

Contenido

10. Aplicación de patrones	1
10.1. Patrón de arquitectura Modelo-Vista-Controlador.....	1
10.2. Patrones de diseño	1
10.2.1. Nombre del patrón 1	1
10.2.2. Nombre del patrón 2	2
10.2.3. Nombre del patrón 3	3

Anexo: Diagramas de colaboración con clase experto.

10. Aplicación de patrones

10.1. Patrón de arquitectura Modelo-Vista-Controlador

<Explicación textual de como se ha utilizado el patrón e imagen del diagrama (clases y/o componentes) en el que se vea su aplicación.>

10.2. Patrones de diseño

<Para cada uno de los patrones de diseño elegidos, al menos tres, explicación textual de como se ha utilizado el patrón e imagen del diagrama (clases y/o colaboración) en el que se vea su aplicación. Se utilizarán patrones vistos en teoría y no se puede utilizar el patrón Controlador ya que se usa en el patrón de arquitectura>

10.2.1. Fachada de Base de Datos

La fachada ofrece una interfaz única y común al servidor de base de datos para todas las clases y funciones del sistema.

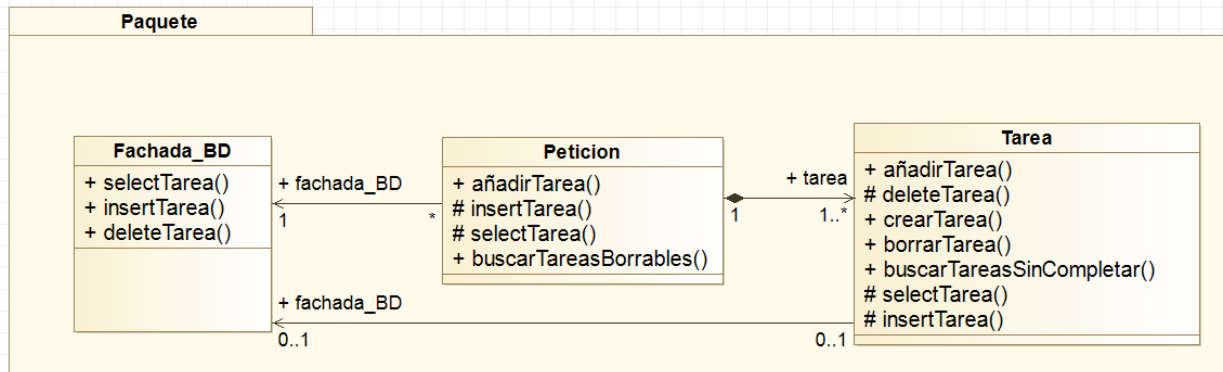
Se ha implementado como una única clase con métodos para seleccionar, insertar y eliminar instancias de cada tabla de la base de datos. Salvo por las clases hijas cada clase está relacionada con una tabla en la base de datos. Las clases con relación de herencia comparten tabla en el modelo relacional.

Como cada método de la fachada está relacionado a una clase, la manera por la que las clases llaman a la fachada es mediante un método con visibilidad protegida en la clase que llama a la fachada. Es decir, las clases tienen métodos protegidos que llaman a los métodos respectivos de la fachada, los cuales son de visibilidad pública.

Los métodos protegidos en las clases se utilizan mediante llamadas por el resto de los métodos de la propia clase, así pueden trabajar con la base de datos sin requerir manejarla directamente.

Por esta solución se aumenta la cohesión relacionando directamente todas las clases con la base de datos. También reduce el acoplamiento ya que existe una clase intermedia entre los métodos llamadores y la propia base de datos. Gracias a este patrón la mantenibilidad se ve beneficiada y debería de significar un aumento en la calidad y rendimiento totales de la aplicación.

Como ejemplo de una aplicación del patrón fachada, este es el diagrama que representa las relaciones con la clase Fachada_BD para gestionar las instancias de clase Tareas. (Por simplificar el diagrama no se han incluido atributos ni parámetros, solo los métodos relacionados y si son públicos, privados o protegidos).



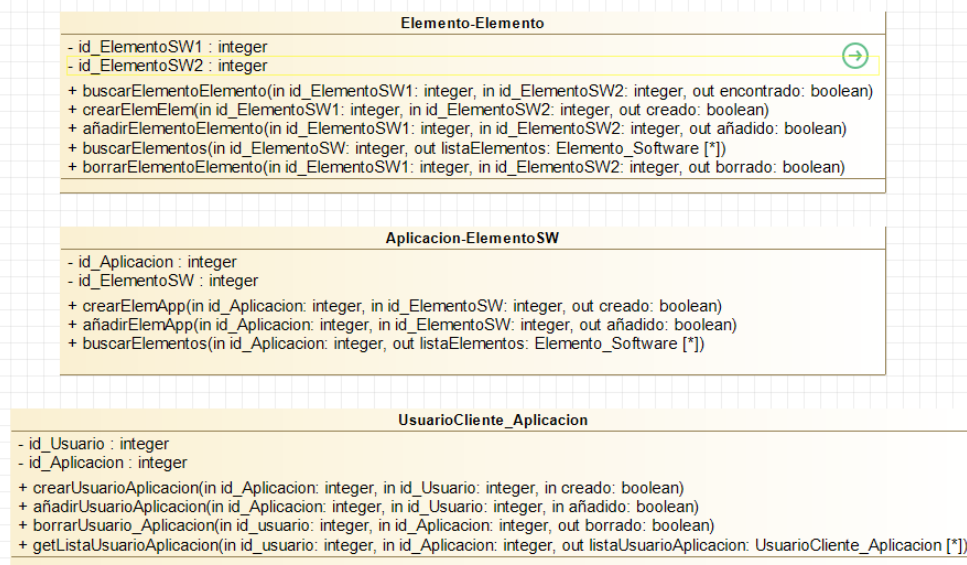
10.2.2. Experto

Existen clases intermedias que gestionan las relaciones entre clases, estas son UsuarioCliente-Aplicacion, Aplicacion-ElementoSW y Elemento-Elemento.

Estas clases asumen la responsabilidad de manejar la información que se transfiere entre las dos clases que se designan en su nombre, separadas por un guion “-”.

Conviene utilizar un patrón para estas relaciones ya que mediante ellas se asignan usuarios a aplicaciones, elementos software a aplicaciones y relacionarlos entre ellos. Estas asignaciones son información clave y dedicada y se debe tratar independientemente de los objetos que relaciona en sí.

Los diagramas para cada una de las clases mencionadas sería la siguiente, solo se relacionaría a las clases mencionadas en el nombre de cada experto:



Cabe mencionar que el uso de expertos modifica los diagramas de colaboración, estos han sido anexados al final del documento para explicar cómo se aplican los patrones expertos.

10.2.3. Creador

Existen funciones creadoras de objetos ajenas a las clases y no aparecen en el diagrama de clases. Particularmente para crear usuarios, proyectos, aplicaciones y peticiones.

Estas se utilizan mediante la interfaz de la aplicación web a la hora de inscribirse como usuario, donde genera los objetos relacionados a la cuenta y al cliente o empleado. También se refiere a las clases que utiliza el sistema cuando un usuario crea una aplicación, petición o proyecto, dependiendo de si es un usuario cliente o un empleado de la empresa.

Las tareas y los elementos software son creados por las clases petición y aplicación respectivamente, estas dos clases podrían beneficiarse del uso de un patrón creador.

Para su implementación contiene atributos que toman el valor del identificador más reciente de cada clase que crea, de forma que al crear una instancia no repita el identificador y aumente automáticamente. Su diagrama sería el siguiente:

