Unidad 2

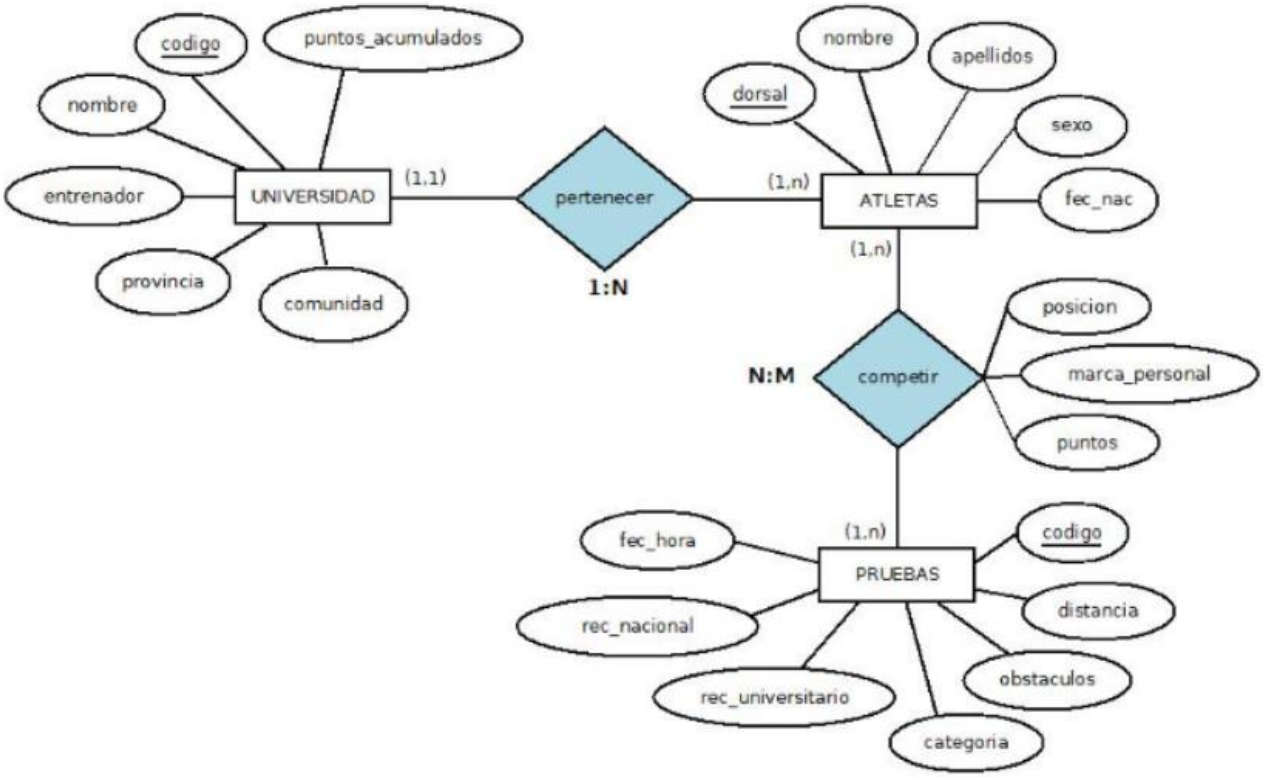
Manejo de conectores

Tarea AD02

Manejando conectores

1. Enunciado de la tarea

El diagrama del modelo E/R de la base de datos **Campeonato_Atlentismo** es el siguiente:



Partiendo de esta base de datos realizar los siguientes ejercicios:

EJERCICIO 1:

Lanza el script SQL de creación del esquema de la base de datos **Campeonato_atletismo** y asegúrate de que se crean todas las tablas y se insertan los registros correspondientes. Puedes usar, bien un cliente gráfico (MySQL Workbench) o bien un cliente en modo texto (Línea de comandos MySQL). Debes documentar los pasos realizados en el documento de entrega de la tarea.

EJERCICIO 2:

Crea un proyecto en Netbeans con nombre **Nombre_Apellido1_AD2_E2** para crear una aplicación con un interfaz gráfico usando Swing, que permita realizar las siguientes acciones:

A. Establecer conexión con la base de datos permitiendo al usuario introducir usuario y contraseña.

B. Mostrar en la aplicación, el resultado de implementar a través de sentencias preparadas y parametrizadas las siguientes consultas:

   1. Obtener el nombre y apellidos de los atletas junto al nombre de la universidad a la que pertenece de todos aquellos atletas de la comunidad que indique el usuario de la aplicación (la aplicación debe permitir al usuario indicar una comunidad).

2. Calcular la cantidad total de puntos que ha obtenido un atleta determinado (identificado por su dorsal) en las diferentes pruebas (la aplicación debe permitir al usuario indicar el número de dorsal).

C. Modificar, a través de una sentencia preparada y parametrizada, la distancia de una prueba determinada indicada por su código (el usuario introducirá el código de la prueba y la nueva distancia).

D. Ejecutar un procedimiento almacenado llamado **atletas_posicion** que devuelva en un parámetro de salida (que se mostrará en la aplicación), el número total de atletas que hayan acabado en una determinada posición en alguna prueba. Se debe pasar como parámetro al procedimiento la posición por la que se quiere consultar, que será introducida por el usuario (se debe adjuntar en la entrega de la tarea el script de creación del procedimiento almacenado).

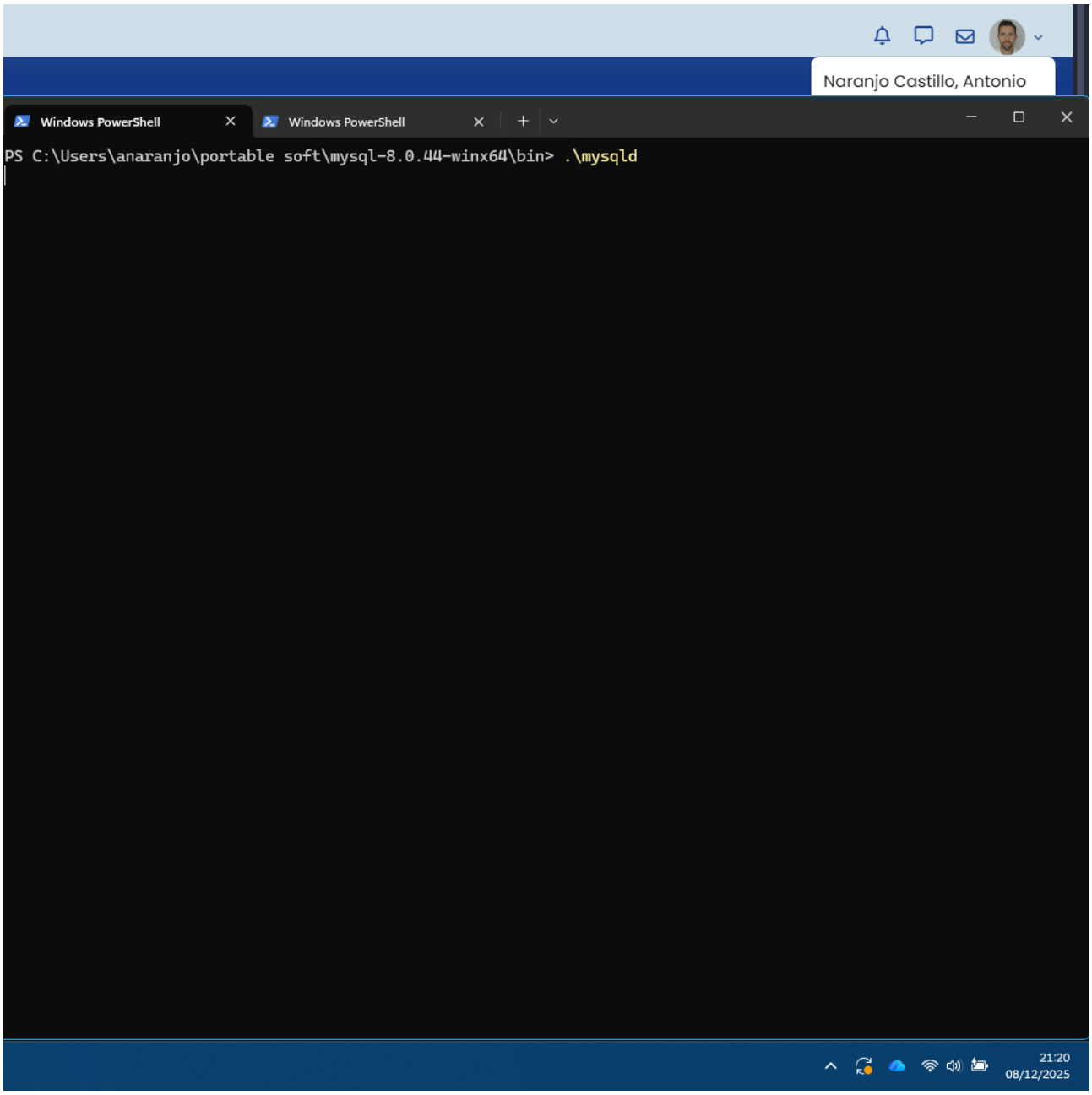
IMPORTANTE:

- La aplicación debe gestionar las posibles excepciones y errores que puedan presentarse, así como el cierre de recursos utilizados (usa la sentencia try-catch-finally o equivalente).

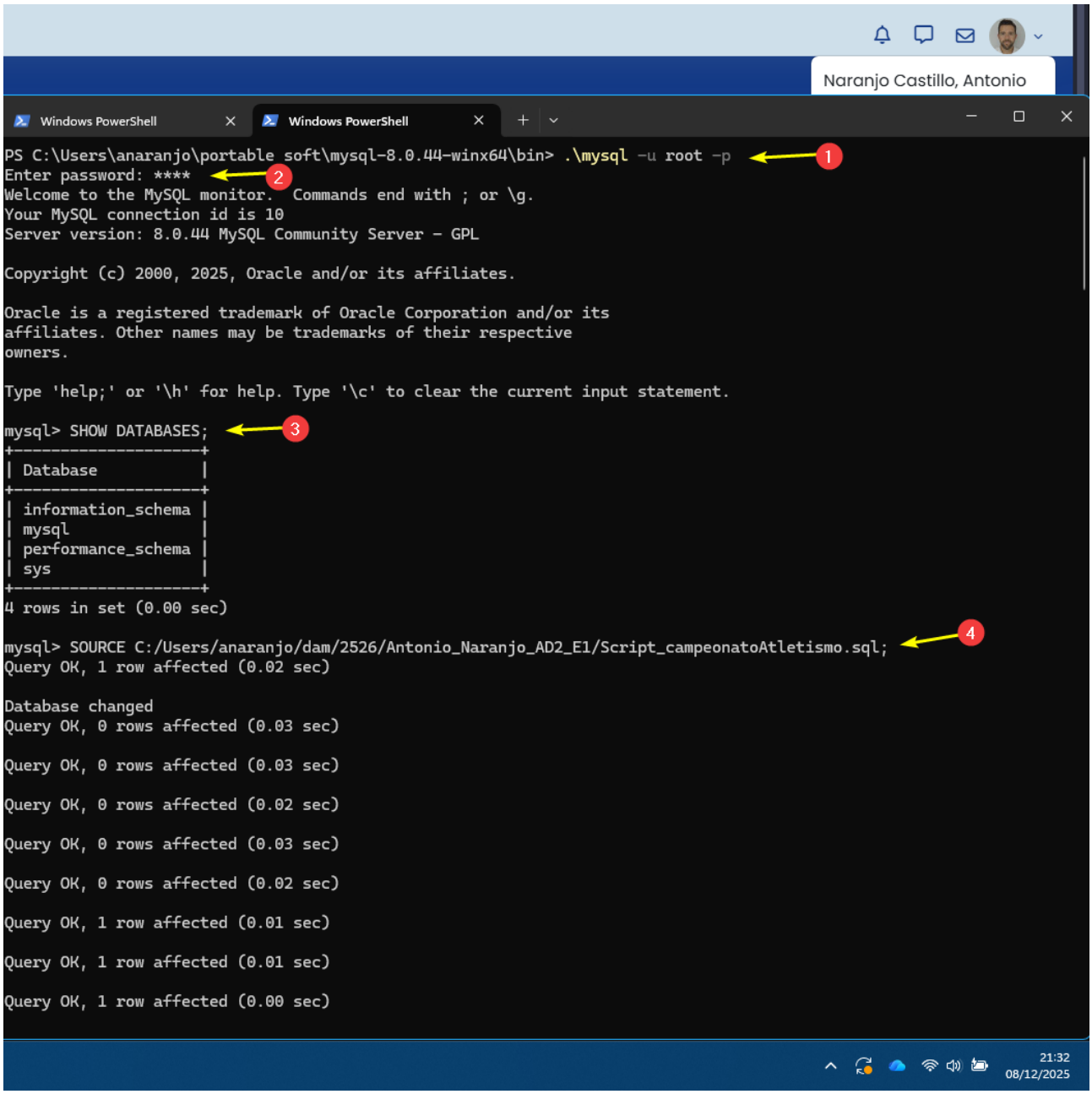
- La aplicación deberá informar adecuadamente al usuario de los posibles errores, datos esperados, etc., para guiar al usuario final en el uso de la misma.

2. Preparación del Script, acceso a la consola y comprobación de datos

Se ejecuta el cliente en modo texto de MySQL versión 8.0.44 desde línea de comandos.

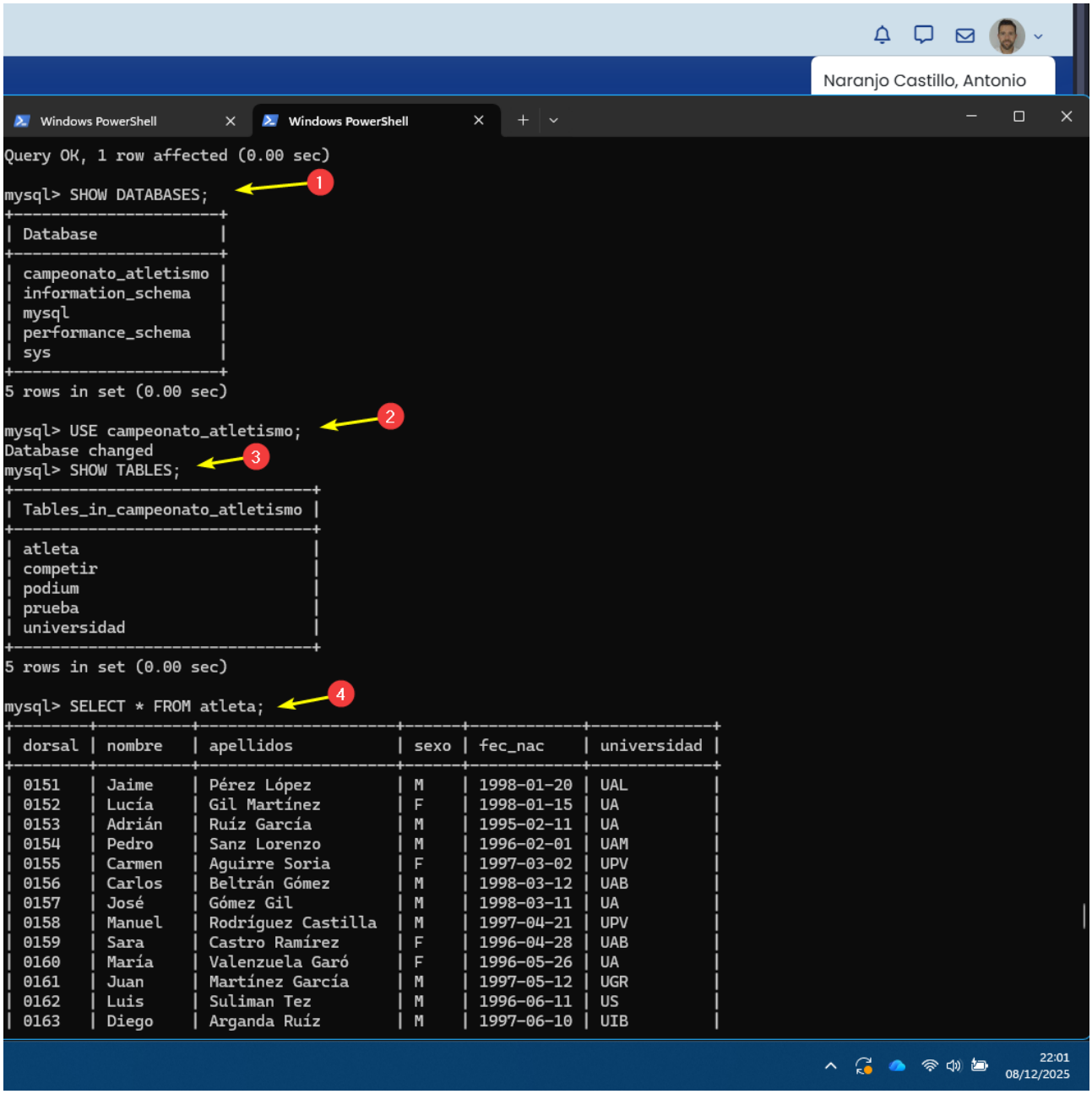


Desde otra pestaña de la línea de comandos, se accede al cliente MySQL con las credenciales de administrador (usuario root y contraseña root) para asegurar los permisos necesarios para la creación de la base de datos y la carga de datos.

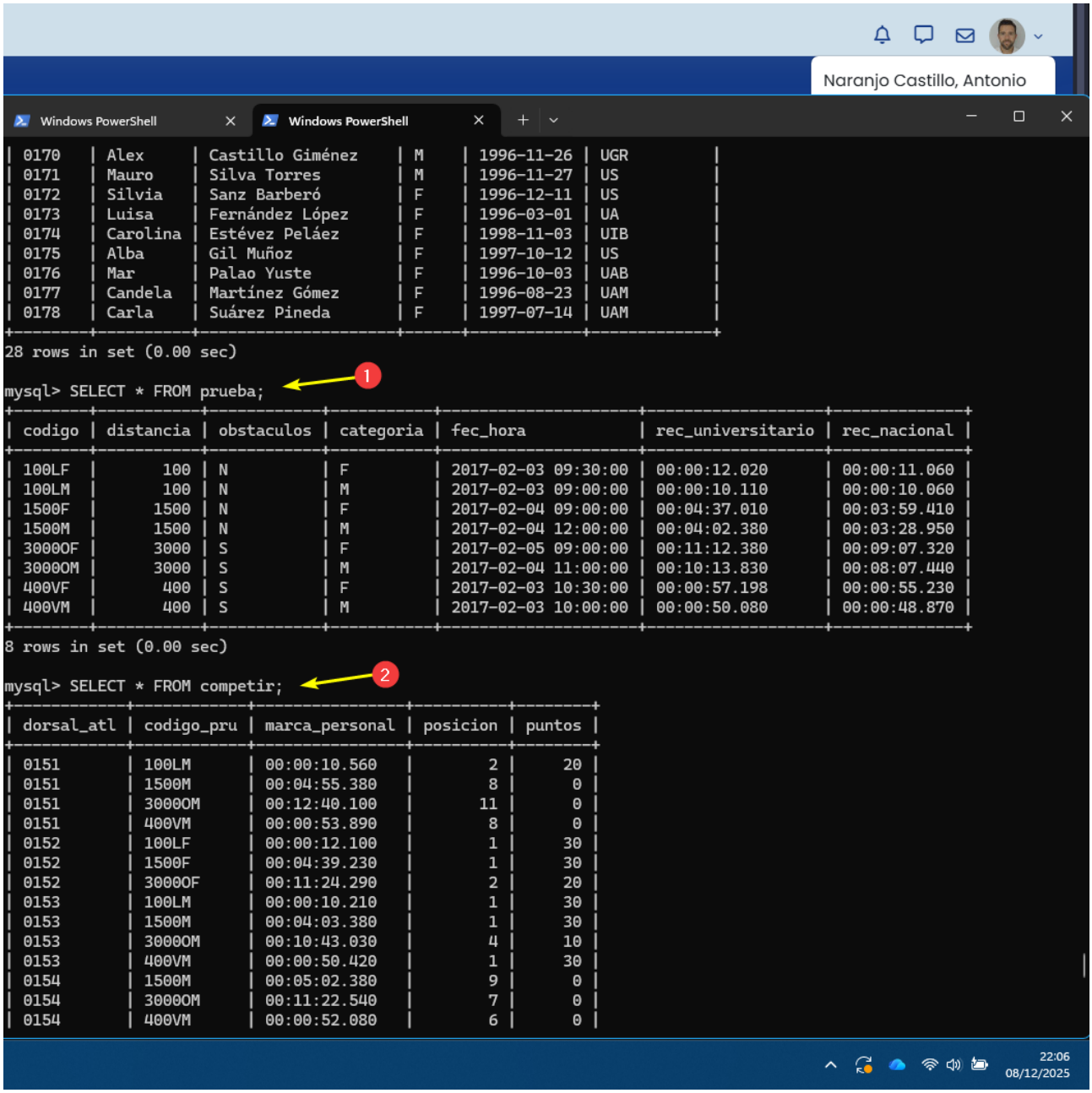


Luego, se consultan las bases de datos existentes, y se ejecuta el script facilitado para llevar a cabo la presente tarea.

Se comprueba que la base de datos campeonato_atletismo se ha cargado con éxito.
Se asegura que la base de datos en uso sea campeonato_atletismo.
Se muestran las tablas existentes en la base de datos.
Se muestran los datos de la tabla atleta, a modo de comprobación de que los datos han sido incorporados satisfactoriamente.

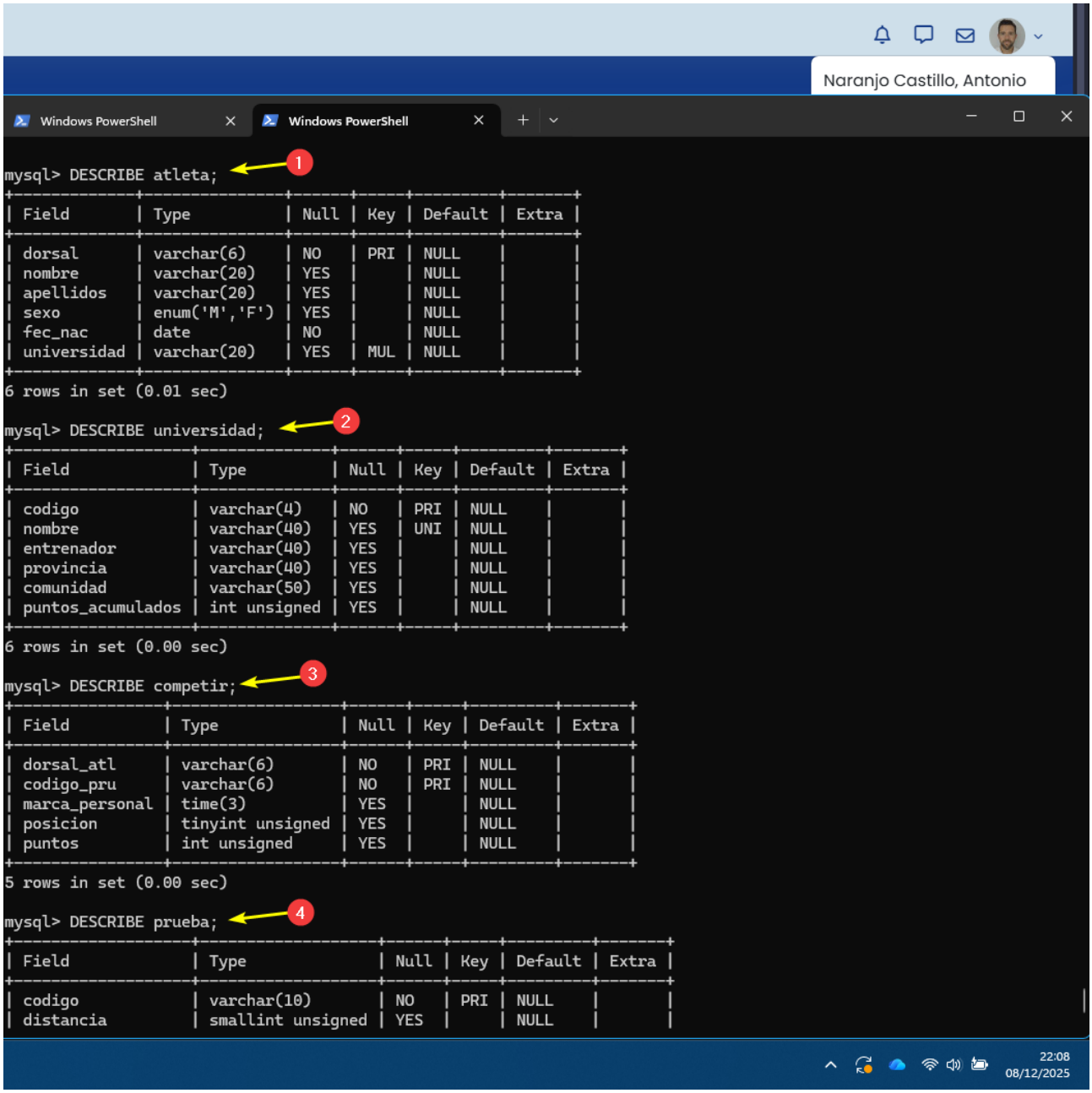


Se comprueban los datos de algunas tablas más PRUEBA y COMPETIR.




```
| 0170 | Alex     | Castillo Giménez | M | 1996-11-26 | UGR |
| 0171 | Mauro    | Silva Torres     | M | 1996-11-27 | US  |
| 0172 | Silvia   | Sanz Barberó     | F | 1996-12-11 | US  |
| 0173 | Luisa    | Fernández López  | F | 1996-03-01 | UA  |
| 0174 | Carolina | Estévez Peláez   | F | 1998-11-03 | UIB |
| 0175 | Alba     | Gil Muñoz        | F | 1997-10-12 | US  |
| 0176 | Mar      | Palao Yuste      | F | 1996-10-03 | UAB |
| 0177 | Candela  | Martínez Gómez   | F | 1996-08-23 | UAM |
| 0178 | Carla    | Suárez Pineda    | F | 1997-07-14 | UAM |
+------+----------+------------------+---+------------+-----+
28 rows in set (0.00 sec)

mysql> SELECT * FROM prueba;
+--------+-----------+-----------+-----------+---------------------+-------------------+--------------+
| codigo | distancia | obstaculos | categoria | fec_hora            | rec_universitario | rec_nacional |
+--------+-----------+-----------+-----------+---------------------+-------------------+--------------+
| 100LF  |       100 | N         | F         | 2017-02-03 09:30:00 | 00:00:12.020      | 00:00:11.060 |
| 100LM  |       100 | N         | M         | 2017-02-03 09:00:00 | 00:00:10.110      | 00:00:10.060 |
| 1500F  |      1500 | N         | F         | 2017-02-04 09:00:00 | 00:04:37.010      | 00:03:59.410 |
| 1500M  |      1500 | N         | M         | 2017-02-04 12:00:00 | 00:04:02.380      | 00:03:28.950 |
| 3000OF |      3000 | S         | F         | 2017-02-05 09:00:00 | 00:11:12.380      | 00:09:07.320 |
| 3000OM |      3000 | S         | M         | 2017-02-04 11:00:00 | 00:10:13.830      | 00:08:07.440 |
| 400VF  |       400 | S         | F         | 2017-02-03 10:30:00 | 00:00:57.198      | 00:00:55.230 |
| 400VM  |       400 | S         | M         | 2017-02-03 10:00:00 | 00:00:50.080      | 00:00:48.870 |
+--------+-----------+-----------+-----------+---------------------+-------------------+--------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM competir;
+------------+------------+----------------+----------+--------+
| dorsal_atl | codigo_pru | marca_personal | posicion | puntos |
+------------+------------+----------------+----------+--------+
| 0151       | 100LM      | 00:00:10.560   |        2 |     20 |
| 0151       | 1500M      | 00:04:55.380   |        8 |      0 |
| 0151       | 3000OM     | 00:12:40.100   |       11 |      0 |
| 0151       | 400VM      | 00:00:53.890   |        8 |      0 |
| 0152       | 100LF      | 00:00:12.100   |        1 |     30 |
| 0152       | 1500F      | 00:04:39.230   |        1 |     30 |
| 0152       | 3000OF     | 00:11:24.290   |        2 |     20 |
| 0153       | 100LM      | 00:00:10.210   |        1 |     30 |
| 0153       | 1500M      | 00:04:03.380   |        1 |     30 |
| 0153       | 3000OM     | 00:10:43.030   |        4 |     10 |
| 0153       | 400VM      | 00:00:50.420   |        1 |     30 |
| 0154       | 1500M      | 00:05:02.380   |        9 |      0 |
| 0154       | 3000OM     | 00:11:22.540   |        7 |      0 |
| 0154       | 400VM      | 00:00:52.080   |        6 |      0 |
```

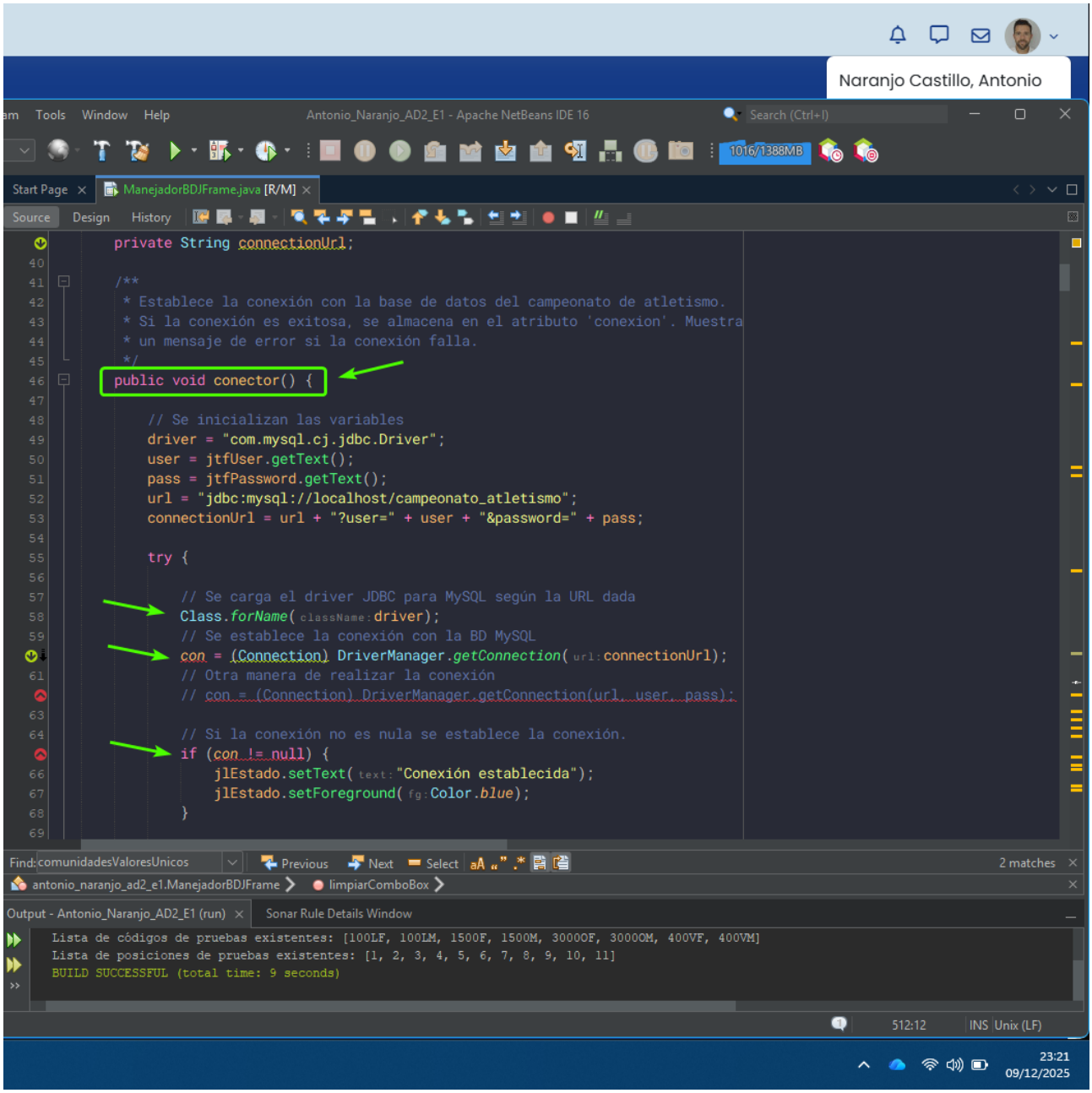
Y los tipos de datos almacenados en cada una de ellas.




```
mysql> DESCRIBE atleta;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| dorsal      | varchar(6)    | NO   | PRI | NULL    |       |
| nombre      | varchar(20)   | YES  |     | NULL    |       |
| apellidos   | varchar(20)   | YES  |     | NULL    |       |
| sexo        | enum('M','F') | YES  |     | NULL    |       |
| fec_nac     | date          | NO   |     | NULL    |       |
| universidad | varchar(20)   | YES  | MUL | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> DESCRIBE universidad;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| codigo           | varchar(4)   | NO   | PRI | NULL    |       |
| nombre           | varchar(40)  | YES  | UNI | NULL    |       |
| entrenador       | varchar(40)  | YES  |     | NULL    |       |
| provincia        | varchar(40)  | YES  |     | NULL    |       |
| comunidad        | varchar(50)  | YES  |     | NULL    |       |
| puntos_acumulados | int unsigned | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> DESCRIBE competir;
+---------------+-----------------+------+-----+---------+-------+
| Field         | Type            | Null | Key | Default | Extra |
+---------------+-----------------+------+-----+---------+-------+
| dorsal_atl    | varchar(6)      | NO   | PRI | NULL    |       |
| codigo_pru    | varchar(6)      | NO   | PRI | NULL    |       |
| marca_personal | time(3)        | YES  |     | NULL    |       |
| posicion      | tinyint unsigned | YES  |     | NULL    |       |
| puntos        | int unsigned    | YES  |     | NULL    |       |
+---------------+-----------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> DESCRIBE prueba;
+------------------+-----------------+------+-----+---------+-------+
| Field            | Type            | Null | Key | Default | Extra |
+------------------+-----------------+------+-----+---------+-------+
| codigo           | varchar(10)     | NO   | PRI | NULL    |       |
| distancia        | smallint unsigned | YES  |     | NULL    |       |
```

3. Estructura del proyecto Netbeans y establecimiento de la conexión a la base de datos

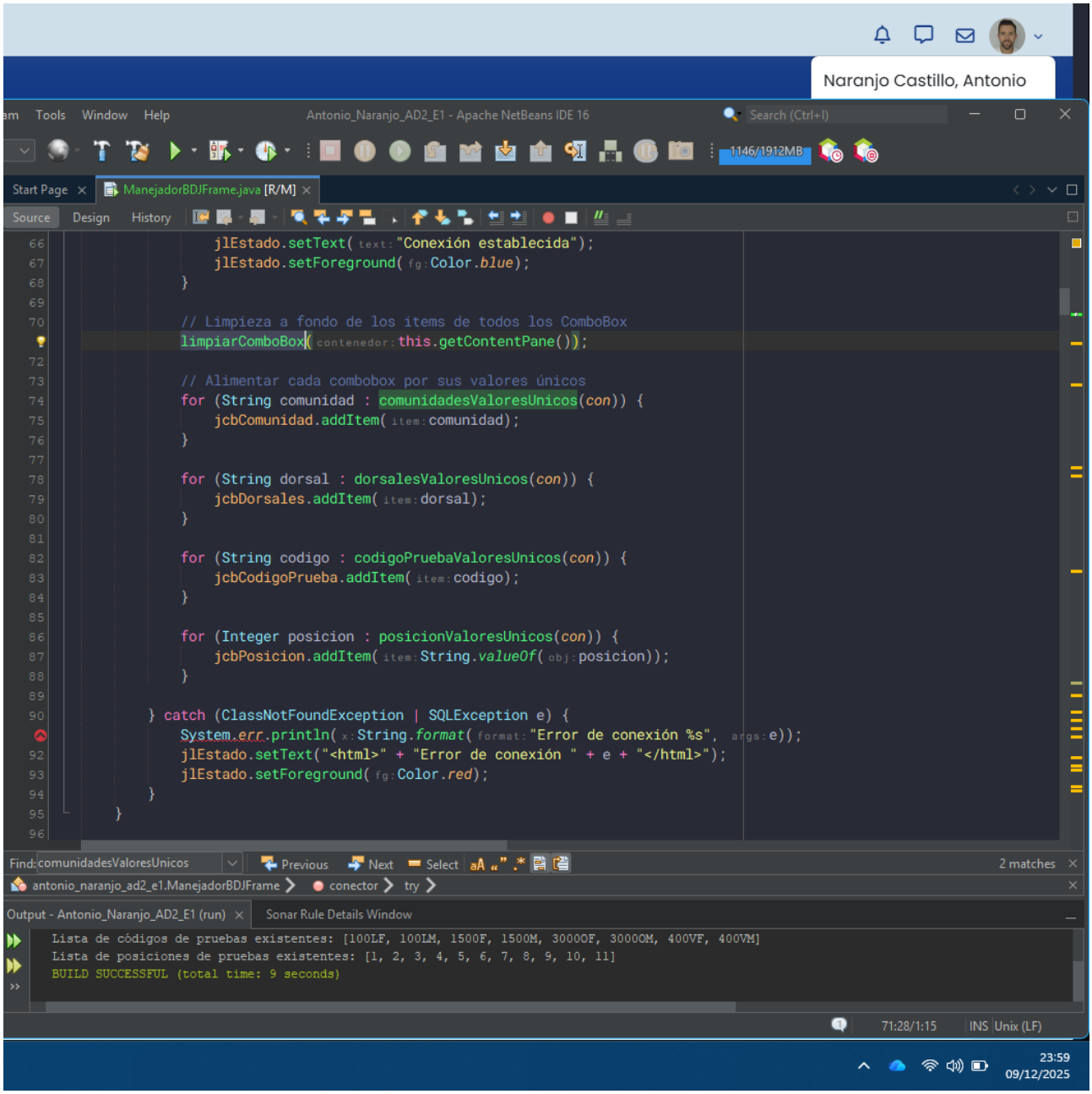
Se crea el proyecto en NetBeans Antonio_Naranjo_AD2_E1 y la clase principal ManejadorBDJFrame.
Para el manejo de la conexión se implementa un método llamado conector().
Este método establece la conexión JDBC según la siguiente imagen.



Se carga el driver JDBC empleando el método estático forname() de la clase Class, pasando como argumento el nombre del conector como un String.
Se establece la conexión creando un objeto Connection e iniciándolo por medio del método estático getConnection de la clase DriverManager, pasando como argumento la dirección URL completa donde se define tanto la dirección de la base de datos campeonato_atletismo (ubicada en el propio PC o localhost), así como el usuario y la contraseña. Se deja comentado un método más moderno

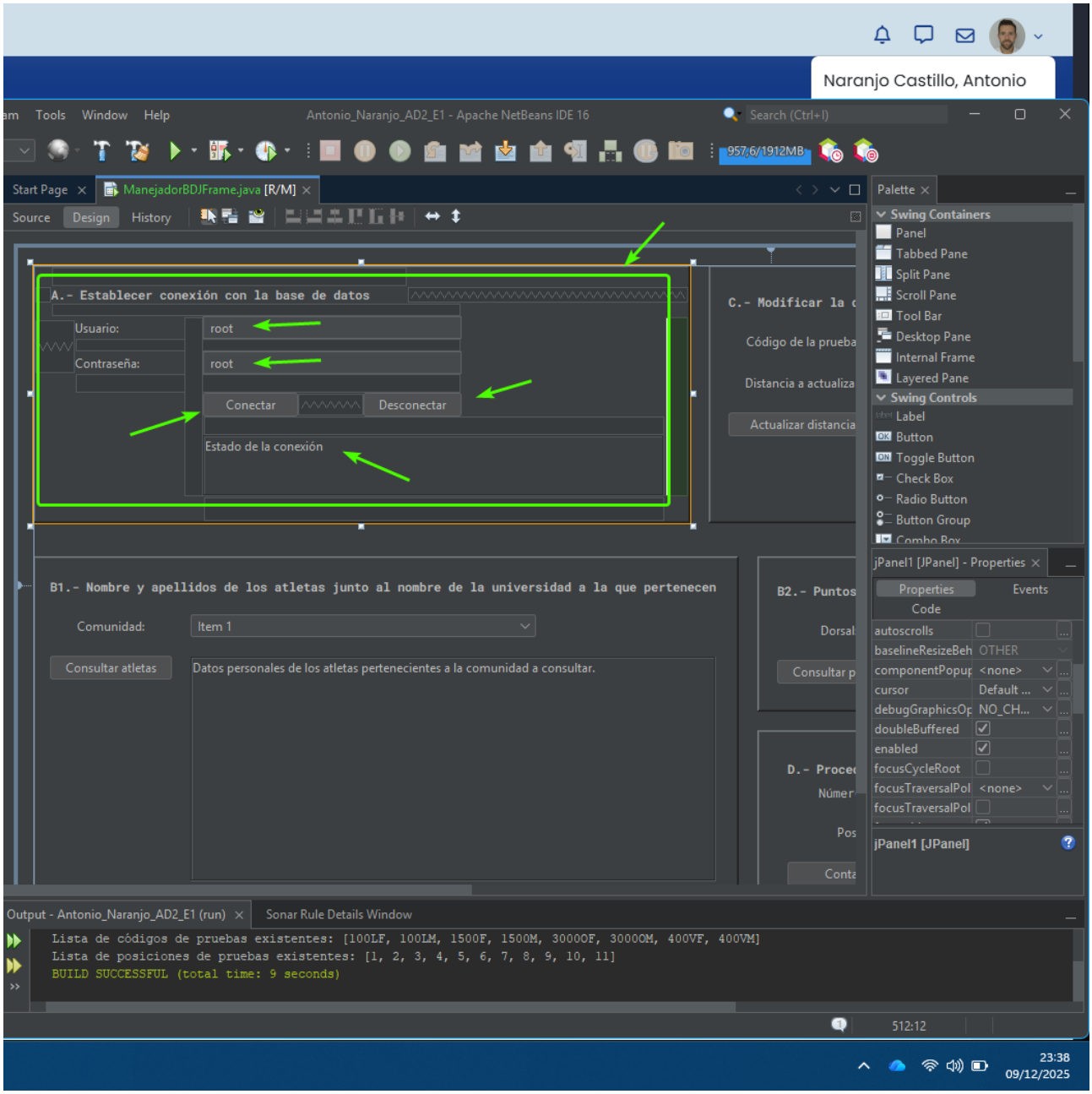
para establecer dicha conexión, llamando directamente al método getConnetion anterior y pasándole por argumento directamente la dirección completa de la base de datos, usuario y contraseña.

Se aprovecha este mismo método conector() para cargar los comboBox de otros paneles del aplicativo Swing diseñado. Previamente se limpian de datos existentes mediante el método limpiarComboBox() aportando como argumento el contenedor JFrame.



Tanto si la conexión se establece con éxito o no se lanza un mensaje en una etiqueta establecida en la interfaz gráfica.

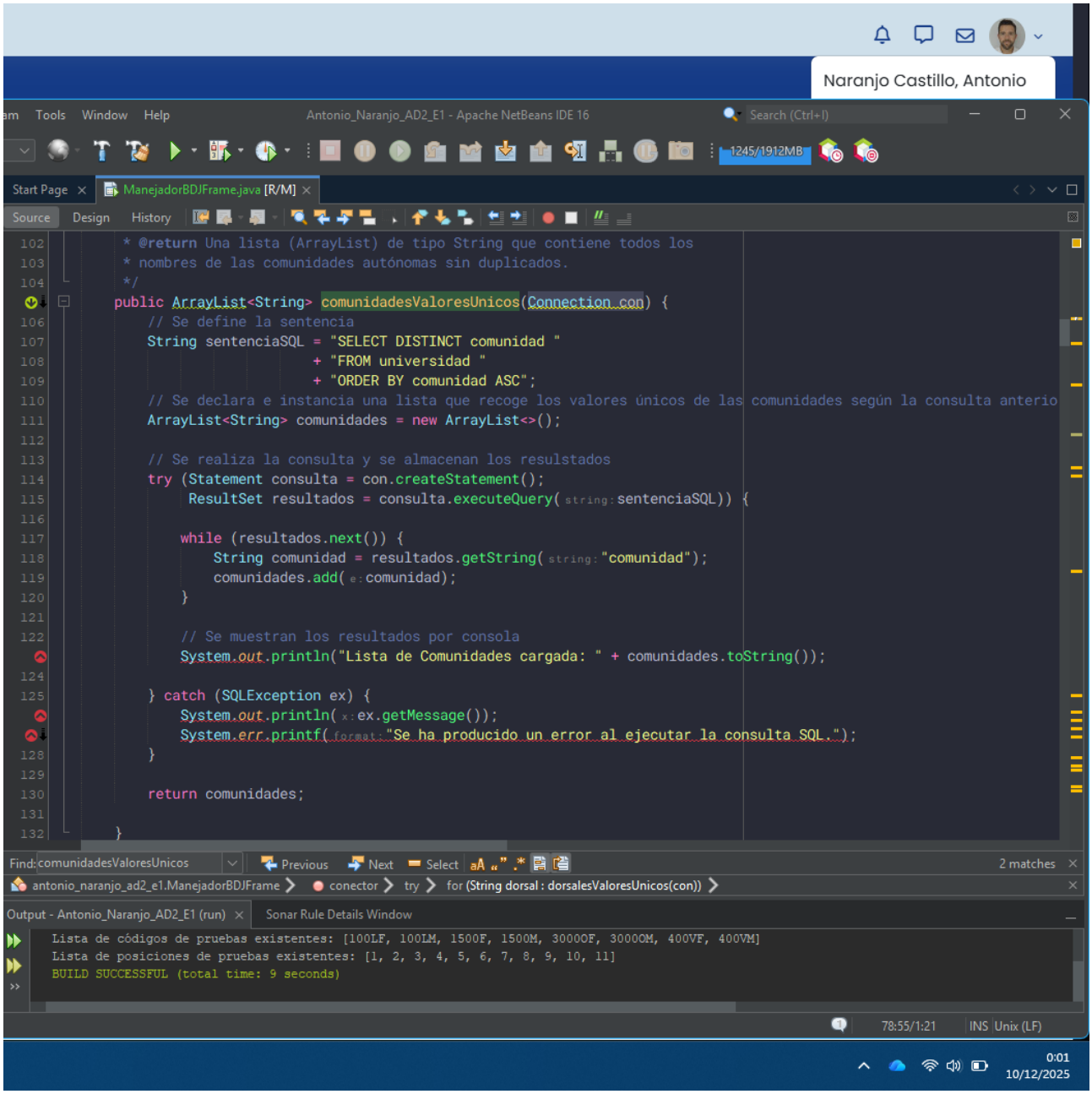
Se diseña una interfaz usando Swing con un panel reservado para establecer la conexión, incluyendo campos de texto para introducir usuario y contraseña, botones para activar/desactivar la conexión y una etiqueta para mostrar el resultado obtenido.



4. Consulta B1: Atletas por comunidad

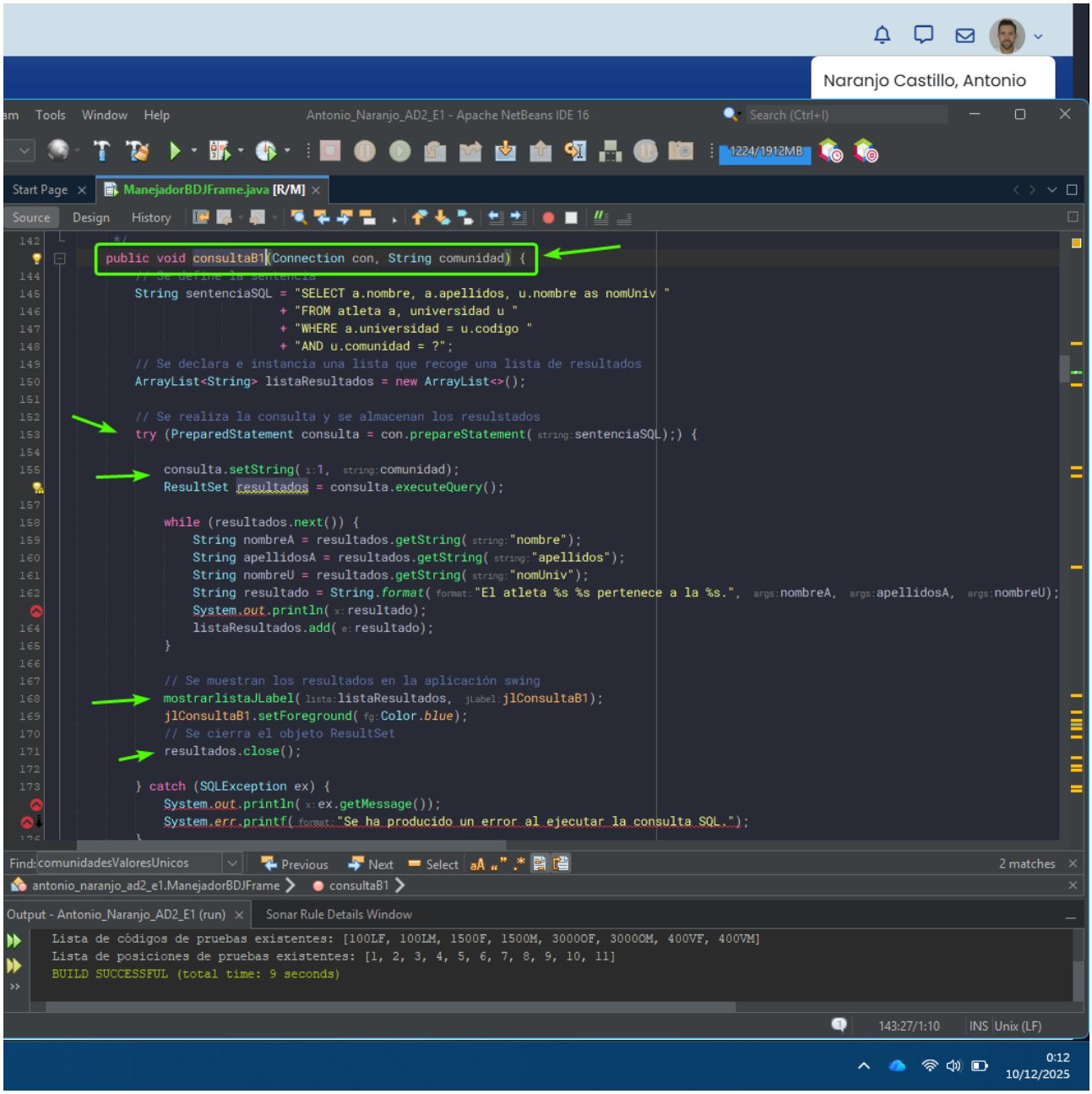
Se obtienen el nombre y apellidos de los atletas junto al nombre de la universidad a la que pertenece de todos aquellos atletas de la comunidad que indique el usuario de la aplicación.

Se preparan los ítems del ComboBox que podrá emplear el usuario de la aplicación para seleccionar la comunidad. Para ello se implementa un método llamado comunidadesValoresUnicos() al cual se le aporta como argumento el objeto Connection **con** creado mediante el método conector().



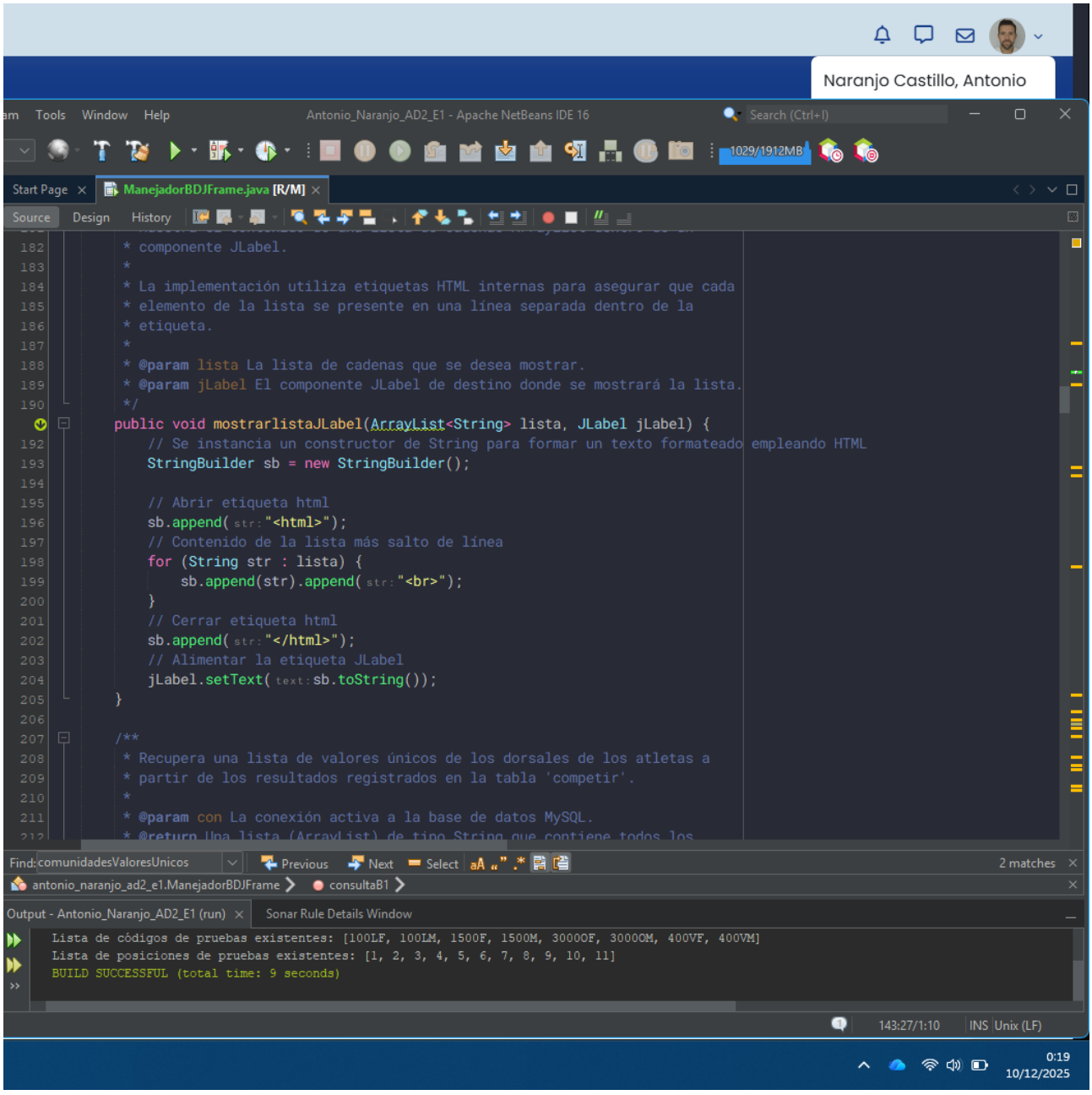
Se define la sentencia que selecciona los valores únicos de las comunidades que existen en la entidad universidad.
Se emplea la clase Statement y la clase ResultSet para crear la sentencia con la primera y recoger los datos de la consulta con la segunda. Finalmente, los datos obtenidos de almacenan en una lista de tipo ArrayList.

Una vez aliementado el comboBox se procede definiri el método que, de la misma manera, define la sentencia, ejecuta y presenta los resultados para obtener los nombres y apellidos de los atletas de una universidad que pertenece a la comunidad seleccionada por el usuario.
Este método se ha denominado consultaB1() y recibe como argumentos la conexión establecida y la comunidad seleccionado por el usuario.



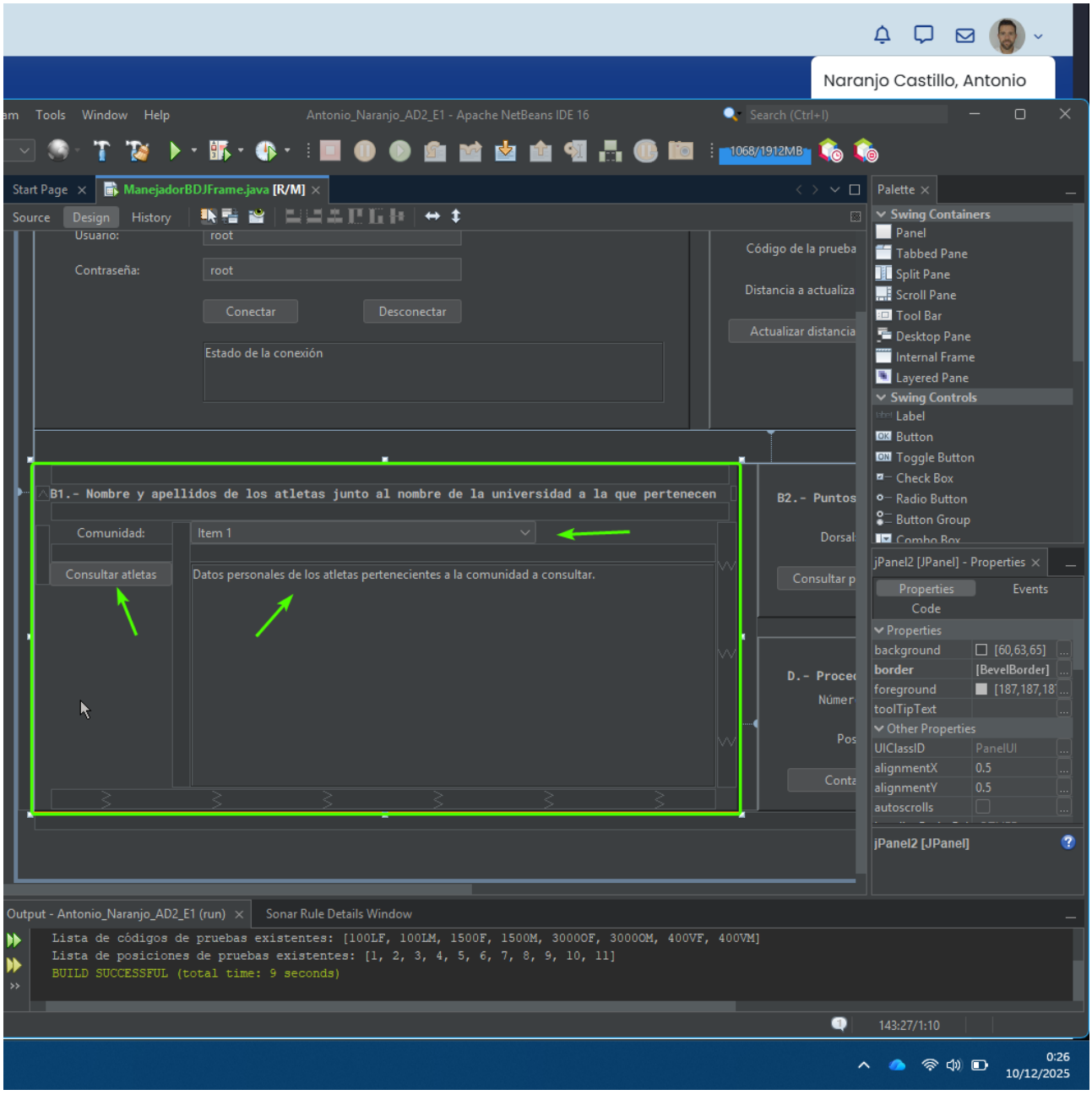
En este método se utiliza la clase PreparedStatement en lugar de la clase Statement, porque la sentencia SQL requiere de un parámetro de entrada para su definición, representado en la imagen anterior como un símbolo de interrogación.
Si la consulta se ejecuta con éxito se recogen los resultados mediante un objeto de la clase ResultSet, y mediante un bucle tipo while, mientras existan resultados que mostrar éstos se almacenarán en una lista que posteriormente se formateará para mostrarla en una etiqueta de la interfaz gráfica. Por último, se cierra el objeto resultados de la clase ResutlSet puesto que se encuentra fuera de la condición del bloque try-catch.

Para el formateo de los resultados se define un método denominado mostrarListaJLabel() que recibe como argumentos la lista donde se han almacenado los resultados de la consulta SQL anterior y la etiqueta donde se requiere que se muestren los datos formateados.



Se hace uso de un constructor de cadenas StringBuilder para formatear los datos de la lista ayudado de código HTML.

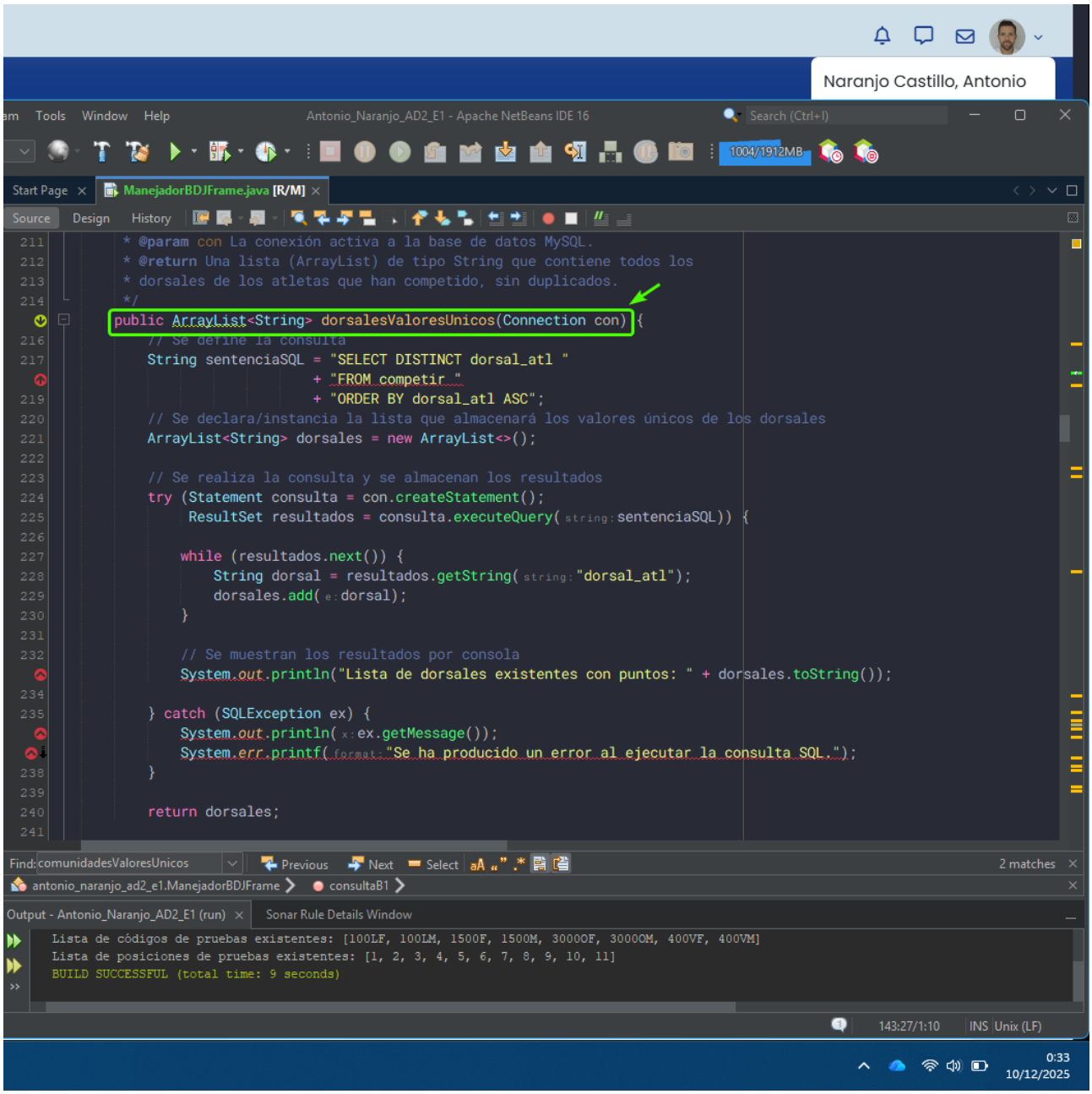
En la interfaz gráfica Swing, se asigna un panel para la consulta en cuestión, completado por un comboBox que recoge los valores únicos de las comunidades, un botón que ejecuta el método anterior consultaB1() y una etiqueta que presenta los resultados obtenidos de dicha consulta.



5. Consulta B2. Puntos totales por dorsal

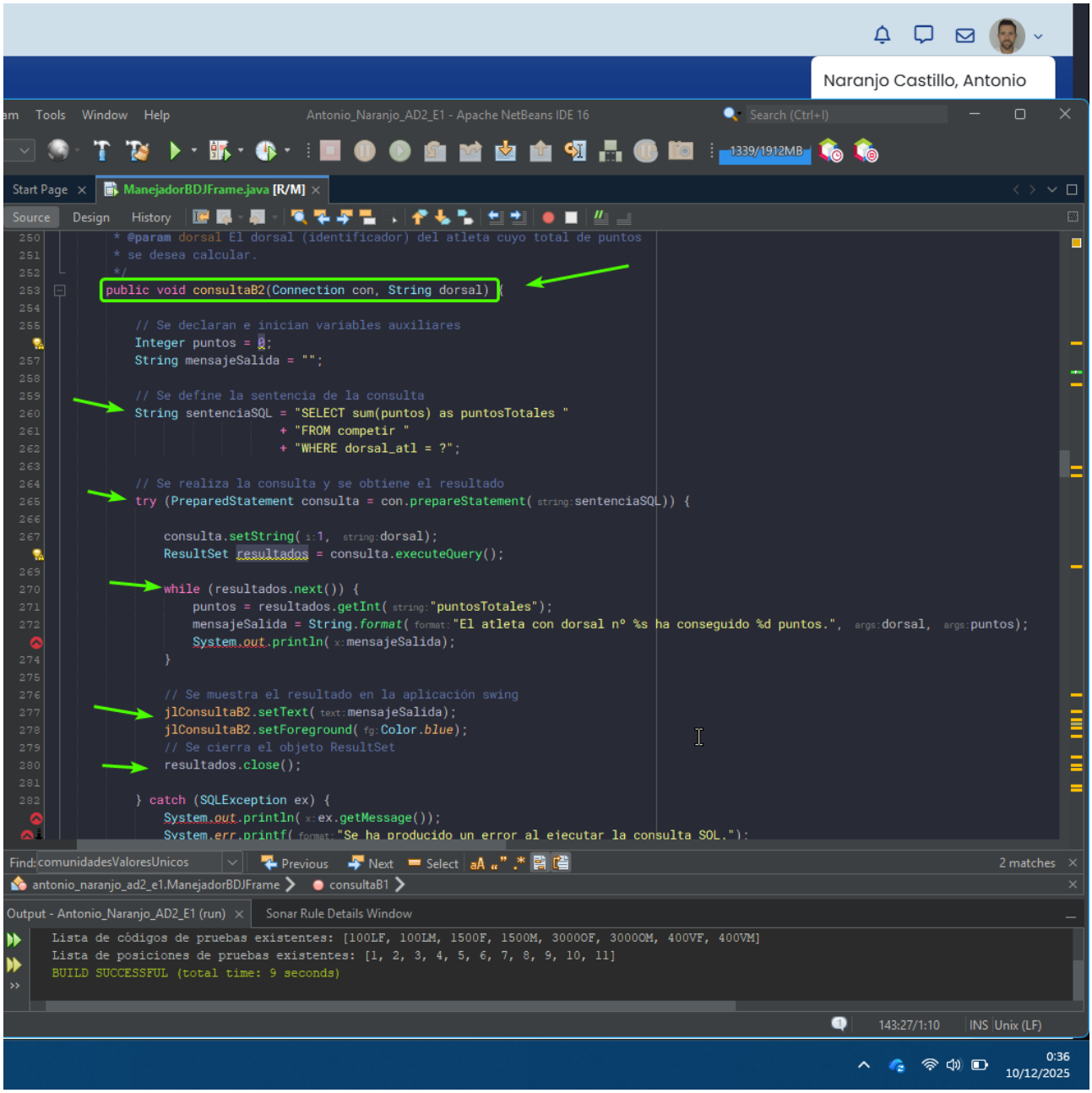
De manera similar a la anterior, se prepara los datos a listar por un comboBox, valores únicos de los números de dorsales ordenados de manera ascendente.

Para ello se ejecuta la sentencia SQL necesaria y se almacenan los datos obtenidos en una nueva lista.



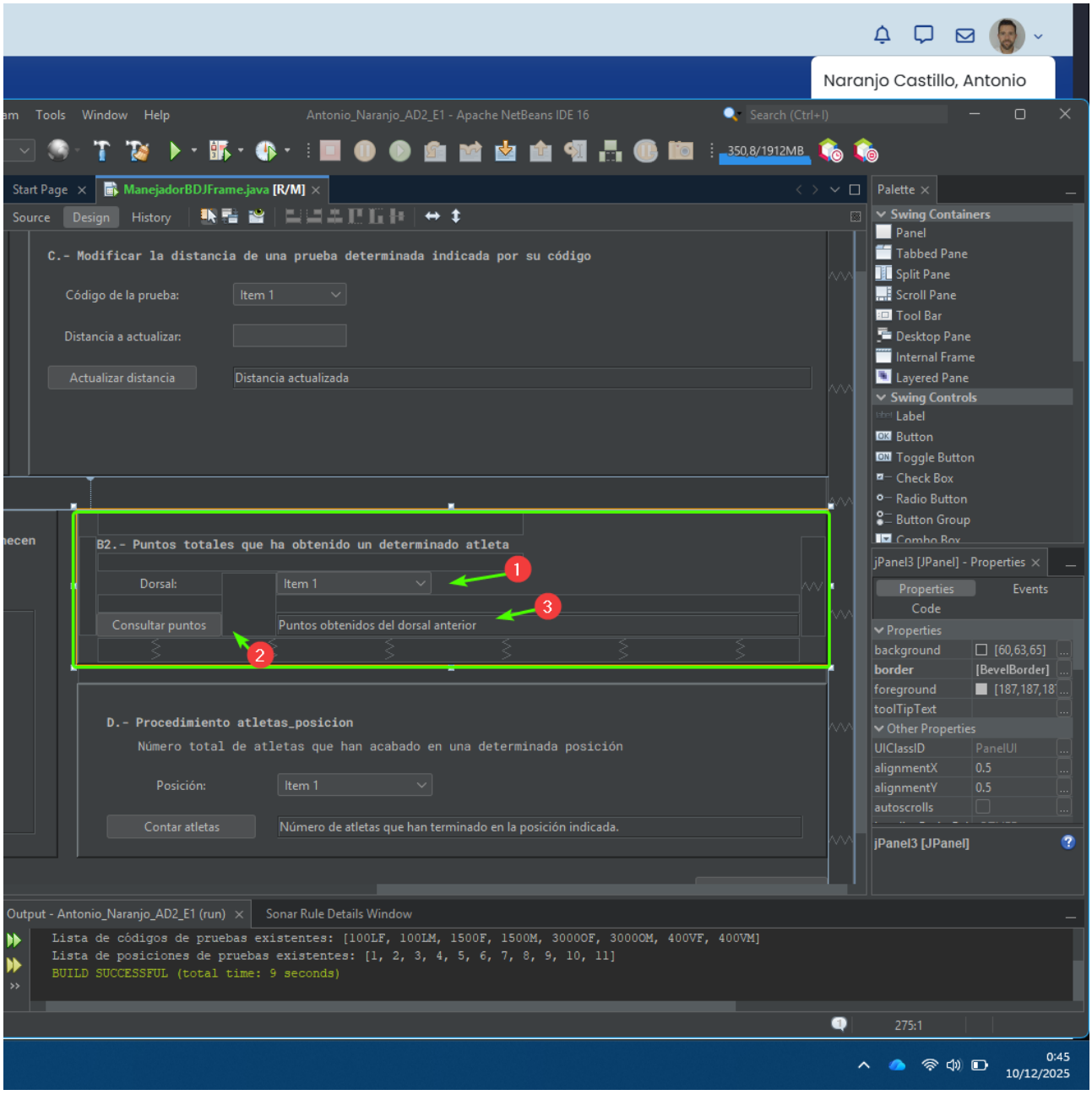
Al igual que anteriormente, se hacen uso de las clases Statement para ejecutar la sentencia y ResultSet para almacenar los datos obtenidos. En este caso no es necesario cerrar el objeto ResultSet porque se encuentra dentro de la condición del bloque try-catch.

Se implementa el método consultaB2() que recibe como argumento el objeto Connection y un String dorsal seleccionado por el usuario de la aplicación por medio del comboBox.



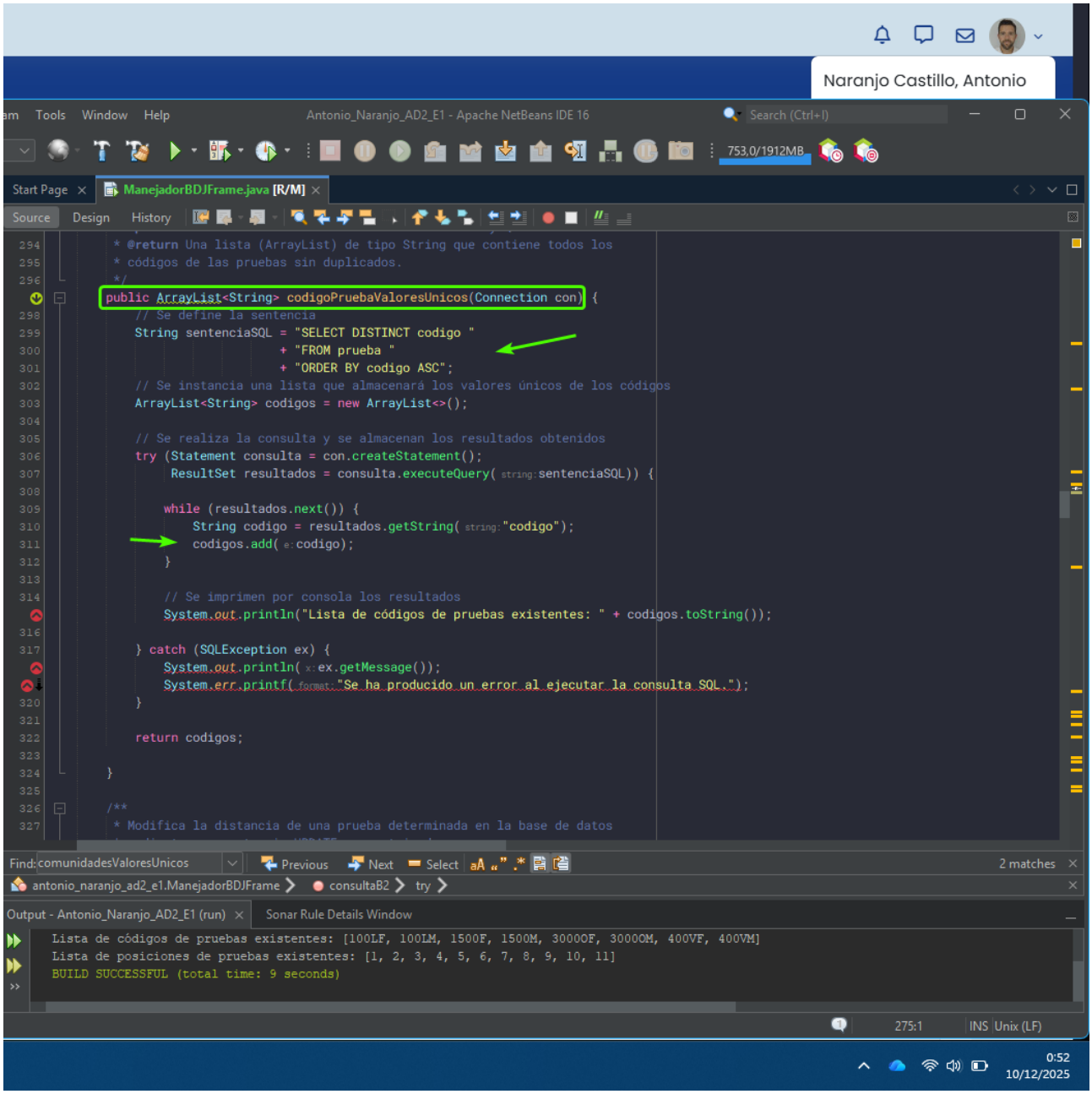
Se ejecuta la sentencia pasando como argumento el String dorsal anterior.
Todo de manera similar al método consulta anterior, pero en esta ocasión solo se obtiene un resultado por motivo de implementarse en la sentencia la función SUM que solo devolverá una fila.
El resultado obtenido se formatea y se muestra en una etiqueta de la interfaz gráfica.

En la interfaz gráfica se incorpora un nuevo panel para alojar en su interior el comboBox que recibe los valores únicos de los dorsales, un botón que ejecutará el método consultaB2() y una etiqueta que mostrará el resultado de la consulta, es decir, los puntos totales del dorsal seleccionado.



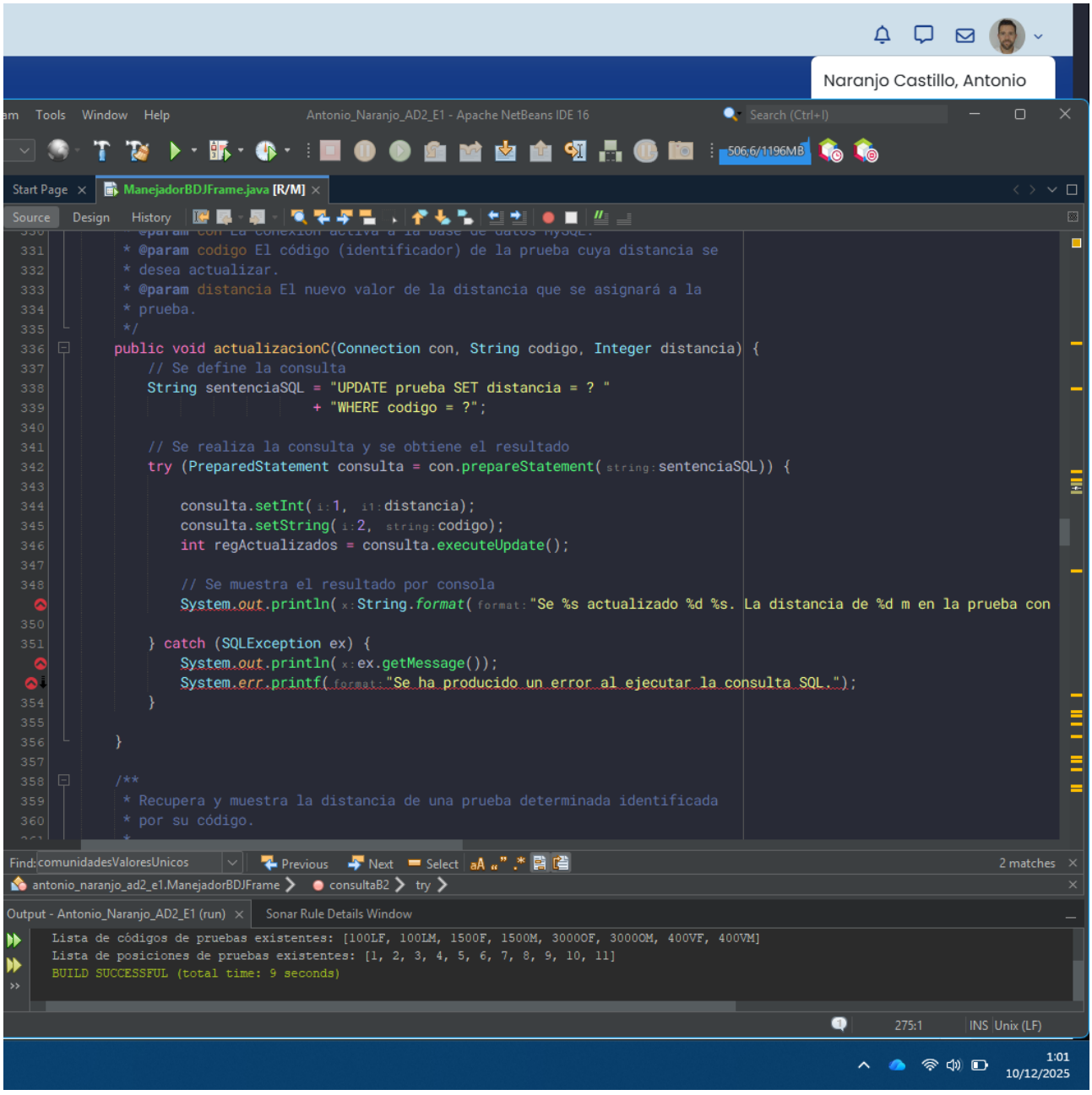
6. Actualización C. Modificación de la distancia de una prueba

De manera similar a casos anteriores, se alimenta un comboBox con una lista de valores únicos de códigos de prueba. Cabe decir que, el único atributo “unique” es el nombre de las universidades, por ello, las listas que alimentan los comboBox siempre se realiza una consulta de valores únicos y ordenados de manera ascendente. Más adelante se mostrará un anexo de imágenes aclaratorias y consultas realizadas en el cliente mysql en línea de comandos. También hay que destacar la importancia de fijar los datos a seleccionar por el usuario de la aplicación en un comboBox, descontando gran cantidad de posibles excepciones que deberían implementarse en caso de recibir los datos del usuario de la aplicación por medio de un campo de texto.



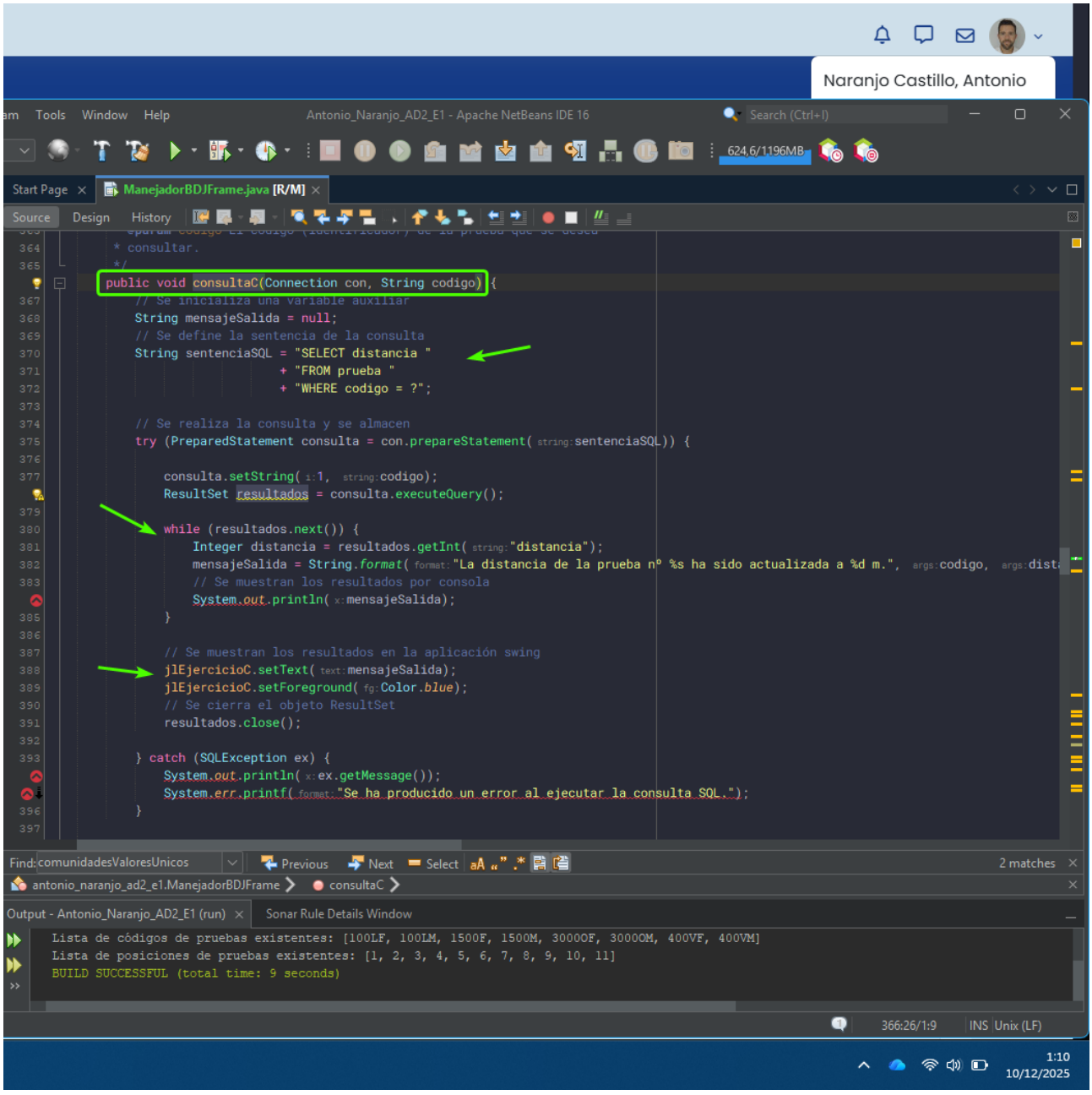
Como en casos anterior, finalmente los datos obtenidos se almacenan en la lista.

Se implementa el método actualizacionC que recibe como argumento la conexión, objeto con de la clase Connection, el código seleccionado por el usuario por medio del comboBox y la distancia a modificar o actualizar a través de un campo de texto o TextField.



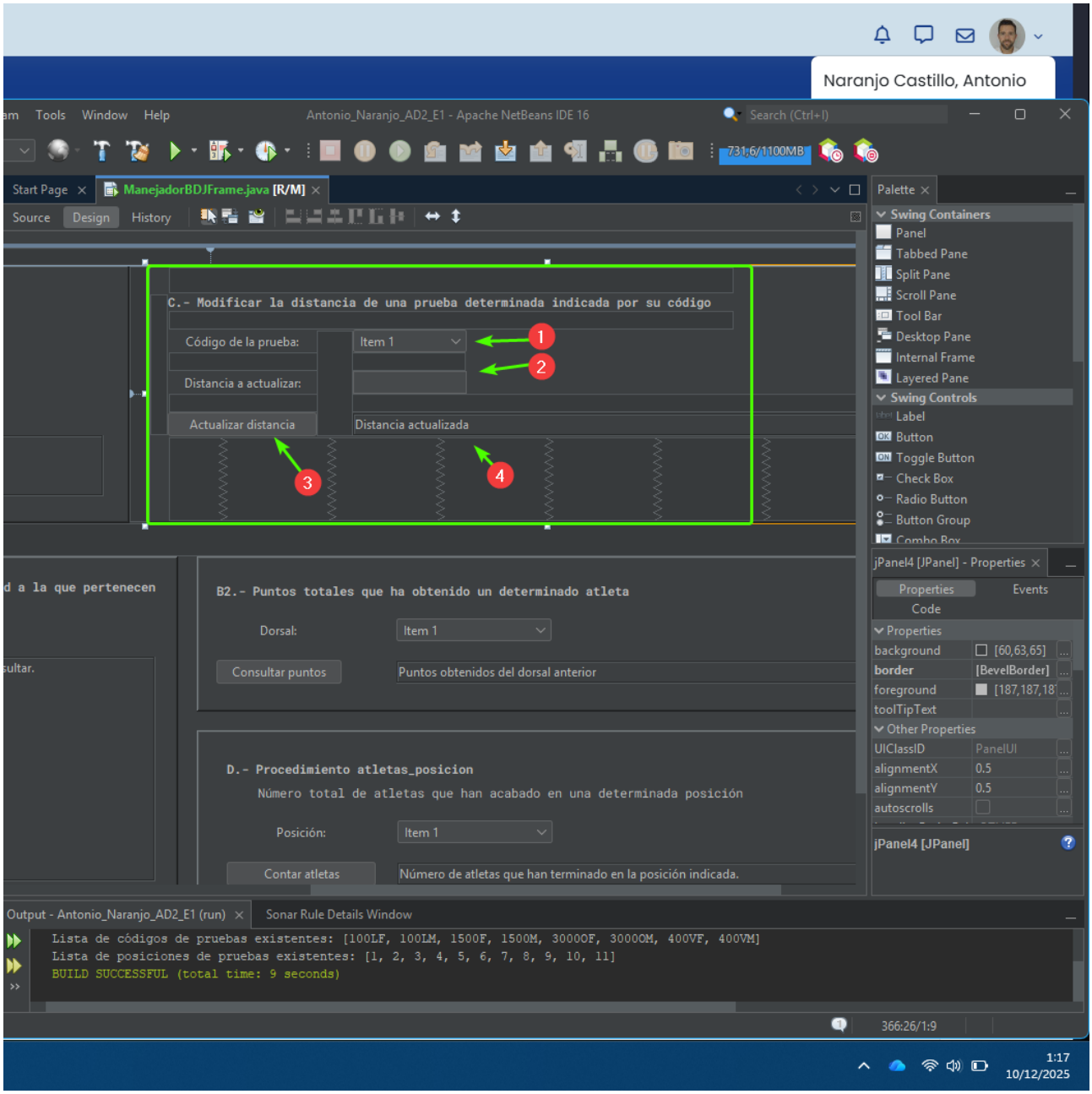
En esta ocasión el método recoge los registros actualizados y los muestra en consola.

Una vez ejecutado el código del método anterior, se prodece a realizar una consulta SQL implementando el método consultaC() que, de manera similar a anteriores casos, recibe como parámetros el objeto Connection, y un String código.



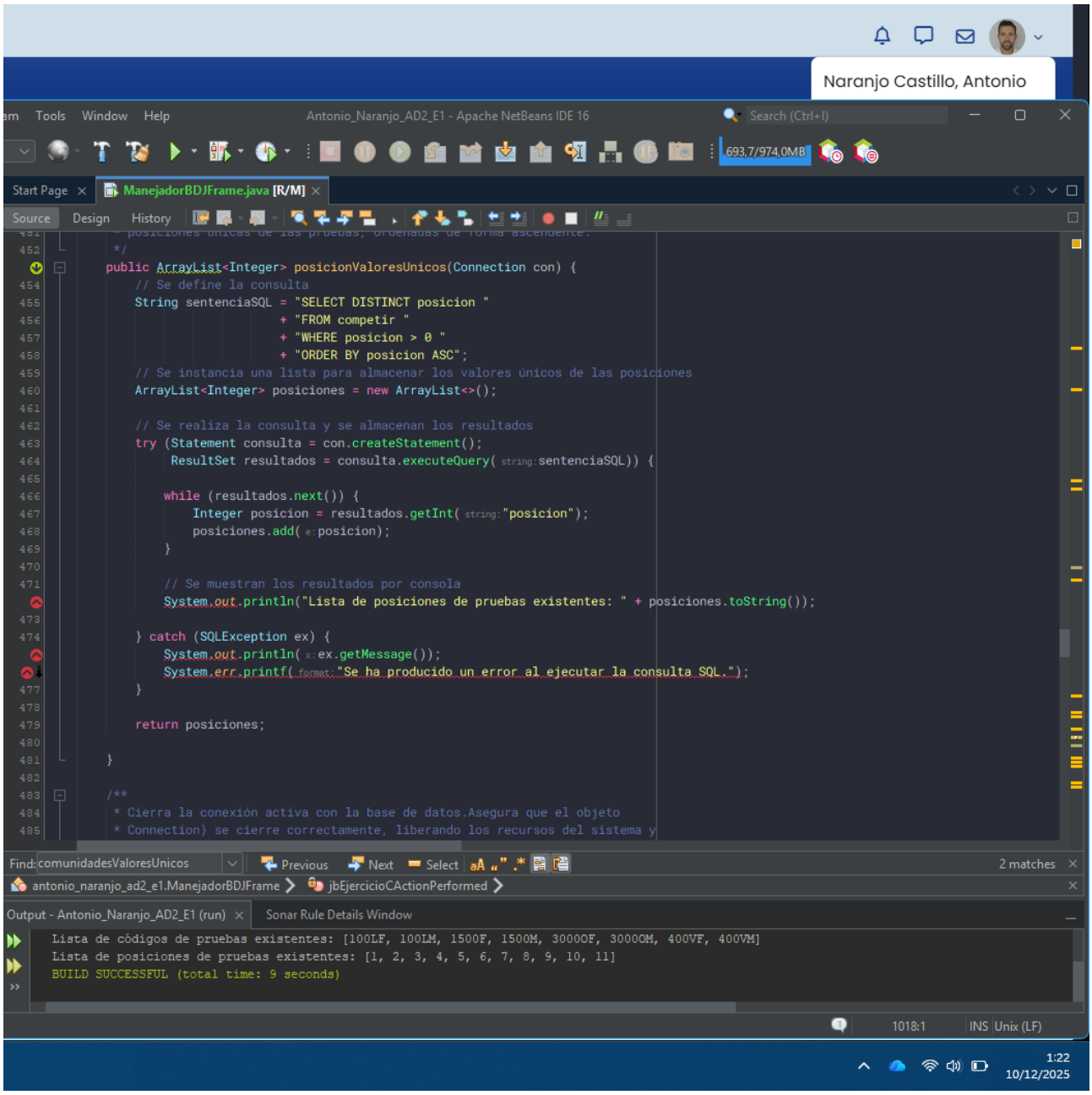
Se ejecuta la consulta que determina la nueva distancia que ha sido actualizada para una determinada prueba seleccionada a través de su clave primaria código.
Se recogen los datos obtenidos, en este caso una fila, y se muestra en la aplicación Swing, por medio de una etiqueta. Finalmente se cierra el objeto ResultSet.

En cuanto a la interfaz gráfica, se introduce un nuevo panel que contiene un comboBox para recibir los valores de los códigos de las distintas pruebas, un TextField para recibir la distancia a modificar requerida por el usuario de la aplicación, un botón para ejecutar tanto el método actualizacionC() como el método consultaC(), y por último, una etiqueta que mostrará el resultado de la actualización.



7. Procedimiento D. Ejecución de Procedimiento Almacenado

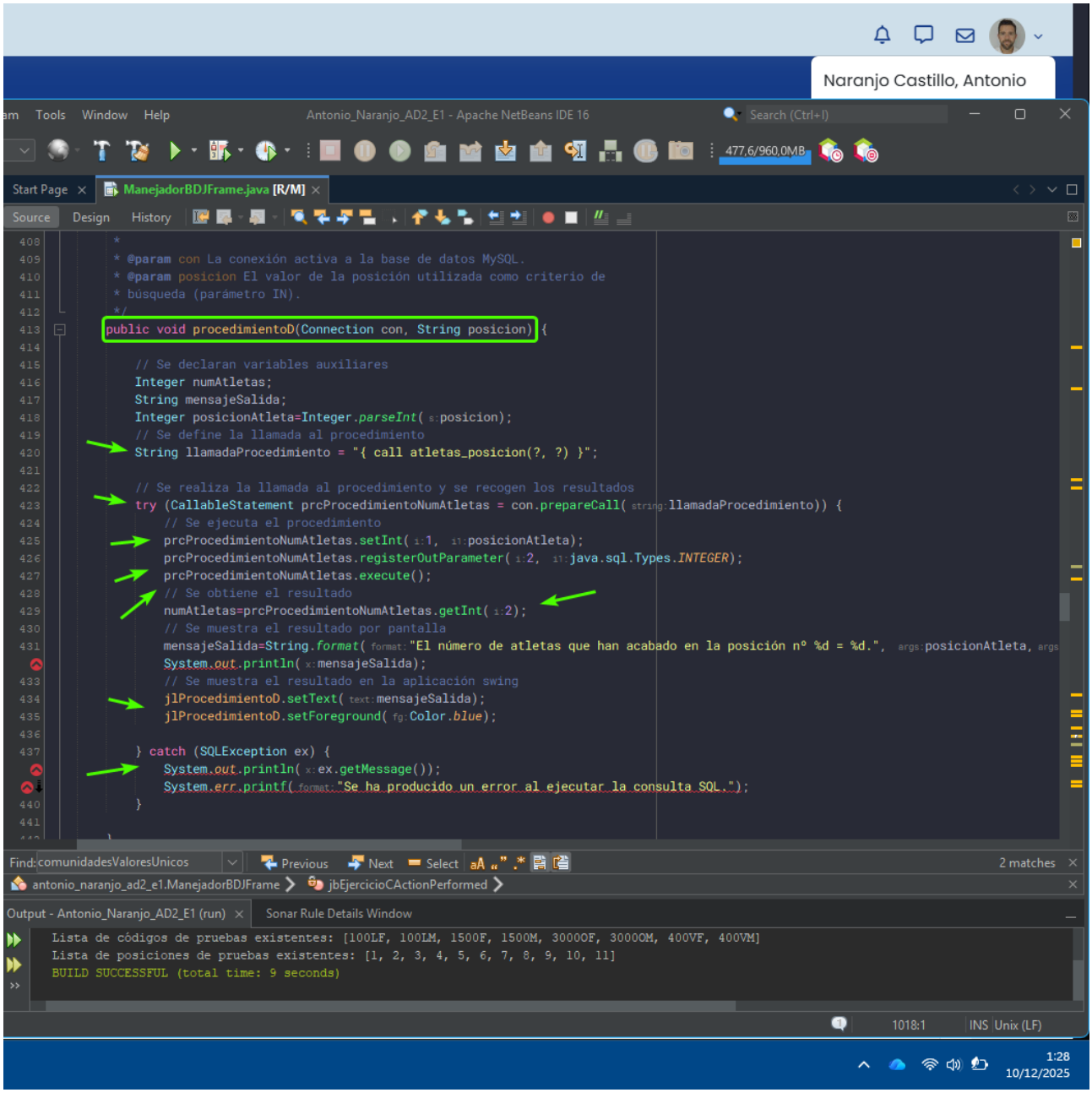
Como en anteriores ocasiones, se emplea un comboBox para recoger los datos de una lista que contiene los valores únicos ordenada ascendentemente de las posiciones de todas las pruebas.



La posición será seleccionada por el usuario de la aplicación, accediendo a las posiciones por medio del mencionado comboBox.

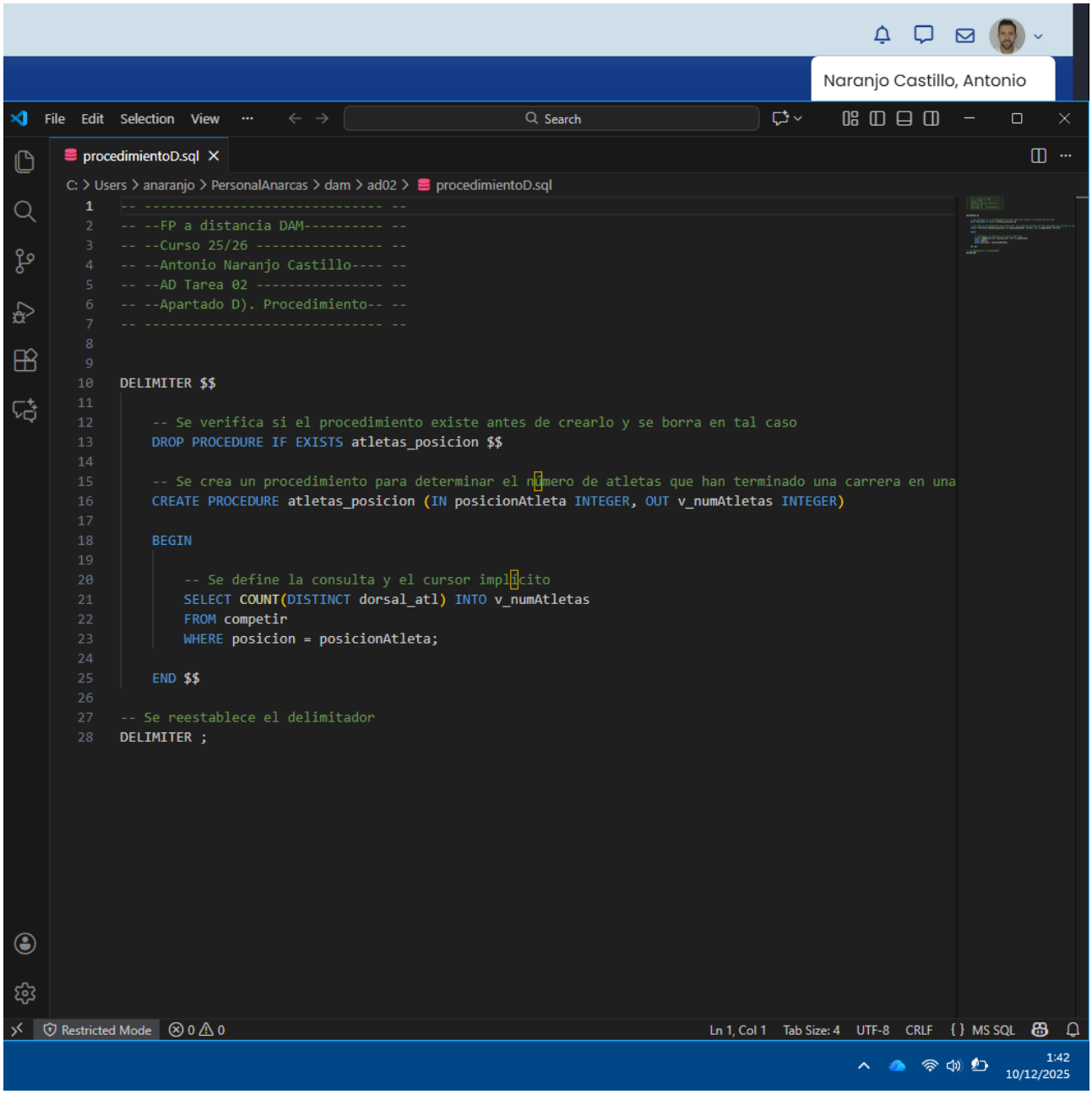
Se implementa el método procedimientoD() que recibe como argumentos el objeto Connection y un objeto String posicion.

En esta ocasión se almacena en un String la llamada al procedimiento atletas_posicion disponiendo de dos argumentos, uno de entrada IN y otro de salida OUT.
Se realiza la llamada al procedimiento mediante la clase CallableStatement, haciendo uso del método estático prepareCall de la clase Connection que recibe como parámetro el String llamada al procedimiento atletas_posicion.
Se establece el parámetro de entrada IN con el método setInt() de la clase CallableStatement y se recibe el parámetro de salida OUT con el método getInt() de la misma clase. Previamente, se definió el parámetro de salida mediante el método registerOutParameter() y se ejecutó el procedimiento con el método execute().
El resultado obtenido se formatea y se muestra tanto en consola como en una etiqueta de la interfaz grafica.



Finalmente, re recogen las posibles excepciones en caso de producirse un error al ejecutar la sentencia del procedimiento.

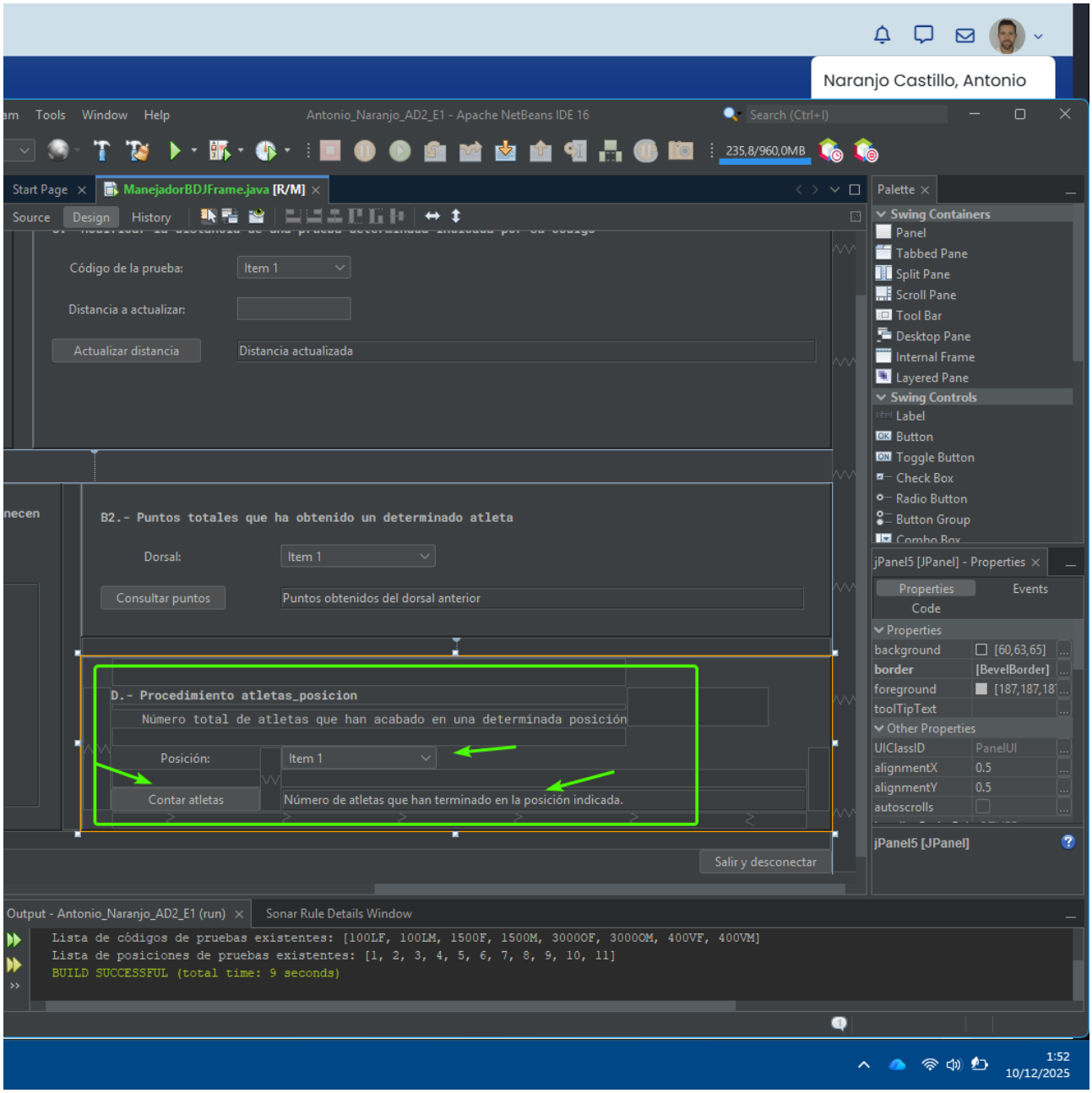
Previamente a la ejecución del método anterior, se tuvo que preparar la sentencia SQL del procedimiento atletas_posicion en un script e incorporarlo a la base de datos campeonato_atletismo. El código de dicho procedimiento se muestra a continuación.



Se incluyen las directivas DELIMITER y la cláusula DROP IF EXISTS para su correcta carga en el cliente mysql.

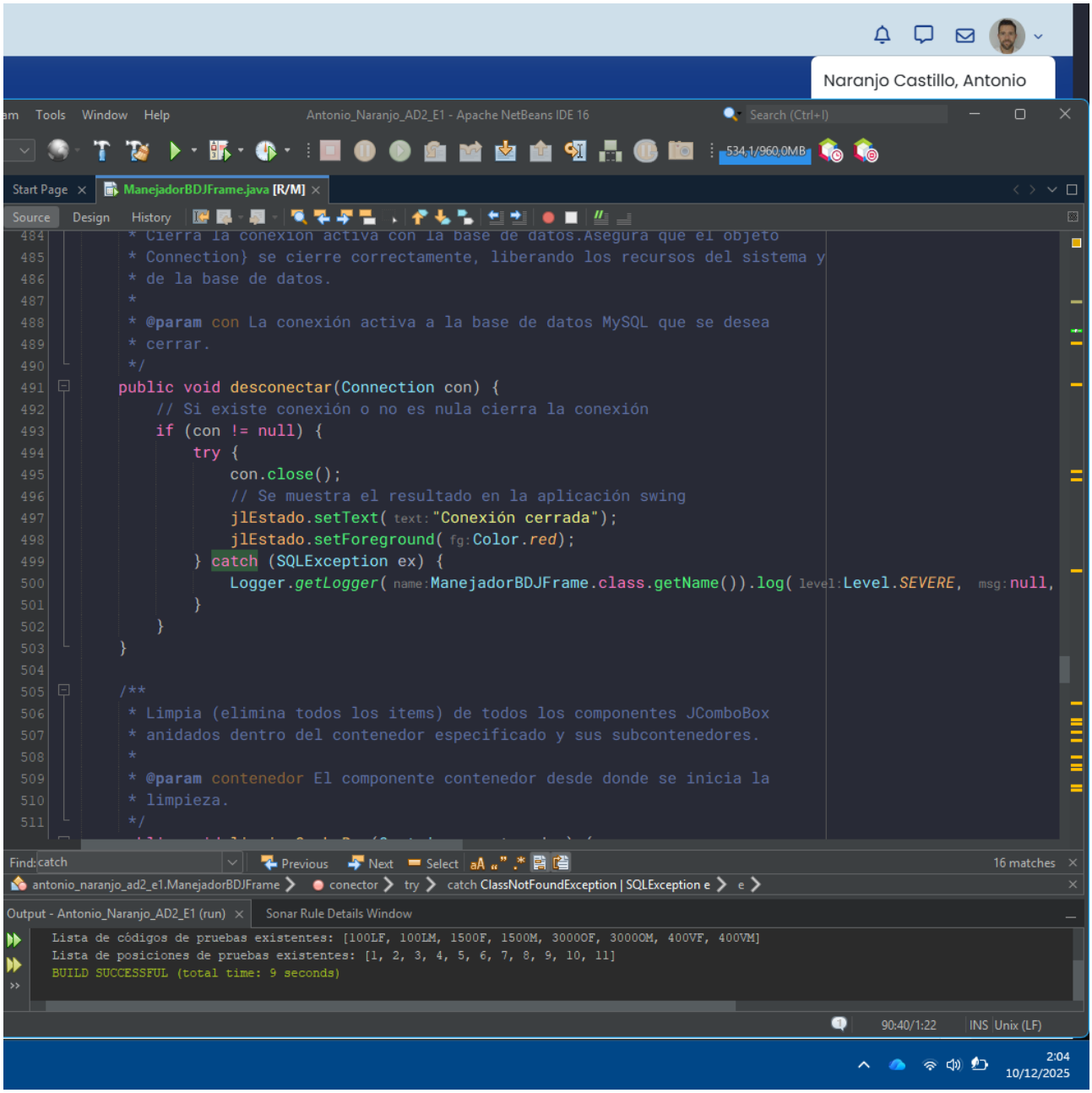
Se define una consulta que almacena su resultado único en un cursor implícito v_numAtletas, el cual será el parámetro de salida que recogerá nuestra aplicación tras ejecutar el método procedimiento D().

Por último, una vez más se establece una interfaz gráfica disponiendo de un panel contenedor para recibir componentes que simplifiquen la aplicación del procedimiento anterior por parte el usuario de la aplicación.
Se dispone de un comboBox que recibe los valores únicos de las posiciones obtenidas en cada una de las pruebas, el usuario seleccionará la posición que corresponda, un botón para ejecutar el procedimiento atletas_ posición y una etiqueta que presenta el valor del resultado del tal procedimiento.



8. Gestión de Excepciones y Cierre de Recursos

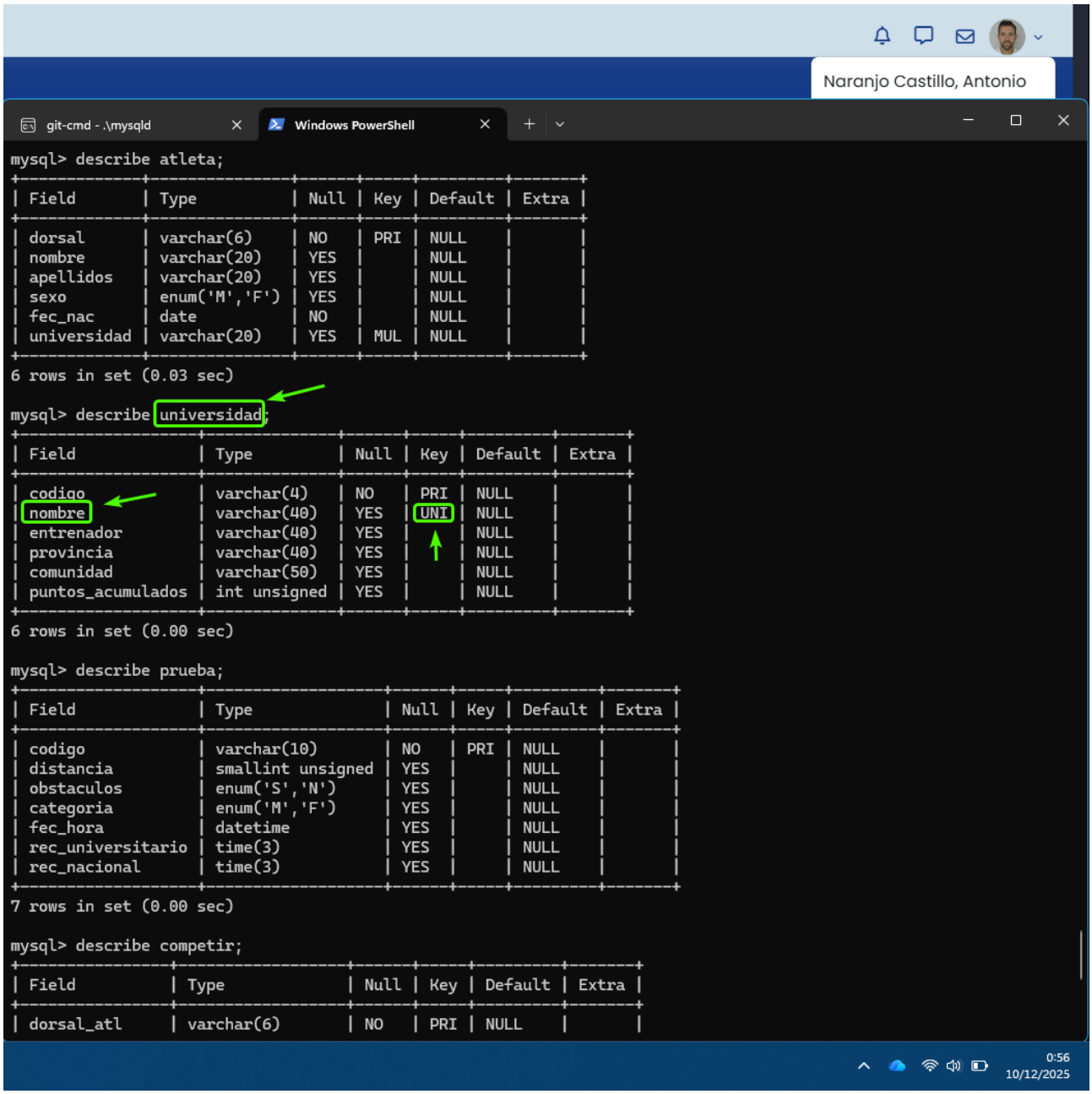
En cada una de las capturas anteriores se puede visutalizar el uso de excepciones para recoger los posibles problemas que se pudieran presentar empleando bloque try-catch para su manejo, del tipo SQLException en caso de errores en la ejecución de las sentencias SQL y ClassNotFoundException para el caso de no encontrar el driver correspondiente en el método conector().
Por otra parte, la correcta liberación de recursos ResultSet siempre y cuando no se encuentren en la condición del bloque try-catch y otros como Statement, PreparedStatement o CallableStatement siempre han estado dentro de dicha condición try-catch por lo que, se cierran de manera automática tras la ejecución del bloque try-catch.
También es necesario cerrar la conexión con la base de datos para liberar recursos, por ello se implementa el siguiente método desconectar().



9. Anexos

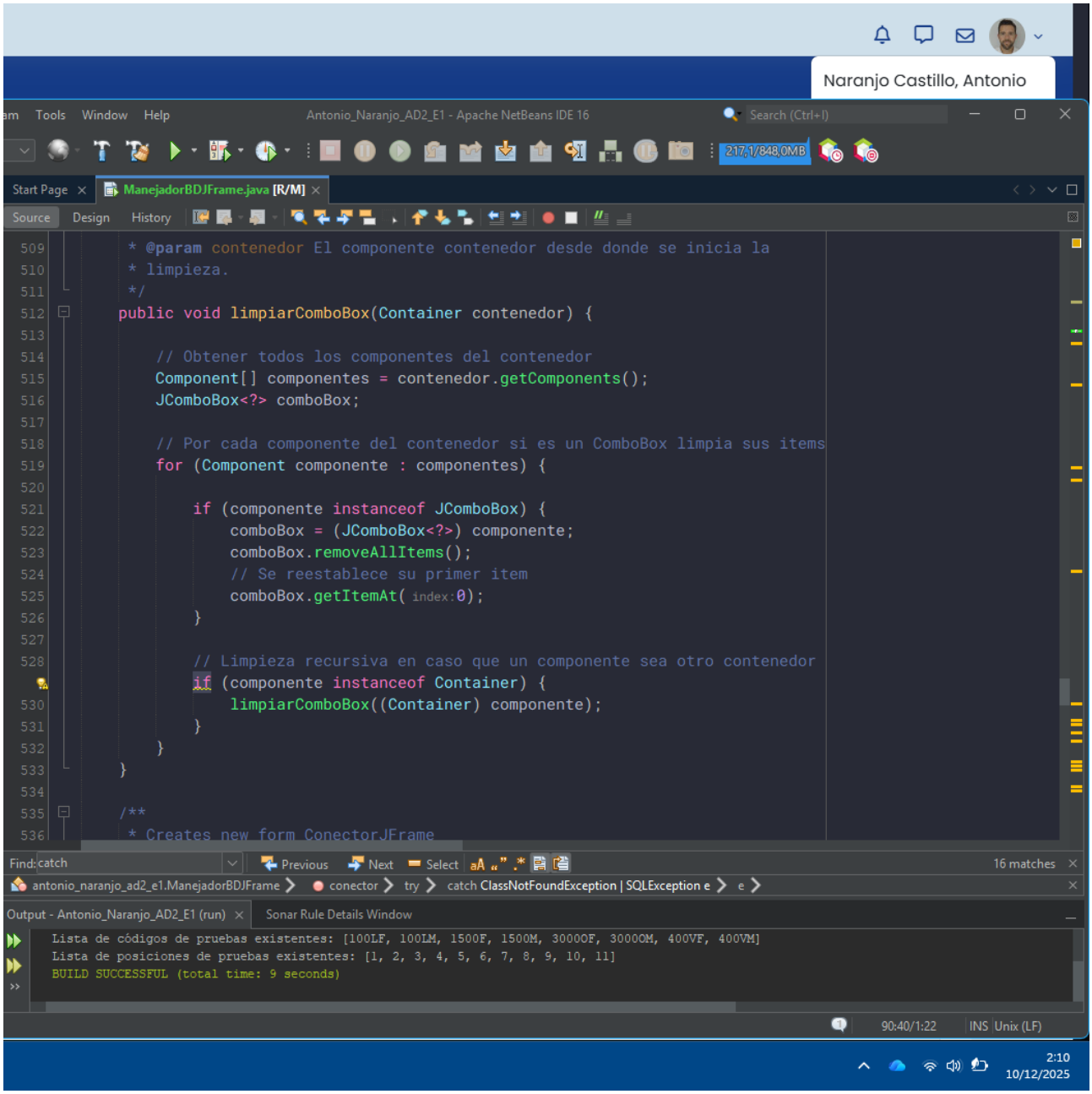
a) Atributos únicos

El único atributo definido como 'unique' es el nombre de las universidades. Importante conocer este dato a la hora de diseñar la sentencia correcta de cara a conseguir los valores únicos y ordenados para alimentar cada uno de los comboBox empleados en la interfaz gráfica Swing presentada en la tarea.



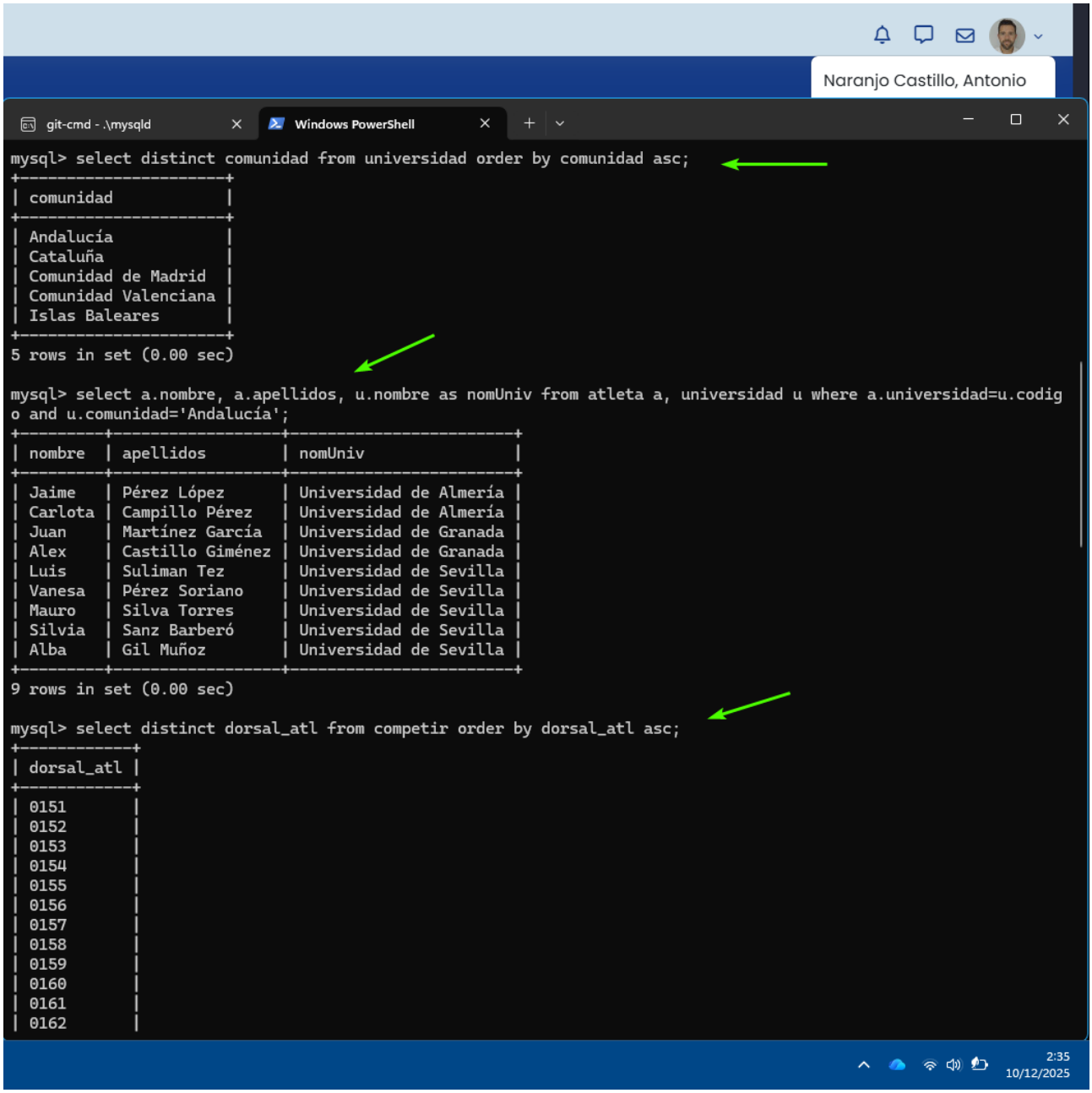
b) Limpieza de los objetos ComboBox

Se implementa un método complementario para limpiar cada uno de los objetos ComboBox que se disponen en la interfaz gráfica, actuando de manera recursiva sobre componentes que pertenecen a otros componentes, a modo de ejemplo, un comboBox que pertenece a un panel que a su vez pertenece a un frame.

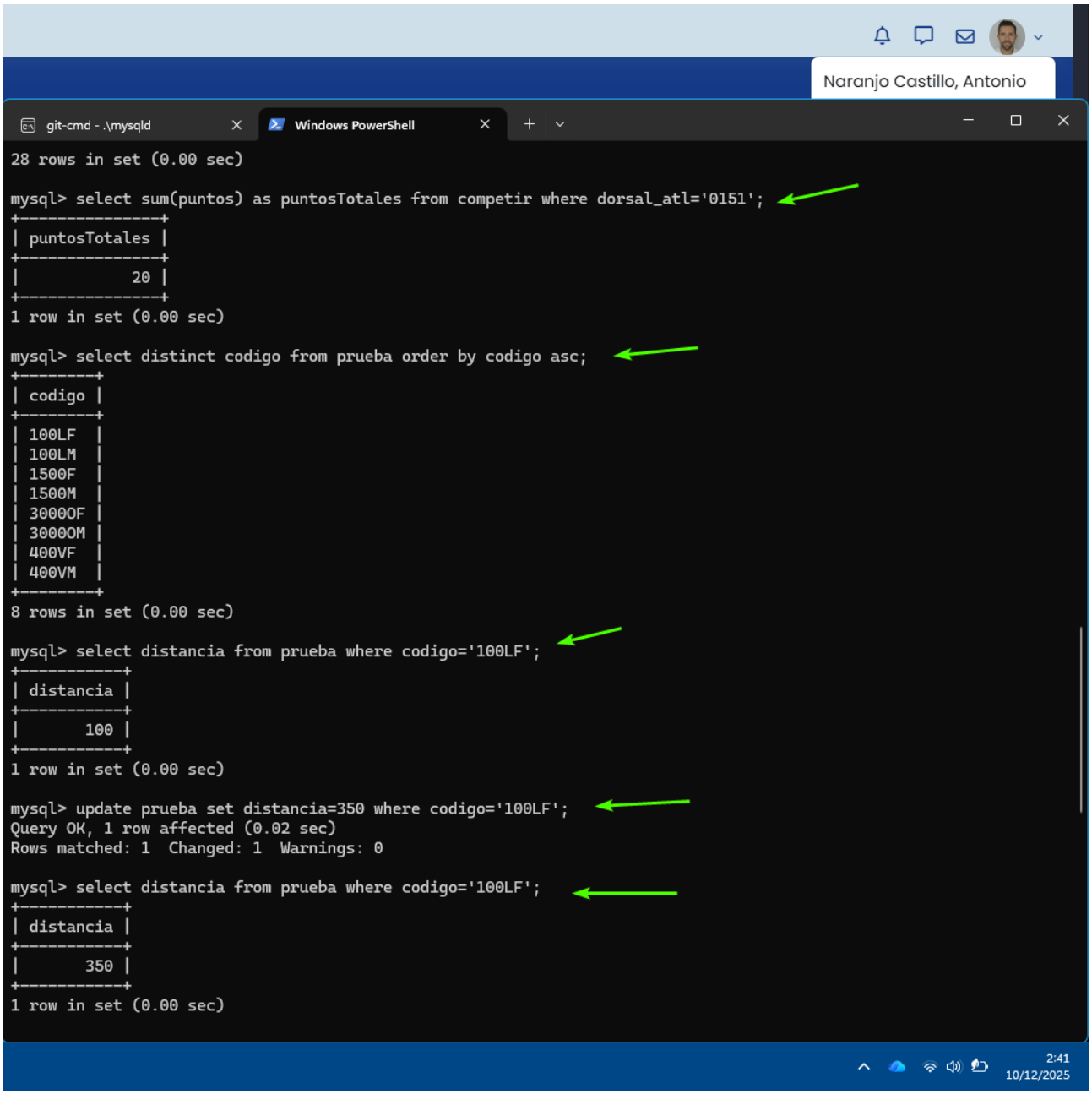


c) Sentencias SQL implementadas en el código java del presente proyecto netbeans

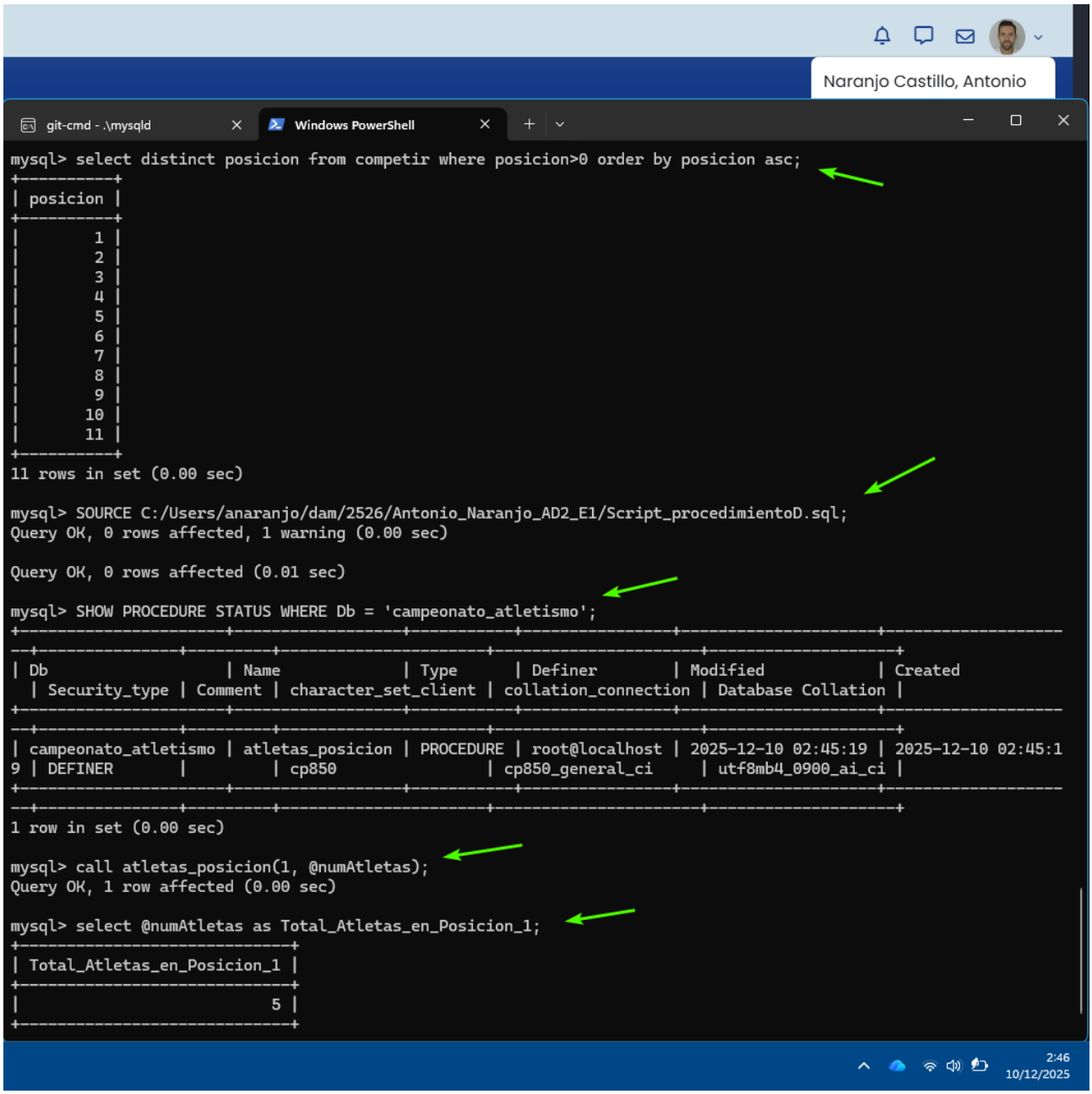
- Valores únicos de las comunidades por orden ascendente.
- Nombre y apellidos de los atletas de una determinada universidad que pertenece a la comunidad de Andalucía.
- Valores únicos de los dorsales por orden ascendente.



- Puntos totales del dorsal 0151.
- Valores únicos de los códigos de las distintas pruebas ordenados de manera ascendente.
- Distancia de la prueba con código 100LF.
- Modificación de la distancia de la prueba con código 100LF a 350.
- Comprobación de la distancia actualmente modificada.

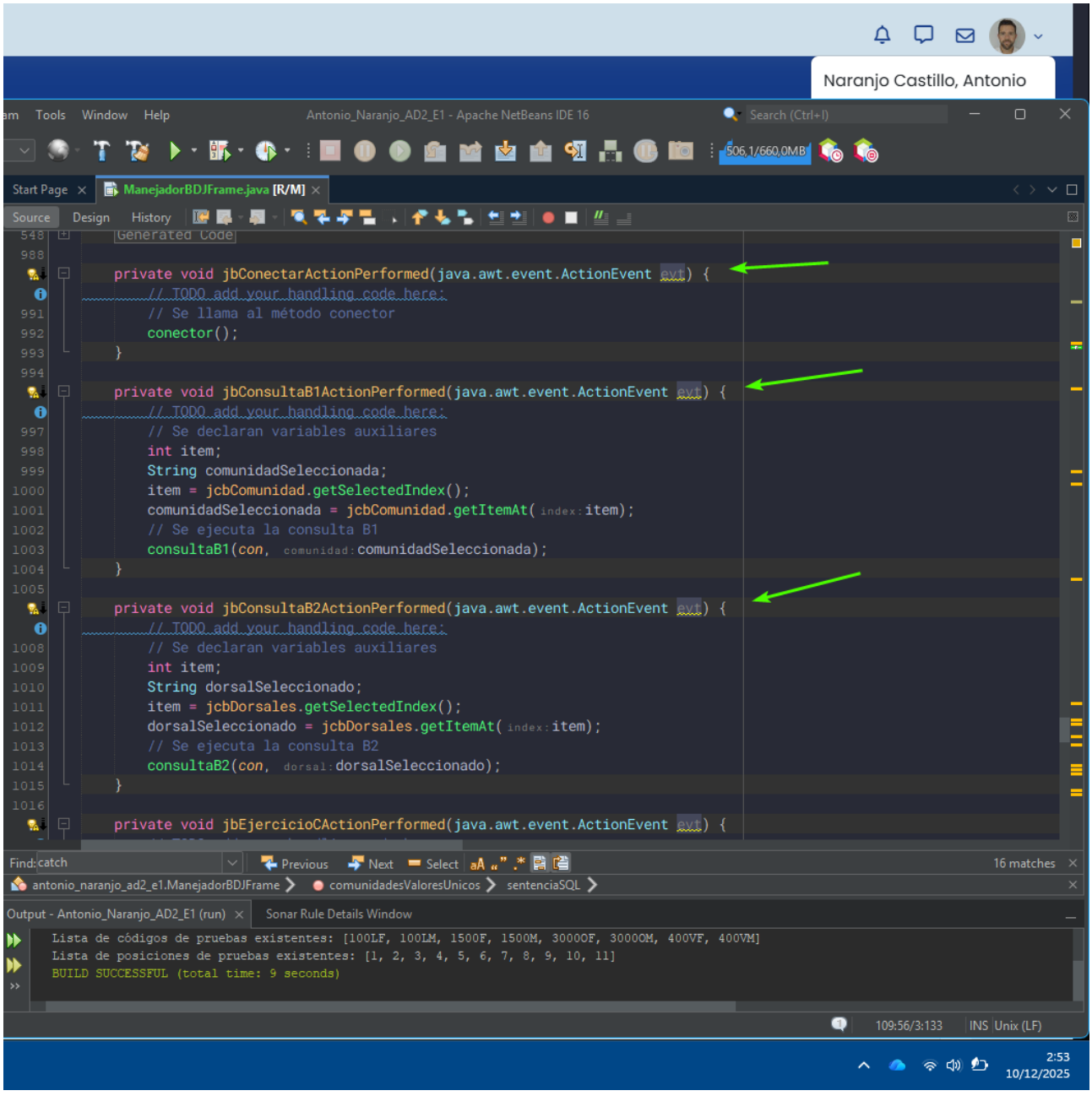


- Valores únicos de las posiciones mayores que cero y ordenadas ascendentemente.
- Se carga el script que contiene el procedimiento del apartado D.
- Se comprueba que está correctamente cargado en la base de datos.
- Se llama a tal procedimiento definiendo como parámetro de entrada la posición 1 y como parámetro de salida la variable @numAtletas
- Se consulta el valor adquirido por la variable @numAtletas una vez ejecutado el procedimiento.

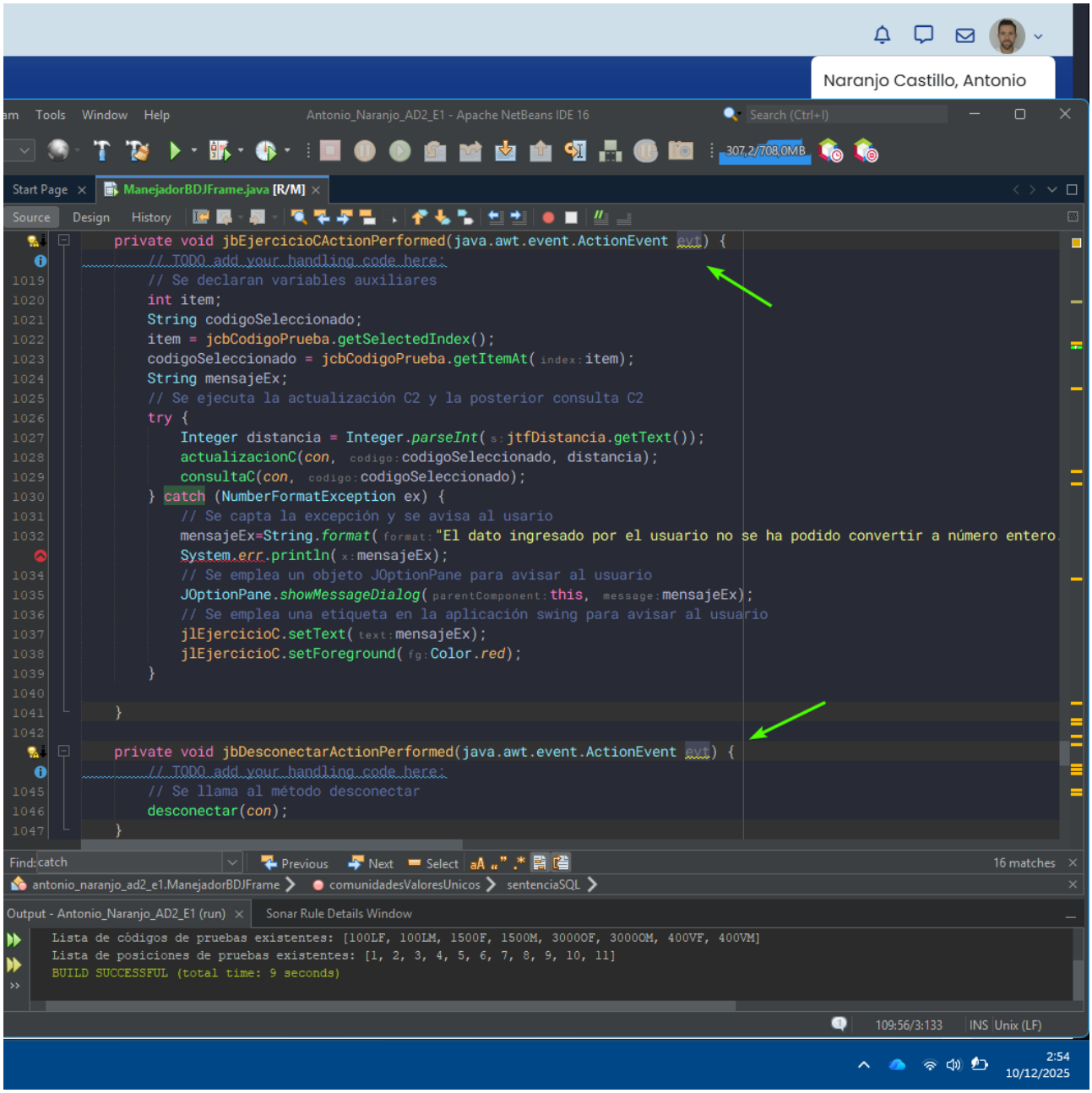


d) Código de cada uno de los botones de la interfaz gráfica

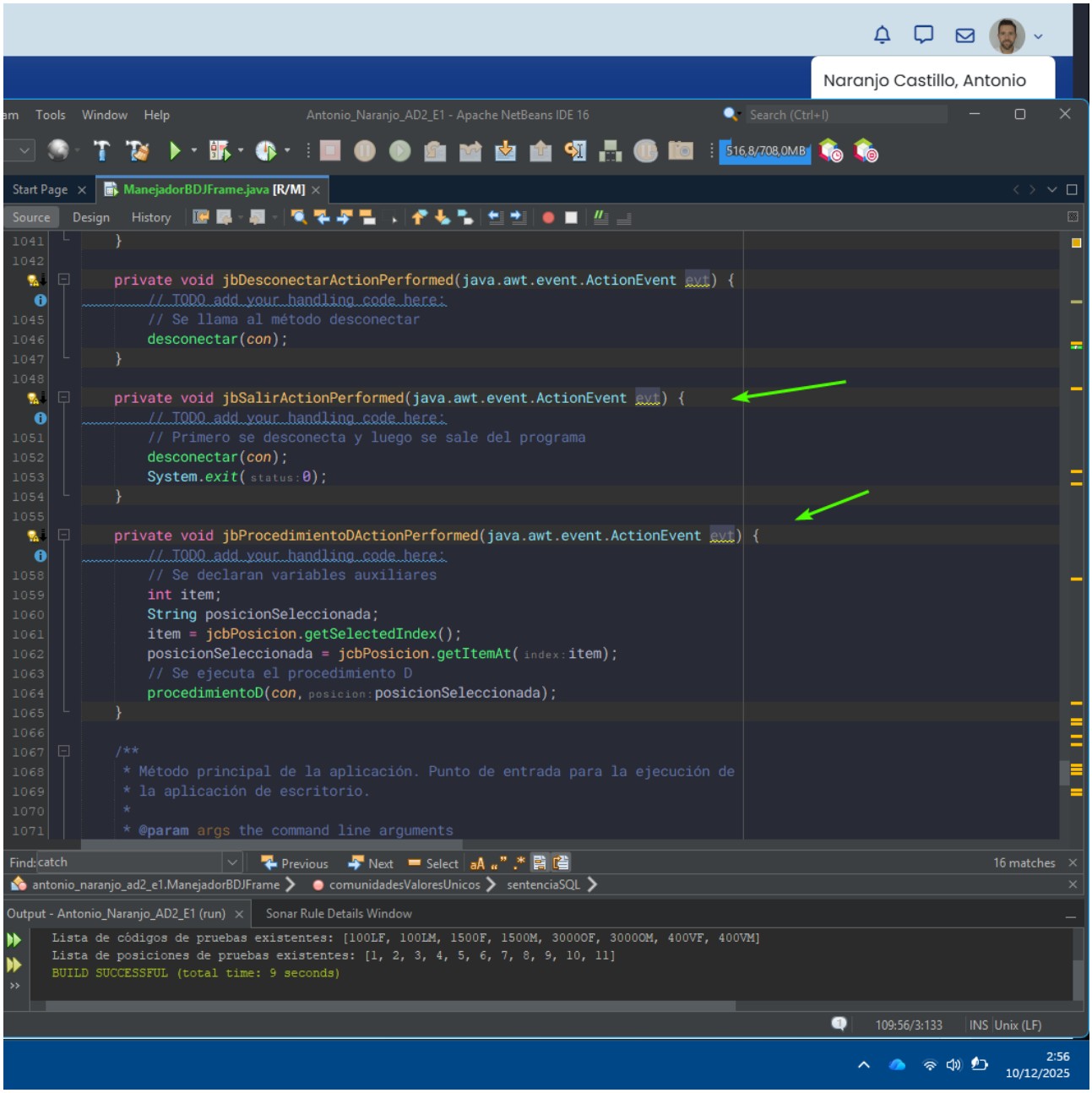
- Botón conectar Ejercicio A.
- Botón consulta Ejercicio B1.
- Botón consulta Ejercicio B2.



- Botón actualización y consulta Ejercicio C.
- Botón desconectar.



- Botón Salir y desconectar (para salir del programa).
- Botón procedimiento Ejercicio D.



e) Puesta en servicio de la aplicación

Se presenta la interfaz gráfica elaborada con Swing para el aplicativo de la tarea.

- Ventana superior izquierda: Ejercicio A, establecimiento de la conexión a la base de datos.
- Ventana inferior izquierda: Ejercicio B1.
- Ventana central derecha: Ejercicio B2.
- Ventana superior derecha: Ejercicio C.
- Ventana inferior derecha: Ejercicio D.