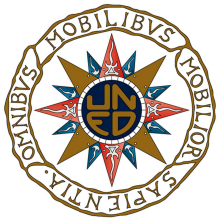


UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Carrera de Ingeniero Informático

IMPLEMENTACIÓN, DESPLIEGUE Y ESTUDIO DE LA EFICIENCIA DE UNA LIBRERÍA DE GENERACIÓN AUTOMÁTICA DE PREGUNTAS

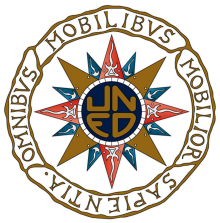
JUAN MANUEL SÁNCHEZ TURRIÓN

Dirigido por: MANUEL LUQUE GALLEGO

y por: JOSÉ MANUEL CUADRA TRONCOSO

Codirigido por: RICARDO CONEJO MUÑOZ

Curso: 2012/2013 (convocatoria de Marzo)



IMPLEMENTACIÓN, DESPLIEGUE Y ESTUDIO DE LA EFICIENCIA DE UNA LIBRERÍA DE GENERACIÓN AUTOMÁTICA DE PREGUNTAS

Proyecto de Fin de Carrera de modalidad *oferta específica*

Realizado por: JUAN MANUEL SÁNCHEZ TURRIÓN (firma)

Dirigido por: MANUEL LUQUE GALLEGO (firma)

y por: JOSÉ MANUEL CUADRA TRONCOSO (firma)

Codirigido por: RICARDO CONEJO MUÑOZ (firma)

Tribunal calificador:

Presidente: D./D^{a}. . . . . .

(firma)

Secretario: D./D^{a}. . . . . .

(firma)

Vocal: D./D^{a}. . . . . .

(firma)

Fecha de lectura y defensa: . . . . . .

Calificación: . . . . . .

Para Ana, sin quién este proyecto simplemente no habría sido posible.

Resumen

Título del proyecto: IMPLEMENTACIÓN, DESPLIEGUE Y ESTUDIO DE LA EFICIENCIA DE UNA LIBRERÍA DE GENERACIÓN AUTOMÁTICA DE PREGUNTAS

Resumen: Debido al número creciente de alumnos en la Universidad Nacional de Educación a Distancia, junto con el énfasis que el Espacio Europeo de Educación Superior pone en que el alumno realice actividades didácticas de forma autónoma, es necesario que los docentes proporcionen un número suficiente de estas actividades para aumentar las posibilidades formativas que se ofrecen a los alumnos.

La realización de estas actividades exige un cierto tiempo a los profesores que deben de realizar muchas otras tareas para formar a los estudiantes.

El presente trabajo proporciona una librería de generación automática de preguntas de tipo test en el ámbito de la teoría básica de conjuntos para ser utilizada en la plataforma web www.siette.org permitiendo su utilización a distancia y en cualquier momento.

Las preguntas generadas cubren conceptos relativos a conceptos básicos de conjuntos, así como sus operaciones y propiedades básicas. De igual modo, se cubren ciertos aspectos relacionados con las tuplas.

Puesto que la plataforma Siette aloja otros trabajos, se ha realizado un considerable esfuerzo en verificar que la librería generaba las preguntas en tiempos razonables para no influir en el rendimiento general de la plataforma y ofrecer una adecuada experiencia de usuario.

El resultado es una herramienta que contribuye positivamente como apoyo al aprendizaje de los conceptos cubiertos.

Palabras clave: Generación automática de preguntas, test, teoría de conjuntos, Siette, rendimiento.

Abstract

Project title: IMPLEMENTATION, DEPLOYMENT AND PERFORMANCE STUDY OF AN AUTOMATIC QUESTIONS GENERATION LIBRARY

Abstract: Due to the growing number of students at the Universidad Nacional de Educación a Distancia, along with the emphasis the European Space of Higher Education puts in the student performing autonomously learning activities, it is necessary for the teachers to provide a sufficient number of these activities to increase training opportunities offered to the students.

The implemementation of these activities requires some time to teachers who must perform many other tasks to train students.

This paper provides a library of automatic generation of multiple choice questions in the field of basic set theory for use on the web platform www.siette.org allowing remote use at any time.

The questions generated cover basics concepts of sets, their operations and basic properties. Likewise, they covered certain aspects of tuples.

Since Siette platform hosts other works, it has been made a considerable effort to verify the library generates questions in reasonable time in order not to influence the overall performance of the platform and to offer an adequate user experience.

The result is a tool that contributes positively to support the learning of the concepts covered.

Keywords: Automatic generation of questions, test, set theory, Siette, performance.

Agradecimientos

Este proyecto ha sido posible gracias al apoyo y colaboración de muchas personas, a todas ellas quiero mostrarles mi más profundo agradecimiento por ayudarme a finalizar otra etapa más de mi vida y enseñarme todo lo que he aprendido.

Para comenzar quiero agradecer este proyecto a los directores de este proyecto, Manuel Luque Gallego y José Manuel Cuadra Troncoso, que siempre han estado accesibles para ayudarme en todo lo que les he pedido para asegurar el buen fin de este proyecto.

Así mismo deseo resaltar toda la colaboración prestada por el profesor Ricardo Conejo Muñoz durante el desarrollo de este proyecto, sin sus ayudas y consejos sobre la plataforma Siette la dificultad de este proyecto se habría incrementado en gran medida.

Seguidamente quiero dar las gracias a mi familia (en especial a los más cercanos: Ana, Alba, Mary, Mamá y Papá) y amigos, que siempre han estado dispuestos a soportarme en los momentos más duros de este camino, facilitándome el trabajo en todo lo que han podido y sacrificando la cercanía para permitirme llegar con éxito al final de este duro camino.

Continúo dando las gracias a mis compañeros y amigos del trabajo, que siempre me han apoyado cargando a veces con tareas que no les correspondían para que pudiera disponer de un poco más de tiempo, así como aportando ideas y sugiriendo mejoras.

No quiero olvidarme de Dios, que me ha permitido no caer enfermo (en especial en la recta final del desarrollo del proyecto) pese a los excesos que el tesón ha exigido de mi.

Y para terminar, quiero agradecérselo especialmente (de nuevo, ya que nunca será suficiente) a Ana, que en todo momento me ha ayudado, física y emocionalmente, dándome ánimos siempre que lo he necesitado y liberándome de cuantas tareas mundanas ha podido. A ti, a parte de darte las gracias, te dedico este proyecto.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo introductorio se presenta el proyecto fin de carrera *"Implementación, despliegue y estudio de la eficiencia de una librería de generación automática de preguntas"*.

1.1. Motivación del proyecto

Debido a la actual coyuntura socio-económica, el número de alumnos en la Universidad Nacional de Educación a Distancia (UNED) está elevándose de forma constante, ya que la enseñanza a distancia ofertada por la UNED supone un aumento de flexibilidad para el alumnado que añade un valor diferenciador para este modelo de enseñanza con respecto a los modelos tradicionales de enseñanza presencial.

Por otro lado, el énfasis que el Espacio Europeo de Educación Superior (EEES) pone en que el alumno realice actividades didácticas de forma autónoma, se traduce en la necesidad de proporcionar, por parte del personal docente, un número suficiente de estas actividades para aumentar la oferta formativa que se ofrece a los alumnos, así como fomentar la autonomía en el aprendizaje de los mismos. Estas nuevas circunstancias generan un esfuerzo adicional al personal docente, que debe realizar muchas otras tareas para llevar a cabo la labor de formar a los estudiantes.

De la necesidad de reducir las nuevas cargas al personal docente surge la motivación para aportar una solución que contribuya a facilitar, en todo o en parte, las labores que implican

las nuevas circunstancias que concurren en el momento actual.

El aporte del presente trabajo se centra en el ámbito de la generación automática de preguntas con el fin de contribuir a ofrecer a los alumnos un alto grado de personalización de las actividades propuestas y de facilitar al personal docente la realización de un elevado número de ejercicios para los alumnos con un menor coste de tiempo.

Puesto que el ámbito de la generación de preguntas es excesivamente amplio se ha centrado en el dominio de la Teoría básica de conjuntos, debido a que este es un dominio que por sus importantes bases matemáticas es más fácilmente modelable en términos informáticos que aquellos otros que no cuentan con una relación tan estrecha con el mundo matemático.

Se ha tratado pues, de cubrir una necesidad real con los conocimientos adquiridos durante la carrera, demostrando así tanto el dominio de los conocimientos como la aplicación de los mismos.

1.2. Finalidad del proyecto

Cuando una persona ha ido avanzando a lo largo de su formación académica, la perspectiva de alumno de la enseñanza ha sido observada en multitud de ocasiones, pero la perspectiva del profesorado, muchas veces no ha sido adecuadamente percibida.

En este proyecto de fin de carrera, es necesario situarse en esa nueva posición de observador de la enseñanza para asumir los retos que se plantean, ya que el profesor debe acompañar al alumno en su labor de aprendizaje y, entre las muchas tareas que este hecho supone, se encuentra la de proponer actividades que permitan al alumno lograr al menos tres objetivos primordiales:

- Ejercitar los conocimientos adquiridos en fases previas del proceso de aprendizaje.
- Adquirir nuevos conocimientos que se hayan pasado por alto en fases previas del proceso de aprendizaje.
- Evaluar el nivel de los conocimientos adquiridos.

Por lo tanto, para favorecer la consecución del primer objetivo, han de proponerse actividades que recorran todo el espectro de conocimiento que se pretende ejercitar. Cuanto más diversas sean las actividades, más productivo será para el alumno.

En cuanto al segundo objetivo, es importante que las actividades propuestas faciliten ayudas que permitan al estudiante obtener conocimientos adicionales.

De cara a alcanzar el tercer objetivo, las actividades propuestas han de ser fácilmente evaluables.

Para ayudar al profesor a alcanzar estos objetivos genéricos, se ha desarrollado un sistema que automatice el trabajo de generación y corrección de actividades, al menos en la medida de lo posible.

1.2.1. Objetivos del proyecto

A continuación se comentan brevemente los principales objetivos perseguidos con la realización de este proyecto:

1. Realizar un estudio sobre el dominio de conocimiento a implementar, en este caso *"Teoría básica de conjuntos"*.

2. Implementar una solución al problema planteado que utilice únicamente software de libre distribución.

3. Desplegar en la plataforma web Siette el sistema desarrollado y utilizarlo desde la misma.

4. Aplicar los conocimientos adquiridos durante la realización de los estudios de Ingeniero en Informática para la creación de un proyecto funcional y bien documentado.

5. Sentar las bases para extender el trabajo realizado al resto del dominio de conocimiento abordado o incluso a otros dominios diferentes.

1.3. Desarrollo del proyecto

Para llevar a cabo este proyecto se ha implementado una librería con capacidad para generar cuestiones relacionadas con el dominio elegido (como se expondrá más adelante con más detalle: *"Teoría básica de conjuntos"*). Posteriormente se han realizado pruebas sobre la librería que aseguraban que se cumplían los requisitos funcionales y de rendimiento especificados, así como que los errores eran tratados adecuadamente. Y ya para finalizar, se desplegará la librería implementada en una plataforma web (www.siette.org) que facilita el uso de la misma por parte del alumnado.

En este trabajo se ha elegido como ciclo de vida del software el ciclo de vida evolutivo expuesto en [Agustín et al., 2002], ya que las características del proyecto indicaban que este podría ser un buen modelo, puesto que no necesita conocer todos los requisitos al comienzo y permite realizar añadidos de nuevas funcionalidades tras disponer de productos entregables finalizados.

En la figura 1.1 se puede el citado modelo de ciclo de vida.

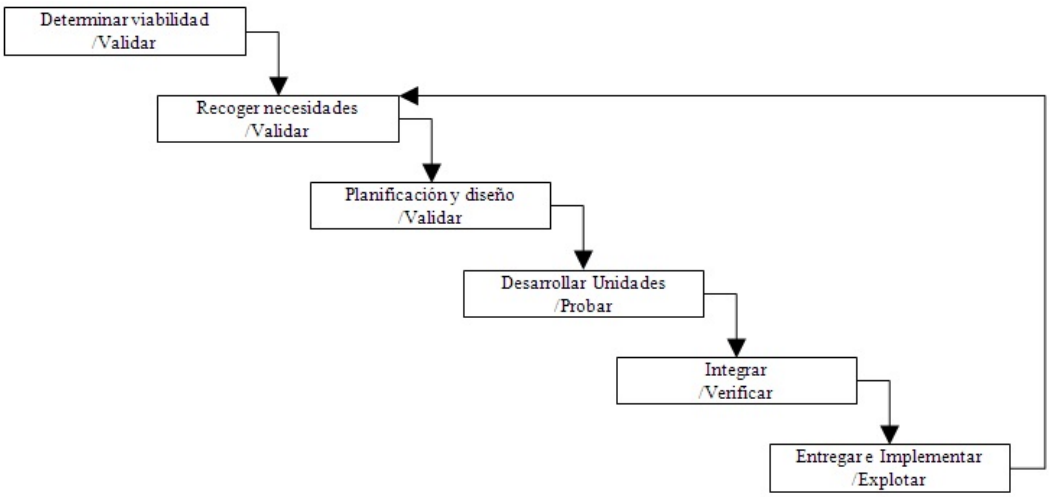


Figura 1.1: Modelo de ciclo de vida evolutivo.

La naturaleza del dominio cubierto también contribuía positivamente a la elección de este modelo de ciclo de vida, ya que los elementos individuales que constituyen el desarrollo son bastante independientes y modulares, permitiendo abordar cada uno de ellos en el momento

más oportuno.

Inicialmente se ha desarrollado un prototipo sencillo en un dominio (la aritmética básica) algo diferente del finalmente elegido, para familiarizarse con el entorno, el lenguaje de programación, las herramientas y las características específicas del tipo de desarrollo, así como facilitar la estimación del esfuerzo del trabajo final.

Posteriormente se ha realizado un análisis básico sobre el dominio definitivo a cubrir y tras ello se ha desarrollado el sistema final con unos requisitos iniciales que se fueron ampliando en cada una de las distintas evoluciones del proyecto.

Debido a los requisitos del proyecto, que hacían especial hincapié en las pruebas del sistema, y en la necesidad del despliegue de la librería en el sistema Siette, se han agrupado y renombrado las diversas fases del ciclo de vida evolutivo en tres grandes fases. A continuación se comentan brevemente estas fases, que serán posteriormente ampliadas a lo largo de esta memoria.

Fase 1: Desarrollo de la librería

Esta fase incluye la revisión de las necesidades del cliente, la planificación y el diseño de la solución y la implementación de la solución diseñada.

Para la documentación de esta fase se ha utilizado el lenguaje de modelado UML (Unified Modeling Language) como herramienta principal.

Tras finalizar cada una de estas fases de desarrollo, se han realizado pruebas que permitían verificar que las nuevas funcionalidades implementadas se ajustaban correctamente a lo especificado como se describe a continuación.

Fase 2: Realización de pruebas de la librería

Las pruebas que se realizaban en cada evolución eran de tres tipos:

- Unitarias, para comprobar el correcto funcionamiento de cada uno de los módulos implementados.

- De rendimiento, para comprobar que la generación de preguntas se realiza en tiempos aceptables.

- Funcionales, para comprobar que la funcionalidad implementada era la solicitada. En este caso se dividieron las pruebas en dos subtipos:

    - Simuladas*, para comprobar la funcionalidad implementada sin interferir en el sistema Siette.

    - Definitivas†, para verificar que la funcionalidad implementada se comportaba adecuadamente al integrarse la librería en el sistema Siette. Estas pruebas se realizaban después de la fase de Despliegue de la librería.

Una vez superadas las pruebas, se daba paso a la siguiente fase, el despliegue de la librería en la plataforma web.

Fase 3: Despliegue de la librería

En esta fase se ha realizado tanto el despliegue de la librería en la plataforma web Siette (para comprobar la correcta integración entre la librería y Siette), como la implementación en la plataforma de una asignatura‡ que haga uso de las capacidades desarrolladas por la librería.

Cuando las capacidades implementadas tanto por la librería como por la asignatura de la plataforma adquieran entidad suficiente, se dará por concluido el desarrollo del presente proyecto de fin de carrera.

*Se han denominado *pruebas funcionales simuladas* a las pruebas que permiten comprobar la funcionalidad de la librería sin la necesidad del sistema Siette. Esto se ha conseguido simulando el comportamiento del sistema Siette mediante una página web que hace uso de la librería.

†Se han denominado pruebas funcionales definitivas a las pruebas que comprueban la funcionalidad de la librería interactuando en el sistema Siette.

‡En el sistema Siette, los contenidos son desplegados en unidades conceptuales denominadas asignaturas.

1.4. Organización de la memoria

Adicionalmente al contenido de esta memoria, se ha habilitado un espacio web:
`https://sites.google.com/site/aqgpfc/`
Cuya intención es facilitar el acceso a ciertos contenidos de esta memoria así como posibilitar contribuciones futuras en caso de considerarse adecuado.

En cuanto al contenido del documento, esta memoria esta estructurada comenzando con este capítulo introductorio cuyo fin, ya comentado, consiste en proporcionar una visión inicial sobre la motivación, la finalidad y el desarrollo del proyecto. Seguidamente se podrá encontrar un capítulo en el que se profundizará sobre los distintos temas subyacentes que componen el marco de trabajo de este proyecto como son la teoría básica de conjuntos, la labor docente en la Universidad Nacional de Educación a Distancia, la plataforma web Siette, Java y LaTeX.

A continuación, la memoria se centra en el desarrollo del proyecto, proporcionando un capítulo dedicado a la gestión del proyecto y otros relacionados con las distintas fases de desarrollo: especificación de requisitos, análisis del sistema, diseño de la solución, implementación y por último resultados y pruebas.

Finalmente se expondrán en sendos capítulos las conclusiones a las que se ha llegado tras la realización del proyecto, así como las posibles líneas de trabajo futuras para continuar el trabajo realizado.

Además se incluirá en los apéndices el documento inicial de análisis del dominio del problema, así como un manual de usuario del sistema, que abordará la forma correcta de utilizar la librería y otro manual complementario de instalación de la librería, que contendrá las instrucciones técnicas específicas para la compilación y despliegue de la librería. Adicionalmente se incluye otro apéndice dedicado a la descripción del contenido del CD-ROM que acompaña la memoria de este proyecto de fin de carrera.

Capítulo 2

Marco de trabajo

Este capítulo persigue explicar las bases teóricas sobre las que se asienta este proyecto para lograr una comprensión completa del mismo.

2.1. Teoría de conjuntos

Este apartado podría perfectamente ser el último al exponer el marco de trabajo de este proyecto, pero la realidad es que ha constituido una parte excepcionalmente importante dentro del trabajo global, tomando una inusitada relevancia que le hace totalmente merecedor de figurar en primer lugar.

En el momento de elegir la temática sobre la que se realizaría la generación automática de preguntas, se barajaron diversas opciones entre varias asignaturas impartidas en la UNED (Universidad Nacional de Educación a Distancia), se buscaban asignaturas con una marcada orientación matemática, pues siempre son más fáciles de modelar que las de cualquier otra índole, por ello las candidatas más claras eran:

- Lógica y Estructuras Discretas (1^{er} cuatrimestre, 1^{er} curso, 6 créditos)
- Fundamentos Matemáticos (1^{er} cuatrimestre, 1^{er} curso, 6 créditos)
- Estadística (2^{o} cuatrimestre, 1^{er} curso, 6 créditos)

Estas tres son las asignaturas en que se divide la materia "Fundamentos Matemáticos" y que son comunes tanto al *Grado de Ingeniería Informática* como al *Grado de Ingeniería de las Tecnologías de la Información*, formando parte del bloque de formación básica de ambas titulaciones.

De entre ellas se eligió la primera "Lógica y Estructuras Discretas" puesto que facilita ciertos importantes fundamentos formales comunes:

1. Facilita estructuras matemáticas sobre las que modelizar datos (conjuntos, relaciones, funciones, árboles, grafos, etc.)

2. Facilita un lenguaje preciso y universal para especificar restricciones y problemas (preguntas, especificaciones) sobre estos modelos

3. Facilita técnicas de construcción y comprobación de soluciones (mecanismos deductivos, inducción y recursión, verificaciones)

Convirtiéndola en una asignatura de gran importancia, ya que ya desde el primer curso, el estudiante puede apreciar el valor instrumental de esta asignatura tanto para la comprensión de las otras dos de la misma materia, como para la comprensión de otras asignaturas, especialmente:

- Fundamentos de Programación
- Estrategias de Programación
- Estructuras de Datos y Autómatas, Gramáticas y Lenguajes

Asimismo proporciona dos importantes competencias generales:

1. Competencias cognitivas superiores: análisis, síntesis, razonamiento crítico

2. Competencias de expresión y comunicación: las que requieren un lenguaje formal preciso de difusión y discusión de contenidos.

Por todo ello, esta asignatura se planteaba como una buena opción para el desarrollo del proyecto, no obstante la extensión de la asignatura convertía en titánica la labor de abordar

todos sus conceptos, por lo que era importante elegir una parte de la misma para trasladarla al presente trabajo, con ese fin se debía llevar a cabo un estudio de la misma, revisando el mapa conceptual de la asignatura y eligiendo un subconjunto de epígrafes a recubrir, inicialmente se acordó con el tutor atender los apartados:

- 1.1.1 Conceptos básicos de conjuntos
- 1.1.2 Operaciones sobre conjuntos

Posteriormente se ampliaría el número de conceptos abarcados, para añadir mayor completitud al proyecto.

Para contextualizar los conceptos elegidos, se revisaron los exámenes de los últimos años de la asignatura y se realizó un breve documento (también se incluye como apéndice A) recogiendo el resultado del análisis, así como una equivalencia con el texto base [Tremblay and Grassmann, 2010] de la asignatura.

A continuación se incluye un breve repaso a la parte de "Teoría de conjuntos" elegida, comenzando con una breve introducción histórica a la materia recopilada de diversas fuentes en donde ya se encuentra expuesta de manera muy interesante (como por ejemplo en [Sánchez and Arjona, 2002]).

2.1.1. Historia de la teoría de conjuntos

En el último cuarto del siglo XIX se vivió un episodio apasionante de la historia de las matemáticas que las ligaría desde entonces a la historia de la lógica. Primero, Georg Boole (1815-1864) en su *Mathematical Analysis of Logic* trató de presentar la lógica como parte de las matemáticas. Poco después Gottlob Frege (1848-1925) intentó mostrar que la aritmética era parte de la lógica en su *Die Grundlagen der Arithmetik*. Pero, dando un gran paso tanto en la historia de las matemáticas como en la historia de la lógica, G. Cantor se había adelantado a Frege con una fundamentación lógica de la aritmética. Cantor había demostrado que la totalidad de los números naturales comprendidos en el intervalo de extremos 0 y 1 no es numerable, en el sentido de que su infinitud no es la de los números naturales. Como una consecuencia de esa situación, Cantor creó una nueva disciplina matemática entre

1874 y 1897: la teoría de conjuntos. Su obra fue admirada y condenada simultáneamente por sus contemporáneos. Desde entonces los debates en el seno de la teoría de conjuntos han sido siempre apasionados, sin duda por hallarse estrechamente conectados con importantes cuestiones lógicas.

Según la definición de conjunto de Cantor, éste es "una colección en un todo de determinados y distintos objetos de nuestra percepción o nuestro pensamiento, llamados los elementos del conjunto". Frege fue uno de los admiradores de la nueva teoría de Cantor, y dio una definición de conjunto similar.

En 1903 B. Russell demostraría que la teoría de conjuntos de Cantor era inconsistente y cuestionaría la definición de conjunto en la teoría de Cantor. Pero pronto la teoría axiomática de Zermelo (1908) y refinamientos de ésta debidos a Fraenkel (1922), Skolem (1923), von Newman (1925) y otros sentaron las bases para la teoría de conjuntos actual.

Es indiscutible el hecho de que la teoría de conjuntos es una parte de las matemáticas, es además, la teoría matemática dónde fundamentar la aritmética y el resto de teorías matemáticas. Es también indiscutible que es una parte de la lógica y en particular una parte de la lógica de predicados.

En esta historia cruzada de las matemáticas, la lógica y los fundamentos de ambas, la teoría de conjuntos permitiría por un lado una fundación logicista de las matemáticas; pero por otro lado la teoría de conjuntos mirada como parte de las matemáticas proporciona el metalenguaje, el contexto o sustrato de las teorías lógicas. Finalmente, puede ser completamente expresada en un lenguaje de primer orden y sus axiomas y teoremas constituyen una teoría de primer orden a la que pueden aplicarse los resultados generales que se aplican a cualquier teoría de primer orden.

A continuación se presenta primero la teoría intuitiva de conjuntos, basada en la original de Cantor, para seguir con sus problemas de inconsistencia.

2.1.2. Teoría intuitiva de conjuntos

Cantor, Frege y las bases de una teoría intuitiva de conjuntos

La definición inicial de Cantor es totalmente intuitiva: *"un conjunto es cualquier colección* ***C*** *de objetos determinados y bien distintos* ***x*** *de nuestra percepción o nuestro pensamiento (que se denominan elementos de* ***C****), reunidos en un todo"*. Igual que en Frege su idea de lo que es un conjunto coincide con la extensión de un predicado (la colección de objetos que satisface el predicado).

Esta idea sencilla y tan intuitiva resulta ser también ingenua porque produce enormes contradicciones de inmediato, como por ejemplo la *paradoja de Russell* (que se analizará más adelante). Para poder mostrarlo es necesario empezar por formalizar esta teoría intuitiva que, aparte de los símbolos para los conjuntos y sus elementos (x, C, etc.), tendrá los símbolos de **pertenencia** $\in$ e **igualdad** $=$ (de los objetos del lenguaje formal). Que x es un elemento del conjunto C se expresa "*x pertenece a C*" o bien $x \in^* C$. Que x no es un elemento de C se expresa "*x no pertenece a C*" ($x \notin C$).

Se define que dos conjuntos A y B son **iguales** cuando tienen exactamente los mismos elementos, es decir que cada elemento de A es un elemento de B y que cada elemento de B es un elemento de A.

El **conjunto vacío**, a menudo denotado como $\emptyset$ y en otras ocasiones como $\{\}$, es un conjunto sin ningún miembro. Puesto que un conjunto está determinado completamente por sus elementos, solo puede haber un conjunto vacío, pese a que el conjunto vacío no tiene miembros, el puede ser miembro de otros conjuntos, por ello $\emptyset \neq \{\emptyset\}$ ya que en el primer caso no hay miembros y en el segundo existe un miembro.

Se ha de tener en cuenta que no es necesario denotar siempre con mayúsculas a los conjuntos y con minúsculas a sus elementos, ya que un conjunto puede ser a su vez un elemento de otro conjunto e incluso se puede considerar que en esta teoría no hay objetos que no sean conjuntos. Pese a ello para facilitar la comprensión del alumno, la solución se ha diseñado denotando con letras mayúsculas a los nombres de los conjuntos y para los

*El símbolo $\in$ es una derivación de la letra griega epsilon "ε" que fue introducida por Peano en el año 1888.

elementos de los conjuntos se han elegido tres representaciones básicas: números arábigos, letras minúsculas griegas y letras minúsculas latinas.

¿Cómo se determina una colección?

- *Listando sus objetos*. De acuerdo con la definición intuitiva de Cantor, un conjunto queda definido si es posible describir completamente sus elementos. El procedimiento más sencillo de descripción es nombrar cada uno de sus elementos, se llama *definición por extensión*; es ampliamente conocida la notación de encerrar entre llaves los elementos del conjunto.

  Ejemplos: $A = \{a, b, c\}$. Donde A es el conjunto formado por la colección de objetos a, b y c. $B = \{1, 2, 3\}$. Donde B es el conjunto formado por la colección de objetos 1, 2 y 3. Entonces es cierto que $b \in A$ y que $b \notin B$. El inconveniente para este método de listado o enumeración de los elementos del conjunto es que éstos deben poseer un número finito de elementos y, en la práctica, un número muy pequeño.

¿Qué hacer cuando la colección es infinita, o cuando es finita pero numerosa?

- *Describir los objetos*. Cuando el número de elementos del conjunto es infinito (como el de los número impares) o demasiado numeroso (como el de todas las palabras que pueden formarse con el alfabeto latino) se utiliza el método de *definición por intensión*, que consiste en la descripción de un conjunto como la *extensión de un predicado*, esto es, mediante una o varias propiedades (el predicado) que caracterizan a los elementos de ese conjunto. En principio podría tomarse cualquier lengua natural para describir los objetos (español, inglés, italiano, vasco, catalán, etc), aunque es preferible utilizar un lenguaje formal que ofrezca rigor y precisión.

  Ejemplos: $C = \{\text{números naturales pares}\}$. $D = \{\text{libros escritos por Albert Einstein}\}$.

Se defina como se defina el conjunto, habitualmente al número de elementos que contiene se le denomina cardinalidad del conjunto y para un conjunto A se denota de la siguiente manera: $|A|$.

Problemas en la teoría intuitiva de conjuntos: la paradoja de Russell

Pero la definición intuitiva de conjunto como el de una colección de objetos 'describible' por un predicado conduce inevitablemente a ciertas contradicciones que se llaman paradojas, la más célebre es la conocida como paradoja de Russell, que en ocasiones se ha expresado en términos cotidianos siendo el más conocido el de la paradoja del barbero:

> En un lejano poblado de un antiguo emirato había un barbero llamado As-Samet *diestro en afeitar cabezas y barbas, maestro en escamondar pies y en poner sanguijuelas*. Un día el emir se dio cuenta de la falta de barberos en el emirato, y ordenó que los barberos sólo afeitaran a aquellas personas que no pudieran hacerlo por sí mismas. Cierto día el emir llamó a As-Samet para que lo afeitara y él le contó sus angustias:
>
> > "En mi pueblo soy el único barbero. No puedo afeitar al barbero de mi pueblo, ¡que soy yo!, ya que si lo hago, entonces puedo afeitarme por mí mismo, por lo tanto ¡no debería afeitarme! Pero, si por el contrario no me afeito, entonces algún barbero debería afeitarme, ¡pero yo soy el único barbero de allí!"
>
> El emir pensó que sus pensamientos eran tan profundos, que lo premió con la mano de la más virtuosa de sus hijas. Así, el barbero As-Samet vivió para siempre feliz.

Expresado en términos más formales, si se considera el conjunto $A = \{x : x \in x\}$, descrito mediante el predicado del lenguaje formal $x \in x$. Obviamente, para cualquier B, $B \in A$ si y sólo si $B \in B$. Es decir, está en A cuando verifica las condiciones que definen a A. Pero, ¿qué sucede con el propio A? Evidentemente, $A \in A$ si y sólo si $A \notin A$. Pero este resultado es contradictorio. En vano se debe intentar descubrir un error en el razonamiento, más bien parece que el problema proviene de admitir expresiones como $A \in A$ (o conjuntos como el conjunto de todos los conjuntos que produce también paradojas). Se ha visto claramente que el concepto de conjunto no es tan sencillo y que identificarlo sin mayor investigación con el de colección resulta problemático. Para evitar la paradoja de Russell, y otras de esta

naturaleza, es necesario tener más cuidado en la definición de conjunto. Otras paradojas, de hecho las primeras en descubrirse, afectaban a colecciones grandes, como por ejemplo la de los ordinales, o la de todos los conjuntos. Estas colecciones no podrán ser conjuntos.

Solución de las paradojas

Una solución radical al problema de las paradojas es la propuesta en 1903 por Russell, su *Teoría de Tipos*. Russell observa que en todas las paradojas conocidas hay una componente de reflexividad, de circularidad. Técnicamente se evitan las paradojas al eliminar del lenguaje las formaciones circulares. Se reconoce que el universo matemático no es plano, sino jerarquizado, por niveles, y que el lenguaje más adecuado para hablar de este universo debe tener diversos tipos de variables que correspondan a cada nivel; en particular, la relación de pertenencia se da entre objetos de distinto nivel.

En 1908 Zermelo da como solución la definición axiomática de la Teoría de Conjuntos, refinada más tarde por Fraenkel, Skolem, von Neumann y otros. En esta teoría se evita que las colecciones que llevaban a las paradojas puedan ser conjuntos. De hecho, en la solución de Zermelo-Fraenkel, una colección de objetos será un conjunto si los axiomas la respaldan. Dichos axiomas permiten formar conjuntos a partir de conjuntos previamente construidos y postulan la existencia del $\emptyset$ y de al menos un conjunto infinito. Sin embargo, en la solución de von Neumann se admiten colecciones que no son conjuntos, las denominadas clases últimas. Se definen clases mediante propiedades, sin restricción, pero habrá que mostrar que se trata de conjuntos viendo que pertenecen a alguna clase. Las clases últimas, como la clase universal o la de los ordinales, no pertenecen a ninguna otra clase.

No obstante para la realización del presente proyecto de fin de carrera, se pueden plantear ejercicios basados en la noción intuitiva planteada por Cantor, ya que no se llegará al nivel de profundidad matemática que necesita de una teoría axiomática de conjuntos como la de Zermelo-Fraenkel.

2.1.3. Álgebra de conjuntos

Si bien ya se han examinado en apartados anteriores algunos de los conceptos implicados en el álgebra de conjuntos (como son la igualdad, la pertenencia, el conjunto vacío ...) existen algunos conceptos que es importante reseñar para completar el marco teórico que se desarrollará durante la implantación de la solución propuesta en este proyecto.

Subconjuntos

Dados dos conjuntos A y B se dice que A es un **subconjunto** (denotado habitualmente como $A \subseteq B$) de B si cada elemento de A es también un elemento de B. En particular cada conjunto es subconjunto de sí mismo; en el caso concreto de que $B \neq A$ se dice además que B es un **subconjunto propio** (y se denota normalmente con $A \subset B$) de A.

De forma inversa se dice que A es un **superconjunto** de B (denotado como $A \supseteq B$) si B es un subconjunto de A.

Algunos autores utilizan los símbolos "$\subset$" y "$\supset$" para referirse a sub/superconjuntos mientras que otros los utilizan exclusivamente para los sub/superconjuntos propios. A lo largo del proyecto se utilizará esta segunda interpretación*.

Se deduce inmediatamente de la definición de igualdad dada previamente que dados dos conjuntos A y B, A es igual a B si y solo si $A \subseteq B$ y $B \subseteq A$†. Adicionalmente se puede extraer que el conjunto vacío es miembro de cualquier conjunto.

El conjunto de todos los subconjuntos de un conjunto dado A es llamado **conjunto potencia** de A y se denota como 2^{A}‡ o $\mathscr{P}(A)$.

Operaciones con conjuntos

En este punto es interesante introducir los **diagramas de Venn** que son ilustraciones usadas para mostrar gráficamente conjuntos, representando cada conjunto mediante un círculo o un óvalo. La posición relativa en el plano de tales círculos muestra la relación entre los

*Por claridad se podrían utilizar los símbolos "$\subsetneq$" y "$\supsetneq$" para indicar expresamente la desigualdad, pero no se ha considerado necesario.

†De hecho esta es a menudo la definición de igualdad dada: $A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$.

‡Si el conjunto A tiene n elementos, entonces $\mathscr{P}(A)$ tendrá 2^n elementos.

conjuntos. Por ejemplo, si los círculos de dos conjuntos A y B se solapan, se muestra un área común a ambos conjuntos que contiene todos los miembros contenidos a la vez en A y en B. Si el círculo del conjunto A aparece completamente dentro del círculo de otro conjunto B, es que todos los elementos de A también están contenidos en B*.

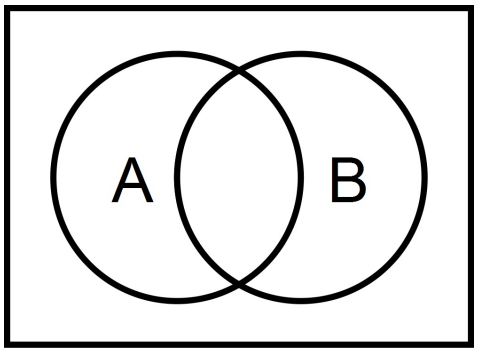


Figura 2.1: Diagrama de Venn de dos conjuntos A y B.

Los diagramas de Venn tienen el nombre de su creador, John Venn, matemático y filósofo británico. Estudiante y más tarde profesor en el Caius College de la Universidad de Cambridge, desarrolló toda su producción intelectual entre esas cuatro paredes.

Venn introdujo el sistema de representación que hoy se conoce en julio de 1880, con la publicación de su trabajo titulado *"De la representación mecánica y diagramática de proposiciones y razonamientos"* en el *Philosophical Magazine and Journal of Science*, provocando un cierto revuelo en el mundo de la lógica formal.

Más adelante desarrolló algo más su nuevo método en su libro *Lógica simbólica*, publicado en 1881 con el ánimo de interpretar y corregir los trabajos de Boole en el campo de la lógica formal. Aunque no tuvo demasiado éxito en su empeño, su libro se convirtió en una excelente plataforma de ejemplo para el nuevo sistema de representación.

La primera referencia escrita al término "diagrama de Venn" de la que se tiene constancia es muy tardía (1918), en el libro *A Survey of Symbolic Logic*, de Clarence Irving Lewis.

Los diagramas de Venn se emplean hoy día para enseñar matemáticas elementales y para reducir la lógica y la Teoría de conjuntos al cálculo simbólico puro.

*Noción de subconjunto expuesta previamente.

En este apartado se utilizarán para apoyar gráficamente la descripción de las diversas operaciones con conjuntos.

Dentro del álgebra de conjuntos, existen tres operaciones principales con conjuntos:

Unión: Dados dos conjuntos A y B, su unión es el conjunto compuesto por todos los miembros que pertenecen a A o pertenecen a B o a ambos simultáneamente. Se denota como $A \cup B$. Su representación gráfica puede verse en la Figura 2.2.

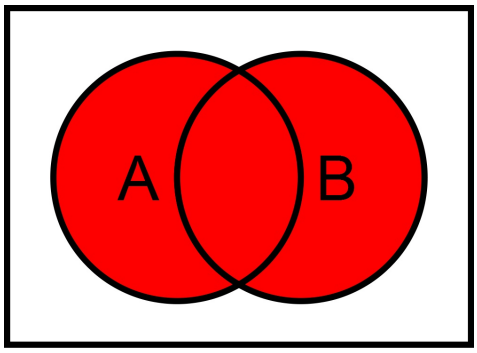


Figura 2.2: Diagrama de Venn representando la unión de dos conjuntos A y B.

Intersección: Dados dos conjuntos A y B, su intersección es el conjunto compuesto por todos los miembros que pertenecen a A y simultáneamente pertenecen a B. Se denota como $A \cap B$. Su representación gráfica puede verse en la Figura 2.3. En el caso especial de que la intersección de dos conjuntos sea el conjunto vacío, es decir, ambos conjuntos no tengan ningún elemento en común. Se dice que ambos conjuntos son **conjuntos disjuntos** (Si $A \cap B = \emptyset$, entonces A y B son disjuntos).

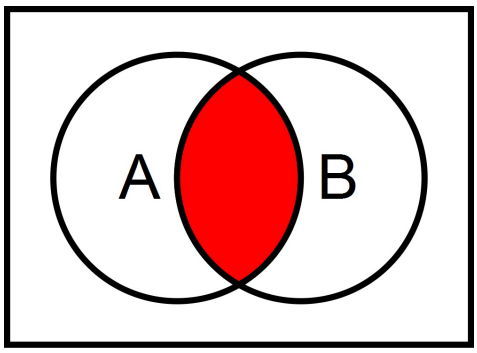


Figura 2.3: Diagrama de Venn representando la intersección de dos conjuntos A y B.

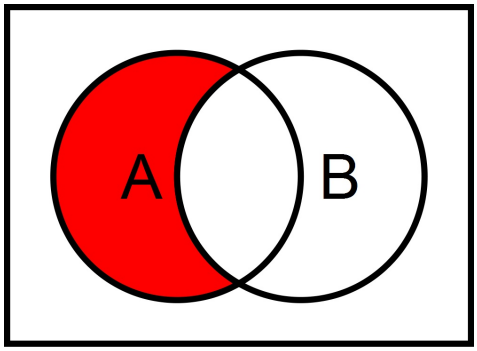


Figura 2.4: Diagrama de Venn representando la diferencia de dos conjuntos A y B.

Diferencia: Dados dos conjuntos A y B, su diferencia* es el conjunto compuesto por todos los miembros de A o de B o de ambos. Se denota como $A \setminus^{\dagger} B$. Su representación gráfica puede verse en la Figura 2.4.

Mención aparte merece la operación denominada "complemento", ya que antes de describirla, es necesario definir un concepto no expuesto aún el de **conjunto universal**‡.

En ciertos contextos se puede considerar que todos los conjuntos considerados son subconjuntos de un conjunto universal dado o de otro modo dado un universo de discurso el conjunto universal es el conjunto que incluye a todos los individuos del universo. Por ejemplo, si se está trabajando con conjuntos de letras del alfabeto latino, se puede considerar al alfabeto latino completo como el conjunto universal.

Una vez introducido el concepto de conjunto universal se puede definir el complemento:

Complemento: Dados dos conjuntos A y U en el que U es el *conjunto universal* y por tanto $A \subseteq U$, el complemento de A es el conjunto compuesto por todos los miembros que no pertenecen a A. Se denota como $\sim A$§. Su representación gráfica puede verse en la Figura 2.5.

*La diferencia puede encontrarse identificada como *"complemento relativo"*.

†Algunos autores denotan la diferencia como "$A - B$", pero no ha sido la notación utilizada en este proyecto.

‡En las teorías de conjuntos estándar no se incluye un verdadero conjunto universal, pero si se incluye en otras teorías de conjuntos no-estándar.

§Algunos autores denotan al complemento de la forma A^C, e incluso se puede ver como A'.

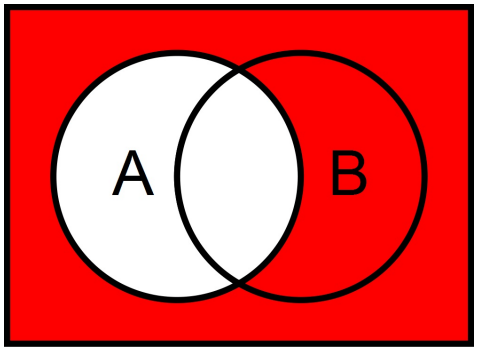


Figura 2.5: Diagrama de Venn representando el complemento de A.

Tuplas

Intuitivamente se pueden definir las tuplas (abreviatura de n-tuplas) como una serie ordenada de objetos colocados en un determinado orden.

La propiedad fundamental de las tuplas es que dos tuplas son iguales si los son sus miembros comparados siguiendo el mismo orden, es decir: si el primer elemento de la primera tupla es igual al primer elemento de la segunda tupla y el segundo elemento de la primera tupla es igual al segundo elemento de la segunda tupla y ... y el último elemento de la primera tupla es igual al último elemento de la segunda tupla.

la tuplas de dos elementos (2-tuplas), denotadas habitualmente en la forma (a,b)*, en las que el primer elemento pertenece a un conjunto A y el segundo elemento pertenece a otro conjunto B (ambos conjuntos pueden ser el mismo) se denomina **producto cartesiano** de A y B, y se escribe como $A \times B$. Se suele denominar al primer elemento de la tupla primera coordenada y al segundo elemento segunda coordenada.

Se podría extender esta definición a conjuntos $A \times B \times C$ de 3-tuplas y de forma más general a conjuntos ordenados de n-tuplas, pero no se han contemplado en el presente proyecto.

Los productos cartesianos fueron desarrollados por primera vez por René Descartes en el contexto de la geometría analítica. Si $\mathbb{R}$ denota el conjunto de todos los números reales, entonces $\mathbb{R}^2 := \mathbb{R} \times \mathbb{R}$ representa el plano euclídeo y $\mathbb{R}^3 := \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ representa el espacio tridimensional de la geometría euclidiana.

*No confundir con la misma notación utilizada para definir intervalos de la recta de números reales.

Conjuntos notables

Hasta ahora se ha supuesto que el universo de discurso era uniforme, en el sentido de que todos los miembros de los conjuntos podrían tratarse de la misma manera, pero esto no es así. Existen muchos tipos diferentes de individuos, tales como personas, cosas e ideas, y las operaciones que tienen sentido para algunos de estos tipos no los tienen para otros. Generalmente un tipo es un conjunto y cualquier conjunto que se utilice de esta manera puede ser considerado un tipo (esto es de especial aplicación en la informática y las matemáticas). En muchos casos de importancia práctica, existe cierto número de tipos iniciales (que se pueden utilizar para crear tipos adicionales).

A continuación se exponen algunos de estos conjuntos:

- El conjunto de los números naturales, que se denota como $\mathbb{N}$
- El conjunto de los números enteros, que se denota como $\mathbb{Z}$
- El conjunto de los números racionales, que se denota como $\mathbb{Q}$
- El conjunto de los números reales, que se denota como $\mathbb{R}$

Existen otros conjuntos importantes que no se han tratado a lo largo del presente proyecto, como el conjunto de los caracteres latinos, el conjunto que contiene verdadero y falso (para definir álgebras booleanas) ...

2.2. La docencia en la UNED

El presente proyecto está enfocado a la generación de actividades para una determinada asignatura de un plan de estudios impartido por la UNED. Para comprender plenamente todas las implicaciones que ello supone, se hace necesario comprender las peculiaridades de esta universidad y de la enseñanza a distancia que la caracteriza.

La UNED es la primera universidad de España por número de estudiantes, por oferta académica, por experiencia y prestigio en enseñanza a distancia, por materiales virtualizados e incluso por cercanía y proximidad.

La extensa red de Centros Asociados, el trabajo de los profesores tutores y de equipos con amplia experiencia docente e investigadora, y las plataformas virtuales y audiovisuales, permiten aprender a distancia, pero con la máxima cercanía y apoyo.

En la UNED se pueden obtener Titulaciones Oficiales con el aval de una universidad pública que ha ganado su prestigio gracias a los cientos de miles de estudiantes que, gracias a su esfuerzo y al apoyo de sus familias, a costa de horas de ocio y descanso, han consolidado la máxima valoración de empresas y los mejores resultados en oposiciones oficiales.

Cerca de 10.000 personas componen el equipo profesionales encargados de desempeñar esa labor.

2.2.1. Breve recorrido por el pasado de la UNED

Al principio de la década de los 70 se decide crear la Universidad Libre a Distancia, en cuyo escudo aparece una paloma. La idea cuaja, pero el nombre y el logotipo no. Por fin, en agosto de 1972, un Decreto Ley da vida a la Universidad Nacional de Educación a Distancia. En un principio son sólo 3 despachos: uno para el Rector, otro para la Secretaría General y otro para el Gabinete de Prensa. Ubicada en el Caserón de San Bernardo, en el distrito centro de Madrid, compartirá espacio con el Consejo Nacional de Educación.

La recién nacida UNED dedica sus primeros años a aumentar el número de alumnos, que va creciendo en progresión geométrica. Es preciso entonces adecuar su estructura a las necesidades docentes. Se crean los dos primeros vicerrectorados, uno de Humanidades y otro de Ciencias, y se abre una oficina de atención al público. Las unidades didácticas se envían a los estudiantes a sus casas, por correo postal, totalmente gratis.

El siguiente paso es llevar la educación superior a los núcleos de población, alejados de las grandes metrópolis, que no disponen de universidad. La creación de centros regionales servirá para asentar la UNED y su peculiar método docente en toda la península y en las islas. En estos centros los tutores actuarán como guías y asesores de los alumnos.

Llega el momento de mirar más allá y se trabaja en la estructura internacional. América Latina es el siguiente objetivo. La UNED se implanta en algunos países y su modelo metodológico se “exporta”: se convierte en líder de la AIESAD (Asociación Iberoamericana de

Educación Superior a Distancia).

Garantizar la igualdad de oportunidades es uno de los objetivos explícitos de esta universidad. La UNED ha posibilitado el acceso a los estudios superiores a personas que no habrían podido conseguirlo por razones de renta, por su lugar de residencia o por cualquier otra dificultad.

La UNED ha potenciado muy especialmente la incorporación de la mujer a la universidad y al mercado de trabajo. No es casualidad que en 1982 fuera elegida Rectora de la UNED Elisa Pérez Vera, la primera mujer que llegaba a tal cargo en la universidad española.

El otro gran objetivo también se va cumpliendo: sus listados de matrículas se nutren de personas que alternan su jornada laboral con su formación universitaria. Sus programas se convierten en la segunda oportunidad para muchos ciudadanos ávidos de saber que, en su momento, por distintas razones, no accedieron a la enseñanza superior convencional.

En una década la UNED ha recorrido un largo camino en su implantación social, que continuará a lo largo del los años 80: centros asociados en casi todas las provincias; más centros en el extranjero y una permanente ampliación de su oferta educativa que, curso tras curso, va incorporando nuevas titulaciones y nuevos programas de formación continua.

Comienza entonces la apuesta por la difusión. Al uso de la radio y la televisión como sistemas de emisión de contenidos se añaden, en los 90, las nuevas tecnologías. La incorporación de sistemas multimedia, tanto en la elaboración de materiales como en su distribución, se hace extensiva a todas las disciplinas.

Los sistemas digitales e Internet han hecho posible que, en la actualidad, la “distancia” entre la UNED y sus estudiantes haya desaparecido: cada alumno tiene toda la universidad en su mesa de estudio, a sólo un “cliq” del teclado de su ordenador.

Hoy, la UNED es una gran institución: la mayor universidad de España con sus más de 260.000 estudiantes; con una oferta educativa que abarca 26 títulos de Grado, 43 másteres, más de 600 programas de Formación Continua, 12 cursos de idiomas, más de un centenar de Cursos de Verano y casi 400 actividades de Extensión Universitaria. Más de 10.000 personas, desde la sede central y desde los centros asociados, se esfuerzan por apoyar día a día la dura marcha de los estudiantes hacia la meta de su formación.

2.2.2. Características específicas de la enseñanza a distancia

La educación a distancia nace de la necesidad de una sociedad que sufre un gran cambio: la industrialización. Se requiere una mano de obra cualificada, con un nivel cultural y una preparación técnica cada vez mayor.

El concepto de aula y de profesor del que dependen directamente los alumnos se vuelve en ocasiones insuficiente a la hora de cubrir las peculiares circunstancias de una sociedad en la que los alumnos no siempre disponen de tiempo o de medios para realizar un seguimiento presencial y constante.

En este contexto, un sistema a distancia empieza a tomar forma. Por un lado tiene la desventaja de la impersonalidad, ya que el seguimiento presencial de las lecciones y el contacto directo del profesor es una ayuda incuestionable con la que no se cuenta, pero por otra parte, este hándicap se suple con constancia y voluntad.

La educación presencial se sustituye inicialmente por una educación diferida, en la que el alumno tiene los textos y dispone de tutorías programadas de forma periódica para solucionar dudas, así como la posibilidad de comunicarse con el tutor por correo o telefónicamente en unos horarios determinados. Esta es la forma quizá más costosa para el alumno en cuanto a planificación y constancia: las dudas deben esperar para ser resueltas y el contacto no es inmediato, ya que los medios con los que se cuenta aún no permiten mejorar este aspecto. Posteriormente, con el auge de las nuevas tecnologías, se mejora en cuanto a cercanía con el profesor y en cuanto a inmediatez a la hora de solucionar problemas, conocer los resultados del trabajo realizado o incluso auto-evaluarse, empleando herramientas como la que se ha desarrollado en el presente proyecto.

Según Peters (1983) la educación a distancia es una forma industrial de enseñar y aprender. Se considera al método tradicional, en el que los alumnos dependen directamente del profesor y de lo que les va enseñando puntualmente, como algo artesanal. El avance en el estudio depende solo del profesor, y es él exclusivamente quien determina el ritmo de aprendizaje de toda la clase.

El cambio que se produce en la educación tiene también que ver con la economía: se piensa en la eficiencia de los docentes intentando optimizarla, se produce una división del

trabajo, hay una especialización en cuanto a las tareas – tanto lectivas como administrativas – y se controla y analiza el sistema educativo para optimizarlo.

El sistema que adoptan universidades como la UNED, se basa en esto precisamente, una gran especialización de profesores, un gran equipo administrativo que tramita todos los pormenores que no tengan que ver exclusivamente con la docencia y la gran importancia de unos canales de comunicación con los alumnos que han convertido la enseñanza diferida en un contacto constante y un feedback incluso inmediato que sirve de gran ayuda en el aprendizaje, logrando que la distancia física sea una barrera cada vez menos infranqueable y un aprendizaje independiente y flexible. Las plataformas de aprendizaje virtuales (e-learning) han sustituido a las aulas físicas como contexto fundamental de aprendizaje. El profesor no condiciona el avance de los alumnos, sino que es una figura que está centralizando el proceso de enseñanza.

Los planteamientos son los siguientes:

- Fomento de la relación personal profesores - alumnos. Ello reforzará su motivación hacia el estudio.
- Dicho sentimiento puede promoverse mediante un adecuado material autoinstructivo y un diligente sistema de intercomunicación a distancia que proporcione al alumno las respuestas que necesite.
- El clima, las convenciones y el lenguaje establecidos en la comunicación entre alumno y docente también favorecen la percepción de que existe una relación personal alumno-profesor.
- Los objetivos de aprendizaje se consiguen mejor aumentando la motivación hacia el estudio, lo que supone percibir éste como placer intelectual.
- Los mensajes dados y recibidos en forma de diálogo se recuerdan con mayor facilidad.
- Cuanto la conversación didáctica guiada es más evidente, más vinculados personalmente se sienten los estudiantes con el docente y con el sistema de educación a distancia.

- Cuanto mayor interés perciben los estudiantes por parte del docente y del sistema de educación a distancia en que el estudio resulte relevante para ellos, mayor es su participación en el proceso.

- Cuanto más sienten los estudiantes que están participando e implicándose en su propio proceso de aprendizaje en el seno del sistema de educación a distancia, mayor es su motivación y más efectivo su aprendizaje.

Las tutorías individuales se pueden realizar bien en tiempo real – presencialmente, por teléfono o de manera telemática – o diferido – por correo ordinario o e-mail-. También existen las tutorías colectivas, que cumplen una función socializadora, haciendo al alumno entrar en contacto con otros estudiantes que se enfrentan a los mismos retos y dificultades y en los cuales poder apoyarse para superar satisfactoriamente los estudios.

Analizando las diferencias entre la enseñanza presencial y la enseñanza a distancia, se pueden clasificar en tres grupos dependiendo del punto de vista: punto de vista técnico, punto de vista del profesor y punto de vista del alumno.

Punto de vista técnico

Si se considera el aspecto económico, se puede apreciar que la enseñanza presencial tiene unos costes directamente relacionados con el número de alumnos y por tanto de profesores, y que en general se requiere una inversión inicial menor y muy directamente relacionada con los costes de personal. En cambio la enseñanza a distancia necesita una inversión inicial muy notable para cubrir la puesta en marcha tecnológica, pero el coste anual por alumno desciende notablemente al incrementarse el número de alumnos, ya que los costes son fijos, por lo que en una entidad de la magnitud y el número de alumnos de una universidad como la UNED, la inversión en mejoras tecnológicas resulta rentable.

En cuanto a las características organizativas cabe señalar que la educación presencial puede ser planificada de forma más simple, más a corto plazo y en función del personal docente, mientras que la educación a distancia esta planificación es mucho más compleja, ya que debe ser realizada antes del comienzo del curso (o incluso con una previsión plurianual) y cuyo pilar principal es la gran división y especialización de las tareas entre el profesorado y

el personal de administración y servicios. Así, con una gran especialización se consigue que el índice entre profesores y alumnos sea muy alto, mientras que en la enseñanza presencial el espacio físico y la atención personal limitan este aspecto.

Punto de vista del profesor

En un sistema presencial, el profesor cuenta con las bazas a su favor de la socialización, el aprendizaje cooperativo, la capacidad del refuerzo inmediato y la voz como recurso para explicar la información que precisa, mientras que un sistema diferido es más complicado para estimular el aprendizaje individualizado del alumno, ya que el profesor es más bien un guía y un apoyo puntual y específico para el estudiante, que cuenta con los textos lectivos desde el principio.

Por otro lado, en relación a la eficiencia, se observa, que mientras que la educación presencial se dirige a personas de características homogéneas en cuanto a edad, ocupación e incluso lugar de residencia, la enseñanza a distancia cubre las necesidades de alumnos muy diferentes entre sí, muy motivados, con mayor fuerza de voluntad y en general mayor nivel formativo, que tienen mayor familiaridad en términos generales con las tecnologías y que en ocasiones no tiene la disponibilidad de tiempo o carecen de medios para acceder a una enseñanza tradicional. Estas características del alumnado facilitan la enseñanza al profesorado aumentando la eficiencia del proceso educativo en relación a los recursos docentes empleados.

En cualquier caso, no debe menospreciarse el esfuerzo que han de realizar los docentes, que deben de estar preparados para trabajar con estas nuevas tecnologías, suponiendo este hecho un esfuerzo adicional y una preparación extra que no existiría en la docencia presencial.

Punto de vista del alumno

Los alumnos que presentan un perfil adecuado a rendir en una enseñanza a distancia son aquellos con gran capacidad de trabajo autónomo, planificación y fuerza de voluntad que no pueden supeditar otras ocupaciones al rol de estudiante, en tanto que los alumnos que

prefieren la enseñanza presencial tienden a aprovechar el aprendizaje cooperativo, depender del seguimiento y la planificación del profesor y necesitan un ámbito de socialización más definido.

2.3. Siette

Siette es un sistema web que permite la creación y mantenimiento de bancos de preguntas, la realización de tests, que implementa la Teoría Clásica de Test (CTT), la Teoría de Respuesta al Ítem (TRI), permite realizar Tests Adaptativos Informatizados (TAI) y puede usarse como herramienta para el Aprendizaje colaborativo. Además puede usarse como módulo de evaluación de un Sistema Tutor Inteligente (STI) o conectado a un Sistema Gestor de Contenidos Educativos (LMS) como Moodle.

Siette ha sido desarrollado para ayudar a los profesores en el desarrollo de tests y/o material de evaluación. Siette no sólo permite la creación de pruebas, sino también el seguimiento de los resultados obtenidos por los estudiantes. Por otra parte, los tests realizados en Siette pueden ser útiles para los estudiantes como preparación a un examen. En conclusión, Siette podría ser utilizado por cualquier institución educativa que aplique tecnologías informáticas en sus métodos de enseñanza.

Lo que diferencia a Siette de otras herramientas similares es la amplia gama de posibilidades que ofrece. Algunas de las características más relevantes son las siguientes:

- Gran variedad de preguntas que se pueden crear.
- Creación de preguntas generativas mediante JSP.
- Soporte para la definición de nuevos tipos de preguntas mediante applets.
- Test Adaptativos Informatizados según el nivel de conocimiento del alumno.
- Test de autoaprendizaje basado en el método de repaso espaciado.
- Modificación de preguntas a posteriori y revisión automática de la evaluación.

- Capacidad para reconocer respuestas cortas escritas por el estudiante y su evaluación de forma automática.
- Método de evaluación configurable: por porcentajes, por puntos y por TRI.
- Herramientas de análisis estadístico de datos basadas en la Teoría Clásica de Test y la Teoría de Respuesta al Ítem.
- Posibilidad de incluir sugerencias y comentarios en las opciones de respuesta que podrán mostrarse a los estudiantes durante o después de realizar el examen con el fin de ayudarles a comprender sus errores y mejorar su aprendizaje.
- Posibilidad de realizar un examen de forma colaborativa, entre varios estudiantes, pudiendo trabajar desde ubicaciones diferentes. Esto ayudará a mejorar los resultados de todos.

En Siette existen tres tipos de usuarios claramente diferenciados:

Administrador: Tiene asignadas acciones de gestión y mantenimiento del sistema. Además de sus privilegios, tiene los que poseen los profesores y los alumnos.

Profesor: Que podrá definir asignaturas, temas, preguntas, tests, activarlos y administrarlos. Además de sus privilegios, también tiene los que posee un alumno.

Alumno: Que podrá realizar tests.

2.3.1. Contenidos en Siette

Los contenidos en Siette se estructuran en asignaturas, cada asignatura viene representada por su árbol curricular de temas y cada tema tiene asignado un conjunto de ítems que permiten evaluarlo. Los ítems pueden ser de distintos tipos:

Ítems básicos: Se llaman ítems básicos porque cualquier tipo de ítem que se puede definir en Siette se corresponde con uno de estos. Hay tres tipos de ítem básico:

Ítems de múltiple opción y respuesta simple: Son aquellos ítems en los que los alumnos tienen que seleccionar una sola de entre un conjunto de posibles respuestas, pudiendo dejar la pregunta en blanco, es decir, sin señalar ninguna respuesta.

Ítems de múltiple opción y respuesta múltiple: Son similares en formato a los anteriores, pero en este caso los alumnos pueden seleccionar más de una respuesta de entre el conjunto de alternativas.

Ítems de respuesta corta: Son aquellos ítems en los que los alumnos tienen que dar una respuesta escrita, relativamente corta, dado cierto enunciado. A excepción del enunciado no se le proporciona al alumno ninguna otra información sobre la respuesta.

Ítems generativos: Los Ítems generativos son ítems que se implementan mediante lenguajes embebidos en HTML como JSP. Se tratan de plantillas con parámetros que se instanciarán a la hora de ser mostrados.

Ítems externos: Los Ítems externos son ítems que están ubicados fuera del servidor en el que se encuentra el sistema, y por lo tanto, su presentación no está controlada directamente por Siette, sino que una aplicación externa se encarga de realizarla.

2.3.2. ¿Por qué Siette?

Ya se ha visto que es una herramienta muy versátil y completa, pero conviene especificar los motivos que contribuyen a su elección para el presente proyecto:

1. Mediante la facilidad de generar diferentes preguntas a través de las posibilidades de los ítems generativos, se pueden conseguir pruebas de evaluación muy diversas para que el alumnado disponga de un amplio rango de cuestiones mediante las cuales mejorar y evaluar su conocimiento.

2. Siette aumenta enormemente las posibilidades de disponibilidad del contenido al tratarse de tecnología web que puede ser accedida en cualquier momento desde cualquier parte (con acceso a tecnología web online).

3. Presenta herramientas de evaluación de resultados que aportan un gran valor añadido a la labor docente, que permite identificar puntos débiles comunes en el conocimiento del alumnado lo que proporciona la posibilidad de iniciar mecanismos para corregir esas deficiencias.

4. Permite la posibilidad de mostrar ecuaciones matemáticas con gran potencia gracias a la utilización del motor de renderizado para matemáticas Mathjax* con soporte para notación TEX.

5. La incorporación de las tecnologías de Tests Adaptativos Informatizados y Tests de Auto-aprendizaje basados en el método de repaso espaciado, dotan al sistema de un comportamiento dinámico y adaptable a los conocimientos y necesidades específicos de cada alumno.

6. Gracias a la utilización de refuerzos y ayudas se complementa la experiencia de usuario, favoreciendo tanto la evaluación como el aprendizaje.

2.4. Java

Java es un lenguaje de programación originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en el 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. El lenguaje deriva mucho de su sintaxis de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede correr en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora. Java es un lenguaje de programación de propósito general, concurrente, basado en clases, y orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere

*Motor de código libre basado en JavaScript.

decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, actualmente, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, y librerías de clases en 1991 y las publicó por primera vez en el 1995. A partir de mayo del 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar el paradigma de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (Common Object Request Broker Architecture), Internet Communications Engine o OSGi (Open Services Gateway Initiative).

2.4.1. Historia

Java se creó como una herramienta de programación para ser usada en un proyecto de set-top-box en una pequeña operación denominada the Green Project en Sun Microsystems en el año 1991. El equipo (Green Team), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road en Menlo Park en su desarrollo.

El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a denominarse Green tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas y finalmente se renombró a Java.

Es frecuentada por algunos de los miembros del equipo. Pero no está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iniciales de sus creadores: James Gosling, Arthur Van Hoff, y Andy Bechtolsheim. Otros abogan por el siguiente acrónimo, Just Another Vague Acronym ("sólo otro acrónimo ambiguo más"). La hipótesis que más fuerza tiene es la que Java debe su nombre a un tipo de café disponible en la cafetería cercana, de ahí que el icono de java sea una taza de café caliente. Un pequeño signo que da fuerza a esta teoría es que los 4 primeros bytes (el número mágico) de los archivos.class que genera el compilador, son en hexadecimal, 0xCAFEBABE. A pesar de todas estas teorías, el nombre fue sacado al parecer de una lista aleatoria de palabras.

Los objetivos de Gosling eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++. Entre junio y julio de 1994, tras una sesión maratoniana de tres días entre John Gaga, James Gosling, Patrick Naughton, Wayne Rosing y Eric Schmidt, el equipo reorientó la plataforma hacia la Web. Sintieron que la llegada del navegador web Mosaic, propiciaría que Internet se convirtiese en un medio interactivo, como el que pensaban era la televisión por cable. Naughton creó entonces un prototipo de navegador, WebRunner, que más tarde sería conocido como HotJava.

En 1994, se les hizo una demostración de HotJava y la plataforma Java a los ejecutivos de Sun. Java 1.0a pudo descargarse por primera vez en 1994, pero hubo que esperar al 23 de mayo de 1995, durante las conferencias de SunWorld, a que vieran la luz pública Java y HotJava, el navegador Web. El acontecimiento fue anunciado por John Gage, el Director Científico de Sun Microsystems. El acto estuvo acompañado por una pequeña sorpresa adicional, el anuncio por parte de Marc Andreessen, Vicepresidente Ejecutivo de Netscape, de que Java sería soportado en sus navegadores. El 9 de enero del año siguiente, 1996, Sun fundó el grupo empresarial JavaSoft para que se encargase del desarrollo tecnológico. Dos semanas más tarde la primera versión de Java fue publicada.

La promesa inicial de Gosling era Write Once, Run Anywhere (Escríbelo una vez, eje-

cútalo en cualquier lugar), proporcionando un lenguaje independiente de la plataforma y un entorno de ejecución (la JVM) ligero y gratuito para las plataformas más populares de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma.

El entorno de ejecución era relativamente seguro y los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustadas en las páginas web.

Java ha experimentado numerosos cambios desde la versión primigenia, JDK 1.0, así como un enorme incremento en el número de clases y paquetes que componen la biblioteca estándar.

Desde J2SE 1.4, la evolución del lenguaje ha sido regulada por el JCP (Java Community Process), que usa Java Specification Requests (JSRs) para proponer y especificar cambios en la plataforma Java. El lenguaje en sí mismo está especificado en la Java Language Specification (JLS), o Especificación del Lenguaje Java.

2.4.2. ¿Por qué Java?

En primer lugar por que es uno de los requisitos especificados para el proyecto, pero más allá de esa circunstancia existen otros motivos para hacer de Java el lenguaje idóneo para este desarrollo:

1. Las aplicaciones codificadas mediante Java pueden ser accedidas desde la plataforma web Siette mediante los items generativos basados en JSP.

2. Por lo extendido del lenguaje que lo ha convertido en uno de los lenguajes más utilizados para desarrollar aplicaciones web.

3. Debido a la extensión comentada previamente, ha sido uno de los lenguajes más utilizados en la Ingeniería Informática que precede a este proyecto de fin de carrera.

4. Su filosofía de ejecución multiplataforma a través de máquinas virtuales específicas para cada entorno, permiten grandes posibilidades de expansión de los desarrollos sin necesidad de esfuerzo adicional.

2.5. LaTeX

LaTeX es un sistema de composición de textos que está formado mayoritariamente por órdenes construidas a partir de comandos de TeX -un lenguaje "de bajo nivel", en el sentido de que sus acciones últimas son muy elementales- pero con la ventaja añadida de "poder aumentar las capacidades de LaTeX utilizando comandos propios del TeX descritos en The TeXbook". Esto es lo que convierte a LaTeX en una herramienta práctica y útil pues, a su facilidad de uso, se une toda la potencia de TeX. Estas características hicieron que LaTeX se extendiese rápidamente entre un amplio sector científico y técnico, hasta el punto de convertirse en uso obligado en comunicaciones y congresos, y requerido por determinadas revistas a la hora de entregar artículos académicos.

Su código abierto permitió que muchos usuarios realizasen nuevas utilidades que extendiesen sus capacidades con objetivos muy variados, a veces ajenos a la intención con la que fue creado: aparecieron diferentes dialectos de LaTeX que, a veces, eran incompatibles entre sí. Para atajar este problema, en 1989 Lamport y otros desarrolladores iniciaron el llamado "Proyecto LaTeX3". En 1993 se anunció una re-estandarización completa de LaTeX, mediante una nueva versión que incluía la mayor parte de estas extensiones adicionales (como la opción para escribir transparencias o la simbología de la American Mathematical Society) con el objetivo de dar uniformidad al conjunto y evitar la fragmentación entre versiones incompatibles de LaTeX 2.09. Esta tarea la realizaron Frank Mittlebach, Johannes Braams, Chris Rowley y Sebastian Rahtz junto al propio Leslie Lamport. Hasta alcanzar el objetivo final del "Proyecto 3", a las distintas versiones se las viene denominando LaTeX2e (o sea, "versión 2 y un poco más…"). Actualmente cada año se ofrece una nueva versión, aunque las diferencias entre una y otra suelen ser muy pequeñas y siempre bien documentadas.

Con todo, además de todas las nuevas extensiones, la característica más relevante de este esfuerzo de re-estandarización fue la arquitectura modular: se estableció un núcleo central (el compilador) que mantiene las funcionalidades de la versión anterior pero permite incrementar su potencia y versatilidad por medio de diferentes paquetes que solo se cargan si son necesarios. De ese modo, LaTeX dispone ahora de innumerables paquetes para todo tipo de objetivos, muchos dentro de la distribución oficial, y otros realizados por terceros, en algunos

casos para usos especializados.

2.5.1. Uso

LaTeX presupone una filosofía de trabajo diferente a la de los procesadores de texto habituales (conocidos como WYSIWYG, es decir, "lo que ves es lo que obtienes") y se basa en comandos. Tradicionalmente, este aspecto se ha considerado una desventaja (probablemente la única). Sin embargo, LaTeX, a diferencia de los procesadores de texto de tipo WYSIWYG, permite a quien escribe un documento centrarse exclusivamente en el contenido, sin tener que preocuparse de los detalles del formato. Además de sus capacidades gráficas para representar ecuaciones, fórmulas complicadas, notación científica e incluso musical, permite estructurar fácilmente el documento (con capítulos, secciones, notas, bibliografía, índices analíticos, etc.), lo cual brinda comodidad y lo hace útil para artículos académicos y libros técnicos.

Con LaTeX, la elaboración del documento requiere normalmente de dos etapas: en la primera hay que crear mediante cualquier editor de texto llano un fichero fuente que, con las órdenes y comandos adecuados, contenga el texto que queramos imprimir. La segunda consiste en procesar este fichero; el procesador de textos interpreta las órdenes escritas en él y compila el documento, dejándolo preparado para que pueda ser enviado a la salida correspondiente, ya sea la pantalla o la impresora. Ahora bien, si se quiere añadir o cambiar algo en el documento, se deberá hacer los cambios en el fichero fuente y procesarlo de nuevo. Esta idea, que puede parecer poco práctica a priori, es conocida a los que están familiarizados con el proceso de compilación que se realiza con los lenguajes de programación de alto nivel (C, C++, etc.), ya que es completamente análogo.

El modo en que LaTeX interpreta la "forma" que debe tener el documento es mediante etiquetas. Por ejemplo, \documentclass{article} le dice a LaTeX que el documento que va a procesar es un artículo. Puede resultar extraño que hoy en día se siga usando algo que no es WYSIWYG, pero las características de LaTeX siguen siendo muchas y muy variadas. También hay varias herramientas (aplicaciones) que ayudan a una persona a escribir estos documentos de una manera más visual (LyX, TeXmacs y otros). A estas herramientas se les llama WYSIWYM ("lo que ves es lo que quieres decir").

Una de las ventajas de LaTeX es que la salida que ofrece es siempre la misma, con independencia del dispositivo (impresora, pantalla, etc.) o el sistema operativo (MS Windows, MacOS, Unix, GNU/Linux, etc.) y puede ser exportado a partir de una misma fuente a numerosos formatos tales como Postscript, PDF, SGML, HTML, RTF, etc. Existen distribuciones e IDEs de LaTeX para todos los sistemas operativos más extendidos, que incluyen todo lo necesario para trabajar. Hay, por ejemplo, programas para Windows como TeXnicCenter, para Linux como Kile, o para MacOS como TeXShop, todos liberados bajo la Licencia GPL. Existe además un editor multiplataforma (para MacOS, Windows y Unix) llamado Texmaker, que también tiene licencia GPL.

2.5.2. ¿Por qué LaTeX?

Dos hechos propiciaban desde el principio la elección de esta tecnología:

- La recomendación del equipo docente de utilizar esta tecnología para la redacción de la memoria.
- La facilidad para mostrar fórmulas matemáticas de esta solución.

No obstante no han sido sólo estas circunstancias las que han determinado finalmente la adopción de esta tecnología, sino en especial la posibilidad de utilizar el mismo lenguaje de representación para un formato imprimible y para otro web. Esto es posible gracias a que la plataforma Siette permite utilizar el motor de representación MathJax capaz de representar gráficamente fórmulas expresadas en LaTeX.

Adicionalmente hay que resaltar que es una herramienta de libre distribución, ampliamente difundida para fines científicos y por las facilidades que presenta en cuanto a gestión de la bibliografía, generación de listados e índices, etc...

Capítulo 3

Gestión de proyecto

En este capítulo se recoge toda la información necesaria para la gestión y el control de este proyecto en términos de actividades, recursos e hitos a alcanzar durante el desarrollo del mismo. Además, este documento también ha servido para comprobar si se cumplían los plazos estimados para el desarrollo del proyecto.

El contenido específico de este capítulo se ha separado en tres partes, la primera es un estudio de viabilidad del proyecto destinado principalmente a realizar una estimación inicial del coste de desarrollo del proyecto, la segunda es una parte dedicada a la planificación inicial del proyecto (haciendo especial hincapié en la gestión de riesgos) y la tercera y última está destinada al seguimiento del proyecto.

En la primera parte se ha utilizado un modelo matemático paramétrico para realizar la estimación inicial del coste de desarrollo del proyecto.

Dentro de la segunda parte se han expuesto las actividades a realizar, los recursos disponibles, la planificación temporal general y la descripción de los riesgos que pueden darse en el proyecto.

Y por último, en la tercera parte, se han mostrado los tiempos reales de ejecución y el análisis final de ocurrencia de los riesgos y su efecto.

3.1. Estudio de viabilidad

Normalmente este apartado de un proyecto se dedica a establecer la viabilidad de la ejecución del proyecto generalmente siguiendo una aproximación de análisis del coste frente al beneficio obtenido, pero en este caso concreto al tratarse de un proyecto de fin de carrera cuya finalidad es la obtención de un título académico y cuya realización es condición indispensable para la consecución del objetivo académico no tiene demasiado sentido un análisis de ese tipo ya que no existen alternativas a la realización del proyecto que permitan conseguir el mismo objetivo.

No obstante, una parte importante de los estudios de viabilidad es el estudio inicial de costes que permite valorar diferentes soluciones y adicionalmente planificar la posterior ejecución de la solución elegida.

3.1.1. Estimación de costes

La estimación de cualquier desarrollo es de una importancia estratégica en cualquier proyecto. Esta estimación, que se suele realizar en las primeras fases del ciclo de vida y que comprende la evaluación del esfuerzo, la duración, y los recursos necesarios para realizar cada una de las tareas en las que se descompone un desarrollo complejo, cuenta con una cierta incertidumbre por lo que adicionalmente se debe realizar un análisis de riesgos que permita conocer con antelación los posibles errores que se cometan en la estimación.

La estimación sirve para:

- Conocer el coste aproximado del proyecto en tiempo y recursos.
- Conocer en cada momento el estado de evolución del proyecto.

Para realizar una estimación adecuada se pueden utilizar métodos de distinta índole: modelos matemáticos paramétricos, estimación basada en la experiencia de expertos, técnicas orientadas al aprendizaje o modelos dinámicos.

En este proyecto se ha estimado el esfuerzo a través de un modelo paramétrico no lineal que fue publicado por primera vez por Boehm en 1995. Se trata de COCOMO II, cuyo

nombre procede de COnstructive COst MOdel y que está compuesto por tres submodelos: Composición de aplicaciones, Diseño inicial y Post-arquitectura.

En este caso se ha utilizado únicamente el submodelo de Post-arquitectura, ya que se ha realizado la estimación tras finalizar un primer prototipo y tras contar con un diseño de alto nivel del sistema. La estimación se puede realizar en diferentes momentos del ciclo de vida del software: cuando se analiza la oferta, antes de la especificación de requisitos, tras la especificación de requisitos, etc ... pero se ha elegido este momento concreto por contar con importante información que permite realizar una adecuada estimación del esfuerzo.

No se va a describir en profundidad el submodelo utilizado, para ello se puede consultar [Agustín et al., 2002], y se ha expuesto directamente el resultado de la estimación:

Valores de los factores de escala:

- Precedentes: Alto, valor numérico 2,48
- Flexibilidad de desarrollo: Nominal, valor numérico 3,04
- Arquitectura/Solución de riesgo: Nominal, valor numérico 4,24
- Cohesión del equipo/interacción: Nominal, valor numérico 3,29
- Madurez del proceso: Nominal, valor numérico 4,68

Multiplicadores de esfuerzo:

- RELY: Bajo, valor numérico 0,92
- DATA: Bajo, valor numérico 0,90
- CPLX: Alto, valor numérico 1,17
- RUSE: Nominal, valor numérico 1,00
- DOCU: Alto, valor numérico 1,11
- TIME: Nominal, valor numérico 1,00
- STOR: Nominal, valor numérico 1,00

- PVOL: Nominal, valor numérico 1,00
- ACAP: Nominal, valor numérico 1,00
- AEXP: Bajo, valor numérico 1,10
- PCAP: Nominal, valor numérico 1,00
- PEXP: Bajo, valor numérico 1,09
- LTEX: Alto, valor numérico 0,91
- PCON: Muy Bajo, valor numérico 1,29
- TOOL: Bajo, valor numérico 1,09
- SITE: Nominal, valor numérico 1,00
- SCED: Nominal, valor numérico 1,00

Cálculo del esfuerzo en MM (man-month, hombres-mes):

$Exponente = 0{,}91 + 0{,}01 * (2{,}48 + 3{,}04 + 4{,}24 + 3{,}29 + 4{,}68) = 1{,}087$

$Esfuerzo = e = 2{,}94 * (4{,}2)^{1}{,}087 * (0{,}92 * 0{,}9 * 1{,}17 * 1{,}11 * 1{,}10 * 1{,}09 * 0{,}91 * 1{,}29 * 1{,}09) =$ 21,30 *MM*

Cálculo del tiempo de desarrollo:

$Tiempodedesarrolloenmeses = t = 3 * 21{,}30^{0{,}33+0{,}2*(0{,}91-1{,}01)} * (100/100) = 7{,}74$ *meses*

$Tiempodedesarrolloenhoras = t * 20 * 8 = 1270$ *horas*

Por lo tanto, la estimación inicial del tiempo de desarrollo expresada en horas* es de 1270. Esta es una cifra bastante alta, pero eso es debido a que los métodos de estimación son muy sensibles a la forma en que se miden los proyectos y por tanto no se los debe de considerar un método exacto de estimación de tiempo sino más bien guías que sirvan

*La estimación se ha calculado en horas debido a que este proyecto no se ha abordado de forma empresarial (con dedicación exclusiva) sino que el esfuerzo de desarrollo se ha combinado con otros compromisos: académicos, profesionales, personales, familiares, etc... que han obligado a la distribución irregular de las jornadas de trabajo.

para comparar distintos proyectos utilizando exactamente las mismas métricas y los mismos métodos de estimación.

No obstante, esta estimación se realizó cuando ya se contaba con los datos del desarrollo del prototipo aritmético, de hecho el tamaño final en KLOC de la librería se aproximó teniendo en cuenta que el prototipo inicial tenía 0,28 KLOC y que como se debían implementar unas 15 preguntas distintas (finalmente fueron 14 con unas 33 variantes en total) una buena aproximación podría ser multiplicar el tamaños del prototipo por 15, de ahí salieron los 4,2 KLOC de tamaño final de la librería.

El hecho de realizar esta estimación tras haber desarrollado el prototipo inicial también sirve para estimar el desfase entre el método de estimación y los datos reales, aplicando el mismo modelo de estimación al prototipo se obtiene un tiempo de desarrollo de 522 horas, puesto que el tiempo de desarrollo real fue de alrededor de 80 horas, se puede extrapolar que el tiempo final de desarrollo se debería de estimar en unas 210 horas que ha sido la estimación se ha utilizado para la planificación que se expone más adelante.

3.2. Plan del proyecto

3.2.1. Planificación temporal

Cada una de las evoluciones de este proyecto se divide en tres fases que ya han sido explicadas en el capítulo introductorio por lo que en este punto solamente se van a volver a reflejar las ideas claves relativas a la planificación temporal.

Los datos mostrados se refieren al tiempo total dedicado a cada fase y no a cada evolución, para apreciar la distribución del tiempo en las distintas evoluciones se ha añadido al final de esta sección un diagrama de Gantt que permita observar el reparto de tiempos para cada evolución.

Tras todo el desarrollo, se realizó una revisión de la documentación de una duración estimada de 20 horas. La memoria se fue realizando de forma progresiva durante el desarrollo, pero esta revisión tenía el fin de subsanar posibles errores y poder finalizar el proyecto.

Utilizando la estimación inicial y dividiendo el esfuerzo para cada una de las fases, se

han obtenido los siguientes tiempos de desarrollo para cada una de las siguientes fases:

Fase	Objetivos	Duración (en horas)
Fase 1: Desarrollo de la librería.	Construir una librería de generación automática de preguntas sobre *"Teoría básica de conjuntos"*.	290
Fase 2: Realización de pruebas de la librería.	Implementación y aplicación de pruebas a la librería.	100
Fase 3: Despliegue de la librería.	Despliegue de la librería en Siette e implementación de una asignatura de prueba.	20

Tabla 3.1: Planificación del proyecto en fases.

Las diversas fases expuestas se dividieron en subtareas tal y como se expone a continuación (en este caso el reparto de tiempos si diferencia la evolución a la que pertenece cada una de las subtareas):

Tarea	Objetivos	Duración (en horas)
Subtarea 1: Desarrollo de prototipo aritmético.	Construir una librería de generación automática de preguntas sobre *"Aritmética"*.	80
Subtarea 2: Desarrollo de la primera evolución del sistema.	Implementación inicial del sistema y del primer subconjunto de tipos de preguntas.	60
Subtarea 3: Desarrollo de la segunda evolución del sistema.	Implementación del segundo subconjunto de tipos de preguntas.	70
Subtarea 4: Desarrollo de la tercera evolución del sistema.	Implementación del segundo subconjunto de tipos de preguntas.	80

Tabla 3.2: Subtareas de la fase de "Desarrollo de la librería".

Las tareas específicas de las pruebas y despliegue no han entrado dentro de la estimación realizada para el resto del desarrollo por tratarse de tareas completamente diferenciadas del desarrollo propiamente dicho, por lo que estas estimaciones se han establecido basándose en experiencias previas.

Tarea	Objetivos	Duración (en horas)
Subtarea 1: Implementación de JSP de prueba.	Construir una página JSP que simule la interacción del sistema Siette con la librería.	20
Subtarea 2: Implementación de pruebas unitarias.	Desarrollar las pruebas unitarias del sistema con JUnit.	20
Subtarea 3: Implementación de pruebas de rendimiento.	Creación de pruebas para medir los tiempos de generación de preguntas de la librería.	20
Subtarea 4: Realización de las pruebas.	Ejecución, obtención de datos y análisis de resultados de las pruebas.	40

Tabla 3.3: Subtareas de la fase de “Realización de pruebas de la librería”.

Tarea	Objetivos	Duración (en horas)
Subtarea 1: Despliegue de la librería.	Empaquetado de la librería en formato JAR y despliegue en el sistema Siette.	10
Subtarea 2: Desarrollo de una asignatura en Siette.	Utilizar la librería para completar el desarrollo de una asignatura en Siette.	10

Tabla 3.4: Subtareas de la fase de “Despliegue de la librería”.

Los datos que se han expuesto (incluyendo la división de tiempos entre evoluciones) se han recogido gráficamente en el diagrama de Gantt de la figura 3.1.

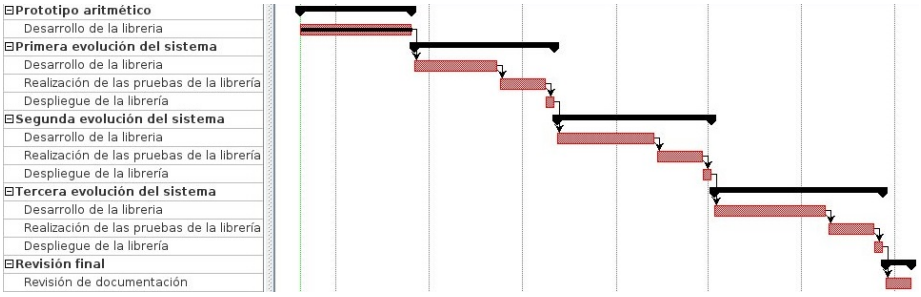


Figura 3.1: Diagrama de Gantt con la planificación temporal de las fases.

3.2.2. Recursos del proyecto

En este apartado se especifican todos los recursos necesarios para la correcta realización de este proyecto. Para una mayor claridad estos recursos están divididos en tres apartados: recursos físicos, lógicos y humanos.

Recursos físicos

Los materiales de los que se dispone para realizar este proyecto son los siguientes:

Ordenador personal de sobremesa: Intel Core 2 Quad Q6600 2.4 GHz, 8.0 GB de memoria RAM, disco duro de 1,5 TB, disco duro de 500 GB. Este ordenador dispone del sistema operativo Windows 7.

Ordenador personal portátil: Intel Core i5 430M 2,26 Ghz, 4.0 Gb de memoria RAM, discoduro de 500 GB. Este ordenador dispone del sistema operativo Windows 7.

Recursos lógicos

Las aplicaciones de las que se dispone para realizar este proyecto son los siguientes:

VMware Server 2: Herramienta de virtualización. Utilizada para virtualizar un sistema operativo Ubuntu 12.04.1 LTS.

Eclipse 3.7: IDE (Integrated Development Environment) para Java.

Java 6: Implementación openjdk.

Tomcat 7: Servidor de aplicaciones Java.

TeX-Live: Distribución de TEXpara GNU/Linux.

Texmaker: Editor de textos TEX/LATEX.

Umbrello: Herramienta para realizar diagramas en UML (Unified Modeling Language).

OpenProj 1.4: Herramienta para realizar diagramas de Gantt.

LibreOffice Calc 3.5: Herramienta para trabajar con hojas de cálculo.

Recursos humanos

Las personas que participan en el desarrollo de este proyecto son los siguientes:

Dirección: D. Manuel Luque Gallego y José Manuel Cuadra Troncoso, profesores de la Universidad Nacional de Educación a Distancia.

Desarrollo: Juan Manuel Sánchez Turrión, alumno de Ingeniería Informática en la Universidad Nacional de Educación a Distancia.

Soporte Siette: Ricardo Conejo Muñoz, profesor de la Universidad de Málaga.

3.3. Gestión de riesgos

Cuando se trabajan los riesgos hay que tener en cuenta que no todos tienen la misma incidencia sobre el proyecto. Es necesario calcular este impacto para poder establecer una preferencia a la hora de dedicar mayor esfuerzo a combatir los riesgos que tengan mayor incidencia sobre el proyecto. Para llevar a cabo esta tarea existen unas pautas establecidas que se basan en asignar un número entre 0 y 1 a la probabilidad de ocurrencia de un riesgo y otro entre 0 y 10 al impacto de dicho riesgo.

Probabilidad	Impacto
Muy baja [0, 0.2)	Despreciable [0, 2)
Baja [0.2, 0.4)	Marginal [2, 4)
Media [0.4, 0.6)	Crítico [4, 8)
Alta [0.6, 0.8)	Catastrófico [8, 10]
Muy alta [0.8, 1.0]	

Tabla 3.5: Cuantificación de la probabilidad e impacto de los riesgos.

El valor de incidencia en el proyecto de un riesgo se consigue multiplicando el valor asignado a la probabilidad y al impacto o cruzando las columnas de la tabla anterior como muestra la siguiente tabla.

	Despreciable	**Marginal**	**Crítico**	**Catastrófico**
Muy baja	Baja	Baja	Baja	Moderada
Baja	Baja	Baja	Baja	Moderada
Media	Baja	Baja	Moderada	Alta
Alta	Moderada	Moderada	Alta	Alta
Muy alta	Moderada	Moderada	Alta	Alta

Tabla 3.6: Incidencia en el proyecto de los riesgos.

3.3.1. Gestión de riesgos de la fase 1

En este apartado se describen los riesgos que pueden ocurrir durante la primera fase junto con las consecuencias que provocarían, la probabilidad de ocurrencia, el impacto sobre el proyecto y el plan de contingencia definido para disminuir los daños en caso de que estos riesgos se produzcan.

Cambio de requisitos

Descripción: Se produce un cambio en la especificación de los requisitos una vez comenzado el desarrollo del mismo.

Consecuencias: Se deberá repetir el trabajo realizado que se vea afectado por los cambios ocurridos. Como los módulos que componen el sistema son independientes la cantidad de trabajo a repetir no será excesiva si cambian los requisitos, aunque no es muy probable que ocurra este riesgo.

Incidencia en el proyecto: La probabilidad de ocurrencia es muy baja y el impacto es despreciable por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es baja.

Plan de contingencia: En caso de que se produzca este riesgo se intentará adecuar el trabajo a los nuevos requisitos si se prevée que no ocasione un retraso muy importante o si afecta a trabajo que aún no esta realizado.

Planificación demasiado optimista

Descripción: Los tiempos estimados para la duración de las tareas son menores de lo que se tarda en realizarlas, lo que provoca un retraso acumulativo en la planificación.

Consecuencias: Los plazos no se cumplen o lo hacen bajo mucha presión. En cualquier caso la calidad del sistema disminuye debido a que para poder realizar a tiempo las tareas se dedica menos tiempo a analizarlas, a probarlas, etc.

Incidencia en el proyecto: La probabilidad de ocurrencia es media pero el impacto es crítico por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: Debido a las características del proyecto y a que no hay un límite claro en la funcionalidad que se debe conseguir en el sistema, si ocurre este riesgo se reducirá parte de la funcionalidad planificada con el fin de disponer de tiempo para realizar las tareas planificadas con unos niveles adecuados de calidad.

Retraso en cascada

Descripción: Se produce un retraso en alguna tarea del camino crítico que obliga a retrasar la fecha de finalización de la tarea.

Consecuencias: Los retrasos obligan a revisar la planificación inicial para poder desarrollar el sistema con la calidad prevista.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta pero el impacto es marginal por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: Puesto que el efecto que tiene este riesgo sobre el proyecto es similar a lo que ocurriría si se diera el caso de una planificación demasiado optimista, el plan de contingencia es similar al de este riesgo.

Fallo en los recursos físicos

Descripción: Se produce un fallo en algún componente físico de los equipos con los que se realiza este proyecto.

Consecuencias: No se pueden realizar las tareas en las que sea necesario disponer del equipo dañado, provocando un retraso en la planificación.

Incidencia en el proyecto: La probabilidad de ocurrencia es muy baja y el impacto es marginal por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es baja.

Plan de contingencia: En caso de que ocurra este riesgo se procederá a la reparación o sustitución de los recursos dañados tan rápido como sea posible. Además, se comenzará a avanzar trabajo en otras tareas que se puedan realizar sin la utilización de los equipos dañados. El retraso provocado no debe ser muy grande puesto que, al no depender de otra persona para solucionar el problema, el tiempo que se tarde en solucionarlo no debería ser muy grande.

Fallo en los recursos lógicos

Descripción: Se produce un fallo en algún componente lógico que se utiliza en la realización del proyecto.

Consecuencias: Este riesgo, en caso de ocurrir, conlleva un retraso en la realización de las tareas que utilicen este recurso.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta pero el impacto es despreciable por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: En caso de que se produzca este riesgo se procederá a intentar encontrar y solucionar el problema que ha ocurrido para conseguir que los recursos lógicos afectados vuelvan a funcionar con normalidad. En caso de que el tiempo se alargue de forma que pueda llegar a desencadenar un riesgo de retraso en cascada se procederá a sustituir los recursos lógicos afectados por otros con las mismas funcionalidades. Además, al igual que ocurre con los recursos físicos, también se comenzará a avanzar en tareas que no requieran los recursos dañados para su realización.

Disconformidad del tutor

Descripción: El tutor no está de acuerdo con alguno de los pasos desarrollados en el proyecto y justifica este hecho de forma convincente.

Consecuencias: Se deben modificar los puntos de disconformidad por lo que se produce un retraso en la planificación del proyecto.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta y el impacto es crítico por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es alta.

Plan de contingencia: En caso de que ocurra este riesgo se procederá a solucionar los puntos de disconformidad rápidamente y consultando al tutor frecuentemente para comprobar que el nuevo trabajo es correcto. Para evitar que este riesgo se produzca la comunicación con el tutor será fluida y las decisiones críticas serán tomadas tras consultarlas con él.

Factores externos al proyecto

Descripción: Por motivos externos al proyecto, como exámenes o trabajo, no se puede dedicar todo el tiempo planeado a la realización del proyecto.

Consecuencias: Esto provoca un retraso en la planificación del proyecto.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta y el impacto es crítico por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es alta.

Plan de contingencia: Puesto que en caso de que se produzca este riesgo lo más normal es que se genere un retraso en cascada se procederá de la misma manera que en los casos anteriores. Para evitar que este riesgo se produzca se ha decidido planificar hasta el nivel de tareas individuales, no planificando dentro de la misma, para dejar un margen de libertad sobre la forma de realizar las tareas.

Pérdida de datos

Descripción: Se produce un borrado de datos del proyecto.

Consecuencias: La ocurrencia de este riesgo provoca tener que conseguir la información desaparecida de nuevo, lo que implica un retraso en la planificación proporcional a la dificultad de conseguir los datos perdidos de nuevo.

Incidencia en el proyecto: La probabilidad de ocurrencia es baja pero el impacto es catastrófico por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: En este caso el plan de contingencia consiste en volver a obtener los datos perdidos. Para evitar que se produzca esta situación se procederá a realizar copias de seguridad periódicamente para que, en caso de una pérdida de datos, puedan recuperarse la mayoría de una forma rápida.

3.3.2. Gestión de riesgos de la fase 2

Planificación demasiado optimista

Descripción: El tiempo estimado para el desarrollo de las pruebas es superior al planificado para ellas.

Consecuencias: Los plazos no se cumplen y no es posible realizarlo más rápido porque sin finalizar esta fase no se puede avanzar.

Incidencia en el proyecto: La probabilidad de ocurrencia es baja pero el impacto es catastrófico por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: Para evitar que se produzca este riesgo se ha planificado el desarrollo de las pruebas de forma simultanea al resto del desarrollo.

En caso de que se produzca este riesgo se intentará reducir el margen de maniobra temporal de las siguientes pruebas. Si esto no es posible se intentará reducir lo mínimo imprescindible el conjunto de pruebas a realizar.

Retraso en cascada

Descripción: Se produce un retraso en alguna tarea del camino crítico que obliga a retrasar la fecha de finalización de las tareas.

Consecuencias: Los retrasos obligan a revisar la planificación inicial para poder desarrollar el sistema con la calidad prevista.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta pero el impacto es marginal por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: Puesto que el efecto que tiene este riesgo sobre el proyecto es similar a lo que ocurriría si se diera el caso de una planificación demasiado optimista, el plan de contingencia es similar al de este riesgo.

Factores externos al proyecto

Descripción: Por motivos externos al proyecto, como exámenes o trabajo, no se puede dedicar todo el tiempo planeado a la realización del proyecto.

Consecuencias: Esto provoca un retraso en la planificación del proyecto.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta y el impacto es crítico por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es alta.

Plan de contingencia: Puesto que en caso de que se produzca este riesgo lo más normal es que se genere un retraso en cascada se procederá de la misma manera que en los casos anteriores. Para evitar que este riesgo se produzca se ha decidido retrasar lo máximo posible los periodos vacacionales para poder hacer uso de ellos como medida de contingencia.

Fallo en los recursos físicos

Descripción: Se produce un fallo en algún componente físico de los equipos con los que se realiza este proyecto.

Consecuencias: No se pueden realizar las tareas en las que sea necesario disponer del equipo dañado, provocando un retraso en la planificación.

Incidencia en el proyecto: La probabilidad de ocurrencia es muy baja y el impacto es marginal por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es baja.

Plan de contingencia: En caso de que ocurra este riesgo se procederá a la reparación o sustitución de los recursos dañados tan rápido como sea posible. Además, se comenzará a avanzar trabajo en otras tareas que se puedan realizar sin la utilización de los equipos dañados. El retraso provocado no debe ser muy grande puesto que, al no depender de otra persona para solucionar el problema, el tiempo que se tarde en solucionarlo no debería ser muy grande.

Fallo en los recursos lógicos

Descripción: Se produce un fallo en algún componente lógico que se utiliza en la realización del proyecto.

Consecuencias: Este riesgo, en caso de ocurrir, conlleva un retraso en la realización de las tareas que utilicen este recurso.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta pero el impacto es despreciable por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: En caso de que se produzca este riesgo se procederá a intentar encontrar y solucionar el problema que ha ocurrido para conseguir que los recursos lógicos afectados vuelvan a funcionar con normalidad. En caso de que el tiempo se alargue de forma que pueda llegar a desencadenar un riesgo de retraso en cascada se procederá a sustituir los recursos lógicos afectados por otros con las mismas funcionalidades. Además, al igual que ocurre con los recursos físicos, también se comenzará a avanzar en tareas que no requieran los recursos dañados para su realización.

3.3.3. Gestión de riesgos de la fase 3

Problemas con la plataforma Siette

Descripción: La plataforma Siette no está disponible cuando va a ser usada o no funciona adecuadamente. Este es un caso específico del caso genérico *Factores externos al proyecto*, pero se ha considerado a parte debido a su relevancia en esta fase.

Consecuencias: El tiempo disponible para realizar esta fase es menor que el necesario.

Incidencia en el proyecto: La probabilidad de ocurrencia es baja y aunque el impacto es crítico, según lo explicado anteriormente, la incidencia sobre el proyecto es baja.

Plan de contingencia: En caso de que se produzca este riesgo se avisará a los responsables de la plataforma lo antes posible. Para labores de prueba funcional de la librería se puede utilizar la página de prueba desarrollada en JSP.

Planificación demasiado optimista

Descripción: El tiempo estimado para la realización del despliegue es superior al planificado para ello.

Consecuencias: El tiempo disponible para realizar esta fase es menor que el necesario, bien sea por una mala planificación inicial o por un retraso en cascada acumulado de las fases anteriores.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta pero el impacto es marginal por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: En caso de que se produzca este riesgo se procurará disponer de todo el día para la realización del proyecto, cogiendo vacaciones en el puesto de trabajo para poder dedicar el máximo tiempo posible a esta tarea.

Retraso en cascada

Descripción: Se produce un retraso en alguna tarea del camino crítico que obliga a retrasar la fecha de finalización de la iteración y seguidamente la fecha de inicio de la iteración siguiente.

Consecuencias: Los retrasos obligan a revisar la planificación inicial para poder desarrollar el sistema con la calidad prevista.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta pero el impacto es marginal por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: Puesto que el efecto que tiene este riesgo sobre el proyecto es similar a lo que ocurriría si se diera el caso de una planificación demasiado optimista, el plan de contingencia es similar al de este riesgo.

Factores externos al proyecto

Descripción: Por motivos externos al proyecto, como exámenes o trabajo, no se puede dedicar todo el tiempo planeado a la realización del proyecto.

Consecuencias: Esto provoca un retraso en la planificación del proyecto.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta y el impacto es crítico por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es alta.

Plan de contingencia: Puesto que en caso de que se produzca este riesgo lo más normal es que se genere un retraso en cascada se procederá de la misma manera que en los casos anteriores. Para evitar que este riesgo se produzca se ha decidido planificar hasta el nivel de iteración, no planificando dentro de la misma, para dejar un margen de libertad sobre la forma de realizar las tareas.

Fallo en los recursos físicos

Descripción: Se produce un fallo en algún componente físico de los equipos con los que se realiza este proyecto.

Consecuencias: No se pueden realizar las tareas en las que sea necesario disponer del equipo dañado, provocando un retraso en la planificación.

Incidencia en el proyecto: La probabilidad de ocurrencia es muy baja y el impacto es marginal por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es baja.

Plan de contingencia: En caso de que ocurra este riesgo se procederá a la reparación o sustitución de los recursos dañados tan rápido como sea posible. Además, se comenzará a avanzar trabajo en otras tareas que se puedan realizar sin la utilización de los equipos dañados. El retraso provocado no debe ser muy grande puesto que, al no depender de otra persona para solucionar el problema, el tiempo que se tarde en solucionarlo no debería ser muy grande.

Fallo en los recursos lógicos

Descripción: Se produce un fallo en algún componente lógico que se utiliza en la realización del proyecto.

Consecuencias: Este riesgo, en caso de ocurrir, conlleva un retraso en la realización de las tareas que utilicen este recurso.

Incidencia en el proyecto: La probabilidad de ocurrencia es alta pero el impacto es despreciable por lo que, según lo explicado anteriormente, la incidencia sobre el proyecto es moderada.

Plan de contingencia: En caso de que se produzca este riesgo se procederá a intentar encontrar y solucionar el problema que ha ocurrido para conseguir que los recursos lógicos afectados vuelvan a funcionar con normalidad. En caso de que el tiempo se alargue de forma que pueda llegar a desencadenar un riesgo de retraso en cascada se procederá a sustituir los recursos lógicos afectados por otros con las mismas funcionalidades. Además, al igual que ocurre con los recursos físicos, también se comenzará a avanzar en tareas que no requieran los recursos dañados para su realización.

3.4. Seguimiento del proyecto

3.4.1. Riesgos ocurridos durante el desarrollo del proyecto

Durante todo el desarrollo del proyecto se ha producido frecuentemente el riesgo denominado *"factores externos al proyecto"*, causando graves retrasos en la planificación y provocando que hayan tenido que priorizarse el proyecto sobre otros compromisos adquiridos durante el tiempo de desarrollo del mismo. Por ello ya se indicó en el apartado dedicado a la planificación que la estimación debía realizarse en horas, ya que los distintos compromisos personales, profesionales, familiares, ... influían notablemente en las planificaciones dificultando en gran medida la estimación de fechas concretas.

Un riesgo que no se había contemplado era el de *"fallo en la estimación de dificultad de las tareas"* ya que conforme se iba avanzando en la materia, la dificultad de las funcionalidades a implementar aumentaba considerablemente ralentizando los tiempos estimados de desarrollo. No obstante el mayor problema residía en la labor creativa que suponía cada una de las fases de desarrollo de la librería, ya que no solo incluía la parte de diseño del código que daría soporte al requisito, sino también el diseño en sí de las preguntas que debían de adaptarse al formato test demandado por la plataforma Siette.

El riesgo de la fase 3, *"problemas con la plataforma Siette"* si que se ha producido en alguna ocasión, pero el soporte ofrecido por el profesor Ricardo Conejo, fue excelente lo que no provocó problemas en el desarrollo del proyecto.

3.4.2. Tiempos finales de ejecución

Debido a la ocurrencia de los riesgos comentados los tiempos planificados en la estimación inicial no han sido muy acertados. Este hecho se trató de corregir en cuanto se terminó la primera evolución del sistema y se comprobó que los tiempos no eran los esperados.

Utilizando el concepto de valor conseguido [Humphrey, 2001], se fue adaptando la estimación de finalización del proyecto de acuerdo al avance real de la tareas terminadas. Del mismo modo se fueron aplicando todas las medidas de contingencia que fueron posibles de cara a reducir el impacto de la mala estimación inicial en el desarrollo de todas las labores del proyecto.

Finalmente tras todos los esfuerzos realizados los tiempos obtenidos se muestran en las tablas que se encuentran a continuación.

Fase	**Tiempo final empleado (en horas)**
Fase 1: Desarrollo de la librería.	510
Fase 2: Realización de pruebas de la librería.	88
Fase 3: Despliegue de la librería.	21

Tabla 3.7: Seguimiento de la planificación del proyecto en fases.

Como se ha indicado previamente, las diferentes fases constan de distintas subtareas, cuyos tiempos de realización finales se detallan en las próximas tablas.

Fase	Tiempo final empleado (en horas)
Subtarea 1: Desarrollo de prototipo aritmético.	83
Subtarea 2: Desarrollo de la primera evolución del sistema.	124
Subtarea 3: Desarrollo de la segunda evolución del sistema.	131
Subtarea 4: Desarrollo de la tercera evolución del sistema.	172

Tabla 3.8: Seguimiento de las subtareas de la fase de "Desarrollo de la librería".

En la tabla 3.8, se observa la desviación con respecto a la estimación inicial de la Subtarea 2, aparentemente la Subtarea 3 no se desvió tanto con relación a la Subtarea 2, pero finalmente la Subtarea 4 puso de relieve la inexperiencia en la estimación desviándose desmesuradamente y no respondiendo a las medidas de contingencia aplicadas.

Fase	Tiempo final empleado (en horas)
Subtarea 1: Implementación de JSP de prueba.	12
Subtarea 2: Implementación de pruebas unitarias.	18
Subtarea 3: Implementación de pruebas de rendimiento.	15
Subtarea 4: Realización de las pruebas.	43

Tabla 3.9: Seguimiento de las subtareas de la fase de "Realización de pruebas de la librería".

Fase	Tiempo final empleado (en horas)
Subtarea 1: Despliegue de la librería.	13
Subtarea 2: Desarrollo de una asignatura en Siette.	8

Tabla 3.10: Seguimiento de las subtareas de la fase de "Despliegue de la librería".

La duración final de la tarea de revisión de la documentación fue de 27 horas.

Los datos presentados pueden observarse de forma gráfica en la figura 3.2, en la que se ha utilizado la misma escala temporal que en la figura 3.1 para que las divergencias puedan apreciarse a simple vista.

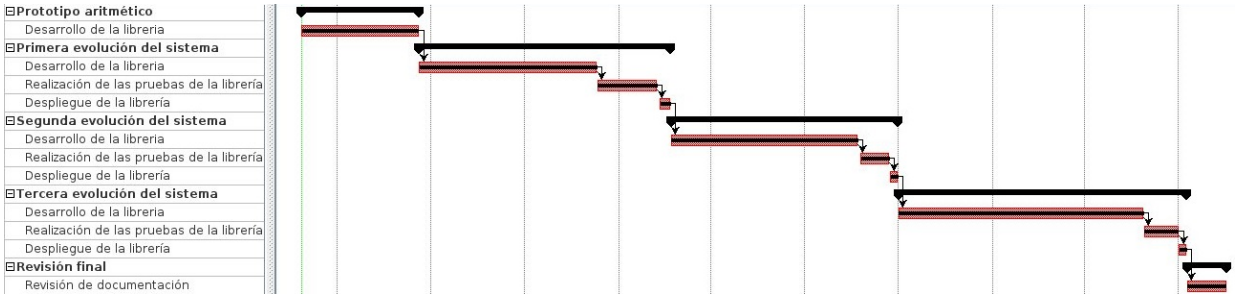


Figura 3.2: Diagrama de Gantt con la planificación temporal final de las fases.

3.4.3. Tamaño final del producto

Una vez finalizado todo el trabajo de codificación de la librería merece la pena analizar el volumen de código obtenido.

Finalmente el tamaño del código de la librería es de unas 4,2 KLOC, de las cuales 3,1 KLOC son de código como tal (que divergen bastante de las 4,2 KLOC inicialmente estimados) y 1,1 son comentarios, lo cual supone más de una cuarta parte del producto final (26 %). Este hecho indica que el requisito referido a la documentación estaría correctamente abordado al menos en lo que a volumen se refiere.

Por otro lado las pruebas (codificadas a parte del código propio de la librería) han supuesto 2,1 KLOC de las cuales 1,6 KLOC son de código como tal y 0,5 KLOC son comentarios, se mantiene pues la relación comentarios/código (24 %).

La modularidad del sistema generado era uno de los elementos claves que marcaron el desarrollo del mismo, este hecho se ve ratificado por el elevado número de ficheros que componen el código de la librería que asciende a 63 de los cuales 35 son para pruebas y 28 para la librería como tal. La relación código/fichero por tanto es de aproximadamente 150 LOC/fichero en el caso del código de la librería y 60 LOC/fichero en el caso del código de las pruebas. No obstante es importante recalcar que la distribución de código entre ficheros para el código de la librería no es especialmente uniforme y el grueso de la codificación se encuentra en los métodos que generan las distintas preguntas de la librería.

Capítulo 4

Especificación de requisitos

En este capítulo se establece una explicación general de los requisitos del sistema.

Dado que los requisitos no contaban con una dificultad excesiva y puesto que no se iban a modificar con el tiempo (si bien si se añadirían módulos a lo largo del tiempo con la intención de aumentar la diversidad de preguntas, los requisitos no variaban y no influían en la arquitectura del sistema), no se ha precisado de una gestión de los cambios de requisitos. No obstante la gestión del cambio tanto de requisitos como de funcionalidades es necesaria en proyectos algo más extensos, o en los que hubiera que llegar a acuerdos con el solicitante del producto.

A continuación se expone de forma resumida la declaración de necesidades inicial del cliente y posteriormente se expondrá la especificación formal de los requisitos identificados.

4.1. Declaración de necesidades del cliente

El cliente enumeró las siguientes necesidades básicas que debía de cubrir el sistema:

1. El principal objetivo del proyecto es la generación automática de preguntas para los alumnos (pueden ser de tipo test o de otro si se considera adecuado).

2. El dominio sobre el que versarán las preguntas será la *"Teoría básica de conjuntos"* de la asignatura *"Lógica y Estructuras Discretas"* de primer curso de los *Grados de Ingeniería Informática y de Tecnologías de la Información*.

3. El lenguaje de programación en que se implementará el proyecto es Java.

4. El software desarrollado tendrá que componerse de dos elementos principales:

   a) Una librería (o conjunto de librerías) que permitan a un programa cliente obtener preguntas generadas para la asignatura concreta.

   b) Una asignatura de prueba en el sistema Siette que demuestre que la librería implementada funciona adecuadamente.

5. La librería podrá hacer uso de la API de Java y llamar a archivos JAR disponibles en internet siempre que se trate de software libre (no está permitido el uso de software local).

6. Se deben de abordar al menos las siguientes pruebas:

   a) Pruebas de unidad (JUnit).

   b) Pruebas que demuestren que la librería funciona en un entorno Unix (Linux, Mac, etc.) mediante una página JSP de prueba.

   c) Un estudio de la eficiencia de la librería que muestre estadísticas de tiempo, que garanticen un rendimiento aceptable de la librería.

7. La documentación generada para todo el proyecto deberá ser clara.

4.2. Especificación de requisitos del sistema

Una vez se cuenta con la declaración de necesidades del cliente, se puede pasar a describir formalmente los requisitos, para lo que es posible comenzar dividiendo los requisitos en funcionales y no funcionales.

4.2.1. Requisitos funcionales

1. **Sistema de generación automática de preguntas.** Debe de permitir obtener enunciados de preguntas relacionados con la *"Teoría básica de conjuntos"*, así como respuestas a esos enunciados permitiendo la posibilidad de identificar si la respuesta es

correcta o incorrecta. Este requisito genérico se dividió en otros más específicos para cada una de las fases evolutivas de las que constó el proyecto:

Primera evolución del sistema: El subconjunto específico de tipos de preguntas de esta fase son:

a) Operaciones simples con conjuntos.

b) Cardinalidad de conjuntos.

c) Pertenencia de elementos a conjuntos.

Segunda evolución del sistema: El subconjunto específico de tipos de preguntas de esta fase son:

a) Operaciones complejas con conjuntos.

b) Subconjuntos y superconjuntos.

Tercera evolución del sistema: El subconjunto específico de tipos de preguntas de esta fase son:

a) Conjuntos disjuntos.

b) Partición de un conjunto.

c) Identificación de expresiones equivalentes.

d) Aplicación de propiedades básicas de conjuntos.

e) Producto cartesiano.

f) Conjunto potencia.

g) Conjuntos notables.

h) Preguntas sobre conceptos teóricos.

4.2.2. Requisitos no funcionales

1. El lenguaje de desarrollo será Java.

   a) Se implementarán métodos de acceso a los datos que oculten la arquitectura interna del sistema y simplifiquen el uso de la librería.

2. Se debe desarrollar una asignatura en la plataforma Siette que haga uso del *Sistema de generación automática de preguntas*.

3. Las librerías utilizadas por el *Sistema de generación automática de preguntas* solo podrán ser propias de Java o de libre distribución.

4. Deberán de implementarse al menos tres tipos de pruebas:

   a) **Unitarias.** Las pruebas unitarias se desarrollarán utilizando el framework JUnit.

   b) **De rendimiento.** Las pruebas de rendimiento verificarán que los tiempos de generación del *Sistema de generación automática de preguntas* son inferiores a un segundo.

   c) **De simulación.** Se desarrollará una página JSP que permita comprobar la funcionalidad del *Sistema de generación automática de preguntas* fuera de la plataforma Siette.

5. La documentación generada ha de ser concisa, escueta y facilitar la comprensión de los conceptos explicados. En concreto este requisito debe de aplicarse a:

   - La memoria del proyecto.
   - Los comentarios del código.

Capítulo 5

Análisis del sistema

En este capítulo se mostrará la documentación obtenida durante el proceso de análisis y diseño de la aplicación desarrollada durante el proyecto fin de carrera. Puesto que esta aplicación es muy sencilla, las diferencias entre la parte de análisis y diseño son mínimas y en ocasiones es difícil establecer una clara separación, no obstante se ha tratado de dar la mayor claridad posible a la exposición.

Para comenzar, se presentará el diagrama de casos de uso junto con la especificación de los mismos. A continuación se mostrarán los diagramas de secuencia para los casos de uso identificados y posteriormente se pasará al capítulo de diseño.

5.1. Casos de uso

Los diagramas de casos de uso muestran la funcionalidad y comportamiento de un sistema mediante la interacción del mismo con otros usuarios y/o sistemas. Tanto los usuarios como los sistemas reciben el nombre de actores.

En el sistema inicialmente solo iba a existir un actor (la plataforma Siette), pero debido a la versatilidad de la librería y a las tareas realizadas con el prototipo inicial basado en aritmética, se decidió permitir su utilización directa por parte del usuario (a través de ciertas interfaces de usuario que sirven para generar tests en ficheros que pueden ser leídos fuera de la plataforma Siette), por eso han sido necesarios dos actores.

Figura 5.1: Diagrama de casos de uso del sistema.

Como puede observarse en la figura 5.1, el diagrama es extremadamente sencillo, ya que la arquitectura básica del sistema es bastante simple y el esfuerzo debe centrarse en los algoritmos de generación de preguntas como se verá a lo largo de la presente exposición.

Básicamente el caso de uso principal es el de Generar Pregunta, que es el encargado de proporcionar una pregunta al actor o caso de uso que lo utiliza, como se ha comentado previamente el caso de uso Generar Test incluye al caso de uso Generar Pregunta, ya que hace uso del mismo para generar cada una de las preguntas individuales que componen un fichero de test completo.

Especificación de los casos de uso

A continuación se explica el comportamiento concreto de cada uno de los casos de uso que aparecen en el diagrama de la figura 5.1.

Tabla 5.1: Caso de uso: Generar Test.

CU-01	**Generar Test**
Descripción	Este caso de uso se realiza cuando un usuario solicita un conjunto de preguntas la librería. Para generar cada una de las preguntas se utiliza el caso de uso “Generar Pregunta”.
Precondiciones	El usuario debe de especificar el número de preguntas que desea.
Postcondiciones	El usuario obtiene las preguntas solicitadas.
Paso	**Acción**
1	El usuario informa del número de preguntas que desea.

CU-01	Generar Test (continuación)
2	La librería comprueba que el número de preguntas solicitadas es adecuado, de no ser así, restablecerá el número de preguntas al valor por defecto definido por la librería.
3	La librería genera al azar un tipo de pregunta, y su configuración específica de ser necesario, para cada una de las preguntas solicitadas.
4	La librería pone a disposición del usuario los datos de las preguntas generadas.

Tabla 5.2: Caso de uso: Generar Pregunta.

CU-02	Generar Pregunta
Descripción	Este caso de uso se realiza cuando un sistema solicita la generación de una pregunta.
Precondiciones	El sistema solicitante ha de especificar el tipo de pregunta a generar, así como la configuración específica de cada tipo de pregunta de ser necesario.
Postcondiciones	El sistema solicitante puede recoger los datos de la pregunta solicitada.
Paso	**Acción**
1	El sistema solicitante especifica el tipo de pregunta.
2	El sistema solicitante especifica la configuración específica para el tipo de pregunta especificado en 1.
3	La librería valida los datos introducidos y si son correctos genera los datos del tipo de pregunta solicitada.

CU-02	Generar Pregunta (continuación)
$3'$	Si los datos suministrados no son correctos, la librería los restablece a los valores por defecto y genera los datos del tipo de pregunta solicitada.
4	La librería pone a disposición del sistema solicitante los datos de la pregunta generada.

El caso de uso Generar Pregunta, podría haberse especializado en otros casos de uso más específicos atendiendo al subtipo específico de pregunta, pero las diferencias entre esos casos de uso especializados apenas sería apreciable y se ha preferido exponer esa especialización en el diseño de las preguntas que se abordará más adelante.

En el apartado siguiente se exponen de forma gráfica los diagramas de secuencia que se corresponden con cada uno de los casos de uso expuestos en esta sección.

5.2. Diagramas de secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Los diagramas de secuencia contienen detalles de la implementación del escenario expuesto en el caso de uso, incluyendo los objetos y clases que participan en el escenario y cuales son los mensajes que intercambian.

Los diagramas expuestos a continuación incluyen algunos aspectos iniciales de diseño, mostrando ya algunos de los métodos que finalmente ha de implementar cada clase.

La herramienta utilizada para modelar los diagramas (Umbrello), no es especialmente versátil en este tipo de diagramas y se han modelado intentando mostrar la mejor claridad posible. No obstante, estos diagramas junto con las descripciones realizadas previamente han de servir para ofrecer una visión general del modelo a implementar.

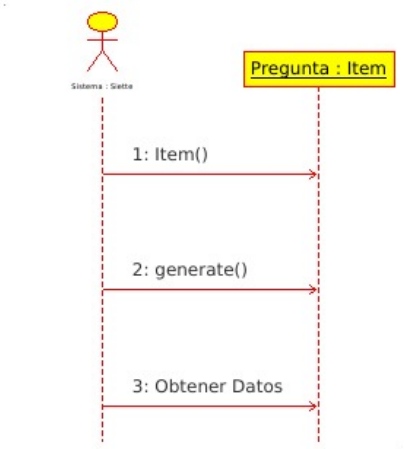


Figura 5.2: Diagrama de secuencia para el caso de uso *Generar Pregunta*.

En el diagrama de secuencia de la figura 5.2, se ha presentado la secuencia básica del caso de uso Generar Pregunta, como puede verse consta de 3 pasos básicos, en primer lugar la instanciación de un objeto Pregunta de la clase Item (más adelante se detallará en que manera se especializará esta clase para dar una cobertura completa al dominio de conocimiento que se quiere cubrir), posteriormente se encuentra el paso de generación de los datos que componen la pregunta (el método generate()), finalmente se han agrupado en un solo paso genérico (Obtener datos), una serie de pasos orientados a la obtención de los datos que componen la pregunta (enunciado, opciones de respuesta, ayudas, etc. . .).

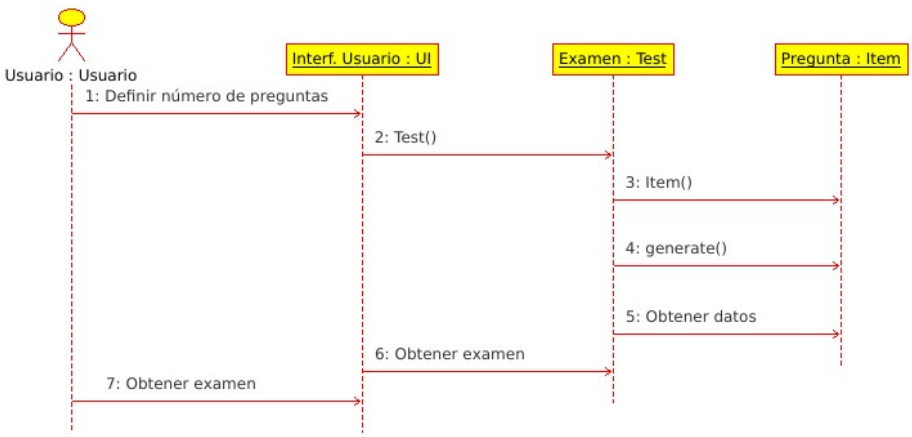


Figura 5.3: Diagrama de secuencia para el caso de uso *Generar Test*.

En el diagrama de secuencia de la figura 5.3, se ha presentado la secuencia básica del caso de uso Generar Test, como puede verse en este caso (pasos centrales de la secuencia, numerados con orden 3, 4 y 5), existen los mismos tres pasos que se comentaron para el diagrama de secuencia básica del caso de uso Generar Pregunta, esto es debido a que el caso de uso Generar Test incluye al caso de uso Generar Pregunta.

Los pasos adicionales que se han incluido en este caso de uso se corresponden en primer lugar al paso en el que el usuario define en el objeto Interfaz de Usuario de la clase UI el número de preguntas que tendrá el test (paso Definir número de preguntas), la Interfaz de Usuario a su vez instancia un objeto Examen de la clase Test que será el encargado de ir instanciando el número de las diferentes preguntas que ha definido el usuario. A continuación (una vez definidas todas las preguntas indicadas por el usuario) el paso Obtener examen traslada el examen a la Interfaz de Usuario, que a su vez, en el último paso de esta secuencia, presenta el Examen al usuario.

5.3. Descripción de los datos

Si bien en este proyecto no se ha hecho uso de ninguna tecnología de acceso/almacenamiento de datos en una base de datos y como tal no existe un diagrama entidad relación que defina el modelo de datos utilizado, si que se puede realizar una descripción de la entidad más importante de toda la librería en cuanto a los datos que la componen.

La entidad que modela esta arquitectura de datos es la clase Pregunta que básicamente está compuesta por los siguientes datos:

Enunciado Es un texto que expone el contenido de la pregunta.

Opciones Son diversos textos que proponen respuestas correctas/incorrectas al enunciado de la pregunta, internamente contienen los siguientes datos:

Respuesta Es el texto de la respuesta propuesta.

Corrección Es un campo que indica si la respuesta es correcta o incorrecta.

Explicación En caso de que la respuesta sea incorrecta este es un texto que ofrece una explicación de porqué esta respuesta es incorrecta.

Ayudas Son uno o varios textos que facilitan la resolución de la pregunta enunciada.

Refuerzo Es un texto que contribuye a reforzar el conocimiento al que se refiere la pregunta.

Como se ha comentado, esta es una descripción informal de los principales datos de los que hace uso la librería, ya que debido a su sencillez, prácticamente cualquier técnica de modelado que se pueda utilizar, no va a aportar valor añadido a la presente exposición, no obstante más adelante, dentro de los diagramas de clase dedicados a modelar esta entidad se puede obtener una representación gráfica más formal de lo que se ha expuesto en esta sección.

Adicionalmente, como se comenta en múltiples ocasiones a lo largo de la presente memoria, existe una entidad más dentro del sistema que merece la pena reseñar si bien no era imprescindible incluirla en la librería siguiendo estrictamente la Especificación de Requisitos del Software, pero se ha considerado que aportaba funcionalidades importantes para el esfuerzo que requería su implementación, esta entidad es la que define los tests (agrupaciones de preguntas) y que se ha utilizado para dar consistencia conceptual al proceso de generación de exámenes en fichero.

Los datos que integran la citada entidad se pueden describir, brevemente y de la misma manera informal utilizada para describir a la entidad que modela las preguntas, de la siguiente manera:

Título Es un texto que sirve para definir el examen o test.

Descripción Es un texto que sirve para describir con algo más de amplitud que el título el contenido del examen o test.

Número de preguntas Es un número que indica de cuantas preguntas consta el examen o test.

Tipo de preguntas Es un identificador que define la tipología de preguntas que forman el examen o test.

Preguntas Es un conjunto de preguntas que componen todo el examen o test. Estas preguntas se corresponden con la entidad descrita previamente.

No se ha considerado relevante describir en este apartado los datos que se han utilizado para realizar la implementación específica de los métodos de generación de preguntas (conjuntos, expresiones, etc...) ya que se ha considerado que esos eran detalles más relativos al diseño y la implementación específica de la solución y por tanto no tenían cabida en esta fase del desarrollo.

5.4. Análisis básico de la arquitectura de clases

En este punto de la exposición merece la pena introducir brevemente un diagrama de clases (mostrado en la figura 5.4) que describa someramente la arquitectura de clases básica que puede dar soporte al sistema analizado hasta ahora.

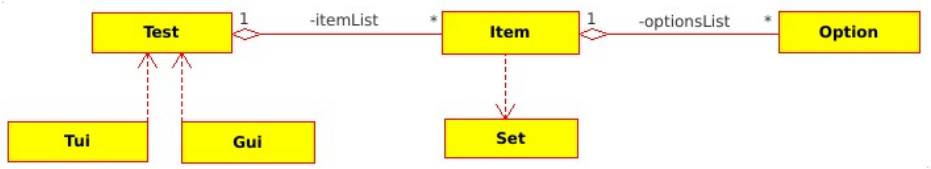


Figura 5.4: Diagrama de clases modelando el análisis inicial del sistema.

En la figura 5.4 se pueden observar algunas de las clases más relevantes del sistema: Dos interfaces de usuario: una de texto (Tui) y otra gráfica (Gui) que se sirven de la clase que modela los tests (Test) para dar soporte a la generación de tests en formato de fichero. Los tests están compuestos de preguntas (Item) que hacen uso de la entidad conjunto (Set) para representar la entidad homónima del dominio del conocimiento cubierto por la librería y que está compuesta por diversas opciones de respuesta (Options).

Este diagrama será refinado en fases posteriores de la documentación del proyecto, pero en el ya se pueden observar las clases que conformarán el núcleo básico de la librería.

Capítulo 6

Diseño de la solución

En este capítulo se expondrá la arquitectura que se ha diseñado para dar soporte al sistema, basándose en los datos recogidos previamente.

6.1. Diagramas de clases

Un diagrama de clases representa las clases que contiene el sistema, de forma estática, con sus relaciones estructurales y de herencia. En las clases se incluye la información sobre propiedades y operaciones que definen la información relevante para los objetos que se crearán a partir de esas clases y las acciones que podrán hacer los mismos.

Con los casos de uso y diagramas de secuencia expuestos hasta ahora, así como el análisis preliminar realizado tras el prototipo aritmético, se puede mostrar un diagrama de clases que represente el análisis inicial del sistema que puede verse en la figura 6.1, este diagrama ya fue expuesto en el apartado 5.4, pero en este punto de la memoria merece la pena repetirlo ya que servirá como introducción al diagrama más refinado que se expondrá a continuación.

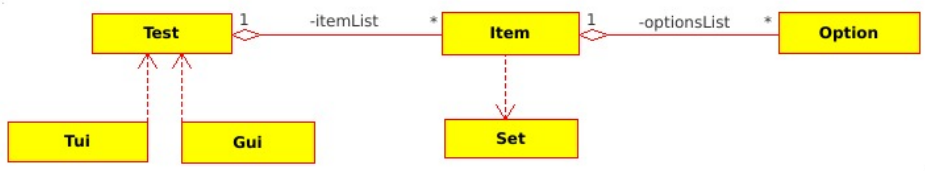


Figura 6.1: Diagrama de clases modelando el análisis inicial del sistema.

En la figura 6.1 se pueden observar algunas de las clases más relevantes del sistema: Dos interfaces de usuario, una de texto (Tui) y otra gráfica (Gui) que utilizan la clase que modela los tests (Test) que está compuesta de preguntas (Item) que hacen uso de la entidad conjunto (Set) y que están compuestas por opciones de respuesta (Options).

6.2. Diseño básico

Una vez analizado el sistema se puede pasar a desarrollar un diseño del mismo a un nivel básico que puede observarse en la figura 6.2.

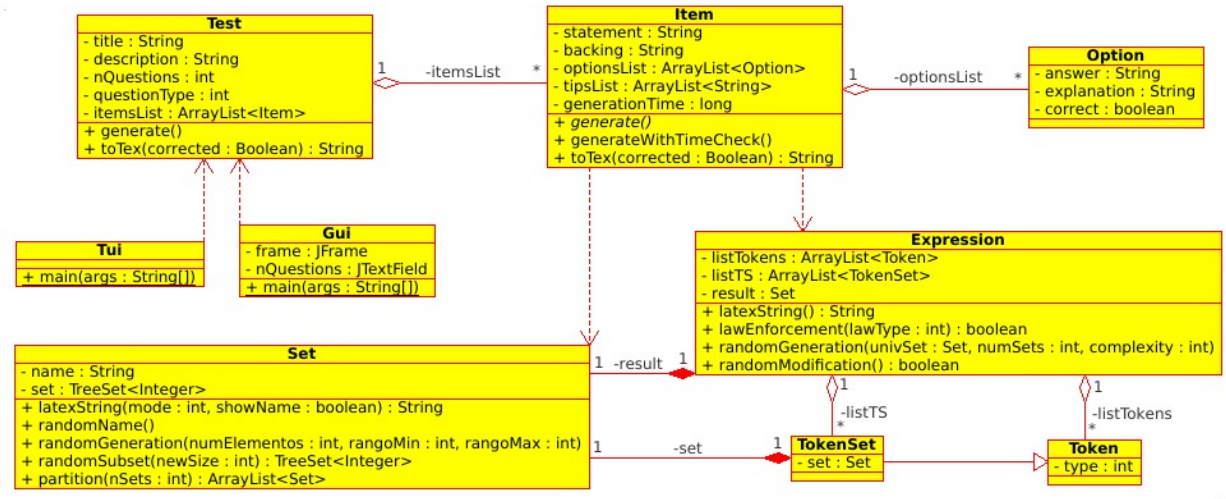


Figura 6.2: Diagrama de clases modelando el diseño básico del sistema.

En el diseño expuesto, se ha utilizado la nomenclatura (tanto para propiedades como para métodos) definitiva que se ha empleado en la implementación definitiva del diseño. Es importante reseñar que todas las clases han de poseer métodos de construcción y acceso a propiedades, que para facilitar la exposición no se han incluido en el diagrama.

En esta ocasión, figuran en el diagrama las clases que se dedicarán a las interfaces de usuarios (*Tui* y *Gui*) que mediante la clase *Test*, permitirán generar exámenes sin necesidad de conexión a Internet.

La labor de generación de exámenes se desarrollará apoyándose en el núcleo de la librería que será la clase *Item* que es la clase que modela las preguntas. Al tratarse de preguntas del

dominio de la Teoría de Conjuntos, se modelará en la clase *Set* la entidad matemática de conjunto, que ayudará a la clase *Item* a desarrollar sus funciones. Adicionalmente la clase *Option* es la que almacena las diversas respuestas de una pregunta.

En este diagrama de diseño aparecen ciertas clases adicionales que no figuran en el análisis inicial (*Expression*, *Token* y *TokenSet*), esto es debido a que, como se comentará más adelante, es el único caso en el que se ha modificado la arquitectura inicial identificada para poder modelar las expresiones con conjuntos.

A continuación se exponen algunas de las características más relevantes del diseño expuesto:

Las clases *Tui* y *Gui*, deben poseer el método principal de ejecución (main) y en el caso de *Gui* una propiedad para almacenar el parámetro de número de preguntas (nQuestions). En el caso de la clase *Tui*, el número de preguntas se almacena como parámetro.

La clase *Test* contiene propiedades que definen el examen: título (title), descripción (description), número de preguntas del examen (nQuestions), tipo de preguntas del examen (questionType) y el conjunto de preguntas que forman el examen (itemsList). Las dos funcionalidades básicas de esta clase son:

- Un método de generación del examen (generate).
- Un método de representación en formato LaTeX(toTex).

La clase *Item*, tiene como propiedades básicas: enunciado (statement), refuerzo (backing), conjunto de respuestas (optionsList), conjunto de ayudas (tipsList) y de cara a medir rendimientos, el tiempo de generación de la pregunta (generationTime). Al igual que la clase *Test*, tiene dos funcionalidades básicas:

- Un método de generación de la pregunta (generate), pese a que es un detalle específico de implementación, en el diagrama se ha incluido el método relacionado *generateWithTimeCheck* que se utilizará para facilitar la realización de pruebas de rendimiento de tiempo.
- Un método de representación en formato LaTeX(toTex).

Esta clase deberá, mediante especialización implementar los diferentes tipos de preguntas que darán funcionalidad completa a la librería, a continuación se indican cuales han sido las funcionalidades que se han implementando en las diferentes evoluciones del sistema (se ha utilizado la nomenclatura de clases utilizada durante la implementación del sistema):

- **Primera evolución:**
  1. SimpleOperation
  2. Complement
  3. Cardinality
  4. Membership
- **Segunda evolución:**
  1. ComplexOperation
  2. SubSuperSetRelations
  3. SubSuperSetTerms
- **Tercera evolución:**
  1. Disjoint
  2. Partition
  3. EquivalentExpressions
  4. LawEnforcement
  5. CartessianProduct
  6. PowerSet
  7. SpecialSets
  8. Theory

La especialización de la clase *Item* no se ha mostrado en el diagrama de la figura 6.2 puesto que, como se verá más adelante incluye ciertos aspectos específicos de la implementación

finalmente utilizada y a estas alturas de la memoria es suficiente con la descripción textual aportada, que cumple con la doble función de exponer el rango de la especialización de la clase *Item*, así como de indicar cuales han sido los módulos abordados en cada evolución de la construcción del sistema.

No obstante más adelante (en el apartado 6.2.1) se incluye una descripción más pormenorizada del diseño seguido en cada una de las preguntas recogidas en el listado previo.

Continuando con el análisis del diagrama expuesto en la figura 6.2, la clase *Set* está compuesta por un nombre (name) y un conjunto de elementos (set) y necesita diversas funcionalidades:

- Un método de generación aleatoria de elementos del conjunto (randomGeneration).
- Un método de generación aleatoria del nombre del conjunto (randomName).
- Un método de representación en formato LaTeX(latexString).
- Métodos de operaciones de conjuntos (randomSubset y partition).

La clase *Expression* está compuesta por una secuencia de operandos y operadores (listTokens), un conjunto de operandos (listTS) y el resultado de la operación (result) y ha de contar con diversas funcionalidades:

- Un método de generación aleatoria de la expresión (randomGeneration).
- Un método de representación en formato LaTeX(latexString).
- Un método de aplicación de una propiedad básica de conjuntos (lawEnforcement).
- Un método que permita modificar la expresión (randomModification).

Las clases *Token* y *TokenSet* tan solo necesitan las propiedades: tipo de elemento (type) y conjunto operador (set) respectivamente.

La clase *Option* simplemente modela una respuesta y está compuesta por: la respuesta (answer), una explicación en caso de no ser una respuesta correcta (explanation) y un indicador de si la respuesta es correcta o no (correct).

6.2.1. Diseño de las preguntas

Para conseguir cierta amplitud en la cobertura de conocimientos de la librería, se han diseñado distintos tipos de preguntas.

A continuación se exponen las diferentes preguntas diseñadas. El orden de presentación de las preguntas se corresponde con el orden de implementación a lo largo de las diferentes evoluciones del desarrollo del sistema. Se ha utilizado la ya expuesta nomenclatura de clases que se ha manejado finalmente para la implementación del sistema:

SimpleOperation: Esta clase modela preguntas en las que se exponen dos conjuntos y se pregunta por el resultado de una operación de conjuntos (unión, intersección o diferencia) realizada con esos conjuntos como operadores. Las opciones de respuesta que se presentan son diferentes conjuntos entre los que se encuentra el resultado correcto de la operación.

Complement: Esta clase modela preguntas en las que se expone un conjunto, así como el conjunto universal y se pregunta por el complemento del conjunto. Las opciones de respuesta que se presentan son diferentes conjuntos entre los que se encuentra el resultado correcto de la operación complemento. Esta pregunta se ha modelado fuera de la clase SimpleOperation puesto que presentaba ciertas características diferenciadoras del resto de operaciones: era una operación unaria y no binaria y además requería la presentación del conjunto universal.

Cardinality: Esta clase modela preguntas en las que se expone un conjunto o una expresión y se pregunta por la cardinalidad del conjunto o del resultado de la expresión. Las opciones de respuesta que se presentan son diferentes números enteros cercanos en su magnitud al resultado correcto de la pregunta que se encuentra entre las opciones presentadas.

Membership: Esta clase modela preguntas en las que se expone una relación de pertenencia de un elemento a un conjunto y se solicita el listado de miembros de ese conjunto para que esa relación de pertenencia sea correcta. Las opciones de respuesta que se presentan son los diferentes listados de miembros para el conjunto, de los cuales solamente uno cumple la relación de pertenencia enunciada.

ComplexOperation: Esta clase modela preguntas en las que se exponen varios conjuntos y se pregunta por el resultado de varias operaciones de conjuntos (unión, intersección, diferencia o complemento) realizadas con esos conjuntos como operadores. Las opciones de

respuesta que se presentan son diferentes conjuntos entre los que se encuentra el resultado correcto de las operaciones.

SubSuperSetRelations: Esta clase modela preguntas en las que se exponen dos conjuntos y se pregunta por la relación existente entre ambos (en términos de subconjuntos). Las opciones de respuesta que se presentan son distintas relaciones de subconjunto o superconjunto que cumplen o no los conjuntos enunciados.

SubSuperSetTerms: Esta clase modela preguntas en las que directamente se solicita que se encuentren las opciones correctas entre las expuestas. Las opciones de respuesta que se presentan constan de dos conjuntos y una relación de subconjunto o superconjunto que puede ser cierta o falsa.

Disjoint: Esta clase modela preguntas en las que se solicita seleccionar la pareja de conjuntos que cumplen el requisito de que son disjuntos, es decir que no contienen elementos comunes a ambos conjuntos. Las opciones de respuesta que se presentan constan de pares de conjuntos, de todas las parejas de conjuntos expuestos solo una cumple el requisito solicitado.

Partition: Esta clase modela preguntas en las que se expone un conjunto y se solicita seleccionar una partición válida del mismo. Las opciones de respuesta que se presentan constan de particiones aparentemente válidas del conjunto pero que en realidad solo una de ellas es correcta, ya que el resto cuentan con alguna característica que las convierte en inválidas.

EquivalentExpressions: Esta clase modela preguntas en las que expone una expresión y se solicita encontrar una expresión equivalente. Las opciones de respuesta que se presentan son diversas expresiones similares entre las que se encuentra una que es equivalente a la enunciada. Para poder implementar expresiones equivalentes de una manera no demasiado complicada, se han utilizado diversas propiedades básicas de conjuntos que aplicadas sucesivamente generan expresiones equivalentes.

LawEnforcement: Esta clase modela preguntas en las que se exponen dos expresiones equivalentes que se diferencian porque una de ellas tiene aplicada alguna de las propiedades básicas de conjuntos y lo que se solicita es identificar cual de esas propiedades se ha aplicado. Las opciones de respuesta que se presentan son diferentes propiedades básicas de conjuntos

entre las cuales se encuentra la que se puede aplicar para convertir las dos expresiones en iguales.

CartessianProduct: Esta clase modela preguntas en las que exponen dos conjuntos y lo que se solicita es identificar una tupla válida utilizando ambos conjuntos. Las opciones de respuesta que se presentan son diferentes tuplas de las cuales una de ellas es una tupla válida para esos conjuntos.

PowerSet: Esta clase modela preguntas en las que se expone un conjunto inicial cualquiera y se solicita identificar un conjunto que pertenezca al conjunto potencia del conjunto inicial. Las opciones de respuesta que se presentan son diferentes conjuntos entre los que se encuentra uno perteneciente al conjunto potencia del conjunto inicial.

SpecialSets: Esta clase modela preguntas en las que se expone un elemento de alguno de los conjuntos notables matemáticos y se solicita identificar el conjunto notable más restrictivo (ya que unos conjuntos contienen a otros) al que pertenezca. Las opciones de respuesta que se presentan son diferentes conjuntos notables entre los que se encuentra el conjunto notable al que pertenece el elemento expuesto.

Theory: Esta clase modela preguntas en las que se expone algún concepto básico de teoría de conjuntos y se solicita identificar el concepto expuesto. Las opciones de respuesta que se presentan son distintos conceptos de los cuales, solamente uno de ellos es correcto con respecto a los datos expuestos en el enunciado.

Capítulo 7

Implementación

Este capítulo expone algunas de las cuestiones más relevantes sobre la implementación de los diferentes módulos que integran la librería.

Solo se expondrán aquellos detalles que realmente hayan requerido una atención especial durante el proceso de implementación, los detalles menos importantes se pueden consultar en la documentación del código en formato Javadoc que se encuentra disponible para su descarga en el espacio web habilitado para albergar los contenidos adicionales de este proyecto: `https://sites.google.com/site/aqgpfc/`.

Utilizando el diseño inicial mostrado, puede definirse una estructura de paquetes que englobe las entidades conceptuales identificadas. Se ha procurado que la agrupación de clases por paquete cumpla la función de separar por finalidad los distintos grupos de clases que integran todo el sistema.

Como punto de partida, en la figura 7.1 se muestra el diagrama de paquetes utilizado. A continuación se mostrará el contenido de cada uno de estos paquetes.

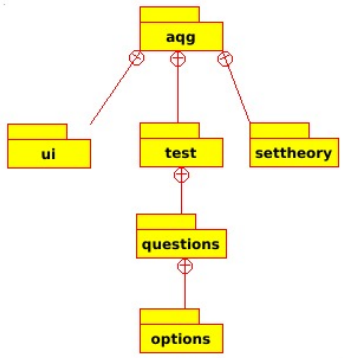


Figura 7.1: Diagrama de distribución de paquetes.

Como puede observarse, el paquete que engloba toda la arquitectura de clases es el *aqg* (que es un acrónimo para Automatic Question Generation), por otro lado está el paquete *settheory* que es el que contiene las clases dedicadas al modelado de funcionalidades relativas a la teoría de conjuntos y luego aparece el paquete *test* que contiene las clases para modelar los tests y el subpaquete *questions* que contiene las clases para modelar las diferentes preguntas y el subpaquete *options* que es el paquete que contiene la clase dedicada al modelado de las opciones de respuesta de las preguntas.

Existe una clase (*es.uned.dia.aqg.Config*), que no ha sido representada en los diagramas de clases debido a que simplemente se ha implementado con propósitos demostrativos de la capacidad de extensibilidad de la librería mediante ficheros XML, de propiedades, etc...

La citada clase simula el acceso a los comentados ficheros de configuración, para que en caso de desearse hacer efectivo este funcionamiento, bastaría con reescribir la clase para proporcionar la funcionalidad.

7.1. Paquete *settheory*

El primer paquete que se va a desarrollar es el de *settheory*. En la figura 7.2 se puede ver la composición de este paquete que es en el que se engloban todas las clases encargadas de dar soporte a operaciones relacionas con la teoría básica de conjuntos.

Inicialmente se identificó únicamente el objeto que modela la entidad conjunto, pero en

la segunda evolución del sistema, se hizo necesario diseñar clases auxiliares para las operaciones complejas con conjuntos. Es el único caso en el que ha existido alguna modificación reseñable de la arquitectura en las diversas evoluciones.

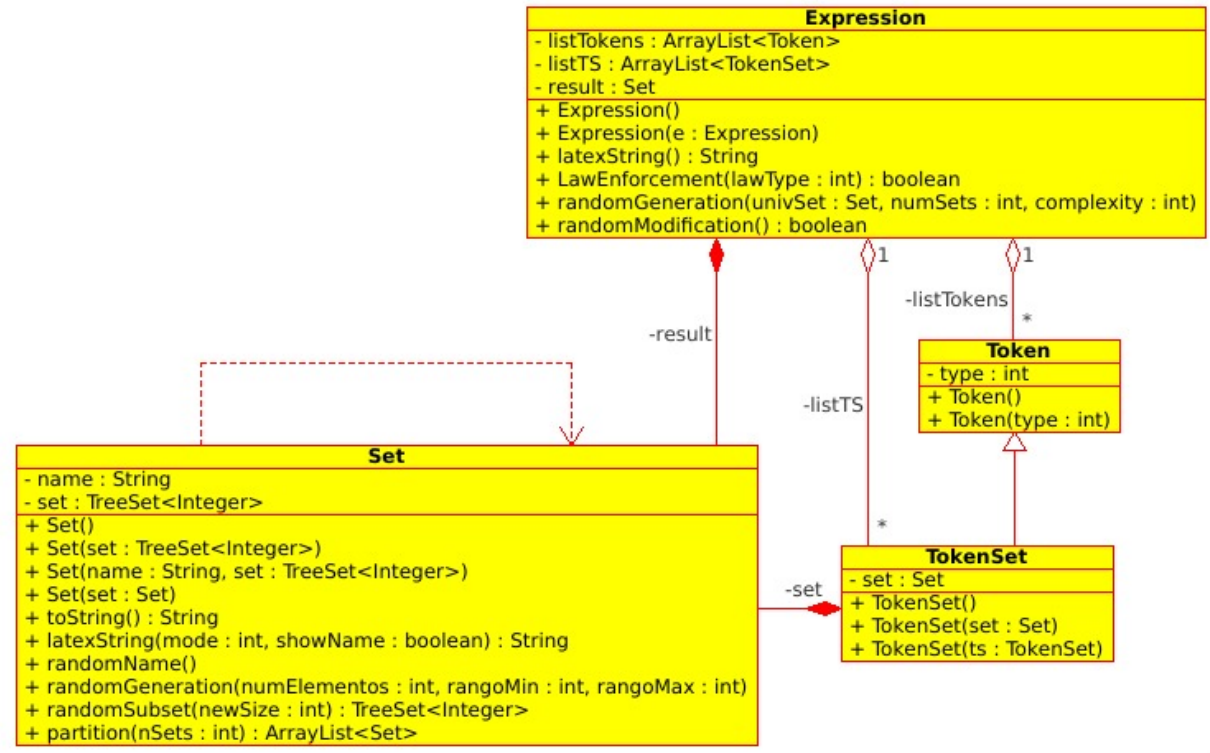


Figura 7.2: Diagrama de clases del paquete *settheory*.

Como se puede ver en la figura 7.2, la clase *es.uned.dia.aqg.settheory.Set* está basada en la clase *java.util.TreeSet* que es una clase miembro del Framework de Java *Collections* que implementa la interfaz *java.util.Set* que es la encargada de modelar la abstracción matemática de conjunto. La clase *java.util.TreeSet* aporta ordenación natural al conjunto además de garantizar tiempos de orden $log(n)$ para las operaciones básicas (añadir elemento al conjunto, eliminar elemento del conjunto y consultar pertenencia de un miembro al conjunto).

Pese a que la clase *es.uned.dia.aqg.settheory.Set* está basada en un conjunto ordenado de enteros, al representar el conjunto en LaTeX(mediante el método), los enteros se pueden sustituir por letras arábigas o griegas.

En la clase *es.uned.dia.aqg.settheory.Set* se han añadido funciones que no estaban imple-

mentadas en la interfaz *java.util.Set* que no obstante aportaba bastantes operaciones básicas de conjuntos tales como la unión (mediante el método *addAll(Collection<? extends E>c)*), la intersección (mediante el método *retainAll(Collection<?>c)*) y la sustracción (mediante el método *removeAll(Collection<?>c)*). Principalmente los métodos añadidos están relacionadas con la generación aleatoria, tanto de un nombre para el conjunto (mediante el método *randomName()*), como de los miembros del conjunto (mediante el método *randomGeneration(int numElementos, int rangoMin, int rangoMax)*), así como un método que suministra un subconjunto de elementos aleatorios del conjunto (mediante el método *randomSubset(int newSize)*). A parte de los métodos expuestos se ha incluido un método (*partition(int nSets)*) que suministra una lista de *nSets* subconjuntos del conjunto que cumplen las características del concepto de "partición de un conjunto".

La clase *es.uned.dia.aqg.settheory.Expression*, es la encargada de modelar la abstracción matemática de expresión algebraica de conjuntos, para ello se han tenido en cuenta conceptos del análisis sintáctico [Louden, 2004] (sin llegar a implementarse de forma completa), por lo que se han incluido las clases *es.uned.dia.aqg.settheory.Token* y *es.uned.dia.aqg.settheory.TokenSet* de cara a favorecer en el futuro la aplicación de esos conceptos.

La clase *es.uned.dia.aqg.settheory.Expression*, posee métodos destinados a la generación aleatoria de expresiones (*randomGeneration(Set univSet, int numSets, int complexity)*), así como modificaciones aleatorias dentro de las propias expresiones para generar otras expresiones que no sean equivalentes (método *randomModification()*). Por otro lado era necesario un método que facilitara la aplicación de las distintas propiedades básicas de conjuntos (método *LawEnforcement(int lawType)*).

7.2. Paquete *test*

El siguiente paquete que se va a exponer es *test*.

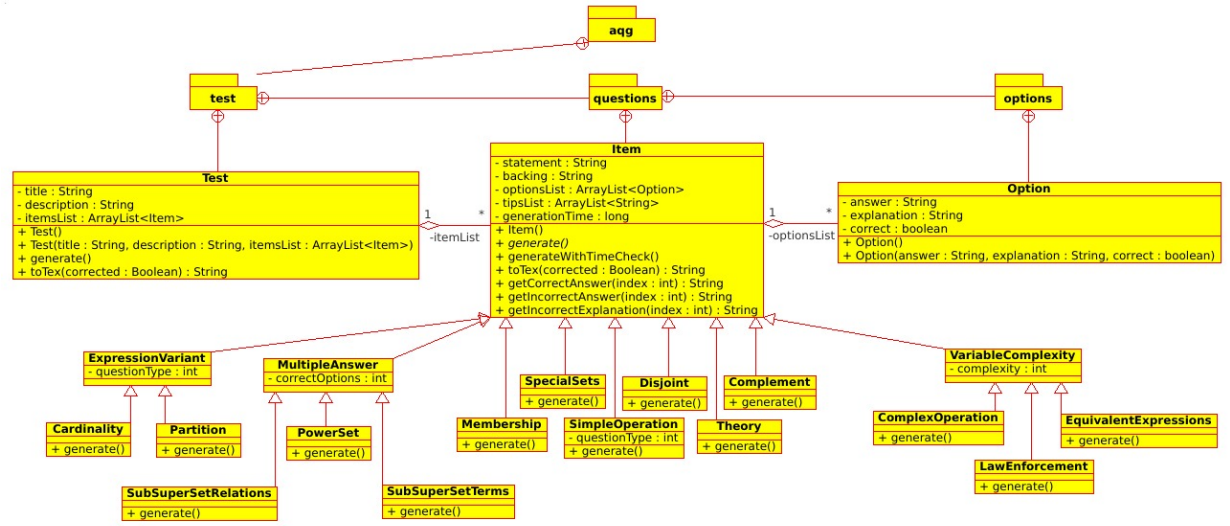


Figura 7.3: Diagrama de clases del paquete *test*.

Como se observa en la figura 7.3, se han diseñado tres entidades para implementar los conceptos analizados inicialmente. La clase *es.uned.dia.aqg.test.questions.Item*, finalmente se ha diseñado con la ayuda de la clase *es.uned.dia.aqg.test.questions.options.Option* que modela las posibles respuestas de una pregunta.

La clase *es.uned.dia.aqg.test.questions.Test* no es mucho más que una agrupación de preguntas con la capacidad de generar aleatoriamente esas preguntas a través del método *generate()*. Es una clase que no es necesaria para implementar los requisitos básicos del proyecto, pero debido a su sencillez y a que aportaba el valor añadido de permitir generar exámenes sin necesidad de contar con la plataforma Siette, se ha mantenido durante todo el desarrollo.

La clase *es.uned.dia.aqg.test.questions.Item* a parte de contar con métodos de acceso a sus datos (que no se muestran en el diagrama expuesto para mejorar la claridad del mismo) dispone de unos métodos que facilitan su utilización desde el sistema: *getCorrectAnswer(int index)*, *getIncorrectAnswer(int index)* y *getIncorrectExplanation(int index)*, que no son más que envoltorios que simplifican la forma de acceder a ciertos datos de uso frecuente en el sistema Siette.

Por otro lado la clase *es.uned.dia.aqg.test.questions.Item* define el método *generate()*

que será el encargado de generar los datos de las preguntas. Con este fin se han definido diferentes implementaciones de este método en las clases que heredan de *es.uned.dia.aqg.test.questions.Item*.

De forma genérica se han seguido dos algoritmos básicos para la generación de preguntas, por un lado el utilizado para las preguntas de respuesta simple y por otro el utilizado para las preguntas de respuesta múltiple, ambos se exponen a continuación.

Algorithm 1 Algoritmo de generación de pregunta de respuesta simple.

Require: El número de respuestas totales a generar (nOptions)

```
Generar enunciado de pregunta
Generar respuesta correcta
for i = 1 → (nOptions − 1) do
    unique ← true
    repeat
        Generar respuesta incorrecta
        for j = 0 → (i − 1) do
            if respuesta[j] = respuesta[i] then
                unique ← false
            end if
            j ← j + 1
        end for
    until unique = true
    i ← i + 1
end for
```

El algoritmo de generación de pregunta de respuesta simple produce preguntas con una respuesta correcta y el resto de respuestas incorrectas, un ejemplo de este tipo de pregunta se puede observar en la figura 7.4.

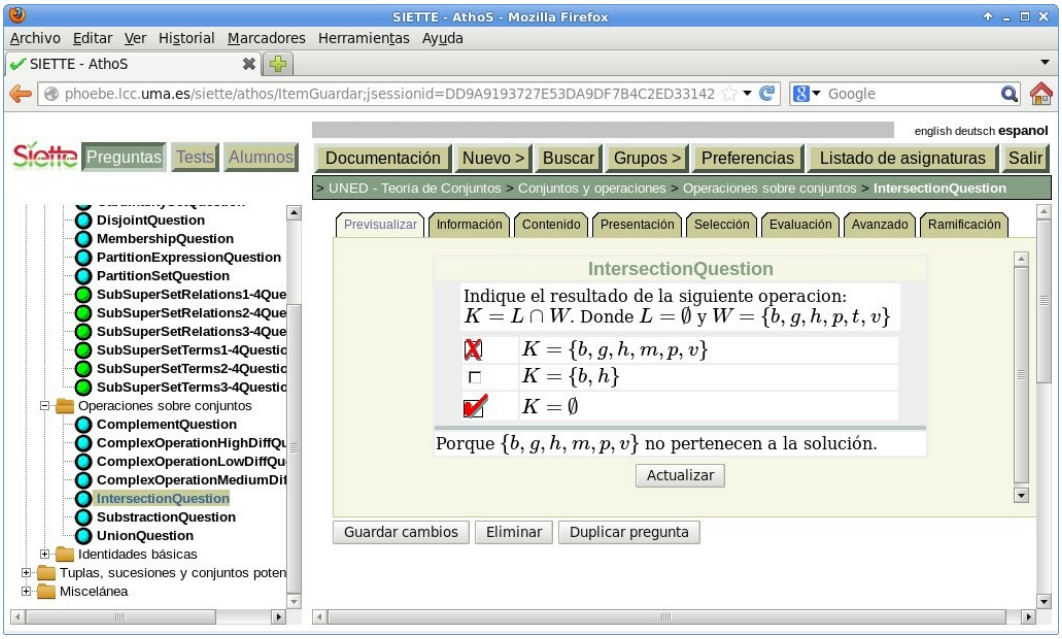


Figura 7.4: Ejemplo de pregunta de respuesta simple.

En la figura 7.4, se pueden observar algunas de las características que muestran las preguntas en Siette, se puede ver el enunciado en primer lugar, mostrando mediante la librería MathJax los elementos en LaTeX, a continuación se exponen las posibles opciones de respuesta y tras responder a la pregunta se muestra la opción correcta así como una explicación de la incorrección de la respuesta señalada en caso de ser incorrecta.

El algoritmo de generación de pregunta de respuesta múltiple produce preguntas con varias respuestas correctas y el resto de respuestas incorrectas, un ejemplo de este tipo de pregunta se puede observar en la figura 7.5.

El algoritmo de generación de respuesta múltiple se ha utilizado en preguntas de esta tipología específica que adicionalmente han contado con una jerarquía propia dentro de la arquitectura de herencia de la clase *es.uned.dia.aqg.test.questions.Item*, la clase *es.uned.dia.aqg.-test.questions.MultipleAnswer*.

Esta característica es la más distintiva dentro de la plataforma Siette, pues una de las divisiones principales que se hacen entre las plantillas de preguntas que se pueden generar es la que se define entre las preguntas de respuesta simple y las preguntas de respuesta múltiple como es este último caso.

Algorithm 2 Algoritmo de generación de pregunta de respuesta múltiple.

Require: El número de respuestas incorrectas a generar (correctOptions)

Require: El número de respuestas totales a generar (nOptions)

Generar enunciado de pregunta
for $i = 0 \rightarrow (correctOptions - 1)$ **do**
$unique \leftarrow true$
repeat
Generar respuesta correcta
for $j = 0 \rightarrow (i - 1)$ **do**
if $respuesta[j] = respuesta[i]$ **then**
$unique \leftarrow false$
end if
$j \leftarrow j + 1$
end for
until $unique = true$
$i \leftarrow i + 1$
end for
for $i = correctOptions \rightarrow (nOptions - 1)$ **do**
$unique \leftarrow true$
repeat
Generar respuesta incorrecta
for $j = 0 \rightarrow (i - 1)$ **do**
if $respuesta[j] = respuesta[i]$ **then**
$unique \leftarrow false$
end if
$j \leftarrow j + 1$
end for
until $unique = true$
$i \leftarrow i + 1$
end for

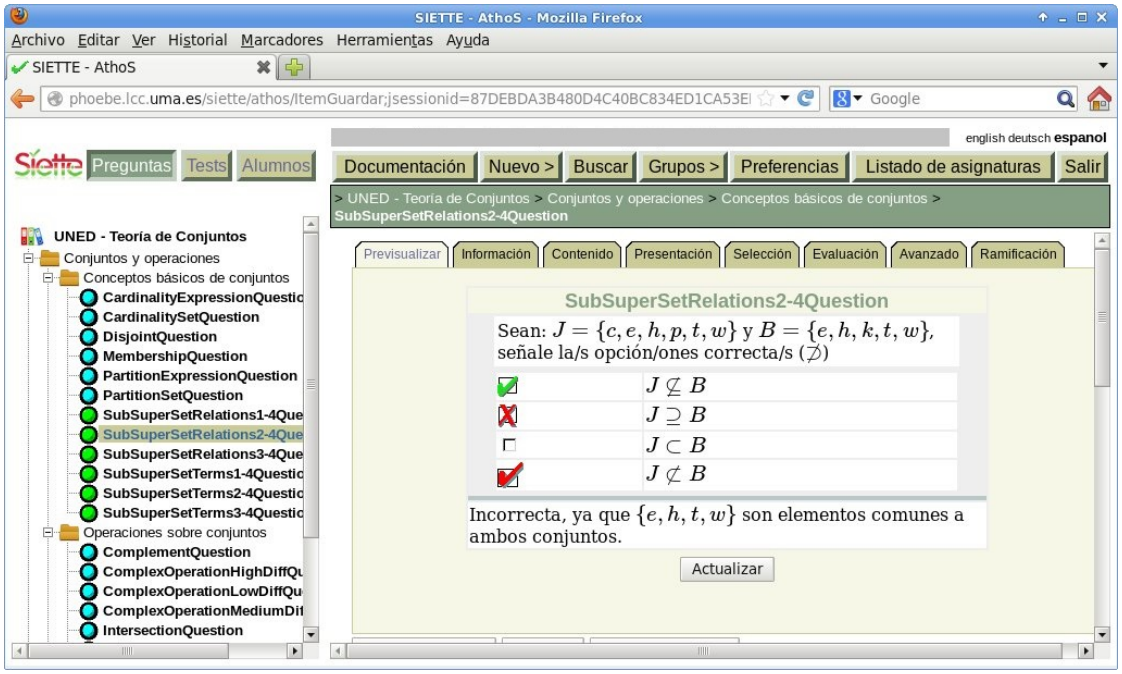


Figura 7.5: Ejemplo de pregunta de respuesta múltiple.

Igual que en la figura 7.4, en la figura 7.5 se pueden observar más características adicionales que muestran las preguntas en Siette en este caso para las preguntas de respuesta múltiple, se puede ver (como en las preguntas de respuesta simple) el enunciado, las posibles opciones de respuesta y tras responder a la pregunta se muestran la/s opción/es correcta/s así como una explicación de la incorrección en caso de fallo.

Como puede verse el código de iconos y colores que utiliza Siette para las correcciones es el siguiente:

Icono con una uve verde: La respuesta ha sido señalada y es correcta.

Icono con una equis roja: La respuesta ha sido señalada pero es incorrecta.

Icono con una uve roja: La respuesta no ha sido señalada y sin embargo era correcta.

A lo largo de las distintas fases de desarrollo se han ido identificando ciertas características comunes a algunas de las preguntas que se habían diseñado, por ello se han implementado tres clases intermedias para agrupar estas funcionalidades:

ExpressionVariant: Esta clase abstrae el comportamiento de aquellas preguntas (*Cardinality* y *Partition*) que pueden presentar dos variantes: una en la que se evalúa una propiedad de un conjunto y otra en la que se evalúa una propiedad del resultado que produce una expresión de conjuntos.

MultipleAnswer: Esta clase abstrae el comportamiento de aquellas preguntas (*PowerSet*, *SubSuperSetRelations* y *SubSuperSetTerms*) que tienen opciones de respuesta múltiple, es decir que puede haber más de una respuesta correcta para cada pregunta.

VariableComplexity: Esta clase abstrae el comportamiento de aquellas preguntas (*ComplexOperation*, *LawEnforcement* y *EquivalentExpressions*) que por tratarse de preguntas sobre expresiones, pueden presentar dificultad variable en función de la complejidad de la expresión que haya de ser evaluada.

7.3. Paquete *ui*

El último paquete que a exponerse es *ui*.

Figura 7.6: Diagrama de clases del paquete *ui*.

Al igual que la clase *es.uned.dia.aqg.test.questions.Test*, las clases implementadas en este paquete cumplen la función de presentar una interfaz (gráfica en el caso de *es.uned.dia.aqg.ui.Gui* y de consola en el caso de *es.uned.dia.aqg.ui.Tui*) para que un usuario pueda interactuar con la librería sin necesidad del sistema Siette.

Son clases muy sencillas, ya que las posibilidades ofrecidas por estas interfaces de usuario se reducen a ofrecer al usuario la oportunidad de especificar el número de preguntas que contendrá el fichero que se genere.

Puesto que no formaban parte de los requisitos solicitados no se ha dedicado mucho esfuerzo al desarrollo de estas clases en el sistema final, pero dado que fueron diseñadas e implementadas en el prototipo aritmético inicial, parece conveniente mantenerlas.

En especial se ha adoptado la postura de mantenerlas en el sistema final, como ya se ha comentado a lo largo de la memoria, porque aportaban una funcionalidad completa e interesante sin esfuerzo adicional.

7.4. Asignatura en Siette

Una parte de la implementación que no está directamente relacionada con la codificación y que por tanto no tiene cabida en ninguno de los paquetes expuestos anteriormente, es la creación de una asignatura en el sistema Siette que haga uso de la librería construida.

En este apartado se describirá brevemente el trabajo realizado.

La asignatura implementada se ha denominado: "UNED - Teoría de conjuntos". La implementación de la asignatura va más allá del simple hecho de hacer uso de la librería en el sistema Siette, hay que organizar y dar coherencia a las funcionalidades de la librería, para ello se ha definido la temática que se expone en la figura 7.7.

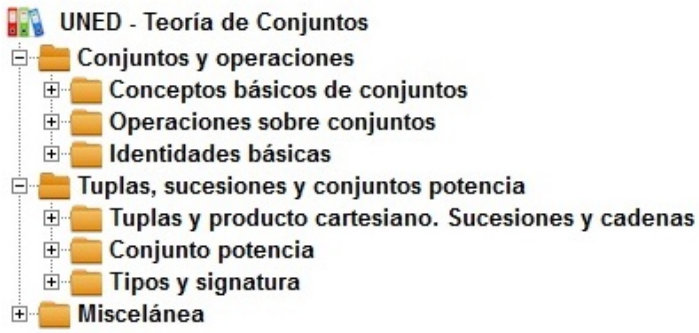


Figura 7.7: Asignatura "UNED - Teoría de conjuntos" (temas).

La división temática que se ha elegido está muy relacionada con el mapa conceptual de la asignatura abordada por el sistema ("Lógica y Estructuras Discretas") y además se ha procurado que los diferentes temas estuvieran lo más balanceados posibles en cuanto a

contenidos, si bien es cierto que aparentemente los temas iniciales parece que contienen un mayor número de preguntas, los conceptos abordados son más básicos que en el resto.

La distribución de las preguntas dentro de los diferentes temas se puede observar en la figura 7.8.

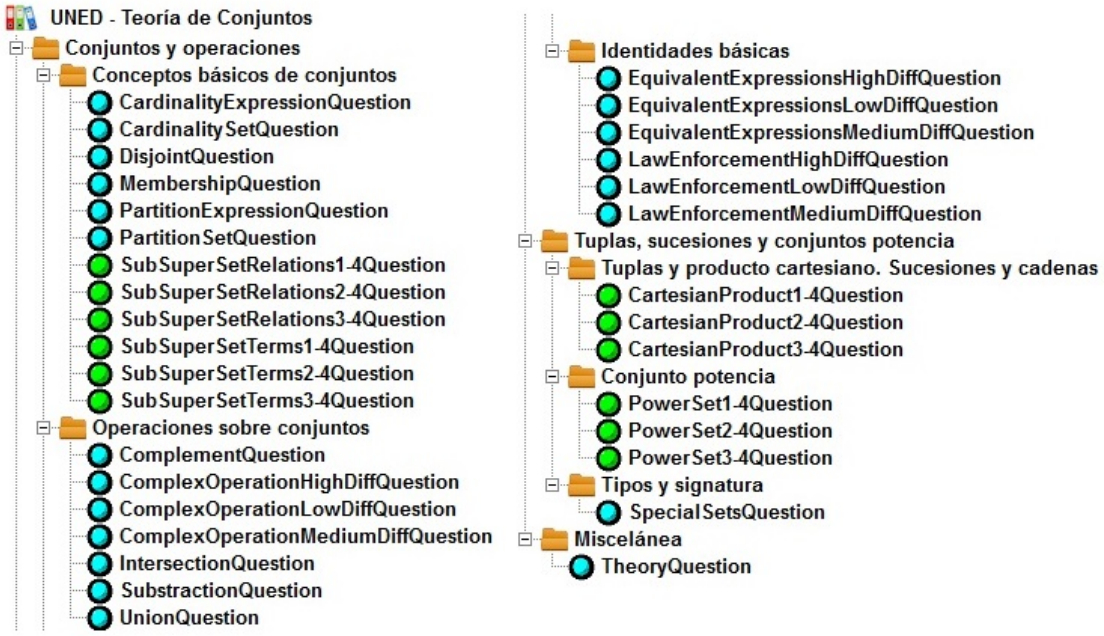


Figura 7.8: Asignatura “UNED - Teoría de conjuntos” (preguntas).

En la figura 7.8 se pueden identificar a simple vista aquellas preguntas que pertenecen a la tipología de preguntas con respuesta múltiple (que el sistema Siette identifica mediante el icono de la esfera verde al lado de la pregunta) y aquellas otras preguntas que se corresponden con la tipología de preguntas con respuesta simple (icono de esfera azul claro en el sistema Siette).

Así mismo pueden identificarse las preguntas que por su implementación poseen algún tipo de variante (dificultad variable, conceptos similares, . . .) y que en el sistema Siette deben de separarse en plantillas de preguntas diferentes.

Se ha procurado que los nombres de todas las preguntas implementadas poseyeran bastante información que describiese la pregunta: número de respuestas correctas en las preguntas de respuesta múltiple, dificultad específica de las preguntas de dificultad variable, tipos

de variante conjunto o expresión, ...

Es importante resaltar que el sistema Siette facilita en gran medida la creación de contenidos y por tanto el mayor esfuerzo en este apartado se pudo dedicar a la definición temática de la asignatura y a la implementación de las primeras preguntas.

Capítulo 8

Resultados y pruebas

Durante este proyecto se han desarrollado un conjunto de casos de prueba que permitieran verificar que la librería cumplía con los adecuados requisitos antes de realizar el despliegue en la plataforma web.

Estos casos de prueba se realizarán en la aplicación cuando se acabe su desarrollo para comprobar que la salida obtenida del sistema coincide con la esperada, no dando por finalizada la aplicación hasta que ambas salidas coincidan en todos los casos de prueba.

Es importante destacar que cada vez que se realice una modificación en el código de la aplicación deben volver a pasarse todos los casos de prueba por si acaso se ha modificado por error algo que afecte al comportamiento del sistema.

8.1. Pruebas unitarias

Para la realización de pruebas unitarias se ha utilizado JUnit, ya que es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

Se trata de un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada

se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

En la actualidad los entornos de desarrollo como Eclipse, cuentan con plug-ins que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas.

No obstante dado que la librería trabaja en gran medida con el azar, se reducen las posibilidades de habilitar pruebas unitarias que permitan verificar el correcto funcionamiento de la aplicación por lo que se han diseñado pruebas limitadas que verifiquen al menos que las plantillas que se utilizarán desde la plataforma web no sufren comportamientos inesperados de la librería.

Los resultados finales de las pruebas unitarias con JUnit (así como su implementación y scripts para ejecutarlas mediante ANT) se encuentran en el CD-ROM que acompaña esta memoria, por lo que pueden ser consultados allí, pero para mayor claridad, se incluye a continuación un breve resumen.

Tabla 8.1: Resumen de resultados de pruebas con Junit.

	Opciones correctas	**Opciones incorrectas**
Cardinality Expression	Superado	Superado
Cardinality Set	Superado	Superado
Disjoint	Superado	Superado
Membership	Superado	Superado
Partition Expression	Superado	Superado
Partition Set	Superado	Superado
SubSuperSetRelations 1-4	Superado	Superado

	Opciones correctas	Opciones incorrectas
SubSuperSetRelations 2-4	Superado	Superado
SubSuperSetRelations 3-4	Superado	Superado
SubSuperSetTerms 1-4	Superado	Superado
SubSuperSetTerms 2-4	Superado	Superado
SubSuperSetTerms 3-4	Superado	Superado
Simple Operation Complement	Superado	Superado
ComplexOperation Low Difficulty	Superado	Superado
ComplexOperation Medium Difficulty	Superado	Superado
ComplexOperation High Difficulty	Superado	Superado
Simple Operation Intersection	Superado	Superado
Simple Operation Susbtraction	Superado	Superado
Simple Operation Union	Superado	Superado
EquivalentExpressions Low Difficulty	Superado	Superado
EquivalentExpressions Medium Difficulty	Superado	Superado
EquivalentExpressions High Difficulty	Superado	Superado
LawEnforcement Low Difficulty	Superado	Superado
LawEnforcement Medium Difficulty	Superado	Superado
LawEnforcement High Difficulty	Superado	Superado
CartesianProduct 1-4	Superado	Superado
CartesianProduct 2-4	Superado	Superado
CartesianProduct 3-4	Superado	Superado
PowerSet 1-4	Superado	Superado
PowerSet 2-4	Superado	Superado

	Opciones correctas	Opciones incorrectas
PowerSet 3-4	Superado	Superado
SpecialSets	Superado	Superado
Theory	Superado	Superado

Como puede observarse todas las pruebas realizadas han sido superadas.

Las pruebas implementadas verificaban por un lado que el número de respuestas correctas generadas fuera el adecuado para la pregunta (existen preguntas que devuelven solo una respuesta correcta y otras que devuelven dos o tres respuestas correctas)y que el número de respuestas incorrectas también estuviera de acuerdo a las especificaciones de las preguntas (existen preguntas con opciones incorrectas que varían entre uno y tres).

Las pruebas unitarias se han verificado en múltiples ocasiones, ya que los modernos entornos de desarrollo tienen una gran integración con este tipo de pruebas, lo que facilita tanto su desarrollo como su utilización.

Cabe destacar que debido al amplio rango de resultados que puede ofrecer la librería, se han tenido que implementar un alto número de casos de prueba.

8.2. Pruebas de rendimiento

Se han realizado varios conjuntos de pruebas para comprobar el rendimiento general de la librería. En concreto se han realizado 10 conjuntos de 100 iteraciones cada uno, para verificar los tiempos medios de generación de preguntas aislándolos de los tiempos iniciales fuera de orden (debido a las operaciones de instanciación e inicialización).

Es importante indicar que las mediciones se han realizado fuera del sistema Siette para no interferir en el normal funcionamiento de la plataforma y tampoco se ha tenido en cuenta el tiempo de representación de la librería MathJax, ya que alterar los tiempos de funcionamiento de la librería MathJax no era uno de los requisitos de este proyecto, por otro lado, al ser un requisito de la plataforma Siette para poder representar símbolos matemáticos no se podía evitar su uso, por lo que no tenía sentido plantearse su rendimiento.

El conjunto completo de los datos obtenidos durante las pruebas de rendimiento finales se puede consultar en formato CSV en el CD-ROM que acompaña la presente memoria. Adicionalmente y al igual que se ha hecho con el resto de pruebas, el código de implementación de estas pruebas se encuentra en el mismo CD-ROM.

A continuación se muestran los datos estadísticos obtenidos tras las pruebas.

Tabla 8.2: Datos estadísticos de las pruebas de rendimiento.

	Sin datos iniciales		
	Media	**Desviación**	**Máximo**
Cardinality Expression	4,878	11,764	211,077
Cardinality Set	1,095	4,326	47,843
Disjoint	0,516	1,644	22,205
Membership	0,186	1,065	20,548
Partition Expression	1,644	5,140	77,147
Partition Set	0,385	1,366	20,927
SubSuperSetRelations 1-4	0,151	0,851	17,353
SubSuperSetRelations 2-4	0,327	2,925	62,250
SubSuperSetRelations 3-4	0,219	1,958	40,878
SubSuperSetTerms 1-4	0,579	7,262	218,818
SubSuperSetTerms 2-4	0,696	7,295	153,995
SubSuperSetTerms 3-4	0,511	5,343	149,897
Simple Operation Complement	0,171	1,061	17,815
ComplexOperation Low Difficulty	0,460	4,263	121,737
ComplexOperation Medium Difficulty	0,598	5,197	136,758
ComplexOperation High Difficulty	0,742	5,666	108,516

	Sin datos iniciales		
	Media	**Desviación**	**Máximo**
Simple Operation Intersection	0,277	2,129	55,864
Simple Operation Susbtraction	0,197	1,130	24,420
Simple Operation Union	0,566	5,576	134,742
EquivalentExpressions Low Difficulty	0,311	1,550	23,673
EquivalentExpressions Medium Difficulty	0,482	1,871	20,274
EquivalentExpressions High Difficulty	0,706	6,508	193,677
LawEnforcement Low Difficulty	0,134	0,758	18,600
LawEnforcement Medium Difficulty	0,260	1,471	26,319
LawEnforcement High Difficulty	0,585	6,682	189,404
CartesianProduct 1-4	0,232	1,240	22,808
CartesianProduct 2-4	0,248	1,904	48,266
CartesianProduct 3-4	0,234	2,167	57,981
PowerSet 1-4	0,202	2,260	66,859
PowerSet 2-4	0,209	1,408	32,963
PowerSet 3-4	0,124	1,152	22,129
SpecialSets	0,011	0,117	3,652
Theory	0,011	0,034	0,922

Puesto que, como se ha indicado previamente, los tiempos iniciales de cada conjunto de pruebas no se encontraban en la misma magnitud que el resto se ha decidido realizar un estudio separado de los mismos cuyos datos se muestran a continuación.

Tabla 8.3: Datos estadísticos de los casos iniciales de las pruebas de rendimiento.

	Datos iniciales		
	Media	**Desviación**	**Máximo**
Cardinality Expression	29,447	28,467	98,243
Cardinality Set	1,028	2,157	7,135
Disjoint	0,382	0,056	0,489
Membership	1,011	2,205	7,283
Partition Expression	0,875	0,273	1,394
Partition Set	0,418	0,424	1,569
SubSuperSetRelations 1-4	5,719	17,160	54,557
SubSuperSetRelations 2-4	0,078	0,021	0,116
SubSuperSetRelations 3-4	0,088	0,017	0,114
SubSuperSetTerms 1-4	0,284	0,154	0,717
SubSuperSetTerms 2-4	0,131	0,055	0,269
SubSuperSetTerms 3-4	0,119	0,023	0,153
Simple Operation Complement	0,644	1,279	4,282
ComplexOperation Low Difficulty	1,064	1,252	4,580
ComplexOperation Medium Difficulty	0,143	0,030	0,197
ComplexOperation High Difficulty	0,438	0,313	0,957
Simple Operation Intersection	4,173	12,152	38,759
Simple Operation Susbtraction	0,126	0,025	0,184
Simple Operation Union	0,216	0,315	1,105
EquivalentExpressions Low Difficulty	0,429	0,274	1,146

	Datos iniciales		
	Media	**Desviación**	**Máximo**
EquivalentExpressions Medium Difficulty	0,188	0,040	0,243
EquivalentExpressions High Difficulty	0,221	0,103	0,463
LawEnforcement Low Difficulty	0,224	0,045	0,316
LawEnforcement Medium Difficulty	0,105	0,019	0,137
LawEnforcement High Difficulty	0,226	0,112	0,445
CartesianProduct 1-4	0,272	0,077	0,471
CartesianProduct 2-4	0,121	0,051	0,211
CartesianProduct 3-4	0,463	0,476	1,259
PowerSet 1-4	0,896	2,058	6,746
PowerSet 2-4	0,096	0,028	0,146
PowerSet 3-4	0,078	0,048	0,205
SpecialSets	0,116	0,078	0,331
Theory	19,530	36,437	102,152

Pese a los visibles cambios de magnitud entre los distintos casos (iniciales y el resto) que pueden observarse entre las medias de las dos tablas, es curioso observar que los máximos se obtienen dentro de los casos no iniciales. Dado que los tiempos máximos suelen ser datos puntuales se han achacado a circunstancias puntuales del entorno de proceso, es importante recordar que se trata de un entorno afectado por un múltiples factores (máquina virtual Java ejecutándose en un sistema operativo que ejecuta muchos otros procesos y que además está virtualizado dentro de otro sistema operativo de escritorio que a su vez se encuentra ejecutando múltiples procesos) y que aunque se ha tratado de realizar las pruebas de manera que se minimice el impacto de los factores externos, es imposible eliminarlos todos.

En cualquier caso, si bien es cierto que se observa una desviación en los datos muy alta, en el conjunto de los datos (es posible verificarlo en los conjuntos de datos suministrados en

formato CSV) se puede apreciar que la tendencia es a mantenerse habitualmente en torno a la media.

Por otro lado es importante resaltar que las magnitudes de los datos (expresados en milisegundos) es bastante baja, por lo que cualquier factor interno puede hacer variar en gran medida los datos obtenidos.

Finalmente hay que señalar que se ha cumplido con creces el requisito de generar las preguntas en menos de un segundo, ya que el peor de todos los casos medidos ha sido de 218 milisegundos, obtenido en una pregunta de tipo SubSuperTerms con una respuesta correcta de cuatro posibles. Y aunque este dato ya de por sí es prácticamente un 80 % inferior al solicitado los resultado apenas llegan al milisegundo de media, siendo este un gran logro por parte de la librería desarrollada.

8.3. Pruebas de simulación

Dado que la plataforma web que finalmente utilizaría la librería no tenía posibilidad de replicarse en un entorno controlado para poder realizar pruebas fuera del entorno final, se desarrolló una página en JSP que permitiese simular el comportamiento de la plataforma web.

Para ello se habilitó un servidor de aplicaciones Java Apache Tomcat en su versión 7.0 y se desplegó la página JSP junto con la librería generada, de manera que fuera posible verificar el comportamiento funcional de la librería sin necesidad recurrir a la plataforma web.

Para acercar lo máximo posible el comportamiento de la simulación al del sistema final, la página JSP cuenta con el motor JavaScript MathJax que es el mismo con el que cuenta Siette.

En la figura 8.1, se muestra el comportamiento de la página JSP que sirve para simular la funcionalidad del sistema Siette.

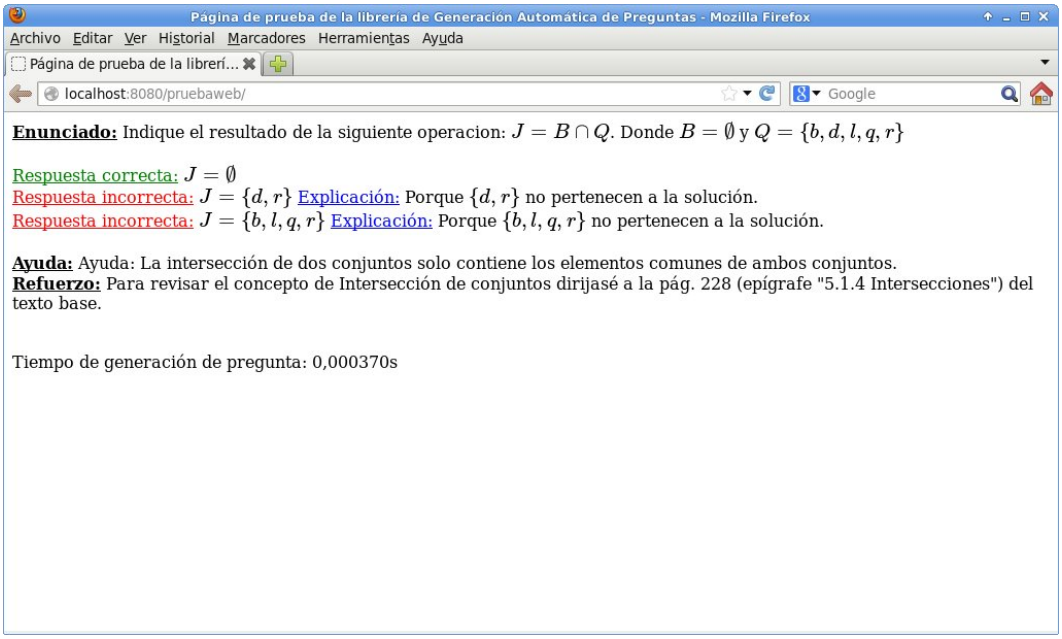


Figura 8.1: JSP de pruebas funcionales de simulación.

Dado que no aporta demasiado valor exponer en este punto el código de la página, se ha incluido en el CD-ROM que acompaña la memoria por si desea ser consultado, pero si que merece la pena comentar algunos de los aspectos más relevantes que se han contemplado, ya que ha sido necesario diseñar la página de prueba de manera que mostrase la máxima información posible de cada pregunta (enunciado, opciones de respuesta, ayudas, etc...) pero de una forma muy flexible ya que dependiendo de las preguntas, el número de opciones correctas e incorrectas puede variar. Así mismo se ha utilizado un diseño básico y sencillo que facilite la lectura de los resultados con textos resaltados y coloreados para que un primer vistazo ya proporcione bastante información.

Capítulo 9

Conclusiones

Este capítulo contiene las resoluciones obtenidas sobre este proyecto fin de carrera una vez que su realización ha finalizado. Estas resoluciones o conclusiones se muestran a continuación:

- Se ha cumplido con todos los objetivos marcados al inicio del proyecto, consiguiendo de esta manera desarrollar una sistema completo y funcional.
- Se ha conseguido hacer un profuso y extensivo uso de un gran número de las asignaturas de la carrera poniendo en práctica muchos de los conocimientos adquiridos durante el proceso formativo.
- Se ha empleado un tiempo notable en la implementación y realización de pruebas que han sido un pilar básico durante todo el desarrollo como bien se había estudiado. De hecho se ha convertido en una fase completamente diferenciada que tomaba relevancia por sí misma.
- Se ha trabajado con éxito con un sistema real externo (el sistema Siette) que es una de las circunstancias más comunes en el mundo profesional actual, experimentando de esta manera las problemáticas y soluciones que este tipo de contexto implica.

No obstante durante todo el desarrollo del proyecto se han encontrado dificultades y peculiaridades que también merece la pena reseñar:

- Las particularidades de la educación a distancia, específicamente estudiadas en el proyecto por la naturaleza del sistema, han marcado la realización del proyecto.

  En experiencias previas en el desarrollo de un proyecto de fin de carrera de manera presencial (para la Ingeniería Técnica), la interacción con la dirección del proyecto era más fácil de conseguir, pero en este caso es importante resaltar que los esfuerzos por todas las partes (dirección de proyecto, autoría de proyecto y soporte para de la plataforma Siette) han reducido todos los inconvenientes y es una experiencia que contribuye a fomentar la educación a distancia.

- Debido a las peculiaridades del sistema (no es una aplicación con interfaz de usuario, no accede a base de datos, se debe integrar con otro sistema, ...) en un primer momento no se supo adecuar el desarrollo a un patrón tan extendido como es el MVC (Model View Controller), no obstante se ha tenido en mente durante toda la realización del mismo la separación conceptual que implica el citado patrón. En cualquier caso a lo largo del avance del proyecto si que se detectó que la aplicación del patrón podría mejorar el sistema (sobre todo para hacerlo más extensible) pero debido a lo avanzado del proyecto no se pudo abordar y se ha marcado como una de las posibles líneas de trabajo futuro.

- Es de destacar el hecho de que el proyecto posee un alto componente creativo que ha sido un reto notable tanto en el desarrollo, como en las estimaciones que se hacían sobre el avance del proyecto. Como se ha comentado en la memoria, el diseño de las preguntas dentro del dominio de la Teoría de conjuntos en ocasiones requería más esfuerzo que la traslación del diseño a lenguaje de codificación y viceversa. En este aspecto la dirección del proyecto (especializada en la materia) ha jugado un papel positivamente decisivo al apoyar en todo momento todo el proceso de desarrollo.

- Se ha comprobado que ante cualquier desarrollo software, por muy sencillo que parezca, es necesario realizar un estudio profundo de los requisitos y que aunque estos parezcan claros y el solicitante haga un esfuerzo notable por trasladarlos al lenguaje técnico de la informática, siempre existen dificultades a la hora de trasmitir ideas,

especialmente cuando estas forman parte de otro dominio como en este caso ha sido la "Teoría de conjuntos". Las dificultades preliminares provocadas por la falta de conocimiento inicial se van eliminando según avanza el proyecto y se va adquiriendo conocimiento en el dominio sobre el que se está trabajando.

- Cabe resaltar que los conocimientos adquiridos durante la formación académica han sido vitales a la hora de llevar a cabo este proyecto y que de hecho ha sido necesario reducir el enfoque inicial dado al trabajo ya que el tiempo era limitado y los recursos finitos y eso obliga en todos los casos a una labor de focalización del esfuerzo, para evitar que la magnitud de la que se quiere dotar al proyecto lo haga inabordable. Por ello se incluyen, más adelante, unas líneas de trabajo futuras que recogen ideas que se han ido madurando a lo largo de este trabajo.

- El autor del presente proyecto desarrolla su actividad profesional en el ámbito de la administración de sistemas, lo cual supone un punto de vista radicalmente distinto al necesario para el desarrollo de este proyecto. Por un lado esto supone una refrescante experiencia que enriquece la formación del autor y por otro supone una drástica ruptura con su habitual forma de trabajo. Dada la diversidad actual de disciplinas que engloba la Ingeniería Informática, quizás convendría modificar la orientación de la oferta de proyectos de fin de carrera abriéndola, en la medida de lo posible, a ámbitos que se salgan del terreno del desarrollo (estudios de rendimiento, definición de arquitecturas de despliegue, desarrollos con tecnologías móviles, ...).

- Otro aspecto destacado de este proyecto ha sido el trabajo con una oferta de soluciones tecnológicas extraordinariamente diversas, esto es, para cada una de las tareas a realizar (documentación, codificación, pruebas, ...) el abanico de herramientas a utilizar es muy amplio, obligando a realizar estudios de adecuación de soluciones que suponen esfuerzos adicionales a los del propio desarrollo. Actualmente es la propia tecnología la que en su constante y vertiginosa evolución contribuye a desaprovechar y diluir recursos. Quizás merezca la pena racionalizar el avance tecnológico en aras de poder realizar un uso más eficiente de las propias mejoras que conlleva.

Capítulo 10

Trabajo futuro

En este capítulo se explican algunos de los posibles caminos que se pueden seguir para ampliar el trabajo realizado hasta el momento.

- La línea más obvia de trabajo para continuar con el proyecto, es la ***ampliación de la temática de las preguntas a generar***. Actualmente se han cubierto aspectos relativos a "Conjuntos y operaciones" y "Tuplas, sucesiones y conjuntos potencia", si se desea continuar con el enfoque actual de seguir el hilo conceptual de la asignatura "Lógica y estructuras discretas", lo más natural sería optar por abordar los apartados "Relaciones" y "Relaciones binarias: propiedades", aunque también se puede preferir acometer el apartado de "Funciones".

- Por otro lado, y de cara a la generalización de la librería, merecería la pena ***aplicar algunos patrones de diseño*** que si bien inicialmente suponen un ligero esfuerzo adicional en cuanto al desarrollo, podrían ayudar a la extensibilidad de la librería, así como a estandarizar su diseño. Algunos de estos patrones a modo de ejemplo podrían ser:

  MVC (Model View Controller): Para separar datos, de interfaz de usuario y de la lógica de negocio.

  Abstract Factory: Para gestionar por ejemplo distintas salidas de la librería (fichero, web, pdf, etc...).

Factory Method: Para facilitar la creación de los diversos tipos de preguntas.

- En relación al último punto expuesto quedaría la posibilidad de ***definir una arquitectura común*** para ser reutilizada en otros proyectos de esta misma índole. Consiguiendo así una única librería desarrollada modularmente, con una interfaz de trabajo propia y múltiples posibilidades de uso. Como ya se ha comentado a lo largo del proyecto, las preguntas suelen ser bastante independientes entre sí, por lo que se pueden desarrollar por separado con bastante facilidad. Existen dos vertientes en las que se podría avanzar en esta dirección:

  - Modificación de la librería para que acepte el uso de librerías externas que aborden otros dominios de conocimiento.
  - Creación de un módulo que permita generar preguntas sin necesidad de tener conocimientos de programación.

  Este mismo proyecto podría servir como base inicial, consiguiendo una librería que actuase como Base de conocimiento (Knowledgebase) de preguntas de asignaturas de la UNED. Para ello sería conveniente usar una ontología para especificar su estructura y su esquema de clasificación.

- Uno de los aspectos que sería interesante mejorar es el ***análisis tanto sintáctico como semántico de las expresiones con conjuntos***. Ya que, si bien se han sentado las bases para emprender esta labor, aún quedan múltiples posibilidades por explotar aplicando conceptos vistos en "Procesadores de lenguaje" [Louden, 2004].

- Una característica a considerar es la posibilidad de ***añadir opciones de configuración desde fuentes externas*** (XML, ficheros de propiedades, bases de datos, ...) que ampliarían las posibilidades de la librería sin necesidad de realizar cambios en el código (y por tanto sin necesidad de conocimientos informáticos). Las opciones de configuración pueden ir desde nuevos conjuntos de elementos, opciones de configuración de preguntas, datos de preguntas, etc...

Bibliografía

[Agustín et al., 2002] Agustín, G. C., Seco, A. D. A., Gilabert, T. S. F., and Villalón, J. A. C.-M. (2002). *Gestión del Proceso Software*. Centro de Estudios Ramón Areces.

[Damato and Neidhardt, 2012] Damato, A. and Neidhardt, P. (2012). Latex. Web. `http://en.wikibooks.org/wiki/LaTeX/`.

[de Tecnologías Educativas y de Formación del Profesorado, 2012] de Tecnologías Educativas y de Formación del Profesorado, I. N. (2012). Educación presencial y a distancia. Web. `http://www.ite.educacion.es/formacion/materiales/90/cd_09/cursofor/cap_1/cap1a.htm`.

[Humphrey, 2001] Humphrey, W. S. (2001). *Introducción al Proceso Software Personal (PSP)*. Addison Wesley.

[León, 1997] León, A. C. (1997). *Psicología general: Memoria, pensamiento y lenguaje*. Centro de Estudios Ramón Areces.

[Lipschutz, 1992] Lipschutz, S. (1992). *Teoría de conjuntos y temas afines*. McGraw-Hill.

[Louden, 2004] Louden, K. C. (2004). *Construcción de compiladores: principios y práctica*. Paraninfo.

[Manzano and Sánchez, 2004] Manzano, M. and Sánchez, M. A. H. (2004). *Lógica para principiantes*. Alianza Editorial.

[Mira et al., 2003] Mira, J., Delgado, A. E., Boticario, J. G., and Díez, F. J. (2003). *Aspectos Básicos de la Inteligencia Artificial*. Sanz y Torres.

[Oracle, 2012] Oracle (2012). Java^{tm} platform, standard edition 6 api specification. Web. `http://docs.oracle.com/javase/6/docs/api/overview-summary.html`.

[Pressman, 1997] Pressman, R. S. (1997). *Ingenieria del software: un enfoque practico*. McGraw-Hill.

[Sánchez and Arjona, 2002] Sánchez, A. H. and Arjona, M. M. (2002). Teoría de conjuntos. Web. `http://www.ucm.es/info/pslogica/teoriaconjuntos.pdf`.

[Tremblay and Grassmann, 2010] Tremblay, J. P. and Grassmann, W. K. (2010). *Matemática discreta y lógica*. Prentice Hall.

[UNED, 2012] UNED (2012). Manuales para las aplicaciones. Web. `http://portal.uned.es/portal/page?_pageid=93,23523312&_dad=portal&%_schema=PORTAL`.

Listado de siglas, abreviaturas y acrónimos

AIESAD Asociación Iberoamericana de Educación Superior a Distancia

AQG Automatic Question Generation, Generación automática de preguntas

COCOMO COnstructive COst MOdel, Modelo Constructivo de Coste

CORBA Common Object Request Broker Architecture

CSV Comma Separated Value

CTT, TCT Clasic Test Theory, Teoría Clasica de Test

EEES Espacio Europeo de Enseñanza Superior

GNU GNU no es Unix

GUI Graphical User Interface, Interfaz gráfica de usuario

IDE Integrated Development Environment, Entorno de Desarrollo Integrado

JAR Java ARchive

JCP Java Community Process

JLS Java Language Specification

JSP Java Server Pages

JSRs Java Specification Requests

JVM, MVJ Java Virtual Machine, Máquina Virtual de Java

LMS, SGCE Learning Management System, Sistema Gestor de Contenidos Educativos

MVC Model View Controller, Modelo Vista Controlador

OSGi Open Services Gateway Initiative

PDF Portable Document Format, Formato de Documento Portátil

STI Sistema Tutor Inteligente

TAI Tests Adaptativos Informatizados

TRI Teoría de Respuesta al Ítem

TUI Text-based User Interface, Interfaz de usuario basada en texto

UML Unified Modeling Language, Lenguaje Unificado de Modelado

UNED Universidad Nacional de Educación a Distancia

WORA Write Once, Run Anywhere, Escribir una vez, ejecutar en cualquier lugar

WYSIWYG What You See Is What You Get, Lo que ves es lo que obtienes

WYSIWYM What You See Is What You Mean, Lo que ves es lo que quieres decir

XML eXtensible Markup Language, Lenguaje de Marcas Extensible

Apéndices

Apéndice A

Análisis Preliminar

En este apéndice se incluye el documento del análisis preliminar generado durante la fase de análisis del desarrollo de la librería.

Análisis preliminar de documentación

para el proyecto:

Implementación, despliegue y estudio de la eficiencia de una librería de generación automática de preguntas

Autor: Juan Manuel Sánchez Turrión

Índice general

Capítulo 1

Equivalencias Temáticas

1.1. Equivalencias del mapa conceptual con el libro de texto:

Los conceptos con identificador mayor que 9 no figuran en el mapa conceptual pero han sido igualmente considerados:

Id.	Concepto	Epigrafe	Página
	1.1.1 Conceptos básicos de Conjuntos	5.1 Conjuntos y operaciones de conjuntos	223
1	Definición	5.1.2 Los conjuntos y sus miembros	224
2	Pertenencia	5.1.2 Los conjuntos y sus miembros	224
3	Sub/Super conjunto	5.1.3 Subconjuntos	226
4	Cardinalidad de un conjunto	5.1.3 Subconjuntos	228
5	Conjunto universal	5.1.2 Los conjuntos y sus miembros	224
10	Igualdad de conjuntos	5.1.2 Los conjuntos y sus miembros	225
11	Subconjunto propio	5.1.3 Subconjuntos	227
12	Conjunto vacio	5.1.3 Subconjuntos	227

	1.1.2 Operaciones sobre conjuntos	5.1 Conjuntos y operaciones de conjuntos	228
6	Intersección	5.1.4 Intersecciones	228
7	Unión	5.1.5 Uniones	229
8	Diferencia	5.1.6 Diferencias y complementos	231
9	Complemento	5.1.6 Diferencias y complementos	232
13	Conjuntos disjuntos	*	*
14	Partición de un conjunto	5.4.7 Relaciones de equivalencia	263

El concepto 13 “Conjuntos disjuntos” no aparece en el libro de texto consultado.

Conjuntos Disjuntos: Aquellos que no tienen ningún elemento en común . Por ejemplo, {1, 2, 3} y {4, 5, 6}. Formalmente, dos conjuntos A y B son disjuntos si su intersección es el conjunto vacío.

1.2. Equivalencias del test de ejemplo (Modelo Anticipado 2011-12) con el mapa conceptual:

Pregunta	Respuesta	Id. Conceptos	Notas
1	a	9, 7, 3	
1	b	2, 6	
1	c	6, 3	
1	d	6, 1	La igualdad de conjuntos se define en el axioma de extensionalidad
2	a	6, 7, 1	La igualdad de conjuntos se define en el axioma de extensionalidad
2	b	9, 7, 6, 1	La igualdad de conjuntos se define en el axioma de extensionalidad
2	c	7	Incluye el concepto conj. potencia que se analiza en epígrafes posteriores
2	d	3	Incluye el concepto de subconjunto propio

Las preguntas 3 y 4 pertenecen al apartado "5.3 Relaciones".

La pregunta 5 hace alusión al Id. Concepto 4 (Cardinalidad de un conjunto), pero introduce en las respuestas los factoriales para su cálculo.

1.3. Equivalencias del test de ejemplo (Febrero 2011 A) con el mapa conceptual:

Pregunta	Respuesta	Id. Conceptos	Notas
1	a	2	
1	b	2, 12	
1	c	12, 11	Incluye el concepto conj. potencia que se analiza en epígrafes posteriores
2	a	9, 6	El enunciado hace referencia a los conceptos 7 y 9
2	b	9, 7	
2	c	6	
4	a	9	
4	b	12	
4	c		El enunciado hace referencia a los conceptos 6, 7 y 9

Las preguntas 3 y 5 pertenecen al apartado "5.3 Relaciones".

1.4. Equivalencias del test de ejemplo (Febrero 2011 B) con el mapa conceptual:

El examen de tipo B es igual que el de tipo A, pero con otro orden de preguntas, la equivalencia de preguntas se señala a continuación:

Preguntas A	Preguntas B
1	2
2	4
3	5
4	1
5	3

1.5. Equivalencias del test de ejemplo (Septiembre 2011 A) con el mapa conceptual:

Pregunta	Respuesta	Id. Conceptos	Notas
11	*	3	Incluye el concepto conj. potencia que se analiza en epígrafes posteriores

Las preguntas 10, 12, 13 y 14 pertenecen al apartado "5.3 Relaciones".

Este examen es completamente distinto de los analizados hasta ahora, las preguntas relacionadas con los epígrafes analizados se encuentran hacia el final del examen y no hay ninguna que se centre exclusivamente en los epígrafes iniciales.

1.6. Equivalencias del test de ejemplo (Septiembre 2011 B) con el mapa conceptual:

Pregunta	Respuesta	Id. Conceptos	Notas
10	*	3	Incluye el concepto producto cartesiano que se analiza en epígrafes posteriores
11	*	4	Incluye el concepto conj. potencia que se analiza en epígrafes posteriores

Las preguntas 12, 13 y 14 pertenecen al apartado "5.3 Relaciones".

A diferencia de los exámenes vistos hasta el momento no existe relación entre las preguntas del examen de tipo A y el de tipo B.

1.7. Equivalencias del test de ejemplo (Febrero 2012 A) con el mapa conceptual:

Pregunta	Respuesta	Id. Conceptos	Notas
1	a	2, 7	
1	b	3, 6	
1	c	3, 6	
1	d	3	
2	a	6, 7, 10	
2	b	6, 9, 10	
2	c	9, 10	
2	d	*	Incluye el concepto conj. potencia que se analiza en epígrafes posteriores
3	a	6, 10	Introduce el superíndice "signo menos"
3	b	3, 7	Introduce el superíndice "signo menos"
3	c	2	Introduce el operador "círculo"
3	d	*	Hace referencia al apartado "5.3 Relaciones"

Las preguntas 4, 5 y 6 pertenecen al apartado "5.3 Relaciones".

1.8. Equivalencias del test de ejemplo (Febrero 2012 B) con el mapa conceptual:

El examen de tipo B es igual que el de tipo A, pero con otro orden de preguntas, la equivalencia de preguntas se señala a continuación:

Preguntas A	Preguntas B
1	3*
2	1
3	2
4	4
5	5
6	6

* Adicionalmente al cambio de orden de preguntas, en este caso concreto se cambia también el orden de las opciones de respuesta.

1.9. Equivalencias del test de ejemplo (Septiembre 2012 A) con el mapa conceptual:

Pregunta	Respuesta	Id. Conceptos	Notas
9	a	5	
9	b	12	
9	c	6, 9, 12	El enunciado hace referencia a los conceptos 10 y 6
11	*	3	Incluye el concepto producto cartesiano que se analiza en epígrafes posteriores
13	*	3	Incluye el concepto conj. potencia que se analiza en epígrafes posteriores

Las preguntas 8, 10 y 12 pertenecen al apartado "5.3 Relaciones".

1.10. Equivalencias del test de ejemplo (Septiembre 2012 B) con el mapa conceptual:

Existen ciertas similitudes entre el examen de tipo A y el de tipo B, la equivalencia de preguntas encontrada se señala a continuación:

Preguntas A	Preguntas B
9	12
11	11
12	13
13	8

No obstante, las preguntas 8 y 10 (que no tienen equivalencia con el examen de tipo A) pertenecen al apartado "5.3 Relaciones".

Capítulo 2

Análisis de preguntas de ejemplos

Análisis de la primera pregunta del test de ejemplo (Modelo Anticipado 2011-12):

1. Elegir operador de "comparación" (pertenencia, subconjunto o igualdad)
2. Generar el operando izquierdo en función del operador (conjunto o elemento)
3. En el caso de que el operando izquierdo sea un conjunto, generar el conjunto mediante la aplicación de operaciones estudiadas (unión, intersección y complemento, no se utiliza la operación diferencia)
4. Se genera el operando derecho, siguiendo la misma algoritmia que en el paso 3

Capítulo 3

Propuestas de preguntas a generar

Preguntas sencillas:

1. Identifique el conjunto universal

    a) Se genera un conjunto universal (letras o números) de cardinalidad configurable, se le asigna una posición de respuesta al conjunto generado, se eliminan recursivamente elementos del conjunto generado y se les asignan posición de respuesta a los nuevos subconjuntos hasta que se completan todas las opciones.

    b) Se genera un conjunto universal (letras o números) de cardinalidad configurable, se le asigna una posición de respuesta al conjunto generado, se eliminan al azar elementos del conjunto generado, se comprueba que el nuevo conjunto no esta asignado a ninguna posición de respuesta, se le asigna una posición de respuesta, se iteran los dos últimos pasos hasta que se completan todas las posiciones de respuesta.

2. A que conjunto pertenece un elemento

    a) Se genera un elemento al azar (letra o número) para incluirlo en la pregunta, se genera un conjunto aleatorio (de cardinalidad configurable) que posea el elemento generado al azar inicialmente, se le asigna una posición de respuesta al conjunto generado, se generan conjuntos aleatorios (de cardinalidad configurable) que no posean el elemento generado al azar, se le asigna una posición de respuesta al conjunto generado, se iteran los dos últimos pasos hasta que se completan todas las posiciones de respuesta.

Apéndice B

Manual de usuario

En este capítulo se ofrecen unas guías de la forma en la que se puede utilizar la librería desarrollada.

Como ya se indicaba en 1.4, se puede encontrar material referente al presente capítulo en el espacio web habilitado para complementar la presente memoria:
`https://sites.google.com/site/aqgpfc/`

La finalidad con la que se ha desarrollado la librería es ser utilizada desde la plataforma Siette en forma de item generativo JSP, que es un tipo de pregunta basada en plantillas con capacidad para utilizar código JSP (desde el que se puede acceder a métodos de una librería JAR que es lo que se pretende). No obstante durante el desarrollo del proyecto se ha encontrado útil desarrollar ciertas interfaces de usuario que permitan hacer uso de la librería sin necesidad de tener acceso a Internet, la forma de utilizar estas interfaces se detalla en el sub-apartado siguiente.

B.1. Uso de la librería desde la plataforma Siette

Previamente a dar comienzo a su utilización, la librería debe de estar correctamente desplegada en el sistema Siette, para ello se pueden seguir las instrucciones indicadas en el Apéndice C.

Figura B.1: Página de inicio de Siette.

En primer lugar es necesario autenticarse* en el sistema a través de su pantalla inicial, mostrada en la figura B.1.

Una vez dentro de la plataforma web la primera página que se muestra es la principal que lista las asignaturas disponibles. Desde ella habrá de seleccionarse la asignatura en la que se va a incluir el item generativo JSP que va a acceder a la librería desarrollada en este proyecto.

Desde la página de gestión de la asignatura elegida, se puede elegir un tema (se encuentran en el árbol de la izquierda) de la asignatura que será al que se asigne la pregunta a crear.

Una vez seleccionado el tema en el que se ubicará finalmente la pregunta/item, se puede proceder a la generación del mismo a través del botón *"Nuevo >"*, que desplegará un sub-menú con las opciones *"Nuevo tema"* y *"Nuevo pregunta"*, se debe seleccionar *"Nuevo pregunta"*, lo cual desplegará un nuevo sub-menú con las opciones que pueden verse en la ficgura B.4.

De las opciones mostradas, las desde las que es más adecuado utilizar la librería son las

*Las credenciales utilizadas para acceder al sistema deberán de tener privilegios de profesor como mínimo.

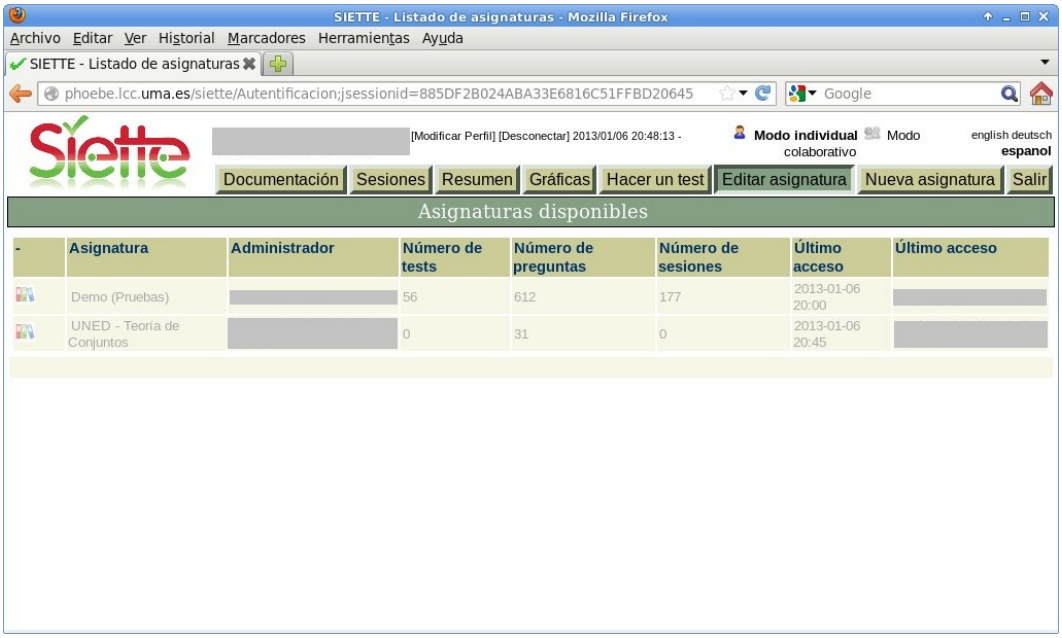


Figura B.2: Página principal de Siette.

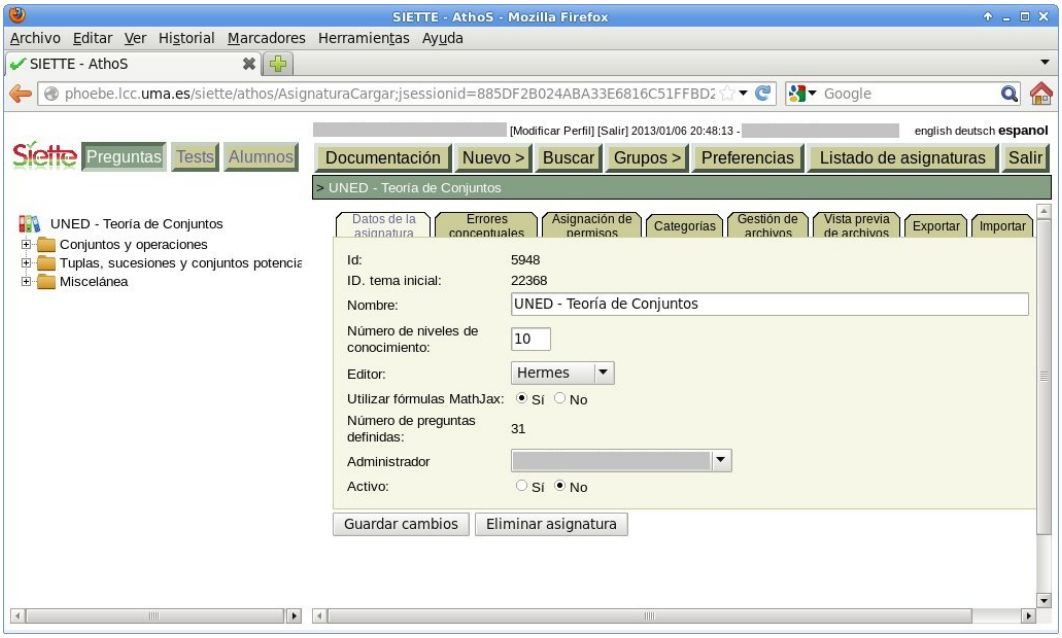


Figura B.3: Página de gestión de asignatura.

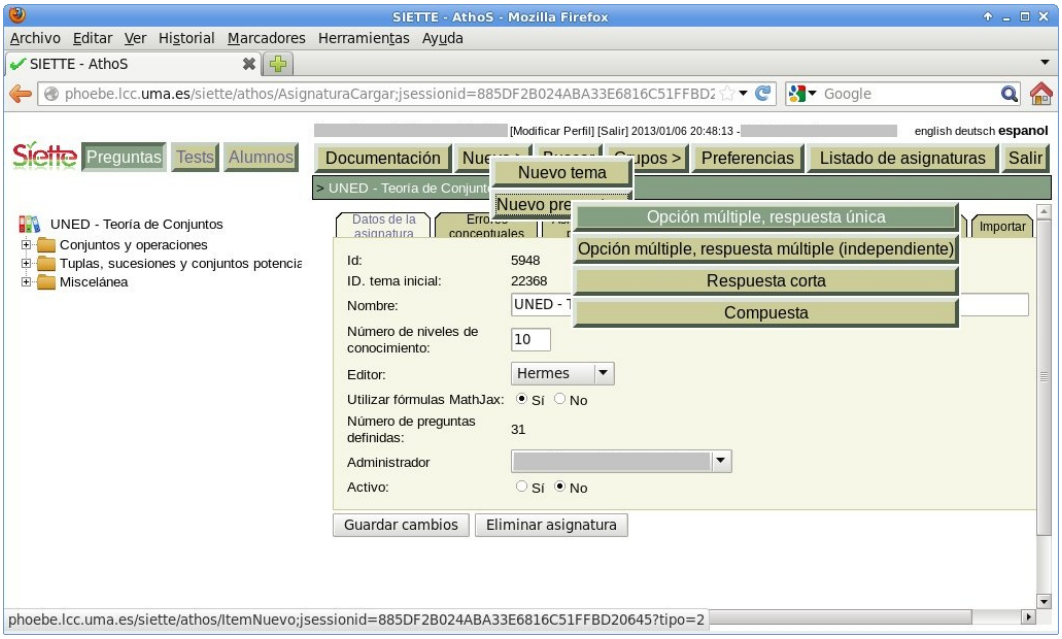


Figura B.4: Opciones de creación de Items.

primeras:

- Opción múltiple, respuesta única
- Opción múltiple, respuesta múltiple (independiente)

Los pasos explicados hasta este momento pueden ser revisados con más profundidad en la sección de documentación de la plataforma Siette:
`http://es.wiki.siette.org/index.php/Página_principal`

Una vez creado el item, es necesario realizar una serie de operaciones importantes para poder utilizar la librería. Para ello hay que dirigirse en primer lugar a la pestaña de gestión avanzada del item, tal como se muestra en la figura B.5, en esta página es necesario definir el *"Esquema de generación de pregunta"* al valor **JSP**, si adicionalmente se desea que este tipo de pregunta se repita en más de una ocasión en cada test, es necesario adicionalmente especificar el valor máximo de ocurrencias por test en *"Restringir núm de ocurrencias a"*.

Cuando ya se han llevado a cabo los pasos previos, se está en disposición de utilizar la librería para definir una plantilla de pregunta/item. Para realizar esta labor es necesario dirigirse a la pestaña de gestión de contenido que se muestra en la figura B.6.

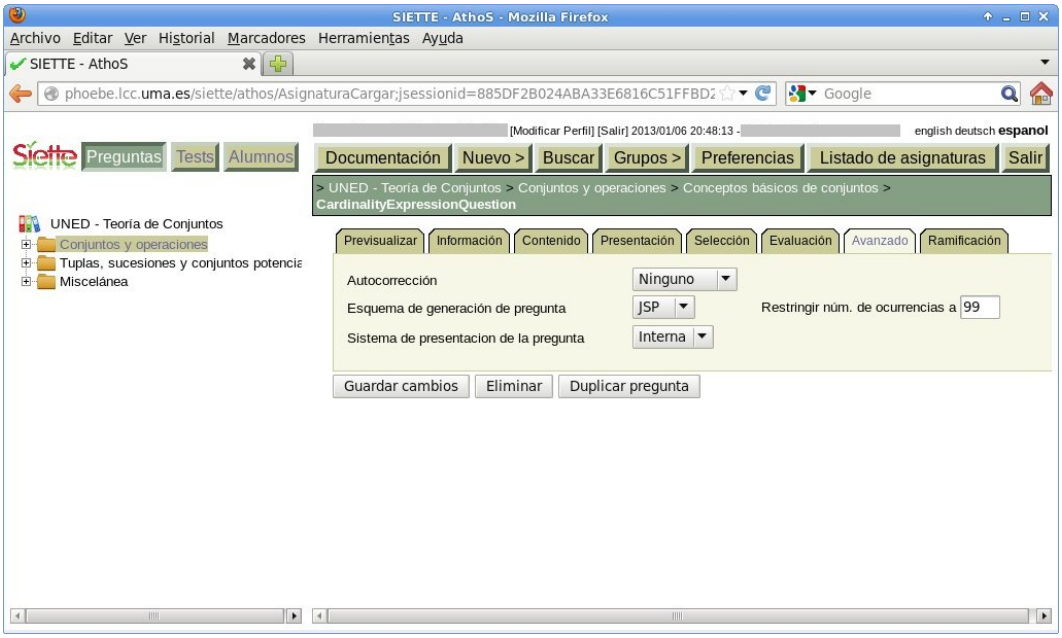


Figura B.5: Pestaña de gestión avanzada de Item.

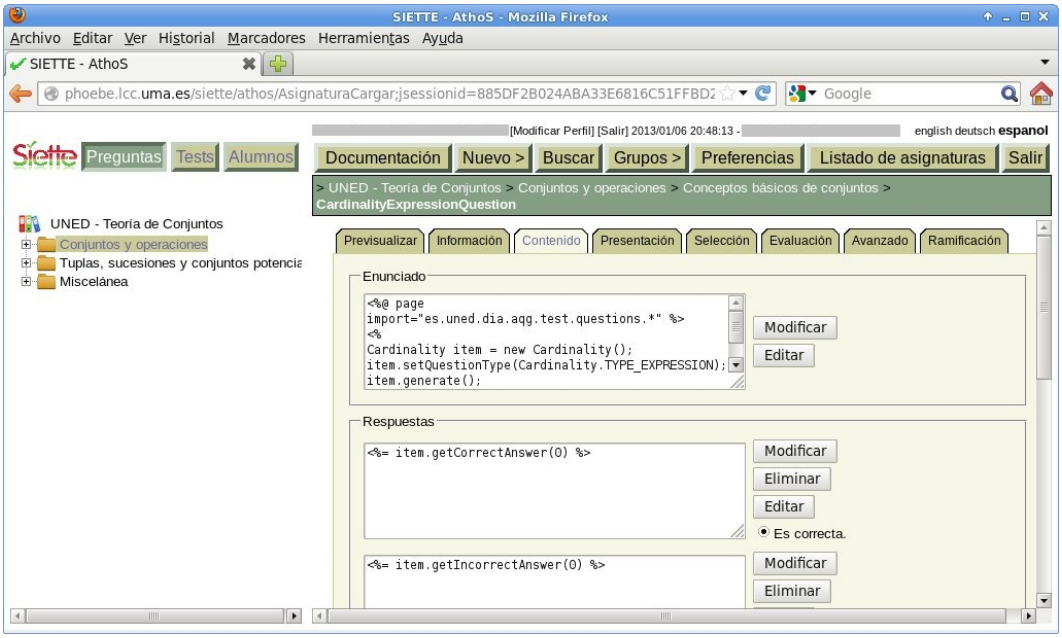


Figura B.6: Pestaña de gestión de contenido de Item.

En la página de gestión de contenido de Item se deben de definir al menos el enunciado y las posibles opciones de respuesta a la pregunta enunciada (Adicionalmente se pueden definir ayudas y refuerzos para apoyar la resolución de la pregunta).

La forma habitual de trabajar con la librería es:

1. Importar las clases de preguntas de la librería (*es.uned.dia.aqg.test.questions.**).
2. Instanciar alguno de los tipos de preguntas disponibles.
3. Configurar el tipo de pregunta si es necesario.
4. Generar la pregunta (invocando el método *generate()*)
5. Obtener los datos de la pregunta generada mediante los métodos habilitados para ello.

Listado B.1: Ejemplo de uso de la librería.

```
1  <%@ page import="es.uned.dia.pfc.jsanchez681.test.
       questions.*" %>
2  <% Cardinality item = new Cardinality();
3  item.setQuestionType(Cardinality.TYPE_SET);
4  item.generate(); %>
5  <%= item.getStatement() %>
```

Las clases de preguntas disponibles en la librería son: Cardinality, Disjoint, Membership, Partition, SubSuperSetRelations, SubSuperSetTerms, Complement, ComplexOperation, SimpleOperation, EquivalentExpressions, LawEnforcement, CartesianProduct, PowerSet, SpecialSets y Theory.

Los métodos que ofrece la librería para acceder a los distintos datos de una pregunta se exponen en la tabla B.1.

Método	Descripción
getStatement()	Proporciona un String con el contenido del enunciado de la pregunta.

Método	Descripción
getBacking()	Proporciona un String con el contenido del refuerzo (en Siette el refuerzo es un texto que se muestra si el alumno deja la pregunta sin contestar) de la pregunta.
getCorrectAnswer(int index)	Proporciona un String con el contenido de la respuesta correcta que se pase por parámetro. index es un entero mayor que 0 que indica el orden de la respuesta correcta dentro de la lista de respuestas.
getIncorrectAnswer(int index)	Proporciona un String con el contenido de la respuesta incorrecta que se pase por parámetro. index es un entero mayor que 0 que indica el orden de la respuesta incorrecta dentro de la lista de respuestas.
getIncorrectExplanation(int index)	Proporciona un String con el contenido de la explicación de la incorrección de la respuesta que se pase por parámetro. index es un entero mayor que 0 que indica el orden de la respuesta incorrecta dentro de la lista de respuestas.
getHelp(int index)	Proporciona un String con el contenido de la ayuda (en Siette una ayuda es un texto, que de estar habilitado, proporciona consejos para facilitar la resolución de la pregunta) que se pase por parámetro. index es un entero mayor que 0 que indica el orden de la ayuda dentro de la lista de ayudas disponibles.

Tabla B.1: Métodos de acceso a datos de preguntas.

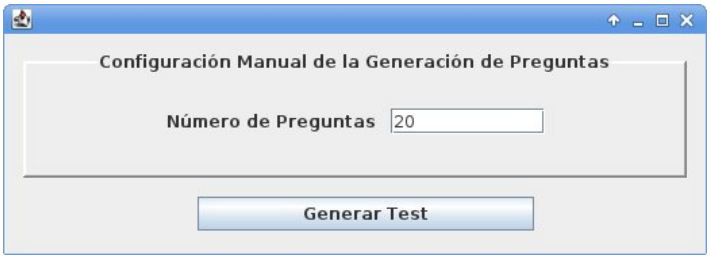


Figura B.7: Interfaz gráfica de usuario.

B.2. Uso independiente fuera de línea

La librería cuenta con dos pequeñas interfaces de usuario, una textual y otra gráfica. Dichas interfaces pueden ser invocadas de las maneras expuestas a continuación:

Listado B.2: Instrucciones de ejecución.

```
1  user@host:\~/bin\$ java -cp ./aqg.jar es.uned.dia.aqg.ui.
       Tui 20
2  user@host:\~/bin\$ java -cp ./aqg.jar es.uned.dia.aqg.ui.
       Gui
3  user@host:\~/bin\$ java -jar ./aqg.jar
```

1. La clase Tui se corresponde con la interfaz textual (TUI, Text-based User Interface) y se le puede pasar como parámetro un entero mayor que cero que se corresponderá con el número de preguntas que se desean generar. Si no se le pasa ningún parámetro se generarán por defecto 20 preguntas.

2. La clase Gui se corresponde con la interfaz gráfica (GUI, Graphical User Interface) y mostrará una pantalla como la de la figura B.7. En dicha pantalla se podrá especificar el número de preguntas que se desean generar, por defecto 20 preguntas.

3. La tercera forma de ejecución (llamada directa al JAR) invoca a la interfaz gráfica del punto anterior. En algunos sistemas podría ser suficiente con hacer doble clic sobre el icono del fichero JAR para realizar esta invocación.

Sea cual sea el método de invocación, el resultado es la generación en el directorio en el que se esté ubicado de dos ficheros PDF*:

test.tex Fichero que contiene únicamente las preguntas y sus opciones de respuesta.

testCorregido.tex Fichero que contiene además de las preguntas y las respuestas, indicaciones sobre la corrección de cada respuesta, así como las ayudas de las preguntas.

El método para trabajar con estos ficheros consiste en tratar de responder a las preguntas del test sin corregir y posteriormente cotejar los resultados con el test corregido.

*Se debe de contar con permisos de escritura en el directorio pertinente, así como encontrarse disponible en el sistema la utilidad *"pdflatex"*.

Apéndice C

Manual de instalación

En este apéndice se explica la forma en la que debe compilar y desplegar la librería desarrollada en este proyecto fin de carrera.

Para ello es necesario contar con el JDK de java (versión 6.0 o superior) adecuadamente instalado en el sistema.

Como ya se indicaba en 1.4, se puede encontrar material referente al presente capítulo en el espacio web habilitado para complementar la presente memoria:
`https://sites.google.com/site/aqgpfc/`

C.1. Compilación de la librería

Dentro de esta sección se explica tanto como compilar la librería de su código fuente, como la manera de comprimir la estructura de directorios de las clases en formato JAR (Java ARchive).

Para proceder a la compilación en primer lugar hay que situarse en el directorio *src*, que contiene los fuentes de la librería (en el CD-ROM sumunistrado se encuentra dentro del directorio principal *src*). Adicionalmente debería de estar correctamente configurado el CLASSPATH y el PATH para poder ejecutar el compilador de java: *javac*.

Una vez cumplidos los requisitos basta con ejecutar la siguiente secuencia de comandos*:

Listado C.1: Instrucciones de compilación.

```
1  user@host:\~/src\$ javac ./es/uned/dia/aqg/settheory/*.
       java
2  user@host:\~/src\$ javac ./es/uned/dia/aqg/test/*.java
3  user@host:\~/src\$ javac ./es/uned/dia/aqg/test/questions
       /*.java
4  user@host:\~/src\$ javac ./es/uned/dia/aqg/test/questions/
       options/*.java
5  user@host:\~/src\$ javac ./es/uned/dia/aqg/ui/*.java
6  user@host:\~/src\$ javac ./es/uned/dia/aqg/*.java
```

Los comandos previos generarán los ficheros binarios *.class* correspondientes a cada clase. Ahora se debe de proceder al empaquetado de todos los binarios generados en un fichero JAR. Las siguientes instrucciones suponen que se está ubicado en el directorio *src* indicado previamente y que en el nivel superior existe un directorio *bin*, que es donde se va a generar el fichero JAR.

Listado C.2: Instrucciones de empaquetado.

```
1  user@host:\~/src\$ jar cmvf ../MANIFEST.MF ../../bin/aqg.
       jar ./es/uned/dia/aqg/settheory/*.class
2  user@host:\~/src\$ jar uvf ../../bin/aqg.jar ./es/uned/dia
       /aqg/test/*.class
3  user@host:\~/src\$ jar uvf ../../bin/aqg.jar ./es/uned/dia
       /aqg/test/questions/*.class
4  user@host:\~/src\$ jar uvf ../../bin/aqg.jar ./es/uned/dia
       /aqg/test/questions/options/*.class
5  user@host:\~/src\$ jar uvf ../../bin/aqg.jar ./es/uned/dia
       /aqg/ui/*.class
6  user@host:\~/src\$ jar uvf ../../bin/aqg.jar ./es/uned/dia
       /aqg/*.class
```

*Todas las secuencias de comandos del presente proyecto se refieren a sistemas Linux, no obstante dada su sencillez no es difícil portarlas a otros sistemas.

Una vez seguidos los pasos expuestos (si no ocurre ningún error) la librería estará disponible en el directorio *bin*, con el nombre *aqg.jar*.

No es necesario pero podría ser conveniente eliminar los ficheros *.class* generados, para ello se pueden seguir las instrucciones que se indican a continuación.

Listado C.3: Instrucciones de eliminación de binarios.

```
1 user@host:\~/src\$ rm -f ./es/uned/dia/aqg/settheory/*.
    class
2 user@host:\~/src\$ rm -f ./es/uned/dia/aqg/test/*.class
3 user@host:\~/src\$ rm -f ./es/uned/dia/aqg/test/questions
    /*.class
4 user@host:\~/src\$ rm -f ./es/uned/dia/aqg/test/questions/
    options/*.class
5 user@host:\~/src\$ rm -f ./es/uned/dia/aqg/ui/*.class
6 user@host:\~/src\$ rm -f ./es/uned/dia/aqg/*.class
```

C.2. Despliegue de la librería en la plataforma Siette

En primer lugar es necesario contar con un usuario con permisos suficientes para realizar el despliegue (rol mínimo de profesor).

Desde la pantalla inicial de autenticación en Siette (figura C.1) se introducirán las credenciales necesarias para acceder al sistema.

Una vez dentro de la plataforma se seleccionará cualquier asignatura* (figura C.2).

Desde la sección de administración de la asignatura (figura C.3) se debe seleccionar la pestaña *Gestión de Archivos*.

En la pestaña de *Gestión de Archivos* (figura C.4), en primer lugar se debe especificar el directorio en el que desplegar la librería, normalmente este directorio tendrá una forma similar a "<id_asignatura>/WEB-INF/lib".

Posteriormente se ha de seleccionar el fichero en el sistema local de ficheros mediante el botón *Examinar...*. En este punto es importante resaltar que se recomienda comprimir

*No importa la asignatura seleccionada, ya que la ubicación de las librerías es común para todo el contexto de la plataforma Siette.

Figura C.1: Página de inicio de Siette.

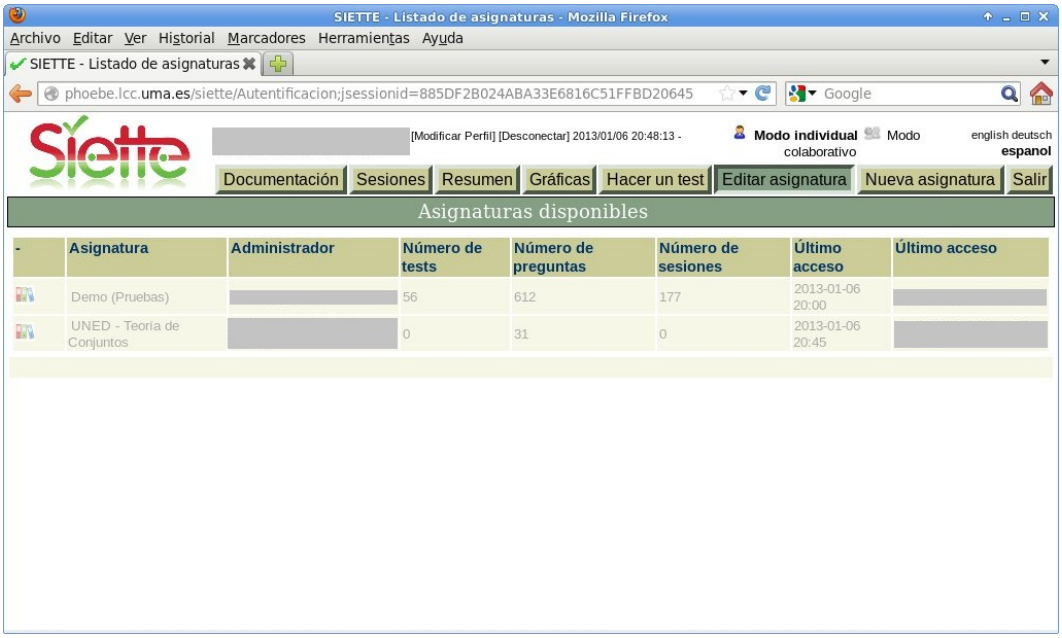


Figura C.2: Página principal de Siette.

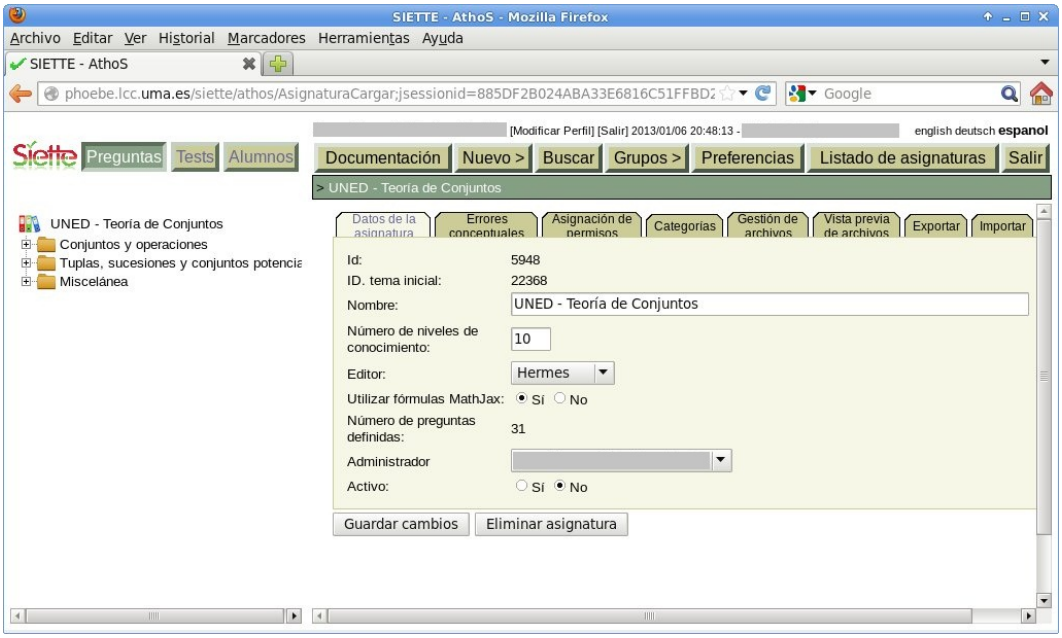


Figura C.3: Página de gestión de asignatura.

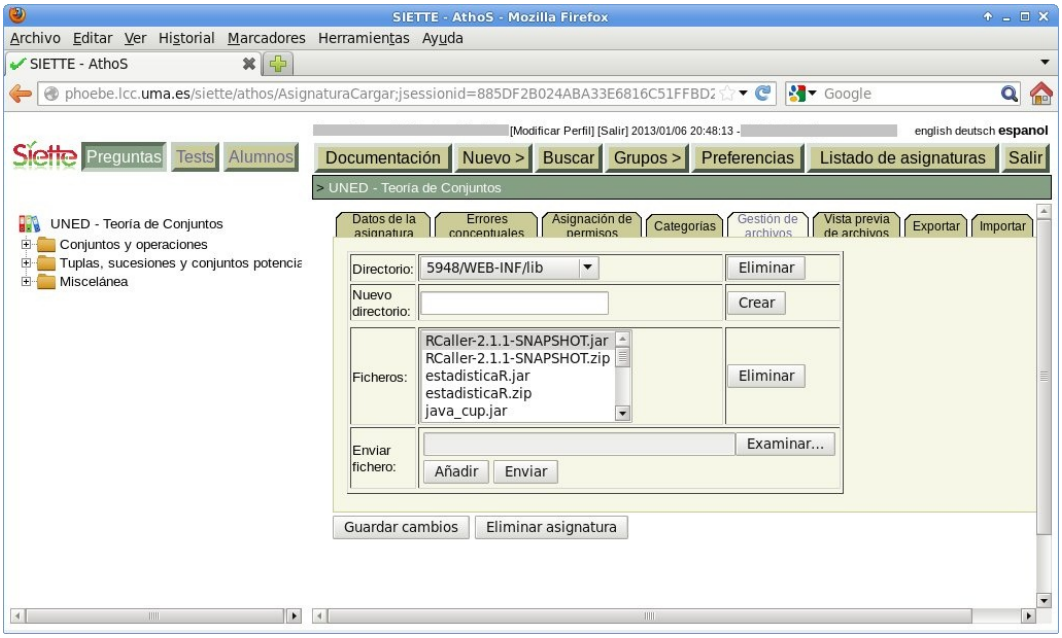


Figura C.4: Pestaña de gestión de archivos.

previamente la librería en formato ZIP*, esto es debido a que puede corromperse el fichero si se despliega en su formato nativo JAR.

A continuación se muestra la instrucción que permitiría realizar la citada compresión. Se supone que el usuario se encuentra en el directorio donde está ubicada la librería *aqg.jar*.

Listado C.4: Instrucciones de compresión de librería.

```
1  user@host:\~/bin\$ zip aqg.zip aqg.jar
```

Una vez realizados estos pasos ya es posible utilizar la librería desde cualquier item generativo JSP de la plataforma Siette.

*La plataforma Siette automáticamente descomprimirá el contenido del ZIP, extrayendo la librería dentro del sistema. Se recomienda tras el despliegue la eliminación del fichero ZIP intermedio para un uso eficiente del espacio de disco en Siette.

Apéndice D

Contenido del CD-ROM

Adjunto a esta memoria se entrega un CD-ROM con el contenido de la misma. Dicho CD-ROM está estructurado de la siguiente manera:

bin: En este directorio se aloja la librería lista para ser usada.

doc: Este directorio incluye la presente memoria en formato PDF.

src: Este directorio contiene los ficheros fuentes de la librería, de sus pruebas unitarias (junto con scripts ANT para su ejecución) y de las de rendimiento.

- **test:** Ficheros fuente de las pruebas unitarias, de rendimiento y de simulación.
  - **index.jsp:** Página JSP utilizada para la simulación funcional del sistema Siette.
- **src:** Ficheros fuente de la librería.
- **MANIFEST.MF:** Fichero de manifiesto para poder empaquetar adecuadamente la librería.
- **junitTest.xml:** Tarea ANT para facilitar la ejecución de las pruebas unitarias.
- **junitTestWithCompilation.xml:** Tarea ANT para facilitar la compilación de la librería y sus pruebas unitarias y la posterior ejecución de las pruebas unitarias.

test: Este directorio contiene material relacionado con las pruebas realizadas a la librería:

performance: Directorio que contiene los resultados de las pruebas de rendimiento del sistema en formato CSV*.

TEST-es.uned.dia.aqg.test.questions.AllTests.xml Fichero que contiene los resultados de la ejecución de las pruebas de JUnit en formato XML†.

leeme.txt: Este fichero contiene la misma información que este apéndice.

*CSV del inglés Comma-Separated Values.

†XML, del inglés eXtensible Markup Language.