MANUAL TÉCNICO

BODEGA ADMINISTRADORA DE TIENDAS: PROYECTO DEL CURSO IPC2: APLICACIÓN WEB EN JAVA

HERRAMIENTAS UTILIZADAS

● LENGUAJE DE PROGRAMACIÓN JAVA - VERSIÓN 17

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.

El lenguaje Java proporciona:

- El paradigma de la programación orientada a objetos.
- Ejecución de un mismo programa en múltiples sistemas operativos y plataformas.
- Es posible utilizarlo para múltiples propósitos, desde aplicaciones de escritorio hasta en servidores web.
- Tiene una curva de aprendizaje media pero también toma lo mejor de otros lenguajes orientados a objetos, como C++.

También se utilizaron librerías/bibliotecas externas como:

- **JDBC -** Para la conexión a la base de datos
- **Jakarta EE -** Servicios web
- **Apache Tomcat -** Servidor web
- **Bootstrap -** Componentes web predefinidos

● RDBMS (SISTEMA DE BASES DE DATOS RELACIONAL): MYSQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

MySQL es muy utilizado en aplicaciones web, como Joomla, Wordpress, Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

- ## IDE: INTELLIJ IDEA ULTIMATE

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) escrito en Java para desarrollar software informático escrito en Java, Kotlin, Groovy y otros lenguajes basados en la JVM. Está desarrollado por JetBrains (anteriormente conocido como IntelliJ) y está disponible como una edición comunitaria con licencia de Apache 2, y en una edición comercial patentada. Ambos se pueden utilizar para el desarrollo comercial.

- ## GITHUB

Es un sistema de control de versiones de código y gestión de proyectos, a su vez también funciona como una plataforma de estilo red social diseñada para desarrolladores para poder compartir código entre más personas y colaborar en el mismo.

- ## DIAGRAMS.NET - DRAW.IO

Draw.io es una herramienta de creación y edición de diagramas libre que permite la integración con diversas plataformas, principalmente Google Drive, el cual permite la compartición y colaboración de varias personas dentro del mismo proyecto. El software consiste en una aplicación web realizada mayoritariamente en JavaScript y licenciada con Apache v2, lo que la hace funcionar en una amplia gama de navegadores y permite la creación de diagramas, contando con modelos para diversos tipos como pueden ser diagramas UML, esquemas de red, flujogramas, etc. También permite crear colecciones de diagramas e imágenes personalizados para utilizar en los diagramas.

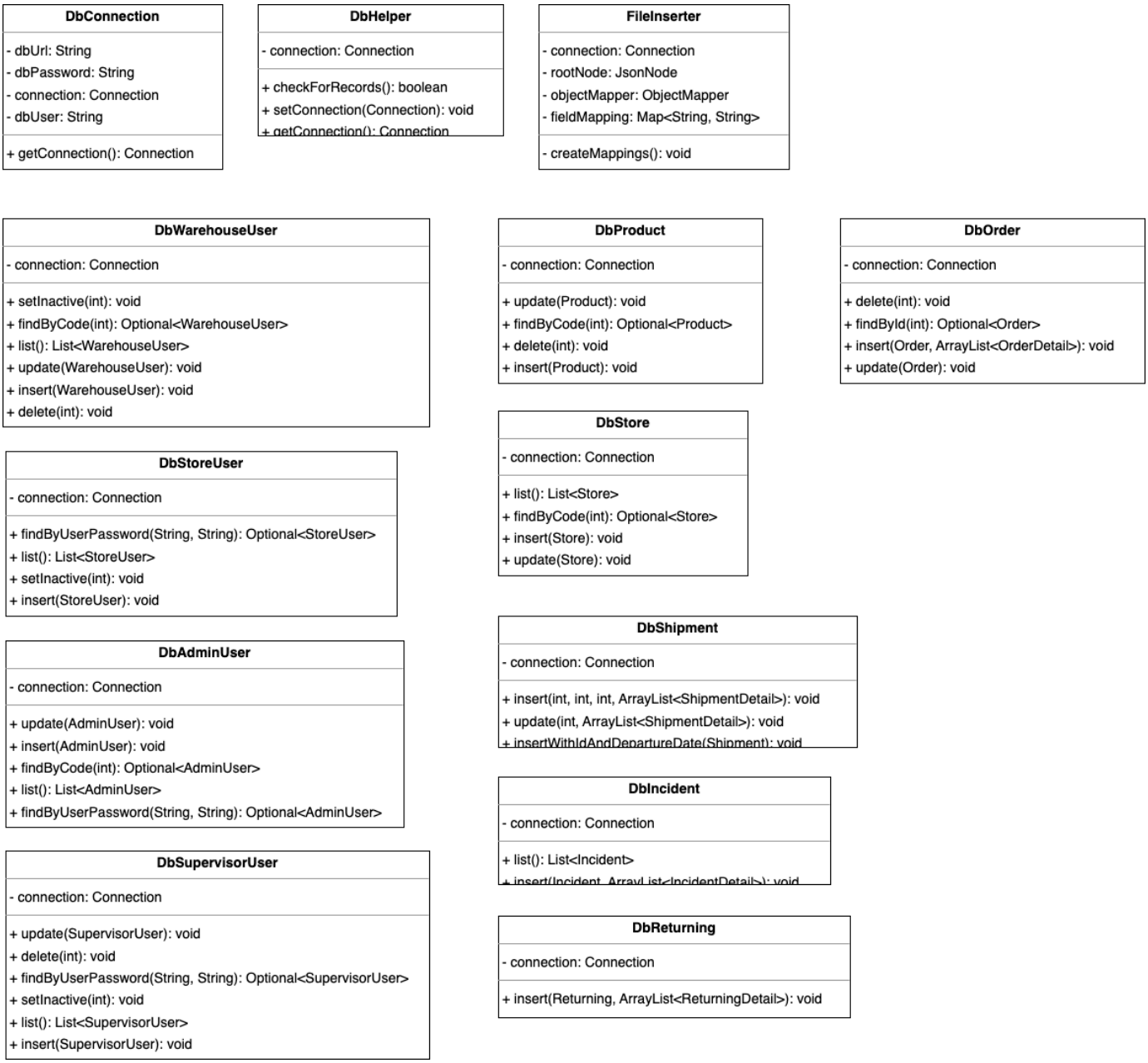
- ## MACOS v13 - SISTEMA OPERATIVO

macOS (previamente Mac OS X, luego OS X) es una serie de sistemas operativos gráficos desarrollados y comercializados por Apple desde 2001. Es el sistema operativo principal para la familia de computadoras Mac de Apple. Dentro del mercado de computadoras de escritorio, portátiles, hogareñas y mediante el uso de la web.

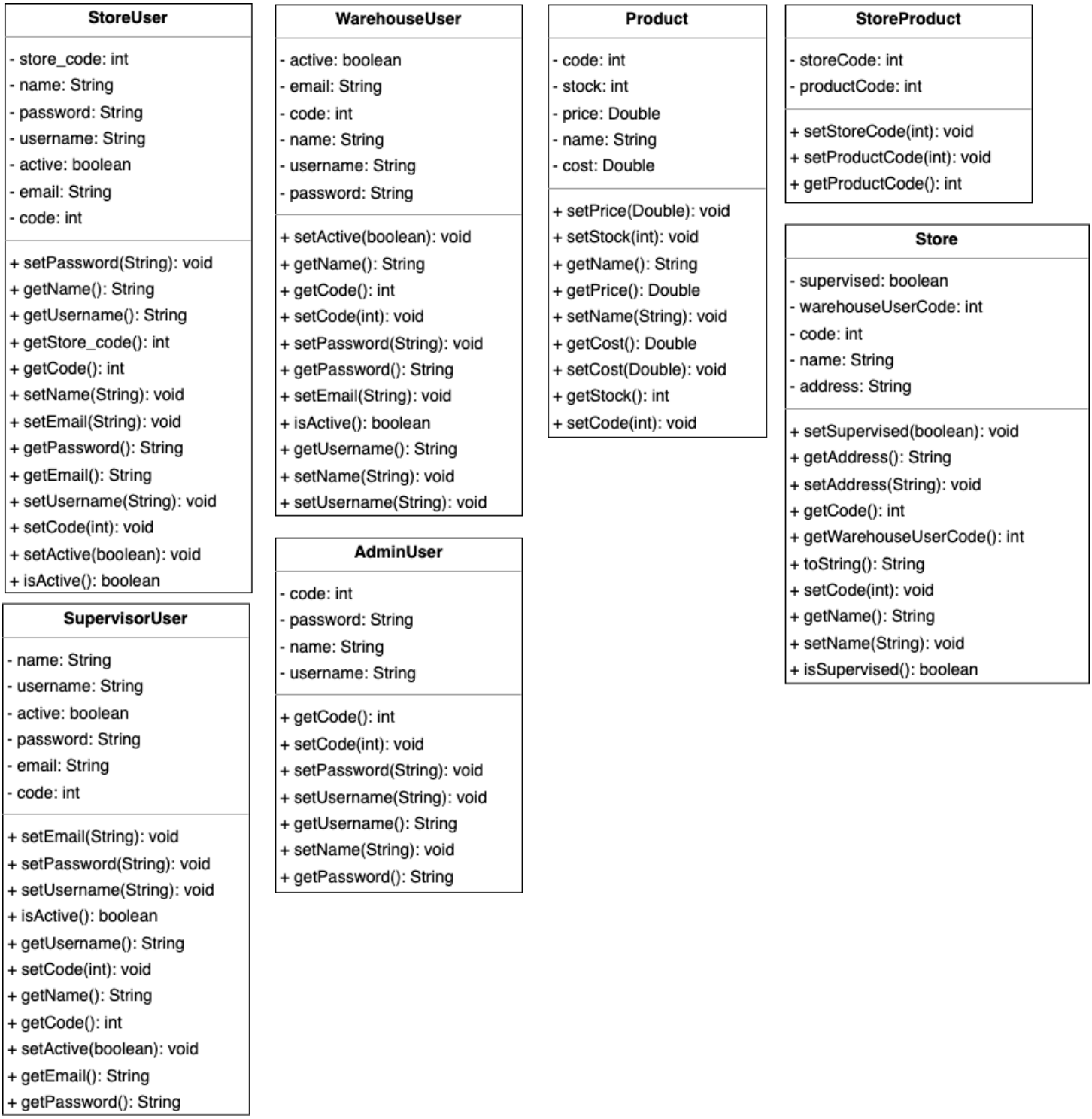
macOS se basa en tecnologías desarrolladas entre 1985 y 1997 en NeXT. Se logró la certificación UNIX 03 para la versión Intel de Mac OS X 10.5 Leopard y todos los lanzamientos de Mac OS X 10.6 Snow Leopard hasta la versión actual también tienen la certificación UNIX 03. macOS comparte su núcleo basado en Unix, llamado Darwin, y muchos de sus frameworks con iOS 16, tvOS y watchOS.

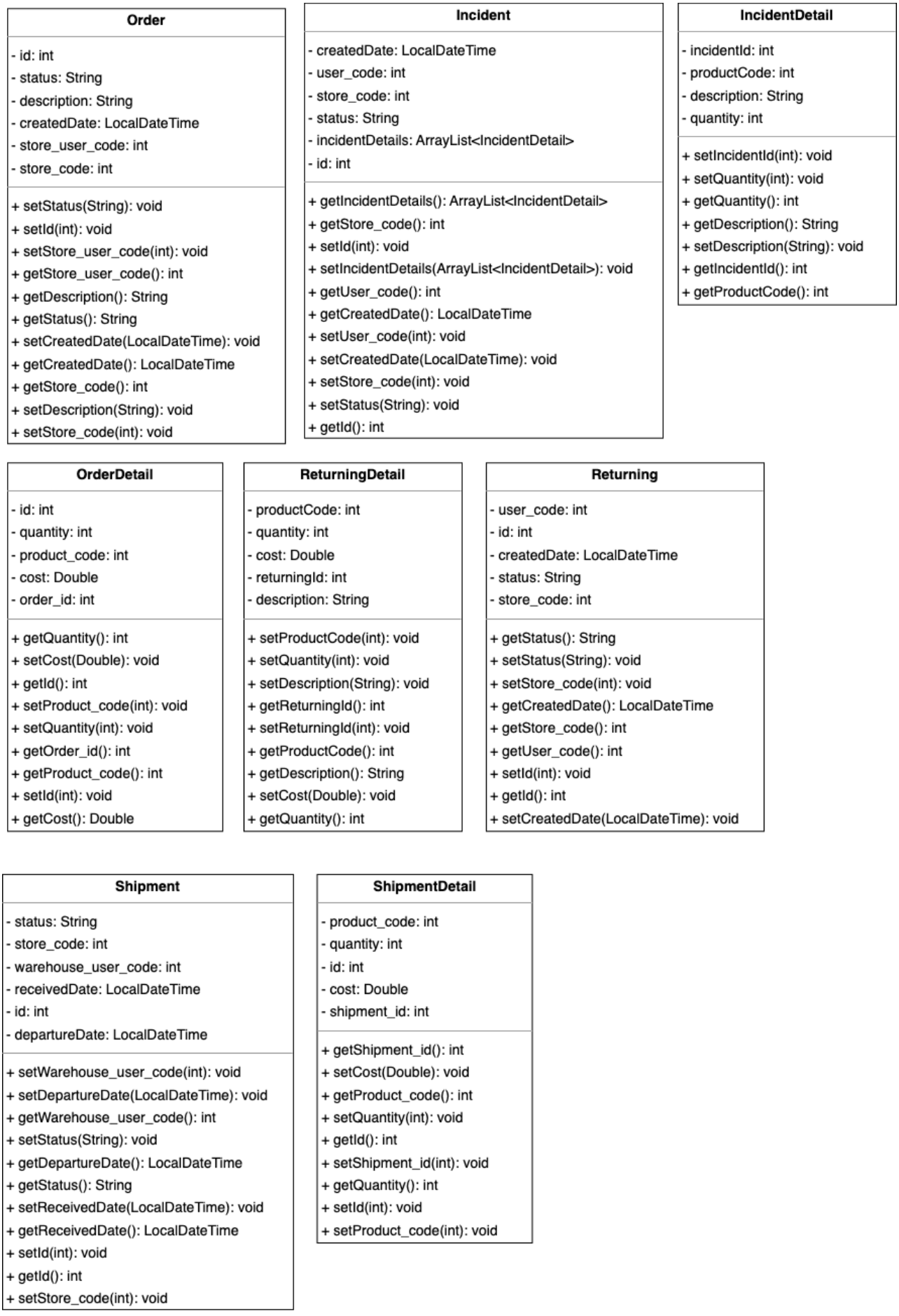
DIAGRAMA DE CLASES

Parte: Conexiones a la base de datos



Parte: Modelos




Order
- id: int
- status: String
- description: String
- createdDate: LocalDateTime
- store_user_code: int
- store_code: int
+ setStatus(String): void
+ setId(int): void
+ setStore_user_code(int): void
+ getStore_user_code(): int
+ getDescription(): String
+ getStatus(): String
+ setCreatedDate(LocalDateTime): void
+ getCreatedDate(): LocalDateTime
+ getStore_code(): int
+ setDescription(String): void
+ setStore_code(int): void
Incident
- createdDate: LocalDateTime
- user_code: int
- store_code: int
- status: String
- incidentDetails: ArrayList<IncidentDetail>
- id: int
+ getIncidentDetails(): ArrayList<IncidentDetail>
+ getStore_code(): int
+ setId(int): void
+ setIncidentDetails(ArrayList<IncidentDetail>): void
+ getUser_code(): int
+ getCreatedDate(): LocalDateTime
+ setUser_code(int): void
+ setCreatedDate(LocalDateTime): void
+ setStore_code(int): void
+ setStatus(String): void
+ getId(): int
IncidentDetail
- incidentId: int
- productCode: int
- description: String
- quantity: int
+ setIncidentId(int): void
+ setQuantity(int): void
+ getQuantity(): int
+ getDescription(): String
+ setDescription(String): void
+ getIncidentId(): int
+ getProductCode(): int
OrderDetail
- id: int
- quantity: int
- product_code: int
- cost: Double
- order_id: int
+ getQuantity(): int
+ setCost(Double): void
+ getId(): int
+ setProduct_code(int): void
+ setQuantity(int): void
+ getOrder_id(): int
+ getProduct_code(): int
+ setId(int): void
+ getCost(): Double
ReturningDetail
- productCode: int
- quantity: int
- cost: Double
- returningId: int
- description: String
+ setProductCode(int): void
+ setQuantity(int): void
+ setDescription(String): void
+ getReturningId(): int
+ setReturningId(int): void
+ getProductCode(): int
+ getDescription(): String
+ setCost(Double): void
+ getQuantity(): int
Returning
- user_code: int
- id: int
- createdDate: LocalDateTime
- status: String
- store_code: int
+ getStatus(): String
+ setStatus(String): void
+ setStore_code(int): void
+ getCreatedDate(): LocalDateTime
+ getStore_code(): int
+ getUser_code(): int
+ setId(int): void
+ getId(): int
+ setCreatedDate(LocalDateTime): void
Shipment
- status: String
- store_code: int
- warehouse_user_code: int
- receivedDate: LocalDateTime
- id: int
- departureDate: LocalDateTime
+ setWarehouse_user_code(int): void
+ setDepartureDate(LocalDateTime): void
+ getWarehouse_user_code(): int
+ setStatus(String): void
+ getDepartureDate(): LocalDateTime
+ getStatus(): String
+ setReceivedDate(LocalDateTime): void
+ getReceivedDate(): LocalDateTime
+ setId(int): void
+ getId(): int
+ setStore_code(int): void
ShipmentDetail
- product_code: int
- quantity: int
- id: int
- cost: Double
- shipment_id: int
+ getShipment_id(): int
+ setCost(Double): void
+ getProduct_code(): int
+ setQuantity(int): void
+ getId(): int
+ setShipment_id(int): void
+ getQuantity(): int
+ setId(int): void
+ setProduct_code(int): void

Parte: Servlets

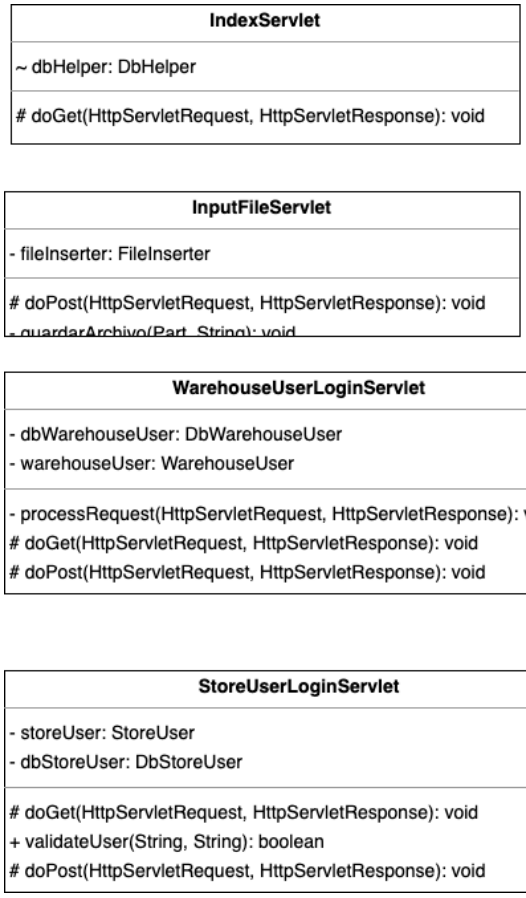


DIAGRAMA ENTIDAD RELACIÓN

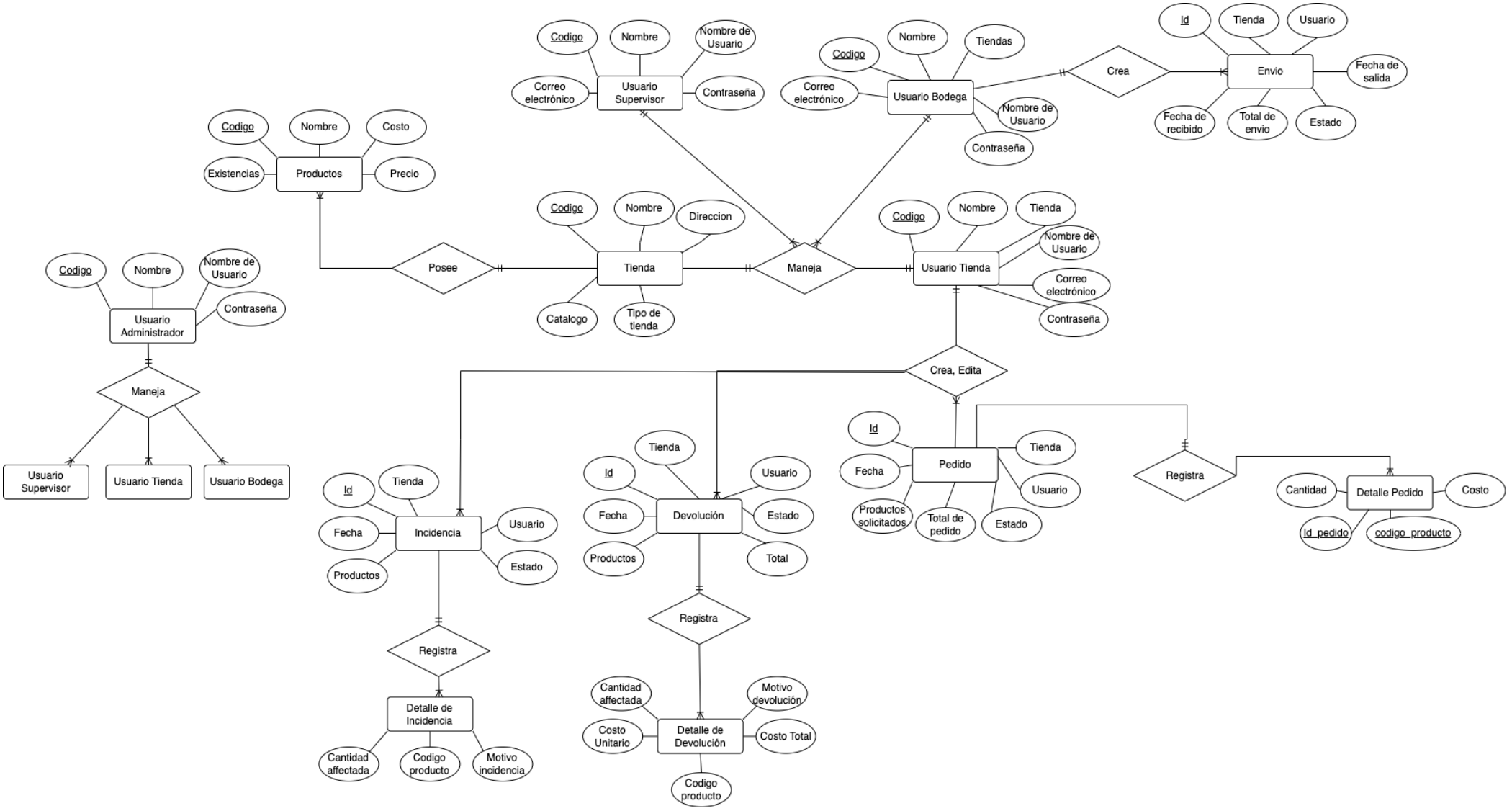
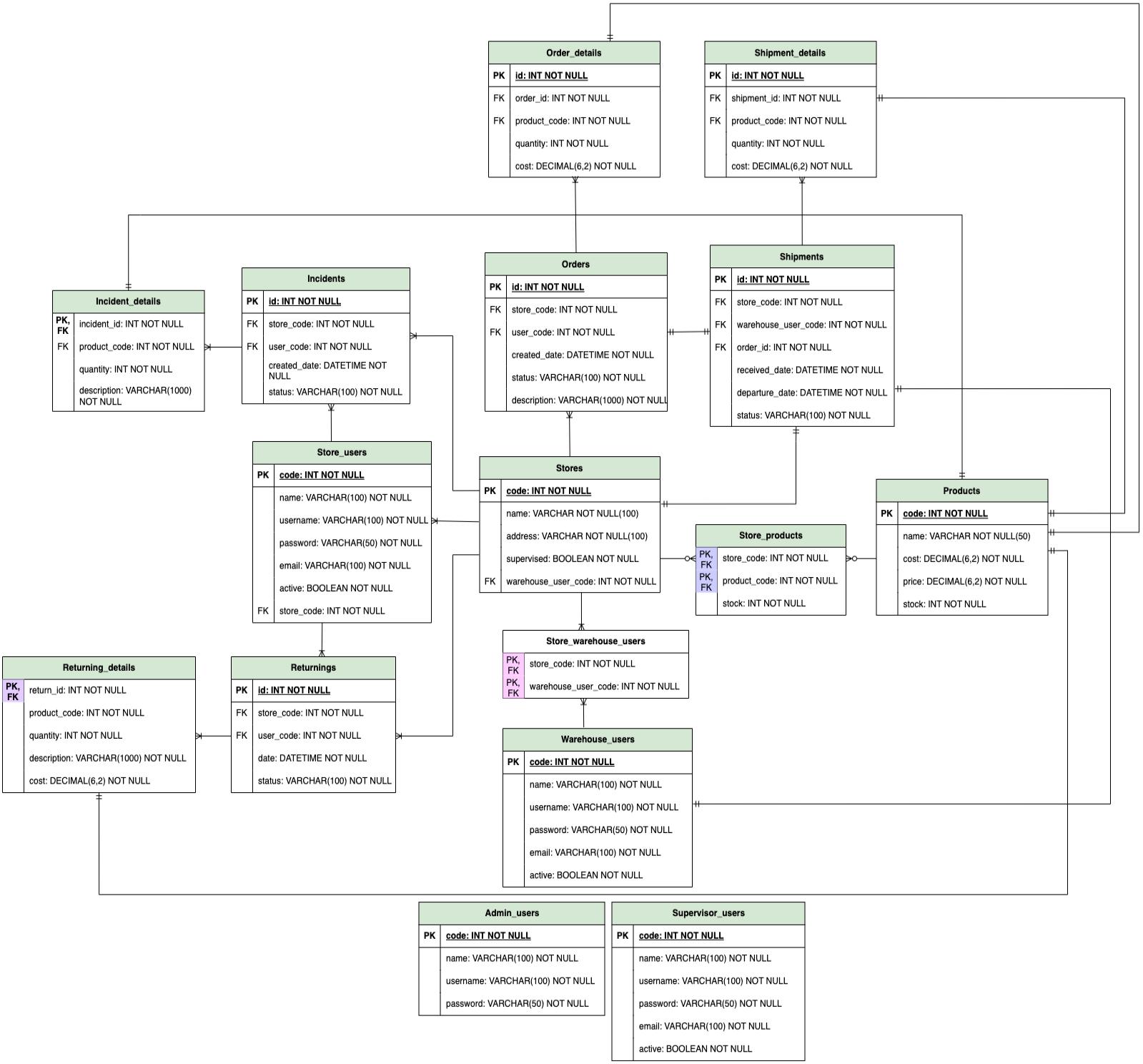


DIAGRAMA DE TABLAS



MAPEO FÍSICO DE LA BASE DE DATOS

```
// database.sql
DROP DATABASE IF EXISTS warehouse;
CREATE DATABASE warehouse;
USE warehouse;

CREATE TABLE warehouse_users (
 code INT NOT NULL,
 name VARCHAR(100) NOT NULL,
 username VARCHAR(100) NOT NULL,
 password VARCHAR(50) NOT NULL,
 email VARCHAR(100) NOT NULL,
 active BOOLEAN NOT NULL DEFAULT TRUE,
 PRIMARY KEY (code)
);

CREATE TABLE admin_users (
 code INT NOT NULL,
 name VARCHAR(100) NOT NULL,
 username VARCHAR(100) NOT NULL,
 password VARCHAR(50) NOT NULL,
 PRIMARY KEY (code)
);

CREATE TABLE supervisor_users (
 code INT NOT NULL,
 name VARCHAR(100) NOT NULL,
 username VARCHAR(100) NOT NULL,
 password VARCHAR(50) NOT NULL,
 email VARCHAR(100) NOT NULL,
 active BOOLEAN NOT NULL DEFAULT TRUE,
 PRIMARY KEY (code)
);

CREATE TABLE products (
 code INT NOT NULL,
 name VARCHAR(100) NOT NULL,
 cost DECIMAL(6,2) NOT NULL,
 price DECIMAL(6,2) NOT NULL,
 stock INT NOT NULL,
 PRIMARY KEY (code)
);
```

```
CREATE TABLE stores (
 code INT NOT NULL,
 name VARCHAR(100) NOT NULL,
 address VARCHAR(200) NOT NULL,
 supervised BOOLEAN NOT NULL,
 PRIMARY KEY (code)
);

CREATE TABLE store_warehouse_users (
   warehouse_user_code INT NOT NULL,
   store_code INT NOT NULL,
   PRIMARY KEY (warehouse_user_code, store_code),
   FOREIGN KEY (warehouse_user_code) REFERENCES warehouse_users(code),
   FOREIGN KEY (store_code) REFERENCES stores(code)
);

CREATE TABLE store_users (
 code INT NOT NULL,
 name VARCHAR(100) NOT NULL,
 username VARCHAR(100) NOT NULL,
 password VARCHAR(50) NOT NULL,
 email VARCHAR(100) NOT NULL,
 active BOOLEAN NOT NULL DEFAULT TRUE,
 store_code INT NOT NULL,
 PRIMARY KEY (code),
 FOREIGN KEY (store_code) REFERENCES stores (code)
);

CREATE TABLE orders (
 id INT AUTO_INCREMENT NOT NULL,
 created_date DATETIME not null DEFAULT CURRENT_TIMESTAMP,
 status VARCHAR(100) NOT NULL,
 description VARCHAR(1000),
 store_code INT NOT NULL,
 store_user_code INT NOT NULL,
 PRIMARY KEY (id),
 FOREIGN KEY (store_code) REFERENCES stores (code),
 FOREIGN KEY (store_user_code) REFERENCES store_users (code)
);
```

```
CREATE TABLE shipments(
id INT AUTO_INCREMENT NOT NULL,
store_code INT NOT NULL,
warehouse_user_code INT NOT NULL,
order_id INT NOT NULL,
departure_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
received_date DATETIME,
status VARCHAR(100) NOT NULL DEFAULT 'DESPACHADO',
PRIMARY KEY (id),
FOREIGN KEY (store_code) REFERENCES stores (code),
FOREIGN KEY (warehouse_user_code) REFERENCES warehouse_users (code),
FOREIGN KEY (order_id) REFERENCES orders (id)
);

CREATE TABLE shipment_details(
id INT AUTO_INCREMENT NOT NULL,
shipment_id INT NOT NULL,
product_code INT NOT NULL,
quantity INT NOT NULL,
cost DECIMAL(6,2) NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY (shipment_id) REFERENCES shipments (id),
FOREIGN KEY (product_code) REFERENCES products (code)
);

CREATE TABLE order_details (
   id INT AUTO_INCREMENT NOT NULL,
   order_id INT NOT NULL,
   product_code INT NOT NULL,
   quantity INT NOT NULL,
   cost DECIMAL(6,2) NOT NULL,
   PRIMARY KEY (id),
   FOREIGN KEY (order_id) REFERENCES orders (id),
   FOREIGN KEY (product_code) REFERENCES products (code)
);
```

```
CREATE TABLE incidents (
 id INT AUTO_INCREMENT NOT NULL,
 created_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
 status VARCHAR(100) NOT NULL DEFAULT 'ACTIVA',
 store_code INT NOT NULL,
 user_code INT NOT NULL,
 solution VARCHAR(1000),
 shipment_id INT NOT NULL,
 PRIMARY KEY (id),
 FOREIGN KEY (store_code) REFERENCES stores (code),
 FOREIGN KEY (user_code) REFERENCES store_users (code),
 FOREIGN KEY (shipment_id) REFERENCES shipments (id)
);

CREATE TABLE incident_details(
 incident_id INT NOT NULL,
 product_code INT NOT NULL,
 quantity INT NOT NULL,
 description VARCHAR(1000),
 PRIMARY KEY (incident_id),
 FOREIGN KEY (incident_id) REFERENCES incidents(id),
 FOREIGN KEY (product_code) REFERENCES products(code)
);

CREATE TABLE returnings (
 id INT AUTO_INCREMENT NOT NULL,
 created_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
 status VARCHAR(100) NOT NULL,
 store_code INT NOT NULL,
 user_code INT NOT NULL,
 shipment_id INT NOT NULL,
 PRIMARY KEY (id),
 FOREIGN KEY (store_code) REFERENCES stores (code),
 FOREIGN KEY (user_code) REFERENCES store_users (code),
 FOREIGN KEY (shipment_id) REFERENCES shipments (id)
);
```

```
CREATE TABLE returnings_details(
 returning_id INT NOT NULL,
 product_code INT NOT NULL,
 quantity INT NOT NULL,
 description VARCHAR(1000),
 shipment_id INT NOT NULL,
 cost DECIMAL(6,2) NOT NULL,
 PRIMARY KEY (returning_id),
 FOREIGN KEY (returning_id) REFERENCES returnings(id),
 FOREIGN KEY (product_code) REFERENCES products(code),
 FOREIGN KEY (shipment_id) REFERENCES shipments (id)
);

CREATE TABLE store_products(
 store_code INT NOT NULL,
 product_code INT NOT NULL,
 stock INT NOT NULL,
 PRIMARY KEY (store_code, product_code),
 FOREIGN KEY (store_code) REFERENCES stores (code),
 FOREIGN KEY (product_code) REFERENCES products (code)
);
```