UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Carrera de Ingeniero Informático

Generador automático de preguntas y respuestas de problemas de Combinatoria

José Antonio Sánchez Escudero

Dirigido por: Manuel Luque Gallego

Curso 2012/13 (convocatoria Diciembre-Enero)

Generador automático de preguntas y respuestas de problemas de Combinatoria

Proyecto Fin de Carrera de la modalidad general

Realizado por: José Antonio Sánchez Escudero

Dirigido por: Manuel Luque Gallego

Tribunal Calificador:

Presidente: D./Dª. .....

(firma)

Secretario: D./Dª. .....

(firma)

Vocal: D./Dª. .....

(firma)

Fecha de Lectura y Defensa.....

Calificación.....

Resumen

Es innegable el continuo desarrollo evolutivo en la enseñanza, desde los tiempos del trinomio profesor-alumnado y pizarra a la actualidad, donde la educación está fuertemente ligada al uso de los medios digitales.

En este arduo camino, profesionales como los ingenieros informáticos han allanado el recorrido desarrollando herramientas que faciliten el proceso de enseñanza-aprendizaje del alumnado, así como ayudando en la transmisión del conocimiento entre el docente-discente.

Actualmente, estamos asistiendo a una continúa revolución tecnológica donde cada día surgen nuevos recursos. Los profesionales en el campo educativo, deben adaptarse continuamente al uso y manejo de nuevos medios digitales.

La Universidad UNED posee gran experiencia en el campo de la educación telemática y en el uso de medios tecnológicos en la enseñanza, debido a las necesidades especiales surgidas en la educación del alumno no presencial.

Este proyecto pretende ser un acercamiento en el uso de las nuevas tecnologías dentro del ámbito educativo; una nueva herramienta para el aprendizaje del alumnado, en concreto en la asignatura de matemáticas y en especial, en la parte de combinatoria.

El sistema “siette” abre un nuevo canal de aprendizaje para el alumnado, generando de manera interactiva test de preguntas y respuestas que faciliten el asentamiento de los conocimientos ya adquiridos. El presente proyecto añade nuevas ramas de conocimiento en dicho sistema.

Por último y no menos importante, no debemos olvidar la interacción con el alumnado. Son numerosos los estudios que explican que el uso de medios interactivos facilitan el aprendizaje, que podemos simplificar en una frase: “una imagen vale más que mil palabras”. Este proyecto

utiliza el recurso de la imagen al incluir dibujos realizados en SVG en cada uno de los ejercicios, de forma que el alumnado pueda comprender mejor lo que se le pregunta y le permita asimilar más fácilmente los conocimientos adquiridos mediante un ejemplo práctico.

En este sentido, este proyecto supone una herramienta gráfica novedosa para favorecer y asentar el aprendizaje del alumno de forma creativa e interactiva.

Palabras clave

HTML5

SVG

SIETTE

Combinatoria

Java

Generador automático de preguntas respuestas

Matemática Discreta

Abstract

It is undeniable the continuous evolutionary development in education, from the past when the trinity “teacher- students-blackboard” prevailed, until now, where education is strongly linked to the use of digital media.

In this difficult path, professionals like the computer engineers, have smoothed the way, by means of the developing of tools to facilitate the teaching-learning process of the students, assisting by this way in the transfer of knowledge between teacher and pupil.

Currently, we are witnessing a continuous technological revolution, where new resources appear each day. Professionals in the field of education, must continually adapt themselves to the use and management of new digital media.

The UNED University has extensive experience in the field of telematics education and the use of technological means in teaching, due to the special needs arising in the education of the non-attendance student.

This project aims to be an approach to the use of new technologies in the field of education, a new tool for the pupils learning, particularly in the mathematics subjects and especially in the part of combinatorics.

The "SIETTE" system opens a new channel of learning for students, by generating, in a interactively way, of question tests and their answers to facilitate the stablishment of the knowledge acquired. This project adds new branches of knowledge in that system.

Last but not least, we must not forget the interaction with students. They are many studies which explain that the use of interactive media facilitates the learning. As everybody knows it is said that "an image is worth more than a thousand words" so this project makes use of this

resource by including SVG drawings in every exercises, so that students can better understand what is asked and easily assimilate the knowledge gained through a practical example.

In this sense, this project represents a new graphical tool to assist and consolidate the student learning in a creative and interactive way.

Keywords

HTML5

SVG

SIETTE

Combinatorial numbers

Java

Automatic system generator of answer questions

Discrete mathematics

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Contexto del PFC

1.1. Marco Actual

El avance en las tecnologías de la información ha permitido y posibilita nuevas formas de aprendizaje para el alumnado. Hoy en día, disponemos de nuevas herramientas que facilitan la adquisición de conocimientos. El auge sufrido desde la revolución de la era Internet, es comparable al que supuso la televisión en la vida de nuestros padres.

Herramientas como el sistema SIETTE, permiten la generación de baterías de preguntas de test que ayudan en la asimilación de nuevos conocimientos del alumnado.

La interactividad en la que está focalizado SIETTE permite que el alumno avance en su aprendizaje a medida que va resolviendo ejercicios. No podemos resumir la función de SIETTE en un mero libro de ejercicios resueltos, es más que todo eso. Haciendo uso de otra de sus características, vemos que facilita el aprendizaje colaborativo entre compañeros, el intercambio de conocimiento entre los mismos,... etc.

Dentro del sistema SIETTE podemos encontrar test sobre diferentes temas o áreas de conocimiento, como por ejemplo, arte y literatura, botánica, etc. Este proyecto añade una nueva área de conocimiento a SIETTE, desarrollando la parte de combinatoria.

SIETTE admite diferentes formatos de entrada de sus ejercicios, tales como ejercicios desarrollados en txt, páginas jsp e incluso se pueden incluir applets y librerías jar que faciliten o

mejoren la visibilidad de los ejercicios. Podemos considerar a SIETTE como una herramienta adaptable al tiempo actual y pluridisciplinar, al no estar limitado su uso a una única área de conocimiento.

Actualmente, el alumnado maneja nuevas herramientas de trabajo que han sufrido un gran auge en los últimos años, como los smartphones y las tabletas.

El auge de los smartphones sucedido en nuestro país ha superado enormemente las expectativas de penetración, siendo líderes en el mercado europeo (2011) con un índice del 33 % de introducción (según el estudio publicado en "http://www.thinkwithgoogle.com/mobileplanet/es/"). Sin embargo aún siendo lideres, otro estudio realizado por Smartcom revela otro dato muy interesante: la población de Reino Unido es la que más saca partido al uso del smartphone, utilizando más aplicaciones en los terminales; así mismo, también lidera el consumo de navegación a través del mismo.

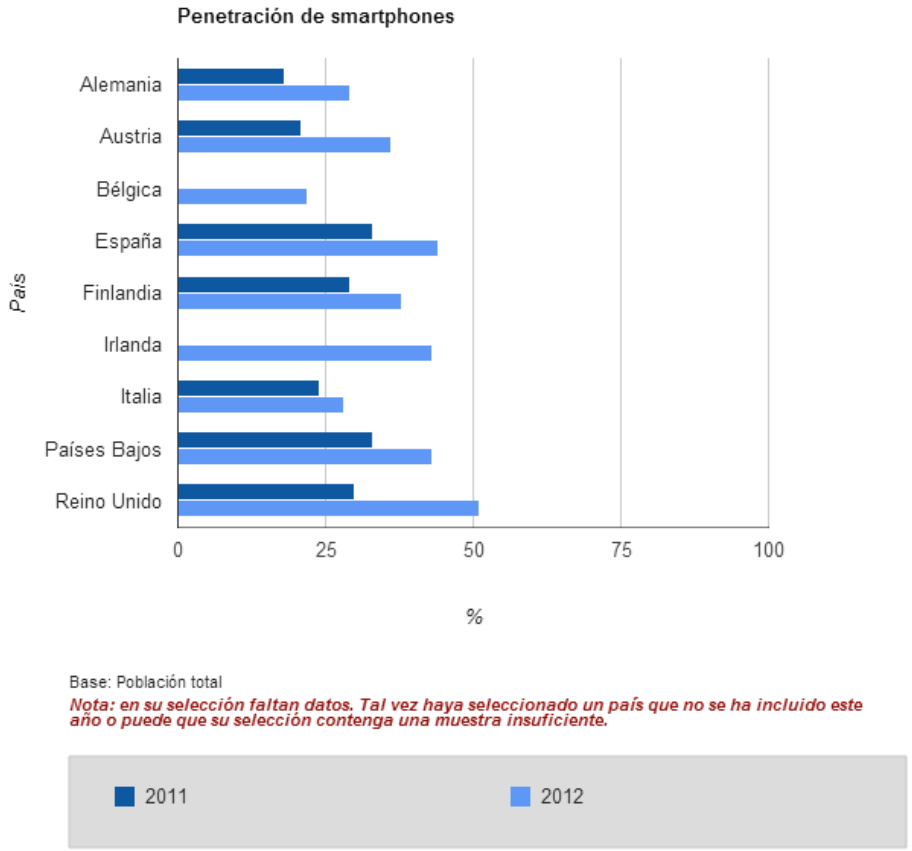

Figura 1.1.1: *Estadística de introducción de smartphone 2011-2012*

Al igual que en los smartphone, el uso de las tabletas ha supuesto una verdadera revolución en los últimos años. Se estima que el año 2012 cierre con un número de 1.6 millones de tabletas vendidas. Por otra parte, existen previsiones de que el mercado de tabletas supere al mercado de pc de sobremesa en el año 2012.

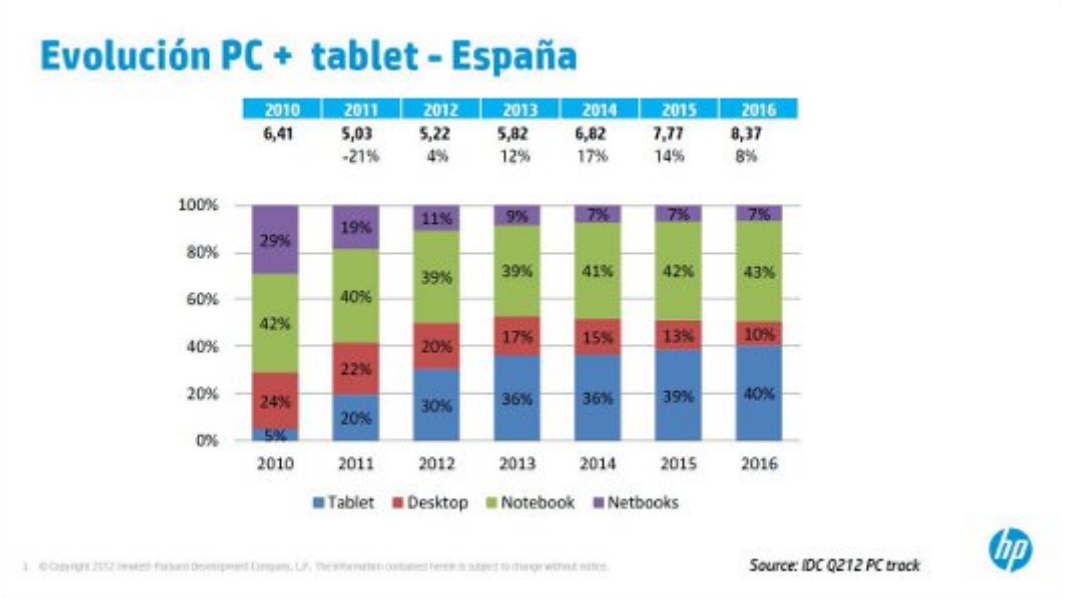

Figura 1.1.2: *Previsiones HP sobre la evolución de la ventas de informática*

Observando la evolución del mercado de smartphone y tabletas, vemos que estos dispositivos han llegado para quedarse y por tanto, debemos adaptar nuestras aplicaciones para que puedan ser usadas en los mismos. Estas adaptaciones conllevarán problemas que debemos analizar y subsanar para conseguir nuestro objetivo.

La gran variedad de fabricantes de dispositivos en el mercado, ha facilitado el auge de los mismos y una estrecha competencia entre las características y especificaciones de cada dispositivo. La competencia no solo existe a nivel del hardware del dispositivo, sino también en los múltiples sistemas operativos implementados.

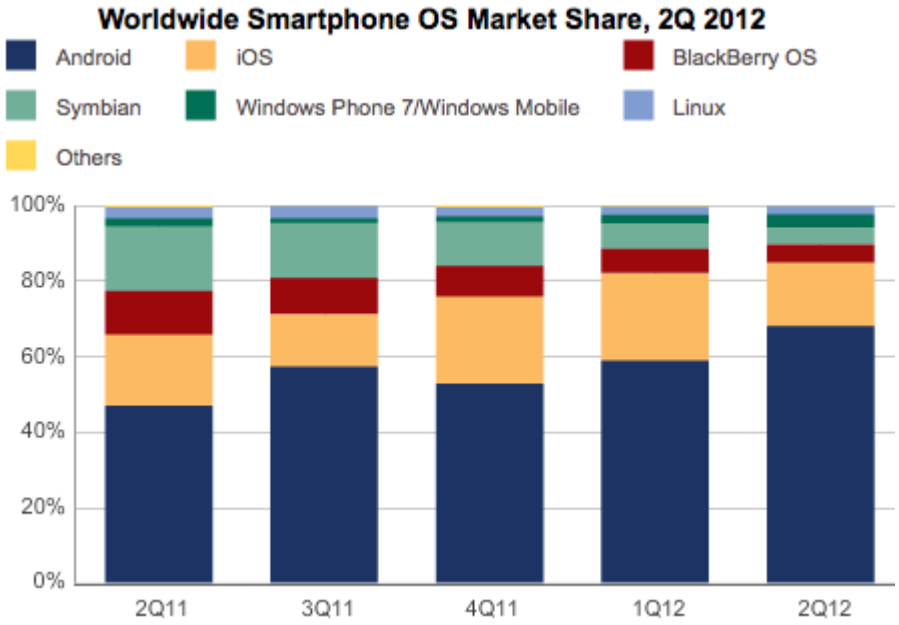

Figura 1.1.3: *Distribución por sistema operativo smartphone*

Tanto en los smartphone como en las tabletas, los principales sistemas operativos utilizados son android e ios pero no soportan java en la actualidad, aunque existen ciertas aplicaciones que permiten correr adaptaciones de java dentro de dichos dispositivos. Este inconveniente impone la utilización de nuevos lenguajes para partes del ejercicio desarrollado en SIETTE, que sean compatibles con los sistemas operativos actuales.

En este proyecto vamos a trabajar con HTML5 y en concreto con SVG para incluir figuras en los ejercicios.

SVG es un sistema de gráficos vectoriales compatible en la actualidad en los navegadores firefox y chrome, que se incluyen en los anteriores dispositivos; también es compatible con las nuevas versiones de ios, y por tanto, de su navegador leopard.

Por otra parte, HTML5 es la quinta revisión importante del lenguaje básico de la World Wide Web y aporta diferentes ventajas como la adicción de nuevas etiquetas que facilitan la web semántica, la integración de dibujos y animaciones de manera nativa mediante SVG, geolocalización, etc.

1.2. Objetivos

Como se ha indicado anteriormente, el uso de nuevos dispositivos y la necesidad de nuevas aplicaciones para los mismos que faciliten la labor docente al profesorado, conforman conjuntamente la motivación de este proyecto.

Este proyecto tendrá como objetivo principal la creación de una librería desarrollada en java para el entorno SIETTE, la cual versará sobre problemas de combinatoria.

Una vez establecido el objetivo principal de este proyecto, se extraen los siguientes objetivos secundarios que serán necesarios para la consecución del objetivo principal o adicionales al mismo:

- Cada ejercicio ofrecerá una respuesta correcta y dos incorrectas.
- Las respuestas incorrectas serán parecidas a las respuestas correctas, favoreciendo el desarrollo del problema por parte del alumnado y evitando el azar en la resolución del mismo.
- Las respuestas pueden ser mostradas en diversos formatos: numérico, fórmula o mediante expresión de tiempo.
- Cada problema vendrá acompañado de un dibujo que será parte fundamental del problema, siendo necesario el visionado del mismo para llegar a la correcta resolución del ejercicio.
- El tiempo de carga de cada ejercicio no superará la décima de segundo.
- La librería completa debe ser probada antes de la fase de explotación, verificando que es estable bajo las situaciones descritas en cada uno de los ejercicios que componen esta librería.

1.3. Trabajos anteriores

Son innumerables los diversos acercamientos realizados en el mundo de la informática que facilitan el aprendizaje al alumnado. La mayoría de éstos se basan en un entorno estático, que

consiste en la repetición de los mismos ejercicios de forma reiterativa. SIETTE nos permite a los desarrolladores realizar ejercicios dinámicos, de tal forma que los alumnos siempre se podrán enfrentar a variantes de un mismo problema. Además, añade un sistema correctivo que ayuda al alumno a entender el error y subsanarlo.

En Internet podemos encontrar numerosos ejemplos de software que implementan una batería de ejercicios de test de manera estática. Podemos resumirlos en dos tipologías: La primera, software local, se instala directamente en el equipo del usuario. Puede actualizarse en la web del autor, pero no es necesario para su uso la conexión a Internet. El segundo, una página web, implementa los ejercicios haciendo uso mayoritariamente dos tipos de tecnología: Java, mediante un applet, o Javascript. Este tipo de desarrollo facilita a programador y alumno la modificación de los ejercicios en tiempo real. Estas dos metodologías de desarrollo no permiten la interacción del alumnado y, por consiguiente, no suponen una guía en su aprendizaje.

Actualmente, existe una tendencia a la creación de blogs por parte del profesorado, como forma de comunicación con el alumnado y seguimiento de su aprendizaje. En ellos, el profesor suministra desde contenido adicional a lo trabajado en clase, pudiendo dirigir a los alumnos hacia sitios web donde encontrar materiales que permitan ampliar dichos contenidos, hasta la corrección de los ejercicios enviados.

Dentro de los recursos dirigidos al profesorado que podemos encontrar en la web, destacaremos la proporcionada por el MECD “Descartes”. Este recurso supone una herramienta al profesorado y alumnado que ofrece ejercicios implementados mediante applets. Además, incluyen un enunciado teórico previo a los ejercicios para facilitar su comprensión. Sin embargo, carecen de interacción con el alumnado y no posibilitan un aprendizaje constructivo al no disponer de una explicación del error. En conclusión, las herramientas anteriores suponen un avance en la metodología, ya que ofrece una línea de trabajo más interactiva y acorde a los nuevos tiempos.

1.4. Aportaciones

En el apartado anterior, hemos analizado diversas herramientas que permiten una generación automática de ejercicios, pero que carecen de la interacción ofrecida por el sistema SIETTE. En nuestro proyecto, también proponemos una batería de ejercicios, sin embargo, gracias a las funcionalidades que posee SIETTE, vamos a poder interaccionar con el alumno. A continuación, expondremos dichas aportaciones.

- La resolución de los ejercicios se realiza en función de la respuesta proporcionada por el alumno. En este sentido, no es importante el echo de que el alumno haya sabido responder correctamente al ejercicio, sino que el sistema le ofrece una explicación del porqué de su respuesta errónea.

- Nuestra herramienta permite que los valores mostrados en los enunciados sean diferentes en cada generación, siempre dentro de un rango de parámetros.

- El profesor, puede aumentar o disminuir el grado de dificultad de los ejercicios, para dar respuesta a las diferentes necesidades del alumnado mediante el uso de parámetros de entrada en cada ejercicio.

- La mayor aportación de nuestro sistema supone la inclusión de una figura descriptiva en cada ejercicio, lo cual ofrece una vía más atractiva y motivante para el alumnado.

- SVG y HTML5 proporcionan a nuestro proyecto una compatibilidad con los nuevos dispositivos, permitiendo la realización de los ejercicios mediante el uso de un smartphone o tableta.

Todas estas aportaciones posibilitan un aumento en la motivación del alumno, al ofrecer aplicaciones prácticas de lo aprendido en un ambiente más interactivo.

1.5. Método de trabajo

Una vez establecido el marco actual en el cuál se desarrolla este proyecto y los objetivos del mismo, podemos abordar el desarrollo del trabajo subdividiendo la consecución del mismo en la siguientes subtareas:

- Toma de requisitos: a partir de los objetivos del proyecto y de las sucesivas entrevistas con el cliente, debemos extraer un documento que contendrá cada uno de los requisitos que debe contener el sistema final para que sea considerado válido. El documento de requisitos no sólo deberá contener las especificaciones del sistema a desarrollar, sino datos sobre la plataforma donde será explotado así como otros requisitos que afecten al desarrollo del mismo, tales como las herramientas de trabajo, etc.

- Cálculos de costes: a través del punto anterior, ya conocemos que debe hacer nuestro sistema y donde será desplegado. Es hora de determinar los costes de desarrollo y para ello, debemos subdividir el cálculo en el número de personas necesarias para el desarrollo del mismo y el cálculo del computo del tiempo en la elaboración del producto. Así mismo, no debemos olvidar el cálculo de costes de los sistemas y herramientas utilizadas.

- Análisis y diseño del sistema: es hora de extraer de los requisitos las funcionalidades que deberá tener nuestro sistema al final del proyecto, a través de ellas y mediante el uso de herramientas como diagramas de clases, casos de uso, étc, generaremos los componentes necesarios y desarrollaremos el código del esqueleto de nuestro sistema.

- Desarrollo de los ejercicios de test que componen el proyecto, en la fase anterior se han desarrollado los paquetes base que componen el proyecto, a partir de ellos en esta fase y haciendo uso de la herencia, generaremos cada uno de los ejercicios que formarán parte del sistema final.

- Despliegue y configuración, el objetivo de este proyecto es generar un sistema que pueda ser utilizado en el mundo real, el despliegue del mismo y su configuración en un servidor de producción es el paso final del proyecto.

- Conclusiones y lineas futuras, en este apartado analizaremos los conocimientos aprendidos, la evolución del proyecto y estudiaremos posibles vías de ampliación que permitan mejorar el presente proyecto.

Capítulo 2

Análisis económico y estudio de viabilidad

En el capítulo 1 se han establecido los objetivos y motivaciones de este proyecto y posteriormente se ha analizado la situación actual o punto de partida, dónde se establece qué aplicaciones o modelos se están utilizando y dónde deseamos llegar mediante los objetivos marcados.

Establecido el marco actual (grano grueso), es el momento de ir desglosándolo en subtareas. Partíamos de los objetivos del proyecto y en este punto, llegamos a la toma de requisitos (el grano fino).

2.1. Requisitos

La toma de requisitos es una de las principales fases a la hora de desarrollar un proyecto informático. Se podría afirmar que los principales problemas que ocurren en el desarrollo de un proyecto informático, son la falta de concreción en la toma de requisitos, la ambigüedad de los mismos,... etc. Este tipo de errores genera falsas expectativas en el cliente o producen productos que no se ajustan a las necesidades del cliente.

Existen dos subtipos de requisitos en función de las necesidades que engloban:

- Requisitos funcionales: son aquellos que se ocupan de las características del aplicativo, las tareas que desea el cliente que realice el aplicativo, las funcionalidades que tendrá el mismo,... etc.

- Requisitos no funcionales: en este grupo se encuentran aquellos requerimientos que son necesarios para el correcto funcionamiento del aplicativo o bien, aquellos que afectan al proceso de desarrollo del mismo; véase como ejemplo sus capacidades, etc.

2.2. Requisitos Funcionales

Tras una valoración inicial del proyecto a desarrollar, establecemos que dentro de los requisitos funcionales debemos incluir dos tipologías en función de los actores que interaccionan con el aplicativo.

- Requisitos del alumno: son aquellos que se desarrollan en el ámbito del alumnado, tales como la funcionalidad de la aplicación o la facilidad e interacción de la misma.

- Requisitos del profesor: establecen y permiten la definición de parámetros de trabajo del aplicativo, regulando factores como la dificultad del mismo.

Es importante reseñar que en la fase de especificación de requisitos suele ocurrir que algunos de ellos, o bien pertenecen a dos actores, o la barrera entre uno y otro actor puede ser bastante difusa; aún así, no tiene por qué ser un motivo que dificulte el resto del proyecto.

2.2.1. Requisitos funcionales del alumno

Requisito	Descripción
RA01	El sistema generado, será un sistema de preguntas tipo test, que permita a los alumnos la realización de ejercicios de la materia de combinatoria
RA02	Cada ejercicio posee una ayuda que facilite la resolución del mismo, en caso de desconocer la respuesta.

Cuadro 2.1: *Requisitos Funcionales del alumno*

2.2.2. Requisitos funcionales del profesor

Requisito	Descripción
RP01	Cada ejercicio implementado debe ser ajustable por el docente, en su nivel de dificultad.
RP02	Cada ejercicio poseerá un dibujo explicativo que versará sobre el enunciado del problema; así mismo, para la correcta resolución del mismo será necesario el uso del la figura que acompaña al ejercicio.
RP03	Cada ejercicio tendrá un respuesta correcta y 2 incorrectas generadas aleatoriamente, que han de ser muy parecidas a la solución del ejercicio.
RP04	La respuesta a los ejercicios serán enunciadas de tres maneras diferentes en función de la dificultad del ejercicio y de un valor aleatorio, que determine en tiempo de ejecución la modalidad mostrada. Las opciones posibles serán: ▪ Formato numérico, la solución al problema es una cantidad que representa la solución del ejercicio. ▪ Formato fórmula, en este modo, la solución es presentada mediante una ecuación matemática. ▪ Formato tiempo, en ciertos ejercicios, para elevar la dificultad del mismo, la solución será mostrada como un número de días, horas, minutos y segundos.
RP05	Cada ejercicio, mostrará una solución al finalizar el mismo (sí es activado por el profesor) explicando el planteamiento para llegar a la solución.
RP06	Cada ejercicio, poseerá una explicación en caso de seleccionar una respuesta incorrecta, donde se expondrá porque esa respuesta no puede ser la correcta.

Cuadro 2.2: *Requisitos funcionales del profesor*

2.3. Requisitos no funcionales

2.3.1. Ámbito de trabajo

Requisito	Descripción
RT01	La librería de preguntas estará implementada íntegramente en Java.
RT02	La librería será desplegada en el sistema SIETTE, por tanto debe ser compatible con el mismo y aprovechar sus funcionalidades.
RT03	La librería se comunicará con SIETTE mediante páginas web desarrolladas en jsp.
RT05	Para el uso de fórmulas matemáticas, se utilizará la librería Mathjax.
RT06	El desarrollo de toda la documentación del presente pfc, será escrita mediante el editor Lyx que usa el lenguaje LaTeXcomo base de la escritura de documentos.
RT07	Como sistema de almacenamiento y repositorio común de código fuente almacenado en la nube, se usará Bitbucket en conjunción con TortoiseHG.

Cuadro 2.3: *Requisitos no funcionales. Ámbito de trabajo*

2.4. Estudio de alternativas a la solución

Son pocas las opciones abiertas a la hora de establecer la solución, en el caso del sistema que almacenará nuestro proyecto. Éste está previamente establecido como requisito no funcional. El lenguaje de programación también se encuentra preestablecido, así como el programa usado para realizar la documentación, etc.

Uno de los requisitos que debe ser evaluado previamente al desarrollo de este proyecto, es la carga gráfica que posee. Se estudian 3 posibles soluciones:

- Archivos “jpg”: Es un formato de almacenamiento de imagen dentro de un archivo, siendo el más común de los formatos utilizados en fotografía y en webs; así mismo, los archivos “jpg” poseen la cualidad de almacenar su contenido de forma comprimida. Su principal inconveniente en la utilización dentro de este proyecto, es que los archivos son estáticos; dicho de otra forma, las imágenes son almacenadas y mostradas en tiempo de ejecución pero siempre serían las mismas.

- Applets: un applet es un componente de una aplicación que se ejecuta en el contexto de

otro programa. En la mayoría de las ocasiones, este programa suele ser una web. Está desarrollado íntegramente en java y posee cualidades gráficas que facilitarían su uso dentro de este proyecto. Sin embargo, presenta una pequeña desventaja cuando realizamos los test en SIETTE en dispositivos smartphones o tabletas, ya que a día de hoy java no es compatible en dichos dispositivos.

- Dibujos vectoriales SVG. Es un lenguaje abierto que permite crear gráficos vectoriales 2D basado en , tanto estáticos y como animados. La mayoría de los navegadores soportan gráficos SVG, pues el formato ha sido recomendado por la en septiembre de 2001. En el ámbito de este proyecto SVG nos permite la creación de imágenes que pueden ser estáticas o dinámicas en tiempo real. Así mismo, otra cualidad que posee SVG es su compatibilidad con un gran número de dispositivos, ya que como comentábamos en el punto anterior es un estándar recomendado e implementado en la mayoría de los navegadores actuales.

2.5. Selección de la solución

La compatibilidad del aplicativo entre los diferentes dispositivos no es un “requisito no funcional”, aunque puede ser un valor añadido para el uso de este aplicativo.

Otro aspecto a valorar es el ancho de banda utilizado, así como el tiempo de carga. En el caso de los archivos jpg, aunque el tamaño de los mismos sea pequeño, es necesario una carga así como un almacenaje en el servidor SIETTE.

Respecto a la resolución de la imagen y su calidad, los dibujos vectoriales poseen una mayor calidad que las imágenes almacenadas en un mapa de bits. Otro aspecto importante, aunque no afecte a la calidad de este proyecto, es que las imágenes vectoriales son compactas y ocupan menos tamaño que las anteriores ya que almacenan su información mediante fórmulas matemáticas.

Todos estos factores nos llevan a tomar la elección de utilizar gráficos vectoriales “SVG” en este proyecto, tanto por su versatilidad como por su tamaño.

2.6. Cálculos de tiempo

Para la realización de este proyecto se estima que es necesario un equipo formado por dos personas, un analista y un programador (Ver figura inferior analista de color azul y programador de color verde). Inicialmente es el analista el encargado de recoger los requisitos y dar forma al proyecto, por tanto se estima que el programador pueda empezar su trabajo a partir de mayo.

El analista realizará diversas tareas y también llevará a cabo la realización de la de documentación, la cual se subdividirá en dos tareas distintas, documentación pre-desarrollo y documentación pos-desarrollo.

En total se estima que el programador utilice 172 días naturales y el analista realice su trabajo en 68 días.

Actividad	Fecha de Inicio	Duración (días)	Fecha de Fin
Recogida de requisitos	01/03/2012	5	06/03/2012
Marco actual	06/03/2012	2	08/03/2012
Investigación de trabajos Anteriores	08/03/2012	6	14/03/2012
Calculo de alternativas	14/03/2012	8	22/03/2012
Ampliación de requisitos	22/04/2012	5	27/04/2012
Análisis paquete ejercicio	27/04/2012	5	02/05/2012
Análisis paquete Dibujo	02/05/2012	4	06/05/2012
Análisis paquete Operación	06/05/2012	3	09/05/2012
Implementación paquete ejercicio	02/05/2012	21	23/05/2012
Implementación paquete Dibujo	23/05/2012	14	06/06/2012
Implementación paquete Operación	06/06/2012	7	13/06/2012
Análisis ejercicios con urnas	09/05/2012	4	13/05/2012
Implementación ejercicios con urnas	13/06/2012	30	13/07/2012
Análisis resto de ejercicios	13/05/2012	5	18/05/2012
Implementación resto de ejercicios	13/07/2012	90	11/10/2012
Documentación hasta análisis	18/05/2012	10	28/05/2012
Pruebas	11/10/2012	10	21/10/2012
Documentación pos desarrollo	22/10/2012	5	27/10/2012
Conclusiones	27/12/2012	6	02/01/2013

Figura 2.6.1: *Diagrama de actividades*

2.7. Diagrama de Gantt

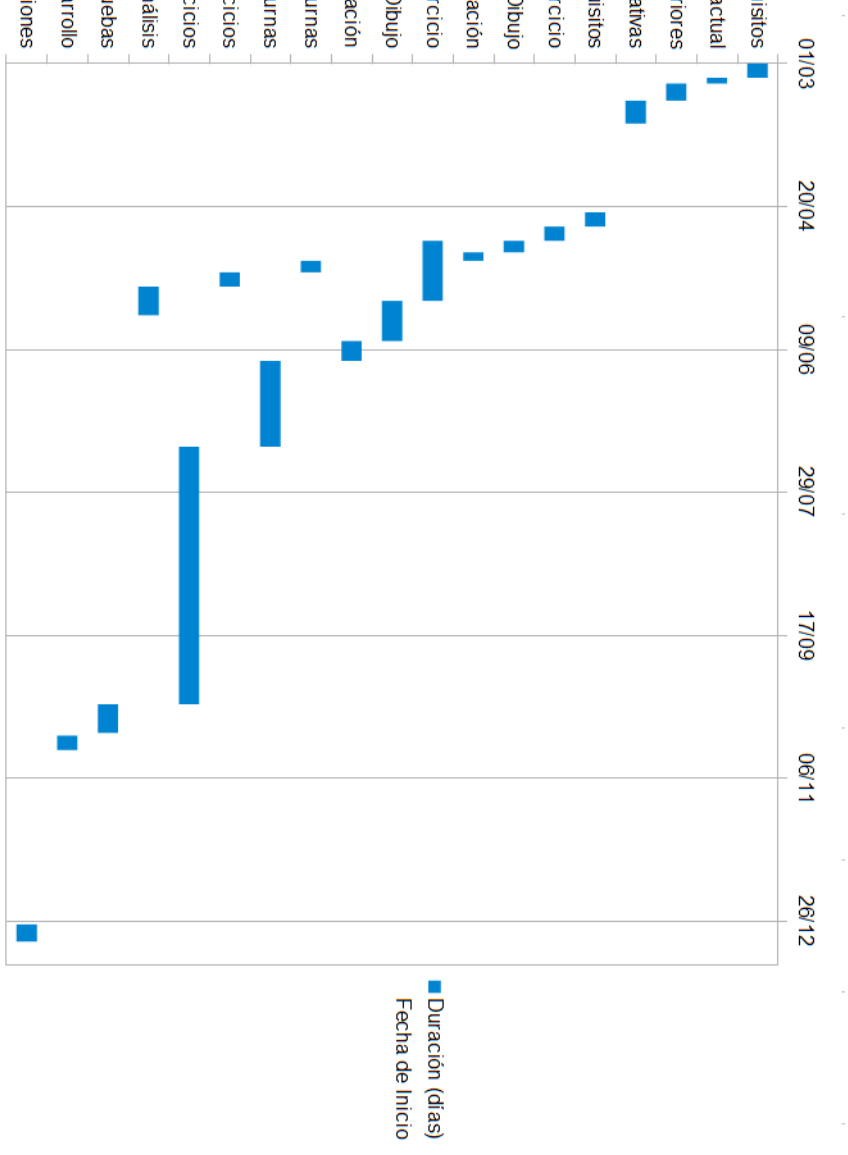

Figura 2.7.1: *Diagrama de actividades*

2.8. Análisis económico inicial

Como fase inicial a cada proyecto y una vez tomados los requisitos del mismo. Debemos proceder a realizar un cálculo inicial del coste del producto a desarrollar.

Son muchos los factores que van a influir en el coste del desarrollo del producto y es por ello que se establece necesario como paso previo a la generación del producto, el cálculo de cada uno de los elementos que intervienen el proceso de fabricación.

Dentro de los gastos generados, podemos diferenciar entre aquellos debidos al coste del personal, al coste de los sistemas informáticos y su mantenimiento, así como aquellos relativos a las herramientas usadas en el desarrollo del producto.

2.8.1. Gastos de personal

Mediante las medidas de tiempo obtenidas en los cálculos de tiempo realizados anteriormente, podemos extraer el coste del personal en función de los días contratados. En la actualidad el salario de un Analista rondaría los 3.000 euros brutos y el de un desarrollador estaría cercano a los 1.800 euros brutos, con lo que el gasto total asciende a lo siguiente:

$DíasProgramador * Salario/30 \Rightarrow 172 * 1800/30 = 10{,}320€$

$DíasAnalista * Salario/30 \Rightarrow 68 * 3000/30 = 6{,}800€$

2.8.2. Gastos del sistema informático y mantenimiento

Durante de la elaboración de un proyecto, hay una serie de gastos nada desdeñables, que en ciertas ocasiones son ignorados pero que suponen un desembolso inicial o bien mensual que afectan al coste final del producto.

En el proyecto que nos ocupa nos enfrentaremos a los siguientes gastos:

- Coste ordenador personal. $OrdenadorGamaMedia * 2 * CosteUnitario \Rightarrow 2 * 700 = 1{,}400€$
- Coste de la energía del mismo. $ConsumoOrdenador * CosteKw * Iva * DiasNaturales * horasDia * 5/7 \Rightarrow 0{,}4 * 0{,}15 * 1{,}21 * (68 + 172) * 8 * 5/7 = 99{,}57€$

- Coste de la estancia para el desarrollo. En este proyecto los gastos realizados en alquilar el lugar de trabajo se reducen a 0 euros, ya que se cuenta con el hogar del analista como sitio donde realizar el desarrollo.

2.8.3. Gastos de herramientas para el desarrollo

En función de las herramientas usadas podemos incurrir en gastos en la adquisición de licencias de software:

- Eclipse, necesaria para el desarrollo del aplicativo en java, se distribuye bajo licencia "EPL" y su coste es de *0€*.
- Apache Tomcat, es un servidor de paginas web que nos permitirá la visualización de los jsp generados para el despliegue del aplicativo en SIETTE, su licencia es "ASF" y su coste es de *0€*.
- Lyx, editor de textos para la escritura de la documentación tiene licencia "GNU" y su coste es de *0€*.
- TortoiseHG, repositorio o almacén del código fuente, se distribuye bajo licencia "GNU" y su coste es de *0€*.
- BitBucket, repositorio en la nube con licencia "GNU" su coste es de *0€*.
- OpenOffice, generador de documentos de oficina editor de textos y tablas, tiene licencia "GNU" y su coste es de *0€*.

2.9. Gasto estimado en la realización del proyecto

Por último sumando las diferentes partidas de gastos, obtenemos el coste total de este proyecto.

- Gastos de personal, 17.120 €.

- Gastos del sistema informático y de mantenimiento, 1.499,57 €.
- Gastos de herramientas para el desarrollo, 0 €.

Sumando todas las partidas y estimando un beneficio de un 30 % obtenemos que el coste del proyecto será de 24.205,44€.

Capítulo 3

Análisis y diseño del sistema

3.1. Casos de uso

El diseño de los casos de uso nos permitirá definir los actores que interactuarán con el sistema y los escenarios posibles de aplicación. Para ello, se realizará un diagrama de casos de uso junto a la descripción del mismo.

En este diagrama se representan las funcionalidades y comportamientos del sistema y su interacción con los dos actores de nuestro aplicativo, “alumno” y “profesor”

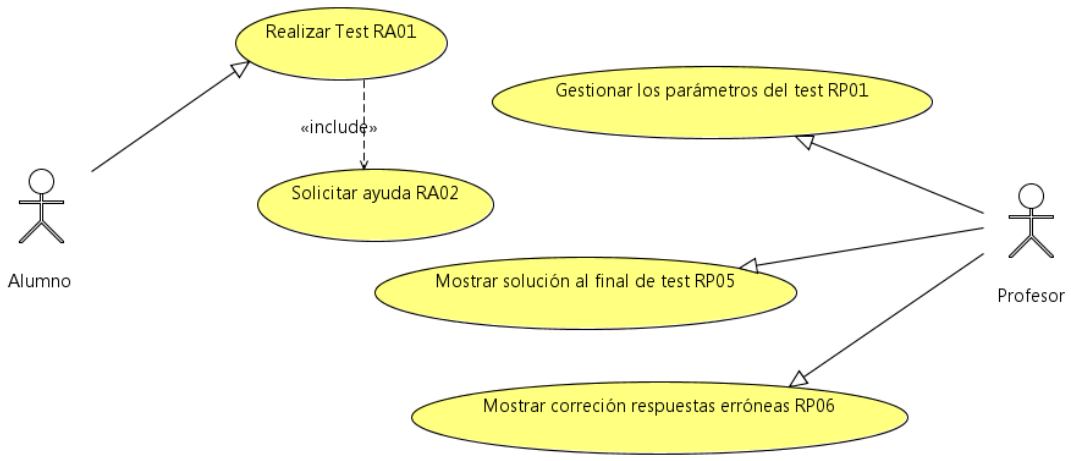

Figura 3.1.1: *Diagrama de casos de uso*

3.1.1. Realizar test

	Descripción
Nombre	Realización de test
Descripción	El alumno recibe una secuencia de preguntas por pantalla con tres opciones posibles
Actores	El Alumno o un conjunto de ellos.
Precondiciones	Ninguna
Flujo normal	El alumno inicia el test, recibe una pregunta con 3 opciones posibles, marca una de las opciones y continua con la realización del test.
Flujo alternativo	El alumno, desconoce la opción correcta y pulsa el botón de ayuda. El sistema le ofrece una ayuda para la resolución del ejercicio.
Poscondiciones	Al final del ejercicio al menos una de las opciones ha sido seleccionada.

Cuadro 3.1: *Caso de uso "Realizar test"*

3.1.2. Gestionar los parámetros del test

	Descripción
Nombre	Gestionar parámetros del test
Descripción	Permite modificar la dificultad de los ejercicios modificando los parámetros admitidos por estos.
Actores	El profesor de la asignatura.
Precondiciones	Ninguna
Flujo normal	▪ El profesor accede al sistema SIETTE. ▪ Dentro de cada pregunta del test, existe un jsp donde se invoca a la librería que contiene el ejercicio. ▪ Mediante el uso del manual incluido en el capítulo 7, modifica los valores de los parámetros de entrada de los ejercicios. ▪ Guarda las modificaciones realizadas en el jsp y el sistema automáticamente en la próxima ejecución de dicha pregunta la muestra con los nuevos parámetros
Flujo alternativo	No se aplica.
Poscondiciones	El profesor ajusta cada uno de los ejercicios en función del nivel del alumnado.

Cuadro 3.2: *Caso de uso "Gestionar parámetros del test"*

3.1.3. Mostrar solución al final del ejercicio

	Descripción
Nombre	Mostrar solución al final del ejercicio
Descripción	Permite modificar la resolución mostrada cuando se ha terminado por completo el test
Actores	El profesor de la asignatura.
Precondiciones	Ninguna
Flujo normal	El profesor accede al sistema SIETTE y dentro de las opciones sobre el test generado, marca o desmarca mostrar la resolución del ejercicio. Es obligatorio que todos los ejercicios tengan implementada una resolución detallada.
Flujo alternativo	No se aplica.
Poscondiciones	El profesor define sí desea realizar una explicación pormenorizada en clase, o que la explicación sea dada por el sistema

Cuadro 3.3: *Caso de uso "Mostrar refuerzo en respuestas erróneas"*

3.1.4. Mostrar refuerzo en respuestas erróneas

	Descripción
Nombre	Mostrar refuerzo en respuestas erróneas.
Descripción	Permite modificar la resolución mostrada cuando se ha terminado por completo el test, mostrando el porqué son erróneas las respuestas marcadas
Actores	El profesor de la asignatura.
Precondiciones	Ninguna
Flujo normal	El profesor accede al sistema SIETTE y dentro de las opciones sobre el test generado, marca o desmarca mostrar el refuerzo. Es obligatorio que todos los ejercicios tengan implementada una explicación de porque las opciones incorrectas, no son el resultado de dicho ejercicio
Flujo alternativo	No se aplica.
Poscondiciones	El profesor define sí el alumno debe investigar por su cuenta, que le indujo a cometer el error.

Cuadro 3.4: *Caso de uso "Gestionar parámetros del test"*

3.2. Análisis de paquetes

Después de la toma de requisitos y tras haber abordado el análisis de los mismos con el fin de recopilar la información suficiente para desarrollar este proyecto, establecemos una serie de paquetes básicos a partir de los cuáles construiremos el resto del proyecto.

En esta primera subdivisión perseguimos tres objetivos claros:

- Modularidad: descomponiendo problemas en subproblemas mas simples, aplicando una técnica fuertemente utilizada en informática el “divide y vencerás”.
- Cohesión: aunando en las mismas clases, paquetes, ... aquellos elementos que están relacionados funcionalmente.
- Bajo acoplamiento: buscando la máxima independencia entre cada una de las clases, lo que va a favorecer su modificación o ampliación.

Con el fin de alcanzar al máximo estos objetivos y en función de las funcionalidad realizadas por cada paquete de software debemos subdividir el problema principal, “diseñar un sistema generador de preguntas-respuestas de combinatoria” en los siguientes puntos:

- ¿Que tipos de datos y operaciones, utilizaría un generador de preguntas en el ámbito de la combinatoria?
- ¿Como se implementa un ejercicio?, ¿como se almacena su enunciado y respuestas?, ¿como se generan las respuestas correctas e incorrectas?.
- ¿Como se realizarían las figuras de cada ejercicio?

Los tres subproblemas enunciados, se implementan en los siguientes paquetes implicados.

3.3. Paquetes del aplicativo

El aplicativo está formado por tres paquetes principales, más aquellos paquetes que contienen los ejercicios:

- Paquete “operacion”: En el se definen los tipos de datos soportados por el aplicativo y las operaciones permitidas entre ellos. Números enteros en el caso de problemas de combinatoria.

- Paquete “ejercicioDef”: Define el esqueleto de todos los ejercicios desarrollados en este proyecto, en él se van a definir los métodos asociados a un ejercicio, como se muestra su enunciado, se genera la respuesta correcta y las incorrectas.

- Paquete “dibujoSVG”: Encargado de generar y gestionar cada una de las figuras que son mostradas en los distintos ejercicios, así mismo define propiedades y métodos que ayudan en la realización de los mismos. Como aquellos que permiten dividir un dibujo en diferentes viñetas para mostrar varias figuras en un mismo ejercicio.

Así mismo cada ejercicio está empaquetado en un solo paquete, de forma que la búsqueda de un error, o la ampliación del mismo sea sencilla y que los cambios producidos en un ejercicio no afecten a otro.

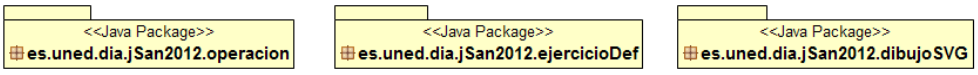

Figura 3.3.1: *Paquetes del aplicativo*

3.4. Paquete “operación”

El paquete operación contiene dos clases únicamente:

- La clase “OpEntero”, donde se define y estructura que es un número entero y las operaciones que tiene asociadas.

- La clase “OpComb” que establece e implementa las operaciones que pueden ser realizadas por un número entero en el cálculo de la combinatoria. Es una clase hija de la clase padre “OpEntero” y tiene acceso a gran parte de los métodos de su clase padre.

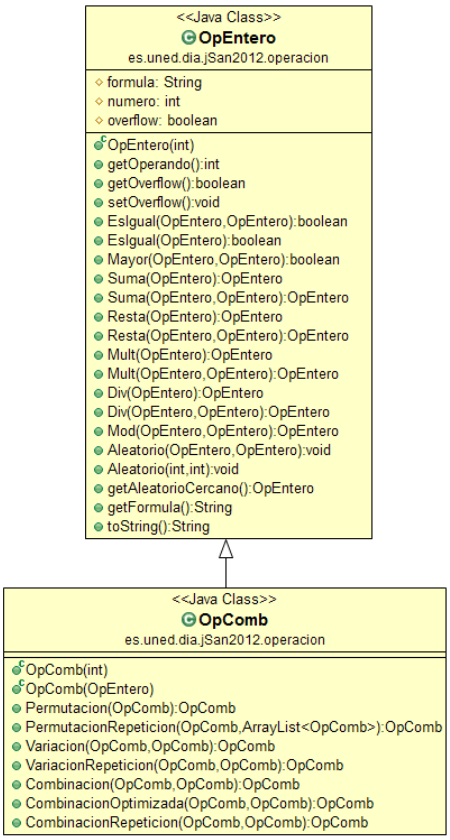

Figura 3.4.1: *Paquete "operación"*

3.4.1. Clase "OpEntero"

Implementa las posibles operaciones con números enteros y a su vez, encapsula un número entero dentro de ella. Sus propiedades son:

- "formula": almacena la fórmula derivada de las múltiples operaciones realizadas entre objetos de este paquete, por ejemplo sí tenemos un objeto de la clase opEntero fruto de la multiplicación de otros dos objetos de la misma clase en esta propiedad encontraríamos la siguiente cadena "$3 * 2$".

- "numero": almacena el valor entero del número encapsulado dentro del objeto.

- "overflow": guarda un valor booleano que indica si ha habido un desbordamiento en el número almacenado.

Sus métodos se subdividen en dos tipologías, aquellos cuyo resultado devuelve un nuevo objeto de la clase “opEntero” con el resultado de la operación realizada y aquellos que el resultado la anterior es almacenado en el objeto que los invoco, por ejemplo:

- “a.Suma(b)”: modifica el objeto “a” almacenando el valor de la operación suma con b.
- “x:=a.Suma(b,c)”: a “x” se le asigna un puntero al objeto resultado de la operación suma “b” con “c”.

En función de las operaciones realizadas, existen métodos para sumar, restar, multiplicar y dividir números enteros.

Del resto de los métodos implementados es conveniente destacar los siguientes:

- EsIgual: devuelve cierto cuando ambos número son iguales.
- Mayor: devuelve cierto cuando el número almacenado en el objeto “a” es mayor que el número almacenado en el objeto “b”.
- Mod: devuelve el módulo de dos objetos de la clase “OpEntero”.
- Aleatorio(x,y): devuelve un objeto que contiene un número entero aleatorio que estará comprendido entre la franja $x..y$
- getAleatorioCercano: retorna un objeto que almacena un número aleatorio cercano al del objeto que lo invocó, para ello calcula el número de cifras del número almacenado en el objeto y en función del mismo establece mediante una función cual es la máxima diferencia permitida con el mismo. Ej para 100 un aleatorio cercano sería 105 en el caso de 15.088 sería 15.150.

3.4.2. Clase “OpComb”

Implementa todas aquellas funciones relacionadas con el cálculo de la combinatoria, así mismo implementa un constructor de un objeto de la clase “OpComb” a través de un objeto de la clase de su padre.

- "Permutacion": calcula la permutación de un número. $n!$
- "PermutacionRepeticion": calcula las permutaciones con repetición de un número y una lista de números que indican el número de repeticiones. ${P_k}^{r1,r2,r^n}$
- "Variacion": calcula las variaciones de dos números a y b. $V_{a,b}$
- "VariacionRepeticion": calcula las variaciones con repetición de dos números a y b. $VR_{a,b}$
- "Combinacion": calcula las combinaciones de dos números a y b. $Comb_{a,b}$
- "CombinacionRepeticion": calcula las combinaciones con repetición de dos números a y b. $CombR_{a,b}$
- "CombinacionOptimizada": calcula las combinaciones de dos números a y b de manera optimizada, para ello previo al cálculo de los factoriales existentes en el numerador y en el denominador, simplifica los mismos eliminándolos de ambas partes. Por ej.:$Comb_{5,2} = \frac{5!}{2!(5-2)!} = \frac{5*4*3*2*1}{(2*1)*(3*2*1)} = \frac{5*4}{2*1}$

3.4.3. Gestión del desbordamiento

Cuando trabajamos con números procedentes del cálculo de la combinatoria, es frecuente que sean de gran tamaño provocándose el desbordamiento en el caso de números enteros en java. Recordemos que en java el máximo número entero es $2^{32} \simeq 4*10^9$; como java utiliza un bit para el signo, un número entero en java estará comprendido entre $2^{31} \simeq 2*10^9$ y $-2^{31} \simeq -2*10^9$. Pero, ¿qué ocurre si llegados al extremo de los enteros positivos le sumamos 1?, absolutamente nada, el valor almacenado pasa a ser el extremo negativo y java no lo interpreta como un error de desbordamiento.

Con el fin de evitar las situaciones anteriores, cada vez que se realiza una operación aritmética se comprueba el resultado de la misma por si fuera incoherente y por tanto, se ha producido un desbordamiento. Sí el desbordamiento es detectado, se almacena "-1" en el campo número del objeto y el campo "overflow" pasa a la posición true.

3.5. Paquete “ejercicioDef”

El paquete “ejercicioDef” es el encargado de gestionar e implementar la estructura de los ejercicios del aplicativo, así como de manejo de los parámetros de entrada que recibe cada ejercicio, está compuesto de las siguientes clases:

- “Parametro”: define la estructura de los campos de entrada que admiten los ejercicios.
- “LstParametro”: aúna en una lista un conjunto de parámetros.
- “EjParametro”: almacena para cada ejercicio, una lista de los parámetros de entrada.
- “EjToHtml”: facilita la conversión de los ejercicios de formato texto a formato html.
- “Ej”: define e implementa las características de un ejercicio completo.

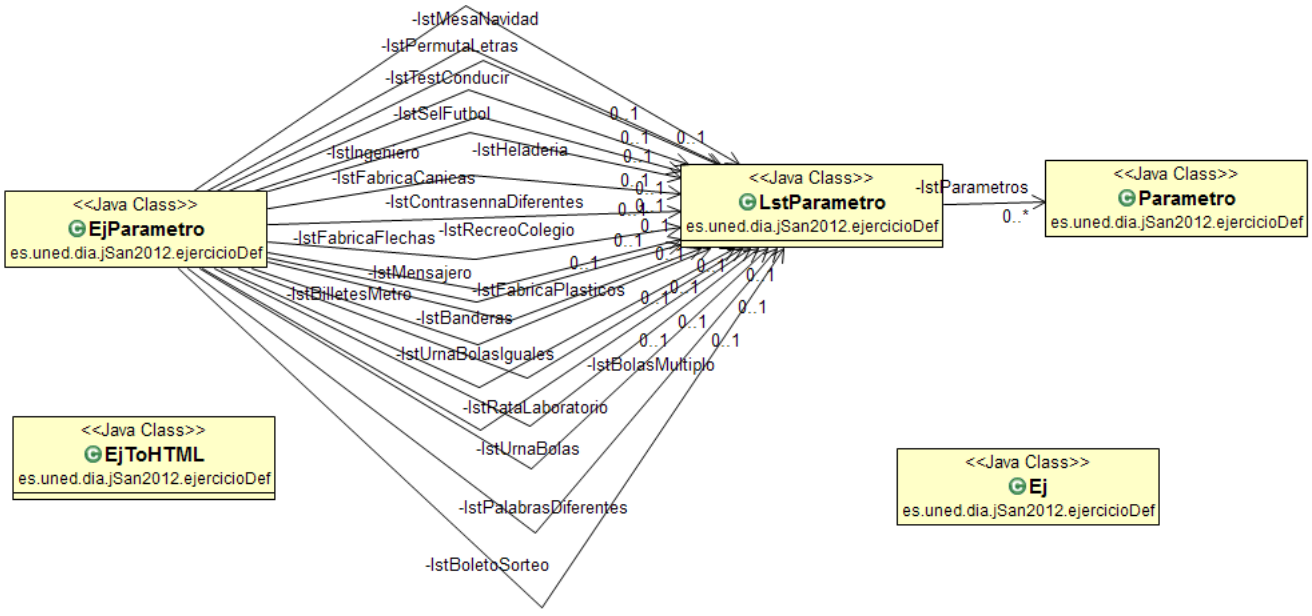

Figura 3.5.1: *Paquete “ejercicioDef”*

3.5.1. Clase “Parametro”

Uno de los requisitos funcionales del aplicativo es la adaptabilidad a diversos estudiantes para ello los ejercicios admiten parámetros y esta clase implementa las propiedades y métodos de los mismos.

En la figura inferior observamos la estructura de la clase “parametro”, la cuál posee tres propiedades que fundamentan la raíz de su utilidad:

- “nombre”: indica el nombre del parámetro.
- “extI”: extremo inferior del ejercicio, valor inferior que admite el parámetro.
- “extF”:extremo superior del ejercicio, valor superior que admite el parámetro.

Cada vez que se invoca un ejercicio recibe una lista de estos parámetros, con el valor de los dos extremos se genera un número aleatorio comprendido entre ambos que será utilizado como parámetro de entrada al ejercicio.

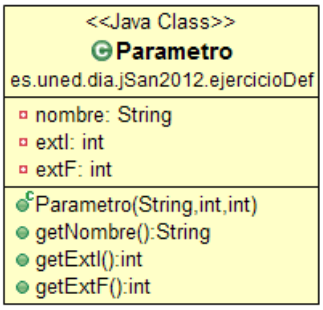

Figura 3.5.2: *Clase “Parametro”*

3.5.2. Clase “LstParametro”

Su función principal es encapsular en una lista los parámetros que serán utilizados posteriormente, como método de entrada en la invocación de cada uno de los ejercicios implementados. Los métodos que posee permiten recuperar o almacenar un parámetro, así mismo también existe un método para devolver el número de parámetros almacenados por un objeto de esta clase.

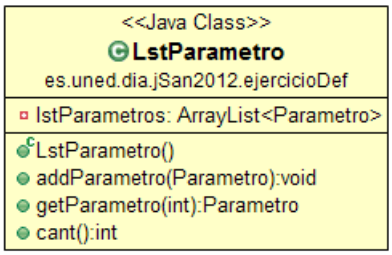

Figura 3.5.3: *Clase "LstParametro"*

3.5.3. Clase "EjParametro"

Gestiona cada una de la lista de parámetros que sirve de entrada a cada uno de los ejercicios, mediante sus métodos implementados almacena una nueva lista de parámetros de un ejercicio o bien recupera la lista de parámetros del mismo. Su método principal es "getLstParam" que mediante un entero que identifica el número de ejercicio, devuelve una lista con los parámetros del mismo.

Figura 3.5.4: *Clase "EjParametro"*

3.5.4. Clase “EjToHtml”

Con esta clase se simplifica la creación del texto de los ejercicios, mediante ella se realiza una gestión cómoda y sencilla del paso de un ejercicio en formato texto a formato HTML.

Su método de trabajo es el siguiente:

- Se crea un objeto de tipo “EjToHtml”.
- Se le pasa como argumento el texto que deseamos convertir en formato de cadena al método “retornaHTML”.
- El método “retornaHTML” invoca los dos métodos privados de la clase “generaAcentos” e “incluyeTabulador”, estos dos métodos lo que hacen es convertir a código HTML la cadena de entrada en función de sí encuentra un vocal acentuada o un tabulador.

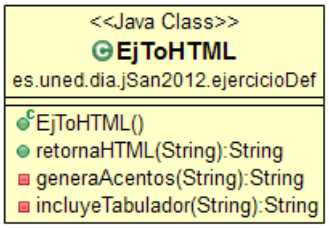

Figura 3.5.5: *Clase “EjToHtml”*

3.5.5. Clase “Ej”

Es la clase base de todos los ejercicios contenidos en el aplicativo, todos los ejercicios de la aplicación heredan sus propiedades y métodos. Sus principales propiedades son las siguientes:

- “enunRespInc1” y “enunRespInc2”: almacena los datos de las respuestas incorrectas, dentro del aplicativo puede ser: un número, una formula o un número expresando en formato de horas, minutos y segundos.
- “enunRespCorr”: guarda el dato de la respuesta correcta, al igual que en la propiedad anterior puede ser un número, una fórmula o una cantidad de tiempo.

- “enunRespCorrForm”: en este caso sólo se almacena el enunciado de la fórmula de la respuesta correcta.
- “ayudaEnunciado”: En el momento que el usuario solicita la ayuda de un ejercicio se muestra la ayuda contenida en este campo.
- “ayudaRespCorr”, “ayudaRespInc1”, “ayudaRespInc2”: corresponde con cada uno de los enunciados que serán mostrados al visualizar la solución del test de cada uno de los ejercicios.
- “pemiteFormulacion”: en cierto tipo de ejercicios el enunciado presentado al alumno puede ser más complejo, pero la visualización de una fórmula directa, impediría el objetivo del ejercicio, el aprendizaje. Es por ello que existe esta propiedad y que cuando está establecida a false impide que se muestre la formulación en el enunciado de las respuestas y sólo aparezca el resultado de forma numérica.
- “formatoTiempo”: al igual que en el parámetro anterior, permite aumentar la dificultad del ejercicio, para ello muestra la solución en formato de días, horas, minutos y segundos.
- “op”: se usa en cálculos auxiliares.
- “opSolucion”: es la propiedad que almacena el resultado del ejercicio a través de un objeto “OpComb”.
- “respIncor1”, “respIncor2”: almacena los resultados incorrectos mostrados al alumno.
- “enunciadoPregunta”: establece el enunciado del ejercicio mostrado al alumno.

Dada la gran cantidad de opciones que posee la clase ejercicio, dispone de varios métodos que ayudan a la generación y configuración de los mismos, destacando los siguientes:

- “generaRespuestas”: Se encarga de calcular mediante medios probabilistas sí un ejercicio será presentado mediante fórmulas, número o rango de tiempo, por supuesto sólo serán presentados las respuesta en formato de fórmula matemática aquellos ejercicios que tengan establecido a “true” la propiedad “permiteFormulacion”. También se encargará de

generar la respuestas en formato tiempo en las ejercicios que esté habilitado esta propiedad.

- "getEnunciadoPregunta": En función de el parámetro de entrada sea cierto o falso devuelve o bien una cadena con el enunciado de la pregunta o bien el código HTML de dicho enunciado.
- "getAyudaEnunciado": Devuelve la cadena almacenada con la ayuda del enunciado.
- "getAyudaRespCorr", "getAyudaRespInc1", "getAyudaRespInc2": devuelven la corrección del ejercicio una vez completado el test y en función de la respuesta marcada.
- "getEnunRespCorr", "getEnunRespInc1", "getEnunRespInc2": Retorna el valor de la cadena almacenado en cada uno de los campos de igual nombre.
- "getEnunciadoRespCorrForm": devuelve la fórmula correspondiente a la solución del ejercicio.
- "generaEnunciadoRespuesta": recibiendo como entrada los parámetros, número de respuesta incorrecta, sí se desea una salida en formato fórmula o bien sí se desea en formato tiempo, genera una respuesta incorrecta cercana a la respuesta correcta.
- "dameFormatoTiempo": recibe como parámetro un número entero y devuelve como resultado dicho número en formato de días, horas, minutos y segundos.
- "simplificaRespuesta": es usada cuando la respuesta está en modo formulación para simplificar aquellos operandos que no aportan nada nuevo a la fórmula, por ejemplo multiplicar por uno, sumar cero al resultado etc.

Las siguiente cuatro funciones son utilizadas en el método anterior para generar respuestas incorrectas en modo formulación muy parecidas a la solución del ejercicio.

- "cuentaCaracter": devuelve el número de ocurrencias de un carácter en una cadena.
- "cuentaNumero": devuelve el número de dígitos de una cadena.

- "reemplazaCarPos": reemplaza la ocurrencia "n" del carácter "x" por otro suministrado como parámetro.
- "remplazaNumero": remplaza la ocurrencia del dígito "x" por otro suministrado como parámetro.

La forma de trabajo de las anteriores funciones es a través del método "generaEnunciadoRespuesta". El mismo cuando tiene que generar una respuesta incorrecta, comprueba primero sí es en formato fórmula, tiempo o numérica. En el primer caso cuenta caracteres como el operador suma o multiplicación y aleatoriamente sustituye alguno de los que encuentre. En el segundo caso cuenta las ocurrencias de dígitos en la cadena y aleatoriamente sustituye alguno con otro valor aleatorio.

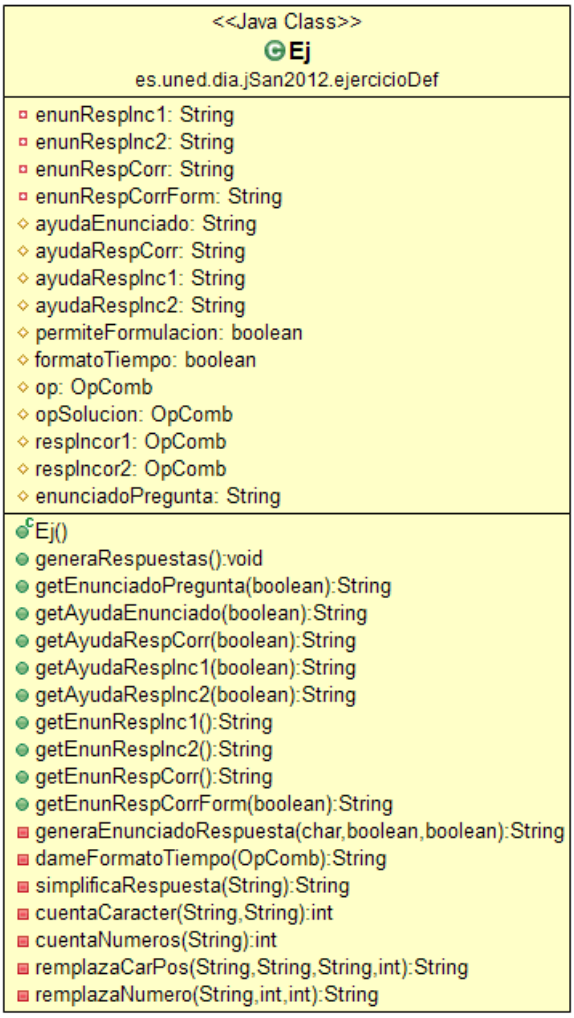

Figura 3.5.6: *Clase "Ej"*

3.6. Paquete “dibujoSVG”

Uno de los atractivos del presente proyecto es su parte gráfica, como se ha comentado en puntos anteriores cada ejercicio incluye un dibujo descriptivo del enunciado. Con este fin se ha generado el siguiente paquete que incluye todas las funcionalidades necesarias para desarrollar dibujos sencillos y figuras a modo de ayuda al alumno. En función del tipo utilidad añadida podemos distinguir 4 grupos distintos:

3.6.1. Superclase

- Clase “ToSVG”: es la clase padre de todas las figuras geométricas desarrolladas, es una clase abstracta y define el método abstracto “toSVG”, el cuál debe ser implementado obligatoriamente por cada una de sus clases hijas y este devolverá en una cadena el código SVG generado para cada figura geométrica.

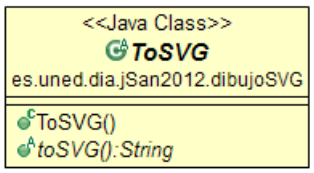

Figura 3.6.1: *Clase “ToSVG”*

3.6.2. Clases que dibujan figuras geométricas

En este grupo encontramos todas aquellas clases, que facilitan la tarea de dibujo en cada uno de los ejercicios. Todas las clases heredan de la clase “toSVG” e implementan una figura geométrica distinta, también se incluye una clase que añade funcionalidades a las figuras geométricas la clase “Animacion”, encarga de generar movimiento en las figuras.

Las clases que dibujan figuras básicas son las siguientes:

- Clase “Circulo”.
- Clase “Linea”.

- Clase “Poligono”.
- Clase “Rectangulo”.
- Clase “Trapecio”.
- Clase “Texto”.

El funcionamiento de cada una de las clases, se basa en la tarea de simplificar el dibujo de figuras, como podemos observar en la clase polígono tenemos un constructor que define parámetros como: las coordenadas del polígono, el tamaño del borde, color del borde y color de relleno. Mediante estos parámetros dicha clase genera un código HTML en formato SVG de la figura, que es retornado mediante la invocación a su método “toSVG”.

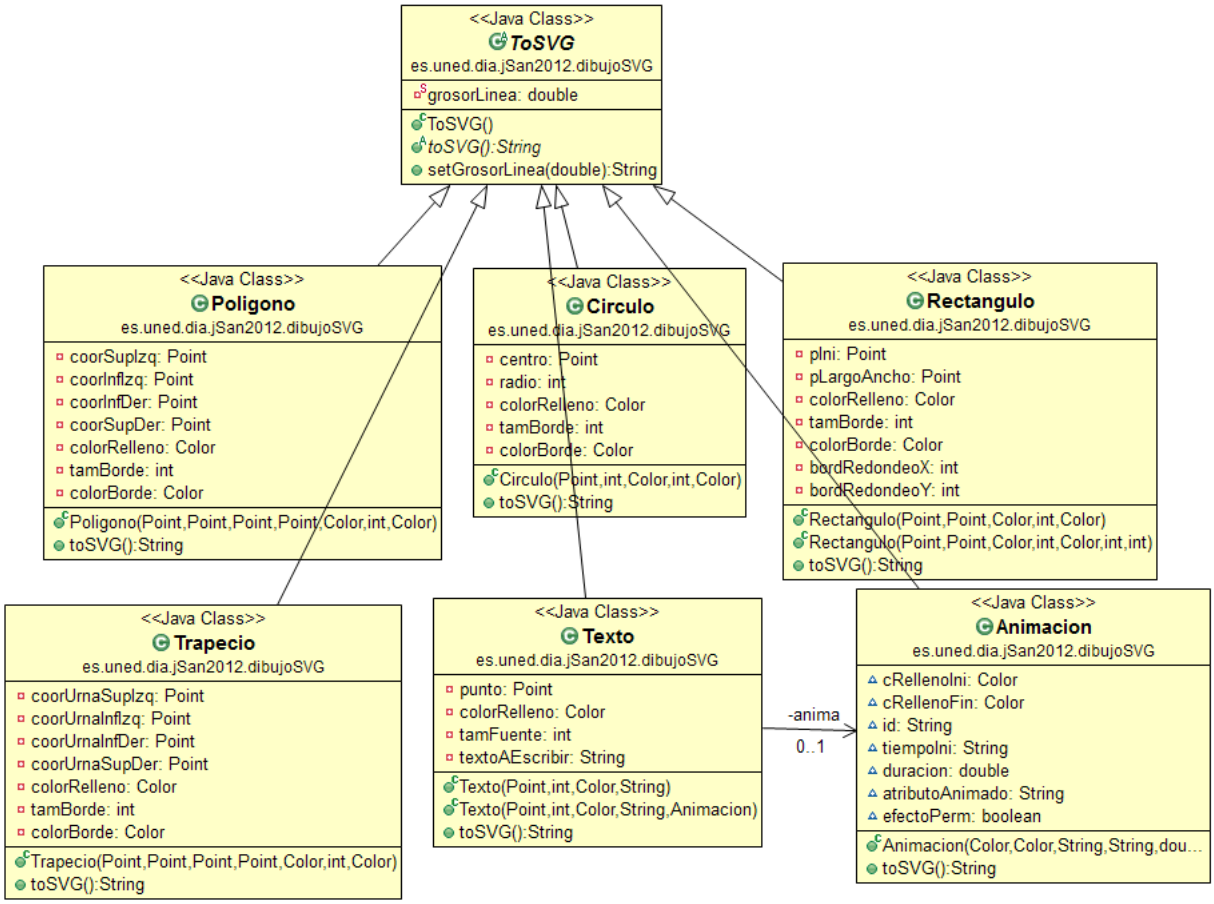

Figura 3.6.2: *Diagrama de Figuras SVG*

Respecto a a la clase animación define los parámetros admitidos para realizar la animación

de un texto, para ello se definen parámetros esenciales como el inicio en tiempo de la animación la duración de la misma y que atributo va a ser el objeto de la animación.

3.6.3. Clases que permiten la comparación de figuras o viñetas

Cada ejercicio que compone el sistema es acompañado por dibujo, el cual es gestionado mediante las clases siguientes que lo encapsulan y manejan.

- Clase “Figura”: define y almacena todos los aspectos necesarios para la generación de un dibujo sus propiedades son las siguientes:

  - “pExtInf” y “pExtSup”: establecen mediante dos puntos la esquina inferior izquierda y superior derecha del dibujo.
  - “marco”: indica sí el valor es cierto que debe dejarse un espacio entre el dibujo y la figura que lo contienen, un ejemplo de tal propiedad sería el marco de un cuadro.
  - “nomFigura”: mediante una cadena se le puede dar un nombre a la figura.
  - “espacioMarco”: establece el espacio que debe existir entre el marco de la figura y el dibujo contenido.

- Clase “FigImg”: Heredando de la clase padre “Figura” obtiene todos los parámetros y métodos de su superclase. Así mismo añade todos los métodos que permiten dibujar figuras geométricas en un dibujo.

- Clase “Dibujo”: Define y almacena las distintas figuras que son mostrados en un ejercicio. Cuando se crea un objeto de la clase dibujo se invoca a su método privado “generaFiguras”, el cual se encarga de calcular el tamaño de las figuras contenidas mediante los parámetros indicados en su constructor.

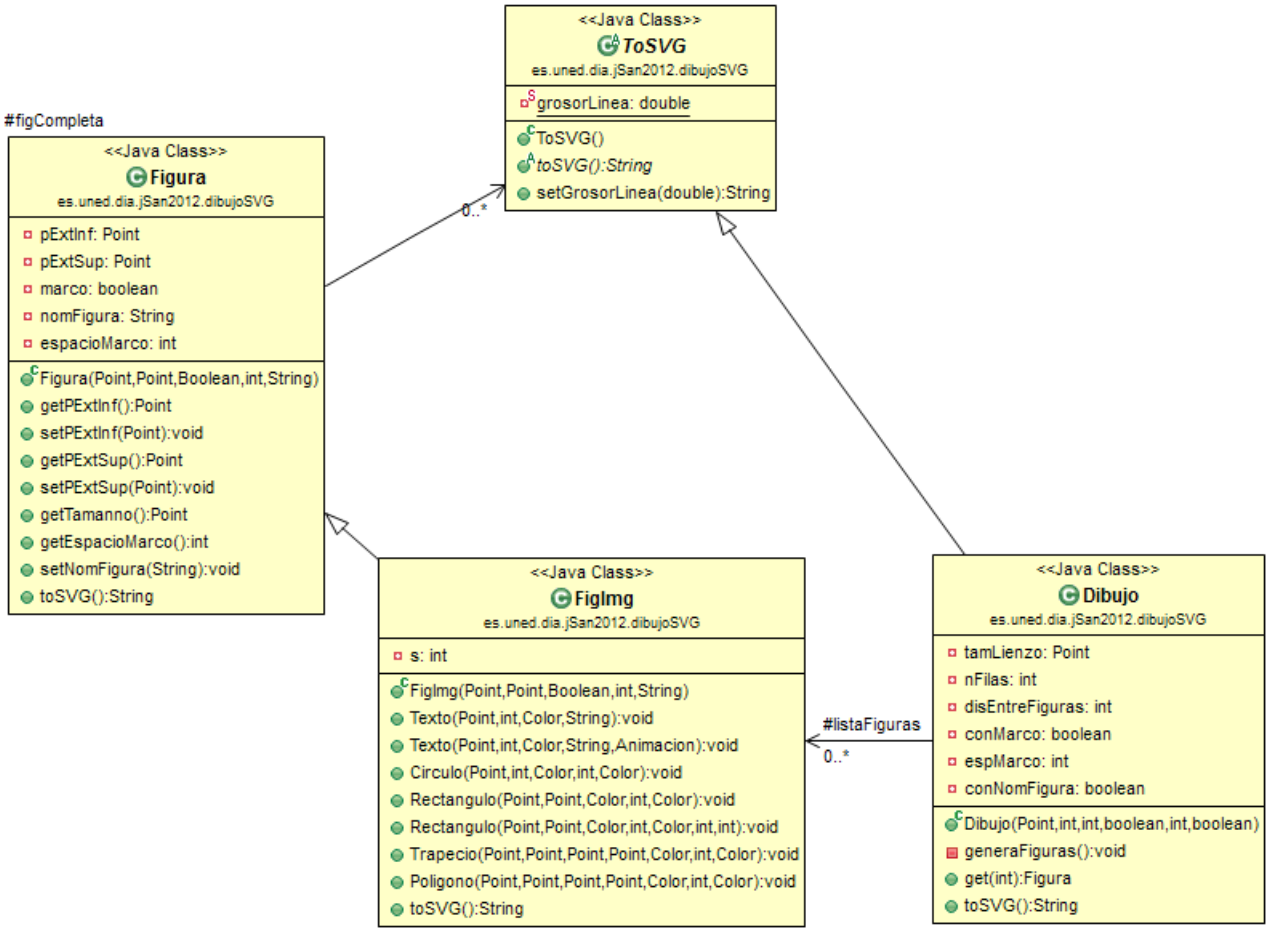

Figura 3.6.3: *Diagrama de figuras o viñetas SVG*

3.6.4. Clases que regulan los atributos de las figuras geométricas

En este apartado encontramos una sola clase, la clase color. Esta clase define el nombre de cada uno de los colores admitidos por SVG.

SVG permite utilizar en sus atributos de dibujo, códigos para los colores o bien nombres de color. La clase color añade otra funcionalidad mediante su método “getColorComb”, que devuelve un color cuando es invocado con un parámetro entero. Este método es utilizado en multitud de ejercicios, dado que los colores están seleccionados según el contraste que existe entre cada color.

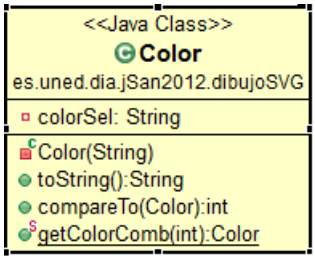

Figura 3.6.4: *Clase Color*

3.7. Estructura del aplicativo

En anteriores puntos se ha desarrollado la estructura o esqueleto donde se asienta el sistema generador, en este punto analizamos del aplicativo ya completo, donde se incluyen todos los paquetes que lo conforman, en función de su tipología distinguimos los siguientes subtipos:

- Paquetes “dibujoSVG”, “ejercicioDef” y “operacion” comentado en los puntos anteriores.
- Paquetes que empiezan por “es.uned.dia.jSan2012.ej.*”, son los paquetes que almacenan cada uno de los ejercicios del sistema, existen en total 20 y cada uno suele estar formado por una clase que hereda de la clase “Ej” más una o dos clases auxiliares que se encargan de generar el dibujo del ejercicio.
- Paquete “urnaConBolas”, existen en el aplicativo varios ejercicios con urnas y bolas, dentro de este paquete se definen aquellas clases que facilitan su dibujo.

- Paquete “test”, contiene los test que son realizados sobre el paquete “operacion” para comprobar su exactitud y estabilidad.

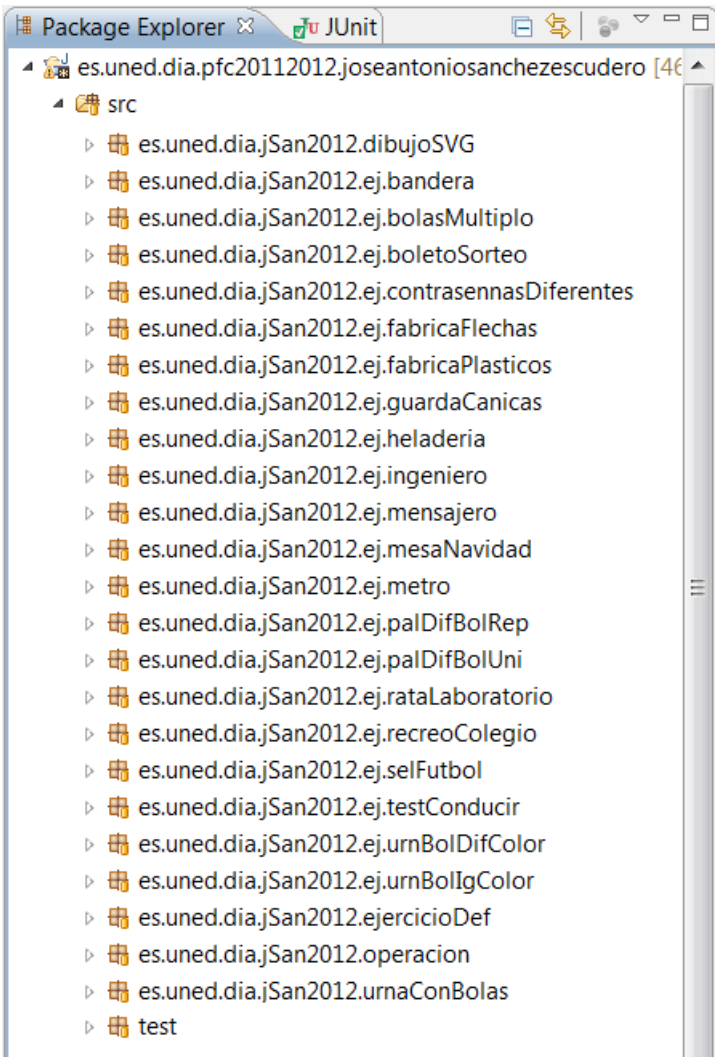

Figura 3.7.1: *Estructura del aplicativo*

Capítulo 4

Ejercicios Implementados

4.1. Introducción

En el diseño de este sistema, hemos intentado elaborar ejercicios adecuados para alumnos de un primer cuatrimestre de Matemática Discreta o para aquellos estudiantes, que estén realizando una asignatura con carga de combinatoria.

El estudio de la matemática de por si, suele ser bastante abstracto y a veces confuso para el alumnado. Por tanto, a la hora de desarrollar los presentes ejercicios hemos intentado adaptarlos a un entorno cercano al mundo real.

Es esta nuestra principal motivación y es por ello que los ejercicios no están distribuidos equitativamente, sino por los factores anteriores. Aún careciendo de una distribución equitativa, es interesante destacar que la gran mayoría de los mismos combinan varias técnicas sobre el cálculo combinatorio, siendo necesario el manejo de todo el temario para su correcta resolución.

Una vez definidos los objetivos principales en el desarrollo de los ejercicios, hemos distribuido los mismos ajustándolos a la carga docente de una unidad básica de combinatoria.

4.1.1. Definición de combinatoria

La Combinatoria es la parte de las Matemáticas que se dedica a buscar procedimientos y estrategias para el recuento de los elementos de un conjunto o la forma de agrupar los elementos

de un conjunto.

4.2. Regla de la suma

Principio de adición. Para contar los elementos de dos conjuntos, basta con sumar el número de elementos de cada uno de los conjuntos y posteriormente en el caso que tengan elementos en común, sustraer la intersección de los mismos.

$A \cap B \neq \emptyset \Longrightarrow cardinal(A) + cardinal(B) - cardinal(A \cap B)$

4.2.1. Calcular el número de múltiplos posibles

Se introducen en una urna “n” bolas etiquetadas cada una con un número (ver figura A). Posteriormente, se extraen dos bolas de una en una de la urna y se suma el resultado. ¿De cuántas formas se puede obtener un múltiplo de “m” o de “p”?.

Mediante este ejercicio vamos a practicar los fundamentos de la regla de la suma; para ello, debemos realizar los siguientes pasos:

- Calcular todos los posibles conjuntos de 2 elementos formados por la extracción de dos bolas consecutivas.
- Calcular cuales de ellos son múltiplos de ”m”
- Calcular cuales de ellos son múltiplos de ”p”
- Calcular cuales de ellos son múltiplos de $m \cap p$
- Aplicar la fórmula de la suma: $\Longrightarrow cardinal(m) + cardinal(p) - cardinal(m \cap p)$

Ejemplo: Se introducen en una urna 6 bolas etiquetadas cada una con un número (ver figura A). Posteriormente, se extraen dos bolas de una en una de la urna y se suma el resultado" ¿De cuántas formas se puede obtener un múltiplo de 2 o de 4?

- Posibles conjuntos de 2 elementos: ((1,2),(1,3),(1,4),(1,5),(1,6), (2,1),(2,3),(2,4),(2,5),(2,6), (3,1),(3,2),(3,4),(3,5),(3,6), (4,1),(4,2),(4,3),(4,5),(4,6), (5,1),(5,2),(5,3),(5,4),(5,6), (6,1),(6,2),(6,3),(6,4),(6,5))
- Múltiplos de 2: ((1,3),(1,5),(2,4),(2,6),(3,1),(3,5),(4,2),(4,6),(5,1),(5,3),(6,2),(6,4))
- Múltiplos de 4: ((1,3),(2,6),(3,1),(3,5),(5,3),(6,2))
- Múltiplos de 2 y de 4: ((1,3),(2,6),(3,1),(3,5),(5,3),(6,2))

Por tanto, la solución sería: 12+6-6=12

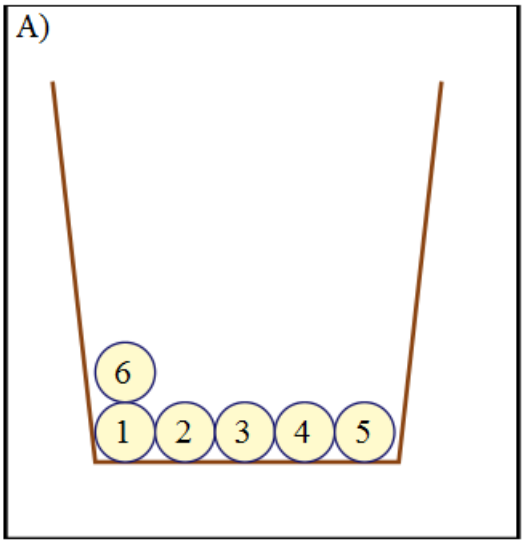

Figura 4.2.1: *Calcular múltiplos posibles*

4.3. Regla del producto

Principio de multiplicación. Para contar los elementos de un conjunto, donde sus elementos están formados por pares de elementos y en los que el primer elemento pertenece al primer conjunto y el segundo elemento al segundo conjunto (producto cartesiano), se multiplica el número de elementos de cada conjunto.

$cardinal(A * B) = cardinal(A) * cardinal(B)$

4.3.1. Ingeniero Electrónico

Una Ingeniero electrónico debe comprobar con un sólo cable que un circuito funciona adecuadamente; para ello, une cada borne de la placa "A" con cada borne de la placa "B" y comprueba sí existe continuidad. Si para cada comprobación dedica "n" segundos, ¿cuánto tiempo le llevará la comprobación del circuito?

En este ejercicio tenemos un ejemplo de la aplicación directa del principio de la multiplicación. Si pensamos en una representación del problema, podemos establecer la siguiente notación: enumerar los bornes del circuito "A" con letras y los bornes del circuito "B" con números, de tal manera que una comprobación posible sería "C3", que significa que el tercer borne del circuito "A" está conectado con el tercer borne del circuito "B". Observando dicho ejemplo, extraemos la siguiente conclusión: el número de comprobaciones será igual al producto del número de bornes del circuito "A" por el número de bornes del circuito "B".

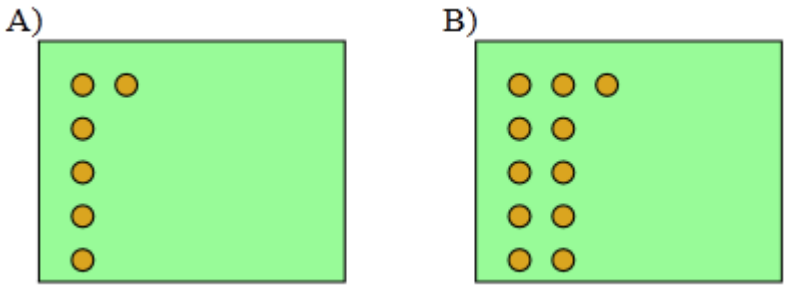

Figura 4.3.1: *Comprobación de circuitos*

4.4. Definición de factorial de un número

El factorial de un número se define como el producto de todos los números naturales comprendidos desde el "1" hasta el número seleccionado "n". Por lo tanto:

$n! = n * (n-1) * (n-2) \ldots 2 * 1$

En el campo de la combinatoria se define como las diferentes maneras de ordenar "n" elementos distintos.

4.4.1. Maestro de escuela

Un maestro está preparando a sus alumnos para salir al recreo y para ello los coloca en fila india. De repente le surge la siguiente cuestión: ¿de cuántas formas posibles se pueden ordenar los niños en una fila india?, ¿sabrías responder al maestro?.

Este ejercicio es una aplicación directa de la definición de factorial de un número, son las "n" posibles formas diferentes de ordenar "n" alumnos en una fila india.

$n!$

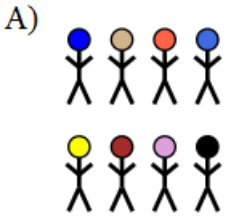

Figura 4.4.1: *Maestro de escuela*

4.5. Definición de número combinatorio

El número combinatorio se define como el coeficiente binomial generado por la siguiente fórmula:

$$\binom{n}{k} = \frac{n!}{k!*(n-k)!}$$

En el campo de la combinatoria se define como el número de formas en que se pueden extraer subconjuntos a partir de un conjunto dado (*Wikipedia*).

4.5.1. Fábrica de Materiales

Una fábrica diseña nuevos materiales mezclando diferentes compuestos químicos. Para ello, selecciona “n” ingredientes de “m” posibles (figura A). Con la llegada de un nuevo cliente, el fabricante desea generar una pequeña muestra de cada material. Si para cada muestra dedica “t” segundos, ¿de cuánto tiempo debe disponer para la producción de dichos materiales?.

Este ejercicio es una aplicación directa de la definición de número combinatorio, es decir, seleccionar “n” ingredientes de “m” posibles donde el orden no afecta a la selección.

El ejercicio se resume en aplicar la fórmula del número combinatorio y multiplicar por el tiempo que se tarda en generar cada material.

$Comb(m,n) * t$

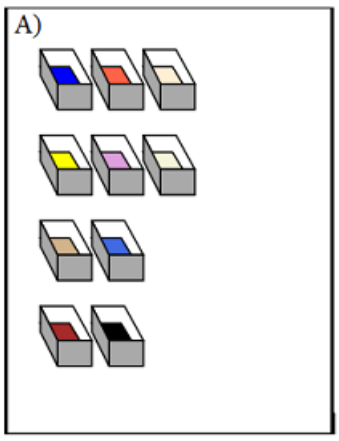

Figura 4.5.1: *Fábrica de materiales*

4.6. Variaciones

4.6.1. Variaciones sin repetición

4.6.1.1. Banderas Ukumundu

Los diseñadores del país de Ukumundu han recibido un nuevo encargo del ministerio de presidencia. Deben diseñar la bandera del reino.

Como única condición, el gobierno les ha impuesto que la bandera del reino estará formada por franjas horizontales y que los colores de las franjas no se repitan. A tal efecto, la bandera de la figura “A” y de la figura “B” se consideran diferentes. ¿Cuántas banderas diferentes pueden realizar los diseñadores de Ukumundu con “n” colores diferentes y un máximo de “m” franjas horizontales?.

En el presente ejercicio sí importa el orden tal y como nos indica las figuras; no es lo mismo la bandera que empieza en azul, que la que empieza en amarillo. Así mismo, tenemos que seleccionar grupos de “m” colores entre “n” posibles, por tanto nos encontramos ante un ejercicio de variaciones sin repetición.

$V_{m,n}$

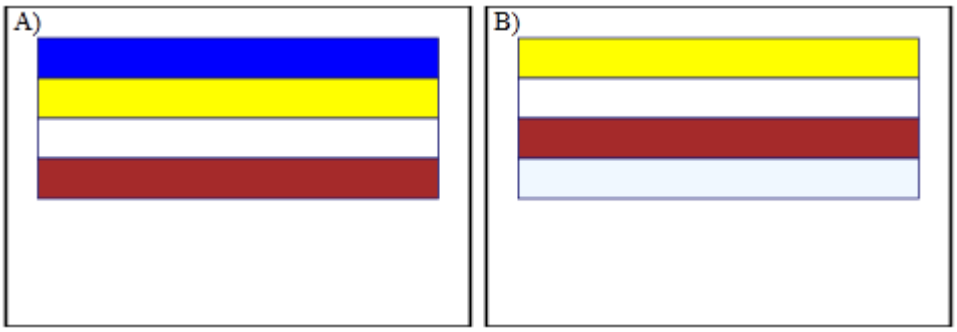

Figura 4.6.1: *Banderas Ukumundu*

4.6.1.2. Metro de Ukumundu

El Metro de Ukumundu dispone de “n” líneas y de las estaciones representadas con un círculo negro en el plano del metro (“x”). Para viajar en el mismo, es necesario sacar un billete donde figura la estación origen y la estación de destino pudiéndose realizar los trasbordos necesarios para llegar a la estación de destino. ¿Cuántos billetes distintos pueden existir en la red de metro de Ukumundu?.

En este ejercicio sí importa sí el orden, ya que dos billetes serán diferentes si tienen sus estaciones de origen y destino invertidas. Por otra parte, el número de líneas no aporta nada relevante para el cálculo solicitado. Por último, otro dato importante es que no puede haber billetes con la misma estación de origen y destino, serían absurdos.

Con los datos anteriores se extrae como conclusión, que para resolver el ejercicio hay que aplicar la fórmula de variaciones sin repetición.

$V_{x,2}$

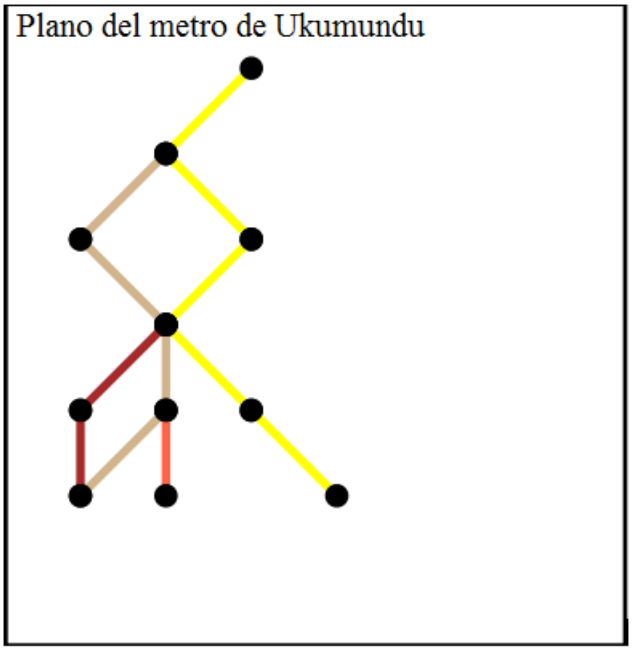

Figura 4.6.2: *Metro de Ukumundu*

4.6.1.3. Flechas Olimpiada de Ukumundu

Una fábrica produce flechas oficiales para la olimpiada de Ukumundu. Cada flecha está formada por una punta, una varilla y una pluma. Cada parte de la flecha es de un sólo color y la federación deportiva permite que la punta y la pluma puedan ser del mismo color, pero no así la varilla y la punta. Sí para la fabricación de cada flecha se destinan "n_{seg}" segundos, ¿cuánto tiempo se necesitará para producir "$n_{flechas}$" flechas de cada modelo sí se disponen "$n_{colores}$" colores diferentes de cada de elemento que forma la flecha?.

El problema puede ser resuelto fácilmente descomponiéndolo en dos subproblemas:

- Subproblema nº 1: número de colores permitidos en la fabricación de la punta y la varilla. En este subproblema importa el orden y no se permite la repetición, por tanto son Variaciones. $V_{n_{colores},2}$
- Subproblema nº 2: colores permitidos en la varilla con el resto del conjunto; no existen restricciones y el resultado es la multiplicación de los dos subproblemas (principio de la multiplicación).

$Subproblema_1 * Suproblema_2$

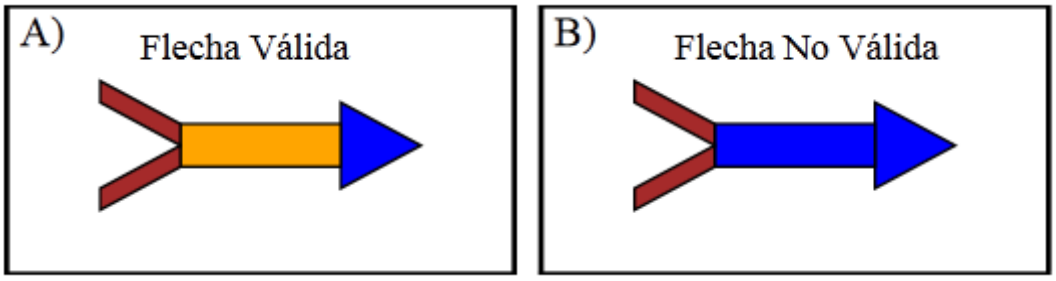

Figura 4.6.3: *Flechas Olimpiada Ukumundu*

4.6.2. Variaciones con repetición

4.6.2.1. Contraseñas Alpha Asociados

En la empresa “Alpha Asociados" se va a implantar una nueva política de seguridad. Cada usuario de “Alpha Asociados” tendrá una contraseña de entre “m” y “n” letras de tamaño, y los únicos caracteres permitidos son: $(X_0, X_1, .., X_{y-1}, X_y)$. ¿Cuántas contraseñas distintas admite el sistema?.

En este ejercicio si importa el orden dentro del grupo, pues no es lo mismo la contraseña “AB” que “BA”, por tanto son variaciones. Así mismo, como para cada posición del grupo se pueden utilizar todos los caracteres X_i, nos encontramos ante un ejercicio de variaciones con repetición.

Dentro de los caracteres permitidos, existe la restricción de que ningún componente del grupo es igual a otro del mismo.

$$\sum_{i=m}^{n} VR_{y,i} \Longrightarrow VR_{y,m} + VR_{y,m+1} + .. + VR_{y,n-1} + VR_{y,n}$$

Figura 4.6.4: *Contraseña Alpha Asociados*

4.7. Combinaciones

4.7.1. Combinaciones sin repetición.

4.7.1.1. Ratón de laboratorio en laberinto

Un ratón de laboratorio debe recorrer el laberinto de la figura desde el punto “A” donde se encuentra al punto “B” donde está ubicado el queso, evitando en su recorrido la estancia donde se encuentra el veneno. ¿Cuántas rutas diferentes puede tomar desde “A” hacia “B” sí solo puede desplazarse hacia arriba y hacia la derecha?¡

Cada ruta puede ser descrita como una cadena de bits donde los “1” indican subir hacia arriba y los “0”, moverse a la derecha. De este modo, el ejercicio puede reducirse a una cadena de B_x ceros +B_y unos, donde existen los siguientes caminos:

$Comb(B_x + B_y, B_x)$ (Dado que el punto A, está situado en (0,0))

Como existe una estancia donde se encuentra el veneno, debemos restar de las combinaciones totales las combinaciones que llevan desde el inicio hasta la estancia con el veneno, multiplicadas por las que salen de dicha estancia hasta el punto B. Por tanto, la solución del mismo sería:

$Comb(B_x + B_y, B_x)$-$(Comb(X_x + X_y, X_x) * Comb((B_x - X_x) + (B_y - X_y), (B_x - X_x)))$

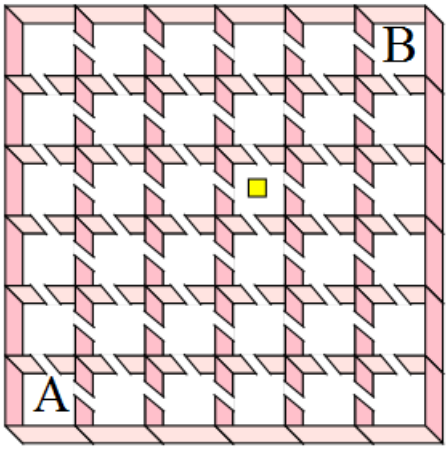

Figura 4.7.1: *Ratón de laboratorio en laberinto*

4.7.1.2. Mensajero

Un Mensajero de Londres debe ir desde su oficina situada en "A" hasta el punto de recogida "B", con el fin de recoger un paquete. Posteriormente, debe ir desde el punto de recogida "B" hasta el punto de recogida "C" y recoger otro paquete. Por último, debe depositar ambos paquetes en una oficina situada en "D". ¿Cuántas rutas diferentes puede tomar el mensajero pasando en orden por las oficinas A-B-C-D, sí solo le está permitido ir hacia arriba y hacia la derecha?

Al igual que en el ejercicio anterior, podemos describir la ruta que realiza el mensajero como una cadena de bits, donde los “1” indican subir hacia arriba y los “0” moverse a la derecha; por tanto, el ejercicio puede reducirse a una cadena de B_x ceros +B_y unos.

A diferencia de el ejercicio anterior, donde se calculaban todas las rutas posibles y se restaban las que iban o salían de la estancia con el veneno, en este ejercicio debemos calcular las diferentes rutas existentes entre cada punto de recogida.

$CombAB = Comb(B_x + B_y, B_x)$ (Dado que el punto A, está situado en (0,0)).

$CombBC = Comb((C_x - B_x) + (C_y - B_y), (C_x - B_x))$

$CombCD = Comb((D_x - C_x) + (D_y - C_y), (D_x - C_x))$

$CombABCD = CombAB * CombBC * CombCD$

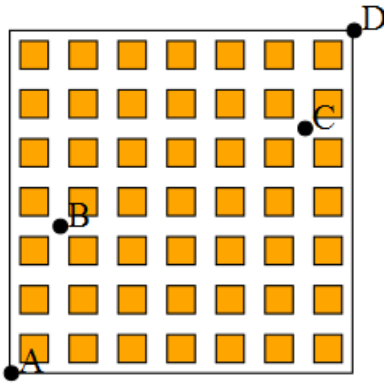

Figura 4.7.2: *Mensajero*

4.7.1.3. Lotería Ukumundu

Un boleto "sencillo" del sorteo especial de Ukumundu, está formado por tres columnas con números como los descritos en la figura. En el boleto sólo se permiten marcar ”n” números de la columna “A”, ”m” de la “B” y ”o” de la “C”. Llegado el sorteo, se extraen ”n” bolas con los números de la columna “A”, ”m” bolas con los de la columna B y ”o” bolas con los de la columna C que corresponden al primer premio. ¿Cuántos boletos de la lotería de Ukumundu habría que rellenar para asegurarse el premio de máxima categoría?

El problema se resume en la selección en un grupo de “n” números de “m” posibles, por tanto es un problema de combinaciones. Como no puede haber dos bolas con el mismo número, es un problema de combinaciones sin repetición. Por otra parte, tiene añadida una dificultad que es que existen tres combinaciones y para acertar el primer premio, es necesario que ocurran las tres a la vez. De modo que el resultado sería el siguiente.

$Comb(A_n, A_s) * Comb(B_n, B_s) * Comb(C_n, C_s)$

Siendo A_n: La cantidad de números de la columna “A”.

Siendo A_s: Las bolas extraídas del bombo de la columna “A”.

Siendo B_n: La cantidad de números de la columna “B”.

Siendo B_s: Las bolas extraídas del bombo de la columna “B”.

Siendo C_n: La cantidad de números de la columna “C”.

Siendo C_s: Las bolas extraídas del bombo de la columna “C”.

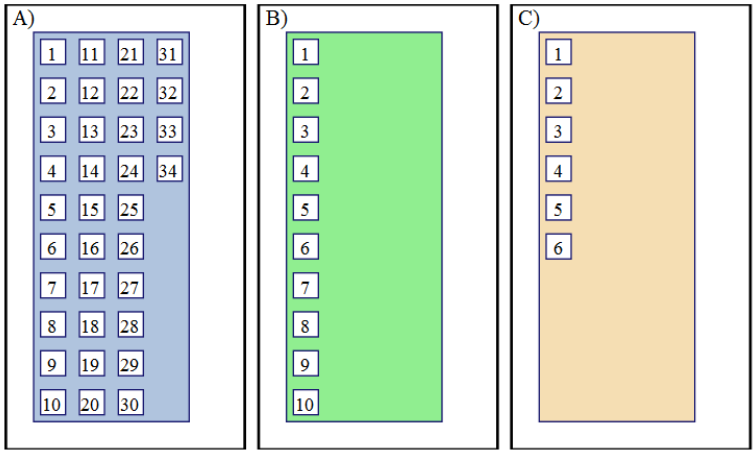

Figura 4.7.3: *Lotería Ukumundu*

4.7.1.4. Urnas con bolas de distinto color

Tenemos “n” urnas: "Urna X_1", "Urna X_2",.. "Urna X_n", . Cada urna contiene las bolas indicadas en la figura. Al extraer $iBolas$ de cada urna, ¿de cuántas formas posibles podemos extraer $iBolas * nUrnas$ de $unColorEspecífico$?.

En este problema siempre se cumple que número de Bolas de un color en una urna es inferior al de bolas extraídas.

Por tanto, existen varias bolas del mismo color dentro de cada urna y las mismas son indistinguibles. Así mismo, no importa el orden en la extracción al ser indistinguibles, vemos que son combinaciones. Por último, deseamos que el suceso ocurra en todas las urnas a la vez, por tanto es una intersección de los distintos sucesos y se obtiene multiplicando las distintas combinaciones obtenidas en cada urna.

$$CombRep_{X_1}(iBolas, m_1) * CombRep_{X_2}(iBolas, m_2) * .. * CombRep_{X_n}(iBolas, m_n)$$

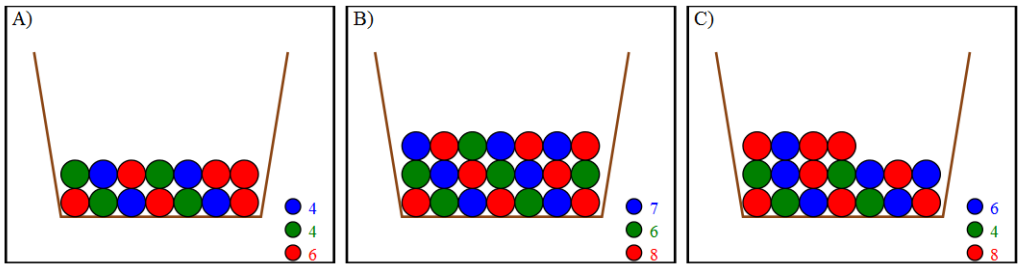

Figura 4.7.4: *Urnas con bolas de distinto color*

4.7.1.5. Test de conducir Ukumundu

El examen para el carnet de conducir en Ukumundu consta de 3 bloques de preguntas de distinta dificultad. bloque "A", bloque "B" y bloque "C". Para generar el examen, los examinadores de Ukumundu escogen ”n” cuestiones de entre las preguntas del bloque "A", ”m” cuestiones de entre las preguntas del bloque "B" y ”o” cuestiones de entre las preguntas del bloque "C". Realizado un examen de ”$n+m+o$” preguntas en total, ¿cuántos exámenes distintos pueden generar los examinadores de Ukumundu?.

Cada examen consta de una selección de ”n” números entre ”m” de cada bloque. Dentro de cada bloque da igual sí en la selección la ”$Pregunta_x$” se fórmula antes de la ”$Pregunta_y$” o al revés, por tanto el orden no es importante en la selección. Tampoco existe repetición, ya que en un examen sería absurdo incluir dos veces la misma pregunta. En conclusión, estamos hablando de combinaciones sin repetición.

Por otra parte el examen consta de las preguntas de 3 bloques y la solución es por tanto la intersección de los sucesos y el producto de los mismos.

$$Comb(A_n, A_s) * Comb(B_n, B_s) * Comb(C_n, C_s)$$

Siendo A_n: La cantidad de preguntas del bloque “A”.

Siendo A_s: El número de cuestiones seleccionadas del bloque “A”.

Siendo B_n: La cantidad de preguntas del bloque “B”.

Siendo B_s: El número de cuestiones seleccionadas del bloque “B”.

Siendo C_n: La cantidad de preguntas del bloque “C”.

Siendo C_s: El número de cuestiones seleccionadas del bloque “C”.

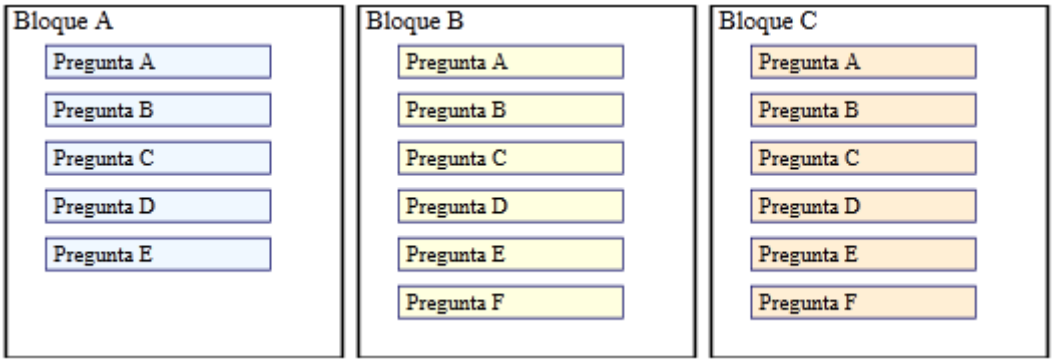

Figura 4.7.5: *Test de conducir Ukumundu*

4.7.1.6. Seleccionador de Fútbol

Un entrenador de fútbol está realizando la alineación del equipo que jugará el próximo partido. Sí solo dispone de "n" delanteros (Figura A) y de "m" defensas (Figura B), ¿de cuántas formas puede seleccionar los jugadores sí solo piensa jugar con "p" delanteros y "q" defensas?.

Se dispone de "n" delanteros de "p"posibles, está claro que el orden no importa; da igual seleccionar primero al delantero "A" que al "B" y tampoco puede existir repetición. Estamos hablando entonces, de combinaciones sin repetición. Por otra parte, también debemos seleccionar "m" defensas de "q" posibles y por tanto, la solución del problema será el producto de las 2 selecciones.

$Comb(p,n) * Comb(q,m)$

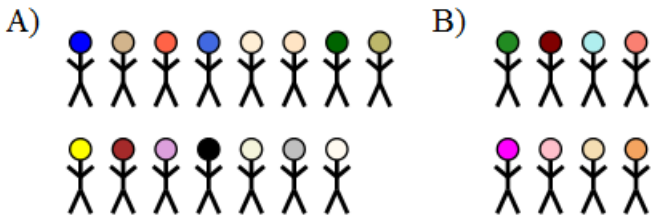

Figura 4.7.6: *Seleccionador de Fútbol*

4.7.2. Combinaciones con repetición

4.7.2.1. Urnas con bolas del mismo color

Tenemos “n” urnas: "Urna X_1", "Urna X_2",.. "Urna X_n", . Cada urna contiene bolas de un solo color .¿De cuántas formas posibles podemos seleccionar m Bolas?.

En este problema siempre se cumple que el número de bolas de cada urna es inferior o igual al número de bolas extraídas.

Las bolas de cada urna son indistinguibles entre sí, y es por ello, que son combinaciones al no importar el orden entre ellas. Existe repetición ya que como indica el enunciado se pueden elegir entre varias bolas del mismo tipo, con lo cual el resultado sería el siguiente:

$$CombRep(n,m) = Comb(\begin{matrix} n+m-1 \\ n \end{matrix})$$

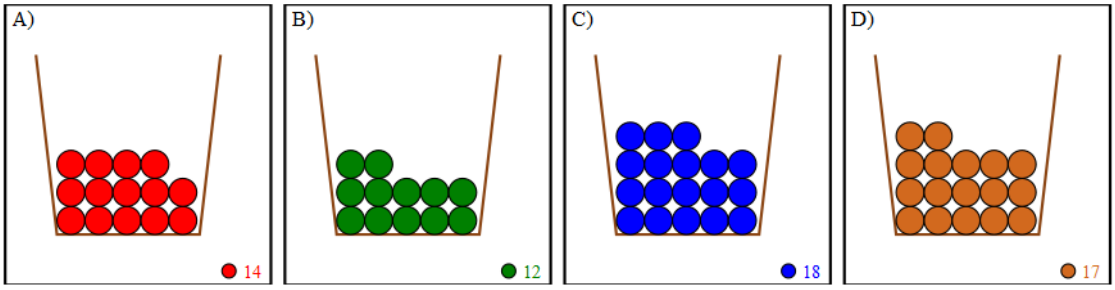

Figura 4.7.7: *Urnas con bolas del mismo color*

4.7.2.2. Heladería

Una heladería dispone de “n” sabores diferentes para sus bolas de helado y vende las tipologías de helados que van desde “p” hasta “q” bolas de helado. Pudiéndose repetir sabor en el mismo helado, ¿cuántos helados diferentes dispone para la venta la heladería?.

En este problema no importa orden, ya que dos helados se consideran iguales si por ejemplo la bola de helado de menta está situada a la izquierda o a la derecha. Existe repetición, ya que según el enunciado se permiten helados con 2 o más bolas del mismo sabor. Por tanto, estamos hablando de combinaciones con repetición.

Por otra parte existen varias tipologías que van desde “p” bolas hasta “q” bolas, con lo cual el resultado será la suma de todas las tipologías que son vendidas en la heladería.

$$CombRep(n,p)+CombRep(n,(p+1))..CombRep(n,(q-1))+CombRep(n,q)$$

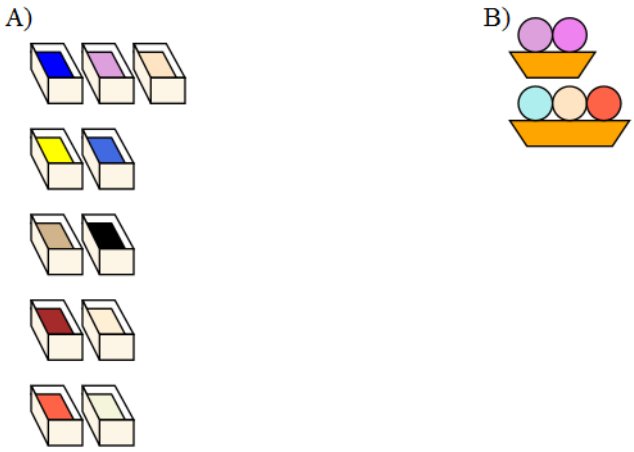

Figura 4.7.8: *Heladería*

4.7.2.3. Guarda canicas en cajas

Un niño dispone de “n” canicas iguales y desea guardarlas en “m” cajas. ¿De cuántas maneras distintas puede guardar las canicas en las cajas sí todas las canicas caben en una sola caja y no es necesario que todas las cajas contengan alguna canica? .

En este problema no importa orden, ya que todas las canicas son iguales y por tanto indistinguibles. Otro dato importante que nos indica el problema, es que una caja puede estar vacía o bien puede contener las “n” canicas.

Una posible secuencia de este problema sería “111122” que indica que hay 4 canicas en la caja 1 y 2 canicas en la caja 2, a la vista del ejemplo se extrae fácilmente la conclusión de que estamos hablando de combinaciones con repetición.

$CombRep(m,n)$

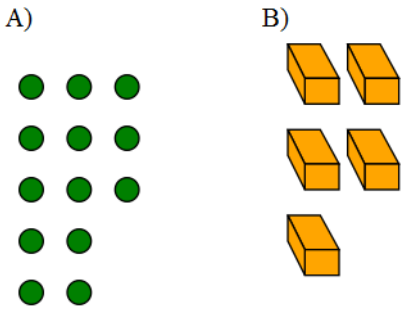

Figura 4.7.9: *Guarda canicas en cajas*

4.8. Permutaciones

4.8.1. Permutaciones sin repetición

4.8.1.1. Mesa de Navidad

En una boda los comensales se sientan alrededor de una mesa rectangular como en la disposición representada en la figura “A”. ¿De cuántas formas pueden sentarse “n” comensales, evitando que las personas “A” y “C” se sienten una en frente de la otra?. Atención, la figura "A" y la "B" se consideran disposiciones idénticas.

En este problema si importa el orden, ya que lo que estamos calculando es la disposición de las personas ante una mesa rectangular. Por otra parte, intervienen todos los elementos en la selección, pues todas las personas se sentarán ante la mesa.

- Procedemos inicialmente calculando las posiciones posibles (permutaciones):
  - $posPosibles = nComensales!$
- Posteriormente, calculamos las posiciones en las que se cumple que están en frente la/s parejas incompatibles:
  - $situPareja = Variacion(\frac{nComensales}{2}, nParejasProhibidas)$
- Calculamos las posiciones libres, que son aquellas donde la gente puede variar suponiendo fijadas las posiciones de las parejas incompatibles:
  - $posLibres = nComensales - (nParejasProhibidas * 2)$
- Eliminamos la simetría (Es lo mismo BDFACE que ECAFDB):
  - $posPosiblesNoSimetría = \frac{posPosibles}{2}$

- Por último restamos de todas las posiciones posibles, aquellas donde ocurre la/s parejas incompatible

  - $posPosiblesNoSimetría - (situPareja * posLibres!)$

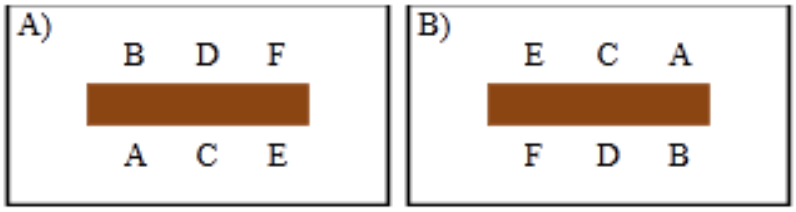

Figura 4.8.1: *Mesa de Navidad*

4.8.1.2. Palabras Diferentes (Urna con bolas unívocas)

Se introducen en una urna “n” bolas etiquetadas cada una con una letra (Ver figura A). Posteriormente, se extraen todas las bolas de la urna formando una palabra "ABCDEF”. ¿Cuántas palabras diferentes tengan sentido o no, pueden formarse en las sucesivas extracciones con todas las bolas de la urna?

En este ejercicio, se trabajan los fundamentos de las permutaciones sin repetición. En el mismo, se establece que el tamaño de la palabra coincide con el número de letras diferentes, y por tanto siempre se cumple lo siguiente:

$$letras_{diferentes} = tamaño_{palabra}$$

$$palabras_{diferentes} = letras_{diferentes}!$$

$$palabras_{diferentes} = n!$$

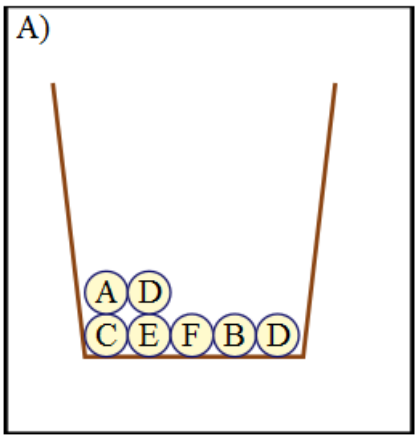

Figura 4.8.2: *Urna con bolas unívocas*

4.8.2. Permutaciones con repetición

4.8.2.1. Palabras Diferentes (Urna con bolas repetidas)

Se introducen en una urna “n” bolas etiquetadas cada una con una letra (Ver figura A). Posteriormente, se extraen todas las bolas de la urna formando una palabra, "XXYYYZX” . ¿Cuántas palabras diferentes tengan sentido o no, pueden formarse en las sucesivas extracciones con todas las bolas de la urna?

En este ejercicio, si importa el orden ya que la palabra formada por las letras “XY” es diferente a “YX”; así mismo, se seleccionan grupos formados por todas las bolas, con lo cual estamos hablando de permutaciones con repetición:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}^{Letras} = \begin{pmatrix} r1 \\ r2 \\ r_n \end{pmatrix}^{Repeticiones}$$

Por tanto $P_k{}^{r1,r2,r^n} = \frac{k!}{r_1!.r_2!.r_n}$ *Siendo* k igual al número de bolas de la urna.

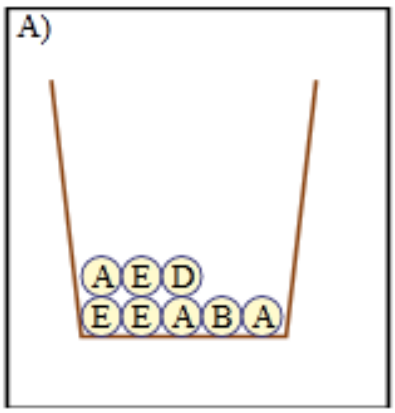

Figura 4.8.3: *Urna con bolas repetidas*

Capítulo 5

Despliegue y configuración

5.1. Carga Inicial

El aplicativo sobre el que versa este proyecto viene empaquetado en un archivo jar facilitando su despliegue en el sistema SIETTE.

El sistema SIETTE facilita enormemente la creación de nuevos ejercicios mediante una sencilla interfaz donde podemos indicar diversos parámetros de ejecución de nuestros ejercicios, así como los archivos auxiliares que serán usados por los mismos, en el proyecto que nos ocupa un archivo jar con las clases necesarias para la correcta ejecución de cada ejercicio.

Una vez creado un nuevo tema en siette, procedemos a la carga del archivo jar mediante la interfaz de siette “Gestión de archivos”, donde indicamos la ruta del archivo “jar” contenedor del proyecto.

La carga es sencilla y rápida ya que no debemos indicar ninguna otra opción para proceder a la misma.

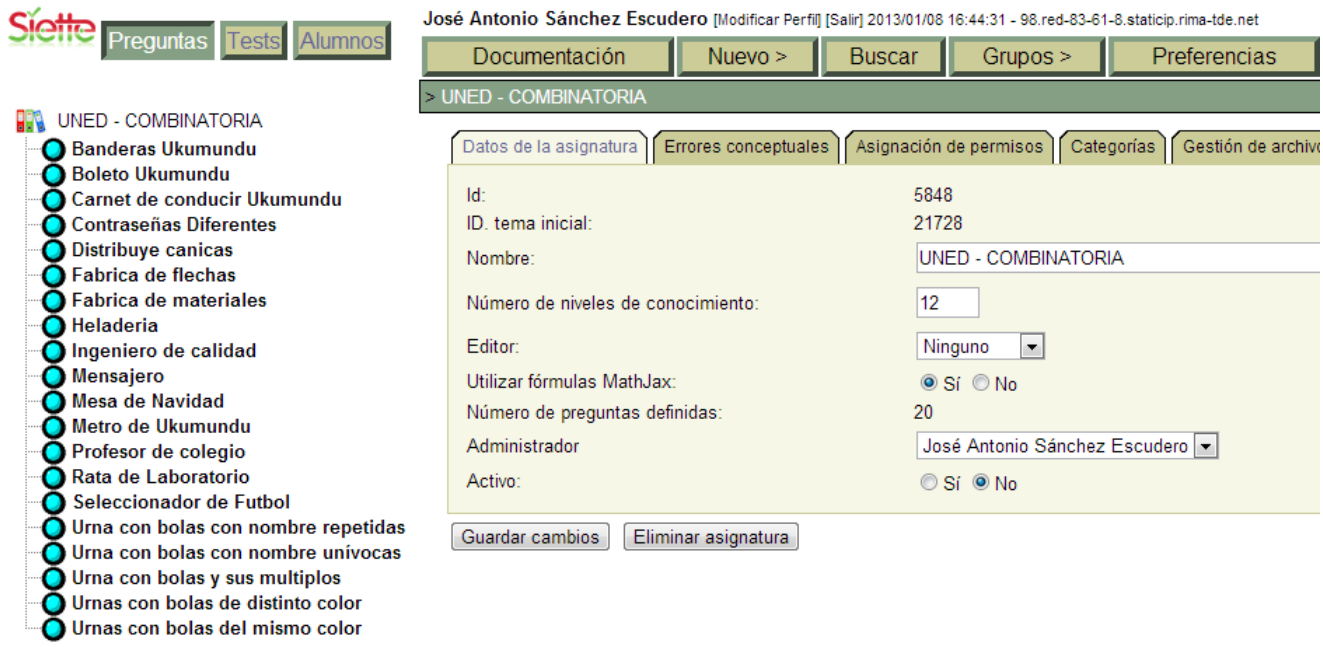

Figura 5.1.1: *Interfaz de carga en SIETTE*

Terminada la carga procedemos a la configuración de los parámetros de cada uno de los ejercicios de los que consta el aplicativo y que han sido explicados el tema anterior.

5.2. Ajuste de los ejercicios

Cada ejercicio puede ser parametrizado mediante el jsp que se utiliza en la carga del enunciado en siette.

Los parámetros que admite cada ejercicio serán desglosados en apartados posteriores pero de manera introductoria se puede decir que el ajuste de los mismos permite adecuar por el profesor el grado de dificultad de los ejercicios (al variar el tamaño de sus valores) así como otros aspectos de interfaz como el tamaño de los dibujos que acompaña a cada ejercicio.

5.2.1. Calcular el número de múltiplos posibles

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Bolas Múltiplo          *
 *  *************************************************/
//Número de bolas
param=new Parametro("Bolas Diferentes", 6, 15);
ejParam.GuardaParametro(15, param);
//Múltiplo A
param=new Parametro("Múltiplo A", 2, 3);
ejParam.GuardaParametro(15, param);
//Múltiplo B
param=new Parametro("Múltiplo B", 4, 6);
ejParam.GuardaParametro(15, param);
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 250 , 250);
ejParam.GuardaParametro(15, param);
```

Figura 5.2.1: *Parámetros del ejercicio "Ingeniero Electrónico"*

Parámetros admitidos:

- Parámetro "Bolas Diferentes": define el número mínimo y máximo entre el que estará el número de bolas de la urna
- Parámetro "Múltiplo de A": rango de valores que comprende el número del que debe ser múltiplo las sucesivas extracciones.
- Parámetro "Múltiplo de B": rango de valores que comprende el número del que debe ser múltiplo las sucesivas extracciones.
- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en función del ancho y el largo coordenadas(x,y).

5.2.2. Ingeniero Electrónico

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Ingeniero *
 *  *************************************************/
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 400 , 150);
ejParam.GuardaParametro(14, param);
//Número de Bornes Placa A
param=new Parametro("Número de Bornes Placa A", 6, 15);
ejParam.GuardaParametro(14, param);
//Número de Bornes Placa B
param=new Parametro("Número de Bornes Placa B", 6, 15);
ejParam.GuardaParametro(14, param);
//Tiempo de comprobación
param=new Parametro("Tiempo comprobación", 11, 25);
ejParam.GuardaParametro(14, param);
```

Figura 5.2.2: *Parámetros del ejercicio "Ingeniero Electrónico"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en función del ancho y el largo coordenadas(x,y).
- Parámetro "Número Bornes Placa A": rango de valores que comprende el número de bornes que contendrá la placa de circuito "A".
- Parámetro "Número Bornes Placa B": rango de valores que comprende el número de bornes que contendrá la placa de circuito "B".
- Parámetro "Tiempo de comprobación": establece el tiempo que se tarda en comprobar la unión de un borne de cada placa.

5.2.3. Maestro de escuela

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Recreo Colegio          *
 *  *************************************************/
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 400 , 150);
ejParam.GuardaParametro(18, param);
//Número de escolares
param=new Parametro("Número de escolares", 6, 12);
ejParam.GuardaParametro(18, param);
```

Figura 5.2.3: *Parámetros ejercicio "Maestro de escuela"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en coordenadas (x,y) ancho y alto.
- Parámetro "Número de escolares": establece entre que dos valores estará comprendido el número de escolares que contiene la fila india.

5.2.4. Fábrica de materiales

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Fábrica de Materiales  *
 *  ************************************************/
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 200 , 250);
ejParam.GuardaParametro(19, param);
//Número de Compuestos
param=new Parametro("Número de Compuestos",5, 15);
ejParam.GuardaParametro(19, param);
//Número de cajas
param=new Parametro("Número de Ingredientes Mezcla", 2, 8);
ejParam.GuardaParametro(19, param);
//Tiempo de fabricación
param=new Parametro("Tiempo de fabricación", 300, 600);
ejParam.GuardaParametro(19, param);
```

Figura 5.2.4: *Parámetros ejercicio "Maestro de escuela"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en coordenadas (x,y) ancho y alto.
- Parámetro "Número de compuestos": establece entre que dos valores estará comprendido el número de ingredientes para generar los nuevos materiales.
- Parámetro "Número de ingredientes mezcla": establece cuantos ingredientes va a contener la mezcla.
- Parámetro "Tiempo de fabricación": establece entre que valores estará comprendido el tiempo en generar cada nuevo material.

5.2.5. Banderas Ukumundu

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Bandera                 *
 *  *************************************************/
param=new Parametro("Tamaño Rejilla", 600 , 200);
ejParam.GuardaParametro(9, param);
//Número de franjas horizontales
param=new Parametro("Cantidad de Franjas Horizontales",3,6 );
ejParam.GuardaParametro(9, param);
//Número de colores
param=new Parametro("Cantidad de Colores", 5, 10);
ejParam.GuardaParametro(9, param);
```

Figura 5.2.5: *Parámetros ejercicio "Bandera Ukumundu"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en coordenadas (x,y) ancho y alto.
- Parámetro "Cantidad de Franjas Horizontales": establece entre que dos valores estará comprendido el número de franjas horizontales que componen la bandera.
- Parámetro "Cantidad de Colores": permite definir el rango entre el que se sitúa el número de colores posibles para cada una de las franjas, por supuesto el número de colores no puede ser inferior al del número de franjas horizontales.

5.2.6. Metro de Ukumundu

Código Jsp del enunciado:

```
/*******************************************************
 *  Parámetros del ejercicio Plano del Metro Ukumundu *
 *  ******************************************************/
//Cantidad de Estaciones
param=new Parametro("Cantidad de Estaciones",8 ,15 );
ejParam.GuardaParametro(11, param);
//Cantidad de Lineas
param=new Parametro("Tamanno Plano", 300, 300);
ejParam.GuardaParametro(11, param);
```

Figura 5.2.6: *Parámetros ejercicio "Bandera Ukumundu"*

Parámetros admitidos:

- Parámetro "Tamaño Plano": define el tamaño del plano del Metro en coordenadas (x,y) ancho y alto.
- Parámetro "Cantidad de Estaciones": establece entre que dos valores estará comprendido el número de estaciones que componen la red de Metro. El número de líneas es generado automáticamente de manera aleatoria cuando se genera el ejercicio y por tanto no admite un rango de valores.

5.2.7. Flechas Olimpiada de Ukumundu

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Fábrica de flechas *
 *  ************************************************/
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 400 , 100);
ejParam.GuardaParametro(13, param);
//Número de colores
param=new Parametro("Número de colores", 4, 10);
ejParam.GuardaParametro(13, param);
//Tiempo de fabricación flecha individual
param=new Parametro("Tiempo fabricación", 10, 20);
ejParam.GuardaParametro(13, param);
//Número de flechas por tipología
param=new Parametro("Número Flechas Tipología", 50, 100);
ejParam.GuardaParametro(13, param);
```

Figura 5.2.7: *Parámetros del ejercicio "Ingeniero Electrónico"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en función del ancho y el largo coordenadas(x,y).
- Parámetro "Número de colores": rango de valores que comprende el número de colores permitido en la fabricación de las flechas.
- Parámetro "Tiempo fabricación": rango de valores que comprende el tiempo necesario para una sola flecha.
- Parámetro "Número de Flechas Tipología": establece el número de flechas necesarias para completar un pedido.

5.2.8. Contraseñas Alpha Asociados

Código Jsp del enunciado:

```
/*********************************************************
 *  Parámetros del ejercicio Contraseña Alpha Asociados *
 *  ******************************************************/
//Tamaño Contraseña Extremo Inferior
param=new Parametro("Tamaño Contraseña", 4, 8);
ejParam.GuardaParametro(0, param);
//Tamaño Contraseña Extremo Superior
param=new Parametro("Tamaño Contraseña", 9, 12);
ejParam.GuardaParametro(0, param);
//Número de letras diferentes Contraseña
param=new Parametro("Letras diferentes Contraseña", 2, 5);
ejParam.GuardaParametro(0, param);
//Tamaño del dibujo
param=new Parametro("Tamaño Dibujo",360 , 200);
ejParam.GuardaParametro(0, param);
```

Figura 5.2.8: *Parámetros ejercicio "Contraseñas Alpha Asociados"*

Parámetros admitidos:

- Parámetro "Tamaño contraseña extremo inferior": establece entre que valores estará situado el mínimo de caracteres de la contraseña.
- Parámetro "Tamaño contraseña extremo superior": establece entre que valores estará situado el máximo de caracteres de la contraseña. Por supuesto entre ambos rangos no debe existir la intersección de los mismos.
- Parámetro "Letras diferentes contraseña": define entre que valores estará comprendido el número de letras diferentes que se permitirán para la creación de la contraseña.
- Parámetro "Tamaño Dibujo": define el tamaño del dibujo en coordenadas (x,y) ancho y alto.

5.2.9. Ratón de Laboratorio

Código Jsp del enunciado:

```
/*************************************************
 *  Parámetros del ejercicio Ratón de Laboratorio    *
 *  ************************************************/
param=new Parametro("Tamaño Laboratorio eje X", 200, 400);
ejParam.GuardaParametro(5, param);
param=new Parametro("Tamaño Laboratorio eje Y", 200, 400);
ejParam.GuardaParametro(5, param);
//Tamaño de lado de cada cuadrado que conforma la rejilla
param=new Parametro("Tamaño Estancia", 40, 40);
ejParam.GuardaParametro(5, param);
//Tamaño de la calle que existe entre los cuadrados
//(Numero par y menor de Tamaño Cuadrado)
param=new Parametro("Altura Pared", 10, 10);
ejParam.GuardaParametro(5, param);
```

Figura 5.2.9: *Parámetros ejercicio "Ratón de Laboratorio"*

Parámetros admitidos:

- Parámetro "Tamaño Laboratorio eje X": establece el ancho de la pared del dibujo del laboratorio.
- Parámetro "Tamaño Laboratorio eje Y": establece el alto de la pared del dibujo del laboratorio

Los dos parámetros anteriores definen y gestionan la dificultad del ejercicio, cuanto mayor sea el tamaño de las paredes mayor será el laberinto y aumentará su complejidad.

- Parámetro "Tamaño estancia": establece el ancho y alto de cada pared que conforma una habitación.
- Parámetro "Altura pared": define el valor de la altura que tendrá la pared de cada estancia, dado que la misma está dibujada en perspectiva .

5.2.10. Mensajero

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Mensajero               *
 *  *************************************************/

//Tamaño Rejilla Coordenadas Finales de la rejilla (x,y)
param=new Parametro("Tamaño Rejilla CoorX", 100, 400);
ejParam.GuardaParametro(3, param);
param=new Parametro("Tamaño Rejilla CoorY", 100, 400);
ejParam.GuardaParametro(3, param);
//Tamaño de lado de cada cuadrado que conforma la rejilla
param=new Parametro("Tamaño Cuadrado", 16, 0);
ejParam.GuardaParametro(3, param);
//Tamaño de la calle que existe entre los cuadrados
//(Numero par y menor de Tamaño Cuadrado)
param=new Parametro("Tamaño Calle", 12, 0);
ejParam.GuardaParametro(3, param);
```

Figura 5.2.10: *Parámetros del ejercicio "Mensajero"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla Coor X": establece el ancho de la pared del dibujo de la ciudad.
- Parámetro "Tamaño Rejilla Coor Y": establece el alto de la pared del dibujo de la ciudad.

Los dos parámetros anteriores definen el ancho y alto de la ciudad y por tanto la dificultad del ejercicio.

- Parámetro "Tamaño cuadrado": establece el ancho y alto de la pared de un edificio.
- Parámetro "Tamaño calle": define la distancia que existe entre dos edificios.

5.2.11. Lotería Ukumundu

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Lotería Ukumundu       *
 *  *************************************************/
//Número de bolas bombo A
param=new Parametro("Cantidad de Bolas A",20,40 );
ejParam.GuardaParametro(7, param);
//Número de bolas bombo B
param=new Parametro("Cantidad de Bolas B", 5, 10);
ejParam.GuardaParametro(7, param);
//Número de bolas bombo C
param=new Parametro("Cantidad de Bolas C", 5, 10);
ejParam.GuardaParametro(7, param);
//Número de bolas extraídas bombo A
param=new Parametro("Cantidad de Bolas Extraidas A", 2, 5);
ejParam.GuardaParametro(7, param);
//Número de bolas bombo B
param=new Parametro("Cantidad de Bolas Extraidas B", 2, 5);
ejParam.GuardaParametro(7, param);
//Número de bolas bombo B
param=new Parametro("Cantidad de Bolas Extraidas C", 2, 5);
ejParam.GuardaParametro(7, param);
```

Figura 5.2.11: *Parámetros del ejercicio "Lotería Ukumundu"*

Parámetros admitidos:

- Parámetro "Cantidad de bolas A": rango de valores donde esta comprendido el número de bolas del bombo "A".
- Parámetro "Cantidad de bolas B": rango de valores donde esta comprendido el número de bolas del bombo "B".
- Parámetro "Cantidad de bolas C": rango de valores donde esta comprendido el número de bolas del bombo "C".
- Parámetro "Cantidad de bolas extraídas bombo A ": rango de valores donde esta comprendido el número de bolas extraídas bombo "A".
- Parámetro "Cantidad de bolas extraídas bombo B ": rango de valores donde esta comprendido el número de bolas extraídas bombo "B".

- Parámetro “Cantidad de bolas extraídas bombo C”: rango de valores donde esta comprendido el número de bolas extraídas bombo “C”.

5.2.12. Urnas con bolas de distinto color

Código Jsp del enunciado:

```
/*******************************************************
 *  Parámetros del ejercicio Urna Bolas Diferente Color *
 *  *****************************************************/
//Número de urnas
param=new Parametro("Número Urnas", 2, 4);
ejParam.GuardaParametro(6, param);
//Número de bolas posibles en cada urna
param=new Parametro("Número Bolas Rojas", 4, 8);
ejParam.GuardaParametro(6, param);
param=new Parametro("Número Bolas Verdes", 4, 8);
ejParam.GuardaParametro(6, param);
param=new Parametro("Número Bolas Azules", 4, 8);
ejParam.GuardaParametro(6, param);
param=new Parametro("Tamaño Rejilla", 800 , 200);
ejParam.GuardaParametro(6, param);
```

Figura 5.2.12: *Parámetros del ejercicio "Urna con bolas de diferente color"*

Parámetros admitidos:

- Parámetro "Número Urnas": rango de valores donde esta comprendido el número urnas que se mostrarán en el ejercicio.
- Parámetro "Número Bolas Rojas": rango de valores que comprende el número de bolas rojas por cada urna.
- Parámetro "Número Bolas Verdes": rango de valores que comprende el número de bolas rojas por cada urna.
- Parámetro "Número Bolas Azules": rango de valores que comprende el número de bolas rojas por cada urna.
- Parámetro "Tamaño Rejilla": define en función de las coordenadas (x,y) el ancho y alto del dibujo que contendrá todas las urnas del ejercicio.

5.2.13. Test de conducir Ukumundu

Código Jsp del enunciado:

```
/*******************************************************
*  Parámetros del ejercicio Test de Conducir Ukumundu *
*  *******************************************************/
//Cantidad de Ejercicios Bloque A
param=new Parametro("Cantidad de Ejercicios Bloque A", 5,8 );
ejParam.GuardaParametro(10, param);
//Cantidad de Ejercicios Bloque B
param=new Parametro("Cantidad de Ejercicios Bloque B", 5, 8);
ejParam.GuardaParametro(10, param);
//Cantidad de Ejercicios Bloque C
param=new Parametro("Cantidad de Ejercicios Bloque C", 5, 8);
ejParam.GuardaParametro(10, param);
//Número de Ejercicios Bloque A
param=new Parametro("Nº Ejercicios Bloque A", 2, 5);
ejParam.GuardaParametro(10, param);
//Número de Ejercicios Bloque B
param=new Parametro("Nº Ejercicios Bloque B", 2, 5);
ejParam.GuardaParametro(10, param);
//Número de Ejercicios Bloque B
param=new Parametro("Nº Ejercicios Bloque C", 2, 5);
ejParam.GuardaParametro(10, param);
```

Figura 5.2.13: *Parámetros del ejercicio "Test de conduccir Ukumundu"*

Parámetros admitidos:

- Parámetro "Cantidad de ejercicios bloque A": rango de valores donde esta comprendido el número de ejercicios del bloque "A".
- Parámetro "Cantidad de ejercicios bloque B": rango de valores donde esta comprendido el número de ejercicios del bloque "B".
- Parámetro "Cantidad de ejercicios bloque C": rango de valores donde esta comprendido el número de ejercicios del bloque "C".
- Parámetro "Nº Ejercicios bloque A": establece entre que valores estará situado el número de ejercicios seleccionados del bloque "A".
- Parámetro "Nº Ejercicios bloque B": establece entre que valores estará situado el número de ejercicios seleccionados del bloque "B".

- Parámetro “Nº Ejercicios bloque C”: establece entre que valores estará situado el número de ejercicios seleccionados del bloque “C”.

5.2.14. Seleccionador de Fútbol

Código Jsp del enunciado:

```
 /*****************************************************
  *  Parámetros del ejercicio SeleccionadorFutbol     *
  *  *************************************************/
//Tamaño Rejilla
 param=new Parametro("Tamaño Rejilla", 400 , 150);
 ejParam.GuardaParametro(16, param);
 //Número de delanteros fig A
 param=new Parametro("Número de delanteros fig A", 6, 15);
 ejParam.GuardaParametro(16, param);
//Número de defensas fig B
 param=new Parametro("Número de defensas fig B", 6, 15);
 ejParam.GuardaParametro(16, param);
//Número de delanteros alineación
 param=new Parametro("Número de delanteros alineación", 2, 5);
 ejParam.GuardaParametro(16, param);
//Número de defensas alineación
 param=new Parametro("Número de defensas alineación", 2, 5);
 ejParam.GuardaParametro(16, param);
//
```

Figura 5.2.14: *Parámetros del ejercicio "Seleccionador de Fútbol"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define en función de las coordenadas (x,y) el ancho y alto del dibujo que contendrá las fotos de los jugadores.
- Parámetro "Número de delanteros fig. A": rango de valores donde está comprendido el número de delanteros posibles en la alineación.
- Parámetro "Número de defensas fig. B": rango de valores donde está comprendido el número de defensas posibles en la alineación.
- Parámetro "Número de delanteros alineación": rango de valores donde está comprendido el número de delanteros que serán convocados para jugar como titulares en el equipo.
- Parámetro "Número de defensas alineación": rango de valores donde está comprendido el número de defensas que serán convocados para jugar como titulares en el equipo.

5.2.15. Urnas con bolas del mismo color

Código Jsp del enunciado:

```
/*************************************************************
 *  Parámetros del ejercicio Urna con bolas del mismo color *
 *  *************************************************************/
//Número de urnas
param=new Parametro("Número de Urnas", 3, 4);
ejParam.GuardaParametro(8, param);
//Número de bolas posibles en cada urna
param=new Parametro("Número Bolas Rojas", 10, 20);
ejParam.GuardaParametro(8, param);
param=new Parametro("Número Bolas Verdes", 10, 20);
ejParam.GuardaParametro(8, param);
param=new Parametro("Número Bolas Azules", 10,20);
ejParam.GuardaParametro(8, param);
param=new Parametro("Número Bolas Amarillas", 10, 20);
ejParam.GuardaParametro(8, param);
param=new Parametro("Tamaño Rejilla", 800 , 200);
ejParam.GuardaParametro(8, param);
```

Figura 5.2.15: *Parámetros del ejercicio "Urna con bolas del mismo color"*

Parámetros admitidos:

- Parámetro "Número Urnas": rango de valores donde esta comprendido el número urnas que se mostrarán en el ejercicio.

- Parámetro "Número Bolas Rojas": rango de valores que comprende el número de bolas rojas que únicamente contendrá la primera urna.

- Parámetro "Número Bolas Verdes": rango de valores que comprende el número de bolas verdes que únicamente contendrá la segunda urna.

- Parámetro "Número Bolas Azules": rango de valores que comprende el número de bolas azules que únicamente contendrá la tercera urna.

- Parámetro "Número Bolas Amarillas": rango de valores que comprende el número de bolas amarillas que únicamente contendrá la cuarta urna.

- Parámetro “Tamaño Rejilla”: define en función de las coordenadas (x,y) el ancho y alto del dibujo que contendrá todas las urnas del ejercicio.

5.2.16. Heladería

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Heladería *
 *  ***********************************************/
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 560 , 300);
ejParam.GuardaParametro(12, param);
//Número de sabores
param=new Parametro("Número de sabores", 2, 15);
ejParam.GuardaParametro(12, param);
//Número de bolas extremo inferior
param=new Parametro("Número Bolas Inferior", 1, 2);
ejParam.GuardaParametro(12, param);
//Número de bolas extremo superior
param=new Parametro("Número Bolas Superior", 3, 5);
ejParam.GuardaParametro(12, param);
```

Figura 5.2.16: *Parámetros del ejercicio "Heladería"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define en función de las coordenadas (x,y) el ancho y alto del dibujo.
- Parámetro "Número de sabores": rango de valores que comprende el número de sabores permitidos en los helados.
- Parámetro "Número Bolas Inferior": rango de valores que comprende el número de bolas que como mínimo poseerá el más pequeño de los helados.
- Parámetro "Número Bolas Superior": rango de valores que comprende el número de bolas que como máximo poseerá el más grande de los helados.

5.2.17. Canicas en cajas

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Canicas en cajas *
 *  ***********************************************/
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 300 , 200);
ejParam.GuardaParametro(17, param);
//Número de canicas
param=new Parametro("Número de Canicas",10, 25);
ejParam.GuardaParametro(17, param);
//Número de cajas
param=new Parametro("Número de Cajas", 2, 6);
ejParam.GuardaParametro(17, param);
```

Figura 5.2.17: *Parámetros del ejercicio "Canicas en cajas"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define en función de las coordenadas (x,y) el ancho y alto del dibujo.
- Parámetro "Número de sabores": rango de valores que comprende el número de canicas del problema.
- Parámetro "Número de cajas": rango de valores que comprende el número de cajas que habrá disponible para guardar las canicas.

5.2.18. Mesa de Navidad

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Mesa Navidad          *
 *  *************************************************/
//Tamaño Rejilla Coordenadas Finales de la rejilla (x,y)
param=new Parametro("Tamaño Rejilla", 400 , 100);
ejParam.GuardaParametro(4, param);
//Número Comensales
param=new Parametro("Número Comensales", 4, 8);
ejParam.GuardaParametro(4, param);
//Número de parejas incompatibles
param=new Parametro("Parejas Incompatibles", 1, 2);
ejParam.GuardaParametro(4, param);
```

Figura 5.2.18: *Parámetros del ejercicio "Urna con bolas de diferente color"*

Parámetros admitidos:

- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en función del ancho y el largo coordenadas(x,y).
- Parámetro "Número Comensales": rango de valores que comprende el número de comensales que se sentarán en la mesa.
- Parámetro "Parejas Incompatibles": almacena el número de parejas incompatibles que se sentarán en la mesa. Cuando una pareja es incompatible los comensales que forman dicha pareja no se pueden sentar uno enfrente a otro.
- Parámetro "Número Bolas Azules": rango de valores que comprende el número de bolas rojas por cada urna.
- Parámetro "Tamaño Rejilla": define en función de las coordenadas (x,y) el ancho y alto del dibujo que contendrá todas las urnas del ejercicio.

5.2.19. Palabras Diferentes (Urna con bolas unívocas)

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Palabras Diferentes *
 *  ************************************************/
//Tamaño Palabra
param=new Parametro("Tamaño Palabra", 6, 12);
ejParam.GuardaParametro(1, param);
//Rango Caracteres A-Z
param=new Parametro("Rango Caracteres", 65, 70);
ejParam.GuardaParametro(1, param);
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 200 , 200);
ejParam.GuardaParametro(1, param);
```

Figura 5.2.19: *Parámetros del ejercicio "Palabras diferentes (Urna con bolas unívocas)"*

Parámetros admitidos:

- Parámetro "Tamaño palabra": rango de valores que comprende el número de letras que contendrá la palabra generada.
- Parámetro "Rango Caracteres": rango de valores que comprende el número de caracteres diferentes que puede tener una letra.
- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en función del ancho y el largo coordenadas(x,y).

5.2.20. Palabras Diferentes (Urna con bolas repetidas)

Código Jsp del enunciado:

```
/*****************************************************
 *  Parámetros del ejercicio Urna Letras Repetidas  *
 *  *****************************************************/
//Tamaño Palabra
param=new Parametro("Tamaño Palabra", 6, 12);
ejParam.GuardaParametro(2, param);
//Tamaño Rejilla
param=new Parametro("Tamaño Rejilla", 200 , 200);
ejParam.GuardaParametro(2, param);
```

Figura 5.2.20: *Parámetros del ejercicio "Palabras diferentes (Urna con bolas unívocas)"*

Parámetros admitidos:

- Parámetro "Tamaño palabra": rango de valores que comprende el número de letras que contendrá la palabra generada.
- Parámetro "Tamaño Rejilla": define el tamaño del dibujo en función del ancho y el largo coordenadas(x,y).

Capítulo 6

Pruebas

Las pruebas de software se integran dentro del ciclo de vida del software y nos permite determinar el nivel de calidad del producto final, además de comprobar el grado de cumplimiento con respecto a las especificaciones de requisitos iniciales del sistema.

Para la realización de las pruebas se ha utilizado un equipo de sobremesa con las siguientes características:

- Procesador AMD 1090T.
- 8 Gigabytes de RAM .
- Sistema operativo Windows 7, 64 bits.
- Disco duro SATA2 7200 rpm.

El software utilizado para dichas pruebas es JUnit. JUnit es un conjunto de clases 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.

En este proyecto se realizaron dos tipos de pruebas, pruebas unitarias y pruebas de integración.

6.1. Pruebas unitarias

Son aquellas que se realizan dentro del ámbito de un módulo o clase y se encargan de comprobar que el código funciona correctamente, dentro de unos márgenes previamente establecidos. Con el fin de verificar que el software hace aquello para lo que estaba diseñado correctamente, se han realizado cuatro tipos diferentes de comprobaciones:

- Comprobación del extremo inferior, ¿funciona correctamente la operación en el extremo inferior? por ejemplo 0!.
- Comprobación de un valor intermedio, ¿funciona correctamente la operación realizada con un valor intermedio? por ejemplo 5!.
- Comprobación del desbordamiento, ¿que ocurre cuando realizamos la operación con un número que genere un desbordamiento?, por ejemplo 30!.
- Valores fuera de rango, ¿que ocurre cuando se invoca a la función, con un valor fuera del rango de entrada permitido?, por ejemplo -1!.

Como se puede observar en la siguiente imagen todas las pruebas realizadas fueron exitosas y los tiempos de ejecución de cada una de las operaciones no sobrepasaron la milésima de segundo.

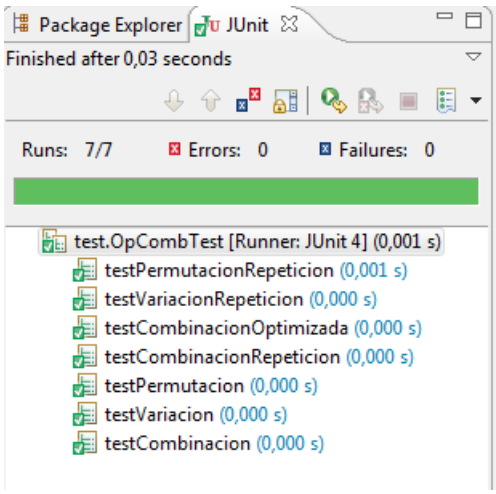

Figura 6.1.1: *Test sobre la clase "opComb"*

6.2. Pruebas de Integración

Dentro de este apartado se recogen aquellas pruebas que son realizadas al software completo o partes del mismo, en nuestro caso para realizarlas se ha optado por subdividir en software en veinte partes, que son los ejercicios que lo componen.

En el tema anterior vimos que los ejercicios que componen este proyecto admiten parámetros de trabajo tales como el número de bolas dentro de una urna, el número de escolares dentro de una fila, etc. por tanto para comprobar que el software es robusto, las pruebas realizadas han consistido en definir valores máximos fuera del rango permitido por dichos ejercicios.

Todas las pruebas realizadas fueron satisfactorias, obteniendose el valor "-1" en el enunciado de las respuestas. Recordemos que en el capítulo 3 se habló de la gestión del desbordamiento, cuando este ocurría, el objeto de la clase "opComb" utilizado en el cálculo, cambiaba su valor a -1 y modificaba su propiedad "overflow" a cierto.

Por tanto mediante el diseño usado en este proyecto aunque los parámetros de entrada estén fuera del rango permitido, el sistema sigue siendo estable no provocando un fallo en el servidor SIETTE.

Capítulo 7

Conclusiones

7.1. Conclusiones

Actualmente vivimos en una sociedad en continuo cambio en la que la tecnología debe dar respuesta a las nuevas necesidades que la sociedad plantea. Dentro del ámbito educativo, la incursión de los recursos tecnológicos ha supuesto una auténtica revolución metodológica. En este sentido, las nuevas tecnologías permiten al profesorado disponer de recursos más motivadores que amplían y refuerzan su labor como docente, permitiendo un mayor seguimiento del proceso de aprendizaje de sus alumnos. A su vez, los alumnos disponen de una herramienta para construir su aprendizaje de forma interactiva y más atractiva y cercana a sus intereses.

El sistema SIETTE supone una herramienta que permite a los ingenieros desarrollar componentes orientados al ámbito matemático. En concreto, en este proyecto se ha enfocado al área de la combinatoria. Nuestro estudio pretende dar respuesta a las carencias de los recursos existentes en el mercado, no siendo una mera propuesta de cálculo matemático, sino una aplicación práctica del ámbito de la combinatoria. En este sentido, ofrece una aplicación práctica de la combinatoria al mundo real.

Una de las principales bondades que aporta nuestro proyecto es la inclusión de figuras en el enunciado, siendo fundamentales para la correcta resolución del ejercicio. En este sentido, nuestro proyecto no se limita a una mera recopilación de imágenes, sino que las figuras que

acompañan a los ejercicios son generadas en tiempo real.

Adaptándonos a los tiempos que corren y analizando las posibles líneas futuras con respecto a los dispositivos tecnológicos, el uso de la especificación HTML5 y, en concreto, de SVG, ha revolucionado el mundo de los navegadores, ya que está rediseñando el concepto de la inclusión de imágenes en páginas web. En concreto, en nuestro proyecto, ha supuesto que la realización de los ejercicios no se reduzca al uso del ordenador, sino que facilite su ejecución en otros dispositivos móviles como smarthphone, tabletas, etc. El uso de SVG nos aporta otra ventaja, sus gráficos vectoriales, los cuales, a diferencia de los mapas de bits, poseen una gran calidad gráfica.

Nuestro proyecto ofrece facilidades al profesor como el ajuste de los parámetros de entrada de los ejercicios. Uno de los problemas del alumnado a la hora de abordar un ejercicio matemático de combinatoria, consiste en las dificultades que encuentra al enfrentarse al ejercicio con valores elevados. En este sentido, nuestro componente, permite graduar el rango de valores de entrada ajustándolo al nivel del alumnado.

Otra de las virtudes de este componente y de su uso dentro del entorno SIETTE, es la variación de los valores del enunciado. El alumno, no debe fijarse en los valores mostrados en el enunciado sino en el algoritmo o forma de resolución, afianzándose los aprendizajes.

El hecho de que SIETTE permita analizar la respuesta del alumno, posibilita que podamos analizar el error cometido al optar por una respuesta errónea, dándole una explicación del porqué de dicho razonamiento errado. En este sentido, no sólo se permite una interacción del alumno con el programa, sino que se posibilita un aprendizaje significativo y la generalización del conocimiento a otros contextos.

Nuestro trabajo ofrece múltiples posibilidades en el ámbito de la enseñanza a distancia, ya que permite una orientación en las dificultades encontradas y una autocorreción a tiempo real. Esto supone una evolución en la metodología de la enseñanza, al utilizarse nuevas herramientas que favorezcan un aprendizaje autodidacta.

Por último, me gustaría reseñar que todos los objetivos se han cumplido de manera satisfactoria. Se ha desarrollado un sistema generador de preguntas con un apartado gráfico que permite

su ampliación fácilmente.

Durante la realización de este proyecto, han surgido dos grandes dificultades. Por un lado, la generación del "esqueleto" del que heredan todos los ejercicios, que supone el paquete base a partir del cual se han desarrollado todas las cuestiones que componen el sistema. Nuestro objetivo principal, era diseñar un conjunto de clases que abstrayeran la entidad ejercicio, y contuvieran todas las propiedades y métodos que son necesarios para implementar un ejercicio del sistema. Se realizaron varios acercamientos hasta llegar al modelo final expuesto en este proyecto. Por otro lado, partíamos de un gran desafío, añadir una parte gráfica a los ejercicios que fuera compatible con SIETTE. Se estudiaron tres opciones diferentes y, después de realizar diferentes pruebas, se optó por usar SVG (ver estudio de viabilidad). Esta decisión supuso realizar un paquete completo con cada una de las interfaces que fueron usadas en este proyecto, facilitando la modularidad a la hora de realizar una figura y, por tanto, abstrayendo la realización de un dibujo.

Todas estas dificultades iniciales han resultado beneficiosas y han enriquecido el proyecto, al facilitar la ampliación del mismo, permitiendo añadir nuevos ejercicios sin modificar sus clase base y, al añadir un gestor de la parte gráfica que pueda ser manejada fácilmente por otros desarrolladores sin tener conocimientos previos de SVG.

El desarrollo de este proyecto ha resultado provechoso para mi formación como futuro ingeniero, ya que supone un acercamiento al mundo real de todo lo aprendido durante el mismo. Este proyecto no se limita al ámbito teórico, sino que pretende ser una herramienta práctica para el aprendizaje de la combinatoria aplicable a diferentes niveles educativos.

En mi opinión, SIETTE es un entorno atractivo y asequible en su uso para los desarrolladores. Su fuerte orientación al ámbito educativo simplifica el desarrollo de herramientas docentes, ya que permite multitud de opciones configurables que posibilitan un aprendizaje guiado. Considero que SIETTE puede mejorar y facilitar el aprendizaje a los futuros ingenieros. Por último, reseñar la importancia de la combinatoria en el ámbito de la ingeniería informática, dada su utilidad práctica como herramienta puente en otras materias, siendo este proyecto un recurso más que facilite su aprendizaje.

7.2. Líneas futuras

Nuestro proyecto supone un avance en los sistemas de generación automática de pregunta-respuesta en el ámbito de la combinatoria, sin embargo, admite mejoras y avances que faciliten tanto la interacción del profesor con el sistema como la amigabilidad con el alumno. En este sentido, expondremos a continuación diversas líneas de trabajo.

- Como se ha descrito anteriormente, los ejercicios admiten el ajuste de sus parámetros de entrada, el cual se realiza de manera manual al modificar los "JSP" incluidos en SIETTE. Una nueva funcionalidad consistiría en desarrollar una herramienta gráfica que modifique los "JSP" en segundo plano, facilitando de este modo la interacción del profesor con el sistema.

- Otro ámbito de trabajo a desarrollar, supondría añadir efectos de animación a todos los ejercicios. En la actualidad, sólo uno de ellos posee esta cualidad. Sería interesante que todos los ejercicios contuvieran animaciones. aumentando de este modo su atractivo visual.

- En este proyecto hemos desarrollado veinte preguntas, abarcando así todo el temario de combinatoria. Sería conveniente aumentar el número de preguntas, facilitando el refuerzo del aprendizaje.

- El sistema SIETTE es una herramienta de trabajo muy plural, no adscribiéndose a un sólo ámbito de conocimiento. En el desarrollo de este proyecto hemos podido observar que SIETTE podría ser adaptable a otras materias, no necesariamente relacionadas con el área de las matemáticas.

- No es posible prever la evolución de los dispositivos tecnológicos a diez años vista. Se habla de pantallas enrollables y de imágenes holográficas. Sí se conocen las tendencias para este año, hablándose de un nuevo concepto de televisión llamado "Smartv", el cual, además de permitir una interacción más cómoda mediante símbolos gestuales, posibilita la instalación de aplicaciones compatibles del fabricante, así como la navegación por

Internet. Una gran mayoría de estos dispositivos son compatibles con HTML5, y permiten los gráficos con SVG. Por tanto, van a permitir que los alumnos puedan realizar los test directamente a través del televisor, con una mayor comodidad.

Bibliografía

- Bennett, Simon. Skelton, John. Lunn, Kenn (2005). *"Schaum's outlines UML"*. Second Edition.
- Lipschutz, Seymour. Lipson, Marc (2008). *"Matemática Discreta Schaum's"*. Second Edition".
- Calderon Vica, Hugo David (2008). *"Matemáticas Discretas para la ciencia de la computación"*.
- Rosen, Kenneth H. (2004). *"Matemática Discreta y sus aplicaciones"*. Quinta Edición.
- Grimaldi. Ralph P. (1997). *"Matemáticas Discreta y combinatoria. Una introducción con aplicaciones"*. Tercera edición.
- Gonzalez Gutiérrez, Francisco Jose (2004). *"Apuntes de Matemática Discreta. (Universidad de Cádiz)"*.
- Ramos Gonzalez, Jose Manuel. (2008). *"Ejercicios resueltos de Matemática Discreta: combinatoria, funciones generatrices y sucesiones recurrentes (Universidad de A Coruña)"*.
- Criado, Regino. Muñoz, Roberto. *"Un semestre de Matemática Discreta (Universidad Rey Juan Carlos)"*.
- Wilhelmi, Miguel R (2004). *"Combinatoria y probabilidad"*.

- *"SVG Tutorial"*.http://www.w3schools.com/svg/default.asp
- "*Scalable Vector Graphics (SVG)"*. http://www.w3.org/Graphics/SVG/
- *"Mathjax"*. http://www.mathjax.org/
- Benavent, Xaro. De Ves, Esther. Gutierrez, Juan (2004). *"Curso avanzado de generación de documentos con LaTeX, HTML y paquetes hyperref, fancyhdr"*. http://www.uv.es/~jgutierr/LatexAvanzado/SesionesEsther/LatexAvanzado_e2.pdf
- Wikipedia http://es.wikipedia.org/

Nomenclatura

AMD Advanced Micro Devices, página 93

RAM Random Access Memory, página 93

SATA Serial Advanced Technology Attachment, página 93

ASF Apache Software License, página 19

W3C World Wide Web Consortium, página 15

XML eXtensible Markup Language, página 15

EPL Eclipse Public License, página 19

GNU General Public License, página 19

HTML HyperText Markup Language, página 34

jsp Java Server Page, página 1

SVG Scalable Vector Graphics, página 4

Anexo

A. Tabla de tiempos de ejecución

Uno de los requisitos principales del proyecto era que la carga de los ejercicios se realizase en un tiempo razonable, inferior a 1 segundo por ejercicio, en esta tabla se indica el número de ejercicio con los tiempos utilizados en cada carga.

Número ej	Descripción y tiempo
EJ00	Contraseñas Diferentes (EJ0): 12 miliseg
EJ01	Palabras diferentes bolas repetidas (EJ1): 5 miliseg
EJ02	Palabras diferentes bolas unívocas (EJ2): 2 miliseg
EJ03	Mensajero (EJ3): 8 miliseg
EJ04	Mesa de navidad (EJ4): 4 miliseg
EJ05	Ratón de laboratorio (EJ5): 20 miliseg
EJ06	Urna con bolas de diferente color (EJ6): 5 miliseg
EJ07	Sorteo Ukumundu (EJ7): 6 miliseg
EJ08	Urna con bolas de igual color (EJ8): 4 miliseg
EJ09	Bandera de Ukumundu (EJ9): 1 miliseg
EJ10	Test de conducir Ukumundu (EJ10): 3 miliseg
EJ11	Metro de Ukumundu (EJ11): 3 miliseg
EJ12	Heladería (EJ12): 4 miliseg
EJ13	Fábrica de flechas (EJ13): 2 miliseg
EJ14	Ingeniero (EJ14): 2 miliseg
EJ15	Bolas múltiplo (EJ15): 2 miliseg
EJ16	Seleccionador de fútbol (EJ16): 6 miliseg
EJ17	Guarda canicas (EJ17): 2 miliseg
EJ18	Profesor de colegio (EJ18): 3 miliseg
EJ19	Fábrica de materiales (EJ19): 2 miliseg

Cuadro 7.1: *Tiempo de carga de cada uno de los ejercicios*