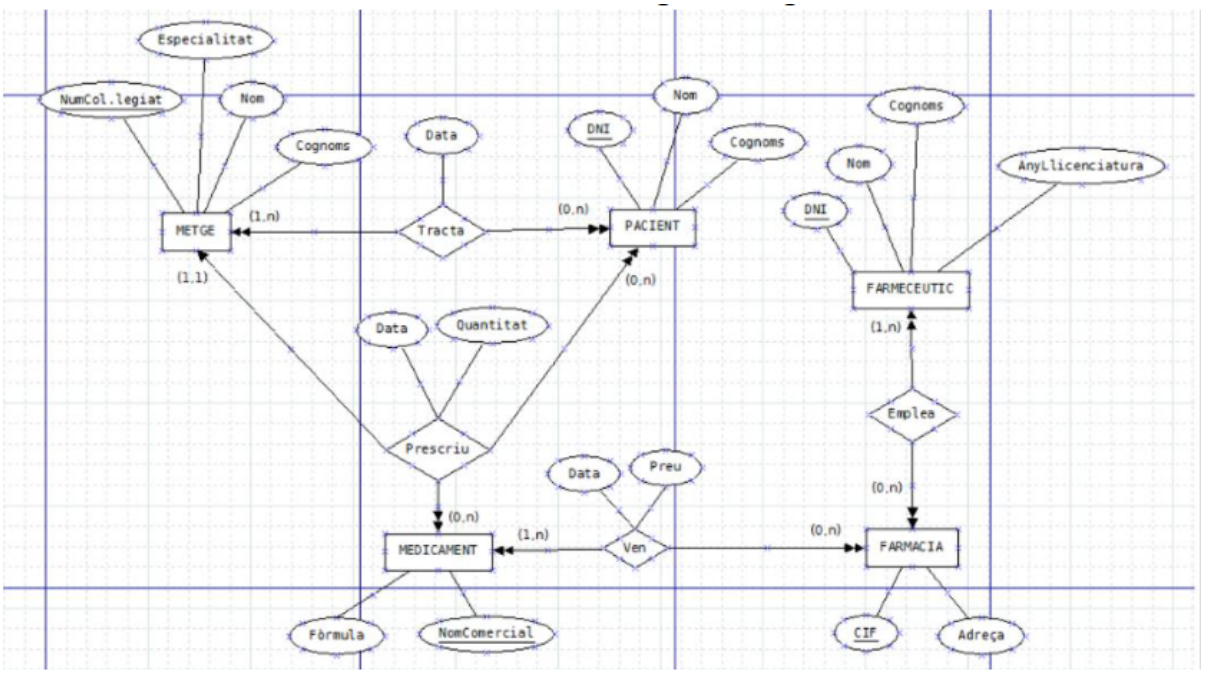
Nom i cognoms: Albert Lozano Blasco. **Curs i mòdul:** 2n DAM - ADA.

Práctica 1. JDBC.

CRUD i JDBC. Treballarem amb la BD FARMACIA.

Aquest és el model Entitat-Relació de la base de dades:



1. Dedica el temps necessari a entendre el diagrama i després, crea les taules ardients amb l'eina de Model de MySQLWorkbench. FInalment, genera l'scrpit de creació de la base de dades.

Per a entendre el diagrama ER de la base de dades, he decidit, de primer, fer un diagrama propi en la aplicació web *draw.io* amb un llenguatge *pseudoSQL*.

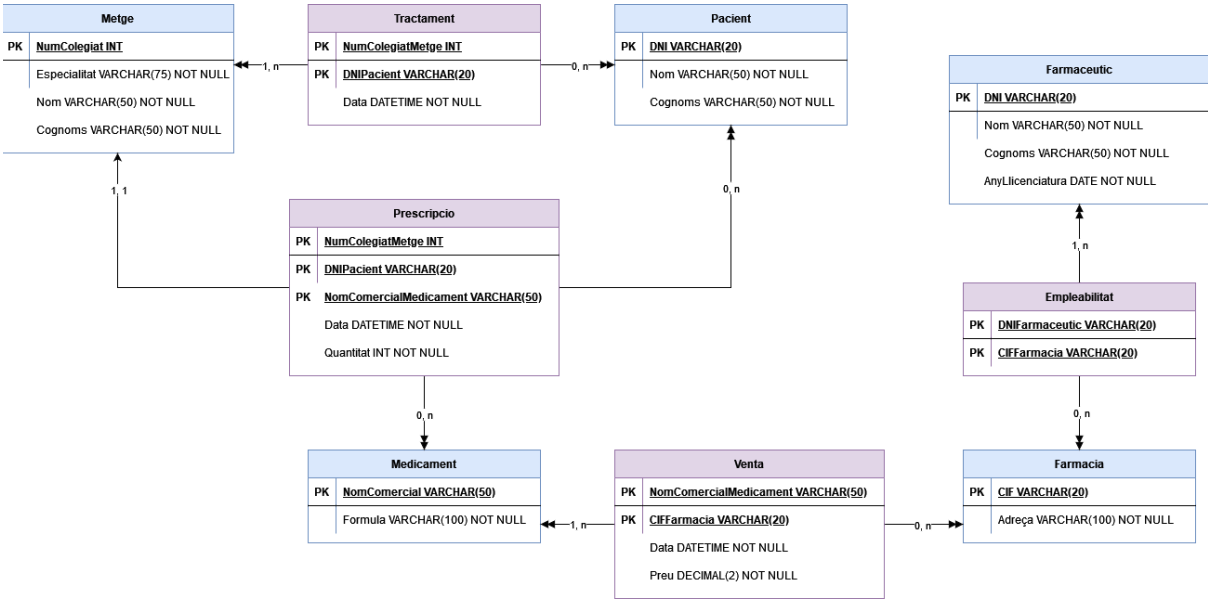
Una vegada dissenyat aquest diagrama, he iniciat *MySQLWorkbench*, he creat l'esquema de la base de dades "Farmacia" i he creat les diferents taules per a les entitats. També he creat una taula per a cada relació, degut a les característiques d'aquestes.

Després de crear les taules, he definit les relacions assignant les foreign keys necessàries, mitjançant codi *SQL*. Aconseguint les interaccions desitjades, també indicant la opció *Cascade* per als Updates i Deletes.

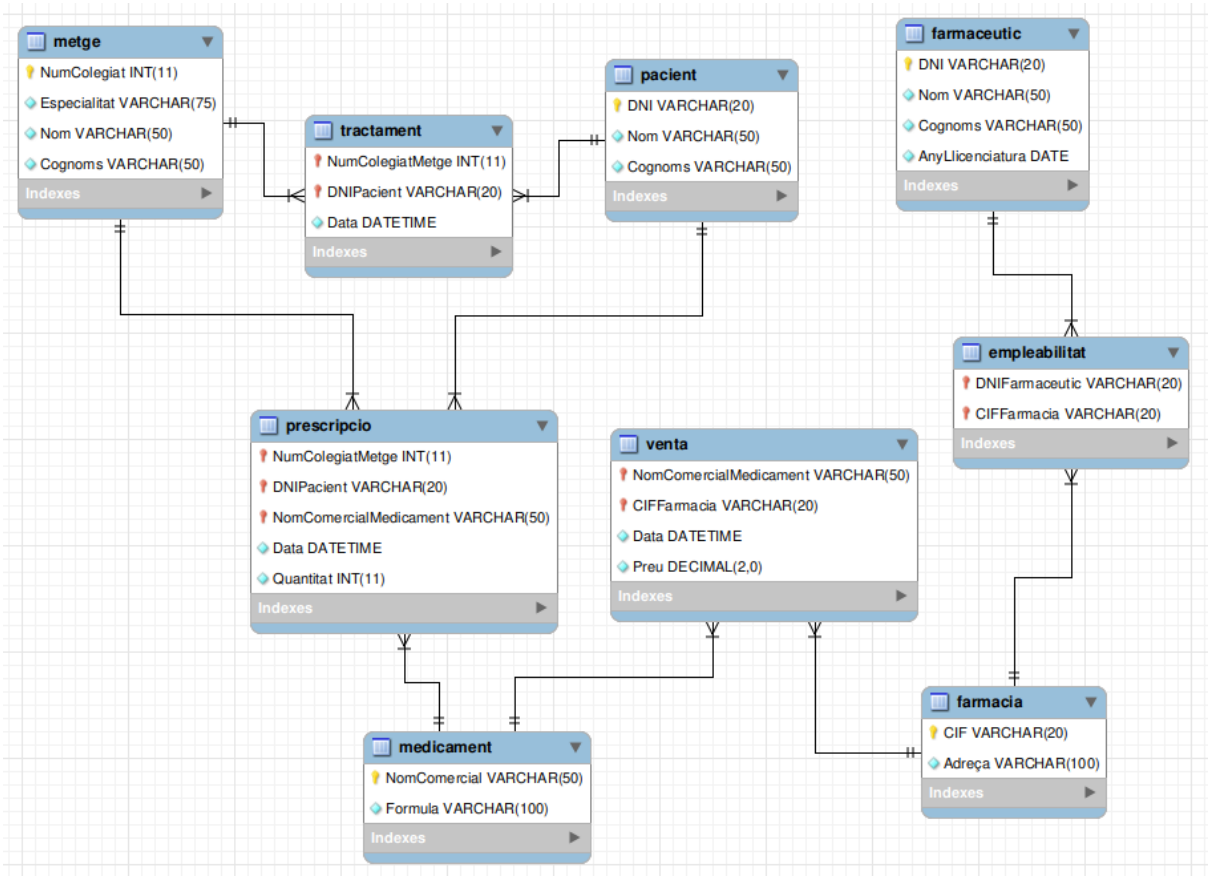
Finalment, fent ús de les opcions *Reverse Engineering* i *Forward Engineering*, he generat el diagrama de la base de dades i l'script de la seua creació.

A continuació; inserte una imatge de cada pas:

1. Diagrama *pseudoSQL* realitzat en l'aplicació web *draw.io*:



2. Diagrama de la base de dades en *MySQLWorkbench*, una vegada finalitzada la creació, aconseguida fent ús de l'opció *Reverse Engineering*:



3. Script de la creació de la base de dades, generat amb l'opció *Forward Engineering*:

DADES INCIALS: He inclòs sentències d'INSERT al final per a tindre dades inicials amb les que poder treballar des d'un principi.

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,N
O_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema farmacia
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema farmacia
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `farmacia` DEFAULT CHARACTER SET utf8mb4 ;
USE `farmacia` ;

-- -----------------------------------------------------
-- Table `farmacia`.`farmaceutic`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`farmaceutic` (
  `DNI` VARCHAR(20) NOT NULL,
  `Nom` VARCHAR(50) NOT NULL,
  `Cognoms` VARCHAR(50) NOT NULL,
  `AnyLlicenciatura` DATE NOT NULL,
  PRIMARY KEY (`DNI`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`farmacia`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`farmacia` (
  `CIF` VARCHAR(20) NOT NULL,
  `Adreça` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`CIF`))
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`empleabilitat`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`empleabilitat` (
  `DNIFarmaceutic` VARCHAR(20) NOT NULL,
  `CIFFarmacia` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`DNIFarmaceutic`, `CIFFarmacia`),
  INDEX `CIFFarmacia` (`CIFFarmacia` ASC) VISIBLE,
  CONSTRAINT `empleabilitat_ibfk_1`
          FOREIGN KEY (`DNIFarmaceutic`)
          REFERENCES `farmacia`.`farmaceutic` (`DNI`),
  CONSTRAINT `empleabilitat_ibfk_2`
          FOREIGN KEY (`CIFFarmacia`)
          REFERENCES `farmacia`.`farmacia` (`CIF`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`medicament`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`medicament` (
  `NomComercial` VARCHAR(50) NOT NULL,
  `Formula` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`NomComercial`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`metge`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`metge` (
  `NumColegiat` INT(11) NOT NULL,
  `Especialitat` VARCHAR(75) NOT NULL,
  `Nom` VARCHAR(50) NOT NULL,
  `Cognoms` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`NumColegiat`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`pacient`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `farmacia`.`pacient` (
  `DNI` VARCHAR(20) NOT NULL,
  `Nom` VARCHAR(50) NOT NULL,
  `Cognoms` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`DNI`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`prescripcio`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`prescripcio` (
  `NumColegiatMetge` INT(11) NOT NULL,
  `DNIPacient` VARCHAR(20) NOT NULL,
  `NomComercialMedicament` VARCHAR(50) NOT NULL,
  `Data` DATETIME NOT NULL,
  `Quantitat` INT(11) NOT NULL,
  PRIMARY KEY (`NumColegiatMetge`, `DNIPacient`, `NomComercialMedicament`),
  INDEX `DNIPacient` (`DNIPacient` ASC) VISIBLE,
  INDEX `NomComercialMedicament` (`NomComercialMedicament` ASC) VISIBLE,
  CONSTRAINT `prescripcio_ibfk_1`
          FOREIGN KEY (`NumColegiatMetge`)
          REFERENCES `farmacia`.`metge` (`NumColegiat`),
  CONSTRAINT `prescripcio_ibfk_2`
          FOREIGN KEY (`DNIPacient`)
          REFERENCES `farmacia`.`pacient` (`DNI`),
  CONSTRAINT `prescripcio_ibfk_3`
          FOREIGN KEY (`NomComercialMedicament`)
          REFERENCES `farmacia`.`medicament` (`NomComercial`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`tractament`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`tractament` (
  `NumColegiatMetge` INT(11) NOT NULL,
  `DNIPacient` VARCHAR(20) NOT NULL,
  `Data` DATETIME NOT NULL,
  PRIMARY KEY (`NumColegiatMetge`, `DNIPacient`),
  INDEX `DNIPacient` (`DNIPacient` ASC) VISIBLE,
  CONSTRAINT `tractament_ibfk_1`
          FOREIGN KEY (`NumColegiatMetge`)
          REFERENCES `farmacia`.`metge` (`NumColegiat`),
  CONSTRAINT `tractament_ibfk_2`
          FOREIGN KEY (`DNIPacient`)
```

```
        REFERENCES `farmacia`.`pacient` (`DNI`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `farmacia`.`venta`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `farmacia`.`venta` (
  `NomComercialMedicament` VARCHAR(50) NOT NULL,
  `CIFFarmacia` VARCHAR(20) NOT NULL,
  `Data` DATETIME NOT NULL,
  `Preu` DECIMAL(4,2) NOT NULL,
  PRIMARY KEY (`NomComercialMedicament`, `CIFFarmacia`),
  INDEX `CIFFarmacia` (`CIFFarmacia` ASC) VISIBLE,
  CONSTRAINT `venta_ibfk_1`
        FOREIGN KEY (`NomComercialMedicament`)
        REFERENCES `farmacia`.`medicament` (`NomComercial`),
  CONSTRAINT `venta_ibfk_2`
        FOREIGN KEY (`CIFFarmacia`)
        REFERENCES `farmacia`.`farmacia` (`CIF`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

INSERT INTO `farmacia`.`metge` (`NumColegiat`, `Especialitat`, `Nom`, `Cognoms`)
VALUES
  (1, 'Cardiología', 'Juan', 'Gómez'),
  (2, 'Pediatría', 'María', 'López'),
  (3, 'Dermatología', 'Pedro', 'Martínez'),
  (4, 'Ginecología', 'Luisa', 'Fernández'),
  (5, 'Neurología', 'Carlos', 'Rodríguez');

INSERT INTO `farmacia`.`pacient` (`DNI`, `Nom`, `Cognoms`)
VALUES
  ('12345678A', 'Laura', 'González Pérez'),
  ('98765432B', 'Manuel', 'Martínez Ruiz'),
  ('56789012C', 'Ana', 'López García'),
  ('34567890D', 'Carlos', 'Fernández Sánchez'),
  ('87654321E', 'Isabel', 'Rodríguez Díaz');
```

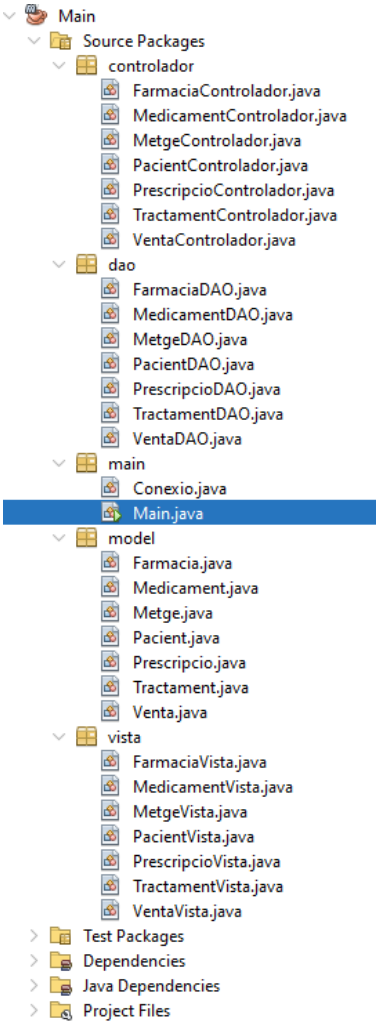
```
INSERT INTO `farmacia`.`medicament` (`NomComercial`, `Formula`)
VALUES
  ('Aspirina', 'Ácido acetilsalicílico'),
  ('Paracetamol', 'Paracetamol'),
  ('Ibuprofeno', 'Ibuprofeno'),
  ('Amoxicilina', 'Amoxicilina'),
  ('Omeprazol', 'Omeprazol');

INSERT INTO `farmacia`.`farmacia` (`CIF`, `Adreça`)
VALUES
  ('A1234567', 'Calle Principal 123'),
  ('B9876543', 'Avenida Central 456'),
  ('C5678901', 'Plaza del Pueblo 789'),
  ('D3456789', 'Carrera Grande 567'),
  ('E8765432', 'Paseo Bonito 234');

INSERT INTO `farmacia`.`tractament` (`NumColegiatMetge`, `DNIPacient`, `Data`)
VALUES
  (1, '12345678A', '2023-10-10 08:00:00'),
  (2, '98765432B', '2023-10-11 09:15:00'),
  (3, '56789012C', '2023-10-12 10:30:00'),
  (4, '34567890D', '2023-10-13 11:45:00'),
  (5, '87654321E', '2023-10-14 13:00:00');

INSERT INTO `farmacia`.`prescripcio` (`NumColegiatMetge`, `DNIPacient`,
`NomComercialMedicament`, `Data`, `Quantitat`)
VALUES
  (1, '12345678A', 'Aspirina', '2023-10-10 08:00:00', 1),
  (2, '98765432B', 'Paracetamol', '2023-10-11 09:15:00', 2),
  (3, '56789012C', 'Ibuprofeno', '2023-10-12 10:30:00', 1),
  (4, '34567890D', 'Amoxicilina', '2023-10-13 11:45:00', 3),
  (5, '87654321E', 'Omeprazol', '2023-10-14 13:00:00', 1);

INSERT INTO `farmacia`.`venta` (`NomComercialMedicament`, `CIFFarmacia`, `Data`,
`Preu`)
VALUES
  ('Aspirina', 'A1234567', '2023-10-10 08:00:00', 5.25),
  ('Paracetamol', 'B9876543', '2023-10-11 09:15:00', 3.99),
  ('Ibuprofeno', 'C5678901', '2023-10-12 10:30:00', 4.95),
  ('Amoxicilina', 'D3456789', '2023-10-13 11:45:00', 8.20),
  ('Omeprazol', 'E8765432', '2023-10-14 13:00:00', 6.10);
```

2. Crear les classes necessàries per a l'implementació de les sentències SELECT, INSERT, DELETE Y UPDATE de METGE, PACIENT, FARMACIA i MEDICAMENT.

Per a poder realitzar l'implementació de les sentències SELECT, INSERT, DELETE y UPDATE i poder tindre un sistema *CRUD*, deurem crear les següents classes en *Java*: Metge, Pacient, Farmacia i Medicament.

Vaig a treballar amb el patró de disseny *MVC* (Model-Vista-Controlador), per a tindre una arquitectura sòlida i correcta, separar les responsabilitats i facilitar el manteniment i l'escalabilitat.

Com a idea principal, el projecte en *NetBeans* tindrà els següents paquets:

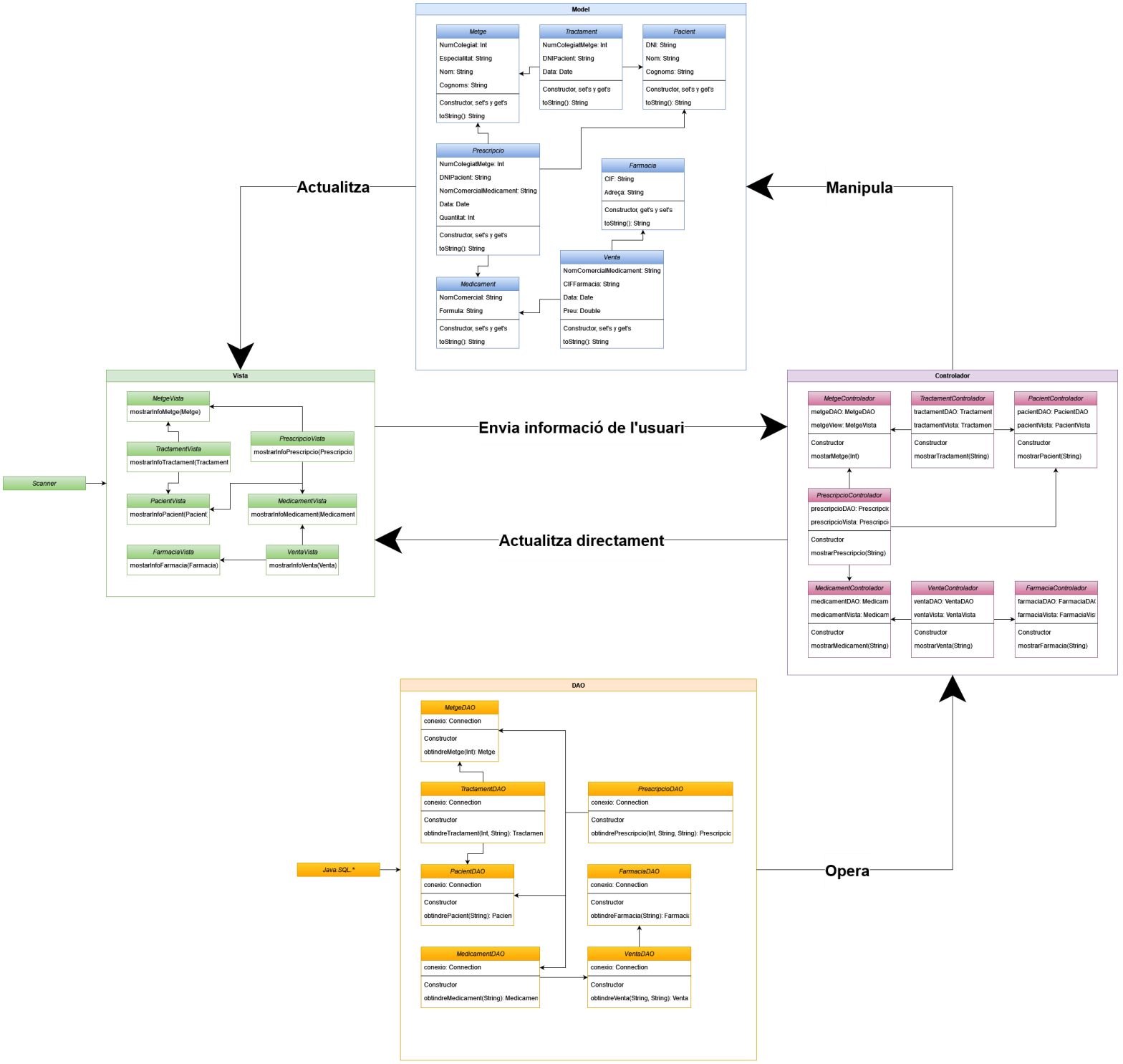


model: Inclourà les classes que representen el model de dades.
vista: interfaç d'usuari, o, en aquest cas, vista en consola.
controlador: S'encarregarà de comunicar amb la base de dades. Cada classe "entitat" podria tindre la seua pròpia classe "controladora".
dao: (*Data Acces Object*) inclourà les classes que s'encarreguen de la comunicació directa amb la base de dades.
main: ací estarà implementada la funció *main*, per a executar l'aplicació i la conexió a la base de dades.

Primerament, he decidit crear/dissenyar un diagrama de classes, per a planificar, optimitzar, comunicar i documentar la creació de l'aplicació orientada a objectes.

Finalment, he implementat el codi en *NetBeans*.

A continuació, inserte una captura del diagrama de classes que he dissenyat en *draw.io*:



3. Implementació de les següents funcionalitats:

Gràcies a la sòlida base del projecte i a l'arquitectura MVC, podem escalar les funcionalitats de l'aplicació més fàcilment. Cridarem a aquestes funcionalitats des de la classe *Main*.

a. Mètode que, donat el número de col·legiat d'un metge, lliste a tots els pacients que tracta i amb la data que els ha tractat.

He implementat aquest mètode en, primer *TractamentDAO*, i després en *TractamentControlador*. D'aquesta forma, treballe per capes en el model MVC.

Captura del codi (Classe DAO):

```
//Mètode que, donat el número de col·legiat d'un metge, lliste a tots els pacients que tracta i amb la data que els ha tractat.
public ArrayList<Tractament> llistarPacientsPerMetge(int numColegiat) {
    ArrayList<Tractament> pacientsTractats = new ArrayList<>();

    try {
        String consulta = "SELECT DNIPacient, Data FROM tractament WHERE NumColegiatMetge = ?";
        PreparedStatement ps = conexio.prepareStatement(sql:consulta);
        ps.setInt(parameterIndex: 1, x: numColegiat);

        ResultSet rs = ps.executeQuery();

        while (rs.next()) {
            String dniPacient = rs.getString(columnLabel:"DNIPacient");
            Date dataTractament = rs.getTimestamp(columnLabel:"Data");
            pacientsTractats.add(new Tractament(numColegiatMetge: numColegiat, dniPacient, data: dataTractament));
        }
    } catch (SQLException e) {
        System.out.println(x: "ERROR! No s'han pogut obtenir les dades dels pacients tractats pel metge.");
    }

    return pacientsTractats;
}
```

Captura del codi (Classe Controlador):

```
//Mètode que, donat el número de col·legiat d'un metge, lliste a tots els pacients que tracta i amb la data que els ha tractat.
public void llistarPacientsPerMetgeBaseDades(int numColegiat) {
    tractamentVista.mostrarInformacioTractaments(tractaments:tractamentDAO.llistarPacientsPerMetge(numColegiat));
}
```

b. Codifica l'operació d'inserir la prescripció d'un medicament per un pacient determinat.

He implementat aquesta operació en, primer *PrescripcioDAO*, i després en *PrescripcioControlador*. D'aquesta forma, treballe per capes en el model MVC.

Captura del codi (Classe DAO):

```
//Mètode per a insertar una prescripció d'un medicament per a un pacient determinat.
public void insertarPrescripcio(int numColegiatMetge, String dniPacient, String nomComercialMedicament, Date data, int quantitat) {
    if (!existeixPrescripcio(numColegiatMetge, dniPacient, nomComercialMedicament)) {
        try {
            String consulta = "INSERT INTO prescripcio (NumColegiatMetge, DNIPacient, NomComercialMedicament, Data, Quantitat) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement ps = conexio.prepareStatement(sql:consulta);

            ps.setInt(parameterIndex: 1, x: numColegiatMetge);
            ps.setString(parameterIndex: 2, x: dniPacient);
            ps.setString(parameterIndex: 3, x: nomComercialMedicament);
            ps.setTimestamp(parameterIndex: 4, new Timestamp(time: data.getTime()));
            ps.setInt(parameterIndex: 5, x: quantitat);
            ps.executeUpdate();

            System.out.println(x: "");
            System.out.println(x: "S'ha insertat la prescripció amb èxit.");
        } catch (SQLException e) {
            System.out.println(x: "ERROR! No s'ha pogut insertar la prescripció.");
        }
    } else {
        System.out.println(x: "Ja existeix una prescripció del mateix medicament, realitzada pel mateix metge, al mateix pacient");
        System.out.println(x: "En esta farmacia no es permet que un mateix metge realitze la prescripció d'un medicament a un mateix pacient més d'una vegada.");
    }

}
```

Captura del codi (Classe Controlador):

```java
//Mètode per a insertar una prescripció a la base de dades.
public void insertarPrescripcioBaseDades(int numColegiatMetge, String dniPacient, String nomComercialMedicament, Date data, int quantitat) {
    prescripcioDAO.insertarPrescripcio(numColegiatMetge, dniPacient, nomComercialMedicament, data, quantitat);
}
```

c. Operació que esborra tots els medicaments venuts per una farmàcia determinada.

He implementat aquesta operació en, primer *VentaDAO*, i després en *VentaControlador*. D'aquesta forma, treballe per capes en el model MVC.

Captura del codi (Classe DAO):

```java
//Mètode per a eliminar tots els medicaments venuts per auna farmàcia determinada.
public void eliminarVentesPerFarmacia(String cifFarmacia) {
    try {
        String consulta = "DELETE FROM venta WHERE CIFFarmacia LIKE ?";
        PreparedStatement ps = conexio.prepareStatement(sql:consulta);
        ps.setString(parameterIndex: 1, x: cifFarmacia);

        int filasAfectades = ps.executeUpdate();

        if (filasAfectades > 0) {
            System.out.println("Eliminat/s " + filasAfectades + " registre/s de venta per a la farmàcia amb CIF: " + cifFarmacia);
        } else {
            System.out.println("No s'han trobat registres de venta per a la farmàcia amb CIF: " + cifFarmacia);
        }
    } catch (SQLException e) {
        System.out.println(x: "ERROR! No s'han pogut eliminar els registres de venda.");
    }
}
```

Captura del codi (Classe Controlador):

```java
//Mètode per a eliminar les ventes de medicaments de una determinada farmacia.
public void eliminarVentesPerFarmacia(String cifFarmacia) {
    ventaDAO.eliminarVentesPerFarmacia(cifFarmacia);
}
```