/file/3LeSufa2tYmC_bcRKSFfbg. [Accessed: 27-Jun-2015].
[19] Crypton, “Build private applications,” 2015. [Online]. Available: https://crypton.io/. [Accessed: 27-Jun-2015].
[20] “Cloud computing 2014: Moving to a zero-trust security model | Computerworld.” [Online]. Available: http://www.computerworld.com/article/2487123/data-privacy/cloud-computing-2014--moving-to-a-zero-trust-security-model.html. [Accessed: 01-Jul-2015].
[21] A. Whitten and J. D. Tygar, “Why Johnny Can’T Encrypt: A Usability Evaluation of PGP 5.0,” in *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, 1999, p. 14.
[22] Wikipedia, “Usage share of operating systems,” 2015. [Online]. Available: https://en.wikipedia.org/wiki/Usage_share_of_operating_systems. [Accessed: 27-Jun-2015].
[23] NVIDIA, “NVIDIA GRID,” 2015. [Online]. Available: http://www.nvidia.es/object/nvidia-grid-es.html. [Accessed: 27-Jun-2015].
[24] G. Español, “Plan de TIC en PYME y comercio electrónico,” 2015. [Online]. Available: http://www.agendadigital.gob.es/planes-actuaciones/Bibliotecaticpyme/1. Plan/Plan-ADpE-2_TIC-PYME.pdf. [Accessed: 27-Jun-2015].
[25] “World Internet Users Statistics and 2015 World Population Stats.” [Online]. Available: http://www.internetworldstats.com/stats.htm. [Accessed: 01-Jul-2015].
[26] J. P. Kesan, C. M. Hayes, and M. N. Bashir, “Information Privacy and Data Control in Cloud Computing: Consumers, Privacy Preferences, and Market Efficiency,” *Wash. Lee L. Rev.*, vol. 70, p. 341, 2013.
[27] Wikipedia, “Comparativa de navegadores web,” 2015. [Online]. Available: https://es.wikipedia.org/wiki/Anexo:Comparativa_de_navegadores_web.
[28] HTML5TEST, “Comparación de características de los navegadores web de escritorio,” 2015. [Online]. Available: https://html5test.com/results/desktop.html. [Accessed: 27-Jun-2015].

[29] Wikipedia, “Standards-compliant,” 2015. [Online]. Available: https://en.wikipedia.org/wiki/Standards-compliant. [Accessed: 27-Jun-2015].
[30] O. of I. Technology, “Browser Standards,” 2015. [Online]. Available: http://www.oit.uci.edu/browser-standards/. [Accessed: 27-Jun-2015].
[31] Device Atlas, “DeviceAtlas | World’s Leading Device Detection & Intelligence Solution.” [Online]. Available: https://deviceatlas.com/. [Accessed: 01-Jul-2015].
[32] A. Mourouzis, A. Leonidis, M. Foukarakis, M. Antona, and N. Maglaveras, “A novel design approach for multi-device adaptable user interfaces: concepts, methods and examples,” in *Universal Access in Human-Computer Interaction. Design for All and eInclusion*, Springer, 2011, pp. 400–409.
[33] T. Dahm, “Browser Compatibility Tutorial,” 2015. [Online]. Available: http://www.netmechanic.com/products/Browser-Tutorial.shtml. [Accessed: 27-Jun-2015].
[34] O. W. Systems, “Flock,” 2015. [Online]. Available: https://whispersystems.org/blog/flock/. [Accessed: 27-Jun-2015].
[35] D. Arroyo, J. Diaz, and V. Gayoso, “On the Difficult Tradeoff Between Security and Privacy: Challenges for the Management of Digital Identities,” in *International Joint Conference*, 2015, pp. 455–462.
[36] A. W. Kosner, “The Appification Of Everything Will Transform The World’s 360 Million Web Sites,” *2015*. [Online]. Available: http://www.forbes.com/sites/anthonykosner/2012/12/16/forecast-2013-the-appification-of-everything-will-turn-the-web-into-an-app-o-verse/. [Accessed: 27-Jun-2015].
[37] J. D. Gauchat, *El gran libro de HTML5, CSS3 y JavaScript*. Marcombo, 2012.
[38] J. N. Robbins, *Learning web design: A beginner’s guide to HTML, CSS, JavaScript, and web graphics*. “ O’Reilly Media, Inc.,” 2012.
[39] R. Harmes, D. Diaz, and S. Willison, *Pro JavaScript Design Patterns*, vol. 1. Apress, 2008.
[40] D. Flanagan, *JavaScript: the definitive guide*. “ O’Reilly Media, Inc.,” 2006.
[41] MDN, “Notification API,” 2015. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/notification. [Accessed: 27-Jun-2015].
[42] Wikipedia, “Comparison of JavaScript frameworks,” 2015. [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks. [Accessed: 27-Jun-2015].
[43] Wikipedia, “Comparison of file hosting services,” 2015. [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_file_hosting_services. [Accessed: 27-Jun-2015].
[44] Oa. C. Site, “OAuth 2.0,” 2015. [Online]. Available: http://oauth.net/. [Accessed: 27-Jun-2015].
[45] D. Stinson, *Cryptography Theory And Practice*, 3rd ed. CRC Press, 2003, p. 593.
[46] O. Alliance, “OpenPGP Message Format,” 2015. .

[47] IETF, “Internet Calendaring and Scheduling Core Object Specification (iCalendar),” 2015. .
[48] A. Mesbah and A. Van Deursen, “Migrating multi-page web applications to single-page Ajax interfaces,” in *Software Maintenance and Reengineering, 2007. CSMR’07. 11th European Conference on*, 2007, pp. 181–190.
[49] C. Sotelo, “Evolution of the Single Page Application,” 2015. .
[50] S. Gurses, C. Troncoso, and C. Diaz, “Engineering Privacy by design,” in *Computers, Privacy & Data Protection*, 2011, vol. 317, no. 5842, pp. 1178–9.
[51] Á. García Delgado and others, “Implementaci{ó}n de un proveedor de autorizaciones OAuth 2.0 con Scala,” 2014.