Elaboración de interfaces mediante documentos XML.

Caso práctico

Ministerio de Educación y Formación Profesional. (Elaboración propia)

En BK producciones han recibido el encargo de una cadena hotelera de desarrollar un proyecto de software para la gestión de sus hoteles.

En BK lo han recibido con entusiasmo puesto que se ajusta bastante bien a una idea de gestión de desarrollo en la que están trabajando hace algún tiempo, emplear lenguajes basados en XML para desarrollar las interfaces de los programas. En concreto **Ada** quiere analizar si es más rentable para el equipo tener dos grupos de desarrollo, uno encargado de crear las interfaces y otro que programe las funciones que debe implementar el software, o realizar todo el proceso de forma continuada.

Para ello sabe que, en primer lugar, tendrá que poner a su equipo a analizar y recordar las características del lenguaje XML, para poder trabajar con toda la información sobre el tema. Después tendrán que estudiar los lenguajes para la creación de interfaces basados en XML más conocidos y ver cuál se adapta mejor a sus necesidades, así como las herramientas existentes en el mercado para, finalmente, seleccionar algún lenguaje y herramienta concretos para poderse manos a la obra.

1.- Lenguajes basados en XML.

Caso práctico

Tal y como tenía planeado, **Ada** reúne a su equipo para comunicarle sus planes. El primer punto a tratar es hacer un pequeño estudio del origen y características del lenguaje XML que permita a su equipo trabajar con los documentos de interfaz que tendrán que generar. El equipo debe estar preparado para poder editarlos y modificarlos, por lo que deben conocer bien cuál es la estructura interna de un documento XML y las reglas para su creación.

Ministerio de Educación y Formación Profesional.

Ada pone a su equipo manos a la obra, y comienzan a buscar información en Internet y libros especializados, llegando a estas conclusiones...

¿Qué es XML?

XML (eXtensible Markup Languaje 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.

Ministerio de Educación y Formación Profesional. (Elaboración propia)

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. Goldfab, por encargo de IBM desarrolló el lenguaje GML, Generalized Markup Languaje, 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. Golfab, que lo desarrolló por encargo de IBM."
- Conseguiré algo parecido a esto: El padre del lenguaje **GML** fue **Charles F. Golfab**, 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.

XMLsin 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.

XML nace con el objetivo de convertirse en un estándar para el intercambio de información estructurada entre diferentes plataformas: Windows, Android, Linux, iOS, etc. Se puede utilizar en bases de datos, editores de texto, cálculos, dispositivos móviles, etc... Actualmente es la base para la creación de interfaces gráficas de muchos generadores de códigos o IDEs como Visual Studio, Android Studio, NetBeans y Eclipse.

Reflexiona

XML no incluye ninguna orientación al diseño. Esto quiere decir que no vamos a hacer referencia a ningún aspecto relativo a la aplicación final en la que visualicemos el documento. Entonces ¿cómo crees que podemos usar el mismo archivo para ver en un navegador que en un teléfono móvil? ¿Qué necesitaremos para visualizar ese contenido de formas diferentes?

Mostrar retroalimentación

Efectivamente, los archivos XML se pueden visualizar en diferentes plataformas, como navegadores web o teléfonos móviles, para salvar las diferencias se debe acompañar de un archivo auxiliar, que interpreta cómo debe visualizarse el contenido XML en cada dispositivo. Este archivo se denomina XSL y contiene la definición de cómo se verán los elementos XML en un dispositivo concreto.

Autoevaluación

¿Cuál de las siguientes afirmaciones es correcta?

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

No es correcta, se consideraba este lenguaje difícil de aprender y de usar por lo que no tuvo mucho auge.

No es correcta, precisamente el principal objetivo de XML es hacer independiente la estructura del documento de la aplicación final donde será visualizado.

No es la respuesta correcta porque con HTML podemos hacer referencia a cuestiones de diseño del documento final, y este aspecto no se contempla en un documento XML.

Es correcta ya que, a pesar de estar basado en XML, HTML si hace referencia a cuestiones relativas al diseño de la página web como por ejemplo el formato de la fuente o el párrafo.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

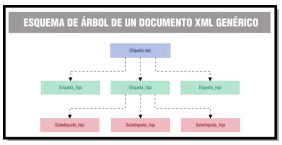
Para saber más

Puedes encontrar un completo artículo sobre XML en este enlace:

Artículo sobre XML.

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.



Ministerio de Educación y Formación Profesional.
(Elaboración propia)

> Las etiquetas están delimitadas 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
<Etiqueta_raiz> <etiqueta_hija> <subetiqueta_hija>...</subetiqueta_hija> <subetiqueta_hija>...</subetiqueta_hija> ... </etiqueta_hija> </Etiqueta_raiz>	 Ministerio de Educación y Formación Profesional. (Elaboración propia)

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>	 Ministerio de Educación y Formación Profesional. (Elaboración propia)

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:

INTEFP (CC BY-NC-SA)

```
<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?

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

No es correcta porque los nombres de las etiquetas no pueden contener espacios en blanco.

No es correcta porque los valores de los atributos deben ir entrecomillados siempre.

No es correcta porque no cumple con el anidamiento de etiquetas conforme a la estructura de árbol del documento.

Correcta.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

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

Caso práctico

Una vez que el equipo conoce las bases del lenguaje XML, **Ada** les recomienda que hagan una búsqueda de los lenguajes más empleados en la elaboración de interfaces mediante documentos XML, que aprendan cuáles son sus bases, que tiene en común y también sus diferencias.

La creación de aplicaciones de calidad requiere de la intervención de múltiples disciplinas además de un gran esfuerzo de programación, lo que nos interesa, fundamentalmente, es observar la posibilidad, usando la técnica de desarrollo de interfaces con lenguajes XML, de separar el proceso de elaboración de la interfaz de la programación de la funcionalidad de un proyecto software. Así es posible dentro de un mismo equipo realizar ambas tareas de forma simultánea, lo que supone un ahorro de tiempo considerable, si lo planificamos bien podremos crear interfaces muy elaboradas para nuestras aplicaciones sin que nos suponga mucho esfuerzo extra y sin que influya en exceso en el trabajo de programación.

Ministerio de Educación y Formación Profesional. (Elaboración propia)

El desarrollo de interfaces de usuario es una actividad compleja que requiere de la intervención de múltiples factores. El desarrollador o desarrolladora se enfrenta a una serie de hitos que debe superar:

Ministerio de Educación y Formación Profesional. (Elaboración propia)

- En primer lugar, se debe definir lo que se espera de la interfaz, que requisitos debe cubrir y el entorno en el que se integra.
- También se debe tener en cuenta la variedad de herramientas y lenguajes de programación que exigen un elevado nivel de conocimientos.
- Por otra, parte necesitamos interfaces (así como las aplicaciones subyacentes) para múltiples contextos de uso, que dependen del usuario (por su idioma, preferencias o posibles discapacidades), de la plataforma (ordenadores personales, móviles, miniportátiles, pantallas, etc.), del modo de presentación de la interfaz (gráfica o texto) o del dispositivo final (pantallas de ordenador, pizarras digitales, proyectores, etc.).

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.

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.

Reflexiona

El uso de tecnologías XML permite separar completamente la creación de la interfaz de la programación de la aplicación subyacente. ¿Qué ventajas crees que aporta esto? ¿Puede mejorar el procedimiento de desarrollo de aplicaciones con interfaces gráficas complejas?

Mostrar retroalimentación

De hecho, así es, utilizar tecnologías XML para la creación de interfaces tiene como principal ventaja que, después de planificar adecuadamente qué esperamos de la aplicación y qué información debe proporcionarle la interfaz, el equipo puede dividirse totalmente el trabajo de desarrollo, ya que ambas tareas pueden realizarse completamente por separado con el consiguiente ahorro de tiempo. Además, las modificaciones que se pueden realizar influyen mucho menos en la aplicación global y suelen afectar tan solo al ámbito que corresponde. Por otra parte, al utilizar lenguajes y herramientas específicas, normalmente, se consiguen interfaces más elaboradas ya que se pone a nuestra disposición elementos concretos de este ámbito de la programación.

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?

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

No es correcta ya que el desarrollo de interfaces está íntimamente relacionado con el usuario final y debe hacerse de acuerdo a sus necesidades.

No es correcto porque la adaptación de la interfaz a diferentes dispositivos depende del lenguaje, si el lenguaje no es adecuado para, por ejemplo, un teléfono móvil no podremos usar esa interfaz en una aplicación para móviles.

No es correcta ya que las interfaces descritas con lenguajes basados en XML deben sufrir un proceso de transformación para integrarse en la aplicación final.

Correcto, se puede dividir el trabajo en dos grupos, uno que genere los documentos con todas las interfaces y otro que desarrolle toda la funcionalidad, dejando la integración para el final del proyecto.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

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 Firefox, de hecho, la interfaz de ambos está desarrollada en XUL. Este navegador funciona 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:

```
<?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/solo.xul">
<!-- El resto de elementos iran aqui -->
</window>
```

María José Navascués González (Elaboración propia)

- **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.

XUL nos proporciona una gran cantidad de componentes para crear una interfaz gráfica entre las que destacan:

- Controles de entrada como TextBox, CheckBox, etc.
- Barra de herramientas con otro contenido o botones.
- Menús de barras de diálogos o menús y diálogos emergentes.
- Árboles, Atajos del teclado.
- Scripts y eventos.
- Overlay, elemento creado para Firefox.

Los **tipos de aplicaciones XUL** son:

- **Extensiones para Firefox**: barras de herramientas, menús u otros documentos XUL que agregan funcionalidades a Firefox.
- **Aplicaciones independientes**: son creadas mediante XULRunner. No es necesario el navegador para ejecutarlas.
- **Paquete XUL**: son aplicaciones que se instalan como extensiones de Firefox, pero actúan como una aplicación independiente de Firefox.
- **Aplicaciones XUL** Remotas: aplicaciones que van a ser instaladas en servidores Web y son ejecutadas de forma remota.

Para saber más

Página oficial de XUL donde poder encontrar la referencia del lenguaje, ejemplos de aplicaciones. Cabe destacar el enlace a la aplicación XULRunner que te permitirá crear y ejecutar aplicaciones XUL.

Página oficial de XUL.

En este enlace, encontrarás un pequeño tutorial, muy sencillo de seguir, para ver cómo funciona una aplicación web muy básica escrita en XUL.

Mi primera aplicación XUL.

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="EspacioEjemplo.PaginaEjemplo">
  <Button Click="Accion_Click" >Haz Click!</Button>
</Page>
```

María José Navascués González (Elaboración propia)

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

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.

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.

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="Dialogo">
  <GROUP CLASS="Dialog" NAME="DialogoConBoton">
    <ELEM CLASS="DialogMessage" NAME="Mensaje"/>
    <ELEM CLASS="DialogButton" NAME="Boton_OK"/>
  </GROUP>
</APP>
<DEFINE NAME="Boton_OK">
  <PROPERTIES>
  <ACTION
    VALUE="Dialogo.EXISTS=false"
    TRIGGER="Selected"
  />
  </PROPERTIES>
</DEFINE>
</UIML>
```

María José Navascués González (Elaboración propia)

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.

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, 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.

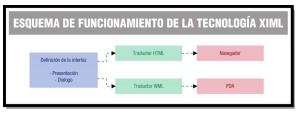
Para saber más

Aquí tienes un enlace donde poder encontrar la especificación del lenguaje, tutoriales y ejemplos de aplicaciones.

Enlace a la página oficial de UIML.

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.



Ministerio de Educación y Formación Profesional. (Elaboración propia)

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...

- ○ ...son lenguajes de programación de alto nivel.
- ○ ...permiten desarrollar aplicaciones completas siempre sin necesidad de que intervengan otros lenguajes de programación.
- ○ …solo permiten generar aplicaciones para Internet.
- ○ ...necesitan que, posteriormente, se mapeen los elementos XML a objetos que entienda el dispositivo final.

Falso. Los lenguajes basados en XML se emplean para definir qué elementos gráficos participan en la interfaz, dónde se visualizan, y, normalmente, permiten indicar cierta funcionalidad muy limitada, pero no van a generar un programa de aplicación por sí solos.

No es correcto. El código de la interfaz se suele transformar para poder integrarse con el código de la aplicación final en un lenguaje de programación de alto nivel.

No es correcta ya que con esta técnica se pueden generar interfaces de usuario para todas aplicaciones finales, independientes del lenguaje y del dispositivo final.

Es correcto, normalmente el archivo XML sufre un proceso de análisis antes de ser renderizados o integrados en una aplicación.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

2.6.- SVG (Scalable Vector Graphics).

Los gráficos vectoriales redimensionales o SVG son una especificación para describir gráficos vectoriales bidimensionales. Estos gráficos pueden ser animados o estáticos. Poseen un formato XML. Actualmente la mayoría de los navegadores soporta SVG.

Hay que destacar que las imágenes vectoriales pueden ser redimensionadas sin perder calidad. No ocurre lo mismo con las imágenes en mapa de bits.

Harvey Rayner

SVG permite tres tipos de gráficos:

- Elementos geométricos vectoriales: líneas, círculos, rectas, etc.
- Imágenes en mapa de bits.
- Textos.
- Los objetos gráficos pueden ser agrupados, transformados y compuestos en objetos previamente renderizados. Se les puede aplicar los mismos estilos a cada grupo.

Para saber más

Si quieres profundizar más en SVG lo puedes hacer a través del siguiente enlace: Información adicional sobre el formato SVG

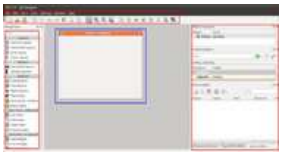
3.- Herramientas para la creación de interfaces multiplataforma.

Caso práctico

Ada y su equipo ya tienen una idea general de cómo emplear un lenguaje para la elaboración de interfaces basado en XML, y diferentes formas de mapear el documento obtenido para visualizar la interfaz. De hecho, a estas alturas ya pueden empezar a programar una interfaz en modo texto. Sin embargo, para Ana que es una recién llegada, aún resulta un poco difícil crear estos documentos con una sintaxis tan específica en modo texto, así que decide comentarlo con su jefa, ya que piensa que merece la pena dedicar un poco de tiempo a estudiar las herramientas existentes en el mercado, tanto libres como propietarias, para el desarrollo de interfaces basadas en XML utilizando un entorno gráfico, pensando en el beneficio que va a suponer el ahorro de tiempo al emplear una herramienta que genere todo el código de la interfaz sin más que hacer unos cuantos clics.

Ministerio de Educación y Formación Profesional. (Elaboración propia)

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).



María José Navascués González (Elaboración propia)

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). En algunas aplicaciones denominan **Widgets** a los controles o elementos que podemos añadir a una interfaz.

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 necesarias 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...

- ○ ...son muy complicadas de aprender.
- ○ ...no se integran nunca con otras herramientas de desarrollo de aplicaciones.
- ○ ...se usan para editar la interfaz y poder modificarla en modo texto.
- ○ ...permiten al programador o programadora concentrarse en los aspectos relativos al diseño.

No es correcta, aprender a manejar estas herramientas suele ser un proceso intuitivo y bastante sencillo debido a su estructura de ventanas y a que la construcción de la interfaz se basa en el procedimiento arrastrar y soltar.

No es correcto, hoy día encontramos entornos integrados de desarrollo que incorporan todas las herramientas necesarias para la construcción de aplicaciones en todas las fases de desarrollo.

No es correcto, el uso habitual de estas herramientas es en modo gráfico.

Correcta, al abstraernos de la sintaxis y aspectos internos de cada lenguaje de modelado conseguimos centrarnos en el diseño de la interfaz.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

3.1.- Presentación de algunas herramientas.

Las herramientas de diseño de interfaces gráficas de usuario forman parte de las herramientas RAD (Rapid Application Development, Desarrollo rápido de Aplicaciones).

Ministerio de Educación y Formación Profesional.
(Elaboración propia)

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:

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

Estas herramientas, 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.
  - Scene Builder
- Propietarias:
  - Expression Blend de Microsoft.
  - Flex de Adobe.

Las aplicaciones RIA (Rich Internet application), "aplicación de Internet enriquecida" o "aplicación rica de internet", es una aplicación web que tiene la mayoría de las características de las aplicaciones de escritorio

Para saber más

Si tienes curiosidad por saber algo más de estas herramientas aquí tienes lo enlaces que te llevarán a sus páginas oficiales:

Página oficial de QT Creator. Página oficial de GLADE. Página oficial de FLEX.

Página oficial de Microsoft Expression Blend. Página de MSDN para Expressión Blend.

Autoevaluación

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

- ○ Aquella que solo ejecuta en un terminal.
- ○ Un programa con una interfaz de usuario compleja.
- ○ Una aplicación para realizar operaciones bancarias.
- ○ Aquella que se ejecuta en un navegador web y su aspecto y funcionalidad es el de una aplicación de escritorio.

No es correcta, no tiene nada que ver.

No es correcto, no son programas independientes.

No es correcto, te has despistado.

Correcta, por eso es tan importante la creación de interfaces multiplataforma.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

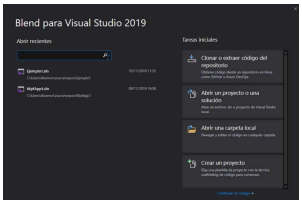
4-. Blend para Visual Studio.

Blend para Visual Studio nos permite construir el diseño de aplicaciones para la Tienda de Windows, Windows Phone, WPF y Silverlight. Actualmente se instala junto con Visual Studio.

Puedes descargar Visual Studio 2019 desde la página de Microsoft a través del siguiente enlace:

Descargar Visual Studio 2019

Actualmente podemos descargar 3 versiones de Visual Studio. Te recomendamos la versión Community que es gratuita para estudiantes y trae todo lo necesario para poder comenzar a trabajar con Blend Studio.



Montaña Martín Vergel (Elaboración propia)

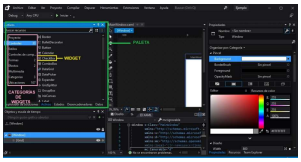
Para comenzar a trabajar, necesitaremos crear un proyecto. Para ello, podemos utilizar plantillas existentes que nos facilitan el diseño de la aplicación.

Montaña Martín Vergel (Elaboración propia)

4.1.- Controles. Contenedores.

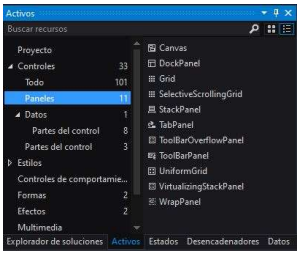
Los controles que podemos utilizar los podemos encontrar en la Paleta. Se muestra en el lado izquierdo y organiza los widgets por categorías. Las categorías que podemos encontrar son:

Proyecto, Controles, Estilos, Controles de componentes, Formas, Efectos, Multimedia, Categorías y Ubicaciones.



Montaña Martín Vergel (Elaboración propia)

Los widgets contenedores están situados en la paleta dentro del menú Controles -> Paneles

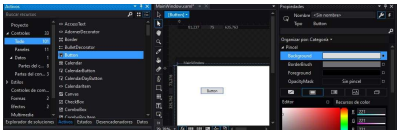


Montaña Martín Vergel (Elaboración propia)

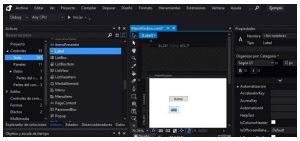
4.2.- Controles y propiedades.

Los controles más utilizados en Blend Studio son: Cuadros de textos o entrada de texto, etiquetas, botones, opciones, listas despegables y casillas de verificación.

- Entrada de texto (Text Entry o TextBox): nos permiten introducir datos en nuestras aplicaciones, aunque también es posible mostrar información a los usuarios en tiempo de ejecución a través de ellos.
- Etiqueta (Label): sirven para mostrar por pantalla texto.
- Botón (Button): nos permite que la aplicación ejecute un conjunto de acciones cuando el usuario los seleccione.
- Opciones (Radio /RadioButton): permite indicar varias opciones para que el usuario solo seleccione una.
- Casillas de verificación: permite presentar una o varias opciones. El usuario podrá seleccionar una o varias con independencia de la que seleccionemos.
- Lista despegable: nos permite mostrar un conjunto de valores para que el usuario seleccione uno de ellos.

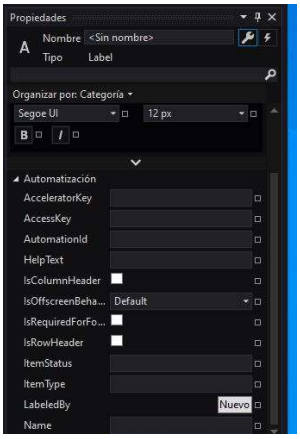


Montaña Martín Vergel (Elaboración propia)



Montaña Martín Vergel (Elaboración propia)

La mayoría de los controles, tienen una serie de propiedades: el color, la ubicación, los estilos que se le aplican, el tamaño, etc.



Montaña Martín Vergel (Elaboración propia)

5.- Glade.

Glade (o Glade Interface Designer) Diseñador de interfaces Glade, es una herramienta de desarrollo de interfaces gráficas mediante GTK/GNOME. Es independiente del lenguaje de programación y no genera código fuente sino un archivo XML.

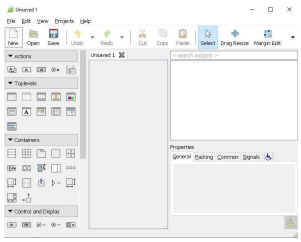
Las herramientas de Glade son implementadas como widgets, Esto permite que se puedan integrar más fácilmente en los IDE's.

Glade utiliza un tipo de formato XML denominado GtkBuilder para almacenar los elementos de las interfaces diseñadas. Estos archivos pueden emplearse para construir la interfaz en tiempo de ejecución.

Debes conocer

Podemos descargar e instalar Glade desde el siguiente enlace: Descarga de Glade (SO Windows). Descarga de Glade (Linux)

La instalación es sencilla ya que al iniciarla se nos presenta la licencia de la aplicación que debemos de aceptarla. A continuación, nos aparece la ruta o path en donde vamos a instalar la aplicación. Procederemos a pulsar el botón Install y habremos terminado la instalación.

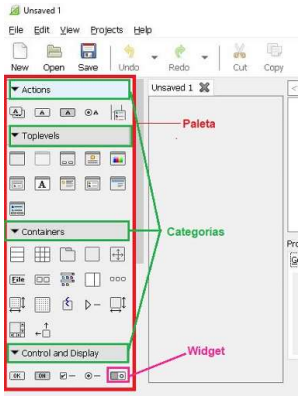


Montaña Martín Vergel (Elaboración propia)

.

5.1.- Paletas y vistas.

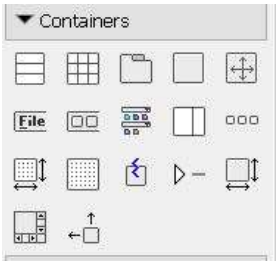
Glade utiliza Widgets o controles para construir el diseño de la interfaz. Estos se encuentran situados en la paleta. Al acceder a la aplicación, la paleta se encuentra situada en el lado izquierdo y contiene los Widgets organizados por categorías. Las categorías son: Actions, TopLevels, Containers, Color and Display, Composite Widgets y Miscellaneous.



Montaña Martín Vergel (Elaboración propia)

5.2.- Contenedores de controles.

Para organizar los controles o Widget dentro del proyecto podemos utilizar los componentes denominados cajas o contenedores (Containers). En la siguiente imagen se puede apreciar cuales son:



Montaña Martín Vergel (Elaboración propia)

Al utilizar los contenedores horizontales y verticales, Glade te preguntas cuántas filas o columnas queremos añadir. Este número posteriormente puede ser modificado ya que nos permite eliminar filas o columnas e insertar filas y columnas.

Una vez añadido todos los contenedores que necesitemos, podemos añadir los widgets que vayamos a utilizar en su interior. Estos elementos pueden ser cortados, copiados, pegados o borrados desde el menú *Edit* o seleccionando el botón derecho del ratón sobre ellos.

> Cada Widget tendrá un nombre único identificativo dentro del proyecto.

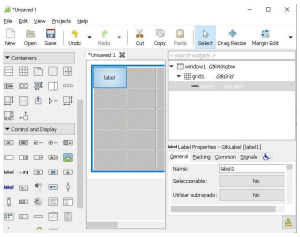
5.3.- Controles y propiedades. Alineación, tamaño y ubicación de los controles.

La mayoría de los controles poseen una serie de propiedades que pueden ser modificadas. Por ejemplo, el tamaño, el color, la posición, etc...

Los controles que más se utilizan son:

- **Entrada de texto (Text Entry o TextBox)**: nos permite introducir información a la aplicación, aunque también es posible utilizarlo para mostrar información a través de ellos.
- **Etiqueta (Label)**: nos permite modificar texto a través de ella y en tiempo de ejecución no es posible ser modificada.
- **Botón (Button)**: nos permite insertar un botón. Tienen la característica de que se pueden insertar un texto indicando la acción que se realizará al presionarla.
- **Opciones (Radio / RadioButton):** nos permite seleccionar un valor entre posibles valores.
- **Casillas de verificación (Check / CheckBox)**: Se utilizan para representar valores que pueden ser o no seleccionados por los usuarios. Ejemplo típico: la casilla que se muestra cuando tenemos que aceptar las condiciones de la licencia de una aplicación en su proceso de instalación.
- **Lista Desplegable (Combo / Select / Combox)**: típica lista que nos presenta un conjunto de valores para seleccionar uno. Por ejemplo, podemos mostrar un desplegable con las provincias españolas para que el usuario seleccione en la que resida.

Al seleccionar el objeto en la parte de la derecha nos aparece el panel de propiedades (Properties):



Montaña Martín Vergel (Elaboración propia)

Podemos colocar los controles un cualquier lugar de la interfaz de la misma forma que podemos variar su tamaño y posición.

En Glade, para colocar un control solo hay que arrastrarlo hacia el lugar del contenedor en donde queramos colocarlo. Por lo tanto, el tamaño, la posición y la alineación del control estará condicionado a la distribución de los contenedores.

6.- Scene Builder.

La aplicación Scene Builder pertenece a la compañía Gluon. Se trata de una aplicación que nos permite construir la interfaz de una aplicación independientemente del código de programación. El código asociado a la creación de la interfaz se almacena en un fichero XML con el formato FXML,

Es una aplicación gratuita y de código abierto. Existen versiones para los sistemas operativos Windows, Linux y Mac OS.

Se suele utilizar junto con aplicaciones JavaFX y se puede integrar en IDE's como Eclipse o NetBeans.

Se utiliza de forma fácil ya que permite arrastrar y soltar los elementos que queremos que aparezcan en las interfaces.

Para utilizarlo es necesario descargarlo e instalarlo. Lo podemos hacer desde el siguiente enlace: Descargar Scene Builder

Montaña Martín Vergel (Elaboración propia)

Dentro de las librerías que nos aparecen en el panel situado a la izquierda tenemos las siguientes categorías: containers, controls, Gluon, Menú, Miscellanous, shapes, Charts y 3D.

- **Containers**: el primer elemento que introduciremos para poder añadir el resto de elementos sobre el escenario.
- **Controls**: agrupa los controles como botones, casillas de verificación, desplegables, etc.
- **Gluon**: agrupan controles personalizados por la empresa Gluon y que reutilizan en las aplicaciones de la compañía.
- **Menú**: agrupa los controles como barra de menús,
- **Miscellanous**: agrupa diversos controles: escenario, canvas, tooltip, etc.
- **Shapes**: nos permite utilizar herramientas de dibujo con círculos, textos, polígonos, líneas, etc.
- **Charts**: agrupa controles para añadir gráficos.
- **3D**: herramientas para modificar las perspectivas de los objetos, luminosidad, etc.

7.- Eventos. Secuencia de eventos.

En la programación dirigida por eventos, tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema. En este tipo de programas serás el usuario quien determine el flujo de la aplicación en función de lo que vaya accionando en el programa.

El programador debe de definir los eventos que manejarán sus aplicaciones y la acción que se realizará al producirse cada uno de ellos, lo que se conoce como **administrador de eventos.**

Cuando ejecutamos una aplicación dirigida por eventos, al comienzo de ella, se producirán las inicializaciones correspondientes y a continuación el programa quedará inactivo hasta que se produzca un evento.

La programación dirigida por eventos es la base de lo que se denomina interfaz de usuario.

Los pasos para la construcción de una aplicación dirigida a eventos son:

1-. Escribir una serie de subrutinas llamados rutinas controlador de eventos que se encargarán de controlar los eventos a los que el programa principal responderá. Actualmente, muchos entornos de programación suministran plantillas de eventos para que el programador solo tenga que introducir su código asociado.

2-. Se debe de enlazar los controladores de eventos a los eventos para que se ejecute el código cuando se genere el evento.

3-. Deberemos de definir un bucle principal. Este bucle se encargará de comprobar la ocurrencia de los hechos y a continuación llamará al controlador de evento correspondiente para procesarlo. Actualmente los entornos de programación facilitan la creación de este bucle. Esto facilitas el trabajo al programador.

8.- Análisis y edición de documentos XML.

Siempre que trabajemos con documentos XML es conveniente su validación. La validación consiste en la comprobación de que un documento XML se encuentre bien formado y se ajusta a una estructura bien definida. Un documento bien formado sigue las reglas básicas de XML establecidas para el diseño de documentos. Un documento válido, además, respecta las normas indicadas en el fichero DTD (definición tipo de documento) al que se debe de encontrar asociado.

La validación consiste en:

1-. La corrección de los datos: permite detectar valores nulos o fuera de rangos.

2-. La integridad de los datos: se comprueba que toda la información obligatoria se encuentre presente en el interior del documento.

3-. El entendimiento compartido de los datos: se comprueba que el emisor y el receptor van a recibir el documento de la misma forma y que lo puedan interpretar igual.

Para la creación de documentos XML existen herramientas que evitan que se introduzcan datos erróneos, errores tipográficos, sintácticos y de contenidos. Estas herramientas se conocen con el nombre de Editores XML.

Los editores de XML tienen que ser capaces de:

1-. Leer la DTD correspondiente al documento y presentar una lista despegable con los elementos disponibles enumerados en la DTD, evitando la inclusión de algún elemento incluido en el esquema.

2-. Advertir del olvido de una etiqueta obligatoria e incluso no permitir este tipo de descuido o errores, No permite la finalización del documento si existen errores.

Debes conocer

Existen editores de XML que ocultan el código del documento y lo presentan en un formato más fácil de utilizar.

Ejemplos de editores XML son:

Gratuitas:

XMLPad de WMHelp. Lo puedes descargar desde el siguiente enlace: XMLPAD

XML Copy Editor. Posee licencia GNU. Lo puedes descargar desde el siguiente enlace: XML Copy Editor

XML Explorer: Lo puedes descargar desde el siguiente enlace: XMLExplorer

Versiones gratuitas de software comercial con periodos de evaluación:

XMLSpy. Lo puedes descargar desde el siguiente enlace: Altova XMLSpy

EditiX XML Editor. Lo puedes descargar desde el siguiente enlace: EditiX XML Editor

File Viewer Plus 3. Lo puedes descargar desde el siguiente enlace: File Viewer Plus 3

9.- Código para diferentes plataformas.

Los documentos XML son procesados a través de analizadores que leen el documento, lo interpretan y genera una salida en base a su contenido. El resultado lo mostrará, por ejemplo, en un dispositivo, en una ventana de un navegador, en una impresora, etc.

> Se denomina **parsing o análisis-sintáctico** al hecho de que una aplicación reciba un fichero XML, lo cargue en memoria y lo procese para obtener unos resultados. A este tipo de aplicaciones se les denomina parsers o analizadores (léxicos-sintácticos).
>
> Se dice que el **parsing XML** es el proceso mediante el cual se lee y se analiza un documento XML para comprobar que está bien formado para pasar su contenido a una segunda aplicación para su procesamiento,

Existen dos tipos de analizadores XML:

DOM (parsers DOM): es un metalenguaje con una estructura jerárquica. Las marcas dentro del documento tienen una relación padre-hijo. Es decir, posee una estructura en forma de árbol. Cada uno de los elementos del documento recibe el nombre de nodo, Estos nodos serán: elementos, atributos, comentarios, etc.

SAX: el API de SAX especifica como ha de comportarse el parser, es decir, especifica interfaces de programación y no clases. SAX se encuentra compuesto por dos interfaces: el XMLReader que representa el parser y el ContentHandler que recibe los datos del parser.

10.-Ejemplo de desarrollo de una interfaz utilizando XML.

Caso práctico

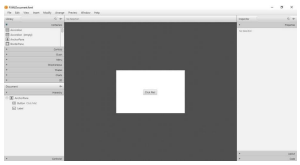
Finalmente, el equipo ha decidido optar por la Scene builder para crear la aplicación de gestión hotelera. Esta herramienta es muy útil, gratuita y de código abierto, que para lo que ellos quieren hacer, genera interfaces que pueden alcanzar gran complejidad, y facilita la inclusión de cierto código, común en todas las interfaces gráficas, como cerrar una ventana al pulsar un botón, o limpiar un formulario, directamente desde el editor de la interfaz.

La empresa BK Programación trabaja con el lenguaje java, por ello, utilizarán Scene builder integrado con NetBeans. Para ello, crearán sus proyectos basándose en la tecnología JavaFX. Creado por Oracle, permite la creación de Rich Internet Applications (RIAs), esto es, aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas. Las tecnologías incluidas bajo la denominación JavaFX son JavaFX Script y JavaFX Mobile, además de otros productos.

Scene Builder es una aplicación de diseño visual que permite a los usuarios diseñar rápidamente interfaces de usuario sin codificación. Los usuarios pueden arrastrar y soltar componentes de la IU a un área de trabajo, modificar sus propiedades, aplicar hojas de estilo y el código que genera lo almacena en formato FXML. El resultado es un archivo FXML que luego se puede combinar con un proyecto Java al vincular la interfaz de usuario a la lógica de la aplicación.

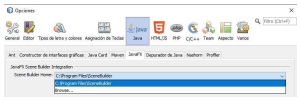
Debes conocer

Para utilizar Scene Builder es necesario tenerlo instalado en el equipo. Puedes descargarlo desde el siguiente enlace: Descargar Scene Builder



Montaña Martín Vergel (Elaboración propia)

Para modificar los ficheros FXML a través de la aplicación Scene Builder desde NetBeans, es necesario modificar la configuración de NetBeans. Para ello, accedemos a la opción Herramientas --> Opciones y seleccionar la pestaña JavaFX. En el apartado Scene Builder Home deberemos de indicar la ruta absoluta en donde se encuentre instalada la aplicación Scene Builder.



Montaña Martín Vergel (Elaboración propia)

Una vez seleccionado pulsaremos Aplicar y Aceptar. Una vez hecho esto, ya estamos en disposición de crear proyectos JavaFX desde NetBeans, utilizando la aplicación Scene Builder para crear la interfaz de la aplicación.

10.1.- Descripción del problema.

En primer lugar, hay que destacar que en estos materiales se pretende dar un vistazo rápido al funcionamiento de una herramienta para la creación de interfaces de usuario usando XML, así como la posterior integración de esta interfaz en una aplicación. Por lo tanto, no podrá considerarse esta unidad como un manual completo de aprendizaje de JavaFX Scene Builder, sino como ejemplo didáctico de la materia que se pretende trasladar al alumnado.

Para ilustrar el uso de la herramienta JavaFX Scene Builder y la inclusión de los archivos de interfaz generados en una aplicación Java vamos a desarrollar una pequeña parte de una aplicación de **gestión de un hotel**, en concreto, haremos una interfaz muy sencilla para añadir una reserva de un cliente. Tendremos que escribir los datos personales del cliente y los datos de la reserva, que incluyen la fecha de entrada y salida, número de personas que se van a alojar en el hotel, tipo de habitación, pudiendo elegir entre doble de uso individual, doble, junior suite y suite, y si la habitación será para fumadores o no fumadores. En caso de que se seleccione una habitación para fumadores aparecerá un mensaje indicando que no está permitido fumar en ninguna otra zona del hotel.

Alex Coiro (GNU/GPL)

El ejemplo se ha adaptado para poder incluir un mayor número de widgets en la interfaz y observar su funcionamiento. Esto nos permitirá usar los controles más básicos, como cuadros de texto, listas desplegables, o botones, y también alguno más elaborado como los calendarios. Para completar un poco el ejemplo crearemos una ventana principal extremadamente sencilla que permita ver como abrir esta interfaz desde un menú.

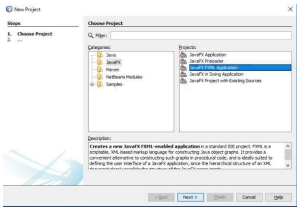
No vamos a implementar la funcionalidad subyacente a la interfaz, es decir, añadir la reserva a la base de datos del hotel, esto se deja para otros módulos del ciclo.

Reflexiona

Las interfaces de ejemplo son muy genéricas. Ten en cuenta que según el tratamiento que demos finalmente a la interfaz podremos obtener una aplicación de escritorio tradicional, una aplicación web o una aplicación para móviles. Sin embargo, la interfaz a generar será siempre la misma y podremos usar el mismo archivo XML en los tres casos.

10.2.- Creación proyecto JavaFX.

Comenzaremos por crear en NetBeans un nuevo proyecto JavaFX. Para ello, seleccionamos Archivo --> Nuevo proyecto -->JavaFX --> JavaFX FXML Application:

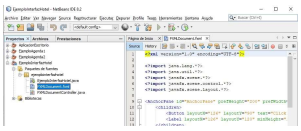


Montaña Martín Vergel (Elaboración propia)

A continuación, introducimos el nombre del proyecto, por ejemplo, EjemploInterfazHotel y pulsamos ***Terminar***.

Si nos fijamos en los ficheros que nos ha creado al generar el proyecto, podremos observar que nos ha creado tres ficheros entre los que destaca el fichero con extensión fxml. Si lo observamos podemos ver que en su interior aparece código XML. A través del código XML podremos configurar la interfaz de la aplicación. Por defecto, al generar una nueva aplicación JavaFX, nos inserta en la aplicación un contenedor para albergar un botón que contiene la etiqueta o el texto Púlsame.

En la siguiente imagen puede verse el proyecto generado y los ficheros que contiene:



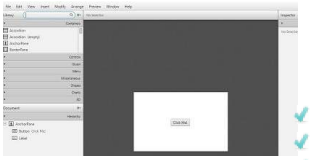
Montaña Martín Vergel (Elaboración propia)

Si ejecutamos la aplicación, se nos mostrará el formulario junto con el botón indicado.

Montaña Martín Vergel (Elaboración propia)

10.3.- Edición de la interfaz.

Para comenzar el diseño de la interfaz, seleccionaremos el fichero en formato fxml, en el ejemplo tiene el nombre FXLDocument.xml. Accederemos al menú contextual y seleccionamos la opción Open. Veremos cómo se comienza a ejecutar la aplicación Scene Builder y en su interior nos aparece la configuración que hasta ahora tiene el FXML.



Montaña Martín Vergel (Elaboración propia)

Requisitos de la interfaz.

Se pretende crear una interfaz para gestionar las reservas de un hotel. Tenemos que poder escribir los datos personales de la persona que hace la reserva:

- DNI.
- Nombre.
- Dirección.
- Localidad.
- Provincia.

También tenemos que poder incluir los datos de la reserva:

- Fechas de llegada y salida.
- Número de personas.
- Tipo de habitación. El tipo de habitación puede ser:
  - Doble de uso individual.
  - Doble.
  - Junior suite.
  - Suite.
- Si es para fumador o no fumador.
- Régimen de alojamiento. Puede ser:
  - alojamiento y desayuno.
  - Media pensión.
  - Pensión completa.

Cómo requerimiento adicional se pide que si la reserva es para fumador se muestre el siguiente mensaje:

"En virtud de la ley de sanidad se informa a los clientes de que solo podrán fumar en las habitaciones reservadas para tal fin."

La interfaz debe contar con las facilidades de atajos de teclado y manejo del tabulador.

Una vez terminada se verá así:

Montaña Martín Vergel (Elaboración propia)

10.4.- Diseño de la interfaz.

Comenzaremos a crear la interfaz, accediendo a la aplicación Scene Builder, que se nos abrió automáticamente cuando accedimos al menú contextual sobre el fichero FXMLDocument, creado en el apartado anterior, y seleccionar la opción Open (Abrir).

Para empezar, ampliaremos el panel (AnchorPanel) que nos aparece por defecto y eliminaremos el botón que nos ha insertado por defecto.

A continuación, incluiremos los textos y los cuadros de textos necesarios. Necesitaremos etiquetas para mostrar los textos: el título (Datos del Cliente), Nombre y apellidos, DNI, dirección, Localidad, Provincia, título Datos de la reserva, Fecha de llegada, Fecha de salida, Número de habitaciones, Tipo de habitación, Régimen y el texto que se desea mostrar cuando se selecciona una habitación para fumador.

Montaña Martín Vergel (Elaboración propia)

Para insertar las etiquetas seleccionamos, ***Controls*** y la opción ***Label***. Iremos arrastrando a nuestra interfaz cada una de las etiquetas. Una vez insertada en el escenario, accederemos al panel de inspección y modificamos las propiedades para conseguir el aspecto deseado. Modificaremos la propiedad ***Text*** para introducir el texto que deseamos mostrar.

El siguiente paso es insertar los cuadros de texto. Para ello, seleccionamos el objeto ***TextField*** que podemos encontrar en el interior de la librería **Controls.** Podemos modificar sus propiedades a través del panel ***Properties***.

Montaña Martín Vergel (Elaboración propia)

El siguiente paso es añadir los objetos para que aparezca el calendario y se puedan seleccionar la fecha deseada. Para ello, hay que seleccionar el control **DataPicker (FX8)**

Para introducir el número de personas, utilizaremos el control **TextField** y para indicar el tipo de habitación emplearemos un **ComboBox**. Los valores que debemos de mostrar en el ComboBox no se pueden introducir desde Scene Builder, lo tenemos que realizar mediante código.

Montaña Martín Vergel (Elaboración propia)

Para ello, realizamos los siguientes pasos:

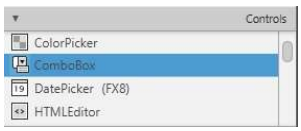
1-. Insertamos mediante Scene Builder el combobox en el lugar deseado.

2-. Accedemos a NetBeans y editamos el fichero FXMLDocumentController.java para introducir el siguiente código:

```
@FXML private ComboBox comboBox; //Esta línea la pondremos justo antes de la declaración public void initialize....
```

3-. En el interior del método public void initialize... debemos de insertar la línea:

```
comboBox.getItems().addAll("Doble de uso individual","Doble","Junior Suite","Suite");
```

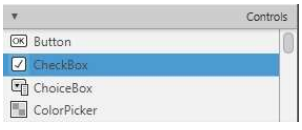


Montaña Martín Vergel (Elaboración propia)

4-. Debemos importar `import javafx.scene.control.ComboBox;`

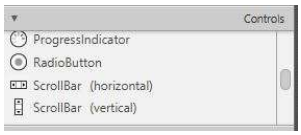
5-. Volvemos a Scene Builder para modificar el fichero FXMLDocument y seleccionamos dentro del panel Code para en el campo `fx:id` seleccionar en el desplegable el valor comboBox.

Para indicar si es fumador o no debemos de utilizar un control de tipo CheckBox.



Montaña Martín Vergel (Elaboración propia)

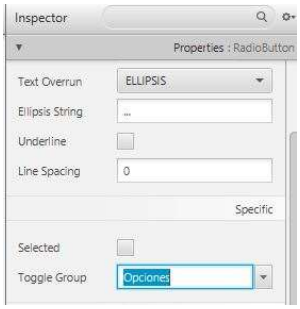
Para indicar el tipo de régimen de alojamiento, utilizaremos controles Radio Button. Al ser cuatro opciones, cuando se pulse alguna de ellas se tiene que desactivar las otras tres. Para ello, deberemos de modificar la propiedad `Toogle Group` e indicarles a todos los radios que contengan el mismo nombre de grupo.



Montaña Martín Vergel (Elaboración propia)

Para que quepan todos los controles, tendremos que agrandar la ventana del diálogo, se agranda arrastrando con el ratón desde los bordes de la ventana. Para mover los botones basta con seleccionarlos con el ratón y arrastrarlos.

Para introducir los botones Aceptar y Cancelar, utilizaremos **Controls** de tipo ***Button.***



Montaña Martín Vergel (Elaboración propia)

La interfaz resultante sería similar a:

Montaña Martín Vergel (Elaboración propia)

10.5.- Funcionalidad CheckBox Fumador.

Necesitamos que el texto "*En virtud de la ley de sanidad se informa a los clientes de que solo podrán fumar en las habitaciones reservadas para tal fin"* se muestre cuando se encuentre activado el checkBox asociado al valor Fumador.

Para ello, tenemos que seguir los siguientes pasos:

1-. Modificar el fichero **`FXMLDocumentController.java`**: insertaremos el siguiente código:

Antes de la definición del método initialize, deberemos de declarar la instrucción:

```
@FXML private CheckBox checkBox;

    @FXML private Label labelAviso;

Añadiremos el siguiente método justo después de la declaración del método initialize(...):

public void handleCheckBox(){
      if (checkBox.isSelected()){
                  //etiqueta será visible
                  labelAviso.setVisible(true);
      }else{
              labelAviso.setVisible(false);
     }
 }
```

Modificaremos el arranque de la aplicación para que se oculte la etiqueta al iniciar la aplicación. Para ello, escribiremos dentro del método public void initialize(URL url, ResourceBundle rb) la instrucción:

```
labelAviso.setVisible(false);
```

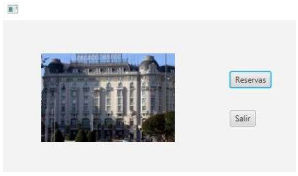
2-. Accederemos a Scene Builder para modificar el fichero FXMLDocument:

Seleccionamos del checkbox asociado a Fumador y el panel code (parte derecha), seleccionamos la opción ***fx:ide*** para seleccionar en el despegable el valor checkBox. Además, seleccionaremos el apartado ***on Action*** para indicar handleCheckBox.

Seleccionar la etiqueta asociada al texto que deseamos mostrar y ocultar para asignarle en el panel code, la propiedad ***fx:ide*** igual a labelAviso (debe de aparecer en el despegable).

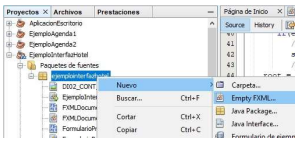
10.6.- Crear Formulario principal.

Vamos a crear a la aplicación un formulario principal que tenga el siguiente aspecto:



Montaña Martín Vergel (Elaboración propia)

Comenzaremos por añadir a nuestro proyecto un nuevo **FXMLDocument.** Para ello, nos situamos en NetBeans sobre el paquete que contiene los ficheros ya existentes, accedemos al menú contextual y pulsamos el botón Nuevo --> Empty FXML. A continuación, introducimos el nombre, que en nuestro caso le vamos a asignar el nombre ***FormularioPrincipal***. A continuación, activaremos la opción "***Use java controller"*** y después marcaremos la opción "***Use cascading Style Sheets***" sí queremos crear una hoja de estilos asociada a este nuevo formulario.



Montaña Martín Vergel (Elaboración propia)

Una vez insertado, veremos que nos aparecen tres nuevos ficheros: *FormularioPrincipal.fxml* , *FormularioPrincipalController.java* y si hemos insertado hojas de estilos *formularioprincipal.css*.

Montaña Martín Vergel (Elaboración propia)

A continuación, abrimos el fichero **`FormularioPrincipal.fxml`** desde NetBeans y accedemos a Scene Builder. Insertaremos un container de tipo ***Pane*** y lo agrandaremos para que ocupe el espacio necesario para incluir el resto de controles. Sobre este *Pane*, procederemos a crear la interfaz añadiendo dos controles de tipo ***Button*** para los botones "Reservas" y "Salir". Para colocar la imagen del hotel se debe de utilizar un control de tipo ***ImageView.*** Para que se vea la imagen, debemos de colocarla en la misma carpeta o directorio en donde tengamos los ficheros fuentes de nuestro proyecto, y modificaremos la propiedad ***Image*** para buscar la imagen que queramos mostrar.

Accedemos a NetBeans para modificar el código. Comenzaremos por modificar el fichero **`FormularioPrincipalConmtroller.java`** para insertar:

Comenzaremos por realizar las siguientes importaciones al comienzo de la clase en el caso de que no las tuviéramos:

```
import javafx.application.Platform;
	import javafx.event.ActionEvent;
	import javafx.fxml.FXML;
	import javafx.fxml.FXMLLoader;
	import javafx.fxml.Initializable;
	import javafx.scene.Parent;
	import javafx.scene.Scene;
	import javafx.scene.control.Button;
	import javafx.stage.Stage;
```

Justo antes de la definición del método **`public void initialize(URL url, ResourceBundle rb)`**, escribiremos:

```
@FXML private Button btnReservas;
	@FXML private Button btnSalir;
```

Después de la definición del método initialize, insertamos el siguiente código:

```
public void handlebtnReservas(ActionEvent event) throws IOException{
  Stage stage =(Stage) btnReservas.getScene().getWindow();
  Parent root = FXMLLoader.load(getClass().getResource("FormularioPrincipal.fxml"));
  if(event.getSource()==btnReservas){
    stage=(Stage) btnReservas.getScene().getWindow();
    root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
  }
   Scene scene = new Scene(root);
   stage.setScene(scene);
   stage.show();
   }
  public void handlebtnSalir(ActionEvent event) throws IOException{
   Platform.exit();
  System.exit(0);
}
```

El manejador del primer método, corresponde al botón Reservas. El código permite acceder al formulario para definir las reservas. El manejador del segundo botón nos permite cerrar la aplicación.

Ahora accedemos de nuevo a Scene Builder para modificar el fichero FormularioPrincipal.fxml. Seleccionamos el botón Reservas y en el panel de ***Code*** (parte derecha) seleccionamos la opción ***fx:id*** y en el despegable seleccionamos btnReservas. En el apartado ***on Acfion*** seleccionamos handlebtnReservas.

A continuación, seleccionamos el botón Salir y seleccionamos la opción fx:id sobre el panel Code. Seleccionamos la opción btnSalir y el apartado on Action seleccionamos handlebtnSalir

Para finalizar accedemos desde NetBeans al fichero "EjemploInterfazHotel,java" para modificar el método start para que quede de la siguiente manera:

```
public void start(Stage stage) throws Exception {
   Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
   Parent root2 = FXMLLoader.load(getClass().getResource("FormularioPrincipal.fxml"));

   Scene scene = new Scene(root2);
   stage.setScene(scene);
   stage.show();

 }
```

Ya estamos en disposición de poder probar nuestra interfaz.