Practica AD

Daniel Castellote y Alejandro López

Base de Datos Relacional.

Explicación diagramas:

La práctica se compone por 6 Entidades principales Departamento, Programadores, Proyectos, Repositorios, Commits e Issues.

Vamos explicar brevemente las relaciones que hemos decidido tomar entre ellas,

Primeramente para el Departamento hemos asociado un Proyecto con cardinalidad uno a muchos un departamento puede tener 1 o varios Proyectos pero solo puede haber un Proyecto por Departamento, Aparte de esto hemos decidido añadir una composición ya que un departamento no puede existir sin proyectos asignados

Después para la relación entre el departamento y el programador hemos decidido que pueden haber varios programadores en un mismo departamento pero que un programador solo puede estar en un único departamento aparte de esto añadimos la asociación entre las dos entidades. Para salvar claves externas hemos decidido crear una entidad aparte llamada jefe de departamento que contendrá la clave externa de departamento y que a su vez estará asociado a un programador siendo no un programador pero sí jefe del departamento.

Para la relación entre los proyectos y los programadores ya que un programador puede tener asociados 1 o muchos proyectos y un proyecto puede estar compuesto por 1 o muchos programadores ya que es una relación muchos a muchos hemos decidido cortarla para ello hemos creado una entidad externa llamada participación esta participación lo que hará es asociar una participación a un programador y una para un proyecto, Aparte de ello hemos creado una entidad que es jefe de proyecto este tendrá las características de un programador ya que formará parte del proyecto pero no podrá trabajar en él por lo que hemos decidido que el proyecto no podrá existir sin un jefe de proyecto ya que ahí le hemos añadido una composición.

La clase del repositorio estará asociada a un proyecto esta es una relación 1 a 1 ya que un proyecto solo está asociado a un repositorio y un repositorio a un proyecto.

Por último decir que los repositorios estarán compuestos por Commits e Issues y que este repositorio no puede asistir si no existen estos, a su vez un programador será el que realiza los cómmits e Issues, para las Issues ya que existe una relación 1 a muchos y a que un programador podría crear muchas Issues y una Issue puede ser creada por muchos programadores creamos la clase externa creación para asignar una única creación a un programador.

Para llevar esto acabo de forma práctica tendremos que crear Modelos, que son las clases donde guardaremos la información que necesitemos sobre, por ejemplo las entidades principales, ya que para poder operar con la información en nuestro programa necesitaremos una clase con los atributos correspondientes, un claro ejemplo es la clase creada Departamento, en esa clase se puede observar atributos como idDepartamento, nombre, presupuesto…, son atributos muy importantes porque serán los campos que rellenaremos con la información y posteriormente serán una parte de las tablas de las bases de datos.

En cuanto a los DTO son clases paralelas a los Modelos tendrán los mismos atributos que sus correspondientes clases(Departamento=DepartamentoDTO), pero los DTO como su nombre indica son Data Transfer Object , es decir, son objetos que sirve para transportar datos y lo más importante estos objetos pueden recibir información de diferentes tablas de una base de datos y crear un solo objeto con ello.

Entonces para poder proseguir con el proyecto hemos tenido que después de crear todos los Modelos que hemos creído necesarios, crear sus respectivos DTO para que cuando nos conectemos a la base de datos podamos interactuar con ella de forma rápida. También necesitaremos una clase Mapper, las clases Mapper como su nombre indica son para mapear objetos por lo que tendremos generalmente dos métodos que son fromDTO y toDTO con estos dos métodos podremos mover la información de un lado a otro sin ningún problema ya que dará igual si me entregan un DTO con la información cargada, si yo quiero pasar esos datos a otro objeto podre gracias a estos métodos.

En cuanto a los servicios son clases que tendremos que crear cuando tengamos los datos y los queramos filtrar y enviarlos a una base de datos, para ellos necesitaremos nuestra clase Mapper, la clase Mapper correspondiente al Servicio que estés creando(si es DepartService utilizaras el DepartMapper), tendremos métodos como por ejemplo getAllDepartamentos(), con ese método lo que estaremos haciendo es mappear a DTO y filtrarlo buscando todos los departamentos , en el servicio es donde tendremos que implementar la bi direccionalidad para poder acceder con un mismo servicio a diferentes tablas de información.

Los Repositorios son otra clase en la cual tendremos que hacer las consultas a la base de datos y preparar los métodos para sacar la información filtrando de la forma que nosotros queramos como por ejemplo con el método findALL() y la sentencia “SELECT * FROM departamento” que seleccionaríamos todo de la tabla departamento, habría que implementar los demás métodos CRUD para poder acceder a cualquier información que requiramos de la base de datos de forma automatizada ya que tienes el método con la sentencia correcta para la filtración de los datos.

Aparte del findAll() tendremos todos los métodos para las operaciones básicas CRUD las cuales, gracias al controlador donde creamos métodos de salida en formato Json para todas las sentencias CRUD podremos visualizar el resultado de las operación en Json.

Por último hemos creado unos sencillos test de Junit5 los cuales cargamos con valores un objeto de la clase a testear y lanzamos consultas para recibir un resultado, este resultado lo testearemos con AssertEquals para ver si es el esperado y realiza la sentencia como queremos.

Por ultimo para informarte de cómo hemos creado nuestra propia BBDD, hemos creado un dockerfile que nos genera una base de datos Mariabd, además asociamos al servicio mariaDb el adminer para visualizar las tablas creadas a mano en los ficheros .sql.