Práctica 3 - Encapsular el acceso a una aplicación BASIC/MS-DOS

Lunes, 7/12/20

Eduardo Gimeno - 721615

Jorge Turbica - 723883

José Navarro - 721230

Preparación

En primer lugar se han obtenido todos los archivos necesarios para ejecutar el simulador de MS-DOS y se ha probado su funcionamiento introduciendo los datos desde un teclado normal. Se han analizado los ficheros que componen la aplicación para un mejor entendimiento de cómo está almacenada la información y se ha intentado trastear con el simulador para ver si había alguna funcionalidad que pudiera ser de utilidad.

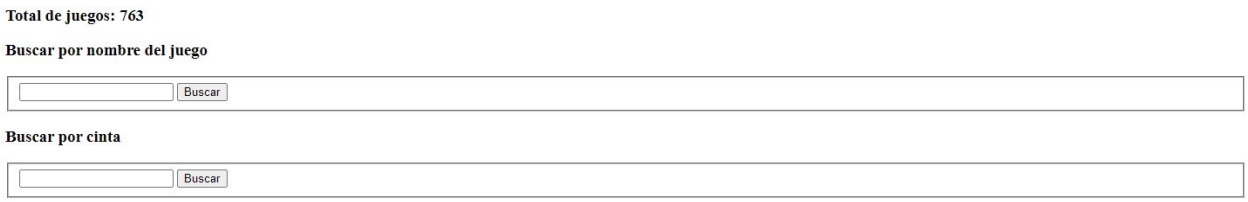
A la hora de decidir cómo plantear el proyecto se han barajado varias opciones. En primer lugar decidir el sistema operativo sobre el que se quería realizar las pruebas y desplegar el sistema; y en segundo lugar las tecnologías para conseguir el objetivo.

Finalmente se ha decidido desplegar una aplicación JAVA construida a través de Spring Boot sobre una máquina Windows, que ejecuta el simulador a su vez.

Web y EndPoints

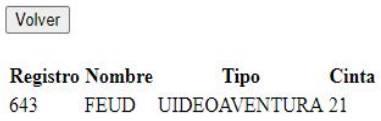
La razón principal por la que se ha decidido emplear Spring Boot es por la posibilidad ya preparada para crear una interfaz web que permita la introducción de texto para las peticiones y los controladores que las reciben. Para la carga de las páginas web con contenido dinámico se ha optado por Thymeleaf.

De esta manera se ha creado un portal muy sencillo que al abrirlo ya realiza la primera petición para calcular el total de juegos en el sistema.



Como se ve en la imagen los otros dos EndPoints son para realizar las dos funciones que se piden: buscar por el título del juego y buscar los títulos que aparecen en una cinta.

Este es el resultado tras buscar el juego FEUD:



Y este es el resultado tras buscar todos los juegos de la cinta X:

Wrapper

Para realizar la interacción con el simulador se ha empleado la clase Robot de Java sugerida en el enunciado. De esta manera se ha creado un servicio en el sistema que permite traducir los textos introducidos en pulsaciones equivalentes en el simulador sin necesidad de la interacción humana.

Para obtener el comportamiento esperado del sistema y alcanzar los resultados deseados se han creado a su vez las pulsaciones necesarias para moverse por los diferentes menús y preguntas que van apareciendo.

Para evitar tener que ejecutar cada vez la máquina de simulación se han creado todas las posibilidades existentes de error, de manera que siempre se acaba llegando al menú principal y estar listo para la siguiente ejecución.

En este servicio de wrapper también se ha creado una función que captura la parte de la pantalla donde se ejecuta el simulador.

```
public void pressKey(char character) {
  try {
     boolean upperCase = Character.isUpperCase( character );
     Thread.sleep(250);
     if (upperCase) robot.keyPress( KeyEvent.VK_SHIFT );
     robot.keyPress(KeyEvent.getExtendedKeyCodeForChar(character));
     robot.keyRelease(KeyEvent.getExtendedKeyCodeForChar(character));
     if (upperCase) robot.keyRelease( KeyEvent.VK_SHIFT );
  } catch (InterruptedException e) {
     e.printStackTrace();
  }
}
public BufferedImage takeCapture() {
  return robot.createScreenCapture(new Rectangle(0, 40, 650, 400));
}
```

OCR

Para la interpretación de las capturas que se toman con el wrapper, se ha decidido emplear el proyecto Tesseract (en Java 'Tess4J'). Se ha elegido por su precisión y por tener una evolución continuada.

Durante las pruebas previas realizadas para la detección de texto se observó que el fichero de entrenamiento proporcionado solo resolvía correctamente las líneas pertenecientes a los resultados de los juegos tras las búsquedas. Por ello se han buscado los ficheros de entrenamiento más recientes para el Español y se han descargado. Tras probar con los nuevos se ha observado que las listas de juegos ahora no se entienden pero los menús se detectan claramente.

De esta manera se ha configurado el sistema para hacer reconocimientos personalizados en los que a la hora de realizar la identificación se puede elegir que datos de entrenamiento usar. Con esto se ha conseguido mejorar considerablemente la detención de la interfaz del simulador y obtener datos más fiables.

Distribución del trabajo

Tarea	Eduardo	Jorge	José
Preparación			
Web y Controladores			
Wrapper			
OCR			
Informe			

Bibliografía

Tess4J - JNA wrapper for Tesseract

Optical Character Recognition with Tesseract

How to capture selected screen of other application using java?

How to capture screenshot programmatically in Java

Robot (Java Platform SE 7)

Tess4j doesn't use it's tessdata folder

tesseract-ocr

When is it safe (or mandatory) to use Image.flush()?

RegExr: Learn, Build, & Test RegEx

The Java Headless Mode

Spring Request Parameters with Thymeleaf

Tutorial

Binding a List in Thymeleaf

Spring Boot Thymeleaf y su configuración