Primera Aplicación. Parte 4

Últimos retoques

La aplicación ya es funcional, pero vamos a hacerle unos últimos retoques, para:

- Remarcar los campos requeridos
- Mostrar una ventana de error y finalizar la aplicación en caso de no poder conectar con la BD
- bloquear el redimensionado de la ventana
- Impedir que el texto en el campo 'notas' sobrepase el límite horizontal
- Agregar escuchadores para email y password
- Localización o Internacionalización: traducir la aplicación a varios idiomas
- Utilización de controles ChoiceBox (para seleccionar el idioma)
- Limitar el tiempo en que se muestran los mensajes en la parte inferior de la aplicación

Remarcar los campos requeridos

Normalmente se suelen remarcar de algún modo los datos que el usuario ha de rellenar obligatoriamente. En este caso todos menos los Apellidos y las Notas. Un modo fácil sería añadir una etiqueta a los mismos, por ejemplo, un * en un color diferente al texto:

Mostrar una ventana de error y finalizar la aplicación en caso de no poder conectar con la BD

Toda la aplicación está basada en operaciones sobre una tabla de una BD. Si la conexión con la BD no se puede establecer, no tiene sentido continuar con la aplicación. Lo que haremos será mostrar una ventana (Alerta) de error que una vez cerrada de por finalizada la aplicación (no se llegará a mostrar la pantalla principal).

Esto lo podemos hacer en JavaFX mediante la clase Alert. Hay varios tipos de Alert:

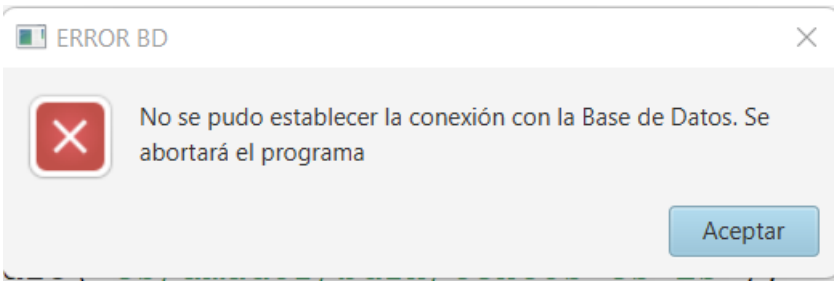
- ERROR
- INFORMATION
- WARNING
- CONFIRMATION

El funcionamiento es similar en todas. Asignamos un texto como título, opcionalmente un texto como cabecera, y por último un texto con el mensaje principal. Este podría ser un método para mostrar una alerta de error en caso de no podernos conectar a la BD:

```
  static private void mostrarAlertError(String titulo, String texto) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setHeaderText(null);
    alert.setTitle(titulo);
    alert.setContentText(texto);
    alert.showAndWait();
   }
```

Y lo llamaríamos, en caso de que el método conectar() fallase:

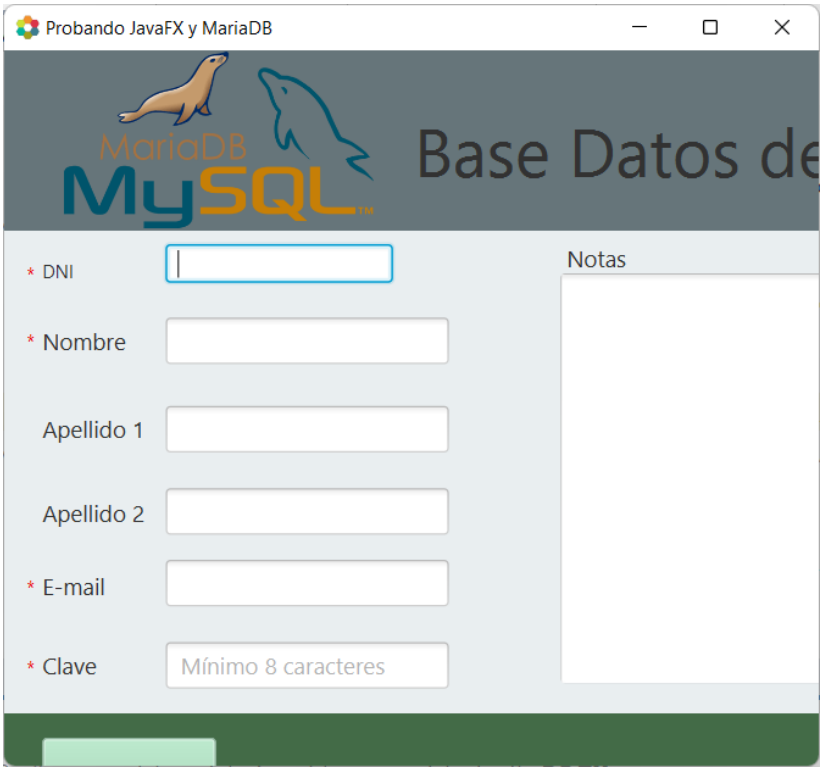
```
     if (!bd.conectar()) {
      mostrarAlertError("ERROR BD",”No se pudeo conectar con la BD. Se abortará el programa”);
      Platform.exit();
      System.exit(0);
     }
```



Hay que tener en cuenta que ahora la llamada al método para conectar no podrá estar en el método 'main'. Tal como lo teníamos, en 'main' conectábamos con la BD, llamábamos al método 'launch' y por último desconectábamos de la BD. Pero, el método 'launch' es el que inicializa el entrono gráfico, con lo cual, si vamos a querer presentar una ventana de error, no lo podremos hacer antes de haber lanzado 'launch'. Lo que podemos hacer es quitar el código que teníamos antes de la llamada a 'launch' y ponerlo dentro del método 'start'.

Bloquear el redimensionado de la ventana

Por defecto las ventanas pueden ser redimensionadas libremente, pero en ocasiones puede ser interesante bloquear esta característica parar por ejemplo impedir que al hacer más pequeña la ventana, se pierdan de vista parte de los elementos.



Vemos que, al redimensionar, perdemos de vista los botones de operaciones sobre la BD y el botón de "Borrar Datos" solo es parcialmente visible. Para impedir esto, añadiremos una línea de código al final del método "start" de la clase "App"

```
@Override
public void start(Stage stage) throws IOException {
  [...]
  stage.setResizable(false);
 }
```

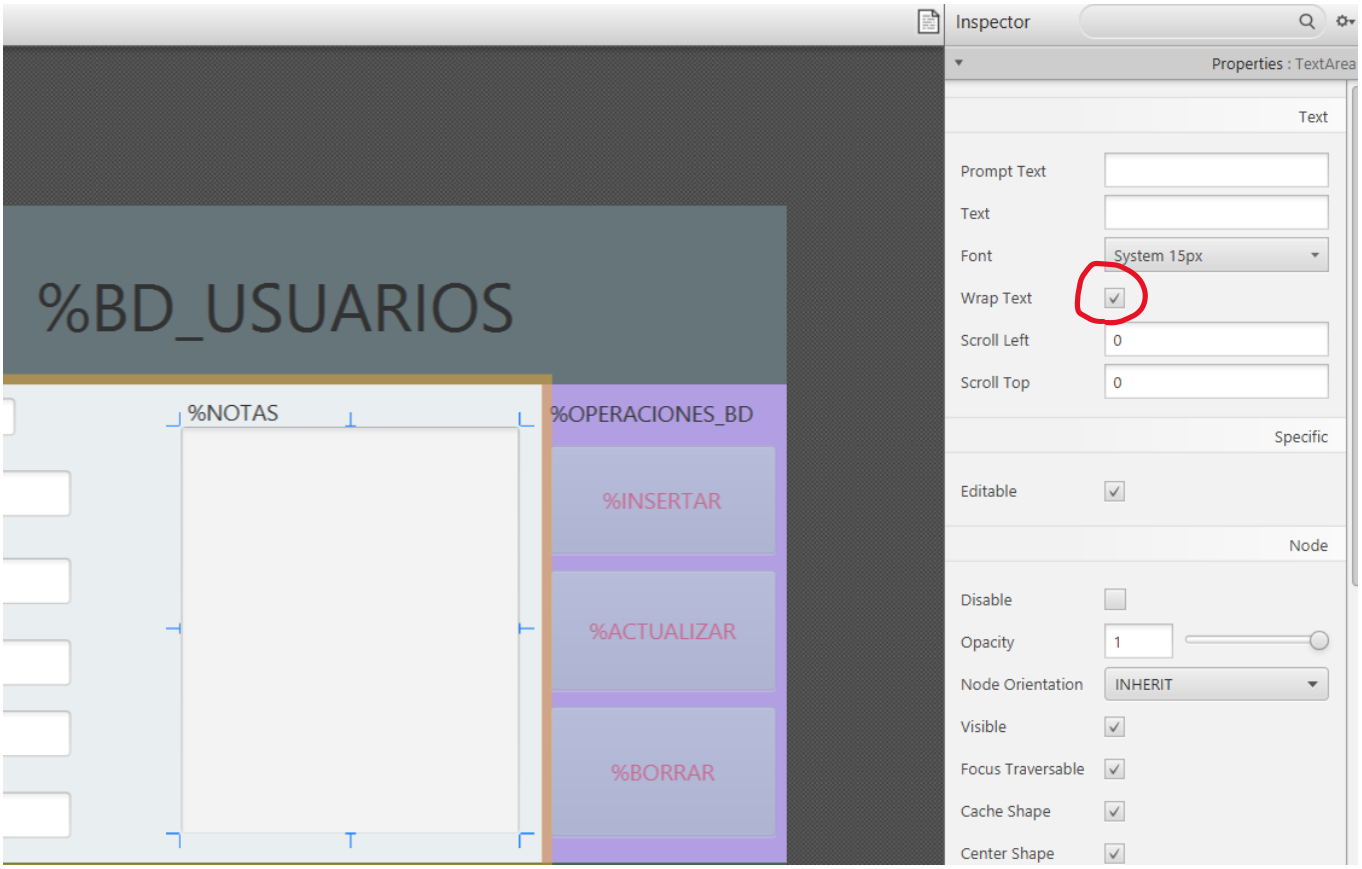
Impedir que el texto en el campo ‘notas’ sobrepase el límite horizontal

Si en una ‘TextArea’ metemos texto que sobrepase sus límites horizontal o verticalmente, automáticamente se mostrarán barras de desplazamiento que nos permitirán acceder a todo el contenido del texto. Mientras que el desplazamiento vertical es inevitable (salvo que limitemos la cantidad de líneas posibles como entrada), en el campo horizontal tenemos la posibilidad de hacer que si una línea sobrepasa el límite de la zona de la ‘TextArea’ su texto pasa a visualizarse automáticamente una línea más abajo (es solo la visualización, NO se inserte en el texto ningún carácter de Nueva Línea o Retorno de Carro), lo que aumenta la visibilidad del texto.

Ejemplo: Introducimos una línea de texto que sobrepasa el tamaño horizontal de la ‘TextArea’. Solo vemos una parte de este. Para ver el resto debemos utilizar la barra de desplazamiento que se ha añadió en la parte inferior.

Activando ‘Wrap Text’, el mismo texto se vería así:

Para activar esta opción podemos añadirla a mano en el '.fxml', o bien desde SceneBuilder seleccionar el chekbox que se muestra a continuación:



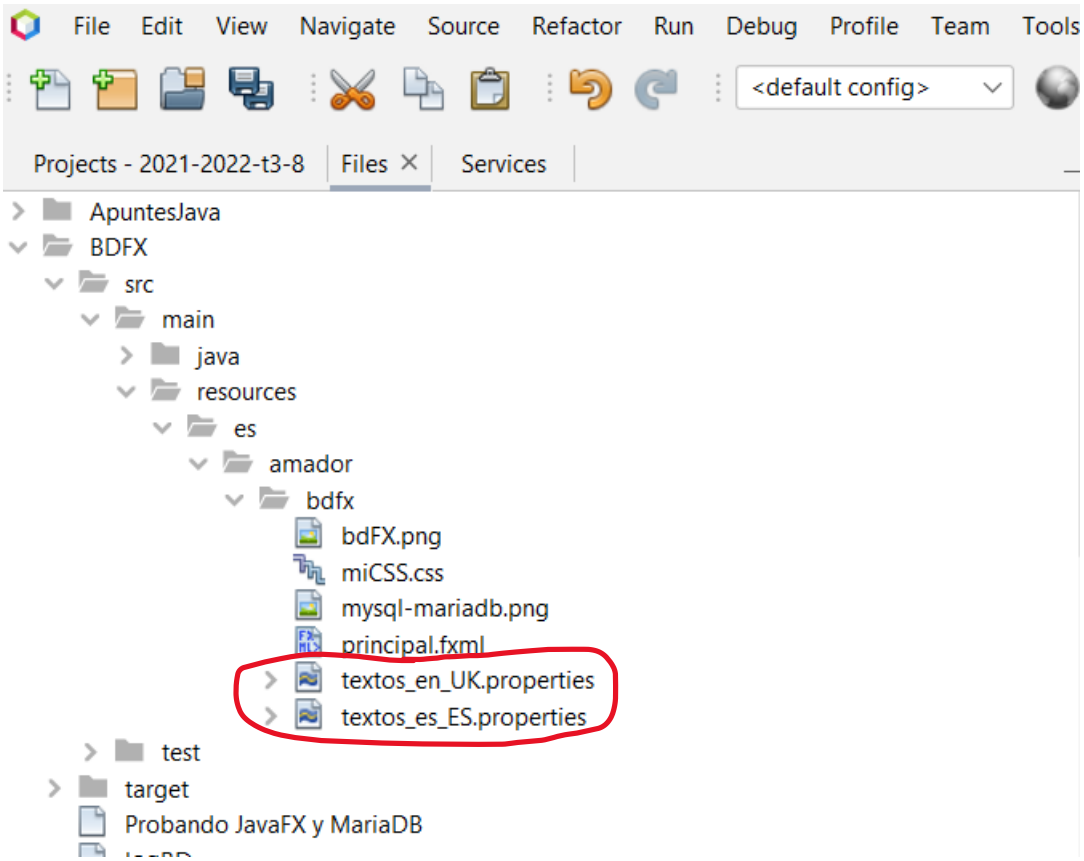
Agregar escuchadores para email y password

Hay un par de campos que necesitan una comprobación adicional. Email debe validarse frente a un patrón para determinar si es una dirección válida sintácticamente; password requiere al menos 8 caracteres. Tal como está desarrollada la aplicación, estas comprobaciones no se realizan hasta pulsar sobre uno de los botones de operaciones de BD, junto con la comprobación de que no existan campos requeridos vacíos.

Una adición interesante sería que la comprobación de estos dos campos se realizase en el mismo momento en que son editados. Para lleva a cabo esto operaríamos de modo similar a como hemos hecho con el campo 'DNI', esto es, le asignamos un 'listener' en el que pondremos el código de validación.

Localización o Internacionalización: traducir la aplicación a varios idiomas

Queremos que nuestra aplicación se pueda mostrar en diferentes idiomas sin tener que cambiar el código para cada una de ellas. Tal como hemos desarrollado la aplicación, todos los textos, tanto en la parte Java como en la parte FXML están 'hard-coded'; lo que vamos a hacer es, en lugar de colocar el texto a mostrar en nuestro código, colocaremos unas palabras clave. Con estas claves construiremos tantos ficheros como idiomas queramos utilizar. En cada uno de estos ficheros aparecerá cada una de las claves junto con el texto a mostrar en dicho idioma.
Los ficheros con las traducciones pueden tener cualquier nombre, pero se suele utilizar un sufijo con el código del idima, por ejmplo, es_ES para español internacional, o en_UK para inglés de Gran Bretaña. Sus extensiones serán ".properties", y deben de estar colocados bajo la carpeta 'resources'



Contenido de 'textos_en_UK.properties'

```
ERROR_BD=BD ERROR
R_RECUPERADO=Record recovered
CONTRASEÑA_CORTA=Password too short. Minimum 8
EMAIL_INVÁLIDO=Email not valid
CAMPOS_REQUERIDOS=There are required fields without value
ERROR_ELIMINACIÓN=Error deleting row
R_ELIMINADO=Row deleted
ERROR_ACTUALIZACIÓN=Error updating row
[…]
```

Contenido de 'textos_es_ES.properties'

```
ERROR_BD=ERROR BD
R_RECUPERADO=Registro recuperado
CONTRASEÑA_CORTA=Contraseña demasiado corta. Mínimo 8
EMAIL_INVÁLIDO=Email no válido
CAMPOS_REQUERIDOS=Hay campos requeridos sin valor
ERROR_ELIMINACIÓN=Error en la eliminación del registro
R_ELIMINADO=Registro eliminado
ERROR_ACTUALIZACIÓN=Error en la actualización del registro
[…]
```

Como podemos ver, en cada línea va una clave, el signo '=', y el texto correspondiente a esa clave en el idioma que corresponde al fichero.

En el código java, comenzamos por cargar el fichero correspondiente. En la clase App.

```
public class App extends Application {

  public static String idioma = "Castellano"; // la utilizaremos más adelante
  static ResourceBundle textos = ResourceBundle.
         getBundle("es/amador/bdfx/textos_es_ES"); // asocia el fichero
```

Si hemos traducido también textos de la interfaz (parte en .fxml), necesitamos indicar al cargador de fxml que asocie al mismo el fichero .properties. Lo hacemos en el método 'loadFXML

```
private static Parent loadFXML(String fxml) throws IOException {
  FXMLLoader fxmlLoader
          = new FXMLLoader(App.class.getResource(fxml + ".fxml"),
                  textos);
  return fxmlLoader.load();
}
```

Nos falta colocar esas claves en el código. Distinguimos la parte Java de la parte FXML

En los ficheros Java. Donde tuviéramos un texto hard-coded como, por ejemplo:

```
if (passsword.getText().length() < 8) {
   mensaje(true, "Contraseña demasiado corta. Mínimo 8"));
   return false;
 }
```

Lo sustituimos por una llamada al método 'getString' pasándole como parámetro la clave correspondiente.

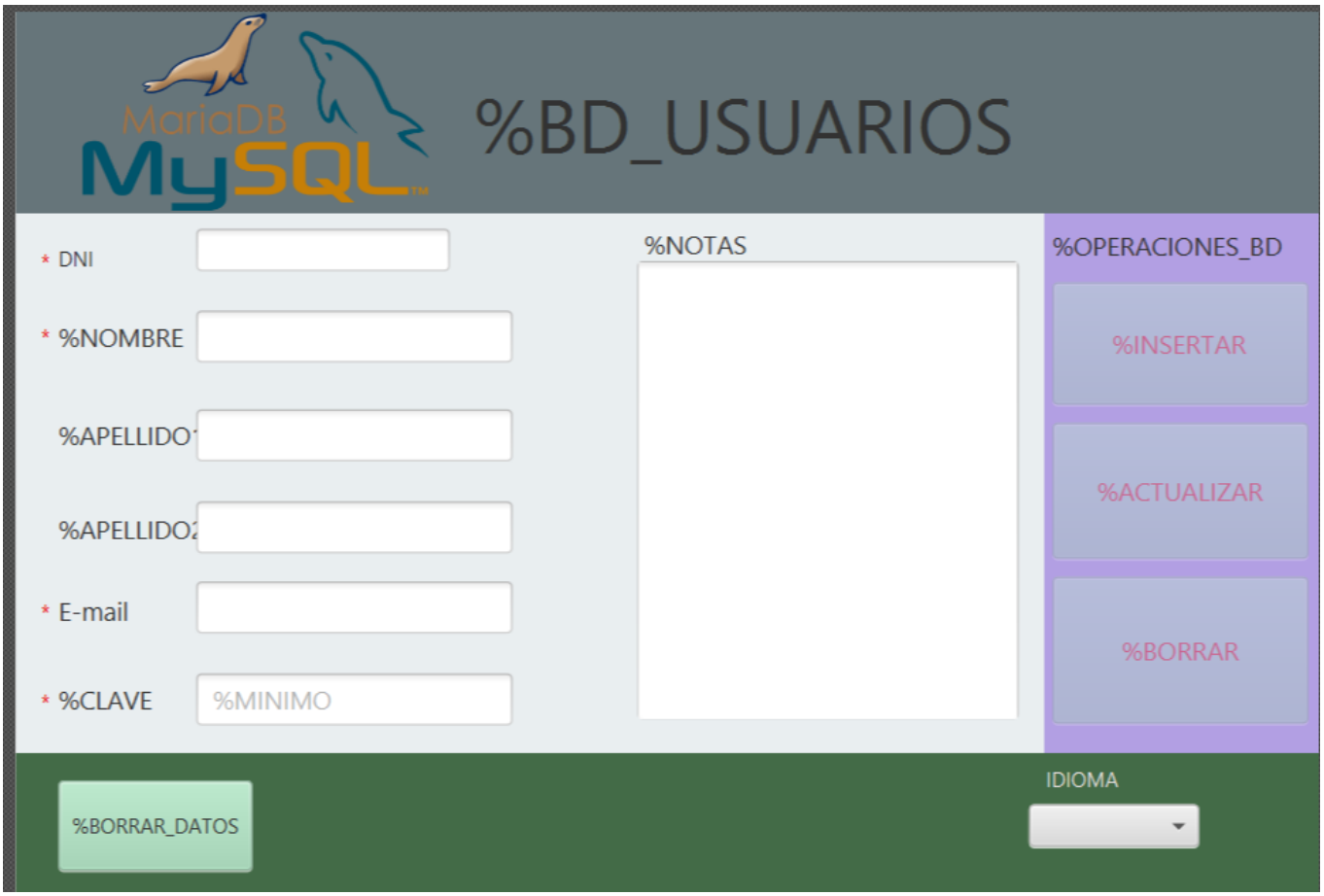
```
if (passsword.getText().length() < 8) {
   mensaje(true, App.textos.getString("CONTRASEÑA_CORTA"));
   return false;
 }
```

En los ficheros FXML, sustituimos los textos por la clave precedida por el símbolo %

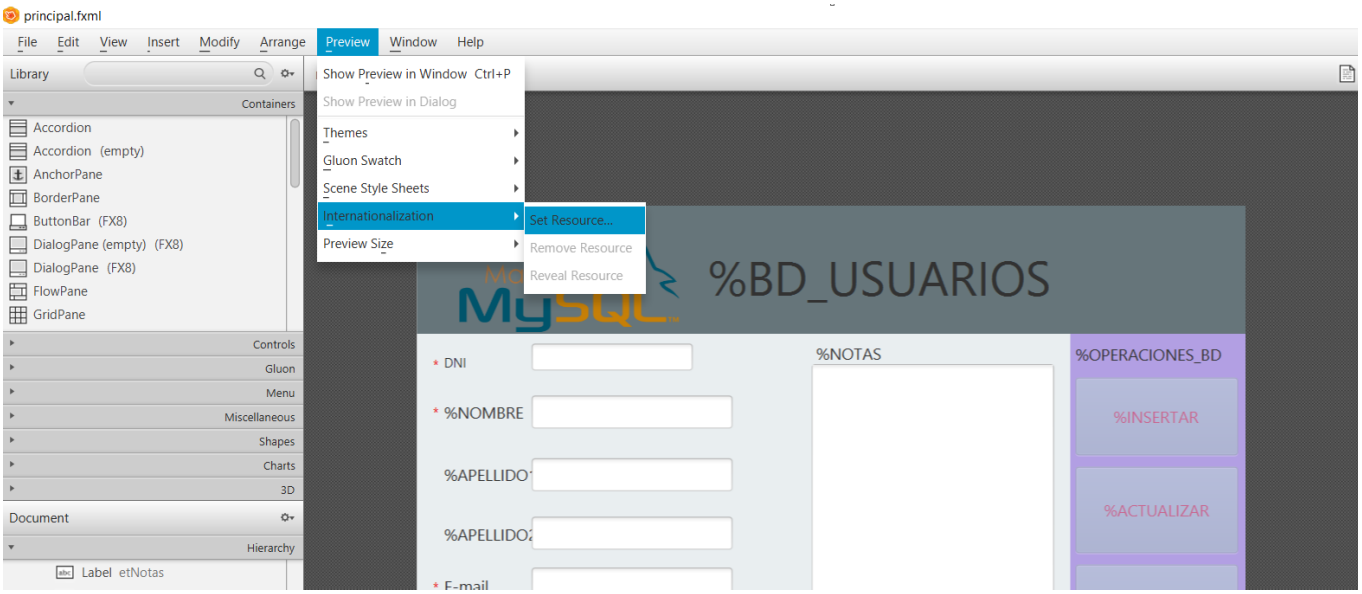
```
<PasswordField fx:id="passsword" layoutX="106.0" layoutY="270.0"
               promptText="%MINIMO">
   <font>
     <Font size="15.0" />
   </font>
 </PasswordField>
```

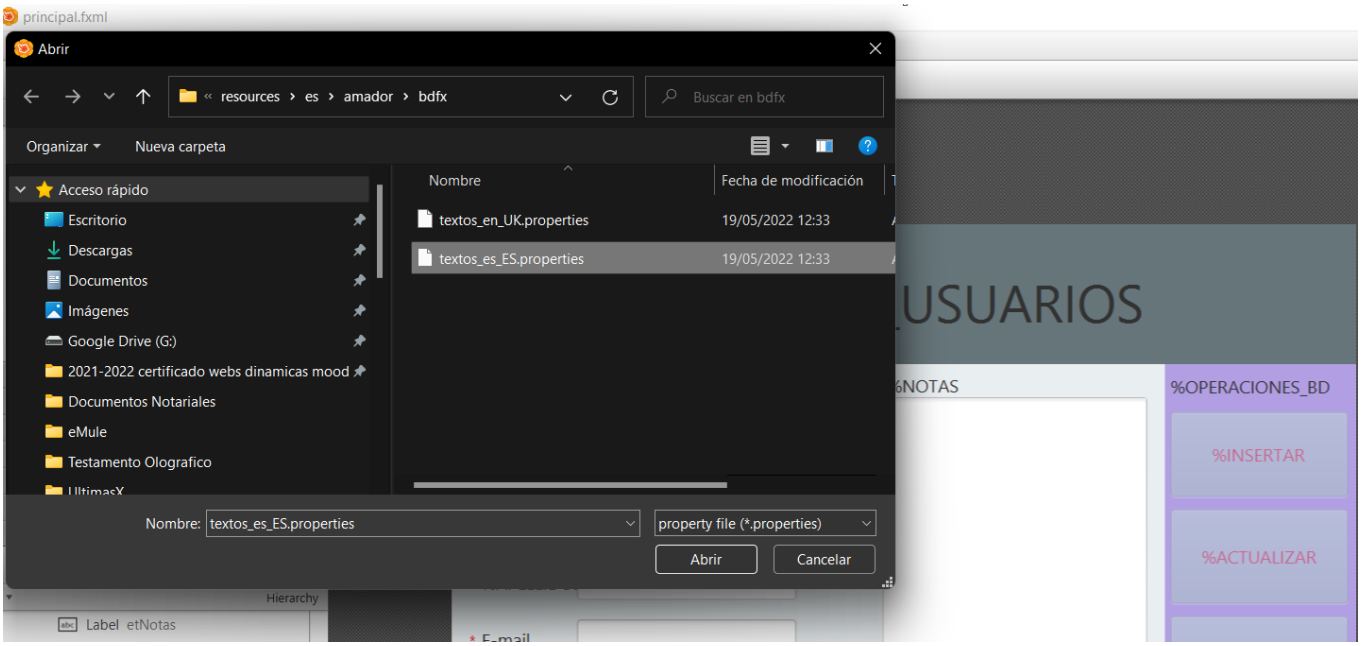
O bien desde SceneBuilder pulsando sobre la rueda dentada a la derecha de 'Text' escogemos 'Replace with internacionalized String'

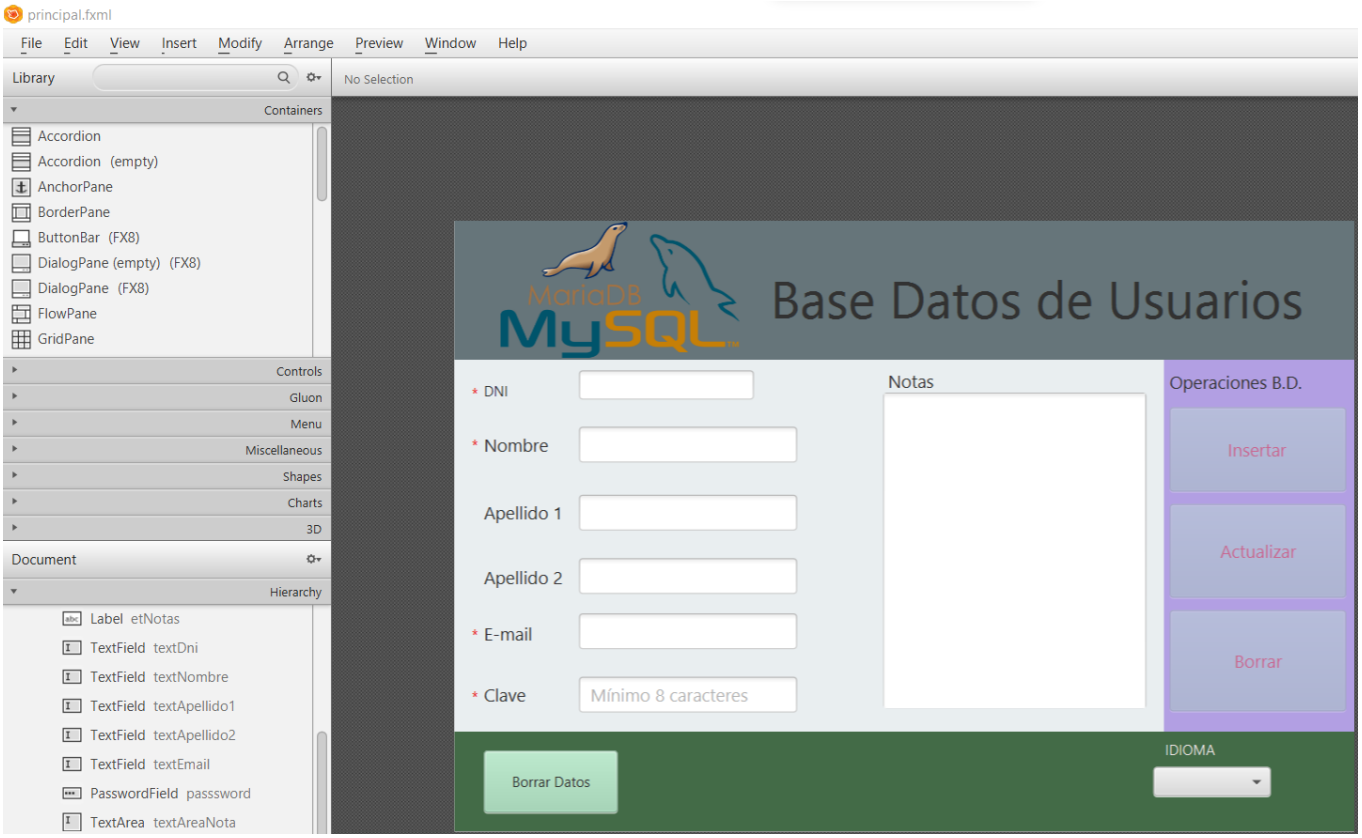
Nos quedará al final algo como:



Podemos previsualizar como quedaría con la aplicación en funcionamiento seleccionando uno de los ficheros de idioma en:


principal.fxml
File Edit View Insert Modify Arrange Preview Window Help
Library
Containers
Accordion
Accordion (empty)
AnchorPane
BorderPane
ButtonBar (FX8)
DialogPane (empty) (FX8)
DialogPane (FX8)
FlowPane
GridPane
Controls
Gluon
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
Label etNotas
Show Preview in Window Ctrl+P
Show Preview in Dialog
Themes
Gluon Swatch
Scene Style Sheets
Internationalization
Preview Size
Set Resource...
Remove Resource
Reveal Resource
%BD_USUARIOS
* DNI
* %NOMBRE
%APELLIDO
%APELLIDO2
* E-mail
%NOTAS
%OPERACIONES_BD
%INSERTAR
%ACTUALIZAR


principal.fxml
Abrir
resources › es › amador › bdfx
Buscar en bdfx
Organizar
Nueva carpeta
Acceso rápido
Escritorio
Descargas
Documentos
Imágenes
Google Drive (G:)
2021-2022 certificado webs dinamicas mood
Documentos Notariales
eMule
Testamento Olografico
Nombre
Fecha de modificación
textos_en_UK.properties 19/05/2022 12:33
textos_es_ES.properties 19/05/2022 12:33
Nombre: textos_es_ES.properties
property file (*.properties)
Abrir
Cancelar
Hierarchy
Label etNotas
USUARIOS
%NOTAS
%OPERACIONES_BD
%INSERTAR
%ACTUALIZAR

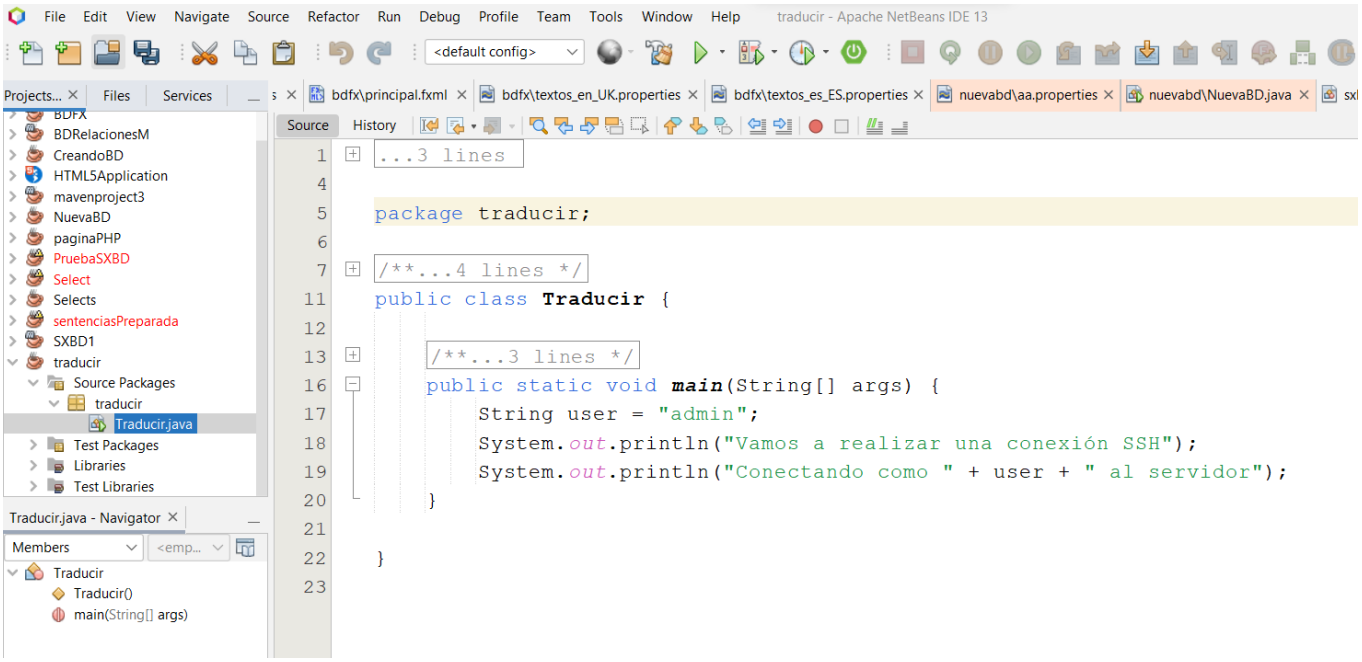


Asistente de Internacionalización

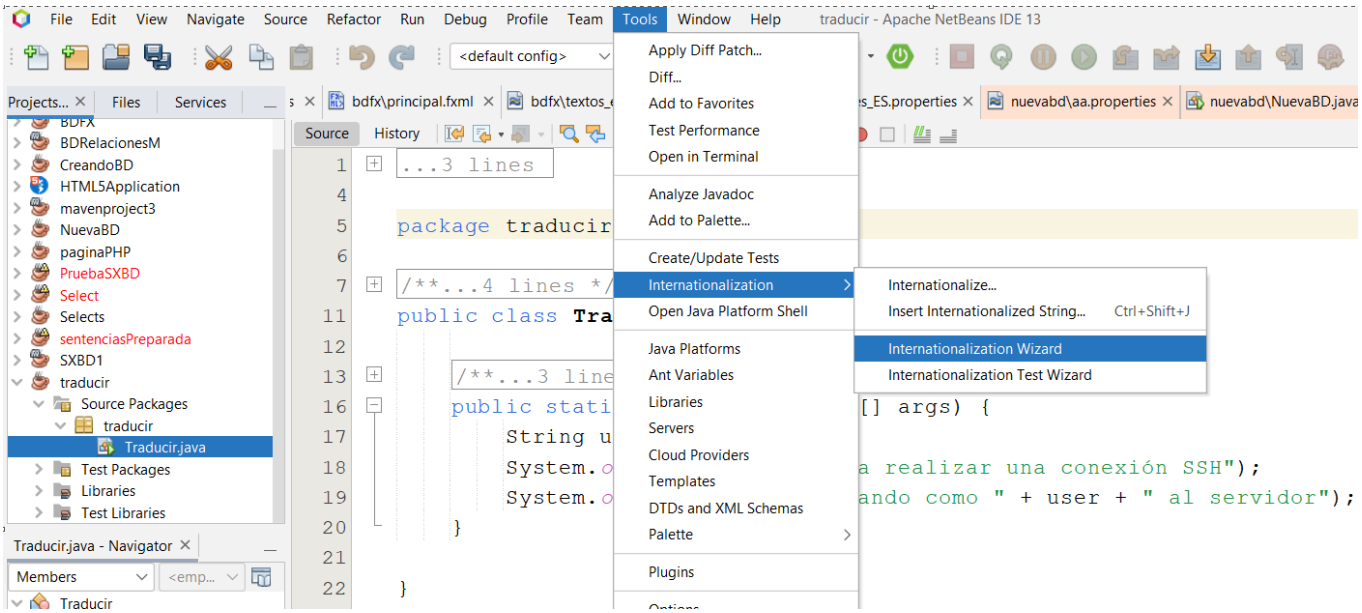
En Netbeans disponemos también de un asistente que nos puede ayudar con el proceso de internacionalización dentro del código Java.

IMPORTANTE: El asistente buscará todos los textos entrecomillados, pero no todos deben ser traducidos, por ejemplo, una sentencia SQL, o el nombre de usuario y la url para la conexión a la BD. Así que es muy importante revisar uno por uno los resultados que nos mostrará antes de aplicarlos.

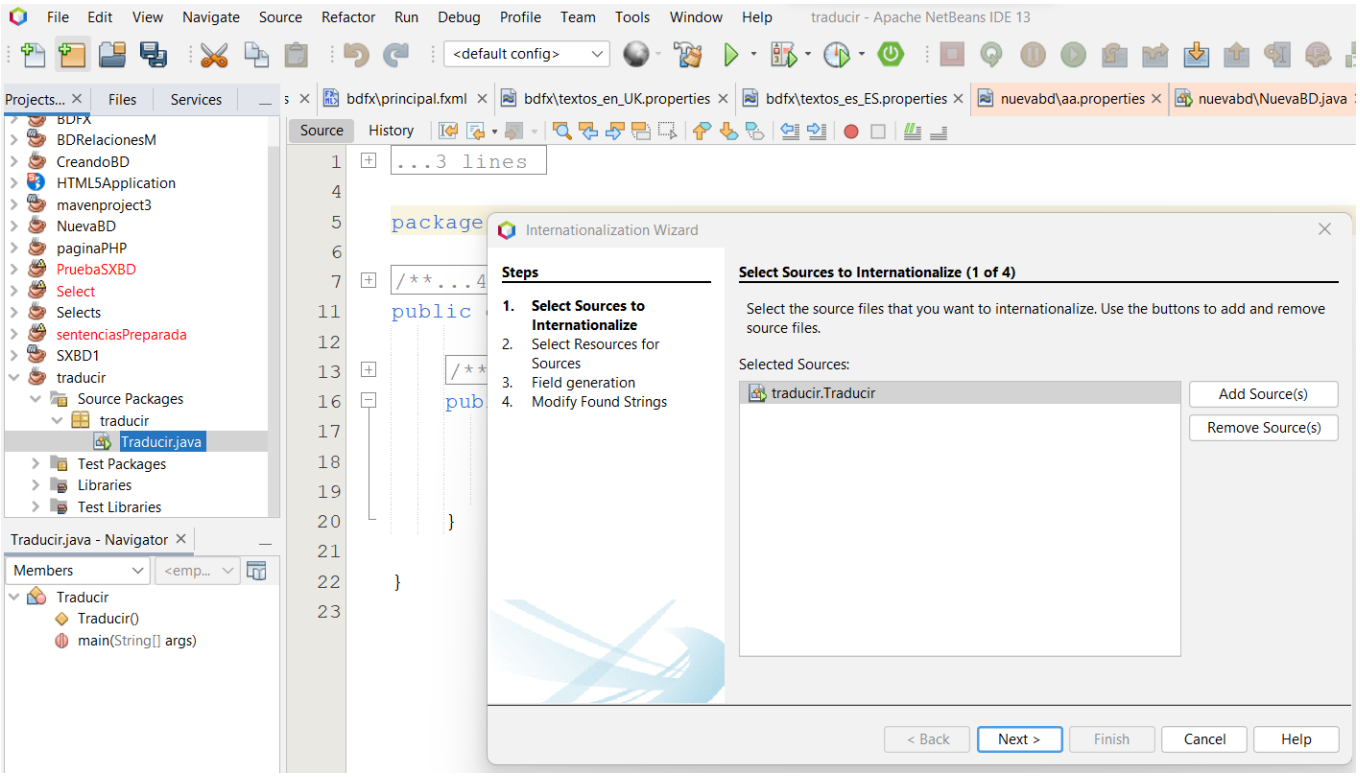
Vamos a ver un ejemplo con un programa sencillo para simplificar la explicación. En este programa solo hay tres cadenas. La primera no queremos que sea traducida pues es el nombre de usuario que vamos a utilizar en una conexión. La segunda y la tercera sí, pero la tercera está compuesta por concatenación de Strings con una variable.



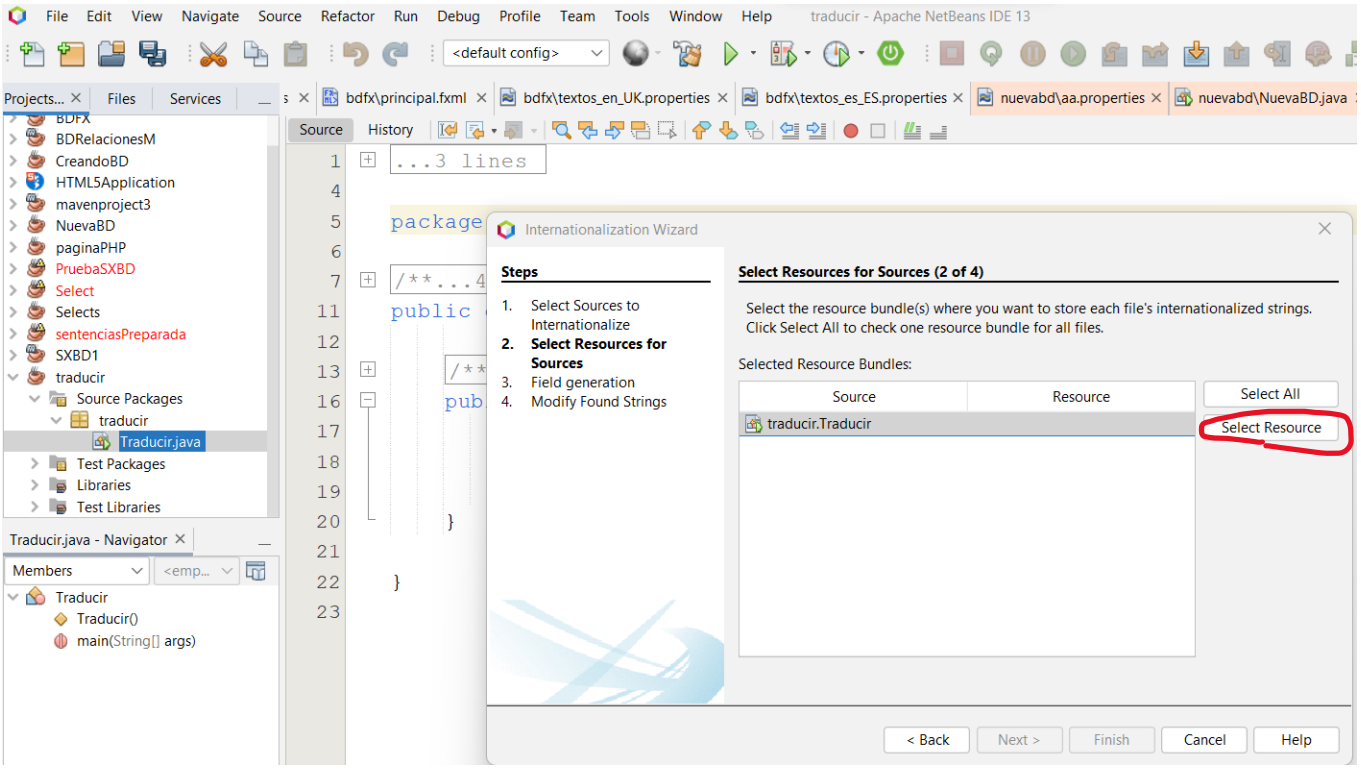
Lanzamos el asistente:

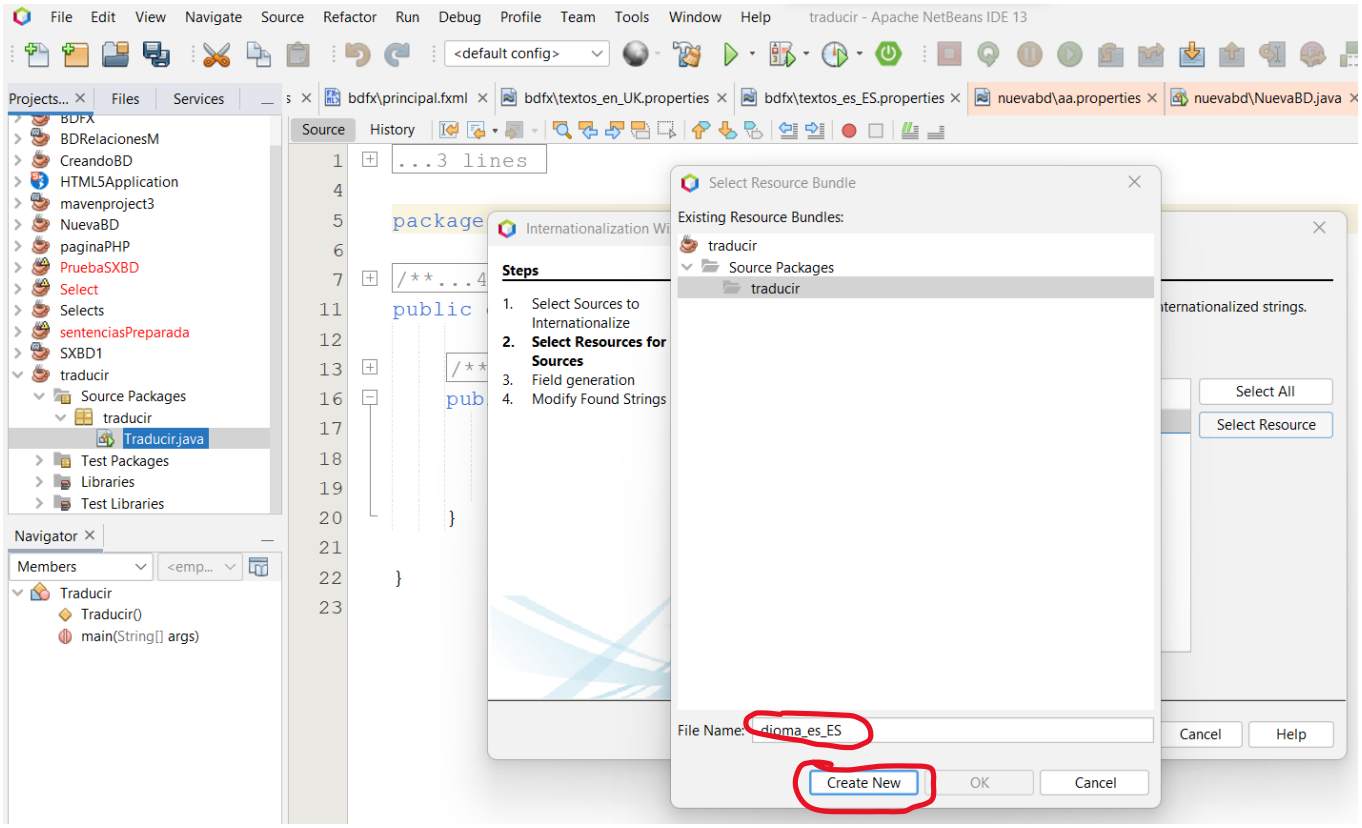


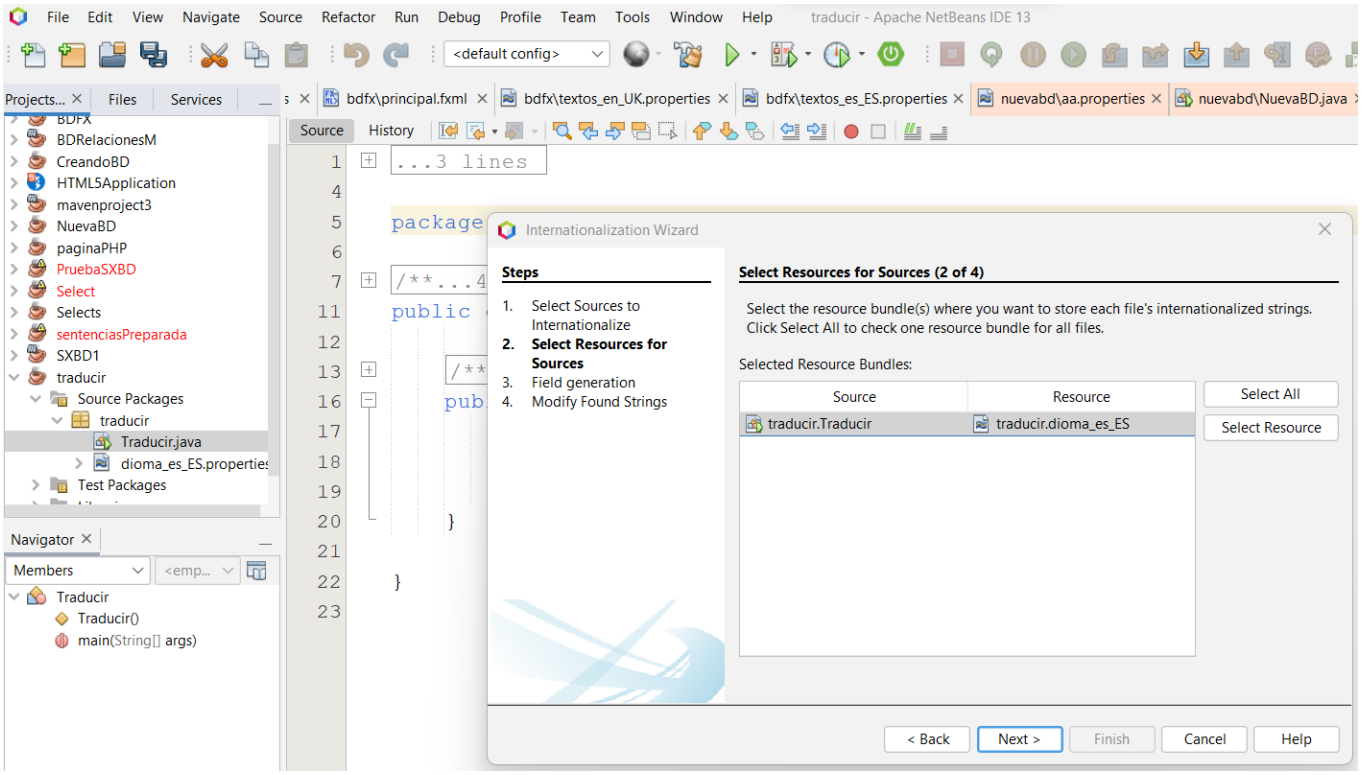
Seleccionamos los ficheros con cadenas a traducir. En este caso solo hay uno, Traducir.java



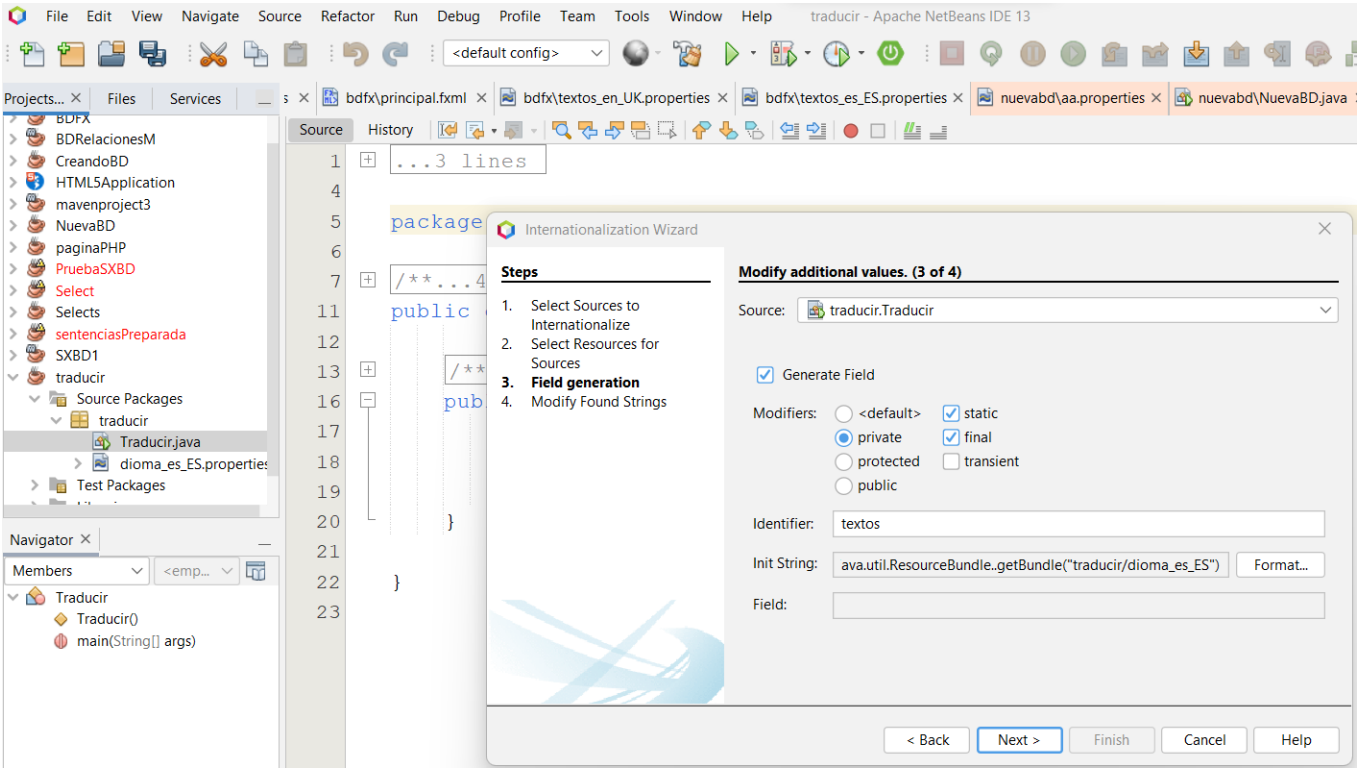
A continuación, nos pide el fichero que contendrá las traducciones. Podemos seleccionarlo si ya lo tenemos (no es el caso) o pedir que cree uno nuevo.



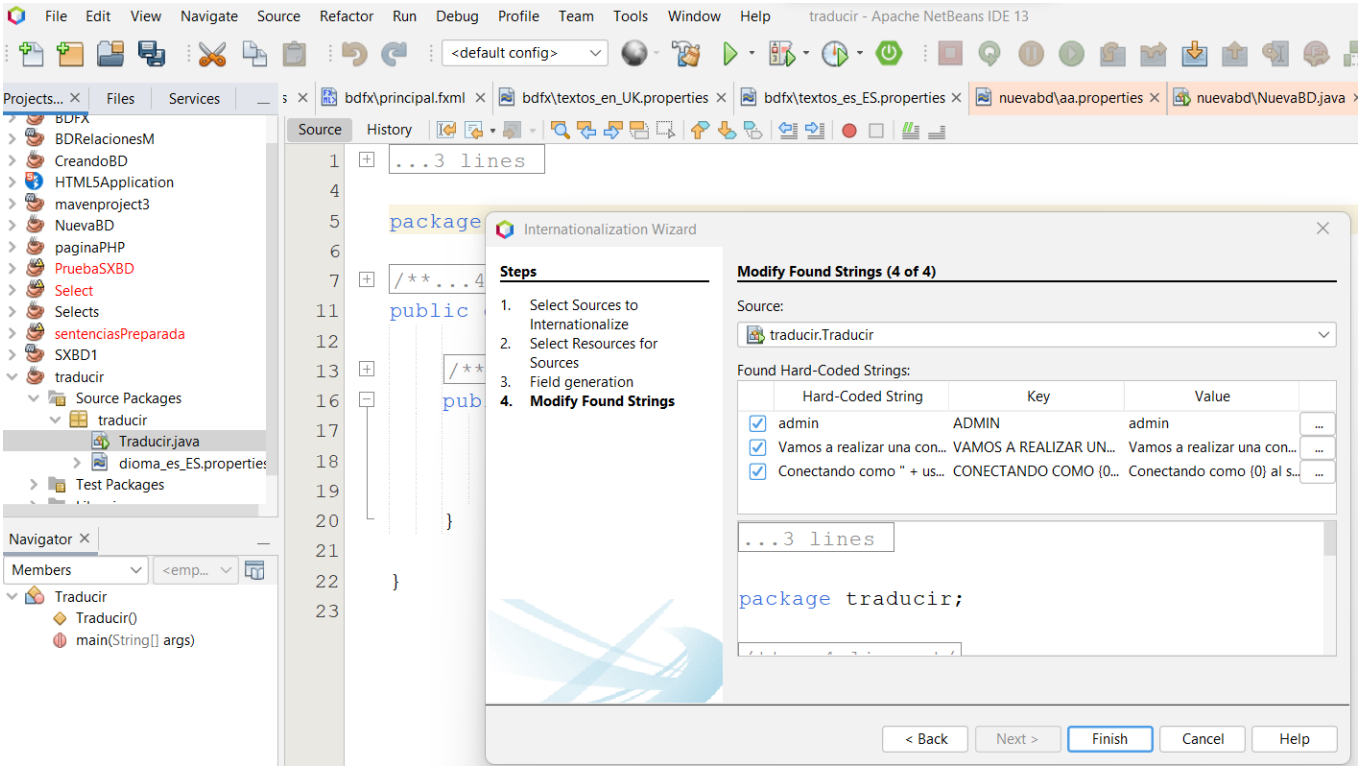

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
traducir - Apache NetBeans IDE 13
Select Resource Bundle
Existing Resource Bundles:
traducir
Source Packages
traducir
Steps
1. Select Sources to Internationalize
2. Select Resources for Sources
3. Field generation
4. Modify Found Strings
File Name: dioma_es_ES
Create New
OK
Cancel
Select All
Select Resource
Help


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
traducir - Apache NetBeans IDE 13
Internationalization Wizard
Steps
1. Select Sources to Internationalize
2. Select Resources for Sources
3. Field generation
4. Modify Found Strings
Select Resources for Sources (2 of 4)
Select the resource bundle(s) where you want to store each file's internationalized strings.
Click Select All to check one resource bundle for all files.
Selected Resource Bundles:
Source
Resource
traducir.Traducir
traducir.dioma_es_ES
Select All
Select Resource
< Back
Next >
Finish
Cancel
Help

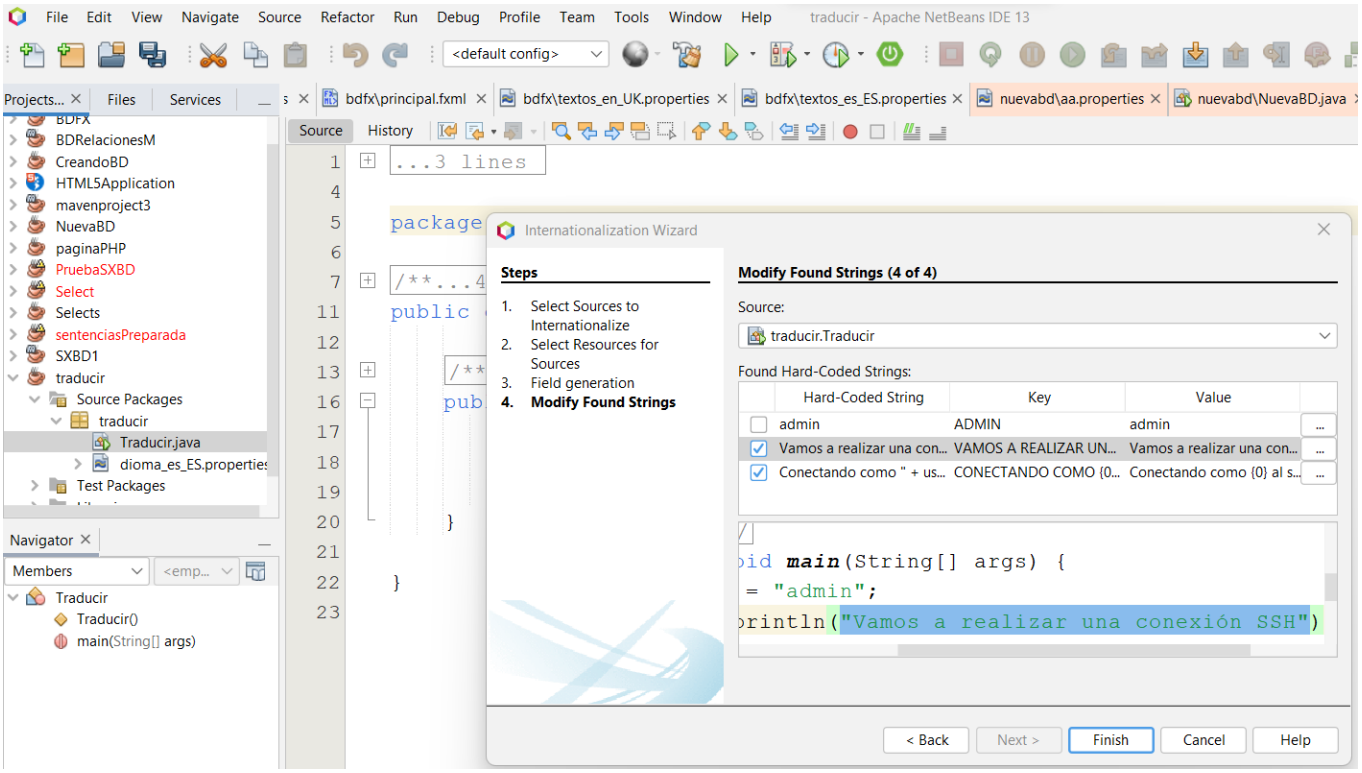
Creamos una variable asociada al fichero de recursos:

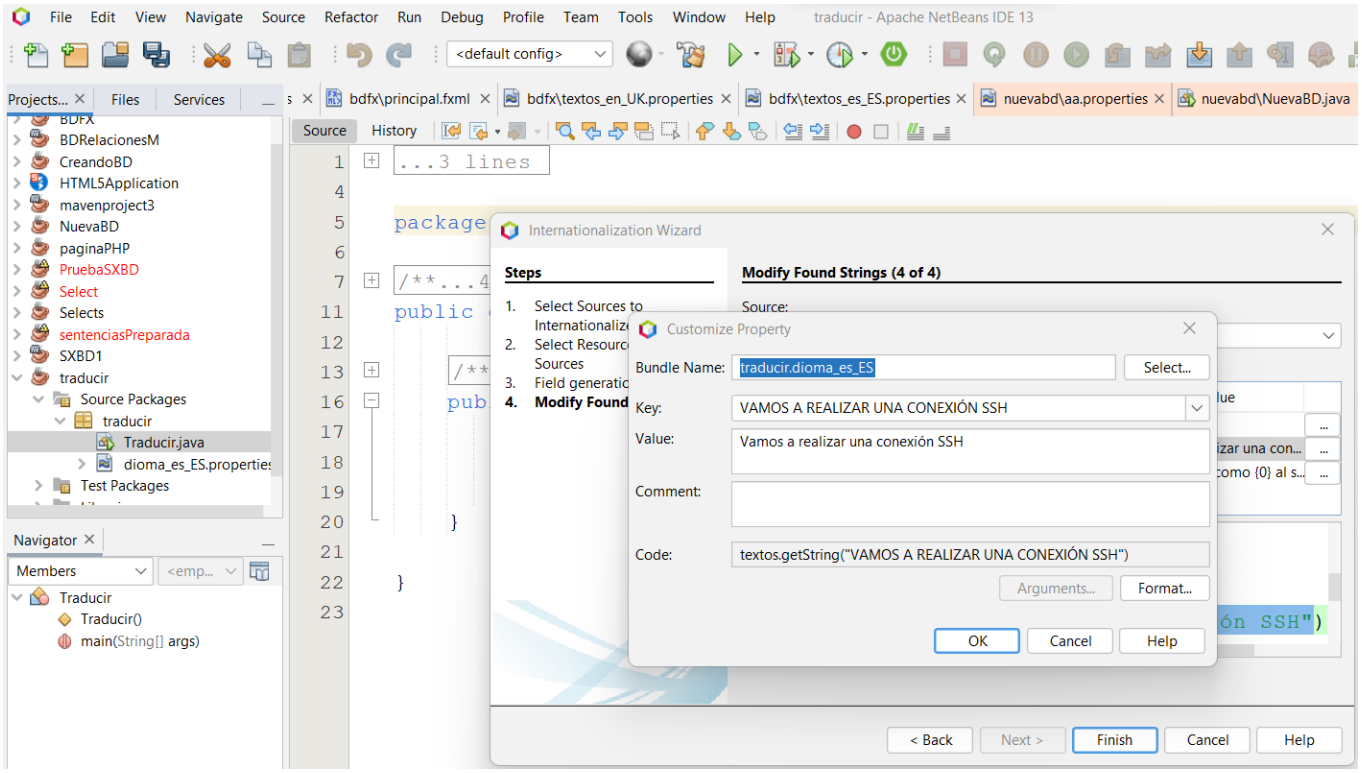


Y se nos mostrarán todas las posibles cadenas a traducir.

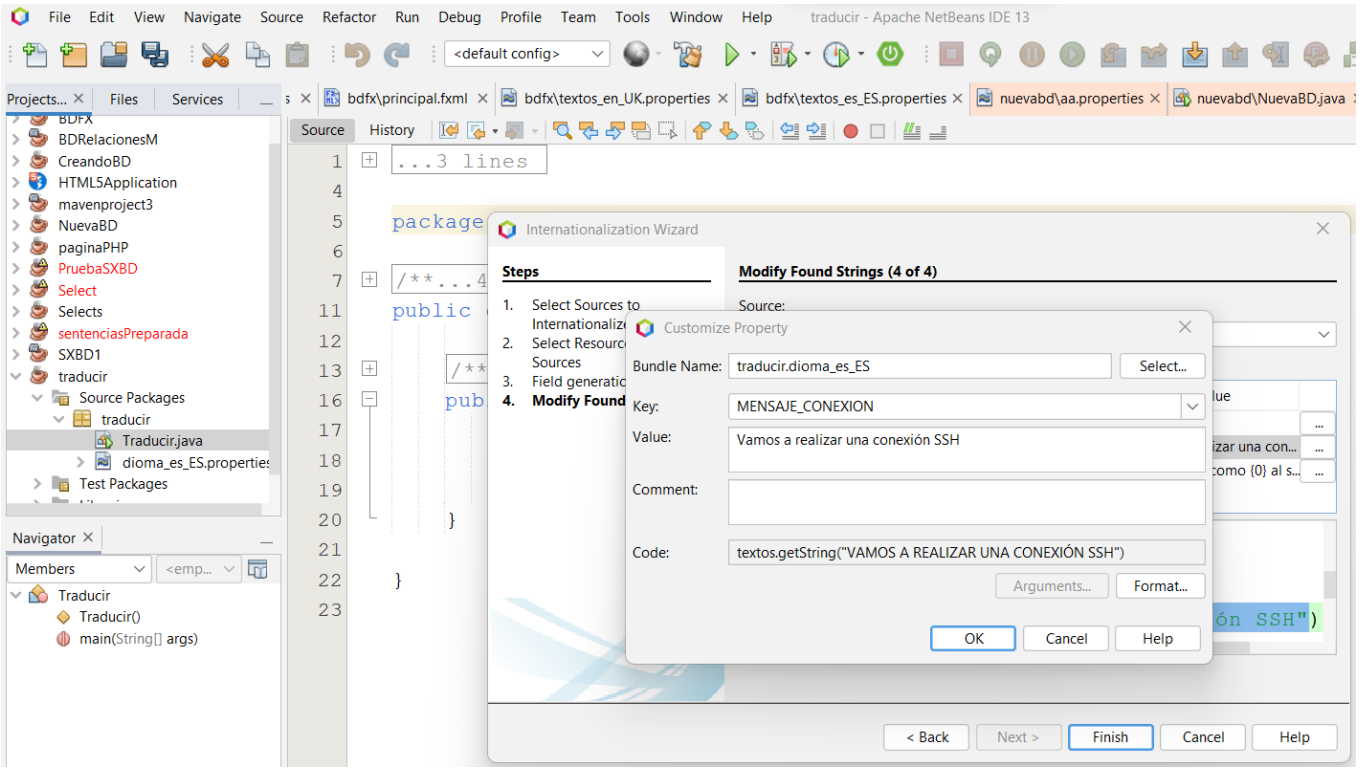


Al ponernos sobre cada una, se muestra el trozo de código donde aparece. Desmarcamos la primera, que se corresponde con el nombre de usuario, que evidentemente será el mismo independientemente del idioma. Y para cada una de las otras dos pulsamos en el botón de los tres puntos situado a su derecha

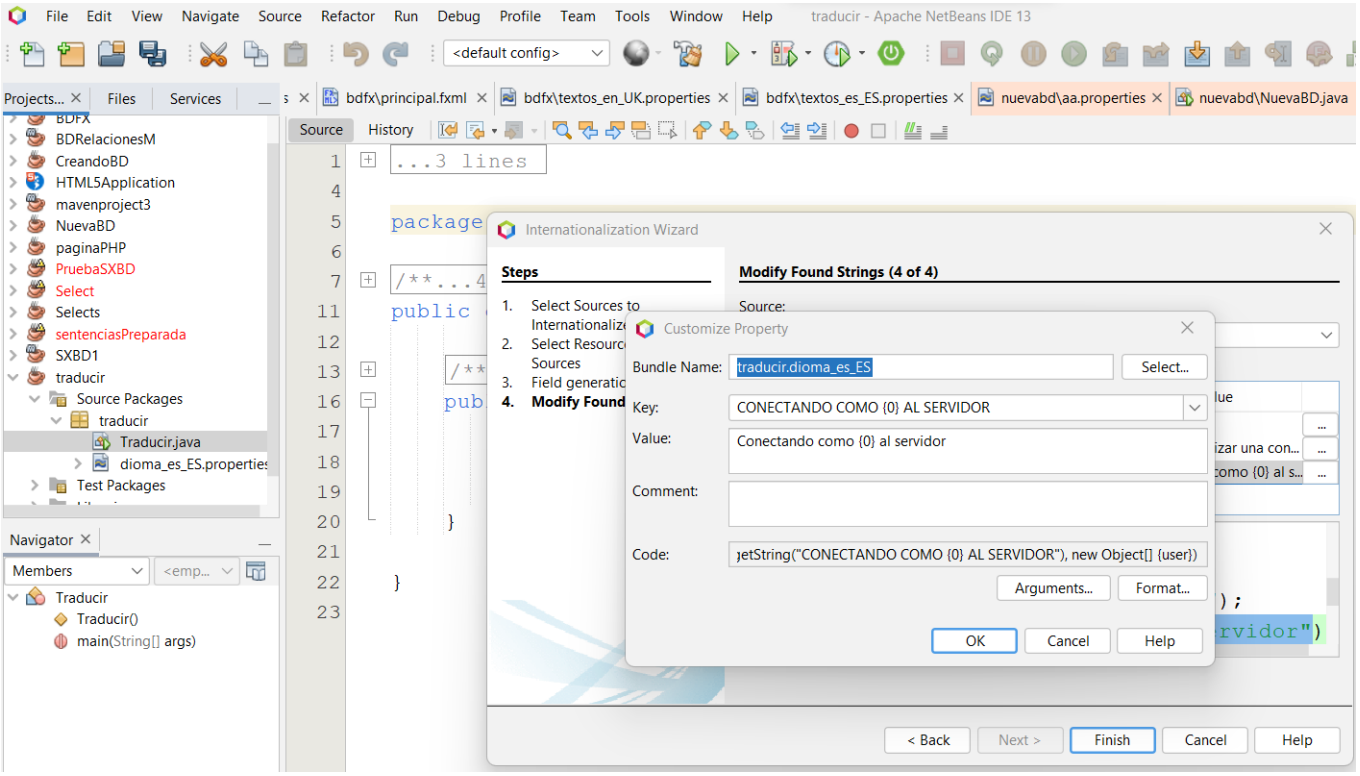




Deberíamos cambiar el campo 'Key'. Por defecto el asistente lo rellena con el contenido del String pasado a mayúsculas. Pero esto no es muy útil, al queda claves demasiado largas y con espacios en blanco, etc. Deberíamos poner claves cortas pero identificativas, y en CAMEL_CASE. Las cambiamos:

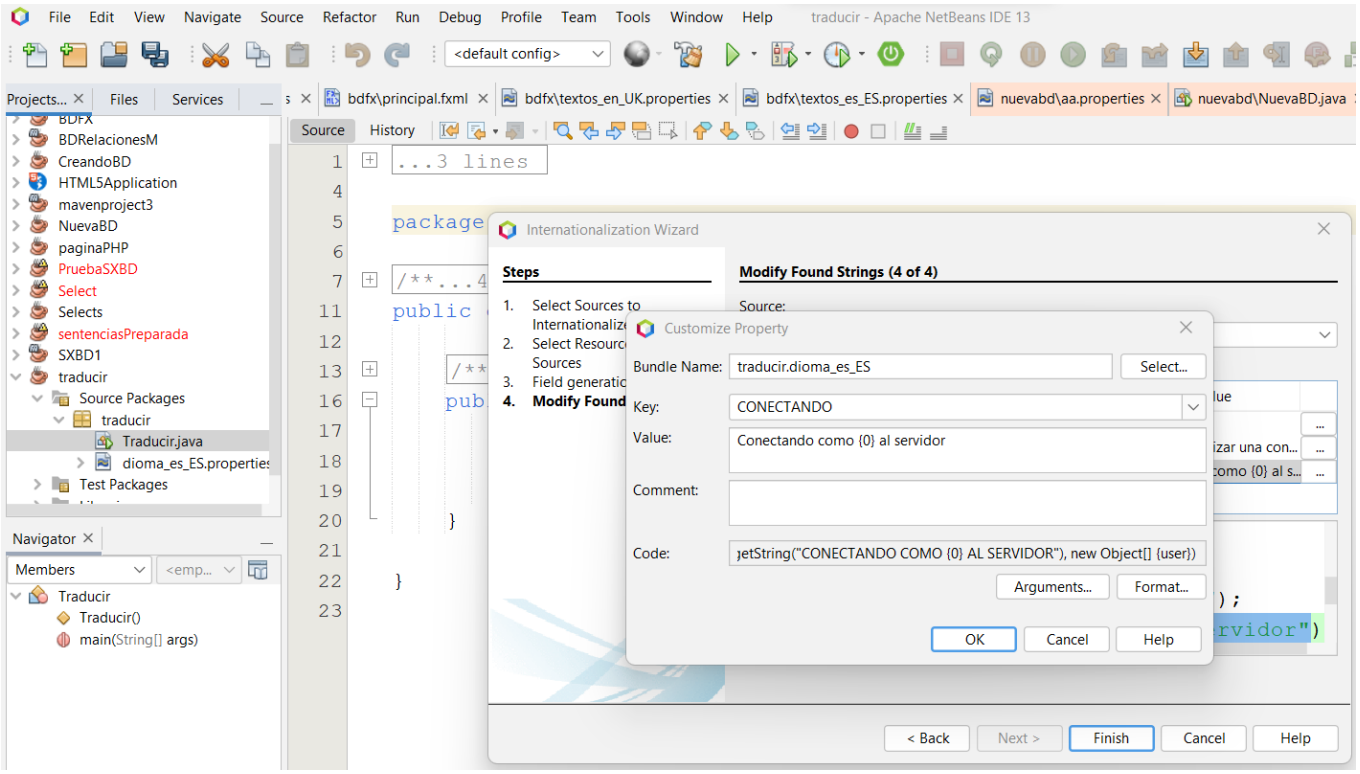


Procedemos igual con la tercera cadena:

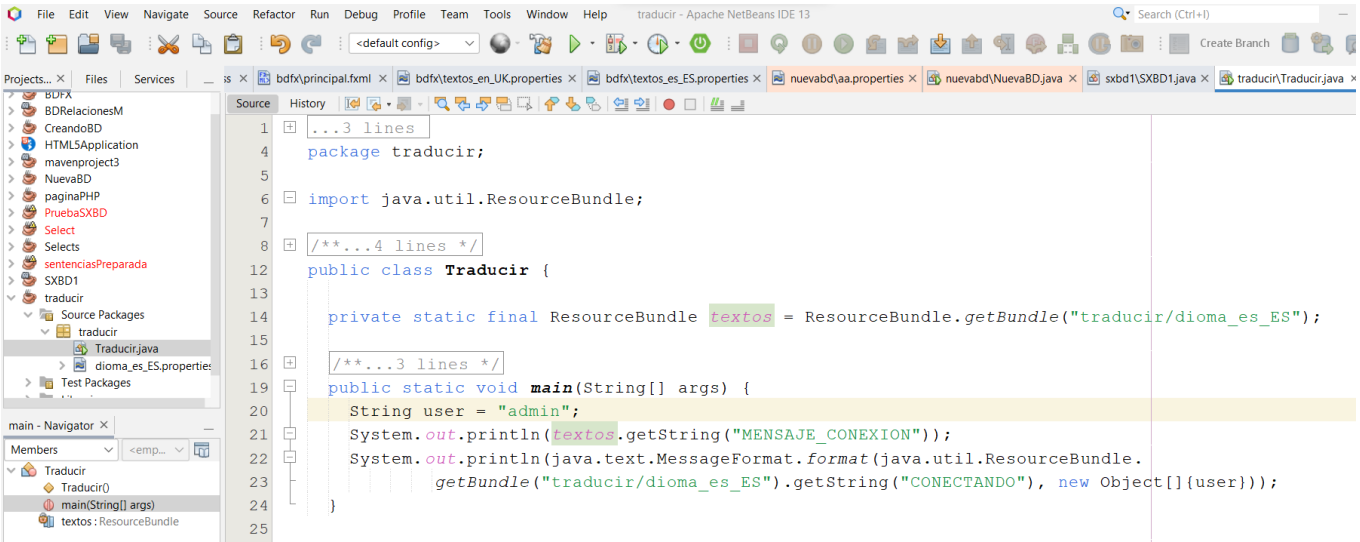


Aquí aparecen los caracteres {0} , esto es un place-holder, lo que indica que en su lugar aparecerá el resultado de evaluar una expresión o el contenido de una variable. En este caso, la variable 'user'. Si hubiera más variables o expresiones concatenadas dentro de la cadena, serán sustituidas por {1}, {2}, y así sucesivamente.

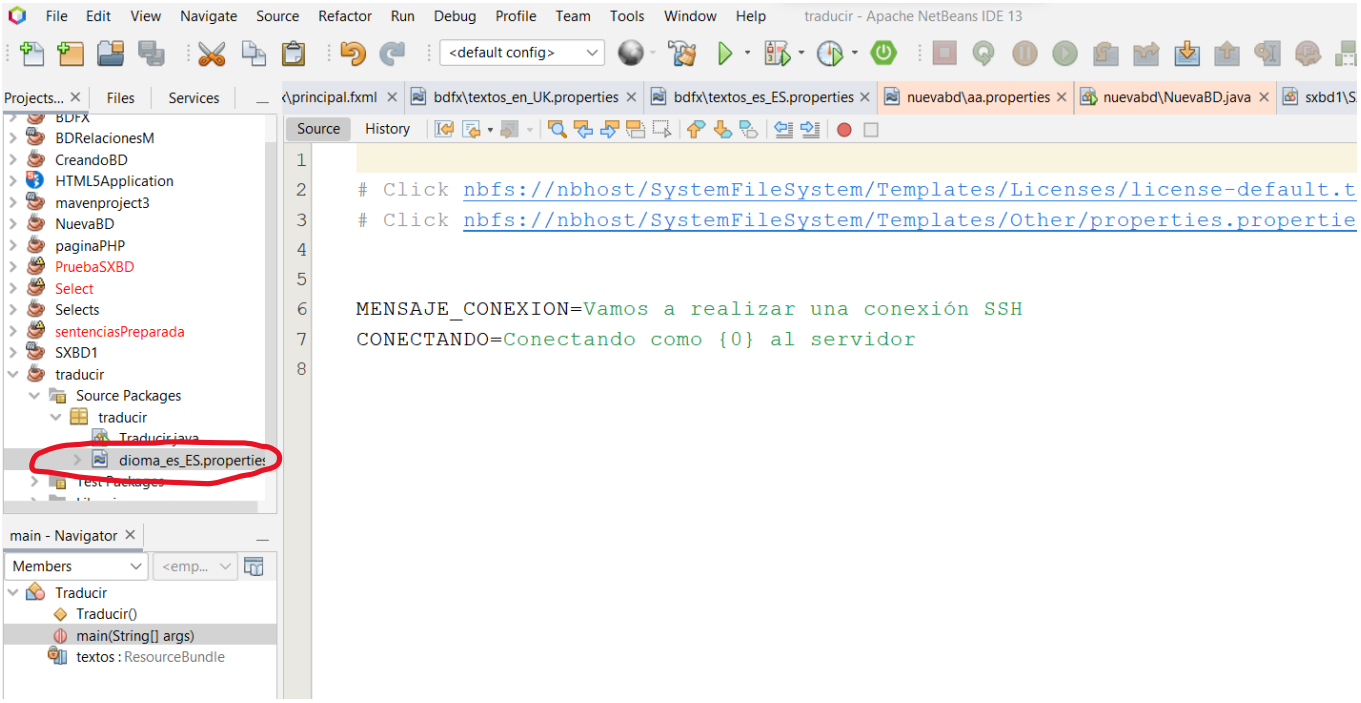
Cambiamos la clava (no necesita place-holder)



Y finalizamos el asistente. El siguiente sería el resultado. Ha modificado el código Java:



Y creado el fichero de traducción en uno de los idiomas (para el resto de los idiomas, copiamos este fichero, dejamos las claves como están y traducimos los textos).



Utilización de controles ChoiceBox (para seleccionar el idioma)

El modo más usual de escoger entre los diferentes idiomas es utilizar ficheros de configuración, que se leen al arrancar la aplicación y normalmente si cambiamos esas opciones será necesario un reinicio de la aplicación para efectuarlas. En esta aplicación vamos a realizar el cambio según se escoja la opción de idioma de una lista desplegable que añadiremos a la interfaz. Esto tienes unas limitaciones:

- Al no guardar la configuración, siempre se arrancará con el mismo idioma
- La barra de título de la ventana no se verá afectada por el cambio.
- La ventana de alerta no se verá afectada por el cambio (de salir lo hace antes de que podamos escoger idioma, y a continuación cerrará la aplicación.
- Para que los textos que pertenecen a la interfaz gráfica (.fxml) sean cambiados, necesitamos reiniciar la ventana, con lo que se perderá lo que en ese momento tengamos introducido en los diferentes campos.

Editando el .fxml o desde SceneBuilder añadimos un controlador ChoiceBox y le colocamos una etiqueta (Label)

Dentro de la clase 'Controlador' método 'initialize' inicializaremos sus valores:

- Añadiéndole los elementos a mostrar (en este caso castellano e inglés)
- Asignando el valor seleccionado a la variable 'idioma' que habíamos definido en la clase 'App'
- Y enganchándole un listener a su propiedad ValueProperty (ejecutará el código cuando haya un cambio en el valor seleccionado del ChoiceBox). De los tres parámetros que se le pasan, solo vamos a hacer uso del último 'p2' que es el valor nuevo seleccionado (`p1' sería el valor previo) para en función del mismo cargar el fichero de recursos apropiado y a continuación reiniciar la interfaz gráfica.

```
@FXML
private void initialize() {
  […]
  comboIdioma.getItems().addAll("Castellano", "Inglés");
  comboIdioma.getSelectionModel().select(App.idioma);
  comboIdioma.valueProperty().addListener((ov, p1, p2) -> {
    App.idioma = p2;
    App.textos = ResourceBundle.getBundle("es/amador/bdfx/textos_"
            + (p2.equals("Castellano") ? "es_ES" : "en_UK"));
    try {
      App.setRoot("principal");
    } catch (IOException ex) {
      Logger.getLogger(Controlador.class.getName()).log(Level.SEVERE, null, ex);
    }
  });
 […]
}
```

Limitar el tiempo en que se muestran los mensajes en la parte inferior de la aplicación

Por último, vamos a hacer que los textos que se muestran en la parte inferior de la ventana desaparezcan después de un tiempo determinado. Lo haremos aplicando un efecto de desvanecimiento o fade out. Estos mensajes los estamos mostrando, cambiando el texto asociado a una Label que hemos situado a la derecha del botón de reseteo. Crearemos un efecto que asociaremos a esta Label y lo aplicaremos cada vez que cambiemos su contenido.

En la clase 'Controlador' creamos una variable para el efecto FadeTransition, indicando la duración en segundos.

```
  private final FadeTransition fadeOut = new FadeTransition(
          Duration.seconds(5)
  );
```

En el método 'initialize' asignamos y configuramos la transición.

```
private void initialize() {
    [...]

    fadeOut.setNode(etMensaje);  // asignamos el efecto a la etiqueta
    fadeOut.setFromValue(1.0);  // partimos de una visibilidad completa 1.0
    fadeOut.setToValue(0.0);  // acabamos con el texto oculto, visibilidad 0.0
  }
```

Y ejecutamos el efecto de desvanecimiento, cada vez que mostramos un mensaje:

```
  void mensaje(boolean error, String mensaje) {
    if (error) {
      etMensaje.setTextFill(Color.RED);
    } else {
      etMensaje.setTextFill(Color.WHITE);
    }
    etMensaje.setText(mensaje);
    fadeOut.playFromStart();  // ejecutamos el efecto
  }
```