DAO EntityManager

Comenzamos con **IConfiguration**, interfaz que define los métodos a implementar en Configuration, que son getUser(), getPassword() y getUrl().

Configuration es un singleton que implementa la interfaz IConfiguration, instancia IConfiguration y la inicializa a null. Obtiene de ella los métodos y los declara de manera que devuelvan las variables de entorno. El método getConfiguration() devolverá un nuevo objeto (al ser singleton, sólo podrá haber una instancia de éste) de esta misma clase si configuration sigue siendo null.

La interfaz **IRunnables** define dos métodos que serán implementados por Runnables, estos son getSQL() y run(final PreparedStatement statement) que ejecutará un PreparedStatement.

La clase **Runnables** implementa la interfaz anterior y declara los métodos, getSQL(), que retornará un String que será recibido en el constructor de esta clase, el cual recibe una sentencia sql (String), una entidad genérica(T) y un Statement genérico (Statement<T>). Y otro método run(PreparedStatement statement) que llamará al método run(PreparedStatement statement, T entity) que se alberga en la interfaz **Statement**. Este método puede ser llamado ya que hemos definido en Runnables como parámetro Statement<T> statement; además de String sql y T entity. Son generics ya que podrán recibir cualquier objeto, ya sea pizza, ingrediente, etc.

Statement es una interfaz funcional con un método void run(PreparedStatement statement, T entity) que ejecuta un preparedStatement con una entidad T(genérica), con los que completa la query a través de una función lambda.

ResultSetForLambda es una interfaz funcional también con un método void run(ResultSet resultSet, T entity) encargada de insertar los datos obtenidos a través del resultSet. Es diferente a la anterior porque ésta será usada en el select() y Statement será usada en el save(), ya que los parámetros necesarios para escritura o lectura en una base de datos son diferentes.

IEntityManager contiene los métodos que implementará EntityManager.

EntityManager es una fluent interface ya que los retornos this de algunos métodos son utilizados por el siguiente método. Esta clase declara los métodos que implementa de IEntityManager, estos métodos hacen de conexión entre la entidad a utilizar y la creación de los Runnables. Uno se usará para un runnable singular y otro para varios o una lista. Los métodos save() y select() realizan la conexión a la base de datos utilizando la configuración que obtiene de Configuration, lo hacen a través del método buildConfiguration() y guarda sus datos en un atributo de tipo Configuration a través de su constructor. Save() será también el encargado de hacer los commits pertinentes y rollbacks, cerrará la conexión y ejecutará las sentencias DDL que reciba a través de los runnables mediante executeUpdate(). Cuando acabe también limpiará la lista de runnables. A continuación explico algunos de los métodos:

- ➔ addStatement: Recibe una entidad genérica <T>, un string (sentencia sql) y un Statement<T> (también genérico) que guardará en un objeto IRunnables que será añadido a la lista de IRunnables que tiene EntityManager como atributo. Se devuelve a sí mismo para poder ser utilizado en el siguiente método (fluent

interface), ya sea save() o select(). Hará uso de una función lambda para completar el statement, la lambda utilizará un PreparedStatement y una entidad para esto.

- ➔ addRangeStatement: Igual que el método anterior pero para una lista de runnables mayor a 1, hará uso de un for para crearlos y añadirlos todos.
- ➔ save: Se encarga de conectar con la base de datos y ejecutar los runnables que haya en la lista. Al terminar cierra la conexión y limpia la lista de runnables. A diferencia de select(), utiliza un executeUpdate() ya que save se usará para escritura en la base de datos y select() para lectura, el cual hará uso de executeQuery().
- ➔ select: Conecta con la base de datos y devuelve un genérico de tipo Entity a través de un SELECT a la base de datos con el que obtiene un ResultSet. Instancia Class<T> clazz, (la cual recibe como parámetro al igual que una interfaz funcional llamada ResultSetForLambda que utilizaremos en la llamada al método en una función lambda). También limpiará la lista de runnables y cerrará la conexión cuando acabe.

En el paquete **domain** encontraremos Entity, Ingredient, Pizza, IDAO y PizzaDAO. De Entity heredan Ingredient y Pizza, a través de la que obtendrán su atributo id. IDAO es una interfaz genérica que contiene los métodos necesarios para el tratamiento de los datos de la base de datos, esta será implementada por PizzaDAO, donde se escribirán los cuerpos de los métodos. De esta manera, en la clase App sólo tendremos que instanciar PizzaDAO y a través de esta llamar a los métodos insert, update, delete, etc, ya que dichos métodos están desarrollados ya en esta clase.