LC2 - P2 – Proyecto grupo 9

Integrantes: Mateo Vivas, Alejo Tassello, Juan Pablo Mulé

Stock ferretería

**Que hace?**

Aplicación back-end en springboot para poder visualizar lista de todos los productos de un comercio en una tabla, la cual especifica un id o código del producto, el nombre del producto, la cantidad en existencia, y el precio unitario por unidad de cada articulo; pudiendo visualizar el listado de todos los productos en el browser , como así también buscar por id individualmente cada uno de ellos.

Al ser C.R.U.D además se puede crear un nuevo producto, modificar uno ya existente como asi también eliminar.

**Clases y Paquetes**

**StockFerreteriaAplication.java:** es el main de la aplicación, aquí indica que es springbootapplication e importa los paquetes automáticamente para facilitar las tareas

**CONTROLLERS/**

**StockController.java:** la clase StockController pertenece al paquete Controllers, su tarea es atender los request HTTP y según el método que reciba analiza el contenido del request y decide a que método llamar y le pasa los parámetros necesarios.

Contiene la anotación @RestController la cual es la combinación de @Controller y @ReponseBody; y la anotación @RequestMapping con la cual nos permite en la URL agregarle luego del localhost8080 la anotación correspondientes para realizar las llamadas de esta aplicación el en browser.

También posee la anotación @Autowired lo que significa que spring va a inyectar en esta clase un bean llamado stockRepository, con la cual extiende CrudRepositori en la interfaz StockRepository; y de esta forma solo declaramos la variable stockRepository de tipo StockRepository y invocando a la anotación @Autowired se configura e inicializa automáticamente para que podamos utilizarla.

Luego están declaradas las anotaciones @PostMapping(“/add”) y (“/delete/{id}”) para agregar y o eliminar un producto, lo cual se realiza a travez del POSTMAN ya que es un Post; como asi también @GetMapping (“/id”) y (“/all”) para poder visualizar un producto especifico o toda la tabla, con su correspondiente declaración de HTML que le da un estilo de visualización. Y al final de la clase un @GetMapping(“”) el cual esta vacion y al no poner nada luego del 8080 en el browser le dará un mensaje de bienvenida al usuario.

**MODELS/**

**Stock.java:** se encuentra la anotación @Data que genera todo el código boiler plate que normalmente se asocia con POJO simples (plain old java objects) y beans como getters para todos los campos y setters para todos los campos no finales e implementaciones apropiadas de toString, equals y hashCode que involucran los campos de la clase y un constructor que inicializa todo los campos no finales, asi como todos los campos no finales isn inicializador que se haya marcado con @NonNull, para garantizar que el campo nunca sea nulo. Esta anotación no tiene nada que ver con el mecanismo de inyección de dependencias, lo que hace es evitarnos la necesidad de programar por nosotros mismos todos estos métodos.

Y también encontramos la anotación @Entity la cual le dice a Hibernate(convertirá los datos entre los tipos utilizados por Java y los definidos por SQL, luego generará sentencias SQL y liberará al desarrollador del manejo manual de los datos, manteniendo la portabilidad entre todos los motores de bases de datos acelerando el tiempo de ejecución) que haga una tabla con esta clase.

Luego se declaran la variable con la anotación @Id para determinar el ID de una entidad es tan simple como poner la anotación @Id sobre la propiedad que sería el ID de la entidad; y la anotación @GenerateValue. Y a continuación las otras variables de nuestro programa

**REPOSITORIES/**

**StockRepository.java:** contiene la anotación @Repository ; indica que StockRepository extiende CrudRepository y es candidata a la inyección de dependencias

**Plan de desarrollo:**

* **Elección del proyecto a desarrollar.**
* **Determinar las variables y armar la tabla que se muestra en el browser.**
* **Desarrollo de base de datos en SQL y utilización de XAMPP y Phpmyadmin para poder correrla adecuadamente.**
* **Desarrollo de la aplicación con Spring Initializr**
* **Creacion de paquetes controller/models/repositories**
* **Pruebas con PostMan**