EXAMEN SPRING BOOT

Implementar consultas avanzadas y filtros dinámicos mediante **Spring Data JPA Specifications** para obtener reportes detallados de inventario, movimientos y auditoría.

Instrucciones

Desarrolla dentro del mismo proyecto los siguientes componentes:

- Repositorios con soporte de Specifications
- Extiende los repositorios ProductoRepository, MovimientoRepository y/o AuditoriaProductoRepository para implementar JpaSpecificationExecutor.

```
public interface ProductoRepository extends JpaRepository<Producto, Long>,
JpaSpecificationExecutor<Producto> {}
```

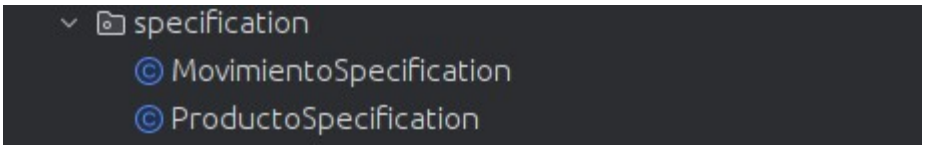
- Clase de especificaciones
- Crea una clase con métodos estáticos que permitan combinar filtros opcionales:
- Ejemplo:

```
public static Specification<Producto> nombreContiene(String nombre) { ... }
public static Specification<Producto> stockEntre(Integer min, Integer max)
{ ... }
public static Specification<Movimiento> tipoEs(TipoMovimiento tipo) { ... }
public static Specification<Movimiento> fechaEntre(LocalDateTime desde,
LocalDateTime hasta) { ... }
```

- Controlador de reportes
- Crea un nuevo controlador:
- GET /api/reportes/inventario
- Parámetros opcionales:
- nombre (String)
- stockMin (Integer)
- stockMax (Integer)
- GET /api/reportes/movimientos
- Parámetros opcionales:
- tipo (ENTRADA, SALIDA, AJUSTE)
- desde (LocalDate)
- hasta (LocalDate)
- productoId (Long)
- Si no se envía ningún parámetro, retorna todos los registros.
- Si se combinan parámetros, aplica los filtros dinámicamente con Specifications.
- Servicio de reportes (ReportService)
- Crea métodos que construyan dinámicamente las especificaciones y devuelvan los resultados filtrados.
- El servicio debe combinar criterios según los parámetros recibidos (sin consultas SQL personalizadas).
- Evidencias
- Captura del endpoint /api/reportes/inventario filtrando por rango de stock.
- Captura del endpoint /api/reportes/movimientos filtrando por tipo y rango de fechas.
- Evidencia de resultados correctos en Postman.

SOLUCION

Lo que hize fue crear un nuevo paquete llamado Specification Para ahi añadir las clases



En el cual empece a crear las clases que indica el examen, empezando con la primera que es ProductoSpecification: En el cual el codigo esta aca abajo.

```
package com.campus.proyecto_springboot.specification;

import com.campus.proyecto_springboot.model.Producto;
import org.springframework.data.jpa.domain.Specification;

public class ProductoSpecification {
    public static Specification<Producto> nombreContiene(String nombre) {
        return (root, query, cb) ->
                nombre == null ? null : cb.like(cb.lower(root.get("nombre")), "%" +
nombre.toLowerCase() + "%");
    }

    public static Specification<Producto> stockEntre(Integer min, Integer max) {
        return (root, query, cb) -> {
            if (min == null && max == null) return null;
            if (min != null && max != null)
                return cb.between(root.get("stock"), min, max);
            if (min != null)
                return cb.greaterThanOrEqualTo(root.get("stock"), min);
            return cb.lessThanOrEqualTo(root.get("stock"), max);
        };
    }
}
```

Seguimos con la siguiente clase de Specification que es MovimientoSpecification

```java
package com.campus.proyecto_springboot.specification;

import com.campus.proyecto_springboot.model.TipoMovimiento;
import org.springframework.data.jpa.domain.Specification;

import java.time.LocalDate;

public class MovimientoSpecification {
    public static Specification<MovimientoSpecification> tipoEs(TipoMovimiento tipo) {
        return (root, query, cb) ->
                tipo == null ? null : cb.equal(root.get("tipo"), tipo);
    }

    public static Specification<MovimientoSpecification> fechaEntre(LocalDate desde,
LocalDate hasta) {
        return (root, query, cb) -> {
            if (desde == null && hasta == null) return null;

            if (desde != null && hasta != null)
                return cb.between(root.get("fechaMovimiento"), desde.atStartOfDay(),
hasta.atTime(23, 59, 59));

            if (desde != null)
                return cb.greaterThanOrEqualTo(root.get("fechaMovimiento"),
desde.atStartOfDay());

            return cb.lessThanOrEqualTo(root.get("fechaMovimiento"), hasta.atTime(23, 59,
59));
        };
    }

    public static Specification<MovimientoSpecification> productoEs(Long productoId) {
        return (root, query, cb) ->
                productoId == null ? null : cb.equal(root.get("producto").get("id"),
productoId);
    }
}
```

Ya digiriendome a clases ya existentes lo que hize fue modificar el ReporteService creandole unos metodos para hacer lo que me pide el examen, Que es hacer consultas avanzadas y filtros dinamicos. Y haciendo lo mismo en el reporteserviceimpl

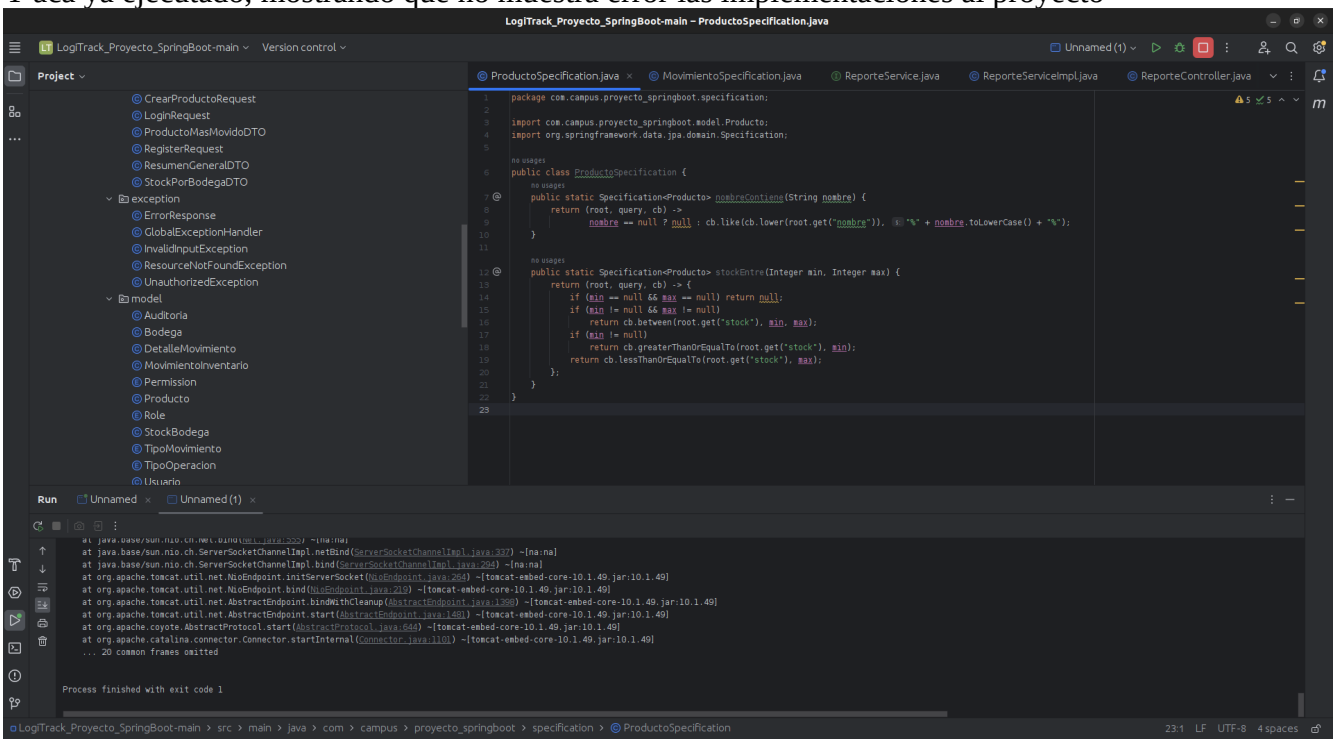
ReporteService

```
    1 usage  1 implementation
    Object Movimientos(TipoMovimiento tipo, LocalDate desde, LocalDate hasta, Long productoId);

    1 usage  1 implementation
    List<Producto> filtrarInventario(String nombre, Integer stockMin, Integer stockMax);
}
```

ReporteServiceImpl

```
    1 usage
    @Override
    public Object Movimientos(TipoMovimiento tipo, LocalDate desde, LocalDate hasta, Long productoId) {
        return null;
    }

    1 usage
    @Override
    public List<Producto> filtrarInventario(String nombre, Integer stockMin, Integer stockMax) {
        return null;
    }
}
```

Y aca ya ejecutado, mostrando que no muestra error las implementaciones al proyecto



Y ya yendo a ReporteController implemente tambien lo debido

```java
package com.campus.proyecto_springboot.controller;

import com.campus.proyecto_springboot.dto.ResumenGeneralDTO;
import com.campus.proyecto_springboot.model.MovimientoInventario;
import com.campus.proyecto_springboot.model.Producto;
import com.campus.proyecto_springboot.model.TipoMovimiento;
import
com.campus.proyecto_springboot.service.MovimientoInventario.MovimientoInventarioServiceImpl;
import com.campus.proyecto_springboot.service.Reporte.ReporteService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.time.LocalDate;
import java.util.List;

@RestController
@RequestMapping("/api/reportes")
@RequiredArgsConstructor

public class ReporteController {

    private final ReporteService reportService;

    // --------------------------
    // INVENTARIO
    // --------------------------
    @GetMapping("/inventario")
    public ResponseEntity<List<Producto>> filtrarInventario(
            @RequestParam(required = false) String nombre,
            @RequestParam(required = false) Integer stockMin,
```

```
            @RequestParam(required = false) Integer stockMax
    ) {
        return ResponseEntity.ok(reportService.filtrarInventario(nombre, stockMin,
stockMax));
    }

    // --------------------------
    // MOVIMIENTOS
    // --------------------------
    @GetMapping("/movimientos")
    public ResponseEntity<List<MovimientoInventarioServiceImpl>> filtrarMovimientos(
            @RequestParam(required = false) TipoMovimiento tipo,
            @RequestParam(required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
LocalDate desde,
            @RequestParam(required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
LocalDate hasta,
            @RequestParam(required = false) Long productoId
    ) {
        return ResponseEntity.ok((List<MovimientoInventarioServiceImpl>)
reportService.Movimientos(tipo, desde, hasta, productoId));
    }
}
```