# Calidad de los Sistemas Informáticos, 2018-2019
# Práctica 5 – Interfaz de búsqueda

## Objetivo

El objetivo de esta práctica es la ampliación del proyecto creado en las prácticas anteriores para dotarlo de una interfaz de búsqueda que muestre un conjunto de elementos que se ajusten a determinados criterios de localización, a la vez de invocar al formulario de detalle realizado en la práctica anterior.

Para realizar esta práctica se empleará una clase modelo, que hereda de la clase de Java `AbstractTableModel` y que se encarga de poblar automáticamente un componente `JTable`, evitando el trabajo de hacerlo a mano.

## Requisitos previos

Para la realización de esta práctica se requieren los recursos y resultados de las prácticas previas.

## Criterios sintácticos

- `Courier`: código fuente, nombres de archivos, paquetes, entidades, atributos, tablas y campos.
- *Cursiva*: términos en otro idioma.
- Negrita: contenido resaltado.
- **$**Entre dólar**$**: contenido no textual, generalmente a decidir por el desarrollador.

## Instrucciones previas

- En aquellas partes del código en la que se consideren adecuadas precondiciones (generalmente, al inicio de cada método), que deberán indicarse mediante el comentario hasta el último punto de esta práctica, en el que se propone la implementación de las mismas.

```
// TODO: Preconditions
```

- En esta fase sigue requiriéndose avisar al usuario en caso de error o excepción. Para ello, todas las excepciones y errores deben mostrar un mensaje en pantalla mediante el uso de `JOptionPane`, de la forma en la que se muestra, garantizando que el mensaje de error es suficientemente descriptivo como para garantizar cierto grado de fiabilidad de la aplicación.

```
JOptionPane.showMessageDialog(null,
    "mensaje", "Error", JOptionPane.ERROR_MESSAGE);
```

Pablo de la Torre Moreno
Departamento de Ingeniería Informática
11 de noviembre de 2018

# Procedimiento

## 1. Creación del formulario de búsqueda

1. Crear, en el paquete `es.uca.gii.csi18.$equipo$.gui`, un formulario con la clase denominada `Ifr$Entidades$` (ojo, que está en plural), a través de WindowBuilder mediante [New... > Other... > WindowBuilder > Swing Designer > JInternalFrame]. Por ejemplo, `IfrUsers`, `IfrLicencias`, o el que toque.
2. Eliminar el método `main` y su comentario.
3. Con el código de `Ifr$Entidades$.java` visible, pulsar en la pestaña [Design] (generalmente, abajo) para mostrar el diseñador.
    - En [Structure > Components], pulsar sobre `(javax.swing.JInternalFrame)` y, en la sección [Properties] dar un texto a que `title` del estilo "**$**Entidades**$**" (por ejemplo, "Users" o "Licencias"). En la misma sección, poner `closable` a `true` (de lo contrario, no será posible cerrar el formulario) y `resizable` a `true`.
    - En [Structure > Components], pulsar sobre `getContentPane()` y, en la sección [Properties] comprobar que `Layout` tiene el valor `java.awt.BorderLayout` (ponerlo a dicho valor, si no).

### Información de búsqueda

4. En la sección [Palette > Containers], pulsar `JPanel` y luego pulsar sobre el tercio superior del boceto de ventana. En [Properties], poner la propiedad `Layout` de dicho panel a `java.awt.GridLayout`.
5. Colocar en el panel tantas etiquetas (`JLabel`) y cajas de texto (`JTextField`), u otros elementos, como campos de búsqueda se permitan para la clase de mapeo (parámetros del método estático `Select` creado en prácticas anteriores[1]), sin olvidar la notación húngara y demás criterios de sintaxis[2]. Por ejemplo, puede haber un `JLabel` llamado `lblNombre` y un `JTextFields` llamado `txtNombre`.
6. Añadir un botón "Buscar" al final de la lista[3].

### Lista de resultados

7. Colocar un componente `JTable` justo en el centro de del boceto de ventana, y denominarlo `tabResult`.

## 2. Invocación del formulario desde el menú principal

1. En el boceto de la vista de diseño del formulario principal `FrmMain`, sobre el menú [Buscar > **$**Entidad**$**], pulsar con el botón derecho y seleccionar [Add event handler >

[1] Al igual que en la práctica anterior, se puede obviar la fecha.
[2] Si no es posible pinchar sobre el panel donde van los elementos, tras pinchar en el control a agregar, pulsar sobre componente `panel` en la sección [Structure > Components].
[3] Si el diseño no es elegante, una vez terminada y probada la práctica puede cambiarse la propiedad `layout` de `panel` y hacer las modificaciones adecuada, si se desea.

action > actionPerformed]. En el código, incluir algo similar a esto (cambiar los límites en `setBounds`, si no se ajusta bien a la pantalla):

```
Ifr$Entidades$ ifr$Entidades$ = new Ifr$Entidades$();
ifr$Entidades$.setBounds(12, 28, 244, 192);
// El segundo parámetro es para que siempre aparezca delante
frame.getContentPane().add(ifr$Entidades$, 0);
ifr$Entidades$.setVisible(true);
```

2. Ejecutar la aplicación y observar que el formulario de búsqueda aparece cuando es invocado, aunque no tiene funcionalidad.

## 3. Construcción de la clase modelo

1. En el mismo paquete `gui`, crear una clase pública llamada `$Entidades$TableModel` (por ejemplo, `UsuariosTableModel` o `LicenciasTableModel`), que herede de la clase `javax.swing.table.AbstractTableModel`. Esto requerirá:
    a. importar `javax.swing.table.AbstractTableModel`;
    b. incluir los métodos abstractos (Eclipse lo hace automáticamente al pulsar sobre la bombilla con la "X", junto a los números de línea, o si la herencia se incluye en el formulario de creación).
2. Crear dentro de la clase una variable privada de tipo `List<$Entidad$>` o `ArrayList<$Entidad$>`, llamada `_aData`. Dicha variable se usará para contener el resultado de la llamada al método estático `$Entidad$.Select`, creado en las prácticas anteriores, y posteriormente para poblar la tabla de resultados.
3. Crear un constructor que reciba como parámetro de entrada `List<$Entidad$> aData` o `ArrayList<$Entidad$> aData`, y en cuyo código se asigne dicho parámetro a `_aData`.
4. Teniendo en cuenta la variable `_aData` y los campos de la clase `$Entidad$` que se deseen mostrar en la lista de resultados, implementar los métodos requeridos por `$Entidades$TableModel`:
    a. `public int getColumnCount()`: número de columnas que tendrá la tabla.
    b. `public int getRowCount()`: número de filas de la tabla (que será el tamaño del *array* de resultados `_aData`)[4].
    c. `public Object getValueAt(int iRow, int iCol)`: valor que la clase `$Entidades$TableModel` se encargará de devolver el contenido de la casilla `iRow` y `iCol`[5] a mostrar (la tabla lo dibujará automáticamente), y que obtendrá de `_aData`[6].

[4] El modo que Java facilita para obtener el tamaño y un elemento específico de un `ArrayList` es mediante el método `size()` del `ArrayList`.
[5] Es posible que Java haya nombrado los parámetros de otra forma. Si es así, renombrarlo como se indica.
[6] El modo que Java facilita para obtener un registro específico de un `ArrayList` es mediante el método `get(int iIndex)` del `ArrayList`.

## 4. Implementación y prueba de la búsqueda

1. En el formulario de búsqueda, sobre el botón "Buscar" pulsar con el botón derecho y seleccionar [Add event handler > action > actionPerformed]. En el código, incluir algo similar a esto:

   ```
   tabResult.setModel(
       new $Entidades$TableModel($Entidad$.Select($parámetros$));
   ```

   donde $parámetros$ son todos aquéllos necesarios para poder invocar el método $Entidad$.Select, y que deben obtenerse de los campos de búsqueda previamente incluidos en el formulario, justo delante del botón "Buscar".

2. Probar la búsqueda y ver que se cargan los valores encontrados con distintos criterios de búsqueda.

## 5. Invocación al detalle del elemento seleccionado

1. En el formulario de búsqueda, crear una variable privada de tipo Container y llamada pnlParent. Dicha variable contendrá la referencia al formulario principal, para hacer que éste muestre el formulario detalle cuando se pinche dos veces sobre un elemento de la lista. Deberá por tanto modificarse el código del formulario principal para que pase la variable frame (creada en la práctica anterior) como parámetro en el constructor de Ifr$Entidades$., el cual la asignará a pnlParent. Esto generará un error en FrmMain, fácilmente solucionable.
2. En el formulario de búsqueda, pulsar con el botón derecho sobre la tabla table y seleccionar [Add event handler > mouse > mouseClicked]. En el código, incluir lo siguiente (modificando cuando sea necesario las clases propias del proyecto):

   ```
   // Se activa con doble clic sobre una fila
   if (e.getClickCount() == 2) {
     int iRow = ((JTable)e.getSource()).getSelectedRow();

     $Entidad$ $entidad$ =
       (($Entidades$TableModel)tabResult.getModel()).getData(iRow);

     if ($entidad$ != null) {
       Ifr$Entidad$ ifr$Entidad$ = new Ifr$Entidad$($entidad$);
       ifr$Entidad$.setBounds(10, 27, 244, 192);
       pnlParent.add(ifr$Entidad$, 0);
       ifr$Entidad$.setVisible(true);
     }
   }
   ```

   Para arreglar los errores que aparecerán se requiere además realizar los siguientes dos puntos.
3. Crear el método public $Entidad$ getData(int iRow) en $Entidades$TableModel, tal y como se considere que debe ser su función.
4. Cambiar el constructor de Ifr$Entidad$ para que reciba un parámetro $Entidad$ $entidad$ y lo asigne a _$entidad$. Si no es nulo, además deberá rellenar los

campos del formulario de detalle, según se considere. Esto generará un error en `FrmMain` de solución sencilla.

5. Compilar, ejecutar y hacer pruebas. Observar cómo las modificaciones del formulario detalle repercuten en el formulario de búsqueda (suele ser necesario seleccionarlo para que refresque).
6. Estudiar las distintas partes de código, para observar las relaciones entre los elementos.
7. Ajustar código (variables, espacios, tabulaciones) a la sintaxis de la asignatura y los criterios de la asignatura antes de entregar.