Unidad Didáctica V.- Desarrollo de aplicaciones de escritorio con JDeveloper

SI Andalucía *se está consolidando en el mercado como una de las empresas más punteras del sector.* ***Jesús, María y José*** *como socios de la empresa se encuentran muy orgullosos de lo que han conseguido y así se lo hacen saber al resto de personal, particularmente a* ***Carmen*** *y* ***Víctor*** *que han trabajado muy duro y que están muy involucrados en la investigación y desarrollo de nuevas tecnologías innovadoras para mejorar e incrementar la calidad de servicio hacia los clientes.*

Todo esto ha llegado a oídos del equipo directivo de ***El PC Feliz****, un taller de reparación de equipos y material informático con gran implantación en la zona, que no han dudado en solicitar sus servicios para el desarrollo de una aplicación de gestión a medida. Los directivos de El PC Feliz confían en haber hecho una buena elección de proveedor y desean que el proyecto se lleve a cabo con éxito y en los plazos establecidos.*

El equipo al completo de SI Andalucía se ha puesto manos a la obra. Esta cuenta es un cliente por el que llevaban luchando hace muchos meses frente a su principal competidor, ***KiloSoftware S.A.,*** *por lo que todo debe salir a la perfección. Además, hay otras empresas del sector que sin duda se podrían mostrar interesadas por la aplicación una vez terminada, en SI Andalucía son conscientes que no hay mejor publicidad que la de un cliente satisfecho.*

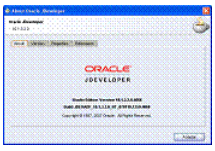
El análisis y diseño de la aplicación ha sido llevado a cabo por ***María****, que es una experta en el tema, y ha dado como resultado un extenso informe que permitirá desarrollar la aplicación en unos plazos razonables, con una estimación de recursos y, muy importante, una indicación de los requisitos y funcionalidad que deben contemplar.*

El proyecto le ha sido encomendado a ***Carmen****, quien dispondrá de la inestimable colaboración de* ***Víctor*** *para que la ayude en tan ardua tarea. Ambos han estado repasando el documento de análisis para conocer más a fondo los requisitos y la funcionalidad de la aplicación que deben desarrollar.*

La aplicación debe llevar a cabo las funciones de Gestión de un Taller de reparaciones y ventas de piezas de material informático. Además del documento de análisis, ***Carmen y Víctor*** *han tenido acceso al diseño de la base de datos y han podido comprobar que el principal proceso que deben contemplar es el de Facturación. Una factura consiste en una serie de líneas de detalle donde vienen especificadas una serie de piezas, que se venden por separado o como consecuencia de una reparación fuera de garantía, así como el importe de los servicios realizados si los hubiera. Asimismo, la aplicación también debe contemplar el registro de las Reparaciones que se lleven a cabo por los empleados y que pueden acarrear el uso de varias piezas. Cada reparación puede requerir una serie de acciones (por ejemplo: formatear, sustituir pieza, etc.).*

También se deberá contemplar el mantenimiento de los clientes de ***EL PC Feliz*** *y las ventas que se hayan realizado para poder obtener posteriores informes de facturación por cliente, reparaciones y/o acciones realizadas.*

Víctor, *el compañero de más reciente incorporación, se encuentra muy ilusionado por ser de uno de los proyectos de más envergadura que va a abordar desde que comenzó en la empresa y desea estar a la altura de todo lo que se espere de él en este aspecto. Así que ha decidido instalarse inmediatamente el entorno JDeveloper y el gestor de base de datos Oracle para no perder más tiempo en conocer todos los detalles de estas herramientas. Ya se ha introducido en el manejo del IDE y los conceptos de patrón MVC para poder estructurar mejor las aplicaciones y reutilizar componentes en futuros desarrollos. El siguiente paso es desarrollar con la ayuda de* ***Carmen*** *una primera aproximación de aplicación de escritorio para* ***El PC Feliz****.*

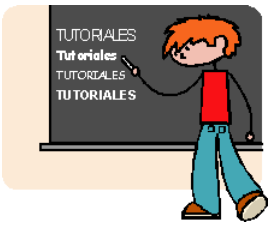


Llegados a este punto tenemos los conocimientos previos necesarios para poder abordar el desarrollo una aplicación de escritorio. El módulo ha comenzado con una visión general y teoría de las interfaces, en la unidad 2 hemos conocido el IDE JDeveloper y en la unidad 3 nos hemos introducido en la filosofía del modelo vista-controlador MVC, tan importante a la hora de desarrollar software que sea reutilizable y que cumpla los cánones establecidos. Asimismo, en la unidad 4 hemos adquirido los conocimientos necesarios para diseñar nuestra base de datos y usarla en nuestra aplicación.

El módulo que nos ocupa, Diseño y Realización de Servicios de Presentación en Entornos Gráficos, está muy relacionado con Programación en Lenguajes Estructurados (PLE), módulo correspondiente al 1er curso de Desarrollo de Aplicaciones Informáticas. Por tanto, sería recomendable tener acceso a los contenidos de dicho módulo a modo de recordatorio porque probablemente necesites consultar aspectos relativos a la programación en el lenguaje Java. En unidades anteriores, y más adelante en este mismo apartado, se te proporcionaban algunos enlaces donde poder aprender algo de Java antes de abordar este módulo, por si no has cursado aún el módulo de PLE, ya que es necesario tener

unos conocimientos bastante buenos de fundamentos de programación, y de Java, para poder terminarlo con éxito.

Esta unidad desarrolla, a través de una metodología completamente práctica, una aplicación de escritorio utilizando el entorno JDeveloper, y Oracle 10g Express Edition como gestor de base de datos. Te recomendamos que le eches un vistazo a la tarea de esta unidad porque quizás te interese ir realizándola conforme vas avanzando en cada apartado, siguiendo las indicaciones que se facilitan.



Algo imprescindible en esta unidad son los tutoriales incluidos como recursos, pues contienen los conocimientos necesarios para poder desarrollar todas las capacidades que se espera que adquieras en esta unidad, mediante la realización de la tarea. Estos tutoriales utilizan un método gráfico e intuitivo y su objetivo es guiarte durante todo el aprendizaje, y obtener como resultado una sencilla aplicación en JDeveloper que luego deberás aplicar al problema propuesto en la tarea y completar con los apartados que se te pidan. Por tanto, te recomendamos que le dediques el tiempo necesario a visualizar los tutoriales, e incluso contemples la posibilidad de ir realizando a la vez los pasos sobre tu aplicación, algo que adelantarás para la tarea y que esperemos te resulte lo más cómodo posible, ya que al ser archivos .swf podrás volver atrás o pausar la demostración siempre que necesites repasar algún concepto.

PARA SABER MÁS

Página oficial de Sun Microsystem, empresa creadora de Java donde podemos encontrar toda la información oficial de Java:

Sun Microsystems español

Página para desarrolladores de Java, con acceso a tutoriales, documentación y códigos de ejemplo (en inglés):

Página Web de Java

Página oficial de Oracle JDeveloper donde accederás a todas las novedades sobre el entorno integrado de Oracle JDeveloper y página de tutoriales e instructivas demostraciones, seguramente ya habrás necesitado consultarlas en unidades anteriores (en inglés):

Pagina oficial de Oracle JDeveloper

Tutoriales Oracle JDeveloper

Y por último te presentamos una página en español sobre JDeveloper donde puedes encontrar algunos artículos interesantes sobre la herramienta:

Página sobre JDeveloper

__Víctor__ se encuentra impaciente por comenzar a desarrollar la aplicación. Necesita las indicaciones de __Carmen__ para detallar algunos aspectos como las características o técnicas que se deben utilizar, dentro de todas las que ofrece JDeveloper. Ha intentado por sí solo crear la aplicación, un paso sencillo, pero se encuentra con un sinnúmero de tecnologías ¿JSF, JSP, Swing ...? y aunque ya está familiarizado con estos conceptos prefiere andar con paso lento pero firme. Tras ponerlo en común, deciden que van a crear una Aplicación Java de Escritorio con Swing y ADF Business Components, éste último término a __Víctor__ no le suena y se muestra interesado, __Carmen__ le tranquiliza "todo a su tiempo Víctor, todo a su tiempo: primero debes crear la aplicación y su interfaz antes de conectarnos con la base de datos y utilizar los ADF Business Components".

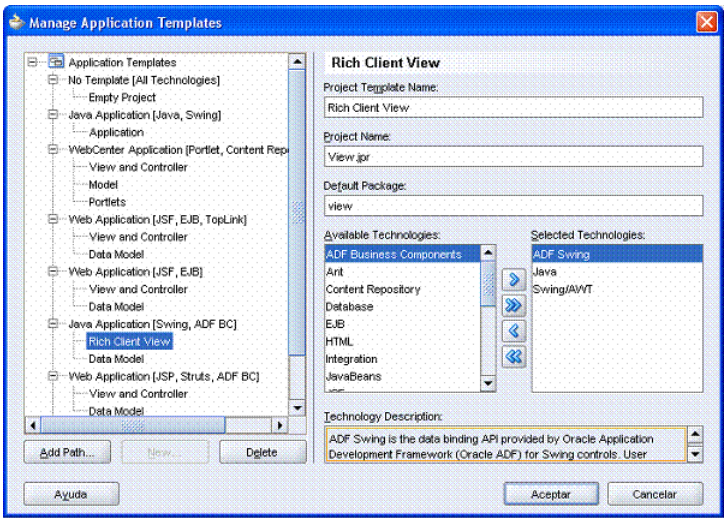
Antes de crear los componentes de la aplicación, debemos primero crear la aplicación siguiendo el modelo vista-controlador. Para ello, desde el navegador de objetos, haz clic con el botón derecho sobre la carpeta **Applications** y elige "**New Application**". En el cuadro de diálogo "**Create Java Application**", introduce los valores que aparecen en la tabla.

Como vimos en la unidad 3, la plantilla que debemos elegir es **Java Application [Swing, ADF BC],** para que nuestra aplicación sea creada con dos proyectos por defecto: **Model y View**, siguiendo la arquitectura MVC. En dicha unidad ya aprendimos que el modelo de un componente está donde están almacenados sus datos, por ejemplo el estado de un botón (pulsado, bajo el ratón, etc.) o los valores de una lista. La vista es la representación en pantalla del componente, por ejemplo la forma (color, bordes, etc.) en que aparece un botón o una lista. Finalmente, el controlador es la parte del componente que gestiona la entrada, por ejemplo el código que describe qué hacer cuando hacemos clic sobre el botón, también llamado manejador del evento. Vamos a utilizar esta filosofía a lo largo de toda la unidad para dividir nuestra aplicación en vistas y documentos

Name	GestionTaller
Directory name	Elige el directorio donde quieras guardar la aplicación
Application template:	Java Application [Swing ADF BC]

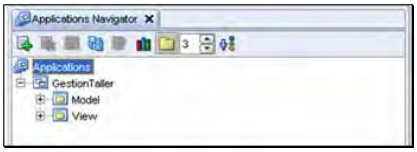
La plantilla de aplicación [Swing, ADF BC], podemos decir que se utiliza para construir aplicaciones rápidas basadas en ADF Swing. Los componentes ADF Business Components se utilizan para construir el modelo de datos. Si continúas en la ventana de "**Create Application**", puedes hacer clic en el botón "**Manage Templates**" si tienes curiosidad por saber qué son estos conceptos.

En la parte derecha de la ventana "**Manage Application Templates**" encontrarás las tecnologías que por defecto se le asignan a la plantilla **[Swing, ADF BC]**. Si seleccionas cualquiera de ellas podrás ver una descripción de dicha tecnología en **Technology Description**.



Cuadro de diálogo "Manage Application Templates"

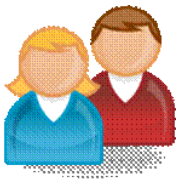
Por tanto, **ADF Swing** es una capa que enlaza los componentes Swing estándar con la capa de **Oracle Application Development Framework (ADF)**. Se usa para construir aplicaciones cliente que requieren una respuesta inmediata hacia el usuario o los eventos que cambian la pantalla del usuario, y que encuentran mejor soporte en Swing que en la Web. Otra razón para construir clientes Swing es cuando, por razones de negocio, las aplicaciones necesitan trabajar off-line, desconectadas de la red. **ADF Business Components** gobierna la interacción entre el resto de la aplicación y los datos almacenados en la base de datos, proveyendo de servicios de validación y otras lógicas de negocio.



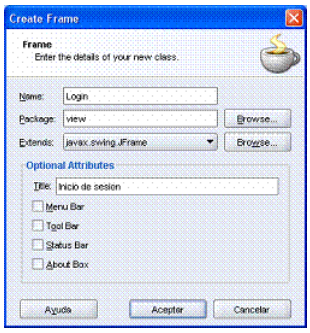
Veremos más conceptos sobre estas tecnologías posteriormente, por ahora lo que vamos a hacer es empezar a crear la interfaz gráfica de nuestra aplicación.

Una vez que hemos visto las tecnologías que vamos a aplicar en nuestra aplicación, no vamos a modificarlas. Posteriormente durante la etapa de desarrollo podremos añadir otras tecnologías si lo necesitáramos. Volvemos a la pantalla anterior dándole a Cancelar. Seguidamente aceptamos la ventana de **Create Application** y nos aparecerá el entorno de JDeveloper con la estructura de nuestra aplicación creada. Acabamos de crear nuestra aplicación **GestionTaller**.

Ventana de login



Nuestro primer formulario va a ser la **ventana de login** de la aplicación. Ahora mismo no nos vamos a complicar demasiado y lo único que hará este formulario es comprobar que el usuario y contraseña coinciden con uno dado. Posteriormente veremos que esto se puede mejorar incorporando un contador de intentos fallidos o, lo que es más importante, validando el usuario con la base de datos para permitir la entrada de varios usuarios a la aplicación, cada uno con sus correspondientes permisos. Pero como hemos dicho esto lo veremos después, ahora vamos a lo fácil que para eso estamos empezando con nuestra aplicación.



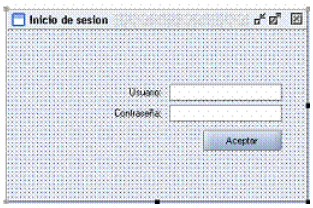
Para crear el formulario nos dirigimos al navegador de objetos, hacemos **clic con el botón derecho** sobre la carpeta **View** de nuestra aplicación y elegimos **New**. Dentro de la categoría **Client Tier (Swing/AWT** elegimos el ítem **Frame** para crear un fichero fuente de tipo **JFrame**. En la ventana siguiente introducimos **Login** como **nombre del formulario** e **Inicio de sesión** en el atributo **título** y pulsamos **Aceptar**.

A la hora de diseñar aplicaciones para entornos gráficos, si bien es muy importante obtener una correcta funcionalidad y ajustarnos a los requisitos del cliente, igualmente cobra importancia la interfaz de nuestra aplicación, como intermediaria entre el usuario y las distintas funciones implementadas. No tendría sentido una aplicación que contemplando toda la funcionalidad requerida, obligara al usuario a un sinnúmero de pasos para realizar una acción que pudiera resumirse en unos pocos, o una interfaz que no contemple el uso del teclado a la hora de interaccionar con el usuario,... Desde el momento en que presentamos un producto terminado, todos los detalles son importantes, y dentro de ello la interfaz que es lo primero que ve nuestro cliente y la imagen de la aplicación. Por eso no sólo nos vamos a centrar en que todo funcione correctamente sino, como corresponde al módulo que nos ocupa, en obtener una interfaz gráfica con las siguientes características:

- **Amigable**: que facilite la tarea al usuario, muestre mensajes de error adecuados, proporcione ayuda sensible al contexto como los tooltips que proporcionan ayuda sin necesidad de que el usuario la solicite, agrupe la información para reducir la cantidad de información a memorizar o cualquier otro tipo de ayuda para hacer más fácil el uso de la misma.
- **Agradable**: utilizar colores suaves y evitar los diseños estridentes de mal gusto, alinear correctamente los elementos, etc.
- **Intuitiva**: lo más fácil es situar los botones, menús, etc., de forma similar a los que el usuario esté acostumbrado, o sea, basarse en las características generales de cualquier aplicación Windows o Linux.
- **Homogénea**: toda la interfaz de nuestra aplicación debe guardar el mismo aspecto, por ejemplo, utilizando distintas tonalidades dentro del mismo color para los fondos, utilizar el mismo tipo de letra para objetos similares, etc.

Comentarios y legibilidad del código: Un aspecto interno de nuestra aplicación relativo a la forma de programar es que nuestros formularios sólo deben tener el código estrictamente necesario para que funcionen

y desarrollen las capacidades previstas. En ocasiones JDeveloper puede generar código automático que nos ayuda pero otras veces sólo entorpece la lectura, cuando nos encontremos con este tipo de situaciones eliminaremos el código sobrante para hacer más legible nuestro código fuente. Además, utilizaremos los comentarios adecuadamente tanto para comentar expresiones complicadas o especiales como para describir la funcionalidad de los métodos o para introducir información adicional como creador del programa, fecha, copyright,... etc. En este apartado también deberemos cuidar la organización de nuestra aplicación: evitar archivos innecesarios, utilizar nombres significativos para las clases, etc.



Estos aspectos, como recordarás, los hemos tratado con más profundidad en las primeras unidades del módulo, pero no está de más volver a tenerlos en cuenta. Si los tenemos siempre presentes conseguiremos que nuestra aplicación tenga un aspecto más serio y profesional.

Seguimos con nuestra **ventana de Login**. En primer lugar vamos a utilizar el Inspector de Propiedades para modificar la propiedad **Size** a **365,233**.

Seguidamente, vamos a crear una serie de **componentes Swing** arrastrándolos hacia el formulario desde la Paleta de Componentes situada en la parte derecha de la aplicación. Como sabes, la clase Swing está basada en AWT pero incorpora más posibilidades de diseño de interfaces gráficas y consume menos recursos (para recordar esto un poco, puedes ver la unidad de PLE que introducía Swing y AWT, o consultar alguno de los tutoriales sobre Swing que te hemos propuesto en unidades anteriores). Estos controles, que nos van a servir para la introducción del usuario y la contraseña, tienen las propiedades siguientes:

Nombre	Tipo	Size	Bounds	Text	Horizontal Alignment	tooltipText
txUsuario	JTextField	135,20	195, 65, 135, 20			Introduzca un usuario valido
txPass	JPasswordField	135,20	195, 90, 135, 20			Introduzca contraseña
lbUsuario	JLabel	55,20		Usuario:	Right	
lbPass	JLabel	75,20		Contraseña:	Right	
btAceptar	JButton	95,25				

Para alinear el **lbUsuario** a la misma altura que está **txUsuario** los seleccionamos ambos, primero uno y luego otro con la tecla **Ctrl** pulsada y elegimos **Align/Top** del menú contextual que aparece al hacer clic con el botón derecho. Igual hacemos con los controles **lbPass** y **txPass**. Por último, alineamos a la derecha los controles **lbUsuario** y **lbPass** con la opción **Align/Right** y el control **btAceptar** a la derecha con los cuadros de texto.

En ocasiones, para propiedades que tienen el mismo valor podemos seleccionar todos los controles con la tecla **Ctrl** y aplicarles la propiedad a todos a la vez. Por ejemplo, pruébalo para establecer la fuente **Ms Sans Serif, 11** en todos los controles del formulario.

Utilizar los prefijos **tx** y **bt** en el nombre de los controles facilita que luego en el código cuando los estemos utilizando tengamos una rápida información sobre el tipo de control de que se trata.

Iconos

Ahora vamos a asignar un icono a la ventana que acabamos de crear, o sea, una imagen que aparezca a la izquierda de la barra de título y en la barra de tareas cuando esté minimizada.

Para conseguir esto, hemos de introducir una serie de líneas de código en nuestra **clase Login**. Así que nos vamos a la parte inferior de el área donde estamos diseñando, donde pone **Source, Desgin, History** y elegimos la primera etiqueta (**Source**). Nos aparecerá el código de nuestra ventana **Login**, que actualmente tan sólo debe estar formado por el constructor de la clase y por el método **jbInit()**, que como sabes se ejecuta cada vez que se invoca el constructor, o sea, cada vez que tiene lugar una sentencia ... **new Login()**.

A continuación, introducimos un nuevo atributo en nuestra clase llamado ***icono***, y una nueva sentencia en el método **jbInit*()*** del formulario **Login**, debe quedar algo como lo que sigue (en negrita lo que debemos introducir):

```
public class Login extends JFrame {
    private JTextField jtxUsuario = new JTextField();
    private JTextField jtxPass = new JTextField();
    ...
    private ImageIcon icono = new ImageIcon(getClass().getResource("../http://www.juntadeandalucia.es/educacion/elearni
    private Image miImagen=icono.getImage();
    public Login() {
    ...
    }
   private void jbInit() throws Exception {
    ...
       this.setIconImage(miImagen);
    ...
   }
```

Por supuesto, tanto la imagen como el directorio deben existir. Si te fijas, la imagen está guardada en un directorio llamado **images** que está dentro del directorio **src**, al mismo nivel de jerarquía que la carpeta **view** de nuestro proyecto.

Al introducir el código, JDeveloper te indicará mediante un globito rojo en la parte izquierda de la ventana los paquetes que son necesarios importar para las clases que estés utilizando, **javax.swing.ImageIcon** y **java.awt.Image** para las clases **ImageIcon** e **Image**, respectivamente.

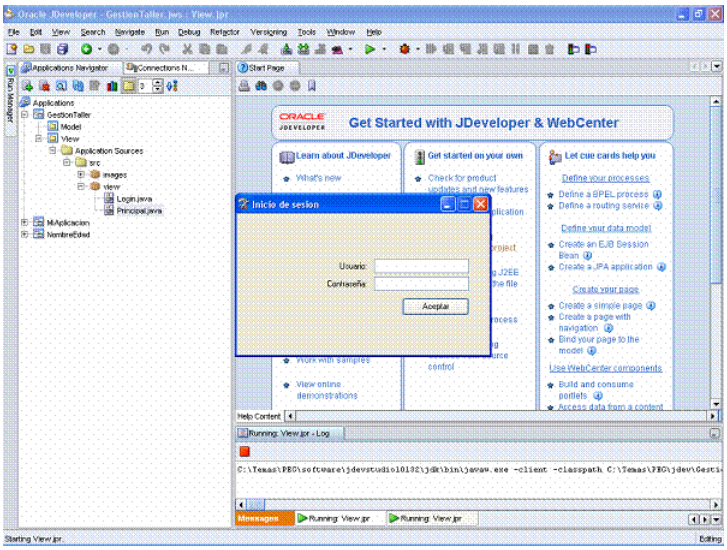
Todo esto queda reflejado en modo visual si nos vamos a las propiedades del formulario, veremos que en la propiedad **iconImage** aparece **miImagen** como icono de la barra de título.

Clase principal

Una vez que hemos creado nuestra ventana de entrada necesitamos añadir una clase principal que contenga la aplicación y que tenga asociada la interfaz gráfica anterior. Para ello hay que hacer **File/New** y dentro de **Client Tier (Swing/AWT** escoger **Java Application**, darle un nombre (por ejemplo **Principal**), e indicar que se le desea añadir un frame por defecto.

Seleccionar la opción "**Existing frame**" y, tras pulsar el botón "**Browse**" escribir el nombre del frame con la interfaz gráfica, que en este caso es **Login**. Conviene dejar activa la caja de comprobación "**Center frame on screen**" para que el frame aparezca justo en el centro del monitor cuando se ejecute la aplicación.

Ya estamos preparados para ejecutar la aplicación. Para ello seleccionamos previamente en el navegador de objetos la clase con la aplicación (**Principal**), y hacemos **Run**.



Ventana de Login en ejecución

Debes recordar que la imagen que indicaste como icono de la ventana debe existir. Podría ser un buen momento para hacerte con una lista de páginas Web donde poder obtener iconos, imágenes o gráficos interesantes para tu aplicación. Las que nosotros te recomendamos las puedes encontrar en el apartado Para saber más.

Autoevaluación

¿Cuáles son los aspectos que definirías como más importantes a la hora de desarrollar nuestra aplicación?

a) Que la aplicación funcione es lo único importante y fundamental.
b) La funcionalidad de la aplicación y ajuste a requerimientos, así como que tenga unas características deseables a nivel de interfaz y estructura.
c) Que la interfaz sea amigable, agradable, intuitiva, homogénea.

Comprobar

¿Cómo se desactivaba el botón de maximizar de un formulario?

a) this.setResizable(false);
b) this.Maximize=false;
c) this.setPreferedSize=0;

Comprobar

PARA SABER MÁS

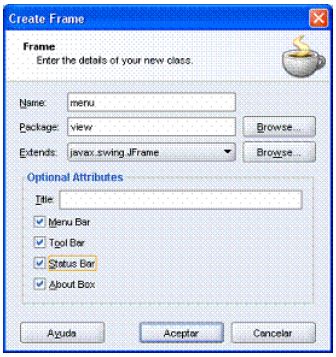
Página dedicada a recursos de Visual Basic con iconos y gráficos válidos para cualquier lenguaje:

Conjunto de iconos

En esta página puedes encontrar todo tipo de iconos en formato *.ico:

Iconos - La Web del Programador
Artículo donde aparecen enlaces de sitios dedicados al diseño de iconos freeware en varios formatos:
Diez sitios para encontrar iconos gratuitos

__Víctor__ en muy poco tiempo tiene lista la ventana de login de la aplicación y ha comprobado que se muestra perfectamente al ejecutar la aplicación en JDeveloper. Pero quiere tener algo más avanzada la interfaz antes de mostrársela a Carmen y continuar con el desarrollo. Lo siguiente que debe hacer es crear un menú de la aplicación con los comandos básicos y, cómo no, una pantalla de bienvenida inicial.



En los apartados anteriores hemos creado una ventana de Login para nuestra aplicación y una clase lanzadora llamada **Principal**. Ahora necesitamos crear la interfaz gráfica de la que constará nuestra aplicación, que aparecerá una vez que el usuario haya introducido los datos de acceso correctos.

Para ello vamos a utilizar el IDE JDeveloper para crear la **ventana de menú** y la típica **ventana de "Acerca de..."** prácticamente de forma automática. Esto lo conseguimos haciendo clic con el botón derecho sobre **View** o clic en el menú ***File***, escogiendo la opción de **New** y dentro de **Client Tier (Swing/AWT**. Así podemos crear un nuevo frame y escoger si queremos que tenga barra de menú, barra de herramientas, barra de estado o cuadro de diálogo "Acerca de". Elegiremos todas las opciones.

Rápidamente podemos comprobar cómo se han creado dos nuevas clases, **menu.java** y **menu_AboutBoxPanel1.java**. Más tarde personalizaremos los elementos de nuestro menú. Ahora sólo te recomendamos que cambies las imágenes que se han creado de sitio y las introduzcas en la carpeta **Images** que está al mismo nivel de jerarquía que la carpeta ***view***de nuestro proyecto. ¿Sabrías qué parte del código de la clase **menú.java** hay que modificar para que haga referencia a la nueva ruta de las imágenes?

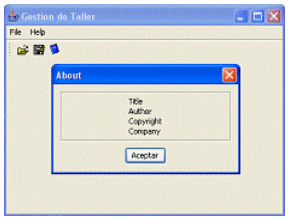
Seguidamente introduciremos el siguiente código en el evento **MousedClicked** del control **btAceptar** de la ventana de Login:

```
try   {
        boolean acceso =this.hacerLogin(this.txUsuario.getText(), this.txPass.getText());
        if (!acceso) {
        this.lbAcceso.setText("Acceso denegado");
        } else {
        menu ppal = new menu();      // creamos la ventana ppal que es de tipo menú
        ppal.setVisible(true);     // y la visualizamos
this.setVisible(false);     // ocultamos la ventana de login
        }
} catch (Exception ex) { ex.printStackTrace(); }
```

De esta forma, cuando el usuario introduzca los datos de acceso correctos y le dé al botón **Aceptar**, se lanzará la clase **menu.java** que contiene la interfaz de la aplicación. El método **hacerLogin** lo introduciremos dentro de la clase **Login** y lo único que hará por ahora será validar el usuario y la contraseña:

```
public boolean hacerLogin (String u, String p)  {
if (u.equals("invitado") && p.equals("invitado")) return true;
return false;
}
```

Nos vamos a la clase Principal y con el botón ▶ ejecutamos la aplicación, o bien con F11. Introducimos el usuario y contraseña (invitado - invitado) y comprobamos cómo al darle a ***Aceptar*** se muestra la interfaz gráfica, y también el cuadro de información que se obtiene al pulsar el menú **Help**.



Observarás que la ventana de menú se ha visualizado en tamaño pequeño y en la esquina superior izquierda de la pantalla. Nos interesa que se visualice de forma maximizada y que, si se restaura, aparezca en el centro de la pantalla. Esto nos va a interesar para todas las ventanas que hagamos, por lo que decidimos crear una nueva clase cuyo código puedan utilizar todas las ventanas que lo necesiten. Llamamos a la clase **pantalla** y el código puede ser algo como lo siguiente:

```
public class pantalla {
    public pantalla() {
```

```
    }
    static void centrar(JFrame f) {
      // Calculo el tamaño de la pantalla y de la ventana
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = f.getSize();
      // Si es mayor, igualo el tamaño de la ventana al de la pantalla
        if (frameSize.height > screenSize.height) {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
            frameSize.width = screenSize.width;
        }
      // Sitúo la ventana en el centro de la pantalla
        f.setLocation( ( screenSize.width - frameSize.width ) / 2, (screenSize.height - frameSize.height ) / 2 );
        f.setVisible(true);
    }
        static void maximizar(JFrame f){
        int state = f.getExtendedState();
        state |= f.MAXIMIZED_BOTH;
      // Maximizamos el frame
        f.setExtendedState(state);
    }
}
```

Podemos utilizar este código cada vez que creemos una ventana nueva, tan sólo tendremos que hacer:

`pantalla.centrar(miventana)` o **`pantalla.maximizar(miventana)`**.

El resultado es que hemos creado una ventana de menú que aparece centrada y maximizada cuando el usuario introduce los datos correctos de acceso, y un cuadro "**Acerca de**". También hemos organizado las imágenes en la carpeta **images**. Todos los pasos de este apartado los tienes en la presentación "Creación de la interfaz gráfica". Al comienzo de la presentación se te muestran las clases que tiene hasta este momento nuestra aplicación: Principal.java y Login.java, para luego continuar creando una nueva ventana llamada "Gestion de Taller"

Creación de la interfaz gráfica

PARA SABER MÁS

¿Conoces los siguientes enlaces? En ellos encontrarás una muy buena profundización sobre Swing, los dos son muy interesantes. El interés de la versión en inglés radica en que se trata del tutorial original elaborado por la propia empresa desarrolladora de Java.

Tutorial en español de Swing

Tutorial de Sun en inglés sobre Swing

Cambios en la interfaz

Llegados a este punto tenemos la ventana menú y el cuadro "Acerca de" que ha creado JDeveloper. Ahora es el momento de hacer los cambios necesarios para adaptar estas ventanas a nuestra aplicación y crear nuevas ventanas para dotar de contenido y consistencia a nuestro proyecto.

Vamos a realizar algunos cambios en nuestra aplicación. Estos cambios podremos realizarlos dirigiéndonos a la pestaña **Source** del formulario que queramos modificar, o por ejemplo arrastrando desde el grupo **Swing** o **Swing Containers** de la Paleta de Componentes, si lo que queremos es crear nuevos controles y/o objetos. Utilizaremos lo aprendido en PLE, para saber lo que hace cada componente swing y así saber cuál queremos utilizar. Algunos de los cambios que podemos hacer a nuestra aplicación son los siguientes:

- Poner en español los nombres de los menús, el texto tooltip, los nombres de los controles para que sean más descriptivos, etc.
- Introducir códigos mnemotécnicos para facilitar el acceso a los menús (estos códigos aparecerán cuando el usuario pulse la tecla **Alt** y permitirán acceder a los elementos del menú pulsando la tecla del carácter subrayado)
- Modificar botones existentes o los menús y crear nuevos. Por ejemplo, en el botón de ayuda del contenedor JToolbar (la barra de herramientas de la ventana menú) especificar que la acción a realizar es que abra la ventana de "acerca de", o cambiar las imágenes de los botones y su utilidad.
- Además, vamos utilizar el método JOptionPane.showMessageDialog para mostrar un cuadro de diálogo de tipo mensaje.

¿Se te ocurren más modificaciones para mejorar la interfaz? Ánimo, ponte con ellas. En el siguiente recurso tienes algunas modificaciones

que hemos realizado a la aplicación Gestión de Taller, pero seguro que necesitaría muchas más:

Modificación de la interfaz gráfica

PARA SABER MÁS (información complementaria)

Te recomendamos que tengas a mano la información sobre componentes Swing que se te facilita en el módulo de Programación en Lenguajes Estructurados, te será muy útil a la hora de decidir qué componente utilizar para tu aplicación, particularmente las siguientes unidades:

Componentes Swing

Apartados complemetarios: "Introducción de interactividad. Componentes Swing básicos" y "Más componentes Swing: menús, JFileChooser, JInternalFrame, cuadros de diálogo, etc." que encontrarás en esta unidad.

Pantalla de bienvenida

La pantalla de bienvenida o ventana splash es una práctica habitual en las aplicaciones de hoy en día. Usar una **ventana splash** es una forma de avisar al usuario de que algo está ocurriendo en la aplicación durante el periodo de arranque.

- Algunas ventanas splash pueden ser cerradas pinchando en cualquier lugar de la ventana,
- otras se cierran solas y
- otras permanecen después que la aplicación haya comenzado.

Lo ideal sería lanzar la ventana splash y la ventana de login al mismo tiempo, así la segunda se cargaría mientras la ventana splash avisa al usuario de que se está cargando la aplicación, pero eso requiere utilizar **threads** y complica un poco el código, así que para facilitar las cosas nosotros vamos a lanzar primero la ventana splash y después la ventana de login.

En primer lugar vamos a crear un **JPanel** (File/New/Swing/AWT (Panel) que contenga lo que queremos que muestre la ventana splash, que va a ser tan sólo una barra de progreso utilizando el componente Swing **JProgressBar**. Seleccionamos los valores mínimos y máximos de la barra de progreso en 0 y 40, respectivamente (propiedades **minimium** y **maximium**). También se pueden seleccionar estos valores con los métodos **setMinimum** y **setMaximum**, o en el constructor al crear la barra de progreso. La propiedad **StringPainted** hace que la barra de progreso muestre un String (cadena) de porcentaje dentro de sus límites. Por defecto, la cadena indica el porcentaje completo de la barra de progreso. El String de porcentaje es el valor devuelto por el método **getPercentComplete** formateado a porcentaje. Otra alternativa es mostrar un String diferente con **setString**.

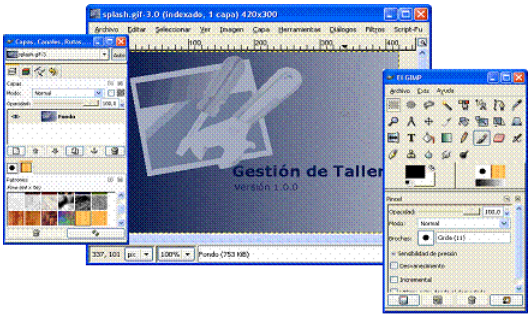
Para ir cambiando el valor de la barra de progreso utilizamos un temporizador (un ejemplar de la clase **Timer**). Este objeto se arranca con el método **Start()** y dispara un evento **action** cada segundo. El método oyente del evento **action** es **ActionPerformed** y lo que hace es llevar un contador para ir modificando el valor de la barra de progreso.

El código completo lo mostramos a continuación. Observa que el **JPanel** que hemos creado tiene un método **main()**, con lo cual ya no necesitamos la clase **Principal** para lanzar nuestra aplicación, sino que se lanzaría desde ésta nueva clase llamada **Inicio**.

```
public class Inicio extends JPanel implements ActionListener {
    private Timer timer = new Timer(100, this);
    private JProgressBar barraprogreso = new JProgressBar();
    private int cont=0;
    public Inicio() {
        try {
            jbInit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        barraprogreso.setBounds(new Rectangle(80, 235, 230, 15));
        barraprogreso.setStringPainted(true);
        barraprogreso.setMaximum(40);
        this.add(barraprogreso, null);
        this.actionPerformed(null);
        this.timer.start();
        this.setBackground(new Color(115, 147, 245));
        this.setLayout(null);
    }
    public void actionPerformed(ActionEvent e) {
        this.cont=this.cont+1;
```

```
            this. barraprogreso.setValue(cont);
        }
        public static void main(String[] a){
            JWindow f = new JWindow();      //creo una ventana sin barra de título
            f.add(new Inicio());         //añado el JPanel a la ventana
            f.pack();
            f.setSize(400,300);
            f.setVisible(true);         //aquí mejor usar el método pantalla.centrar
            JFrame f2 = new Login();            // creamos la ventana de login
            try { Thread.sleep (5000); } catch (Exception e) {}
            pantalla.centrar(f2);
            f.setVisible(false);            //cerramos la ventana splash
         }
    }
```

Pues bien, ya hemos conseguido crear nuestra ventana splash. Si la ejecutamos verás que no queda centrada, y además, nos interesaría que apareciera también una imagen característica de la aplicación. Centrar la ventana se puede hacer con nuestra clase **pantalla.java**, sobrecargando el método **centrar()** para que pueda servir también para objetos **JWindow**. Para poner una imagen de fondo en un **JPanel** debemos sobrecargar el método **public void paintComponent(Graphics g)** para que dibuje la imagen que queramos. La imagen de fondo la puedes crear con un programa de manipulación de imágenes sencillo como es **GIMP**.



En nuestro ejemplo lo que hemos hecho es crear un fondo degradado y añadir una imagen característica y el título y versión de la aplicación. A continuación tienes todos los pasos que hemos seguido para crear la ventana splash:

Pantalla de bienvenida

SABÍAS QUE...
GIMP es un programa de editor de imágenes del proyecto GNU. Se publica bajo la licencia GNU General Public License. Se puede obtener en la siguiente dirección:
Programa de diseño gráfico GIMP

*Una vez construido el esqueleto de la aplicación, **Víctor** se lo ha mostrado a **Carmen,** que ha quedado muy satisfecha con el resultado. Ha llegado el momento de conectar con la base de datos. "¿Y eso como se hace?" Se pregunta **Víctor**. "No es tan complicado como parece", responde **Carmen**, "JDeveloper incorpora gran cantidad de asistentes que generan la conexión con la base de datos y facilitan el acceso a la modificación del diseño o de los datos. Sólo hay un aspecto en el que deberemos incorporar código manual y es para la validación de usuario y contraseña pero tampoco es demasiado complicado". **Carmen** y **Víctor** se ponen manos a la obra para preparar el acceso a la base de datos y la conexión con la aplicación.*

En la unidad anterior hemos aprendido a utilizar la base de datos Oracle 10g Express Edition. En este apartado debemos asegurarnos de que la base de datos está instalada en el usuario **TALLER**. Si no fuera así, utilizaremos los scripts para instalarla en dicho usuario.

Este usuario es el que utilizaremos para conectarnos a la base de datos desde JDeveloper. No vamos a instalar la base de datos en el usuario SYSTEM porque este usuario es el administrador de la base de datos y su utilidad es para otros fines.

Por ello crearemos un usuario **TALLER** y contraseña **TALLER** que disponga de todos los privilegios y será el que utilizaremos en nuestra aplicación para acceder a los objetos de la base de datos. En el siguiente archivo tienes una demostración de los pasos a seguir para crear el usuario en Oracle 10g Express Edition:

Creación del usuario Taller

Una vez creado el usuario, procedemos a la creación de la base de datos. Para ello vamos a necesitar los siguientes scripts, que son archivos que contienen las instrucciones SQL necesarias para crear las tablas de la base de datos, secuencias para los campos

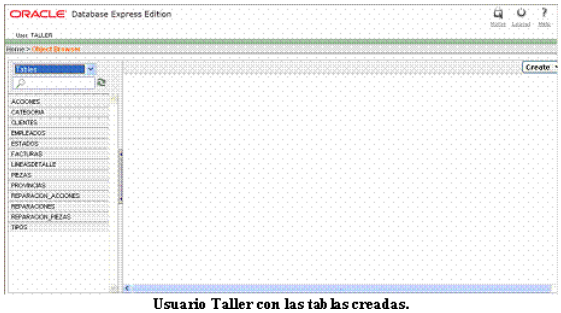
autonuméricos, disparadores para actualización de datos y usuarios:

Scripts para la creación de la BD

Lo siguientes que debemos hacer es cargar los scripts en Oracle para, posteriormente, proceder a su ejecución. A continuación tienes una demostración de los pasos a seguir:

Creación de la BD

Tras realizar los pasos anteriores, debemos tener el usuario Taller con las tablas y demás objetos de la base de datos. También hemos creado unos usuarios adicionales llamados USER1, USER2 y USER3, con sus respectivas contraseñas iguales al nombre de usuario, los cuales podremos utilizar posteriormente para entrar a la aplicación con diferentes permisos de ejecución.



Usuario Taller con las tablas creadas.

Conexión con la base de datos

Una vez que tenemos la base de datos instalada en Oracle, vamos a **crear una conexión** con ella desde JDeveloper. Esta conexión la utilizaremos para acceder a la base de datos desde la aplicación, así como para introducir los datos de las tablas.

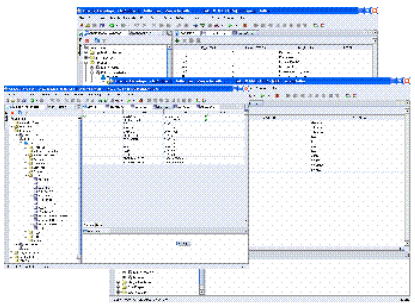
Los pasos a seguir son muy sencillos, y ya los conocemos de la unidad anterior. Para ello, debemos situarnos en nuestro proyecto abierto en JDeveloper en dirigirnos a la pestaña **Connection Navigator**, situada en la parte izquierda de la ventana, o bien en el menú **View/Connection Navigator**. Desde este panel haremos clic con el botón derecho en la carpeta **Database** que aparece colgada de **Connections**. A partir de aquí JDeveloper incorpora un asistente para conexión con bases de datos, que nos pide los siguientes datos:

Hostname:	localhost
Port:	1521
SID:	XE

Tras crear la conexión con la base de datos, en el siguiente apartado procederemos a introducir los datos en la misma. Ahora sigue uno a uno los pasos que se indican en la siguiente demostración:

Conexión con la BD

Introducción de datos en las tablas



Para introducir los datos en la base de datos podemos hacerlo a través de JDeveloper, o bien podemos cargar los scripts directamente en la base de datos desde Oracle 10g Express Edition.

En JDeveloper introducir los datos es sencillo, tan sólo tenéis que dirigiros a la pestaña **Data** de cada tabla e introducir los registros correspondientes. Recordad hacerlo en el orden correcto para respectar las reglas de integridad referencial. Que una base de datos cumpla la integridad referencial implica que en todo momento dichos datos son correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas. Recordad también que no hace falta introducir los campos autonuméricos definidos por secuencias (las claves principales de cada una de las tablas).

Por ejemplo, vamos a introducir los datos de la tabla CATEGORIAS a través de JDeveloper. El resto lo haremos cargando el script **Datosv2.sql** en la base de datos para ahorrar tiempo. Ya sabemos cómo cargar un script en Oracle y ejecutarlo (ver la presentación sobre creación de la BD). El siguiente recurso te muestra cómo hacerlo con el entorno integrado.

Introducción de datos en la BD en JDeveloper

Autoevaluación

¿Qué definición darías para la integridad referencial?

a) Es la propiedad que tiene una base de datos relacional de garantizar que los registros de una entidad se relacionan con otras entidades válidas, es decir, que existen en la base de datos.

b) Es la propiedad que tiene cualquier base de datos de asegura que todas las entidades tienen claves primarias y claves ajenas.
c) Es la propiedad que garantiza que la base de datos está normalizada, o sea, se encuentra al menos en 3ª forma normal.

Comprobar

Acceso a la base de datos mediante código

En el apartado VENTANA DE LOGIN de esta unidad, creábamos una ventana de Login, que para facilitar las cosas sólo comprobaba el usuario y contraseña introducido con uno dado. Ahora vamos a introducir el código necesario para realizar la conexión con Oracle y dejar pasar sólo a los usuarios que se encuentren registrados en nuestra base de datos.

Es necesario crear una nueva tabla en nuestra base de datos que se llame USUARIOS y que contenga los campos usuario, password e intentos, cuyos nombres son suficientemente descriptivos. El campo intentos lo vamos a utilizar para registrar el número de intentos fallidos que un usuario dado tiene. Este campo lo pondremos a cero cuando el usuario entre de forma correcta a la aplicación.

La creación de la tabla **USUARIOS** la podemos hacer desde el entorno de Oracle o bien desde el Navegador de Conexiones de JDeveloper. En el siguiente recurso tienes una demostración de cómo se crearía la tabla utilizando la segunda alternativa:

Modificación de la base de datos en JDeveloper

El código que se detalla a continuación es el que hemos introducido en la ventana de Login para establecer la conexión con la base de datos y dar acceso a la aplicación si el usuario y contraseña facilitados son correctos. Como se puede apreciar el método **hacerLogin** se ha modificado completamente. Este método devuelve **true** si la sentencia **`ResultSet r=s.executeQuery();`** devuelve algún resultado, lo cual querrá decir que el **usuario** y **password** introducidos son correctos, porque coinciden con alguna fila de la tabla **USUARIOS**.

Sobre el código tienes algunos comentarios que facilitan su comprensión. Al final de la unidad viene el código fuente completo de la aplicación y puedes consultar cualquier duda sobre su funcionamiento ejecutándola directamente desde el entorno de JDeveloper. Recuerda que debes añadir lo que aparece en negrita, el resto del código ya lo habíamos insertado anteriormente.

```
public class Login extends JFrame {
    private JTextField txUsuario = new JTextField();
    private JLabel lbUsuario = new JLabel();
    ...
PreparedStatement s;      // s es una sentencia SQL precompilada
    Connection c;                  // c es la conexión con la base de datos
    Statement st;                  // st contiene la sentencia SQL ya creada
    ...
    private void jbInit() throws Exception {
       this.getContentPane().setLayout( null );
        this.setSize(new Dimension(365, 215));
    ...
        try {
// establecemos el driver de la conexión
                Class.forName("oracle.jdbc.driver.OracleDriver");
// servidor, usuario y password para realizar la conexión
                c =  DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:XE","TALLER",
"TALLER");
// sentencia de donde obtener usuario y password
                s =  c.prepareStatement("select * from taller.usuarios
where nombre=? and password=?");
                st = c.createStatement();
        } catch (ClassNotFoundException e) {
                    System.out.println("Clase no encontrada");
                    } catch (Exception ex) {  ex.printStackTrace(); }
    ...
    public boolean hacerLogin(String u, String p) {
```

```
        try {
           s.setString(1,u);
           s.setString(2,p);
// si el método executeQuery devuelve algún registro (fila
// entonces el usuario u y password p son correctos
           ResultSet r=s.executeQuery();
           boolean res = r.next();
           if (res) {
             st.executeUpdate("update taller.usuarios set Intentos=0
where nombre='"+u+"'");
           } else {
              st.executeUpdate("update taller.usuarios set
Intentos=Intentos+1 where nombre='"+u+"'");
           }
           return res;
        } catch (SQLException e) {System.out.println("Error en el SQL:
"+e.toString()); return false;
        } catch (Exception ex) {
System.out.println("Error: "); ex.printStackTrace();
return false;}
    }
    public void cerrarConexion()  {
      try   {
          c.close();
      } catch (SQLException e) {  e.printStackTrace();
      } catch (Exception ex)   {  ex.printStackTrace();  }
    }
```

Para que este código funcione correctamente, es preciso comprobar que la librería **Oracle JDBC** está añadida al proyecto tal y como se explica en la unidad anterior.

*"Ahora llega lo bueno", comenta **Víctor**. "Debemos ponernos a programar interminables líneas de código para implementar los formularios de cada una de las tablas que permitan las altas, bajas, modificaciones de clientes, empleados ... ufff! que agobio me está entrando ...". De nuevo **Carmen** debe intervenir para tranquilizarle, y le explica que con los ADF Business Components tienen acceso a una serie de componentes que les van a facilitar mucho el trabajo, convirtiéndose la mayor parte del tiempo en la sencilla tarea de arrastrar y colocar. "Te voy a explicar cómo crear un formulario maestro-detalle y a partir de ahí podrás diseñar el resto de formularios de la aplicación sin problema". **Víctor** se siente aliviado pero todavía no llega a imaginar qué tipo de componentes serán estos y decide prestar mucha atención a las explicaciones de **Carmen**.*

Oracle ADF Business Component (**Oracle ADF BC**) llamado anteriormente **BC4J** (Componentes de Negocio para Java) es un framework que nos provee de un conjunto de librerías con funcionalidades que nos permiten crear componentes de negocio específicos de forma sencilla en la plataforma J2EE. Hace de puente entre la base de datos y la aplicación conteniendo la lógica de negocio y encargándose de la función de validar los datos. Aunque no es necesario ADF BC para desarrollar servicios de negocio en JDeveloper (podemos usar otros servicios o incluso una implementación propia), Oracle ADF BC hace más fácil crear un entorno óptimo, escalable, portable y servicios de negocio reutilizables, con un mínimo de código.

Los ADF Business Components se encargan de las validaciones y de la lógica de negocio. Están basados en el modelo MVC y nos permiten trabajar con aplicaciones Web o de escritorio ya que se centran en la lógica de negocio, la cual nos da la ventaja de la facilidad del mantenimiento de nuestra aplicación. Por tanto, los ADF BC separan nuestra aplicación en tres niveles:

- **MODELO**. Esta capa se encarga de la interacción del data-source y la ejecución de la lógica de negocio.
- **VISTA**. Esta capa se encarga de la interacción de la aplicación y la interfaz de usuario.
- **CONTROLADOR**. Esta capa se constituye en una interfaz entre la capa del modelo y la capa de vista.

En el siguiente apartado vamos a ver cómo construir el Modelo de Negocio que se almacenará en la carpeta **Model** de nuestro proyecto.

PARA SABER MÁS

Puedes acceder a la ayuda on-line de JDeveloper donde tendrás a tu disposición una completa información sobre los componentes ADF BC (en inglés):

Información general sobre ADF BC

Desarrollo con ADF BC

Autoevaluación

¿Qué es ADF BC?

a) Oracle ADF Business Components son unas interfaces del modelo View que sirven para acceder a los datos de la base de datos
b) ADF Business Components es una completa tecnología para el desarrollo de servicios de negocio en una aplicación MVC
c) ADF Business Components es una tecnología de desarrollo de base de datos de Java que simplifica la interacción entre componentes Swing y el modelo de negocio. Está formada por clases Java y una API para el enlace de los componentes Swing con el modelo de negocio definido en XML

Comprobar

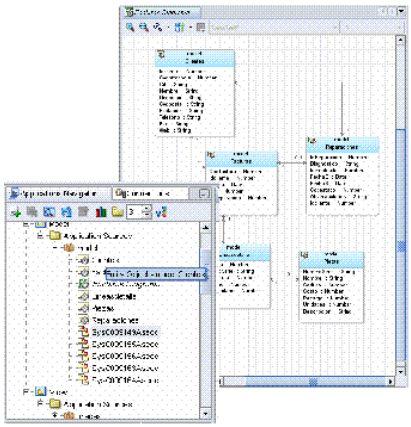
Implementación de la lógica de negocio

Acabas de ver que los ADF Business Components de Java se encargan de las validaciones y de la lógica de negocio, pero ¿qué hacen exactamente? Pues bien, debes saber que...

- incorporan una lógica de negocio reutilizable,
- usan una arquitectura basada en componentes,
- separan la lógica de negocio de la lógica de presentación y
- generan la mayoría de código que necesita una aplicación.

La generación de código automático es una gran ventaja, ya que elimina la necesidad de escribir código de acceso a la base de datos manualmente, y con ello las posibilidades de error.

Para comenzar vamos a crear un **Business Component Diagram**. Este tipo de diagramas nos permiten crear y modificar visualmente entidades de nuestra base de datos y otros objetos. Para activar esta opción debemos hacer clic con el botón derecho sobre la carpeta **Model**, y elegir **New (Adf Business Components (Business Components Diagram**. Llamamos al diagrama "**Facturas Diagrama**". A continuación, nos vamos al Navegador de Conexiones y nos situamos en nuestra conexión **DBConexionTaller**. Expandimos la carpeta **Tables** y arrastramos hacia el diagrama las tablas **Facturas**, **LineasDetalle** y **Piezas**. Posteriormente añadimos las tablas **Reparaciones** y **Clientes**.



Una vez creado nuestro Diagrama, se habrán creado las entidades en la carpeta Model de nuestro proyecto y dentro de ella en Applicagtion Sources, subcarpeta model, y desde ahí podremos modificar de forma gráfica e intuitiva cualquier propiedad de las entidades y atributos de nuestra base de datos. Concretamente podemos identificar varios tipos de objetos:

- Entidades
- Diagramas
- Asociaciones

En el siguiente recurso tienes los pasos completos que hemos realizado para crear el modelo de negocio de un maestro / detalle con las tablas Facturas, LineasDetalle, Clientes, Reparaciones y Piezas:

Usando ADF Business Components

Para facilitar la comprensión de estos conceptos, vamos a repasar lo que hemos aprendido en la animación anterior, **que no debes dejar de ver con atención**:

1. Creación de un **Business Components Diagram**, o Diagrama de Componentes de Negocio
2. Creación de Objetos de tipo Entidad para las tablas Facturas, LineasDetalle y Piezas.
3. Identificación de cada objeto en la carpeta ApplicationSource a través de su icono
4. Modificación de aspectos del Diagrama, como:
    1. el nombre de las relaciones (asociaciones) y de atributos de las entidades (IVA, PrecioUnitario ...)
    2. Mover entidades dentro del diagrama y cambiar el zoom
5. Acceso al Editor de Objetos mediante:
    1. doble clic sobre las entidades del diagrama
    2. doble clic sobre el objeto en el Navegador de Aplicaciones
    3. la opción Edit "Objeto" que aparece al hacer clic derecho sobre el objeto en el Navegador de Aplicaciones
6. Utilizar el Editor de Objetos para:
    1. Introducir texto de ayuda (Tooltip text) de los campos
    2. Determinar la etiqueta que aparecerá cuando insertemos ese campo en un formulario
    3. Incorporar máscaras para campos de tipo fecha
    4. Definir cuándo un atributo es actualizable mediante la clausula While New, por ejemplo, si quisiéramos guardar en una fecha oculta cuándo se ha creado la factura
    5. Crear reglas de validación adicionales a las que incorpore la base de datos, por ejemplo que el precio unitario sea mayor o igual a cero
7. Creación de Objetos de tipo Vista para las tablas anteriores.
8. Creación de múltiples Objetos Entidad y Vista para Reparaciones y Clientes, utilizando el comando **New (Adf Business Components (Business Components from Tables**.
9. Adición de nuevos objetos a un Diagrama de Componentes de Negocio (Objetos Entidad Reparaciones y Clientes).

10. Creación de Application Module, módulo que encapsula los Objetos Vista de nuestra aplicación.
11. Testeo del modelo de negocio con la opción **Test de ADF BC**, mediante el Explorador de Oracle Business Component, donde:
    1. Podemos ver todas las opciones anteriores que hemos introducido en la lógica de negocio de nuestra aplicación, desde los textos de ayuda de los campos hasta las reglas de validación como la del precio unitario
    2. Podemos probar todas la VO (View Object) y la VL (ViewLink) de nuestro Data Model, que son las que están incluidas en nuestro Application Module
    3. Podemos navegar entre registros, insertar, modificar y borrar registros, buscar registros por sus camos y guardar los datos en la base, usando el commit o el rollback para deshacer los cambio.

Es importante que observemos que el **Business Component Diagram**, aunque es una herramienta muy útil para tenerlo todo organizado y obtener una visualización general de todos nuestros componentes de negocio, no es estrictamente necesario en nuestro proyecto. Podemos crear ADF BC (entidades, vistas, ...) sin necesidad de un diagrama, haciendo clic con el botón derecho sobre la carpeta **model** y eligiendo la opción **New (Adf Business Components** y dentro de ella el componente que queramos crear (Entity Object, View Object, etc), lo cual crearía estos objetos dentro de la carpeta **Model** y para modificarlos haríamos doble clic sobre ellos igual que hemos hecho sobre el diagrama.

Autoevaluación

¿Cuál de las siguientes afirmaciones es la correcta?

a) Los objetos View son los componentes reutilizables del modelo de negocio desarrollando con ADF BC, al contrario que los objetos Entidad, que deben ser definidos para cada aplicación que creemos.
b) Un módulo de aplicación es el contenedor de todos los archivos de nuestra aplicación.
c) Dentro del modelo de negocio podemos encontrarnos entre otros objetos Entity Objects, que se corresponden con las tablas de la base de datos, View Objects, que son diferentes vistas de los datos y objetos domain, que son un tipo de datos especial usado por los atributos de los ADF BC.

Comprobar

Los Application Module ...

a) Encapsulan los Objetos Vista de nuestra aplicación.
b) Se muestran en la Data Control Palette
c) Ambas preguntas son correctas

Comprobar

PARA SABER MÁS

Acceso a la ayuda de JDeveloper 10.1.3 que trata sobre la implementación de la lógica de negocio en Oracle ADF Business Components (en inglés):

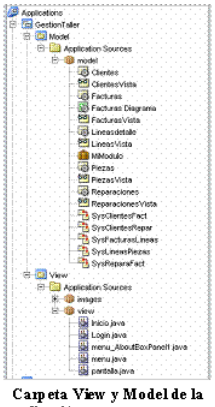
Implementando la Lógica de Negocio

Creación de formularios con ADF Swing y ADF BC

Una vez que hemos definido la lógica de negocio, ¿qué nos queda por hacer? Pues seguro que estás pensando, con razón, que lo que nos queda es ponernos a trabajar sobre los formularios de la aplicación, que estarán basados en dicho modelo de negocio. Esta parte seguro que te va a gustar...

Para ilustrar el modo en que debemos realizar estos formularios, en esta sección vamos a crear un **formulario maestro-detalle de Facturas y LineasDetalle** respectivamente. El objetivo es:

- visualizar los datos completos de las facturas, con indicación del cliente al que se le realiza la factura,
- si esa factura proviene de alguna reparación, la fecha en que se ha realizado,
- así como cada una de las líneas que componen la factura (nº serie de la pieza, cantidad, precio unitario, ...).



Carpeta View y Model de la aplicación.

Vamos a resumir lo que tenemos hasta ahora en la aplicación.

En primer lugar debemos tener las dos carpetas que separan nuestra aplicación: **View** y **Model**.

- En la primera (**View**) hasta ahora tan sólo tenemos los formularios que hemos creado al inicio de la unidad: **Inicio**, **Login**, **Pantalla** y la **interfaz de la aplicación**, compuesta por las ventanas **Menú** y **Acerca de**.
- Y en la segunda carpeta (**Model**), se encuentran almacenados todos los objetos que hemos creado en el apartado anterior mediante la tecnología ADF Business Components, esto es, la **lógica de negocio de nuestra aplicación**.

Para crear el formulario utilizaremos el comando **New (Adf Swing (Empty Form**, disponible al hacer clic derecho sobre la carpeta **View**. El asistente que aparece creará un JFrame con las indicaciones que le facilitemos. El formulario generado contendrá un código específico de Oracle ADF para inicializar el vínculo con la base de datos. Después usaremos la **Data Control Palette** (si actualmente no está activa la puedes activar mediante **View (Data Control Palette**) para diseñar el formulario. Este proceso es tan sencillo como arrastrar los componentes y soltarlos sobre el formulario, además podemos modificar algunos aspectos del modelo de negocio, y si lo consideramos necesario, incorporar nuevos campos a las vistas o modificar el formato de los existentes. El proceso completo de creación del formulario se encuentra en las siguientes presentaciones:

Creación de formularios usando ADF BC y ADF Swing (Parte I)

En este recurso hemos aprendido importantes conceptos que no debes dejar de estudiar. Concretamente, hemos aprendido lo siguiente:

1. Diseño de un formulario maestro-detalle utilizando ADF BC disponibles en Data Control Palette y controles ADF Swing.
2. Utilización de la vista **Data Control Palette** para seleccionar los elementos ADF BC de nuestra aplicación.
3. Utilización de FormLayout para situar los componentes del formulario Facturas.java

4. Utilización del comando **AddChild** para adición al formulario de controles vinculados con los datos, con la simple acción de arrastrar y colocar desde un ApplicationModule: **textfield**, **labelfor, table, navigationbar, combobox**.
5. Identificación de Vínculos a Objetos Vista en un ApplicationModule.
6. Creación de componentes vinculados con **CreateBinding**, utilizando el componentes Swing JTextField y el componente ADF Swing JULabel.
7. Utilización del componente Swing Container **JScrollPane** para contener el detalle de la factura (**LineasVista**). Este tipo de control proporciona barras de desplazamiento cuando el contenido no cabe en la ventana para moverse por todo el panel.
8. Uso de la vista **Constraints** para modificar los aspectos del JScrollPane como alineamiento y posición
9. Utilización de **COMMIT** para reflejar los cambios en la base de datos.
10. Modificación de aspectos del formulario a través de los controles **bindings del archivo .xml** que define dicho formulario, a los cuales se accede mediante la ventana Structure. Uso del **Table Binding Editor** (opción Edit ADFm Binding Propierties haciendo clic derecho sobre el formulario) para modificar los objetos del módulo.
11. Modificación del modelo de negocio utilizando el **View Object Editor para modificar Objetos Vista.** Estos cambios van a estar sólo en la vista y no afectan a la base de datos. Por otra parte, los cambios introducidos en los bindings mediante **Table Binding Editor** sólo estarán en el formulario, y no afectarán ni a las vistas ni a la base de datos:
    1. Creación de un nuevo campo Pieza en el objeto LineasVista:
        - Utilización del modo experto para modificar la sentencia SQL de la vista. Creación del nuevo campo Pieza dentro de la vista.
        - Modificación de aspectos del formulario en el **Table Binding Editor** para incluir ese nuevo campo Pieza en el formulario (por defecto aparece oculto en la vista)
    2. Creación de un combobox en el campo Pieza que tome valores dinámicos, para que muestre el nombre de la pieza en lugar del número de serie. Determinación en el **List Binding Editor** de:
        - la fuente de datos que recibirá el valor del control cuando sea seleccionado (**Base Data Source**),
        - fuente de datos que proporciona la lista de valores que los usuarios ven cuando se despliega el combobox (**List Data Source**),
        - atributo común en el área de mapping,
        - atributo de entre todos los que hay en el **List Data Source** que va a mostrar el combobox (nombre)
    3. Creación de un campo calculado:
        - Utilización del modo experto para modficar la sentencia SQL de la vista. Creación del nuevo campo calculado Importe.
        - Modificación de aspectos del formulario en el **Table Binding Editor** para incluir ese nuevo campo Importe en el formulario (por defecto aparece oculto en la vista)
    4. Creación de campos combobox en Cliente y en Reparaciones, utilizando la opción **Edit ADFm Bindign Propierties** e introduciendo los valores en el **List Binding Editor.**

Creación de formularios usando ADF BC y ADF Swing (Parte II)

En este recurso es continuación del anterior y básicamente lo que hacemos en él es establecer un vínculo entre el maestro y detalle de nuestro formulario con un nuevo objeto **ViewLink**. Este tipo de objetos sirven para coordinar las relaciones de jerarquía entre las vistas. Posteriormente creamos un nuevo Application Module y volvemos a realizar los cambios en los Objetos Vista que hemos hecho en los pasos anteriores. Concretamente, los pasos que hemos seguido han sido los siguientes:

- Creación de un ViewLink usando el comando **New ViewLink** disponible al hacer clic sobre la carpeta model
- En el cuadro de diálogo Create View Link, selección de las vistas que queremos enlazar
- Creación de un nuevo Application Module y adición en él de las vistas FacturasVista y LineasVista, observando cómo en este módulo ambas vistas aparecen sí vinculadas
- Modificación del formulario, borrando los campos y creándolos de nuevo a partir del nuevo Application Module en el que sí existe el vínculo maestro-detalle
- Realización del paso 11 modificación del modelo de negocio, para introducir los cambios en los objetos **binding** del formulario .

Autoevaluación

¿Qué tipo de componentes podemos encontrarnos en la Data Control Palette?

a) Elementos ADF Business Components de nuestra aplicación.
b) Elementos ADF Swing de nuestra aplicación
c) Ambas respuestas son correctas.

Comprobar

Según las carpetas y archivos que se han creado en el tutorial, ¿a qué tipo de archivos crees que se refiere <pageName>PageDef.xml?

a) Son archivos cuya estructura podemos ver a través de la ventana Structure y están compuestos por parámetros, ejecutables y los bindings de la aplicación.
b) Son archivos que se crean dentro de la carpeta model y que contienen los enlaces al modelo de negocio de ADF BC.
c) Son archivos que se crean dentro de la carpeta view.pageDefs que utilizan xml como lenguaje de definición de datos y sólo están disponibles cuando diseñamos páginas Web.

Comprobar

¿Cómo podemos acceder a la estructura del archivo .xml que define nuestro formulario?

a) Seleccionando el archivo <pageName>PageDef.xml, eligiendo View/Structure y dentro del panel que aparecerá en la parte inferior de la ventana, escogiendo la carpeta bindings.
b) Seleccionando el comando View/Data Control Palette y eligiendo el Application Module donde esté nuestra aplicación
c) Ambas respuestas son correctas

Comprobar

Puedes acceder al proyecto completo con los fuentes de Gestión de Taller en el siguiente enlace:

Demo Gestión de Taller

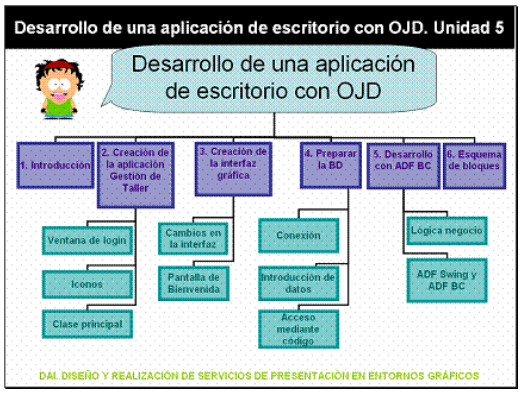
PARA SABER MÁS

Si necesitas ayuda para desarrollar aplicaciones con ADF Swing, esta página contiene información general sobre esta tecnología y cómo crear los distintos componentes de los que consta (en inglés):

Desarrollo con ADF Swing

Esta página te muestra cómo crear un formulario maestro-detalle de manera similar a como lo hemos hecho nosotros,

controles del tipo combobox y formularios de edición de datos (en inglés):
Maestro-detalle con ADF BC y ADF Swing
En la siguiente página tienes una demostración de la creación de un modelo de negocio y un formulario maestro-detalle (en inglés):
Maestro-detalle con ADF BC y ADF Swing DEMO



Desarrollo de una aplicación de escritorio con JDeveloper