UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

HERRAMIENTAS PARA EL CURSO DE
INGENIERÍA DE SOFTWARE

REPORTE DE ACTIVIDAD DOCENTE

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

LUIS ROGELIO OTERO GONZÁLEZ

TUTORA: MARÍA GUADALUPE ELENA
IBARGÜENGOITIA GONZÁLEZ

2008

Hoja de Datos del Jurado

1. Datos del alumno
   Otero
   González
   Luis Rogelio
   56425909
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Licenciado en Ciencias de la Computación
   300167254

2. Datos del asesor
   M. en C.
   Ibargüengoitia
   González
   María Guadalupe Elena

3. Datos del sinodal 1
   Dra.
   Oktaba
   Hanna

4. Datos del sinodal 2
   Dra.
   López
   Gaona
   Amparo

5. Datos del sinodal
   L. I.
   Dávila
   Muñoz
   Magdalena

6. Datos del sinodal
   Mat.
   Levy
   Amselle
   Sylviane

7. Datos de la tesis
   Herramientas para el curso de Ingeniería de Software
   119 p.
   2007

Índice general

Introducción

El curso de Ingeniería de Software que se imparte en la Facultad de Ciencias de la UNAM, que corresponde al séptimo semestre de acuerdo a los planes vigentes (plan 1995), tiene por objetivo "*enseñar las mejores prácticas de la Ingeniería de Software de forma pragmática* [1]*, mientras se desarrolla un sistema en dos ciclos de desarrollo, trabajando en equipo con roles asignados a cada miembro del equipo*" (Ibargüengoitia y Oktaba, 2006).

El objetivo del presente trabajo es servir como punto de apoyo a los alumnos de dicho curso, para que éstos cuenten con una serie de tutoriales de algunas herramientas aplicables en el curso y, en general, en el desarrollo de software, considerando que el aprendizaje en el uso de éstas, es algo que puede tomar un largo tiempo, y que realmente no tiene mucha relevancia para los fines que se persiguen en el curso, como sí lo es, por ejemplo, el aprendizaje de una buena metodología en el proceso de desarrollo.

La experiencia adquirida al momento de cursar esta asignatura, me ha mostrado que los alumnos enfrentamos una serie de dificultades con el software de apoyo que usaremos a lo largo del curso. El primer problema es elegir una buena herramienta, en el sentido de que sea fácil de conseguir, de instalar y de usar, y que cumpla con el objetivo que de ésta se espera. Posteriormente, las dificultades son precisamente la instalación y el uso, pues muchas veces hay que buscar tutoriales en Internet, algunas veces sin mucho éxito, ya sea por la disponibilidad, simplicidad de los ejemplos, complejidad de instalación o de uso, idioma, antigüedad de elaboración, etc.; además de la necesidad de ir navegando y consultando en varios sitios web para cubrir todas las herramientas necesarias para el curso.

El propósito de la elaboración y publicación del material aquí presentado, es aminorar, en la medida de lo posible, tales dificultades, para que los alumnos que cursan la materia puedan enfocarse a los verdaderos objetivos del curso de Ingeniería de Software.

El lenguaje de programación base usado es Java, pues éste es el de uso más difundido en la carrera de Ciencias de la Computación de la Facultad de Ciencias.

Así pues, la estructura del trabajo es la siguiente:

En el capítulo 1 se presenta una clasificación del software de acuerdo a su distribución, uso, posibilidad de modificación y de distribución; esto es para ubicarse mejor entre la amplia gama de posibilidades que tenemos para elegir software de acuerdo a nuestras necesidades. Además, se muestra una lista con algunas opciones de herramientas de acuerdo a la fase de desarrollo de proyectos.

En el capítulo 2 se presentan los tutoriales de instalación y uso básico de una herramienta para cada fase de desarrollo. La elección de éstas fue tomando en consideración aquéllas de uso más difundido actualmente y en mi propia elección al momento de cursar la asignatura con aceptable éxito. Además se incluye una breve descripción y reseña histórica de cada una como

[1] que se centra en el valor práctico.

complemento.

En los siguientes dos capítulos, en cada uno, se presenta el desarrollo de una aplicación web a modo de ejemplo de integración de las herramientas presentadas en los tutoriales. Se trata del clásico ejemplo de la *Autenticación*, que, aunque sencilla, muestra de forma práctica la manera de incluir tales herramientas en el desarrollo. Sin embargo, se presentan de maneras distintas, como se indica a continuación:

En el capítulo 3 se usa como herramienta base el IDE (Entorno Integrado de Desarrollo) NetBeans y, en general, para cada fase de desarrollo se usa una herramienta en específico para llevar a cabo la tarea en turno.

En el capítulo 4 se usa el IDE Eclipse y se muestra la inclusión en éste de diversos componentes adicionales (*plugins*) para cubrir todos los aspectos considerados en el capítulo 3.

El propósito de hacerlo de las dos maneras es ampliar a los alumnos el panorama de opciones para elegir el mejor conjunto de herramientas; por un lado, una herramienta para cada tarea y la integración para funcionar como un todo, y por el otro lado, la manera de explotar una herramienta "todo en uno". Así el alumno podrá formarse la idea de las ventajas y desventajas que cada una representa.

En el capítulo 5 se hace una explicación detallada de la forma de agregar los tutoriales en el sitio web del curso. Los tutoriales están disponibles en éste, y se pueden acceder fácilmente mediante ligas organizadas (dentro del mismo sitio) de acuerdo a las fases de desarrollo propuestas en el curso de Ingeniería de Software.

Todas las instrucciones, tanto de instalación como de uso de las herramientas, así como la aplicación de ejemplo, han sido ejecutadas y verificadas como se puede observar en las imágenes incluídas a lo largo de los manuales.

Espero que el presente trabajo sea del agrado de los usuarios y, sobre todo, de gran utilidad para el mejor aprovechamiento del curso de Ingeniería de Software, pues haciendo uso de estos manuales, los alumnos se pueden ahorrar un buen tiempo que podrán aprovechar mejor en el cumplimiento de los objetivos de la asignatura.

Capítulo 1

Herramientas de Apoyo para el Desarrollo de Software

Al igual que en cualquier profesión, en el desarrollo de software es indispensable contar con herramientas que nos faciliten ciertas tareas.

En general, diremos que una herramienta de apoyo en el desarrollo de software es, a su vez, un software que tiene como propósito facilitar o agilizar algunas tareas durante el proceso de desarrollo, consiguiendo así, aumentar la productividad al reducir los costos de tiempo y esfuerzo. Adicionalmente, si encontramos herramientas de distribución gratuita, como las presentadas en este documento, se reduce también el costo económico. Actualmente existen, tanto herramientas enfocadas particularmente a una fase en el proceso, como aquéllas que integran diversas características que les permiten ser utilizadas en dos o más fases, como en el caso de los IDEs (Entornos de Desarrollo Integrados).

En el presente trabajo, se hace una breve presentación de algunas de estas herramientas, así como una descripción de su funcionalidad, propósitos, y la forma en que se integran con otras herramientas, para desembocar en un ejemplo de aplicación que involucre su uso conjunto. Se hace una mención de las principales herramientas por fase en el desarrollo de software, para posteriormente mostrar su instalación y ejemplo de uso. Antes, daremos un vistazo a una clasificación del software, considerando que en el presente trabajo se exponen herramientas de distribución *gratuita*, ya sean *libres* u *Open Source*, por lo que se pretende dejar en claro las diferencias entre estas dos corrientes y se muestra un panorama general de éstas y las demás categorías del software de acuerdo a esta clasificación.

1.1. Una Clasificación de Software

Dentro del universo de las herramientas consideradas como apoyo en el desarrollo del software, al igual que en casi cualquier ámbito incursionado por los programas informáticos, encontramos diversas opciones al momento de querer obtener algunas de ellas. Por ejemplo, encontramos aquéllas por las que debemos hacer un desembolso económico (pequeño o grande) y por las que no; aquéllas que podemos descargar de Internet sin mayores restricciones; o las que la empresa

desarrolladora nos permite usar gratis sólo por un cierto tiempo, o que nos permiten descargar una versión de prueba sin que todas las funcionalidades que podrían alcanzar estén habilitadas antes de hacer el correspondiente pago; también están las que no sólo nos restringen en cuanto a su adquisición y uso, sino también en cuanto a su modificación, pues puede que se nos esté o no permitido explorar su código fuente; algunas nos restringen o nos dan, además, ciertas libertades en cuanto a su redistribución y copiado.

Podemos englobar todos las consideraciones descritas dentro de estos cuatro aspectos:

- Distribución
- Uso
- Copia
- Modificación

Considerando éstos, mencionaremos algunas categorías del software aun cuando, muchas de las veces, un mismo programa puede caer en más de una categoría, como se verá en su momento. A continuación se tratará de dar una reseña acerca de las principales categorías del software considerando estos cuatro aspectos, empezando por el desarrollo histórico que condujo a muchas de las ramificaciones que conocemos hoy en día.

1.1.1. Desarrollo Histórico

Hasta principios de la década de los 80, poderosas empresas como Microsoft se habían ya consolidado en el ámbito de desarrollo de programas para computadoras, y otras como Apple y Unix tomaban cada vez más fuerza en el desarrollo de sistemas operativos. Fue por estos años cuando comenzó a surgir un movimiento en Estados Unidos que buscaba promover la filosofía de compartir todo el conocimiento que la humanidad fuera capaz de generar, particularmente en el rubro de la generación de software.

Como todo movimiento social, éste se centraba en la figura de un líder, de nombre Richard Stallman. Desde sus años como estudiante de Física en la Universidad de Harvard, Stallman se convirtió en un *hacker* [1] del Laboratorio de Inteligencia Artificial del MIT (Instituto Tecnológico de Massachussets) y junto con algunos de sus compañeros, trataban de buscar una alternativa al monopolio del software que empleaba su Universidad. En particular, buscaban la creación de un sistema operativo que reemplazara a Unix, con apariencia y funcionalidades similares, es decir, tratar de *emular* Unix.

En 1985, Stallman publicó su *Manifiesto GNU*, en el que declaraba sus intenciones y motivos para desarrollar su propio sistema operativo, al que denominó GNU, acrónimo recursivo que significa *GNU No es Unix*. Junto a sus colaboradores desarrollaron gran parte del proyecto, incluyendo herramientas como el conocido editor de texto *Emacs*, el compilador *GCC* para lenguaje C y

[1] Persona con grandes conocimientos y curiosidad en algún tema, en particular, en sistemas informáticos, comúnmente confundido con un *cracker*, quien a diferencia del primero, busca hacer daño en los sistemas.

un ambiente gráfico llamado *XWindow*. Además, iniciaron el proyecto de la Free Software Foundation (FSF), organismo que pretende, hasta nuestros días, el cumplimiento de estos ideales: generar y distribuir software libre. Inicialmente funcionaba con trabajo voluntario y donativos, pero después se hizo necesario conseguir otra fuente de recursos para financiar muchas de sus tareas, encontrando una buena alternativa en la venta de software y manuales de usuario para el mismo. Esto podría parecer un tanto contradictorio: "¡pagar por software libre!", sin embargo, como veremos más adelante, es sólo una de tantas percepciones equivocadas que muchos hemos tenido alguna vez. También, del trabajo de Stallman y sus socios surgió el concepto de *Copyleft* o *Izquierdo de copia*, en una clara oposición al ya conocido *Copyright*. La idea fundamental que se defiende con este concepto, es que el software producido bajo Copyleft puede ser modificado a voluntad de quien lo obtenga. Sin embargo, le confiere la responsabilidad de compartir los cambios que en ese software haya producido. El Copyleft tomó forma física en la GNU/GPL (General Public License o Licencia Pública General de GNU).

Pero el término **libre** llegó a implicar más de lo que pareciera. Para Stallman es toda una filosofía, una forma de vida, incluso ha sido caricaturizado en múltiples ocasiones por su, para muchos, obsesiva manera de abordar el asunto, a tal grado que se dice que él lo ve como que quien no comparta esta ideología está cometiendo crímenes contra la humanidad.

Regresando a la tarea de construir un nuevo sistema operativo, como se mencionó, el equipo de Stallman realizó varias de las tareas necesarias para este fin, aunque aún les hacía falta el núcleo de su naciente sistema operativo. Afortunadamente, por esos días, Linus Torvalds, creador del núcleo Linux, liberaba éste mediante la GPL, así que vieron la posibilidad de unir esfuerzos y surgió el sistema operativo *GNU/Linux*, del que se han desprendido múltiples versiones como Red Hat y Debian.

A principios de los 90 otro movimiento (derivado de GNU) tomaba fuerza como alternativa al software libre: El movimiento *Open Source*, que como su nombre lo indica, tiene por objetivo distribuir software con su correspondiente código fuente disponible para el usuario. La filosofía de este movimiento se basa en que, al compartir el software y tener libre acceso al código para su modificación, con el tiempo, inevitablemente traería mejoras en los programas. Algunos consideran este movimiento más práctico y menos envuelto en cuestiones éticas que el movimiento del software libre. Finalmente ambos movimientos tienen como "enemigo común" al *software privativo o propietario*. Personas de ambos grupos trabajan en proyectos similares e incluso conjuntamente, aun cuando sus principios oficialmente son distintos y la lista de licencias avaladas por cada una es distinta.

Aquí se presenta la lista de premisas que cada movimiento considera como la base de su ideología, aunque muchos encuentran una cierta equivalencia en ambas:

Movimiento del Software Libre

- *Libertad 0:* Ejecutar el programa con cualquier propósito (privado, educativo, público, comercial, etc.).
- *Libertad 1:* Estudiar y modificar el programa (para lo cual es necesario poder acceder al

código fuente).

- *Libertad 2:* Copiar el programa de manera que se pueda ayudar al vecino o a cualquiera.
- *Libertad 3:* Mejorar el programa y publicar las mejoras.

Movimiento del Open Source

1. *Libre redistribución:* El software debe poder ser regalado o vendido libremente.
2. *Código fuente:* El código fuente debe estar incluido u obtenerse libremente.
3. *Trabajos derivados:* La redistribución de modificaciones debe estar permitida.
4. *Integridad del código fuente del autor:* Las licencias pueden requerir que las modificaciones sean redistribuidas sólo como parches.
5. *Sin discriminación de personas o grupos:* Nadie puede dejarse fuera.
6. *Sin discriminación de áreas de iniciativa:* Los usuarios comerciales no pueden ser excluidos.
7. *Distribución de la licencia:* Deben aplicarse los mismos derechos a todo el que reciba el programa.
8. *La licencia no debe ser específica de un producto:* El programa no puede licenciarse sólo como parte de una distribución mayor.
9. *La licencia no debe restringir otro software:* La licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
10. *La licencia debe ser tecnológicamente neutral:* No debe requerirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software.

Cabe destacar que un motivo que generó este movimiento con otro nombre fue desambiguar el término en inglés *free software*, que, traducido a otros idiomas como el español, significa tanto libre como gratuito. Como hemos visto, el concepto de *software libre* no tiene que ver con gratuidad. El software libre puede distribuirse tanto gratuitamente como cobrando miles de pesos; lo importante aquí reside en la libertad que se otorga sobre el software en la obtención del código.

1.1.2. Otras Categorías

Por otro lado, podemos mencionar otras categorías del software que de alguna manera se relacionan con el software libre. Por ejemplo, el *software de dominio público*, que se puede considerar un caso especial de software libre sin Copyleft. La característica principal es que no se encuentra protegida por derechos de autor.

El software semilibre es aquél que está permitido su uso, copia, modificación y redistribución sin fines de lucro. Para la FSF, el software semilibre no podría ser usado en un sistema libre, pues

lo convertiría en “un todo semilibre”.

El software privativo es aquél que restringe o incluso prohibe, ya sea la copia, uso, modificación o redistribución.

El *freeware* no tiene una definición precisa, pero a menudo se relaciona con aquel software que se puede distribuir pero no modificar libremente.

El *shareware* es el software que permite redistribuir copias, pero por cada una de ellas, se debe pagar una licencia. Es uno de los tipos de software más común. Su código fuente no está disponible.

El software privado o a la medida es el que es desarrollado para un usuario en específico, como una empresa. Este usuario es el propietario y por lo general, no lo libera al público.

El software comercial es el que es desarrollado por un negocio para obtener dinero por su utilización.

1.1.3. Licencias

Una licencia de software es la autorización concedida por el titular de los derechos del producto al usuario de ese software, determinando la forma en que ha de ser utilizado y precisando los derechos de copia, modificación y redistribución. Puede ser gratuita o con al gún costo. Puede indicar también la duración, el lugar geográfico de aplicación, etc.

A continuación se mencionan las principales licencias con que se distribuye el software, de acuerdo a las clasificaciones mencionadas:

- *Licencia Pública General (GPL) de GNU:* La más conocida de las licencias de software libre. Es de tipo Copyleft.
- *Licencia Pública General Reducida de GNU (GNU LGPL):* Licencia de software libre que no contiene Copyleft, pues permite mezclarse con módulos no libres.
- *Licencia BSD (Berkeley Software Distribution):* Las condiciones más importantes obligan a mantener intacto el Copyright y mencionar el crédito de la Universidad de Berkeley, California.
- *Licencia MIT:* Licencia de software creada por el MIT (Instituto Tecnológico de Massachussets), que no tiene Copyright, así que está permitida su modificación.
- *El dominio público:* No es una licencia, sino todo lo contrario: se refiere al software que no está protegido por ninguna licencia. GNU lo considera software libre sin Copyleft.
- *Licencia Artística:* La versión original está considerada demasiado vaga y no es libre, sin embargo, la versión 2.0 es compatible con la GNU GPL (es de software libre).
- *Licencia de Software Abierto (OSL) 1.0:* Es una licencia de software libre. Sus autores dicen que se trata de una licencia de tipo Copyleft, pero la GNU no la considera así.

- *Licencia de Apache:* El software tiene pocas restricciones y no impide sacar una copia de Apache, modificarlo y vender versiones binarias. La única restricción es que ya no se le puede llamar Apache.
- *Licencia Pública de IBM:* Licencia de software libre incompatible con la GPL; requiere que se den ciertas patentes sobre las licencias.
- *Licencia Pública de Netscape (NPL):* Licencia publicada por Netscape cuando la empresa decidió publicar su navegador con código abierto. Similar a la BSD.
- *Licencia Pública de Mozilla (MPL):* 'Prima' de la NPL, que fue creada para proteger las contribuciones públicas del proyecto Mozilla.

Y muchas otras que, en algunos casos, han tomado como base las anteriores y en varios casos se desarrollan para un programa en específico.

1.1.4. Conclusiones

Como hemos visto, las categorìas del software suelen confundirse fácilmente respecto a su forma de licenciamiento. Podemos ver que no todo el software que se paga es *software comercial o propietario* y no todo el *software no comercial* es *software libre*, como es una percepción muy difundida. Vemos que es válido pagar por software libre sin que eso constituya una violación a los principios de esta corriente, pues el término **free** del concepto original en inglés *free software* hace referencia a la libertad y no a la gratuidad. De la misma manera que el software no libre no necesariamente es software comercial. Un programa es comercial si se desarrolla como una actividad económica, pero un programa puede o no ser libre dependiendo de su licencia. Al hablar de software gratuito, no debe confundirse con software libre. Vimos también la diferencia, principalmente ética entre el software libre y el open source, haciendo notar que **no son sinónimos**.

En fin, después de todo, está claro que si sólo queremos *hacer uso* de una herramienta de software, podríamos simplemente decidir entre el software que podemos descargar de Internet gratuitamente y el que tenemos que pagar por su licencia. Por supuesto que en la búsqueda de la mejor herramienta para cada una de las necesidades en el desarrollo de nuestro software, ésta no es la mejor referencia para decidirnos por una de ellas. Sin embargo, es bueno saber que hay más categorías y que podemos evitar un mal uso de la jerga computacional.

1.2. Algunas Herramientas de Acuerdo a la Etapa en el Ciclo de Desarrollo

A continuación se muestran algunas de las herramientas más empleadas en ciertas fases de desarrollo. Sólo se mencionan aquellas fases en las que la ayuda de una herramienta es necesaria y que existe más software con ese fin.

Al principio de cada una de las fases presentadas se describen los objetivos de las mismas, para contextualizar mejor. Las listas de objetivos fueron tomadas del libro *Ingeniería de Software Pragmática* [1].

Se incluye el tutorial de instalación y uso en las herramientas que así se indica.

1.2.1. Fase de Estrategia (*Herramientas para la Administración de la Configuración*)

Los principales objetivos de esta fase son:

- La planeación y elección de una estrategia para desarrollar el sistema propuesto en los 2 ciclos que se manejan en el curso: A partir de la definición detallada de lo que debe hacer el software a desarrollar, se plantean varias estrategias para desarrollar el producto en 2 ciclos, considerando que al final de cada ciclo se debe tener un producto funcional. Al término de la fase se debe haber elegido la mejor estrategia.
- En el segundo ciclo, aprender los fundamentos de la Administración de la Configuración, y la necesidad de controlar los cambios a los documentos generados en el primer ciclo.

Es en este último punto que se centra esta sección.

La Administración de la Configuración se refiere al control de las versiones de los productos elaborados en el desarrollo del sistema, principalmente del código fuente, para que todos los miembros del equipo de trabajo estén al tanto de las modificaciones.

Las herramientas de uso más difundido para este fin son:

- **CVS (Concurrent Versions System)**: Herramienta tradicional en el control de versiones desarrollada por el proyecto GNU. Posee algunas deficiencias, como el hecho de que no se pueden sustituir versiones, sino que es necesario borrar la versión anterior e incorporar la nueva, así como problemas con el renombrado de archivos.
- **Subversion**: Herramienta colocada como sucesor de CVS. Desde su creación el principal propósito ha sido atacar las deficiencias de este último. Además, se han desarrollado *plugins* para una fácil interacción con IDEs como Eclipse y NetBeans (Ver tutorial en el Capítulo 2.1).

La principal idea atrás de estas herramientas es poseer una arquitectura cliente - servidor, donde el servidor contiene las versiones de los productos y un historial, y los clientes se conectan a éste para hacer y compartir las modificaciones. Tanto el servidor como un cliente pueden residir en la misma computadora.

1.2.2. Fase de Planeación (*Herramientas para la Administración de Proyectos*)

Los principales objetivos de esta fase son:

- Hacer la planeación de las actividades del equipo indicando qué actividades se harán, cuándo y por quién (Plan de trabajo del equipo).

- Introducir el concepto de calidad e introducir las revisiones entre colegas como técnica de calidad: Se pretende ir asegurando la calidad en el producto durante el proceso de desarrollo. Se propone la técnica de revisiones entre colegas periódicamente con el fin de detectar posibles problemas y corregirlos antes de que éstos causen otros mayores.

El plan de trabajo del equipo se modela en un Diagrama de Gantt, herramienta típica dentro de la administración de proyectos.

La administración de proyectos consiste en gestionar la generación de un producto dentro del tiempo dado y los límites de recursos.

Dentro de los puntos que comprende están:

- El costo total del proyecto
- Las capacidades de los productos
- La calidad de los productos
- La duración del proyecto
- Estructura (elementos organizacionales involucrados)
- Proceso administrativo (responsabilidades y supervisión de los miembros del equipo de trabajo)
- Proceso de desarrollo (métodos, herramientas, lenguajes, documentación y apoyo)
- Programa (tiempos en los que debe realizarse las porciones de trabajo)

Ejemplos de herramientas para apoyar la administración de proyectos son:

- **Microsoft Project**: Es un programa de la suite de Microsoft Office que sirve para gestión de proyectos, por lo que sólo está disponible en plataforma Windows. Si se está familiarizado con el ambiente de la suite ofimática de Microsoft Office (Word, Excel, etc.) es muy fácil su uso.
- **GanttProject**: Es un programa codificado en Java de distribución gratuita, un tanto simple pero muy práctico. Permite generar informes en formato HTML y PDF (Ver tutorial en el Capítulo 2.2).

1.2.3. Fase de Diseño (*Herramientas para el Diseño o Modelado*)

Los principales objetivos de esta fase son:

- Definir el proceso del diseño para el trabajo en equipo: En esta parte del desarrollo, entre otras cosas, se decide las tecnologías y el ambiente de implementación y se establece el estándar de diseño, que consiste en una convención entre los miembros del equipo que les facilite trabajar por separado y reunir su trabajo sin dificultad.

- Describir las partes de las cuales se va a componer el sistema y mostrar sus relaciones en la arquitectura: La arquitectura establece las partes o componentes del software, sus propiedades y sus relaciones.
- Hacer el Plan de pruebas de integración: En esta parte se determina el orden en que se integrarán los componentes del producto de software.
- Establecer las clases con que se construirá el software en sus vistas estática (diagramas de clases) y dinámica (diagramas de secuencia y de estados).

Esta sección se ocupa del último punto, al recomendar herramientas de apoyo en la elaboración de los diagramas señalados, es decir, en el diseño del sistema.

El diseño es la actividad cuya meta es preparar por completo el proyecto para su construcción. En otras palabras, los programadores deben describir un diseño detallado para luego concentrarse en aspectos de código.

Las herramientas de modelado se ocupan de la representación gráfica de sistemas de software utilizando diagramas UML para visualizar, diseñar y estructurar las características del proyecto, incluyendo aspectos conceptuales, tales como procesos y funcionalidades del sistema, así como aspectos concretos, tales como expresiones del lenguaje en que se esté desarrollando el sistema, esquemas de bases de datos, etc.

Para apoyar estas tareas se tienen los siguientes ejemplos de herramientas:

- **ArgoUML**: Es una herramienta para construir diseño orientado a objetos. Está codificado completamente en Java, por lo que está soportado en cualquier plataforma con la máquina virtual de Java. Soporta la última especificación de UML. Es modular y extensible (Ver tutorial en el Capítulo 2.3).
- **Dia**: Puede ser usado para modelar varios tipos de diagramas UML. Es parte del proyecto GNOME. Además de diagramas UML puede ser usado para crear diagramas de flujo. Puede trabajar en Windows y Linux.
- **Poseidon For UML**: Es una herramienta comercial derivada de ArgoUML. Soporta UML 1.2 y genera código Java así como C#, VB.Net, C++ y otros lenguajes a partir de diagramas. Además posee una arquitectura de *plugins* que permite añadir nuevas características.
- **Visio**: Herramienta desarrollada por Microsoft. Era desarrollada por una empresa independiente hasta que Microsoft la compró y la incorporó a su suite ofimática en el año 2000.

1.2.4. Fase de Construcción y Pruebas Unitarias

El principal objetivo de esta fase es la implementación del software, incluyendo la programación y las pruebas unitarias. Se busca satisfacer los requerimientos de la manera que especifica el diseño detallado.

1.2.4.1. *Herramientas para el Diseño y Construcción de Bases de Datos*

Los Sistemas Manejadores de Bases de Datos (SMBD) se encargan del almacenamiento y manipulación de los datos disponibles en un sistema. Un SMBD implementa el Lenguaje de Consulta Estructurado (SQL), lenguaje estándar en el manejo de datos en bases de datos relacionales. Además, tiene como objetivo el servir como enlace entre las bases de datos, el usuario y las aplicaciones que dispondrán de esos datos.

Los SMBD de uso más difundido en el mundo académico son:

- **MySQL**: Es desarrollado por la empresa MySQL AB en dos distribuciones: una libre bajo la licencia GNU GPL y la otra a través de la venta de una licencia, que incluye soporte técnico. Es el manejador de bases de datos más usado en el mundo con más de 6 millones de usuarios, según cifras de la empresa (Ver tutorial en el Capítulo 2.4.1).
- **PostgreSQL**: Inició como proyecto estudiantil en la Universidad de Berkeley, California con el nombre de Ingres. Posteriormente el proyecto fue adoptado por una gran cantidad de desarrolladores que han colaborado en su crecimiento (Ver tutorial en el Capítulo 2.4.2).

En cuanto al diseño, contamos con modeladores de bases de datos. Estas herramientas nos permiten tener una visión gráfica de las tablas y relaciones que intervienen en nuestra base de datos, particularmente si ésta está basada en el modelo Entidad - Relación. Aquí se presenta el tutorial de **DB Designer Fork**, herramienta *Open Source* (Ver tutorial en el Capítulo 2.4.3).

1.2.4.2. *Servidor de Aplicaciones*

Un servidor de aplicaciones web es un programa que implementa el protocolo HTTP para ejecutar páginas web. El servidor se mantiene a la espera por parte de los clientes (a través de un navegador) y cuando una información es solicitada, ésta es enviada por parte del servidor. En el caso de una aplicación Java, ésta se ejecuta en el servidor antes de enviar el código HTML derivado de esa ejecución, y que ha de ser interpretado por el navegador del lado del cliente.

El servidor de aplicaciones Java por excelencia es:

- **Tomcat**: Herramienta escrita en Java desarrollada por el proyecto Jakarta, que implementa los Java Server Pages (JSP) y los Java Servlets, tecnologías Java desarrolladas por Sun Microsystems para el web (Ver tutorial en el Capítulo 2.5).

1.2.4.3. *Entornos de Desarrollo Integrados*

En este rubro, los Ambientes de Desarrollo Integrados (IDEs) tienen un amplio uso para permitir que los programadores produzcan más código en menos tiempo. Reúnen varias características en una herramienta tales como: un editor de texto, un compilador o un intérprete o ambos, un depurador, control de versiones del código, visualización de la estructura del proyecto, entre otras. Adicionalmente se pueden agregar funcionalidades y capacidades para diversos lenguajes de programación mediante mecanismos llamados *plugins*.

Algunos ejemplos de estas herramientas son:

- **Jbuilder**: Herramienta desarrollada por Borland. Posee tres distribuciones principales: Enterprise (para aplicaciones J2EE), Developer (aplicaciones Java completas) y Foundation (con capacidades básicas y gratuita, al menos por el momento).
- **Visual C++**: Herramienta desarrollada por Microsoft. Es un IDE para aplicaciones en C y C++. Cuenta con una versión de distribución gratuita (Express Edition).
- **VisualAge for Java**: IDE desarrollado por IBM.
- **Forte for Java**: IDE desarrollado por Sun Microsystems.
- **NetBeans**: Empezó como un proyecto estudiantil, luego pasó a ser patrocinada por Sun y actualmente es una herramienta desarrollada por la empresa homónima. Es el IDE de uso más difundido en aplicaciones Java (Ver tutorial en el Capítulo 2.6).
- **Eclipse**: Herramienta desarrollada por la Fundación Eclipse. Es el IDE de uso más universal, pues soporta diversos lenguajes de programación y se pueden agregar varias funcionalidades mediante *plugins*. En palabras de la gente del propio proyecto, Eclipse es "*para todo y nada en particular*" (Ver tutorial en el Capítulo 2.7).

1.2.4.4. *Herramienta para las Pruebas*

Las *pruebas unitarias* son las primeras que se aplican, al igual que con el aseguramiento de la calidad en general, es recomendable que las pruebas de código sean realizadas por personas distintas a las que lo desarrollaron. Cuando un desarrollador prueba su propio código, tiende a ocultar justo lo que debería descubrir, pues se forma una visión de lo que debería hacer su código en circunstancias típicas.

El siguiente nivel de pruebas lo constituyen las *pruebas de integración*. Esto valida la funcionalidad global de cada etapa parcial de la aplicación. Los casos de uso son la base para algunas pruebas. Las entradas para el proceso de planeación de pruebas son los requerimientos y el diseño detallado.

La principal herramienta, específicamente para aplicaciones Java es:

- **Junit**: Es un conjunto de clases Java para hacer pruebas unitarias de aplicaciones (también Java). Se debe crear una clase que contenga instancias de las clases a probar con sus respectivos métodos. La ejecución se da de manera controlada al escribir en esta nueva clase, lo que los métodos a probar deben devolver. Si el resultado es el esperado, la prueba es exitosa (Ver tutorial en el Capítulo 2.9).

1.3. Conclusiones del Capítulo

Hoy en día, el desarrollo de software requiere de varias tareas, lo que sugiere el uso y la integración de varias herramientas para que nos apoyen con su realización. Como hemos visto a lo largo de este capítulo, ya no sólo se requieren herramientas para la edición y la compilación del código, sino que se necesita también de herramientas que nos auxilien en la administración

de tareas de los miembros del equipo, así como formas de comunicación más eficientes y a la vez productivas entre los mismos; planeación y ejecución de pruebas de todos nuestros productos, generación de reportes relativos al desarrollo, control de cambios, etc.

Todos estos aspectos se han ido considerando por los desarrolladores y proveedores con el tiempo, por lo que la forma en que se han utilizado ha ido variando. Una parte de esta evolución ha ido enfocada a la integración y al tipo de herramientas "todo en uno", que buscan reunir las diversas funcionalidades de las demás herramientas. En mi experiencia, los IDEs son herramientas muy útiles, que sin embargo, requieren de mucha memoria RAM; aunque sabemos que en el mundo del hardware, lo que más se abarata es precisamente la memoria.

Hemos echado un vistazo por herramientas libres, de código abierto, gratuitas, comerciales, etc., conociendo características, manejo, instalación y ejemplos de uso. Pero hay que tener presente que el emplear una herramienta nunca reemplazará un proceso bien definido. Lo principal son los recursos humanos, y por supuesto, la preparación y organización de los mismos.

En general, el software evoluciona rápidamente y con ello, el proceso de desarrollo también; todo en busca de la satisfacción de los clientes y de productos con cada vez más calidad. Hay que estar preparados y abiertos a estos cambios, y, ¿por qué no?, pensar también en desarrollar nuevas herramientas de apoyo, pues como todo en el mundo laboral, debe existir competencia y búsqueda por la mejora continua.

Capítulo 2

Tutoriales de las Herramientas de Apoyo en el Desarrollo de Software

A continuación se muestra una guía básica para la instalación de las herramientas de uso más difundido en el desarrollo de Software, organizadas por etapa de desarrollo, así como una breve reseña histórica y algunos ejemplos de uso, esperando que sean de gran utilidad para su rápido aprendizaje y mejor aprovechamiento.

2.1. Herramienta para la Administración de la Configuración

Subversion es un software de sistema de control de versiones para proyectos que involucren varias personas, particularmente usado por desarrolladores de software. Fue diseñado específicamente para reemplazar a CVS, el cual posee varias deficiencias. Es software Open Source bajo una licencia de tipo *Apache/BSD*.

2.1.1. Origen del Proyecto Subversion

A principios del año 2000, la empresa CollabNet Inc. comenzó a buscar desarrolladores para crear una alternativa a CVS. Al principio, CollabNet usaba CVS como manejador del control de versiones para algunos de sus proyectos, pero, al encontrar muchas deficiencias, decidieron empezar un nuevo sistema para tal fin desde cero, manteniendo las características esenciales de CVS.

2.1.2. Instalación de Subversion

1. Visita la página:
   http://subversion.tigris.org/ (Fig. 2.1).

2. Descarga la última versión de Subversion (Al momento de realizar esta guía es la 1.4.3), *Windows installer with the basic Subversion Win32 binaries.*

3. Ejecuta el archivo dando *doble click* sobre el ícono (no es recomendable instalarlo en directorios que tengan nombres con espacios).

4. Al terminar, teclea en línea de comandos desde cualquier directorio:

   ```
   C:/>svn
   ```

   Tras lo cual debe aparecer:

   ```
   Tipee 'svn help' para ver el modo de uso.
   ```

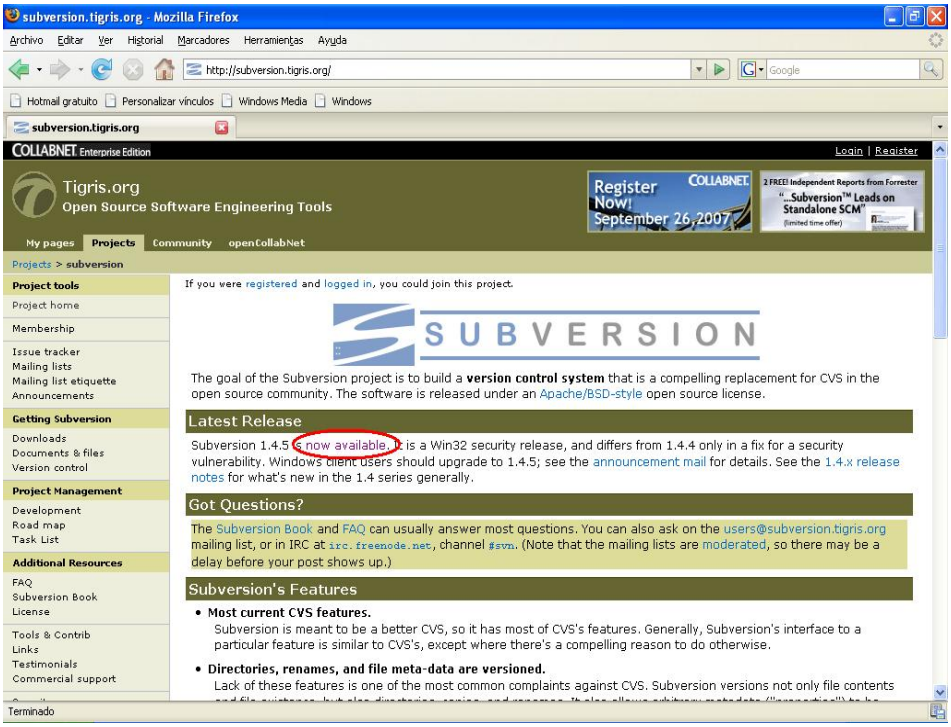

Figura 2.1: Sitio de Subversion.

2.1.2.1. Crear un Repositorio

Un repositorio funciona como un árbol de directorios donde se alojarán los archivos y los cambios en los mismos.

- Desde línea de comandos, colócate en el directorio donde quieras crear el repositorio y teclea:

  ```
  svnadmin create Repositorios
  ```

  Puedes observar la estructura de directorios creada.

Para mayor comodidad, se puede acceder al repositorio desde algún navegador mediante el protocolo HTTP, por lo que Apache es un muy buen complemento para este fin. A continuación se muestra el proceso de instalación.

2.1.3. Instalación de Apache

- Visita la página de descarga de Apache **http://httpd.apache.org/download.cgi** y descarga la versión 2.0.x (es la última compatible con Subversion al momento de hacer esta guía) (Fig. 2.2). Es recomendable descargar el archivo ejecutable y no el comprimido, la sección con archivos ejecutables está en:
  http://archive.apache.org/dist/httpd/binaries/win32/, localiza el archivo:
  apache_2.0.59-win32-x86-no_ssl.msi y descárgalo.

- Da *doble click* en el ícono para iniciar la instalación (Fig. 2.3).

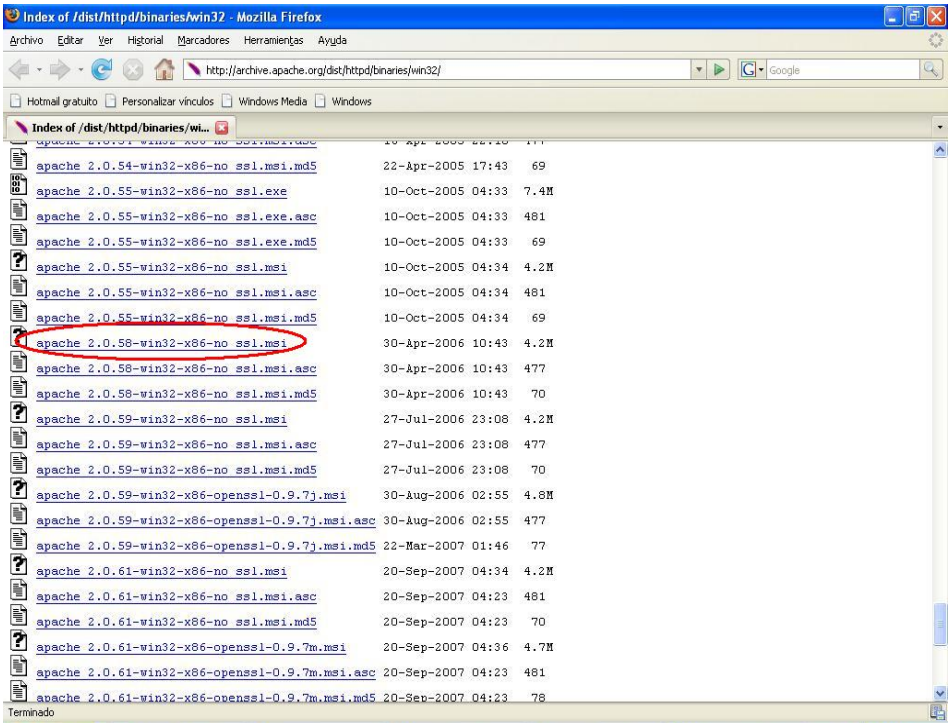

Figura 2.2: Página para descargar Apache.

- Al igual que en el caso de Subversion, evita instalar Apache en carpetas cuyos nombres contengan espacios en blanco (Fig. 2.4, por default se instala en una subcarpeta **Apache2**). Supongamos que se instala en **C:/**.

- Por default, se carga como un servicio de Windows y aparece un ícono en la parte inferior de la pantalla, junto al reloj.

- Puedes verificar la instalación tecleando **http://localhost:80** (Fig. 2.5).

Para que Apache pueda ser utilizado conjuntamente con el repositorio que acabas de crear, se requiere hacer ciertas modificaciones en la configuración.

Figura 2.3: Instalación de Apache.

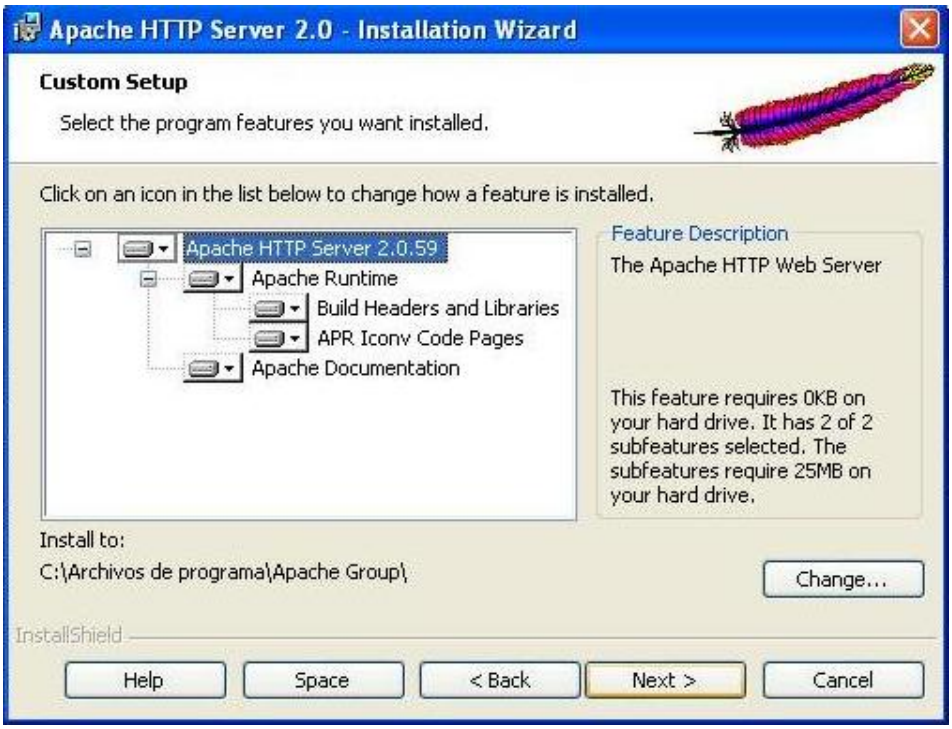

Figura 2.4: Selecciona la carpeta de destino de instalación de Apache.

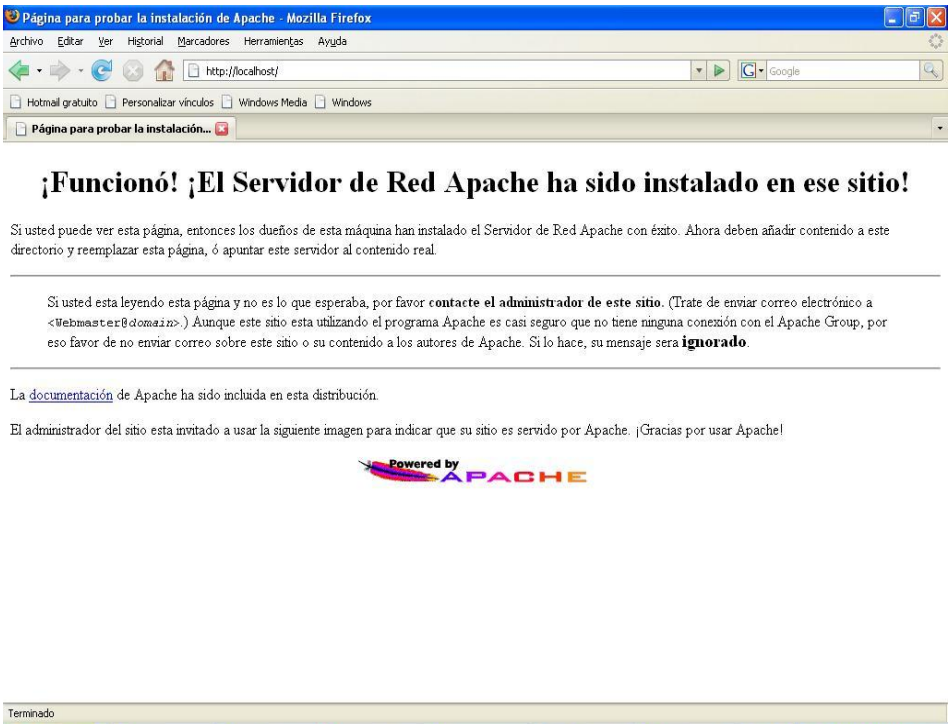

Figura 2.5: Verificación de la correcta instalación de Apache.

- Copia los archivos **mod_authz_svn.so** y **mod_dav_svn.so** de la carpeta **bin** de Subversion en la carpeta **modules** de Apache, así como los archivos *.dll* de la carpeta **Apache2/bin** en **Apache2/modules**.
- Abre el archivo **httpd.conf** de la carpeta **conf** de Apache con un editor de texto.
- Localiza la sección de cargar módulos (LoadModules) y añade al final las líneas:

  ```
  LoadModule dav_svn_module "C:/Apache2/modules/mod_dav_svn.so"
  LoadModule authz_svn_module "C:/Apache2/modules/mod_authz_svn.so"
  ```

  Que corresponden a los archivos que acabas de copiar.
- Para que Apache identifique el repositorio creado, añade al final del archivo las siguientes líneas (suponiendo que el repositorio fue creado en C:/Repositorios).

  ```
  <Location /repos>
  DAV svn
  SVNPath C:/Repositorios/proyecto1
  </Location>
  ```

  Esto indica que el proyecto será invocado desde el navegador como *http://localhost/repos*

- En una terminal (Símbolo del Sistema) colócate en la carpeta **bin** de Apache y teclea:

  ```
  apache -e debug
  ```

- Si no te ha marcado error alguno, la terminal se queda "esperando" y ya puedes acceder al repositorio desde el navegador. Teclea la dirección **http://localhost/repos/proy1/**. Debe aparecer una imagen como la indicada (Fig. 2.6). Si existe algún error, verifica los pasos anteriores.

- Teclea *Ctrl + c* para salir y ya puedes reiniciar Apache desde el Monitor de Apache (junto al reloj).

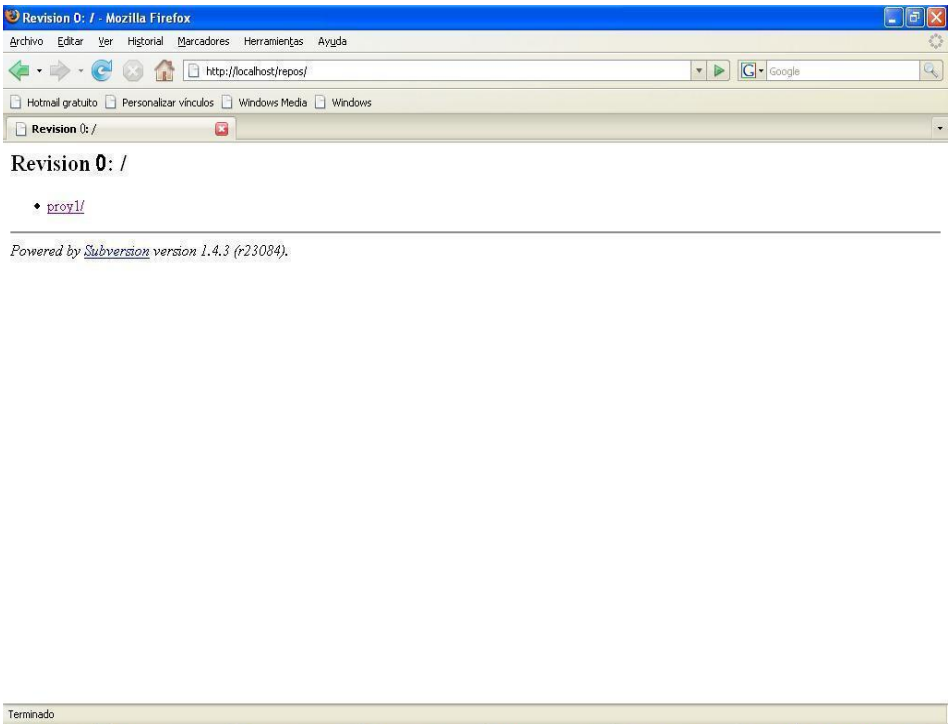

Figura 2.6: Visualización del proyecto desde el navegador.

Para restringir el acceso al proyecto, se pueden usar distintos niveles de seguridad. Aquí se presenta el proceso para autentificación básica.

- Crea un archivo que contendrá los passwords. En una terminal (Archivos de Programa) colócate en la carpeta **bin** de Apache.

- Teclea lo siguiente:

  ```
  htpasswd.exe -nb <login> <password> > usuarios.txt
  ```

  Donde ¡login¿es tu clave de usuario y ¡password¿es tu contraseña.

- Y volvemos a modificar el archivo **httpd.conf** de la carpeta **conf** de Apache

```
<Location /repos>
DAV svn
SVNPath C:/Repositorios/proyecto1
AuthType Basic
AuthName "Repositorio Subversion"
AuthUserFile C:/Apache2/bin/usuarios.txt
require valid-user
<LimitExcept GET PROPFIND OPTIONS REPORT>
Require valid-user
</LimitExcept>
</Location>
```

- Ejecuta de nuevo:

  ```
  apache -e debug
  ```

  para verificar posibles errores.

- Si todo ha salido bien, teclea en el navegador **http://localhost/repos/**. La autentificación proporcionada anteriormente, será ahora requerida (Fig. 2.7).

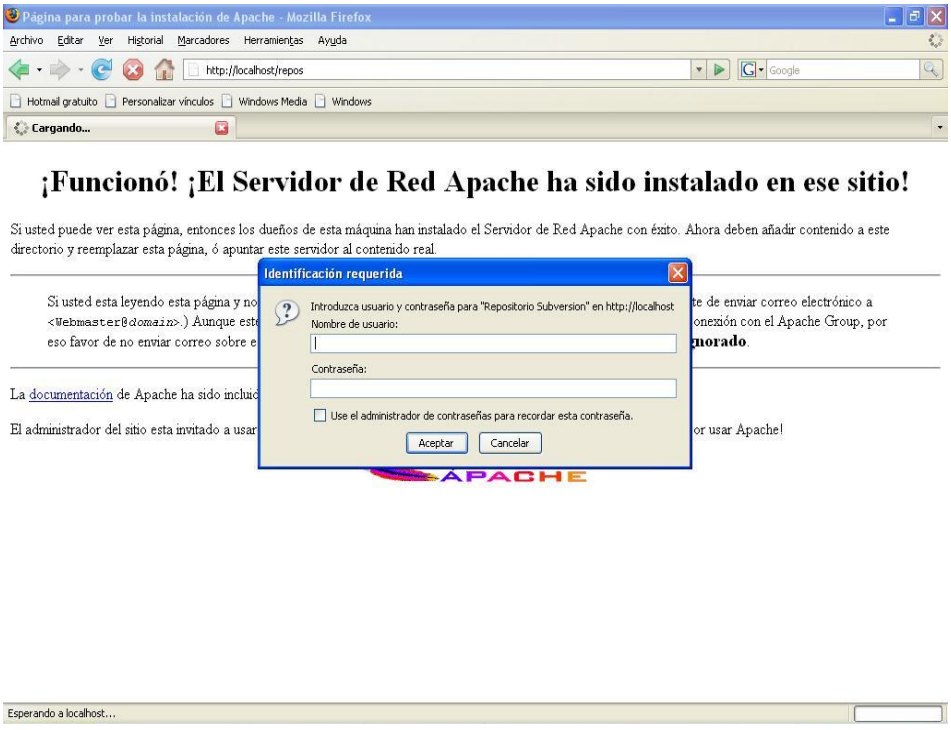

Figura 2.7: Solicitud de autentificación.

2.1.4. Comandos Básicos

- Cuando creamos un nuevo repositorio no es posible hacer transacciones directamente sobre éste, sino que tenemos que realizar una copia de trabajo. Cuando realizamos una copia de trabajo local, es decir, de un repositorio creado en nuestra máquina, tecleamos:

  ```
  > svn checkout file:///Repositorios/Proyecto proyecto1
  ```

  Suponiendo que el repositorio del que queremos hacer una copia de trabajo es **Proyecto** que está en la carpeta **Repositorios** y le queremos dar el nombre **proyecto1**.

- Para descargar el proyecto desde otra computadora, desde una terminal (Símbolo del Sistema), sitúate en el directorio donde quieras descargar el proyecto y teclea:

  ```
  > svn checkout http://<dir_ip>:80/carpeta_proyecto nombre_proyecto
  ```

  Donde **dir_ip** es la dirección ip de la computadora en la que se encuentra el proyecto a descargar, **carpeta_proyecto** es la carpeta donde se aloja el proyecto a utilizar, y **nombre_proyecto** es la manera en que se llamará la carpeta que contenga una copia de trabajo del repositorio.

- Para ver la bitácora de movimientos en el proyecto:

  ```
  > svn log
  ```

- Para agregar un archivo, cópialo en la carpeta de tu proyecto (en la copia de trabajo), sitúate en esta carpeta y teclea:

  ```
  > svn add <archivo>
  ```

- Una vez que se ha agregado, para que la modificación surta efecto, teclea:

  ```
  > svn commit <archivo> -m 'archivo_agregado'
  ```

  Donde **-m** es la bandera para indicar un mensaje de identificación al nuevo movimiento Ahora puedes ver tu archivo desde el navegador actualizando la página donde se visualiza tu proyecto.

- Para eliminar un archivo, teclea:

  ```
  > svn remove <archivo>
  ```

Si bien el proceso de instalación y aprendizaje en el manejo de Subversion puede parecer un poco complejo, una vez que se adquiere el dominio en su uso resulta una herramienta sumamente útil para la colaboración entre los miembros del equipo. Además, actualmente los IDEs poseen *plugins* que facilitan el uso de Subversion. En el presente manual se presenta una manera básica e individual de usar Subversion. El lector puede investigar por su cuenta un poco más al respecto.

Ojalá este pequeño tutorial te haya sido de ayuda. Suerte.

2.2. Herramienta para la Administración de Proyectos

GanttProject es un programa usado en la gestión de proyectos desarrollado por Source Forge. Implementa la principal herramienta para la administración del tiempo y los recursos con que se cuenta, como lo es el Diagrama de Gantt, desarrollada por Henry Laurence Gantt en 1910.

Es una herramienta Libre, cubierta por diversas licencias, en su mayoría GPL.

2.2.1. Origen de GanttProject

El desarrollo de esta herramienta comenzó en la Universidad de Marne la-Valle, Francia, en diciembre de 2002. Con el tiempo muchos desarrolladores han colaborado para su mejoramiento. Es usado por muchas universidades, compañías y equipos de desarrollo.

Recientemente un departamento del gobierno Francés mandó mejorar el producto, para lo que contrató a la empresa Actimage para añadir nuevas características.

2.2.2. Instalación de GanttProject

GanttProject está codificado en Java, por lo que se requiere tener instalado el JRE (Java Runtime Environment).

- Visita la página **http://ganttproject.biz/download.php**.
- Da click en **Windows Installer**.
- Descarga el archivo (Fig. 2.8).
- Inicia la instalación con doble click en el archivo descargado (Fig. 2.9).
- Sigue los pasos hasta terminar.

2.2.3. Uso de GanttProject

Vamos a mostrar un ejemplo sencillo de uso de GanttProject, las actividades a modelar serán precisamente las de descargar, instalar y hacer ejemplos de uso de esta herramienta.

- Abre la ventana de GanttProject, se muestra el consejo del día (Fig. 2.10). A la izquierda se visualizan las tareas del proyecto y a la izquierda el diagrama de Gantt correspondiente.
- En la barra superior, elige *Proyecto* → *Nuevo*.
- En la ventana escribe la descripción general del proyecto (Fig. 2.11).
- Elige *Tarea* → *Nueva Tarea* y teclea la descripción.
- Da click derecho en la tarea recién creada para editar las propiedades como: fechas de inicio y fin, prioridad, asignación de recursos, etc (Fig. 2.12).

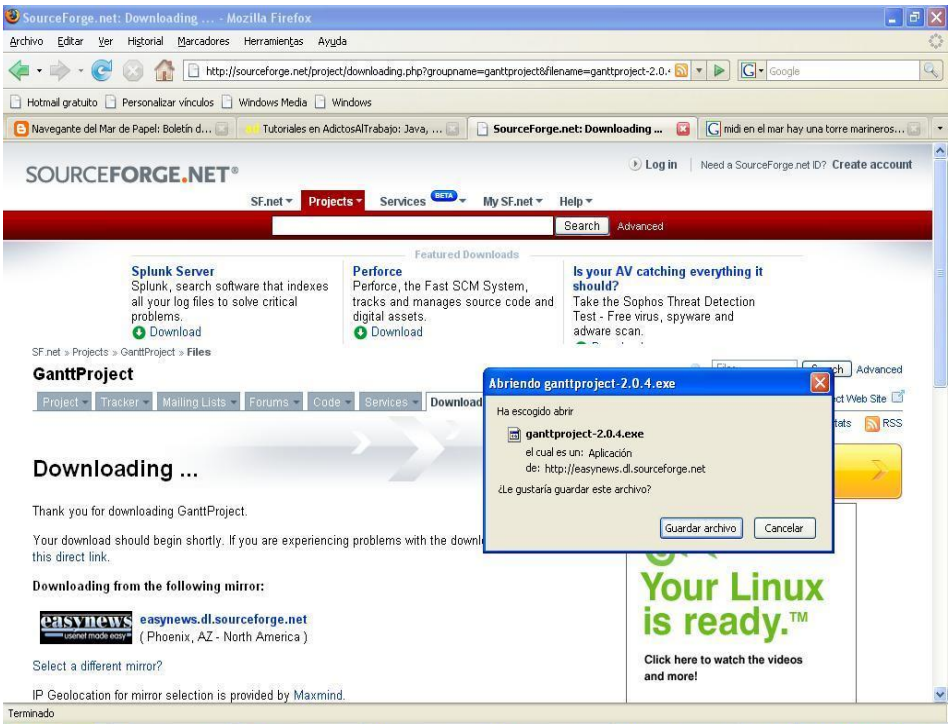

Figura 2.8: Página de descarga de GanntProject.

Figura 2.9: Instalación de GanntProject.

- Si una tarea es subtarea de otra, da click en la opción *indentar* (Fig. 2.13).
- Para asignar recursos humanos, selecciona *Recursos* → *Nuevo Recurso* en la barra superior.
- Introduce los datos para la descripción del nuevo recurso (Fig. 2.14).
- Al regresar a la ventana con las tareas y editar las propiedades de una de ellas, estará disponible el nuevo recurso para que le sea asignada la tarea seleccionada. (Fig. 2.15).

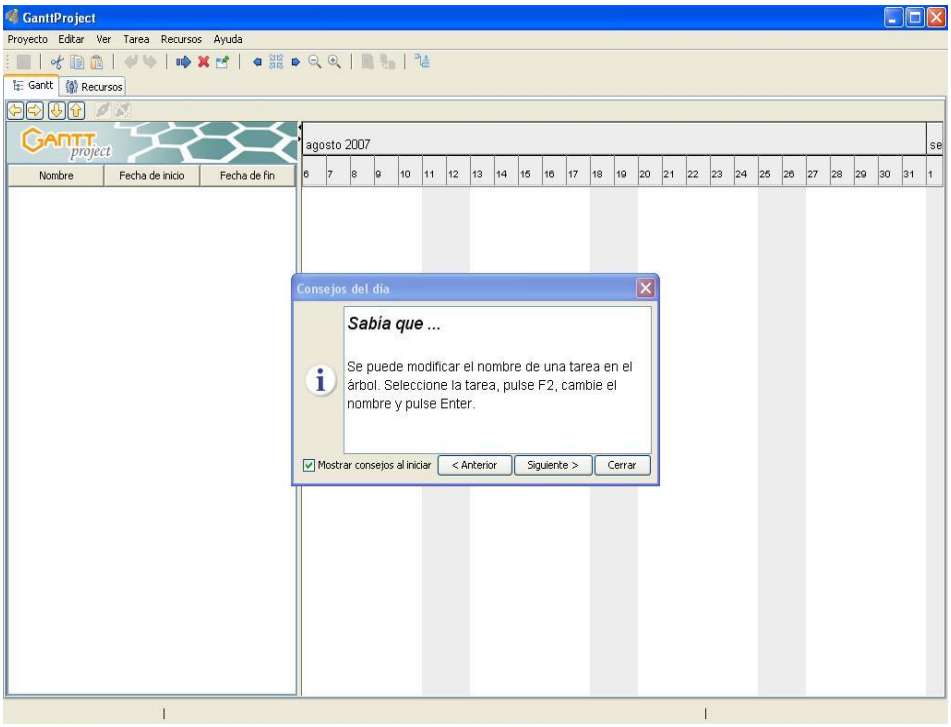

Figura 2.10: Ventana de inicio de GanntProject.

GanttProject resultó ser una herramienta de manejo muy sencillo e intuitivo, a primera vista parece un tanto simple pero resultó cubrir adecuadamente los fines que se buscaban en esta fase de desarrollo.

Ojalá este pequeño tutorial te haya sido de ayuda. Suerte.

2.3. Herramienta para el Diseño o Modelado

ArgoUML es una herramienta para el modelado de proyectos por medio de UML (Lenguaje Unificado de Modelado). Está codificado en Java, por lo que basta tener instalado el JRE (Java Runtime Environment) para ejecutarlo. Esto lo hace independiente de la plataforma. Es una herramienta *Open Source* muy fácil de usar.

Dentro de los incovenientes se encuentran: el que no tiene un botón para *deshacer* y consume

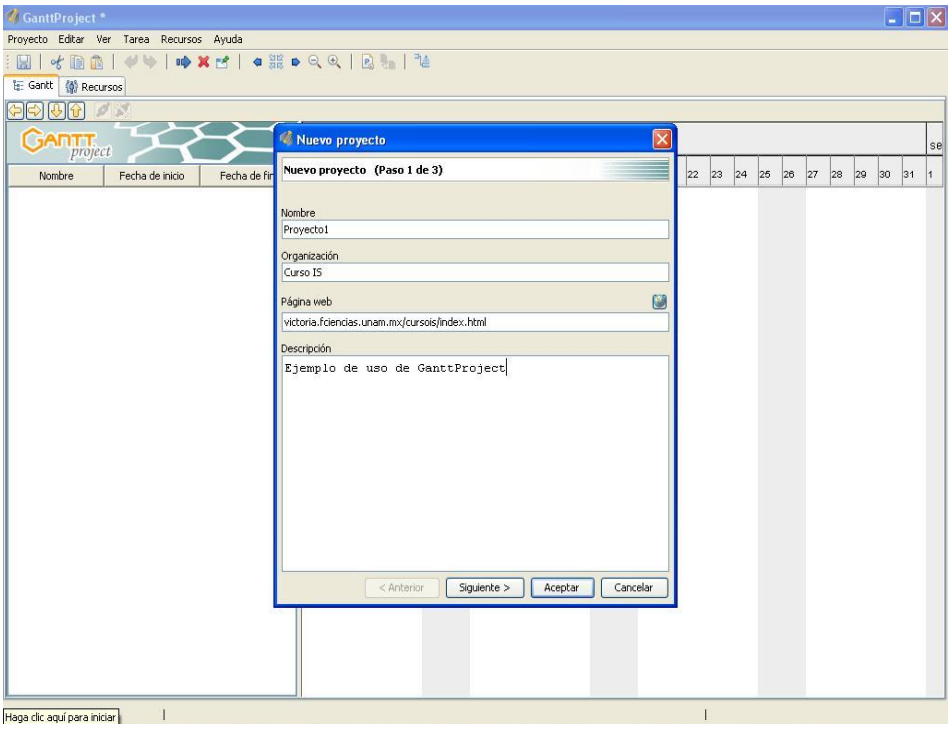

Figura 2.11: Descripción del proyecto.

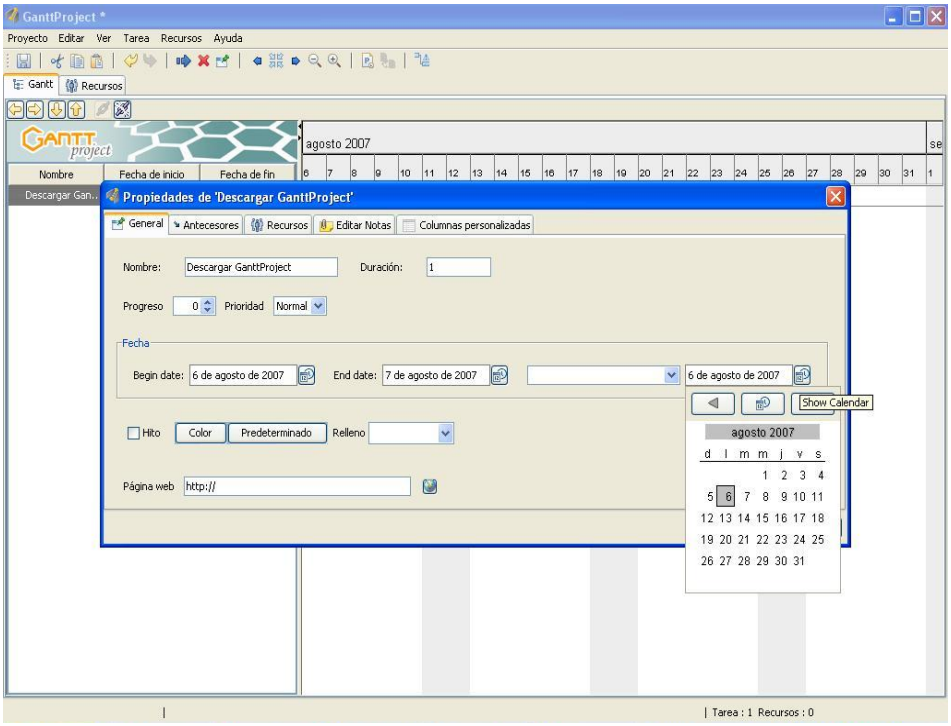

Figura 2.12: Propiedades de las tareas.

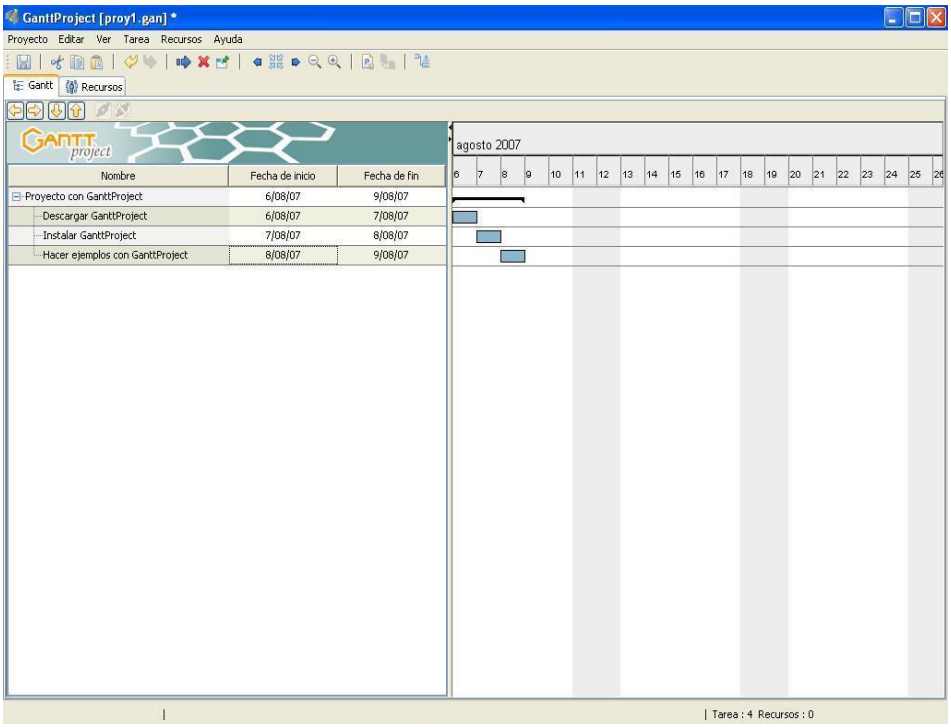

Figura 2.13: Subtareas.

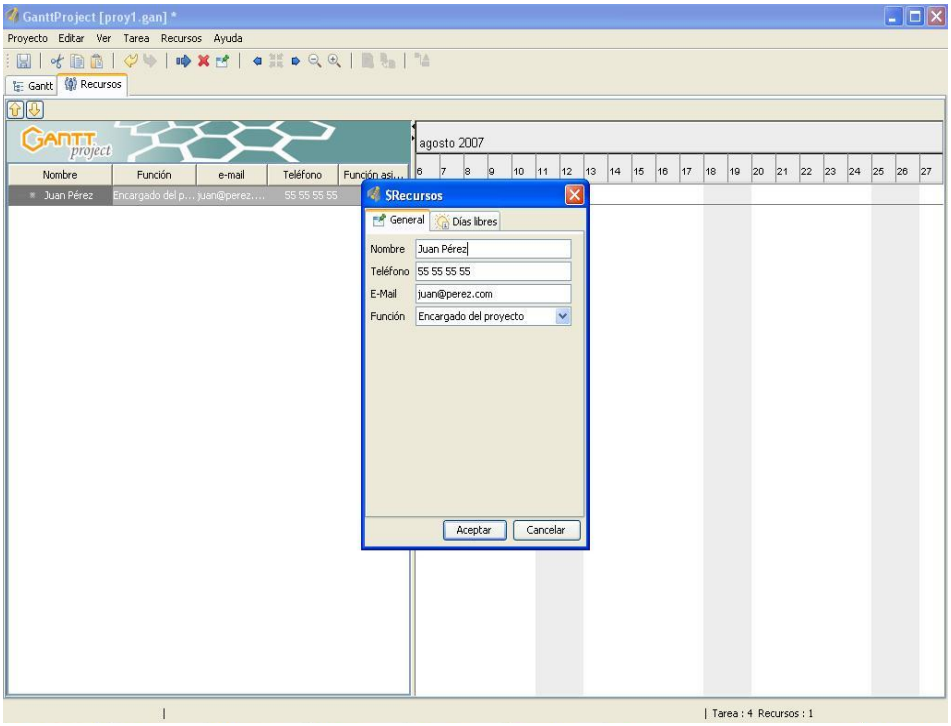

Figura 2.14: Nuevo Recurso.

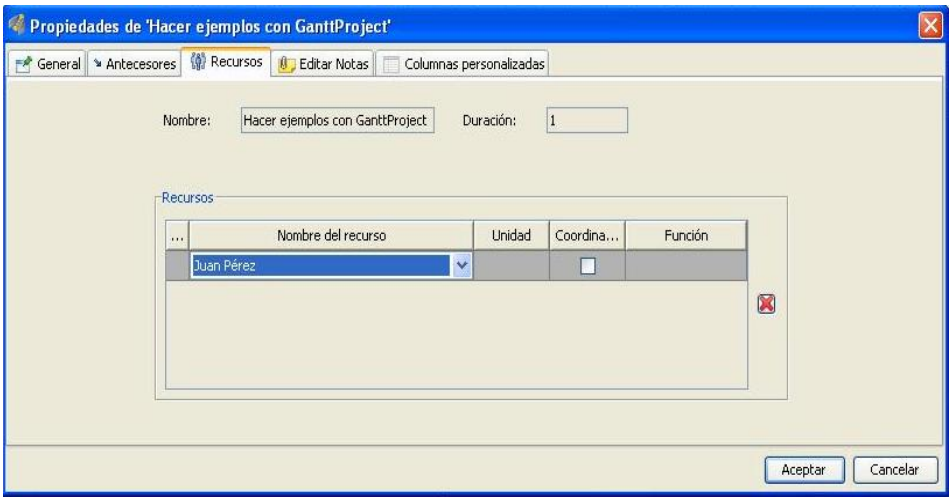

Figura 2.15: Asignación del nuevo recurso a la tarea.

mucha memoria RAM.

ArgoUML soporta diversos tipos de diagramas de UML:

1. *Diagrama de Casos de Uso*: modela las funcionalidades del sistema describiendo las acciones, tanto en el caso del flujo normal de eventos, como las acciones a tomar en caso de eventos excepcionales.

2. *Diagrama de Clases*: representa el conjunto de clases que participan en el proyecto, se pueden organizar en paquetes y se pueden modelar las relaciones entre ellas, como la herencia.

3. *Diagrama de Secuencia*: muestra la trayectoria de las actividades a realizar por separado dentro de un proyecto, enfatizando la interacción mediante mensajes.

4. *Diagrama de Colaboración*: muestra la interacción entre los objetos involucrados en el proyecto, resaltando la organización estructural.

5. *Diagrama de Estados*: es una gráfica con estados y transiciones entre ellos. Generalmente incluye un nodo de inicio y uno o varios de término. Por ejemplo, muestra la navegación dentro de un sistema en Internet, donde cada nodo representa una página.

6. *Diagrama de Actividades*: simplifica el Diagrama de Estados modelando el comportamiento mediante flujos de actividades.

7. *Diagrama de Despliegue*: muestra un conjunto de nodos con su función correspondiente y sus relaciones.

2.3.1. Origen de ArgoUML

Durante la década de los 80s, varias metodologías y notaciones en el Análisis y Diseño Orientado a Objetos fueron desarrolladas por algunos equipos de trabajo e investigación. Durante la siguiente década se hizo una unificación de éstas, surgiendo el estándar conocido como UML (Lenguaje de Modelado Unificado).

ArgoUML fue concebido como una herramienta y un ambiente para el análisis y diseño de Sistemas de Software Orientados a Objetos por Jason Robbins y su equipo de colaboradores en la Universidad de California.

El primer lanzamiento de ArgoUML estuvo disponible en 1998 y más de 100 000 descargas hasta mediados de 2001 mostraron el impacto de este proyecto.

2.3.2. Instalación de ArgoUML

- **ArgoUML** está hecho en Java, así que, para empezar, debes tener instalada alguna versión del JRE (Java Runtime Environment) en tu computadora.
- Visita la página **http://argouml.tigris.org/** y descarga la última versión disponible. Al momento de hacer esta guía, es la **0.24**.

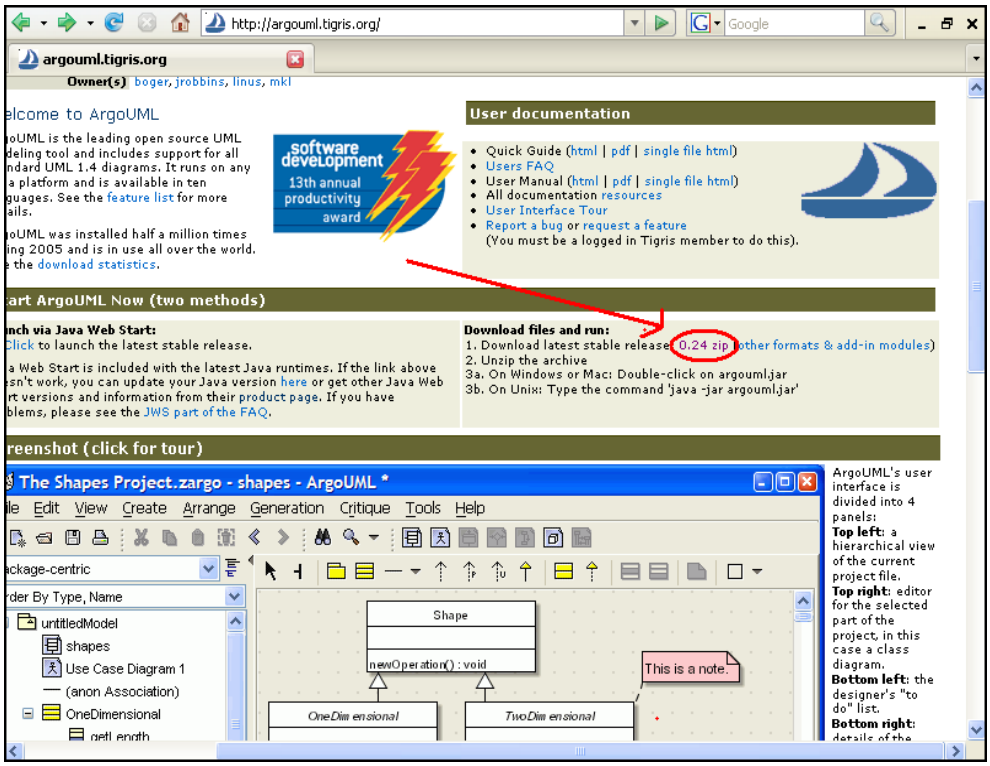

Figura 2.16: Página de ArgoUML.

- Descomprime el archivo descargado.
- En tu directorio **ArgoUML** recién creado, da *doble click* en el archivo ejecutable **argouml.jar**.
- Aparece una ventana como la siguiente (Fig. 2.17):

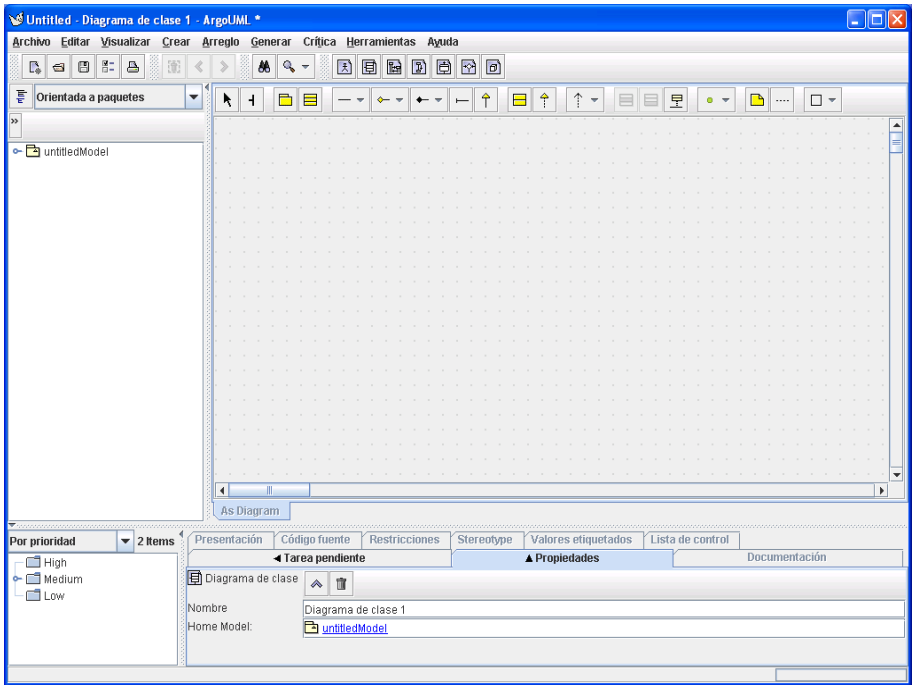

Figura 2.17: Ventana Principal de ArgoUML.

2.3.3. Uso de ArgoUML.

Veamos un ejemplo de uso para el caso de los *Diagramas de Clases*.

- Selecciona el tipo de diagrama **Diagrama de clase**, en la parte superior de la ventana de ArgoUML puedes ver esta barra con los distintos tipos de diagrama soportados por **ArgoUML** (Fig. 2.18).

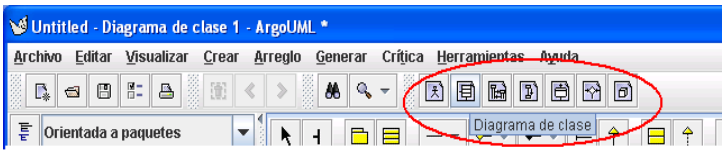

Figura 2.18: Barra de selección del tipo de diagrama.

- Automáticamente se actualiza la barra inferior con los componentes del tipo de diagrama seleccionado, tales como *paquetes, clases, asociaciones*, etc.
  Supongamos que queremos modelar los *actores* involucrados en una escuela.

  - Básicamente éstos serían los profesores y los alumnos.
  - Podemos generalizar los dos grupos en una clase llamada *Persona*, de la que las otras dos heredarán.
  - Seleccionamos **Paquete nuevo** y damos click sobre el área de trabajo. En la parte inferior podemos dar nombres y añadir características a los elementos del modelo (atributos, métodos, visibilidad, etc), así como generar código fuente (al menos el *esqueleto*) de las clases modeladas. Llamaremos *Escuela* al paquete (Fig. 2.19).

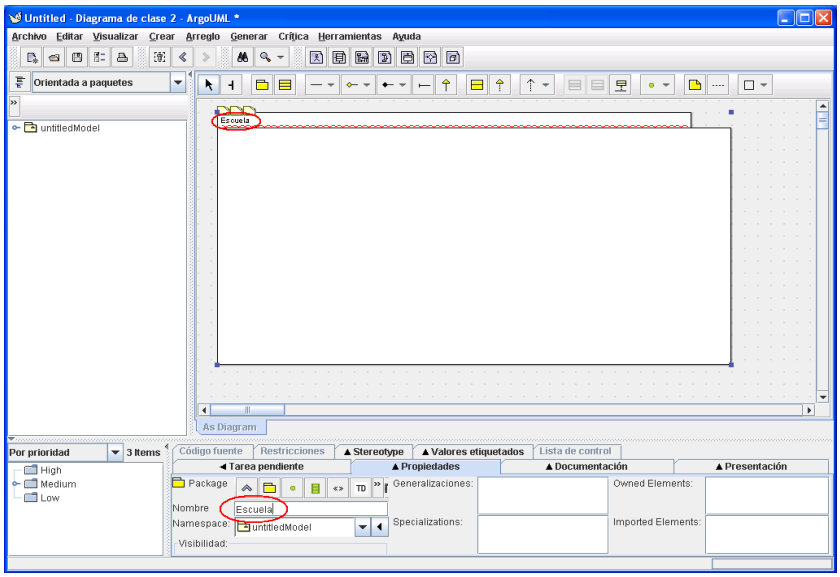

Figura 2.19: Paquete Escuela

  - La *Clase padre Persona* tendrá los atributos *nombre, direccion, telefono y fechaNacimiento*, cuyos tipos son *String, String, int y Date*, respectivamente. Además tendrá los métodos *darDeAlta() y darDeBaja()* (Fig. 2.20).
  - Mediante las flechas de herencia, *asociamos* las clases *Profesor* y *Alumno* con la clase *Persona*. Vemos que del lado izquierdo de la pantalla se va formando el árbol con la estructura de nuestro proyecto (Fig. 2.21).
  - Podemos generar la estructura básica del paquete que hemos creado yendo a **Generar → Generar las clases** (Fig. 2.22).

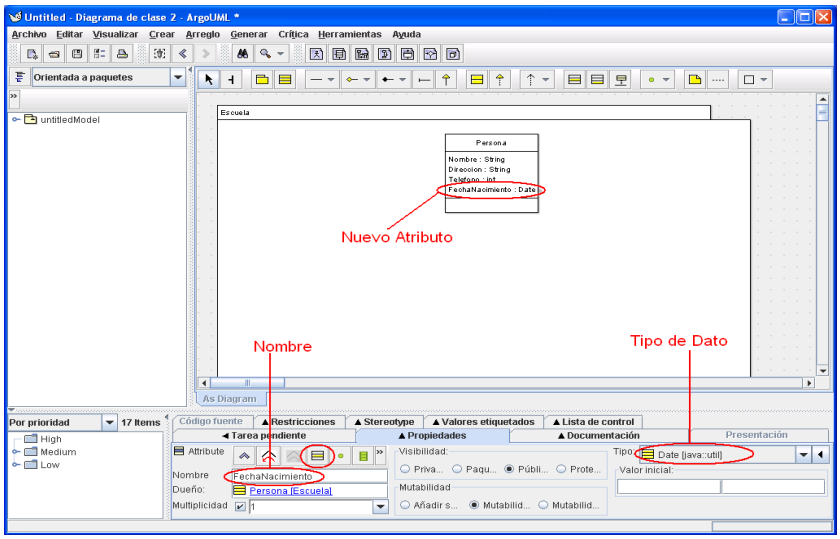

Figura 2.20: Definición de atributos y métodos

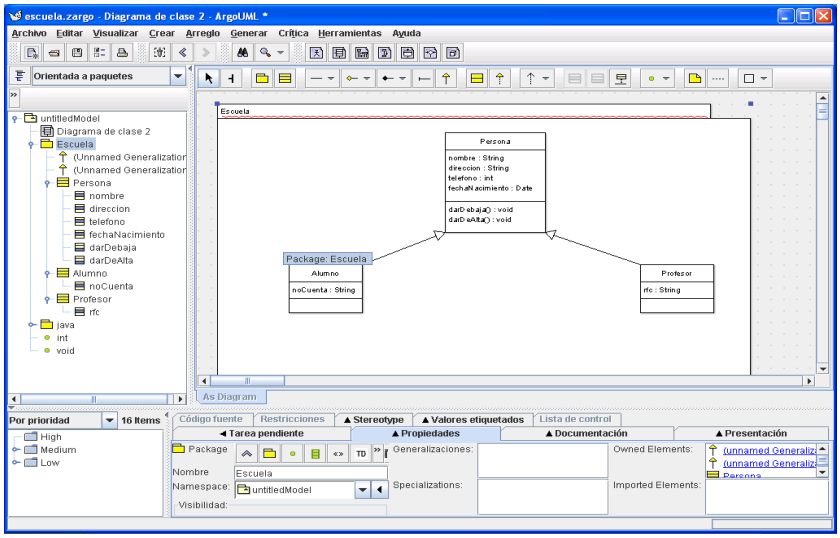

Figura 2.21: Estructura del paquete “Escuela”

- Y se genera este código:
  - Para la clase Persona:

```
package Escuela;

import java.util.Date;

public class Persona {

    public String Nombre;
    public String Direccion;
```

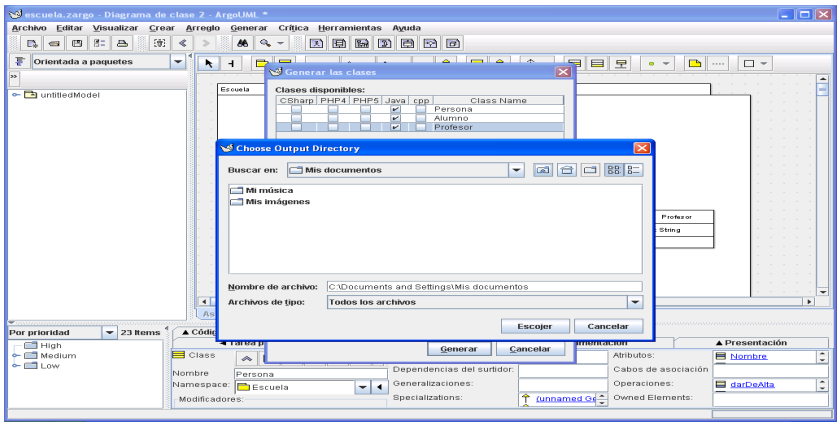

Figura 2.22: Generación de código fuente

```
    public int Telefono;
    public Date FechaNacimiento;

    public void darDeAlta() {
    }

    public void darDeBaja() {
    }
}
```

- Para la clase Alumno:

```
package Escuela;

public class Alumno extends Persona {

  public String noCuenta;
}
```

- Para la clase Profesor:

```
package Escuela;

public class Profesor extends Persona {

  public String rfc;
}
```

ArgoUML te permite, además del diagrama con extensión *.zargo*, guardar el diagrama como un gráfico en distintos formatos (.png, .gif, .svg, .ps o .eps). Ve a *Archivo → Guardar los gráficos.*

ArgoUML mostró ser una herramienta sencilla pero eficaz en la generación de diagramas UML. Quizá tiene algunos puntos negativos como el hecho de carecer de la opción *Undo*, por lo que no es fácil reparar errores. Otro aspecto negativo es que a veces los botones de opciones no funcionan adecuadamente, para lo que es necesario guardar el proyecto y volverlo a abrir.

Ojalá esta breve guía te haya sido de ayuda. Suerte.

2.4. Herramientas para Diseño y Construcción de Bases de Datos

2.4.1. MySQL

MySQL es un sistema de gestión de base de datos, multihilo y multiusuario (puede ser utilizado por varios usuarios y haciendo varias tareas a la vez). La empresa MySQL AB desarrolla MySQL en dos distribuciones:

- Bajo la Licencia GNU GPL.
- Puede vender la licencia que permita a la empresa solicitante, incorporarlo en productos privativos; incluye soporte técnico.

2.4.1.1. Origen de MySQL

La empresa MySQL AB fue fundada por David Axmark, Allan Larsson, y Michael Widenius en 1995 en Suecia, con el objetivo de que MySQL cumpliera con el estándar de SQL, establecido por la ANSI [1], además de asegurar velocidad y fiabilidad.

Al principio, la empresa MySQL AB intentaba gestionar sus propias tablas con mSQL, un gestor de Bases de Datos que había en ese momento, pero la velocidad y fiabilidad dada no les satisfizo, por lo que decidieron hacerle mejoras y posteriormente desarrollar su propio gestor.

El origen del nombre tiene dos versiones: la primera indica que, en los últimos años, todas las bibliotecas de esta empresa han llevado el prefijo *My*; mientras que la segunda dice que la hija de uno de los desarrolladores de MySQL se llama precisamente My.

En cuanto a la mascota, los desarrolladores eligieron un delfín, pues, según ellos, éste representa los valores de la compañía como son: rapidez, precisión, potencia y naturalidad. *Sakila* es su nombre, el cual se eligió mediante un concurso de convocatoria internacional.

[1] American National Standards Institute (Instituto Nacional Estadounidense de Estándares), es una organización que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas.

2.4.1.2. Instalación de MySQL.

1. Visita la página **www.mysql.com**. *Fig.* 2.23.

2. Descarga la última versión estable de MySQL (Al momento de realizar esta guía es la 5.0.41).

3. Ejecuta el archivo dando *doble click* sobre el ícono. *Fig.* 2.24.

4. Selecciona el tipo de instalación (típica o completa), el puerto donde escuchará, el número de conexiones a soportar, el conjunto de caracteres a soportar, etc. *Fig.* 2.25.

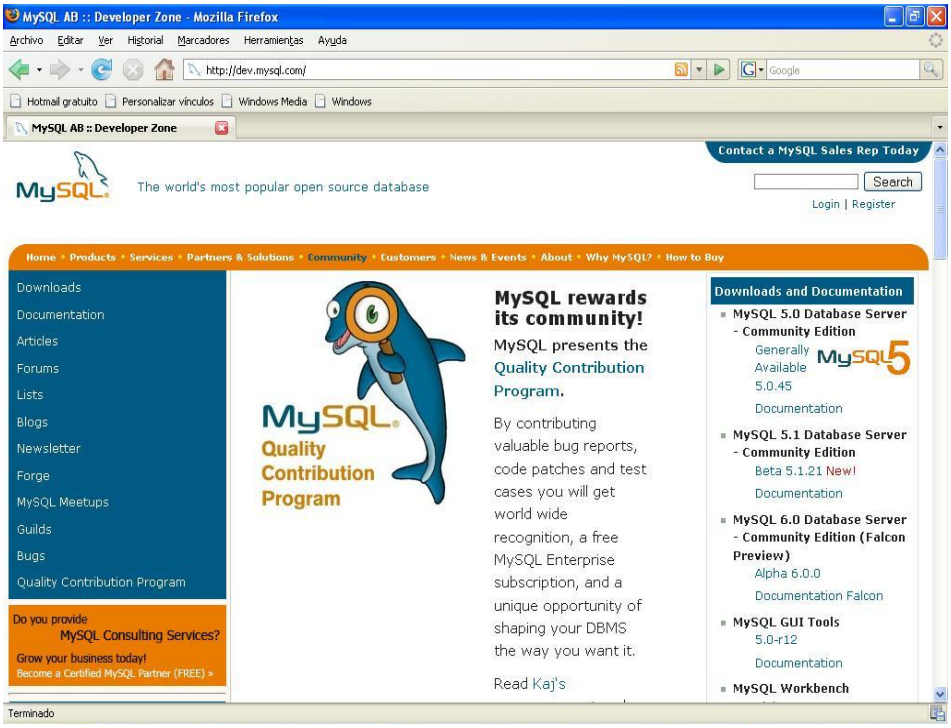

Figura 2.23: Página principal de MySQL.

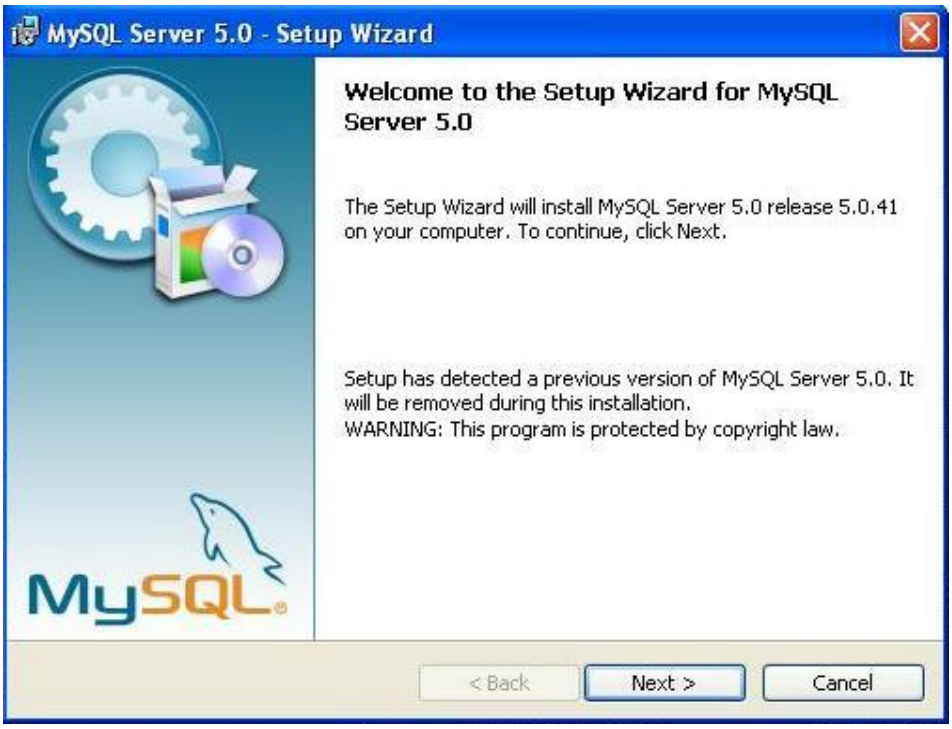

Figura 2.24: Instalación de MySQL.

Figura 2.25: Configuración del servidor MySQL.

2.4.1.3. Ejemplo de Uso de MySQL

1. Desde una ventana de *Símbolo del Sistema* escribe

   ```
   mysql -u root -p
   ```

2. A continuación solicita el password que proporcionaste al momento de la instalación.

3. Ya puedes utilizar las sentencias de SQL.

Sin embargo, existe la posibilidad de hacer uso de MySQL desde Java mediante un conector JDBC (Java DataBase Connectivity).

1. Descarga el archivo .zip en la dirección **http://dev.mysql.com/downloads/connector/j/5.0.html**.

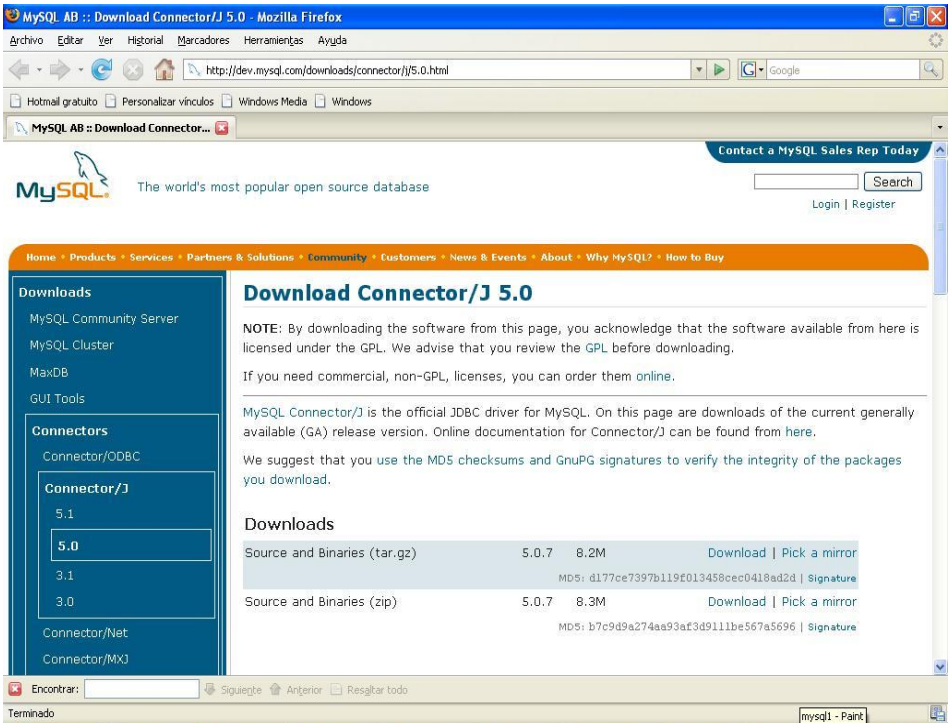

Figura 2.26: Descarga del conector JDBC.

2. Descomprime el archivo en la ruta deseada.

3. En la carpeta creada, ubica el archivo ejecutable *.jar*.

4. Agrega este archivo con su ruta completa al CLASSPATH (Escritorio → Mi PC → *click derecho* → Propiedades → Opciones Avanzadas → Variables de Entorno).

5. Se presenta el siguiente código a manera de ejemplo:

```
import java.sql.*;

public class Prueba{

    String bDatos;
    String servidor;
    String user;
    String password;
    String driver;
    Connection conexion;

    public Prueba(){
        bDatos = "Prueba";
        servidor = "localhost";
        user = "root";
        password = "admin";
        driver = "com.mysql.jdbc.Driver";
    }

    public void conectar() throws SQLException{
        try{
            Class.forName(driver);
            System.out.println("El driver se ha cargado con exito");
            conexion = DriverManager.getConnection("jdbc:mysql://" +
                servidor + ":3306/" + bDatos, user, password);
            System.out.println("La conexion con la Base de Datos " +
                                "se ha realizado con exito");
        }catch(Exception e){
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void main(String args[]){
        try{
            Prueba prueba = new Prueba();
            prueba.conectar();
        }catch(Exception e){
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Primero se declararon los datos que se necesitan para la conexión como el usuario y el password para acceder a MySQL, como variables de tipo String.

A continuación se declara la variable de tipo Connection que guardará la referencia a la conexión realizada, para ser utilizada posteriormente.

6. Si se ha cargado correctamente el conector JDBC, el resultado de ejecutar el programa es:

```
> java Prueba
El driver se ha cargado con exito
La conexion con la Base de Datos se ha realizado con exito
```

Es importante que cada vez que abramos una conexión y se utilicen datos asociados a esa conexión, hagamos el correspondiente cierre, pues cada conexión consume recursos del equipo.

El método para hacer la desconexión es **.close**. Así, se puede construir un método como el siguiente:

```
public void desconecta() throws SQLException{
    try{
        conexion.close();
    }catch(SQLException e){
        System.out.println("SQLException: " + e.getMessage());
    }
}
```

MySQL es un manejador de bases de datos muy confiable, sobre todo en el ámbito de aplicaciones web. Una vez que se ha instalado y configurado, así como establecido la conexión entre Java y MySQL a través del conector JDBC, el acceso a la base de datos para inserción de filas, consultas, etc., es muy sencillo. Muchos consideran que el rendimiento de MySQL no es el óptimo con gran número de actualizaciones concurrentes. Hasta el momento no he tenido oportunidad de verificar este comentario.

Ojalá este pequeño tutorial te haya sido de ayuda. Suerte.

2.4.2. PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido desarrollado de varias formas desde 1977.

2.4.2.1. Origen de PostgreSQL

PostgreSQL comenzó como un proyecto estudiantil llamado Ingres, en la Universidad de Berkeley, California. Posteriormente fue adoptado por Michael Stonebraker para crear un sistema de bases de datos objeto-relacionales llamado Postgres. En 1996 Postgres fue renombrado a PostgreSQL con una muy incrementada funcionalidad. Actualmente, el proyecto sigue en continuo desarrollo a nivel mundial.

2.4.2.2. Instalación de PostgreSQL

- Descarga la última versión estable del instalador de Postgres para Windows de la página: **http://www.postgresql.org/ftp/binary/v8.3.0/win32/** (al momento de hacer esta guía es la 8.3).
- Descomprime el archivo *.zip*.
- Localiza el archivo ejecutable (no el que incluye *int* en el nombre) e inicia la instalación con doble click (Fig. 2.27).

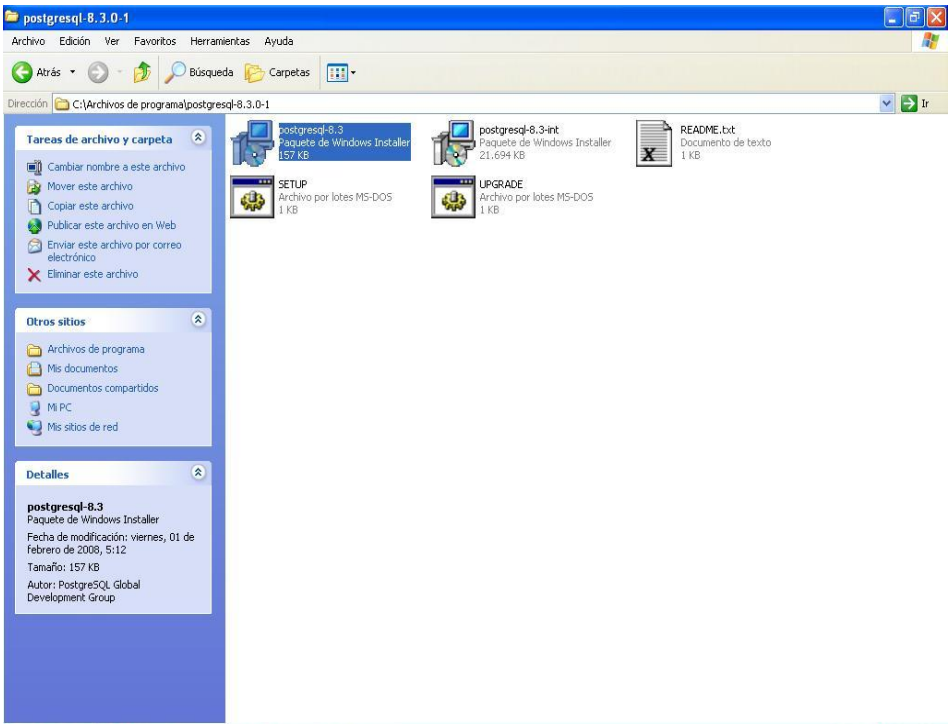

Figura 2.27: Archivo para instalar Postgres.

- Antes de comenzar la instalación se configuran ciertos valores. El nombre para la cuenta por default es **postgres**, puedes proporcionar un password o dejar el espacio en blanco para que se genere uno automático (Fig. 2.28). Esta contraseña será necesaria sólo en caso de actualización.
- La siguiente contraseña que se solicita deberá proporcionarse cada vez que se desee conectarse con el servidor de Postgres, así que es ésta sí es necesario memorizarla, a diferencia de la anterior.
- Continúa con la instalación hasta terminar el proceso.

El paquete de instalación incluye la herramienta **PgAdmin III**, que es una interfaz gráfica para el manejo de Postgres, la cual permite generar nuevas bases de datos, tablas, columnas,

inserciones, consultas, etc.

Para probar la instalación, abre la ventana de pgAdmin (*Inicio→Programas→PostgresQL 8.3 →pgAdmin*).

- En el panel izquierdo, aparece el servidor Postgres, haz click derecho sobre éste y selecciona *Conectar*.
- Proporciona la contraseña que diste al instalar.
- En el panel derecho se muestran las propiedades del servidor (Fig. 2.29).

Adicionalmente, para tener acceso a Postgres desde Java, podemos descargar el driver JDBC desde: **http://www.postgresql.org/download** (Fig. 2.30).

- Descarga el archivo *.jar* en una nueva carpeta, por ejemplo **C:/Archivos de programa/PostgreSQL/pgjdbc** y añade la ruta completa hasta este archivo al classpath.
- En el caso de una aplicación web, copia el driver en la carpeta *lib* de tu aplicación.

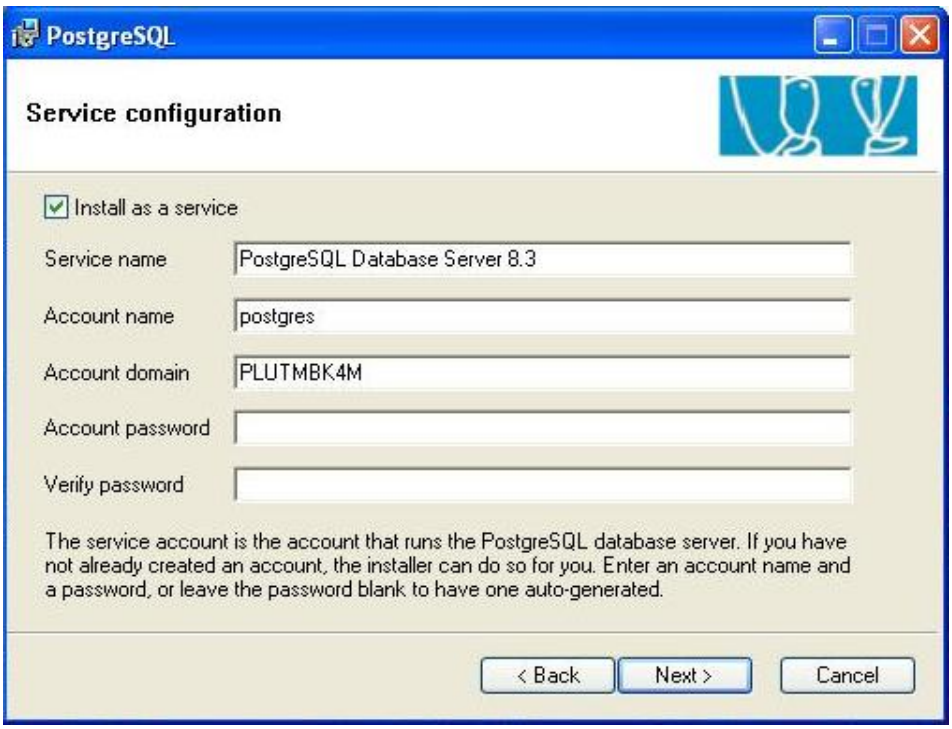

Figura 2.28: Configuración previa.

2.4.2.3. Ejemplo de Uso de PostgreSQL

Veamos un ejemplo de uso de Postgres a través de PgAdmin III.

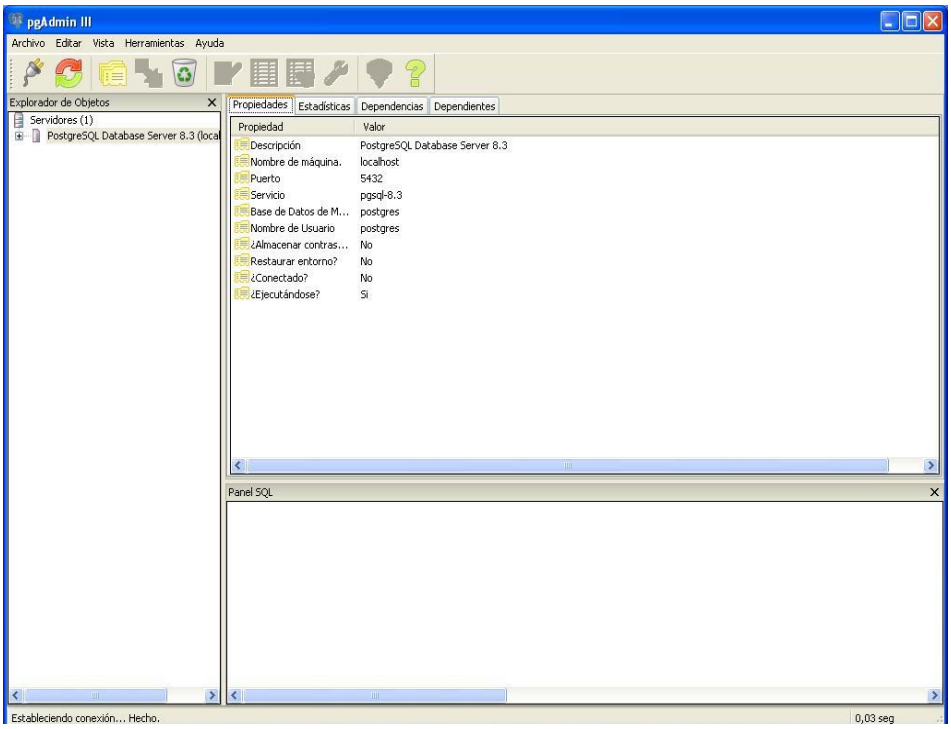

Figura 2.29: Propiedades del servidor.

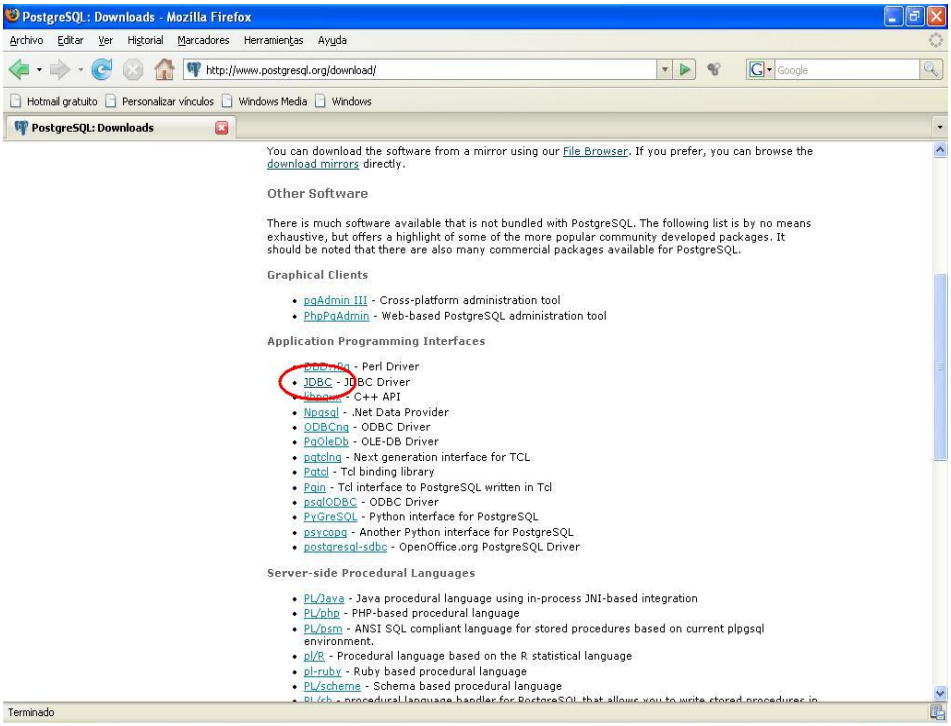

Figura 2.30: Descarga del driver JDBC.

- Ve a Inicio → Programas → PostgreSQL 8.3 → pgAdmin III.
- En el panel izquierdo aparece el servidor de Postgres, selecciona *conectar* y proporciona la contraseña que diste en la instalación.
- Da click derecho en el ícono de *Base de Datos* y selecciona *Nueva Base de Datos* (Fig. 2.31).
- Asigna datos como el nombre, la codificación de caracteres, comentarios, etc. (Fig. 2.32)
- En el árbol del panel izquierdo, selecciona *Esquemas* y da click derecho en *Tablas*, para crear una nueva tabla.
- En el panel superior derecho, aparecen las tablas. Da click derecho sobre la tabla para crear columnas (Fig. 2.33).
- Para insertar una fila, da click derecho sobre la tabla en el panel izquierdo, selecciona *scripts* → *sentencia INSERT* (Fig. 2.34).
- Selecciona *Consulta* → *ejecutar consulta*.
- De la misma manera para hacer una consulta (Fig. 2.35).

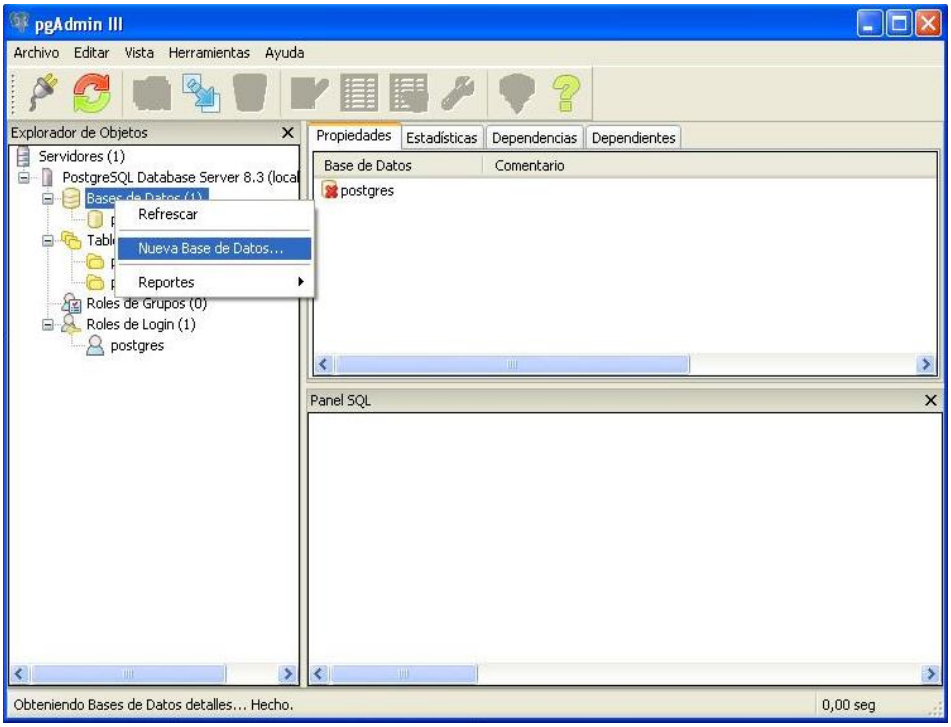

Figura 2.31: Nueva base de datos.

Ahora probaremos el conector JDBC, haremos una pequeña aplicación en Eclipse (Referencia: Tutorial de Eclipse).

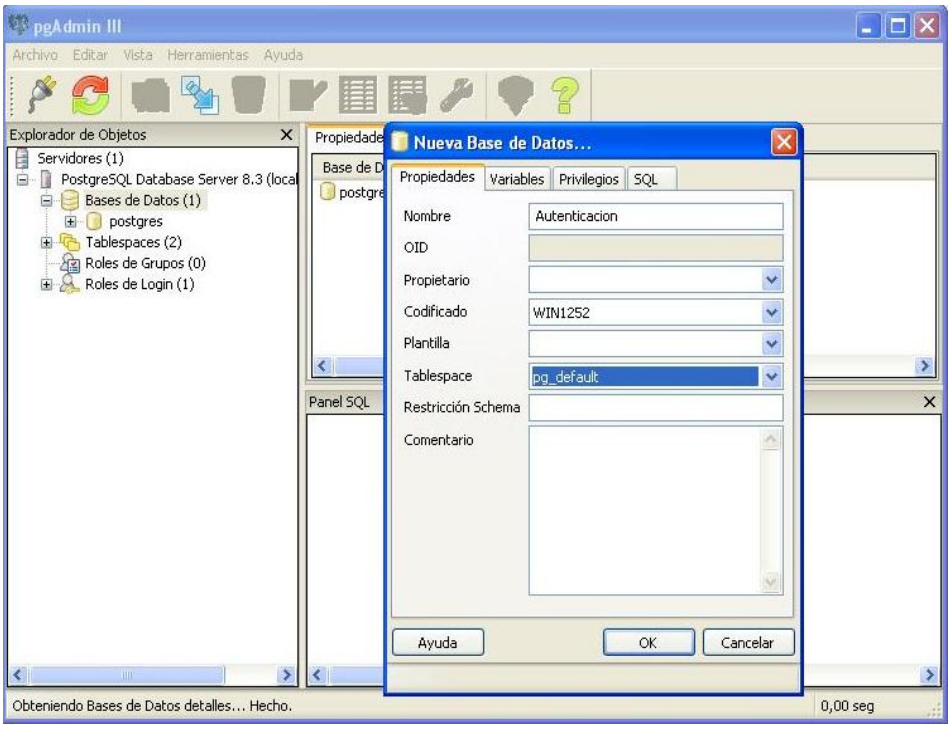

Figura 2.32: Asignando valores a la nueva base.

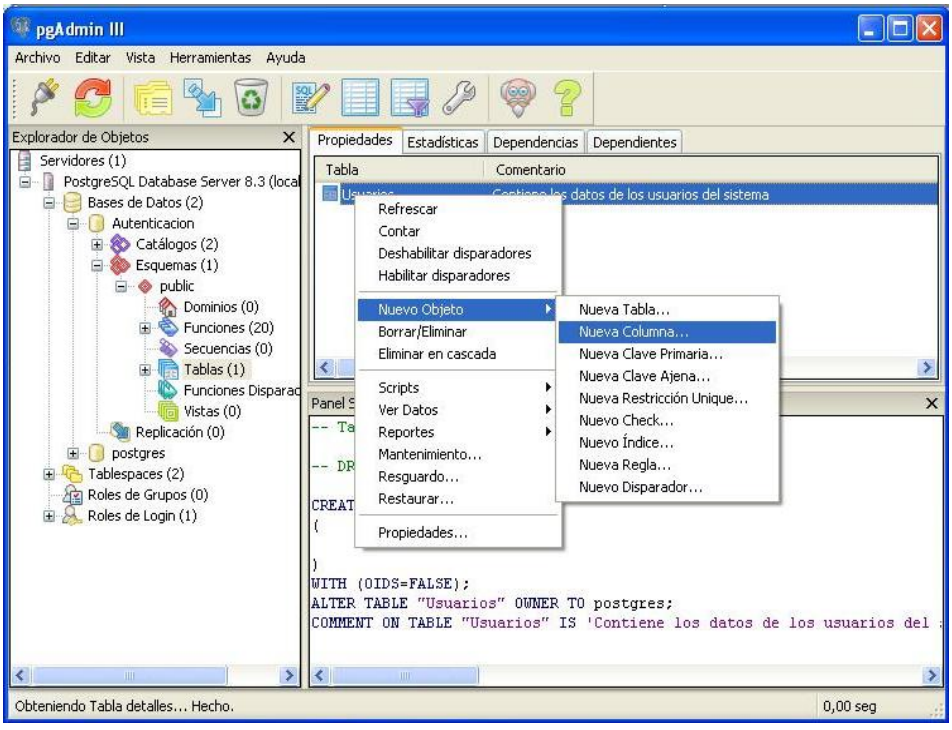

Figura 2.33: Nueva columna.

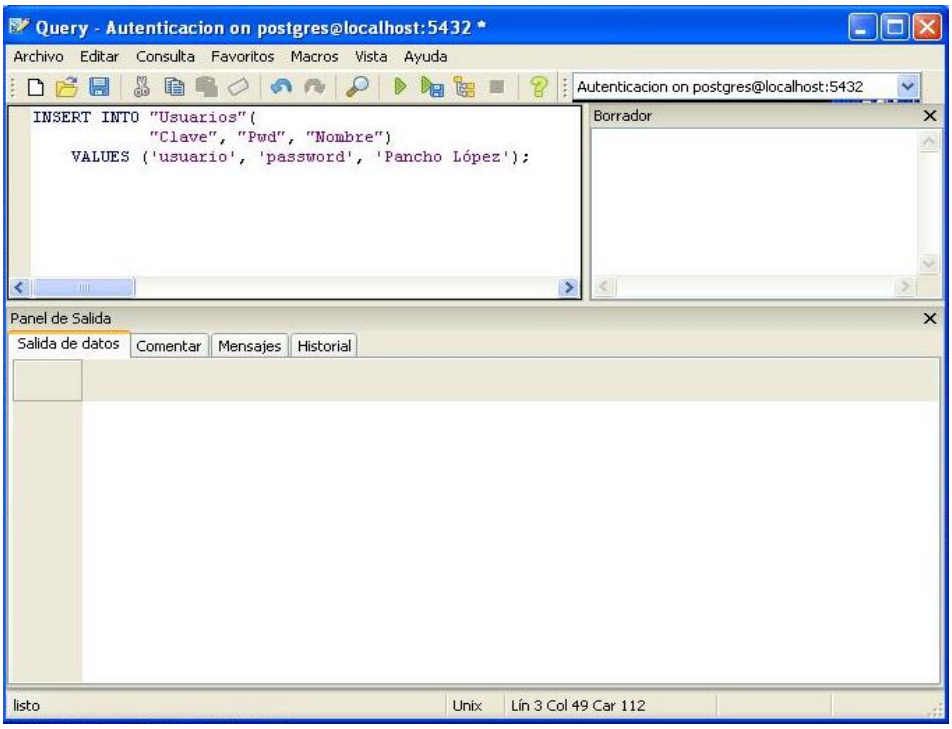

Figura 2.34: Insertar fila.

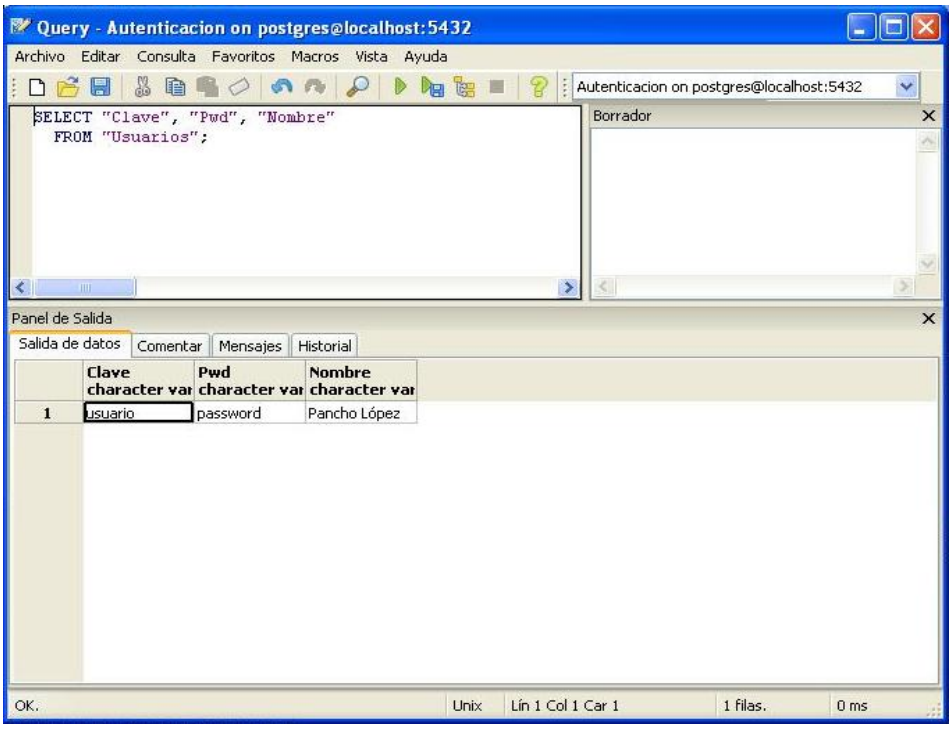

Figura 2.35: Consulta.

- Crea un nuevo proyecto Java en Eclipse.
- Crea una nueva clase para hacer una conexión mediante el conector JDBC, con el siguiente código.

```
import java.sql.*;

/**
 * @author CursoIS
 *
 */

public class ConexionBD {

    private String bd;
    private String servidor;
    private String usuario;
    private String password;
    private String driver;
    private Connection con;

    public ConexionBD() {
        this.bd = "Autenticacion";
        this.servidor = "localhost";
        this.usuario="postgres";
        this.password="76819104";
        this.driver="org.postgresql.Driver";
    }

    public void conectar(){
        try{
            Class.forName(driver);
            System.out.println("El driver se ha cargado con exito");
            this.con = DriverManager.getConnection("jdbc:postgresql://" +
            this.servidor + ":5432/" + this.bd, usuario, password);
            System.out.println(" se ha realizado la conexión" +
            " satisfactoriamente");
        }catch(Exception e){System.out.println("Error: " + e.getMessage());}
    }

    public static void main(String[] args) {
        ConexionBD cbd = new ConexionBD();
        cbd.conectar();
    }
}
```

- No olvides agregar el conector como *archivo externo .jar* en Eclipse.

- Ejecuta como aplicación Java (Fig. 2.36).

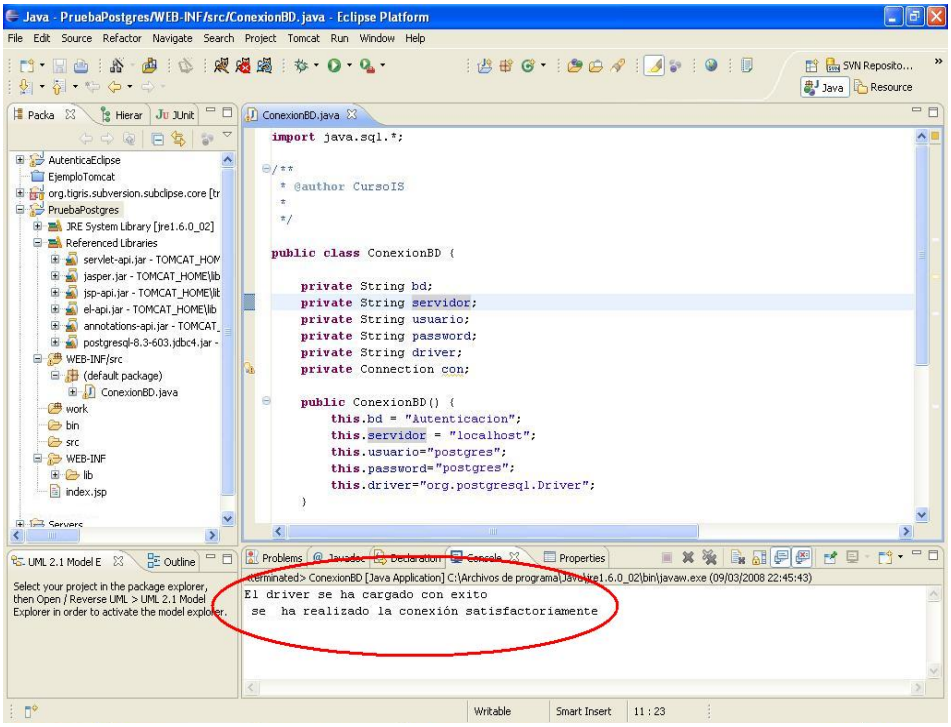

Figura 2.36: Ejecución de la clase.

PostgreSQL es un manejador de bases de datos muy reconocido por su alta concurrencia y confiabilidad. El hecho de venir acompañado por el administrador *PGAdmin*, hace aún más sencillo las actividades de generación de tablas, inserción de datos, consultas, etc. Éste ha resultado un complemento ideal.

Ojalá esta breve guía te haya sido de ayuda. Suerte.

2.4.3. Herramienta para Diseño de Bases de Datos

DB Designer Fork es una herramienta para el diseño de bases de datos, muy fácil de usar y que genera código SQL para distintos manejadores como MySQL y Oracle. Es de distribución gratuita.

2.4.3.1. Instalación de DB Designer Fork

- Descarga DB Designer Fork desde: **http://sourceforge.net/** y en el buscador teclea "DB Designer Fork")
- Descomprime el archivo.
- En la carpeta *bin* se encuentra el archivo ejecutable.

2.4.3.2. Ejemplo de uso de DB Designer Fork

Vamos a ver de nuevo el ejemplo de la escuela como en el caso del tutorial de ArgoUML. Se manejan tanto **Alumnos** como **Profesores**, ambos son generalizadas en la tabla **Persona**.

La tabla **Persona** tiene los atributos:

- idPersona
- Nombre
- Direccion
- Telefono
- FechaNacimiento

La tabla **Alumno**:

- noCuenta (número de cuenta)

La tabla **Profesor**:

- Rfc (RFC como trabajador)

Veamos la construcción:

- De la barra de herramientas del lado izquierdo, selecciona *New Region*.
- Selecciona *New Table*. Proporciona el nombre y asigna las columnas con sus respectivos nombres y datatypes (Fig. 2.37).
- Crea la relación de generalización seleccionando *New Generalization* de la barra de herramientas izquierda. Mediante doble click en ésta puedes fijar propiedades (Fig. 2.38).
- Al finalizar con las tablas, para obtener el código SQL para MySQL, selecciona *Archivo* → *Export* →*SQL Create Script* (Fig. 2.39).
- En la opción *Target Data Base* selecciona *MySQL*.
- Salva el código en un archivo. El código obtenido es:

```
CREATE TABLE Persona (
  IdPersona VARCHAR(20)  NOT NULL  ,
  Nombre VARCHAR(20)  NOT NULL  ,
  Direccion VARCHAR(20)  NULL  ,
  Telefono INTEGER  NULL  ,
  FechaNacimiento DATE  NULL    ,
PRIMARY KEY(IdPersona))
TYPE=InnoDB;
```

```
CREATE TABLE Profesor (
  Rfc VARCHAR(20)  NOT NULL  ,
  Persona_IdPersona VARCHAR(20)  NOT NULL    ,
PRIMARY KEY(Rfc, Persona_IdPersona)  ,
INDEX Profesor_FKIndex1(Persona_IdPersona),
  FOREIGN KEY(Persona_IdPersona)
    REFERENCES Persona(IdPersona)
      ON DELETE CASCADE
      ON UPDATE CASCADE)
TYPE=InnoDB;


CREATE TABLE Alumno (
  noCuenta VARCHAR(20)  NOT NULL  ,
  Persona_IdPersona VARCHAR(20)  NOT NULL    ,
PRIMARY KEY(noCuenta, Persona_IdPersona)  ,
INDEX Alumno_FKIndex1(Persona_IdPersona),
  FOREIGN KEY(Persona_IdPersona)
    REFERENCES Persona(IdPersona)
      ON DELETE CASCADE
      ON UPDATE CASCADE)
TYPE=InnoDB;
```

El cual se puede ejecutar en la consola de MySQL.

- El esquema final es: (Fig. 2.40)

DB Designer Fork demostró ser una herramienta práctica y de manejo sencillo, aunque un tanto simple.

Ojalá esta breve guía te haya sido de ayuda. Suerte.

2.5. Herramienta para la Construcción (Tomcat).

Tomcat es un contenedor de aplicaciones web que soporta las tecnologías Java Servlets y Java Server Pages (JSP). Está diseñado por Jakarta, un proyecto patrocinado por Apache que construye software basado en Java, a través de la Apache Software Foundation.

Gracias a la característica de estar escrito en Java, Tomcat funciona en cualquier plataforma que tenga instalada la máquina virtual de Java.

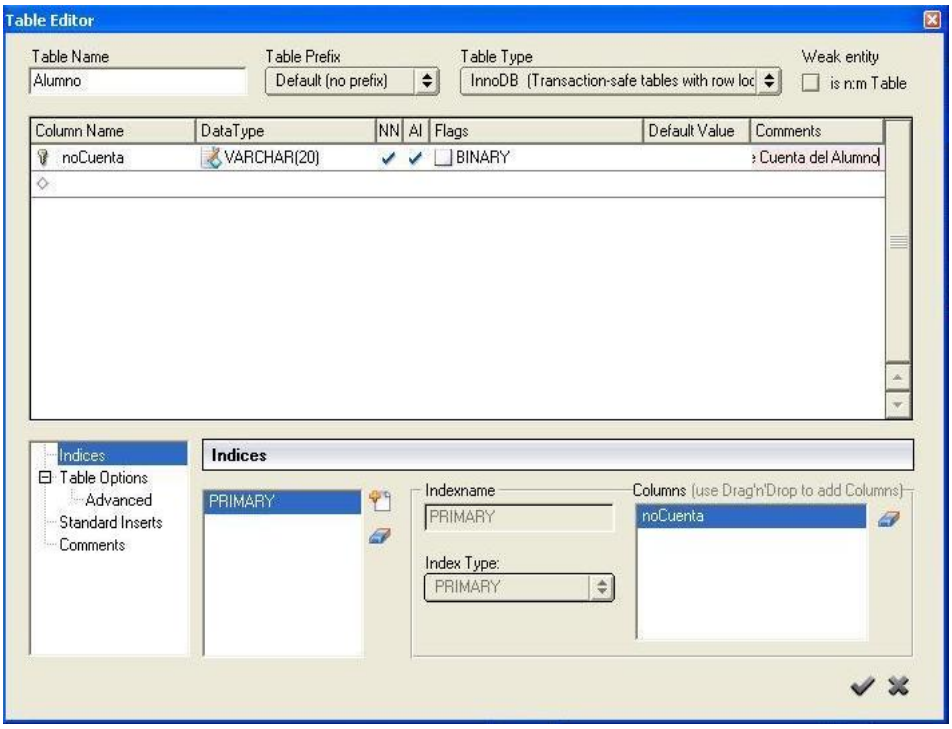

Figura 2.37: Tabla Alumno.

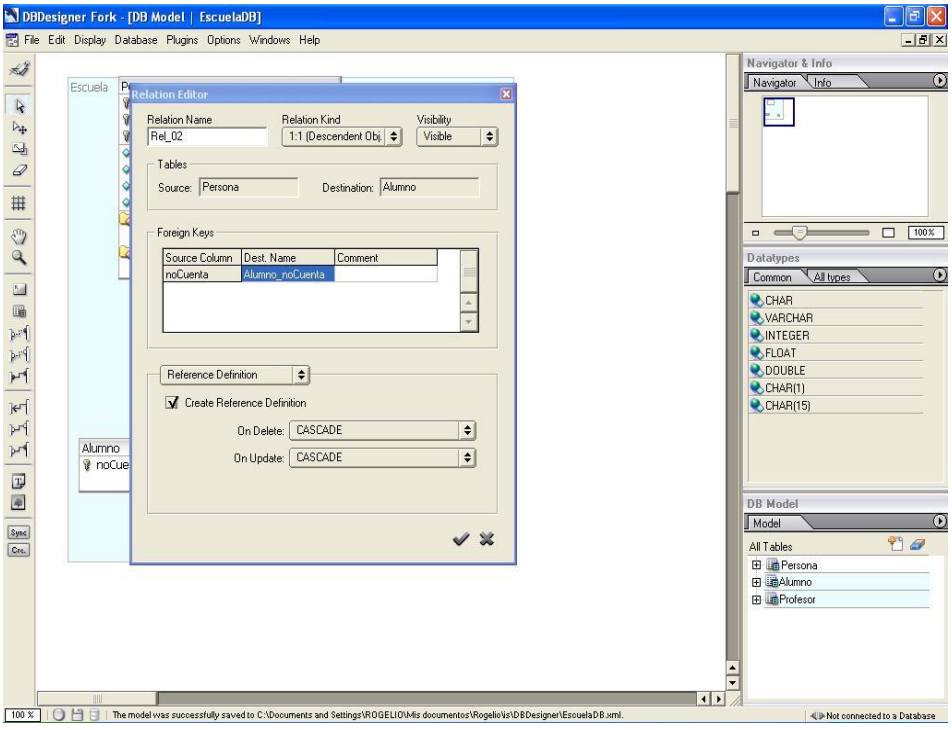

Figura 2.38: Nueva Generalización.

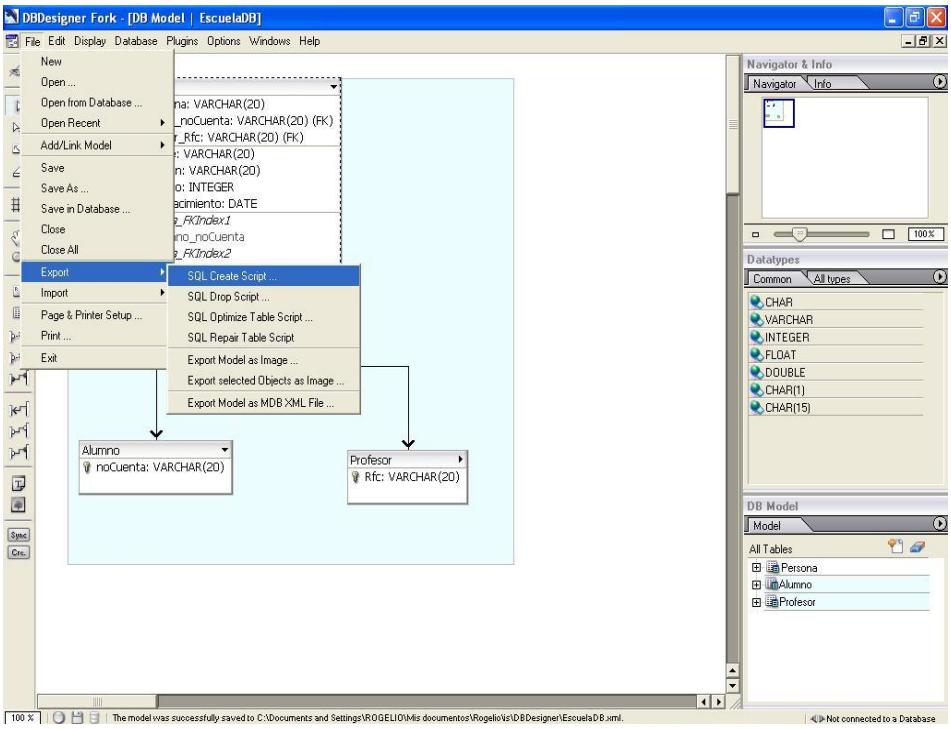

Figura 2.39: Obteniendo el código SQL.

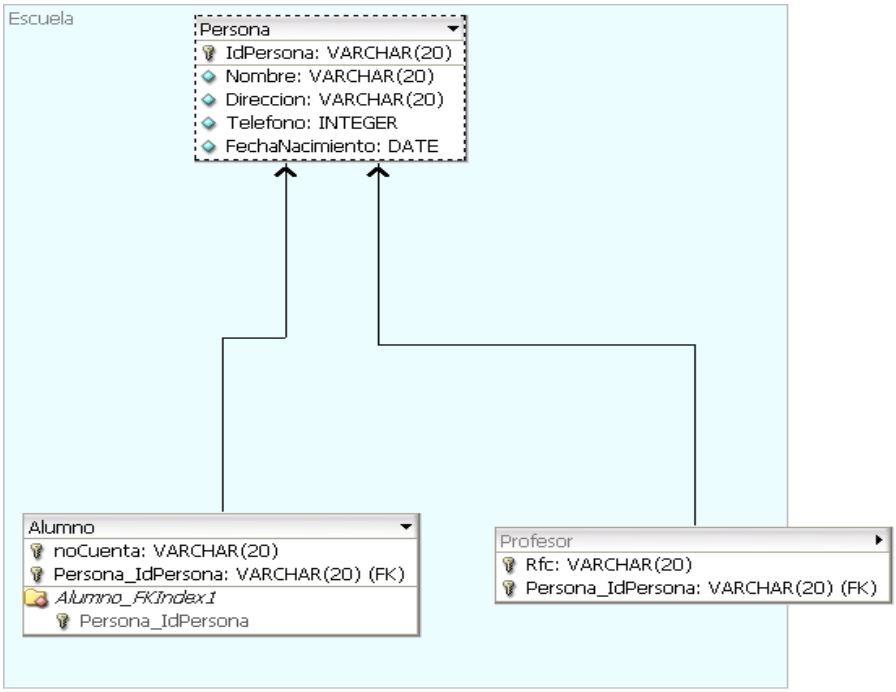

Figura 2.40: Esquema completo.

2.5.1. Origen de Tomcat

El proyecto lo inició James Duncan Davidson, buscando desarrollar una implementación de la especificación de los servlets. Con el tiempo, el producto fue donado a la Apache Software Foundation como *Open Source*.

El origen del nombre y la mascota provienen del hecho de tener que elegir un animal, como la mayoría de software *Open Source* tienen libros de la editorial O'Reilly con un animal asociado en la portada, y se eligió **Tomcat** (gato macho), para representar la capacidad de ser independiente.

2.5.2. Instalación de Tomcat.

1. Descarga la última versión de Apache Tomcat desde **http://tomcat.apache.org**.

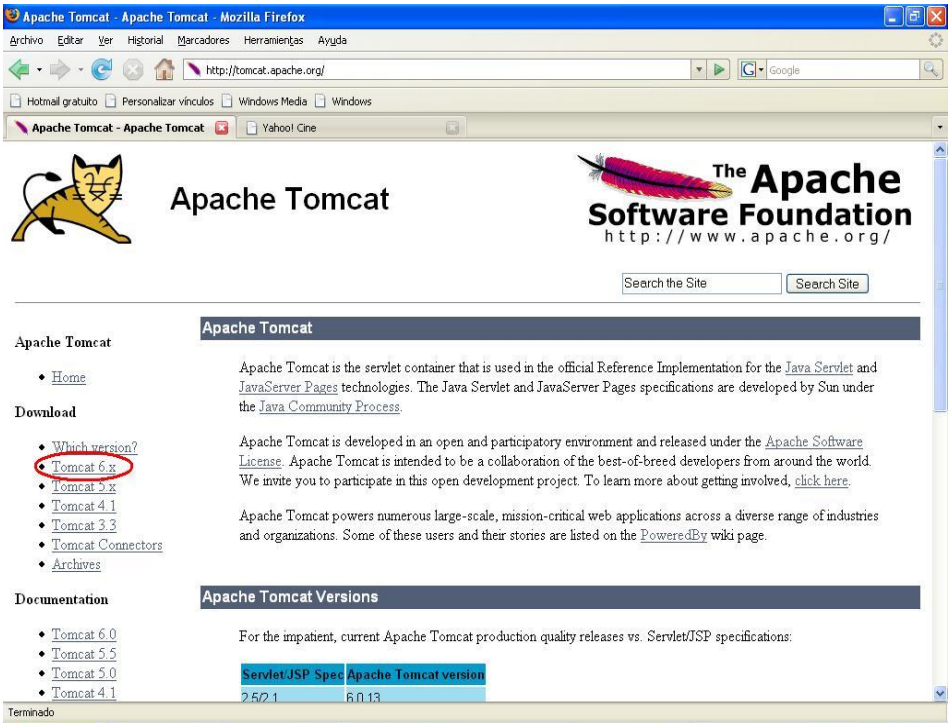

Figura 2.41: Página principal de Tomcat.

2. Inicia la instalación con *doble click* sobre el icono. Selecciona opciones como la ubicación, el puerto donde escuchará la aplicación (por default, el 8080), un nombre de usuario y un password (por default, admin). El instalador encontrará los JDK's disponibles en tu computadora para ejecutar los JSP's y Servlets.

3. Una vez terminado el proceso, ve a *Inicio* → *Ejecutar* y ve a la ubicación del archivo tomcatX.exe (donde X es la versión de tu archivo).

4. Abre una ventana del navegador y teclea la dirección **http://localhost:8080**.

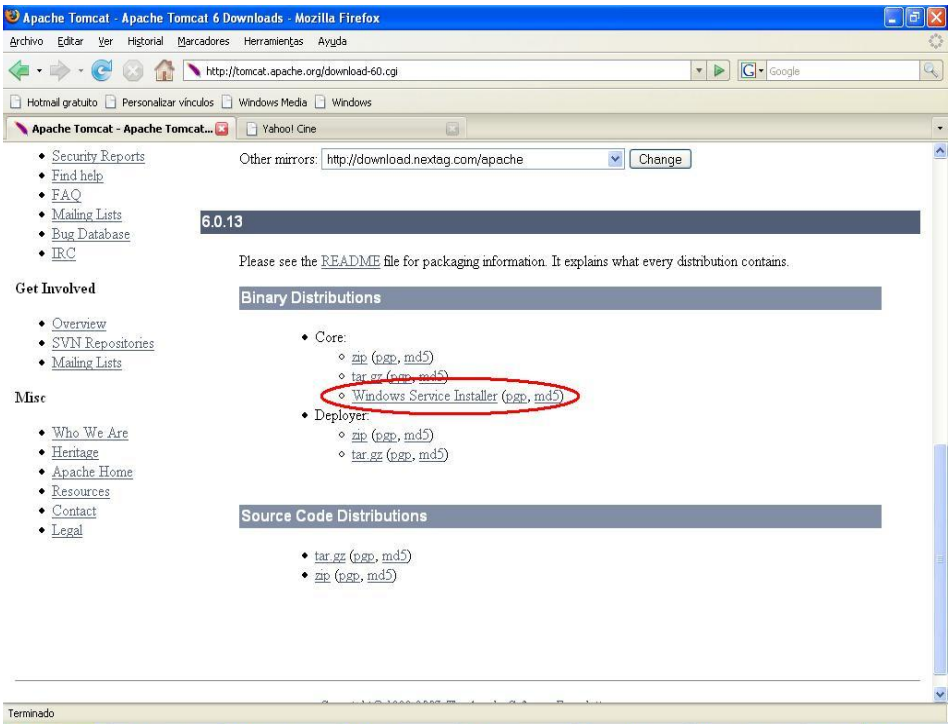

Figura 2.42: Descarga del Instalador.

5. Si se observa la documentación de Tomcat, es que se ha instalado correctamente, en otro caso, repetir los pasos anteriores.

2.5.2.1. Estructura de Directorios

Antes de dar un ejemplo, se muestra una breve descripción de la estructura de directorios que se crea al instalar Tomcat, la cual posee gran relevancia, como veremos.

- */bin:* Contiene los archivos de ejecución por lotes utilizados para arrancar y detener el servidor.
- */lib:* Contiene los archivos java (.jar) de los que depende Tomcat.
- */conf:* Contiene los archivos de configuración. El más importante es *server.xml*, donde se establece la configuración del propio servidor.
- */logs:* Contiene los archivos con mensajes que se van generando de lo que ocurre en las ejecuciones en el servidor.
- */work:* En este directorio, Tomcat coloca los servlets generados a partir de las páginas JSP.
- */webapps:* Contiene las aplicaciones web del desarrollador.

A continuación se muestra el desarrollo de una aplicación muy simple con el propósito de mostrar, principalmente la estructura básica para aplicaciones más grandes.

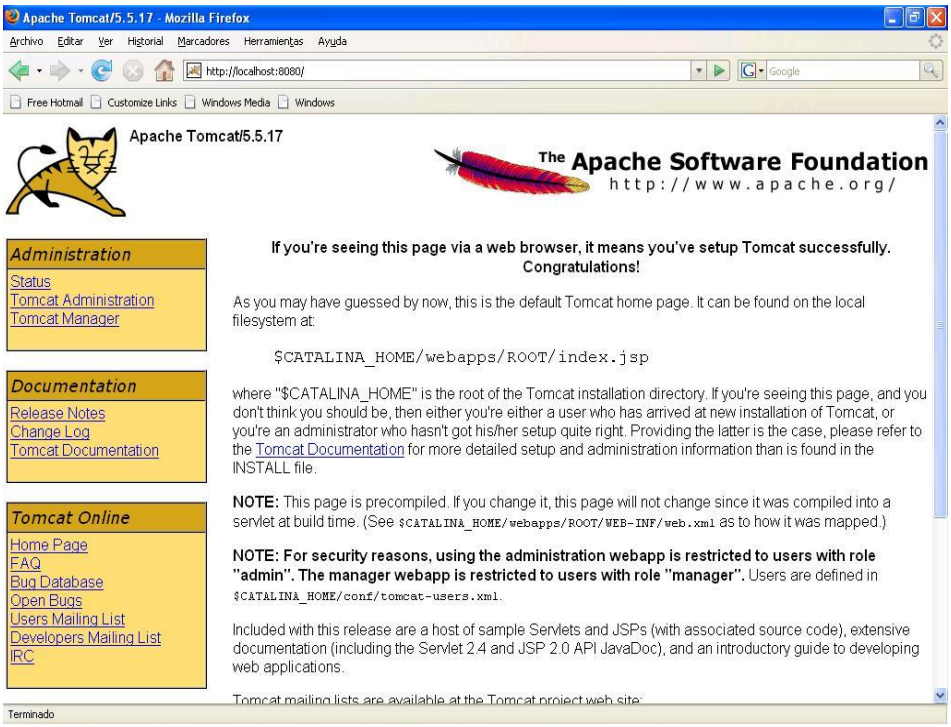

Figura 2.43: Pantalla de éxito en la instalación de Tomcat.

1. Primero hay que seguir una estructura estándar para el árbol de directorios de cada aplicación. Suponiendo que nuestra aplicación se llama *ejemploJSP*, crea la siguiente estructura de directorios, a partir de la carpeta *webapps* de Tomcat:

   - **/ejemploJSP**: Con el mismo nombre que nuestra aplicación. Alojará todos los recursos necesarios para la misma, en subcarpetas que contengan imágenes, páginas html, etc.
   - **/ejemploJSP/WEB-INF**: Contiene elementos Java, a través de otras carpetas como las siguientes.
   - **/ejemploJSP/WEB-INF/classes**: Aloja los archivos *.class* como resultado de las clases Java que ocupará nuestra aplicación.
   - **/ejemploJSP/WEB-INF/classes/lib**: Contiene archivos *.jar*, en particular, el conector JDBC.

2.5.3. Ejemplo de Uso de Tomcat.

Veamos el clásico ejemplo *Hola Mundo* para la programación JSP.

```
<html>
  <head>
    <title>Hola Mundo</title>
```

```
  </head>

  <body>
    <h1>Hola Mundo</h1>
    <%
    for ( int i = 0 ; i < 10 ; i ++ ){
    out.print( i + ": Hola Mundo!<br>");
    }
    %>
  </body>
</html>
```

2. Llama a este archivo hola_mundo.jsp y colócalo en la carpeta *ejemploJSP*.

3. Inicializa Tomcat y carga la página *http://localhost:8080/ejemploJSP/hola_mundo.jsp*.

4. El resultado se muestra en la figura 4.

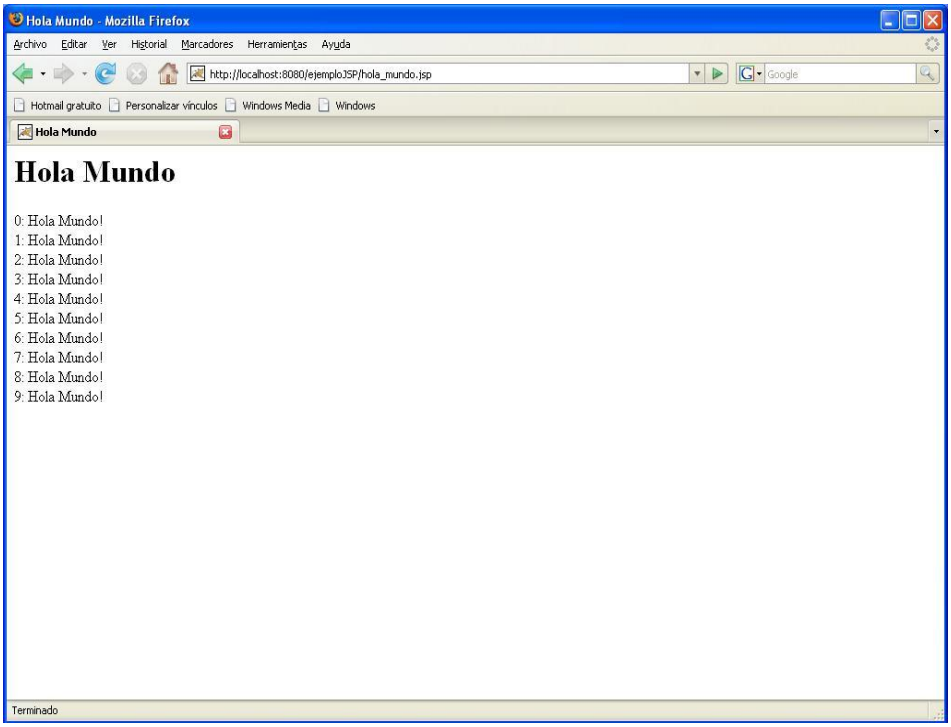

Figura 2.44: Aplicación "Hola Mundo".

En lo personal, la creación de este tipo de herramienta me parece una maravilla. El hecho de tener un servidor de aplicaciones que se encargue de ejecutar el código Java de acuerdo a las peticiones del cliente es digno de mención, además de manejarse en forma tan sencilla. A cambio sólo se pide una convención en la estructura de carpetas que no es nada del otro mundo y con un poco de práctica y con el apoyo de los IDEs se hace aún más fácil.

Ojalá esta breve guía te haya sido de ayuda. Suerte.

2.6. Herramienta para la Construcción (IDE NetBeans)

NetBeans es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés, Integrated Development Environment). Esto quiere decir que reúne muchas de las herramientas necesarias para el desarrollo de software, como son, un editor de texto con marcadores (el código se resalta con un color de acuerdo a la sintaxis), un compilador, un depurador, etc., y adicionalmente se pueden añadir características mediante módulos llamados *plugins*. Es posible ejecutarlo en múltiples plataformas

2.6.1. Origen de NetBeans

NetBeans comenzó como un proyecto estudiantil en República Checa en 1996. Originalmente fue llamado *Xelfi*. Tiempo después un empresario se interesó en el proyecto y lo apoyó económicamente. Jarda Tulach, quien diseñó la arquitectura original, propuso el nombre.

En 1999, Sun Microsystems se interesó en la herramienta y se llegó a un acuerdo. Al siguiente año, NetBeans pasó a ser *Open Source*, siendo así la primera herramienta de este tipo patrocinada por Sun. En junio de 2000 fue lanzado NetBeans.org.

2.6.2. Instalación de NetBeans.

- Visita la página **http://www.netbeans.info/downloads/index.php** y descarga la última versión disponible. Al momento de elaborar esta guía, está disponible la versión 5.5.
- Para iniciar la instalación, da doble click sobre el icono (Fig. 2.45).
- El instalador de NetBeans detectará el (los) JDK instalado(s) en tu computadora. Selecciona el que será usado por NetBeans para compilar.
- Verifica el directorio de instalación y que haya suficiente espacio en disco duro.
- Da click en *next* para iniciar la instalación.

2.6.3. Ejemplo de Uso de NetBeans.

Mostramos el uso básico de NetBeans mediante el tradicional *"Hola Mundo"*

- Primero crea un nuevo **Proyecto**, que es donde se alojará tu aplicación.
- Elige **File** → **New Project**.
- Aparece la ventana **New Project**. Elige **Categoría *General*, Proyecto *Java Application***.
- Da clik en *Next*.

Figura 2.45: Instalación de NetBeans (ventana inicial).

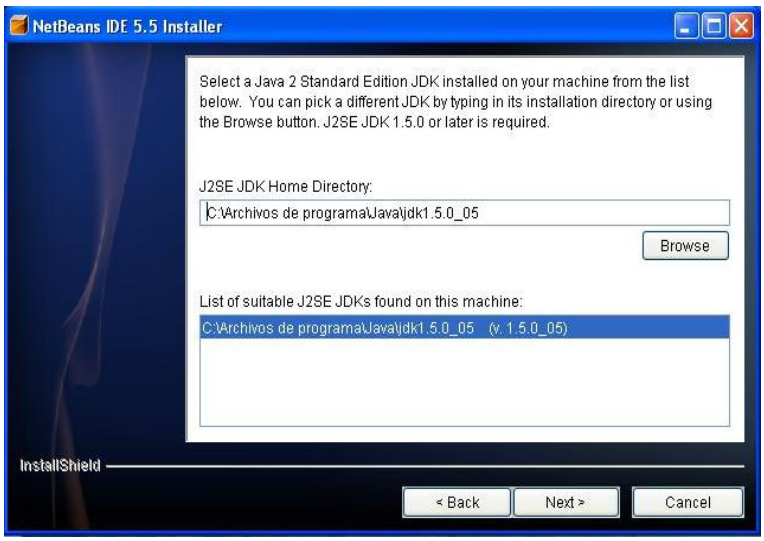

Figura 2.46: Instalación de NetBeans (selección del JDK).

- Aparece la ventana **New Java Application** donde das nombre al nuevo proyecto y eliges su ubicación. También da la opción para generar *el esqueleto* de una clase Main para ejecutar tu proyecto. Selecciona **Finish** para terminar. *Fig.* 2.48
- Ahora, para crear paquetes para la estructura de tu proyecto, colócate en la ventana **Projects**, da click derecho en la carpeta **Source Packages** de tu proyecto y selecciona **New** → **Java Package**.
- En la ventana **New Java Package** introduce el nombre de tu paquete. Selecciona **Finish** para terminar. *Fig.* 2.49

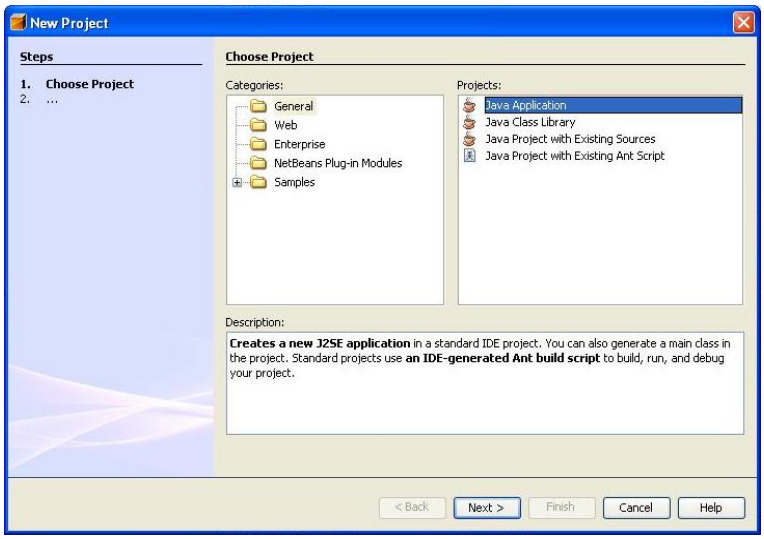

Figura 2.47: Selección del tipo de proyecto.

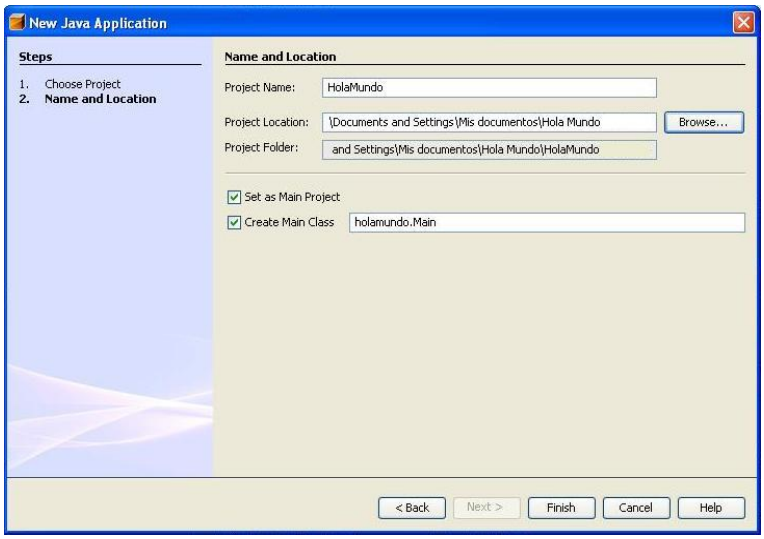

Figura 2.48: Nombre y ubicación del nuevo proyecto.

- Para crear una nueva clase, selecciona el paquete que has creado de la ventana **Projects** con click derecho **New** → **File/Folder**
- En la ventana **New File** elige la **Categoría *Java Classes*** y el **Tipo de archivo*Java Main Class***. *Fig.* 2.50
- Indica el nombre de tu clase (en este caso, **HolaMundo**) y selecciona la ubicación y paquete. Por defecto, el paquete es que el acabas de crear y la ubicación es la carpeta **src**. Da click en *Finish*. *Fig.* 2.51
- Añade *System.out.println("Hola Mundo")* al main.

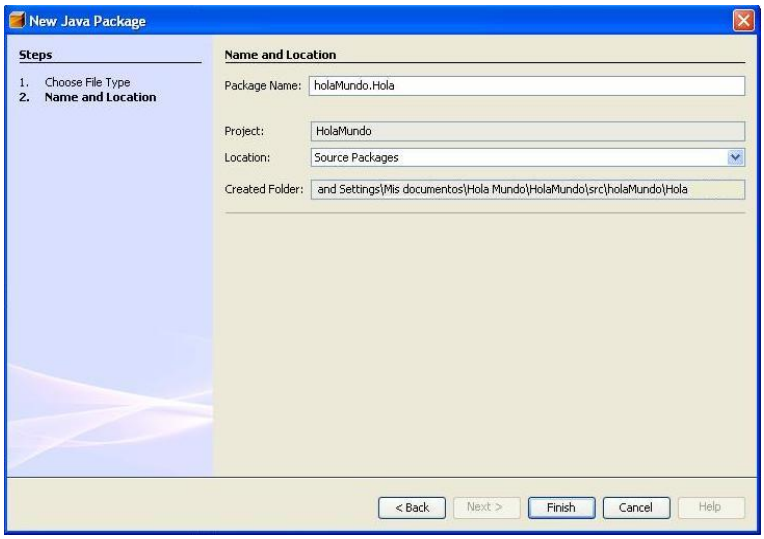

Figura 2.49: Nombre del nuevo paquete.

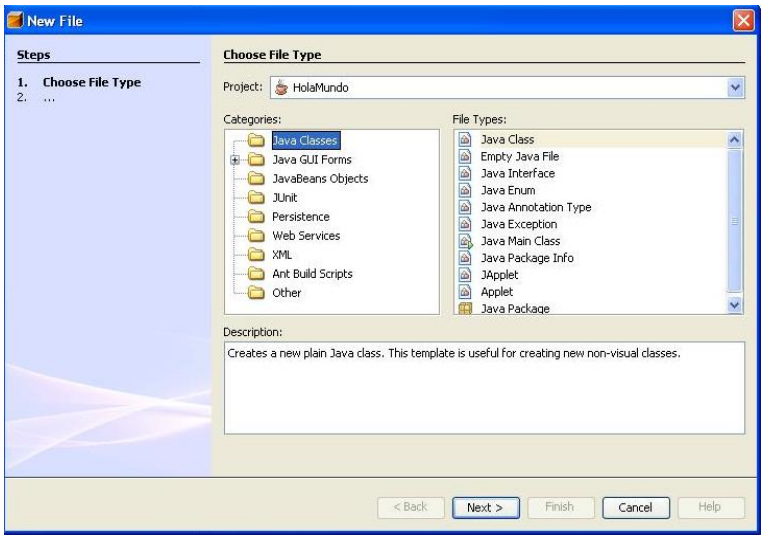

Figura 2.50: Crear una nueva clase.

- Para compilar, da click derecho en la clase HolaMundo.java de la ventana **Projects** y selecciona **Compile File**. El resultado se muestra en la parte inferior.
- Para ejecutar, da click derecho en la clase HolaMundo.java de la ventana **Projects** y selecciona **Run File**. El resultado se muestra en la parte inferior. *Fig.* 2.52

Ojalá este pequeño tutorial te haya sido de ayuda. Suerte.

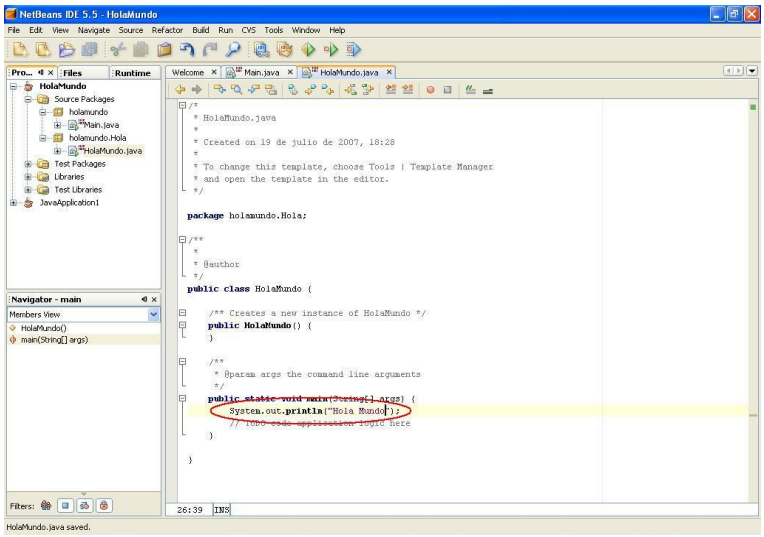

Figura 2.51: Código generado de la clase HolaMundo.

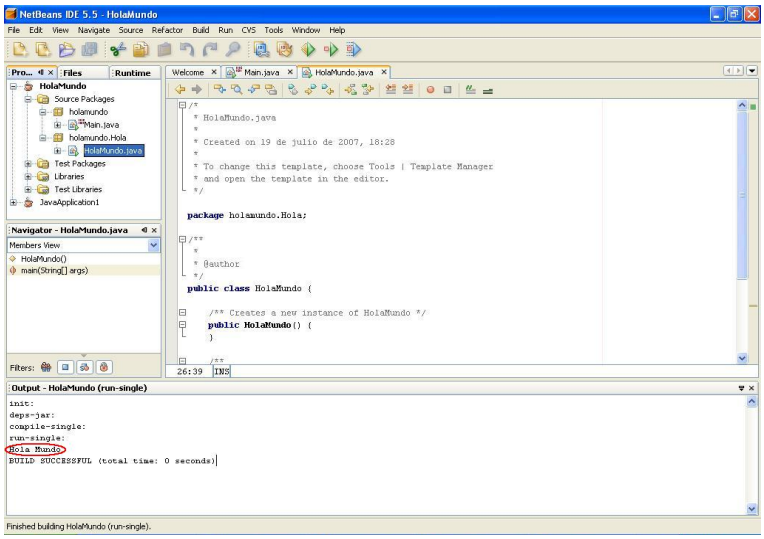

Figura 2.52: Ejecución de la clase HolaMundo.java.

2.7. Herramienta para la Construcción (IDE Eclipse)

Eclipse es una comunidad de desarrollo de software Open Source, sustentada por la Comunidad Eclipse, una empresa no lucrativa que se sostiene con la ayuda colaborativa de miembros en todo el mundo, tanto económica como en el desarrollo de software.

A la vez, Eclipse es un entorno de desarrollo integrado (IDE, por sus siglas en inglés), desarrollado por esta comunidad, bajo la licencia EPL (Eclipse Public License) una licencia aprobada por la OSI (Open Source Initiative), por lo que es considerada Open Source. Es una herramienta que funciona independientemente de la plataforma en que se trabaje.

2.7.1. Origen de Eclipse.

El Proyecto Eclipse fue originalmente creado por IBM en noviembre de 2001 y era soportado por un consorcio de empresas como Borland, Red Hat, SuSE y TogetherSoft. Para finales de 2003 el número de miembros había crecido a 80. Originalmente se creó para reemplazar la herramienta VisualAge. La Fundación Eclipse fue creada en enero de 2004, independiente de IBM.

La versión actual de Eclipse es la 3.3, llamada Europa.

2.7.2. Instalación de Eclipse.

- Visita la página **http://www.eclipse.org/** y descarga la última versión disponible (Al momento de desarrollar esta guía, la versión 3.3, llamada Europa). Debe ser la edición para desarrollo de aplicaciones web (Fig. 2.53).

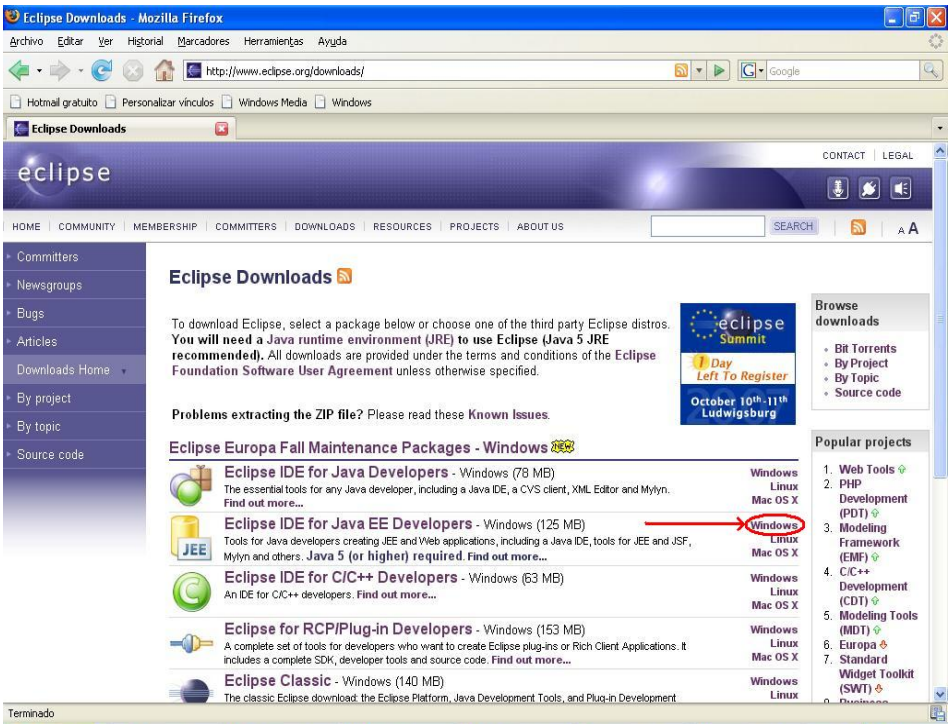

Figura 2.53: Página oficial de Eclipse.

Nota: Se asume que ya ha sido instalado el JDK de Java y el servidor de aplicaciones Tomcat en tu computadora.

- Descomprime el archivo descargado (como de costumbre, preferentemente en la carpeta *Archivos de Programa*).

- En la carpeta recién creada ya puedes ejecutar Eclipse dando doble click sobre el icono.

- Sin embargo, para facilitar varias de nuestras tareas requeriremos de la instalación de plugins extra, como veremos a continuación. El uso de éstos hace que se vea afectado el desempeño de Eclipse en cuanto a que no cuente con la suficiente memoria asignada. Para resolver este problema, Eclipse permite pasar argumentos directamente a la máquina virtual de Java, mediante el comando **-vmargs**. Así, para incrementar el heap de memoria disponible para Eclipse:

  - Cierra el IDE.
  - Abre una línea de comandos (Símbolo del sistema).
  - Colócate en la carpeta donde está ubicado el archivo ejecutable de Eclipse, por ejemplo: **C:\Archivos de programa\eclipse-jee-europa-fall-win32\eclipse**.
  - Ejecuta la siguiente instrucción:

    ```
    eclipse -vmargs -Xmx512M -XX:PermSize=64M -XX:MaxPermSize=128M
    ```

    Donde **-Xmx512M** es el máximo de heap asignable para el uso de la máquina virtual. **PermSize** es el tamaño de la sección de almacenamiento dinámico para la generación permanente de elementos por parte de la máquina virtual (clases, copias auxiliares, etc.). Por supuesto que esto dependerá de las capacidades de la máquina donde se trabaje, obviamente, ninguno de los parámetros debe ser igual o mayor a la memoria total disponible en la máquina que estás trabajando.

Cuando Eclipse inicia, muestra el espacio de trabajo (Workspace), redirecciónalo a la carpeta *webapps* de Tomcat, pues ahí residen las aplicaciones web con este servidor.

La inclusión de funcionalidades a través de plugins es sencilla; en la mayoría de los casos, éstos vienen comprimidos, así que basta con descomprimir el archivo dentro de la carpeta **plugins** de Eclipse y reiniciar el IDE. La próxima vez que se inicie, la nueva funcionalidad estará disponible.

2.7.2.1. Integración con Tomcat

- Visita la página **www.eclipsetotale.com/tomcatPlugin.html** y descarga la última versión disponible del plugin para Tomcat de Sysdeo, compatible con Eclipse 3.3 (Fig. 2.54).

- Coloca los archivos en la carpeta *plugins* y reinicia Eclipse.
- Observa los nuevos botones correspondientes a Tomcat en la barra de herramientas (Fig. 2.55).
- Ahora indica la ubicación de la carpeta de Tomcat para poder hacer uso de éste desde el IDE.

  - Selecciona *Window→Preferences→Tomcat*
  - Selecciona la versión de Tomcat que hayas instalado e indica la ubicación de la carpeta raíz. Por ejemplo: **C:\Archivos de programa\Apache Software Foundation\Tomcat 6.0** (Fig. 2.56).

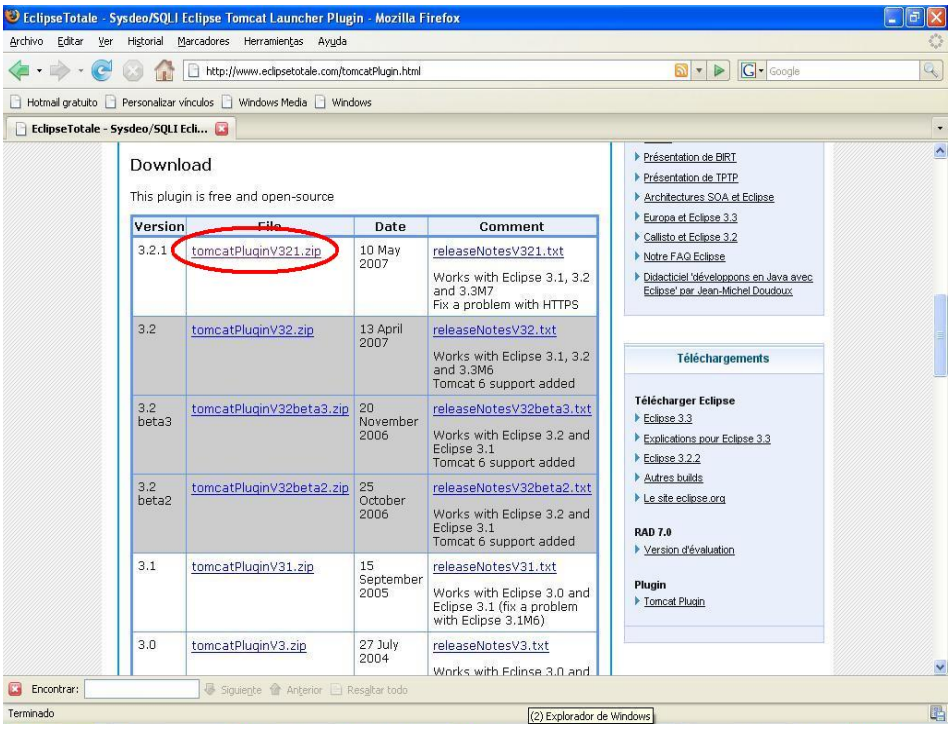

Figura 2.54: Descarga del plugin para integrar Tomcat.

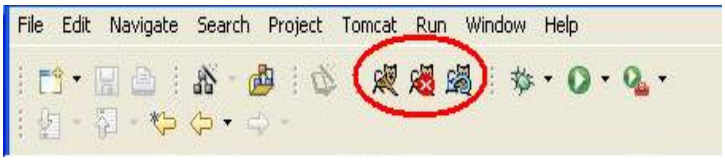

Figura 2.55: Botones de control de Tomcat.

- Listo, tenemos la integración de Eclipse con Tomcat. A continuación veremos un ejemplo de aplicación. La gran ventaja es que al elegir hacer un nuevo proyecto de Tomcat (Tomcat Project) nos generará sólo la estructura de carpetas estándar necesaria, a diferencia de la que se genera cuando elegimos una aplicación web típica.

2.7.2.2. Integración con UML

Para elaborar diagramas UML, usaremos el plugin de la empresa Omondo, con la ayuda adicional de que genera código Java automáticamente a partir de los diagramas, además se pueden desarrollar los principales diagramas de UML.

- Descarga el plugin para tu sistema operativo desde la página **http://www.eclipsedownload.com/** y colócalo en la carpeta *plugins* de Eclipse. Al reiniciar el IDE estarán disponibles las nuevas funcionalidades.

- Para verificar, selecciona *File→New→Other* y estará disponible la opción para diagramas

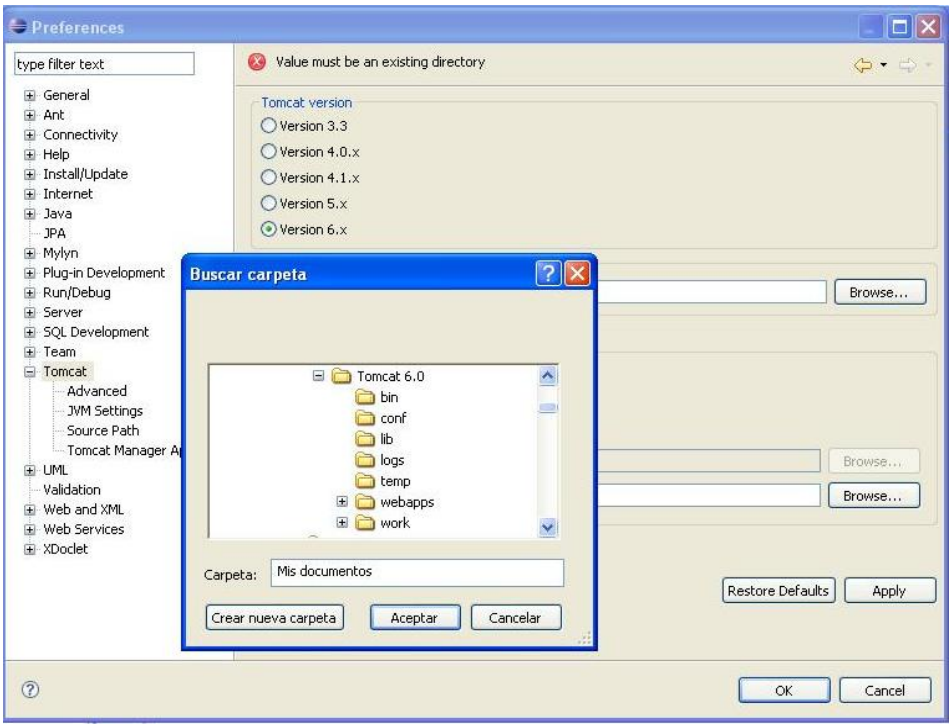

Figura 2.56: Selección del servidor.

UML (Fig. 2.57).

- Listo, a continuación veremos el uso de este plugin como parte del desarrollo de una aplicación, así como la forma de generar código a partir de los diagramas.

2.7.2.3. Ejemplo de Uso del Plugin de UML

- Crea la nueva aplicación *File→New→Other→Tomcat Project* (Fig. 2.58).
- Llama a la nueva aplicación *AutenticaEclipse*.

- Click derecho sobre el nuevo proyecto en el *Package Explorer* y selecciona *Nuevo Folder* y llámalo **Diagramas**.
- Sobre éste, *click derecho → New → Other → UML Class Diagram*
- Aparece el área de dibujo y la barra de elementos correspondientes al tipo de diagrama. Comienza por nombrar el paquete al que pertenecerán las clases (Fig. 2.59 y 2.60).
- Selecciona el elemento *Class* y define la clase *Persona* con los siguientes atributos (Fig. 2.61 y 2.62):
  - nombre
  - fechaNacimiento

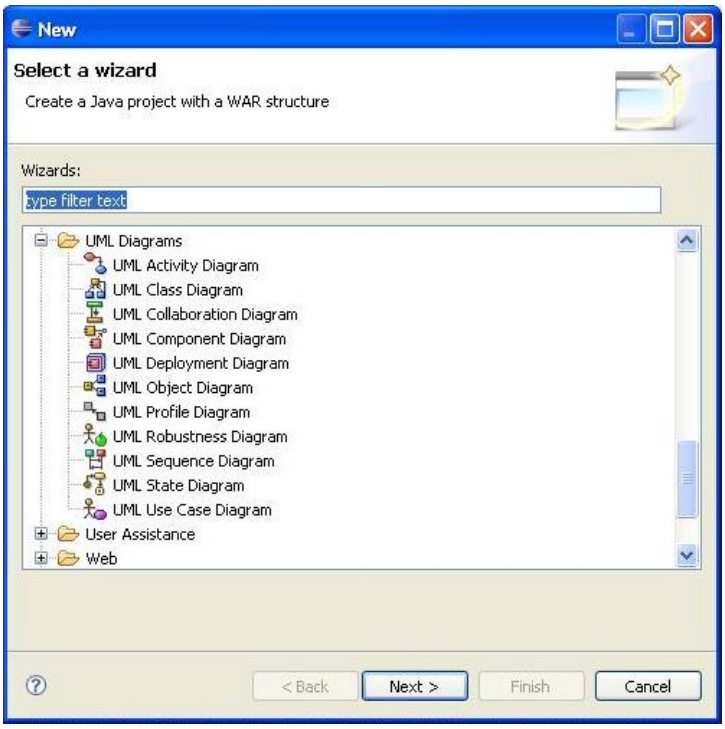

Figura 2.57: Opciones de diagramas UML.

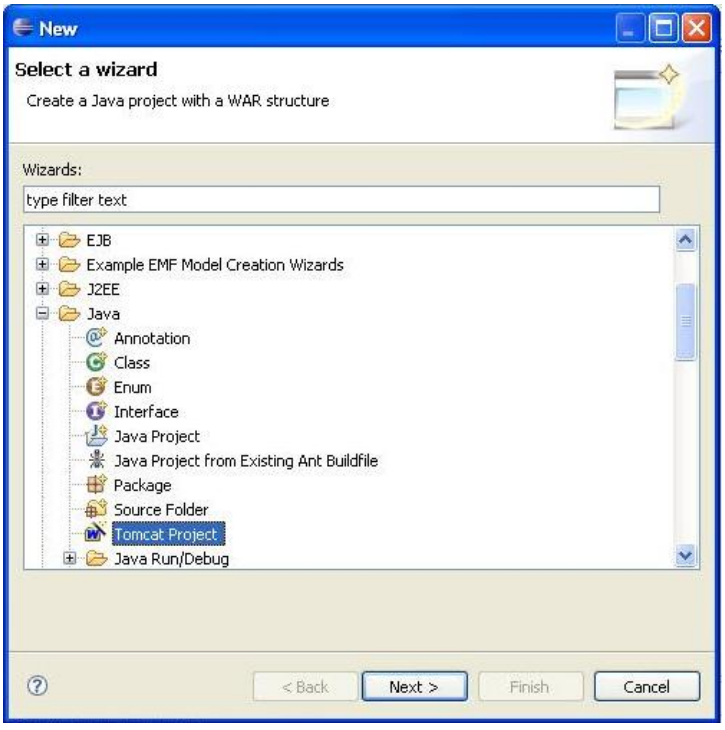

Figura 2.58: Nuevo Proyecto de Tomcat.

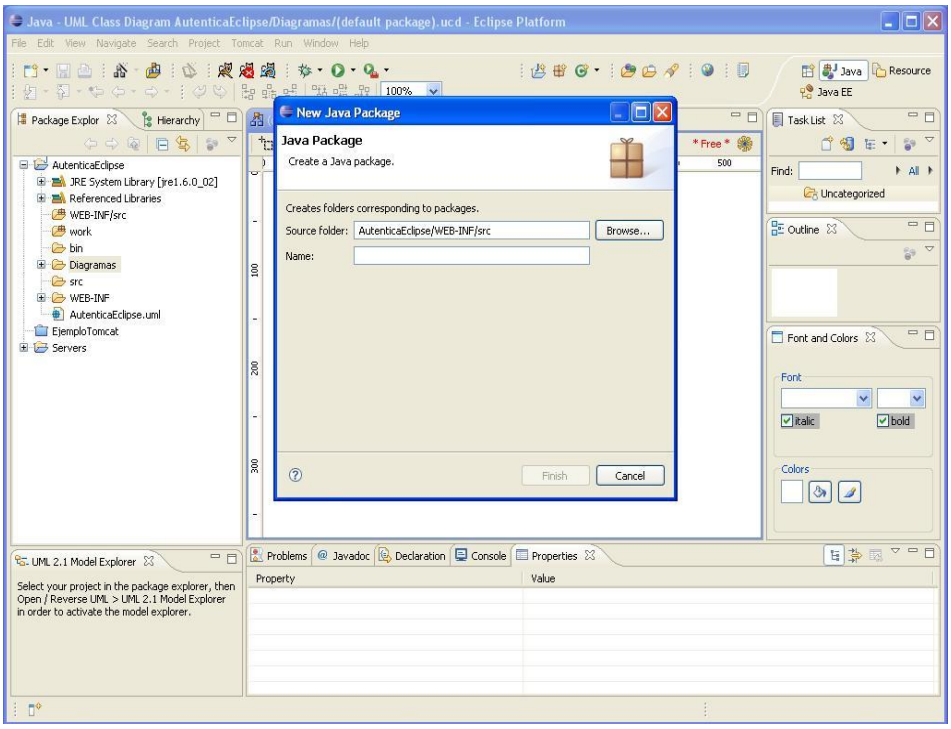

Figura 2.59: Asigna nombre al paquete.

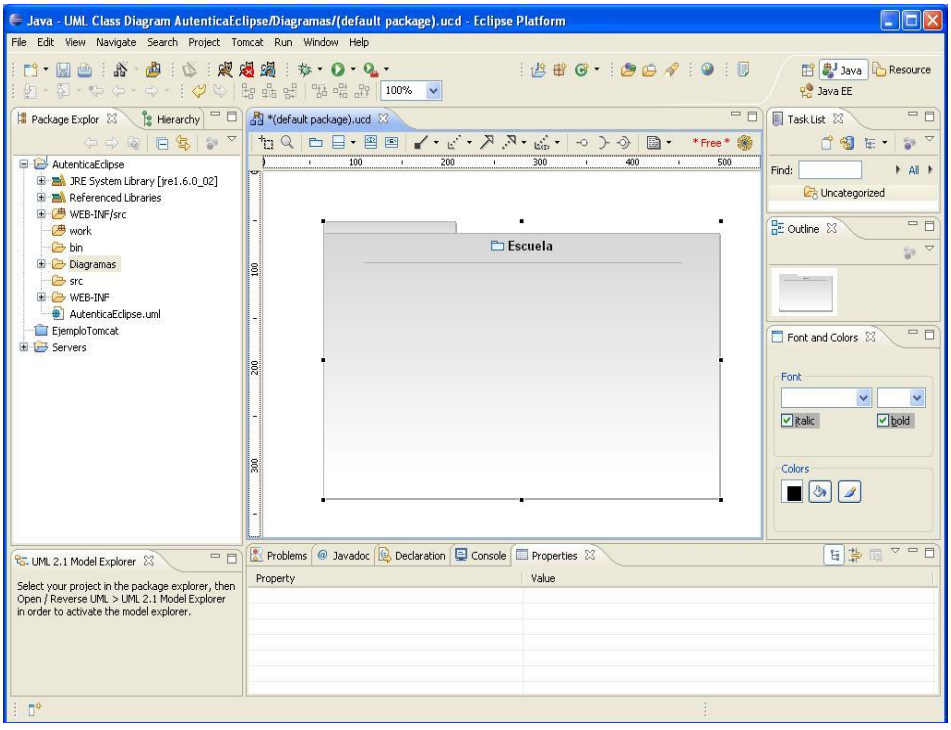

Figura 2.60: Creación del paquete Escuela.

- direccion
- telefono

Los *getter* y *setter* correspondientes son automáticamente añadidos como métodos.

- Agrega los métodos
  - darDeBaja
  - darDeAlta

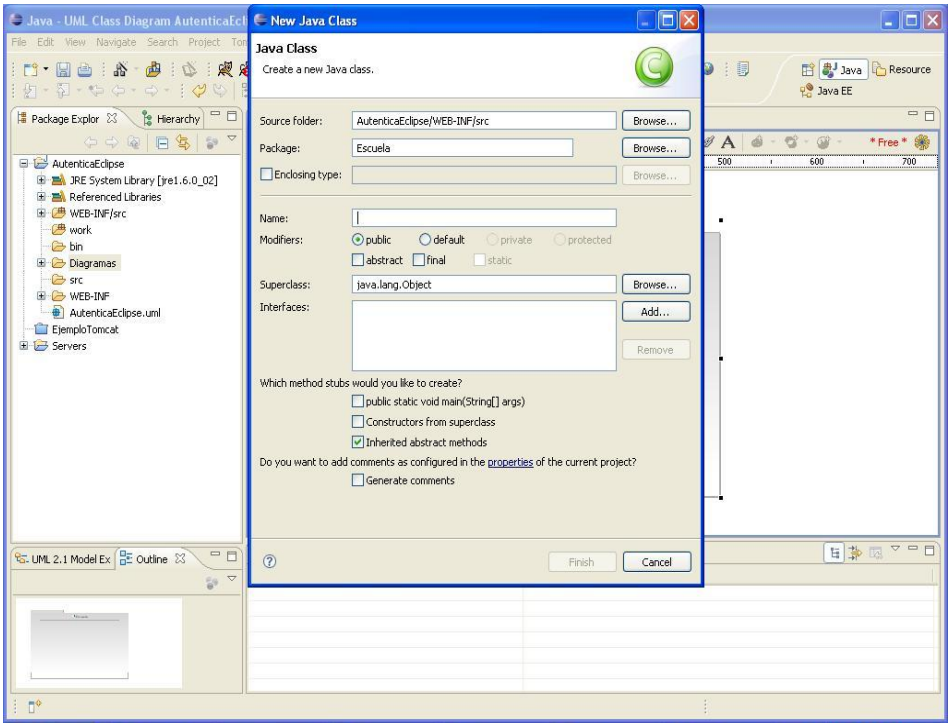

Figura 2.61: Nueva clase.

- Genera las subclases *Alumno* y *Profesor*, con los atributos *noCuenta* (número de cuenta) y *rfc* (RFC como trabajador), respectivamente. Al terminar, el código Java ha sido automáticamente generado y guardado en la subcarpeta **WEB-INF/src** (Fig. 2.63).

  Y se puede ver el árbol del código generado (Fig. 2.64).

  Al seleccionar estas clases se muestra el código generado (Fig. 2.65).

Ojalá este pequeño tutorial te haya sido de ayuda. Suerte.

2.8. Conclusiones al Usar NetBeans y Eclipse

Con la experiencia adquirida, tanto propia como de algunos compañeros colegas al consultarlos y leer las opiniones que algunos cibernautas dejan en foros en Internet, se podría decir que

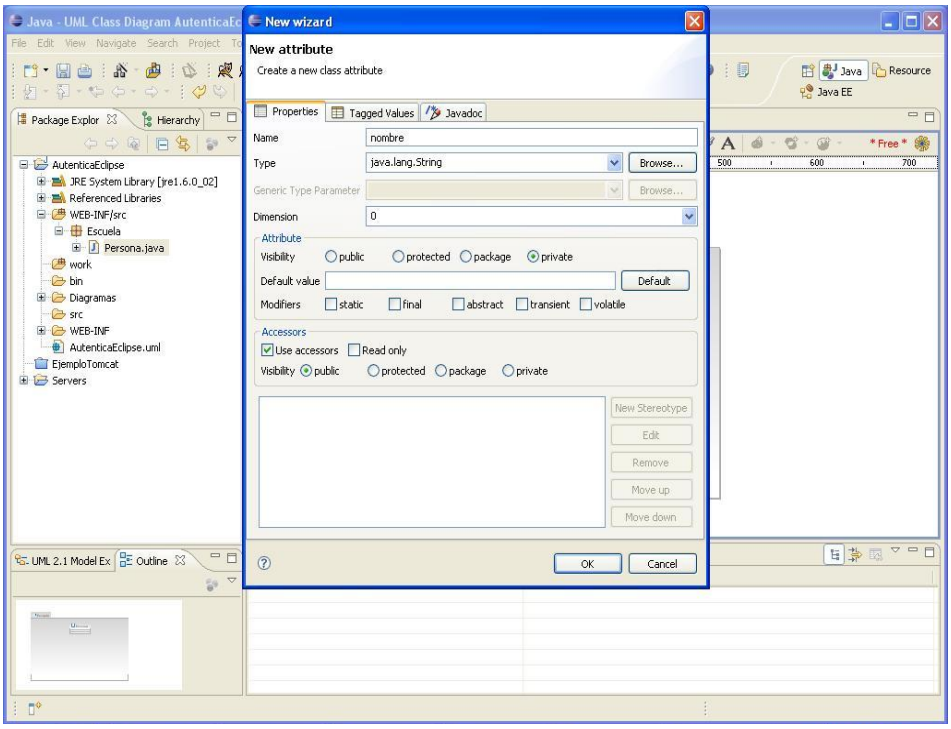

Figura 2.62: Nuevo atributo.

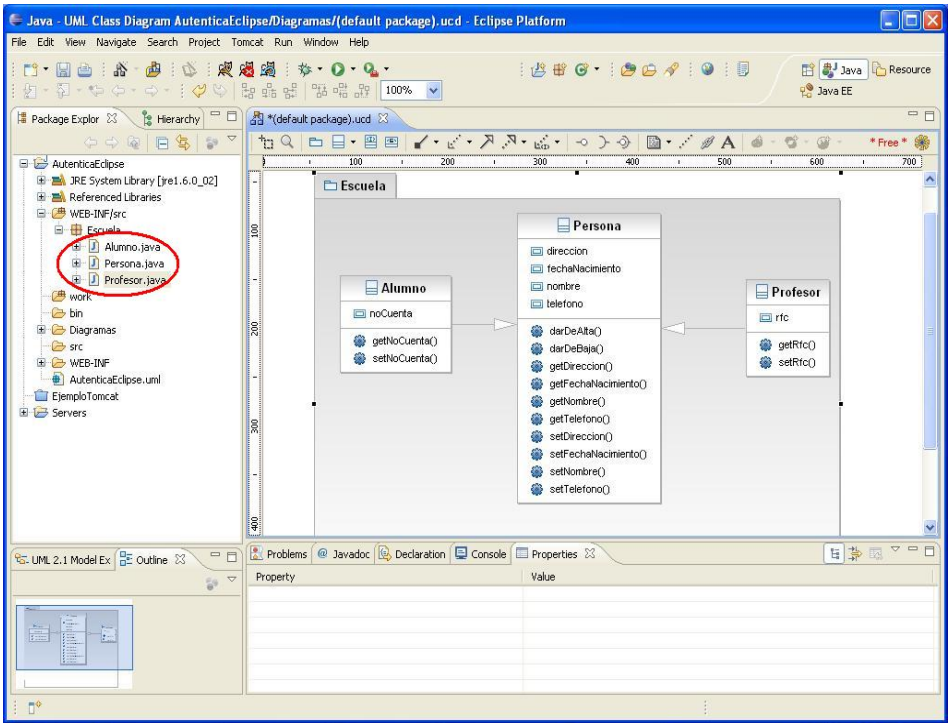

Figura 2.63: Vista del diagrama de clases.

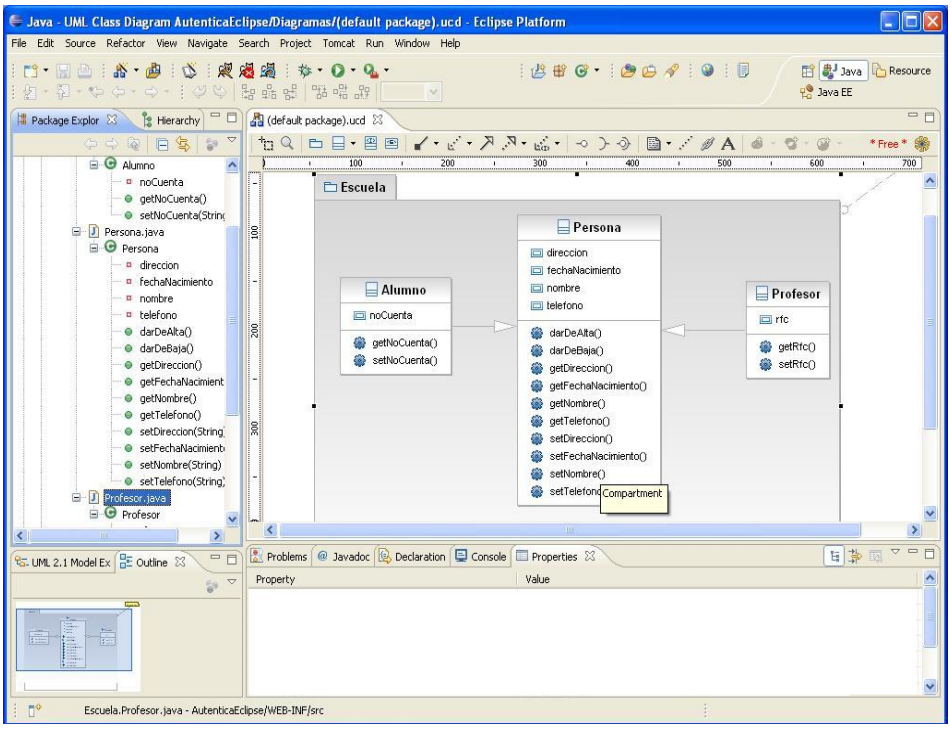

Figura 2.64: Árbol del código generado.

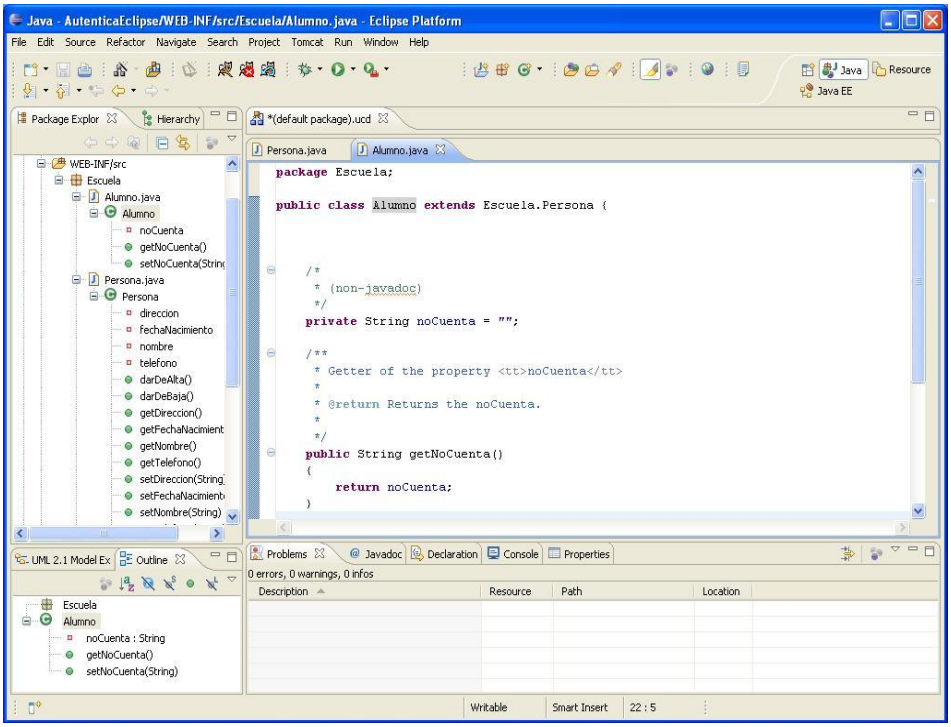

Figura 2.65: Código generado de la clase Alumno.

la cantidad de preferencias por cada uno de estos IDEs es muy pareja. En mi opinión, ambas herramientas han demostrado ser muy útiles y estables, no por nada son las de uso más difundido en la programación Java y continuamente se liberan nuevas versiones, pero por supuesto que lo mejor será utilizarlas y experimentar un buen rato con ellas.

Algunas características encontradas y mencionadas entre estos IDEs son:

- NetBeans trae más plugins por default y, en particular, trae más plugins para trabajar en construcción de aplicaciones Java, pues fue pensado con este fin, sin embargo en Eclipse tú puedes instalar sólo los que vayas necesitando.
- Eclipse por default ocupa menos recursos, aunque, como vimos en este tutorial, eso depende de la cantidad de plugins instalados y esa cantidad de memoria disponible se puede asignar manualmente.
- Parece haber más plugins disponibles para Eclipse que para NetBeans.
- En Eclipse hay que indicar explícitamente los archivos .jar que la aplicación va a utilizar, no así en NetBeans, pues éste sabe el tipo de contenido de las carpetas del proyecto.

La mejor elección la tiene el usuario al experimentar con cada uno, y elegir el más adecuado a sus necesidades, pues como en muchos casos, lo que es útil para unos, para otros puede ser innecesario o representar un estorbo. Además se debe tener en cuenta la disposición de los recursos de la computadora, la interfaz gráfica del IDE, así como el sentirse cómodo en el ambiente.

Ojalá estos tutoriales ayuden a tu elección.

2.9. Herramienta para las Pruebas

Junit es un conjunto de clases Java que sirve para hacer pruebas en el código fuente durante el proceso de desarrollo de aplicaciones (igualmente Java), para verificar que los resultados obtenidos sean los esperados. Cuenta con dos formas de visualización de los resultados, gráfica o modo texto. En lo últimos años se ha difundido ampliamente su utilización dentro de grupos de trabajo de desarrollo de software, en gran parte debido a su distribución gratuita, pero sobre todo a su alto grado de confiabilidad.

Es software Open Source bajo la Common Public License Version 1.0.

2.9.1. Origen de JUnit

JUnit fue desarrollado por Erich Gamma y Kent Beck. El primero es un informático suizo, miembro de la conocida "Banda de los cuatro", un grupo de autores que escribieron el libro "Design Patterns" (referencia en el campo de la programación orientada a objetos), mientras que el segundo, estadounidense, figura en el campo de las metodologías ágiles en el desarrollo de software.

2.9.2. Instalación de JUnit.

- Primero descargamos la carpeta comprimida desde la página de JUnit:
  http://www.junit.org/index.htm y en la sección **Download** descargamos el archivo **junit4.3.1.zip**, que es la versión más reciente disponible de JUnit. Sin embargo ésta tiene la limitante de no contar con una interfaz gráfica, por lo que si tú deseas contar con una, la versión 3.8 cuenta con dos tipos de interfaz gráfica.
  Entonces descarga al archivo **junit3.8.1.zip** de esta página:
  http://sourceforge.net/project/downloading.php?groupname=junit&filename=junit3.8.1.zip&use_mirror=internap

- Descomprime el archivo que has descargado.

- Coloca tus variables de entorno correspondientes para correr JUnit (procura descomprimir la carpeta en un directorio cuyo nombre no contenga espacios, pues esto podría ocasionar problemas): Desde **Escritorio** haz click derecho sobre el icono **Mi PC**; selecciona **Propiedades** → **Opciones Avanzadas** → **Variables de Entorno**. Crea una nueva variable **CLASSPATH** (si ya existe, sólo modifícala) y su valor será la ruta hasta tu archivo **junit-4.1.jar** (o **junit.jar**, en el caso de **Junit3.8.1** [no olvides agregar .jar]). (Fig. 2.66). También debes añadir la ruta hasta la carpeta que contiene este archivo (sin incluir el archivo). (Fig. 2.67).

- Da click en Aceptar hasta cerrar todas las opciones de **Propiedades del sistema**.

Suponiendo la instalación de la versión 3.8.1 y que se descomprimió en **Archivos de programa**:

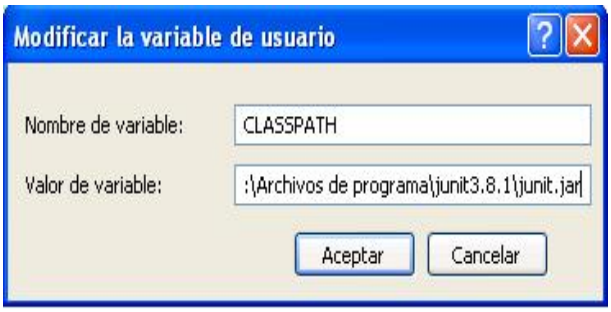

Figura 2.66: Asignación de valores a variables de entorno: Archivo.

- Abre una ventana de Símbolo del sistema.

- Para comprobar la correcta instalación, parado en cualquier directorio puedes teclear:

  ```
  java junit.swingui.TestRunner
  ```

  y aparecerá la siguiente ventana: (Fig. 2.68).

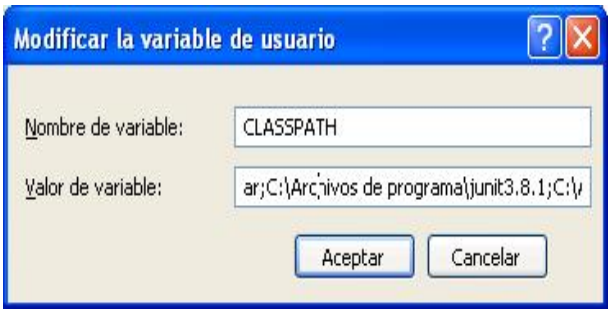

Figura 2.67: Asignación de valores a variables de entorno: Carpeta.

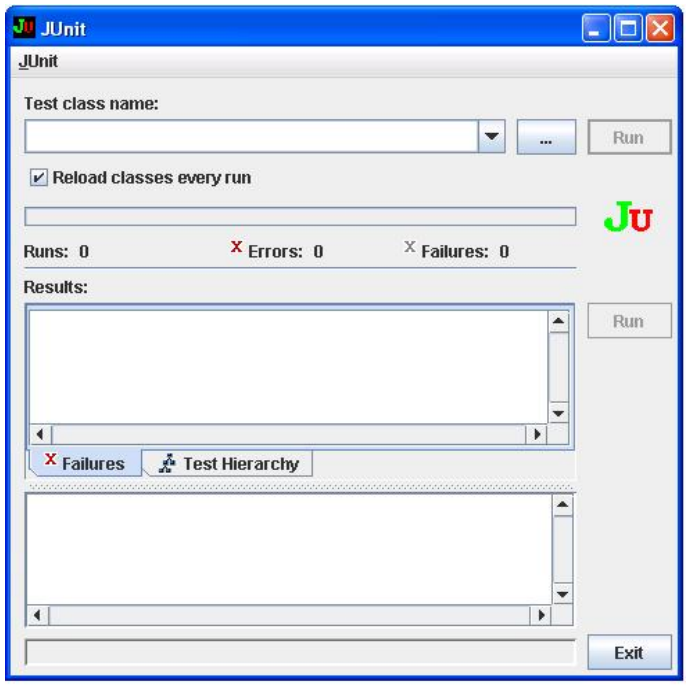

Figura 2.68: Ventana Principal de JUnit.

Aquí tú puedes hacer pruebas a las clases (archivos .class) que tengas en la carpeta donde estás parado. Para hacer una prueba rápida de JUnit, ejecuta

```
java junit.swingui.TestRunner junit.samples.AllTests
```

Y se despliega la misma ventana con las pruebas correspondientes: (Fig. 2.69).

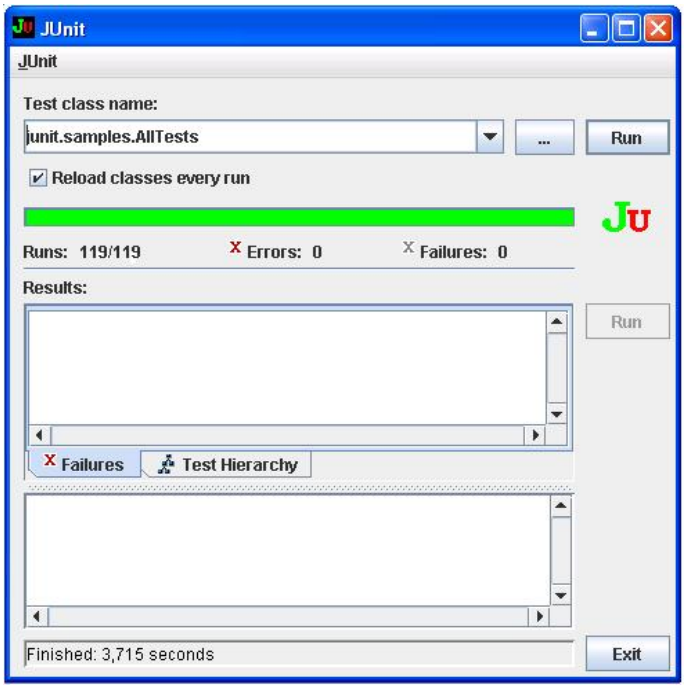

Figura 2.69: Ventana Principal de JUnit con la prueba ejecutada.

2.9.3. Requerimientos en las Pruebas

- Por principio de cuentas, una clase a probar con JUnit debe importar el paquete **junit.framework**, por lo que debemos añadir **import junit.framework.*;** a nuestra clase.
- La clase a probar debe extender de la clase **TestCase**.
- Los métodos a probar deben llevar el prefijo *test*, JUnit reconocerá como métodos a probar sólo a aquéllos que cumplan con esta regla.
- Para ejecutar las pruebas, la clase debe contener un método *main* y opcionalmente (como veremos) un método *suit*

2.9.3.1. Principales Métodos para Pruebas

- assertTrue(exp): comprueba que la expresión **exp** se evalúe a *true*.
- assertFalse(exp): comprueba que la expresión **exp** se evalúe a *false*.

- assertEquals(esperado, obtenido): comprueba que el valor **esperado** sea igual al **obtenido**.
- assertNull(objeto): comprueba que el **objeto** sea *null*.
- assertNotNull(objeto): comprueba que el **objeto** no sea *null*.
- assertSame(objeto_esperado, objeto_obtenido): comprueba que el **objeto_esperado** y el **objeto_obtenido** sean el mismo objeto.
- assertNotSame(objeto_esperado, objeto_obtenido): comprueba que el **objeto_esperado** no sea el mismo objeto que el **objeto_obtenido**.
- fail(): hace que el test termine con fallo.

2.9.4. Ejemplo de Uso de JUnit:

Veremos a continuación un ejemplo de uso de JUnit. Empezaremos por la creación de la clase Dinero.

```
class Dinero{

    //variables: valor es la denominacion de la moneda
    //moneda es el tipo (pesos, dolares, etc.)

    private int valor;
    private String moneda;

    public Dinero(int valor, String moneda){
        this.valor = valor;
        this.moneda = moneda;
    }

    public int getValor(){
        return this.valor;
    }

    public String getMoneda(){
        return this.moneda;
    }

    //suma los valores de dos instancias de Dinero.

    public Dinero suma(Dinero d){
       return new Dinero(getValor() + d.getValor(), getMoneda());
    }

    //Devuelve el valor y el tipo de moneda
```

```
    public String toString(){
        String s = "";
        s += "valor: " + getValor() + ", moneda: " + getMoneda();
        return s;
    }
}
```

Y a continuación la clase de pruebas:

```
import junit.framework.*;

public class DineroTest extends TestCase{

    //Dos instancias de Dinero para sumar sus valores

    private Dinero d1;
    private Dinero d2;

    //metodo para inicializar las variables globales

    protected void setUp(){
        d1 = new Dinero(10, "Pesos");
        d2 = new Dinero(15, "Pesos");
    }

    //metodo para probar la suma de los valores de las instancias de Dinero,
    //se comparan las representaciones completas (par valor, moneda)

    public void testSuma(){
        String esperado = new Dinero(25, "Pesos").toString();
        String resultado = d1.suma(d2).toString();
        assertEquals(esperado, resultado);
    }

    //metodo para emplear algunos de los metodos anteriormente descritos,
    //si se descomenta la tercera linea, JUnit marca un error.

    public void testEquals(){
        Assert.assertTrue(!d1.equals(null));
        //Assert.assertEquals(d1, d2);
        Assert.assertEquals(d1.toString(), new Dinero(10, "Pesos").toString());
        Assert.assertTrue(!d1.equals(d2));
    }

    //metodo para ejecutar JUnit, se llama desde el main
```

```
    public static Test suite(){
        return new TestSuite(DineroTest.class);
    }

    public static void main(String args[]){
        junit.textui.TestRunner.run(suite());

        // se tiene la alternativa de llamar directamente al metodo run para
        // ejecutar la prueba.

        junit.swingui.TestRunner.run(DineroTest.class);
    }
}
```

Se muestra el resultado de la ejecución en la ventana de JUnit. (Fig. 2.70).

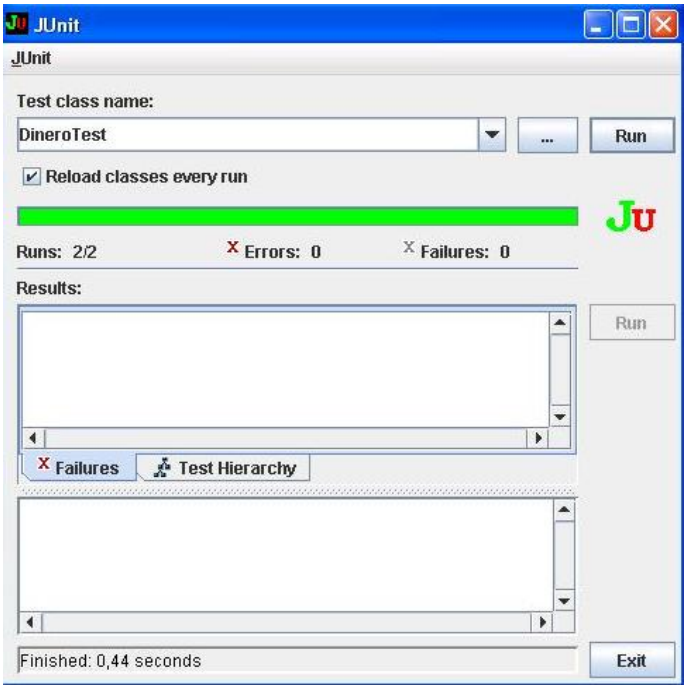

Figura 2.70: Ejecución en JUnit.

El manejo de JUnit, en mi opinión, es otro de los casos en que resulta un poco complicado al principio, en cuanto a saber qué métodos generar, pero una vez dominado llega a ser de gran utilidad e imprescindible. Y otra vez, afortunadamente los IDEs nos proporcionan una gran ayuda en la generación automática de las clases de prueba.

Ojalá este pequeño tutorial te haya sido de ayuda. Suerte.

Capítulo 3

Ejemplo de Aplicación Web: Autenticación.

Se trata de hacer un software para una autenticación básica para ingresar a un sistema proporcionando clave de usuario y contraseña. A pesar de su simplicidad, muestra la forma de integrar las diversas herramientas presentadas anteriormente.

El funcionamiento de la aplicación es básicamente el siguiente:

Los nombres, claves y contraseñas de los usuarios se encuentran en la tabla *Usuarios*. Para ser autorizado, el usuario debe teclear la clave y la correspondiente contraseña. En caso de coincidir con los registrados en la base de datos, el sistema direcciona a la página *autorizado.jsp* donde se busca el nombre del usuario en la base de datos y se imprime en la pantalla. En caso contrario, se direcciona a la página *noautorizado.jsp* donde se muestra un mensaje solicitando que el usuario verifique la clave y password tecleados y, tras 4 segundos, se redirecciona nuevamente a la página inicial.

Adicionalmente, se puede asignar un valor para indicar si el usuario ya ha sido autorizado a visitar otras páginas del sistema y no acceda "por fuera", es decir, tecleando la url directamente en la barra de direcciones. Para esto se puede hacer uso de *sesiones*.

A continuación se listan los principales pasos para la creación de esta aplicación.

3.1. Planificación del Proyecto con GanttProject y Modelado con ArgoUML

Aquí se muestra una idea de cómo hacer la planificación del proyecto *Autenticación* con GanttProject. Aunque es una aplicación muy sencilla que no toma mucho tiempo como el que se simula, el ejemplo tiene un fin meramente ilustrativo (Fig. 3.1). Asimismo se muestra el diagrama de estados o navegación con ArgoUML. Como se mencionó, sólo se tienen cuatro páginas. En la página de inicio el usuario proporciona la clave y contraseña, en la siguiente (*Validación*) se verifica que éstos coincidan con los registros almacenados en la base de datos. En caso afirmativo

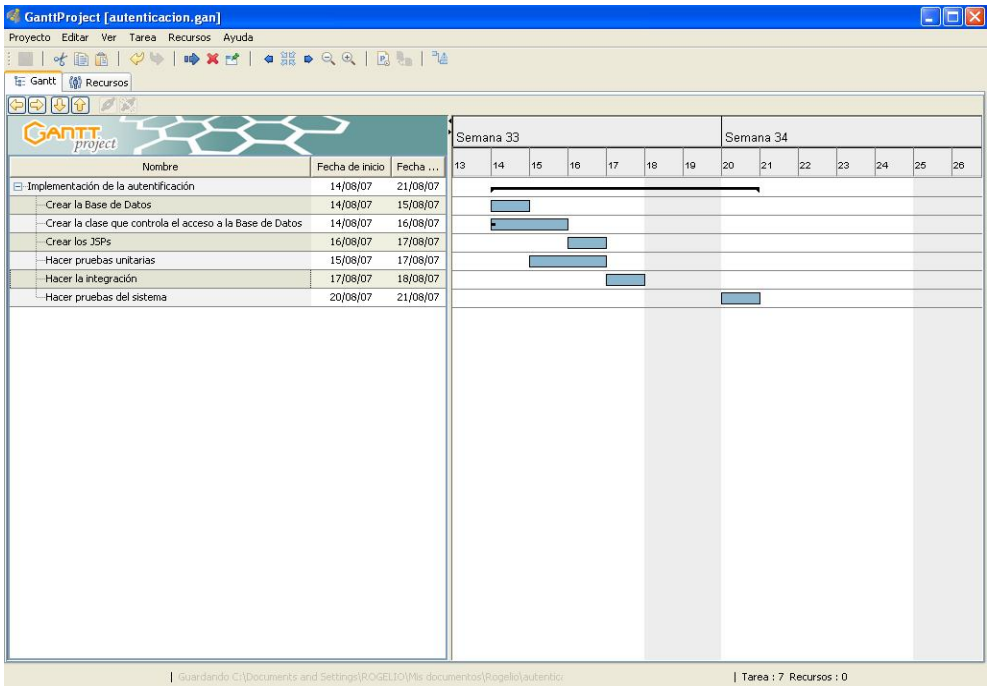

Figura 3.1: Planificación del proyecto.

se direcciona a la página *autorizado* y en caso contrario a la página *noautorizado* (Fig. 3.2).

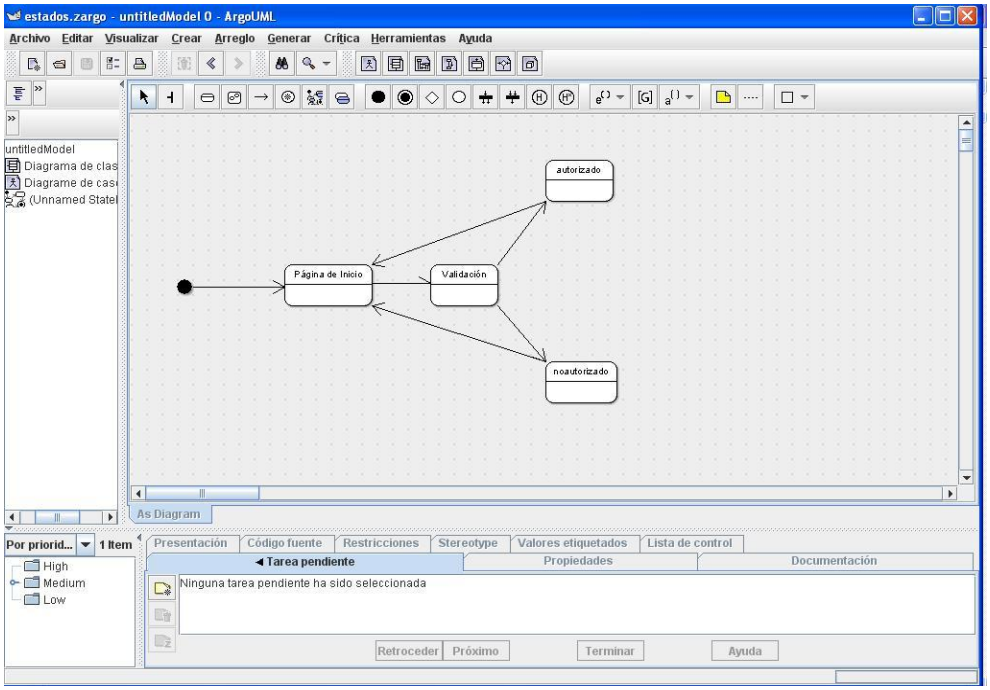

Figura 3.2: Diagrama de navegación.

3.2. Creación de la Base de Datos en MySQL

.

- En MySQL se crea la base de datos *Autenticacion*. Y dentro de ésta, la tabla *Usuarios*, cuyos atributos son:
  - Clave: La clave del usuario, que es la clave primaria.
  - Pwd: El password del usuario.
  - Nombre: El nombre del usuario, es devuelto cuando una vez que el usuario ha sido autorizado.

  Los comandos en MySQL son los siguientes:

```
CREATE DATABASE Autenticacion;
use autenticacion
CREATE TABLE USUARIOS(
Clave VARCHAR (30) NOT NULL,
Pwd VARCHAR (30) NOT NULL,
Nombre VARCHAR (70) NOT NULL,
PRIMARY KEY(Clave)
);
```

  Podemos insertar directamente una tupla en la tabla, aunque por supuesto, será más interesante implementar el registro a través del propio sistema.

```
INSERT INTO Usuarios VALUES('usuario1', 'pwd1', 'Pancho Lopez');
```

3.3. Construcción del Proyecto con NetBeans y Tomcat

El IDE NetBeans contiene un servidor Tomcat integrado. Para activarlo:

- Ejecuta NetBeans y en la barra de herramientas selecciona *Window* → *Runtime*. En la ventana izquierda, despliega el icono *Servers* e identifica **Tomcat**. Da click derecho y selecciona **Start**. La información correspondiente al arrancar el servidor aparece en la ventana inferior (Fig 3.3).
- Para verificar que el servidor ha sido levantado, selecciona *Tools* → *Server Manager* para ver en qué puerto escucha. Por default es el 8084; en tal caso, en el navegador teclea **http://localhost:8084**.

  Para generar la estructura del proyecto se presentan los siguientes pasos:

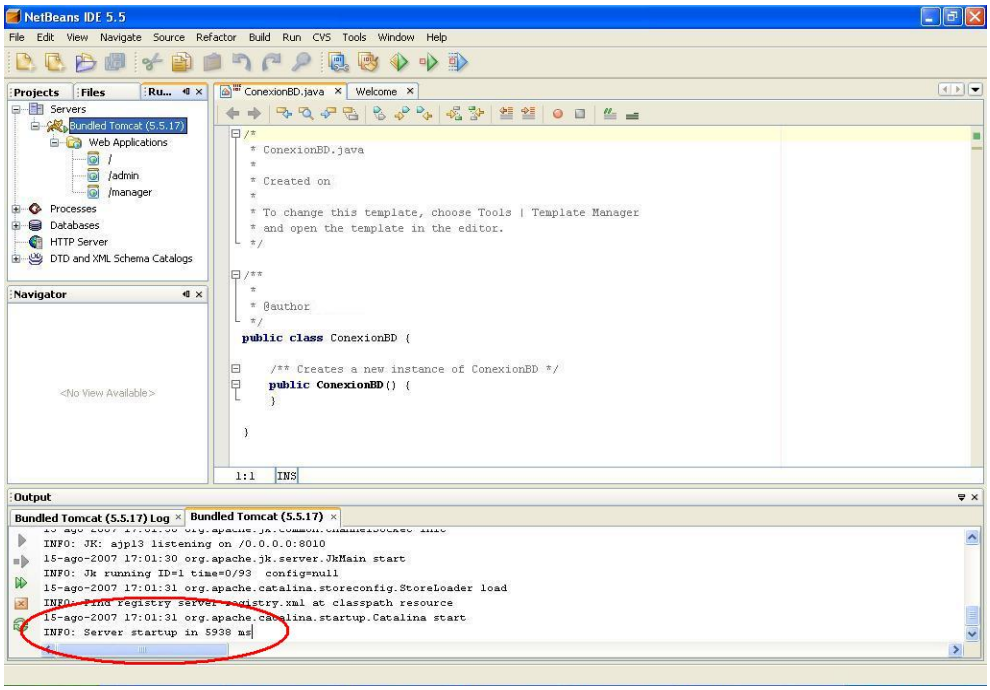

Figura 3.3: Levantando el servidor.

- En NetBeans selecciona **File → New Project** y en la ventana de opciones selecciona la **Categoría: Web**, **Projects: Web Application**.
- Da click en Next. Da nombre a la aplicación, en este caso **Autenticacion**.
- Selecciona la ruta donde se alojará el proyecto. Ésta será la carpeta *Webapps* de *Apache Tomcat* de NetBeans. Por ejemplo: *C:\Archivos de programa\netbeans-5.5 \enterprise3\apache-tomcat-5.5.17\webapps*.
- Selecciona el servidor, en esta caso el que seleccionaste en pasos anteriores.
- Da click en Finalizar (Fig 3.4).

Automáticamente en la pantalla principal se muestra el código del archivo index.jsp. Observa la estructura de directorios creada por NetBeans en la ruta que especificaste.

Para la creación de la clase que llevará el control de la base de datos:

- En la ventana izquierda, en la pestaña *Projects* despliega el nodo *Source Packages*, da click derecho y selecciona *New → Java Class*.
- Asigna el nombre, por ejemplo *ConexionBD* y el paquete, por ejemplo *Autenticacion*. Se muestra el código del archivo ConexionBD.java:

```
/**
 *
```

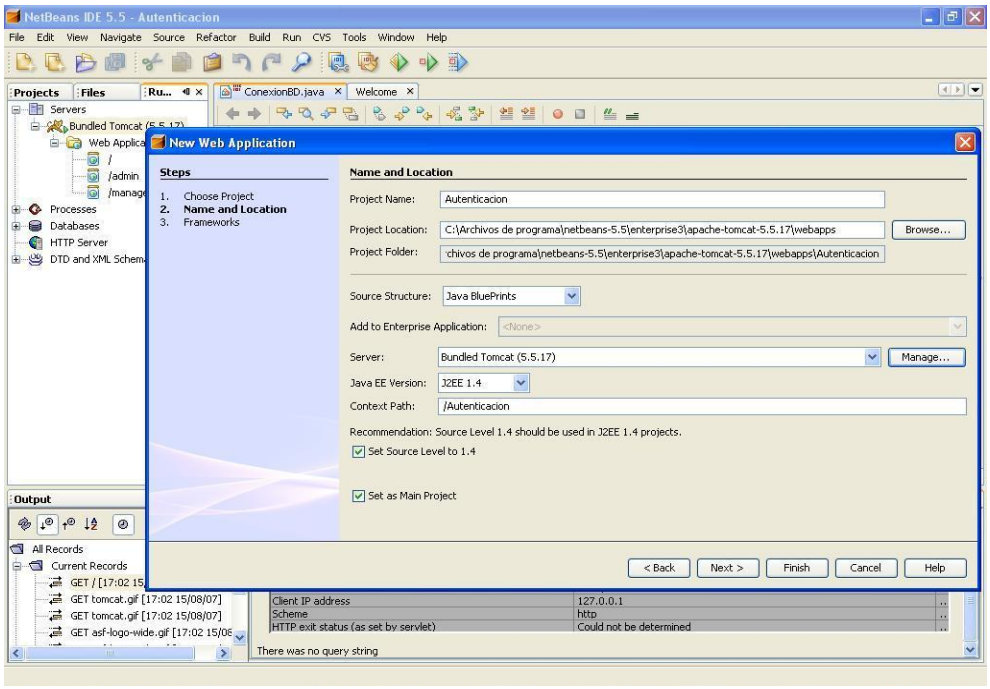

Figura 3.4: Nuevo Proyecto.

```
 * @author Curso IS
 */

import java.sql.*;

public class ConexionBD{

    // Variables para el control de la BD

    private String bDatos;
    private String servidor;
    private String user;
    private String password;
    private String driver;
    private Connection conexion;
    private Statement stmt;
    private ResultSet rs;

    /**
     * Crea una instancia de ConexionBD
     * asi como la conexion con la BD
     * @throws SQLException Lanzada si no se establece
     * correctamente la conexion
```

```
 */

public ConexionBD() throws SQLException{
    bDatos = "Autenticacion";
    servidor = "localhost";
    user = "root";
    password = "admin";
    driver = "com.mysql.jdbc.Driver";
    try{
        Class.forName(driver);
        conexion = DriverManager.getConnection("jdbc:mysql://" +
                                               servidor + ":3306/" +
                                               bDatos , user, password);
    }catch(Exception e){
        System.out.println("Error al conectar: " + e.getMessage());
    }
}

/**
 * Cierra la conexion con la BD
 * @throws SQLException Lanzada si no se puede cerrar
 * correctamente la conexion
 */

public void desconectar() throws SQLException{
    try{
        conexion.close();
    }catch(SQLException ex){
        System.out.println("SQLException: " + ex.getMessage());
    }
}

/**
 * Hace la validación del usuario, si coinciden la clave de
 * usuario y el password, se permite el acceso.
 * @param clave la clave de usuario
 * @param pwd El password
 * @return Un valor indicando si el usuario ha sido o no autorizado
 * @throws SQLException Lanzada si no se ejecuta correctamente el query
 */

public boolean verifica(String clave, String pwd)
    throws SQLException {
    try{
        stmt=conexion.createStatement();
```

```
        rs = stmt.executeQuery( "SELECT * FROM " +
                                "Usuarios WHERE Clave = '" + clave +
                                "' AND pwd = '" + pwd +
                                "';" );
        int cont = 0;
        while(rs.next()){
            cont++;
        }
        if(cont == 0)
            return false;
        else
            return true;
    }catch(SQLException e){
        System.out.println("error" + e.getMessage());
        return false;
    }
}

/**
 * Devuelve el nombre de registro del usuario
 * una vez que se ha autorizado
 * @param clave El nombre de usuario
 * @param pwd El password
 * @return el nombre del usuario
 * @throws SQLException Lanzada si no se ejecuta correctamente el query
 */

public String nombreUsuario(String clave, String pwd)
    throws SQLException {
    String nombre = "";
    try{
        stmt=conexion.createStatement();
        rs = stmt.executeQuery( "SELECT Nombre FROM " +
                                "Usuarios WHERE Clave = '" + clave +
                                "' AND pwd = '" + pwd +
                                "';" );
        int cont = 0;
        while(rs.next()){
            nombre = rs.getString("Nombre");
        }
    }catch(SQLException e){
        System.out.println("error" + e.getMessage());
    }
    return nombre;
}
```

```
}
```

- Compila y coloca el archivo *.class* en la carpeta *WEB-INF/classes*.

Nota: Se sugiere que la carpeta *WEB-INF* esté directamente dentro de la carpeta principal de la aplicación, a pesar de la estructura creada por NetBeans.

3.4. Realización de Pruebas Unitarias con JUnit

Afortunadamente, desde NetBeans también tenemos la posibilidad de hacer nuestras pruebas unitarias con JUnit, basta tener instalado éste último para poder hacer uso de él desde la ventana principal del IDE.

- Con la clase ConexionBD.java en la ventana de NetBeans, da click en *Tools→Create JUnit Tests*. Indica en qué carpeta se guardará el código de la clase ConexionBDTest.java (por default es en la carpeta *Test Packages*).
- Selecciona características como las firmas de los métodos y generación de comentarios adicionales.
- Da click en *OK*.
- Se genera automáticamente el código para realizar las pruebas, sin embargo, hay que hacer algunas modificaciones:
  - Elimina los *fails* del código.
  - Es recomendable colocar las variables globales en la parte superior (como tradicionalmente) e inicializarlas en el método *setUp*. Por ejemplo, inicializa las variables *clave* y *pwd* con valores existentes en la base de datos (en el ejemplo, *clave = "usuario1"*, *pwd = "pwd1"*).
  - Borra las nuevas instancias de éstas variables de cada método (ya están declaradas las globales).

    El código quedaría así:

```
import junit.framework.*;
import java.sql.*;
/*
 * ConexionBDTest.java
 * JUnit based test
 */

/**
 *
 * @author Curso IS
 */
```

```
public class ConexionBDTest extends TestCase {

    ConexionBD instance;
    String clave;
    String pwd;

    public ConexionBDTest(String testName) {
        super(testName);
    }

    protected void setUp() throws Exception {
        instance = new ConexionBD();
        clave = "usuario";
        pwd = "password";
    }

    protected void tearDown() throws Exception {
    }

    /**
     * Test of desconectar method, of class ConexionBD.
     */
    public void testDesconectar() throws Exception {
        System.out.println("desconectar");
        instance.desconectar();
    }

    /**
     * Test of verifica method, of class ConexionBD.
     */
    public void testVerifica() throws Exception {
        System.out.println("verifica");
        // expResult local, de tipo boolean
        boolean expResult = true;
        boolean result = instance.verifica(clave, pwd);
        assertEquals(expResult, result);
    }

    /**
     * Test of nombreUsuario method, of class ConexionBD.
     */
    public void testNombreUsuario() throws Exception {
        System.out.println("nombreUsuario");
        // expResult local, de tipo String
        String expResult = "Pancho Lopez";
```

```
            String result = instance.nombreUsuario(clave, pwd);
            assertEquals(expResult, result);
        }

    }
```

- Localiza esta clase (ConexionBDTest.java) en la ventana *Projects*, da click derecho y elige *Compilar*.
- Lo mismo para ejecutar (Fig. 3.5).

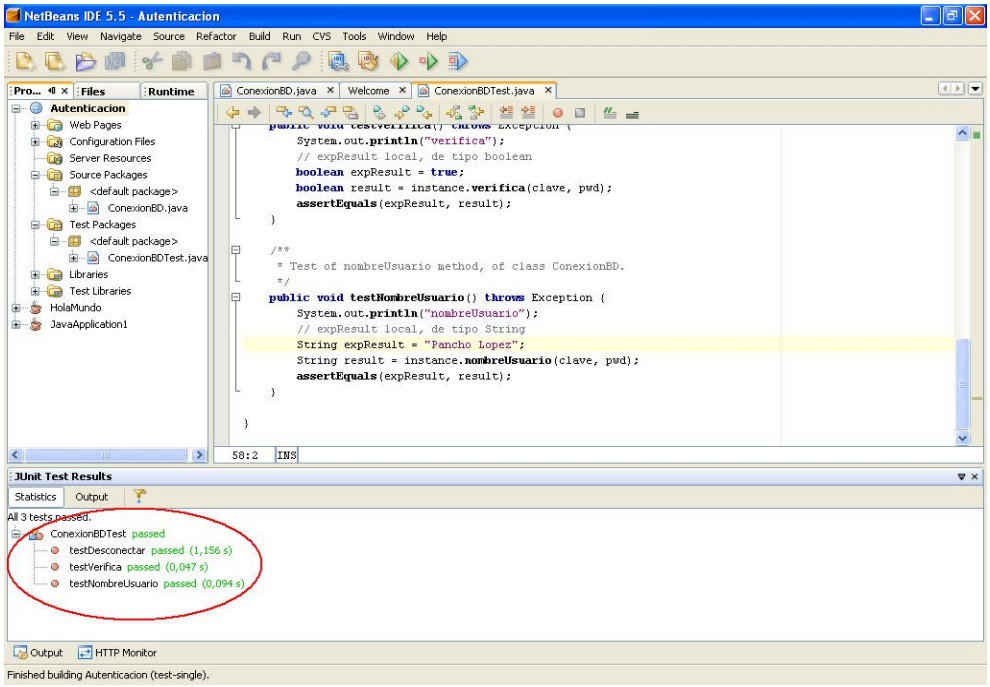

Figura 3.5: Prueba exitosa.

A continuación se muestran los archivos JSP que se muestran como páginas para acceder al sistema. Los cuales se colocan en la carpeta principal del proyecto (Autenticacion).

- Archivo index.jsp:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <title>Autenticacion</title>
  </head>
  <body>
  <%@ page language="java" %>
  <center>
  <h1>Autenticaci&oacute;n</h1>
```

```
  <table>
  <tr>
  <td>
  <form method="post" action="aut2.jsp" name="autenticacion">
  <table>
  <tr>
  <td>
  <b>Usuario</b></td>
  <td>
  <input type="text" name="login">
  </td>
  </tr>
  <tr>
  <td>
  <b>Password</b>
  </td>
  <td>
  <input type="password" name="pwd">
  </td>
  </tr>
  <tr>
  <td align=center colspan=2>
  <input type="submit" value="Entrar">
  </td>
  </tr>
  </table>
  </form>
  </td>
  </tr>
  </table>
  </center>
  </body>
</html>
```

- Archivo que realiza la verificación:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <body>
  <%@ page language="java" %>
  <%! String login = ""; %>
  <%! String pwd = ""; %>
  <%! String redireccionURL = ""; %>
  <jsp:useBean id="conx" scope="session" class="Autenticacion.ConexionBD"/>
```

```
  <body>
  <%
  conx.conectar();
  // Obtenemos los parámetros que ha introducido el usuario
  login = request.getParameter( "login" );
  pwd = request.getParameter( "pwd" );
  if (conx.verifica(login,pwd)){
  redireccionURL = "autorizado.jsp";
  }
  else{
  redireccionURL = "noautorizado.html";
  }
  conx.desconectar();
  %>
  <!-- Funcion en JavaScript que redirecciona dependiendo si
el usuario esta o no autorizado-->
  <script language="javascript">
  setTimeout( "document.location.href='
  <%= redireccionURL %>?login=<%= login %>&pwd=<%= pwd %>'",100)
  </script>
  </body>
</html>
```

- Archivo en caso de autenticación exitosa, se muestra el nombre del usuario.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <title>Validado</title>
  </head>
  <jsp:useBean id="conx" scope="session" class="Autenticacion.ConexionBD"/>
  <body>
  <center>
  <%String login=request.getParameter("login");%>
  <%String pwd=request.getParameter("pwd");%>
  <%String nombre="";%>
  <%
  conx.conectar();
  nombre = conx.nombreUsuario(login,pwd);
  conx.desconectar();
  %>
  <h1>Bienvenido al Sistema, <b> <%= nombre %></b></h1><br>
  <a href="index.jsp">Ir al Inicio</a>
  </center>
```

```
  <hr>
  </body>
</html>
```

- Página que se muestra en caso de autenticación fallida

```
<html>
  <head>
  <title>Login - Rechazado</title>
  <!-- Tras 4 segundos se devuelve a la pagina de inicio-->
  <META http-equiv="refresh" Content="4; URL=index.jsp">
  </head>
  <body>
  <center>
  <h1>Usuario no autorizado</h1>
  Cerciórese de teclear el usuario y el password correctamente.
  </center>
  </body>
</hmtl>
```

- En el navegador, da click en *Tomcat Manager* y proporciona el nombre de usuario y contraseña que diste al instalarlo o bien, crea un nuevo usuario *manager* editando el archivo *tomcat-users* de la carpeta *conf* de Tomcat.
- Localiza la aplicación **Autenticacion** y da click sobre ésta.
- Se muestra la página de inicio con los campos para la clave y contraseña (Fig 3.6). En caso de teclear correctamente el nombre de usuario y la contraseña, se muestra la pantalla de éxito. En el ejemplo, Usuario = usuario1; Password = pwd1 (Fig 3.7). En caso contrario se muestra la página de acceso fallido (Fig 3.8). Tras 4 segundos se devuelve a la página de inicio.

En algunos casos, aunque se modifique el archivo *tomcat-users*, esto no surte efecto para acceder a Tomcat como manager, y como consecuencia, no es posible usar el servidor que viene incluído en la instalación de NetBeans. Para solucionar este problema, si ya tienes instalado en tu computadora un servidor Tomcat, puedes añadirlo a NetBeans para tener control sobre él desde tu IDE, usando el mismo puerto, clave de usuario y contraseña indicados al instalarlo.

- Da click en la barra *Tools* → *Server Manager* → *Add Server* y selecciona la ruta de tu servidor Tomcat previamente instalado.
- Da click en la ventana *Runtime* y localiza el servidor agregado. Igualmente puedes, levantarlo, detenerlo, abrir tus aplicaciones en el navegador, etc.
- Copia o mueve tu aplicación de la carpeta *webapps* del anterior servidor Tomcat en uso a la carpeta *webapps* del nuevo.

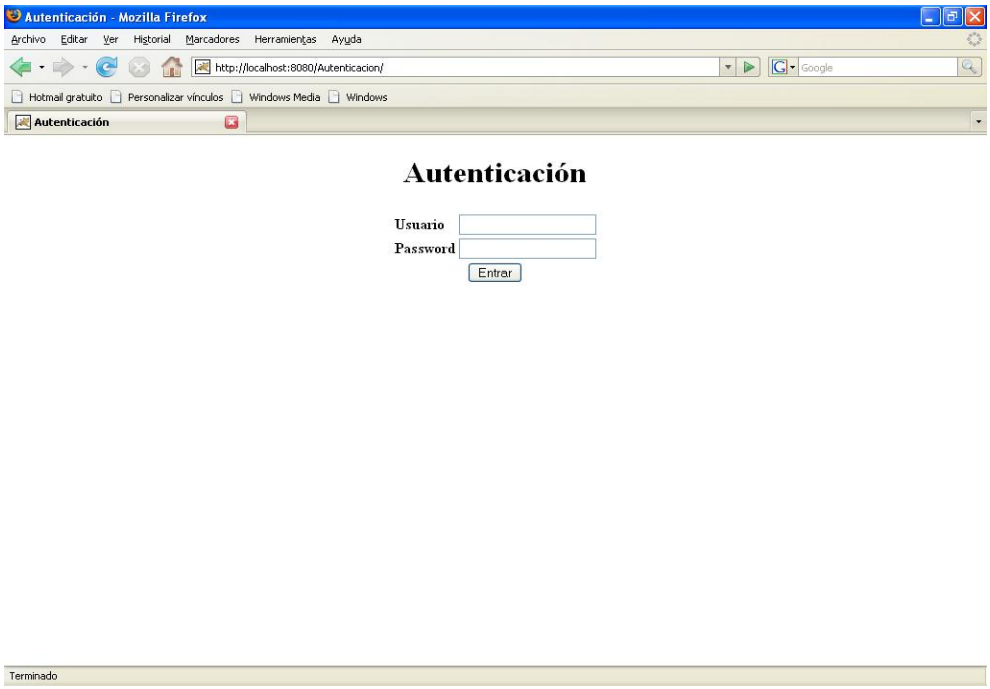

Figura 3.6: Página principal.

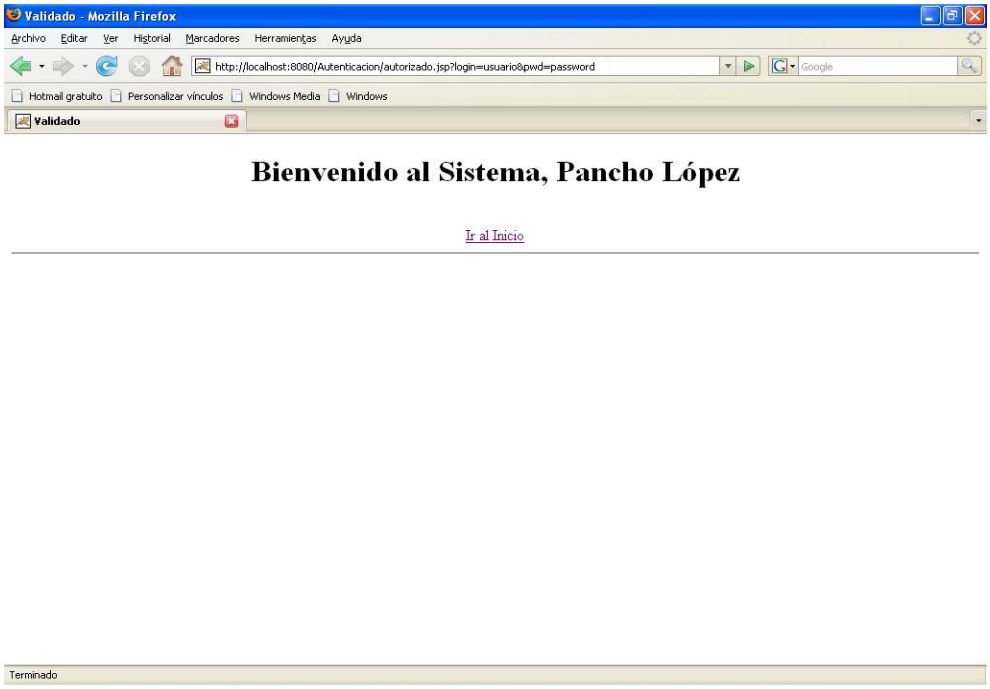

Figura 3.7: Acceso exitoso.

Figura 3.8: Acceso fallido.

3.5. Control de Versiones con Subversion

A continuación se muestran los principales pasos para poder manejar los versiones del código a través de Subversion:

- Crea el repositorio que albergará la aplicación (recuerda que es preferible en una carpeta cuyo nombre y ruta no contengan espacios):

```
> svnadmin create Autenticacion
```

- Modifica nuevamente el archivo **httpd.conf** de la carpeta **conf** de Apache para que "apunte" a tu nuevo repositorio. Las últimas líneas quedarían:

```
<Location /repositorios>
        DAV svn
        SVNPath C:/Repositorios/Autenticacion
</Location>
<Location /repositorios>
        DAV svn
        SVNPath C:/Repositorios/Autenticacion
        AuthType Basic
        AuthName "Repositorio Subversion"
        AuthUserFile C:/Apache2/bin/usuarios.txt
        require valid-user
```

```
        <LimitExcept GET PROPFIND OPTIONS REPORT>
        Require valid-user
        </LimitExcept>
</Location>
```

Con lo que sería llamado desde el navegador como **http://localhost/repositorios**. Además, se solicita la autentificación proporcionada anteriormente (cuando se mostró cómo hacerla en el tutorial de Subversion).

- Ejecuta:

  ```
  > apache -e debug
  ```

  para verificar que todo haya sido modificado correctamente.

- Crea la copia de trabajo. Suponiendo que la copia se llamará 'autenti', teclea:

  ```
  > svn checkout file:///Repositorios/Autenticacion autenti
  ```

- Copia tu aplicación *Autenticacion* de *webapps* en *autenti*. Muévete a esta carpeta y escribe:

  ```
  > svn add Autenticacion
  > svn commit Autenticacion -m 'autenticacion_agregado'
  ```

  Teclea la dirección en el navegador, proporciona tu clave de usuario y contraseña y debe mostrarse tu proyecto. Al dar click en éste, se muestran las carpetas y archivos contenidos (Fig. 3.9).

¡Listo!, tenemos una aplicación completa empleando todas las herramientas presentadas en los tutoriales individuales.

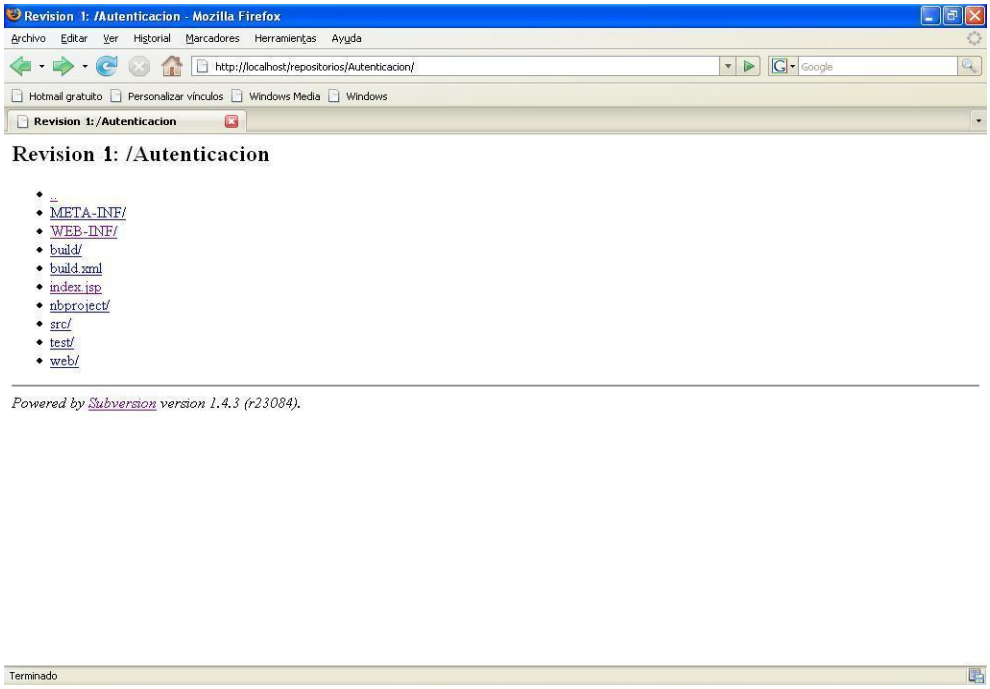

Figura 3.9: Carpetas contenidas en *Autenticacion*.

Capítulo 4

Ejemplo de Aplicación Web con Eclipse: Autenticación

Es prácticamente la misma aplicación desarrollada en el capítulo anterior, pero mostrando la forma de llevarla a cabo sólo con el IDE Eclipse.

Se trata de hacer un software para una autenticación básica para ingresar a un sistema proporcionando clave de usuario y contraseña. A pesar de su simplicidad, muestra la forma de desarrollar una aplicación web con Eclipse.

Los nombres, claves y contraseñas de los usuarios se encuentran en la tabla *Usuarios*. Para ser autorizado, el usuario debe teclear la clave y la correspondiente contraseña. En caso de coincidir con los registrados en la base de datos, el sistema direcciona a la página *autorizado.jsp* donde se busca el nombre del usuario en la base de datos y se imprime en la pantalla. En caso contrario, se direcciona a la página *noautorizado.jsp* donde se muestra un mensaje solicitando que el usuario verifique la clave y password tecleados y, tras 4 segundos, se redirecciona nuevamente a la página inicial.

Adicionalmente, se puede asignar un valor para indicar si el usuario ya ha sido autorizado a visitar otras páginas del sistema y no acceda "por fuera", es decir, tecleando la url directamente en la barra de direcciones. Para esto se puede hacer uso de *sesiones*.

4.1. Creación de la Base de Datos en MySQL

Empezaremos por crear la base de datos en MySQL:

- En MySQL se crea la base de datos *Autenticacion*. Y dentro de ésta, la tabla *Usuarios*, cuyos atributos son:
  - Clave: La clave del usuario, que es la clave primaria.
  - Pwd: El password del usuario.
  - Nombre: El nombre del usuario, es devuelto una vez que el usuario ha sido autorizado.

Los comandos en MySQL son los siguientes:

```
CREATE DATABASE Autenticacion;
use autenticacion
CREATE TABLE USUARIOS(
Clave VARCHAR (30) NOT NULL,
Pwd VARCHAR (30) NOT NULL,
Nombre VARCHAR (70) NOT NULL,
PRIMARY KEY(Clave)
);
```

Podemos insertar directamente una tupla en la tabla, aunque por supuesto, será más interesante implementar el registro a través del propio sistema.

```
INSERT INTO Usuarios VALUES('usuario1', 'pwd1', 'Pancho Lopez');
```

4.2. Construcción del Proyecto con Eclipse

Ahora vamos a la creación del proyecto en Eclipse.

- Da click derecho sobre la carpeta *WEB-INF/src* y selecciona *New* → *Package*.
- Asigna el nombre, en este caso, **Autenticación**.
- Dentro de éste, crea la clase *ConexionBD.java*. El código correspondiente es el siguiente:

```
/**
 *
 * @author Curso IS
 */
package Autenticacion;

import java.sql.*;

public class ConexionBD{

    // Variables para el control de la BD

    private String bDatos;
    private String servidor;
    private String user;
    private String password;
    private String driver;
    private Connection conexion;
    private Statement stmt;
```

```
private ResultSet rs;

/**
 * Crea una instancia de ConexionBD
 * asi como la conexion con la BD
 * @throws SQLException Lanzada si no se establece correctamente la conexion
 */

public ConexionBD() throws SQLException{
    bDatos = "Autenticacion";
    servidor = "localhost";
    user = "root";
    password = "admin";
    driver = "com.mysql.jdbc.Driver";
    try{
        Class.forName(driver);
        conexion = DriverManager.getConnection("jdbc:mysql://" +
                                               servidor + ":3306/" +
                                               bDatos , user, password);
    }catch(Exception e){
        System.out.println("Error al conectar: " + e.getMessage());
    }
}

/**
 * Cierra la conexion con la BD
 * @throws SQLException Lanzada si no se cierra correctamente la conexion
 */

public void desconectar() throws SQLException{
    try{
        conexion.close();
    }catch(SQLException ex){
        System.out.println("SQLException: " + ex.getMessage());
    }
}

/**
 * Hace la validación del usuario, si coinciden la clave de
 * usuario y el password, se permite el acceso.
 * @param clave la clave de usuario
 * @param pwd El password
 * @return Un valor indicando si el usuario ha sido o no autorizado
 * @throws SQLException Lanzada si no se realiza correctamente el query
 */
```

```
public boolean verifica(String clave, String pwd)
    throws SQLException {
    try{
        stmt=conexion.createStatement();
        rs = stmt.executeQuery( "SELECT * FROM " +
                                "Usuarios WHERE Clave = '" + clave +
                                "' AND pwd = '" + pwd +
                                "';" );
        while(rs.next()){
            cont++;
        }
        if(cont == 0)
            return false;
        else
            return true;
    }catch(SQLException e){
        System.out.println("error" + e.getMessage());
        return false;
    }
}

/**
 * Devuelve el nombre de registro del usuario
 * una vez que se ha autorizado
 * @param clave El nombre de usuario
 * @param pwd El password
 * @return el nombre del usuario
 * @throws SQLException Lanzada si no se realiza correctamente el query
 */

public String nombreUsuario(String clave, String pwd)
    throws SQLException {
    String nombre = "";
    try{
        stmt=conexion.createStatement();
        rs = stmt.executeQuery( "SELECT Nombre FROM " +
                                "Usuarios WHERE Clave = '" + clave +
                                "' AND pwd = '" + pwd +
                                "';" );
        int cont = 0;
        while(rs.next()){
            nombre = rs.getString("Nombre");
        }
    }catch(SQLException e){
```

```
            System.out.println("error" + e.getMessage());
        }
        return nombre;
    }
}
```

A continuación se presentan los archivos JSP que se muestran como páginas para acceder al sistema. Los cuales se colocan en la carpeta principal del proyecto (Autenticacion).

- Archivo index.jsp:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <title>Autenticacion</title>
  </head>
  <body>
  <%@ page language="java" %>
  <center>
  <h1>Autenticaci&oacute;n</h1>
  <table>
  <tr>
  <td>
  <form method="post" action="aut2.jsp" name="autenticacion">
  <table>
  <tr>
  <td>
  <b>Usuario</b></td>
  <td>
  <input type="text" name="login">
  </td>
  </tr>
  <tr>
  <td>
  <b>Password</b>
  </td>
  <td>
  <input type="password" name="pwd">
  </td>
  </tr>
  <tr>
  <td align=center colspan=2>
  <input type="submit" value="Entrar">
  </td>
  </tr>
```

```
  </table>
  </form>
  </td>
  </tr>
  </table>
  </center>
  </body>
</html>
```

- Archivo que realiza la verificación:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <body>
  <%@ page language="java" %>
  <%! String login = ""; %>
  <%! String pwd = ""; %>
  <%! String redireccionURL = ""; %>
  <jsp:useBean id="conx" scope="session" class="Autenticacion.ConexionBD"/>
  <body>
  <%
  conx.conectar();
  // Obtenemos los parámetros que ha introducido el usuario
  login = request.getParameter( "login" );
  pwd = request.getParameter( "pwd" );
  if (conx.verifica(login,pwd)){
  redireccionURL = "autorizado.jsp";
  }
  else{
  redireccionURL = "noautorizado.html";
  }
  conx.desconectar();
  %>
  <!-- Funcion en JavaScript que redirecciona dependiendo
si el usuario esta o no autorizado-->
  <script language="javascript">
  setTimeout( "document.location.href='
  <%= redireccionURL %>?login=<%= login %>&pwd=<%= pwd %>'",100)
  </script>
  </body>
</html>
```

- Archivo en caso de autenticación exitosa, se muestra el nombre del usuario.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <title>Validado</title>
  </head>
  <jsp:useBean id="conx" scope="session" class="Autenticacion.ConexionBD"/>
  <body>
  <center>
  <%String login=request.getParameter("login");%>
  <%String pwd=request.getParameter("pwd");%>
  <%String nombre="";%>
  <%
  conx.conectar();
  nombre = conx.nombreUsuario(login,pwd);
  conx.desconectar();
  %>
  <h1>Bienvenido al Sistema, <b> <%= nombre %></b></h1><br>
  <a href="index.jsp">Ir al Inicio</a>
  </center>
  <hr>
  </body>
</html>
```

- Página que se muestra en caso de autenticación fallida

```
<html>
  <head>
  <title>Login - Rechazado</title>
  <!-- Tras 4 segundos se devuelve a la pagina de inicio-->
  <META http-equiv="refresh" Content="4; URL=index.jsp">
  </head>
  <body>
  <center>
  <h1>Usuario no autorizado</h1>
  Cerciórese de teclear el usuario y el password correctamente.
  </center>
  </body>
</hmtl>
```

Para ejecutar la aplicación:

- Puedes dar click derecho sobre el archivo *index.jsp* en el *Package Explorer* y seleccionar *Run as* → *Run on Server* o en la barra superior de herramientas de Eclipse, oprime el botón *Open Web Browser* (Fig. 4.1).

- En la barra de direcciones teclea **http://localhost:8080/AutenticaEclipse** (Fig. 4.2).
- Proporciona los datos requeridos (Fig. 4.3). En caso de que la autentificación falle, se redireciona a la página de error tras 4 segundos (Fig. 4.4). En otro caso, el sistema te da la bienvenida (Fig. 4.5).

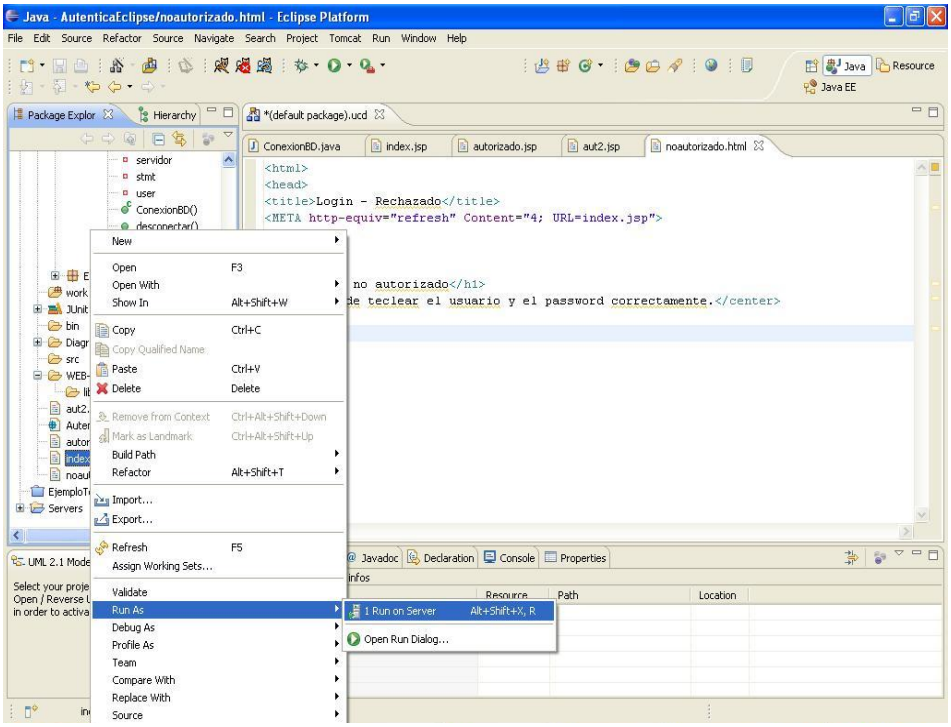

Figura 4.1: Ejecución del archivo index.jsp en el servidor.

4.3. Pruebas con el Plugin de JUnit

Eclipse viene con un plugin para poder usar JUnit desde el mismo IDE y generar código automáticamente. Por ejemplo, vamos a generar la clase de prueba de la clase *ConexionBD*.

- En el *Package Explorer* da click derecho sobre esta clase y selecciona *New* → *New Junit Test Case*.
- A continuación selecciona los métodos a probar, en este caso, probaremos los 4 métodos (Fig. 4.6).
- Se genera el esqueleto de la clase de prueba, sin embargo, la complementaremos para que tenga el siguiente código:

```
package entidad;
```

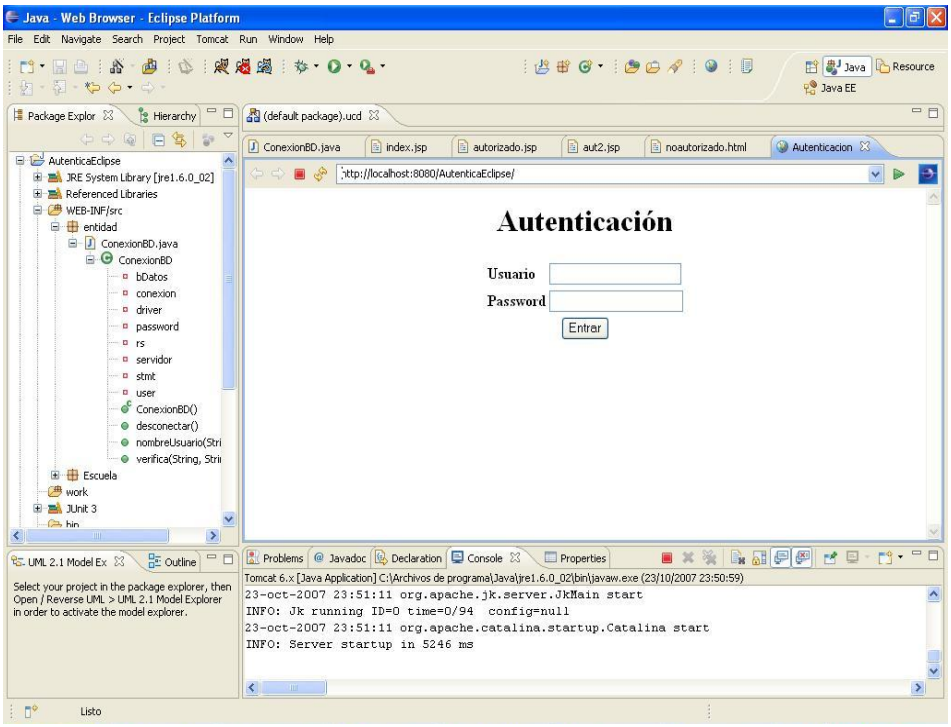

Figura 4.2: Página de inicio de la aplicación.

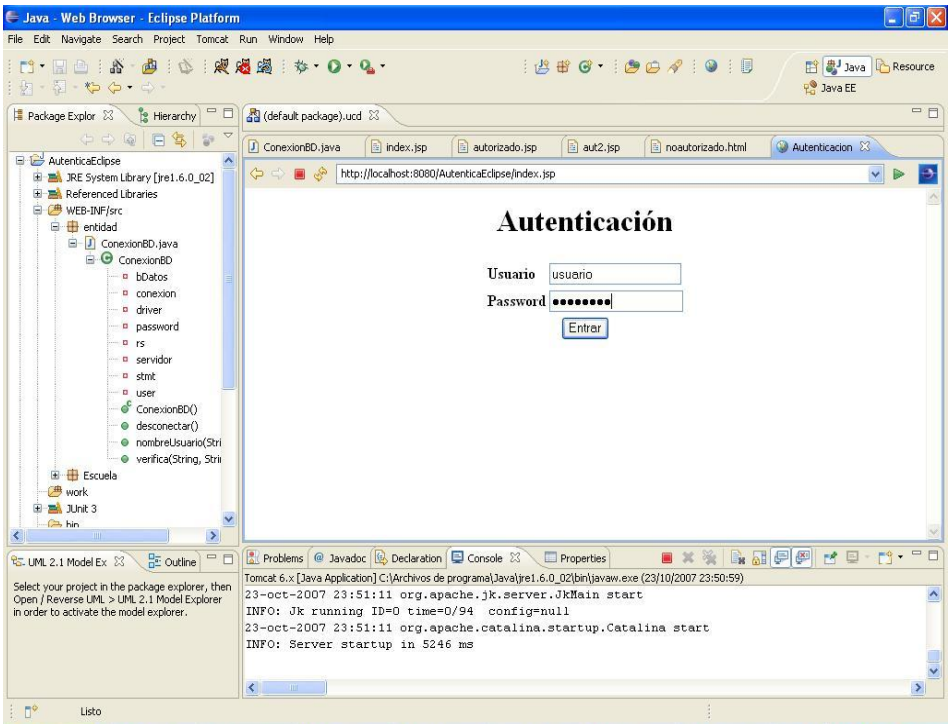

Figura 4.3: Proporcionando los datos.

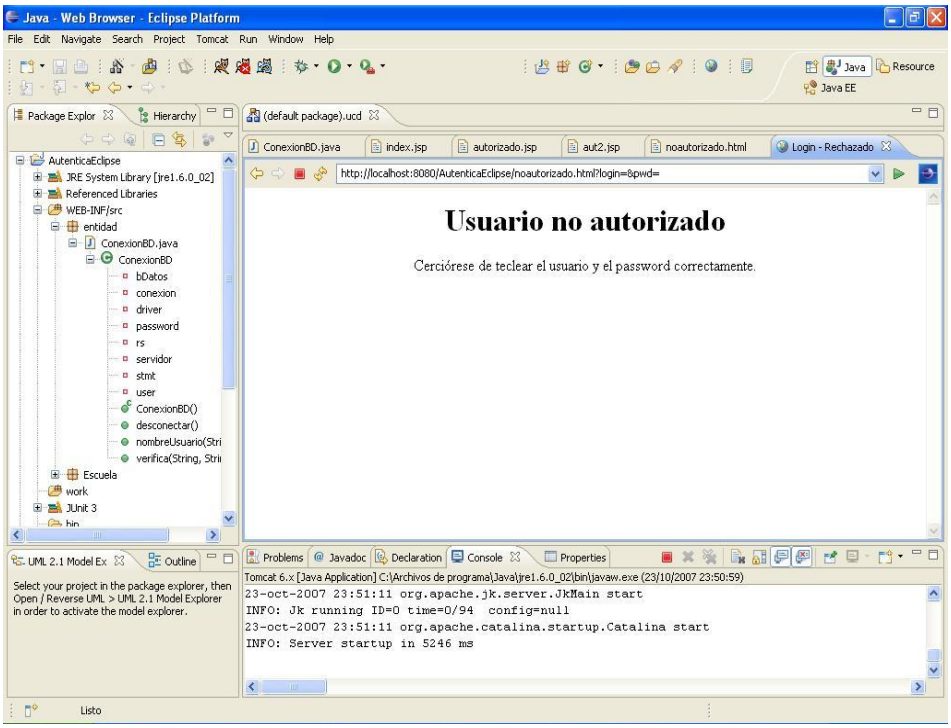

Figura 4.4: En caso de error en la autenticación, se direcciona a esta página.

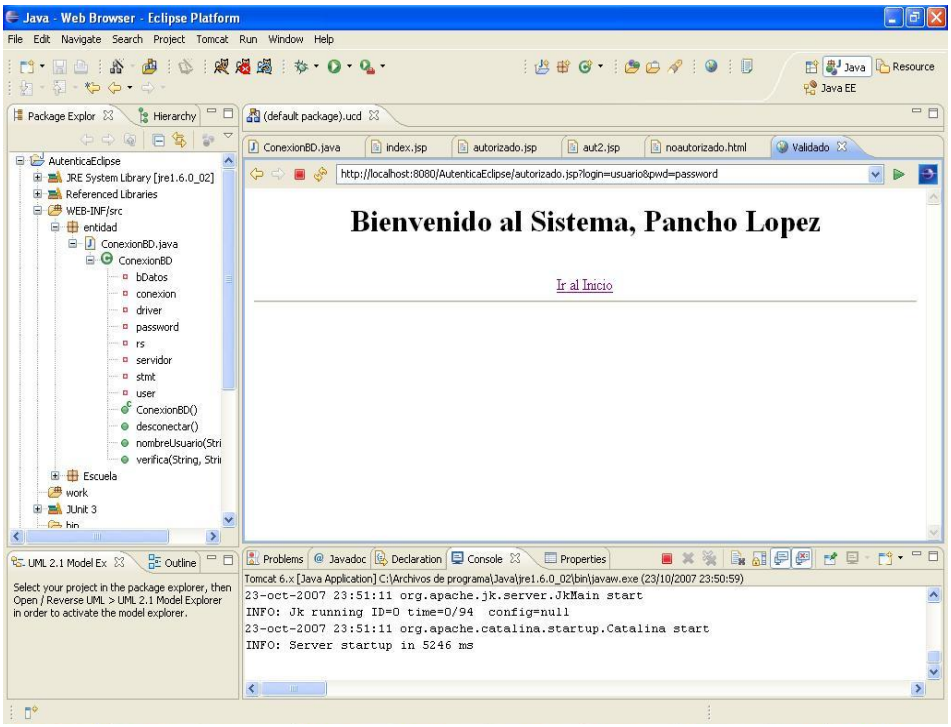

Figura 4.5: Autenticación exitosa.

```
import junit.framework.TestCase;

public class ConexionBDTest extends TestCase {

        ConexionBD instance;
        String clave;
        String pwd;
        String expResult;

        protected void setUp() throws Exception {
                super.setUp();
                instance = new ConexionBD();
                clave = "usuario1";
                pwd = "pwd1";
                expResult = "Pancho Lopez";
        }

        public void testVerifica() throws Exception {
                // expResult local, de tipo boolean
                boolean expResult = true;
                boolean result = instance.verifica(clave, pwd);
                assertEquals(expResult, result);
        }

        public void testNombreUsuario() throws Exception {
                String result = instance.nombreUsuario(clave, pwd);
                assertEquals(expResult, result);
        }

        public void testDesconectar() throws Exception {
                instance.desconectar();
        }
}
```

- Para correr la clase de prueba, basta hacer click derecho sobre el código fuente de ésta y seleccionar *Run as → Run JUnit Test*.
- Se muestra una nueva paleta con el resultado de la ejecución (Fig. 4.7).

4.4. Generación de Javadoc

Para generar la documentación, típicamente se usa la herramienta **javadoc** incluida en el JDK de Java. Por supuesto, ésta puede ser usada desde Eclipse. Como ejemplo, vamos a generar la documentación de la clase *AutenticacionBD*.

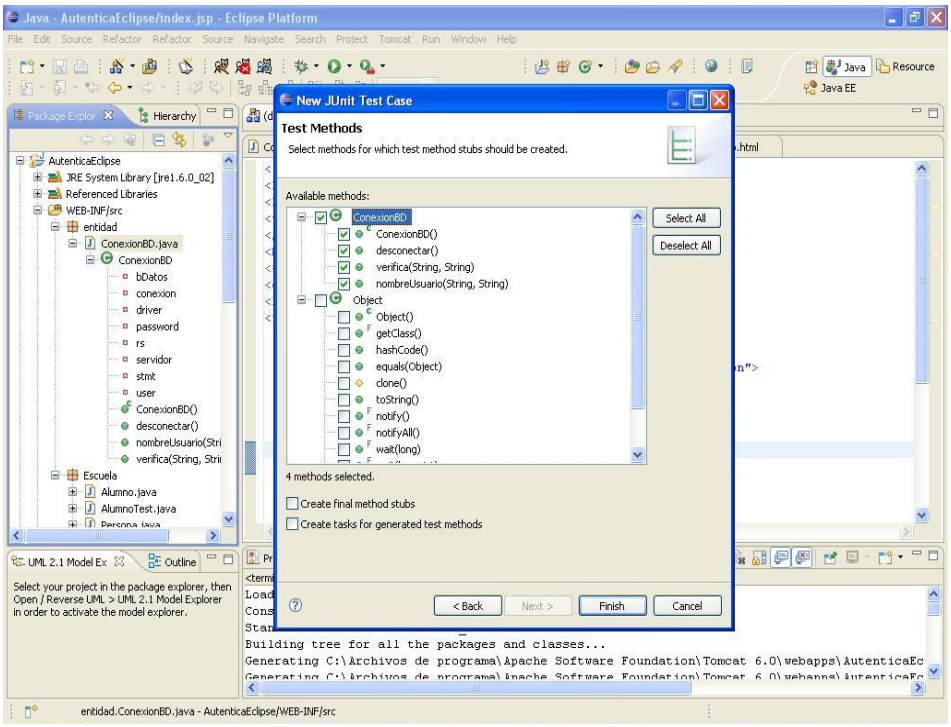

Figura 4.6: Nueva clase de prueba con JUnit.

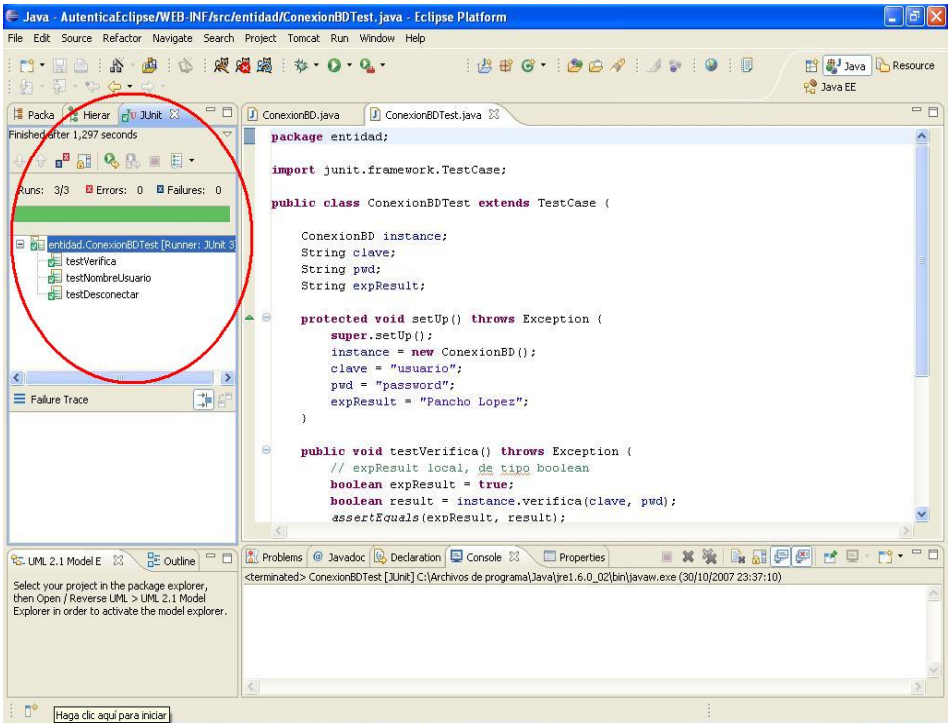

Figura 4.7: Ejecución de la prueba.

- En el *Package Explorer* da click derecho sobre la clase y selecciona *Export* → *Java* → *Javadoc* → *Next* (Fig. 4.8).
- Selecciona las opciones deseadas e indica el comando *javadoc* (Fig. 4.9). Para esto, coloca la ubicación del archivo *javadoc* de la carpeta *bin* de tu JDK.
- *Finish* para terminar.
- La documentación se guarda por default en la carpeta *doc* de la aplicación. Está en formato HTML (Fig. 4.10).

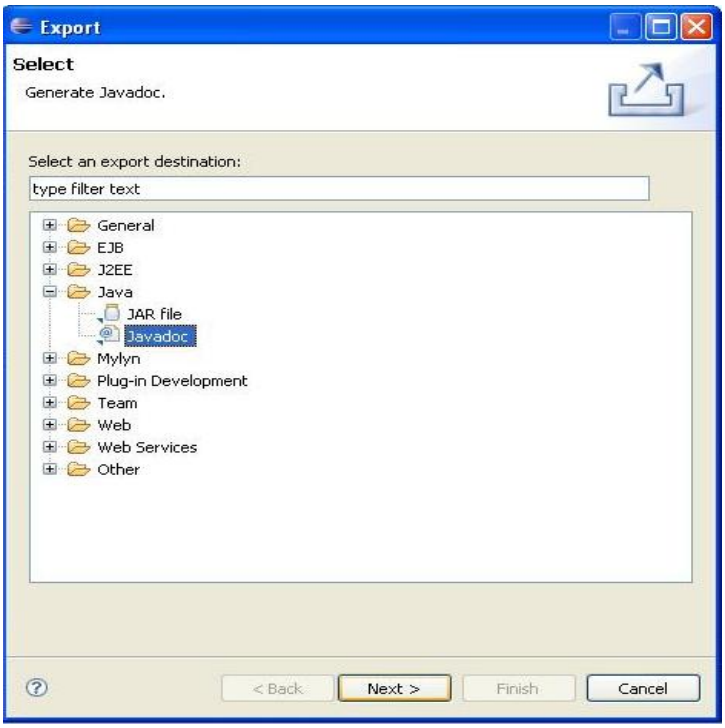

Figura 4.8: Generación de documentación.

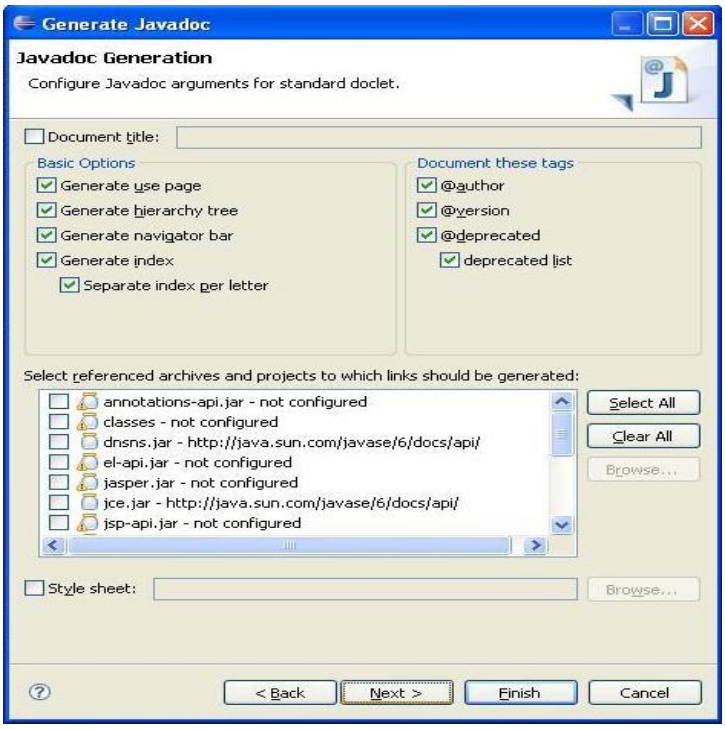

Figura 4.9: Selección de opciones complementarias.

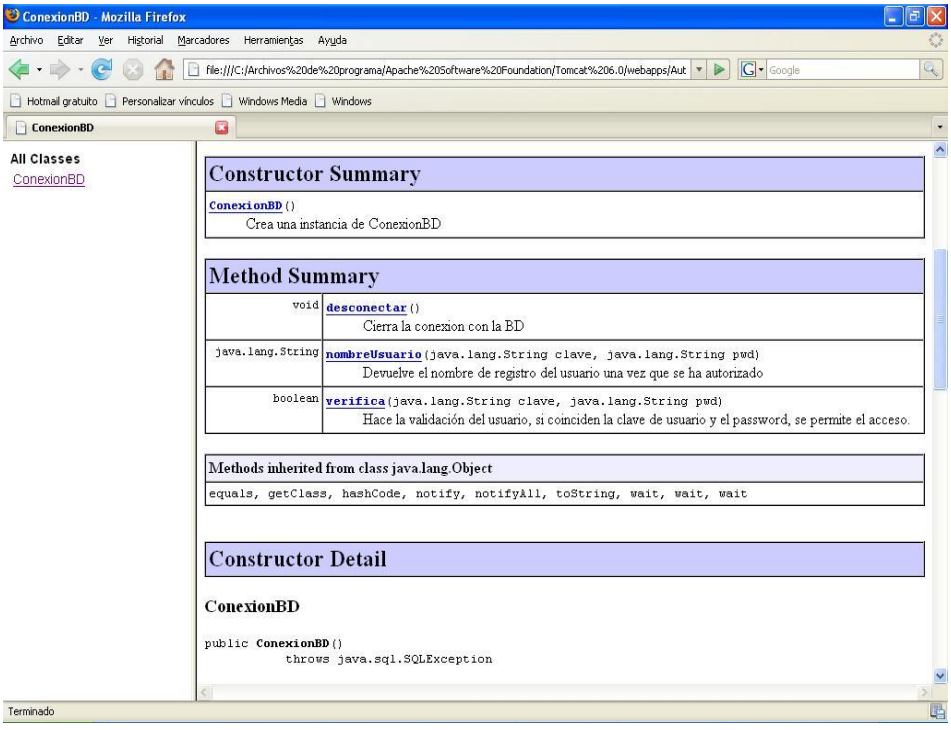

Figura 4.10: Documentación de la clase ConexionBD.

Capítulo 5

Incorporación del Material en el Sitio de Apoyo al Curso de Ingeniería de Software

El sitio de apoyo al curso de la asignatura de Ingeniería de Software contiene diversas secciones para proporcionar al alumno herramientas que le permitan un mejor aprovechamiento, tales como el libro "Ingeniería de Software Pragmática", una sección de material de apoyo a este libro, con diagramas y ejemplos, sección de prácticas, sección de ligas de interés y una sección para que los alumnos consulten sus calificaciones, así como otra donde los profesores y ayudantes asignan calificaciones. Además, la sección que nos ocupa: la sección de *Herramientas*. Para acceder al sitio y, en particular, a la sección, hay que teclear la dirección:
http://victoria.fciencias.unam.mx/cursois/index.html.

Se parte de la liga *Herramientas* del menú de opciones en el marco izquierdo de la página del curso, para acceder al material (Fig. 5.1). Ya en la sección, se presenta el siguiente texto:

5.1. Pasos para la Incorporación del Material

Para incorporar este material en el sitio del curso de Ingeniería de Software, se llevaron a cabo los siguientes pasos:

- Acceder remotamente con la herramienta *SSH Secure Shell Client* al servidor **Victoria**, donde reside el sitio.
- Descargar la carpeta con los archivos html que corresponden al sitio del curso a través de la herramienta *SSH Secure File Transfer Client*.
- Modificar el archivo **herramientas.html**, que corresponde a la sección **Herramientas** en el sitio, para incluir las nuevas ligas a las subsecciones y documentos PDF con la estructura mostrada(Fig. 5.2): El contenido de los archivos es el siguiente:

Figura 5.1: Sitio de Apoyo del Curso de Ingeniería de Software.

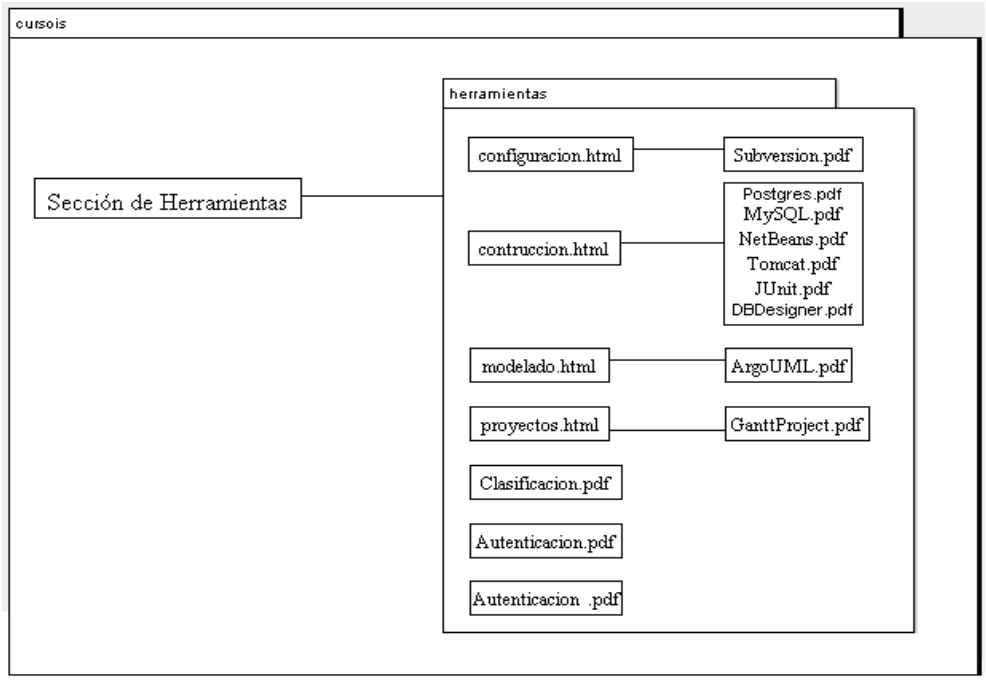

Figura 5.2: Estructura de la sección *Herramientas*.

5.2. Contenido de los Archivos de la Carpeta *Herramientas*

- *configuracion.html*: Contiene la lista de herramientas correspondientes a la Fase de Estrategia de acuerdo al curso. Se presentan dos herramientas para la Administración de la Configuración: Subversion y CVS (Concurrent Versions System).
  - *Subversion.pdf*: Tutorial de la herramienta Subversion, desarrollado en el presente trabajo.
- *construccion.html*: Contiene la lista de herramientas correspondientes a la Fase de Construcción y Pruebas Unitarias de acuerdo al curso. Se presentan las herramientas para Diseño y Construcción de Bases de Datos: MySQL y PostgreSQL; los IDEs (Entornos de Desarrollo Integrados) NetBeans y Eclipse y el servidor de aplicaciones web Tomcat; además, la herramienta para Pruebas: JUnit.
  - *MySQL.pdf*: Tutorial de la herramienta MySQL, desarrollado en el presente trabajo.
  - *Postgres.pdf*: Tutorial de la herramienta PostgreSQL, desarrollado en el presente trabajo.
  - *DBDesigner.pdf*: Tutorial de la herramienta DB Designer Fork, desarrollado en el presente trabajo.
  - *NetBeans.pdf*: Tutorial de la herramienta NetBeans, desarrollado en el presente trabajo.
  - *Eclipse.pdf*: Tutorial de la herramienta Eclipse, desarrollado en el presente trabajo.
  - *Tomcat.pdf*: Tutorial de la herramienta Tomcat, desarrollado en el presente trabajo.
  - *JUnit.pdf*: Tutorial de la herramienta JUnit, desarrollado en el presente trabajo.
- *modelado.html*: Contiene la lista de herramientas correspondientes a la Fase de Diseño de acuerdo al curso. Se presentan las herramientas para el Modelado UML: ArgoUML, Dia, Poseidon y Visio.
  - *ArgoUML.pdf*: Tutorial de la herramienta ArgoUML, desarrollado en el presente trabajo.
- *proyectos.html*: Contiene la lista de herramientas correspondientes a la Fase de Planeación de acuerdo al curso. Se presentan las herramientas de Administración de Proyectos: GanttProject y Microsoft Project.
  - *GantProject.pdf*: Tutorial de la herramienta GanttProject, desarrollado en el presente trabajo.
- *Clasificacion.pdf*: Artículo "Una Clasificación del Software", desarrollado en el capítulo 1 del presente trabajo. Se accede a partir de la liga **herramientas de distribución gratuita** a modo de profundización en el tema.
- *Autenticacion.pdf*: Ejemplo de aplicación web desarrollado en el capítulo 3 del presente trabajo.

- *AutenticacionE.pdf*: Ejemplo de aplicación web con Eclipsedesarrollado en el capítulo 4 del presente trabajo.

- Subir los nuevos archivos al sitio a través de *SSH Secure File Transfer Client*.

- Por último, desde la consola de *SSH Secure Shell Client*, modificar los permisos de lectura, escritura y ejecución para que queden igual al resto de archivos que se encuentran en el sitio. Por ejemplo, para cambiar los permisos de los archivos html, con el comando:

  ```
  chmod 775 *.html
  ```

  Esto con el objeto de facilitar futuras modificaciones a los archivos por más personas, cuando sea necesario.

Conclusiones

Desde hace algunos años, la Ingeniería de Software ha sido una de las actividades productivas con más crecimiento, pues cada vez más y más empresas e instituciones requieren de la automatización de procesos y servicios, así como nuevas formas de comercio, de entretenimiento o, simplemente, nuevas formas de comunicación, que pueden satisfacerse con aplicaciones de software. Con esta creciente demanda, esta rama de la Computación, ha ido evolucionando a pasos agigantados, por lo que actualmente es toda una compleja disciplina que reúne diversas actividades. Afortunadamente para quienes hemos incursionado o pretendemos incursionar en este campo, el facilitar y agilizar muchas de esas actividades ha sido el objetivo del trabajo de diversos grupos de desarrollo, el cual ha desembocado en la generación de las *herramientas de apoyo para el desarrollo de software*, motivo del presente trabajo.

A lo largo de las páginas anteriores, se ha dado a conocer a los alumnos de la materia de Ingeniería de Software, la instalación y el uso, así como un poco de historia de algunas herramientas de apoyo que les permitan un mejor aprovechamiento del curso en aspectos como la metodología, el trabajo en equipo y la experiencia de un primer contacto con el desarrollo integral de una aplicación.

Cabe destacar que al cierre de la redacción de este reporte, el material generado (que está disponible en la página del curso de Ingeniería de Software) ya ha sido utilizado con gran éxito en el semestre 2008-1, recibiendo comentarios positivos por parte de profesoras y ayudantes de los 2 grupos en que se impartió la materia en dicho semestre, pues los manuales fueron aceptados de buena manera y con gran aprovechamiento del alumnado. En palabras de profesoras y ayudantes, se ha destacado la utilidad del material y se ha comentado que la disponibilidad y contenido de los tutoriales han servido para que los estudiantes elijan alguna(s) de las herramientas propuestas o que al menos sean fuertemente consideradas como opción, ampliando el panorama de alternativas de herramientas de apoyo para desarrollar software. Adicionalmente, los estudiantes pueden encontrar una fuente futura de referencias para la instalación y uso de las herramientas propuestas.

Al momento de realizar este material y, por supuesto, al cursar la materia, me he dado cuenta de que reunir e instalar las herramientas necesarias para llevar a buen término el curso, tomando en cuenta la premura del semestre y que el alumno lleva varias asignaturas por semestre, es una tarea poco sencilla y aún menos rápida. Por esta razón resulta atinado proporcionar este tipo de contribución a los alumnos.

Aprovechando que hablamos de estas herramientas, fue importante escribir acerca de una clasifi-

cación de licencias del software, de acuerdo a su distribución, buscando aclarar un poco algunos términos comúnmente mal empleados, sobre todo considerando que las herramientas de las que se presentan los tutoriales son de distribución gratuita, pero como ya se mencionó, no necesariamente todas *libres* o todas *open source*. Los alumnos ahora pueden darse una mejor idea de estos conceptos con el pequeño artículo expuesto en el sitio del curso, desarrollado como parte de este trabajo.

Para mí ha sido un placer colaborar con este material y cooperar de esta manera a fomentar un mejor aprovechamiento del curso de Ingeniería de Software, dirigido al alumnado y, por consecuencia, también para los profesores y ayudantes. Si a alguno de ellos le resulta más fácil la estancia en el curso gracias a estos tutoriales, como ya se ha empezado a observar, la misión se estará cumpliendo.

También puedo decir que la experiencia al desarrollar este trabajo ha sido sumamente provechosa en lo personal, al aprender más sobre cada una de las herramientas presentadas, tanto en historia como en instalación, uso e integración entre sí; también el aprendizaje en cuanto a los tipos de licencia del software y el aclarar muchas ideas equivocadas y arraigadas. Considero que los conocimientos adquiridos en este material pueden ser de gran ayuda en un futuro inmediato al incursionar en el mundo laboral, pues pienso que mi dominio en los procedimientos que guían al curso y en general, en el desarrollo de proyectos de software, es mayor que al iniciar esta actividad y, por supuesto, en el manejo de las herramientas de apoyo en el desarrollo de software.

Ojalá que la contribución plasmada en el presente trabajo siga siendo de gran utilidad y fácil lectura para los alumnos, para contribuir a que los objetivos de formación de los educandos en esta asignatura se lleven a cabo de la mejor manera.

Bibliografía

[1] Ingeniería de Software Pragmática. Guadalupe Ibargüengoitia y Hanna Oktaba. Facultad de Ciencias, UNAM. 2006.

[2] Ingeniería de Software: Una Perspectiva Orientada a Objetos. Braude, Erick J. Ed. Alfaomega, 2003.

[3] La Ofensiva del Software Libre. Wayner, Peter. Ed. Granica, 2001.

[4] Software Libre para una Sociedad Libre. Stallman, Richard. Traducción por Jaron Rowan, Diego Sanz y Laura Trinidad. Ed. Traficantes de Sueños, 2004.

[5] Revista Software Guru.

[6] *www.wikipedia.org*: Wikipedia, la Enciclopedia Libre.

[7] *www.wikilearning.com*: comunidades de Wikis Libres para Aprender.

[8] *www.gnu.org*: sitio web del Proyecto GNU.

[9] *www.yoopeedoo.org*: Sitio web de UPEDO (Unified Process for EDUcation).

[10] *www.netbeans.org*: Sitio web oficial de NetBeans.

[11] *dev.mysql.com*: Sitio web para desarrolladores en MySQL.

[12] *argouml.tigris.org*: Sitio web oficial de ArgoUML.

[13] *ganttproject.sourceforge.net*: Sitio web oficial de GanttProject.

[14] *www.postgresql.com.mx*: Comunidad de usuarios y sitio web de PostgreSQL en México.