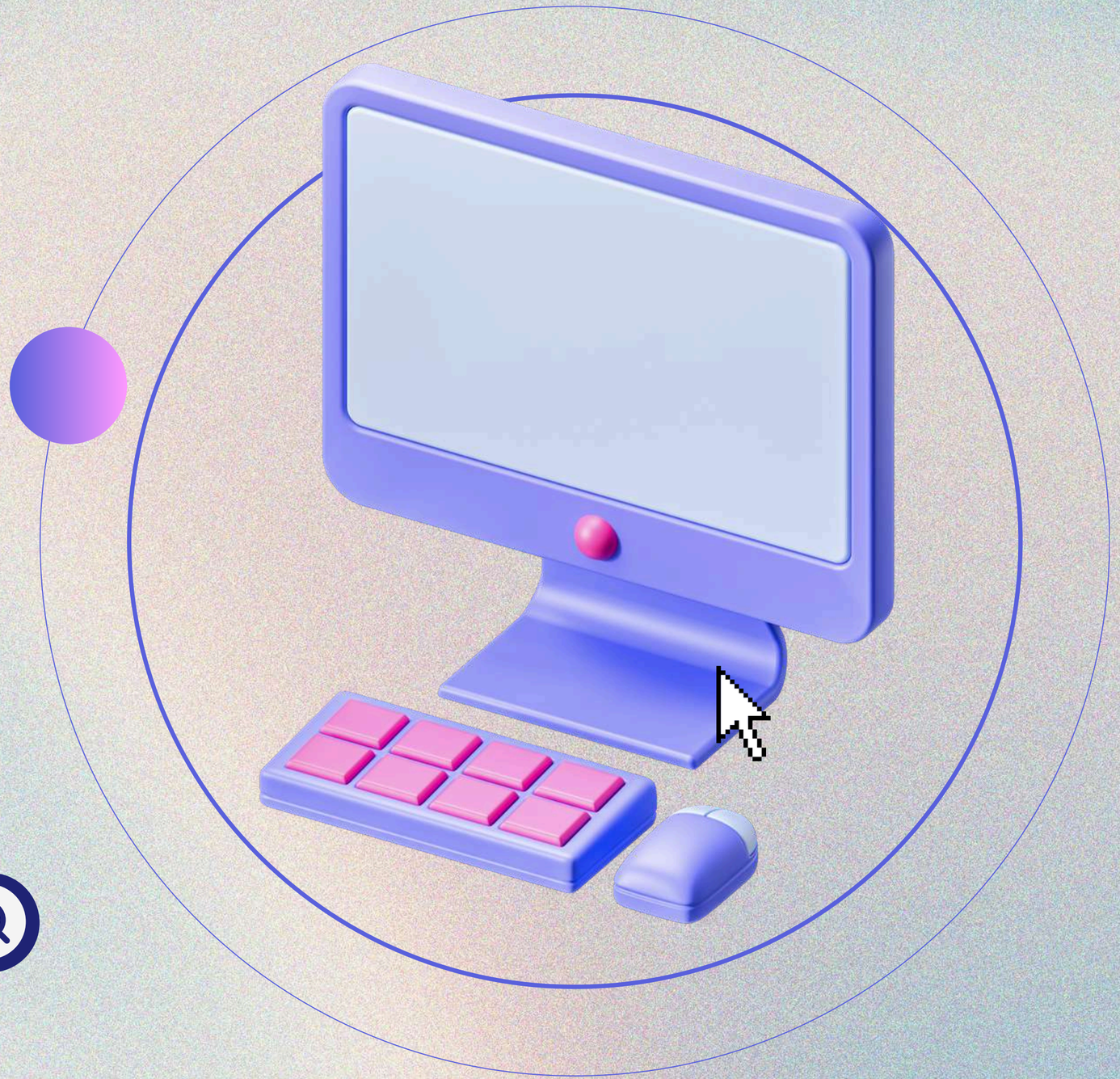
SISTEMA DE CONTROL DE INVENTARIO - SOAP



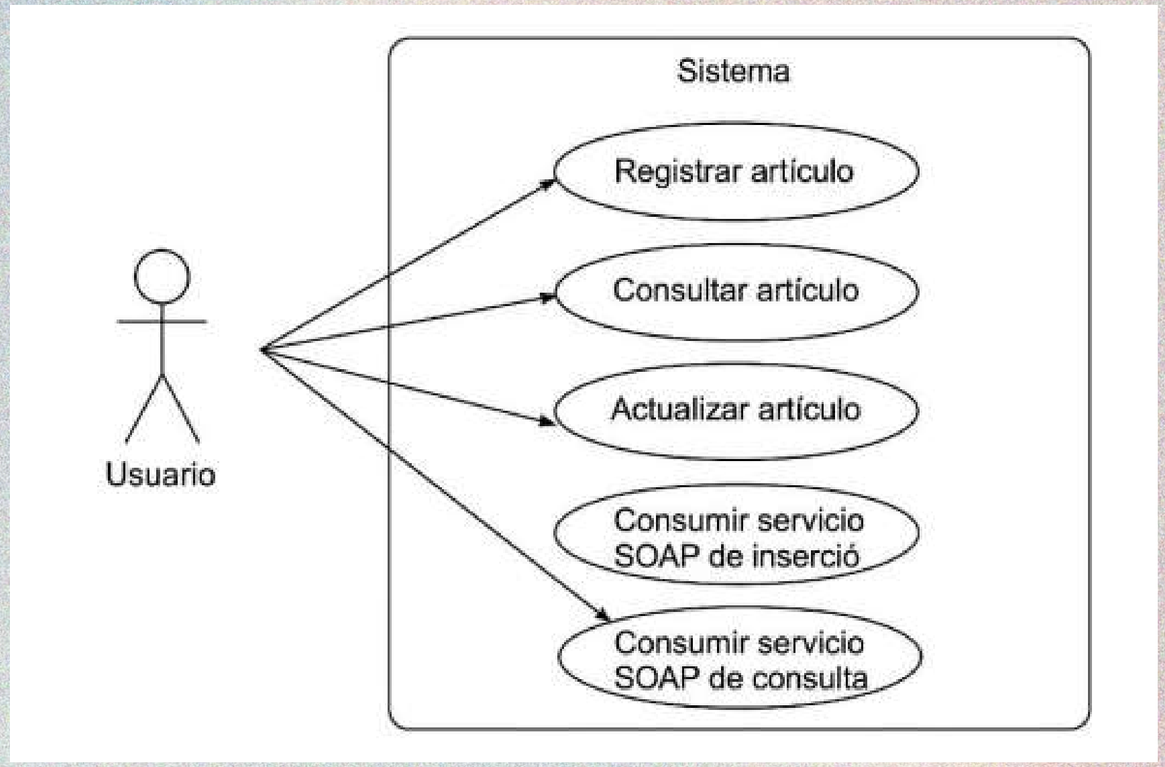
Integrantes : Chafla karem
Gahona Jordan

studio shodwe

OBJETIVO

Desarrollar un sistema de control de inventario basado en arquitectura N-Capas e incorporar una capa de servicios web SOAP que exponga las operaciones de inserción y consulta de artículos, con el fin de asegurar la integridad, disponibilidad y confidencialidad de la información, optimizar y agilizar los procesos de registro, consulta y actualización e implementar validaciones y controles que eviten inconsistencias y mejoren la gestión del stock

DIAGRAMA DE CASOS DE USO

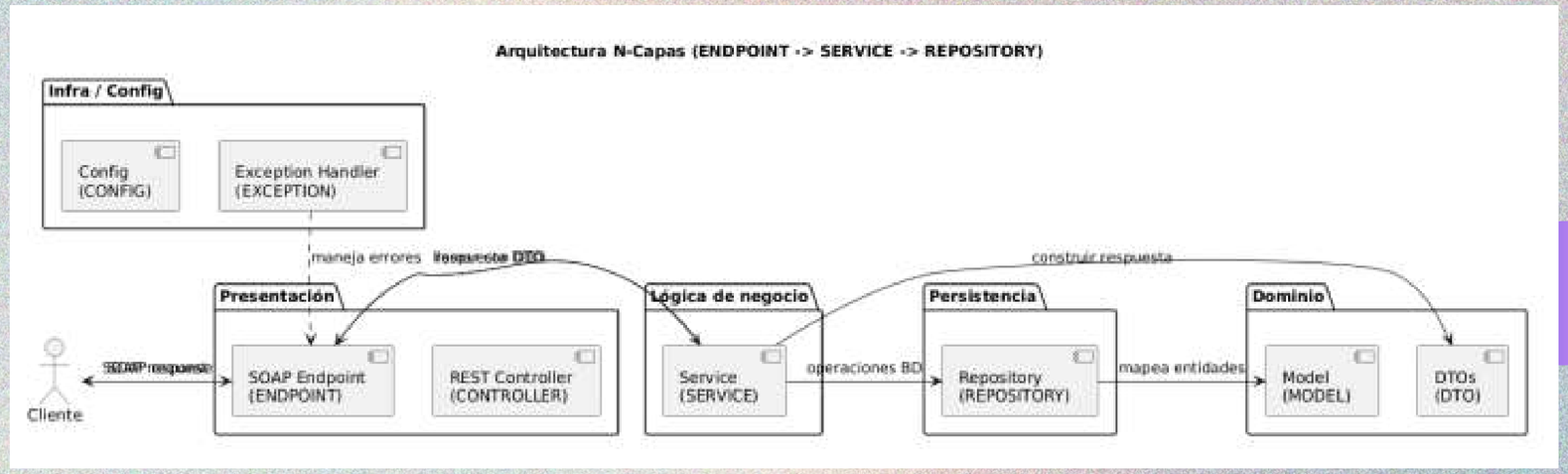


MODELO E/R


Modelo físico (tabla) - ARTICULO
ARTICULO
• id : BIGINT AUTO_INCREMENT «PK»
codigo : VARCHAR(100) NOT NULL «UNIQUE»
nombre : VARCHAR(255) NOT NULL
categoria : VARCHAR(100)
precio_compra : DECIMAL(12,2)
precio_venta : DECIMAL(12,2)
stock : INT DEFAULT 0
stock_minimo : INT DEFAULT 0
proveedor : VARCHAR(255)
created_at : TIMESTAMP DEFAULT CURRENT_TIMESTAMP
updated_at : TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
Restricciones e índices recomendados:
- PK (id)
- UNIQUE(codigo)
- INDEX(nombre)
- INDEX(proveedor)
- INDEX(categoria)

DIAGRAMA DE ARQUITECTURA

El diagrama de arquitectura ilustra la comunicación entre las capas del sistema, mostrando la interacción entre el cliente SOAP, la capa de servicios, la capa de negocio y la base de datos MySQL.



IMPLEMENTACIÓN DE OPERACIONES SOAP

```java
@PayloadRoot(namespace = NAMESPACE_URI, localPart = "InsertarArticuloRequest")  no usages
@ResponsePayload
public InsertarArticuloResponse insertarArticulo(@RequestPayload InsertarArticuloRequest request) {
    try {
        var dto = request.getArticulo();

        ArticuloEntradaDTO entrada = new ArticuloEntradaDTO(
                dto.getCodigo(),
                dto.getNombre(),
                dto.getCategoria(),
                dto.getPrecioCompra(),
                dto.getPrecioVenta(),
                dto.getStock(),
                dto.getStockMinimo(),
                dto.getProveedor()
        );

        articuloService.crearArticulo(entrada);

        InsertarArticuloResponse response = new InsertarArticuloResponse();
        response.setMensaje("Articulo insertado correctamente: " + dto.getCodigo());
        return response;

    } catch (ArticuloException ex) {
        throw new SoapServiceException("Error al insertar articulo: " + ex.getMessage());
```

Función para insertar Datos

IMPLEMENTACIÓN DE OPERACIONES SOAP

```java
@PayloadRoot(namespace = NAMESPACE_URI, localPart = "InsertarArticuloRequest")  no usages
@ResponsePayload
public InsertarArticuloResponse insertarArticulo(@RequestPayload InsertarArticuloRequest request) {
    try {
        var dto = request.getArticulo();

        ArticuloEntradaDTO entrada = new ArticuloEntradaDTO(
                dto.getCodigo(),
                dto.getNombre(),
                dto.getCategoria(),
                dto.getPrecioCompra(),
                dto.getPrecioVenta(),
                dto.getStock(),
                dto.getStockMinimo(),
                dto.getProveedor()
        );

        articuloService.crearArticulo(entrada);

        InsertarArticuloResponse response = new InsertarArticuloResponse();
        response.setMensaje("Articulo insertado correctamente: " + dto.getCodigo());
        return response;

    } catch (ArticuloException ex) {
        throw new SoapServiceException("Error al insertar articulo: " + ex.getMessage());
```

PRUEBAS UNITARIAS

```
@Test
void crearArticulo_success() {
    ArticuloEntradaDTO dto = new ArticuloEntradaDTO(
            codigo: "C001",
            nombre: "Nombre",
            categoria: "Cat",
            new BigDecimal( val: "10.00"),
            new BigDecimal( val: "20.00"),
            stock: 5,
            stockMinimo: 1,
            proveedor: "Prov"
    );

    when(repo.existsByCodigo("C001")).thenReturn( t: false);
    when(repo.save(any(Articulo.class))).thenAnswer( InvocationOnMock inv -> {
        Articulo a = inv.getArgument( i: 0);
        a.setId(1L);
        return a;
    });
```

PRUEBAS

Ingresar los datos por medio de SOAP

Validación de que el precio de compra no debe de ser mayor al precio de venta

CONCLUSIONES

- La arquitectura N-capas está bien aplicada: separación clara entre presentación (endpoints/controllers), lógica de negocio (services), persistencia (repositories) y configuración/infraestructura, lo que facilita mantenibilidad y pruebas.
- La capa SOAP proporciona un contrato estable y versionable (WSDL + XSD), imprescindible para interoperabilidad entre consumidores heterogéneos.
- Mantener al endpoint solo para serializar/deserializar y delegar la lógica al service reduce el acoplamiento y facilita pruebas unitarias.

BIBLIOGRAFIAS

- SOAP 1.2 — W3C Recommendation. Descripción del protocolo SOAP y formato de mensajes. https://www.w3.org/TR/2007/REC-soap12-part1-20070427/ •
- WSDL 1.1 / 2.0 — W3C (WSDL 1.1 es ampliamente usado; WSDL 2.0 es recomendación posterior).

GRACIAS