MISTIC: Master interuniversitario en la seguridad de las TIC

Secure system for personal finances on the Cloud

Adrián Polo Alcaide

Jordi Castellà-Roca

Universitat Rovira i Virgili

Enero 2016

Agradecimientos

Al finalizar un trabajo tan arduo y lleno de dificultades como el desarrollo de un trabajo final de master es inevitable que te asalte un muy humano egocentrismo que te lleva a concentrar la mayor parte del merito en el aporte que se ha realizado. Sin embargo, el análisis objetivo muestra que la magnitud de este aporte no hubiese sido posible sin la participación de personas e instituciones que han facilitado las cosas para que este trabajo haya podido realizarse. Por ello, es para mi un verdadero honor utilizar este espacio para expresarles mis agradecimientos.

Primeramente querría agradecer a mi tutor Jordi Castellà Roca por el tiempo empleado y guiarme en desarrollo del proyecto a la UOC por ofrecerme los conocimientos para dicho desarrollo. A la gente de KeepCoding y en especial a Alberto Casero por la ayuda ofrecida para el desarrollo de la parte de Django y por último a mi familia por apoyarme y aguantarme durante el proyecto.

Resumen

Este documento contiene la memoria de la realización del Trabajo Final de Master que tiene como objetivo la creación de un Cloud para el almacenamiento de forma segura las facturas electrónicas. Así como las facturas se agrupan para facilitar el control de los gastos. Primeramente se introduce en el ámbito y la problemática. A continuación se definen los objetivos del proyecto, la tecnologías empleadas y el diseño. Finalmente se describe brevemente como se realizado el proceso de la creación los resultados obtenidos comparados con los objetivos marcados y se presentan las futuras mejoras y conclusiones.

Palabras Clave:

Servidor, Cloud, Web, Seguridad, Factura Electrónica, Django, Google Cloud Platform, Google App Engine.

Abstract

This document contains the memory of the accomplishment of the final Project, which aims to create a Cloud for secure storage of the electronic invoices. The invoices are grouped for easy control of expenses. First has the introduction with the field and problems. Then explains the objectives of the project, the technologies used and design. Finally we briefly describe the process of creating, the results obtained compared with the objectives set and future improvements and conclusions presented are made.

Index Terms

Server, Cloud, Web, Security, Electronic Invoice, Django, Google Cloud Platform, Google App Engine.

Tabla of Contenidos

Agradecimientos 3

Resumen 4

Palabras Clave: 4

Abstract 4

Index Terms 4

Introducción 10

Objetivos 11

Justificación 12

Organización de la memoria 12

Tecnologías empleadas 13

Servidor (Cloud) 13

Cliente 15

Diseño y Arquitectura teórica 16

Actores 17

Casos de Uso 17

Actor Consumidor: 17

Actor Proveedor: 18

Diagrama de Secuencia 19

Registro 19

Login 20

Datos de Usuario 21

Subida de facturas y recibos. 22

Generar reporte 23

Descargar Factura 24

Diagrama de Clase 25

Planificación 25

Primera Fase. Creación del Cloud 26

Comprensión de la Factura electrónica 27

Creación de la Base de Datos 27
Segunda Fase. Creación de las funcionalidades 27
Tercera Fase. Creación de la aplicación 27
Diagrama de Gantt 28
Control de versiones 28
Desarrollo 29
Primera fase: Creación del Cloud 29
Django [14] 31
La Factura Electrónica 40
El formato de la factura-e [24] 41
Creación de la base de datos 42
El proyecto para la extracción del modelo 45
El fichero de entrada 45
Resultado del software, entidades de base de datos. 46
Segunda Fase. Creación de las funcionalidades 48
La creación del Registro 48
La creación del Login 51
La subida de facturas 52
La Generación de los reportes 53
Creación de la descarga de las facturas 54
La Creación de los Datos de Usuario 55
Tercera Fase. Creación de la aplicación Universal 56
El framework Bootstrap en el proyecto 57
La aplicación de PhoneGap 57
Resultados 58
Garantizar la seguridad y integridad de los datos en la obtención almacenamiento y gestión y mantener la integridad. 58
La creación de Cloud seguro donde se puedan realizar peticiones 58
Implementar una subida de información por parte del usuario. 58
Creación de una aplicación que permita emplearlo. 59

Realizar un estudio de los sistemas criptográficos para dicho ámbito y evaluar cual se adapta mejor a los objetivos anteriores. 59

Futuras Mejoras 60

Conclusión.....61

Bibliografía.....63

Introducción

Según el último informe presentado por Ericsson [1] llamado Mobility Report[1] el 70% de la población mundial tendrá un dispositivo con acceso a Internet en 2020 pasando a 6.100 millones de teléfonos con acceso a Internet, de este informe podemos extrapolar que el teléfono móvil se esta volviendo un dispositivo indispensable para las personas ya que les permiten estar conectados con otros usuarios y realizar tareas cotidianas desde su propio terminal.

Esto ha generado un gran mercado para las empresas de telefonía, aportando más funcionalidad a los dispositivos móviles haciendo unión con diferentes empresas y mercados. La ultima gran funcionalidad que se esta empezando a emplear es el pago por el teléfono apoyada por las grandes entidades bancarias, dicha mejora permite realizar pagos empleando diferentes tecnologías que dispone el dispositivo. Esta nueva funcionalidad implica que el dispositivo contendrá los datos bancarios de nuestras cuentas, por lo que ha generado una desconfianza por que los dispositivos móviles se han convertido en victimas de los ciberdelincuentes tal y como se presenta en un informe presentado por Telefónica [2] donde se detalla que el porcentaje de penetración en los dispositivos móviles ha aumentado de un 63% a un 81% en España. Esta penetración permite recopilar datos personales del dispositivo, si imaginamos que un usuario desea comprar un producto en una tienda donde emiten facturas al realizar el pago, el usuario decide comprar cierto producto y realiza el pago por el dispositivo móvil empleando su código de seguridad validando la compra. Una vez la tienda ha validado el pago por su terminal TPV[2] emite la factura a través del correo electrónico del cliente (o empleando cualquier otro tipo de tecnología) alojando el recibo en su

[1] Es uno de los principales análisis anuales de trafico de datos por red móvil, donde proporciona una visión de las tendencias actuales en el trafico de datos y de mercado.

[2] (Terminal de punto de venta) es un dispositivo que ayuda en la tarea de gestión de un establecimiento comercial de venta al publico, permitiendo realizar transacciones de manera electrónica

terminal el comprobante de la transacción. Dicho usuario después de realizar la transacción (al cabo de unos días) se conecta a una red Wi-Fi vulnerable y realizan un ataque de penetración en el dispositivo. Esto permite al atacante robar la información del usuario, en este caso los datos bancarios estarían protegidos mediante su numero secreto pero el atacante podrá realizar un grafo de las posiciones del usuario mediante las facturas almacenadas en el dispositivo al igual que emplearlas para devolver los productos o reclamar a los comercios.

Este ataque es imposible manteniendo los datos del usuario en un Cloud[3], donde se almacene las facturas del los usuarios registrados manteniendo seguros los datos. Dicho sistema permitirá al usuario tener un orden de las compras realizadas, recuperar las facturas en varios dispositivos y consultar las finanzas de una manera sencilla.

Objetivos

Los objetivos primarios que se van a buscar en este trabajo final de master son:

- Garantizar la seguridad y integridad de los datos en la obtención, almacenamiento y gestión.
- La creación de Cloud seguro donde se puedan realizar peticiones.
- Implementar una subida de información por parte del usuario.
- Creación de una aplicación que permita emplearlo.
- Realizar un estudio de los sistemas criptográficos para dicho ámbito y evaluar cual se adapta mejor a los objetivos anteriores.

Los objetivos secundarios que se pueden desarrollar:

- Creación de una web que mediante tecnologías de acceso se permita consultar los datos del usuario.

[3] Sistema informático que ofrece como servicio de modo que los usuarios puedan acceder a los servicios disponibles sin tener conocimientos en la gestión de los recursos que usan.

- Uso de librerías de visualización para realizar gráficos sobre dichas facturas electrónicas.

Justificación

Actualmente al realizar un pago la tienda debe entregar un recibo o una factura, que especifica que el cliente ha realizado un pago por un producto y lo ha obtenido. Esta forma de actuar esta sufriendo un cambio debido principalmente al ahorro económico ya que la factura electrónica elimina los gastos de impresión, papel, sellos, envío, etc. Pero las ventajas no solo son económicas sino que la inmediatez y seguridad del envió a través de soportes electrónicos es otro gran aporte. Otro benefició directo es la optimización en las gestiones de la empresa reduciendo los tiempos de gestión, se consigue una mayor eficiencia en el funcionamiento de la compañía. Los primeros en impulsar este método de facturación ha sido las entidades publicas que durante el 2015 deberían adoptar el formato de Facturae para emitir facturas y utilizar los servicios telemáticos para presentarlos, según la ley 25/2013 del 27 de diciembre [3].

Organización de la memoria

La memoria se organiza en 4 partes:

1. La introducción.
2. Las tecnologías empleadas y el diseño teórico, como los actores, fases y el diseño de la base de datos y aplicación.
3. Planificación y desarrollo, donde se comentan las partes y técnicas que se han empleado en el desarrollo.
4. Las conclusiones y trabajos futuros, donde se determinan como se ha resuelto el trabajo y que ha sido el proyecto resultante.

Tecnologías empleadas

Este apartado pretende mostrar las tecnologías disponibles y las que se han empleado para el desarrollo del sistema.

Podemos separar el proyecto en dos partes diferenciadas:

- Servidor: es el mismo Cloud.
- Cliente: es el usuario del servicio (Cloud).

Servidor (Cloud)

El Cloud es un sistema de almacenamiento en Internet que permite ser accedido desde cualquier punto. La primera decisión que se ha tomado es respecto a la base de datos si debe ser relacional o no relacional[4], en este caso donde queremos explotar los datos de las facturas electrónicas para mostrar al usuario detalles como gastos mensuales se ha apostado por una base de datos relacional.

La siguiente decisión ha sido donde alojar la aplicación Cloud, para ello se ha hecho un pequeño estudio de las empresas que ofrecen los siguientes requerimientos:

- Alta Disponibilidad: hace referencia a que el sistema debe estar funcionando las 24 horas de los 365 días al año.
- Acceso Mundial: debe ser accesible desde todo el mundo con un tiempo razonable.
- Almacenamiento de alto rendimiento: los datos deben gestionarse (lectura y escritura) de una manera rápida y debe ser tolerante a fallos.

Las empresas que ofrecen dichos servicios (según los requerimientos) son:

- Google Cloud Platform [4]: Este servicio lo aporta la infraestructura de Google y permite instanciar maquinas virtuales, guardar datos en los servidores de

[4] Una base de datos relacional es la que establece una relación entre los datos, las bases de datos no relacionales, se basan en documentos, donde sus datos no se relacionan con otros datos.

datos de Google y crear aplicaciones replicadas en todo el mundo. Este servicio aporta dos maneras de trabajar:

- Maquinas virtuales donde podemos alojarlas en los centros de Google y nosotros nos gestionamos la replicación (a nivel de capa de servidor) y el acceso mundial. Estas maquinas virtuales pueden tener cualquier sistema operativo pagando las licencias pertinentes, lo que permite que el lenguaje no sea un problema.
- App Engine [5]: Este servicio gratuito de Google permite ejecutar aplicaciones sobre la infraestructura de Google, donde se encarga de la replicación y la alta disponibilidad. Esto limita su entorno de programación a los lenguajes Java[5], Python[6], Go[7] y PHP[8] por la necesidad de crear una API para emplear los servicios de Google.

- Amazon Web Services [6]: Es un servicio de Cloud ofrecido por Amazon donde el coste es variable al uso del sistema. Ofrece servicios de maquinas virtuales y crear bases de datos replicadas en los servidores. Amazon únicamente ofrece maquinas virtuales, por lo que debemos gestionar la replicación (capa de servidor) y disponibilidad mundialmente.
- Microsoft Azure [7]: Es el servicio creado por Microsoft y el ultimo en lanzarse al mercado. Este servicio se basa en maquinas virtuales donde puedes elegir quien se encarga de replicarlas (únicamente las maquinas con sistema operativo Microsoft Windows) y posee una plataforma gratuita de aplicaciones estáticas donde no existe conectores de base de datos oficiales ni

[5] Lenguaje de alto nivel, orientado a objetos, se requiere una maquina virtual Java para poder correr los algoritmos.

[6] En un lenguaje de alto nivel, orientado a objetos, donde el código no es compilado sino interpretado y emplea tipado dinámico, que significa que la variable no conocerá el tipo de objeto al cual se le ha asignado, hasta que no se ejecute.

[7] Es un lenguaje compilado y concurrente para sistemas, inspirado por la sintaxis de C y lanzado por Google.

[8] Es un lenguaje de programación de uso general del lado del servidor orientado al desarrollo web y contenido dinámico.

replicación. También posee un servicio de base de datos basado en Microsoft SQL Server[9]

De los diferentes servicios se ha apostado por el Google Cloud Platform, por la disponibilidad de centro de servidores de Google respecto a sus competidores y por la gratuidad de la capa de aplicaciones ya que es la única que admite conexiones a base de datos.

Cliente

El cliente debe poseer un teléfono móvil o un navegador desde donde pueda acceder al sistema (según los objetivos). Las tecnologías móviles disponibles son las siguientes:

- Android [8]: Sistema operativo creado por Google, que fue diseñado inicialmente para dispositivos con pantalla táctil, aunque actualmente lo podemos encontrar desde relojes hasta coches. Android permite la instalación de aplicaciones de dos tipos:
  - Aplicaciones de alto nivel empleando Java como lenguaje de programación que se compila para que la maquina virtual que Android incorpora pueda ejecutarlas, se emplea una SDK[10] para el desarrollo.
  - Aplicaciones de bajo nivel, empleando C/C++ como lenguaje de programación que se compila y se instala al empleado la NDK[11]
- iOS [9]: Sistema operativo creado por Apple, para la gama iPod, iPhone y iPad, es un sistema operativo basado en Unix[12]. Se emplea lenguajes propietarios como Swift y Objective-C

---

[9] Es un sistema de manejo de bases de datos de modelo relacional, desarrollado por Microsoft y disponible únicamente para servidores Microsoft Windows.

[10] Software Development Kit incluye un conjunto de herramientas para el desarrollo para Android, como un depurador, biblioteca de herramientas y un simulador.

[11] Native Development Kit permite instalar aplicaciones escritas C/C++ y permite una interfaz a las llamadas de sistema disponibles en Android.

[12] Es un sistema operativo portable, multitarea y multiusuario, desarrollado en 1969

- Aplicaciones Hibridas: son aplicaciones que permiten funcionar en una variedad de sistemas operativos móviles como Android e iOS. Se basan en un framework[13] que permite desarrollar empleando herramientas genéricas y traducirlas para cada sistema operativo soportado. Los frameworks más conocidos se basan en programación web (HTML, CSS y JS) y el más conocido de ellos es PhoneGap [10] que se basa en Apache Cordova [11].

Sobre la decisión del cliente se ha optado por una programación para aplicaciones hibridas, ya que ha permitido solventar 2 objetivos, la creación de una aplicación para móviles y la creación de una web para que el usuario pueda acceder fuera del terminal. El diseño que se realice para web será adaptable[14] para los terminales, permitiendo que cualquier tipo de dispositivo sea compatible con el Cloud.

Diseño y Arquitectura teórica

En este apartado se detalla el diseño y la arquitectura de la aplicación. El diseño se basan de dos aplicaciones, la parte cliente y servidora (Cloud) y se toma en consideración desde que se ha obtenido el recibo, hasta que se guarda y se procesa al Cloud. Se ha empleado un diseño por UML [12], primero se ha definido los Actores y Casos de uso. Para cada caso se ha creado un diagrama de secuencia y los pasos documentados.

[13] Un conjunto de herramientas que ayudan a un programador a estructurar el código fuente y simplifica el trabajo mediante bibliotecas incluidas para la realización de acciones cotidianas.

[14] Diseño web adaptable, es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las paginas web al dispositivo que se esté utilizando para visualizarla.

Actores

En UML los actores son todas entidades externas al sistema que guarda relación con el y que le demanda una funcionalidad. Esta definición incluye a un actor representativo, el consumidor y se ha tomado en consideración la creación de otro actor para que el entendimiento del sistema sea más claro, el proveedor es el generador de la factura o recibo.

Casos de Uso

Los diagramas de casos de uso describen relaciones y las dependencias entre un grupo de usos y los actores que participan en el proceso, no representan el diseño y no describen los elementos internos de los sistemas, sino muestran un esquema de las características que tendrá el sistema. Inicialmente separamos los casos de uso por actores unificando los procesos en un gráfico UML.

Actor Consumidor:

- Registrarse: El usuario puede crear una nueva cuenta.
- Login en el Cloud: Puede iniciar sesión para ver la información alojada en el Cloud.
- Añadir un recibo o factura. Puede añadir subir una factura al sistema.
- Recibir / Descargar factura o recibo: Puede obtener una factura por parte del Cloud ya almacenada anteriormente.
- Generar un reporte: Puede generar informes sobre sus compras según las facturas o recibos alojadas en el sistema.

Actor Proveedor:

- Emitir recibo o factura: Puede emitir una factura de un producto comprado.

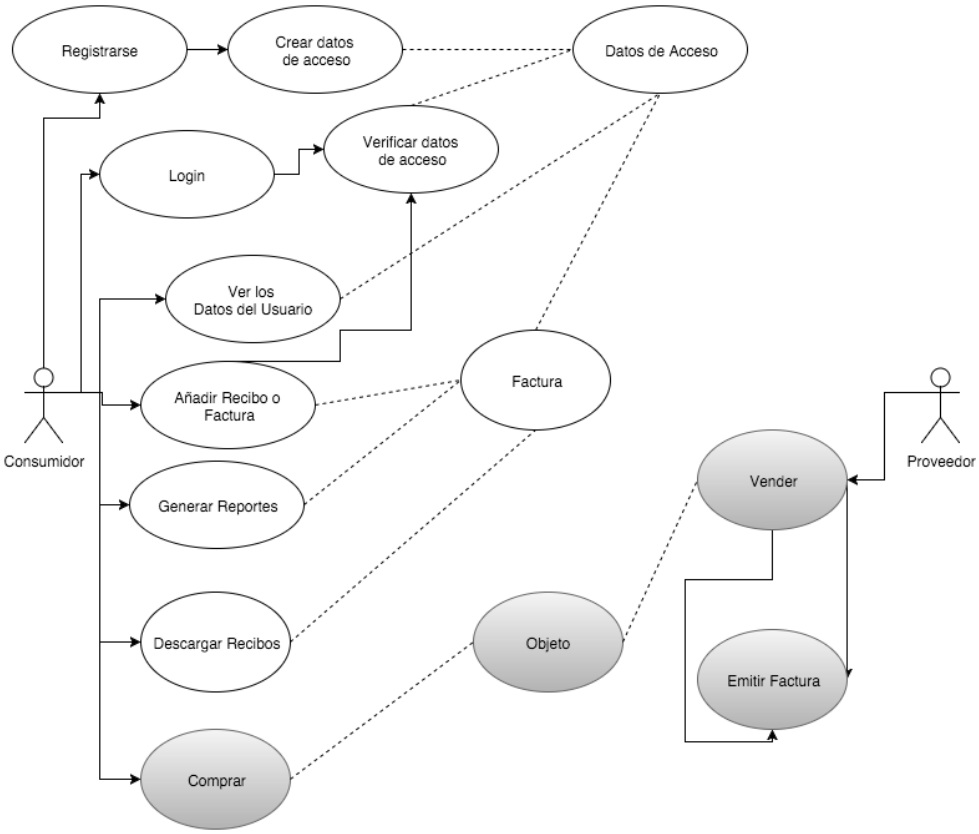

Figura 1: Casos de Uso

El proyecto parte de una vez que el usuario ha recibido la factura o recibo del proveedor. Las partes que actúan en el Cloud son las partes sin fondo de la Figura 1. Se han incluido en el diagrama de casos de uso para la descripción y el entendimiento del sistema.

Diagrama de Secuencia

En este apartado se muestran los diagramas de secuencia de la aplicación y los diagramas de clase empleados en los aplicativos.

Registro

Este paso permite crear un nuevo usuario en el Cloud.

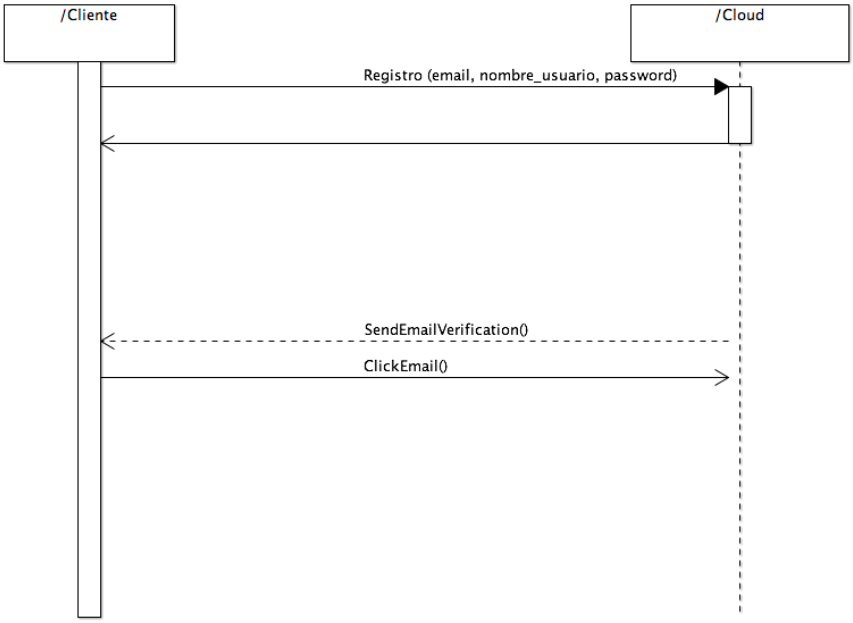

Figura 2: Diagrama de Secuencia - Registro

La Figura 2 muestra como el cliente realiza la acción de Registro. Este proceso se realiza en 3 pasos separados:

- La aplicación muestra una ventana donde se le piden al usuario los siguientes datos:
  - Email: como identificador único para el sistema.
  - Nombre del consumidor: permite realizar la interfaz más amigable.

- Contraseña: que se computa para no guardar contraseñas en claro.
- Esos datos son enviados al Cloud empleando un canal seguro y se crea una nueva entidad en la base de datos que permanece desactivada. El Cloud envía un email al cliente para verificar la cuenta y la identidad del usuario. Dicho email contiene un enlace único empleando una dirección activa durante un cierto tiempo.
- El consumidor debe acceder al enlace enviado por el Cloud, dicha acción activa la cuenta para empezar su uso.

Login

Esta funcionalidad permite entrar en el sistema una vez ya ha sido registrado en el Cloud.

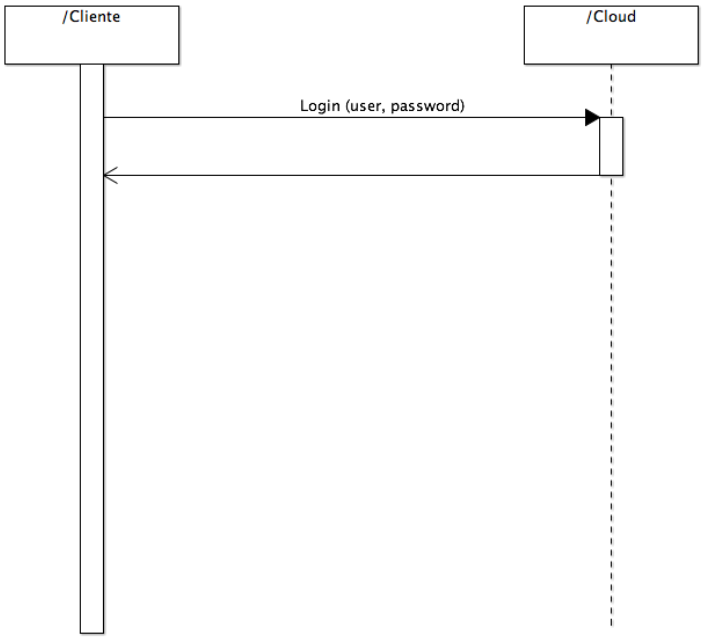

Figura 3: Diagrama de Secuencia - Login

En la Figura 3 se detallan los pasos que se emplean para realizar el inició de sesión cuya descripción es la siguiente:

- Una vez que el usuario desea iniciar la sesión en el Cloud, accediendo al sistema.
- Se introduce un usuario y contraseña. Registrados previamente en el sistema. El

software verifica los datos de acceso con los datos almacenados, si son correctos permitirá el acceso, para las acciones permitidas por el usuario. El Cloud envía un código que debe ser enviado en las siguientes peticiones.

Datos de Usuario

Esta funcionalidad permite al usuario consultar los datos personales en Cloud y las configuraciones.

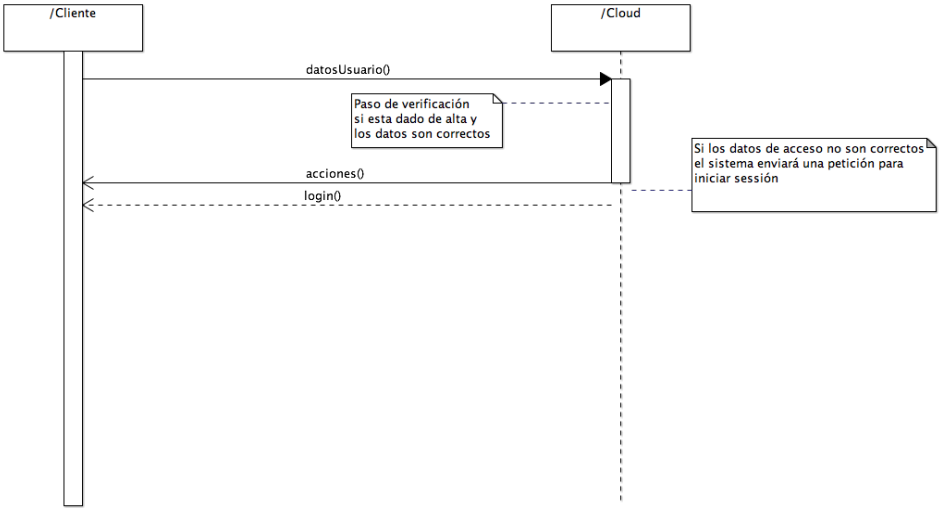

Figura 4: Diagrama de Secuencia - Datos del Usuario

Inicialmente se requiere que el usuario este registrado y con una sesión valida tal y como se muestra en la Figura 4, para ello este diagrama de secuencia detalla los siguientes pasos:

- El usuario desea ver los datos almacenados en el Cloud, para ello accede al sistema y le demanda los datos del Usuario.
- El sistema verifica si esta registrado en el sistema si contiene el código enviado en el inicio de sesión, si es correcto mostrará las acciones permitidas por el usuario.
- Si no es correcto el sistema enviará al cliente a la acción del Login para que el usuario realice el proceso.

Subida de facturas y recibos.

Este apartado define el diagrama que muestra la subida al Cloud de facturas.

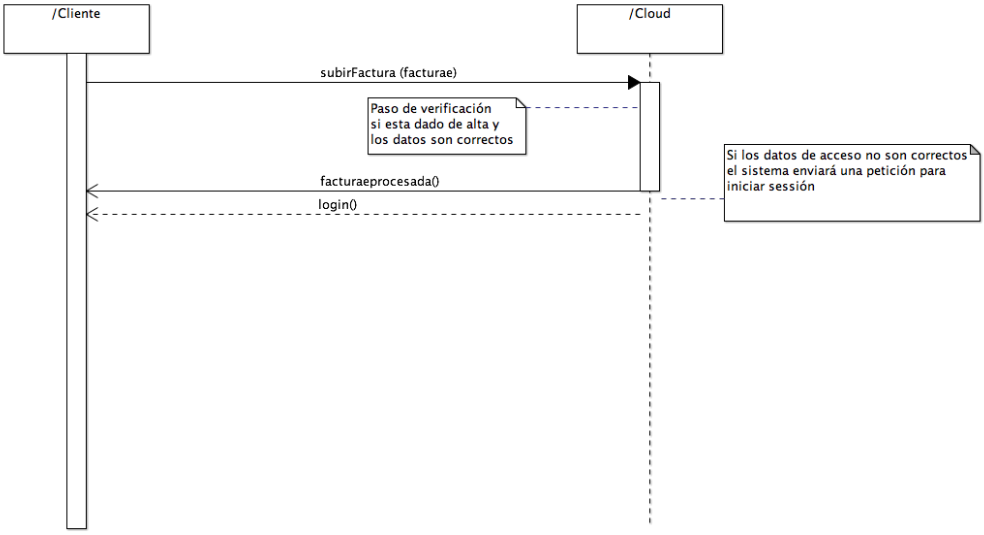

Figura 5: Diagrama de Secuencia - Subida de facturas y recibos

La Figura 5 muestra la subida de una factura electrónica al Cloud, los pasos que permiten realizar la acción son lo siguientes:

- El usuario empieza la acción accediendo al Cloud y realizando la acción de subida de factura.
- El sistema mostrara las carpetas para determinar donde buscar la factura y permitir la subida al Cloud.
- Al subir la factura el sistema comprobara las credenciales del usuario, en caso de que no las tenga, re-direccionará al registro.
- Si la subida es correcta el sistema creará una nueva entidad en la base de datos procesando la factura y emitiendo el detalle de la factura al cliente.

Generar reporte

Esta funcionalidad permite al cliente generar reportes de las compras realizadas y el gasto mensual.

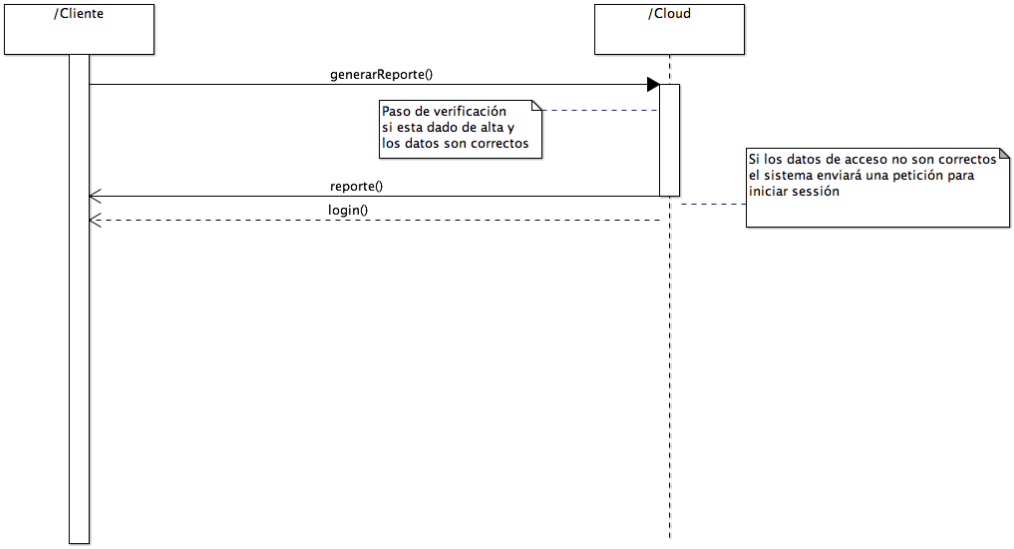

Figura 6: Diagrama de Secuencia - Generar Reporte

El reporte se genera al entrar el usuario en el Cloud como ventana de bienvenida. Pero el usuario si lo desea puede generar nuevos reportes y actualizarlos al después de subir una factura.

La Figura 6 muestra los siguientes pasos que permiten la generación de una nueva reporte:

- El usuario debe acceder al sistema con los datos de acceso
- Si los datos de acceso no son los correctos, el Cloud redireccionará al usuario a la acción de Login para que introduzca los datos correctos de acceso.
- Si los datos de acceso son los correctos el Cloud recuperará las facturas asociadas al usuario almacenadas en la base de datos y creará el reporte empleando los datos.

Descargar Factura

Esta funcionalidad permite descargar una factura almacenada en el Cloud.

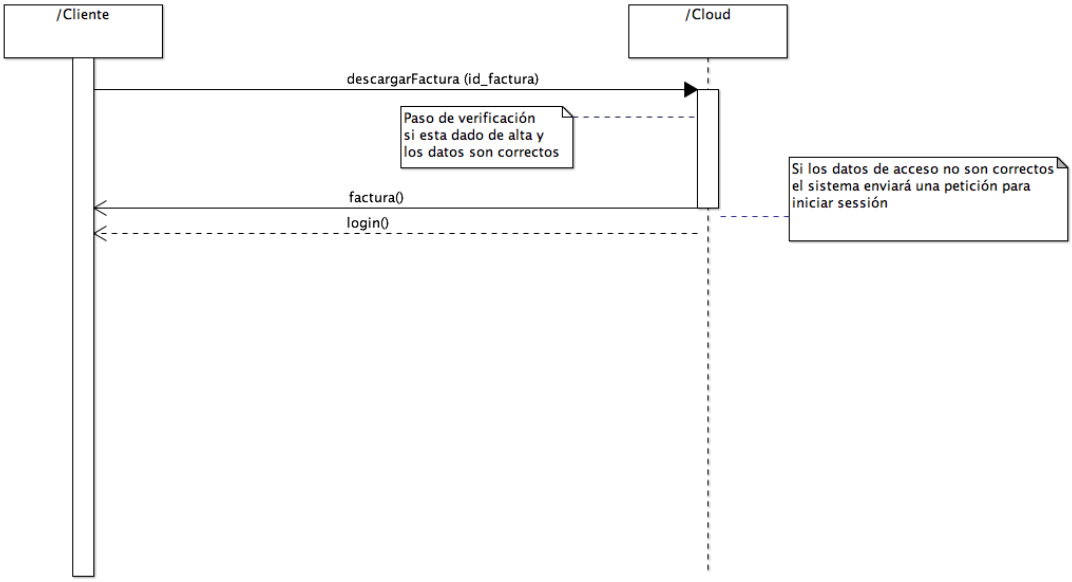

Figura 7: Diagrama de Secuencia - Descarga de Facturas

La Figura 7 muestra los pasos para descargar una factura almacenada en el sistema, para ello se emplean los siguientes pasos:

- El usuario desea descargar una factura almacenada, para ello accede al sistema y envía el mensaje de descargar la factura que desea.
- El Cloud verificará los datos del usuario y si existe dicha factura.
  - Si existe la factura descargarás la factura.
  - Si no existe la factura mostrará un error por pantalla.
  - Si el usuario contiene unas credenciales invalidas o no las contiene el Cloud redireccionará a la acción de Login

Diagrama de Clase

En este apartado se detallan las clases empleadas por el Cloud para cumplir todas las funcionalidades.

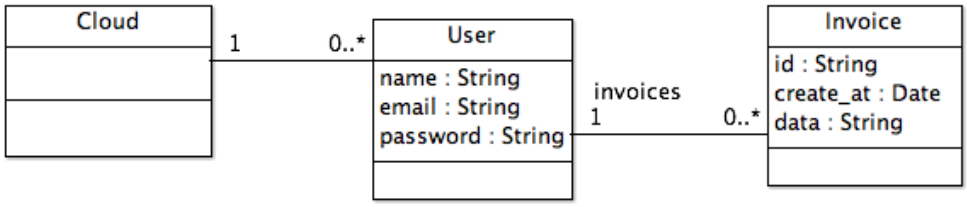

Figura 8: Diagrama de Clases

Como se muestra en la Figura 8 las clases empleadas por el Cloud son las siguientes:

- Cloud: Es el elemento central del sistema, contiene una lista de usuarios almacenados.
- Usuario: El usuario posee los datos, como el nombre de usuario, el correo electrónico y la contraseña computada. El usuario puede poseer facturas cargadas en el sistema.
- Factura: Depende del usuario creado y está vinculado a el. Se identifica por un id interno incrementativo, contiene la fecha de creación que es inmutable y la información sin procesar de la factura.

Planificación

Para la realización del proyecto se ha aplicado una metodología de trabajo iterativa, ya que el proyecto esta conformado únicamente por una sola persona, por ello se ha llevado a cabo una división en 3 paquetes de trabajo. Con fin de mantener un orden en la creación del proyecto y en caso de correr algún riesgo tener la posibilidad de entregar el proyecto dejando partes completamente funcionales y entregada.

Cada funcionalidad identifica una fase, que finaliza en un hito, las fases y subfases se describen a continuación, pudiendo llevarse a cabo en paralelo en un equipo compuesto de varias personas.

Primera Fase. Creación del Cloud

Como primera fase se creo el Cloud en Google Cloud Platform y se configuro según los requerimientos, comprobando el funcionamiento de la base de datos, los conectores y los frameworks disponibles.

Comenzar comentando que uno de los objetivos básicos que se impone el proyecto, como la utilización de Google App Engine para la creación de un Cloud, me llevo a considerar diversas opciones para el desarrollo.

Se realizo esta primera fase para crear un entorno de test lo más cercano al de producción y determinar que tecnologías a emplear dentro de la plataforma presentada por Google.

Esta fase también permitió decidir la utilización de un framework web para la gestión de peticiones por parte del Cloud.

Una vez determinado las decisiones sobre el diseño de la aplicación se realizaron las subfases siguientes

Comprensión de la Factura electrónica

El cometido de esta subfase fue comprender cual es el formato de una factura electrónica, como se generan las facturas electrónicas, como se deben enviar las facturas y la estructura de la firma que contiene y las versiones de la misma.

Creación de la Base de Datos

Esta subfase definió como iba a ser la base de datos y que campos son los requeridos, para el diseño de las funcionalidades según los objetivos establecidos y como se debía guardar una factura electrónica en la base de datos.

Segunda Fase. Creación de las funcionalidades

En la segunda fase se crearon las funcionalidades especificadas en el diseño teórico, también se tuvieron en consideración los cambios que pueda sufrir la plataforma para adaptarse a los cambios impuestos.

Tercera Fase. Creación de la aplicación

En esta fase se ha creado la aplicación web de diseño adaptable y compatible con teléfonos móviles y las funcionalidades que debe tener una web. Se han empleado librerías para el diseño, ya que nos ha facilitado crear el diseño adaptable.

También se ha creado la aplicación universal mediante el uso del framework PhoneGap.

Diagrama de Gantt

El diagrama de Gantt es una herramienta grafica cuyo objetivo es exponer el tiempo previsto para las diferentes tareas o actividades del proyecto a lo largo de un tiempo determinado.

Para el proyecto se separó en las 3 fases cada uno de los diagramas, pero las tareas de documentación se realizaron en paralelo a las de desarrollo generando la memoria a la misma vez que se hacia el proyecto como se muestra en Imagen 1

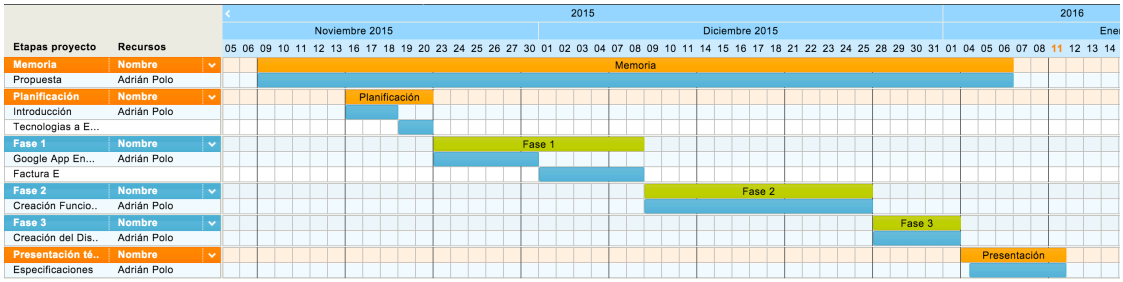

Imagen 1: Diagrama de Gantt

Control de versiones

El control de versiones es la gestión de los diversos cambios que se realizan sobre los elementos de un proyecto. Permite recuperar versiones anteriores y documentar las mejoras, trabajar conjuntamente con un equipo de desarrollo y automatizar procesos cotidianos y mantener el código creado ordenado. De las muchas herramientas se ha elegido GIT por la facilidad de uso y la gratuidad de herramientas online como GitHub [13].

Desarrollo

En esta fase se documenta el desarrollo según la planificación, con las decisiones obtenidas durante el proyecto y las razones por las cuales se han elegido.

Primera fase: Creación del Cloud

En la primera fase se ha creado las instancia en Google Cloud Platform, donde se define como un proyecto separado sin conexiones entre otras ni configuraciones compartidas, cada instancia se le permiten activar módulos que se separan en 5 bloques:

- Recursos Informáticos: son los módulos que aportan computo a la instancia. Esta desde la posibilidad de correr aplicaciones mediante Google App Engine a crear maquinas virtuales.
- Almacenamiento: este bloque contiene los módulos de almacenamiento persistente en los centros de datos de Google.
- Operaciones: módulos de herramientas de supervisión del uso de Cloud como el log de los otros elementos activados.
- Herramientas: módulos de test para realizar un juego de pruebas en el entorno creado.
- Big Data: herramientas para gestionar grandes bloques de datos.

Como el proyecto requiere de almacenamiento persistente mediante una base de datos relacional, la plataforma de Google tiene un modulo llamada Cloud SQL (almacenamiento) que se basa en un motor de MySQL modificado para aportar replicación entre los centro de datos de Google y alta disponibilidad. Actualmente existen dos tipos de base de datos en Cloud SQL:

- Primera Generación: tiene un rendimiento y capacidad de almacenamiento básico, donde las operaciones de copia de seguridad y mantenimiento están automatizadas.

- Segunda Generación: actualmente esta en versión beta, ofrece un rendimiento 7 veces mayor y capacidades de almacenamiento 20 veces superior por empleo de discos de estado solido, permite añadir conmutación por errores y replicación de lectura.

Inicialmente el proyecto se creo empleando la segunda generación por las grandes ventajas sobre la primera versión pero tiene limitaciones al estar en versión beta como la no compatibilidad con App Engine, teniendo que crear una base de datos de Primera Generación como base de datos.

En la instancia de Google Cloud Platform también se activó el módulo de App Engine, donde se recibió un enlace para acceder a nuestra aplicación y se nos pidió instalar un software en la maquina de desarrollo para realizar subidas de código. El modulo soporta 4 lenguajes de programación, de los cuales se escogió Pyhon como lenguaje para la creación web, ya que fue el primero en incluirse en dicha plataforma y la API que permite conectar con otros módulos de la plataforma de Google es la más completa y madura. Al ser un lenguaje de propósito general Google decidió incorporar librerías o frameworks para hacer más fácil el desarrollo. En el ámbito web ofrece dos librerías:

- WebApp2: Es un framework web ligero creado en Python donde se crean clases para cada una URL, especificando los métodos http como funciones de la misma. El código HTML que emplee el proyecto se escribe en el mismas mismas clases guardado como variables.
- Django: Es un framework de desarrollo web escrito en Python creado en 2005, que respeta el patrón de diseño conocido como Modelo – Vista- Controlador. Permite la administración de contenido de las aplicación pudiendo añadir, modificar y eliminar elementos, plantillas HTML con elementos dinámicos empleando etiquetas y la gestión de bases de datos empleando ORM.

Google App Engine emplea el archivo de configuración llamado app.yaml, dicho archivo permite a la plataforma entender como debe ejecutar el proyecto creado y contiene los siguientes campos:

- Aplicación: Contiene el identificador que genera Google Cloud Platform al crear la instancia.
- Versión: Es la versión del proyecto.
- Runtime: Versión del compilado o interprete para correr la aplicación
- Versión de la API: Versión de la interfaz de aplicaciones para que terceras partes accedan.
- ThreadSafe: En la programación multihilos, indica si el código a ejecutar es seguro a emplear más de un hilo
- Handlers: Las rutas las cual escucha Google App Engine y redirecciona a los controladores.
- Librerías: son los paquetes de terceras partes que emplea el proyecto y proporciona Google

Para el proyecto se ha decidió emplear por Django por las múltiples funcionalidades que emplea y con la suficiente madurez para este tipo de proyectos.

Django [14]

Permite la creación de sitios complejos, poniendo en énfasis el re-uso, la conectividad y la extensibilidad de sus componentes, el desarrollo rápido y el principio No te repitas (DRY, del ingles "*Don't Repeat Youself*") [15], esta escrito en Python y todas las partes que lo componen están escritas en Python.

La arquitectura de Django [16]

Aunque el modelo esta inspirado en el Modelo – Vista – Controlador, los desarrolladores declararan que no es el paradigma convencional sino una modificación de conceptos, donde la vista es el código HTML llamado plantilla y el controlador es la vista que mapea el contenido dinámico en la plantilla quedando un paradigma Modelo – Plantilla – Vista. Este framework se basa en las siguientes capas:

- Presentación: Aquí se maneja la interacción entre el usuario y la web, esta capa la forma el template loader que cargar los templates HTML del modulo y template engine que modifica el HTML para añadir los elementos dinámicos y servirlos al usuario, en esta capa también se incluye un mapeador de URL basado en expresiones regulares.
- Control: En esta capa reside la lógica de la aplicación, donde las vistas y los manejadores reciben las peticiones http y actúan en consecuencia empleando la capa de presentación para servir el contenido.
- Entity: Maneja los objetos de negocio, los cuales interactúan con la capa de Control.
- Foundation: Es manejar a bajo nivel el trabajo con la base de datos, donde provee soporte para varias bases de datos y proporciona una API de conexión.
- Mediator: Es el encargado de gestionar la API de la capa Foundation para mapear los resultados de las búsquedas en objetos Entity y viceversa, permitiendo proveer a la aplicación el ORM.

Seguridad en Django [17]

Django siguiendo el principio “No te repitas” ofrece una capa de seguridad actualizada a las vulnerabilidades que van apareciendo. A continuación se detalla los ataques que protege

Protección contra Cross Site Scripting (XSS)

Los ataques XSS permite al usuario inyectar secuencias de comandos del lado del cliente en los navegadores de los usuarios. Esto generalmente se logra mediante el almacenamiento de los scripts maliciosos en la base de datos donde se recupera y se muestra a otros usuarios, o por conseguir a los usuarios hacer clic en un enlace que hará que JavaScript del atacante sea ejecutado por el navegador del usuario. El uso de plantillas de Django ofrece protección contra la mayoría de los taques XSS, escapando de caracteres específicos que son peligrosos para HTML. Pero se recomienda especial cuidado al emplear etiquetas “is_safe” en las plantillas ya que no se realiza el escapado.

Protección contra Cross Site Request Forgery (CSRF)

Estos ataques permiten a un usuario malicioso ejecutar acciones mediante las credenciales de otro usuario sin el conocimiento y consentimiento de ese usuario. El framework incorpora una protección mediante la comprobación de un NONCE[15] [18] en cada solicitud POST. Esto asegura que un usuario malicioso no puede simplemente “replay” de un formulario POST al sitio web y haber otro usuario conectado sin querer presentar ese formulario. El usuario maliciosos debería tener conocimiento de dicho NONCE, que se almacena en el usuario empleando una cookie.

[15] El nonce es un valor que sirve como identificador único y solo es posible emplearse una vez.

Al implementarse HTTPS, el modulo “CsrfViewMiddleware” comprobará que al cabecera http árbitro se establece en una dirección URL en el mismo origen. Debido que HTTPS incorpora una seguridad adicional, es imprescindible para garantizar las conexiones.

Protección contra SQL Inyection

La inyección de SQL es un tipo de ataque en el que el usuario malicioso es capaz de ejecutar código SQL arbitrario en la base de datos, pudiendo borrar registro o generar una fuga de datos.

Django incorpora el uso de QuerySets que escapa el código SQL resultante por el controlador de base de datos, para evitar los caracteres que permitan este tipo de ataque. Aunque al desarrollador puede emplear consultas a la base de datos Django recomienda emplear QuerySets para el acceso a la base de datos.

Protección contra Clickjacking

Clickjacking es un tipo de ataque en un sitio malicioso que envuelve otro sitio en un marco. Este ataque puede dar lugar a un usuario desprevenido siendo engañado realizando acciones no deseadas en el sitio de destino. El framework posee una middleware que bloquea que la web sea introducida dentro de un marco.

Empleo SSL / HTTPS

Para una mejor seguridad en la web se requiere desplegar el sitio detrás de HTTPS, para evitar que usuarios maliciosos consigan las credenciales de autenticación o cualquier otra información que se transfiera entre el cliente y el servidor. Si se va a emplear SSL, Django requiere que se ajuste para bloquearse contra los ataques de CSRF y realizar el ajuste para redireccionar a los usuarios que accedan por http redirigirlos a HTTPS. Si un navegador se conecta inicialmente a través de http, que es el valor por

defecto para la mayoría de los navegadores, es posible que el servidor envié cookies por el canal inseguro haciendo que un atacante obtengas las cookies para realizar ataques CSRF, para ello debemos activar que solo se envíen las cookies a través de HTTPS. Django también es compatible con el HTTP Strict Transport Security (HSTS) [19] que es un mecanismo de política de seguridad web la cual un servidor web declara a los usuarios compatibles (navegadores web) que deben interactuar con ellos únicamente mediante conexiones http seguras donde ese especifica un tiempo durante el cliente deberá acceder al servidor de forma segura.

Las usuarios en Django [20]

Django posee un modulo de usuario y grupos, dicho modulo permite la integración mediante otros módulos, almacenando los datos más característicos de un usuario.

Inicialmente existen dos grupos de usuarios, pudiendo crear más y añadir funcionalidades para cada uno de ellos. El grupo de administradores, que permite el acceso a la parte administrador de Django para la creación de elementos, modificación y borrado de elementos, y el grupo de usuarios que no tiene acceso a dicha parte de gestión.

Si se activa el modulo de administración Django pide la creación de un usuario que pertenece al grupo de administradores.

Protección de las contraseñas de los usuarios en Django

En este paso se define la seguridad que empleará Django para el almacenamiento de las contraseñas para los usuarios. Django emplea una función hash SHA256 que es una función criptográfica que se caracteriza por ser de un solo sentido; es decir, es fácil calcular el código hash de un determinado valor, pero es prácticamente imposible

reconstruir el valor original partiendo únicamente del código hash. Si almacenáramos las contraseñas como texto en claro, cualquiera que pudiera obtener acceso a la base de datos podría saber sin ninguna dificultad todas las contraseñas al instante. Al guardar las contraseñas en forma de códigos hash se reduce el peligro en caso de que se comprometa la seguridad de la base de datos.

No obstante, un atacante que pudiera acceder a la base de datos podría realizar un ataque por fuerza bruta, calculando los códigos hash de millones de contraseñas distintas y comparando esos códigos con los que están almacenados en la base de datos.

Para empeorar las cosas, hay disponibles públicamente lo que se conoce como tablas arco iris (rainbow tables) [21], que consisten en valores hash precalculados de millones de contraseñas de uso habitual. Usando una tabla arco iris, un atacante puede romper la mayoría de las contraseñas en segundos.

Para aumentar la seguridad y evitar este tipo de ataques, se ha configurado Django para que añada un valor inicial aleatorio y diferente a cada contraseña antes de obtener el código hash. Este valor aleatorio es el "salt". Como cada salt es diferente para cada password se evita el uso de tablas arco iris, lo que obliga al atacante a volver al sistema de ataque por fuerza bruta, que a su vez es más complicado al haber aumentado la entropía con el salt.

Otra ventaja es que si dos usuarios eligen la misma contraseña, al añadir el grano de sal los códigos hash resultantes serán diferentes.

Estructura de una aplicación en Django

Django estructura las aplicaciones en módulos independientes. Al crear una aplicación esta genera el modulo base y el modulo de la aplicación. A continuación se detalla las estructuras de los dos módulos.

Modulo Base

El modulo base se compone de 2 partes diferenciadas:

- Archivo de configuración: Este archivo contiene la configuración del proyecto y contiene los siguientes parámetros:
    - De un parámetro para determinar si se esta ejecutando la aplicación en modo Debug.
    - Las conexiones a la base de datos, Django posee múltiples conectores de base de datos, en posible definir más de una conexión a base de datos dependiendo el entorno donde se este ejecutando.
    - La zona horaria donde correrá el servidor principal.
    - El código del lenguaje para el que ha sido creado y así permite la internalización
    - La ruta de los documentos estáticos, como imágenes, librerías para frontend.
    - Los directorios estáticos dentro de la ruta de los documentos.
    - La clave secreta: se emplea para proporcionar una firma criptográfica para la aplicación y se emplea en:
        - Las sesiones si se emplea el modulo de sesiones que aporta por Django.
        - Los mensajes que se emplean por el almacenamiento de Cookies.
        - Al realizar un reset de un password de usuario.

- Una lista con los módulos de carga de templates, donde cada uno de ellos aporta una funcionalidad diferente a los templates.
- Dirección de los ficheros de las URLs del proyecto.
- Directorio donde están los templates.
- Listado de aplicaciones instaladas en el proyecto.
- Estructura para el sistema de logs, se especifica una estructura json para la

- Archivo de URL: este archivo contiene una lista de expresiones regulares que simbolizan una URL.
- Archivo WSGI [22]: Este archivo contiene el Web Server Gateway Interface que es una interface simple y universal entre los servidores web y las aplicaciones web o frameworks, permite que un servidor web implementando la interfaz propuesta por WSGI sea capaz de entender todo los tipos de frameworks web por encima escritos en Python tal y como se puede observar de forma grafica en la Figura 9.

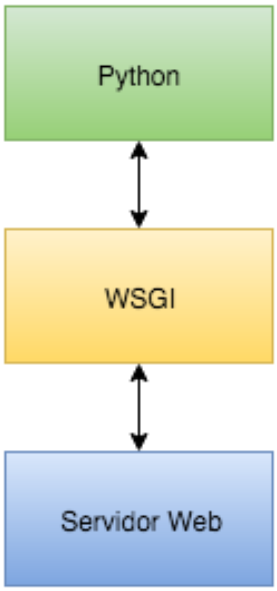

Figura 9: Explicación visual WSGI

Modulo de Aplicación

El modulo de aplicación es un componente del paquete y puede existir más de uno, según los requerimientos de la aplicación. La estructura básica de este tipo de modulo son los siguientes:

- Models: Aquí se detallan mediante clases los objetos de negocio que deseamos guardar en base de datos, estos heredan de una super clase que permite guardarlos en la base de datos y parsearlos directamente de ella.
- Views: Aquí esta la lógica del controlador, se definen métodos que son llamados cuando un usuario se conecta y emplea la url que concuerda con la expresión regular asociada al método configurado en el modulo base, el archivo de URL.
- Admin: Si hemos activado el modulo de administración que nos proporciona de forma automática debemos introducir los modelos que queremos poder modificar, agregar o borrar en la parte de administración.
- Test: Aquí se encuentran los test unitarios del modulo.
- Forms: En caso de necesitar un formulario de POST podemos emplear una clase que estructure que campos se van a requerir y Django añadirá automáticamente dicho forma con sus validaciones de los campos como un elemento dinámico en el template.

A cada nuevo modulo de aplicación que ser cree se debe añadir en el apartado de las aplicaciones instaladas del archivo de configuración del modulo base. Para que Django conozca de la existencia de dicho modulo.

Integración de Django y Google App Engine

Para la integración de Django con Google App Engine se debe realizar una modificaciones en archivo YAML de configuración. Las modificaciones a realizar son las siguientes :

- Ya que Django posee un mapeador de URLs esto entra en conflicto con el proporcionado por Google, para ello empleamos la interfaz WSGI que posee Django para conectar los handlers del archivo de configuración con la interfaz. Para que cuando Google App Engine reciba una petición y acceda a los handlers emplee WSGI para enviarle la petición a Django como se muestra en la Figura 10

Figura 10: Diagrama de Elementos - Integración de Handlers por WSGI

- Para liberar la carga de Django y así poder enviar más peticiones, el framework no debe servir archivos estáticos, para ellos se añadió un handler en el archivo de configuración que especificaba que cualquier ruta que contenga la carpeta /static/ la servirá Google App Engine de una carpeta del proyecto.
- Finalmente para poder correr Django en Google App Engine se debe añadir al apartado de las Librerías que el proyecto depende de Django.

La Factura Electrónica

La factura electrónica [23] tiene los mismos efectos legales que una factura en papel. La factura es un justificante de la entrega de algún bien o la prestación de algún servicio. La definición concreta es que una factura electrónica es una factura que se expide y recibe en formato electrónico y se plantea como una alternativa a la factura en papel. El problema que se nos presenta es que una factura puede variar de un país

a otro y eso pasa lo mismo en el formato electrónico, en España se creo el formato factura-e.

El formato de la factura-e [24]

La factura-e llega por por un medio electrónico en formato XML [25], dicho formato es un lenguaje de marcas desarrollado por World Wide Web Consortium utilizado para almacenar datos de forma legible.

La factura-e se separa por versiones que se van actualizando para corregir errores encontrados, actualmente el formato se encuentra en la versión 3.2.1 de la resolución del 21 de marzo del 2014. El modelo de versionado [26] se basa en la nomenclatura de los proyectos de software, donde se especifican 3 valores para definir la versión:

- El mayor (Primer numero): Cuando se produce un cambio que es incompatible con alguna versión anterior, se aumenta el numero y se elimina la retro-compatibilidad con las versiones anteriores.
- El menor (Segundo numero): Cuando se incrementa es por que se ha añadido alguna funcionalidad compatible con las versiones anteriores, permitiendo la retro compatibilidad con versiones anteriores.
- Patch (Tercer numero): Aumenta cuando se corrigen errores que no modifican ninguna funcionalidad, no realizan cambios en el comportamientos

Esto simboliza que actualmente se admiten facturas en la versión 3.0 y superiores pero el Gobierno de España recomienda actualizar las facturas a la versión 3.2.1,

La factura electrónica en su ultima versión en se compone de 4 elementos principales:

- FileHeader: Contiene la cabecera del fichero XML. Dicha cabecera muestra la versión de la factura, la modalidad de las factura, ya que existe la posibilidad de incluir más de una factura en un documento especificando que la modalidad es en Lote. El tipo de emisor de la factura, que puede tomar los valores de

proveedor, destinatario o tercero. También incluye la firma de una tercera parte que contiene los datos de la misma, el Lote que es un pequeño resumen de la factura como el coste de las facturas y el numero de ellas.

- Parties: Contiene los datos de los sujetos, tanto los datos del emisor y el receptor de la factura. Contiene además la información que suele incluir en formato papel.
- Invoices: Conjunto de facturas que contiene el fichero. Cada factura posee una línea de elementos o servicios.
- Extensions: Son las definiciones estructuradas en formato para cuando sean del interés de las dos partes y no estén descritos en la factura-e.
- Firma: Conjunto de datos asociados a la factura que garantizarán la autoría y la integridad del mensaje. Se define como opcional para facilitar la verificación y el tránsito del fichero. No obstante, debe cumplimentarse este bloque de firma electrónica para que se considere una factura electrónica válida legalmente frente a terceros.

Creación de la base de datos

Para la creación de base de datos se ha empleado el diseño de clases y se ha creado un modelo relacional del mismo. Las facturas dependen del usuario y el Cloud únicamente tiene usuarios. Pero la cosa se complica al obtener el objetivo de mostrar información al usuario sobre sus movimientos en el Cloud, para ello podíamos adoptar dos formas de afrontar el problema:

- La primera era que cuando el usuario generará un nuevo reporte leemos todas las facturas almacenadas del usuario para obtener los valores deseados y creamos el nuevo reporte, el problema de esta forma de afrontar el problema es que si muchos usuarios demandan esta funcionalidad puede surgir problemas de rendimiento de la aplicación por la recuperación de los documentos en base de datos realizar las tareas de conversión de la

información, buscando los valores deseados de dichas facturas y realizar las operaciones pertinentes y mostrar al usuario, tal y como se puede observar en la Figura 11

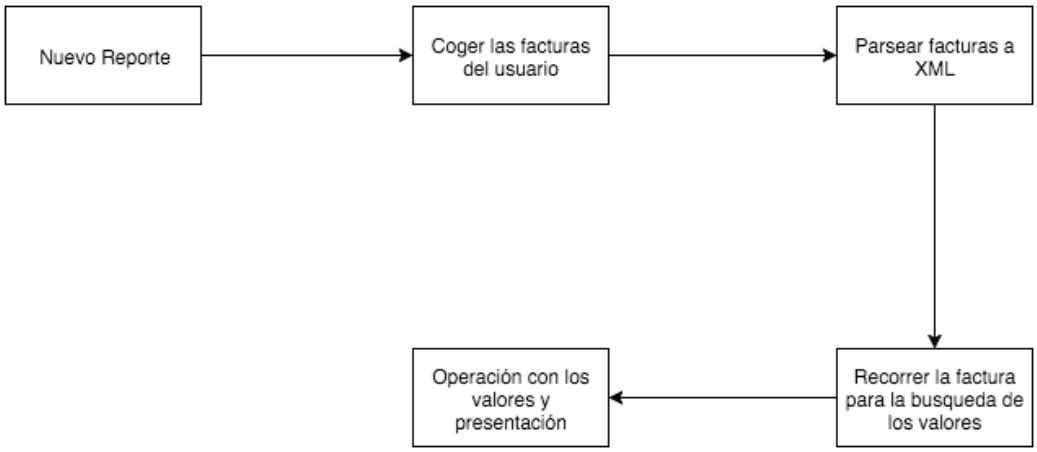

Figura 11: Generación de un nuevo reporte almacenando únicamente las facturas

- El segundo modelo que se pensó fue que cuando el usuario subiera una nueva factura el sistema parseará la información guardándola en la base de datos para que al generar un nuevo reporte los pasos a seguir fueran la obtención de los valores deseados del usuario y mostrarlos en la base de datos.

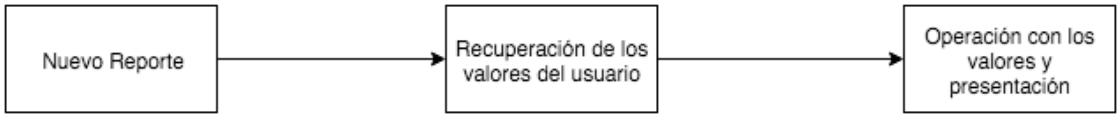

Figura 12: Generación de un nuevo reporte, empleando datos almacenados.

Tal y como se muestra en la Figura 12 podemos ver que los pasos se han reducido a 3, lo que implica que es la mejor opción. El problema se nos presenta cuando tenemos que subir un fichero que ya que debemos hacer un sistema de obtención de los valores de las facturas como se muestra en la Figura 13, dicho sistema implico la creación de una base de datos más compleja ya que debemos obtener valores predeterminados.

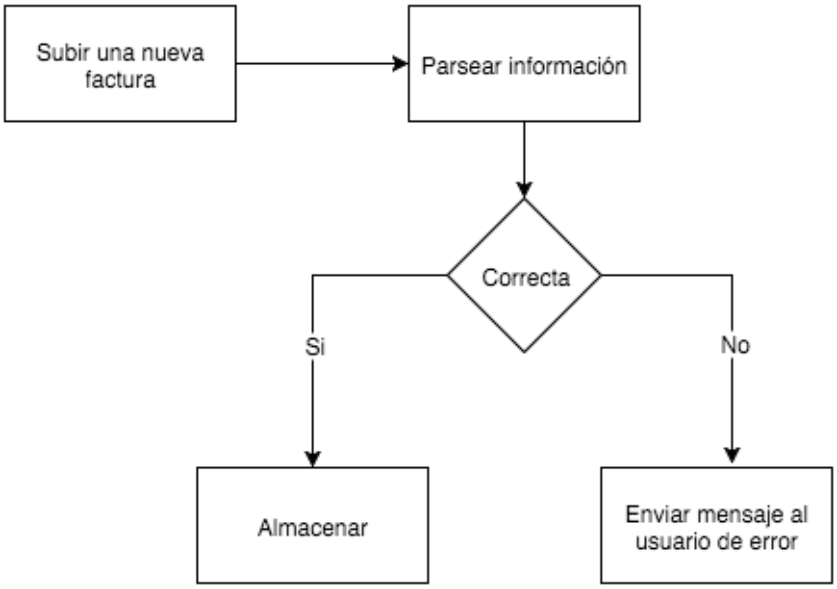

Figura 13: Diagrama de la subida de una factura empleando parseo de información

Por temas de rendimiento se escogió la segunda versión de trabajo y por que el framework Django permite actualizar el modelo y mediante un comando actualizar nuestra base de datos permaneciendo los valores que ya han sido almacenados.

El nuevo diseño de la base de datos se pensó para almacenar todos los valores de las facturas que se subieran para dar una mayor funcionalidad en un futuro, como la posibilidad de mostrar información más precisa y mejor organizada al usuario. Para ello se crearía una entidad que almacenará el usuario que la subiera, la fecha de subida, la factura y los campos internos de la misma. Ya que por tiempo no es viable la creación de los campos a mano se creo un proyecto en Python que permite cargar un documento y obtener las clases de la base de datos en cualquier formato, se creo únicamente la interfaz para Django, pero se dejo preparada una pequeña interfaz que permite extraer los valores de las clases.

El proyecto para la extracción del modelo

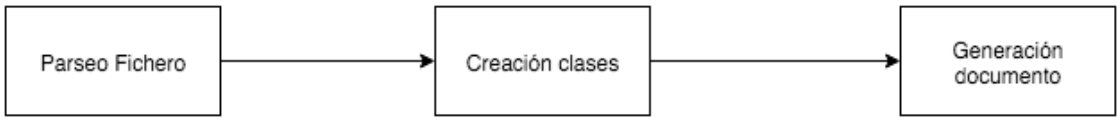

Figura 14: Proyecto para la extracción del modelo de base de datos

El proyecto se creo en Python en formato aplicación de consola. El código se estructura en 3 partes, como muestra la Figura 14 que se definen como:

- El parseador de información: Esta clase lee el documento recibido y genera unas estructuras de datos empleando los valores del documento.
- La creación de clases: Esta clase crea las clases con la información obtenida por el mapeador de información, en este clase esta la lógica de la aplicación ya que es capaz de entender el documento obtenido por el paso anterior y crear las clases, también incluye un método de ordenación de las clases generadas ya que hay lenguajes que no se permite el empleo de un objeto si no se ha declarado con anterioridad.
- Generación del documento: Es una pequeña interfaz con unas funciones a seguir que permite la conversión a cualquier lenguaje, en este caso se ha empleado para Django. Donde la creación de clases llama a unas funciones para la creación de los nuevos campos para finalmente llamar a la función de escribir, que permite extraerlo en el formato deseado.

El fichero de entrada

El gobierno de España al generar un fichero XML genera el correspondiente fichero XML Schema (XSD) [27] que es un documento que describe la estructura y las restricciones de los contenidos en los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Consiguiendo así una percepción del tipo de documento al cual representa con un nivel alto de abstracción. Se basa en formato XML con etiquetas no estándares.

Resultado del software, entidades de base de datos.

Después de ejecutar el software creado hemos obtenido un documento en el formato de estructuración de Django, que hemos copiado a la estructura de modelos y mediante el comando de sincronización de la base de datos, que permite sincronizar el modelo de la base de datos con las clases creadas en el framework con todas las restricciones del modelo, hemos obtenido el diagrama de la Figura 15

Figura 15: Diseño de la base de datos

Segunda Fase. Creación de las funcionalidades

En este apartado se va a comentar los pasos que se han seguido para la creación de las funcionalidades.

La creación del Registro

En este apartado creo al registro al Cloud. El primer paso que se creo fue añadir los módulos de gestión de usuarios a las aplicaciones instaladas del archivo de configuración del proyecto base. Una vez añadido se creo un modulo de aplicación para añadir los controladores y gestionar el registro.

En el nuevo modulo de aplicación se creo en las vistas una clase que permite realizar el registro, el función se accederá cuando acceda a la URL configurada en el proyecto base y seguirá los pasos que se muestran en la Figura 16.

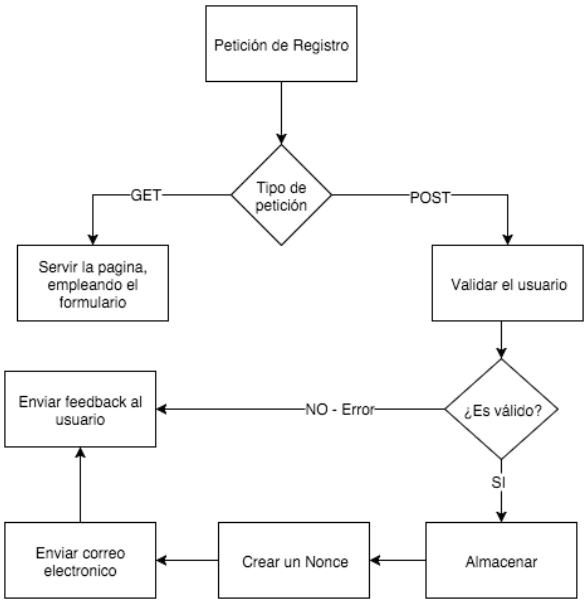

Figura 16: Diagrama de Flujo del algoritmo de registro

Se tomo en consideración que el envió de información por la parte de cliente se realizó empleando una petición POST, por esa razón esta función primeramente comprueba si la petición que envía el cliente es una petición de datos de parte del servidor (GET) y una petición de subida de información (POST)

- Si es una petición GET Django envía la pagina con el formulario que la contiene.
- Si es una petición POST validamos si los datos pasados por el usuario son validos y no existen en la base de datos. Si es valido se almacena el usuario como no activado, se crea un NONCE de usuario el cual se envía por mail empleando la API de google disponible para ello y se envía una notificación al cliente. Si el usuario no es valido se envía un el error notificando al usuario la causa del rechazo.

El código resultante del algoritmo se muestra en la Imagen 2

```
def register(request):
    '''
    Función que permite el registro de un usuario
    :param request: Request enviado por el cliente
    :return: renderizado del template con la inforamción del contexto
    '''
    error_messages = []
    if request.method == 'POST':  # Si es una petición POST
        form = RegisterForm(request.POST)  # Llenamos el formurio
        if form.is_valid():  # Formulario contiene información valida.
            email = form.cleaned_data.get('email')
            username = form.cleaned_data.get('user')
            password = form.cleaned_data.get('password')
            user = User(username=username, email=email)
            user.set_password(password)
            user.save()
            if user is None:
                error_messages.append("Nombre de Usuario o contraseña incorrectos")
            else:
                userNonce = UsersNonce()  ## Creación del Nonce
                userNonce.Owner = user
                userNonce.Code = user.id
                userNonce.save()
                message = mail.EmailMessage(sender="correo@invoiceCloud.com",
                                            subject="Cuenta creada")
                message.to = email
                message.body = "Cuenta creada accede a https://ruta" + str(
                    user.id) + " \n \n Para la activación del usuario" + "\n\n\n Gracias \n El equipo de InvoiceCloud"
                message.send()
                context = {
                    'username': username
                }
                return render(request, 'users/register.html', context)
    else:  # Si es una petición GET
        form = RegisterForm()# Creamos el formulario vació generará los campos y los permitirá añadir de forma dinamica
        # al template
    context = {
        'errors': error_messages,
        'register_form': form
    }
    return render(request, 'users/register.html', context)
```

Imagen 2: Algoritmo del registro

Validación del NONCE y activación del usuario

El siguiente paso de la creación del usuario requiere que el usuario compruebe el correo electrónico y acceda al enlace enviado. Para ello se ha añadido en la configuración de las URL el patrón que se muestra en la Imagen 3 donde se especifica que todas las URL acabadas con ac/<cualquier numero entero> estarán activando una cuenta de usuario

```
url(r'^ac/(?P<id>\d+)', 'users.views.activate_account', name='activate_account'),
```

Imagen 3: URL con variable para la activación de la cuenta

En proceso que permite la activación del usuario es la siguiente. Si el usuario ha puesto un enlace correcto y existe el NONCE en la base de datos, el usuario se activa borrando el NONCE y enviando el feedback al usuario de que la operación ha sido correcta. Si el usuario ha puesto un enlace no correcto se enviará un mensaje al usuario comentando que la dirección a emplear no existe, por lo que si emplea dos veces el enlace de usuario la segunda vez mostrará un error ya que no existirá el NONCE en la base de datos. Dicho proceso se presenta de forma visual en la Imagen 4

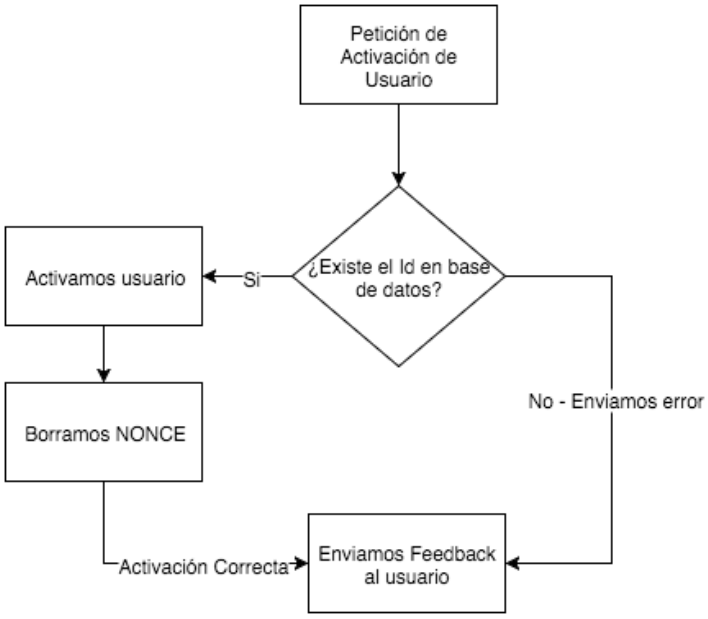

Imagen 4: Diagrama de secuencia de la activación del usuario

La creación del Login

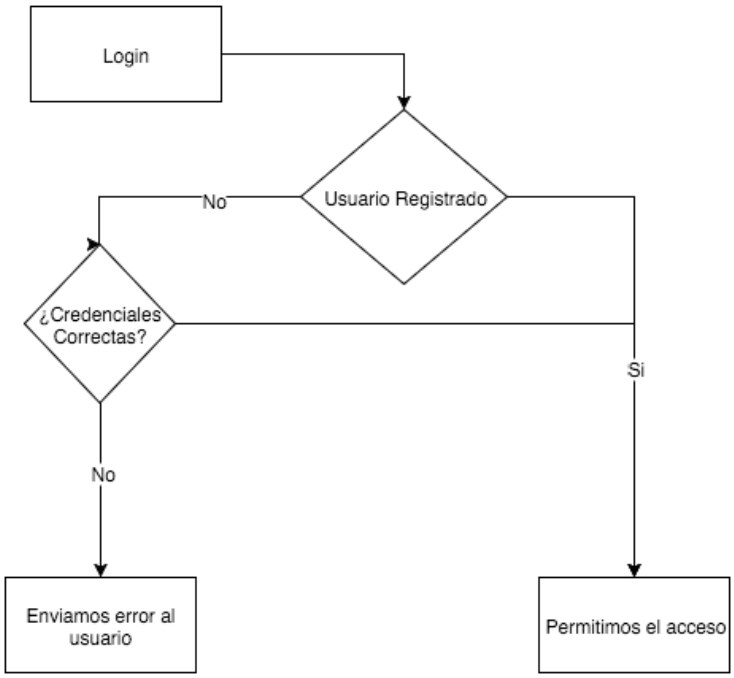

Imagen 5: Diagrama de Flujo - Login

En este paso se creo la lógica de acceso para los usuarios del Cloud, donde los usuarios deben acceder empleando el usuario y contraseña aportado en el registro. Para ello se añadió al modulo base la nueva URL que mapea las peticiones al controlador para el inicio de sesión. El proceso que implementa la lógica del inició de sesión(Imagen 5) sigue los siguientes pasos:

- Verifica que el usuario no este registrado.
    - Si el usuario esta registrado, permite el acceso.
    - Si el usuario no esta registrado y ha accedido empleando unas credenciales correctas Permite el acceso, sino envía el error al usuario.

La lógica de este proceso al emplear Django permite ya que la funciones ofrecidas por Django para la autentificación y el envió de la variables de sesión se realizan de manera automática.

La subida de facturas

Como un prerrequisito para la subida de facturas al sistema se requiere que el usuario este registrado. Si no hubiésemos empleado un framework debemos realizar las acciones para comprobar los valores de sesión, pero al emplear Django poseemos una herramienta llamada decorador[16] que nos permite comprobar si un usuario esta registrado antes de ejecutar la acción del controlador, permitiendo reducir la complejidad del proceso a los siguientes pasos ():

- Al recibir una factura la recibimos por medio de un formulario que validamos si existe un fichero.
- La siguiente validación verifica si los datos enviados por el usuario tienen el formato XML, realizamos un mapeado del documento creando las clases como se definen en la base de datos.
    - Si el proceso de verificación acaba correctamente tenemos el conjunto de clases de base de datos, las cuales empleando el comando de "save" realizamos la iteración para almacenar en la base de datos. También guardamos la factura en formato base64[17] para la posterior descarga.
    - Si el proceso de verificación no es correcto se renderizará la pagina web del formulario mostrando los errores generados por el mapeo.

Para el mapeo de xml se ha empleado una librería que incorpora Python que permite convertir un documento XML en un árbol de elementos.

[16] Un decorador nos permite realizar modificación simples en objetos llamables como son las funciones ,métodos o clases, estas modificaciones se realizan en tiempo de ejecución y permiten añadir nuevas funcionalidades.

[17] Es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII. Esto ha propiciado su uso para codificación de textos pudiendo almacenar caracteres especiales.

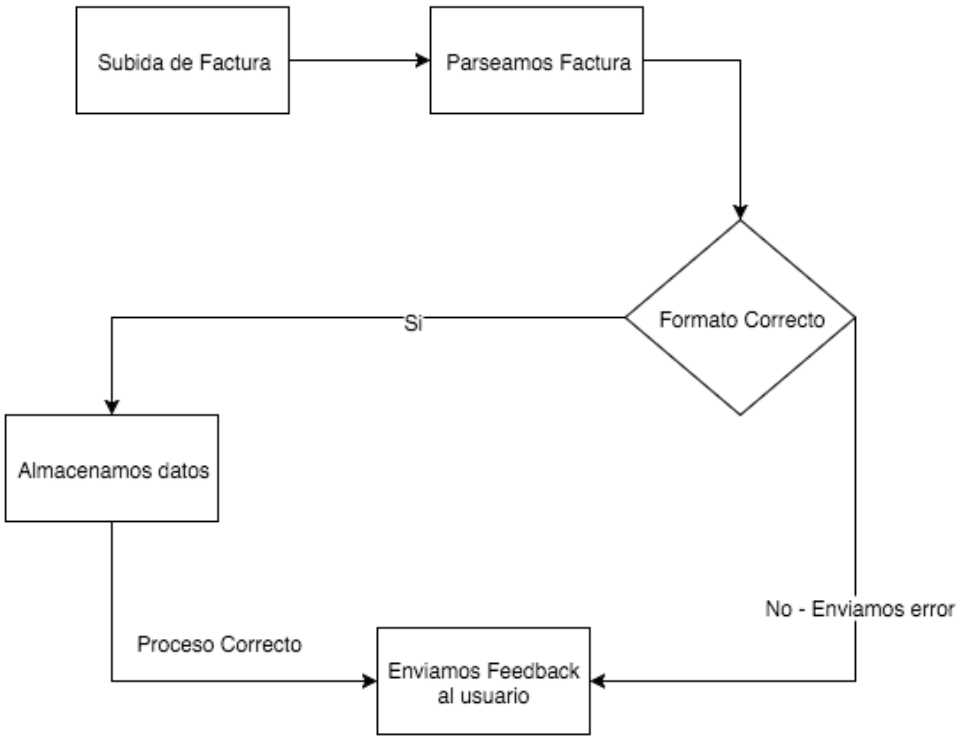

Imagen 6: Diagrama de flujo - Subida de la factura

La Generación de los reportes

En este paso se ha detallado la creación de los reportes, dichos reportes muestran las facturas que actualmente residen en el Cloud, donde se muestran el numero de la factura, la fecha en la que se subió, el total de factura y el enlace de la descarga de la siguiente funcionalidad. En este paso también el requisito de que el usuario debe tener una sesión valida, para ello se empleo la misma herramienta que en la anterior funcionalidad, los decoradores. Los pasos para la creación de los reportes son los siguientes:

- Si el usuario tiene facturas, las recuperamos de la base de datos y generamos una vista con los resultados de la base de datos. El mapeo del ORM permite recuperar los valores mediante clases.
- Si el usuario no tiene facturas, se muestra la web con la acción para subir una nueva factura.

- Si el usuario no esta registrado no permitirá el acceso mostrando que se requiere el acceso.

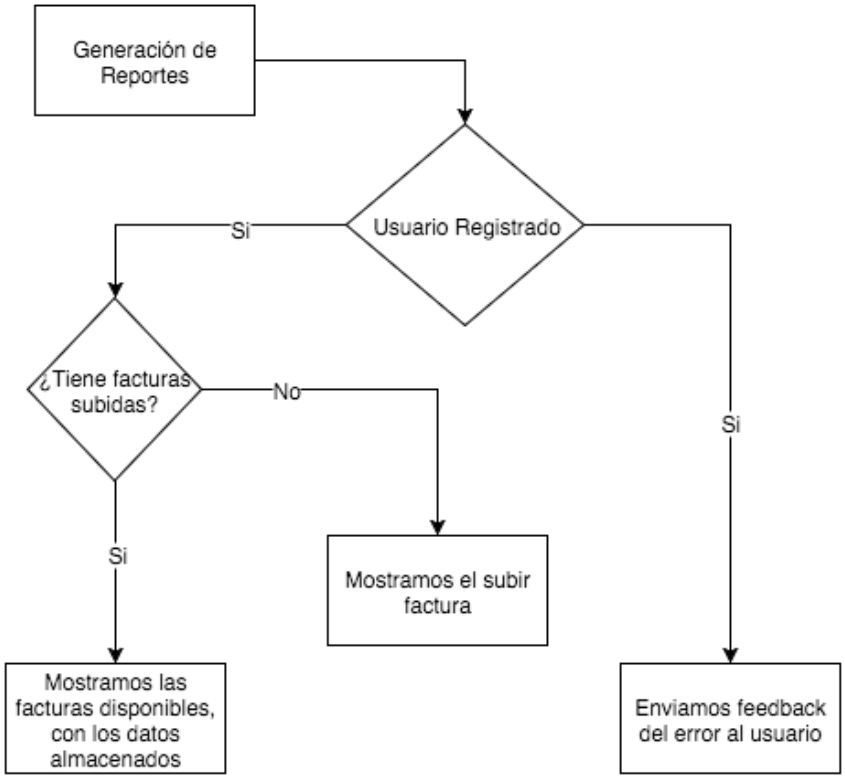

Imagen 7: Diagrama de Flujo - Generar Reportes

Creación de la descarga de las facturas

Para la descarga las facturas almacenadas, se requiere que el usuario este registrado, eso se solvento empleando los decoradores. La factura se debe descargar en formato XML, para ello teníamos dos posibilidades:

- Construir la factura empleando el modelo de datos almacenados.
- Descodificar el campo donde guardamos la factura en formato base64.

Se plantearon las dos opciones por el uso de la base de datos. Pero ya que CloudSQL tiene un tamaño variable sin afectar el rendimiento se empleo la descodificación de la factura.

Los pasos para la emisión de la factura son los siguientes(Imagen 8):

- Inicialmente miramos existe el identificador de la base de datos.

- Si existe el identificador, comprobara si el usuario de la factura es el mismo que el de la petición, si es así enviamos la factura decodificada en base64.
- Si el usuario no es el al de la factura bloqueará la petición enviando el feedback al usuario.
- El el identificador no existe enviamos el error al usuario que ha empleado la acción.

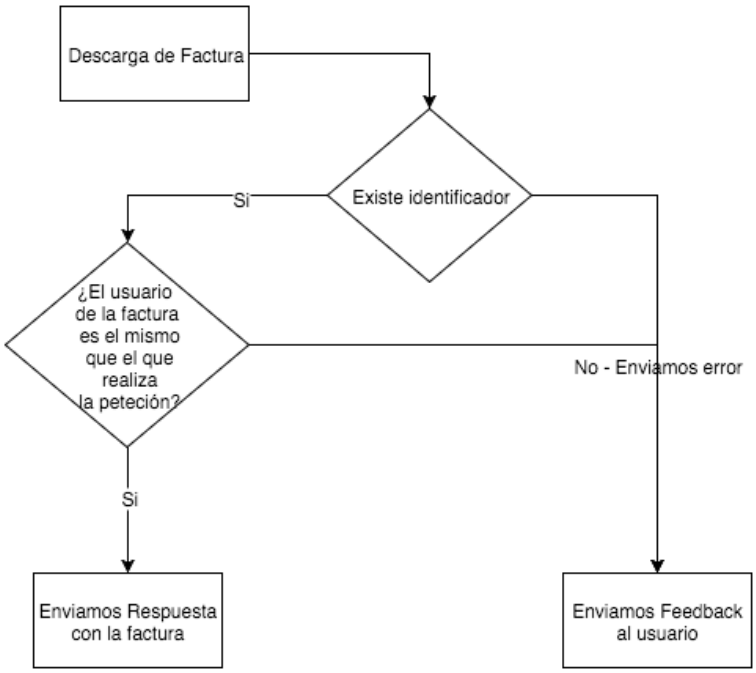

Imagen 8: Diagrama de Flujo - Descarga la Factura

La Creación de los Datos de Usuario

En la parte de los datos de usuario se mostrarán los datos del registro, como el nombre de usuario o el email asociado a ello y si esta activada la cuenta(Imagen 9). Para ello se emplearon los declaradores de Python para la verificación de los datos. Si el usuario tiene una sesión valida se envía al usuario una pagina web con los valores del usuario, para que Django los mapee.

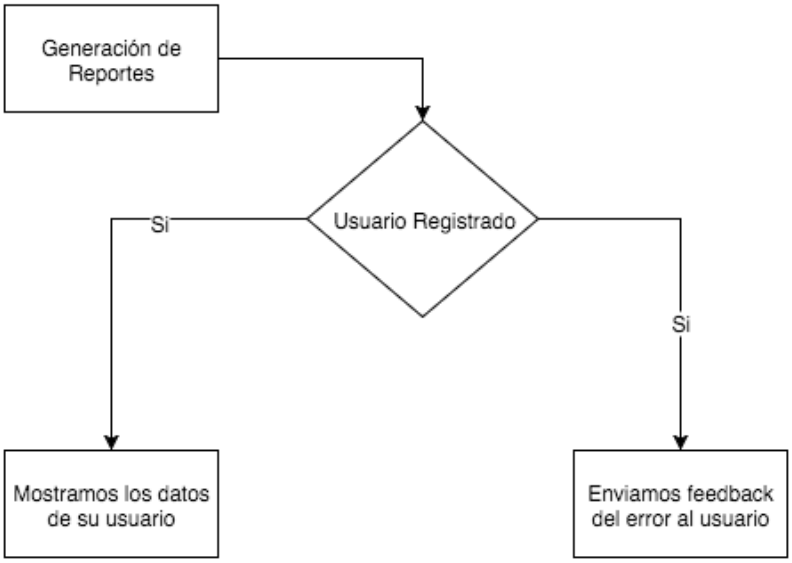

Imagen 9: Diagrama de Flujo de los datos del usuario

Tercera Fase. Creación de la aplicación Universal

El cometido de la aplicación universal es realizar las mismas acciones que permite la web, permitiendo las acciones de la segunda fase. Para ello se ha empleado PhoneGap creando una aplicación que muestre la dirección del Cloud, esto permite a la aplicación realizar las tareas sin que el usuario tenga que ir a la dirección donde esta alojado el sistema y permitir navegar por el dispositivo como lo hace cualquier aplicación.

Para que la aplicación se viera correctamente en todo tipo de pantallas se empleo un diseño adaptativo que se basa en un framework para frontend que permite hacer una pagina web adaptativa. El framework Bootstrap [28] son un conjunto de herramientas para el diseño de sitios y aplicaciones web, conteniendo plantillas, botones, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como una extensión Javascript para aportar funcionalidad adicionales. Este framework es compatible con la mayoría de los navegadores.

El framework Bootstrap en el proyecto

Para la instalación del framework de Bootstrap se descargo la ultima versión ofrecida en su pagina web, y se añadió a la carpeta estática para que Google App Engine la sirviera. El siguiente paso fue instanciar la librería para ello se creo un template llamado “base.html” que contiene la estructura HTML del proyecto y mediante los tags específicos que proporciona Django y la herencia, todos los templates de las dependen de el compartiendo las funcionalidades cargadas. A partir de ese momento se estructuro la pagina en 3 bloques principales

- El titulo que varia en cada proceso definido, mostrando un detalle.
- La barra de navegación: Dicha barra muestra diferentes valores dependiendo si el usuario esta registrado.
- El contenedor : Dicho elemento contiene los campos de cada funcionalidad y los datos a mostrar.

La aplicación de PhoneGap

Primeramente se ha instalado la ultima versión de PhoneGap, dicha framework requiere de Node.JS [29], que es un conocido servidor de propósito general creado empleando el motor de Google Chrome (V8) y que emplea Javascript como lenguaje predeterminado. La aplicación en PhoneGap es muy sencilla ya que PhoneGap se basa en HTML, CSS y Javascript, y no se requiere ninguna funcionalidad que no se capaz de realizar una web la aplicación lo único que debe contener es un webview que contenga la dirección del proyecto, reduciendo a una línea de código.

Resultados

En este apartado se comentan los resultados obtenidos contrastándolos con los objetivos establecidos, comentándolos uno a uno .

Garantizar la seguridad y integridad de los datos en la obtención almacenamiento y gestión y mantener la integridad.

Este objetivo ha podido cumplirse de una forma correcta y tolerante a fallos, empleando el Cloud SQL de la plataforma de Google. Donde se ha configurado un servidor de respaldo, las copias de seguridad automáticas permitiendo una disponibilidad del 99,5%. Cabe comentar que la instancia de Cloud SQL es de pago donde el precio al mes varia según su uso, si se añade el servidor de respaldo se duplica el precio ya que Google consume dos veces por la misma instancia.

La creación de Cloud seguro donde se puedan realizar peticiones

Este objetivo se ha llevado a cabo empleando un framework empleado por multitud de empresas, para la realización de webs complejas que incluye multitud de herramientas para la protección contra ataques y a Google App Engine que se ha activado el modulo SSL con un certificado aportado por Google para las aplicaciones alojadas y configurando el sistema para aceptar SSL 3.0 o TLS 1.2. También se han adoptado las herramientas de seguridad aportadas por Django para el empleo del SSL(Empleo SSL / HTTPS)

Implementar una subida de información por parte del usuario.

En el Cloud se han creado los métodos para permitir la subida de información de los usuarios registrados, dicho objetivo he requerido añadir las funcionalidades de los usuarios.

Creación de una aplicación que permita emplearlo.

Este objetivo se ha llevado a cabo creando mediante el empleo de un framework web que crea ventanas adaptables a todo tipo de dispositivo, para después crear una aplicación universal empleando el framework PhoneGap.

Realizar un estudio de los sistemas criptográficos para dicho ámbito y evaluar cual se adapta mejor a los objetivos anteriores.

Se han empleado los algoritmos criptográficos seguros para la creación del Cloud como la protección del salt añadida para contrarrestar los ataques de las tablas arcoíris por si en algún momento tuviéramos una fuga de información de la base de datos o el uso de todas las protecciones que posee Django para evitar posibles ataque sobre la web. Como la optimización para los ataques de DOS que nos permite una mayor disponibilidad del servidor. El empleo de SHA-256 para la generación de contraseñas o el empleo de SSL o TLS para la seguridad de Google App Engine.

Una vez comentados los objetivos primarios, se han realizado uno de los dos objetivos secundarios como la creación de una web, ya que se ha desarrollado completamente en una web convirtiendo dicho objetivo en primario.

El resultado del trabajo es un Cloud funcional que permite la subida de facturas electrónicas y el alojamiento de ellas sin limite, para poderse descargar en cualquier dispositivo en el cual nos registremos. Realiza un mapeo de las facturas subidas para mostrar información especifica al usuario como el precio de la factura o el numero de factura electrónica. El sistema cumple las características de alta disponibilidad, rendimiento optimo alrededor del mundo y seguridad.

Futuras Mejoras

Al ser un proyecto tan amplio y con muchas funcionalidades las mejoras son muchas y muy variadas, a continuación se da un listado de varias de ellas:

- Al ser un proyecto de ámbito mundial únicamente se están tomando en consideración un modelo de factura electrónica. Habiendo un tipo por cada país, seria interesante la implementación de la mayoría de los tipos.
- Permitir implementar un método para que las entidades gubernamentales subieran los diseño y actualizaciones de sus facturas electrónicas en formato XSD con lo que los usuarios puedan subir facturas en dichos diseños y evitaría el tener que implementar los modelos relacionales para cada una de las facturas.
- Rehacer el diseño de la interfaz visual por parte del cliente para que sea más amigable y entendible.
- Realizar la aplicación Android e iOS sin ser universales, para una mejora en rendimiento de las aplicaciones y el empleo de funcionalidades avanzadas como la subida automática de las facturas obtenidas por diferentes tecnologías.
- La posibilidad de subir facturas a los emisores de las mismas y que al usuario o entidad a la que vaya correspondida se le emita un mensaje de que se ha añadido una nueva factura a su cuenta del Cloud, permitiendo gestionar todas las facturas alojadas en un mismo sitio.
- La creación de una API para que terceras partes se integren, haciendo revalorizar el Cloud, ya que otros programas externos permitieran subir facturas electrónicas.
- Añadir un método Premium para entidades o empresas donde se envíen juegos de reportes de las facturas subidas durante un tiempo.

- La creación del modelo de datos mediante Big Data para la optimización cuando el flujo de usuarios se elevado, ya que el servidor MySQL puede quedarse pequeño dependiendo el numero de usuario del sistema.
- Realizar un versión del sistema para empresas para la instalación en sus propios servidores con respaldo en el Cloud principal, ya que hay empresas que desean tener los datos en sus propios servidores

Conclusión

Durante este proyecto se ha podido demostrar que se requiere una tecnología como esta para un futuro ya que el ahorro económico es considerablemente alto. Se han cumplido casi todos los objetivos haciendo esta plataforma una herramienta bastante compleja y planteándolo como un proyecto con un equipo y financiación puede ser una herramienta innovadora en el sector por la inexistencia.

También cabe destacar que para mi ha supuesto una gran ayuda para adquirir nuevos conceptos y consolidando los aprendidos en el master en un único proyecto al igual que poder disfrutar creando una aplicación de tal magnitud que permita estar disponible en todo el mundo.

Bibliografía

[1] Ericson, «Ericsson Mobility Report: 70 percent of world's population using smartphones by 2020,» 03 06 2015. [En línea]. Available: http://www.ericsson.com/news/1925907. [Último acceso: 03 06 2015].

[2] La Sociedad de la Información en España , «ABC,» 2014. [En línea]. Available: http://www.abc.es/gestordocumental/uploads/internacional/siE2014.pdf. [Último acceso: 02 01 2015].

[3] BOE, «Ley 25/2013, de 27 de diciembre, de impulso de la factura electrónica y creación del registro contable de facturas en el Sector Público.,» 28 12 2013. [En línea]. Available: https://www.boe.es/boe/dias/2013/12/28/pdfs/BOE-A-2013-13722.pdf.

[4] Google, «Google Cloud Platform,» Google Inc, 01 01 2016. [En línea]. Available: https://cloud.google.com/. [Último acceso: 03 01 2016].

[5] Google, «Google App Engine,» 01 01 2015. [En línea]. Available: https://cloud.google.com/appengine/. [Último acceso: 02 01 2015].

[6] Amazon, «About Amazon Web Services,» Amazon Inc, 01 01 2006. [En línea]. Available: http://aws.amazon.com/es/about-aws/. [Último acceso: 02 01 2016].

[7] Microsoft, «Microsft Azure,» Microsoft, 01 01 2016. [En línea]. Available: https://azure.microsoft.com/es-es/. [Último acceso: 02 01 2016].

[8] Google, «Android,» Google Inc, 01 01 2016. [En línea]. Available: https://www.android.com/. [Último acceso: 02 01 2016].

[9] Apple, «iOS,» Apple Inc, 01 01 2016. [En línea]. Available: http://www.apple.com/es/ios/. [Último acceso: 02 01 2016].

[10] PhoneGap, «PhoneGap,» [En línea]. Available: http://phonegap.com/.

[11] The Apache Software Foundation, «Apache Cordova,» 2012. [En línea]. Available: https://cordova.apache.org/. [Último acceso: 2015].

[12] Object Management Group, Inc, «Unified Modeling Language™ (UML®),» Object Management Group, Inc, 1997. [En línea]. Available: http://www.uml.org/. [Último acceso: 2016].

[13] GitHub Inc., «GitHub - Guides,» GitHub Inc., 2008. [En línea]. Available: https://guides.github.com/activities/hello-world/. [Último acceso: 2016].

[14] Django Software Foundation, «Django,» Django Software Foundation, 2005. [En línea]. Available: https://www.djangoproject.com/. [Último acceso: 2016].

[15] J. Likness, «SOLID and DRY,» 3 Enero 2009. [En línea]. Available: http://www.codeproject.com/Articles/36712/SOLID-and-DRY. [Último acceso: 1 Enero 2016].

[16] B. Kumar, «Django - Python MVC Framework,» Citrix Systems, Inc., 25 Octubre 2013. [En línea]. Available: http://es.slideshare.net/balakumarp/django-framework. [Último acceso: 02 Enero 2016].

[17] J. K.-M. Adrian Holovaty, «El libro de Django 1.0,» LibrosWeb , 2006. [En línea]. Available: https://librosweb.es/libro/django_1_0/. [Último acceso: 2016].

[18] Apuntes de Programación, «Seguridad extra para la web: Una guía al nonce,» 2011. [En línea]. Available: http://programacion.jias.es/2012/04/seguridad-extra-para-la-web-una-guia-al-nonce/. [Último acceso: 2015].

[19] J. Hodges, Paypal, C.Jackson, Carnegie Mellon University, A.Barth, Google Inc,, «HTTP Strict Transport Security (HSTS),» 01 November 2012. [En línea]. Available: https://tools.ietf.org/html/rfc6797. [Último acceso: 01 01 2016].

[20] Django Software Foundation, «User authentication in Django,» 2005. [En línea]. Available: https://docs.djangoproject.com/en/1.5/topics/auth/. [Último acceso: 2016].

[21] RainbowCrack Project, «List of Rainbow Tables,» 2003. [En línea]. Available: http://project-rainbowcrack.com/table.htm. [Último acceso: 2016].

[22] Sphinx , «WSGI,» Sphinx Inc, [En línea]. Available: http://wsgi.readthedocs.org/en/latest/.

[23] Gobierno de España, «¿Qué es la factura electrónica ?,» Gobierno de España, [En línea]. Available: http://www.facturae.gob.es/factura-electronica/Paginas/factura-electronica.aspx.

[24] Gobierno de España, «Formato de Factura,» [En línea]. Available: http://www.facturae.gob.es/formato/Documents/EspanolFacturae3_0.pdf.

[25] L. Quin, «Extensible Markup Language (XML),» W3C, 19 05 2015. [En línea]. Available: https://www.w3.org/XML/. [Último acceso: 2016].

[26] T. Preston-Werner, «Semantic Versioning 2.0.0,» [En línea]. Available: http://semver.org/.

[27] I. S. L. A. S. A. L. M. E. L. M. S. M. A. S. V. w. I. David Fallside, «W3C XML Schema Definition Language (XSD),» W3C, 5 April 2012. [En línea]. Available: https://www.w3.org/TR/xmlschema11-1/. [Último acceso: 2016].

[28] Boostrap Inc, «Getting Started,» Boostrap, 2013. [En línea]. Available: http://getbootstrap.com/getting-started/. [Último acceso: 2015].

[29] Node.js Foundation, «About Node.js®,» Node.js Foundation, 2015. [En línea]. Available: https://nodejs.org/en/about/. [Último acceso: 2016].