

Facultad de Ingeniería, Universidad Fidélitas

Lenguajes de Base de Datos SC-504

Profesor: José Pablo Rodríguez Ledezma

Empresa Finnk

Estudiantes:

Beverly Acuña Peñaranda  
Steven Alpízar Ramírez  
Nicole Díaz Guzmán  
Alan Sagot Cespedes

III Cuatrimestre, 2023

## Tabla de Contenidos

<b>Objetivos</b> .....	3
Objetivo general: .....	3
Objetivos específicos:.....	3
<b>Alcance del Proyecto</b> .....	3
<b>Contexto</b> .....	4
<b>Lenguaje Java</b> .....	4
<b>Diagrama Relacional</b> .....	6
<b>Historias de Usuario</b> .....	7
<b>Cronograma</b> .....	10
<b>Creación del Usuario</b> .....	13
<b>Creación de las tablas</b> .....	14
<b>Registro de las Tablas</b> .....	16
<b>Procedimientos almacenados</b> .....	19
<b>Vistas</b> .....	28
<b>Funciones</b> .....	31
<b>Paquetes</b> .....	39
<b>Triggers</b> .....	44
<b>Curores</b> .....	46
<b>Diccionario de Datos</b> .....	53
<b>Programación del Proyecto</b> .....	55
<b>Conclusiones</b> .....	83
<b>Recomendaciones</b> .....	83
<b>Repositorio de GitHub</b> .....	85

## **Objetivos**

Objetivo general:

Desarrollar de una mejor manera el rediseño de una plataforma web implementando la creación de su base de datos para una empresa pequeña.

Objetivos específicos:

- Presentar el funcionamiento de la base de datos creada.
- Desarrollar la modernización de la plataforma web.
- Exponer a los lectores el avance que consiguió la empresa.

## **Alcance del Proyecto**

La tienda virtual se encuentra en un proceso de mejora y optimización de sus bases de datos, con el objetivo de lograr una notable mejora en la gestión de sus productos, clientes y empleados. Actualmente, se ha identificado que estas bases de datos no están funcionando de manera óptima, lo que ha generado dificultades en el desempeño de diversas tareas. Por lo tanto, esta iniciativa de mejora busca solventar estas deficiencias y aprovechar al máximo el potencial de la información almacenada en dichas bases de datos. Con relación a la base de datos de productos, se implementarán mejoras significativas para garantizar que la información disponible sea completa, precisa y actualizada. Asimismo, se establecerá una estructura de categorización mejorada que permitirá una navegación más fluida y una búsqueda más eficiente de productos para los clientes. Estas mejoras se traducirán en una experiencia de compra más enriquecedora y completa para los usuarios.

En cuanto a la base de datos de clientes, se realizarán esfuerzos para enriquecer los perfiles existentes. Esto implicará la recopilación de datos demográficos más detallados, historial de compras y opiniones de los clientes. Asimismo, se llevará a cabo una mejora en la base de datos de empleados, donde se mantendrá un registro actualizado de información relevante, como datos de contacto, historial laboral, habilidades y certificaciones. Con estas mejoras en las bases de datos, se espera que tanto los empleados como los clientes de la tienda virtual experimenten una notable mejoría en sus interacciones con el sistema.

## **Contexto**

Se basa en un emprendimiento familiar el cual se dedica a la importación de ropa, joyería y bolsos. Se desea modificar su página con el conocimiento más actual, que este estéticamente bien y que sea sencilla de navegar, con el fin de poder administrar de mejor manera su tienda. Se procederá a rediseñar y elaborar la página con las necesidades y requerimientos de necesita el cliente. La página web actualmente presenta un problema y es en la base de datos, por lo que requiere una elaboración de una base de datos ya que se ocupa llevar el control de las ventas, empleados, usuarios, accesorio e insumos que son importantes.

## **Lenguaje Java**

El lenguaje seleccionado para el desarrollo del presente proyecto seleccionado por el grupo número 6 es Java en el IDL de NetBeans con la versión 18 para lograr implementar una mejora considerable a la página web e implementando una base de datos a la empresa Finnk, mediante el lenguaje seleccionado se podrá lograr cumplir con las

necesidades requeridas por la empresa y asimismo dando un mejor servicio web a los clientes permitiendo realizar compras en línea, además de que Java permite crear una base de datos la cual será utilizada para llevar un control de inventario exacto en un mismo sistema para poder realizar las consulta de disponibilidad de productos y su respectiva cantidad, por medio de la programación en NetBeans permite crear una página que tenga un diseño intuitivo e interactivo para mayor comodidad de los usuarios.

Al momento de decidir optar por un lenguaje de programación y elegir Java es debido a las grandes capacidades que este lenguaje de programación permite llevar a cabo a los desarrolladores, como grupo se tomó en cuenta principalmente por sus herramientas, sencillas a la hora de desarrollar código, sus extensiones como puede ser SpringBoot y demás, sin dejar atrás su portabilidad multiplataforma, esto dando como resultado que permite ejecutarse en múltiples dispositivos y sistemas operativos, manteniendo el código en un repositorio de GitHub creado por el grupo 6 con cada uno de los integrantes, el cual se logra visualizar a lo largo del presente escrito.

## Diagrama Relacional

En la siguiente imagen se representa el diagrama de entidad relación correspondiente al proyecto.

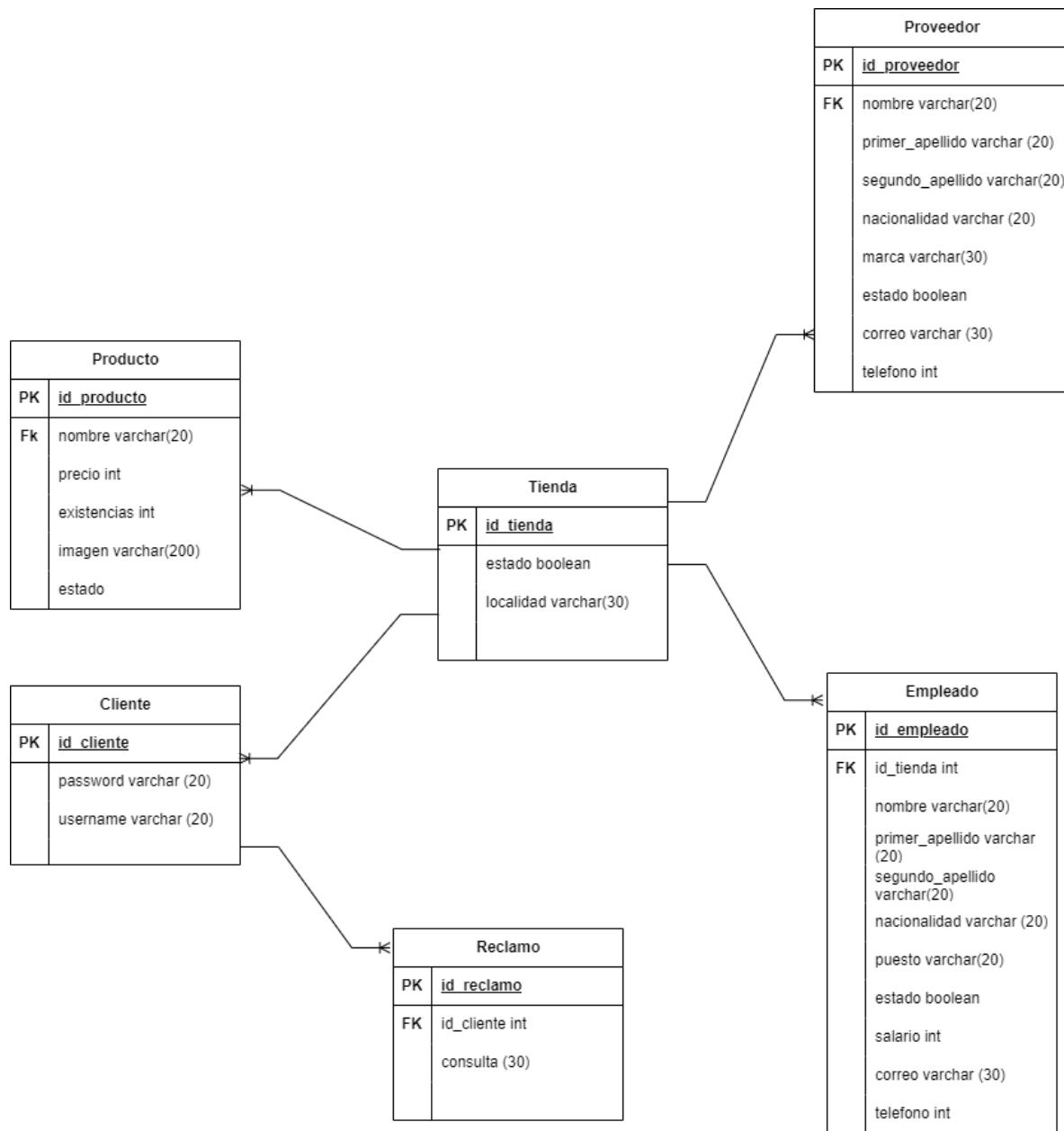


Figura 1. Creación del diagrama relacional.  
Fuente: Propia.

## **Historias de Usuario**

ID de la historia	Rol	Funcionalidad	Resultado
NAA-01	Usuario	Página de logueo y registrarse para tener acceso a la compra de la página.	El usuario ingresa a la página web y deberá registrarse para poder realizar una compra, de lo contrario no podrán realizarla, sin embargo, si podrá ver el sitio web. Se le pedirá un nombre de usuario y contraseña.
NAA-02	Administrador	Página de logueo y registro para tener acceso a la información de la página	El administrador debe ingresar a la página web y deberá registrarse o tener una cuenta logueada o de lo contrario no podrá verificar la información de la página y se le pedirá un username y password.
NAA-03	Administrador	Tener la seguridad básica y necesaria para no tener inconvenientes con respecto a los ataques de seguridad.	El administrador puede verificar el sistema para ver si todo marcha de la mejor manera.
NAA-04	Administrador	El administrador podrá modificar, eliminar y ver antes los productos a la venta.	El administrador entra a la plataforma con el fin de revisar, con el fin de revisar, modificar o eliminar

			los productos existentes.
NAA-05	Usuario	Se requiere una página llamativa y fácil para la navegación de los usuarios en la misma	Para permitir el manejo de una forma mas accesible y sencilla en los usuarios, además de tener acceso rápido en las compras.
NAA-06	Administrador	Diseño de la estructura de la base de datos: Definir y establecer la estructura de la base de datos, incluyendo las tablas, campos y relaciones necesarias para almacenar y organizar la información de manera eficiente.	El resultado esperado sería un diseño completo de la estructura de la base de datos, que incluya todas las tablas necesarias, los campos correspondientes a cada tabla y las relaciones entre ellas, este diseño puede presentarse en forma de diagramas “Entidad-Relación” o utilizando herramientas de modelado de bases de datos.
NAA-07	Usuario	Tener una sección donde el usuario pueda colocar sus consultas o sugerencias de la tienda.	El usuario ingresará a la sección en la página comentar su consulta o sugerencia para la tienda.
NAA-08	Usuario	El usuario podrá adquirir un carrito de compras con los diferentes productos que ha seleccionado.	Al finalizar la selección de sus productos el usuario podrá verificar en el carrito que cosas desea comprar o remover.
NAA-09	Administrador	Capacidad de almacenamiento adecuado: Se debe tener suficiente	El resultado deseado sería que la base de datos tenga suficiente espacio

		espacio de almacenamiento en la base de datos para asegurar que puedan almacenar todos los datos requeridos sin limitaciones o restricciones.	de almacenamiento para acomodar todos los datos requeridos, considerando tanto el tamaño actual como el crecimiento futuro. Esto implica asegurarse de que haya suficiente en el disco disponible y establecer políticas de gestión de almacenamiento adecuadas.
NAA-10	Administrador	Acceso y seguridad de la base de datos: Necesitan tener la capacidad de controlar el acceso a la base de datos, establecer roles y permisos de usuarios y garantizar la seguridad de la información almacenada.	El resultado esperado sería tener un sistema de autenticación y autorización implementado en la base de datos. Se tendrían sus propias credenciales de acceso y se le asignarían roles y permisos específicos según las responsabilidades.
NAA-11	Administrador	Integridad de los datos: Se necesita asegurarse de que los datos almacenados en la base de datos sean precisos, coherentes y confiables, implementando restricciones y validaciones adecuadas para garantizar la integridad de los datos.	El resultado deseado sería que se implementen restricciones y validaciones en la base de datos para garantizar que los datos almacenados sean precisos y consistentes. Esto incluye la definición de restricciones de integridad como los tipos de llaves ya sean primarias o foráneas, y la

			validación de los datos ingresados para asegurar su coherencia y validez.
NAA-12	Administrado	Realización de consultas y generación de informes: Los usuarios deben poder ejecutar consultas y generar informes personalizados en la base de datos para obtener información específica según sus necesidades, facilitando el análisis de datos y la toma de decisiones.	El resultado esperado sería que puedan ejecutar consultas y obtener los resultados deseados de manera eficiente. Esto implica contar consultas optimizadas, índices adecuados y una estructura de datos organizada. Asimismo, se espera poder generar informes personalizados que presenten la información de manera clara y comprensible, ya sea mediante consultas SQL o herramientas de generación de informes.

## Cronograma

Cronograma de trabajo			
Actividad	Encargado	Duración	Estado
Reunión para escoger el grupo de trabajo.	Beverly Acuña Peñaranda, Nicole Díaz Guzmán, Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 2	Completado

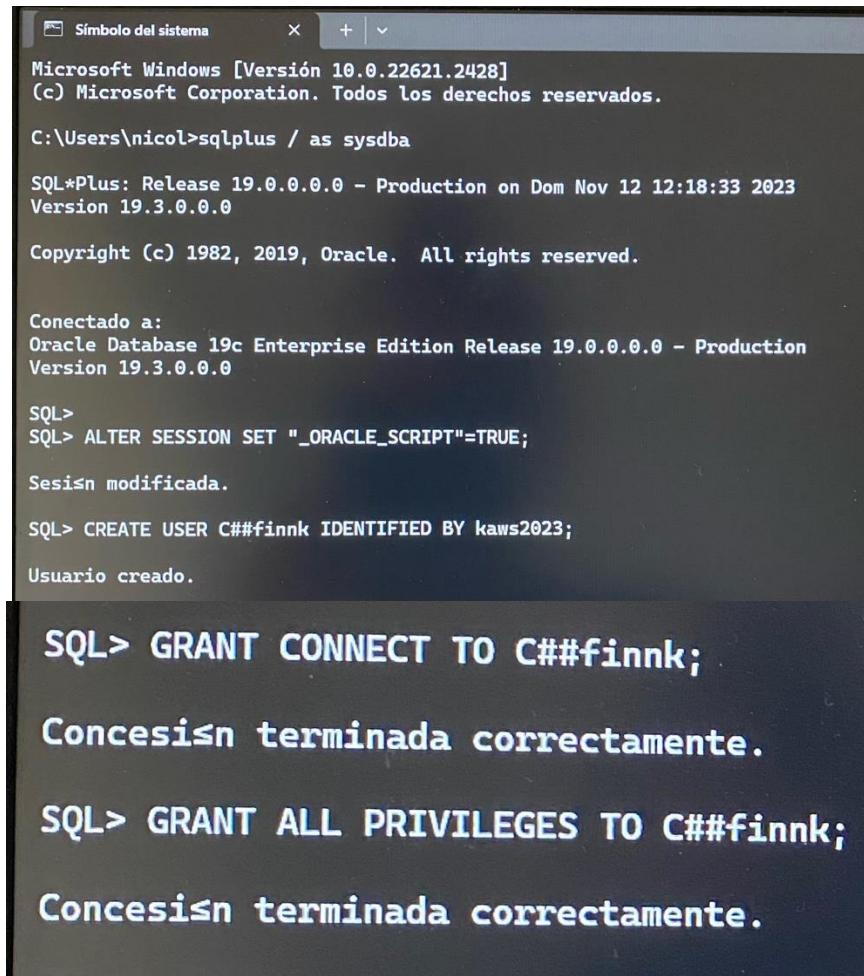
Escogencia tema del proyecto.	Beverly Acuña Peñaranda, Nicole Díaz Guzmán, Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 2	Completado
Propuesta del grupo.	Nicole Díaz Guzmán	Semana 2	Completado
Diseño de página web.	Nicole Díaz Guzmán	Semana 3-5	Completado
Construcción del modelo relacional de la base de datos.	Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 3-4	Completado
Portada	Nicole Díaz Guzmán	Semana 3-5	Completado
1 objetivo principal, 3 objetivos específicos.	Nicole Díaz Guzmán	Semana 3-5	Completado
Alcance.	Nicole Díaz Guzmán	Semana 3-5	Completado
Contexto de la emprendimiento.	Beverly Acuña Peñaranda	Semana 3-5	Completado
Lenguaje de programación que se eligió para conectarse a la base de datos, justificando su decisión.	Steven Alpízar Ramírez	Semana 3-5	Completado
Diagrama Relacional	Beverly Acuña Peñaranda	Semana 3-5	Completado

Requerimientos de usuario.	Beverly Acuña Peñaranda, Alan Sagot Cespedes	Semana 3-5	Completado
Enlace al repositorio de GitHub donde estará el proyecto.	Alan Sagot Cespedes	Semana 3-5	Completado
Mostrar un cronograma de trabajo.	Nicole Díaz Guzmán	Semana 3-5	Completado
Conclusiones y recomendaciones.	Steven Alpízar Ramírez	Semana 3-5	Completado
Conexión a la base de datos con el lenguaje.	Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 5-10	Completado
50% de la programación del proyecto.	Beverly Acuña Peñaranda, Nicole Díaz Guzmán	Semana 5-10	Completado
Todas las tablas de su modelo relacional deben tener un CRUD.	Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 5-10	Completado
La base de datos ya debe tener 25 procedimientos almacenados, 10 vistas, 15 funciones, 10 paquetes, 5 triggers, 15 cursorres.	Beverly Acuña Peñaranda, Nicole Díaz Guzmán, Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 5-10	Completado
Aportes de cada miembro del grupo a GitHub.	Beverly Acuña Peñaranda, Nicole Díaz Guzmán, Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 5-10	Completado

Documento con el diccionario de datos.	Beverly Acuña Peñaranda, Nicole Díaz Guzmán	Semana 5-10	Completado
Defensa final del proyecto.	Beverly Acuña Peñaranda, Nicole Díaz Guzmán, Alan Sagot Cespedes, Steven Alpízar Ramírez	Semana 14	En Proceso

## Creación del Usuario

Se realiza la creación del Usuario C##finnk con su respectiva contraseña kaws2023 y se le asignan los privilegios que se requiere.



```

Símbolo del sistema
Microsoft Windows [Versión 10.0.22621.2428]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\nicol>sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Dom Nov 12 12:18:33 2023
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL>
SQL> ALTER SESSION SET "_ORACLE_SCRIPT"=TRUE;

Sesi&n modificada.

SQL> CREATE USER C##finnk IDENTIFIED BY kaws2023;

Usuario creado.

SQL> GRANT CONNECT TO C##finnk;

Concesi&n terminada correctamente.

SQL> GRANT ALL PRIVILEGES TO C##finnk;

Concesi&n terminada correctamente.

```

Figura 2. Creación de usuario y asignación de privilegios.

Fuente: Propia.

## Creación de las tablas

En este apartado se muestran las tablas que fueron creadas una vez que se unifican las formas normales de las relaciones mencionadas previamente, las mismas se mostrarán en el orden que se encuentran en el script para evitar que se generen problemas al crear la base de datos, así mismo se muestran imágenes de estas tablas en el script para poder ejercitarse aún más la lectura de estos elementos al trabajar en Oracle, además en los mismos se pueden visualizar los tipos de datos e inclusive sus llaves foráneas para la relación.

Tabla Clientes:

```
--Tabla Clientes
CREATE TABLE C##finnk.tab_listado_clientes (
    id_cliente      NUMBER GENERATED BY DEFAULT AS IDENTITY,
    correo_cliente  VARCHAR2(20) NOT NULL,
    CONSTRAINT llavep_cliente PRIMARY KEY ( id_cliente )
);
```

Figura 3. Tabla Clientes.  
Fuente: Propia.

Tabla de Productos:

```
--Tabla Productos
CREATE TABLE C##finnk.tab_catalogo_productos (
    id_catalogo      NUMBER
        GENERATED BY DEFAULT AS IDENTITY,
    imagen_producto  VARCHAR2(200) NOT NULL,
    nombre_producto  VARCHAR2(40) NOT NULL,
    precio_producto  NUMBER NOT NULL,
    existencias_producto NUMBER NOT NULL,
    estado_producto  CHAR(1) NOT NULL CHECK (estado_producto IN ('Y', 'N')),
    CONSTRAINT llavep_catalogo PRIMARY KEY ( id_catalogo )
);
```

Figura 4. Tabla Productos.  
Fuente: Propia.

Tabla de Proveedores:

```
--Tabla proveedores
CREATE TABLE C##finnk.tab_listado_proveedores (
    id_proveedor      NUMBER
        GENERATED BY DEFAULT AS IDENTITY,
    nombre_proveedor  VARCHAR2(40) NOT NULL,
    apellidos_proveedor  VARCHAR2(40) NOT NULL,
    correo_proveedor  VARCHAR2(40) NOT NULL,
    telefono_proveedor  NUMBER NOT NULL,
    marca_proveedor  VARCHAR2(40) NOT NULL,
    nacionalidad_proveedor  VARCHAR2(40) NOT NULL,
    estado_proveedor  CHAR(1) NOT NULL CHECK (estado_proveedor IN ('Y', 'N')),
    CONSTRAINT llavep_proveedor PRIMARY KEY ( id_proveedor )
);

```

Figura 5. Tabla Proveedores.

Fuente: Propia.

### Tabla de Empleados:

```
--Tabla empleados
CREATE TABLE C##finnk.tab_listado_empleados (
    id_empleado      NUMBER
        GENERATED BY DEFAULT AS IDENTITY,
    nombre_empleado  VARCHAR2(40) NOT NULL,
    apellidos_empleado  VARCHAR2(40) NOT NULL,
    correo_empleado  VARCHAR2(40) NOT NULL,
    telefono_empleado  NUMBER NOT NULL,
    salario_empleado  NUMBER NOT NULL,
    puesto_empleado  VARCHAR2(40) NOT NULL,
    nacionalidad_empleado  VARCHAR2(40) NOT NULL,
    estado_empleado  CHAR(1) NOT NULL CHECK (estado_empleado IN ('Y', 'N')),
    fk_tienda NUMBER,
    CONSTRAINT llavep_empleado PRIMARY KEY ( id_empleado ),
    CONSTRAINT fk_tienda_listadot FOREIGN KEY (fk_tienda) REFERENCES C##finnk.tab_listado_tienda(id_tienda)
);

```

Figura 6. Tabla Empleados.

Fuente: Propia.

### Tabla de Tiendas:

```
--Tabla tiendas
CREATE TABLE C##finnk.tab_listado_tienda (
    id_tienda      NUMBER
        GENERATED BY DEFAULT AS IDENTITY,
    localidad_tienda  VARCHAR2(40) NOT NULL,
    estado_tienda  CHAR(1) NOT NULL CHECK (estado_tienda IN ('Y', 'N')),
    CONSTRAINT llavep_tienda PRIMARY KEY ( id_tienda )
);

```

Figura 7. Tabla Tiendas.

Fuente: Propia.

### Tabla de Reclamos:

```
--Tabla Reclamos
CREATE TABLE C##finnk.tab_listado_reclamos (
    id_reclamos      NUMBER
        GENERATED BY DEFAULT AS IDENTITY,
    nombre_reclamo   VARCHAR2(40) NOT NULL,
    comentario_reclamo VARCHAR2(150) NOT NULL,
    fk_reclamos NUMBER,
    CONSTRAINT llavep_reclamos PRIMARY KEY ( id_reclamos ),
    CONSTRAINT fk_reclamos_listador FOREIGN KEY (fk_reclamos) REFERENCES C##finnk.tab_listado_clientes(id_cliente)
);

```

Figura 8. Tabla Reclamos.  
Fuente: Propia.

## Registro de las Tablas

El comando INSERT se utiliza para agregar nuevos registros o filas a una tabla existente en la base de datos creada, a continuación, se presentan los registros realizados para las tablas del proyecto.

INSERT para la tabla de tienda:

```
--Tienda insert
select * from C##finnk.tab_listado_tienda;
BEGIN
    INSERT INTO C##finnk.tab_listado_tienda (id_tienda, localidad_tienda, estado_tienda)
    VALUES (1, 'Tienda Finnk Cartago Metropolis', 'Y');
    INSERT INTO C##finnk.tab_listado_tienda (id_tienda, localidad_tienda, estado_tienda)
    VALUES (2,'Tienda Finnk Cartago Metropolis','Y');
    INSERT INTO C##finnk.tab_listado_tienda (id_tienda, localidad_tienda, estado_tienda)
    VALUES (3,'Tienda Finnk Escazu','Y');
    INSERT INTO C##finnk.tab_listado_tienda (id_tienda, localidad_tienda, estado_tienda)
    VALUES (4,'Tienda Finnk San Jose','Y');
    INSERT INTO C##finnk.tab_listado_tienda (id_tienda, localidad_tienda, estado_tienda)
    VALUES (5,'Tienda Finnk Heredia','Y');
    INSERT INTO C##finnk.tab_listado_tienda (id_tienda, localidad_tienda, estado_tienda)
    VALUES (6,'Tienda Finnk Cartago Centro','N');
    INSERT INTO C##finnk.tab_listado_tienda (id_tienda, localidad_tienda, estado_tienda)
    VALUES (8,'Tienda Finnk Limon', 'X');
END;
```

Figura 9. Insert tabla tiendas.  
Fuente: Propia

INSERT para la tabla de empleados:

```
--Empleados insert
select * from C##finnk.tab_listado_empleados;

BEGIN
  INSERT INTO C##finnk.tab_listado_empleados (id_empleado, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado, sal
VALUES (1, 'luis', 'vargas leiton', 'luivar8899@hotmail.com', '87065480', 'gerente', 'Costarricense', 'Y');

  INSERT INTO C##finnk.tab_listado_empleados (id_empleado, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado, sal
VALUES (2, 'olman', 'vargas vargas', 'olmanfr@gmail.com', '8818823', '375000', 'vendedor', 'Costarricense', 'Y');

  INSERT INTO C##finnk.tab_listado_empleados (id_empleado, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado, sal
VALUES (3, 'maricela', 'vargas robledo', 'mari445ro@outlook.com', '85251321', '520000', 'cajero', 'Costarricense', 'N');

  INSERT INTO C##finnk.tab_listado_empleados (id_empleado, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado, sal
VALUES (4, 'karol', 'vargas robledo', 'mari445ro@outlook.com', '85251321', '520000', 'cajero', 'Costarricense', 'N');

  INSERT INTO C##finnk.tab_listado_empleados (id_empleado, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado, sal
VALUES (5, 'merce', 'vargas robledo', 'mari445ro@outlook.com', '85251321', '520000', 'cajero', 'Costarricense', 'N');

  INSERT INTO C##finnk.tab_listado_empleados (id_empleado, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado, sal
VALUES (6, 'Osvaldo', 'Ovarez Fallas', 'osfallas@outlook.com', '555444', '522000', 'dependiente', 'Costarricense', 'N');

COMMIT;
END;
```

Figura 10. Insert tabla empleado.

Fuente: Propia

### INSERT para la tabla de proveedor:

```
--Proveedor insert
select * from C##finnk.tab_listado_proveedores;

BEGIN
  INSERT INTO C##finnk.tab_listado_proveedores (id_proveedor, nombre_proveedor, apellidos_proveedor, correo_proveedor, telefono_proveedor
VALUES (1,'David','ortega monge','davorte34@hotmail.com','60130495','Bolsos','Costarricense','Y');

  INSERT INTO C##finnk.tab_listado_proveedores (id_proveedor, nombre_proveedor, apellidos_proveedor, correo_proveedor, telefono_proveedor
VALUES (2,'Omar','fernandez fernandez','omarfv@gmail.com','84675931','Joyeria','Costarricense','Y');

  INSERT INTO C##finnk.tab_listado_proveedores (id_proveedor, nombre_proveedor, apellidos_proveedor, correo_proveedor, telefono_proveedor
VALUES (3,'Jefferson','rodriguez monge','Jeffmongel14@outlook.com','72536420','Ropa','Costarricense','N');

  INSERT INTO C##finnk.tab_listado_proveedores (id_proveedor, nombre_proveedor, apellidos_proveedor, correo_proveedor, telefono_proveedor
VALUES (4,'Maria','Jimenes Alvaro','mariajcoj@outlook.com','888888','Joyeria','Costarricense','N');

END;
```

Figura 11. Insert tabla proveedor.

Fuente: Propia

### INSERT para la tabla de productos:

```
--Productos insert
select * from C##finnk.tab_catalogo_productos;

BEGIN
  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (1,'Blusa','https://img.ltwebstatic.com/images3_pi/2021/03/22/1616391124965785dfe2c3bec430704337dd5d5dfef7_thumbnail_900.webp','1')

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (2,'Top halter crop','https://img.ltwebstatic.com/images3_pi/2023/03/20/16772810095557c7b00a3444546e2a10f07cf09bff_thumbnail_900

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (3,'Top hombros descubiertos','https://img.ltwebstatic.com/images3_pi/2022/06/01/1654061340225800040a0ef07ba8c2370464f914_thu

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (4,'Top crop unicolor','https://img.ltwebstatic.com/images3_pi/2023/02/23/1677116808cd87cfe350e272f4c3b8cc1be209411_thumbnail_i

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (5,'Jeans','https://img.ltwebstatic.com/images3_pi/2022/08/01/165932028516100af5c2f7297ea2b174d177300a5_thumbnail_900.webp','1')

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (6,'Destroyed Jeans','https://img.ltwebstatic.com/images3_pi/2021/08/03/162795825317df30b71ad5e72dd22c64b0a8ab37_thumbnail_900

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (7,'Short mscilla','https://img.ltwebstatic.com/images3_pi/2022/06/21/1655780984aeefdddefe3ed44641690b6aa690be0_thumbnail_900

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (8,'Vestido Ajustable','https://img.ltwebstatic.com/images3_pi/2022/04/13/1649814844998c773fd83d5d44c83ff2c5de65fc3a_thumbnail_i

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (9,'Vestido Corto','https://img.ltwebstatic.com/images3_pi/2022/06/21/16558060151577ad59bbe7604e7ff59delcia95557_thumbnail_900

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
VALUES (10,'Bolso','https://img.ltwebstatic.com/images3_pi/2022/08/25/1661414530fe9cb92e5c2bf1cbe6c04ee11c6da_thumbnail_900.webp','1')

  INSERT INTO C##finnk.tab_catalogo_productos (id_catalogo, nombre_producto, imagen_producto, precio_producto, existencias_producto, es
```

Figura 12. Insert tabla productos.

Fuente: Propia

### INSERT para la tabla de clientes:

```
--Clientes insert
select * from C##finnk.tab_listado_clientes;

BEGIN
INSERT INTO C##finnk.tab_listado_clientes (id_cliente, correo_cliente)
VALUES (1,'jcastro@gmail.com');
INSERT INTO C##finnk.tab_listado_clientes (id_cliente, correo_cliente)
VALUES (2,'acontreras@gmail.com');
INSERT INTO C##finnk.tab_listado_clientes (id_cliente, correo_cliente)
VALUES (3,'lmena@gmail.com');
END;
INSERT INTO C##finnk.tab_listado_clientes (id_cliente, correo_cliente)
VALUES (4,'lmena@gmail.com');
END;
```

Figura 13. Insert tabla clientes.  
Fuente: Propia

INSERT para la tabla de reclamos:

```
--Reclamos insert
select * from C##finnk.tab_listado_reclamos;

BEGIN
INSERT INTO C##finnk.tab_listado_reclamos (id_reclamos,nombre_reclamo, comentario_reclamo)
VALUES (1,'Luis','No me llego el producto con el accesorio extra que le compre');
INSERT INTO C##finnk.tab_listado_reclamos (id_reclamos,nombre_reclamo, comentario_reclamo)
VALUES (2,'axel','Excelente servicio');
INSERT INTO C##finnk.tab_listado_reclamos (id_reclamos,nombre_reclamo, comentario_reclamo)
VALUES (3,'karol','Excelente servicio');
INSERT INTO C##finnk.tab_listado_reclamos (id_reclamos,nombre_reclamo, comentario_reclamo)
VALUES (4,'alodra','Excelente');
END;
```

Figura 14. Insert tabla reclamos.  
Fuente: Propia

## Procedimientos almacenados

Son una serie de instrucciones que se pueden invocar desde una consulta en otro lado de las líneas de código, este procedimiento debe ser capaz de insertar, modificar o eliminar un dato. A continuación, se mostrarán los procedimientos almacenados creados.

Procedimiento Almacenado 1:

```
--PROCEDIMIENTOS ALMACENADOS--  
--P1--  
CREATE OR REPLACE PROCEDURE C##finnk.InsertarCliente(  
    p_id_cliente IN C##finnk.tab_listado_clientes.id_cliente%TYPE,  
    p_correo_cliente IN C##finnk.tab_listado_clientes.correo_cliente%TYPE  
) AS  
BEGIN  
    INSERT INTO C##finnk.tab_listado_clientes(id_cliente, correo_cliente)  
    VALUES (p_id_cliente, p_correo_cliente);  
    COMMIT;  
END;
```

Figura 15. Procedimiento almacenado Insertar cliente.

Fuente: Propia

Procedimiento Almacenado 2:

```
--P2--  
CREATE OR REPLACE PROCEDURE C##finnk.ActualizarPrecioProducto(  
    p_id_catalogo IN C##finnk.tab_catalogo_productos.id_catalogo%TYPE,  
    p_precio_producto IN C##finnk.tab_catalogo_productos.precio_producto%TYPE  
) AS  
BEGIN  
    UPDATE C##finnk.tab_catalogo_productos  
    SET precio_producto = p_precio_producto  
    WHERE id_catalogo = p_id_catalogo;  
    COMMIT;  
END;
```

Figura 16. Procedimiento almacenado actualizar precio producto.

Fuente: Propia

Procedimiento Almacenado 3:

```
--P3--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerProveedoresPorEstado(
    p_estado_proveedor IN C##finnk.tab_listado_proveedores.estado_proveedor%TYPE
) AS
    v_proveedor C##finnk.tab_listado_proveedores%ROWTYPE;
BEGIN
    SELECT *
    INTO v_proveedor
    FROM C##finnk.tab_listado_proveedores
    WHERE estado_proveedor = p_estado_proveedor;

    DBMS_OUTPUT.PUT_LINE('ID_Proveedor: ' || v_proveedor.id_proveedor);
    DBMS_OUTPUT.PUT_LINE('Nombre_Proveedor: ' || v_proveedor.nombre_proveedor);

END;
```

Figura 17. Procedimiento almacenado proveedor por estado.  
Fuente: Propia

#### Procedimiento Almacenado 4:

```
--P4--
CREATE OR REPLACE PROCEDURE C##finnk.InsertarEmpleado(
    p_id_empleado IN C##finnk.tab_listado_empleados.id_empleado%TYPE,
    p_id_tienda IN C##finnk.tab_listado_empleados.fk_cienda%TYPE,
    p_nombre_empleado IN C##finnk.tab_listado_empleados.nombre_empleado%TYPE,
    p_apellido_empleado IN C##finnk.tab_listado_empleados.apellidos_empleado%TYPE,
    p_correo_empleado IN C##finnk.tab_listado_empleados.correo_empleado%TYPE,
    p_telefono_empleado IN C##finnk.tab_listado_empleados.telefono_empleado%TYPE,
    p_salario_empleado IN C##finnk.tab_listado_empleados.salario_empleado%TYPE,
    p_puesto_empleado IN C##finnk.tab_listado_empleados.puesto_empleado%TYPE,
    p_nacionalidad_empleado IN C##finnk.tab_listado_empleados.nacionalidad_empleado%TYPE,
    p_estado_empleado IN C##finnk.tab_listado_empleados.estado_empleado%TYPE
) AS
BEGIN
    INSERT INTO C##finnk.tab_listado_empleados(id_empleado, fk_cienda, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado)
    VALUES (p_id_empleado, p_id_tienda, p_nombre_empleado, p_apellido_empleado, p_correo_empleado, p_telefono_empleado, p_salario_empleado);
    COMMIT;
END;
```

Figura 18. Procedimiento almacenado insertar empleado.  
Fuente: Propia

#### Procedimiento Almacenado 5:

```
--P5--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerNumeroTiendasPorLocalidad(
    p_localidad_cienda IN C##finnk.tab_listado_cienda.localidad_cienda%TYPE,
    p_numero_tiendas OUT NUMBER
) AS
BEGIN
    SELECT COUNT(*)
    INTO p_numero_tiendas
    FROM C##finnk.tab_listado_cienda
    WHERE localidad_cienda = p_localidad_cienda;
END;
```

Figura 19. Procedimiento almacenado obtener número tiendas por localidad.  
Fuente: Propia.

#### Procedimiento Almacenado 6:

```

CREATE OR REPLACE PROCEDURE C##finnk.kEliminarReclamo(
    p_id_reclamo IN C##finnk.tab_listado_reclamos.id_reclamos%TYPE
) AS
BEGIN
    DELETE FROM C##finnk.tab_listado_reclamos
    WHERE id_reclamos = p_id_reclamo;
    COMMIT;
END;

```

Figura 20. Procedimiento almacenado eliminar reclamo

Fuente: Propia.

#### Procedimiento Almacenado 7:

```

CREATE OR REPLACE PROCEDURE C##finnk.ObtenerEmpleadosPorTienda(
    p_id_tienda IN C##finnk.tab_listado_tienda.id_tienda%TYPE
) AS
BEGIN
    SELECT *
    FROM C##finnk.tab_listado_empleados
    WHERE fk_tienda = p_id_tienda;
END;

```

Figura 21. Procedimiento almacenado obtener empleado por tienda.

Fuente: Propia.

#### Procedimiento Almacenado 8:

```

CREATE OR REPLACE PROCEDURE C##finnk.ActualizarEstadoProveedor(
    p_id_proveedor IN C##finnk.tab_listado_proveedores.id_proveedor%TYPE,
    p_estado_proveedor IN C##finnk.tab_listado_proveedores.estado_proveedor%TYPE
) AS
BEGIN
    UPDATE C##finnk.tab_listado_proveedores
    SET estado_proveedor = p_estado_proveedor
    WHERE id_proveedor = p_id_proveedor;
    COMMIT;
END;

```

Figura 22. Procedimiento almacenado actualizar estado proveedor.

Fuente: Propia.

#### Procedimiento Almacenado 9:

```
--P9--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerTotalReclamos(
    p_total_reclamos OUT NUMBER
) AS
BEGIN
    SELECT COUNT(*)
    INTO p_total_reclamos
    FROM C##finnk.tab_listado_reclamos;
END;
```

Figura 23. Procedimiento almacenado obtener total reclamos.

Fuente: Propia.

#### Procedimiento Almacenado 10:

```
--P10--
CREATE OR REPLACE PROCEDURE C##finnk.EliminarCliente(
    p_id_cliente IN C##finnk.tab_listado_clientes.id_cliente%TYPE
) AS
BEGIN
    DELETE FROM C##finnk.tab_listado_clientes
    WHERE id_cliente = p_id_cliente;
    COMMIT;
END;
```

Figura 24. Procedimiento almacenado eliminar cliente.

Fuente: Propia.

#### Procedimiento Almacenado 11:

```
--P11--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerProductosAgotados AS
    -- Declara una variable de registro para almacenar los resultados de la consulta
    v_producto C##finnk.tab_catalogo_productos%ROWTYPE;
BEGIN
    -- Utiliza la cláusula INTO para almacenar los resultados de la consulta en la variable de registro
    SELECT *
    INTO v_producto
    FROM C##finnk.tab_catalogo_productos
    WHERE existencias_producto = 0;

    -- Realiza alguna operación con los resultados, por ejemplo, mostrar los datos en la salida
    DBMS_OUTPUT.PUT_LINE('ID_Catalogo: ' || v_producto.id_catalogo);
    DBMS_OUTPUT.PUT_LINE('Imagen_Producto: ' || v_producto.imagen_producto);
    -- Continúa con las demás columnas según tus necesidades
END;
```

Figura 25. Procedimiento almacenado obtener productos agotados.

Fuente: Propia.

#### Procedimiento Almacenado 12:

```

CREATE OR REPLACE PROCEDURE C##finnk.ActualizarSalarioEmpleado(
    p_id_empleado IN C##finnk.tab_listado_empleados.id_empleado%TYPE,
    p_salario_empleado IN c##finnk.tab_listado_empleados.salario_empleado%TYPE
) AS
BEGIN
    UPDATE C##finnk.tab_listado_empleados
    SET salario_empleado = p_salario_empleado
    WHERE id_empleado = p_id_empleado;
    COMMIT;
END;

```

Figura 26. Procedimiento almacenado actualizar salario empleado.

Fuente: Propia.

#### Procedimiento Almacenado 13:

```

CREATE OR REPLACE PROCEDURE C##finnk.ActualizarSalarioEmpleado(
    p_id_empleado IN C##finnk.tab_listado_empleados.id_empleado%TYPE,
    p_salario_empleado IN c##finnk.tab_listado_empleados.salario_empleado%TYPE
) AS
BEGIN
    UPDATE C##finnk.tab_listado_empleados
    SET salario_empleado = p_salario_empleado
    WHERE id_empleado = p_id_empleado;
    COMMIT;
END;

```

Figura 27. Procedimiento almacenado actualizar salario empleado.

Fuente: Propia.

#### Procedimiento Almacenado 14:

```

--P14--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerNumeroEmpleadosPorTienda(
    p_id_tienda IN C##finnk.tab_listado_tienda.id_tienda%TYPE,
    p_numero_empleados OUT NUMBER
) AS
BEGIN
    SELECT COUNT(*)
    INTO p_numero_empleados
    FROM C##finnk.tab_listado_empleados
    WHERE fk_tienda = p_id_tienda;
END;

```

Figura 28. Procedimiento almacenado obtener número empleado por tienda.

Fuente: Propia.

#### Procedimiento Almacenado 15:

```
--P15--
CREATE OR REPLACE PROCEDURE C##finnk.EliminarProducto(
    p_id_catalogo IN C##finnk.tab_catalogo_productos.id_catalogo%TYPE
) AS
BEGIN
    DELETE FROM C##finnk.tab_catalogo_productos
    WHERE id_catalogo = p_id_catalogo;
    COMMIT;
END;
```

Figura 29. Procedimiento almacenado eliminar producto.

Fuente: Propia.

#### Procedimiento Almacenado 16:

```
--P16--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerClientesPorCorreo(
    p_correo_cliente IN VARCHAR2
) AS
    v_cliente C##finnk.tab_listado_clientes%ROWTYPE;
BEGIN
    SELECT *
    INTO v_cliente
    FROM C##finnk.tab_listado_clientes
    WHERE correo_cliente = p_correo_cliente;

    DBMS_OUTPUT.PUT_LINE('ID_Cliente: ' || v_cliente.id_cliente);
    DBMS_OUTPUT.PUT_LINE('Correo_Cliente: ' || v_cliente.correo_cliente);

END;
```

Figura 30. Procedimiento almacenado obtener cliente por correo.

Fuente: Propia.

#### Procedimiento Almacenado 17:

```
--P17--
CREATE OR REPLACE PROCEDURE C##finnk.ActualizarNombreEmpleado(
    p_id_empleado IN C##finnk.tab_listado_empleados.id_empleado%TYPE,
    p_nombre_empleado IN C##finnk.tab_listado_empleados.nombre_empleado%TYPE
) AS
BEGIN
    UPDATE C##finnk.tab_listado_empleados
    SET nombre_empleado = p_nombre_empleado
    WHERE id_empleado = p_id_empleado;
    COMMIT;
END;
```

Figura 31. Procedimiento almacenado actualizar nombre empleado.

Fuente: Propia.

### Procedimiento Almacenado 18:

```
--P18--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerEmpleadosPorSalarioMaximo AS
    v_id_empleado C##finnk.tab_listado_empleados.id_empleado%TYPE;
    v_nombre_empleado C##finnk.tab_listado_empleados.nombre_empleado%TYPE;

BEGIN
    DECLARE
        v_salario_maximo C##finnk.tab_listado_empleados.salario_empleado%TYPE;
    BEGIN
        SELECT MAX(salario_empleado) INTO v_salario_maximo FROM c##finnk.tab_listado_empleados;

        FOR emp IN (SELECT *
                     FROM C##finnk.tab_listado_empleados
                     WHERE salario_empleado = v_salario_maximo)
        LOOP
            v_id_empleado := emp.id_empleado;
            v_nombre_empleado := emp.nombre_empleado;

            DBMS_OUTPUT.PUT_LINE('ID_Empleado: ' || v_id_empleado);
            DBMS_OUTPUT.PUT_LINE('Nombre_Empleado: ' || v_nombre_empleado);
        END LOOP;
    END;
END;
```

Figura 32. Procedimiento almacenado obtener empleado por salario mínimo.

Fuente: Propia.

### Procedimiento Almacenado 19:

```
--P19--
CREATE OR REPLACE PROCEDURE C##finnk.InsertarReclamo(
    p_id_reclamo IN C##finnk.tab_listado_reclamos.id_reclamos%TYPE,
    p_nombre_reclamo IN C##finnk.tab_listado_reclamos.nombre_reclamo%TYPE,
    p_comentario_reclamo IN C##finnk.tab_listado_reclamos.comentario_reclamo%TYPE
) AS
BEGIN
    INSERT INTO C##finnk.tab_listado_reclamos(id_reclamos, nombre_reclamo, comentario_reclamo)
    VALUES (p_id_reclamo, p_nombre_reclamo, p_comentario_reclamo);
    COMMIT;
END;
```

Figura 33. Procedimiento almacenado insertar reclamo.

Fuente: Propia.

### Procedimiento Almacenado 20:

```
--P20--  
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerProductosDisponibles AS  
BEGIN  
    CURSOR c_productos IS  
        SELECT *      FROM C##finnk.tab_catalogo_productos  
        WHERE existencias_producto > 0;  
  
    v_id_catalogo C##finnk.tab_catalogo_productos.id_catalogo%TYPE;  
    v_imagen_producto C##finnk.tab_catalogo_productos.imagen_producto%TYPE;  
  
    OPEN c_productos;  
    FETCH c_productos INTO v_id_catalogo, v_imagen_producto, ...;  
  
    LOOP  
        DBMS_OUTPUT.PUT_LINE('ID_Catalogo: ' || v_id_catalogo);  
        DBMS_OUTPUT.PUT_LINE('Imagen_Producto: ' || v_imagen_producto);  
  
        FETCH c_productos INTO v_id_catalogo, v_imagen_producto, ...;  
  
        EXIT WHEN c_productos%NOTFOUND;  
    END LOOP;  
  
    CLOSE c_productos;  
END;
```

Figura 34. Procedimiento almacenado obtener producto disponible.

Fuente: Propia.

#### Procedimiento Almacenado 21:

```
--P21--  
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerNumeroProductos(  
    p_numero_productos OUT NUMBER  
) AS  
BEGIN  
    SELECT COUNT(*)  
    INTO p_numero_productos  
    FROM C##finnk.tab_catalogo_productos;  
END;
```

Figura 35. Procedimiento almacenado obtener número producto.

Fuente: Propia.

#### Procedimiento Almacenado 22:

```

CREATE OR REPLACE PROCEDURE C##finnk.ActualizarExistenciasProducto(
    p_id_catalogo IN C##finnk.tab_catalogo_productos.id_catalogo%TYPE,
    p_existencias_producto IN C##finnk.tab_catalogo_productos.existencias_producto%TYPE
) AS
BEGIN
    UPDATE C##finnk.tab_catalogo_productos
    SET existencias_producto = p_existencias_producto
    WHERE id_catalogo = p_id_catalogo;
    COMMIT;
END;

```

Figura 36. Procedimiento almacenado actualizar existencia producto.

Fuente: Propia.

### Procedimiento Almacenado 23:

```

--P22--
CREATE OR REPLACE PROCEDURE C##finnk.ObtenerEmpleadosPorSalarioMinimo AS
    v_id_empleado C##finnk.tab_listado_empleados.id_empleado%TYPE;
    v_nombre_empleado C##finnk.tab_listado_empleados.nombre_empleado%TYPE;

BEGIN
    DECLARE
        v_salario_minimo C##finnk.tab_listado_empleados.salario_empleado%TYPE;
    BEGIN
        SELECT MIN(salario_empleado) INTO v_salario_minimo FROM C##finnk.tab_listado_empleados;

        FOR emp IN (SELECT *
                     FROM C##finnk.tab_listado_empleados
                     WHERE salario_empleado >= v_salario_minimo)
        LOOP
            v_id_empleado := emp.id_empleado;
            v_nombre_empleado := emp.nombre_empleado;

            DBMS_OUTPUT.PUT_LINE('ID_Emppleado: ' || v_id_empleado);
            DBMS_OUTPUT.PUT_LINE('Nombre_Emppleado: ' || v_nombre_empleado);
        END LOOP;
    END;
END;

```

Figura 37. Procedimiento almacenado obtener empleado por salario mínimo.

Fuente: Propia.

### Procedimiento Almacenado 24:

```

--P24--
CREATE OR REPLACE PROCEDURE C##finnk.InsertarTienda(
    p_id_tienda IN C##finnk.tab_listado_tienda.id_tienda%TYPE,
    p_localidad_tienda IN C##finnk.tab_listado_tienda.localidad_tienda%TYPE,
    p_estado_tienda IN C##finnk.tab_listado_tienda.estado_tienda%TYPE
) AS
BEGIN
    INSERT INTO C##finnk.tab_listado_tienda(id_tienda, localidad_tienda, estado_tienda)
    VALUES (p_id_tienda, p_localidad_tienda, p_estado_tienda);
    COMMIT;
END;

```

Figura 38. Procedimiento almacenado insertar tienda

Fuente: Propia.

## Procedimiento Almacenado 25:

```
--P25--  
CREATE OR REPLACE PROCEDURE C##finnk.ActualizarExistenciasProducto(  
    p_id_catalogo IN C##finnk.tab_catalogo_productos.id_catalogo%TYPE,  
    p_existencias_producto IN C##finnk.tab_catalogo_productos.existencias_producto%TYPE  
) AS  
BEGIN  
    UPDATE C##finnk.tab_catalogo_productos  
    SET existencias_producto = p_existencias_producto  
    WHERE id_catalogo = p_id_catalogo;  
    COMMIT;  
END;
```

Figura 39. Procedimiento almacenado actualizar existencia producto.

Fuente: Propia.

## Vistas

Una vista es una alternativa para mostrar datos de varias tablas, es una tabla virtual que almacena una consulta. Los datos accesibles a través de la vista no están almacenados en la base de datos, en la base de datos se guarda la definición de la vista y no el resultado de ella. A continuación, se muestran las vistas creadas:

### Vista 1:

```
--VISTA 1  
--Vista que muestre el id,nombre, apellido y salario de todos los empleados  
--con salarios iguales o inferiores al promedio de salarios en la tabla Tab_Listado_Empleados  
CREATE OR REPLACE VIEW C##finnk.Vista_Employados AS  
SELECT id_empleado, nombre_empleado, apellidos_empleado, salario_empleado  
FROM C##finnk.tab_listado_empleados  
WHERE salario_empleado <= (SELECT AVG(salario_empleado) FROM C##finnk.tab_listado_empleados);  
  
SELECT * FROM C##finnk.Vista_Employados;
```

Figura 40. Vista empleados

Fuente: Propia.

### Vista 2:

```
--VISTA 2  
--Vista que muestre el nombre de cada departamento, id de cada empleado y el nombre del empleado  
CREATE OR REPLACE VIEW C##finnk.Vista_Cliente AS  
SELECT id_cliente || '-' || correo_cliente AS Informacion_Cliente  
FROM C##finnk.tab_listado_clientes;  
  
SELECT Informacion_Cliente FROM C##finnk.Vista_Cliente;
```

Figura 41. Vista cliente

Fuente: Propia.

Vista 3:

```
--VISTA 3
--Crear una vista que muestre el nombre, correo, el id del reclamo y el reclamo del cliente
CREATE OR REPLACE VIEW C##finnk.Vista_Reclamos AS
SELECT
    c.id_cliente AS ID_Cliente,
    c.correo_cliente AS Correo_Cliente,
    r.comentario_reclamo AS Reclamo
FROM
    C##finnk.tab_listado_clientes c
JOIN
    C##finnk.tab_listado_reclamos r ON c.id_cliente = r.fk_reclamos;
SELECT * FROM C##finnk.Vista_Reclamos;
```

Figura 42. Vista reclamos

Fuente: Propia.

Vista 4:

```
--VISTA 4
--Ordenar los productos por orden alfabético con su nombre
CREATE OR REPLACE VIEW C##finnk.Vista_Productos_Alfab AS
SELECT id_catalogo , nombre_producto
FROM C##finnk.tab_catalogo_productos
ORDER BY nombre_producto;

SELECT * FROM C##finnk.Vista_Productos_Alfab;
```

Figura 43. Vista productos orden alfabético

Fuente: Propia.

Vista 5:

```
--VISTA 5
---Ordenar el precio de los productos de mayor a menor
CREATE OR REPLACE VIEW C##finnk.Vista_Productos_May_Men AS
SELECT * FROM C##finnk.tab_catalogo_productos
ORDER BY precio_producto ASC;

SELECT * FROM C##finnk.Vista_Productos_May_Men;
```

Figura 44. Vista Productos mayor y menor

Fuente: Propia.

Vista 6:

```
--VISTA 6
--Mostrar la cantidad total de empleados
CREATE OR REPLACE VIEW C##finnk.Vista_Total_Empleados AS
SELECT COUNT(*) "CANTIDAD DE EMPLEADOS" FROM C##finnk.tab_listado_empleados;

SELECT * FROM C##finnk.Vista_Total_Empleados;
```

Figura 45. Vista Total empleados

Fuente: Propia.

Vista 7:

```
--VISTA 7
--Mostrar el salario total de todos los empleados
CREATE OR REPLACE VIEW C##finnk.Vista_Employados_Salarios AS
SELECT sum(salario_empleado) "TOTAL DE SALARIO" FROM C##finnk.tab_listado_empleados;

SELECT * FROM C##finnk.Vista_Employados_Salarios;
```

Figura 46. Vista Empleados salarios

Fuente: Propia

Vista 8:

```
--VISTA 8
--Conteo y agrupamientos de los empleados por puesto , del
CREATE OR REPLACE VIEW C##finnk.Vista_Employado_Puesto AS
SELECT puesto_empleado,
       COUNT(*) AS Conteo_Employados
FROM
C##finnk.tab_listado_empleados
GROUP BY
puesto_empleado;

SELECT * FROM C##finnk.Vista_Employado_Puesto;
```

Figura 47. Vista Empleado puesto

Fuente: Propia.

Vista 9:

```
--VISTA 9
--Mostrar todos los productos que tengan estado = true
CREATE OR REPLACE VIEW C##finnk.Vista_Productos_True AS
SELECT id_catalogo,nombre_producto, estado_producto
FROM
C##finnk.tab_catalogo_productos
WHERE
estado_producto = 'Y';

SELECT * FROM C##finnk.Vista_Productos_True;
```

Figura 48. Vista Productos Activos

Fuente: Propia

Vista 10:

```
--VISTA 10
--Mostrar todos los proveedores que sean Costarricenses
CREATE OR REPLACE VIEW C##finnk.Vista_Proveedores AS
SELECT id_proveedor,nombre_proveedor, nacionalidad_proveedor
FROM
C##finnk.tab_listado_proveedores
WHERE
nacionalidad_proveedor = 'Costarricense';

SELECT * FROM C##finnk.Vista_Proveedores;
```

Figura 49. Vista Proveedores

Fuente: Propia

## Funciones

Función 1:

```

--FUNCIONES--
--F1--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerCorreoClientePorID(
    p_id_cliente IN C##finnk.tab_listado_clientes.id_cliente%TYPE
) RETURN VARCHAR2 AS
    v_correo_cliente C##finnk.tab_listado_clientes.correo_cliente%TYPE;
BEGIN
    SELECT correo_cliente INTO v_correo_cliente
    FROM C##finnk.tab_listado_clientes
    WHERE id_cliente = p_id_cliente;

    RETURN v_correo_cliente;
END;
-- Ejecutando Funcion 1 mediante id 1
DECLARE
    v_email VARCHAR2(100);
BEGIN
    v_email := C##finnk.ObtenerCorreoClientePorID(1);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
END;

```

Figura 50. Función obtener correo de cliente por ID

Fuente: Propia

### Función 2:

```

--F2--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerPrecioProductoPorID(
    p_id_producto IN C##finnk.tab_catalogo_productos.id_catalogo%TYPE
) RETURN NUMBER AS
    v_precio_producto C##finnk.tab_catalogo_productos.precio_producto%TYPE;
BEGIN
    SELECT precio_producto INTO v_precio_producto
    FROM C##finnk.tab_catalogo_productos
    WHERE id_catalogo = p_id_producto;

    RETURN v_precio_producto;
END;

--Ejecutando Funcion mediante id 5 para obtener precio
DECLARE
    v_price NUMBER;
    v_product_id NUMBER := 5;
BEGIN
    v_price := C##finnk.ObtenerPrecioProductoPorID(v_product_id);
    DBMS_OUTPUT.PUT_LINE('Precio: ' || v_price);
END;

select * from C##finnk.tab_listado_proveedores;

```

Figura 51. Función obtener precio de producto por ID

Fuente: Propia

### Función 3:

```
--F3--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerProveedoresPorEstadoFunc(
    p_estado_proveedor IN C##finnk.tab_listado_proveedores.estado_proveedor%TYPE
) RETURN SYS_REFCURSOR AS
    v_proveedores SYS_REFCURSOR;
BEGIN
    OPEN v_proveedores FOR
        SELECT *
        FROM C##finnk.tab_listado_proveedores
        WHERE estado_proveedor = p_estado_proveedor;

    RETURN v_proveedores;
END;

--Ejecutando funcion 3 en donde se ejecutan los unicos que estan en estado True o Y
DECLARE
    v_estado_proveedor C##finnk.tab_listado_proveedores.estado_proveedor%TYPE := 'Y';
    v_cursor SYS_REFCURSOR;
    v_row C##finnk.tab_listado_proveedores%ROWTYPE;
BEGIN
    v_cursor := C##finnk.ObtenerProveedoresPorEstadoFunc(v_estado_proveedor);

    LOOP
        FETCH v_cursor INTO v_row;
        EXIT WHEN v_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('ID Proveedor: ' || v_row.id_proveedor || ', Nombre: ' || v_row.nombre_proveedor);
    END LOOP;

    CLOSE v_cursor;
END;
```

Figura 52. Función obtener proveedor por estado  
Fuente: Propia

#### Función 4:

```
--F4--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerEmpleadosPorTiendaFunc(
    p_id_tienda IN C##finnk.tab_listado_tienda.id_tienda%TYPE
) RETURN SYS_REFCURSOR AS
    v_empleados SYS_REFCURSOR;
BEGIN
    OPEN v_empleados FOR
        SELECT *
        FROM C##finnk.tab_listado_empleados
        WHERE fk_tienda = p_id_tienda;

    RETURN v_empleados;
END;

-- Ejecuta la funcion 4 donde se le envia id 1 para verificar el empleado en esa tienda
DECLARE
    v_id_tienda C##finnk.tab_listado_tienda.id_tienda%TYPE := 1;
    v_cursor SYS_REFCURSOR;
    v_row C##finnk.tab_listado_empleados%ROWTYPE;
BEGIN
    v_cursor := C##finnk.ObtenerEmpleadosPorTiendaFunc(v_id_tienda);

    LOOP
        FETCH v_cursor INTO v_row;
        EXIT WHEN v_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('ID Empleado: ' || v_row.id_empleado || ', Nombre: ' || v_row.nombre_empleado);
    END LOOP;

    CLOSE v_cursor;
END;
```

Figura 53. Función obtener empleado por tienda  
Fuente: Propia

#### Función 5:

```

--F5--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerReclamoPorID(
    p_id_reclamo IN C##finnk.tab_listado_reclamos.id_reclamos%TYPE
) RETURN VARCHAR2 AS
    v_comentario_reclamo C##finnk.tab_listado_reclamos.comentario_reclamo%TYPE;
BEGIN
    SELECT comentario_reclamo INTO v_comentario_reclamo
    FROM C##finnk.tab_listado_reclamos
    WHERE id_reclamos = p_id_reclamo;

    RETURN v_comentario_reclamo;
END;

--Llama la funcion 5 Se obtiene comentario mediante id l
DECLARE
    v_id_reclamo C##finnk.tab_listado_reclamos.id_reclamos%TYPE := 1;
    v_comentario_reclamo VARCHAR2(4000);
BEGIN
    v_comentario_reclamo := C##finnk.ObtenerReclamoPorID(v_id_reclamo);
    DBMS_OUTPUT.PUT_LINE('Comentario de reclamo: ' || v_comentario_reclamo);
END;

```

Figura 54. Función obtener reclamo por ID

Fuente: Propia

#### Función 6:

```

--F6--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerEstadoProveedorPorID(
    p_id_proveedor IN C##finnk.tab_listado_proveedores.id_proveedor%TYPE
) RETURN CHAR AS
    v_estado_proveedor C##finnk.tab_listado_proveedores.estado_proveedor%TYPE;
BEGIN
    SELECT estado_proveedor INTO v_estado_proveedor
    FROM C##finnk.tab_listado_proveedores
    WHERE id_proveedor = p_id_proveedor;

    RETURN v_estado_proveedor;
END;

--Llama a la funcion 6 lo que realiza es obtener el estado del proveedor mediante id
DECLARE
    v_id_proveedor C##finnk.tab_listado_proveedores.id_proveedor%TYPE := 1;
    v_estado CHAR(1);
BEGIN
    v_estado := C##finnk.ObtenerEstadoProveedorPorID(v_id_proveedor);
    DBMS_OUTPUT.PUT_LINE('Estado del proveedor: ' || v_estado);
END;

```

Figura 55. Función obtener estado de proveedor por ID

Fuente: Propia

#### Función 7:

```

--F7--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerTotalReclamosFunc RETURN NUMBER AS
  v_total_reclamos NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_total_reclamos
  FROM C##finnk.tab_listado_reclamos;

  RETURN v_total_reclamos;
END;

--Llama a la funcion 7 y obtiene los reclamos totales

DECLARE
  v_total_reclamos NUMBER;
BEGIN
  v_total_reclamos := C##finnk.ObtenerTotalReclamosFunc();
  DBMS_OUTPUT.PUT_LINE('Total de reclamos: ' || v_total_reclamos);
END;

```

Figura 56. Función obtener total de reclamos

Fuente: Propia

Función 8:

```

--F8--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerNombreEmpleadoPorID(
  p_id_empleado IN C##finnk.tab_listado_empleados.id_empleado%TYPE
) RETURN VARCHAR2 AS
  v_nombre_empleado VARCHAR2(80);
BEGIN
  SELECT nombre_empleado || ' ' || apellidos_empleado INTO v_nombre_empleado
  FROM C##finnk.tab_listado_empleados
  WHERE id_empleado = p_id_empleado;

  RETURN v_nombre_empleado;
END;

--Ejecuta la funcion 8 se prueba con id para obtener el empleado
DECLARE
  v_id_empleado C##finnk.tab_listado_empleados.id_empleado%TYPE := 1;
  v_nombre VARCHAR2(80);
BEGIN
  v_nombre := C##finnk.ObtenerNombreEmpleadoPorID(v_id_empleado);
  DBMS_OUTPUT.PUT_LINE('Nombre del empleado: ' || v_nombre);
END;

```

Figura 57. Función obtener nombre de empleado por ID

Fuente: Propia

Función 9:

```
--F9--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerNumeroTiendasPorLocalidadFunc(
    p_localidad_tienda IN C##finnk.tab_listado_tienda.localidad_tienda%TYPE
) RETURN NUMBER AS
    v_numero_tiendas NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_numero_tiendas
    FROM C##finnk.tab_listado_tienda
    WHERE localidad_tienda = p_localidad_tienda;

    RETURN v_numero_tiendas;
END;

--Ejecuta la funcion 9 y obtiene el numero de tiendas por localidad

DECLARE
    v_localidad_tienda C##finnk.tab_listado_tienda.localidad_tienda%TYPE := 'Cartago';
    v_numero_tiendas NUMBER;
BEGIN
    v_numero_tiendas := C##finnk.ObtenerNumeroTiendasPorLocalidadFunc(v_localidad_tienda);
    DBMS_OUTPUT.PUT_LINE('Número de tiendas en ' || v_localidad_tienda || ': ' || v_numero_tiendas);
END;
```

Figura 58. Función obtener número de tienda por localidad

Fuente: Propia

#### Función 10:

```
--F10--
select * from C##finnk.tab_listado_empleados;
CREATE OR REPLACE FUNCTION C##finnk.ObtenerEmpleadoPorCorreo(
    p_correo_empleado IN C##finnk.tab_listado_empleados.correo_empleado%TYPE
) RETURN C##finnk.tab_listado_empleados%ROWTYPE AS
    v_empleado C##finnk.tab_listado_empleados%ROWTYPE;
BEGIN
    SELECT *
    INTO v_empleado
    FROM C##finnk.tab_listado_empleados
    WHERE correo_empleado = p_correo_empleado;

    RETURN v_empleado;
END;

--Ejecuta funcion 10 se le pasa un correo existente y retorna id y nombre del empleado
DECLARE
    v_correo_empleado C##finnk.tab_listado_empleados.correo_empleado%TYPE := 'olmanefr@gmail.com';
    v_resultado C##finnk.tab_listado_empleados%ROWTYPE;
BEGIN
    v_resultado := C##finnk.ObtenerEmpleadoPorCorreo(v_correo_empleado);

    DBMS_OUTPUT.PUT_LINE('ID Empleado: ' || v_resultado.id_empleado || ', Nombre: ' || v_resultado.nombre_empleado);
END;
```

Figura 59. Función obtener empleado por correo

Fuente: Propia

#### Función 11:

```

--F11-
select * from C##finnk.tab_listado_tienda;
CREATE OR REPLACE FUNCTION C##finnk.ObtenerEstadoTiendaPorID(
    p_id_tienda IN C##finnk.tab_listado_tienda.id_tienda%TYPE
) RETURN CHAR AS
    v_estado_tienda C##finnk.tab_listado_tienda.estado_tienda%TYPE;
BEGIN
    SELECT estado_tienda INTO v_estado_tienda
    FROM C##finnk.tab_listado_tienda
    WHERE id_tienda = p_id_tienda;

    RETURN v_estado_tienda;
END;

--Ejecuta Funcion 11 retorna el estado de tienda por el id
DECLARE
    v_id_tienda C##finnk.tab_listado_tienda.id_tienda%TYPE := 1;
    v_estado CHAR(1);
BEGIN
    v_estado := C##finnk.ObtenerEstadoTiendaPorID(v_id_tienda);
    DBMS_OUTPUT.PUT_LINE('Estado de la tienda: ' || v_estado);
END;

```

Figura 60. Función obtener estado de la tienda por ID

Fuente: Propia

## Función 12:

```

--F12--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerSalarioEmpleadoPorID(
    p_id_empleado IN C##finnk.tab_listado_empleados.id_empleado%TYPE
) RETURN NUMBER AS
    v_salario_empleado C##finnk.tab_listado_empleados.salario_empleado%TYPE;
BEGIN
    SELECT salario_empleado INTO v_salario_empleado
    FROM C##finnk.tab_listado_empleados
    WHERE id_empleado = p_id_empleado;

    RETURN v_salario_empleado;
END;

--Obtiene el salario del empleado por el id proporcionado

DECLARE
    v_id_empleado C##finnk.tab_listado_empleados.id_empleado%TYPE := 1;
    v_salario NUMBER;
BEGIN
    v_salario := C##finnk.ObtenerSalarioEmpleadoPorID(v_id_empleado);
    DBMS_OUTPUT.PUT_LINE('Salario de empleado: ' || v_salario);
END;

```

Figura 61. Función obtener salario de empleador por ID

Fuente: Propia

## Función 13:

```

--F13--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerProductoPorID(
    p_id_producto IN C##finnk.tab_catalogo_productos.id_catalogo%TYPE
) RETURN C##finnk.tab_catalogo_productos%ROWTYPE AS
    v_producto C##finnk.tab_catalogo_productos%ROWTYPE;
BEGIN
    SELECT *
    INTO v_producto
    FROM C##finnk.tab_catalogo_productos
    WHERE id_catalogo = p_id_producto;

    RETURN v_producto;
END;

--Ejecuta la funcion 13 retorna producto por el id
DECLARE
    v_id_producto C##finnk.tab_catalogo_productos.id_catalogo%TYPE := 1;
    v_resultado C##finnk.tab_catalogo_productos%ROWTYPE;
BEGIN
    v_resultado := C##finnk.ObtenerProductoPorID(v_id_producto);

    DBMS_OUTPUT.PUT_LINE('ID Producto: ' || v_resultado.id_catalogo || ', Nombre: ' || v_resultado.nombre_producto);
END;

```

Figura 62. Función obtener producto por ID  
Fuente: Propia

Función 14:

```

--F14--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerNumeroClientesFunc RETURN NUMBER AS
    v_numero_clientes NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_numero_clientes
    FROM C##finnk.tab_listado_clientes;

    RETURN v_numero_clientes;
END;

--Ejecuta la funcion 14 la cual imprime el numero total de clientes

DECLARE
    v_total_clientes NUMBER;
BEGIN
    v_total_clientes := C##finnk.ObtenerNumeroClientesFunc();
    DBMS_OUTPUT.PUT_LINE('Número total de clientes: ' || v_total_clientes);
END;

```

Figura 63. Función obtener número de clientes  
Fuente: Propia

Función 15:

```

--F15--
CREATE OR REPLACE FUNCTION C##finnk.ObtenerNombreProveedorPorID(
    p_id_proveedor IN C##finnk.tab_listado_proveedores.id_proveedor%TYPE
) RETURN VARCHAR2 AS
    v_nombre_proveedor VARCHAR2(80);
BEGIN
    SELECT nombre_proveedor || ' ' || apellidos_proveedor INTO v_nombre_proveedor
    FROM C##finnk.tab_listado_proveedores
    WHERE id_proveedor = p_id_proveedor;

    RETURN v_nombre_proveedor;
END;

--Ejecuta la funcion 15 y retorna el nombre del proveedor por el id

DECLARE
    v_id_proveedor C##finnk.tab_listado_proveedores.id_proveedor%TYPE := 1;
    v_nombre VARCHAR2(80);
BEGIN
    v_nombre := C##finnk.ObtenerNombreProveedorPorID(v_id_proveedor);
    DBMS_OUTPUT.PUT_LINE('Nombre del proveedor: ' || v_nombre);
END;

```

Figura 64. Función obtener Nombre de proveedor por ID

Fuente: Propia

## Paquetes

Paquete 1:

```

-- Paquete 1
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_PRODUCTOS_OBTENER AS
    PROCEDURE ObtenerProductosSP(p_cursor OUT SYS_REFCURSOR);
END PCK_FINNK_PRODUCTOS_OBTENER;

-- Paquete 1 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_PRODUCTOS_OBTENER AS
    PROCEDURE ObtenerProductosSP(p_cursor OUT SYS_REFCURSOR) AS
        BEGIN
            OPEN p_cursor FOR
                SELECT * FROM C##finnk.tab_catalogo_productos;
        END;
    END PCK_FINNK_PRODUCTOS_OBTENER;

```

Figura 65. Paquete obtener productos

Fuente: Propia

Paquete 2:

```

-- Paquete 2
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_PRODUCTOS_ELIMINAR AS
    PROCEDURE EliminarProductoSP(p_id_catalogo IN NUMBER);
END PCK_FINNK_PRODUCTOS_ELIMINAR;

-- Paquete 2 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_PRODUCTOS_ELIMINAR AS
    PROCEDURE EliminarProductoSP(p_id_catalogo IN NUMBER) AS
        BEGIN
            DELETE FROM C##finnk.tab_catalogo_productos
            WHERE id_catalogo = p_id_catalogo;

            COMMIT;
        EXCEPTION
            WHEN OTHERS THEN
                ROLLBACK;
                RAISE;
        END;
    END PCK_FINNK_PRODUCTOS_ELIMINAR;

```

Figura 66. Paquete obtener productos

Fuente: Propia

### Paquete 3:

```

-- Paquete 3
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_PRODUCTOS_EDITAR AS
    PROCEDURE EditarProductoSP(
        p_id_catalogo IN NUMBER,
        p_imagen_producto IN VARCHAR2,
        p_nombre_producto IN VARCHAR2,
        p_precio_producto IN NUMBER,
        p_existencias_producto IN NUMBER,
        p_estado_producto IN CHAR
    );
END PCK_FINNK_PRODUCTOS_EDITAR;
-- Paquete 3 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_PRODUCTOS_EDITAR AS
    PROCEDURE EditarProductoSP(
        p_id_catalogo IN NUMBER,
        p_imagen_producto IN VARCHAR2,
        p_nombre_producto IN VARCHAR2,
        p_precio_producto IN NUMBER,
        p_existencias_producto IN NUMBER,
        p_estado_producto IN CHAR
    ) AS
    BEGIN
        UPDATE C##finnk.tab_catalogo_productos
        SET imagen_producto = p_imagen_producto,
            nombre_producto = p_nombre_producto,
            precio_producto = p_precio_producto,
            existencias_producto = p_existencias_producto,
            estado_producto = p_estado_producto
        WHERE id_catalogo = p_id_catalogo;
        COMMIT;
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            RAISE;
    END;

```

Figura 67. Paquete obtener productos

Fuente: Propia

### Paquete 4:

```

-- Paquete 4
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_PRODUCTOS_AGREGAR AS
PROCEDURE AgregarProductoSP(
    p_imagen_producto IN VARCHAR2,
    p_nombre_producto IN VARCHAR2,
    p_precio_producto IN NUMBER,
    p_existencias_producto IN NUMBER,
    p_estado_producto IN CHAR
);
END PCK_FINNK_PRODUCTOS_AGREGAR;
-- Paquete 4 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_PRODUCTOS_AGREGAR AS
PROCEDURE AgregarProductoSP(
    p_imagen_producto IN VARCHAR2,
    p_nombre_producto IN VARCHAR2,
    p_precio_producto IN NUMBER,
    p_existencias_producto IN NUMBER,
    p_estado_producto IN CHAR
) AS
BEGIN
    INSERT INTO C##finnk.tab_catalogo_productos (
        imagen_producto, nombre_producto, precio_producto,
        existencias_producto, estado_producto
    ) VALUES (
        p_imagen_producto, p_nombre_producto, p_precio_producto,
        p_existencias_producto, p_estado_producto
    );
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
END PCK FINNK PRODUCTOS AGREGAR;

```

Figura 68. Paquete obtener productos

Fuente: Propia

#### Paquete 5:

```

-- Paquete 5
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_EMPLEADOS_OBTENER AS
    PROCEDURE ObtenerEmpleadosSP(p_cursor OUT SYS_REFCURSOR);
END PCK_FINNK_EMPLEADOS_OBTENER;
-- Paquete 5 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_EMPLEADOS_OBTENER AS
    PROCEDURE ObtenerEmpleadosSP(p_cursor OUT SYS_REFCURSOR) AS
    BEGIN
        OPEN p_cursor FOR
            SELECT e.id_empleado, e.nombre_empleado, e.apellidos_empleado, e.correo_empleado, e.telefono_empleado, e.salario_empleado,
                   e.puesto_empleado, e.nacionalidad_empleado, e.estado_empleado,
                   C##finnk.ObtenerInformacionAdicional(e.id_empleado, e.fk_tienda) AS info_adicional
            FROM C##finnk.tab_listado_empleados e;
    END;
END PCK_FINNK_EMPLEADOS_OBTENER;

```

Figura 69. Paquete obtener productos

Fuente: Propia

#### Paquete 6:

```

-- Paquete 6
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_EMPLEADOS_ELIMINAR AS
    PROCEDURE EliminarEmpleadoSP(p_id_empleado IN NUMBER, p_resultado OUT SYS_REFCURSOR);
END PCK_FINNK_EMPLEADOS_ELIMINAR;
-- Paquete 6 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_EMPLEADOS_ELIMINAR AS
    PROCEDURE EliminarEmpleadoSP(p_id_empleado IN NUMBER, p_resultado OUT SYS_REFCURSOR) AS
    BEGIN
        p_resultado := C##finnk.EliminarEmpleadosPorID(p_id_empleado);
    END;
END PCK_FINNK_EMPLEADOS_ELIMINAR;

```

Figura 70. Paquete obtener productos  
Fuente: Propia

### Paquete 7:

```
-- Paquete 7
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_EMPLEADOS_EDITAR AS
PROCEDURE EditarEmpleadoSP(
    p_id_empleado IN NUMBER, p_nombre_empleado IN VARCHAR2,
    p_apellidos_empleado IN VARCHAR2, p_correo_empleado IN VARCHAR2,
    p_telefono_empleado IN NUMBER, p_salario_empleado IN NUMBER,
    p_puesto_empleado IN VARCHAR2, p_nacionalidad_empleado IN VARCHAR2,
    p_estado_empleado IN CHAR, p_cursor OUT SYS_REFCURSOR);
END PCK_FINNK_EMPLEADOS_EDITAR;
--pageute 7 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_EMPLEADOS_EDITAR AS
PROCEDURE EditarEmpleadoSP(
    p_id_empleado IN NUMBER, p_nombre_empleado IN VARCHAR2,
    p_apellidos_empleado IN VARCHAR2, p_correo_empleado IN VARCHAR2,
    p_telefono_empleado IN NUMBER, p_salario_empleado IN NUMBER,
    p_puesto_empleado IN VARCHAR2, p_nacionalidad_empleado IN VARCHAR2,
    p_estado_empleado IN CHAR, p_cursor OUT SYS_REFCURSOR
) AS
BEGIN
    p_cursor := C##finnk.EditarEmpleado(
        p_id_empleado, p_nombre_empleado,
        p_apellidos_empleado, p_correo_empleado,
        p_telefono_empleado, p_salario_empleado,
        p_puesto_empleado, p_nacionalidad_empleado,
        p_estado_empleado
    );
END;
END PCK_FINNK_EMPLEADOS_EDITAR;
```

Figura 71. Paquete obtener productos  
Fuente: Propia

### Paquete 8:

```
-- Paquete 8
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_EMPLEADOS_AGREGAR AS
PROCEDURE AgregarEmpleadoSP(
    p_nombre_empleado IN VARCHAR2, p_apellidos_empleado IN VARCHAR2,
    p_correo_empleado IN VARCHAR2, p_telefono_empleado IN NUMBER,
    p_salario_empleado IN NUMBER, p_puesto_empleado IN VARCHAR2,
    p_nacionalidad_empleado IN VARCHAR2, p_estado_empleado IN CHAR,
    p_cursor OUT SYS_REFCURSOR
);
END PCK_FINNK_EMPLEADOS_AGREGAR;

-- Paquete 8 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_EMPLEADOS_AGREGAR AS
PROCEDURE AgregarEmpleadoSP(
    p_nombre_empleado IN VARCHAR2, p_apellidos_empleado IN VARCHAR2,
    p_correo_empleado IN VARCHAR2, p_telefono_empleado IN NUMBER,
    p_salario_empleado IN NUMBER, p_puesto_empleado IN VARCHAR2,
    p_nacionalidad_empleado IN VARCHAR2, p_estado_empleado IN CHAR,
    p_cursor OUT SYS_REFCURSOR
) AS
BEGIN
    p_cursor := C##finnk.AgregarEmpleado(
        p_nombre_empleado, p_apellidos_empleado,
        p_correo_empleado, p_telefono_empleado,
        p_salario_empleado, p_puesto_empleado,
        p_nacionalidad_empleado, p_estado_empleado
    );
END;
END PCK_FINNK_EMPLEADOS_AGREGAR;
```

Figura 72. Paquete obtener productos

Fuente: Propia

### Paquete 9:

```
-- Paquete 9
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_RECLAMOS_OBTENER AS
    PROCEDURE ObtenerReclamosSP(p_cursor OUT SYS_REFCURSOR);
    FUNCTION ObtenerReclamos RETURN SYS_REFCURSOR;
END PCK_FINNK_RECLAMOS_OBTENER;

--Paquete 9 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_RECLAMOS_OBTENER AS
    PROCEDURE ObtenerReclamosSP(p_cursor OUT SYS_REFCURSOR) AS
    BEGIN
        OPEN p_cursor FOR
            SELECT * FROM C##finnk.tab_listado_reclamos;
    END;

    FUNCTION ObtenerReclamos RETURN SYS_REFCURSOR AS
        v_cursor SYS_REFCURSOR;
    BEGIN
        OPEN v_cursor FOR
            SELECT * FROM C##finnk.tab_listado_reclamos;
        RETURN v_cursor;
    END;
END PCK_FINNK_RECLAMOS_OBTENER;
```

Figura 73. Paquete obtener productos

Fuente: Propia

### Paquete 10:

```
-- Paquete 10
CREATE OR REPLACE PACKAGE C##finnk.PCK_FINNK_RECLAMOS_ELIMINAR AS
    PROCEDURE EliminarReclamoSP(p_id_reclamo IN NUMBER, p_resultado OUT SYS_REFCURSOR);
    FUNCTION EliminarReclamosPorID(p_id_reclamo IN NUMBER) RETURN SYS_REFCURSOR;
END PCK_FINNK_RECLAMOS_ELIMINAR;

--Paquete 10 body
CREATE OR REPLACE PACKAGE BODY C##finnk.PCK_FINNK_RECLAMOS_ELIMINAR AS
    PROCEDURE EliminarReclamoSP(p_id_reclamo IN NUMBER, p_resultado OUT SYS_REFCURSOR) AS
    BEGIN
        p_resultado := C##finnk.EliminarReclamosPorID(p_id_reclamo);
    END;
    FUNCTION EliminarReclamosPorID(p_id_reclamo IN NUMBER) RETURN SYS_REFCURSOR AS
        v_cursor SYS_REFCURSOR;
        v_resultado VARCHAR2(100);
    BEGIN
        BEGIN
            DELETE FROM C##finnk.tab_listado_reclamos WHERE id_reclamos = p_id_reclamo;
            COMMIT;
            v_resultado := 'Reclamo con ID ' || TO_CHAR(p_id_reclamo) || ' eliminado correctamente';
        EXCEPTION
            WHEN OTHERS THEN
                v_resultado := 'Error: ' || SQLERRM;
        END;
        OPEN v_cursor FOR
            SELECT v_resultado AS mensaje FROM dual;
        RETURN v_cursor;
    END;
END PCK_FINNK_RECLAMOS_ELIMINAR;
```

Figura 74. Paquete obtener productos

Fuente: Propia

# Triggers

Trigger 1:

```
--Trig1--  
--Verifica si el correo del cliente ya existe, si el correo ya existe,  
--el trigger presenta un mensaje de error.  
  
CREATE OR REPLACE TRIGGER C##finnk.TRIGGER_TAB_LISTADO_CLIENTES  
BEFORE INSERT ON C##finnk.tab_listado_clientes  
FOR EACH ROW  
DECLARE  
    v_correo_existente NUMBER;  
BEGIN  
    -- Verificar si el correo electrónico ya existe en la tabla  
    SELECT COUNT(*) INTO v_correo_existente FROM C##finnk.tab_listado_clientes WHERE correo_cliente = :NEW.correo_cliente;  
  
    IF v_correo_existente > 0 THEN  
        -- Mostrar un mensaje de error si el correo electrónico ya existe  
        raise_application_error(-20001, 'El correo electrónico ya está registrado en la tabla.');
```

Figura 75. Trigger obtener Nombre de proveedor por ID

Fuente: Propia

Trigger 2:

```
--Trig2--  
--Muestra la informacion realizada de insercion, actualizacion o eliminacion y el ID del producto deseado.  
CREATE OR REPLACE TRIGGER C##finnk.TRIGGER_TAB_CATALOGO_PRODUCTOS  
AFTER INSERT OR UPDATE OR DELETE ON C##finnk.tab_catalogo_productos  
FOR EACH ROW  
BEGIN  
    IF INSERTING THEN  
        DBMS_OUTPUT.PUT_LINE('Se ha insertado un nuevo producto con ID: ' || :NEW.id_catalogo);  
    ELSIF UPDATING THEN  
        DBMS_OUTPUT.PUT_LINE('Se ha actualizado el producto con ID: ' || :NEW.id_catalogo);  
    ELSIF DELETING THEN  
        DBMS_OUTPUT.PUT_LINE('Se ha eliminado el producto con ID: ' || :OLD.id_catalogo);  
    END IF;  
END;
```

Figura 76. Trigger obtener Nombre de proveedor por ID

Fuente: Propia

Trigger 3:

```
--Trig3--  
--Muestra la informacion sobre el proveedor insertado con su ID, nombre, apellidos, correo, telefono, marca y nacionalidad.  
CREATE OR REPLACE TRIGGER C##finnk.TRIGGER_INSERT_PROVEEDOR  
AFTER INSERT ON C##finnk.tab_listado_proveedores  
FOR EACH ROW  
BEGIN  
    DBMS_OUTPUT.PUT_LINE('Se ha insertado un nuevo proveedor con ID: ' || :NEW.id_proveedor);  
    DBMS_OUTPUT.PUT_LINE('Nombre del proveedor: ' || :NEW.nombre_proveedor || ' ' || :NEW.apellidos_proveedor);  
    DBMS_OUTPUT.PUT_LINE('Correo del proveedor: ' || :NEW.correo_proveedor);  
    DBMS_OUTPUT.PUT_LINE('Teléfono del proveedor: ' || :NEW.telefono_proveedor);  
    DBMS_OUTPUT.PUT_LINE('Marca del proveedor: ' || :NEW.marca_proveedor);  
    DBMS_OUTPUT.PUT_LINE('Nacionalidad del proveedor: ' || :NEW.nacionalidad_proveedor);  
END;
```

Figura 77. Trigger obtener Nombre de proveedor por ID

Fuente: Propia

#### Trigger 4:

```
--Trig4
--Muestra la informacion del empleado insertado con incluyendo su ID, nombre, apellidos, correo, telefono,
--salario, puesto, nacionalidad y el ID de la tienda.
CREATE OR REPLACE TRIGGER C##finnk.TRIGGER_INSERT_EMPLEADO
AFTER INSERT ON C##finnk.tab_listado_empleados
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('Se ha insertado un nuevo empleado con ID: ' || :NEW.id_empleado);
    DBMS_OUTPUT.PUT_LINE('Nombre del empleado: ' || :NEW.nombre_empleado || ' ' || :NEW.apellidos_empleado);
    DBMS_OUTPUT.PUT_LINE('Correo del empleado: ' || :NEW.correo_empleado);
    DBMS_OUTPUT.PUT_LINE('Teléfono del empleado: ' || :NEW.telefono_empleado);
    DBMS_OUTPUT.PUT_LINE('Salario del empleado: ' || :NEW.salario_empleado);
    DBMS_OUTPUT.PUT_LINE('Puesto del empleado: ' || :NEW.puesto_empleado);
    DBMS_OUTPUT.PUT_LINE('Nacionalidad del empleado: ' || :NEW.nacionalidad_empleado);
    DBMS_OUTPUT.PUT_LINE('ID de la tienda a la que pertenece: ' || :NEW_fk_tienda);
END;
```

Figura 78. Trigger obtener Nombre de proveedor por ID

Fuente: Propia

#### Trigger 5:

```
--Trig5
--Verifica que la localidad de la tienda sea correcta y el estado de la tienda este en 'Y' o 'N'.
CREATE OR REPLACE TRIGGER C##finnk.TRIGGER_TIENDA
BEFORE INSERT ON C##finnk.tab_listado_tienda
FOR EACH ROW
BEGIN
    IF :NEW.localidad_tienda IS NULL OR LENGTH(:NEW.localidad_tienda) > 40 THEN
        RAISE_APPLICATION_ERROR(-20001, 'La localidad de la tienda no es válida.');
    END IF;
    IF :NEW.estado_tienda NOT IN ('Y', 'N') THEN
        RAISE_APPLICATION_ERROR(-20002, 'El estado de la tienda debe ser "Y" o "N".');
    END IF;
END;
```

Figura 79. Trigger obtener Nombre de proveedor por ID

Fuente: Propia

## Cursos

Cursor 1:

```
--cursor 1

DECLARE
  CURSOR correos_clientes_cursor IS
    SELECT correo_cliente FROM C##finnk.tab_listado_clientes c;
    correo_cliente C##finnk.tab_listado_clientes.correo_cliente%TYPE;
BEGIN
  OPEN correos_clientes_cursor;
  LOOP
    FETCH correos_clientes_cursor INTO correo_cliente;
    EXIT WHEN correos_clientes_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Correo cliente: ' || correo_cliente);
  END LOOP;
  CLOSE correos_clientes_cursor;
END;
```

Figura 80. Cursor correos de clientes

Fuente: Propia

Cursor 2:

```
--cursor 2

DECLARE
  CURSOR productos_cursor IS
    SELECT id_catalogo, imagen_producto, nombre_producto, precio_producto, existencias_producto, estado_producto
    FROM C##finnk.tab_catalogo_productos
    WHERE precio_producto <= 10000;
    producto_rec productos_cursor%ROWTYPE;
BEGIN
  OPEN productos_cursor;
  LOOP
    FETCH productos_cursor INTO producto_rec;
    EXIT WHEN productos_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || producto_rec.id_catalogo ||
      ', Imagen: ' || producto_rec.imagen_producto ||
      ', Nombre: ' || producto_rec.nombre_producto ||
      ', Precio: ' || producto_rec.precio_producto ||
      ', Existencias: ' || producto_rec.existencias_producto ||
      ', Estado: ' || producto_rec.estado_producto);
  END LOOP;
  CLOSE productos_cursor;
END;
```

Figura 81. Cursor productos

Fuente: Propia

Cursor 3:

```
--cursor 3
DECLARE
CURSOR empleados_cursor IS
    SELECT id_empleado, nombre_empleado, apellidos_empleado, correo_empleado, telefono_empleado,
    salario_empleado, puesto_empleado, nacionalidad_empleado, estado_empleado
    FROM C##finnk.tab_listado_empleados
    WHERE estado_empleado = 'N';
    empleado_rec empleados_cursor%ROWTYPE;
BEGIN
OPEN empleados_cursor;
LOOP
    FETCH empleados_cursor INTO empleado_rec;
    EXIT WHEN empleados_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || empleado_rec.id_empleado ||
        ', Nombre: ' || empleado_rec.nombre_empleado ||
        ', Apellidos: ' || empleado_rec.apellidos_empleado ||
        ', Correo: ' || empleado_rec.correo_empleado ||
        ', Teléfono: ' || empleado_rec.telefono_empleado ||
        ', Salario: ' || empleado_rec.salario_empleado ||
        ', Puesto: ' || empleado_rec.puesto_empleado ||
        ', Nacionalidad: ' || empleado_rec.nacionalidad_empleado ||
        ', Estado: ' || empleado_rec.estado_empleado);
END LOOP;
CLOSE empleados_cursor;
END;
```

Figura 82. Cursor lista empleados

Fuente: Propia

#### Cursor 4:

```
--cursor 4
DECLARE
CURSOR empleados_cursor IS
    SELECT id_empleado, nombre_empleado, apellidos_empleado, correo_empleado,
    telefono_empleado, salario_empleado, puesto_empleado, nacionalidad_empleado, estado_empleado
    FROM C##finnk.tab_listado_empleados
    WHERE salario_empleado >=250000 ;
    empleado_rec empleados_cursor%ROWTYPE;
BEGIN
OPEN empleados_cursor;
LOOP
    FETCH empleados_cursor INTO empleado_rec;
    EXIT WHEN empleados_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || empleado_rec.id_empleado ||
        ', Nombre: ' || empleado_rec.nombre_empleado ||
        ', Apellidos: ' || empleado_rec.apellidos_empleado ||
        ', Correo: ' || empleado_rec.correo_empleado ||
        ', Teléfono: ' || empleado_rec.telefono_empleado ||
        ', Salario: ' || empleado_rec.salario_empleado ||
        ', Puesto: ' || empleado_rec.puesto_empleado ||
        ', Nacionalidad: ' || empleado_rec.nacionalidad_empleado ||
        ', Estado: ' || empleado_rec.estado_empleado);
END LOOP;
CLOSE empleados_cursor;
END;
```

Figura 83. Cursor salario empleados

Fuente: Propia

#### Cursor 5:

```
--cursor 5
DECLARE
    CURSOR productos_cursor IS
        SELECT id_catalogo, imagen_producto, nombre_producto, precio_producto, existencias_producto, estado_producto
        FROM C##finnk.tab_catalogo_productos
        WHERE existencias_producto >= 10;
    BEGIN
        FOR producto_rec IN productos_cursor
        LOOP
            DBMS_OUTPUT.PUT_LINE('ID: ' || producto_rec.id_catalogo ||
                ', Imagen: ' || producto_rec.imagen_producto ||
                ', Nombre: ' || producto_rec.nombre_producto ||
                ', Precio: ' || producto_rec.precio_producto ||
                ', Existencias: ' || producto_rec.existencias_producto ||
                ', Estado: ' || producto_rec.estado_producto);
        END LOOP;
    END;
```

Figura 84. Cursor existencias productos

Fuente: Propia

#### Cursor 6:

```
--cursor 6
DECLARE
    CURSOR proveedores_cursor IS
        SELECT id_proveedor, nombre_proveedor, apellidos_proveedor, correo_proveedor,
        telefono_proveedor, marca_proveedor, nacionalidad_proveedor, estado_proveedor
        FROM C##finnk.tab_listado_proveedores
        WHERE nacionalidad_proveedor = 'Costarricense';
    proveedor_rec proveedores_cursor%ROWTYPE;
BEGIN
    OPEN proveedores_cursor;

    FETCH proveedores_cursor INTO proveedor_rec;
    WHILE proveedores_cursor%FOUND
    LOOP
        DBMS_OUTPUT.PUT_LINE('ID: ' || proveedor_rec.id_proveedor ||
            ', Nombre: ' || proveedor_rec.nombre_proveedor ||
            ', Apellidos: ' || proveedor_rec.apellidos_proveedor ||
            ', Correo: ' || proveedor_rec.correo_proveedor ||
            ', Teléfono: ' || proveedor_rec.telefono_proveedor ||
            ', Marca: ' || proveedor_rec.marca_proveedor ||
            ', Nacionalidad: ' || proveedor_rec.nacionalidad_proveedor ||
            ', Estado: ' || proveedor_rec.estado_proveedor);

        FETCH proveedores_cursor INTO proveedor_rec;
    END LOOP;
    CLOSE proveedores_cursor;
END;
```

Figura 85. Cursor lista proveedores

Fuente: Propia

#### Cursor 7:

```
-- Cursor 7: Lista las nacionalidades unicas de los empleados

DECLARE
    CURSOR empleados_nacionalidad_cursor IS
        SELECT DISTINCT nacionalidad_empleado
        FROM C##finnk.tab_listado_empleados;
        nacionalidad_empleado C##finnk.tab_listado_empleados.nacionalidad_empleado%TYPE;
BEGIN
    OPEN empleados_nacionalidad_cursor;
    LOOP
        FETCH empleados_nacionalidad_cursor INTO nacionalidad_empleado;
        EXIT WHEN empleados_nacionalidad_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Nacionalidad empleado: ' || nacionalidad_empleado);
    END LOOP;
    CLOSE empleados_nacionalidad_cursor;
END;
```

Figura 86. Cursor nacionalidades únicas empleados

Fuente: Propia

## Cursor 8:

```
-- Cursor 8: Lista los productos con bajo stock (menos de 5 unidades en stock)

DECLARE
    CURSOR productos_bajo_stock_cursor IS
        SELECT id_catalogo, nombre_producto, existencias_producto
        FROM C##finnk.tab_catalogo_productos
        WHERE existencias_producto < 5;
        producto_bajo_stock_rec productos_bajo_stock_cursor%ROWTYPE;
BEGIN
    OPEN productos_bajo_stock_cursor;
    LOOP
        FETCH productos_bajo_stock_cursor INTO producto_bajo_stock_rec;
        EXIT WHEN productos_bajo_stock_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('ID: ' || producto_bajo_stock_rec.id_catalogo ||
                            ', Nombre: ' || producto_bajo_stock_rec.nombre_producto ||
                            ', Existencias: ' || producto_bajo_stock_rec.existencias_producto);
    END LOOP;
    CLOSE productos_bajo_stock_cursor;
END;
```

Figura 87. Cursor productos poco stock

Fuente: Propia

## Cursor 9:

```
-- Cursor 9: Lista los correos de los proveedores

DECLARE
    CURSOR proveedores_correo_cursor IS
        SELECT correo_proveedor
        FROM C##finnk.tab_listado_proveedores;
        correo_proveedor C##finnk.tab_listado_proveedores.correo_proveedor%TYPE;
BEGIN
    OPEN proveedores_correo_cursor;
    LOOP
        FETCH proveedores_correo_cursor INTO correo_proveedor;
        EXIT WHEN proveedores_correo_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Correo del proveedor: ' || correo_proveedor);
    END LOOP;
    CLOSE proveedores_correo_cursor;
END;
```

Figura 88. Cursor lista correos de proveedores

Fuente: Propia

Cursor 10:

```
-- Cursor 10: Lista los productos inactivos en el catalogo
DECLARE
    CURSOR productos_inactivos_cursor IS
        SELECT id_catalogo, nombre_producto, estado_producto
        FROM C##finnk.tab_catalogo_productos
        WHERE estado_producto = 'Inactivo';
    producto_inactivo_rec productos_inactivos_cursor%ROWTYPE;
BEGIN
    OPEN productos_inactivos_cursor;
    LOOP
        FETCH productos_inactivos_cursor INTO producto_inactivo_rec;
        EXIT WHEN productos_inactivos_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('ID: ' || producto_inactivo_rec.id_catalogo ||
                            ', Nombre: ' || producto_inactivo_rec.nombre_producto ||
                            ', Estado: ' || producto_inactivo_rec.estado_producto);
    END LOOP;
    CLOSE productos_inactivos_cursor;
END;
```

Figura 89. Cursor productos inactivos

Fuente: Propia

Cursor 11:

```
-- Cursor 11: Lista los nombres de los proveedores y su nacionalidad
DECLARE
    CURSOR proveedores_nacionalidad_cursor IS
        SELECT DISTINCT nacionalidad_proveedor
        FROM C##finnk.tab_listado_proveedores;
    nacionalidad_proveedor C##finnk.tab_listado_proveedores.nacionalidad_proveedor%TYPE;
BEGIN
    OPEN proveedores_nacionalidad_cursor;
    LOOP
        FETCH proveedores_nacionalidad_cursor INTO nacionalidad_proveedor;
        EXIT WHEN proveedores_nacionalidad_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Nacionalidad proveedor: ' || nacionalidad_proveedor);
    END LOOP;
    CLOSE proveedores_nacionalidad_cursor;
END;
```

Figura 90. Cursor nombres y nacionalidad proveedores

Fuente: Propia

Cursor 12:

```

-- Cursor 12: Lista los empleados con salarios mas altos
DECLARE
  CURSOR empleados_salarios_altos_cursor IS
    SELECT id_empleado, nombre_empleado, apellidos_empleado, salario_empleado
    FROM C##finnk.tab_listado_empleados
    WHERE salario_empleado >= (SELECT MAX(salario_empleado) FROM C##finnk.tab_listado_empleados);
  empleado_salario_alto_rec empleados_salarios_altos_cursor%ROWTYPE;
BEGIN
  OPEN empleados_salarios_altos_cursor;
  LOOP
    FETCH empleados_salarios_altos_cursor INTO empleado_salario_alto_rec;
    EXIT WHEN empleados_salarios_altos_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || empleado_salario_alto_rec.id_empleado ||
      ', Nombre: ' || empleado_salario_alto_rec.nombre_empleado ||
      ', Apellidos: ' || empleado_salario_alto_rec.apellidos_empleado ||
      ', Salario: ' || empleado_salario_alto_rec.salario_empleado);
  END LOOP;
  CLOSE empleados_salarios_altos_cursor;
END;

```

Figura 91. Cursor salarios más altos

Fuente: Propia

### Cursor 13:

```

-- Cursor 13: Lista los productos activos con existencias mayores o iguales a 50
DECLARE
  CURSOR productos_activos_existencias_cursor IS
    SELECT id_catalogo, nombre_producto, existencias_producto
    FROM C##finnk.tab_catalogo_productos
    WHERE estado_producto = 'Activo' AND existencias_producto >= 50;
  producto_activo_existencias_rec productos_activos_existencias_cursor%ROWTYPE;
BEGIN
  OPEN productos_activos_existencias_cursor;
  LOOP
    FETCH productos_activos_existencias_cursor INTO producto_activo_existencias_rec;
    EXIT WHEN productos_activos_existencias_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || producto_activo_existencias_rec.id_catalogo ||
      ', Nombre: ' || producto_activo_existencias_rec.nombre_producto ||
      ', Existencias: ' || producto_activo_existencias_rec.existencias_producto);
  END LOOP;
  CLOSE productos_activos_existencias_cursor;
END;

```

Figura 92. Cursor productos con mas de 50 unidades en stock

Fuente: Propia

### Cursor 14:

```

-- Cursor 14: Lista los empleados con salario entre 200000 y 300000
DECLARE
  CURSOR empleados_salario_rango_cursor IS
    SELECT id_empleado, nombre_empleado, apellidos_empleado, salario_empleado
    FROM C##finnk.tab_listado_empleados
    WHERE salario_empleado BETWEEN 200000 AND 300000;
  empleado_salario_rango_rec empleados_salario_rango_cursor%ROWTYPE;
BEGIN
  OPEN empleados_salario_rango_cursor;
  LOOP
    FETCH empleados_salario_rango_cursor INTO empleado_salario_rango_rec;
    EXIT WHEN empleados_salario_rango_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || empleado_salario_rango_rec.id_empleado ||
      ', Nombre: ' || empleado_salario_rango_rec.nombre_empleado ||
      ', Apellidos: ' || empleado_salario_rango_rec.apellidos_empleado ||
      ', Salario: ' || empleado_salario_rango_rec.salario_empleado);
  END LOOP;
  CLOSE empleados_salario_rango_cursor;
END;

```

Figura 93. Cursor rango salarial

Fuente: Propia

Cursor 15:

```
-- Cursor 15: Lista los clientes con correo que incluya "gmail.com"

DECLARE
    CURSOR clientes_gmail_cursor IS
        SELECT correo_cliente
        FROM C##finnk.tab_listado_clientes
        WHERE correo_cliente LIKE '%gmail.com';
    correo_cliente_gmail C##finnk.tab_listado_clientes.correo_cliente%TYPE;
BEGIN
    OPEN clientes_gmail_cursor;
    LOOP
        FETCH clientes_gmail_cursor INTO correo_cliente_gmail;
        EXIT WHEN clientes_gmail_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Correo cliente con Gmail: ' || correo_cliente_gmail);
    END LOOP;
    CLOSE clientes_gmail_cursor;
END;
```

Figura 90. Cursor correos de cliente con gmail

Fuente: Propia

## Diccionario de Datos

En este se encuentra información como los privilegios que se obtiene del usuario, restricciones de integridad definidas.

Como primer paso tenemos el Generate DB Doc...

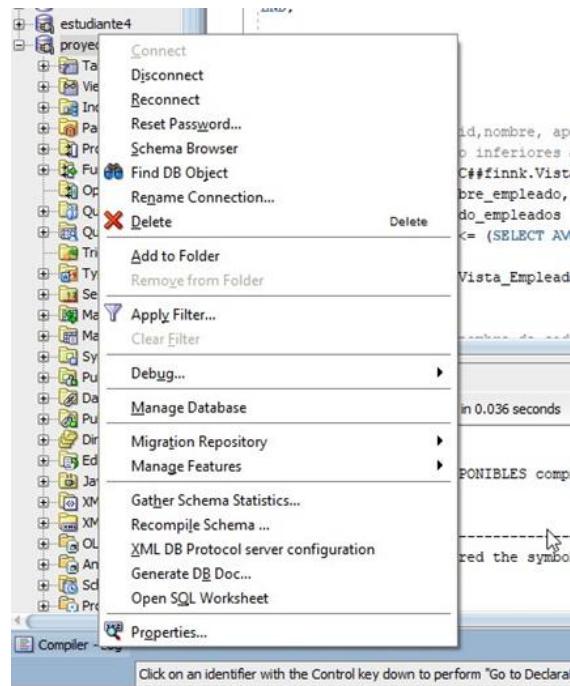


Figura 91. Generate DB Doc

Fuente: Propia

A la derecha se seleccionarán las columnas que se desean presentar y adquirir.

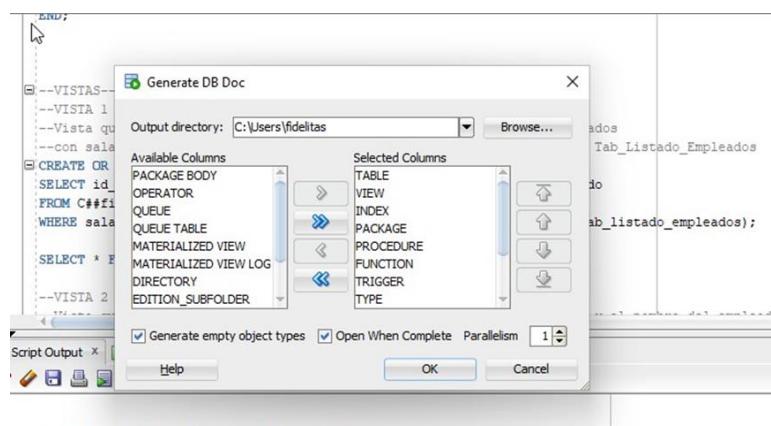


Figura 92. Columnas  
Fuente: Propia

Una vez se cumpla esta serie de pasos se obtendrá de manera correcta el diccionario de datos.

The screenshot shows the Oracle Database SQL Developer interface. On the left, there's a sidebar with various database objects like Tables, Views, Procedures, Functions, Triggers, and Types. Under 'Tables', 'TAB\_CATALOGO\_PRODUCTOS' is selected. The main area displays the table structure with the following columns:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
ID_CATALOGO	NUMBER	No	"C##FINNK"."SEQ\$\$.73954".nextval	1	null
IMAGEN_PRODUCTO	VARCHAR2(200 BYTE)	No	null	2	null
NOMBRE_PRODUCTO	VARCHAR2(40 BYTE)	No	null	3	null
PRECIO_PRODUCTO	NUMBER	No	null	4	null
EXISTENCIAS_PRODUCTO	NUMBER	No	null	5	null
ESTADO_PRODUCTO	CHAR(1 BYTE)	No	null	6	null

Figura 93. Diccionario de datos  
Fuente: Propia

## **Programación del Proyecto**

Esta clase, llamada `ProveedorServiceImpl`, implementa la interfaz `ProveedorService` y se encarga de manejar la lógica relacionada con los proveedores en un sistema. Aquí se utilizan llamadas a procedimientos almacenados en una base de datos Oracle para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en la entidad "Proveedor".

En el método `getProveedors()`, se establece una conexión a la base de datos Oracle. Luego, se invoca un procedimiento almacenado para obtener una lista de proveedores desde la base de datos y se almacenan en objetos `Proveedor`. Estos objetos se agregan a una lista que se devuelve al final del método.

El método `getProveedor(Proveedor proveedor)` toma un objeto `Proveedor` como entrada y busca ese proveedor en la lista obtenida del método anterior. Si lo encuentra, devuelve los detalles de ese proveedor.

`deleteProveedor(Proveedor proveedor)` realiza la eliminación de un proveedor. Utiliza un procedimiento almacenado que toma el ID del proveedor a eliminar y, tras ejecutarlo, muestra un mensaje de éxito si la operación fue exitosa.

`saveProveedor(Proveedor proveedor)` realiza la inserción de un nuevo proveedor en la base de datos. Se llama a un procedimiento almacenado que toma los datos del proveedor como parámetros y, después de ejecutarlo, muestra un mensaje con el ID asignado al nuevo proveedor.

editarProveedor(Proveedor proveedor) permite actualizar los detalles de un proveedor. Llama a un procedimiento almacenado que toma los nuevos valores de los detalles del proveedor y los aplica en la base de datos

```

1 package com.proyecto.serviceImpl;
2
3 import com.proyecto.dao.ProveedorDao;
4 import com.proyecto.domain.Proveedor;
5 import com.proyecto.service.ProveedorService;
6 import java.sql.CallableStatement;
7 import java.sql.Connection;
8 import java.sql.DriverManager;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11 import java.util.ArrayList;
12 import java.util.List;
13 import oracle.jdbc.OracleTypes;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.stereotype.Service;
16
17 @Service
18 public class ProveedorServiceImpl implements ProveedorService{
19
20     //Esto crea una unica copia en un objeto //
21     @Autowired
22     public ProveedorDao proveedorDao;
23
24
25     @Override
26     public List<Proveedor> getProveedores() {
27         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
28         List<Proveedor> lista = new ArrayList<>();
29
30         try (Connection conexion = DriverManager.getConnection(jdbcUrl, "nicole", "987654")) {
31             System.out.println("Conectado a la base de datos");
32
33             String callStatement = "{ call C##finnk.obtener_proveedores_sp(? ) }";
34
35             try (CallableStatement llamadaExecute = conexion.prepareCall(callStatement)) {
36                 llamadaExecute.registerOutParameter(parameterIndex: 1, sqlType: OracleTypes.CURSOR);
37
38                 llamadaExecute.execute();
39                 ResultSet resultSet = (ResultSet) llamadaExecute.getObject(parameterIndex: 1);
40
41                 while (resultSet.next()) {
42                     Proveedor proveedor = new Proveedor();
43                     proveedor.setIdProveedor(resultSet.getLong(columnLabel: "id_proveedor"));
44                     proveedor.setNombre(resultSet.getString(columnLabel: "nombre_proveedor"));
45                     proveedor.setApellidos(resultSet.getString(columnLabel: "apellidos_proveedor"));
46                     proveedor.setCorreo(resultSet.getString(columnLabel: "correo_proveedor"));
47                     proveedor.setTelefono(resultSet.getInt(columnLabel: "telefono_proveedor"));
48                     proveedor.setMarca(resultSet.getString(columnLabel: "marca_proveedor"));
49                     proveedor.setNacionalidad(resultSet.getString(columnLabel: "nacionalidad_proveedor"));
50                     proveedor.setEstado(resultSet.getString(columnLabel: "estado_proveedor"));
51
52                     lista.add(e: proveedor);
53                 }
54                 System.out.println("Lectura de proveedores realizada correctamente");
55
56                 resultSet.close();
57             }
58         } catch (SQLException e) {
59             System.out.println("Error");
60             e.printStackTrace();
61         }
62
63         return lista;
64     }
65
66     @Override
67     public Proveedor getProveedor(Proveedor proveedor) {
68
69
70         List<Proveedor> proveedores=getProveedores();
71         Proveedor proveedorRetorno = new Proveedor();
72         int idProveedor=proveedor.getIdProveedor().intValue();
73
74
75         for (int i=0; i<proveedores.size(); i++){
76
77             if (proveedores.get(index: i).getIdProveedor()==idProveedor){
78                 System.out.println(idProveedor+"ID a editar");
79
80                 System.out.println("I del producto del bucle"+proveedores.get(index: i).getIdProveedor());
81                 proveedorRetorno=proveedores.get(index: i);
82
83                 break;
84             }
85         }
86
87     }
88     return proveedorRetorno;
89 }
90
91     @Override
92
93     public void deleteProveedor(Proveedor proveedor) {
94         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
95         String callStatement = "{ call C##finnk.EliminarProveedorSP(?, ? ) }";
96         String resultado = null;

```

Figura 94. Proveedor

Fuente: Propia

```

98     try (Connection conexion = DriverManager.getConnection(url, user: "nicole", password: "987654")) {
99         CallableStatement callableStatement = conexion.prepareCall(sql.callStatement);
100        callableStatement.setLong(parameterIndex: 1, : proveedor.getIdProveedor());
101        callableStatement.registerOutParameter(parameterIndex: 2, sqlType: OracleTypes.CURSOR);
102
103        callableStatement.execute();
104
105        ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 2);
106
107        if (resultSet.next()) {
108            resultado = resultSet.getString(columnLabel: "mensaje");
109            System.out.println(resultado + " EXITO");
110        }
111
112        resultSet.close();
113    } catch (SQLException e) {
114        System.out.println("Error detected");
115        e.printStackTrace();
116    }
117 }
118
119
120 @Override
121 public void saveProveedor(Proveedor proveedor) {
122     String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
123
124     try (Connection connection = DriverManager.getConnection(url:jdbcUrl, user: "C##finnk", password: "kaws2023")) {
125         String callStatement = "( call AgregarProveedorSP(?, ?, ?, ?, ?, ?, ?) )";
126
127         try (CallableStatement callableStatement = connection.prepareCall(sql:callStatement)) {
128             callableStatement.setString(parameterIndex: 1, : proveedor.getNombre());
129             callableStatement.setString(parameterIndex: 2, : proveedor.getApellidos());
130             callableStatement.setString(parameterIndex: 3, : proveedor.getCorreo());
131             callableStatement.setInt(parameterIndex: 4, : proveedor.getTelefono());
132             callableStatement.setString(parameterIndex: 5, : proveedor.getMarca());
133             callableStatement.setString(parameterIndex: 6, : proveedor.getNacionalidad());
134             callableStatement.setString(parameterIndex: 7, : proveedor.getEstado());
135             callableStatement.registerOutParameter(parameterIndex: 8, sqlType: OracleTypes.CURSOR);
136
137             callableStatement.execute();
138
139             ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 8);
140
141             if (resultSet.next()) {
142                 System.out.println("Proveedor agregado correctamente. ID: " + resultSet.getInt(columnLabel: "id_proveedor"));
143             }
144
145             resultSet.close();
146
147         }
148
149     } catch (SQLException e) {
150         System.out.println("Error detected");
151         e.printStackTrace();
152     }
153 }
154
155
156
157 @Override
158 public void editarProveedor(Proveedor proveedor) {
159     String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
160
161     try (Connection connection = DriverManager.getConnection(url:jdbcUrl, user: "nicole", password: "987654")) {
162         System.out.println("Connected to the Oracle database");
163
164         String callStatement = "( call C##finnk.EditarProveedorSP(?, ?, ?, ?, ?, ?, ?, ?, ?) )";
165
166         try (CallableStatement callableStatement = connection.prepareCall(sql:callStatement)) {
167             callableStatement.setInt(parameterIndex: 1, : proveedor.getIdProveedor().intValue());
168             callableStatement.setString(parameterIndex: 2, : proveedor.getNombre());
169             callableStatement.setString(parameterIndex: 3, : proveedor.getApellidos());
170             callableStatement.setString(parameterIndex: 4, : proveedor.getCorreo());
171             callableStatement.setInt(parameterIndex: 5, : proveedor.getTelefono());
172             callableStatement.setString(parameterIndex: 6, : proveedor.getMarca());
173             callableStatement.setString(parameterIndex: 7, : proveedor.getNacionalidad());
174             callableStatement.setString(parameterIndex: 8, : proveedor.getEstado());
175
176             // Registro del parámetro de salida para el cursor
177             callableStatement.registerOutParameter(parameterIndex: 9, sqlType: OracleTypes.CURSOR);
178
179             callableStatement.execute();
180             System.out.println("Edicion de proveedor completada.");
181
182             // Obtener el resultado del cursor
183             ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 9);
184             while (resultSet.next()) {
185                 // Procesar los resultados si es necesario
186             }
187         }
188     } catch (SQLException e) {
189         System.out.println("Error detected");
190         e.printStackTrace();
191     }
192 }

```

Figura 95. Proveedor

Fuente: Propia

Esta clase, llamada ProductoServiceImpl, implementa la interfaz ProductoService y se encarga de manejar la lógica relacionada con los productos en un sistema. Similar al código previo, aquí también se utilizan procedimientos almacenados en una base de datos Oracle para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en la entidad "Producto".

El método getProductos() establece una conexión a la base de datos Oracle y ejecuta un procedimiento almacenado para obtener una lista de productos. Los detalles de cada producto, como su imagen, nombre, precio, existencias y estado, se extraen del ResultSet y se almacenan en objetos Producto. Estos objetos se agregan a una lista que se devuelve al final del método.

El método getProducto(Producto producto) toma un objeto Producto como entrada y busca ese producto en la lista obtenida del método anterior. Si lo encuentra, devuelve los detalles de ese producto.

deleteProducto(Producto producto) realiza la eliminación de un producto. Utiliza un procedimiento almacenado que toma el ID del producto a eliminar y, tras ejecutarlo, muestra un mensaje de éxito si la operación fue exitosa.

saveProducto(Producto producto) realiza la inserción de un nuevo producto en la base de datos. Se llama a un procedimiento almacenado que toma los datos del producto como parámetros y, después de ejecutarlo, muestra un mensaje de éxito.

`editarProducto(Producto producto)` permite actualizar los detalles de un producto.

Llama a un procedimiento almacenado que toma los nuevos valores de los detalles del producto y los aplica en la base de datos.

```

1 package com.proyecto.service.Impl;
2
3 import com.proyecto.dao.ProductoDao;
4 import com.proyecto.domain.Producto;
5 import com.proyecto.service.ProductoService;
6 import java.sql.CallableStatement;
7 import java.sql.Connection;
8 import java.sql.DriverManager;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11 import java.sql.Statement;
12 import java.util.ArrayList;
13 import java.util.List;
14 import oracle.jdbc.OracleTypes;
15
16 import org.springframework.beans.factory.annotation.Autowired;
17 import org.springframework.stereotype.Service;
18
19 @Service
20 public class ProductoServiceImpl implements ProductoService {
21
22     //Esto crea una unica copia en un objeto //
23     @Autowired
24     public ProductoDao productoDao;
25
26     @Override
27     public List<Producto> getProductos() {
28
29         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
30
31         List<Producto> lista = new ArrayList<>();
32
33         try (Connection conexion = DriverManager.getConnection(url:jdbcUrl, user: "nicole", password: "987654321")) {
34             System.out.println("Conectado a la base de datos");
35
36             String callStatement = "{ call C##finnk.PCK_FINNK_PRODUCTOS_OBTENER.ObtenerProductosSP(?)
37             System.out.println("Paquete preparado");
38
39             // Prepara la llamada al execute
40             try (CallableStatement llamadaExecute = conexion.prepareCall(sql:callStatement)) {
41
42                 llamadaExecute.registerOutParameter(parameterIndex: 1, sqlType:OracleTypes.CURSOR);
43
44                 // Ejecuta el store procedure
45                 llamadaExecute.execute();
46
47                 // Recive resultados y utiliza una libreria llamada resultset
48                 ResultSet resultSet = (ResultSet) llamadaExecute.getObject(parameterIndex: 1);
49
50
51                 while (resultSet.next()) {
52                     Producto producto = new Producto();
53                     producto.setIdProducto(idproducto: resultSet.getLong(columnLabel:"id_catalogo"));
54                     producto.setImagen(imagen: resultSet.getString(columnLabel:"imagen_producto"));
55                     producto.setNombre(nombre: resultSet.getString(columnLabel:"nombre_producto"));
56                     producto.setPrecio(precio: resultSet.getInt(columnLabel:"precio_producto"));
57                     producto.setExistencias(existencias: resultSet.getInt(columnLabel:"existencias_producto"));
58                     producto.setEstado(estado: resultSet.getString(columnLabel:"estado_producto"));
59
60                     System.out.println(producto.getImagen());
61                     lista.add(producto);
62                 }
63
64                 resultSet.close();
65
66             }
67         } catch (SQLException e) {
68             System.out.println("Error");
69             e.printStackTrace();
70         }
71
72         return lista;
73     }
74
75     @Override
76     public Producto getProducto(Producto producto) {
77
78
79         List<Producto> productos= getProductos();
80         Producto productoRetorno = new Producto();
81         int idProducto=producto.getIdProducto().intValue();
82
83
84         for (int i=0; i<productos.size(); i++) {
85
86             if (productos.get(index: i).getIdProducto()==idProducto) {
87
88
89
90

```

Figura 96. Productos

Fuente: Propia

```

90         if (productos.get(index: i).getIdProducto() == idProducto) {
91             System.out.println(idProducto + " ID a editar");
92
93             System.out.println("I del producto del bucle" + productos.get(index: i).getIdProducto());
94             productoRetorno = productos.get(index: i);
95
96             break;
97         }
98     }
99 }
100
101 return productoRetorno;
102 }
103
104 @Override
105
106     public void deleteProducto(Producto producto) {
107         Long id = producto.getIdProducto();
108         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
109         String callStatement = "( call C##finnk.PCK_FINNK_PRODUCTOS_ELIMINAR.EliminarProductoSP(?) )";
110         System.out.println("Eliminacion preparada con paquete");
111
112         Connection conexion = DriverManager.getConnection(url: jdbcUrl, user: "nicole", password: "9876");
113         CallableStatement callableStatement = conexion.prepareCall(sql: callStatement);
114         int pid = producto.getIdProducto().intValue();
115         callableStatement.setInt(parameterIndex: 1, value: pid);
116         callableStatement.execute();
117
118         System.out.println("StoredProcedure executed successfully.");
119     } catch (SQLException e) {
120         System.out.println("Error detected");
121         e.printStackTrace();
122     }
123 }
124
125 @Override
126     public void saveProducto(Producto producto) {
127
128         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
129
130
131         try (Connection connection = DriverManager.getConnection(url: jdbcUrl, user: "nicole", password: "8876")) {
132             System.out.println("Connected to the Oracle database");
133
134             String callStatement = "( call C##finnk.PCK_FINNK_PRODUCTOS_AGREGAR.AgregarProductoSP(?, ?) )";
135             System.out.println("Agregar mediante paquete");
136             try (CallableStatement callableStatement = connection.prepareCall(sql: callStatement)) {
137
138                 callableStatement.setString(parameterIndex: 1, value: producto.getImagen());
139                 callableStatement.setString(parameterIndex: 2, value: producto.getNombre());
140                 callableStatement.setDouble(parameterIndex: 3, value: producto.getPrecio());
141                 callableStatement.setInt(parameterIndex: 4, value: producto.getExistencias());
142                 callableStatement.setString(parameterIndex: 5, value: producto.getEstado());
143
144                 callableStatement.execute();
145                 System.out.println("StoredProcedure de creacion ejecutado exitosamente.");
146             } catch (SQLException e) {
147                 System.out.println("Error detected");
148                 e.printStackTrace();
149             }
150         }
151
152     }
153
154 @Override
155     public void editarProducto(Producto producto) {
156
157         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
158
159
160         try (Connection connection = DriverManager.getConnection(url: jdbcUrl, user: "nicole", password: "8876")) {
161             System.out.println("Connected to the Oracle database");
162
163             String callStatement = "( call C##finnk.PCK_FINNK_PRODUCTOS_EDITAR.EditarProductoSP(?, ?, ?) )";
164             System.out.println("Edicion mediante paquete");
165
166             try (CallableStatement callableStatement = connection.prepareCall(sql: callStatement)) {
167                 callableStatement.setInt(parameterIndex: 1, value: producto.getIdProducto().intValue());
168                 callableStatement.setString(parameterIndex: 2, value: producto.getImagen());
169                 callableStatement.setString(parameterIndex: 3, value: producto.getNombre());
170                 callableStatement.setDouble(parameterIndex: 4, value: producto.getPrecio());
171                 callableStatement.setInt(parameterIndex: 5, value: producto.getExistencias());
172                 callableStatement.setString(parameterIndex: 6, value: producto.getEstado());
173
174                 callableStatement.execute();
175                 System.out.println("StoredProcedure executed successfully.");
176             } catch (SQLException e) {
177                 System.out.println("Error detected");
178             }
179         }
180     }

```

Figura 97. Productos

Fuente: Propia

Esta clase, llamada EmpleadoServiceImpl, implementa la interfaz EmpleadoService y se encarga de manejar la lógica relacionada con los empleados en un sistema. Al igual que los códigos previos, aquí también se utilizan procedimientos almacenados en una base de datos Oracle para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en la entidad "Empleado".

El método getEmpleados() establece una conexión a la base de datos Oracle y ejecuta un procedimiento almacenado para obtener una lista de empleados. Los detalles de cada empleado, como su nombre, apellidos, correo, salario, puesto, etc., se extraen del ResultSet y se almacenan en objetos Empleado. Estos objetos se agregan a una lista que se devuelve al final del método.

El método getEmpleado(Empleado empleado) toma un objeto Empleado como entrada y busca ese empleado en la lista obtenida del método anterior. Si lo encuentra, devuelve los detalles de ese empleado.

deleteEmpleado(Empleado empleado) realiza la eliminación de un empleado. Utiliza un procedimiento almacenado que toma el ID del empleado a eliminar y, tras ejecutarlo, muestra un mensaje de éxito si la operación fue exitosa.

`saveEmpleado(Empleado empleado)` realiza la inserción de un nuevo empleado en la base de datos. Se llama a un procedimiento almacenado que toma los datos del empleado como parámetros y, después de ejecutarlo, muestra un mensaje de éxito.

`editarEmpleado(Empleado empleado)` permite actualizar los detalles de un empleado. Llama a un procedimiento almacenado que toma los nuevos valores de los detalles del empleado y los aplica en la base de datos.

En resumen, esta clase proporciona métodos para interactuar con empleados en una base de datos Oracle utilizando procedimientos almacenados. Al igual que los códigos anteriores, maneja operaciones de lectura, inserción, actualización y eliminación de empleados, y gestiona la comunicación con la base de datos mediante JDBC (Java Database Connectivity).

```

1 package com.proyecto.service.Impl;
2
3 import com.proyecto.dao.EmpleadoDao;
4 import com.proyecto.dao.TiendaDao;
5 import com.proyecto.domain.Empleado;
6 import com.proyecto.domain.Producto;
7 import com.proyecto.domain.Tienda;
8 import com.proyecto.service.EmpleadoService;
9 import java.sql.CallableStatement;
10 import java.sql.Connection;
11 import java.sql.DriverManager;
12 import java.sql.ResultSet;
13 import java.sql.SQLException;
14 import java.util.ArrayList;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.stereotype.Service;
17
18 import java.util.List;
19 import oracle.jdbc.OracleTypes;
20
21 @Service
22 public class EmpleadoServiceImpl implements EmpleadoService {
23
24     private final EmpleadoDao empleadoDao;
25
26
27     @Autowired
28     public EmpleadoServiceImpl(EmpleadoDao empleadoDao) {
29         this.empleadoDao = empleadoDao;
30     }
31
32
33
34     @Override
35     public List<Empleado> getEmpleados() {
36         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
37         List<Empleado> lista = new ArrayList<>();
38
39         try (Connection conexion = DriverManager.getConnection(jdbcUrl, "nicole", "987654")) {
40             System.out.println("Conectado a la base de datos");
41
42             String callStatement = "(call C##finnk.FINNK_EMPLEADOS_OBTENER.ObtenerEmpleadosSP(?))";
43             System.out.println("Obtencion de empleados mediante paquete");
44             try (CallableStatement llamadaExecute = conexion.prepareCall(callStatement)) {
45                 llamadaExecute.registerOutParameter(parameterIndex: 1, sqlType: OracleTypes.CURSOR);
46
47                 llamadaExecute.execute();
48                 ResultSet resultSet = (ResultSet) llamadaExecute.getObject(parameterIndex: 1);
49
50                 while (resultSet.next()) {
51                     Empleado empleado = new Empleado();
52                     empleado.setIdEmpleado(resultSet.getLong(columnLabel: "id_empleado"));
53                     empleado.setNombre(nombre: resultSet.getString(columnLabel: "nombre_empleado"));
54                     empleado.setApellidos(apellidos: resultSet.getString(columnLabel: "apellidos_empleado"));
55                     empleado.setCorreo(correo: resultSet.getString(columnLabel: "correo_empleado"));
56                     empleado.setTelefono(telefono: resultSet.getInt(columnLabel: "telefono_empleado"));
57                     empleado.setSalario(salario: resultSet.getInt(columnLabel: "salario_empleado"));
58                     empleado.setPuesto(puesto: resultSet.getString(columnLabel: "puesto_empleado"));
59                     empleado.setNacionalidad(nacionalidad: resultSet.getString(columnLabel: "nacionalidad_empleado"));
60                     empleado.setEstado(estado: resultSet.getString(columnLabel: "estado_empleado"));
61                     empleado.setInfoAdicional(infoAdicional: resultSet.getString(columnLabel: "info_adicional"));
62
63                     lista.add(empleado);
64                 }
65             }
66
67             resultSet.close();
68         } catch (SQLException e) {
69             System.out.println("Error");
70             e.printStackTrace();
71         }
72
73         return lista;
74     }
75
76
77     @Override
78     public Empleado getEmpleado(Empleado empleado) {
79
80
81         List<Empleado> empleados = getEmpleados();
82         Empleado empleadoRetorno = new Empleado();
83         int idEmpleado = empleado.getIdEmpleado().intValue();
84
85
86
87         for (int i=0; i<empleados.size(); i++) {
88

```

Figura 98. Empleado

Fuente: Propia

Figura 99. Empleado

Fuente: Propia

```
178 |         callableStatement.setString(parameterIndex: 4, :: empleado.getCorreo());
179 |         callableStatement.setInt(parameterIndex: 5, :: empleado.getTelefono());
180 |         callableStatement.setDouble(parameterIndex: 6, :: empleado.getSalario());
181 |         callableStatement.setString(parameterIndex: 7, :: empleado.getPuesto());
182 |         callableStatement.setString(parameterIndex: 8, :: empleado.getNacionalidad());
183 |         callableStatement.setString(parameterIndex: 9, :: empleado.getEstado());
184 |         callableStatement.registerOutParameter(parameterIndex: 10, sqlType:OracleTypes.CURSOR);
185 |
186 |         callableStatement.execute();
187 |
188 |         ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 10);
189 |         // Manejar el resultado del cursor como sea necesario
190 |     } catch (SQLException e) {
191 |         System.out.println(:: "Error detected");
192 |         e.printStackTrace();
193 |     }
194 | }
195 }
196
197 }
```

Figura 100. Empleado

Fuente: Propia

Esta clase, llamada TiendaServiceImpl, implementa la interfaz TiendaService y se encarga de manejar la lógica relacionada con las tiendas en un sistema. Al igual que los códigos previos, aquí también se utilizan procedimientos almacenados en una base de datos Oracle para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en la entidad "Tienda".

El método getTiendas() establece una conexión a la base de datos Oracle y ejecuta un procedimiento almacenado para obtener una lista de tiendas. Los detalles de cada tienda, como su localidad y estado, se extraen del ResultSet y se almacenan en objetos Tienda. Estos objetos se agregan a una lista que se devuelve al final del método.

El método gettienda(Tienda tienda) toma un objeto Tienda como entrada y busca esa tienda en la lista obtenida del método anterior. Si la encuentra, devuelve los detalles de esa tienda.

deletetienda(Tienda tienda) realiza la eliminación de una tienda. Utiliza un procedimiento almacenado que toma el ID de la tienda a eliminar y, tras ejecutarlo, muestra un mensaje de éxito si la operación fue exitosa.

savetienda(Tienda tienda) realiza la inserción de una nueva tienda en la base de datos. Se llama a un procedimiento almacenado que toma los datos de la tienda como parámetros y, después de ejecutarlo, muestra un mensaje de éxito.

`editarTienda(Tienda tienda)` permite actualizar los detalles de una tienda. Llama a un procedimiento almacenado que toma los nuevos valores de los detalles de la tienda y los aplica en la base de datos. Luego, muestra el resultado del cursor, que contiene la información actualizada de la tienda.

```

1 package com.proyecto.service.Impl;
2
3 import com.proyecto.dao.TiendaDao;
4 import com.proyecto.domain.Tienda;
5 import com.proyecto.service.TiendaService;
6 import java.sql.CallableStatement;
7 import java.sql.Connection;
8 import java.sql.DriverManager;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11 import java.util.ArrayList;
12 import java.util.List;
13 import oracle.jdbc.OracleTypes;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.stereotype.Service;
16
17 @Service
18 public class TiendaServiceImpl implements TiendaService{
19
20     //Esto crea una unica copia en un objeto //
21     @Autowired
22     public TiendaDao tiendaDao;
23
24     @Override
25     public List<Tienda> getTiendas() {
26         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
27         List<Tienda> lista = new ArrayList<>();
28
29         try (Connection conexion = DriverManager.getConnection(url:jdbcUrl, user:"nicole", password:"987654321")) {
30             System.out.println("Conectado a la base de datos");
31
32             String callStatement = "{ call C##finnnk.obtener_tiendas_sp(?) }";
33
34             try (CallableStatement callableStatement = conexion.prepareCall(sql:callStatement)) {
35                 callableStatement.registerOutParameter(parameterIndex: 1, sqlType:OracleTypes.CURSOR);
36                 callableStatement.execute();
37
38                 ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 1);
39
40                 while (resultSet.next()) {
41                     Tienda tienda = new Tienda();
42                     tienda.setIdTienda(idTienda:resultSet.getLong(columnLabel:"id_tienda"));
43                     tienda.setLocalidad(localidad: resultSet.getString(columnLabel:"localidad_tienda"));
44                     tienda.setEstado(estado: resultSet.getString(columnLabel:"estado_tienda"));
45
46                     lista.add(tienda);
47                 }
48                 System.out.println("Lectura de tiendas realizada correctamente desde procedimiento");
49
50                 resultSet.close();
51             }
52             catch (SQLException e) {
53                 System.out.println("Error");
54                 e.printStackTrace();
55             }
56
57             return lista;
58         }
59
60     }
61
62     @Override
63     public Tienda gettienda(Tienda tienda) {
64
65
66         List<Tienda> tiendas=getTiendas();
67         Tienda tiendaRetorno = new Tienda();
68         int idTienda=tienda.getIdTienda().intValue();
69
70
71         System.out.println("Ventana de lectura de tienda ejecutada con exito");
72         for (int i=0; i<tiendas.size(); i++){
73
74             if (tiendas.get(index: i).getIdTienda()==idTienda){
75                 System.out.println(idTienda+" ID a editar");
76
77                 System.out.println("I del producto del bucle"+tiendas.get(index: i).getIdTienda());
78                 tiendaRetorno=tiendas.get(index: i);
79
80                 break;
81             }
82
83         }
84
85         return tiendaRetorno;
86     }
87

```

Figura 101. Tienda

Fuente: Propia

```

68     @Override
69     public void deletetienda(Tienda tienda) {
70         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
71         String callStatement = "{ call C##finnk.EliminarTiendaSP(?, ?) }";
72         String resultado = "";
73
74         try (Connection conexion = DriverManager.getConnection(url:jdbcUrl, user:"nicole", password:"98765");
75              CallableStatement callableStatement = conexion.prepareCall(sql:callStatement);
76              callableStatement.setLong(parameterIndex: 1, value:tienda.getIdTienda());
77              callableStatement.registerOutParameter(parameterIndex: 2, sqlType:OracleTypes.CURSOR);
78
79              callableStatement.execute();
80
81              ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 2);
82
83              if (resultSet.next()) {
84                  resultado = resultSet.getString(columnLabel:"mensaje");
85                  System.out.println(resultado + " EXITO");
86              }
87
88              resultSet.close();
89          } catch (SQLException e) {
90              System.out.println("Error detected");
91              e.printStackTrace();
92          }
93      }
94
95      @Override
96      public void savetienda(Tienda tienda) {
97          String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
98
99          try (Connection connection = DriverManager.getConnection(url:jdbcUrl, user:"C##finnk", password:"lk");
100             String callStatement = "{ call AgregarTiendaSP(?, ?, ?) }";
101
102             try (CallableStatement callableStatement = connection.prepareCall(sql:callStatement)) {
103                 callableStatement.setString(parameterIndex: 1, value:tienda.getLocalidad());
104                 callableStatement.setString(parameterIndex: 2, value:tienda.getEstado());
105                 callableStatement.registerOutParameter(parameterIndex: 3, sqlType:OracleTypes.CURSOR);
106
107                 callableStatement.execute();
108
109                 ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 3);
110                 System.out.println("tienda agregada mediante sp ");
111                 if (resultSet.next()) {
112                     System.out.println("Tienda agregada correctamente. ID: " + resultSet.getInt(columnLabel:"id_tienda"));
113                 }
114
115                 resultSet.close();
116             } catch (SQLException e) {
117                 System.out.println("Error detected");
118                 e.printStackTrace();
119             }
120         }
121     }
122
123     @Override
124     public void editarTienda(Tienda tienda) {
125         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
126
127         try (Connection connection = DriverManager.getConnection(url:jdbcUrl, user:"nicole", password:"98765");
128             System.out.println("Connected to the Oracle database");
129
130             String callStatement = "{ call C##finnk.EditarTiendaSP(?, ?, ?, ?) }";
131
132             try (CallableStatement callableStatement = connection.prepareCall(sql:callStatement)) {
133                 callableStatement.setInt(parameterIndex: 1, value:tienda.getIdTienda().intValue());
134                 callableStatement.setString(parameterIndex: 2, value:tienda.getLocalidad());
135                 callableStatement.setString(parameterIndex: 3, value:tienda.getEstado());
136                 callableStatement.registerOutParameter(parameterIndex: 4, sqlType:OracleTypes.CURSOR);
137
138                 callableStatement.execute();
139                 System.out.println("Edición de tienda completada.");
140
141             } catch (SQLException e) {
142                 System.out.println("Error detected");
143                 e.printStackTrace();
144             }
145         }
146     }
147
148     @Override
149     public void consultarTienda() {
150         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
151
152         try (Connection connection = DriverManager.getConnection(url:jdbcUrl, user:"nicole", password:"98765");
153             System.out.println("Connected to the Oracle database");
154
155             String callStatement = "{ call C##finnk.ConsultarTiendaSP(?) }";
156
157             try (CallableStatement callableStatement = connection.prepareCall(sql:callStatement)) {
158                 callableStatement.registerOutParameter(parameterIndex: 1, sqlType:OracleTypes.CURSOR);
159
160                 callableStatement.execute();
161                 System.out.println("Datos de la tienda consultada.");
162
163                 // Obtener el resultado del cursor
164                 ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 1);
165                 while (resultSet.next()) {
166                     int idTienda = resultSet.getInt(columnLabel:"id_tienda");
167                     String localidad = resultSet.getString(columnLabel:"localidad_tienda");
168                     String estado = resultSet.getString(columnLabel:"estado_tienda");
169
170                     System.out.println("Resultado del cursor - ID Tienda: " + idTienda + ", Localidad: " + localidad + ", Estado: " + estado);
171                 }
172             } catch (SQLException e) {
173                 System.out.println("Error detected");
174                 e.printStackTrace();
175             }
176         }
177     }

```

Figura 102. Tienda

Fuente: Propia

Esta clase, llamada ReclamoServiceImpl, implementa la interfaz ReclamoService y se encarga de manejar la lógica relacionada con los reclamos en un sistema. Aquí se muestra un ejemplo de cómo se podrían realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en la entidad "Reclamo", utilizando procedimientos almacenados.

El método getReclamos() establece una conexión a la base de datos Oracle y ejecuta un procedimiento almacenado para obtener una lista de reclamos. Los detalles de cada reclamo, como su nombre y consulta, se extraen del ResultSet y se almacenan en objetos Reclamo. Estos objetos se agregan a una lista que se devuelve al final del método.

deleteReclamo(Reclamo reclamo) realiza la eliminación de un reclamo. Utiliza un procedimiento almacenado que toma el ID del reclamo a eliminar y, tras ejecutarlo, muestra un mensaje de éxito si la operación fue exitosa.

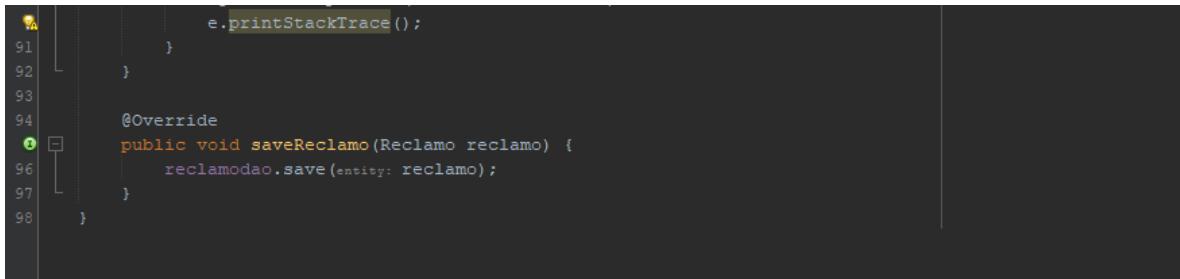
```

1 package com.proyecto.service.Impl;
2
3 import com.proyecto.dao.ReclamoDao;
4 import com.proyecto.domain.Reclamo;
5 import com.proyecto.service.ReclamoService;
6 import java.sql.CallableStatement;
7 import java.sql.Connection;
8 import java.sql.DriverManager;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11 import java.util.ArrayList;
12 import java.util.List;
13 import oracle.jdbc.OracleTypes;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.stereotype.Service;
16
17 @Service
18 public class ReclamoServiceImpl implements ReclamoService{
19
20     //Esto crea una unica copia en un objeto //
21     @Autowired
22     public ReclamoDao reclamodao;
23
24     @Override
25     public List<Reclamo> getReclamos() {
26         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
27         List<Reclamo> listaReclamos = new ArrayList<>();
28
29         try (Connection connection = DriverManager.getConnection(url;jdbcUrl, user: "nicole", password: "987654321")) {
30             System.out.println("Connected to the Oracle database");
31
32             String callStatement = "(call C##finnk.PCK_FINNK_RECLAMOS_OBTENER.ObtenerReclamosSP(?))";
33             System.out.println("Obtener reclamos mediante el paquete");
34
35             try (CallableStatement callableStatement = connection.prepareCall(sql:callStatement)) {
36                 callableStatement.registerOutParameter(parameterIndex: 1, sqlType: OracleTypes.CURSOR);
37                 callableStatement.execute();
38
39                 ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 1);
40
41                 while (resultSet.next()) {
42                     Reclamo reclamo = new Reclamo();
43                     reclamo.setIdReclamo(idReclamo: resultSet.getLong(columnLabel: "id_reclamos"));
44                     reclamo.setNombre(nombre: resultSet.getString(columnLabel: "nombre_reclamo"));
45                     reclamo.setConsulta(consulta: resultSet.getString(columnLabel: "comentario_reclamo"));
46
47                     listaReclamos.add(reclamo);
48                 }
49
50                 resultSet.close();
51             }
52             } catch (SQLException e) {
53                 System.out.println("Error detected");
54                 e.printStackTrace();
55             }
56
57             return listaReclamos;
58         }
59
60     }
61
62     @Override
63     public Reclamo getReclamo(Reclamo reclamo) {
64         return reclamodao.findById(id: reclamo.getIdReclamo()).orElse(null);
65     }
66
67     @Override
68     public void deleteReclamo(Reclamo reclamo) {
69         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
70         String callStatement = "(call C##finnk.PCK_FINNK_RECLAMOS_ELIMINAR.EliminarReclamoSP(?, ?))";
71         System.out.println("Reclamo eliminado mediante paquete");
72         String resultado = null;
73
74         try (Connection conexion = DriverManager.getConnection(url;jdbcUrl, user: "nicole", password: "987654321")) {
75             CallableStatement callableStatement = conexion.prepareCall(sql:callStatement);
76             callableStatement.setLong(parameterIndex: 1, value: reclamo.getIdReclamo());
77             callableStatement.registerOutParameter(parameterIndex: 2, sqlType: OracleTypes.CURSOR);
78
79             callableStatement.execute();
80
81             ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 2);
82             System.out.println("Reclamo eliminado mediante procedimiento, funcion y cursor.");
83             if (resultSet.next()) {
84                 resultado = resultSet.getString(columnLabel: "mensaje");
85                 System.out.println(resultado + " EXITO");
86             }
87
88             resultSet.close();
89         } catch (SQLException e) {
90             System.out.println("Error detected");
91             e.printStackTrace();
92         }
93     }

```

Figura 102. Reclamos

Fuente: Propia



```
91     e.printStackTrace();
92 }
93
94 @Override
95 public void saveReclamo(Reclamo reclamo) {
96     reclamodao.save(entity: reclamo);
97 }
98 }
```

Figura 103. Reclamos

Fuente: Propia

Esta clase, llamada InformacionServiceImpl, implementa la interfaz InformacionService y se encarga de manejar la lógica relacionada con la información de clientes en un sistema. Al igual que las clases anteriores, aquí se realiza la interacción con la base de datos Oracle para realizar operaciones CRUD en la entidad "Informacion" utilizando procedimientos almacenados.

getInformacion() establece una conexión a la base de datos Oracle y ejecuta un procedimiento almacenado para obtener la información básica de los clientes, como su ID y correo electrónico. Los detalles de cada cliente se extraen del ResultSet y se almacenan en objetos Informacion. Estos objetos se agregan a una lista que se devuelve al final del método.

deleteInformacion(Informacion informacion) realiza la eliminación de la información de un cliente. Utiliza un procedimiento almacenado que toma el ID del cliente a eliminar y, tras ejecutarlo, muestra un mensaje de éxito si la operación fue exitosa.

```

1 package com.proyecto.service.Impl;
2
3 import com.proyecto.dao.InformacionDao;
4 import com.proyecto.domain.Informacion;
5 import com.proyecto.service.InformacionService;
6 import java.sql.CallableStatement;
7 import java.sql.Connection;
8 import java.sql.DriverManager;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11 import java.util.ArrayList;
12 import java.util.List;
13 import oracle.jdbc.OracleTypes;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.stereotype.Service;
16
17 @Service /*Se utiliza para que sea visible en el controlador */
18
19 public class InformacionServiceImpl implements InformacionService{
20
21     //Esto crea una unica copia en un objeto //
22     @Autowired
23     public InformacionDao informaciondao;
24
25     @Override
26     public List<Informacion> getInformacion() {
27         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
28         List<Informacion> lista = new ArrayList<>();
29
30         try (Connection conexion = DriverManager.getConnection(url:jdbcUrl, user:"nicole", password:"987654")) {
31             System.out.println("Conectado a la base de datos");
32
33             String callStatement = "( call C##finnk.obtener_informacion_clientes_sp(?) )";
34
35             try (CallableStatement llamadaExecute = conexion.prepareCall(sql:callStatement)) {
36                 llamadaExecute.registerOutParameter(parameterIndex: 1, sqlType:OracleTypes.CURSOR);
37
38                 llamadaExecute.execute();
39                 ResultSet resultSet = (ResultSet) llamadaExecute.getObject(parameterIndex: 1);
40
41                 while (resultSet.next()) {
42                     Informacion informacion = new Informacion();
43                     informacion.setIdPersona(idPersona: resultSet.getLong(columnLabel:"id_cliente"));
44                     informacion.setCorreo(correo: resultSet.getString(columnLabel:"correo_cliente"));
45
46                     lista.add(: informacion);
47                 }
48
49                 resultSet.close();
50             }
51         } catch (SQLException e) {
52             System.out.println("Error");
53             e.printStackTrace();
54         }
55
56         return lista;
57     }
58
59     @Override
60     public Informacion getInformacion(Informacion informacion) {
61         return informaciondao.findById(id: informacion.getIdPersona()).orElse(null);
62     }
63
64     // ...
65
66     @Override
67     public void deleteInformacion(Informacion informacion) {
68         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";
69         String callStatement = "( call C##finnk.eliminar_cliente_sp(?, ?) )";
70         String resultado = "";
71
72         try (Connection conexion = DriverManager.getConnection(url:jdbcUrl, user:"nicole", password:"987654")) {
73             CallableStatement callableStatement = conexion.prepareCall(sql:callStatement);
74             callableStatement.setLong(parameterIndex: 1, : informacion.getIdPersona());
75             callableStatement.registerOutParameter(parameterIndex: 2, sqlType:OracleTypes.CURSOR);
76
77             callableStatement.execute();
78
79             ResultSet resultSet = (ResultSet) callableStatement.getObject(parameterIndex: 2);
80
81             if (resultSet.next()) {
82                 resultado = resultSet.getString(columnLabel:"mensaje");
83                 System.out.println(: resultado);
84             }
85
86             resultSet.close();
87         } catch (SQLException e) {
88             System.out.println("Error detected");
89             e.printStackTrace();
90         }
91     }

```

Figura 104. Información

Fuente: Propia

```
91 |     }
92 |
93 |
94 |     @Override
95 |     public void saveInformacion(Informacion informacion) {
96 |         informaciondao.save(entity: informacion);
97 |     }
98 | }
```

Figura 105. Información  
Fuente: Propia

Conexión hacia la base de datos por medio del URL y se ingresa con el username C##finnk y la contraseña kaws2023, esto fue creado en la base de datos.

```
server.port=8080
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:orcl

spring.datasource.username=C##finnk
spring.datasource.password=kaws2023
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

# Opcional: Configuración adicional de Hibernate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.Oracle12cDialect
spring.jpa.hibernate.ddl-auto=none
```

Figura 106. Conexión a la base de datos  
Fuente: Propia

### Anotaciones importantes:

@Controller: Esta anotación marca la clase EmpleadoController como un controlador de Spring.

@RequestMapping("/empleado"): Indica que todas las solicitudes que comiencen con "empleado" se manejarán por este controlador.

### Inyección de dependencias:

@Autowired: Marca el campo empleadoService para que Spring inyecte una instancia de EmpleadoService en el controlador.

Métodos del controlador:

Inicio (Model model): Este método maneja las solicitudes HTTP GET dirigidas a "/empleado/InventarioB". Recupera todos los empleados del EmpleadoService y los almacena en el modelo con el nombre "empleados". Luego, agrega el tamaño de la lista de empleados como "totalEmpleados" en el modelo y devuelve la vista "empleado/InventarioB".

eliminaEmpleado(Empleado empleado): Recibe un objeto Empleado como parámetro, que se obtiene de la ruta como una variable de ruta (PathVariable). Luego, utiliza el empleadoService para eliminar el empleado correspondiente y redirige al usuario nuevamente a la vista "empleado/InventarioB". nuevoEmpleado(Empleado empleado): Este método maneja las solicitudes HTTP GET dirigidas a "/empleado/nuevo". Recibe un objeto Empleado como parámetro, aunque no parece estar siendo utilizado en el método actualmente. Luego, devuelve la vista "empleado/modificaB".

guardarEmpleado(Empleado empleado): Recibe un objeto Empleado como parámetro, que debería contener los datos del nuevo empleado a guardar. Luego, utiliza el empleadoService para guardar el nuevo empleado y redirige al usuario nuevamente a la vista "empleado/InventarioB".

modificaEmpleado(Empleado empleado, Model model): Recibe un objeto Empleado como parámetro y un objeto Model, permitiendo modificar los empleados.

```

    package com.proyecto.controller;

    import com.proyecto.domain.Empleado;
    import com.proyecto.service.EmpleadoService;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Controller;
    import org.springframework.ui.Model;
    import org.springframework.web.bind.annotation.GetMapping;
    import org.springframework.web.bind.annotation.PostMapping;
    import org.springframework.web.bind.annotation.RequestMapping;

    @Controller
    @RequestMapping("/empleado")

    public class EmpleadoController {
        @Autowired
        private EmpleadoService empleadoService;

        @GetMapping("/InventarioB")
        public String inicio(Model model) {
            var empleados : List<Empleado> = empleadoService.getEmpleados();
            model.addAttribute("empleados", attributeValue: empleados);
            model.addAttribute("totalEmpleados", attributeValue: empleados.size());
            return "empleado/InventarioB";
        }

        @GetMapping("/eliminar/{idEmpleado}")
        public String eliminaEmpleado(Empleado empleado) {
            empleadoService.deleteEmpleado(empleado);
            return "redirect:/empleado/InventarioB";
        }

        @GetMapping("/nuevo")
        public String nuevoEmpleado(Empleado empleado){
            return "empleado/modificaB";
        }

        @PostMapping("/guardar")
        public String guardarEmpleado(Empleado empleado) {
            empleadoService.saveEmpleado(empleado);
            return "redirect:/empleado/InventarioB";
        }

        @GetMapping("/modificarB/{idEmpleado}")
        public String modificaEmpleado(Empleado empleado, Model model){
            empleado=empleadoService.getEmpleado(empleado);
            model.addAttribute("empleado", attributeValue: empleado);
            return "empleado/modificaB";
        }
    }

```

Figura 95. Empleado Controller  
Fuente: Propia

### Anotación @Entity:

La anotación @Entity se coloca encima de la clase Empleado y le indica a Jakarta

### Persistence

(JPA) que esta clase es una entidad que puede ser persistida en una base de datos.

### Atributos de la clase:

La clase Empleado tiene varios atributos que representan las columnas de la tabla

"tab\_listado\_empleados" en la base de datos. Cada atributo tiene su propia

anotación @Column, que indica cómo se debe mapear en la base de datos. Por ejemplo,

@Column(name = "nombre\_empleado") le dice a JPA que el atributo nombre debe

mapearse a la columna "nombre\_empleado" en la tabla.

### Atributo @Id:

El atributo idEmpleado está marcado con @Id, lo que indica que es la clave primaria de la entidad. En este caso, se generará automáticamente mediante una estrategia definida por

@GeneratedValue(strategy = GenerationType.IDENTITY). Esta estrategia suele utilizar una columna de autoincremento en la base de datos para generar valores únicos para la clave primaria.

### Anotación @Table:

Se coloca en la clase y especifica el nombre de la tabla en la base de datos a la que está asociada esta entidad. En este caso, la tabla se llama "tab\_listado\_empleados".

### Constructor y Anotación @Data:

La clase tiene dos constructores: uno sin vacío y otro que toma todos los atributos como parámetros.

```
@Data
@Entity
@Table(name = "tab_listado_empleados")
public class Empleado implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_empleado")
    private Long idEmpleado;

    @Column(name = "nombre_empleado")
    private String nombre;

    @Column(name = "apellidos_empleado")
    private String apellidos;

    @Column(name = "correo_empleado")
    private String correo;

    @Column(name = "telefono_empleado")
    private int telefono;

    @Column(name = "salario_empleado")
    private int salario;

    @Column(name = "puesto_empleado")
    private String puesto;

    @Column(name = "nacionalidad_empleado")
    private String nacionalidad;

    @Column(name = "estado_empleado")
    private String estado;

    public Empleado() {
    }

    public Empleado(Long idEmpleado, String nombre, String apellidos, String correo, int telefono, int salario, String puesto, String nacionalidad, String estado) {
        this.idEmpleado = idEmpleado;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.correo = correo;
        this.telefono = telefono;
        this.salario = salario;
        this.puesto = puesto;
        this.nacionalidad = nacionalidad;
        this.estado = estado;
    }
}
```

Figura 96. Domain Empleado

Fuente: Propia

La interfaz se llama EmpleadoDao, y extiende la interfaz JpaRepository, esta interfaz JpaRepository se parametriza con dos tipos.

Empleado, esta es manejada por esta interfaz en este caso esta clase representa la tabla de empleados.

Long: Clave primaria en la clase Empleado, su atributo idEmpleado de tipo Long como su clave primaria.

Operaciones CRUD:

Estas se proporcionan de una forma automática, estas operaciones son las de crear, leer, actualizar y eliminar la entidad empleado, algunos de los métodos utilizados son.

Save(): Se utiliza para guarda un nuevo empleado o actualizar algún empleado que ya exista.

findById(): Este es utilizado para buscar un empleado por su id.

findAll(): Este es usado para obtener todos los empleados.

deleteById(): Elimina un empleado por su identificador.

```
package com.proyecto.dao;

import com.proyecto.domain.Empleado;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EmpleadoDao extends JpaRepository<Empleado, Long>{}
```

Figura 96. Empleado Dao  
Fuente: Propia

La interfaz EmpleadoService proporciona una abstracción sobre las operaciones de persistencia relacionadas con la entidad Empleado.

```
package com.proyecto.service;

import com.proyecto.domain.Empleado;
import java.util.List;

public interface EmpleadoService {
    public List<Empleado> getEmpleados();

    //obtiene el registro de la tabla empleado
    //que tiene el idEmpleado pasado por el empleado
    public Empleado getEmpleado(Empleado empleado);

    //elimina el registro de la tabla empleado
    //que tiene el idEmpleado pasado por el empleado
    public void deleteEmpleado(Empleado empleado);

    //si el idempleado pasado no existe o es nulo se crea un registro nuevo
    //en la tabla empleado
    // si el id empleado existe se actualiza la informacion
    public void saveEmpleado(Empleado empleado);
}
```

Figura 97. Empleado Service

Fuente: Propia

EmpleadoServiceImpl es una clase de servicio que implementa la lógica de negocio para las operaciones CRUD relacionadas con la entidad Empleado. Esta tiene el permiso para acceder a la capa de persistencia representada por EmpleadoDao.

```
package com.proyecto.service.Impl;

import com.proyecto.dao.EmpleadoDao;
import com.proyecto.dao.TiendaDao;
import com.proyecto.domain.Empleado;
import com.proyecto.domain.Tienda;
import com.proyecto.service.EmpleadoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmpleadoServiceImpl implements EmpleadoService {

    private final EmpleadoDao empleadoDao;

    @Autowired
    public EmpleadoServiceImpl(EmpleadoDao empleadoDao) {
        this.empleadoDao = empleadoDao;
    }

    @Override
    public List<Empleado> getEmpleados() {
        List<Empleado> empleados = empleadoDao.findAll();

        return empleados;
    }

    @Override
    public Empleado getEmpleado(Empleado empleado) {
        return empleadoDao.findById(empleado.getIdEmpleado()).orElse(null);
    }

    @Override
    public void deleteEmpleado(Empleado empleado) {
        empleadoDao.delete(empleado);
    }

    @Override
    public void saveEmpleado(Empleado empleado) {
        empleadoDao.save(empleado);
    }
}
```

Figura 98. Empleado Service Impl

Fuente: Propia

## **Conclusiones**

La importancia de la tienda virtual es mejorar y optimizar sus bases de datos para resolver problemas de rendimiento y aprovechar mejor la información almacenada.

Se identifican errores y deficiencias en las bases de datos actuales, por medio de estas afectan diversas áreas de la página web de la tienda, incluyendo la gestión de productos, clientes y empleados.

Las mejoras en la base de datos de productos incluirán la garantía de información completa, precisa y actualizada, así como una estructura de categorización mejorada para mejorar la experiencia del usuario.

La base de datos de clientes se enriquecerá mediante la recopilación de datos informáticos y detallados, historiales de compras y opiniones, lo que permitirá una mejor comprensión y atención a las necesidades de los clientes.

La base de datos de empleados se actualizará con información relevante, como datos de contacto, historial laboral, habilidades y certificaciones, lo que facilitará la gestión de recursos humanos.

## **Recomendaciones**

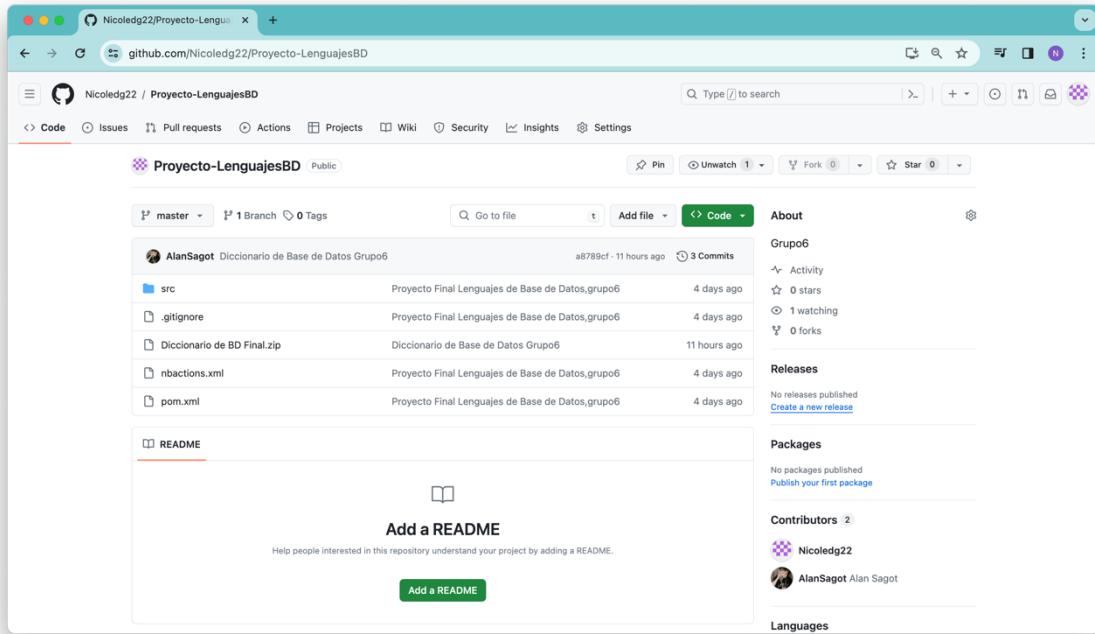
1. Se debe de realizar un análisis exhaustivo de los requerimientos del proyecto antes de comenzar la implementación. Incluyendo el definir en detalle las funcionalidades esperadas de la tienda, así como sus datos importantes, como lo son los flujos de datos y operaciones necesarios para su correcto funcionamiento.
  
2. Se recomiendan aplicar buenas prácticas de diseño y programación es importante utilizar las convenciones de nomenclatura claras y ordenes, actualizar el código para

facilitar su mantenimiento, asegurarse de que todos los datos estén guardados y digitados de una manera correcta para que corra de una manera exitosa.

3. Aplicar pruebas para garantizar la calidad y la integridad de la página web.

Implicando probar los diferentes escenarios de la página web, en donde se valida un uso de manipulación correcta de los datos ingresados, y se verifica que no exista ningún tipo de error o falla en el código y pagina web.

# Repositorio de GitHub



[Link Git](#)

<https://github.com/Nicoledg22/Proyecto-LenguajesBD>