

Pregunta 1

Tareas a llevar a cabo

1. Implementación de DBAccessor.java
2. Implementación de Exercise1PrintReportOverQuery.java

Configuración de la Conexión: DBAccessor

La clase DBAccessor ha de utilizar para gestionar las conexiones a la base de datos PostgreSQL de manera eficiente.

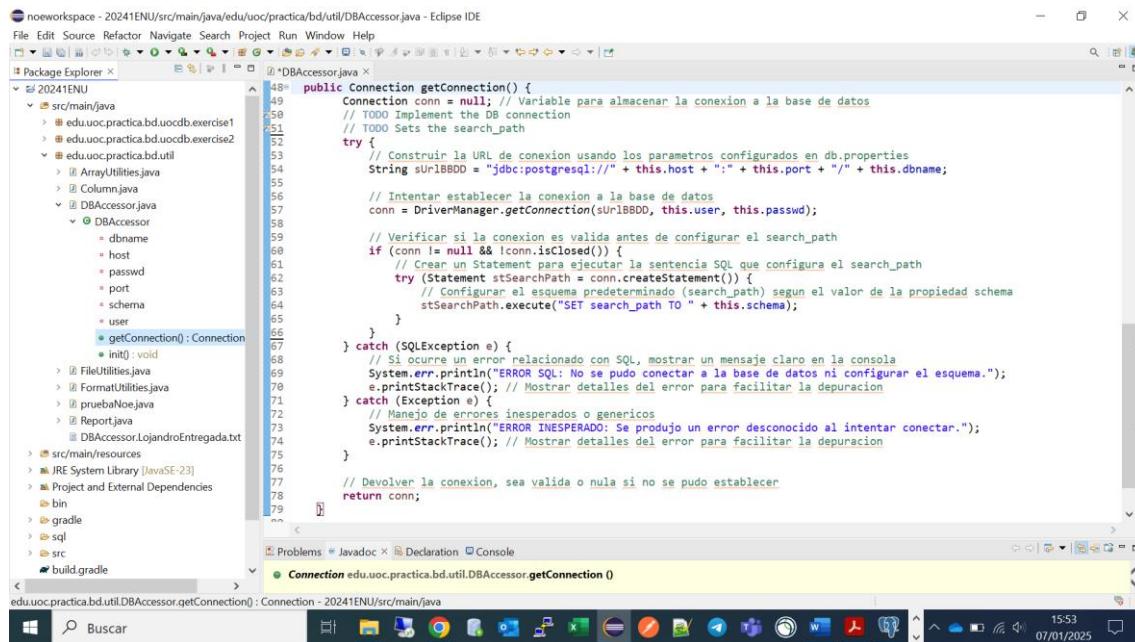
Por un lado se realiza la carga de Configuración desde db.properties. La clase lee los parámetros necesarios para la conexión (como host, port, dbname, user, passwd y schema) desde el archivo db.properties utilizando la clase Properties.

Por otro lado, realiza el establecimiento de la Conexión. La conexión a la base de datos se establece mediante el método getConnection, que utiliza DriverManager.getConnection para generar una conexión con la URL construida a partir de los parámetros cargados.

Lleva a cabo la configuración del search_path, ya que una vez establecida la conexión, se ha de ejecutar un comando SQL (SET search_path TO ...) para configurar el esquema predeterminado.

La clase ha de manejar de forma robusta posibles errores de conexión o SQL, mostrando mensajes según lo indicado en el enunciado.

Código resultante, método getConnection:



```
noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/util/DBAccessor.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X
20241ENU
src/main/java
edu.uoc.practica.bd.uocdb.exercise1
edu.uoc.practica.bd.uocdb.exercise2
edu.uoc.practica.bd.util
ArrayUtilities.java
Column.java
DBAccessor.java
DBAccessor
  dbname
  host
  passwd
  port
  schema
  user
  getConnection(): Connection
  init(): void
FileUtilities.java
FormatUtilities.java
pruebaNoe.java
Report.java
DBAccessor.LojandoEntregada.txt
src/main/resources
JRE System Library [JavaSE-23]
Project and External Dependencies
bin
gradle
sql
src
build.gradle
Connection edu.uoc.practica.bd.util.DBAccessor.getConnection()
48  public Connection getConnection() {
49      Connection conn = null; // Variable para almacenar la conexión a la base de datos
50      // TODO Implement the DB connection
51      // TODO Sets the search_path
52      try {
53          // Construir la URL de conexión usando los parámetros configurados en db.properties
54          String urlBDD = "jdbc:postgresql://" + this.host + ":" + this.port + "/" + this.dbname;
55
56          // Intentar establecer la conexión a la base de datos
57          conn = DriverManager.getConnection(urlBDD, this.user, this.passwd);
58
59          // Verificar si la conexión es válida antes de configurar el search_path
60          if (conn != null && !conn.isClosed()) {
61              // Crear un Statement para ejecutar la sentencia SQL que configura el search_path
62              try (Statement stSearchPath = conn.createStatement()) {
63                  // Configurar el esquema predeterminado (search_path) según el valor de la propiedad schema
64                  stSearchPath.execute("SET search_path TO " + this.schema);
65              }
66          }
67      } catch (SQLException e) {
68          // Si ocurre un error relacionado con SQL, mostrar un mensaje claro en la consola
69          System.err.println("ERROR SQL: No se pudo conectar a la base de datos ni configurar el esquema.");
70          e.printStackTrace(); // Mostrar detalles del error para facilitar la depuración
71      } catch (Exception e) {
72          // Manejo de errores inesperados o genericos
73          System.err.println("ERROR INESPERADO: Se produjo un error desconocido al intentar conectar.");
74          e.printStackTrace(); // Mostrar detalles del error para facilitar la depuración
75      }
76
77      // Devolver la conexión, sea válida o nula si no se pudo establecer
78      return conn;
79  }
```

El archivo properties tiene el siguiente contenido:

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left displays the project structure for 'noeworkspace - 20241ENU/src/main/resources/db.properties'. The main editor window shows the code for `DBAccessor.java`, which contains a static method `getConnection()` that reads properties from `db.properties`. The properties file contains:

```

host localhost
port 5432
dbname postgres
user noe
passwd noe
schema udd_20241
    
```

The bottom status bar indicates the code is Writable and at line 1:1:0.

Para validar de manera unitaria este componente he implementado una clase de pruebas que únicamente realiza un `getConnection` y vemos su resultado:

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left displays the project structure for 'noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/util/pruebaNoe.java'. The main editor window shows the code for `pruebaNoe.java`, which contains a `main` method that creates a `DBAccessor` instance and attempts to get a connection from the database. The code is as follows:

```

package edu.uoc.practica.bd.util;
import java.sql.Connection;
public class pruebaNoe {
    public static void main(String[] args) {
        // Crear una instancia de DBAccessor
        DBAccessor dbAccessor = new DBAccessor();
        try {
            // Inicializar el DBAccessor para cargar los parametros desde db.properties
            dbAccessor.init();
            // Intentar establecer la conexion a la base de datos
            try (Connection conn = dbAccessor.getConnection()) {
                if (conn != null && !conn.isClosed()) {
                    System.out.println("Conexion a la base de datos establecida correctamente.");
                } else {
                    System.err.println("Error: No se pudo establecer la conexion con la base de datos.");
                }
            }
        } catch (Exception e) {
            // Manejo de errores inesperados
            System.err.println("Se produjo un error inesperado durante la conexion.");
            e.printStackTrace();
        }
    }
}
    
```

The bottom status bar indicates the code is Writable and at line 1:1:0.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with files like DBAccessor.java, db.properties, and pruebaNoe.java.
- DBAccessor.java Content:**

```
1 package edu.uoc.practica.bd.util;
2
3 import java.sql.Connection;
4
5 public class pruebaNoe {
6
7    public static void main(String[] args) {
8        // Crear una instancia de DBAccessor
9        DBAccessor dbAccessor = new DBAccessor();
10
11        try {
12            // Inicializar el DBAccessor para cargar los parametros desde db.properties
13            dbAccessor.init();
14
15            // Intentar establecer la conexion a la base de datos
16            try (Connection conn = dbAccessor.getConnection()) {
17                if (conn != null && !conn.isClosed()) {
18                    System.out.println("Conexion a la base de datos establecida correctamente.");
19                } else {
20                    System.err.println("Error: No se pudo establecer la conexion con la base de datos.");
21                }
22            }
23        } catch (Exception e) {
24            e.printStackTrace();
25        }
26    }
27}
```
- Terminal Window:** Shows the output of the application execution:

```
Conexion a la base de datos establecida correctamente.
```
- Status Bar:** Shows the status "terminated> pruebaNoe [Java Application]" and the date "(7 ene 2024)".

La solución de lo que quedaba pendiente de llenar a la que se ha llegado es la siguiente:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** Shows the project structure for "20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise1".
- Editor View:** Displays the Java code for `Exercise1PrintReportOverQuery.java`. The code implements a `main` method that creates an instance of `Exercise1PrintReportOverQuery` and calls its `run` method. The `run` method initializes a `DBAccessor`, gets a connection, and creates a `Report` object.
- Bottom Status Bar:** Shows the current file as `Exercise1PrintReportOverQuery.java` and the line number as 16.

```
newworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise1/Exercise1PrintReportOverQuery.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X DBAccessor.java db.properties pruebaNoe.java create_view.sql Exercise1PrintReportOverQuery - copiaLo.txt Exercise1PrintReportOverQuery.java
20241ENU
src/main/java
  edu.uoc.practica.bd.uocdb.exercise1
    Exercise1PrintReportOverQuery.java
    Exercise1RowJava
    Exercise1PrintReportOverQuery - copiaLo.txt
  edu.uoc.practica.bd.uocdb.exercise2
  edu.uoc.practica.bd.util
    ArrayUtilities.java
    Column.java
    DBAccessor.java
      DBAccessor
        dbname
        host
        passwd
        port
        schema
        user
        getConnection(): Connection
        init(): void
    FileUtils.java
    FormatUtilities.java
    pruebaNoe.java
    Report.java
    DBAccessor.LojadnroEntregada.txt
src/main/resources
  db.properties
  exercise2.data
JRE System Library [JavaSE-23]
Project and External Dependencies

1 package edu.uoc.practica.bd.uocdb.exercise1;
2
3 import edu.uoc.practica.bd.util.*;
4 import java.sql.*;
5 import java.util.*;
6
7 public class Exercise1PrintReportOverQuery {
8
9     public static void main(String[] args) {
10         Exercise1PrintReportOverQuery app = new Exercise1PrintReportOverQuery();
11         app.run();
12     }
13
14     private void run() {
15         DBAccessor dbaccessor = new DBAccessor();
16         dbaccessor.init();
17         Connection conn = dbaccessor.getConnection();
18
19         if (conn != null) {
20             Statement cstmt = null;
21             ResultSet resultSet = null;
22
23             try {
24                 List<Column> columns = Arrays.asList(new Column("Zone", 12, "zone_name"),
25                     new Column("Capital", 12, "capital_town"),
26                     new Column("Climate", 15, "climate"),
27                     new Column("Region", 20, "region"),
28                     new Column("Last selling", 12, "last_selling"),
29                     new Column("Total", 5, "total_quantity")
30                 );
31
32             Report report = new Report():
33
34         }
35     }
36
37     public Column(String label, int columnWidth, String attributeName) {
38         this.label = label;
39         this.columnWidth = columnWidth;
40         this.attributeName = attributeName;
41     }
42
43     private String label;
44     private int columnWidth;
45     private String attributeName;
46
47     @Override
48     public String toString() {
49         return "Column{" + "label=" + label + ", columnWidth=" + columnWidth + ", attributeName=" + attributeName + '}';
50     }
51
52     public static void main(String[] args) {
53     }
54 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "20241ENU".
- Editor:** Displays the Java code for `Exercise1PrintReportOverQuery.java`. The code is a JDBC query to select data from a view named `best_selling_zones`.
- Bottom Bar:** Includes tabs for `Problems`, `Javadoc`, `Declaration`, and `Console`.
- Code Snippet:**

```
31     Report report = new Report();
32     report.setColumns(columns);
33     List<Object> list = new ArrayList<Object>();
34
35     /////////////////////////////////////////////////
36     //INI
37
38     //Crear un Statement para ejecutar la consulta SQL
39     cstmt = conn.createStatement();
40
41     //Ejecutar la consulta. La consulta usa la vista best_selling_zones
42     resultSet = cstmt.executeQuery(
43         "select zone_name, capital_town, climate, region, last_selling, total_quantity " +
44         "from best_selling_zones"
45     );
46     /* Realizamos la consulta sobre la vista creada mediante el script entregado:
47      * create view best_selling_zones as
48      * select z.zone_name,
49      * z.capital_town,
50      * z.climate,
51      * z.region,
52      * MAX(o.order_date) as last_selling,
53      * sum(o.quantity) AS total_quantity
54      * from zone z
55      * NATURAL JOIN wine w
56      * NATURAL JOIN order_line l
57      * NATURAL JOIN customer_order o
58      * group by z.zone_name, z.capital_town, z.climate, z.region
59      * order by total_quantity DESC
60      * LIMIT 5;
61
62 */
```

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise1/Exercise1PrintReportOverQuery.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Left Sidebar (Package Explorer):**
 - 20241ENU
 - src/main/java
 - edu.uoc.practica.bd.uocdb.exercise1
 - Exercise1PrintReportOverQuery.java
 - Exercise1Row.java
 - Exercise1PrintReportOverQuery.java
 - edu.uoc.practica.bd.uocdb.exercise2
 - FileUtilities.java
 - FormatUtilities.java
 - pruebaNoejava
 - ReportJava
 - DBAccessorLojandoEntreRegada.txt
 - src/main/resources
 - db.properties
 - exercise2.data
 - JRE System Library [JavaSE-23]
 - Project and External Dependencies
 - Central Area (Code Editor):** The code editor displays Java code for `Exercise1PrintReportOverQuery.java`.

```
//Crear una variable para guardar cada fila del resultado
Exercise1Row exercise1Row = null;

//Leer las filas del resultado una a una
while (resultSet.next()) {
    // Por cada fila obtener los valores de las columnas y crear un objeto Exercise1Row
    exercise1Row = new Exercise1Row(
        resultSet.getString("zone_name"),           // Columna zone_name
        resultSet.getString("capital_town"),         // Columna capital_town
        resultSet.getString("climate"),              // Columna climate
        resultSet.getString("region"),               // Columna region
        resultSet.getString("last_selling"),          // Columna last_selling
        resultSet.getLong("total_quantity")          // Columna total_quantity
    );

    // Agregar el objeto Exercise1Row a la lista
    list.add(exercise1Row);
}

//Verificar si se obtuvieron resultados
if (list.isEmpty()) {
    System.out.println("No se encontraron registros en la vista best_selling_zones.");
} else {
    // Llamar a printReport para mostrar la lista de resultados
    report.printReport(list);
}

//FIN
////////////////////////////////////////////////////////////////
}
catch (Exception e) {
    // Mostrar un mensaje si ocurre un error inesperado
}
```
 - Bottom Bar:** Problems Javadoc Declaration Console

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd.uocdb/exercise1/Exercise1PrintReportOverQuery.java - Eclipse IDE
- Toolbar:** Writable, Smart Insert, 16 : 29 : 403
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Left Sidebar (Package Explorer):**
 - 20241ENU
 - src/main/java
 - edu.uoc.practica.bd.uocdb.exercise1
 - Exercise1PrintReportOverQuery.java
 - Exercise1Row.java
 - Exercise1PrintReportOverQuery - copied 1
 - edu.uoc.practica.bd.uocdb.exercise2
 - edu.uoc.practica.bd.util
 - ArrayUtilities.java
 - Column.java
 - DBAccessor.java
 - DBAccessor
 - dbname
 - host
 - passwd
 - port
 - schema
 - user
 - getConnection(): Connection
 - init(): void
 - FileUtilities.java
 - FormatUtilities.java
 - pruebaNoe.java
 - Report.java
 - DBAccessor.LojandroEntregada.txt
 - src/main/resources
 - db.properties
 - exercise2.data
 - JRE System Library [JavaSE-23]
 - Project and External Dependencies
- Central Area:** Code editor showing Java code for `Exercise1PrintReportOverQuery.java`. The code handles database connections, statements, and result sets to print reports.
- Bottom Status Bar:** Problems, Javadoc, Declaration, Console

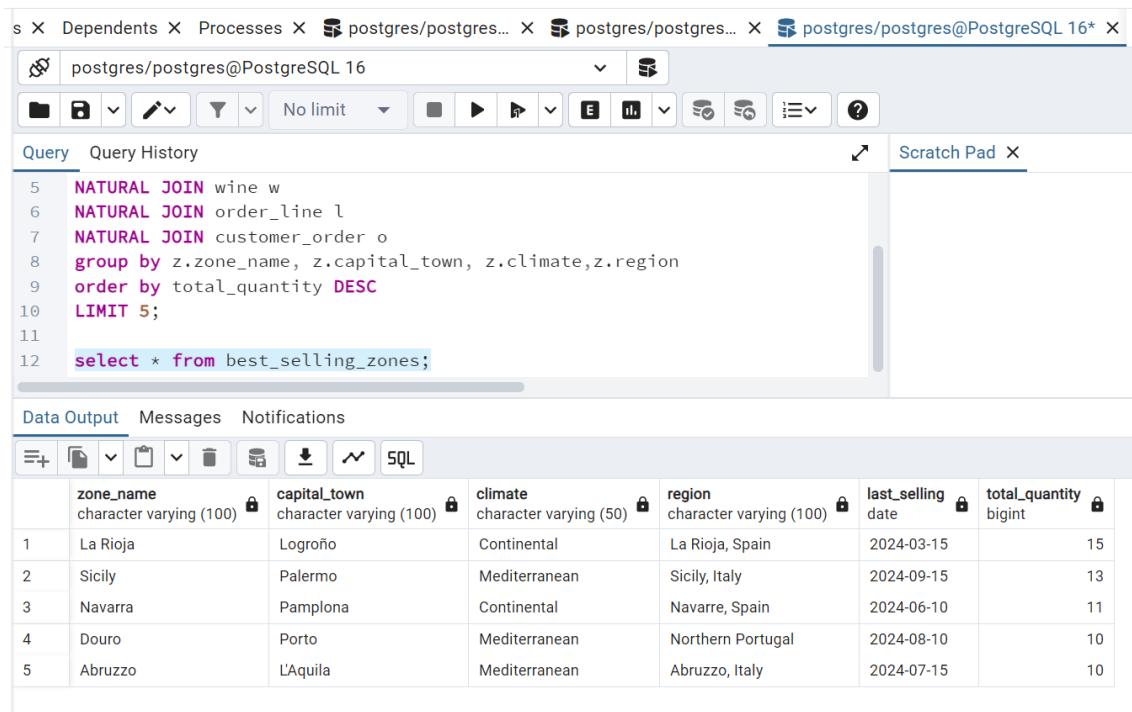
Se ha intentado comentar suficientemente el código para facilitar la corrección. Incluyo, de todos modos, por aquí una breve explicación:

En la clase Exercise1PrintReportOverQuery, se utilizó DBAccessor para conectar con la base de datos y realizar una consulta sobre la vista BEST_SELLING_ZONES. Los pasos específicos fueron:

1. Preparación de la Vista: Se ejecutó el script create_view.sql que define la vista BEST_SELLING_ZONES.
2. Consulta y Procesamiento de Resultados: Con la conexión proporcionada por DBAccessor, se utilizó un Statement para ejecutar la siguiente consulta SQL:
SELECT zone_name, capital_town, climate, region, last_selling, total_quantity
FROM best_selling_zones;
Los datos obtenidos se almacenaron en una lista de objetos Exercise1Row, representando cada fila de la consulta.
3. Presentación del Reporte: La lista resultante fue pasada a la clase Report para imprimir los datos en un formato tabular.
4. Gestión de Recursos y Errores: Se cerraron todos los recursos (Connection, Statement, ResultSet) en un bloque finally para evitar fugas de memoria. En caso de error, se muestra un mensaje genérico ERROR: List not available, cumpliendo con los requisitos del ejercicio.

Esta implementación permite realizar la consulta de manera eficiente y garantizar que la configuración y conexión a la base de datos sean consistentes en todo el programa.

Cuando lo ejecutemos, para saber si todo va bien deberíamos contrastar con lo que se muestra en BBDD directamente:



The screenshot shows the pgAdmin 4 interface. The top bar has tabs for 'Dependents', 'Processes', and three 'postgres/postgres@PostgreSQL 16*' sessions. The active session's toolbar includes icons for file operations, search, and connection management. Below the toolbar is a 'Query History' tab and a 'Scratch Pad' tab. The main area contains a SQL editor with the following code:

```
5  NATURAL JOIN wine w
6  NATURAL JOIN order_line l
7  NATURAL JOIN customer_order o
8  group by z.zone_name, z.capital_town, z.climate,z.region
9  order by total_quantity DESC
10 LIMIT 5;
11
12 select * from best_selling_zones;
```

Below the editor is a 'Data Output' tab, which is currently selected, showing a results grid. The grid has columns: zone_name, capital_town, climate, region, last_selling_date, and total_quantity. The data is as follows:

	zone_name	capital_town	climate	region	last_selling_date	total_quantity
1	La Rioja	Logroño	Continental	La Rioja, Spain	2024-03-15	15
2	Sicily	Palermo	Mediterranean	Sicily, Italy	2024-09-15	13
3	Navarra	Pamplona	Continental	Navarre, Spain	2024-06-10	11
4	Douro	Porto	Mediterranean	Northern Portugal	2024-08-10	10
5	Abruzzo	L'Aquila	Mediterranean	Abruzzo, Italy	2024-07-15	10

Procedemos a ejecutar nuestra clase Java:

The screenshot shows the Eclipse IDE interface with the following details:

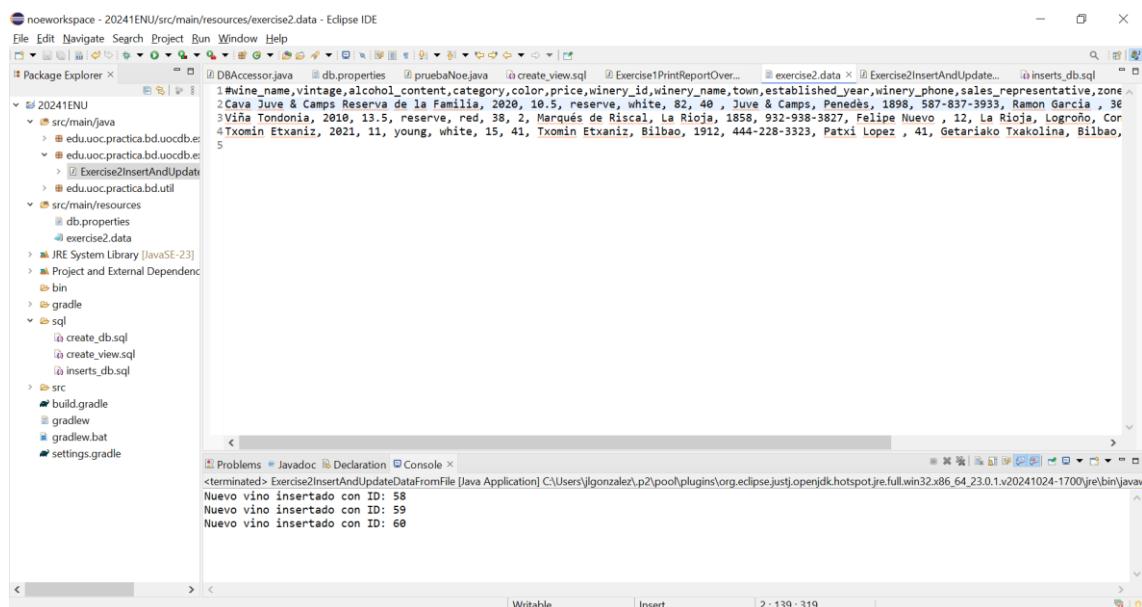
- Project Explorer:** Shows the project structure for "20241ENU".
- Editor:** Displays the code for `Exercise1PrintReportOverQuery.java`. The code reads from a database, processes the results, and prints them to the console.
- Console:** Shows the output of the program, which is a table of Spanish regions and their capitals.

```
DBAccessor.java db.properties pruebaNoe.java create_view.sql Exercise1PrintReportOverQuery.java
86         } else {
87             // Llamar a printReport para mostrar la lista de resultados
88             report.printReport(list);
89         }
90     } //FIN
91     /////////////////////////////////
92 }
93 catch (Exception e) {
94     // Mostrar un mensaje si ocurre un error inesperado
95     System.out.println("ERROR: List not available.\n");
96     e.printStackTrace();
97 }
98 finally {
99     // Cerrar todos los recursos abiertos
100    if (resultSet!=null)
101        try {
102            resultSet.close();
103        } catch(Exception ex) {}
104    if (cstmt!=null)
105
Zone Capital Climate Region Last selling Total
=====
La Rioja Logrono Continental La Rioja, Spain 2024-03-15 15
Sicily Palermo Mediterranean Sicily, Italy 2024-09-15 13
Navarra Pamplona Continental Navarre, Spain 2024-06-10 11
Douro Porto Mediterranean Northern Portugal 2024-08-10 18
Abruzzo L'Aquila Mediterranean Abruzzo, Italy 2024-07-15 18
```

Pregunta 2

En esta actividad, se desarrolló un programa en Java que permite procesar un archivo de datos y realizar operaciones de inserción y actualización sobre una base de datos PostgreSQL. El objetivo principal fue gestionar información relacionada con bodegas, zonas y vinos, actualizando los registros existentes o añadiendo nuevos cuando fuese necesario.

El archivo proporcionado, llamado exercise2.data, contiene información estructurada que se utiliza para sincronizar los datos de la base de datos. Este archivo incluye detalles como el identificador de la bodega, datos de contacto, información de las zonas geográficas y características de los vinos.



Se ha retocado el primer registro ya que se asociaba un vino a una zona equivocada.

Para abordar esta actividad, se emplearon conceptos clave de programación en Java con JDBC, incluyendo:

- Uso de transacciones (commit y rollback) para garantizar la consistencia de los datos.
- Sentencias preparadas (PreparedStatement) para mejorar la seguridad y el rendimiento de las operaciones.
- Manejo de excepciones para controlar errores durante la ejecución del programa y garantizar la estabilidad del sistema.

Se nos proporciona una clase Exercise2InsertAndUpdateDataFromFile a modo de guía con una serie de funcionalidades ya implementadas, como son el tratamiento del fichero de entrada y una serie de métodos auxiliares que utilizaremos para las sentencias sql.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "20241ENU".
- Code Editor:** Displays Java code for `DBAccessor.java`. The code handles database connections, transaction management, and prepared statements for inserting/updating data from a file.
- Bottom Status Bar:** Shows tabs for "Problems", "Javadoc", "Declaration", and "Console". The active tab is "edu.uoc.practica.bd.uocdb.exercise2.Exercise2InsertAndUpdateDataFromFile".
- Bottom Navigation:** Includes buttons for "Writable", "Smart Insert", and the current time "18:50 : 529".

```
43     }
44 }
45 DBAccessor dbaccessor = new DBAccessor();
46 dbaccessor.init();
47 Connection conn = dbaccessor.getConnection();
48
49 if (conn == null) {
50     return;
51 }
52
53 // TODO Prepare everything before updating or inserting
54
55 try {
56     // TODO Update or insert the wine, winery and zone from every row in file
57
58     // TODO Validate transaction
59
60     // TODO Close resources and check exceptions
61     finally {
62
63     }
64
65
66
67
68 private void setPSUpdateWinery(PreparedStatement updateStatement, List<String> row)
69 throws SQLException {
70     String[] rowArray = (String[]) row.toArray(new String[0]);
71
72     setValueOrNull(updateStatement, 1, getValueIfNotNull(rowArray, 10)); // winery_phone
73     setValueOrNull(updateStatement, 2, getValueIfNotNull(rowArray, 11)); // salesRepresentative
74     setValueOrNull(updateStatement, 3,
```

El código al que he llegado es el siguiente:

The screenshot shows the Eclipse IDE interface with the following details:

- File Bar:** File, Edit, Source, Refactor, Navigate, Project, Run, Window, Help.
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure:
 - 20241ENU
 - src/main/java
 - edu.uoc.practica.bd.uocdb.exercise1
 - edu.uoc.practica.bd.uocdb.exercise2
 - Exercise2InsertAndUpdateDataFromFile.java
 - edu.uoc.practica.bd.util
 - src/main/resources
 - db.properties
 - exercise2.data
 - JRE System Library [JavaSE-23]
 - Project and External Dependencies
 - bin
 - gradle
 - sql
 - create_db.sql
 - create_view.sql
 - inserts_db.sql
 - src
 - build.gradle
 - gradlew
 - gradlew.bat
 - settings.gradle
- Editor:** Displays the content of `Exercise2InsertAndUpdateDataFromFile.java`. The code reads data from a file named `exercise2.data` and inserts it into a database table.
- Bottom Bar:** Problems, Javadoc, Declaration, Console.
- Status Bar:** Writable, Smart Insert, 28:25 - 761.

noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE

```

private void run() {
    List<String> fileContents = null;
    try {
        fileContents = fileUtilities.readFileFromClasspath("exercise2.data");
    } catch (FileNotFoundException e) {
        System.err.println("Error: El archivo de datos no se encontro.");
    } catch (IOException e) {
        System.err.println("Error: Ocurrio un problema de entrada/salida al leer el archivo.");
        e.printStackTrace();
    }
    if (fileContents == null) {
        return;
    }
    DBAccessor dbaccessor = new DBAccessor();
    dbaccessor.init();
    Connection conn = dbaccessor.getConnection();
    if (conn == null) {
        return;
    }
    ////////////////////////////////////////////////////////////////////INI
    try {
        // Iniciamos una transaccion
        conn.setAutoCommit(false);
        // Definimos las consultas SQL para las operaciones necesarias
        String sqlCheckWinery = "select * from winery where winery_id = ?";
        String sqlUpdateWinery = "update winery set winery_phone = ?, sales_representative = ? where winery_id = ?";
        String sqlInsertWinery = "insert into winery (winery_id, winery_name, town, established_year, winery_phone, sales_representative) values (?, ?, ?, ?, ?, ?)";
        String sqlSelectZone = "select * from zone where zone_id = ?";
        String sqlInsertZone = "insert into zone (zone_id, zone_name, capital_town, climate, region) values (?, ?, ?, ?, ?)";
        String sqlInsertWine = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id, stock) values (?, ?, ?, ?, ?, ?, ?, ?)";
        String sqlInsertWineWithGeneratedKeys = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id, stock) values (?, ?, ?, ?, ?, ?, ?, ?) returning generated_key";
        PreparedStatement checkWineryStmt = conn.prepareStatement(sqlCheckWinery);
        PreparedStatement updateWineryStmt = conn.prepareStatement(sqlUpdateWinery);
        PreparedStatement insertWineryStmt = conn.prepareStatement(sqlInsertWinery);
        PreparedStatement selectZoneStmt = conn.prepareStatement(sqlSelectZone);
        PreparedStatement insertZoneStmt = conn.prepareStatement(sqlInsertZone);
        PreparedStatement insertWineStmt = conn.prepareStatement(sqlInsertWine);
        PreparedStatement insertWineWithGeneratedKeysStmt = conn.prepareStatement(sqlInsertWineWithGeneratedKeys);
        // Iteraremos sobre cada fila del archivo de datos para procesar la informacion
        for (List<String> datosFila : fileContents) {
            checkWineryStmt.setString(1, datosFila.get(0));
            updateWineryStmt.setString(1, datosFila.get(1));
            updateWineryStmt.setString(2, datosFila.get(2));
            insertWineryStmt.setString(1, datosFila.get(3));
            insertWineryStmt.setString(2, datosFila.get(4));
            insertWineryStmt.setString(3, datosFila.get(5));
            insertWineryStmt.setString(4, datosFila.get(6));
            insertWineryStmt.setString(5, datosFila.get(7));
            insertWineryStmt.setString(6, datosFila.get(8));
            insertWineryStmt.setString(7, datosFila.get(9));
            selectZoneStmt.setString(1, datosFila.get(10));
            insertZoneStmt.setString(1, datosFila.get(11));
            insertZoneStmt.setString(2, datosFila.get(12));
            insertZoneStmt.setString(3, datosFila.get(13));
            insertZoneStmt.setString(4, datosFila.get(14));
            insertZoneStmt.setString(5, datosFila.get(15));
            insertWineStmt.setString(1, datosFila.get(16));
            insertWineStmt.setString(2, datosFila.get(17));
            insertWineStmt.setString(3, datosFila.get(18));
            insertWineStmt.setString(4, datosFila.get(19));
            insertWineStmt.setString(5, datosFila.get(20));
            insertWineStmt.setString(6, datosFila.get(21));
            insertWineStmt.setString(7, datosFila.get(22));
            insertWineWithGeneratedKeysStmt.setString(1, datosFila.get(23));
            insertWineWithGeneratedKeysStmt.setString(2, datosFila.get(24));
            insertWineWithGeneratedKeysStmt.setString(3, datosFila.get(25));
            insertWineWithGeneratedKeysStmt.setString(4, datosFila.get(26));
            insertWineWithGeneratedKeysStmt.setString(5, datosFila.get(27));
            insertWineWithGeneratedKeysStmt.setString(6, datosFila.get(28));
            insertWineWithGeneratedKeysStmt.setString(7, datosFila.get(29));
            insertWineWithGeneratedKeysStmt.setString(8, datosFila.get(30));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE

```

try {
    // Iniciamos una transaccion
    conn.setAutoCommit(false);
    // Definimos las consultas SQL para las operaciones necesarias
    String sqlCheckWinery = "select * from winery where winery_id = ?";
    String sqlUpdateWinery = "update winery set winery_phone = ?, sales_representative = ? where winery_id = ?";
    String sqlInsertWinery = "insert into winery (winery_id, winery_name, town, established_year, winery_phone, sales_representative) values (?, ?, ?, ?, ?, ?)";
    String sqlSelectZone = "select * from zone where zone_id = ?";
    String sqlInsertZone = "insert into zone (zone_id, zone_name, capital_town, climate, region) values (?, ?, ?, ?, ?)";
    String sqlInsertWine = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id, stock) values (?, ?, ?, ?, ?, ?, ?, ?)";
    String sqlInsertWineWithGeneratedKeys = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id, stock) values (?, ?, ?, ?, ?, ?, ?, ?) returning generated_key";
    PreparedStatement checkWineryStmt = conn.prepareStatement(sqlCheckWinery);
    PreparedStatement updateWineryStmt = conn.prepareStatement(sqlUpdateWinery);
    PreparedStatement insertWineryStmt = conn.prepareStatement(sqlInsertWinery);
    PreparedStatement selectZoneStmt = conn.prepareStatement(sqlSelectZone);
    PreparedStatement insertZoneStmt = conn.prepareStatement(sqlInsertZone);
    PreparedStatement insertWineStmt = conn.prepareStatement(sqlInsertWine);
    PreparedStatement insertWineWithGeneratedKeysStmt = conn.prepareStatement(sqlInsertWineWithGeneratedKeys);
    // Iteraremos sobre cada fila del archivo de datos para procesar la informacion
    for (List<String> datosFila : fileContents) {
        checkWineryStmt.setString(1, datosFila.get(0));
        updateWineryStmt.setString(1, datosFila.get(1));
        updateWineryStmt.setString(2, datosFila.get(2));
        insertWineryStmt.setString(1, datosFila.get(3));
        insertWineryStmt.setString(2, datosFila.get(4));
        insertWineryStmt.setString(3, datosFila.get(5));
        insertWineryStmt.setString(4, datosFila.get(6));
        insertWineryStmt.setString(5, datosFila.get(7));
        insertWineryStmt.setString(6, datosFila.get(8));
        insertWineryStmt.setString(7, datosFila.get(9));
        selectZoneStmt.setString(1, datosFila.get(10));
        insertZoneStmt.setString(1, datosFila.get(11));
        insertZoneStmt.setString(2, datosFila.get(12));
        insertZoneStmt.setString(3, datosFila.get(13));
        insertZoneStmt.setString(4, datosFila.get(14));
        insertZoneStmt.setString(5, datosFila.get(15));
        insertWineStmt.setString(1, datosFila.get(16));
        insertWineStmt.setString(2, datosFila.get(17));
        insertWineStmt.setString(3, datosFila.get(18));
        insertWineStmt.setString(4, datosFila.get(19));
        insertWineStmt.setString(5, datosFila.get(20));
        insertWineStmt.setString(6, datosFila.get(21));
        insertWineStmt.setString(7, datosFila.get(22));
        insertWineWithGeneratedKeysStmt.setString(1, datosFila.get(23));
        insertWineWithGeneratedKeysStmt.setString(2, datosFila.get(24));
        insertWineWithGeneratedKeysStmt.setString(3, datosFila.get(25));
        insertWineWithGeneratedKeysStmt.setString(4, datosFila.get(26));
        insertWineWithGeneratedKeysStmt.setString(5, datosFila.get(27));
        insertWineWithGeneratedKeysStmt.setString(6, datosFila.get(28));
        insertWineWithGeneratedKeysStmt.setString(7, datosFila.get(29));
        insertWineWithGeneratedKeysStmt.setString(8, datosFila.get(30));
    }
} catch (SQLException e) {
    e.printStackTrace();
}

```

noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE

```

try {
    // Iniciamos una transaccion
    conn.setAutoCommit(false);
    // Definimos las consultas SQL para las operaciones necesarias
    String sqlCheckWinery = "select * from winery where winery_id = ?";
    String sqlUpdateWinery = "update winery set winery_phone = ?, sales_representative = ? where winery_id = ?";
    String sqlInsertWinery = "insert into winery (winery_id, winery_name, town, established_year, winery_phone, sales_representative) values (?, ?, ?, ?, ?, ?)";
    String sqlSelectZone = "select * from zone where zone_id = ?";
    String sqlInsertZone = "insert into zone (zone_id, zone_name, capital_town, climate, region) values (?, ?, ?, ?, ?)";
    String sqlInsertWine = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id, stock) values (?, ?, ?, ?, ?, ?, ?, ?)";
    String sqlInsertWineWithGeneratedKeys = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id, stock) values (?, ?, ?, ?, ?, ?, ?, ?) returning generated_key";
    PreparedStatement checkWineryStmt = conn.prepareStatement(sqlCheckWinery);
    PreparedStatement updateWineryStmt = conn.prepareStatement(sqlUpdateWinery);
    PreparedStatement insertWineryStmt = conn.prepareStatement(sqlInsertWinery);
    PreparedStatement selectZoneStmt = conn.prepareStatement(sqlSelectZone);
    PreparedStatement insertZoneStmt = conn.prepareStatement(sqlInsertZone);
    PreparedStatement insertWineStmt = conn.prepareStatement(sqlInsertWine);
    PreparedStatement insertWineWithGeneratedKeysStmt = conn.prepareStatement(sqlInsertWineWithGeneratedKeys);
    // Iteraremos sobre cada fila del archivo de datos para procesar la informacion
    for (List<String> datosFila : fileContents) {
        checkWineryStmt.setString(1, datosFila.get(0));
        updateWineryStmt.setString(1, datosFila.get(1));
        updateWineryStmt.setString(2, datosFila.get(2));
        insertWineryStmt.setString(1, datosFila.get(3));
        insertWineryStmt.setString(2, datosFila.get(4));
        insertWineryStmt.setString(3, datosFila.get(5));
        insertWineryStmt.setString(4, datosFila.get(6));
        insertWineryStmt.setString(5, datosFila.get(7));
        insertWineryStmt.setString(6, datosFila.get(8));
        insertWineryStmt.setString(7, datosFila.get(9));
        selectZoneStmt.setString(1, datosFila.get(10));
        insertZoneStmt.setString(1, datosFila.get(11));
        insertZoneStmt.setString(2, datosFila.get(12));
        insertZoneStmt.setString(3, datosFila.get(13));
        insertZoneStmt.setString(4, datosFila.get(14));
        insertZoneStmt.setString(5, datosFila.get(15));
        insertWineStmt.setString(1, datosFila.get(16));
        insertWineStmt.setString(2, datosFila.get(17));
        insertWineStmt.setString(3, datosFila.get(18));
        insertWineStmt.setString(4, datosFila.get(19));
        insertWineStmt.setString(5, datosFila.get(20));
        insertWineStmt.setString(6, datosFila.get(21));
        insertWineStmt.setString(7, datosFila.get(22));
        insertWineWithGeneratedKeysStmt.setString(1, datosFila.get(23));
        insertWineWithGeneratedKeysStmt.setString(2, datosFila.get(24));
        insertWineWithGeneratedKeysStmt.setString(3, datosFila.get(25));
        insertWineWithGeneratedKeysStmt.setString(4, datosFila.get(26));
        insertWineWithGeneratedKeysStmt.setString(5, datosFila.get(27));
        insertWineWithGeneratedKeysStmt.setString(6, datosFila.get(28));
        insertWineWithGeneratedKeysStmt.setString(7, datosFila.get(29));
        insertWineWithGeneratedKeysStmt.setString(8, datosFila.get(30));
    }
} catch (SQLException e) {
    e.printStackTrace();
}

```

```

noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE
File Edit Source Refactor Navigate Project Run Window Help
Package Explorer DBAccessor.java db.properties pruebaNoejava create_view.sql ExercisePrintReportOverQu... exercise2.data Exercise2InsertAndUpdateData...
20241ENU
src/main/java
edu.uoc.practica.bd.uocdb.exercise1
edu.uoc.practica.bd.uocdb.exercise2
Exercise2InsertAndUpdateDataFromFile
src/main/resources
db.properties
exercise2.data
JRE System Library [JavaSE-23]
Project and External Dependencies
bin
gradle
sql
create_db.sql
create_view.sql
inserts_db.sql
src
build.gradle
gradlew
gradlew.bat
settings.gradle

109         insertWineStmt.executeUpdate(); // Ejecutamos el INSERT
110         try (ResultSet generatedKeys = insertWineStmt.getGeneratedKeys()) {
111             if (generatedKeys.next()) {
112                 // Mostramos el ID del nuevo vino insertado
113                 System.out.println("Nuevo vino insertado con ID: " + generatedKeys.getInt(1));
114             } else {
115                 throw new SQLException("No se pudo recuperar el ID del vino insertado.");
116             }
117         }
118
119         // Confirmamos los cambios realizados en la base de datos
120         conn.commit();
121     } catch (Exception e) {
122         // Si ocurre un error, revertimos todos los cambios realizados en la transaccion
123         System.err.println("Error: ocurrio un problema durante la transaccion. se revirtieron los cambios.");
124         try {
125             conn.rollback();
126         } catch (Exception ex) {
127             System.err.println("Error: fallo al intentar revertir los cambios.");
128         }
129     } finally {
130         // Cerramos todos los recursos para liberar memoria y evitar problemas
131         try {
132             if (conn != null) conn.close();
133         } catch (Exception ex) {
134             System.err.println("Error: no se pudieron cerrar correctamente los recursos.");
135         }
136     }
137
138     //FIN
139     ///////////////////////////////////////////////////////////////////
140
141 }

136     }
137
138     //FIN
139     ///////////////////////////////////////////////////////////////////
140
141
142     private void setPSCheckWinery(PreparedStatement stmt, List<String> row) throws SQLException {
143
144     private void setPSUpdateWinery(PreparedStatement stmt, List<String> row) throws SQLException {
145
146     private void setPSInsertWinery(PreparedStatement stmt, List<String> row) throws SQLException {
147
148     private void setPSSelectZone(PreparedStatement stmt, List<String> row) throws SQLException {
149
150     private void setPSInsertZone(PreparedStatement stmt, List<String> row) throws SQLException {
151
152     private void setPSInsertWine(PreparedStatement stmt, List<String> row) throws SQLException {
153
154     private Integer getIntegerFromStringOrNull(String integer) {
155
156     private Double getDoubleFromStringOrNull(String double) {
157
158     private String getStringIfNotNull(String[] rowArray, int index) {
159
160     private void setValueOrNull(PreparedStatement stmt, int parameterIndex, Integer value) throws SQLException {
161
162     private void setValueOrNull(PreparedStatement stmt, int parameterIndex, Double value) throws SQLException {
163
164     private void setValueOrNull(PreparedStatement stmt, int parameterIndex, String value) throws SQLException {
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227

```

Comento nuevamente lo más relevante para facilitar la corrección:

- Inicio de la Transacción: Antes de procesar el archivo, se configura la conexión para trabajar en modo transaccional (conn.setAutoCommit(false)), lo que permite agrupar las operaciones como una unidad atómica. Esto garantiza que, en caso de error, todos los cambios realizados hasta ese punto puedan deshacerse mediante un rollback.
- Preparación de Sentencias SQL:
Se definieron varias sentencias SQL en forma de PreparedStatement para las diferentes operaciones necesarias:
 - o Verificación de registros existentes: SELECT para comprobar si una bodega o una zona ya existe.
 - o Actualización de registros existentes: UPDATE para modificar datos de contacto o representantes de ventas en bodegas existentes.
 - o Inserción de nuevos registros: INSERT para agregar nuevas bodegas, zonas y vinos con un stock inicial de 0.

- Procesamiento de Datos:
El programa itera sobre cada fila del archivo exercise2.data y realiza las siguientes acciones:
 - o Bodega:
Comprueba si la bodega ya existe mediante un SELECT.
Si existe, actualiza los datos correspondientes con un UPDATE.
Si no existe, inserta una nueva bodega con un INSERT.
 - o Zona:
Comprueba si la zona ya existe mediante un SELECT.
Si no existe, la inserta en la base de datos con un INSERT.
 - o Vino:
Inserta un nuevo registro de vino con un stock inicial de 0 utilizando un INSERT.
Si se inserta correctamente, muestra el nuevo identificador del vino generado automáticamente.
- Gestión de Errores: En caso de que ocurra un error durante el procesamiento, se captura la excepción y se revierte la transacción completa con un rollback. Esto asegura que la base de datos quede en un estado consistente, sin cambios parciales.
- Confirmación de la Transacción: Si todas las operaciones se completan sin errores, se confirman los cambios mediante un commit, consolidando las modificaciones en la base de datos.
- Liberación de Recursos Finalmente, se cierran todos los recursos abiertos, incluidos los PreparedStatement y la conexión (Connection), para liberar memoria y evitar fugas de recursos.

Las sentencias sql utilizadas en este ejercicio son:

- o Verificar si una bodega ya existe
SELECT * FROM winery WHERE winery_id = ?;
- o Actualizar datos de una bodega existente
UPDATE winery
SET winery_phone = ?, salesRepresentative = ?
WHERE winery_id = ?;
- o Insertar una nueva bodega
INSERT INTO winery (winery_id, winery_name, town, establishedYear,
winery_phone, salesRepresentative)
VALUES (?, ?, ?, ?, ?, ?);
- o Verificar si una zona ya existe
SELECT * FROM zone WHERE zone_id = ?;
- o Insertar una nueva zona
INSERT INTO zone (zone_id, zone_name, capitalTown, climate, region)
VALUES (?, ?, ?, ?, ?);
- o Insertar un nuevo
INSERT INTO wine (wine_name, vintage, alcoholContent, category, color,
winery_id, zone_id, price, stock)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, 0);

Para demostrar el buen funcionamiento, se ejecuta la clase

```

121     conn.commit();
122 } catch (Exception e) {
123     // Si ocurre un error, revertimos todos los cambios realizados en la transaccion
124     System.err.println("Error: ocurrio un problema durante la transaccion. se revirtieron los cambios.");
125     try {
126         conn.rollback();
127     } catch (Exception ex) {
128         System.err.println("Error: fallo al intentar revertir los cambios.");
129     }
130 } finally {
131     // Cerramos todos los recursos para liberar memoria y evitar problemas
132     try {
133         if (conn != null) conn.close();
134     } catch (Exception ex) {
135         System.err.println("Error: no se pudieron cerrar correctamente los recursos.");
136     }
137 }
138

```

Problems Javadoc Declaration Console
terminated> Exercise2InsertAndUpdateDataFromFile [Java Application]
Nuevo vino insertado con ID: 58
Nuevo vino insertado con ID: 59
Nuevo vino insertado con ID: 60

Si pasamos a revisar la base de datos, vemos que los vinos se insertaron:

```

1 set search_path to ubd_20241;
2 select * from wine where wine_id in (58,59,60);
3
4
5
6
7
8
9

```

wine_id	wine_name	vintage	alcohol_content	category	color	winery_id	zone_id	st
1	Cava Juve & Camps Reserva de la Familia	2020	10.50	reserve	white	40	30	
2	Viña Tondonia	2010	13.50	reserve	red	2	12	
3	Txomin Etxaniz	2021	11.00	young	white	41	41	

En lo que respecta a la bodegas afectadas, vamos a ver que datos han quedado tras la ejecución:

```

1 set search_path to ubd_20241;
2 select * from wine where wine_id in (58,59,60);
3
4 select * from winery where winery_id in (40,41,2);
5
6
7
8
9

```

winery_id	winery_name	town	established_year	winery_phone	salesRepresentative
1	Juve & Camps	Penedès	1898	587-837-3933	Ramon Garcia
2	Marqués de Riscal	La Rioja	1858	932-938-3827	Felipe Nuevo
3	Txomin Etxaniz	Bilbao	1912	444-228-3323	Patxi Lopez

La bodega 2, tal y como se ve en el fichero de inserts, existía

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists project files like DBAccessor.java, db.properties, pruebaNoe.java, create_view.sql, Exercise1PrintReport.java, exercise2.data, Exercise2InsertAndUpdateDataFromFil, and inserts_db.sql. The main editor window displays the contents of inserts_db.sql, which contains several INSERT statements for the WINERY and ZONE tables. The bottom status bar indicates the file is named inserts_db.sql.

```

noeworkspace - 20241ENU/sql/inserts_db.sql - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
Package Explorer X DBAccessor.java db.properties pruebaNoe.java create_view.sql Exercise1PrintReport.java exercise2.data Exercise2InsertAndU... inserts_db.sql X
20241ENU
src/main/java
  edu.uoc.practica.bd.uocdb.exercise1
  edu.uoc.practica.bd.uocdb.exercise2
    Exercise2InsertAndUpdateDataFromFil
    edu.uoc.practica.bd.util
src/main/resources
  db.properties
  exercise2.data
JRE System Library [javaSE-23]
Project and External Dependencies
bin
gradle
sql
  create_db.sql
  create_view.sql
  inserts_db.sql
src
build.gradle
gradlew
gradlew.bat
settings.gradle
Problems Javadoc Declaration Console
edu.uoc.practica.bd.uocdb.exercise2.Exercise2InsertAndUpdateDataFromFil

```

Y como se ve se ha actualizado los datos de teléfono y salesRepresentative.

Las bodegas 40 y 41 se insertaron correctamente, ya que no estaban incluidas en el fichero de inserts de creación.

En cuanto a las zonas, tras la ejecución de mi código solución queda así:

The screenshot shows the pgAdmin 4 interface connected to a PostgreSQL 16 database. The left sidebar shows the server tree with databases like postgres, public, and ubd_20241. The main area has a query editor with the following SQL code:

```

set search_path to ubd_20241;
select * from wine where wine_id in (58,59,60);
select * from winery where winery_id in (40,41,2);
select * from zone where zone_id in (30,12,41);

```

The results pane shows a table with the following data:

zone_id	zone_name	capital_town	climate	region
12	La Rioja	Logroño	Continental	La Rioja, Spain
30	Penedès	Vilafranca del Penedès	Mediterranean	Catalonia, Spain
41	Getariako Txakolina	Bilbao	Oceanic	Euskadi Spain

Total rows: 3 of 3 Query complete 00:00:00.133 Ln 6, Col 1 Successfully run. Total query runtime: 133 msec. 3 rows affected.

Para finalizar, se va a modificar el código para mostrar por consola cada una de las acciones sobre bbdd y que las ejecuciones de prueba que puedan llevarse a cabo para la corrección puedan ser más sencillas.

Recreamos la base de datos y volvemos a ejecutar y verificando el código:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the project structure for "20241ENU".
- Code Editor:** Shows a Java file named "DBAccess.java" containing code for database operations. The code includes:
 - Comments explaining the steps: "1. Insertamos una nueva zona", "2. Verificamos si la zona ya existe", and "3. Insertamos un nuevo vino".
 - SQL statements using PreparedStatement: `setPSSelectZone`, `setPSInsertZone`, and `setPSInsertWine`.
 - Java code for handling results and outputting to System.out.
- Console:** Shows the output of the executed SQL queries, including:
 - "Alta winery con winery_id 48"
 - "Nuevo vino insertado con ID: 58"
 - "Registro a tratar Cava Juve & Camps Reserva de la Familia"
 - "Alta winery con winery_id 49"
 - "Nuevo vino insertado con ID: 59"
 - "Registro a tratar Víno Tondonia"
 - "Actualización winery con winery_id 2"
 - "Nuevo vino insertado con ID: 59"
 - "Registro a tratar Txomin Etxaniz"
 - "Alta winery con winery_id 41"
 - "Alta zone con zone_id 41"
 - "Nuevo vino insertado con ID: 60"
- Bottom Status Bar:** Shows the path "edu.uoc.practica.bd.uocdb.exercise2.Exercise2InsertAndUpdateDataFromFile.java - 20241ENU/src/main/java".