Conector Zimbra con software de gestión

Pere Víctor Vives Viña

Resumen—. Este trabajo consiste en llevar un proyecto habitual en la Ingeniería del software, es decir tomar varias decisiones de diseño y implementación de software. Principalmente una empresa que se llama Efficensway quiere añadir distintas posibilidades que ofrece una plataforma de mailing llamada Zimbra a su programa E.R.P intentando así minimizar los accesos a esta plataforma y hacer posible funcionar solamente utilizando el E.R.P. Zimbra ofrece varias prestaciones, servicio de correo, agenda de contactos, calendarios, tareas y maletín. Para este proyecto será necesario integrar las tres primeras. Para hacer esto, será necesario trabajar principalmente en Genero, y utilizar SOAP para las llamadas al servidor, aunque las herramientas a utilizar serán muy variadas y aquí es donde radica la dificultad de este proyecto. Además, tendré que hacer distintas modificaciones en las bases de datos para almacenar la información que será necesaria para esta adaptación. También se deberá mejorar el módulo de proyectos del programa E.R.P mejorando sus prestaciones.

Paralabras clave—Zimbra, Genero, email service, E.R.P, Enterprise resource planning, SOAP, Ticketin, mailing, contact list, calendars.

Abstract—This project consists of managing a regular software engineering project, from the initial design until the implementation. Efficensway, a software development company, wants to add to its E.R.P. certain possibilities from a mailing platform called Zimbra for the sake of being able to use the platform only from the E.R.P. Zimbra offers a wide range of benefits: mailing, contact list, calendars, task list and briefcase. In this project, we will only focus the main three. For doing this, we will have to work with Genero, and use SOAP for the server calls. However, the difficulty of the project resides in the diverse and complex tools we will use. Moreover, certain modifications in the database will be needed in order to store the information that will be needed for the adaptation into the E.R.P. Lastly, the E.R.Ps projects module should also be improved to enhance its benefits.

Index Terms—Zimbra, Genero, email service, E.R.P, Enterprise resource planning, SOAP, Ticketin, mailing, contact list, calendars.

1 Introducción

Este trabajo consiste en integrar varias de las funcionalidades de una plataforma de mailing llamada Zimbra al programa E.R.P de una empresa llamada EfficensWay. Para hacer esto será necesario utilizar una infrastructura de desarrollo llamada Genero referencia [1] ya que la empresa la utiliza. Estas modificaciones serán programadas en un servidor de pruebas. La versión de zimbra con la que se trabajará será la 8.0.7_GA_6020.

2 Objetivos

Los objetivos del trabajo són basicamente adaptar, las funcionalidades de correo, de contactos y de agenda de la plataforma Zimbra al programa E.R.P, además de añadir distintas visualizaciones más al módulo de proyectos de la empresa. En concreto las funcionalidades a incorporar pretenden automatizar varios procesos que actualmente són hechos de manera manual.
En concreto los objetivos són:

- Conseguir el enviamiento de correos automáticamente mediante Zimbra, no es adecuado utilizar otro medio, porque el propósito final es que el correo esté certificado y que los mensajes automáticos enviados, aparezcan en la bandeja de salida del zimbra, para poder saber en todo momento que se ha enviado, a quién, cuándo y por qué.
- Mostrar una bandeja de entrada de correos, para hacer esto, será necesario añadir una ventana nueva para mostrar una bandeja de entrada de correos y a su vez, hacer que cuando llegue un mensaje nuevo, se envie una respuesta de confirmación de recibimiento, en otras palabras, un Ticketin simple.
- Zimbra funciona mediante carpetas, y en estas se pueden realizar varias funciones, así como, buscar cualquier información en ellas, y el programa tendrá que ser capaz de sincronizar cualquier carpeta de Zimbra en él.
- Diseñar y implementar un sistema de contactos en el que se pueda hacer la sincronización bidireccional, es decir, dar la posibilidad tanto de enviar contactos del programa a Zimbra como de el Zimbra al programa. Y sobretodo evitar duplicados, para no colapsar las bases de datos, hay que tener en cuenta que normalmente las empresas que utilizan estos programas de gestión, sus contactos son fundamentales y el sistema tiene que ser una herramienta para agilizar todos los procesos de este ámbito como podrían ser, busque-

das de contactos simple o compartición de contactos, todo esto sin que el usuario tenga que tener un control exhaustivo de ellos.
- En el módulo de proyectos, primeramente, incorporar distintas visualizaciones para poder ver las tareas según el tiempo, para ello habrá que estudiar diversas posibilidades que ofrece el mercado para conseguir una framework que se adapte a las necesidades del programa y de la empresa. Y en concreto en la Agenda Empresa habilitar la funcionalidad para poder enviar tareas al Zimbra como citas en el calendario.
- Conseguir en cualquier implementación un nivel de calidad óptimo, en otras palabras, que sea un producto acabado, que no haya ni inconsistencia de información, y sobretodo que en ningún caso haya errores que conlleven un bloqueo en el programa.

El objetivo del trabajo general, es conseguir aplicar los conocimientos adquiridos, ya no solamente los conocimientos puramente informáticos, sino también los que derivan de ellos, para poder abarcar un proyecto, típico en un ámbito empresarial, y también relativa a la Ingeniería del Software puramente dicha, donde la mayoría de herramientas a utilizar són conocidas a nivel general, pero no en detalle, no he tenido el placer de diseñar ni de implementar nada similar y ver si realmente puedo llegar a superar todos los inconvenientes que pueda tener.

3 ESTADO DEL ARTE

Las distintas empresas que utilizan el programa E.R.P tienen que realizar algunas acciones manualmente, como, por ejemplo:

- Al generar una factura mediante el programa, tiene que haber un usuario del programa que realice la tarea de coger el pdf generado por el programa y mandarlo por correo al respectivo cliente que ha realizado la operación.
- Asignar citas y reuniones manualmente mediante la interfaz de cliente de zimbra, sin tener notación mediante el programa, supone a veces un descontrol.
- Cada vez que tienen que buscar un contacto, tienen que hacer distintas búsquedas, a veces tienen que buscar sólo en los contactos del programa, a veces tienen que utilizar otra agenda personal, donde ir guardando cada contacto, y cuando tienen un contacto nuevo tienen que ingresarlo manualmente, primero en su lista de contactos y luego en la lista del programa.
- No hay ningún sistema de gestión de los correos entrantes, esto hace que se llenen las bandejas de entrada de los técnicos de mensajes que carecen de información relevante, o bien que no sean el destinatario final de estos y que ese correo en realidad, tendría que ir a un departamento distinto, es decir, si se recibe un correo y éste no es respondido, hay clientes que mandan más correos de confirmación del estilo: *"¿Has recibido el correo que te he mandado?"* esto supone a parte de llenar aún más la bandeja, que el técnico tenga que utilizar más tiempo para responder a mensajes de este tipo y esto también nos da información de que el cliente no recibe un buen feedback desde el programa.
- La información relativa a los proyectos, se muestra en distintas tablas y no ayuda a que sea comprensible, es decir para ver una tarea, la información se entiende bien, pero si queremos ver la trazabilidad entre varias tareas o bien, qué tareas tiene un técnico asociadas en un día determinado, la visualización actual es precaria. Tampoco existe la posibilidad de generar gráficos donde se muestren los tiempos de los trabajos.

Estas tareas o acciones a realizar, en estos diferentes contextos, puede parecer que sean acciones rápidas y que carezca de importancia invertir tiempo en mejorarlas. Pero a nivel empresarial los usuarios de estas empresas mientras realizan estas acciones, están perdiendo tiempo de trabajo para hacer otras cosas y eso repercute directamente a la empresa, ya que el tiempo de cada usuario supone un coste. Así que, con la realización de este trabajo pretendo evitar el sobrecoste que estas y tareas similares conllevan.
Las ventanas a mejorar són las siguientes:

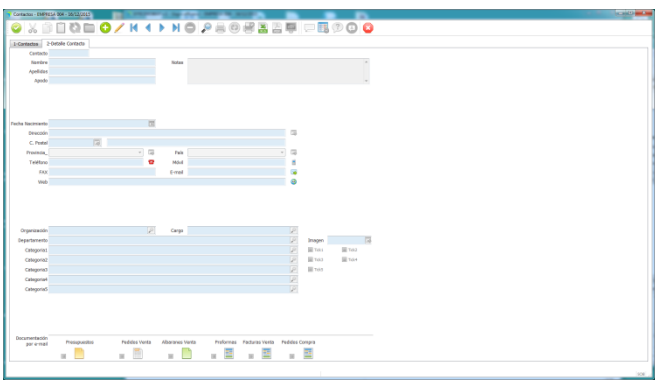


Figura 1 Ventana de Contactos

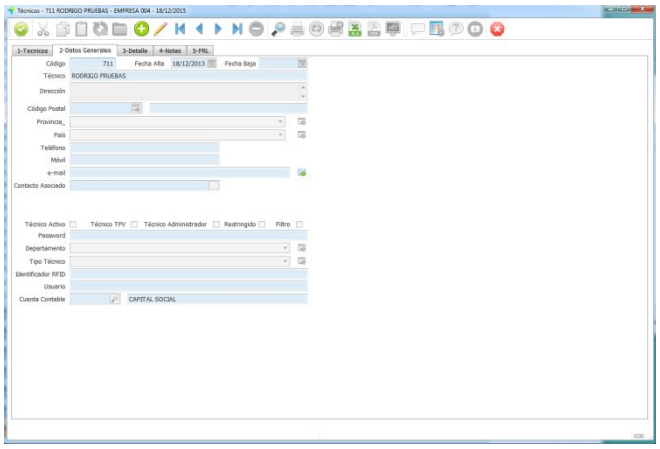


Figura 2 Ventana de Técnicos

3 CONCEPTOS

3.1 Qué es Zimbra

Zimbra es una plataforma open source ideada para dar servicio de correo electrónico, pero que en la actualidad integra varias funcionalidades, cómo puede ser su función cómo lista de contactos, cómo agenda de calendarios, cómo servicio de compartición de archivos mediante el maletín y el propio sistema de mailing.

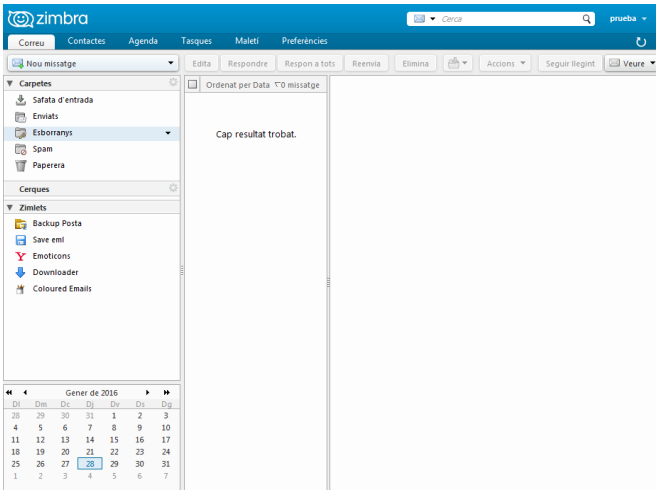


Figura 3 Zimbra

3.2 Qué es Genero

Genero es una plataforma de programación destinada al desarrollo de aplicaciones para empresas y negocios que requieran básicamente interactuar con sus respectivas bases de datos, es propiedad de 4js. Es muy versátil ya que se adapta a los distintos sistemas operativos sin que haya demasiada diferencia. Utiliza Business Development Language BDL, como lenguaje, prácticamente no hay palabras de lenguaje y se reducen los errores de desarrollo y la longitud del código en una gran magnitud. Básicamente está ideado para lograr la independencia en tres sentidos:

- Independencia de sistema operativo.
- Independencia de tipo de cliente GUI.
- Independencia de base de datos.

Básicamente se compila hacia un bytecode independiente de plataforma, y se ejecuta utilizando una Dynamic Virtual Machine (DMV). Es similar al método que utiliza Java para conseguir la independencia de plataforma.

3.3 Qué es Soap y por qué es tan necesario en este proyecto

Siglas de Simple Object Acces Protocol es un protocolo que define las bases de comunicación entre varios procesos. Para este proyecto es fundamental, porque es mediante él protocolo que se comunica el programa E.R.P con el servidor de Zimbra.
SOA no está ligado a http, puede utilizar otros medios como SMTP. Utiliza XML estructurado para comunicar la información y es independiente de la capa de transporte. SOA está muy estandarizado y permite usar WSDL para comunicarse y en si SOAP provee maneras de publicar y de descubrir servicios.

3.4 Programa E.R.P

Básicamente un E.R.P Enterprise Resource Planning es un software de gestión. Que sirve para agilizar la gestión empresarial, en líneas generales:

- simplifica el papeleo utilizando varias automatizaciones.
- permite controlar y ver la salud de su empresa.
- Todo el proceso desde los almacenes al cliente final es automatico, sin perder la noción en todo momento de lo que está pasando.
- Cómo tienes control total en cada momento permite tomar decisiones ágiles.

En concreto el E.R.P de la empresa Efficensway tiene los siguientes modulos, tal y como está explicado en la página web referencia[4]:

Figura 4 Módulos del software de la Empresa

En este proyecto los módulos a mejorar, serán proyectos, ventas, compras, y otros.

- proyectos: con la incorporación de distintas visualizaciones.
- ventas y compras, para poder enviar directamente las facturas por correo.
- otros: porqué es donde están las funcionalidades de administración interna de los técnicos y de los contactos, y añadir un subapartado en otros para gestionar los correos entrantes.

4 METODOLOGÍA

La implementación de estas mejoras en la parte del programa, se realizará en un servidor de pruebas, en éste trabajan varios técnicos y será necesario siempre que se hace una modificación, seguir el control de versiones establecido.
En concreto trabajaré en un archivo nuevo que se llamará zimbraSOAP donde residiran las funciones que interactuarán directamente contra el webservice de zimbra mediante SOAP, además de incluir pequeñas modificaciones en contactos, técnicos y en proyectos.
Para implementar cada petición SOAP he tenido que utilizar una herramienta que se llama SOAP UI para ajustar los parámetros de la api de Zimbra referencia[2], concretar lo indispensable que tenía que rellenar en cada paso, y luego adaptar la llamada con el lenguaje del programa, Genero facilita está parte al disponer de un adaptador WSDL donde básicamente las variables de request y de response se generan automáticamente, pero aún así hay varias cuestiones a tener en cuenta, como algunos errores que tiene este generador, y que he tenido que solventar, como que el header de la respuesta soap, no se espera que esté vacío y Zimbra lo envía vacío y varias dificultades que explicaré más adelante.

5 FUNCIONALIDADES IMPLEMENTADAS

5.1 Funcionalidad básica, variable de sesión:

Primero de todo tenemos que tener en cuenta que para interactuar con Zimbra necesitamos una variable de sesión llamada AuthToken, ésta tiene duración limitada y expira al cabo de un tiempo. Es decir, para enviar un correo o para realizar cualquier otra acción siempre necesitaremos de éste AuthToken. Y para conseguirlo tenemos que hacer una llamada vía Soap a la función Auth del webservice de zimbra, con nuestra cuenta y la contraseña.

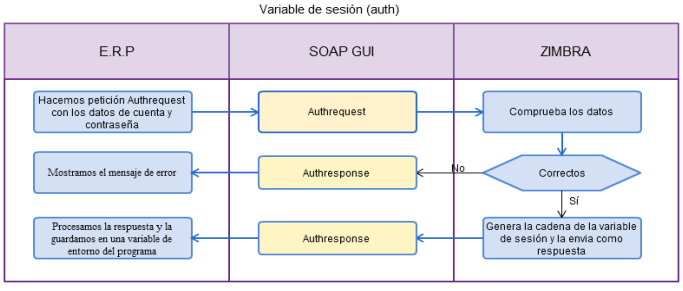


Figura 5 Variable de sesión

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:zimbra" xmlns:urn1="urn:zimbraAccount">
   <soapenv:Header>
      <urn:context>

      </urn:context>
   </soapenv:Header>
   <soapenv:Body>
      <urn1:AuthRequest>
         <urn1:account by="name">CUENTA</urn1:account>

         <urn1:password>CONTRASEÑA</urn1:password>

      </urn1:AuthRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 6 ejemplo llamada soap

5.2 Funcionalidad enviar correos

Para poder enviar un correo primero debemos tener una variable de sesión, y luego tenemos que comprobar si el mensaje que queremos enviar contiene documentos adjuntos o no, si contiene, entonces deberemos hacer una

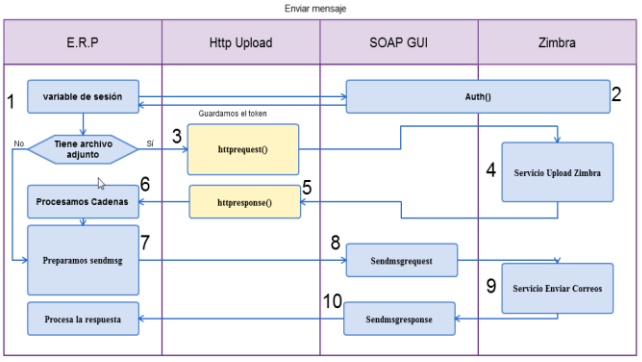


Figura 7 Enviar mensaje

petición POST vía httprequest al servidor de subida de archivos de zimbra por cada archivo adjunto, pasándole el token de sesión como cookie. Una vez hecho esto, hacemos una llamada sendmsg.
1 Se requiere una variable de sesion Token
2 Devuelve la variable de sesión
3 Httprequest Post zimbra: Hacemos una llamada http request al sistema de subida de archivos de zimbra por cada documento adjunto
4 Servicio upload zimbra: Generará un identificador único para cada documento que le hayamos enviado y lo enviará como respuesta a cada petición POST
5 httpresponse: Devuelve un código identificador único para cada archivo adjunto.
6 Procesamos cadenas: Guardamos cada cadena asociada a cada adjunto.
7 Preparamos sendmsg: Hacemos una llamada a la función sendmsg añadiendo cada uno de los códigos de los adjuntos como attach en la llamada si es que teníamos adjunto
8 Sendmsgrequest llamada soap propiamente dicha para enviar mensaje.
9 Servicio Enviar Correos: Procesa la petición sendmsg y si es válida envia los correos y genera un identificador único de mensaje.
10 Sendmsgresponse si se ha enviado con éxito devuelve un identificador único de mensaje, y guarda el mensaje en la bandeja de enviados.

5.3 Funcionalidad carpetas

Zimbra funciona por carpetas, es decir, cada apartado tiene su carpeta y cada carpeta tiene su identificador, y si es una carpeta dentro de otra, tiene un identificador de la carpeta de la que cuelga. Así como otros atributos definidos en el Anexo. Para obtener el listado de carpetas, hay una función soap qué se llama getfolders, en la que he tenido varias dificultades sobretodo porque el programa no reconocía bien la respuesta, al contener esta, tags dentro de tags con el mismo nombre como se muestra en la figura 8. Es decir, en xml al abrir un tag sea <folder> buscará un </folder> como final y claro al tener folders den-

tro de folders reconocía el final del folder interno, cómo el final del folder padre y esto daba errores. Para solventar esto, he tenido que parsear directamente la respuesta xml como si fuera un árbol, yendo primero a los hijos y hacer una función recursiva para abarcar todo el sistema de ficheros.

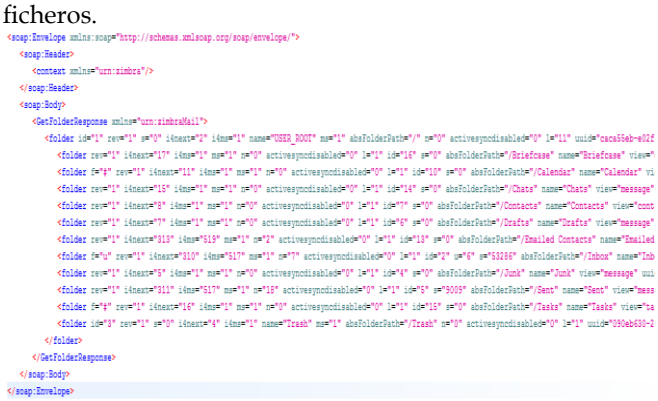


Figura 8 Ejemplo respuesta a la llamada SOAP getfolders

Cada técnico tiene sus carpetas en el zimbra, y puede querer compartirlas o no. Para eso, en la parte de técnicos he implementado una série de funcionalidades que permiten, primeramente, visualizar todas las carpetas que el técnico tiene disponibles en su cuenta, y que pueda vincular estas con el programa simplemente marcando un checkbox, y guarda la información de éstas carpetas sincronizadas en una tabla nueva de la base de datos llamada zm_folders, cada vez que se sincroniza una carpeta se inserta un registro en ella y cada vez que se quita, se elimina este.

	COLUMN_NAME	DATA_TYPE
1	ID_EMP	CHAR(3 BYTE)
2	ID_TEC	NUMBER(10,0)
3	ZM_ID	VARCHAR2(50 BYTE)
4	ZM_UUID	VARCHAR2(50 BYTE)
5	ZM_VIEW	VARCHAR2(20 BYTE)
6	ZM_NAME	VARCHAR2(128 BYTE)
7	ZM_TYPE	VARCHAR2(50 BYTE)
8	ZM_RUUID	VARCHAR2(50 BYTE)
9	ZM_RID	VARCHAR2(50 BYTE)
10	ZM_OWNER	VARCHAR2(128 BYTE)

Figura 9 tabla zm_folders

id_emp: identificador de la empresa,
zm_uuid: identificador único de la carpeta,
id_tec: identificador del tecnico,
zm_view: este campo sirve para identificar el tipo de la carpeta, es decir si es "contact", "appointment", "task", "message" o "document".
zm_name: corresponde al nombre de la carpeta,
zm_type: indica si es una carpeta tuya o es un link a una carpeta compartida.
zm_ruuid: es el identificador único de la carpeta dela que viene, (es decir si es una carpeta sincronizada donde nosotros tenemos el link, este atributo nos servirá para identificar la carpeta de la que procede de forma única).
zm_rid es el numero identificador de la carpeta dentro de la cuenta de la que proviene,
zm_owner cuenta del propietario de la carpeta de la que proviene.

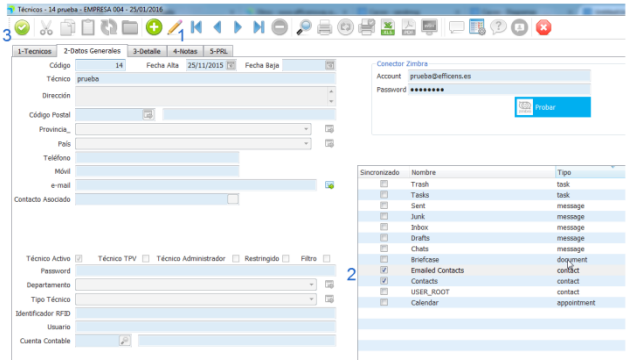


Figura 10 Ventana de técnicos modificada

En la ventana de técnicos modificada figura 10. Si procedemos a editar un técnico (1), luego podemos marcar las carpetas que queremos sincronizar (2) y grabar (3). En el backend coge la información de cada carpeta y la registra en la base de datos. Pero en el caso de que sea una carpeta de contactos, lo que haremos es pasar a la funcionalidad de contactos.
En está ventana la parte derecha es la que he añadido, es decir, dos campos para añadir cuenta y contraseña, un botón para verificar si la información es correcta y abajo una tabla donde se muestran las distintas carpetas disponibles en esa cuenta.
Para este paso, he tenido también que añadir dos campos en la tabla de técnicos del programa, con los nombres ZM_ACCOUNT y ZM_PASSWORD, uno para guardar la cuenta de zimbra y el otro para guardar su contraseña.

5.4 Funcionalidad contactos

Los contactos de la empresa son generales y están en el módulo de contactos, pero a su vez cada técnico dispone de una cuenta zimbra y dentro de ésta, tiene una série de contactos organizados en las carpetas, a veces entre técnicos tienen contactos iguales y a veces no, el propósito de esta funcionalidad es que cualquier técnico pueda ir a la parte de contactos y pueda vincular cualquier contacto en sus carpetas de contactos de zimbra. Como poder incorporar todos los contactos de una carpeta a los contactos de la empresa.
Hay dos direcciones de sincronización, es decir de Zimbra al programa y del programa a Zimbra.

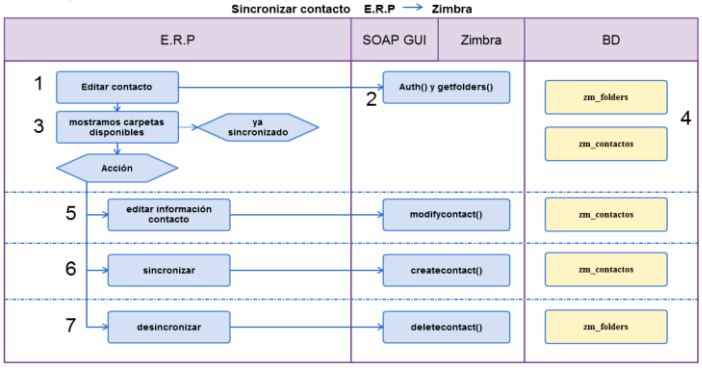


Figura 11 E.R.P -> Zimbra

1 Al entrar a editar un contacto necesitaremos tener en nuestro usuario(técnico) una cuenta de zimbra asociada, en la tabla de técnicos.
2 Actualizamos la tabla de zm_folders para comprobar que no haya carpetas nuevas.
3 Por cada carpeta disponible que mostramos, se hace una comprobacion en la base de datos (4) para ver si en la tabla zm_contactos el contacto que estamos mostrando ya está sincronizado en la carpeta, si lo está, se muestra el checkbox como seleccionado si no, aparecerá desseleccionado.
5 Si editamos la información dentro del programa, se realiza una llamada SOAP a cada cuenta donde el contacto esté sincronizado, para mantener la misma información, además se actualiza la fecha de actualización en la tabla zm_contactos.
6 Si sincronizamos un contacto, realizamos un createcontact con la información de contacto del programa hacia el Zimbra y añadimos el registro en la tabla zm_contactos
7 Si dessincronizamos un contacto se hace un 'deletecontact()' del contacto de la carpeta seleccionada.

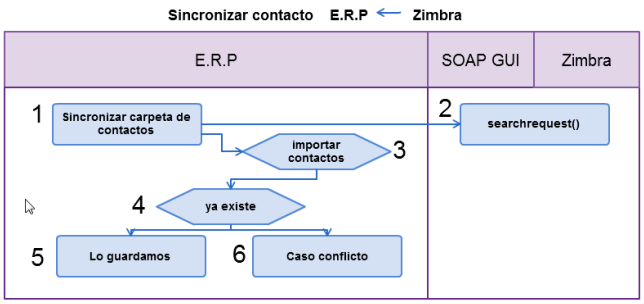


Figura 12 E.R.P <- Zimbra

Para la segunda, figura 12, si en la acción de insertar una carpeta desde técnicos, la carpeta a insertar es de contactos (1), lo que hará será hacer una llamada searchrequest(2) en la carpeta seleccionada buscando contactos, y lo que hará es comprobar cuántos contactos contiene y preguntar si desea sincronizarlos (3), en caso de si querer, comprueba si hay contactos que compartan el mismo email y/o el mismo teléfono móvil (4), si no es así, directamente crea un contacto nuevo con esta información en la tabla de contactos (5), si hay coincidencias, muestra una pantalla para cada uno de los contactos conflicto (6).

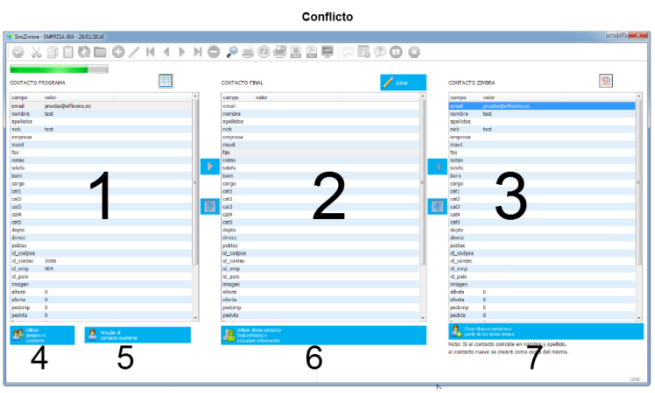


Figura 13 Caso de conflicto

En la ventana de caso de conflicto, figura 13, hay tres columnas una con la información que hay en el programa (1), otra vacía que será el resultado final(2) y otra con los datos provinientes del contacto en conflicto del zimbra(3) y hay varias opciones, como copiar todos los casos de conflicto usando siempre los datos del programa (4), utilizar los datos del programa para este caso (5), utilizar los datos provenientes del zimbra y actualizar el contacto en todas las carpetas sincronizadas (7) o bien editar el contacto y guardar también el resultado en todas las agendas sincronizadas donde este éste contacto presente (6).

	COLUMN_NAME	DATA_TYPE
1	ID_EMP	CHAR(3 BYTE)
2	ID_CONTACT	NUMBER(10,0)
3	ZM_CONTACT_ID	VARCHAR2(50 BYTE)
4	ZM_ID_FOLDER	VARCHAR2(50 BYTE)
5	ZM_TEC_FOLDER	NUMBER(10,0)
6	FECHA	DATE

Figura 14 tabla zm_contactos

id_emp: numero de empresa,
id_contact: identificador del contacto dentro del programa de la empresa,
zm_contact_id: identificador del contacto dentro dela cuenta zimbra correspondiente,
zm_id_folder: carpeta a la que pertañe,
zm_tec_folder: técnico propietario de la carpeta,
fecha: data de actualización.

5.5 Funcionalidad Calendarios

El próposito aquí se desvincula un poco de zimbra, aunque la finalidad justifique los medios.
La base reside en que el E.R.P tiene en el módulo de proyectos, visualizaciones de información ordenada similar a las visualizaciones de la información de una base de datos, es decir unas tablas con la información ordenada, pero el fin de este modulo es facilitar la gestión de los proyectos, los trabajos y las tareas en función del tiempo, para así conseguir un timeline seguible y justificable en el que cualquier usuario pueda mirar su agenda y ver qué tareas, trabajos o proyectos tiene programados. Para lograr estas visualizaciones, puedo incorporar cualquier framework externa siempre y cuando los costes esten justificados.
En concreto en el módulo de proyectos, una orden de trabajo puede tener varios registros de OT (tareas) y un proyecto puede tener varias órdenes de trabajo. En los calendarios tenemos que mostrar las distintas tareas o Registros de Ot de la tabla detordtra, para que luego podamos en cada una de estas disponer de un botón para enviar esa información a la agenda Zimbra.

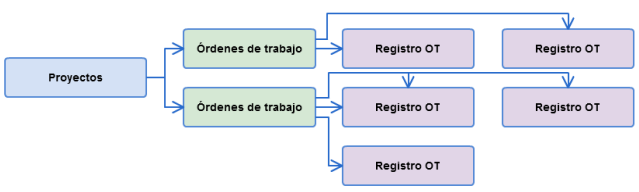


Figura 15 Jerarquía Proyectos

5.5.1 Requerimientos de las visualizaciones

Se requiere una interfaz visual que trabaje mediante javascript, que sea robusta, que esté bien documentada, que no contenga inconcluencias y que, en concreto, en esta visualización de calendarios, permita a simple vista saber cuantas y quétareas tiene asociado un técnico, en un determinado período de tiempo. Así como mover tareas entre técnicos de manera visual y simple.Y a poder ser que permita mejorar otras visualizaciones del programa en líneas futuras.

5.5.2 DHTMLX vs Kendo UI

Después de buscar distintas frameworks he acabado sólo con dos candidatas, que más o menos entran en las condiciones.

	Ventajas	Desventajas
DHTMLX	El código sigue lenguaje habitual de javascript, es decir simplemente le pasas un array como inputs y ya se muestra.	Hay detalles de visualización que disciernen un poco de las visualizaciones deseadas, como algun tirador que falta.
Kendo UI	Dispone de una amplia gama de visualización de cualquier tipo de dato. Una vez entienda como funciona, será sencillo utilizar cualquiera de ellas.	La forma de programación es estándar para cada visualizacion, pero no por ello deja de ser especifica de esta plataforma, así que el tiempo de programación augmentará.

Al final, analizando las características y las posibilidades que ofrece una y la otra, además de analizar en líneas futuras si seria más util una o la otra, la determinación es de utilizar la framework Kendo UI referencia [3].

5.5.3 Implementación

Como he comentado anteriormente, la framework funciona sobre javascript, pero con unos estándares propios, así la interacción entre E.R.P y la framework será:

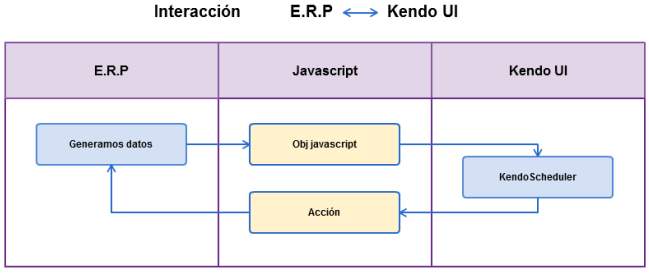


Figura 16 Interacción E.R.P <-> Kendo UI

Donde primeramente se hace una select para obtener los datos correspondientes, y estos se mandan al javascript, estos datos contienen tanto la información de los registros que se tienen que mostrar en los calendarios como la tabla de técnicos a la que corresponden dichos registros.
Ésta información la envio utilizando JSON (JavaScript Object Notation) y luego por javascript asocio cada parámetro proviniente del E.R.P a los parámetros de entrada de kendo.
Y una vez mostrada la visualización, cada movimiento que haga, cada acción que haga, supondrá una acción, que será enviada hacia el E.R.P con la información nueva, este reaccionará a esas acciones dependiendo de lo que se haya modificado y volverá a generar los datos y los mandará de nuevo hacia el Kendo.
Las acciones pueden ser desde modificar una tarea, hasta cambiar de vista tanto a nivel de visualización, como avanzar o retroceder en la línea de tiempo.

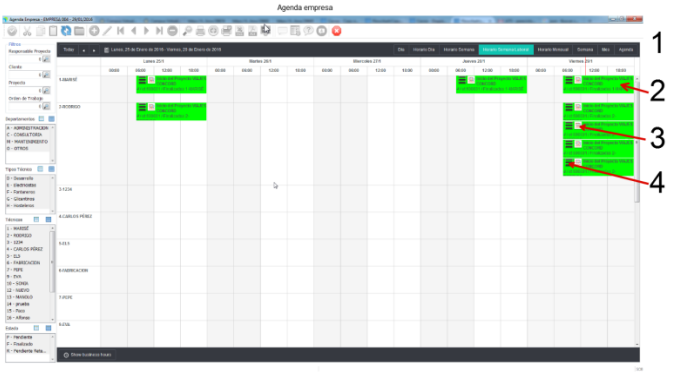


Figura 17 Visualización Calendarios

1 Selector de vistas, permite cambiar de visualización entre, mes, vista mensual por horas, semana, semana por horas, semana laboral, agenda y día.
2 La tarea con su visualización. Se muestra el nombre de la tarea y el nombre del técnico al que está asociado.
3 Icono de zimbra, botón para enviar está tarea a la visualización del zimbra del técnico asociado.
4 Opciones de la tarea, abre otra ventana del E.R.P para hacer modificaciones.

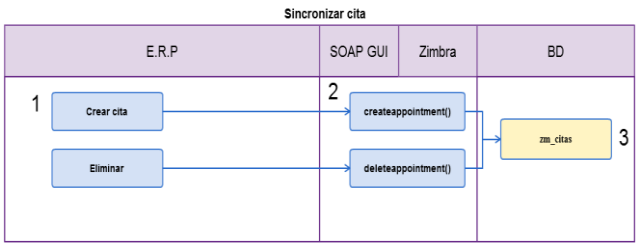


Figura 18 Sincronizar cita

5.5.4 Aplicación zimbra

Una vez pulsamos para enviar a zimbra una tarea, se crea una cita (1) con los datos que de dicha tarea y en la carpeta de calendario por defecto del técnico que tenga la tarea seleccionada asignada. También necesitamos la variable de sesión para hacer las llamadas de createappointment en caso de querer crear una cita, o la de deleteappointment, también existe la función de modifyappointment, pero en este caso preferiré eliminar y volver a crear las citas. Para esto necesito otra tabla llamada zm_citas en la que cada vez que se cree una cita nueva (2) añadiré un registro en la misma (3), añadiendo también información adicional, como si queremos que la cita sea pública en el calendario o privada, o si requiere confirmación de asis-

tencia. Además, sí una tarea está ya asignada y registrada en la tabla zm_citas, el icono del botón se mostrará de distinto color. Y si lo apretamos, eliminará tanto la cita del calendario, como de la tabla zm_citas.

	COLUMN_NAME	DATA_TYPE
1	ID_EMP	CHAR(3 BYTE)
2	ZM_ID_FOLDER	VARCHAR2(50 BYTE)
3	ZM_TEC_FOLDER	NUMBER(10,0)
4	ZM_ID_APPOINTMENT	VARCHAR2(50 BYTE)
5	ID_OT	NUMBER(10,0)
6	ID_OTREG	NUMBER(10,0)
7	ID_NODE_ID	VARCHAR2(12 BYTE)
8	ASUNTO	VARCHAR2(120 BYTE)
9	TEXTO	VARCHAR2(3000 BYTE)
10	PRIVACIDAD	CHAR(3 BYTE)
11	ORGANIZADOR	VARCHAR2(120 BYTE)
12	RESPREQ	NUMBER(5,0)
13	ALLDAY	NUMBER(5,0)
14	STATUS	CHAR(4 BYTE)
15	FECINI	DATE
16	FECFIN	DATE
17	HORINI	CHAR(50 BYTE)
18	HORFIN	CHAR(50 BYTE)

Figura 19 Tabla zm_citas

id_emp: identificador de la empresa,
zm_id_folder: identificador de la carpeta,
zm_tec_folder: identificador del técnico de la carpeta,
zm_id_appointment: identificador único de la cita dentro de la cuenta del técnico propietario del calendario,
id_ot: identificador de la ot de la que se ha sacado la información,
id_otreg: identificador del registro del que procede la información,
id_node_id: identificador del nodo único de la tarea,
asunto: asunto de la cita,
texto: texto de la cita,
privacidad: si es un evento público o privado o personalizado,
organizador: éste identificador corresponderá al tecnico que ha asignado esa tarea y que ha hecho la acción de añadirlo al zimbra desde el programa,
respreq: si la cita requiere respuesta,
allday: si la cita es de todo el dia,
status: si la cita se ha confirmado o esta pendiente de confirmar.
fecini: fecha inicial,
fecfin: fecha final,
horini: hora de inicio,
horfin: hora de finalización.

5.5.5 Extra gantt:

Aprovechando el mismo framework Kendo UI y que éste, tiene una visualización de diagramas en forma de gantt, también he añadido una opción para representar las ordenes de trabajo. De la misma manera que funciona en la otra representación, pero teniendo en cuenta que, en éste caso, no se puede modificar dicha información representada, es decir solo sirve para mostrar la información en un diagrama y no es modificable desde el mismo. Aunque si que tiene que existir la posibilidad de extraerla en formato PDF.

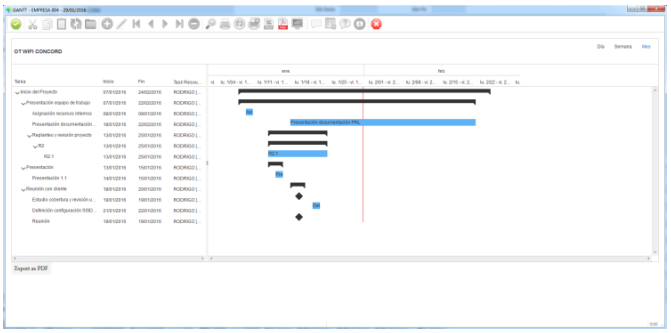


Figura 20 Extra Gantt

5.6 Bandeja de correos entrantes

Este apartado constará de una ventana nueva, tal que:

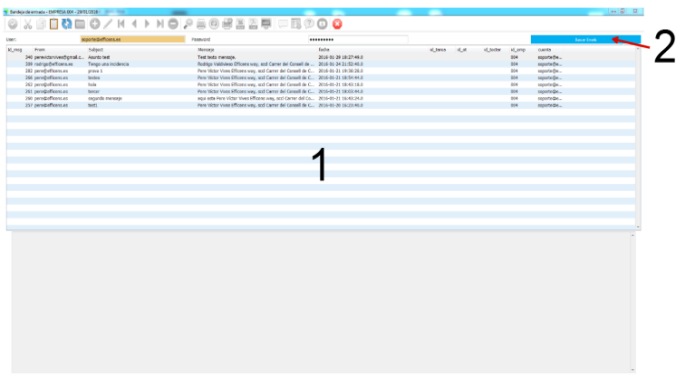


Figura 21 Bandeja de correos entrantes

Donde hay una tabla (1) que cada cierto tiempo o cuando pulsamos en la acción (2) realiza el siguiente proceso:

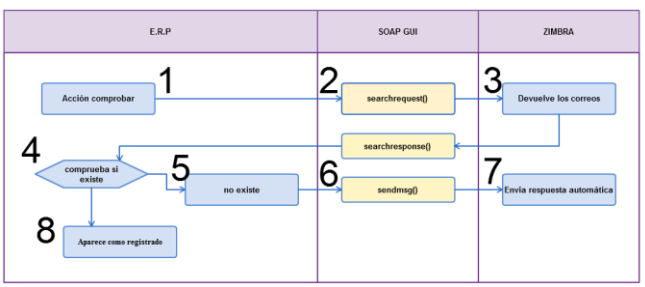


Figura 21 Funcionalidad Bandeja de correos entrantes

Cuando se lanza (1), lo primero que hace es realizar una llamada searchrequest en la bandeja de correos de entrada de la cuenta que hemos ingresado en los campos inputs de cuenta y contraseña y estos han sido válidados. Entonces está llamada devuelve un listado de los ultimos 100 correos recibidos (3), y para cada uno de ellos comprueba si está ya registrado en la tabla ticket (4), en caso de no estar registrado, primeramente, envia una respuesta automática a la dirección origen mediante una llamada

sendmsg() con los parámetros asignados (6) y zimbra envia dichas respuestas (7) una vez realizado, inserta un registro en la tabla ticket con el mensaje recibido para evitar que se envien respuestas más de una vez. Y en caso de que si que estuviese registrado simplemente lo muestra.
Al final obtenemos una tabla donde podemos observar todos los correos que nos han enviado, y que han sido correctamente respondidos, con el mensaje automático.

	COLUMN_NAME	DATA_TYPE
1	ID_EMP	CHAR(3 BYTE)
2	ID_MSG	NUMBER(10,0)
3	ID_FROM	VARCHAR2(120 BYTE)
4	SUBJECT	VARCHAR2(255 BYTE)
5	FECTIC	TIMESTAMP(1)
6	MESSAGE	VARCHAR2(1000 BYTE)
7	ID_OT	NUMBER(10,0)
8	ID_TAREA	NUMBER(10,0)
9	ID_TECTAR	NUMBER(10,0)
10	CUENTA	VARCHAR2(120 BYTE)

Figura 22 Tabla zm_ticket

id_emp: identificador de empresa.
id_msg: identificador único del mensaje en la cuenta donde se ha recibido.
id_from: correo origen del mensaje.
subject: asunto del mensaje.
fectic: fecha de recepción de mensaje.
message: mensaje.
id_ot, id_tarea, id_tectar: servirán para en líneas futuras asignar directamente un correo como un registro.
cuenta: cuenta que recibe el mensaje.

6 RESULTADOS

6.1 Tecnologias utilizadas:

He aprendido a mejorar conocimientos en varias herramientas que para mi eran o bien completamente desconocidas o simplemente que no había trabajado demasiado con ellas:

- Genero: Como he comentado anteriormente, es principalmente con la herramienta que he trabajado.
- SQL: Genero utiliza este lenguaje para accede a los datos de las bases de datos.
- Oracle: Especificamente la base de datos utilizada es Oracle 11 y las definiciones de los tipos de los atributos son de este tipo.
- Soap: Protocolo utilizado para la comunicación entre Zimbra y el E.R.P.
- Soap UI: Programa utilizado para hacer pruebas directamente contra el SOAP de zimbra.
- Zimbra: plataforma de mailing, utilizada además de ser la finalidad del trabajo, también la he utilizado para emular las llamadas que hace para cada acción.
- Javascript: Lenguaje utilizado para manejar datos a nivel de navegador cliente.
- Framework Kendo UI: framework javascript utilizada.
- Json: Objetos utilizados para la transmisión de información entre el E.R.P y el javascript.
- Html: Evidentemente es el destinatario final de utilizer javascript.
- Css: Utilizado para ajustar colores, tamaños y algunas personalizaciones como el tamaño de la letra en cada caso.
- Ftp: Servicio utilizado para enviar los documentos html y javascript al servidor cloud de el E.R.P
- Sshfs: Programa que permite establecer una conexión ssh directamente a un servidor cloud de forma segura.
- Http: protocolo utilizado para enviar documentos via post al servidor de upload de zimbra.
- Wireshark: utilizado para poder ver como eran exactamente las llamadas Post hechas al servidor upload de Zimbra.
- Xml: SOAP funciona via xml, así que ha sido básico en este proyecto.
- Cacoo: Plataforma online que permite hacer diagramas.
- Vi: editor de texto utilizado para editar archivos dentro de servidores Linux.

6.2 Que suponen estas mejoras para un usuario final.

Para un usuario del E.R.P estas mejoras suponen agilizar en parte su tiempo dedicado a tareas que realmente no eran propias de su trabajo, monótonas y que, a la larga, invertían una cantidad de tiempo considerable en ellas.
En líneas generales las mejoras incorporadas són:

- El tiempo invertido en enviar una factura por email, que antes oscilaba entre uno y dos minutos, pasa a ser instantáneo. Es decir, si inviertía una media de dos horas al día para enviar todas las facturas al mes le suponían unas 62 horas de trabajo extras, ahora pasa a ser 5 minutos al día lo que serían unas 3 horas al mes.
- El tiempo invertido en crear un contacto, y pasarlo luego, pasa a ser prácticamente nulo, porque tan solo desvinculando una carpeta de contactos y volviéndola a añadir, marcando la opción de utilizar siempre la información de los datos del contacto, ya puedes tener uno, veinte, o cien contactos nuevos, que el tiempo necesario será prácticamente instantaneo. Pasa de invertir 31 horas al mantenimiento, a invertir unas 5 horas.
- El módulo de proyectos presenta una visualización más intuitiva y fácil de entender, para agilizar el proceso desde que se mira una información hasta que la comprendes.
- Las tareas son asignables directamente, sin tener que salir del programa.

- Ahora el E.R.P dispone de una bandeja de entrada que podría ser fácilmente adaptada para conseguir un ticketin completo. Pero la primera piedra ya está fijada para línias futuras.

6.3 Eficacia de las funcionalidades añadidas

Las funcionalidades añadidas son robustas y carecen de errores, hay algunas visualizaciones de las ventanas del programa que podrían mejorarse. Pero en líneas generales són eficientes y enlazan bien con la dinámica de las funciones del propio programa E.R.P. Además, que he intentado minimizar en todo momento la cantidad de espacio de base de datos a utilizar, así cómo simplificando el código lo máximo posible.

7 Conclusiones

A lo largo de este trabajo me he encontrado con una gran cantidad de problemas que hasta que no te enfrentas a un proyecto de estas características no puedes ni llegar a comprender. Pero aún así me ha sorprendido como han evolucionado mis capacidades de buscarme la vida. Y para mi está ha sido una muy buena experiencia de la que he aprendido mucho y que me ha ayudado a sobretodo valorar la complejidad de la informática, de algo que a priori a la vista de la gente normal, parece algo simple de realizar, que por detrás haya tantas y tantas decisiones que se han tenido que tomar para lograr que esa representación de la información que acaban siendo los programas en si, sean de esa manera y no de otra.

También he aprendido a comprender que no todas las formas de resolver los problemas, son adecuadas, es decir, que a veces planteamos soluciones atípicas, y el hecho de poder discernir entre qué idea es buena y qué idea no lo es tanto, es algo que me ha costado bastante, pero creo que he podido mejorar un poco en ese aspecto.

Además, estoy muy agradecido al poder realizar un proyecto propio de la ingeniería del software de estas características, porque considero que haber trabajado con muchas herramientas distintas me da un mejor abanico de posibilidades de ahora en adelante para cuando se planteen problemas de las mismas características salir airoso de ellos de las mejores formas posibles.

También me ha servido para poder aplicar en cierta manera mucha parte de la teoría de la carrera para un propósito, es decir, aplicar toda la información adquirida para un fin, que realmente es lo que a nivel personal me gusta de la informática, programar para una finalidad.

He aprendido además a estructurar mejor mis ideas y a poder transformarlas mejor en código.

Y en líneas generales puedo decir que hacer este proyecto en una empresa me ha servido para abrir más los ojos y a madurar en la medida de lo posible cómo informatico propiamente dicho, que también entiendo que es la finalidad de este proyecto.

8 Agradecimientos

Agradezco la ayuda a la realización de este trabajo sobretodo a Rodrigo Valdivieso, por ayudarme, ya sea con consejos referentes a qué partes de lo que hacía eran funciones necesarias y cuales no, y sobretodo por ayudarme a comprender cómo funcionan los clientes reales del día a día, que lo fundamental es que cuánto más simple y eficiente sea cualquier implementación mejor, y a que, tenemos que valorar más nuestro trabajo, porque a veces no le damos la importancia ni el valor real que merece. Y también mencionar a los demás miembros de Efficens.

Agradezco también a mi tutor, Fernando Vilariño ayudarme a estructurar mis ideas y a poder plasmar el trabajo realizado en un documento, que tengo que decir que no es nada fàcil, una vez tienes unas implementaciones hechas, plasmar realmente lo que hacen y sobretodo el intentar que personas de fuera del ámbito de la informática puedan llegar a comprender ni que sea en líneas generales las líneas básicas de este proyecto.

9 Bibliografia

[1] Genero Studio User Guide 3.00 url: *http://www.4js.com/online_documentation/fjs-gst-manual-html/?path=fjs-gst-manual* última visita: 1 February 2016.

[2] Zimbra SOAP API Reference 8.0.7_GA_6020 url: *https://files.zimbra.com/docs/soap_api/8.0.7/soap-docs-807/api-reference/index.html* última visita: 1 February 2016.

[3] Kendo UI html5 framework – Telerik url: *http://www.telerik.com/kendo-ui-html5-framework* última visita: 1 February 2016.

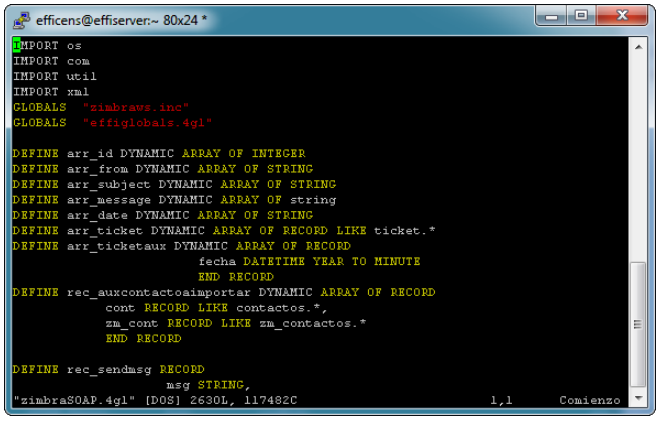
[4] EfficensWay url: *www.efficens.es* última visita: 1 February 2016.

[5] Zimbra Efficens url: *mail.efficens.es* última visita: 1 February 2016.

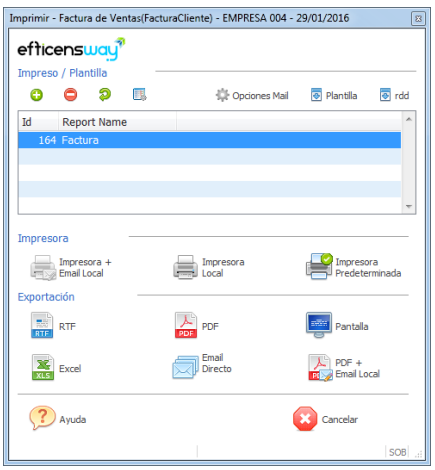
APÉNDICE

A1. EJEMPLOS

Ejemplo de ventana de programación utilizando vi, por terminal.



A2. ENVIAR EMAIL



Email directo, se enviará directamente la factura al cliente via email.

A3. SOAP llamadas

Auth:

Funcionalidad que permite loguear y devuelve una cadena de carácteres variable de sesión.
zm_AuthRequest(in_zm_account,in_zm_password)

SendMsg:

Funcionalidad que permite el enviamiento de mensajes.
zm_sendmsg(token,mailorigen,etiquetaorigen,msg,asunto,maildestino,etiquetadestino,ccmaildestino,bccmaildestino,arr_adjunto)

UploadAttachment

Función no de la api que consiste en hacer una llamada httprequest POST hacia el servidor dónde se guardan los archivos adjuntos de zimbra.
zm_uploadattachment(token,filename)

GetMsg

Funcionalidad que permite ver los detalles de un mensaje del que ya tienes el identificador.
zm_getmsg(token,msgid)

GetFolder

Si no le especificas una carpeta concreta, te devuelve todo el arbol de carpetas que tengas dentro del mail. Y si lo haces te devuelve el tipo de carpeta que es.
zm_getfolders(out_authtoken)

Search

Funcionalidad que permite buscar varios tipos de información dentro del zimbra tanto si són conversaciones, mensajes, contactos, citas de la agenda, tareas, documentos, como devolver los contactos de una carpeta en concreto. Cómo tiene varias funcionalidades finales hay varias llamadas personalizadas de esta función:
searchcontactsfromnamefolder(token,namefolder)

CreateAppointment

Funcionalidad para crear citas en los calendarios.
zm_createappointment(token,l,fnc_start,fnc_end,asunto,privacitat,a,respreq)

ModifyAppointment

Sirve para modificar.
zm_modifyappointment(token,id,l,fnc_start,fnc_end,asunto,privacitat,a,respreq)

CancelAppointment

Permite cancelar una cita del calendario.
zm_cancelappointment(token,id)

CreateContact

Permite crear un contacto.
zm_createcontact(token,folder,nombre,email,cognom,profes-
sio,empresa,mobilephone,workphone,homestreet,homecity,homestate,homepostalcode,homecountry,homeurl,anniversary,notes,img)

ModifyContact

Permite modificar un contacto.
zm_modifycontact(token,contactid,replace,folder,nombre,email,cognom,professio,empresa,mobilephone,workphone,homestreet,homecity,homestate,homepostalcode,homecountry,homeurl,anniversary,notes,adjunto).