Capítulo 5 - Presentación del sistema

5.1 Introducción

El software a desarrollar, se construirá teniendo en mente que será utilizado por el equipo de QA dentro de una organización, por lo que, para comenzar daremos un breve pantallazo de las tareas que realizan los miembros del equipo.

El equipo QA generalmente esta compuesto por expertos de QA y revisores de QA (analista de QA), los expertos dan soporte a los revisores durante las revisiones. Los revisores de QA son personal experimentado, entrenado en los procesos y actividades de QA, idealmente son externos (independientes) al proyecto que relevarán.

Para cada proyecto el equipo de QA debe pasar por tres etapas. La primera de ellas, es planificar las revisiones de QA, es decir, planificar las actividades de QA, las distintas auditorías que se realizarán sobre el proyecto en los distintos estadios del mismo.
Luego, la segunda etapa, consiste en ejecutar las revisiones de QA, es decir, realizar las auditorías, por ejemplo entre las tareas de esta fase se encuentra:

1. Asegurar la disponibilidad de la documentación y que la misma esté actualizada con las últimas revisiones.
2. Verificar que la documentación generada cumpla con los estándares y planes de proyecto.
3. Documentar el resultado de las revisiones de QA.
4. Definir un plan de acción para resolver las no conformidades.
5. Dar a conocer los resultados de las revisiones de QA y el plan de acción a ejecutar.

Por último, la tercera etapa, consiste en hacer un seguimiento de las actividades de QA, para poder

1. Verificar el cumplimiento del plan de acción definido.
2. Hacer seguimiento de los planes de acción y resolución de no conformidades.
3. Detectar fortalezas y debilidades de los procesos definidos.

Para cada proyecto las responsabilidades del revisor de QA son:

- Liderar la revisión de QA.
- Revisar la documentación.
- Moderar las entrevistas
- Completar los resultados.
- Colaborar activamente con el coordinador del proyecto para crear el plan de acción.
- Seguir el plan de acción de QA asegurando que se resuelvan las no conformidades detectadas.

5.2 Objetivo del sistema

El objetivo del sistema que propone ésta tesis será ayudar a alcanzar la certificación de nivel 2 para el modelo CMMI, a través del seguimiento de las auditorías y del registro y tratamiento de las no conformidades, requeridos por el área de PPQA para este nivel dejando evidencia de la aplicación de control de los procesos definidos.

5.3 Funcionalidad

La funcionalidad mínima que deberá proveer el sistema, es aquella necesaria para cubrir los requerimientos del área de soporte PPQA, que ya fueron detallados en el capítulo 3. Recordemos que las metas de esta área de proceso son:

- Evaluar objetivamente la ejecución de los procesos, los elementos de trabajo y servicios en contraste a los procesos, estándares y procedimientos definidos.
- Identificar y documentar no conformidades. Recordemos que definimos una no conformidad como el incumplimiento de un estándar o proceso de la organización o del proyecto.
- Proporcionar información a las personas que están usando los procesos y a la gerencia del proyecto, de los resultados de las actividades del aseguramiento de la calidad.
- Asegurar de que las no conformidades son tratadas.

Como ya mencionamos en la introducción, la idea es realizar un sistema que ayude en la tarea diaria a la persona encargada de realizar QA. Su función consistirá, principalmente, en generar evidencia sobre la forma de trabajar de una organización.

Esto es, generar evidencia que muestre que la organización realmente aplica el proceso que dice aplicar y que cuando no lo hace, permita registrar tales fallas como no conformidades y poner en funcionamiento los mecanismos que sean necesarios para corregirlas. Por lo que el sistema deberá permitir, registrar la ejecución de auditorías sobre los distintos proyectos de la organización, registrar las no conformidades encontradas en las auditorías realizadas, debe permitir el seguimiento y control de las no conformidades encontradas, notificando al nivel de autoridad que corresponda. El sistema deberá también facilitar la realización de auditorías sobre los proyectos permitiendo agendar las distintas auditorías y controlando su vencimiento.

Para ver en detalle la funcionalidad que vamos a implementar a continuación definimos nuestros casos de usos. Para su mejor comprensión se recomienda leer el apéndice de esta tesis.

5.3.1 Casos de uso

El objetivo de esta sección es ayudar a comprender cual es la funcionalidad de nuestro sistema. A fin de no extendernos demasiado y con la certeza de que no se va a ver comprometido el entendimiento, para los casos de ABM solo nos limitamos a especificar las altas, que pensamos que son mas interesantes respecto de la interacción con el usuario. Creemos que las bajas y modificaciones no aportan mucho a la comprensión.

Antes de comenzar a describir los casos de uso, algunas aclaraciones para que se entiendan mejor:

- **Auditoría pendiente:** entendemos como auditoría pendiente, a aquella auditoría asociada a un proyecto que tiene tareas en estado “no iniciada” o “abierta”.
- **No conformidad**: para nosotros una no conformidad es una tarea abierta. Un tarea define un aspecto a verificar como por ejemplo “Que el documento cumpla con el template”.
- **Proyecto en gestión**: un proyecto en gestión es aquel que no se encuentra en el estado finalizado.

- **RN**: son reglas de negocio, nosotros aquí las utilizamos como notas del caso de uso para la aclaración de cuestiones que ayudan a entender el funcionamiento del caso de uso pero que exceden la interacción del usuario con el sistema.

Diagrama de los casos de uso mas importantes

Vamos a presentar tres diagramas, cada uno de ellos agrupa los casos de uso de una vista del sistema, el panel de control, la agenda y por último la configuración,

Panel de Control.

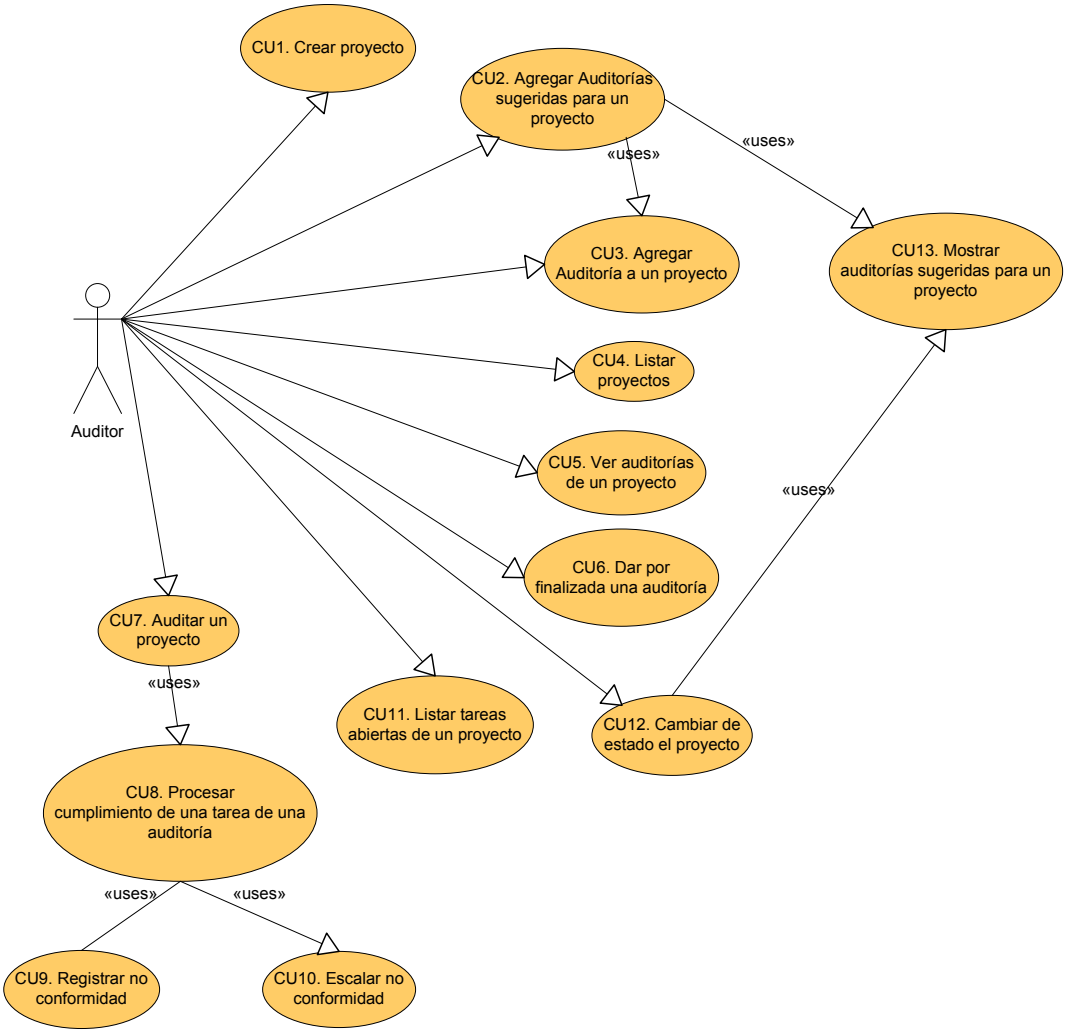

Figura 11: Diagrama de los casos de usos del panel de control.

Agenda

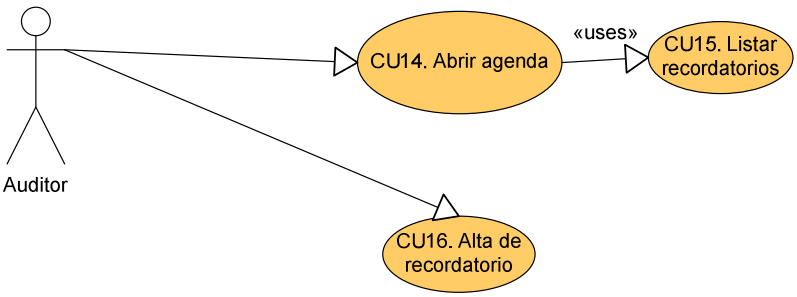

Figura 12: Diagrama de los casos de usos de la agenda.

Configuración

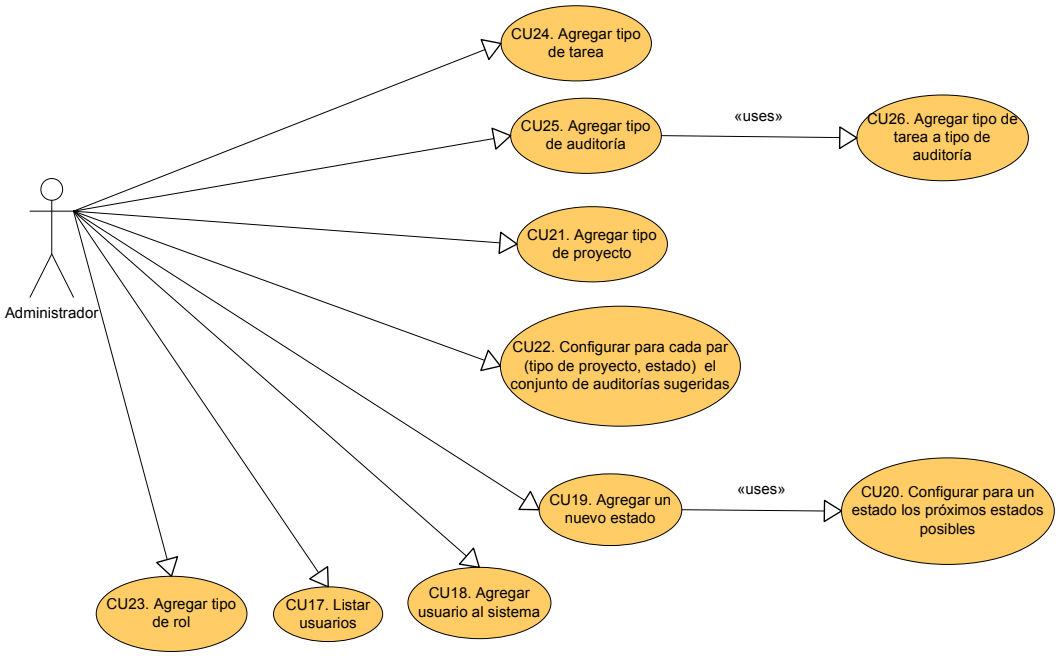

Figura 13: Diagrama de los casos de uso de la vista de configuración

Detalle de los casos de uso mas importantes

CU 1_Crear proyecto

Objetivo: Agregar un nuevo proyecto al sistema.
Contexto de uso: Cuando llega un nuevo pedido, se da de alta en el sistema el proyecto correspondiente.
Precondiciones:

Poscondiciones:

1. El proyecto se dio de alta en el sistema.
2. El proyecto esta en gestión.
3. Si eligió aceptar las auditorías sugeridas, las auditorías sugeridas están dadas de altas en estado pendientes.

Escenario principal:

00_Proyecto dado de alta

1.El usuario solicita dar de alta un nuevo proyecto.
2.El sistema solicita los datos requeridos, nombre del proyecto, tipo de proyecto y que usuarios van a cumplir los distintos roles.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4.En caso que existan auditorías para el tipo de proyecto seleccionado, el sistema muestra una lista de auditorías sugeridas.
5.El usuario puede seleccionar las auditorías sugeridas para el tipo de proyecto.
6.Invocar **2_Agregar Auditorías sugeridas para un proyecto.**
7.El sistema completa el estado del proyecto.
8.El sistema informa que el proyecto ha sido dado de alta.
9.FIN

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no esta completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos proyectos con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN.

Escenario Alternativo:

A1_El usuario no desea crear las auditorías sugeridas para el tipo de proyecto
4.a Condición: El usuario no desea crear las auditorías sugeridas para el tipo de proyecto.
4.a.1 va al paso 7 para completar el alta del proyecto.
4.a.2 FIN.

CU 2_Agregar Auditorías sugeridas para un proyecto.

Objetivo: Agregar a un proyecto las auditorías que se sugieren en función de su tipo y estado.

Contexto de uso: Cuando se da de alta un nuevo proyecto o cuando a uno existente se le modifica su estado se pueden asociarle las auditorías sugeridas por el sistema para el tipo de proyecto en el estado actual.
Precondiciones:

1. El proyecto se encuentra en estado en gestión.

Poscondiciones:

1. Las auditorías han sido agregadas al proyecto.

Escenario principal:

00_Auditorías agregadas al proyecto.

1.Invocar **13_ Mostrar auditorías sugeridas para un proyecto**
2.El usuario selecciona los tipos de auditorías a agregar.
3.El sistema configura al proyecto según RN02.
4.FIN.

RN02
El sistema crea las auditorías seleccionadas por el usuario y las agrega a la lista de auditorías pendientes del proyecto.

CU 3_Agregar Auditoría a un proyecto

Objetivo: Agregar a un proyecto una nueva auditoría.
Contexto de uso: En un momento posterior al alta del proyecto el usuario decide agregarle una nueva auditoría a realizar.
Descripción resumida:
Precondiciones:

1. El proyecto se encuentra en estado en gestión.

Poscondiciones:

1. La auditoría ha sido agregada al proyecto elegido.

Escenario principal:

00_Auditoría agregada exitosamente
1.El usuario solicita agregar una auditoría a un proyecto
2.El usuario selecciona un proyecto en estado en gestión.
3.El sistema lista los distintos tipos de auditorías.
4.El usuario selecciona el tipo de auditoría que desea agregarle al proyecto.
5.El sistema crea una nueva auditoría del tipo elegido.
6.El sistema actualiza el proyecto según RN01.
7. El sistema informa el éxito de la operación.
8.FIN

RN01
Luego de crear la nueva auditoría el sistema la agrega a la lista de auditorías pendientes del proyecto seleccionado.

CU 4_Listar proyectos

Objetivo: Ver la lista de los proyectos en gestión y finalizados.
Contexto de uso: El usuario debe tener acceso a los proyectos cargados en el sistema para poder auditarlos.
Descripción resumida: Cuando el usuario ingresa al sistema se le muestra la lista de proyectos para que pueda empezar a operar. Para administrar los proyectos se deben mostrar para cada proyecto de la lista, entre otra información, el estado en el que se encuentra.
Precondiciones:

1. Existen proyectos para mostrar.

Poscondiciones:

1. Se listan los proyectos.

Escenario principal:

00_Lista de proyecto
1.El usuario solicita ver la lista de proyectos.
2. El sistema mostrará la lista de proyectos mostrando para cada proyecto sus datos más relevantes e información de su estado.
3.FIN.

CU 5_Ver auditorías de un proyecto

Objetivo: Poder ver para un proyecto las auditorías asociadas.
Contexto de uso: una vez listados los proyectos el usuario puede ver el estado de las auditorías realizadas a un proyecto seleccionado.
Descripción resumida: El usuario podrá ver las auditorías pendientes y finalizadas. También vera las tareas abiertas de las auditorías realizadas sobre el proyecto.
Precondiciones:

1. Hay un proyecto seleccionado.

Poscondiciones:

1. Se listan las auditorías asociadas al proyecto, junto con sus tareas abiertas y cerradas.

Escenario principal:

00_Lista de auditorías
1. El usuario solicita ver las auditorías asociadas a un proyecto.
2. El sistema muestra la lista de las auditorías asociadas al proyecto.
3. El sistema muestra las tareas asociadas a las auditorías del proyecto.
4. El sistema indica el estado de las auditorías listadas, si están pendientes o finalizadas.
5. FIN.

CU 6_Dar por finalizada una auditoría

Objetivo: Dar por cumplida una auditoría
Contexto de uso: hay situaciones en las que se requiere dar por cumplida una auditoría pendiente de un proyecto.

Descripción resumida: Dar por finalizada una auditoría implica cerrarla y pasarla a la lista de auditorías finalizadas.
Precondiciones:

1. El sistema verifica que haya una auditoría seleccionada.
2. El sistema verifica que la auditoría seleccionada este en estado pendiente.

Poscondiciones:

1. La auditoría fue finalizada.

Escenario principal:
00_Auditoría finalizada.
1. El usuario solicita dar por finalizada una auditoría pendiente (con tareas no iniciadas o abiertas.)
2. El sistema pregunta al usuario si desea dar por cumplida la auditoría.
3. El usuario responde que quiere dar por cumplida la auditoría.
4. El sistema cierra la auditoría y deja las tareas en el estado que en se encuentran para que puedan ser consultadas.
5. El sistema pasa la auditoría del proyecto de pendiente a finalizada.
6. El sistema informa el éxito de la operación.
7.FIN.

Escenario alternativo:

A1_El usuario responde que no quiere cerrar la auditoría
3.1Condicion: El usuario responde que no quiere cerrar la auditoría
3.1.a FIN

CU 7_Auditar un proyecto.

Objetivo: Realizar una auditoría sobre un proyecto
Contexto de uso: Este caso se de uso se dispara cuando un usuario quiere realizar una auditoría sobre un proyecto
Descripción resumida: Permite auditar un proyecto en gestión
Precondiciones:

1. El proyecto a auditar esta en gestión.

Poscondiciones:

1. El proyecto esta auditado

Escenario Principal:

00_Audita proyecto

1.El usuario solicita realizar auditoría sobre un proyecto.
2.El sistema lista las auditorías pendientes del proyecto.
3.El usuario selecciona una auditoría.
4.Invocar **8_ Procesar cumplimiento de una tarea de una auditoría**
5.El sistema actualiza los datos de la auditoría, fecha de actual y quien la realizó.
6.El sistema informa éxito de la auditoría.
7.FIN.

Escenario alternativo

A1 La auditoría a realizar no está entre las pendientes del proyecto
3.1 Condición: La auditoría a realizar no esta entre las pendientes del proyecto
3.a.1 Invocar **3_Agregar Auditoría a un proyecto.**
3.a.2 El sistema vuelve al paso 2 para que el usuario seleccione una auditoría.
3.a.3 FIN.

CU 8_Procesar cumplimiento de una tarea de una auditoría

Objetivo: Verificar el cumplimiento de una tarea.
Contexto de uso: El realizar una auditoría sobre un proyecto implica controlar el cumplimiento de las tareas requeridas por la misma.
Descripción resumida: Permite abrir o cerrar una tarea de las requerida por la auditoría en cuestión.
Precondiciones:

1. La tarea elegida no esta en estado "Cerrada".

Poscondiciones:

1. La tarea no esta en estado "no iniciada"

Escenario Principal:

00_Tarea verificada

1. El sistema lista todas las tareas para cumplir la auditoría seleccionada.
2. El usuario selecciona la tarea a verificar.
3. El usuario marca la tarea como cumplida.
4. El sistema pasa a estado "Cerrada" a la tarea
5 El sistema procesa la auditoría según RN03
8.FIN

Escenario alternativo

A1_El usuario marca la tarea como no cumplida.
3 Condición: El usuario marca la tarea como no cumplida.
3.a.1 La tarea esta en estado abierta
3.a.2 Invocar **10_Escalar no conformidad**.
3.a.3 FIN.

Escenario alternativo

A1A1 La tarea esta en estado no iniciada
3.a.1.a Condición: La tarea esta en estado no iniciada
3.a.1.a.1 Invocar **9_Registrar no conformidad**.
3.a.1.a.2 FIN

RN03
Si todas las tareas de la auditoría están en estado "cerrada" entonces el sistema da por cumplida la auditoría. Esto implica sacar la auditoría de las lista de auditorías pendientes del proyecto y agregarla a la lista de auditorías finalizadas.

CU 9_Registrar no conformidad

Objetivo: Registrar una no conformidad.
Contexto de uso: Cuando se está verificando el cumplimiento de una tarea, si esta no se cumple se debe registrar una no conformidad.
Descripción resumida: Permite asociar no conformidades que surgen al auditar un proyecto en gestión.
Precondiciones:

1. La tarea esta en estado "no iniciada"

Poscondiciones:

1. La tarea esta en estado "abierta".

Escenario Principal:

00_Registra no conformidad

1. El sistema pone a la tarea en estado "abierta".
2. El sistema solicita una descripción.
3. El sistema completa información de estado de la tarea, auditor, fecha y nivel de escalamiento.
4.El sistema informa de la no conformidad al nivel correspondiente.
5.El sistema informa el éxito de la operación.
7.FIN

CU 10_Escalar no conformidad

Objetivo: Informar a las autoridades del proyecto que hay una no conformidad abierta.
Contexto de uso: Cuando se está auditando un proyecto y se encuentra que una tarea esta abierta luego del plazo establecido se debe escalar la no conformidad
Descripción resumida: Permite escalar o notificar a los responsables del proyecto cuando una tarea abierta esta sin solución durante el lapso que tiene de acuerdo al tipo de auditoría.
Precondiciones:

1. La tarea esta en estado "abierta"

Poscondiciones:

1. La no conformidad fue escalada.
2. La tarea tiene información actualizada.

Escenario Principal:

00_No conformidad escalada
1. El usuario solicita al sistema escalar una no conformidad.
2. El sistema informa al próximo nivel de escalamiento.
3. El sistema actualiza la información de la tarea, fecha de actual y auditor.
4. El sistema informa el éxito de la operación.
5.FIN.

CU 11_Listar tareas abiertas de un proyecto

Objetivo: Ver todas las tareas abiertas sobre un proyecto.
Contexto de uso: Cuando el usuario ingresa al sistema puede solicitar ver cuales son las no conformidades asociadas a un proyecto, es decir, todas las tareas abiertas.
Descripción resumida: Se mostrará la lista de tareas abiertas de l proyecto seleccionado, indicando a que auditoría pertenece , el motivo y la fecha en la que se reportó.
Precondiciones:

1. Hay un proyecto seleccionado.
2. Existen tareas abiertas asociadas al proyecto elegido.

Poscondiciones:

1.Se muestran las tareas abiertas del proyecto.

Escenario Principal:

00_Lista de tareas abiertas.
1.El usuario solicita ver la lista de no conformidades de un proyecto.
2.El sistema lista las tareas abiertas del proyecto elegido.
3.FIN.

CU 12_ Cambiar de estado el proyecto.

Objetivo: Poder indicar el estado de un proyecto
Contexto de uso:
Descripción resumida: Permite cambiar de estado a un proyecto.
Precondiciones:

1. Hay un proyecto seleccionado.

Poscondiciones:

1. El proyecto cambio de estado.

Escenario Principal:

00_Cambio de estado

1. El usuario solicita cambiar el estado de un proyecto
2. El sistema muestra los posibles estados destino en función del estado origen.
3. El usuario selecciona un estado destino.
4. El sistema cambia de estado al proyecto.
5. El sistema muestra la lista de auditorías sugeridas.

Invocar **13 Mostrar auditorías sugeridas para un proyecto**

6. El sistema informa el éxito de la operación.
7.FIN.

CU 13_ Mostrar auditorías sugeridas para un proyecto

Objetivo: Consultar cual es el próximo tipo auditoría sugerida en función del estado y del tipo del proyecto.

Contexto de uso: Un usuario puede pedirle al sistema que le sugiera cuales son los tipos de auditorías que se pueden realizar en el estado y tipo del proyecto sobre el que se encuentra trabajando.
Descripción resumida: Permite consultar el próximo tipo de auditoría sugerida para un proyecto. El sistema sugiere la próxima auditoría en función del tipo del proyecto seleccionado y de su estado.
Precondiciones:

1. Hay un proyecto seleccionado.

Poscondiciones:

1. Lista de auditorías sugeridas.

Escenario Principal:

00_Lista de auditorías sugeridas
1.El usuario solicita ver los próximos tipos de auditorías sugeridas.
2. El sistema lista los tipos de auditorías sugeridas.
3.FIN.

CU 14_Abrir agenda.

Objetivo: Mostrar las auditorías y tareas abiertas que hay agendadas por el usuario para los distintos proyectos.
Contexto de uso: Cuando el usuario ingresa al sistema puede consultar la agenda para agendar o listar las auditorías o tareas abiertas. Las auditorías y tareas que se agenden luego a su debido tiempo se le recordaran al usuario cuando ingrese al sistema.
Descripción resumida: El sistema muestra al usuario una bandeja de entrada donde se le muestran las auditorías o tareas abiertas que tiene agendadas.
Precondiciones:
Poscondiciones:

Escenario Principal:
00_Agenda
1.El usuario solicita abrir la agenda.
2. Invocar al **CU 15 Listar recordatorios.**
3.FIN.

CU 15_Listar recordatorios

Objetivo: Acceder a la lista de todas las tareas abiertas y auditorías pendientes agendadas por el usuario.
Contexto de uso: Cuando el usuario realiza una auditoría sobre un proyecto y encuentra no conformidades puede que éste desee agendar la no conformidad para no olvidarse de chequearla mas tarde. También existe la posibilidad de agendar la realización de la auditoría en cuestión o de una próxima.
Descripción resumida: Cuando el usuario ingresa a la aplicación se le muestran los recordatorios vencidos o próximos a vencer que tiene agendados. También puede acceder a la agenda del sistema.
Precondiciones:

1. Existen recordatorios asociados al usuario para mostrar.

Poscondiciones:

1. Se muestran todas las tareas abiertas y las auditorías agendadas por el usuario logueado.

Escenario Principal:

00_Lista de recordatorios
1.El usuario solicita ver todas las auditorías por él agendadas.
2.El sistema muestra la lista de recordatorios de auditorías.
3. FIN.

Escenario Alternativo:

A1_El usuario solicita ver todas las no conformidades por el agendadas.
1.1 Condición: El usuario quiere ver todas las tareas abiertas agendadas-
1.a.1 El sistema lista los recordatorios de no conformidades agendados.
1.a.2 FIN.

CU 16_ Alta de recordatorio

Objetivo: Agendar una auditoría a realizar sobre un proyecto o una no conformidad a verificar del mismo para que el sistema le recuerde el evento en el momento señalado.
Contexto de uso: el usuario selecciona una auditoría pendiente de un proyecto o una tarea abierta de una auditoría para agendarla.
Descripción resumida: El usuario puede indicarle al sistema que le recuerde en una determinada fecha o con cierta frecuencia que le tiene que realizar una determinada auditoría a un proyecto o que tiene que rechequear una tarea abierta. Por ejemplo agendarse la próxima auditoría a realizar sobre el proyecto.
Precondiciones:

1. La auditoría o tarea a agendar no esta cumplida.

Poscondiciones:

1. La auditoría o tarea ha sido agendada.

Escenario Principal:

00_Recordatorio creado

1.El usuario solicita agendar una auditoría.
2.El sistema lista las auditorías pendientes de los proyectos.
3 El usuario selecciona una auditoría de la lista.
4 El usuario completa una descripción e indica la fecha y/o la frecuencia para el recordatorio.
5.El sistema guarda el recordatorio
6. FIN.

Escenario Alternativo:

A1_El usuario solicita agendar una tarea abierta o no conformidad.

1.1 Condición: El usuario quiere agendar una tarea abierta.
1.a.1 El sistema lista las no conformidades de los proyectos.
1.a.2 El usuario selecciona la no conformidad.
1.a.3 El sistema vuelve al paso 4 para que el usuario completa la información del recordatorio.

CU 17_Listar usuarios

Objetivo: Consultar los usuarios del sistema
Contexto de uso: Para operar sobre los usuarios el administrador debe ver la lista de todos los usuarios.
Descripción resumida: Permite ver todos los usuarios del sistema.
Precondiciones:

1. El usuario es administrador.

Poscondiciones:

1. Se listan los usuarios del sistema.

Escenario Principal:

00_Lista de usuarios

1. El usuario solicita ver la lista de usuarios.
2. El sistema lista todos los usuarios del sistema
3.FIN.

CU 18_Agregar usuario al sistema

Objetivo: Dar de alta en el sistema a un nuevo usuario.
Contexto de uso: Durante la configuración del sistema el usuario administrador debe dar de alta todos los usuarios del mismo.
Descripción resumida: Permite agregar un nuevo usuario del sistema
Precondiciones:

1. El usuario es administrador.

Poscondiciones:

1. El nuevo usuario ha sido agregado.

Escenario Principal:

00_Alta de usuario
1. El usuario solicita dar de alta un nuevo usuario
2. El sistema solicita los datos requeridos, nombre, usuario ,clave, email, rol etc.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4. El sistema almacena el nuevo usuario.
5. FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos

3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos usuarios con el mismo nombre de usuario.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 FIN.

CU 19_Agregar un nuevo estado.

Objetivo: Agregar un estado posible para un proyecto.
Contexto de uso: Durante la configuración del sistema se deben especificar los estados por los que puede pasar un proyecto.
Descripción resumida: Permite agregar un nuevo estado.
Precondiciones:

1. El usuario es administrador

Poscondiciones:

1. El estado ha sido dado de alto.

Escenario Principal:

00_Alta de estado.

1. El usuario solicita dar de alta un nuevo estado.
2. El sistema solicita los datos requeridos, nombre.
3. Invocar al **CU 20_Configurar para un estado los próximos estados posibles.**
3. El sistema verifica que los datos requeridos estén completos y sean válidos.
4. El sistema almacena el nuevo estado
5.FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son validos.
3.b Condición: Algunos de los campos solicitados no son validos.
El sistema controla que no existan dos estados con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN.

CU 20_Configurar para un estado los próximos estados posibles

Objetivo: Configurar las transiciones de estados
Contexto de uso: Durante la configuración del sistema se deben especificar las transiciones de estados
Descripción resumida: Permite configurar las transiciones de estados.
Precondiciones:

1. Hay un estado origen seleccionado.
2. El usuario es administrador.

Poscondiciones:

1. El estado origen tiene asociados posibles estados destino.

Escenario Principal:

00_Configurar las transiciones de estados.

1. El usuario selecciona un estado para configurar las transiciones.
2. El sistema muestra la lista de estados disponibles.
3. El usuario selecciona los estados destino.
4. El sistema da de alta las transiciones.
5. FIN.

CU 21_Agregar tipo de proyecto

Objetivo: Dar de alta un nuevo tipo de proyecto.
Contexto de uso: En la etapa de configuración del sistema se deben dar de alta los diferentes tipos de proyectos.
Descripción resumida: Permite que se configuren los distintos tipos de proyectos válidos
Precondiciones:

1. El usuario es administrador

Poscondiciones:

1. El nuevo tipo de proyecto se ha agregado al sistema.

Escenario Principal:

00_Alta de tipo de proyecto

1.El usuario solicita dar de alta un nuevo tipo de proyecto.
2.El sistema solicita los datos requeridos, nombre y descripción.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
7.El sistema guarda el nuevo tipo de proyecto.
8.FIN

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.

3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de proyecto con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN

CU 22_ Configurar para cada par (tipo de proyecto, estado) el conjunto de auditorías sugeridas

Objetivo: Configurar las auditorías sugeridas para un tipo de proyecto en los diferentes estados por los que puede atravesar.
Contexto de uso: Cuando se edita un tipo de proyecto se le pueden agregar las auditorías sugeridas para el tipo de proyecto en determinados estados.
Descripción resumida: se agrega al tipo de proyecto un tipo de auditoría sugerida para el mismo.
Precondiciones:

1 Hay un tipo de proyecto seleccionado.
2. El usuario es administrador.

Poscondiciones:

1.Se agrego un tipo de auditoría al tipo de proyecto.

Escenario Principal:

00_Tipo de auditoría agregada al tipo de proyecto.

1. El usuario solicita agregar un tipo de auditoría al tipo de proyecto.
2. El usuario selecciona el estado para el que quiere sugerir un tipo de auditoría.
3. El sistema lista los tipos de auditorías existentes.
3. El usuario selecciona los tipos de auditorías a sugerir cuando los proyecto de este tipo se encuentren en el estado seleccionado.
4. El sistema asocia al tipo de proyecto y al estado con los tipos de auditorías seleccionados.
5. FIN.

CU 23_Agregar tipo de rol

Objetivo: Dar de alta un nuevo tipo de rol en sistema.
Contexto de uso: En el despliegue del sistema, el administrador deberá crear los distintos tipos de roles que tendrán los proyectos y para cada uno de ellos el nivel de escalamiento.
Descripción resumida:
Precondiciones:

1. El usuario es administrador.

Poscondiciones:

1. Se dio de alta el nuevo tipo de rol.

Escenario Principal:

00_Alta de tipo de rol
1. El usuario administrador solicita dar de alta un nuevo tipo de rol
2. El sistema solicita los datos requeridos, nombre y nivel de escalamiento.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4. El sistema almacena el nuevo usuario.
5. FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de rol con el mismo nombre y que no existan dos tipos de rol con el mismo nivel de escalamiento.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 FIN.

CU 24_Agregar tipo de tarea

Objetivo: Configurar los tipos de tareas que determinan los aspectos a controlar que permiten detectar una no conformidad.
Contexto de uso: Cuando se instala el sistema inicialmente se deben indicar todos los tipos de tareas . El administrador dará de alta los distintos tipo de tareas que sirven para encontrar no conformidades. También se podría agregar un tipo de tarea durante la operatoria del sistema.
Descripción resumida: permite crear los tipos de tareas que indican los aspectos para describir las no conformidades que pueden tener los proyectos.
Precondiciones:
1. El usuario es administrador.

Poscondiciones:
1. El nuevo tipo de tarea se ha agregado al sistema.

Escenario Principal:
00_Alta de tipo de tarea.

1.El usuario solicita dar de alta un nuevo tipo de tarea.
2.El sistema solicita los datos requeridos, nombre y descripción.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4.El sistema guarda el nuevo tipo de auditoría.
7.FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de tareas con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN.

CU 25_Agregar tipo de auditoría

Objetivo: Configurar las diferentes auditorías que se pueden realizar sobre los proyectos..
Contexto de uso: cuando se instala el sistema inicialmente se deben indicar los distintos tipos de auditorías que se podrán realizar sobre los proyectos. Solo los administradores podrán realizar esta operación. También se podría agregar un tipo de auditoría durante la operatoria del sistema.

Descripción resumida: permite crear los tipos de auditorías que sirven para auditar los proyectos.
Precondiciones:
1 El usuario que realiza la operación es administrador.
Poscondiciones:
1. El nuevo tipo de auditoría se ha agregado al sistema.

Escenario Principal:

00_Alta de tipo de auditoría

1.El usuario solicita dar de alta un nuevo tipo de auditoría.
2.El sistema solicita los datos requeridos, nombre y la lista de tareas que conforman la auditoría.
3.El usuario invoca **26 Agregar tipo de tarea a tipo de auditoría**.
4.El sistema guarda el nuevo tipo de auditoría.
5.FIN

Escenario de Excepción:

E1_Los datos requeridos no están completos
4.a Condición: Los campos obligatorios no esta completos.
4.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
4.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.

4.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
4.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de auditoría con el mismo nombre.
4.b.1 El sistema informa al usuario cuales campos son inválidos.
4.b.2 FIN

CU 26_Agregar tipo de tarea a tipo de auditoría

Objetivo: Configurar un tipo de auditoría
Contexto de uso: Cuando se crea un nuevo tipo de auditoría se le deben agregar las tareas a realizar para cumplir con la auditoría. Este caso también se puede ejecutar al momento de modificar un tipo de auditoría existente.
Descripción resumida: se agrega al tipo de auditoría un nuevo tipo de tarea a realizar para cumplir con una auditoría de este tipo.
Precondiciones:

1. Hay un tipo de auditoría seleccionado.
2. El usuario es administrador.

Poscondiciones:

1.Se agrego un tipo de tarea al tipo de auditoría.

Escenario Principal:

00_Tipo de tarea agregado a la auditoría.

1. El usuario solicita agregar un tipo de tarea al tipo de auditoría.
2. El sistema lista los tipos de tareas existentes que el tipo de auditoría aún no tiene.
3. El usuario selecciona los tipos de tarea a agregar.
4. El sistema agrega al tipo de auditoría los tipos de tareas seleccionados.

5.4 Arquitectura y diseño de alto nivel.

5.4.1 Arquitectura.

La arquitectura de la aplicación está basada en capas. Cuando pensamos un sistema en términos de capas imaginamos un subsistema principal con forma de torta de varias capas, donde cada capa se apoya en la capa inferior. En este esquema las capas superiores hacen uso de servicios definidos en las capas inferiores, pero las capas inferiores no tienen conocimiento de las capas superiores. Cada capa esconde a las superiores las capas inferiores.

Cuando se habla de capas de arquitecturas, a menudo hay una confusión entre los términos capas(layer) y niveles o pisos(tier). A veces son usados como sinónimos, pero para la mayoría de la gente los niveles(tier) implican una separación física. Los sistemas cliente/servidor en general son descritos como sistemas de dos niveles y la separación es física. Usamos el termino capa para enfatizar que no necesariamente tenemos que correr las capas en diferentes máquinas.

Por su naturaleza, ésta es una aplicación cliente/servidor y usamos una arquitectura de cuatro capas primarias , la capa de presentación, la capa de servicios, la capa de dominio y la capa de fuente de datos. La capa de presentación provee la comunicación con el usuario permitiéndole invocar los servicios y mostrándole la información del resultado. La capa de servicios define la forma y el medio de comunicación del cliente con la lógica de dominio. La capa de dominio se encargará de la lógica del sistema y la capa de fuente de datos, se encarga de la comunicación con la base de datos.

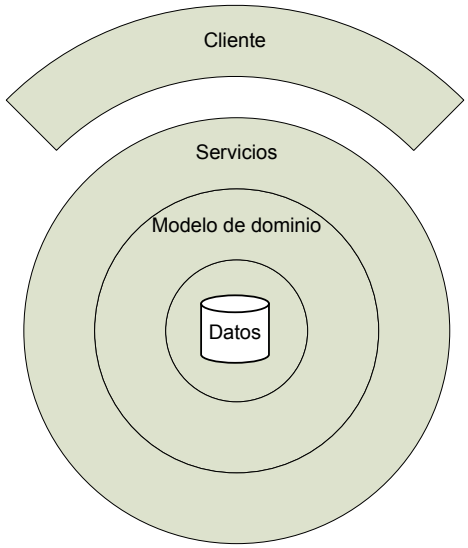

Figura 4: Modelo de las capas de arquitectura

Como ya mencionamos vamos a tener una arquitectura orientada a servicios (SOA), definiendo una fachada del servidor.

5.4.1.1 Tecnologías seleccionadas para implementar la arquitectura

Para dar soporte a la arquitectura diseñada, utilizamos en las diferentes capas una serie de tecnologías que a continuación detallamos.

Introducción

A la hora de definir la arquitectura del sistema decidimos hacer una aplicación WEB en FLEX con java y motor de persistencia hibernate. Tenemos un modelo de objetos java, una fachada del servidor y comunicación java- flex / flex-java con objetos DTO, para dar soporte a esta comunicación utilizamos BlazeDS. Para la fachada de servicios utilizamos como middleware Spring. La base de datos relacional utilizada es MySQL 5 y como servidor elegimos usar Tomcat 6.
Lo novedoso de las tecnologías usadas es uno de los puntos fuertes de este sistema. Todas ellas son tecnologías *open source*.

Adobe Flex 3

Para la capa de presentación decidimos realizar el cliente de esta aplicación con la herramienta Flex 3.
La plataforma incluye:

- Un lenguaje basado en XML llamado MXML que soporta la programación declarativa de componentes GUI para captar a los diseñadores.
- El lenguaje orientado a objetos standard, ActionScript 3.0, basado en la última especificación ECMAScript.
- Integración con el lado del servidor vía Flex Data Services (FDS) dando a las aplicaciones cliente acceso transparente al mundo J2EE.
- Componentes de charting, acceso a controles multimedia, etc.
- Un IDE basado en Eclipse con despliegue automatizado, debugging, y facilidades de traza.

La plataforma Flex 3 es fácilmente extensible y se integra bien con la parte del servidor, ya sea Java, ColdFusion, PHP, Ruby, ASP, o parecidos.
Pueden hacerse aplicaciones Flex más avanzadas incluyendo interacción con sistemas de la parte del servidor a través de Flex Data Services, que proveen acceso remoto a los objetos Java y componentes Java EE del lado del servidor, soporte de mensajería

extenso (incluyendo integración JMS), sincronización con información persistida, la integración con otras tecnologías persistentes.

Java

Optada ya la tecnología para la capa de presentación, y en base a la misma, parece lógico y natural inclinarse por tecnologías Java para la capa de Modelo. Es bien conocida la buena sintonía entre Flex y Java y su uso conjunto está cada vez más extendido en el ámbito empresarial.
Además de ser el lenguaje de programación mas popular hoy en día, java es libre y es el que mas dominamos por estar trabajando diariamente con él. También posee una integración optima, a parte de la que tiene con Flex , con las demás tecnologías elegidas para el desarrollo del producto.

BlazeDS

La capa de servicios la conectamos a Flex usando los llamados *Data Services*. Estos servicios son un componente del lado del servidor que traduce las peticiones Flex a una tecnología servidor – en nuestro caso a Java – y las respuestas de estos servicios los traduce de vuelta a un formato binario de Adobe (AMF) de esta forma para Flex es transparente lo que se esté usando en el servidor.

Adobe comercializa una versión de los Data Services llamada Adobe LiveCycle(LCDS), además existen alternativas opensource como GraniteDS. Sin embargo, junto con Flex 3.0 se liberó BlazeDS: la versión opensource y gratuita de LiveCycle y la que usaremos en nuestra tesis. Al ser gratuita no tiene todas las características de la versión que se comercializa (LCDS), pero creemos que para el alcance de esta tesis cubre los requerimientos. BlazeDS no cuenta con el módulo de Data Management que provee sincronización de datos en tiempo real y carga lazy o paginado.

BlazeDS se despliega como un conjunto de archivos JAR como parte de la aplicación Web ejecutándose en un servidor de aplicaciones Java EE o en un contenedor de servlets.

BlazeDS consiste en tres servicios clave:

- El Servicio Remoto permite a las aplicaciones Flex invocar directamente métodos de objetos Java desplegados en el servidor de aplicación. Esto es lo que utilizamos en nuestra aplicación.
- El Servicio de Mensajes provee una infraestructura publicar/suscribir que permite a las aplicaciones Flex publicar mensajes y suscribirse a un destino de mensajes, permitiendo el desarrollo de data push en tiempo real así como de aplicaciones colaborativas.
- El Servicio Proxy permite a las aplicaciones Flex hacer peticiones de servicio cross-domain de forma segura y controlada. En otras palabras, permite a las aplicaciones Flex acceder a un servicio disponible en un dominio diferente al dominio de donde la aplicación fue descargada (sin tener que desplegar un archivo con política cross-domain.xml sobre el dominio en cuestión).

BlazeDS es la capa intermedia que actúa a modo de *pegamento* entre los clientes Flex y los sistemas de backend.

Spring

Para el backend del sistema utilizamos el Framework Spring. Básicamente es un contenedor liviano de objetos que soporta inversión de control y aspectos (Spring in action [26]), sirve para múltiples propósitos y para esta aplicación aprovechamos las siguientes característica:

- Su potente gestión de configuración basada en JavaBeans, aplicando los principios de Inversión de Control (*IoC*). Esto hace que la configuración de aplicaciones sea rápida y sencilla. Ya no es necesario tener *singletons* ni archivos de configuración, una aproximación consistente y elegante. Esta factoría de *beans* puede ser usada en cualquier entorno, desde *applets* hasta contenedores J2EE. Estas definiciones de *beans* se realizan en lo que se llama el contexto de aplicación.

- Una capa genérica de abstracción para la gestión de transacciones, permitiendo gestores de transacción enchufables (*pluggables*), y haciendo sencilla la demarcación de transacciones sin tratarlas a bajo nivel. Se incluyen estrategias genéricas para JTA y un único JDBC DataSource. En contraste con el JTA simple o EJB CMT, el soporte de transacciones de Spring no está atado a entornos J2EE.
- Integración con Hibernate, JDO e iBatis SQL Maps en términos de soporte a implementaciones DAO y estrategias con transacciones. Especial soporte a Hibernate añadiendo convenientes características de *IoC*, y solucionando muchos de los comunes problemas de integración de Hibernate. Todo ello cumpliendo con las transacciones genéricas de Spring y la jerarquía de excepciones DAO.

Acegi

Para la autenticación y autorización del sistema, utilizamos el Framework Acegi (actualmente llamado Spring Security) y sus principales características son:

- Facilidad de uso.
- Integración completa en Spring, utiliza los mecanismos de configuración de Spring. Configuración mediante el contexto de aplicación de Spring o basada en atributos.
- Seguridad a nivel de instancia de objetos del dominio.
- En muchas aplicaciones es deseable definir listas de control de acceso (Access Control Lists o ACLs) para instancias de objetos del dominio individuales. Proporciona un completo paquete ACL con características que incluyen máscaras de bits, herencia de permisos, un repositorio utilizando JDBC, caché y un diseño pluggable utilizando interfaces.
- Configuración no intrusiva, mantiene los objetos libres de código de seguridad.
- La totalidad del sistema de seguridad puede funcionar en una aplicación web utilizando los filtros que proporciona. No hay necesidad de hacer cambios especiales o añadir librerías al contenedor de Servlets o EJB.
- Integración opcional con los contenedores

- Las características de autenticación y autorización que proporcionan los contenedores Servlet o EJB pueden ser usadas utilizando los adaptadores que se incluyen. Actualmente existe soporte para los principales contenedores: Catalina (Tomcat), Jetty, JBoss y Resin
- Muchas aplicaciones necesitan proteger datos a nivel de objeto basándose en cualquier combinación de parámetros (usuario, hora del día, autoridad del usuario, método que es llamado, parámetros del método invocado etc.). Acegi da esta flexibilidad sin necesidad de añadir código a los objetos de negocio.
- Protección de peticiones HTTP. Además de proteger los objetos, el proyecto también permite proteger las peticiones HTTP. Ya no es necesario depender de restricciones de seguridad definidas en el archivo web.xml. Lo mejor de todo es que las peticiones HTTP pueden ser protegidas por una serie de expresiones regulares o expresiones de paths como las utilizadas por Ant, así como autenticación, autorización y gestores de reemplazo de credenciales para la ejecución como otro usuario, todo ello totalmente pluggable.
- El sistema de seguridad puede redirigir automáticamente las peticiones a un canal de transmisión adecuado. Comúnmente esto se aplica para asegurar que las páginas seguras estarán sólo disponibles sobre HTTPS, y las páginas públicas sobre HTTP, aunque es suficientemente flexible para soportar cualquier tipo de requisitos de "canal". También soporta combinaciones de puertos no usuales y gestores de decisión de transporte pluggables.
- Esta autenticación es adecuada para aquellas aplicaciones que prefieren una simple ventana de login del navegador en lugar de un formulario de login. Acegi Security puede procesar directamente peticiones de autenticación HTTP BASIC siguiendo el RFC 1945.
- Librería de etiquetas. Proporciona una librería de etiquetas que puede ser utilizada en JSPs para garantizar que contenido protegido como enlaces y mensajes son únicamente mostrados a usuarios que poseen los permisos adecuados.
- Permite seleccionar el método utilizado para la configuración de seguridad del entorno, tanto a través del contexto de aplicación de Spring o utilizando atributos en el código fuente utilizando Jakarta Commons Attributes.
- Distintos métodos de almacenamiento de la información de autenticación

- Acegi incluye la posibilidad de obtener los usuarios y permisos utilizando archivos XML, fuentes de datos JDBC o implementando un interfaz DAO para obtener la información de cualquier otro lugar.
- Soporte para eventos. Utilizando los servicios para eventos que ofrece Spring, se pueden configurar receptores propios para eventos como login, contraseña incorrecta y cuenta deshabilitada. Esto permite la implementación de sistemas de auditoría y bloqueo de cuentas, totalmente desacoplados del código de Acegi Security.
- Utilizando EHCache u otra implementación propia se puede hacer caché de la información de autenticación, evitando que la base de datos o cualquier otro tipo de fuente de información no sea consultado repetidamente.

Hibernate

Como mapeador objeto/relacional usamos el Framework Hibernate, el cual es el ORM(Object relacional mapping) más popular en el mundo Java.

Estas herramientas hacen la programación más fácil y permiten una adhesión más cercana a los ideales de la programación orientada a objetos y a un desarrollo arquitectónico multi-capa.

Las principales características de Hibernate son:

- Persistencia transparente.
- Hibernate puede operar proporcionando persistencia de una manera transparente para el desarrollador.
- Modelo de programación natural.
- Hibernate soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.
- Soporte para modelos de objetos con una granularidad muy fina.
- Permite una gran variedad de mapeos para colecciones y objetos dependientes.
- Sin necesidad de mejorar el código compilado (bytecode).
- No es necesaria la generación de código ni el procesamiento del bytecode en el proceso de compilación.
- Escalabilidad extrema.

- Hibernate posee un alto rendimiento, tiene una caché de dos niveles y puede ser usado en un cluster. Permite inicialización perezosa (lazy) de objetos y colecciones.
- Lenguaje de consultas HQL
- Este lenguaje proporciona una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.
- Soporte para transacciones de aplicación
- Hibernate soporta transacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución) y gestiona la política optimistic locking automáticamente.
- Generación automática de claves primarias.
- Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos (secuencias, columnas auto incrementales) así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.

Base de datos

En este punto se optó directamente por una base de datos libre tan consolidada como MySQL por su rapidez, interoperabilidad y estabilidad. Para el desarrollo utilizamos la versión 5. MySQL funciona sobre múltiples plataformas.

Servidor

Para soportar esta arquitectura puede usarse en cualquier servidor J2EE, pero para la funcionalidad de la herramienta producto de esta tesis nos alcanza con un contenedor de servlets, por lo que utilizaremos Tomcat 6.
Tomcat 6 es un servidor Java EE open source ampliamente utilizado.

Resumiendo

Fue un factor importante a la hora de elegir las tecnologías para el desarrollo el hecho de que muchas de ellas son libres y son las que utilizamos a diario en nuestros trabajos a excepción de Flex 3. Gráficamente la arquitectura de nuestro sistema cliente/servidor es la siguiente:

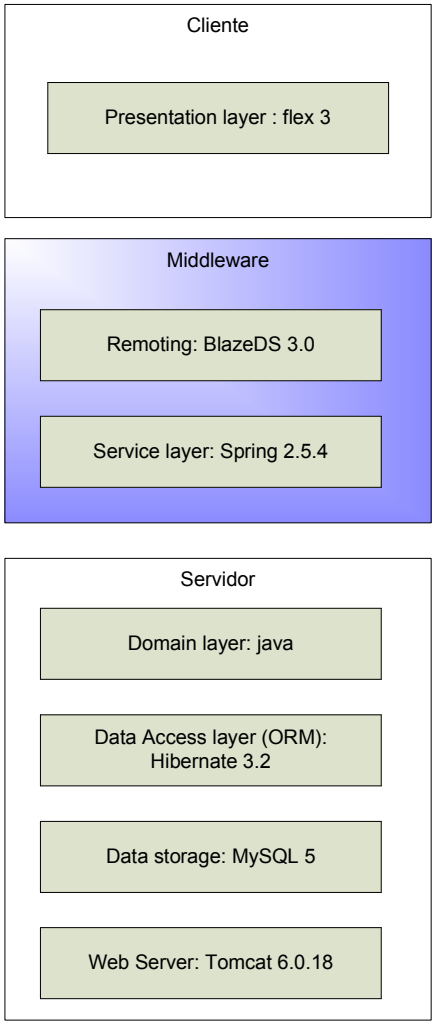

Figura 5:*Arquitectura del sistema.*

5.4.2 Diseño de alto nivel

Una vez establecida la arquitectura, hablaremos del diseño de alto nivel mostrando los distintos módulos o subsistemas de la aplicación y la interacción entre ellos.

Podemos tener varias vistas de la aplicación, una ella es la vista de módulos. Un módulo lo podemos ver como un agrupamiento lógico no necesariamente físico, de cierta funcionalidad que le da significado al mismo. Entre los módulos de este sistema se encuentran: el módulo de Seguridad, el módulo de manejo de la agenda, el módulo de gestión de auditorías, el módulo de gestión de los proyectos, el módulo de reportes,

módulo de configuración, el módulo AssemblerDTO, el módulo de Persistencia. Gráficamente:

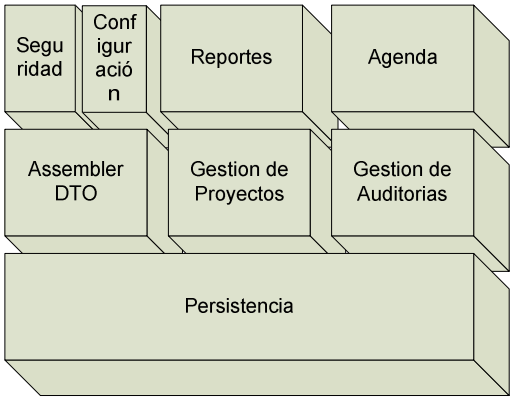

Figura 6: Vista de los módulos

- **El módulo de Persistencia:** este módulo se encarga de encapsular la interacción con hibernate , el Framework de persistencia que estamos usando. Su función es hacer transparente para la aplicación el ORM(Object Relational Mapping) e independizarnos de la base de datos.
- **El módulo AssemblerDTO:** este módulo se encarga de realizar la transformación de los objetos de dominios a DTO´s(Data transfer object) y viceversa, los cuales son utilizados para que puedan viajar al cliente, y el cliente enviarlos al servidor. En la próxima sección se explicarán los DTO con más detalle.
- **El módulo de Gestión de Proyectos:** este módulo se encarga del manejo y seguimiento de los proyectos, por ejemplo este módulo en colaboración con el de gestión de auditorías se encargaran de sugerir las próximas auditorías para un determinado proyecto. También se encarga del ABM de proyectos.
- **El módulo de Gestión de Auditorías:** se encarga del manejo de las auditorías realizadas sobre los proyectos. Se encarga también de controlar el estado de las tareas de las auditorías, del seguimiento y escala de no conformidades.
- **El módulo Agenda:** este módulo se encarga del manejo de la agenda que permite agendar las auditorías en el tiempo, controlar el vencimiento de las mismas y de las no conformidades.
- **El módulo de Reportes:** este módulo permitirá imprimir reportes sobre las auditorías pendientes de los proyectos y exportarlos a diferentes formatos.

También se van a poder sacar informes de las no conformidades registradas para un proyecto o de un grupo de proyectos.

- **El módulo de Seguridad:** se encargará de la autenticación y autorización de los usuarios. También manejará los ABM y demás operaciones sobre los usuarios.
- **El módulo de Configuración:** permitirá al administrador del sistema instanciar o desplegar la aplicación antes de que se pueda operar con ella.

5.5 Detalles de implementación

Introducción

En esta sección hablaremos de los detalles de implementación que consideramos más relevantes. También presentaremos el diagrama de clases surgido del diseño realizado para soportar los requerimientos, junto a una breve explicación del modelo.

Servicios.

Al tener una arquitectura SOA, la comunicación del cliente con el servidor es a través de servicios. En los servicios se han tratado de incluir las funciones de negocio.
Los hemos implementados como beans de Spring para que puedan ser invocados remotamente. Nuestros servicios no tienen estado, son stateless. Sus métodos son transaccionales.
Tratamos de agrupar los servicios en función de la entidad sobre la que actúan. Nosotros llamamos servicio a un método de un clase. La clase implementará una serie de métodos que operan sobre una misma entidad. Por ejemplo la clase ProyectoServiceImpl.java en la que se definen los métodos para realizar el ABM de proyectos.
Hemos implementado dos capas de servicios una remota (RemoteFacade, Fowler[17]) y otra local, la local trabaja sobre objetos de dominio y la remota sobre DTO´s(Data transfer object) de los mismos.
El tener un capa remota, nos permite ubicar en ella comportamiento adicional, por ejemplo , la creación de DTO´s(Data transfer object).

DTO

Flex al utilizar su propio algoritmo de serialización/deserialización (AMF), por problemas compatibilidad con hibernate, nos imposibilitaba poder utilizar objetos detachados para trabajar con ellos en el cliente. Como solución utilizamos DTO´s (Data transfer object) de los objetos de dominio, Fowler[17].

Al utilizar DTO en la comunicación con el servicio, tuvimos que implementar una clase que se encargue de la transformación de objetos de dominio a DTO y viceversa, para esto implementamos la clase AssemblerDTO.java. Por ejemplo para la clase proyecto, cuando cliente solicita listar los proyectos la secuencia es la siguiente:

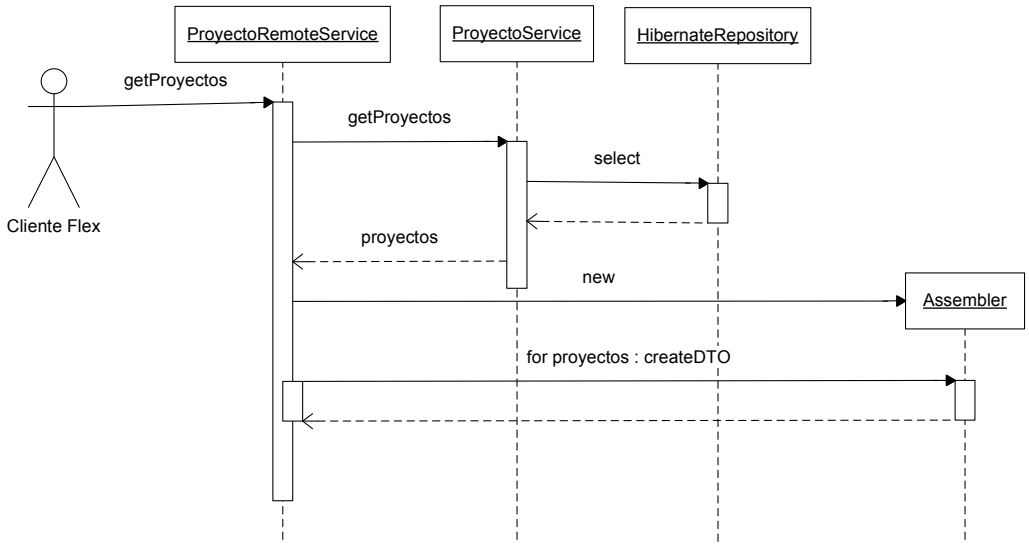

Figura 7: Diagrama de secuencia de la obtención de los proyectos.

De esta manera el cliente puede trabajar con los DTO´s. Cuando el cliente modifica el DTO, este se envía al servicio para que se pueda recuperar el objeto de dominio correspondiente y realizarle las modificaciones pertinentes.

Capa de persistencia.

Al tener un modelo orientado a objetos y una base de datos relacional necesitamos unir estos dos mundos con un ORM (Object Relational Mapping), por lo cual elegimos hibernate por su facilidad de uso y madurez.
Para encapsular la interacción con hibernate y que la lógica para la persistencia de datos no quede mezclada con la lógica del negocio implementamos una capa de persistencia representada a través de la clase HibernateRepository, de esta manera cuando la capa de lógica de negocio necesita interactuar con la base de datos, ésta utilizará los métodos ofrecidos por esta clase. Generalmente la clase de operaciones que ofrece la capa de datos se le conoce como CRUD (Create, Read, Update y Delete).

Hasta aquí los detalles de implementación mas importantes. Ahora veremos el modelo de objetos que soportan el sistema.

Diagrama de clase

A continuación exponemos el modelo de clases persistentes o de dominio que utiliza la aplicación que vamos a desarrollar.

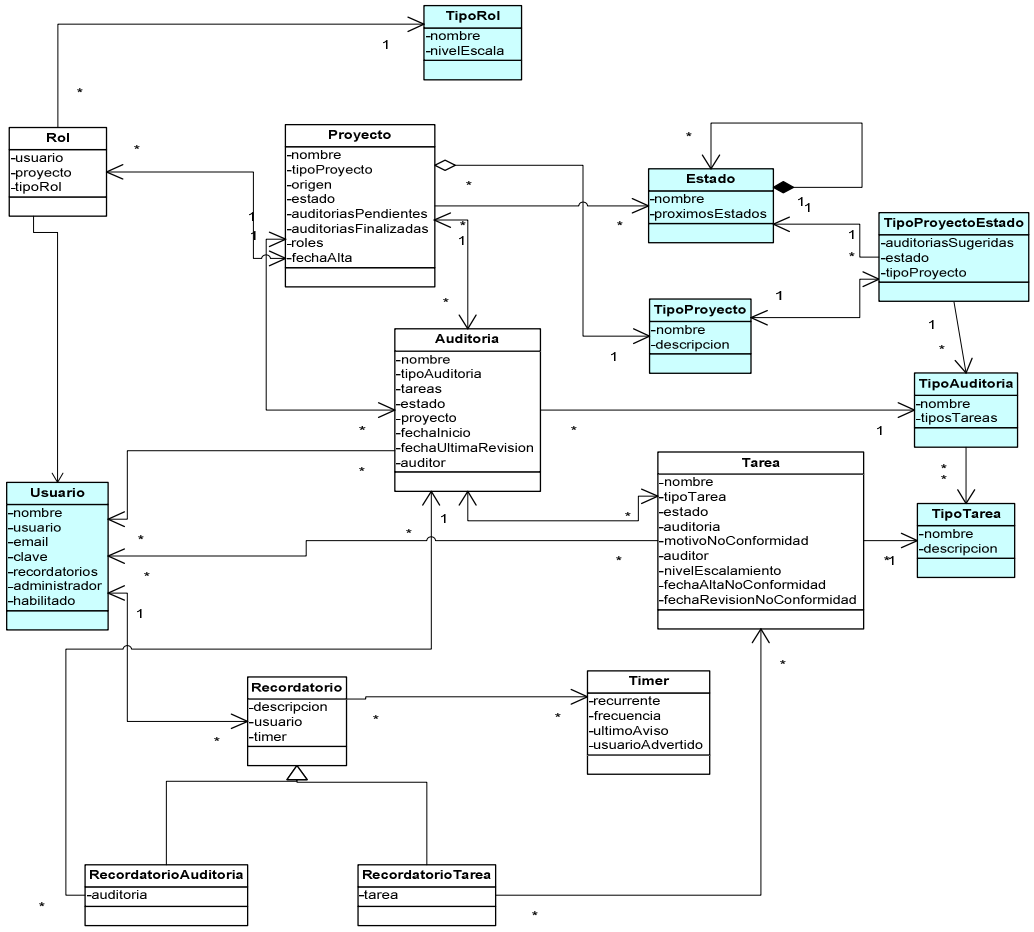

Figura 8: Diagrama de clases del sistema

Un proyecto es la unidad de trabajo del sistema, un proyecto además de los datos básicos que lo identifican, a lo largo de su ciclo de vida puede pasar por diferentes estados. Naturalmente hay proyectos de diversos tipos, el tipo de un proyecto junto con el estado en el que se encuentra en un momento dado permiten sugerir posibles auditorías a realizar. A medida que el proyecto comienza a transcurrir se le van asociando auditorías para realizarle, estas auditorías las llamamos pendientes. Cuando se realiza una auditoría sobre un proyecto y ésta es exitosa, se pasa a la lista de auditorías finalizadas el proyecto.

Los diferentes tipos de auditorías que se pueden realizar sobre un proyecto están conformadas por distintos tipos de tareas que se deben cumplir para poder dar por finalizadas las auditorías.

Inicialmente, cuando se asocia una auditoría a un proyecto, la auditoría y todas las tareas que la conforman se encuentran en estado "No iniciada". Al momento que se esta auditando un proyecto y nos encontramos con que cierta tarea no se cumple ésta pasa al estado "Abierta", si la tarea se da por cumplida pasa al estado "Cerrada". Si la auditoría tiene alguna de sus tareas en estado "Abierta" entonces su estado es "Abierta". La auditoría pasa a estado "Cerrada" cuando todas sus tareas están cerradas, en este momento se da por finalizada la auditoría.

La agenda que provee el sistema permite a los usuarios agendarse auditorías o no conformidades a realizar o revisar sobre los distintos proyectos de manera que a su debido tiempo se les recuerde. Los recordatorios pueden ser sobre auditorías o no conformidades y se puede configurar la frecuencia del aviso, es decir, si la alarma es recurrente, cada cierta cantidad de días o si solo se quiere que se le recuerde por única vez en una determinada fecha.

En un proyecto participan entre otras, personas de la organización con roles distinguidos dentro de cada proyecto como por ejemplo el líder del proyecto, el gerente etc.

En la configuración del sistema el usuario administrador deberá instanciar un conjunto de clases para que la aplicación quede operativa y pueda comenzar a funcionar. En el diagrama están en color celeste. Estas clases son de configuración, por ejemplo el administrador deberá cargar los usuarios del sistema, configurar los diferentes tipos de auditorías con su conjunto de tareas, además tendrá que definir cuales son los roles que tendrán los proyectos, los diferentes tipos de proyectos y los estados por los que podrán pasar.
Además si desea podrá indicar para cada par tipo de proyecto – estado los tipos de auditorías sugeridas.

5.6 Herramientas de desarrollo

Para el desarrollo de la aplicación utilizamos las siguientes herramientas

Eclipse

Como entorno de desarrollo utilizamos Eclipse, una plataforma de herramientas universal, un entorno integrado de desarrollo (IDE) abierto y extensible, para cualquier utilidad y no utilizamos ninguna extensión en particular.

Pese a que Eclipse esté escrito en su mayor parte en java(salvo el núcleo), se ejecute sobre una JVM (maquina virtual java) y su uso más popular sea como IDE para java, es una herramienta neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, XML, etc.

La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plugins que van conformando la funcionalidad final, así, los nuevos aportes se integran sin dificultad ni conflictos.

Flex builder 3

Para el desarrollo de las interfaces graficas del usuario en Flex, utilizamos el plugin para eclipse Flex builder, esta herramienta nos permite crear con rapidez y sencillez complejas pantallas, simplemente arrastrando y soltando sobre el área de trabajo los diferentes controles.

Maven 2

Para realizar la integración, hacer el deploy en el servidor, mantener un repositorio de librerías utilizamos una herramienta de gestión de información de proyectos llamada Maven(versión 2).

Maven tiene muchos objetivos, pero resumiendo Maven intenta hacer la vida del desarrollador sencilla proporcionando una estructura de proyecto bien definida, unos procesos de desarrollo bien definidos a seguir, y una documentación coherente que mantiene a los desarrolladores y clientes informados sobre lo que ocurre en el proyecto. Maven aligera en gran cantidad lo que la mayoría de desarrolladores consideran trabajo pesado y aburrido y les permite proseguir con las tareas más atractivas (típicamente codificación y modelado). Esto es esencial en proyectos open source donde no hay

mucha gente dedicada a la tarea de documentar y propagar la información crítica sobre el proyecto que es necesaria para atraer potenciales nuevos desarrolladores y clientes.
La ambición de Maven es hacer que el desarrollo interno del proyecto sea altamente manejable con la esperanza de proporcionar más tiempo para el desarrollo entre proyectos. Se puede llamar polinización entre proyectos o compartir el conocimiento sobre el desarrollo del proyecto.

Las principales características de Maven son:

- El modelo de objetos del proyecto POM es la base de cómo Maven trabaja. El desarrollo y gestión del modelo está controlado desde el modelo del proyecto.
- Un único conjunto de métodos son utilizados para todos los proyectos que se gestionan. Ya no hay necesidad de estar al tanto de innumerables sistemas de compilación. Cuando las mejoras se hacen en Maven todos los usuarios se benefician.
- Publicación del sitio web basado en el POM. Una vez que el POM es exacto los desarrolladores pueden publicar fácilmente el contenido del proyecto, incluyendo la documentación personalizada más el amplio conjunto de documentación generada por Maven a partir del código fuente.
- Publicación de distribuciones basada en el POM.
- Maven favorece el uso de un repositorio central de librerías, utilizando un mecanismo que permite descargar automáticamente aquellas necesarias en el proyecto, lo que permite a los usuarios de Maven reutilizar librerías entre proyectos y facilita la comunicación entre los mismos, para asegurar que la compatibilidad entre distintas versiones es correctamente tratada.
- Guías para la correcta disposición de los directorios. Maven contiene documentación sobre como disponer los directorios de forma que una vez que es aprendida se puede ver fácilmente cualquier otro proyecto que use Maven.

MySQL Query Browser , MySQL Administrator

Para crear la base de datos y posteriormente trabajar con ella utilizamos como front end las herramientas MySQL Query Browser y MySQL Administrator que fueron bajadas

gratuitamente del sitio oficial de MySQL.(url [25]) Con estas herramientas nos alcanzo de sobremanera para realizar éste trabajo.

SVN – Tortoise SVN

Actualmente existen dos grandes corrientes en cuanto al control de versiones se refiere: CVS y Subversion.

Por un lado, CVS es el sistema de control de versiones concurrentes más utilizado en el mundo del software libre, sin embargo es bastante arcaico (nació en 1985) y tiene bastantes carencias, como son la falta de versionado de directorios, de renombrado o de copias; su uso poco eficiente de la red o el que no almacene archivos binarios completos, solo los cambios entre versiones.

Subversion es un CVS de código abierto, bajo una licencia del tipo Apache/BSD, que nació como alternativa a CVS y que intenta solucionar estos problemas y carencias.

Subversion es la alternativa de este proyecto para el control de versiones, la herramienta concreta a utilizar será Tortoise SVN. Tortoise SVN es un cliente Subversion, implementado como una extensión al shell de Windows. Es software libre, liberado bajo la licencia GNU GPL.