

DI06.- Elaboración de interfaces mediante documentos XML.

1.- Lenguajes basados en XML.

¿Qué es XML?

XML (eXtensible Markup Language o Lenguaje de Marcado eXtensible) es un lenguaje basado en texto de definición de documentos, que permite representar información estructurada de gran variedad de documentos.

Al estar puramente basado en texto, un documento XML será un **archivo plano** en el que, a través de una serie de elementos propios se definen aspectos del documento a generar, pero no de su diseño propiamente dicho, sino de su estructura.



[\(link: DI02_CONT_R03_Eschema_Lenguajes_XML.png.\)](#)

Orígenes

Hacer independiente la estructura de un documento de la constante evolución de las herramientas que se usan para visualizarlo ha sido, desde hace tiempo, foco de interés de los desarrolladores y desarrolladoras de aplicaciones informáticas. En los años sesenta, Charles F. Goldfarb, por encargo de **IBM** desarrolló el lenguaje **GML**, Generalized Markup Language, cuyo objetivo era describir la estructura de los documentos de forma que el resultado no dependiese de una determinada plataforma o una aplicación específica. De la evolución de GML surgió **SGML**, que se convirtió en el estándar internacional **ISO 8879**. Sin embargo tuvo difusión tan sólo en medios académicos y de la administración, pero no tuvo demasiado éxito entre los usuarios medios debido a su dificultad.

El hito que supuso la amplia difusión de las tecnologías de la comunicación y la circulación de documentos electrónicos entre cualquier punto del mundo fue la creación de la World Wide Web (Tim Berners-Lee, en 1990), y, asociado a ello la aparición de **HTML**, lenguaje de descripción de documentos basado en SGML para la web.

HTML es un lenguaje que permite combinar en un solo documento el contenido con el formato, por ejemplo:

- ✓ Si en un documento **HTML** se añade el siguiente código: "El padre del lenguaje `GML` fue `Charles F. Goldfarb`, que lo desarrolló por encargo de `IBM`."
- ✓ Conseguiré algo parecido a esto: El padre del lenguaje **GML** fue **Charles F. Goldfarb**, que lo desarrolló por encargo de **IBM**.

Ya que encerrar un texto entre `` y `` equivale a aplicar el formato negrita. Estos elementos de HTML como `` y `` se denominan **etiquetas**, cada etiqueta tiene un significado en el diseño del documento final o en su estructura.

XML, sin embargo, **no incluye ninguna orientación al diseño** es puramente un lenguaje estructural cuyo objetivo es definir cómo se organiza la información en un documento, el diseño se definirá después, de esta forma se hace independiente al documento de la plataforma, e incluso de la aplicación con la que se va a visualizar (Web, impreso, como documento de texto, y más). Por ejemplo, si definimos una página web con XML podremos verla fácilmente en un navegador web, en un teléfono móvil, o, incluso, en un lector Braille, sin modificar el documento de base. Lo que cambia es la forma en que se interpreta el contenido.

Autoevaluación

¿Cual de las siguientes afirmaciones es correcta?.

- ☐ [\(link: \)](#)
SGML ha sido un lenguaje de muy amplia difusión desde su creación en los años ochenta.
- ☐ [\(link: \)](#)
La estructura de un documentos XML depende totalmente de la aplicación con la que se visualizará.
- ☐ [\(link: \)](#)
El lenguaje HTML es un lenguaje XML puro.
- ☐ [\(link: \)](#)
Con HTML podemos indicar cuestiones tales como el color de la fuente o si vamos a visualizar un texto centrado.

1.1.- Etiquetas.

XML es un tanto parecido a HTML en el sentido de que utiliza etiquetas para definir elementos dentro de una estructura, con la diferencia de que cuando usamos HTML se trabaja con una serie de etiquetas predefinidas, y en XML se pueden definir según convenga a las necesidades del proyecto.



[\(link: DI02_CONT_R04_Estructura_Arbol_Generica.png.\)](#)

Las etiquetas están delimitados por ángulos (<,>) e identifican el contenido que delimitan. Pueden tener atributos. Siguen la estructura:

<etiqueta atributo="valor">Contenido</etiqueta>

Las etiquetas en un documento XML tienen siempre **etiqueta de cierre**, de manera que toda la información referente al elemento queda comprendida entre ambas.

Siempre existe una **etiqueta raíz**, que hace referencia al tipo de objeto que se describe. El resto son características o elementos del objeto, y se colocan anidadas dentro de la etiqueta raíz como etiquetas hijas. Por eso decimos que un documento XML tiene **estructura de árbol**.

Las etiquetas en un documento XML tienen siempre etiqueta de cierre, de manera que toda la información referente al elemento queda comprendida entre ambas.

Siempre existe una etiqueta raíz, que hace referencia al tipo de objeto que se describe. El resto son características o elementos del objeto, y se colocan anidadas dentro de la etiqueta raíz como etiquetas hijas. Por eso decimos que un documento XML tiene estructura de árbol.

Estructura de árbol de un documento XML genérico

Documento XML	Estructura de árbol
<pre><Etiqueta_raiz> <etiqueta_hija> <subetiqueta_hija>...</subetiqueta_hija> <subetiqueta_hija>...</subetiqueta_hija> ... </etiqueta_hija> </Etiqueta_raiz></pre>	<div></div> <div>(link: DI02_CONT_R04_Estructura_Arbol_Generica.png.)</div>

Por **ejemplo**, si queremos usar XML para almacenar la información de nuestra agenda de contactos podríamos definir una etiqueta llamada contacto para cada entrada de la agenda, y dentro de los contactos otras etiquetas para el nombre, teléfono, fecha de nacimiento, grupo, entre otras. Si tengo un amigo que se llama Javier, cuyo número de teléfono es 637-059874 y que nació el 14 de diciembre de 1982, almacenaría la información referente a él en mi agenda de la siguiente manera:

Estructura de árbol del documento XML para la agenda

Documento XML	Estructura de árbol
---------------	---------------------

Documento XML

Estructura de árbol

```
<agenda>
<contacto>
  <nombre>Javier</nombre>
  <teléfono>637059874</teléfono>
  <fecha_nacimiento>14-12-1982</fecha_nacimiento>
  <grupo>Amigos</grupo>
</contacto>
</agenda>
```



[\(link: DI02_CONT_R05_Estructura_arbol_ejemplo.png.\)](#)

Donde la etiqueta raíz sería agenda, contacto sería una etiqueta hija y nombre, teléfono, fecha_nacimiento y grupo funcionarían como subetiquetas.

1.2.- Atributos y valores.

Se dice que un elemento XML tiene contenido cuando se ha añadido algún texto entre la etiqueta de apertura y cierre. Sin embargo pueden darse casos en los que no se pueda almacenar toda la información pertinente del contenido sólo en el texto que encierran las etiquetas.

Cuando necesitamos añadir información adicional a un elemento XML de alguna manera modificamos la etiqueta añadiéndole **atributos**.

`<etiqueta atributo="valor">Contenido</etiqueta>`

Los atributos permiten proporcionar información adicional sobre el elemento. Por ejemplo, puedo precisar si el teléfono que he guardado para mi amigo Javier es su teléfono fijo, su móvil o el teléfono del trabajo añadiendo el atributo tipo:



```
<agenda>
  <contacto>
    <nombre>Javier</nombre>
    <teléfono tipo="móvil">637059874</teléfono>
    <fecha_nacimiento>14-12-1982</fecha_nacimiento>
    <grupo>Amigos</grupo>
  </contacto>
</agenda>
```

No siempre ocurre que cumpliendo los requisitos anteriormente comentados el documento XML sea correcto. Para asegurarnos de que un documento XML esté bien formado debe cumplir las siguientes reglas:

- ✓ Debe existir un elemento raíz.
- ✓ Todos los elementos XML deben tener su correspondiente etiqueta de cierre.
- ✓ Es sensible a mayúsculas.
- ✓ El anidamiento debe hacerse conforme a la estructura del árbol del documento. Es decir, si abro la etiqueta y a continuación se cierran en sentido inverso, no puedo cerrar antes y después
- ✓ Los valores de los atributos van entrecomillados siempre.

A la hora de poner nombre a las etiquetas se deben seguir las siguientes recomendaciones:

- ✓ Se pueden usar letras, número y otros caracteres.
- ✓ No se puede empezar por un número o signo de puntuación.
- ✓ No se puede empezar por las letras XML.
- ✓ Los nombres no pueden contener espacios en blanco.

Autoevaluación

Texto de la pregunta: ¿cuál es la expresión correcta?

- (link:)
<fecha de nacimiento>14-12-1982</fecha de nacimiento>
- (link:)
<teléfono tipo=movil>637059874</teléfono>
- (link:)
<contacto>
<nombre>
Javier
</contacto>
</nombre>
- (link:)
<email tipo="personal">javier@gmail.com</email>.

2.- Lenguajes de descripción de interfaces basados en XML.



2.1.- Proceso de elaboración de las interfaces.

La primera posibilidad para desarrollar una interfaz suele ser emplear el mismo lenguaje, normalmente de alto nivel, que se usa para implementar la funcionalidad de la aplicación como Java, C#, etc. Tiene como ventaja la sencillez, puesto que no es preciso adquirir nuevos conocimientos, la creación de la interfaz se integra en el proceso de desarrollo de la aplicación y, normalmente, no es necesario trabajar con herramientas diferentes, ni en la programación ni posteriormente a la hora de **compilar** o ejecutar el programa. Sin embargo, tiene como principal desventaja la dependencia de la plataforma, del dispositivo y del propio lenguaje. Básicamente, la portabilidad de las interfaces creadas de esta manera serán las que proporcione el lenguaje.



[./link: DI02 CONT R08 esquema mapeado.png./](#)

La creación de interfaces de usuario usando **lenguajes de descripción** basados en XML pretende solventar esto, proporcionando un medio que permita construir interfaces mediante descripciones de alto nivel en los distintos aspectos de la interfaz: estructura y comportamiento, de modo que a partir de la descripción se pueda generar de forma automatizada la interfaz de usuario final. Además este proceso permite centrar el desarrollo en las necesidades del usuario, puesto que no se realiza siguiendo las directrices marcadas por el lenguaje.

En este ámbito se utilizará una notación de alto nivel para desarrollar la interfaz, que se almacena en un archivo aparte, en formato XML. Posteriormente se trata este archivo para obtener el código de la interfaz que se integrará en la aplicación. A este procedimiento se le denomina **mapear** la interfaz. El proceso de mapeado depende del lenguaje concreto que se esté usando.

Tras el correspondiente proceso se podrá visualizar la interfaz en un dispositivo final.

A continuación veremos algunos ejemplos de lenguajes para el desarrollo de interfaces basados en XML y como realizan el proceso de mapeado.

Autoevaluación

¿Qué frase es correcta con respecto al desarrollo de interfaces?

- ☐ [\(link: \)](#)
Es un proceso sencillo porque la misma interfaz es válida para todos los usuarios.
- ☐ [\(link: \)](#)
Una vez creada una interfaz en un lenguaje de alto nivel podremos reutilizarla para cualquier tipo de dispositivo sin necesidad de modificarla.
- ☐ [\(link: \)](#)
Una interfaz descrita con un lenguaje basado en XML se integra directamente en la aplicación.
- ☐ [\(link: \)](#)
El uso de interfaces descritas con lenguajes basados en XML permite desarrollar por separado la interfaz gráfica de una aplicación y comportamiento.

2.2.- XUL.

Es un lenguaje de marcas basado en XML para definir interfaces de usuario complejas, en las que podremos insertar los elementos habituales en un formulario, pero que también dispone de otros más complejos, como menús, barras de herramientas, estructuras jerárquicas o teclas de acceso directo. Así mismo, permite colocar los contenidos de la interfaz usando distribuciones complejas.

Este lenguaje se utiliza específicamente para desarrollar aplicaciones en red. Los documentos escritos en este lenguaje podrán visualizarse en los navegadores Mozilla y Netscape, de hecho la interfaz de ambos está desarrollada en XUL. Ambos navegadores funcionan sobre el motor de **renderizado** Gecko, que es la base que necesitamos para visualizar correctamente el contenido de un documento XUL. Gracias a esto, las aplicaciones XUL son multiplataforma y multidispositivo, ya que siempre podremos encontrar un navegador Gecko para la plataforma y dispositivo deseados.

Uno de los principales atractivos de este lenguaje es que ofrece la posibilidad de crear complementos para el navegador Mozilla .

Por contra su principal desventaja es que no se podrá ejecutar en otros navegadores que, como Internet Explorer, no cumplan con esta característica.

Proceso de mapeado: Un documento XUL es interpretado para visualizar su contenido, en la renderización intervienen la definición de los elementos y las hojas de estilo. Es muy parecido a HTML, utiliza hojas de estilo para diseñar el aspecto y javascript para implementar el comportamiento de sus elementos. Para definir una interfaz se especifican tres grupos de componentes distintos:

- ✓ **Content:** Aquí se encuentran los documentos XUL, que definen el diseño de la interfaz. Incluye las etiquetas y la referencia a los documentos javascript.
- ✓ **Skin:** Contiene las hojas de estilos (**CSS**) y las imágenes, las cuales definen la apariencia de la interfaz.
- ✓ **Locale:** Los documentos **DTD** se encuentran aquí, estos documentos facilitan la localización de páginas XUL.

Cada uno de estos componentes se almacena en un archivo diferente, que puede ser editado.

Este es un ejemplo de archivo XUL, puedes encontrar un ejemplo un poco más extendido en este enlace:

[Ejemplo de XUL \(link: D102_CONT_R09_ejemplo_XUL.txt\).](#)

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window
  id="Principal:window"
  title="Ventana principal"
  orient="horizontal"
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/so.xul">
  <!-- El resto de elementos irán aquí -->
</window>
```

[\(link: D102_CONT_R10_Interfaz_XUL.png.\)](#)

2.3.- XAML.

Es un lenguaje de marcas empleado para la creación de interfaces en el modelo de programación .NET **Framework** de Microsoft. Las aplicaciones creadas podrán ejecutarse en entornos Windows.

```
<Page
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="EjemploPaginaEjemplo">
<Button Click="Accion_Click" >Haz Click</Button>
</Page>
```

Mediante XAML se pueden crear lo que se conoce como **RIA** (Rich Interface Applications) que contienen abundante material gráfico.

[.\(link: DI02_CONT_R11_Interfaz_XAML.png.\)](#)

XAML consta de una serie de elementos XML para representar los principales componentes gráficos, así como la distribución, paneles y manejadores de eventos. Puede hacerse programando directamente la interfaz mediante un editor de texto, o mediante el entorno de desarrollo gráfico incluido en la herramienta Expresión Blend. El código asociado a la interfaz es puramente declarativo, es decir, hace referencia tan solo al aspecto visual, pero no añade funcionalidad, salvo ciertas respuestas muy sencillas a interacciones con el usuario.

La realización de cálculos se añaden en un archivo independiente. Esto permite al equipo de desarrollo y al de diseño trabajar por separado, sin interferir mutuamente en su trabajo.

Proceso de mapeado: se realiza en tiempo de ejecución, los elementos de la interfaz se conectan con objetos del framework .NET y los atributos con propiedades de estos objetos para integrarlos en la aplicación. Para facilitar la traducción de XAML a código .NET se ha creado una relación para cada elemento XAML con una clase de la plataforma .NET.

Tienes el código del ejemplo en el siguiente documento:

[Documento con ejemplo para XAML. \(link: DI02_CONT_R12_ejemplo_XAML.txt \)](#)

Para saber más

Si quieres conocer algo más sobre XAML te sugerimos una página donde poder encontrar una introducción a XAML.

Página oficial de XAML. [.\(link: http://msdn.microsoft.com/es-es/library/ms752059.aspx \)](http://msdn.microsoft.com/es-es/library/ms752059.aspx)

Y también puedes visitar la siguiente página donde poder encontrar la referencia del lenguaje y ejemplos de aplicaciones dentro del ámbito del entorno de desarrollo Expression Blend.

Página de la MSDN de XAML. [.\(link: http://msdn.microsoft.com/es-es/library/cc295215.aspx \)](http://msdn.microsoft.com/es-es/library/cc295215.aspx)

2.4.- UIML.

Este lenguaje permite generar interfaces que son independientes de la plataforma y el lenguaje subyacente. Para generar una interfaz UIML se precisa crear un documento XML con la definición de la interfaz, utilizando los elementos de UIML y una hoja de estilos que traslade esa definición a la plataforma y lenguaje seleccionado, de esta forma para una aplicación concreta tan solo necesitaremos un documento UIML, y tantas hojas de estilo para la traducción como nos sean necesarias.

```
<UIML>
<HEAD>
<AUTHOR>María José Navascués</AUTHOR>
<DATE>1 Mayo, 2011</DATE>
<VERSION>1.0</VERSION>
</HEAD>
<APP CLASS="App" NAME="Dialog">
<GROUP CLASS="Dialog" NAME="DialogConBoton">
<ELEM CLASS="DialogMessage" NAME="Mensaje">
<ELEM CLASS="DialogButton" NAME="Boton_OK">
</GROUP>
</APP>
<DEFINE NAME="Boton_OK">
<PROPERTIES>
<ACTION
VALUE="Dialog EXISTS=false"
TRIGGER="Selected"
/>
</PROPERTIES>
</DEFINE>
</UIML>
```

Este sistema tiene como principal ventaja que solo se precisa un único diseño de la interfaz de la aplicación, independientemente del dispositivo donde será visualizado, con lo que evitamos el peligro de desarrollar interfaces para dispositivos que no estén en el mercado en el futuro.

[\(link: DI02 CONT R13 Interfaz UIML.png.\)](#)

UIML describe una interfaz en tres niveles: presentación, contenido y lógica.

La presentación se refiere a la apariencia de la interfaz; el contenido se refiere a los componentes de la interfaz y la lógica se refiere a la interacción usuario – interfaz (eventos del ratón, teclado...).

En UIML, una interfaz de usuario es una jerarquía de elementos XML. Cada uno de los componentes de una interfaz (botones, cajas de texto, etiquetas...) son una entidad XML. Estas entidades son elementos **Part**. Cada uno tiene asociado un elemento **Content**, que puede ser texto, imagen, etc.

Para definir el **comportamiento** de la interfaz se realiza el **mapeo** de las partes a los elementos correspondientes del lenguaje de implementación elegido y la conexión con la lógica de aplicación.

Estos bloques facilitan la separación entre los elementos que componen la interfaz, distinguiendo entre el modelado estructural, la visualización y el modelado del comportamiento.

El lenguaje UIML permite la traducción automática al lenguaje utilizado por el dispositivo final. El proceso de traducción se realiza en el propio dispositivo o en el servidor de la interfaz dependiendo del dispositivo del que se trate.

Tienes el código del ejemplo en el siguiente enlace:

[Ejemplo de UIML. \(link: DI02 CONT R14 ejemplo UIML.txt \)](#)

2.5.- XIML.

Es un lenguaje de desarrollo de interfaces cuyo objetivo es cubrir todo el ciclo de vida del software, incluyendo las fases de diseño, operación y evaluación, por lo tanto, además de proporcionar la infraestructura para poder diseñar una interfaz gráfica de usuario con cierto nivel de complejidad, también proporciona herramientas para atender a todo el proceso de desarrollo de la misma.



[./link: DI02_CONT_R15_Eschema_XIML.png./](#)

XIML trabaja con los elementos abstractos y concretos de una interfaz de usuario. Los elementos abstractos hacen referencia al contexto en el que se interacciona con la interfaz, y se representan en:

- ✓ **Tareas:** procesos o tareas de usuario que puede ejecutar la interfaz.
- ✓ **Dominio:** conjunto de objetos y clases con los que se opera desde la interfaz. Están distribuidos jerárquicamente.
- ✓ **Usuarios:** también se organizan jerárquicamente y representan usuarios o grupos de usuarios que hacen uso de los datos y procesos en las tareas.

Los elementos concretos hacen referencia a aquello que se representa físicamente en la interfaz, como los controles de formulario, y se representan en:

- ✓ **Presentación:** colección jerárquica de elementos que interaccionan con el usuario desde la interfaz, puede ser desde una ventana a un botón, pasando por controles más complejos como un control ActiveX.
- ✓ **Dialogo:** colección jerárquica de acciones de interacción que permiten al usuario comunicarse con los elementos de la interfaz.

Los elementos de la interfaz interaccionan a través de relaciones en las que se indica que elementos intervienen y el tipo de relación que hay entre ambos.

Proceso de mapeado: En este lenguaje se separa completamente la definición de la interfaz de la forma en que es renderizada. Precisa de un traductor que traslade la definición a código visible para cada dispositivo específico en el que se quiera usar la interfaz.

De esta forma se consigue un lenguaje completamente **multiplataforma**.

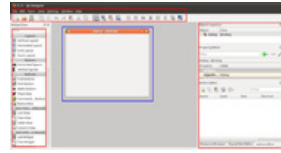
Autoevaluación

Texto de la pregunta: Los lenguajes de desarrollo de interfaces basados en XML...

- ☐ [\(link: \)](#)
...son lenguajes de programación de alto nivel.
- ☐ [\(link: \)](#)
...permiten desarrollar aplicaciones completas siempre sin necesidad de que intervengan otros lenguajes de programación.
- ☐ [\(link: \)](#)
...solo permiten generar aplicaciones para Internet.
- ☐ [\(link: \)](#)
...necesitan que, posteriormente, se mapeen los elementos XML a objetos que entienda el dispositivo final.

3.- Herramientas para la creación de interfaces multiplataforma.

Existen muchos tipos de software para la creación de interfaces de usuario. Habitualmente estas herramientas tienen en común que para generar interfaces gráficas usan un sistema de ventanas, las cuales permiten la división de la pantalla en diferentes regiones rectangulares, llamadas "ventanas" cuya parte central es el conjunto de herramientas (toolkit).



[.\(link: DI02_CONT_R17_Toolkit.jpg.\)](#)

Cuando creamos una aplicación con interfaces gráficas, en primer lugar hay que **diseñar la interfaz**, normalmente el **toolkit** contiene los objetos gráficos más empleados, tales como menús, botones, etiquetas, barras de scroll, y campos para entrada de texto que compondrán la interfaz, mientras que el sistema de ventanas provee de procedimientos que permiten dibujar figuras en la pantalla y sirve como medio de entrada de las acciones del usuario. En la imagen puedes apreciar los conjuntos de herramientas enmarcados en rojo (en la imagen cuadro derecho y barra de herramientas superior) y la zona de dibujo en azul (en la imagen cuadro central).

A continuación se **añade funcionalidad** a los elementos de la interfaz a través de una serie de procedimientos definidos por el programador o programadora. La función de estos procedimientos es el decidir la forma en que se comportarán los objetos gráficos.

Por último, se **conecta la interfaz generada con la aplicación de destino**. Esto se puede realizar de diferentes maneras. Si usamos un único lenguaje de programación para la interfaz y la funcionalidad, lo usual es contar con un entorno de desarrollo integrado común para todas las fases de desarrollo como al utilizar la biblioteca swing con Java. Sin embargo, cuando el lenguaje final es uno y el lenguaje de la interfaz otro, como es el caso de las tecnologías basadas en XML que estamos viendo, normalmente se requiere de un proceso de **traducción** de los elementos XML a elementos que entienda la plataforma final, como en el caso de XAML que traduce cada elemento XML a clases de la plataforma .NET. En el caso de XUL es el motor de renderización el que se encarga de hacer las transformaciones necesarios para visualizar la interfaz. Este proceso puede ser más o menos automático en función de las tecnologías seleccionadas.

A continuación veremos algunos ejemplos de herramientas existentes, tanto libres como propietarias, que difieren en la manera que tratan los archivos XML con las interfaces para incluirlos en la aplicación final.

Autoevaluación

Texto de la pregunta: Las herramientas para la generación de interfaces...

- ☐ [\(link: \)](#)
...son muy complicadas de aprender.
- ☐ [\(link: \)](#)
...no se integran nunca con otras herramientas de desarrollo de aplicaciones.
- ☐ [\(link: \)](#)
...se usan para editar la interfaz y poder modificarla en modo texto.
- ☐ [\(link: \)](#)
...permiten al programador o programadora concentrarse en los aspectos relativos al diseño.

3.1.- Presentación de algunas herramientas.

El **principal objetivo** de estas herramientas es ocultar la sintaxis de los lenguajes de modelado y proporcionarles una interfaz que permita especificar adecuadamente el modelo de interfaz en los tres aspectos que hemos visto, a saber:



[\(link: DI02_CONT_R18_esquema_uso_herramientas.png.\)](#)

La interfaz se almacena en un archivo de texto plano siguiendo las directrices del estándar XML.

Estas herramienta disponen de editores, intérpretes, generadores y otras aplicaciones útiles para llevar a cabo tareas relacionadas con la elaboración, manipulación o generación de modelos de interfaz.

Entre otras, con estas características podemos encontrar las siguientes aplicaciones:

- ✓ Libres:
 - ✓ QT Designer.
 - ✓ Glade.
- ✓ Propietarias:
 - ✓ Expression Blend de Microsoft.
 - ✓ Flex de Adobe.

Debes conocer

Aquí tienes las características más importantes de las herramientas mencionadas en el texto.

Resumen textual alternativo ([link: DI02_Descripcion_Presentacion_Herramientas_para_desarrollo_de_interfaces_basadas_XML.html](#))

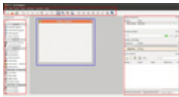





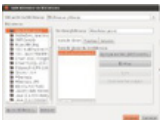









Autoevaluación










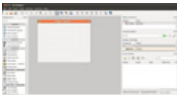
Texto de la pregunta: ¿Qué es una aplicación RIA ?

- ☐ ([link:](#))
Aquella que solo ejecuta en un terminal.
- ☐ ([link:](#))
Un programa con una interfaz de usuario compleja.
- ☐ ([link:](#))
Una aplicación para realizar operaciones bancarias.
- ☐ ([link:](#))
Aquella que se ejecuta en un navegador web y su aspecto y funcionalidad es el de una aplicación de escritorio.

Anexo.- Licencias de recursos.

Licencias de recursos utilizados en la Unidad de Trabajo

Recurso (1)	Datos del recurso (1)	Recurso (2)	Datos del recurso (2)
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: qt-jambi.org.</p> <p>Licencia: Creative Commons Attribution, 5</p> <p>Procedencia: http://qt-jambi.org/</p>
	<p>Autoría: Alex Coiro.</p> <p>Licencia: Creative Commons, share alike, atribution.</p> <p>Procedencia: http://commons.wikimedia.org/wiki/File:Recepci%C3%B3n_-_Hotel_Humboldt.JPG</p>		<p>Autoría: Pilar Acero López.</p> <p>Licencia: CC reconocimiento, compartir i</p> <p>Procedencia: Banco de imágenes del ITE http://recursostic.educacion.es/bancoim</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Captura de pantalla de la aplica Designer bajo licencia LGPL.</p> <p>Procedencia: Elaboración propia.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación NetBeans bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p>

	<p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: ZyMOS.</p> <p>Licencia: Dominio público.</p> <p>Procedencia: http://commons.wikimedia.org/wiki/File:windows_os.svg </p>
	<p>Autoría: gg3po.</p> <p>Licencia: Dominio público.</p> <p>Procedencia: http://commons.wikimedia.org/wiki/File:NewTux.svg </p>		<p>Autoría: android.com.</p> <p>Licencia: GNU.</p> <p>Procedencia: http://commons.wikimedia.org/wiki/File:uselang=es </p>
	<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la aplicación QT Designer bajo licencia LGPL.</p>		<p>Autoría: María José Navascués González.</p> <p>Licencia: Uso educativo no comercial.</p> <p>Procedencia: Captura de pantalla de la a Designer bajo licencia LGPL.</p>

