PFC- Aplicaciones Web para trabajo colaborativo:

Aplicación para Control de una Integración de S.I.

2º Ciclo Ingeniería Informática Curso 2011-2012

Memoria

Consultor : Fatos Xhafa
Autor : Miguel Angel Pineda Cruz

Dedico este trabajo a mi mujer Olga que sin su apoyo y su tiempo no hubiera podido seguir adelante.

A mis hijas María y Lucía, por el tiempo que no he podido estar con ellas.

Miguel Ángel Pineda Cruz, Enero de 2012

Índice de Contenidos

Plan de Trabajo

Resumen

El Proyecto final de Carrera (PFC) es una asignatura que está pensada para realizar un trabajo de síntesis de los conocimientos adquiridos en otras asignaturas de la carrera y que requiere ponerles en práctica conjuntamente en un trabajo concreto.

Hoy en día, estamos viviendo una época de integraciones de Cajas y con ellas sus S.I. para intentar ahorrar en costes. Entre las muchas tareas que surgen en una integración, se encuentra la de controlar en qué estado se encuentra dicha integración. Para tal fin, existen en el mercado herramientas propietarias como Microsoft Project 2010 Server que en parte te facilitan dicha labor pero que tienen ciertos inconvenientes como precio de la licencia y que no nos permiten llevar un seguimiento a nivel de artefacto con diferentes cambios de estado y cambio de asignaciones de usuarios.

También hay que apuntar, que existen herramientas no propietarias como GanttProject que nos permite también llevar dicho control, pero que no suelen estar preparadas para trabajar en un ambiente colaborativo y adolece del mismo problema comentado anteriormente respecto a los diferentes cambios de estado y asignaciones de usuario.

Con la realización del presente proyecto intentaremos realizar una herramienta no propietaria que nos permita llevar un histórico de los diferentes cambios de estado y asignaciones por los que ha pasado un artefacto. También nos permitirá obtener una foto del estado de todas las tareas, al igual que hacemos con Microsoft Project 2010 Server.

Por último la aplicación nos permitirá generar diversos informes, comunicaciones opcionales mediante correo a los usuarios y control de los usuarios a la aplicación mediante autorizaciones.

Descripción del Proyecto

En esta sección nos centraremos en la definición de los objetivos del proyecto, de los resultados que esperamos alcanzar y de los riesgos que nos podemos encontrar durante su realización.

Objetivos del Proyecto

Este Proyecto tiene como principal objetivo el desarrollo de una aplicación web que permita llevar el control de una integración de Sistemas de Información (SI). Dicha aplicación permitirá por un lado el registro de usuarios con diferentes perfiles y por otro lado una gestión de tareas.

Además la aplicación nos permitirá generar comunicaciones a los usuarios con los cambios de estados de las tareas que tengan asignadas, así como la generación de informes predefinidos de evolución.

Dicha aplicación será accesible vía web, usará herramientas no propietarias y será independiente de cualquier sistema anfitrión. De esta forma logramos que sea accesible desde cualquier sitio a través de un navegador web, no tendremos dependencias por usar herramientas propietarias y la haremos portable a cualquier plataforma ya sea Windows, Linux, Mac, etc.

Como Objetivos Generales podemos enumerar los siguientes:

- Desarrollo de una aplicación web para trabajo colaborativo
- Uso de tecnologías de desarrollo web estándar, no propietarias y portable

Como Objetivos Específicos podemos enumerar los siguientes:

- Control de una Integración de Sistemas de Información
- Control de Acceso a la aplicación
- Generación de Comunicaciones
- Generación de Informes

Resultados Esperados

A lo largo del Proyecto se deberá llegar a una serie de hitos y documentos que reflejen la progresión y el conocimiento que se va obteniendo:

- Documento con el Plan de Trabajo a seguir a lo largo del proyecto además de las líneas esperadas: documentación, resultados finales y la planificación del mismo. Se trata de este mismo documento.

- Documentación asociada a la instalación y configuración de la aplicación
- Manual de Usuario con el funcionamiento de la aplicación.

Por otro lado, una vez realizado el Proyecto se espera obtener:

- Una aplicación web que nos permita en todo momento reflejar la situación de una integración.
- Unos informes y comunicados normalizados para los usuarios
- Unas interfaces amigables con una información bien estructurada.
- Una información contextual suficiente para que el usuario sepa en todo momento donde se encuentra y que puede hacer.

Análisis de Riesgos

A partir de la información que tenemos para la elaboración del presente Plan de Proyecto, hemos identificado los siguientes riesgos:

RI-01: Requisitos incompletos

Descripción: Todas las acciones que se puedan llevar a cabo en la aplicación deben estar definidas como requisitos.

Impacto: Plazos de Desarrollo

Probabilidad: Alta

Acciones: Revisar los requisitos recogidos; elaborar un prototipo para identificar las posibles acciones que falten.

RI-02: Falta de Definición de los Requisitos

Descripción: Todas las acciones que se puedan llevar a cabo en la aplicación deben estar bien definidas, no debe quedar dudas sobre su comportamiento.

Impacto: Plazos de Desarrollo

Probabilidad: Media

Acciones: Revisar los requisitos recogidos; elaborar un prototipo para identificar las posibles definiciones que falten.

RI-03: Periodos de inactividad

Descripción: Puede haber paradas en el proyecto debido a enfermedad, trabajo o problemas familiares.

Impacto: Plazos de Desarrollo

Probabilidad: Baja

Acciones: Recuperar las horas perdidas realizando un esfuerzo los fines de semana.

RI-04: Pérdida de documentación

Descripción: El equipo donde se está realizando el proyecto se puede estropear por diversos motivos.

Impacto: Plazos de Desarrollo

Probabilidad: Baja

Acciones: Realizar copias semanales del trabajo sobre dispositivos externos; instalación de un nuevo equipo.

Alcance de la Propuesta

Se considera dentro del alcance de este proyecto:

- Estudio de las tecnologías de desarrollo web
- Estudio de diseño de interfaces amigables
- Estudio de Herramientas de Prototipaje
- Instalación y Desarrollo de la aplicación antes citada

1. ***Estudio de las tecnologías de desarrollo web***
   Dado que nuestra aplicación se va a ejecutar en un servidor web, será necesario realizar un estudio de las tecnologías web no propietarias que hay actualmente y determinar cuál o cuáles son las más adecuadas para nuestra aplicación.

2. ***Estudio de diseño de interfaces amigables***
   Para que el uso de nuestras interfaces sean lo más amigable posible para el usuario final estudiaremos las recomendaciones actuales que hay en cuanto a Usabilidad Web.

3. *Estudio de Herramientas de Prototipaje*

Para la generación de un prototipo, buscaremos una herramienta que nos permita evaluar además de los requisitos recogidos de la aplicación, la usabilidad de nuestras interfaces.

4. *Instalación y Desarrollo de la aplicación citada*

La aplicación que vamos a desarrollar constará de las siguientes características:

- Monitorización de los estados de las Tareas a medida que se van cambiando.
- Persistencia de la información las Tareas, es necesario almacenar dicha información en Bases de Datos.
- Persistencia de la información de los Usuarios, es necesario almacenar dicha información en Bases de Datos para el control de los usuarios.

Organización del Proyecto

En esta sección cubriremos todas las actividades que tenemos que realizar durante el desarrollo del proyecto, incluiremos el calendario con los trabajos a realizar con los hitos principales del proyecto, el equipo de trabajo junto con sus roles, y por último estableceremos los mecanismos de control para que el proyecto se realice correctamente.

1. *Relación de Actividades*

A continuación describiremos las diferentes actividades que hemos definido para la correcta realización del proyecto. Las dependencias que existen entre las actividades, se pueden consultar de forma gráfica en el fichero de "*Microsoft Project 2007*" que se adjunta con este documento (*MiguelAngelPinedaCruz_Calendario.mpp*).

AC-01: Lanzamiento del Proyecto

Descripción: Esta actividad consideramos que se realiza con la entrada en funcionamiento del aula virtual y los primeros contactos con el tutor para definir el proyecto que al final se va seguir. Estas reuniones ya se han llevado a cabo y se ha definido seguir el proyecto detallado que se muestra en el presente documento.

Duración estimada: 5 días

Participantes: Consultor, Alumno

Predecesoras: Ninguna

AC-02: Gestión y Coordinación del Proyecto

Descripción: abarca todas las acciones que serán necesarias para la correcta marcha del proyecto. En caso de desviación, se aplicarán las medidas correctivas que creamos oportunas.

Duración estimada: Durante todo el Proyecto

Participantes: Alumno

Predecesoras: AC-01: Lanzamiento del Proyecto

AC-03: Planificación y Requisitos

Descripción: esta actividad está formada a su vez por una serie de sub-actividades. Las sub-actividades que engloba son la elaboración del Plan de Trabajo y la Especificación de los Requisitos.

Al final de esta actividad obtendremos los siguientes artefactos:

- Documento con el Plan de Trabajo
- Documento con Planificación de las Tareas (Calendario)
- Documento con las Especificación de los Requisitos

Duración estimada: 26 días

Participantes: Alumno y Consultor

Predecesoras: AC-01: Lanzamiento del Proyecto

AC-03-01: Plan de Trabajo

Descripción: con esta acción definiremos el plan inicial de trabajo. Se trata de este mismo documento.

Duración estimada: 18 días

Participantes: Alumno y Consultor aportando las guías a seguir

Predecesoras: AC-01: Lanzamiento del Proyecto

AC-03-02: Especificación de Requisitos

Descripción: con esta acción enumeraremos los requisitos funcionales y no funcionales de nuestra aplicación.

Duración estimada: 7 días

Participantes: Alumno y Consultor brindando los requisitos y validando el documento final de requisitos necesarios.

Predecesoras: AC-03-01: Plan de Trabajo

AC-04: Estudio de las Tecnologías Web

Descripción: esta actividad está formada a su vez por una serie de sub-actividades. Las sub-actividades que engloba son la elección de una herramienta de Prototipaje, el estudio de las tecnologías web y de la normativa en cuanto a usabilidad web.

Duración estimada: 20 días

Participantes: Alumno

Predecesoras: AC-03: Planificación y Requisitos

AC-04-01: Herramienta de Prototipaje

Descripción: durante esta actividad seleccionaremos una herramienta que nos permita diseñar de forma rápida una página web y su navegación. Esta actividad cubre la selección e instalación de dicha Herramienta.

Duración estimada: 2 días

Participantes: Alumno

Predecesoras: AC-03-02: Especificación de requisitos

AC-04-02: Tecnologías Web

Descripción: durante esta actividad seleccionaremos la tecnología web que mejor se adapte para la implementación de los requisitos funcionales y no funcionales de nuestra aplicación. Durante el desarrollo de esta actividad iremos preparando el entorno de desarrollo.

Duración estimada: 15 días

Participantes: Alumno

Predecesoras: AC-04-01: Herramienta de Prototipaje

AC-04-03: Usabilidad Web

Descripción: durante esta actividad consultaremos diversa bibliografía referente a la usabilidad en el diseño de interfaces para que la experiencia del usuario final sea la adecuada.

Duración estimada: 3 días

Participantes: Alumno

Predecesoras: AC-04-02: Tecnologías Web

AC-04: Análisis, Diseño y Prototipo

Descripción: esta actividad está formada a su vez por una serie de sub-actividades. Las sub-actividades que engloba son definición de los casos de uso, análisis, diseño y creación de prototipo.

Al final de esta actividad obtendremos los siguientes artefactos:

- Documento con los Casos de Uso
- Documento con Análisis de la Aplicación
- Documento con Diseño de la Aplicación
- Implementación de Prototipo

Duración estimada: 46 días

Participantes: Alumno, Consultor

Predecesoras: AC-03: Planificación y Requisitos

AC-04-01: Casos de Uso

Descripción: en esta actividad generaremos todos los casos de uso asociados a los requisitos funcionales detectados en la actividad AC-03-02

Duración estimada: 10 días

Participantes: Alumno, Consultor validando el documento final con los casos de uso

Predecesoras: AC-03-02: Especificación de Requisitos

AC-04-02: Análisis

Descripción: en esta actividad se realiza el análisis de la aplicación a realizar

Duración estimada: 10 días

Participantes: Alumno, Consultor validando el documento final con el análisis de la aplicación

Predecesoras: AC-04-01: Casos de Uso

AC-04-03: Diseño

Descripción: en esta actividad se realiza el diseño de la aplicación a realizar

Duración estimada: 10 días

Participantes: Alumno, Consultor validando el documento final con el diseño de la aplicación

Predecesoras: AC-04-02: Análisis

AC-04-04: Prototipo

Descripción: en esta actividad se realiza el prototipo de la aplicación a realizar

Duración estimada: 16 días

Participantes: Alumno, Consultor validando el prototipo

Predecesoras: AC-04-03: Diseño

AC-05: Entrega Final

Descripción: esta actividad está formada a su vez por una serie de sub-actividades. Las sub-actividades que engloba son la implementación de la aplicación, las pruebas unitarias y de integración, generación de la memoria, y presentación virtual.

Al final de esta actividad obtendremos los siguientes artefactos:

- Código fuente de la aplicación
- Documento con la Memoria del Proyecto y el Manual de Usuario
- Presentación virtual de la aplicación

Duración estimada: 41 días

Participantes: Alumno, Consultor

Predecesoras: AC-04: Análisis, Diseño y Prototipo

AC-05-01: Implementación

Descripción: en esta actividad generaremos la aplicación en sí.

Duración estimada: 30 días

Participantes: Alumno, Consultor validando la implementación

Predecesoras: AC-04-04: Prototipo

AC-05-02: Pruebas Unitarias y de Integración

Descripción: en esta actividad verificaremos que la aplicación funciona correctamente.

Duración estimada: 5 días

Participantes: Alumno

Predecesoras: AC-05-01: Implementación

AC-05-03: Memoria

Descripción: en esta actividad se finaliza la memoria y se genera el manual de usuario.

Duración estimada: 3 días

Participantes: Alumno

Predecesoras: AC-05-02: Pruebas Unitarias y de Integración

AC-05-04: Presentación Virtual

Descripción: en esta actividad se realiza una presentación virtual del funcionamiento de la aplicación.

Duración estimada: 3 días

Participantes: Alumno

Predecesoras: AC-05-03: Memoria

AC-06: Defensa del TFC

Descripción: esta actividad representa la defensa que tendrá que hacer el alumno ante un tribunal (debate virtual) del TFC.

Duración estimada: 1 día

Participantes: Alumno, Tribunal

Predecesoras: AC-05: Entrega Final

2. *Calendario de Trabajo*

Se puede consultar con mayor detalle el archivo de Microsoft Project: MiguelAngelPinedaCruz_Calendario.mpp donde se puede apreciar claramente todas las características del calendario.

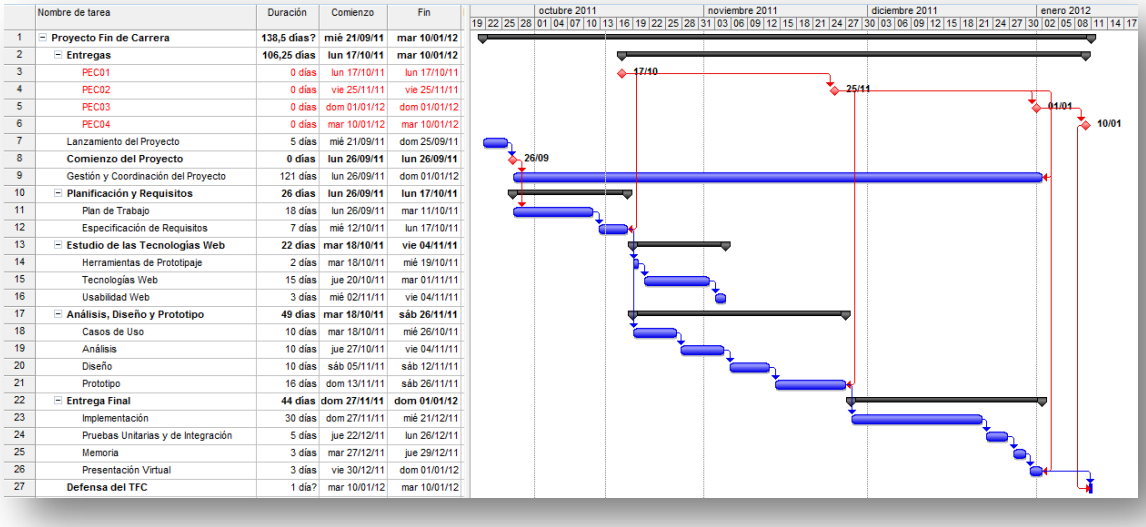

Figura 1: Diagrama de Gantt del TFC

3. *Hitos Principales*

Los Hitos definidos para este TFC, coinciden con las entregas parciales de las PECS

Hito	Descripción	Fecha
1	Propuesta de Proyecto	25/09/2011
2	Plan de Trabajo y Especificación de Requisitos	17/10/2011
3	Análisis, Diseño y Prototipo de la Aplicación	25/11/2011
4	Entrega Final	01/01/2012
5	Defensa del TFC	10/01/2012

Tabla 1: Hitos del TFC

4. *Equipo de Trabajo*

El equipo de trabajo de este proyecto se compone de dos personas:

- Consultor del proyecto: Fatos Xhafa que hará las veces de usuario final y validará que se ha cumplido todos los hitos que se han marcado para el proyecto y dando soporte para obtener la solución final de las diferentes fases.

- Alumno: Miguel Ángel Pineda Cruz que asumirá el resto de roles dentro de un equipo de trabajo.

5. *Definición de Roles*

En la siguiente tabla se recoge los roles que cada persona del equipo de trabajo desempeña en el Proyecto.

Rol	Persona asignada
Coordinador del Proyecto	Fatos Xhafa
Jefe de Proyecto	Miguel Ángel Pineda Cruz
Analista	Miguel Ángel Pineda Cruz
Arquitecto	Miguel Ángel Pineda Cruz
Desarrollador	Miguel Ángel Pineda Cruz
Usuario Final	Miguel Ángel Pineda Cruz

Tabla 2: Roles

6. *Mecanismos de Control*

Para la consecución del objetivo final del Proyecto se establecen los siguientes mecanismos de control:

- El primer mecanismo de control serán los propios hitos que hemos establecidos como principales para el proyecto y que coinciden con las entregas parciales de las pecs.
- Como segundo mecanismo de control y de verificación del trabajo realizado se le pasarán borradores y guías de documentación al consultor para que valide antes de las entregas definitivas que todo marcha en el sentido adecuado. Este control se podrá llevar a cabo cada semana o 10 días máximos dependiendo por supuesto de la tarea en que se encuentre involucrado el proyecto en cada momento y su criticidad.

Valoración Económica

El coste del proyecto se presupone nulo debido a que en todo momento es posible utilizar de forma adecuada software gratuito y de libre distribución; tanto para el entorno de producción (base de datos, tecnología y servidor de aplicaciones) como para el entorno de desarrollo (editores de código, visuales, entorno de trabajo...). Las horas dedicadas a la elaboración y seguimiento del mismo forman parte del calendario del PFC asignado.

Especificación de Requisitos

En este apartado avanzaremos en el análisis de la aplicación que vamos a realizar, describiendo los actores y los casos de uso más importantes.

Actores

En nuestra aplicación podemos distinguir claramente tres tipos de actores:

- ***Administrador***: Se encargará de las tareas de configuración de la aplicación, de la gestión de usuarios, de la generación de comunicados y de la definición de cada proyecto. Al menos debe existir uno en el sistema.

- ***Usuario Avanzado***: Se encargará de definir los artefactos que hay en cada proyecto, de la generación de comunicados e informes.

- ***Usuario Normal***: Se encargará de realizar el seguimiento de los artefactos que tenga asignado, de la generación de comunicados e informes.

Si representamos mediante un diagrama la jerarquía de herencia que existe entre estos tres actores obtenemos la siguiente representación gráfica:

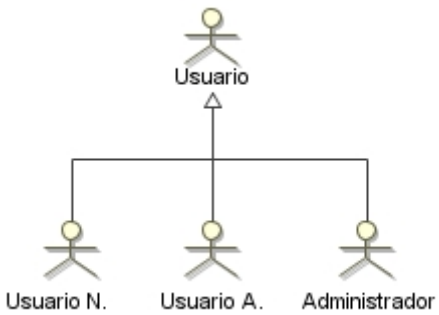

Figura 2: Jerarquía de Herencia

Casos de Uso

A continuación vamos a presentar los casos de usos que se han detectado a partir del detalle de los requisitos funcionales de la aplicación. Los casos de usos que hemos considerado son:

- CU_00:Autorización a la aplicación
  - CU_01:Identificarse en la aplicación
  - CU_02:Salir de la aplicación

- CU_10:Gestión de Usuarios
  - CU_11:Alta de Usuario
  - CU_12:Listar Usuario
  - CU_13:Criterio de Búsqueda
  - CU_14:Desbloquear Usuario
  - CU_15:Modificar Datos de Usuario
  - CU_16:Baja de Usuario

- CU_20:Gestión de Proyectos
  - CU_201:Alta de Proyecto
  - CU_202:Crear a partir de
  - CU_203:Listar Proyecto
  - CU_204:Criterios de Búsqueda
  - CU_205:Asignación de Rol-Proyecto
  - CU_206:Asignación de Estado-Proyecto
  - CU_207:Definición de WorkFlow
  - CU_208:Modificar datos del Proyecto
  - CU_209:Listar Usuarios-Proyecto
  - CU_210:Asignación de Usuarios-Roles
  - CU_211:Baja de Proyecto

- CU_30:Gestión de Artefactos
  - CU_31:Alta de Artefacto
  - CU_32:Listar Artefacto
  - CU_33:Criterios de Búsqueda
  - CU_34:Modificar Datos Artefacto
  - CU_35:Baja de Artefacto
  - CU_36:Cambiar de Estado/Asignar Artefacto
  - CU_37:Consulta Histórico de Artefacto
  - CU_38:Adjuntar Documento

- CU_40:Gestión de Comunicaciones
  - CU_41:Comunicar datos a Usuario
  - CU_42:Comunicar cambio de Estado

- CU_50:Informes
  - CU_51:Solicitud de Informe
  - CU_52:Generar Informe Completo de un Proyecto

- CU_53:Generar Resumen de un Proyecto
- CU_54:Generar Informe Evolución de un Proyecto

- CU_60:Configuración Aplicación
  - CU_61:Parametrización Correo
  - CU_62:Parametrización Usuario

Descripción textual de los casos de uso

En este párrafo vamos a describir detalladamente los casos de usos presentados anteriormente.

CU_00: Autorización a la aplicación

En el siguiente diagrama presentamos los casos de uso que están relacionados con la autorización a la aplicación.

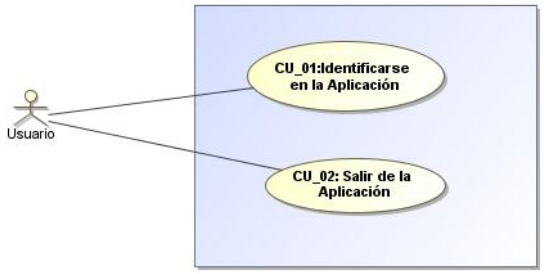

Figura 3: Autorización a la Aplicación

CU_01:Identificarse a la Aplicación

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso, cada vez que un usuario quiera acceder a la aplicación.

Actores: Usuario registrado.

Precondiciones: no estar identificado en el sistema.

Postcondiciónes: el usuario accederá a la aplicación y se le mostrará la pantalla "*Área de Trabajo*" cuyo contenido dependerá del tipo de usuario.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El usuario indica al sistema que quiere acceder a la aplicación.
2. El sistema muestra la pantalla "*Inicio*" de la aplicación.
3. El usuario registrado introduce sus datos de acceso.
4. El sistema comprueba que los datos son correctos.
5. El sistema actualiza el número de intentos a cero.

6. El sistema determina el tipo de actor.
7. El sistema muestra la pantalla "*Área de Trabajo*" cuyo contenido dependerá del tipo de actor.

Flujos alternativos:

4.1. El usuario no está registrado.

4.1.1. El sistema muestra al usuario la pantalla "*Inicio*" que contendrá un mensaje de error con la descripción de dicho error.

4.1.2 Finaliza la ejecución de este caso.

4.2. El usuario registrado está bloqueado.

4.2.1. La aplicación muestra al usuario registrado la pantalla "*Inicio*" que contendrá un mensaje de error con la descripción de dicho error.

4.2.2. Finaliza la ejecución de este caso.

4.3. Los datos están incompletos.

4.3.1. El sistema muestra al usuario registrado la pantalla "*Inicio*" que contendrá un mensaje de error con los campos que son obligatorios.

4.3.2. La ejecución del caso continúa en el punto 3.

4.4. Los datos no son válidos.

4.4.1. El sistema actualiza el número de intentos.

4.4.2. El sistema muestra al usuario registrado la pantalla "*Inicio*" que contendrá un mensaje de error con la descripción de dicho error.

4.4.3. La ejecución del caso continúa en el punto 3.

4.5. Error no esperado de la aplicación

4.5.1. El sistema muestra al usuario registrado la pantalla "*Inicio*" que contendrá un mensaje de error con la descripción de dicho error.

4.5.2. Finaliza la ejecución de este caso.

Observaciones:

- ***El número de intentos de acceso permitidos será uno de los parámetros configurable por el administrador.***

CU_02: Salir de la Aplicación

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando un usuario registrado quiera cerrar su sesión con la aplicación.

Actores: Administrador, Usuario Avanzado, Usuario Normal.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: al usuario se le muestra la pantalla de "*Inicio*" de la aplicación.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El usuario registrado indica al sistema que quiere cerrar la sesión con la aplicación.
2. El sistema cierra la sesión del usuario con la aplicación.
3. La aplicación muestra la pantalla "*Inicio*".

Flujos alternativos:

2.1. Error no esperado de la aplicación

2.1.1. El sistema muestra al usuario registrado un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso

CU_10: Gestión de Usuarios

En el siguiente diagrama presentamos los casos de uso que están relacionados con la gestión de usuarios de la aplicación.

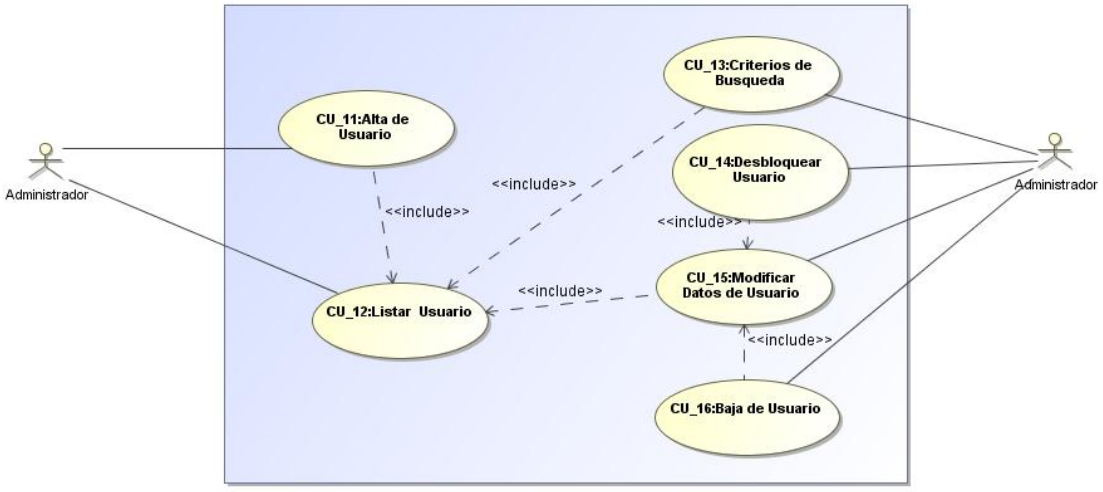

Figura 4: Gestión de Usuarios

CU_11: Alta de Usuario

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera dar de alta a un usuario.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: nuevo usuario registrado en la aplicación.

Casos de usos relacionados: CU_12: Listar Usuario, CU_41: Comunicar datos a Usuario.

Escenario principal:

1. El Administrador indica al sistema que quiere registrar un nuevo usuario en la aplicación desde la pantalla "*Listado de Usuarios*".
2. El sistema muestra la pantalla "*Datos de Usuario*" de la aplicación.
3. El Administrador introduce la información del nuevo usuario.
4. El sistema valida los datos introducidos.
5. El sistema registra al usuario.
6. El sistema genera comunicado al usuario.
7. El sistema muestra la pantalla "*Datos de Usuario Correcto*".

Flujos alternativos:

4.1. El usuario ya está registrado.

4.1.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario"* que contendrá un mensaje de error con la descripción de dicho error.

4.1.2. La ejecución del caso continúa en el punto 3.

4.2. Los datos están incompletos

4.2.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con los campos que son obligatorios.

4.2.2. La ejecución del caso continúa en el punto 3.

4.3. Los datos no son válidos.

4.3.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

4.3.2. La ejecución del caso continúa en el punto 3.

4.4. Error no esperado de la aplicación.

4.4.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

4.4.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

6.1. El Administrador indica que no se genere comunicado.

6.1.1. La ejecución del caso continúa en el punto 7.

Observaciones:

- ***El Administrador decidirá si comunica mediante email los datos de acceso a la aplicación del usuario.***
- ***El Administrador decidirá si bloquea o no la cuenta del usuario durante su registro en la aplicación.***
- ***En la aplicación no se permite tener dados de altas dos usuarios con el mismo login o mail.***

CU_12: Listar Usuario

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera consultar los usuarios registrados de la aplicación.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: se obtiene lista con los usuarios registrados de la aplicación.

Casos de usos relacionados: CU_13: Criterios de Búsqueda.

Escenario principal:

1. El Administrador indica al sistema que quiere consultar los usuarios registrados en la aplicación desde la pantalla *"Área de Trabajo"*.
2. El sistema muestra al Administrador la pantalla "*Listado de Usuarios*" de la aplicación.
3. El Administrador introduce los criterios de búsqueda.
4. El sistema ejecuta la consulta para obtener todos los usuarios registrados en la aplicación.
5. El sistema ejecuta el caso de uso CU_13: Criterios de Búsqueda sobre el conjunto de datos obtenidos en el paso 4.
6. El sistema muestra la pantalla *"Listado de Usuarios"* con los datos de usuarios del paso 5.

Flujos alternativos:

4.1. Error no esperado de la aplicación.

4.1.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios*" que contendrá un mensaje de error con la descripción de dicho error.

4.1.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

6.1. No hay datos que cumplan los criterios.

6.1.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios*" que contendrá un aviso con la descripción de no hay datos.

6.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Los criterios de búsqueda que usaremos serán por login o email del usuario.***
- ***Al aplicar los criterios usaremos la comparación "que contenga el valor".***
- ***Si no se informa ningún criterio de filtrado, se obtendrá todos los usuarios registrados en la aplicación.***

CU_13: Criterios de Búsqueda

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera aplicar un filtro sobre los datos de los usuarios.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: datos de usuarios filtrados.

Casos de usos relacionados: CU_12: Listar Usuario.

Escenario principal:

1. El Administrador indica al sistema que quiere aplicar un filtrado sobre los datos de los usuarios.
2. El sistema ejecuta el filtro sobre los datos de los usuarios.
3. La aplicación devuelve los datos filtrados de los usuarios.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Al aplicar el filtrado puede ocurrir que no haya datos que devolver.***

CU_14: Desbloquear Usuario

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera desbloquear un usuario registrado.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_15: Modificar Datos de Usuario. Debe de existir al menos un usuario registrado bloqueado.

Postcondiciónes: usuario registrado desbloqueado.

Casos de usos relacionados: CU_15: Modificar Datos de Usuario.

Escenario principal:

1. El Administrador indica al sistema que quiere desbloquear un usuario que se encuentra en estado bloqueado en la pantalla "*Datos de Usuario*".
2. El sistema valida el estado actual del usuario seleccionado
3. El sistema actualiza el estado del usuario seleccionado ha desbloqueado.
4. El sistema muestra al Administrador la pantalla "*Datos de Usuario Correcto*" con la información actualizada del estado del usuario seleccionado (desbloqueado).

Flujos alternativos:

2.1. Usuario no bloqueado.

2.1.1. La ejecución del caso continúa en el punto 4.

2.2. Error no esperado de la aplicación.

2.2.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

2.2.2. Finaliza la ejecución de este caso.

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso

CU_15: Modificar datos de Usuario

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera modificar la información de un usuario registrado.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_12: Listar Usuario. Debe de existir al menos un usuario registrado.

Postcondiciónes: CU_12: Listar Usuario, CU_14: Desbloquear Usuario, CU_16: Baja de Usuario.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El Administrador indica al sistema que quiere modificar la información de un usuario en la pantalla "*Listado de Usuarios*".
2. El sistema consulta la información actual del usuario seleccionado.
3. El sistema muestra al Administrador la pantalla *"Datos de Usuario"*.
4. El Administrador modifica la información del usuario.
5. La aplicación valida los datos introducidos.
6. La aplicación actualiza la información del usuario.
7. La aplicación genera comunicado al usuario.
8. La aplicación muestra la pantalla "*Datos de Usuario Correcto*".

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios*" que contendrá un mensaje de error con los campos que son obligatorios.

2.1.2. Finaliza la ejecución de este caso.

4.1. El Administrador no modifica la información del usuario.

4.1.1. Finaliza la ejecución de este caso.

5.1. Los datos están incompletos

5.1.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con los campos que son obligatorios.

5.1.2. La ejecución del caso continúa en el punto 4.

5.2. Los datos no son válidos.

5.2.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

5.2.2. La ejecución del caso continúa en el punto 4.

5.3. Error no esperado de la aplicación.

5.3.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

5.3.2. Finaliza la ejecución de este caso.

6.1. Error no esperado de la aplicación.

6.1.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

6.1.2. Finaliza la ejecución de este caso.

7.1. El Administrador indica que no se genere comunicado.

7.1.1. La ejecución del caso continúa en el punto 8.

CU_16: Baja de Usuario

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera dar de baja un usuario registrado.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_15: Modificar Datos de Usuario. Debe de existir al menos un usuario registrado y no dado de baja.

Postcondiciónes: CU_15: Modificar Datos de Usuario.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El Administrador indica al sistema que quiere dar de baja a un usuario que se encuentra en la pantalla "*Datos de Usuario*".
2. El sistema da de baja al usuario.
3. El sistema muestra al Administrador la pantalla "*Listado de Usuarios*" con la información actualizada de los usuarios registrados (activos).

Flujos alternativos:

1.1. El Administrador no confirma la baja del usuario

1.1.1. Finaliza la ejecución de este caso

2.1. Usuario dado de baja.

2.1.1. La ejecución del caso continúa en el punto 3.

2.2. Error no esperado de la aplicación.

2.2.1. El sistema muestra al Administrador la pantalla "*Datos de Usuario*" que contendrá un mensaje de error con la descripción de dicho error.

2.2.2. Finaliza la ejecución de este caso.

CU_20: Gestión de Proyectos

En el siguiente diagrama presentamos los casos de uso que están relacionados con la gestión de proyectos de la aplicación.

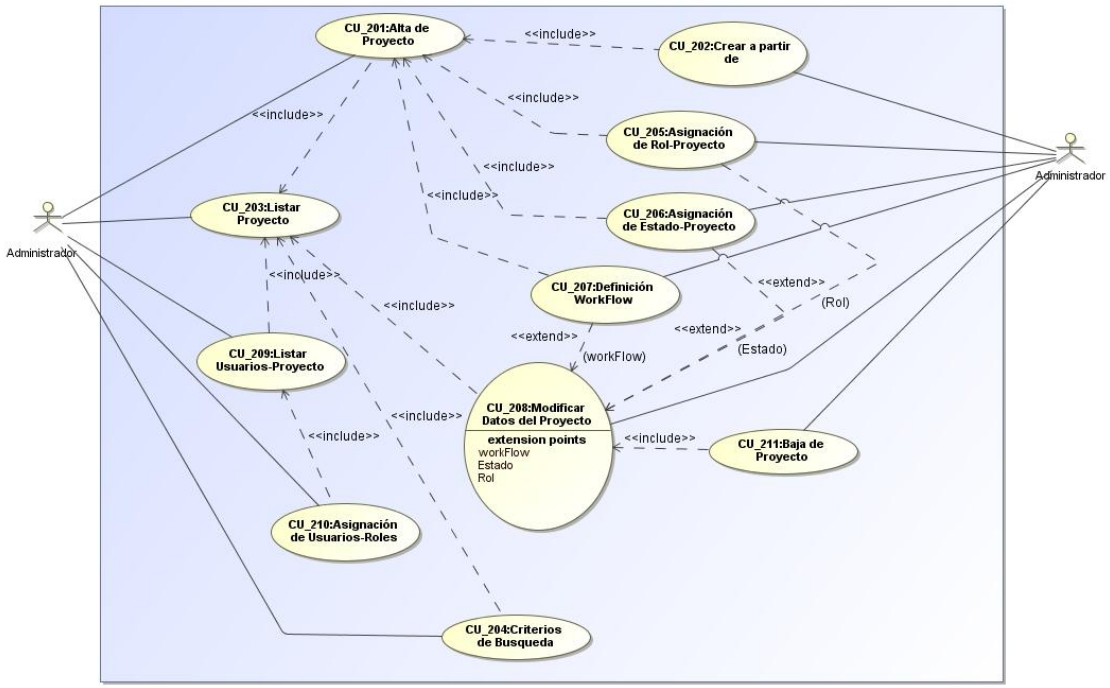

Figura 5: Gestión de Proyectos

CU_201: Alta de Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un Administrador quiera dar de alta un nuevo proyecto.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_203: Listar Proyecto.

Postcondiciónes: definición de nuevo proyecto.

Casos de usos relacionados: CU_203: Listar Proyecto**,** CU_202: Crear a partir de, CU_205: Asignación de Rol-Proyecto, CU_206: Asignación de Estado-Proyecto, CU_207: Definición de WorkFlow.

Escenario principal:

1. El Administrador indica al sistema que quiere definir un nuevo proyecto en la aplicación desde la pantalla *"Listado de Proyectos"*.
2. El sistema muestra la pantalla "*Datos de Proyecto*" de la aplicación.
3. El Administrador introduce la información del proyecto.
4. El sistema valida los datos introducidos.
5. El sistema registra el proyecto.

6. El sistema muestra la pantalla "*Resumen de Proyecto Correcto*"

Flujos alternativos:

4.1. El proyecto ya está definido.

4.1.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto"* que contendrá un mensaje de error con la descripción de dicho error.

4.1.2. La ejecución del caso continúa en el punto 3.

4.2. Los datos están incompletos.

4.2.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con los campos que son obligatorios.

4.2.2. La ejecución del caso continúa en el punto 3.

4.3. Los datos no son válidos.

4.3.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

4.3.2. La ejecución del caso continúa en el punto 3.

4.4. Error no esperado de la aplicación.

4.4.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

4.4.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***En la aplicación no se permite tener dados de altas dos proyectos con el mismo nombre o clave.***

CU_202: Crear a partir de

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un Administrador quiera definir un nuevo Proyecto a partir de otro.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_203: Listar Proyecto.

Postcondiciónes: definición de nuevo proyecto.

Casos de usos relacionados: CU_201: Alta de Proyecto.

Escenario principal:

1. El Administrador indica al sistema que quiere definir un nuevo proyecto en la aplicación a partir de otro que ya está definido desde la pantalla "*Datos de Proyecto*".
2. El sistema consulta los Proyectos definidos en la aplicación.
3. El Administrador introduce la información del proyecto nuevo y selecciona el proyecto a partir del cual se definirá su configuración (roles, estados, WorkFlow).
4. El sistema valida los datos introducidos.
5. El sistema registra el proyecto.
6. El sistema muestra la pantalla "*Resumen de Proyecto Correcto*".

Flujos alternativos:

4.1. El proyecto ya está definido.

4.1.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto"* que contendrá un mensaje de error con la descripción de dicho error.

4.1.2. La ejecución del caso continúa en el punto 3.

4.2. Los datos están incompletos

4.2.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con los campos que son obligatorios.

4.2.2. La ejecución del caso continúa en el punto 3.

4.3. Los datos no son válidos.

4.3.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

4.3.2. La ejecución del caso continúa en el punto 3.

4.4. Error no esperado de la aplicación.

4.4.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

4.4.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Administrador la pantalla "*Datos de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***En la aplicación no se permite tener dados de altas dos proyectos con el mismo nombre o clave.***

CU_203: Listar Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un Administrador quiera consultar los proyectos definidos de la aplicación.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: se obtiene lista con los proyectos definidos de la aplicación.

Casos de usos relacionados: CU_204: Criterios de Búsqueda, CU_201: Alta de Proyecto, CU_208: Modificar Datos del Proyecto.

Escenario principal:

1. El Administrador indica al sistema que quiere consultar los proyectos definidos en la aplicación desde la pantalla *"Área de Trabajo"*.
2. El sistema muestra la pantalla "*Listado de Proyectos*" de la aplicación.
3. El Administrador introduce los criterios de búsqueda.
4. El sistema ejecuta la consulta para obtener todos los proyectos definidos en la aplicación.
5. El sistema ejecuta el caso de uso CU_203: Criterios de Búsqueda sobre el conjunto de datos obtenidos en el paso 4.
6. El sistema muestra la pantalla *"Listado de Proyectos"* con los datos de usuarios del paso 5.

Flujos alternativos:

4.1. Error no esperado de la aplicación.

4.1.1. El sistema muestra al Administrador la pantalla "*Listado de Proyectos*" que contendrá un mensaje de error con la descripción de dicho error.

4.1.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Administrador la pantalla "*Listado de Proyectos*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

6.1. No hay datos que cumplan los criterios.

6.1.1. El sistema muestra al Administrador la pantalla "*Listado de Proyectos*" que contendrá un aviso con la descripción de no hay datos.

6.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Los criterios de búsqueda que usaremos serán por nombre o clave del proyecto.***
- ***Al aplicar los criterios usaremos la comparación "que contenga el valor".***
- ***Si no se informa ningún criterio de filtrado, se obtendrá todos los proyectos definidos en la aplicación.***

CU_204: Criterios de Búsqueda

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera aplicar un filtro sobre los proyectos definidos.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: datos de proyectos definidos.

Casos de usos relacionados: CU_203: Listar Proyecto.

Escenario principal:

1. El Administrador indica al sistema que quiere aplicar un filtrado sobre los datos de los proyectos definidos.
2. El sistema ejecuta el filtro sobre los datos de los proyectos definidos.
3. El sistema devuelve los datos filtrados de los proyectos definidos.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Al aplicar el filtrado puede ocurrir que no haya datos que devolver.***

CU_205: Asignación de Rol-Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera definir los roles permitido en un Proyecto.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01: Identificarse a la Aplicación y CU_201: Alta de Proyecto.

Postcondiciónes: Conjunto de roles permitidos en un Proyecto.

Casos de usos relacionados: CU_201: Alta de Proyecto, CU_208: Modificar Datos del Proyecto.

Escenario principal:

1. El sistema consulta los roles actuales
2. El sistema muestra al Administrador la pantalla "*Roles Proyecto*" con los Roles Asignados
3. El Administrador indica al sistema que quiere dar de Alta un rol en un Proyecto desde la pantalla "*Roles Proyecto*".

4. El Administrador introduce la información del rol.
5. El sistema valida la información del rol
6. El sistema registra la información del rol
7. El sistema muestra al Administrador la pantalla "*Roles Proyecto*" con la información actualizada de los roles asignados al Proyecto.

Flujos alternativos:

1.1. Error no esperado de la aplicación.

1.1.1. El sistema muestra al Administrador la pantalla "*Roles Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

1.1.2. Finaliza la ejecución de este caso.

3.1. El Administrador indica al sistema que quiere dar de Baja un rol en un Proyecto desde la pantalla "*Roles Proyecto*".

3.1.1. El Administrador selecciona el rol.

3.1.2. El sistema da de baja el rol.

3.1.3. La ejecución del paso continúa en el punto 7.

3.2. El Administrador indica al sistema que quiere modificar un rol en un Proyecto desde la pantalla "*Roles Proyecto*".

3.2.1. El Administrador selecciona el rol

3.2.2. El Administrador modifica la información

3.2.3. La ejecución del paso continúa en el punto 5.

3.1.2. Error no esperado de la aplicación.

3.1.2.1. El sistema muestra al Administrador la pantalla "*Roles Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2.2. Finaliza la ejecución de este caso

5**.1.** El Rol ya está definido.

5.1.1. El sistema muestra al Administrador la pantalla "*Roles Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

6. Error no esperado de la aplicación.

6.1. El sistema muestra al Administrador la pantalla "*Roles Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

6.2. Finaliza la ejecución de este caso

Observaciones:

- ***El sistema usará el nombre del rol como criterio de búsqueda.***

CU_206: Asignación de Estado-Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera definir los estados permitido para un artefacto en un Proyecto.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01: Identificarse a la Aplicación y CU_201: Alta de Proyecto.

Postcondiciónes: Conjunto de estados permitidos para un artefacto en un Proyecto.

Casos de usos relacionados: CU_201: Alta de Proyecto, CU_208: Modificar Datos del Proyecto.

Escenario principal:

1. El sistema consulta los estados actuales
2. El sistema muestra al Administrador la pantalla "*Estados Proyecto*" con los Estados Asignados
3. El Administrador indica al sistema que quiere dar de Alta un nuevo estado en un Proyecto desde la pantalla "*Estados Proyecto*".
4. El Administrador introduce la información del estado.
5. El sistema valida la información del estado
6. El sistema registra la información del estado
7. El sistema muestra al Administrador la pantalla "*Estados Proyecto*" con la información actualizada de los estados asignados al Proyecto.

Flujos alternativos:

1.1. Error no esperado de la aplicación.

1.1.1. El sistema muestra al Administrador la pantalla "*Estados Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

1.1.2. Finaliza la ejecución de este caso.

3.1. El Administrador indica al sistema que quiere dar de Baja un estado en un Proyecto desde la pantalla "*Estados Proyecto*".

3.1.1. El Administrador selecciona el estado.

3.1.2. El sistema da de baja el estado.

3.1.3. La ejecución del paso continúa en el punto 7.

3.2. El Administrador indica al sistema que quiere modificar un estado en un Proyecto desde la pantalla "*Estados Proyecto*".

3.2.1. El Administrador selecciona el estado.

3.2.2. El Administrador modifica la información

3.2.3. La ejecución del paso continúa en el punto 5.

3.1.2. Error no esperado de la aplicación.

3.1.2.1. El sistema muestra al Administrador la pantalla "*Estados Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2.2. Finaliza la ejecución de este caso

5**.1.** El Estado ya está definido.

5.1.1. El sistema muestra al Administrador la pantalla "*Estados Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

6. Error no esperado de la aplicación.

6.1. El sistema muestra al Administrador la pantalla "*Estados Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

6.2. Finaliza la ejecución de este caso

Observaciones:

- ***El sistema usará el nombre del estado como criterio de búsqueda.***

CU_207: Definición de WorkFlow

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera definir el WorkFlow permitido para los artefactos de un Proyecto.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01: Identificarse a la Aplicación y CU_201: Alta de Proyecto.

Postcondiciónes: Conjunto de estados permitidos para un artefacto en un Proyecto.

Casos de usos relacionados: CU_201: Alta de Proyecto, CU_208: Modificar Datos del Proyecto.

Escenario principal:

1. El sistema consulta los Roles asignados a un Proyecto
2. El sistema consulta los Estados asignados a un Proyecto
3. El sistema consulta el WorkFlow definido en un Proyecto
4. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" con los datos del WorkFlow definido en el Proyecto.
5. El Administrador indica al sistema que quiere dar de Alta un nuevo paso en el WorkFlow del Proyecto.
6. El Administrador selecciona el estado y rol origen.
7. El Administrador selecciona el estado destino.
8. El sistema valida la información del nuevo paso del WorkFlow.
9. El sistema registra la información del nuevo paso del WorkFlow.
10. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" con la información actualizada del WorkFlow del Proyecto.

Flujos alternativos:

1.1. Error no esperado de la aplicación.

1.1.1. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

1.1.1. Finaliza la ejecución de este caso.

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso.

5.1. El Administrador indica al sistema que quiere dar de Baja un paso del WorkFlow en un Proyecto desde la pantalla "*WorkFlow Proyecto*".

5.1.1. El Administrador selecciona el paso del WorkFlow que quiere borrar.

5.1.2. El sistema da de baja el paso del WorkFlow.

5.1.3. La ejecución del paso continúa en el punto 10.

5.2. El Administrador indica al sistema que quiere modificar un paso del WorkFlow en un Proyecto desde la pantalla "*WorkFlow Proyecto*".

5.2.1. El Administrador selecciona el paso del WorkFlow.

5.2.2. El Administrador modifica la información del paso del WorkFlow.

5.2.3. La ejecución del paso continúa en el punto 8.

5.1.2. Error no esperado de la aplicación.

5.1.2.1. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2.2. Finaliza la ejecución de este caso

8.1. El paso ya está definido.

8.1.1. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

8.1.2. Finaliza la ejecución de este caso.

9. Error no esperado de la aplicación.

9.1. El sistema muestra al Administrador la pantalla "*WorkFlow Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

9.2. Finaliza la ejecución de este caso.

CU_208: Modificar datos del Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera modificar la información de un proyecto definido.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_203: Listar Proyecto. Debe de existir al menos un proyecto definido.

Postcondiciónes: datos de proyecto modificado.

Casos de usos relacionados: CU_203: Listar Proyecto, CU_205: Asignación de Rol-Proyecto, CU_206: Asignación de Estado-Proyecto, CU_207: Definición WorkFlow.

Escenario principal:

1. El Administrador indica al sistema que quiere modificar la información de un proyecto en la pantalla "*Listado de Proyecto*".
2. El sistema consulta la información actual del proyecto seleccionado.
3. El sistema muestra al Administrador la pantalla *"Resumen de Proyecto"*.
4. El Administrador modifica la información del proyecto.
5. La aplicación valida los datos introducidos.
6. La aplicación actualiza la información del proyecto.
7. La aplicación muestra la pantalla "*Resumen de Proyecto Correcto*".

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador la pantalla "*Resumen de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

4.1. El Administrador no modifica la información del proyecto.

4.1.1. Finaliza la ejecución de este caso.

5.1. Los datos están incompletos

5.1.1. El sistema muestra al Administrador la pantalla "*Resumen de Proyecto*" que contendrá un mensaje de error con los campos que son obligatorios.

5.1.2. La ejecución del caso continúa en el punto 4.

5.2. Los datos no son válidos.

5.2.1. El sistema muestra al Administrador la pantalla "*Resumen de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

5.2.2. La ejecución del caso continúa en el punto 4.

5.3. Error no esperado de la aplicación.

5.3.1. El sistema muestra al Administrador la pantalla "*Resumen de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

5.3.2. Finaliza la ejecución de este caso.

6.1. Error no esperado de la aplicación.

6.1.1. El sistema muestra al Administrador la pantalla "*Resumen de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

6.1.2. Finaliza la ejecución de este caso.

CU_209: Listar Usuarios-Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un Administrador quiera consultar los usuarios asignados a un proyecto definido en la aplicación.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_203: Listar Proyecto. Debe de existir al menos un proyecto definido.

Postcondiciónes: se obtiene lista con los usuarios asignados a un proyecto definido en la aplicación.

Casos de usos relacionados: CU_203: Listar Proyecto, CU_210: Asignación de Usuarios-Roles.

Escenario principal:

1. El Administrador indica al sistema que quiere consultar los usuarios asignados a un proyecto definidos en la aplicación mediante la selección de un proyecto en la pantalla "*Listado de Proyectos*".
2. El sistema ejecuta la consulta para obtener todos los usuarios asignados al proyecto que ha indicado el Administrador.
3. El sistema muestra la pantalla *"Listado de Usuarios Proyecto"* con los datos de los usuarios asignados a un Proyecto.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador la pantalla "*Listado de Proyectos*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

3.1. No hay usuarios asignados.

3.1.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios Proyectos*" que contendrá un aviso con la descripción de no hay datos.

3.1.2. Finaliza la ejecución de este caso.

CU_210: Asignación de Usuarios-Roles

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera asignar un usuario a un proyecto definido con un rol determinado.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01: Identificarse a la Aplicación y CU_209: Listar Usuarios-Proyecto.

Postcondiciónes: lista actualizada de usuarios asignados a un Proyecto.

Casos de usos relacionados: CU_209: Listar Usuarios-Proyecto.

Escenario principal:

1. El Administrador indica al sistema que quiere dar de Alta un usuario registrado con un determinado rol en un proyecto desde la pantalla "*Listado de Usuarios Proyecto*".
2. El Administrador introduce la información del usuario registrado y su rol en la pantalla "*Listado de Usuarios Proyecto*".
3. El sistema valida la información del usuario y su rol para un proyecto
4. El sistema registra la información del usuario y su rol para un proyecto
5. El sistema muestra al Administrador la pantalla "*Listado de Usuarios Proyecto*" con la información actualizada de los usuarios y roles asignados al Proyecto.

Flujos alternativos:

1.1. El Administrador indica al sistema que quiere dar de Baja un determinado usuario con un rol asignado en un Proyecto desde la pantalla "*Listado de Usuarios Proyecto*".

1.1.1. El Administrador selecciona un usuario-rol de la pantalla "*Listado de Usuarios Proyecto*".

1.1.2. El sistema da de baja el par usuario, rol seleccionado en el proyecto.

1.1.3. La ejecución del paso continúa en el punto 5.

1.1.2. Error no esperado de la aplicación.

1.1.2.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

1.1.2.2. Finaliza la ejecución de este caso

3**.1. E**l par usuario, rol ya existe en el proyecto.

3.1.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso.

4. Error no esperado de la aplicación.

4.1. El sistema muestra al Administrador la pantalla "*Listado de Usuarios Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

4.2. Finaliza la ejecución de este caso

CU_211: Baja de Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera eliminar un proyecto previamente definido en la aplicación.

Actores: Administrador.

Precondiciones: Se han realizado correctamente los casos de uso CU_01: Identificarse a la Aplicación y CU_208: Modificar Datos del Proyecto.

Postcondiciónes: baja de un proyecto previamente definido.

Casos de usos relacionados: CU_208: Modificar Datos del Proyecto.

Escenario principal:

1. El Administrador indica al sistema que quiere dar de Baja un determinado proyecto previamente definido desde la pantalla "*Resumen de Proyecto*".
2. El sistema actualiza la información del proyecto indicado.
3. El sistema recupera la información de los proyectos definidos.
4. El sistema muestra al Administrador la pantalla "*Listado de Proyectos*" con la información actualizada de los proyectos definidos.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador la pantalla "*Resumen de Proyecto*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Administrador la pantalla "*Resumen de Proyectos*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso

CU_30: Gestión de Artefactos

En el siguiente diagrama presentamos los casos de uso que están relacionados con la gestión de artefactos de un proyecto en la aplicación.

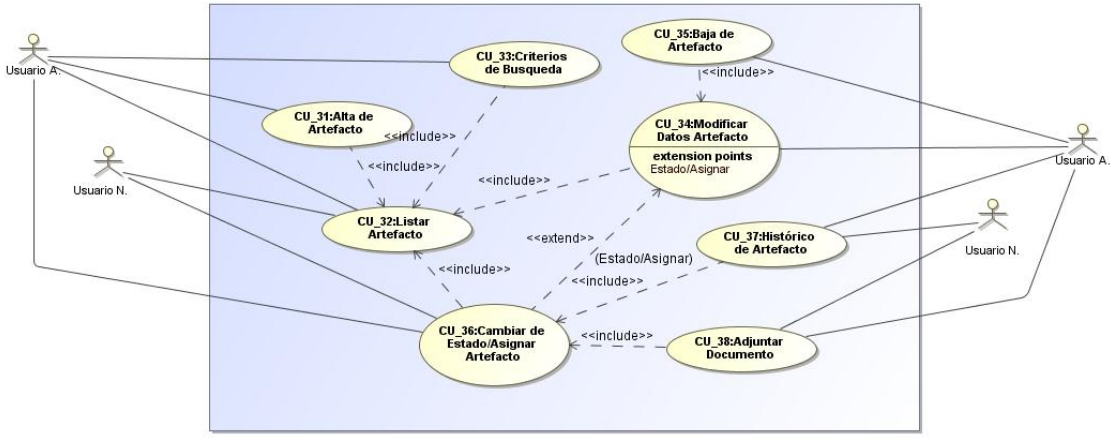

Figura 6: Gestión de Artefactos

CU_31: Alta de Artefacto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario autorizado quiera dar de alta un nuevo artefacto en un proyecto.

Actores: Usuario Autorizado.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_32: Listar Artefacto.

Postcondiciónes: definición de nuevo artefacto en un proyecto.

Casos de usos relacionados: CU_32: Listar Artefacto.

Escenario principal:

1. El Usuario Autorizado indica al sistema que quiere definir un nuevo artefacto en la aplicación desde la pantalla *"Listado de Artefactos"*.
2. El sistema muestra la pantalla "*Datos de Artefacto*" de la aplicación.
3. El Administrador introduce la información del artefacto.
4. El sistema valida los datos introducidos.
5. El sistema registra el artefacto en un proyecto.
6. El sistema muestra la pantalla "*Resumen de Artefacto Correcto*"

Flujos alternativos:

4.1. El artefacto ya está definido en un proyecto.

4.1.1. El sistema muestra al Administrador la pantalla "*Datos de Artefacto"* que contendrá un mensaje de error con la descripción de dicho error.

4.1.2 La ejecución del caso continúa en el punto 3.

4.2. Los datos están incompletos

4.2.1. El sistema muestra al Administrador la pantalla "*Datos de Artefacto*" que contendrá un mensaje de error con los campos que son obligatorios.

4.2.2. La ejecución del caso continúa en el punto 3.

4.3. Los datos no son válidos.

4.3.1. El sistema muestra al Administrador la pantalla "*Datos de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

4.3.2. La ejecución del caso continúa en el punto 3.

4.4. Error no esperado de la aplicación.

4.4.1. El sistema muestra al Administrador la pantalla "*Datos de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

4.4.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Administrador la pantalla "*Datos de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***En la aplicación no se permite tener dados de altas dos artefactos con el mismo nombre dentro de un proyecto.***

CU_32: Listar Artefacto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario autorizado o usuario normal quiera consultar los artefactos definidos en un proyecto de la aplicación.

Actores: Usuario Autorizado, Usuario Normal.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: se obtiene lista con los artefactos definidos en un proyecto de la aplicación.

Casos de usos relacionados: CU_33: Criterios de Búsqueda, CU_31: Alta de Artefacto, CU_34: Modificar Datos del Artefacto.

Escenario principal:

1. El Usuario Autorizado o Usuario Normal indican al sistema que quieren consultar los artefactos definidos en un proyecto de la aplicación desde la pantalla *"Área de Trabajo"*.
2. El sistema muestra la pantalla "*Listado de Artefactos*" de la aplicación.

3. El Usuario Autorizado o Usuario Normal introduce los criterios de búsqueda.
4. El sistema ejecuta la consulta para obtener todos los artefactos definidos en la aplicación.
5. El sistema ejecuta el caso de uso CU_33: Criterios de Búsqueda sobre el conjunto de datos obtenidos en el paso 4.
6. El sistema muestra la pantalla *"Listado de Artefactos"* con los datos de los artefactos de un proyecto del paso 5.

Flujos alternativos:

4.1. Error no esperado de la aplicación.

4.1.1. El sistema muestra al Usuario Autorizado o Usuario Normal la pantalla "*Listado de Artefactos*" que contendrá un mensaje de error con la descripción de dicho error.

4.1.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Usuario Autorizado o Usuario Normal la pantalla "*Listado de Artefactos*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

6.1. No hay datos que cumplan los criterios.

6.1.1. El sistema muestra al Usuario Autorizado o Usuario Normal la pantalla "*Listado de Artefactos*" que contendrá un aviso con la descripción de no hay datos.

6.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Los criterios de búsqueda que usaremos serán por nombre o clave del proyecto.***
- ***Al aplicar los criterios usaremos la comparación "que sea igual a".***
- ***Si no se informa ningún criterio de filtrado, no se obtendrán los artefactos definidos en la aplicación.***

CU_33: Criterios de Búsqueda

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario autorizado o usuario normal quiera aplicar un filtro sobre los artefactos definidos.

Actores: Usuario Autorizado, Usuario Normal.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación. Es necesario que venga informado el nombre o clave del proyecto.

Postcondiciónes: datos de artefactos definidos.

Casos de usos relacionados: CU_32: Listar Artefacto.

Escenario principal:

1. El Usuario Autorizado o Usuario Normal indican al sistema que quieren aplicar un filtrado sobre los datos de los artefactos definidos.
2. El sistema ejecuta el filtro sobre los datos de los artefactos definidos.
3. El sistema devuelve los datos filtrados de los artefactos definidos.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Autorizado o Usuario Normal un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Al aplicar el filtrado puede ocurrir que no haya datos que devolver***

CU_34: Modificar datos Artefacto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario autorizado quiera modificar la información de un artefacto definido en un proyecto.

Actores: Usuario Autorizado.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_32: Listar Artefacto. Debe de existir al menos un artefacto definido en un proyecto.

Postcondiciónes: datos de artefacto modificado.

Casos de usos relacionados: CU_32: Listar Artefacto, CU_35: Baja de Artefacto, CU_36: Cambiar de Estado/Asignar Artefacto.

Escenario principal:

1. El Usuario Autorizado indica al sistema que quiere modificar la información de un artefacto asignado a un proyecto en la pantalla "*Listado de Artefactos*".
2. El sistema consulta la información actual del artefacto seleccionado.
3. El sistema muestra al Usuario Avanzado la pantalla *"Resumen de Artefacto"*.
4. El Usuario Avanzado modifica la información del artefacto.
5. La aplicación valida los datos introducidos.
6. La aplicación actualiza la información del artefacto.
7. La aplicación muestra la pantalla "*Resumen de Artefacto Correcto*".

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.2.1. El sistema muestra al Usuario Autorizado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

2.2.2. Finaliza la ejecución de este caso.

4.1. El Usuario Autorizado no modifica la información del artefacto.

4.1.1. Finaliza la ejecución de este caso.

5.1. Los datos están incompletos

5.1.1. El sistema muestra al Usuario Autorizado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con los campos que son obligatorios.

5.1.2. La ejecución del caso continúa en el punto 4.

5.2. Los datos no son válidos.

5.2.1. El sistema muestra al Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

5.2.2. La ejecución del caso continúa en el punto 4.

5.3. Error no esperado de la aplicación.

5.3.1. El sistema muestra al Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

5.3.2. Finaliza la ejecución de este caso.

6.1. Error no esperado de la aplicación.

6.1.1. El sistema muestra al Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

6.1.2. Finaliza la ejecución de este caso.

CU_35: Baja de Artefacto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario avanzado quiera eliminar un artefacto de un proyecto previamente definido en la aplicación.

Actores: Usuario Avanzado.

Precondiciones: Se han realizado correctamente los casos de uso CU_01: Identificarse a la Aplicación y CU_34: Modificar Datos Artefacto.

Postcondiciónes: baja de un artefacto de un proyecto previamente definido.

Casos de usos relacionados: CU_34: Modificar Datos Artefacto.

Escenario principal:

1. El Usuario Avanzado indica al sistema que quiere dar de Baja un determinado artefacto de un proyecto previamente definido desde la pantalla "*Resumen de Artefacto*".
2. El sistema actualiza la información del artefacto indicado.
3. El sistema recupera la información de los artefactos de un proyecto definido.
4. El sistema muestra al Usuario Avanzado la pantalla "*Listado de Artefactos*" con la información actualizada de los artefactos definidos de un proyecto.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso

CU_36: Cambiar de Estado/ Asignar Artefacto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario autorizado o usuario normal quiera cambiar el estado y/o asignar el artefacto a otro usuario del proyecto en el que se encuentra definido el artefacto.

Actores: Usuario Autorizado, Usuario Normal.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_32: Listar Artefacto. Debe de existir al menos un artefacto definido en un proyecto.

Postcondiciónes: cambio de estado y/o asignación de un artefacto.

Casos de usos relacionados: CU_32: Listar Artefacto, CU_34: Modificar Datos de Artefacto, CU_37: Histórico de Artefacto, CU_38: Adjuntar Documento.

Escenario principal:

1. El Usuario Autorizado o Usuario Normal indican al sistema que quieren cambiar el estado y/o asignar a otro usuario un artefacto de un proyecto en la pantalla "*Listado de Artefactos*".
2. El sistema consulta la información actual del artefacto seleccionado.
3. El sistema recupera los usuarios del Proyecto en el que se encuentra definido el artefacto
4. El sistema recupera los estados permitidos para el artefacto según el rol que tiene el usuario y WorkFlow definido para el proyecto.
5. El sistema muestra al Usuario Avanzado o Usuario Normal la pantalla *"Resumen de Artefacto"*.
6. El Usuario Avanzado o Usuario Normal modifica el estado y/o asigna el artefacto a otro usuario del proyecto.
7. El sistema valida los datos introducidos.
8. El sistema actualiza la información del artefacto.
9. El sistema muestra la pantalla "*Resumen de Artefacto Correcto*".

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Autorizado o Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Usuario Autorizado o Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso.

4.1. Error no esperado de la aplicación.

4.1.1. El sistema muestra al Usuario Autorizado o Usuario Avanzado la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

4.1.2. Finaliza la ejecución de este caso.

4.2. El Usuario Autorizado o Usuario Normal no modifica la información del artefacto.

4.2.1. Finaliza la ejecución de este caso.

4.3. Los datos están incompletos

4.3.1. El sistema muestra al Usuario Autorizado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con los campos que son obligatorios.

4.3.2. La ejecución del caso continúa en el punto 6.

4.4. Los datos no son válidos.

4.4.1. El sistema muestra al Usuario Avanzado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

4.4.2. La ejecución del caso continúa en el punto 6.

4.5. Error no esperado de la aplicación.

4.5.1. El sistema muestra al Usuario Avanzado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

4.5.2. Finaliza la ejecución de este caso.

8.1. Error no esperado de la aplicación.

8.1.1. El sistema muestra al Usuario Avanzado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

8.1.2. Finaliza la ejecución de este caso.

CU_37: Histórico de Artefacto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario autorizado o usuario normal quiera consultar los cambios de estados y/o asignación que ha tenido un artefacto definido en un proyecto de la aplicación.

Actores: Usuario Autorizado, Usuario Normal.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación y CU_36: Cambiar de Estado/Asignar Artefacto.

Postcondiciónes: se obtiene lista con los cambios de estados y/o asignación que ha tenido un artefacto definido en un proyecto de la aplicación.

Casos de usos relacionados: CU_36: Cambiar de Estado/Asignar Artefacto.

Escenario principal:

1. El Usuario Autorizado o Usuario Normal indican al sistema que quieren consultar los estados y/o asignaciones por las que ha pasado un determinado artefacto en un proyecto de la aplicación desde la pantalla *"Resumen de Artefacto"*.
2. El sistema recupera los datos históricos del artefacto.
3. El sistema muestra la pantalla *"Resumen de Artefacto"* con los datos del histórico del artefacto.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Autorizado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Al menos existirá en los datos del histórico del artefacto un movimiento con el estado y asignación actual.***

CU_38: Adjuntar Documento

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario autorizado o usuario normal quieran adjuntar documentación a un artefacto definido en un proyecto de la aplicación.

Actores: Usuario Autorizado, Usuario Normal.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación y CU_36: Cambiar de Estado/Asignar Artefacto.

Postcondiciónes: se adjunta documentación al artefacto.

Casos de usos relacionados: CU_36: Cambiar de Estado/Asignar Artefacto.

Escenario principal:

1. El Usuario Autorizado o Usuario Normal indican al sistema que quieren adjuntar un documento a un determinado artefacto en un proyecto de la aplicación desde la pantalla *"Resumen de Artefacto"*.
2. El sistema muestra la pantalla "Selección Documento".
3. El Administrador informa el nombre del documento que quiere adjuntar.
4. El sistema valida la información
5. El sistema actualiza la información
6. El sistema muestra la pantalla *"Resumen de Artefacto"* con el documento adjunto al artefacto.

Flujos alternativos:

4.1. El Usuario Autorizado o Usuario Normal no informa el nombre del documento.

4.1.1. Finaliza la ejecución de este caso.

4.2. Los datos están incompletos

4.2.1. El sistema muestra al Usuario Autorizado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con los campos que son obligatorios.

4.2.2. La ejecución del caso continúa en el punto 3.

4.3. Los datos no son válidos.

4.3.1. El sistema muestra al Usuario Avanzado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

4.3.2. La ejecución del caso continúa en el punto 3.

4.4. Error no esperado de la aplicación.

4.4.1. El sistema muestra al Usuario Avanzado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

4.4.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Usuario Avanzado o Usuario Normal la pantalla "*Resumen de Artefacto*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Sólo permitiremos adjuntar un documento por cada cambio de estado y/o asignación de un artefacto.***

CU_40: Gestión de Comunicaciones

En el siguiente diagrama presentamos los casos de uso que están relacionados con la gestión de comunicaciones en la aplicación.

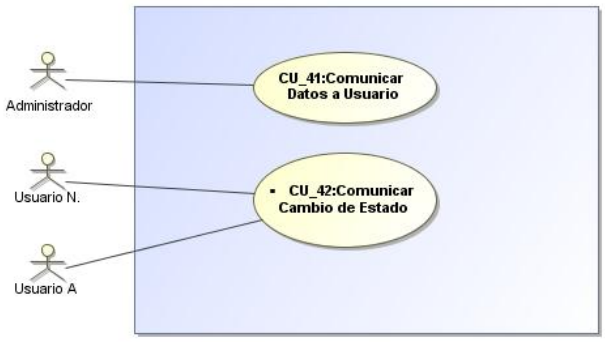

Figura 7: Gestión de Comunicaciones

CU_41: Comunicar Datos a Usuario

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera comunicarse con un usuario.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación. Es necesario que venga informado el email del usuario.

Postcondiciónes: envío de comunicado a usuario.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El Administrador indica al sistema que quiere comunicarse con un usuario.
2. El sistema recupera los datos del usuario.
3. El sistema manda un mensaje al usuario con sus datos.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Para poder enviar mensajes a los usuarios es necesario configurar los parámetros del correo.***

CU_42: Comunicar Cambio de Estado

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario avanzado o usuario normal quiera comunicar a otro usuario el cambio de estado y/o asignación de un artefacto de un proyecto a un usuario.

Actores: Usuario Avanzado, Usuario Normal.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación. Es necesario que venga informado el email del usuario.

Postcondiciónes: envío de comunicado a usuario.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El Usuario Avanzado o Usuario Normal indica al sistema que quiere comunicarse con un usuario.
2. El sistema recupera los datos del artefacto
3. El sistema recupera los datos del usuario.
4. El sistema manda un mensaje a un usuario con el estado y/o asignación de un artefacto.

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Avanzado o Usuario Normal un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Usuario Avanzado o Usuario Normal un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso.

Observaciones:

- ***Para poder enviar mensajes a los usuarios es necesario configurar los parámetros del correo.***

CU_50: Gestión de Informes

En el siguiente diagrama presentamos los casos de uso que están relacionados con la gestión de informes en la aplicación.

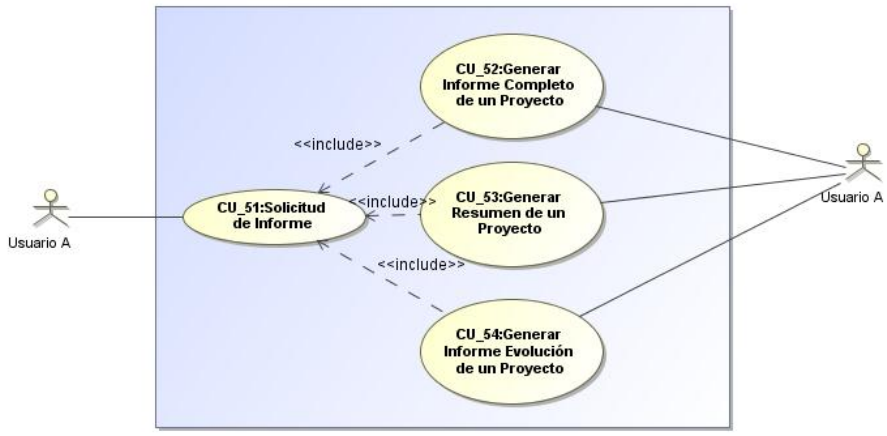

Figura 8: Gestión de Informes

CU_51: Solicitud de Informe

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario avanzado quiera generar un informe en la aplicación.

Actores: Usuario Avanzado.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación. **Postcondiciónes:** generación de informe sobre un proyecto.

Casos de usos relacionados: CU_52: Generar Informe Completo de un Proyecto, CU_53: Generar Resumen de un Proyecto, CU_54: Generar Informe de Evolución de un Proyecto.

Escenario principal:

1. El Usuario Avanzado indica al sistema que quiere generar un informe
2. El sistema muestra la pantalla "*Generación de Informe*".
3. El Usuario Avanzado informa el proyecto y el informe a generar
4. El sistema valida la información
5. El sistema genera el informe
6. El sistema muestra pantalla "*Informe Proyecto*" con los datos del informe.

Flujos alternativos:

4.1. Los datos están incompletos

4.1.1. El sistema muestra al Usuario Avanzado la pantalla "*Generación de Informe*" que contendrá un mensaje de error con los campos que son obligatorios.

4.1.2. La ejecución del caso continúa en el punto 3.

4.2. Error no esperado de la aplicación.

4.2.1. El sistema muestra al Usuario Avanzado la pantalla "*Generación de Informe*" que contendrá un mensaje de error con la descripción de dicho error.

4.2.2. Finaliza la ejecución de este caso.

5.1. Error no esperado de la aplicación.

5.1.1. El sistema muestra al Usuario Avanzado la pantalla "*Generación de Informe*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. Finaliza la ejecución de este caso.

CU_52: Generar Informe Completo de un Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario avanzado quiera generar un informe completo de un proyecto en la aplicación.

Actores: Usuario Avanzado.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_51: Solicitud de Informe.

Postcondiciónes: generación de informe completo sobre un proyecto.

Casos de usos relacionados: CU_51: Solicitud de Informe.

Escenario principal:

1. El Usuario Avanzado indica al sistema que quiere generar un informe completo sobre un proyecto
2. El sistema recupera los datos del proyecto
3. El sistema genera el informe

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Avanzado un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Usuario Avanzado un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso.

CU_53: Generar Informe Resumen de un Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario avanzado quiera generar un informe resumen de un proyecto en la aplicación.

Actores: Usuario Avanzado.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_51: Solicitud de Informe.

Postcondiciónes: generación de informe completo sobre un proyecto.

Casos de usos relacionados: CU_51: Solicitud de Informe.

Escenario principal:

1. El Usuario Avanzado indica al sistema que quiere generar un informe resumen sobre un proyecto
2. El sistema recupera los datos del proyecto
3. El sistema genera el informe

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Avanzado un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Usuario Avanzado un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso.

CU_54: Generar Informe de Evolución de un Proyecto

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un usuario avanzado quiera generar un informe de evolución de un proyecto en la aplicación.

Actores: Usuario Avanzado.

Precondiciones: Se han realizado correctamente los casos de uso CU_01:Identificarse a la Aplicación y CU_51: Solicitud de Informe.

Postcondiciónes: generación de informe completo sobre un proyecto.

Casos de usos relacionados: CU_51: Solicitud de Informe.

Escenario principal:

1. El Usuario Avanzado indica al sistema que quiere generar un informe de evolución de un proyecto

2. El sistema recupera los datos del proyecto
3. El sistema genera el informe

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Usuario Avanzado un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

3.1. Error no esperado de la aplicación.

3.1.1. El sistema muestra al Usuario Avanzado un mensaje de error con la descripción de dicho error.

3.1.2. Finaliza la ejecución de este caso.

CU_60: Configuración Aplicación

En el siguiente diagrama presentamos los casos de uso que están relacionados con la configuración de la aplicación.

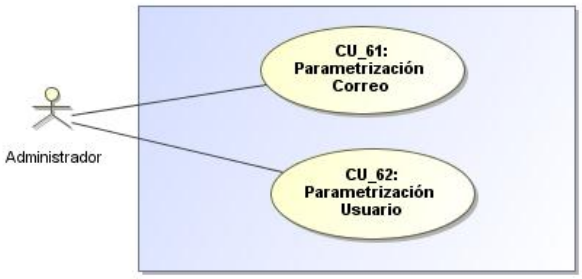

Figura 9: Configuración Aplicación

CU_61: Parametrización Correo

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera modificar los parámetros del correo en la aplicación.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: modificación de parámetros del correo.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El Administrador indica al sistema que quiere configurar algunos parámetros del correo desde la pantalla "*Parámetros*".
2. El sistema recupera los parámetros del correo.
3. El sistema muestra la pantalla "*Parámetros Correo*" con los parámetros actuales.
4. El Administrador informa los parámetros del correo.
5. El sistema valida los parámetros del correo.
6. El sistema registra los parámetros del correo.
7. El sistema muestra la pantalla "*Parámetros Correo Correcto*".

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador la pantalla "*Parámetros*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

5.1. Datos no válidos.

5.1.1. El sistema muestra al Administrador la pantalla "*Parámetros Correo*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. La ejecución del caso continúa en el punto 4.

5.2. Datos incompletos.

5.2.1. El sistema muestra al Administrador la pantalla "*Parámetros Correo*" que contendrá un mensaje de error con los campos obligatorios.

5.2.2. La ejecución del caso continúa en el punto 4.

5.3. Error no esperado de la aplicación.

5.3.1. El sistema muestra al Administrador la pantalla "*Parámetros Correo*" que contendrá un mensaje de error con la descripción de dicho error.

5.3.2. Finaliza la ejecución de este caso.

6.1. Error no esperado de la aplicación.

6.1.1. El sistema muestra al Administrador la pantalla "*Parámetros Correo*" que contendrá un mensaje de error con la descripción de dicho error.

6.1.2. Finaliza la ejecución de este caso.

CU_62: Parametrización Usuario

El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cada vez que un administrador quiera modificar los parámetros que afectan a los usuarios de la aplicación.

Actores: Administrador.

Precondiciones: Se ha realizado correctamente CU_01:Identificarse a la Aplicación.

Postcondiciónes: modificación de parámetros de usuario.

Casos de usos relacionados: ninguno.

Escenario principal:

1. El Administrador indica al sistema que quiere configurar algunos parámetros sobre los usuarios desde la pantalla "*Parámetros*".
2. El sistema recupera los parámetros de los usuarios.
3. El sistema muestra la pantalla "*Parámetros Usuarios*" con los parámetros actuales.
4. El Administrador informa los parámetros de los usuarios.
5. El sistema valida los parámetros de los usuarios.
6. El sistema registra los parámetros de los usuarios.
7. El sistema muestra la pantalla "*Parámetros Usuarios Correcto*".

Flujos alternativos:

2.1. Error no esperado de la aplicación.

2.1.1. El sistema muestra al Administrador la pantalla "*Parámetros*" que contendrá un mensaje de error con la descripción de dicho error.

2.1.2. Finaliza la ejecución de este caso.

5.1. Datos no válidos.

5.1.1. El sistema muestra al Administrador la pantalla "*Parámetros Usuarios*" que contendrá un mensaje de error con la descripción de dicho error.

5.1.2. La ejecución del caso continúa en el punto 4.

5.2. Datos incompletos.

5.2.1. El sistema muestra al Administrador la pantalla "*Parámetros Usuarios*" que contendrá un mensaje de error con los campos obligatorios.

5.2.2. La ejecución del caso continúa en el punto 4.

5.3. Error no esperado de la aplicación.

5.3.1. El sistema muestra al Administrador la pantalla "*Parámetros Usuarios*" que contendrá un mensaje de error con la descripción de dicho error.

5.3.2. Finaliza la ejecución de este caso.

6.1. Error no esperado de la aplicación.

6.1.1. El sistema muestra al Administrador la pantalla "*Parámetros Usuarios*" que contendrá un mensaje de error con la descripción de dicho error.

6.1.2. Finaliza la ejecución de este caso.

Análisis

A continuación presentaremos mediante un diagrama de clases UML[1] la descripción del modelo conceptual de datos de nuestra aplicación.

Modelo Conceptual de Datos

En este apartado vamos a identificar las diferentes clases que intervienen en nuestra aplicación y sus relaciones mediante diferentes diagramas de clases UML, así como las restricciones de integridad que hayamos identificado.

Identificación de las Clases

A continuación presentaremos por orden alfabético las clases que hemos identificado en nuestra aplicación.

ADMINISTRADOR

Cada objeto de esta clase representará a un usuario con el tipo de autorización de administrador.

ARTEFACTO

Cada objeto de esta clase representará a un tipo de elemento que es susceptible de desarrollarse dentro de un proyecto.

Los atributos de esta clase son:

- **Nombre:** nombre del artefacto
- **Descripción:** descripción breve del artefacto
- **Detalle:** descripción detallada del artefacto
- **Prioridad:** importancia que tiene el artefacto respecto a otros artefactos.
- **Fecha inicio:** fecha inicial de desarrollo del artefacto
- **Fecha final:** fecha final de desarrollo del artefacto
- **Estado actual:** el estado actual en que se encuentra el artefacto
- **Esfuerzo actual:** representa mediante un porcentaje el avance estimado de desarrollo del artefacto.
- **Usuario actual:** el usuario que tiene asignado actualmente el artefacto
- **Próxima secuencia:** representa un número entero secuencial. Dicho secuencial se irá incrementando cada vez que hagamos una modificación sobre el artefacto.
- **TimeStamp Alta:** fecha de alta del artefacto.
- **Usuario Alta:** usuario que realizó el alta del artefacto.
- **TimeStamp Modificación:** fecha de la última modificación del artefacto.
- **Usuario Modificación:** usuario que ha realizado la última modificación del artefacto.

ARTEFACTO-U

[1] UML son las siglas de **Unified Modeling Language**

Cada objeto de esta clase representará la asignación de un usuario a un artefacto, con un rol y esfuerzo determinado en una fecha. Esta clase nos permitirá controlar el balanceo de carga que tiene un usuario.

Los atributos de esta clase son:

- **Fecha inicio:** fecha inicial de asignación usuario a artefacto
- **Fecha final:** fecha final de asignación usuario a artefacto
- **Rol:** funcionalidad que el usuario desempeña en el desarrollo del artefacto en la fecha propuesta
- **Porcentaje:** representa el porcentaje de esfuerzo que destina el usuario de su jornada laboral a dicho artefacto.

CATEGORIA

Cada objeto de esta clase representará un tipo de categoría. En nuestra aplicación definiremos tres tipos de categorías: Usuario, Correo, Proyecto.

CATEGORIACONFIG

Cada objeto de esta clase representará un tipo de categoría de configuración.

Los atributos de esta clase son:

- **Nombre:** contiene la descripción de un tipo de categoría.
- **Configs:** se trata de una lista de parámetros configurable.

CONFIG

Cada objeto de esta clase representará un parámetro de la aplicación configurable por el administrador de la aplicación.

Los atributos de esta clase son:

- **Nombre:** nombre del parámetro.
- **Valor:** valor asignado al parámetro.

DOCUMENTO

Cada objeto de esta clase representará un archivo adjuntado a un artefacto en un estado determinado.

Los atributos de esta clase son:

- **Orden:** secuencial que nos permite saber en qué orden se asignó un documento a un archivo.
- **Descripción:** descripción breve del contenido del archivo.
- **File Name Externo:** nombre del fichero externo asociado.
- **File Name Interno:** nombre generado de forma automática por el sistema. El nombre se compondrá con los siguientes atributos: código de artefacto + código de estado + fecha + hora + orden + extensión del archivo externo.

ESTADO

Cada objeto de esta clase representará un estado que podrá ser asignado a un artefacto en un proyecto determinado. Dentro de cada proyecto existirán diferentes estados.

Los atributos de esta clase son:

- **Nombre:** nombre del estado.

HISTORIA

Cada objeto de esta clase representará el estado en que se encontraba un artefacto antes de modificarlo.

Los atributos de esta clase son:

- **Nombre:** nombre del artefacto
- **Descripción:** descripción breve del artefacto
- **Detalle:** descripción detallada del artefacto
- **Prioridad:** importancia que tiene el artefacto respecto a otros artefactos.
- **Fecha inicio:** fecha inicial de desarrollo del artefacto
- **Fecha final:** fecha final de desarrollo del artefacto
- **Estado actual:** el estado actual en que se encuentra el artefacto
- **Esfuerzo actual:** representa mediante un porcentaje el avance estimado de desarrollo del artefacto.
- **Usuario actual:** el usuario que tiene asignado actualmente el artefacto
- **Próxima secuencia:** representa un número entero secuencial. Dicho secuencial se irá incrementando cada vez que hagamos una modificación sobre el artefacto.
- **TimeStamp Alta:** fecha de alta del artefacto.
- **Usuario Alta:** usuario que realizó el alta del artefacto.
- **TimeStamp Modificación:** fecha de la última modificación del artefacto.
- **Usuario Modificación:** usuario que ha realizado la última modificación del artefacto.

INFORME

Cada objeto de esta clase representará uno de los tipos de informes que puede generar un usuario autorizado.

Los atributos de esta clase son:

- **Nombre:** nombre del informe
- **Descripción:** descripción breve del informe

INFORME COMPLETO

Cada objeto de esta clase representará un tipo de informe completo que puede generar un usuario autorizado.

INFORME EVOLUCION

Cada objeto de esta clase representará un tipo de informe evolución que puede generar un usuario autorizado.

INFORME RESUMEN

Cada objeto de esta clase representará un tipo de informe resumen que puede generar un usuario autorizado.

PROYECTO

Cada objeto de esta clase representará la definición de un proyecto por parte de un administrador.

Los atributos de esta clase son:

- **Código:** código del Proyecto
- **Nombre:** nombre del Proyecto
- **Descripción:** descripción breve del Proyecto
- **Fecha Inicio:** fecha estimada de inicio del Proyecto
- **Fecha Fin:** fecha estimada de finalización del Proyecto
- **Roles:** lista de roles dentro de un proyecto
- **Estados:** lista de estados dentro de un proyecto
- **Usuarios:** lista de usuarios y sus roles dentro de un proyecto
- **WorkFlow:** lista de flujo de estados para un artefacto

ROL

Cada objeto de esta clase representará una funcionalidad que un usuario asignado a un proyecto puede desempeñar.

Los atributos de esta clase son:

- **Nombre:** nombre de la funcionalidad
- **Descripción:** descripción breve de la funcionalidad

USUARIO

Cada objeto de esta clase representará a un tipo de usuario de la aplicación.

Los atributos de esta clase son:

- **Login:** nombre de usuario para el acceso a la aplicación.
- **Password:** contraseña del usuario para el acceso a la aplicación.
- **Nombre:** nombre completo del usuario.
- **Empresa:** empresa a la que pertenece el usuario.
- **Email:** correo electrónico del usuario.
- **Intentos:** número de intentos fallidos de acceso a la aplicación.
- **Locked:** indica si el usuario está bloqueado o activo para la aplicación.

USUARIO AUTORIZADO

Cada objeto de esta clase representará a un tipo de usuario autorizado de la aplicación.

USUARIO NORMAL

Cada objeto de esta clase representará a un tipo de usuario normal de la aplicación.

USUARIO-ROL

Cada objeto de esta clase representará a una asociación de un usuario a un rol en un proyecto determinado.

Los atributos de esta clase son:

- **Login:** nombre de usuario para el acceso a la aplicación.
- **Rol:** nombre de la funcionalidad

WORKFLOW

Cada objeto de esta clase representará un estado al que puede pasar un artefacto. El cambio a ese estado dependerá del estado actual del artefacto y del rol del usuario que tenga asignado.

Los atributos de esta clase son:

- **Estado:** nombre del estado actual de un artefacto.
- **Rol:** nombre de una funcionalidad de un usuario
- **Próximos estados:** nombre de los estados a los que puede cambiar un artefacto desde el estado y rol actual.

Identificación de las relaciones entre Clases

A continuación mostraremos las diferentes relaciones entre clases en varios sub-apartados.

TIPOS DE USUARIOS

En el sistema nos encontraremos con tres tipos de usuarios, tal como se recoge en el documento de requisitos. Estos tipos de usuarios son: Administrador, Usuario Avanzado y Usuario Normal. Cada uno de ellos corresponde a una de las clases definidas anteriormente (*Administrador*, *Usuario Avanzado* y *Usuario Normal*) y dado que comparten los mismos atributos que la clase *Usuario*, definiremos una herencia con esta clase. Por otro lado, dado que ningún usuario puede tener los tres tipos de autorizaciones a la vez, se trataría de una relación disjunta.

A continuación mostramos la parte del diagrama de clases que modela esta relación:

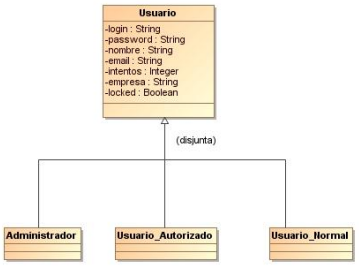

Figura 10: Diagrama de Clases Tipos de Usuarios

CONFIGURACION

Para facilitar al usuario administrador el mantenimiento de los parámetros, es necesario que dichos parámetros estén organizados en categorías (*clase CategoríaConfig*). A una categoría pueden pertenecer de cero a muchos parámetros (*clase Config*). Esta relación entre una categoría y un parámetro se representa en nuestro diagrama como una relación de uno-a-muchos entre las clases *CategoríaConfig* y *Config* de tipo composición.

Por otro lado, en nuestra aplicación encontramos tres tipos de categorías las cuales se representan mediante la clase enumerada *Categoria*. Esta nueva clase hace que tengamos que definir una relación de tipo asociación entre las clases *CategoriaConfig* y *Categoria*.

Además un usuario administrador podrá modificar uno o más parámetros dentro de una categoría y a su vez un parámetro sólo podrá ser modificado por un usuario administrador. Se trata de una relación de uno-a-muchos de tipo asociación entre las clases *CategoriaConfig* y *Administrador*.

A continuación mostramos la parte del diagrama de clases que modela esta relación:

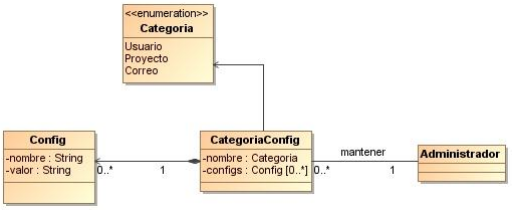

Figura 11: Diagrama de Clases Configuración

DEFINICION USUARIO

En el sistema que vamos a desarrollar se encargará del mantenimiento de los usuarios el administrador del sistema. Esto hace que tengamos que definir una relación de asociación entre las clases *Administrador* y *Usuario*, y con una multiplicidad de uno-a-muchos. En el lado de la clase Administrador tendremos una multiplicidad de uno para indicar que un usuario es siempre mantenido por un administrador y en el lado de la clase Usuario tendremos una multiplicidad de cero a muchos para indicar que un administrador puede que mantenga cero o muchos usuarios.

A continuación mostramos la parte del diagrama de clases que modela esta relación:

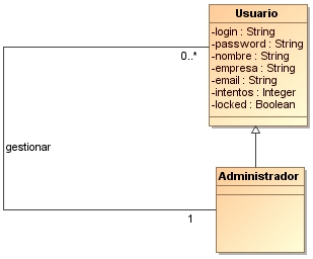

Figura 12: Diagrama de Clases Definición Usuario

DEFINICION PROYECTO

La aplicación que vamos a desarrollar permitirá a un usuario administrador definir uno o más proyectos y un proyecto ser definido por un único usuario administrador. Esta relación entre un usuario Administrador y un Proyecto se representa en nuestro diagrama como una relación de uno-a-muchos de tipo asociación entre las clases *Administrador* y *Proyecto*.

Durante la definición de dicho proyecto el administrador tiene que decidir sobre diversos aspectos tales como:

- Que roles se van a definir para el proyecto. Esto se representa como una relación de uno-a-muchos de tipo asociación entre las clases *Administrador* y *Rol*. Esta relación muestra el hecho de que un usuario administrador define cero o más roles para un proyecto y que un rol dentro de un proyecto es definido por un usuario administrador.

- Que estados se van a definir para el proyecto. Esto se representa como una relación de uno-a-muchos de tipo asociación entre las clases *Administrador* y *Estado*. Esta relación muestra el hecho de que un usuario administrador define cero o más estados para un proyecto y que un estado dentro de un proyecto es definido por un usuario administrador.

- Que flujo de estados se va a definir para el proyecto. Esto se representa como una relación de uno-a-muchos de tipo asociación entre las clases *Administrador* y *WorkFlow*. Esta relación muestra el hecho de que un usuario administrador define cero o más flujos de estados para un proyecto y que un flujo de estado dentro de un proyecto es definido por un usuario administrador.

- Que usuarios con un determinado rol se va a definir para el proyecto. Esto se representa como una relación de uno-a-muchos de tipo asociación entre las clases *Administrador* y *Usuario-Rol*. Esta relación muestra el hecho de que un usuario administrador define cero o más usuarios con roles para un proyecto y que un determinado usuario con un rol dentro de un proyecto es definido por un usuario administrador.

Cuando se asigna un rol a un usuario dentro de un proyecto se ha de crear una instancia del objeto *Usuario-Rol* la cual nos servirá por un lado para recoger las características de esta asociación en los atributos de dicho objeto (*login del usuario y descripción del rol*) y por otro para asignar dicha instancia a la clase *Proyecto*. Esto se traduce en nuestro diagrama en la definición de dos relaciones:

- La primera entre las clases *Usuario* y *Rol* de tipo asociación donde cada vez que se cree una instancia de dicha relación se creará un objeto de la clase *Usuario-Rol.*

- La segunda se define como una relación de uno-a-muchos de tipo composición entre el objeto de la clase *Usuario-Rol* y la clase *Proyecto*. Con esta relación indicamos que en un Proyecto se definen muchos usuarios con roles determinados y que un usuario con un rol determinado es definido dentro de un proyecto.

Cuando se asigna un estado a un rol dentro de un proyecto se ha de crear una instancia del objeto *WorkFlow* la cual nos servirá por un lado para recoger las características de esta asociación en los atributos de dicho objeto (*nombre del estado, nombre del rol y próximo estado*) y por otro para asignar dicha instancia a la clase *Proyecto*. Esto se traduce en nuestro diagrama en la definición de tres relaciones:

- La primera entre las clases *Estado* y *Rol* de tipo asociación donde cada vez que se cree una instancia de dicha relación se creará un objeto de la clase *Estado-Rol.*

- La segunda se define como una relación de uno-a-muchos de tipo composición entre el objeto de la clase *Estado-Rol* y la clase *Estado*. Con esta relación indicamos que desde un estado y rol determinado se puede pasar a muchos estados y que a un estado se llega desde un estado y rol determinado.

- La tercera se define como una relación de uno-a-muchos de tipo composición entre el objeto de la clase *Estado-Rol* y la clase *Proyecto*. Con esta relación indicamos que en un Proyecto se definen muchos flujos de estados y que un flujo de estado determinado es definido dentro de un proyecto.

Cuando un administrador define un rol o un estado siempre ha de realizarlo dentro del contexto de un proyecto en concreto. Esto se traduce en que deberá existir una relación de uno-a-muchos de tipo composición entre las clases *Rol* y *Proyecto*, y otra relación igual a la anterior entre las clases *Estado* y *Proyecto*.

A continuación mostramos la parte del diagrama de clases que modela esta relación:

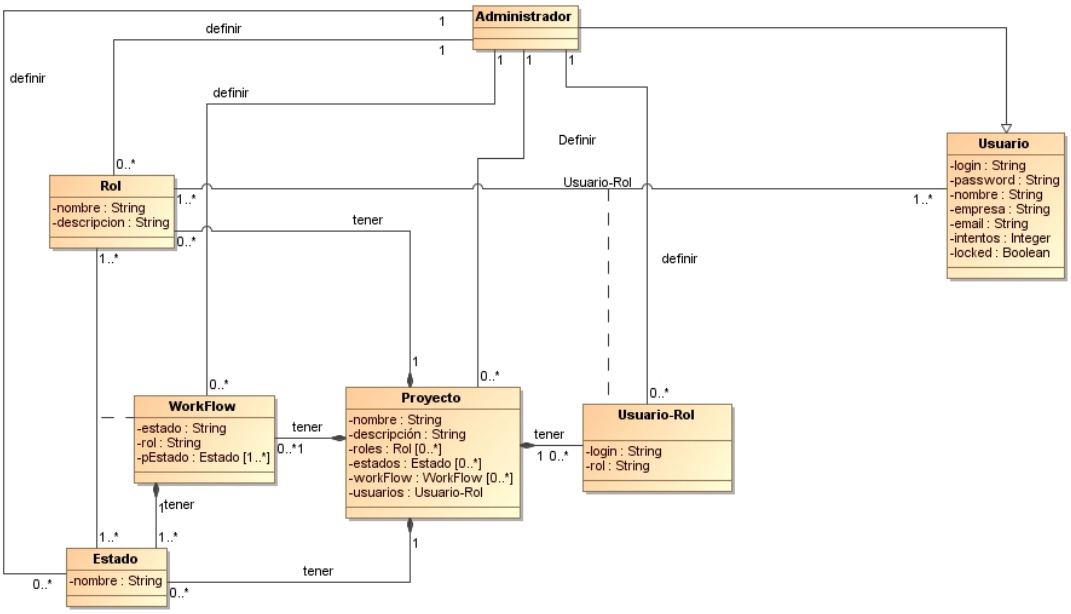

Figura 13: Diagrama de Clases Definición Proyecto

DEFINICIÓN ARTEFACTO

En nuestra aplicación se definirán y mantendrán artefactos, los cuáles serán gestionados por los usuarios autorizados. Esta labor que realizará un usuario autorizado se representa en nuestro diagrama como una relación de uno-a-muchos de tipo asociación entre las clases *Artefacto* y *Usuario Autorizado*. Del lado de la clase *Artefacto* tendremos una multiplicidad de cero a muchos para indicar que un usuario autorizado gestiona uno o más artefactos. Del lado de la clase *Usuario Autorizado* definimos una multiplicidad de uno para indicar que un artefacto siempre es gestionado por un usuario autorizado.

Los artefactos serán definidos dentro del contexto de un Proyecto. Este hecho hace que definamos una relación de uno-a-muchos de tipo composición entre las clases *Proyecto* y *Artefacto.* Esta relación se define de tipo composición dado que para que exista una instancia de la clase *Artefacto* es necesario que exista una instancia de la clase *Proyecto*.

Un artefacto en un momento determinado puede tener un estado distinto al que se creó o respecto a un instante anterior. Esto provoca que cuando asignemos un nuevo estado a un artefacto dentro de un proyecto sea necesario crear una instancia del objeto *Historia* la cual nos servirá para recoger los atributos que tenía o tiene un objeto *Artefacto*. Lo anteriormente expuesto se traduce en que existirá una relación entre las clases *Estado*, *Artefacto* e *Historia* de tipo asociación.

Dado que durante la asignación de un estado podemos adjuntar documentación, se hace necesario definir una relación de uno-a-muchos entre las clases *Documento* e *Historia* de tipo composición. La multiplicidad del lado de la clase *Documento* será de cero a muchos para representar que un artefacto en un estado puede tener o no un documento asignado. La multiplicidad del lado de la clase *Historia* será de uno para indicar que un documento siempre está asociado a un estado determinado de un artefacto.

Para poder realizar un balanceo de carga adecuado en la asignación de los artefactos a los usuarios se hace necesario controlar en todo momento la carga que tendrá un usuario en un momento determinado, así como que artefactos tendrá asignados. Este hecho nos indica que debemos definir varias relaciones:

- Una relación de tipo asociación entre las clases *Artefacto*, *Usuario-Rol* y *Artefacto-U*. Por cada instancia de la relación entre las clases *Artefacto* y *Usuario-Rol*, debemos de crear una instancia de la clase *Artefacto-U*.

- Una relación de uno-a-muchos de tipo asociación entre la clase *Historia* y *Artefacto-U*. Con una multiplicidad de cero a muchos en el lado de la clase *Historia* para representar que un usuario con un rol determinado tiene asignado de cero a muchos artefactos y con una multiplicidad de uno en el lado de la clase *Artefacto-U* para representar que un artefacto siempre está asignado a un usuario.

- Una relación de uno-a-muchos de tipo asociación entre la clase *Historia* y *Artefacto-U*. Con una multiplicidad de cero a muchos en el lado de la clase *Historia* para representar que un usuario con un rol determinado ha modificado de cero a muchos artefactos y con una multiplicidad de uno en el lado de la clase *Artefacto-U* para representar que un artefacto siempre es cambiado por un usuario.

A continuación mostramos la parte del diagrama de clases que modela esta relación:

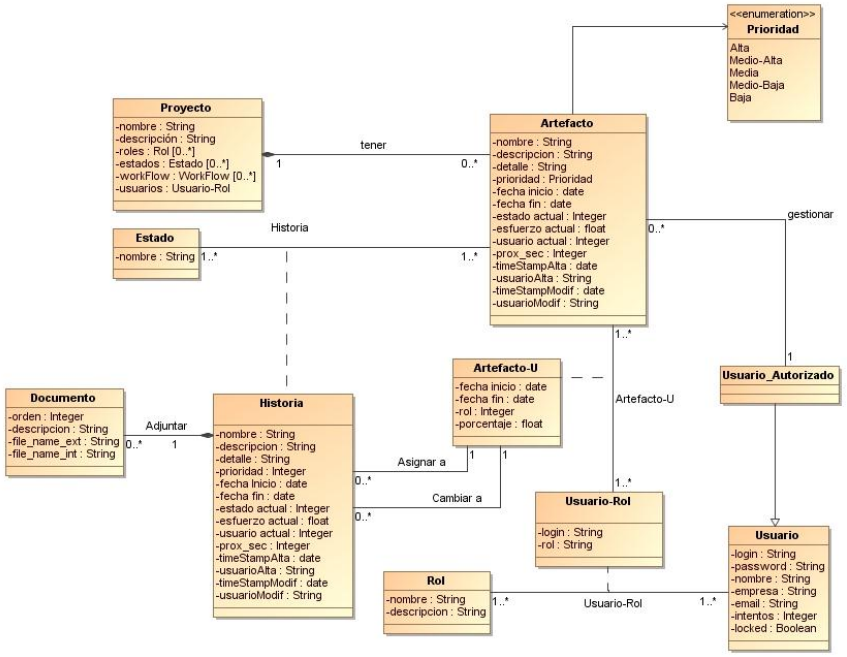

Figura 14: Diagrama de Clases Definición Artefacto

TIPOS DE INFORMES

El sistema permitirá a un usuario autorizado generar diversos informes predefinidos. Este hecho define una relación de uno a muchos entre las clases *Usuario Autorizado* e *Informes* de tipo asociación. La multiplicidad de uno a muchos se encuentra en el lado de la clase *Usuario Autorizado* para indicar que un informe es generado por más de un usuario autorizado. La multiplicidad de cero a muchos se encuentra al lado de la clase *Informe* para indicar que un usuario autorizado genera cero o más informes.

En nuestro sistema distinguimos tres tipos de informes diferentes: Informe Completo, Informe Resumen e Informe Evolución. Cada uno de estos informes comparten unas series de características comunes con el resto de informes (nombre del informe, descripción del informe) y otras que son particulares de cada uno de ellos. Esto implica que definamos una relación de generalización entre las clases *Informe* y las tres que representa a cada uno de los informes citados.

Por otro lado esta relación de generalización la debemos de considerar disjunta ya que un informe sólo puede ser de un tipo.

A continuación mostramos la parte del diagrama de clases que modela esta relación:

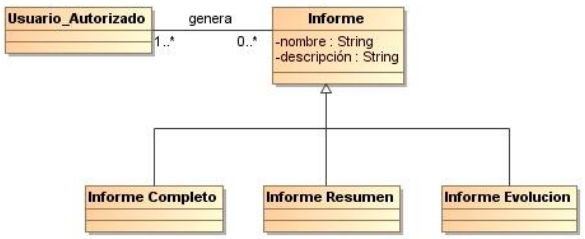

Figura 15: Diagrama de Clases Tipos de Informes

Restricciones de Integridad

A continuación relacionaremos todas las restricciones que hemos descubierto en los datos que utiliza la aplicación a desarrollar.

- Un nombre de usuario sólo puede estar asociado a un usuario, dado que es el identificador que usaremos para el acceso a la aplicación.
- Un email de usuario sólo puede estar asociado a un usuario dado que se usará para las comunicaciones con el usuario.
- Deberá existir una configuración inicial. Dicha configuración contendrá diversos parámetros los cuáles son necesarios para el correcto funcionamiento de la aplicación.
- Siempre deberá existir definido un único usuario administrador.
- Siempre existirá por defecto un rol de tipo "*default"* en todos los proyectos. Dicho rol es asignado a un usuario cuando se le asigna por primera vez a un proyecto con un rol.
- Siempre existirán por defecto dos estados para un artefacto *"abierto"* y *"cerrado"*. Estos estados no serán mantenidos. El estado abierto es asignado al artefacto cuando se crea dentro de un proyecto. El estado cerrado será siempre el último estado al que debe de pasar un artefacto cuando finalizamos.
- El nombre del proyecto y su código son únicos para cada proyecto que definamos.
- El nombre de un artefacto es único para cada artefacto que definamos dentro de un proyecto.
- El nombre de un estado es único dentro de un proyecto.
- El nombre de un rol es único dentro de un proyecto
- Las relaciones de usuarios y roles son únicas para cada proyecto
- Dentro de un flujo de datos no se permiten relaciones duplicadas para un proyecto.

- Sólo se permite modificar el estado de un artefacto al usuario que tenga asignado el artefacto o al usuario administrador.

Diagrama de Clases Final

A continuación mostraremos el diagrama de clases completo que hemos ido construyendo a lo largo de esta fase de análisis.

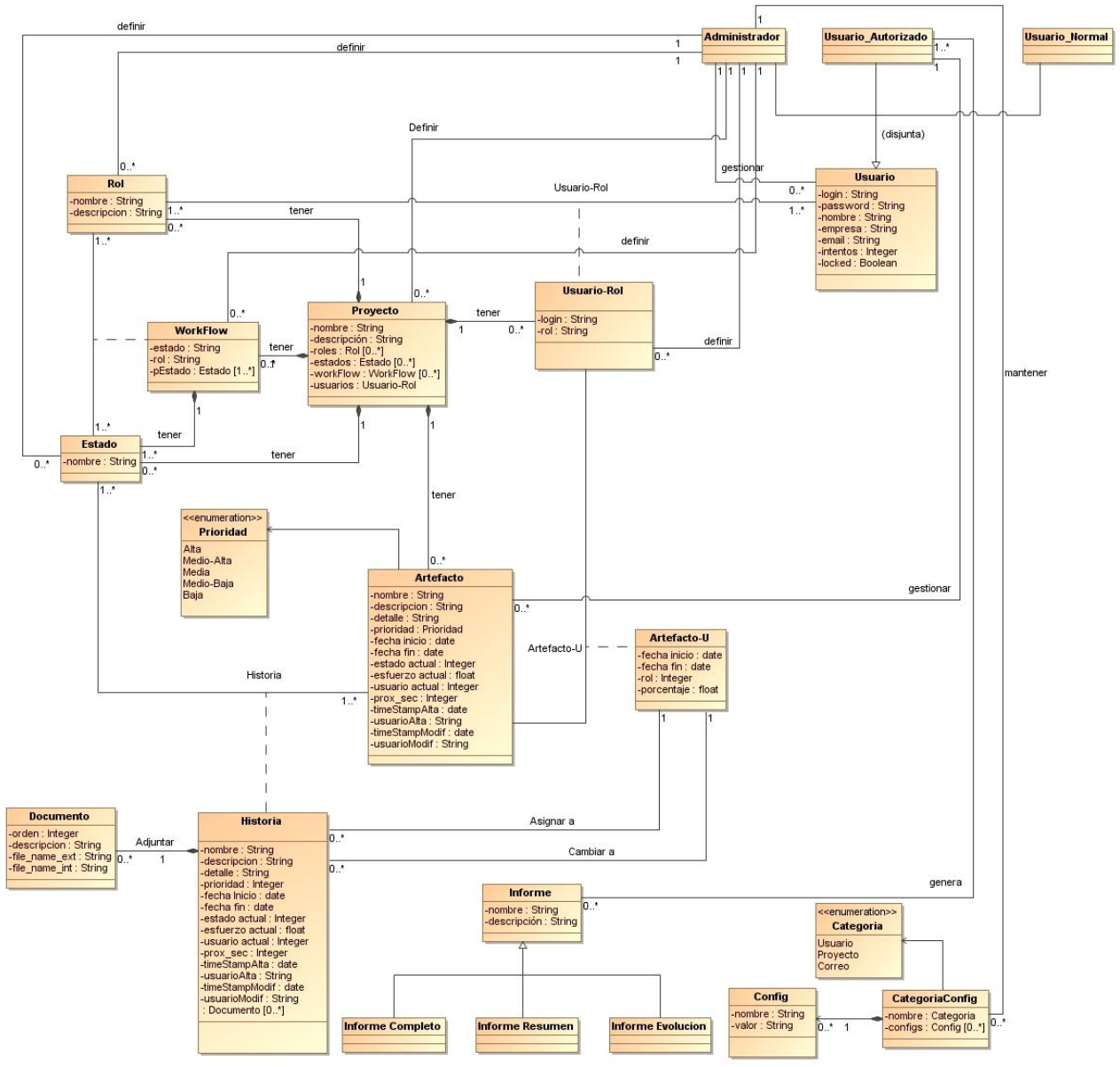

Figura 16: Diagrama de Clases Final

Diseño

Dado que lo que queremos es diseñar una aplicación web, adoptaremos un estilo arquitectónico heterogéneo, modelo cliente/servidor, y en tres capas.

- Una que se encargará de interactuar con los usuarios del sistema
- Otra encargada de implementar las funcionalidades de la aplicación
- Y otra que se encargará de interactuar con las fuentes de datos que almacenan la información persistente.

Para llevar a cabo esta separación vamos a descomponer cada uno de los componentes anteriores en otros componentes de grano más fino, cada uno residiendo en una capa distinta, con responsabilidades más definidas, y cada uno encargado de implementar una serie de interfaces.

Arquitectura de tres Capas

Capa de Presentación

Definiremos un único punto de acceso al sistema (*Patrón Front Controller*[2]) para que el manejo de las peticiones a la capa de presentación soporte la integración de los servicios del sistema, recuperación de contenidos, control de vistas, y navegación. Este punto de acceso lo representaremos en nuestro sistema como un componente que denominaremos *"PIntegra"*.

Dicho componente constará de los siguientes elementos:

- Un componente que se encargará de recibir, autorizar y procesar las peticiones que se hagan al sistema (*Front Controller*).

- Un componente que se encargará de redirigir las peticiones a un componente adecuado dentro de la capa de presentación (*Dispatcher*[3]).

- Una interfaz a la cual denominaremos *"IPIntegra"* que ofrecerá todas las funcionalidades que se pueden solicitar al sistema. Estas funcionalidades serán las necesarias para que se puedan realizar los casos de usos que hemos definido durante la etapa de análisis.

[2] **Referencia** : http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html
[3] **Dispatcher**: Es responsable del control de la vista y de la navegación. Puede encapsularse directamente dentro del controlador o se puede extraer en un componente separado.

La representación de dicho componente mediante un diagrama sería la siguiente:

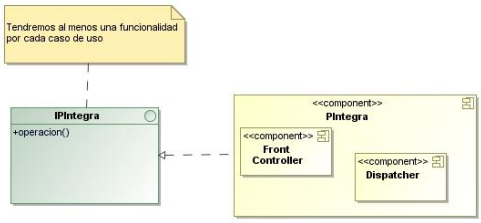

Figura 17: Diagrama del Componente PIntegra

Además, será necesario definir en la capa de presentación una serie de componentes especializados que agrupen y atiendan las funcionalidades descritas en la interfaz *IPIntegra*. Estos componentes recibirán peticiones del Dispatcher, una vez que el componente Front Controller las autorice. Para el caso que nos ocupa definiremos 6 componentes:

- PConfManager: gestionará todas las funcionalidades relacionadas con los parámetros de correo, usuarios y balanceo.
- PAuthManager: gestionará todas las funcionalidades relacionadas con la autorización de acceso al sistema.
- PUserManager: gestionará todas las funcionalidades relacionadas con la gestión de usuarios.
- PProjManager: gestionará todas las funcionalidades relacionadas con la gestión de proyectos y asignación de usuarios a un proyecto.
- PArteManager: gestionará todas las funcionalidades relacionadas con la gestión de los artefactos de un proyecto y asignación de usuarios a un artefacto.
- PInfoManager: gestionará todas las funcionalidades relacionadas con la generación de informes.

Cada uno de los componentes anteriores recogerá en una interfaz exterior todas las funcionalidades que deben de implementar. En las siguientes tablas podemos observar la relación de interfaces externas que usaremos en nuestro sistema por componente especializado y las funcionalidades que son necesarias implementar para cada caso de uso presentado durante la etapa de análisis, junto al componente que se encargará de su implementación.

Interfaz de Presentación	**Componente de Presentación**
IPConfManager	PConfManager
IPAuthManager	PAuthManager
IPUserManager	PUserManager
IPProjManager	PProjManager
IPArteManager	PArteManager
IPInfoManager	PInfoManager

Tabla 3: Relación de Interfaces-Componentes Especializados

Casos de Uso	Operación de la Capa de Presentación	Componente
CU_01:Identificarse en la Aplicación	signin(login: String, password: String): TOUser	PAuthManager
CU_02:Salir de la Aplicación	signout()	PAuthManager
CU_11:Alta de Usuario	createUser(login: String, password: String, name: String, company: String, email: String, type: Integer)	PUserManager
CU_12:Listar Usuario	listUser():List<TOUser>	PUserManager
CU_13:Criterio de Búsqueda	listUser(login: String, name: String, company: String, email: String, type: Integer):List<TOUser>	PUserManager
CU_14:Desbloquear Usuario	unlockUser(login: String):TOUser	PUserManager
CU_15:Modificar Datos de Usuario	modifyUser(login: String, password: String, name: String, company: String, email: String, type: Integer):TOUser	PUserManager
CU_16:Baja de Usuario	deleteUser(login: String)	PUserManager
CU_41:Comunicar Datos de Usuario	emailDatoUser(email: String, name: String, login: String, password: String)	PUserManager
CU_201:Alta de Proyecto	createProject(Key: String, name: String, descript: String)	PProjManager
CU_202:Crear a partir de	createFromProject(KeyFrom: String, KeyTo: String, nameTo: String, descriptTo: String)	PProjManager
CU_203:Listar Proyecto	listProject():List<TOProject>	PProjManager
CU_204:Criterios de Búsqueda	listProject(Key: String, name: String):List<TOProject>	PProjManager
CU_205:Asignación de Rol-Proyecto	createRolProject(project: TOProject, nameRol: String, descriptRol: String) deleteRolProject(project: TOProject, rol: TORol) modifyRolProject(project: TOProject, nameRol: String, descriptRol: String):TORol listRolProject(project: TOProject):List<TORol>	PProjManager
CU_206:Asignación de Estado-Proyecto	createStateProject(project: TOProject, nameState: String, descriptState: String) deleteStateProject(project: TOProject, state: TOState) modifyStateProject(project: TOProject, nameState: String, descriptState: String):TOState listStateProject(project: TOProject):List<TOState>	PProjManager
CU_207:Definición de WorkFlow	createWorkFlowProject(project: TOProject, stateFrom: TOState, rolFrom: TORol, statesTo: List<TOState>):TOWorkF deleteWorkFProject(project: TOProject, workf: TOWorkF) modifyWorkFProject(project: TOProject, workf: TOWorkF, statesTo: List<TOState>):TOWorkF listWorkFProject(project: TOProject):List<TOWorkF>	PProjManager
CU_208:Modificar datos del Proyecto	modifyProject(Key: String, name: String, descript: String):TOProject	PProjManager
CU_209:Listar Usuarios-Proyecto	listUsersProject(project: TOProject):List<TOUserRol>	PProjManager
CU_210:Asignación de Usuarios-Roles	createUserRProject(project: TOProject, user: TOUser, rol: TORol):TOUserRol deleteUserRolProject(project: TOProject, userRol: TOUserRol) modifyUserRolProject(project: TOProject, userRol: TOUserRol, roles: List<TORol):TOUserRol	PProjManager
CU_211:Baja de Proyecto	deleteProject(name: String)	PProjManager

CU_31:Alta de Artefacto	createArtefact(project: TOProject, name: String, descript: String, detail: String, dateI: date, dateF: date, stateA: TOEstado , effortA: float, userA: TOUserRolF, userC: List <TOUserRolF>):TOArtefact createUserArtefact(project: TOProject, artefact: TOArtefact, user: TOUserRol, dateI: date, dateF: date, porcentaje: int):TOUserRolF modifUserArtefact(project: TOProject, artefact: TOArtefact, userRolF: TOUserRolF):TOUserRolF deleteUserArtefact(project: TOProject, artefact: TOArtefact, userRolF: TOUserRolF) listUserArtefact(project: TOProject, artefact: TOArtefact):List<TOUserRolF>	PArteManager
CU_32:Listar Artefacto	listArtefactProject(project: TOProject):List<TOArtefact>	PArteManager
CU_33:Criterios de Búsqueda	listArtefactProject(project: TOProject, name: String, dateA: date, stateA: TOEstado, userA: TOUserRolF):List<TOArtefact>	PArteManager
CU_34:Modificar Datos Artefacto	modifyArtefact(project: TOProject, artefact: TOArtefact):TOArtefact	PArteManager
CU_35:Baja de Artefacto	deleteArtefact(project: TOProject, name: String)	PArteManager
CU_36:Cambiar de Estado/Asignar Artefacto	changeArtefact(project: TOProject, artefact: TOArtefact, state: TOEstado, user: TOUserRolF):TOArtefact createHistory(project: TOProject, artefact: TOArtefact):TOHistory	PArteManager
CU_37:Consulta Histórico de Artefacto	listHistoryArtefact(project: TOProject, artefact: TOArtefact):List<TOHistory>	PArteManager
CU_38:Adjuntar Documento	atachmentArtefact(project: TOProject, artefact: TOArtefact, descript: String, file_nameE: String, file_NameI: String)	PArteManager
CU_42:Comunicar Cambio de Estado	emailChangeArtefact(project:TOProject,artefact:TOArtefact,user:TOUserRolF)	PArteManager
CU_51:Solicitud de Informe	listInform():List<TOInform>	PInfoManager
CU_52:Informe Completo Proyecto	informComp(project: TOProject):TOInfC	PInfoManager
CU_53.Informe Resumen Proyecto	informResu(project: TOProject):TOInfR	PInfoManager
CU_54:Informe Evolución Proyecto	informEvol(project: TOProject):TOInfE	PInfoManager
CU_61:Parámetrización de Correo	modifConfigEmail(param: String, value: String): TOParam listConfigEmail():List<TOParam>	PConfManager
CU_62:Parámetrización de Usuario	modifConfigUser (param: String, value: String): TOParam listConfigUser():List<TOParam>	PConfManager
CU_62:Parámetrización de Balanceo	modifConfigBal(param: String, value: String): TOParam listConfigBal():List<TOParam>	PConfManager

Tabla 4: Relación Casos de Uso-Funciones-Componentes

Vamos a explicar cada una de las funcionalidades que hemos presentado en la Figura anterior, agrupadas por componentes.

COMPONENTE PAUTHMANAGER

- signin(login: String, password: String): TOUser
  Esta función es llamada cuando un usuario quiere acceder al sistema. Su función consiste en verificar que un usuario está registrado en el sistema. Si el usuario está registrado se retorna un objeto que contendrá la información del usuario.

- signout()
  Esta función es llamada cuando un usuario quiere abandonar el sistema. Para su invocación es necesario que el usuario haya realizado previamente un acceso al sistema.

COMPONENTE PCONFMANAGER

Para invocar cualquier función de este componente es necesario previamente verificar que el usuario sea un administrador.

- modifConfigEmail(param: String, value :String): TOParam
  Dicha función modifica la información relativa a un parámetro de correo. Una vez realizada la modificación devolverá un objeto *TOParam* actualizado.

- listConfigEmail():List<TOParam>
  Dicha función es llamada cuando se quiere consultar los valores que tenemos definidos para el correo. Una vez realizada devolverá una lista de objetos *TOParam* que contendrá por cada parámetro del correo su valor.

- modifConfigUser(param: String, value :String): TOParam
  Dicha función modifica la información relativa a un parámetro de usuario. Una vez realizada la modificación devolverá un objeto *TOParam* actualizado.

- listConfigUser():List<TOParam>
  Dicha función es llamada cuando se quiere consultar los valores que tenemos definidos para los usuarios. Una vez realizada devolverá una lista de objetos *TOParam* que contendrá por cada parámetro de usuario su valor.

- modifConfigBal(param: String, value :String): TOParam
  Dicha función modifica la información relativa a un parámetro de balance de carga. Una vez realizada la modificación devolverá un objeto *TOParam* actualizado.

- listConfigBal():List<TOParam>
  Dicha función es llamada cuando se quiere consultar los valores que tenemos definidos para el balanceo de carga. Una vez realizada devolverá una lista de objetos *TOParam* que contendrá por cada parámetro del balanceo de carga su valor.

COMPONENTE PUSERMANAGER

Para invocar cualquier función de este componente es necesario previamente verificar que el usuario sea un administrador.

- createUser(login: String, password: String, name: String, company: String, email: String, type: Integer)
  Esta función dará de alta a un usuario al sistema a partir de la información que se la pasa como parámetro. Dicha función comprobará previamente que ni el email, ni el login ya existen dados de altas en el sistema, y que venga el tipo de autorización informado (*usuario Normal*, *usuario Avanzado*).

- listUser(): List<TOUser>
  Esta función es llamada cuando se quiere consultar todos los usuarios que tenemos definidos en el sistema. Una vez realizada devolverá una lista de objetos *TOUser* que contendrá los datos de cada usuario.

- listUser(login: String, name: String, company: String, email: String, type: Integer): List<TOUser>
  Esta función realizará una búsqueda de los usuarios que cumplen ciertas características (*que el nombre contenga…, que pertenezca a la compañía…, etc.*). A partir de la información que se le pasa como parámetro a la función se montarán los criterios de búsqueda. Una vez aplicado los criterios sobre el conjunto total de usuarios registrados en el sistema, obtendremos una lista de objetos *TOUser* que contendrá los datos de cada usuario que cumplan los criterios.

- unlockUser(login: String): TOUser
  Esta función realizará el desbloqueo de un usuario e inicializará el número de intentos al sistema. Para que la función realice el desbloqueo es necesario que previamente esté el usuario bloqueado. Una vez realizado el desbloqueo la función devolverá un objeto *TOUser* con la información actualizada.

- modifyUser(login: String, password: String, name: String, company: String, email: String, type: Integer): TOUser
  Esta función actualizará la información de un usuario cuyo login se le pasa como parámetro. Antes de actualizar la información deberá de comprobar que el usuario está registrado y si se modifica el login o el email verificar que no lo tiene asignado otro usuario. Como resultado de la actualización la función devuelve un objeto *TOUser* con la información actualizada.

- deleteUser(login: String)
  Esta función dará de baja a un usuario del sistema.

- emailDatoUser(email: String, name: String, login: String, password: String)
  Esta función generará un comunicado de email con los datos de acceso al sistema al email recibido como parámetro.

COMPONENTE PPROJMANAGER

Para invocar cualquier función de este componente es necesario previamente verificar que el usuario sea un administrador.

- createProject(Key: String, name: String, descript: String)
  Esta función dará de alta a un proyecto en el sistema a partir de la información que se le pasa como parámetro. Dicha función comprobará previamente que ni el Key, ni el nombre ya se encuentran definidos en el sistema.

- createFromProject(KeyFrom: String, KeyTo: String, nameTo: String, descriptTo: String)
  Esta función dará de alta a un proyecto en el sistema a partir de la definición de otro Proyecto existente. Dicha función comprobará previamente que ni el KeyTo, ni el nameTo se encuentren ya definidos en el sistema. Además tendrá que verificar previamente que el Proyecto del que vamos a heredar sus características existe.

- listProject(): List<TOProject>
  Esta función es llamada cuando se quiere consultar todos los proyectos que tenemos definidos en el sistema. Una vez realizada devolverá una lista de objetos *TOProject* que contendrá los datos de cada proyecto.

- listProject(Key: String, name: String): List<TOProject>
  Esta función realizará una búsqueda de los proyectos que cumplen ciertas características (*que la clave contenga…y/o que el nombre contenga…*). A partir de la información que se le pasa como parámetro se montarán los criterios de búsqueda. Una vez aplicado los criterios sobre el conjunto total de proyectos registrados en el sistema, obtendremos una lista de objetos *TOProject* que contendrá los datos de cada proyecto que cumpla los criterios.

- createRolProject(project: TOProject, nameRol: String, descriptRol: String): TORol
  Esta función dará de alta un nuevo rol dentro de un proyecto en el sistema a partir de la información que se la pasa como parámetro. Dicha función deberá verificar que el objeto *TOProject* corresponde a un proyecto existente en el sistema, y que el rol que vamos a definir a partir de los atributos nombre y descripción del rol tampoco existe en el proyecto correspondiente al objeto *TOProject*.

- deleteRolProject(project: TOProject, rol: TORol)
  Esta función dará de baja un rol en un proyecto a partir de la información que se le pasa como parámetro. Si el proyecto o el rol no existen no se hará nada.

- modifyRolProject(project: TOProject, nameRol: String, descriptRol: String):TORol
  Esta función actualizará la información de un rol a partir de la información que se le pasa como parámetro. Antes de actualizar la información deberá de comprobar que el objeto *TOProject* está registrado y que el nombre del rol hace referencia a un rol del objeto *TOProject*. Como resultado de la actualización la función devuelve un objeto *TORol* con la información de la descripción actualizada.

- listRolProject(project: TOProject):List<TORol>
  Esta función es llamada cuando se quiere consultar todos los roles definidos en un proyecto que tenemos definido en el sistema. Una vez realizada devolverá una lista de objetos *TORol* que contendrá los datos de cada rol.

- createStateProject(project: TOProject, nameState: String, descriptState: String)
  Esta función dará de alta un nuevo estado dentro de un proyecto en el sistema a partir de la información que se la pasa como parámetro. Dicha función deberá verificar que el objeto *TOProject* corresponde a un proyecto existente en el sistema, y que el estado que vamos a definir a partir de la información de los atributos nombre y descripción tampoco existe en el proyecto correspondiente al objeto *TOProject*.

- deleteStateProject(project: TOProject, state: TOState)
  Esta función dará de baja un estado en un proyecto a partir de la información que se le pasa como parámetro. Si el proyecto o el estado no existen no se hará nada.

- modifyStateProject(project: TOProject, nameState: String, descriptState: String):TOState
  Esta función actualizará la información de un estado a partir de la información que se le pasa como parámetro. Antes de actualizar la información deberá de comprobar que el objeto *TOProject* está registrado y que la información del atributo nombre de estado exista también. Como resultado de la actualización la función devuelve un objeto *TOState* con la información de la descripción actualizada.

- listStateProject(project: TOProject):List<TOState>
  Esta función es llamada cuando se quiere consultar todos los estados definidos en un proyecto. Una vez realizada devolverá una lista de objetos *TOState* que contendrá los datos de cada estado.

- createWorkFlowProject(project: TOProject, stateFrom: TOState, rolFrom: TORol, statesTo: List<TOState>)
  Esta función dará de alta un nuevo workflow dentro de un proyecto en el sistema, a partir de la información que se la pasa como parámetro. Dicha función deberá verificar que los objetos *TOProject*, *TOState*, *TORol* corresponden respectivamente a un proyecto, un estado y un rol definido en el sistema. También deberá verificar que el objeto *TOWorkF* resultante no esté definido en el sistema.

- deleteWorkFProject(project: TOProject, workf: TOWorkF)
  Esta función dará de baja un workflow en un proyecto a partir de la información que se le pasa como parámetro. Si el proyecto o el workflow no existen no se hará nada.

- modifyWorkFProject(project: TOProject, workf: TOWorkF, statesTo: List<TOState>): TOWorkF
  Esta función actualizará la información de un workflow a partir de la información que se le pasa como parámetro. Antes de actualizar la información deberá de comprobar que los objetos *TOProject* y *TOWorkF* están registrados. Como resultado de la actualización la función devuelve un objeto *TOWorkF* con la información de la descripción actualizada.

- listWorkFProject(project: TOProject): List<TOWorkF>
  Esta función es llamada cuando se quiere consultar todos los estados definidos en un proyecto que tenemos definido en el sistema. Una vez realizada devolverá una lista de objetos *TOState* que contendrá los datos de cada estado.

- modifyProject(Key: String, name: String, descript: String): TOProject
  Esta función actualizará la información de un proyecto a partir de la información que se le pasa como parámetro. Además deberá comprobar que el proyecto que queremos modificar existe y que el nombre en caso de haberse modificado no existe en otro proyecto. Como resultado de la actualización la función devuelve un objeto *TOProject* con la información de la descripción actualizada.

- listUsersProject(project: TOProject): List<TOUserRol>
  Esta función es llamada cuando se quiere consultar todos los usuarios y sus roles que tienen definidos en un proyecto. Una vez realizada devolverá una lista de objetos *TOUserRol* que contendrá los datos de cada usuario, rol.

- createUserRProject(project: TOProject, user: TOUser, rol: TORol)
  Esta función dará de alta un nuevo usuario, rol dentro de un proyecto en el sistema a partir de la información que se la pasa como parámetro. Dicha función deberá verificar que los objetos *TOProject*, *TOUser*, *TORol* corresponden respectivamente a un proyecto, un usuario, un rol existentes en el sistema. También deberá verificar que el objeto *TOUserRol* resultante no esté definido en el sistema.

- deleteUserRolProject(project: TOProject, userRol: TOUserRol)
  Esta función dará de baja un usuario, rol en un proyecto a partir de la información que se le pasa como parámetro. Si el proyecto o la pareja usuario, rol no existen no se hará nada.

- modifyUserRolProject(project: TOProject, userRol: TOUserRol): TOUserRol
  Esta función actualizará la información de un proyecto a partir de la información que se le pasa como parámetro. Deberá verificar que los objetos *TOProject*, *TOUserRol* corresponden respectivamente a un proyecto y un par usuario, rol. También deberá verificar que el objeto *TOUserRol* resultante no esté definido en el sistema. Como resultado de la actualización la función devuelve un objeto *TOUserRol* con la información actualizada.

- deleteProject(name: String)
  Esta función dará de baja un proyecto a partir de la información que se le pasa como parámetro. Si el proyecto no existe no se hará nada.

COMPONENTE PARTEMANAGER

- createArtefact(project: TOProject, name: String, descript: String, detail: String, dateI: date, dateF: date, stateA: TOEstado , effortA: float, userA: TOUserRolF, userC: List <TOUserRolF>)
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función dará de alta un nuevo artefacto dentro de un proyecto en el sistema a partir de la información que se la pasa como parámetro. Dicha función deberá verificar que el objeto *TOProject* corresponde a un proyecto existente en el sistema.

- createUserArtefact(project: TOProject, artefact: TOArtefact, user: TOUserRol, dateI: date, dateF: date, porcentaje: int)
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función dará de alta a un usuario en un artefacto a partir de la información que se la pasa como parámetro. Dicha función deberá verificar que el objeto *TOProject* corresponde a un proyecto existente en el sistema y que los objetos *TOArtefact* y *TOUserRol* corresponden respectivamente a un artefacto y un usuario del proyecto que representa el objeto *TOProject*. Además se tendrá que asegurar que la carga que tenga dicho usuario en un periodo determinado no exceda del permitido (*este parámetro se define en la configuración*).

- modifUserArtefact(project: TOProject, artefact: TOArtefact, userRolF: TOUserRolF): TOUserRolF
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función actualizará la información del rol de un usuario definido en un artefacto a partir de la información que se le pasa como parámetro. Además deberá comprobar que los objetos *TOProject* y *TOArtefact* correspondan respectivamente a un proyecto y a un artefacto, existentes en el sistema. Como resultado de la actualización la función devuelve un objeto *TOUserRolF* con la información actualizada.

- deleteUserArtefact(project: TOProject, artefact: TOArtefact, userRolF: TOUserRolF)
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función dará de baja a un usuario con un rol definido dentro de un artefacto en un proyecto a partir de la información que se le pasa como parámetro. Si el proyecto o el artefacto no existen no se hará nada.

- listUserArtefact(project: TOProject, artefact: TOArtefact):List<TOUserRolF>
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función es llamada cuando se quiere consultar todos los usuarios y sus roles que tienen definidos en un artefacto de un proyecto. Una vez realizada devolverá una lista de objetos *TOUserRolF* que contendrá los datos de cada usuario, rol.

- listArtefactProject(project: TOProject):List<TOArtefact>
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función es llamada cuando se quiere consultar todos los artefactos que tiene definidos un proyecto. Una vez realizada devolverá una lista de objetos *TOArtefact* que contendrá los datos de cada artefacto.

- listArtefactProject(project: TOProject, name: String, dateA: date, stateA: TOEstado, userA: TOUserRolF):List<TOArtefact>
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función realizará una búsqueda de los artefactos asignados a un proyecto que cumplen ciertas características (*que el nombre contenga…y/o que la fecha es mayor o igual a…*, *etc.*). A partir de la información que se le pasa como parámetro se montarán los criterios de búsqueda. Una vez aplicado los criterios sobre el conjunto total de artefactos registrados en el sistema para un proyecto en concreto, obtendremos una lista de objetos *TOArtefact* que contendrá los datos de cada proyecto que cumpla los criterios.

- modifyArtefact(project: TOProject, artefact: TOArtefact):TOArtefact
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función actualizará la información de un artefacto a partir de la información que se le pasa como parámetro. Además deberá comprobar que los objetos *TOProject* y *TOArtefact* correspondan respectivamente a un proyecto y a un artefacto, existentes en el sistema. Como resultado de la actualización la función devuelve un objeto *TOArtefact* con la información actualizada.

- deleteArtefact(project: TOProject, name: String)
  Para invocar esta función es necesario previamente verificar que el usuario sea un usuario avanzado. Esta función dará de baja a un artefacto definido dentro de un proyecto a partir de la información que se le pasa como parámetro. Si el proyecto o el artefacto no existen no se hará nada.

- changeArtefact(project: TOProject, artefact: TOArtefact, state: TOEstado, user: TOUserRolF):TOArtefact
  Esta función es llamada cuando se quiere cambiar de estado y/o asignar a otro usuario un artefacto a partir de la información que se le pasa como parámetro. Esta función deberá asegurar que el artefacto se encuentra definido dentro del proyecto que representa el objeto *TOProject* y que el estado y/o usuario que se van a modificar en el artefacto es uno de los permitidos. Como resultado de la actualización la función devuelve un objeto *TOArtefact* con la información actualizada.

- createHistory(project: TOProject, artefact: TOArtefact)
  Esta función es llamada cada vez que realicemos una modificación del estado y/o usuario asignado a un artefacto. Dicha función guardará la información de un artefacto que se le pasa como parámetro. La función deberá de asegurar que el objeto *TOArtefact* representa un artefacto y que se encuentra definido en el proyecto que representa el objeto *TOProject*.

- listHistoryArtefact(project: TOProject, artefact: TOArtefact):List<TOHistory>
  Esta función es llamada cuando se quiere consultar la historia de un artefacto que se encuentra definido en un proyecto a partir de la información que se le pasa como parámetro. Una vez realizada devolverá una lista de objetos *TOHistory* que contendrá los datos de cada artefacto.

- atachmentArtefact(project: TOProject, artefact: TOArtefact, descript: String, file_nameE: String, file_NameI: String)
  Esta función es llamada cuando se quiere adjuntar un documento a un artefacto a partir de la información que se le pasa como parámetro. La función deberá de asegurar que el objeto *TOArtefact* representa un artefacto y que se encuentra definido en el proyecto que representa el objeto *TOProject*.

- emailChangeArtefact(project: TOProject, artefact: TOArtefact, user: TOUserRolF)
  Esta función podrá ser llamada cada vez que realicemos una modificación del estado y/o usuario asignado a un artefacto. Dicha función generará una comunicación a un usuario con la información de un artefacto que se le pasa como parámetro. La función deberá de asegurar que el objeto *TOArtefact* representa un artefacto y que se encuentra definido en el proyecto que representa el objeto *TOProject*.

COMPONENTE PINFOMANAGER

- listInform():List<TOInform >
  Esta función es llamada cuando se quiere consultar todos los tipos de informes que tenemos definidos en el sistema. Una vez realizada devolverá una lista de objetos *TOInform* que contendrá las descripciones de cada informe.

- informComp(project: TOProject): TOInfC
  Esta función es llamada cuando se quiere generar un informe completo a partir de la información que se le pasa como parámetro. La función deberá de asegurar que el objeto *TOProject* representa un proyecto en el sistema. Una vez realizada devolverá un objeto *TOInfC* que contendrá los datos del informe.

- informResu(project: TOProject): TOInfR
  Esta función es llamada cuando se quiere generar un informe resumen a partir de la información que se le pasa como parámetro. La función deberá de asegurar que el objeto *TOProject* representa un proyecto en el sistema. Una vez realizada devolverá un objeto *TOInfR* que contendrá los datos del informe.

- informEvol(project: TOProject): TOInfE
  Esta función es llamada cuando se quiere generar un informe de evolución a partir de la información que se le pasa como parámetro. La función deberá de asegurar que el objeto *TOProject* representa un proyecto en el sistema. Una vez realizada devolverá un objeto *TOInfE* que contendrá los datos del informe.

Para el transporte de los datos a la capa de presentación y al resto de capas utilizaremos el patrón de diseño Data Transfer Object[4] (DTO). Este patrón nos permitirá encapsular los datos de negocio dentro de un objeto de transferencia para reducir el número de llamadas a los mismos datos de negocio. En las funcionalidades presentadas anterior podemos encontrar las siguientes clases que representan objetos DTO.

- TOParam, contiene la información relativa a un parámetro.
- TOUser, contiene la información relativa a un usuario.
- TOProject, contiene la información relativa a un proyecto.
- TORol, contiene la información relativa a un rol.
- TOState, contiene la información relativa a un estado.
- TOWorkF, contiene la información de un workflow.
- TOUserRol, contiene la información relativa a un usuario con un rol determinado.
- TOUserRolF, contiene la información relativa a un usuario con un rol en un artefacto.
- TOArtefact, contiene la información relativa a un artefacto.
- TOHistory, contiene la información relativa a un artefacto en un momento determinado.
- TOInform, contiene la información relativa a los tipos de informes
- TOInfC, contiene la información relativa al Tipo de informe Completo
- TOInfR, contiene la información relativa al Tipo de informe Resumen
- TOInfE, contiene la información relativa al Tipo de informe Evolución

[4] **Referencia**: http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html

Por último siguiendo las buenas prácticas vamos a aplicar en los seis componentes especializados el patrón MVC[5]. Esto hará que cada uno de estos componentes tenga un componente que se encargará de la lógica de presentación (*validaciones de campos, re direccionamientos de acciones*) y otro componente que se encargará de la lógica de las vistas (*determina la siguiente vista a mostrar*).

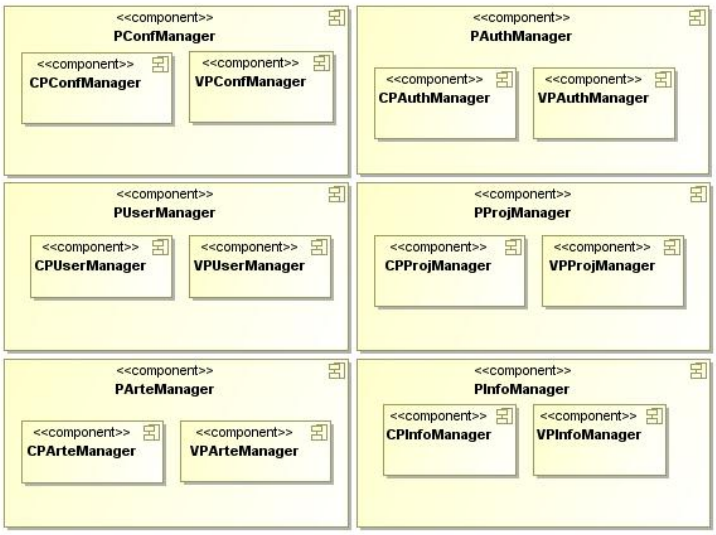

Figura 18: Componentes con patrón MVC

En el siguiente diagrama de componentes representamos como se relacionan todos los compontes que hemos definido en la *Capa de Presentación*.

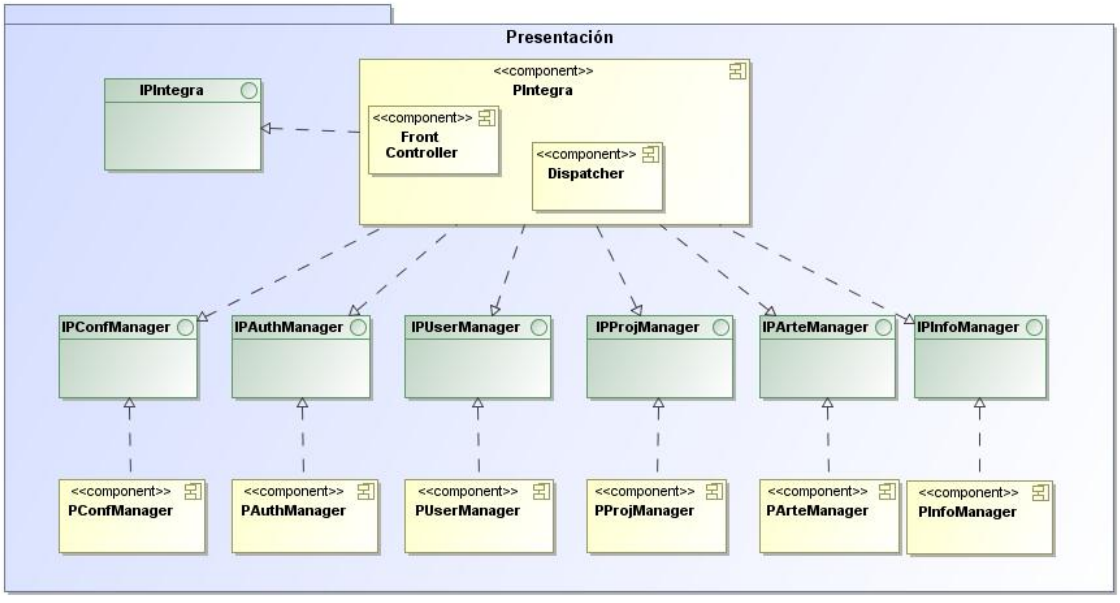

Figura 19: Diseño Capa de Presentación

[5] **Referencia**: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Capa de Lógica de Negocio

Esta capa será la encargada de atender las solicitudes de los usuarios al sistema, las cuáles nos llegarán desde la capa de presentación. Además, se encargará de interactuar con la capa de integración para ejecutar su lógica. Vamos a suponer que las operaciones que realicemos sobre los componentes se realizarán de forma síncrona (es decir, los usuarios esperan por la respuesta del componente servidor).

En este caso, definiremos un componente y una interfaz dentro de esta capa por cada uno de los componentes que hayamos definido en la *Capa de Presentación*. Cada componente ofrecerá a través de su interfaz las funcionalidades que pueden ser invocadas por la *Capa de Presentación*. Dichas funcionalidades son las mismas que las que se ofrecen desde la *Capa de Presentación* al usuario. En la siguiente tabla se muestran los seis componentes que hemos definido en la *Capa de Presentación* y, que interfaz y componente de la *Capa de Lógica de Negocio* se encargará de procesar sus peticiones.

Componente de Presentación	Interfaz Lógica de Negocio	Componente de Lógica de Negocio
PConfManager	IBLConfManager	BLConfManager
PAuthManager	IBLAuthManager	BLAuthManager
PUserManager	IBLUserManager	BLUserManager
PProjManager	IBLProjManager	BLProjManager
PArteManager	IBLArteManager	BLArteManager
PInfoManager	IBLInfoManager	BLInfoManager

Tabla 5: Relación interfaces, componentes

En el siguiente diagrama de componentes representamos como se relacionan los compontes de la *Capa de Presentación* con *la Capa de Lógica de Negocio* a través de las interfaces que hemos definido.

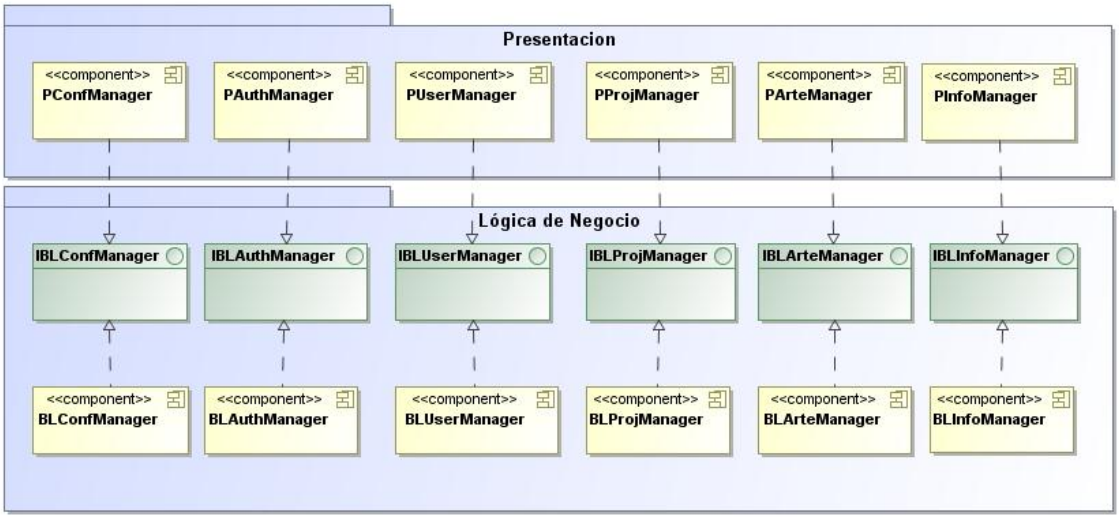

Figura 20: Diseño capa de Presentación-Lógica de Negocio

Capa de Integración

Esta capa se encargará de implementar el acceso a los datos que necesita nuestra aplicación.

En este caso, al igual que en la *Capa de Lógica de Negocio* con la *Capa de Presentación*, definiremos un componente y una interfaz dentro de esta capa por cada uno de los componentes que hayamos definido en la *Capa de Lógica de Negocio*. Cada componente ofrecerá a través de su interfaz las funcionalidades que pueden ser invocadas por la *Capa de Lógica de Negocio*. En la siguiente tabla se muestran los seis componentes que hemos definido en la *Capa de Lógica de Negocio* y, que interfaz y componente de la *Capa de Integración* se encargará de procesar sus peticiones.

Componente de Lógica de Negocio	Interfaz Integración	Componente de Integración
BLConfManager	IIConfManager	IConfManager
BLAuthManager	IIAuthManager	IAuthManager
BLUserManager	IIUserManager	IUserManager
BLProjManager	IIProjManager	IProjManager
BLArteManager	IIArteManager	IArteManager
BLInfoManager	IIInfoManager	IInfoManager

Tabla 6: Relación interfaces, componentes

En el siguiente diagrama de componentes representamos como se relacionan los compontes de la *Capa de Lógica de Negocio* con la *Capa de Integración* a través de las interfaces que hemos definido.

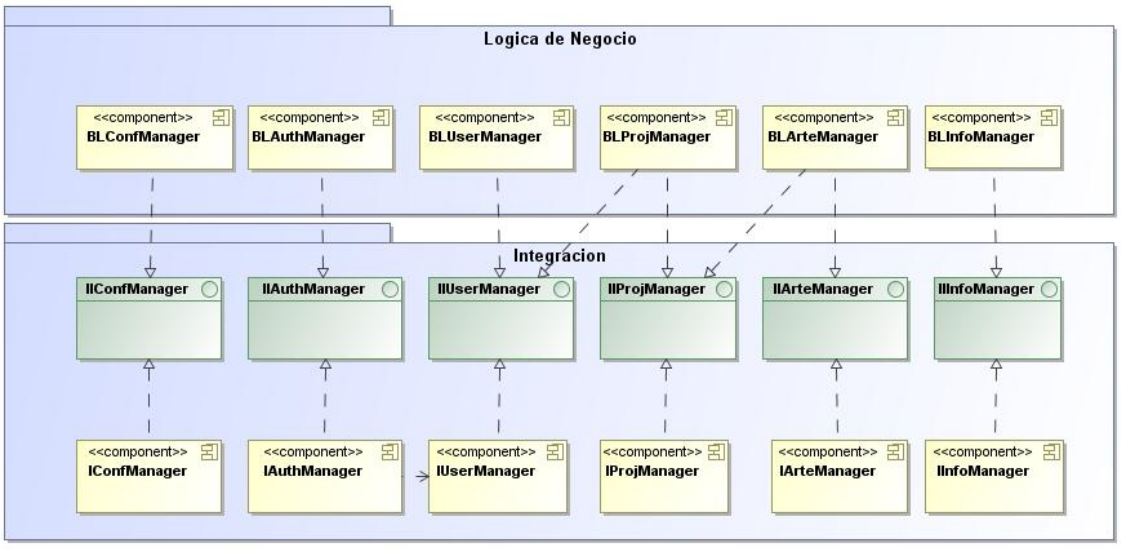

Figura 21: Diseño capa de Lógica de Negocio-Integración

Por último vamos a representar las relaciones que existen entre los componentes de la *Capa de Integración* y el *Diagrama de Clases* que hemos obtenido en la etapa de análisis.

COMPONENTE ICONFMANAGER

Este componente ofrece una serie de funcionalidades a la capa de negocio relacionadas con la gestión de configuración del sistema. Por tanto podemos deducir que la única clase del diagrama de clases que utilizará será *Config* tal como vemos en la siguiente Figura.

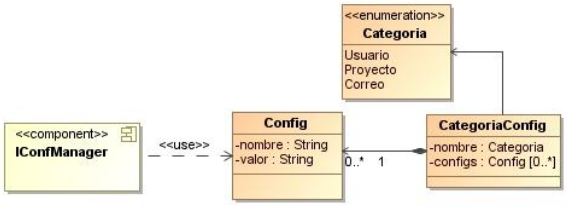

Figura 22: Componente IConfManager

COMPONENTE IAUTHMANAGER

Este componente ofrece una serie de funcionalidades a la capa de negocio relacionadas con el acceso al sistema. Por tanto podemos deducir que la única clase del diagrama de clases que utilizará será *Usuario* tal como vemos en la siguiente Figura.

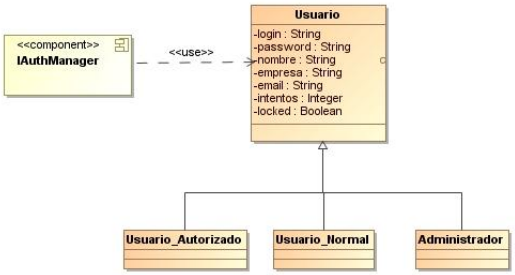

Figura 23: Componente IAuthManager

COMPONENTE IUSERMANAGER

Este componente ofrece una serie de funcionalidades a la capa de negocio relacionadas con la gestión de los usuarios en el sistema. Por tanto podemos deducir que la única clase del diagrama de clases que utilizará será *Usuario* tal como vemos en la siguiente Figura.

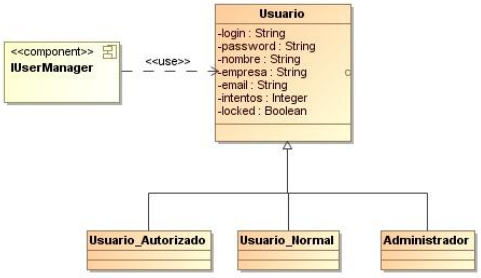

Figura 24: Componente IUserManager

COMPONENTE IPROJMANAGER

Este componente ofrece una serie de funcionalidades a la capa de negocio relacionadas con la gestión de los proyectos en el sistema. Por tanto podemos deducir que las clases del diagrama de clases que utilizará serán *Proyecto, Rol, Estado, WorkFlow, Usuario-Rol y Usuario* tal como vemos en la siguiente Figura.

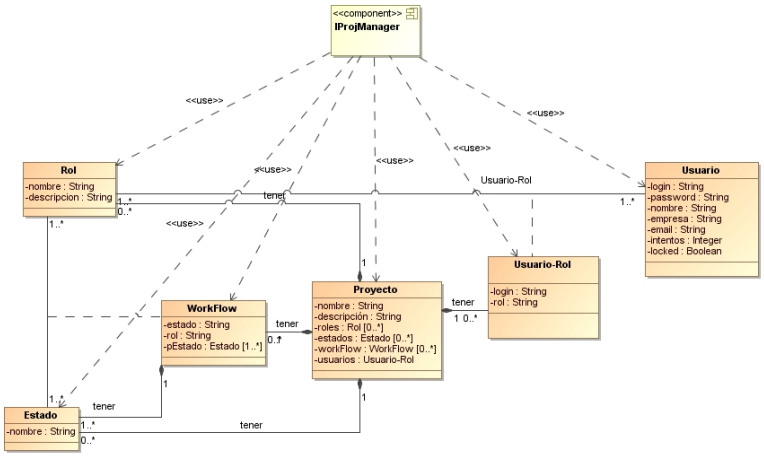

Figura 25: Componente IProjManager

COMPONENTE IARTEMANAGER

Este componente ofrece una serie de funcionalidades a la capa de negocio relacionadas con la gestión de los artefactos en el sistema. Por tanto podemos deducir que las clases del diagrama de clases que utilizará serán *Proyecto, Artefacto, Historia, Artefacto-U, Documento, WorkFlow y Usuario-Rol* tal como vemos en la siguiente Figura.

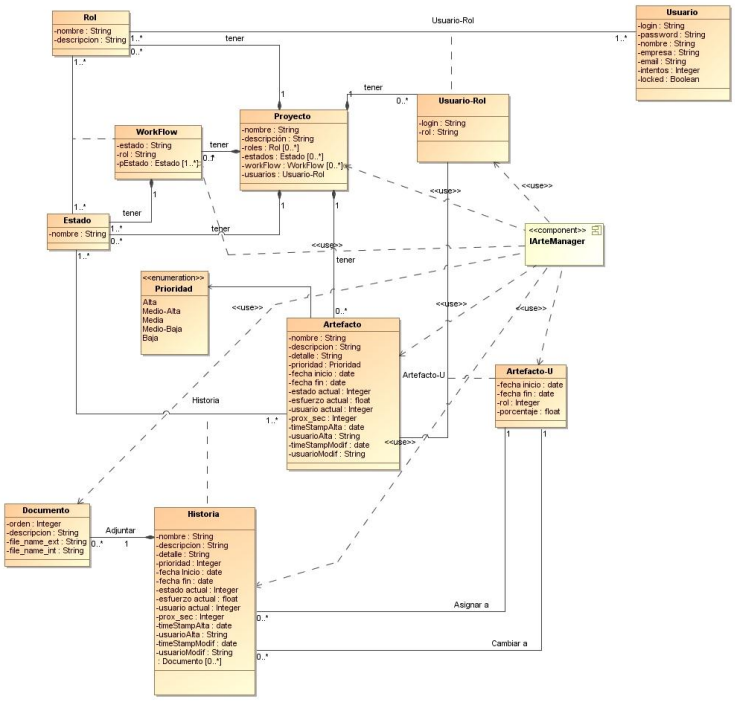

Figura 26: Componente IArteManager

COMPONENTE IINFOMANAGER

Este componente ofrece una serie de funcionalidades a la capa de negocio relacionadas con la generación de informes en el sistema. Por tanto podemos deducir que las clases del diagrama de clases que utilizará serán *Informe, Proyecto, Artefacto, Historia* tal como vemos en la siguiente Figura.

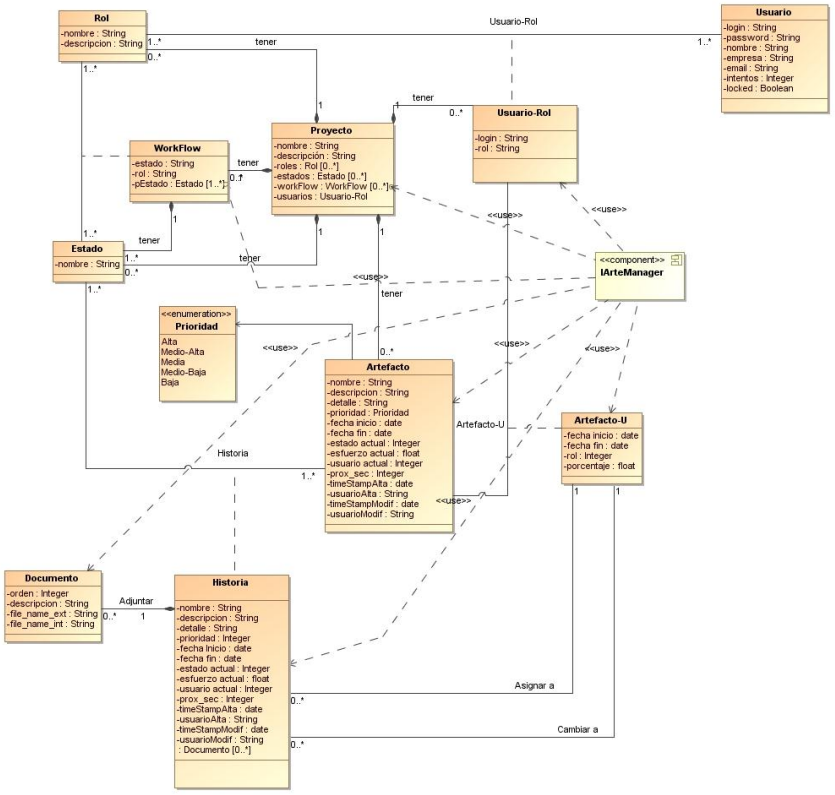

Figura 27: Componente IInfoManager

Diagrama de Diseño Final

A continuación mostraremos el diagrama de componentes completo que hemos ido construyendo a lo largo de esta fase de diseño.

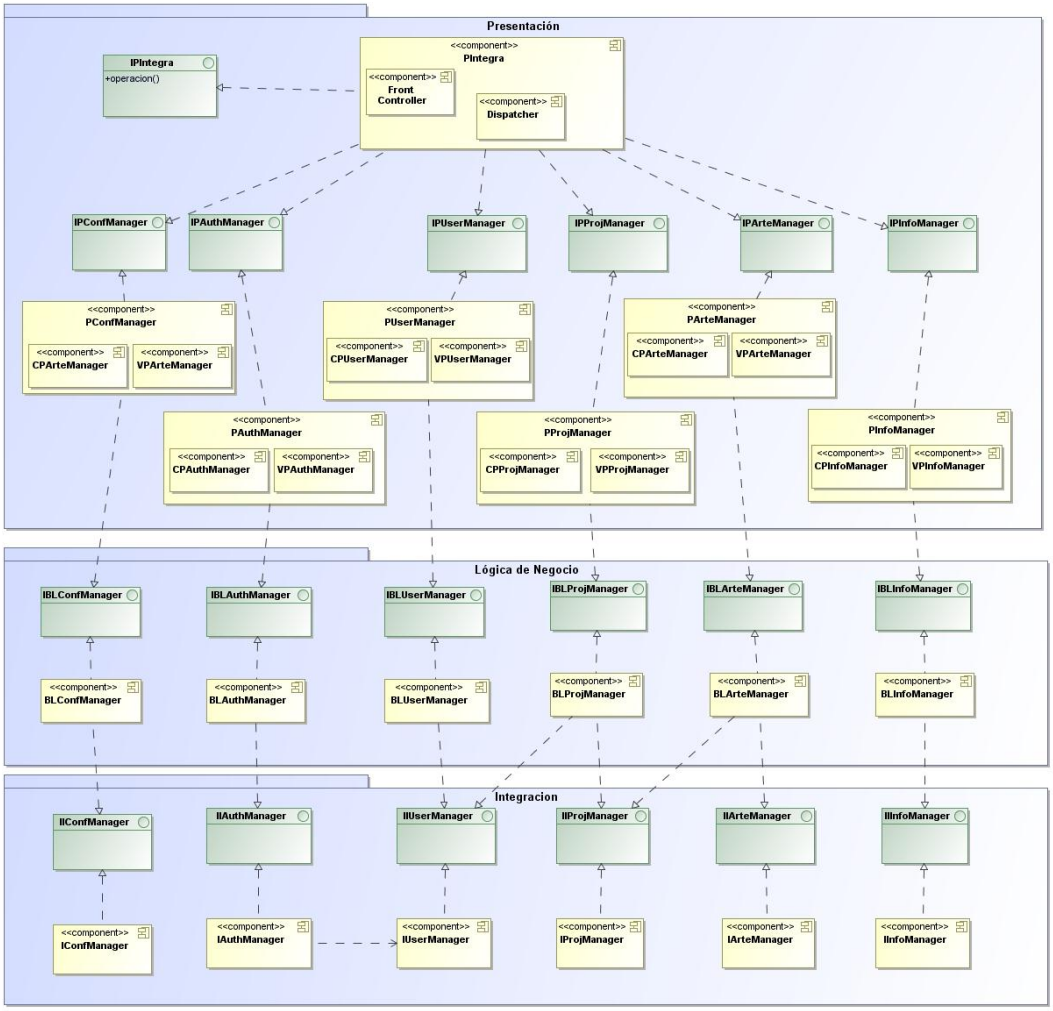

Figura 28: Diseño final

Por último, vamos a representar mediante un Diagrama de Secuencia[6] el caso de uso *CU_01:Identificarse en la Aplicación* para ver como interacciona en nuestro diseño propuesto los diferentes componentes de las distintas capas para resolver dicha petición.

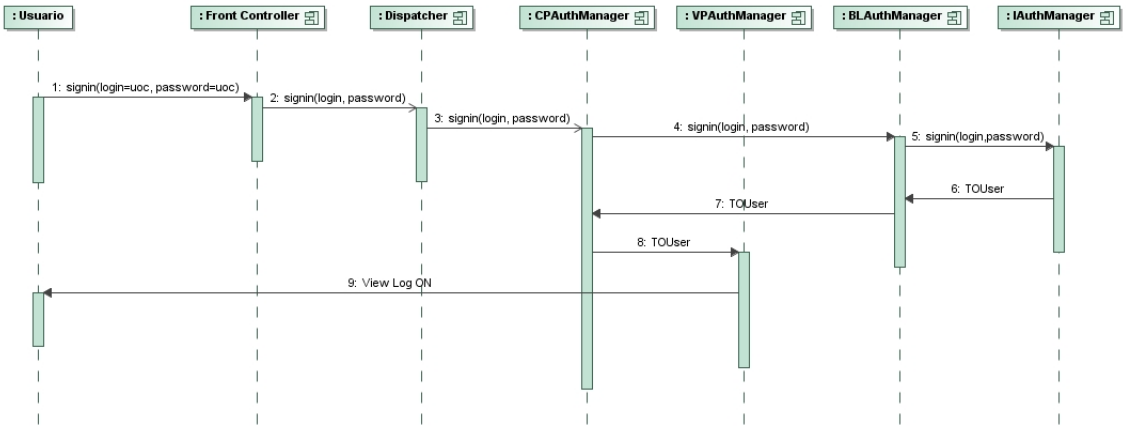

Figura 29: Diagrama de Secuencia

[6] **Diagrama de Secuencia**, muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

Prototipo

Para la realización del prototipo se ha usado una herramienta de Prototipaje llamada Justinmind Prototyper[7]. Dicha Herramienta nos permite realizar un primer prototipo, sin necesidad de realizar ninguna implementación inicial.

Para su ejecución sólo es necesario descomprimir el archivo *integra.zip* y ejecutar el archivo *index.html*. Una vez arrancado el prototipo, nos encontraremos con una pantalla como la de la siguiente figura:

Figura 30: Pantalla Inicial del Prototipo

Dicho prototipo nos permite realizar la navegación por la aplicación de dos formas. Una primera que sería ir accediendo a las diferentes vistas de la aplicación, desde el panel de la izquierda y otra que sería la navegación natural que consistiría en ir interactuando con las diferentes vistas por las que nos va guiando la aplicación.

Para poder usar el prototipo con una navegación natural, se han definido varios usuarios con diferentes tipos de autorizaciones, los cuales tienen definidos como password su nombre de usuario. Los nombres de usuarios son:

- user0, usuario Administrador
- user1, usuario Avanzado
- user2, usuario Normal

[7] **Referencia:** http://www.justinmind.com/

Implementación

A continuación realizaremos[8] una descripción breve de las tecnologías que hemos utilizado en la implementación del prototipo, así como las herramientas utilizadas.

Tecnologías

- Hibernate, framework que hemos utilizado para gestionar la capa de persistencia. Dicha herramienta nos facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

- JavaMail, biblioteca de java que hemos utilizado para el envío de e-mails a los usuarios.

- Struct, framework que hemos utilizado en la capa de presentación para implementar el modelo-vista-controlador (MVC) bajo la plataforma Java EE (Java Enterprise Edition).

- JQuery, se trata de una biblioteca de javascript que nos simplifica el trabajo en el tratamiento de eventos, animaciones, interacciones, etc., en la capa de presentación.

- HTML, siglas de HyperText Markup Language (“*lenguaje de marcado de hipertexto”*). Se ha utilizado en la elaboración de las páginas web.

- CSS, siglas de Cascading Style Sheets (*“hojas de estilo en cascada”*). Se trata de un lenguaje que es usado para estructurar la información en los documentos HTML, XML y XHTML.

- JSP, siglas de JavaServer Pages. Es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

- JSTL, siglas de Java Standard Template Library (“*biblioteca de etiquetas estándar para Java”*). Se trata de una biblioteca que implementa un conjunto de funciones de uso frecuente en aplicaciones JSP.

- JavaScript, se trata de un lenguaje de programación interpretado, que puede se utilizado tanto desde la parte servidora como cliente. En el prototipo, se ha usado en la parte del cliente para mejorar las páginas web.

- J2EE, siglas de Java Platform, Enterprise Edition. Define un conjunto de especificaciones de APIs Java para la construcción de aplicaciones empresariales. Proporciona un modelo para el desarrollo de aplicaciones distribuidas multicapa.

[8] Para la descripción de las tecnologías se ha buscado información principalmente en http://es.wikipedia.org

Herramientas

- Eclipse[9], es un entorno de desarrollo integrado de código abierto multiplataforma. Todo el prototipo se ha desarrollado con dicha plataforma.

- Xampp[10], es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl.

- MySQLWorkBench[11], es una herramienta visual de diseño de bases de datos que integra desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

- MySQL[12], es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

- TOMCAT[13], funciona como un contenedor de servlets.

[9] http://www.eclipse.org/

[10] http://www.apachefriends.org/es/xampp.html

[11] http://www.mysql.com/products/workbench/

[12] http://www.mysql.com/

[13] http://tomcat.apache.org/

Pruebas

Durante el desarrollo de la aplicación se han ido realizando diversos tipos de pruebas.

Pruebas Unitarias, las hemos ido realizando a medida que se iban implementando las funcionalidades de la aplicación. Dichas pruebas han consistido en verificar que cada una de las funciones implementadas han funcionado tal como se esperaba con la definición de un conjunto de casos de uso de prueba.

Pruebas de Integración, nos ha permitido verificar que un módulo funciona correctamente, tanto individualmente como de forma colaborativa con otros módulos.

Pruebas funcionales, nos ha permitido verificar que una funcionalidad que debe de ofrecer la aplicación, se encuentra implementada y que funciona correctamente.

Instalación de la Aplicación

Para que la aplicación funcione correctamente es necesario seguir los siguientes pasos:

a) Definir la Base de datos que utiliza la aplicación.

   Para tal fin se suministra con este documento un archivo denominado *BDIntegra.sql* que contiene los Scritps necesarios para la creación de dicha base de datos. Además de crear la Base de datos, realizará una serie de inserts para la carga de los parámetros de configuración iniciales de la aplicación y la definición del usuario Administrador.

b) Una vez creada la Base de datos, procederemos a la edición del archivo *Integra.xml*, que también se suministra con este documento. Dentro de dicho archivo habrá que modificar los siguientes parámetros:

   username : nombre del usuario con privilegios de acceso a la Base de Datos
   password : contraseña del usuario con privilegios de acceso a la Base de Datos
   driverClassName : jdbc que tenemos instalado en el servidor

c) Una vez modificado dicho archivo, lo copiaremos al servidor de aplicaciones en la ruta "..*conf**Catalina**localhost*".

d) Despliegue de la aplicación en el servidor.

   Copiaremos el archivo *Integra.war*, que también se suministra con este documento y lo colocaremos en la ruta donde tengamos nuestro servidor de aplicaciones.

Conclusiones

A continuación procederé a enumerar las conclusiones que he podido obtener con la realización de este Proyecto Fin de Carrera (PFC).

- La realización de este Proyecto me ha permitido profundizar en el estudio de diversas tecnologías web.
- También me ha permitido ver como el uso de nuevas tecnologías tiene un coste alto, debido a su aprendizaje.
- Y como de importante es, determinar correctamente los factores de riesgos que puedan hacer peligrar un proyecto.

Índice de Ilustraciones

Índice de Tablas

Bibliografía

- Thomas A. Powell (2001), Manual de Referencia HTML 4. Editorial Osborne McGraw-Hill.
- Eguíluz Pérez, Javier (2009), CSS Avanzado. Editorial www.librosweb.es
- Cavaness, Ch. (2002), Jakarta Struts. Editorial O'Reilly.
- Arnold Doray (2006), Beginning Apache Struts: From Novice to Professional. Editorial Apress.
- Jérôme LAFOSSE (2010), Struts 2 El framework de desarrollo de aplicaciones Java EE Ediciones eni.
- Bear Bibeault,Yehuda Katz (2010), Jquery 1.4. Editorial Anaya Multimedia.
- Peak , Patrick y Heudecker Nick (2006), Hibernate Quickly. Editorial Manning.
- Ramón Rodríguez, José Ramón y Lamarca, José Ignacio (2006) .Metodología y Gestión de Proyectos Informáticos Barcelona. Editorial UOC.
- CABOT, J. y otros (2006). Ingeniería del Software Orientada a Objetos. Barcelona. Editorial UOC.
- CABOT, J. y otros (2006). Ingeniería del Software de Componentes y Sistemas Distribuidos. Barcelona. Editorial UOC.
- CANYET, J.M. (2006). Interacción humana con los Ordenadores. Barcelona. Editorial UOC.