UNIVERSIDAD DE CANTABRIA Facultad de Ciencias

Proyecto Fin de Carrera

Herramienta de generación de reportes para la plataforma "Google Apps for Business"

(Google Apps for Business Reporting Tool)

Para acceder al Título de

INGENIERO EN INFORMÁTICA

Autor: Diego Gandarillas Pérez
Director: Mario Aldea Rivas
Codirector: Urbano Llamas del Agua

Junio – 2014

Agradecimientos

Llegados a este punto de mi vida, creo necesario dar las gracias a un grupo de personas sin las cuales posiblemente este proyecto no podría haberse llevado a cabo.

En primer lugar a mi familia, sobre todo a mis padres y a mi hermano, ya que me han aportado las energías y bienes económicos necesarios para entrar a formar parte de una carrera universitaria y sin ellos no hubiera sido posible.

También quiero hacer una mención importante a Coral, mi pareja, por su apoyo y su ánimo en los momentos de flaqueza y negándome la posibilidad de rendirme.

No quiero olvidar a los informáicos de Solvay Química, en especial Urbano, Jacinto, Fernando y David, que me han dado la posibilidad de saber lo que es trabajar a su lado, además de ofrecerme este proyecto y ayudarme durante todo el desarrollo.

A Mario por aceptar mi petición de ser el jefe de proyecto y por todo el soporte que me ha dado durante estos meses.

A todos mis compañeros de todos estos años y a todos los profesores que nos ayudan día a día a ser más sabios y mejores personas.

Sinceramente y de todo corazón

Muchas gracias a todos.

Resumen

La multinacional Solvay S.A. ha empezado a moverse a la nube. Un cambio en la filosofía de la empresa que se ha visto reflejado también en la informática es la evolución de los sistemas tradicionales de hosting dedicados a SaaS (software como servicios).

Uno de esos cambios, es la aparición de la tecnología "Google Apps for business" en el mapa de sistemas de la información de la empresa. Este cambio reemplaza a Microsoft como sistema de gestión documental, correo, contenidos de gestión en web, etc…

Debido a que es una tecnología reciente, tiene carencias que hay que suplir con desarrollos, personalizaciones y procesos.

El proyecto forma parte del proceso de control de los sistemas de gestión documental. La problemática aparece cuando los usuarios son capaces de gestionar sus propios documentos y compartirlos con entidades externas. Para evitar pérdidas y filtraciones de información sensible, es necesario añadir una capa, que permita un mejor control a los propios usuarios.

La tarea consiste en la realización de un programa de reporte de archivos, que genere en el servicio Google Drive, una hoja de cálculo con la lista de ficheros que encajen dentro de las características de la opción que el usuario escoja, así como varios datos internos de los mismos.

Además es necesaria la creación de una interfaz para permitir la selección del reporte y ofrecer la posibilidad de introducir ciertos datos adicionales para algunos de ellos.

Palabras clave:

App Engine, Cliente web, Solvay, reportes, Java, Servlet, Jsp, Google Drive, Google Spreadsheet, html, cola de tareas.

Summary

The multinational Solvay S.A. has started moving to the Cloud. A change of the philosophy of the company that has also been reflected in computer science is the evolution of the traditional hosting system dedicated to SaaS (Software as a Service).

One of these changes is the appearance of the technology “Google Apps for Business” in the system map of the company’s information. This event replaces Microsoft as system documental management, mail, web content, etc.

Due to the fact it is a recent technology, it has faults which need to be replaced with developments, customizations and processes.

The project is a part of the documental management control system. The problem appears when the users are able to handle their own documents and share it with external entities. To avoid losses and filtrations of sensitive information is necessary to add a layer that permits a better control to the users.

The task consists on making a files reporting program which generate, on Google Drive service, a spreadsheet with the list of the files that fit inside the characteristics of the option that the user chooses, as well as several information of the same ones.

It is necessary to create an interface to permit the report’s selection and offer the possibility of introduce additional data for some of they too.

Key Words:

App Engine, web client, Solvay, reports, Java, Servlet, Jsp, Google Drive, Google Spreadsheet, html, task queue.

Índice de contenidos

Índice de imágenes

1. Introducción y objetivos

1.1. Presentación del problema

G.A.F.B (Google Apps for Business) es un conjunto de aplicaciones de colaboración incluyendo Gmail, Drive, Grupos, calendarios, etc. A pesar de su facilidad de uso, la capacidad de reporting es muy limitada, siendo algunos de estos reportes muy importantes tanto para la gestión de la plataforma como para los propios usuarios finales.

Se pretende realizar el análisis, diseño e implementación de una aplicación de reporting y scans basada en Google Apps junto a una interfaz web que de la capacidad de selección al usuario y que además ofrezca la posibilidad introducir parámetros adicionales en alguno de estos reportes.

La aplicación estará orientada totalmente a servicios de Google dentro del dominio de “solvay.com”, es decir, para usuarios que tengan una cuenta de correo electrónico contratada en la empresa, aunque podrá ser ejecutada por cualquier persona que tenga una cuenta de correo electrónico ligada a Google.

1.2. Objetivos

Google Apps es una herramienta muy flexible a la hora de compartir la información. Usuarios finales pueden compartir información con el exterior, sin embargo, Google no tiene ninguna granularidad al respecto, ya que por ejemplo, no existen white/black lists a la hora de definir con qué dominios se puede compartir información.

Muchos de los informes que se propondrán en esta herramienta van encaminados a ayudar a los propietarios de la información (usuarios finales) a gestionar esta información.

Además, se estudiará la posibilidad de realizar un segundo bloque que irán encaminados a los administradores del dominio, aunque en este segundo grupo ya existen varias herramientas en el mercado, es necesario que la empresa posea una herramienta de este estilo evitando el coste económico que supone, además de la mayor libertad funcional de una aplicación propia.

Los objetivos específicos iniciales serán:

- Desarrollo de la aplicación de reporting para diferentes servicios de Google realizando la codificación de todas las clases, atributos y métodos necesarios para la gestión de la misma.

- Elaboración de la interfaz web donde el usuario pueda conectarse, establecer los permisos necesarios, escoger el reporte e introducir los parámetros necesarios en los que lo requieran.

2. Análisis de Requisitos

2.1. Planing inicial

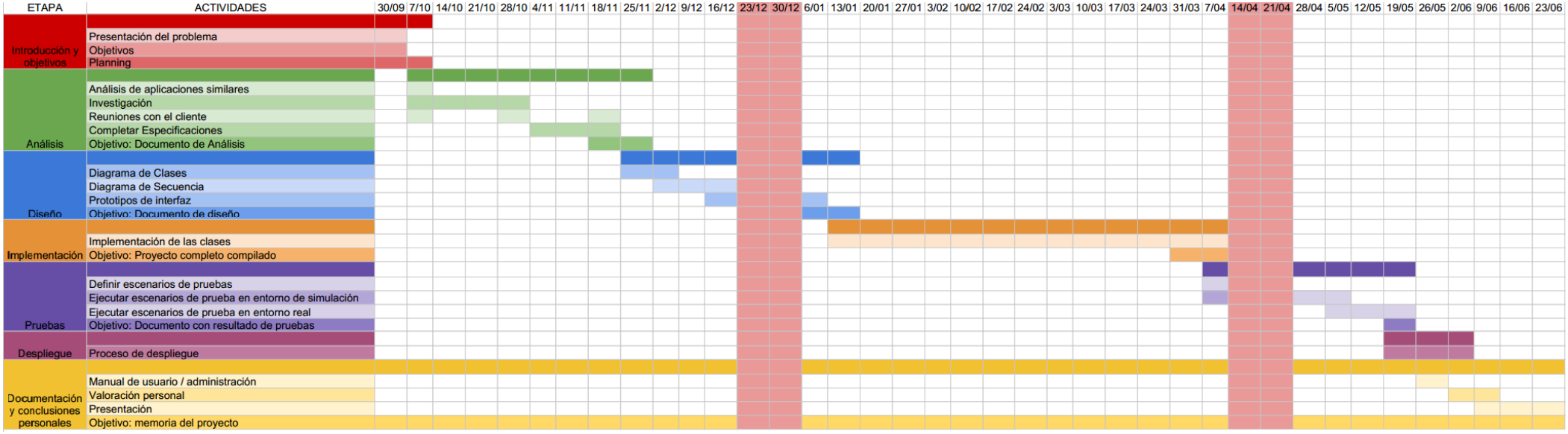

Image 1 – Planing Inicial

2.2. Análisis de aplicaciones similares

General Audit Tools for Google Apps (GAT)

Es la aplicación existente con más similitudes a este proyecto. Solo permite ser instalada en un dominio y ser ejecutada por el administrador. Se basa en una sencilla interfaz html que redirige a distintas urls. Era una herramienta totalmente gratuita pero actualmente algunas de sus funcionalidades requieren realizar un pago después de pasar un período de varios días de prueba.

En la pantalla principal se muestra un resumen de los datos principales del usuario (dominio, si es administrador, número de usuarios asociados,...) además contiene varios botones que redirigen a opciones, hay dos de ellos reseñables, uno que muestra los reportes realizados y otro las estadísticas.

Dentro del apartado de reportes se puede observar una tabla con la hora de inicio y fín, el estado y la duración de cada uno de ellos, así como la cantidad de usuarios, documentos y grupos que poseía el usuario cuando el reporte se realizó. Esta página además posee diferentes filtros de ordenación, además ofrece las opciones de visualizar el contenido de los reportes de manera online a través de un Google spreadsheet o descargarlos en formato csv.

En la propia página de estadísticas se encuentran, en versión html, una gran cantidad de datos numéricos actuales sobre los diferentes servicios, por ejemplo, en referencia a Drive se muestra la cantidad máxima de usuarios que comparten un mismo documento junto a pequeños gráficos que muestran dichos valores en un intervalo de tiempo.

Jasper Reports

Librería Java desarrollada por la empresa JasperSofts que realiza reportes embebidos para una web u otro tipo de aplicación Java. El objetivo es proporcionar al usuario una serie de informes con diferentes formatos que están orientados a una visualización sencilla o a la impresión.

Crystal Reports

Aplicación desarrollada por SAP cuyo objetivo es el diseño y generación de informes a partir de una base de datos. Permite al usuario escoger las filas o columnas que crea necesarias y automáticamente es actualizado cuando lo hace la fuente de la que proviene.

2.3. Material y métodos

2.3.1. Google App Engine

Google App Engine es un sistema de software creado en la primavera del año 2008 y basado en la tecnología "cloud computing" (desarrollo en la nube), posee un modelo de servicio del tipo PaaS (Platform as a service) que ofrece al usuario una plataforma de desarrollo de aplicaciones además de la capacidad de almacenamiento para las mismas.

Aporta a los desarrolladores que lo utilizan una mayor velocidad y cantidad de peticiones concurrentes, debido al uso del motor Google, que al ejecutar las aplicaciones en entorno local. Permite principalmente dos lenguajes de programación: Java y Python pero cada vez ofrece más servicios para otros lenguajes como GO (lenguaje de Google) o PHP. Para el lenguaje Java soporta además diferentes "frameworks" como son los "java-servlets" de los cuales constara mayoritariamente el proyecto y se entrará en detalle más adelante en esta memoria.

A través de un panel de administración disponible en la web *https://appengine.google.com/*, el usuario puede controlar crear, modificar o eliminar sus aplicaciones hasta un límite gratuito de 10, se mostraran también las aplicaciones activas, los datos de las mismas y el número de peticiones por unidad de tiempo.

Proceso de conexión a una aplicación App Engine basada en cliente web

Cuando el cliente accede a una aplicación alojada en el entorno App Engine y que precise del uso de APIs, los primeros pasos que debe realizar son la autenficiación y concesión de los permisos pertinentes, este proceso recibe el nombre de protocolo Oauth2.0. Una vez realizado podrá acceder al proyecto

de APIs de la aplicación, que se encargará de hacer las llamadas necesarias a las clases y métodos de los servicios de Google.

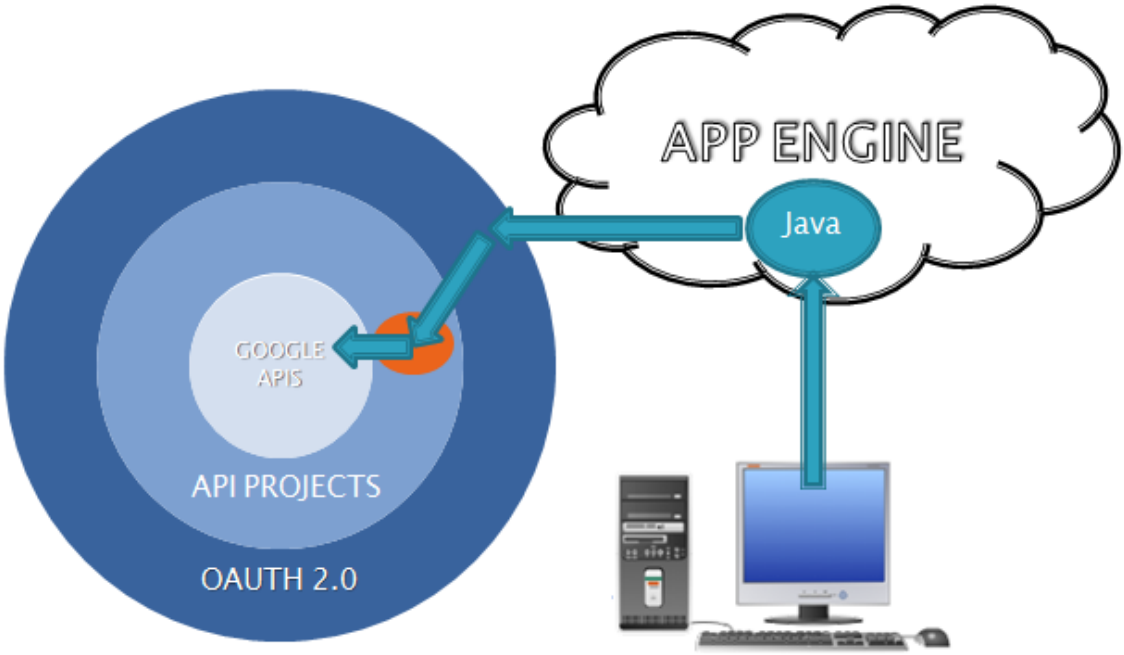

Image 2 – Proceso de conexión

2.3.2. OAUTH 2.0

Es el tipo de protocolo necesario para poder trabajar con las aplicaciones en el entorno Google App Engine controlando las peticiones por usuario. Existen cinco diferentes en función del tipo de aplicación:

- Aplicaciones basadas en servidor web
- Aplicaciones instaladas
- Aplicaciones del lado del cliente
- Aplicaciones basadas en dispositivos de entradas limitadas
- Aplicaciones basadas en cuentas de servicio.

El proyecto actual pertenece al tipo basado en servidor web y para poder acceder es necesario una serie de pasos mostrados en la siguiente figura:

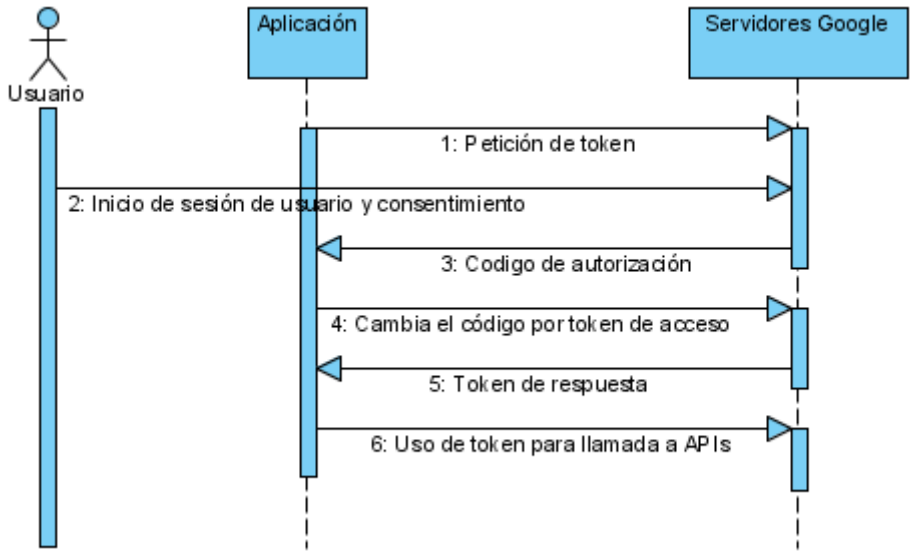

Image 3 - Pasos requeridos del protocolo Oauth 2.0 para las aplicaciones basadas en servidor web

Para comenzar este proceso, durante la ejecución de la aplicación se formará una dirección web para pedir la autorización, a cuyo comienzo (*https://accounts.google.com/o/oauth2/auth*), se le añade una serie de parámetros, los más importantes son:

- Client_id: la id del cliente que ejecuta la aplicación
- Redirect_uri: el identificador uniforme de recursos (URI) al cual .la aplicación redigirá al usuario una vez realizada la autentificación.
- Scope: los permisos de los servicios de Google que el usuario tendrá que aceptar.
- Acces_type: indica si es necesario que el usuario esté presente en el navegador (online) o no (offline). Por defecto es online.
- Approval_promp: puede ser “force” si es necesario pedir permisos al usuario cada vez que se ejecute la aplicación o “auto” si solo es necesario la primera vez. Por defecto es “auto”.

Durante el paso anterior si el usuario no ha iniciado sesión en una cuenta perteneciente a algún dominio de Google, será redirigido a la página de login. Una vez que se conecte si es la primera vez que ejecuta la aplicación teniendo un “approval_promp” como “auto”, o siempre en el caso de ser “forcé” aparecerá una pantalla para que acepte los scopes.

Los servidores de Google retornarán la dirección web “*http://oauth2-login-demo.appspot.com*” incluyendo un parámetro denominado “code” que es diferente en función de si la petición es aceptada o no, en el primer caso su valor será el código de autorización y en el segundo la palabra “error”.

Posteriormente con ese código, el id del cliente, la URI de redirección y un nuevo parámetro llamado “client_secret” que es el código secreto del cliente, se lo envía a los servidores para intercambiarlo por un token.

Ese token de acceso enviado por los servidores contendrá como parámetros principales:

- Acces_token: token de acceso para las APIs
- Refresh_token: token necesario para obtener otro nuevo, solo está presente cuando el tipo de acceso es offline.
- Expires_in: la duración temporal durante la cual este token será válido ya que por seguridad no lo pueden ser permanentemente.

El administrador de la aplicación podrá revocar el acceso a cualquiera de los usuarios de la aplicación en cualquier momento si lo ve necesario.

2.3.3. Proyecto de APIs

Para poder utilizar las diferentes APIs de google App Engine es necesario primero tener creado un proyecto de APIs.

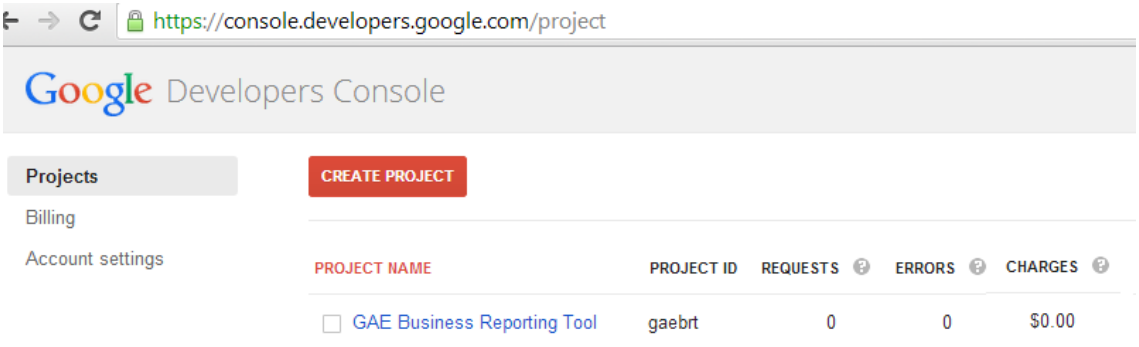

Image 4 - Consola de proyectos de API de Google

Un usuario de la aplicación que entre en la dirección web “*https://console.developers.google.com/project*”, podrá acceder a los diferentes procesos que tenga creados, así como ver su id, el número de peticiones y de errores de la aplicación en las últimas 24 horas y si ha existido algún cargo por sobrepasar el límite de peticiones gratuito de alguno de los servicios.

Cada uno de los proyectos deberá tener habilitadas las APIs a las que va a necesitar acceder, esto se puede realizar a través de la sección APIs & oauth en la pestaña de APIs. Para este proyecto de toda la lista de ellas solo son

necesarias las dos de Drive, como se puede comprobar la primera de ella te muestra además el porcentaje de peticiones realizadas.

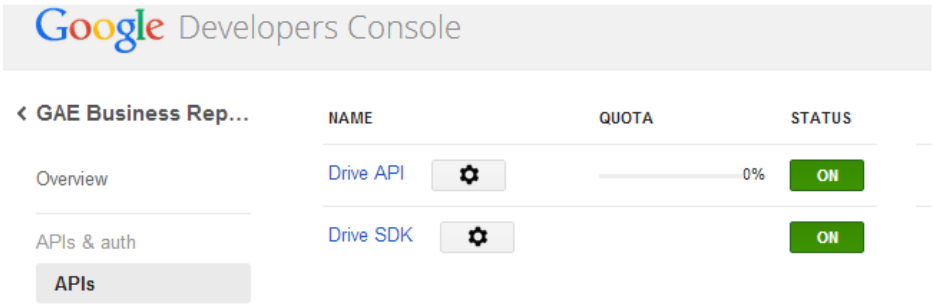

Image 5 - Lista de Apis de un proyecto

Dentro de la misma sección, la pestaña "Credentials" contiene las credenciales necesarias para poder ejecutar el proyecto, serán diferentes en función del tipo de aplicación a la que pertenezca. Por ejemplo en una aplicación del tipo servidor web destacarán los campos "Client id", "Client secret", "Redirect Uris" e "Email address".

Estos cuatro campos son necesarios para llevar a cabo el control de accesos de usuarios y la cantidad de peticiones:

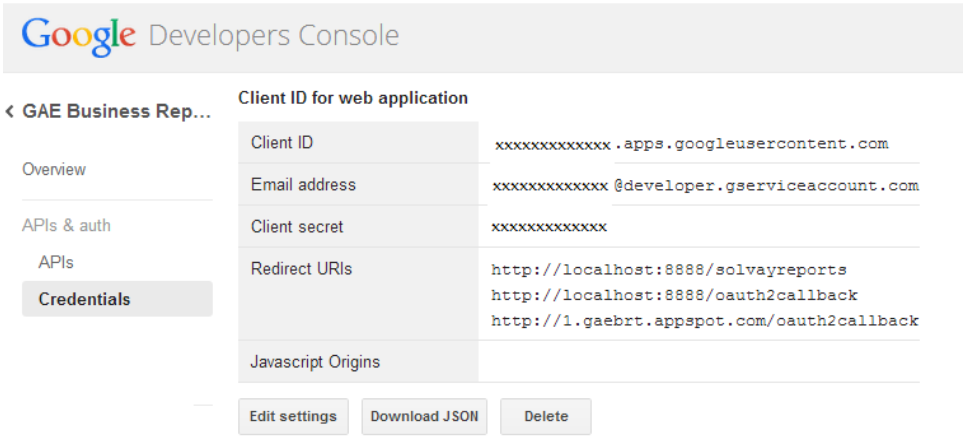

Image 6 – Credenciales de la aplicación

Por último, es necesario dentro de la siguiente subsección "Consert Screen" introducir el eMail del administrador de la aplicación y el nombre de la misma.

2.3.4. API de Google Drive

Dispone al programador de una gran cantidad de clases y subclases relacionadas con la aplicación Google Drive por ejemplo ofrece la posibilidad

de crear, descargar y trabajar tanto con los ficheros como con los directorios. El usuario podrá acceder a una gran cantidad información de los mismos (titulo, descripción, fecha de ultimo acceso,…) así como modificarla, incluso podrá trasladar un archivo de un directorio a otro.

Las principales subclases que se van a utilizar en la aplicación son:

- Files: para trabajar directamente con los ficheros, sus principales métodos:
    - Copy (POST): copia un fichero en otro
    - Delete (DELETE): elimina un fichero
    - Get (GET): obtiene un fichero
    - Insert (POST): inserta un nuevo fichero en Google Drive
    - List (GET): obtiene los ficheros de un usuario, permite además a través de un texto llamado query establecer los siguientes filtros:
        - Title: si el titulo contiene o es igual a un texto.
        - Fulltext: si el fichero contiene el texto indicado.
        - Mimetype: si el fichero es del tipo especificado.
        - ModifiedDate: si el fichero ha sido modificado en una fecha determinada.
        - LastViewedBymeDate: si el fichero ha sido abierto por el usuario en una fecha determinada.
        - Trashed: si el fichero está en la papelera o no.
        - Starred: si el fichero esta marcado como importante.
        - Parents: si las carpetas pasadas como parámetro contienen al fichero.
        - Owners: retorna los ficheros de ese propietario.
        - Writers: retorna los ficheros cuyos escritores son los pasados como parámetro.
        - Readers: retorna los ficheros cuyos lectores son los pasados como parámetro.
        - SharedWithMe: obtiene los ficheros que están compartidos con el usuario.

- Parents: para trabajar con las carpetas contenedoras de los ficheros, sus principales métodos son:
    - Delete (DELETE): elimina un directorio de un fichero.
    - Get (GET): obtiene un determinado directorio padre de un archivo.
    - Insert (POST): agrega a un fichero un nuevo directorio.

    - List (GET): obtiene una lista con todos los directorios padres de un determinado archivo.

- Permission: para trabajar con los usuarios que tienen acceso a un fichero, tiene como métodos principales:
    - Delete (DELETE): elimina los permisos que tiene un determinado usuario.
    - Get (GET): obtiene un determinado permiso de un fichero.
    - Insert (POST): agrega un nuevo permiso de usuario.
    - List (GET): obtiene todos los permisos de un fichero.
    - GetIdForEmail (GET): obtiene la id de un permiso a partir de un determinado eMail.
    - Update (PUT): modifica un permiso.

Cada uno de los métodos anteriores pertenece a un tipo de petición incluida en el protocolo http, principalmente serán utilizadas cuatro de ellas:

- GET: obtiene la información de un determinado recurso.
- DELETE: elimina un recurso.
- POST: envía datos para que sean procesados por un recurso
- PUT: agrega contenido o modifica un recurso específico.

Cada archivo alojado en esta aplicación posee principalmente tres tipos de rol con diferentes permisos para cada uno para los usuarios que tienen acceso al mismo:

- Owner: usuario que crea y es propietario del archivo, posee la capacidad de modificar o añadir permisos y usuarios, además de leerlo o modificarlo.
- Writers: usuarios escritores del archivo con la opción de modificar o añadir permisos y usuarios (si ha sido habilitado por el propietario), también podrán leerlo o modificarlo.
- Readers: lectores del archivo, el único permiso que poseen es acceder al archivo en modo de lectura.

2.3.5. Java Servlets y JSPs

Los servlet son programas escritos en Java del estilo de las applets, que a diferencia de estas, permiten al cliente interactuar con el servidor (puede ser web o no) en lugar del navegador. El primer paso que realiza el servidor es

iniciar el servlet utilizando un método especial denominado “doGet” cuyos parámetros son la petición del cliente y la respuesta del servidor. Son objetos de las clases HttpRequest y HttpResponse respectivamente pertenecientes a la Api de Java.

El programa interactúa con la petición obteniendo los datos de la sesión o del usuario actual que crea necesarios, y maneja el contenido de la respuesta mostrándolo o almacenándolo, también redirige la aplicación a otra página web, archivo jsp u a otro servlet. Finalmente el servidor una vez concluida la ejecución del código elimina el java servlet.

Los archivos Java server page (JSP) tienen un tipo de tecnología muy parecida a la que usan los servlet, pero a diferencia de ellos lo que hacen es introducir código java utilizando la etiqueta “<% %>” entre las líneas html. A diferencia del código embebido javascript es ejecutado en el servidor antes de mostrar al usuario la página web.

La diferencia principal entre ambos tipos de archivo es que los JSP siempre ejecutaran código java dinámicamente dentro de una página web, mientras que un java servlet producirá una respuesta que no tiene por qué ser un documento html.

2.4. Reuniones con el cliente

Durante toda la elaboración de la aplicación con cierta periodicidad se establecieron varias reuniones con el cliente con el fin de mostrar los avances y evaluar los diferentes prototipos, la mayoría de ellas tienen una estructura similar, siendo relevantes cuatro de ellas:

Primera reunión con el cliente (15/10/13)

Asistentes:

- Urbano Llamas, Fernando López y Jacinto Pelayo en calidad de clientes pertenecientes a la empresa Solvay Química S.A.
- Diego Gandarillas en calidad de gestor de proyecto

Principales temas a tratar:

- Tipo de aplicación y de interfaz

- Servicios de Google de los que realizar reportes
- Formato para los reportes

Resumen:

El cliente quiere una aplicación basada en servidor web integrada en el servicio Google App Engine desde la cual se obtengan los reportes. Se ha decido de este tipo principalmente porque es sencilla de implementar y no es necesario instalar ningún tipo de software de escritorio, ya que en caso contrario sería inviable por ser una empresa con gran cantidad de personal.

Queda acordado que solo para tres de los servicios de Google es necesario realizar los reportes, debido a que son los más utilizados en la empresa. En orden de importancia son: Google Drive, Google Sites y Google Groups.

Los reportes serán almacenados en un Google Spreadsheet dentro de un árbol de carpetas cuyo primer nivel será el título de la aplicación y que contendrá una carpeta por cada servicio. Estos directorios serán creados previamente, si no existe, en la carpeta personal de Google Drive del usuario de la aplicación.

La interfaz web deberá ser sencilla y tener un diseño básico con diferentes páginas html o jsp e incluirá una presentación de la aplicación, un menú con los diferentes servicios y reportes para cada uno de ellos, otra por si es necesario introducir información adicional y por último una página web de finalización, que permita redirigir al usuario de nuevo al menú para poder realizar otro reporte y que mostrará un enlace a la carpeta de Google Drive donde será almacenado.

Conclusiones y tareas a realizar:

- Investigación del entorno Google App Engine, sobre las APIs de los diferentes servicios y lenguajes y herramientas que soporta.
- Investigación sobre el tamaño máximo de los Spreadsheet de Google
- Elaboración del diagrama de casos de uso

Segunda reunión con el cliente (06/11/13)

Asistentes:

- Fernando López y Jacinto Pelayo en calidad de clientes pertenecientes a la empresa Solvay Química S.A.
- Diego Gandarillas en calidad de gestor de proyecto

Principales temas a tratar:

- Reportes a realizar de cada servicio y columnas a mostrar
- Tipos de usuario para los que va dirigido

- Forma de ejecución de los reportes
- Mostrar al cliente el resultado de investigaciones

Resumen:

Se decide que los reportes estarán orientados hacia usuarios finales, es decir que no será necesario para los administradores de dominio y se dividirán en relación a los servicios Google. Serán los descritos a continuación.

Reportes Drive:

- Archivos de los que el usuario es propietario.
- Archivos en los que el usuario es editor y lector.
- Archivos a los que el usuario tiene acceso, compartidos con cuentas fuera del dominio solvay.com.
- Archivos a los que el usuario tiene acceso, compartidos con cuentas de un dominio configurable.
- Archivos compartidos con un usuario configurable que actúa como propietario.
- Archivos compartidos con un usuario configurable.
- Archivos en los que el usuario es propietario o editor y están compartidos con el dominio vía visibilidad.

Reportes Groups:

- Grupos de los que el usuario es miembro.
- Todos los archivos a los que un grupo configurable tiene acceso.

Reportes Sites:

- Sitios de los que el usuario es miembro
- Sitios en los que el usuario es propietario o administrador.

El resultado obtenido de la investigación respecto al tamaño máximo de un Google Spreadsheet es de 400.000 celdas. El cliente, tras comentarle este dato, lo asume como suficiente y no ve necesario buscar algún tipo de alternativa en caso de ser sobrepasado.

Debido a la gran cantidad de documentos que poseen algunos usuarios el tiempo de obtención de algunos de los reportes puede ser demasiado elevado, por lo que se decide que es posible que sea necesario encontrar alguna forma de ejecutar estos reportes de manera diferida y notificar al usuario a través de un correo electrónico su finalización.

Se decide que las columnas básicas para mostrar en todos los reportes (dependiendo del tipo de reporte tendrán más) sean:

- Title: título del archivo
- Mime Type: tipo de dato del archivo (documento,pdf,comprimido)
- Url: dirección web de acceso al archivo.
- Last modifying user: último usuario que modificó el archivo.
- Last viewing date by me: último acceso del usuario al archivo.
- Shareable by writers: si los escritores pueden compartir el archivo con otros usuarios.
- Type of Access: el rol que ejerce el usuario en el archivo.
- Owners: correo electrónico del propietario del archivo.
- Writers: correos electrónicos de los escritores del archivo.
- Readers: correos electrónicos de los lectores del archivo.

Conclusiones y tareas a realizar:

- Búsqueda de información respecto a ejecución en diferido dentro del entorno GAE.
- Investigación sobre acceso a información de archivos en Google Drive
- Investigación sobre la creación de Spreadsheet en un programa GAE.

Tercera reunión con el cliente (18/03/14)

Asistentes:

- Urbano Llamas, Fernando López y Jacinto Pelayo en calidad de clientes pertenecientes a la empresa Solvay Química S.A.
- Diego Gandarillas en calidad de gestor de proyecto.

Principales temas a tratar:

- Modificación de servicios
- Visibilidad en archivos
- Complejidad de obtención de roles

Resumen:

Tras mostrarle los avances al llegar a la fase de implementación y realizar pruebas del proyecto, el cliente cree necesario modificar alguna de las especificaciones, con lo que cambia el planteamiento inicial de un modelo de desarrollo de software en cascada por uno en espiral retornando de nuevo al análisis y posteriormente al diseño.

Los únicos reportes que realmente le interesan son los relacionados con Google Drive, Groups y Sites no serán incluidos en la aplicación. Además no existe manera de acceder al tipo de visibilidad de un archivo que no sea manualmente a través del servicio, por lo tanto, el reporte "Archivos en los que el usuario es propietario o editor y están compartidos con el dominio vía visibilidad" tampoco será incluido.

Por las nuevas especificaciones, el primer aspecto a modificar será la interfaz del menú, eliminando la selección del servicio y el reporte que no se puede obtener. Además no se ve necesario el uso de una página web exclusiva para la introducción de parámetros si no que es preferible que estén dentro del propio menú.

Para obtener los roles es necesario aumentar en mucho la complejidad y por tanto el tiempo de ejecución de la aplicación, en la reunión queda decidido de manera definitiva que el cálculo de los reportes será ejecutado en diferido sin que el usuario tenga que esperar a que termine.

Conclusiones y tareas a realizar:

- Modificación del diseño de la interfaz
- Ejecución del código de cálculo de reporte en diferido

Cuarta reunión con el cliente (20/05/14)

Asistentes:

- Urbano Llamas, Fernando López y Jacinto Pelayo en calidad de clientes pertenecientes a la empresa Solvay Química S.A.
- Diego Gandarillas en calidad de gestor de proyecto.

Principales temas a tratar:

- Entrega de la aplicación
- Evaluación de pruebas
- Comienzo de desplieguea

Resumen:

Última reunión con el cliente para poder entregarle y mostrarle todo el trabajo realizado. Considera que los resultados de las superan las expectativas y valida

la aplicación, además muestra cada uno de los pasos que se realizarán durante el proceso de despliegue.

Conclusiones y tareas a realizar:

- Incluir el proceso de despliegue en la memoria.

2.5. Completar especificaciones

2.5.1. Lenguaje y herramientas a utilizar

De los lenguajes que ofrece Google para la implementación de sus aplicaciones, Java será el utilizado para este proyecto ya que ha sido el más importante durante toda mi trayectoria universitaria y profesional. Eclipse será la plataforma de desarrollo por motivos similares, ya que además Google ofrece al usuario una herramienta para poder realizar en ese entorno las aplicaciones y posteriormente añadirlas al servicio App Engine.

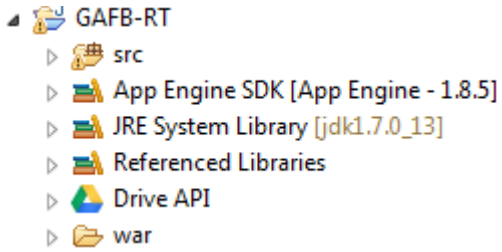

Image 7 - Proyecto Google App Engine en Eclipse

Este plugin permite construir fácilmente un nuevo proyecto de Google, creando un árbol de carpetas preestablecido, el directorio “src” contendrá los diferentes java-servlets, así como el resto de las clases Java a utilizar y un archivo json con las credenciales necesarias para crear el proyecto. Por el contrario en la carpeta “war” se deben introducir los diferentes archivos html y jsp.

Por último el plugin te ofrece una selección de APIs a utilizar. Una vez escogidas las necesarias, automáticamente serán incluidas sus librerías dentro del proyecto. Cuando existe algún tipo de cambio, aparece una notificación visual en el icono de esa API y sencillamente mediante un click se puede llevar a cabo este proceso de actualización.

Las pruebas serán ejecutadas a través de un servidor virtual que emulará el comportamiento de una aplicación almacenada en AppEngine ejecutándola en el puerto local 8888, ahorrando al programador una moderada cantidad de tiempo requerida para subir la aplicación al servidor.

2.5.2. Esquema general

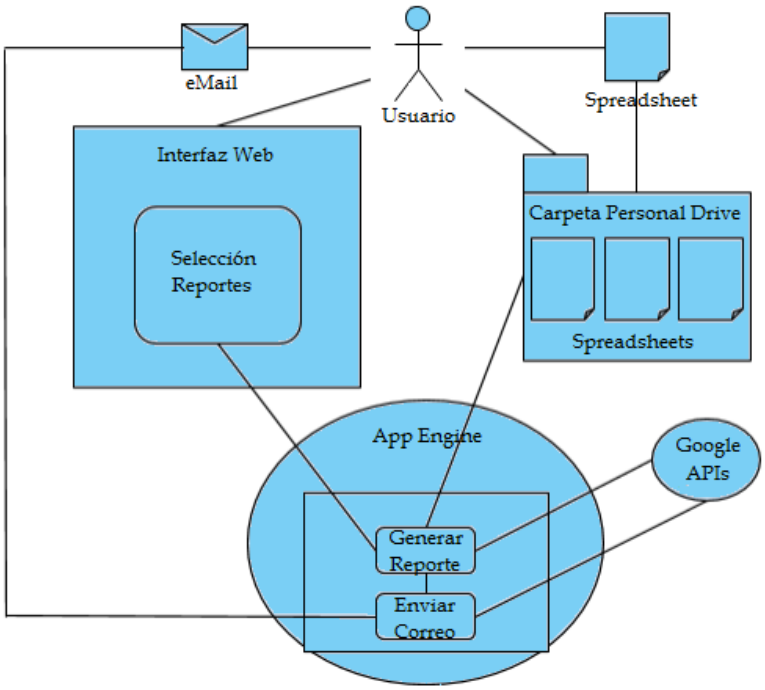

Image 8 Esquema de la aplicación

Interfaz web: Es un conjunto de páginas web donde el usuario iniciará sesión y establecerá los permisos, si es necesario, y donde podrá elegir el reporte e introducir los datos adicionales.

Proyecto en AppEngine: Es una aplicación programada en Java que obtendrá el reporte elegido por el usuario conectándose a las APIs pertinentes y lo almacenará en un Google Spreadsheet dentro de la carpeta creada en Google Drive; por último enviará un correo electrónico cuando finalice.

Carpeta personal de Drive: Lugar de almacenamiento de los reportes generados, desde allí el usuario podrá acceder a ellos y descargarlos en diferentes formatos.

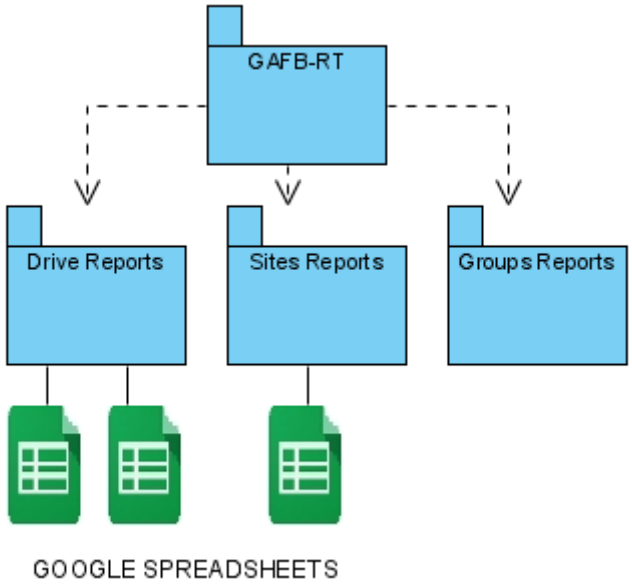

Image 9 - Árbol de carpetas (predefinido) en Google Drive

La figura anterior muestra el árbol de directorios perteneciente al enunciado de la aplicación original. De la raíz con el nombre de la aplicación parten tres carpetas que contendrán los diferentes reportes por servicio. Tras la tercera reunión con el cliente dejan de ser necesarios dos de ellos y queda decidido que todos los reportes sean almacenados en la carpeta “Drive Reports”.

2.5.3. Toma de requisitos

Para finalizar el capítulo de análisis a continuación se muestran los diferentes requisitos en alto nivel:

- Requisito 1: se desean generar reportes para el servicio Google Drive de los siguientes aspectos:
  - Archivos de los que el usuario es propietario.
  - Archivos en los que el usuario es editor y lector.
  - Archivos compartidos con un usuario configurable que actúa como propietario.
  - Archivos compartidos con un usuario configurable.
  - Archivos a los que el usuario tiene acceso, compartidos con cuentas fuera del dominio solvay.com.

    - Archivos a los que el usuario tiene acceso, compartidos con cuentas de un dominio configurable.
- Requisito 2: Los reportes serán generados en una hoja de cálculo dentro de una carpeta llamada “Drive Reports” en el servicio Google Drive del usuario, creada en caso de que no exista.
- Requisito 3: Se enviará un mail tras la finalización.
- Requisito 4: La generación del reporte se ejecutará en tiempo diferido.
- Requisito 5: La aplicación contará con una interfaz gráfica que guiara al usuario, a través de diferentes páginas web, durante todo el proceso.

Como parte de estos requisitos aparecerían los siguientes casos de uso:

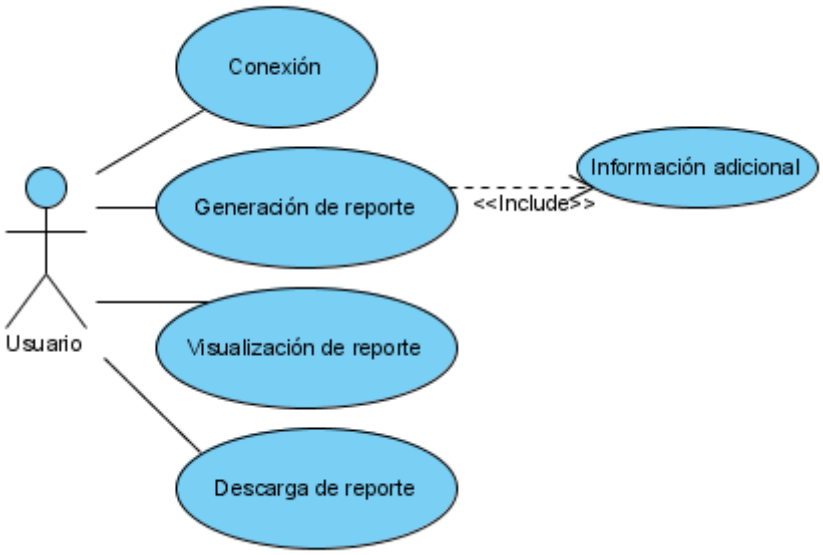

Image 6 – Modelo de casos de uso

Descripción detallada del caso de uso “Conexión”:

Precondiciónes:

El usuario dispone de conexión a Internet.

Secuencia de eventos de flujo:

1. El usuario accede a la página web “http:\\1.gaebrt.appspot.com”
2. Hace click en el botón Start
3. Inicia sesión en una cuenta de correo electrónico de Google en caso de que no lo haya hecho.
4. Acepta los permisos en caso de que no lo haya hecho.

Postcondiciónes:

En caso de que no haya cerrado la página el usuario estará conectado en la aplicación y habrá llegado al menú principal.

Descripción detallada del caso de uso "Generación de reporte":

Precondiciónes:

El usuario ya se ha conectado a la aplicación.

Secuencia de eventos de flujo:

1. El usuario selecciona un reporte de la lista.
2. Introduce información adicional en caso de que sea necesario
3. Hace click en el botón "ok".
4. Si quiere acceder a la carpeta de reportes de Google Drive hace click sobre esa opción en la nueva ventana.
5. Si quiere retornar al menú para realizar un nuevo reporte hace click sobre esa opción.
6. Si quiere finalizar la aplicación cierra la ventana

Postcondiciónes:

En caso de que el usuario haya seleccionado un reporte y la información adicional introducida posea el formato correcto, se generará el reporte dentro de la carpeta mencionada y se le notificará al finalizar mediante un correo electrónico.

Descripción detallada del caso de uso "Visualización de reporte":

Precondiciónes:

El usuario se ha conectado al servicio Google Drive.

Secuencia de eventos de flujo:

1. El usuario hace click en la carpeta "Drive Reports".
2. Hace click en el reporte que crea pertinente.

Postcondiciónes:

El usuario tendrá en su pantalla un documento del tipo hoja de cálculo con toda la información del reporte escogido.

Descripción detallada del caso de uso “Descarga de reporte”:

Precondiciónes:

El usuario se ha conectado al servicio Google Drive.

Secuencia de eventos de flujo:

1. El usuario hace click en la carpeta “Drive Reports”.
2. Hace click en el cuadrado a la derecha del reporte pertinente.
3. Hace click en el botón mas escogiendo la opción descargar.
4. Escoge el formato y hace click en la opción descargar.

Postcondiciónes:

El usuario tendrá dentro de su carpeta de descargas el reporte con el formato elegido.

3. Diseño

3.1. Diagrama de clases

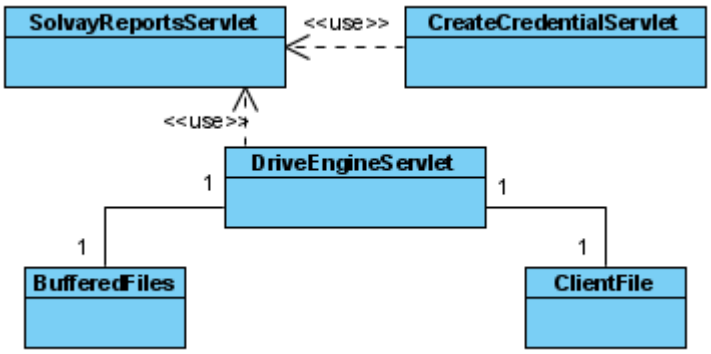

Ilustración 10 - Diagrama de clases Java

El código de la aplicación estará dividido en cinco clases java, tres de ellas serán servlets y dos auxiliares, serán explicadas detalladamente en la siguiente sección de esta memoria:

- SolvayReportsServlet: primer servlet de la aplicación llevará a cabo el proceso de autentificación, inicio de sesión y redirección al menú
- CreateCredentialServlet: clase para crear y almacenar las credenciales Oauth2 en caso de que el usuario actual no las posea
- DriveEngineServlet: se encargará de todo el proceso de obtención de los archivos del reporte elegido y posteriormente introducción en Google Drive y envío de eMail al usuario
- BufferedFiles: transforma todos los datos necesarios de los ficheros en un buffer de texto
- ClientFile: necesaria para añadir el buffer obtenido a un fichero de Google Drive

Image 11 Ficheros jsp y html

El proyecto también contiene dos ficheros jsp y html utilizados para la creación de la interfaz de usuario:

- Index.html: pagina web con mensaje de introducción.
- Menu.html: menú principal donde el usuario podrá escoger el reporte e introducir parámetros.
- Generate.jsp: leerá del queryString el reporte y los parámetros y los almacenará en la sesión.
- Finally.jsp: enviará el mensaje de finalización y mostrará la carpeta de Google Drive donde se generará el reporte, además dará la opción al usuario de regresar al menú.

Para mostrar el orden de ejecución de las clases e usará el siguiente diagrama:

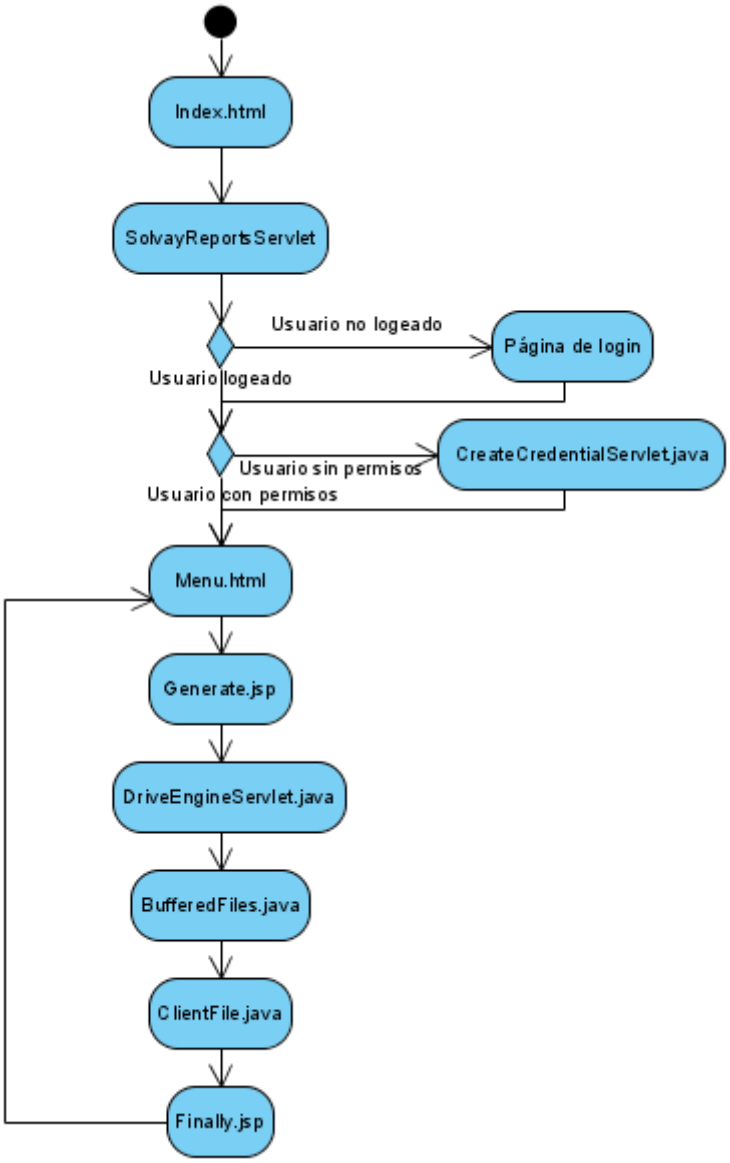

Image 12 - Diagrama de actividad

Al entrar a la aplicación aparecerá la página web “index” que mostrará un mensaje de bienvenida. Tras presionar el botón de aceptar comenzara a ejecutarse el servlet “SolvayReports”cuya primera función es comprobar si el usuario ha iniciado sesión, en caso negativo aparecerá la página de login de Google.

Acto seguido, si no han sido aceptados anteriormente los permisos de aplicaciones, al usuario se le mostrará un mensaje para que lo haga y el servlet “CreateCredentials” creará y almacenará las credenciales de autentificación.

La siguiente página que aparecerá será el menú principal de la aplicación con todos los reportes y cajas de texto para introducir correos de otros usuarios o dominios. Una vez escogido y pulsado el botón de aceptar, el fichero “generate.jsp” leerá y almacenará los datos del menú y dará paso a la clase “DriveEngineServlet”. Esta obtendrá los datos almacenados y redirigirá la aplicación al archivo “finally.jsp” que contendrá un enlace a la carpeta personal de Google Drive donde se almacenará el reporte y un vínculo para regresar al menú principal.

Mientras tanto en tiempo diferido se creará, en caso de que no exista, la carpeta contenedora y además se obtendrán todos los datos referentes al reporte seleccionado. Se utilizará la clase “BufferedFiles” para almacenar todos esos datos en un buffer de texto y la clase “ClientFile” para añadir ese buffer a un fichero. Por último el usuario automáticamente será notificado de la finalización a través de un correo electrónico con un vínculo al Google Spreadsheet creado.

3.2. Secuencia de ejecución

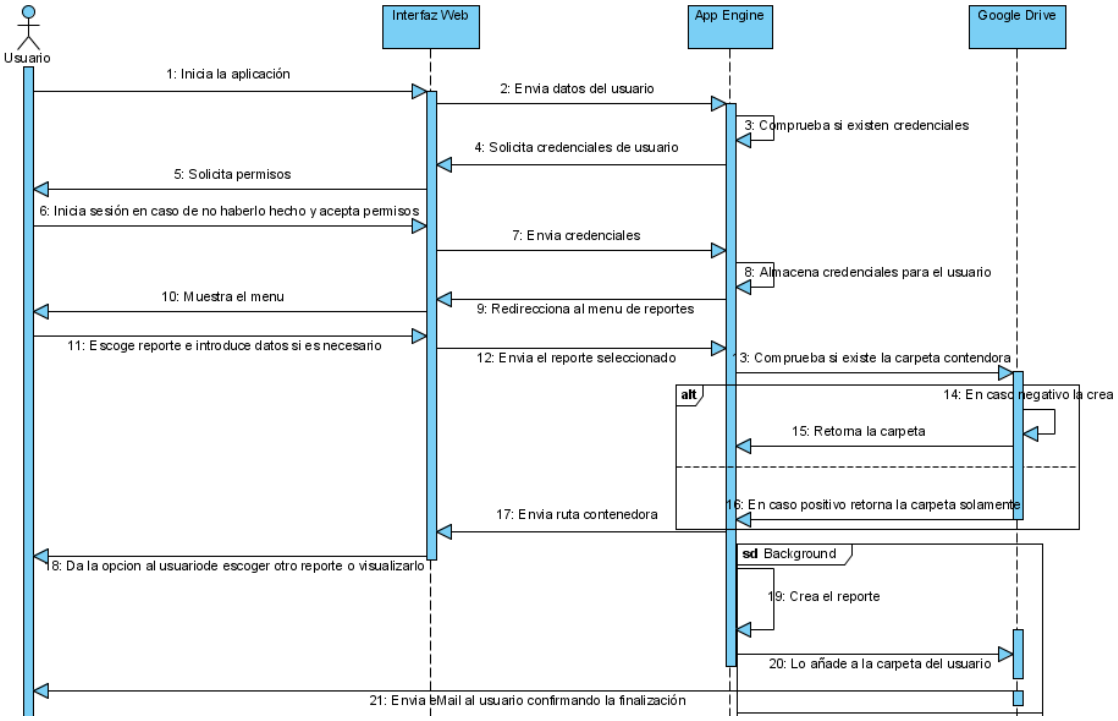

Image 13 - Diagrama de secuencia

3.3. Prototipos del menú de la interfaz

Primer diseño (09/01/14)

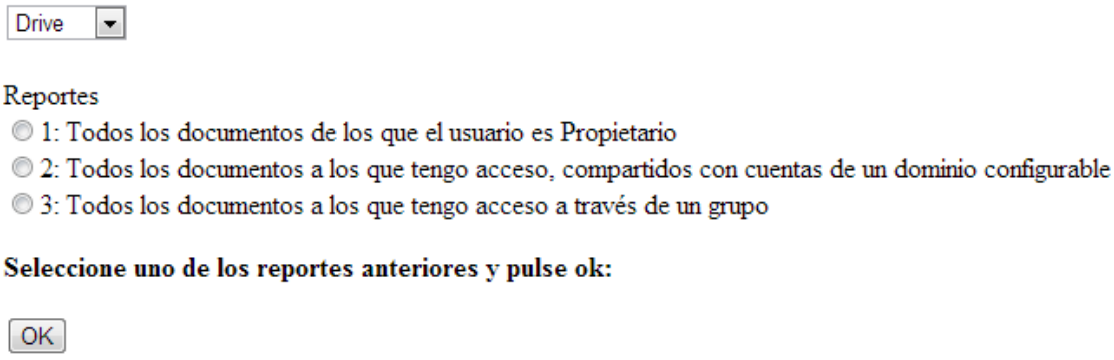

Image 14 - Primer diseño de interfaz

El primer prototipo de menú contenía un selector para elegir el servicio y tres botones radio con ejemplo de reportes. Este diseño sirvió de muestra básica para el cliente, pero hay ciertas cosas que mejorar e incluir:

- Cambio de idioma de texto a inglés ya que será una aplicación ejecutada por la empresa en varios países.
- Mostrar diferentes reportes en función del servicio seleccionado.
- Centrar el formulario en la pantalla ya que aparecía alineado a la izquierda.

Segundo diseño (17/01/14)

Image 15 - Segundo diseño de menú de interfaz

En este segundo prototipo los aspectos a mejorar ya estaban solucionados. Dinámicamente mostraba diferentes reportes en función del servicio, el idioma de todo el menú pasó a ser el inglés, y el diseño estaba centrado en la pantalla. El cliente quedo temporalmente conforme aunque se ve necesario realizar un diseño algo más atractivo.

Tercer diseño (25/03/14)

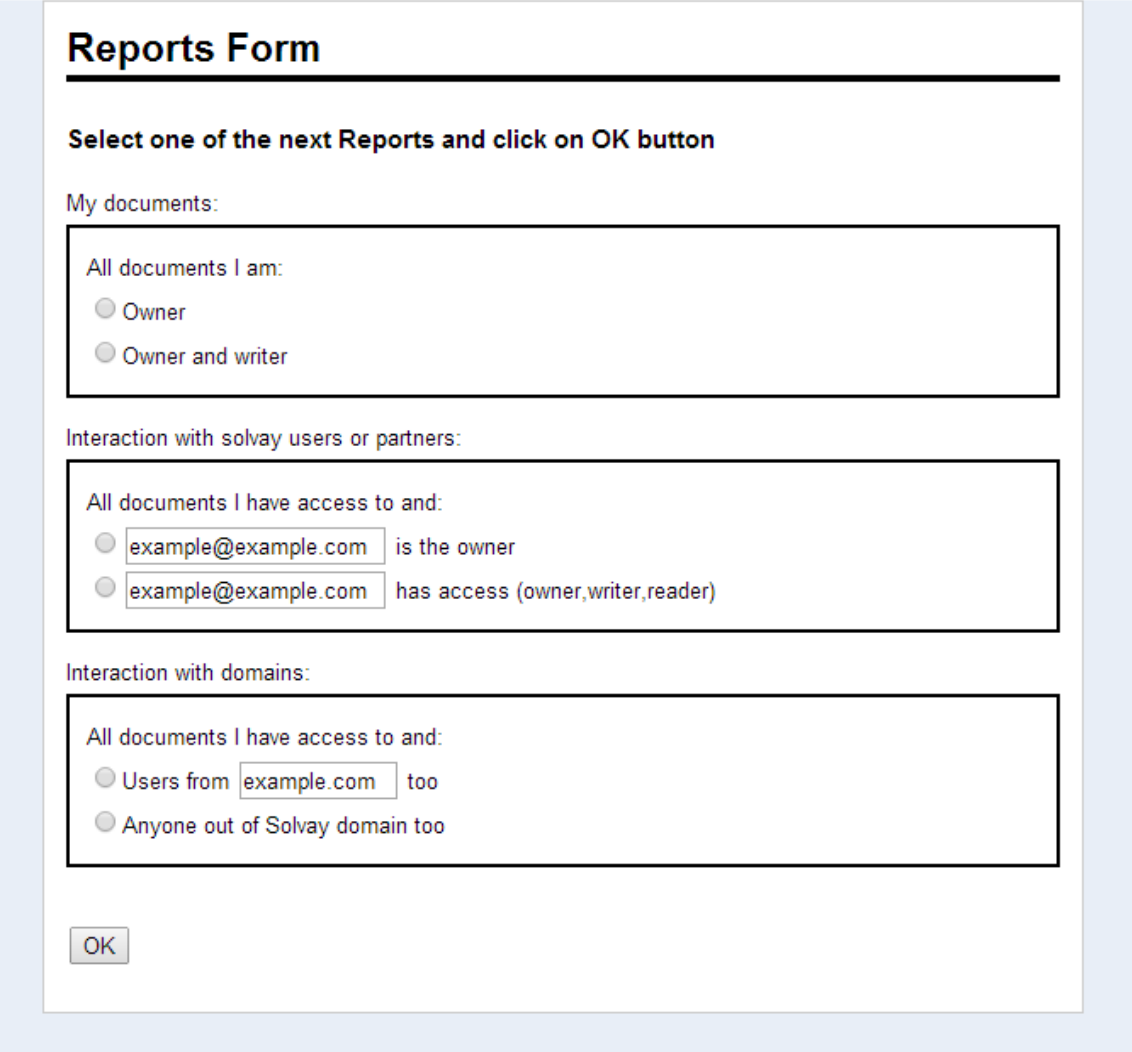

Image 16 - Tercer diseño de menú de interfaz

Tras establecer con el cliente que ya no son necesarios los reportes de dos de los servicios, el selector es eliminado y en la pantalla siempre aparecerán los mismos seis reportes (ya que el séptimo referente a la visibilidad no es posible realizarlo). Además es correcto separar los reportes en función del tipo que sean: usuario de la aplicación, otro usuario y dominios. El menú modifica su tamaño en función del tamaño de la ventana y el nuevo color del fondo hace que destaque más.

Por último, ya que es útil, se rellenan los campos con correos electrónicos o dominios de ejemplos y se evita continuar si el formato no es el correcto o no se ha seleccionado ningún reporte, mostrando en pantalla los avisos correspondientes.

4. Implementación

4.1. Implementación de las clases

4.1.1. SolvayReportsServlet

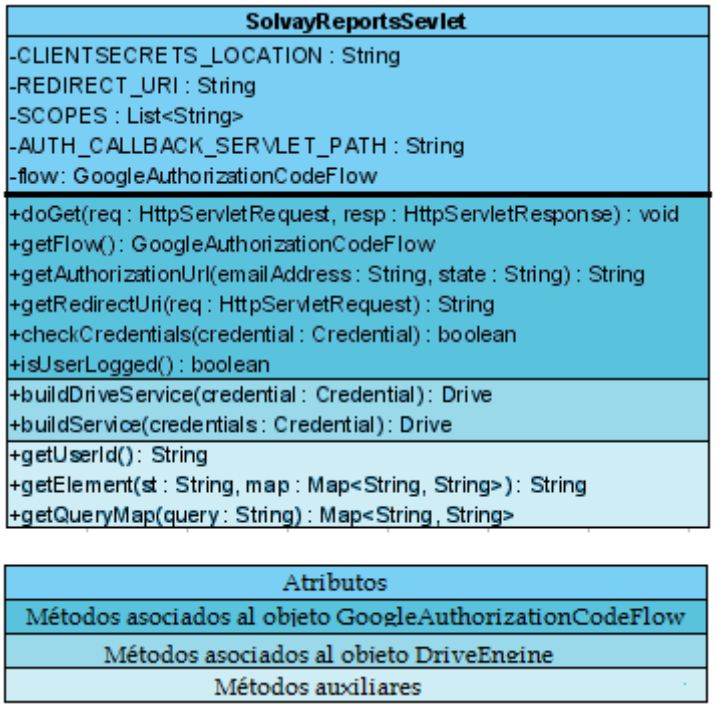

Image 17 – Diseño detallado de la clase SolvayReportsServlet

Es una clase dedicada al proceso de autentificación Oauth, la mayoría de los métodos trabajan con un objeto perteneciente de la clase GoogleAuthorizationCodeFlow que lleva a cabo este proceso en el lenguaje Java. Es necesario a la hora de crearlo asociarlo a una base de datos "AppEngineCredentialStore" donde quedan almacenadas las credenciales de cada usuario, y a un fichero gson con los datos obtenidos de la consola del proyecto

Existen dos métodos (buildDriveService y buildService) que crean el servicio de Drive utilizados posteriormente en la clase "DriveEngineServlet" para realizar las llamadas directamente a la API y obtener los diferentes reportes.

Por último el resto de los métodos son auxiliares para el resto de las clases, permitiendo obtener la id del usuario, almacenar los campos del "QueryString" de una página web en un mapa y obtenerlos**.**

4.1.2. CreateCredentialsServlet

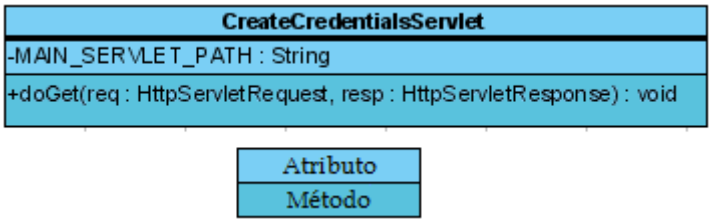

Image 18 - Diseño detallado clase CreateCredentialsServlet

Es el servlet utilizado como URI de redirección del flow tras retornar de la página web de habilitación de permisos, su función es la de crear y almacenar en la base de datos establecida en la iniciación del flow, las credenciales del usuario.

4.1.3. DriveEngineServlet

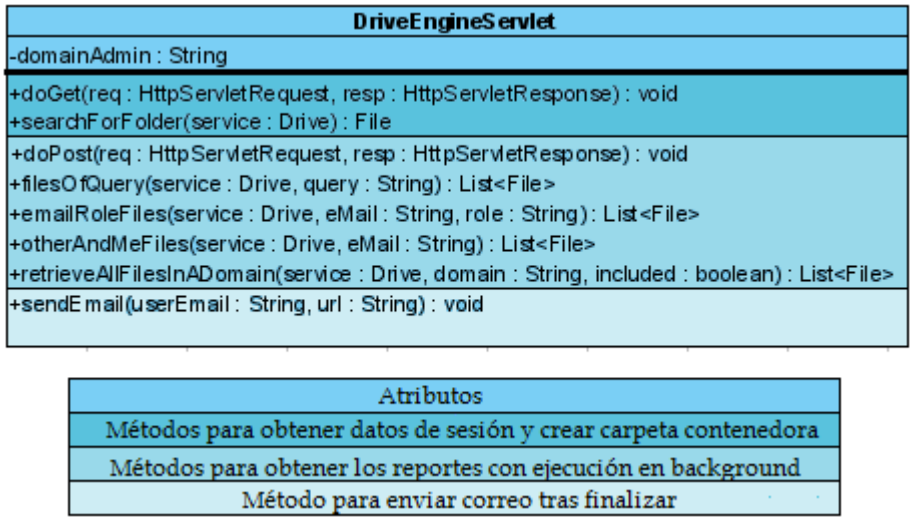

Image 19 - Diseño detallado clase DriveEngineServlet

Clase destinada a la obtención de los reportes en función de lo que haya elegido el usuario en el menú principal. Su método doGet, al comenzar la ejecución, se encarga de obtener los datos del reporte almacenados en la sesión y el email del usuario, en último término, inicia el servicio de Drive y la cola de ejecución en background que está asociada a la ejecución método

doPost. Utilizando el método searchForFolder se comprueba si existe la carpeta contenedora y en caso negativo la crea.

El método doPost realiza la obtención de los ficheros de Drive que coincidan con el reporte elegido, ayudado de diferentes métodos que los retornan en forma de lista.

Por último el método sendEmail realiza un envió automático de correo electrónico al usuario junto con un vínculo al Spreadsheet del reporte calculado.

4.1.4. BufferedFiles

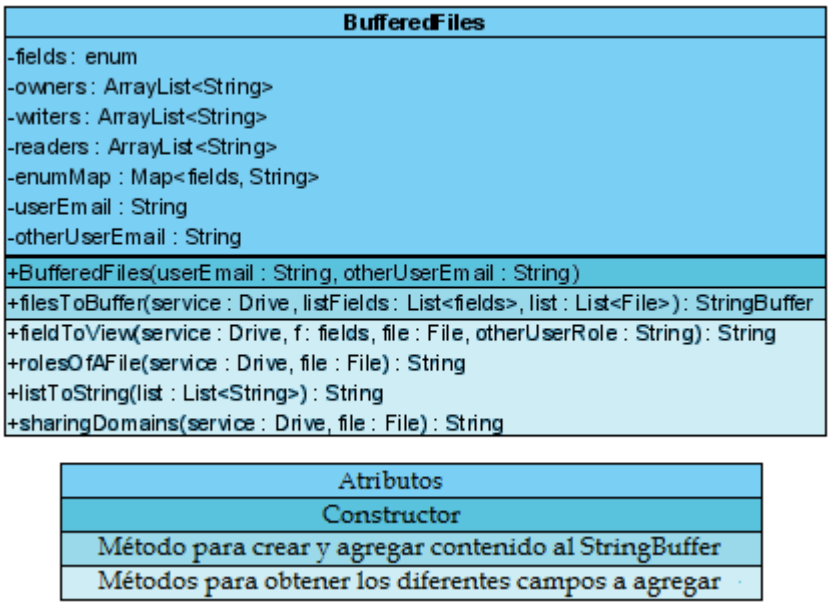

Image 20 - Diseño detallado clase BufferedFiles

Clase encargada de obtener todo el contenido de los ficheros del reporte y ser añadirlo posteriormente a un buffer de texto. De este proceso se encargará el método principal “filesToBuffer” que primero añadirá el título de las columnas a mostrar y por cada uno de los ficheros resultantes obtendrá el propietario, los escritores y lectores y los almacenará en los arraylist correspondientes como atributos.

Posteriormente llama al método “fieldToView” que retorna el valor de los campos básicos requeridos mencionados anteriormente (título,tipo,url,…) y los tres arraylist de los roles de usuario convertidos en texto. Para los reportes que están relacionados con el dominio llamará al método “SharingDomains”

que permite retornar un texto con los diferentes dominios con los que el archivo se comparte.

4.1.5. ClientFile

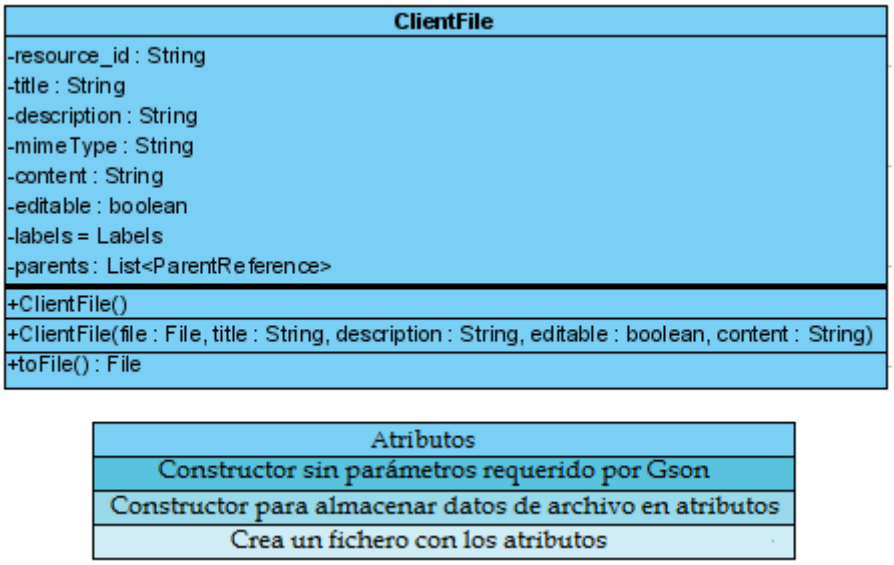

Image 21 - Diseño detallado clase ClientFile

Clase ligada a "DriveEngineServlet" su única función es almacenar los futuros datos de un fichero y el contenido del buffer de texto en los atributos de la clase. Acto seguido cuando se ejecuta el método "toFile" se añaden al fichero que será el spreadsheet.

4.2. Ejemplo de método doGet()

Inicialización del método

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)throws IOException {
```

Image 22 - Ejemplo I de método doGet

Como se puede comprobar, es necesario pasarle al método cómo parámetros un objeto "HttpServletRequest" que se encarga de la petición y otro HttpServletResponse que lo hace de la respuesta, siempre es necesario declarar que puede lanzar la excepción del tipo "IOException" ya que este método característico de los java servlets tiene riesgo de ello.

Obtención de datos necesarios

```
//The needs parameters
String userID = SolvayReportsServlet.getUserId();
String report= req.getSession().getAttribute("report").toString();
```

Image 23 - Ejemplo II de método doGet

Se utiliza el método “getUserId” del servlet SolvayReports para obtener la id del usuario que ejecuta la aplicación. También es necesario el reporte almacenado en el atributo “report” de la sesión por el fichero “generate.jsp”.

Obtención de parámetros adicionales

```
//Get the eMail of the session
String box= req.getSession().getAttribute("box").toString();
if (box.contains("%40")){
    //Replace %40 by @
    box = box.replace("%40","@");
}
```

Image 24 - Ejemplo III de método doGet

Si el usuario ha escogido cualquiera de los reportes de interacción con otras personas o el primero de los dos con otros dominios, se le requiere que introduzca un parámetro donde con ese otro usuario o dominio. Cuando se ejecuta el código contenido en el fichero “generate.jsp” la aplicación almacena en la sesión en un atributo llamado “box”. Este valor y en esta parte del método doGet se obtiene.

El carácter “@”, necesario para el correo de otros usuarios, el código lo transforma en “%40” cuando lo almacena, por tanto, es necesario volver a transformarlo en “@” cuando se lee.

Obtención de dominio del usuario

```
//Obtain all parameters needed for the user
UserService userService = UserServiceFactory.getUserService();
String usereMail = userService.getCurrentUser().getEmail();
String userDomain = usereMail.split("@")[1];
```

Image 25 - Ejemplo IV de método doGet

Para el último reporte (Obtener todos los ficheros compartidos con un dominio diferente al usuario) se necesita el dominio al que pertenece el usuario. Como no existe un método en si para realizar esto, la forma en que se hace es obtener su eMail y ejecutar el método “Split” que lo divide en un array en función del texto o letra que tenga de parámetro. Por tanto si el parámetro es “@” la primera posición de este array será la parte del usuario y la segunda, que es la que se necesita, será el dominio.

Creación y uso del servicio Drive

```
//Use the object AuthorizationCodeFlow
GoogleAuthorizationCodeFlow flow = SolvayReportsServlet.getFlow();
Credential credential = flow.loadCredential(userID);
//Create drive service
Drive service = SolvayReportsServlet.buildService(credential);
```

Image 26 - Ejemplo V de método doGet

Es necesario crear un objeto de la clase Drive, que es lo que utiliza Java para interactuar con los archivos alojados en este servicio. Para ello se utilizan las credenciales del usuario que están almacenadas en el objeto "GoogleAuthorizationCodeFlow" creado previamente.

Creación de la carpeta contenedora

```
//Folder
File folder= searchForFolder(service);
```

Image 27 - Ejemplo VI de método doGet

"SearchForFolder" es un método que busca si ya hay creada una carpeta denominada "Drive Reports" (lugar donde se almacenaran los futuros Spreadsheets de los reportes) en la carpeta personal del usuario del servicio Google Drive. En caso negativo la crea y en ambos casos retorna este directorio y lo almacena en una variable de tipo File.

Cola de tareas

```
// Build a task using the TaskOptions Builder pattern from ** above
Queue queue = QueueFactory.getQueue("optimize-queue");
queue.add(TaskOptions.Builder.withUrl("/driveEngine").method(TaskOptions.Method.POST).param("userID",userID)
```

Image 28 - Ejemplo VII de método doGet

El uso de una cola de ejecución de tareas, en la cual el cálculo de los reportes se realiza en tiempo diferido tiene dos partes; la primera consiste en la creación de la misma asociándola a una ya definida dentro del fichero queue.xml.

```
<queue-entries>
<queue>
<name>optimize-queue</name>
<rate>20/s</rate>
<bucket-size>40</bucket-size>
<max-concurrent-requests>10</max-concurrent-requests>
<retry-parameters>
<task-retry-limit>7</task-retry-limit>
</retry-parameters>
</queue> </queue-entries>
```

Image 29 - Fichero queue.xml

En este fichero se establecen los parámetros básicos de las colas que se utilizan en la aplicación:

- Name: nombre de la cola.
- Rate: cuantas veces son procesadas las tareas por unidad de tiempo.
- Bucket-Size: el tamaño del buffer de la cola.
- Max-concurrent-requests: máximo número de peticiones concurrentes.
- Retry-parameters: parámetros para cuando se produce un fallo.

```
<servlet>
    <servlet-name>SolvayReports</servlet-name>
    <servlet-class>reports.SolvayReportsServlet</servlet-class>
</servlet>
<servlet>
    <servlet-name>CreateCredentials</servlet-name>
    <servlet-class>reports.CreateCredentialsServlet</servlet-class>
</servlet>
<servlet>
    <servlet-name>DriveEngine</servlet-name>
    <servlet-class>reports.DriveEngineServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>SolvayReports</servlet-name>
    <url-pattern>/solvayreports</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>CreateCredentials</servlet-name>
    <url-pattern>/oauth2callback</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>DriveEngine</servlet-name>
    <url-pattern>/driveEngine</url-pattern>
</servlet-mapping>
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

Image 30 - Fichero web.xml

La segunda parte para el uso de la cola es añadirla una url de ejecución. En este caso es el mismo java servlet (DriveEngineServlet) que aparece asociado al nombre “DriveEngine” que a su vez esta mapeado a la página /driveEngine. Todo esto se debe introducir dentro del archivo “web.xml”. Dentro de esa web es necesario especificar que método se va a ejecutar que será el doPost; por último para obtener el reporte son necesarios todos los parámetros anteriormente calculados y se le deben pasar.

Redirección final:

```
//Redirect to finally.jsp
req.getSession().setAttribute("folder",folder.getAlternateLink());
resp.sendRedirect("/finally.jsp");
```

Image 31 - Fichero web.xml

Finalmente, mientras se calcula el reporte, la aplicación redirige al usuario al jsp “finally” donde tiene las opciones de regresar al menú de reportes o de ir a la carpeta contenedora de reportes de Google Drive (por eso se almacena la dirección en la sesión).

5. Pruebas

5.1. Pruebas de tiempo

5.1.1. Cola de ejecución

La cola de ejecución tiene una serie de parámetros modificables dentro del archivo “queue.xml”. Uno de ellos es el “rate”, que define el número de tareas a ejecutar por unidad de tiempo. Dentro del entorno de pruebas (ya que se debe variar el código), para un mismo reporte con el mismo número de resultados se ejecutan tres pruebas dentro de un rango de “rates” y se puede obtener el siguiente gráfico:

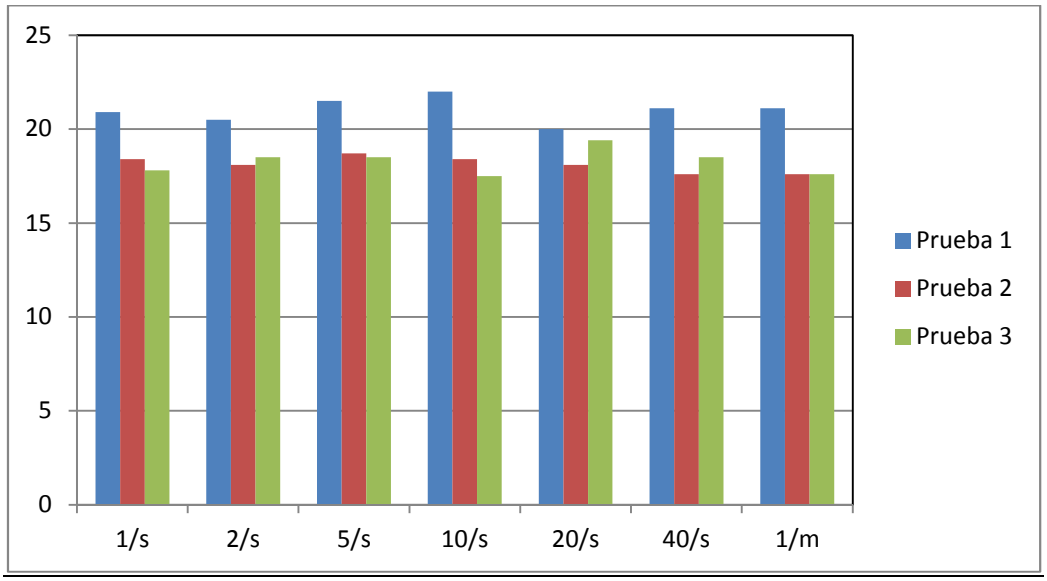

Image 32 - Tiempo de ejecución para tres pruebas de un mismo reporte con diferente rate

Los resultados son en la mayoría de los casos muy similares, por lo que se puede demostrar que cada reporte constituye una sola tarea y quizás puedan encontrarse diferencias de tiempo si se ejecutan varios reportes alternativamente.

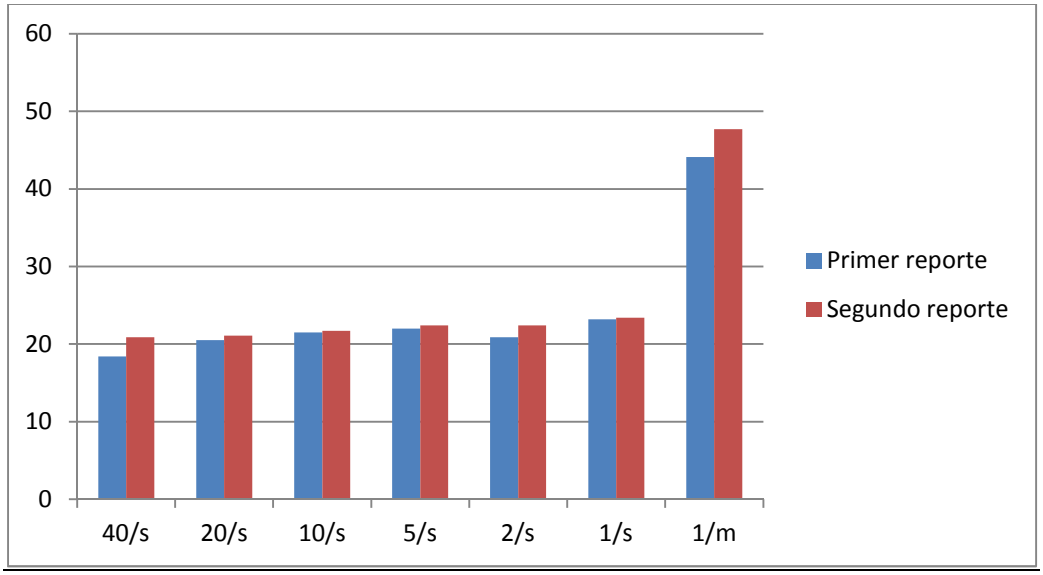

Image 33 - Tiempo de ejecución para dos reportes simultáneamente con diferente rate

Como se puede comprobar tampoco ocurre a no ser que el rate sea muy bajo. Por tanto queda asumido que el tamaño de tareas no es suficientemente grande como para que modificar el "rate" (a no ser que sea considerablemente) constituya una gran mejora.

Algo muy parecido ocurre con el resto de los parámetros que no será necesario modificarlos considerablemente, ya que de manera simultánea en ejecución habrá un máximo de tres reportes y sus valores mínimos no sobrecargarán el sistema.

5.1.2. Tiempo de ejecución total

Para realizar las mediciones de tiempo total de ejecución se realizan diferentes pruebas variando el número de archivos que cumplan las características, para incluirse en cada reporte. Se ha decidido establecer un rango mínimo de 5, que encaja en la media de ficheros de usuarios que utilizan poco el servicio Google Drive, hasta 500, que lo hace con los que lo hacen frecuentemente. Por último, es necesario destacar que estas pruebas, a diferencia de las anteriores, han sido ejecutadas en el entorno App.Engine ya que no se modifica el código de la aplicación.

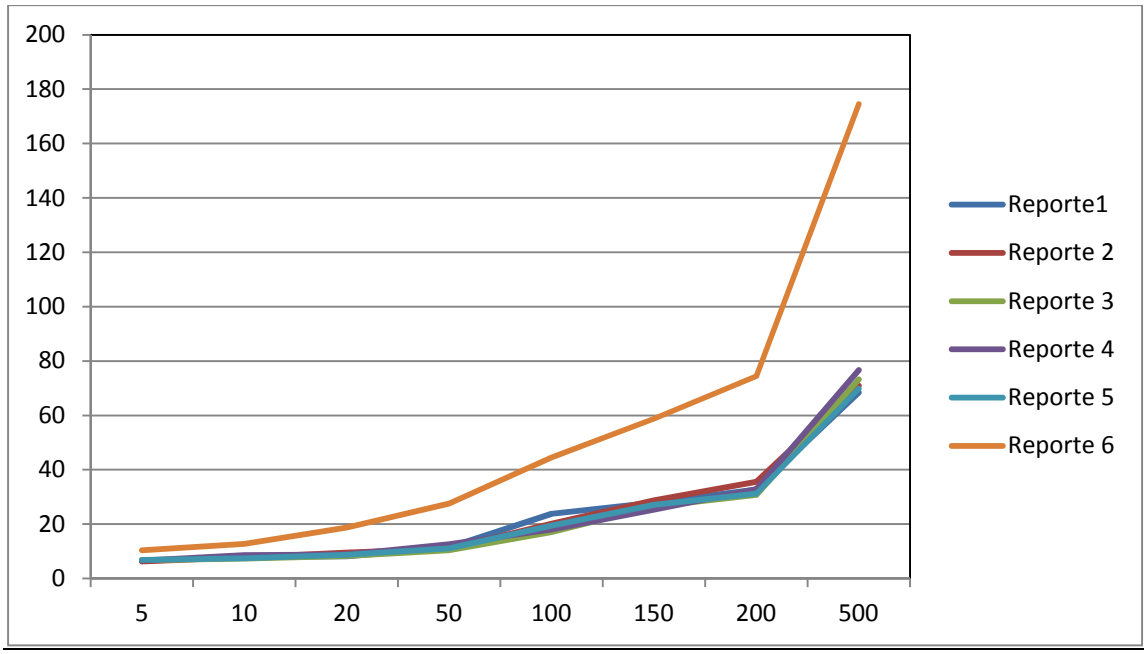

Image 34 - Tiempo de ejecución de los reportes para diferentes cantidades de archivos

Los cinco primeros reportes obtienen los datos de una manera similar, basada en mandar peticiones solo para los archivos que se requieran, de ahí que sus líneas temporales no diverjan demasiado, e incluso para obtener más de 500 archivos, solo es necesario poco más de un minuto.

Para el reporte 6 (Documentos compartidos con un dominio diferente al usuario) es necesario realizar peticiones para descargar todos los archivos contenidos en Google Drive y posteriormente comprobar todos los dominios, filtrarlos y descartar los ficheros que no cumplan la condición del reporte.

5.2. Pruebas de visibilidad

5.2.1. Pruebas de visibilidad de correos electrónicos

Debido a las políticas de privacidad de Google a la hora de mostrar los correos electrónicos de sus usuarios, es importante realizar pruebas tanto de administrador como de usuario normal debido a que ambos ejercen diferentes roles dentro de un mismo documento y puede suponer una diferencia a la hora de los eMails que se pueden obtener.

DENTRO DEL DOMINIO

El primer paso es realizar las pruebas con todos los usuarios pertenecientes al dominio “solvay.com”. Para ello se utilizarán tres documentos en los que el usuario será propietario, escritor y lector respectivamente y cada uno de ellos será compartido con dos usuarios de la empresa con diferentes roles dentro del archivo:

Usuario Administrador		Correos a los que el usuario puede acceder		
		Owner	Writers	Readers
Rol de Usuario	Owner	Todos los correos	Todos los correos	Todos los correos
	Writer	Todos los correos	Todos los correos	Todos los correos
	Reader	Todos los correos	Ningun correo	Ningun correo

Image 35 - Visibilidad de correos para un usuario administrador con usuarios dentro de un mismo dominio

Usuario Normal		Correos a los que el usuario puede acceder		
		Owner	Writers	Readers
Rol de Usuario	Owner	Todos los correos	Todos los correos	Todos los correos
	Writer	Todos los correos	Todos los correos	Todos los correos
	Reader	Todos los correos	Ningun correo	Ningun correo

Image 36 - Visibilidad de correos para un usuario administrador con usuarios dentro de un mismo dominio

Para administradores y usuarios normales se obtienen los mismos resultados; un usuario podrá visualizar todos los correos que comparten un documento, a no ser que sea lector del mismo, que solo podrá acceder al del propietario. No supone un problema crítico, ya que para la mayoría de archivos de los que se necesitan obtener estos datos la persona que ejecuta la aplicación no tiene el rol de lector.

FUERA DEL DOMINIO

Análogamente, para tres nuevos documentos compartidos con dos usuarios de diferentes dominios se realizan las mismas pruebas que en el caso anterior, dando lugar a diferentes resultados:

Usuario Administrador		Correos a los que el usuario puede acceder		
		Owner	Writers	Readers
Rol de Usuario	Owner	Todos los correos	Solo los habilitados	Solo los habilitados
	Writer	Todos los correos	Solo los habilitados	Solo los habilitados
	Reader	Todos los correos	Ningun correo	Ningun correo

Image 37 - Visibilidad de correos para un usuario administrador con usuarios de diferentes dominios

Usuario Normal		Correos a los que el usuario puede acceder		
		Propietario	Writers	Readers
Rol de Usuario	Owner	Todos los correos	Solo los habilitados	Solo los habilitados
	Writer	Todos los correos	Solo los habilitados	Solo los habilitados
	Reader	Todos los correos	Ningun correo	Ningun correo

Image 38 - Visibilidad de correos para un usuario normal con usuarios de diferentes dominios

De nuevo se concluye que no importa el tipo de usuario a la hora de acceder a los correos electrónicos que compartan un documento, es decir sea

administrador o no, se obtienen los mismos resultados. Siempre se mostrarán los propietarios de un archivo y al igual que el caso anterior los lectores no podrán visualizar ningún tipo de correo aparte de este, pero esta vez, en el resto de los casos solo podrán visualizarse los eMails si el usuario lo tiene habilitado en las preferencias de su cuenta de Google+.

A la hora de realizar los reportes cuando se comparte información con dominios diferentes a "solvay.com" el dato realmente necesario es dicho dominio, por lo que el hecho de que no se muestren los correos que no hayan sido habilitados no constituye un gran problema.

5.2.2. Pruebas de visibilidad de archivos

Existe una restricción a la hora de acceder a un archivo y ésta es el tipo de visibilidad que tenga. Normalmente existen tres opciones; que solo puedan acceder a un archivo las personas incluidas en el mismo, las personas que reciban el enlace o cualquier persona. El dominio solvay.com también posee dos opciones más; permitir el acceso solo a las personas del dominio o solo a las que reciban el enlace dentro del dominio.

A la hora de realizar los reportes es importante saber qué documentos se incluyen en ellos en función de su visibilidad, antes de abrirse por primera vez y después de ello, debido a que al realizar esta acción, se crea una copia de los archivos con visibilidad pública (total o solo para el dominio) dentro de la carpeta "compartidos", en el directorio del servicio Google Drive del usuario.

Usuario Administrador	Sin abrir	Abierto
Archivos de visibilidad publica	No se muestran	Se muestran
Archivos de visibilidad publica dentro del dominio	No se muestran	Se muestran
Archivos de visibilidad privada	Se muestran	Se muestran

Image 39 - Acceso a archivos en función de su visibilidad para un usuario administrador de dominio

Usuario Normal	Sin abrir	Abierto
Archivos de visibilidad publica	No se muestran	Se muestran
Archivos de visibilidad publica dentro del dominio	No se muestran	Se muestran
Archivos de visibilidad privada	Se muestran	Se muestran

Image 40 - Acceso a archivos en función de su visibilidad para un usuario normal

Como era de esperar solo se muestran en los reportes los archivos de visibilidad pública para ambos tipos de usuario que ya han sido abiertos. Esto

tiene sentido debido a que hasta que no se acceda a él, no se incluirá entre los del usuario.

No es el caso de los archivos de visibilidad privada, ya que cuando se incluye un correo electrónico en un documento, este ya aparece directamente en el servicio Drive del usuario.

5.3. Pruebas de funcionamiento

Reporte1: Documentos de los que el usuario es propietario

Pruebas	Resultado esperado	Resultado obtenido
Documentos mostrados	7	7
Campos mostrados por documento	10	10
Creación de la carpeta contenedora en Google Drive	Si	Si
Introducción de spreadsheet en carpeta contenedora	Si	Si
Envío de correo	Si	Si

Reporte 2: Documentos de los que el usuario es propietario o escritor

Pruebas	Resultado esperado	Resultado obtenido
Documentos mostrados	14	14
Campos mostrados por documento	10	10
Creación de la carpeta contenedora en Google Drive	No	No
Introducción de spreadsheet en carpeta contenedora	Si	Si
Envío de correo	Si	Si

Reporte 3: Documentos de los que otro usuario es propietario

Pruebas	Resultado esperado	Resultado obtenido
Documentos mostrados	7	7
Campos mostrados por documento	11	11
Creación de la carpeta contenedora en Google Drive	No	No
Introducción de spreadsheet en carpeta contenedora	Si	Si
Envío de correo	Si	Si

Reporte 4: Documentos a los que tiene acceso otro usuario

Pruebas	Resultado esperado	Resultado obtenido
Documentos mostrados	10	10
Campos mostrados por documento	11	11
Creación de la carpeta contenedora en Google Drive	No	No
Introducción de spreadsheet en carpeta contenedora	Si	Si
Envío de correo	Si	Si

Reporte 5: Documentos compartidos con otro dominio

Pruebas	Resultado esperado	Resultado obtenido
Documentos mostrados	4	4
Campos mostrados por documento	10	10
Creación de la carpeta contenedora en Google Drive	No	No
Introducción de spreadsheet en carpeta contenedora	Si	Si
Envío de correo	Si	Si

Reporte 6: Documentos compartidos con un dominio diferente al del usuario

Pruebas	Resultado esperado	Resultado obtenido
Documentos mostrados	6	6
Campos mostrados por documento	11	11
Creación de la carpeta contenedora en Google Drive	No	No
Introducción de spreadsheet en carpeta contenedora	Si	Si
Envío de correo	Si	Si

Image 41 - Resultados de pruebas de funcionamiento

La imagen anterior muestra el resultado de una prueba de cada reporte realizada desde una cuenta de correo electrónico de usuario normal. La carpeta contenedora “Drive Reports” no ha sido creada, por tanto que lo haga en el primer reporte es un resultado correcto.

Estas pruebas fueron realizadas en documentos compartidos con el administrador del dominio de test de la empresa y con algunas cuentas del dominio principal. En todos los casos los resultados coinciden con los esperados, por tanto queda concluido que la aplicación funciona correctamente.

6. Despliegue

6.1.Proceso

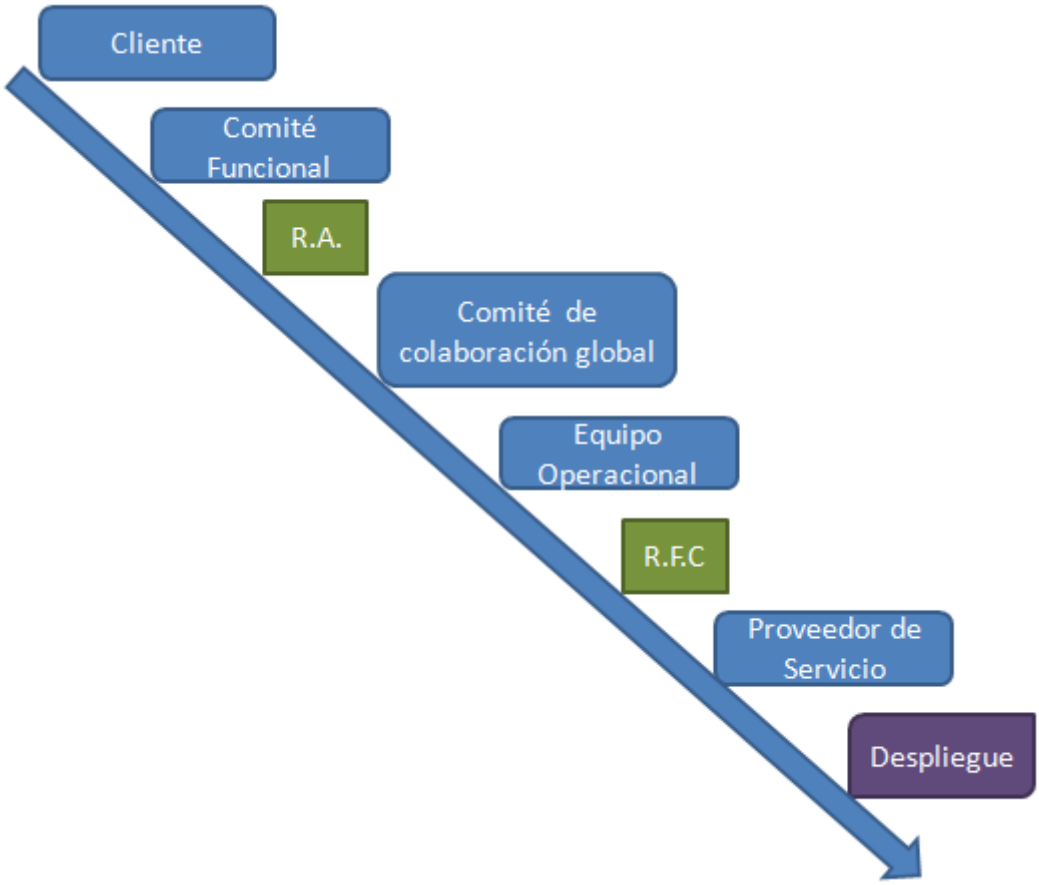

Image 42 – Proceso de Despliegue

Dentro de la empresa, a la hora de realizar una petición formal de despliegue de un nuevo software, son necesarios una serie de pasos:

Una vez finalizada la aplicación, se entrega a un comité funcional que se encarga de instalarlo en el entorno de test, para comprobar, a través de una serie de pruebas, si el funcionamiento es correcto.

Mediante reuniones semanales, se discute el resultado, y en caso de ser correcto, el equipo redacta un documento llamado “Request Analysis” que envía a un equipo de colaboración global formado por miembros expertos en Google Apps.

El objetivo del comité es evaluar el documento, y en caso de aceptarlo, se lo notifica a un equipo operacional que crea un documento R.F.C (request for change) y se lo manda al proveedor de servicio, en este caso la empresa externa Revevol.

Finalmente el proveedor realiza el despliegue en el entorno de producción y el comité funcional realiza las últimas pruebas. En algunos casos primero para un pequeño grupo de usuarios y posteriormente para toda la empresa.

7. Conclusiones y trabajos futuros

7.1. Planing Definitivo

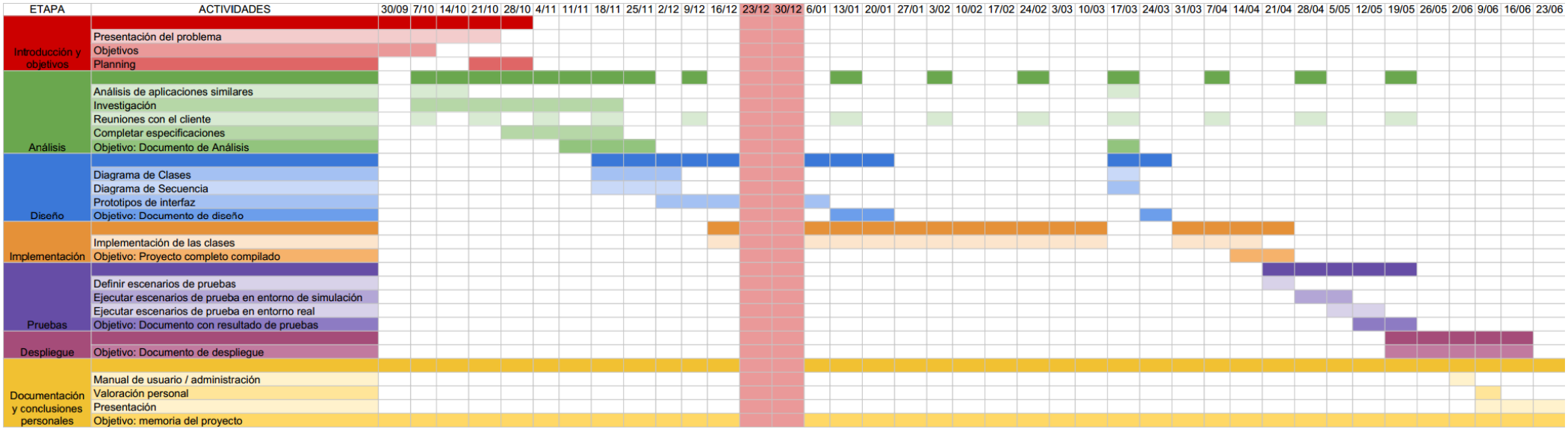

Image 43 – Planing Definitivo

7.2. Conclusiones personales

Al haber sido una aplicación realizada para un cliente final, que en este caso es una empresa a nivel internacional, me ha servido como preparación para un futuro laboral como desarrollador de software. Además las necesidades del cliente han ido variando durante el todo el proceso de vida del proyecto y un gran mérito propio ha sido adaptarme, conseguir resolver todos los retos planteados y realizar por primera vez una memoria asociada a un proyecto de este calibre.

Debo hacer una mención a las reuniones establecidas con la empresa con cierta periodicidad, ya que en ellas se mostraron los resultados de las investigaciones y se indicó lo que es posible o no realizar dentro de los márgenes de tiempo establecidos. Es esencial que el cliente haya podido comprobar que el proyecto está avanzando y poder validar los diseños y prototipos que le fui mostrando para que el resultado final después de los múltiples cambios sea el que desea.

Al ser como he dicho antes una empresa internacional, muchas de las entidades para las cuales va dirigido este trabajo no dominan mi lenguaje natal, el español, por ello tanto la aplicación en sí, al igual que todos los comentarios en el código, han sido realizados en el idioma internacional, el inglés. Además casi todos los tutoriales, ejemplos y manuales que hay en la red sobre esta tecnología también están en dicho idioma. Casualmente estoy en la preparación de un examen para evaluar mi nivel del mismo y todo lo anterior ha sido de gran ayuda.

Varias veces durante el proyecto han aparecido dificultades que han impedido el avance y ha sido necesario una búsqueda intensa en la red, a veces durante varias horas o días, para solventar dichos problemas. Es un trabajo que ha evitado la aparición de problemas similares en el futuro y además me ha permitido realizar comparativas entre las diferentes soluciones, ofreciéndome la posibilidad escoger la que mejor se adapte a la aplicación, consiguiendo muy gratos resultados.

Por último quiero añadir que partiendo de un desconocimiento completo del entorno Google App Engine, del desarrollo de los java servlet y de los archivos jsp, además de un conocimiento muy básico del lenguaje html, es muy gratificante el saber que tras unos pocos meses he sido capaz de dominarlo y realizar un programa muy útil para una empresa englobada dentro de las mayores internacionalmente en el campo de la química.

7.3. Dificultades encontradas y soluciones tomadas

7.3.1. Visibilidad de los ficheros

Una de las características de los ficheros de Google Drive es su visibilidad, es decir, el grupo de usuarios que pueden acceder al mismo. Por defecto son tres, aunque dependiendo del dominio pueden ser hasta cinco.

Uno de los reportes requeridos estaba relacionado con esta propiedad, pero el problema radica en que el campo que muestra esta propiedad es de solo escritura. Por tanto no se puede acceder como lectura e imposibilita la realización de este reporte. La solución es eliminarlo de la aplicación.

7.3.2. Mostrar los correos

Cada archivo posee una serie de propiedades, una por cada usuario con el que esta compartido. Una de ellas es el correo electrónico del mismo, que es un dato requerido para esta aplicación, aunque Google posee una fuerte política de privacidad respecto al tema.

El primer problema apareció a la hora de intentar acceder a los correos compartidos con otros dominios diferentes. Con ésta política no es posible mostrar los correos electrónicos cruzando dominios a no ser que el usuario al que se referencia tenga habilitada esta opción. Según el cliente esto no es un problema crítico ya que la mayoría de correos electrónicos necesarios estarán dentro del dominio “solvay.com”.

El segundo problema importante es que dependiendo del rol que el usuario desempeñe en el fichero podrá visualizar o no ciertos eMails. Este es el caso del usuario lector que solo podrá tener acceso al del propietario, aun perteneciendo los escritores y lectores al mismo dominio. Tampoco el cliente lo consideró como un problema muy grave, ya que siendo lector de un documento creyeron que no se tiene por qué tener acceso al resto de correos.

Pero el problema más importante surgió a finales de Abril cuando la aplicación estaba ya implementada, y al realizar una prueba, fue imposible el

acceso a cualquier correo de usuario de un fichero. El caso es que la empresa lanzó una actualización para su API de Drive en la cual se produce este error. Afortunadamente la solución fue automática ya que días después quedo corregido.

7.3.3. Agregar el contenido a un Spreadsheet

Google Apps Script, que permite el desarrollo de pequeños programas integrados en los diferentes tipos de fichero de Google, a la hora de agregar contenido a un Spreadsheet, provee una API en la que se permite el acceso a cada una de las celdas de manera muy sencilla.

Al estar integrado en el entorno App Engine, intuitivamente, debería poder realizarse de una manera similar pero no es así. Tras buscar mucha información, pude encontrar que la manera de realizarlo es construir una clase, agregar el texto del contenido como atributo de un objeto y luego utilizarlo para crear el fichero.

La segunda dificultad es ordenar la información en filas y columnas, ya que está todo en un mismo texto. La solución es utilizar un buffer de texto en el cual se avanza de columna utilizando el carácter “,” y de fila con el salto de línea. Esta opción hace sea que sea necesario reemplazar las comas de los títulos de los ficheros y se decide que sea por “;”.

7.3.4. Tiempo de ejecución elevado

Poco a poco iba aumentando la complejidad de la aplicación. Por cada reporte se obtiene un número n de ficheros y cada fichero tiene m usuarios, por tanto mínimamente es de O (m*n), sin contar otras operaciones. M y n son normalmente elevados, por lo que el tiempo también lo es y no es factible hacer esperar al usuario hasta que se realicen los cálculos. En resumen, estos cálculos se deben hacer en tiempo diferido.

Google ofrece a los desarrolladores un tipo de configuración de programa llamada “back-end” con cuatro niveles diferentes en función de la CPU y cantidad de memoria que se quiera contratar y ahí radica el problema. Es necesario contratar los servicios realizando un pago económico.

La solución gratuita fue optar por otro de sus servicios que son las “task-queues”; añadiendo los cálculos a una cola de tareas se puede librar al usuario de esta espera. La complejidad de esta opción está en encontrar los parámetros óptimos para configurar la cola. De ahí radica la realización de varias pruebas relacionadas.

7.3.5. Asociar una base de datos de credenciales de usuario

Mediante las primeras pruebas de la aplicación se pudo observar que el usuario cada vez que la iniciaba tenía que aceptar los permisos, y no se quedaban almacenados. Se debía encontrar algún tipo de lugar que almacene estas credenciales.

El resultado de un pequeño proceso de investigación es que las nuevas bases de datos de credenciales no son gratuitas. En versiones más antiguas de la API al crear el objeto “G.A.C.Flow” se utilizaba el método al que se le pasaba un objeto “AppEngineCredentialStore”. Aunque actualmente esta descatalogado, se decidió realizar una prueba. El resultado es el esperado y, por el momento, el método no será eliminado de la API de Google Drive, por tanto es la opción utilizada.

7.3.6. Envío de correos electrónicos

Google provee al usuario de métodos en Java para realizar el envío de correos electrónicos de una manera sencilla, aunque posee dos principales limitaciones; la primera es que solo es posible dicho envío al ejecutar el programa dentro del entorno de App Engine, es decir, que no es posible que se realice en el puerto del entorno local donde se ejecutan las pruebas. No ocasiona un problema grave ya que cuando este desplegada la aplicación no será ejecutada en el entorno de pruebas.

El segundo impedimento es que no es posible enviar un correo electrónico si el remitente no es el administrador de la aplicación. Por ello, al comienzo de la clase “DriveEngineServlet” hay una variable global para que el programador que utilice la aplicación escriba el correo de este superusuario.

7.4. Posibles mejoras futuras

Esta aplicación posiblemente podrá ser ampliada en el futuro, ya que ha sido creada una base sólida con todo el esfuerzo dedicado a la investigación y han ido apareciendo opciones que se pueden añadir:

Aunque actualmente la empresa no requiere reportes de servicios diferentes a Google Drive, podrían realizarse más reportes de este mismo servicio o de otros, si se cree necesario y sin la necesidad de modificar demasiado ninguna de las clases ya realizadas que no pertenezcan a la interfaz de usuario. Además al estar ya implementado todo el proceso de obtención y verificación de permisos y de credenciales no será necesario realizarlo para nuevos servicios.

Otra posible ampliación es añadir opciones que se puedan realizar posteriormente a la realización de los reportes, por ejemplo, una vez obtenidos los ficheros en los que el usuario es propietario, ofrecer la posibilidad de hacer una copia de todos directamente desde el programa. Otro ejemplo es poder eliminar el acceso de un determinado dominio a los archivos propios de la empresa SOLVAY; esto es útil si esta empresa deja de trabajar con el dominio y evita al usuario encargado de esta tarea, el tener que ir uno a uno realizándolo.

Estos son solo dos ejemplos pero existen una gran cantidad más; crear tipos diferentes de reportes diferentes para administrador de dominio y usuario normal, permitir al usuario establecer una periodicidad automática de uno o varios reportes , dar la posibilidad de crear los reportes en un formato diferente como, por ejemplo, las recientemente creadas Google Fusion Tables,…

8. Manual de usuario

1) El usuario para comenzar debe introducir la siguiente dirección web (*http://1.gaebrt.appspot.com/*) en el navegador, se le mostrará la pantalla de introducción a la aplicación:

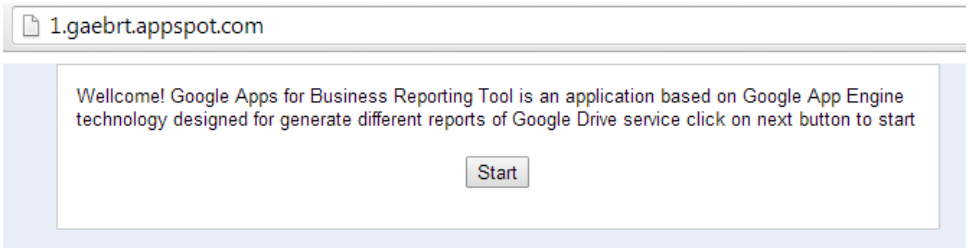

Image 44 – Pantalla de introducción a la aplicación

2) Una vez presionado el botón “start” si no ha iniciado sesión en algún tipo de correo electrónico de Google será redirigido a la página de login:

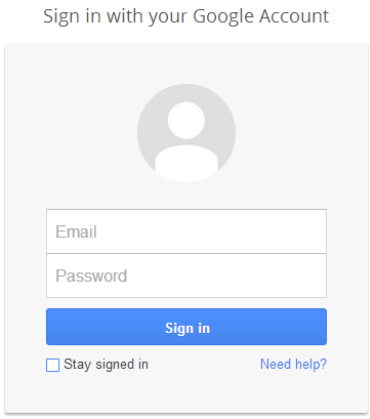

Image 45 - Pantalla de login

3) Si no ha aceptado los permisos necesarios de ejecución anteriormente le aparecerá un mensaje para que lo haga:

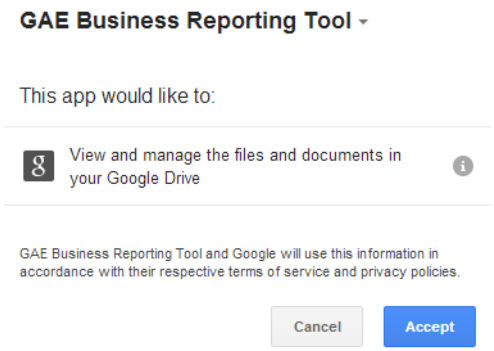

Image 46 - Mensaje de permisos

4) Esto dará paso al menú de la aplicación donde aparecen las diferentes opciones de reporte que se le ofrecen al usuario:

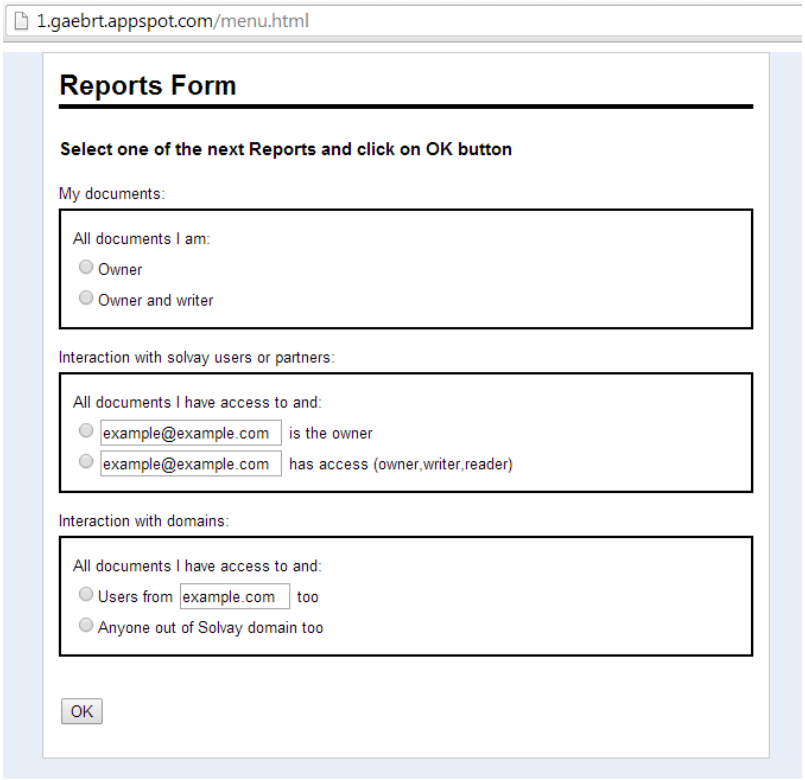

Image 47 - Menu de la aplicación

5) Tras seleccionar cualquiera de ellos y presionar el botón “ok” la aplicación ejecutara las operaciones necesarias para obtenerlo y a través de la siguiente dirección web ofrecerá la opción de visualizar la carpeta de Google Drive donde será introducido o volver de nuevo al menú para realizar otro reporte.

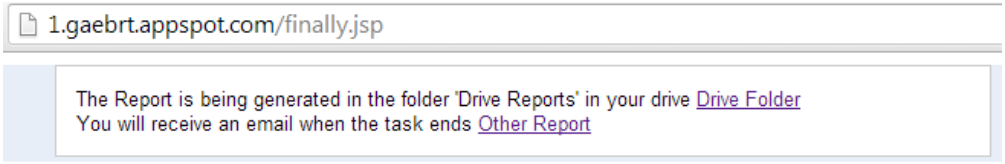

Image 48 - Pantalla de finalización

6) Una vez finalizada la ejecución, el reporte aparecerá en la carpeta “Drive Reports” en el servicio Google Drive y el usuario será notificado a través de un correo electrónico con el administrador de dominio como entidad que lo envía:

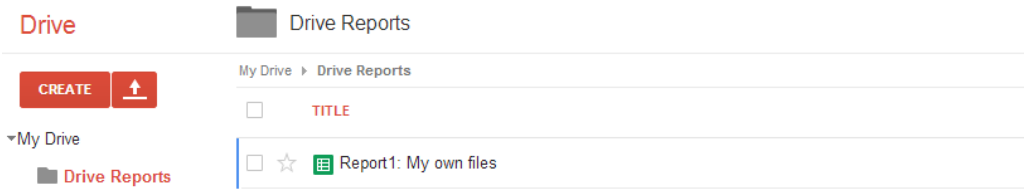

Image 49 - Carpeta “Drive Reports” de Google Drive

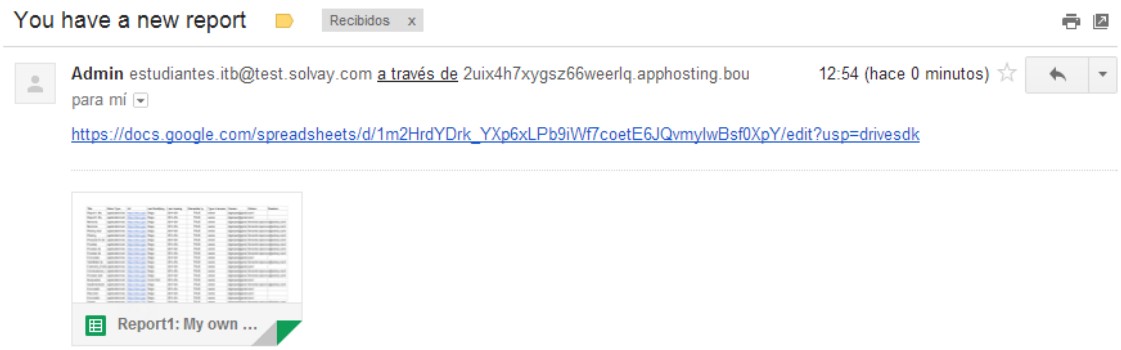

Image 50 - Email de finalización de reporte

9. Bibliografía

https://generalaudittool.com/

http://es.wikipedia.org/wiki/JasperReports

http://es.wikipedia.org/wiki/Crystal_Reports

http://en.wikipedia.org/wiki/Google_App_Engine

https://developers.google.com/appengine/training/intro/whatisgae

http://jesusnoseq.com/2012/12/probando-la-api-de-google-drive/

https://developers.google.com/accounts/docs/OAuth2?hl=EN

https://developers.google.com/accounts/docs/OAuth2WebServer?hl=EN

https://developers.google.com/drive/v2/reference/

http://stackoverflow.com/questions/11894772/google-drive-mime-types-listing

https://groups.google.com/forum/#!topic/google-appengine/tr9Bqiar3ng

https://developers.google.com/drive/web/search-parameters

http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

https://developers.google.com/appengine/docs/java/backends/

https://developers.google.com/appengine/docs/java/taskqueue/overview-push?hl=es&csw=1

https://code.google.com/p/google-oauth-java-client/wiki/OAuth2

http://users.dcc.uchile.cl/~jbarrios/servlets/general.html

https://developers.google.com/appengine/docs/java/config/queue