

Arkaitz Garro Elgueta



Contenido

Prólogo 5 Bienvenido al Manual del estudiante de Epaile

Capítulo 1 6 Introducción al módulo Tareas

Capítulo 2 7 Utilización de Epaile

9 Envío de soluciones

10 Consulta de notas

Bienvenido al *Manual del estudiante de Epaile*

En este documento PDF se detallan las instrucciones de utilización de Epaile por parte del estudiante.

En este documento se detallan las instrucciones necesarias para la utilización del plugin Epaile en el sistema por parte del estudiante. Esto incluye el envío de soluciones y su evaluación automática por parte del sistema.

Si necesita instrucciones detalladas para realizar tareas específicas, o cree que es necesario aclarar algún punto de este manual, no dude en contactar conmigo:

Arkaitz Garro: arkaitz.garro@gmail.com

Introducción al módulo *Tareas*

Una pequeña introducción al módulo Tareas.

En Moodle una *Tarea* es cualquier trabajo, labor o actividad que nos es asignada por parte de los y que no está cubierta por otro módulo de Moodle. Típicamente los alumnos debemos devolver la solución al problema planteado como un fichero de ordenador: un documento de texto con un trabajo escrito, una presentación con diapositivas, una imagen gráfica, un video, un programa de ordenador: cualquier cosa susceptible de ser encapsulada como un archivo informático.

Concretamente para el plugin Epaile, el entregable se corresponde con un programa informático que debe ser evaluado por el propio sistema, ateniéndose a los criterios de evaluación definidos por el profesor. En el siguiente capítulo se muestra la forma de enviar esos entregables.

Utilización de Epaile

En este capítulo se explica como hacer uso de Epaile para enviar las soluciones a un ejercicio.

Durante la realización de un curso, el profesor o profesores proponen una serie de ejercicios de programación que debemos resolver. Esos ejercicios son mostrados como *Tareas* en la planificación semanal del curso.



Podemos acceder a la descripción de las tareas a través de su nombre. Se mostrará una página con información detallada sobre la tarea a realizar. Los datos mostrados son los siguientes:

 Descripción detallada sobre la tarea a realizar. Esta descripción incluye una explicación del ejercicio a resolver, las precondiciones y poscondiciones que debe cumplir el programa y un pequeño ejemplo de entrada-salida.

7

 Lenguaje de programación: lenguaje de programación sobre el que se tiene que desarrollar la solución al ejercicio.

```
Lenguaje de programación: Java
```

Fecha de inicio y fecha límite de entrega de la solución.

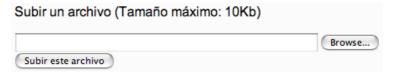
```
Disponible en: lunes, 3 de septiembre de 2007, 16:40

Fecha límite de entrega: domingo, 23 de septiembre de 2007, 16:40
```

• Último fichero enviado como solución, si se ha enviado alguno.

Último fichero enviado:	Fibonacci.java

• Campo de formulario que permite el envío de la solución. El sistema no permitirá enviar el fichero después de la fecha y hora límite. Dependiendo de la configuración de la tarea, una vez enviado el archivo la tarea puede seguir abierta y permitir que usted envíe nuevas versiones del trabajo. Cada versión reemplazará a la anterior. El profesor sólo verá la última que usted haya mandado.



Una vez calificada la tarea, cuando entremos en la misma veremos un cuadro de texto que indicará qué profesor ha evaluado el trabajo, la fecha de la evaluación, la calificación obtenida y un comentario más o menos largo sobre el ejercicio. Adicionalmente, se mostrarán los posibles avisos o errores de compilación.



Bajo la calificación global y los posibles comentarios del profesor o sistema, se muestran los resultados obtenidos en cada uno de los casos de prueba con los que se ha testeado la solución enviada.

Ξ	Status ⊡	Error -	Score 🗆
Test #1	Correcto		33,33
Test #2	Correcto		33,33
Test#3	Respuesta incorrecta.		0,00

Envío de soluciones

Como hemos comentado anteriormente, si la tarea se encuentra abierta y existe la posibilidad de enviar soluciones, se mostrará un campo de formulario que permite seleccionar el fichero y enviarlo al servidor para su evaluación.

Así pues, seleccionamos el fichero correcto y pulsamos en *Subir este archivo*. El fichero seleccionado será enviado al servidor y comenzará la evaluación automática. Tenga paciencia y no intente parar ni recargar la página. Los resultados no tardarán en aparecer.

Una vez que haya concluido la evaluación automática, podemos obtener diferentes resultados. Uno de ellos es que se hayan producido errores al compilar el código fuente, por lo que el sistema nos invitará a revisar nuestro código fuente y nos ofrece la posibilidad de volver a enviar la solución.

```
Compile output

1. GRAGN in ./Main.java (at line 28)
    public class Fibonacci {
The public vpc Fibonacci must be defined in its own file

2. GRAGN in ./Main.java (at line 30)
    if (n <= 1) return n;

Syntax error on token "=", delete this token

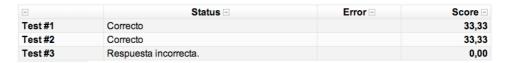
2 problems (2 errors)

[[Compile errors. Please, check your code.]]

More information about this error

Continue
```

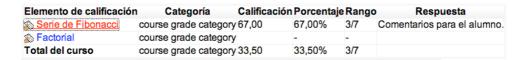
En el caso de que el código fuente compile sin errores, se evaluará la solución y se mostrará la calificación obtenida.



Consulta de notas

Para consultar las notas que vamos obteniendo a lo largo del curso, accedemos al apartado *Calificaciones*, que encontraremos en el menú de *Administración*del curso.

Se mostrará una tabla en la que aparecen los ejercicios existentes a lo largo del curso, mostrando para cada uno de ellos el nombre, la calificación obtenida, el porcentaje sobre el total, la posición con respecto a la clase y los posibles comentarios realizados por el profesor. Además, a pie de tabla se muestra la calificación media obtenida a lo largo del curso.



A través del nombre del ejercicio podemos acceder a la pantalla de visualización de datos del ejercicio.

Formato en el envío de soluciones

A continuación se muestran algunos códigos de ejemplo con el formato que deben cumplir al ser enviados por los alumnos. Estos ejemplos son soluciones para el siguiente problema: dado un entero n y tantos *strings* como sea definido en n, escribir por la salida estándar "Hola <nombre>!".

Entrada	Salida
3	Hola Mundo!
Mundo	Hola Arkaitz!
Arkaitz	Hola Juanan!
Juanan	

Solución en C

```
#include <stdio.h>
int main()
{
  int ntests, test;
  char name[100];
  scanf("%d\n", &ntests);

for (test=1; test <= ntests; test++) {
    scanf("%s\n", name);
    printf("Hello %s!\n", name);
}

return 0;
}</pre>
```

NOTA: los programas escritos en C siempre deben devolver 0 como valor de retorno.

Solución en C++

```
using namespace std;
#include <iostream>
#include <string>
int main()
{
  int ntests, test;
  string name;
  cin >> ntests;

for (test=1; test <= ntests; test++) {
    cin >> name;
    cout << "Hello " << name << "!" << endl;
}
  return 0;
}</pre>
```

NOTA: los programas escritos en C++ siempre deben devolver 0 como valor de retorno.

Solución en Pascal

```
program example(input, output);

var
  ntests, test : integer;
  name : string[100];

begin
  readln(ntests);

for test := 1 to ntests do
  begin
      readln(name);
      writeln('Hello', name, '!');
  end;
end.
```

Solución en Java

```
import java.io.*;

class Main
{
  public static BufferedReader in;

  public static void main(String[] args) throws
  IOException
  {
     in = new BufferedReader(new
        InputStreamReader(System.in));

     int nTests = Integer.parseInt(in.readLine());

     for (int test=1; test <= nTests; test++) {
         String name = in.readLine();
         System.out.println("Hello "+name+"!");
     }
}</pre>
```

Y finalmente, el código para Haskell

Solución en Haskell

```
import Prelude

main :: IO ()
main = do input <- getContents
putStr.unlines.solveAll.tail.lines $ input

solveAll :: [String] -> [String]
solveAll = map (\x -> "Hello " ++ x ++ "!")
```