Desarrollador Carlos Pac 201931012

Guía lógica y con fines de mantenimiento del programa de administración de ventas y productos de INTELAF

MANUAL TECNICO

PROGRAMA DE ADMINISTARCION DE PRODUCTOS Y VENTAS DE INTELAF

Contenido

[INTRODUCCION 3](#_Toc49106674)

[PLATAFORMA DE DESARROLLO 4](#_Toc49106675)

[REQUISITOS DE SISTEMA 4](#_Toc49106676)

[ALCANCES TENICOS 4](#_Toc49106677)

[ENTIDADES REPRESENTADAS EN EL PROGRAMA 5](#_Toc49106678)

[Entidades 5](#_Toc49106679)

[1. Clase Users: 5](#_Toc49106680)

[2. Clase Client: 7](#_Toc49106681)

[3. Clase Empleoyee: 8](#_Toc49106682)

[4. Clase Pedido: 8](#_Toc49106683)

[5. Calse Product: 11](#_Toc49106684)

[6. Clase Store: 13](#_Toc49106685)

[7. Clase TimeStoreToStore: 16](#_Toc49106686)

[8. Clase Ventas: 17](#_Toc49106687)

[Base de Datos 18](#_Toc49106688)

[1. ConexionDB 19](#_Toc49106689)

[2. ConsultasDB 19](#_Toc49106690)

[3. ModificacionesDB 22](#_Toc49106691)

[4. RegistroDB 23](#_Toc49106692)

[Generación De Archivos 25](#_Toc49106693)

[Lectura de Archivo de Carga 27](#_Toc49106694)

[Interfaz Grafica 28](#_Toc49106695)

[DIAGRAMAS DE LA ESTRUCTURACION DEL SOFTWARE 29](#_Toc49106696)

[Diagrama de clases UML: 29](#_Toc49106697)

[Diagrama Entidad Relación: 29](#_Toc49106698)

[ESTRUCTURACION DE LA BASE DE DATOS 29](#_Toc49106699)

# INTRODUCCION

En el presente documento encontrara un guía a nivel técnico del funcionamiento de administración de ventas y productos de la empresa INTELAF adjuntando la lógica de programación con fines de mantenimiento a futuro o actualización en expansión del programa.

Se utilizará simplificaciones de términos con fines de entendimiento y no sobrecargar la explicación de la lógica.

## PLATAFORMA DE DESARROLLO

* Lenguaje Java en su versión 11 LTS ORACLE.
* Lenguaje Java en su distribución OPEN-JDK
* Sistema de gestión de proyecto Apache Maven 3.6.3.
* Máquina Virtual (build 11.0.7+8-LTS) Windows.
* Máquina Virtual () Linux Ubuntu y derivados.

## REQUISITOS DE SISTEMA

* Windows 10 / Linux Ubuntu 20.04 y derivados.
* 4 Gb en RAM.
* Conexión de área local si el servidor es manejado por otro equipo.
* Procesador a 2.5 GHz
* Mysql (8.0.21) Windows/Linux

## ALCANCES TENICOS

El alcance planteado al programa es la solución al momento de manejar pedidos ventas y existencias de los productos vendidos por la empresa, así como la generación de reportes de los mismos para tener un control del maejo del producto que ofrece la empresa.

## ENTIDADES REPRESENTADAS EN EL PROGRAMA

El programa basado en POO por lo cual está dividido en los siguientes espacios orientados para el trabajo en conjunto alimentando lo que sea necesario durante el programa:

* Entidades
* Base de Datos
* Generación De Archivos
* Lectura de Archivo de Carga
* Interfaz Grafica

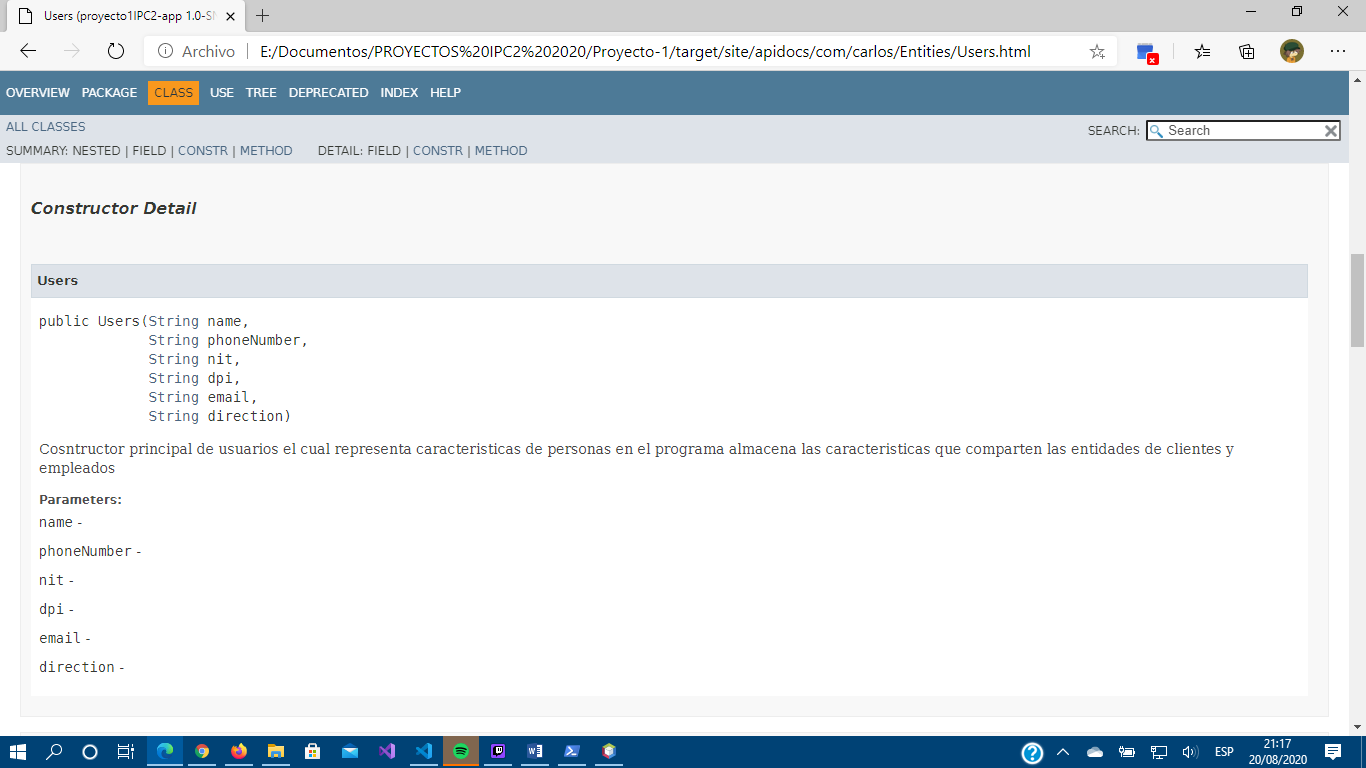
## Entidades

El grupo de entidades contiene los siguientes objetos denominados de la siguiente manera:

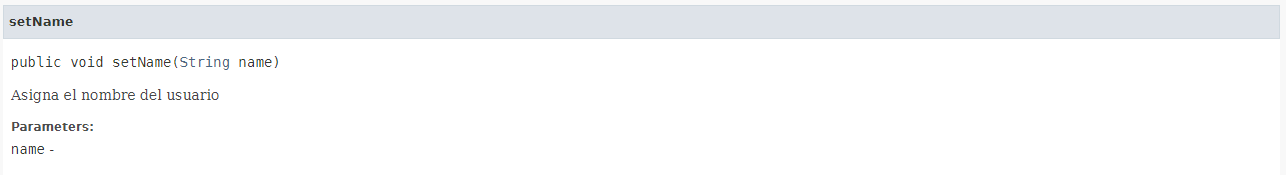
|  |  |
| --- | --- |
| Nombre de Objeto | Descripción de la Objeto |
| Users | Clase padre de los objetos Client y Empleoyee, donde se heredan algunas de sus características |
| Client | Clase hija de Users en donde se abstrae la información de los clientes de la tienda |
| Empleoyee | Clase hija de Users en donde se abstrae información que comparte con cualquier usuario pero ciertas características únicas de un empleado |
| Pedido | Clase en la cual se abstrae la información necesaria de un pedido, para más adelante ser utilizada |
| Product | Clase en la cual se abstrae la información necesaria de un producto para más adelante se utilizada |
| Store | Clase en la cual se abstrae la información necesaria de un tienda para más adelante ser utilizada |
| TimeStoreToStore | Clase la cual se abstrae la información de la movilización de un producto de una tienda a otra |
| Ventas | Clase en la cual se abstrae la informa necesaria de una ventas que más adelante se registra en el programa |

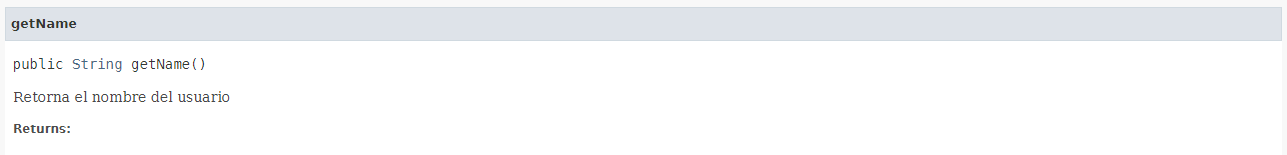
A continuación, se presenta los métodos que contiene cada objeto perteneciente a Entidades:

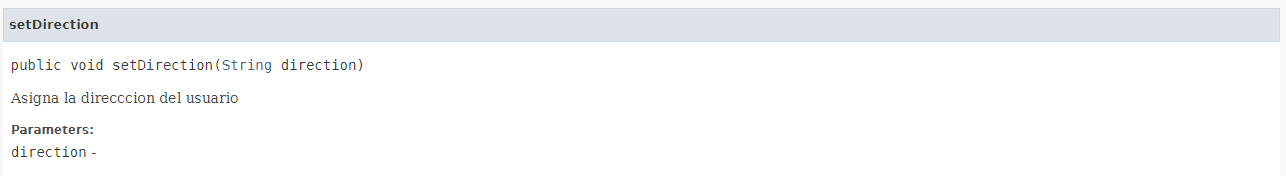
### Clase Users:

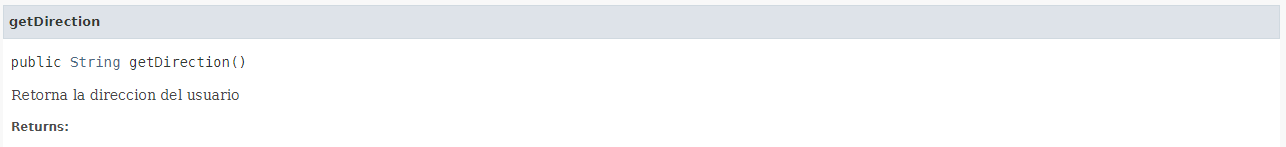


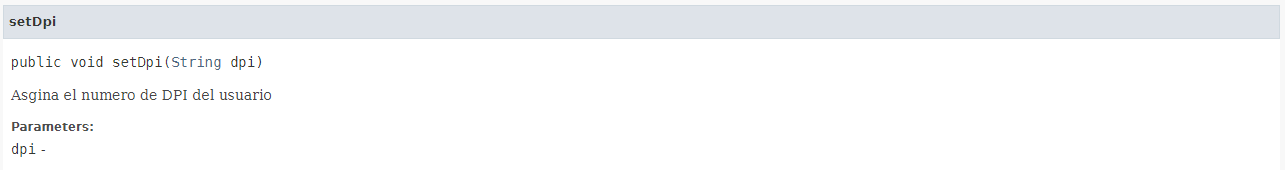


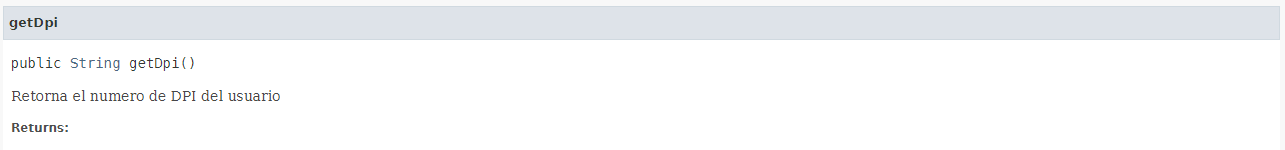


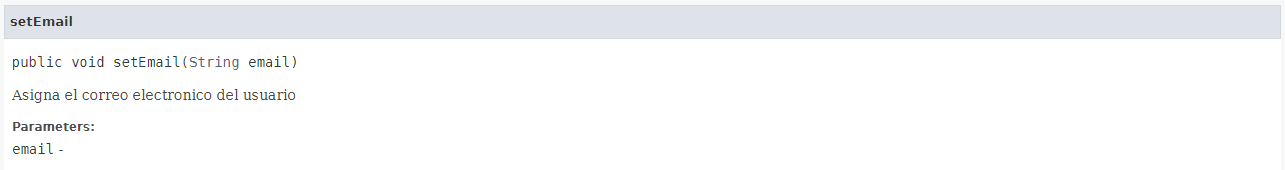


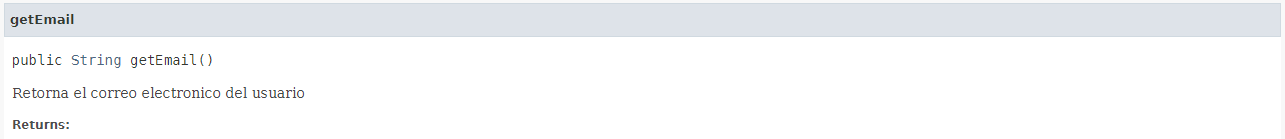


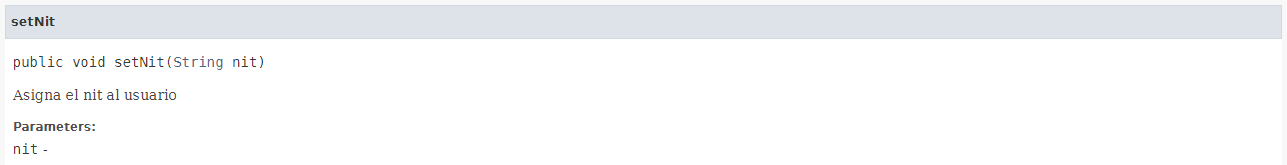


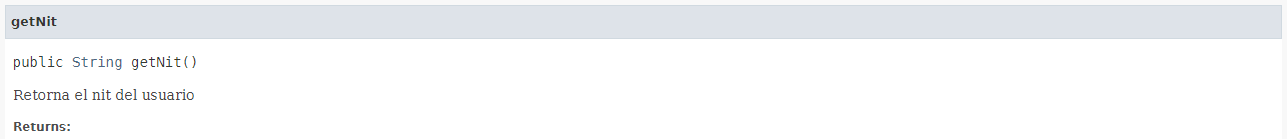


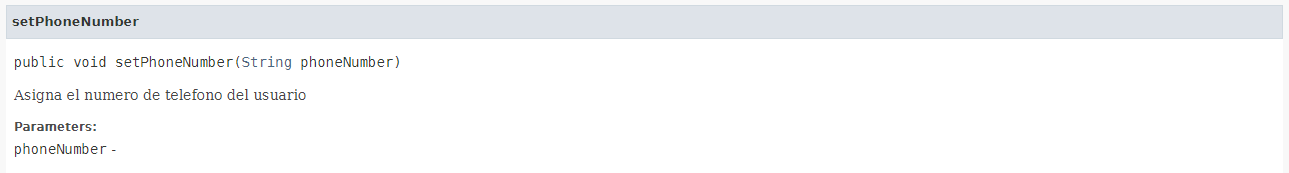


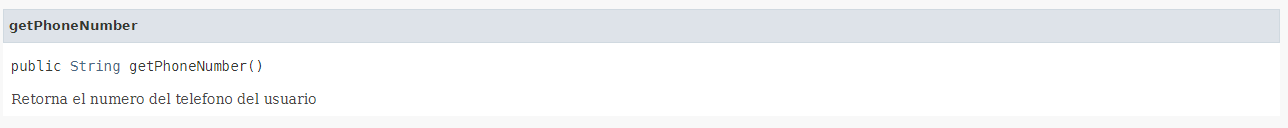




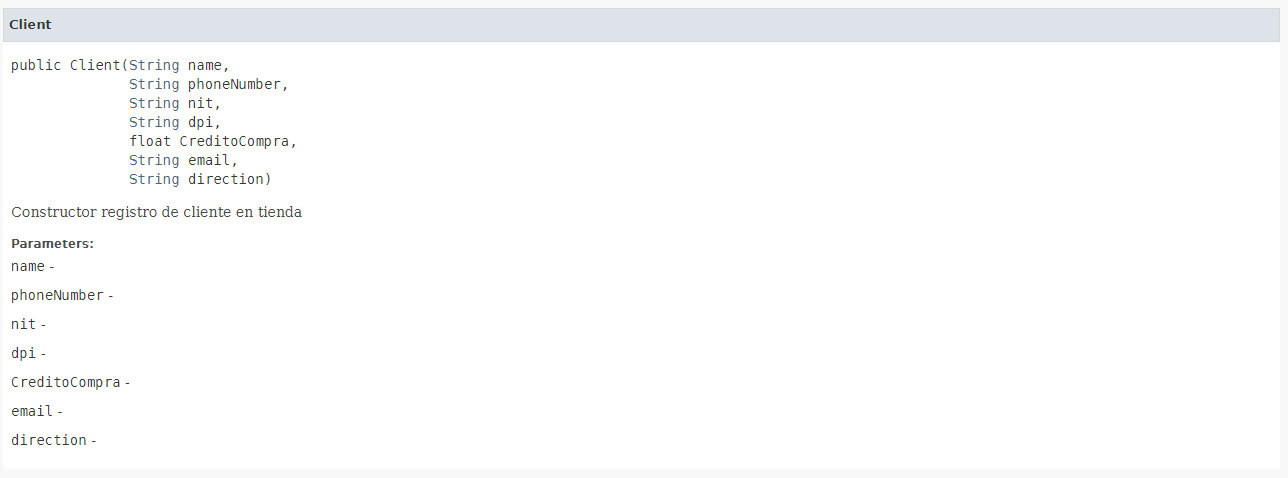


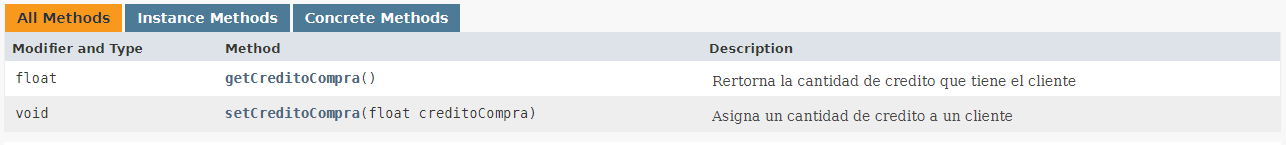


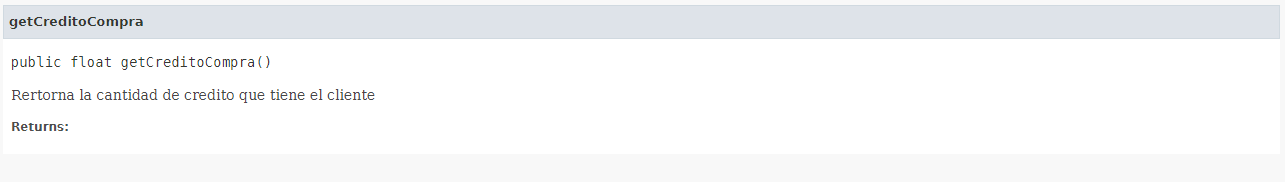


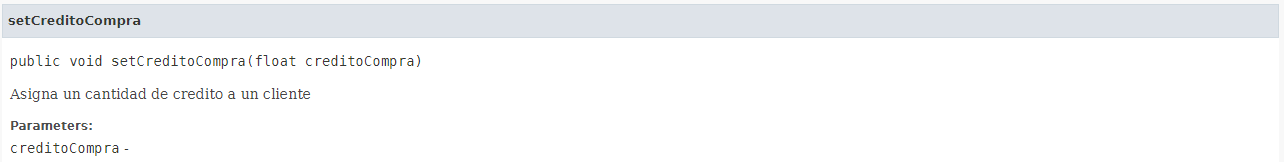


### Clase Client:

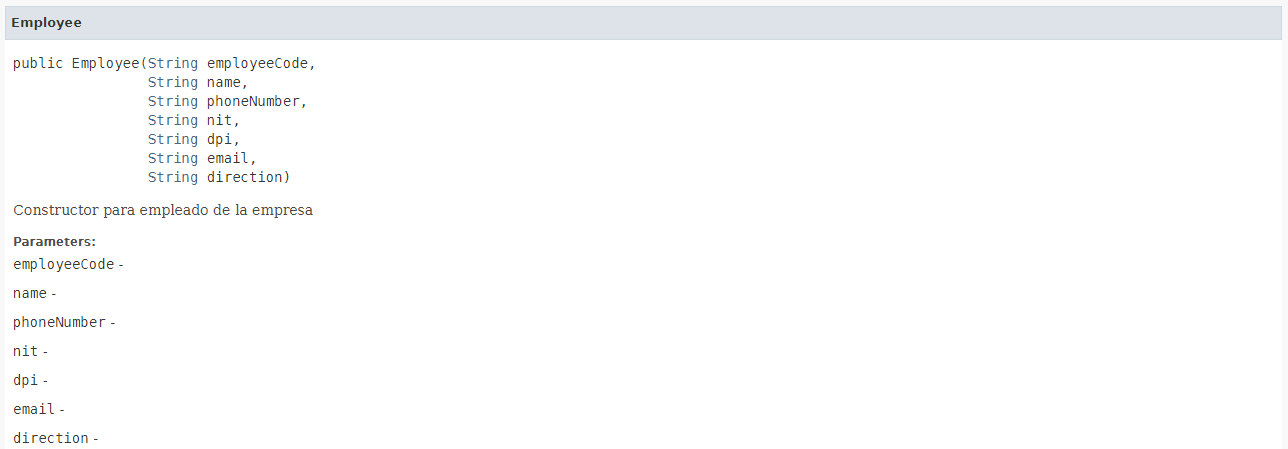


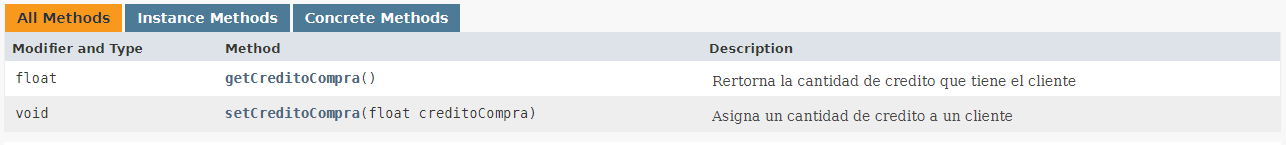


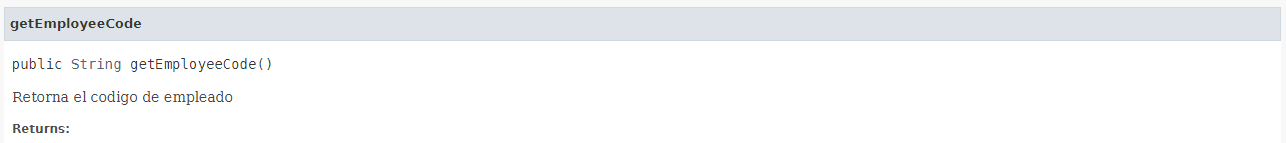


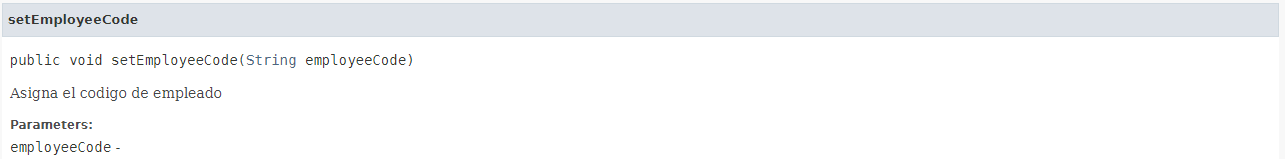


### Clase Empleoyee:

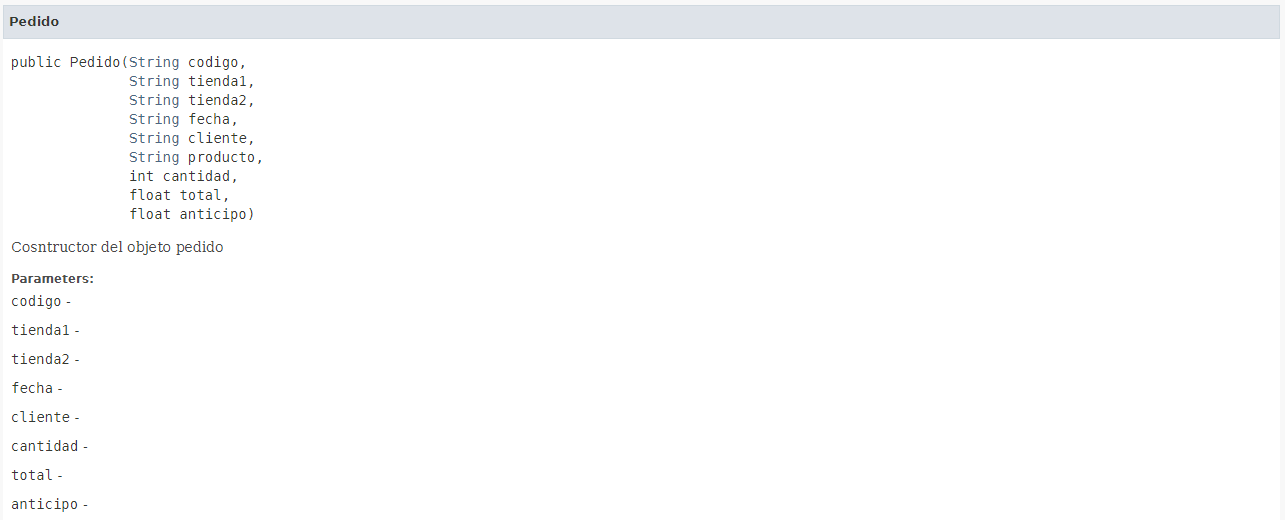




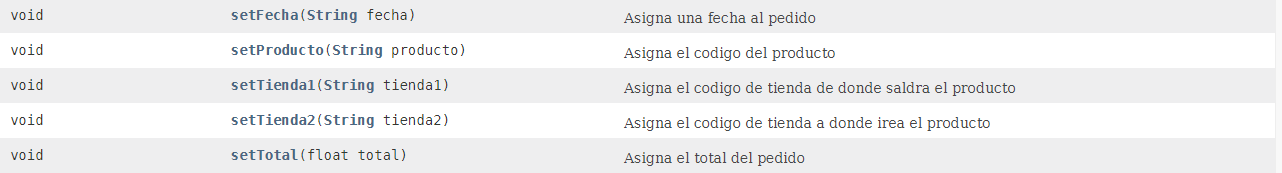


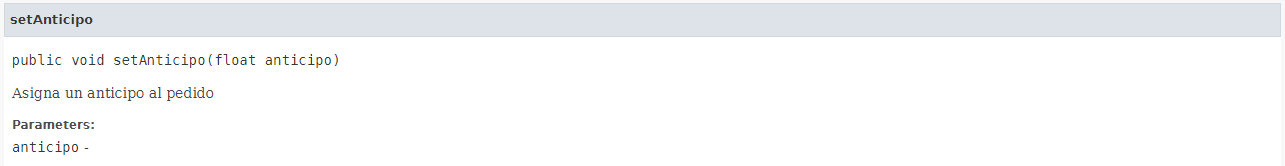


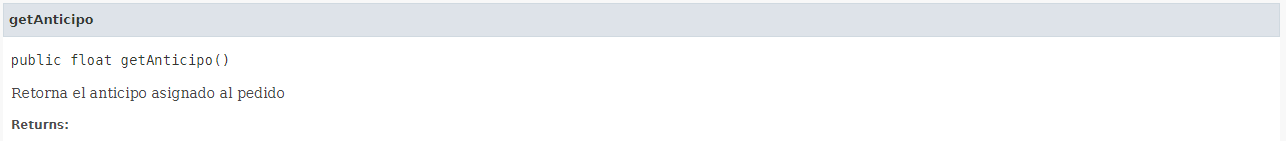
### Clase Pedido:

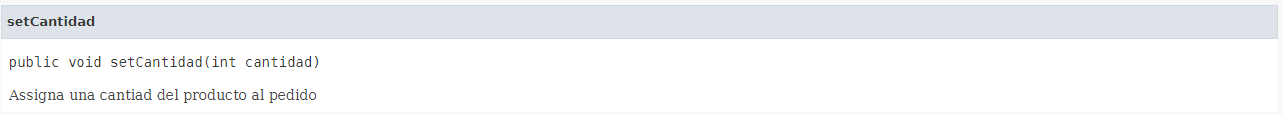


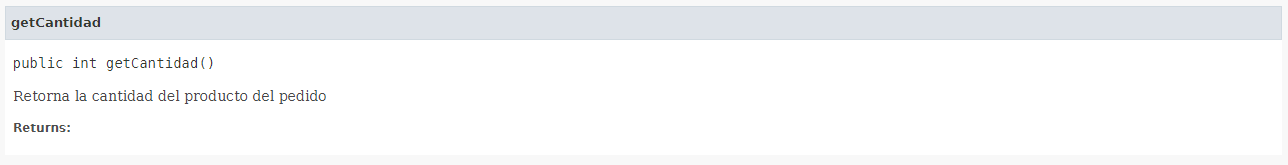


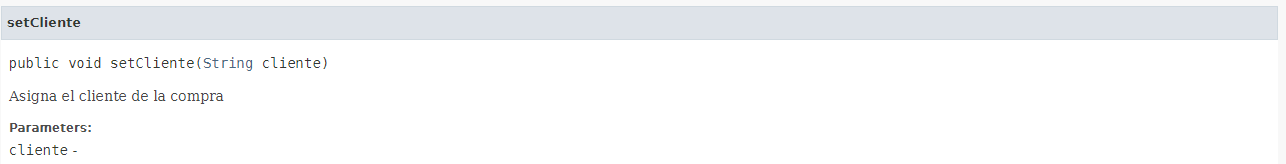


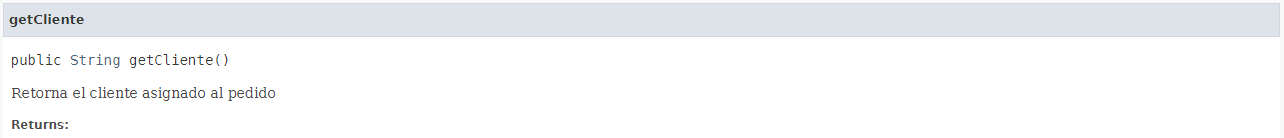


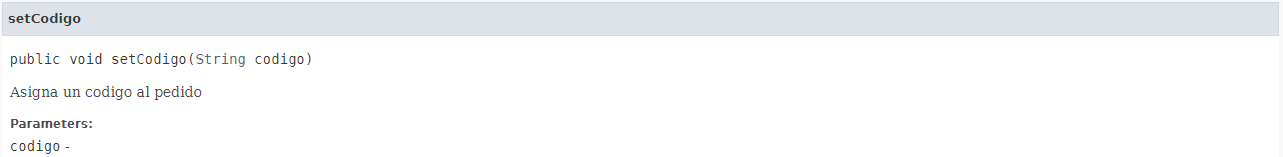


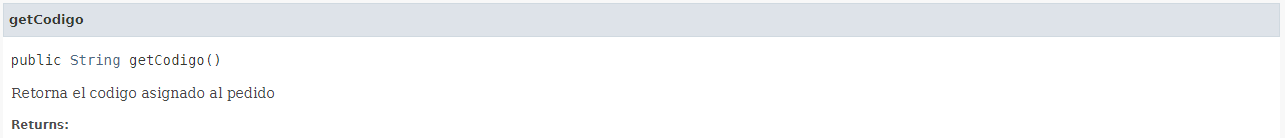


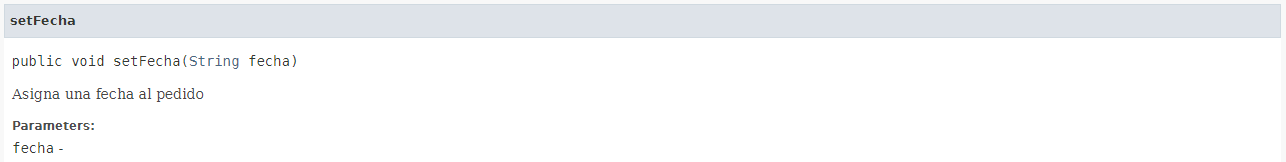


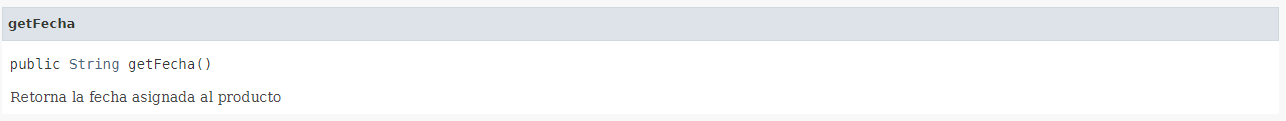


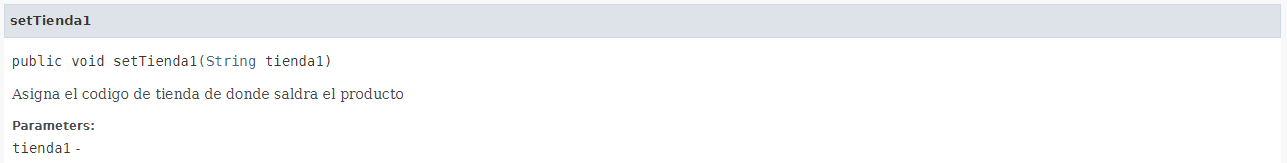


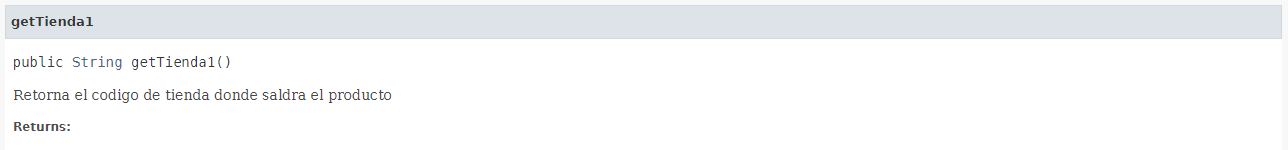


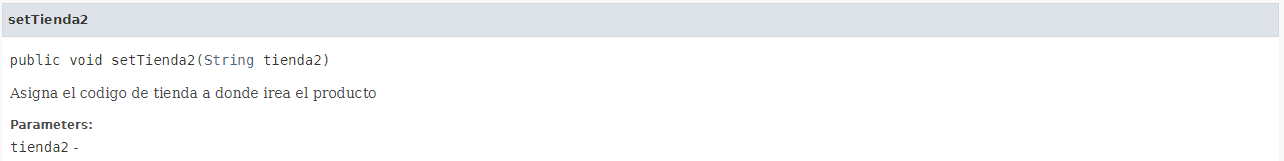


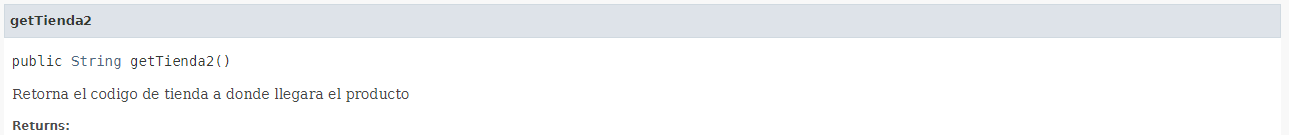


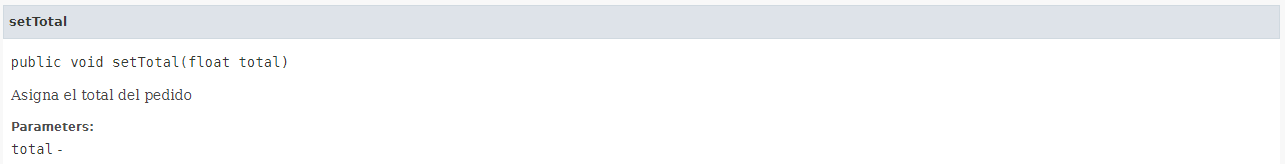


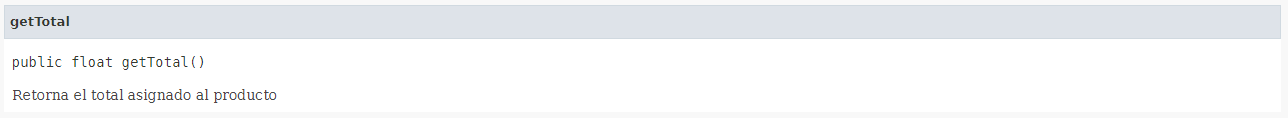


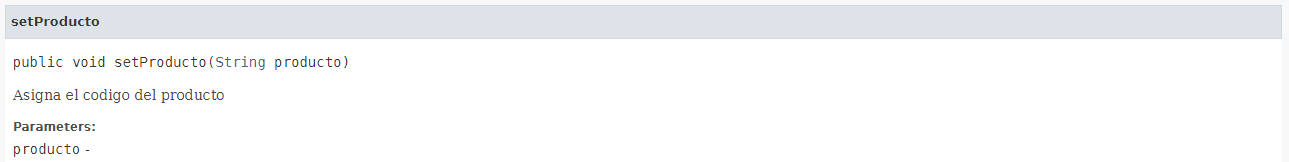


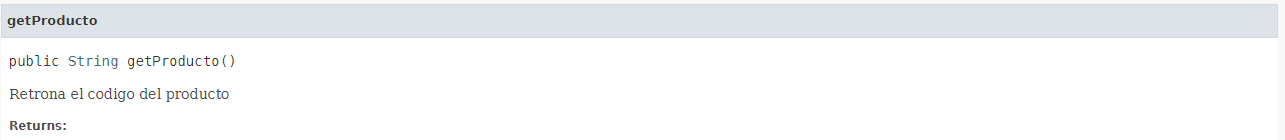




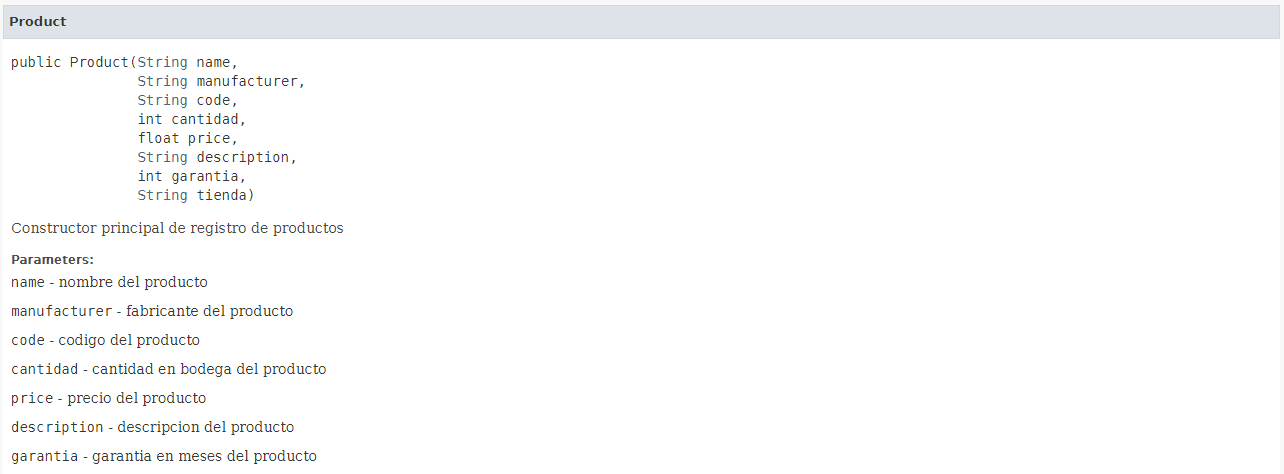






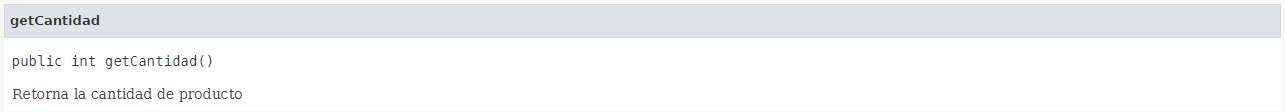


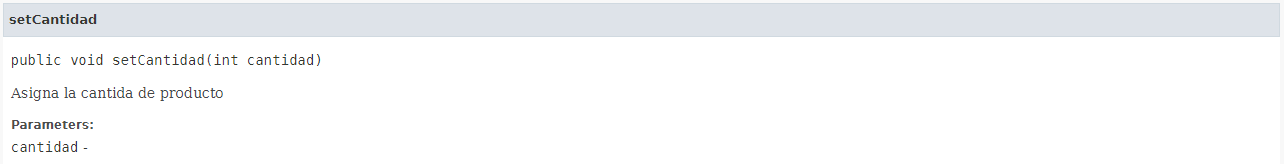
### Calse Product:

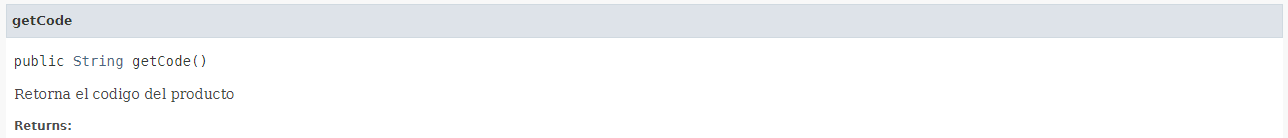


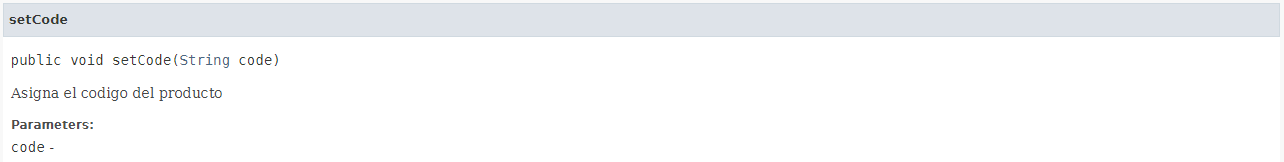


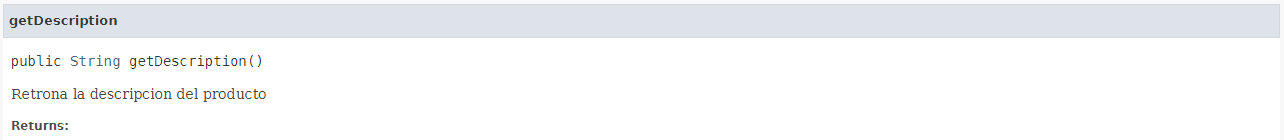


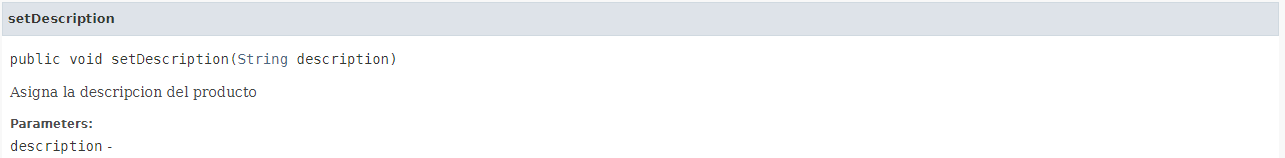


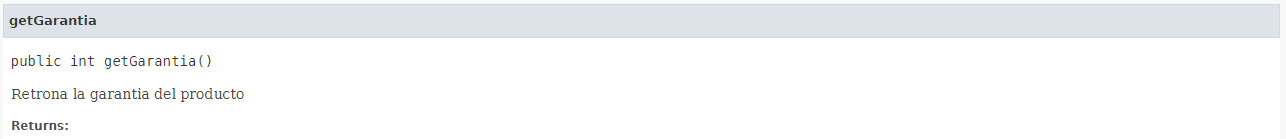


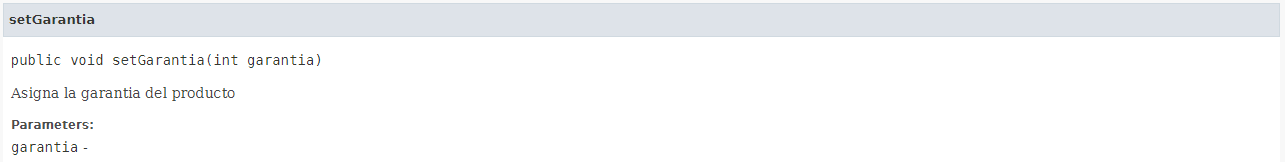


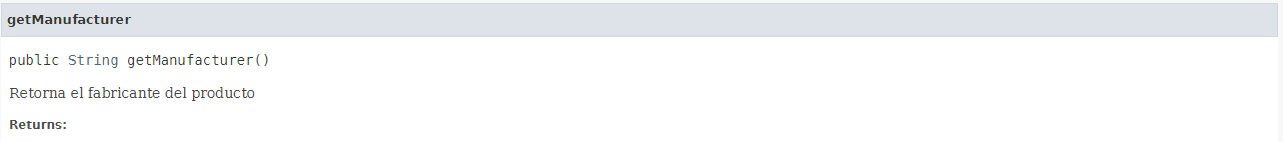


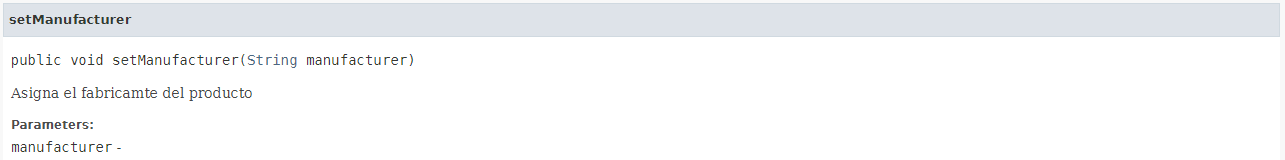


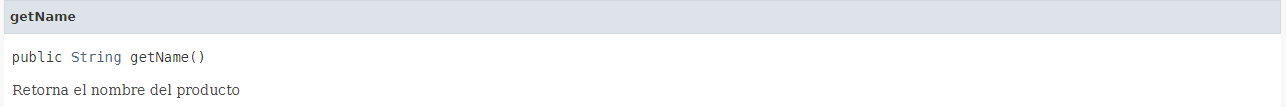


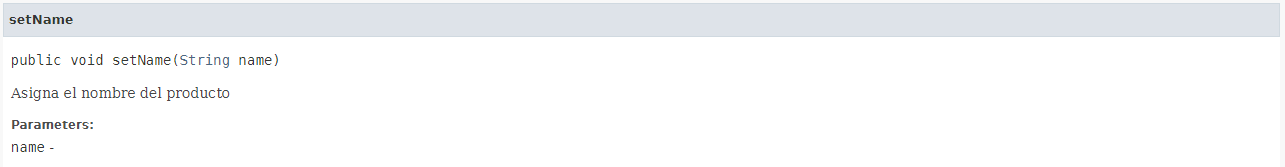


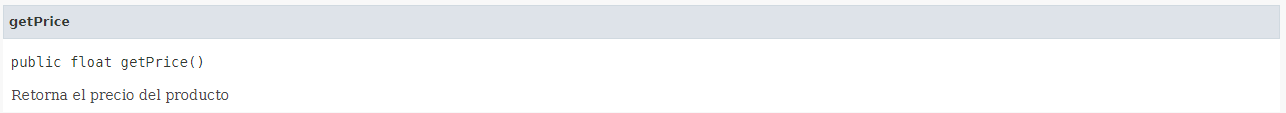


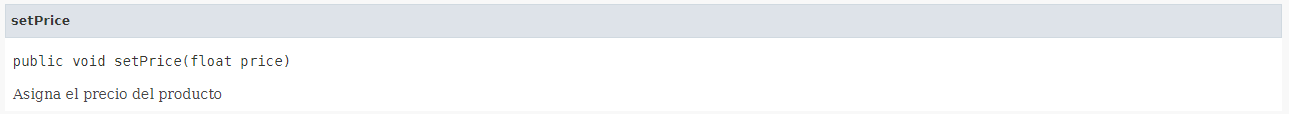


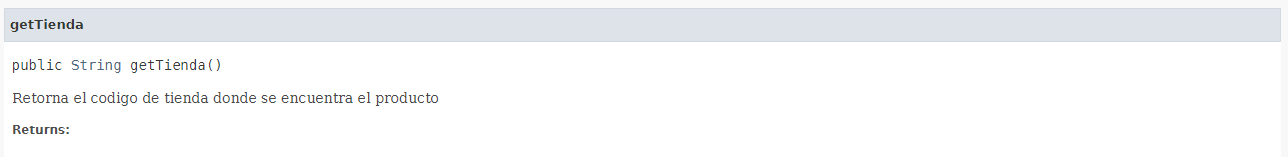


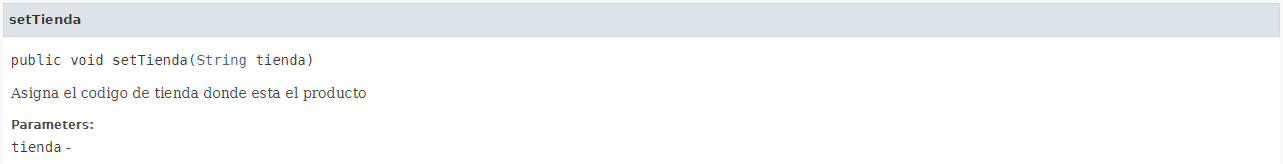




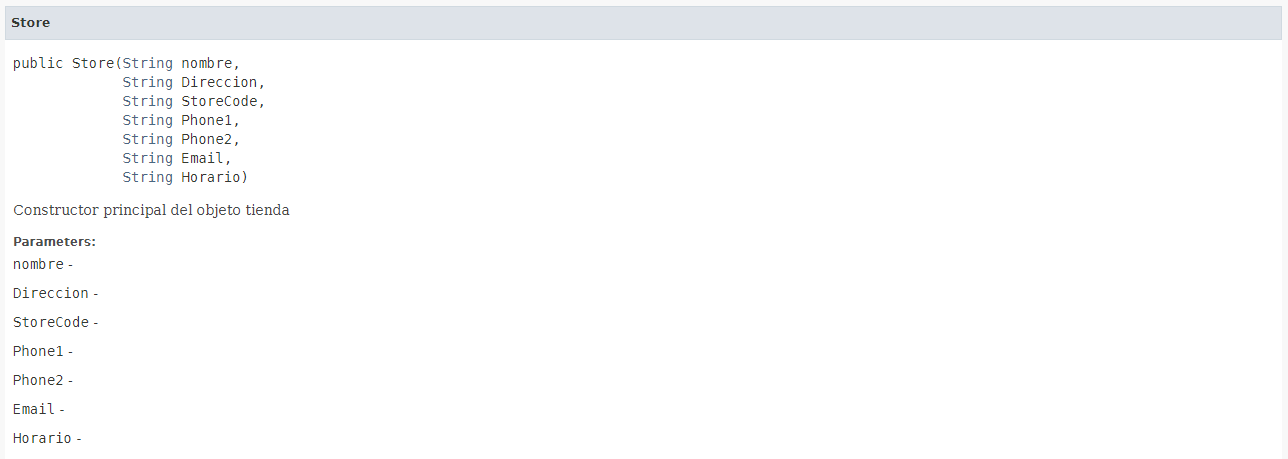




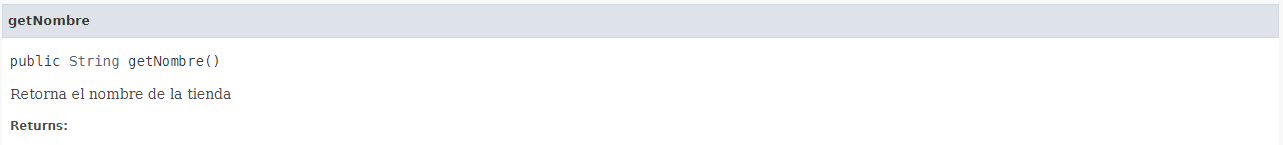


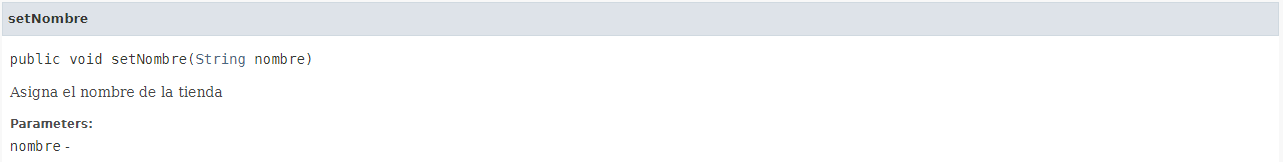


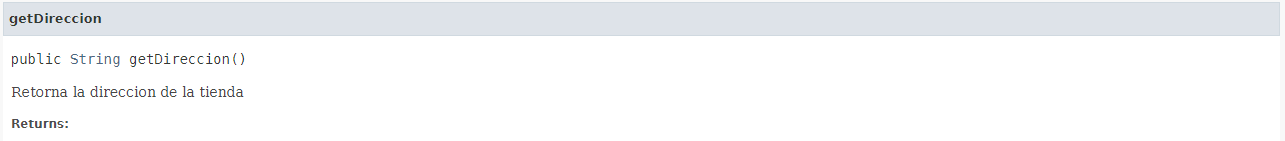
### Clase Store:

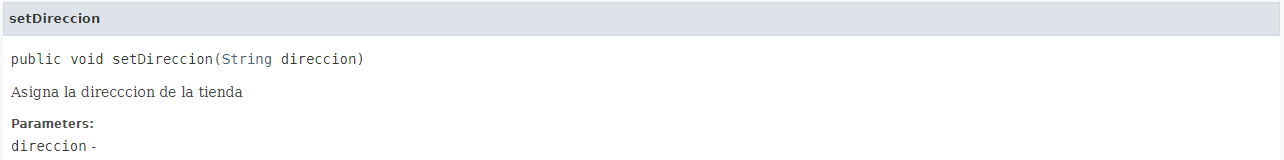


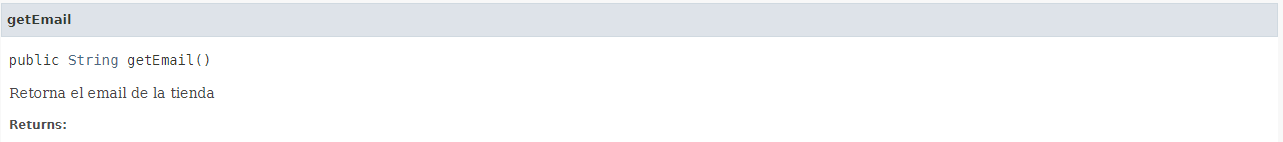


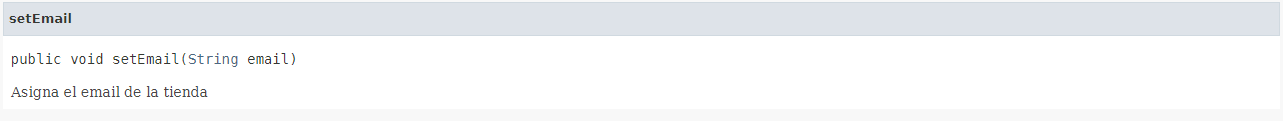


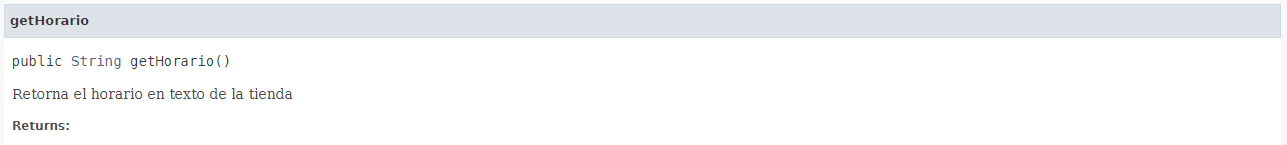


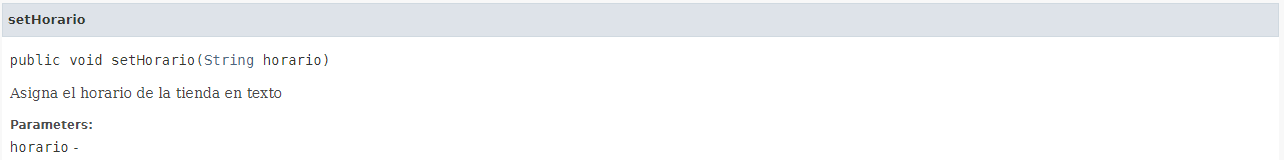


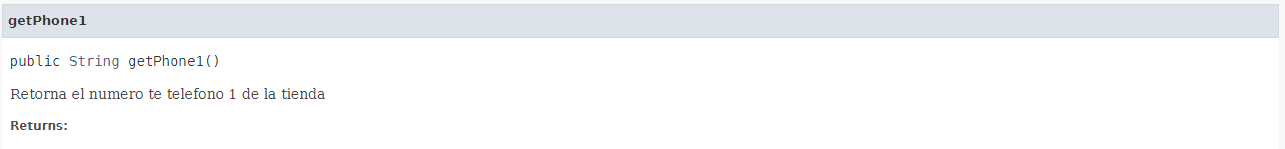


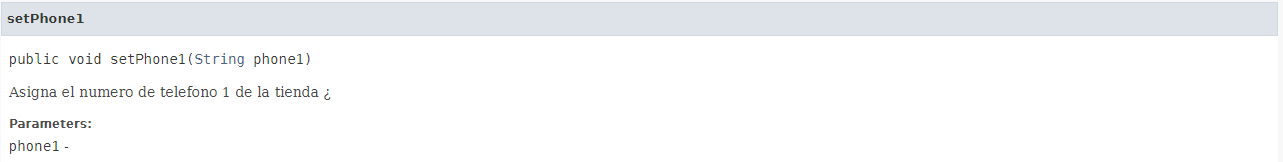


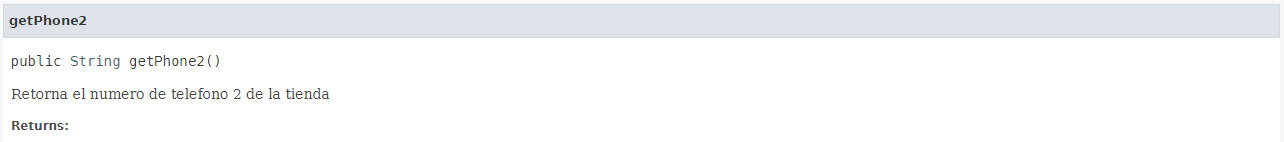


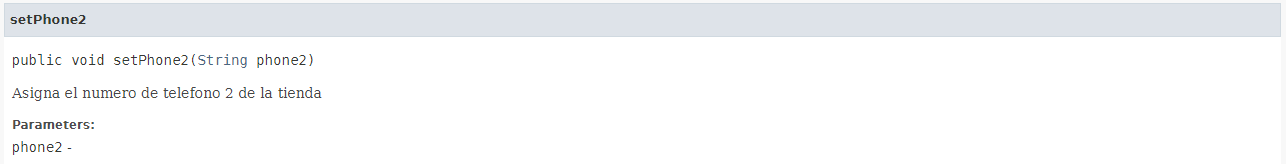


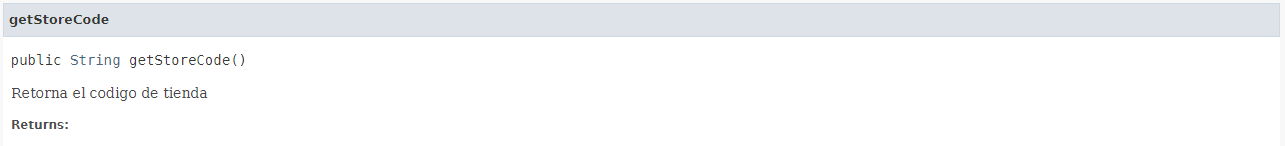


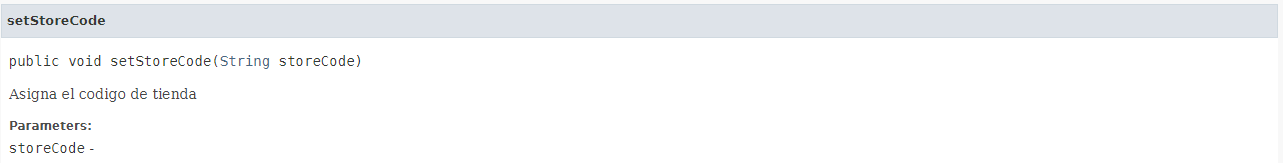




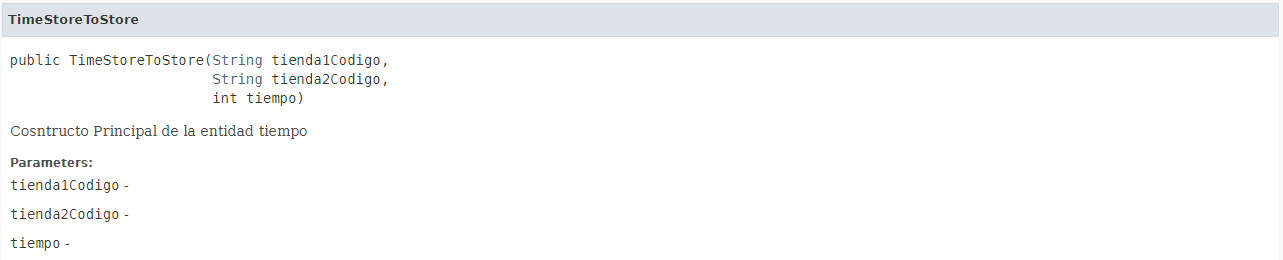




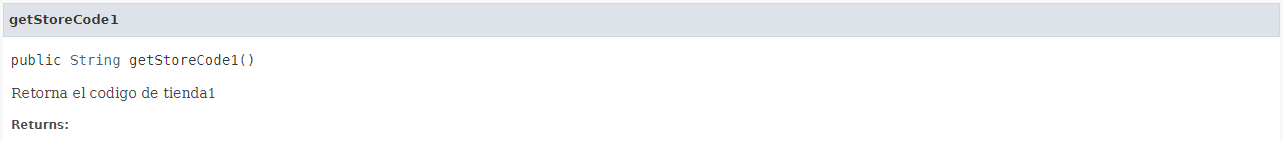


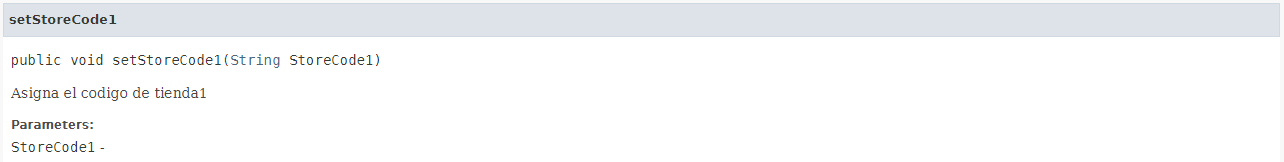


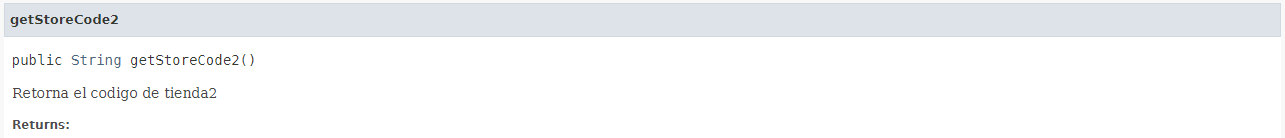
### Clase TimeStoreToStore:

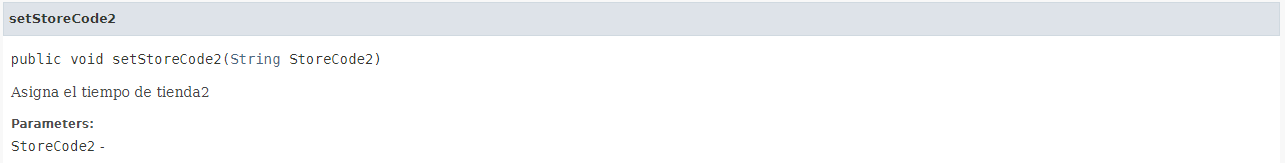


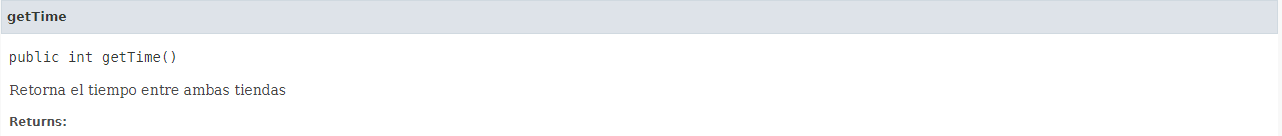


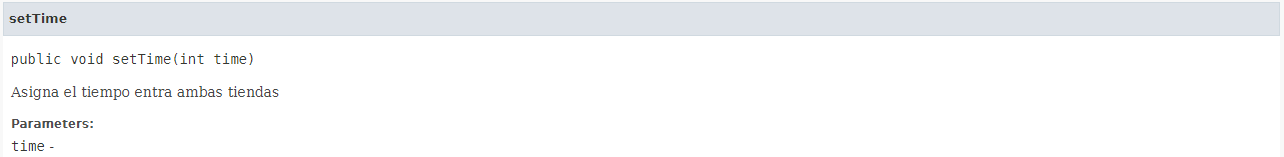


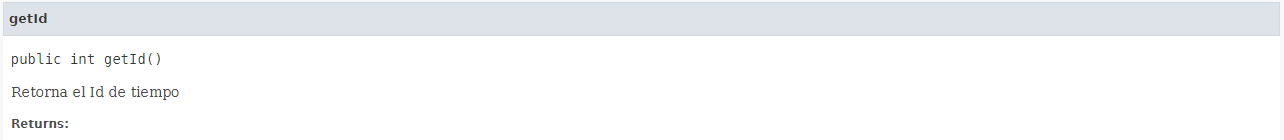


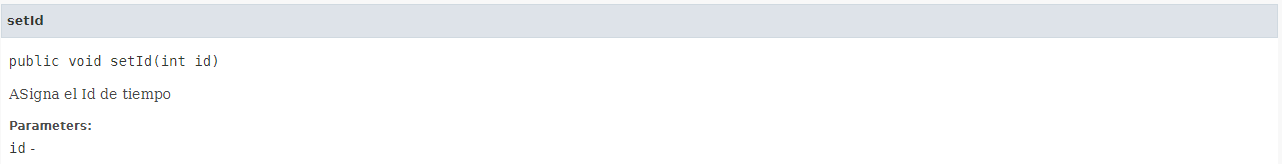




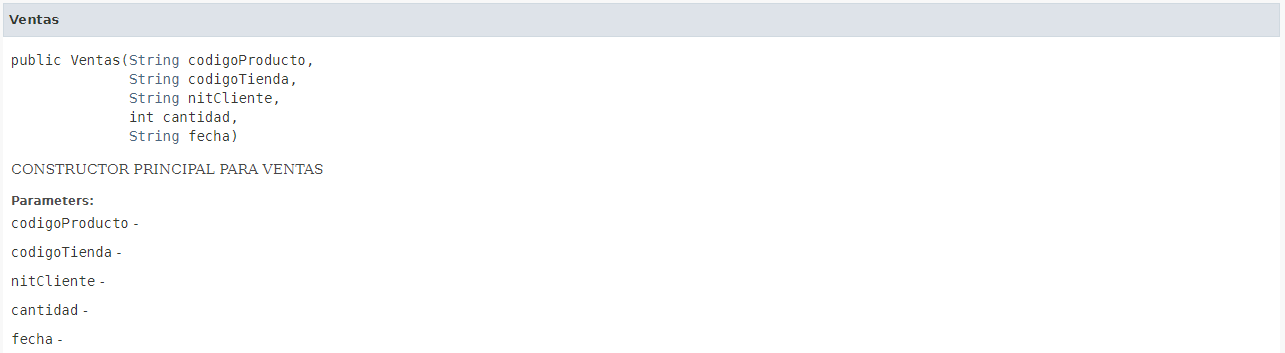


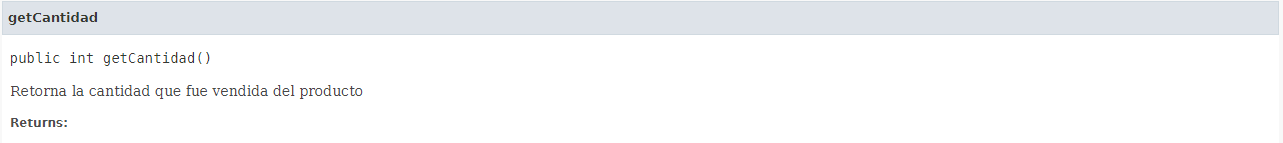


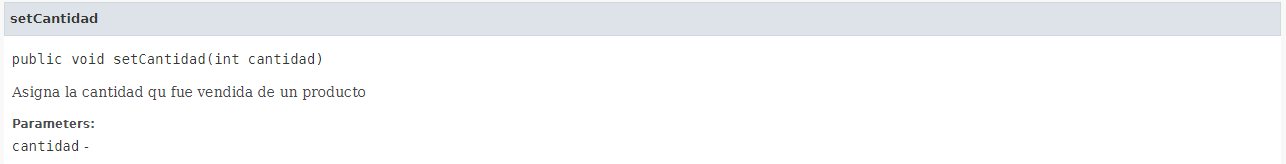


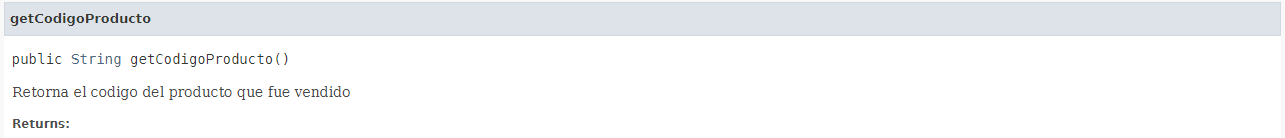


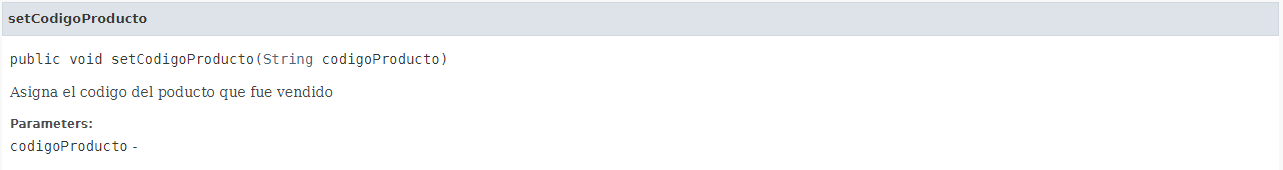
### Clase Ventas:

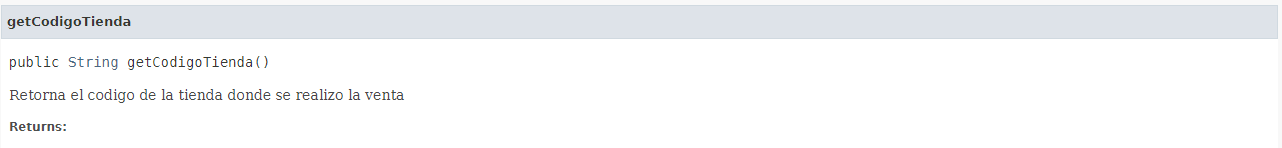


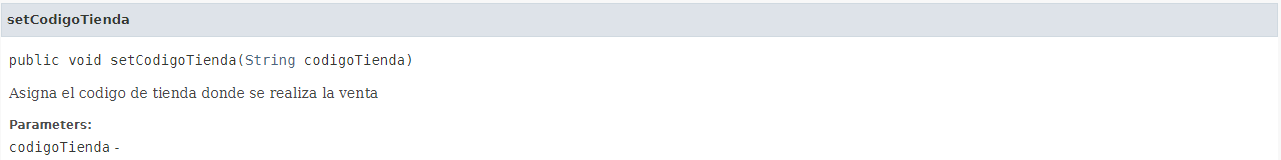


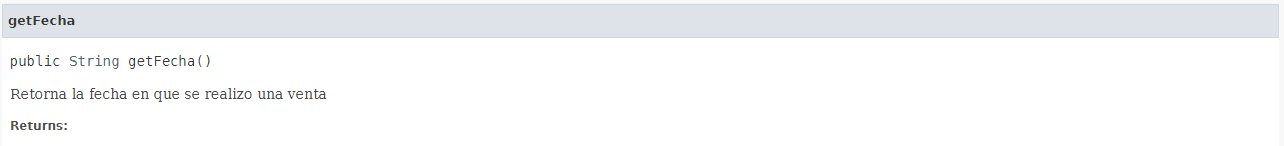


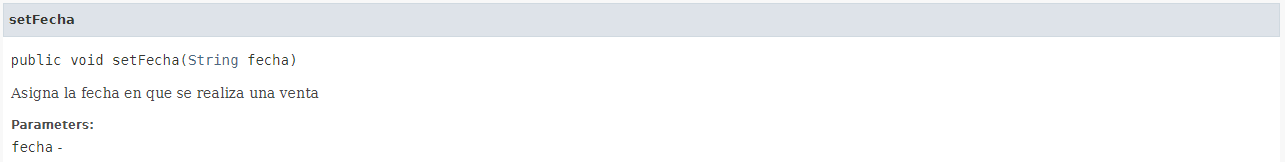


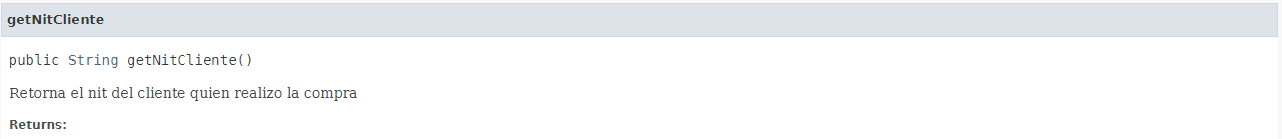


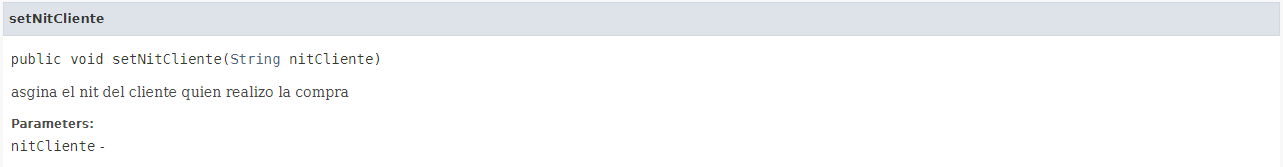












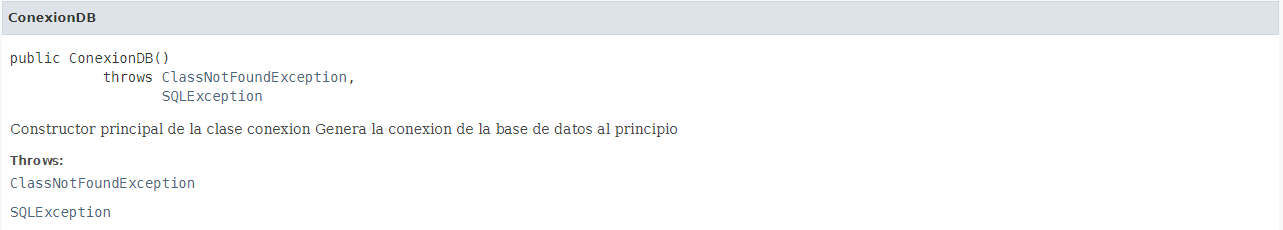
## Base de Datos

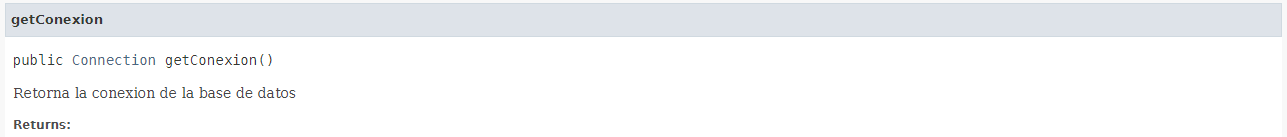
La conexión a la base de datos Mysql se encuentra en el paquete DBSuport donde la clase ConexiónDB es la encargada de manejar los accesos para poder acceder a la información de la base de datos.

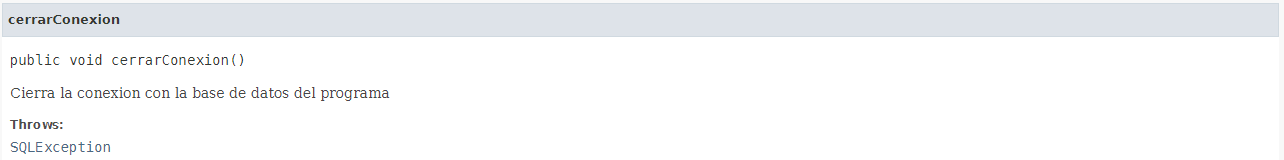
|  |  |
| --- | --- |
| Nombre de Objeto | Descripción de la Objeto |
| ConexionDB | Clase que maneja la conexión hacia la base de datos es la única abierta durante la ejecución del todo programa. |
| ConsultasDB | Clase que maneja las Sentencias de consultas SQL de la cual se alimenta todo el programa. |
| ModificacionesDB | Clase que maneja la modificación de la información ya introducida en la base de datos. |
| RegistroDB | Clase encargada del registro de las nuevas entidades en la base de datos. |

Cabe mencionar que ConsultasDB, ModificacionsDB y RegistroDB, en todos sus métodos poseen similitudes para la entrega de datos, por lo cual se describirá el funcionamiento de una sola, ya que la lógica es la misma en todas, solamente se aclarara la consulta que se hace en el programa.

### ConexionDB

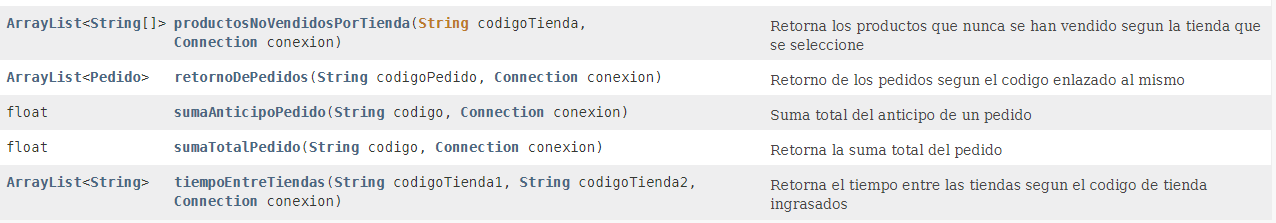
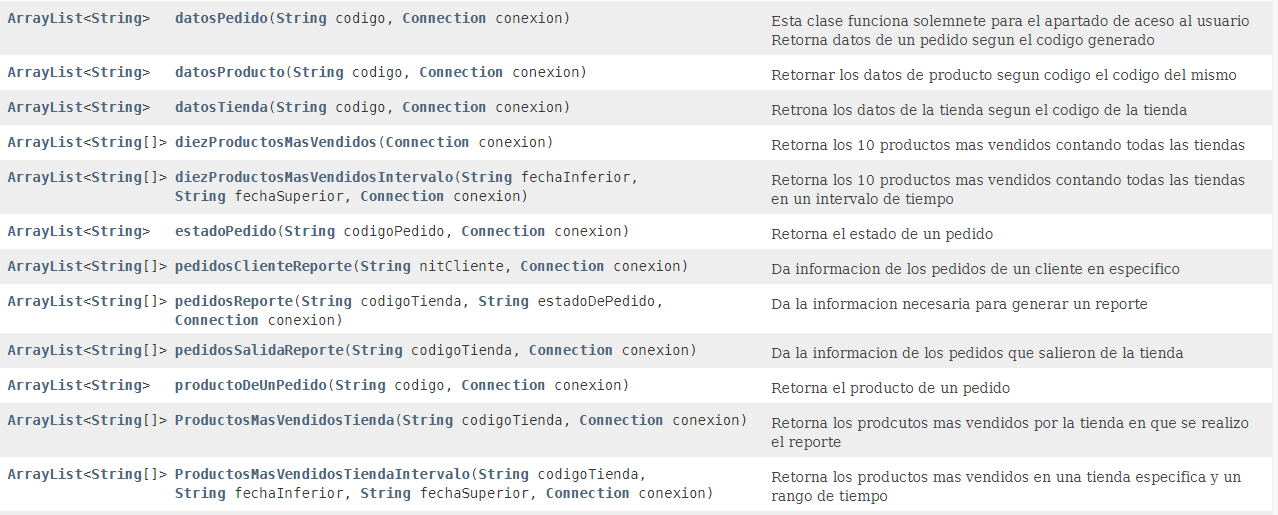
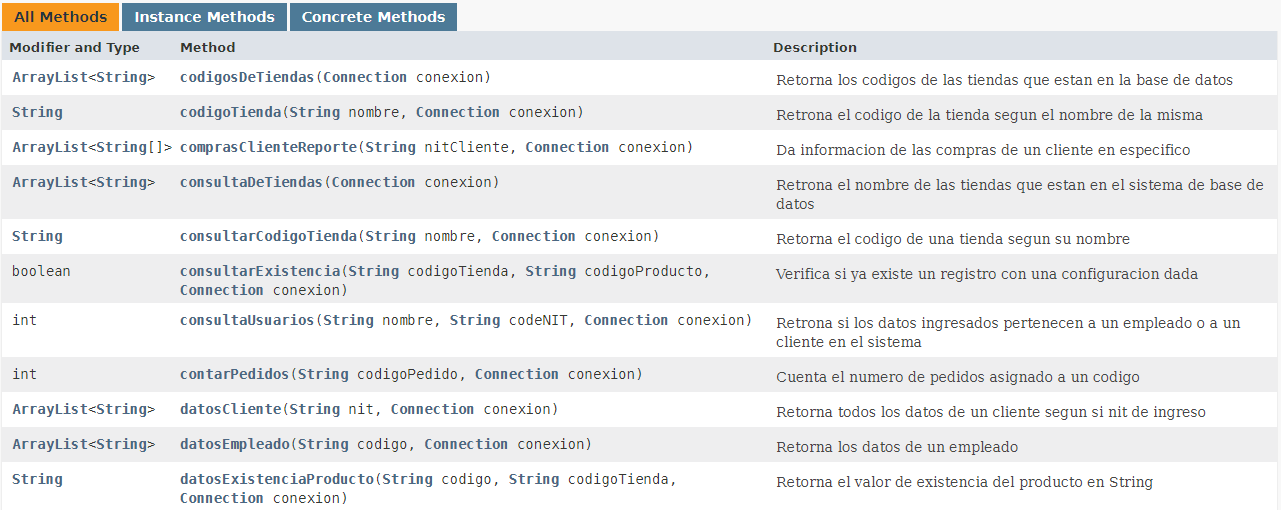






### ConsultasDB

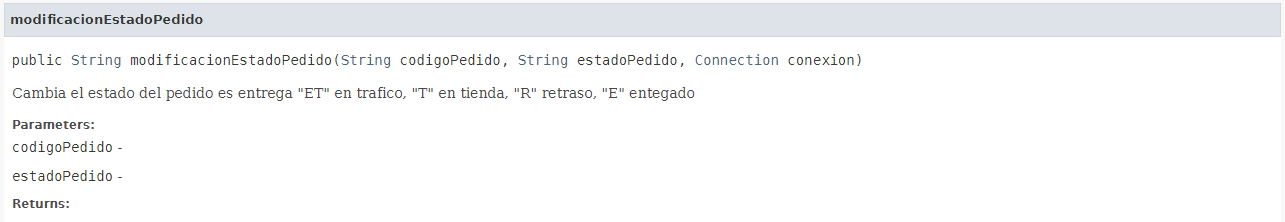
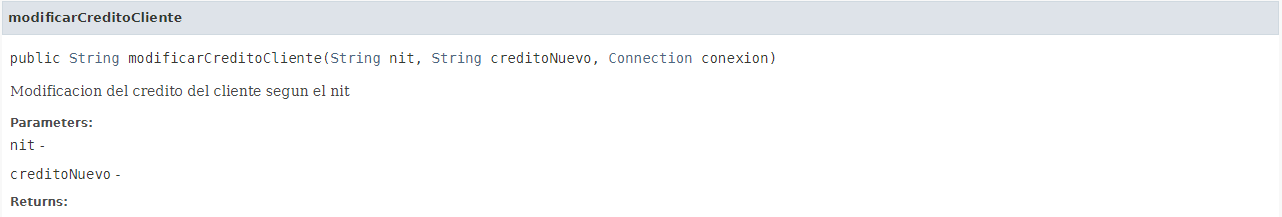
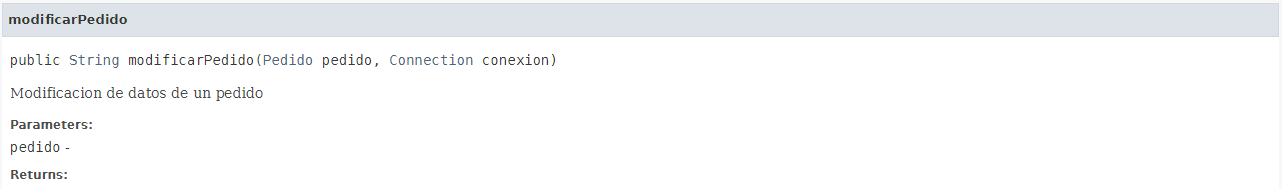
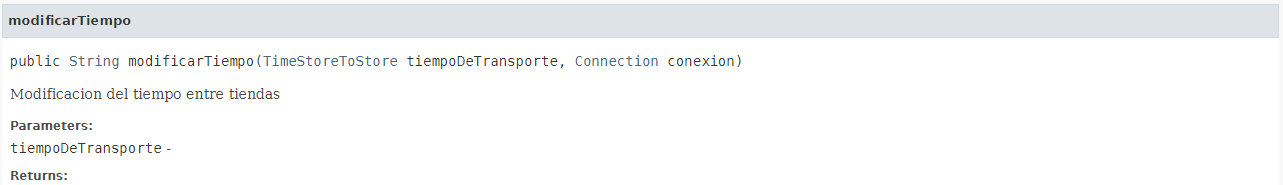
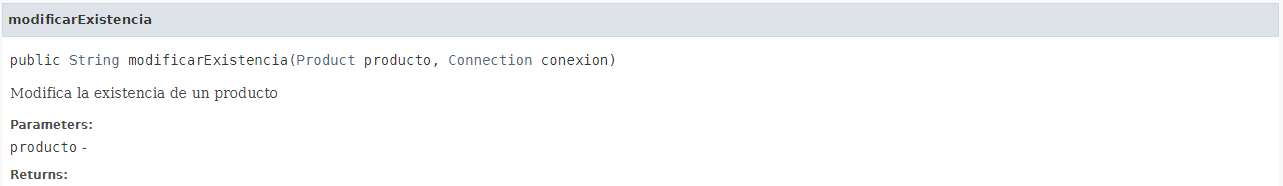
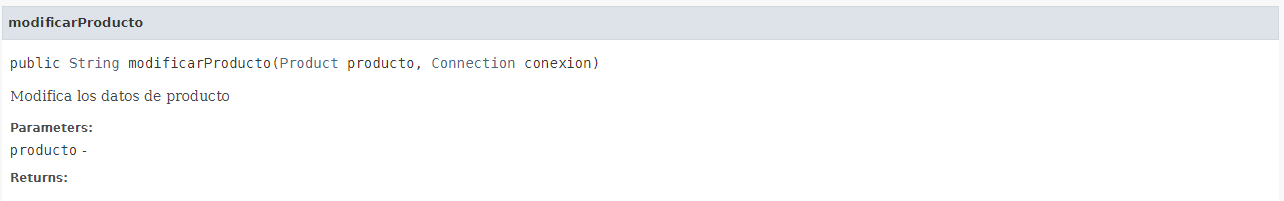
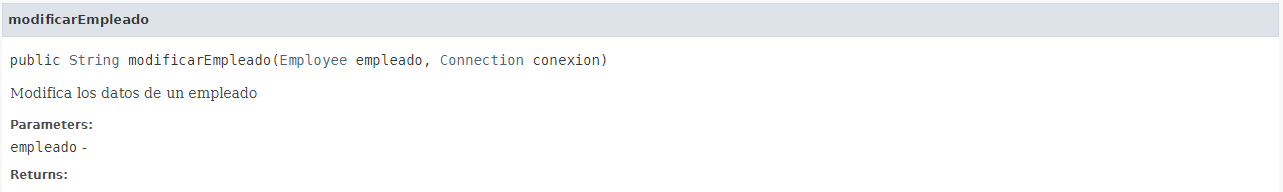
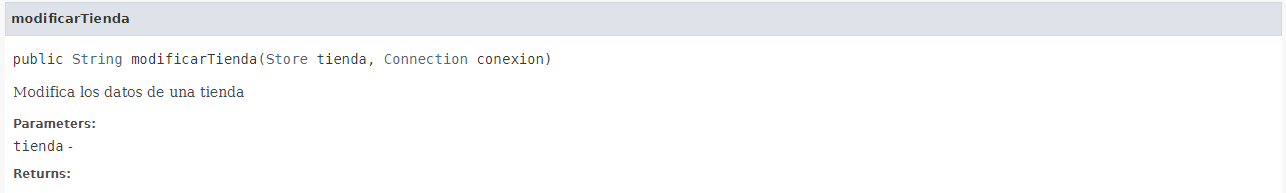
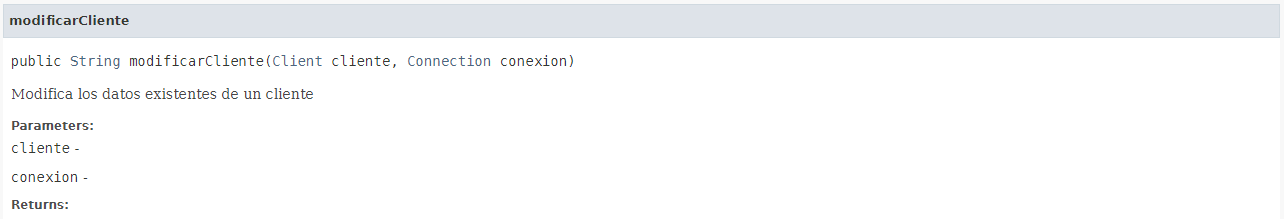
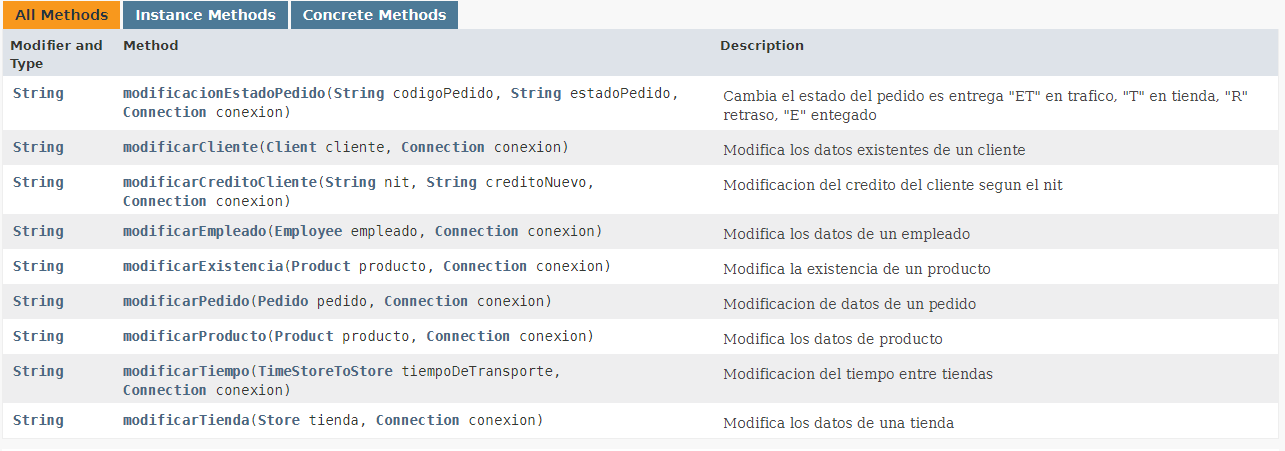
Posee un constructor vacío.



|  |  |
| --- | --- |
| Método | Consulta de SQL |
| codigosDeTiendas | SELECT codigo FROM TIENDA |
| codigoTienda | SELECT codigo FROM TIENDA WHERE nombre = ? |
| comprasClienteReporte​ | SELECT TIENDA\_codigo,PRODUCTO\_codigo,fecha\_venta,cantidad\_producto FROM VENTAS WHERE CLIENTE\_nit = ? |
| consultaDeTiendas | SELECT nombre FROM TIENDA |
| consultarCodigoTienda | SELECT codigo FROM TIENDA WHERE nombre = ? |
| consultarExistencia | SELECT id FROM EXISTENCIA WHERE TIENDA\_codigo = ? AND PRODUCTO\_codigo = ? |
| consultaUsuarios | SELECT nombre,nit FROM CLIENTE WHERE nit = ? AND nombre = ? |
| contarPedidos | SELECT COUNT(\*) FROM PEDIDO WHERE codigo = ? |
| datosCliente | SELECT nombre,telefono,credito,dpi,email,direccion FROM CLIENTE WHERE nit = ? |
| datosEmpleado | SELECT nombre,telefono,dpi,nit,email,direccion FROM EMPLEADO WHERE codigo = ? |
| datosExistenciaProducto​ | SELECT cantidad FROM EXISTENCIA WHERE PRODUCTO\_codigo = ? AND TIENDA\_codigo = ? |
| datosPedido | SELECT anticipo,TIENDA\_codigo\_salida,TIENDA\_codigo\_llegada FROM PEDIDO WHERE codigo = ? LIMIT 1 |
| datosProducto | SELECT anticipo,TIENDA\_codigo\_salida,TIENDA\_codigo\_llegada FROM PEDIDO WHERE codigo = ? LIMIT 1 |
| datosTienda | SELECT nombre,direccion,telefono\_1,telefono\_2,email,horario FROM TIENDA WHERE codigo = ? |
| diezProductosMasVendidos | SELECT P.nombre,V.PRODUCTO\_codigo,P.fabricante,sum(V.cantidad\_producto) FROM ventas V INNER JOIN producto P ON V.PRODUCTO\_codigo=P.codigo GROUP BY(V.PRODUCTO\_codigo) ORDER BY sum(V.cantidad\_producto) DESC LIMIT 10 |
| diezProductosMasVendidosIntervalo | SELECT P.nombre,V.PRODUCTO\_codigo,P.fabricante,sum(V.cantidad\_producto) FROM ventas V INNER JOIN producto P ON V.PRODUCTO\_codigo=P.codigo AND V.fecha\_venta BETWEEN ? AND ? GROUP BY(V.PRODUCTO\_codigo) ORDER BY sum(V.cantidad\_producto) DESC LIMIT 10 |
| estadoPedido | SELECT cantidad,total,anticipo,fecha\_orden,estado\_pedido,CLIENTE\_nit,PRODUCTO\_codigo,TIENDA\_codigo\_salida,TIENDA\_codigo\_llegada FROM PEDIDO WHERE codigo = ? LIMIT 1 |
| pedidosClienteReporte | SELECT codigo,PRODUCTO\_codigo,cantidad,TIENDA\_codigo\_salida,TIENDA\_codigo\_llegada FROM PEDIDO WHERE CLIENTE\_nit = ? AND estado\_pedido != 'entregado' |
| pedidosReporte​ | SELECT PEDIDO.codigo,CLIENTE.nombre,PEDIDO.CLIENTE\_nit,PEDIDO.TIENDA\_codigo\_salida FROM PEDIDO,CLIENTE WHERE PEDIDO.TIENDA\_codigo\_llegada = ? AND PEDIDO.CLIENTE\_nit = CLIENTE.nit AND PEDIDO.estado\_pedido= ? |
| pedidosSalidaReporte | SELECT PEDIDO.codigo,CLIENTE.nombre,PEDIDO.CLIENTE\_nit,PEDIDO.TIENDA\_codigo\_llegada FROM PEDIDO,CLIENTE WHERE PEDIDO.TIENDA\_codigo\_salida = ? AND PEDIDO.CLIENTE\_nit = CLIENTE.nit AND PEDIDO.estado\_pedido='ET' |
| productoDeUnPedido | SELECT PRODUCTO\_codigo FROM PEDIDO WHERE codigo = ? |
| ProductosMasVendidosTienda | SELECT PRODUCTO.codigo,PRODUCTO.nombre,PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO\_codigo FROM VENTAS WHERE VENTAS.TIENDA\_codigo = ? GROUP BY PRODUCTO\_codigo ORDER BY count(PRODUCTO\_codigo) DESC) |
| ProductosMasVendidosTiendaIntervalo | SELECT PRODUCTO.codigo,PRODUCTO.nombre,PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO\_codigo FROM VENTAS WHERE VENTAS.TIENDA\_codigo = ? AND fecha\_venta BETWEEN ? AND ? GROUP BY PRODUCTO\_codigo ORDER BY count(PRODUCTO\_codigo) DESC) |
| productosNoVendidosPorTienda | SELECT PRODUCTO.codigo,PRODUCTO.nombre,PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO\_codigo FROM EXISTENCIA WHERE ((SELECT COUNT(\*) FROM VENTAS WHERE VENTAS.TIENDA\_codigo = ? AND VENTAS.PRODUCTO\_codigo = EXISTENCIA.PRODUCTO\_codigo) = 0) AND EXISTENCIA.TIENDA\_codigo= ? ) |
| retornoDePedidos | SELECT cantidad,total,anticipo,fecha\_orden,estado\_pedido,CLIENTE\_nit,PRODUCTO\_codigo,TIENDA\_codigo\_salida,TIENDA\_codigo\_llegada FROM PEDIDO WHERE codigo = ? |
| sumaAnticipoPedido​ | SELECT SUM(anticipo) FROM PEDIDO WHERE codigo = ? |
| sumaTotalPedido​ | SELECT SUM(total) FROM PEDIDO WHERE codigo = ? |
| tiempoEntreTiendas​ | SELECT tiempo,codigo FROM TIEMPO\_TRASLADO WHERE TIENDA\_codigo1 = ? AND TIENDA\_codigo2 = ? |

### ModificacionesDB

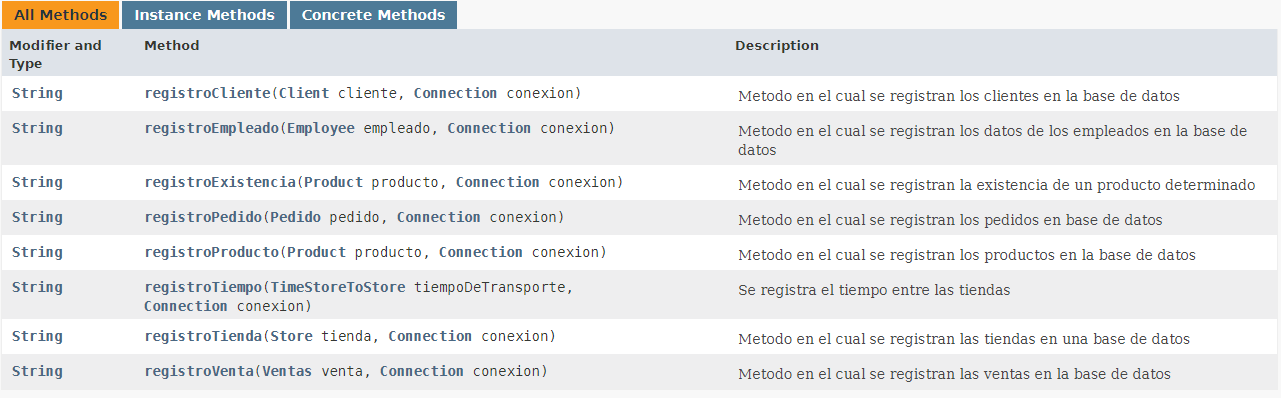
Constructor de la clase vacío

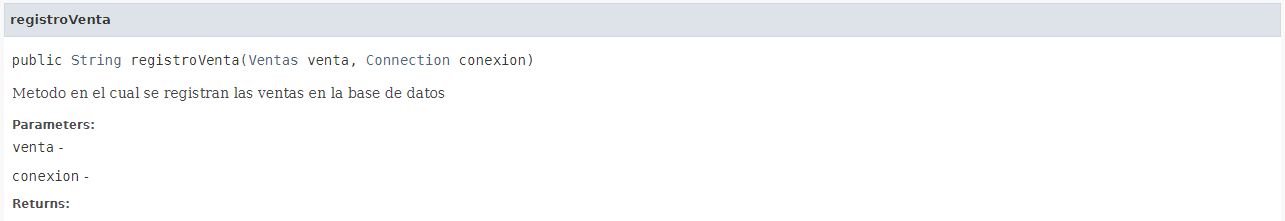
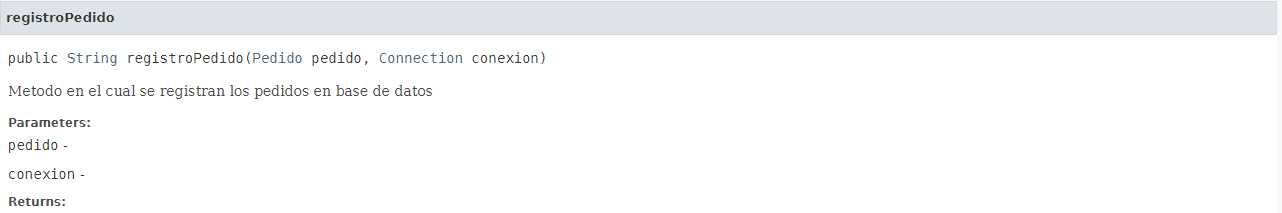
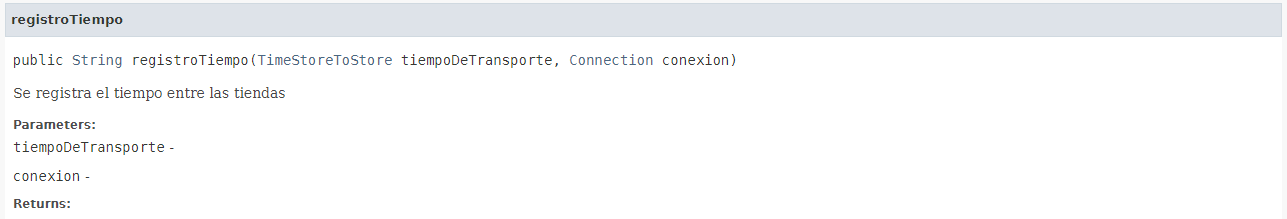
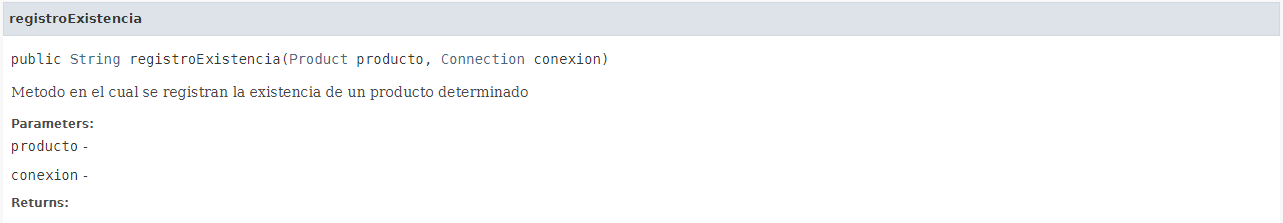
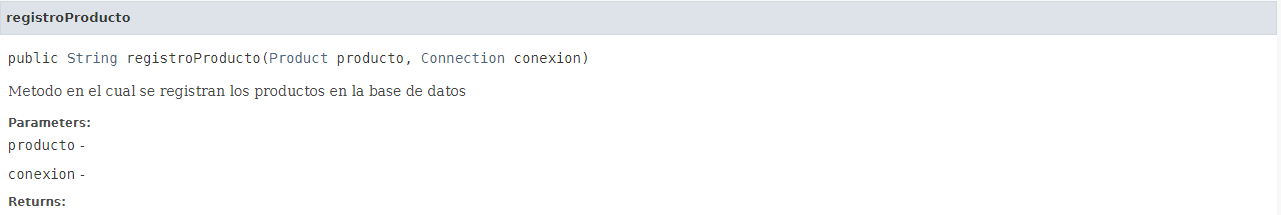
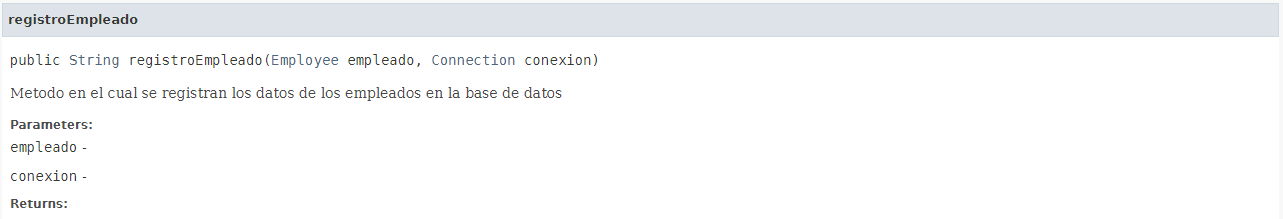
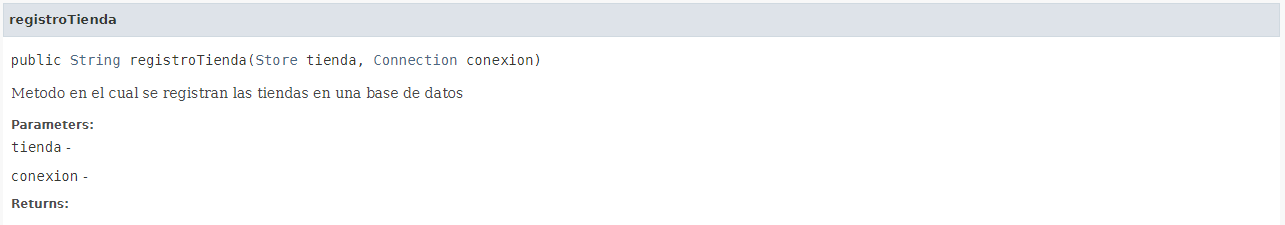
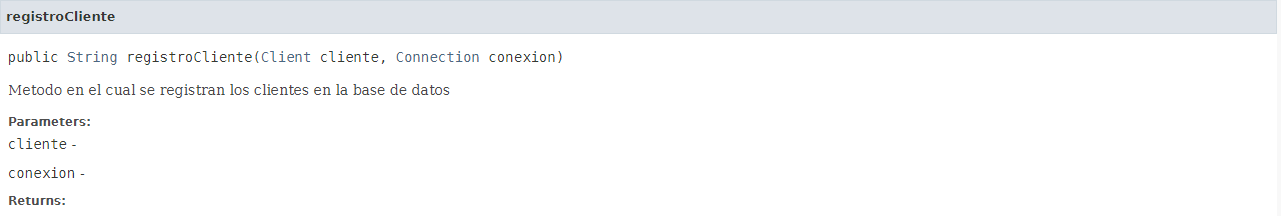


|  |  |
| --- | --- |
| Método | Sentencia de SQL |
| modificacionEstadoPedido | UPDATE PEDIDO SET estado\_pedido = ? WHERE codigo = ? |
| modificarCliente | UPDATE CLIENTE SET nombre = ?,telefono = ?,credito = ?,dpi = ?,email = ?,direccion = ? WHERE nit = ? |
| modificarCreditoCliente | UPDATE CLIENTE SET credito = ? where nit= ? |
| modificarEmpleado | UPDATE EMPLEADO SET nombre = ?,telefono = ?,dpi = ?,nit = ?,email = ?,direccion = ? WHERE codigo = ? |
| modificarExistencia | UPDATE EXISTENCIA SET cantidad = ? WHERE TIENDA\_codigo = ? AND PRODUCTO\_codigo = ? |
| ModificarPedido | UPDATE PEDIDO SET cantidad = ?,total = ?,anticipo = ?,fecha\_orden = ?,CLIENTE\_nit = ?,PRODUCTO\_codigo = ?,TIENDA\_codigo\_salida = ?,TIENDA\_codigo\_llegada = ? WHERE codigo = ? |
| modificarProducto | UPDATE PRODUCTO SET nombre = ?,fabricante = ?,precio = ?,descripcion = ?,garantia = ? WHERE codigo = ? |
| modificarTiempo | UPDATE TIEMPO\_TRASLADO SET tiempo = ? WHERE TIENDA\_codigo1 = ? AND TIENDA\_codigo2 = ? |
| modificarTienda​ | UPDATE TIENDA SET nombre = ?,direccion = ?,telefono\_1 = ?,telefono\_2 = ?,email = ?,horario = ? WHERE codigo = ? |

### RegistroDB

La clase posee un constructor vacío.



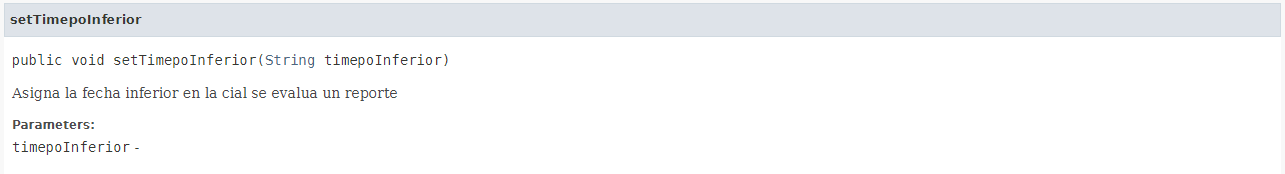
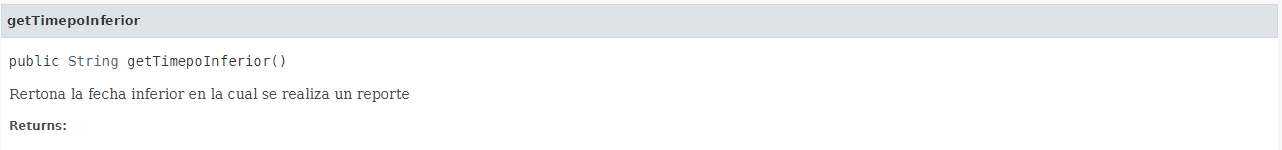
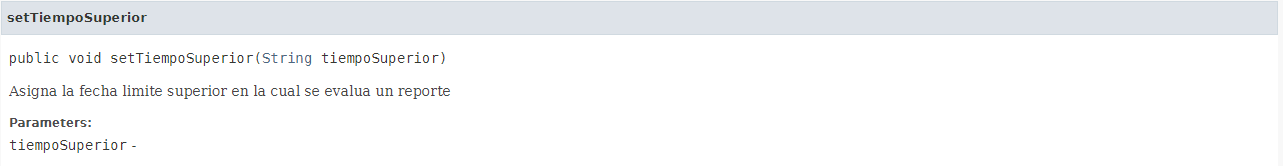
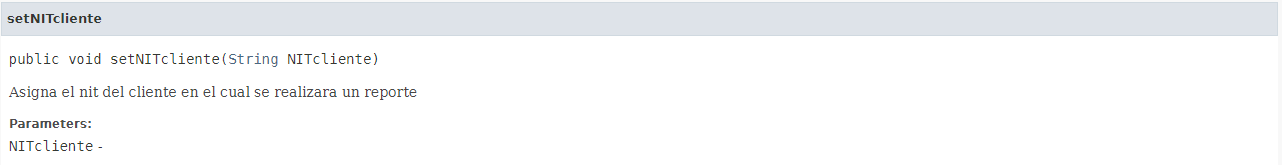
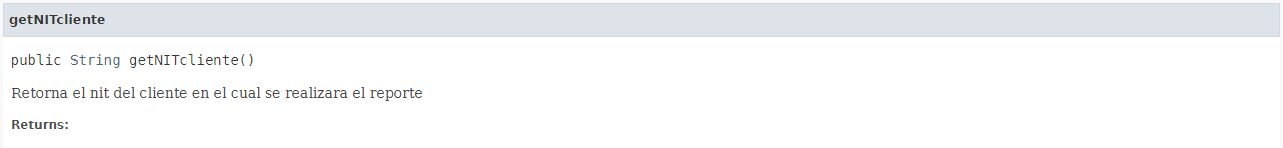
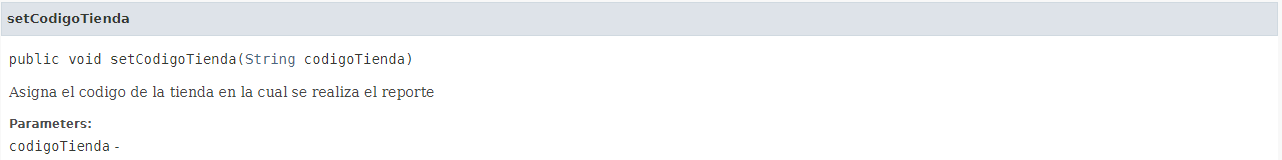
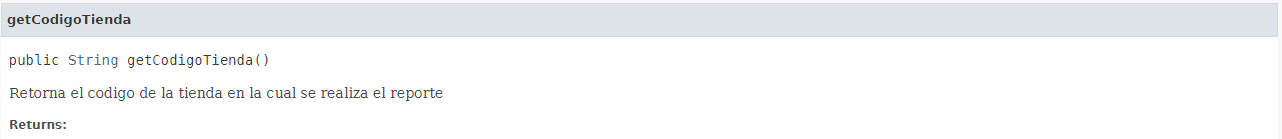
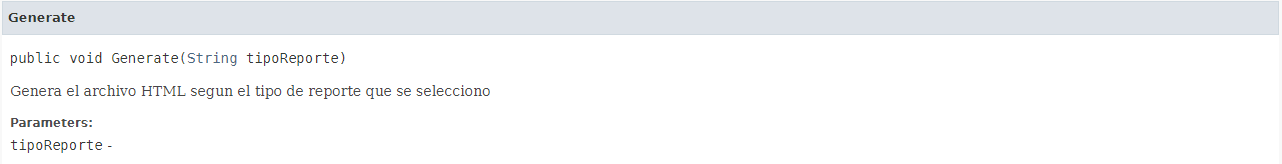
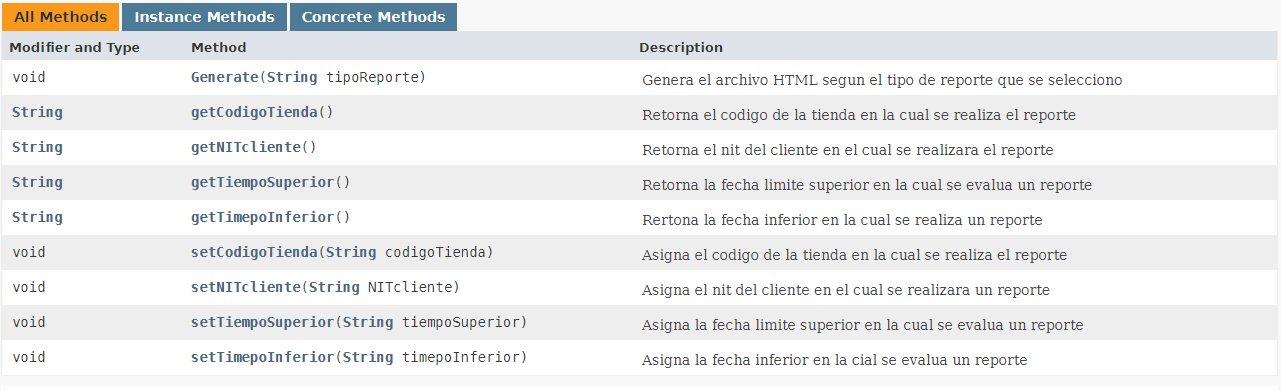
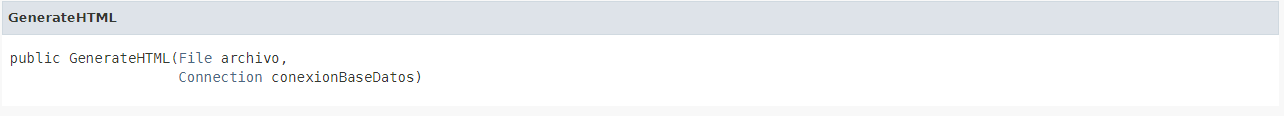


|  |  |  |
| --- | --- | --- |
| Método | Sentencia de SQL | |
| registroCliente​ | | INSERT INTO CLIENTE VALUES (?,?,?,?,?,?,?) | |
| registroTienda | | INSERT INTO TIENDA VALUES (?,?,?,?,?,?,?) | |
| registroEmpleado | | INSERT INTO EMPLEADO VALUES (?,?,?,?,?,?,?) | |
| registroProducto | | INSERT INTO PRODUCTO VALUES (?,?,?,?,?,?) | |
| registroExistencia | | INSERT INTO EXISTENCIA (TIENDA\_codigo,PRODUCTO\_codigo,cantidad) VALUES (?,?,?) | |
| registroTiempo | | INSERT INTO TIEMPO\_TRASLADO (TIENDA\_codigo1,TIENDA\_codigo2,tiempo) VALUES (?,?,?) | |
| registroPedido | | INSERT INTO PEDIDO (codigo,cantidad,total,anticipo,fecha\_orden,estado\_pedido,CLIENTE\_nit,PRODUCTO\_codigo,TIENDA\_codigo\_salida,TIENDA\_codigo\_llegada) VALUES (?,?,?,?,?,?,?,?,?,?) | |
| registroVenta | | INSERT INTO VENTAS (TIENDA\_codigo,PRODUCTO\_codigo,CLIENTE\_nit,fecha\_venta,cantidad\_producto) VALUES (?,?,?,?,?) | |

## Generación De Archivos

La generación de archivos es utilizada al momento de generar los reportes del programa, en este caso generada través de un archivo con formato HTML para poder visualizar la información que está dentro del programa, mediante la Clase GenerateHTML la cual escribe el formato del archivo con los métodos que contiene.

GenerateHTML es una clase que se apoya por medio de las consultas descritas anteriormente desde el programa a la base de datos.



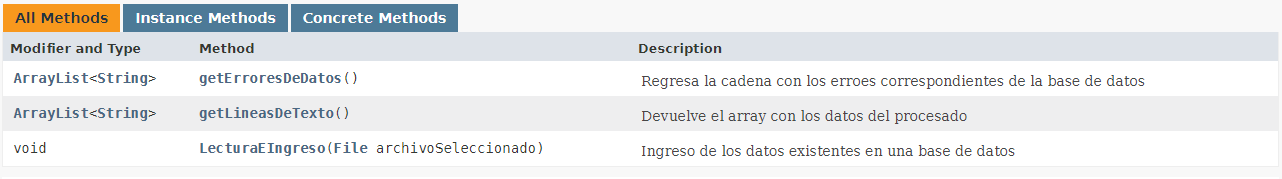
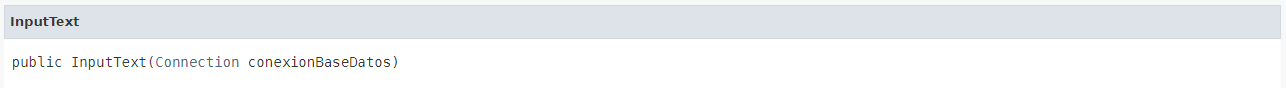
El conjunto anterior de métodos solo representa los atributos de la clase, los encardos de realizar el armado del archivo HTML son los métodos del tipo PrintWriter.

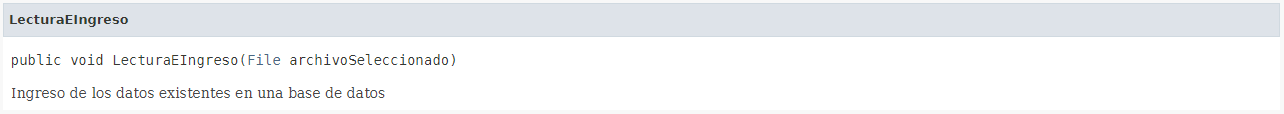
|  |  |
| --- | --- |
| Nombre del método | Tipo de Reporte encargado a realizar |
| htmlContentPedidosTienda | Realiza el archivo HTML en el cual reporta que pedidos llegaran a la tienda. |
| htmlContentPedidosAtrazadosTienda | Realiza el archivo HTML en el cual reporta que pedidos llegaron con atrasó a la tienda. |
| htmlContentPedidosExpendidosPorTienda | Realiza el archivo HTML en el cual reporta que pedidos salieron de la tienda. |
| htmlPedidosPorCliente | Realiza el archivo HTML en el cual reporta que pedidos fueron hechos por un cliente. |
| htmlDiezProductosMasVendidos | Realiza el archivo HTML en el cual reporta cuales son los 10 productos más vendidos. |
| htmlDiezProductosMasVendidosIntervalo | Realiza el archivo HTML en el cual reporta cuales son los 10 productos más vendidos en un intervalo de tiempo. |
| htmlProductosMasVendidosTienda | Realiza el archivo HTML en el cual reporta cuales son los producto más vendidos por una tienda. |
| htmlProductosMasVendidosTiendaIntervalo | Realiza el archivo HTML en el cual reporta cuales son los productos más vendidos en un tienda en un intervalo de tiempo. |
| htmlProductosNoVendidosPorTienda | Realiza el archivo HTML en el cual reporta cuales son los productos que no se han vendido. |

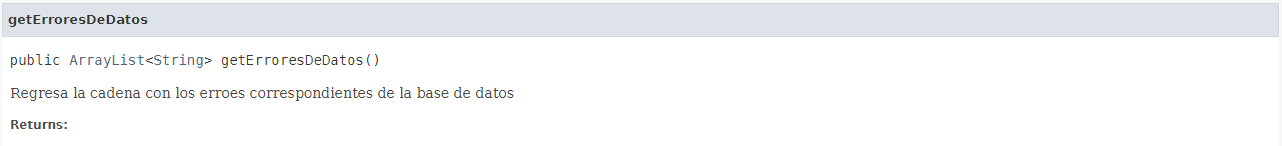
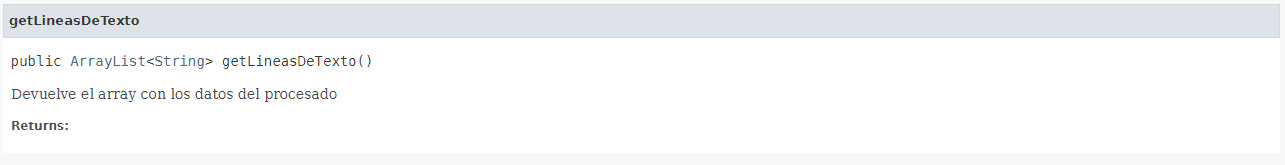
## Lectura de Archivo de Carga

Este apartado es el encargado de realizar la lectura de un archivo de información de la tienda e introducirlo en la base de datos.

La clase encargada de esto es InputText.



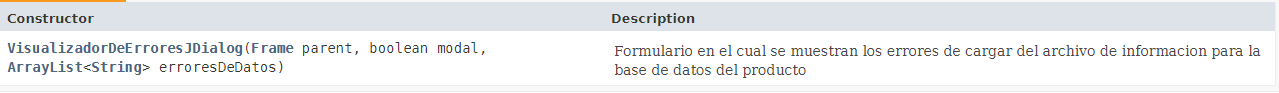
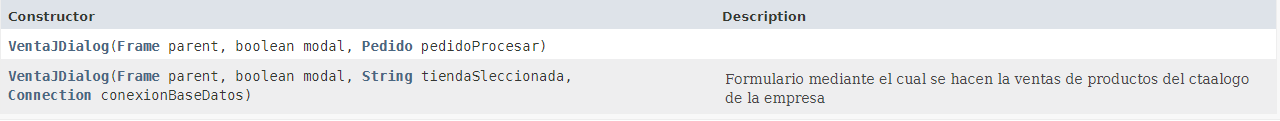
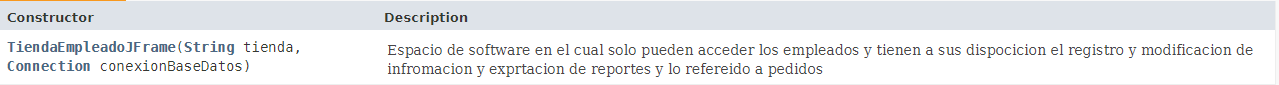
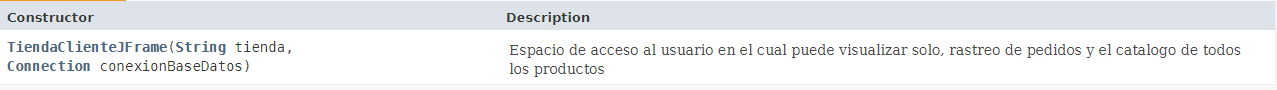
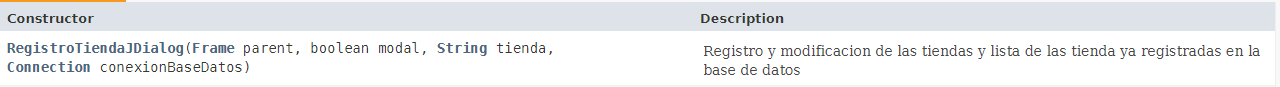
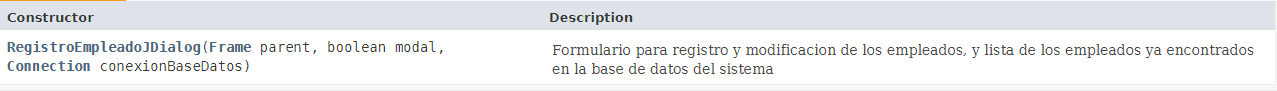
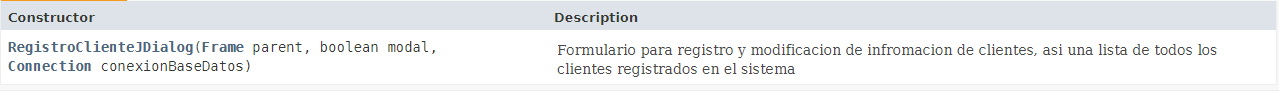
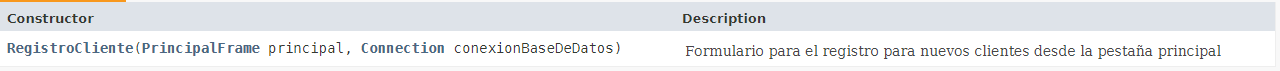
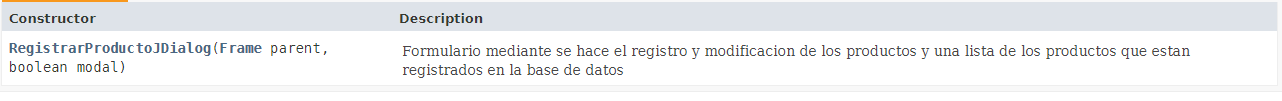
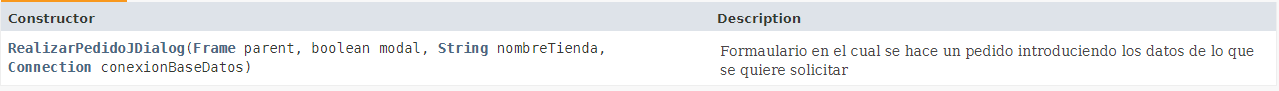
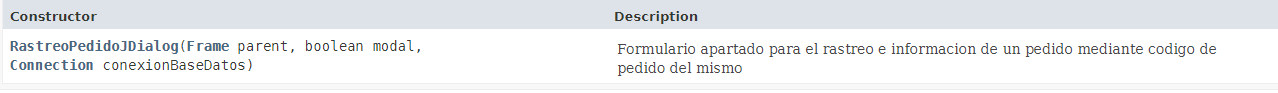
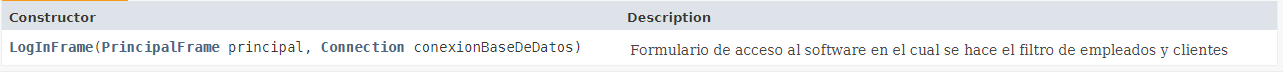
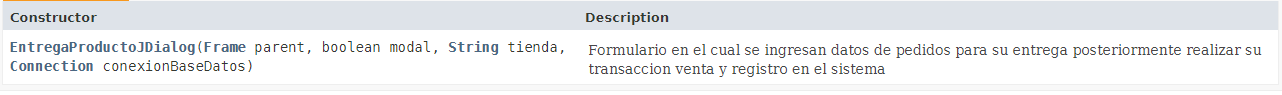
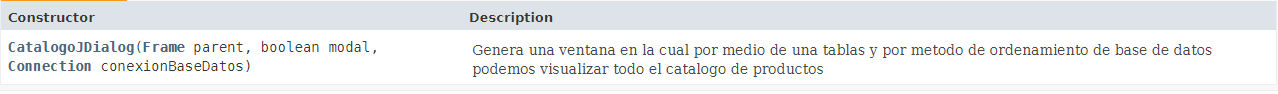
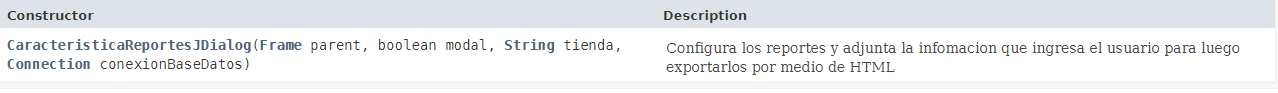


La descomposición lógica del archivo para la carga de información está dada en reconocer principalmente de una palabra clave para el inicio de la información, el resto de operación se realizan de la siguiente manera:

1. Identificar la cantidad de separadores de información.
2. Cada palabra clave tiene su respectivo número de datos.
3. Realizar Filtros de información, con los cuales pueda identificar texto, números, teléfonos en las posiciones requeridas.
4. Utilizar la entidad relacionada con la información procesada para ingresarla en la base de datos
5. Manejar los errores al momento de ingresar los datos en base de datos.

## Interfaz Grafica

Información de los elementos pertenecientes a la interfaz gráfica del sistema.



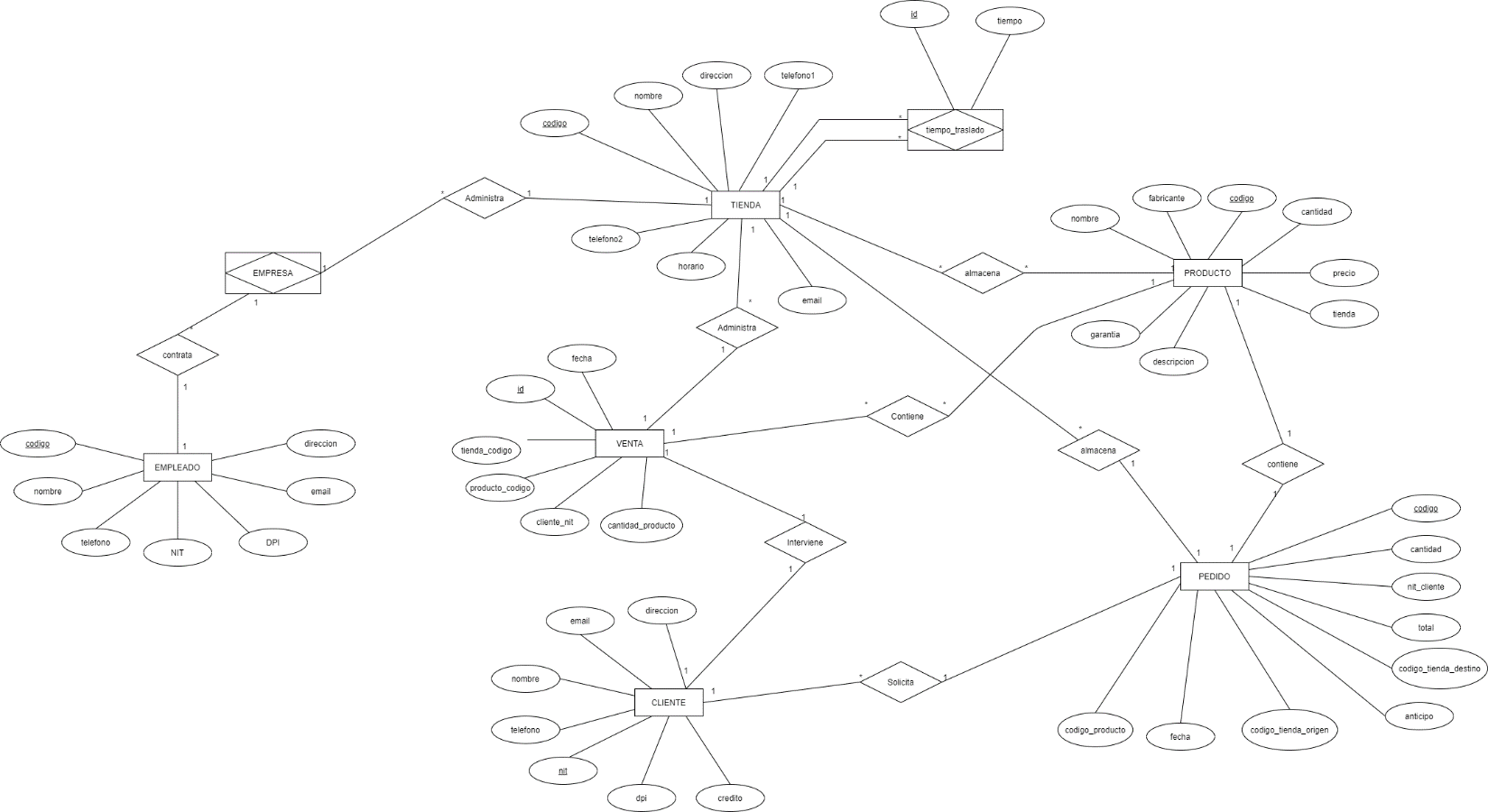
# DIAGRAMAS DE LA ESTRUCTURACION DEL SOFTWARE

## Diagrama de clases UML:

Debido a su gran tamaño en diagrama es detallado en un archivo diferente con formato HTML para poder visualizar de mejor manera la relacione que existen entre las clases del software realizado.

## Diagrama Entidad Relación:

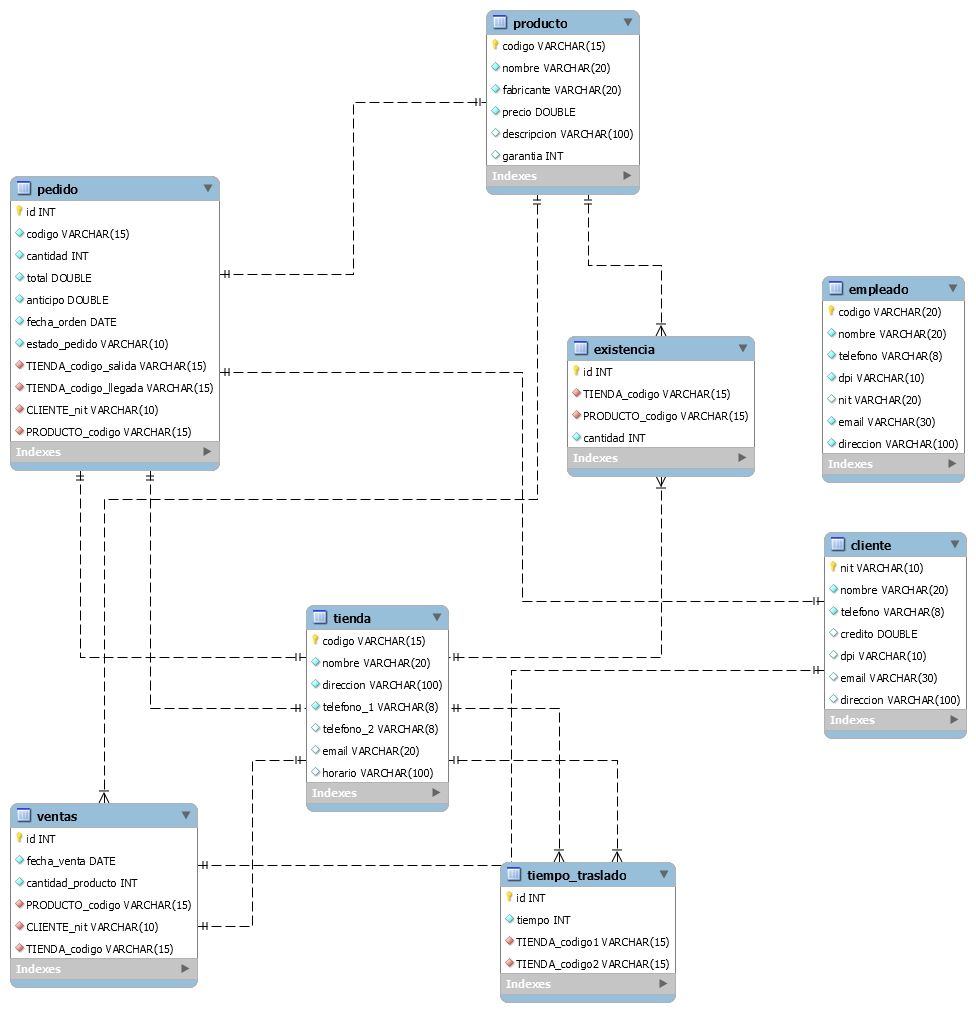
Aparte de este medio se proporciona un Diagrama de Entidad Relación en formato PNG



## ESTRUCTURACION DE LA BASE DE DATOS

Estructuración de la base de datos Mysql donde se presentan las tablas donde se almacenan los datos:

También se adjunta un mapeo físico del proyecto:



Código SQL para la generación de la base de datos:

CREATE SCHEMA IF NOT EXISTS mi\_empresa\_proyecto1;

USE mi\_empresa\_proyecto1;

CREATE TABLE IF NOT EXISTS TIENDA(

codigo VARCHAR(15) NOT NULL,

nombre VARCHAR(20) NOT NULL,

direccion VARCHAR(100) NOT NULL,

telefono\_1 VARCHAR(8) NOT NULL,

telefono\_2 VARCHAR(8),

email VARCHAR(20),

horario VARCHAR(100),

PRIMARY KEY (codigo),

UNIQUE (codigo,nombre)

);

CREATE TABLE IF NOT EXISTS CLIENTE (

nit VARCHAR(10) NOT NULL,

nombre VARCHAR(20) NOT NULL,

telefono VARCHAR(8) NOT NULL,

credito DOUBLE,

dpi VARCHAR(10),

email VARCHAR(30),

direccion VARCHAR(100),

PRIMARY KEY (nit),

UNIQUE (nit,dpi)

);

CREATE TABLE IF NOT EXISTS EMPLEADO (

codigo VARCHAR(20) NOT NULL,

nombre VARCHAR(20) NOT NULL,

telefono VARCHAR(8) NOT NULL,

dpi VARCHAR(10) NOT NULL,

nit VARCHAR(20),

email VARCHAR(30) NOT NULL,

direccion VARCHAR(100) NOT NULL,

PRIMARY KEY (codigo),

UNIQUE (codigo,dpi,nit)

);

CREATE TABLE IF NOT EXISTS PRODUCTO(

codigo VARCHAR(15) NOT NULL,

nombre VARCHAR(20) NOT NULL,

fabricante VARCHAR(20) NOT NULL,

precio DOUBLE NOT NULL,

descripcion VARCHAR(100),

garantia INT,

PRIMARY KEY (codigo),

UNIQUE (codigo)

);

CREATE TABLE IF NOT EXISTS EXISTENCIA(

id INT NOT NULL AUTO\_INCREMENT,

TIENDA\_codigo VARCHAR(15) NOT NULL,

PRODUCTO\_codigo VARCHAR(15) NOT NULL,

cantidad INT NOT NULL,

PRIMARY KEY (id),

UNIQUE (id),

FOREIGN KEY (TIENDA\_codigo) REFERENCES TIENDA(codigo),

FOREIGN KEY (PRODUCTO\_codigo) REFERENCES PRODUCTO(codigo)

);

CREATE TABLE IF NOT EXISTS PEDIDO(

id INT NOT NULL AUTO\_INCREMENT,

codigo VARCHAR(15) NOT NULL,

cantidad INT NOT NULL,

total DOUBLE NOT NULL,

anticipo DOUBLE NOT NULL,

fecha\_orden DATE NOT NULL,

estado\_pedido VARCHAR(10) NOT NULL,

CLIENTE\_nit VARCHAR(10) NOT NULL,

PRODUCTO\_codigo VARCHAR(15) NOT NULL,

TIENDA\_codigo\_salida VARCHAR(15) NOT NULL,

TIENDA\_codigo\_llegada VARCHAR(15) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (CLIENTE\_nit) REFERENCES CLIENTE(nit),

FOREIGN KEY (PRODUCTO\_codigo) REFERENCES PRODUCTO(codigo),

FOREIGN KEY (TIENDA\_codigo\_salida) REFERENCES TIENDA(codigo),

FOREIGN KEY (TIENDA\_codigo\_llegada) REFERENCES TIENDA(codigo)

);

CREATE TABLE IF NOT EXISTS TIEMPO\_TRASLADO(

id INT NOT NULL AUTO\_INCREMENT,

TIENDA\_codigo1 VARCHAR(15) NOT NULL,

TIENDA\_codigo2 VARCHAR(15) NOT NULL,

tiempo INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (TIENDA\_codigo1) REFERENCES TIENDA(codigo),

FOREIGN KEY (TIENDA\_codigo2) REFERENCES TIENDA(codigo)

);

CREATE TABLE IF NOT EXISTS VENTAS(

id INT NOT NULL AUTO\_INCREMENT,

TIENDA\_codigo VARCHAR(15) NOT NULL,

PRODUCTO\_codigo VARCHAR(15) NOT NULL,

CLIENTE\_nit VARCHAR(10) NOT NULL,

fecha\_venta DATE NOT NULL,

cantidad\_producto INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (CLIENTE\_nit) REFERENCES CLIENTE(nit),

FOREIGN KEY (TIENDA\_codigo) REFERENCES TIENDA(codigo),

FOREIGN KEY (PRODUCTO\_codigo) REFERENCES PRODUCTO(codigo)

);