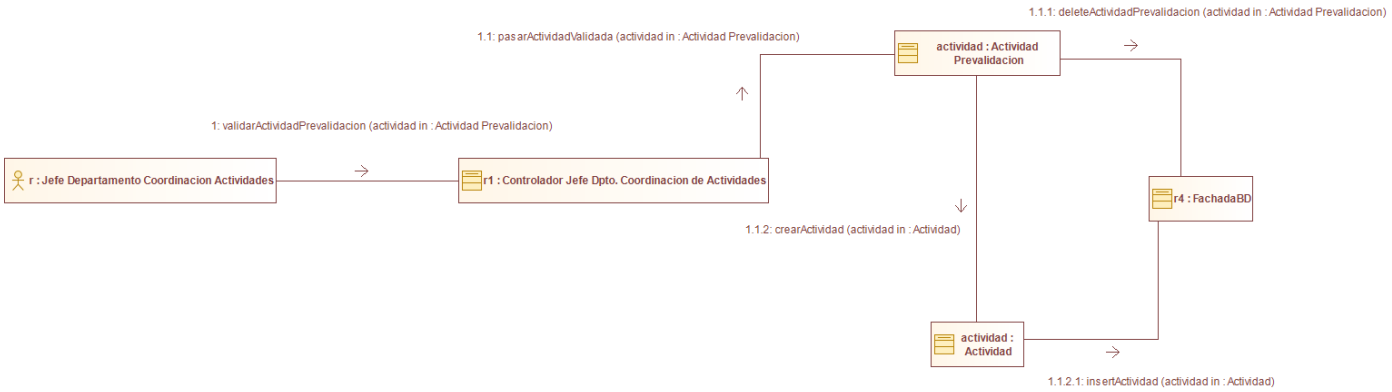
Contenido

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.>

- **Definición:** el Modelo-Vista-Controlador es un patrón de arquitectura de software que separa los datos, la lógica de negocio y la interfaz que se le presenta al usuario. Para ello propone la construcción de componentes: el modelo, la vista y el controlador.
  De esta manera, se consigue un bajo acoplamiento debido a la independencia interfaz-dominio y una gran cohesión porque los controladores solo se encargan de gestionar eventos que recibe, atenderlas y procesarlas. Mediante el controlador, se comunican el modelo y la vista.
- **Modelo:** es el componente que se encarga de manipular, gestionar y actualizar los datos. Contiene una representación de los datos que maneja el sistema y su lógica de negocio. En nuestro caso, FachadaBD va a ser una clase que siempre aparecerá cuando se modifiquen datos en la base de datos. Además, también deberán estar las clases que deban ser instanciadas. Por ejemplo, al añadir un preso deberá aparecer las clases que deben aparecer son Preso y FachadaBD.
- **Vista:** es la interfaz del usuario que se compone de la información mostrada y de los mecanismos que interaccionan con el usuario. En nuestro caso, siempre va a existir el actor que vaya a utilizar el caso de uso correspondiente.
- **Controlador:** como ya hemos mencionado se encarga de comunicar al modelo y a la vista. En nuestro caso hemos creado únicamente los controladores de los actores involucrados en los casos de uso realizados en el artefacto 7.
- **En nuestro caso:** como podemos observar en el diagrama de clases, este patrón de arquitectura ha sido seguido para realizar todos los casos de uso. Por ejemplo, en el CU6-Validar una actividad:



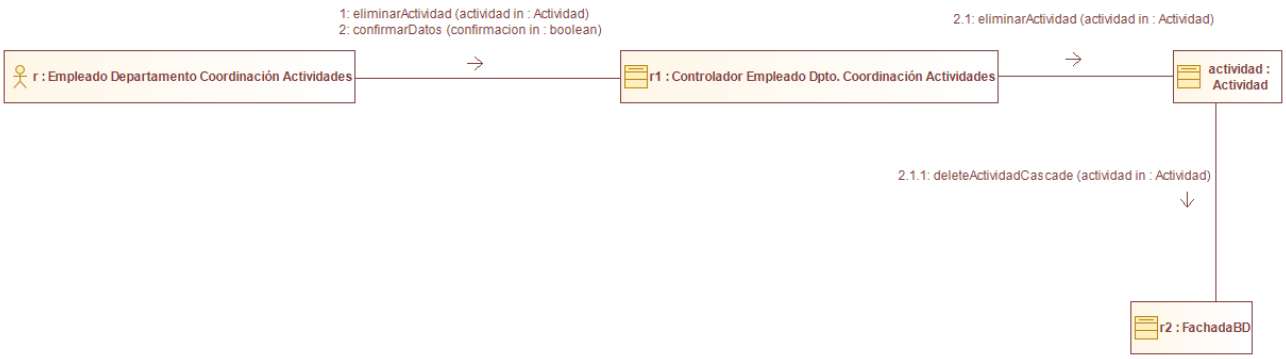
En este caso de uso el controlador es el jefe del departamento de coordinación de actividades, el modelo estaría formado por Actividad Prevalidación, Actividad y FachadaBD, y el actor del jefe del departamento de coordinación de actividades estaría representando los eventos realizados por el usuario que los hace a través de la vista.

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.>

10.2.1. Patrón Fachada

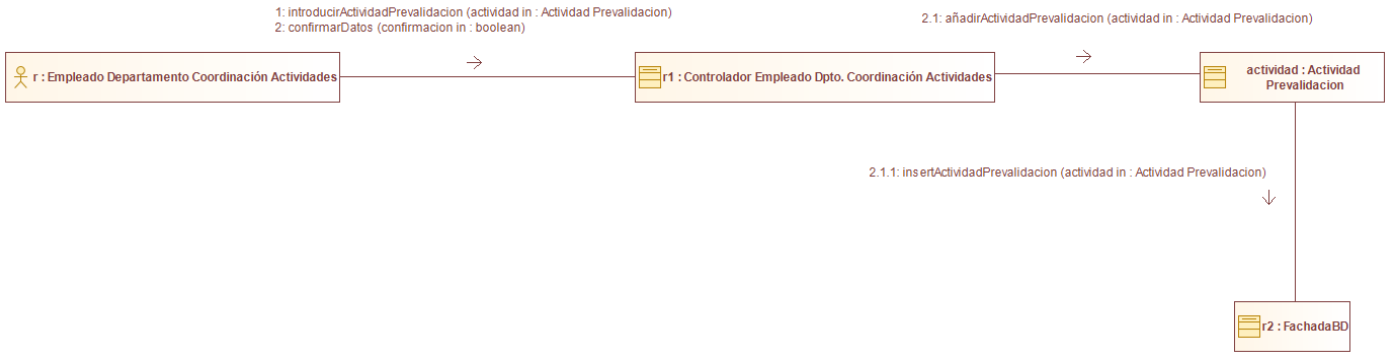
- **Definición:** es un tipo de patrón de diseño estructural, que nos hace simplificar la comunicación entre una clase y otra clase compleja u otro subsistema de clases. Este patrón tiene una clase cliente que quiere utilizar a una clase objetivo para determinadas operaciones, pero la clase destino es muy compleja, por lo que se creará una fachada que simplifique la interfaz de la clase destino adaptándola y simplificándola para un uso concreto. Además, el patrón fachada permite el desacoplamiento entre clientes y sistemas complejos y subsistemas, permitiendo crear una interfaz de uso de varios sistemas para un fin concreto mediante una única interfaz.
- **En nuestro caso:** hemos utilizado este patrón para la base de datos que tendrá el sistema. El motivo de creación es que todas las operaciones para el contacto con la base de datos aparezcan en esta clase Fachada, y no en las clases de los objetos. Hemos creado una clase, FachadaBD que proporcione la funcionalidad de manera sencilla a nuestro cliente, en este caso el empleado del departamento de coordinación de actividades.
  El diagrama es el siguiente:



Donde actividad es una instancia de la clase actividad que hace uso de la base de datos mediante la fachada. Por otro lado, FachadaBD trata de ofrecer la funcionalidad que demanda el cliente mediante una interfaz sencilla donde, internamente, utiliza las clases complejas

10.2.2. Patrón Creador

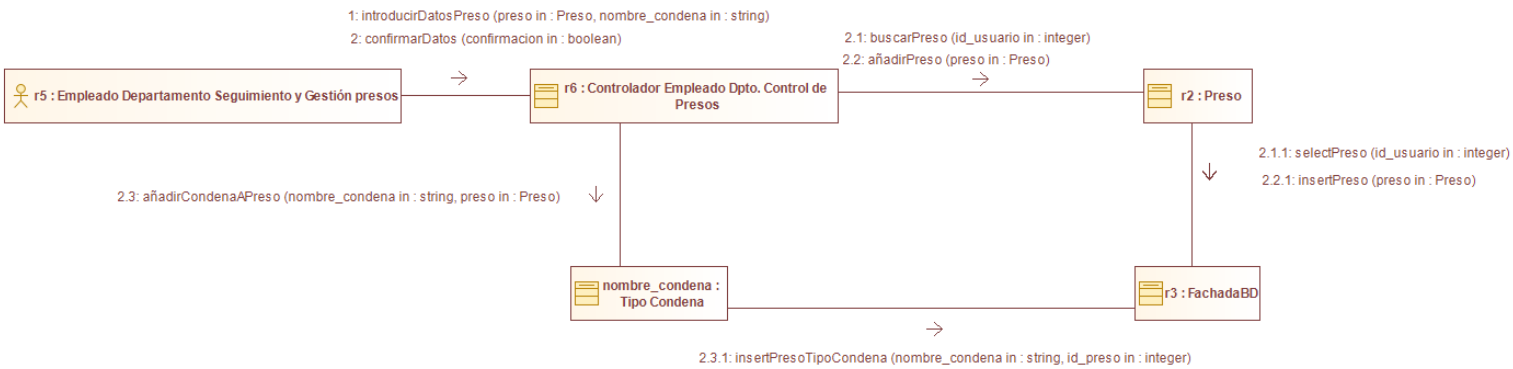
- **Definición:** el patrón creador nos ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización.
- **En nuestro caso:** hemos utilizado este patrón para la creación de nuevos objetos de las clases Preso y Tipo Condena al introducir un nuevo preso en la prisión.
  El diagrama es el siguiente:



Donde Controlador empleado del departamento de coordinación de actividades representa al sistema que tiene la responsabilidad de crear una instancia de la clase Actividad Prevalidación.

10.2.3. Patrón Experto

- **Definición:** es el principio básico de asignación de responsabilidades. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).
- **En nuestro caso:** en general todas las clases utilizadas en los diagramas manejan la información correspondiente a su clase. Como ejemplo utilizaremos el siguiente diagrama:



Como podemos observar, cada clase tiene sus métodos correspondientes a cada objeto creado. Por ejemplo, en la clase Preso, tiene los métodos correspondientes a: buscar si el preso existe e insertarlo en la base de datos en caso de que no exista.