Proyecto SAR

Documento de Diseño

EC-DD. Versión 1.0
06/03/2018
Estatus: Restringido

Resumen

En este documento vamos a presentar la parte que corresponde al diseño que hemos planteado de nuestra aplicación.

Se ha desarrollado un diagrama de clases del que se partirá a la hora de codificar la aplicación, desarrollando también lo que contiene todas y cada una de las clases que lo forman y explicando por qué creemos que debe de ser así.

Por último, se han realizado los diagramas de secuencia de los distintos casos de uso que hemos propuesto en el documento de análisis y que pertenecen a la primera iteración del diseño de la aplicación que se nos propone.

Índice de Contenidos

1. Descripción de la Arquitectura del Sistema

En este apartado vamos a explicar todas las clases y relaciones que podemos observar en el diagrama de clases de la aplicación que se encuentra en el siguiente apartado del documento.

La aplicación cuenta con una parte de usuarios. Las clases en ella son las siguientes:

- **Usuario**: es una clase abstracta que cubre el global de todos los usuarios de la aplicación. De esta clase heredan los considerados usuarios finales de la aplicación. Contiene los siguientes atributos:
    - id: identificador único para cada uno de los usuarios de la aplicación.
    - nombre: nombre del usuario.
    - apellidos: apellido del usuario.
    - e-mail: correo electrónico del usuario.
    - contraseña: necesaria para autenticarse en la aplicación.

  Contiene los métodos *autenticarse() y listarProyectosExistentes().* Ambos métodos serán sobreescritos luego en las clases que heredan de Usuario que los necesiten implementar por sí mismos.

- **Trabajador**: clase abstracta. Hereda de Usuario y es la clase que modela de la forma más general a los usuarios que trabajan para la empresa de la aplicación, que son, el *Ingeniero y el Director de Proyecto.* Contiene un atributo *sueldo* que hemos decido incluir porque nos parecía lógico saber lo que cobran los empleados de la empresa.

- **Director e Ingeniero:** heredan de Trabajador y sobreescriben los métodos *autenticarse() y listarProyectosExistentes()* ya que cada uno de los usuarios tiene unos privilegios dentro de la aplicación.

- **Cliente**: es la última clase de tipo Usuario de la aplicación y en esta solamente se sobreescribe el método *autenticarse()* ya que no tienen privilegios para listar los proyectos existentes en la aplicación.

Una vez terminada la explicación de los usuarios vamos a pasar a explicar todo lo que tiene que ver con los requisitos y el proyecto, primero se comentarán las clases por separado y después porque están relacionadas unas con otras de esa manera.

- **Proyecto:** contiene 4 atributos:
    - id: identificador único por proyecto.
    - nombre: nombre que identifica al proyecto.
    - descripción: breve descripción del proyecto.
    - fechaCreacion: cuando entra este proyecto en el sistema.

Esta clase contiene 5 métodos, todos con mucha importancia en la aplicación. Estos métodos son los siguientes:

- crearProyecto(): crea un nuevo proyecto y los añade a la lista de proyectos del Sistema.
- listarRequisitos(): lista todos los requisitos que tiene el proyecto.
- consultarRequisitoPor(): permite hacer un filtrado de los distintos requisitos del proyecto por fecha de creación, nombre del proyecto y por estado.
- añadirCliente(): añade un nuevo cliente al proyecto.
- añadirIngeniero(): asigna un nuevo Ingeniero al proyecto para que se ocupe de nuevas tareas de programación.
- consultarRequisito(): que permite filtrar los distintos requisitos asociados a un proyecto por fecha, nombre, si es funcional o no y por estado del desarrollo en el que se encuentra.

Esta clase está relacionada con una agregación fuerte con el equipo de ingenieros y los requisitos, ya que sin ellos no sería posible realizar el proyecto, conteniendo una lista tanto de Requisitos como de Ingenieros. Además, todo proyecto tiene un Director y un Cliente. También puede tener o no, solicitudes de requisitos por parte del Cliente, por lo tanto, tendrá una agregación débil con la clase Solicitud. Por último, ya que el Director va a poder pedir un reporte del proyecto, también tenemos una relación con la clase Reporte.

- **Requisito:** esta clase es muy relevante ya que todos los proyectos están compuestos por distintos requisitos. Los atributos que componen la clase son:
  - id: identificador único por requisito.
  - nombre: nombre del requisito.
  - descripcion: describe en qué consiste el requisito.
  - creadoPorSolicitud: si el requisito se crea en respuesta a una solicitud del cliente será TRUE, si no FALSE.
  - funcional: si es un requisito funcional esta variable será TRUE, en caso de que sea no funcional el valor que tomará será FALSE.
  - fechaCreacion: fecha en la que se crea el requisito.

  También va a contar con un atributo de tipo EstadoRequisito, que es un enumerado que nos informa de si el requisito se encuentra EN_PROCESO, COMPLETADO o SIN_EMPEZAR.
  Esta clase contiene los siguientes métodos:
  - nuevoRequisito(): permite crear un requisito nuevo dentro de un proyecto.
  - asociarArtefacto(): asocia uno de los artefactos que tengamos creados en la aplicación a su correspondiente requisito.
  - listarArtefactos(): como cada requisito contendrá una lista con los artefactos que lo componen, nos parece útil tener una función que liste todo y cada uno de ellos.

- modificarRequisito(): permite modificar alguno de los atributos de un requisito en el momento en el que lo desee alguno de los usuarios con permisos de la aplicación.

Esta clase está relacionada con una agregación débil a la clase Artefacto, ya que puede contener alguno o ninguno.

- **Artefacto:** los artefactos tienen relación con los requisitos. Hemos considerado que tienen los siguientes atributos:
  - id: identificador único del artefacto.
  - nombre: nombre que identifica al artefacto.
  - descripcion: descripción concreta de qué es el artefacto.
  - fechaCreacion: que describe la fecha en la que se crea un artefacto.

  No se especifica mucho sobre los artefactos en el enunciado de la práctica, por lo que hemos definido los subtipos que se mencionan simplemente como clases que heredan de "Artefacto". Las clases que heredan de artefacto son **Prototipo, Diagrama** y **Prueba.** Se distinguen también entre distintos tipos de diagramas y pruebas, que heredan de las clases Diagrama y Prueba, respectivamente. Los diagramas pueden ser **D_TransicionEstados, D_Componentes y D_Clases,** a su vez, las pruebas **P_Plan** o **P_Caso.**

- **Solicitud:** esta clase está relacionada con la solicitud de requisitos por parte de los clientes de un proyecto. Por ello, la clase *Proyecto* va a contener a todas las solicitudes que haya en el sistema. Cada solicitud va a tener asociada un *Cliente* y también un *Director*, el cliente hará esa solicitud y el directo aceptará o rechazará.
  Una solicitud tiene los siguientes atributos:
  - id: identificador único asociado a la solicitud.
  - nombre: nombre de la solicitud.
  - descripcion: descripción detallada de en qué consiste la solicitud.
  - fechaCreacion: informa de cuando se crea una solicitud.

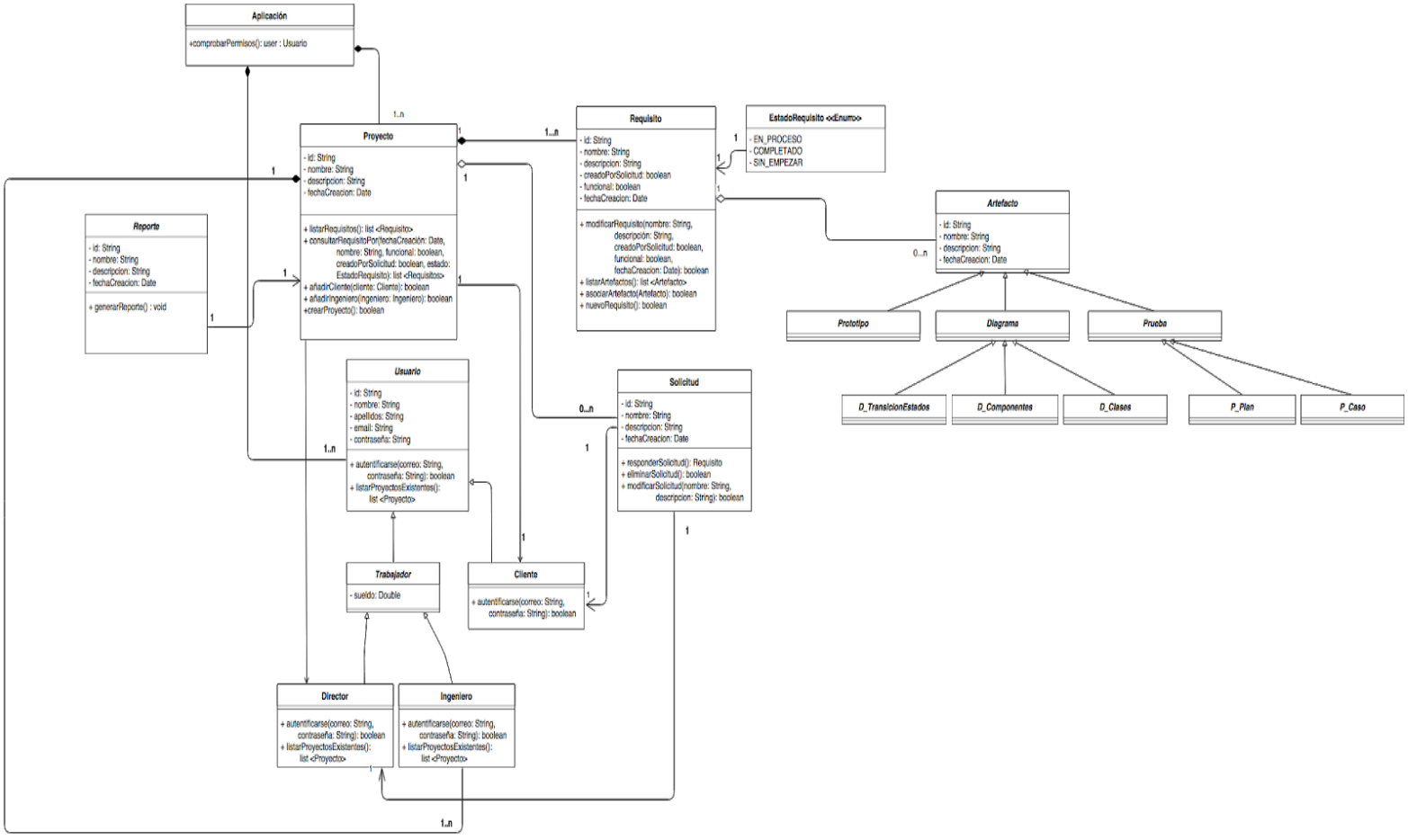
  Contiene además tres métodos que nos permiten operar con las distintas solicitudes:
  - responderSolicitud(): acepta o deniega la solicitud de requisito que el cliente quiere añadir al proyecto.
  - eliminarSolicitud(): elimina una solicitud del sistema.
  - modificarSolicitud(): se modifica una solicitud de algún requisito.

- **Reporte:** otro requisito que tiene la aplicación es la de generar reportes de un proyecto. Solo pueden ser realizados por el Director, por lo tanto, la clase reporte llevará asociado un proyecto, que tienen Director y al que se le permitirá reportar cuando lo desee. Contiene los mismos atributos que la clase anterior ya que no tenemos mucha más descripción de la clase y un

método generarReporte() que generará el reporte cuando el director del proyecto lo estime oportuno.

- **Aplicacion:** esta clase se utiliza para guardar todos los proyectos de la aplicación y todos los usuarios que forman la empresa. Además, su funcionalidad más importante va a ser la de comprobar en todo momento los permisos de los distintos usuarios para acceder a los datos de los proyectos. Para ello, contará con un método comprobarPermisos() que nos permitirá controlar lo anteriormente descrito.

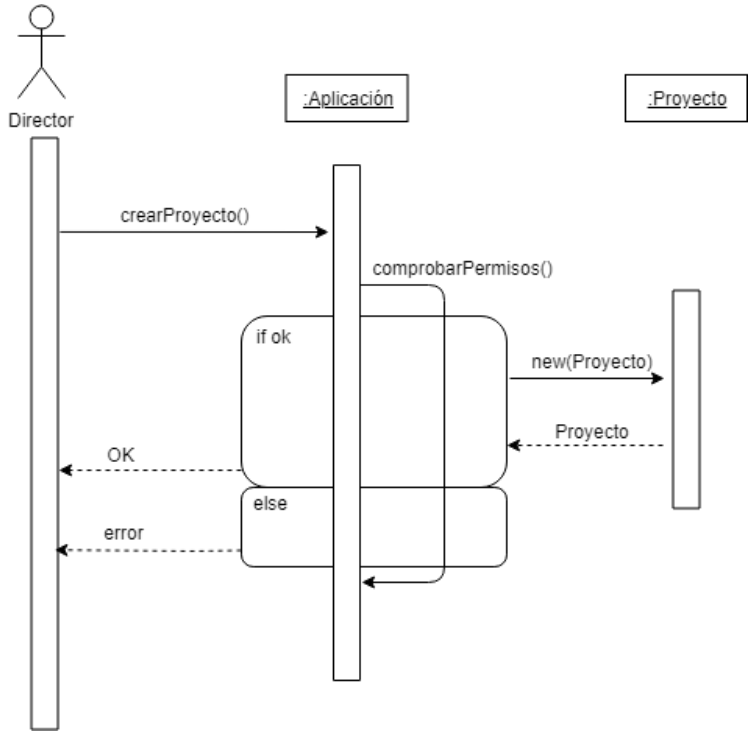
2. Diagrama de Clases



El fichero con el diagrama de clases original se encuentra en la carpeta entregada por
si algo no quedase claro o no se viese bien en el documento.

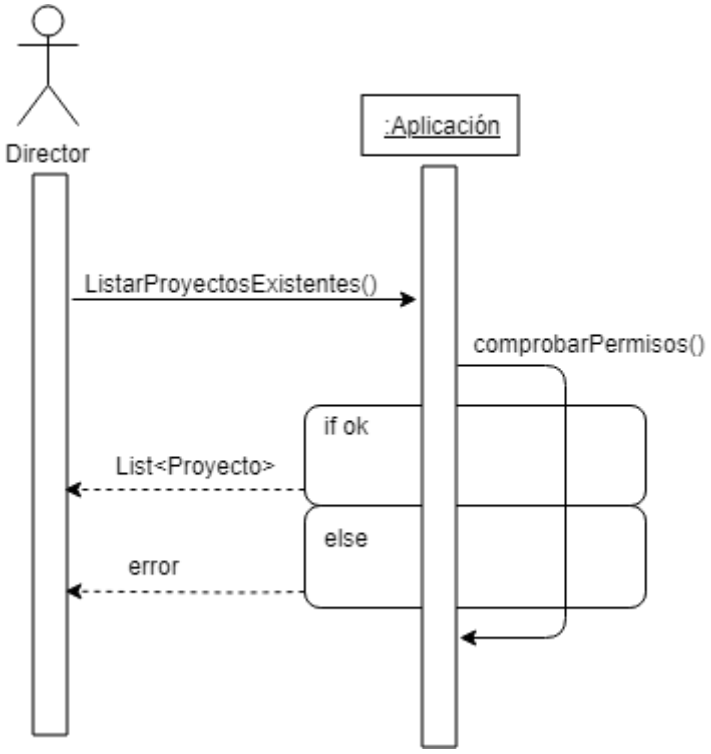
3. Diagramas de Secuencia

3.1 Diagrama de Secuencia 1 – Especificar Requisito



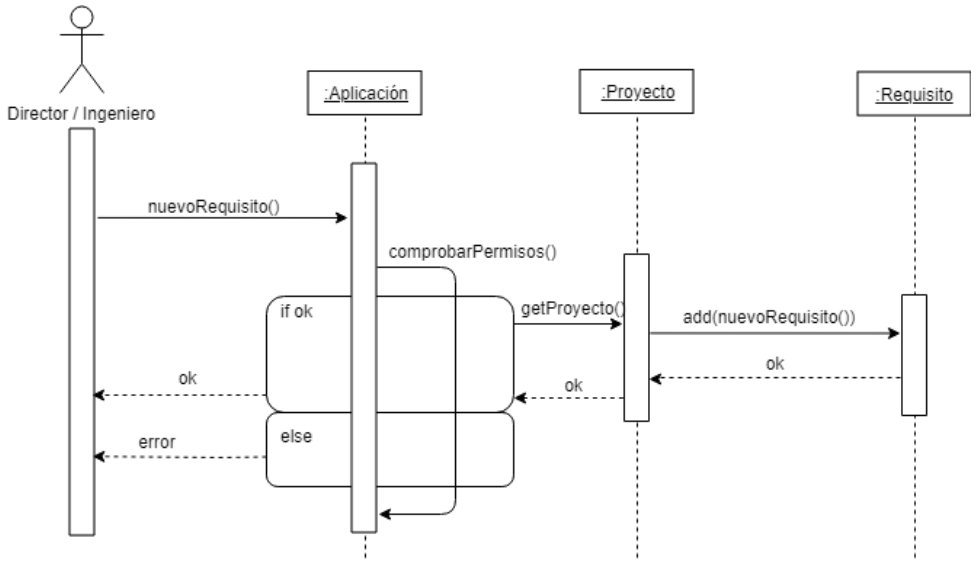
En este diagrama representamos la funcionalidad de crear proyectos, cuyo actor debe ser únicamente el director; en esta acción, el director accederá a la aplicación, esta comprobará que tiene los requisitos necesarios para cumplir esta funcionalidad, es decir, si el usuario que está accediendo es un director, o un ingeniero o un cliente, y posteriormente, si tiene permiso para realizar esta funcionalidad, el sistema accederá a su lista de proyectos y añadirá uno nuevo, en caso de que todo funcione correctamente se devolverá “OK”, pero si sucede algún error, la función devolverla “ERROR”.

3.2 Diagrama de Secuencia 2 - Listar Proyectos



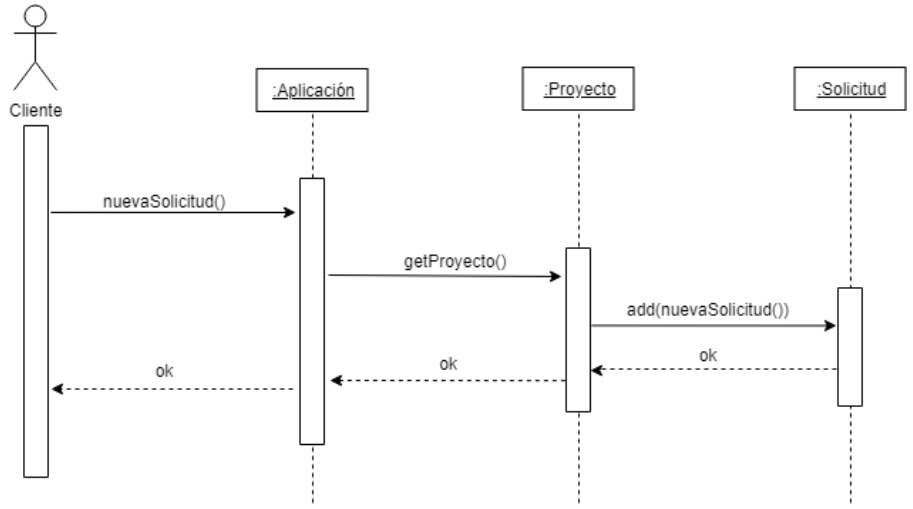
En este diagrama representamos la funcionalidad de listar proyectos, cuyo actor debe ser únicamente el director; en esta acción, el director accederá a la aplicación, esta comprobará que tiene los requisitos necesarios para realizar esta funcionalidad, y en caso afirmativo (si es el director quien está accediendo), la función devolverá la lista de proyectos que contiene la aplicación, en caso de que ocurra cualquier error, la función devolverá “ERROR”.

3.3 Diagrama de Secuencia 3 - Responder Solicitud de Requisito



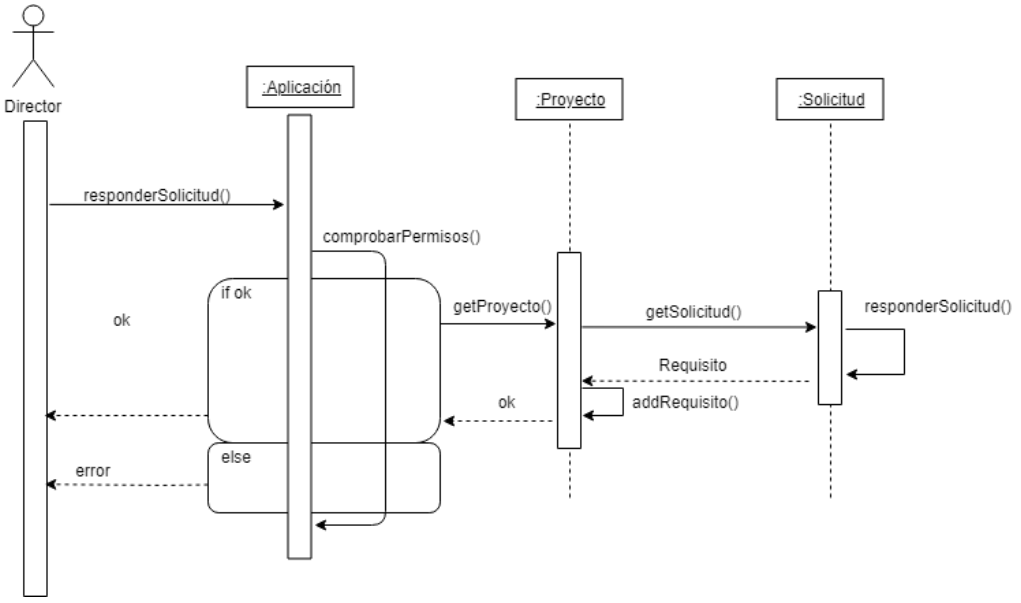
En este diagrama representamos la funcionalidad de definir un nuevo requisito, cuyo actor puede ser tanto el director, como ingeniero; en esta acción, el usuario accederá a la aplicación, esta comprobará que tiene los requisitos necesarios para realizar esta funcionalidad, y en caso afirmativo, la aplicación accederá al proyecto específico, y en este, introducirá un nuevo requisito en su lista de requisitos, la función devolverá "OK" en caso de que no ocurra ningún fallo, o en caso de que ocurra cualquier error, devolverá "ERROR".

3.4 Diagrama de Secuencia 4 – Crear Proyecto



En este diagrama representamos la funcionalidad de crear una nueva solicitud, cuyo actor debe ser el cliente; en esta acción, el usuario accederá a la aplicación, la aplicación accederá al proyecto específico, y en este, introducirá una nueva solicitud en su lista de solicitudes la función devolverá "OK" en caso de que no ocurra ningún fallo, o en caso de que ocurra cualquier error, devolverá "ERROR".

3.5 Diagrama de Secuencia 5 – Crear Solicitud de Requisito



En este diagrama representamos la funcionalidad de responder a una solicitud, cuyo actor debe ser el director; en esta acción, el usuario accederá a la aplicación, esta comprobará que tiene los requisitos necesarios para realizar esta funcionalidad, y en caso afirmativo, la aplicación accederá al proyecto específico, y en este, cogerá la solicitud y verá si la acepta o la rechaza, en caso de que la acepte, esta solicitud se convertirá en un requisito, y este requisito será añadido en la lista de requisitos del proyecto , la función devolverá “OK” en caso de que no ocurra ningún fallo, o en caso de que ocurra cualquier error, devolverá “ERROR”.

4. Glosario

TÉRMINO	DESCRIPCIÓN