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.

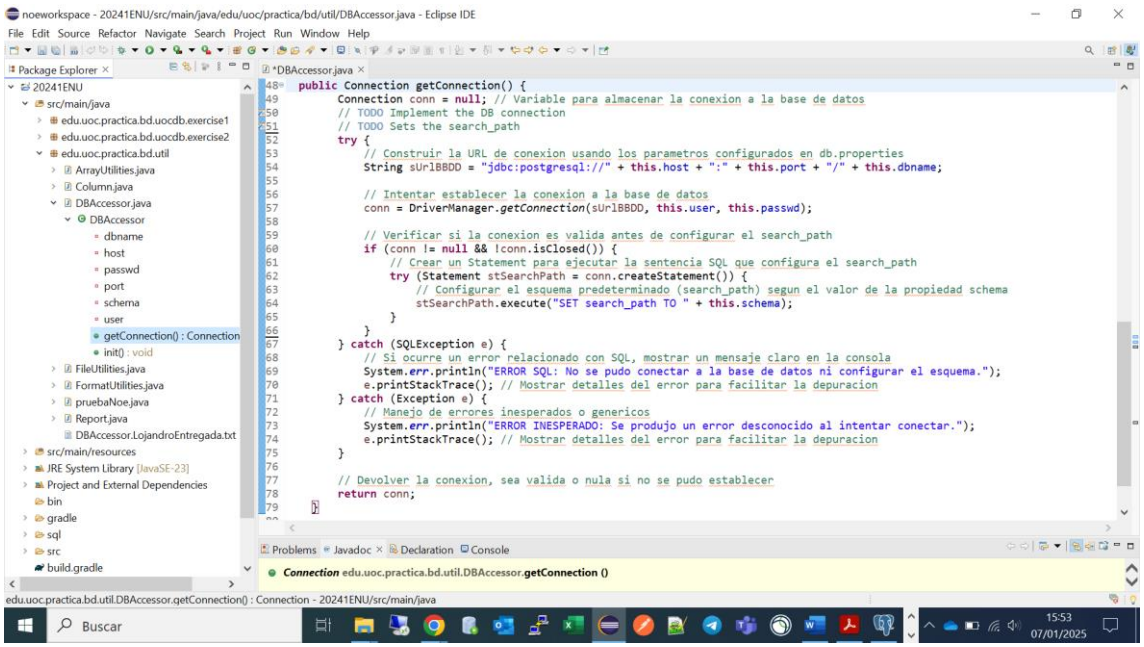
Por un lado se realiza la carga de Configuración desde db.properties. La clase lee los parámetros necesarios para la conexión (como host, port, dbname, user, passwd y schema) desde el archivo db.properties utilizando la clase Properties.

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.

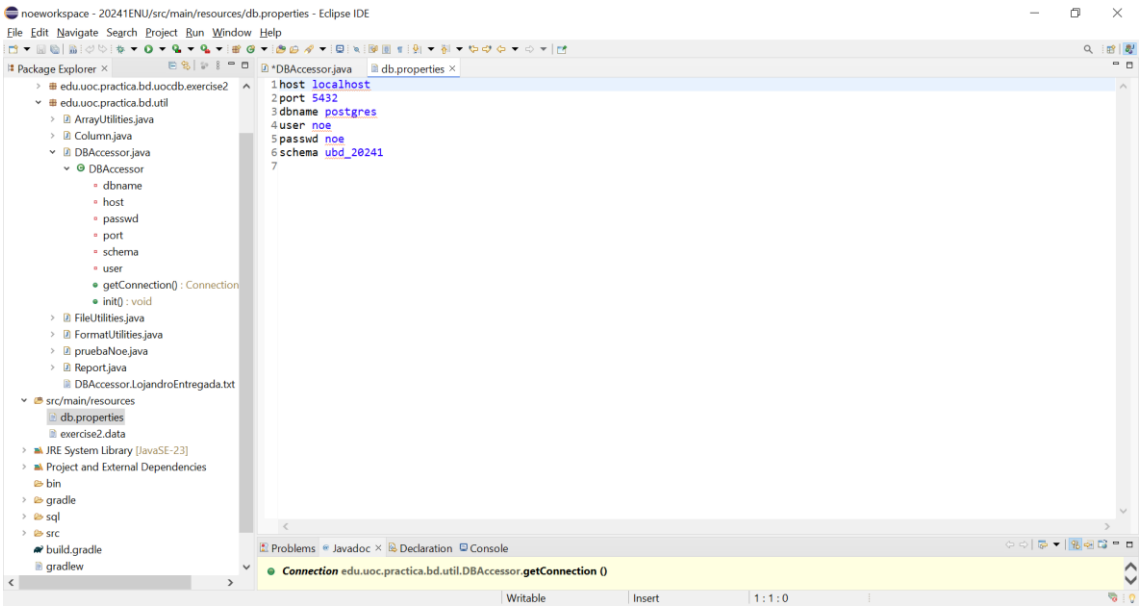
Lleva a cabo la configuración del search_path, ya que una vez establecida la conexión, se ha de ejecutar un comando SQL (SET search_path TO ...) para configurar el esquema predeterminado.

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

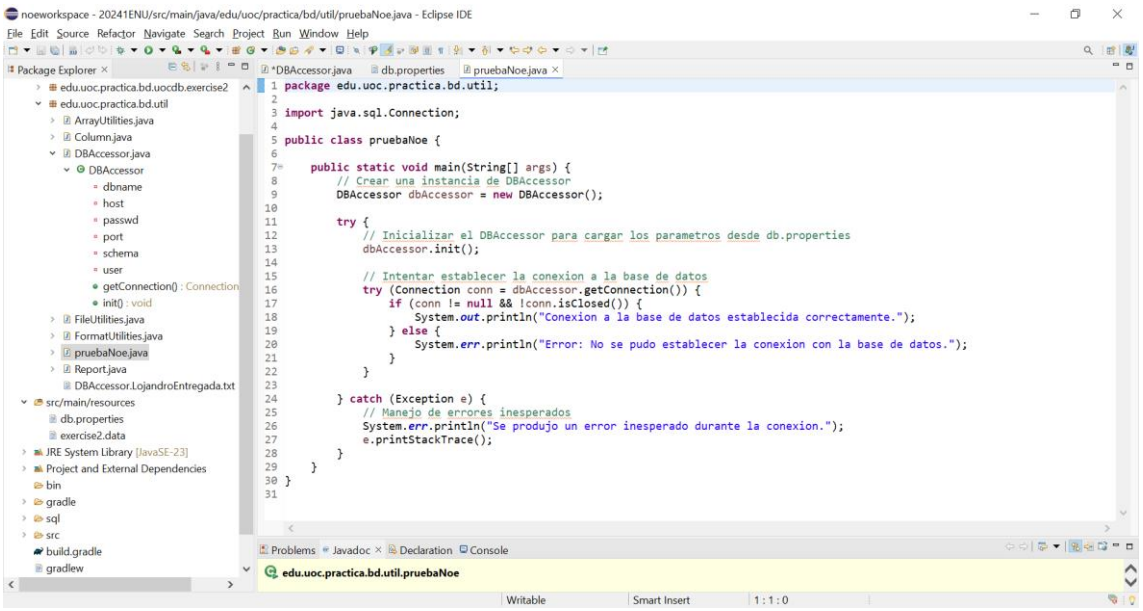
Código resultante, método getConnection:

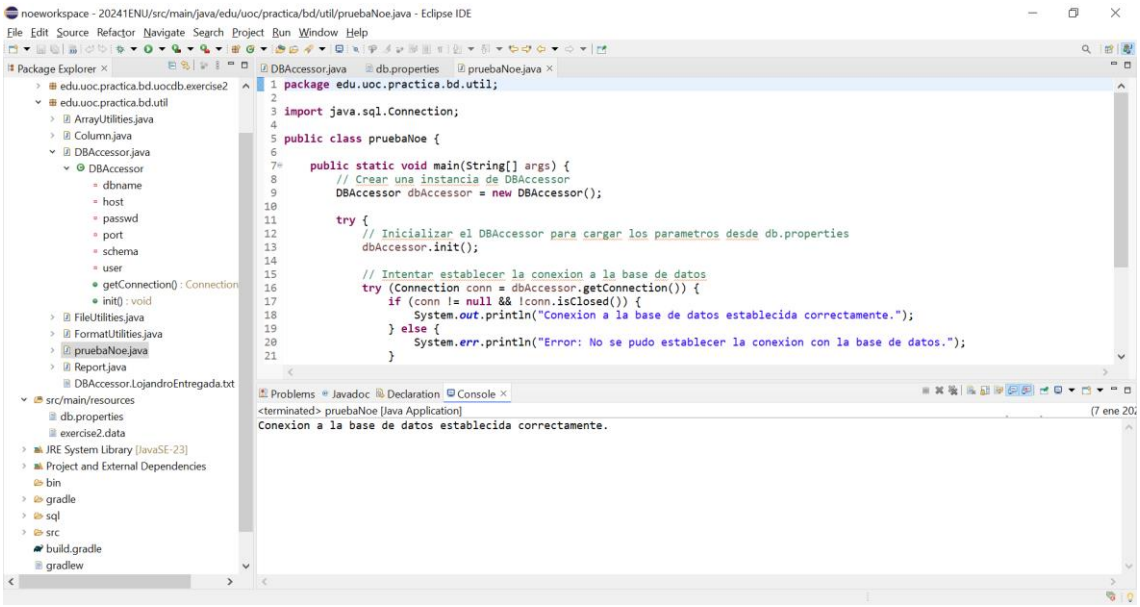


El archivo properties tiene el siguiente contenido:

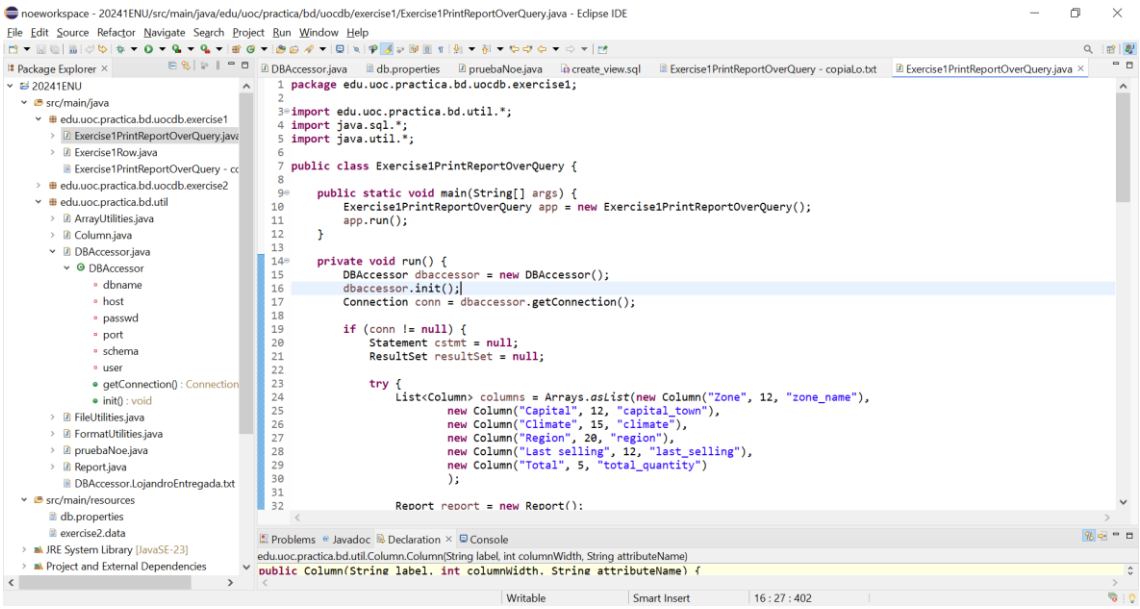


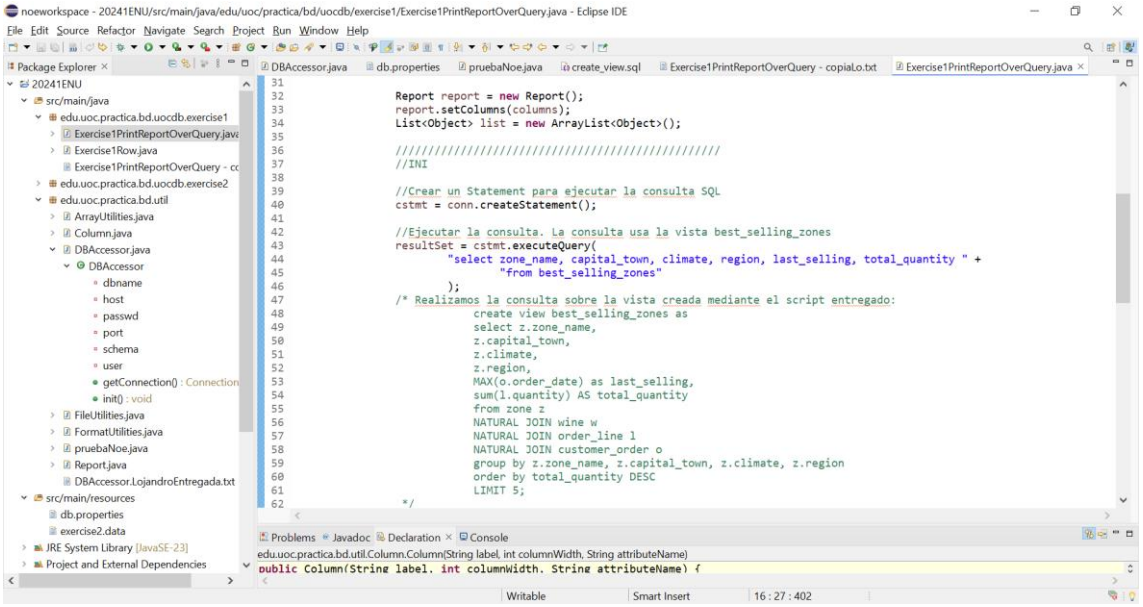
Para validar de manera unitaria este componente he implementado una clase de pruebas que únicamente realiza un getConnection y vemos su resultado:

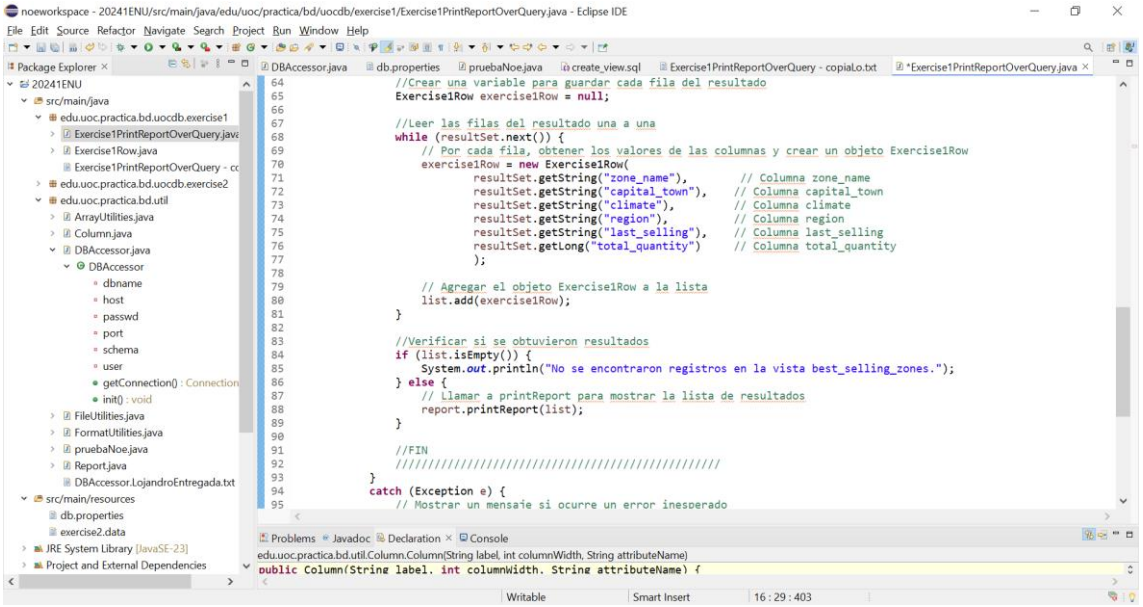


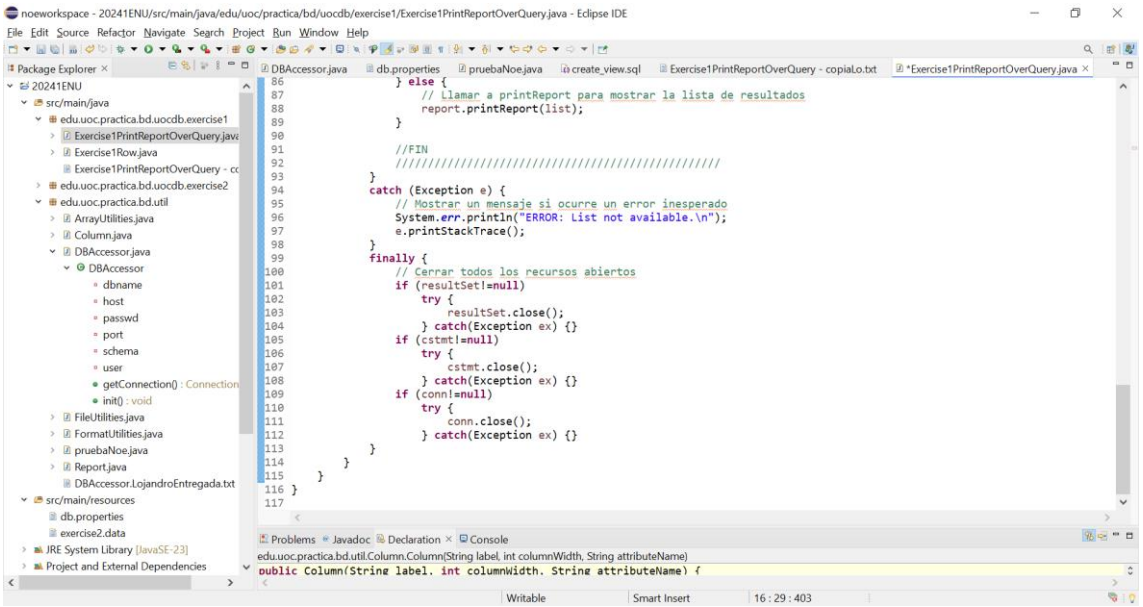


La solución de lo que quedaba pendiente de rellenar a la que se ha llegado es la siguiente:




noeworkspace - 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
DBAccessor.java
db.properties
pruebaNoe.java
create_view.sql
Exercise1PrintReportOverQuery - copiaLo.txt
Exercise1PrintReportOverQuery.java
Report report = new Report();
report.setColumns(columns);
List<Object> list = new ArrayList<Object>();
//INI
//Crear un Statement para ejecutar la consulta SQL
cstmt = conn.createStatement();
//Ejecutar la consulta. La consulta usa la vista best_selling_zones
resultSet = cstmt.executeQuery(
"select zone_name, capital_town, climate, region, last_selling, total_quantity " +
"from best_selling_zones"
);
/* Realizamos la consulta sobre la vista creada mediante el script entregado:
create view best_selling_zones as
select z.zone_name,
z.capital_town,
z.climate,
z.region,
MAX(o.order_date) as last_selling,
sum(l.quantity) AS total_quantity
from zone z
NATURAL JOIN wine w
NATURAL JOIN order_line l
NATURAL JOIN customer_order o
group by z.zone_name, z.capital_town, z.climate, z.region
order by total_quantity DESC
LIMIT 5;
*/
Problems Javadoc Declaration Console
edu.uoc.practica.bd.util.Column.Column(String label, int columnWidth, String attributeName)
public Column(String label, int columnWidth, String attributeName) {
Writable
Smart Insert
16 : 27 : 402


noeworkspace - 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
DBAccessor.java
db.properties
pruebaNoe.java
create_view.sql
Exercise1PrintReportOverQuery - copiaLo.txt
*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
Problems Javadoc Declaration Console
edu.uoc.practica.bd.util.Column.Column(String label, int columnWidth, String attributeName)
public Column(String label, int columnWidth, String attributeName) {
Writable
Smart Insert
16 : 29 : 403


noeworkspace - 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
DBAccessor.java
db.properties
pruebaNoe.java
create_view.sql
Exercise1PrintReportOverQuery - copiaLo.txt
*Exercise1PrintReportOverQuery.java
} 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
System.err.println("ERROR: List not available.\n");
e.printStackTrace();
}
finally {
// Cerrar todos los recursos abiertos
if (resultSet!=null)
try {
resultSet.close();
} catch(Exception ex) {}
if (cstmt!=null)
try {
cstmt.close();
} catch(Exception ex) {}
if (conn!=null)
try {
conn.close();
} catch(Exception ex) {}
}
}
}
Problems Javadoc Declaration Console
edu.uoc.practica.bd.util.Column.Column(String label, int columnWidth, String attributeName)
public Column(String label, int columnWidth, String attributeName) {
Writable
Smart Insert
16 : 29 : 403

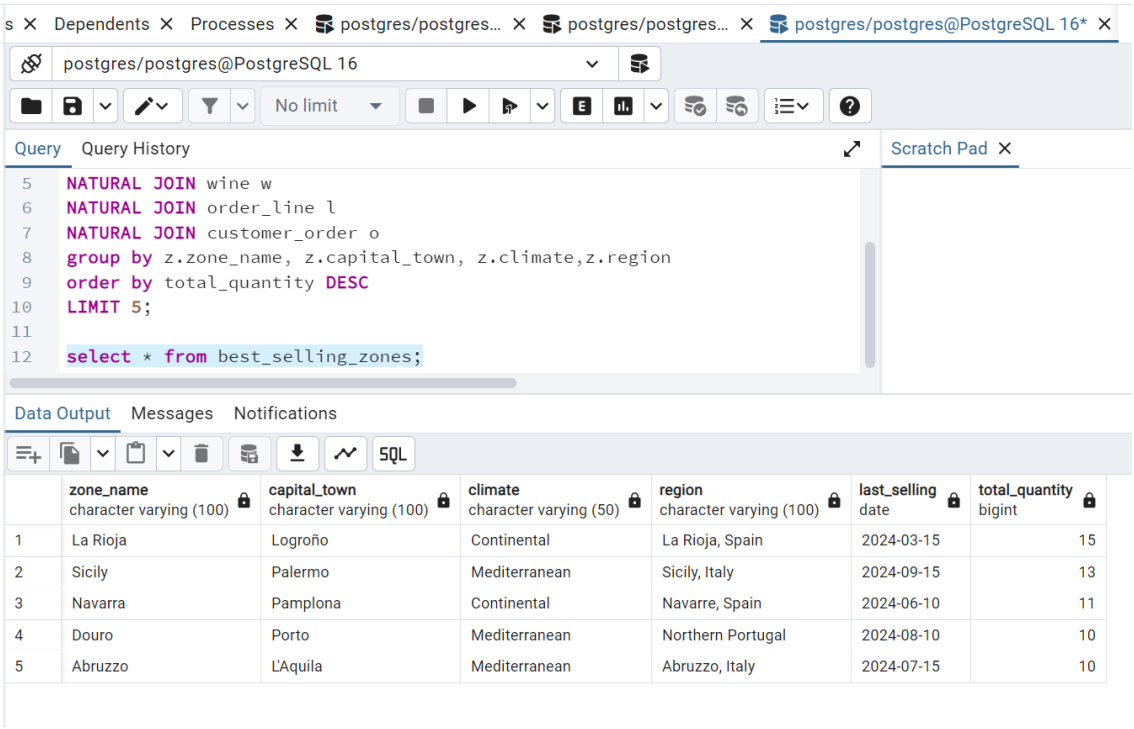
Se ha intentado comentar suficientemente el código para facilitar la corrección. Incluyo, de todos modos, por aquí una breve explicación:

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.

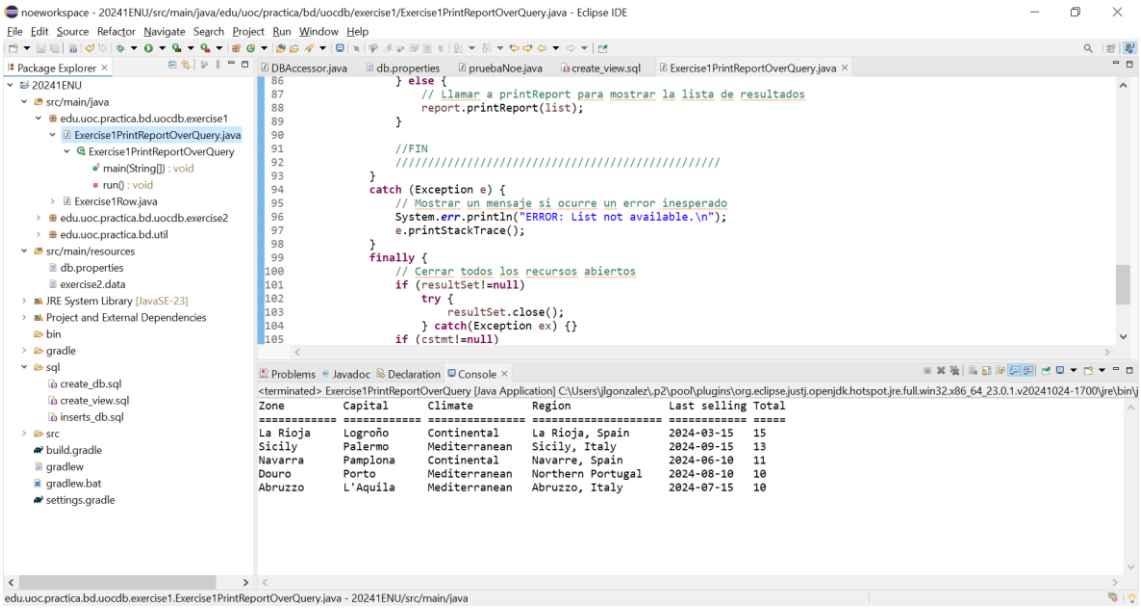
Esta implementación permite realizar la consulta de manera eficiente y garantizar que la configuración y conexión a la base de datos sean consistentes en todo el programa.

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



	zone_name character varying (100)	capital_town character varying (100)	climate character varying (50)	region character varying (100)	last_selling date	total_quantity bigint
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

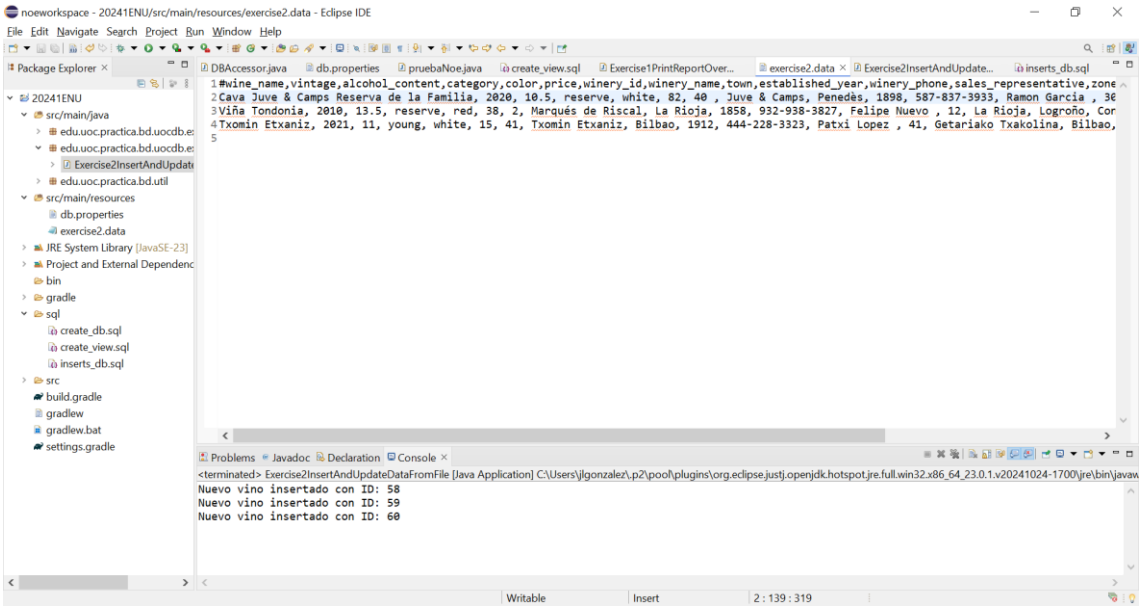
Procedemos a ejecutar nuestra clase Java:



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.

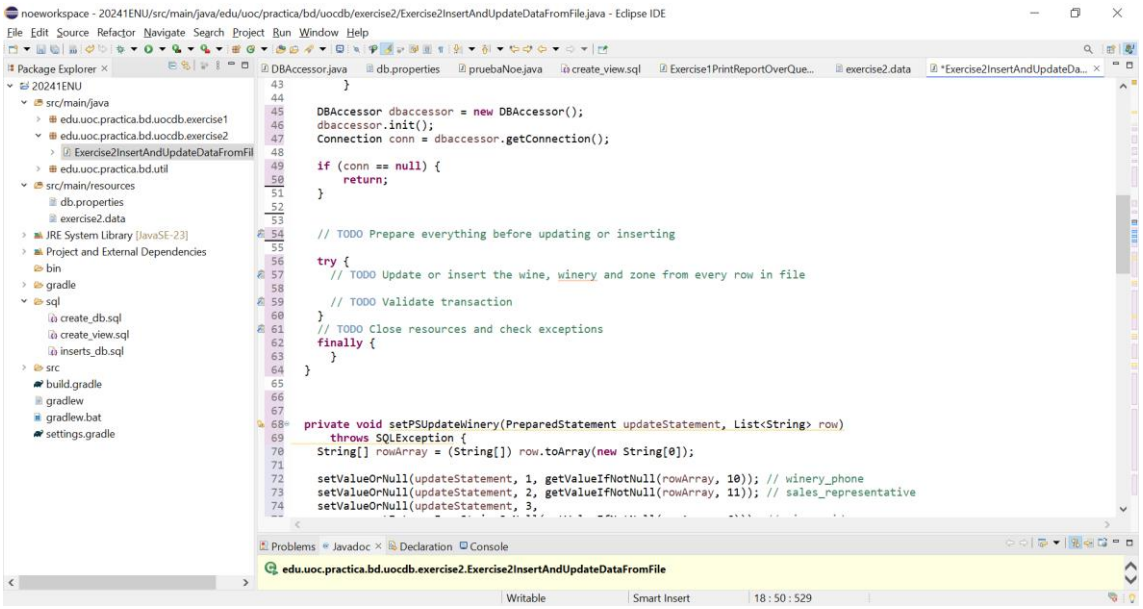


Se ha retocado el primer registro ya que se asociaba un vino a una zona equivocada.

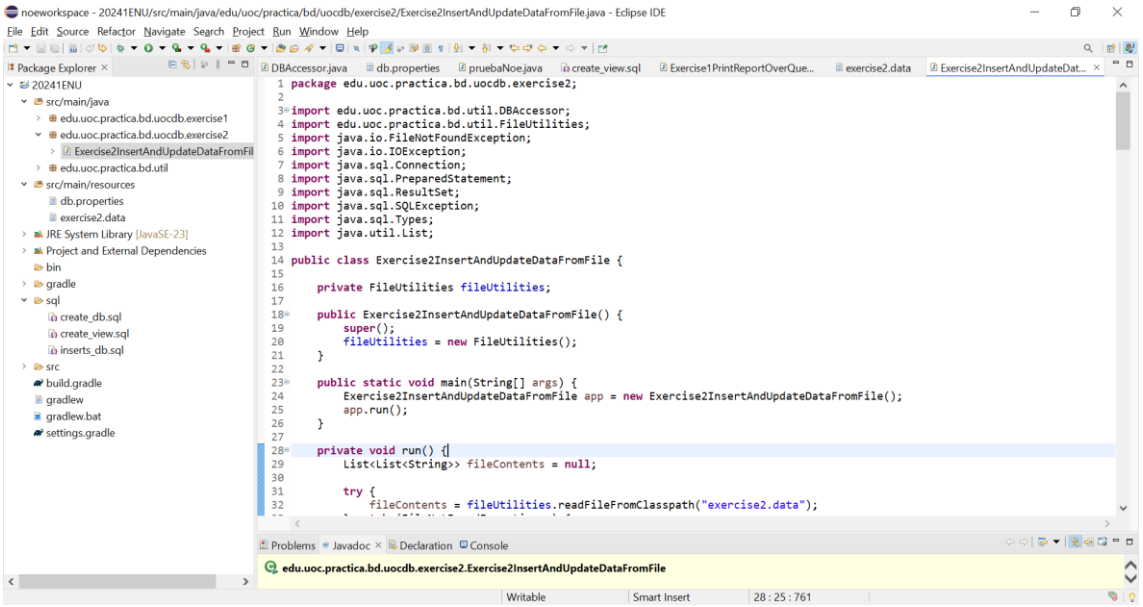
Para abordar esta actividad, se emplearon conceptos clave de programación en Java con JDBC, incluyendo:

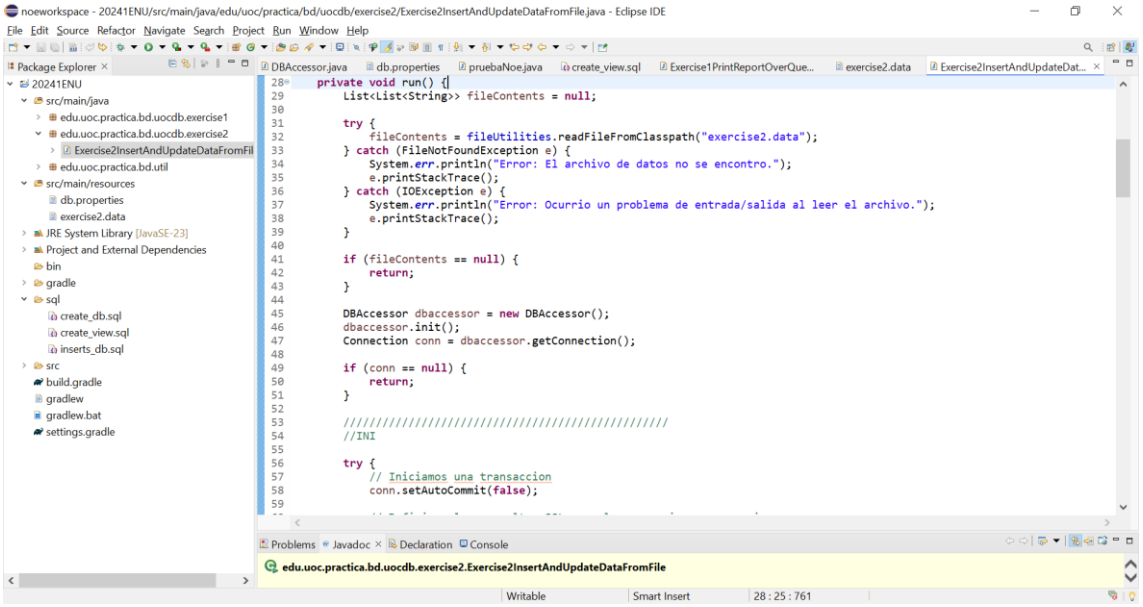
- Uso de transacciones (commit y rollback) para garantizar la consistencia de los datos.
- Sentencias preparadas (PreparedStatement) para mejorar la seguridad y el rendimiento de las operaciones.
- Manejo de excepciones para controlar errores durante la ejecución del programa y garantizar la estabilidad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

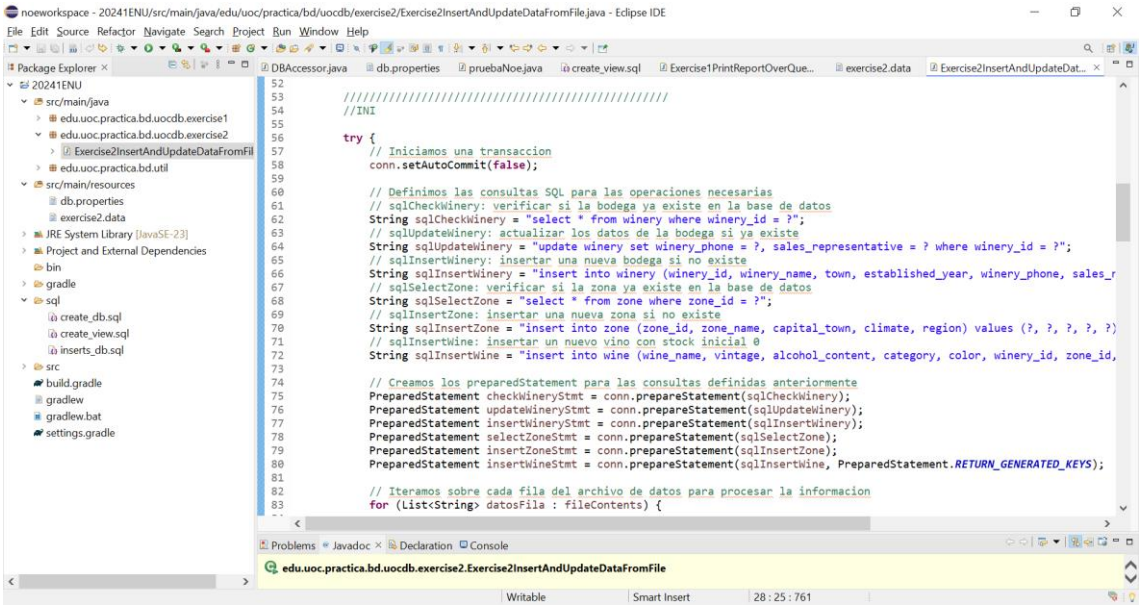
Se nos proporciona una clase Exercise2InsertAndUpdateDataFromFile a modo de guía con una serie de funcionalidades ya implementadas, como son el tratamiento del fichero de entrada y una serie de métodos auxiliares que utilizaremos para las sentencias sql.

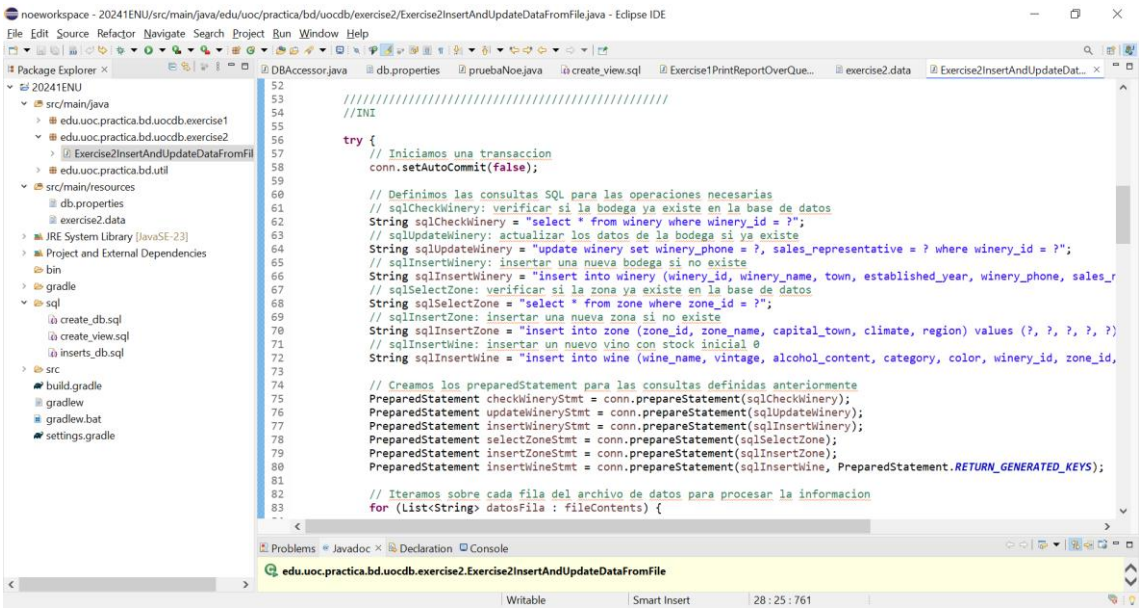


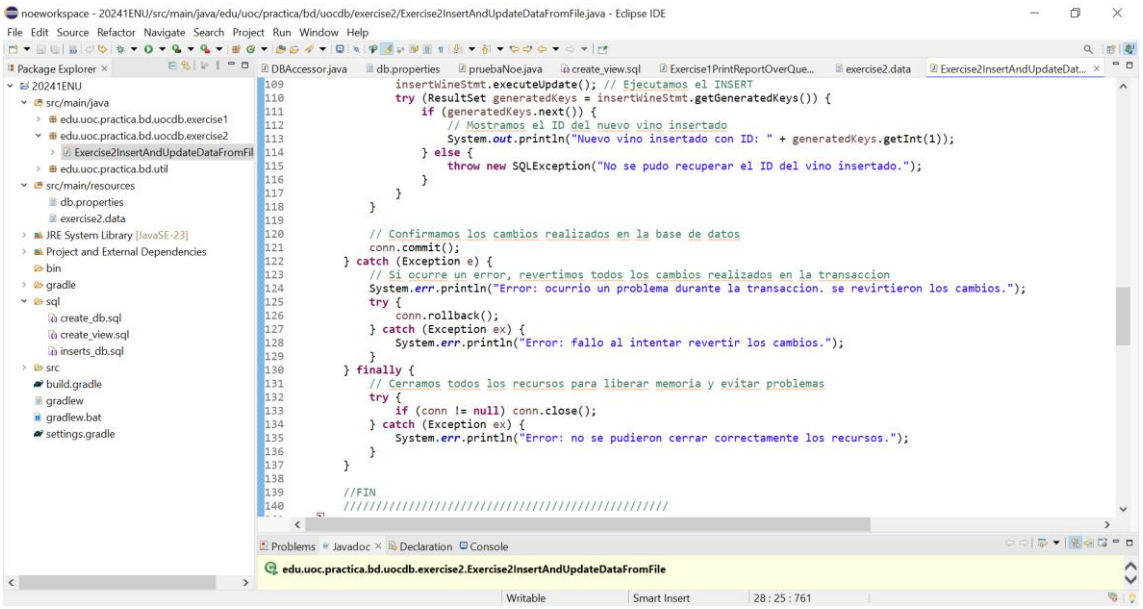
El código al que he llegado es el siguiente:

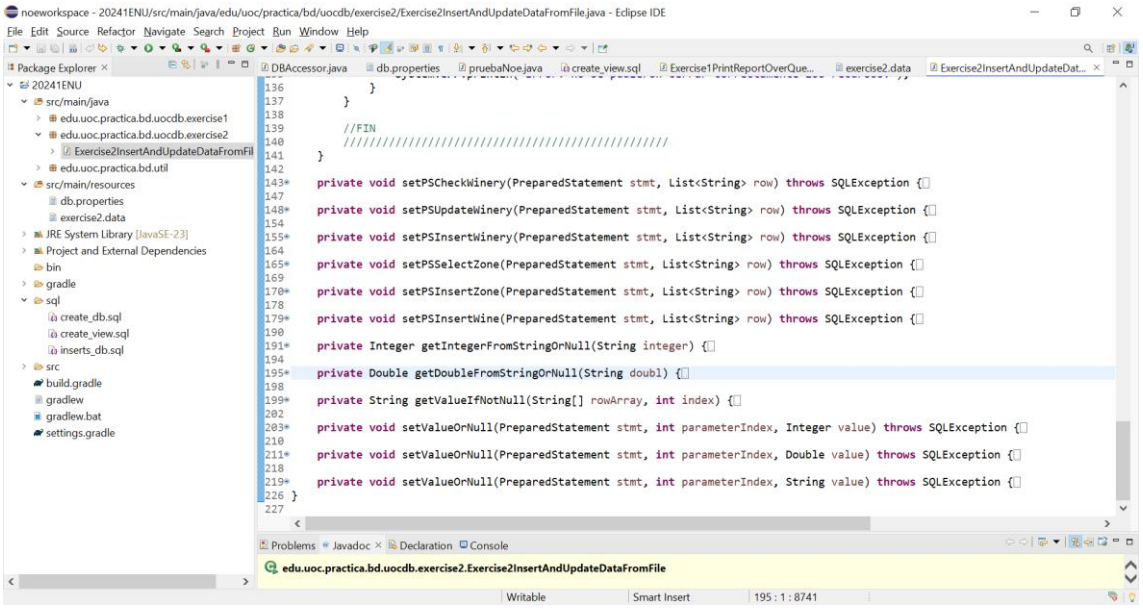



noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
20241ENU
src/main/java
edu.uoc.practica.bd.uocdb.exercise1
edu.uoc.practica.bd.uocdb.exercise2
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
DBAccessor.java
db.properties
pruebaNoe.java
create_view.sql
Exercise1PrintReportOverQue...
exercise2.data
Exercise2InsertAndUpdateDat...
private void run() {
List<List<String>> fileContents = null;
try {
fileContents = fileUtilities.readFileFromClasspath("exercise2.data");
} catch (FileNotFoundException e) {
System.err.println("Error: El archivo de datos no se encontro.");
e.printStackTrace();
} 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);
Problems Javadoc Declaration Console
edu.uoc.practica.bd.uocdb.exercise2.Exercise2InsertAndUpdateDataFromFile
Writable
Smart Insert
28 : 25 : 761


noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
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
DBAccessor.java
db.properties
pruebaNoe.java
create_view.sql
Exercise1PrintReportOverQue...
exercise2.data
Exercise2InsertAndUpdateDat...
//
//INI
try {
// Iniciamos una transaccion
conn.setAutoCommit(false);
// Definimos las consultas SQL para las operaciones necesarias
// sqlCheckWinery: verificar si la bodega ya existe en la base de datos
String sqlCheckWinery = "select * from winery where winery_id = ?";
// sqlUpdateWinery: actualizar los datos de la bodega si ya existe
String sqlUpdateWinery = "update winery set winery_phone = ?, sales_representative = ? where winery_id = ?";
// sqlInsertWinery: insertar una nueva bodega si no existe
String sqlInsertWinery = "insert into winery (winery_id, winery_name, town, established_year, winery_phone, sales_r
// sqlSelectZone: verificar si la zona ya existe en la base de datos
String sqlSelectZone = "select * from zone where zone_id = ?";
// sqlInsertZone: insertar una nueva zona si no existe
String sqlInsertZone = "insert into zone (zone_id, zone_name, capital_town, climate, region) values (?, ?, ?, ?, ?)
// sqlInsertWine: insertar un nuevo vino con stock inicial 0
String sqlInsertWine = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id,
// Creamos los preparedStatement para las consultas definidas anteriormente
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.RETURN_GENERATED_KEYS);
// Iteramos sobre cada fila del archivo de datos para procesar la informacion
for (List<String> datosFila : fileContents) {
Problems Javadoc Declaration Console
edu.uoc.practica.bd.uocdb.exercise2.Exercise2InsertAndUpdateDataFromFile
Writable
Smart Insert
28 : 25 : 761


noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
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
DBAccessor.java
db.properties
pruebaNoe.java
create_view.sql
Exercise1PrintReportOverQue...
exercise2.data
Exercise2InsertAndUpdateDat...
//
//INI
try {
// Iniciamos una transaccion
conn.setAutoCommit(false);
// Definimos las consultas SQL para las operaciones necesarias
// sqlCheckWinery: verificar si la bodega ya existe en la base de datos
String sqlCheckWinery = "select * from winery where winery_id = ?";
// sqlUpdateWinery: actualizar los datos de la bodega si ya existe
String sqlUpdateWinery = "update winery set winery_phone = ?, sales_representative = ? where winery_id = ?";
// sqlInsertWinery: insertar una nueva bodega si no existe
String sqlInsertWinery = "insert into winery (winery_id, winery_name, town, established_year, winery_phone, sales_r
// sqlSelectZone: verificar si la zona ya existe en la base de datos
String sqlSelectZone = "select * from zone where zone_id = ?";
// sqlInsertZone: insertar una nueva zona si no existe
String sqlInsertZone = "insert into zone (zone_id, zone_name, capital_town, climate, region) values (?, ?, ?, ?, ?)
// sqlInsertWine: insertar un nuevo vino con stock inicial 0
String sqlInsertWine = "insert into wine (wine_name, vintage, alcohol_content, category, color, winery_id, zone_id,
// Creamos los preparedStatement para las consultas definidas anteriormente
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.RETURN_GENERATED_KEYS);
// Iteramos sobre cada fila del archivo de datos para procesar la informacion
for (List<String> datosFila : fileContents) {
Problems Javadoc Declaration Console
edu.uoc.practica.bd.uocdb.exercise2.Exercise2InsertAndUpdateDataFromFile
Writable
Smart Insert
28 : 25 : 761





Comento nuevamente lo más relevante para facilitar la corrección:

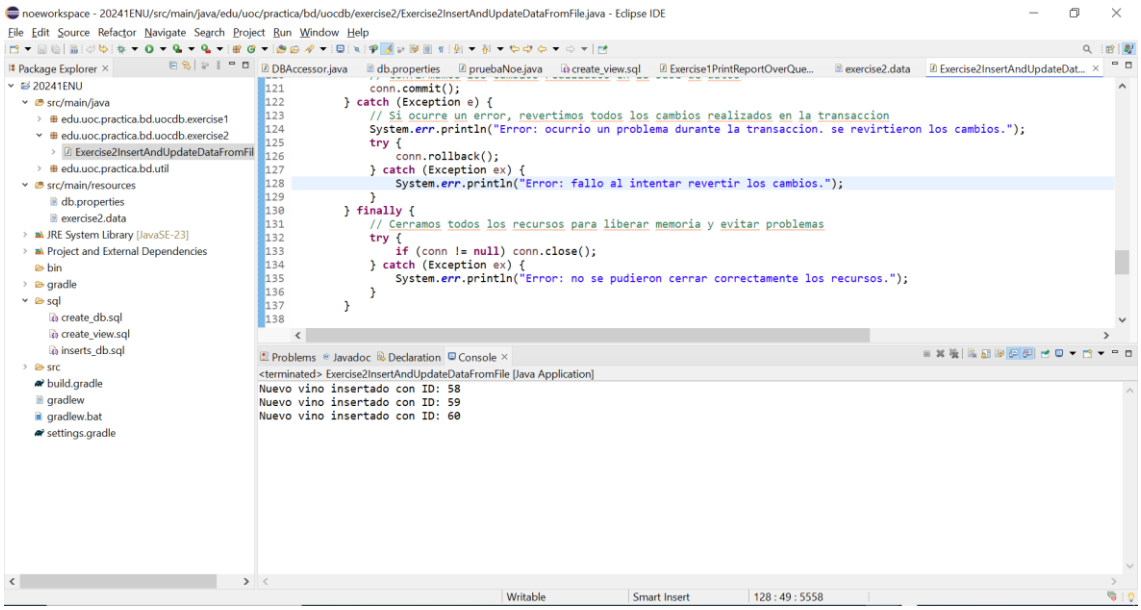
- Inicio de la Transacción: Antes de procesar el archivo, se configura la conexión para trabajar en modo transaccional (conn.setAutoCommit(false)), lo que permite agrupar las operaciones como una unidad atómica. Esto garantiza que, en caso de error, todos los cambios realizados hasta ese punto puedan deshacerse mediante un rollback.
- Preparación de Sentencias SQL:
  Se definieron varias sentencias SQL en forma de PreparedStatement para las diferentes operaciones necesarias:
  - Verificación de registros existentes: SELECT para comprobar si una bodega o una zona ya existe.
  - Actualización de registros existentes: UPDATE para modificar datos de contacto o representantes de ventas en bodegas existentes.
  - 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:
  - 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.
  - Zona:
    Comprueba si la zona ya existe mediante un SELECT.
    Si no existe, la inserta en la base de datos con un INSERT.
  - 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.

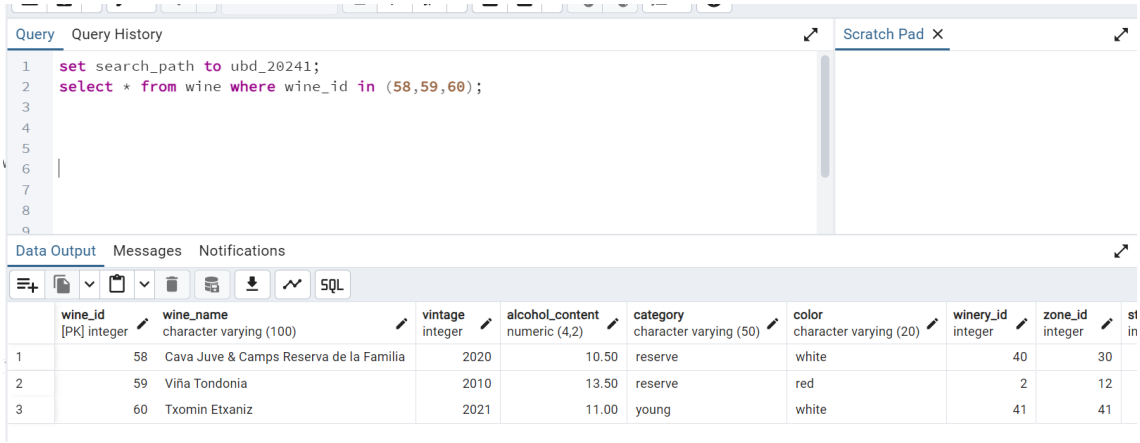
Las sentencias sql utilizadas en este ejercicio son:

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

Para demostrar el buen funcionamiento, se ejecuta la clase

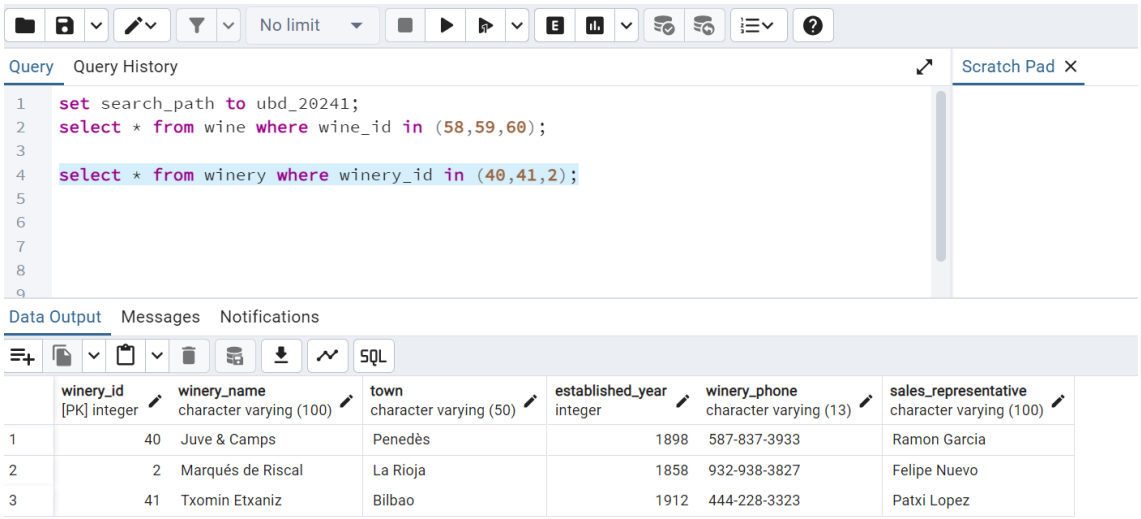


Si pasamos a revisar la base de datos, vemos que los vinos se insertaron:



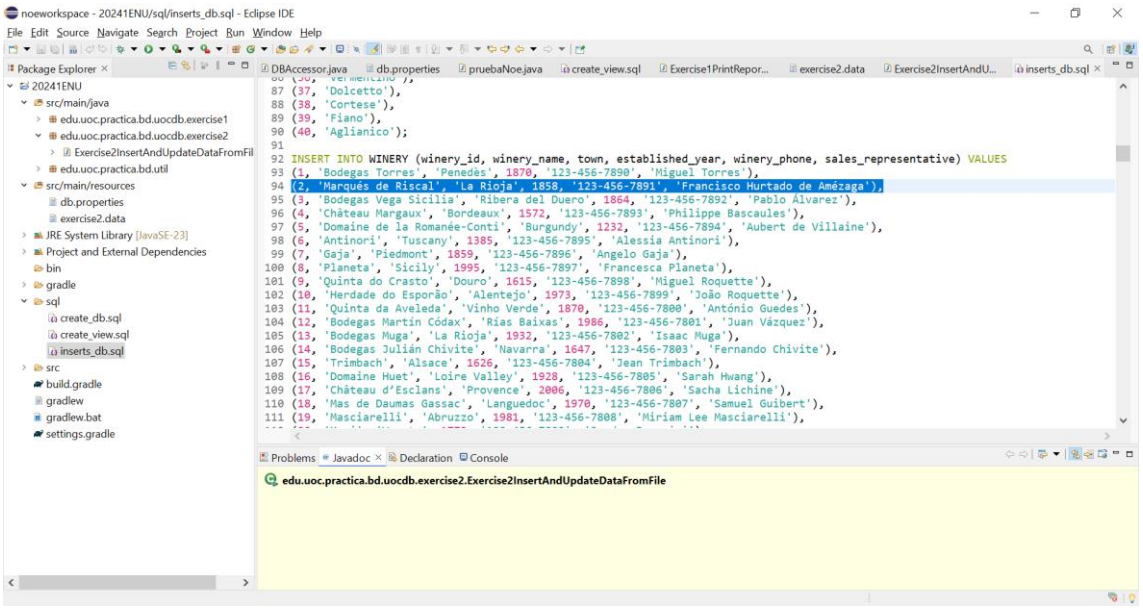
	wine_id [PK] integer	wine_name character varying (100)	vintage integer	alcohol_content numeric (4,2)	category character varying (50)	color character varying (20)	winery_id integer	zone_id integer
1	58	Cava Juve & Camps Reserva de la Familia	2020	10.50	reserve	white	40	30
2	59	Viña Tondonia	2010	13.50	reserve	red	2	12
3	60	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:



	winery_id [PK] integer	winery_name character varying (100)	town character varying (50)	established_year integer	winery_phone character varying (13)	sales_representative character varying (100)
1	40	Juve & Camps	Penedès	1898	587-837-3933	Ramon Garcia
2	2	Marqués de Riscal	La Rioja	1858	932-938-3827	Felipe Nuevo
3	41	Txomin Etxaniz	Bilbao	1912	444-228-3323	Patxi Lopez

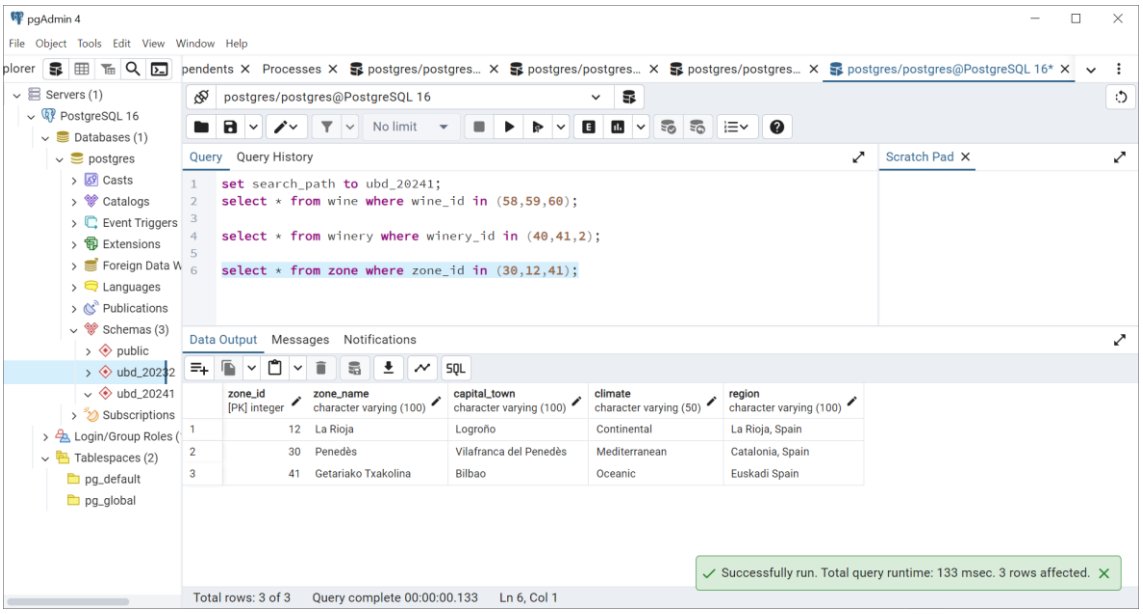
La bodega 2, tal y como se ve en el fichero de inserts, existía



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

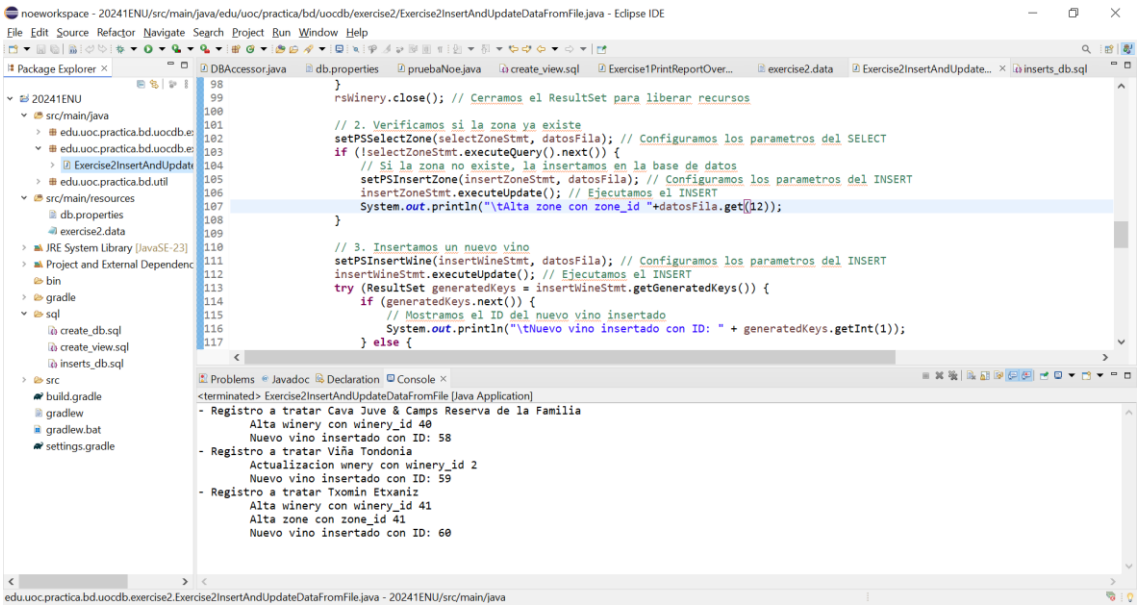
Las bodegas 40 y 41 se insertaron correctamente, ya que no estaban incluidas en el fichero de inserts de creación.

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



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 codigo:


noeworkspace - 20241ENU/src/main/java/edu/uoc/practica/bd/uocdb/exercise2/Exercise2InsertAndUpdateDataFromFile.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
20241ENU
src/main/java
edu.uoc.practica.bd.uocdb.e
edu.uoc.practica.bd.uocdb.e
Exercise2InsertAndUpdat
edu.uoc.practica.bd.util
src/main/resources
db.properties
exercise2.data
JRE System Library [JavaSE-23]
Project and External Dependenc
bin
gradle
sql
create_db.sql
create_view.sql
inserts_db.sql
src
build.gradle
gradlew
gradlew.bat
settings.gradle
DBAccessor.java
db.properties
pruebaNoe.java
create_view.sql
Exercise1PrintReportOver...
exercise2.data
Exercise2InsertAndUpdate...
inserts_db.sql
98 }
99 rsWinery.close(); // Cerramos el ResultSet para liberar recursos
100
101 // 2. Verificamos si la zona ya existe
102 setPSSelectZone(selectZoneStmt, datosFila); // Configuramos los parametros del SELECT
103 if (!selectZoneStmt.executeQuery().next()) {
104 // Si la zona no existe, la insertamos en la base de datos
105 setPSInsertZone(insertZoneStmt, datosFila); // Configuramos los parametros del INSERT
106 insertZoneStmt.executeUpdate(); // Ejecutamos el INSERT
107 System.out.println("\tAlta zone con zone_id "+datosFila.get(12));
108 }
109
110 // 3. Insertamos un nuevo vino
111 setPSInsertWine(insertWineStmt, datosFila); // Configuramos los parametros del INSERT
112 insertWineStmt.executeUpdate(); // Ejecutamos el INSERT
113 try (ResultSet generatedKeys = insertWineStmt.getGeneratedKeys()) {
114 if (generatedKeys.next()) {
115 // Mostramos el ID del nuevo vino insertado
116 System.out.println("\tNuevo vino insertado con ID: " + generatedKeys.getInt(1));
117 } else {
Problems Javadoc Declaration Console
<terminated> Exercise2InsertAndUpdateDataFromFile [Java Application]
- Registro a tratar Cava Juve & Camps Reserva de la Familia
Alta winery con winery_id 40
Nuevo vino insertado con ID: 58
- Registro a tratar Viña Tondonia
Actualizacion wnery 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
edu.uoc.practica.bd.uocdb.exercise2.Exercise2InsertAndUpdateDataFromFile.java - 20241ENU/src/main/java