UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Aplicación de Gestión de un Corrector Automático de Ejercicios de Programación (CAP)

Proyecto Final de Carrera

Ingeniería Técnica en Informática de Gestión

Autor: Daniel Iniesta González

Director: Oscar Sapena Vercher

Septiembre 2013

Índice General

Capítulo 1
Introducción

En el primer capítulo de esta memoria se va a exponer el trabajo a realizar en este proyecto final de carrera. Para comenzar vamos a analizar el porqué de este proyecto, exponer la motivación que ha llevado a su realización y la descripción de la problemática a resolver.

1.1 Introducción

Los correctores automáticos de programas son herramientas que nos permiten corregir un problema de programación de forma automática, sin necesidad de intermediarios, solamente interactuando el usuario y el programa. Típicamente los correctores automáticos funcionan como una caja negra, comparando la salida de un programa con una serie de soluciones predeterminadas para el problema en concreto, determinando así si la solución propuesta por el alumno es correcta o no.

Estas herramientas están principalmente orientadas a actividades docentes, ya que

permiten al alumno saber de forma automática y sin necesidad de la ayuda de un profesor si su propuesta de solución a un problema es correcta o no. Aunque también podemos ver presentes este tipo de herramientas en competiciones de programación, donde se debe corregir de forma totalmente imparcial y con los mismos criterios a todos los participantes.

CAP [3] son las siglas del Corrector Automático de Programas desarrollado por profesores del departamento DSIC de la UPV. CAP cuenta con una serie de mejoras a la hora de corregir los programas de forma automática, ya que no solo indica si la solución propuesta por el alumno es correcta, si no que también le indica al alumno donde ha cometido errores y como debería solucionarlos. Proporciona al alumno una retroalimentación modulada, no actúa como un mero oráculo.

Esta herramienta no impone una disposición fija para la definición de los problemas, por lo que no obliga a los programas a desarrollar que tengan que recibir unos datos de entrada determinados y generar una salida con un determinado formato. Esto permite corregir cualquier tipo de programa, clase o método desarrollado en Java, aunque no se requiera ningún dato de entrada o no genere ninguna salida.

Sin embargo CAP no solo cuenta con mejoras de cara al alumnado, también pretende ser de utilidad al profesorado. Añade utilidades que permiten a los profesores agilizar el proceso de creación de ejercicios para los alumnos, además de permitirle hacer un seguimiento del progreso de cada alumno por separado, o la progresión del curso en general, para así saber en que puntos incidir más o que aspectos de la metodología debería cambiar.

La obtención de estadísticas sobre las tareas planteadas hace de CAP una herramienta muy valiosa, ya que supone una realimentación al profesorado inmediata que, en base a ellas, puede seleccionar con mejor criterio las tareas a proponer y mejorar los casos de prueba.

1.2 Descripción del problema

CAP ofrece al alumnado una herramienta educativa muy potente para que perfeccione y afiance sus conocimientos en el ámbito de la programación. Sin embargo el profesorado

debería ser capaz de introducir en la base de datos de problemas para los alumnos material docente de forma rápida y lo más sencillamente posible. La finalidad de la aplicación para la gestión del corrector automático de programas consiste en otorgar a los usuarios, en este caso profesorado de distintas asignaturas relacionadas con la programación, una forma sencilla de añadir material.

Las acciones que puede llevar a cabo el gestor del corrector automático de programas es la inserción de nuevos Temas a la base de datos. Estos temas contendrán una serie de ejercicios relacionados. Evidentemente se podrán realizar operaciones de gestión sobre los temas, como modificar alguno de sus atributos o eliminarlo. Estos ejercicios a su vez van a tener asignadas una serie de clases que le servirán al alumno como apoyo para la resolución del problema en cuestión. Estas clases que van a tener asignadas los ejercicios podrán ser añadidas por el profesorado. Podrán existir diferentes versiones de una misma clase. Como en el caso de los temas, las clases también se van a poder modificar además de poder eliminarlas. A parte de las clases asociadas a un determinado ejercicio también existe una serie de documentos que pueden ir asociados. La aplicación permitirá al usuario añadir o eliminar alguno de los documentos de un ejercicio determinado. El profesorado podrá comprobar que la creación de un ejercicio contiene las clases necesarias para su resolución abriendo todas las clases del ejercicio en un mismo directorio con la aplicación BlueJ [4]. Esto permite ver las relaciones correspondientes entre las clases de un mismo ejercicio, asimismo como realizar la resolución del problema en el entorno BlueJ para comprobar el correcto funcionamiento de las clases.

Los alumnos al usar CAP y realizar un determinado ejercicio recibirán una determinada puntuación, que se guardará en la base de datos. El sistema permite ver que alumnos han realizado un determinado ejercicio, indicando además que nota es la que han obtenido. Por otro lado también tenemos la opción contraria, de saber que alumnos han realizado determinado ejercicio y cual ha sido la nota obtenida.

En definitiva, se debe crear todo el material docente correspondiente para que el alumnado pueda realizar una serie de ejercicios diseñados por el profesorado. Estos ejercicios serán evaluados mediante CAP. Mientras que para el profesorado se debe facilitar la información correspondiente a las notas obtenidas por los alumnos en los ejercicios que realicen.

1.3 Objetivos del proyecto

El objetivo que se pretenden cumplir con este PFC es la creación de una aplicación para la gestión del corrector automático de ejercicios de programación (CAP). Esta aplicación se pretende que conste de una aplicación gráfica de usuario fácil y cómoda de uso, optando en todo momento por un diseño minimalista pero funcional. Los elementos los cuales va a hacer uso el gestor del corrector automático de ejercicios de programación y, que por lo tanto, sobre los que se va a guardar información son los siguientes:

- **Temas:** Los temas determinan que tipo de ejercicios nos vamos a encontrar en cada uno de ellos. Toda la información de los temas será almacenada para su gestión. Los ejercicios desarrollados por el profesorado siempre irán asociados a un tema determinado.

- **Clases:** Estos elementos proporcionan el código fuente necesario para la realización de los ejercicios propuestos al alumnado. Una misma clase podrá tener diferentes versiones, las cuales se asignarán a determinados ejercicios.

- **Documentos:** Son archivos que tienen información relacionada con el tema y el tipo de ejercicio al cual están asociados. Típicamente estos archivos tienen extensión PDF. Los documentos sirven como respaldo teórico a los alumnos, para que puedan revisar de forma rápida información relevante para la resolución del problema.

- **Ejercicios:** Necesitamos saber cuales son los ejercicios presentes en la base de datos, para así poder gestionarlos.

- **Alumnos:** Los alumnos presentes en la base de datos serán todos aquellos que estén matriculados ese año en alguno de los cursos que usen la herramienta CAP para la evaluación del curso académico. De los alumnos necesitaremos saber todos sus datos, tales como el nombre, el DNI, el mail de contacto, etc.

- **Notas:** Ya que el propósito principal de CAP es evaluar a los alumnos, se deben guardar todas las notas de los ejercicios que realicen.

1.4 Beneficios a obtener

Con este proyecto se pretende obtener dos grandes ventajas principales, ambas orientadas al profesorado:

- **Gestión de ejercicios de implementación:** Realizar una interfaz gráfica que ayude al profesorado a gestionar los ejercicios de forma rápida, sencilla e intuitiva. En la realización de esta IGU se tendrá que tener en cuenta la gestión de toda la información pertinente para cada ejercicio trabajando con una base de datos SQL. Entre otros datos que se tendrán que gestionar serán las clases disponibles, los temas a los que corresponden, etc.

Además, se añadirán algunas funcionalidades, tales como poder abrir un ejercicio en Bluej , lo que supone agrupar todas las clases necesarias para la realización y compilación del ejercicio en un mismo directorio y el lanzamiento del programa automáticamente. O también gestionar determinados documentos que pueden ser de utilidad, como anexos, apendices, guias, etc. en cualquier formato (tipicamente .pdf).

- **Estadísticas y notas:** Implementar una serie de módulos que permitan al profesorado ver estadísticas que le puedan ayudar a mejorar los ejercicios a realizar. Las estadísticas que le pueden ser más relevantes al profesorado pueden ser las notas que han sacado sus alumnos para un ejercicio determinado, y saber para cada alumno que notas ha sacado en los ejercicios que haya realizado.

Con esto pueden saber que tipo de ejercicios son los que más dificultades presentan a los alumnos, y cuales ya están completamente claros y no haría falta hacer más incapié en ellos. Además hace más fácil el seguimiento de un alumno en particular si fuese necesario, pudiendo tener en cuenta las fechas que realización de los ejercicios, la progresión de notas, la cantidad de intentos para cada ejercicio, etc.

Estos objetivos podrán repercutir en una mejora de la calidad de los ejercicios propuestos para los alumnos, por lo que el objetivo de CAP se verá complementado.

Capítulo 2
Marco teórico

Para empezar con la memoria vamos a hacer una pequeña introducción teórica de los aspectos utilizados para la creación del proyecto, sabiendo de antemano que se va a abordar el problema desde una visión orientada a objetos. Estos conocimientos teóricos que vamos a usar para el desarrollo del software son: Programación orientada a objetos, modelado conceptual, diseño conceptual y diseño de interfaces gráficas.

2.1 Programación orientada a objetos

La programación orientada a objetos [7] (POO) es un paradigma de programación relativamente nuevo, el cual encapsula toda la información en objetos. Estos objetos son entidades propias que tienen un determinado estado, comportamiento e identidad.

El estado de un objeto es la composición de datos asignados a ese determinado objeto. Este estado es definido por el valor de los atributos que contenga ese objeto en particular. El estado y el comportamiento de un objeto están estrechamente relacionados entre si, ya que

proporcionan información al conjunto del objeto.

Las principales características de la programación orientada a objetos son las siguientes:

Encapsulación: Todo objeto muestra determinada información de si mismo a otros objetos, especificando cómo estos otros podrán interactuar con él (mediante la especificación de métodos públicos). Todos los demás datos que tenga el objeto son privados, por lo que sólo él mismo podrá acceder a estos para realizar sus operaciones de manera interna.

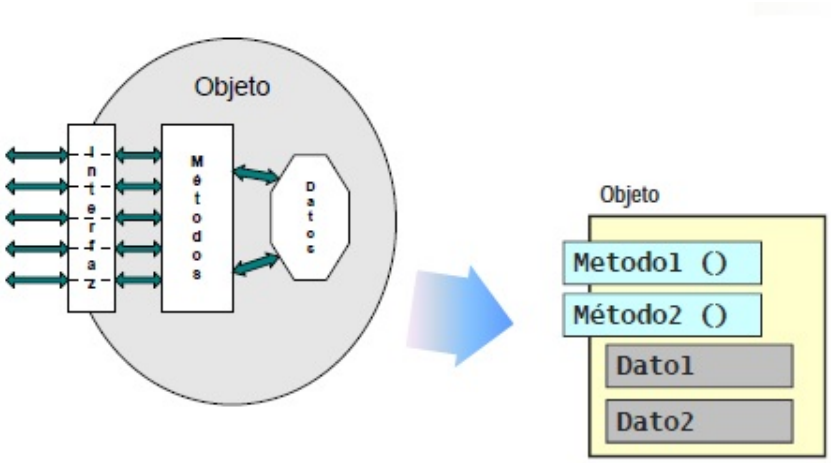

Figura 2.1 Definición gráfica de Objeto

Herencia: Este mecanismo permite la especificación de clases. Una clase heredada de otra contiene todos los atributos e información referente a la clase padre, además de la que pueda implementar por si misma.

Polimorfismo: El polimorfismo conjunto al mecanismo de herencia hace de la programación orientada a objetos uno de los paradigmas más potentes. Esto permite que el código de determinada función sea escrito una vez, ya que se podrá utilizar en otros objetos que requieran de esta función. Normalmente se ve referente a la definición de métodos en una clase padre, la cual proporciona ese método a todas las clases hijas que pueda tener.

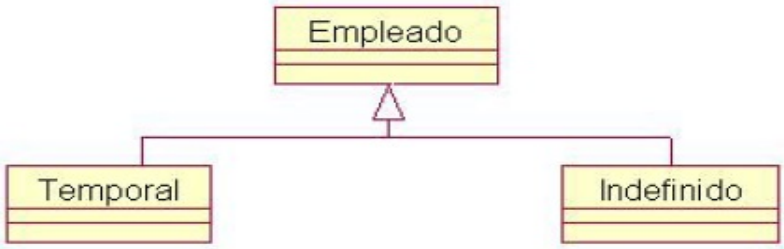

Figura 2.2 Diagrama jerarquía de herencia

2.2 Modelado y análisis conceptual

Para la construcción de una aplicación software se debe hacer uso de los distintos mecanismos y diagramas que se nos ofrecen. Esto facilita la comunicación entre todas las partes implicadas en la creación de dicho proyecto, tanto por parte de los clientes como de los desarrolladores. El uso de modelos proporciona ventajas tales como:

- Facilita la comunicación entre todas las partes implicadas.
- Agiliza el proceso cognitivo para la correcta comprensión de sistemas complejos.
- Se pretende que sea una solución general, la cual se pueda reutilizar si es necesario.
- En principio es menos ambigua que una descripción en lenguaje natural.
- Nos ayuda a la resolución de problemas similares a los ya tratados.
- Al tratarse de un estándar va a estar muy extendido.

Los modelos nos ayudan a especificar de forma detallada todos los aspectos necesarios para que correcta implementación de un proyecto. Estos modelos especifican que se debe conseguir, pero no cómo hay que hacerlo.

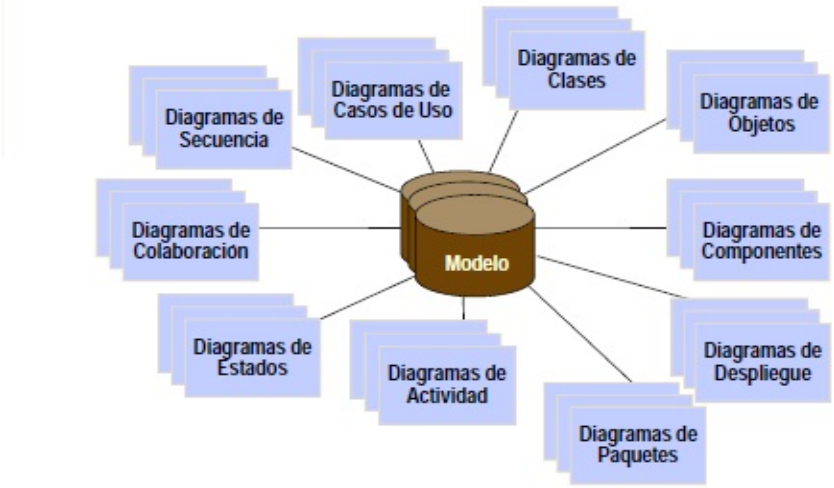

Figura 2.3 Modelado diagramas UML

2.2.1 Métodos de modelado orientado a objetos

Para el fin de realizar modelos en un paradigma de programación orientado a objetos tenemos el estándar UML [10] (*Unified Modeling Language*). En este estándar de modelado conceptual se presentan distintos tipos de diagramas que el desarrollador puede usar para la resolución de un proyecto software. Los distintos diagramas que podemos encontrar tipificados en UML son los siguientes:

Diagrama de casos de uso: Este diagrama determina como se debe comportar el sistema bajo el punto de vista del usuario. Un caso de uso es la descripción semiformal de un requisito. Los diagramas de casos de uso son fácilmente comprensibles por el usuario, por lo que podrá ayudar a establecer los requisitos funcionales que debería tener el sistema a desarrollar.

Diagrama de clases: Un diagrama de clases representa las clases que habrá en el sistema. Se indican todas las relaciones estructurales entre las distintas clases y la herencia si la hubiese. Este tipo de diagramas son los principales para el modelado y diseño orientado a objetos. Esto es así ya que en un mismo diagrama se incluye la definición de atributos de las clases, las relaciones estructurales existentes entre ellas

(asociación, agregación y composición) y de herencia. Además indica la multiplicidad presente entre las distintas clases que interactuan entre ellas.

2.3 Diseño conceptual

Al analizar el problema a resolver se debe decidir la estrategia a seguir para la resolución del proyecto. Entre otras cosas se debe determinar que tipo de arquitectura es la más eficiente, o conveniente, para la resolución del proyecto.

2.3.1 Arquitectura de la aplicación

La arquitectura de una aplicación debe ser proporcionada por el desarrollador de la misma, ya que debe estructurarla de forma que sea lo más eficiente posible. Se pretende que la arquitectura de un sistema pueda ser tal que la realización del trabajo sea lo más eficiente posible.

La arquitectura de un sistema se puede dividir en subsistemas, lo cual podemos organizarlo como una secuencia de capas horizontales (arquitectura multicapa), donde cada una de las capas se establecen con respecto a la capa inmediatamente inferior. Por lo que cada capa proporciona un punto base para la implementación de las capas superiores.

2.3.2 Arquitectura en tres capas

Este tipo de arquitectura es de las más utilizadas en el desarrollo de proyectos software. Estas tres capas son las siguientes:

Primera capa. Capa de presentación: Es donde se ubican los componentes gráficos de la aplicación. Determina que es lo que verá el usuario, y establecerá la forma en la que interactua con él.

Segunda capa. Capa de Negocio: En esta capa se ubican todos los componentes software que sean necesarios para el manejo interno de la aplicación a desarrollar. Determina el comportamiento que va a tener la aplicación.

Tercera capa. Capa de Persistencia: Aquí tendremos todos los componentes software que permiten a la aplicación mantener de forma permanente todos los datos necesarios.

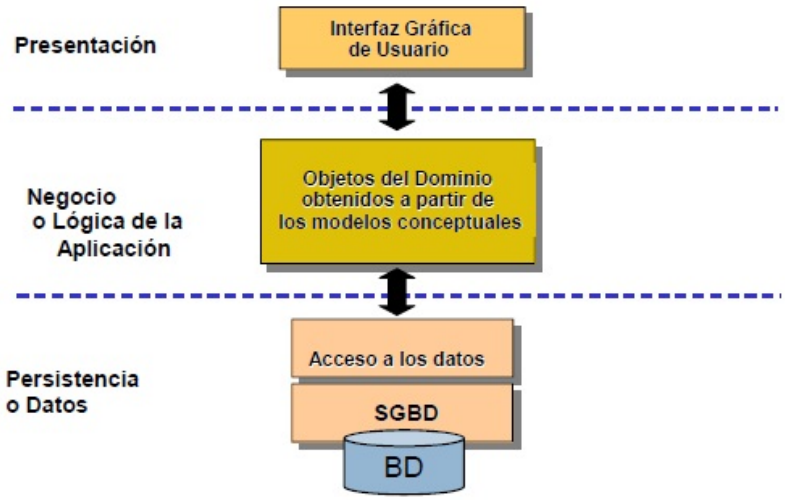

Figura 2.4 Arquitectura multicapa

2.4 Interfaz gráfica de usuario

La interfaz gráfica de usuario (IGU) es uno de los puntos más importantes de una aplicación visual, ya que es la que va a permitir al usuario interactuar con el sistema. Un interfaz consta de distintos formularios con los que el usuario puede ir modificando o consultando información relacionada con el sistema. Siempre debe haber un formulario principal desde el cual se pueda acceder a los demás formularios.

El objetivo de una correcta interfaz gráfica es desarrollar o mejorar la seguridad, la utilidad, la efectividad, la eficiencia y la usabilidad del sistema. Para la realización de una interfaz gráfica de usuario correcta de puede seguir una base de recomendaciones o estándares que se suponen recomendados para tal fin. A continuación se presentan las 'reglas de oro' para el diseño de interfaces gráficas de usuario.

- Consistencia, en términos de sintaxis, terminología, acciones y distribución de información.

- Métodos abreviados, proporcionando facilidades de interacción al usuario experto.

- Información de feedback, proporcionando respuestas del sistema (de modesta a sustancial).

- Diseño organizado y agrupado, con las secuencias de acciones claramente estructuradas desde principio a fin (cronología intuitiva).

- Prevención y gestión sencilla de errores, evitando que el usuario pueda cometer errores graves (y en caso de cometerlos, detectarlos).

- Permitir deshacer acciones, reduciendo la ansiedad del usuario para familiarizarse con nuevas opciones.

- Control del usuario, para que el usuario sea el agente ejecutor de acciones, y no un simple receptor.

- Minimizar la memorización a corto plazo, con diseño sencillos e intuitivos (realidad y metáforas).

Como podemos observar en la siguiente figura, para el diseño de la interfaz gráfica deberíamos realizar los pasos marcados en la misma.

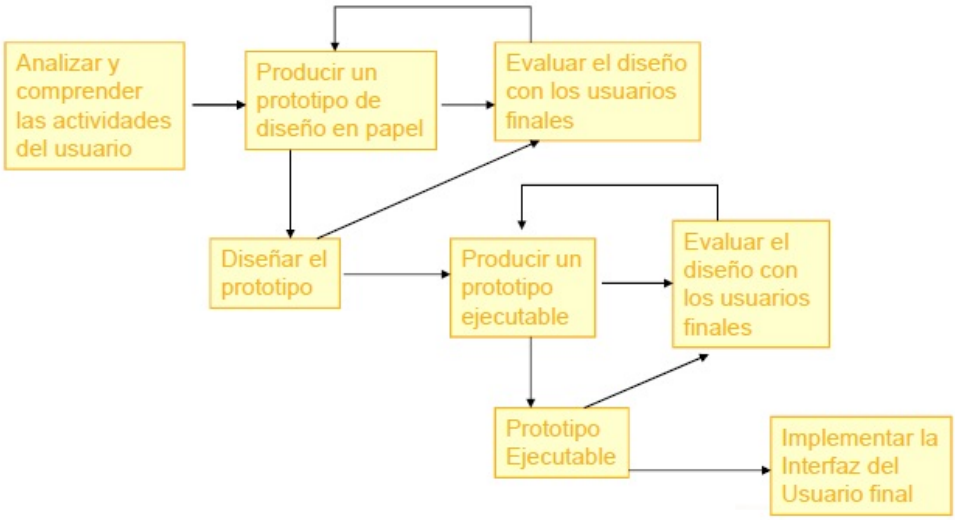

Figura 2.6 Pasos para la creación de una interfaz gráfica

Capítulo 3

Especificación de requisitos software

En este capítulo se va a presentar la especificación de requisitos (ERS) de la aplicación para la gestión del corrector automático de ejercicios de programación (CAP) que se ha realizado para este proyecto final de carrera.

3.1 Introducción

En el proceso de Ingeniería de software el primer paso que debemos realizar es la especificación de requisitos. Este análisis nos permite obtener la descripción del sistema tal y como se quiere ver desarrollada.

Estos requisitos son dados al desarrollador por el usuario que solicita la aplicación. Típicamente el usuario proporciona los requisitos en lenguaje natural, de forma informal, de tal forma que sea de comprensión sencilla para todos los involucrados en el proyecto. Esto tiene como objetivo que los desarrolladores de la aplicación sepan con exactitud que deben hacer.

Los objetivos principales de esta primera fase en el proceso de Ingeniería de software son los siguientes:

- El desarrollador debe tener claro todos los aspectos de la aplicación a desarrollar desde el primer momento, con la funcionalidad concreta que desee el usuario.

- El usuario debe saber con precisión qué funcionalidades quiere que tenga la aplicación.

Es imprescindible que en esta etapa todos los involucrados en el desarrollo de la aplicación se entiendan a la perfección. Esta fase es crítica en el desarrollo de software, ya que unos requisitos funcionales mal establecidos o mal interpretados pueden conllevar un gran inconveniente a la hora de generar el proyecto. Estos inconvenientes se ven reflejados en las entregas de prototipos, modificación completa o parcial de parte de la funcionalidad, etc.

Se puede definir un requisito como “Una condición o capacidad necesaria para que un usuario resuelva un problema o alcance un objetivo” [1].

Existen varios tipos de requisitos:

- **Requisitos de Usuario:** Declaraciones de los servicios que se espera que el sistema proporcione y las restricciones bajo las que debería funcionar.

- **Requisitos de Sistema:** Establece con detalle las funciones, servicios y restricciones operativas del sistema. Estos requisitos deben ser precisos y exactos.

- **Requisitos funcionales:** Declaraciones de los servicios que el sistema debe proveer, comportamiento bajo determinadas situaciones y como reaccionará en todos los

aspectos.

- **Requisitos no funcionales:** Restricciones que deben tener los servicios o funciones. Por ejemplo el uso de estándares, el proceso de desarrollo, etc.

Una buena especificación de requisitos software debe definir todos los requisitos correctamente, no describir ningún detalle de diseño o implementación (define que hacer, no como) y no imponer restricciones de software. Para lograr este objetivo se pueden seguir las directrices que nos marca el estándar IEEE 830-1998 [2]; No es necesario que se siga estrictamente la organización y el formato dados en el estándar, siempre y cuando se refleje en el documento toda la información que se requiere.

En resumen, una buena especificación de requisitos ayuda a los clientes a describir con total exactitud y claridad que es lo que se desea obtener mediante el desarrollo del software, y ayudar a los desarrolladores a entender qué es lo que quiere exactamente el usuario.

3.2 Propósito

En este capítulo se pretende presentar los requerimientos que debe cumplir la aplicación para la gestión de un corrector automático de programación (CAP). Con este estudio de los requisitos del problema que se nos plantea se pretende obtener una aplicación correcta intentando que los errores sean mínimos, seguir una metodología para la creación de software de calidad y tener claros desde el primer momento todas las características que debería cumplir la aplicación. Así como el tipo de usuario que hará uso de la aplicación.

3.3 Descripción general

A continuación pasamos a describir de forma muy informal todos los aspectos que debería contener la aplicación a desarrollar. Esto es definir cuales van a ser las funciones que va a realizar nuestra aplicación.

3.3.1 Funciones de la aplicación

En este punto se establecen las funciones con las que va a contar nuestro proyecto.

- Operaciones sobre los Temas:
  - Añadir un nuevo tema a la base de datos.
  - Modificar los datos de un tema.
  - Listar los ejercicios de ese tema.
  - Eliminar un tema de la base de datos.
- Operaciones sobre las Clases:
  - Añadir una nueva clase a la base de datos.
  - Modificar el código de una clase.
  - Eliminar una clase de la base de datos.
- Operaciones sobre los Ejercicios:
  - Añadir documentos a un ejercicio.
  - Eliminar documentos de un ejercicio.
  - Abrir las clases correspondientes con BlueJ
- Operaciones de consulta:
  - Listado de las notas de un alumno para un ejercicio.
  - Listado de los ejercicios realizados por un alumno.

3.3.2 Características del usuario

Típicamente el usuario de la aplicación debería ser un profesional de la enseñanza en el

campo de la programación, o asignaturas relacionadas con la informática. Por lo que se da por supuesto que dicho usuario tendrá nociones de sobra del manejo de aplicaciones basadas en ventanas gráficas.

3.4 Requisitos específicos

En este punto se presentan las operaciones que va a realizar la aplicación con más detalle. Ya que se ha elegido un enfoque orientado a objetos en el desarrollo del software, utilizaremos una organización por clases/objetos para representar todos los requisitos del sistema.

La notación usada a continuación corresponde a la declaración del tipo de dato correspondiente en cada una de las clases:

- Para la representación de las cadenas de caracteres utilizaremos el tipo String.
- La representación de cadenas numéricas se hará el uso de los tipos Int, Double, etc. según sea necesario por la especificación.
- Para los atributos de “verdadero/falso” se hará el uso del tipo Boolean.
- El tipo fecha se expresa de la forma Date.

3.4.1 Clase 'Clase'

Las Clases son una de las entidades principales del sistema, ya que son uno de los elementos que vamos a gestionar.

Atributos

Descripción	Tipo	Formato	Ejemplo
IdExer	Numérico	Entero	10011
Name	Texto	Cadena	ArrayCola
Versión	Numérico	Entero	1
Visible	V/F	Booleano	True
Code	Texto	Cadena	public class...

Operaciones

Las operaciones que se realizan con la clase Clase son las típicas operaciones de gestión. Esto es: la creación de nuevas clases, la modificación de una clase y eliminar una clase.

Crear una nueva clase	
Entradas:	Datos de la clase (Nombre, Versión y Código).
Proceso:	Se verifica que el nombre y la versión de la clase a crear no se encuentran ya en la base de datos. En caso de que alguna de las comprobaciones no sean válidas se informa al usuario y se impide continuar.
Salidas:	Se crea una nueva Clase en el sistema o un mensaje de error.

Modificar una clase	
Entradas:	Datos de la clase (Nombre, Versión y Código).
Proceso:	Se verifica que el nombre y la versión de la clase a modificar se encuentran en la base de datos. En caso de que alguna de las comprobaciones no sean válidas se informa al usuario y se impide continuar.
Salidas:	Se actualiza la Clase en el sistema o un mensaje de error.

Eliminar una clase	
Entradas:	Datos de la clase (Nombre y Versión).
Proceso:	Se verifica que el nombre y la versión de la clase a modificar se encuentran en la base de datos. En caso de que alguna de las comprobaciones no sean válidas se informa al usuario y se impide continuar.
Salidas:	Se elimina la Clase en el sistema o un mensaje de error.

3.4.2 Clase 'Documento'

Un documento es un archivo de texto (típicamente un archivo .PDF) que va asociado a un determinado ejercicio.

Atributos

Descripción	Tipo	Formato	Ejemplo
Id	Numérico	Entero	1
nameC	Texto	Cadena	Tema 1 de EDA
nameV	Texto	Cadena	Tema 1 de EDA
nameE	Texto	Cadena	Topic 1 of EDA
Extension	Texto	Cadena	PDF
Content	Datos	Byte[]	'BLOB'

Operaciones

Las operaciones que se realizaran con los Documentos son los de asignación de documentos a un ejercicio, y eliminar un documento de un ejercicio.

Asignar un nuevo documento.	
Entradas:	Datos del documento (Id, NombreC, NombreV, NombreE, Extensión y Contenido) y el ejercicio a añadirlo (Id).
Proceso:	Se verifica el Id y los nombres del documento a asignar no se encuentren ya en la base de datos. Se comprueba también que el ejercicio a insertar existe en la base de datos. En caso de que alguna de las comprobaciones no sean válidas se informa al usuario y se impide continuar.
Salidas:	Se añade el nuevo documento al ejercicio en el sistema o un mensaje de error.

Eliminar un documento.	
Entradas:	Datos del documento (Id) y el ejercicio donde eliminarlo (Id).
Proceso:	Se verifica el Id del documento y el del ejercicio. En caso de que alguna de las comprobaciones no sean válidas se informa al usuario y se impide continuar.
Salidas:	Se elimina el documento del ejercicio en el sistema o un mensaje de error.

3.4.3 Clase 'Ejercicio'

Esta es una clase abstracta que estructura la información presente en los ejercicios que existen en el corrector de programas automático (CAP).

Atributos

Descripción	Tipo	Formato	Ejemplo
Id	Numérico	Entero	10011
Type	Numérico	Entero	1

Title	Texto	Cadena	Invertir cola
Statement	Texto	Cadena	“Ampliar la ...”
CurrentDate	Fecha	Fecha	28/08/13
VisibleFrom	Fecha	Fecha	01/09/13
VisibleTo	Fecha	Fecha	30/09/13
SolutionVisibleFrom	Fecha	Fecha	01/10/13
SolutionVisibleTo	Fecha	Fecha	07/10/13
Topic	Tema	Tema	Diseño EDA

3.4.4 Clase 'Ejercicio de implementación'

Es la especialización de la clase Ejercicio.

Atributos

Los atributos marcados con * son aquellos heredados de la clase padre.

Descripción	Tipo	Formato	Ejemplo
Id *	Numérico	Entero	10011
Type *	Numérico	Entero	1
Title *	Texto	Cadena	Invertir cola
Statement *	Texto	Cadena	“Ampliar la ...”
CurrentDate *	Fecha	Fecha	28/08/13
VisibleFrom *	Fecha	Fecha	01/09/13
VisibleTo *	Fecha	Fecha	30/09/13
SolutionVisibleFrom *	Fecha	Fecha	01/10/13
SolutionVisibleTo *	Fecha	Fecha	07/10/13
Topic *	Tema	Tema	Diseño EDA
StartingCode	Texto	Cadena	“Public class Array...”
Solution	Texto	Cadena	“Public class Array...”
InsertInClass	Clase	Clase	ArrayCola
InsertInVersion	Numérico	Entero	1
InsertInLine	Numérico	Entero	5
Test	Texto	Cadena	“mark.setMark(10);...”

MaxInstructions	Numérico	Entero	100000
WatchVariables	V/F	Booleano	False
StyleTest	Texto	Cadena	“mark.setMark(10);...”
StyleValue	Numérico	Doble	01/03/00
Clases	Clase	Lista	Cola, ColaExt, ...

Operaciones

Las operaciones que se realizaran con los Ejercicios será listarlos, para así poder realizar distintas operaciones sobre ellos, como abrirlos con Bluej o gestionar sus Documentos.

Listar Ejercicios	
Entradas:	Datos del tema al que pertenecen (Id).
Proceso:	Se verifica la Id del tema del que se quieren listar los ejercicios. En caso de que la comprobación no sea válida se informa al usuario y se impide continuar.
Salidas:	Se devuelve una lista de ejercicios o un mensaje de error.

3.4.5 Clase 'Notas'

Esta clase es la que contiene las notas de los alumnos un ejercicio que hayan resuelto.

Atributos

Descripción	Tipo	Formato	Ejemplo
IdStudent	Numérico	Entero	10
Date	Fecha	Fecha	15/09/13
idExercise	Texto	Entero	10011
Event	Numérico	Entero	1
Mark	Numérico	Doble	01/05/00
IdMessage	Numérico	Entero	1

Operaciones

Las notas las vamos a utilizar para realizar listas, dependiendo de si queremos saber que notas han obtenido los alumnos en un determinado ejercicio, o que notas ha sacado un alumno en los ejercicios que haya realizado.

Listar Notas	
Entradas:	Datos del ejercicio (id) y del alumno (Id o DNI)
Proceso:	Se verifica la Id del ejercicio del que se quieren listar las notas y el Id del alumno. En caso de que alguna de las comprobaciones no sea válida se informa al usuario y se impide continuar.
Salidas:	Se devuelve una lista de notas o un mensaje de error.

3.4.6 Clase 'Estudiante'

La clase Estudiante es la que encapsula todos los datos de los Alumnos pertenecientes a asignaturas que puedan hacer uso de CAP.

Atributos

Descripción	Tipo	Formato	Ejemplo
id	Numérico	Entero	10
DNI	Texto	Cadena	22573864k
Name	Texto	Cadena	Daniel Iniesta
Group	Texto	Cadena	2A1
Mail	Texto	Cadena	daingon1@ei.upv.es
login	Texto	Cadena	daingon1

Operaciones

Las operaciones a realizar con los alumnos van a ser listarlos para así saber las notas que han obtenido en los ejercicios.

Listar Alumnos	
Entradas:	No recibe ninguna entrada
Proceso:	Se pide al sistema que devuelva una lista con todos los alumnos presentes en la base de datos. En caso de que ocurra alguna complicación se informa al usuario y se impide continuar.
Salidas:	Se devuelve una lista de alumnos o un mensaje de error.

3.4.7 Clase 'Tema'

La clase Tema es la que determina los datos correspondientes de los temas presentes en la base de datos. Los temas nos indican de que tipo serán los ejercicios que existan en la base de datos.

Atributos

Descripción	Tipo	Formato	Ejemplo
id	Numérico	Entero	1
NameC	Texto	Cadena	Diseño EDA
NameV	Texto	Cadena	Diseny EDA
NameE	Texto	Cadena	EDA design

Operaciones

Las operaciones que se van a realizar con los objetos de la clase Tema son las típicas de gestión, esto es añadir un nuevo tema, modificar y/o eliminarlo.

Crear un nuevo tema	
Entradas:	Datos de la clase Tema (id, NombreC, NombreV y NombreE).
Proceso:	Se verifica que el nombre en cualquier de los 3 idiomas y el id no existe en ninguno de los temas ya existentes de la base de datos. En caso de que alguna de las comprobaciones no sean válidas se informa al usuario y se impide continuar.
Salidas:	Se crea un nuevo Tema en el sistema o un mensaje de error.

Modificar un tema	
Entradas:	Datos de la clase Tema (NombreC, NombreV y NombreE).
Proceso:	Se verifica que el nombre en cualquier de los 3 idiomas no existe en ninguno de los temas ya existentes de la base de datos. En caso de que alguna de las comprobaciones no sean válidas se informa al usuario y se impide continuar.
Salidas:	Se modifica el Tema correspondiente en el sistema o un mensaje de error.

Eliminar un tema	
Entradas:	Datos de la clase tema (id).
Proceso:	Se verifica que el id ya existentes de la base de datos. En caso de que la comprobación no sea válida se informa al usuario y se impide continuar.
Salidas:	Se elimina el Tema correspondiente en el sistema o un mensaje de error.

3.4.8 Clase Sistema

En la clases Sistema es donde se gestionan todas las clases anteriores. Un sistema contiene toda la información referente a los Temas, Estudiantes, Clases, Ejercicios de implementación y Documentos existentes en la base de datos. Sistema es el encargado de realizar todas las comprobaciones que se deben hacer para la gestión de las demás clases.

Atributos

Descripción	Tipo	Formato	Ejemplo
Topics	Tema	Lista	“Diseño EDA; EDA's lineales, ...”
students	Estudiante	Lista	“Iniesta González, Daniel; Sapena Vercher, Oscar; ...”
Exers	Ejercicio de implementación	Lista	“Invertir Cola; Cola con PI; ...”
docs	Documento	Lista	“Diseño EDA; Implementaciones lineales”

Operaciones

Las operaciones que se van a realizar en la clase sistema es la carga de la base de datos y la comprobación de todos los requisitos de las clases anteriores. En esta clase es donde vamos a comprobar si ya existen en la base de datos elementos iguales, o si por el contrario no están presentes. Estas comprobaciones conllevan el desarrollo de una u otra acción en la gestión de las demás clases presentadas anteriormente.

3.5 Conclusión

En este apartado se ha presentado la especificación de requisitos (ERS) de la aplicación para la gestión del corrector automático de ejercicios de programación (CAP). La especificación de requisitos permite tanto al usuario como al desarrollador saber que es lo que se va a desarrollar en la aplicación, y cuales van a ser las operaciones realizadas por cada una de ellas.

Al cliente le ayuda a describir lo que pretende que haga la aplicación a desarrollar, mientras que a los desarrolladores les permite entender con total precisión cuales son las peticiones del cliente. Además, unas buena ERS puede proporcionar distinto beneficios, a parte de los anteriormente mencionados. Estos beneficios pueden ser:

- La descripción completa de un ERS ayuda al cliente a decidir si todos los requisitos hasta el momento establecidos son los necesarios para la resolución completa y correcta de la aplicación software que solicita. El usuario puede considerar que faltan funcionalidades o por el contrario que existen algunas que se pueden obviar. De esta forma el usuario puede decidir si la especificación es correcta, o si se debe realizar alguna modificación.

- Los desarrolladores consiguen minimizar los consecuentes problemas que puedan surgir derivados en fases de desarrollo mas avanzadas. Unos requisitos minuciosamente establecidos tienen menos posibilidades de incurrir en fallos de diseño. Esto conlleva una reducción de esfuerzo de desarrollo considerable, ya que un error en las fases mas tempranas del desarrollo pueden ser fatales si no se detectan a tiempo.

- Con la especificación de requisitos se pueden estimar de una forma bastante acertada los costes y la planificación del proyecto. El coste temporal del proyecto es directamente proporcional al número de requisitos que se hayan que cumplir. Por otra parte el coste de la aplicación también depende en cierta forma de la cantidad de requisitos a cumplir.

- La fase de pruebas de verificación y validación del proyecto puede verse optimizada a partir del ERS. Al conocer todos los requisitos funcionales desde el principio se puede hacer una planificación de los procesos de verificación y validación de forma muy ajustada.

- Puede ser usado para planificar nuevas mejoras una vez ya se tenga el proyecto concluido. Estas mejoras deberían establecerse una vez ya se ha terminado todo el proceso software y tenemos como resultado un buen software.

Para concluir, la especificación de requisitos nos permite dar respuestas a las preguntas básicas que tanto el cliente como los desarrolladores se pueden plantear: ¿Qué debe hacer la aplicación?, ¿Qué puede hacer el usuario final con el?, ¿Se debería utilizar algún estándar en particular?, ¿Existe alguna restricción concreta a cumplir?, etc.

Capítulo 4

Desarrollo de la aplicación

En este capítulo se presentan los pasos que se han seguido para llevar a cabo el desarrollo de la aplicación que se pretende realizar para la presentación de un proyecto final de carrera. A continuación veremos que pasos se han ido siguiendo a lo largo de la realización del proyecto para conseguir llevarlo a cabo. Se expondrán todos los pasos por orden, empezando por la planificación, después el diseño de la aplicación, siguiendo por la la codificación y terminando por las pruebas realizadas al sistema para comprobar su correcto funcionamiento.

4.1 Planificación

Ya que estamos siguiendo el desarrollo normal para la creación de una aplicación software [8], las fases de desarrollo coinciden con las etapas de desarrollo software. Las fases de desarrollo o evolución de un software son generalmente las siguientes:

Fase de Especificación de requisitos: En este momento es cuando se realiza un estudio conjunto por parte del cliente y el desarrollador para establecer cuales deberían ser las funcionalidades del software. Para esto es indispensable que ambas partes de junten para así poder llegar a un entendimiento. Se deben especificar tantos los requisitos de usuario como los de interfaz. Duración: 4 días.

Fase de Modelado conceptual: Es en esta fase cuando se construye el modelo de objetos. Se deben establecer todas las clases y atributos de las mismas, así como la especificación de métodos. Duración: 10 días.

Fase de Diseño: En la fase de diseño se establece el tipo de arquitectura a usar en el desarrollo de la aplicación. En este caso de opta por una arquitectura estructurada en tres capas: Capa de Negocio, Capa de Presentación y Capa de Persistencia. Ya que el modelo relacional de la base de datos la proporciona el director del proyecto tan solo tenemos que ajustarnos a la especificación ya establecida. Duración: 10 días.

Fase de Codificación: En la fase de codificación es donde se implementa la funcionalidad de la aplicación. En este caso se utiliza Java como lenguaje de Programación Orientado a Objetos (POO) [7], usando el IDE Netbeans como entorno donde desarrollar la aplicación. Duración: 50 días.

Fase de Pruebas: Esta fase es la que comprende todas las pruebas del sistema y de aceptación por parte del cliente. Duración: 15 días.

Redacción de la memoria: Cuando ya se han realizado todas las fases anteriores podemos proceder a la realización de esta memoria, donde se pretende exponer de forma concreta todas los pasos seguidos para la realización del proyecto final de carrera. Duración: 20 días.

Para tener algo más clara la distribución del tiempo empleado para cada una de las fases que han conllevado la realización del proyecto, se ha realizado un diagrama de Gantt donde se muestra la planificación. Todas las fases del proyecto se encuentran parcialmente solapadas. Esto es así ya que podemos ir avanzando en las distintas fases aunque no hayamos completado totalmente la/s anterior/es. En este caso no existe más de una fase solapada a la vez. Sin embargo, podría haber sido así, ya que la redacción de la memoria se podría haber realizado en paralelo a mediados de la fase de codificación.

Las distintas fases del proyecto se detallan a continuación, exponiéndolas con más detalle.

4.2 Diseño de la aplicación

El primer paso para la realización del proyecto ha consistido en concretar los requisitos necesarios para el desarrollo del proyecto. Para este fin se establecieron una serie de entrevistas con el director del proyecto, donde se fueron concretando todos los requisitos. Cuando ya finalizó la fase de especificación de requisitos , se continuó con la fase de diseño del sistema, basando el modelo de la aplicación concorde a estos requisitos.

4.2.1 Análisis y especificación de los requisitos

Como ya se ha visto en el capítulo 3, la especificación de requisitos nos permite obtener una descripción completa de la lógica del programa a realizar. Para este fin se realizaron dos entrevistas, en las cuales se establecieron los requisitos que debería cumplir el sistema.

4.2.2 Análisis, diseño orientado a objetos

Cuando ya se tienen claros los requisitos a cumplir para el proyecto, podemos empezar a plantearnos que arquitectura es la más adecuada para desarrollarlo. Para este fin podemos realizar una serie de diagramas UML que nos ayuden a estructurar la aplicación. El principal diagrama que deberíamos tener en cuenta es el diagrama de clases. Este diagrama determina todas las relaciones internas que existen entre las diferentes clases presentes en el sistema. Esto proporciona mucha información al desarrollador de la aplicación.

Figura 4.1 Diagrama de Gantt

4.2.3 Diagrama Entidad-Relación

El diagrama Entidad-Relación permite establecer la estructura interna de la base de datos. a En nuestro caso solo nos hemos tenido que amoldar a la estructura ya establecida por el director del problema, ya que se requiere del uso de una base de datos ya creada e implementada. Dicha base de datos contiene una serie de tablas que se pueden aplicar para la gestión de distintos cursos/asignaturas sin necesidad de cambiarlas.

4.2.4 Arquitectura de la aplicación

La arquitectura seguida para el desarrollo de la aplicación ha sido una arquitectura basada en tres capas. He decidido que esto fuese así ya que es el tipo de arquitectura más usado, además que permite tener los tres aspectos primordiales de una aplicación software bien distinguidos entre si.

Las capas son las siguientes:

Capa de Presentación: En esta capa es donde se ubican todos los formularios de la aplicación. Uno de estos formularios debe ser el principal, el responsable de la comunicación con la capa inmediatamente inferior. Estos formularios son lo que se le mostrarán al usuario para que interactúe con el sistema.

Capa de Negocio: Aquí es donde se encuentran todas las clases que determinan la lógica del programa. Tan solo encontramos archivos .java en esta capa. Uno/a de estos/as archivos/clases debe ser el/la principal, el/la responsable de la comunicación con la capa inmediatamente superior.

Capa de Persistencia: Se compone de una base de datos gestionada en un servido MySQL. Esto viene impuesto en la especificación del proyecto, pero de todos modos sería una de las mejores opciones a contemplar, ya que se trata del tipo de base de datos mas extendido.

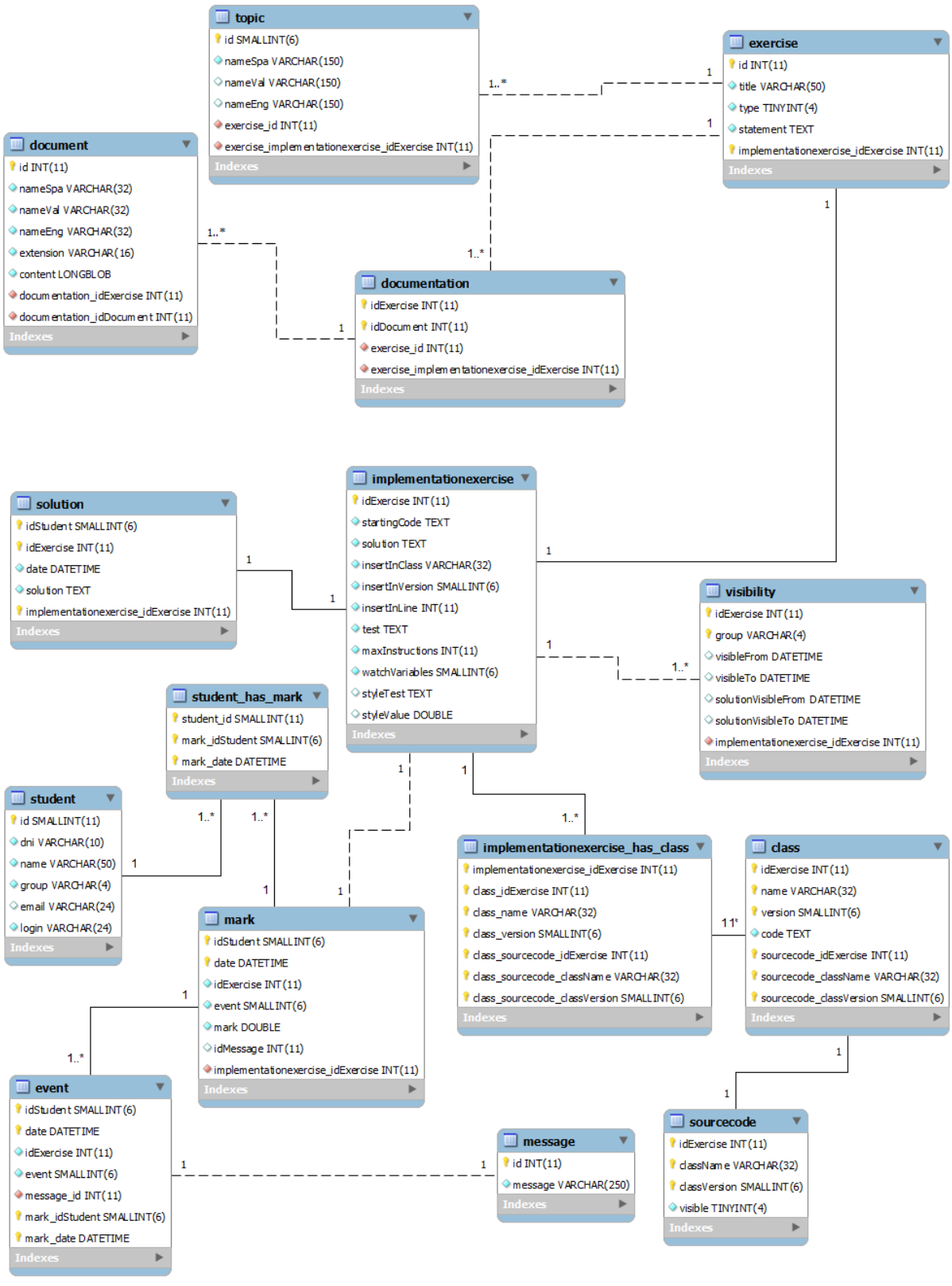

Figura 4.2 Diagrama Entidad-Relación

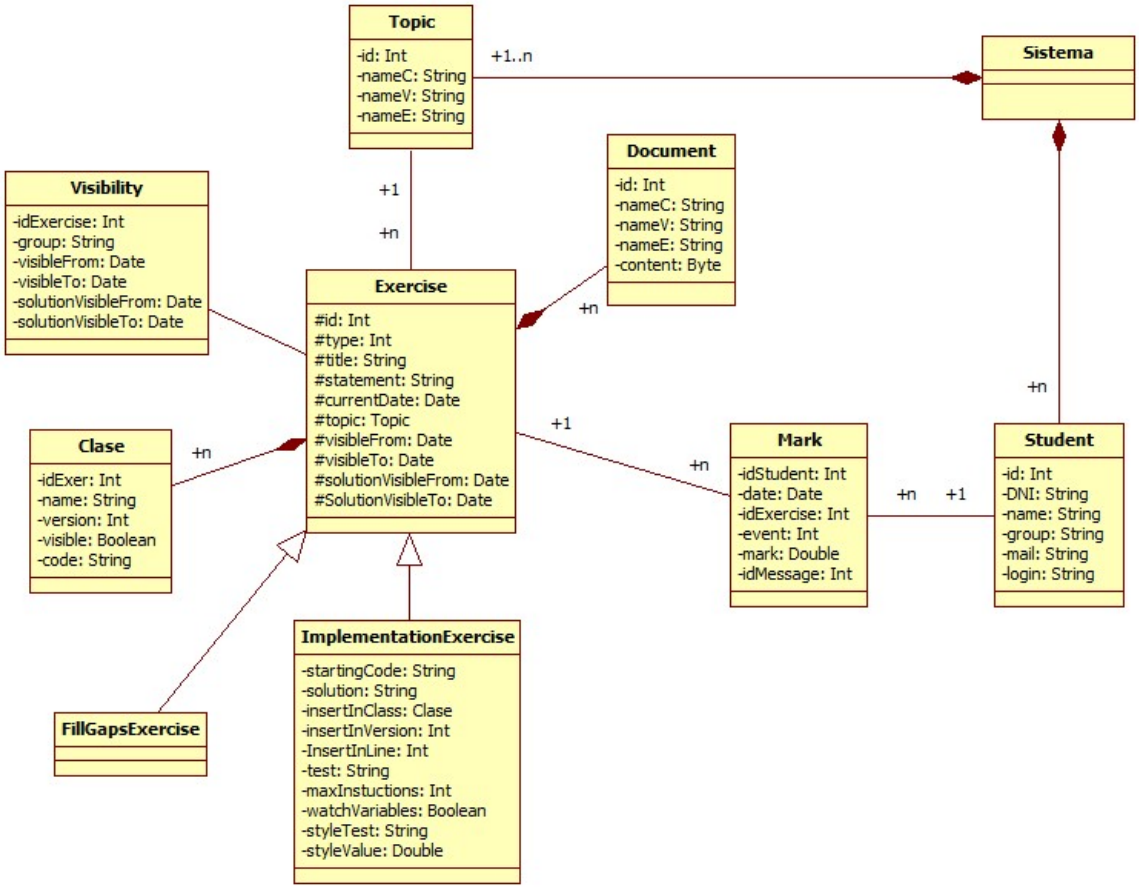

Figura 4.3 Diagrama de clases

4.3 Codificación

A continuación se debe proceder a la implementación de la lógica de software, dotandolo con las funcionalidades especificadas en el análisis de requisitos. Para realizar la codificación se decidió hacer uso de Java.

```
1   package Negocio;
2
3   import java.util.ArrayList;
4   import Tools.*;
5   import Persistencia.*;
6   import javax.swing.JLabel;
7   import javax.swing.JOptionPane;
8   import javax.swing.JTextArea;
9
10  public class Sistema {
11
12      private Lang l;
13      private DataBase db;
14      private KeepAliveThread aliveThread;
15      private ArrayList<Topic> topics;
16      private ArrayList<Student> students;
17      private ArrayList<Clase> clases;
18      private ArrayList<Implementationexercise> Exers;
19      private ArrayList<Document> docs;
20
21
22      public Sistema(){
23          try{
24          System.setSecurityManager(null);
25
26          this.l = new Lang("C");
27          this.db = new DataBase("eda");
28          aliveThread = new KeepAliveThread(db);
29          this.topics = db.getTopics(l);
30          this.students = db.getStudents(l);
31          this.clases = db.getClases(l);
32          this.Exers = db.getImplementationExercises(l);
33
34
36          }catch(Exception e){
37              ErrorCarga(e.getMessage(), l);
38          }
39
40      }
```

Figura 4.4 Ejemplo Código

4.4 Pruebas

Las pruebas del software es el último paso a realizar antes de la entrega del producto. Estas pruebas deben poner a prueba la calidad del software desarrollado. Además suponen una última verificación de la especificación, el diseño y la codificación.

El software debe pasar una serie de pruebas pensadas para que el sistema falle. Si se superan todas estas pruebas, podemos decir que se trata de un software de calidad. A parte de comprobar que la funcionalidad del software es la correcta, también debemos probar que si intentamos introducir datos no validos en el sistema no nos lo permite (Gestión de errores).

Para comprobar la funcionalidad de la aplicación se han realizado numerosas pruebas, como podrían ser:

- Insertar numerosos registros nuevos en todos los objetos que se gestionan en el sistema. Esto es: en temas, clases y documentos.

- Realizar la inserción de registros tanto válidos como incorrectos,ya sea por datos inválidos o datos ya existentes en la base de datos. Se crearon determinados objetos sabiendo que ya existían.

- Borrar registros en el sistema. Con esto se comprueba si se realiza correctamente la opción de borrado de un objeto.

- Modificar datos ya existentes en la base de datos y comprobar que la modificación se lleva a cabo correctamente.

Si todas estas pruebas son positivas, esto quiere decir que la funcionalidad del software es correcta. Con lo que ya se podría entregar el proyecto.

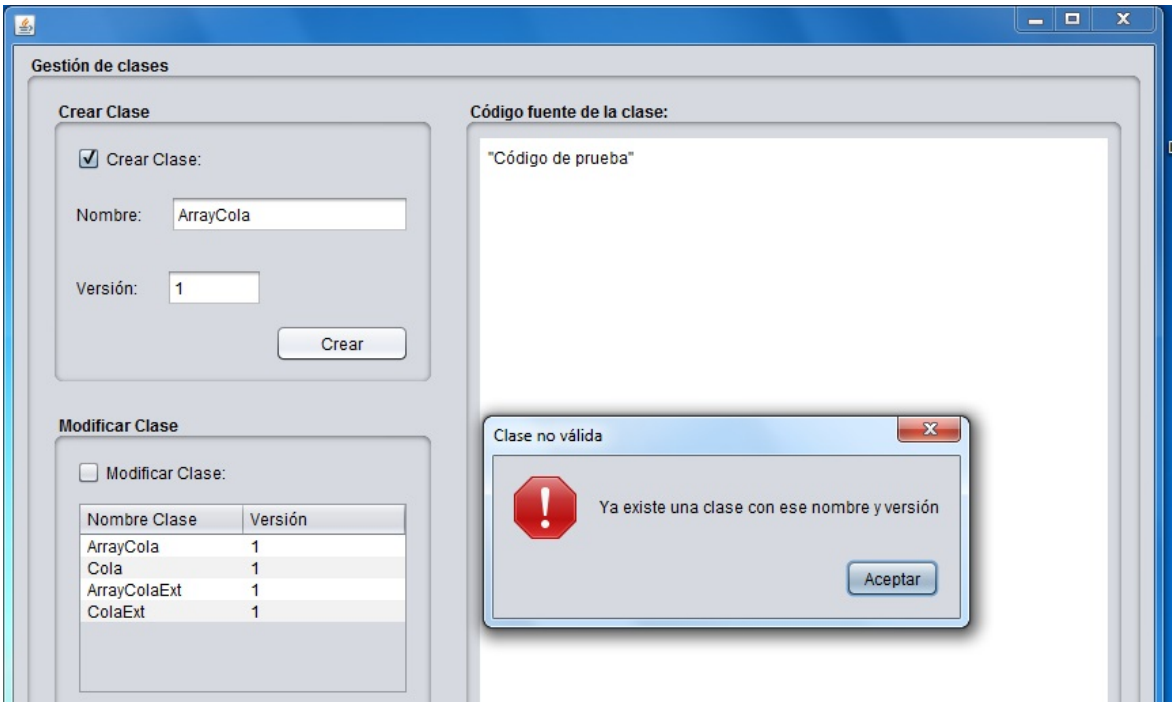

Figura 4.5 Pruebas

4.5 Conclusión

A lo largo de todo este capítulo se han ido presentando todos los pasos a seguir para el desarrollo de un sistema de información. Ya que se ha seguido un método de ingeniería software para la creación de dicho sistema, estos pasos coinciden con los descritos en dicho método. Primero se han establecido los requisitos funcionales, seguido del modulado conceptual, pasando por el diseño y la codificación, terminando con las pruebas del sistema.

En el desarrollo de una aplicación software no basta con 'saber programar', hace falta un enfoque puramente ingenieril para así poder desarrollar software de calidad. El desarrollador de software debe dominar todas las fases, desde el análisis hasta la implementación, pasando por el diseño.

Como todo ingeniero debe tener capacidad de abstracción y síntesis, debe discernir entre qué aspectos son válidos y cuales no lo son para la creación un sistema correcto, eficaz y eficiente. Esto lo consigue siguiendo un método determinado, y por supuesto, con la práctica y estudio.

Capítulo 5
Uso de la aplicación

A continuación se va a presentar una pequeña demostración de la aplicación software creada para gestionar la herramienta CAP. Al ser una aplicación destinada al uso docente vamos a suponer que ya se encuentra cargada en el entorno de desarrollo deseado por el profesor.

5.1 Iniciando la aplicación

Para iniciar la aplicación, al tratarse de un desarrollo pensado para que el profesorado pueda adaptarlo a otras aplicaciones ya desarrolladas, el inicio de la aplicación puede ser importándolo a la plataforma de desarrollo que prefiera. Si embargo puede que tan solo le interese ejecutarlo para la gestión de una determinada base de datos. En este último caso puede iniciarlo con el archivo ejecutable que se genera desde el IDE [5] (en este caso Netbeans).

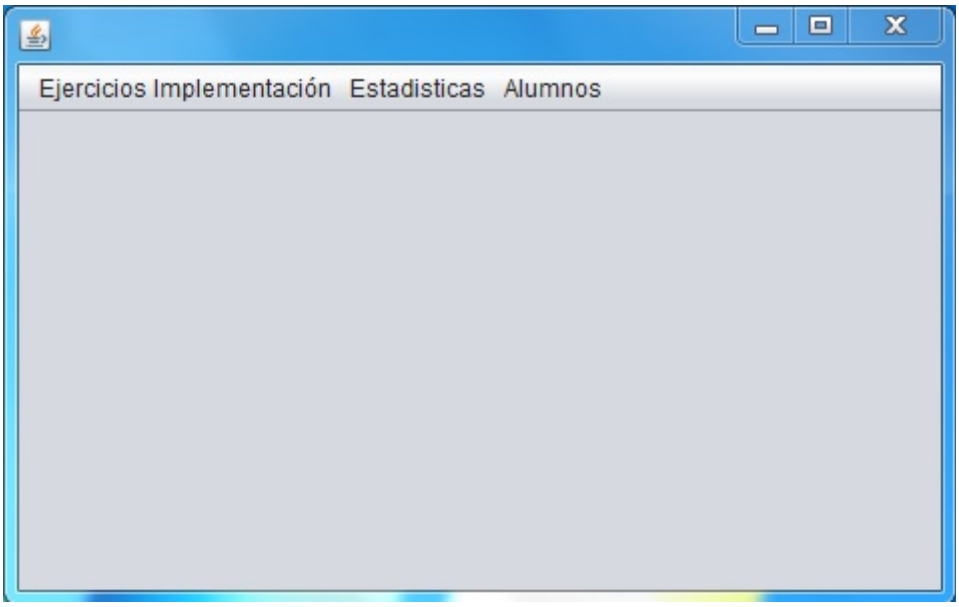

Figura 5.1 Inicio de la aplicación

A partir de la ventana principal de la aplicación podemos acceder a las demás ventanas de gestión. Accedemos a estas mediante los botones correspondientes en la barra de menú que encontramos en la parte superior.

5.2 Gestionar Temas

Cuando un usuario pretende gestionar los temas presentes en la base de datos, ya sea añadiendo uno nuevo, modificando o eliminando uno de los ya existentes debe dirigirse a **Ejercicios Implementación | Gestión de temas**. Una vez ya se encuentre en este formulario puede observar que están todas las opciones de gestión de los temas en una sola ventana.

5.2.1 Crear / modificar Tema

Desde esta ventana puede decidir que opción realizar. Si decide crear un nuevo Tema deberá rellenar todos los campos necesarios para la creación del mismo. En caso de que no rellenase todos los campos, o que alguno de los valores que le quiere asignar al nuevo tema ya existan en un tema presente en la base de datos, entonces se mostraría una ventana de error. Esta ventana de error notifica al usuario con un texto donde indica que es lo que ha causado el error tratado e impide continuar al usuario con la operación.

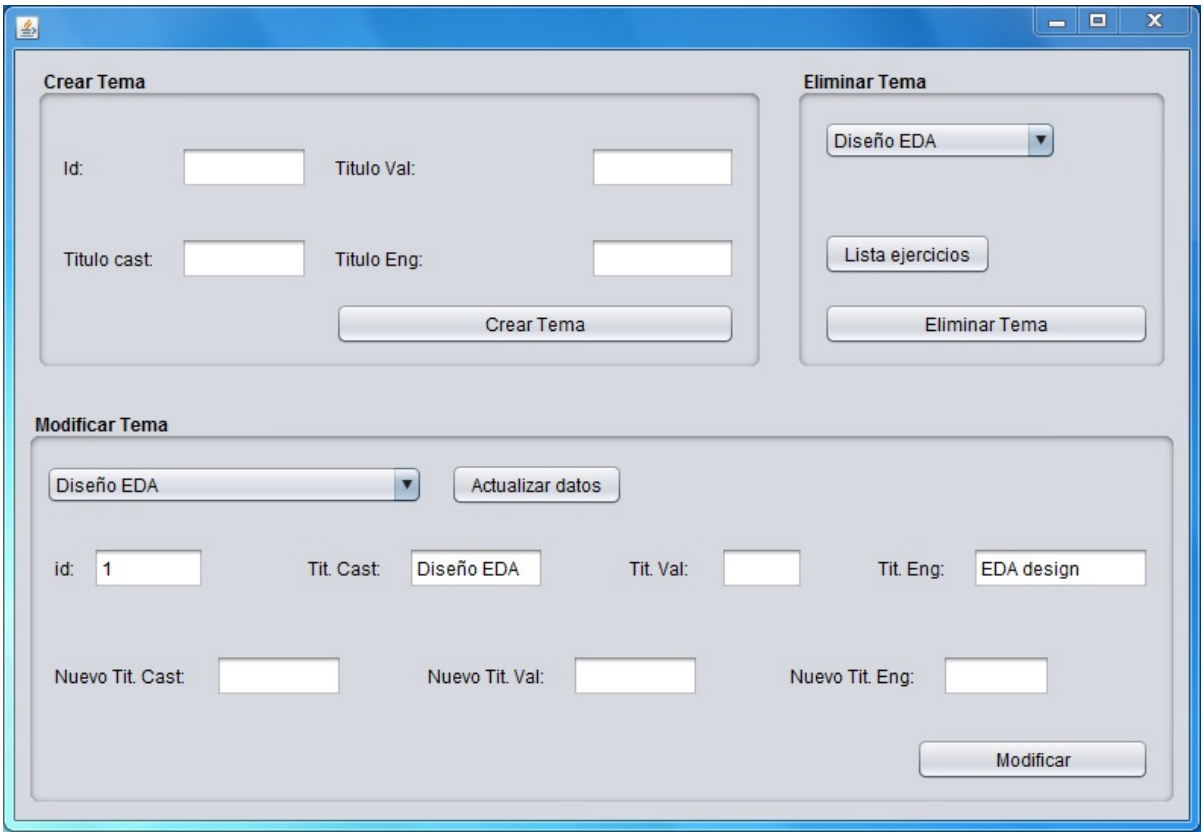

Figura 5.2 Gestionar temas

5.2.2 Eliminar Tema

En caso de que el usuario quiera eliminar un tema, tiene una opción de ver todos los ejercicios asociados a ese tema. Si un tema es eliminado, también se eliminan los ejercicios asociados a este. Antes de realizar esta opción el sistema advierte al usuario de esta condición. Solo se hace efectiva la eliminación de un tema si el usuario está completamente seguro de que quiere eliminarlo.

5.3 Gestión de Clases

Para gestionar las clases se debe acceder a través del menú **Ejercicio Implementación | Gestión Clases**. Como en la ventana de gestión de temas tenemos todas las opciones de gestión de clases centralizadas en un mismo formulario. Desde la misma ventana podemos

crear, modificar o eliminar una clase.

En el panel izquierdo tenemos los apartados correspondientes a la gestión de las clases. Cada uno de estos apartados disponen de un checkbox, el cual nos permite seleccionar la opción correspondiente. Por defecto tendremos la opción de 'Crear Tema' seleccionada. El seleccionar cada uno de los checkbox's nos permite realizar determinadas acciones con el panel de la derecha. En este panel encontramos un Área de Texto (TextArea).

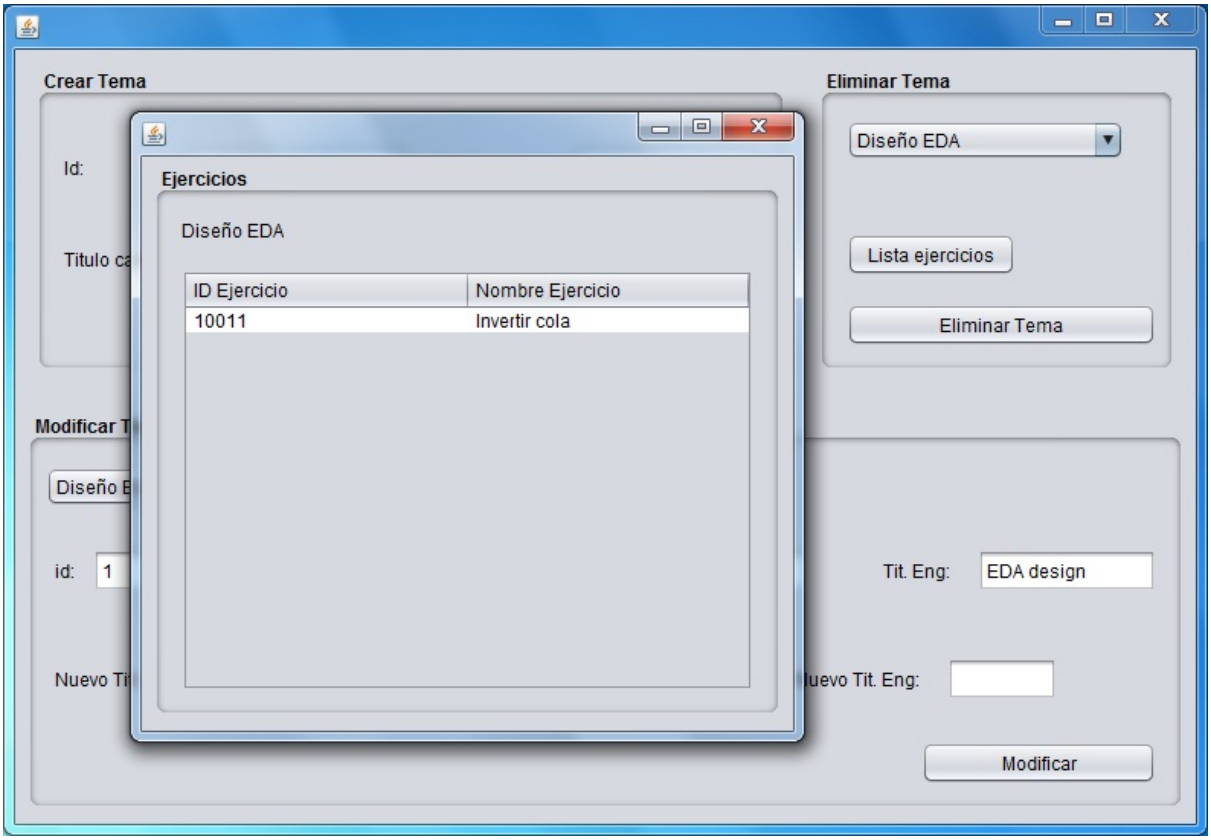

Figura 5.3 Listar ejercicios de un tema

5.3.1 Creación Clase

Cuando se selecciona la opción de creación de clases, esta textarea sirve para introducir el código fuente correspondiente a la clase a crear; Como es habitual, para la creación de una clase nueva se tiene que comprobar que todos los campos necesarios para la creación de la clase no están vacíos (a excepción del textarea), además de ser únicos (no se encuentran en la base de datos).

5.3.2 Modificar Clase

Por otro lado, si seleccionamos modificar una clase, primero debemos seleccionar que clase y versión se quiere modificar. Una vez se selecciona la clase a modificar se rellena la textarea con el código fuente de la clase a modificar. La modificación de una clase consiste en modificar su código fuente. Si se quisiese introducir una nueva versión de una clase ya existente, no se puede modificar el número de versión, se deberá crear con los pasos anteriormente expuestos. Mencionar que en la creación de clases no se pueden crear nuevas clases que ya tengan el mismo nombre y versión que una ya existente.

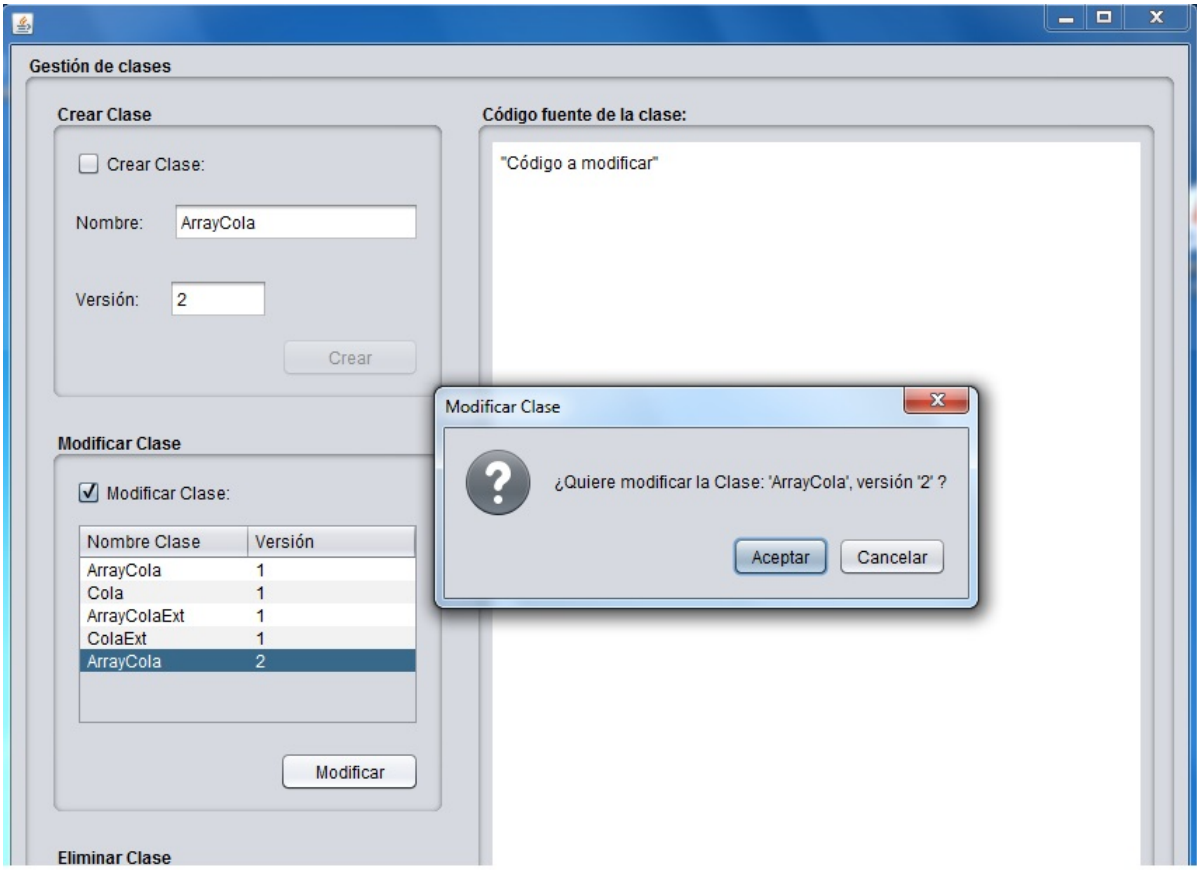

Figura 5.4 Modificación de una clase

5.3.3 Eliminar Clase

Por último, para eliminar una clase primero debemos seleccionar el checkbox correspondiente a esta acción. Una vez ya lo hemos hecho, podemos seleccionar de la lista una de las clases disponibles en la base de datos. En cuanto la seleccionamos aparecerá en el textarea el código fuente asociado a esa clase en particular. En cuanto decidamos eliminar la clase seleccionada saldrá un mensaje de confirmación. Si el usuario está seguro que quiere eliminar esa clase, la clase es eliminada del sistema y se notifica que la acción ha sido llevada a termino correctamente. Este mensaje de confirmación también está presente cuando decidimos modificar el código fuente de una clase ya existente (modificar clase).

5.4 Gestión de Documentos / BlueJ

El profesorado puede gestionar los documentos asociados a los ejercicios desde este formulario. Además de poder añadir o eliminar un documento de un ejercicio puede abrir todas las clases pertinentes a un ejercicio en BlueJ[4]. Esto le permite verificar que un ejercicio determinado contiene todas las clases necesarias para su resolución. Se puede acceder al formulario de gestión de documentos y abrir un ejercicio en BlueJ desde **Ejercicio Implementación | Gestión Documentos/BlueJ**.

5.4.1 Añadir un Documento

Para añadir un documento el usuario lo primero que debe hacer es seleccionar en que ejercicio añadirlo. Para este propósito puede encontrar una lista arriba a la izquierda del formulario donde se muestran todos los ejercicios que existen en la base de datos. Una vez se selecciona un ejercicio aparece en la lista del panel de la derecha todos los documentos asociados a ese ejercicio. Para añadir un documento a ese ejercicio se deberá hacer clic en el botón 'Añadir Documento'. A continuación nos aparecerá una nueva ventana con un nuevo FileChooser desde el cual podremos navegar por el sistema de ficheros hasta encontrar el documento a añadir. Al seleccionar un documento y darle al botón de 'aceptar' se añadirá ese documento a la lista de los documentos asociados al ejercicio anteriormente seleccionado. Por

el contrario, si se aprieta el botón 'Cancelar', como es evidente de cierra la ventana de selección de archivos y se aborta la operación, volviendo al formulario de gestión.

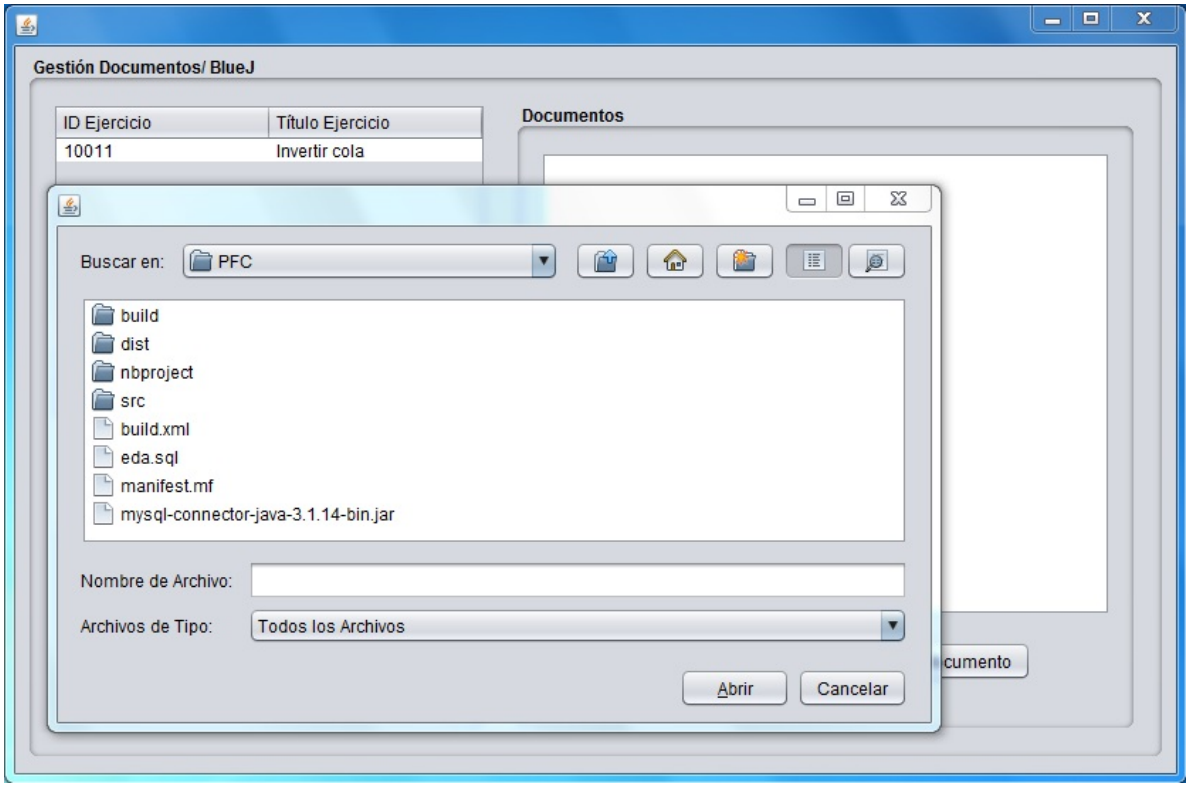

Figura 5.5 Añadir nuevo documento

5.4.2 Eliminar un Documento

Por otra parte, si al seleccionar uno de los ejercicios listados en la parte superior izquierda lo que se pretende es eliminar uno de los documentos, debemos seleccionar uno de estos documentos y hacer clic en el botón 'Eliminar Documento'. Esta acción conlleva la notificación de un mensaje de control. Si el usuario le da a 'Aceptar' en el mensaje de control mencionado anteriormente el documento seleccionado se elimina. De otro modo, si le da a 'Cancelar' o cierra el mensaje la operación de eliminar un documento aborta y se mantiene la información del sistema tal y como estaba.

5.4.3 Abrir ejercicio con BlueJ

Otra de las operaciones de gestión que podemos realizar en este formulario es abrir todas las clases relacionadas con el ejercicio seleccionado (tal y como se describe en los puntos anteriores) en BlueJ.

Para realizar esta acción tan solo debemos seleccionar el ejercicio y hacer clic en el botón que se encuentra abajo a la izquierda del formulario, 'Abrir con BlueJ'. Una vez hacemos clic se abrirá un nuevo proyecto en BlueJ con todas las clases del ejercicio. Esto nos ayuda a ver la relación que hay entre las clases pertinentes, además de que se puede verificar el correcto funcionamiento de todas las clases y si se puede resolver el ejercicio en cuestión.

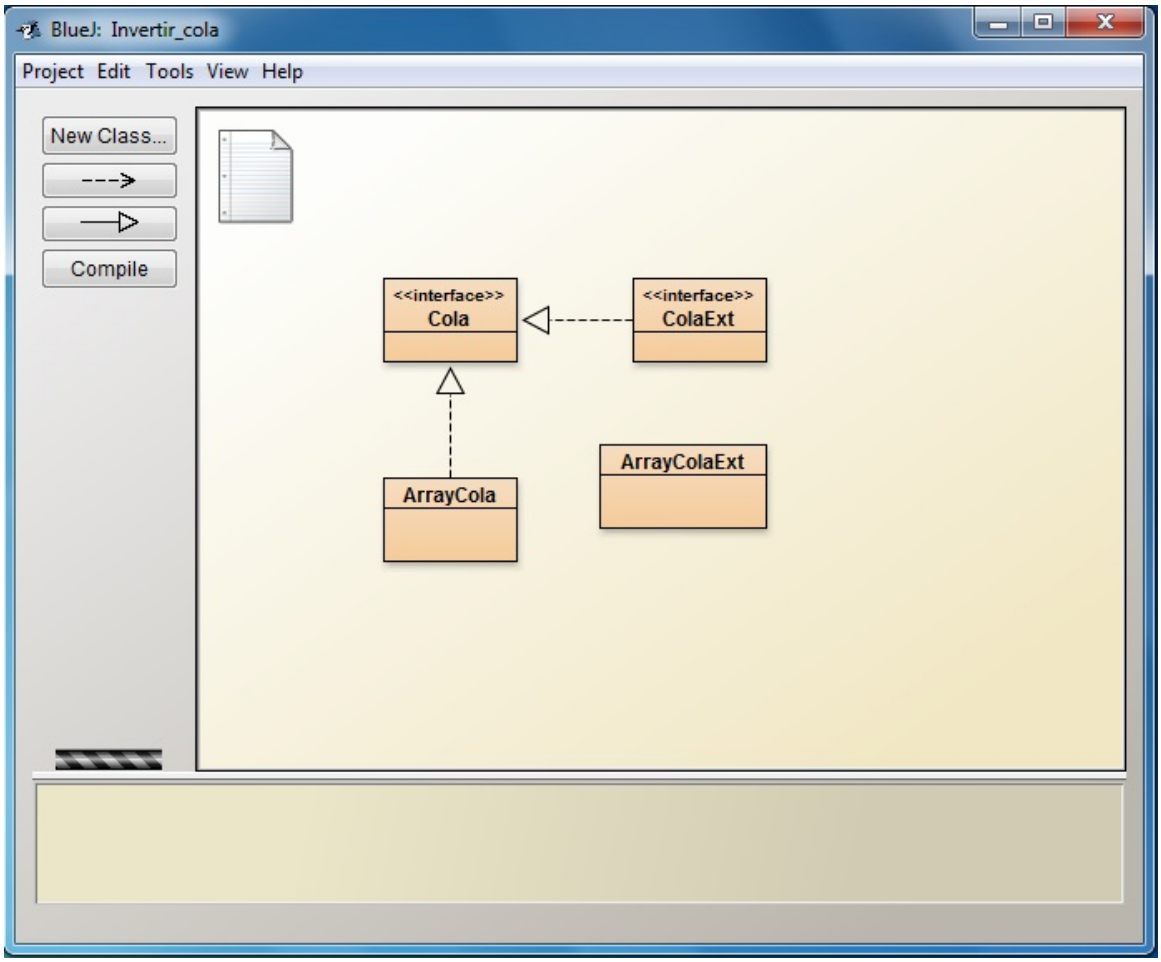

Figura 5.6 Abrir ejercicio con BlueJ

5.5 Notas de los alumnos para un ejercicio determinado

A esta opción se accede mediante el menú **Estadísticas | Notas Ejercicio**.

En este formulario es donde el profesorado puede ver todas las notas de los alumnos para un ejercicio determinado. Se puede seleccionar uno de los ejercicios ya creados en la base de datos. Arriba a la izquierda tenemos un ComboBox que nos permite seleccionar un ejercicio. Una vez seleccionado aparecerán en la tabla de abajo todos los alumnos que han realizado ese ejercicio y las notas que han obtenido. Se puede cribar el resultado de la lista pulsando el chechbox que se encuentra en la parte superior derecha. Esto permite que en la lista solo aparezcan la mejor nota de cada uno de los alumnos, no todas las notas obtenidas en los distintos intentos que haya podido realizar hasta la correcta resolución del problema. Por defecto se muestran todas las notas de los alumnos para el primer ejercicio cargado en el sistema.

5.6 Ejercicios realizados por un alumno determinado.

El usuario también puede ver una lista de los ejercicio que ha realizado una alumno. Esta opción la puede encontrar en **Estadísticas | Ejercicios Alumno**.

De forma análoga al apartado anterior, el usuario puede seleccionar uno de los alumnos que existen en la base de datos y ver los ejercicios que ha solucionado. Al igual que con las notas de los alumnos para un ejercicio, por defecto se muestran los ejercicios realizados por el primer alumno cargado en el sistema.

Capítulo 6

Conclusiones y posibles ampliaciones

Por último presento las conclusiones que he podido obtener con la realización de este Proyecto Final de Carrera. Sin olvidar mencionar las posibles mejoras que se podrían realizar para el software, siempre en pro de la ayuda al profesorado para la realización de ejercicios más completos y eficientes. Esto repercutirá directamente en la calidad de enseñanza ya que el alumnado, al resolver estos ejercicios, afianzará de forma más rápida y concisa los conocimientos que se pretende que adquieran.

6.1 Beneficios del proyecto

El desarrollo de esta aplicación software para el profesorado que vaya a hacer uso de CAP creo que proporciona ciertos beneficios que estimo son indispensables para una mejor gestión de los ejercicios. Considero que estos beneficios son los siguientes:

- Gestión centralizada de los distintos items necesarios para realizar los ejercicios a proponer al alumnado. Esto ayuda a que la realización de los mismos sea cada vez más rápida y eficiente.

- Consta de una interfaz gráfica de usuario fácil e intuitiva de manejar, donde todas las acciones de gestión que se pueden realizar están centralizadas en los mismos formularios. i.e. toda la gestión de los temas se puede realizar desde la misma ventana.

- El acceso a las funciones que puede desear realizar el usuario es rápido y eficiente. La distribución en el menú de las distintas operaciones de gestión es muy intuitiva.

- Se pueden hacer consultas de las notas de los alumnos de forma fácil. Además de ofrecer la opción de ver que alumnos han realizado determinado ejercicio.

- Ofrece la posibilidad de comprobar que los ejercicios desarrollados son correctos, ya que se ofrece la posibilidad de abrirlo con BlueJ para este determinado fin.

- Se puede introducir en otro software ya desarrollado para la gestión de CAP. Así aumentaría la funcionalidad.

- Considero que es fácilmente ampliable con nuevas funcionalidades.

6.2 Posibles ampliaciones

Ya que este software se ha desarrollado para funciones docentes, en este contexto pueden acontecer diferentes cambios a tener en cuenta. Puede que los criterios de corrección, las metodologías de enseñanza o incluso las necesidades de los alumnos cambien en cualquier momento. Sin embargo considero que las ampliaciones más interesantes son aquellas que ayuden al profesorado a identificar y analizar las necesidades del alumnado.

Estas ampliaciones pueden ser las siguientes:

- Realizar el listado de las notas para un ejercicio determinado agrupado por alumnos de un determinado grupo docente. Esto puede ayudar a los profesores a identificar que grupos van adquiriendo los conocimientos pretendidos antes y/o mejor. Con esta información se puede decidir tomar alguna acción determinada, como la revisión del material docente usado por cada uno de los profesores correspondientes, o simplemente hacer más hincapié en determinados aspectos teóricos o prácticos.

- Crear un módulo que permita realizar gráficas con las notas de determinado alumno para todos los ejercicios que ha realizado. Estas gráficas podrían contener las notas del alumno seleccionado, comparándolas con las notas medias obtenidas por los demás alumnos del curso, o por los alumnos de su mismo grupo.

- Del mismo modo que en el punto anterior, realizar gráficas con la información de los ejercicios que han sido resueltos con la nota media más alta. Ya podría ser bien por grupos o de forma global, para todo el curso. Esto aporta mucha información al respecto de que ejercicios deberían ser revisados, o que temas deberían ser explicados de nuevo o reforzarlos de algún modo (ya sea con la implementación de nuevos ejercicios o con un cambio de estrategia docente).

- Considero que también podría ser interesante guardar toda esta información (notas, ejercicio, estadísticas, etc.) de años anteriores en algún fichero que se pudiese importar al sistema. Con esto se podrían comparar resultados de un año a otro, identificando que ejercicios o incluso que cambios han sido beneficiosos para el aprendizaje del alumnado.

Estas cuatro ampliaciones creo que serían las más adecuadas para el correcto seguimiento del alumnado. Esto se enmarca de forma excelente en el sistema docente actual de Grado, donde se debe evaluar al alumno de forma continua.

6.3 Conclusiones

La realización de este proyecto me ha servido para comprobar si las supuestas habilidades que había conseguido a lo largo de la carrera podían ser útiles. Efectivamente, creo que la realización de un software que es funcional y que en principio tendrá un uso concreto, me ha ayudado a ser consciente de mis propias habilidades y ganar conciencia de todo lo aprendido a lo largo de cursar las distintas asignaturas de la carrera.

Considero que el enfrentarse a un problema 'real' (no simplemente prácticas de laboratorio) es lo que determina si realmente se han adquirido realmente todos los conocimientos que se pretendía. También creo que no se trata de aplicar todos estos conocimientos tal y como nos han ido indicando, si no que se trata mas bien de poder ser capaces de investigar de forma independiente un problema nuevo que debamos resolver, y poder completarlo de forma correcta.

ANEXOS

Anexo A

Instalación y guía básica de uso del IDE Netbeans

El IDE [5] que se ha utilizado para la realización de este proyecto final de carrera ha sido Netbeans [6] 7.3.1. A continuación se detallarán los pasos a seguir para la instalación y el uso básico de este IDE para la creación de soluciones software.

El uso de Netbeans creo que ha sido el más adecuado para este proyecto ya que proporciona unas herramientas fáciles de gestionar para la creación de interfaces gráficas de usuario (IGU).

Para comenzar debemos iniciar la aplicación. Una vez ya la tenemos en marcha veremos la pantalla de inicio, donde podemos ver las noticias relacionada con Netbeans y distintas opciones de ayuda para el usuario no experimentado con este software.

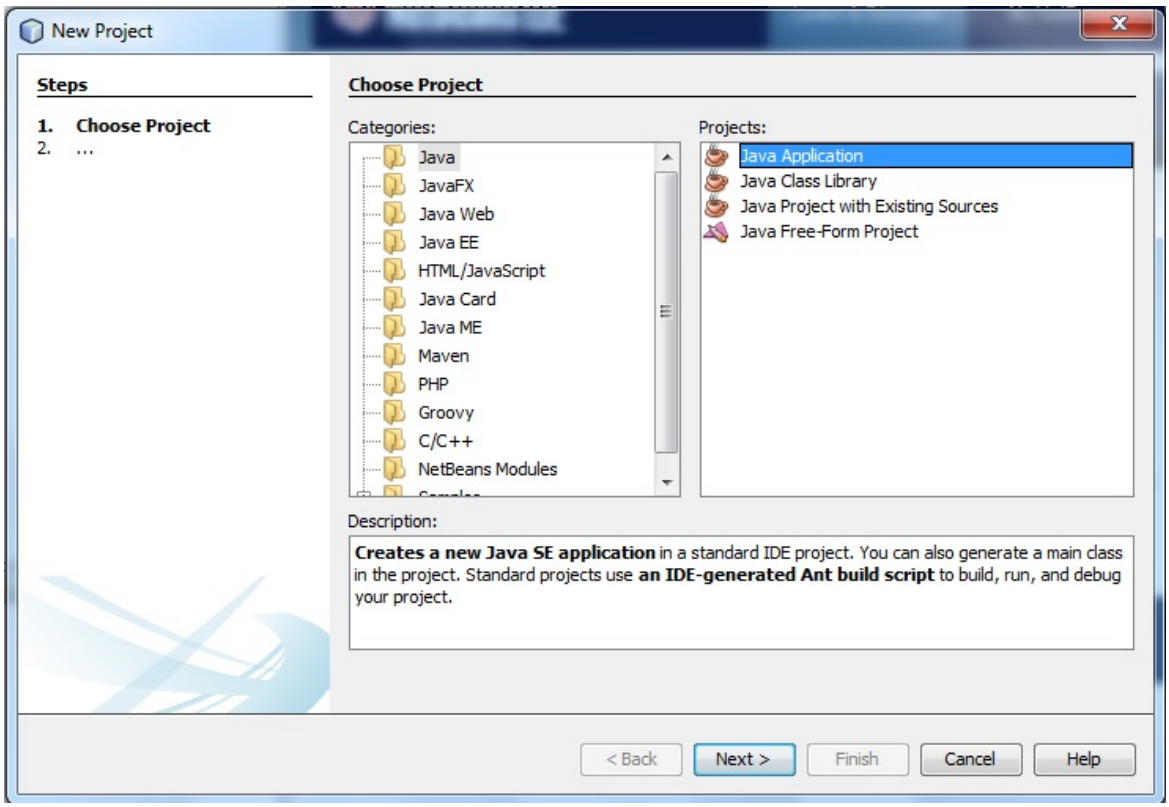

Figura A.1 Nuevo proyecto

Con el propósito de empezar la creación de un nuevo proyecto Java debemos dirigirnos a **File | New Project | Java | Java application** . Una vez hemos seleccionado esa ruta nos pide que asignemos distintos parámetros al nuevo proyecto, tales como el nombre, la localización o la carpeta donde se irán guardando todos los archivos correspondientes del mismo. Podemos decidir si nuestro proyecto debe o no tener una clase Main.

Una vez creado nos aparecerá en el panel de la izquierda un nuevo símbolo (una taza de café) desplegable, con el nombre que le hayamos asignado al proyecto. Si lo desplegamos podemos ver que por defecto tiene dos paquetes. Estos son 'Source packages' y 'Libraries'.

Podemos añadir mas paquetes haciendo clic derecho encima del proyecto, **new | Java package**, le asignamos un nombre y nos aparecerá el nuevo paquete creado dentro del proyecto. Estos paquetes nos sirven para estructurar el proyecto por capas, o por funcionalidades, o sea como fuere que hayamos decidido gestionarlo.

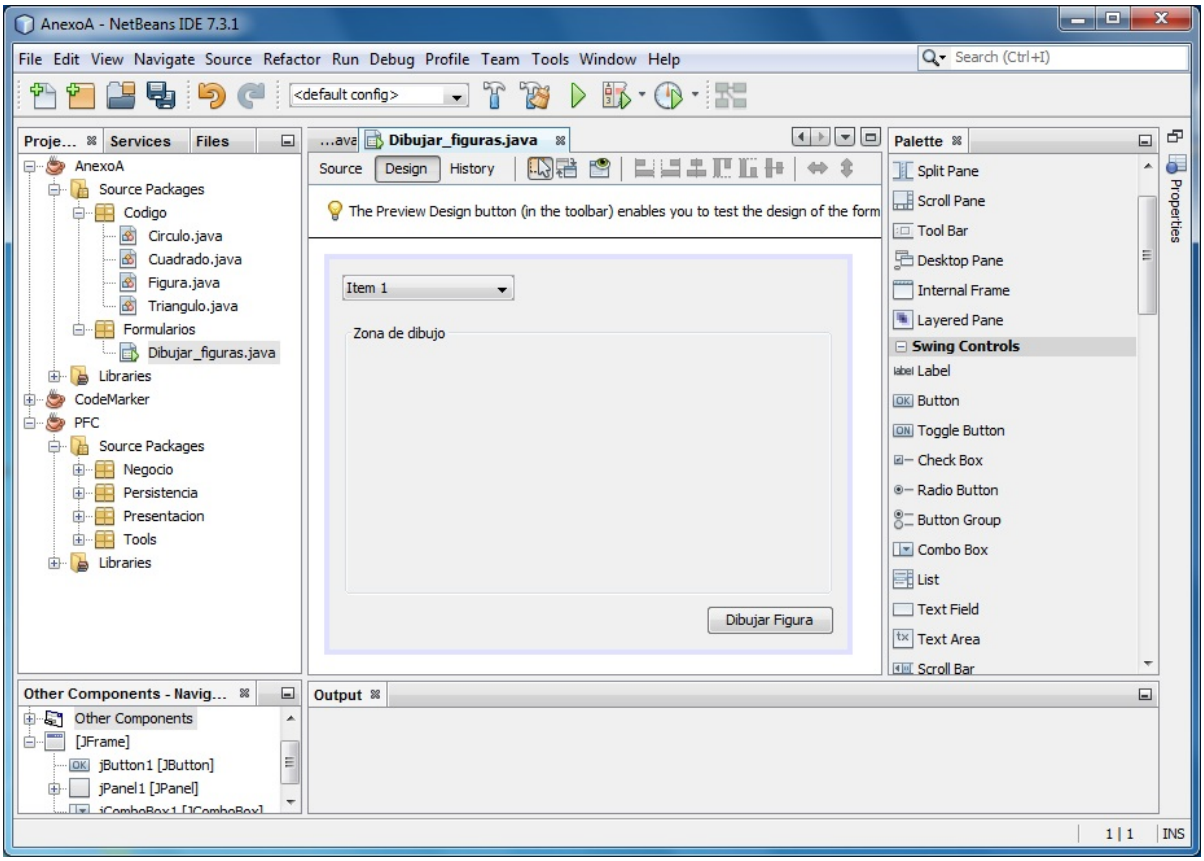

Figura A.2 Ejemplo proyecto

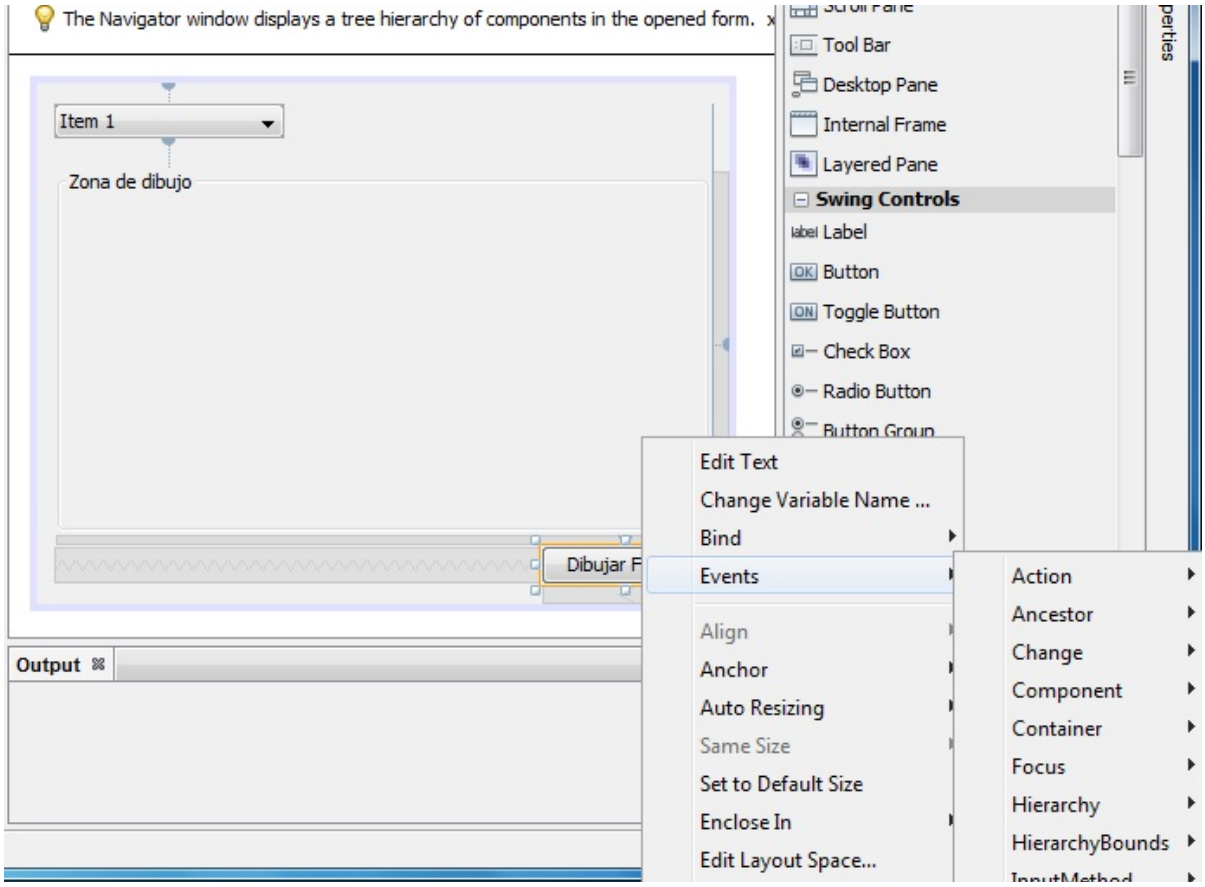

Figura A.3 Eventos

Dentro de los paquetes iremos colocando las distintas clases, formularios o distintos tipos de archivos necesarios para la realización del proyecto. En las Java Classes es donde diseñaremos las clases pertinentes. Aquí se diseñará la capa de negocio, tal y como hemos visto en el marco teórico.

Por otro lado tenemos los Java Forms, que son los documentos que nos permiten diseñar la interfaz gráfica de usuario. En estos formularios encontramos cantidad de componentes que nos pueden interesar para nuestra interfaz. Dichos componentes pueden ser desde botones, ventanas de selección, menús, etc. Si el usuario ya está familiarizado con las interfaces de usuario le será muy intuitivo. Netbeans tiene la facilidad de que los componentes para la interfaz solo los tienes que elegir desde una paleta y arrastrarlos hasta la zona deseada dentro de la ventana. No hay que codificar cada uno de los componentes, ya que Netbeans genera de forma automática el código necesario.

A parte de esta principal ventaja con respecto a la creación de una interfaz de usuario, Netbeans también nos proporciona la opción de asignar distintos tipos de eventos a los elementos que queramos. Esto nos permite, por ejemplo, que se activen determinadas funcionalidades cuando el usuario le da a un botón, o gestionar la funcionalidad interna del programa.

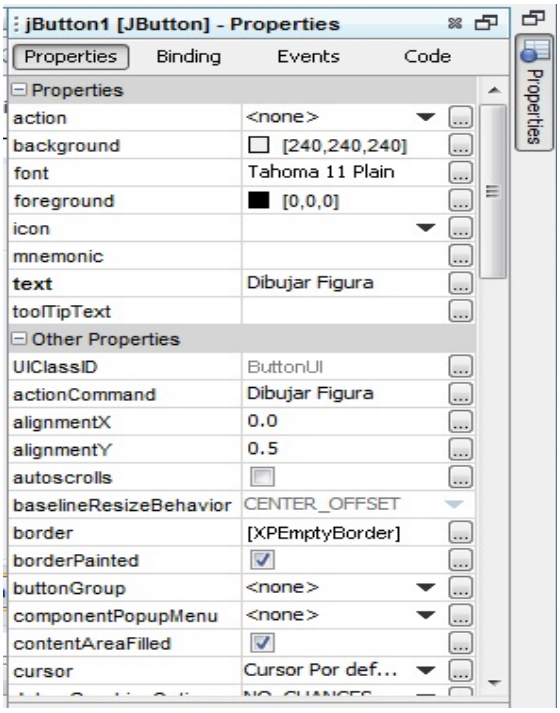

Figura A.4 Propiedades de los elementos

Anexo B

Instalación y gestión básica de MySQL

Para uso de una base de datos el usuario debería tener un sistema de gestión de bases de datos con el que poder trabajar. Para este proyecto se ha necesitado dicho sistema SGBD [6] ya que se pretende que la aplicación almacenase de forma permanente toda la información necesaria.

El sistema SGBD a instalar en el sistema es MySQL Workbench. Para este fin debemos ir a la página oficial de oracle/MySQL y bajar el instalador. Una vez ya tenemos el instalador en el sistema debemos ejecutarlo y seleccionar las acciones que queremos realizar.

MySQL Workbench es una aplicación para el diseño y documentación de bases de datos pensada para ser usada con el sistema de gestión de bases de datos MySQL. Por esta razón en el instalador nos da la opción de instalar ambos sistemas. (MySQL Server y la aplicación MySQL Workbench CE 6.0.7.)

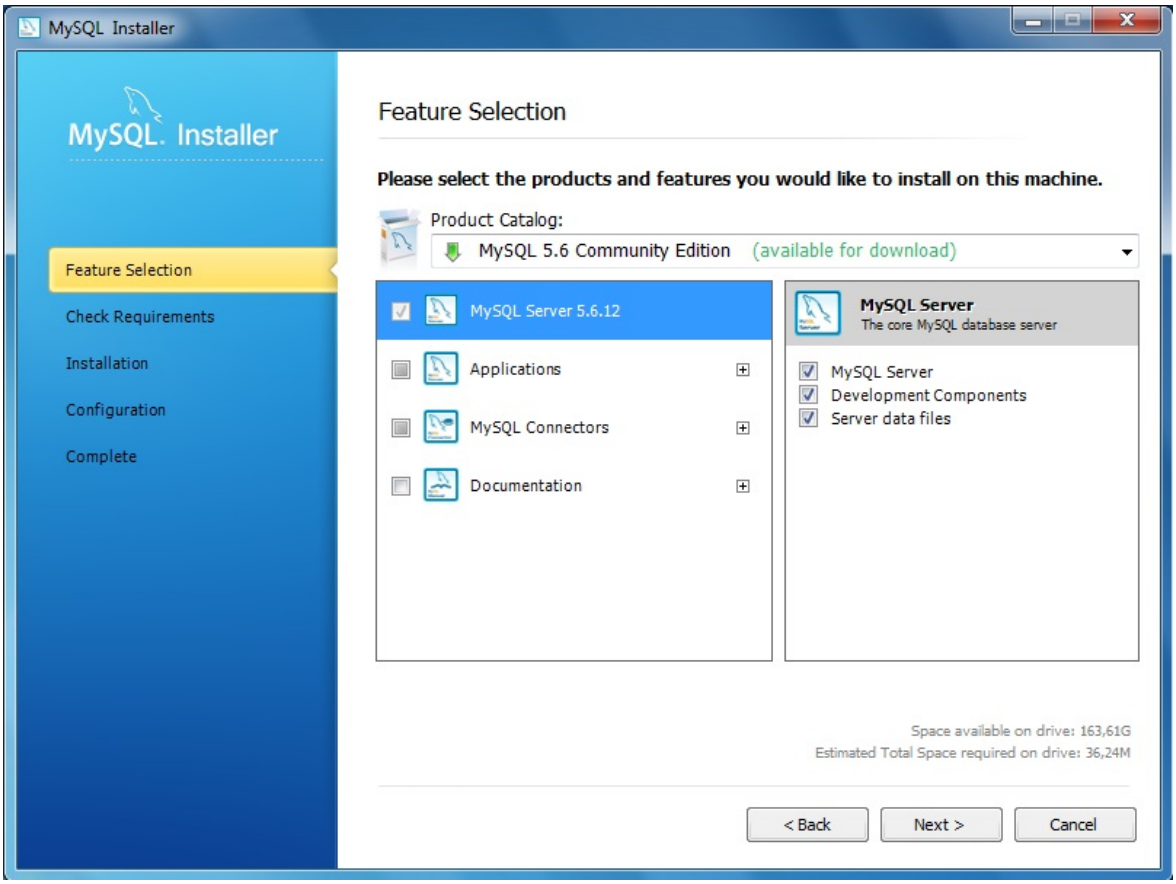

Figura B.1 Instalador MySQL

Sin embargo para la realización del proyecto se puede instalar el MySQL Server, sin necesidad de MySQL Workbench, ya que no vamos a hacer una gestión muy exhaustiva de la base de datos. Tan solo vamos a necesitar el server MySQL para poder vincular la base de dato proporcionada por el profesorado a la aplicación software realizada en este proyecto.

La gestión de una base de datos [9] a través de MySQL es a base de comandos de consola. Para inicializar la consola tan solo debemos ir al directorio donde hayamos instalado el software y abrir el cliente ejecutable 'Command Line Client – Unicode'.

Una vez introducimos la contraseña correspondiente para identificarnos en el sistema ya podremos realizar la gestión que deseemos en nuestras bases de datos. Podemos importar bases de datos .sql al servidor mediante la siguiente secuencia de comandos:

- CREATE DATABASE 'nombre_base_datos'. Con este comando creamos una nueva base de datos con nombre 'nombre_base_datos' vacía.

- USE 'nombre_base_datos'. A partir de ahora se usará la base de datos 'nombre_base_datos' para todos los comandos que insertemos.

- SOURCE /ruta/del/archivo/'nombre_base_datos'.sql. Ahora ya hemos cargado la base de datos que tenemos en un archivo .sql a nuestra base de datos 'nombre_base_datos' en nuestro sistema.

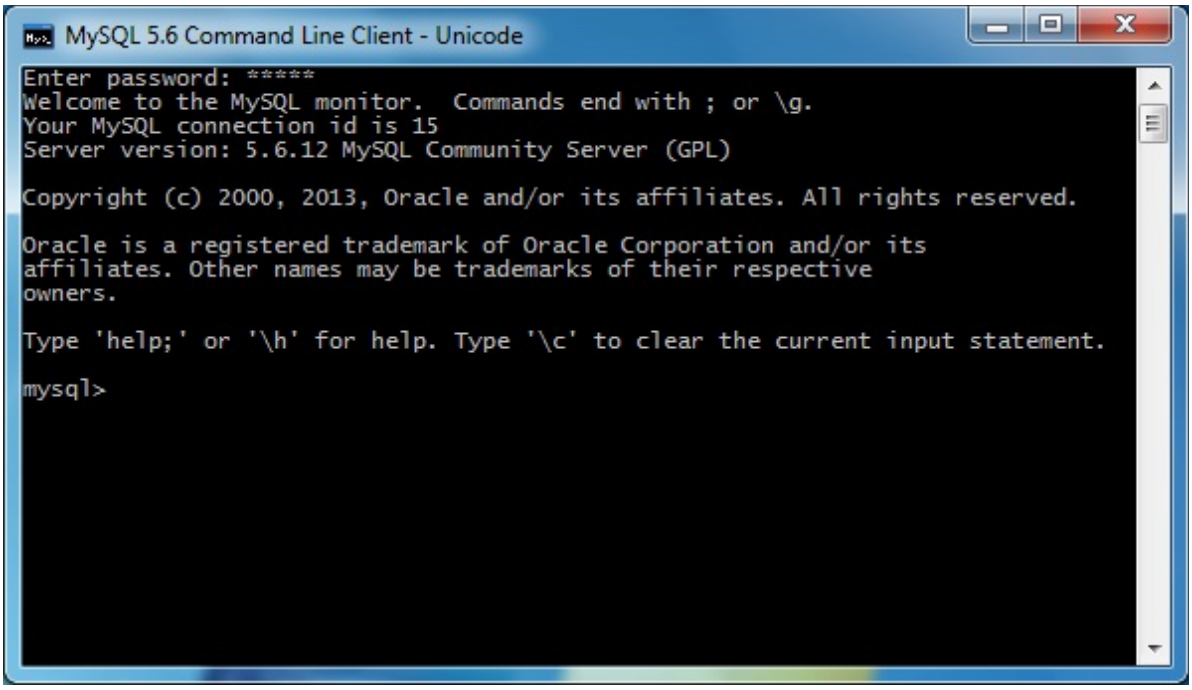

Figura B.2 Consola MySQL Line Client

Ahora ya podemos comprobar si la base de datos se ha importado correctamente si le decimos al servido que nos muestre las tablas presentes en la base de datos con nombre 'nombre_base_datos'. Para ello debemos usar el comando 'SHOW TABLES;'. Entonces aparecerán en la consola las tablas presentes en nuestra base de datos.

A partir de este momento ya se puede usar esta base de datos para usarla conjuntamente con Netbeans para la creación del proyecto y usar el conector para bases de datos de java lbdc:mysql.

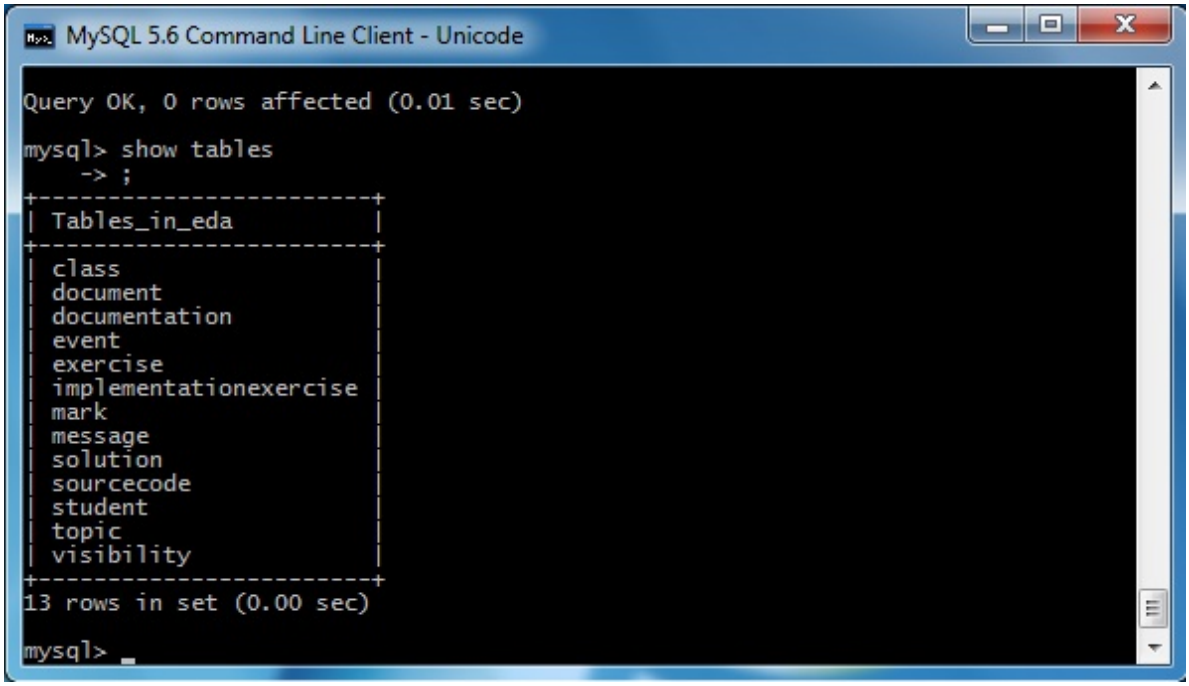

Figura B.3 Show tables

Referencias

[1] Std.610-1990, IEEE Standard Glossary of Software Engineering Terminology. http://www.ieee.org/

[2] Std.830-1998, IEEE Standard Glossary of Software Engineering Terminology. http://www.ieee.org/

[3] O.Sapena, M.Galiano, N.Prieto y M.Llorens. Evaluación continua con CAP, un Corrector Automático de tareas de Programación. En Jornadas de Innovación Educativa de la UPV, 2012.

[4] BlueJ – The interactive Java environment. http://www.bluej.org/

[5] IDE – Siglas anglosajonas para el termino Entorno de Desarrollo Integrado. http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[6] Netbeans. https://netbeans.org/

[7] T. Budd. Introducción a la programación orientada a objetos. Addison-Wesley, 1994.

[8] Sommerville, I. Ingeniería del Software. Addison-Wesley, 2002

[9] Guía rápida para la gestión de una base de datos. http://www.xtec.cat/~acastan/textos/Administracion%20de%20MySQL.html#EJEMPLO

[10] G. Booch. El lenguaje unificado de modelado. Guía de usuario. Addison-Wesley, 2000.

[11] M. Celma, J. C. Casamayor y L. Mota. Bases de datos relacionales. Servicio de Publicaciones UPV. SPUPV-97.767, 1997.

[12] B. Shneiderman. Designing the User Interface: Strategies for Effective Human-Computer Interaction – 3rd Ed. Addison-Wesley, Massachusetts, 1998.