UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

TFG del Grado en Ingeniería Informática

ANEXOS

Generador de preguntas de Programación Dinámica

ProgDinQuiz

Presentado por Asier Alonso Morante

en Universidad de Burgos – <Septiembre de 2016>

Tutores: Juan José Rodriguez Díez

Carlos López Nozal

Índice de contenido

Índice de ilustraciones

Índice de tablas

I - Especificación de Requisitos.

1. Introducción

En este capítulo se recogen todos los requisitos de la aplicación a construir, de tal forma que sirve como contrato entre el cliente y el desarrollador de la misma. A continuación se exponen el conjunto de requisitos, tanto funcionales como no funcionales, mediante una lista y diagramas de casos de uso con sus correspondientes plantillas.

1.1. Requisitos funcionales

La aplicación deberá ser capaz de cumplir los requisitos citados a continuación:

RF1: La aplicación deberá ser capaz de generar problemas tipo de la programación dinámica con parámetros recibidos por el usuario y valores generados aleatoriamente:

RF1.1: Los problemas que generará la aplicación serán ejemplos de problemas capaces de ser resueltos por el método de Programación Dinámica, en este proyecto se generarán los problemas de la Mochila, Subsecuencia Común más Larga, Algoritmo de Floyd y Multiplicación encadenada de Matrices, más explicados en la memoria, en la sección de Conceptos teóricos (1.3).

RF 1.2 Cada problema tendrá una semilla que se creará a partir de los parámetros de cada problema y de la que se generarán los valores aleatorios de los problemas.

RF1.3: La aplicación permitirá visualizar en una pantalla todos los problemas que se han generado desde el inicio de la aplicación.

RF2: La herramienta deberá ser capaz de exportar a diferentes formatos todos los problemas generados por en la aplicación:

RF2.1: Uno de los formatos a los que se permita exportar deberá ser en un formato XML importable por el entorno de aprendizaje de Moodle.

RF3: Puesto que cada problema tendrá una semilla única, la herramienta permitirá recuperar un problema a partir de una semilla válida introducida.

RF4: El usuario podrá visualizar una ayuda sobre la aplicación.

1.2. Requisitos no funcionales

Eficiencia: Se debe buscar la eficiencia de la aplicación para que pueda ejecutarse cualquier navegador de cualquier ordenador o dispositivo.

Escalabilidad: Facilidad para que implementar nuevas preguntas, nuevos problemas o nuevos formatos para exportar sea más sencillo y permita modificar el menor código posible.

Sencillez: se busca crear un código poco complejo y bien comentando que sea fácil de asimilar por desarrolladores que quieran implementar nuevas funcionalidades en la aplicación. También se busca una sencillez dentro del diseño gráfico de la aplicación, con una interfaz intuitiva y fácil de usar.

1.3. Diagrama de casos de uso

En esta subsección se puede ver el diagrama general de casos de uso de la aplicación (ver Ilustración 1). Además de este diagrama, en esta sección se añaden cada una de las tablas correspondientes a éstos:

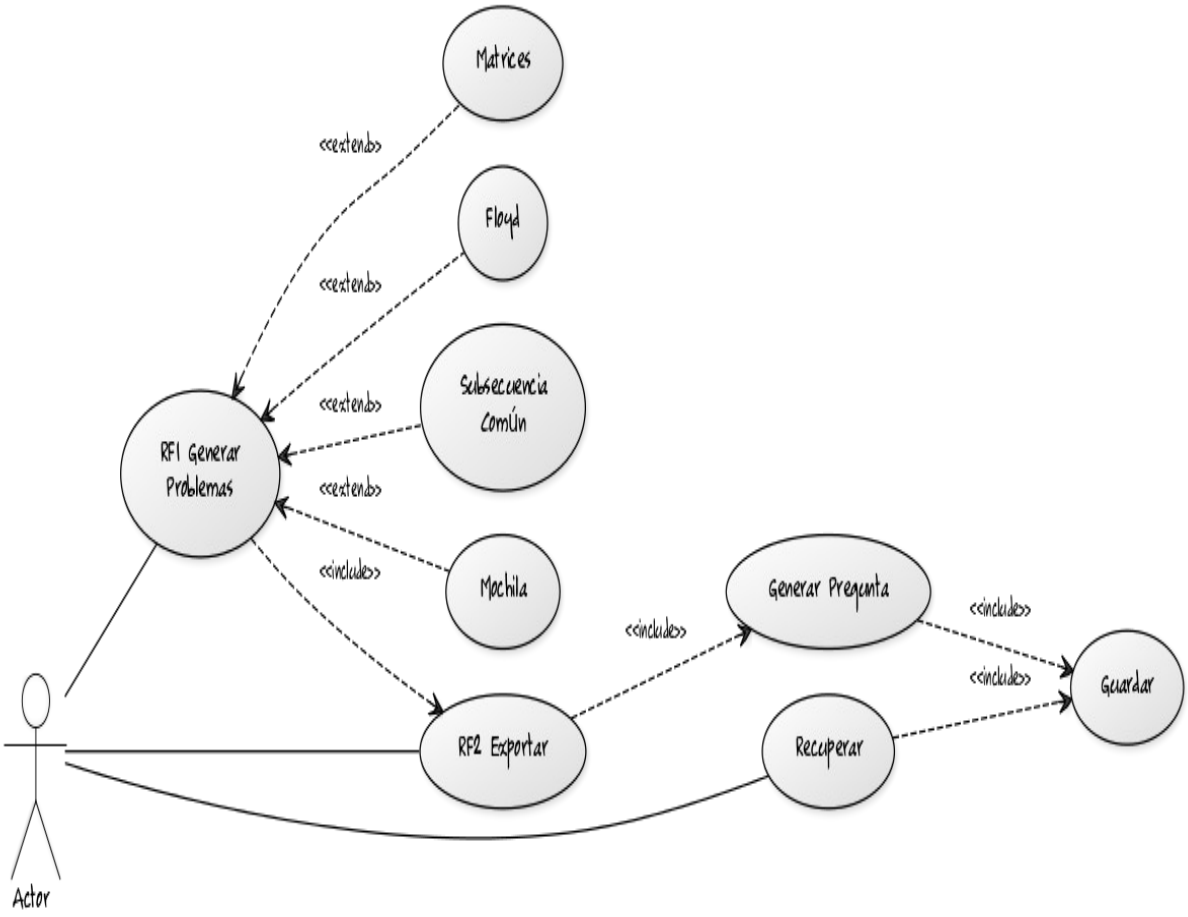


Ilustración 1: Diagrama de Caso de Uso de la Aplicación

RF1. Generar Problema	
Actor Principal:	Usuario de la herramienta
Precondiciones:	Pulsar en la pantalla principal el botón del problema deseado.
PostCondiciones:	Si se genera un problema aparecerá un mensaje de información en la pantalla.
Extensiones:	Una vez generado, el sistema añadirá el problema a una lista de problemas generados y el usuario podrá pasar a la pantalla de Exportar si desea exportarlo.
Requisitos especiales,	Introducir parámetros válidos para generar el problema.
Frecuencia:	Alta

Tabla 1: Caso de Uso de Generación de Problemas

RF2. Exportar Problema	
Actor Principal:	Usuario de la herramienta
Precondiciones:	Debe existir al menos un problema generado dentro del sistema.
PostCondiciones:	Si se genera un problema aparecerá un mensaje de información en la pantalla.
Extensiones:	
Requisitos especiales,	Después de exportar el problema, existe un archivo en el formato especificado que podremos utilizar en nuestro sistema, fuera de la herramienta.
Frecuencia:	Alta

Tabla 2: Caso de Uso Exportar Problemas

RF3. Recuperar Problema	
Actor Principal:	Usuario de la herramienta
Precondiciones:	Debe introducirse una semilla de problema válida.
PostCondiciones:	Una vez recuperado el problema, existirá la posibilidad de almacenar el problema, y añadirlo a la lista interna de la aplicación de problemas generados.
Extensiones:	Una vez recuperado el problema y almacenado se podrá exportar el problema con la semilla introducida por el usuario.
Requisitos especiales,	
Frecuencia:	Alta

Tabla 3: Caso de Uso Recuperar Problemas

II - Especificación de diseño

En este capítulo visualizaremos la forma en la que se ha diseñado el proyecto, primero con un diagrama de paquetes, y después con los diagramas de clases.

1.1. Diagramas de paquetes.

La aplicación está organizada con la siguiente estructura de cinco paquetes.

- *Proyecto.src.exportar:* en este paquete se encontraran todas las clases que hacen referencia a la forma de exportar las preguntas generadas por la aplicación.

- *Proyecto.src.gui:* este paquete contiene todas las clases creadas con Window Builder donde se definirán las ventanas de la aplicación.

- *Proyecto.src.problemas:* este paquete contiene la lógica de la aplicación, en el estan definidos y resueltos todos los problemas de la aplicación.

- *Proyecto.src.preguntas:* en este paquete se encuentran las clases que definen las preguntas que se realizan en la aplicación.

- *Proyecto.src.utiles:* el paquete útiles contiene clases con aquellos métodos y variables que son comunes y accesibles a todas las clases de la aplicación.

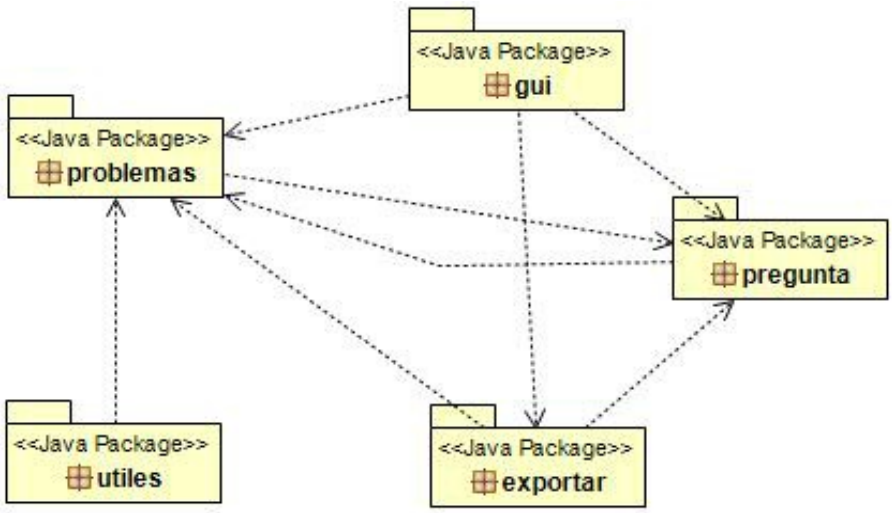


Ilustración 2: Diagrama de clases de paquetes

1.2. Diagramas de clases

1.3. Diagrama clases paquete exportar

En la siguiente imagen observamos el diagrama de clases del paquete exportar, lo mas llamativo es el uso del Patrón de diseño Estrategia.

El usuario de forma dinámica elegirá en que formato deseará exportar las preguntas generadas, por lo tanto, el patrón que mas se adecuaba a esta necesidad era este.

Tenemos una interfaz, Exportar, con unos métodos comunes a todas las clases que heredarán de ella, cada una de esas subclases creará las preguntas en un formato determinado.

Para determinar que clase utilizar según las indicaciones del usuario, creamos la clase ExportarEstrategia, que recibirá un atributo indicando a qué clase debe llamar para continuar con la operación.

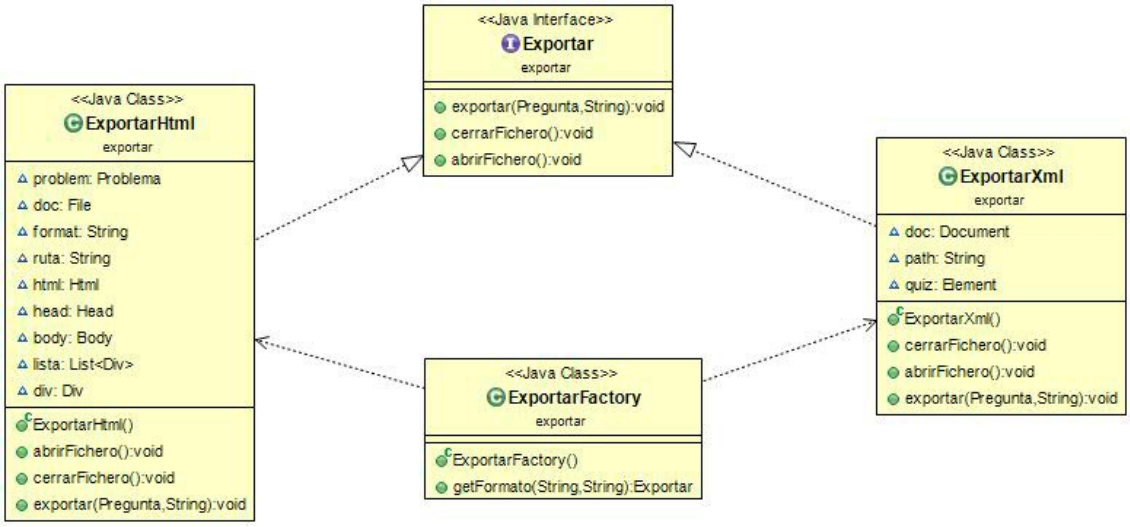


Ilustración 3: Diagrama de clases Exportar

1.4. Diagrama de clases paquete problemas

El paquete problemas que define la lógica de la aplicación contiene una interfaz Problema que contendrá los métodos comunes a todos los problemas que se creen en la aplicación, ya que heredarán de esa misma clase.

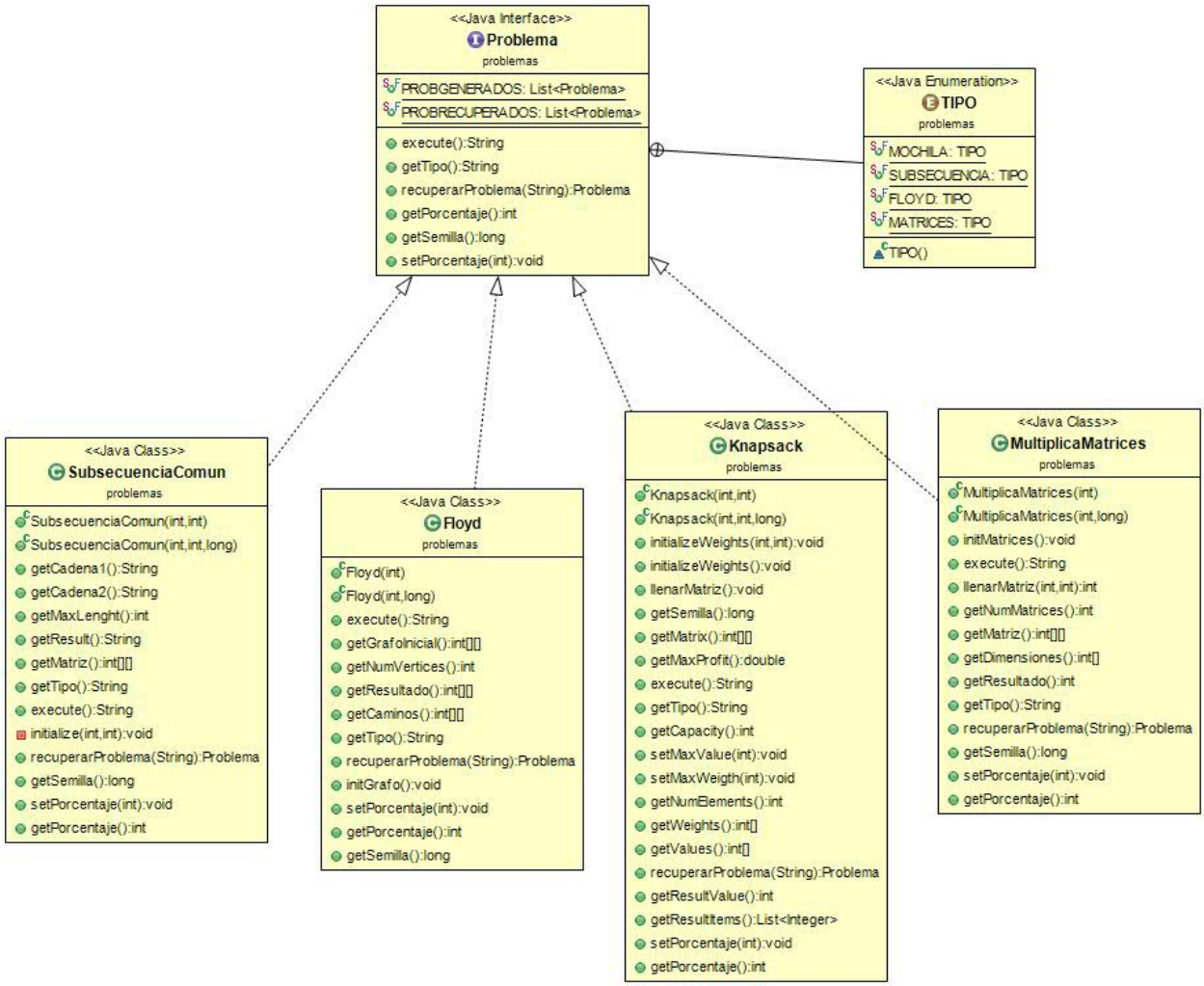


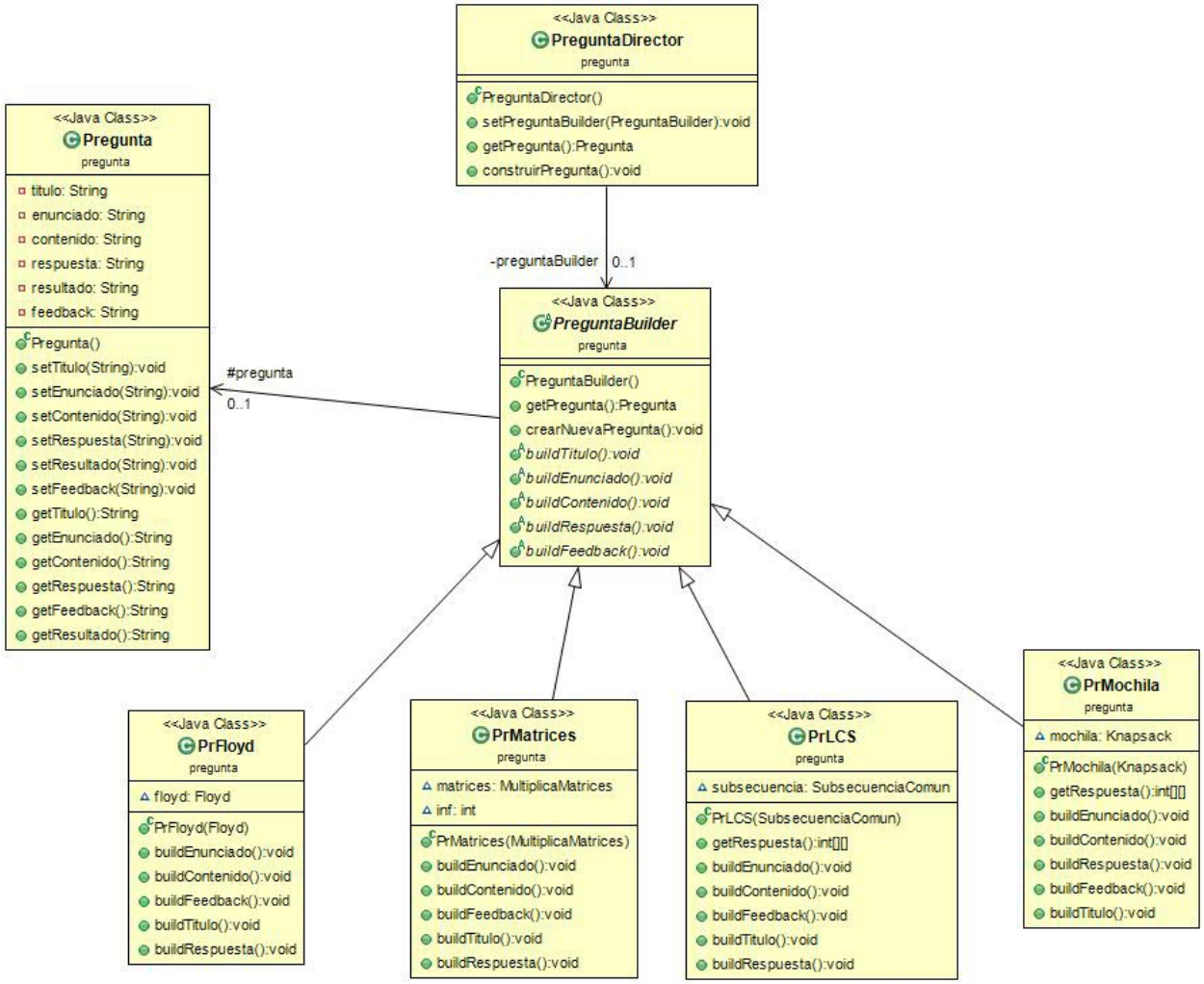
Ilustración 4: diagrama clases paquete problemas

1.5. Diagrama de clases paquete Pregunta

En este paquete también utilizamos un patrón de diseño, en este caso el patrón Builder, (ver información en memoria, conceptos teóricos).

Con este patrón, tendremos una clase Pregunta Director, una clase abstracta Pregunta Builder y varias clases, una por cada formato, que heredarán de la clase abstracta Pregunta Builder

Todas las subclases contendrán los mismo métodos y será la clase Pregunta Director, la encargada de determinar los métodos que serán ejecutados.



III - Manual del Programador

1. Introducción

La aplicación está desarrollada en Java, por lo que podrá ser instalada y utilizada en cualquiera plataforma que tenga instalada la máquina virtual de Java (JVM).

El código de la aplicación se encontrará en el CD entregado al tribuna y ademas se encontrará alojado en un repositorio de github en la siguiente Url:

https://github.com/aam0093/Proyecto-Quiz-Prog.-Dinamica.git

2. Estructura de los directorios

Tanto en el CD como en el repositorio, la estructura de directorios será la misma. Dentro de la carpeta src, se encontrarán todos los paquetes de los que se compone la herramienta, para ver el contenido de cada paquete ver la sección de Aspectos Relevantes.

En la carpeta principal se encontrará también el ejecutable llamado *ProgDinQuiz.jar.* Ejecutando ese fichero comenzará a ejecutarse la aplicación con la página inicial de la aplicación.

Existirá otra carpeta llamada *doc* donde se encontrará la memoria junto con los anexos de la misma.

3. Instalación de la máquina virtual de Java (JVM)

Para ejecutar la aplicación será la única herramienta que se necesitará en el sistema. Saber si se encuentra instalada en el sistema es muy sencillo, bastará con acudir al símbolo del sistema e introducir el comando, *java -version,* en caso de estar instalada devolverá algo la versión que estará instalada.

```
C:\Users\asier_000>java -version
java version "1.7.0_17"
Java(TM) SE Runtime Environment (build 1.7.0_17-b02)
Java HotSpot(TM) 64-Bit Server VM (build 23.7-b01, mixed mode)
```

De no existir JVM en el sistema, devolverá un mensaje semejante a que no ha sido encontrada.

Si es necesario instalar o actualizar la máquina virtual de Java, los pasos a seguir serán los siguientes:

1. Dirigirse a la página de java: http://www.java.com/es/
2. Una vez allí seleccionar la opción descarga gratuita de Java y comenzará la instalación.
3. Al aceptar los términos del contrato comenzará la descarga de la máquina
4. Ejecutar el instalador y seguir los pasos para instalar JVM

Una vez finalizado, para comprobar si la instalación se ha realizado correctamente, se vuelve a introducir el comando *java -version* y se verá como el mensaje ha cambiado.

Con la máquina virtual de Java ya instalada lo único que hará falta para que la aplicación comience a ejecutarse, es ir a la carpeta principal del CD o del repositorio Github y arrancar el fichero ejecutable *ProgDinQuiz.jar*Ejecutar la aplicación

Una vez instalada la máquina virtual correctamente, será sencillo ejecutar la aplicación, ya que como se ha indicado anteriormente, solo será necesario utilizar el ejecutable alojado en la carpeta principal y ejecutarlo para que poder comenzar a utilizar la aplicación.

4. Pruebas de la aplicación

4.1. Test unitarios

En este proyecto se han realizado también alguna prueba de test de la aplicación. Puesto que muchos de los datos que se han utilizado por los problemas son generados de forma aleatoria, ha resultado mas complicado realizar los test.

La batería de test se han realizado con la herramienta Junit y están alojados en el paquete de la aplicación *test.*

Los test se realizan sobre la lógica de la herramienta, esto es, sobre las clases del paquete *problemas* donde se creará cada problema.

Dentro de las clases que formarán la interfaz se intentarán controlar todas las entradas de datos para que sólo permitan la introducción de datos válidos al sistema y muestren un mensaje de error en caso de no ser válidos.

En la siguiente imagen () se pueden ver como los test han sido superados por la aplicación.

Ilustración 5: Test unitarios de la aplicación

4.2. Test funcionales

Los test que más se han realizado y los más útiles, han sido test funcionales, ya que los test unitarios no podían determinar bien si los problemas eran ejecutados de forma correcta debido a los valores aleatorios que poseían.

Mediante los test funcionales se ha comprobado que los elementos que se exportan eran correctos, que la aplicación funciona como se desea, y que se muestra estable mientras se ejecuta.

IV - Manual de usuario

1. Descripción de la aplicación

Esta aplicación se compone de un problemas sobre la programación dinámica. Estos problemas serán generados con valores aleatorios por la aplicación. Al generar los problemas se podrá controlar como serán las preguntas, el porcentaje de incógnitas a resolver por el alumno tendrán.

Una vez generados los problemas, la herramienta permite exportarlos a diferentes formatos como Html, Moodle xml o Pdf.

La herramienta también permite, además de la generación y exportación de problemas, la recuperación de los problemas introduciendo una semilla válida en el programa.

2. Guía de instalación

A continuación se detalla como instalar el software necesario para el funcionamiento de ProgDinQuiz y la instalación aplicación elaborada para el proyecto. ´

Todo el software necesario para la instalación de la aplicación en un PC de sobremesa se proporciona en el CD-ROM que se adjunta con la memoria, no obstante puede obtenerse también en

https://github.com/aam0093/Proyecto-Quiz-Prog.-Dinamica.git

Para utilizar la herramienta en una máquina, simplemente se necesita tener instalada la máquina virtual de java (JVM), en caso de no disponer de ella, se puede descargar fácilmente de la página de Oracle.

Después de verificar que esta disponible la máquina virtual de Java, lo único que se necesitará es descargar el fichero .jar de la carpeta /bin y ejecutarlo.

3. Manual de usuario de la aplicación

3.1. Inicio

La página de inicio es la página principal de la aplicación, en ella podremos ver todas las funcionalidades que nos da la aplicación.

Generar cualquiera de los problemas pulsando en los iconos de cada uno de ellos que abrirán la ventana correspondiente para poder introducir los parámetros deseados para cada problema.

Exportar los problemas, en el caso de que exista algún problema generado, o Recuperar algún problema introduciendo una semilla.

También aparecen los iconos de la ayuda, o el botón “A cerca de” con la información del autor y fecha de la aplicación.

Ilustración 6: Pantalla de Inicio

3.2. Mochila

Al pulsar sobre el icono de la mochila en la pantalla de inicio, aparecerá la siguiente ventana.

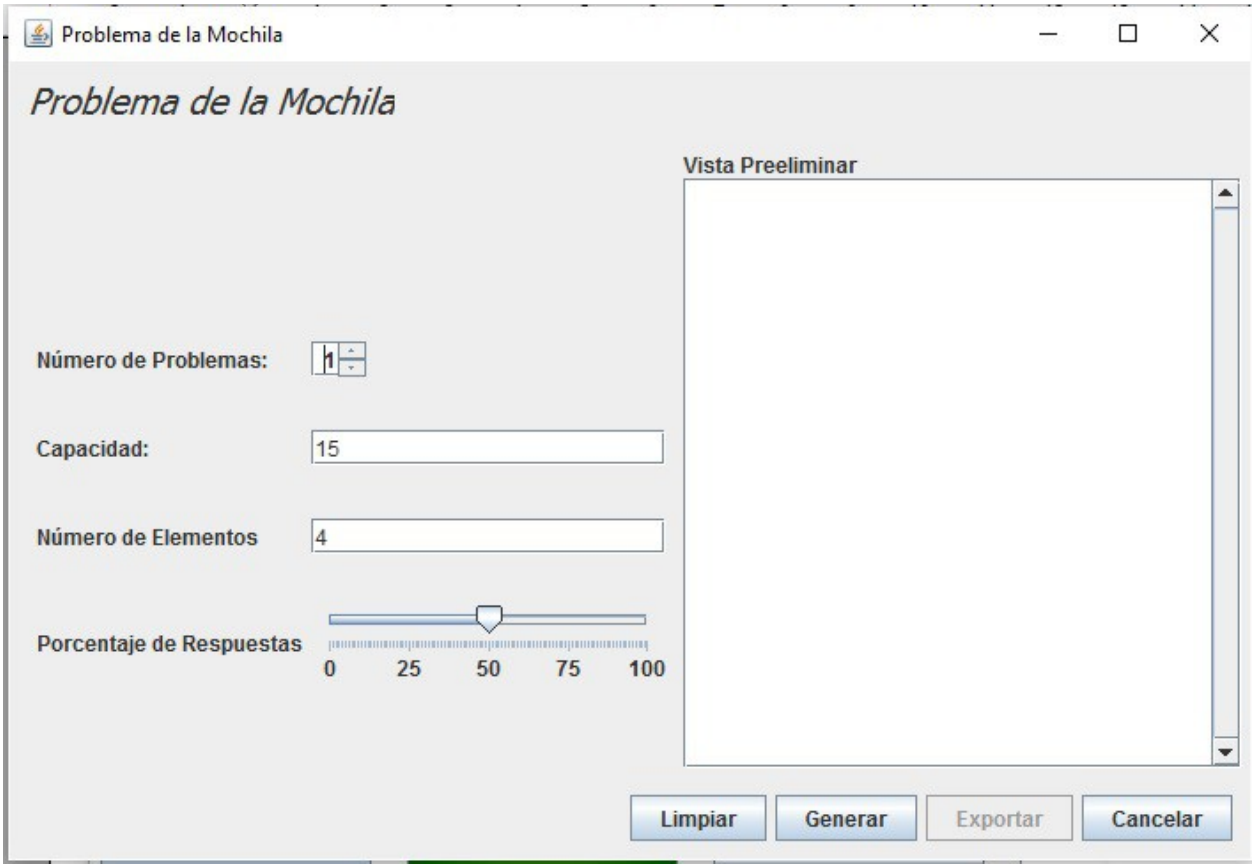


Ilustración 7: Pantalla Problema Mochila

En ella se ven tres paneles diferentes.

En el panel de la izquierda, aparecerán todos los parámetros de los problemas que se vayan a generar. En este caso, tendremos un texto donde introducir la *Capacidad* y *Número de Elementos* que deseemos para los problemas.

El slider de Porcentaje de Respuestas, permitirá seleccionar el número de incógnitas que tendrá el problema. Si se selecciona el valor 0, el problema no tendrá ninguna incognita, luego se exportará entera resuelta, por el contrario si se selecciona el valor 100, indicará que el 100% de las incógnitas aparecerá resuelta a la hora de exportar el problema.

En el panel derecho de la ventana, podremos ver los problemas que se han generado en la aplicación, es una vista global del problema, por lo tanto aparecerá la solución completa, sin ninguna incógnita. Es simplemente para verificar la forma que tendrá el problema antes de ser exportado.

El panel inferior, estarán los botones de la pantalla, el botón *Exportar* no estará habilitado hasta que no exista algún problema generado dentro de la aplicación. El botón *Limpiar*, reseteará el

panel de vista, pero no borrará los problemas de la aplicación.

3.3. Subsecuencia común mas larga

Variará con respecto al de la mochila en los parámetros que pedirá. El resto del funcionamiento es idéntico al problema de la Mochila.

Los parámetros que habrá que indicarle al problema, son los de la longitud de las dos cadenas. El resto de valores los generará la herramienta. Para la generación de cadenas seleccionará los caracteres que van desde la *a* hasta que finalice el tamaño la cadena indicado.

Una vez generado el problema, aparecerá en el panel vista a continuación del resto de problemas generados anteriormente.

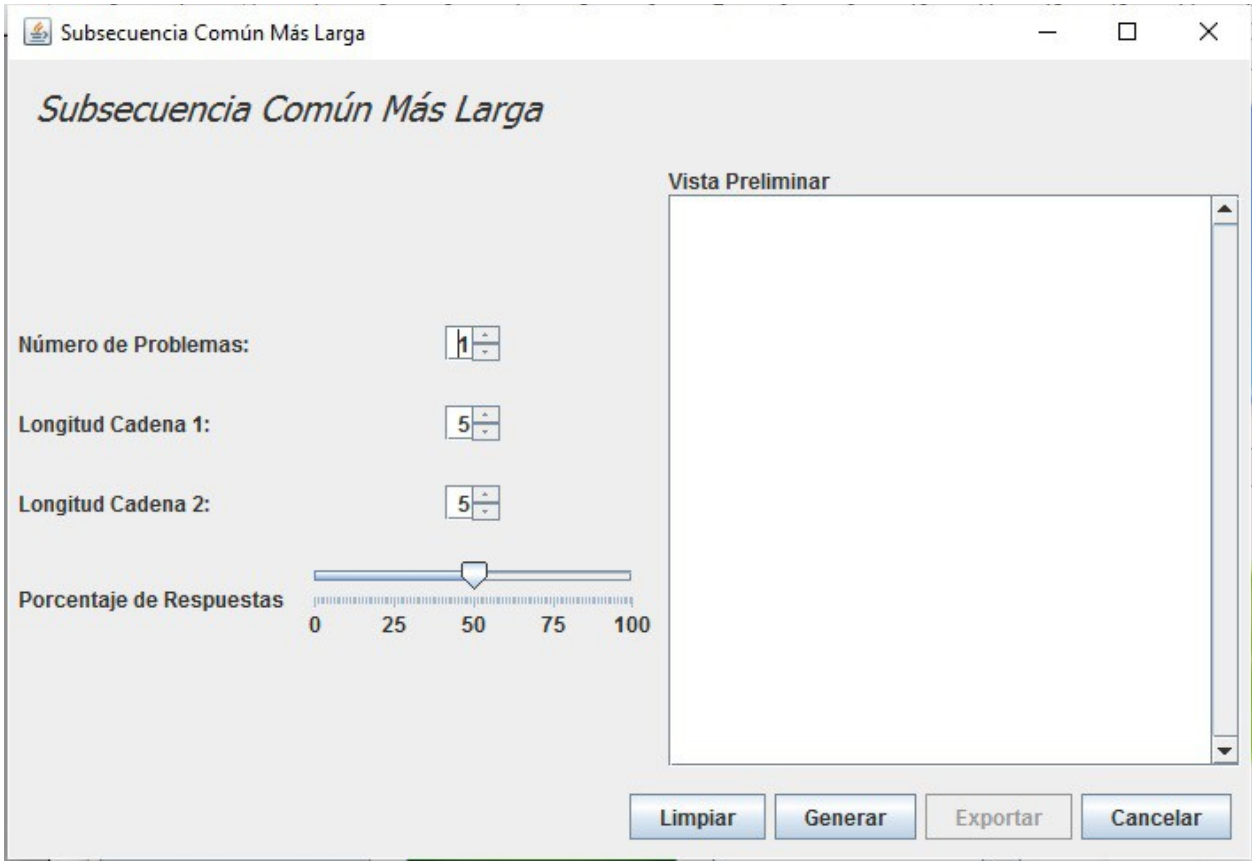


Ilustración 8: Pantalla Problema Subsecuencia Común Más Larga

3.4. Floyd y Multiplicación de Matrices

Poseen la misma funcionalidad que los anteriores, reciben un parámetro que corresponderá al número de vértices o número de matrices que se incluirán, respectivamente. Por defecto en ambas pantallas el número de matrices y de vértices es de 2, ya que no tendría sentido que fuese menor.

En ambos casos, la herramienta generará con números aleatorios, los caminos que formarán el grafo que solucionará el algoritmo de Floyd y las dimensiones que tendrán las matrices.

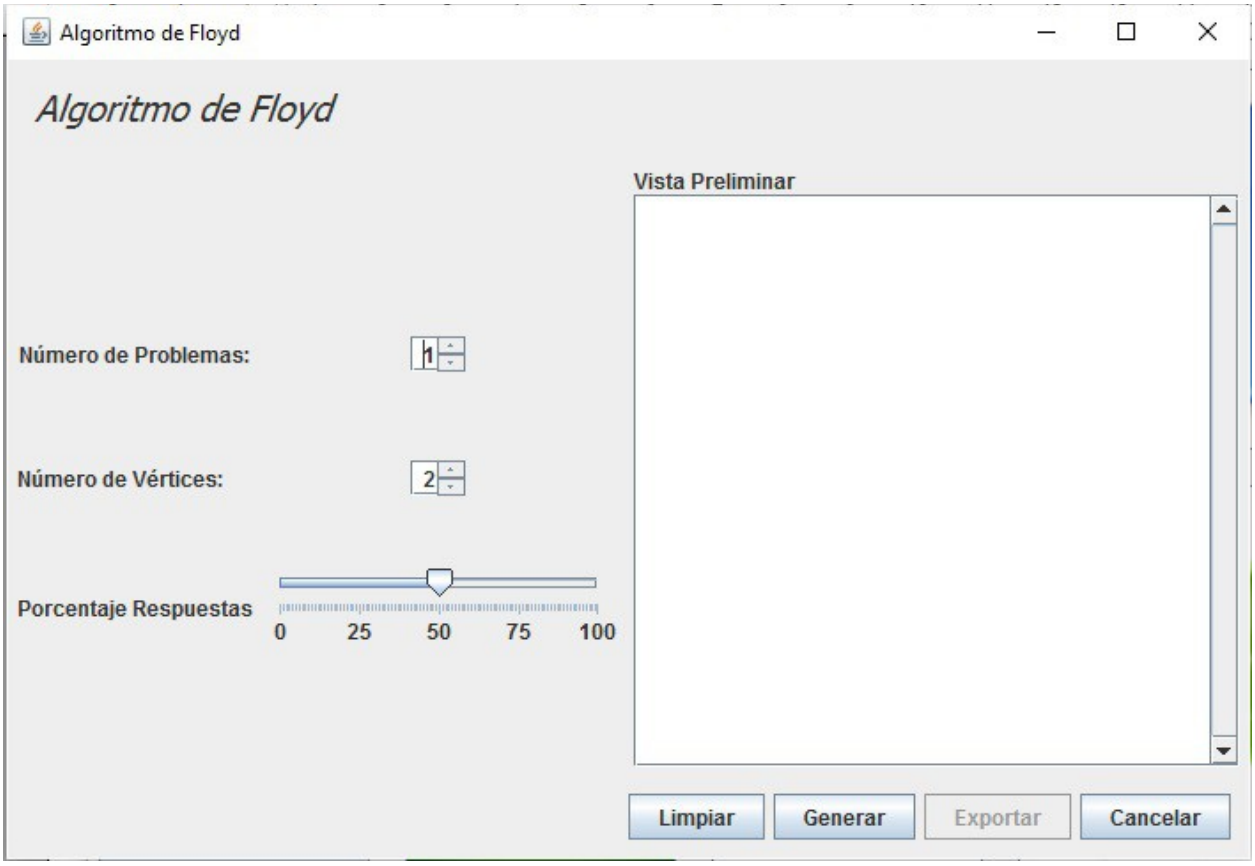


Ilustración 9: Pantalla Algoritmo de Floyd

3.5. Exportar

La pantalla *Exportar* no estará disponible hasta que no se haya generado algún problema dentro de la aplicación.

Dentro de la pantalla, habrá una Textbox donde se pedirá introducir una ruta donde almacenar todos los problemas generados hasta ese momento desde que se abrió la aplicación. Por defecto se almacenará en la carpeta donde se esté ejecutando la aplicación con el nombre "Problema + la fecha del dia en que se generen".

Para seleccionar un directorio, tendremos el botón Browse, que nos llevará por todos los directorios del sistema. Una vez seleccionado el directorio, introduciremos un nombre para el fichero. Y pulsaremos en *Guardar*.

Si al pulsar en Guardar no se ha determinado una extensión al fichero, la herramienta permitirá seleccionar una de entre las tres que tiene disponibles y añadirá la extensión al documento.

De lo contrario, si se ha indicado la extensión deseada, si corresponde con algún formato de los que dispone la aplicación, la herramienta no permitirá seleccionar otro formato para que no existan dos indicaciones contradictorias.

Ilustración 10: Pantalla Exportar

3.6. Recuperar

En la pantalla *Recuperar,* encontraremos un campo de texto donde introducir una semilla. No valdrá usar cualquier número ya que se controlará que el número introducido sea una semilla válida.

Si la semilla introducida es válida, se mostrará por el panel vista de la pantalla el problema generado. Existirá la opción de enviar todos los Problemas Recuperados a la lista de Problemas Generados como si hubiesen si creados en las pantallas de cada problema.

De esta forma podremos volver a exportar un problema que ya había sido generado con anterioridad.

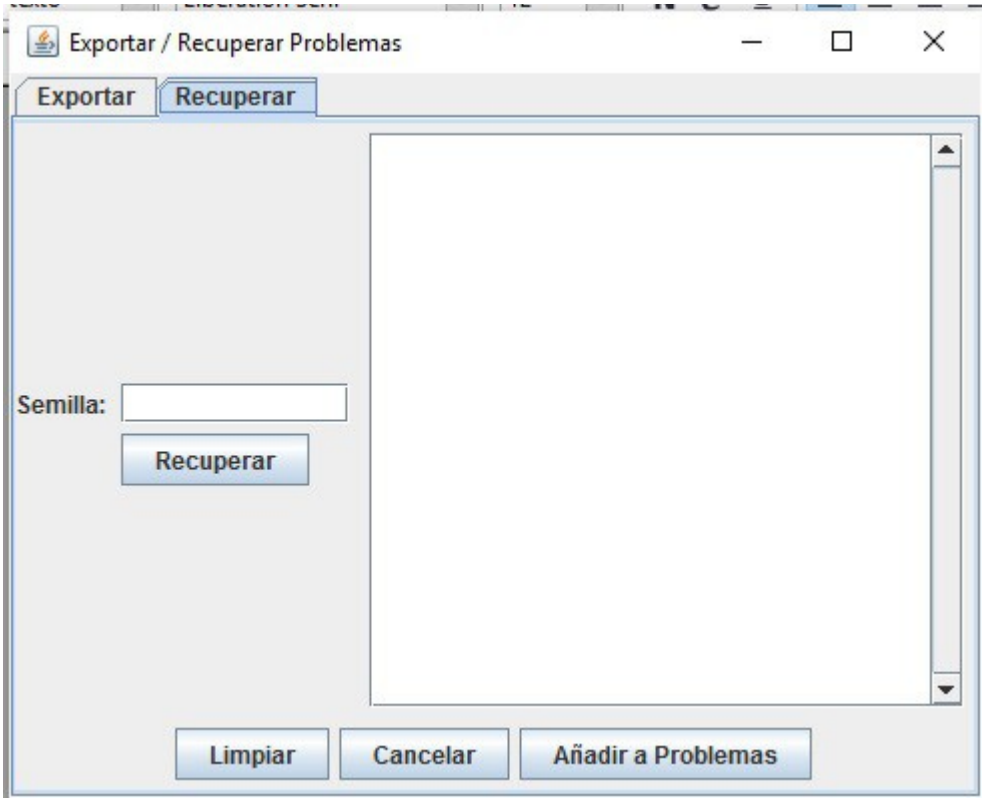


Ilustración 11: Pantalla Recuperar

.

3.7. Acerca De

El botón Acerca De, simplemente nos mostrará un poco de información sobre la aplicación.

Ilustración 12: Pantalla Acerca De

V - REFERENCIAS

Impreso en Burgos el viernes, 16 de septiembre de 2016