

Unidad 1

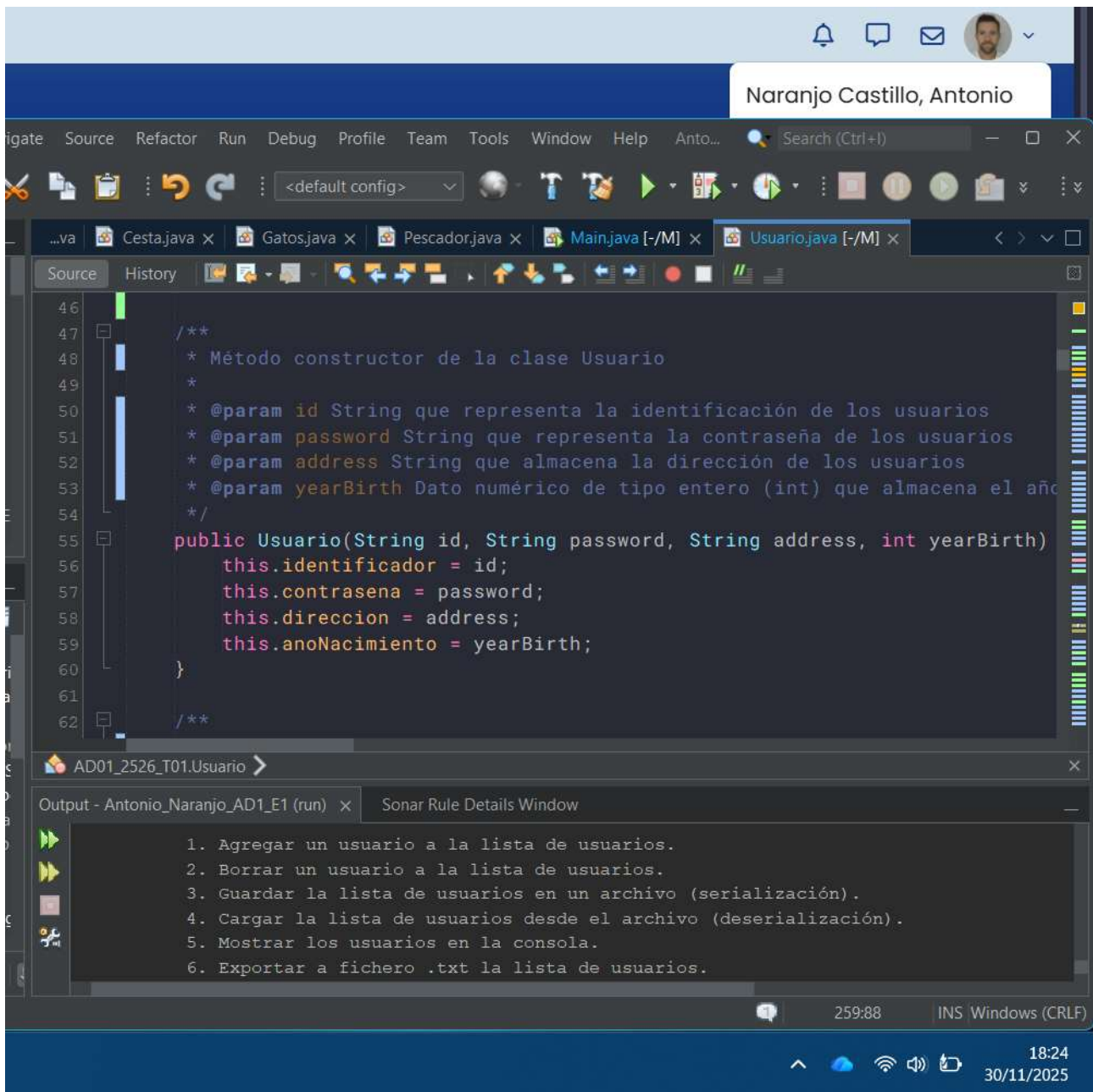
Introducción al acceso a datos y manejo de ficheros

Tarea AD01

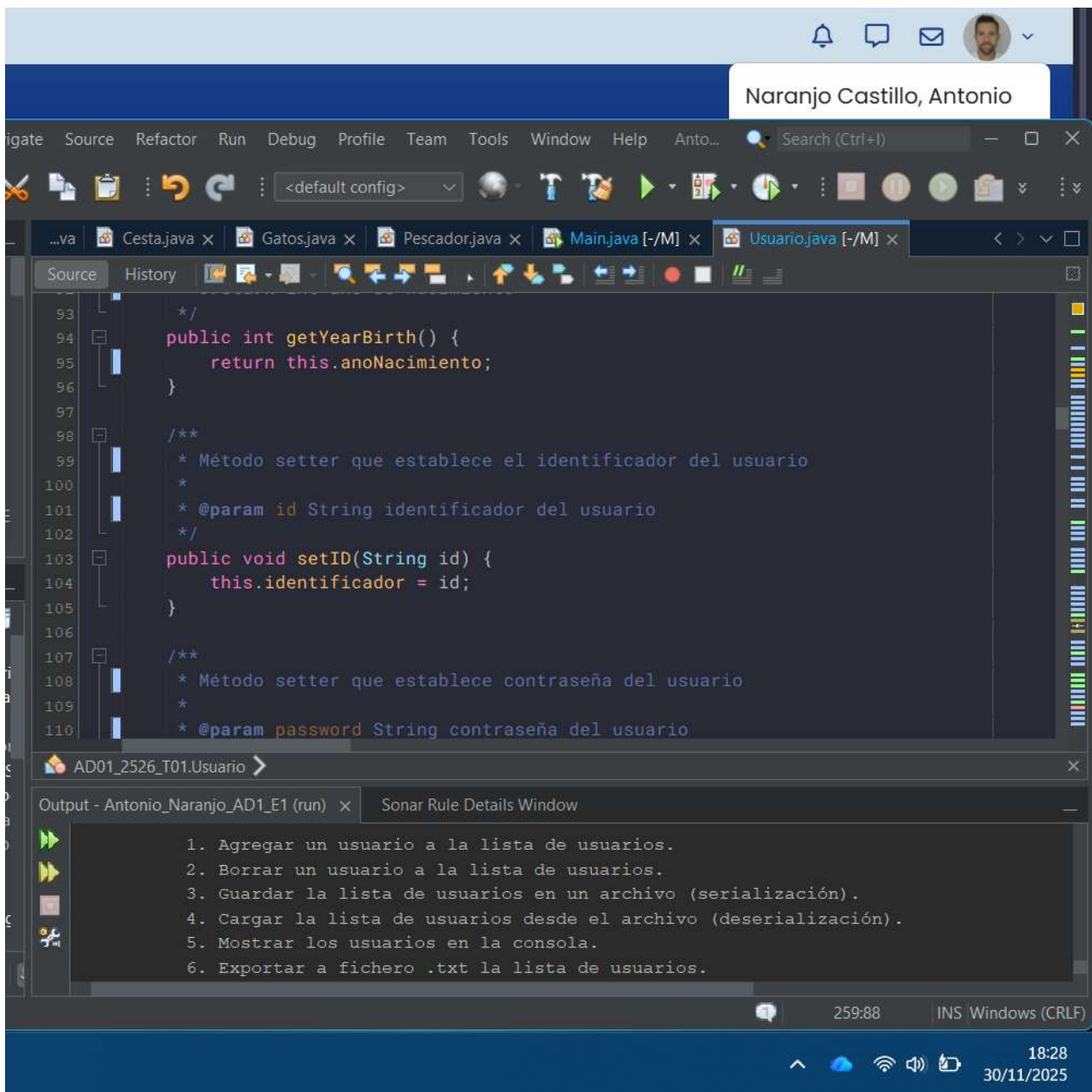
Manejando información almacenada en ficheros

1.	Crear una clase Usuario.....	2
2.	Implementación del menú	5
3.	Agregar un usuario a la lista de usuarios.....	6
4.	Borrar un usuario de la lista de usuarios.....	9
5.	Guardar la lista de usuarios en un archivo (serialización).....	11
6.	Cargar la lista de usuarios desde el archivo (deserialización)	13
7.	Mostrar los usuarios en consola	17
8.	Exporta a fichero .txt la lista de usuarios.....	19
9.	Salir de la aplicación	21
10.	¿Existe el archivo 'user.dat'?	22
11.	Advertencia sobre datos no guardados antes de salir	23
12.	¿Recuperar datos de disco?	24
13.	Anexo. Resultados del programa en consola.	25
a.	Inicio del programa	25
b.	Agregar un usuario	27
c.	Mostrar los usuarios en consola	28
d.	Borrar usuario	29
e.	Guardar lista (Serialización)	31
f.	Cargar lista (deserialización).....	32
g.	Exportación archivo TXT.....	33
h.	¿Cargar datos guardados?.....	34
i.	Salida del programa.....	36

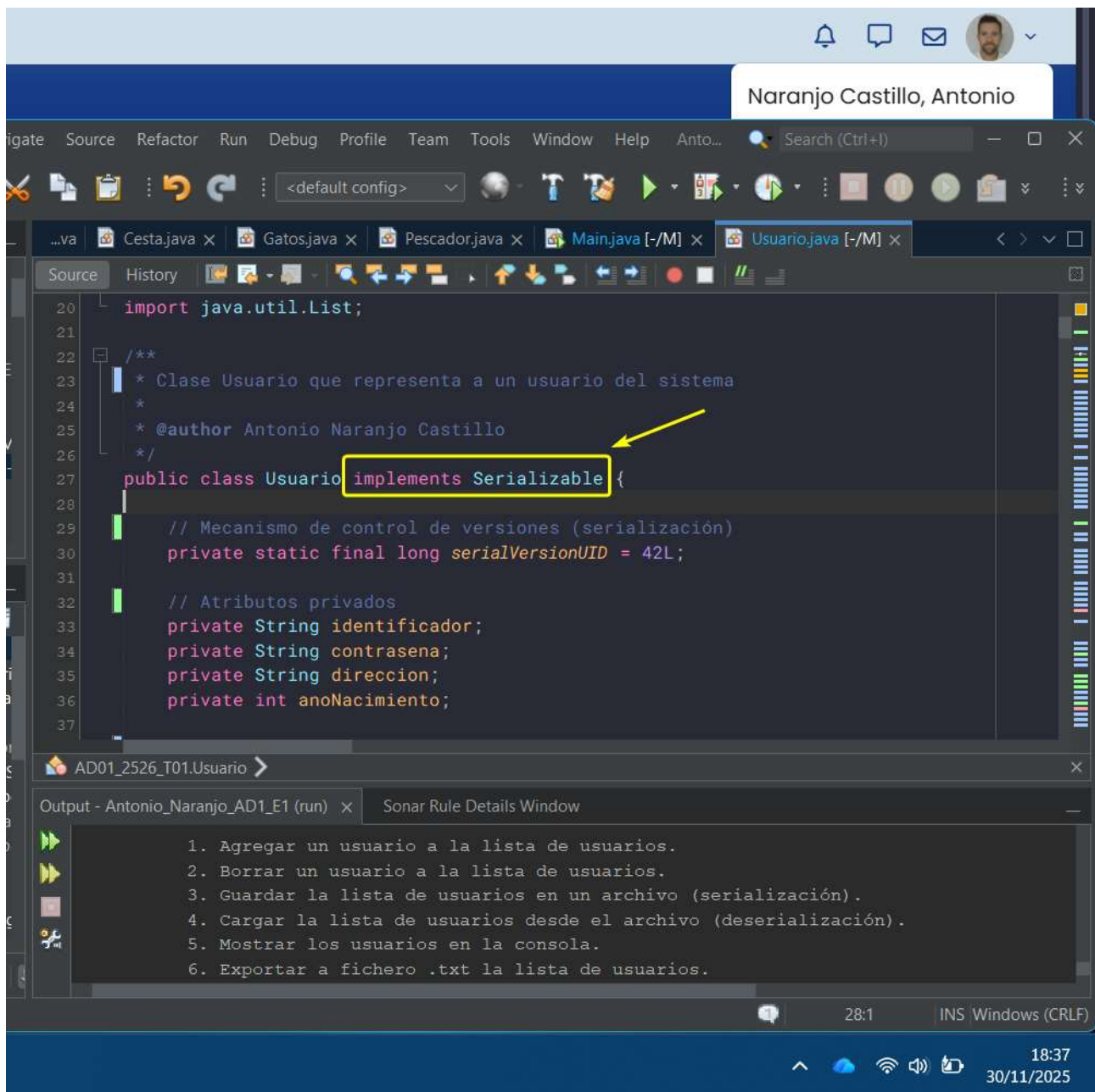
1. Crear una clase Usuario



Se crea una clase `Usuario` implementando el método constructor que define los cuatro atributos solicitados en la tarea, identificador (de tipo `String`), contraseña (de tipo `String`), dirección (de tipo `String`) y año de nacimiento (de tipo `int`).

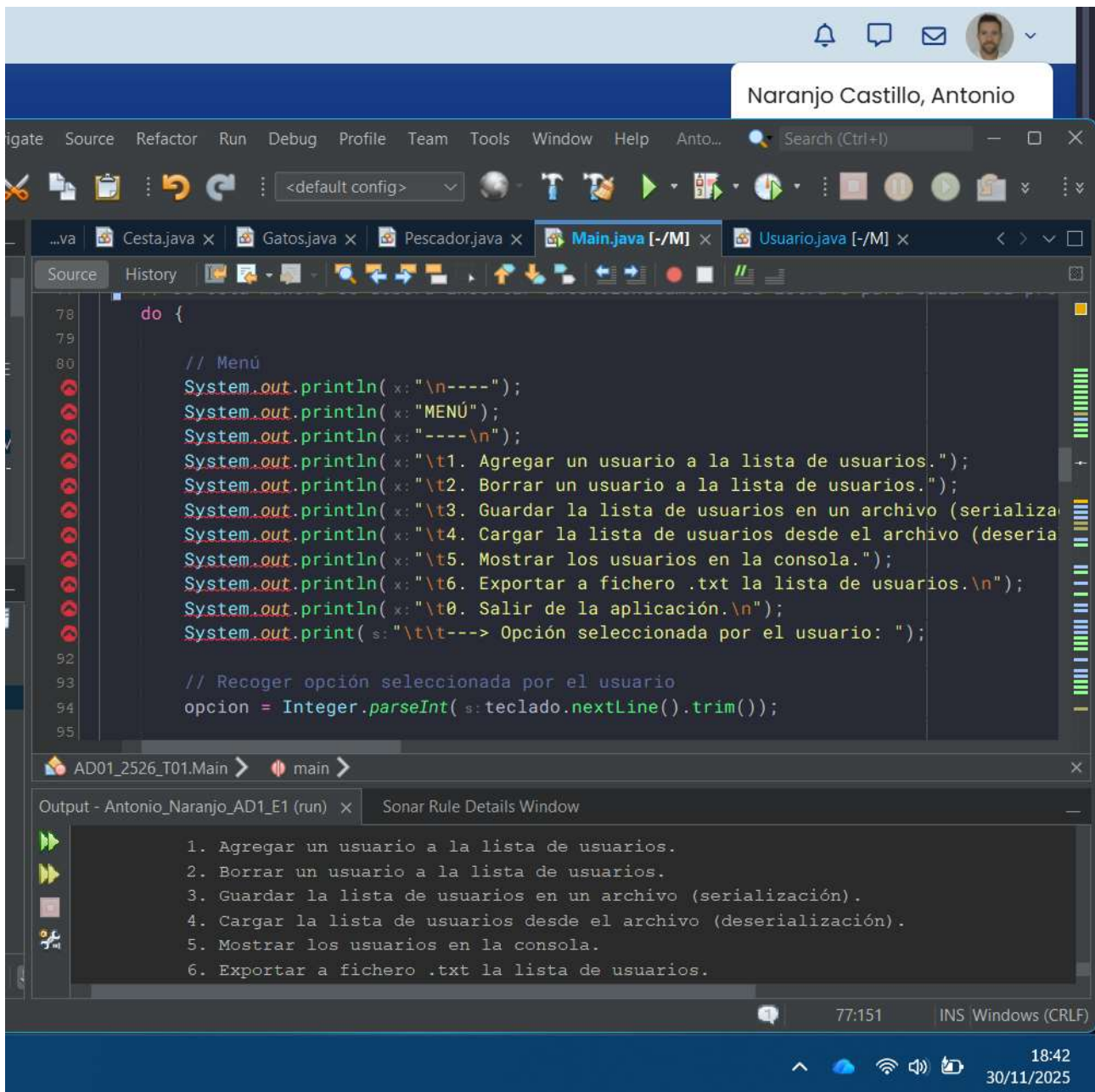


Se implementan los métodos getters y setters para obtener o establecer los atributos anteriores para cada objeto usuario creado mediante la clase Usuario.



Importante, para poder serializar una lista de objetos de la clase `Usuario`, de modo que se pueda guardar en un archivo, y que de igual modo se pueda deserializar esa lista desde el archivo, la clase `Usuario` debe implementarse desde la interfaz `Serializable`.

2. Implementación del menú



The screenshot shows an IDE window with the following components:

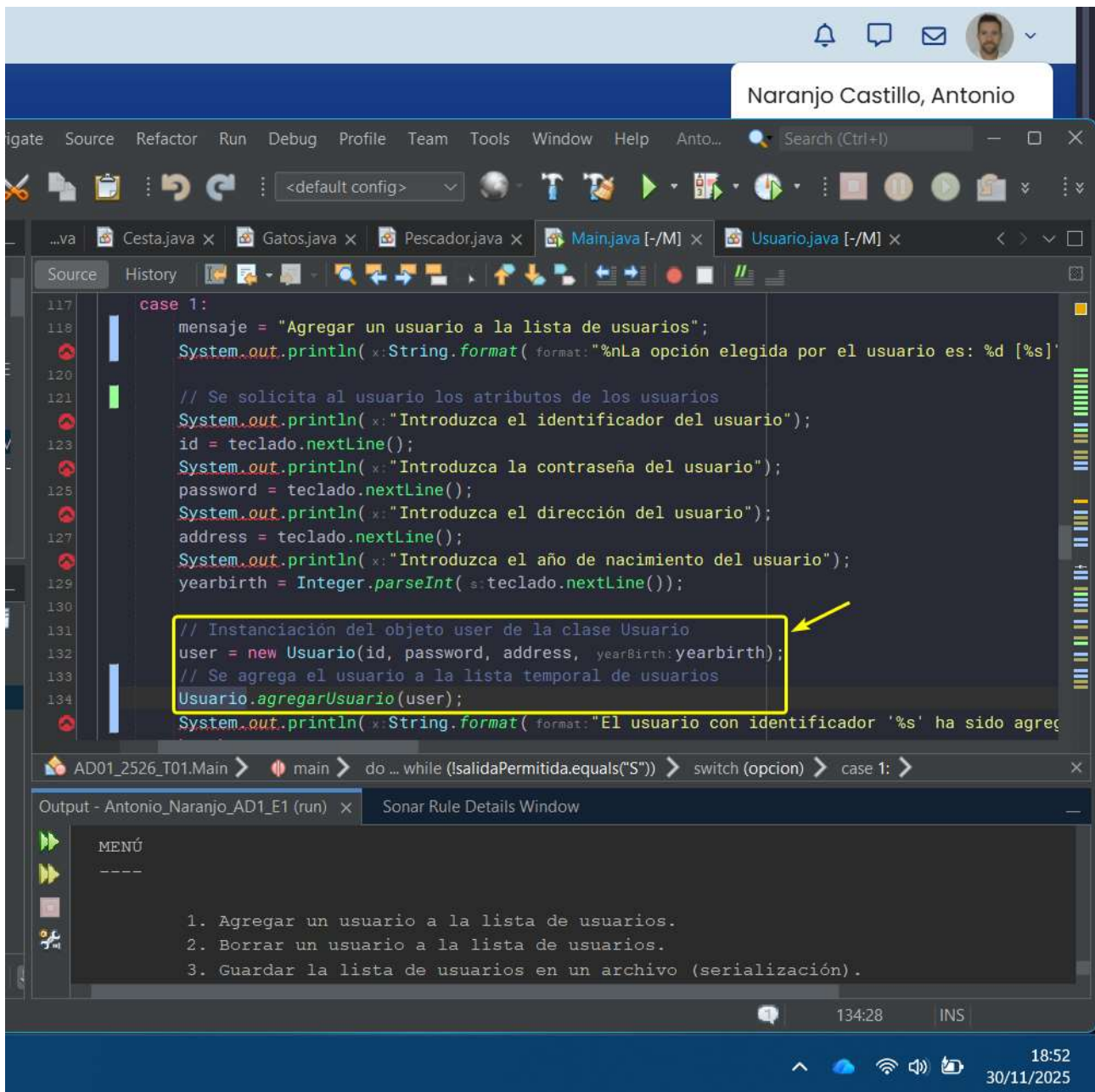
- Top Bar:** User profile "Naranjo Castillo, Antonio" and navigation icons.
- Menu Bar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar.
- Toolbar:** Icons for file operations, running, and debugging.
- Editor:** Displays the code for `Main.java`. The code is as follows:

```
78 do {
79
80     // Menú
81     System.out.println(x: "\n----");
82     System.out.println(x: "MENÚ");
83     System.out.println(x: "----\n");
84     System.out.println(x: "\t1. Agregar un usuario a la lista de usuarios.");
85     System.out.println(x: "\t2. Borrar un usuario a la lista de usuarios.");
86     System.out.println(x: "\t3. Guardar la lista de usuarios en un archivo (serializa");
87     System.out.println(x: "\t4. Cargar la lista de usuarios desde el archivo (deseria");
88     System.out.println(x: "\t5. Mostrar los usuarios en la consola.");
89     System.out.println(x: "\t6. Exportar a fichero .txt la lista de usuarios.\n");
90     System.out.println(x: "\t0. Salir de la aplicación.\n");
91     System.out.print(s: "\t\t--> Opción seleccionada por el usuario: ");
92
93     // Recoger opción seleccionada por el usuario
94     opcion = Integer.parseInt(s: teclado.nextLine().trim());
95 }
```
- Output Window:** Shows the output of the program:

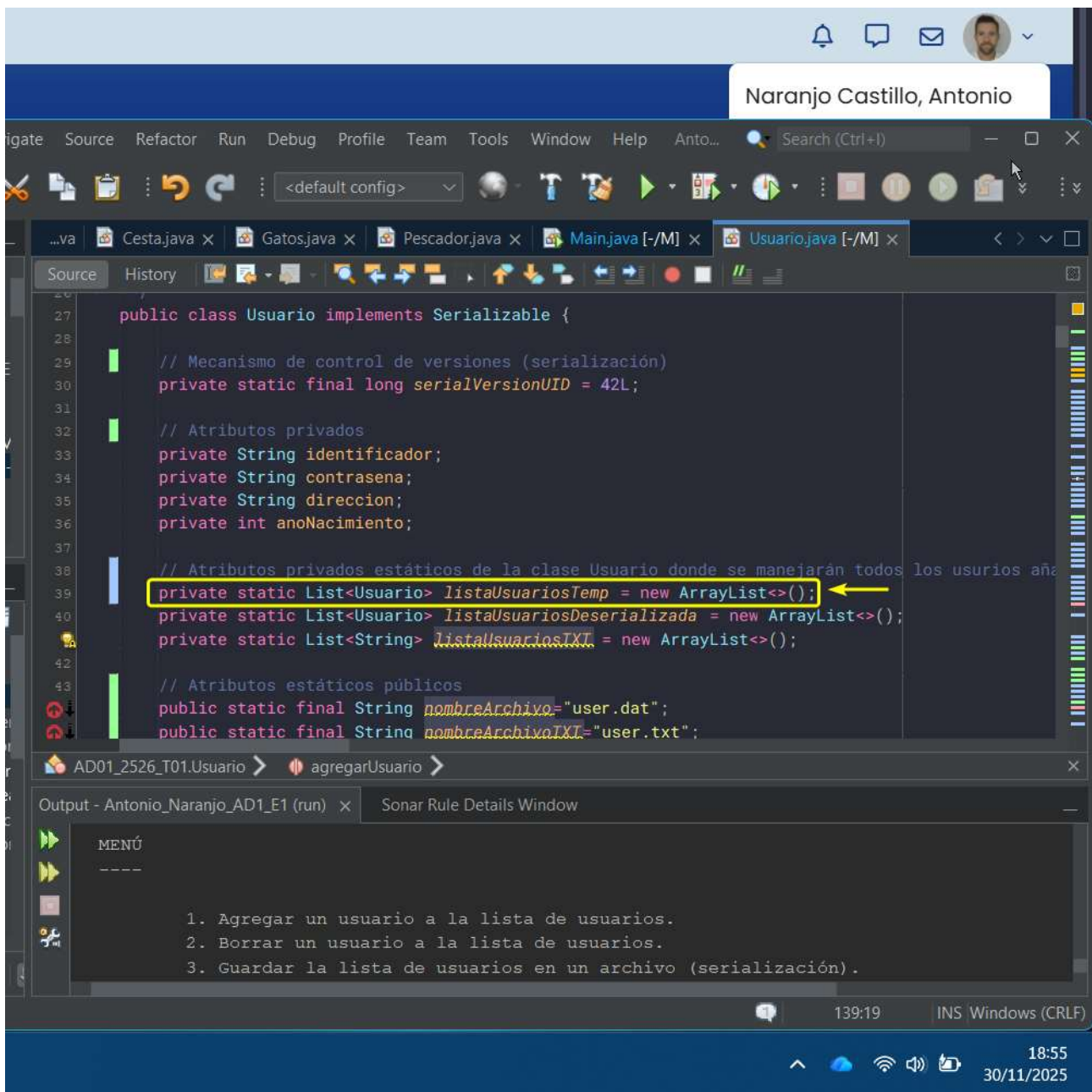
```
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.
```
- Bottom Bar:** Shows the file path `AD01_2526_T01.Main > main >`, the output window title `Output - Antonio_Naranjo_AD1_E1 (run)`, and system information: `77:151 INS Windows (CRLF)` and the date/time `18:42 30/11/2025`.

Se implementa un bucle do-while para reproducir el menú tantas veces como el String salidaPermitida distinta de "S" (Salida permitida = SI) se presenten por parte del usuario. Se deberá insertar intencionadamente la letra S para salir del programa, evitando errores de salida del programa al pulsar accidentalmente cualquier otra tecla del teclado.

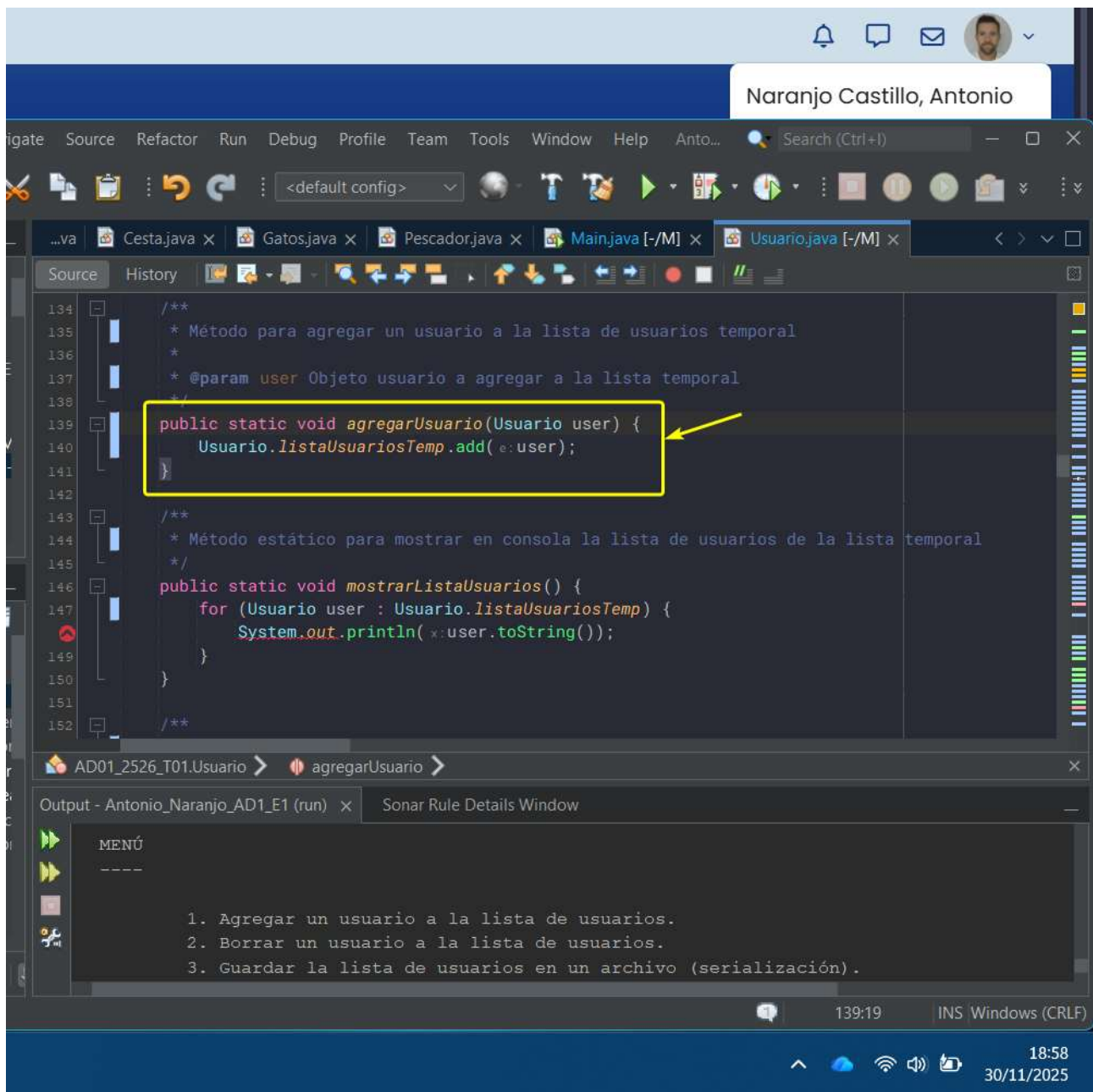
3. Agregar un usuario a la lista de usuarios



Una vez seleccionada la opción 1 por el usuario, se procede a solicitar los cuatro atributos que definen el objeto usuario de la clase Usuario, para termina con la instanciación de dicho objeto usuario, así como la aplicación del método estático `agregarUsuario()` de la clase Usuario aportando como argumento el objeto usuario recientemente instanciado.

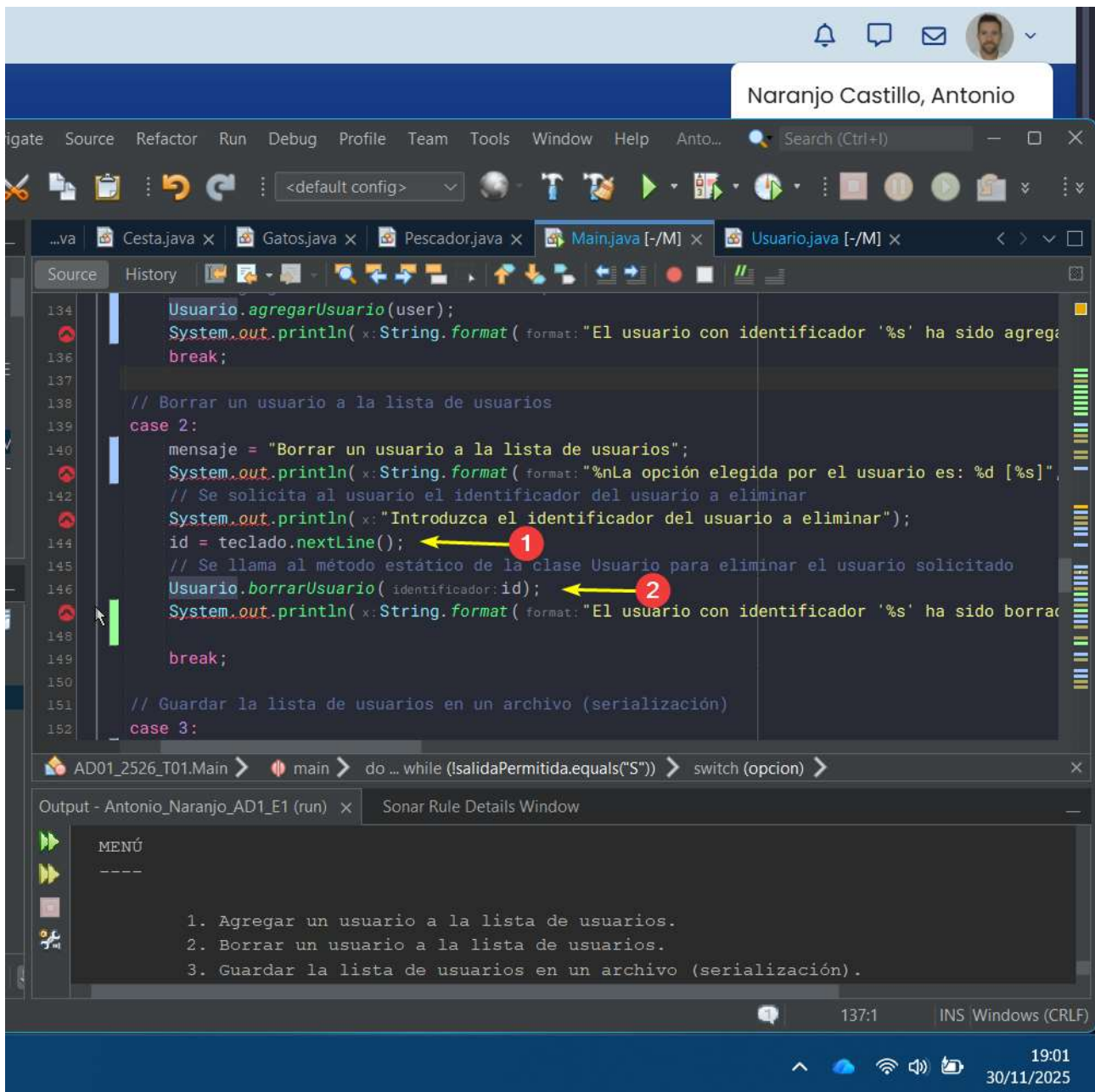


En la clase `Usuario` se crea una lista temporal donde se almacenarán los distintos objetos usuarios que se vayan instanciando, se trata de un atributo estático privado de la clase `Usuario`.

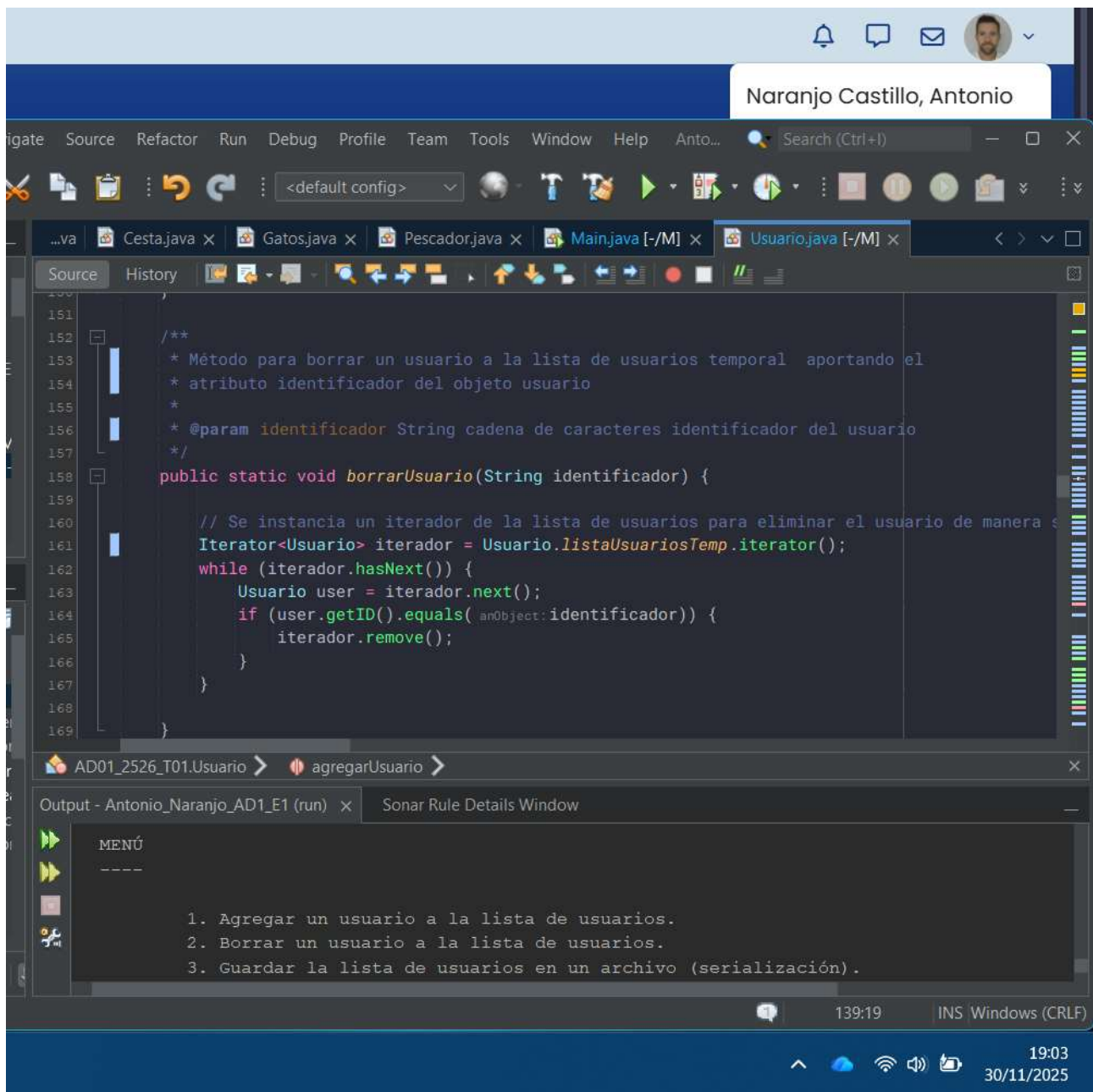


En esta imagen se presenta el método estático `agregarUsuario()` para incorporar a la lista temporal indicada con anterioridad a cada uno de los usuarios que se vayan dando de alta.

4. Borrar un usuario de la lista de usuarios

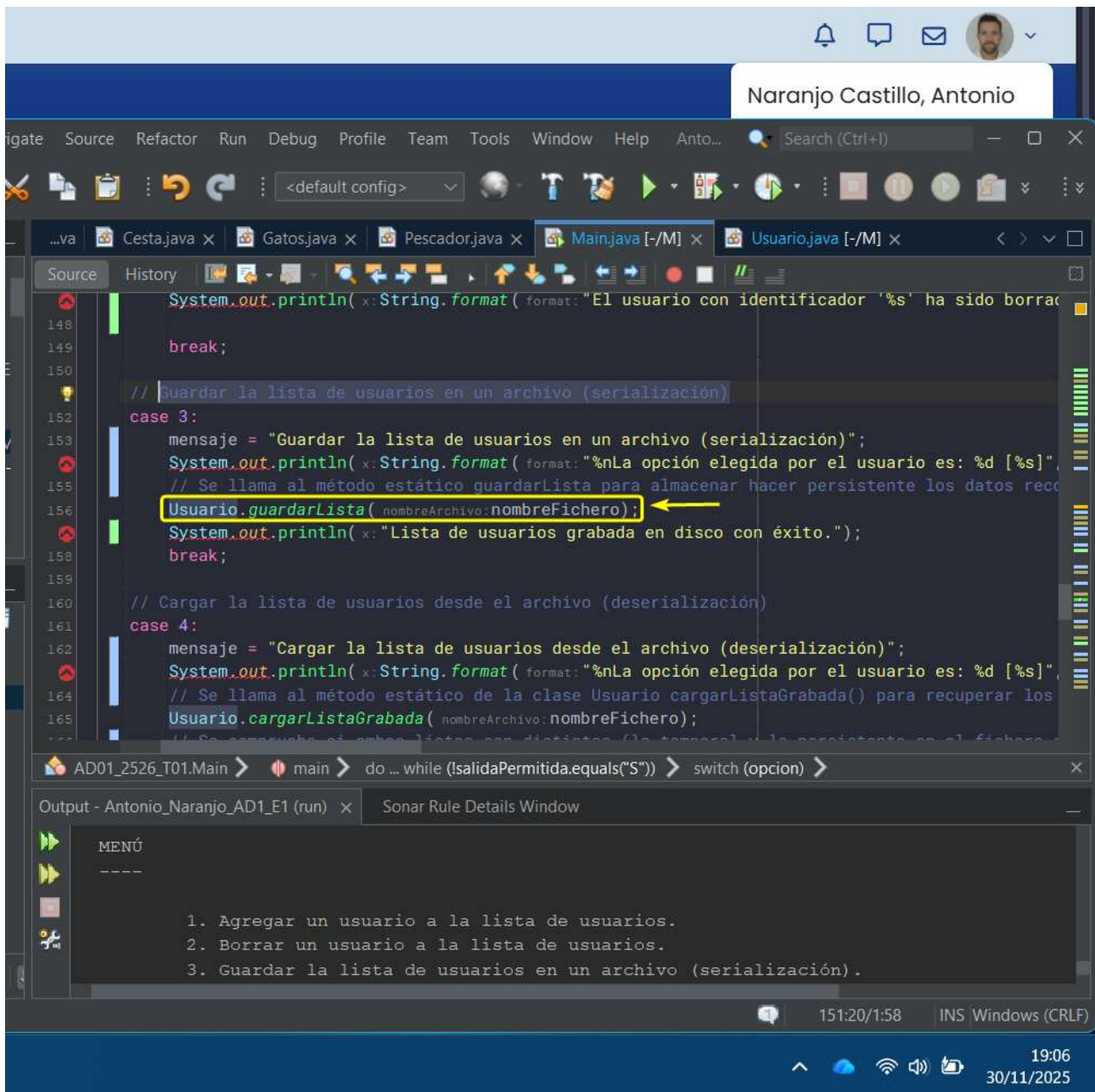


Para borrar un usuario de la lista temporal de usuarios, el usuario del programa deberá seleccionar la opción 2, acto seguido, el programa solicitará la identificación del usuario a eliminar para ejecutar el método estático de la clase Usuario `borrarUsuario()` pasando como argumento el atributo `identificador` del usuario a borrar.



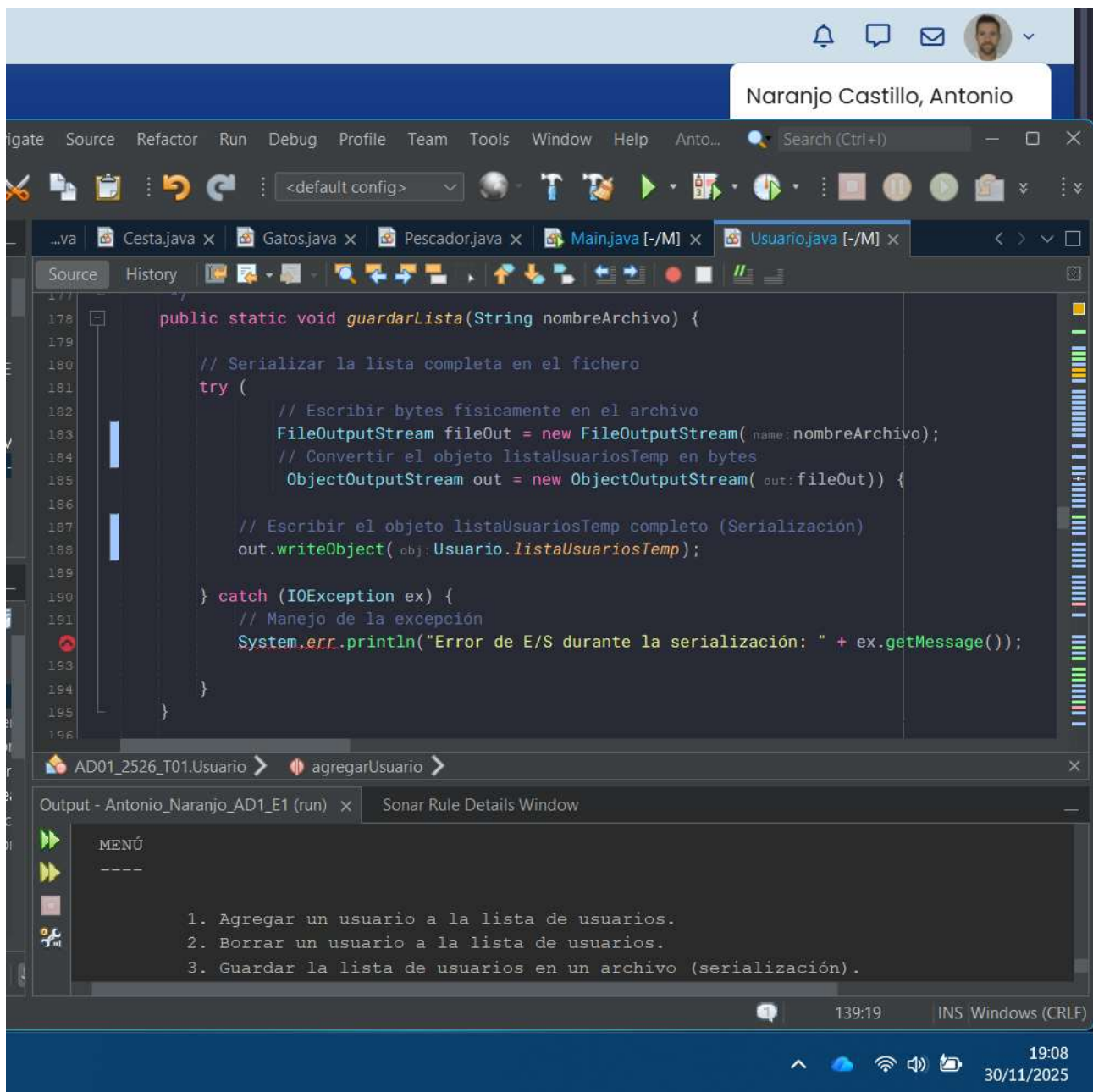
Se presenta el método `borrarUsuario` estático de la clase `Usuario`, haciendo uso de un iterador para garantizar borrar el usuario de manera segura de la lista temporal.

5. Guardar la lista de usuarios en un archivo (serialización)



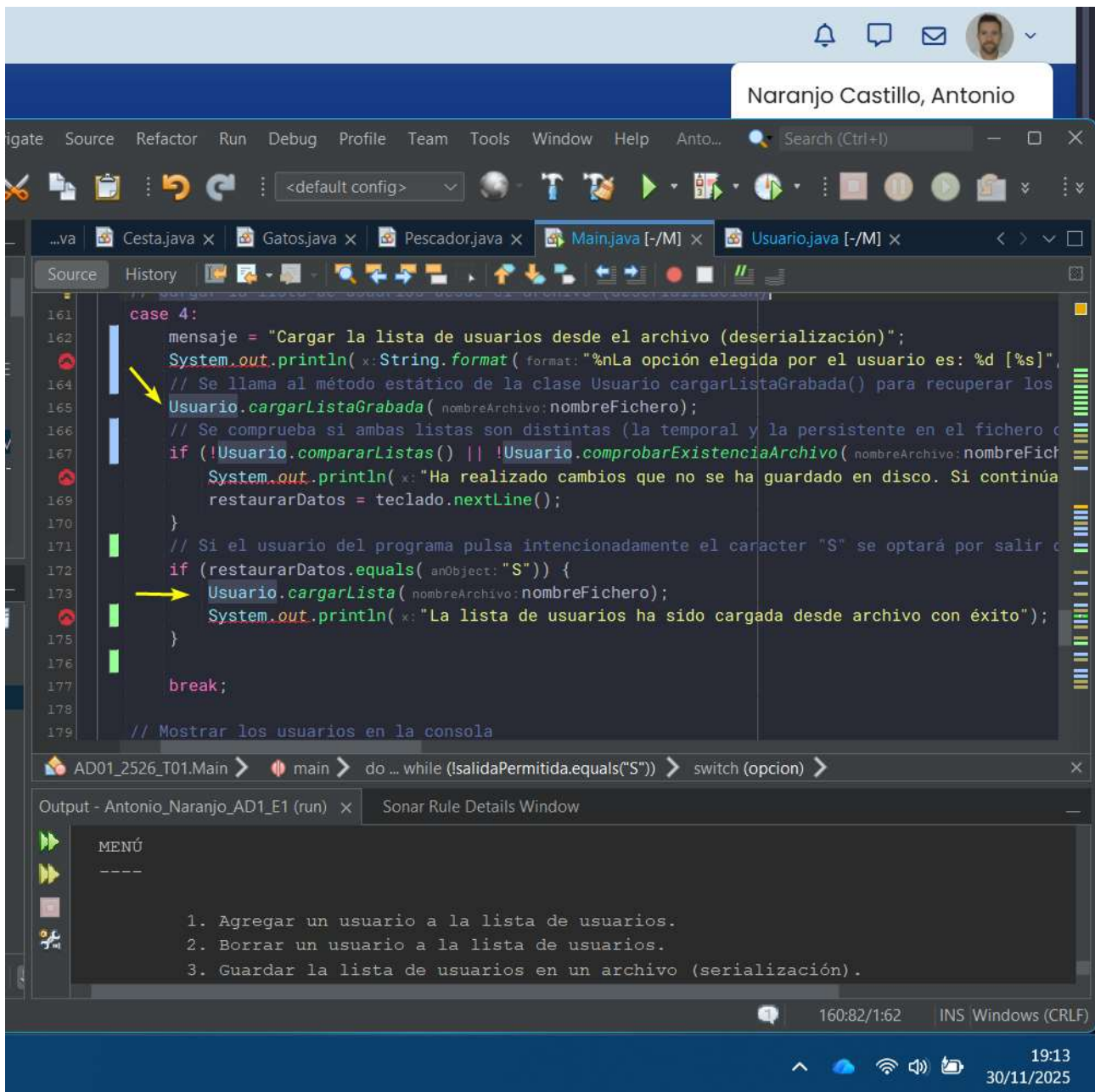
Para guardar la lista temporal de usuarios en un fichero ubicado en el disco duro del PC, el usuario deberá establecer la opción 3, de esta manera se conseguirá la persistencia de los datos quedando grabados en un archivo del disco duro.

Se ejecuta el método estático de la clase Usuario `guardarLista()` aportando como argumento el nombre del archivo en cuestión.

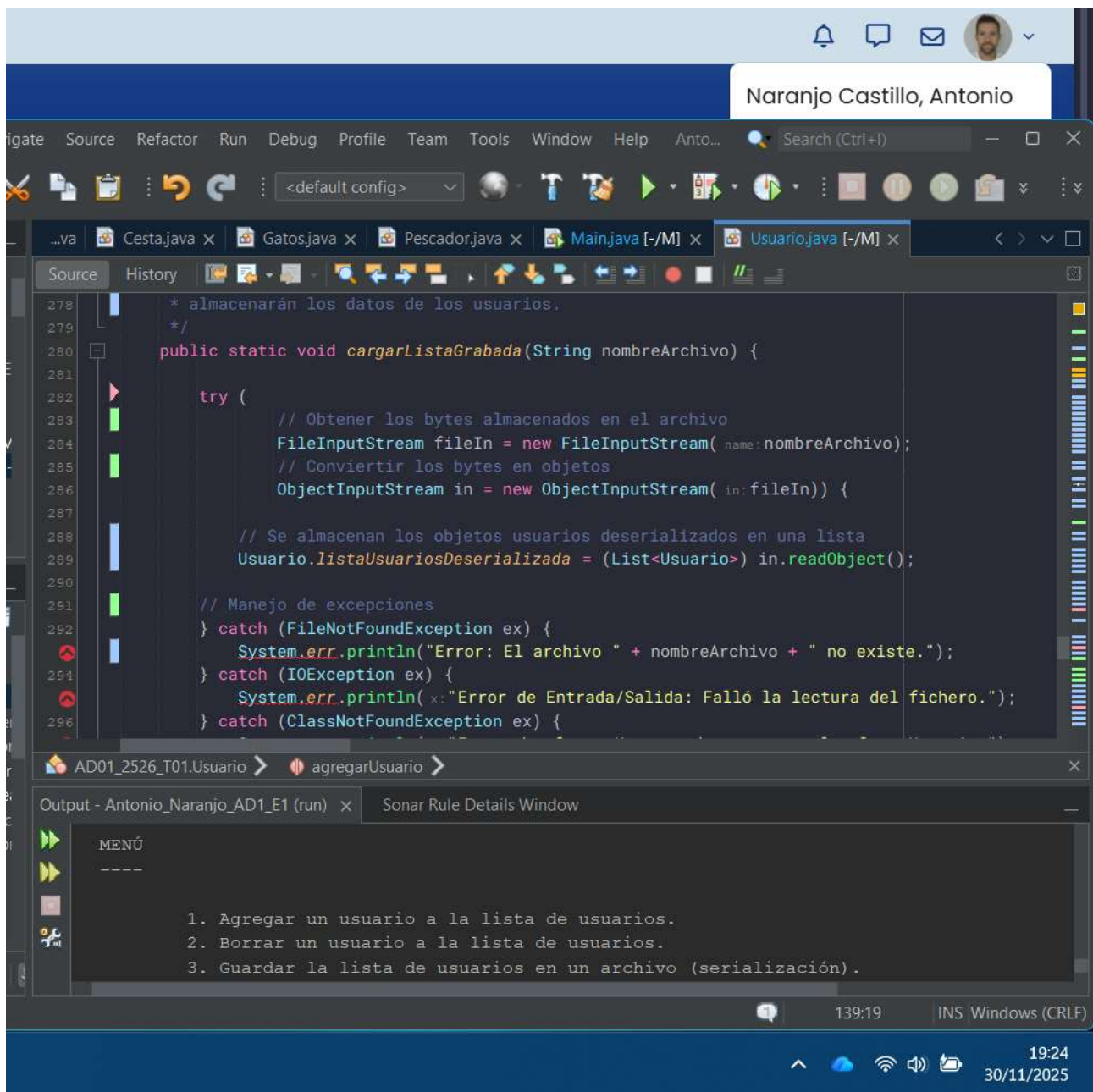


Se muestra el método estático de la clase `Usuario` `guardarLista`, para el cual, se emplean objetos de las clases `FileOutPutStream` y `ObjectOutPutStream`, para transformar la lista de objetos usuarios en bytes (out), y luego, estos bytes guardarlos en el archivo en cuestión (fileOut).

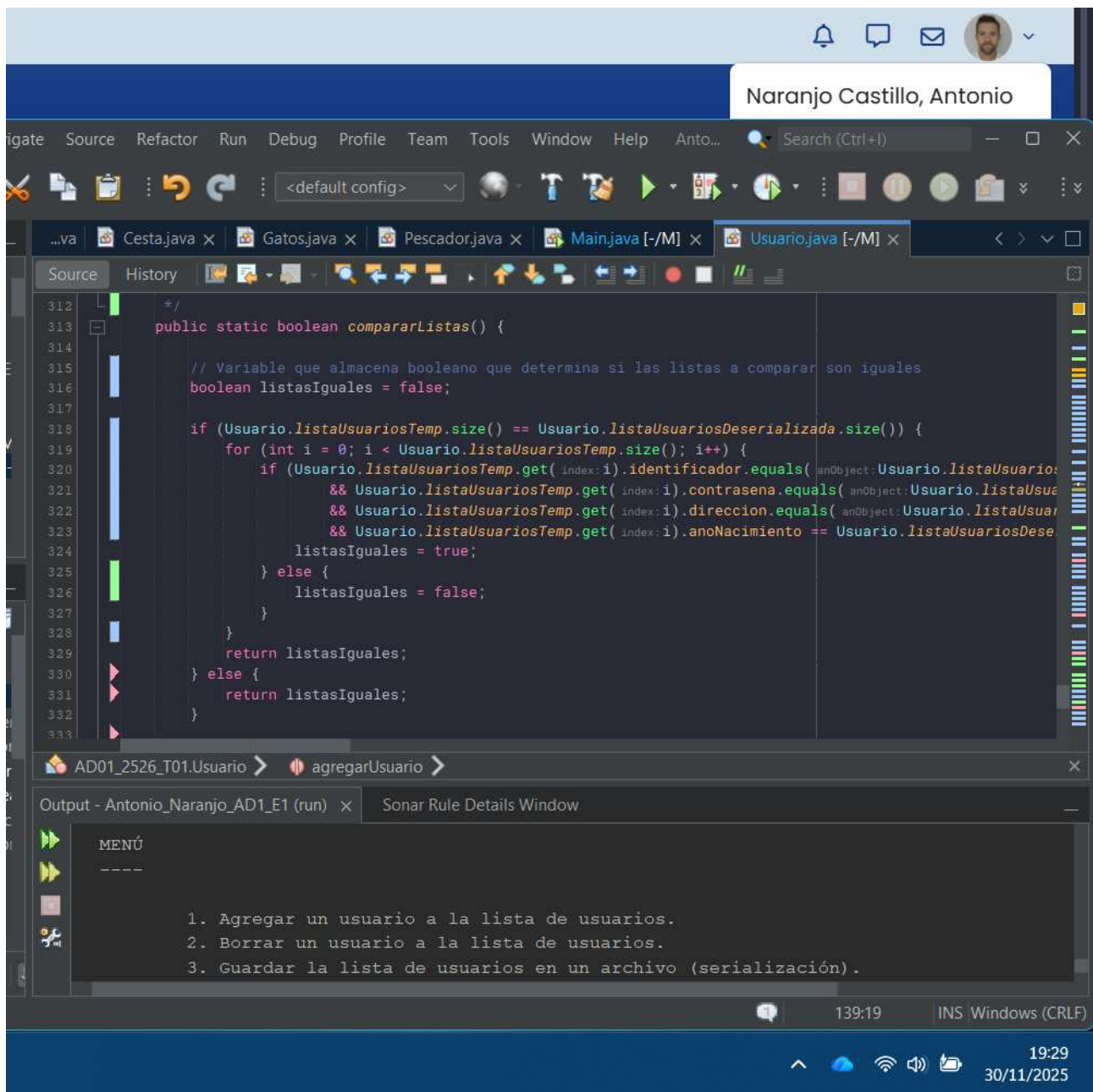
6. Cargar la lista de usuarios desde el archivo (deserialización)



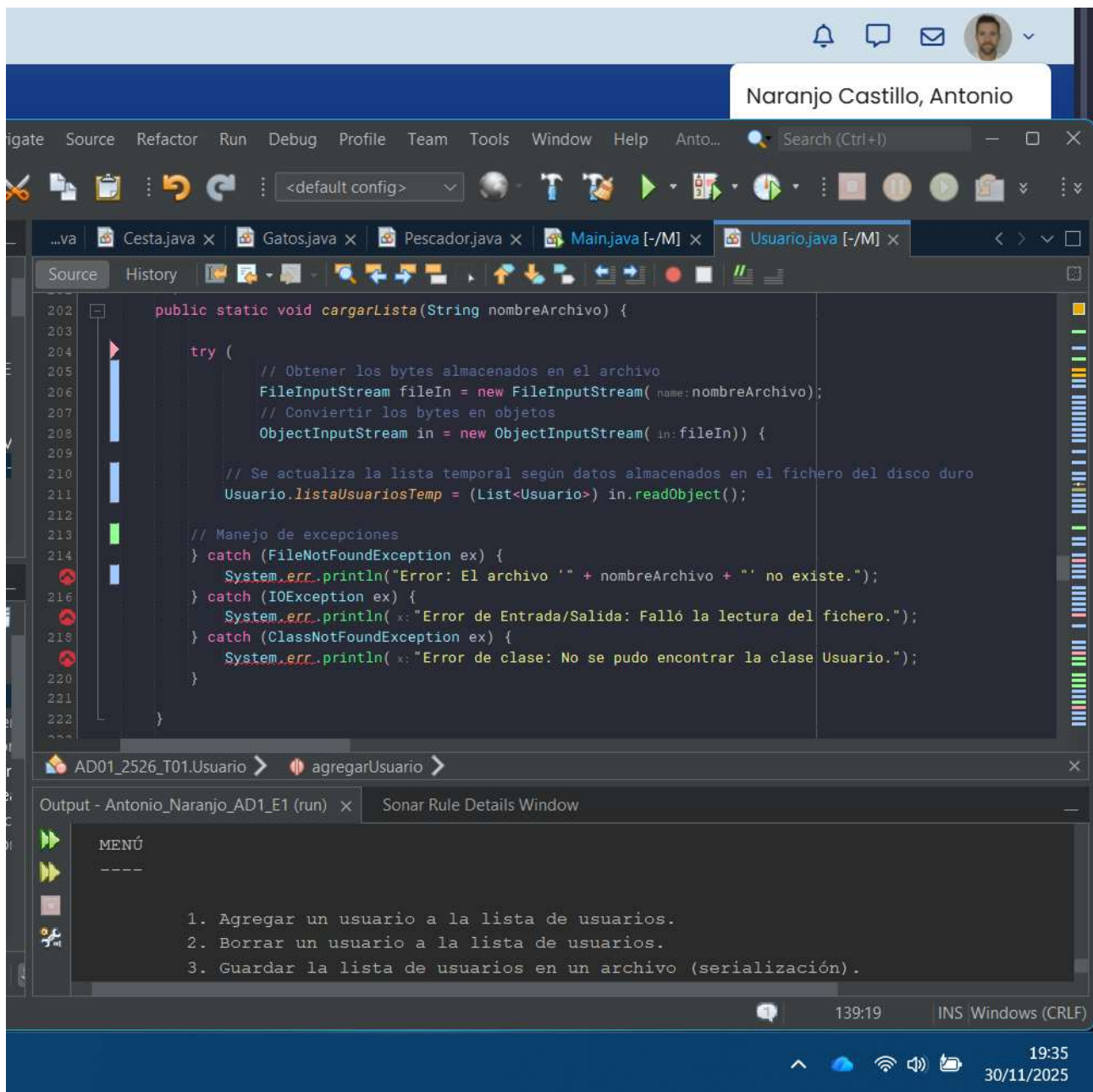
El usuario del programa seleccionará la opción 4 para cargar los datos persistentes del fichero ubicado en el disco duro del PC. Posteriormente, se ejecutará el método estático de la clase `Usuario` `cargarListaGrabada` aportando el nombre del fichero a implementar. La lista de usuarios almacenada en tal fichero se guardará en la lista de usuarios deserializada, y posteriormente, se comparará con la lista temporal por medio del método estático `cargarLista()` para determinar si ambas listas son iguales, además de, comprobar la existencia del fichero en cuestión, para que en caso de que no lo sean o no exista el fichero, el usuario del programa podrá determinar si desea actualizar los datos o realizar cualquier otra opción del menú.



Se presenta el método `cargarListaGrabada` empleándose objetos `FileInputStream` y `ObjectInputStream`, el primero para obtener los bytes del archivo guardado en el disco, y el segundo para transformar los bytes en una lista de objetos usuarios que posteriormente se almacenarán en la lista de usuarios deserializada.

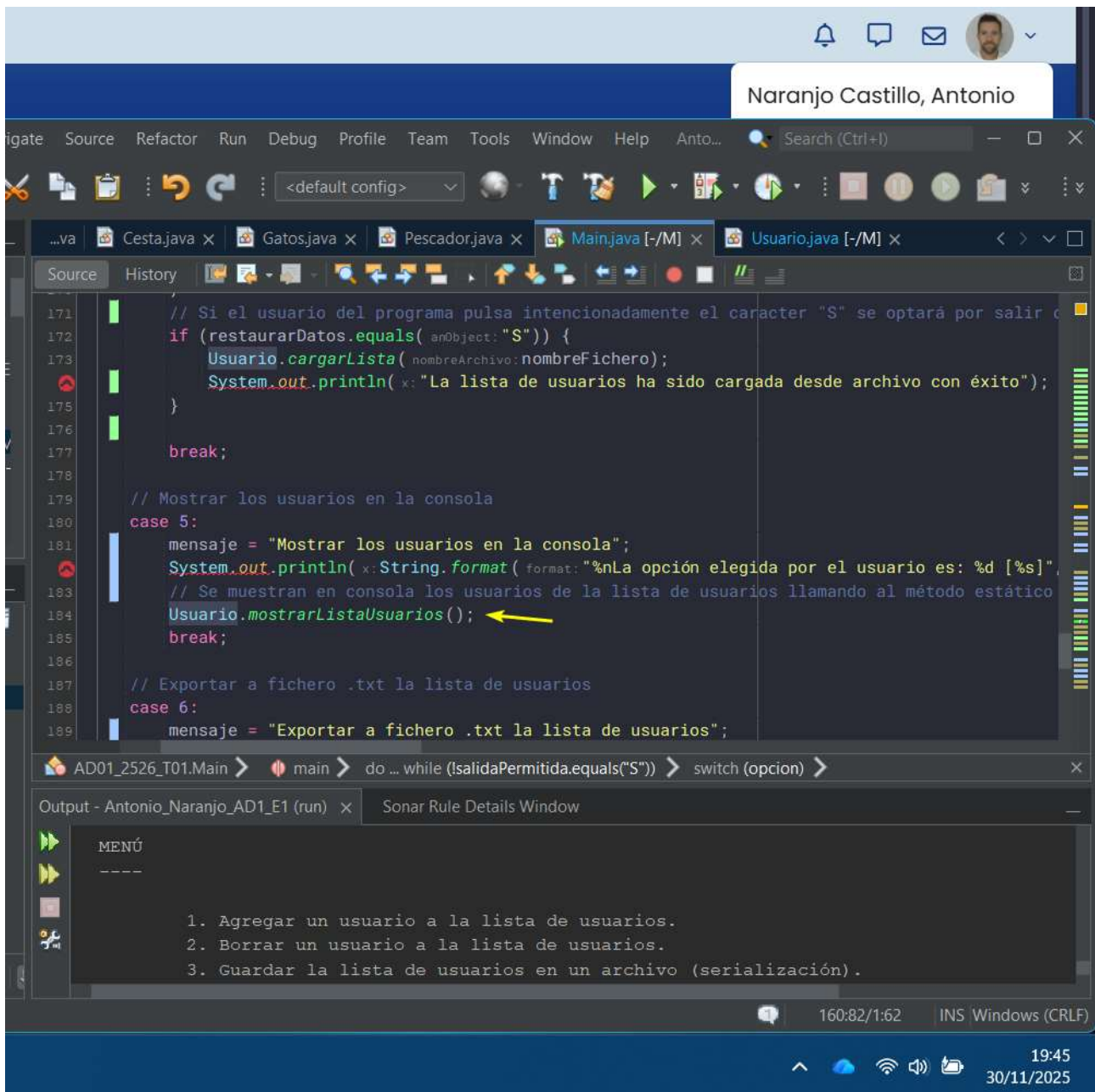


Posteriormente, con el método igualmente estático `compararListas()`, se compararán las listas deserializada y temporal para comprobar si son iguales, para ello, primero se comprueba si ambas listas tienen el mismo número de elementos, en segundo lugar, para cada elemento se comprueba que sus atributos sean iguales, a la vez que se comprueban que los usuarios siguen el mismo orden en ambas listas.

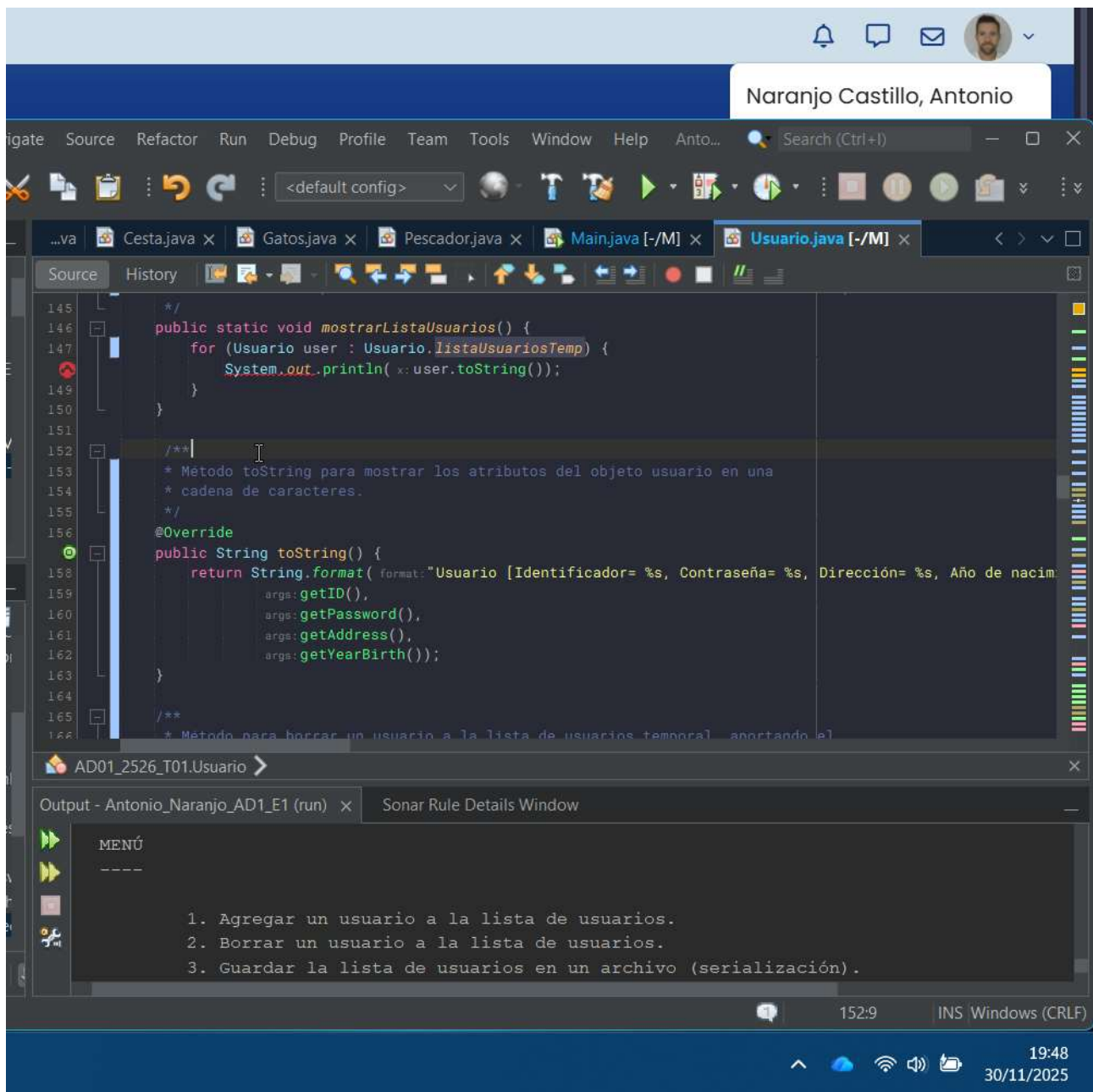


Se muestra el método `cargarLista()`, mediante el cual se actualizan los objetos usuarios de la lista temporal a partir de los datos obtenidos del fichero grabado en el disco duro del PC. De manera similar al método `cargarListaGrabada`, se empleará un objeto `FileInputStream` envuelto en un objeto `ObjectInputStream` al cual se le pasa como argumento para obtener los bytes del archivo del disco duro y transformarlos en una lista de objetos usuarios.

7. Mostrar los usuarios en consola



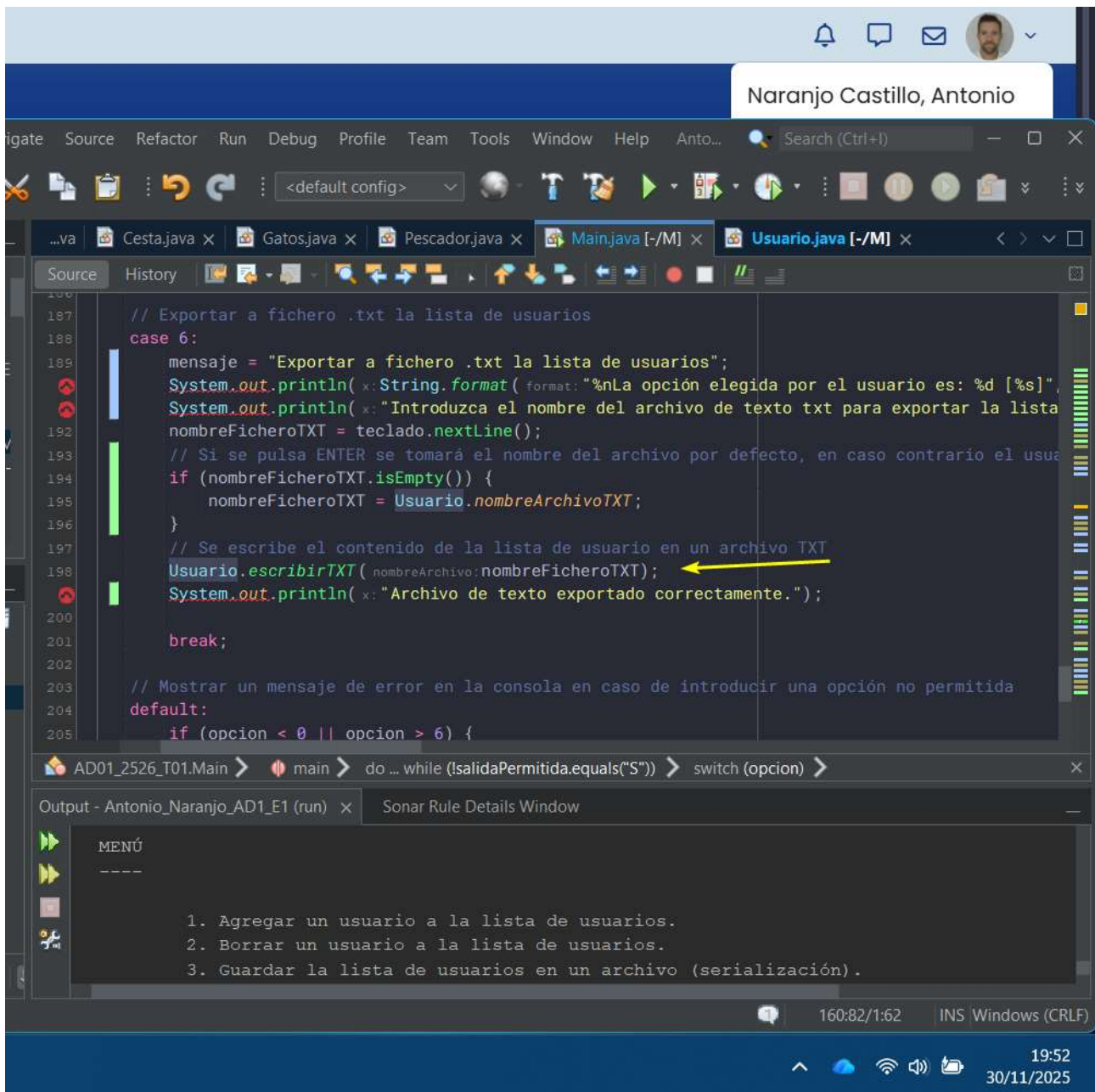
Cuando el usuario del programa seleccione la opción 5 se mostrarán los resultados en la consola, mostrando todos los usuarios almacenados en la lista temporal de usuarios.



Se presentan los métodos `mostrarListaUsuarios()`, estático, y `toString()`, para imprimir a modo de cadena de texto todos y cada uno de los objetos usuarios contenidos en la lista temporal de usuarios.

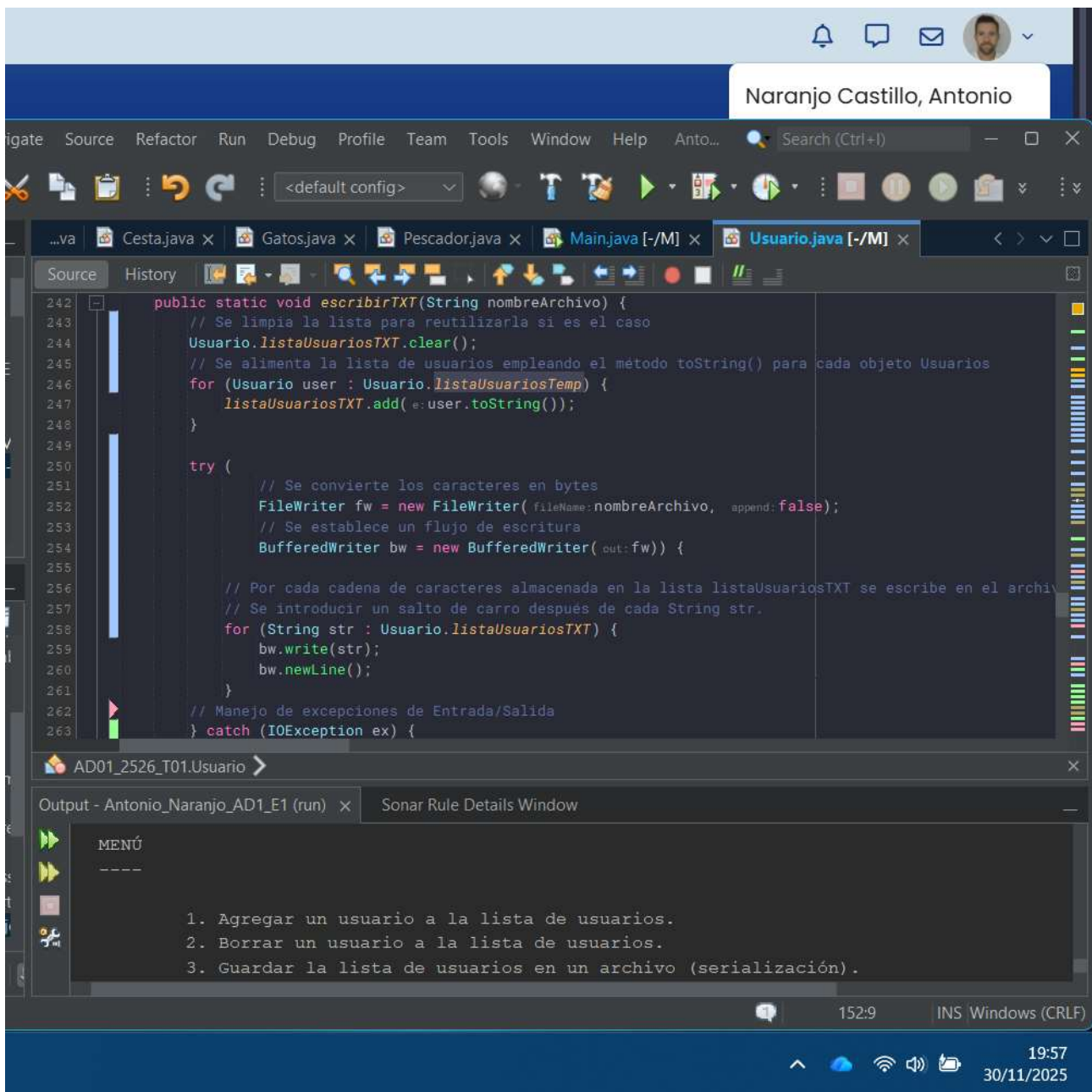
El método `toString()` devuelve un `String` cadena de caracteres mostrando todos los atributos que definen a cada usuario, según su método constructor presentado al inicio de este documento.

8. Exporta a fichero .txt la lista de usuarios



Se selecciona la opción 6 para escribir en un archivo de texto cada uno de los objetos almacenados en la lista temporal. Para ello se emplea el método estático escribirTXT().

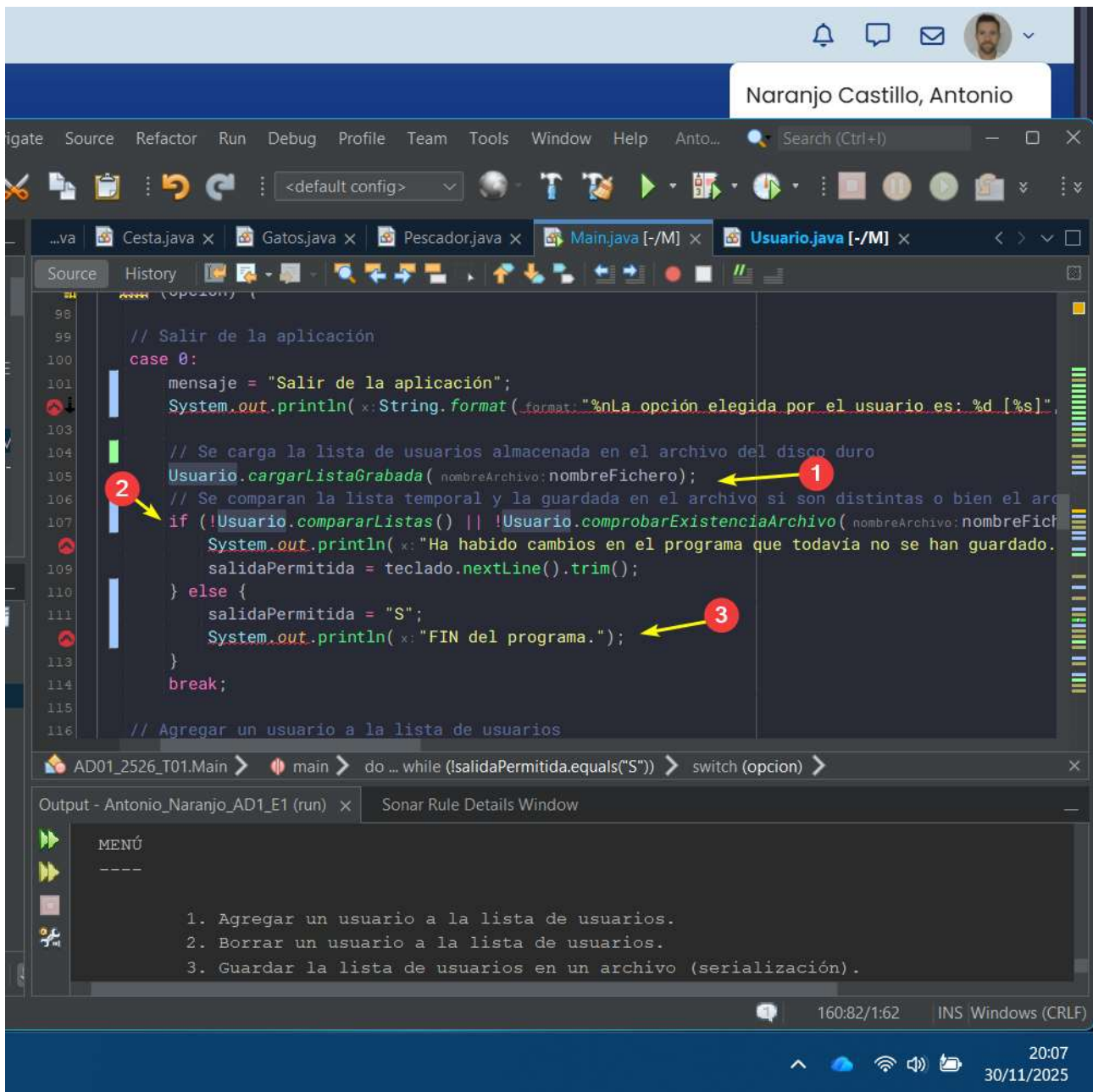
Se define como atributo estático público de la clase Usuario el nombre del archivo "user.txt" de esta manera al pulsar ENTER por defecto se usará tal archivo (de la misma manera se procede con el archivo user.dat al iniciar el programa).



El método `escribirTXT` estático, de la clase `Usuario`, emplea dos objetos para transformar los caracteres en bytes, un objeto `BufferedWriter` que envuelve a un `FileWriter`, el primero capta un flujo de caracteres y el segundo los transforma en bytes para finalmente hacerlos persistente en el fichero *.txt en cuestión.

Previamente, se define una lista de cadenas de caracteres `listaUsuariosTXT`, limpia inicialmente, Strings obtenidos de aplicar el método `toString()` sobre cada objeto usuario almacenada en la lista temporal. Se emplea un bucle for-each para ello, y posteriormente se vuelve a utilizar para escribir cada String en el archivo .txt sumándole un salto de carro y de línea mediante el método `newLine()`.

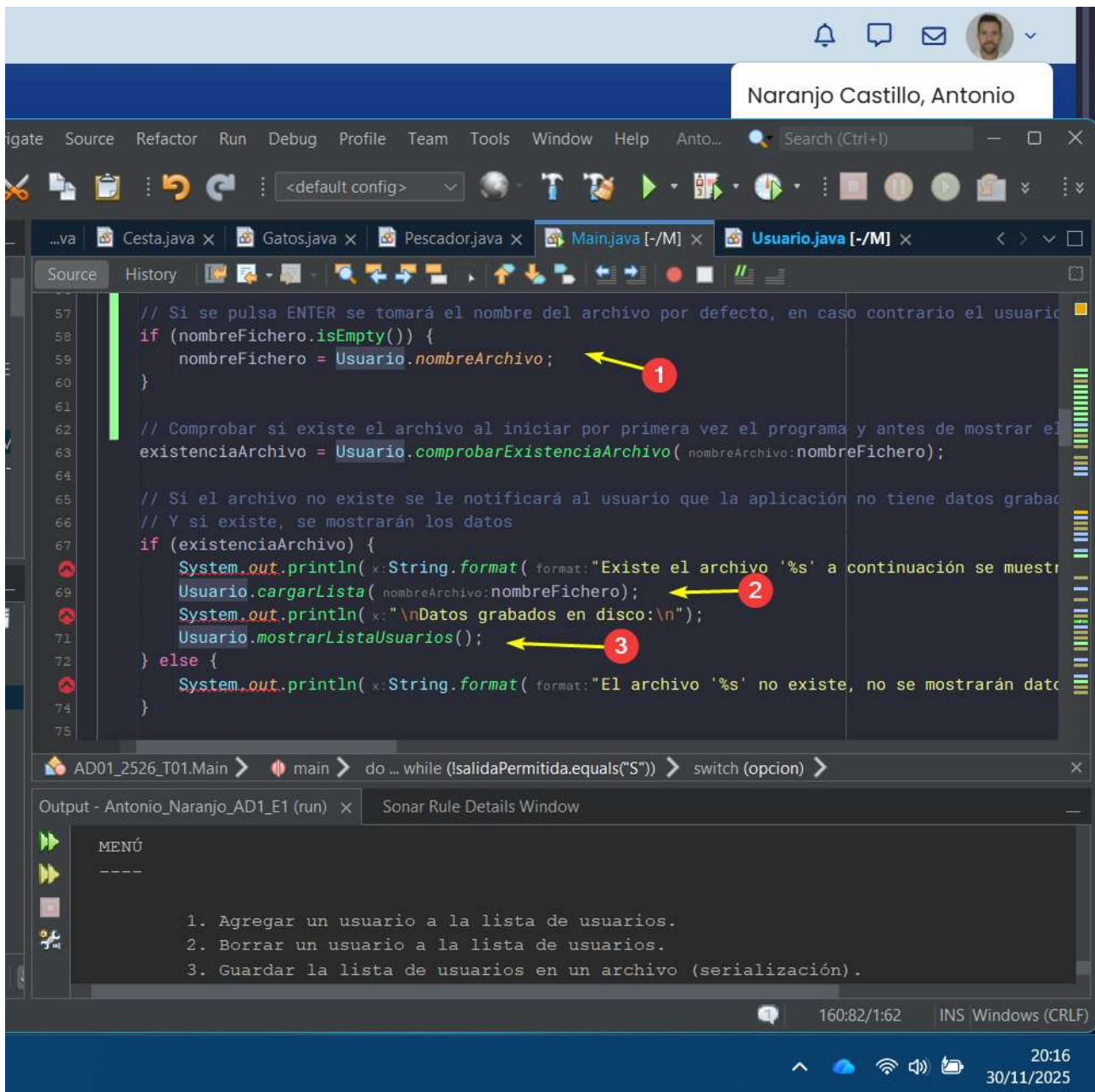
9. Salir de la aplicación



Se selecciona la opción 0 para salir del programa, pero antes, se llamará al método `cargarListaGrabada()` pasando como argumento el nombre del fichero en disco, para posteriormente comprobar si la lista obtenida deserializada coincide con la lista temporal y también comprobar si ese archivo existe en el disco duro, ofreciendo la oportunidad de volver atrás, no salir del programa y poder llevar a cabo otra opción ofrecida por el menú del aplicativo.

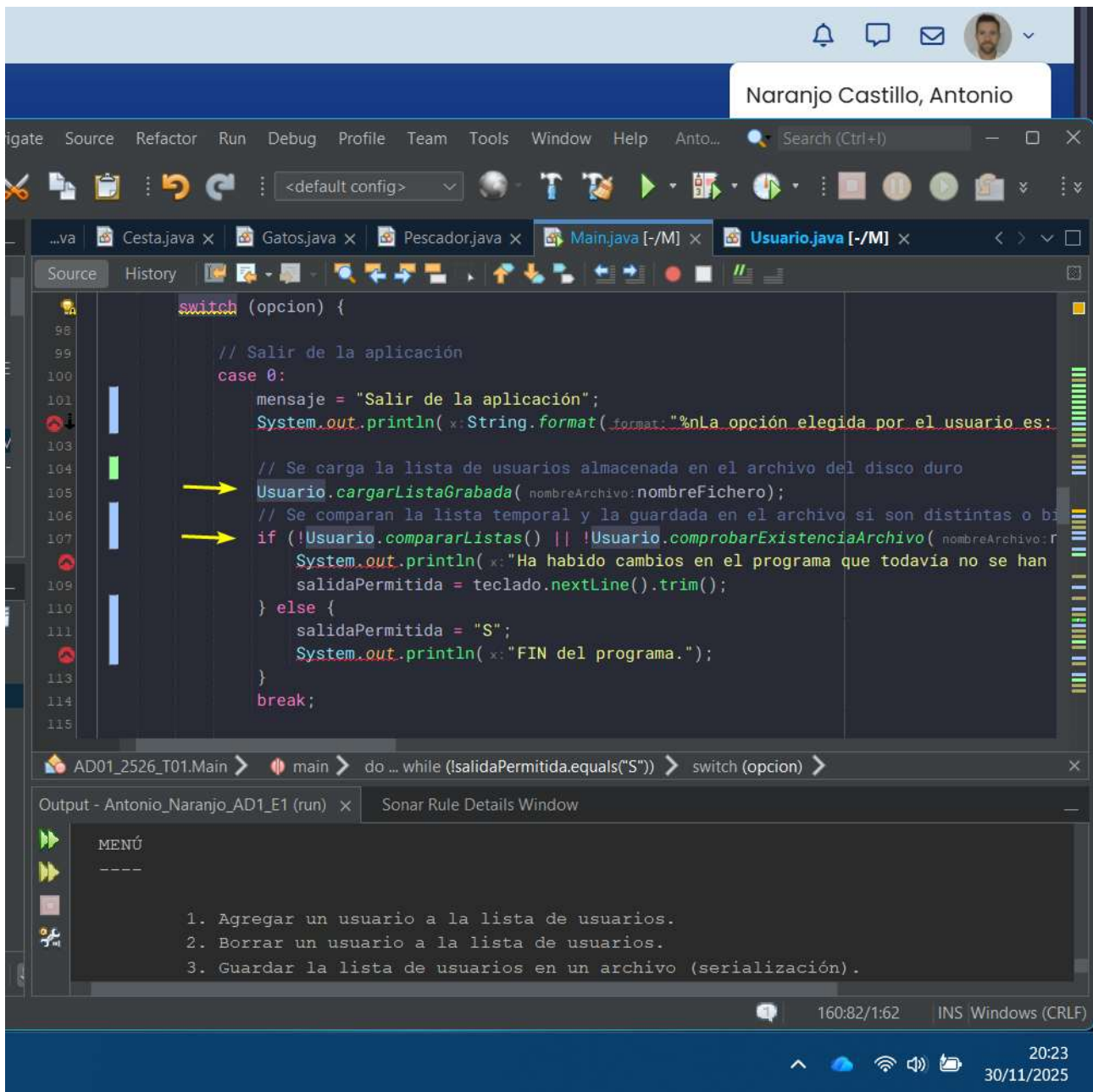
Finalmente, si el archivo existe y ambas listas de objetos usuarios, tanto la temporal como la deserializada, son iguales, el programa sí finalizará son preguntar al usuario dado que los cambios están guardados en el fichero del disco duro.

10. ¿Existe el archivo 'user.dat'?



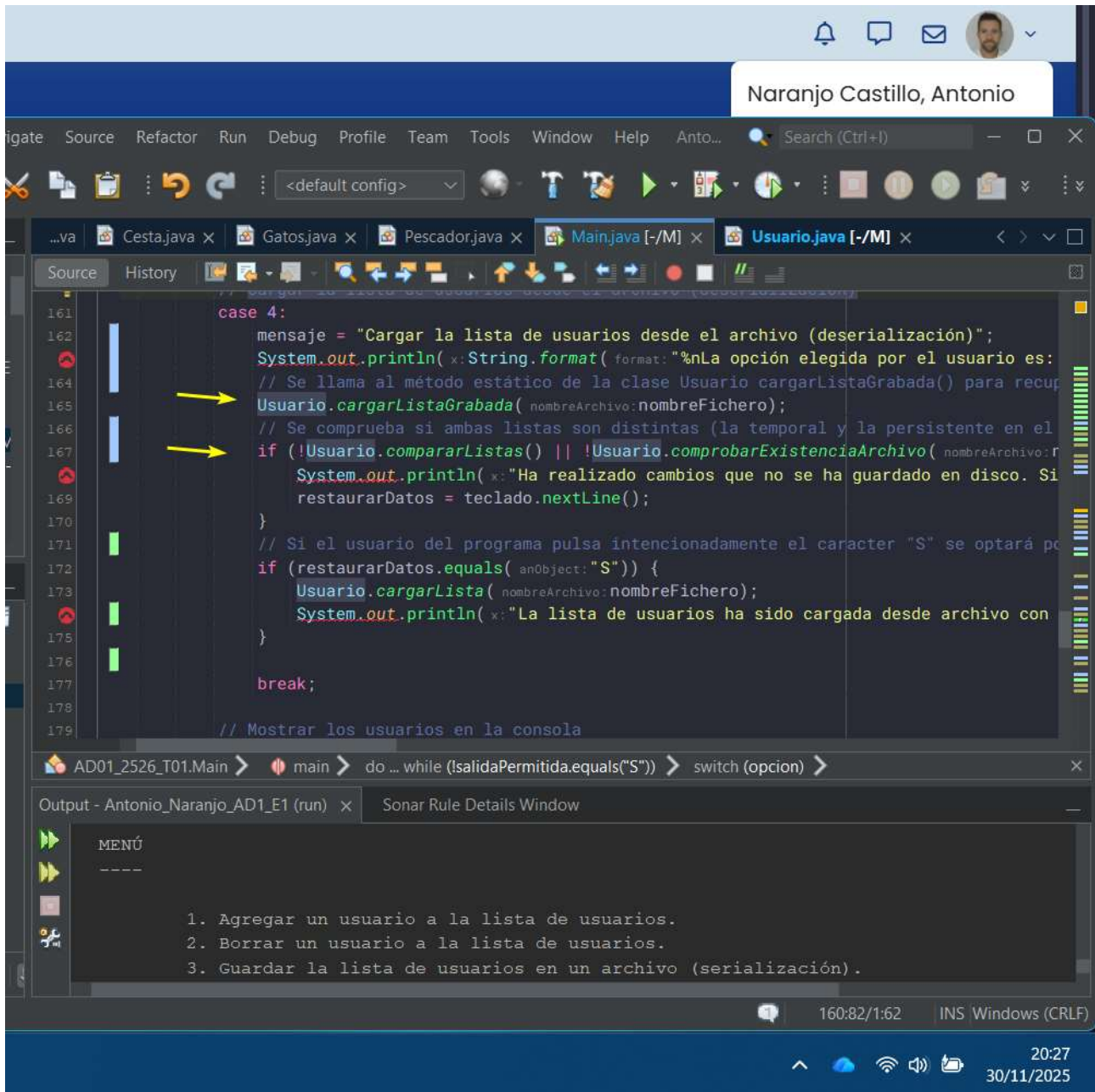
Al ejecutarse la aplicación, se comprueba si existe el fichero user.dat. Para el caso, se ha tenido en cuenta que si el usuario presiona ENTER se usará el nombre user.dat por defecto. Para ello, se emplea el método estático `comprobarExistenciaArchivo()`. Si no existe, se avisará al usuario de que la aplicación no tiene datos grabados. Si existe, se cargarán los datos guardados de modo que los usuarios que hubiera en el fichero se cargarán en una lista empleando para ello el método estático `cargarLista()`, y luego, con el método `mostrarListaUsuarios()` mostrando en consola la lista de los objetos usuarios `toString()` almacenados en el fichero.

11. Advertencia sobre datos no guardados antes de salir



En caso de ejecutar la aplicación y modificar algún dato, si el usuario del programa intenta salir de la aplicación sin haber guardado los cambios, se le avisa indicando que los cambios no han sido guardados. Tal y como se explicó en el punto anterior sobre la selección de la opción 0, se comprueba si la lista temporal es igual a la lista deserializada, y si no es así o el archivo no existe, se le avisará al usuario del programa antes de salir.

12. ¿Recuperar datos de disco?



Si se ejecuta la opción de recuperar datos, habiendo realizado cambios en el programa, se advertirá al usuario que los cambios se perderán, puesto que se cargarán los datos del fichero en la lista, y se le pedirá confirmación antes de continuar. Este punto se tiene en cuenta al aplicar la opción 4 del menú, y en cierto modo ya se ha detallado cómo se procesa este aspecto, básicamente, se compara la lista deserializada del archivo con la lista temporal de tal manera que si son diferentes se le da la opción al usuario de retroceder antes de sustituir el contenido de la lista temporal.

El usuario deberá seleccionar el carácter “S” intencionadamente para que se proceda con la carga de la lista deserializada, evitando en cierto modo, que accidentalmente el usuario pudiera pulsar otra tecla del teclado.

13. Anexo. Resultados del programa en consola.

a. Inicio del programa

```
case 4:
    mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
    System.out.println(x:String.format( format: "%nLa opción elegida por el usuario es:
    // Se llama al método estático de la clase Usuario cargarListaGrabada() para recup
    Usuario.cargarListaGrabada( nombreArchivo:nombreFichero);
    // Se comprueba si ambas listas son distintas (la temporal y la persistente en el
    if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo( nombreArchivo:
    System.out.println(x:"Ha realizado cambios que no se ha guardado en disco. Si
    restaurarDatos = teclado.nextLine();
```

Output - Antonio_Naranjo_AD1_E1 (run) x Sonar Rule Details Window

```
run:
Introduzca el nombre del archivo para grabar la lista de usuarios, pulsa ENTER para introducir por c

Existe el archivo 'user.dat' a continuación se muestran los datos grabados.

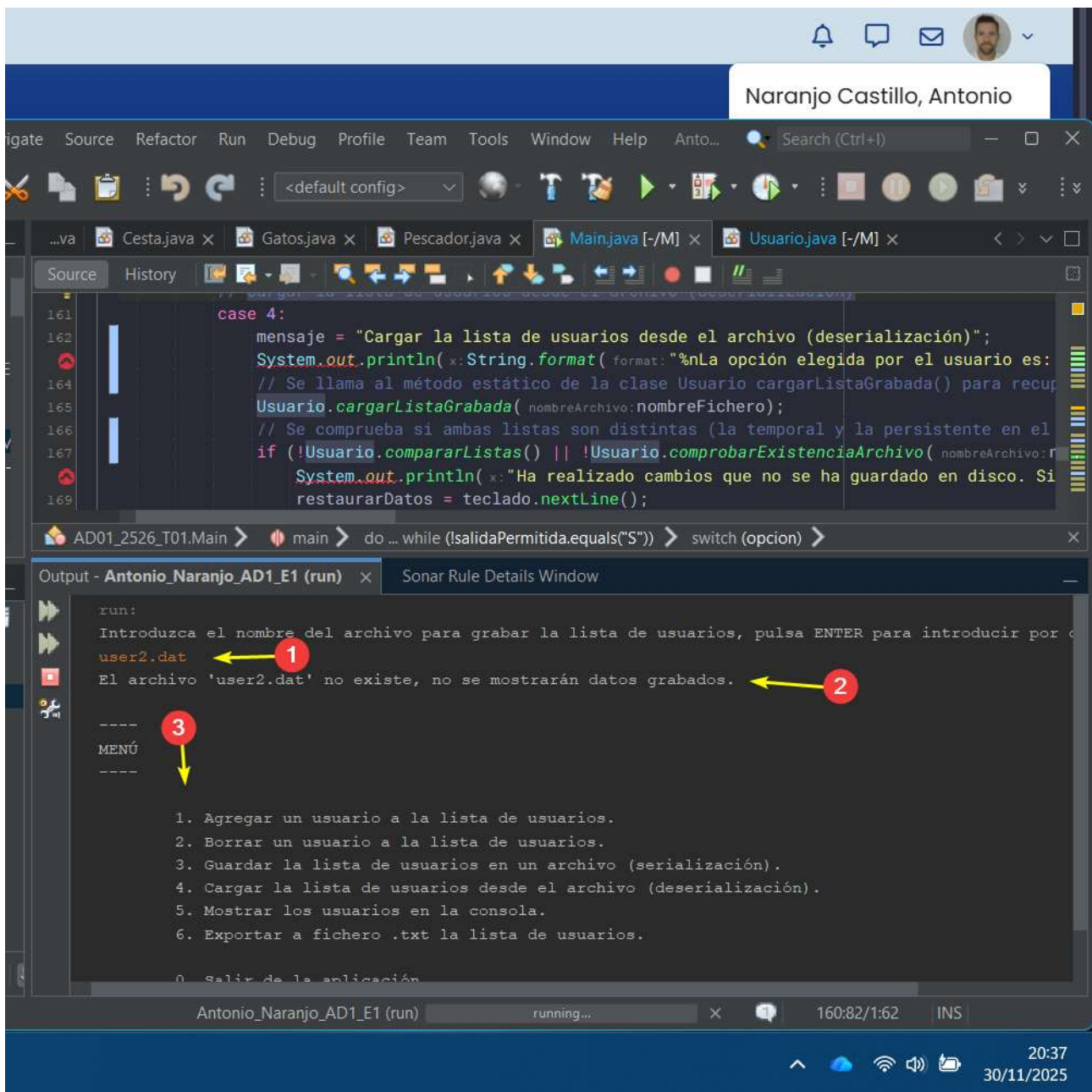
Datos grabados en disco:

Usuario [Identificador= 879wrtow, Contraseña= phoaeirhgqwe, Dirección= pjowehifq, Año de nacimiento=
Usuario [Identificador= 3248g, Contraseña= aksdhfi, Dirección= 23342fsadf, Año de nacimiento= 3432]
Usuario [Identificador= 898gerh, Contraseña= apsohdfw8e8rg, Dirección= 848489hapsi, Año de nacimient
Usuario [Identificador= 23245, Contraseña= asdfqwef, Dirección= qwertqwet, Año de nacimiento= 444]
Usuario [Identificador= yyggy, Contraseña= asQdoi, Dirección= qwrghopgbi, Año de nacimiento= 333]
Usuario [Identificador= 777, Contraseña= asdf, Dirección= qrwet, Año de nacimiento= 999]
Usuario [Identificador= 33234, Contraseña= asdf, Dirección= hqethr, Año de nacimiento= 778899]
Usuario [Identificador= 5678, Contraseña= asjtj, Dirección= ytkktr, Año de nacimiento= 3445]
Usuario [Identificador= 998877, Contraseña= bqiiBQnb, Dirección= qbqebe, Año de nacimiento= 43]
Usuario [Identificador= 8686, Contraseña= aUildhg, Dirección= naciifdbb, Año de nacimiento= 781
```

Antonio_Naranjo_AD1_E1 (run) running... 160:82/1:62 INS

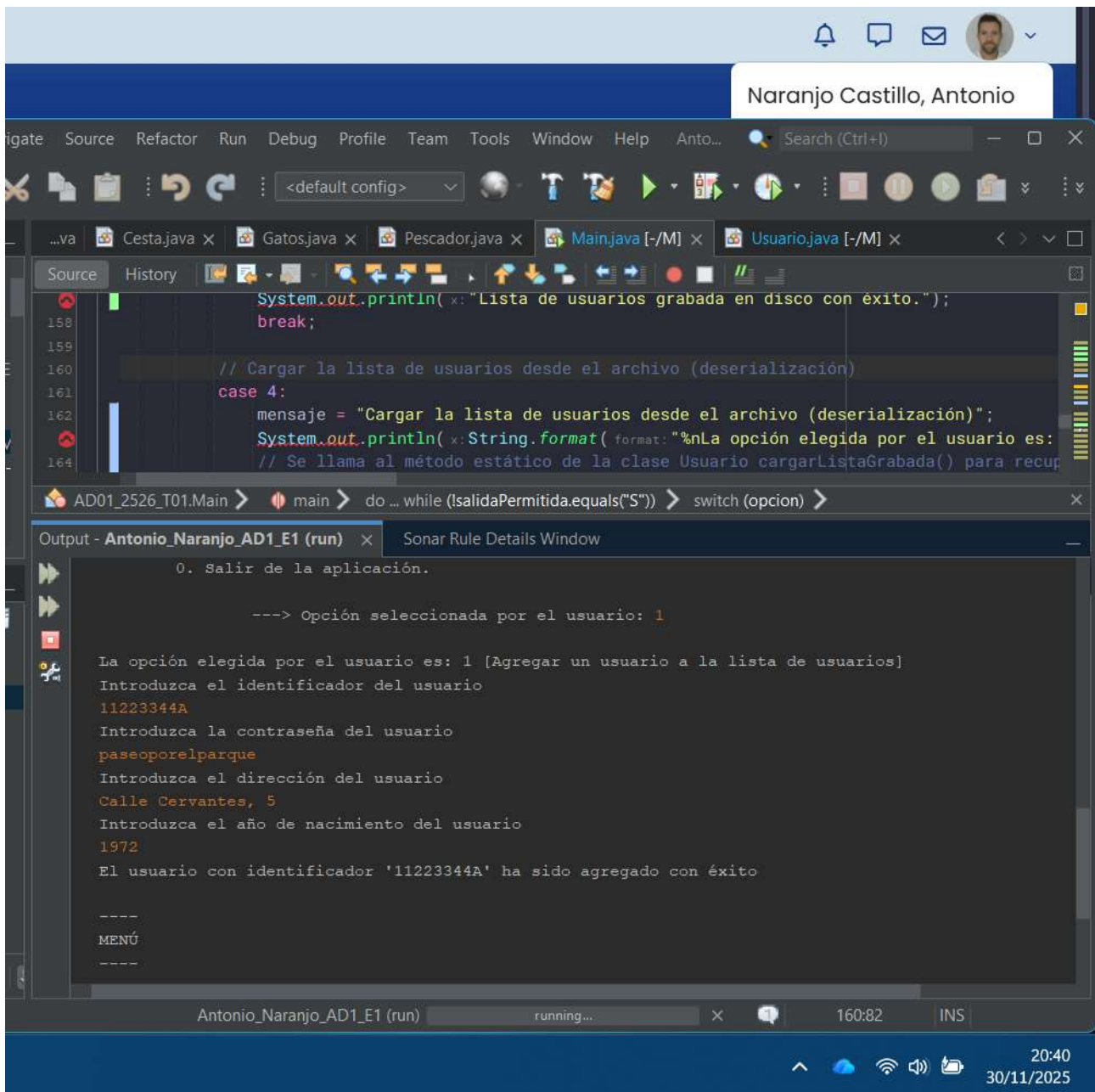
Al iniciar el programa se pregunta al usuario del programa el nombre del archivo binario. Si pulsa ENTER se asignará el nombre por defecto 'user.dat'.

Luego, si el archivo existe se imprime en consola y acto seguido se presenta la lista de usuarios guardados en el fichero.



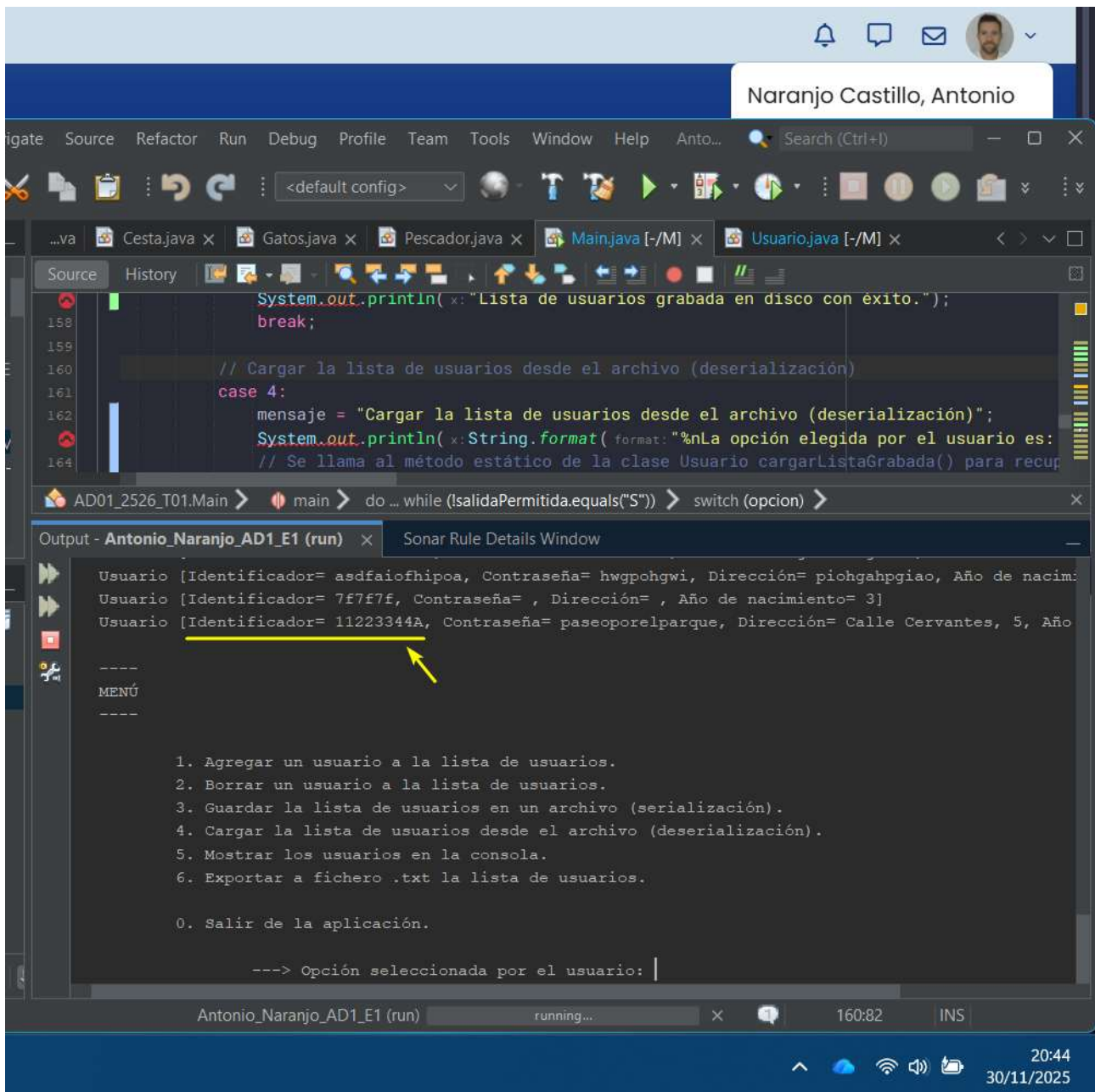
En caso de no existir se indica que no existe, para ello, se establece un nombre no existente en el directorio raíz del proyecto. Luego, se muestra el menú del programa.

b. Agregar un usuario



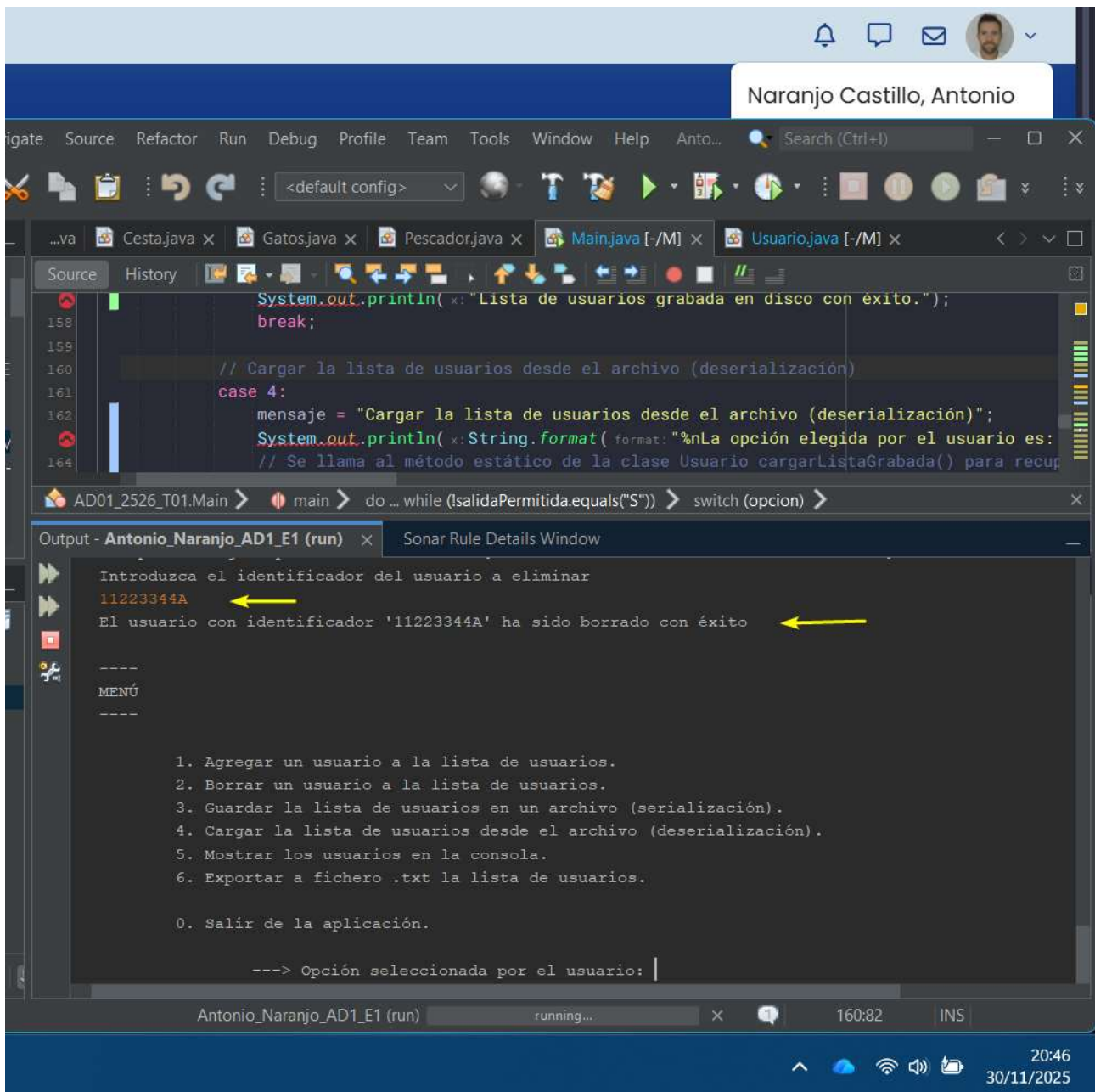
La opción 1 agrega un usuario previa solicitud de los atributos del objeto usuario. Acto seguido vuelve a mostrarse el menú.

c. Mostrar los usuarios en consola

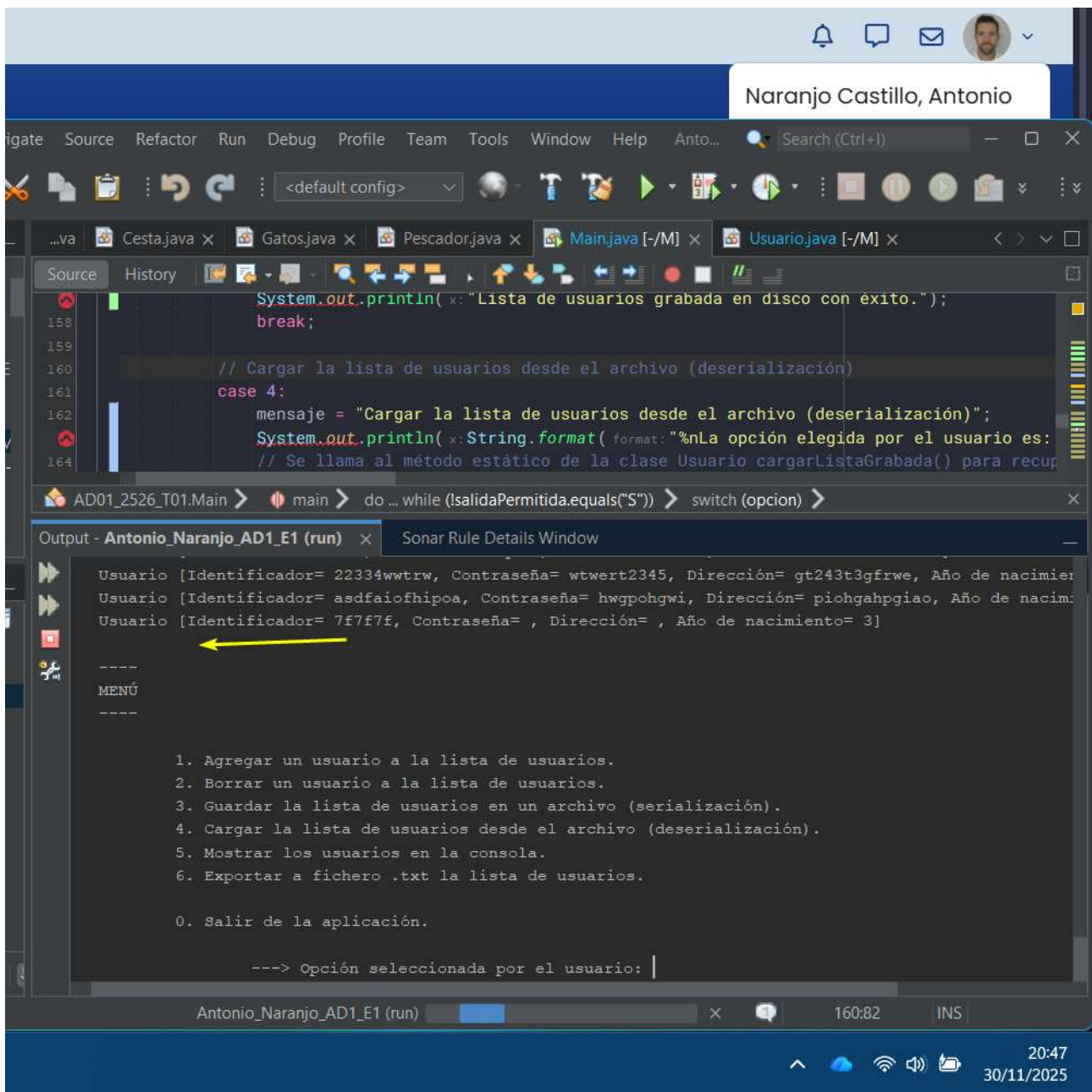


Se selecciona la opción 5 para mostrar todos los usuarios observándose en último lugar el usuario recientemente añadido (a la lista temporal).

d. Borrar usuario

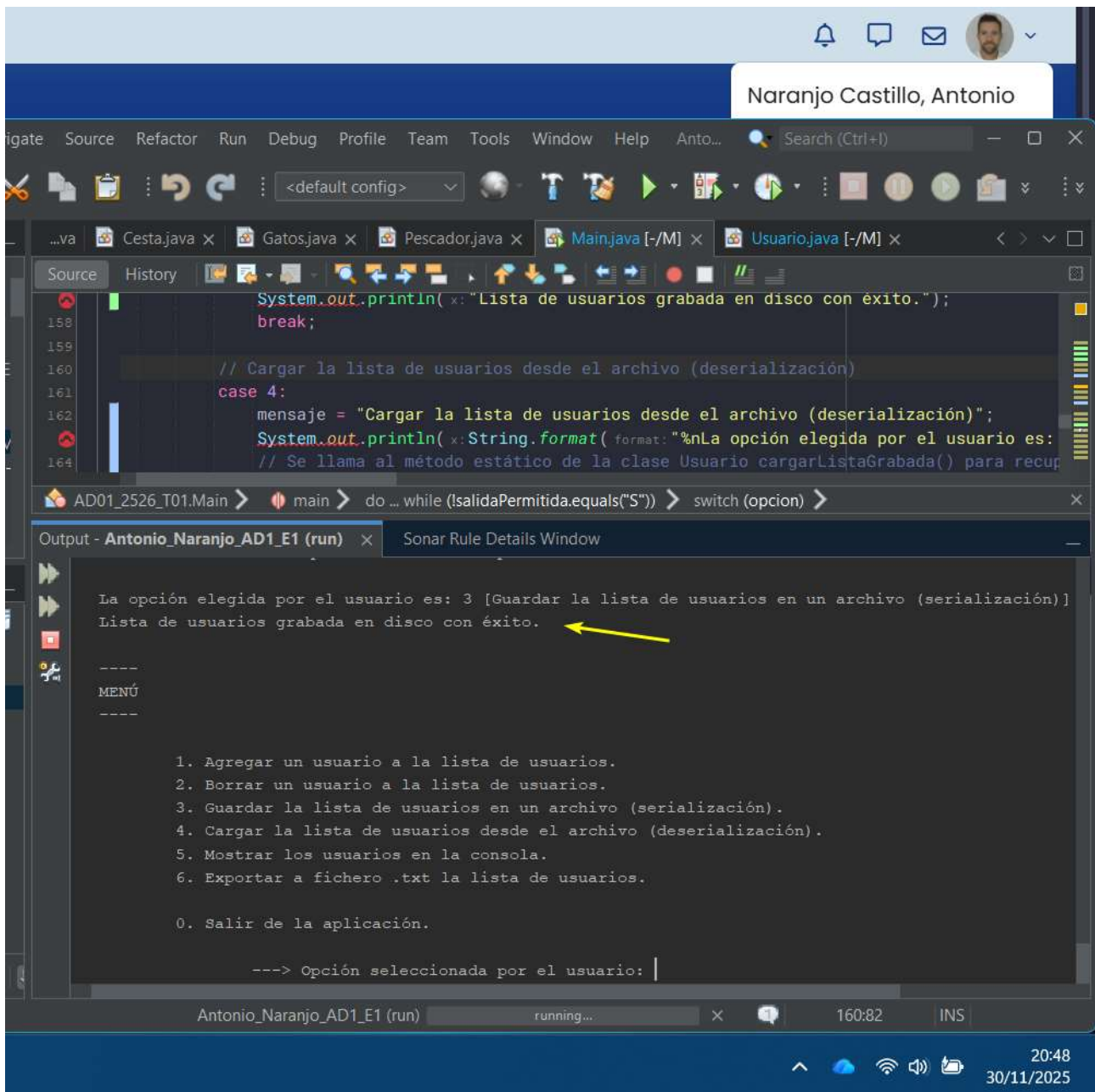


Se selecciona la opción 2 y se pasa como argumento la identificación del usuario, se aporta la identificación del usuario recientemente aportado.

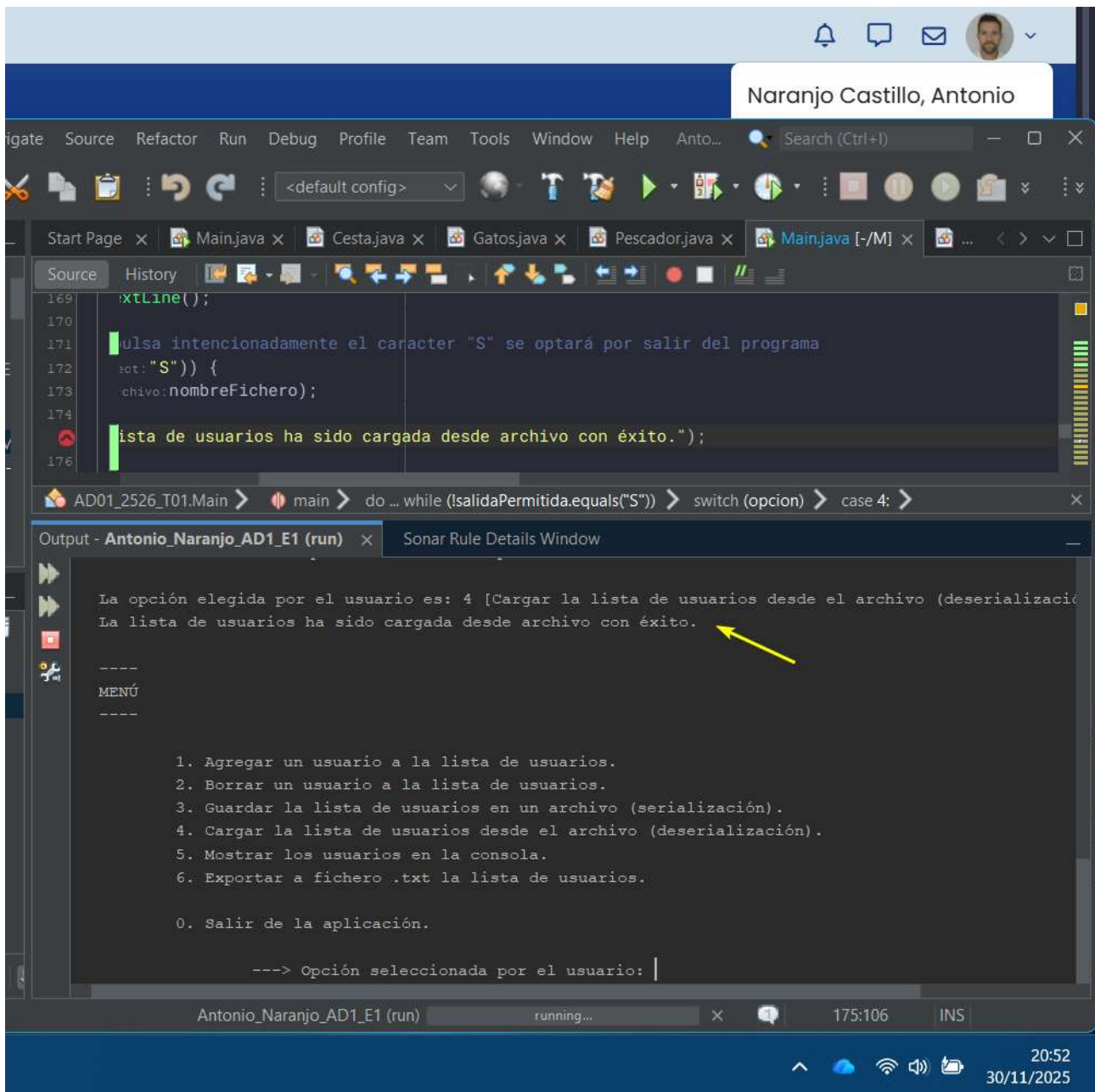


Se comprueba que el usuario ha sido eliminado con éxito volviendo a mostrar la lista de usuarios y observando que en último lugar ya no aparece el usuario anterior.

e. Guardar lista (Serialización)

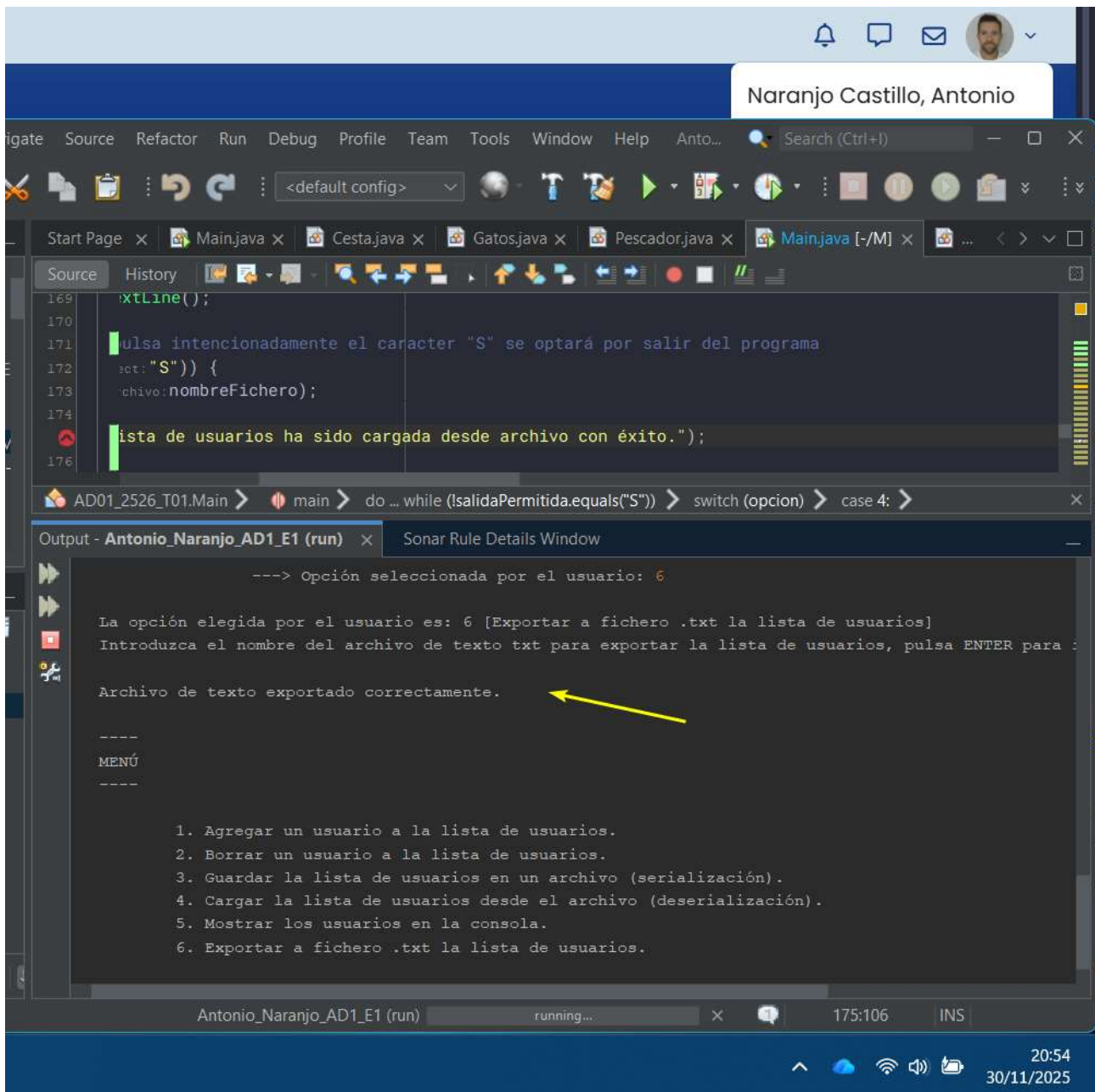


Se selecciona la opción 3 del menú y se muestra un mensaje tras el guardado satisfactorio.

f. Cargar lista (deserialización)

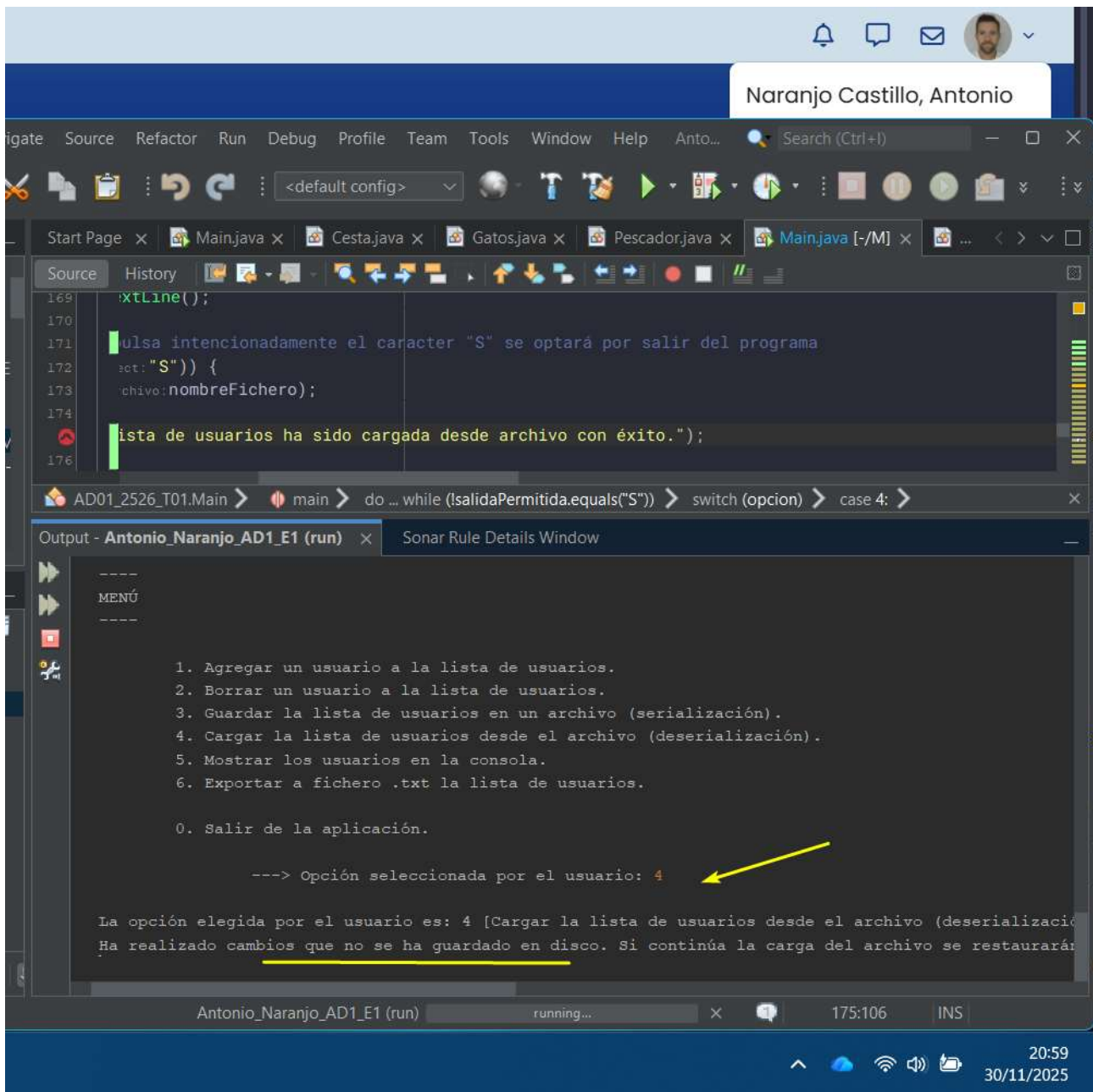
Se selecciona la opción 4 del menú y se muestra un mensaje tras la carga satisfactoria.

g. Exportación archivo TXT



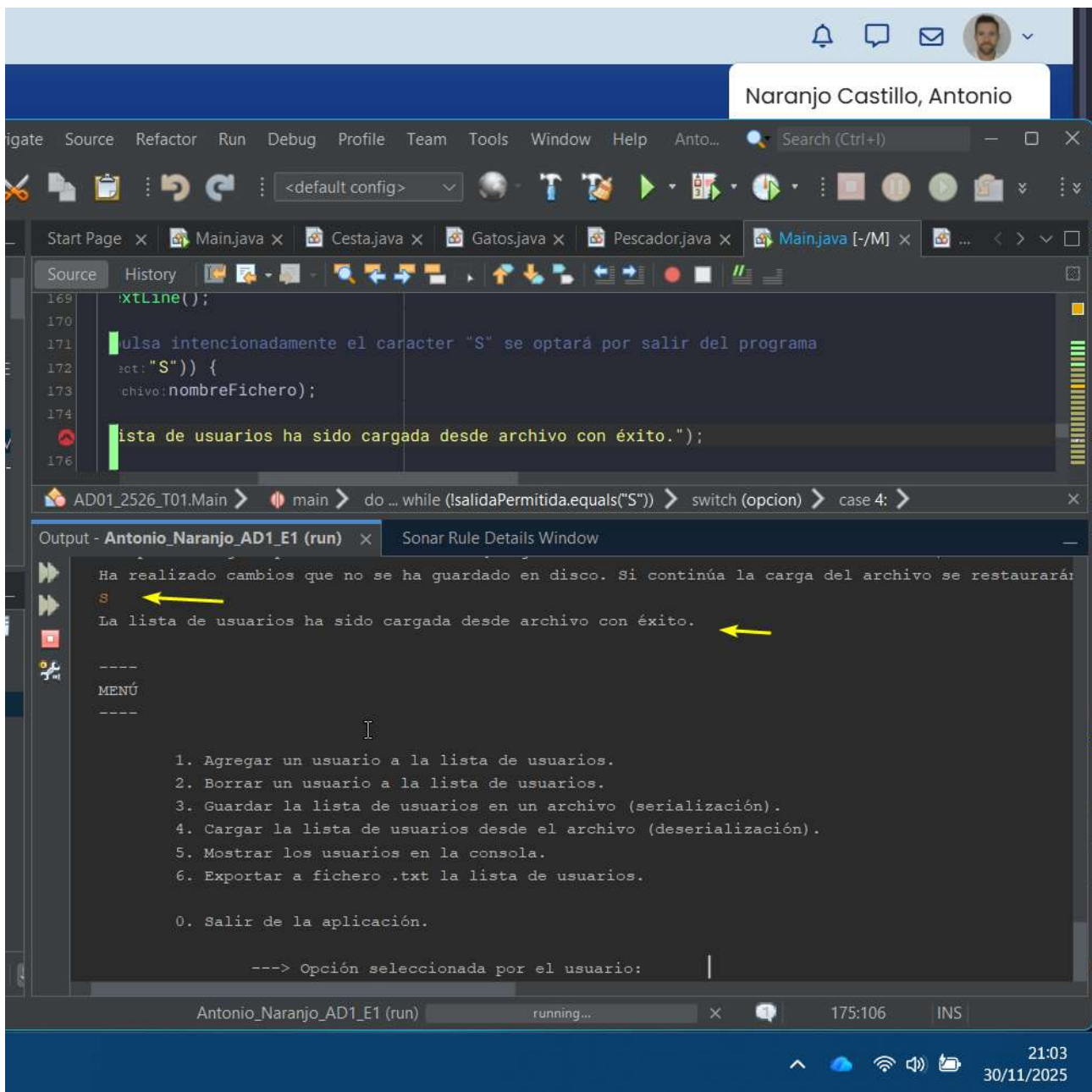
Se selecciona la opción 6 para proceder con la exportación del fichero de texto, se lanza un mensaje si se ha llevado a cabo satisfactoriamente y se vuelve a mostrar el menú.

h. ¿Cargar datos guardados?



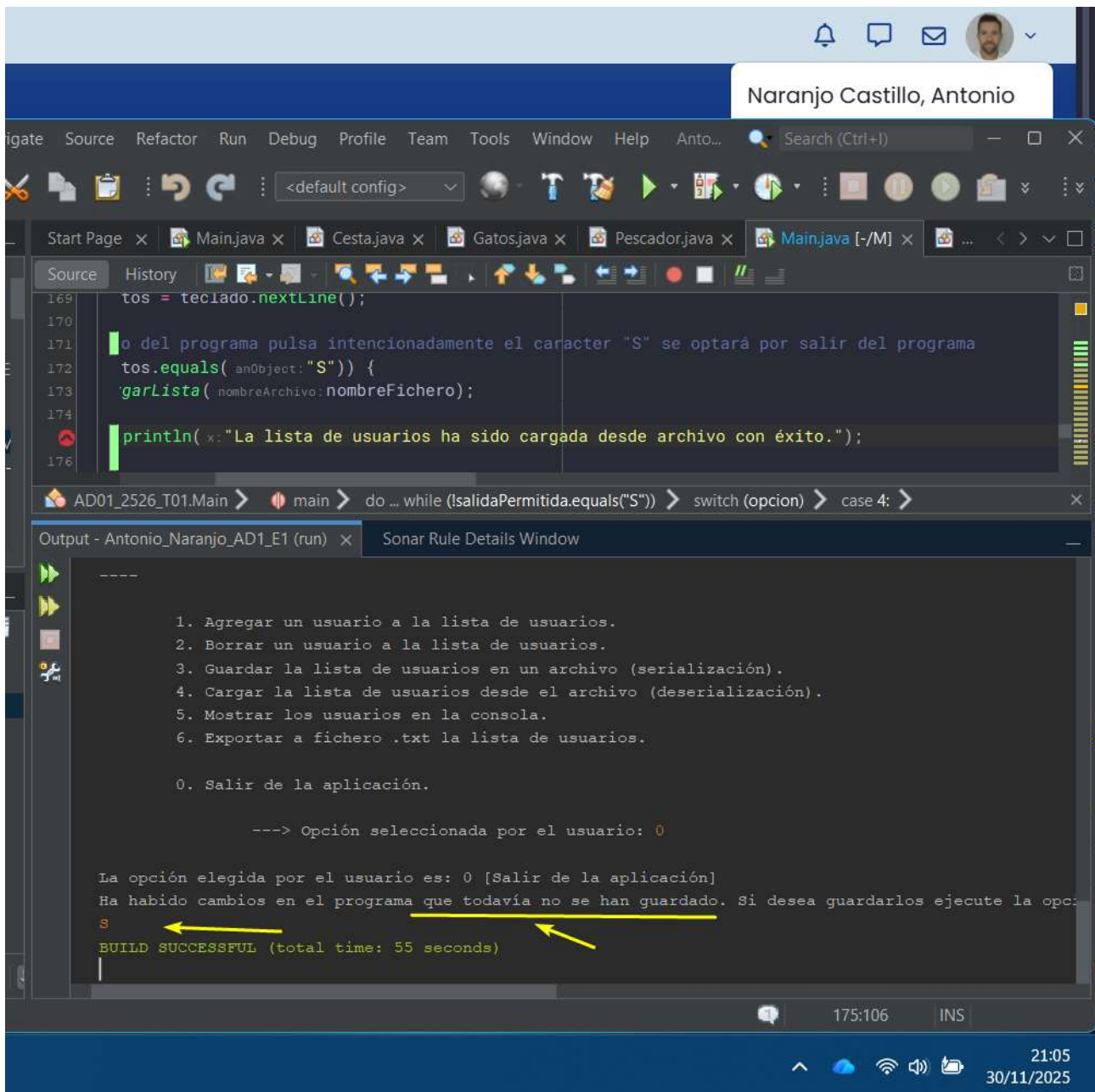
En caso de que el usuario pretenda cargar los datos guardado en el fichero binario, primeramente, el programa comparará la lista temporal de usuarios con la lista deserializada, de tal manera que si existen cambios de la opción de volver atrás y pueda guardar cambios o sustituya los valores si es finalmente la intención del usuario.

A modo de ejemplo se creó un usuario nuevo para alterar la lista temporal y así hacerla diferente a la deserializada.



Solo si el usuario pulsa "S" serán cuando se sustituyan los valores, de esta manera se elimina la posibilidad de pulsar cualquier otra letra accidentalmente.

i. Salida del programa



Al salir del programa, opción 0 del menú, si existen datos sin guardar se le mostrará al usuario un mensaje de salida advirtiéndolo de ello, solo si el usuario pulsa sobre el carácter "S" saldrá del programa.