UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO  
  
 FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

Y TELECOMUNICACIONES



**SISTEMA DE VENTAS USANDO FRAMEWORK SPRINTBOOT**

Estudiante: Carlos Gutierrez Tito

Registro: 218062591

Materia: Software 1

Docente: MARTINEZ CANEDO ROLANDO ANTONIO

FECHA: 15 abril 2024

Santa Cruz de la Sierra – Bolivia

abril 2024

**INTRODUCCION**

Spring Boot es un framework de Java que simplifica el desarrollo de aplicaciones Java, especialmente aplicaciones basadas en Spring, proporcionando un enfoque basado en convenciones y configuraciones por defecto. Aquí hay algunos puntos clave sobre Spring Boot:

**Características Principales**

Facilita la configuración: Spring Boot reduce la cantidad de configuración necesaria para iniciar una aplicación.

Incorpora servidores embebidos: Viene con servidores web embebidos como Tomcat, lo que significa que no necesitas desplegar la aplicación en un servidor externo.

Manejo de dependencias: Utiliza Maven o Gradle para gestionar las dependencias, lo que simplifica la gestión de librerías y versiones.

**JPA (Java Persistence API)**

Es una especificación de Java que define una API para el manejo de bases de datos relacionales en aplicaciones Java. Aquí hay algunos puntos clave sobre JPA:

* Objetos y Entidades: JPA permite mapear objetos Java a entidades de bases de datos relacionales. Estas entidades son clases Java con anotaciones especiales que indican cómo se mapean a las tablas de la base de datos.
* Consultas: Proporciona un lenguaje llamado JPQL (Java Persistence Query Language) para realizar consultas a las entidades.
* Relaciones: Permite definir relaciones entre entidades, como relaciones uno a uno, uno a muchos y muchos a muchos.

**Arquitectura de Spring Boot con JPA**

La arquitectura de una aplicación Spring Boot con JPA típicamente sigue el patrón MVC (Modelo-Vista-Controlador):

Modelo (Model): Representa la capa de datos y la lógica de negocio. En JPA, los modelos son las entidades que representan las tablas de la base de datos.

Vista (View): Representa la capa de presentación, normalmente son las páginas HTML o las plantillas que se muestran al usuario. En Spring Boot, se utilizan motores de plantillas como Thymeleaf o JSP.

Controlador (Controller): Representa la capa de control, que maneja las solicitudes HTTP, interactúa con el modelo y decide qué vista mostrar al usuario. En Spring Boot, los controladores anotados con @Controller manejan esto.

DTO (Data Transfer Object)

Los DTO son objetos simples que transportan datos entre la capa de presentación (Controller) y la capa de servicio (Service) o entre la capa de servicio y la capa de acceso a datos (Repository). Aquí hay algunas características y propósitos de los DTO:

* Encapsulan Datos: Representan una estructura de datos con campos y getters/setters, pero sin lógica de negocio.
* Transferencia de Datos: Se utilizan para transferir datos entre capas de la aplicación, especialmente cuando los datos necesitan ser transformados o adaptados para diferentes capas.
* Reducción de Dependencias: Ayudan a reducir las dependencias entre las capas, ya que los DTOs proporcionan una interfaz más simple y específica para cada capa.

Service Layer

La capa de servicio (Service Layer) es responsable de la lógica de negocio de la aplicación. Algunas características y propósitos de esta capa son:

* Lógica de Negocio: Implementa la lógica de negocio de la aplicación, como validaciones, cálculos y operaciones complejas que no pertenecen a la capa de acceso a datos.
* Transacciones: Maneja las transacciones, asegurando que las operaciones de la base de datos se realicen de manera coherente y atómica.
* Reutilización: Permite la reutilización de lógica de negocio en diferentes partes de la aplicación, evitando la duplicación de código.

Repository Layer

La capa de repositorio (Repository Layer) es responsable de interactuar con la base de datos y realizar operaciones CRUD (Create, Read, Update, Delete). Algunos puntos importantes sobre esta capa son:

* Abstracción de la Base de Datos: Proporciona una interfaz orientada a objetos para interactuar con la base de datos, ocultando los detalles de la implementación específica de JPA.
* Consultas: Define métodos para realizar consultas a la base de datos utilizando JPQL o consultas derivadas de métodos.
* Operaciones CRUD: Proporciona métodos para crear, leer, actualizar y eliminar entidades en la base de datos.

**Ejemplo de Flujo de Trabajo**

1. El cliente hace una solicitud HTTP, como GET /productos.
2. El controlador correspondiente en Spring Boot recibe la solicitud.
3. El controlador llama al servicio (si es necesario) para obtener los datos del modelo.
4. El servicio interactúa con el repositorio JPA para realizar operaciones en la base de datos.
5. El servicio devuelve los datos al controlador.
6. El controlador selecciona la vista (plantilla HTML) adecuada y pasa los datos del modelo a la vista.
7. La vista renderiza la página HTML con los datos recibidos y se envía de vuelta al cliente.

**Ventajas de trabajar con SprintBoot**

* Productividad: Permite a los desarrolladores centrarse en la lógica de la aplicación en lugar de configuraciones tediosas.
* Facilidad de uso: Provee anotaciones y convenciones que simplifican tareas comunes, como la creación de controladores y acceso a bases de datos.
* Integración: Se integra bien con otras tecnologías de Spring, como Spring Data JPA, Spring Security, y más.