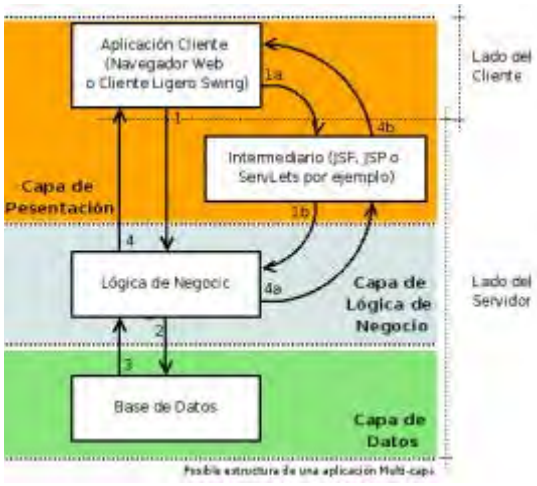
Unidad Didáctica X.- Proyecto globalizador

Caso.

*Hasta ahora **Víctor** no tenía que enfrentarse a un diseño de una aplicación en solitario. Sólo había colaborado en pequeñas tareas, pero ahora, después del tiempo que lleva en **SI Andalucía**, le han encomendado desarrollar una aplicación completa para una clínica veterinaria.*

*Para **Víctor** ese proyecto es muy importante y desea hacerlo lo mejor posible, pues tiene que demostrar su valía para que en el futuro le asignen más proyectos y para poder ir ascendiendo puestos en la empresa. Por ese motivo, antes de comenzar el proyecto, está interesado en investigar las diferentes alternativas que tiene para llevarlo a cabo. **Víctor** es buen conocedor de que aparte de las técnicas que ya ha utilizado existen otras alternativas que puede usar, y no hay mejor persona que **Carmen** para que le asesore en ese aspecto, por lo que no duda en preguntarle.*



***Carmen** le responde que tiene varias alternativas para desarrollar su aplicación. Una de las alternativas por la que podría optar es la de separar la lógica de negocio del resto de componentes creando una estructura por capas, es decir, mantener separados los datos de la empresa y las operaciones que se pueden hacer sobre esos datos. La capa de negocio implementará la lógica de negocio y será un componente fundamental, pues dará soporte a todas las operaciones de la empresa y acceso a todos los datos, que generalmente se almacenarán en una base de datos.*

*A **Víctor** ese concepto le resulta un poco raro, por lo que empieza a poner caras raras. **Carmen**, que se da cuenta, le explica que la capa de negocio no es la aplicación que usaría el cliente, sino la parte de la aplicación que contiene el funcionamiento interno. El cliente realmente usaría otra parte de la aplicación que estaría desarrollada en Swing o JSP, pero para ambos casos la aplicación utilizaría la capa de negocio para operar y trabajar con los datos.*

*Entonces a **Víctor** se le ilumina la cara de felicidad y dice: "¡¡es como la arquitectura modelo-vista-controlador!!" **Carmen** reflexiona un momento y hace un gesto de duda con la cabeza, luego le explica que pueden*

parecer lo mismo, pero en realidad, ella habla de algo diferente.

En esta unidad tendrás que enfrentarte con el diseño de tu propio proyecto de principio a fin. La idea es que profundices en los contenidos vistos a lo largo de las unidades anteriores en un único proyecto. Para ello te pediremos que realices una aplicación de nivel medio, es decir, algo cercano a una aplicación real pero dejando algunas funciones sin implementar. Se trata de un proyecto que no te debe llevar demasiado tiempo, pero que debe cubrir una serie de requisitos mínimos.

Cuando se desarrolla una aplicación es necesario realizar un análisis destallado y un posterior diseño siguiendo alguna metodología de desarrollo, y si es posible, algún lenguaje de modelado como UML. No vamos a exigirte un análisis y diseño detallado, pero tienes que tenerlos en mente si realmente quieres desarrollar, en el futuro, aplicaciones de calidad.

Para este propósito, se plantean en esta unidad diferentes partes. Por un lado, intentamos continuar acercándote a las aplicaciones actuales para que adquieras una mejor visión de lo que hoy día se desarrolla. Por otro, introduciremos un caso real de una aplicación, a modo de ejemplo, y después se propondrán una serie de casos prácticos a partir de los cuales puedes desarrollar tu propia aplicación.

Tu trabajo, por tanto, es desarrollar una aplicación para uno de esos casos prácticos expuestos más adelante, o bien para un caso práctico elegido por ti que cumpla con unos requisitos mínimos, también expuestos más adelante.

Para que tu trabajo tenga la calificación más alta es importante que intentes hacer el trabajo de forma ordenada, cuidando la presentación e intentando desarrollar soluciones elegantes y prácticas. Así que no te agobies por el trabajo que tienes por delante, directamente pon tus manos a la obra sin pensarlo dos veces.

Proyecto globalizador: Desarrollo de una aplicación.

Caso.

*Después de la conversación anterior, a **Víctor** le pica la curiosidad y quiere ponerse a investigar cuanto antes. Se pregunta, ¿qué será eso de la capa de negocio? ¿qué es eso de las aplicaciones multi-capa? Después de un rato leyendo documentos en Internet, está hecho un*

*auténtico lío. Por lo que decide preguntarle a **María**.*

__María__ le comenta que a ella también le costó trabajo adquirir esos conceptos, y le explica que eso de "aplicación Multi-Capa" es un tipo de arquitectura de aplicación, también llamado, patrón arquitectural.

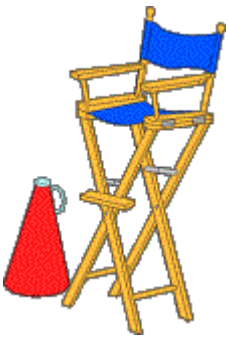
Entonces __Víctor__ pregunta "pero MVC también es una arquitectura de aplicación, ¿no?" __María__ asiente con la cabeza y le comenta que las arquitecturas de aplicación ofrecen un posible esquema o estructura organizativa para nuestra aplicación.

"¿Pero cómo sé qué arquitectura de aplicación debo usar?", le pregunta ansioso __Víctor__. __María__ le responde que eso depende de lo que quiera hacer y de sus necesidades. Le comenta que hay muchos patrones arquitecturales pero que esos dos son los más usados. Cada uno tiene sus ventajas e inconvenientes.

Después de eso, __Víctor__ se queda un poco desolado por todo lo que le queda por aprender. __María__ que se da cuenta, le comenta que para su proyecto lo mejor es utilizar la arquitectura MVC, usando un framework como JSF o ADF, los cuales conoce a la perfección. Le dice además que lo que realmente debe preocuparle es implementar una solución adecuada, elegante y eficiente.

En este apartado, no se va a hacer un estudio detallado de las aplicaciones que existen hoy día, pero dado que es el último módulo creemos que es bueno que pienses en cómo son y qué persiguen. Aunque intuimos que más o menos ya lo sabes.

En las aplicaciones actuales, aparte de los parámetros típicos de diseño software, como pueden ser cohesión, acoplamiento y modularidad, se persiguen otros parámetros que de alguna forma garantizan el éxito. Algunos de estos factores de calidad, quizás los más importantes, son:



1. **Funcionalidad y corrección**: El software debe proporcionar solución a las necesidades del usuario, aunque dar solución a todas las necesidades es imposible.
2. **Fiabilidad:** El software debe estar funcionando eficientemente durante el máximo tiempo posible. El software operativo 24x7 (24 horas 7 días a la semana) es un ejemplo de alta fiabilidad.
3. **Usabilidad y accesibilidad**: El software debe intentar ser fácilmente entendido, aprendido y utilizado por el usuario.
4. **Eficiencia**: Capacidad del software para producir un buen rendimiento. Un software que realice la misma tarea que otro en la mitad de tiempo será el doble de eficiente.
5. **Robustez y amigabilidad**: Cómo se comporta el sistema ante algo inesperado, por ejemplo, introducir una letra cuando se esperaba un número. Si la aplicación explica que se ha equivocado y vuelve a dar opción de insertar el dato, diremos que es amigable, pero si la aplicación simplemente finaliza, diremos que es robusto. Sin embargo, si el sistema simplemente falla o se queda colgado, diremos que nuestro software es poco robusto y poco fiable.

Es obvio que estas características marcan el éxito del software, pero como habrás supuesto a lo largo del módulo, mejorar esas características tiene un coste, el cual, se reduce considerablemente con las herramientas tipo RAD (Rapid Application Development). Por lo que otros factores de éxito del software son:

1. **Coste de producción del software**: Es más probable que un software tenga éxito si su coste es menor, lo cual se consigue, reduciendo el tiempo de desarrollo.
2. **Mantenibilidad**: generalmente entendido como la capacidad del software de ser fácilmente modificable a lo largo del tiempo. Aunque a esta definición también hay que añadir todas aquellas tareas que hay que realizar día a día para que el software siga funcionando. Todas esas tareas suelen llevar un coste adicional (alquiler de servidores, copias de seguridad, actualizaciones, etc.).

Ahora, debes intentar tener en cuenta todos estos factores a la hora de desarrollar tu proyecto, o por lo menos, intentar tenerlos en mente.

> ***Para saber más***
>
> ***En este enlace podrás encontrar más información sobre calidad del software:***
>
> ***Calidad en Ingeniería del Software***

Autoevaluacion

1 ¿Cuál de las siguientes características de un navegador web dirías que es más amigable?

a) Permitir guardar una página como bookmark o marcador.
b) Autocompletar un formulario por nosotros.
c) Cuando un dominio no existe, mostrar una pantalla que diga "servidor no encontrado" y dar la oportunidad de poner otra dirección
d) Permitir retroceder en el historial de navegación.

Comprobar

La expansión de internet y B2X

La expansión de Internet ha llegado y ha traído una

nueva oleada de aplicaciones. En unidades anteriores se han revisado conceptos como HTTP, HTML, AJAX, JSP, etc. que están involucrados en el desarrollo de aplicaciones web. Pero, ¿cuáles son las principales características de las aplicaciones web de hoy día?

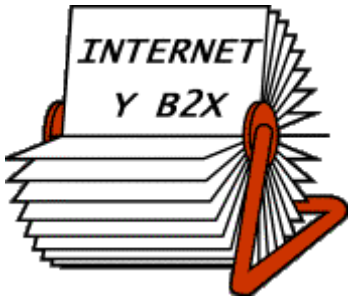
Es difícil responder a esta pregunta, pero vamos a intentar aproximarnos. Para ello, vamos a intentar pensar en las aplicaciones web desde el punto de vista del comercio electrónico o e-commerce. En ese mundo podemos considerar varios perfiles cuando desarrollamos una aplicación web:

1. **Clientes**. Serán los destinatarios principales de las aplicaciones Web. Las aplicaciones webs ofrecen servicios a nivel de cliente (no confundir con servicio web), y los clientes usarán un navegador web para acceder a la aplicación. Estas aplicaciones se caracterizan por tener una interfaz con una estética bastante más elaborada que las aplicaciones de escritorio, generalmente todas las aplicaciones de escritorio tienden a usar una estética común para facilitar su aprendizaje, pero en la web pasa todo lo contrario. Es importante pues que la aplicación web capte al usuario no sólo por la funcionalidad, sino por el buen gusto y la usabilidad. Este perfil será el B2C (***Business To Consumer***).
2. **Empresas asociadas**. Cuando dos empresas alcanzan un acuerdo de colaboración, sea del tipo que sea, generalmente implica un intercambio de información o la utilización de servicios especiales. Este intercambio de información y servicios no se suele dar en la misma forma que en el lado del cliente, aunque también se aprovecha Internet para tal objetivo. En este caso es muy común hablar del concepto de "**Servicios Web**" (en ingles, *Web Services*), es decir, intercambio de información y servicios a través de la web. Este caso es la del perfil B2B (***Business to Business***).
    - ***Ejemplo de servicio web***: a modo ilustrativo supongamos que tenemos un negocio en el que ofrecemos descuentos a alumnos de la universidad. La forma de llevar a cabo el sistema de descuentos es complicada sino se tiene acceso en tiempo real a la base de datos de alumnos de la universidad, pero tampoco podemos tener acceso a todos los datos de la universidad porque es obviamente peligroso. Una de las formas de solucionar esto es que la universidad ofrezca un servicio web en el que, una vez dados los datos del alumno (Nombre, apellidos y DNI) se verifique que realmente es alumno de la universidad. Para esto, cuando la aplicación cliente disponga de los datos realizará una consulta al servicio WEB enviando un documento XML mediante un protocolo como SOAP, después el servicio WEB devolverá otro documento XML con la respuesta "SI" o "NO" (también usando el protocolo SOAP para responder). Si la respuesta del servicio WEB es "SI", permitirá al alumno aprovecharse de los descuentos.

Esquema de funcionamiento de los Web Services

3. **Empleados**. Cuando en una empresa existe una sección que ofrece servicios exclusivos para sus empleados (formación on-line, por ejemplo), entonces hablamos de un perfil B2E (***Business To Employee***). Generalmente, se caracteriza por tener una estética menos elaborada que las aplicaciones destinadas a un perfil B2C.

B2C, B2E y B2B son casos claros de servicios que se pueden ofrecer a través de aplicaciones WEB. Algunas tecnologías implicadas en esto, aparte de las ya conocidas son:



1. **XML** (eXtended Mark-up Language). Documento para el intercambio electrónico de datos. XML está asociado inevitablemente a los servicios web, en los que intervienen tecnologías como XML-RPC y SOAP (como protocolos de comunicación basados en XML y HTTP) , WSLD (como mecanismo de descripción de servicios web) y UDDI (para acceso a repositorios de servicios web). Asociados a XML también tenemos tecnologías como XSL, para la transformación de documentos XML, DTD y XML-Schema, como sistemas para la definición de documentos XML, y XPath, para la consulta de información de documentos XML. Es ampliamente usado en el intercambio de información entre organizaciones (B2B).

2. **EDI** (Electronic Data Interchange). Estándar destinado a cubrir las transacciones y la estructura de la información que se intercambian electrónicamente empresas y organizaciones (como por ejemplo, órdenes de compra y facturas). Es un estándar destinado al perfil B2B, que puede ser usado en una gran variedad de medios (Módem, E-Mail, HTTP, FTP, etc...).
3. **HTTPS, SSL, TLS, PGP, SET**, etc. Cuando la información trasmitida es sensible, es decir, su pérdida o difusión puede causar graves consecuencias, sean del tipo que sean, es conveniente utilizar mecanismos en los que la información vaya cifrada y no pueda ser falseada por terceros. Es el caso de las tecnologías HTTPS (para páginas web), PGP (para correo electrónico) y SET (para transacciones electrónicas seguras de tarjeta de crédito), aunque hay algunas más.


Logotipo de la famosa Wikipedia

Como ves, hay bastante terminología en la que ahondar, la idea es que tengas en mente su existencia. Lo último que nos queda por hacer en este apartado es responder a la siguiente pregunta:

¿Hacia dónde van las aplicaciones web de hoy día?

Quizás la respuesta la tenemos al revisar un concepto relativamente reciente: **Web 2.0**.

La Web 2.0 es aquella en la que el usuario no es un simple observador de contenidos, en la Web 2.0 el usuario forma parte activa de la web, puede participar e incluso puede "construir" la web. Algunos ejemplos de Web 2.0 son la famosa Wikipedia o el portal del.icio.us, que permite compartir nuestros bookmarks. En definitiva, la Web 2.0 no es una tecnología, sino una actitud.

> ***Para saber más***
>
> ***Si quieres entender mejor qué es la WEB 2.0, no dejes de visitar este enlace:***

¿Qué es la web 2.0?

Seguro que también te interesa asomarte a este enlace para conocer algo más sobre XML:

XML en Castellano

Autoevaluación

1 ¿Cual de las siguientes afirmaciones es cierta?

a) La web 2.0 es aquella que se basa en la utilización de servicios web y XML para su implantación.
b) Los servicios web se utilizan ampliamente para las operaciones asociadas a perfiles B2E.
c) EDI y SOAP son tecnologías que se utilizan en los servicios web.
d) Los servicios web se utilizan ampliamente para las operaciones asociadas al perfil B2C.

Comprobar

Proyecto globalizador: Desarrollo de una aplicación.

Las aplicaciones de intranet

Una Intranet es algo más que una simple red de área local en la que existen recursos compartidos y donde los usuarios intercambian información. Es una red en la que se usan servicios típicos de Internet (e-mail, HTTP, etc.) para facilitar la comunicación entre individuos de una organización.

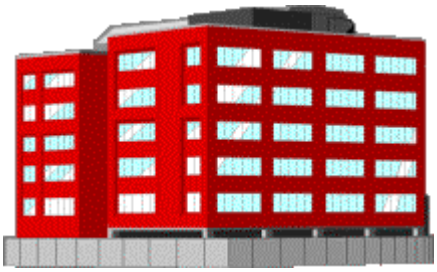
Pero podemos ir más lejos. Si a través de protocolos típicos de Internet hiciéramos aplicaciones específicas a la lógica de negocio que agilizaran el trabajo a los empleados, la organización sería mucho más productiva, ¿no crees?

Imaginemos, una aplicación web que consista simplemente en un formulario para meter datos de facturación, que almacene los datos en una base de datos para luego poder buscarlos e imprimirlos, ¿será útil?

Obviamente, ésta es una de las cuestiones más habituales en las empresas, y es lo que se denomina "aplicación de Intranet" o "backoffice" (trastienda). Para suplir este tipo de necesidades, muchas empresas recurren a aplicaciones web por varios motivos:

1. Casi todo el mundo sabe usar un navegador web y no hay que instalar software adicional. Además, la actualización es más rápida que en el software tradicional (no hay que actualizar el software ordenador por ordenador).

2. Se puede usar la misma infraestructura que se utiliza para dar servicio a clientes (B2C). Es decir, paralelamente a la aplicación web que da servicio a los clientes se realiza una web para ayudar a la labor del empleado. Obviamente, en este caso hay que tener especial cuidado porque el backoffice contendrá datos especialmente sensibles.
3. Una aplicación web se puede hacer tan privada como uno quiera, es decir, puede estar sólo disponible desde dentro de las oficinas de la empresa, o incluso disponible a través de Internet para que el empleado pueda teletrabajar. Son muchas las empresas que ofrecen a sus empleados acceso remoto a su Intranet y/o a sus aplicaciones de Intranet, aquí tienes un ejemplo: https://cas.csic.es/login.



4. Es relativamente sencillo cubrir varios perfiles de empleados con una misma aplicación web, creando un sencillo sistema de permisos: director, diseñador, operador, etc.

Revisando el concepto de "Aplicación de Intranet" se puede llegar a la conclusión de que la Intranet se ha "salido" de la red de área local (LAN). Hoy en día es fácil dar acceso remoto a Intranets a través de tecnologías como VPN, por poner un ejemplo, por lo que no es necesario estar físicamente en la "oficina" para acceder a las aplicaciones de Intranet.

> ***Para saber más***
>
> ***Si quieres saber qué significa el término backoffice, consulta este enlace:***
>
> ***¿Qué podemos entender por backoffice?***

Autoevaluación

1 ¿Cuál de las siguientes afirmaciones NO es aplicable a una aplicación de Intranet?

a) Dentro de una empresa suelen tener ámbito interno, los clientes no suelen tener acceso.
b) Sólo pueden ser accesibles desde dentro de la red de área local de la empresa.
c) Es común utilizar aplicaciones web para la creación de aplicaciones de Intranet.
d) Están destinadas a facilitar el trabajo del empleado, lo cual aumenta su productividad.

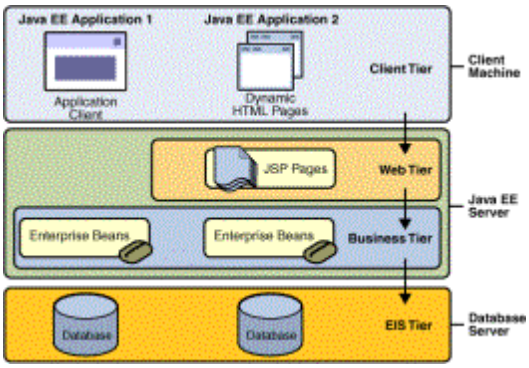
Comprobar

Las aplicaciones distribuidas y J2EE

Las aplicaciones distribuidas son un concepto difícil de tratar y muy amplio (se podrían incluir en este apartado los servicios web, por ejemplo), por lo que aquí se darán unas simples pinceladas. Se puede decir, para empezar, que hablar de aplicaciones distribuidas es hablar de aplicaciones en las que se distinguen diferentes componentes, situados generalmente en diferentes entornos y que se comunican (generalmente a través de una red) para dar solución a un problema.

El concepto de aplicación distribuida puede variar según el contexto, por lo que aquí vamos a intentar dar una visión de lo que sería una aplicación distribuida en Java. Según la definición dada en la documentación de Java 2 Enterprise Edition (J2EE), podemos distinguir 3 tipos de aplicaciones distribuidas:

1. **Two-Tier**. Aplicaciones que separan los componentes en dos niveles, son las aplicaciones típicas cliente/servidor.
2. **Three-Tier**. Aplicaciones que separan los componentes en tres niveles, Cliente-MiddleWare-Servidor. Donde MiddleWare es una capa o nivel cuya misión es facilitar la comunicación entre componentes software y/o aplicaciones, eliminando la complejidad de tener que programar la conexión entre cliente y servidor, lo cual suele ser muy costoso.
3. **Multi-Tier**. Aplicaciones que en las que hay uno o varios componentes cliente, múltiples middlewares y múltiples servidores.



El termino "***Tier***" hace referencia a áreas funcionales aisladas pero que se comunican entre sí, es similar al concepto de capa o nivel. En una estructura Multi-Tier es común hablar de:

- **Business Tier**, como la capa del servidor que contiene la lógica de negocio,
- **Web Tier** y **Client Tier**, como capas dedicadas a la presentación a usuario,
- **EIS Tier**, como la capa que da acceso a las fuentes de datos (bases de datos, por ejemplo).

En un sistema distribuido, por tanto, los componentes de diferentes capas deben comunicarse entre sí. Existen bastantes protocolos y estándares para **comunicar remotamente** los componentes de un sistema distribuido, algunos de estos, quizás los más importantes, son:

1. **RMI o Invocación Remota de Métodos**. Es una tecnología para aplicaciones de dos capas (Two-Tier) que permite comunicar dos componentes (cliente y servidor) de forma sencilla a través de la red. El uso de esta tecnología suele ahorrar bastantes quebraderos de cabeza a los programadores, es sencilla y

da buenos resultados.

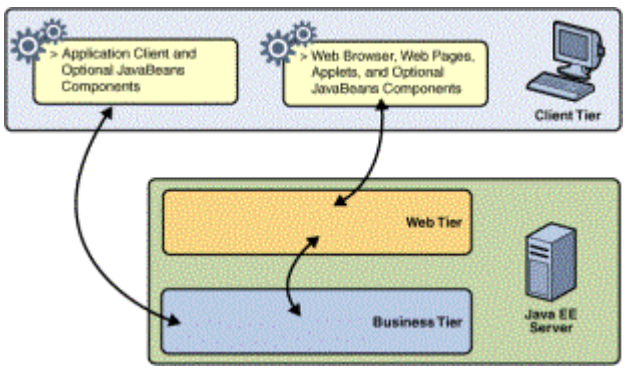
2. **CORBA**. Es una compleja tecnología para aplicaciones distribuidas de tres capas. Esto implica que tiene un componente middleware que se encarga de permitir el uso remoto de objetos a través de la red (podemos tener una colección de objetos accesible remotamente). La gran ventaja de CORBA es que podemos usar muchos lenguajes de programación, como por ejemplo C++, Ada y obviamente JAVA.
3. **RMI-IIOP**. Es un puente entre RMI y CORBA. Esto es, provee interoperatibilidad entre ambos, permitiendo a un componente diseñado para RMI ser fácilmente adaptado para trabajar con CORBA.
4. **SOAP** y **XML-RPC** son también protocolos usados en este ámbito, pues permiten la intercomunicación de componentes basándose en tecnologías tales como HTTP y XML. Como ya se comentó en apartados anteriores, están íntimamente relacionados con los Web Services.

Otro aspecto interesante de las aplicaciones distribuidas en Java, es el uso de los Entreprise Java Beans o simplemente Enterprise Beans (EJB). Los EJBs están relacionados con las aplicaciones distribuidas porque son componentes que permiten encapsular fácilmente la lógica de negocio, y que además pueden ser accedidos tanto localmente como remotamente. En modelo de tres capas (three-tier), el EJB estaría en el lado del servidor. Si aprovechamos las posibilidades de comunicación de los EJBs (que hacen uso de RMI-IIOP), podemos implementar el cliente de forma separada y comunicarlo remotamente con la lógica de negocio a través de un servidor de aplicaciones.

Esto implica que por un lado se ejecutaría de forma independiente el EJB, con la ayuda de un servidor de aplicaciones, y por otro lado la aplicación cliente.

- El **EJB** implementaría por tanto la lógica de negocio y el control de acceso a datos (sólo o trabajando en conjunto con otros EJBs),
- el **cliente** implementaría la presentación a usuario, que podría ser una aplicación web o una aplicación de escritorio, y
- el **servidor** de aplicaciones facilitaría la comunicación entre ambos. Si el cliente, además, es una aplicación web, el servidor de aplicaciones también se encargará de realizar las funciones de servidor web.

Como has podido deducir, un servidor de aplicaciones es, por tanto, algo más que un simple servidor web. El servidor de aplicaciones incluido en el paquete J2EE 5 de Sun Microsystems distingue varios tipos de componentes. Entre ellos podemos encontrar:



1. **EJBs**, que son los

denominados componentes de
negocio (el servidor de aplicaciones Java se encargará de su ejecución).
2. **Java Servlets**, **JavaServer Faces** y **JavaServer Pages** como componentes Web (también el servidor de aplicaciones se encargará de su ejecución).
3. Y aplicaciones de escritorio, applets y navegadores web que se ejecutarán en el lado del cliente.

Además, un servidor de aplicaciones incluirá obviamente los protocolos y mecanismos necesarios para comunicar dichos componentes entre si, y para mantener una colección de componentes accesibles remota o localmente. En unos casos el protocolo será el famoso HTTP, en otros, podremos usar protocolos más complejos, como los expuestos antes.

Los servidores de aplicaciones y las aplicaciones distribuidas son un mundo complejo, que incluyen tecnologías de lo más variopintas, aquí se ha dado un esbozo, pero habría que incluir otras tecnologías importantes como los Web Services. Algunos servidores de aplicaciones importantes son:

- J2EE,
- BEA WebLogic,
- IBM WebSphere,
- JBoss,
- OAS (Oracle Aplication Server),
- Macromedia JRun,
- etc.

Para saber más

Aquí encontrarás la documentación del servidor de aplicaciones J2EE:

The Java(tm) EE 5 Tutorial

Para aprender un poco más sobre RMI, te recomendamos visitar el siguiente enlace:

Tutorial de Java RMI

Autoevaluación

1 Cuando hablamos de una aplicación basada en el patrón arquitectural Multi-Tier, ¿en qué capa estaría JSF, y en consecuencia ADF Faces al ser una implementación de JSF?

a) Client Tier.
b) Web Tier.
c) Business Tier.
d) EIS Tier.

Comprobar

Las aplicaciones para dispositivos móviles

Seguro que no es la primera vez que has oído hablar de aplicaciones para dispositivos móviles. Probablemente estás cansado de ver miles de anuncios en revistas y televisión vendiéndote juegos para tu móvil, y seguro que al verlos te has preguntado si es fácil desarrollar aplicaciones para ese tipo de dispositivos.

La respuesta: es más fácil de lo que parece, pues has aprendido java y muchas de esas aplicaciones están hechas en java.

Sin embargo, es evidente que no es lo mismo un PC que un dispositivo móvil. Cualquier teléfono móvil o PDA tiene una memoria y velocidad de procesamiento bastante inferior a la de un ordenador actual. Esto implica que no podemos pretender ejecutar un programa en java diseñado para PC directamente en un dispositivo móvil.

Como sabrás, java utiliza una máquina virtual (JVM) que independiza el software del hardware donde se ejecuta. Para los dispositivos móviles es necesario proveer de **una JVM especial**, una que necesite menos hardware para ejecutar programas java y que provea al programador de **una API específica** para aplicaciones móviles, es decir, de **un conjunto de clases y paquetes que den acceso a funcionalidad específica de dispositivos móviles.** Todo esto, sin contar con que los dispositivos móviles no se comunican usando las mismas tecnologías que un PC, en un móvil es habitual contar con conexiones GPRS para actualizar datos desde un servidor remoto.

Para este tipo de aplicaciones tenemos **J2ME** (***Java 2 Micro Edition***). J2ME se divide en tres capas:

Estructura en capas de J2ME

- **Configuración** (*Configuration*),
- **Perfil** (*Profile*) y
- **Paquetes Opcionales** (*Optional Packages*).

La configuración provee de la JVM y algunas clases base, y el perfil (construido sobre la capa de configuración) ofrece la parte más importante del API que utilizaremos para programar nuestras aplicaciones. Opcionalmente, también se ofrecerán algunos paquetes extra para hacer aplicaciones con características especiales, situados en la capa llamada "Paquetes Opcionales".

Las configuraciones y perfiles más conocidos son:

- configuración CLDC y
- perfil MIDP,

Ambos son pertenecientes al J2ME de Sun Microsystems. Ambos se complementan muy bien, permitiendo ejecutar una JVM en sistemas cuya memoria no supera los 128 KB. MIDP proporcionará el API necesario para programar aplicaciones para dispositivos móviles, dentro de él podemos encontrar paquetes

como:

- "**javax.microedition.lcdui**" que ofrece capacidades para la creación de interfaces de usuario (imágenes, listas, cajas de texto o textboxs, etc...),
- "**javax.microedition.lcdui.game**" que provee un API para el desarrollo de juegos,
- "**javax.microedition.rms**" que provee de un API para el almacenamiento persistente de datos y su posterior lectura.

Una pregunta común llegado este punto es si es necesario instalar CLDC y MIDP en el dispositivo móvil antes de ejecutar una aplicación java. La respuesta es **no,** por las características de esta tecnología el fabricante debe integrar estas capas (y los paquetes opcionales) en el dispositivo móvil. Lo único que tenemos que tener en cuenta es que versión de CLDC y MIDP tiene el dispositivo móvil, pues el API cambiará de una versión a otra. Tanto para CLDC como para MIDP la última versión es la 2.0.

Otra pregunta común es ¿que tengo que hacer para desarrollar una aplicación para este tipo de dispositivos? La respuesta es difícil de resumir, pero se puede esquematizar en los siguientes pasos:

1. Lo primero es descargar e instalar un kit de desarrollo de J2ME (por ejemplo el toolkit de Sun Microsystems, J2ME Wireless Toolkit 2.2), o bien, si existe, el recomendado del fabricante del dispositivo móvil.
2. Lo siguiente sería diseñar la aplicación, para lo cual podemos hacer uso de aplicaciones RAD específicas (JDeveloper dispone de un módulo para desarrollo de aplicaciones móviles). Las aplicaciones basadas en J2ME son similares al funcionamiento de un applet de java, pero en este caso, extendiendo la funcionalidad de una clase base llamada **MIDlet**. Un ejemplo sencillo de **MIDlet** podría ser el siguiente:

```
import javax.microedition.lcdui.Display;
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Alert;
public class HolaMundo extends MIDlet {
Alert alerta_de_hola;
public HolaMundo() {
alerta_de_hola = new Alert("Hola Mundo!");
}
// Código que se ejecuta al comenzar la aplicación
public void startApp() {
Display.getDisplay(this).setCurrent(alerta_de_hola);
}
// Código que se ejecuta cuando se interrumpe o pausa la
aplicación
public void pauseApp() {
}
// Código que se ejecuta al finalizar la aplicación
public void destroyApp(boolean unconditional) {
```

```
    }
    }
```

3. El siguiente paso es compilar la aplicación, usando para esto el compilador de java (javac) con una configuración de clases nativas diferente: "**javac -bootclasspath ..\lib\cldcapi20.jar;..\lib\midpapi20.jar HolaMundo.java**".
4. Una vez compilado, preverificamos que el código es válido para las especificaciones del JVM del dispositivo, usando la herramienta "**preverify**".
5. Empaquetamos la aplicación usando la herramienta "**jar**" y creamos el descriptor de la aplicación java (JAD), que será un archivo de texto con extensión ".jad" que tiene un formato específico. En este paso se obtienen dos archivos, uno con extensión .jar y otro con .jad, pero que tienen el mismo nombre: **HolaMundo.jar** y **HolaMundo.jad**. Ambos forman la aplicación.

6. Comprobamos que la aplicación funciona probándola en un emulador de nuestro dispositivo móvil, los fabricantes suelen ofrecer este software de forma gratuita.
7. Desplegamos la aplicación en nuestro dispositivo móvil, ya sea por cable USB, por Bluetooth u otro método disponible.

Para saber más

¿Te gustan los juegos? ¿Te imaginas programando tu propio juego? Aquí encontrarás un Tutorial de programación de juegos para móviles:

Programación de juegos para móviles con J2ME

Autoevaluación

1 Si estamos desarrollando una aplicación para un dispositivo móvil, ¿qué capa de J2ME es la que proporciona la parte principal del API que utilizaremos para programar nuestra aplicación?

a) Profile MIDP.
b) Configuration MIDP.
c) Profile CLDC.
d) Configuration CLDC.

Comprobar

Al igual que ocurre en otros módulos, cuando llega el final te proponemos que realices un proyecto que te resulte interesante y en el que pongas en práctica todos los conocimientos que has adquirido a lo largo del curso. El

objetivo de esta unidad es consolidar los conocimientos adquiridos, y somos conscientes de que esos conocimientos perdurarán más tiempo en la memoria si resultan entretenidos.

Como programador, seguro ya te has dado cuenta de que cuando te pones a programar sobre algo que te gusta, la programación se hace más amena e incluso fascinante. Algunos programadores comparan la programación con la poesía, porque cuando tienes algo que te inspira, las líneas de código fluyen de tus dedos y te embarga la "pasión" por lo que estás creando.

Nos pasa a todos, por ese motivo, si tienes muchas ganas de realizar un proyecto determinado, te invitamos a que lo realices. Es interesante que lo lleves a la práctica por varios motivos: hará que esta práctica sea mucho más fácil, te servirá como tarea para esta unidad y podrás utilizarla para tus propios fines.

Pero claro, **si decides realizar tu propio proyecto**, es necesario que tengas en cuenta las siguientes cosas:

1. **Tu proyecto debe cumplir con los requisitos mínimos**. De esta forma todos realizaréis un proyecto similar y con una dificultad más o menos parecida.
2. Debes **enviarle a tu tutor o tutora un documento en el que narres el proyecto que vas a realizar**. El tutor te confirmará si puedes realizar el proyecto o te sugerirá cambios para hacer que tu proyecto cumpla con los requisitos mínimos. **Intenta leer alguno de los proyectos propuestos para hacerte una idea de las dimensiones de los proyectos y qué tipo de cosas se piden en los mismos**.
3. No escojas un proyecto muy largo salvo que realmente pienses que te va a dar tiempo hacerlo. Lee el ejemplo guía expuesto más adelante y realiza un proyecto más o menos del mismo tamaño. Siempre podrás completarlo y añadirle funcionalidad más adelante, cuando ya hayas superado este módulo.

No te preocupes si no tienes ningún proyecto en mente que puedas realizar. Para ese caso te proponemos una serie de proyectos entre los que puedes escoger. Lee los proyectos (de la sección de ejemplos guía) y elige el que más te guste, eso si, cuando hayas elegido el proyecto que mejor te parezca, envía un correo al tutor o tutora indicándole cuál es el proyecto que has escogido.

Es posible que al leer alguno de los proyectos propuestos te surjan posibles mejoras sobre los mismos, lo cual es perfecto, pero eso sí, antes de realizarlas debes ponerte en contacto con el tutor o tutora para que las autorice. Recuerda incluir los detalles de las mejoras realizadas (sobre el proyecto propuesto original) en la documentación que tienes que entregar al final del proyecto.

Cuando te pongas a realizar algún proyecto, siempre es conveniente que sigas las siguientes recomendaciones:

1. **Lee el problema con cuidado**, y antes de hacer nada, **dedica dos o tres horas al diseño del modelo de datos**. Primero intenta identificar las entidades que tiene y sus posibles atributos. Luego identifica las relaciones. Una vez que tienes hecho esto, crea la base de datos. Esto te ayudará a tener una buena comprensión del problema.
2. **No dejes la documentación de sistema para el final**. Escribe los comentarios javadoc conforme vayas desarrollando la aplicación.
3. **Deja para el final la "decoración" de la aplicación**, tu tutor o tutora valorará más que funcione a que sea un proyecto "bonito" pero inútil.

Consideraciones generales para todos los proyectos.

Antes de comenzar tu proyecto, ya sea uno elegido entre los ejemplos propuestos o uno de tu propia creación, debes tener en cuenta además los siguientes factores:

1. El proyecto se puede realizar entre 1 ó 2 personas. En caso de realizar el trabajo en grupo deben seguirse las consideraciones para trabajo en grupo incluidas más adelante.
2. El proyecto se evaluará sólo si se entregan todos los documentos expuestos en la sección de documentación a entregar, y en el caso del trabajo en grupos, los documentos indicados en consideraciones para trabajo en grupo.
3. Para verificar la autoría de los proyectos se podrá solicitar a los autores que realicen alguna modificación sobre alguna de las partes que han desarrollado. Si el autor o autores no son capaces de responder a dicha modificación se entenderá que el alumno o alumna no es el autor de dicha actividad, y la actividad se considerará suspensa.
4. De igual forma, para verificar la autoría de los proyectos el tutor o tutora podrá ponerse en contacto con con cada uno de los autores o autoras de la actividad para realizar preguntas concretas sobre el proyecto desarrollado. Estas preguntas se formularán haciendo uso del chat y/o del teléfono, y si se estima oportuno, del correo electrónico. En el caso de los trabajos en grupo, todos los miembros del grupo deben conocer en profundidad el proyecto desarrollado, independientemente de que se trate de una parte que no hayan desarrollado ellos.
5. Aunque el alumno se reserve los derechos de explotación del proyecto desarrollado es condición indispensable que, junto a la totalidad de los documentos que debe entregar, el alumno **entregue todo el código fuente del proyecto** que ha desarrollado.

Requisitos mínimos del proyecto.

En este apartado se expone lo que debe hacer tu proyecto y lo que hará que tu aplicación sea bien calificada. A lo largo de todo el módulo has estudiado dos perspectivas de la programación en entornos gráficos: una a nivel de escritorio y otra a nivel de aplicación web. El proyecto que nosotros te pedimos tendrá, por ese motivo, tres partes. Siguiendo el patrón arquitectónico Modelo-Vista-Controlador tu proyecto deberá tener dos vistas y un modelo:

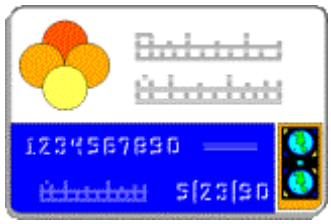
1. **Vista web**. Debes desarrollar una vista web de parte de la funcionalidad de la aplicación. Aquí deberás implementar la funcionalidad del lado del cliente (B2C). Para desarrollar esta parte puedes utilizar directamente JSP o ADF Faces, como prefieras, aunque desarrollarás el proyecto más rápido al utilizar ADF Faces.
2. **Vista de escritori**o. En este caso, debes desarrollar una vista de escritorio de la otra parte de la funcionalidad de la aplicación. En este caso, la funcionalidad a implementar es la del empleado, es decir, esta vista debe cubrir tareas que son propias de un empleado, o lo que es lo mismo, debes implementar el backoffice. Para desarrollar esta parte puedes usar tanto Swing como ADF Swing, aunque desarrollarás el proyecto más rápido al usar ADF Swing.
3. **Modelo**. Contendrá la lógica de negocio, lo cual se traducirá en todas las operaciones contra base de datos de tu proyecto. Para facilitar la actividad, ambas vistas utilizarán el mismo modelo para trabajar. El acceso a la base de datos se producirá por tanto sola y exclusivamente en el modelo, y las vistas sólo accederán a la base de datos a través del modelo. Si te es más fácil, puedes utilizar ADF Business Components para el desarrollo del modelo.

Bien, una vez visto esto, establezcamos unas **bases para un buen proyecto**:

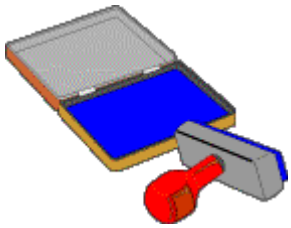
1. **Base de datos**
    1. La base de datos tendrá al menos 4 tablas. Entre todas las tablas tiene que haber, al menos, 2 claves externas, a fin de que puedan existir al menos dos relaciones del tipo maestro-detalle.
    2. Una de las tablas estará destinada a recoger los datos de cliente, en dicha tabla existirán los campos: login y password.
2. **Modelo**
    1. Como se ha comentado antes, debe encapsular toda la funcionalidad de acceso a la base de datos y la lógica de negocio.
    2. En caso de no usar ADF Business Components, el modelo implementará todas las funciones de inserción, eliminación, modificación y selección de registros de la base de datos necesarias para el correcto funcionamiento del proyecto.
    3. Todas las operaciones contra base de datos serán consistentes y atenderán a la concurrencia (posibilidad de que varios usuarios estén

usando el sistema simultáneamente). Es decir, ninguna operación podrá dejar la base de datos en un estado incorrecto y/o inconsistente.

3. **Vista Web**
    1. Deberá tener dos secciones:
        1. Sección privada: sólo para clientes. A esta sección los usuarios deben acceder utilizando su login y su password. Dentro de ella un cliente sólo podrá hacer uso de sus propios datos, es decir, no podrá ver ni usar datos vinculados con otros clientes.
        2. Sección pública: para todo el mundo. Donde sólo se podrán ver datos de carácter público.
    2. En la vista web es necesario que al menos se implemente la funcionalidad necesaria para:
        1. Hacer funcionar al menos un formulario de búsqueda, como resultado de la búsqueda debe mostrar el correspondiente listado de resultados.
        2. Mostrar al menos un Maestro-Detalle. Se valorará positivamente que el Maestro-Detalle sea accedido desde el resultado de una búsqueda como la descrita en el punto anterior.



        3. Hacer funcionar al menos un formulario para inserción de datos (independiente del maestro-detalle).
        4. Hacer funcionar al menos un formulario para la modificación de datos previos, enlazado por ejemplo desde el resultado de una búsqueda (independiente del maestro-detalle).
        5. Hacer funcionar al menos un formulario para la eliminación de datos previos (independiente del maestro-detalle)
    3. Se incluirán objetos característicos de la web como:
        1. Imágenes. Preferentemente incluir alguna imagen a modo de logotipo o anagrama.
        2. Hojas de estilo. Incluir al menos una hoja de estilo adicional en todos los scripts realizados, con al menos dos clases y dos redefiniciones de estilos para algún tag HTML específico.
        3. Se valorará positivamente la inclusión de algún objeto swf, animaciones y sonidos.
4. **Vista Escritorio**
    1. En este caso, la aplicación contendrá la funcionalidad del lado del empleado, para ello es necesario que se implementen una serie de funciones, las cuales, deben ser obviamente diferentes a las funciones de la vista web.



    2. En la vista de escritorio es necesario que al menos se implemente la funcionalidad necesaria para:
        1. Permitir al empleado hacer una búsqueda de datos, y obtener el consecuente listado de resultados.
        2. Permitir al empleado hacer una inserción de datos en una tabla o varias tablas.
        3. Permitir al empleado hacer una modificación de datos, partiendo por ejemplo de una búsqueda previa.
        4. Permitir al empleado el uso de un formulario tipo Maestro-Detalle (independiente de las operaciones de inserción y modificación de los puntos anteriores).

5. Permitir al empleado la eliminación de registros de una tabla.
6. Permitir al empleado generar dos informes, para ello se pueden usar los paquetes iReport y JasperReports.

3. Además, la vista de escritorio dispondrá de una documentación de usuario (en este caso, de empleado). Se valorará muy positivamente que dicha ayuda sea accesible por contexto.

Y para todo el conjunto de la aplicación:

1. Se generará una documentación de sistema con JavaDoc. Todas las clases y métodos deben contener comentarios JavaDoc que describan adecuadamente su comportamiento.
2. Se valorará muy positivamente que la aplicación creada tenga una adecuada presentación y disposición de los elementos.
3. Se valorará muy positivamente que el uso de la aplicación sea intuitivo, es decir, que tenga un alto grado de usabilidad.
4. La aplicación debe desarrollarse en JDeveloper y Java 2, evitando el uso de clases y métodos obsoletos (deprecated).
5. El código fuente debe ser fácilmente comprensible por cualquier programador, es decir, debe estar bien estructurado y documentado.
6. Y muy importante, **como resultado de una operación ninguna parte de la aplicación debe terminar de forma abrupta ni producir excepciones.** Es complicado que una aplicación esté libre de errores, pero debes intentarlo.

A nuestro entender, el proyecto mínimo que aquí se expone no es muy exigente, aunque eso depende obviamente de la opinión de cada cual. La idea de esta unidad no es hacer un macro proyecto, sino un proyecto pequeño pero bien hecho, que ponga en práctica todos los conocimientos adquiridos.

¡¡ Pon todo tu buen hacer en ello !!

Documentación a entregar.

El proyecto deberás entregarlo comprimido en un archivo tipo ".zip". El nombre del archivo zip debe seguir la siguiente estructura: "**NombreApellido1Apellido2_PEG_PF.zip**". En dicho zip debes incluir los siguientes archivos y directorios:

1. ***miproyecto.pdf.*** Memorandum del desarrollo de tu proyecto. En él debes incluir la siguiente información:
   1. **Portada.** En la portada debes incluir el nombre del centro de referencia ("I.E.S. Aguadulce"), el nombre

del ciclo formativo ("Desarrollo de Aplicaciones Informáticas"), el año del curso (curso 2007/2008 por ejemplo), el nombre del proyecto, y lo más importante, los nombres, apellidos y DNIs de los autores y/o autoras.

2. **Resumen del proyecto elaborado**. Aquí debes relatar de forma somera los objetivos del proyecto que has elaborado, incluyendo cuáles son las funciones implementadas en la vista web y cuáles las implementadas en la vista de escritorio.***Si has decidido hacer un proyecto propio, en lugar de seguir uno de los propuestos, deberás incluir en este apartado una descripción detallada de tu proyecto.***Por otro lado, si has decidido escoger uno de los proyectos propuestos y has modificado o añadido alguna característica o requisito (previa autorización del tutor), también deberás hacerlo constar aquí.
3. **Defensa del proyecto**. Debes prestar especial atención a este apartado. En él, tienes que intentar "vendernos" la aplicación, es decir, contarnos qué es lo que la hace interesante como si fuéramos un posible comprador. Debes intentar reflejar los siguientes aspectos:

   1. **Puntos fuertes**. Intenta reflejar cuáles son los principales aspectos que hacen tu aplicación especial y útil.
   2. **Dificultades.** Cuéntanos aquellas cosas que te han resultado más difíciles de llevar a cabo y por qué. Señala aquellos aspectos del funcionamiento interno de tu proyecto que te han resultado más complejos de llevar a cabo.
   3. **Posibles Mejoras**. Intenta reflejar aquellas mejoras que incluirías en tu aplicación y que, por falta de tiempo, se han quedado pendientes.
4. **Reparto del trabajo**. En caso de que el trabajo haya sido elaborado en grupo debéis reflejar en esta sección cómo habéis repartido la carga de trabajo. No obstante, debéis saber que, de cara a posibles preguntas del tutor o tutora, todos los miembros del grupo tendréis que conocer en profundidad todo el proyecto.
5. **Conjunto de casos de prueba**. Se trata de un conjunto de usos de la aplicación que reflejen la funcionalidad de la misma, por ejemplo:el alta de un nuevo cliente en el sistema. Incluye para ello los datos a utilizar en el caso de prueba, y recuerda que esos casos de uso pueden ser simplemente los mismos que a ti te han permitido verificar que tu proyecto funciona correctamente.
6. **Conclusiones**.

2. ***bd.sql***. Script de creación de la base de datos en Oracle, incluyendo su estructura y un conjunto de datos de prueba que permitan al tutor usar toda la funcionalidad sin ningún problema desde el primer momento (por ejemplo, para poder realizar una baja o eliminación de un registro es necesario que previamente existan datos).
3. ***/manual***: Carpeta que contendrá el manual de usuario en HTML de la vista de escritorio. Se debe documentar la utilización de la herramienta desde el punto de vista de un hipotético empleado.
4. ***instalacion.pdf***. Manual de instalación del sistema que describa qué pasos hay que seguir para la instalación y mantenimiento de la aplicación desarrollada, así como cualquier tipo de excepción requerida para su

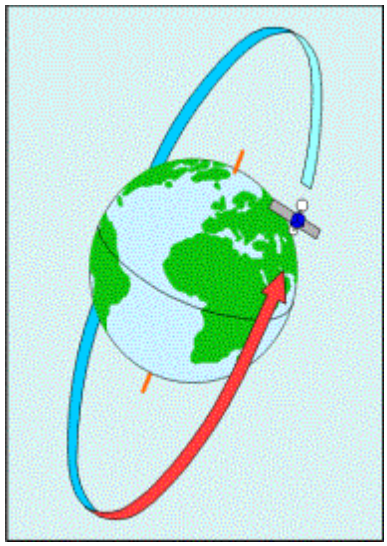
funcionamiento. En este apartado debes prestar especial atención a la configuración de la base de datos, señalando dónde y cómo configurar los parámetros para la conexión con el servidor de base de datos Oracle (lo cual incluye, dirección ip del servidor, puerto, nombre de usuario y contraseña) o cualquier otro tipo de servicio utilizado.

5. ***./proyecto***. Carpeta que contendrá todo el código fuente del proyecto desarrollado en JDeveloper.
6. ***./sysdoc***. Carpeta que contendrá la documentación de sistema generada con JavaDoc. Esta documentación debe ser suficientemente completa como para permitir a otro programador continuar con el proyecto.

Importante: Por compatibilidad entre plataformas es conveniente evitar el uso de acentos y letras como la ñ o la ç, en nombres de archivos, métodos y clases. Además, es conveniente que todos los nombres de archivos estén en minúscula. Es común que sistemas operativos tan comunes como Windows y Linux utilicen codificaciones de carácter diferentes (Linux generalmente utiliza UTF-8 y Windows generalmente utiliza ISO-8859-1 o ISO-8859-15), por lo que hay que intentar evitar caracteres que entren en conflicto.

Consideraciones para trabajos en grupo.

La idea de realizar proyectos en grupo es la de fomentar el trabajo en equipo. Por ese motivo, cuando se opta por trabajar en grupo cada miembro tiene el deber de participar tanto como si estuviera haciendo el proyecto de forma individual. Muchas veces, se puede pensar que trabajar en equipo es una mera división en partes iguales del trabajo, y en algunos casos puede ser así, pero en los proyectos informáticos casi nunca suele ocurrir eso.



Cuando se divide el trabajo de un proyecto informático, la coordinación del equipo de desarrollo suele llevar mucho tiempo, y dada esa dificultad, es muy fácil recurrir a un reparto de tareas desequilibrado, aunque posiblemente más cómodo.

Con el fin de evitar un reparto desequilibrado de tareas, creemos que los proyectos en grupo deben ser tratados de forma diferente. Por ello, los proyectos en grupo tendrán unas condiciones diferentes a los proyectos individuales. Estas condiciones son:

1. El número máximo de componentes del grupo es 2.

2. En la documentación a entregar, en el documento llamado "**miproyecto.pdf",** los miembros del grupo están obligados a incluir el apartado llamado "***Reparto del Trabajo***".
3. Independientemente de la división del trabajo, todos los miembros del grupo deben conocer el proyecto en profundidad, en previsión de posibles aclaraciones por parte del tutor o tutora.
4. **Requisitos mínimos para proyectos en grupo.** Como imaginarás, es de esperar que un proyecto desarrollado por un grupo implemente un número mayor de funcionalidades que un proyecto desarrollado individualmente. Para evitar ambigüedades, vamos a ver las características mínimas de los proyectos en grupo. Para ello, a los requisitos mínimos de un proyecto (definidos en apartados anteriores) debéis añadir los siguientes aspectos:
    1. La base de datos deberá tener al menos 6 tablas, en lugar de 4. Además, entre las tablas de la base de datos deberá haber al menos 3 claves externas, en lugar de 2.
    2. Para la vista web, debéis hacer funcionar al menos dos formularios de búsqueda (en lugar de uno), cada uno de los cuales debe trabajar con datos diferentes. Paralelamente a esto, debéis hacer funcionar dos formularios de inserción de datos, dos de modificación de datos, dos de eliminación y dos maestro-detalle. Recuerda que los maestro-detalle son independientes de los formularios de inserción y modificación. Es buena idea que enlacéis los formularios desde el resultado de una búsqueda previa, hará que vuestra aplicación parezca más profesional.

    3. Para la vista de escritorio es similar al caso anterior, pero desde el punto de vista del empleado. En este caso, el empleado podrá al menos realizar dos búsquedas de datos (sobre tipos de datos diferentes), podrá utilizar al menos dos formularios de inserción de datos, dos de modificación de datos, dos de eliminación y dos maestro-detalle. Y al igual que ocurría en el punto anterior, los maestro-detalle deberán ser independientes de los formularios de inserción y modificación.
    4. Para la vista de escritorio, además, el empleado podrá generar al menos tres informes con iReport/JasperReports en lugar de dos.

Proyecto globalizador: Desarrollo de una aplicación.

Si aún no tienes claro qué proyecto vas a realizar o cómo debe ser tu proyecto, no te preocupes, es posible que aquí se aclaren todas tus dudas. En primer lugar se expone un proyecto que puedes tomar como ejemplo de partida, y en segundo lugar se exponen una lista de proyectos entre los cuales puedes elegir.

Si decides escoger uno de los proyectos aquí propuestos, es posible que quieras añadir o modificar alguna característica que hagan mejor tu proyecto. Si es así, primero debes comunicar tales modificaciones a tu tutor o tutora, el cual debe autorizarlas. Después no debes olvidar comentar expresamente las características añadidas o modificadas en la documentación de tu proyecto,

concretamente en el documento ***"miproyecto.pdf"***, en la sección *"Resumen del proyecto elaborado".*

Ejemplo de proyecto: iVeterin.

El proyecto iVeterin es un proyecto didáctico, no es un proyecto en uso, pero con algunas modificaciones podría serlo. El motivo de presentarte aquí este proyecto es, simplemente, para que puedas hacerte una idea del proyecto que esperamos que entregues.

El recurso aquí disponible contiene la estructura de un proyecto como el que tienes que entregar, con todos los documentos y carpetas requeridas. Esperamos que te sea de mucha ayuda, aunque observarás que tiene algunas características más de las requeridas en el apartado de requisitos mínimos de tu proyecto. ¡¡No dejes que eso te agobie!!

El proyecto iVeterin es un proyecto para la gestión de una clínica veterinaria, que incluye la gestión de citas y la venta on-line de productos veterinarios. En el desarrollo de este proyecto se ha usado lo siguiente:

1. ADF Bussines Components para la implementación del modelo.
2. ADF Swing para la implementación la vista de escritorio.
3. ADF Faces para la implementación de la vista web.
4. iReport para la creación de informes.

Encontrarás más información del proyecto dentro del siguiente recurso:

iVeterinWebPlus.zip

Proyecto globalizador: Desarrollo de una aplicación.

Posibles proyectos.

Aquí tienes algunas propuestas de proyectos que puedes llevar a cabo. Si decides elegir uno de estos proyectos recuerda que debes comunicarlo a tu tutor o tutora. Todos los proyectos tienen dos perfiles, uno para desarrollo individual, y otro, a modo de ampliación del original, para desarrollo en grupo.

Es posible que estos proyectos te parezcan grandes y pienses que tienes poco tiempo para realizarlos. **Piensa que llevar a cabo todos los requisitos de los proyectos aquí expuestos junto con el resto de documentos requeridos**

significa que la tarea la tienes perfecta y con la máxima nota. No tienes la obligación de hacer una tarea perfecta, simplemente, haz lo que te de tiempo y tu tutor o tutora valorará tu esfuerzo de la forma adecuada.

Proyecto 1.- ***Restaurante Pizzina*** (Complejidad: media)

El restaurante ***Pizzina*** permite la reserva on-line de mesas, lo cual permite a sus clientes degustar los mejores platos italianos de la ciudad. Eso sì, para garantizar que la reserva no es falsa se exige a los clientes que faciliten un número de tarjeta de crédito durante la reserva. No tendrás que realizar la parte de validación bancaria del número de tarjeta de crédito, obviamente, pero sí tendrás que realizar una gestión adecuada de las reservas para que no haya varios clientes sentados en la misma mesa, en una mesa con insuficiente número de sillas, o para no ofrecer reservas en días que el restaurante está cerrado. Ten en cuenta que hay clientes que esperan más de 15 días por una mesa, ¿merecerá la pena tanta espera?

Pizzina

Proyecto 2.- ***TelePaella*** (Complejidad: media)

El restaurante ***TelePaella*** ofrece sus maravillosas paellas por Internet. Puedes elegir entre varios tipos de paella y otros sabrosos platos típicamente mediterráneos (así como una gran variedad de bebidas y postres). Abren todos los días del año excepto en vacaciones y festivos, y sólo admiten pedidos para ese mismo día y en unas zonas específicas de la ciudad. Para pedir uno o más platos todos sus clientes tienen que registrarse en el sistema. El pedido se servirá en la dirección que indique el cliente en el momento de la compra, teniendo en cuenta que mediante el código postal se comprobará si la zona donde vive el cliente entra dentro del área de venta. Además, al realizar la compra, los clientes deben facilitar una tarjeta de crédito válida para evitar falsos pedidos.

Telepaella

Proyecto 3.- ***JoJobs Trabajo Temporal S.L.*** (Complejidad: media)

La agencia de trabajo temporal ***JoJobs*** publica on-line todas las ofertas de trabajo que recibe. Sus clientes (los candidatos y candidatas a un trabajo) acceden a esas ofertas a través de la web, y a través de la web un candidato o candidata puede registrarse, modificar sus datos, entrar en el proceso de selección de las ofertas que seleccione, ver todas las ofertas de una misma empresa y ver los datos de una determinada empresa. Las ofertas de trabajo se publican durante un periodo determinado, después de ese periodo los candidatos no podrán suscribirse a la oferta de trabajo. ***JoJobs*** es una empresa de ámbito nacional, por lo que publicará ofertas desde varias sucursales y provincias.

Jojobs.pdf

Proyecto 4.- ***Recambios Christine S.L.*** (Complejidad: media)

Sin duda, ***Recambios Christine***, es la empresa más rápida sirviendo los recambios de coche a todos los talleres de la ciudad. El secreto de su éxito radica en su ágil gestión de existencias y previsión de demanda. Su sistema tiene ordenado cada recambio por la marca de coche para la que es válido, el cliente puede hacer un pedido on-line y recibirlo en menos de 24 horas. Trabajan todos los días del año.

Recambios_christine

Proyecto 5.- ***HorarioLibre 1.0*** (Complejidad: media-alta)

En este caso deberás desarrollar una pequeña aplicación de gestión de horarios para una empresa de tele-marketing. La empresa ha decidido crear un sistema flexible en el que cada empleado puede cambiar su horario de trabajo cada semana (sólo pueden cambiar el horario una vez por semana). Hay unos 100 puestos de trabajo en los que el empleado puede sentarse para llamar a los clientes y unos 150 empleados (repartidos en unos 5 proyectos diferentes) y podrán elegir la hora de trabajo en grupos de 4 horas (8 cada día). Eso si, la empresa fija dos horas al día, que serán las horas en las que haya menos teleoperadores trabajando en la empresa en cada uno de los proyectos. En este caso el teleoperador será el cliente que accederá vía web a la aplicación. ¿Podrás llevar a cabo este difícil proyecto?

Horariolibre.pdf

Proyecto 6.- ***IncidenciasPC 1.0*** (Complejidad: alta)

Los del departamento de informática de la empresa **FPAD S.A**. necesitan una aplicación para organizar las reparaciones y pedidos de sistemas informáticos dentro de la empresa. Para eso, el jefe de cada departamento tendrá una cuenta en el sistema ***IncidenciasPC***, mediante esa cuenta el jefe de departamento podrá solicitar la reparación o sustitución de un equipo informático perteneciente a su departamento, y/o pedir equipos nuevos. El departamento de informática tiene inventariados todos los equipos por departamento (apuntando además el nombre y DNI del empleado que lo usa), pudiendo localizar rápidamente el equipo en caso de avería. Además, cada 4 meses buscan las mejores ofertas en productos informáticos, creando un catalogo en el que los jefes de departamento pueden buscar rápidamente nuevas adquisiciones o sustituciones.

Incidenciaspc.pdf

Proyecto 7.- ***Park & Go! S.L.*** (Complejidad: media)

La empresa ***Park & Go!*** es una empresa gestora de varios parkings en varias ciudades. En una de ellas, va a instalar

un pionero sistema que permite a los clientes adquirir un bono en lugar de alquilar una plaza de forma permanente. De esta forma, el cliente podrá reservar una la plaza para los días del mes que necesite, el inconveniente es que un día le puede tocar una plaza y otro día le puede tocar otra, pero siempre tendrá una plaza si la ha reservado previamente. De esta forma ***Park & Go!*** maximiza la utilización de las plazas de parking y abarata el coste de los bonos, dando más facilidades a sus clientes. ¿Te parece una buena idea?

Park&go

En este apartado se exponen los aspectos que se evaluarán de tu proyecto y el peso que tendrán dentro de la nota de tu tarea, así como las características que harán que tu proyecto suba nota. Los aspectos que se evaluarán son los siguientes:

1. Presentación y organización del proyecto (30 % de la nota)
2. Contenido de la documentación (10% de la nota).
3. Operatividad del proyecto (55 % de la nota).
4. Defensa del proyecto (5% de la nota).

Proyecto globalizador: Desarrollo de una aplicación.

Presentación y organización del proyecto.

La presentación suele ser la primera impresión de un proyecto, y generalmente, lo que hace que entre por los ojos a un hipotético cliente. Por ese motivo, nos gustan las aplicaciones "*bonitas y bien organizadas*", qué se le va ha hacer, los tutores y tutoras somos así.

> ***En tu proyecto la presentación tendrá un peso del 30%.***

A nuestro entender, la presentación y organización de tu proyecto debería estar especialmente cuidada en tres aspectos:

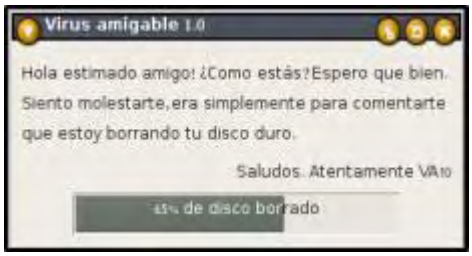
1. Presentación y organización de la documentación de usuario.
2. Presentación y organización de las interfaces de usuarios (vista web y de escritorio).
3. Estructura del código fuente.

El hecho de que se hayan remarcado estos aspectos no quiere decir que haya que descuidar la presentación y

organización de otros apartados, como la documentación
de sistema o los documentos que tienes que entregar a tu tutor o tutora.

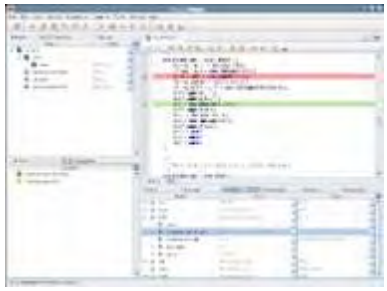
En estos aspectos, se valorarán los siguientes puntos:

1. **Presentación y organización de la documentación de usuario**.
    1. **Estructura de la documentación en secciones** asociadas a cada tipo de operaciones que el usuario puede hacer en el sistema (Gestión de clientes, Gestión de compras, etc.). Es recomendable que dentro de cada sección, la documentación esté organizada por operaciones que el usuario puede hacer (en gestión de clientes podríamos tener las operaciones de Alta Cliente, Baja Cliente, etc.).
    2. **Distinción adecuada de un apartado con el siguiente**, así como, si existen, de los diferentes subapartados entre si.
    3. Existencia de un **índice** que facilite el acceso rápido a cada apartado.
    4. Existencia de una **página principal** en la documentación.
    5. Usar un **tipo, tamaño y color** de letra que facilite la lectura.
    6. Usar **colores** agradables a la vista.
    7. Se valorará muy positivamente que la **documentación sea accesible por contexto**, es decir, mientras se está utilizando la aplicación.
2. **Presentación y organización de las interfaces de usuarios (vista web y de escritorio).**
    1. **Usabilidad.** En tu proyecto debes intentar que la aplicación sea fácilmente utilizable, usando para ello los elementos gráficos oportunos y enlazándolos adecuadamente. Por ejemplo, debemos intentar que sea fácil buscar a un cliente en nuestra aplicación aunque no recordemos el nombre exacto del cliente. Otro ejemplo, para ver las compras de un cliente simplemente debe bastar con hacer clic sobre el nombre del cliente en un listado.
    2. **Amigabilidad**. Cuando la aplicación no permita hacer una operación, cuando falte un dato para un formulario, o situaciones similares, la aplicación debe intentar dar una explicación al usuario de qué sucede, permitiéndole rectificar en lugar de abortar directamente. También es recomendable que en la interfaz de usuario exista información sobre lo que el usuario está realizando en ese momento y sobre cómo debe hacerlo.
    3. **Acceso fácil e intuitivo a todas las operaciones que se puedan realizar con la aplicación**.
    4. **Adecuada disposición de cada elemento gráfico respecto a los demás**. Hay que intentar evitar cosas como poner elementos gráficos muy separados o muy juntos, o poner elementos agrupados en una esquina dejando el resto del espacio libre.



    5. **Elementos gráficos bien dimensionados.** Hay que evitar cosas como poner un botón de Aceptar enorme o demasiado pequeño, o un logotipo que ocupe demasiado espacio.
    6. **Los elementos gráficos que contienen texto (como botones) deben tener un nombre descriptivo.** Por ejemplo, en un botón es preferible poner "Dar de alta cliente" a poner "Aceptar".
    7. Utilización de **colores y tipografía** que facilite la utilización de la herramienta.

3. **Estructura del código fuente.**
    1. **Cuidar la organización del código**, procurando que cada función de tu proyecto se resuelva en uno o más métodos de las clases adecuadas. En definitiva, intentar buscar una solución elegante.



    2. **Agrupar archivos de código fuente relacionados funcionalmente en un mismo paquete**. Dos archivos de código fuente estarán vinculados funcionalmente cuando el objetivo del código fuente que contienen esté destinado a completar la misma función, o funciones, en el caso de que sean varias.
    3. **Mantener la indentación en el código fuente**.
    4. **Poner nombres de variables, propiedades, métodos, clases, archivos y demás objetos que sean representativos.**

Contenido de la documentación.

Si es importante la presentación porque es lo primero que ve el cliente, la documentación es también muy importante porque es a lo primero que recurre cuando no sabe cómo hacer algo. Si la documentación está bien organizada y es útil, podrá resolver sus problemas rápidamente.

Lo mismo ocurre con la documentación de sistema. Si has incluido todos los comentarios javadoc adecuados, te será más fácil hacer una modificación en tu aplicación. Por todo esto, se valorarán especialmente los siguientes aspectos de la documentación, que hacen referencia al contenido de la misma:

1. **Documentación de usuario**
    1. **Documentación concisa y breve, pero completa**. Es decir, no debemos perder tiempo en explicaciones que probablemente no interesan al usuario final, hay que ir al grano.

    2. **Documentar todas las operaciones básicas que un usuario puede hacer**, indicando paso por paso como empezar cada operación y como terminarla.
    3. **Enlazar apartados de la documentación de temáticas similares** (en la sección de la documentación dedicada a *"Altas de cliente"*incluir un enlace a *"Modificación de datos de cliente"*)
    4. Incluir una sección llamada "***Problemas Comunes",*** donde se expliquen posibles excepciones de utilización de la aplicación. Por ejemplo:

Problema: ***No puedo dar de alta un producto.***

Posibles causas: ***Para poder insertar productos en el sistema debe haber al menos un departamento dado de alta para asociar el producto al departamento.***

2. **Documentación de sistema.**
    1. **Utilización de JavaDoc para la documentación de sistema, respetando la estructura y convenios necesarios para este sistema documentación.**
    2. Incluir una descripción concisa pero completa de cada método y clase creado.

En tu proyecto el contenido de la documentación tendrá un peso del 10%.

Proyecto globalizador: Desarrollo de una aplicación.

Operatividad del proyecto.

¿Qué es lo más importante que se le puede pedir a un proyecto?

Simplemente que funcione. Pues eso es lo que aquí te pedimos. En el apartado de "requisitos mínimos del proyecto" ya se expone este hecho, aquí simplemente te lo recordamos. Recuerda, tu proyecto debe estar completamente operativo en el momento de la entrega, lo cual significa que:

1. Lo más importante: **debe implementar todas las operaciones básicas descritas en el proyecto cumpliendo con los objetivos marcados en los requisitos mínimos**, ya se trate de un proyecto propio o de uno propuesto.
2. **La aplicación no debe terminar de forma abrupta o producir excepciones de forma inesperada como resultado de una acción de usuario.** Es decir, debes intentar que tu aplicación sea tolerante a fallos.
3. **Como resultado de una operación de usuario, la aplicación debe realizar la acción solicitada sin comprometer la integridad de la base de datos ni la lógica de negocio**. Por ejemplo, si se realiza una baja de un cliente sin borrar sus compras puede que se desencadene un fallo en la aplicación al consultar dichas compras, en tal caso deberás preparar tu aplicación si deseas conservar dicha información en la base de datos para su posterior consulta.

4. Se valorará especialmente que añadas características innovadoras a tu proyecto y/o que uses componentes que no han sido tratados en este módulo o que hayan sido tratados de forma superficial (EJBs por ejemplo).

En tu proyecto la operatividad tendrá un peso del 55%.

Defensa del proyecto.

Ya hemos comentado este aspecto anteriormente, en el apartado de "*Documentación a entregar*" aparece un punto llamado "*Defensa del proyecto*". Efectivamente, la defensa del proyecto es parte de lo que tienes que hacer, y como se comentó en aquel apartado, en esa sección tienes que intentar "*vendernos la aplicación*".

Dentro de este punto se valorarán los puntos fuertes de tu aplicación, las dificultades que has tenido en el desarrollo del proyecto y posibles mejoras sobre el mismo. Pero también se tendrán en cuenta otros apartados del documento ***miproyecto.pdf***, como el apartado de "*resumen del proyecto elaborado",* que servirá al tutor o tutora como orientación sobre el proyecto que has desarrollado, y las "*conclusiones*", que te servirá a ti para reflexionar sobre tu propio proyecto, sobre la utilidad de esta unidad y otros aspectos de tu interés.

En tu proyecto la defensa tendrá un peso del 5%.

Proyecto globalizador: Desarrollo de una aplicación.

Y esta es la última unidad, con lo que, ánimo y a por ello.

Te incluimos aquí dos mapas conceptuales que resumen los contenidos de la unidad:

Mapa1

Mapa2

¡¡¡¡ Felicidades por haber llegado hasta aquí !!!

Proyecto globalizador: Desarrollo de una aplicación.