

BYTE SQUAD

ESPECIFICACIÓN FUNCIONAL

FACTURACIÓN

Versión 1.0

Mayo 09 de 2019

|  |  |  |  |
| --- | --- | --- | --- |
| **REGISTRO DE CAMBIOS AL DOCUMENTO** | | | |
|  | | | |
| **Versión** | **Fecha** | **Descripción del cambio** | **Autor** |
| 1.0 | 29/04/2019 | Creación de la base de datos en postgres: DDL, DML. | Karen Pacasira, Jairo Cárdenas  ,Juan Ulloa  ,Lizeth Cipamocha |
| 1.1 | 30/04/2019 | Creación del Modelo: Mapeo de las tablas del schema facturación. | Juan Ulloa |
| 1.1.1 | 02/05/2019 | Separación de DDL y DML en la base de datos | Juan Ulloa |
| 1.1.2 | 06/05/2019 | Modelo de la base de datos y documentación del modelo | Karen Pacasira, Jairo Cárdenas  ,Juan Ulloa  ,Lizeth Cipamocha |
| 2.0 | 21/05/2019 | Creación CRUD - bean,Fachada y servicio | Laura castillo,Julian Franco,Ericka Sora,Juan Pablo Acevedo |
| 2.0.1 | 21/05/2019 | Creación CRUD - bean |  |
| 2.0.2 | 21/05/2019 | Creación CRUD - fachada y servicio |  |
| 2.1 | 21/05/2019 | Implementación reglas de negocios en la tabla Proveedor, | Laura Castillo, Julian Franco, |
| 2.1.1 | 21/05/2019 | Documentación Servicio | Juan Pablo Acevedo |
| 3.0 | 11/05/2019 | Documentación Web | Los 5 furiosos  (Paula Galeano,  Laura Estévez,  CristhianCamargo,  Santiago Jiménez) |
| 3.1 | 12/06/2019 | Mejora de documentación y descripción web de cada tabla y cada servicio web | Los 5 furiosos  (Paula Galeano,  Laura Estévez,  CristhianCamargo,  Santiago Jiménez) |

**TABLA DE CONTENIDO**

[INTRODUCCIÓN 4](#_heading=h.30j0zll)

[1.](#_heading=h.1fob9te) DESCRIPCIÓN DE LA BASE DE DATOS 5

[1.1.](#_heading=h.3znysh7) PROVEEDOR 5

[1.1.1.](#_heading=h.2et92p0) ATRIBUTOS 6

[1.1.2.](#_heading=h.tyjcwt) ÍNDICES 6

[1.2.](#_heading=h.3dy6vkm) CLIENTE 6

[1.2.1.](#_heading=h.1t3h5sf) ATRIBUTOS 6

[1.2.2.](#_heading=h.4d34og8) ÍNDICES 7

[1.2.3.](#_heading=h.2s8eyo1) REGISTROS DE CLIENTES 8

[1.3.](#_heading=h.17dp8vu) PROVEEDOR PRODUCTO 8

[1.3.1.](#_heading=h.3rdcrjn) ATRIBUTOS 8

[1.3.2. ÍNDICES 8](#_heading=h.26in1rg)

[1.4.](#_heading=h.lnxbz9) VENTA DETALLE 9

[1.4.1.](#_heading=h.35nkun2) ATRIBUTOS 9

[1.5.](#_heading=h.1ksv4uv) VENTA 10

[1.5.1.](#_heading=h.44sinio) ATRIBUTOS 10

[1.5.2.](#_heading=h.2jxsxqh) ÍNDICES 10

[1.6.](#_heading=h.z337ya) CATEGORIA PRODUCTO. 11

[1.6.1.](#_heading=h.3j2qqm3) ATRIBUTOS 11

[1.6.2.](#_heading=h.1y810tw) ÍNDICE 11

[1.7.](#_heading=h.4i7ojhp) PRODUCTO 11

[1.7.1.](#_heading=h.2xcytpi) ATRIBUTOS 12

[1.7.2.](#_heading=h.1ci93xb) ÍNDICE 12

2. DESCRIPCIÓN DEL WEB 27

2.1 DESCRIPCIÓN DE CONTROL CATEGORIA PRODUCTO 27

2.2 DESCRIPCIÓN DE CONTROL CLIENTE 27

2.3 DESCRIPCIÓN DE CONTROL PRODUCTO 27

2.4 DESCRIPCIÓN DE CONTROL PROVEEDOR 27

2.5 DESCRIPCIÓN DE CONTROL PROVEEDOR PRODUCTO 27

2.6 DESCRIPCIÓN DE CONTROL VENTA 27

2.7 DESCRIPCIÓN DE CONTROL VENTA DETALLE 27

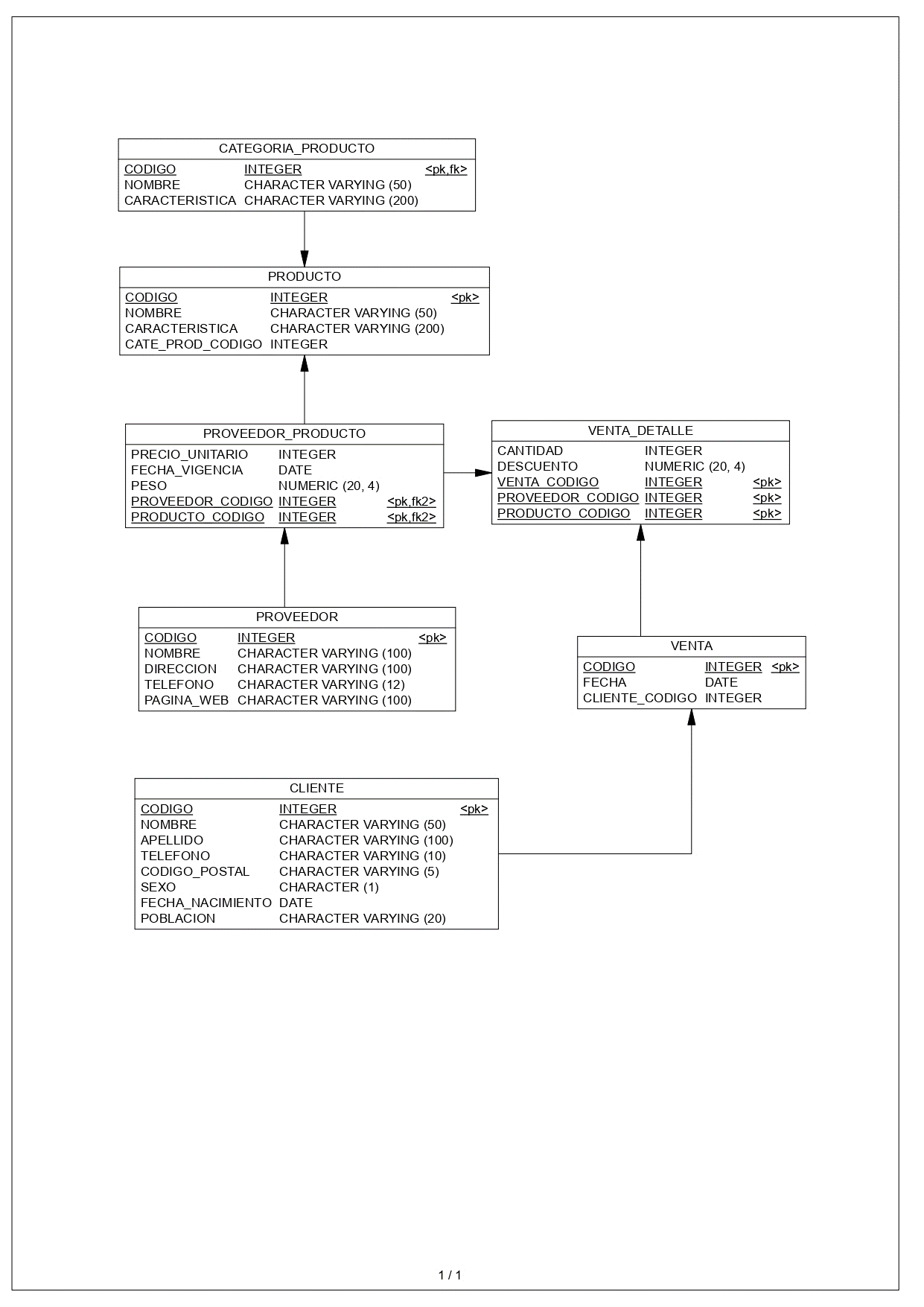
# INTRODUCCIÓN

En este documento se muestra el módulo de facturación, asimismo el archivo DDL y DML del esquema facturación y el JPA del Módulo. Este módulo hace referencia a la creación de facturas, clientes, proveedores, ventas y productos.

# DESCRIPCIÓN DE LA BASE DE DATOS

El esquema del módulo facturación continene 7 tablas.

Diagrama de entidad relación:

****

**Imagen1.Modelo entidad relación.**

# PROVEEDOR

La tabla proveedor contiene la información del proveedor, con el fin de llevar un control de los distribuidores.

# ATRIBUTOS

* **Código:** Código es de tipo numérico auto incremental y máximo de 6 dígitos. Establecido para ser único en la tabla proveedor, con el fin de identificar el proveedor.
* **Nombre:** El campo nombre es de tipo varchar máximo de 100 dígitos que se designan y se distinguen para cada proveedor con el fin que de diligenciar nombres y apellidos.
* **Dirección:** El campo dirección es de tipo varchar máximo de 100 dígitos. Se designan y se distinguen para cada persona con el fin de ingresar o diligenciar el domicilio de un proveedor o la ubicación de un edificio del mismo.
* **Teléfono:** El campo dirección es de tipo varchar máximo de 12 dígitos. Se designan y se distinguen para cada proveedor con el fin de ingresar una secuencia de dígitos utilizada para identificar una línea telefónica.
* **Página web:** El campo dirección es de tipo varchar máximo de 100 dígitos. Se designan para cada proveedor con el fin de ingresar un conjunto de caracteres que posibilita la búsqueda de una dirección exclusiva a un recurso que se encuentra disponible en el espacio virtual. Si el usuario no tiene una página web este campo podrá ser asignado como "null".

# ÍNDICES

* **DDL**

CREATE TABLE facturacion."PROVEEDOR"

(

"CODIGO" INTEGER NOT NULL,

"NOMBRE" CHARACTER VARYING (100) NOT NULL,

"DIRECCION" CHARACTER VARYING (100) NOT NULL,

"TELEFONO" CHARACTER VARYING (12) NOT NULL,

"PAGINA\_WEB" CHARACTER VARYING (100) NULL,

PRIMARY KEY ("CODIGO")

);

# CLIENTE

Esta tabla contiene los respectivos atributos con sus tipos de datos correspondientes para generar la factura con los datos a los que pertenece.

# ATRIBUTOS

* **Código:** es de tipo auto incremental que indica el tipo de id única e irrepetible, representa la llave primaria con el fin de identificar el cliente.

|  |  |
| --- | --- |
| **Abreviatura** | **Descripción** |
| COD | CODIGO |

**Tabla1. Tipo de código**

* **Nombre:** Es un atributo de tipo varying, con un tamaño máximo de 50 y representa el nombre del cliente.
* **Apellido:** Es un atributo de tipo varying, con un tamaño máximo de 50 y representa el apellido del cliente.
* **Teléfono:** Es un atributo de tipo varying, con un tamaño máximo de 10 y representa el número telefónico del cliente.
* **Código postal:** Es un atributo de tipo varying, con un tamaño máximo de 5 y representa la combinación de números que se asigna a una población y a las distintas zonas donde está el cliente.
* **Sexo:** Valor Carácter con un tamaño máximo de 1 dígito que indica el tipo de Sexo del cliente, únicamente puede ser ‘1’ o ‘0’, donde 1 es hombre y 0 es mujer.
* **Fecha de nacimiento:** Es un atributo de tipo date, representa la fecha de nacimiento del cliente cuyo formato es de año mes y día.
* **Población:** Es un atributo de tipo varying, con un tamaño máximo de 20, representa la población donde vive el cliente.

# INDICES

* **LLAVE PRIMARIA:** En la tabla de cliente la llave primaria representa el número el cual va a identificar cada cliente.
* **DDL TABLA CLIENTE**

Código ddl de cliente con los respectivos atributos y el tipo de dato de dicho atributo, de igual manera muestra la llave primaria de la tabla cliente el cual es código, es preciso señalar que esta tabla no tiene llave foránea ya que el cliente es una de las tablas padre.

CREATE TABLE facturacion."CLIENTE"

(

"CODIGO" INTEGER NOT NULL,

"NOMBRE" CHARACTER VARYING (50) NOT NULL,

"APELLIDO" CHARACTER VARYING (100) NOT NULL,

"TELEFONO" CHARACTER VARYING (10),

"CODIGO\_POSTAL" CHARACTER VARYING (5),

"SEXO" CHARACTER (1) NOT NULL,

"FECHA\_NACIMIENTO" DATE NOT NULL,

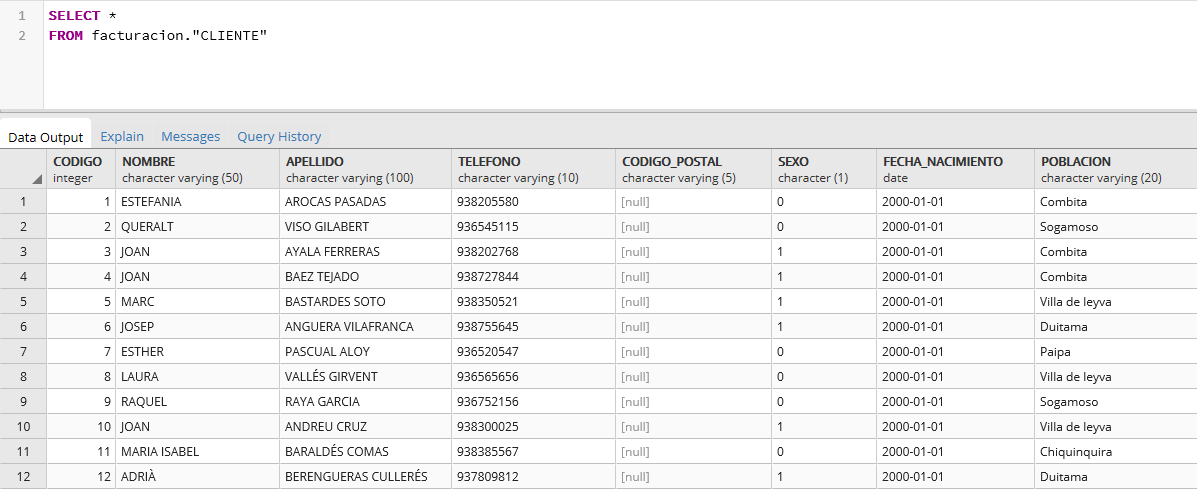
"POBLACION" CHARACTER VARYING (20) NOT NULL,

PRIMARY KEY ("CODIGO")

);

# REGISTROS DE CLIENTES

La siguiente imagen muestra la consulta de algunos clientes registrados en la base de datos en Postgres.

**Imagen2.Registro de clientes**

# PROVEEDOR PRODUCTO

Esta tabla contiene los productos que tiene cada proveedor, el valor que este pide por cada producto y el registro de venta que este hace.

# ATRIBUTOS

* **Proveedor Código:** Valor numérico que trae la información personal de un proveedor, también tiene como fin de identificar cada uno de los registros de la tabla proveedor producto; este número es auto incrementable.
* **Producto Código:** Valor numérico que trae la información de cada producto, también tiene como fin de identificar cada uno de los registros de la tabla proveedor producto; este número es auto incrementable.
* **Precio Unitario:** Valor numérico de tipo INTEGER, pertenece a el valor que el proveedor vende cada producto.
* **Peso:** Valor numérico que tiene una longitud mínima de 4 dígitos, máxima 20; representa el peso de los productos que vende cada proveedor.
* **Fecha Vigencia:** Valor de tipo date que identifica la fecha de vencimiento de cada producto.

# 1.3.2. ÍNDICES

* **Llave Primaria:** En esta tabla, proveedor código y producto código son las llaves primarias.
* **Llave Foránea:** En esta tabla las llaves foráneas son; Proveedor código la cual permite traer la información de un proveedor de la tabla proveedor y producto código la cual permite traer la información de un producto de la tabla producto.

* **DDL**

CREATE TABLE facturacion."PROVEEDOR\_PRODUCTO"

(

"PRECIO\_UNITARIO" INTEGER NOT NULL,

"FECHA\_VIGENCIA" DATE NOT NULL,

"PESO" NUMERIC (20, 4) NOT NULL,

"PROVEEDOR\_CODIGO" INTEGER,

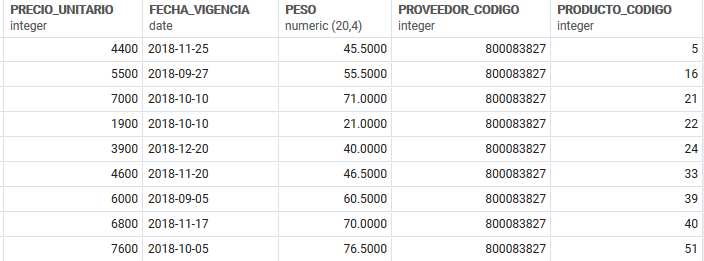
"PRODUCTO\_CODIGO" INTEGER,

PRIMARY KEY ("PROVEEDOR\_CODIGO", "PRODUCTO\_CODIGO")

);

* **Registros**

Registros de la tabla PROVEEDOR\_PRODUCTO



**Imagen3. Registros Proveedor producto.**

# VENTA DETALLE

Contiene la los detalles de las ventas en la cual se puede encontrar la cantidad, el descuento, y la información del producto

# ATRIBUTOS

* **Cantidad:** Valor numérico el cual corresponde a la cantidad de productos vendidos.
* **Descuento:** Valor numérico el cual responde al descuento aplicado al producto.
* **Venta Código:** Valor numérico el cual corresponderá a ser el código que va a heredar a la tabla venta.
* **Proveedor código:** Valor numérico el cual corresponderá a ser código que va heredar la tabla proveedor producto.
* **Producto Código:** Valor numérico el cual corresponderá el código que va a heredar a la tabla proveedor producto.
  + 1. **INDICES**
* **Llaves Foráneas:** Esta tabla cuenta con dos llaves foráneas, para realizar conexión con la tabla proveedor\_producto.
* **DDL**

CREATE TABLE facturacion."VENTA\_DETALLE"

(

"CANTIDAD" INTEGER NOT NULL,

"DESCUENTO" NUMERIC (20, 4) DEFAULT 0 NOT NULL,

"VENTA\_CODIGO" INTEGER NOT NULL,

"PROVEEDOR\_CODIGO" INTEGER,

"PRODUCTO\_CODIGO" INTEGER,

PRIMARY KEY ("VENTA\_CODIGO", "PROVEEDOR\_CODIGO", "PRODUCTO\_CODIGO")

);

# VENTA

Contiene la información de las ventas realizadas por el cliente.

# ATRIBUTOS

* **Código:** Valor numérico el cual corresponde al número de la factura, este es otorgado por la Dian.
* **Fecha:** Valor tipo date el cual corresponde a la fecha de elaboración de la factura.
* **Cliente Código:** Valor numérico el cual corresponderá a ser el código que va a heredar a la tabla cliente.

# INDICES

* **Llave Primaria:** En esta tabla la llave primaria es el código de la factura, este es único y es el encargado de identificar cada factura.
* **Llave Foránea: Esta** tabla cuenta con una llave foránea, para realizar conexión con la tabla cliente.
* **DDL**

CREATE TABLE facturacion."VENTA"

(

"CODIGO" INTEGER NOT NULL,

"FECHA" DATE NOT NULL,

"CLIENTE\_CODIGO" INTEGER,

PRIMARY KEY ("CODIGO")

);

# CATEGORIA\_PRODUCTO.

Contiene la información de las características del producto

* + 1. **ATRIBUTOS**
* **Código**: Es un atributo es de tipo integer, por lo cual es autoincremental, es obligatorio, también es la llave primaria de la tabla CATEGORIA\_PRODUCTO y es la llave foránea de la tabla PRODUCTO.
* **Nombre**: Es un atributo de tipo character varying con una longitud máxima de 50 caracteres, donde se especifican los nombres y no puede ser nulo este atributo.
* **Característica**: Es un atributo de tipo character varying con una longitud máxima de 200 caracteres y este atributo puede ser nulo, puede almacenar la descripción de cada concepto.
  + 1. **ÍNDICE**
* **Llave Primaria:** En esta tabla la llave primaria es el código la categoría de cada producto, este es único y es el encargado de identificar cada categoría.
* **DLL**

CREATE TABLE facturacion."CATEGORIA\_PRODUCTO"

(

"CODIGO" INTEGER NOT NULL,

"NOMBRE" CHARACTER VARYING (50) NOT NULL,

"CARACTERISTICA" CHARACTER VARYING (200) NOT NULL,

PRIMARY KEY ("CODIGO")

);

# PRODUCTO

Contiene la información del producto.

* + 1. **ATRIBUTOS**
* **CÓDIGO**: Es un atributo es de tipo Integer, por lo cual es auto incremental, es obligatorio y es la llave primaria de esta tabla.
* **NOMBRE**: Es un atributo de tipo character varying, con una longitud máxima de 50 caracteres, donde se especifican los nombres y no puede ser nulo este atributo.
* **CARACTERÍSTICA**: Es un atributo de tipo character varying con una longitud máxima de 200 caracteres y este atributo puede ser nulo, puede almacenar la descripción de cada concepto.
* **CATE\_PROD\_CODIGO**: Es un atributo de tipo Integer y es la llave heredada de la tabla CATEGORIA\_PRODUCTO**.**

# ÍNDICE

* **Llave Primaria:** En esta tabla la llave primaria es el código del producto, este es único y es el encargado de identificar cada producto.
* **Llave Foránea:** En esta tabla cuenta con una llave foránea, para realizar la conexión con la tabla CATEGORIA\_PRODUCTO.
* **DDL**

CREATE TABLE facturacion."PRODUCTO"

(

"CODIGO" INTEGER NOT NULL,

"NOMBRE" CHARACTER VARYING (50) NOT NULL,

"CARACTERISTICA" CHARACTER VARYING (200) NOT NULL,

"CATE\_PROD\_CODIGO" INTEGER NOT NULL,

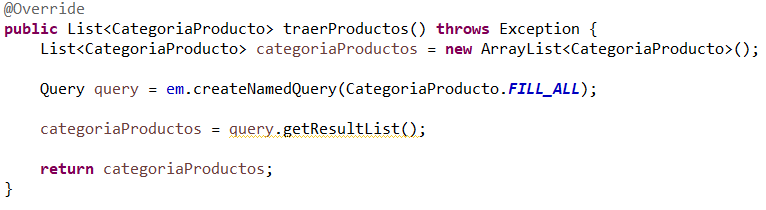
PRIMARY KEY ("CODIGO")

);

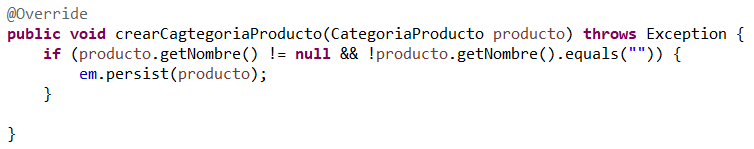
**2.** **Creación CRUD**

**2.0.1. Bean:**

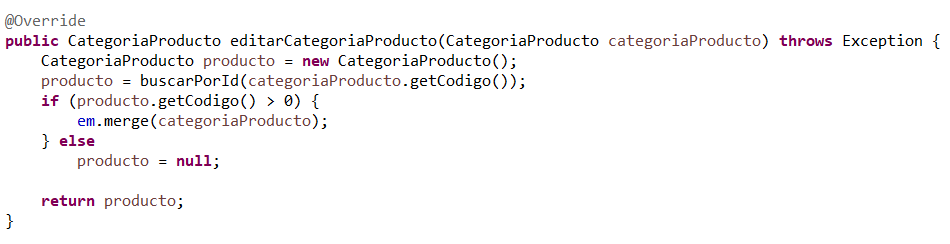
* **CRUD tabla CategoriaProducto:**
* **traerProductos:** Método de tipo List<CategoriaProducto> que crea un ArrayList de tipo CategoriaProducto



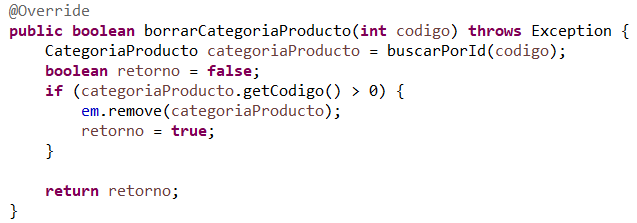
* **crearCategoriaProducto:** Método de tipo void que crea una categoría a un producto



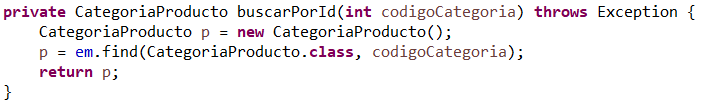
* **editarCategoriaProducto:** Método de tipo CategoriaProducto utilizado para actualizar la categoria



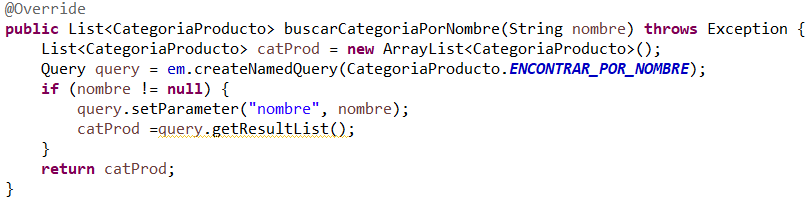
* **borrarCategoriaProducto:** Método de tipo boolean que borra la categoria de un producto



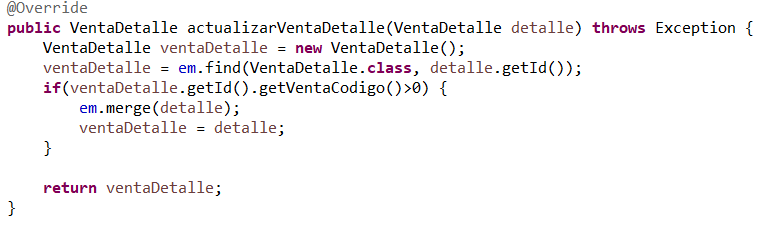
* **buscarPorId:** Método de tipo CategoriaProducto que realiza una búsqueda por id de la categoría



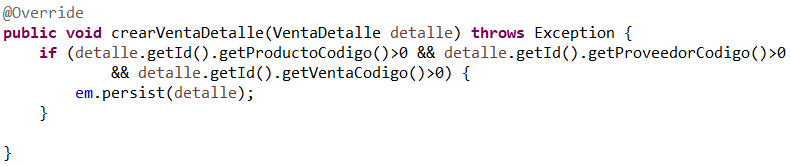
* **buscarCategoriaPorNombre:** Método de tipo List<CategoriaProducto> que realiza una búsqueda por el nombre de la categoría



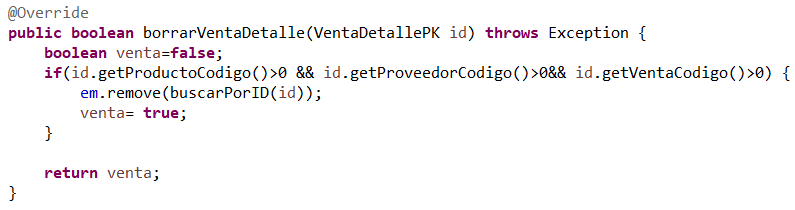
* **CRUD tabla VentaDetalle:**
* **actualizarVentaDetalle:** Método de tipo VentaDetalle utilizado para actualizar la venta detalle de una compra

****

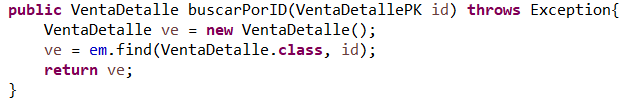
* **crearVentaDetalle:** Método de tipo void que crea un detalle de venta

****

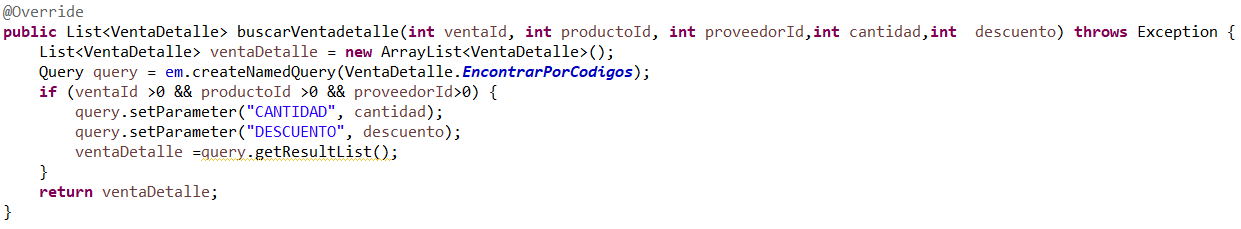
* **borrarVentaDetalle:** Método de tipo boolean que elimina una venta detalle.

****

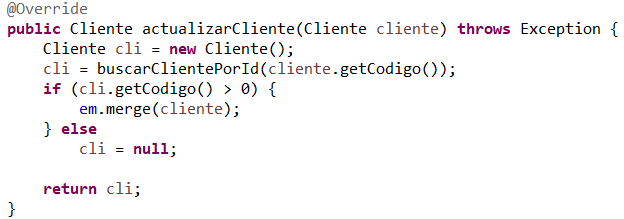
* **buscarPorId:** Búsqueda de la venta detalle por la id

****

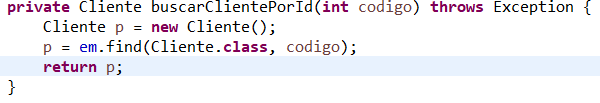
* **buscarVentaDetalle:** Búsqueda de venta detalle

****

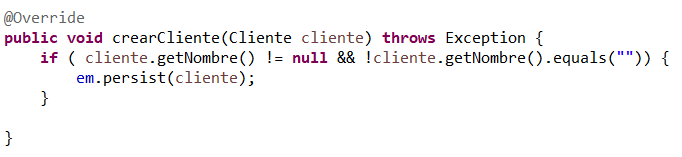
* **CRUD tabla Cliente**
* **actualizarCliente:** Método de tipo Cliente que se utiliza para actualizar la tabla Cliente

****

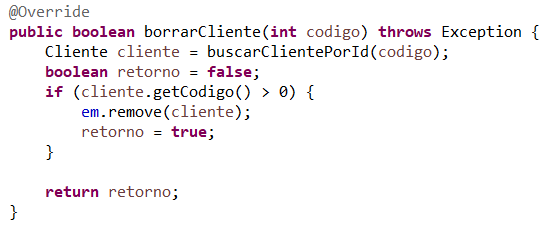
* **buscarClientePorId:** Búsqueda del cliente por la id

****

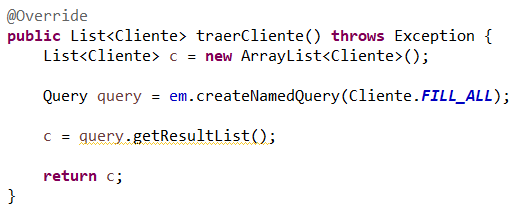
* **crearCliente:** Método de tipo void que crea un cliente

****

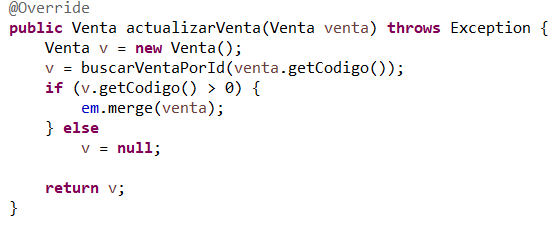
* **borrarCliente:** Método de tipo boolean que elimina un cliente.

****

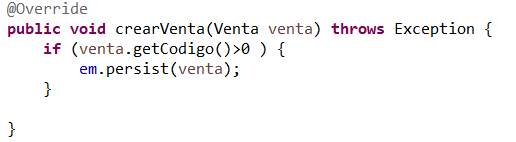
* **traerCliente:** Método de tipo List<Cliente> que realiza una búsqueda por el nombre de la categoría

****

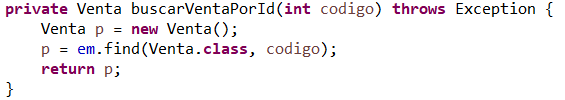
* **CRUD tabla Venta**
* **actualizarVenta:** Método de tipo venta que se utiliza para actualizar la tabla

****

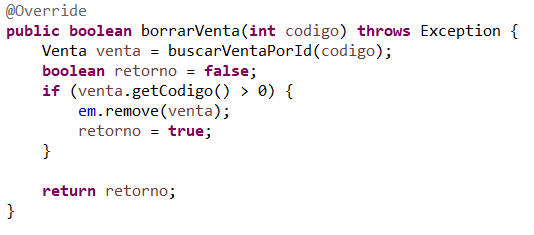
* **crearVenta:** Método de tipo void que crea una venta

****

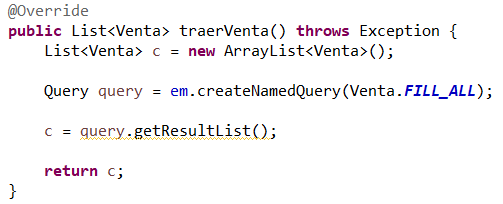
* **buscarVentaPorId:** Búsqueda de la venta por la id

****

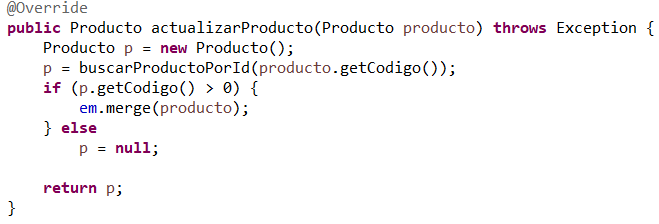
* **borrarVenta:** Método de tipo boolean que elimina una venta

****

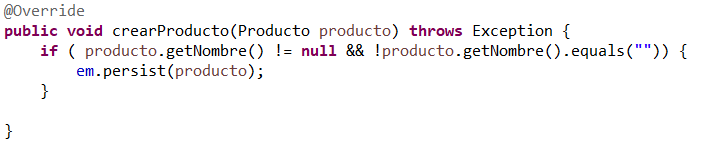
* **traerVenta:** Método de tipo List<venta> que realiza una búsqueda por el nombre de la categoría

****

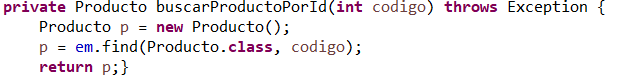
* **CRUD tabla Producto**
* **actualizarProducto:** Método de tipo - que se utiliza para actualizar la tabla

****

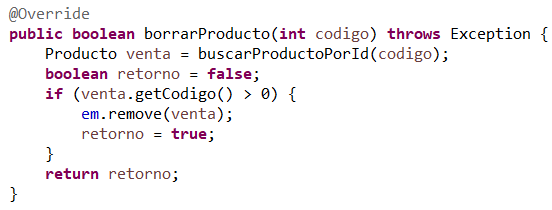
* **crearProducto:** Método de tipo void que crea un producto

****

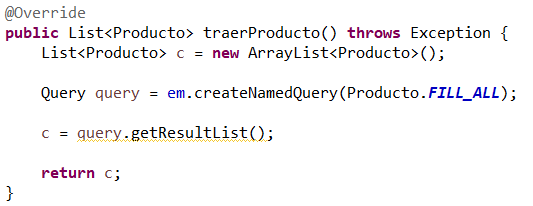
* **buscarProductoPorId:** Búsqueda del producto por la id

****

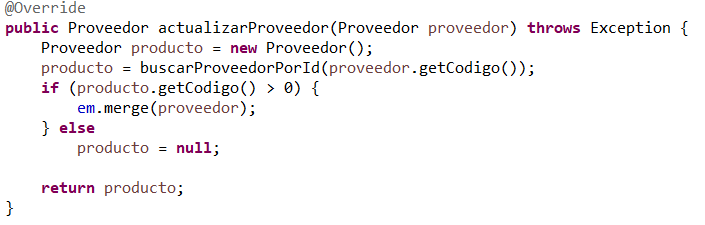
* **borrarProducto:** Método de tipo boolean que elimina un producto

****

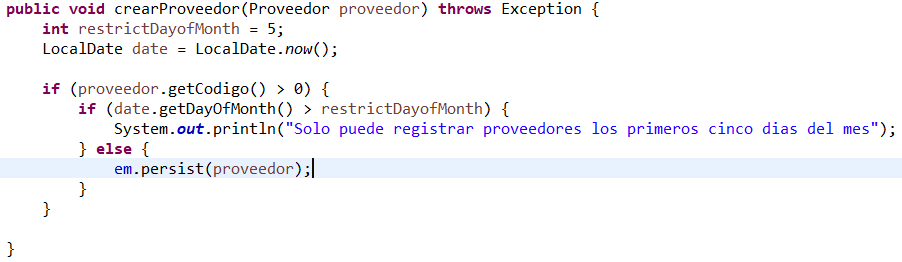
* **traerProducto:** Método de tipo List<producto> que realiza una búsqueda por el nombre del producto

****

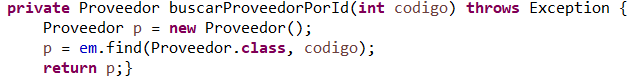
* **CRUD tabla Proveedor**
* **actualizarProveedor:** Método de tipo Proveedor que se utiliza para actualizar la tabla proveedor

****

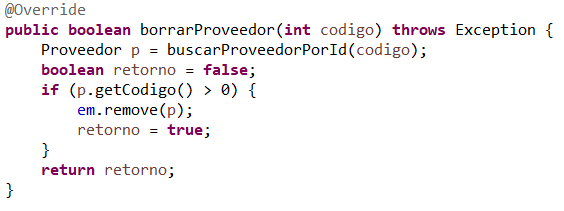
* **crearProveedor:** Método de tipo void que crea un proveedor

****

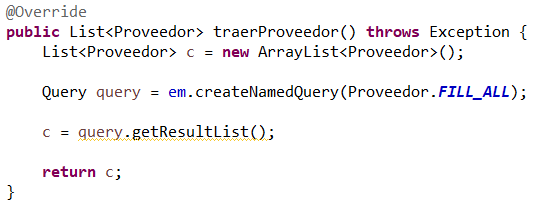
* **buscarProveedorPorId:** Búsqueda del proveedor por la id

****

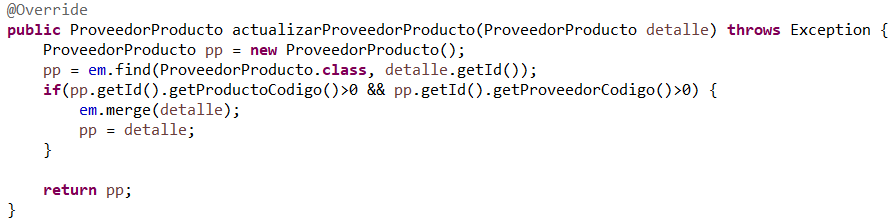
* **borrarProveedor:** Método de tipo boolean que elimina un proveedor

****

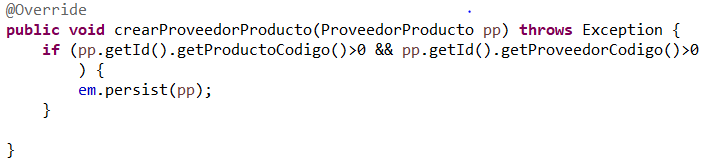
* **traerProveedor:** Método de tipo List<Proveedor> que realiza una búsqueda por el nombre del Proveedor

****

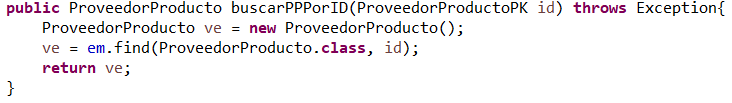
* **CRUD tabla ProveedorProducto**
* **actualizarProveedorProducto:** Método de tipo ProveedorProducto que se utiliza para actualizar la tabla ProveedorProducto

****

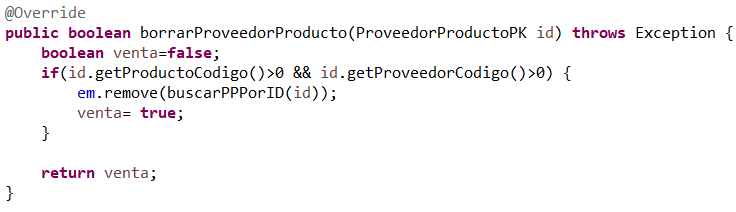
* **crearProveedorProducto:** Método de tipo void que crea un proveedorProducto

****

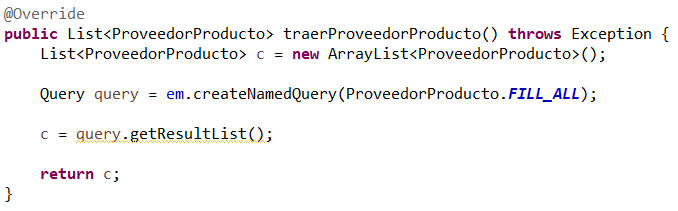
* **buscarPPPorId:** Búsqueda del proveedor producto por la id

****

* **borrarProveedorProducto:** Método de tipo boolean que elimina un proveedor producto

****

* **traerProveedorProducto:** Método de tipo List<ProveedorProducto> que realiza una búsqueda por el nombre del ProveedorProducto

****

**2.0.2. Fachada y servicio**

**//------------------------CRUD DE CATEGORIAPRODUCTO--------------------------------**

**/\*\***

**\* en este metodo se va a listar las categorias correspondientes**

**\* al los productos**

**\* @return List<CategoriaProducto>: listado de productos**

**\* @throws Exception Capturar errores posibles sobre ejecucion**

**\*/**

**public List<CategoriaProducto> traerProductos() throws Exception;**

**/\*\***

**\* metodo que crea la categoria correspodiente a un producto**

**\* @param producto: tipo de objeto a crear**

**\* @throws Exception errores posibles sobre ejecucion**

**\*/**

**public void crearCagtegoriaProducto(CategoriaProducto producto) throws Exception;**

**/\*\***

**\* metodo encargado de actualizar la cagtegoria**

**\* @param categoriaProducto**

**\* @return la Categoria de Producto que se creo**

**\* @throws Exception**

**\*/**

**public CategoriaProducto editarCategoriaProducto(CategoriaProducto categoriaProducto) throws Exception;**

**/\*\***

**\* metodo que borra la categoria del producto seleccionado**

**\* @param codigo**

**\* @return true: donde si borra y false donde hay error de ejecucion**

**\* @throws Exception**

**\*/**

**public boolean borrarCategoriaProducto(int codigo) throws Exception;**

**/\*\***

**\* metodo que busca por el nombre de la categoria del produco**

**\* @param nombre: nombre de la categoria**

**\* @return: la objeto categoria producto**

**\* @throws Exception**

**\*/**

**public List<CategoriaProducto> buscarCategoriaPorNombre(String nombre) throws Exception;**

**//------------------------CRUD DE VENTA-DETALLE - PK--------------------------------**

**/\*\***

**\* metodo que actualiza la venta detalle de una compra**

**\* @return**

**\* @throws Exception**

**\*/**

**public VentaDetalle actualizarVentaDetalle(VentaDetalle detalle) throws Exception;**

**/\*\***

**\* Crea un detalle de venta**

**\* @param detalle Objeto VentaDetalle**

**\* @throws Exception**

**\*/**

**public void crearVentaDetalle(VentaDetalle detalle) throws Exception;**

**/\*\***

**\* Eliminar una venta detalle**

**\* @param \*\*/**

**public boolean borrarVentaDetalle(VentaDetallePK id) throws Exception;**

**/\*\***

**\* metodo que busca por el PK de la venta detalle**

**\* @param VentaDetallePK**

**\* @return: VentaDetalle**

**\* @throws Exception**

**\*/**

**public VentaDetalle buscarPorID(VentaDetallePK id) throws Exception;**

**//------------------------CRUD DE CLIENTE--------------------------------**

**/\*\***

**\* metodo que actualiza la información del cliente**

**\* @return**

**\* @throws Exception**

**\*/**

**public Cliente actualizarCliente(Cliente cliente) throws Exception;**

**/\*\***

**\* Crear un nuevo cliente**

**\* @param detalle Objeto Cliente**

**\* @throws Exception**

**\*/**

**public void crearCliente(Cliente cliente) throws Exception;**

**/\*\***

**\* Eliminar un cliente**

**\* @param \*\*/**

**public boolean borrarCliente(int codigo) throws Exception;**

**/\*\***

**\* metodo que trae todos los clientes**

**\* @return: la objeto cliente**

**\* @throws Exception**

**\***

**\*/**

**public List<Cliente> traerCliente() throws Exception;**

**/\*\***

**\* metodo que trae las ventas detalle**

**\* @return: la lista de objetos venta detalle**

**\* @throws Exception**

**\*/**

**public List<VentaDetalle> buscarVentadetalle(int ventaId, int productoId, int proveedorId,int cantidad,int descuento) throws Exception;**

**//------------------------CRUD DE VENTA--------------------------------**

**/\*\***

**\* metodo que actualiza la información de la venta**

**\* @return**

**\* @throws Exception**

**\*/**

**public Venta actualizarVenta(Venta venta) throws Exception;**

**/\*\***

**\* Crear una nueva venta**

**\* @param detalle Objeto Venta**

**\* @throws Exception**

**\*/**

**public void crearVenta(Venta venta) throws Exception;**

**/\*\***

**\* Eliminar una venta**

**\* @param \*\*/**

**public boolean borrarVenta(int codigo) throws Exception;**

**/\*\***

**\* metodo que trae las ventas**

**\* @return: la objeto venta**

**\* @throws Exception**

**\*/**

**public List<Venta> traerVenta() throws Exception;**

**//------------------------CRUD DE PRODUCTO--------------------------------**

**/\*\***

**\* metodo que actualiza la información del producto**

**\* @return**

**\* @throws Exception**

**\*/**

**public Producto actualizarProducto(Producto producto) throws Exception;**

**/\*\***

**\* Crear un nuevo producto**

**\* @param detalle Objeto Producto**

**\* @throws Exception**

**\*/**

**public void crearProducto(Producto producto) throws Exception;**

**/\*\***

**\* Eliminar un Producto**

**\* @param \*\*/**

**public boolean borrarProducto(int codigo) throws Exception;**

**/\*\***

**\* metodo que trae los Productos**

**\* @return: la lista de objetos Producto**

**\* @throws Exception**

**\*/**

**public List<Producto> traerProducto() throws Exception;**

**//------------------------CRUD DE PROVEEDOR--------------------------------**

**/\*\***

**\* metodo que actualiza la información del proveedor**

**\* @return**

**\* @throws Exception**

**\*/**

**public Proveedor actualizarProveedor(Proveedor proveedor) throws Exception;**

**/\*\***

**\* Crear un nuevo proveedor**

**\* @param detalle Objeto proveedor**

**\* @throws Exception**

**\*/**

**public void crearProveedor(Proveedor proveedor) throws Exception;**

**/\*\***

**\* Eliminar un proveedor**

**\* @param \*\*/**

**public boolean borrarProveedor(int codigo) throws Exception;**

**/\*\***

**\* metodo que trae los proveedores**

**\* @return: la objeto proveedor**

**\* @throws Exception**

**\*/**

**public List<Proveedor> traerProveedor() throws Exception;**

**//------------------------CRUD DE PROVEEDOR-PRODUCTO -PK--------------------------------**

**/\*\***

**\* metodo que actualiza el ProveedorProducto**

**\* @return**

**\* @throws Exception**

**\*/**

**public ProveedorProducto actualizarProveedorProducto(ProveedorProducto detalle) throws Exception;**

**/\*\***

**\* Crea un ProveedorProducto**

**\* @param detalle Objeto ProveedorProducto**

**\* @throws Exception**

**\*/**

**public void crearProveedorProducto(ProveedorProducto detalle) throws Exception;**

**/\*\***

**\* Eliminar un ProveedorProducto**

**\* @param ProveedorProductoPK segun sus llaves primarias**

**\*\*/**

**public boolean borrarProveedorProducto(ProveedorProductoPK id) throws Exception;**

**/\*\***

**\* metodo que trae todos los ProveedorProducto**

**\* @return: List ProveedorProducto**

**\* @throws Exception**

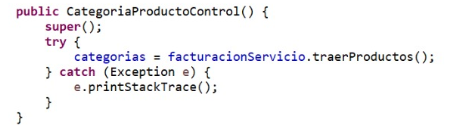
**\*/**

**public List<ProveedorProducto> traerProveedorProducto() throws Exception;**

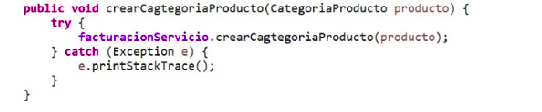
**2. DESCRIPCIÓN FACTURACION WEB**

**2.1 CONTROLADOR CATEGORIA PRODUCTO:**

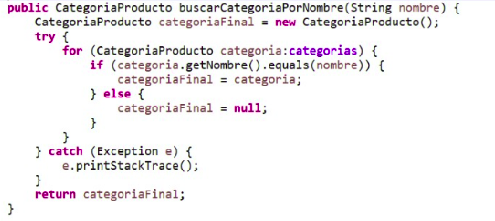
* **Categoría Producto Control:**

****

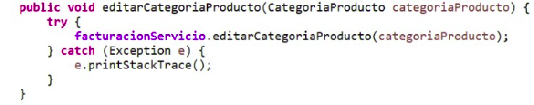
* **Crear Categoria Producto:** Este es un método que se utiliza para crear las categorias de los productos, tiene como parametro a un producto, al cual se le asignará una categoría. No retorna nada ya que se basa en la creación de la categoría, no es necesario que retorne algún dato. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

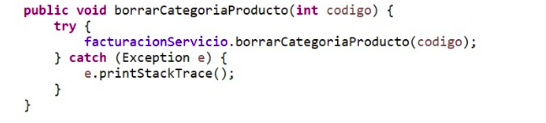
* **Buscar categoría por nombre:** Busca las categorías mediante el uso de la variable “nombre” al ser llamada en el método. Este método buscará el nombre en el listado de categorías, si no encuentra nada retorna nulo, de lo contrario, habrá encontrado la categoría dando como resultado el objeto categoriaFinal. Incluye una excepción con el fin de capturar los errores en la ejecución del método.

****

* **Editar Categoría Producto:** Este método nos permite acceder a las categorías ya existentes y modificarlas. Se toma como parámetro a la categoriaProducto, el cual traerá en si el objeto de la categoría correspondiente. Incluye una excepción con el fin de capturar los errores en la ejecución del método.

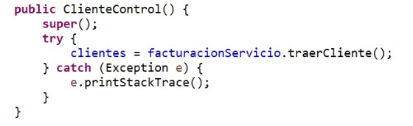
****

* **Borrar Categoria Producto:** este método borra una categoría de un producto. Lo hace por medio del código de la categoría que se implementa como un parámetro. Incluye una excepción con el fin de capturar los errores en la ejecución del método.

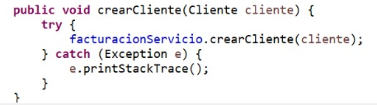
****

**2.2. CONTROLADOR CLIENTE**

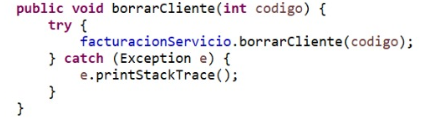
* **Cliente Control:**Este es un método que se utiliza para traer a el o los Clientes ya existente en nuestra base de datos, retorna una lista. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

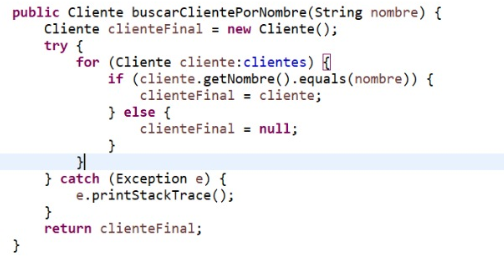
* **Crear Cliente:** este método consigue crear un cliente. Es un método que no retorna nada ya que el método se enfoca en la creación del cliente, no es necesario conocer lo que retorna. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

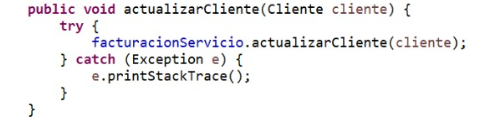
* **Borrar Cliente:** Este es un método que se utiliza para borrar el o los Clientes existentes en nuestra base de datos. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

* **Buscar Cliente Por Nombre:** Busca las categorías mediante el uso de la variable “nombre” al ser llamada en el método. Este método buscará el nombre del cliente en el listado de Clientes, si no encuentra nada retorna nulo, de lo contrario, habrá encontrado el respectivo nombre. Incluye una excepción con el fin de capturar los errores en la ejecución del método.

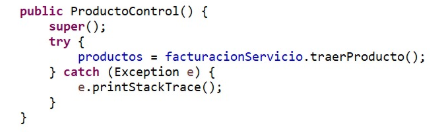
****

* **Actualizar Cliente:** este método busca actualizar un cliente, es decir, sobreescribir y/o editar información sobre el cliente. Debido a esto el método no retorna nada, únicamente cumple con editar los datos. Este método incluye una excepción que captura los posibles errores que puedan surgir durante la ejecución de la instrucción.

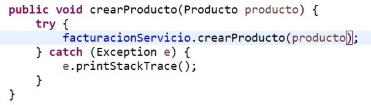
****

**2.3. CONTROLADOR PRODUCTO**

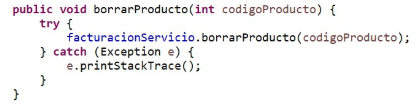
* **Producto Control:** Este es un método que se utiliza para traer a el o los Productos ya existente en nuestra base de datos, retorna una lista. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

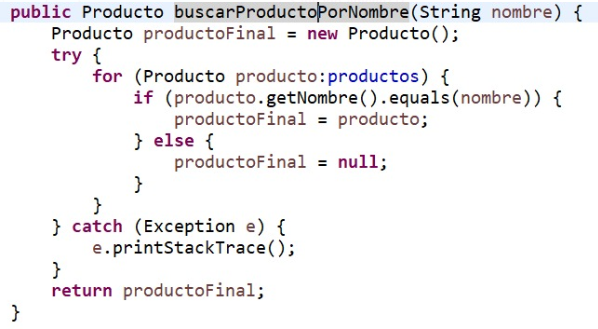
* **Crear Producto:**Este método consigue crear un Producto. Es un método que no retorna nada ya que el método se enfoca en la creación del Producto, no es necesario conocer lo que retorna. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

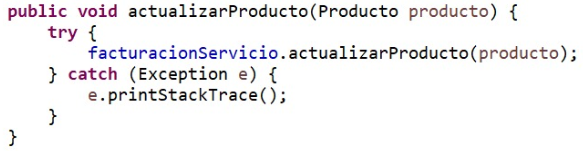
* **Borrar Producto:** Este es un método que se utiliza para borrar el o los Productos existentes en nuestra base de datos. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

* **Buscar Producto Por Nombre:** Busca las los productos mediante el uso de la variable “nombre” al ser llamada en el método. Este método buscará el nombre del producto en el listado de Productos, si no encuentra nada retorna nulo, de lo contrario, habrá encontrado el respectivo nombre. Incluye una excepción con el fin de capturar los errores en la ejecución del método.

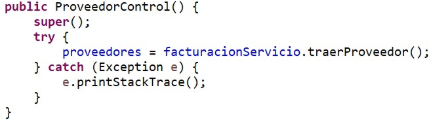
****

* **Actualizar Producto:** Este método busca actualizar un Producto, es decir, sobreescribir y/o editar información sobre el producto . Debido a esto el método no retorna nada, únicamente cumple con editar los datos. Este método incluye una excepción que captura los posibles errores que puedan surgir durante la ejecución de la instrucción.

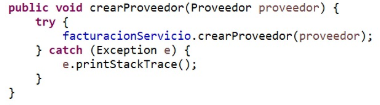
****

**2.4. CONTROLADOR PROVEEDOR**

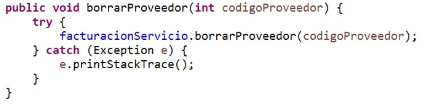
* **Proveedor Control:** Este es un método que se utiliza para traer a el o los Proveedores ya existente en nuestra base de datos, retorna una lista. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

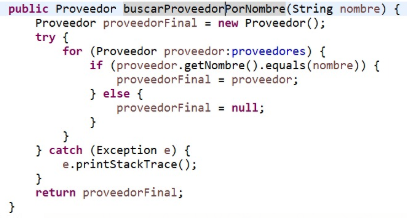
* **Crear Proveedor:** Este método consigue crear un Proveedor. Es un método que no retorna nada ya que el método se enfoca en la creación del proveedor, no es necesario conocer lo que retorna. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

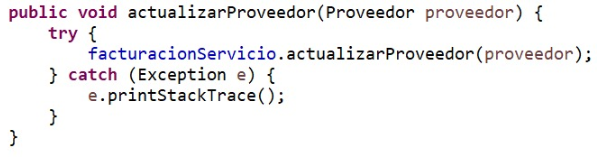
* **Borrar Proveedor:** Este es un método que se utiliza para borrar el o los Proveedores existentes en nuestra base de datos. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

****

* **Buscar Proveedor Por Nombre:** Busca los proveedores mediante el uso de la variable “nombre” al ser llamada en el método. Este método buscará el nombre del proveedor en el listado de Proveedores, si no encuentra nada retorna nulo, de lo contrario, habrá encontrado el respectivo nombre. Incluye una excepción con el fin de capturar los errores en la ejecución del método.

****

* **Actualizar Proveedor:** Este método busca actualizar un Proveedor, es decir, sobreescribir y/o editar información sobre el proveedor. Debido a esto el método no retorna nada, únicamente cumple con editar los datos. Este método incluye una excepción que captura los posibles errores que puedan surgir durante la ejecución de la instrucción.

****

**2.5 CONTROLADOR PROVEEDOR PRODUCTO**

* **Proveedor Producto Control:**
* **Crear Proveedor Producto:**
* **Borrar Proveedor Producto:**
* **Buscar Proveedor Producto:**
* **Actualizar Proveedor Producto:**

**2.6 CONTROLADOR VENTA**

* **Venta Control:**
* **Crear Venta:** Este método consigue crear una venta. Es un método que no retorna nada ya que el método se enfoca en la creación de la venta, no es necesario conocer lo que retorna. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.
* **Borrar Venta:**
* **Buscar Venta:**
* **Actualizar Venta:**

**2.7 CONTROLADOR VENTA DETALLE**

* **Venta Detalle Control:** Este es un método que se utiliza para traer la o las ventasDetalle ya existente en nuestra base de datos, retorna una lista. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.

* **Crear Venta Detalle Producto:** Este método consigue crear una ventaDetalle de cada producto. Es un método que no retorna nada ya que el método se enfoca en la creación de la ventaDetalle, no es necesario conocer lo que retorna. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.
* **Borrar Venta Detalle:** Este es un método que se utiliza para borrar la o las ventaDetalle existentes en nuestra base de datos. Este método incluye una excepción que nos permite capturar posibles errores durante la ejecución de instrucciones.
* **Buscar Venta Detalle por Nombre:** Busca las ventaDetalle mediante el uso de la variable “nombre” al ser llamada en el método. Este método buscará el nombre de ventaDetalle en el listado, si no encuentra nada retorna nulo, de lo contrario, habrá encontrado el respectivo nombre. Incluye una excepción con el fin de capturar los errores en la ejecución del método.
* **Actualizar Venta Detalle:** Este método busca actualizar ventaDetalle, es decir, sobreescribir y/o editar información sobre la ventaDetalle. Debido a esto el método no retorna nada, únicamente cumple con editar los datos. Este método incluye una excepción que captura los posibles errores que puedan surgir durante la ejecución de la instrucción.