Manual Tecnico

16/03/2023

Carlos Benjamin Pac Flores
Manejo e Implementación de Archivos
Centro Universitario de Occidente
Universidad San Carlos de Guatemala

Introducción:

Propuesta de programa para el manejo de ventas y control de productos para la empresa “Electronic-Homes” en donde se pone en práctica el uso de la herramienta de base de datos relacional sql Postgres y de un interfaz de backend y frontend desarrollada en Java y Java swing respectivamente.

Tecnologia Aplicadas:

PostgreSQL:

PostgreSQL (también conocido como Postgres) es un sistema de gestión de bases de datos relacionales de código abierto y gratuito. Es uno de los sistemas de gestión de bases de datos más avanzados y populares del mundo, y se utiliza en una amplia variedad de aplicaciones empresariales y de software.

PostgreSQL ofrece una amplia gama de funciones avanzadas, incluyendo soporte para transacciones ACID, escalabilidad, seguridad y soporte para múltiples lenguajes de programación y plataformas. Además, PostgreSQL es altamente personalizable y se puede integrar con una variedad de herramientas y sistemas, lo que lo hace ideal para empresas y organizaciones que necesitan una base de datos confiable y escalable para sus aplicaciones y sistemas.

Java 11:

Java 11 es una versión de Java Standard Edition (Java SE) que fue lanzada en septiembre de 2018. Es una versión a largo plazo (LTS) y está diseñada para proporcionar una plataforma estable y confiable para desarrollar aplicaciones de software en múltiples plataformas y dispositivos.

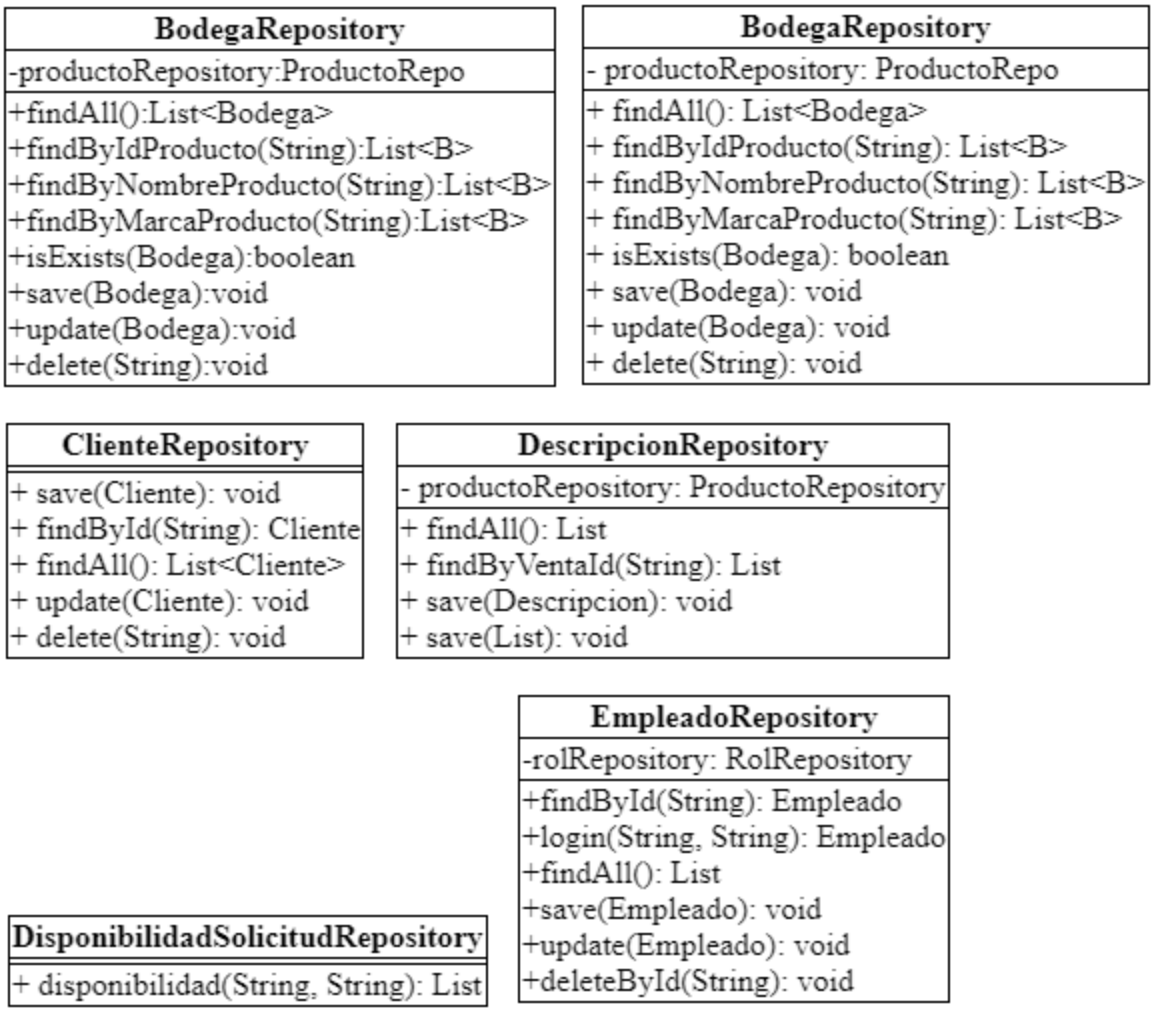
Java 11 incluye una serie de mejoras y nuevas características, como la mejora del rendimiento de la JVM (Java Virtual Machine), la inclusión de nuevas API y funcionalidades, la adición de nuevas bibliotecas y herramientas, y la implementación de nuevas características de seguridad y privacidad.

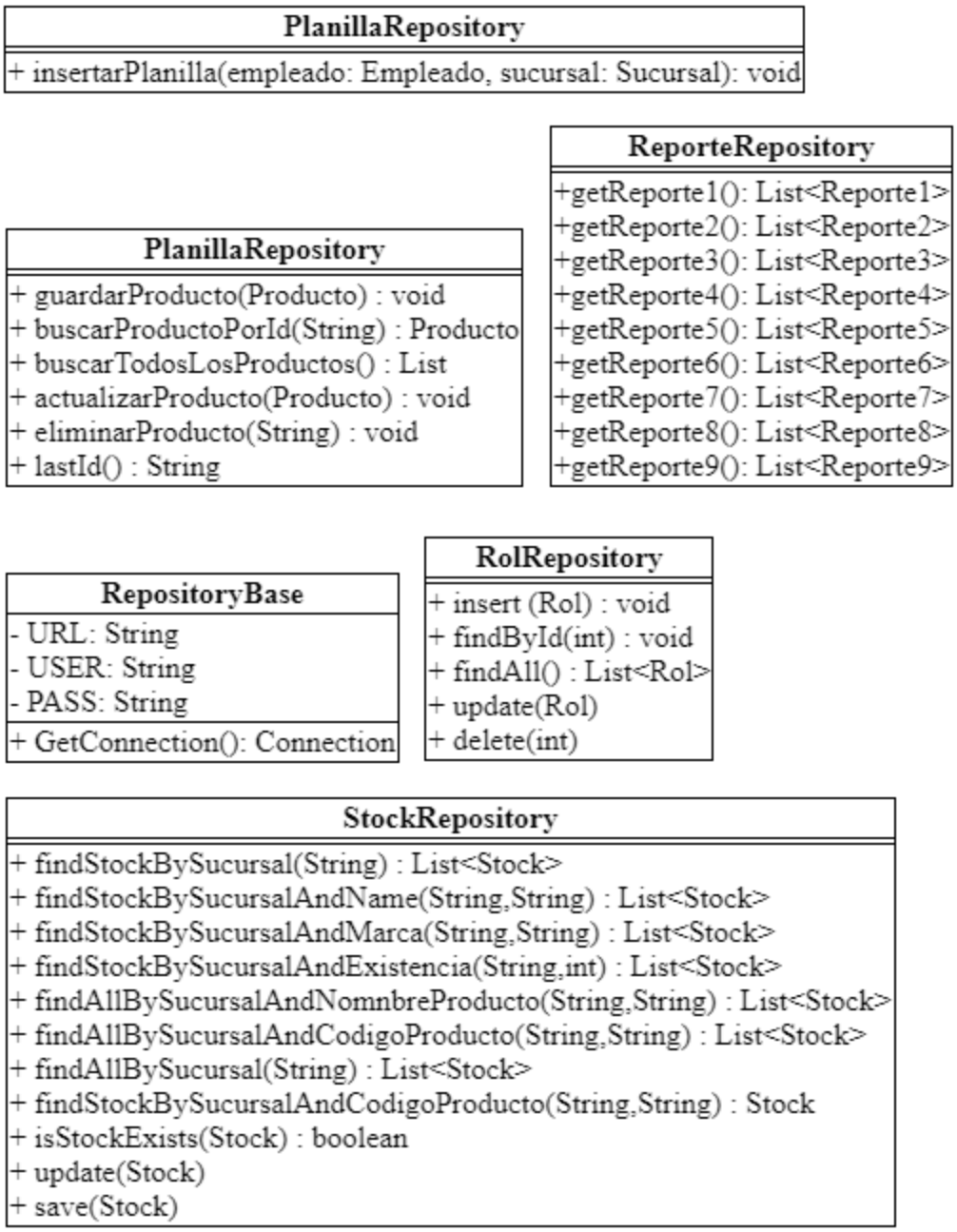
Además, Java 11 incluye algunas características adicionales, como el soporte para variables locales en expresiones lambda, la inclusión de una nueva biblioteca para procesamiento de HTTP (JEP 321), la eliminación de algunos módulos de plataforma obsoletos y la actualización de algunas bibliotecas y dependencias.

En resumen, Java 11 es una versión importante de Java SE que ofrece mejoras significativas en el rendimiento, funcionalidad, seguridad y privacidad para desarrollar aplicaciones de software en múltiples plataformas y dispositivos.

Estructura del Proyecto JAVA:

Diagrama de Clases de los bloques importantes del proyectos del sistema de productos, se excluyen las clases correspondientes a las ventanas los cuales solo manejas la captación de información.




PlanillaRepository
+ insertarPlanilla(empleado: Empleado, sucursal: Sucursal): void
ReporteRepository
+getReporte1(): List<Reporte1>
+getReporte2(): List<Reporte2>
+getReporte3(): List<Reporte3>
+getReporte4(): List<Reporte4>
+getReporte5(): List<Reporte5>
+getReporte6(): List<Reporte6>
+getReporte7(): List<Reporte7>
+getReporte8(): List<Reporte8>
+getReporte9(): List<Reporte9>
PlanillaRepository
+ guardarProducto(Producto) : void
+ buscarProductoPorId(String) : Producto
+ buscarTodosLosProductos() : List
+ actualizarProducto(Producto) : void
+ eliminarProducto(String) : void
+ lastId() : String
RepositoryBase
- URL: String
- USER: String
- PASS: String
+ GetConnection(): Connection
RolRepository
+ insert (Rol) : void
+ findById(int) : void
+ findAll() : List<Rol>
+ update(Rol)
+ delete(int)
StockRepository
+ findStockBySucursal(String) : List<Stock>
+ findStockBySucursalAndName(String,String) : List<Stock>
+ findStockBySucursalAndMarca(String,String) : List<Stock>
+ findStockBySucursalAndExistencia(String,int) : List<Stock>
+ findAllBySucursalAndNomnbreProducto(String,String) : List<Stock>
+ findAllBySucursalAndCodigoProducto(String,String) : List<Stock>
+ findAllBySucursal(String) : List<Stock>
+ findStockBySucursalAndCodigoProducto(String,String) : Stock
+ isStockExists(Stock) : boolean
+ update(Stock)
+ save(Stock)

SucursalRepository
+ getAll() : List<Sucursal> + findByNombre(String) : Sucursal + findById(String) : Sucursal + findByIdEmpleado(String) : Sucursal

VentaRepository
+ agregarVenta(Venta) : void + obtenerCostoUltimaCompra(String) : double + obtenerVentas() : List<Venta>

Descripcion
- producto: Producto - venta: Venta - cantidad: int - stock: Stock
+ Descripcion(producto: Producto, venta: Venta, cantidad: int) + getProducto(): Producto + setProducto(producto: Producto): void + getVenta(): Venta + setVenta(venta: Venta): void + getCantidad(): int + setCantidad(cantidad: int): void + getStock(): Stock + setStock(stock: Stock): void + toString(): String + equals(o: Object): boolean + hashCode(): int

DisponibilidadSolicitud
- isSucursal: boolean - cantidad: int - lugar: String - stockOrigen: Stock - stockDestino: Stock
+ DisponibilidadSolicitud(isSucursal: boolean, lugar: String, origen: Stock, destino: Stock, cantidad: int) + isIsSucursal(): boolean + setIsSucursal(isSucursal: boolean): void + getCantidad(): int + setCantidad(cantidad: int): void + getStockDestino(): Stock + setStockDestino(stockDestino: Stock): void + getStockOrigen(): Stock + setStockOrigen(stockOrigen: Stock): void + getLugar(): String + setLugar(lugar: String): void + toString(): String

Empleado
- nickname: String - passw: String - nombre: String - rol: Rol
+ Empleado(nickname: String, passw: String, nombre: String, rol: Rol) + getNickname(): String + setNickname(nickname: String): void + getPassw(): String + setPassw(passw: String): void + getNombre(): String + setNombre(nombre: String): void + getRol(): Rol + setRol(rol: Rol): void + toString(): String + equals(obj: Object): boolean + hashCode(): int

Producto
- id: String - nombre: String - marca: String - valor: double - descripcion: String
+ Producto():void + Producto(id: String,nombre: String, marca: String, valor: double,descripcion : String):void + getId(): String + setId(id: String):void + getNombre(): String + setNombre(nombre: String):void + getMarca(): String + setMarca(marca: String):void + getValor(): double + setValor(valor: double):void + getDescripcion():String + setDescripcion(descripcion:String):void + toString(): String + equals(o: Object):void + hashCode(): int

Rol
+ VENDEDOR: int + INVENTARIO: int + BODEGA: int + ADMIN: int - id: int - nombre: String
+ Rol(int, String) + getId(): int + setId(int): void + getNombre(): String + setNombre(String): void + toString(): String + equals(Object): boolean + hashCode(): int

Sucursal
- id: String - nombre: String
+ Sucursal(id: String, nombre: String) + getId(): String + setId(id: String): void + getNombre(): String + setNombre(nombre: String): void + toString(): String + equals(obj: Object): boolean + hashCode(): int

Venta
- id: String - fecha: String - cliente: Cliente - empleado: Empleado - descuento: double - descripcion: List - valor: double - sucursal: String
+ Venta(id: String, fecha: String, cliente: Cliente, empleado: Empleado,descuento: double,descripcion: List,valor: double,sucursal: String) + getId(): String + setId(id: String) + getFecha(): String + setFecha(fecha: String) + getCliente(): Cliente + setCliente(cliente: Cliente) + getEmpleado(): Empleado + setEmpleado(empleado: Empleado) + getDescuento(): double + setDescuento(descuento: double) + getDescripcion():List + setDescripcion(descripcion: List) + getValor(): double + setValor(valor:double) + getSucursal():String + setSucursal(sucursal: String) + toString(): String + equals(o: Object) + hashCode(): int

Reporte1
- id: String - nombre: String - marca: String - valor: double - cantidad: int
+ Reporte1(id: String,nombre: String,marca: String,valor: double,cantidad: int) + getId(): String + setId(id: String): void + getNombre(): String + setNombre(nombre: String): void + getMarca(): String + setMarca(marca: String) + getValor(): double + setValor(valor: double) + getCantidad(): int + setCantidad(cantidad:int): void + toString(): String + hashCode(): int + equals(Object): boolean

Reporte2
- nit: String - nombre: String - gastado: double
+ Reporte2(nit: String, nombre: String, gastado: double) + getNit(): String + setNit(nit: String): void + getNombre(): String + setNombre(nombre: String): void + getGastado(): double + setGastado(gastado: double): void + toString(): String + hashCode(): int + equals(obj: Object): boolean

Reporte3
- id: String - nombre: String - cantidad: int
+ Reporte3(String, String, int) + getId(): String + setId(String): void + getNombre(): String + setNombre(String): void + getCantidad(): int + setCantidad(int): void + toString(): String + hashCode(): int + equals(Object): boolean

Reporte4
-id: String -nombre: String -ingreso: double
+Reporte4(id: String, nombre: String, ingreso: double) +getId(): String +setId(id: String): void +getNombre(): String +setNombre(nombre: String): void +getIngreso(): double +setIngreso(ingreso: double): void +toString(): String

Reporte5
- id: String - nombre: String - cantidad: int
+ Reporte5(id: String, nombre: String, cantidad: int) + getId(): String + setId(id: String): void + getNombre(): String + setNombre(nombre: String): void + getCantidad(): int + setCantidad(cantidad: int): void + toString(): String + equals(o: Object): boolean + hashCode(): int

Reporte6
- id: String - nombre: String - valor: double
+ Reporte6(id: String, nombre: String, valor: double) + getId(): String + setId(id: String): void + getNombre(): String + setNombre(nombre: String): void + getValor(): double + setValor(valor: double): void + toString(): String

Reporte7
- id: String - nombre: String - marca: String - valor: double
+ Reporte7(id: String,nombre: String, marca: String, valor: double) + getId(): String + setId(id: String): void + getNombre(): String + setNombre(nombre: String): void + getMarca(): String + setMarca(marca: String): void + getValor(): double + setValor(valor: double): void + toString(): String

Reporte8
-id: String -nombre: String -marca: String -cantidad: int
+Reporte8(id: String, nombre: String, marca: String, cantidad: int) +getId(): String +setId(id: String): void +getNombre(): String +setNombre(nombre: String): void +getMarca(): String +setMarca(marca: String): void +getCantidad(): int +setCantidad(cantidad: int): void +toString(): String

Reporte9
- id: String - nombre: String - marca: String - ingreso: double
+Reporte9(id: String, nombre: String, marca: String, ingreso: double) +getId(): String +setId(id: String): void +getNombre(): String +setNombre(nombre: String): void +getMarca(): String +setMarca(marca: String): void +getIngreso(): double +setIngreso(ingreso: double): void +toString(): String

Encriptar
+encriptar(password)

DateManagment
+getIdDateTime() +currentDateTime() +currentDate()

Reportes
- formato: DecimalFormat - dateManagment: DateManagment
+ genReporte1(reporte1: List<Reporte1>, empleado: Empleado): JasperPrint + genReporte2(reporte2: List<Reporte2>, empleado: Empleado): JasperPrint + genReporte3(reporte3: List<Reporte3>, empleado: Empleado): JasperPrint + genReporte4(reporte4: List<Reporte4>, empleado: Empleado): JasperPrint + genReporte5(reporte5: List<Reporte5>, empleado: Empleado): JasperPrint + genReporte6(reporte6: List<Reporte6>, empleado: Empleado): JasperPrint + genReporte7(reporte7: List<Reporte7>, empleado: Empleado): JasperPrint

Estructura de la base de datos:

El código elaborado crea una base de datos llamada "electronichomes" y define seis esquemas dentro de ella: colaborador, infraestructura, mercancia, comercio y consumidor. Luego, se crean varias tablas dentro de cada uno de estos esquemas, como la tabla "empleado" en el esquema "colaborador", la tabla "sucursal" en el esquema "infraestructura", la tabla "producto" en el esquema "mercancia", y así sucesivamente.

También se definen varias relaciones entre las tablas, como las claves foráneas que conectan la tabla "empleado" con la tabla "rol" en el esquema "colaborador" y la tabla "stock" con la tabla "sucursal" en el esquema "mercancia". Además, se agregan algunos datos de prueba a las tablas, como los roles del sistema, las sucursales de la empresa y el cliente "Consumidor Final".

El fin de esta base de datos, es que la empresa pueda gestionar su personal, sucursales, inventario, ventas y clientes. Así también cada tabla tiene sus propias columnas que permiten la inserción de información específica.

```
CREATE DATABASE electronichomes;
\c electronichomes;
CREATE SCHEMA colaborador;
CREATE SCHEMA infraestructura;
CREATE SCHEMA mercancia;
CREATE SCHEMA comercio;
CREATE SCHEMA consumidor;

CREATE TABLE colaborador.rol(
    id INT NOT NULL,
    nombre VARCHAR(10) NOT NULL,
    PRIMARY KEY(id)
);
CREATE TABLE colaborador.empleado(
    nickname VARCHAR(20) NOT NULL,
    passw VARCHAR(200) NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    rol INT NOT NULL,
    PRIMARY KEY(nickname),
    Foreign Key (rol) REFERENCES colaborador.rol(id)
```

```
);
CREATE TABLE infraestructura.sucursal(
    id VARCHAR(10) NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    PRIMARY KEY(id)
);
CREATE TABLE colaborador.planilla(
    empleado VARCHAR(20) NOT NULL,
    sucursal VARCHAR(10) NOT NULL,
    PRIMARY KEY(empleado),
    Foreign Key (empleado) REFERENCES colaborador.empleado (nickname),
    Foreign Key (sucursal) REFERENCES infraestructura.sucursal(id)
);
CREATE TABLE mercancia.producto(
    id VARCHAR(10) NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    marca VARCHAR(50) NOT NULL,
    valor DECIMAL(10,2) NOT NULL,
    descripcion VARCHAR(200) NOT NULL,
    PRIMARY KEY(id)
);
CREATE TABLE mercancia.stock(
    producto VARCHAR(10) NOT NULL,
    sucursal VARCHAR(10) NOT NULL,
    cantidad INT NOT NULL,
    Foreign Key (producto) REFERENCES mercancia.producto(id),
    Foreign Key (sucursal) REFERENCES infraestructura.sucursal(id)
);
CREATE TABLE mercancia.bodega(
    producto VARCHAR(10) NOT NULL,
    cantidad INT NOT NULL,
    PRIMARY KEY(producto),
    Foreign Key (producto) REFERENCES mercancia.producto(id)
);
CREATE TABLE consumidor.cliente(
    nit VARCHAR(20) NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    PRIMARY KEY(nit)
);
CREATE TABLE comercio.venta(
    id VARCHAR(50) NOT NULL,
    fecha TIMESTAMP NOT NULL,
    cliente VARCHAR(20) NOT NULL,
    empleado VARCHAR(20) NOT NULL,
    descuento DECIMAL(10,2) NOT NULL,
    valor DECIMAL(10,2) NOT NULL,
    sucursal VARCHAR(10) NOT NULL,
```

```
    PRIMARY KEY(id),
    Foreign Key (empleado) REFERENCES colaborador.empleado(nickname),
    Foreign Key (cliente) REFERENCES consumidor.cliente(nit),
    Foreign Key (sucursal) REFERENCES infraestructura.sucursal(id)
);
CREATE TABLE comercio.descripcion(
    producto VARCHAR(10) NOT NULL,
    venta VARCHAR(50) NOT NULL,
    cantidad INT NOT NULL,
    Foreign Key (producto) REFERENCES mercancia.producto (id),
    Foreign Key (venta) REFERENCES comercio.venta (id)
);
```

La información de inicio del sistema estaba basada en las siguientes inserciones:

```
--Inserción de roles del sistema
INSERT INTO colaborador.rol VALUES(1,'VENDEDOR');
INSERT INTO colaborador.rol VALUES(2,'INVENTARIO');
INSERT INTO colaborador.rol VALUES(3,'BODEGA');
INSERT INTO colaborador.rol VALUES(4,'ADMIN');

--Ingreso de la sucursales de la empresa
INSERT INTO infraestructura.sucursal VALUES('S0001','Sucursal Central');
INSERT INTO infraestructura.sucursal VALUES('S0002','Sucursal Norte');
INSERT INTO infraestructura.sucursal VALUES('S0003','Sucursal Sur');

--Ingreso del cliente consumidor final
INSERT INTO consumidor.cliente VALUES('C/F','Consumidor Final');

--Administrador por defecto
INSERT INTO colaborador.empleado VALUES ('usuario17',
'a63f9709abc75bf8bd8f6e1ba9992573', 'Lucas Sánchez', 4);
```

También es de mencionar que el funcionamiento de la base de datos para conexión e interactuar con la base de datos es el usuario “admin”.

Diagrama Entidad Relación (Peter Chen):