Pizza DAO

Configuration.java:

- Interfaz que se encarga de definir las variables para el usuario, contraseña, y url necesarios para realizar la conexión con la base de datos.

ConfigurationImp.java:

- Clase que utiliza el patrón de diseño Singleton, y que desarrolla las variables obtenidas al implementar la interfaz “Configuration.java”.

Statement.java:

- Interfaz funcional que obtiene por parámetros un PreparedStatement y una entidad genérica. Puede lanzar una Excepción SQL.
- Se encarga de obtener los datos de la entidad genérica e insertarlos en el PreparedStatement para completar su query SQL.

Resultset.java:

- Interfaz funcional que obtiene por parámetros un ResultSet y una entidad genérica.
- Se encarga de insertar los datos obtenidos del ResultSet a la entidad genérica dándole cuerpo al método a través de Lambda.

Runables.java:

- Interfaz que define un método que se encarga de obtener querys SQL, y otro para ejecutar un PreparedStatement obteniendo a través de su parámetro.

RunablesImp.java:

- Clase que implementa la interfaz “Runables.java”, almacena en variables de tipo “final” una query SQL, una entidad genérica (T entity), y una interfaz funcional (Statement<T>). Implementa los métodos obtenidos a través de la interfaz, uno que devuelve el valor de la variable con la query SQL, y otro que ejecuta la interfaz funcional pasandole a esta por parámetros el PreparedStatement obtenido del método de la interfaz y la entidad guardada en la variable final de la clase.

EntityManager.java:

- Interfaz que define 5 métodos:
    - *buildConnection()*.
    - *addStatement()*.
    - *addRangeStatement()*.
    - *select()*.
    - *save()*.

EntityManagerImp.java:

Clase que implementa la interfaz EntityManager.java e implementa y desarrolla sus métodos:

- ***buildConnection:*** Obtiene las variables de configuración de la conexión a la base de datos a través de un objeto Configuration.java
- ***addStatement:*** Crea un Runable con una query SQL y una entidad utilizando Lambda para completar la query SQL según la entidad y añadirlo a una lista de Runables.
- ***addRangeStatement:*** Crear varios Runable con una misma query SQL pasando por parámetro un Iterable de entidades y la query SQL siendo completada con Lambda según la entidad.
- ***select:*** Realiza una conexión con la base de datos y devuelve un Optional<T> obtenido a través de una query SQL de lectura ejecutando el primer Runable de la lista de Runables, y de ahí saca un ResultSet, luego creando una instancia de Class<T> que es pasada por parámetro al método y ejecutando la interfaz funcional Resultset<T> para sacar los datos del ResultSet los mete en la entidad, por último devolvemos un Optional de esta entidad. Al finalizar se cierra la conexión con la base de datos y se limpia la lista de Runables.
- ***save:*** Realiza una conexión con la base de datos y se encarga de ejecutar todos los Runable de la lista de Runables utilizando una transacción. Al finalizar se cierra la conexión con la base de datos y limpia la lista de Runables.