Lo primero que hace es construir un nuevo objeto Conexión. El proceso lo hemos descrito anteriormente por lo que no me extenderé. Una vez se tiene el objeto Conexión intenta crear un objeto SingletonLocal que como ya hemos comentado es la implementación de la entidad Almacén_Local.

Este objeto también implementa el patrón Singleton ya que es un objeto que contiene todas las listas locales de la aplicación y que por tanto si existieran varias instancias de el su tamaño en memoria sería muy grande.

Este proceso devuelve una nueva instancia del objeto SingletonLocal si no estaba creado.

Por último, se actualizan los atributos del objeto Singleton con las instancias devueltas por los procesos Conexión y getInstance y se devuelve el objeto Singleton a la aplicación para que ésta lo utilice.

Flujos Alternativos:

En caso de que el objeto Singleton o el objeto SingletonLocal ya estén creados no se vuelven a crear, sino que se devuelve un puntero a la instancia existente del objeto.

De este modo se garantiza una única conexión por aplicación así como una única instancia del objeto SingletonLocal en memoria.

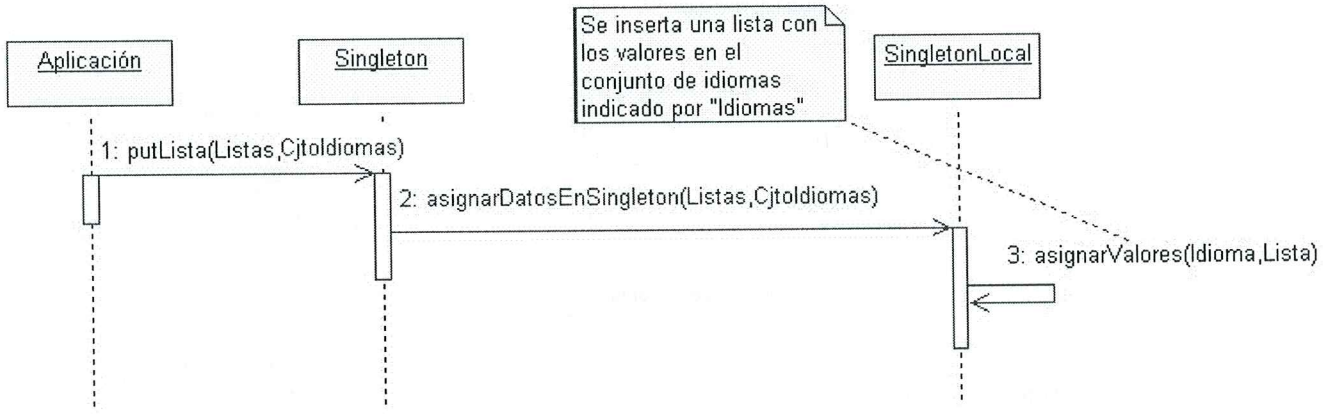
Casos de Error:

En caso de que no seamos capaces de conectar con el Servidor o no podamos obtener una instancia del objeto SingletonLocal, se registra este evento en el log y se propaga la excepción a las capas superiores para que la aplicación actúe en consecuencia.

Proceso asignarDatosEnSingleton(Lista, Identificador, Idiomas)

Este proceso es invocado por la funcionalidad putLista(Lista,Idioma) del objeto Singleton forma parte del proceso de cargaInicial del cliente.

Este proceso es el responsable de asignar las listas propias de la aplicación al objeto SingletonLocal que al final al igual que la entidad Almacen_Servidor del servidor es una Hashtable.



Descripción: Una aplicación desea almacenar una lista propia en el Almacen_Local para ello invoca el proceso put del objeto Singleton (del que previamente tiene que haber obtenido una instancia).

El proceso put invoca al proceso asignarDatoEnSingleton(Listas,CjtoIdiomas) del objeto SingletonLocal que como ya he comentado antes, es la implementación de la entidad Almacén_Local.

La aplicación ha pasado como atributos una linkedHashmap que contiene la lista a almacenar en los diferentes idiomas indicados por el atributo Idiomas.

El proceso asignarDatosEnSingleton recibe esos parámetros y los almacena igual que lo hace el Servidor cuando almacena las listas públicas y manteniendo la misma estructura.

Flujos Alternativos:

Este proceso no tiene flujos alternativos

Casos de Error:

En caso de que suceda algún error intentando almacenar las listas lo que se hace es registrarlo en el log de la aplicación y continuar con la ejecución.

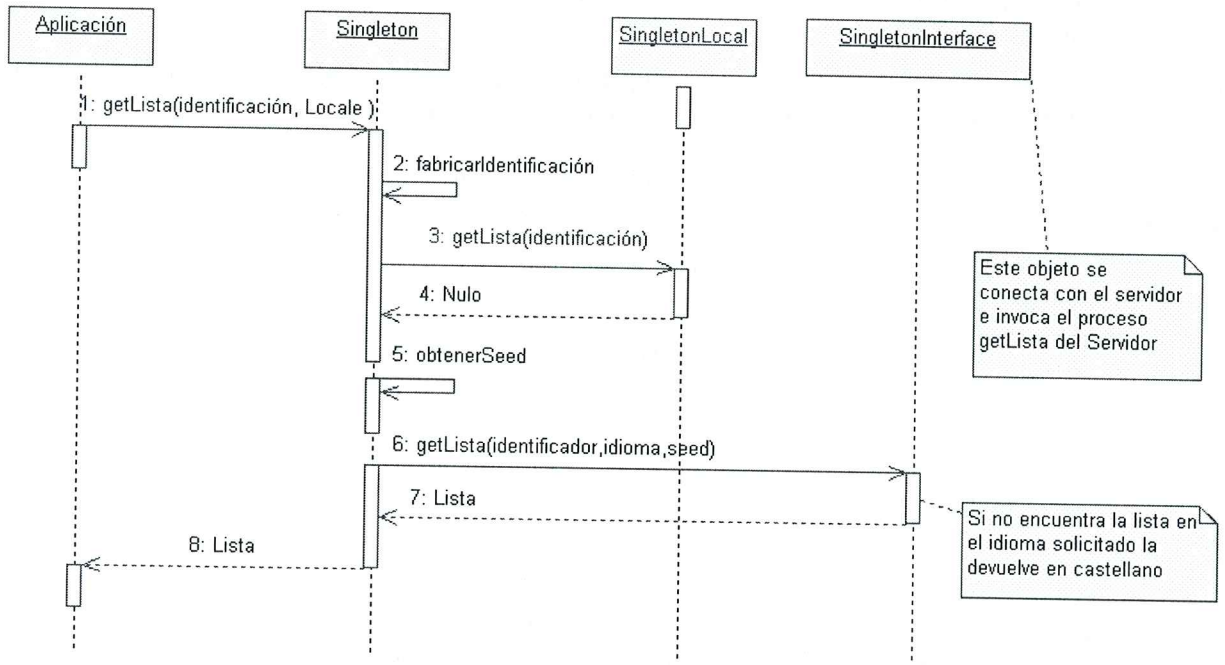
Proceso eliminarLista(Identificador):

Este proceso es exactamente igual que putLista pero eliminando la lista indicada por el identificador para todos sus idiomas en vez de añadirla.

Todos los casos de error y los flujos son iguales.

Proceso getLista(Identificación,Idioma)

Este es el proceso más utilizado por las distintas aplicaciones que utilizan esta librería. Está implementado en el objeto Singleton y implementa el subsistema Obtención Listas del subsistema Cliente.



Descripción: Este proceso es invocado por una aplicación que desea obtener una lista (ya sea una pública o bien una cargada localmente por la propia aplicación).

Se invoca el proceso getLista del objeto Singleton (del cual se ha de obtener una instancia antes). Este proceso construirá el identificador de la lista concatenando el parámetro identificación con el idioma de la locale e invocará con este nuevo identificador al proceso getLista del objeto SingletonLocal. Este objeto contiene todas las listas almacenadas en una hash por lo que devolverá si existe el objeto asociado al identificador que recibe.

En caso de que no exista la lista en local se obtiene el parámetro de seguridad Seed del que ya hemos hablado, y se invoca al proceso remoto getLista del servidor a través del objeto SingletonInterface del cual hemos obtenido una instancia al obtener una instancia del objeto Singleton.

Este proceso ya lo hemos comentado antes y por tanto no lo repetiré, pero en todo caso me gustaría comentar que devuelve vía RMI la lista solicitada en el idioma solicitado.

El objeto Singleton devolverá esta lista a la aplicación.

Flujos Alternativos:

1. ***La lista se encuentra en el objeto SingletonLocal.***

En ese caso se devuelve inmediatamente la lista obtenida del objeto SingletonLocal sin necesidad de "molestar" al Servidor haciendo el proceso prácticamente instantáneo.

2. ***La lista no se encuentra en local ni se encuentra en el idioma solicitado en el Servidor***

En ese caso y tal y como comentamos al explicar el proceso getLista del subsistema Servidor se intenta devolver la lista en Castellano que es el idioma por defecto de ESADE.

Casos de Error:

1. ***La conexión con el servidor nos de una excepción del tipo "ConnexionRefused"***

En ese caso la librería Cliente interpretará que la instancia de conexión que posee ya no es válida e intentará reconectar.

Este tipo de excepciones se producen cuando se rearranca el sistema Servidor o cuando por cualquier motivo el servidor de aplicaciones debe ser reiniciado.

En ese caso la el proceso getLista del objeto Singleton invocará al proceso Reconexión que explicaremos más adelante y volverá a intentar obtener la lista del Servidor.

Si la excepción se repite re registrará en el log y se propagará a las capas superiores para notificarlo a la Aplicación.

2. ***Surge una excepción de cualquier otro tipo:***

Se registra el evento y se propaga la excepción para que la Aplicación que ha invocado el proceso sea consciente y actúe en consecuencia.

Proceso getLista(Identificador)

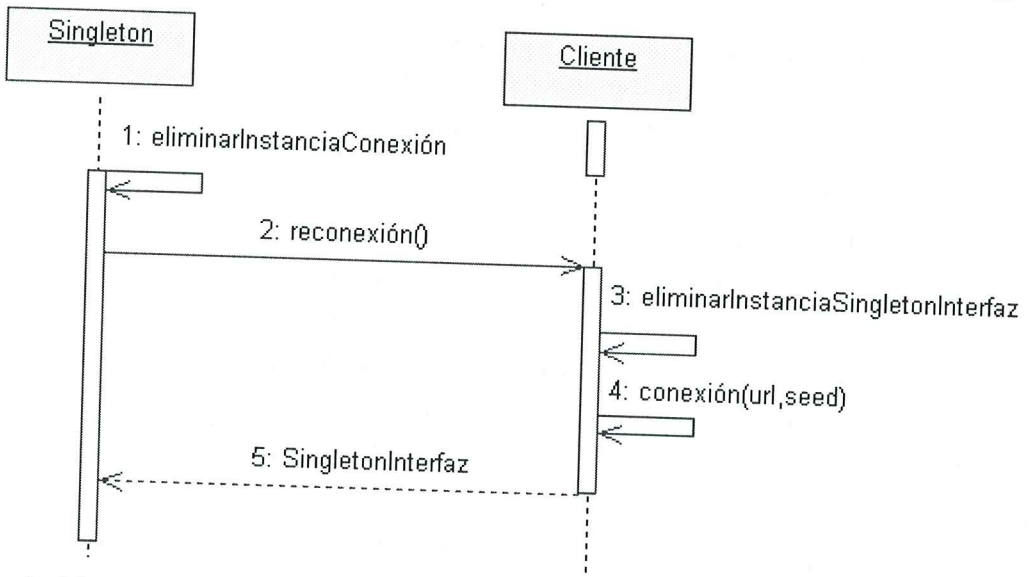
Este proceso es exactamente igual que el proceso anterior con las variantes de que el identificador es el que utilizará el proceso getLista del SingletonLocal y no se ha de concatenar con el idioma de la Locale.

También varía el hecho de que se invoca al proceso getLista(identificador, seed) del subsistema Servidor que devuelve la lista en el único idioma que la posee y del cual ya hemos hablado antes.

Proceso reconexión

Este proceso es muy importante y surge para solventar el problema de que al reiniciar el subsistema Servidor todas las aplicaciones que estaban conectadas a él dejaban de funcionar y debían ser reiniciadas, al ser incapaces de obtener una nueva instancia válida del objeto SingletonInterfaz.

Para automatizar la acción de reconexión y minimizar el número de excepciones que se propagan a las capas superiores, este proceso es invocado automáticamente por cualquier proceso de la librería cliente que reciba la excepción "ConnexionRefused" es decir los dos procesos getLista() y el proceso recargarLista del que aún no hemos hablado lo invocarán en caso de que el primer intento de conexión sea fallido.



Descripción: Un proceso de la librería Cliente ha obtenido una excepción "ConnectionRefused" intentando acceder al subsistema Servidor.

Se invoca el proceso reconexión en el objeto Singleton. Éste elimina la instancia que tiene almacenada del objeto Cliente e invoca al proceso reconexión del Cliente.

Este proceso lo primero que hace es eliminar la instancia del objeto SingletonInterface que posee el Cliente y volver a realizar el proceso conexión del que ya hemos hablado anteriormente.

El objeto SingletonInterfaz obtenido es devuelto al objeto Singleton de la librería que lo utilizará para comunicarse con el Servidor.

Flujos Alternativos:

Este proceso no tiene flujos alternativos.

Casos de Error:

En caso de que surja cualquier error se almacenará en el log y se propagará a las capas superiores para que la aplicación lo detecte.

Ya para finalizar existe el equivalente al proceso recargaAdministrador implementado por la action RecargaAdministradorAction.

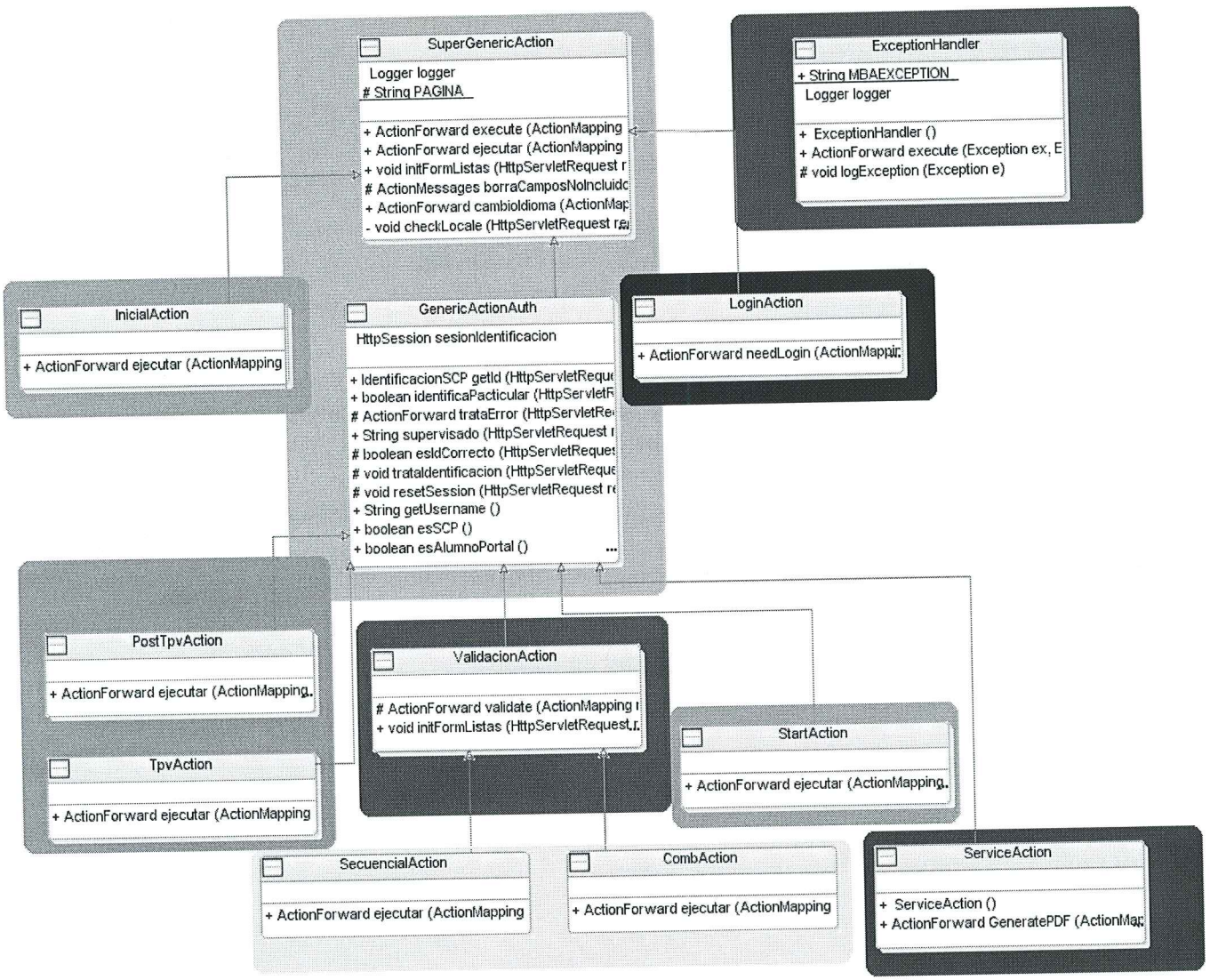
Este proceso puede ser invocado por el cliente en cualquier momento a través de una url y fuerza la recarga automática de las listas locales de la aplicación.

El flujo es exactamente el mismo que para su equivalente en el subsistema Servidor y por tanto no lo repetiremos.

7.2 Definición de los distintos componentes del sistema.

7.2.1 Definición de los distintos componentes del SAW:

Para intentar que los componentes sean lo más claros posibles he separado por colores las diferentes actions del SAW, en función del componente del que forman parte. En ESADE se acostumbra a desarrollar los diferentes subsistemas completos y por orden de prioridad, de manera que esta misma imagen nos servirá más adelante para decidir el orden de desarrollo de los diferentes componentes.



Leyenda:

Librería genericActions independiente de la aplicación y responsable del cambio de idioma	
Componente ClientePagoVirutal	
Componente GestionNavegabilidad	
Componente Validación	
Componente GeneraciónDocumentos	
Componente ProcesoAutentificación	
Componente GestionLogs	
Componente ConstrucciónFormulario	

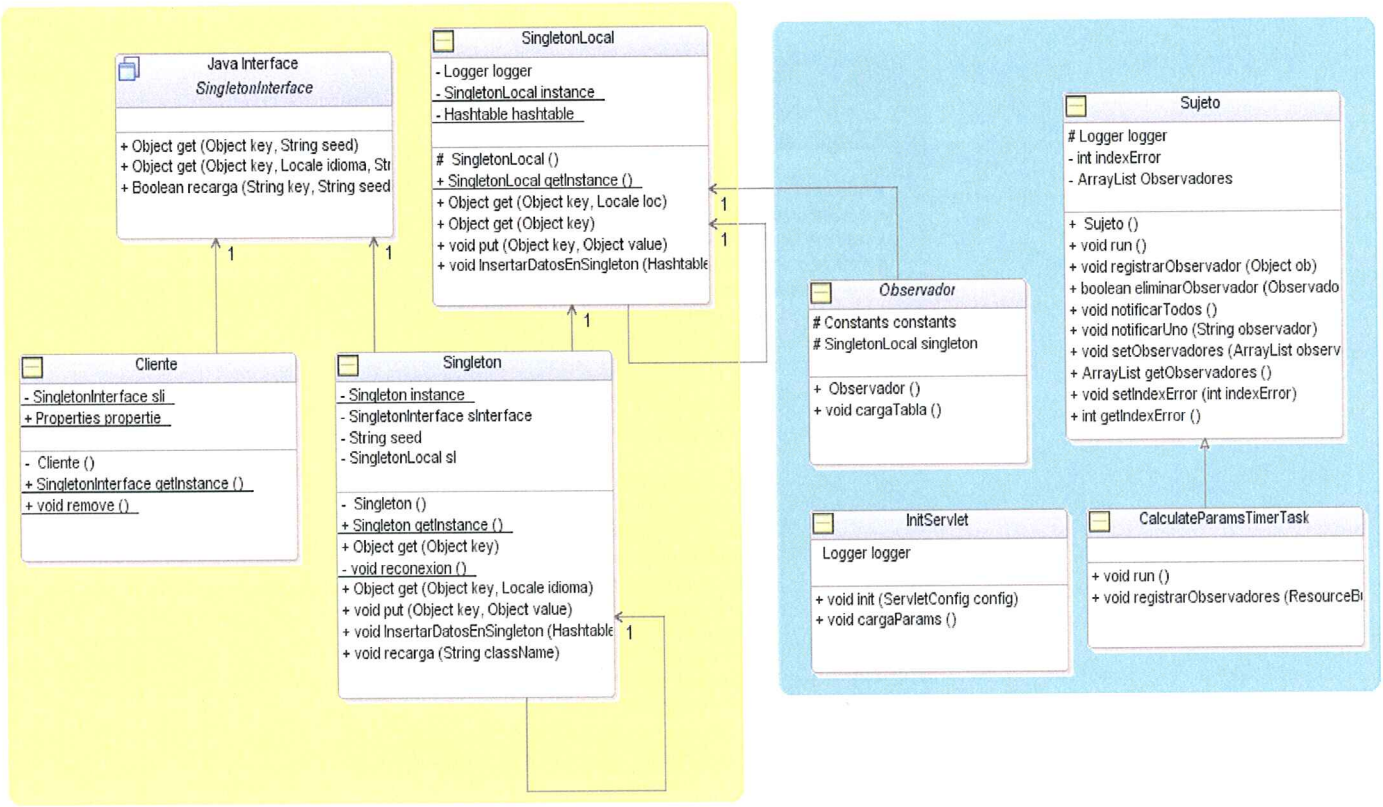
En el dibujo anterior solamente hemos visto la parte de la capa de presentación pero evidentemente cada una de estas actions utiliza sus respectivos BO que a su vez utilizan objetos DAO que a su vez utilizan objetos Model.

Pues bien, entenderemos que un objeto forma parte de un componente del sistema y por tanto se ha de desarrollar conjuntamente con el resto de miembros de ese componente, todo aquel objeto que sea utilizado por la action/s integrantes de dicho componente o bien por algún objeto utilizado por esa action.

7.2.2 Definición de los distintos componentes del Singletonserver:

Para realizar este apartado como siempre hemos dividido la aplicación Singletonserver en los subsistemas *Cliente y Servidor.*

Los componentes del *subsistema Cliente* son:



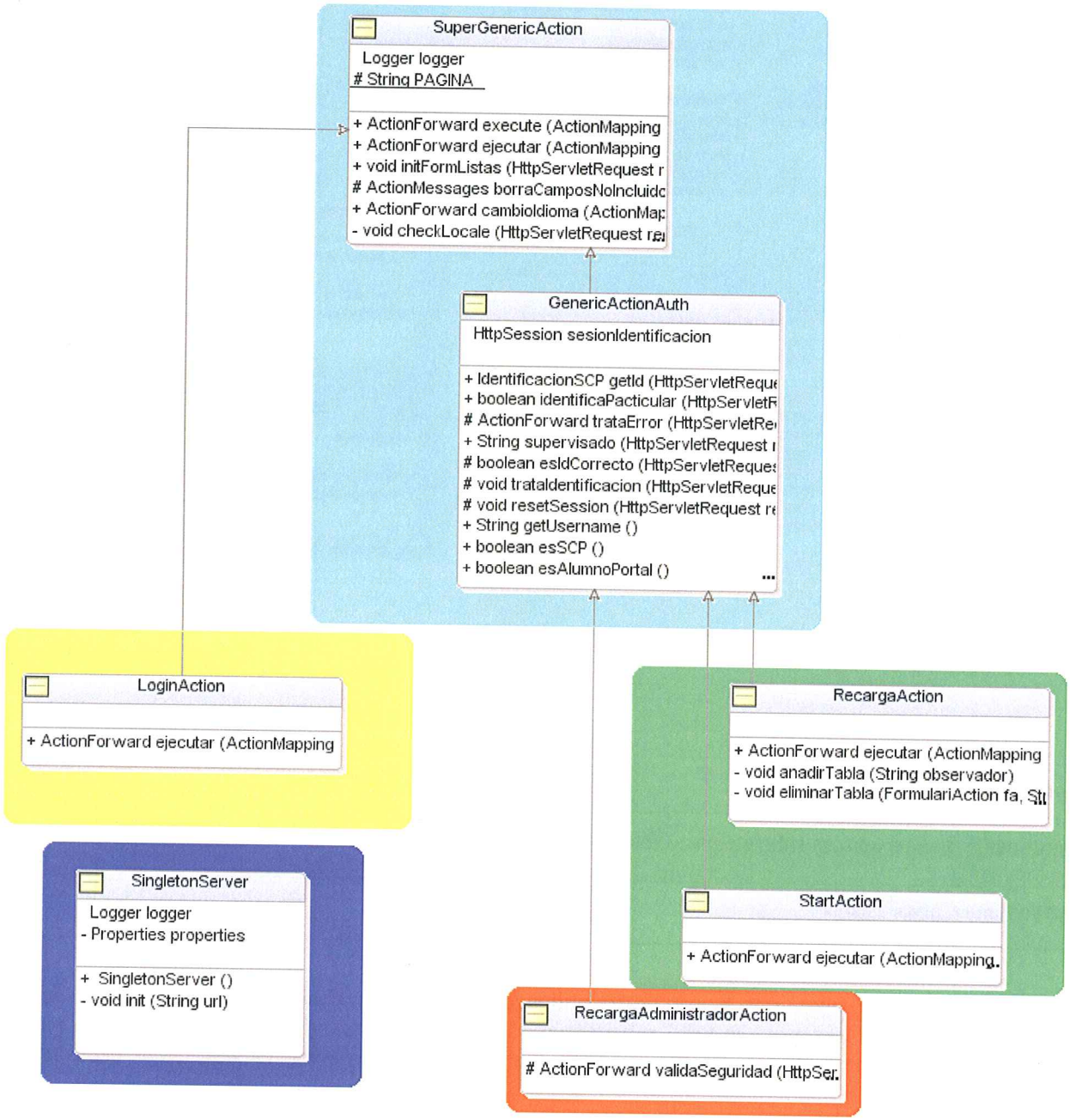
Leyenda:

Componente que agrupa los subsistemas: ObtenciónConexión/ ObtenciónListas/ RecargarListas	
Componente Cargar Listas	

Como se puede comprobar, nos hemos visto obligados a agrupar muchos subsistemas en un único componente principalmente por que los 3 subsistemas están muy ligados entre sí y el desarrollo de partes de uno era necesario para el otro, etc. Por ejemplo, tanto obtenerListas como recargarListas necesitaban ObtenciónConexion.

En cuanto al subsistema CargarLista, es igual tanto para el servidor como para el cliente por lo que se desarrollará una vez y se compartirá.

Subsistema servidor:



Me gustaría comentar que al igual que en el SAW, este diagrama solo muestra las actions pero consideraremos que forma parte de este componente cualquier objeto que sea utilizado por una de estas actions o por algún objeto utilizado por un action.

Librería genericActions independiente de la aplicación y responsable de la seguridad	
Componente que agrupa los subsistemas obtenerListaRemota/ recargarListaRemota /eliminarListaRemota	
Componente que forma parte de la interfaz web y que agrupa el ProcesoAutentificación	

Componente que agrupa lo elementos del proceso RecargaAdministrador	
Componente que agrupa los elementos del proceso servidorRMI	

Me gustaría destacar que en la parte del Servidor no se han contemplado en el diagrama ni los más de 30 observadores ni los casi 20 objetos BO con sus respectivos DAO.

Como ya he comentado antes, el proceso cargarListas es común para la parte cliente y Servidor pero en la parte Servidor éste consta además de toda la lógica necesaria para cargar todas las listas comunes.

El conjunto de clases Observador con los identificadores de las listas de las que son responsables son los siguientes:

CargaAgrupacionesEstudios	lista_agrupaciones_estudios_
CargaAreasFuncCargos	areaFuncional.codigo
CargaAreasFuncionales	lista_areas_func_
CargaAreasFuncionalesEq	lista_areas_func_eq_
CargaCargos	CargosEmpresa.codigo
CargaConocimientosIdiomas	conocimientosIdiomas
CargaEstadoCivil	lista_estado_civil
CargaEstadosEntrevistas	lista_estados_entrevista_
CargaEstudiosCodigoMba	lista_estudios_codigo_mba
CargaEstudiosIdi	listaEstudios
CargaEstudiosMba	lista_estudios_mba_
CargaGentilicios	lista_gentilicios_
CargaGentiliciosBusquedaCv	lista_gentilicios_busqueda_cv_
CargaGentiliciosPaises	lista_gentilicios_paises
CargaGradAc	lista_grad_ac_
CargaIdentificadores	identificador.codigo
CargaIdiomas	lista_idiomas_mba_
CargaIdiomasBusquedaCv	lista_idiomas_busqueda_cv_
CargaIdiomasDatosMba	lista_idiomas_datos_mba_
CargaInternMba	lista_inten_mba_
CargaNivelesOrg	nivelOrganizativo.codigo
CargaPaisesMba	lista_paises_mba_
CargaProgramasMba	lista_programas_mba_
CargaProgramasMbaOferta	lista_programas_mba_ofertas
CargaProgramasMbaPrimavera	lista_programas_mba_primavera

CargaProgramasMbaVerano	lista_programas_mba_verano
CargaSectores	lista_sectores_
CargaSexo	lista_sexo
CargaTipoEmpresa	tipoEmpresa
CargaTiposAntiguedad	lista_tipos_antiguedad_
CargaTiposExpediente	lista_tipos_expediente_
CargaTiposJornada	lista_tipos_jornada_
CargaUniDobleTit	lista_uni_doble_tit
CargaUniIntMba	lista_uni_int_mba
CargaZonasGeo	lista_zonas_geo_
CargaZonasGeoCv	lista_zonas_geo_cv_
CargaMenusGestion	lista_menus_

7.3 Especificar el entorno tecnológico del Sistema:

Para los dos sistemas desarrollados, el entorno tecnológico ha sido el mismo por lo que lo describiré conjuntamente.

En ESADE todas las aplicaciones se desarrollan en el mismo entorno o procurando que éste varíe lo mínimo posible para favorecer la homogeneidad de las diferentes aplicaciones.

El entorno de desarrollo es siempre JDeveloper en su versión 1.3.

Actualmente existe la versión 1.3.1 pero la empresa aún no se ha decidido a migrar ya que el esquema de las carpetas varía y los proyectos abiertos con esta versión del Developer dejan de ser compatibles con la versión anterior.

La versión del java que se utiliza tanto para desarrollar como para ejecutar, es 1.4 puesto que el servidor de aplicaciones corre con esta versión y generaría problemas el desarrollar con la versión 1.5

En cuanto a la base de datos y las comunicaciones con la misma, se usa Oracle 9 que se comunica con Hibernate en su versión 3.1 utilizando el controlador JDBC que trae este por defecto.

Para finalizar, las dos aplicaciones se ejecutan en un servidor de aplicaciones de Oracle en lo que se conoce como Contenedor_OC4J

La aplicación Singletonserver tiene un contenedor para ella sola debido a lo crítico que sería un fallo en esta aplicación, mientras que SAW se encuentra "conviviendo" con otras dos aplicaciones del Departamento de Admisiones.

El hecho de estar en un contenedor para ella sola o compartirlo, importa, ya que se pueden definir una serie de permisos, recursos, logs, etc.. a nivel de contenedor que dan flexibilizad al rendimiento de la aplicación y permiten optimizar los recursos destinados a ella.

En cuanto al Hardware Utilizado, las máquinas de desarrollo son pcs Dell con un procesador Pentium 4 HT y 1 GB de RAM.

El servidor es un servidor Dell, también con varios procesadores y varios Gigas de RAM, pero en nuestro caso no nos influye ya que la aplicación solo verá los recursos asignados por el contenedor que en el caso del Singleton son :

500Mb de memoria, una única conexión a la base de datos e ilimitadas conexiones RMI

En cuanto al SAW dispone de 500Mb de memoria también pero se le permiten hasta 20 conexiones simultáneas con la base de datos.

7.4 Diseño del plan de pruebas

En ESADE hasta hace poco no existía un modelo estándar para validar la corrección o no de los diferentes componentes de un sistema.

Lo que se hacía era que el propio programador o un usuario final, era el responsable de ir probando los diferentes componentes del sistema a medida que se iban desarrollando.

De un tiempo a esta parte se está intentando implantar el modelo de pruebas de "Cactus" para aplicaciones hechas con Struts.

Como no es mi caso, he de decir que no se definió un plan de pruebas para ninguna de las dos aplicaciones y que tal y como he comentado antes, lo que se hacía era ir entregando componentes al usuario para que los probara y poder de este modo ir detectando fallos a la vez que se iban resolviendo detalles que quizá en los prototipos de las pantallas le gustaban al usuario mientras que después en las pruebas decidía que no eran de su agrado.

He de comentar que este proceso lo encontré lento y poco cuidado ya que en muchos casos el usuario acababa por probar una misma pantalla mas de 3 veces dándose una mala imagen del Departamento de Programación.

También he de decir, que a pesar de lo dicho y de que se está implantando utilizar "testCase" con cactus, los usuarios reclaman poder probar las diferentes pantallas antes de recibir el producto final, precisamente para poder solicitar los ya mencionados cambios en el diseño.

8 Construcción de subsistemas (Fase 3)

En esta fase existe una primera parte llamada "Preparación del entorno de Desarrollo,pruebas y procedimientos de operación" que no existió para ninguna de las dos aplicaciones que realicé, ya que el entorno de trabajo ya estaba preparado.

No fue necesario ni instalar la base de datos, ni el servidor de aplicaciones, ni el Jdeveloper.

Lo único que se tuvo que hacer, fue crear las diferentes tablas a las que acceden las dos aplicaciones y que no estuvieran ya creadas; y crear un contendor nuevo para la aplicación Singletonserver que fue tarea del Departamento de Sistemas.

8.1 Desarrollar y probar los diferentes componentes del sistema.

Esta etapa estuvo muy marcada por la división en componentes realizada en el apartado 6.

Tanto la aplicación Singletonserver como la aplicación SAW se desarrollaron por componentes. El orden que se siguió lo explicaremos a continuación, pero la filosofía era simple.

Primero se realizaron aquellos componentes que satisfacían un mayor número de requisitos prioritarios, o bien aquellos componentes que eran necesarios para un gran número de componentes y que por tanto si no estaban hechos había varios componentes que no se podían probar.

En cuanto a las pruebas, se fueron desarrollando a medida que se iban finalizando los componentes. Algunas pruebas las realizaron los usuarios finales (En el caso del SAW una secretaria del Departamento de Admisiones entraba solicitudes como si fuera un Candidato) y en el caso del Singletonserver, yo mismo fui el encargado de comprobar que todos los módulos funcionaban correctamente.

8.1.1 Orden de desarrollo de los componentes del SAW

Componente ConstrucciónFormulario
Componente GestionNavegabilidad
Componente GestiónLogs
Componente Validación
Componente ProcesoAutentificación
Componente GeneraciónDocumentos
Componente ClientePagoVirtual

Me gustaría comentar que aunque la posibilidad de pagar un curso por TPV es un requisito de prioridad elevada, el componente que lo satisface se ha implementado al final puesto que se consideraba que era prioritario que los candidatos se pudieran matricular aunque después tuvieran que ir a pagar el curso a Secretaría Académica, frente a carecer de posibilidad de generar un PDF o entrar datos sin validar, etc.

Por otro lado, también quiero destacar que el orden de desarrollo fue designado por mi jefe de proyecto en la empresa, el Sr. Enrique Lobato, en consenso con sus clientes.

8.1.2 Orden de desarrollo de los componentes del Singletonserver

En cuanto al Singletonserver, primero se desarrolló el *subsistema Servidor* ya que sin él no había manera de probar los componentes del *subsistema Cliente*.

El orden de desarrollo fue el siguiente:

Componente que agrupa los subsistemas obtenerListaRemota/ recargarListaRemota /eliminarListaRemota
Componente Cargar Listas (forma parte de la librería Cliente)
Componente que agrupa los elementos del proceso servidorRMI
Componente que agrupa lo elementos del proceso RecargaAdministrador

Luego se realizaron los componentes de la librería cliente

Componente que agrupa los subsistemas: ObtenciónConexión/ ObtenciónListas/ RecargarListas

Y por último se realizó este componente:

Componente que forma parte de la interfaz web y que agrupa el ProcesoAutentificación

La interfaz web del Singletonserver fue el último componente añadido ya que no era necesario para realizar ninguna prueba y no ofrece ninguna funcionalidad a más que no se pueda realizar vía RMI.

8.2 Diseño de la estructura de los procedimientos de usuario.

Debido a que la aplicación del SAW está pensada para ser utilizada por usuarios sin formación ninguna, no se realizó ningún tipo de documentación ni preparación para los futuros usuarios, de manera que el correcto funcionamiento de la aplicación y la correcta utilización por parte de los usuarios dependía absolutamente de una buena elección de los prototipos de pantalla y de unos cuidadosos sistemas de validación.

En cuanto al Singletonserver, esta aplicación será utilizada simplemente por un administrador y sus funcionalidades son tan simples que no se consideró necesario documentar los procesos de usuario más de lo que ya se realizó en la fase de especificación.

9 Implantación de Sistemas (Fase 4)

El objetivo de esta fase es conseguir la aceptación total por parte de los usuarios del sistema.

Ya que las aplicaciones aquí comentadas son muy distintas los planes de implantación para cada una de ellas también lo son.

9.1 Implantación del sistema SAW

En ESADE se realizaron un conjunto de reuniones con los responsables del Departamento de Admisiones en las que se les presentó el producto definitivo o en un estado casi definitivo, y se comprobó que éste cumplía las especificaciones firmadas, además de garantizar el cumplimiento de los requisitos acordados.

En estas reuniones se propusieron una serie de modificaciones pero ninguna relevante, sobretodo temas de estilo, como el texto de los mail que se enviaban o la disposición de los botones de navegación.

En resumen, el hecho de que ESADE sea su propio cliente fomenta una proximidad y una fluidez de las comunicaciones entre departamentos que permite que estos "detallitos" se puedan hacer de mutuo acuerdo y sin traumas para ninguna de las partes.

Con esta política se facilita la integración de las aplicaciones y la satisfacción por parte de los usuarios.

Para la implantación del SAW el departamento de Desarrollo no realizó ningún tipo de estrategia, simplemente nos limitamos a entregar una versión funcional de la aplicación y fueron los propios departamentos los que se encargaron de integrar la aplicación y notificar a los usuarios la existencia de un nuevo método de matriculación.

Evidentemente surgieron algunas incidencias, pero sobretodo fueron más temas de estilos los que tuvimos que solventar en esta fase que actualmente aún dura.

En la actualidad todos los departamentos utilizan SAW para matricular a sus Candidatos excepto en algunos cursos en concreto. Por lo que podemos comentar que el sistema ha sido un éxito.

9.2 Implantación del sistema Singletonserver

El plan de implantación del Singletonserver estaba muy claro y definido desde el principio.

El objetivo era que todas las aplicaciones web que utilizasen combobox comunes a mas de una de ellas, debían obtener esos combobox del Singletonserver en vez de cargarlos en local.

Por tanto, desde el momento en que la aplicación estuvo probada y funcionando se empezó un proceso de implantación en todas las aplicaciones.

El primer paso fue modificar estas aplicaciones en el servidor de Desarrollo, que es un servidor que se utiliza para este tipo de pruebas.

Las aplicaciones que se modificaron para trabajar con la librería Cliente fueron SAW, SIA, PortalMBA y Admisiones.

Cuando en Desarrollo comprobamos que las listas se cargaban correctamente, que el tiempo de respuesta era el adecuado y que no teníamos problemas con las conexiones RMI (estuvimos aproximadamente 1 semana haciendo pruebas) entonces hicimos los mismos cambios en el servidor de producción.

En resumen, un plan de implantación breve y concisa que no resultó en absoluto traumático para la organización ya que todos los errores aparecieron en el periodo de pruebas realizado en el servidor de Desarrollo, hemos de destacar que en Producción el único problema que tuvimos fue con un Grant en una tabla.

10 Valoración Económica

A continuación intentaremos estimar el valor de los dos proyectos desarrollados.

Por un lado tenemos el número de horas dedicadas a los proyectos. Para realizar estos cálculos he tenido en cuenta la fecha de iniciación de los proyectos, la fecha de finalización y las horas al día que pasaba yo en ESADE.

El SAW fue el más largo de los dos, empezó el 15 de febrero, la fecha de finalización es el 1 de Julio y las horas al día que yo trabajaba en ESADE (5 horas)

Estimamos pues que el coste en horas es aproximadamente 500 de las cuales 320 horas han sido horas de desarrollo mías y unas 120 han sido horas dedicadas por mi jefe de proyecto, Enrique Lobato en reuniones, análisis de requisitos, pruebas del código, etc.

Por último unas 60 horas han sido dedicadas a la resolución de incidencias y retoques finales.

Todas las reuniones mantenidas con personal de los distintos departamentos no solo se puede enfocar como horas "gastadas" por Enrique Lobato, ya que los miembros de los distintos departamentos también dedicaban horas y por tanto hay que contabilizarlas.

En el caso del SAW asistieron a las reuniones responsables de 3 Departamentos y estimamos que el coste ha sido de unas 20 horas.

En cuanto al diseño de la imagen de la aplicación para que esta se integre en el portal de ESADE hay que pensar que en el caso del SAW existen unos CSS que nos fueron facilitados y que tienen un coste de desarrollo de unas 15 horas aproximadamente.

Ya por último la aplicación está funcionando en un servidor de aplicaciones y hay que configurar una serie de cosas, por un lado la gestión de los logs, aumentar los recursos del contenedor en el que va a funcionar, asignarle un datasource, etc... este conjunto de cosas realizadas por el departamento de sistemas supuso en el caso del SAW toda una mañana, unas 5 horas.

En cuanto al Singletonserver, este proyecto es más breve y su coste en horas es muy inferior al del SAW. Estimo que le habré dedicado unas 120 horas de desarrollo y unas 40 en el mantenimiento y en portar las otras aplicaciones al nuevo sistema ya que en 2 meses el Singletonserver estaba funcionando.

Mi jefe dedicó mucho menos tiempo a este proyecto, ya que no tubo que mantener entrevistas, ni que hacer pruebas y se limitó a analizar conmigo los requisitos, y verificar que los cumplía al finalizar la aplicación. El tiempo dedicado por el es de unas 20 horas.

No existen CSS para el Singletonserver, ni reuniones con otros departamentos, lo que si que existe son horas dedicadas por el departamento de sistemas, y más que para el SAW, ya que el Singletonserver es una aplicación muy crítica y que funciona en un contenedor para ella sola.

Evidentemente el coste por hora mío no es el mismo que el del Sr. Enrique Lobato.

Haciendo una estimación en función de su posición y de cómo está el mercado actualmente, creo que puedo afirmar que su precio hora es del orden de los 20 euros, mientras que el mío es de 6.6 euros

El valor de cada hora dedicada por un miembro de un departamento lo estimaremos en 15 euros, y el valor de las horas dedicadas a realizar el diseño de los CSS en un poco menos, unos 10 euros.

Resumiendo de momento tenemos lo siguiente:

	Valor Hora	Horas SAW	Horas Singletonserver
Horas Enrique Lobato	20	120	20
Horas Jorge Luis Rodríguez	6.6	380	160
Horas miembros Departamentos Implicados	15	20	0
Horas diseño de los CSS	10	15	0
Horas Departamento de Sistemas	15	5	8
Valor total		**5433 euros**	**1576 euros**

Por otro lado el desarrollo de estas aplicaciones a supuesto la utilización de equipos, software y otros elementos propios de la organización, como luz, espacio, agua, etc.

Dado que el coste de la luz, el espacio ocupado, etc.. es muy difícil de medir y su valor poco relevante no lo tendremos en cuenta, pero el valor del Hardware y del Software sí.

Asumiendo que en ESADE el hardware se amortiza cada 3 años y que las licencias de Oracle las hay que renovar cada año el coste es aproximadamente el siguiente.

Una licencia Oracle Application Server Enterprise Edition como la que tienen en ESADE, vale 6000 euros al año.

En cuanto al Hardaware, mi equivo era el mismo que el de mi jefe de proyecto Enrique Lobato, eran ordenadores Dell Pentium 4 con 1Gb de ram. El valor de estos equipos en el mercado es de 600 euros. Hemos de pensar que se amortizan en 3 años y que un año tiene aproximadamente 480 horas laborales, por tanto, estos pcs se amortizan en 1440 horas. El valor hora de este equipo es aproximadamente 0,42.

En cuanto al Hardware del Servidor de aplicaciones ESADE tiene un servidor cuyas características desconozco, pero teniendo en cuenta el volumen de trabajo que gestionan, puedo afirmar que no es malo.

Si nos basamos en el valor medio de un servidor Dell, el precio ronda los 6000 euros y por tanto el valor hora de ese servidor es: 4.2 euros.

Nos queda la siguiente tabla.

	Valor Hora	Horas SAW	Horas Singletonserver
Horas equipos personales	0.42	380	160
Horas servidor desarrollo	4.2	380	160
Horas Licencias Oracle	12.5	380	160
Valor total		**6505.6 euros**	**2739.2 euros**

En resumen y teniendo en cuenta siempre que mucha de la información es aproximada, El valor del SAW es de unos 12000 euros, mientras que el valor de Singletonserver es de 4200 euros.

11 Conclusiones

El proyecto SAW era una aplicación que había de cumplir unos objetivos muy claros de cara a la organización:

Descripción del objetivo	Estado
Construcción de un sistema que permita a los alumnos de ESADE matricularse de los cursos ofertados por la institución.	Conseguido
Permitir configurar los campos que se mostrarán en la solicitud de admisión en función del curso	Conseguido
Facilitar a los Candidatos el proceso de introducir sus datos	Conseguido
Permitir que los usuarios paguen los cursos directamente desde la aplicación	Conseguido
Garantizar la validez de los datos introducidos por los usuarios antes de gestionar los trámites para la matriculación	Conseguido
Permitir visualizar la solicitud de admisión en cualquier punto de su cumplimentación por parte del alumno	Conseguido
Notificación al usuario y al responsable del servicio de admisiones mediante un mail que la nueva solicitud de admisión se ha cumplimentado con éxito.	Conseguido
Reducir los costes asociados al proceso de admisión de candidatos	Conseguido
Reducir la burocracia asociada a una Solicitud	Conseguido
Unificar el proceso de admisión de Candidatos en todos los departamentos	Conseguido
Permitir seguir el estado de la aplicación mediante un Log	Conseguido

Tal y como se ve en el cuadro todos estos objetivos se han conseguido.

Personalmente considero que son pocos los procesos que consiguen cumplir todos los objetivos que se marcan en un principio. Pienso que en este sentido el SAW ha sido todo un éxito ya que ha logrado satisfacer las expectativas de todos los departamentos que lo han utilizado.

Considero que las claves de este éxito han sido dos.

- Un muy buen análisis de requisitos, análisis basado en el profundo conocimiento de la organización que posee Enrique Lobato y como no, en las numerosas reuniones con miembros de las distintas Unidades. Reuniones que sirvieron para perfilar los distintos objetivos.

- Una buena comunicación con las distintas unidades. Yo fui testigo y entre comillas "sufrí" los cambios de idea de algunos departamentos, que decidían a última hora añadir, modificar o incluso eliminar algún proceso del SAW. Estos cambios se realizaban o no después de negociar con todos los departamentos implicados las características del cambio. Esto que podría parecer negativo en realidad es muy positivo, ya que el Departamento se siente escuchado y ve como el programa se realiza en función de sus necesidades y peticiones (siempre y cuando estas sean coherentes) ofreciéndole una sensación de programa hecho a medida que siempre se agradece.

En cuanto al proyecto Singletonserver puedo afirmar que también ha sido un éxito.

- Actualmente todas las aplicaciones cargan más deprisa, ya que no han de cargar las listas públicas al inicio.
- Son más fáciles de mantener, ya que cuando hay que cambiar una lista no se cambia en todas las aplicaciones sino que se cambia únicamente en el Singletonserver.
- Son más fáciles de ampliar. Si una aplicación ha de mostrar una pantalla más que contiene una lista que no poseía, es muy probable que la lista ya la cargue el Singletonserver y que por tanto, no necesite generar el código de cargar la lista.
- Existe una mayor coherencia entre aplicaciones. Antes del Singletonserver algunas aplicaciones tenían las mismas listas pero con distintos valores y ahora al existir una única instancia de esas listas esto ya no pasa.
- Se ha liberado una gran cantidad de memoria del servidor de aplicaciones al no tener las listas replicadas hasta 5 veces.
- Se han reducido sensiblemente las peticiones a la base de datos, ya que había aplicaciones que solicitaban las listas siempre que las necesitaban. Esto en aplicaciones web puede suponer pedir la lista 6 o 7 veces por minuto, en función del número de usuario concurrentes.
- La creación de nuevas aplicaciones es más simple y por tanto más económica. Lo cual es siempre de agradecer.
- Existe un control total sobre las listas que están cargadas, los idiomas en que están cargadas y la fecha de la última actualización de sus valores.

En resumen ha sido un proyecto que ha aportado muchas mejoras al Departamento de Desarrollo felicitando sobretodo la labor de los programadores a la hora de desarrollar nuevas aplicaciones.

12 Valoración personal

La principal conclusión obtenida es que he aprendido mucho. Me explicaré.

Este ha sido mi primer proyecto de envergadura realizado en una empresa.

El hecho de hacerlo en una empresa, me ha obligado a cambiar mi manera de trabajar para amoldarme a los métodos de la organización, además de tener que verificar que mi código cumplía toda una serie de requisitos.

Éste, como la mayoría de los cambios, ha tenido sus cosas buenas y sus cosas malas.

Por un lado, he adquirido experiencia y un gran conocimiento técnico de Java, Struts, Hibernate, RMI, JSP, JavaScript, CSS, Taglibs, Tiles. He asentado conocimientos que ya poseía como: diseño de sistemas, aplicación de patrones, análisis de requisitos, etc.

Pero, por otro lado, también es verdad que al ser una empresa, uno no puede hacer las cosas como quiere, y a veces, se ha de presentar una solución menos "elegante" para poder cumplir los plazos.

Esto desde un punto de vista más bien técnico.

Desde un punto de vista más general, considero que el conocimiento del negocio, así como la experiencia adquirida en trato con clientes, negociación de requisitos, presentación de proyectos, etc; tienen un valor inestimable en la formación de cualquier ingeniero.

En resumen, me siento orgulloso del proyecto, tiene algunas cosas mejorables, pero otras de sus características son muy destacables y si hoy pudiera volver a escoger tema de proyecto, pienso que repetiría.

13 Bibliografía

Para la realización de este proyecto he consultado:

- Hibernate Reference 3.1
- Java JDK 1.4
- Struts reference 1.3
- PLSQL Developer guide
- Página web "Ask Tom"
- Libro de estilo para aplicaciones (manual de ESADE) v1.2
- Wikipedia
- Google
- Apuntes de EDA "Estructura de Datos y Algoritmos"
- Apuntes de ES1 "Enginyeria del Software 1"
- Apuntes de ES2 "Enginyeria del Software 2"
- Apuntes de PGPSI "Planificación y Gestión de Proyectos de Sistemas de Información"
- Apuntes de HDC "Habilidades Directivas y de Comunicación"
- Apuntes de PP "Proyecto de Programación"
- Apuntes de XC "Xarxes de Computadors"
- Apuntes de PXC "Projecte de Xarxes de Computadors"
- Apuntes de BD "Bases de Datos"
- Apuntes de DABD "Diseño y administración de Bases de Datos"
- Métrica v2.1 guía de referencia
- Métrica v2.1 guía de técnicas
- Métrica v2.1 guía de usuario