**LA INTERFAZ GRÁFICA**

**Introducción**

En Java la historia de la Interfaz gráfica de usuario, (a veces conocida como GUI, siglas de Graphical User Interface, Interfaz Gráfica de Usuario) entendida como el uso de controles gráficos para controlar una aplicación, cubre un amplio marco de refinamientos incrementales. Cronológicamente las diferentes soluciones propuestas por JAVA son AWT, Swing, SWT y JavaFX.

A**wt**

Fue definido como un conjunto de paquetes para la construcción de interfaces gráficas. Está basado en contenedores gráficos más componentes gráficos, más modelo de eventos. Los componentes de AWT están implementados en código nativo, es decir que están restringidos a las características de la plataforma.

Por ejemplo, si la plataforma no permite imágenes en los botones, entonces no será posible construir un botón con una imagen en lugar del texto. La mayoría de los componentes de AWT son subclases de Component, excepto los contenedores de mayor nivel como Frame, Dialog y Applet.

**Swing**

Si bien ha quedado inmortalizado como Swing, su denominación es JFC (Java Foundation Classes) ya que incluye otros paquetes adicionales. Las JFC incluyen un grupo de características para la creación de interfaces gráficas. Los componentes Swing GUI es una de las características de las JFC.

Está basado en contenedores gráficos más componentes gráficos más un modelo de eventos. Es el conjunto de paquetes utilizados para la construcción de interfaces gráficas. Los componentes Swing están implementados sin código nativo, es decir que están no restringidos a las características de la plataforma.

Por ejemplo, si la plataforma no permite imágenes en los botones, con Swing es posible agregarlos. La mayoría de los componentes de Swing son subclases de JComponent, excepto los contenedores de mayor nivel como JFrame, JDialog y JApplet. Swing utiliza algunas clases de AWT.

**Swt**

SWT son las siglas en inglés de Standard Widget Toolkit. Es un conjunto de componentes para construir interfaces gráficas en Java, (widgets) desarrollados por el proyecto Eclipse.

Recupera la idea original de la biblioteca AWT de utilizar componentes nativos, con lo que adopta un estilo más consistente en todas las plataformas, pero evita caer en las limitaciones de ésta.

Si bien SWT salió a la luz más tarde que SWING, son tecnologías contemporáneas, cada una tiene sus pros y sus contras. Entre sus diferencias el aspecto visual es uno de los aspectos que más se destaca, recordemos que SWT toma el aspecto de los controles del sistema operativo donde corren, pero para tener una idea Netbeans está desarrollado con SWING y Eclipse está desarrollado con SWT.

**JavaFX**

JavaFX amplía la potencia de Java permitiendo a los desarrolladores utilizar cualquier biblioteca de Java en aplicaciones JavaFX. De esta forma, los desarrolladores pueden ampliar sus capacidades en Java y utilizar la tecnología de presentación que JavaFX proporciona para crear atractivo visual.

**Características principales de JavaFX**

* Permite a los desarrolladores integrar gráficos vectoriales, animación, sonido y activos web de vídeo en una aplicación interactiva, completa y atractiva
* Amplía la tecnología Java permitiendo el uso de cualquier biblioteca de Java en una aplicación JavaFX
* Permite mantener un eficaz flujo de trabajo entre diseñador y desarrollador en el que los diseñadores pueden trabajar en las herramientas que deseen mientras colaboran con los desarrolladores

JavaFX es la alternativa oficial de Java para aplicaciones de escritorio, esto quiere decir que no existirán nuevos desarrollos sobre el paquete SWING.

**Herramientas que utilizaremos**

* Última versión de Java JDK 8 (incluye JavaFX 8).
* Netbeans
* Scene Builder 2.0 o superior

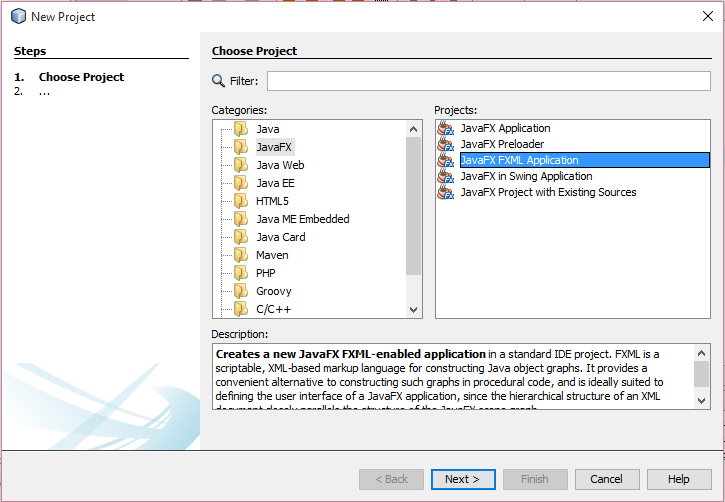
**Scene Builder**

JavaFX Scene Builder es una herramienta visual que permite diseñar rápidamente interfaces de usuario (UI) para aplicaciones JavaFX sin la necesidad de escribir una sola línea de código. Simplemente con drag-and-drop se pueden posicionar componentes gráficos de interfaz de usuario en una escena JavaFX. Incluso puede ser utilizada por perfiles no programadores para realizar el prototipo de una aplicación.

**Como crear una aplicación JavaFX**

Existe más de una forma para desarrollar una aplicación JavaFX, si bien en nuestro caso utilizaremos una herramienta visual para diseñar y crear las ventanas, las mismas podrían ser creadas codeando las clases al igual que sucede con SWING o SWT donde podemos optar por utilizar una herramienta visual que genera las clases codeadas o podemos codear las clases por nuestra cuenta. A continuación, explicaremos la forma de hacerlo utilizando las facilidades que nos brindan NetBeans y Scene Builder.

Desde Netbeans dirigiéndose al menú “File” -> “New Project” se debe seleccionar la opción “JavaFX” -> “Java FXML Application” tal como se observa en la imagen a continuación.



De esta forma observaremos que en NetBeans se crea un proyecto con tres elementos:

1. Un archivo FXMLDocument.fxml.
2. Una clase Java FXMLDocumentController.java (Controlador).
3. Una clase Java con el nombre del proyecto (Clase principal).

**Archivos .fxml**

Tal como adelantamos existen dos formas de crear la interfaz de usuario, codeando en Java o mediante un archivo XML. En Internet hay mucha información relativa a ambos métodos. En nuestro caso utilizaremos XML (archivo con la extensión .fxml) para casi todo. De esta forma puede resultar más fácil mantener el controlador y la vista separados entre sí.  Esto también permite utilizar la herramienta de edición visual Scene Builder, la cual nos evita tener que trabajar directamente con el XML.

**Controladores**

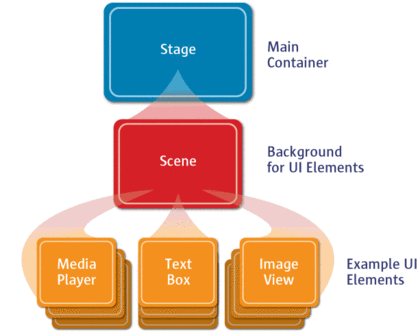
Cada uno de los archivos .fxml debe estar asociado a una clase Java donde se encuentra programado el comportamiento que sucederá a medida que se disparen eventos en la interfaz gráfica, ya sea al presionar un botón, hacer foco sobre un control o cualquier evento sobre el cual se necesite desarrollar un comportamiento. A estas clases se las conoce como controladores respondiendo al patrón de diseño Model-View-Controller.

**Clase principal**

El tercer archivo que observamos es lo que se denomina Clase principal, es la clase que contiene el “main” que dispara nuestra aplicación JavaFX. La clase generada extiende de la clase Application y además del main contiene un método llamado “start”.

El método start(Stage primaryStage) es invocado automáticamente cuando la aplicación es lanzada desde el main y en él podemos observar que hace referencia al archivo  .fxml donde se encuentra el diseño de la ventana.

El método start(...) toma un Stage como parámetro, en el siguiente gráfico siguiente podemos observar la estructura de una aplicación JavaFX:



Observando el gráfico nos podemos dar cuenta que la estructura de la aplicación es una analogía a una obra de teatro donde el Stage (escenario) es el contenedor principal, normalmente una ventana con borde y los botones para maximizar, minimizar o cerrar la ventana y dentro del Stage se puede añadir una Scene (escena), la cual puede cambiarse dinámicamente por otra Scene. Dentro de un Scene se añaden los nodos JavaFX, tales como AnchorPane, TextBox, etc.

**MDI Documento de Múltiples Interfaces**

**Introducción a MDI**

El hecho de necesitar agrupar funcionalidades distintas en una única aplicación genera la necesidad de ejecutar una única aplicación y tener acceso a resolver varias funcionalidades.

Un ejemplo de esto es la aplicación de una clínica donde se necesita resolver en una misma aplicación el concepto de turnos para los pacientes, agendas de los médicos, y datos personales e historias clínicas de los pacientes.

Todo esto puede ser agrupado en un mismo sistema, en una misma aplicación, al poder ejecutar varias interfaces o varias ventanas dentro de una ventana principal. El modo de acceder a estas múltiples ventanas de a una puede lograrse a través de un menú de selección de la funcionalidad que se requiere utilizar.

En nuestro ejemplo conceptual podemos tener un Menú Pacientes, y un submenú Datos Personales, de modo tal de resolver los temas de los datos personales del paciente, luego un submenú Historia Clínica, donde aparecen la lista de consultas y se puede seleccionar que se resolvió como diagnóstico en cada consulta, luego un submenú de Turnos del paciente, donde se consoliden los datos de asignación de pacientes a médicos desde la perspectiva de un paciente.

Como otro menú podemos tener el Menú Médico, el cual puede tener un submenú Datos Personales del Médico, por ejemplo, datos de contactos y horarios en los que se lo puede encontrar, luego un submenú Agenda Semanal, donde se puede acceder a la información de agenda diaria del médico, que pacientes ya tiene asignados en que horarios y en qué momento queda libre para una nueva asignación.

Lógicamente se espera que todas estas ventanas internas, ventanas que se disparan a partir de los submenús vivan dentro de la aplicación de la ventana principal, de modo de poder abrir varias ventanas internas y poder pasar de una a otra sin cerrarlas.