

Bases de Datos

Trabajo colaborativo

Aingeru García Blas
Joseba Gómez Rodríguez

Índice

| | |
|--|-----------|
| 1. Introducción | 2 |
| 2. Aprendizaje colaborativo. Fases. | 2 |
| 2.1. Fase 1 | 2 |
| 2.1.1. Consultas generadas | 2 |
| 2.1.2. Evaluación recibida | 4 |
| 2.2. Fase 2 | 5 |
| 3. Laboratorio 10: JDBC | 5 |
| 3.1. Sobre la clase Database | 5 |
| 3.2. Ejercicios 8 y 9 | 7 |
| 3.2.1. Procedimientos de Aingeru: | 10 |
| 3.2.2. Procedimientos de Joseba: | 10 |
| 3.3. Ejecución de los ejercicios | 11 |
| 4. Comentarios y observaciones | 15 |
| 5. Cambios | 15 |
| 6. Bibliografía | 16 |

1. Introducción

En este informe, presentamos el trabajo destacado que hemos realizado en la asignatura. Durante el desarrollo de este proyecto, hemos llevado a cabo una serie de consultas en un sistema de gestión de bases de datos, utilizando tanto el lenguaje **RELAX** [1] como **SQL**. Además, hemos ido un paso más allá al crear un programa en **Java** que establece una conexión con la base de datos y realiza diversas operaciones.

Adicionalmente, detallaremos el trabajo colaborativo realizado a lo largo del cuatrimestre. Presentaremos las consultas realizadas, compartiremos las evaluaciones recibidas en eGela, mostraremos el código del programa **Java** desarrollado e incluiremos evidencias de la ejecución exitosa de los ejercicios correspondientes al **Laboratorio 10**.

El objetivo de este informe es brindar una visión completa y enriquecedora de nuestro proyecto en la asignatura de Bases de Datos, destacando los pasos y tareas clave del trabajo colaborativo.



Figura 1: Bases de Datos

2. Aprendizaje colaborativo. Fases.

El aprendizaje colaborativo fomenta la participación activa de los estudiantes en la construcción conjunta del conocimiento. La evaluación anónima de ejercicios en este contexto promueve un ambiente de igualdad y objetividad, permitiendo una retroalimentación sincera y constructiva entre los estudiantes. Esta práctica facilita la expresión libre de ideas y opiniones, promoviendo un ambiente de respeto y confianza mutua, y enriqueciendo el proceso de aprendizaje con diversas perspectivas y enfoques.

2.1. Fase 1

2.1.1. Consultas generadas

A continuación, presentamos algunas de las consultas que hemos generado, tanto en **RELAX** como en **SQL**, junto con sus respectivos resultados. Estas consultas se han realizado con el objetivo de obtener información específica de nuestra base de datos.

Consulta 1:

- Álgebra relacional

Autor: Aingeru García.

Enunciado: Obtener los nombres de los buques que hayan sido utilizados en algún proyecto y cuyo dueño sea una persona cuyo nombre empieza por la letra K u O.

Consulta: $\pi_{\text{NombreBuque}}(\sigma_{\text{Dueno LIKE 'K \%'} \text{ OR Dueno LIKE 'O \%'}}(\text{Buque} \bowtie \text{HAN_UTILIZADO}))$

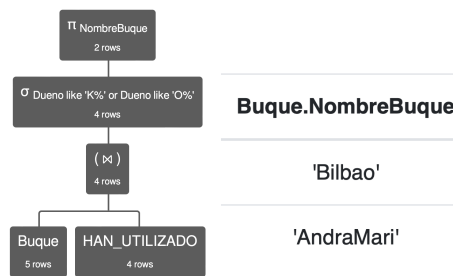


Figura 2: Resultado Relax

Consulta 2:

- Álgebra relacional

Autor: Joseba Gómez.

Enunciado: Número del departamento que haya trabajado con un buque con fecha de inicio posterior a la fecha de inicio del director del departamento y el nombre de dicho buque.

Consulta: $\pi_{\text{DEPARTAMENTO.NumeroD, HAN_UTILIZADO.NombreBuque}}(\text{HAN_UTILIZADO} \bowtie_{\text{FechaInicio} > \text{FechaInicDire}} \text{DEPARTAMENTO})$

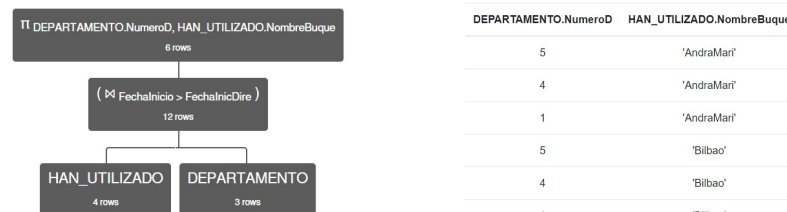
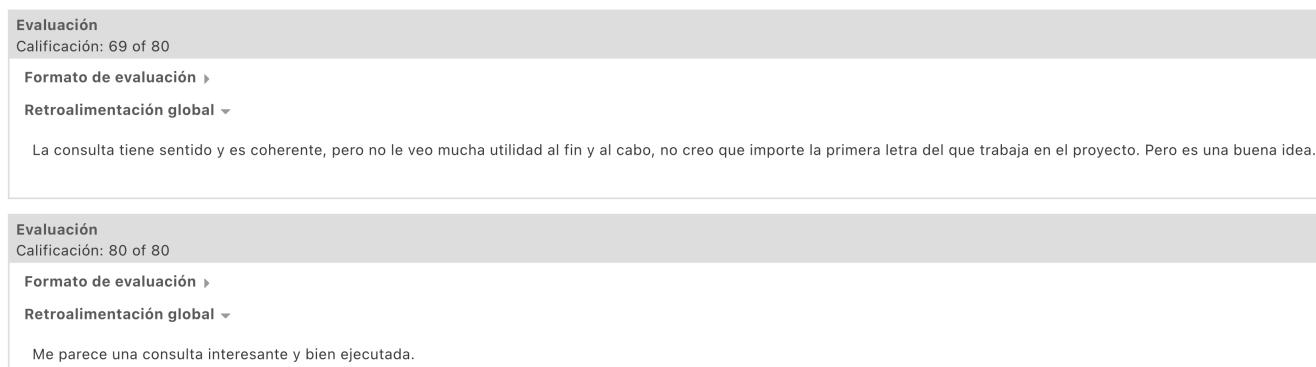


Figura 3: Resultado Relax

2.1.2. Evaluación recibida

Aquí se presenta una imagen que muestra la evaluación que hemos recibido en eGela para cada una de las consultas.

Consulta 1



The image shows two screenshots of the eGela evaluation interface. The top screenshot is for 'Consulta 1' with a score of 69 out of 80. It includes a 'Formato de evaluación' link, a 'Retroalimentación global' dropdown, and a feedback comment: 'La consulta tiene sentido y es coherente, pero no le veo mucha utilidad al fin y al cabo, no creo que importe la primera letra del que trabaja en el proyecto. Pero es una buena idea.' The bottom screenshot is for 'Consulta 2' with a score of 80 out of 80. It includes the same links and a feedback comment: 'Me parece una consulta interesante y bien ejecutada.'

Evaluación
Calificación: 69 of 80

[Formato de evaluación](#) ▶

Retroalimentación global ▼

La consulta tiene sentido y es coherente, pero no le veo mucha utilidad al fin y al cabo, no creo que importe la primera letra del que trabaja en el proyecto. Pero es una buena idea.

Evaluación
Calificación: 80 of 80

[Formato de evaluación](#) ▶

Retroalimentación global ▼

Me parece una consulta interesante y bien ejecutada.

Figura 4: Evaluación de la consulta 1.

Consulta 2



The image shows two screenshots of the eGela evaluation interface. The top screenshot is for 'Consulta 1' with a score of 80 out of 80. It includes a 'Formato de evaluación' link, a 'Retroalimentación global' dropdown, and a feedback comment: 'La consulta tiene sentido y es coherente, pero no le veo mucha utilidad al fin y al cabo, no creo que importe la primera letra del que trabaja en el proyecto. Pero es una buena idea.' The bottom screenshot is for 'Consulta 2' with a score of 80 out of 80. It includes the same links and a feedback comment: 'Me parece una consulta interesante y bien ejecutada.'

Ebaluazioa
Kalifikazioa: 80/80

Ebaluazio-formularioa ▶

Feedback orokorra ▼

Puede ser interesante esa información

Ebaluazioa
Kalifikazioa: 75/80

Ebaluazio-formularioa ▶

Feedback orokorra ▼

Es una consulta interesante, práctica y bien ejecutada.

Figura 5: Evaluación de la consulta 2.

2.2. Fase 2

Durante ésta fase, nosotros hemos sido los encargados de evaluar las consultas SQL en las que unos compañeros han trabajado. Todo esto, de manera totalmente **anónima**.

3. Laboratorio 10: JDBC



Figura 6: Java Database Connectivity

En esta sección, se presenta el desarrollo del Laboratorio 10, que se enfoca en el uso de JDBC (Java Database Connectivity) para interactuar con bases de datos desde una aplicación Java. JDBC es una API estándar de Java que proporciona un conjunto de clases e interfaces para permitir la conexión, consulta y manipulación de bases de datos relacionales.

Durante el Laboratorio 10, se exploran diferentes aspectos relacionados con JDBC, como la configuración de la conexión a la base de datos, la ejecución de consultas SQL, la gestión de transacciones y la manipulación de resultados. A través de ejercicios prácticos, hemos adquirido experiencia básica en el uso de JDBC y en la implementación de operaciones de bases de datos en sus aplicaciones Java.

Presentaremos una descripción detallada de los ejercicios realizados durante el Laboratorio 10, junto con el código Java correspondiente y los resultados obtenidos.

3.1. Sobre la clase Database

El motivo principal de modularizar y crear la conexión a parte es mejorar la estructura del **código** y evitar la repetición innecesaria de **código** en los 9 ejercicios del laboratorio. Además, esto facilita la gestión de las conexiones a las bases de datos y mejora la eficiencia en el manejo de las mismas.

La clase `Database` sigue el patrón de diseño **Singleton**, lo que significa que solo puede haber una instancia de esta clase en toda la aplicación. Esto se logra mediante el uso de una variable estática `instance` y un método `getInstance()` que devuelve la instancia existente o crea una nueva si no existe. El constructor (privado) inicializa las conexiones se hace en el método `initConnections()`, donde se verifica si las conexiones ya han sido establecidas. En caso contrario, se utiliza el `DriverManager` para establecer la conexión tanto con una base de datos MySQL como con una base de datos Oracle.

La clase proporciona dos métodos públicos para obtener las conexiones a las bases de datos: `getMySQLConn()` y `getOracleConn()`. Estos métodos verifican si las conexiones ya han sido inicializadas mediante el método `initConnections()`. En caso contrario, se invoca dicho método para establecer las conexiones antes de devolverlas.

```
package Managers;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Database {
    private static Database instance = null;
    private static Connection mysqlConn;
    private static Connection oracleConn;

    private Database() throws SQLException {
        initConnections();
    }

    public static synchronized Database getInstance() throws SQLException {
        if (instance == null) {
            instance = new Database();
        }
        return instance;
    }

    private static void initConnections() throws SQLException {
        if (mysqlConn == null) {
            mysqlConn = DriverManager.getConnection("jdbc:mysql://dif-mysql.ehu.es:3306/DBC33?useSSL=false", "DBC33", "DBC33");
        }
        if (oracleConn == null) {
            oracleConn = DriverManager.getConnection("jdbc:oracle:thin:@vsids11.si.ehu.es:1521:gipuzkoa", "BDC33", "BDC33");
        }
    }

    public static Connection getMySQLConn() throws SQLException {
        if (mysqlConn == null) {
            initConnections();
        }
        return mysqlConn;
    }

    public static Connection getOracleConn() throws SQLException {
        if (oracleConn == null) {
            initConnections();
        }
        return oracleConn;
    }
}
```

Figura 7: Código de la clase Database

3.2. Ejercicios 8 y 9

Código de ejercicio, al cual hemos llamado Ejercicio89.java

```

/*
 * This file is part of the Lab10 project.
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the
 * Free Software Foundation; either version 3 of the License, or (at your
 * option) any later version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
 * more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 *
 * @authors - Aingeru García | Github: https://github.com/geru-scotland
 *          - Joseba Gómez   | Github: https://github.com/JotaUwU
 */

import Managers.Database;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.*;

public class Ejercicios89 {

    public static void main(String[] args) {
        Connection mysqlConn = null;
        PreparedStatement stmt = null;
        ResultSet res = null;

        try {
            // 1. Crear conexión
            mysqlConn = Database.getMysqlConn();

            // 2. Preparar y ejecutar consulta para mostrar todos los viajes
            stmt = mysqlConn.prepareStatement("{call SHOW_VIAJES()}");
            res = stmt.executeQuery();

            // 3. Mostrar resultados de todos los viajes
            System.out.println("#####");
            System.out.println("                Todos los viajes                ");
            System.out.println("#####");
            System.out.printf("%-15s %-15s %-5s %-10s %-15s %-10s\n", "Destino",
                "FechaSalida", "Dias", "PrecioDia", "CiudadSalida", "DNI");

```



```

while (res.next()) {
    System.out.printf("%-15s %-15s %-5s %-10s %-15s %-10s%n",
        res.getString("Destino"),
        res.getString("FechaSalida"),
        res.getString("Dias"),
        res.getString("PrecioDia"),
        res.getString("CiudadSalida"),
        res.getString("DNI"));
}

closeResources(stmt, res);

// 4. Preparar y ejecutar consulta para mostrar viajes cortos
stmt = mysqlConn.prepareStatement("{call SHOW_SHORT_VIAJES()}");
res = stmt.executeQuery();

// 5. Mostrar resultados de viajes cortos
System.out.println("\n#####");
System.out.println("                Viajes cortos (< 5 días)                ");
System.out.println("#####");
System.out.printf("%-15s %-15s %-5s %-10s %-15s %-10s%n", "Destino",
    "FechaSalida", "Dias", "PrecioDia", "CiudadSalida", "DNI");

while (res.next()) {
    System.out.printf("%-15s %-15s %-5s %-10s %-15s %-10s%n",
        res.getString("Destino"),
        res.getString("FechaSalida"),
        res.getString("Dias"),
        res.getString("PrecioDia"),
        res.getString("CiudadSalida"),
        res.getString("DNI"));
}

// 6. Preparar y ejecutar consulta para mostrar todos los hoteles
stmt = mysqlConn.prepareStatement("{call SHOW_HOTEL()}");
res = stmt.executeQuery();

// 7. Mostrar resultados de todos los hoteles
System.out.println("#####");
System.out.println("                Todos los hoteles                ");
System.out.println("#####");
System.out.printf("%-20s %-20s %-20s %-20s%n", "IdHotel", "Nombre",
    "Localidad", "Capacidad");

while (res.next()) {
    System.out.printf("%-20s %-20s %-20s %-20s%n",
        res.getString("IdHotel"),
        res.getString("Nombre"),
        res.getString("Localidad"),
        res.getString("Capacidad"));
}

closeResources(stmt, res);

```

```

// 8. Preparar y ejecutar consulta para mostrar todos los hoteles de Donostia
stmt = mysqlConn.prepareStatement("{call SHOW_HOTEL_DONOS()}");
res = stmt.executeQuery();

// 9. Mostrar resultados de todos los hoteles de donostia
System.out.println("#####");
System.out.println("                Todos los hoteles de Donostia");
System.out.println("#####");
System.out.printf("%-20s %-20s %-20s %-20s\n", "IdHotel", "Nombre",
"Localidad", "Capacidad");

while (res.next()) {
    System.out.printf("%-20s %-20s %-20s %-20s\n",
        res.getString("IdHotel"),
        res.getString("Nombre"),
        res.getString("Localidad"),
        res.getString("Capacidad"));
}

} catch (SQLException ex) {
    System.out.println(ex.getMessage());
} finally {
    closeResources(stmt, res);

    // 9. Cerrar conexión
    try {
        if (mysqlConn != null) {
            mysqlConn.close();
        }
    } catch (SQLException e) {
        System.out.println("Error al cerrar la conexión: " + e.getMessage());
    }
}

}

private static void closeResources(PreparedStatement stmt, ResultSet res) {
    try {
        if (stmt != null) {
            stmt.close();
        }
        if (res != null) {
            res.close();
        }
    } catch (SQLException e) {
        System.out.println("Error al cerrar recursos: " + e.getMessage());
    }
}
}

```

A continuación, se presentará el código de ejemplo de procedimientos almacenados en una base de datos MySQL. Estos procedimientos estarán disponibles en la base de datos y podrán ser invocados desde un programa Java para llevar a cabo operaciones específicas. Estos ejemplos sirven como referencia para comprender cómo implementar y utilizar procedimientos almacenados en MySQL a través de Java.

3.2.1. Procedimientos de Aingeru:

Procedimiento 1

```
DELIMITER
CREATE PROCEDURE SHOW_VIAJES()
BEGIN
    SELECT * FROM viaje;
END
DELIMITER;
```

Procedimiento 2

```
DELIMITER
CREATE PROCEDURE SHOW_SHORT_VIAJES()
BEGIN
    SELECT * FROM viaje WHERE dias < 5;
END
DELIMITER;
```

3.2.2. Procedimientos de Joseba:

Procedimiento 3

```
DELIMITER
CREATE PROCEDURE SHOW_HOTEL()
BEGIN
    SELECT * FROM hotel;
END
DELIMITER;
```

Procedimiento 4

```
DELIMITER
CREATE PROCEDURE SHOW_HOTEL_DONOS()
BEGIN
    SELECT * FROM hotel WHERE Localidad = 'Donostia';
END
DELIMITER;
```

3.3. Ejecución de los ejercicios

Aquí se muestra una imagen de la ejecución exitosa de nuestro código Java. Como se puede ver, el programa se conecta correctamente a la base de datos y realiza las operaciones previstas.

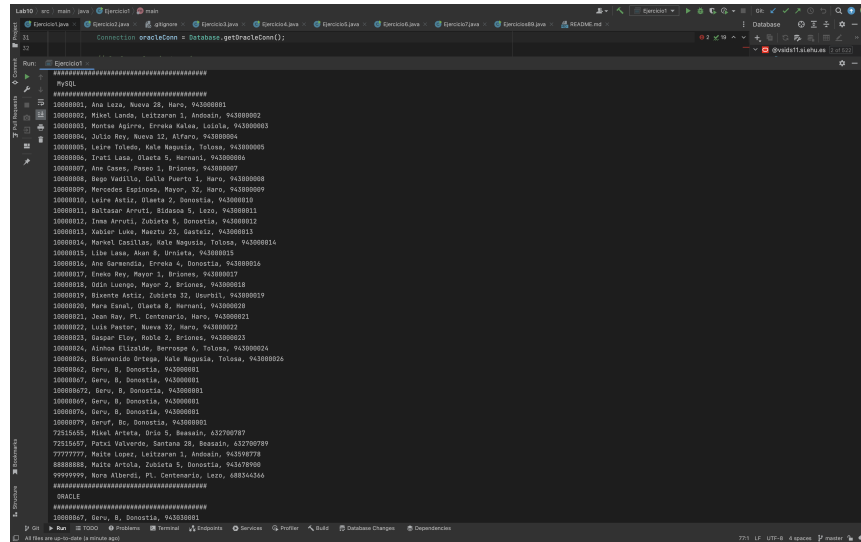


Figura 8: Ejecución del Ejercicio 1

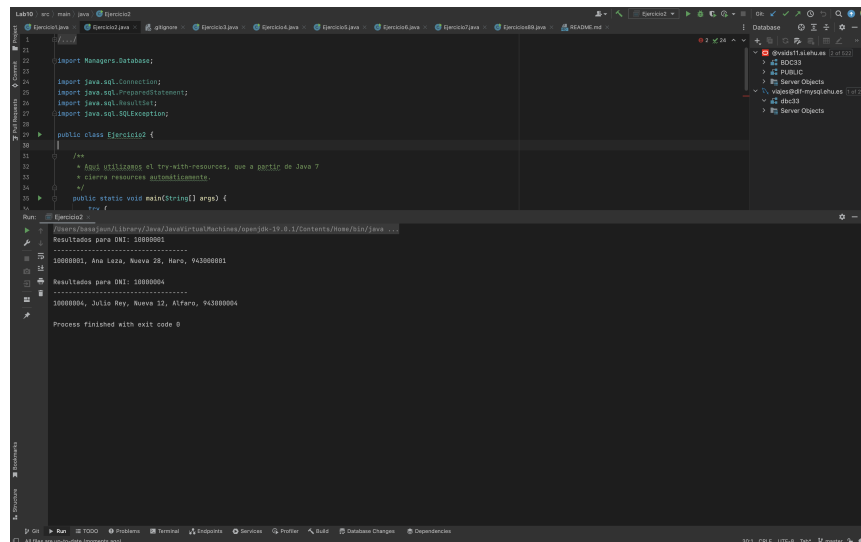


Figura 9: Ejecución del Ejercicio 2

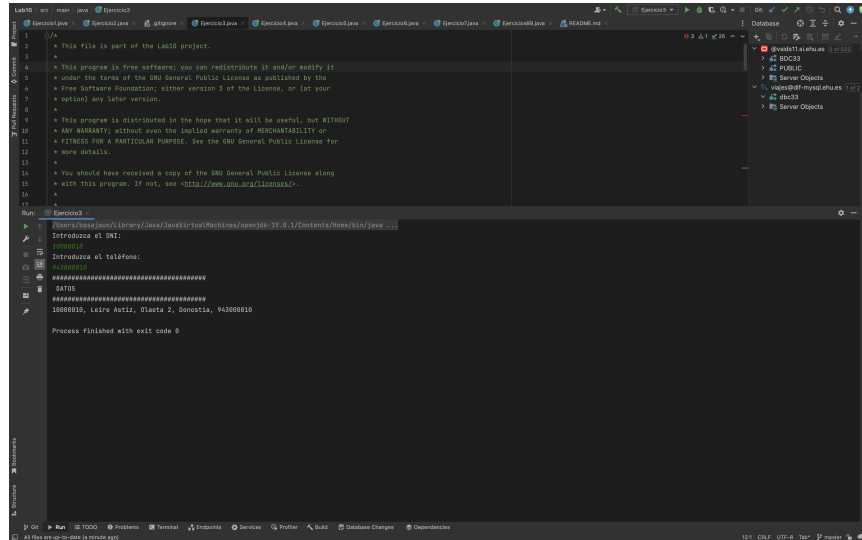


Figura 10: Ejecución del Ejercicio 3

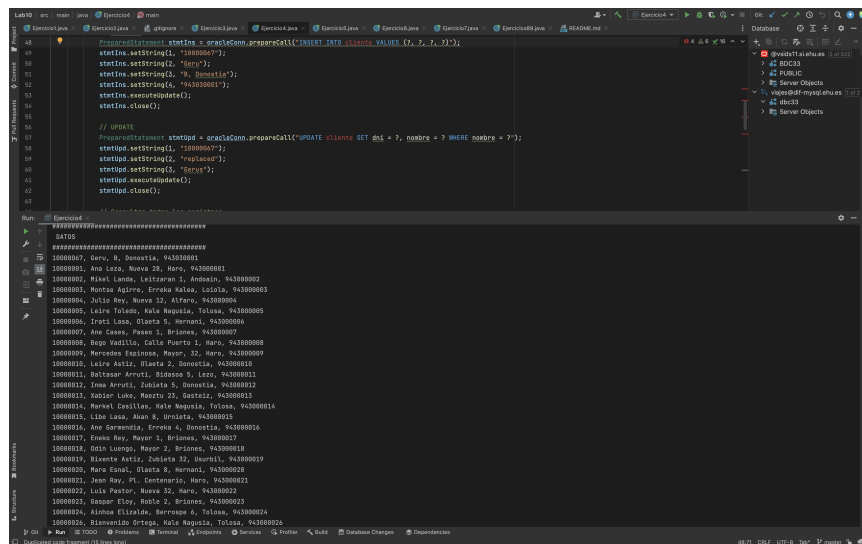


Figura 11: Ejecución del Ejercicio 4

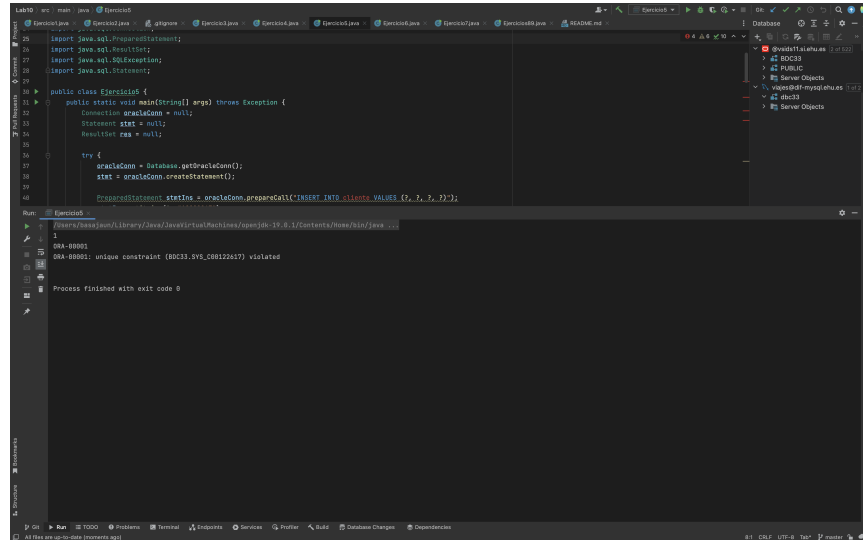


Figura 12: Ejecución del Ejercicio 4 (Control correcto)

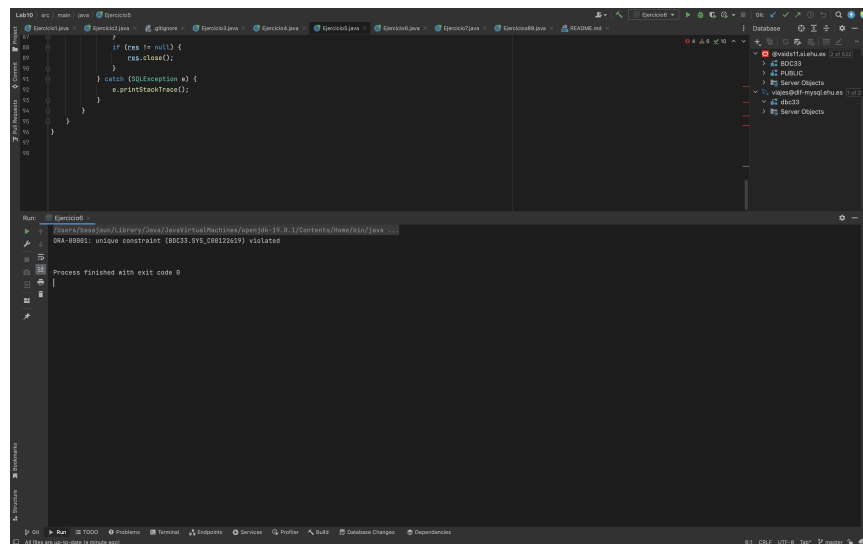


Figura 13: Ejecución del Ejercicio 6 (Control correcto)

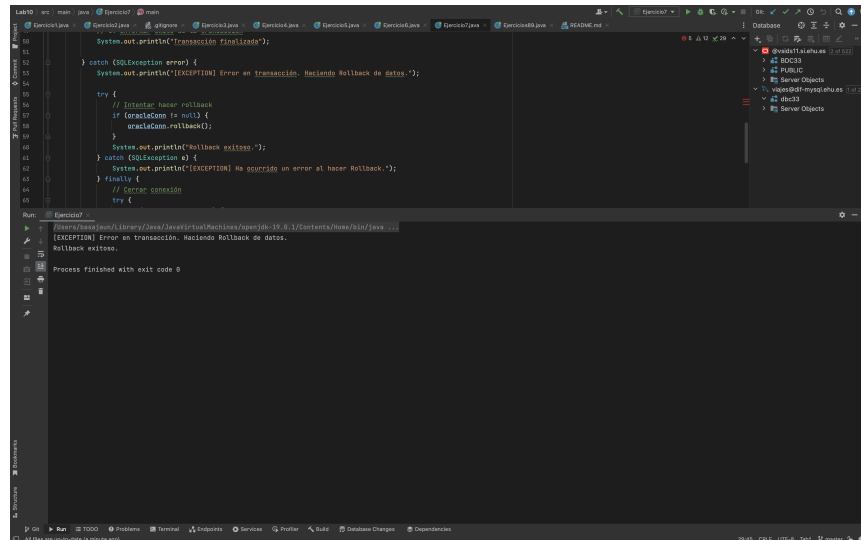


Figura 14: Ejecución del Ejercicio 7 (Rollback exitoso)

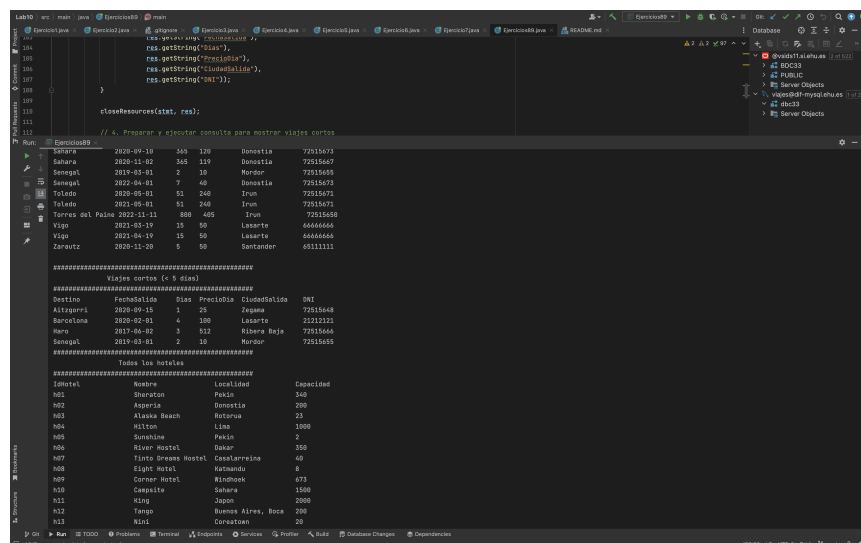


Figura 15: Ejecución del Ejercicio 8+9

4. Comentarios y observaciones

Este proyecto ha sido una excelente oportunidad para aplicar los conocimientos adquiridos en la asignatura de Bases de Datos. A través de este trabajo, hemos aprendido más sobre cómo realizar consultas en SQL y RELAX, así como a crear y gestionar una base de datos. También hemos mejorado nuestras habilidades en programación Java, especialmente en lo que respecta a la conexión y manipulación de bases de datos.

Consideramos un muy buen punto el trabajo por aprendizaje colaborativo. No obstante, creemos que si hubiese podido ser algo más constante y frecuente a lo largo del curso, podríamos habernos aprovechado algo más de la retroalimentación.

5. Cambios

Nos han sido solicitados algunos cambios tras la revisión del documento. Hemos realizado los siguientes:

- **Tema de Relax:** Hemos cambiado el tema de relax al modo *light*, para que tanto el árbol como los resultados se vean acorde con lo indicado.
- **Consulta 2:** Al parecer, la consulta fué mal formulada al pasarla a L^AT_EX, sin embargo tanto en eGela como en Relax se había formulado de otra manera, la cual consideramos correcta:

$$\pi_{\text{DEPARTAMENTO.NumeroD, HAN_UTILIZADO.NombreBuque}} (\text{HAN_UTILIZADO} \bowtie_{\text{FechaInicio} > \text{FechaInicDire}} \text{DEPARTAMENTO})$$

Figura 16: Consulta original (actual)

Consulta: $\pi_{\text{DEPARTAMENTO.NumeroD, HAN_UTILIZADO.NombreBuque}} (\sigma_{\text{HAN_UTILIZADO.FechaInicio} > \text{DEPARTAMENTO.FechaInicDire}} (\text{HAN_UTILIZADO} \bowtie \text{DEPARTAMENTO}))$

Figura 17: Consulta con error al pasar a L^AT_EX (previa)

6. Bibliografía

Referencias

- [1] <https://dbis-uibk.github.io/relax/landing>

*«La pregunta es la más creativa de las conductas
humanas»*

–Alex Osborn, en exclusiva para eGela.

12 de Mayo de 2023