Universidad de los Andes

Proyecto: Entrega 2

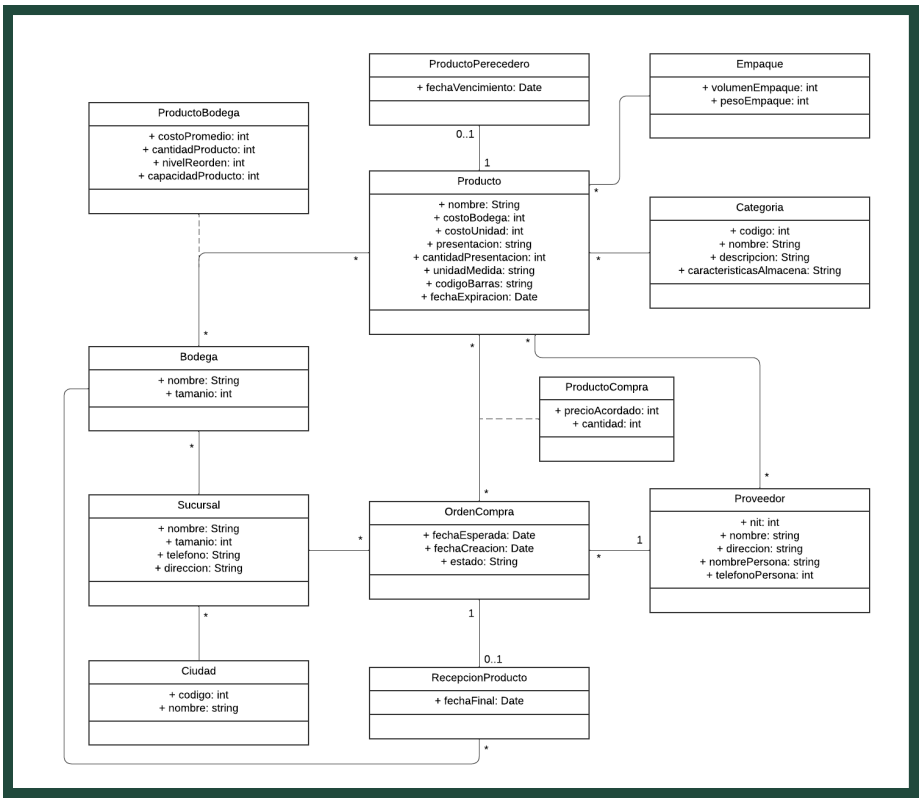
Sistemas Transaccionales 2024-20

Estudiante 1:	Johan Camilo Suarez Sinisterra
Estudiante 2:	Kalia Angelica Gonzalez Jimenez
Estudiante 3:	Juan Miguel Delgado

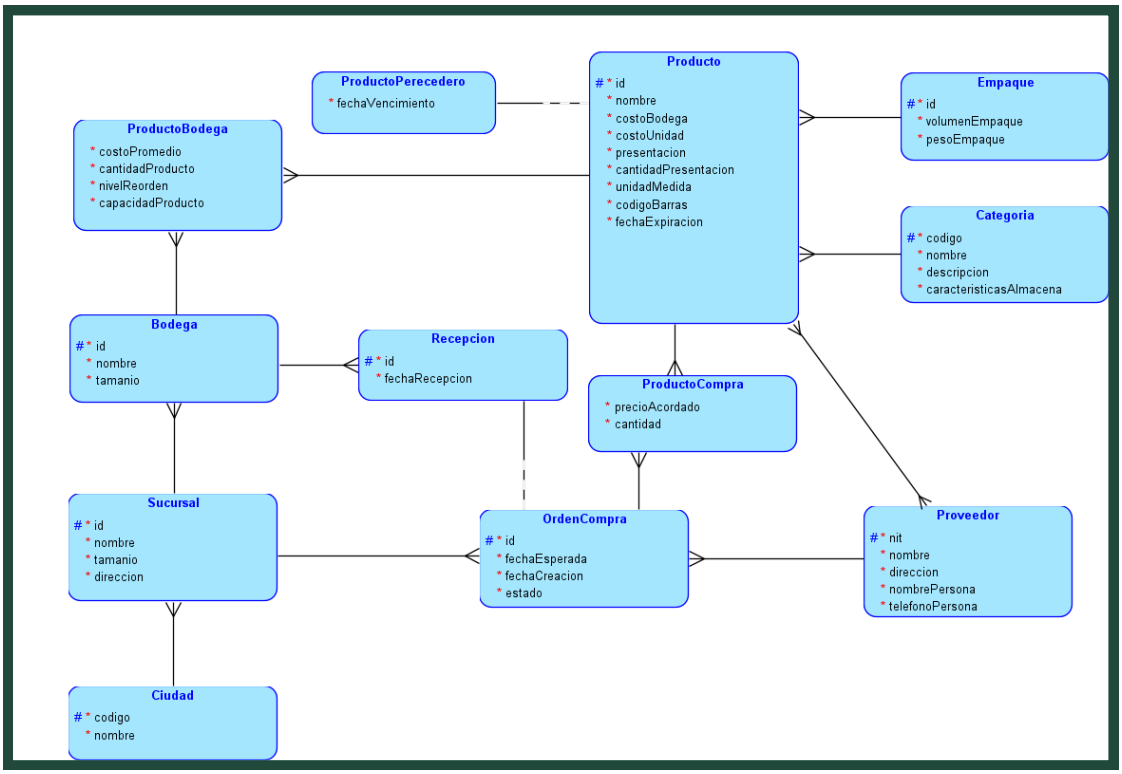
Contenido

Modelo de Datos

Modelo UML: Modificado



Modelo E/R: Modificado



Modelo Relacional: Modificado

Link del Excel con el modelo relacional:

ModeloRelacional.xlsx

Modificación Modelo

Para esta entrega, se necesitaba crear un documento que guardara toda la información del ingreso (recepción en nuestro caso) de los productos que llegaran a una sucursal luego de ser completada la orden de compra. Para esto, ya teníamos una tabla llamada Recepción que tenia la lógica de esta nueva funcionalidad, sin embargo, le realizamos cambios a la entidad para que se pudiera modelar de manera adecuada con la base de datos y con lo que se requería. Así quedó la entidad Recepción.java:

```java
@Entity
@Table(name = "recepcion")
public class Recepcion {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "recepcion_sequence_gen")
    @SequenceGenerator(name = "recepcion_sequence_gen", sequenceName = "recepcion_sequence", allocationSize = 1)
    @Column(name = "id")
    private Integer id;

    @Column(name = "fecharecepcion")
    private Date fechaRecepcion;

    @Column(name = "sucursal_nombre")
    private String sucursalNombre;

    @Column(name = "proveedor_nombre")
    private String proveedorNombre;

    @Column(name = "bodega_nombre")
    private String bodegaNombre;

    // FOREIGN KEYS

    @ManyToOne
    @JoinColumn(name = "bodega_id", referencedColumnName = "id")
    private Bodega bodega;

    @OneToOne
    @JoinColumn(name = "ordencompra_id", referencedColumnName = "id")
    private OrdenCompra ordenCompra;

    public Recepcion(Date fecharecepcion, String sucursalNombre, String proveedorNombre, String bodegaNombre){

        this.fechaRecepcion = fecharecepcion;
        this.sucursalNombre = sucursalNombre;
        this.proveedorNombre = proveedorNombre;
        this.bodegaNombre = bodegaNombre;

    }
```

Esta entidad, tiene 3 nuevos atributos, los cuales son: sucursal_nombre, proveedor_nombre y bodega_nombre. Estos son importantes, ya que son requeridos para poder tener un mejor entendimiento de cuando llegaron los productos, a donde llegaron, quien los envió y donde serán almacenados.

Documentación nuevo RF

Para este nuevo requerimiento, se tenia que realizar una transacción en donde se registrara la entrada de distintos productos luego de que la orden de compra ya fuera recibida en la sucursal que la hizo. Para esto, se hizo una nueva clase llamada RecepcionService.java, en donde se maneja la transacción completamente para que en caso de un error, esta pueda hacer rollback sin ningún inconveniente.

Así mismo, se realizaron nuevas funciones en las clases RecepcionRepository.java y RecepcionController.java para manejar cada las consultas y endpoints de este nuevo requerimiento. A continuación se muestran los cambios realizados para el cumplimiento de esta nueva funcionalidad:

```
@Modifying
@Transactional
@Query(value = "INSERT INTO recepcion (id, fecharecepcion, bodega_id, ordencompra_id, sucursal_nombre, proveedor_nombre, bodega_nombre) "+//
               "VALUES (recepcion_sequence.NEXTVAL, :fecharecepcion, :bodega_id, :ordencompra_id, :sucursal_nombre, :proveedor_nombre, :bodega_nombre)", nativeQuery = true)
void insertarRecepcionRF10(@Param("ordencompra_id") int ordencompra_id, @Param("bodega_id") int bodega_id, @Param("fecharecepcion") Date fecharecepcion, @Param("sucursal_nombre") String sucursal_nombre, @Param("proveedor_nombre") Str
```

Estas sentencias se encargan de poder registrar cada uno de los documentos de recepción de productos con toda la información solicitada. En esta, se puede observar que se reciben varios parámetros, el id de la orden de compra y el id de la bodega donde se van a guardar los productos, los cuales son los únicos datos que ingresa el usuario, y el resto de parametros, son los que se piden que se guarden para la posterior consulta del documento.

Luego, se actualiza el costo promedio en bodega, la cantidad en bodega y la orden de compra pasa a estado “Entregada”. Esto se hace con las siguientes sentencias:

```
@Modifying
@Transactional
@Query(value = "UPDATE ProductoBodega SET "+//
               "ProductoBodega.cantidadproducto = ProductoBodega.cantidadproducto + :cantidad "+//
               "ProductoBodega.costopromedio = ROUND(((ProductoBodega.cantidadproducto * ProductoBodega.costopromedio) + (:cantidad*:precioacordado))/ProductoBodega.cantidadproducto + :cantidad) "+//
               "WHERE ProductoBodega.bodega_id = :bodega_id AND ProductoBodega.producto_codigobarras = :producto_codigobarras", nativeQuery = true)
void actualizarProducto(@Param("bodega_id") int bodega_id, @Param("producto_codigobarras") int producto_codigobarras, @Param("cantidad") int cantidad, @Param("precioacordado") int precioacordado);
```

```
@Modifying
@Transactional
@Query(value = "UPDATE OrdenCompra SET estado=:estado WHERE id=:id", nativeQuery = true)
void actualizarOrdenCompra(@Param("id") int id, @Param("estado") String estado);
```

Ya con las sentencias realizadas, lo único que se debe hacer, es conectar el service y el controller adecuadamente para que el requerimiento pueda funcionar correctamente. Esto lo hacemos mediante los siguientes métodos:

```
@Transactional(isolation = Isolation.SERIALIZABLE, rollbackFor = Exception.class)
public void recepcionRF10(int ordencompra_id, int bodega_id){

    try{

        OrdenCompra ordenCompra = ordenCompraRepository.darOrdenCompra(ordencompra_id);
        Date fechaRecepcion = ordenCompra.getFechaEsperada();
        String sucursal_nombre = ordenCompra.getSucursal().getNombre();
        String proveedor_nombre = ordenCompra.getProveedor().getNombre();
        Bodega bodega = bodegaRepository.darBodega(bodega_id);
        String bodega_nombre = bodega.getNombre();


        recepcionRepository.insertarRecepcionRF10(ordencompra_id, bodega_id, fechaRecepcion, sucursal_nombre,proveedor_nombre, bodega_nombre);
        ordenCompraRepository.actualizarOrdenCompra(ordencompra_id, estado:"Entregada");
        Collection<ProductoCompra> productosCompra = recepcionRepository.darInfoProductoCompra(ordenCompra.getId());
        for (ProductoCompra prod : productosCompra) {
            System.out.println("Producto: " + prod.getPk().getProducto_codigobarras().getNombre());
        }

        for(ProductoCompra prod: productosCompra){
            int producto_codigobarras = prod.getPk().getProducto_codigobarras().getCodigoBarras();
            int cantidad = prod.getCantidad();
            int precioacordado = prod.getPrecioAcordado();
            recepcionRepository.actualizarProducto(bodega_id, producto_codigobarras, cantidad, precioacordado);
        }

    }catch (Exception e){
        System.out.println(x:"Error al registrar la recepcion de productos.");
    }
}
```

```
@PostMapping("/recepcion/{ordencompra_id}/{bodega_id}/new/save")
public ResponseEntity<String> guardarRecepcion(@PathVariable("ordencompra_id") int ordencompra_id, @PathVariable("bodega_id") int bodega_id) {
    try {
        recepcionService.recepcionRF10(ordencompra_id, bodega_id);
        return new ResponseEntity<>(body:"Recepción creada exitosamente", HttpStatus.CREATED);
    } catch (Exception e) {
        return new ResponseEntity<>(body:"Error al crear la Recepción", HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Documentación nuevos RFC

- **RFC6 Consulta Documentos de ingreso de Productos a Bodega - SERIALIZABLE**
  - RecepcionRepository: En el repósitorio de la identidad , se tiene elmetodo obtenerRecepcion(idB,idS) cuyos parametros son el id de la bodega y el id de la sucursal. Este retorna una coleccion de recepciones que cumple con lo ingresado.
  - RecepecionService: En el servicio, se tiene el metodo consultarRfc6(idB,idS) donde llava al metodo obtenerRecepcion explicado anteriormente y realiza una pausa de 30000 para mantener el bloqueo y luego proseguir a retornar la coleccion de recepciones despues de dicho tiempo.
  - RecepcionController: En el controlador , se tiene el metodo consultaRfc6(idS,idB) donde llama el metodo del servicio explicado anteriormente y si no aparece error alguno, retorna el nombre de la recepcion que esta en la bodega y sucursal ingresados.
- **RFC7 Consulta Documentos de ingreso de Productos a Bodega - READ COMMITED**
  - RecepcionRepository: En el repósitorio de la identidad , se tiene elmetodo obtenerRecepcion(idB,idS) cuyos parametros son el id de la bodega y el id de la sucursal. Este retorna una coleccion de recepciones que cumple con lo ingresado.
  - RecepecionService: En el servicio, se tiene el metodo consultarRfc7(idB,idS) donde llava al metodo obtenerRecepcion explicado anteriormente y realiza una pausa de 30000 para mantener el bloqueo y luego proseguir a retornar la coleccion de recepciones despues de dicho tiempo.
  - RecepcionController: En el controlador , se tiene el metodo consultaRfc7(idS,idB) donde llama el metodo del servicio explicado anteriormente y si no aparece error alguno, retorna el nombre de la recepcion que esta en la bodega y sucursal ingresados.

Escenarios de prueba

Usuario Oracle

Usuario: ISIS2304A2824020

En este usuario se encuentra la base de datos de nuestra aplicación.