JavaFx

Ahora el límite lo pone tu imaginación...

Universidad Tecnológica Nacional
Materia: Seminario
Profesor: Leonel Guccione

Alumnos:
Alberola, Gustavo Alejandro
Alvarez Durán, Matías Emiliano
08/06/2009

CONTENIDO

INTRODUCCIÓN

JavaFx es el nombre de la nueva tecnología desarrollada por la empresa Sun Microsystem.

Esta tecnología presenta compatibilidades con su predecesor JavaEE, JavaME y javaEE, asi como también, es compatible con varias plataformas, Mobile, Desktop, Web, T.V.

COMO LUCE JAVAFX?

Para responder a esta pregunta, se investigó y desarrolló una aplicación en JavaFx para la búsqueda on-line de libros, la cual interactua con un servicio diseñado en Php, el cual se comunica con sitios como Google Books, y Amazon.

Como se desarrollaron las aplicaciones, el diseño de las mismas, ventajas y desventajas, la sintaxis, y todo lo referente a JavaFx sera detallado en el siguiente informe.

JAVAFX VERSIÓNES ANTERIORES A LA RELEASE 1.0

Este proyecto de investigación busca objetivo presentar la nueva tecnología que **SUN** ha estado desarrollando (y que hoy en día continúa en este estado), para el desarrollo de interfaces visuales.

Para poder abarcar la mayor parte del contenido de esta nueva plataforma se realizo una exhaustiva investigación y documentación de la misma.

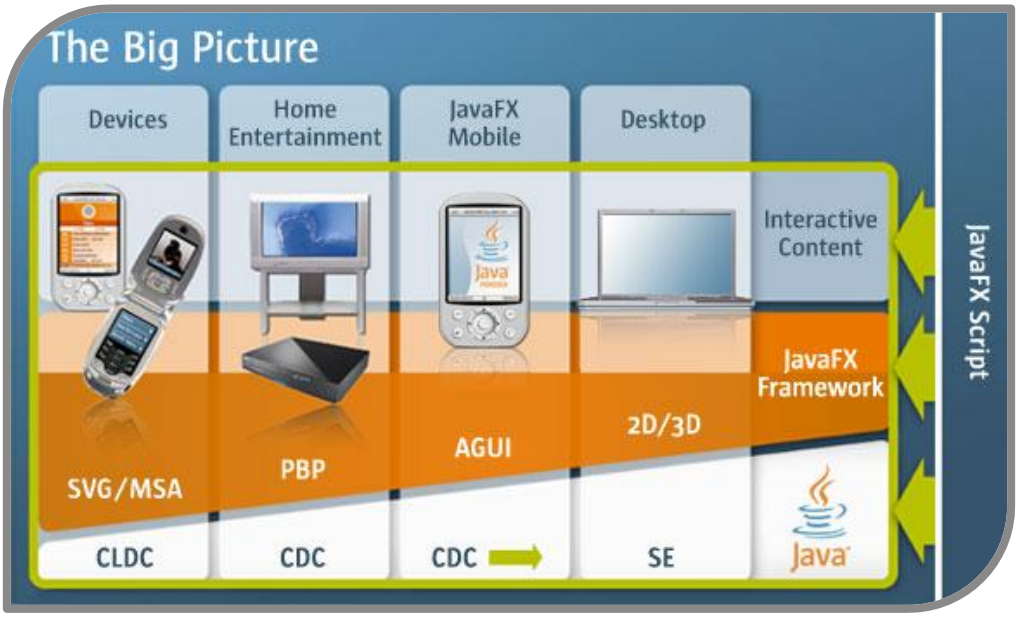
La documentación consta de una descripción de cada uno de los componentes de esta nueva familia de productos y utilidades, (que según **SUN**, empresas reconocidas y varios desarrolladores) que marcarán el camino de las aplicaciones del futuro.

El trabajo concluye con la presentación de una aplicación que demuestra el potencial que estas fantásticas herramientas nos brindarán.

INTRODUCCIÓN (JAVAFX PREVIEW SDK)

Una de las últimas plataformas presentadas en la **JavaOne** (conferencia anual organizada por **SUN MICROSYSTEMS** para discutir las tecnologías de Java, sobre todo entre los desarrolladores de Java) en Mayo del 2007 fue **JavaFX.**

JavaFX es una familia de productos y tecnologías pensados para el desarrollo de *"Rich Internet Applications"* (RIA's). Este producto es la contrapartida de SUN, para las herramientas de desarrollo de este tipo de aplicaciones, como **Flash** de ADOBE y **Silverlight** de MICROSOFT. Una de las características principales de esta tecnología es la posibilidad del desarrollo de interfaces visuales para escritorio, web, dispositivos móviles y TV; con todo el soporte de **JAVA SE**, **JAVA EE** y **JAVA ME**.



FAMILIAS DE PRODUCTOS Y TECNOLOGÍAS DE JAVAFX

Esta novedosa tecnología está compuesta por un amplio abanico de productos:

- JavaFX: Es un nuevo lenguaje diseñado para animaciones y gráficos. Permite la vinculación de componentes escritos en otros lenguajes (principalmente en Java) con rapidez, código simple y completamente orientado a objetos.
- Designer Tools: Un conjunto de herramientas pensadas para diseñadores gráficos profesionales. Estas no son solo un conjunto de plugins, sino herramientas desarrolladas explícitamente para los mismos. Un ejemplo de estas herramientas es Project Nile: Es un conjunto de herramientas para la conversión de gráficos a un formato que puede ser utilizado para aplicaciones de JavaFX. Permite al diseñador y al desarrollador el trabajo conjunto en las aplicaciones gráficas. El experto gráfico puede definir capas y elementos en sus diseños utilizando potentes aplicaciones de diseño (Adobe Illustrator ó Adobe Photoshop), y el desarrollador puede acceder a los mismos referenciándolos desde código JavaFX y animarlas.
- JavaFX Mobile: Esta basado en la plataforma Java ME, aprovechando su potente nivel de capacidades para dispositivos móviles. Actualmente la plataforma JavaME ejecuta en más de 2 mil millones de teléfonos, y SUN se encuentra bien posicionado para introducir al mercado a JavaFX como la principal tecnología para el desarrollo de RIA's en dispositivos móviles.
- JRE (Java Runtime Enviroment): Son un conjunto de mejoras y nuevas características para JavaSE, las cuales serán incluidas en las nuevas actualizaciones de la versión de Java 6. Estas nuevas características serán lanzadas en varias versiones a partir de la primavera de 2009:
  - El conjunto de herramientas y plugins puede ser incluido en el navegador con una simple línea de código JavaScript, para poder ejecutar o instalar las aplicaciones.
  - Un nuevo instalador más confiable, sencillo y estable.
  - Actualmente **SUN** integró los componentes de Java FX de manera que cuando los usuarios inicien las aplicaciones, solo necesiten descargar el código y no las librerías para ejecutar este.
  - Un nuevo servicio *"Quick Starter"* desarrollado para Windows, para optimizar la velocidad de ejecución de las aplicaciones Java manteniendo en la caché las librerías de Java.
  - Nuevos soportes para medios de comunicación (Video y audio), incluyendo un códec que estará integrado.

EL LENGUAJE JAVAFX

Lenguaje orientado al diseño de interfaces de usuario, utilizando una simple sintaxis, los desarrolladores pueden crear RIA´s. JavaFX es un lenguaje declarativo y completamente orientado a objetos. Al igual que su antecesor Java.

COMO SE RELACIONA JAVAFX CON JAVA?

La programación de interfaces de usuarios con las herramientas Swing que brinda Java permiten realizar una increíble cantidad de funcionalidad, pero a su vez, esta puede llegar a ser muy compleja.

JavaFX impulsa todo el poder de Java, debido a que su código puede utilizar en su totalidad las librerías de este lenguaje. Muchas de las capacidades de la interfaz de usuario de JavaFX hacen uso de las clases de Java.

EL efecto producido por esta nueva tecnología es que desarrolladores y diseñadores pueden utilizar un lenguaje simple y elegante que utilice todos los beneficios de Java y Java Swing.

MEJORAS Y VENTAJAS DE JAVAFX

Estas son algunas de las fortalezas del lenguaje JavaFX:

- Es simple, la sintaxis declarativa es utilizada para declarar interfaces de usuario, incluyendo una amplia colección de herramientas que facilitan en gran medida los desarrollos de las interfaces de usuario. Los diseñadores de contenido pueden crear interfaces atractivas sin tener que ser expertos programadores.
- Su habilidad innata para soportar patrón Model-View-Controller debido a su poderosa capacidad de enlace (bind). Esta funcionalidad complementa y habilita la sintaxis de programación declarativa debido a que los atributos de los objetos, incluyendo los objetos de interfaz de usuario, pueden ser enlazados a valores contenido en el modelo de clase. Este enlace puede ser bidireccional.
- El concepto de triggers habilita la sintaxis de programación declarativa, de la misma manera que vuelve los desarrollos de interfaces de usuario relativamente sencillos, ya que los *setters* y *getters* son reemplazados por *triggers*, los cuales son invocados automáticamente cuando el valor de un atributo es modificado.
- Los programas realizados con JavaFX podrán ejecutarse en cualquier lugar que pueda correr una aplicación Java, debido a que se ejecutan dentro del contexto de la **JVM** (*Java Virtual Machine)*.

EL ESTADO DE JAVAFX

Al comienzo de la investigación, JavaFX se encontraba en estado Alfa (para test interno), por ende, sufrió fluctuaciones, constantes actualizaciones y mejoras, y cambios en las librerías de JavaFX.

Al final de la codificación de la implementación (WebooksFX), SUN liberó la version 1.2 de JavaFx. Se corrigieron y agregaron modificaciones críticas, quedando el código de la aplicación desactualizado.

La aplicación WebooksFX solo corre bajo la version 1.1 de la SDK.

ENTORNOS DE DESARROLLO QUE SOPORTAN JAVAFX

Los entornos de desarrollo más comunes que soportan esta nueva tecnología son:

- JavaFXPad: esta es una buena herramienta para familiarizarse con JavaFX.
- Eclipse: con el plug-in para JavaFX, Eclipse es un Entorno de Desarrollo Integrado muy completo. Esta es una buena opción para el desarrollo de este tipo de aplicaciones.
- NetBeans: con el plug-in para javaFX, NetBeans es otro Entorno de Desarrollo Integrado muy completo, el cual cuenta con el completo respaldo de **SUN MICROSYSTEMS**.

JAVAFX, OTRA HERRAMIENTA PARA EL DESARROLLO DE RIA´S?

En el mercado actual podemos encontrar soluciones para el desarrollos de RIA´s, como lo son Silverligth de Microsoft o Flash de Adobe. Entonces, por que elegir JavaFX?

JavaFX no es solo otro lenguaje para el diseño de RIA´s, de manera que compararlo con las tecnologías anteriormente mencionadas no es del todo correcto. La ventaja de utilizar JavaFX es el poder de la plataforma Java. No es el simple hecho de desarrollar interfaces de usuario o servicios web, sino el hecho de utilizar esta plataforma y todo lo que la misma ofrece.

MÁS ALLÁ DE LO FUNCIONAL (EL NUEVO DESAFÍO DE LAS APLICACIONES MODERNAS)

INTRODUCCIÓN A LAS APLICACIONES *"RICH CLIENTS"*

Las aplicaciones modernas son cada vez más atractivas y complejas visualmente. La utilización de toolkits de componentes visuales no es del todo buena. Ya que si bien nos permite el desarrollo rápido de interfaces visuales, por otro lado nos impide la creación de efectos gráficos avanzados tales como animaciones, efectos de filtro o incluso 3D. Por consecuente esto trae ciertos problemas a las aplicaciones en desarrollo:

- La mayoría de los desarrolladores de software no son capaces de realizar diseños atractivos gráficamente.
- Los diseñadores no suelen ser capaces de comprender la lógica de negocio y transportarla al desarrollo de las interfaces.
- Ambos campos deben trabajar juntos de manera productiva, para generar un producto de calidad.

LA SOLUCIÓN PRODUCTIVA DE JAVAFX

El conjunto de herramientas (de JavaFX) pensadas para diseñadores gráficos profesionales, permite a estos crear complejos y atractivos diseños en potentes aplicaciones y desplegarlos sobre la plataforma Java. Favoreciendo a los desarrolladores a ser más productivos, dejando el trabajo de la creación de las interfaces visuales a los expertos en este campo.

Ventajas de la utilización de las herramientas de diseño:

- Productividad
  - Las cosas simples son fáciles de crear, y las difíciles son posibles.
  - Desarrollos pensados en semanas, no en meses.
- Creatividad
  - Aplicaciones atractivas y gráficas animadas, no interfaces rígidas generadas a partir de toolkits.
- Trabajo conjunto entre diseñadores y desarrolladores.
- No es necesario descartar aplicaciones que hoy en día estén basadas en la tecnología Java, solo con cambiar la interfaz de usuario de la misma es suficiente.

El siguiente, en un ejemplo de una interfaz gráfica creada con la plataforma JavaFX (extraída de la presentación *"Bringing Designers and Developers Together with JavaFX™"* de la JavaOne del 2008) :

Para el desarrollo de este tipo de "Interfaces de Usuario" no son necesarios más de dos profesionales. Gracias a esta novedosa plataforma todo el desarrollo se vuelve más ágil y ordenado.

ROLES EN EL DESARROLLO DE APLICACIONES *"RICH CLIENTS"*

Tanto los desarrolladores como los diseñadores, tienen la misma meta, desarrollar aplicaciones atractivas y funcionales.

- Sin embargo, el uso de diferentes herramientas y enfoques del problema trae varios problemas.
- Ambos desarrollan distintas piezas de un mismo sistema.
- La iteración es frecuente.
- Hay que asegurarse que la integración de ambas piezas sea lo más fácil posible.

Roles:

- Diseñador:
  - Responsable de la creación de la interfaz de usuario.
  - Ofrece gráficos, contenido multimedia, etc.
  - Debe asegurarse que la aplicación es intuitiva, y de que se puede utilizar.
- Desarrollador:
  - Se centra en el desarrollo de la lógica de negocio.
  - Se asegura que la aplicación cumple con los requisitos funcionales.

CONECTANDO A DISEÑADORES Y DESARROLLADORES

El paradigma de la tecnología JavaFX puede verse desde dos perspectivas:

- El desarrollo de pequeños proyectos ó componentes encapsulados, en donde se combinan gráficos directamente con el código de las aplicaciones.

- El desarrollo de grandes y ambiciosos proyectos, en donde es necesario la separación de los gráficos y pantallas con el código de la aplicación, en diferentes "capas". Esto se debe a que este tipo de proyectos están compuestos de gráficos complejos, muchas pantallas, etc.).

Los desarrolladores y diseñadores necesitan trabajar en gráficos y código como si fueran componentes separados. Pero a su vez, utilizar herramientas que aseguren una integración de los mismos en cualquier momento.

La plataforma JavaFX presenta una tecnología de definiciones de formato (FXD). Esta no es más que un formato de texto para el almacenamiento de gráficos y demás. La verdadera innovación es que puede ser utilizada desde JavaFX, ya que posee una sintaxis declarativa para el manejo de este formato.

DESIGNER TOOLS

Project Nile es una de las herramientas orientadas a los diseñadores, desarrolladas solo con el objetivo de la conversión de gráficos al formato FXD que puede ser utilizado para aplicaciones **JavaFX**. Como se menciono anteriormente, el diseñador puede definir capas y elementos en sus gráficos, y el desarrollador puede acceder a estos referenciándolos desde código **JavaFX**.

Usando los plugins de **Project Nile**, se pueden exportar gráficos desde Adobe **Illustrator** ó Adobe **Photoshop**, ya sea en formato FXD ó código **JavaFX**. Al realizar la exportación en formato FXD, el plugin también genera código **JavaFX** , el cual permite a los desarrolladores acceder a los datos definidos en el archivo FXD.

En conclusión, **Project Nile** está compuesto por seis componentes:

- Ilustrator Plugin: *Adobe Illustrator CS3 plugin*, permite la exportación de archivos a formato FXD ó código **JavaFX** .
- Photoshop Plugin: *Adobe Photoshop CS3 plugin*, permite la exportación de archivos a formato FXD ó código **JavaFX** .
- SVG Converter: Herramienta que permite la conversión de SVG a gráficos FXD ó código **JavaFX** , desde la línea de comandos ó una interfaz.
- JavaFX Graphics Viewer: Herramienta que permite la visualización de gráficos generados por **Project Nile**.
- Librería: Una librería (archivo JAR) que debe adjuntarse a cualquier proyecto, para la utilización del formato de archivos FXD.
- Ejemplos: Aplicaciones de muestra que se pueden ejecutar utilizando la IDE **NetBeans** 6.5 (que trae integrado el plugin de JavaFX). Estas, contienen archivos de diferentes orígenes (Photoshop e Illustrator), más todas las fuentes necesarias para ejecutarlas.

JAVAFX MOBILE

Las capacidades de proceso y conectividad en los dispositivos móviles de hoy en día tienen el potencial para poder otorgar al usuario una nueva clase de RIA´s. Para que los desarrolladores pudieran acceder a estos beneficios, **SUN MICROSYSTEMS** se encuentra brindando las tecnologías de JavaFX para dispositivos móviles. Aun más, JavaFX Mobile se lanzará a la cabeza del mercado, el cual es liderado por la tecnología de Java ME (actualmente es utilizado por más de dos mil millones de dispositivos móviles), obteniendo una gran ventaja de esta situación.

Uniendo estas dos tecnologías, permitirán que los desarrolladores de contenidos no tengan más barreras que las de su imaginación, brindando un ambiente en el cual expresarse y entregar contenido competente en el mercado. Lo más beneficioso de todo esto para los desarrolladores, es que, el contenido creado con la tecnología de JavaFX tiene la capacidad de ejecutarse en casi cualquier pantalla (Computadores de escritorio, aplicaciones Web, Celulares, TV). Esto significa que el desarrollo de una interfaz de usuario para un tipo de pantalla será mucho más fácil de trasladarlo a otra pantalla que con cualquier otra tecnología.

Por si esto fuera poco, **SUN** está trabajando con otros miembros en el ecosistema de la telefonía móvil para construir una completa solución compatible con la mayoría de dispositivos de la actualidad. Además, el contenido desarrollado con JavaFX Mobile será completamente compatible con los sistemas cada vez más complejos de las nuevas generaciones, volviendo menos complicado la migración hacia estos últimos.

Esta tecnología no fue lanzada al mercado hasta la versión 1.1 de JavaFX, en la cual se incluyó un emulador para móviles, permitiendo a los desarrolladores crear aplicaciones para este tipo de dispositivos.

JAVAFX RELEASE 1.0

En diciembre de 2008, Sun lanzó la primer release de JavaFX, esta traía varias mejoras, y muchas modificaciones a la version anterior (Beta).

SE AGREGARON NUEVAS CLASES PARA EL MANEJO DE GRÁFICOS Y ANIMACIONES

- Path-based animation (animación de objetos a través de un camino)
- Transition library (listado de transiciones que permiten al desarrollador incorporar efectos)
- Slide (Desplazamiento)
- Zoom
- Rotate (Rotación)
- Image mask (Máscaras)
- Stroke styles (subrallado, tachado, etc.)
- Advanced clip support (Soporte para Clips avanzado)
- Multiline text node (Nodos de texto multilinea)

MULTIMEDIA

- Cross-platform Video (Video multiplataforma)
- Cross-platform Audio (Audio multiplataforma)
- Volume control (Control de volume)
- Track control (Control de pistas)
- Native media framework support for the Mac (Soporte native para multimedia en Mac OS)

WEB SERVICES

- HTTP Client API
- JSON Parser (parseador JSON)
- XML parser (parseador XML)

MEJORAS DEL LENGUAJE

Nuevos mecanismos de control de acceso:

- Public
- Protected
- Package

Dentro de las mejoras al lenguaje, destacamos una incompatibilidad muy grande entre las versions anteriores a la 1.0 y esta. El identificador def ahora solo es implementado estrictamente para variables cuyo valor no puede ser modificado a lo largo de la ejecución del programa (no por asignación = , pero si mediante *bindings*)

Las clases ahora tienen atributos privados, restringiendo el acceso a los mismos. La desventaja es que estos no pueden ser inicializados mediante el lenguaje declarativo de JavaFx, sino que deben ser cargados luego accediendo al setter de ese parámetro.

La visibilidad por defecto de un atributo en una clase ahora es privado.

Se reemplazan los operadores % por mod, y <> por !=

MEJORAS EN LA PERFORMANCE

- Se redujo el uso de la memoria de ejecución en las aplicaciones
- Se redujo el uso de la memoria en tiempo de compilación

LA PLATAFORMA, HERRAMIENTAS, Y REFLECTION

- Se mejoro el debuggin de la plataforma Fx.
- Soporte para la API de reflection, que permite la llamada de las API´s de JavaFX desde Java, y JavaME.

JAVAFX RELEASE 1.1

En esta version contamos con un emulador para teléfonos móbiles, permitiendonos desarrollar aplicaciones para los mismos. Aunque la SDK de JavaFX aun no se encuentra disponible para los teléfonos.

Se mejoró la performance y estabilidad de las aplicaciones de escritorio.

Mejor soporte para la portabilidad de aplicaciónes entre la plataforma Mobile y Desktop.

MEJORAS DE LA PLATAFORMA

Soporte para aplicaciones en modo "Pantalla Completa"

Se mejoró el soporte para aplicaciones que requieren acceso a recursos en multiples dominios.

MEJORAS DEL LENGUAJE

Se agregaron tipos numéricos a los tipos definidos del sistema:

- Float
- Double
- Long
- Int
- Byte

Al sobreescribir un método de la clase padre, es estrictamente necesario definir el mismo con el identificador *override*, de lo contrario generará un error en tiempo de compilación.

CLASES QUE SOLO FUNCIONAN EL APLICACIONES DE ESCRITORIO

- Javafx.ext.swing Todas las clases contenidas dentro de este paquete no son compatibles con mobile.
- Javafx.reflect
- Javafx.scene.effect
- Javafx.scene.light
- Javafx.scene.shape.ShapeIntersect
- Javafx.scene.shape.ShapeSubtract
- Javafx.stage.AppletStageExtension
- Javafx.util.FXEvaluator
- Javafx.util.StringLocalizer

JAVAFX RELEASE 1.2

La version de JavaFX 1.2 representa una mejora bastante grande con respecto a su versión predecesora, la 1.1. Muchas de las modificaciones y agregados de la versión 1.2 hacen que esta no sea compatible con las anteriores. Asi como también, se han eliminado clases de la API.

La SDK de JavaFX 1.2 no soporta los archivos binarios de la version 1.1, esto significa que para que los desarrollos realizados con versiones anteriores funcionen, es necesario recompilarlos con la última version.

Para ver la lista completa de cambios es posible acceder al sitio de JavaFx:

http://javafx.com/docs/articles/javafx1-2.jsp

DECLARACIÓN DE VARIABLES

JavaFX proporciona dos palabras reservadas para la declaración de variables

- var
- def

El primero se utiliza para declarar variables que podrán (o no) ser modificadas en un futuro.

```
var x : Integer = 0;
x = 45;
```

El último se utiliza para declarar variables a las cuales, solo puede asignarsele un valor al momento de la declaración, y el mismo no podrá ser modificado posteriormente.

```
Def x : Integer = 0;
x = 45; //Esta linea produciría un error, ya que fue declarada
        //con def. Su valor no puede variar.
```

OBJETOS

JavaFX, al igual que su predecesor, es un lenguaje tipado orientado a objetos. La diferencia entre estos radica en que JavaFX, propone otra sintaxis para inicializar un objeto.

```
Persona{
        nombre: "Gustavo";
        apellido: "Alberola";
}
```

En este código, estamos creando un objeto de tipo *Persona*, al cual le asignamos a sus variables *nombre* y *apellido* los valores *Gustavo* y *Alberola* respectivamente.

Nota: En el ejemplo, estamos utilizando el carácter ";" para denotar el fin de linea y separar la declaración de variables, este también puede ser reemplazado por el carácter "," o por un salto de línea.

De esta manera también podríamos anidar un objeto dentro de otro.

```
Persona{
        nombre: "Gustavo";
        apellido: "Alberola";
        direccion: Direccion{
                calle: "Av. Siempre Viva";
                numero: "000000";
        }
}
```

Ahora el objeto *Persona* contiene en la variable *direccion* un objeto de tipo *Direccion*, el cual a su vez contiene los datos de la *calle* y el *numero*.

TIPOS PRIMITIVOS

Como mencionamos, JavaFX es un lenguaje tipado, asi que describiremos los diferentes tipos de datos contenidos en este lenguaje

- String
- Number e Integer
- Boolean
- Duration

- Void
- Null

STRING

Al igual que en java, el String representa una cadena de caracteres. La misma es declarada entre " (comillas dobles) o ' (comilla simlple).No hay diferencia entre una u otra, y puedes escribir una dentro de la otra.

```
var cadena : String = "Soy una cadena, y esto ' es una comilla"
```

Para mostrar variables, las mismas deben de estar contenidas dentro de { } (llaves). También se pueden introducir sentencias lógicas dentro de estas.

```
var verdadero : Boolean = true;
var cadena : String = "verdadero = { if (verdadero) "SI" else "NO" }";
```

INTEGER Y NUMBER

Estos tipos de datos almacenan valores numéricos, la única diferencia entre uno y otro, es que *Integer* almacena valores enteros y *Number* valores con punto flotante.

```
var integer : Integer = 1;
var number : Number = 2.7;
```

Nota: si no se van a utilizar valores con punto flotante, utilizar el tipo de dato *Integer*, ya que consume menos proceso.

BOOLEAN

Este tipo de dato alberga solo dos posibles estados *true* (verdadero) o *false* (falso).

```
var selección : Boolean = true;
var decisión : Boolean = false;
```

DURATION

Este tipo de dato representa una duración de tiempo, la cual es definida con *time literals* (literales de tiempo). Estos están formados por un número y un literal, que indica la fracción de tiempo.

```
var miliSegundo : Duration = 4ms;       //Milisegundos
var segundo : Duration = 1s;            //Segundos
var minuto : Duration = 12m;            //Minutos
var hora : Duration = 4h;               //Horas
```

VOID

Este tipo de dato es utilizado para indicar que la función no devuelve ningun tipo de valor.

```
function hazAlgo () : Void {
// Algo de código …
}
```

NULL

Este tipo de dato se utiliza para indicar una variable que apunta a una posición de memoria no válida como dato, identificando esto como que la variable no contiene valor alguno. Null permite realizar comparaciones.

```
var persona : Persona = null;
if ( persona == null ){
}
```

SEQUENCIAS

Además de los tipos de datos básicos, JavaFX provee estructuras de datos especiales llamadas *sequences* (secuencias). Estas representan listas de objetos. Los objetos contenidos dentro de la *sequence* son llamados *items* (elementos). Las *sequences* son declaradas entre "[]", y cada item es separado por una ",".

CREACIÓN DE SECUENCIAS

Hay varias maneras de declarar una secuencia:

```
var sen1 = [  "Lunes" , "Martes" , "Miercoles" ];
var sen2 : String[] = [ "Lunes" , "Martes" , "Miercoles" ];
var nums : Integer[] = [ 1 .. 100 ];
```

UTILIZANDO PREDICADOS

Es posible, también, utilizar una expresion boleana (también llamada *predicado*), para declarar una nueva secuencia como un subconjunto de una secuencia existente.

```
def seq : Integer[] = [ 1 , 2 , 3 , 4 , 5 ];
def seq2 : Integer[] = [ n | n > 2 ];
```

seq2 contiene los valores de *seq* que sean mayores a 2.

ACCESO A LAS SECUENCIAS

Una secuencia se comporta de manera similar a un array, de manera que se agrega dentro de los corchetes el númer de elemento al cual se quiere acceder. Las secuencias (al igual que los vectores), comienzan a numerarse por el cero, siendo su último elemento *tamaño secuencia -1*

```
def seq : String[] = [ "Primer valor" , "Segundo valor" , "Tercer valor" ];
println(seq[0]);                //Muestra Primer valor
println(seq[1]);                //Muestra Segundo valor
println(seq[2]);                //Muestra Tercer valor
println("{sizeof seq}");        //Muestra 3
```

INSERTAR ELEMENTOS EN LA SECUENCIA

La palabra clave *insert* permite agregar una elemento dentro de la secuencia, antes o después de un item.

```
var dias : String[] = [ "Martes" ];
//dias [ "Martes" ]
insert "Jueves" into dias;//Agrega al final de la secuencia
//dias [ "Martes","Jueves" ]
insert "Lunes" before dias[0];//Agrega antes del elemento 0
//dias [ "Lunes","Martes","Jueves" ]
insert "Miercoles" after dias[1];//Agrega luego del elemento 1
//dias [ "Lunes","Martes","Miercoles","Jueves" ]
```

ELIMINAR ELEMENTOS EN LA SECUENCIA

La palabra clave *delete* permite eliminar items de una secuencia.

```
var dias : String[] = [ "Lunes" , "Martes" , "Miercoles" , "Jueves" ];
delete "Lunes" from dias;
delete dias[1];
```

```
delete dias;
```

SECUENCIA EN REVERSA

Otra de las opciones de JavaFX nos permite cambiar el orden de los valores de una secuencia, produciendo como resultado su reversa.

```
var numeros : Integer[] = [ 1 , 2 , 3  , 4 , 5 ];
reverse numeros;
```

COMPARAR SECUENCIAS

Al comparar secuencias con el operador ==, la comparación se realiza primero a nivel de cantidad de elementos, si la cantidad es igual, se procede a examinar item por item de cada secuencia. En caso de que también estos valores concuerden, la comparación devuelve *true*, en caso contrario, devuelve *false*.

```
var numeros : Integer[] = [ 1 , 2 , 3 ];
var numeros2 : Integer[] = [ 1 , 2 , 3];
var numeros3 : Integer[] = [ 2 , 3 ];

println("{if (numeros == numeros2) "IGUALES" else "DIFERENTES"}"); //Muestra IGUALES

println("{if (numeros == numeros3) "IGUALES" else "DIFERENTES"}"); //Muestra DIFERENTES
```

SEQUENCE SLICES (PORCIONES)

Los *slices* proveen acceso a una porción de la secuencia.

```
var numeros : Integer[]  = [1 .. 10 step 2];//1,3,5,7,9
var numeros2 : Integer[] = numeros[0 .. <3];//1,3,5
var numeros3 : Integer[] = numeros [ n | n > 5];//7,9
var numeros3 : Integer[] = numeros [ 0 .. <];//1,3,5,7
```

En el primer caso, recorre los valores del 1 al 10 de 2 en 2.

En el segundo caso, copia los *items* 0 a 2 de la secuencia *numeros*.

En el tercer caso, copia los valores que sean mayores a 5.

En el cuarto caso, copia todos los *items* menos el que se encuentra en la última posición.

EXPRESIONES

Un *bloque de expresión* consiste en una lista de declaraciones o exprsiones rodeadas por "{}" y separadas por ";". El valor de un *bloque de expresión* es el valor de la última expresión. Si el *bloque de expresión* no contiene expresiones, el mismo contiene un tipo de dato *Void.*

NOTA: VAR Y DEF SON EXPRESIONES.

```
var nums = [5, 7, 3, 9];
var total = {
     var suma = 0;
     for (a in nums) { suma += a }; //Sumatoria
     suma; //La última línea representa el valor devuelto
}
println("El total es {total}.");//El total es 24.
```

IF

El operador *if* permite controlar el flujo de ejecución al verificar si una condición es verdadera o no.

```
var num : Integer = 1;
if ( num == 1 )
        println("Valía 1");
else if ( num == 2 )
        println("Valía 2");
else
        println("No valía ni 1 ni 2");
```

EXPRESIONES DE RANGO

Permiten delcarar de manera rápida una serie de numeros mediante algoritmos sencillos.

```
var nums : Integer[] = [ 1 .. 5 ];//1,2,3,4,5
var nums2 : Integer[] = [ 1 .. 5 step 2 ];//1,3,5
var nums3 : Integer[] = [ 10 .. 5 step - 1 ];//10,9,8,7,6,5
```

FOR

Otra sentencia mas de iteración.

```
var days:String[] = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"];
for (day in days) {
     println(day);
}
var cuadrados : Integer[] = for (i in [1..10]) i*i;
var capitalDays :Integer[]= for (day in days) day.toUpperCase();
```

WHILE

En JavaFX, la sentencia *do .. while* no existe.

```
var count : Integer = 0;
while (count < 10) {
    println("count == {count}");
    count++;
}
```

TRY, CATCH, FINALLY

Al igual que su predecesor, JavaFx tiene la capacidad de generar excepciones y tratarlas.

```
import java.lang.Exception;

foo();

function foo() {
     var somethingWeird : Boolean = false;

     if(somethingWeird){
          throw new Exception("Algo paso!");
     } else {
          println("Logramos ejecutarlo sin problemas");
     }
}
try {
     foo();
} catch (e: Exception) {
     println("{e.getMessage()} (pero lo atrapamos)");
} finally {
     println("Y finalmente, el bloque finally");
}
```

Al ejecutar este script obtenemos el sigiente mensaje:

```
Logramos ejecutarlo sin problemas
Y finalmente, el bloque finally
```

Esto indica que no ocurrio ninguna excepcion, pero si somethingWeird valiera true, el resultado sería:

```
Algo paso! (pero lo atrapamos)
Y finalmente, el bloque finally
```

BINDING Y TRIGGERS

La palabra *bind* asocia el valor de una variable a una expresion unida. La expresion a la cual se puede unir, puede ser una variable, un objeto, el retorno de una función, o el retorno de una expresión.

BINDING A VARIABLES Y OBJETOS

Como funciona basicamente el bind con variables:

```
var x : Integer = 0;
def y : Integer = bind x;
x = 1;
println(y); // y vale 1
x = 47;
println(y); // y vale 47
```

Automaticamente, al enlazar el valor de y a x, al modificar x, el valor de y se ve afectado de igual manera.

El siguiente fragmento representa un ejemplo de binding a objetos.

```
var myStreet = "1 Main Street";
var myCity = "Santa Clara";
var myState = "CA";
var myZip = "95050";

def address = bind Address {
     street: myStreet;
     city: myCity;
     state: myState;
     zip: myZip;
};
println("address.street == {address.street}");
myStreet = "Av. Siempre viva";
println("address.street == {address.street}");
myStreet = "1 Main Street"; //La volvemos a su estado original
def address2 = Address {
     street: bind myStreet;
     city: bind myCity;
     state: bind myState;
     zip: bind myZip;
};
println("address2.street == {address.street}");
myStreet = "Av. Siempre viva";
println("address2.street == {address.street}");
//SALIDA POR PANTALLA

address.street == 1 Main Street
address.street == Av. Siempre viva
address.street == 1 Main Street
address.street == Av. Siempre viva
```

A pesar de que los dos resultados producen el mismo efecto, la JVM de Java, los procesa de diferente manera.

En el primer caso (*bind* al objeto), todos los atributos del objeto estan enlazados, y cuando uno de ellos cambia, se genera un nuevo objeto de tipo Address, el cual es asociado a la variable *address.*

En el segundo caso (*bind* a los atributos del objeto) , cuando alguno de los atributos de este objeto se ve modificado, se regenera el objeto asociado a la variable que se modificó, pero la clase Address no es regenerada.

NOTA: PARA LA JVM, ES MENOS COSTOSO (EN CUESTIONES DE PROCESAMIENTO) LA SEGUNDA OPCIÓN (BIND A LOS ATRIBUTOS DEL OBJETO).

BINDING A FUNCIONES

Existen dos maneras de declarar una función, normal y de tipo *bound*.

```
var scale : Number = 1.0;
bound function makePoint(xPos : Number, yPos : Number) : Point {
     Point {
          x: xPos * scale
          y: yPos * scale
     }
};
//Definición de la clase Point
class Point {
     var x : Number;
     var y : Number;
};
var myX = 3.0;
var myY = 3.0;
def pt = bind makePoint(myX, myY);
println(pt.x);

myX = 10.0;
println(pt.x);

scale = 2.0;
println(pt.x);
```

Podemos notar que cuando se modifican tanto myX, myY o scale, la función makePoint es invocada, alterando los atributos de pt. Esto es posible, ya que la función fue declarada con la palabra *bound*, lo cual indica que cuando algun valor asociado a la misma cambia, la función es invocada (siempre que la función se encuentre enlazada a un elemento).

En cambio, si la función no hubiera sido declarada con *bound,* al modificar el valor de *scale*, la función no hubiera sido invocada.

```
//RESULTADOS CON LA FUNCIÓN CON BOUND
3.0
10.0
20.0
//RESULTADOS CON LA FUNCIÓN SIN BOUND
3.0
10.0
10.0 //El valor no fue afectado luego de modificar scale
```

BINDING A SECUENCIAS

```
var seq1 : Integer[] = [1..10];
def seq2 : Integer[] = bind for (item in seq1) item*2;
insert 11 into seq1;
printSeqs();

function printSeqs() {
     println("First Sequence:");
     for (i in seq1){print("{i} - ");}println("");
     println("Second Sequence:");
     for (i in seq2){println("{i} - ");}println("");
```

En este caso, generamos una nueva secuencia a partir de otra mediente binding. Luego, al modificar la secuencia de enlace (seq1), la secuencia enlazada se ve afectada de igual manera.

```
First Sequence:
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11
Second Sequence:
2 - 4 - 6 - 8 - 10 - 12 - 14 - 16 - 18 - 20 - 22
```

TRIGGERS DE REEMPLAZO

Los *triggers de reemplazo* son bloques de código que se enlazan a una variable, y se ejecutan cuando el valor de la misma se ve modificada.

```
var password = "foo" on replace oldValue {
     println("\nALERT! Password has changed!");
     println("Old Value: {oldValue}");
     println("New Value: {password}");
};

password = "bar";
```

Y la respuesta por pantalla

```
ALERT! Password has changed!
Old Value:
New Value: foo

ALERT! Password has changed!
Old Value: foo
New Value: bar
```

En este caso, el trigger es ejecutado dos veces. La primera cuando la variable es inicializada, y la segunda, cuando modificamos el valor de la variable.

ANIMACIONES

JavaFX implementa el concepto de *animación por frames (Key Frame animation)*. Esto significa que los estados de transición de las animaciones del Canvas son declarados como imagenes de comienzo y de final (key frames) del estado del gráfico en un punto de tiempo determinado. Establecidos estos dos estados, el sistema puede realizar la animación automáticamente. Puede parar, pausar, resumir, rebobinar, o repetir el movimiento siempre que se requiera.

TIMELINE

Toda animación en JavaFX ocurre dentro de una **Linea de tiempo (timeline)**. Dentro de toda animación, deberán existir al menos dos Frames.

EJEMPLO DE FUNCIONAMIENTO

```
import javafx.animation.Timeline;
import javafx.animation.KeyFrame;
import javafx.animation.Interpolator;

var x: Number;

Timeline {
  keyFrames: [
         KeyFrame{
              time: 0s
              values: x => 0.0
         },

         KeyFrame{
              time: 4s
              values: x => 158.0 tween Interpolator.LINEAR
         }
  ]
}.play();
```

La variable *time define el tiempo* transcurrido en el cual los valores asociados serán incluidos en el ciclo del objeto *TimeLine*. El operador *tween* es un literal de construcción para una lista de pares clave=>valor. Esto produce que la variable x comience en 0 y llegue al valor 158 al cabo de 4 segundos.

Además, las animaciones *KeyFrame* son objetos típicos de JavaFX, para el cual se provee una sintaxis especial para expresar la animación con la sintaxis literal estandar. La cláusula *trigger* posibilita asociar una llamada arbitraria con el key frame. El tiempo especificado como *at* es relativo al comienzo de la linea de tiempo. Esta capacidad brinda un código simplificado:

```
import javafx.animation.Timeline;
import javafx.animation.Interpolator;
import javafx.animation.KeyFrame;

var x: Number;

Timeline {
  keyFrames: [
         at (0s) {x => 0.0},
         at (4s) {x => 158.0 tween Interpolator.LINEAR}
  ]
}.play();
```

INTERPOLATOR

Cada vez que generamos un *Timeline*, y le asignamos un par de frames, estamos especificando un conjunto de estos. Por defecto, el conjunto funcionaría de la misma manera que si aplicamos una interpolación *LINEAR*, es decir, se desplaza la misma cantidad de pixeles por unidad de tiempo.

Pero también tenemos otros tipos de interpolaciones, que nos permiten agregar otros efectos a las animaciones.

- DISCRETE (Discreta)
- EASEBOTH (Acrecentando la velocidad al final y al comienzo)
- EASEIN (Acrecentando la velocidad al comienzo)
- EASEOUT (Acrecentando la velocidad al final)
- LINEAR (Constante)

SERVICIO WEB PHP WEBOOKS

Si bien el lenguaje de scripting PHP (PHP Hypertext Processor) no era el fin de investigación de este proyecto, se ha desarrollado un servicio web en dicho lenguaje. Cuyo objetivo no es otro que comunicarse con diferentes servicios web por internet para realizar búsquedas de libros. Este servicio se desarrollo con diferentes objetivos, uno de los cuales (y más importantes) fue desacoplar la aplicación JavaFX, para que esta solo se encargase de presentar los resultados de las búsquedas de libros on-line.

IDEAS QUE MOTIVARON AL DESARROLLO DEL SERVICIO WEB EN PHP

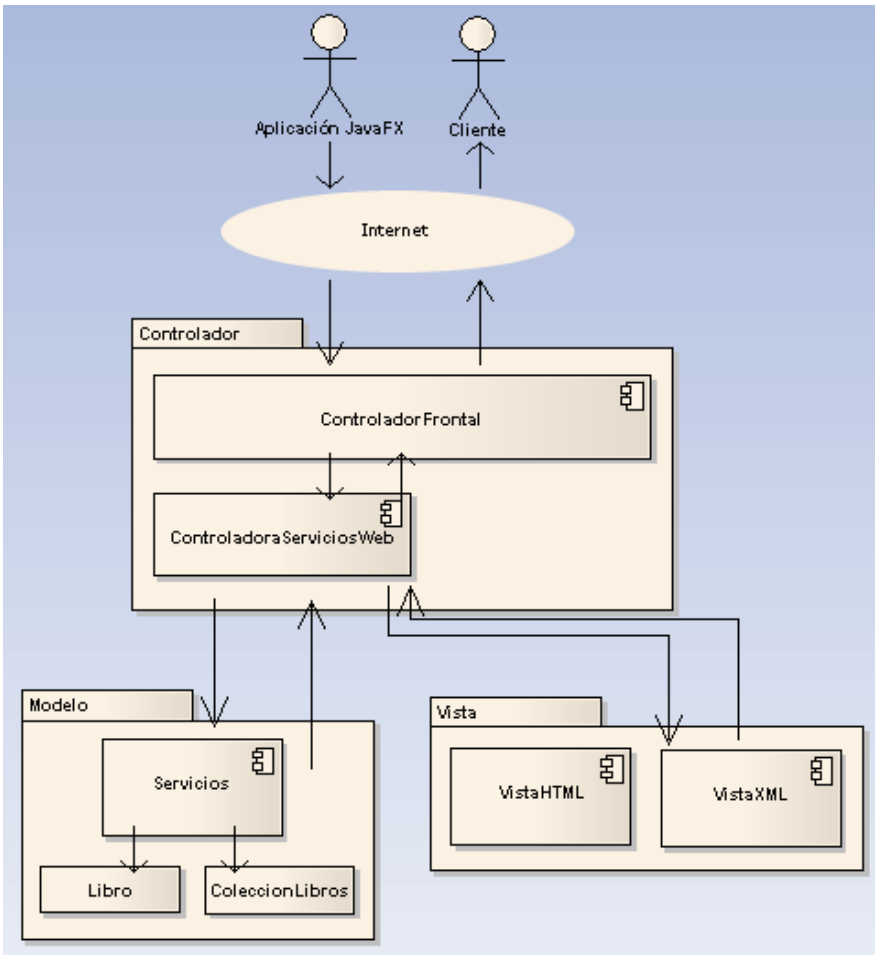
- Desacoplar la aplicación JavaFX para que solo sea responsable de presentar los datos al usuario (fin porque fue desarrollado la plataforma JavaFX). De este modo se pudo dedicar más tiempo en tratar de mejorar la experiencia del usuario en la aplicación.
- JavaFX es una tecnología realmente nueva, la cual se encuentra actualmente bajo desarrollo. Lo que implica muchas modificaciones en las API's base de la plataforma (la última versión liberada por SUN es la 1.2 en Mayo de 2009). Por este motivo no se decidió realizar las peticiones hacia los sitios de búsqueda de los datos desde la aplicación JavaFX.
- Evitar las diferentes limitaciones de cada implementación de la tecnología FX. Es decir, evitar las incompatibilidades entre las versiones desktop, web y mobile.
- Debido a que el lenguaje PHP está bien asentado, se conocía bien la plataforma para poder crear un servicio estable y con buena performance en poco tiempo.
- Debido a la flexibilidad del lenguaje PHP, la existencia de módulos útiles para el manejo del formato xml.
- Permitir que el servicio webooks pueda ser utilizado por otras aplicaciones (como podría ser una implementación web de webooks desde que cualquier lenguaje), desde otros lenguajes de programación, o incluso por terceras partes como una API funcional para la búsqueda on-line de libros.

Una de las principales ideas con las que se comenzó a desarrollar el servicio web, fue la de permitir agregar fácilmente nuevos servicios de búsqueda. Es decir, una vez que se definió el campo en el cuál se iba a realizar la implementación, se investigaron los posibles motores de búsqueda de libros que poseían API's para acceder libremente. Como resultado se obtuvieron muchos, pero se eligieron dos de los más grandes: los servicios web de Amazon y Google books.

A partir de estos servicios web se obtienen los datos de internet. El servicio webooks permite realizar búsquedas sobre estos motores, pero no se limita a ellos. Es fácilmente expansible la lista de motores de búsqueda con los que trabaja. Esto se debe gracias a que se implemento íntegramente orientado a objetos y se aplicaron diferentes patrones de diseño para mejorar la reutilización y expansión del mismo.

ARQUITECTURA DEL SERVICIO

El siguiente diagrama muestra a grandes rasgos la arquitectura del servicio web:



El servicio esta implementado mediante el patrón MVC. Donde todas las peticiones al mismo, ingresan por un único lugar ("ControladorFrontal" en donde se gestionan cuestiones de seguridad y funcionamiento general de la aplicación) y se responden por el mismo (una instancia de la Vista con ciertos valores se devuelve al usuario).

Cuando un nuevo request llega al servicio webooks se ejecutan los siguientes pasos:

1. Se instancia el Controlador Frontal (el cual se encargará de verificar el tipo de petición, en este caso XML) y se crea un objeto de la clase AutoLoader (el cual se encarga de realizar la carga automática de definiciones de clases por demanda).
2. Luego se delega el control de la petición a una instancia de la clase llamada ControladoraServiciosWeb la cuál verifica los parámetros y crea los objetos necesarios para conectarse con el servicio.
3. Los servicios son objetos que devolverán los datos (clases pertenecientes al paquete "Servicios" dentro del Modelo). Estos "Servicios" crearán objetos del modelo (ColeccionLibros o Libro) como resultados de sus peticiones a la web.
4. Una vez finalizadas las búsquedas a través de los servicios ("Amazon", "Google", etc.), la ControladoraServiciosWeb instancia una Vista correspodiente para mostrar los datos, pasando como parámetros los objetos del modelo.

5. Las vistas realizan el tratamiento de la presentación del modelo, y devuelven dicha información hacia la ControladoraServiciosWeb.
6. Se informa al ControladorFrontal la salida de la petición y se presentan los resultados.

AGREGANDO NUEVAS FUENTES DE BÚSQUEDA PARA WEBOOKS

Desde el principio cuando se comenzó el diseño de esta aplicación, se tuvo en cuenta la posibilidad de expandir el abanico de motores de búsqueda de libros con los que se trabajaba. Es decir, implementar la aplicación php de modo que cuando se finalizara su desarrollo, sea posible agregar nuevos servicios con los cuales interactuar para obtener los resultados de las búsquedas.

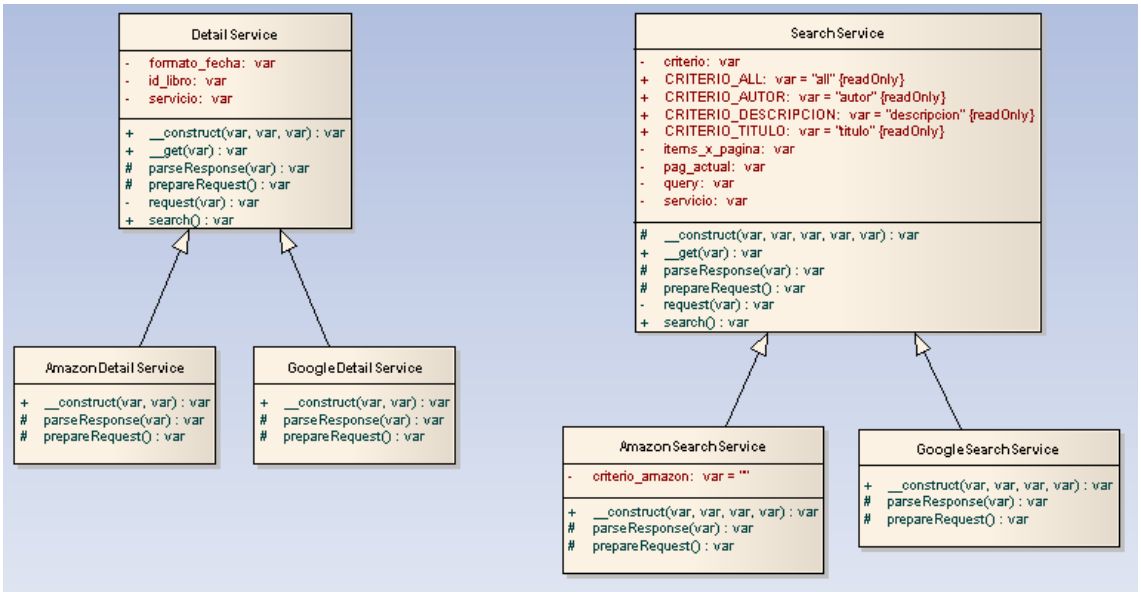
Por el motivo citado en el párrafo anterior, se diseño un xml el cuál contendría los servicios con los que trabaja la aplicación:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<root value="config">
  <services>
    <service>
      <id>amazon</id>
      <key>AKIAJ2FGT4D7EQR7HF3Q</key>
      <url_sitio>http://www.amazon.com/</url_sitio>
      <operations>
        <operation value="search">
          <url>http://ecs.amazonaws.com/onca/xml</url>
          <class_name>AmazonSearchService</class_name>
          <class_definition>/modelo/servicios/AmazonSearchService.php</class_definition>
        </operation>
        <operation value="detail">
          <url>http://ecs.amazonaws.com/onca/xml</url>
          <class_name>AmazonDetailService</class_name>
          <class_definition>/modelo/servicios/AmazonDetailService.php</class_definition>
        </operation>
      </operations>
    </service>
    <service>
      <id>google</id>
      <key></key>
      <url_sitio>http://www.google.com/</url_sitio>
      <operations>
        <operation value="search">
          <url></url>
          <class_name>GoogleSearchService</class_name>
          <class_definition>/modelo/servicios/GoogleSearchService.php</class_definition>
        </operation>
        <operation value="detail">
          <url></url>
          <class_name>GoogleDetailService</class_name>
          <class_definition>/modelo/servicios/GoogleDetailService.php</class_definition>
        </operation>
      </operations>
    </service>
  </services>
</root>
```

Gracias a la inyección de dependencia, se pueden agregar nuevos servicios de búsquedas con solamente hacer las siguientes modificaciones:

- Modificar el xml de configuración, agregando un nodo <service> con sus correspondientes datos (dentro de los cuales se encuentran dos fundamentales para webooks: "class_name" y "class_definition").
- Agregar las clases correspondientes para el parseo de los xml de respuesta y generación del xml webooks.

El siguiente es un diagrama que muestra el árbol de dependencias de estas clases y las implmentaciones para Amazon y Google:



El método search() de la clase SearchService es un "template method", el cual tiene definido el orden en que se realizarán las operaciones para comunicarse con un servicio de búqueda. Al tener definido este método, desde las clases hijas solo es necesario redefinir los métodos parseResponse() y prepareRequest(). Gracias al método search, se accederá al nuevo servicio y se obtendrán los resultados.

```php
<?php
class SearchService {
    //...

    /*
     * @return ColeccionLibros
     */
    public function search() {
        return $this->parseResponse($this->request($this->prepareRequest()));
    }
    //...
}
?>
```

Como se ve en el fragmento de código anterior el método search() realiza el parseo (invocando el método parseResponse() de la clase hija) del resultado de una petición (invocando el método request() de la clase padre) a la cual se pasa como parámetro ciertos valores relacionados a un servicio de búsqueda en especial (invocando al método prepareRequest() de la clase hija).

WEBOOKSFX

WebooksFX es el nombre de la aplicación que se desarrolló para realizar las muestras de este nuevo lenguaje. La misma fue compilada bajo la version 1.1 de la SDK de JavaFX.

El desarrollo sufrió varios cambios, ya que atravesó todas las versiones de javaFX, desde la Beta, hasta la 1.1, teniendo que modificar partes de código para volverlo compatible.

La mayor modificación que se realizó, fue el paso de la versión Beta a la versión 1.0, aquí fue donde se modificaron la mayor parte de las clases que se habían generado generado.

ARQUITECTURA DE LA APLICACIÓN

JavaFx Desktop tiene ciertas incompatibilidades con el resto de los dispositivos, de manera que la implementación de la aplicación JavaFX se aisló en dos partes:

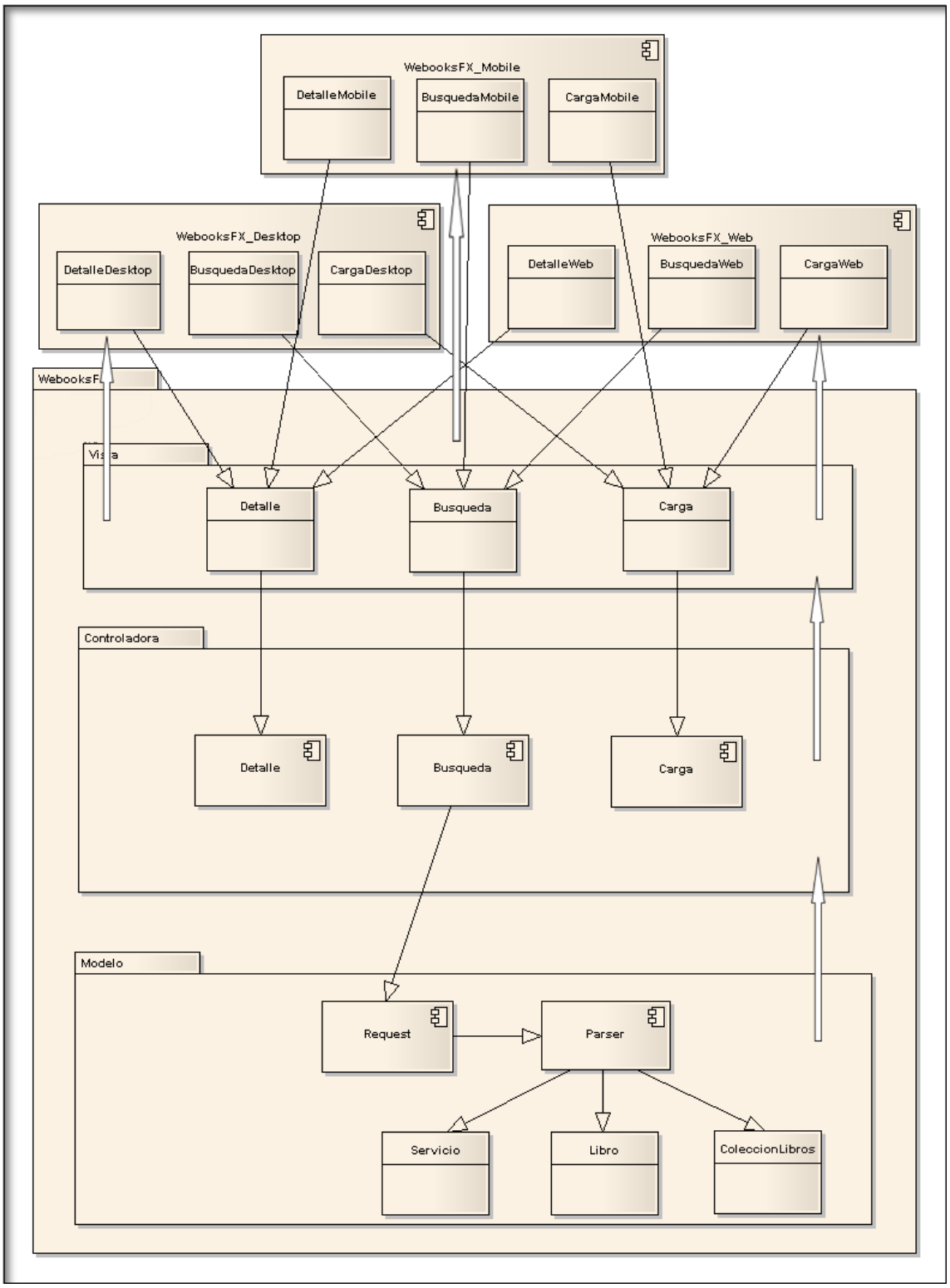
- WebooksFX
- WebooksFX_Desktop

La primera compone todo el set de clases correspondientes a la capa de Modelo, Controladora y Vista abstracta.

La segunda compone todo el set de clases correspondientes a la implementación de la Vista para una aplicación de escritorio. Todas estas heredan parte de su lógica de la Vista abstracta definida en WebooksFX.

De esta manera, si se quisiera realizar una aplicación para mobiles, solo es necesario rediseñar la vista, dejando el Modelo, la Controladora y la Vista abstracta intactas.

ESQUEMA DE LA APLICACIÓN



ARQUITECTURA MVC

VISTA

Dentro de la vista se encuentran los componentes visuales. Estos permiten que la aplicación interactue con el usuario. Esta capa no contiene lógica de negocio, solo contiene los componentes visuales, y métodos para alterar sus estados. La lógica es manejada por la controladora.

CARGANDO

Esta es la pantalla que se muestra al comienzo de la aplicación, y es la encargada de mostrar un mensaje de espera mientras se establece la conexión con el servidor para obtener los servicios disponibles, contra los cuales la aplicación podrá interactuar más tarde.

El resto de las pantallas se encuentran ocultas, de manera de dar un efecto de transparencia, para que solo sea visible la imagen del logo.

BÚSQUEDA

Esta pantalla es la que contiene las opciones para que el usuario pueda realizar una búsqueda de libros. Dentro de esta pantalla podemos encontrar: la barra de búsqueda, los resultados, y el paginador de resultados.

La barra de búsqueda nos permite ingresar un texto como parámetro de búsqueda. También posee opciones de configuración avanzadas, permitiendo filtrar por un criterio en particular, y seleccionar dentro de los servicios disponibles, cual de los motores de búsqueda utilizar.

Una vez que se realize una búsqueda, se muestra una ventana de carga para indicar que la petición ha sido enviada, y se está procesando. Cuando el procesamiento haya terminado, la ventana de carga desaparece, y se generan varios elementos gráficos, conteniendo los resultados de la petición.

A partir de este punto se puede realizar otra petición de búsqueda, o acceder al detalle de un libro. En el primer caso, se repite el proceso anterior. En el segundo, se vuelve a mostrar la ventana de carga, y se envía una petición para obtener el detalle de un libro.

DETALLE

Esta pantalla refleja todos los datos detallados de un libro en particular (Imagen, nombre, autor, editorial, sinopsis, cantidad de páginas).

CONTROLADORA

La controladora es la encargada de procesar las acciones iniciadas en la vista, y desencadenar un evento para las mismas, ya sea, derivando el proceso al modelo, y/o modificando el estado de la vista.

Por cada vista implementada, hay una controladora, de manera que en el proyecto existen 3 controladoras: ControladoraVista, ControladoraModelo y ControladoraDetalle. A su vez, algunas controladoras pueden derivar el proceso ellas.

Las controladoras también atienden la respuesta que el modelo haya procesado, alterando el estado de la vista en base al resultado obtenido.

MODELO

Capa que contiene el modelo de negocio. En el caso de la aplicación WebooksFX, es la que se encarga de realizar las peticiones al Servicio Web Webooks, y parsear la respuesta obtenida, devolviendo objetos de tipo: Libro, Servicio, o ColeccionLibros.

POR QUE MVC?

JavaFx es una tecnología orientada al desarrollo de interfaces de usuario con alto contenido multimedia y gráficamente enriquecidas, era necesario diseñar la aplicación de manera que permitiera modificar la vista, sin tener que alterar el resto de la aplicación. En caso de que se quisiera portar a otro medio (Mobiles, T.V., etc.).

Model View Controller es una arquitectura de aplicación que cumple con este requisito, de manera que se diseño en una aplicación las controladoras, y la lógica de negocios, dejando solamente una vista abstracta.

Luego, para cada medio que se quisiera desarrollar una interfaz de usuario, solo era necesario dar un comportamiento a la vista abstracta, y generar los objetos gráficos. De esta manera se obtiene portabilidad y rehusabilidad.

CONCLUSIONES

JavaFx es una herramienta potente y muy simple de utilizar para el desarrollo de aplicaciones RIA´s. Su lenguaje declarativo, y su simplicidad permiten generar aplicaciones con alto contenido multimedia en poco tiempo, cosa que con las herramientas anteriores esto era casi imposible.

JavaFx cuenta con el respaldo de Java, pudiendo utilizar una aplicación con código diseñado en JavaSE, JavaEE, o JavaME. Además, de que Java es una plataforma que se encuentra actualmente en casi todos los ordenadores, y en un 80% de los teléfonos mobiles, de manera que la distribución de las aplicaciones realizadas con estos lenguajes no representa para nada un problema.

Tiene portabilidad entre distintos S.O. Dado que Java es en realidad un lenguaje semi-compilado (compilado a Bytecodes), el cuál puede ejecutarse en cualquier ordenador que soporte la JRE (Java Runtime Enviroment) con el complemento JavaFX 1.2 (1.1 para el caso de la aplicación WebooksFX).

Fomenta la unión de desarrolladores y diseñadores para realizar proyectos, permitiendo la interacción hasta con las diferentes capas de un dibujo. De manera que cada uno, pueda concentrarse en la parte que le corresponde enriqueciendo la aplicación, y disminuyendo los tiempos totales del proyecto.

Como desventaja nos encontramos con un lenguaje que se encuentra en sus etapas iniciales. Al realizar la aplicación nos topamos con varios elementos faltantes, varios Glitches, y algunos bugs que hicieron que la aplicación pasara de ser una investigación apasionante, a un trabajo tedioso. La falta de componentes básicos y cuestiones de performance hacen que esta plataforma no este realmente lista para ser utilizada en aplicaciones grandes y ambiciosas.

No se encuentran disponibles herramientas gráficas para el desarrollo de interfaces de usuario. Esto en realidad para los desarrolladores no es una desventaja, pero si para aquellos que son diseñadores, y se encuentran familiarizados con herramientas del estilo Flash, en la cual, sabiendo poco o nada de programación, puede realizar una aplicación pequeña, o media con esfuerzo.

El motor de renderizado es bastante pobre. Dentro de la herramienta encontramos varios efectos coloridos, y muy llamativos, pero al aplicar lineas de tiempo, para poder producir un efecto de animación, vemos como los cuadros de reproducción no son tan fluidos, o tienen saltos bastante molestos a la vista, lo cual es un punto muy en contra para una aplicación con contenido puramente gráfico. Esto posiblemente se deba a que el mismo es realmente nuevo.

En resumen, creemos que en un futuro, JavaFx sera una plataforma que dará mucho de que hablar, y permitirá realizar aplicaciones, que no solo serán visualmente impactantes, sino que también, posibilitará desarrollar aplicaciones de alta calidad y portables, compitiendo de lleno contra los actuales líderes del mercado en tema de RIA´s, como Flash y Silverligth.

A partir de esta investigación se espera un gran futuro para JavaFX, y una mayor expansión de la plataforma Java en todos sus sentidos.

BIBLIOGRAFÍA

http://www.javapassion.com/javafx/

http://java.sun.com/javafx/

http://developers.sun.com/learning/javaoneonline/

http://weblogs.java.net/blog/joshy/archive/2007/09/javafx_javafx_s.html

http://javafx.com/

http://learnjavafx.typepad.com/

JavaFX Script "Dynamic Java Scripting for Rich Internet/Client-Side Applications"
Autor: James L. Weaver, Editorial: Apress, Octubre del 2007