UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 2
CATEDRÁTICO: ING. DANIEL GONZALES

MANUAL TÉCNICO

ELEAZAR NEFTALÍ COLOP COLOP
CARNÉ: 202131418
SECCIÓN: A

GUATEMALA, 12 DE MARZO DE 2025

ÍNDICE

INTRODUCCIÓN

Este manual está diseñado para guiar al usuario y al desarrollador en la comprensión y uso de la aplicación desarrollada con tecnología Java, utilizando NetBeans como entorno de desarrollo, Tomcat como servidor web, MySQL como base de datos, y JDK 21. El propósito de este manual es proporcionar instrucciones claras para que tanto los programadores como los usuarios finales puedan utilizar y modificar la aplicación de manera efectiva.

OBJETIVOS

GENERAL

El objetivo general de este manual es ofrecer una guía completa para la implementación, uso y mantenimiento de la aplicación desarrollada, asegurando que los usuarios y los desarrolladores puedan trabajar eficientemente con la infraestructura técnica utilizada.

ESPECÍFICOS

Objetivo 1: Describir los pasos técnicos que se siguen para configurar el entorno de desarrollo, incluyendo la instalación de Tomcat, MySQL, NetBeans, JDK 21, y la integración entre estos componentes.

Objetivo 2: Explicar el funcionamiento y la estructura del código, detallando las clases, funciones y procedimientos que componen la aplicación.

ALCANCES DEL SISTEMA

Este manual está enfocado en la configuración e implementación de un sistema que se conecta a una base de datos MySQL para gestionar información. Proporciona detalles sobre la instalación y uso del servidor Apache Tomcat, la configuración de MySQL y la ejecución de la aplicación en NetBeans. El alcance está limitado a la configuración y funcionamiento del sistema en un entorno de desarrollo basado en Windows, con la versión de JDK 21, y a los componentes utilizados para la gestión de la base de datos y el servidor web.

ESPECIFICACIÓN TÉCNICA

● REQUISITOS DE HARDWARE

- **Procesador:** 2.0 GHz o superior.
- **Memoria RAM:** 8 GB o más recomendados para el desarrollo de aplicaciones.
- **Espacio en disco:** Al menos 10 GB de espacio libre para la instalación de MySQL, Tomcat y NetBeans.
- **Tarjeta gráfica:** No requerida, a menos que se implementen características visuales intensivas.

● REQUISITOS DE SOFTWARE

- **Sistema Operativo:** Windows 10 o superior.
- **JDK:** JDK 21 para compilar y ejecutar la aplicación Java.
- **Apache Tomcat:** Tomcat 10.1.35 para la implementación de la aplicación web.
- **MySQL Server:** MySQL 8.0 para gestionar la base de datos.
- **NetBeans IDE:** NetBeans 14 o superior para el desarrollo del código Java y gestión del proyecto.
- **Navegador Web:** Cualquier navegador web moderno (como Google Chrome, Mozilla Firefox, etc.) para acceder a la aplicación a través de la red local.

DESCRIPCIÓN DE LA SOLUCIÓN

La solución propuesta ha sido desarrollada utilizando Java con la finalidad de gestionar información de productos o componentes en un sistema de venta de computadoras. La base de datos se gestiona con MySQL, y el sistema web está alojado en Tomcat. La aplicación ha sido diseñada para ofrecer una interfaz administrativa sencilla para gestionar productos, ventas, devoluciones y usuarios. Se tomó en cuenta la escalabilidad y la seguridad en el diseño de la base de datos, así como la interacción eficiente entre el front-end y back-end.

LÓGICA DEL PROGRAMA

La lógica del programa en este proyecto está estructurada en varios componentes esenciales que se interrelacionan para ofrecer una solución funcional para la gestión de los modelos de computadoras en el sistema. En primer lugar, la comunicación con la base de datos es uno de los aspectos fundamentales, ya que la base de datos almacena y organiza toda la información necesaria sobre los modelos de computadoras, los usuarios, las ventas, y otros datos importantes. Para poder interactuar con la base de datos, se hace uso de la librería java.sql.*, que proporciona las herramientas necesarias para establecer la conexión con la base de datos, ejecutar consultas SQL y manejar posibles errores.

La clase Connection de java.sql es la principal responsable de establecer una conexión entre el programa y la base de datos. Mediante esta clase se puede crear una conexión con el servidor de MySQL que contiene la base de datos y, a partir de esta conexión, se pueden ejecutar diversas consultas como SELECT, INSERT, UPDATE, y DELETE, entre otras. Para realizar las consultas de manera más segura, se utiliza la clase PreparedStatement, la cual permite parametrizar las consultas SQL, evitando problemas de inyección SQL y garantizando la integridad de los datos. A través de esta clase se pueden pasar valores dinámicos a las consultas SQL, lo que facilita la ejecución de operaciones como la eliminación de un modelo de computadora de la base de datos. El uso de estas librerías SQL asegura que el sistema interactúe de manera eficiente con la base de datos y que cualquier error de base de datos sea manejado adecuadamente mediante la clase SQLException, la cual permite capturar y gestionar las excepciones que puedan surgir durante la ejecución de las consultas.

Un componente clave del sistema es el uso de Servlets, que permiten procesar las solicitudes del usuario y devolver respuestas apropiadas en función de estas solicitudes. Los Servlets se encargan de manejar las peticiones HTTP provenientes del navegador del usuario, procesarlas, interactuar con la base de datos si es necesario, y devolver una respuesta en forma de una página web (generalmente una página JSP) que será enviada de vuelta al cliente. Los Servlets se desarrollan utilizando la librería javax.servlet.*, que ofrece las clases esenciales para la creación de estos componentes web. La clase HttpServlet es la base para la creación de servlets que manejan peticiones HTTP, y a través de

métodos como doPost() y doGet(), los Servlets pueden recibir y responder a solicitudes enviadas por el cliente. Además, los métodos HttpServletRequest y HttpServletResponse se utilizan para obtener los parámetros de la solicitud del cliente y generar la respuesta correspondiente, respectivamente.

Los Servlets permiten realizar la lógica del negocio en el servidor, como la validación de datos, el procesamiento de formularios de usuario y la manipulación de los datos en la base de datos, todo ello sin exponer la lógica interna al cliente. Por ejemplo, cuando un usuario desea eliminar un modelo de computadora, se realiza una solicitud POST a un Servlet que manejará esta operación. El Servlet recibirá el nombre del modelo de computadora a eliminar a través de los parámetros de la solicitud, interactuará con la base de datos para eliminar el modelo y luego redirigirá al usuario a una página de confirmación o error según el resultado de la operación. Esta separación entre la lógica del negocio y la interfaz de usuario permite que el sistema sea más flexible y mantenible.

El uso de JavaServer Pages (JSP) es esencial para la parte de la interfaz de usuario del sistema. Las páginas JSP son archivos que contienen una mezcla de código Java y HTML, lo que permite generar contenido dinámico en función de las interacciones del usuario y los datos obtenidos del servidor. Los JSP permiten incrustar dinámicamente elementos como tablas, listas y formularios en las páginas web, y su capacidad para interactuar con los Servlets hace que sean una herramienta muy poderosa para la creación de aplicaciones web dinámicas. El motor de JSP se encarga de convertir el código JSP en un servlet de Java antes de ser ejecutado, lo que asegura que el código del lado del servidor se ejecute correctamente y se envíe la respuesta al cliente.

En términos de diseño y presentación de la interfaz de usuario, se ha utilizado el framework Bootstrap. Bootstrap es un conjunto de herramientas de código abierto que facilita la creación de interfaces web modernas y responsivas, lo que significa que la página se adapta automáticamente a diferentes tamaños de pantalla, como las de computadoras, tabletas y teléfonos móviles. Con Bootstrap, se pueden crear rápidamente componentes como formularios, botones, tablas y modales, que son esenciales para la interacción del usuario con el sistema. Además, su integración con las páginas JSP permite mejorar la experiencia de usuario, dándole un diseño limpio y atractivo sin tener que escribir mucho código CSS o JavaScript manualmente.

El proceso de eliminación de un modelo de computadora, por ejemplo, se maneja a través de un Servlet que recibe la solicitud del usuario para eliminar

un modelo específico de la base de datos. El Servlet primero verifica si la solicitud es válida, luego interactúa con la base de datos utilizando la clase EliminarModeloComputadora para ejecutar la consulta SQL de eliminación. Después de ejecutar la consulta, el Servlet redirige al usuario a una página JSP que muestra una confirmación de que el modelo ha sido eliminado correctamente, o bien a una página de error si no se pudo realizar la operación. Este enfoque permite que todo el proceso se maneje de forma eficiente y centralizada en el servidor.

En resumen, la lógica del programa está construida sobre una arquitectura que separa la lógica del negocio (manejo de datos y procesamiento de solicitudes) de la presentación (interfaz de usuario). Utilizando una combinación de Servlets, JSP y una base de datos MySQL, el sistema es capaz de manejar eficazmente las operaciones solicitadas por el usuario, garantizando una experiencia de usuario fluida y segura. Las librerías utilizadas proporcionan una infraestructura robusta para la conexión a la base de datos, el manejo de las solicitudes y respuestas HTTP, y la generación dinámica de contenido web. La integración de Bootstrap asegura que el diseño de la interfaz sea moderno y responsivo, mejorando la accesibilidad del sistema desde cualquier dispositivo.

Ejemplo de codigo:

conexión a db:

```
public class ConexionBaseDeDatos {

                private static final String URL_MYSQL = "jdbc:mysql://localhost:3306/computadorafeliz"; // Asegurarbos de que el nombre de la base de datos sea correcto
    private static final String USER = "root";
    private static final String PASSWORD = "Eleazar123Colop";
    private static Connection connection = null;

    public static Connection getConnection() {
        if (connection != null) return connection;

        try {
            // Cargar el driver JDBC
            Class.forName("com.mysql.cj.jdbc.Driver");
```

```
            // Establecer la conexión
                connection = DriverManager.getConnection(URL_MYSQL, USER,
PASSWORD);
            System.out.println("Conexión exitosa a la base de datos.");

        } catch (ClassNotFoundException e) {
            System.err.println("Error al cargar el driver JDBC: " + e.getMessage());
        } catch (SQLException e) {
                    System.err.println("Error al conectar con la base de datos: " +
e.getMessage());
        }

        return connection;
    }


}
```

Consultas:

```
public class ConsultarUsuario {

    private Connection connection;

    public ConsultarUsuario(Connection connection) {
        this.connection = connection;
    }

    /**
         * Consulta el usuario en la base de datos con su nombre de usuario y
contraseña.
     *
     * @param username El nombre de usuario
     * @param password La contraseña del usuario
      * @return El tipo de usuario (editor, administrador, suscriptor, especial) o
null si las credenciales no son válidas.
     * @throws SQLException En caso de error en la base de datos
     */
```

```
    public String consultarTipoUsuario(String username, String password) throws SQLException {
        String sql = "SELECT rol FROM usuarios WHERE username = ? AND password = ?";
    try (PreparedStatement ps = connection.prepareStatement(sql)) {
        ps.setString(1, username);
        ps.setString(2, password);

        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                // Si el usuario existe, devolver el tipo de usuario
                return rs.getString("rol");
            } else {
                // Si no se encuentra, retornar null
                return null;
            }
        }
    }
  }
}
```

Servlet:

```
/**
 *
 * @author eleaz
 */
@WebServlet(name = "RegistroServlet", urlPatterns = {"/RegistroServlet"})
@MultipartConfig(fileSizeThreshold = 1024 * 1024, maxFileSize = 1024 * 1024 * 5, maxRequestSize = 1024 * 1024 * 5 * 5)
public class RegistroServlet extends HttpServlet {

  @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    // Crear la conexión
```

```
    Connection connection = ConexionBaseDeDatos.getConnection();

    try {
        if (connection == null) {
            System.out.println("No se pudo establecer la conexión a la base de datos.");
            response.sendRedirect(request.getContextPath() + "/error.jsp");
            return;
        }

        // Recibir los datos del formulario
        String nombre = request.getParameter("nombre");
        String username = request.getParameter("username");
        String contrasena = request.getParameter("contrasena");
        String tipoUsuario = request.getParameter("tipoUsuario").toLowerCase();

        // Verificar si el usuario ya existe
        InsertarUsuario insertarUsuario = new InsertarUsuario(connection);
            if (insertarUsuario.usuarioExiste(username)) { // Aquí deberías pasar username en vez de nombre
                response.sendRedirect(request.getContextPath() + "/usuarioRepetido.jsp");
            return;
        }

        // Crear el objeto usuario
        Usuario usuario = new Usuario(
            nombre,
            username,
            contrasena,
            TipoUsuario.valueOf(tipoUsuario)
        );

        // Insertar el usuario en la base de datos
        //insertarUsuario.registrarUsuario(usuario);
```

```
        // Redirigir a una página de confirmación
                                //response.sendRedirect(request.getContextPath()   +
"/confirmacion.jsp");
        // Intentar registrar el usuario
        try {
            insertarUsuario.registrarUsuario(usuario);
                                    response.sendRedirect(request.getContextPath()  +
"/confirmacion.jsp"); // Si el registro es exitoso
        } catch (SQLIntegrityConstraintViolationException e) {
                                    response.sendRedirect(request.getContextPath()  +
"/usuarioRepetido.jsp"); // Usuario duplicado
        }

      } catch (Exception e) {
         e.printStackTrace();
         response.sendRedirect(request.getContextPath() + "/error.jsp"); // Página
de error genérico
      } finally {
         // Cerrar la conexión
      }
   }
}
```

Jsp:

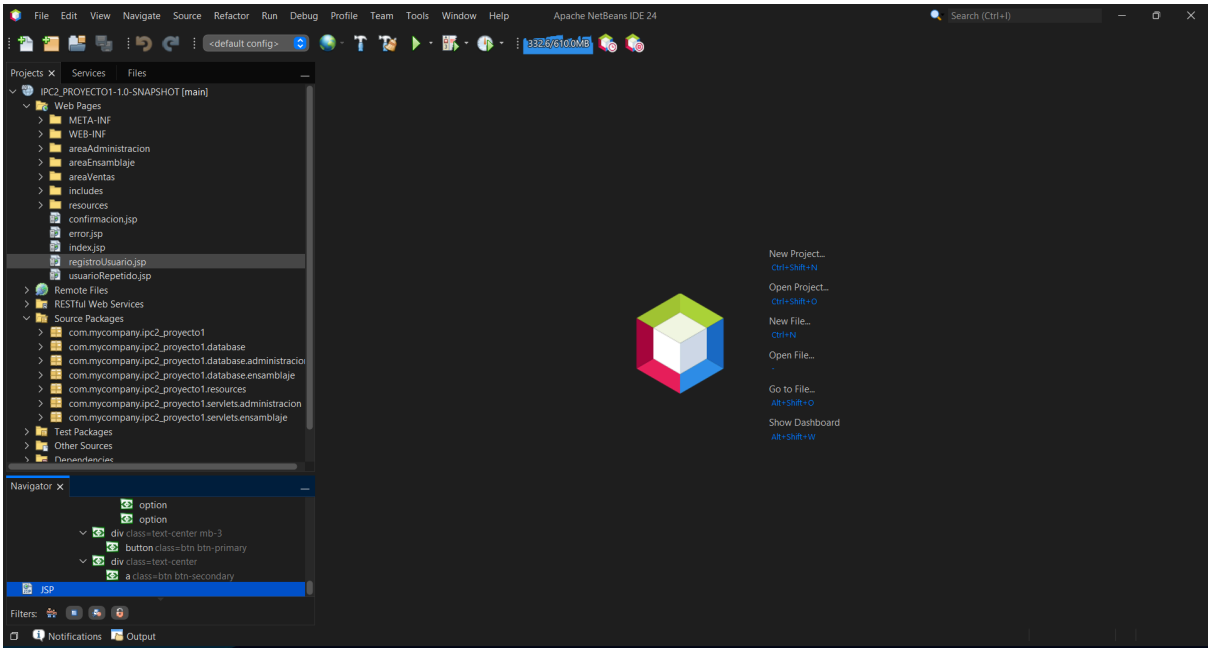
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Registro de Usuarios - COMPUTADORA FELIZ</title>
   <jsp:include page="/includes/resources.jsp"/>
                                                                        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2b
RjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
 <style>
```

```
    /* Asegurar que el body tenga altura completa */
    body {
                                                                background:
url('https://64.media.tumblr.com/cca4f06484b447c0687f0325af5b38c9/428a8db
1dc8ae92f-87/s1280x1920/7c751558b1d93e15c2d885cff2162ddb95059b8d.gif')
no-repeat center center fixed;
        background-size: cover;
    }

    /* Estilo para el texto */
    body, p, h1, h2, h3, h4, h5, h6, span {

        color: white; /* Texto en blanco */
    }
        </style>
</head>
<body>
    <div class="container mt-5">
        <h2 class="text-center">Registro de Usuario</h2>
                                                    <form          method="POST"
action="${pageContext.servletContext.contextPath}/RegistroServlet"
enctype="multipart/form-data">
            <div class="mb-3">
                <label for="nombre" class="form-label">Nombre</label>
                            <input type="text" class="form-control" id="nombre"
name="nombre" required>
            </div>

            <div class="mb-3">
                <label for="username" class="form-label">Username (único)</label>
                            <input type="text" class="form-control" id="username"
name="username" required pattern="[a-zA-Z0-9_]+" title="Solo letras,
números y guion bajo">
            </div>

            <div class="mb-3">
                <label for="contrasena" class="form-label">Contraseña</label>
```

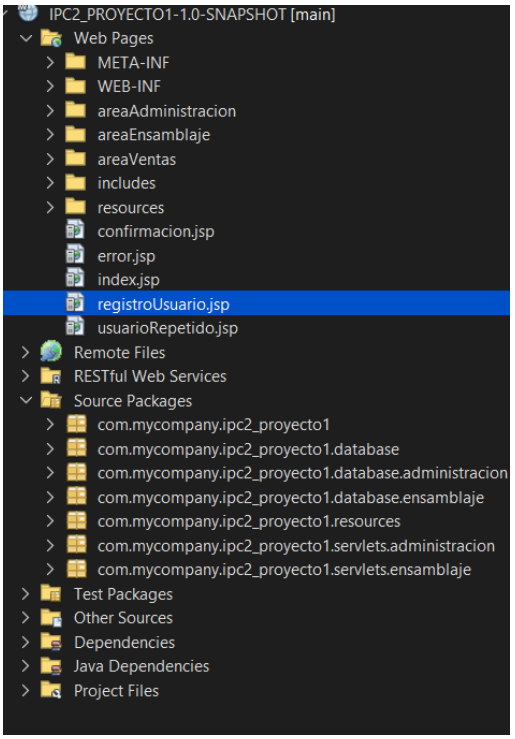
```
                    <input type="password" class="form-control" id="contrasena"
name="contrasena" required minlength="6">
        </div>

        <div class="mb-3">
            <label for="tipoUsuario" class="form-label">Tipo de Usuario (ROL -
COMPUTADORA FELIZ)</label>
                <select class="form-select" id="tipoUsuario" name="tipoUsuario"
required>
                <option value="ensamblaje">Ensamblaje</option>
                <option value="ventas">Ventas</option>
                <option value="administracion">Administración</option>
            </select>
        </div>

        <div class="text-center mb-3">
            <button type="submit" class="btn btn-primary">Registrar</button>
        </div>
        <div class="text-center">
                <a href="index.jsp" class="btn btn-secondary">Volver a la página
principal</a>
        </div>
      </form>
   </div>
</body>
</html>
```

Anexos:

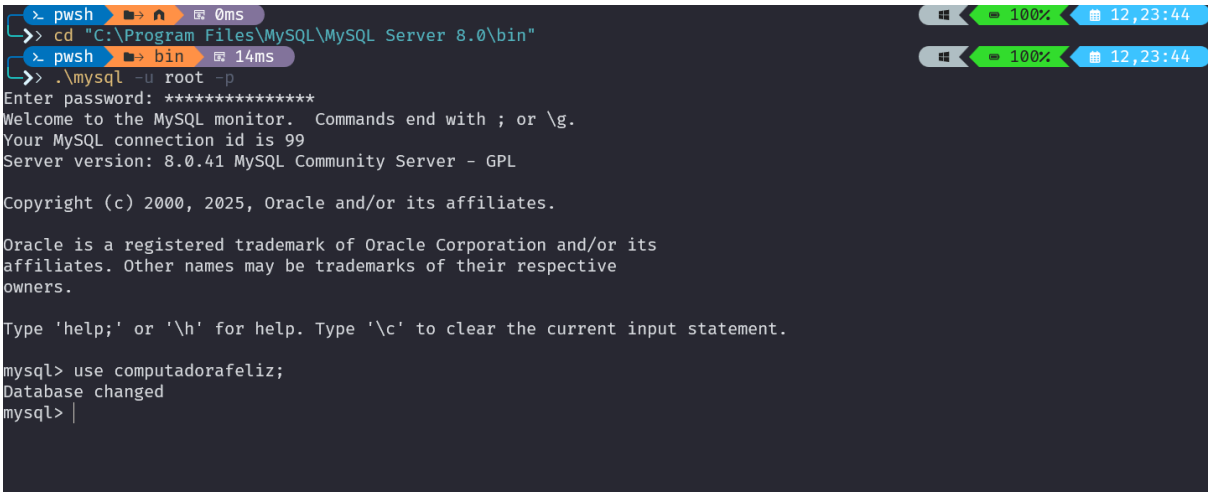
netbeans:



organizacion del proyecto:



ingresar a db desde terminal:



tablas:

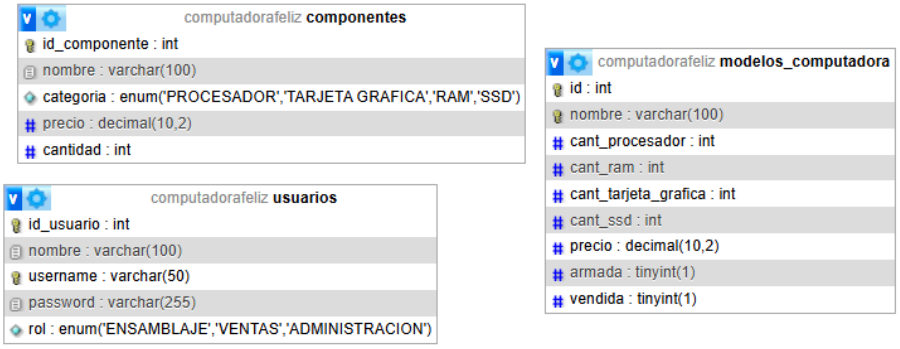


diagrama entidad relacion:

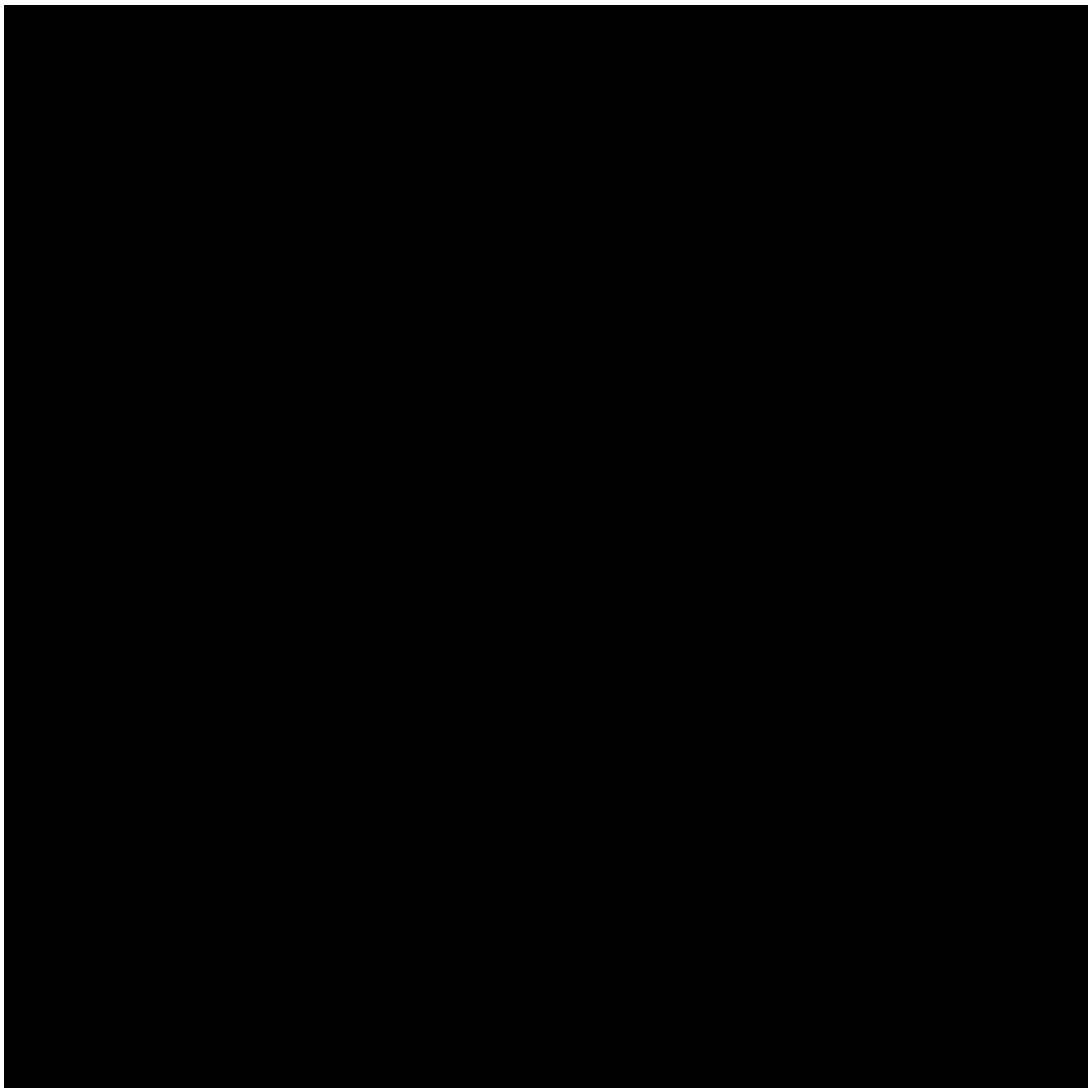


diagrama de clases: