Universidad de las Ciencias Informáticas

Facultad 4

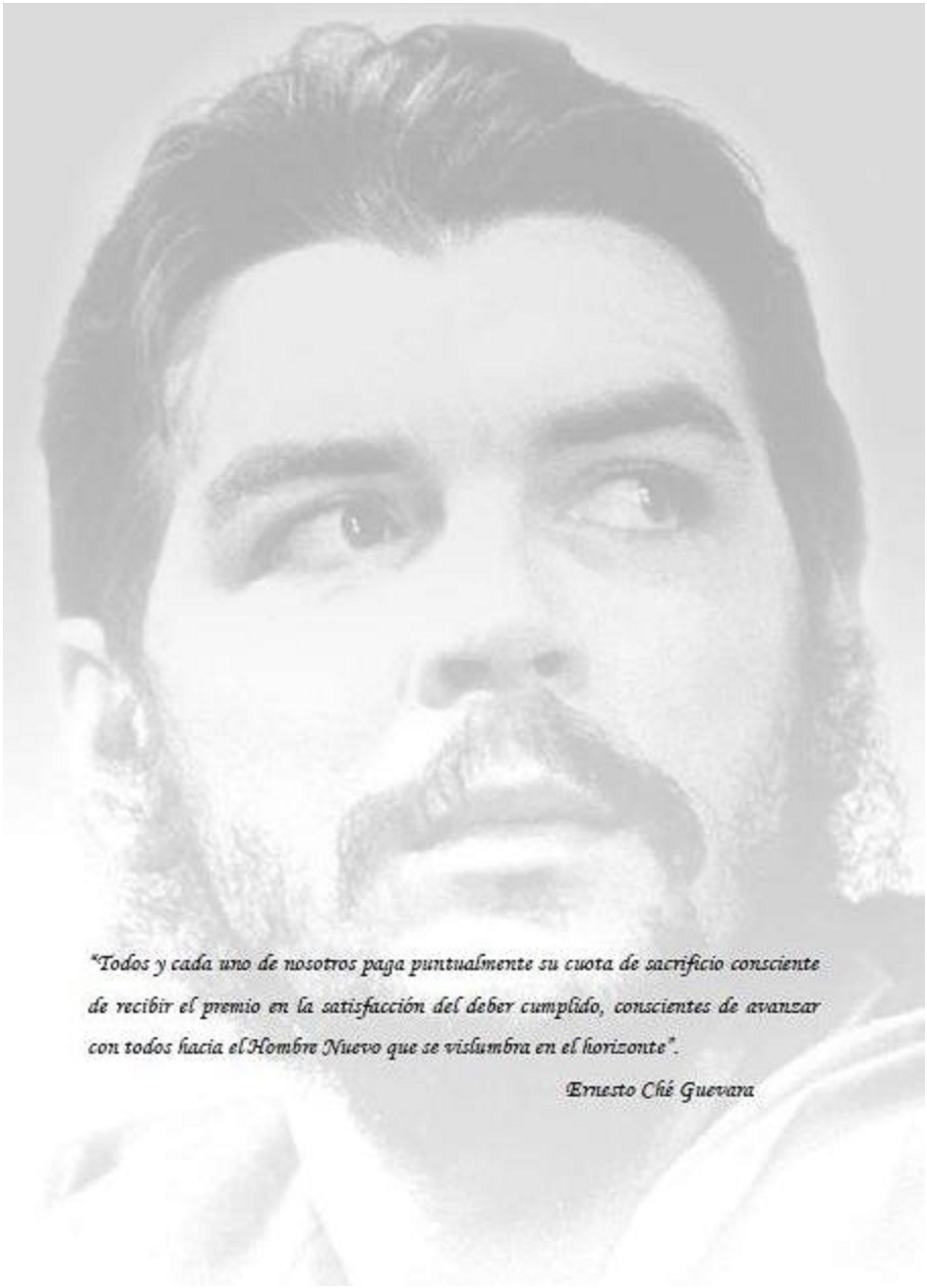

**Título:**

# Generación automática de interfaces gráficas en la Plataforma Java

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

**Autor(as): Yoanna Columbié Cisnero**
**Lissa Curbelo Oliva**
**Tutor: Ing. Adolfo M. Iglesias Chaviano**

Ciudad de La Habana, Junio de 2009



*"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte".*

*Ernesto Ché Guevara*

# Declaración de autoría

Declaramos ser los únicos autores de este trabajo y reconocemos a la Facultad 4 de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firmo la presente a los _____ días del mes de _____ del año _____ .

Autor:
Yoanna Columbié Cisnero

_____

Firma del autor

Autor:
Lissa Curbelo Oliva

_____

Firma del autor

Tutor:
Ing. Adolfo Miguel Iglesias Chaviano

_____

Firma del Tutor

# Datos del tutor:

Graduado en Julio del 2007 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas con Título de Oro y Promedio General de 5.09 puntos. Seleccionado Premio "Julio Antonio Mella" en la primera graduación de dicha Universidad. Ha impartido las asignaturas de Programación I y II, Ingeniería de Software I y II y el curso de Patrones de Diseño en la Universidad de las Ciencias Informáticas. Formó parte del equipo que elaboró dicho curso. Miembro de la Reserva del Comandante en Jefe. Participó como desarrollador en el proyecto SAFRE y en un Sistema de encuesta sobre temas de Nefrología. Analista y desarrollador del Subsistema Sala Situacional del Sistema de Gestión Penitenciaria para la República de Venezuela. Ha elaborado e impartido curso de capacitación al proyecto relacionado con la modernización del Sistema Bancario Cubano. Arquitecto Principal del proyecto relacionado con la modernización del Sistema Bancario Cubano.

Correo electrónico: aiglesias@uci.cu

# Agradecimientos

*A mi mamá por apoyarme todos estos años, por aconsejarme seguir adelante cuando pensaba que era el final. Por ser siempre tan preocupada por mis estudios y enseñarme a ser sacrificada y persistente. Por quererme tanto y por darme siempre lo mejor.*

*A mi hermano por siempre estar pendiente de mí y por brindarme su amor y cariño.*

*A mi prima Marila por ser como una hermana y estar siempre cuando lo necesité.*

*A Lester por tener tanta paciencia, por brindarme su amor y cariño todo este tiempo, por aconsejarme, por hacerme reir cuando en lo único que pensaba era en llorar. Por hacer que mis días en la Universidad fueran lo más lindo posible.*

*A Alberto por ser como un padre, por ayudarnos y apoyarnos siempre a mi mamá y a mi.*

*A mi compañera de tesis por apoyarme y aconsejarme cuando lo necesité y por brindarme ese espíritu que ella siempre tuvo de seguir adelante.*

*A mis compañeros de aula por estar siempre cuando los necesité, por apoyarme con las pruebas de nivel.*

*A mi tutor y a Orestes por sus consejos y por constribuir en la realización de la tesis.*

*A todas las personas que de una forma u otra me ayudaron no solo en la realización de la tesis sino también en el transcurso de mi carrera como profesional.*

*Lissa*

*A mis compañeros de aula por estar siempre a mi lado en los momentos difíciles y de alegría, por brindarme su ayuda y apoyo cuando los necesitaba.*

*A mi compañera de tesis y amiga, ella siempre ha estado atenta a lo que me suceda al igual que yo con ella,me apoya y me da consejos, sobre todas las cosas buenas que ella tiene como persona se sacrificó mucho para lograr entre las dos el desarrollo de este trabajo.*

*A mis padres que si no fuera por ellos yo no estuviera aquí hoy donde estoy, por brindarme su ayuda y apoyo a pesar de estar lejos de mi.*

*A mi mamá a la cual quiero mucho, mucho.... por hacerme la persona que hoy soy, y darme su cariño, comprensión y amor de madre en todos estos años.*

*A mi hermana y hermano por brindarme su apoyo, ayuda y amor a pesar de la distancia que hay entre nosotros.*

*A mis amistades que siempre han brindado su granito de arena para ser de mi una persona mejor y ayudarme en la realización de este trabajo de una forma u otra; en especial a la China, Mildrin, Lissin y Yack.*

*A Reinel, mi esposo que con su amor, paciencia, cariño, comprensión y amor ha logrado ocupar un pedazo en mi corazón.*

*A nuestro tutor por ayudarnos en la realización del trabajo.*

*A todas las demás personas que de una forma u otra han apoyado el desarrollo de este trabajo.*

*Yoanna*

# Dedicatoria

*Dedico este trabajo a la persona más linda y comprensiva de este mundo que es mi mamá, por su confianza y apoyo incondicional en todos estos años. A mi hermano y a Marila por apoyarme y escucharme siempre. Y a mis abuelos que se que estarían muy orgullosos de mi.*

*Lissa*

*A mi mamá por haberme dado el privilegio de nacer, ella es lo más grande para mi al brindarme en todos estos años de mi vida su amor y ternura de madre, y enseñarme a ser cada día más fuerte ante las adversidades y que cada error que se comete se convierte en una experiencia en la vida.*

*A mi esposo por brindarme en estos años todo su amor, comprensión, dedicación, apoyo en mis estudios y demás, entrega y enseñarme que todo en la vida es posible si se es paciente.*

*A mi tío que siempre me brindaba su amor, cariño, comprensión y ayuda, quiero que sepa que aunque no esté a mi lado físicamente yo siempre lo recuerdo, lo quiero y no lo olvidaré jamás. Tío tu anhelo de verme graduada ya te lo doy.*

*Yoanna*

# Resumen

El desarrollo de la Informática en todo el mundo y en especial en nuestro país ha conllevado a que la informatización se lleve a cabo en todas las esferas de la sociedad, para lograr la informatización de los procesos fundamentales de las instituciones, logrando mayor rapidez y confiabilidad del manejo de la información.

En atención a las dificultades que enfrentan los desarrolladores de software para construir interfaces gráficas en las aplicaciones de gestión, se efectuó la investigación de los frameworks que generan automáticamente interfaz gráfica a partir de un modelo en la plataforma Java, con el objetivo de proponer uno de ellos. Se hace un análisis además, de las características, ventajas, licencia, patrones y herramientas que utilizan para su desarrollo estos frameworks.

Para alcanzar el objetivo propuesto se realizó una amplia revisión bibliográfica, lo que permitió elaborar y exponer la fundamentación teórica de los frameworks que generan automáticamente interfaz gráfica a partir de un modelo en la plataforma Java.

Se presentó el framework propuesto para la solución del problema, su definición, componentes que utiliza, concurrencia y escalabilidad entre otros. El framework propuesto genera automáticamente a partir de un modelo (una clase java) una interfaz gráfica por defecto, en este no es necesario escribir la vista ya que genera una por defecto aunque permite modificar la misma y el controlador puede reutilizarse normalmente.OpenXava está basado en el concepto de componente de negocio lo que le permite definir toda la información sobre un concepto de negocio en un único sitio.

**Palabras claves:** aplicaciones de gestión, componente de negocio, framework, interfaz gráfica.

# Índice general

# Índice de figuras

# Introducción

Desde el surgimiento de la Informática, esta ha ido evolucionando al pasar de los años hasta nuestra actualidad; convirtiéndose en unas de las principales ciencias que se desarrollan actualmente. En el mundo de la Informática se encuentran inmersos la mayoría de los países y organizaciones mundiales. Cuba, a pesar de ser un país subdesarrollado y bajo numerosas presiones de diversa índole que le frenan su desarrollo tecnológico, no ha estado ajeno en el desarrollo del avance tecnológico; para lograr esto en nuestro país se han creado estrategias y políticas inteligentes que permitan tener acceso a las Tecnologías de la Informática y las Comunicaciones (TICs), estas políticas tienen como objetivo el uso racional, equitativo, igualitario y educativo de estas tecnologías, las cuales tienen como fin la obtención de habilidades y el enriquecimiento de los conocimientos del hombre.

En nuestro país, al calor de la batalla de ideas, surge la Universidad de las Ciencias Informáticas (UCI); cuyo objetivo es formar profesionales que contribuyan al desarrollo informático de la sociedad cubana y a la vez al desarrollo de la industria del software nacional. En nuestra universidad se desarrollan aplicaciones de gestión, que se diseñan para sustituir uno o varios procedimientos, tanto comerciales como administrativos, que habitualmente realiza una persona en una empresa o institución de forma presencial.

Algunas de las características que presentan las aplicaciones de gestión son: seguras, escalables, flexibles, amigables y rápidas. El flujo de trabajo en el desarrollo de estas aplicaciones de gestión se divide en tres grandes módulos o capas, las cuales son: interfaz gráfica, lógica de negocio y acceso a datos.

**Lógica de negocio:** es la parte de un sistema que se encarga de las tareas relacionadas con los procesos de un negocio, tales como ventas, control de inventario, contabilidad, etc. Son rutinas que realizan entradas de datos, consultas a los datos, generación de informes y más específicamente todo el procesamiento que se realiza detrás de la aplicación visible para el usuario.

**Acceso a Datos:** es la parte que encapsula las funcionalidades en un componente común, es

reusable, no está vinculada a una base de datos o aplicación en particular; define las funcionalidades esenciales parametrizadas, lo cual conlleva a consumir más recursos de conexión a la base de datos e implementar caché de acciones.

**La interfaz gráfica:** es cualquier medio por el cual se puede interactuar con una computadora a través de algún tipo de software gráfico. Comúnmente, esto se consigue a través del control mediante el teclado y el mouse de cursores, menús, ventanas, íconos y cajas de diálogo, pero puede tomar cualquier forma imaginable.

El objetivo final de dichas interfaces es que los programas sean más amigables y fáciles de usar para los usuarios; no obstante, la necesidad de utilizar interfaces gráficas en los programas impone una carga extra a los diseñadores y desarrolladores de software, pues deben incorporarlas en sus aplicaciones aún en etapas muy tempranas del desarrollo; además de que estos deben tener dominio de los lenguajes JavaScript, HTML, entre otros y conocer los componentes visuales que se utilizan en la construcción de estas interfaces.

Debido a esto surge la **situación problemática** la cual plantea: el desarrollo de interfaces gráficas en un sistema informático de corte empresarial se convierte en el flujo de desarrollo más lento dentro del desarrollo de un sistema, sin olvidar además la complejidad que esto lleva. Por lo que es necesario tener un equipo bien preparado, con experiencia en el desarrollo de interfaces gráficas en un entorno web si se quiere tener seguridad a la hora de enfrentar el desarrollo de un sistema. Como no siempre se cuenta con el personal altamente calificado es necesario investigar las tecnologías y herramientas que existen para el desarrollo de interfaces gráficas de manera automática sin pasar por todo el largo proceso de desarrollo de estas interfaces.

A raíz del análisis de lo anteriormente planteado se establece como **Problema**: ¿cómo lograr un mejor desarrollo de las interfaces gráficas a partir de un modelo en la plataforma Java?

El **objeto de estudio** en el cual se enmarca el problema anteriormente planteado es: el desarrollo de aplicaciones de gestión.

Y el **campo de acción**: generación de interfaz gráfica a partir de un modelo en la plataforma Java.

Para el diseño de la investigación se plantea la siguiente **Hipótesis**: si proponemos un framework que genere automáticamente interfaces gráficas a partir de un modelo en la plataforma Java, entonces, el desarrollo de interfaces gráficas en las aplicaciones de gestión sería mucho más fácil.

Y se plantea como **Objetivo General del Trabajo**: proponer la utilización de un framework para la generación de interfaces gráficas de manera rápida a partir de un modelo en la plataforma Java.

Para dar cumplimiento al objetivo general se plantean los siguientes **Objetivos Específicos:**

1. Evaluar los frameworks que generan automáticamente interfaz gráfica a partir de un modelo en la plataforma Java.

2. Elaborar el marco teórico de la investigación.

3. Proponer a partir de un caso de estudio la utilización de un framework para la generación automática de interfaces gráficas a partir de un modelo en la plataforma Java.

El presente trabajo está compuesto por Introducción, tres capítulos, treinta y cinco epígrafes, veintidós subepígrafes, Conclusiones, Recomendaciones, Bibliografía , Glosario de Términos.

En el Capítulo 1, Fundamentación Teórica, se realiza un estudio sobre los frameworks que generan automáticamente interfaz gráfica a partir de un modelo en la plataforma Java. De estos se analizan las características, ventajas, licencia, patrones que utilizan así como las herramientas que usan para su desarrollo . Al finalizar el capítulo se propone cuál de los frameworks utilizar para desarrollar el caso de estudio.

En el Capítulo 2, Presentación de la Solución Propuesta, se realiza la descripción detallada del framework propuesto en el capítulo anterior: OpenXava.

En el Capítulo 3, Caso de Estudio, se profundiza en OpenXava, framework utilizado en el desarrollo del Caso de Estudio. Son analizados a detalle los pasos a seguir tanto para realizar la conexión del Tomcat con el Sistema Gestor de Bases de Datos ( PostgreSQL), así como para crear un nuevo proyecto, paquete y clase.

# Capítulo 1

# Fundamentación Teórica

## 1.1. Introducción

En el presente capítulo se proporciona una visión general de los principales frameworks que generan automáticamente interfaces gráficas a partir de un modelo en la plataforma Java, realizando un estudio del tema.

## 1.2. Fundamentos teóricos de los frameworks que generan automáticamente interfaces gráficas a partir de un modelo en la plataforma Java

El tema de los frameworks es algo de reciente desarrollo e investigación. En el presente capítulo se exponen los conceptos fundamentales, características, ventajas entre otros, relacionados a los frameworks y de algunos frameworks que generan automáticamente interfaces gráficas a partir de un modelo en la plataforma Java.

## 1.3. Definiciones de framework

"...diseño abstracto orientado a objetos para un determinado tipo de aplicación, que se compone de una clase abstracta para cada componente principal del diseño; contendrá normalmente una librería

de subclases que pueden ser utilizadas como componentes del diseño. . . " [29].

Un framework es:

- Una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, librerías y unir los diferentes componentes de un proyecto.
- En Java es un conjunto de clases y librerías las cuales proporcionan soluciones completas que contemplan herramientas de apoyo y motores de ejecución.
- La piedra angular de la moderna ingeniería del software. El desarrollo de frameworks está ganando rápidamente una amplia aceptación debido a su capacidad para promover la reutilización del diseño y del código fuente[33].
- Un producto de software[56] , por lo que se debe aplicar un proceso similar para su desarrollo.

Estos proporcionan una estructura al código y hace que los desarrolladores escriban código más entendible, hace la programación más fácil, convirtiendo complejas funciones en sencillas instrucciones. Algunos autores, plantean que un framework puede ser visto como la implementación de un conjunto de patrones[30].

### 1.3.1. Ventajas de los framework

- Mayor productividad del desarrollador: al estar los frameworks enmarcados en un dominio, permiten que los desarrolladores sean productivos inmediatamente.
- Menos codificación: muchos de los códigos personalizados, ahora son reemplazados en su totalidad por el framework, o es una configuración del mismo; por lo que existe menos código.
- Resultados inmediatos: al dinamizar el proceso de asociación de los requerimientos a funcionalidades, se reduce considerablemente el tiempo de desarrollo. Esto permite un mayor número de iteraciones en un corto plazo, lo que conlleva a un producto de más calidad y mayor inmediatez en la entrega.
- Mayor flexibilidad: el bajo acoplamiento proporcionado por los frameworks aporta una gran flexibilidad, que responde a los cambios de los requisitos tanto del negocio como de tecnología.

- Arquitectos más eficaces: el esqueleto o estructura proporcionada por los frameworks permite que los arquitectos dediquen más tiempo a la toma de decisiones imprescindibles en vez de gastar una enorme cantidad de tiempo en hacer cumplir las mejores prácticas.

## 1.4. Definición de patrón

Los patrones son normas de comportamiento, características que identifican una situación. En informática un patrón es una solución a un problema que recibe un nombre y que puede emplearse en otros contextos[31].

Un patrón es una unidad de información nombrada, instructiva e intuitiva que captura la esencia de soluciones probadas a un problema recurrente dentro de un cierto contexto. El objetivo de los patrones es crear un lenguaje común a una comunidad de desarrolladores para comunicar experiencia sobre los problemas y sus soluciones. Pueden referirse a distintos niveles de abstracción, desde un proceso de desarrollo hasta la utilización eficiente de un lenguaje de programación.

### 1.4.1. Características de los Patrones

1. Atacan problemas recurrentes que ocurren en situaciones específicas y dan una solución.

    *a*) Variabilidad de la interfaz.

2. Documentan experiencias de diseño existentes y bien probadas.

    *a*) Experiencia en el desarrollo de sistemas interactivos.

3. Identifican y especifican abstracciones de alto nivel.

4. Proveen un vocabulario común y comprensible.

5. Son una forma de documentar la arquitectura del software.

6. Facilitan la construcción de software con propiedades definidas.

    *a*) Interfaces intercambiables y modelo reutilizable.

7. Ayudan a construir arquitecturas heterogéneas y complejas.

### 1.4.2. Patrones Arquitectónicos

Según Buschmann los patrones arquitectónicos se pueden ver como la descripción de un problema en particular y recurrente de diseño, que aparece en contextos de diseño arquitectónicos específicos, y representa un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma en que estos colaboran entre sí[5].

Los patrones arquitectónicos se definen sobre aspectos fundamentales de la estructura del sistema software, donde se especifican un conjunto de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes.

### 1.4.3. Patrón NakedObjects

Este patrón se basa en tres principios fundamentales[17]:

1. Toda la lógica de negocio debe encapsularse en los objetos del dominio.
2. La interfaz de usuario debe ser una representación directa de los objetos del dominio, con todas las acciones de usuario consistiendo, explícitamente, en crear o recuperar objetos del dominio y la invocación de métodos en esos objetos.
3. La idea principal de este patrón se basa en la combinación de los dos principios anteriormente mencionados, para así dar formación al tercer principio: la interfaz de usuario se debe crear 100 % automáticamente de la definición de los objetos del dominio. Esto se puede hacer usando diversas tecnologías, incluyendo la generación del código fuente.

#### 1.4.3.1. Ventajas

- Un ciclo de desarrollo más rápido, porque hay un menor número de capas para desarrollar.
- Mayor agilidad, refiriéndose a la facilidad con que una solicitud puede ser modificada para dar capacidad a futuros cambios en los requerimientos del negocio.
- Potenciación de estilo a la interfaz de usuario, al ser la interfaz de usuario orientado a objetos ofrece al usuario mayor flexibilidad que la interfaz de usuario orientado a tareas.

- Más fácil el análisis de requisitos. Con el patrón NakedObject el modelo del dominio forma un lenguaje común entre el usuario y los desarrolladores, por lo que se hace más fácil la discusión de los requisitos. Además se combina con un ciclo de desarrollo rápido posibilitando aplicaciones funcionales de prototipos en tiempo real.

#### 1.4.3.2. Utilidad

Después de hacer un análisis del patrón NakedObjects se puede señalar que este facilita una mayor agilidad y un rápido desarrollo de los framework de generación automática de interfaz gráfica en la plataforma java. Algunos de estos frameworks que utilizan este patrón para su desarrollo son: OpenXava, Naked Objects, Trails y Jmatter.

## 1.5. Frameworks que generan automáticamente interfaces gráficas a partir de un modelo en la plataforma Java

Existen diferentes frameworks que generan automáticamente interfaces gráficas en la plataforma Java, a continuación se conceptualizarán y caracterizarán algunos de ellos:

### 1.5.1. OpenXava

Es un framework de código abierto para desarrollar aplicaciones de gestión de manera rápida y fácil, el cual permite definir aplicaciones simplemente con POJOs (Plain Old Java Object), JPA (Java Persistence API) y anotaciones de Java y con estilo Orientado a Objetos. Además está basado en el concepto de componente de negocio. Con OpenXava se puede poner las clases JPA y a cambio se obtiene una aplicación completa, lista para poner en producción[49].

Con este framework se desarrollan aplicaciones de gestión, usando JPA o portales Java (como Liferay WebSphere Portal o JetSpeed). OpenXava fue creado para hacer aplicaciones de gestión internas (ejemplo Inventario, Nóminas, etc).

OpenXava no es solo para escribir mantenimientos simples para clases simples, sino también mediante este se pueden crear aplicaciones con lógica compleja e interfaces de usuario avanzadas. Este framework soporta referencias, colecciones, herencia, pestañas anidadas, marcos anidados para agrupar información, etc. Es un proyecto de código abierto LGPL (Library General Public License), nacido en

España, pero que actualmente cuenta con una comunidad de programadores internacional[20].

#### 1.5.1.1. Características

- Alta productividad para aplicaciones de gestión; porque en OpenXava solo se escribe el modelo, los controladores se reusan y la vista se genera automáticamente.
- Suficientemente flexible, para crear aplicaciones sofisticadas.
- Basado en el concepto de componente de negocio.
- Adaptado para soportar servidores de aplicaciones como Tomcat, JBoss y WebSphere.
- Adaptado para trabajar con esquemas de bases de datos. Soporta Gestores de Bases de Datos como: Oracle y PostgreSQL.
- Está probado con los portales: JetSpeed 2, WebSphere Portal, Liferay y Stringbeans.
- Mecanismo de persistencia: EJB3 JPA (Enterprise JavaBeans JPA) e Hibernate.
- Soporte completo de AJAX: no se produce ninguna recarga de página.

#### 1.5.1.2. Ventajas

- Es un Motor de Aplicación JPA, se pueden poner las clases JPA y a cambio se obtiene una aplicación completa lista para poner en producción.
- Permite el desarrollo rápido, es fácil de mantener y soporta generación automática de listados.
- Es lo suficientemente flexible para desarrollar complejas aplicaciones de gestión como contabilidad, facturación, gestión de personal, nóminas, gestión de almacenes, etc.
- No se tiene que escribir JavaScript, HTML (Lenguaje de Marcas de Hipertexto), JSP (JavaServer Pages), etc. La interfaz de usuario se genera automáticamente a partir de clases del modelo.

#### 1.5.1.3. Filosofía

La filosofía subyacente es definir las entidades, editores, configuraciones, entre otros, con anotaciones de Java o con XML (lenguaje de marcas ampliable) y programar con Java. El objetivo principal es hacer que las cosas típicas en una aplicación de gestión sean fáciles de hacer, ofreciendo la flexibilidad suficiente para desarrollar las funciones avanzadas y específicas.

#### 1.5.1.4. Tendencias

Actualmente OpenXava genera aplicaciones web Java (J2EE/JavaEE), que pueden ser desplegadas en cualquier portal Java (JSR-168) como una aplicación de portlets.

#### 1.5.1.5. Licencia

Se distribuye bajo la licencia LGPL.

#### 1.5.1.6. Patrones que utiliza

OpenXava utiliza para su desarrollo el patrón NakedObjects.

#### 1.5.1.7. Herramientas que utiliza

El archivo de descarga de OpenXava viene con un *workspace* configurado para Eclipse. Pero OpenXava no usa recursos de Eclipse sino que está basado en la herramienta de código abierto "Ant", por lo que sin ningún problema se puede usar con cualquier otro IDE (Entorno de desarrollo integrado).

### 1.5.2. Naked Objects

Es un framework radical para el desarrollo y la entrega de aplicaciones empresariales, desarrollando aplicaciones de código abierto en la plataforma Java. Genera una UI (interfaz de usuario) desde objetos Java. En tiempo de ejecución el framework Naked Objects examina los objetos del dominio usando la reflexión, y después hace los objetos y los métodos visibles al usuario[39].

Framework de Java en el que se busca que los objetos de negocio sean responsables no sólo de la lógica de negocio sino también del control de las acciones y de mostrar su estado visual, es decir que cada objeto sea modelo, vista y controlador[21]. El modelo lo controla todo, incluyendo las modificaciones, la teoría que sigue es que los requerimientos son expresados y volcados en el modelo, cuando un

requerimiento cambia, el modelo ha de cambiar y se dispara una serie de cambios en las diferentes capas de la aplicación.

#### 1.5.2.1. Características

- Presenta una arquitectura de cuatro capas: capa presentación, capa de control, capa de modelo del dominio y capa de persistencia.
- Apoya diferentes estilos de la interfaz de usuario.
- Crea una Interfaz de usuario orientada a objetos (OOUI) automáticamente.

#### 1.5.2.2. Ventajas

- Crea automáticamente un objeto orientado a la interfaz de usuario y a la base de datos.
- Utiliza el patrón NakedObject.
- Proporciona un lenguaje común entre el desarrollador y el usuario, lo que mejora significativamente la comunicación durante el proceso de exploración de las necesidades.

#### 1.5.2.3. Tendencia

Está disponible en dos formas:
1- Como un producto comercial se encuentra en Microsoft. NET.
2- Como un producto de código abierto se encuentra en la plataforma Java.

#### 1.5.2.4. Patrones que utiliza

Utiliza el patrón NakedObjects que permite obtener un código más limpio, fácil de mantener y permite el cambio de la presentación de manera simple.

#### 1.5.2.5. Licencia

La licencia cambió del GPL(Licencia Pública General) a Apache.

### 1.5.3. Framework MDA (Model Driven Architecture)

La esencia de MDA es generar una aplicación completa (una Interfaz de Usuario) desde un modelo UML (Lenguaje Unificado de Modelado). EL principal elemento del diseño de software es el modelo y la interfaz de usuario es generada. Este permite el desarrollo de aplicaciones empresariales potencialmente en cualquier plataforma existente, abierta o propietaria (servicios Web, J2EE (Java Platform Enterprise Edition), CORBA, .Net u otras).

Representa un nuevo paradigma de desarrollo de software en el que los modelos guían todo el proceso de desarrollo. Este nuevo paradigma se ha denominado Ingeniería de modelos o Desarrollo basado en modelos. MDA se centra en diferenciar entre la descripción del sistema de manera independiente de plataforma y las posibles realizaciones de esa especificación. Este no proporciona indicaciones sobre cómo se deben diseñar los sistemas[?].

#### 1.5.3.1. Características

- Es dirigido por modelos, porque utiliza estos para guiar el ámbito del desarrollo, el diseño, la construcción, el despliegue, la operación, el mantenimiento y la modificación de los sistemas.
- Presenta tres modelos para el desarrollo de las aplicaciones: PIM (Modelo Independiente de la Plataforma), PSM (Modelos Específicos de la Plataforma) y CIM (Modelo Independiente de la Computación)[34].

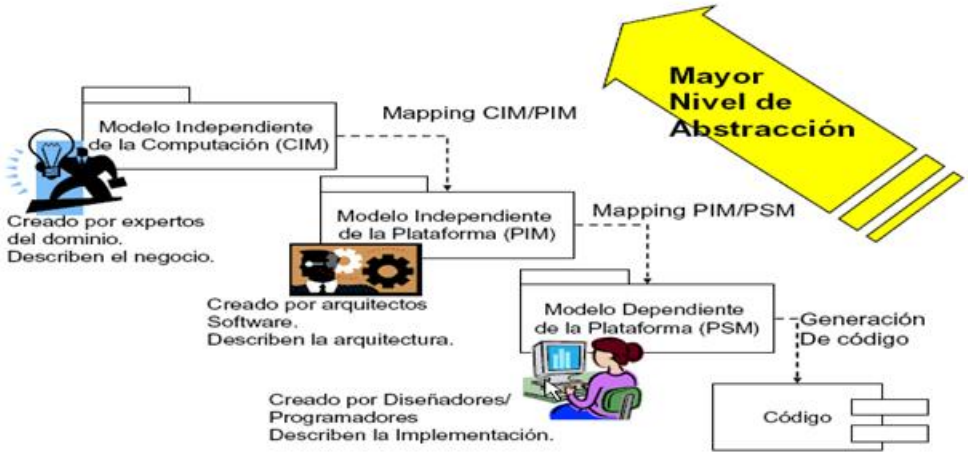



Figura 1.1: Modelos del MDA

#### 1.5.3.2. Ventajas

- Separa la lógica de negocio de la infraestructura que la implementa.
- Interoperabilidad de plataforma.
- Provee una arquitectura que permite independencia de plataforma.
- Generación automática de código para cualquier lenguaje.
- Detalles de implementación separados de la lógica de negocio.
- Dinamiza el proceso de integración entre sistemas "*legacy*" y sistemas futuros.
- Soporte completo para el ciclo de vida de la aplicación.
- Escalabilidad y robustez.
- Determina los tipos de modelos que deben ser usados, como preparar dichos modelos y las relaciones que existen entre ellos[42].

#### 1.5.3.3. Filosofía

MDA pretende separar por un lado la especificación del proceso y los datos de un sistema y por el otro a la plataforma en la cual será construido. Es por esto que establece tres puntos de vista que se emplearán a lo largo del proceso de ingeniería:

**Punto de vista independiente de la computación:** se centra en el entorno del sistema y los requisitos para el mismo. Los detalles de la estructura y procesamiento del sistema no se muestran, o aún no están determinados.

**Punto de vista independiente de la plataforma:** se centra en las operaciones del sistema, mientras oculta los detalles necesarios para una determinada plataforma. Muestra aquellas partes de la especificación del sistema que no cambian de una plataforma a otra.

**Punto de vista de plataforma específica:** combina el punto de vista independiente de la plataforma con un enfoque específico para su uso en una plataforma en un sistema.

#### 1.5.3.4. Herramientas

- AndroMDA de Gentleware.
- ArcStyler de Interactive Objects.
- OptimalJ de Compuware.
- Codagen Architecture de Codagen.
- IQGen de InnoQ.

#### 1.5.3.5. Componentes

- Metalenguaje.
- Lenguaje de definición de transformaciones.
- Definición de la transformación.
- Herramienta de Transformación.

#### 1.5.3.6. Normas

El modelo MDA está relacionado con múltiples normas, incluido el Lenguaje Unificado de Modelado (UML), Meta-Object Facility (MOF), el intercambio de metadatos XML (XMI), Empresa de Computación Distribuida Objeto (EDOC), el Proceso de Ingeniería de Software metamodelo (SPEM) y el almacén metamodelo común (CWM).

#### 1.5.3.7. Uso del MDA

El modelo de origen es el CIM, con el que se modelan los requisitos del sistema, describiendo la situación en que será usado y que aplicaciones se espera que el lleve a cabo, sirviendo de ayuda para entender el problema como una base de vocabulario para usar en los demás modelos. Los requisitos recogidos en el CIM han de ser trazables a lo largo de los modelos PIM y PSM que los implementan. El CIM consiste en un par de modelos UML que muestren tanto la distribución de los procesos (ODP, Open Distributed Processing) como la información a tratar. También puede contener algunos modelos UML adicionales.

A partir del CIM, se construye un PIM, que muestra una descripción del sistema, sin hacer referencia a ningún detalle de la plataforma. Debe presentar especificaciones desde el punto de vista de la empresa, información y ODP. Un PIM se puede ajustar a un determinado estilo de arquitectura, o a varios.

### 1.5.4. RomaFramework

Genera aplicaciones desde POJOs usando archivos XML para personalizar la vista. Es de código abierto, es también una aplicación del MDA, ya que toma los conceptos y la filosofía, pero todos basados en POJO. Tiene por objetivo proporcionar la integración con las tecnologías más avanzadas. Permite desarrollar aplicaciones Java de nivel empresarial[44].

#### 1.5.4.1. Características

- Ofrece soporte automático para todos los niveles de la aplicación.
- Permite construir aplicaciones Web con Ajax.
- Permite desarrollar aplicaciones Java de nivel empresarial.

#### 1.5.4.2. Ventajas

- Tiene por objetivo eliminar la dependencia entre el código de la aplicación y las herramientas que utiliza, permitiendo cambiar la tecnología sin cambiar el código de la aplicación.
- Permite desarrollar proyectos en Java de manera fácil y práctica, integrar frameworks y herramientas usando un Metaframework.

#### 1.5.4.3. Licencia

Se distribuye bajo la Licencia V2.0 de Apache y también cuenta con una licencia comercial.

#### 1.5.4.4. Filosofía

Utiliza la misma filosofía del framework MDA en la cual se pretende separar por un lado la especificación del proceso y los datos de un sistema y por el otro la plataforma en la cual será construido. Para esto se establecen tres puntos de vista:

- Punto de vista independiente de la computación.
- Punto de vista independiente de la plataforma.
- Punto de vista de plataforma específica.

### 1.5.5. Framework Trails

Genera aplicaciones Web desde POJOs anotados. Permite el desarrollo rápido de aplicaciones de forma automática al crear la lista y al editar las páginas POJOs. Es un dominio en el marco de desarrollo impulsado por el espíritu de Ruby on Rails y Naked Object. Apunta al desarrollo de aplicaciones Java radicalmente más simple, permitiendo a los desarrolladores centrarse en el modelo de dominio y teniendo otras partes generadas dinámicamente, además es de código abierto[11].

#### 1.5.5.1. Características

- Crea representaciones y editores en función de su tipo.
- Es un framework completo de pila de aplicaciones web. (Una pila de aplicación web tiene como objetivo principal: proporcionar un conjunto único de los componentes, que abarcan todo el ámbito de las cosas que necesitará para construir aplicaciones web).
- Proceso de desarrollo impulsado por el modelo de dominio DDD (Domain Driven Development).

#### 1.5.5.2. Ventajas

- Permite desarrollar aplicaciones web en J2EE con el menor número de pasos redundantes[25].
- Validación, función y seguridad.

#### 1.5.5.3. Componentes

Consta de tres componentes principales:

1- Un servicio de persistencia.

2- Un descriptor de servicios que proporciona metadatos para el dominio.

3- Un editor de servicios que proporciona el editor de componentes correspondientes basado en metadatos.

#### 1.5.5.4. Licencia

Se distribuye bajo la licencia V2.0 de Apache.

#### 1.5.5.5. Patrón

El patrón que utiliza es el NakedObject.

### 1.5.6. Framework Jmatter

Es un framework de código abierto que construye aplicaciones Swing de gestión para el trabajo en grupo, basándose en clases de modelo y Hibernate para la persistencia. Está enfocado a la creación de aplicaciones empresariales y cuenta con elementos de infraestructura listos para usarse como seguridad, funcionalidades CRUD, funcionalidades de búsqueda, servicios remotos, generación de informes, etc[28].

JMatter es un framework que pretende multiplicar la productividad de los programadores al construir aplicaciones de escritorio que accedan a una base de datos. Para ello existen un conjunto de convenciones de nomenclatura y opciones de configuración por defecto, unido a la posibilidad de construir el esqueleto de una aplicación que acceda a una base de datos simplemente con un comando de "Ant"[22].

#### 1.5.6.1. Características

- Maneja un estándar Naked Objects Architectural Pattern con el cual mientras se escriben las clases se va desarrollando automáticamente la interfaz.
- Una de las diferencias de JMatter con otros frameworks Java es que la interfaz de usuario que genera es 100 % Swing y la distribuye mediante Java Web Start.

#### 1.5.6.2. Licencia

Se distribuye bajo la licencia GPL y también cuenta con una licencia comercial.

#### 1.5.6.3. Patrones que utiliza

Este framework es una implementación del patrón NakedObject.

## 1.6. Conclusiones

En este capítulo se ha realizado la Fundamentación Teórica de los frameworks que generan automáticamente interfaz gráfica a partir de un modelo en la plataforma Java, a continuación se ofrece un breve resumen de los framework caracterizados en el presente capítulo:

- OpenXava, framework de código abierto para desarrollar aplicaciones de gestión de manera rápida y fácil, solo se necesita escribir el modelo, los controladores se reusan y las vistas se genera automáticamente, permitiendo así lograr una alta productividad en el desarrollo de estas aplicaciones. Está basado en el concepto de componente de negocio (Modelo, Vista, Datos tabulares, Mapeo objeto-relacional) porque permite definir toda la información sobre un concepto de negocio en un único sitio.

- Naked Objects, framework de código abierto para el desarrollo y entrega de aplicaciones empresariales, permite crear una interfaz de usuario orientada a objetos automáticamente, utiliza el patrón NakedObject, lo que facilita obtener un código limpio y fácil de mantener. En este framework el modelo lo controla todo, incluyendo las modificaciones, la teoría que lo guía es que los requerimientos son expresados y volcados en el modelo, cuando un requerimiento cambia, el modelo ha de cambiar y se dispara una serie de cambios en las diferentes capas de la aplicación.

- RomaFramework, framework que genera aplicaciones desde POJOs usando archivos XML para personalizar la vista. Permite desarrollar proyectos en Java de manera fácil y práctica, integrar frameworks y herramientas usando un Metaframework, desarrollar aplicaciones Java de nivel empresarial, eliminar la dependencia entre el código de la aplicación y las herramientas que utiliza permitiéndole cambiar la tecnología.

- Framework MDA está basado fundamentalmente en UML y otros estándares de la industria software para visualizar, almacenar, e intercambiar los modelos especificados desde los sistemas. Se centra en los requisitos funcionales de un sistema y no en la tecnología empleada para su implementación. Bajo la estrategia MDA, un sistema debe ser especificado en un modelo, no en el código ya que esta tarea deberá automatizarse a través de herramientas de generación de código.

- Trails, framework de código abierto, que genera aplicaciones Web desde POJOs anotados, apunta al desarrollo de aplicaciones Java, permitiendo a los desarrolladores centrarse en el modelo de

dominio, y teniendo otras partes generadas automáticamente. Permite desarrollar aplicaciones web en J2EE con el menor número de pasos redundantes.

- Jmatter, framework de código abierto que construye aplicaciones Swing de gestión, basándose en clases de modelo e Hibernate para la persistencia. La interfaz de usuario que genera es 100 % Swing y la distribuye mediante Java Web Start. Su principal objetivo es la creación de aplicaciones empresariales y cuenta con elementos de infraestructura listos para usarse como seguridad. Este framework es una implementación del patrón NakedObject y maneja un estándar de este patrón, con el cual mientras se escriben las clases se va desarrollando automáticamente la interfaz.

Proponiendo como framework, de los que generan automáticamente interfaces gráficas, a partir de un modelo en la plataforma Java el OpenXava por las características y facilidades que este ofrece.

# Capítulo 2

# Presentación de la solución propuesta

## 2.1. Introducción

En el presente capítulo se realiza una descripción detallada del framework OpenXava, propuesto para realizar el caso de estudio.

OpenXava es un framework para desarrollar aplicaciones JavaEE/J2EE de manera rápida y fácil. Sigue la filosofía de definir con anotaciones de Java o con XML y programar con Java. El objetivo principal de este framework es lograr que las cosas típicas en una aplicación de gestión sean fáciles de realizar, ofreciendo la flexibilidad suficiente para desarrollar las funciones avanzadas y específicas. OpenXava fue creado para hacer aplicaciones de gestión.

La esencia de OpenXava es que el desarrollador defina en vez de programar, y el framework provee automáticamente la interfaz de usuario, el acceso a los datos, el comportamiento por defecto, etc. Aunque el desarrollador tiene la posibilidad de programar manualmente cualquier parte de la aplicación.

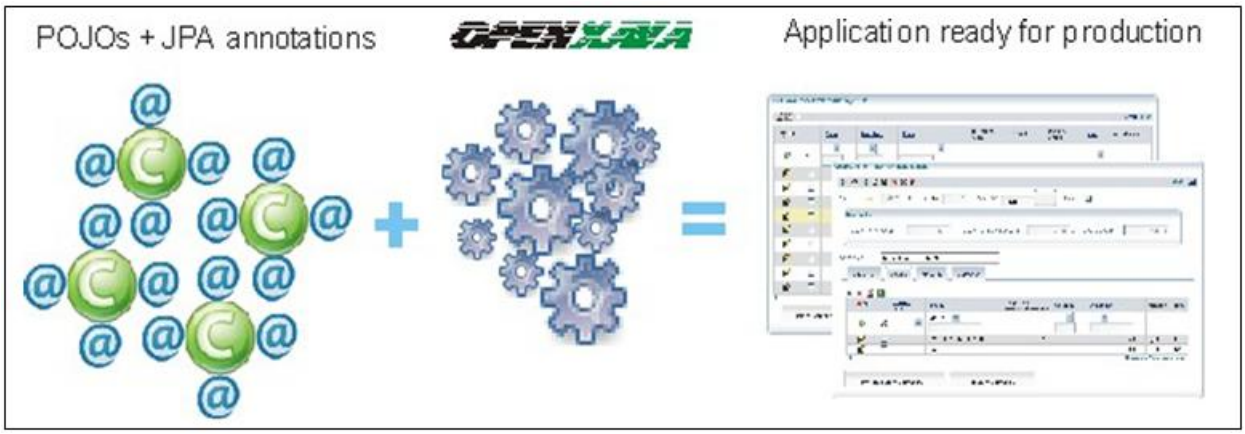



Figura 2.1: OpenXava

Este framework genera automáticamente aplicaciones de gestión solo se necesita escribir el modelo POJOs con anotaciones Java. No es necesario escribir la vista (JSP, JSF (JavaServer Faces), etc), y el controlador (para mantenimiento, generación de informes, etc) normalmente se reutiliza. OpenXava no usa UML [18].

Es decir, solo se necesita escribir una clase como esta:

```
package org.openxava.escuela.modelo;
import javax.persistence.*;
import org.openxava.annotations.*;

@Entity
public class Profesor
 {
  @Id @Column (length=5) @Required
     private String codigo;
   @Column (length=40) @Required
     private String nombre;

  public String getCodigo() { return codigo;}
  public void setCodigo(String codigo) { this.codigo = codigo;}
  public String getNombre() {return nombre;}
 public void setNombre (String nombre) {this.nombre = nombre;}

}
```

Figura 2.2: Entidad de OpenXava

Y se obtendrá una aplicación para el mantenimiento, generación de listados PDF (Formato de

almacenamiento de documentos), exportación a Excel, búsqueda, ordenación, validaciones, etc. Y para todo esto solo se necesita escribir una simple clase de Java.

## 2.2. Características

- Insertar funcionalidades en cualquier punto del desarrollo de la aplicación.
- Generar una aplicación J2EE completa, incluyendo la interfaz de usuario (AJAX).
- Soportar JSR-168: todos los módulos OpenXava también son *portlets* estándar.
- Integrar fácilmente los informes hechos con JasperReports (Excel y PDF).
- Desarrollar aplicaciones comerciales.
- Aunque la interfaz de usuario es generada automáticamente es posible hacer cambios en la presentación.

El alcance de OpenXava es limitado a un ámbito determinado, solamente a las aplicaciones de gestión. OpenXava es un entorno avanzado de programación para desarrollar aplicaciones web J2EE/JavaEE, sobre todo las relativas a negocios y empresas. La utilización de los estándares J2EE y JavaEE permite que las aplicaciones web funcionen en cualquier servidor Java.

## 2.3. Modelo

La capa del modelo en una aplicación orientada a objetos es la que contiene la lógica de negocio, la cual es la estructura de los datos con los que se trabaja y todos los cálculos, validaciones y procesos asociados a esos datos. OpenXava es un framework orientado al modelo, donde el modelo es lo más importante. La forma de definir el modelo en OpenXava es mediante simples clases Java. OpenXava provee una aplicación completamente funcional a partir de la definición del modelo.

## 2.4. Componentes que utiliza

Un componente de negocio incluye todos los artefactos de software necesarios para definir un concepto de negocio. OpenXava está basado en el concepto de componentes de negocio que no es más que

una clase Java (aunque existe también una versión XML) que contiene toda la información necesaria sobre un concepto de negocio para poder crear aplicaciones a partir del negocio.

En un componente de negocio se define:

- Modelo: la estructura de datos, las validaciones, cálculos y en general toda la lógica de negocio asociada a ese concepto.
- Vista: las posibles vistas, esto es, la configuración de todos las posibles interfaces gráficas para este componente.
- Datos tabulares: se define las posibilidades para la presentación tabular de los datos. Esto se usa para el modo lista (consultar y navegar por los datos), los listados, exportación a Excel, etc.
- Mapeo objeto-relacional: lo que incluye información sobre las tablas de la base de datos y la forma de convertir a objetos la información que en ellas hay.

**Modelo:** la capa del modelo en una aplicación orientada a objetos es la que contiene la lógica de negocio, esto es la estructura de los datos con los que se trabaja y todos los cálculos, validaciones y procesos asociados a esos datos.

**Vista:** OpenXava genera a partir del modelo una interfaz gráfica de usuario por defecto. Para muchos casos sencillos esto es suficiente, pero muchas veces es necesario modelar con más precisión la forma de la interfaz de usuario o vista.

**Datos tabulares:** son aquellos que se visualizan en formato de tabla. Cuando se crea un módulo de OpenXava convencional el usuario puede gestionar la información sobre ese componente con una lista.

**Mapeo objeto-relacional:** se declara en que tablas y columnas de la base de datos relacional se guarda la información de la entidad. Las herramientas O/R (Objeto-Relacional) permiten trabajar con objetos, en vez de con tablas, columnas y generan automáticamente el código SQL (Lenguaje de consulta estructurado) necesario para leer y actualizar la base de datos. De esta forma no se necesita acceder directamente a la base de datos con SQL, para eso se tiene que definir con precisión como se mapean las clases a las tablas, y eso es lo que se hace en las anotaciones de mapeo JPA.

Los componentes de negocio no definen lo que un usuario puede hacer con la aplicación; esto se define con los controladores que son un conjunto de acciones, una acción es un botón o vínculo que

el usuario puede pulsar. Los controladores están separados de los componentes de negocio porque un mismo controlador puede ser asignado a diferentes componentes de negocio.

## 2.5. Escalabilidad

OpenXava genera una aplicación Java EE/J2EE estándar, por tanto es tan escalable como Java EE/J2EE. OpenXava usa EJB3 JPA[38] los cuales normalizan el uso de la persistencia en la plataforma Java proporcionando un mecanismo estándar para el mapeo, EntityManager una API (Application Programming Interface) para realizar operaciones CRUD y una manera de extender la EJB-QL (Enterprise JavaBeans Query Language) para recuperar entidades, Hibernate para persistencia, y algo de estado con alcance de sesión a nivel de servlets.

## 2.6. Portales Java

Cualquier portal compatible con JSR-168. Aunque los siguientes están probados:

- WebSphere Portal 5.1 funciona hasta OX3.0.3.
- WebSphere Portal 6.0.
- Apache JetSpeed 2.
- Liferay 4.1, 4.2, 4.3, 5.0 y 5.1.

## 2.7. Tecnologías que utiliza

Una de las tecnologías que usa OpenXava es AJAX la cual es la unión de cuatro tecnologías: HTML, DOM, objeto XML- HTTP Request y XML.

## 2.8. JDK que utiliza

OpenXava funciona con JDK (Java Development Kit) 1.4.2, JDK 5.0 y JDK 6.0. Si se desea usar EJB3 JPA se necesita Java 5.

## 2.9. Concurrencia y propiedad versión en OpenXava

OpenXava usa un esquema de concurrencia optimista. Usando concurrencia optimista los registros no se bloquean permitiendo un alto nivel de concurrencia sin perder la integridad de la información. Por ejemplo, si un usuario A lee un registro y entonces un usuario B lee el mismo registro, lo modifica y graba los cambios, cuando el usuario A intente grabar el registro recibirá un error y tendrá que refrescar los datos y reintentar su modificación. Para activar el soporte de concurrencia para un componente OpenXava solo se necesita declarar una propiedad @Version, de esta forma:

@Version

private int version;

Esta propiedad es para uso del mecanismo de persistencia (Hibernate o JPA), la aplicación ni los usuarios deben acceder directamente a ella.

## 2.10. OpenXava soporta Enum

OpenXava soporta *enums* de Java 5. Un *enum* permite definir una propiedad que solo puede contener los valores indicados.

*Ejemplo:*

```
private Distancia distancia;

public enum Distancia { LOCAL,
NACIONAL, INTERNACIONAL

};
```

Figura 2.3: Ejemplo de Enum

La propiedad distancia solo puede valer LOCAL, NACIONAL o INTERNACIONAL, y como no se ha puesto @Required también permite valor vacío (null).

## 2.11. OpenXava soporta Hibernate Validator

OpenXava tiene soporte completo de Hibernate Validator, el cual pretende resolver el problema de encontrar las validaciones que se repiten en diferentes formularios de la aplicación que tratan con los mismos objetos de negocio, de forma que se defina una única vez las validaciones en los objetos de negocio, y se invoquen esas validaciones, y si se quiere llamar al negocio desde otro sitio que no sea la aplicación web, se tendría que repetir estas validaciones en el servicio web.

Principales características de Hibernate Validator:

- Se definen las validaciones fácilmente con anotaciones.
- Trae un conjunto predefinido de validaciones típicas, conjunto que podemos extender con nuestras propias validaciones.
- El sistema soporta internacionalización: ya trae mensajes de error traducidos a diez idiomas. Estos mensajes se pueden cambiar fácilmente, simplemente escribiendo un propio fichero de propiedades y se sobrescriben los mensajes que se deseen.
- Se integra directamente con Hibernate, y en general con cualquier Mapeo Objeto Relacional, de forma que antes de hacer una inserción o actualización se validan los objetos.
- Si se usa Hibernate, las validaciones que se indiquen se tendrán en cuenta a la hora de generar el DDL (los scripts de creación de la base de datos).

Se puede definir restricciones en las entidades, y OpenXava las reconocerá, mostrando los mensajes de error correspondientes al usuario. Además, las anotaciones de OpenXava @Required, @PropertyValidator y @EntityValidator están definidas como restricciones de Hibernate Validator, esto significa que cuando se grabe una entidad usando directamente JPA estas validaciones se aplicarán.

## 2.12. Por qué usar OpenXava

Existen frameworks y tecnologías las cuales se utilizan en el desarrollo de aplicaciones como Spring el cual es un framework de código abierto para el desarrollo de aplicaciones en la plataforma Java, por su diseño ofrece libertad a los desarrolladores en Java y soluciones documentadas y fáciles de usar para las prácticas comunes. Hibernate es una herramienta de Mapeo objeto-relacional para la

plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, JSF es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones JavaEE, y EJB3 es la tecnología del lado del servidor para la arquitectura de componentes en la plataforma Java Enterprise Edition (Java EE). Pero si se necesita desarrollar aplicaciones de gestión la cantidad de código a escribir sigue siendo grande, y la productividad es baja comparada con otras tecnologías (Visual Basic, Delphi, etc.). OpenXava simplemente tiene un nivel de abstracción superior para obtener así una forma simple y fácil de desarrollar aplicaciones de gestión con J2EE y con estilo Orientado a Objetos.

## 2.13. Actualidad

Actualmente OpenXava genera código para las siguientes cuatro alternativas:

1. Clases de Java convencionales para el modelo usando Hibernate para la persistencia.
2. Clases de Java convencionales para el modelo usando EJB3 JPA para la persistencia.
3. EntityBeans de EJB2 para el modelo y la persistencia.
4. POJOs + Hibernate dentro de un contenedor EJB.

## 2.14. Entidades en OpenXava

Para definir la parte del modelo se define la clase Java con anotaciones. Además de sus propias anotaciones, OpenXava soporta anotaciones de JPA e Hibernate Validator. Esta clase Java es una entidad, es decir, una clase persistente que representa concepto de negocio.

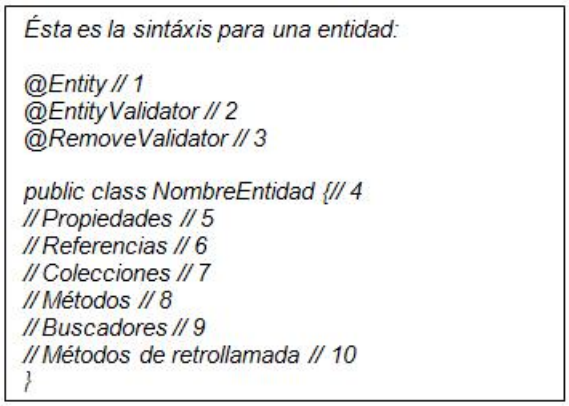



Figura 2.4: Sintáxis de una entidad en OpenXava.

1. @Entity (JPA, uno, obligado): indica que esta clase es una entidad JPA.
2. @EntityValidator (OX, varios, opcional): ejecuta una validación a nivel de modelo. Este validador puede recibir el valor de varias propiedades del modelo.
3. @RemoveValidator (OX, varios, opcional): se ejecuta antes de borrar, y tiene la posibilidad de prohibir el borrado del objeto.
4. Declaración de la clase: como en una clase de Java convencional. Podemos usar extends e implements.
5. Propiedades: propiedades de Java convencionales. Representan el estado principal del objeto.
6. Referencias: referencias a otras entidades.
7. Colecciones: colecciones de referencias a otras entidades.
8. Métodos: métodos Java con lógica de negocio.
9. Buscadores: los buscadores son métodos estáticos que hacen búsquedas usando las prestaciones de consulta de JPA.
10. Métodos de retrollamada: los métodos JPA de retrollamada (callbacks) para insertar lógica al crear, modificar, cargar, borrar, etc.

Para definir una clase como una entidad lo único que se necesita es usar la anotación @Entity en la declaración de la clase. Dentro de la entidad están definidas un conjunto de propiedades, las cuales son las siguientes:

```
@Id // 1
@Column (length=3) // 2
@Required // 3

private int codigo; // 4
private int getCodigo () {// 4
return codigo;
}
private void setCodigo (int codigo) {// 4
this.codigo = codigo;
}
```

Figura 2.5: Propiedades de la entidad.

1. @Id: indica si esta propiedad forma parte de la clave. La clave identifica a cada objeto de forma única y normalmente coincide con la clave en la tabla de base de datos.

2. @Column(length= ): longitud de los datos visualizados. Es opcional, pero suele ser útil para hacer mejores interfaces gráficas y generar las tablas de la base de datos.

3. @Required: indica si hay que validar la existencia de información en esta propiedad antes de crear o modificar.

4. La propiedad definida de la forma usual para una clase Java. Todo tipo válido para una propiedad Java se puede poner, lo que incluye tipos integrados, clases del JDK, clases propias, etc.

5. @Stereotype (OX, opcional): Permite especificar un comportamiento especial para ciertas propiedades.


Figura 2.6: Estereotipos en OpenXava.

## 2.15. Métodos en OpenXava

Los métodos se definen en una entidad JPA como una clase de Java convencional. Por ejemplo:

```
public void incrementarPrecio()
{
setPrecioUnitario(getPrecioUnitario().multiply(
new BigDecimal("1.02")).setScale(2));
}
```

Figura 2.7: Ejemplo de método en OpenXava.

Los métodos son el fundamento de los objetos. Cuando sea posible es mejor poner la lógica de

negocio en los métodos (capa del modelo) que en las acciones (capa del controlador).

## 2.16. Pruebas en OpenXava

Cuando se implementa un software, resulta recomendable comprobar que el código que se ha escrito funcione correctamente. Para ello se implementan pruebas que verifiquen que el programa funcione correctamente al generar los resultados que se esperan. OpenXava usa un sistema de pruebas basado en JUnit y HtmlUnit. Las pruebas JUnit de OpenXava emulan el funcionamiento de un usuario real con un navegador, de esta forma podemos replicar de forma exacta las mismas pruebas que se harían con un navegador. La ventaja de este enfoque es que se prueba de forma sencilla desde la interfaz gráfica al acceso a la base de datos. Mientras que las pruebas HtmlUnit son una extensión de JUnit para la realización de pruebas unitarias web. Su funcionamiento se basa en obtener los diferentes elementos que componen la página web devuelta por el servidor, e interactuar sobre ellos.

## 2.17. Vistas

Con el uso de OpenXava se tiene la posibilidad de modificar la apariencia u otros aspectos de la pantalla que maneja el usuario usando anotaciones Java las cuales son una forma de añadir metadatos al código fuente Java que están disponibles para la aplicación en tiempo de ejecución, mediante estas se pueden indicar como se quiere que salgan distribuidos los datos, se pueden poner dentro de marcos, pestañas, alinear como una tabla unos a continuación de los otros, se puede poner acciones (vínculos, imágenes o botones) dentro de la vista, asociar eventos a cada valor, en conclusión, todo lo que se pueda necesitar para una aplicación de gestión. Hay aplicaciones complejas ( gestión de Inventario, Personal o Nóminas) sin una sola vista personalizada, sino todas generadas por OpenXava.

La interfaz de usuario generada por OpenXava es buena para la mayoría de los casos, pero a veces puede que se necesite personalizar alguna parte de la interfaz de usuario, para esto se necesita crear editores para indicar como visualizar una propiedad, los cuales consisten en una definición XML junto con un fragmento de código JSP.

OpenXava no genera código y las vistas son generadas dinámicamente. Esto permite, modificar el código en el Eclipse, darle al botón de recargar de nuestro navegador, y ver los cambios. OpenXava genera a partir del modelo una interfaz gráfica de usuario por defecto. Muchas veces se necesita modelar con más precisión la forma de la interfaz de usuario o vista.

La anotación @View se puede usar en una entidad o una clase incrustable para definir la disposición de sus miembros en la interfaz de usuario.

La sintaxis para definir una vista (@View) es:

```
@View (name="nombre", // 1
members="miembros" // 2
)
public class MiEntidad {
```

Figura 2.8: Sintaxis para definir una vista.

1. name (opcional): el nombre identifica a la vista, y puede ser usado desde otros lugares de OpenXava (por ejemplo desde aplicacion.xml) o desde otra entidad.

2. members (opcional): indica los miembros que tienen que salir y como tienen que estar dispuestos en la interfaz gráfica. Por defecto visualiza todos los miembros no ocultos en el orden en que están declarados en el modelo.

La vista por defecto que genera OpenXava es la siguiente:

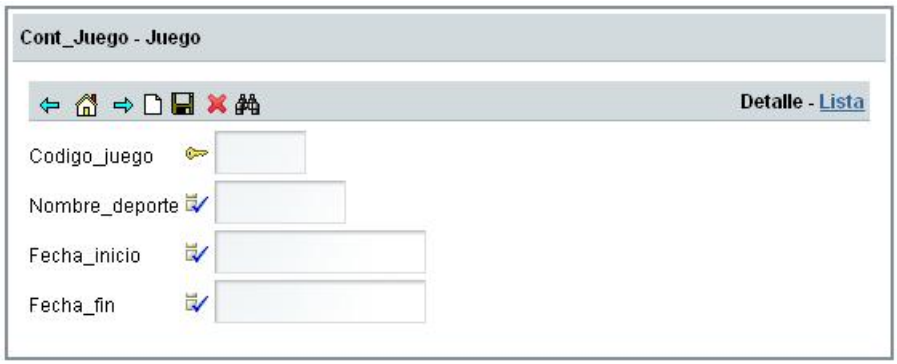



Figura 2.9: Vista por defecto.

Se pueden escoger los miembros que se quieren que aparezcan y en que orden en la vista, con el atributo members, de la siguiente manera:

@View(members="codigo_juego, nombre_deporte;" +"fecha_inicio,fecha_fin")

Se visualizaría de esta forma:


Figura 2.10: Vista generada en el orden de los miembros.

Se puede observar como se separan los nombres de los miembros con comas, punto y comas, esto sirve para indicar la disposición, con la coma el miembro se pone a continuación, y con punto y coma en la línea siguiente.

### 2.17.1. Grupos

Con los grupos podemos agrupar un conjunto de propiedades relacionadas, y esto tiene un efecto visual. Para definir un grupo solo se necesita poner el nombre del grupo y después sus miembros entre corchetes.

De esta forma:

@View(members= "Datos_Generales[ codigo_juego, nombre_deporte];" + "fecha_inicio,fecha_fin" )

En este caso el resultado sería:


Figura 2.11: Vista de grupos.

Se pueden poner varios grupos en una vista de la siguiente forma:

@View(members= "Datos_Generales [" + " codigo_juego;" + " nombre_deporte;" + "]"
+ "Otros_Datos[" + " fecha_inicio;" + " fecha_fin;" + "]" )

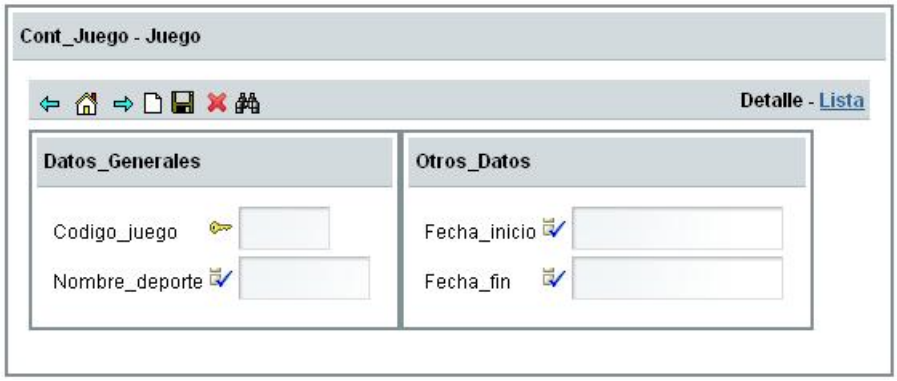



Figura 2.12: Vista de varios grupos.

Si se quiere que aparezca un grupo debajo del otro se debe poner un punto y coma después del grupo.

@View(members= "Datos_Generales [" + " codigo_juego;" + " nombre_deporte;" + "];"
+ "Otros_Datos ["+ " fecha_inicio;" + " fecha_fin;" + "]" )

En este caso el resultado sería:

OpenXava permite anidar grupos. Esta interesante característica permite disponer los elementos de la interfaz gráfica de una forma simple y flexible. Por ejemplo, si definimos una vista como ésta:

@View(members="Datos_Genarales["+"codigo_juego,"+"nombre_deporte;"+"Otros_Datos
["+"fecha_inicio;"+"]"+"Datos_Adicionales["+"fecha_fin;"+"]]")

En este caso el resultado sería:

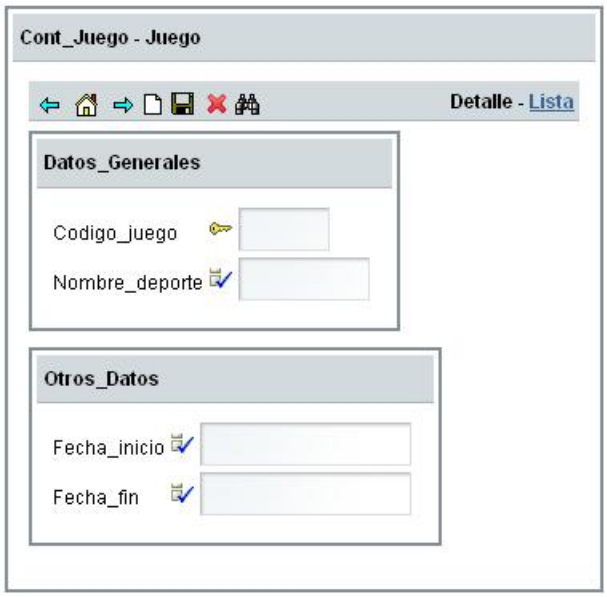



Figura 2.13: Vista para poner un grupo debajo de otro.


Figura 2.14: Vista con grupos anidados.

La información se puede visualizar alineada por columnas, definiendo el grupo de esta forma (se usa [# en vez de [, esto facilita que los miembros salgan alineados):

@View(members= "codigo_juego, nombre_deporte;" + "fecha [#" + "fecha_inicio, fecha_fin;" + "]" )

En este caso el resultado sería:


Figura 2.15: Vista alineada por columnas.

### 2.17.2. Secciones

Además de grupos los miembros se pueden organizar en secciones. Para definir una sección solo se necesita poner el nombre de la sección y después sus miembros entre llaves. Por ejemplo:

@View(members= "codigo_juego;" + "fecha_inicio,fecha_fin;" + "prueba {nombre_deporte }" )

El resultado visual sería:


Figura 2.16: Vista con sección.

### 2.17.3. Clase transitoria: Solo para crear vistas en OpenXava

En OpenXava no se puede tener vistas que no estén asociadas a un modelo. Una clase transitoria no está asociada a ninguna tabla de la base de datos, normalmente se usa solo para visualizar interfaces de usuario no relacionadas con ninguna tabla de la base de datos.

## 2.18. Colecciones en OpenXava

En OpenXava se pueden definir colecciones de referencias a entidades, donde una colección es una propiedad Java que devuelve java.util.Collection. Una declaración de colección convencional de Java tiene sus get y set. La colección se marca con @OneToMany (JPA) o @ManyToMany (JPA) y el tipo ha de ser otra entidad.

La anotación @ManyToMany (JPA) permite definir una colección con una multiplicidad de muchos-a-muchos. Como sigue:

```
@ManyToMany(mappedBy="equipo",cascade=CascadeType.REMOVE)
private Collection<Estudiante> estudiante;

  public void setEstudiante(Collection<Estudiante> estudiante) {
      this.estudiante = estudiante;
  }

  public Collection<Estudiante> getEstudiante() {
      return estudiante;
  }
```

Figura 2.17: Colecciones con multiplicidad de muchos-a-muchos

La anotación @OneToMany (JPA) permite definir una colección con una multiplicidad de uno-a-muchos. Como sigue:

```
@OneToMany(mappedBy="equipo",cascade=CascadeType.REMOVE)
private Collection<Estudiante> estudiante;

  public void setEstudiante(Collection<Estudiante> estudiante) {
      this.estudiante = estudiante;
  }

  public Collection<Estudiante> getEstudiante() {
      return estudiante;
  }
```

Figura 2.18: Colecciones con multiplicidad de uno-a-muchos.

## 2.19. Datos tabulares en OpenXava

Los datos tabulares son aquellos que se visualizan en formato de tabla. Cuando se crea un módulo de OpenXava convencional, el usuario puede gestionar la información sobre ese componente con una lista como ésta:


Figura 2.19: Interfaz gráfica principal.

Esta lista permite al usuario:

- Filtrar por cualquier columna o combinación de ellas.
- Ordenar por cualquier columna con un simple click.

- Visualizar los datos paginados y poder leer eficientemente tablas de millones de registros.
- Personalizar la lista: añadir, quitar y cambiar de orden las columnas.
- Acciones genéricas para procesar la lista: como la de generar un informe en PDF, exportar a Excel o borrar los registros seleccionados.

Cuando se exporta un Excel, los datos existentes en la lista se visualizan de esta forma:


Figura 2.20: Vista del Excel.

Y cuando se exporta un PDF, los datos existentes en la lista se visualizan de esta forma:

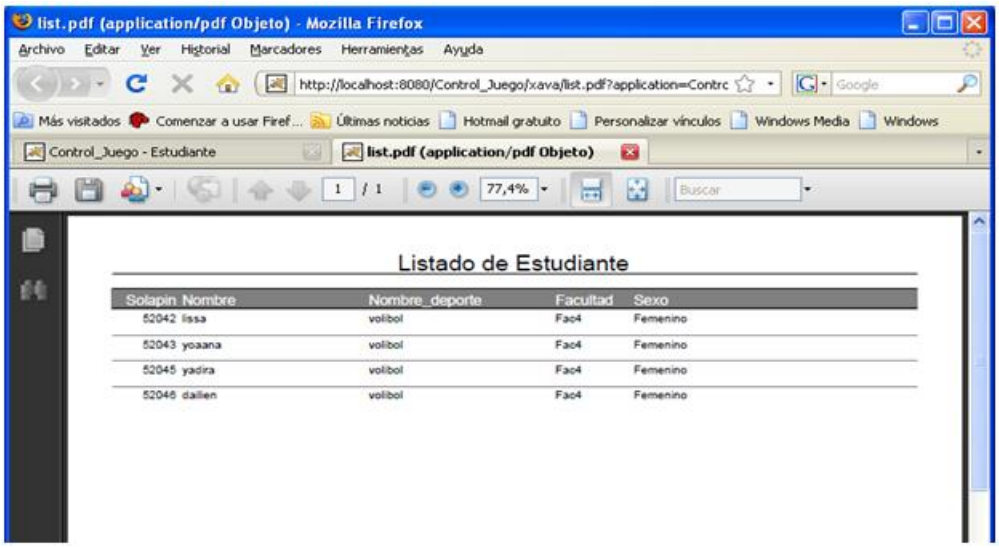



Figura 2.21: Vista del PDF.

La lista por defecto el usuario puede personalizarla. Si se necesita modificar el comportamiento de la lista, esto se hace mediante la anotación @Tab dentro de la definición de la entidad.

```
@Tab(
name="nombre", // 1
filter=clase del filtro, // 2
rowStyles=array de @RowStyle, // 3
properties="propiedades", // 4
baseCondition="condición base", // 5
defaultOrder="orden por defecto" // 6
)
public class MyEntity
{
```

Figura 2.22: Sintaxis de @Tab.

1. name: se pueden definir varios tabs para una entidad (mediante la anotación @Tabs), y ponerle un nombre a cada uno. Este nombre se usará después para indicar que tab queremos usar (normalmente en aplicación.xml al definir un módulo).

2. filter: permite definir un filtro a realizar sobre los valores que introduce el usuario cuando quiere filtrar.

3. rowStyles: una forma sencilla de especificar un estilo de visualización diferente para ciertas filas. Normalmente para resaltar filas que cumplen cierta condición. Especificamos un array de @RowStyle.

4. properties: la lista de propiedades a visualizar inicialmente.

5. baseCondition: es una condición que se aplicará siempre a los datos visualizados, añadiéndose a las que pueda poner el usuario.

6. defaultOrder: para especificar el orden en que aparecen los datos en la lista inicialmente.

## 2.20. Mapeo objeto/relacional en OpenXava

Con el mapeo objeto relacional se declara en que tablas y columnas de la base de datos relacional se guarda la información de la entidad. Las entidades OpenXava son entidades JPA, por lo tanto el mapeo objeto/relacional en OpenXava se hace mediante Java Persistence API (JPA).

### 2.20.1. Mapeo de entidad

La anotación @Table especifica la tabla principal para la entidad. Se pueden especificar tablas adicionales usando @SecondaryTable o @SecondaryTables. Si no se especifica @Table para una entidad se aplicarán los valores por defecto.

```
@Entity
@Table(name="CLI",
schema="XAVATEST")
public class Cliente {
```

Figura 2.23: Ejemplo de Mapeo de Entidad

### 2.20.2. Mapeo propiedad

La anotación @Column se usa para especificar como mapear una propiedad persistente. Si no se especifica @Column se aplican los valores por defecto.

```
@Column(name="DESC", length=512)
private String descripción.
```

Figura 2.24: Ejemplo de mapeo propiedad.

### 2.20.3. Mapeo de referencia

La anotación @JoinColumn se usa para especificar el mapeo de una columna para una referencia.

```
@ManyToOne
@JoinColumn(name="CLI_ID")
private Cliente cliente;
```

Figura 2.25: Ejemplo de mapeo de referencia.

Si se necesita definir un mapeo para una clave foránea compuesta se usa @JoinColumns. Esta anotación agrupa anotaciones @JoinColumn para la misma referencia.

### 2.20.4. Mapeo de colección

Cuando se usa @OneToMany para una colección, el mapeo depende de la referencia usada en la otra parte de la asociación; pero si se usa @ManyToMany, es ventajoso declarar la tabla de unión mediante @JoinTable( si este se omite se aplican los valores por defecto) como sigue:

```
@ManyToMany
@JoinTable(name="CLIENTE_PROVINCIA",
joinColumns=@JoinColumn(name="CLIENTE"),
inverseJoinColumns=@JoinColumn(name="PROVINCIA")
)
private Collection<Provincia> provincias;
```

Figura 2.26: Ejemplo de mapeo de colección.

### 2.20.5. Mapeo de referencia incrustada

Una referencia incrustada contiene información que en el modelo relacional se guarda en la misma tabla que la entidad principal. Por ejemplo si se tiene un incrustable "Equipo asociado a un Estudiante", los datos del "Equipo" se guardan en la misma tabla que los del "Estudiante". Esto se expresa con JPA, usando la anotación @AttributeOverrides(si no se usa @AttributeOverrides se asumen los valores por defectos), de esta forma:

```
@Embedded
@AttributeOverrides({
@AttributeOverride(name="calle",
column=@Column("DIR_CALLE")),
@AttributeOverride(name="codigoPostal",
column=@Column("DIR_CP"))
@AttributeOverride(name="poblacion",
column=@Column("DIR_POB")),
@AttributeOverride(name="pais",
column=@Column("DIR_PAIS"))
})
private Direccion direccion;
```

Figura 2.27: Ejemplo de mapeo de referencia incrustada.

## 2.21. Conclusiones

En este capítulo se presentó la propuesta del framework OpenXava para elaborar el caso de estudio. Donde se analizaron los componentes, escalabilidad, concurrencia, Mapeo Objeto Relacional y sus principales características, entre otros. Dentro de las principales características de este framework se encuentran:

- La generación automática de interfaces gráficas a partir de un modelo en la plataforma Java, logrando así que los diseñadores y los programadores no tengan que realizar todo el largo proceso que requieren estas interfaces para su desarrollo.
- Menos codificación, ya que solamente se necesita escribir el modelo, los controladores se reusan y las vistas se generan automáticamente (aunque estas vistas pueden ser modificadas en la parte del modelo).
- Además integrar los informes hechos con JasperReports (Excel y PDF), así como filtrar y buscar datos según algún criterio.

# Capítulo 3

# Caso de Estudio

## 3.1. Introducción

En el presente capítulo se realiza una descripción detallada del caso de estudio sobre el framework OpenXava propuesto en el capítulo anterior. Además de la descripción de los pasos a seguir para trabajar con este framework.

## 3.2. Caso estudio

La Universidad de las Ciencias Informáticas necesita una aplicación para llevar el control de los Juegos Deportivos a nivel de facultad, el cual es uno de los eventos que más repercusión tiene en la misma. De los estudiantes que son deportistas se conoce el número de solapín, nombre, facultad, deporte al que pertenece y el sexo. De los equipos se conoce el código del equipo, los estudiantes que lo integran y la puntuación después de concluido cada juego. De los juegos se conoce fecha en que estos inician y fecha en que terminan, el código del juego y el nombre del deporte.

La aplicación debe permitir al usuario:

- Insertar estudiantes, juegos y equipos.
- Dar baja.
- Buscar los estudiantes que se encuentran en un equipo determinado.
- Buscar horario de los juegos.

- Modificar equipos.
- Buscar los estudiantes que existen del mismo sexo.

## 3.3. Estructura de un proyecto en OpenXava

Una aplicación OpenXava es un conjunto de módulos. Un módulo une un componente de negocio con uno o más controladores. Cada módulo de la aplicación es lo que al final utiliza el usuario.

Un proyecto OpenXava contiene las siguientes carpetas:

- **[raiz]:** en la raíz del proyecto nos encontraremos el build.xml (con las tareas Ant).
- **src[carpeta fuente]:** contiene el código fuente Java.
- **xava:** los archivos XML para configurar las aplicaciones OpenXava. Los principales son aplicacion.xml y controladores.xml.
- **i18n:** archivos de recursos con las etiquetas y mensajes en varios idiomas.
- **properties[carpeta fuente]:** archivos de propiedades para configurar la aplicación.
- **data:** útil para guardar los scripts para crear las tablas de la aplicación, si estos se desarrollaran.
- **web:** contenido de la parte web (normalmente archivos JSP, lib y clases).

## 3.4. Como trabajar con OpenXava

Para poder trabajar con OpenXava se necesita tener instalado el Java 5.0 o superior, y establecer el valor de la variable JAVA_HOME al directorio de instalación de Java 5(en Windows, en Sistema > Opciones Avanzadas > Variables de Entorno). Se necesita tener instalado el Eclipse o cualquier otro IDE.

Para empezar a trabajar con OpenXava:

- Se ejecuta el eclipse y se apunta al *workspace* de OpenXava (con File > Switch Workspace).
- Después se ejecuta start-tomcat.sh/start-tomcat.bat.

### 3.4.1. Pasos a seguir para crear un nuevo proyecto en OpenXava

1. Luego de abierto el Eclipse y apuntando al *workspace* de OpenXava y arrancado el Tomcat; para crear un nuevo proyecto en OpenXava existen dos formas: clic derecho Package Explorer > New> Project o clic en File> New> Project como se muestra a continuación:

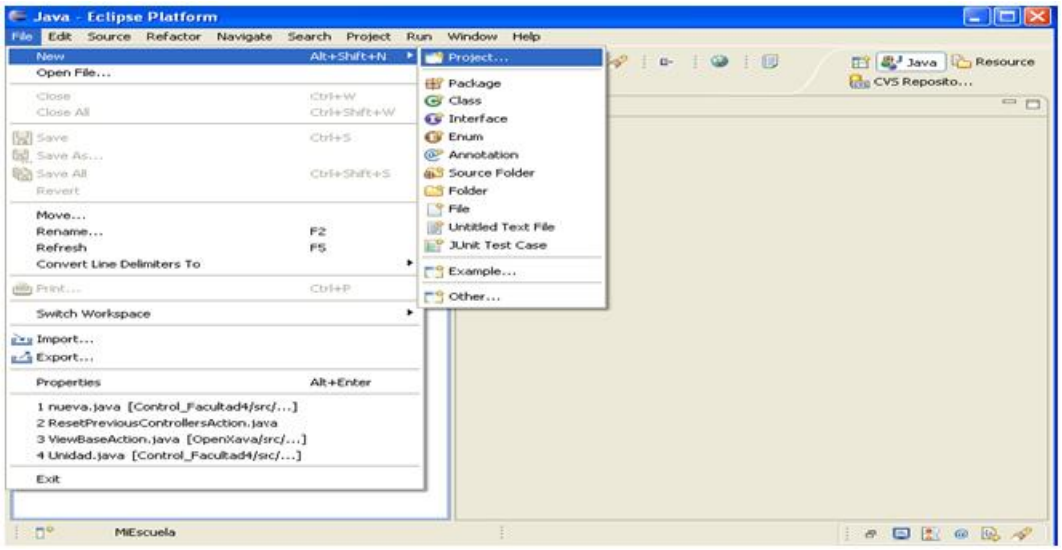



Figura 3.1: Vista para Crear un Nuevo Proyecto

2. Después de dar clic en Project saldrá el siguiente formulario, y se escoge Java>Java Project>Next.

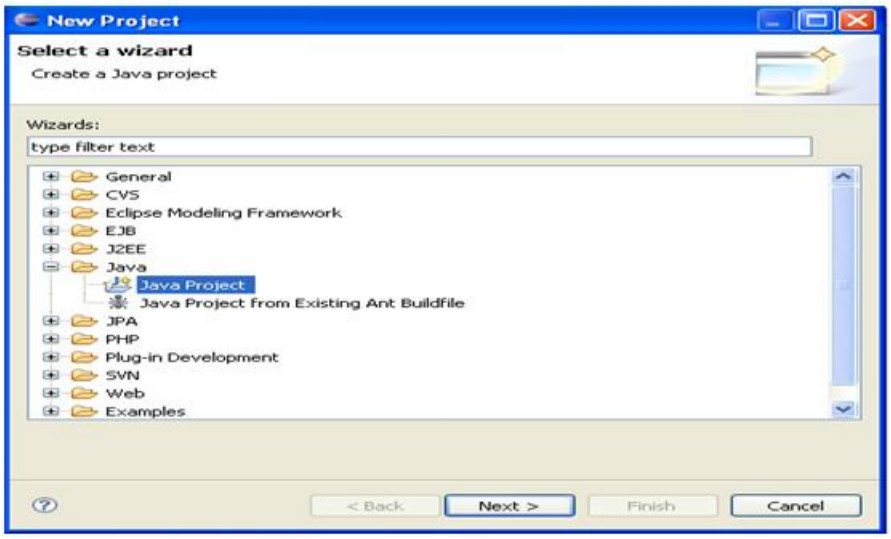



Figura 3.2: Formulario1 para Crear un Nuevo Proyecto

3. Luego se visualiza el siguiente formulario en el cual se define el nombre del nuevo proyecto, se da clic en Next>Finish ya queda creado el nuevo proyecto.

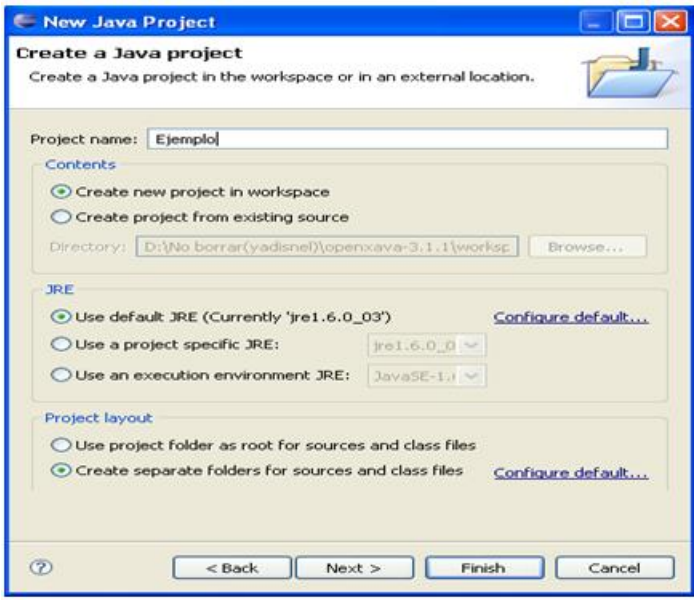



Figura 3.3: Formulario2 para Crear un Nuevo Proyecto

4. Ahora se tiene un proyecto Java vacío en el *workspace*, el siguiente paso es darle la estructura

correcta para un proyecto OpenXava. Se va al proyecto OpenXavaPlantilla y se ejecuta CrearNuevo-Proyecto.xml usando Ant o se puede hacer con Botón Derecho en CrearNuevoProyecto.xml > Run as > Ant Build. Ant nos preguntará por el nombre del proyecto y se pone el mismo nombre que se había definido anteriormente.

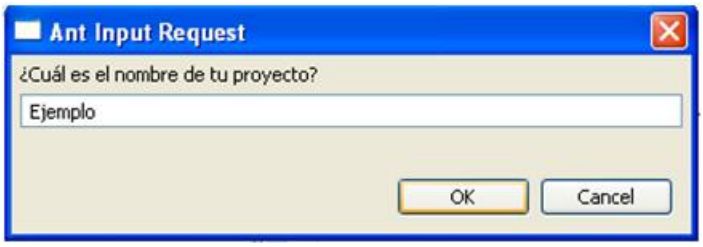



Figura 3.4: Formulario3 para Crear un Nuevo Proyecto

5. Se selecciona el proyecto y se pulsa F5 para refrescar. Con esto ya se tiene el proyecto listo para empezar a trabajar, pero se tiene que tener una base de datos configurada.

## 3.5. Configuración de la Base de Datos (OpenXava y PostgresSQL)

1.Copiar driver de PostgreSQL en tomcat\comon\lib.

2. Definir JDNI en Tomcat\conf\context.xml como sigue (donde dice Tesis se pondría el nombre de la base de datos y username y password son el usuario y la contraseña del PostgreSQL respectivamente:

```
<Resource name="jdbc/TesisDS" auth="Container" type="javax.sql.DataSource"
    maxActive="20" maxIdle="5" maxWait="10000"
    username="postgres" password="postgres "
driverClassName="org.postgresql.Driver"
    url="jdbc:postgresql://localhost:5432/Tesis"/>
```

Figura 3.5: Código para Definir JDNI

3. Localizar en build.xml del nuevo proyecto la ubicación del driver de PostgreSQL de esta forma:

```
<target name="actualizarEsquema">
 <ant antfile="../OpenXava/build.xml" target="updateSchemaJPA">
 <property name="persistence.unit" value="junit"/>
 <property name="schema.path" value="../../tomcat/common/lib/postgresql-8.3-
604.jdbc4.jar"//>
 </ant>
</target>
```

Figura 3.6: Código para la Ubicación del driver

4. Se define lo siguiente en persistence.xml:

```
<!-- Tomcat + Postgres -->
<persistence-unit name="default">
 <non-jta-data-source>java:comp/env/jdbc/TesisDS</non-jta-data-source>
  <class>org.openxava.session.GalleryImage</class>
   <properties>
    <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect" />
     </properties>
</persistence-unit>
```

Figura 3.7: Código a Definir en persistence.xml

```
<!-- JUnit Postgres-->
<persistence-unit name="junit">
 <properties>
  <property name="hibernate.connection.driver_class" value="org.postgresql.Driver" />
  <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect" />
  <property name="hibernate.connection.url" value="jdbc:postgresql://localhost:5432/Tesis" />
  <property name="hibernate.connection.username" value=" postgres " />
  <property name="hibernate.connection.password" value=" postgres " />
  <property name="hibernate.show_sql" value="true" />
</properties>
</persistence-unit>
```

Figura 3.8: Código a Definir en persistence.xml

5. Se define el dialecto en hibernate.cfg.xml de esta forma:

```
<!-- Tomcat + Hypersonic -->
<property name="hibernate.connection.datasource">java:comp/env/jdbc/ TesisDS</property>
<property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/ Tesis </property>
<property name="hibernate.connection.username"> postgres </property>
<property name="hibernate.connection.password"> postgres </property>
<property name="hibernate.show_sql">true</property>
```

Figura 3.9: Código a definir en hibernate.cfg.xml

## 3.6. Pasos para crear clases en OpenXava

1. Se da clic en la carpeta src y se usa Botón Derecho > New >Package.

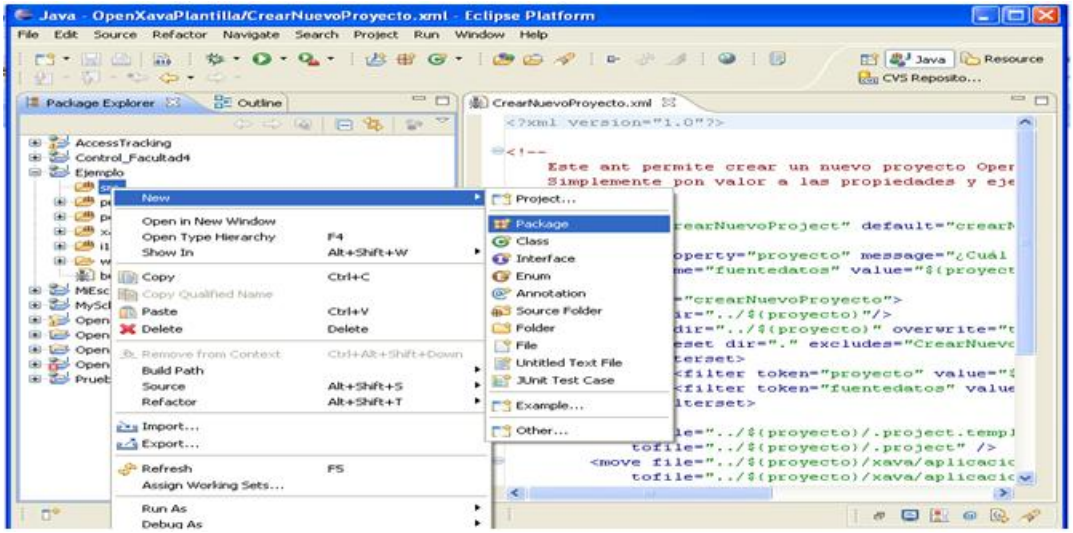



Figura 3.10: Vista para crear un paquete

2. Se crea un paquete llamado org.openxava.nombre_proyecto.modelo.

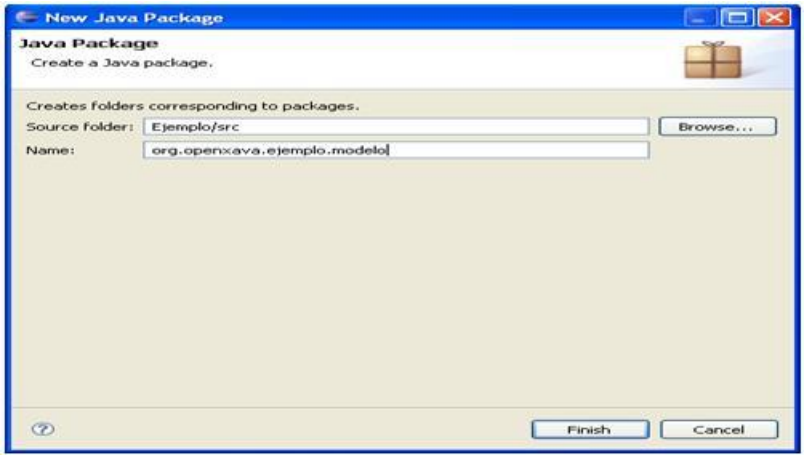



Figura 3.11: Formulario para Nombra el Paquete

3. Después clic derecho en el paquete org.openxava. nombre_proyecto.modelo y se usa New > Class.

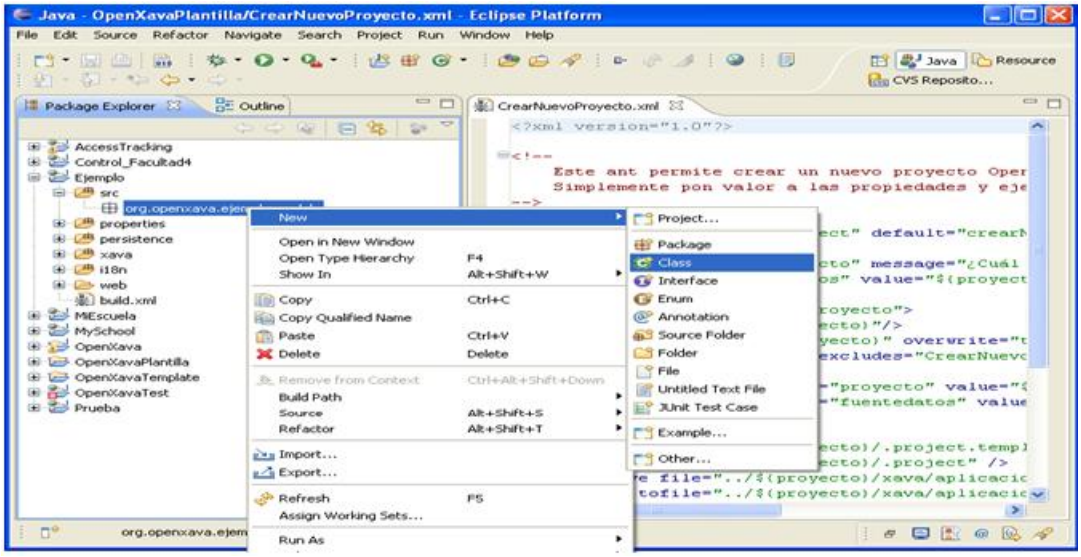



Figura 3.12: Vista para Crear una Clase

## 3.7. Estructura de las entidades del Caso de Estudio y la interfaz gráfica que generan.

### 3.7.1. Entidad Juego:

```
package org.openxava.juego.modelo;
import javax.persistence.*;
import org.openxava.annotations.*;
@View( members="Datos_Juego[" +"codigo_juego," + " fecha_inicio;" + "nombre_deporte," +
"fecha_fin;" + "]")
@Entity
public class Juego
{
    @Id @Column(length=5)@Required
    private int codigo_juego;
    @Column(length=10)@Required
    private String nombre_deporte;
```

```
    @Column(length=20)@Required
    private String fecha_inicio;
    @Column(length=20)@Required
    private String fecha_fin;
    public void setFecha_inicio(String fecha_inicio)
{
    this.fecha_inicio = fecha_inicio;
}
    public String getFecha_inicio()
{
    return fecha_inicio;
}
    public void setCodigo_juego(int codigo_juego)
{
    this.codigo_juego = codigo_juego;
}
    public int getCodigo_juego()
{
    return codigo_juego;
}
    public void setFecha_fin(String fecha_fin)
{
    this.fecha_fin = fecha_fin;
}
    public String getFecha_fin()
{
    return fecha_fin;
}
    public void setNombre_deporte(String nombre_deporte)
{
    this.nombre_deporte = nombre_deporte;
}
```

```
    public String getNombre_deporte()
  {
    return nombre_deporte;
  }
}
```

#### 3.7.1.1. De esta forma se crea un nuevo juego:

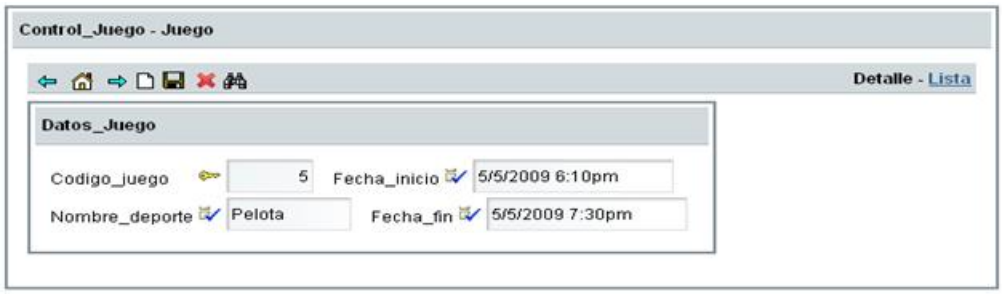



Figura 3.13: Ejemplo para un Nuevo Juego

### 3.7.2. Entidad Equipo

```
package org.openxava.juego.modelo;
import javax.persistence.*;
import org.openxava.annotations.*;
@View(members="Datos_Equipo[" + " codigo_equipo, "+" puntuacion;"+"];"+"juego;" )
@Entity
public class Equipo
{
    @Id @Column(length=5)@Required
    private int codigo_equipo;
    @Column(length=10)
    private float puntuacion;
    @ManyToOne
    private Juego juego;
    public void setCodigo_equipo(int codigo_equipo)
```

```
    {
      this.codigo_equipo = codigo_equipo;
    }
      public int getCodigo_equipo()
    {
      return codigo_equipo;
    }
      public void setPuntuacion(float puntuacion)
    {
      this.puntuacion = puntuacion;
    }
      public float getPuntuacion()
    {
      return puntuacion;
    }
      public void setJuego(Juego juego)
    {
      this.juego = juego;
    }
      public Juego getJuego()
    {
      return juego;
    }
  }
```

#### 3.7.2.1. De esta forma se crea un equipo:

- Si se quiere crear un equipo con un juego que ya existe se va a la vista de juego y se pulsa el gotero  , después se escoge el juego que se desee y quedaría de esta forma:

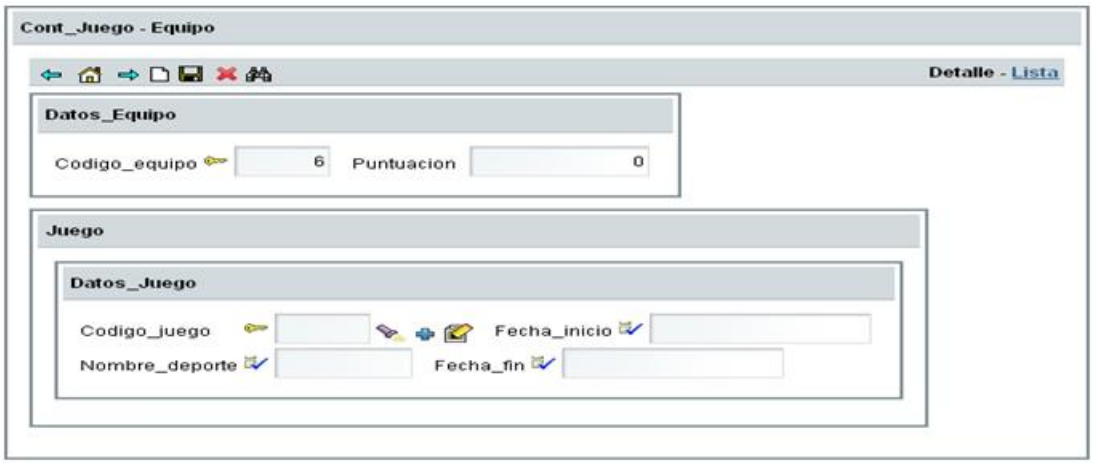



Figura 3.14: Vista para Crear un Equipo

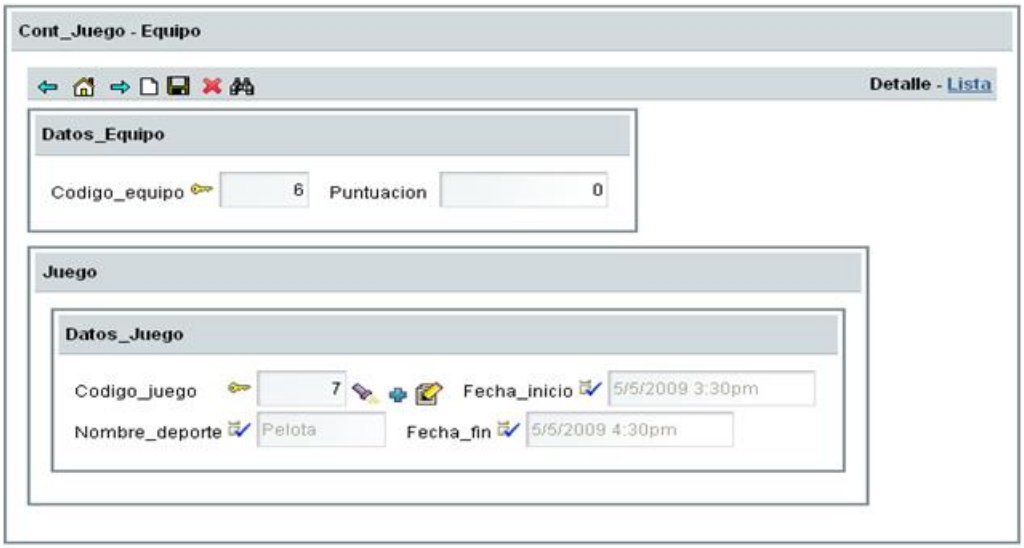



Figura 3.15: Vista para un Equipo con un Juego

- Si se quiere crear un nuevo equipo con un juego que no existe se va a la vista de juego y se pulsa el signo de más , y de esta forma se crea el nuevo juego:


Figura 3.16: Vista para crear un nuevo equipo con un nuevo juego.

Y queda creado el equipo con un nuevo juego:


Figura 3.17: Vista al crear un equipo con nuevo juego.

- Si desea hacer algún cambio en los datos del juego se pulsa en el lápiz y da la posibilidad de modificar el dato que se desee excepto el dato que se declaró como ID en la entidad.

### 3.7.3. Entidad Estudiante:

package org.openxava.juego.modelo;
import javax.persistence.*;

```
import org.openxava.annotations.*;
@View(members=Datos_Estudiante["+"solapin,"+"nombre;"+"facultad,"+"nombre_deporte;"+"sexo"
{equipo}"+ "juego{juego}")
@Entity
public class Estudiante
{
    @Id @Column(length=5)@Required
    private int solapin;
    @Column(length=20)@Required
    private String nombre;
    @Column(length=20)@Required
    private String nombre_deporte;
    @ManyToOne
    private Equipo equipo;
    @ManyToOne
    private Juego juego;
    @Required
    private Facultad facultad;
    @Required
    private Facultad facultad;
    @Required
    private Sexo sexo;
    public enum Facultad{fac1,fac2,fac3,fac4,fac5,fac6,fac7,fac8,fac9,fac10};
    public enum Sexo{Masculino, Femenino};
    public void setSolapin(int solapin)
  {
    this.solapin = solapin;
  }
    public int getSolapin()
  {
    return solapin;
  }
```

```
  public void setNombre(String nombre)
{
  this.nombre = nombre;
}
  public String getNombre()
{
  return nombre;
}
  public void setNombre_deporte(String nombre_deporte)
{
  this.nombre_deporte = nombre_deporte;
}
  public String getNombre_deporte()
{
  return nombre_deporte;
}
  public void setFacultad(Facultad facultad)
{
  this.facultad = facultad;
}
  public Facultad getFacultad()
{
  return facultad;
}
  public void setSexo(Sexo sexo)
{
  this.sexo = sexo;
}
  public Sexo getSexo()
{
  return sexo;
}
```

```
    public void setEquipo(Equipo equipo)
  {
    this.equipo = equipo;
  }
    public Equipo getEquipo()
  {
    return equipo;
  }
    public void setJuego(Juego juego)
  {
    this.juego = juego;
  }
    public Juego getJuego()
  {
    return juego;
  }
}
```

#### 3.7.3.1. Si se quiere insertar un estudiante sería de esta forma:

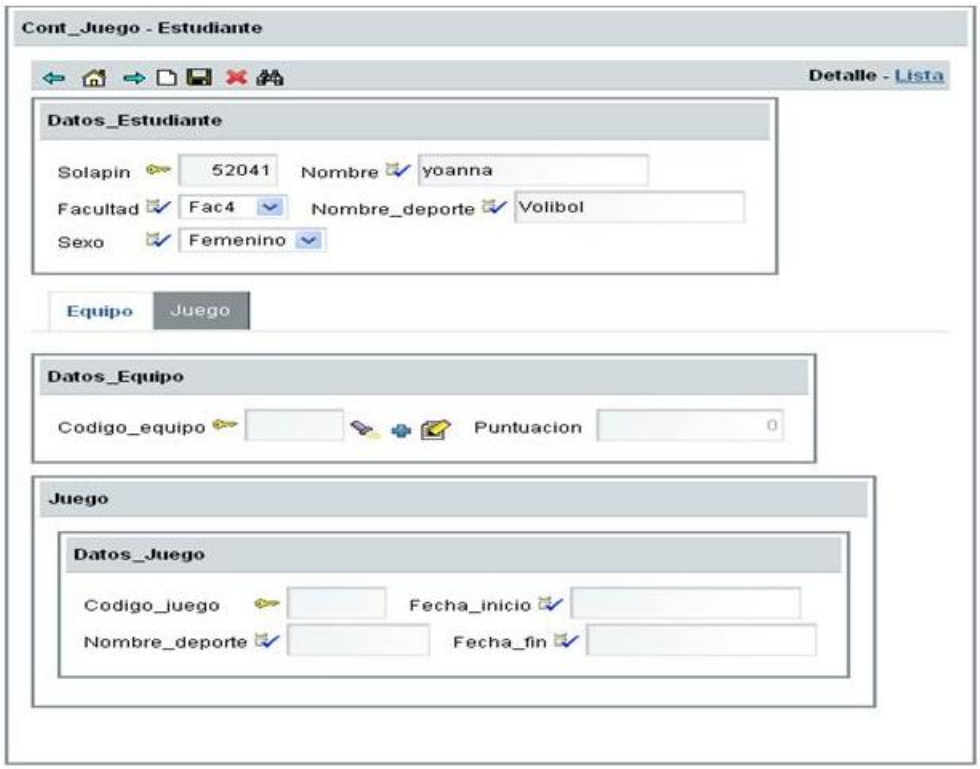



Figura 3.18: Vista para Insertar un Nuevo Estudiante

- Y si se desea crear, un nuevo equipo al igual que un juego, solo se necesita pulsar en el signo "más" de cada vista. Y si se desea que el estudiante este en un equipo que ya fue creado al igual que un juego que ya fue creado se pulsa el gotero .

## 3.8. Conclusiones

En este capítulo se presentó el caso de estudio con sus entidades y lo que estas generan en la interfaz gráfica sobre el framework OpenXava, y se analizaron los pasos a seguir para trabajar con OpenXava, alguno de estos son:

- Como crear un nuevo proyecto en OpenXava.
- Configuración de la base de datos (OpenXava y PostgreSQL).

- Como crear nuevas clases en OpenXava.

# Conclusiones

Al concluir el presente trabajo queda realizado un estudio sobre algunos de los frameworks que generan autómaticamente interfaz gráfica a partir de un modelo en la plataforma Java, a partir del cual se decidió realizar un caso de estudio sobre el framework propuesto ya que el desarrollo de interfaces gráficas, en un sistema informático de corte empresarial, se convierte en el flujo de desarrollo más lento dentro del desarrollo de un sistema, sin olvidar además la complejidad que esto lleva. Como elemento añadido a los objetivos planteados, se profundizó en los elementos de configuración del Tomcat con PostgreSQL para su utilización en la conexión de la aplicación con la base de datos. Puede concluirse que se ha alcanzado de esta manera el objetivo propuesto, dándole solución al problema planteado al iniciar el trabajo, existiendo una correspondencia entre los objetivos planteados y los resultados obtenidos.

# Recomendaciones

Luego de cumplirse los objetivos del trabajo, se recomienda:

- Continuar la investigación sobre los frameworks que genenan automáticamente interfaz gráfica a partir de un modelo en la plataforma Java.
- Aplicar el framework propuesto (OpenXava) para el desarrollo de aplicaciones en la plataforma Java en proyectos de la Universidad.

# Bibliografía


[1] Amaya Barbosa Pablo, González Contreras Carlos, Murillo Rodríguez Juan M. AspectMDA: Hacia un Desarrollo Incremental Consistente Integrando MDA y Orientación a Aspectos, 2009. [Disponible en: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-157/paper09.pdf]

[2] An Ontology-based MDA Framework for Service-based Software Systems Architecting Claus Pahl , Página Oficial, 2009. [Disponible en: http://www.mel.nist.gov/msid/conferences/SWESE/repository/14ont-basedMDAfrmwk4svc_sys.pdf]

[3] barrapunto.com, Página Oficial, 2009. [Disponible en: http://softlibre.barrapunto.com/article.pl?sid=08/12/18/1242245]

[4] BoxSOFTWARE Programas Gratis en Español,Página Oficial,2009. [Disponible en: http://www.boxsoftware.net/PROGRAMAS/OPENXAVA-V3-1-1.ASP]

[5] Bushmann, Frank, Regine Meunier, Hans Rohnert, et al. Pattern-Oriented Software Architecture: A System of Patterns. England, John Wiley and Sons, 1996.

[6] Descarga de OpenXava,Página Oficial,2009. [Disponible en: http://www.gestion400.com/web/guest/downloads]

[7] Documentación de OpenXava,Página Oficial,2009. [Disponible en: http://openxava.wikispaces.com/INDEX_ES]

[8] Dr. Emilio Insfrán, Introducción a MDA(Model Driven Architecture ), 2009. [Disponible en: http://users.dsic.upv.es/~einsfran/mda/]

[9] DZone-JavaLobby, Página Oficial, 2009.[ Disponible en:http://java.dzone.com/news/new-jmatter-release]

[10] DZone-JavaLobby, Página Oficial, 2009. [Disponible en: http://java.dzone.com/news/interview-eitan-suez-creator-j]

[11] eWEEK.com, Página Oficial, 2009. [Disponible en: http://www.eweek.com/c/a/Application-Development/OpenSource-Framework-Means-Happy-Trails-for-Java-Developers/]

[12] Geek Zone,Página Oficial,2009. [Disponible en: http://www.geekzone.com.ar/CONTENT/OPENXAVA-AJAX-PARA-APLICACIONES-EMPRESARIALES]

[13] Guía de Referencia de OpenXava,Página Oficial,2009. [Disponible en: http://www.abcdatos.com/tutoriales/tutorial/z7132.html]

[14] Home Trails , Página Oficial, 2009. [Disponible en: http://www.trailsframework.org/]

[15] Introducción a las aplicaciones de gestión,Página Oficial,2009. [Disponible en: http://sestud.uv.es/MANUAL.ESP/GESTION/GESTION0.HTM]

[16] Java en Castellano, Página Oficial, 2009. [Disponible en: http://www.programacion.com/java/noticia/1582/]

[17] JavaHispano, Página Oficial, 2009. [Disponible en: http://www.javahispano.org/contenidos/es/entrevista_a_eitan_suez_creador_de_jmatter/]

[18] JavaHispano, Página Oficial, 2009. [Disponible en: http://www.javahispano.org/contenidos/es/openxava_3_0_un_motor_de_aplicaciones_jpa/]

[19] JavaHispano, Página Oficial, 2009. [Disponible en: http://www.javahispano.org/contenidos/es/openxava_3_1_2_aplicaciones_ajax_desde_entidades_jpa/]

[20] : JavaHispano, Página Oficial, 2009. [Disponible en: http://www.javahispano.org/contenidos/es/openxava_3_1_disponible_incrementa_la_productividad_eludiendo_mvc/]

[21] Java Hispano, Página Oficial, 2009. [Disponible en: http://www.javahispano.org/contenidos/es/naked_objects__dale_poder_a_tus_objetos_de_negocio/]

[22] JavaHispano, Página Oficial, 2009. [Disponible en: http://www.javahispano.org/contenidos/es/jmatter__framework_para_aplicaciones_de_escritorio_de_dos_capas/]

[23] JavaHispano, Página Oficial, 2009. [Disponible en: http://www.javahispano.org/contenidos/es/entrevista_a_eitan_suez_creador_de_jmatter/]

[24] J. García Molina, M. Menárguez, M.J.Ortín, J.Sánchez. Un Estudio Comparativo de dos Herramientas MDA: OptimalJ y ArcStyler, Departamento de Infórmatica y Sistemas, Universidad de Murcia, Campus de Espinardo 30071 Murcia, 2009. [Disponible en: http://74.125.47.132/search?q=cache:MZppcMRuDbcJ:dis.um.es/~jmolina/tdsdm04jesus_final.doc+framework+MDA&cd=7&hl=es&ct=clnk&gl=cu]

[25] Java.Net, Página Oficial, 2009. [Disponible en: http://today.java.net/pub/a/today/2005/06/23/trails.html]

[26] Java.Net, Página Oficial, 2009. [Disponible en: http://weblogs.java.net/blog/johnreynolds/archive/2006/06/_why_jmatter_ma_1.html]

[27] Java.Net, Página Oficial, 2009. [Disponible en: http://today.java.net/pub/n/jMatterOS]

[28] JMatter, Página Oficial,2009. [Disponible en:http://www.jmatter.org/]

[29] Johnson, Ralph E. and Brian Foote. Designing Reusable Classes. Journal of Object-Oriented Programming, 1988.

[30] Johnson, Ralph E. Components, Frameworks, Patterns. Proceedings of the 1997 symposium on Software reusability, 1997.

[31] Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall, 1999.

[32] Lulu, Página Oficial, 2009.[ Disponible en:http://www.lulu.com/content/710734]

[33] Markiewicz, Marcus E. and Carlos Lucena. Object Oriented Framework Development, 2001.

[34] Model Driven Architecture, Página Oficial, 2009. [Disponible en: http://modeldrivenarchitecture.wordpress.com/2008/08/31/introduction3-the-mda-framework/]

[35] Naked Object,Página Oficial, 2009. [Disponible en: [http://www.nakedobjects.org/home/no_for_java_intro.shtml]]

[36] Naked Object,Página Oficial, 2009. [Disponible en: http://www.nakedobjects.org/product/index.shtml]

[37] OhLohRoot , Página Oficial, 2009. [Disponible en: http://www.ohloh.net/p/trails]

[38] O'reilly-OnJava.com, Página Oficial, 2009. [Disponible en: http://www.onjava.com/pub/a/onjava/2006/05/17/standardizing-with-ejb3-java-persistence-api.html]

[39] O' Reilly- OnJava.com, Página Oficial,2009. [Disponible en: http://www.oreillynet.com/onjava/blog/2003/05/naked_objects.html]

[40] O'Reilly-OnJava.com, Página Oficial, 2009. [Disponible en: http://www.onjava.com/pub/a/onjava/2007/08/16/whats-the-matter-with-jmatter.html]

[41] OverView , Página Oficial, 2009. [Disponible en: http://docs.codehaus.org/display/TRAILS/Overview]

[42] Parra Guzmán Patricia y Duarte Delgado Daniel. MDA Model Driven Architecture,Escuela de Ingeniería de Sistemas y Computación, Universidad Valle, 2008.[Disponible en: http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/MDApresentacion.pdf]

[43] Programania, Página Oficial, 2009. [Disponible en: http://www.programania.net/diseno-de-software/jmatter-swing-hibernate-intranet/]

[44] Roma<Meta>Framework,Página Oficila, 2009. [Disponible en: http://www.romaframework.org/]

[45] Roma<Meta>Framework,Página Oficial, 2009. [Disponible en: http://www.romaframework.org/download.htm]

[46] Roma<Meta>Framework,Página Oficial, 2009. [Disponible en: http://www.romaframework.org/documentation.htm]

[47] Roma<Meta>Framework,Página Oficial, 2009. [Disponible en: http://www.romaframework.org/community.htm]

[48] RomaFramework|freshmeat.net,Página Oficial, 2009. [Disponible en: http://freshmeat.net/projects/romaframework/?branch_id=64511=288088]

[49] Sitio de OpenXava,Página Oficial,2009. [Disponible en: http://www.gestion400.com/web/guest/home]

[50] SlideShare , Página Oficial, 2009. [Disponible en:http://www.slideshare.net/emiliobg/presentacion-mda ]

[51] SoftwareLibre.Net,Página Oficial,2009. [Disponible en: http://www.softwarelibre.net/INTERFAZ_DE_USUARIO_AUTOM%C3%A1TICA_CON_OPENXAVA_EL_CAMINO_EVOLUTIVO]

[52] SourceForge.Net,Página Oficial, 2009. [Disponible en: http://sourceforge.net/projects/romaframework/]

[53] SourceForge.Net,Página Oficial, 2009. [Disponible en: http://sourceforge.net/project/platformdownload.php?group_id=160910]

[54] SourceForge.Net,Página Oficial, 2009. [Disponible en:http://sourceforge.net/projects/nakedobjects/]

[55] Suez,Eitan.Building Software Applications with Jmatter, March 2009. [Disponible en: http://jmatter.org/documentation/html/index.html]

[56] Taligent, Corp. Building Object-Oriented frameworks. Cupertino, CA, 1994.

[57] The Naked Objects Framework.Net,Página Oficial, 2009. [Disponible en: http://c2.com/cgi/wiki?TheNakedObjectsFramework]

[58] Tooling the MDA framework: a new software maintenance scheme proposal.Jean Bézivin and Nicolas Ploquin , Página Oficial, 2009.[Disponible en: http://www.sciences.univ-nantes.fr/lina/atl/www/papers/joop01.pdf]

[59] Tutorial de OpenXava,Página Oficial,2009. [Disponible en: http://openxava.wikispaces.com/tutorial_es]

[60] Versión Cero,Página Oficial,2009. [Disponible en: http://www.versioncero.com/NOTICIA/109/OPENXAVA-11-WEBSPHERE-APPS-SIN-WEBSPHERE-STUDIO]

[61] Xebia, Página Oficial, 2009. [Disponible en:http://blog.xebia.com/2008/04/08/jmatter-framework/]

[62] ZeroProgramas,Página Oficial,2009. [Disponible en: http://www.zeroprogramas.com/PROGRAMAS/OPENXAVA-V3-1-1.ASP]

# Glosario de Términos

AJAX es una combinación de cuatro tecnologías ya existentes XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.

Anotación Java es una forma de añadir metadatos al código fuente Java que están disponibles para la aplicación en tiempo de ejecución.

Ant es una herramienta Open-Source utilizada en la compilación y creación de programas Java.

Concurrencia es la habilidad de una aplicación para permitir que varios usuarios graben datos al mismo tiempo sin perder información.

CRUD crear, leer, actualizar y eliminar son las cuatro funciones básicas de almacenamiento persistente.

Código abierto (en inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente.

EJB QL define las consultas para el buscador y recupera entidades.

Enterprise JavaBeans (EJB) es la tecnología del lado del servidor para la arquitectura de componentes en la plataforma Java

Escalable capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.

Generación de código es una de las fases mediante el cual un compilador convierte un programa sintácticamente correcto en una serie de instrucciones a ser interpretadas por una máquina.

HTML (HyperText Markup Language Lenguaje de Marcas de Hipertexto) es el lenguaje de marcado predominante para la construcción de páginas web.

JasperReports es una herramienta de creación de informes Java open source que tiene la habilidad de entregar contenido enriquecido en el monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV y XML.

Java es un lenguaje de programación orientado a objetos.

Java Development Kit (JDK) es un software que provee herramientas de desarrollo para la creación de programas en Java.

Java Platform Enterprise Edition o Java EE es una plataforma de programación (parte de la Plataforma Java) para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java.

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

JavaServer Faces (JSF)es una tecnología para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para la web, en forma de documentos HTML, XML o de otro tipo.

JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java.

JetSpeed es una de las muchas soluciones existentes en el mercado que implementa la especificación de Portlets, componentes web habilitados para Java.

Jetspeed2 es la nueva generación de portales empresariales en Apache, ofrece varias mejoras arquitectónicas y proporciona un mecanismo estándar para el despliegue de portlets.

JPA (Java Persistence API) es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional.

JSR 168 (Portlet Specification o Especificación de portlets) puede construir portlets que puede ejecutarse en los portales, con independencia de sus proveedores.

La Licencia Pública General de GNU más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation.

La plataforma Java es el nombre de un entorno o plataforma de computación, capaz de ejecutar aplicaciones desarrolladas usando el Lenguaje de programación Java

Lenguaje de marcado o lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

Lenguaje interpretado es lenguaje de programación que fue diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados.

LGPL (Library General Public License) es una licencia de software creada por la Free Software Foundation, que pretende garantizar su libertad de compartir y modificar el software libre, esto es para asegurar que el software es libre para todos sus usuarios.

Licencia Comercial esta licencia es para las organizaciones que no quieren liberar el código fuente de sus aplicaciones de código abierto / software libre.

Liferay es un portal de gestión de contenidos de código abierto escrito en Java.

Metadatos según la definición más difundida de metadatos es que son datos sobre datos.

Metalenguaje es un lenguaje usado para hacer referencia a otros lenguajes.

Navegador web (del inglés, web browser) es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet.

Objetos persistentes (también llamados dinámicos) son creados y destruidos por llamadas a funciones específicas.

POJO (acrónimo de Plain Old Java Object) es utilizado por programadores del lenguaje de programación Java para enfatizar el uso de clases simples y que no dependen de un framework en especial

Portal es un lugar central desde el que se puede poner todo tipo de información a disposición de un público muy diverso.

Portlets son componentes modulares de interfaz de usuario gestionadas y visualizadas en un portal web.

Programación Orientada a Objetos (POO u OOP) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora.

Página web es conocida como página de Internet, es una fuente de información adaptada para la World Wide Web (WWW) y accesible mediante un navegador de Internet que normalmente forma parte de un Sitio web.

Servidor de aplicaciones es un servidor en una red de computadores que ejecuta ciertas aplicaciones.

Sistema legacy se llama legacy a los sistemas antiguos que funcionan en una organización y los mismos no pueden ser remplazados por diversos motivos.

Software se refiere al equipamiento lógico o soporte lógico de un computador digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.

Tiempo de ejecución (Runtime en inglés) es el intervalo de tiempo en el que un programa de computadora se ejecuta en un sistema operativo.

WebSphere es una familia de productos de software propietario de IBM.

WebSphere Portal contiene una amplia variedad de tecnologías de portal que ayudan a desarrollar y mantener portales.

Workpace el entrono de trabajo es cualquier lugar en el que una persona, o un equipo, dependa de las Tecnologías de la Información y las Comunicaciones para trabajar.

World Wide Web (WWW) es conocida como la Web o Red Global Mundial.

XML (Extensible Markup Language o lenguaje de marcas ampliable) es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).