



# ÍNDICE

---

<b>1.</b>	<b>Elaboración de interfaces de usuario.....</b>	<b>3</b>
1.1	Compoñentes .....	5
1.1.1	Bibliotecas de compoñentes.....	6
1.2	Ferramentas para a elaboración de interfaces.....	7
1.2.1	NetBeans.....	9
1.3	Colectores .....	10
1.3.1	Colectores secundarios.....	11
1.4	Compoñentes da interface .....	11
1.4.1	Engadir e eliminar compoñentes da interface .....	12
1.4.2	Modificación de propiedades.....	13
1.4.3	Engadir funcionalidade desde NetBeans .....	14
1.4.4	Localización e aliñamento dos compoñentes.....	14
1.4.5	Enlace de compoñentes a orixes de datos .....	15
1.4.5.1	Formularios mestre-detalle .....	15
1.5	Asociación de accións a eventos .....	15
1.6	Diálogos modais e non modais .....	15
1.6.1	Diálogos non modais.....	16
1.7	Edición de código xerado por ferramentas de deseño. ....	17
1.8	Clases, propiedades, métodos.....	18
1.8.1	Clases .....	18
1.8.2	Métodos.....	20
1.9	Eventos, escoitadores.....	21
1.9.1	Escoitadores.....	22
<b>2.</b>	<b>Tarefas.....</b>	<b>24</b>
2.1	Evolución das interfaces de usuario.....	24
2.2	Colectores e compoñentes .....	25
2.3	Codificado.....	26
<b>Anexos.....</b>	<b>27</b>	
2.4	Nomenclatura .....	27
2.5	Acceso a datos .....	29
<b>3.</b>	<b>Recursos.....</b>	<b>32</b>

# 1. Elaboración de interfaces de usuario

A interface de usuario é o elemento dunha aplicación informática que permite ao usuario comunicarse con ela.

O desenvolvemento de aplicacións hoxe día non pode ser entendido sen dedicar unha porcentaxe bastante importante de tempo a planificar, analizar, deseñar, implementar e probar os seus interfaces de usuario xa que son o medio fundamental polo cal o usuario pode comunicarse coa aplicación e realizar as operacións para as que foi deseñada. Existen diferentes tipos de interfaces de usuario, entre as máis coñecidas:

**Textuais CLI.** A comunicación prodúcese por medio da inserción de ordes escritas nun intérprete de ordes.

**Gráficas GUI.** A interface consta dun conxunto de elementos visuais como iconas ou menús cos que se interacciona, normalmente, mediante un elemento apuntador (o rato por exemplo). Teñen o mérito de ter popularizado o mundo da informática para usuarios noveis.

**Táctiles TUI.** A comunicación prodúcese mediante a través dun dispositivo táctil, xeralmente unha pantalla que pode reaccionar ante a presión táctil de elementos apuntadores ou dos dedos. É habitual combinar con respostas hápticas. Usos en múltiples dispositivos dende móbiles a terminais de puntos de venda.

**Interfaces de voz (VUI - Voice User Interface).** Estas interfaces permiten a interacción co sistema mediante comandos de voz. A súa popularidade está en auge grazas a asistentes virtuais como Siri, Alexa, e Google Assistant. Son útiles cando as mans están ocupadas ou o dispositivo non ten pantalla, como en altavoces intelixentes.

**Interfaces baseadas en xestos (Gesture-Based Interface).** A interacción realízase mediante movementos do corpo ou das mans, capturados por cámaras ou sensores de movemento, como as que atopamos en consolas de videoxogos como a Xbox Kinect ou en sistemas de realidade aumentada. Exemplo: Leap Motion, Microsoft HoloLens.

*Ao longo desta unidade centrarémonos na creación de interfaces gráficas. Veremos que unha interface gráfica está formada por un conxunto de xanelas, chamadas formularios, e que dentro deles podemos colocar diferentes elementos visuais, que se denominan controis ou compoñentes cos que ao interactuar damos ordes ou podemos recibir información.*



Que interfaces de usuario botas en falta na lista anterior?

BCI Brain-Computer Interface, AR, VR?



Historia das interfaces gráficas de usuario

Crash course (13 min) <https://www.youtube.com/watch?v=XIGSJshYb90>

The history of the graphic user interface  
<https://www.youtube.com/watch?v=U1Oy4X5Ni8Y>



### Vo 655 de Iran Air



“The Pentagon issues a statement that the Vincennes has downed an Iranian F-14 fighter. The Navy dismisses Iranian reports that an unarmed civilian jetliner has been shot down. Minutes later, the Navy issues a correction; they have indeed shot down, by mistake, Iran Air 655. 290 people were aboard. There were no survivors” (Lee, 1992, p. 233)

In his final report on the incident, Fogarty concluded that Aegis had provided accurate information. The crew had somehow misinterpreted the data. ... The operators had fallen victim to the one major flaw of the Aegis ... “seemingly trivial design decision.”

“The radar image of the Airbus on one of the giant computer screens displayed the airplane’s position and heading. But critical information about the plane’s altitude was omitted, and instead displayed on different consoles. Correlating the two pieces of information proved difficult at best.”

Recommended human engineering so that Commanding Officer can separate crucial information from other data, and vital data is displayed so they don’t have to shift attention back and forth between displays (Lee, 1992, pp. 234-5)

### Iran Air 655

En 1988, a fragata USS Vincennes disparou un misil a un Airbus A-300, de Iran Air, con 290 persoas.

O sistema de armamento Aegis, a bordo do Vincennes, tiña un software sofisticado para identificar e monitorizar potenciais brancos. Sen embargo, a pantalla principal non amosaba a información acerca da altitude dos potenciais brancos, tendo que lela noutras consolas.

O Airbus foi interpretado como un caza F-14, debido a que non se leu correctamente a altura. Irónicamente, unha nave escolta cun equipamiento máis vello, foi capaz de interpretar a altitude da nave correctamente, pero non poido intervir a tempo.

## 1.1 Componentes

Unha interface gráfica compórtase como un todo para proporcionar un servizo ao usuario permitindo que este realice peticións, e mostrando o resultado das accións realizadas pola aplicación. Con todo componse dunha serie de *elementos gráficos atómicos* que ten as súas propias características e funcións e que se combinan para formar a interface. A estes elementos chámaselles compoñentes ou controis.

Algúns dos compoñentes máis típicos son:

**Etiquetas:** Permiten situar un texto na interface. Non son interactivos e poden utilizarse para escribir texto en varias liñas.

**Campos de texto:** cadros dunha soa liña nos que podemos escribir algún dato.

**Áreas de texto:** cadros de varias liñas nos que podemos escribir parágrafos.

**Botóns:** áreas rectangulares que se poden pulsar para levar a cabo algunha acción.

**Botóns de radio:** botóns circulares que se presentan agrupados para realizar unha selección dun único elemento entre eles. Úsanse para preguntar un dato dentro dun conxunto. O botón marcado represéntase mediante un círculo.

**Cadros de verificación:** botóns en forma de rectángulo. Úsanse para marcar unha opción. Cando está marcada aparece cun tic dentro dela. Pódense seleccionar varios nun conxunto.

**Imaxes:** úsanse para engadir información gráfica á interface.

**Password:** é un cadro de texto no que os caracteres aparecen ocultos. Úsase para escribir contrasinais que non deben ser vistas por outros usuarios.

**Listas:** conxunto de datos que se presentan nun cadro entre os que é posible elixir un ou varios.

**Listas despregables:** combinación de cadro de texto e lista, permites escribir un dato ou seleccionalo da lista que aparece oculta e pode ser despregada.

The image shows a web form titled "Evento" (Event) with two main sections: "Evento" and "Congreso" (Congress).

**Evento Section:**

- Fecha Hora:** A date and time input field showing "7/06/15 18:13".
- Tipo:** Radio buttons for "Banquete", "Jornada", and "Congreso". "Congreso" is selected.
- Asistentes previstos:** A text input field showing "10".
- Tipo de Cocina:** A dropdown menu with options: "Buffé", "Buffé vegetariano", and "Carta". "Carta" is selected.
- Máximo asistentes: Banquete 100, Jornada y Congreso 50 asistentes.

**Congreso Section:**

- Jornadas del congreso:** A spinner input field showing "1".
- Se requiere habitación:** A checkbox that is currently unchecked.
- Número de habitaciones:** A text input field showing "1".
- Tipo de habitación:** A dropdown menu with the option "Individual" selected.



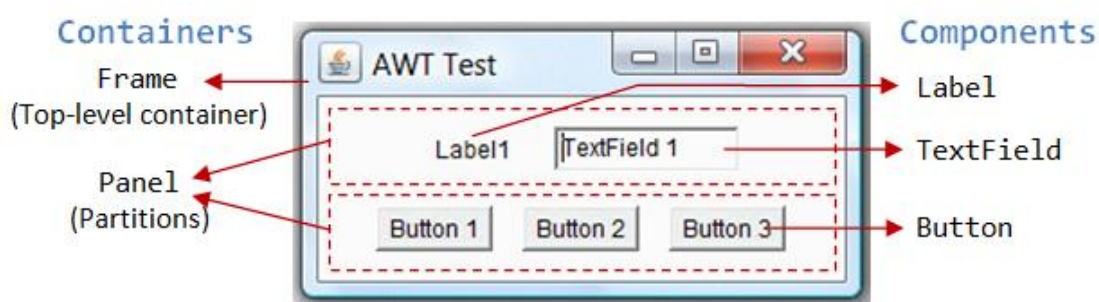
### 1.1.1 Bibliotecas de compoñentes.

Os compoñentes que poden formar parte dunha interface gráfica adóitanse presentar agrupados en bibliotecas coa posibilidade de que o usuario poida xerar os seus propios compoñentes e engadilos ou crear as súas propias bibliotecas. Componse dun conxunto de clases que se poden incluír en proxectos software para crear interfaces gráficas. O uso dunhas bibliotecas ou outras depende de varios factores, un dos máis importantes, por suposto, é a linguaxe de programación ou a contorna de desenvolvemento que se vaia a usar.

Dependendo disto podemos destacar:

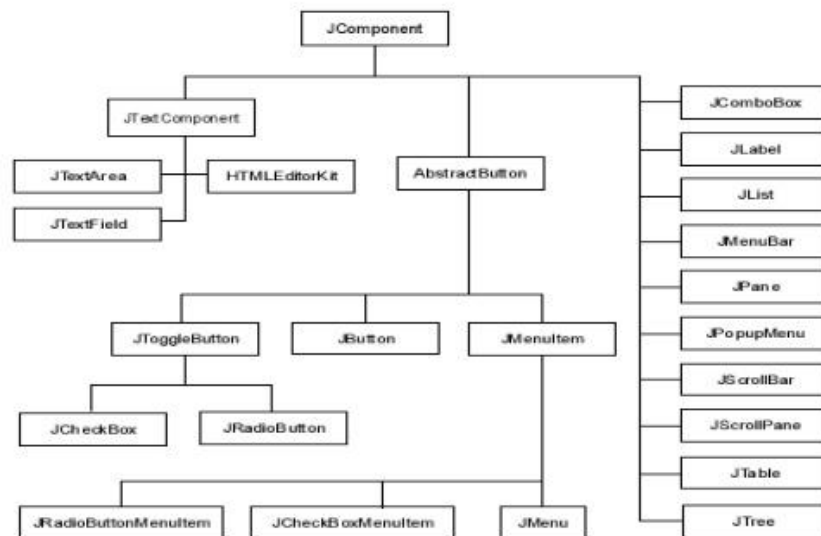
**JAVA Foundation Classes (JFC):** As JFC inclúen as bibliotecas para crear as interfaces gráficas das aplicacións Java e applets de Java.

**AWT:** Primeira biblioteca de Java para a creación de interfaces gráficas. É común a todas as plataformas pero cada unha ten os seus propios compoñentes, escritos en código nativo para eles. Practicamente en desuso.



**Swing:** Xurdida con posterioridade, os seus compoñentes son totalmente multiplataforma porque non teñen nada de código nativo, teñen o seu precursor en AWT, de boto moitos compoñentes swing derivan de AWT, basta con engadir unha J ao principio do nome AWT para ter o nome swing, por exemplo o elemento Button de AWT ten a súa correspondencia swing en JButton aínda que se engadiron gran cantidade de compoñentes novos. É o estándar actual para o desenvolvemento de interfaces gráficas en Java.

# Swing Components Hierarchy



Ademais existen bibliotecas para desenvolvemento grafico en 2D e 3D e para realizar tarefas de arrastrar e soltar (drag and drop).

**Bibliotecas MSDN de Microsoft (C#, ASP, ...):** .NET framework: fai alusión tanto ao compoñente integral que permite a compilación e execución de aplicacións e webs como á propia biblioteca de compoñentes que permite a súa creación. Para o desenvolvemento de interfaces gráficas a biblioteca inclúe ADO.NET, ASP.NET, formularios Windows Forms e a WPF ( Windows Presentation Fundation).



Para coñecer un pouco máis sobre as bibliotecas .NET Framework, en xeral e para o desenvolvemento de interfaces gráficas podes acceder á páxina web que ten MSDN sobre o tema no seguinte enlace.

[MSDN. Biblioteca de clases de .NET Framework.](#)

**Bibliotecas baseadas en XML:** Tamén existen bibliotecas implementadas en linguaxes intermedias baseados en tecnoloxías XML (que se verán na seguinte unidade temática). Normalmente dispoñen de mecanismos para elaborar as interfaces e traducilas a diferentes linguaxes de programación, para despois ser integradas na aplicación final.

Cabe destacar as librarías QT que na actualidade pertencen á compañía Nokia.

## 1.2 Ferramentas para a elaboración de interfaces

Coas bibliotecas tes a base para crear as túas aplicacións, están compostas de código que

podes escribir sen máis, aínda que o habitual é valerse dalgunha contorna integrada de desenvolvemento de software que facilite este labor mediante algún sistema de xanelas, no que se inclúen diferentes rexións rectangulares, chamadas "xanelas" cuxa parte central é un conxunto de ferramentas (toolkit). A este tipo de ferramentas chámase IDE (Integrated Development Environment) e prové de procedementos que permiten incluír os compoñentes das bibliotecas de compoñentes gráficos e engadilos dun modo sinxelo e intuitivo na aplicación que estas a desenvolver.

Tamén serven como medio de entrada das accións do usuario.

A continuación lístanse varios dos IDE máis utilizados na actualidade:

**Microsoft Visual Studio:** É unha contorna para o desenvolvemento de aplicacións en contornas de escritorio, web e dispositivos móbiles coa biblioteca .NET framework de Microsoft. Permite o uso de diferentes linguaxes de programación como C++, C# ou ASP. Na actualidade pódense crear aplicacións para Windows 7, aplicacións web e RIA (Rich Internet Applications).



**NetBeans:** IDE distribuído por Oracle baixo licenza GNU GPL. Está desenvolvido en Java, aínda que permite crear aplicacións en diferentes linguaxes, Java, C++, PHP, Ajax, Python e outras.

**Eclipse:** IDE creado inicialmente por IBM agora é desenvolvido pola fundación Eclipse. Distribúese baixo a Licenza Pública Eclipse, que, aínda que libre, é incompatible con con a licenza GNU GPL. Ten como característica máis destacable a súa lixeireza xa que se basea en módulos, partindo dunha base moi sinxela cun editor de texto con resaltado de sintaxe, compilador, un módulo de probas unitarias con JUnit, control de versións con CVS, integración con Ant, asistentes (wizards) para creación de proxectos, clases, tests, etc. e refactorización. Si precísanse funcionalidades máis aló, estas inclúense como módulos á parte que van completando o IDE.

**JDeveloper:** É unha contorna de desenvolvemento integrado desenvolvido por Oracle Corporation para as linguaxes Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML e outros. As primeiras versións de 1998 estaban baseadas na contorna JBuilder de Borland, pero desde a versión 9i de 2001 está baseado en Java, non estando xa relacionado co código anterior de JBuilder. Tras a adquisición de Sun Microsystems por parte de Oracle, está cada vez en máis desuso, xa que NetBeans ofrece maiores opcións.



En que compoñente se pódense seleccionar varias opcións á vez?

- Botóns de radio.
- Texto.
- Botón.
- Cadros de verificación.



## 1.2.1 NetBeans

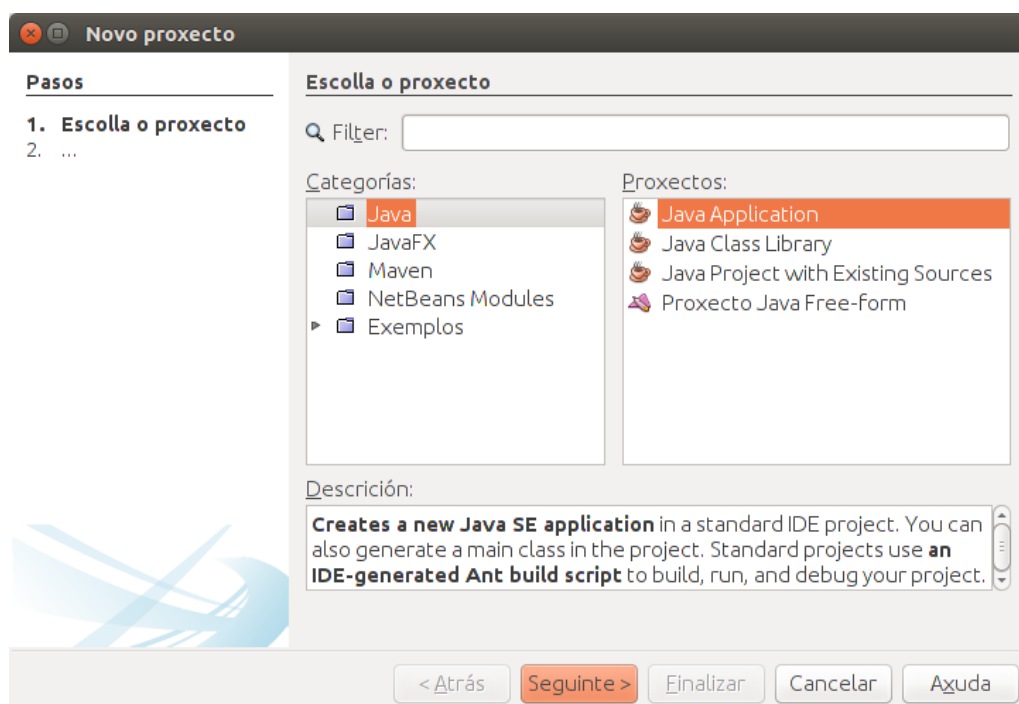
Para desenvolver os contidos desta unidade propómosche NetBeans como entrono de desenvolvemento integrado utilizando a biblioteca swing para a xeración de interfaces. Seleccionouse esta contorna por varios motivos, en primeiro lugar permite crear aplicacións tanto de escritorio, como web como para dispositivos móbiles e ademais distribúese baixo licenza GNU GPL. É multiplataforma e inclúe varias linguaxes de programación.



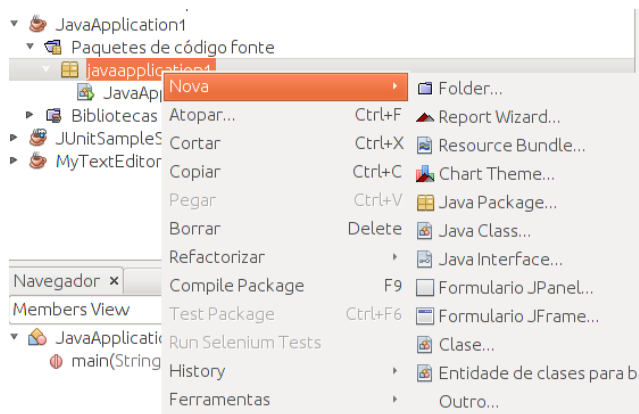
Unha das súas principais vantaxes é que resolve por si mesmo o tema da colocación dos compoñentes nunha interface gráfica, aspecto de certa complexidade á hora de programar, de forma que o desenvolvedor ou desenvolvedora só ten que colocar os controis usando o rato e o IDE encárgase de programalo.

Tamén permite a inclusión de compoñentes creados por outros que completan a paleta swing/AWT.

Unha vez que teñas a contorna instalada e funcionando debes comezar por crear un proxecto que poida utilizar as clases incluídas na libraría swing, para iso só debes seleccionar como tipo de proxecto *Java Application* dentro da categoría Java. A continuación enche os datos principais do proxecto, como o seu nome, a localización dos arquivos en disco e se é unha aplicación básica ou de base de datos (de momento elixiremos a primeira opción).



Despois debes engadir un formulario JFrame para poder comezar a traballar.



A continuación tes unha presentación dos principais elementos desta interface para que te vaías familiarizando con eles.



Outra opción que teñen os desenvolvedores de Java é usar JavaFX en vez de Swing.

<https://openjfx.io/openjfx-docs/#gradle>



## 1.3 Colectores

Un formulario é unha xanela que dispón de tres botóns para minimizarse, maximizarse ou pecharse, unha barra de título e está delimitado por uns bordos. É a base para crear unha aplicación de escritorio. Sobre el engadiranse controis ou compoñentes, sinxelos como botóns ou caixas de texto ou máis complexos, como barras de menú ou reixas de datos, que lle dan funcionalidade.

Unha aplicación de escritorio de NetBeans componse dunha serie de formularios. Para crear un formulario terás que usar un colector Java que é un compoñentes que permite incluír outros compoñentes e outros colectores que se usarán para distribuír os controis. Por iso dise que os colectores forman unha **estrutura xerárquica**.

Un formulario está formado por un colector especial que se chama colector **de nivel superior**. Este tipo inclúe un panel de contido (contentpane) que permite engadir outros compoñente, incluídos outros colectores que se utilicen facilitar a distribución de elementos.

Como colector de nivel superior dun formulario podes elixir entre unha **xanela** (JFrame), un **diálogo** (JDialog) ou un **applet** (JApplet), segundo a necesidade. Todos estes compoñentes derivan, na xerarquía de clases de java, de Window que representa unha xanela típica.



**Xanela** (JFrame): é un formulario con título, os botóns para maximizar, minimizar ou pechar e bordo. Aparece unha icona por defecto en forma de cunca de café que podes modificar, e pode conter unha barra

de menú.

**Dialogo** (JDialog): formularios que se adoitan usar para solicitar información ao usuario. O seu principal características é que poden ser modais ou non, unha xanela modal recibe todas as entradas de usuario e impide que se active outra xanela.

**Applet** (JApplet): xanela que executa unha aplicación Java no contexto dunha páxina web.

Nunha aplicación de escritorio adóitase crear unha xanela principal que sexa de tipo JFrame e se necesitamos xanelas secundarias utilizaremos outras xanelas ou diálogos.

Unha vez que coñeces que é un colector e os distintos tipos que hai, debes saber tamén como incluílos na túa aplicación usando NetBeans e como inflúe iso no código.

### 1.3.1 Colectores secundarios

Tamén se interpretan como *diálogos* os seguintes compoñentes:

**Panel de opcións (JOptionPane)**: xera xanelas con botóns para responder cuestións con respostas do tipo si-non, aceptar-cancelar, aceptar, etc.

**Selector de arquivos (JFileChooser)**: permite seleccionar un arquivo do sistema de arquivos do equipo onde se executa a aplicación.

**Selector de cores (JColorChooser)**: permite seleccionar entre un conxunto de cores e devolveo usando o código adecuado.

Podes usar outro tipo de *colectores* para distribuír o resto dos controis que se inclúen na xanela principais, entre os máis habituais tes:

**Paneis (JPanel)**: representa un colector intermedio, cuxa función principal é a de colocar controis.

**Barra de menú (JMenu)**: permite a creación de menús complexos de opcións.

**Barra de ferramentas (JToolBar)**: utilízase para conter iconas de acceso ás opcións da aplicación.

**Pestanas (JTabbedPane)**: tipo particular de panel que permite a distribución de elementos en pestanas ou cartóns.

**Paneis deslizables (JScrollPane)**: tipo especial de panel que permite desprazar os seus contidos de maneira automática.

**Xanelas internas (JInternalFrame)**: xanelas fillas que non poden pasar os límites da xanela pai onde se crearon. Úsanse en aplicacións que tes varios documentos abertos simultaneamente.

**Paneis divididos (JSplitPane)**: permite visualizar dous compoñentes, un a cada lado, asignando espazo dinamicamente a cada un.

## 1.4 Compoñentes da interface

Os compoñentes ou controis gráficos dun formulario son elementos gráficos que se añan nos colectores para formar aplicacións. Utilízanse para mostrar información, como etiquetas ou imaxes, listas (compoñentes pasivos) ou árbores, pero, sobre todo, para solicitar información do usuario, como cadros de texto, botóns, ou listas de selección (compoñentes activos).

Un compoñente recoñécese pola súa **clase**, que define o seu aspecto e funcionalidade e

polo seu **nome** que o identifica dentro da aplicación. Posto que é un obxecto, segundo a clase á que pertenza terá, unha serie de propiedades que poderemos modificar para adaptar o compoñente, por exemplo, o texto mostrado, ou a cor.

A colocación de compoñentes no formulario réxese por unhas regras, denominadas **Layout**, que establecen a orde e a posición na que deben ser mostrados, podendo ser ao redor dos límites do formulario (norte, sur, leste e oeste), en forma de reixa, ou fluídos, un tras outro, nunha ou varias filas.

Cando un compoñente é susceptible de interactuar co usuario xestiónase mediante o que se coñece como manexo de eventos, unha acción sobre o compoñente que debe provocar unha resposta coñécese como **evento**, a xestión da resposta realízase a través dos **manexadores** de eventos, que son unhas funcións específicas que se asocian a un elemento do compoñente denominado **escoitador**, nas que se programa a acción a realizar.

Vexamos todas estas cousas cun pouco máis de detemento.

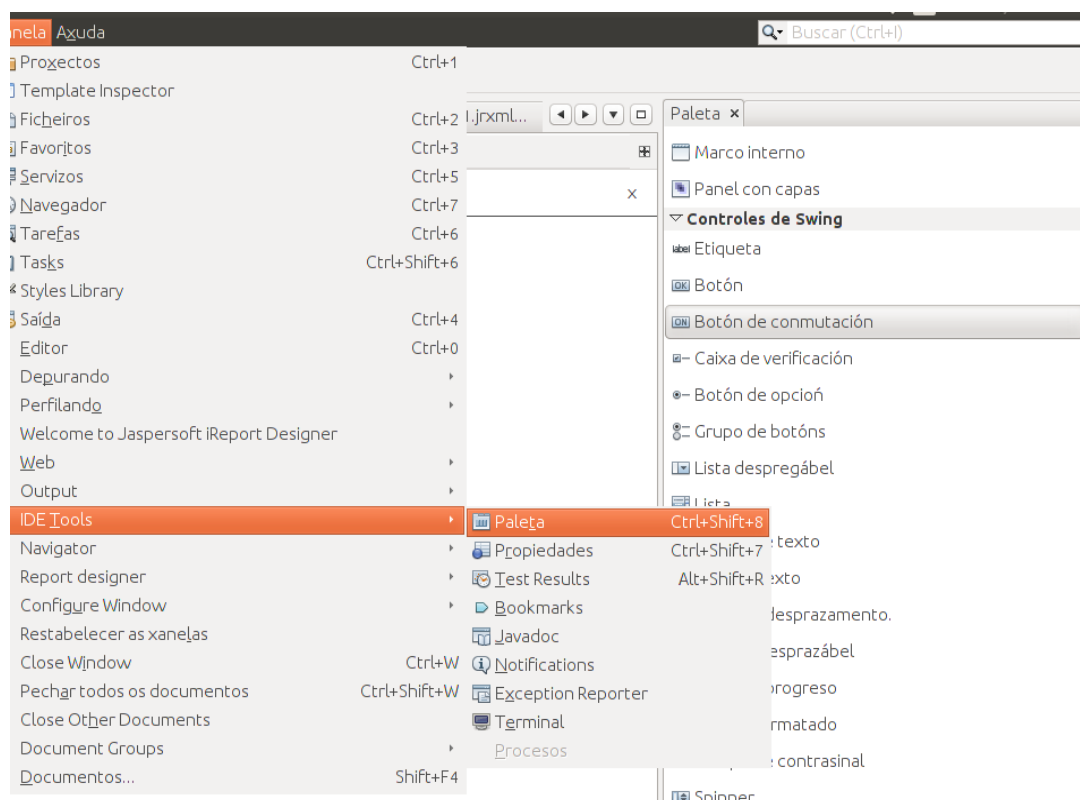


Se queres ampliar un pouco máis os teus coñecementos sobre swing aquí tes o enlace á páxina principal da documentación desta biblioteca no sitio de Oracle.

[Swing en Oracle.](#)

### 1.4.1 Engadir e eliminar compoñentes da interface

Os compoñentes pódense engadir desde a paleta, que, se lembramos adoita ancorarse á dereita da interface do IDE NetBeans, e mentres non se use queda replegada (na imaxe podes ver resaltado o botón que a fai aparecer).



Á dereita tes a lista de controis para engadir a unha interface en NetBeans. Están organizados nas seguintes categorías:

**Colectores swing:** son secundarios na xerarquía de colectores e úsanse para distribuír e organizar o resto de controis.

**Controles swing:** básicos para crear unha interface útil para comunicarse co usuario e mostrar ou solicitar información.

**Menús swing:** inclúen os controis necesarios para crear menús de aplicación complexos, con varios bloques, elementos activos e inactivos, etc. e menús contextuais (Popup Menu)

**Xanelas swing:** permiten engadir á aplicación xanelas (JFrame), diálogos (JDialog), selectores de ficheiros e cores (JFileChooser e JColorChooser) e paneis de opcións (JOptionPane) para crear diálogos que se contestan con Si/Non.

A adición de compoñentes á interface realízase seleccionando o control na paleta e picando sobre a interface que se está construíndo. O control aparece na interface co seu aspecto por defecto que podes modificar. Se arrastras o control moverase sobre a superficie do seu colector e se fas clic sobre unha esquina e desprazas o rato cambiaralo de tamaño.

Ao colocar un control sobre un formulario aparecen unha guías que che permiten colocalo con máis facilidade, isto será así mentres que teñas activa a opción deseño libre, pódelo comprobar no inspector facendo clic co botón secundario no nodo raíz da interface e seleccionando activar xestor de distribución. Se moves un control estas guías permitiránche relacionalo con outros compoñentes para que o poidas alinear mellor.

Conforme vas colocando controis no panel do formulario estes aparecen reflectidos, ademais, no Inspector, de forma que os seleccionas tanto facendo clic sobre eles como sobre o seu nome. Traballar co inspector facilita colocar controis dentro de colectores porque admite operacións de arrastrar e soltar.

Para eliminar un control basta con seleccionalo (de calquera das formas descritas) e pulsar a tecla Supr, ou ben seleccionar a opción Suprimir do menú contextual.

Se tes interese en coñecer un pouco máis acerca dos compoñentes gráficos de swing e velos en acción, a través de exemplos podes visitar estes enlaces á páxina oficial de Java de Oracle e a [javaswing.org](http://www.javaswing.org).



A Visual Guide to Swing Components

<http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html>

## 1.4.2 Modificación de propiedades

Unha vez que colocaches un control no formulario podes modificar as súas propiedades para adaptalo á túa interface. Co control seleccionado accedes ás súas propiedades no panel propiedades, que lxicamente, será diferente para cada tipo de control.

Adóitase comezar por modificar o nome do control para localizalo con máis facilidade no código da clase que xera o formulario.

Para facelo podes sacar o menú contextual do control no Inspector e seleccionar **cambiar nome da variable**.

Outra propiedade moi común é o **ToolTipText**, é un pequeno recuadro de cor amarela, normalmente, no que aparece unha breve descrición de para que serve o control. Pódolo establecer na propiedade ToolTipText.



### 1.4.3 Engadir funcionalidade desde NetBeans

Creamos interfaces para permitir que o usuario poida interactuar coa aplicación pero sempre será necesario engadir código para darlles a funcionalidade para a que foi creada. Cando usamos unha contorna integrada como NetBeans parte deste código xérase de forma automática facilitando en gran medida o traballo do desenvolvedor ou desenvolvedora, pero terás que abrir este código e modificar algunhas cousas para que o formulario realice as tarefas para as que foi deseñado.



Tarefa 2.3

### 1.4.4 Localización e aliñamento dos compoñentes

Internamente a ferramenta emprega o mecanismo de Java para dispor os elementos chamado **Layout**, distribución ou deseño. Swing dispón de *oito* tipos de distribucións:

**BoderLayout**: aloxa os compoñentes nos límites do formulario, polo que cando os colocamos debemos indicar se van ao norte, sur, leste ou oeste.

**GridLayout**: Deseña mediante unha reixa, na que os compoñentes se organizan por filas e columnas.

**GridBagLayout**: semellante a GridLayout, pero permite a un compoñente que ocupe máis dunha cela.

**CardLayout**: deseño por paneis. Permite a colocación de distintos compoñentes en momentos distintos da execución.

**BoxLayout**: deseño en caixa. Coloca os compoñentes nunha fila ou columna axustándose ao espazo que haxa.

**FlowLayout**: deseña aloxando os compoñentes de esquerda a dereita mentres quede espazo, se non queda pasa á fila seguinte.

**GroupLayout**: creouse para ser utilizado en ferramentas de deseño gráfico de interfaces. Traballa por separado a distribución vertical e horizontal para definir exactamente o posicionamento dos compoñentes. Utilízase en NetBeans.

**SpringLayout**: é moi flexible e úsase tamén para ferramentas de deseño gráfico de interfaces. Neste caso especifícanse as relacións entre os límites dos compoñentes baixo o seu control.

Programar o deseño dun formulario é unha das tarefas máis arduas en Java, aínda que está amplamente superado grazas ao uso de IDEs que facilitan a colocación de compoñentes a golpe de rato e sen necesidade de escribir código. Por exemplo NetBeans, usa o deseño GroupLayout.



Se queres ver un exemplo de como se distribúen obxectos con cada unha das distribucións que vimos podes visitar esta páxina:

<http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>

### 1.4.5 Enlace de compoñentes a orixes de datos

Un dos aspectos que achega maior flexibilidade e potencia a unha aplicación é que poida acceder á información contida nunha base de datos.

#### 1.4.5.1 Formularios mestre-detalle

Un formulario mestre detalle é aquel en que participan dúas ou máis táboas, entre as que existe unha relación de tipo crave externa, por exemplo, entre os contactos da axenda e os seus correos electrónicos. Esta formado por un formulario principal, que leva asociado outro formulario máis pequeno a continuación, de forma que si no principal aparecen os datos dos contactos, no secundario aparecerán os datos dos correos do rexistro activo do formulario principal.

## 1.5 Asociación de accións a eventos

Nas aplicacións que utilizan unha interface gráfica de usuario (GUI), a diferenza da execución dun programa de consola, onde a orde de execución das instrucións dependerá do método Main, a orde de execución dependerá da interacción que realiza o usuario ou usuaria sobre a aplicación.

Durante o deseño da aplicación deberase considerar que accións desexamos que realice a nosa aplicación, cando debe realizalas, e definir os manexadores de eventos que serán invocados de maneira automática cando sobre a aplicación prodúzase o evento.

Na linguaxe Java xestiónase o modelo de eventos da seguinte forma: os obxectos sobre os que se producen os eventos (event sources) "rexistran" os obxectos que haberán de xestionalos (event listeners), para o que os event listeners haberán de dispor dos métodos adecuados. Estes métodos chamaranse automaticamente cando se produza o evento. A forma de garantir que os event listeners dispoñen dos métodos apropiados para xestionar os eventos é obrigarlles a implementar unha determinada interface Listener.

As interfaces Listener correspóndense cos tipos de eventos que se poden producir. Nos apartados seguintes veranse con máis detalle os compoñentes que poden recibir eventos, os distintos tipos de eventos e os métodos das interfaces Listener que hai que definir para xestionalos. Neste punto é moi importante ser capaz de buscar a información correspondente na documentación de Java.

Cando se está deseñando unha aplicación, unha vez que se engadiu os diferentes compoñentes da interface, procédese a definir os obxectos Listener, que son os obxectos que se encargan de responder os eventos, para cada evento que se queira soportar. Unha vez definidos os obxectos Listener, procédese a implementar os métodos das interfaces Listener que se van a facer cargo da xestión de eventos.

## 1.6 Diálogos modais e non modais

Os diálogos modais utilízanse de forma xeneralizada en aplicacións e polos propios sistemas operativos.

Nunha aplicación de interface gráfica en Java é necesario definir un obxecto da clase Frame. Este obxecto vai ser o diálogo pai de toda a aplicación, derivando do resto de diálogos que queiramos engadir á aplicación. Para engadir un diálogo modal, Java xa permitía a

creación de diálogos modais a través do construtor da clase `JDialog` como comentamos no punto anterior.

Para crear diálogos modais, no construtor do diálogo establecemos o parámetro "modal" a `true`, de forma que este diálogo creado mantén o foco, impedindo que poida ser tomado por calquera outro, dentro da aplicación, de forma que non podemos seguir executando a aplicación até pechalo.

```
package mipaquete;
import javax.swing.*.*;
public class Xanela {
    public static void main(String[] args) {
        JFrame xanela = new JFrame("");
        xanela.setAlwaysOnTop(true);
        xanela.setSize(300,300);
        xanela.setVisible(true);
        JDialog dialogo_Modal = new JDialog(xanela, "Dialogo Modal", true);
        dialogo_Modal.setSize(300,300);
        dialogo_Modal.setLocationRelativeTo(null);
        dialogo_Modal.setVisible(true);
    }
}
```

A partir de Java 6, pódese definir a modalidade dun diálogo utilizando dúas clases estáticas dentro da clase `Dialog`. As clases son `static class ModalityType` e `static class ModalExclusionType`. As dúas clases incorporan unha enumeración que indica o nivel de bloqueo que poden xerar.

A clase `static class ModalityType` serve para definir o tipo de bloqueo, ou tamén chamado alcance do bloqueo nunha aplicación, a súa enumeración contén:

**APPLICATION\_MODAL:** bloquea todas as xanelas da aplicación, excepto as xanelas que se deriven dela.

**DOCUMENT\_MODAL:** bloquea a xanela pai da xanela e a súa xerarquía superior.

**MODELESS:** non bloquea ningunha xanela.

**TOOLKIT\_MODAL:** bloquea as xanelas de nivel superior que corren no mesmo `TOOLKIT`.

Podemos excluír do bloqueo a unha xanela ou diálogo utilizando a clase `ModalExclusionType`, esta pode excluír do bloqueo segundo algunha destas opcións da súa enumeración:

**APPLICATION\_EXCLUDE:** A xanela que utiliza este valor de enumeración non será bloqueada por ningún diálogo da aplicación.

**NON\_EXCLUDE:** Indica que a xanela non estará excluída do bloqueo si este ocorre.

**TOOLKIT\_EXCLUDE:** Indica que non se bloqueará esta xanela si chamácese a `APPLICATION_MODAL` ou `TOOLKIT_MODAL`.

### 1.6.1 Diálogos non modais

Dentro dunha aplicación de interface gráfico, podémonos atopar con aplicacións que presentan varias xanelas ou diálogos abertos de maneira simultánea. Diremos que os diálogos son non modais, se podemos pasar o foco dun diálogo a outro sen ningún tipo de restrición.

Un exemplo de aplicación GUI que presenta formularios non modais é a propia contorna de desenvolvemento `NetBeans`, onde podemos pasar dunha xanela a outra sen ningún problema.

Para definir diálogo non modais en Java, o código que se utiliza é o seguinte:

```
package mipaquete;
import javax.swing.*;
public class Xanela {
    public static void main(String[] args) {
        JFrame xanela = new JFrame("");
        xanela.setAlwaysOnTop(true);
        xanela.setSize(300,300);
        xanela.setVisible(true);
        JDialog dialogoNoModal = new JDialog(xanela, "Dialogo Non Modal");
        dialogoNoModal.setSize(300,300);
        dialogoNoModal.setLocationRelativeTo(null);
        dialogoNoModal.setVisible(true);
    }
}
```

Un diálogo non modal permite cambiar o foco a calquera outro diálogo ou xanela abertos no sistema.

## 1.7 Edición de código xerado por ferramentas de deseño.

A implementación dunha aplicación de interface gráfica (GUI) require a definición duns moitos obxectos para a interface, establecer as súas propiedades, coñecer os eventos que se queren controlar para poder crear os manexadores de eventos, etc. Si a aplicación é simple, pode resultar asumible a implementación desa forma, pero se a aplicación ten certa envergadura, sería un traballo arduo e lento.

Na actualidade non se concibe a implementación dunha aplicación GUI sen o uso dunha contorna de desenvolvido, que de maneira rápida e visual, permítanos crear os diálogos da aplicación, engadir aqueles obxectos que desexemos e establecer de forma gráfica e rápida o aspecto que nos interese.

Cos IDE, por exemplo NetBeans, podemos crear a interface de maneira rápida, e sen ter que implementar unha gran cantidade de liñas de código, que a contorna realiza por nós.

A pesar de todas as facilidades que nos ofrecen os IDE, sempre é necesario modificar determinadas aspectos do código. Para iso podemos modificar coa xanela de código.

O IDE automaticamente xera bloques de código cando realizas o deseño dunha xanela coas ferramentas gráficas en Vista "Deseño". Este código móstrase enmarcado dentro do ficheiro .java, pero pode ser modificado. Pódese modificar a maneira de inicializar os compoñentes, os formularios, as propiedades dos compoñentes editándoos na xanela "Fonte" que nos mostra o código fonte xerado a resposta ao teu deseño gráfico. Ademais podes escribir código personalizado e específico onde consideres necesario.

Para modificar un compoñente da xanela, pódese actuar da seguinte forma:

Na xanela Inspector, seleccionas o compoñentes que ao que queres editar o seu código de inicialización.

1. Fai clic no botón Código na parte de arriba da xanela de Propiedades para ver o código

das propiedades.

2. Selecciona a propiedade que desexes editar e engade o valor que queiras.

O IDE actualizará o compoñente seleccionado no código que xerara co novo valor. Para modificar a inicialización de código xerado para unha propiedade dun compoñente deber actuar da seguinte forma:

3. Selecciona o compoñente na xanela Inspector.

4. Feixe clic no botón Propiedades na xanela de propiedades.

5. Selecciona a propiedade que queiras modificar no código de inicialización.

6. Ha clic no botón (...) para chegar á xanela de diálogo Editor de Propiedades.

7. Establece o valor de que desexes que teña a propiedade en cuestión.

Para modificar de maneira libre o código xerado polo IDE, simplemente podes seleccionar o botón "Fonte" que aparece na barra de ferramentas do IDE, e poderás modificar directamente calquera parte do código e adaptalo ás túas necesidades.

## 1.8 Clases, propiedades, métodos

Para a implementación de Interfaces Gráficas de Usuario en Java, JavaSoft creou un conxunto de clases que son agradables desde o punto de vista visual e fáciles de utilizar para o programador. Esta colección de clases son as Java Foundation Classes (JFC), que na plataforma Java 2 están constituídas por cinco grupo de clases:

AWT, Java 2D, accesibilidade, arrastrar e soltar e swing.

**AWT.** Bibliotecas de clases Java para o desenvolvemento de Interfaces gráficas de usuario.

**Java 2D.** É un conxunto de clases gráficas baixa licenza IBM/Taligent.

**Accesibilidade,** proporciona clases para facilitar o uso de computadores e tecnoloxía informática a persoas con deficiencias visuais como lupas de pantallas, etc.

**Arrastrar e soltar** (Drag and Drop), son clases na que se soporta os [JavaBeans](#).

**Swing.** É a parte máis importante para o desenvolvemento de aplicacións de interface gráfica. Swing proporciona un conxunto de compoñentes moi ben descritos e especificados, de forma que a súa presentación visual é independente da plataforma onde se executa a aplicación que utilice as súas clases. Swing estende AWT engadindo un conxunto de compoñentes, `JComponents`, e as súas clases de soporte.

Actualmente, swing a desprazado a AWT debido a que os compoñentes de swing, ao estar escritos en Java, ofrecen maior funcionalidade, e independente da plataforma.

### 1.8.1 Clases

No desenvolvemento de interfaces gráficas de usuario baseada na linguaxe Java, actualmente utilízanse os compoñentes swing.

Os compoñentes swing son obxectos de clases derivadas da clase `JComponent` que deriva da clase `java.awt.Component`. Para crear interfaces gráficas de usuario, swing combina os compoñentes da clase `JComponent` en colectores de nivel alto `JWindow`, `JFrame`, `JDialog` e `JApplet`.

**JWindow** é unha xanela sen barra de título e sen botóns.

**JFrame** é unha xanela con barra de título e cos botóns que permiten a súa manipulación.

**JDialog** permite visualizar un cadro de diálogo.

**JApplet** permite crear un **applet swing**.



Para dar funcionalidade ás xanelas, o swing proporciona un conxunto de compoñentes que derivan da clase `JComponent`, os máis utilizados son:

**`JComponent`:** Clase basee para os compoñentes swing.

**`AbstractButton`:** Define o comportamento común dos botóns e os menús.

**`JButton`:** Botón.

**`JMenuItem`:** Elemento dun menú.

**`JCheckBoxMenuItem`:** Elemento do menú que se pode seleccionar ou deseleccionar.

**`JMenu`:** Menú dunha barra de menús.

**`JRadioButtonMenuItem`:** Elemento que forma parte dun grupo de elementos de menú.

**`JToggleButton`:** Botón de estados.

**`JCheckBox`:** Casa de verificación.

**`JRadioButton`:** Botón de opción.

**`JColorChooser`:** Diálogo para seleccionar cores.

**`JComboBox`:** Combinación de caixa de texto e lista despregable.

**`JLabel`:** Etiqueta.

**`JList`:** Lista despregables.

**`JMenuBar`:** Barra de menús.

**`JOptionPane`:** Compoñente que facilita a visualización dun cadro de diálogo.

**`JPanel`:** Colector xenérico.

**`JPopupMenu`:** Menú que aparece cando se selecciona un elemento dunha barra de menús.

**`JProgressBar`:** Barra de progreso.

**`JScrollBar`:** Barra de desprazamento.

**`JScrollPane`:** Área de traballo con barras de desprazamento.

**`JSeparator`:** Separador para colocar entre dous elementos do menú.

**`JSlider`:** Permite seleccionar un valor dentro dun intervalo que define.

**`JTableHeader`:** Utilízase para manexar a cabeceira dunha táboa.

**`JTextComponent`:** Clase basee para os compoñentes de texto.

**`JEditorPane`:** Edita diferentes tipos de contido.

**`JTextPane`:** Edita texto con formato, incluíndo imaxes.

**`JTextArea`:** Caixa de texto multilínea.

**`JTextField`:** Caixa de texto dunha liña.

**`JPasswordField`:** Úsase para introducir contrasinais.

**`JToolBar`:** Barra de ferramentas.

## Propiedades

**As propiedades dos compoñentes vannos a permitir adaptar o seu comportamento e aparencia.**

As propiedades máis importantes que presentan a maioría dos compoñentes swing son as seguintes:

**`background`:** Determina a cor de fondo do compoñente.

**`font`:** Establece o tipo de fonte que vai mostrar o compoñente.

**`foreground`:** Establece a cor da fonte que mostra o compoñente.

**horizontalAlignment:** Establece a aliñación do texto dentro do compoñente no eixo X.

**icon:** Indica si a compoñente mostra ou non unha icona, e cal sería.

**labelFor:** Indicaría si o compoñente é etiquetable.

**text:** Mostra o texto que indica a propiedade en compoñentes como caixa de texto ou etiquetas.

**toolTipText:** Con esta propiedade definimos o texto que aparece como tool tip.

**verticalAlignment:** Establece a aliñación do texto dentro do compoñente no eixo E.

**alignmentX:** Aliñamento horizontal preestablecido para o compoñente.

**alignmentY:** Aliñamento vertical preestablecido para o compoñente.

**autoscrolls:** Determina si o compoñente pode realizar scroll de forma automática cando se arrastra o rato.

**border:** Establece o tipo de bordo que vai presentar o compoñente.

**componentPopupMenu:** Estable o menú contextual que se mostra neste compoñente.

**cursor:** Establece o tipo de cursor que se vai a mostrar cando o curso entre no compoñente.

**disableIcon:** Establece a icona a mostrar si o compoñente non está activo.

**enabled:** Indícanos si o compoñente está ou non activo.

**focusable:** Establece si o compoñente pode ou non, recibir o foco.

**horizontalTextPosition:** Establece a posición horizontal do texto do compoñente, en relación coa súa imaxe.

**iconTextGap:** Si o compoñente ten activo o texto e a icona, con esta propiedade establécese a distancia entre eles.

**inheritsPopupMenu:** Indica si o menú contextual hérdase ou non.

**inputVerifier:** Establece o compoñente de verificación para este compoñente.

**maximunsized:** Establece o tamaño máximo do compoñente.

**minimunsized:** Establece o tamaño mínimo do compoñente.

**name:** Establece o nome do compoñente.

**opaque:** Modifica a opacidade do compoñente.

**preferredSize:** Tamaño predefinido para este compoñente.

**verifyInputWhenFocusTarget:** Si o compoñente é verificado antes de recibir o foco.

**verticalTextPosition:** Establece a posición vertical do texto do compoñente, en relación coa súa imaxe.

As propiedades que enumeramos anteriormente, son as máis habituais que te podes atopar nos compoñentes dos swing de Java. Para modificar os valores das propiedades dispós da xanela de propiedades do IDE, ou ben podes facelo desde o código fonte Java. Si desexas modificar o valor dunha propiedade desde o código fonte, o IDE vaiche a mostrar unha lista de todas as propiedades dispoñibles para o compoñente en cuestión, unha vez que escribas o nome do obxecto e un punto.

## 1.8.2 Métodos

**Cada compoñente que forma parte dunha aplicación GUI utilizando Java, dispón dun conxunto completo de métodos,** que nos permite comunicarnos co compoñente e modificar o seu aspecto ou comportamento. A continuación vaise a mostrar unha lista dos compoñentes máis importantes do swing Java, xunto cos seus métodos máis destacados:

**JFrame.** Os métodos máis importantes son: `getTitle()`, `setBounds(x, y, w, h)`, `setLocation`,

```
setMaximumSize(w,h), setMinimumSize(w,h), setPreferredSize(w,h), setResizable(bool),  
setSize(w,h), setTitle(str) e setVisible(bool).
```

**JPanel.** O panel de contido dispón do método `add()`, para engadir compoñentes ao panel.

**JLabel:** As etiquetas teñen como métodos máis importantes: `setText()` que permite modificar o texto, `setVerticalAlignment()` para modificar a aliñación vertical, etc.

**JButton:** Os botóns presentan entre outros métodos: `setEnabled(bool)` que permite activar ou desactivar o botón, `isEnabled()` que permite comprobar si está ou non activo, `setMnemonic(char)` que permite asociar unha tecla ao botón, etc.

**JProgressBar, JScrollBar.** Estes compoñentes dispoñen dos seguintes métodos: `setExtent()`, `setMaximum()`, `setMinimum()`, `setValue()`, `getValueIsAdjusting()` e `setOrientation()`.

**JSlider:** Este compoñente ten como métodos máis destacados: `setPaintTicks(bool)`, `setPaintLabels(bool)`, `setMajorTickSpacing(int)` e `setMinorTickSpacing(int)` para determinar os valores do intervalo, `setSnapToTicks(true)` e `setInverted(bool)`.

**JRadioButton.** Este compoñente ten como método principal `isSelected()` que devolve o estado do mesmo.

**JList.** Nas listas despregables destacan os métodos seguintes para editar os elementos da lista:

```
addElement(objName), elementAt(index), removeElement(objName), removeAllElements() e  
removeElementAt(id). Para comprobar o estado da lista úsanse os métodos contains(objName),  
indexOf(name) e size(). Para converter a lista en array úsase copyInto(array).
```

**JComboBox.** Dispón de diferentes métodos, destacando `setEditable(true)` que permite editar os items do combo box. Para recuperar os ítems seleccionados úsase `getSelectedItem()` e `getSelectedIndex()`. Outros métodos útiles son `getItemAt(id)`, `getItemCount()`, `setSelectedItem(obj)` e `setMaximumRowCount(x)`.

**JTextPane e JTextArea.** Como métodos máis útiles destes compoñentes destacan `getText()`, `setText()`, `append(str)`, `insert(str,pos)`, `replace(str,beg,end)`, `setEditable(bool)`, `setLineWrap(bool)` e `setWrapStyleWord(bool)`.

**JMenuItem.** Para engadir separadores úsase o método `addSeparator()` ou `insertSeparator(posn)`. O

Menú ítems pódense activar ou desactivar con `setEnabled(bool)` e comprobar si están activos con `isEnabled()`

. Para engadir teclas de acceso rápido úsase o método `setMnemonic(char)`.

Cada compoñente dispón de moitos máis métodos que podes utilizar nas túas aplicacións, utilizando un IDE como NetBeans, unha vez que definas un obxecto, por exemplo o obxecto `etSaludo` da clase `JLabel`, cando invoques `etSaludo` e escribas a continuación un punto, automaticamente aparecerache unha lista con todos os métodos que podes usar para ese obxecto, xunto cunha breve explicación do seu uso.

**Cada compoñente dispón dos seus propios métodos, mediante os cales podemos comunicarnos con el en tempo de execución, permitindo adaptar a súa aparencia e comportamento.**

## 1.9 Eventos, escoitadores

Nunha contorna GUI, cando se produce un movemento de rato, púlsase unha tecla, péchase unha xanela, etc. prodúcese un evento. O evento é recibido por un obxecto, como pode ser un botón, unha caixa de texto, etc. O obxecto que recibe o evento, para poder responder a el, debe implementar a interface apropiada (event [handler](#)) e rexístrala como un escoitador (event listener) no compoñente GUI (event source ou obxecto que vai recibir o evento) apropiado.

Os eventos agrúpanse en función ao compoñente que o produce ou á acción que o provoca. O evento ten os seguintes elementos:

A **fonte do evento** (event source): É o compoñente que orixina o evento.

O **escoitador** (event listener): é o encargado de atrapar ou escoitar o evento.

O **manexador do evento** (event handler), é o método que permite implementar a interface.

O manexador do evento recibe un obxecto evento (ActionEvent) que contén información sobre o evento que se disparou, cando isto sucede, o manexador do evento descifra o evento con devandito obxecto e procesa o solicitado polo usuario ou usuaria.

Un compoñente dunha interface gráfica dispara ou activa os manexadores (event handler) dependendo do tipo de evento que ocorreu. O compoñente pode ter rexistrado máis dun manexador que manexa o mesmo tipo de evento. Un evento pode ser escoitado por máis dun manexador de eventos.

O manexador do evento require que na declaración da clase que manexa o evento (event handler), indique a interface correspondente ao evento (XListener, onde X é o tipo de evento a escoitar). A clase pode implementar máis dun XListener. Se a clase non implementa a interface pode ser que estenda a unha clase que si a implementa:

```
public class miClase implements ActionListener{...}
```

Nos compoñentes que son a fonte do evento (event source) rexístrase unha instancia do manexador do evento (event handler), como un observador ou escoita do tipo de eventos que manexa (XListener). É dicir, dille ao compoñente que vai escoitar os eventos do tipo do manexador.

```
compoñente.addActionListener( instancia_de_miClaseListener);
```

Na clase que manexa o evento (event handler) débense implementar os métodos da interface XListener que descifren o evento (XEvent) e procéseno.

```
public void actionPerformed(ActionEvent e) {  
    ...//Código que reaccione á acción  
}
```

## 1.9.1 Escoitadores

**Un escoitador (listener) é o obxecto da clase que implementa a interface de escoita dun evento e que contén o método de resposta ao propio evento.** Cando se produce un determinado evento dentro da aplicación GUI, si deséxase responder a eses eventos, a aplicación deberá implementar unha serie de métodos que se executen de forma automática cando se produza eses eventos. Os métodos que se van a implementar para responder aos diferentes eventos dentro dunha aplicación de interface gráfica, defínense nunhas interfaces denominadas interfaces de escoita.

Cada interface de escoita contén os métodos para xestionar un determinado grupo de eventos. A interface de escoita WindowListener define o formato dos métodos para a xestión dos eventos de xanela, como poden ser abrir a xanela, pechala, maximizala, minimizala etc. Algúns dos métodos de WindowListener son:

```
public void windowActivated(WindowEvent e): Invocado cando a xanela é a xanela activa.
```

```
public void windowDeactivated(WindowEvent e): Invocado cando a xanela deixa de ser a xanela activa.
```

```
public void windowClosing(WindowEvent e): Cando a xanela se pecha.
```

```
public void windowClosed(WindowEvent e): Cando a xanela se minimiza.
```

```
public void windowIconified(WindowEvent e): Cando a xanela se restablece.
```

```
public void windowDeiconified(WindowEvent e): A primeira vez que a xanela se minimiza.
```

```
public void windowOpened(WindowEvent e): A primeira vez que a xanela se fai visible.
```

Cada interface de escoita contén os seus propios métodos para á xestión de eventos. Exemplos de eventos que son xerados por compoñentes do swing son:

**ActionListener:** Captura certo tipo de acción realizada sobre certos compoñente. Por exemplo,

pulsar un botón, seleccionar un elemento nunha lista despregables ou unha opción nun menú.

**ChangeListener:** Rexistra os cambios sobre certos compoñentes.

**ItemListener:** Recolle o cambio de estado nun compoñente tipo listas despregables.

**MouseListener:** Eventos producidos pola manipulación do rato.

**MouseMotionListener:** Movemento do rato sobre un compoñente.

**ComponentListener:** Visibilidade de compoñentes.

**FocusListener:** Captura eventos relacionados coa recepción ou perda do foco.

**KeyListener:** Captura pulsacións do rato sobre o compoñente activo.

**ListSelectionListener:** Selección de elementos dentro dunha lista.

Para responder a un evento dentro dunha aplicación, deberás definir unha clase que interprete a interface de escoita correspondente ao tipo de evento que queiras xestionar. Aos obxectos desa clase de escoita denomínaselles escoitadores.

**Se se quere responder a un determinado evento nunha aplicación, hai que crear un obxecto que escoite cando se produce o evento e responda a el.**



#### Un manexador de evento

- É o compoñente que provoca o evento.
- É o compoñente que orixina o evento.
- Recibe un obxecto evento, descifrando o evento e procesando o solicitado polo usuario.



## 2. Tarefas

---

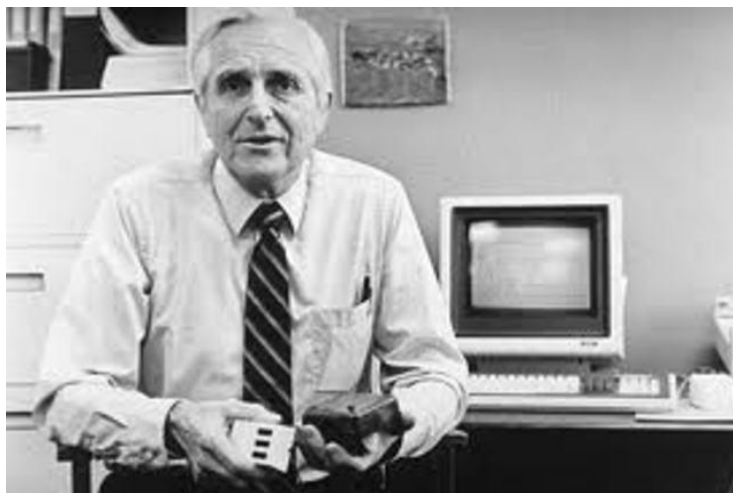
### 2.1 Evolución das interfaces de usuario



<https://www.timetoast.com/timelines/evolucion-multimedia>

Busca tres imaxes, e explica que avance supuxo no ámbito das interfaces de usuario respecto ao que había ata ese momento.

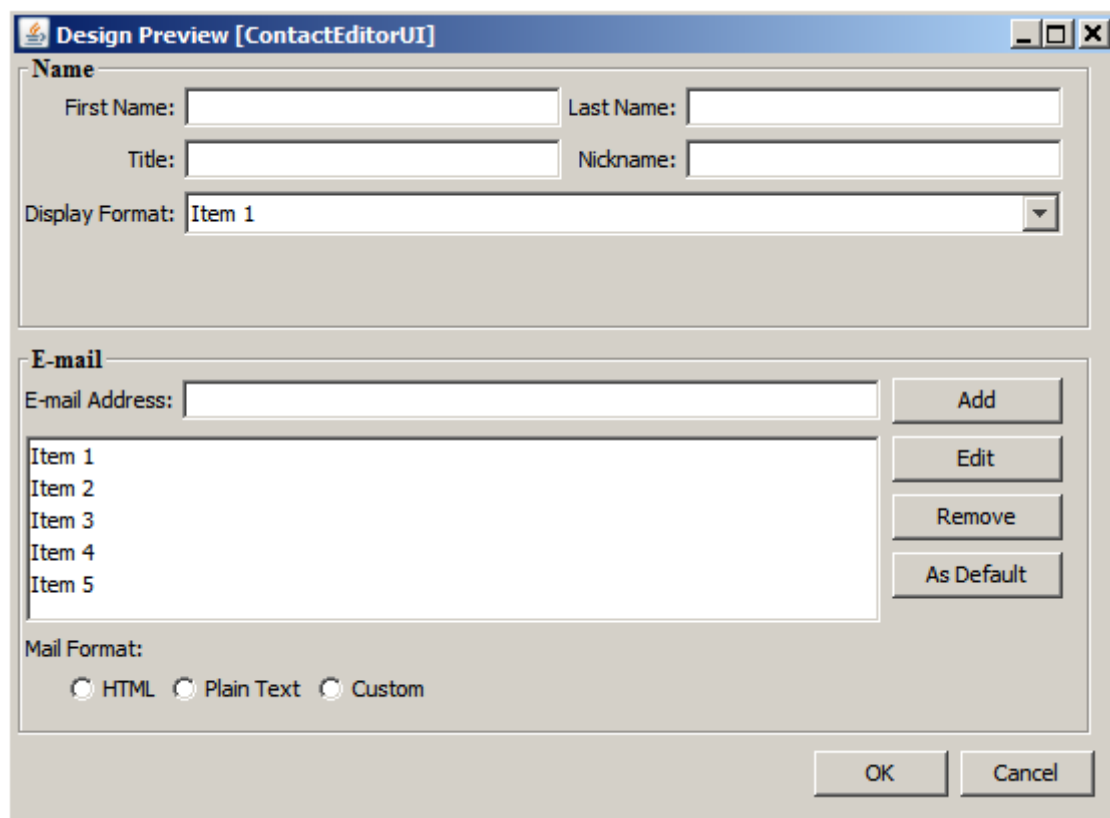
- Sketchpad
- oN-Line System



- Xerox Star 8010
- X Windows system 1
- Windows 1.0
- OPENSTEP
- Gnome

## 2.2 Colectores e compoñentes

Crea a interface de usuario que se observa na figura dimensionando e aliñando os compoñentes. Establece o comportamento de auto-redimensionado e edita as propiedades adecuadas dos compoñentes.



The image shows a 'Design Preview' window for a contact editor. The window has a title bar 'Design Preview [ContactEditorUI]' and standard window controls. It is divided into two main sections: 'Name' and 'E-mail'. The 'Name' section contains four text input fields: 'First Name', 'Last Name', 'Title', and 'Nickname', followed by a 'Display Format' dropdown menu currently set to 'Item 1'. The 'E-mail' section features an 'E-mail Address' input field, a list box containing 'Item 1' through 'Item 5', and four buttons: 'Add', 'Edit', 'Remove', and 'As Default'. At the bottom of the 'E-mail' section are three radio buttons for 'Mail Format': 'HTML', 'Plain Text', and 'Custom'. The 'HTML' radio button is selected. At the bottom right of the window are 'OK' and 'Cancel' buttons.



Se o precisas podes usar a guía correspondente da documentación de NetBeans

<https://netbeans.org/kb/docs/java/quickstart-gui.html>

## 2.3 Codificado

Le o código que permite xerar a interface gráfica. Debuxa o resultado.

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;

public class Ejemplo {
    public Ejemplo() {
        JFrame frame = new JFrame("Ejemplo");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        JLabel label = new JLabel("Etiqueta");
        JButton button = new JButton("Botón");
        panel.add(label);
        panel.add(button);
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,100);
        frame.setVisible(true);
    }
}
```

# Anexos

---

## 2.4 Nomenclatura

En xeral, para definir elementos de Java Swing debemos eliminar as J iniciais e todas as vogais, e extraer as tres consonantes máis significativas do nome do control. A continuación figura a lista de compoñentes e os seus prefixos asociados:

### Nomenclatura de Swing Containers

#### Contedor

#### Prefixo

JInternalFrame	ifrm
JPanel	pan
JScrollPane	scp
JTabbedPane	tpn
JToolBar	tlb

### Nomenclatura de Swing Controls

#### Control

#### Prefixo

JButton	btn
JButtonGroup	btg
JCheckBox	cbx
JComboBox	cmb
JLabel	lbl
JList	lst
JPasswordField	pwd
JProgressBar	pgb
JScrollBar	scb
JTable	tbl
JTextArea	txa
TextField	txt
JTextPane	txp
JTree	jt
JDateChooser	jdc
JCalendar	jcl
JRadioButton	jrb

### Nomenclatura de Swing Menus

#### Menu

#### Prefixo

JMenu	mnu
JMenuBar	mnb
JMenuItem	mni

### Nomenclatura de Swing Windows

#### Window

#### Prefixo

JColorChooser	cch
JDialog	dlg
JFileChooser	jfc

JFrame  
JOptionPane

frm  
opt

### Outros

#### Window

DefaultTableModel  
JDialog  
JFileChooser  
JFrame  
JOptionPane

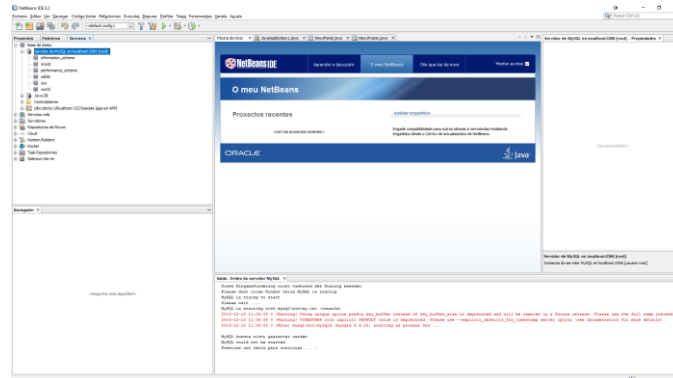
#### Prefixo

dtm  
dlg  
jfc  
frm  
opt

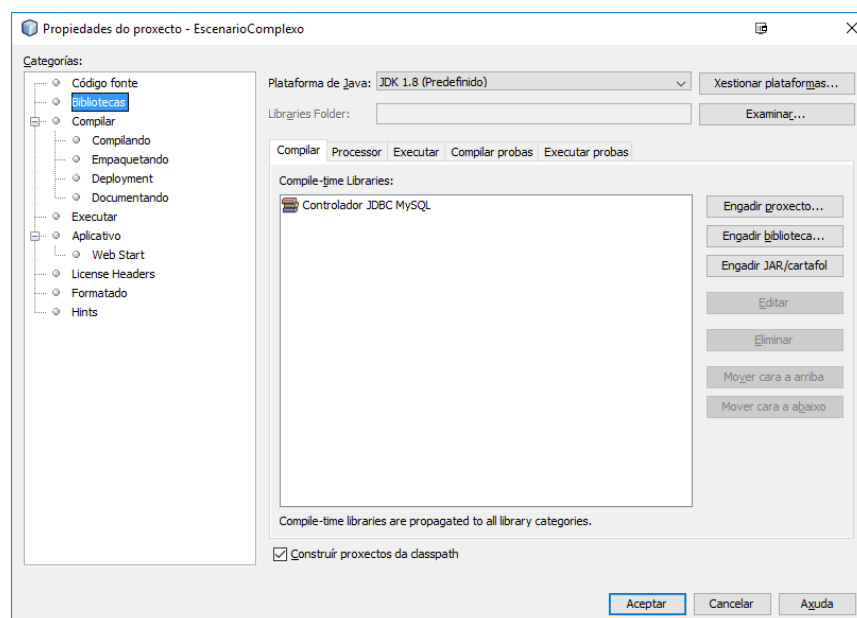


## 2.5 Acceso a datos

Sobre Windows podemos instalar MySQL dende a web do fabricante mediante un dos dous MSI <https://netbeans.org/kb/docs/ide/mysql.html> pero a opción mais recomendable, para comezar e ter a ferramenta phpMyAdmin dipoñible, é usar XAMPP.



O contraseñal estará baleiro por defecto. Se o proxecto é vello e as referencias están rotas só temos que engadir a librería do controlador JDBC MySQL



O código para realizar unha conexión coa base de datos pode ser algo coma isto:

```

public class Conexion {
    private static Connection conexion;

    public static int conectar(String url,String porto,String usuario,String bd,String clave){

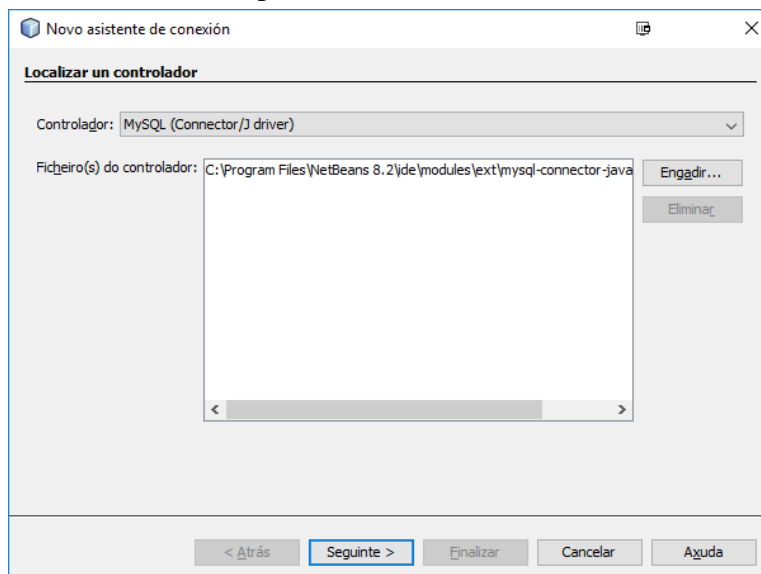
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            String cadeaConexion="jdbc:mysql://"+url+": "+porto+"/ "+bd+"?user="+usuario+"&password="+clave;
            //System.out.println("Cadea de conexion:"+cadeaConexion);
            conexion=DriverManager.getConnection (cadeaConexion);
            return 0;
        }
        catch (SQLException e)
        {
            e.printStackTrace();
            return -1;
        }
        catch (ClassNotFoundException e)
        {
            e.printStackTrace();
            return -2;
        }
    }

    public static Connection getConexion()
    {
        return conexion;
    }

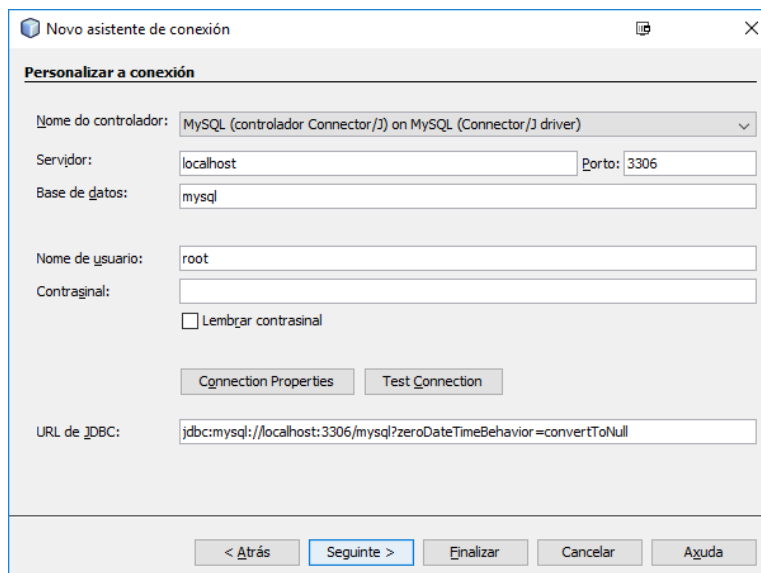
}

```

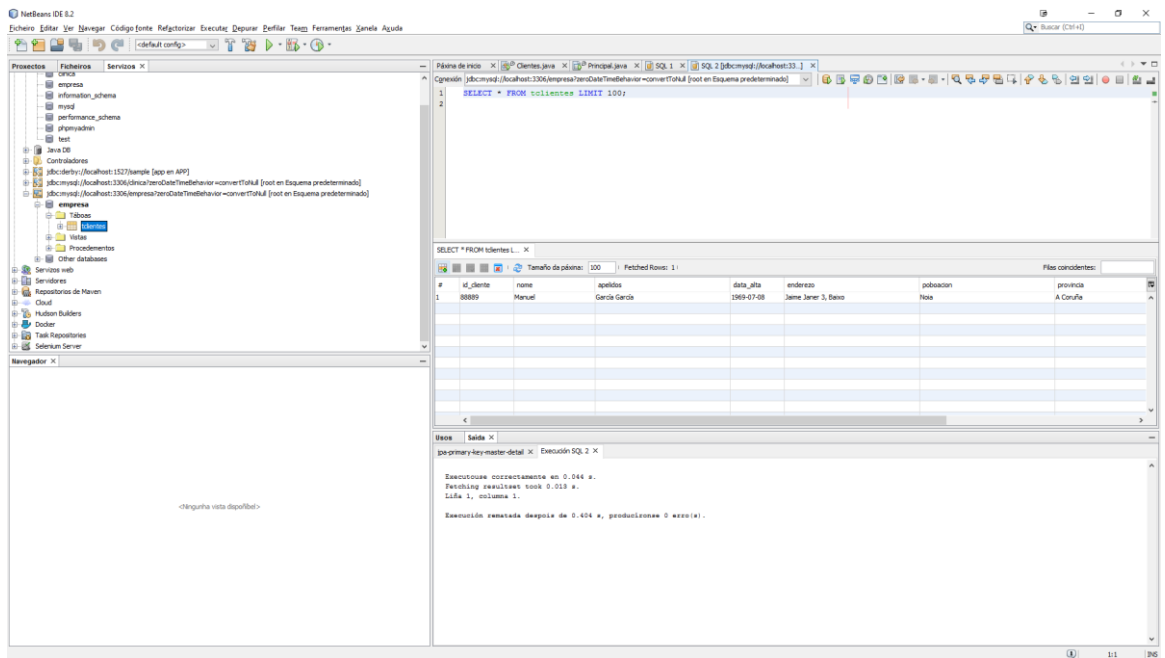
Se temos o XAMP podemos crear as bases de datos dende el.



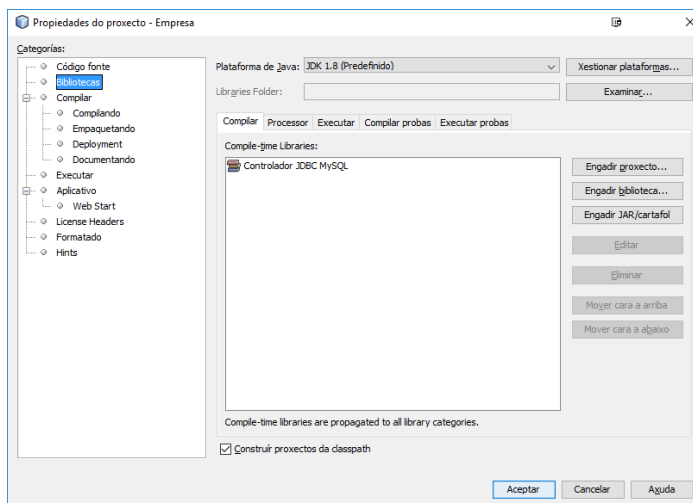
Conectamos a BD



Podemos editar rexistros dende o propio IDE:



Aención a que a librería estea no proxecto!



### 3. Recursos

---



- Plataforma de FP a distancia. <http://www.edu.xunta.es/fp/fpdistancia>
- NetBeans: <http://wiki.netbeans.org/NetBeansUserFAQ#section-NetBeansUserFAQ-GUIEditorMatisse>
- Javacodegeeks.com