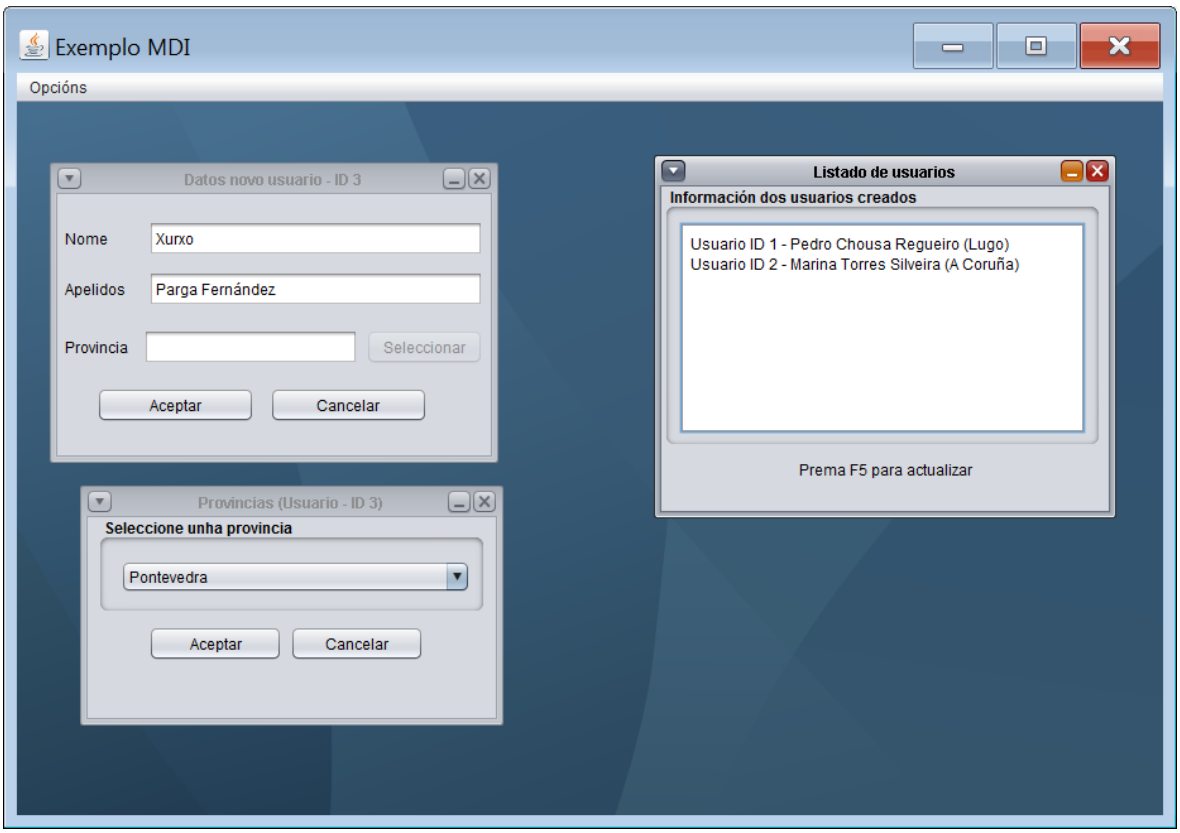
1.1.1.1 Aplicacións multixanela MDI

Cando desenvolvemos aplicacións multixanela empregando un formulario Frame como xanela principal e diálogos para o resto de xanelas, preséntasenos un pequeno inconveniente. A medida que medra o número de xanelas que temos abertas simultaneamente na nosa aplicación, o manexo do noso programa vólvese máis caótico xa que en pantalla teremos tódalas xanelas abertas da nosa aplicación sumadas a tódalas xanelas abertas que pertence a outras aplicacións que esteamos empregando. Un xeito de poñer orde dentro deste maremágnum de xanelas é desenvolver as aplicacións multixanela para que sexan aplicacións de tipo MDI. MDI quere dicir interface de múltiples documentos (multiple document interface). Cando desenvolvemos unha aplicación multixanela de tipo MDI o que facemos é reunir tódalas xanelas que a compoñen dentro dun contedor do cal non poden saírse, de xeito que calquera acción que realicemos queda acoutada a ese contedor, sendo o funcionamento das nosas aplicacións multixanela moito máis ordenado. Tipicamente unha aplicación de tipo MDI está composta por un contedor principal que conterá unha barra de menú na súa parte superior e un escritorio que ocupará o resto do contedor. A barra de menú dispón de distintas opcións que nos permiten abrir as diferentes xanelas que compoñen a nosa aplicación. Las xanelas que abrimos a través do emprego das diferentes opcións presentes na barra de menú sitúanse sobre o escritorio do contedor. O contedor encargarase da xestión das xanelas internas, de xeito que se son minimizadas, o contedor amosará unha barra de ferramentas na cal representaranse graficamente as diferentes xanelas abertas en cada momento de xeito que premendo sobre elas as poidamos restaurar ao seu tamaño orixinal. Ademais, no caso de que maximicemos unha xanela interna, o seu tamaño máximo será o do escritorio do contedor.

Graficamente o aspecto dunha aplicación MDI é o seguinte:



Na imaxe anterior, temos un contedor que na súa parte superior contén unha barra de menú e o resto do contedor é o seu escritorio (zona azul), o cal empregarase para a xestión das sú-

as xanelas internas. Neste caso temos abertas tres xanelas internas que no caso de movelas nunca poderanse saír do contedor, co cal o emprego do programa será máis limpo.

Para acadar este resultado empregamos dous clases de obxectos. Por unha parte un obxecto da clase javax.swing.JDesktopPane que é empregado para xerar o escritorio da aplicación. Por outro lado un obxeto da clase javax.swing.JInternalFrame por cada unha das xanelas que contén a nosa aplicación. Realmente o que faremos será o seguinte: dentro da nosa xanela principal de tipo JFrame colocaremos o JDesktopPane e os JInternalFrames situaranse dentro do JDesktopPane.

Propiedades do contedor JInternalFrame empregadas máis habitualmente e que pódense establecer a través do entorno de programación:

Propiedade	Función
Closable	Mediante esta propiedade indícase se a xanela presenta un botón para pechala.
DefaultCloseOperation	Mediante esta propiedade indícase cal será o comportamento da xanela ao pechala. Pode tomar 3 valores posibles:DISPOSE, HIDE o DO_NOTHING. Con DISPOSE péchase a xanela liberando os recursos consumidos pola mesma. Con HIDE a xanela unicamente será escondida. Con DO_NOTHING non se fará nada.
Iconifiable	Mediante esta propiedade indícase se a xanela presenta un botón para minimizala.
Maximizable	Mediante esta propiedade indícase se a xanela presenta un botón para maximizala.
Resizable	Mediante esta propiedade indícase se a xanela é redimensionable.
Title	Título da xanela.
Visible	Mediante esta propiedade indícase se a xanela é ou non visible.

Eventos do contedor JInternalFrame empregados máis habitualmente e que pódense establecer a través do entorno de programación:

Evento	Lanzamento
InternalFrameClosed	Este evento é disparado cando pechamos a xanela xa sexa ao premer sobre a aspa da xanela ou ao pechala a través de código.

Métodos do contedor JInternalFrame empregados máis habitualmente:

Método	Función
public void dispose()	Este método emprégase para pechar a xanela
public void setSelected(boolean selected)	Mediante este método é posible facer que a xanela pase a ser a xanela activa de entre tódalas xanelas existentes ou tamén pódese empregar para deseleccionar a xanela sobre a que se aplica o método
public void show()	Mediante este método, se a xanela non é visible, visualízase e a sitúase ao fronte de tódalas xanelas

Configuración dunha aplicación multixanela MDI empregando NetBeans

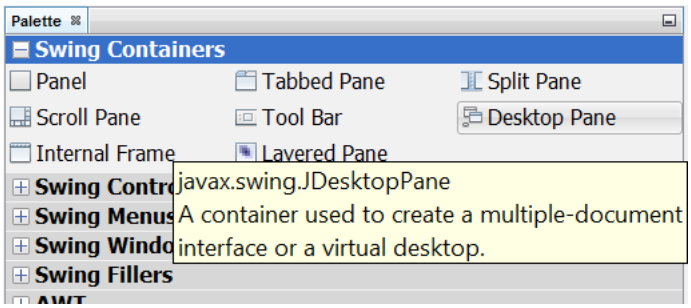
Imos desenvolver unha aplicación parecida á aplicación de xestión de usuarios que creamos para explicar o paso de información entre xanelas, pero a xestión multixanela farémola mediante MDI. Esta aplicación será a seguinte:

A opción de Listado de usuarios abrirá unha xanela na cal amosaranse os diferentes usuarios dados de alta na aplicación. Só poderemos ter aberta unha xanela deste tipo.

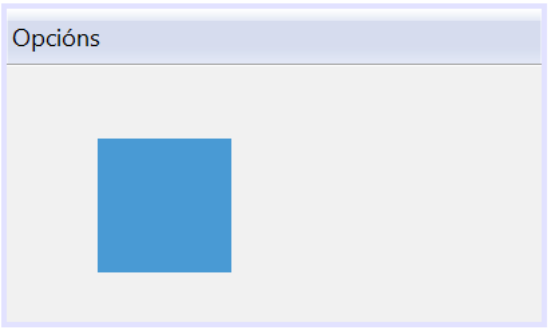
A opción Novo usuario abrirá unha xanela na cal poderemos dar de alta os datos dun novo usuario. Ademais desde esta xanela abriremos unha segunda xanela para elixir dun listado de provincias cal é a provincia do usuario.

A opción Saír finalizará a aplicación.

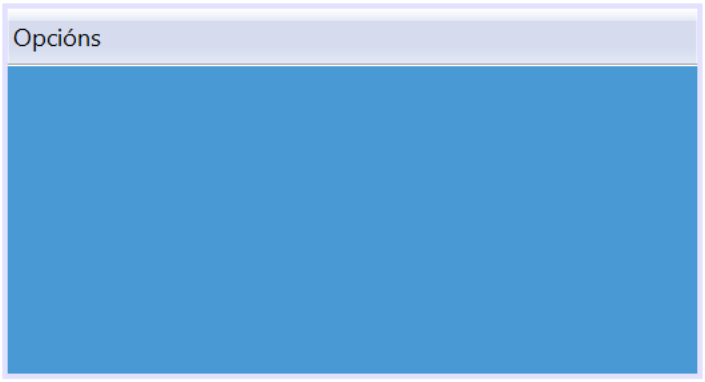
O formulario principal é un formulario de clase JFrame e o menú crearémolo como xa explicamos con anterioridade. A continuación imos ver como crear o escritorio da nosa aplicación MDI. Para elo temos que engadir un compoñente de tipo JDesktopPane no noso formulario, asi que primeiramente seleccionarémolo da paleta de compoñentes do entorno de programación:



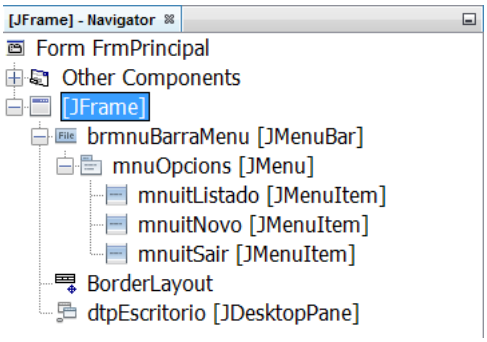
e arrastrámolo ata o formulario:



Unha vez situado dentro do formulario adaptámolo para que teña o aspecto visual que desexamos que teña. O lóxico é que o JDesktopPane ocupe todo o formulario independentemente do tamaño deste último. O xeito máis sinxelo para conseguilo é aplicar un BorderLayout ao formulario:

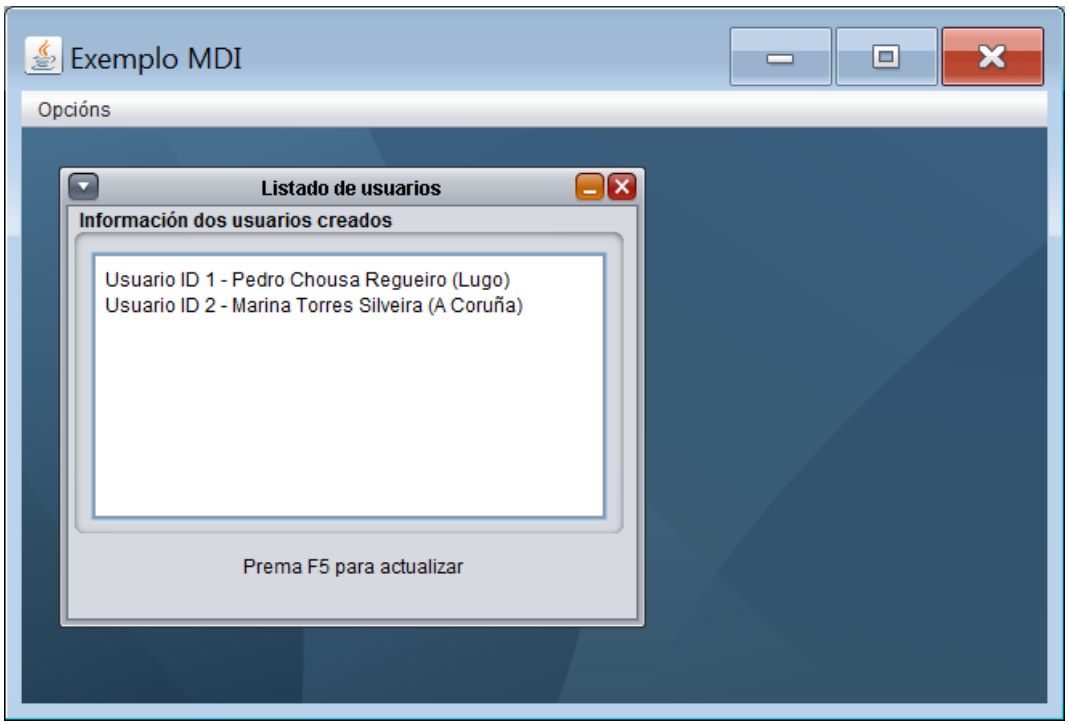


Finalmente o noso formulario principal queda configurado do seguinte xeito:



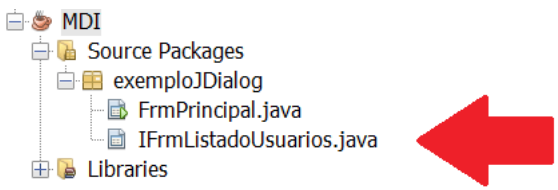
O obxecto dtpEscritorio de clase JDesktopPane é o escritorio sobre o cal situaremos as diferentes xanelas que compoñen a nosa aplicación. A continuación imos ver como colocar as xanelas sobre o escritorio.

Ao premer sobre a opción de menú de Listado de usuarios abrirase unha xanela interna na cal amosarase un resumo dos usuarios que hai dados de alta na aplicación. O aspecto desta xanela interna é o seguinte:

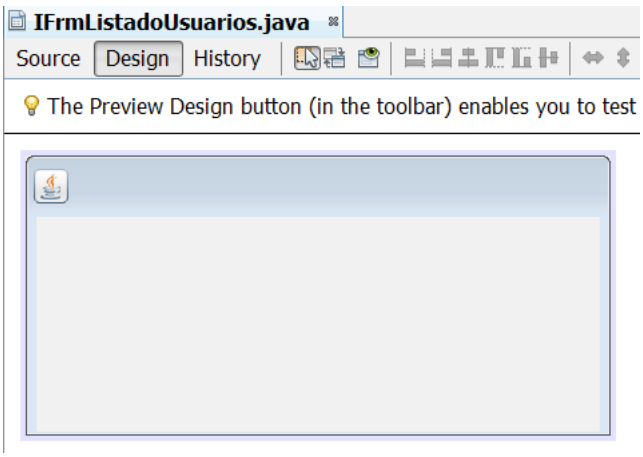


Esta xanela interna é unha xanela de clase JInternalFrame. Para crear este JInternalFrame debemos facer o seguinte: o primeiro é crear unha clase persoalizada que herde de JInternalFrame para definir a nosa xanela interna. O proceso é practicamente idéntico ao explicado para a creación de xanelas de tipo JDialog: prememos co botón dereito do rato sobre o paquete no cal queremos crear a nosa clase persoalizada. Sobre o menú despregado eliximos New e sobre o novo menú despregado eliximos JInternalFrame Form. É posible que a primeira vez que creemos un JInternalFrame, ao premer na opción New non apareza a opción JInternalFrame Form. Nese caso, no mesmo menú deberemos elixir unha opción etiquetada como Other. Ao seleccionala abrirásenos a pantalla que se emprega para crear Clases baseándose nos diferentes modelos que ofrece o entorno de programación. No caso que estamos desenvolvendo elixiremos dentro de Categories a opción Swing GUI Forms e en File Types JInternalFrame Form.

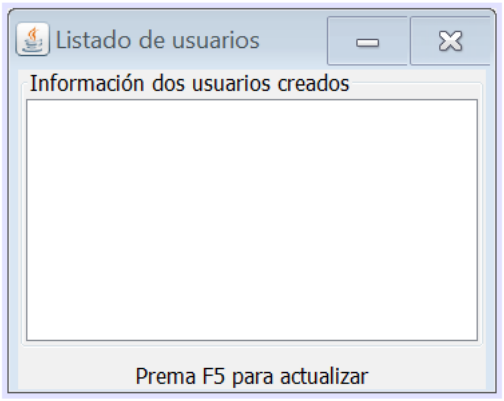
Independentemente de como creemos a xanela interna, ao seleccionar a opción JInternalFrame Form saltará un asistente similar ao de creación dun novo formulario para indicar o nome que lle imos dar á nosa nova clase de tipo JInternalFrame (chamarémola IFrmListadoUsuarios) e opcionalmente para indicar en que paquete queremos gardala. Unha vez que completamos o asistente, na nosa árbore de proxecto teremos unha nova clase que representara ao JInternalFrame que acabamos de crear:



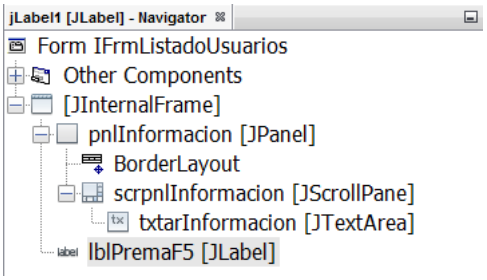
Respecto ao deseño gráfico do JInternalFrame, realízase do mesmo xeito que para un formulario de tipo JFrame. Se seleccionamos a clase que acabamos de crear (IFrmListadoUsuarios), na xanela de deseño do contedor veremos o seguinte:



Como pódese observar é un contedor baleiro. Sobre el estableceremos as súas propiedades, inseriremos os compoñentes, estableceremos os layouts necesarios e escribiremos o código necesario para os diferentes eventos que queiramos xestionar sobre os compoñentes do JInternalFrame. Resumindo, unha vez creado o noso JInternalFrame persoalizado, xestionarémolo tal e como temos feito ata agora cos nosos formularios. Polo tanto, o deseño visual da xanela interna IFrmListadoUsuarios quedará configurado do seguinte xeito:



Na seguinte imaxe amósase a composición gráfica da xanela interna IFrmListadoUsuarios:



A continuación imos ver que é o que temos que facer para que se amose a xanela interna ao premer sobre a opción de menú Listado de usuarios. Para amosar a xanela interna implementamos o actionPerformed da opción de menú Listado de usuarios, sendo o código resultante o seguinte:

```
    private void mnuitListadoActionPerformed(java.awt.event.ActionEvent evt) {
        switch(XestorDeXanelas.podeseAbrirIFrmListadoUsuarios())
        {
            case 0: IFrmListadoUsuarios iFrmListadoUsuarios=new IFrmListadoUsuarios();
                    dtpEscritorio.add(iFrmListadoUsuarios);
                    iFrmListadoUsuarios.show();
                    XestorDeXanelas.abrirIFrmListadoUsuarios();
                break;
            case 1: break;
            case 2: System.out.println("ddd");
                JOptionPane.showMessageDialog(this, "Non hai usuarios na base de datos");
                    return;
        }
    }
```

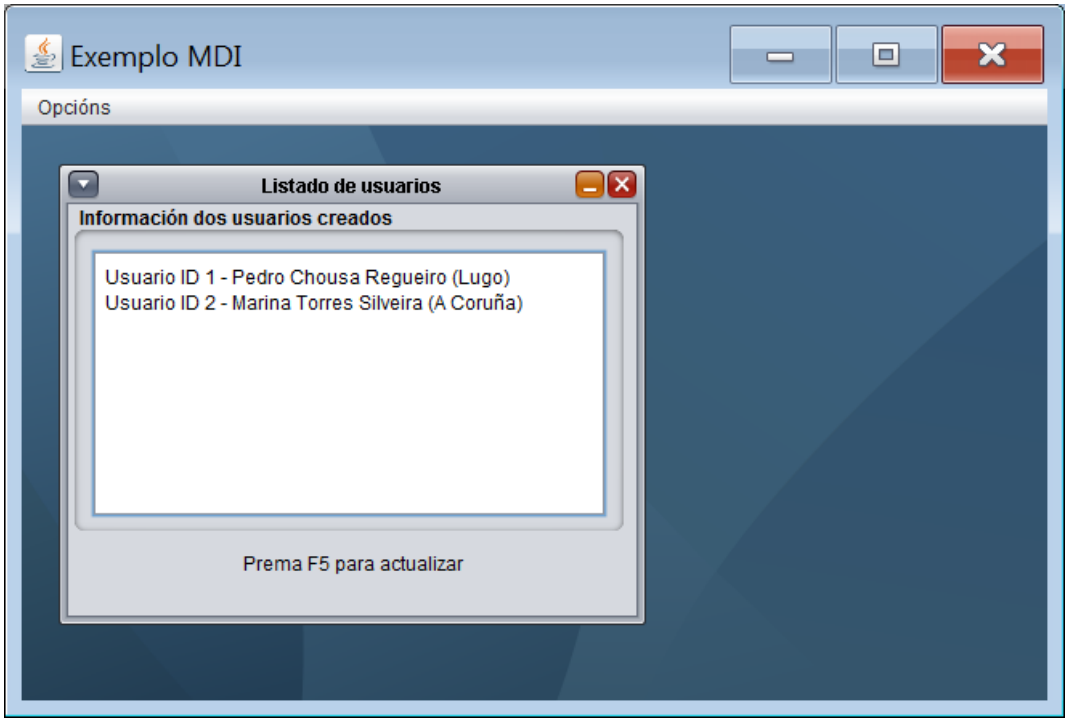
Como dixéramos anteriormente, unicamente podemos ter aberta á vez unha xanela deste tipo, polo tanto realizamos este control a través dun xestor de xanelas (ver clase XestorDeXanelas) dun xeito parecido ao explicado para os diálogos. Nas liñas que acabamos de expoñer, unicamente crearase unha nova xanela no caso de que entremos pola opción 0 do switch, polo tanto centrémonos nesas liñas:

```
            IFrmListadoUsuarios iFrmListadoUsuarios=new IFrmListadoUsuarios();
            dtpEscritorio.add(iFrmListadoUsuarios);
            iFrmListadoUsuarios.show();
            XestorDeXanelas.abrirIFrmListadoUsuarios();
```

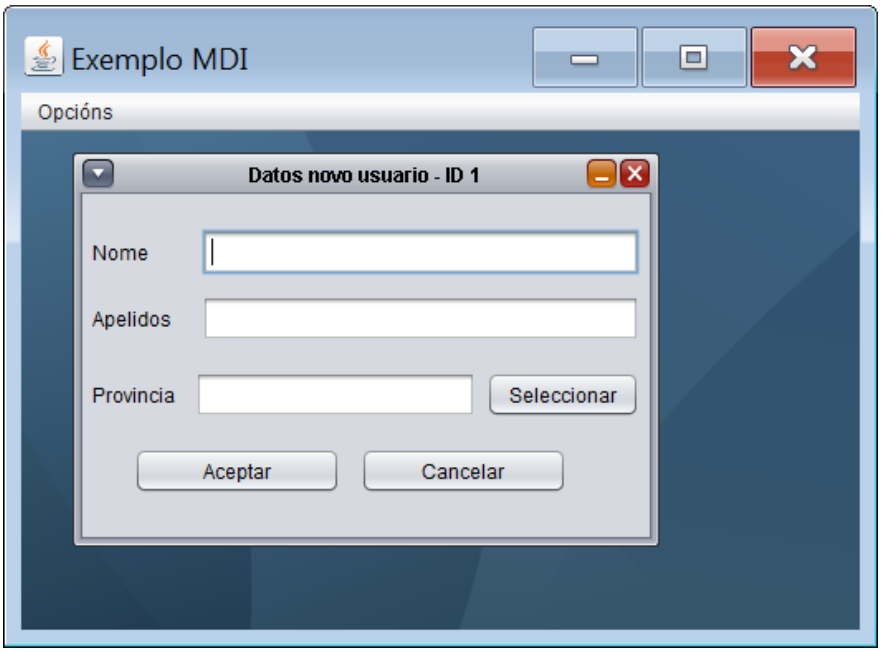
Na primeira liña, o que facemos é instanciar un obxecto da clase IFrmListadoUsuarios empregando o seu construtor. A continuación engadimos o obxecto que acabamos de crear ao noso escritorio mediante o método add do JDesktopPane. Neste momento a xanela interna está creada e situada sobre o JDesktopPane, pero non se visualizará (a menos que teñamos activada a súa propiedade visible con anterioridade). Por ultimo, para que a xanela interna sexa visible empregaremos sobre a xanela interna que acabamos de crear o seu método show o cal fará que sexa visualizable dentro do escritorio da aplicación MDI.

Se executamos a aplicación e pulsamos sobre a opción de menú Listado de usuarios este

será o resultado:

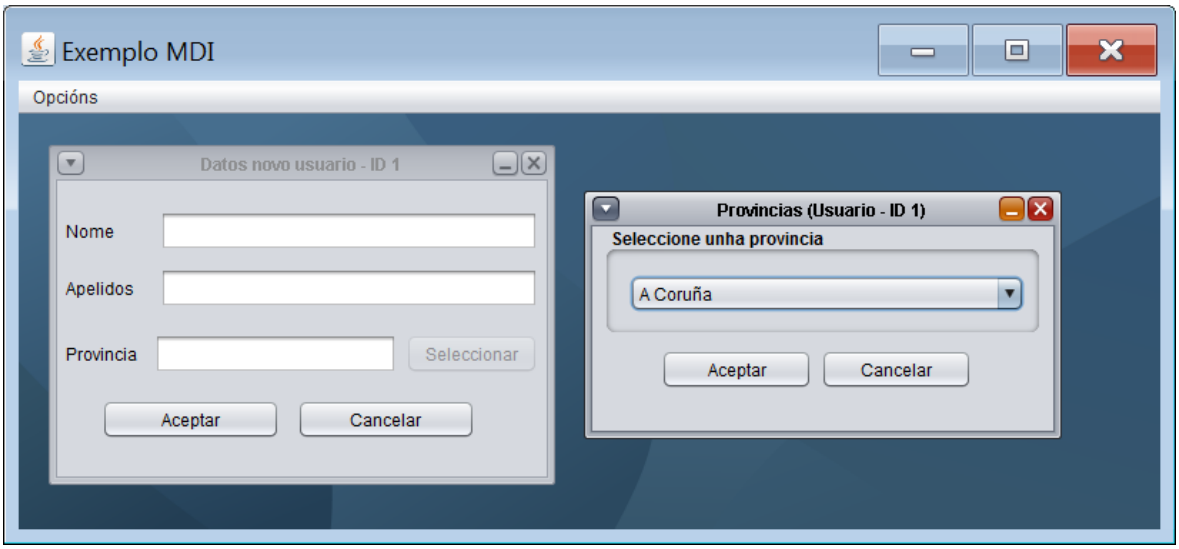


A opción de menú Novo usuario abrirá unha xanela interna na cal poderase dar de alta un novo usuario cuxo identificador será o amosado entre paréntese na opción de menú seleccionada. Nel título da xanela interna amosarase o identificador do usuario que imos crear (esta información pásase á xanela interna desde o formulario principal da aplicación). O aspecto da xanela interna será o seguinte:



Na xanela interna aberta daranse de alta os datos do novo usuario, non obstante, para indicar a provincia non é posible escribir na caixa de texto adxunta, senón que hai que seleccionar a provincia dun combo de provincias. Este combo con provincias atópase noutra xanela inter-

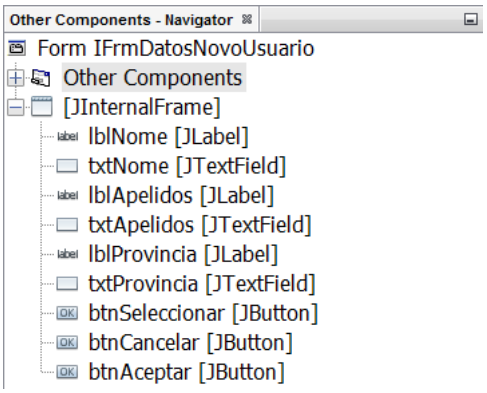
na que se abrirá ao premer sobre o botón seleccionar. O aspecto da xanela interna na cal seleccionaremos a provincia é o seguinte:



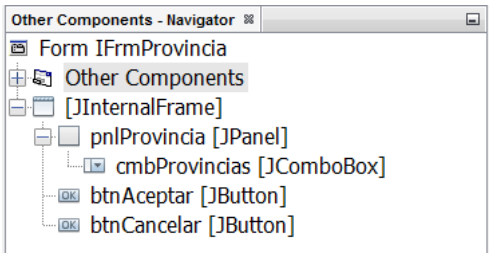
Cando a xanela interna de provincias é aberta desde a xanela interna de Datos novo usuario, esta última pásalle o identificador do usuario que se está creando para que o amose na súa barra de título. Unha vez seleccionada a provincia, ao premer sobre o botón Aceptar da xanela interna de provincias, o valor seleccionado no combo envíase á xanela interna de Datos novo usuario para ser amosado na súa caixa de texto Provincia. Se prememos sobre o botón Aceptar da xanela interna de Datos novo usuario, no caso de que a validación sexa correcta crearase un obxecto de clase Usuario (ver código clase Usuario) que encapsula a información e será enviado á clase AlmacenDeUsuarios (ver código clase AlmacenDeUsuarios). A clase AlmacenDeUsuarios simplemente simula unha pequena base de datos para xestionar a información dos usuarios que imos creando durante a execución do programa (non empregamos unha base de datos real xa que non é parte desta actividade).

As xanelas internas descritas créanse do mesmo xeito que explicamos anteriormente para a xanela interna IFrmListadoUsuarios. Neste caso crearemos as xanelas internas IFrmDatosNovoUsuario (para a alta de usuarios) e IFrmProvincia (para a selección da provincia).

A composición gráfica da xanela interna IFrmDatosNovoUsuario será a seguinte:



mentres que a composición gráfica da xanela interna IFrmProvincia será esta outra:



Imos ver a continuación como creamos a xanela interna IFrmDatosNovoUsuario ao seleccionar a opción de menú Novo usuario. Para elo implementamos o actionPerformed da opción de menú Novo usuario, sendo o código resultante o seguinte:

```
    private void mnuitNovoActionPerformed(java.awt.event.ActionEvent evt) {
        IFrmDatosNovoUsuario iFrmDatosNovoUsuario=new IFrmDatosNovoUsuario(idUsuario);
        dtpEscritorio.add(iFrmDatosNovoUsuario);
        iFrmDatosNovoUsuario.show();
        pintarMenuItemNovo();
    }
```

Na primeira liña, o que facemos é instanciar un obxecto da clase IFrmDatosNovoUsuario empregando o seu construtor. É importante salientar que o construtor ofrecido polo entorno de programación foi modificado para pasar á xanela interna de clase IFrmDatosNovoUsuario o identificador do usuario que vai ser creado (empregamos a técnica de paso de información a través do construtor vista nos diálogos). A continuación engadimos o obxecto que acabamos de crear ao noso escritorio mediante o método add do JDesktopPane. Neste momento a xanela interna está creada e situada sobre o JDesktopPane, pero non se visualizará (a menos que teñamos activada a súa propiedade visible). Por último, para que a xanela interna sexa visible empregamos sobre a xanela interna que acabamos de crear o seu método show, o cal a fará visible dentro do escritorio da aplicación MDI.

Para ver como creamos a xanela de clase IFrmProvincia, temos que estudar o código implementado para o actionPerformed do botón btnSeleccionar:

```
    private void btnSeleccionarActionPerformed(java.awt.event.ActionEvent evt) {
        filla=new IFrmProvincia(idUsuario, this);
        getParent().add(filla);
        filla.show();
        btnSeleccionar.setEnabled(false);
    }
```

O obxecto filla é un atributo a nivel de clase de tipo IFrmProvincia. É empregado polo formulario de clase IFrmDatosNovoUsuario para ter unha referencia sobre a xanela de provincias. Máis adiante veremos para que é necesario o obxecto filla. De momento sigamos analizando o código do actionPerformed do botón btnSeleccionar. Na primeira liña, o que facemos é instanciar un obxecto da clase IFrmProvincia empregando o seu construtor. É importante salientar que o construtor ofrecido polo entorno de programación foi modificado para pasar á xanela interna de clase IFrmProvincia dous informacións. A primeira é o identificador do usuario que vai ser creado. A segunda é unha referencia ao propio formulario que crea a xanela interna de provincias, de xeito que a xanela de provincias teña unha referencia para acceder á xanela que creouna e facer emprego dos seus métodos públicos (para ambas informacións empregamos a técnica de paso de información a través do construtor vista nos diálogos). A continuación engadimos o obxecto que acabamos de crear ao noso escritorio mediante o método add do JDesktopPane. Fixémonos que como desde a xanela interna da clase IFrmDatosNovoUsuario non temos visibilidade sobre o obxecto que referencia ao escritorio da aplicación MDI, o xeito de acadala é empregando o método getParent da clase IFrmDatosNovoUsuario. Este método devolveranos quen é o pai gráfico da xanela interna de clase IFrmDatosNovoUsuario, neste caso o escritorio da aplicación MDI no cal está situada a xanela interna de clase IFrmDatosNovoUsuario. Neste momento a xanela interna está creada e situada sobre o JDesktopPane, pero non se visualizará (a menos que teñamos ac-

tivada a súa propiedade visible). Por último, para que a xanela interna sexa visible empregaremos o seu método show o cal a fará visible dentro do escritorio da aplicación MDI.

Unha vez que na xanela interna de clase IFrmProvincia seleccionamos unha provincia e prememos sobre o botón aceptar, envíase a información do elemento seleccionado á xanela interna de clase IFrmDatosNovoUsuario que creou a xanela interna de clase IFrmProvincia. Para elo, o código implementado no botón Aceptar da xanela interna de clase IFrmProvincia é o seguinte:

```
private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    String provincia=(String)cmbProvincias.getModel().getElementAt(cmbProvincias.getSelectedIndex());
    xanelaChamadora.establecerProvincia(provincia);
    dispose();
}
```

Empregamos o obxecto xanelaChamadora para invocar a un método público da xanela interna de clase IFrmDatosNovoUsuario que creou á actual xanela interna de clase IFrmProvincia (empregamos a técnica vista nos diálogos para invocar métodos públicos de outras xanelas). Este método encargarase de amosar na caixa de texto txtProvincia a provincia seleccionada no combo cmbProvincias. O obxecto xanelaChamadora ten unha referencia á xanela interna de clase IFrmDatosNovoUsuario que creou á actual xanela interna de clase IFrmProvincia. Esta referencia, como viuse anteriormente, foi pasada a través do construtor da xanela interna de clase IFrmProvincia.

Unha vez que enchamos tódolos datos da xanela interna da clase IFrmDatosNovoUsuario e premamos sobre o seu botón Aceptar, valídase a información, créase un obxecto de clase Usuario con ela e envíase á clase AlmacenDeUsuarios. Por ultimo faise un dispose do formulario para pechalo:

```
private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    String nome=txtNome.getText().trim();
    String apelidos=txtApelidos.getText().trim();
    String provincia=txtProvincia.getText().trim();

    if(nome.compareTo("")==0)
    {
        JOptionPane.showMessageDialog(this, "Debe indicar o nome do usuario");
        return;
    }
    if(apelidos.compareTo("")==0)
    {
        JOptionPane.showMessageDialog(this, "Debe indicar os apelidos do usuario");
        return;
    }
    if(provincia.compareTo("")==0)
    {
        JOptionPane.showMessageDialog(this, "Debe indicar a provincia do usuario");
        return;
    }

    Usuario usuario=new Usuario(idUsuario, nome, apelidos, provincia);
    AlmacenDeUsuarios.anhadirUsuario(usuario);
    dispose();

}
```

Non obstante, se prememos sobre o botón Cancelar da xanela interna de clase IFrmDatosNovoUsuario únicamente faise un dispose para pechalo:

```
private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

Nos dous casos executamos un dispose. Esta liña ao igual que ocorre nos diálogos e tamén nos formularios é equivalente a premer no botón de pechar da xanela e o mesmo que esta ultima acción, desencadea o evento internalFrameClosed. Neste caso implementamos este evento do seguinte xeito:

```
private void formInternalFrameClosed(javax.swing.event.InternalFrameEvent evt) {
    if(filla!=null) filla.dispose();
}
```

Tal e como comentamos antes, o obxecto filla é un atributo a nivel de clase de tipo IFrmProvincia empregado polo formulario de clase IFrmDatosNovoUsuario para ter unha referencia sobre a xanela de provincias. Imaxinemos a seguinte situación: abrimos a xanela interna para crear un novo usuario. Prememos no seu botón seleccionar de xeito que tamén

abrimos a xanela interna de provincias. Agora, xa sexa premendo sobre o botón Cancelar da xanela de novo usuario ou sobre o seu botón de pechar, pechamos a xanela interna. Se non implementamos algún código adicional pecharíase a xanela interna de novo usuario pero a súa xanela interna de provincias seguiría estando visualizable sobre o JDesktopPane da aplicación MDI. O que facemos mediante as liñas implementadas a través do evento internaFrameClosed da clase IFrmDatosNovoUsuario é precisamente evitar que isto ocorra.