Ingeniería en Sistemas de Computación

Proyecto:

Tienda La Moda

Integrantes del equipo:

Heymmy Massiel López Juárez

Frander Alfredo Rojas Madrigal

David Alexander Urroz Caballero

Aytor Zumbado Gaitán

Profesor:

Randall Alonso Leiton Jiménez

III Cuatrimestre, 2024

Introducción

El proyecto Tienda La Moda busca desarrollar un sistema de gestión integral utilizando Java y Oracle para optimizar la administración de clientes, empleados, proveedores, productos y ventas. Este sistema permitirá a los empleados de la tienda gestionar inventarios, facturación y datos de manera eficiente a través de una interfaz gráfica. La elección de Java garantiza portabilidad, escalabilidad y una integración robusta con la base de datos, lo que asegura un control efectivo y seguro de la información, mejorando los procesos operativos de la tienda.

Objetivos

Objetivo general Desarrollar un sistema de gestión para la tienda "Tienda La Moda", que permita administrar eficientemente los módulos de clientes, empleados, proveedores, productos y ventas, utilizando el lenguaje de programación Java y el sistema de base de datos relacional.

Objetivos específicos

1- Diseñar y desarrollar un módulo para gestionar la información del cliente: centrarse en registrar, actualizar, eliminar y consultar los datos.

2- Implementar un módulo de gestión de empleados: enfocado a la información sobre trabajo y roles y permisos de la parte de administración.

3- Crear un módulo de ventas: Integrarlo con la gestión de productos y proveedores, debe permitir facturar, controlar el inventario y hacer un reporte de venta en tiempo real.

Alcance

El proyecto "Tienda La Moda" abarca el desarrollo e implementación de un sistema de gestión integral utilizando Java para la lógica de programación y Oracle como base de datos relacional, administrada a través de SQL Developer. El sistema estará compuesto por los siguientes módulos:

1. Módulo de Clientes: Permitirá la creación, modificación, eliminación y consulta de los datos de los clientes, asegurando la integridad y seguridad de la información almacenada en la base de datos Oracle.

2. Módulo de Empleados: Administra la información de los empleados, como detalles personales, roles y permisos, mediante la implementación de consultas y procedimientos almacenados en Oracle.

3. Módulo de Proveedores: Gestionará la información de los proveedores, controlando el registro, actualización y eliminación de datos a través de SQL Developer, con procedimientos que optimicen el manejo de grandes volúmenes de datos.

4. Módulo de Productos: Permitirá la gestión del inventario, incluyendo el registro, actualización y control de productos, mediante consultas eficientes que mantendrán la consistencia de los datos en la base de datos.

5. Módulo de Ventas: Este módulo permitirá gestionar las ventas de la tienda, registrando transacciones y generando reportes de ventas, integrando datos de los módulos de productos y clientes.

El sistema se implementará asegurando la integridad referencial entre las tablas de la base de datos Oracle, con un enfoque en la escalabilidad y la seguridad de los datos. Además, se realizarán pruebas exhaustivas en cada módulo para garantizar un correcto funcionamiento y rendimiento del sistema.

Contexto del emprendimiento

Para llevar a cabo los objetivos planteados del presente proyecto, tomamos como punto de partida un proyecto de tienda de moda denominada "Tienda La Moda", donde se venden artículos de vestimenta, como camisas, blusas, pantalones, faldas, entre otros. El presente proyecto tiene la finalidad de abarcar varias necesidades del negocio a nivel tecnológico, como lo puede ser la gestión de inventarios, empleados activos, clientes registrados, registro de ventas y facturación, proveedores activos, entre otros. Tomamos como referencia el uso de una aplicación con interfaz gráfica conectado a una base de datos y cuyo usuario final en el Front En serían exclusivamente los empleados de la tienda de moda y su uso esperado es desde una computadora ubicada en las cajas de la tienda, como ya es típico encontrar en cualquier tienda semejante de la actualidad.

El lenguaje de programación elegido

El sistema Tienda La Moda fue desarrollado utilizando Java como lenguaje de programación para conectarse a la base de datos Oracle y al entorno de desarrollo NetBeans. Esta elección se basó en una serie de razones técnicas que hicieron de Java una opción adecuada para el proyecto (Nieva, 2017). Java es un lenguaje de programación ampliamente utilizado en aplicaciones empresariales debido a su robustez, escalabilidad y portabilidad (Oracle, 2024). Una de sus mayores fortalezas es su capacidad para integrarse eficientemente con bases de datos relacionales como Oracle utilizando la tecnología JDBC (Java Database Connectivity), lo que facilita la conexión, manipulación y gestión de datos desde Java. Además, Java proporciona una

amplia biblioteca de herramientas y API para ayudar a crear aplicaciones eficientes y seguras (Coppola, 2023).

Ventajas técnicas

- Portabilidad: Java es independiente de la plataforma, lo que significa que las aplicaciones desarrolladas pueden ejecutarse en cualquier sistema operativo que admita la máquina virtual Java (JVM), lo que facilita las migraciones futuras si es necesario.
- Resiliencia y manejo de errores: Java incluye manejo de excepciones de alto nivel que proporciona un manejo preciso de errores en tiempo de ejecución, lo cual es esencial para las aplicaciones que interactúan con bases de datos.
- Bibliotecas y marcos: Java proporciona un amplio conjunto de bibliotecas y marcos que facilitan la conectividad de bases de datos, como Hibernate para la persistencia de datos y Spring para el desarrollo de aplicaciones empresariales. Estos sistemas mejoran la eficiencia del desarrollo y mantenimiento del sistema.
- Seguridad: Java ha incorporado múltiples capas de seguridad para crear aplicaciones seguras, lo cual es un aspecto crítico en el manejo de datos confidenciales de transacciones de ventas, clientes y empleados.

Desafíos

A pesar de las ventajas, el uso de Java en este proyecto también generó algunos problemas.

1- Complejidad de la configuración: configurar correctamente una conexión a un repositorio de Oracle utilizando JDBC puede requerir un conocimiento profundo y una configuración detallada, lo que resulta en una curva de aprendizaje pronunciada para los desarrolladores (Anónimo,2011).

2- Administración de memoria: Java administra automáticamente la memoria a través del recolector de basura, la administración eficiente de recursos y la optimización de la memoria pueden convertirse en un desafío en aplicaciones empresariales grandes con muchas transacciones simultáneas (Hernández, 2023).

3- Rendimiento: Dependiendo del tamaño de la base de datos y la cantidad de usuarios simultáneos, el rendimiento de las consultas y operaciones de la base de datos puede verse

afectado. Para mitigar este desafío, es necesario implementar técnicas de optimización en la programación y configuración de bases de datos (Hernández, 2023).

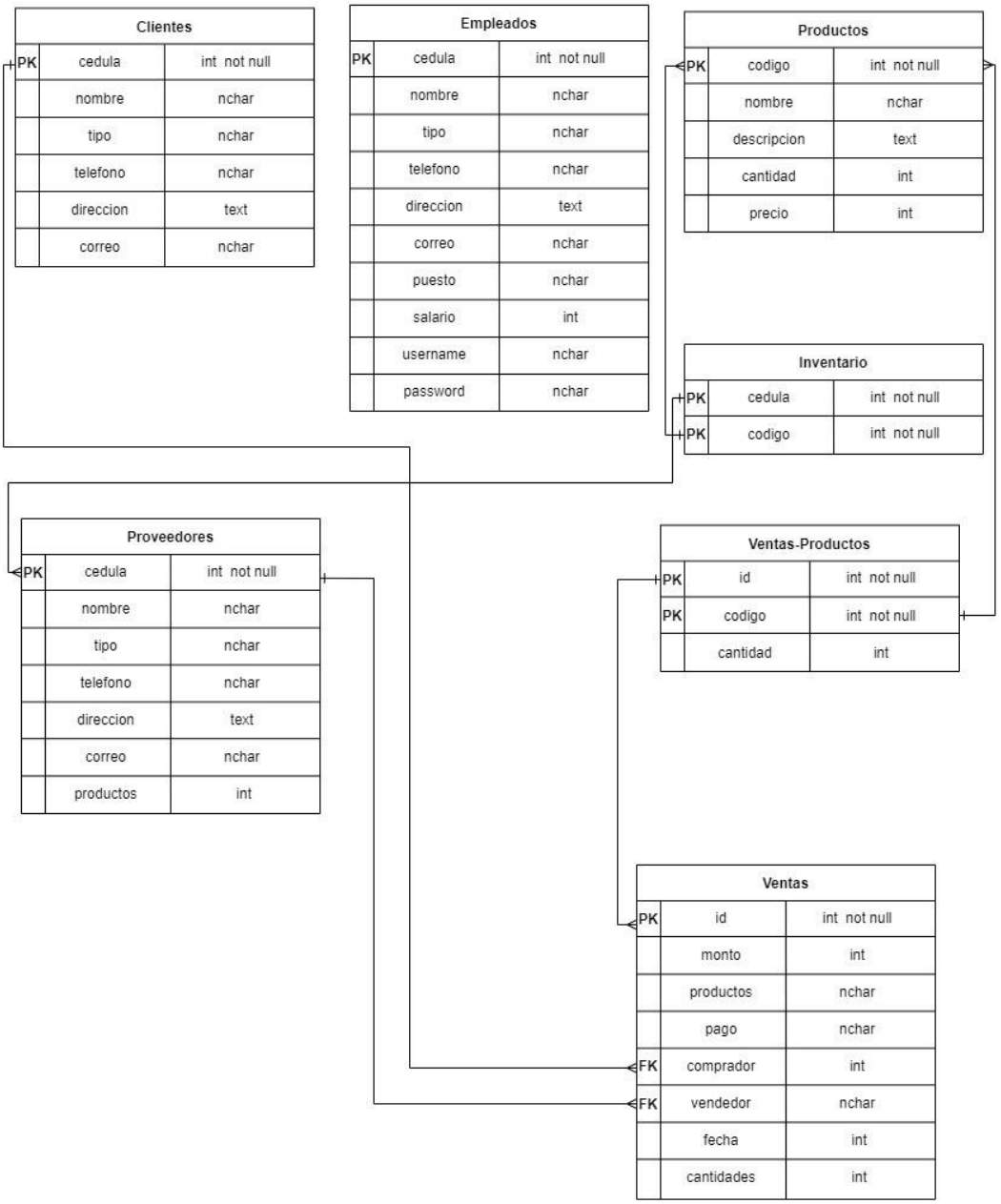
Diagrama Relacional

Dado que las bases de datos es uno de los principales temas técnicos involucrado en el presente proyecto e indispensable para el funcionamiento del aplicativo y cumplimiento de los objetivos, se procede a diagramar las relaciones de las entidades mínimas esperadas a ser utilizadas. Esto será de suma ayuda al momento de realizar la creación de la base de datos y de sus tablas y consiguientes columnas.

A continuación, se expone el diagrama relacional desarrollado.

Figura 1

Diagrama de entidad-relación



Requerimientos

Tras un análisis de las necesidades del negocio "Tienda la Moda" así como la naturaleza de los datos típicos en esta área de negocio, se procede a generar una serie de requerimientos mínimos que el desarrollo del aplicativo debe cumplir para atender todas las necesidades básicas identificadas. En las tablas se mostrarán los requerimientos tanto funcionales como no funcionales que el equipo ha declarado criterio de aceptación del producto final de desarrollo.

Tabla 1

Requerimientos Funcionales

Requerimiento
El aplicativo debe permitir crear, leer, modificar y eliminar empleados
El aplicativo debe permitir crear, leer, modificar y eliminar clientes
El aplicativo debe permitir crear, leer, modificar y eliminar proveedores
El aplicativo debe permitir crear, leer, modificar y eliminar productos
El aplicativo debe permitir crear, leer, modificar y eliminar ventas
El aplicativo debe permitir el inicio y cierre de sesión a los usuarios finales (empleados)
El aplicativo debe permitir imprimir facturas basadas en los datos de una venta
El aplicativo permite realizar búsquedas con filtros v para obtener información de empleados, clientes,
El aplicativo debe registrar los movimientos que realicen los usuarios con sesión iniciada en el aplicativo,

Tabla 2

Requerimientos No Funcionales

Requerimiento
El aplicativo debe operar en los sistemas operativos Microsoft Windows 10 u 11
El aplicativo debe operar con el motor de base de datos Oracle 9c
El código del aplicativo debe ser programado con el lenguaje de programación Java
El aplicativo debe tener una disponibilidad de 24/7
El aplicativo debe poseer la capacidad de soportar como mínimo 20 usuarios simultáneos

GitHub

Para el presente proyecto, dado que será desarrollado en equipo de trabajo y tomando las mejores prácticas del ciclo de vida del software, se procede a hacer uso de la herramienta de repositorios GitHub, famosa por sus aplicaciones en el campo de la tecnología de la información para proyectos de desarrollo de software. En ella se pretende poseer un repositorio donde cada uno de los miembros del equipo pueda tener de manera actualizadas las nuevas modificaciones del proyecto y así mismo actualizar de manera global con las modificaciones que se hagan durante el proyecto.

El enlace al repositorio empleado es el siguiente: https://github.com/rednarfsajor/Proyecto*LBD.git*

Cronograma de Actividades

Semana 2

Sábado 28 de septiembre (2:00 - 3:00 pm):

- Se llega a un acuerdo sobre el tema.
- Se asigna a cada persona las tareas a realizar.
- Se decide el lenguaje de programación.
- Se crea el documento compartido donde se irá rellenando el avance 1.

Semana 3

Martes 1 de octubre (9:00 - 9:30 pm):

- Nos reunimos para ver el avance de cada persona del grupo.
- Interactuamos sobre la base de datos y el lenguaje de programación.
- Comentamos sobre las mejoras de las tablas de la base de datos.

Viernes 11 de octubre (9:00 - 9:30 pm):

- Cada compañero con la tarea asignada debe terminar su trabajo.
- Se conversa por vía Whatsapp si algún compañero necesita ayuda para colaborar.

Sábado 12 de octubre (3:00 - 4:00 pm):

- Nos reunimos para revisar la estructura del documento y asegurar que esté en el formato correcto.
- Se valida la información que se colocó en el documento.
- Se dan recomendaciones para mejorar como equipo y asegurar que el primer avance sea correcto.
- Se acuerda mejorar el documento según corresponda para poder subir el primer avance.

Domingo 13 de octubre (3:00 - 4:00 pm):

- Se sube el primer avance.

Semana 4

Martes 15 de octubre (9:00 - 10:00 pm):

- Revisión y ajuste de la conexión a la base de datos en el lenguaje elegido.
- Distribución de tareas para la creación de CRUDs para las tablas del modelo relacional.
- Planificación de la programación de procedimientos almacenados, vistas, funciones y cursores.

Semana 5

Domingo 20 de octubre (2:00 - 5:00 pm):

- Implementación de CRUDs para el 50% de las tablas.
- Inicio de la implementación de vistas y funciones.

Semana 6

Sábado 26 de octubre (1:00 - 3:00 pm):

- Continuación de la programación de CRUDs.
- Programación del 50% de los procedimientos almacenados, vistas y funciones.
- Subida del código a GitHub y revisión de las contribuciones de cada miembro.

Semana 7

Martes 5 de noviembre (9:00 - 10:00 pm):

- Implementación de triggers y paquetes.
- Revisión de los cursores implementados y optimización del código.
- Continuación de la subida de avances a GitHub.

Semana 8

Martes 12 de noviembre (9:00 - 10:00 pm):

- Finalización del 50% de la programación del proyecto.
- Revisión y ajuste del código en GitHub.
- Creación del diccionario de datos y validación del documento generado automáticamente desde SQL Developer.

Semana 9

Martes 19 de noviembre (9:00 - 10:00 pm):

- Correcciones finales de la programación.
- Verificación de que todos los módulos estén funcionales.
- Entrega del segundo avance.

Sábado 23 de noviembre (1:00 - 3:00 pm):

- Subida del avance final a GitHub.
- Revisión final de todo el código, procedimientos almacenados y el diccionario de datos.
- Preparación del informe del avance para entrega.

Semana 10

Martes 3 de diciembre (9:00 - 1:00 pm):

- Revisión general del proyecto: estado actual del código, tareas pendientes y asignación final.
- Planificación para completar el 100% del proyecto.
- Comenzar con la finalización de los CRUDs restantes para todas las tablas del modelo relacional.
- Finalización de los CRUDs de todas las tablas.
- Implementación de los procedimientos almacenados restantes.
- Revisión y actualización del código en GitHub (control de versiones).
- Verificar la integración de los módulos: clientes, empleados, proveedores, productos y ventas.

Semana 11

Martes 17 de diciembre (9:00 - 1:00 pm):

- Finalización de las vistas, funciones, cursores y triggers pendientes.

- Realizar pruebas completas de funcionalidad para asegurarse de que todos los módulos operen correctamente.
- Realización de pruebas de rendimiento del sistema.
- Corrección de errores y optimización del código.
- Subida de la versión casi final a GitHub para revisión.
- Revisión final de todo el sistema.
- Generación del reporte final de pruebas (funcionalidad y rendimiento).
- Ajustes finales de los módulos.
- Preparación y revisión del diccionario de datos y la documentación completa del proyecto.
- Verificación de la subida de la versión definitiva a GitHub.
- Entrega del 100% del proyecto con todos los módulos finalizados y probados.
- Subida definitiva del código y la documentación completa a GitHub.

Diccionario de Datos

A continuación se presentará el detalle de todos los objetos de la base de datos, así como su propósito, entradas y salidas.

Objeto	**TABLE CLIENTES**
Entradas	**Descripción**
CEDULA	VARCHAR2(9) NOT NULL (PK)
NOMBRE	VARCHAR2(30)
PRIMER_APELLIDO	VARCHAR2(20)
SEGUNDO_APELLIDO	VARCHAR2(20)
TIPO	VARCHAR2(20)
TELEFONO	VARCHAR2(11)
DIRECCION	VARCHAR2(100)
CORREO	VARCHAR2(30)
Salidas	Tabla creada
Función	Almacena la información detallada de los clientes, incluyendo datos personales y de contacto.

Objeto	**TABLE AUDIT_CLIENTES**
Entradas	**Descripción**
LOG_ID	INT CONSTRAINT AUDIT_CLIENTES_PK PRIMARY KEY
CEDULA	VARCHAR2(9)
NOMBRE	VARCHAR2(30)
TIPO	VARCHAR2(20)
TELEFONO	VARCHAR2(11)
DIRECCION	VARCHAR2(100)
CORREO	VARCHAR2(30)
ACCION	VARCHAR (20)
FECHA_ACCION	DATE DEFAULT SYSDATE
Salidas	Tabla creada
Función	Almacena la información detallada de los clientes, incluyendo datos personales y de contacto.

Objeto	**TRIGGER TRG_AUDIT_CLIENTES**
Entradas	**Descripción**
Salidas	Los registros de la tabla AUDIT_CLIENTES se actualizan con la información correspondiente a la operación realizada (UPDATE, DELETE, INSERT)
Función	Audita los cambios realizados en la tabla CLIENTES guardando un historial de modificaciones, eliminaciones y creaciones en la tabla AUDIT_CLIENTES.

Objeto	**TRIGGER TRG_LOG_ID_AUDIT_CLIENTES**
Entradas	**Descripción**
Salidas	El campo LOG_ID en la tabla AUDIT_CLIENTES recibe un valor generado por la secuencia SEC_LOG_ID_AUDIT_CLIENTES.NEXTVAL
Función	Asigna un valor único al campo LOG_ID en cada registro insertado en la tabla AUDIT_CLIENTES usando una secuencia para asegurar la unicidad del identificador.

Objeto	**PROCEDURE REGISTRAR_CLIENTE**
Entradas	**Descripción**
CEDULA_IN	(VARCHAR2)
NOMBRE_IN	(VARCHAR2)
PRIMER_APELLIDO_IN	(VARCHAR2)
SEGUNDO_APELLIDO_IN	(VARCHAR2)
TIPO_IN	(VARCHAR2)
TELEFONO_IN	(VARCHAR2)
DIRECCION_IN	(VARCHAR2)
CORREO_IN	Correo electrónico del cliente (VARCHAR2)
Salidas	Confirmación de inserción (Cliente registrado correctamente: <CEDULA>)
Función	Registra un nuevo cliente en la tabla CLIENTES y muestra un mensaje de confirmación.

Objeto	**VIEW VISTA_CLIENTE**
Entradas	**Descripción**
CEDULA	(VARCHAR2)
NOMBRE	(VARCHAR2)
PRIMER_APELLIDO	(VARCHAR2)
SEGUNDO_APELLIDO	(VARCHAR2)
TIPO	(VARCHAR2)
TELEFONO	(VARCHAR2)
DIRECCION	(VARCHAR2)

CORREO	Correo electrónico del cliente (VARCHAR2)
Salidas	Devuelve la información de los clientes en una vista estructurada.
Función	Proporciona una vista consolidada y accesible de los datos de la tabla CLIENTES.

Objeto	**PROCEDURE VER_CLIENTES**
Entradas	**Descripción**
Salidas	El parámetro CURSOR_CLIENTE devuelve el conjunto de resultados de la vista VISTA_CLIENTE
Función	Recupera todos los registros de la vista VISTA_CLIENTE y los asigna al parámetro de salida CURSOR_CLIENTE, permitiendo que la información sea procesada fuera del procedimiento.

Objeto	**PROCEDURE BUSCAR_CLIENTE**
Entradas	**Descripción**
CED	Parámetro de entrada, tipo INT, que representa la cédula del cliente a buscar en la vista VISTA_CLIENTE
Salidas	El parámetro CURSOR_CLIENTE devuelve el registro del cliente cuyo valor de cédula coincide con el parámetro CED
Función	Recupera el registro del cliente desde la vista VISTA_CLIENTE que coincida con la cédula proporcionada en el parámetro CED, y lo asigna al parámetro de salida CURSOR_CLIENTE

Objeto	**PROCEDURE ACTUALIZAR_CLIENTE**
Entradas	**Descripción**
CEDULA_IN	(VARCHAR2)
NOMBRE_IN	(VARCHAR2)
PRIMER_APELLIDO_IN	(VARCHAR2)
SEGUNDO_APELLIDO_IN	(VARCHAR2)
TIPO_IN	(VARCHAR2)
TELEFONO_IN	(VARCHAR2)
DIRECCION_IN	(VARCHAR2)
CORREO_IN	(VARCHAR2)
Salidas	Confirmación de actualización (Cliente actualizado correctamente: <CEDULA>)
Función	Actualiza los datos de un cliente existente en la tabla CLIENTES y muestra un mensaje de confirmación.
Objeto	**PROCEDURE ELIMINAR_CLIENTE**
Entradas	**Descripción**
CEDULA_IN	(VARCHAR2)
Salidas	Confirmación de eliminación (Cliente eliminado correctamente).
Función	Elimina un cliente de la tabla CLIENTES basándose en la cédula proporcionada.

Objeto	**TABLE PROVEEDORES**
Entradas	**Descripción**
CEDULA	VARCHAR2(9) NOT NULL (PK)
NOMBRE	VARCHAR2(30)
TIPO	VARCHAR2(20)
TELEFONO	VARCHAR2(11)
DIRECCION	VARCHAR2(100)
CORREO	VARCHAR2(30)
PRODUCTOS	NUMBER
Salidas	Tabla creada
Función	Almacena la información detallada de los proveedores, incluyendo datos personales, de contacto y productos que proveen.

Objeto	**TABLE AUDIT_PROVEEDORES**
Entradas	**Descripción**
LOG_ID	INT CONSTRAINT AUDIT_PROVEEDORES_PK PRIMARY KEY
CEDULA	VARCHAR2(9)
NOMBRE	VARCHAR2(30)
TIPO	VARCHAR2(20)
TELEFONO	VARCHAR2(11)
DIRECCION	VARCHAR2(100)
CORREO	VARCHAR2(30)
PRODUCTOS	VARCHAR(300)
ACCION	VARCHAR(20)
FECHA_ACCION	DATE DEFAULT SYSDATE
Salidas	La tabla AUDIT_PROVEEDORES es creada para almacenar un historial de acciones sobre los proveedores.
Función	Almacena la información de los proveedores, incluyendo datos como cédula, nombre, productos, y las acciones realizadas sobre ellos, como

Objeto	**TRIGGER TRG_AUDIT_PROVEEDORES**
Entradas	**Descripción**
Salidas	Los registros de la tabla AUDIT_PROVEEDORES se actualizan con la información correspondiente a la operación realizada (UPDATE, DELETE, INSERT)
Función	Audita los cambios realizados en la tabla PROVEEDORES guardando un historial de modificaciones, eliminaciones y creaciones en la tabla AUDIT_PROVEEDORES.

Objeto	**PROCEDURE INSERTAR_PROVEEDOR**
Entradas	**Descripción**
CEDULA_IN	(VARCHAR2)
NOMBRE_IN	(VARCHAR2)
TIPO_IN	(VARCHAR2)
TELEFONO_IN	(VARCHAR2)
DIRECCION_IN	(VARCHAR2)
CORREO_IN	(VARCHAR2)
PRODUCTOS_IN	(NUMBER)
Salidas	Confirmación de inserción (Proveedor insertado correctamente: <CEDULA>)
Función	Inserta un nuevo proveedor en la tabla PROVEEDORES y muestra un mensaje de confirmación.

Objeto	**TRIGGER TRG_LOG_ID_AUDIT_PROVEEDORES**
Entradas	**Descripción**
Salidas	El campo LOG_ID es asignado con el valor generado por la secuencia SEC_LOG_ID_AUDIT_PROVEEDORES.NEXTVAL antes de la inserción del nuevo registro
Función	Asegura que el campo LOG_ID en la tabla AUDIT_PROVEEDORES reciba un valor único generado por la secuencia antes de la inserción del registro en la tabla de auditoría.

Objeto	**VIEW VISTA_PROVEEDOR**
Entradas	**Descripción**

CEDULA	(VARCHAR2)
NOMBRE	(VARCHAR2)
TIPO	(VARCHAR2)
TELEFONO	(VARCHAR2)
DIRECCION	(VARCHAR2)
CORREO	(VARCHAR2)
PRODUCTOS	(NUMBER)
Salidas	Devuelve la información de los proveedores en una vista estructurada.
Función	Proporciona una vista consolidada y accesible de los datos de la tabla PROVEEDORES.

Objeto	**PROCEDURE VER_PROVEEDORES**
Entradas	**Descripción**
Salidas	El parámetro CURSOR_PROVEEDOR devuelve el conjunto de resultados de la consulta ejecutada sobre la vista VISTA_PROVEEDOR.
Función	Recupera los registros de la vista VISTA_PROVEEDOR y los asigna al parámetro de salida CURSOR_PROVEEDOR, permitiendo a los usuarios acceder a los datos de los proveedores.

Objeto	**PROCEDURE BUSCAR_PROVEEDOR**
Entradas	**Descripción**
CEDULA	CED (INT)
Salidas	El parámetro CURSOR_PROVEEDOR devuelve el conjunto de resultados de la consulta ejecutada sobre la vista VISTA_PROVEEDOR filtrado por la cédula proporcionada.
Función	Permite buscar un proveedor específico en la vista VISTA_PROVEEDOR usando la cédula proporcionada, devolviendo los registros que coinciden con esa cédula a través del cursor de salida CURSOR_PROVEEDOR.

Objeto	**PROCEDURE LEER_PROVEEDORES**
Entradas	**Descripción**
CRITERIO_IN	VARCHAR2
Salidas	Lista de proveedores
Función	Lee y muestra los proveedores almacenados en la tabla TLM.PROVEEDORES. Si se proporciona un criterio, filtra los resultados por nombre.

Objeto	**PROCEDURE ACTUALIZAR_PROVEEDOR**
Entradas	**Descripción**
CEDULA_IN	(VARCHAR2)
NOMBRE_IN	(VARCHAR2)
TIPO_IN	(VARCHAR2)
TELEFONO_IN	(VARCHAR2)
DIRECCION_IN	(VARCHAR2)
CORREO_IN	(VARCHAR2)
PRODUCTOS_IN	Número de productos suministrados por el proveedor (NUMBER)
Salidas	Confirmación de actualización (Proveedor actualizado correctamente: <CEDULA>)
Función	Actualiza los datos de un proveedor existente en la tabla PROVEEDORES y muestra un mensaje de confirmación.

Objeto	**PROCEDURE ELIMINAR_PROVEEDOR**
Entradas	**Descripción**

CEDULA_IN	(VARCHAR2)
Salidas	Confirmación de eliminación (Proveedor eliminado correctamente: <CEDULA>)
Función	Elimina un proveedor de la tabla PROVEEDORES basándose en la cédula proporcionada.

Objeto	**TABLE VENTAS**
Entradas	Descripción
ID	NUMBER (PK)
MONTO	NUMBER
PRODUCTOS	VARCHAR2(50)
PAGO	VARCHAR2(25)
COMPRADOR	VARCHAR2(20) (FK)
VENDEDOR	VARCHAR2(20) (FK)
FECHA	DATE
CANTIDADES	NUMBER
Salidas	Tabla creada
Función	Almacena las ventas realizadas, incluyendo información de clientes, proveedores, productos, fecha y método de pago.

Objeto	**TABLE AUDIT_VENTAS**
Entradas	**Descripción**
LOG_ID (INT)	Identificador único para el registro de auditoría (clave primaria).
ID_VENTA (NUMBER)	Número de la venta registrada.
MONTO (NUMBER)	Monto total de la venta.
PRODUCTOS (VARCHAR2(50))	Lista de productos involucrados en la venta.
PAGO (VARCHAR2(25))	Método de pago utilizado en la venta (e.g., efectivo, tarjeta, etc.).
COMPRADOR (INT)	Identificador del comprador asociado a la venta.
VENDEDOR (INT)	Identificador del vendedor que realizó la venta.
FECHA (DATE)	Fecha en que se realizó la venta.
CANTIDADES (INT)	Cantidad de productos vendidos.
ACCION (VARCHAR2(20))	Tipo de acción realizada (e.g., "MODIFICACION", "ELIMINACION", "CREACION").
FECHA_ACCION (DATE)	Fecha y hora en que se registró la acción, con valor por defecto SYSDATE.
Salidas	Los registros de la tabla AUDIT_VENTAS se actualizan con la información de ventas y su respectiva acción.
Función	Almacena el historial de ventas, incluyendo datos como el monto, productos, comprador, vendedor, y el tipo de acción realizada (modificación, eliminación, o creación).

Objeto	**TRIGGER TRG_AUDIT_VENTAS**
Entradas	**Descripción**
Salidas	Los registros de la tabla AUDIT_VENTAS se actualizan con los valores correspondientes según la acción realizada (modificación, eliminación o creación).
Función	Realiza un seguimiento de todas las operaciones realizadas en la tabla VENTAS (inserciones,

	actualizaciones y eliminaciones), registrando los detalles de cada venta en la tabla de auditoría AUDIT_VENTAS junto con la acción correspondiente y la fecha de la operación.

Objeto	**TRIGGER TRG_LOG_ID_AUDIT_VENTAS**
Entradas	**Descripción**
Salidas	El campo LOG_ID del registro en AUDIT_VENTAS se asigna con un valor único generado por la secuencia.
Función	Asigna un identificador único al campo LOG_ID de los registros insertados en la tabla AUDIT_VENTAS utilizando una secuencia, lo que garantiza la integridad y unicidad de los registros en la tabla de auditoría.

Objeto	**PROCEDURE REGISTRAR_VENTA**
Entradas	Descripción
MONTO_IN	NUMBER
PRODUCTOS_IN	(VARCHAR2)
PAGO_IN	(VARCHAR2)
COMPRADOR_IN	(VARCHAR2)
VENDEDOR_IN	(VARCHAR2)
CANTIDADES_IN	NUMBER
Salidas	Mensaje de confirmación o error
Función	Registra una venta en la tabla TLM.VENTAS, verifica existencias en TLM.PRODUCTOS, y actualiza el inventario.

Objeto	**PROCEDURE LEER_VENTAS**
Entradas	Descripción
CRITERIO_IN	VARCHAR2
Salidas	Ventas registradas
Función	Lee y muestra las ventas almacenadas en la tabla TLM.VENTAS. Si se proporciona un criterio, filtra las ventas por comprador.

Objeto	**VIEW VISTA_VENTA**
Entradas	**Descripción**
ID_VENTA	NUMBER
MONTO	NUMBER
PRODUCTOS	VARCHAR2(50)
PAGO	VARCHAR2(25)
COMPRADOR	INT
VENDEDOR	INT
FECHA	DATE
CANTIDADES	INT
Salidas	La vista se llena con los datos seleccionados de la tabla VENTAS, mostrando información clave de cada venta.
Función	Muestra una vista de los datos de ventas, permitiendo consultar información detallada sobre el monto, productos, compradores, vendedores, y cantidades relacionadas con las ventas registradas.

Objeto	**TABLE VER_VENTAS**
Entradas	**Descripción**

Salidas	Cursor que contiene los registros de la vista VISTA_VENTA.
Función	Este procedimiento abre un cursor de referencia para recuperar todos los registros de la vista VISTA_VENTA.

Objeto	**TABLE BUSCAR_VENTA**
Entradas	**Descripción**
Salidas	Cursor que contiene el registro de la venta con el ID_VENTA especificado.
Función	Este procedimiento abre un cursor de referencia para recuperar un registro de la vista VISTA_VENTA donde el campo ID_VENTA coincide con el valor proporcionado como entrada (IDV).

Objeto	**PROCEDURE ACTUALIZAR_INVENTARIO**
Entradas	Descripción
CODIGO_IN	INT
CANTIDAD_IN	INT
Salidas	Mensaje de confirmación o error
Función	Actualiza la cantidad de un producto en el inventario sumando la cantidad indicada.

Objeto	**FUNCTION CALCULAR_PRECIO_TOTAL**
Entradas	Descripcion
PRECIO_IN	Number
CANTIDAD_IN	Number
Salidas	Precio total calculado (PRECIO_IN * CANTIDAD_IN).
Función	Calcula el precio total de una compra multiplicando el precio unitario por la cantidad de productos.

Objeto	**FUNCTION PRODUCTOS_AGOTANDOSE**
Entradas	Descripción
LIMITE_IN	INT
Salidas	Cursor con los productos cuya cantidad es menor o igual al límite especificado.
Función	Devuelve un cursor con los productos cuyo inventario es menor o igual al límite especificado.

Objeto	**FUNCTION CALCULAR_MONTO**
Entradas	Descripción
MONTO_IN	NUMBER
IMPUESTO_IN	NUMBER
DESCUENTO_IN	NUMBER
Salidas	Monto final calculado: (MONTO_IN + (MONTO_IN * IMPUESTO_IN / 100)) - DESCUENTO_IN
Función	Calcula el monto final aplicando un impuesto y un descuento al monto inicial.

Objeto	**PROCEDURE ACTUALIZAR_VENTA**
Entradas	Descripción
ID_IN	NUMBER
MONTO_IN	NUMBER
PRODUCTOS_IN	(VARCHAR2)
PAGO_IN	(VARCHAR2)
COMPRADOR_IN	(VARCHAR2)
VENDEDOR_IN	(VARCHAR2)
CANTIDADES_IN	NUMBER
Salidas	Mensaje de confirmación o error
Función	Actualiza los datos de una venta existente en la tabla TLM.VENTAS.

Objeto	**PROCEDURE ELIMINAR_VENTA**

Entradas	Descripción
ID_IN	NUMBER
Salidas	Mensaje de confirmación o error
Función	Elimina una venta de la tabla TLM.VENTAS utilizando su ID.

Objeto	**TABLE PRODUCTOS**
Entradas	**Descripción**
CODIGO	NUMBER NOT NULL (PK)
NOMBRE	VARCHAR2(50)
DESCRIPCION	CLOB
CANTIDAD	NUMBER
PRECIO	NUMBER
Salidas	Tabla creada
Función	Tabla para almacenar los productos dentro de la aplicación

Objeto	**TABLE AUDIT_PRODUCTOS**
Entradas	**Descripción**
LOG_ID	NUMBER PK
CODIGO	NUMBER
NOMBRE	VARCHAR2(50)
DESCRIPCION	CLOB
CANTIDAD	NUMBER
PRECIO	NUMBER
ACCION	VARCHAR2(20)
FECHA_ACCION	DATE
Salidas	Tabla creada
Función	Tabla de auditoria para almacenar logs de dentro de la tabla productos

Objeto	**TRIGGER TRG_AUDIT_PRODUCTOS**
Entradas	**Descripción**
Salidas	INSERT de la accion dentro de la tabla audit_productos
Función	Trigger programado para ejecutarse cuando haya un cambio en la tabla Productos y resgistrar su accion en la tabla de auditoria

Objeto	**PROCEDURE INSERTAR_PRODUCTO**
Entradas	**Descripción**
CODIGO_IN	NUMBER
NOMBRE_IN	VARCHAR2
DESCRIPCION_IN	CLOB
CANTIDAD_IN	NUMBER
PRECIO_IN	NUMBER
Salidas	Confirmación de producto insertado correctamente
Función	Procedimiento utilizado para la inserción de datos en la tabla Productos

Objeto	**VIEW VISTA_PRODUCTO**
Entradas	**Descripción**

CODIGO	(NUMBER)
NOMBRE	(VARCHAR2)
DESCRIPCION	(CLOB)
CANTIDAD	(NUMBER)
PRECIO	(NUMBER)
Salidas	Vista de los datos de la tabla productos
Función	Vista dentro de la base de datos para obtener todos los detalles de los datos almacenados en la tabla productos

Objeto	**PROCEDURE VER_PRODUCTOS**
Entradas	**Descripción**
Salidas	Cursor que contiene un select general de la tabla Productos
Función	Procedimiento que permite a la aplicación tener un cursor con los datos de la tabla productos para luego ser consumidos

Objeto	**PROCEDURE BUSQUEDA_PRODUCTOS**
Entradas	**Descripción**
CODIGO_IN	NUMBER
Salidas	Cursor que contiene un select tomando en cuenta el codigo dado de la tabla Productos
Función	Procedimiento que permite a la aplicación tener un cursor con informacion especifica de los datos de la tabla Productos para luego ser consumidos

Objeto	**PROCEDURE ACTUALIZAR_PRODUCTO**
Entradas	**Descripción**
CODIGO_IN	(NUMBER)
NOMBRE_IN	(VARCHAR2)
DESCRIPCION_IN	(CLOB)
CANTIDAD_IN	(NUMBER)
PRECIO_IN	(NUMBER)
Salidas	Mensaje de confirmacion del elemento actualizado
Función	Procedimiento que permite la modificacion de datos dentro de la tabla producto

Objeto	**PROCEDURE ELIMINAR_PRODUCTO**
Entradas	**Descripción**
CODIGO_IN	(NUMBER)
Salidas	Mensaje de confirmación de la eliminación del elemento
Función	Procedimiento que permite la eliminacion de datos dentro de la tabla producto

Objeto	**TABLE EMPLEADOS**
Entradas	**Descripción**
CEDULA	NUMBER NOT NULL (PK)

NOMBRE	VARCHAR2(50)
TIPO	VARCHAR2(20)
TELEFONO	VARCHAR2(15)
DIRECCION	CLOB
CORREO	VARCHAR2(50)
PUESTO	VARCHAR2(50)
SALARIO	NUMBER
USERNAME	VARCHAR2(30)
PASSWORD	VARCHAR2(30)
Salidas	Tabla creada
Función	Tabla para almacenar los empleados que pertenecen a la empresa

Objeto	**TABLE AUDIT_EMPLEADOS**
Entradas	**Descripción**
LOG_ID	NUMBER PK
CEDULA	NUMBER
NOMBRE	VARCHAR2(50)
TIPO	VARCHAR2(20)
TELEFONO	VARCHAR2(15)
DIRECCION	CLOB
CORREO	VARCHAR2(50)
PUESTO	VARCHAR2(50)
SALARIO	NUMBER
USERNAME	VARCHAR2(30)
PASSWORD	VARCHAR2(30)
ACCION	VARCHAR2(20)
FECHA_ACCION	DATE
Salidas	Tabla creada
Función	Tabla de auditoria para almacenar logs de dentro de la tabla empleados

Objeto	**TRIGGER TRG_AUDIT_EMPLEADOS**
Entradas	**Descripción**
Salidas	INSERT de la accion dentro de la tabla audit_empleados
Función	Trigger programado para ejecutarse cuando haya un cambio en la tabla Empleados y resgistrar su accion en la tabla de auditoria

Objeto	**VIEW VISTA_EMPLEADOS**
Entradas	**Descripción**
CEDULA	(NUMBER)
NOMBRE	(VARCHAR2)
TIPO	(VARCHAR2)
TELEFONO	(VARCHAR2)
DIRECCION	(CLOB)
CORREO	(VARCHAR2)
PUESTO	(VARCHAR2)

Salidas	Vista de los datos de la tabla empleados
Función	Vista dentro de la base de datos para obtener todos los detalles de los datos almacenados en la tabla empleados

Objeto	**PROCEDURE VER_EMPLEADOS**
Entradas	**Descripción**
Salidas	Cursor que contiene un select general de la tabla Empleados
Función	Procedimiento que permite a la aplicación tener un cursor con los datos de la tabla Empleados para luego ser consumidos

Objeto	**PROCEDURE BUSQUEDA_EMPLEADOS**
Entradas	**Descripción**
CEDULA_IN	NUMBER
Salidas	Cursor que contiene un select tomando en cuenta la cedula dada de la tabla Empleados
Función	Procedimiento que permite a la aplicación tener un cursor con informacion especifica de los datos de la tabla Empleados para luego ser consumidos

Objeto	**PROCEDURE INSERTAR_EMPLEADO**
Entradas	**Descripción**
CEDULA_IN	(NUMBER)
NOMBRE_IN	(VARCHAR2)
TIPO_IN	(VARCHAR2)
TELEFONO_IN	(VARCHAR2)
DIRECCION_IN	(CLOB)
CORREO_IN	(VARCHAR2)
PUESTO_IN	(VARCHAR2)
SALARIO_IN	(NUMBER)
USERNAME_IN	(VARCHAR2)
PASSWORD_IN	(VARCHAR2)
Salidas	Mensaje de confirmación de la inserción del elemento
Función	Procedimiento que permite agregar datos dentro de la tabla empleados

Objeto	**PROCEDURE ACTUALIZAR_EMPLEADO**
Entradas	**Descripción**
CEDULA_IN	(NUMBER)
NOMBRE_IN	(VARCHAR2)
TIPO_IN	(VARCHAR2)
TELEFONO_IN	(VARCHAR2)
DIRECCION_IN	(CLOB)

CORREO_IN	(VARCHAR2)
PUESTO_IN	(VARCHAR2)
SALARIO_IN	(NUMBER)
USERNAME_IN	(VARCHAR2)
PASSWORD_IN	(VARCHAR2)
Salidas	Mensaje de confirmacion de la actualizacion del elemento
Función	Procedimiento que permite editar datos dentro de la tabla empleados

Objeto	**PROCEDURE ELIMINAR_EMPLEADO**
Entradas	**Descripción**
CEDULA_IN	(NUMBER)
Salidas	Mensaje de confirmación de la eliminación del elemento
Función	Procedimiento que permite eliminar datos dentro de la tabla empleados

Objeto	**PROCEDURE REPORTE_PRODUCTOS_DINAMICO**
Entradas	**Descripción**
FILTRO_CANTIDAD	(VARCHAR2)
FILTRO_PRECIO	(VARCHAR2)
ORDEN_COL	(VARCHAR2)
ORDEN_TIPO	(VARCHAR2)
Salidas	Lista detallada de todos los productos con sus precios y cantidades
Función	Procedimiento para generar reportes dinámicos de productos usando filtros personalizados.

Objeto	**PACKAGE PKG_VENTAS**
Componentes	**Descripción**
PROCEDURE REGISTRAR_VENTAS	Registra una venta validando existencias en inventario
PROCEDURE LEER_VENTAS	Lee las ventas de la aplicación
PROCEDURE ACTUALIZAR_VENTAS	Actualiza la tabla ventas con los datos actuales
PROCEDURE ELIMINAR_VENTAS	Elimina las ventas registradas
FUNCTION CALCULAR_MONTO	Calcula el monto final de una venta
Salidas	
Función	Agrupa procedimientos y funciones relacionados con ventas

Objeto	**PACKAGE PKG_INVENTARIO**
Componentes	**Descripción**
PROCEDURE ACTUALIZAR_INVENTARIO	Aumenta la cantidad de un producto específico en el inventario.
FUNCTION CALCULAR_PRECIO_TOTAL	Calcula el precio total de un producto multiplicando su precio unitario por la cantidad.
Salidas	
Función	Agrupa procedimientos y funciones para gestionar inventarios

Objeto	**EXCEPTION RAISE_APPLICATION_ERROR**
Componentes	**Descripción**

Salidas	
Función	Utilizada en REGISTRAR_VENTA para manejar errores como falta de existencias o datos incompletos

Objeto	**EXCEPTION NO_DATA_FOUND**
Componentes	**Descripción**
Salidas	
Función	Captura errores cuando no se encuentra un producto en el inventario o un cliente en CLIENTES

Objeto	**EXCEPTION OTHERS**
Componentes	**Descripción**
Salidas	
Función	Captura cualquier otro error no manejado explícitamente.

Objeto	**DYNAMIC SQL REPORTE_PRODUCTOS**
Componentes	**Descripción**
Salidas	
Función	Genera consultas dinámicas sobre la tabla PRODUCTOS basadas en condiciones específicas.

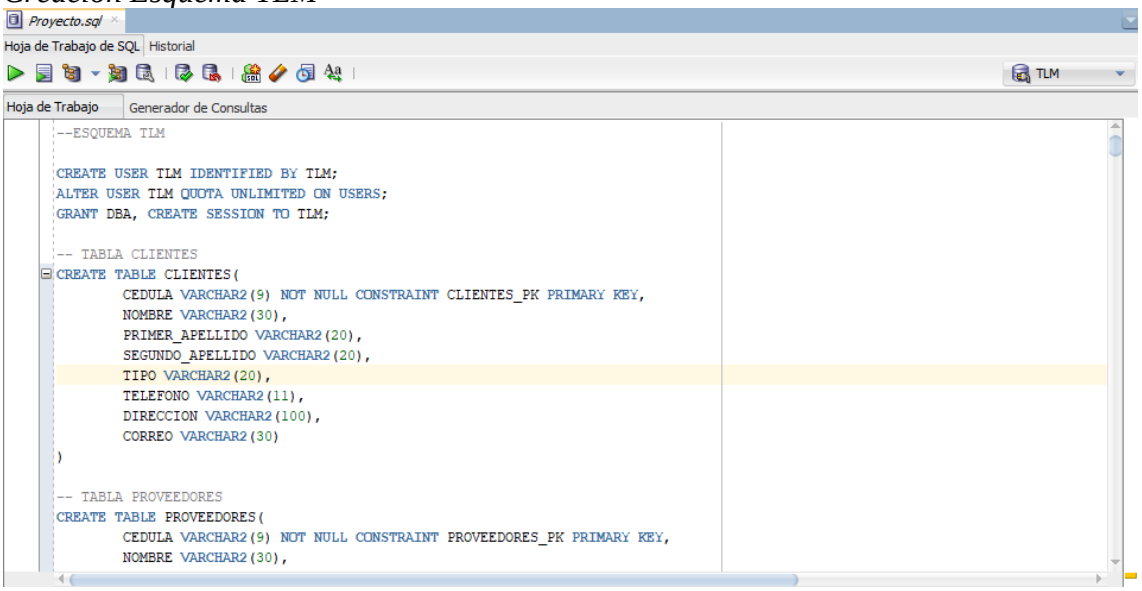
Desarrollo

Evidencias

Se genera el Script para la generación del esquema que se trabajará para el proyecto en la base de datos Oracle, que el equipo ha decido denominar TLM, por las siglas “Tienda La Moda”.

Figura 2

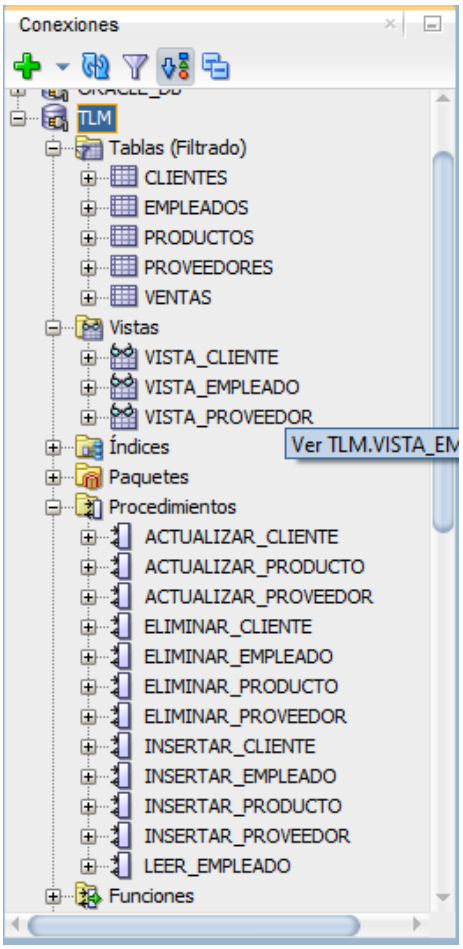
Creación Esquema TLM



Todos los scripts ejecutados

Figura 3

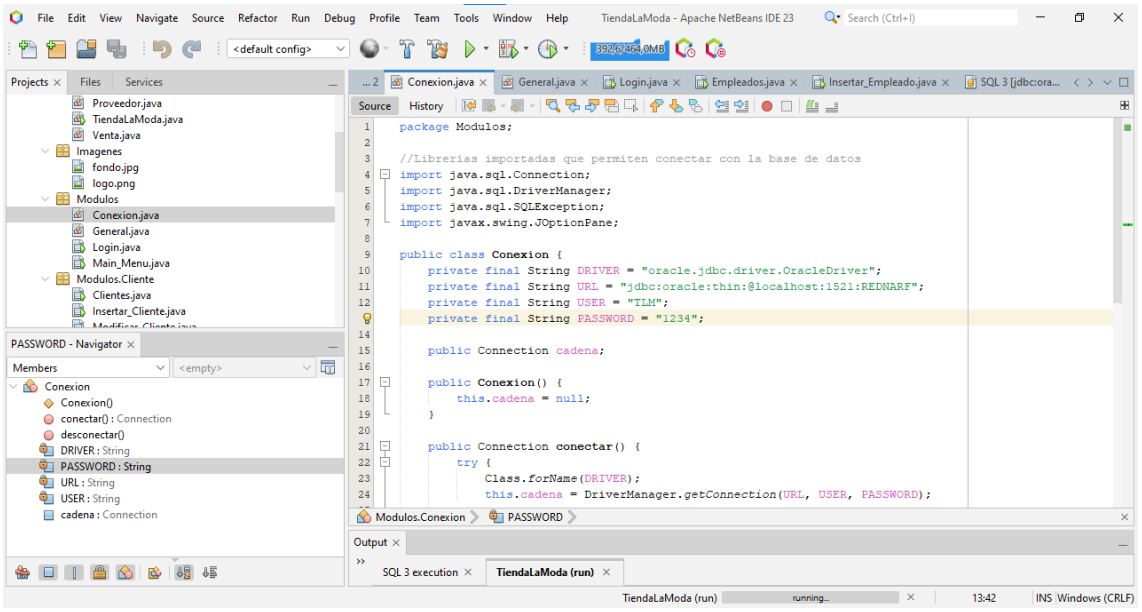
Esquema TLM



En el aplicativo trabajado en NetBeans con el lenguaje de programación Java, generamos la conexión a la base de datos mediante el código el programa mediante la clase Conexión que será utilizada y convocada a lo largo del proyecto para su integración con la base de datos.

Figura 4

Conexión DB y APP



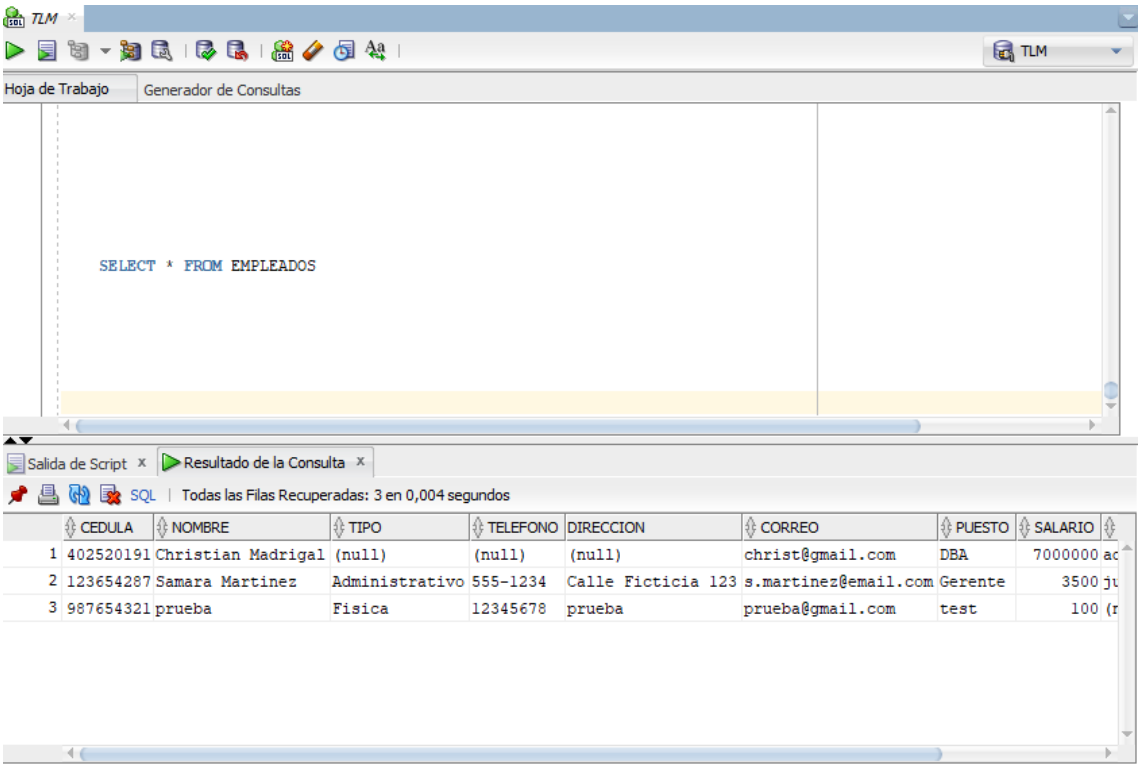
Se realiza una prueba con la consulta a la tabla Empleados.

Figura 5

Empleados App

Figura 6

Tabla empleados DB



También se generan los procedimientos almacenados básicos para interactuar con la base de datos con el aplicativo.

Figura 7

SP DB TLM

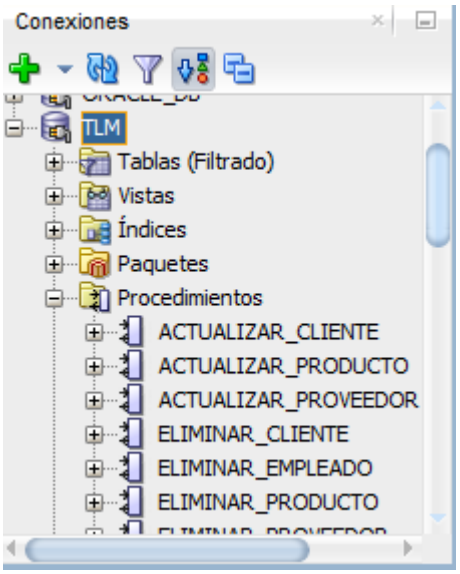
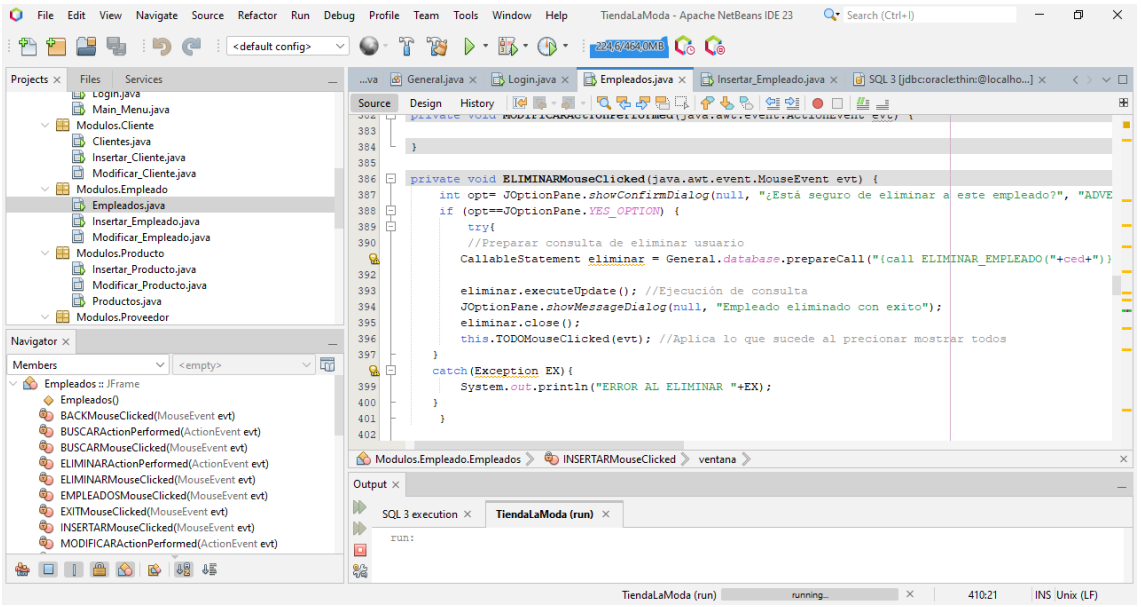


Figura 8

Call SP APP



Conclusiones y Recomendaciones

Conclusiones

- **Conexión segura y confiable:** La implementación de JDBC para conectar Java con Oracle demostró ser una solución eficaz, garantizando la seguridad y la integridad de los datos sensibles, como las transacciones de ventas y la información de los empleados.

- **Escalabilidad futura:** El sistema fue diseñado para ser escalable, lo que permitirá futuras ampliaciones, como la adición de nuevas funcionalidades o la expansión del negocio a nuevas ubicaciones.

- **Integración eficiente:** El uso de Java en conjunto con Oracle permitió crear un sistema robusto y escalable, ideal para gestionar los diferentes módulos de la tienda "Tienda La Moda".

Recomendaciones

- **Capacitación continua:** Es recomendable capacitar continuamente al personal en el uso del sistema para que puedan aprovechar al máximo las funcionalidades de este y evitar errores operativos.
- **Monitoreo del rendimiento:** Se sugiere establecer un sistema de monitoreo del rendimiento del sistema, para detectar y solucionar de manera proactiva cualquier problema de lentitud o sobrecarga.
- **Mantenimiento de seguridad:** Mantener actualizadas las capas de seguridad de Java y Oracle es fundamental para proteger los datos confidenciales de la tienda, especialmente en el manejo de transacciones y datos personales de los clientes y empleados.

Bibliografía

Referencias

Anónimo, U. (2011, 25 febrero). Tutorial básico de bases de datos en Java mediante JDBC - Adictos al trabajo. Adictos Al Trabajo. https://adictosaltrabajo.com/2011/02/25/tutorial-basico-jdbc/

Hernández, Y. (2023, 23 enero). Características de Java como lenguaje de programación. Tutoriales Dongee. https://www.dongee.com/tutoriales/caracteristicas-de-java-como-lenguaje-de-programacion/

Nieva, G. (2017, 16 mayo). Guía básica para usar Netbeans. dCodinGames. https://dcodingames.com/guia-basica-para-usar-netbeans/

Coppola M. (2023,13 febrero) Qué es Java, para qué sirve, características e historia. Hubspot. https://blog.hubspot.es/website/que-es-java

Oracle (2024) Software Java. Oracle. https://www.oracle.com/es/java/