UNIVERSIDAD
COMPLUTENSE
MADRID

Proyecto de Innovación

Convocatoria 2016/2017

Proyecto nº 35

Implementación de un sistema para el aprendizaje de lenguajes de programación mediante tutoriales interactivos

Enrique Martín Martín

Facultad de Informática

Dpto. Sistemas Informáticos y Computación

Índice general

1. Objetivos propuestos en la presentación del proyecto

El aprendizaje de lenguajes de programación requiere una gran carga práctica que es inviable tratar de manera íntegra en el aula, a pesar de contar con horas dedicadas a sesiones prácticas. Además, las primeras tomas de contacto con la programación para alumnos sin ningún conocimiento, al contener una alta cantidad de conceptos y técnicas nuevas, puede generar en ellos una sensación abrumadora que los desmotiva y hace más probable que abandonen la asignatura. Como profesores de informática nos hemos enfrentado a estas situaciones tanto en cursos iniciales como en el resto de niveles. Disponer de un sistema de tutoriales interactivos para apoyar nuestra docencia resolvería los siguientes problemas:

- Escasez de tiempo para la práctica guiada: las asignaturas de introducción a la programación tienen sus horas divididas entre teoría y práctica. Aunque este reparto trata de dar más peso a la práctica, la naturaleza de esta disciplina hace que nunca sea suficiente. Disponer de una herramienta de tutoriales interactivos permitirá que los alumnos practiquen de manera guiada aspectos concretos de la programación y se puedan dedicar las horas lectivas a los puntos más complicados o que se beneficien del ambiente participativo del aula.

- Insuficiente retroalimentación: La retroalimentación es un aspecto imprescindible para que los alumnos aprendan a programar. Lamentablemente, las horas de clases prácticas no son suficientes y el tamaño de los grupos hace inabordable una retroalimentación personal y detallada. Gracias a los tutoriales interactivos los alumnos podrían realizar pequeños ejercicios y recibir pistas y retroalimentación inmediata, dejando las complicaciones más grandes para tratar personalmente en clase.

- Alumnos poco motivados: En asignaturas de programación, sobre todo en enseñanzas distintas de Informática, la curva de aprendizaje suele ser alta debido a la gran diferencia entre esta materia y el resto. Esto hace que algunos alumnos se desmotiven y abandonen si no asimilan completamente los conceptos principales. Los tutoriales interactivos proporcionarían un refuerzo positivo y permitirían actuaciones que involucren y motiven a los alumnos a aprender. Entre estas actuaciones destaca la ludificación, ya que los tutoriales se dividen en pequeños ejercicios que deben superarse para terminarlos, recibiendo un “premio” y permitiendo incluso crear clasificaciones entre distintos alumnos.

- Aprendizaje en otros entornos: Los tutoriales interactivos abren la posibilidad de aplicarlos en entornos diferentes a la docencia universitaria. Por ejemplo los tutoriales introductorios podrían ser puestos a disposición pública, de manera abierta, de tal manera que cualquier

persona interesada pueda aprender de manera autónoma. Estos tutoriales introductorios también podrían ser utilizados en institutos para reforzar las asignaturas de Informática que últimamente han visto aumentada su importancia. Como los tutoriales también pueden centrarse en aspectos muy concretos o bibliotecas específicas (p.ej. minería de datos, representación de resultados, big data, etc.) también podrían servir como herramientas de formación continua en empresas u organizaciones interesadas en reciclar y formar a sus empleados.

El **objetivo principal de este proyecto es desarrollar un sistema estable para el aprendizaje de lenguajes de programación basado en tutoriales interactivos que pueda ser aplicado en la docencia del curso 2017/18**. Para su consecución y planificación hemos dividido este objetivo en otros más pequeños:

O1) Implementación de un sistema de tutoriales interactivos.

O2) Testeo de la herramienta para detectar bugs y corregirlos.

O3) Elaboración de temas de prueba en al menos un lenguaje de programación.

O4) Elaboración de manuales de uso tanto para profesores como para alumnos.

O5) Difusión de la herramienta entre profesores.

O6) Recepción de la retroalimentación del profesorado sobre la herramienta.

2. Objetivos alcanzados

Podemos asegurar que el objetivo principal de este proyecto ha sido alcanzado, ya que hemos desarrollado un sistema estable para el aprendizaje de lenguajes de programación basado en tutoriales interactivos. Dicho sistema se puede encontrar en https://github.com/emartinm/TutorialesInteractivos. El sistema ha sido probado y depurado en distintos sistemas, por lo que consideramos que está lo suficientemente maduro como para ser aplicado en la docencia del curso 2017/18.

Todos los sub-objetivos propuestos en la solicitud han sido alcanzados, tal y como detallamos a continuación:

O1) **Implementación de un sistema de tutoriales interactivos**
Partiendo del prototipo realizado por Rafael Caturla y Carlos Congosto en su Trabajo Fin de Grado (ver https://github.com/Kherdu/TFG y http://eprints.ucm.es/38408/), hemos realizado una implementación de la herramienta que ha pasado por 5 versiones, tal y como se detalla en el Anexo 6.1. Entre las características más interesantes cabe destacar la posibilidad de recordar el progreso del usuario, el soporte para varios lenguajes de programación (C#, C++, Python y Java), la inclusión de preguntas con varios huecos de código y el soporte para internacionalización y fórmulas matemáticas usando notación LaTeX.

O2) **Testeo de la herramienta para detectar bugs y corregirlos**
A lo largo de las diferentes versiones de la herramienta los integrantes del proyecto realizaron pruebas en distintas plataformas (Windows, Mac y Linux). Fruto de estas pruebas se detectaron 6 defectos, todos los cuales fueron corregidos, y se presentaron 18 propuestas de mejora, de las cuales 15 fueron incorporadas en el sistema. Los informes completos se pueden encontrar en el Anexo 6.2.

O3) **Elaboración de temas de prueba en al menos un lenguaje de programación**
A lo largo de este proyecto se han desarrollado 3 temas de prueba:

- Para Python, un tema de ejemplo que muestra las características de la herramienta y otro tema para el aprendizaje de bucles.
- Para C#, un tema con ejercicios sencillos para practicar la construcción `while`, en concreto la inicialización de variables de control del bucle, condiciones de parada y actualización de variables de control.

Los temas y sus programas correctores se pueden encontrar en https://github.com/emartinm/TutorialesInteractivos/tree/master/temas y en el Anexo 6.3.

O4) **Elaboración de manuales de uso tanto para profesores como para alumnos**
Se han desarrollado ambos manuales, que están disponibles en el repositorio de la herramienta:

- Manual de usuario: donde se explica el funcionamiento del sistema desde el punto de vista del alumno, centrándose en la navegación y los distintos elementos de la interfaz gráfica.
- Manual del profesor: donde se describe con detalle el formato de los temas, los distintos elementos que componen las lecciones y cómo se pueden definir nuevos temas.

Ambos manuales se pueden descargar del repositorio (https://github.com/emartinm/TutorialesInteractivos/blob/master/doc) y también han sido incluidos en el Anexo 6.4.

O5) **Difusión de la herramienta entre profesores**
Para la difusión de la herramienta entre los profesores potencialmente interesados se realizaron 3 actuaciones:

- Ponencia titulada *"INNOVA-Docentia. Aprendizaje de lenguajes de programación mediante tutoriales interactivos"* en la *IV Jornada de Innovación Docente* de la Fac. de Informática (https://informatica.ucm.es/iv-jornada-de-innovacion-docente). Ponente: Enrique Martín.
- Participación en la Jornada *Las TIC en la Enseñanza: Experiencias en la UCM*. La comunicación fue publicada en las actas (páginas 69–73, http://eprints.ucm.es/42177/, ver Anexo 6.5) y también fue aceptada para su presentación.
  Autores: Enrique Martín y Adrián Riesco. Ponente: Adrián Riesco.
- Realización del taller *Tutoriales Interactivos en el Aprendizaje de la Programación* celebrado el día 30 de mayo de 2017 en la Fac. de Informática, con una duración de 2 horas. Al taller se inscribieron 18 personas y finalmente asistieron 15. Las transparencias utilizadas en dicho taller se encuentran en https://github.com/emartinm/TutorialesInteractivos/blob/master/doc/Taller_FDI_2017.pdf y también se adjuntan en el Anexo 6.6. Durante el taller recibimos una serie de comentarios interesantes, que están recogidos en el Anexo 6.7.
  Ponentes: Enrique Martín y Adrián Riesco.

O6) **Recepción de la retroalimentación del profesorado sobre la herramienta**
Tras la realización del taller *Tutoriales Interactivos en el Aprendizaje de la Programación* se realizó una encuesta *online* a los participantes para que pudiesen transmitirnos su opinión de la herramienta. De los 15 participantes recibimos 6 envíos, siendo la mayoría muy positivos. La encuesta concreta se puede encontrar en el Anexo 6.8 y los resultados agregados se adjuntan en el Anexo 6.9.

3. Metodología empleada en el proyecto

Nos referimos a cada miembro con sus iniciales. Rafael Caturla (RC), Carlos Congosto (CC), Carlos Gregorio (CG), Francisco López (FL), Enrique Martín (EM), Adrián Riesco (AR), Jaime Sánchez (JS) y Salvador Tamarit (ST).

Para la consecución del proyecto se identificaron 6 tareas:

T1) Estudio y prueba del prototipo desarrollado en el TFG 2015/16 titulado «Sistema de creación de tutoriales interactivos» dirigido por EM y Manuel Montegro y realizado por RC y CC. **Objetivos**: O1. **Duración**: 09/16. **Responsable**: EM. **Miembros**: RC CC CG FL AR JS ST

T2) Codificación y testing de la herramienta.
Objetivos: O1 O2. **Duración**: 10/16–02/17. **Responsable**: EM. **Miembros**: RC CC CG FL AR JS ST

T3) Elaboración de un tema de prueba por cada lenguaje soportado.
Objetivos: O3. **Duración**: 03/17. **Responsable**: JS. **Miembros**: RC CG EM

T4) Elaboración de manuales.
Objetivos: O4. **Duración**: 10/16. **Responsable**: FL. **Miembros**: CC EM AR ST

T5) Difusión de la herramienta a profesores.
Objetivos: O5. **Duración**: 04/17–05/17. **Responsable**: AR. **Miembros**: RC CC CG FL EM JS ST

T6) Retroalimentación de los profesores.
Objetivos: O6. **Duración**: 04/17–05/17. **Responsable**: ST. **Miembros**: RC CC CG FL EM AR JS

Coordinación

Durante la duración del proyecto se organizaron reuniones (presenciales u *online*) para coordinar y revisar el cumplimiento de los objetivos. Además de las reuniones presenciales se utilizó un repositorio compartido y público para el código[1] y otro privado para los demás documentos (Google Drive).

[1] https://github.com/emartinm/TutorialesInteractivos

4. Recursos humanos

El equipo de este proyecto estuvo formado por 6 profesores y 2 alumnos:

- Enrique Martín (responsable del proyecto) ha participado en 4 PIMCD desde 2007 y ha dirigido uno (el 2015/109 en que se basa esta solicitud), todos ellos relacionados con herramientas automáticas para el aprendizaje de la programación utilizando distintos enfoques.
- Carlos Gregorio ha participado en 12 PIMCDs de la UCM, 3 como director, todos ellos entorno al aprendizaje de la programación. Con más de 20 años de experiencia docente, ha impartido una gran variedad de asignaturas en diversos planes de estudios y centros de la UCM. Gestiona desde hace años un servidor Moodle para las asignaturas del DSIC en el que va incorporando algunos de los resultados de los PIMCD.
- Francisco López, actualmente director del DSIC, tiene más de 30 años de experiencia docente en la Universidad, a lo largo de los cuales ha impartido un gran número de asignaturas relativas a programación, tanto en el nivel de grado como de posgrado. Ha participado en diversas actividades de innovación docente, incluyendo un PIMCD de la UCM.
- Adrián Riesco ha participado en 3 PIMCD desde 2011 (incluyendo el 2015/109) y dirigido dos de ellos. El objetivo de todos estos proyectos siempre ha sido mejorar la enseñanza de las asignaturas de programación, explorando nuevas ideas que pueden hacer más llevadera esta tarea. Durante los últimos 5 años ha impartido asignaturas de introducción a la programación en grados de distintas Facultades.
- Jaime Sánchez ha participado en 3 PIMCD, incluyendo el 2015/109. En todos ellos se han abordado metodologías y materiales encaminados a facilitar el aprendizaje de la programación en distintos aspectos y aproximaciones. Ha impartido numerosos cursos de introducción a la programación, tanto en los grados de la Facultad de Informática, como en Matemáticas o Estadística.
- Salvador Tamarit ha impartido docencia, muchas veces en los primeros semestres de la carrera, en diferentes asignaturas de aprendizaje de la programación. Posee experiencia en otros PIMCD y en estudios a nivel docente. También posee una gran experiencia desarrollando software en diferentes lenguajes e interfaces para estas.
- Rafael Caturla es alumno egresado y Carlos Congosto es alumno de Máster. Como alumnos, conocen los problemas que pueden surgir a la hora de aprender un nuevo lenguaje de programación y la necesidad de una ayuda para facilitar dicho aprendizaje. Su participación en este proyecto ha sido muy importante ya que ambos desarrollaron del prototipo que servirá como base, prototipo que constituyó su TFG.

5. Desarrollo de las actividades

Las actividades correspondientes a este proyecto se realizaron sin desviaciones significativas sobre la planificación, y las desviaciones que se sufrieron no tuvieron impacto alguno sobre la consecución de los objetivos. A continuación detallamos los puntos más destacados de cada tarea.

T1) **Estudio y prueba del prototipo desarrollado en el TFG 2015/16.**
Durante septiembre de 2016 los distintos miembros del proyecto se dedicaron a realizar pruebas con el prototipo de la herramienta desarrollado por Rafael Caturla y Carlos Congosto. El resultado de estas pruebas fueron varios informes de defecto y mejoras, incluidos en el Anexo 6.2.

T2) **Codificación y testing de la herramienta.**
Desde octubre de 2016 hasta febrero/marzo de 2017 se realizaron las principales modificaciones sobre la herramienta. La planificación contemplaba 3 iteraciones de desarrollo en las cuales se iban añadiendo y probando nuevas funcionalidades. El alcance original de esta tarea incluía la consideración de preguntas de tipo *gramática* donde se comprobaba que el código introducido por el alumno tenía una determinada estructura gramatical. Sin embargo, debido a la complejidad técnica de este tipo de preguntas y su (esperable) escaso beneficio, se decidió excluirlas del sistema final. También se aumentó el número de iteraciones a realizar hasta 5 (ver detalles completos en Anexo 6.1):

1. **Corrección de errores y mejoras sobre el prototipo inicial (v1.1.0, 14/01/2017)**. En esta iteración se corrigieron todos los defectos detectados y se incluyeron nuevas funcionalidades menores (memoria para recordar las respuestas del alumno en cada fragmento de la lección, uso de un sistema de integración continua —Travis—, soporte para doble clic al seleccionar elementos de la listas, uso del nombre del tema en lugar del nombre de fichero).
2. **Superación de lecciones y temas (v1.1.0, 01/02/2017)**. En esta iteración se incluyó soporte para indicar el progreso dentro de cada lección y cada tema.
3. **Soporte para varios lenguajes de programación (v1.2.0, 06/02/2017)**. En esta iteración se amplió el número de lenguajes soportados, que originalmente incluía a Python. Se añadió soporte para C++, C# y Java. Este soporte se implementó de manera modular, para que en el futuro se pueda soportar de manera sencilla cualquier lenguaje interpretado, compilado o mezcla de ambos. Se dio prioridad a las herramientas gratuitas, pero también se trató de soportar herramientas privativas de amplio uso en las asignaturas de programación como Visual Studio.

4. **Soporte para preguntas de código con varios huecos (v1.3.0, 26/02/2017)**. La principal incorporación en esta iteración fue el soporte para preguntas de tipo código con varios huecos a rellenar por el alumno. Originalmente se había considerado únicamente un hueco por pregunta.
5. **Ampliación de características y pequeñas correcciones (v1.3.5, 30/05/2017)**. Esta iteración se realizó para pulir la herramienta de cara a su presentación en el taller *Tutoriales Interactivos en el Aprendizaje de la Programación*. En esta iteración se añadió soporte soporte para fórmulas LaTeX usando MathJax, soporte para enlaces en navegador externo, soporte multi-idioma; entre otros.

Los informes completos sobre defectos detectados y mejoras propuestas de pueden encontrar en el Anexo 6.2.

T3) **Elaboración de un tema de prueba por cada lenguaje soportado.**
La tarea de elaboración de temas de prueba estaba planificada para marzo de 2017. Dado que no generaba un impacto negativo en el resto de tareas, se decidió asignarle más tiempo (hasta mayo/junio) y así mejorar la calidad de los temas generados. El resultado han sido 3 temas de prueba:

- Tema de ejemplo que muestra de manera simplificada las características de la herramienta, centrada en el lenguaje Python. Este tema sirve para que un profesor se familiarice con el sistema y conozca rápidamente sus capacidades.
- Tema introductorio a bucles en Python.
- Tema introductorio a bucles en C#, donde se proponen ejercicios sencillos para practicar la construcción `while`, en concreto la inicialización de variables de control del bucle, condiciones de parada y actualización de variables de control.

Los temas y sus programas correctores se pueden encontrar en https://github.com/emartinm/TutorialesInteractivos/tree/master/temas y en el Anexo 6.3.

T4) **Elaboración de manuales.**
La tarea de realización de manuales de uso, tanto para alumnos como para profesores, estaba planificada para el mes de octubre de 2016. La fecha de generación de estos manuales se retrasó ligeramente, aunque este hecho no afectó al resto de tareas: el manual del profesor estuvo realizado antes de marzo de 2017, fecha en la que sería utilizado pro la tarea T3; y el manual de usuario estuvo realizado en mayo de 2017, fecha en la que era necesitado por la tarea T5. El resultado de esta tarea fueron 2 manuales en formato PDF (ver Anexo 6.4):

- Manual de usuario: donde se explica el funcionamiento del sistema desde el punto de vista del alumno, centrándose en la navegación y los distintos elementos de la interfaz gráfica. https://github.com/emartinm/TutorialesInteractivos/blob/master/doc/Manual_usuario.pdf.
- Manual del profesor: donde se describe con detalle el formato de los temas, los distintos elementos que componene las lecciones y cómo se pueden definir nuevos temas. https://github.com/emartinm/TutorialesInteractivos/blob/master/doc/Manual_crear_lecciones.pdf.

T5) **Difusión de la herramienta a profesores.**
Esta tarea estaba planificada para los meses de abril y mayo de 2017. Nuestra idea original era únicamente organizar un taller en la Fac. de Informática para presentar la herramienta terminada al profesorado interesado. Sin embargo, a lo largo del curso académico aparecieron otras dos vías de difusión que decidimos aprovechar, a pesar de que la herramienta no estaba madura del todo:

- *IV Jornada de Innovación Docente* de la Fac. de Informática (https://informatica.ucm.es/iv-jornada-de-innovacion-docente), celebrada el martes 13 de diciembre, entre las 10:00 y las 14:00 en la Fac. de Informática. Decidimos presentar una ponencia titulada *"INNOVA-Docentia. Aprendizaje de lenguajes de programación mediante tutoriales interactivos"* para dar a conocer el prototipo de la herramienta y el objetivo final que queríamos obtener.
- Jornada *Las TIC en la Enseñanza: Experiencias en la UCM*, celebrada el 29 de marzo de 2017. Al recibir información sobre esta Jornada decidimos que era un medio idóneo para dar a conocer nuestra herramienta a un grupo de profesores más amplio que el que encontraríamos en nuestra Facultad. Para ello realizamos la comunicación titulada *Sistema para el aprendizaje de la programación mediante tutoriales interactivos* (ver Anexo 6.5 y páginas 69–73 de http://eprints.ucm.es/42177/) además de realizar una ponencia en las propias Jornadas.

Además de estos medios adicionales, organizamos el taller *Tutoriales Interactivos en el Aprendizaje de la Programación*, que se celebró el día 30 de mayo de 2017 en la Fac. de Informática, con una duración de 2 horas. El taller tuvo una asistencia de 15 personas. Las transparencias utilizadas en dicho taller se encuentran en https://github.com/emartinm/TutorialesInteractivos/blob/master/doc/Taller_FDI_2017.pdf y también se adjuntan en el Anexo 6.6. Los principales comentarios recibidos durante el taller están recogidos en el Anexo 6.7.

T6) **Retroalimentación de los profesores.**
Finalmente, en mayo de 2017 recogimos retroalimentación de los profesor asistentes al taller *Tutoriales Interactivos en el Aprendizaje de la Programación* mediante un formulario *online*. Las reacciones a la herramienta han sido positivas. El punto menos positivo es la potencial complejidad en la instalación de la herramienta, aunque esperamos que este punto se subsane con las guías que hemos desarrollado durante el proyecto y, de ser necesario, con atención personalizada para aquellos interesados. Dado que la instalación es un problema puntual, esperamos que no suponga un impedimento para el uso de los tutoriales. Por lo demás, creemos que los resultados son positivos y que además hemos obtenido una retroalimentación interesante en las preguntas abiertas que nos permitirá mejorar la herramienta en el futuro. Un aspecto ligeramente negativo es que la participación en las encuestas fue baja. Sin embargo, es interesante ver que se encuentran representados tanto participantes como estudiantes de doctorado, lo que nos permite estudiar el interés en ambos casos.

La encuesta diseñada para recabar retroalimetnación se puede encontrar en el Anexo 6.8, y los resultados agregados en el Anexo 6.9.

6. Anexos

6.1. Cambios en la herramienta

30/05/2017: v1.3.5

Ampliación de características

- Soporte para fórmulas LaTeX usando MathJax.
- Soporte para enlaces en navegador externo.
- Soporte para múltiples idiomas.
- Lanzadores directos para Windows, Mac y Linux.
- Opción para lanzar la herramienta eliminando cualquier configuración previa.
- Generación de un fichero *log* con información de ejecución para depuración.
- Corrección de errores.

26/02/2017: v1.3.0

Soporte para preguntas de código con varios huecos

- Las preguntas de tipo código admiten varios huecos para rellenar.
- Mejoradas las ventanas para seleccionar archivos y carpetas.
- Corrección de errores.

06/02/2017: v1.2.0

Soporte para varios lenguajes de programación

- Además de lenguajes interpretados como Python, ahora se soportan lenguajes compilados (C++) y compilados y ejecutados (Java, C#). El diseño de la herramienta facilita la incorporación de nuevos lenguajes (Erlang, Haskell, Go...) y distintos compiladores/intérpretes para soportar. distintos sistemas operativos.
- Cambios en el estilo de las explicaciones.
- Corrección de errores.

01/02/2017: v1.1.0

Superación de lecciones y temas

- Las lecciones se marcan como superadas al llegar a la pantalla final.
- Los temas se marcan como superados al completar todas sus lecciones.
- Simplificación del código: rediseño, eliminación de métodos no necesarios, eliminación de comentarios desactualizados.
- Corrección de errores.

14/01/2017: v1.0.0

Corrección de errores y mejoras sobre la versión inicial en https://github.com/Kherdu/TFG

- Corregido el problema al acceder a los ficheros en Linux.
- Corregido el problema con el tamaño inicial de la ventana en Linux.
- Arreglada visualización de contenidos HMTL en Windows.
- Sincronizada la barra de progreso con la posición actual dentro de la lección.
- Arreglado el error *IndexOutOfBounds* al terminar una lección, ahora muestra una pantalla de enhorabuena.
- Incorporada memoria para recordar las respuestas del alumno en cada fragmento de la lección, para que pueda navegar sin perder sus respuestas.
- Configurado un sistema de integración continua: Travis.
- Soporte para doble clic al seleccionar elementos de la listas.
- Uso del nombre del tema en lugar del nombre de fichero al elegir temas.
- Añadido un icono de aplicación provisional.
- Reorganizado el inicio de la aplicación y las ventanas de configuración del sistema.
- Mejorado el campo de texto en el que introducir código.

6.2. Informes de defectos y mejoras

Informe de defecto

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de defecto:

1

Autores:

Enrique Martín

Fecha:

06/10/2016

Sistema (S.O., JDK, versiones concretas):

Ubuntu 14.04 LTS 64 bits, JDK 1.8.0u101

Explicación y forma de reproducirlo (incluir capturas si es necesario):

Tras configurar correctamente los lenguajes y los compiladores, no se muestra ninguna lección aunque se disponen de 2: Tema1.yml y Tema2.yml

(A rellenar por el responsable de valorar el defecto)

Estado:

- ✓ **D**etectado
- ✓ **A**ceptado
- ❑ **D**enegado
- ✓ **R**esuelto

Información adicional:

Resuelto en el commit 6f8ce281105d05630161a514929ca83bd09c7cb8.
Solución: uso de las nuevas librerías java.nio para acceder a Paths y Files.

Informe de defecto

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de defecto:

2

Autores:

Enrique Martín

Fecha:

07/10/2016

Sistema (S.O., JDK, versiones concretas):

Ubuntu 14.04 LTS, Java 1.8.0u101 y Java 1.8.0u102

Explicación y forma de reproducirlo (incluir capturas si es necesario):

Al iniciar la aplicación, la primera ventana aparece con un tamaño minúsculo que no permite ver correctamente los distintos botones y campos de texto.

(A rellenar por el responsable de valorar el defecto)

Estado:

- ✓ **D**etectado
- ✓ **A**ceptado
- ❑ **D**enegado
- ✓ **R**esuelto

Información adicional:

Resuelto en la versión 002285c44a7384e3dfd85203c5affcc0913e1431. Se eliminó el código para establecer el tamaño por defecto de la escena.

Informe de defecto

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de defecto:

3

Autores:

Enrique Martín

Fecha:

06/10/2016

Sistema (S.O., JDK, versiones concretas):

Windows 10, Windows 7 y Windows 8 con Java 1.8.0u102

Explicación y forma de reproducirlo (incluir capturas si es necesario):

No se muestra ningún texto en la ventana que muestra las distintas lecciones de un tema, ni tampoco se muestra el texto de cada pregunta. Para las preguntas no se muestra nada independientemente de su tipo: Comentario, Tipo test, Código o Sintaxis.

Sin embargo al probar con la versión de Java 1.8.0u101 estos textos se muestran con normalidad.

(A rellenar por el responsable de valorar el defecto)

Estado:

- ✓ **D**etectado
- ✓ **A**ceptado
- ❑ **D**enegado
- ✓ **R**esuelto

Información adicional:

Resuelto en la versión 002285c44a7384e3dfd85203c5affcc0913e1431.
Se trataba de algún problema con el componente WebView al crear distintos objetos de la clase Scene. Se arregla reutilizando la misma Scene para todas las ventanas. Se ha probado satisfactoriamente en:

- Ubuntu 14.04 LTS con Java 1.8.0u101 y Java 1.8.0u102
- Windows 10 con Java 1.8.0u101 y Java 1.8.0u102
- Windows 8 con Java 1.8.0u101 y Java 1.8.0u102
- Windows 7 con Java 1.8.0u101 y Java 1.8.0u102

Informe de defecto

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de defecto:

4

Autores:

Enrique Martín

Fecha:

10/10/2016

Sistema (S.O., JDK, versiones concretas):

Ubuntu 14.04 LTS con Java 1.8.0u101

Explicación y forma de reproducirlo (incluir capturas si es necesario):

Debajo de los botones que sirven para navegar entre las distintas explicaciones y preguntas de una lección aparece una barra de navegación. Esta barra de navegación no actualiza su ocupación según se va completando la lección, sino que siempre aparece en su posición inicial mostrando el principio de la lección.

(A rellenar por el responsable de valorar el defecto)

Estado:

- ✓ **D**etectado
- ✓ **A**ceptado
- ❑ **D**enegado
- ✓ **R**esuelto

Información adicional:

Resuelto en el commit cdca6314a9ee632a685569737a1191a42adf0d7b. Se usa el método ScrollPane.setHvalue(double) para centrar la barra de botones en el fragmento actual. Se ha eliminado la componente ScrollBar ya que ScrollPane la incluye de manera automática.

Informe de defecto

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de defecto:

5

Autores:

Enrique Martín

Fecha:

11/10/2016

Sistema (S.O., JDK, versiones concretas):

Ubuntu 14.04 LTS con Java 1.8.0u101

Explicación y forma de reproducirlo (incluir capturas si es necesario):

Dentro de una lección, el último elemento no se muestra nunca, sino que se ve el mismo contenido del elemento que se estaba visualizando anteriormente. En la consola aparece la traza de depuración de la excepción:

```
Exception in thread "JavaFX Application Thread"
java.lang.ArrayIndexOutOfBoundsException:
```

(A rellenar por el responsable de valorar el defecto)

Estado:

- ✓ **D**etectado
- ✓ **A**ceptado
- ❑ **D**enegado
- ✓ **R**esuelto

Información adicional:

Arreglado en el commit a3761f92dd380d27816e07902fb8e78708aa1e19. El problema era que el array de booleanos 'visited' de la clase Controller era más pequeño de lo esperado. Ahora se muestra tanto el fragmento inicial con la introducción a la lección como el fragmento final felicitando al alumno.

Informe de defecto

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de defecto:

6

Autores:

Enrique Martín

Fecha:

022/02/2017

Sistema (S.O., JDK, versiones concretas):

Ubuntu 14.04 LTS 64 bits, JDK 1.8.0u101

Explicación y forma de reproducirlo (incluir capturas si es necesario):

La barra que sirve para separar la explicación de la zona de repuestas vuelve a su posición por defecto cada vez que se selecciona un opción (preguntas de test) o se corrige (todas las preguntas). Es bastante lioso.

(A rellenar por el responsable de valorar el defecto)

Estado:

- ✓ **D**etectado
- ✓ **A**ceptado
- ❑ **D**enegado
- ✓ **R**esuelto

Información adicional:

Resuelto en el commit 04f75fc7675fcaf4874272fb085f5260ba5b19e0

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

1

Autores:

Carlos Gregorio, Salvador Tamarit, Enrique Martín

Fecha:

06/10/2016

Explicación de la mejora:

Al volver a una pregunta previamente contestada, la respuesta elegida o el código introducido ha desaparecido. Lo mismo ocurre al comenzar a contestar una pregunta si retrocedes a páginas anteriores a consultar la teoría y vuelves para terminarla.

Sería muy interesante que la herramienta almacene las respuestas ya realizadas y las que están a medias para que el alumno pueda navegar sin problemas.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit 10fd1c658d69398cd88fb43ca1f5a1d5a7a42093. El sistema almacena y muestra la última solución introducida por el alumno, tanto en preguntas de tipo test como de tipo código. Si el alumno no ha realizado ningún cambio desde la corrección, el sistema muestra además el mensaje de correcto/incorrecto y en su caso las pistas.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

2

Autores:

Enrique Martín

Fecha:

08/10/2016

Explicación de la mejora:

Sería interesante incluir un sistema de integración continua en el repositorio GitHub para detectar en tiempo real aquellas versiones que no compilan o que no pasan los tests de unidad.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

El repositorio GitHub se conectó con el sistema de integración continua Travis a partir de la versión 3a7c23cfb35e0e9f99bc85d90b5fcfac702ecb25.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

3

Autores:

Enrque Martín

Fecha:

10/10/2016

Explicación de la mejora:

Para seleccionar un lenguaje y después un tema y una lección, es necesario hacer clic con el ratón en su nombre en la lista para luego desplazarse hasta el botón de continuar y pulsarlo. Sería interesante permitir hacer doble clic un nombre de lenguaje, tema o lección para elegirlo directamente y pasar a la siguiente vista.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en commit 45c4136cfd7f4abf895d6dad9d889124204c96a8.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

4

Autores:

Enrique Martín

Fecha:

10/10/2016

Explicación de la mejora:

Para escoger un tema, el listado muestra directamente los nombres de fichero. Sería deseable que mostrase los títulos de los temas extraídos del fichero YAML, puesto que será bastante más informativo.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit a93e8e6514c72cf3ac3e4a7428f9d5094fdd8f65. Los distintos temas aparecen con el título establecido en su fichero YAML, y a su vez el listado aparece ordenado por el número de tema que se establece en cada fichero YAML.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

5

Autores:

Enrique Martín

Fecha:

10/10/2016

Explicación de la mejora:

Añadir un icono a la aplicación, para que se pueda distinguir con facilidad en la barra de herramientas.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ☐ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Añadido en el commit 9a92d2773f2251c7ce84b8b9db589cdb04cb43a5. Parece funcionar únicamente para Windows.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

6

Autores:

Jaime

Fecha:

11/10/2016

Explicación de la mejora:

Al abrir la aplicación la ventana sale una ventana muy pequeña (se podría ampliar) y pide "Ruta fichero dependencias". No está demasiado bien explicado en la memoria y cuesta arrancar. En general, la interfaz poco intuitiva para poner las rutas, intérpretes y demás, y arrancar el sistema. La documentación del TFG tampoco ayuda mucho en este sentido.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Integrado en el commit 5e5f1ab318a6998ce23f9a41a1491a91809ba348. Cambios principales:

- En lugar de "ruta de dependencias" se llama "directorio de cursos".
- Eliminado el botón "acepta path". Ahora seleccionar un directorio de cursos actualiza directamente el listado de lenguajes disponibles.
- El botón "volver" se llama "cancelar".
- Ahora no es imprescindible configurar todos los lenguajes para poder avanzar de la configuración, pero sí es obligatorio configurar el directorio de cursos.
- Si tratas de acceder a un lenguaje no configurado, aparece una alerta explicando qué tienes que hacer y te redirige directamente a la ventana de configuración.
- Si tratas de abandonar la sección de configuración sin establecer el directorio de cursos, aparece una alerta avisando y la ventana no cambia.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

7

Autores:

Adrián Riesco

Fecha:

13/octubre/2016

Explicación de la mejora:

Las respuestas para algunas preguntas requieren introducir texto, por lo que se presenta un campo con el mensaje “Introducir texto aquí”. Al pulsar sobre el campo sería interesante que dicho mensaje se borrase automáticamente.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

El texto de “Escriba su código aquí” desaparece automáticamente a partir de la versión f4980cecf45b2fce11931a2ee07f809f0b460d7f.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

8

Autores:

Salvador Tamarit

Fecha:

17/01/2017

Explicación de la mejora:

Al acabar una lección y volver al menú de selección de lecciones, no se sabes cuales se han realizado y cuáles no. Creo que ayudaría si se pudiese resaltar las ya superadas de alguna manera.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit 6b30e1bc8d73cef0985351c91c7474c2315cb750. Ahora las lecciones completadas se marcan con un "tick" en el listado de lecciones. De la misma manera, si todas las lecciones de tema se han completado, el tema completo aparece marcado con un "tick". Todo el progreso del alumno se almacena en un fichero `progress.json` en el directorio de temas.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

9

Autores:

Salvador Tamarit

Fecha:

17/01/2017

Explicación de la mejora:

Cuando se vuelve a una lección anteriormente superada las respuestas no se quedan guardadas. Creo que debería de guardarse o por lo menos dar la opción de cargar o no las respuestas anteriores. De esta forma el usuario puede volver a entrar a una lección tanto para recordar ciertos aspectos, así como para hacerla de nuevo partiendo de 0. Ahora mismo solo se puede hacer la segunda opción.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit 6b30e1bc8d73cef0985351c91c7474c2315cb750. Ahora el estado de la lección (fragmentos vistos, respuestas introducidas, etc.) se almacena en un fichero `progress.json` en el directorio de temas. Esta información se carga al visualizar una lección. No se ha implementado la opción de permitir borrar el estado de la lección, aunque siempre se puede borrar el fichero.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

10

Autores:

Salvador Tamarit

Fecha:

17/01/2017

Explicación de la mejora:

Al acabar una lección podría haber un botón que te llevase directamente a la siguiente lección (o tema si es la última lección). Probado en MacOs Sierra.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ✓ **D**esestimada
- ❑ **A**ceptada
- ❑ **I**ncorporada

Información adicional:

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

11

Autores:

Salvador Tamarit

Fecha:

17/01/2017

Explicación de la mejora:

Los botones que permiten cambiar de tema o de lección podrían sustituirse por un texto clicable que indicase el tema y la lección actual, ya que ahora mismo no hay ninguna información al respecto en la ventana. Al clicar en la lección o en el tema se comportara igual que los botones actuales. Al clicar estos botones debería dar la opción de volver, ya que si se les da click sin querer se pierde todo el progreso en la lección. Probado en MacOS Sierra.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado parcialmente en el commit c37aa4cc4ab4fe9f6c71b3eb880b78eee7e5a587. Ahora los distintos fragmentos de la lección muestran el lenguaje, tema y lección actual.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

12

Autores:

Enrique Martín

Fecha:

27/01/2017

Explicación de la mejora:

Soportar lenguajes que requieran compilación en lugar de lenguajes únicamente interpretados. Esto ampliaría la aplicación de la herramienta a lenguajes importante como C, C++, Java o C#.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit 9375e9adf3d80f1c3b595667157eee7c5e1e7ad1. Se ha rediseñado el mecanismo para corregir los ejercicios: ahora se basa en crear un fichero fuente a partir de la plantilla de la corrección, que contiene un hueco que es rellenado con el código del alumno.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

13

Autores:

Carlos Gregorio

Fecha:

07/02/17

Explicación de la mejora:

Complementando al “tick” que sirve para indicar una lección o tema completado, sería interesante incluir un gráfico o número indicando si ya has avanzado por ahí y cuanto.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ☐ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit 1d45e1adaff8c45cb6dde23c4b8e6e913da81b6b. Se usa el elemento gráfico ProgressIndicator de Java FX. La noción de progreso en una lección es el elemento más adelantado que se ha visualizado. En los temas, el progreso total es la media de los progresos.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

14

Autores:

Jaime Sánchez

Fecha:

20/02/2017

Explicación de la mejora:

Mejoraría la usabilidad si en las preguntas de código de varios huecos, dichos huecos aparecieran directamente en la zona de explicación de la pregunta. Ahora mismo aparecen en la zona inferior como una sucesión de campos de texto.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ❑ **A**ceptada
- ❑ **I**ncorporada

Información adicional:

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

15

Autores:

Carlos Gregorio

Fecha:

20/02/2017

Explicación de la mejora:

Mejoraría la usabilidad de la aplicación si se pudiese desplazar la separación entre la zona de texto con la explicación y la zona de escritura de fragmentos de código, ya que en ocasiones es interesante ampliar el tamaño de una u otra.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit 540371404caf4016915498cbb11743833d86eeb9

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

16

Autores:

Carlos Gregorio

Fecha:

20/02/2017

Explicación de la mejora:

Las preguntas de tipo test únicamente admiten respuestas constantes. Sería interesante permitir alguna manera de definir un “esquema” de pregunta que se instanciara a una pregunta concreta generando valores concretos. Es algo similar a lo que hace Moodle.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ❑ **A**ceptada
- ❑ **I**ncorporada

Información adicional:

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

17

Autores:

Jaime Sánchez

Fecha:

20/02/2017

Explicación de la mejora:

En las preguntas de tipo código, sobre todo en las que tienen varios huecos, a veces resulta complicado tener claro cuál es el fragmento de código que se está generando. Sería bastante interesante disponer de una opción para mostrar el código final con los huecos rellenos.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit 9aa26c0032c0ae73c75992c72743247865732169. Se ha añadido un nuevo botón para mostrar un PopUp con el fragmento de código rellenado. Los programas correctores deben contener etiquetas para marcar el inicio y el final del código a mostrar.

Propuesta de mejora

Proyecto INNOVA-Docencia 2016/17 nº 35

Número de propuesta de mejora:

18

Autores:

Enrique Martín

Fecha:

22/02/2017

Explicación de la mejora:

Los enlaces incluidos en las lecciones se abren directamente en la misma ventana de la aplicación. Como no aparecen botones de navegación, resulta muy poco cómodo. Sería mejor capturar todos los enlaces y abrirlos en el navegador por defecto del sistema operativo.

(A rellenar por el responsable de valorar la propuesta)

Estado:

- ✓ **P**resentada
- ❑ **D**esestimada
- ✓ **A**ceptada
- ✓ **I**ncorporada

Información adicional:

Incorporado en el commit ff6e70eb9ff1126507c0afc1615b48668bb1c70a. Únicamente se capturan los enlaces añadidos directamente por el profesor, mientras que los enlaces incluidos dentro de iframes pueden no funcionar.

6.3. Temas elaborados

6.3.1. Python

Listado 6.1: *Tema0.yml*

```
Subject: 0
Title: Tema de prueba
Intro: Tema para mostrar la potencia de la herramienta
Lessons:
 - Title: Explicaciones
   Elements:
    - Elem: Text
      Content: |
          Las explicaciones sirven para mostrar información al alumno.

          Estas esplicaciones se pueden dividir en varios párrafos, y con [
              ↪ Markdown](https://github.com/adam-p/markdown-here/wiki/Markdown-
              ↪ Cheatsheet) se puede formatear de manera sencilla con **negritas
              ↪ **, *itálicas* y `código`.

          También se pueden insertar bloques de código
          ```

          def incrementa(n):
              print(n)
              print(n+1)
              return n + 1
          ```
    - Elem: Text
      Content: |
          También se pueden establecer distintas cabeceras.

          # H1

          ## H2

          ### H3

          #### H4

          ##### H5

          ###### H6
    - Elem: Text
      Content: |
          Markdown soporta listas:

          # Listas numeradas
          1. Elemento 1
          1. Elemento 2

          # Listas no numeradas
          * Elemento 1
          * Elemento 2
          * Elemento 3
```

```
48
49          # Listas anidadas
50          * Elemento 1
51              1. anidado 1
52              1. anidado 2
53              1. anidada 3
54          * Elemento 2
55              * anidado 1
56              * anidado 2
57      - Elem: Text
58        Content: |
59          Markdown soporta la incorporación de **imágenes**:
60
61          **Imagen desde internet**
62
63          ![texto alternativo](https://upload.wikimedia.org/wikipedia/commons
                ↪ /6/63/Wikipedia-logo.png)
64
65          **Imagen desde el directorio de temas (*con ruta relativa al directorio
                ↪ del lenguaje*)**
66
67          ![triangulo](file:///img/triangulo.jpg)
68
69          **Las imágenes pueden ser GIFs animados**
70
71          ![dijkstra](https://upload.wikimedia.org/wikipedia/commons/5/57/
                ↪ Dijkstra_Animation.gif)
72      - Elem: Text
73        Content: |
74          También se pueden incluir enlaces en las explicaciones:
75
76          * [Texto del enlace](https://www.ucm.es)
77          * [http://costa.ls.fi.upm.es](http://costa.ls.fi.upm.es)
78      - Elem: Text
79        Content: |
80          # Uso de notación matemática
81          Para insertar notación matemática usaremos **MathJax** ([https://www.
                ↪ mathjax.org/](https://www.mathjax.org/)), un
82          procesador JavaScript que muestra código LaTeX en una página web. Para
                ↪ cargar MathJax se necesita tener **conexión
83          a internet**.
84
85          Las fórmulas mátemáticas *inline* se deben definir entre los símbolos **
                ↪ @ @** y **@ @**, mientras que las
86          fórmulas matemáticas que producen salto de línea se deben definir entre
                ↪ símbolos **@ @ @** y **@ @ @**.
87
88          *Ejemplo*:
89
90          When @@a \ne 0@@, there are two solutions to @@ax^2 + bx + c = 0@@ and
                ↪ they are
91          @@@x = {-b \pm \sqrt{b^2-4ac} \over 2a}.@@@
92      - Elem: Text
93        Content: |
94          En Markdown se pueden definir tablas de una manera muy sencilla:
95
```

```
          Cabecera|Cabecera|Cabecera
          :-------|:------:|-------:
           1      | 2      | 3
           1      | 2      | 3
    - Elem: Text
      Content: |
          Donde Markdown no llega, se puede insertar código HTML y se mostrará
              ↪ directamente en la apliación.

          Concretamente esto puede servir para incrustar recursos HTML5, como
              ↪ videos de Youtube:

          **Vídeos de Youtube**

          <iframe width="640" height="360" src="https://www.youtube.com/embed/
              ↪ PDpMgx7avzA" frameborder="0" allowfullscreen target="_self"></
              ↪ iframe>
- Title: Preguntas de tipo test
  Elements:
    - Elem: Options
      Content: |
          Existen 2 tipos de preguntas de tipo test:
          * Preguntas con una sola opción
          * Preguntas con varias opciones

          La pregunta que vemos aquí es de una sola opción válida
      Hint: Se pueden incluir pistas en las preguntas.
      Solution: [1]
      Multiple: no
      Options:
       - OPCIÓN CORRECTA
       - Opción inválida
       - Opción inválida
       - Opción inválida
    - Elem: Options
      Content: |
          Las preguntas de varias opciones permiten selección varias respuestas:
      Hint: Esto es una pista
      Multiple: yes
      Solution: [1,2,4]
      Options:
       - 1) OPCIÓN CORRECTA
       - 2) OPCIÓN CORRECTA
       - 3) Opción inválida
       - 4) OPCIÓN CORRECTA
    - Elem: Options
      Content: |
          Nada impide que las preguntas de varias opciones tengan una sola opción
              ↪ válida
      Hint: Esto es una pista
      Multiple: yes
      Solution: [4]
      Options:
       - 1) Opción inválida
       - 2) Opción inválida
       - 3) Opción inválida
```

```
      - 4) OPCIÓN CORRECTA
    - Elem: Options
      Content: |
          Y tampoco hay nada que impida que las preguntas de varias opciones no
              ↪ tengan ninguna opción válida
      Hint: Esto es una pista
      Multiple: yes
      Solution: []
      Options:
       - 1) Opción inválida
       - 2) Opción inválida
       - 3) Opción inválida
       - 4) Opción inválida
- Title: Preguntas de tipo código
  Elements:
   - Elem: Text
     Content: |
         Las preguntas de código permiten solicitar al usuario framentos de có
             ↪ digo que son insertados en un programa
         corrector, que es posteriormente evaluado.

         El programa corrector es un programa con uno o más *huecos* que son *
             ↪ rellenados* con el código del alumno.
         La salida de este programa corrector rellenado debe almacenar su salida
             ↪ en un fichero **JSON** con el resultado
         de la corrección junto con posibles mensajes y pistas para el alumno.
    - Elem: Code
      Content: Escribe una asignación para almacenar un ```4``` en la variable ```
          ↪ i```
      File: correctores/tema0.py
      Prompt: ["Escriba␣aquí␣el␣código"]
      Hint: Se pueden incluir pistas en generales en las preguntas de código.
    - Elem: Code
      Content: |
       Las preguntas de tipo código pueden contener más de un hueco, por ejemplo
           ↪ dos:

       Rellena los huecos del siguiente código para duplicar el valor de `a` y
           ↪ para almacenar en la variable `c` la multiplicación de `a` y `b`:
       ```

       a = 1
       <Hueco 0>
       b = 3
       <Hueco 1>
       ```
      Gaps: 2
      File: correctores/tema0b.py
    - Elem: Code
      Content: |
       El número de huecos no tiene ningún límite, únicamente debe ser un número
           ↪ superior a ```0```.

       Rellena los huecos del siguiente código para duplicar asignar a las
           ↪ variables`a`, `b`, `c` y `d` los valores 1, 2, 3, y 4 respectivamente
           ↪ .
```

```
        ```

        a = <Hueco 0>
        b = <Hueco 1>
        c = <Hueco 2>
        d = <Hueco 3>
        ```
      Gaps: 4
      Prompt: ["Valor de la variable 'a'","Valor de la variable 'b'","Valor de la variable 'c'","Valor de la variable 'd'"]
      Hint: Esto es una pista general.
      File: correctores/tema0c.py
```

Listado 6.2: *tema0.py*

```
# -*- coding: UTF-8 -*-
import sys
import json
# el json tiene que tener 3 campos
# campo 1: boolean isCorrect: true si está bien, false si está mal
# campo 2: string typeError: titulo del fallo para poner el mensaje en el label de java
# campo 3: lista de string Hints: pistas adicionales para mostrar
def pregunta11(filename):

    try:
        @@@CODE@@@
        dicc = {}
        things = locals()
        if 'i' in things:
            if things['i'] == 4:
                dicc = {'isCorrect':True}
                #value1 = True
                #value2 = ''
                #value3 = ['']
            else:
                dicc = {'isCorrect':False, 'typeError':"Valor incorrecto", 'Hints':['lee el enunciado de nuevo','siempre ayuda']}
                #value1 = False
                #value2 = 'Valor incorrecto'
                #value3 = ['lee el enunciado de nuevo']
        else:
            dicc = {'isCorrect':False, 'typeError':"Variable 'i' no asignada", 'Hints':['prueba a leer el enunciado otra vez']}
            #value1 = False
            #value2 = 'Variable no encontrada'
            #value3 = ['prueba a leer el enunciado otra vez']
        #dicc['isCorrect'] = value1
        #dicc['typeError'] = value2
        #dicc['Hints'] = value3
        with open(filename, 'w') as outfile:
            json.dump(dicc, outfile)
        sys.exit(0);
    except Exception as e:
        e.print_exc()
        sys.exit(1)
        #return False
```

```
40
41
42
43 def main():
44   pregunta11(sys.argv[1])
45
46
47 if __name__ == "__main__":
48   main()
```

Listado 6.3: *tema0b.py*

```
1  # -*- coding: UTF-8 -*-
2  import sys
3  import json
4
5  def pregunta11(filename):
6      try:
7          #@@@SNIPPET@@@ #Inicio del fragmento de código para mostrar
8          a = 1
9          @@@CODE@@@
10         b = 3
11         @@@CODE@@@
12         #@@@SNIPPET@@@ #Fin del fragmento de código para mostrar
13         dicc = {}
14         things = locals()
15         if 'c' in things:
16             if things['c'] == 6 and things['a'] == 2:
17                 dicc = {'isCorrect':True}
18                 #value1 = True
19                 #value2 = ''
20                 #value3 = ['']
21             else:
22                 dicc = {'isCorrect':False, 'typeError':"Valor␣incorrecto␣de␣'a'␣o␣
                       ↪ de␣'c'", 'Hints':['lee␣el␣enunciado␣de␣nuevo','siempre␣ayuda
                       ↪ ']}
23                 #value1 = False
24                 #value2 = 'Valor incorrecto'
25                 #value3 = ['lee el enunciado de nuevo']
26         else:
27             dicc = {'isCorrect':False, 'typeError':"Variable␣'c'␣no␣asignada", '
                   ↪ Hints':['prueba␣a␣leer␣el␣enunciado␣otra␣vez']}
28             #value1 = False
29             #value2 = 'Variable no encontrada'
30             #value3 = ['prueba a leer el enunciado otra vez']
31         #dicc['isCorrect'] = value1
32         #dicc['typeError'] = value2
33         #dicc['Hints'] = value3
34         with open(filename, 'w') as outfile:
35             json.dump(dicc, outfile)
36         sys.exit(0);
37     except Exception as e:
38         e.print_exc()
39         sys.exit(1)
40         #return False
41
42
```

```
43
44 def main():
45   pregunta11(sys.argv[1])
46
47
48 if __name__ == "__main__":
49   main()
```

Listado 6.4: *tema0c.py*

```
1  # -*- coding: UTF-8 -*-
2  import sys
3  import json
4
5  def pregunta(filename):
6      try:
7          #@@@SNIPPET@@@
8          a = @@@CODE@@@
9          b = @@@CODE@@@
10         c = @@@CODE@@@
11         d = @@@CODE@@@
12         #@@@SNIPPET@@@
13         dicc = {}
14         things = locals()
15         if 'a' in things and 'b' in things and 'c' in things and 'd' in things:
16             if things['a'] == 1 and things['b'] == 2 and things['c'] == 3 and
                   ↪ things['d'] == 4:
17                 dicc = {'isCorrect':True}
18             else:
19                 dicc = {'isCorrect':False, 'typeError':"Valores␣incorrectos", '
                       ↪ Hints':['lee␣el␣enunciado␣de␣nuevo','siempre␣ayuda']}
20         with open(filename, 'w') as outfile:
21             json.dump(dicc, outfile)
22         sys.exit(0);
23     except Exception as e:
24         e.print_exc()
25         sys.exit(1)
26         #return False
27
28
29
30 def main():
31   pregunta(sys.argv[1])
32
33
34 if __name__ == "__main__":
35   main()
```

Listado 6.5: *Bucles.yml*

```
1 Subject: 0
2 Title: Primeros Bucles
3 Intro: en Python
4 Lessons:
5  - Title: Explicaciones
6    Elements:
7     - Elem: Text
```

```
8       Content: |
9           Para poder describir algoritmos que resuelvan problemas generales,
                ↪ tenemos que saber expresar en nuestro lenguaje de programación que
                ↪  hay ciertos pasos a seguir que se repetirán tantas veces como sea
                ↪  preciso.
10
11          Python tiene esencialmente dos formas diferentes de describir con
                ↪ precisión estos procesos iterativos: los bucles ‘for‘ y los bucles
                ↪  ‘while‘.
12
13          Trabajando este tema veremos cómo escribir bucles sencillos y, lo que es
                ↪  más importante, entenderemos cuál es la ejecución de los mismos.
14  - Title: Bucles for
15    Elements:
16     - Elem: Text
17       Content: |
18           ## Ejemplo básico
19           Hay muchos casos en los que se puede utilizar un bucle ‘for‘, quizás la
                ↪ más sencilla es para recorrer una lista:
20           ‘‘‘
21
22           l = [1,2,3,4,5]
23           for x in l:
24               print(3*x)
25           ‘‘‘
26           Este código muestra por pantalla los siguientes valores:
27           ‘‘‘
28
29           3
30           6
31           9
32           12
33           15
34           ‘‘‘
35           La variable ‘x‘ va tomando, de uno en uno, los valores de la lista ‘l‘.
                ↪ Con cada valor, ejecuta las instrucciones que se encuentran dentro
                ↪  del bucle, en este caso únicamente ‘print(x*3)‘, aunque podremos
                ↪ hacer cosas mucho más sofisticadas...
36     - Elem: Text
37       Content: |
38           ## Sumar elementos de una lista
39           Podemos utilizar un bucle ‘for‘ para sumar los elementos de una lista.
40           ‘‘‘
41
42           l = [1,2,3,4,5]
43           suma = 0
44           for x in l:
45              suma = suma + x
46           ‘‘‘
47           Este código almacena en la variable ‘suma‘ el resultado de sumar todos
                ↪ los elementos de la lista ‘l‘.
48     - Elem: Code
49       Content: |
50        Rellena los huecos del siguiente código para que la variable ‘suma‘ guarde
             ↪ el valor de la suma de los elementos de la lista ‘l‘.
51        ‘‘‘
```

```
52
53          l = [1,2,3,4,5]
54          suma = 0
55          for x in l:
56              suma = <Hueco 0>
57          ```
58        Gaps: 1
59        File: correctores/bucles1.py
60      - Elem: Code
61        Content: |
62          Rellena los huecos del siguiente código para que la variable `suma` guarde
              ↪ el valor de la suma de los elementos de la lista `moth`.
63          ```
64
65          month = [31,28,31,30,31,30,31,31,30,31,30,31]
66          suma = 0
67          for x in <Hueco 0>:
68              suma = <Hueco 1>
69          ```
70        Gaps: 2
71        File: correctores/bucles2.py
72      - Elem: Code
73        Content: |
74          Rellena los huecos del siguiente código para que la variable `suma` guarde
              ↪ el valor de la suma de los elementos de la lista `moth`.
75          ```
76
77          month = [31,28,31,30,31,30,31,31,30,31,30,31]
78          suma = 0
79          for <Hueco 0> in <Hueco 1>:
80              suma = <Hueco 2>
81          ```
82        Gaps: 3
83        File: correctores/bucles3.py
84  - Title: Bucles while
85    Elements:
86      - Elem: Text
87        Content: |
88            ## Definición
89            Los bucles `while` son útiles cuando no sabemos exáctamente el número de
                ↪  vueltas que tiene que dar nuestro bucle.
90
91            La estructura general de un bucle `while` tiene dos componentes
                ↪ esenciales, además de la palabra reservada `while`, claro!
92
93            Estos componentes son la condición de terminación del bucle y el código
                ↪ que se ejecutará en cada vuelta.
94            ```
95
96            while <<condición>>:
97                <<instrucciones>>
98            ```
99      - Elem: Text
100       Content: |
101           ## Ejemplo básico
```

```
            Veamos un ejemplo. Vamos a escribir un bucle que nos dice cuántos
                ↪ divisores de 2 tiene un determinado número entero.

            ```

            n = 40
            cont = 0
            while n%2 == 0:
                cont = cont + 1
                n = n // 2
            ```
            El código de este bucle almacena en `cont` las veces que el número `n`
                ↪ puede dividirse por 2. En este caso 40 = 5 * 2 * 2 * 2, es decir,
                ↪ `cont` es 3.
    - Elem: Code
      Content: |
       Rellena los huecos del siguiente código para que la variable `cont` guarde
           ↪ el número de veces que `n` es divisible por 2.
           ```

           n = 40
           cont = 0
           while n%2 == 0:
               cont = <Hueco>
               n = n // 2
           ```
      Gaps: 1
      File: correctores/bucles4.py
    - Elem: Code
      Content: |
       Rellena los huecos del siguiente código para que la variable `cont` guarde
           ↪ el número de veces que `n` es divisible por 3.
           ```

           n = 72
           cont = 0
           while n % <Hueco 1> == 0:
               cont = cont + 1
               n = n // 3
           ```
      Gaps: 1
      File: correctores/bucles5.py
```

Listado 6.6: *bucles1.py*

```
# -*- coding: UTF-8 -*-
import sys
import json

def pregunta11(filename):
    try:
        #@@@SNIPPET@@@ #Inicio del fragmento de código para mostrar
        l = [1,2,3,4,5]
        suma = 0
        for x in l:
          suma = @@@CODE@@@
        #@@@SNIPPET@@@ #Fin del fragmento de código para mostrar
```

```
13          dicc = {}
14          things = locals()
15          if 'suma' in things:
16              if things['suma'] == sum(l):
17                  dicc = {'isCorrect':True}
18              else:
19                  dicc = {'isCorrect':False, 'typeError':"Valor␣incorrecto␣de␣la␣
                        ↪ variable␣'suma'", 'Hints':['lee␣el␣enunciado␣de␣nuevo','
                        ↪ siempre␣ayuda']}
20          else:
21              dicc = {'isCorrect':False, 'typeError':"Variable␣'c'␣no␣asignada", '
                    ↪ Hints':['prueba␣a␣leer␣el␣enunciado␣otra␣vez']}
22          with open(filename, 'w') as outfile:
23              json.dump(dicc, outfile)
24          sys.exit(0);
25      except Exception as e:
26          e.print_exc()
27          sys.exit(1)
28          #return False
29
30  def main():
31    pregunta11(sys.argv[1])
32
33
34  if __name__ == "__main__":
35    main()
```

Listado 6.7: *bucles2.py*

```
1   # -*- coding: UTF-8 -*-
2   import sys
3   import json
4
5   def pregunta11(filename):
6       try:
7           #@@@SNIPPET@@@ #Inicio del fragmento de código para mostrar
8           month = [31,28,31,30,31,30,31,31,30,31,30,31]
9           suma = 0
10          for x in @@@CODE@@@:
11            suma = @@@CODE@@@
12          #@@@SNIPPET@@@ #Fin del fragmento de código para mostrar
13          dicc = {}
14          things = locals()
15          if 'suma' in things:
16              if things['suma'] == sum(month):
17                  dicc = {'isCorrect':True}
18              else:
19                  dicc = {'isCorrect':False, 'typeError':"Valor␣incorrecto␣de␣la␣
                        ↪ variable␣'suma'", 'Hints':['lee␣el␣enunciado␣de␣nuevo','
                        ↪ siempre␣ayuda']}
20          else:
21              dicc = {'isCorrect':False, 'typeError':"Variable␣'c'␣no␣asignada", '
                    ↪ Hints':['prueba␣a␣leer␣el␣enunciado␣otra␣vez']}
22          with open(filename, 'w') as outfile:
23              json.dump(dicc, outfile)
24          sys.exit(0);
25      except Exception as e:
```

```
        e.print_exc()
        sys.exit(1)
        #return False

def main():
  pregunta11(sys.argv[1])


if __name__ == "__main__":
  main()
```

Listado 6.8: *bucles3.py*

```
# -*- coding: UTF-8 -*-
import sys
import json

def pregunta11(filename):
    try:
        #@@@SNIPPET@@@ #Inicio del fragmento de código para mostrar
        month = [31,28,31,30,31,30,31,31,30,31,30,31]
        suma = 0
        for @@@CODE@@@ in @@@CODE@@@:
          suma = @@@CODE@@@
        #@@@SNIPPET@@@ #Fin del fragmento de código para mostrar
        dicc = {}
        things = locals()
        if 'suma' in things:
            if things['suma'] == sum(month):
                dicc = {'isCorrect':True}
            else:
                dicc = {'isCorrect':False, 'typeError':"Valor incorrecto de la
                    ↪ variable 'suma'", 'Hints':['lee el enunciado de nuevo','
                    ↪ siempre ayuda']}
        else:
            dicc = {'isCorrect':False, 'typeError':"Variable 'c' no asignada", '
                ↪ Hints':['prueba a leer el enunciado otra vez']}
        with open(filename, 'w') as outfile:
            json.dump(dicc, outfile)
        sys.exit(0);
    except Exception as e:
        e.print_exc()
        sys.exit(1)
        #return False

def main():
  pregunta11(sys.argv[1])


if __name__ == "__main__":
  main()
```

Listado 6.9: *bucles4.py*

```
# -*- coding: UTF-8 -*-
import sys
import json
```

```
4
5  def pregunta11(filename):
6      try:
7          #@@@SNIPPET@@@ #Inicio del fragmento de código para mostrar
8          n = 40
9          cont = 0
10         while n%2 == 0:
11             cont = @@@CODE@@@
12             n = n // 2
13         #@@@SNIPPET@@@ #Fin del fragmento de código para mostrar
14         dicc = {}
15         things = locals()
16         if 'cont' in things:
17             if things['cont'] == 3:
18                 dicc = {'isCorrect':True}
19             else:
20                 dicc = {'isCorrect':False, 'typeError':"Valor incorrecto de la
                       ↪ variable 'cont'", 'Hints':['lee el enunciado de nuevo','
                       ↪ siempre ayuda']}
21         else:
22             dicc = {'isCorrect':False, 'typeError':"Variable 'c' no asignada", '
                   ↪ Hints':['prueba a leer el enunciado otra vez']}
23         with open(filename, 'w') as outfile:
24             json.dump(dicc, outfile)
25         sys.exit(0);
26     except Exception as e:
27         e.print_exc()
28         sys.exit(1)
29         #return False
30
31 def main():
32   pregunta11(sys.argv[1])
33
34
35 if __name__ == "__main__":
36   main()
```

Listado 6.10: *bucles5.py*

```
1  # -*- coding: UTF-8 -*-
2  import sys
3  import json
4
5  def pregunta11(filename):
6      try:
7          #@@@SNIPPET@@@ #Inicio del fragmento de código para mostrar
8          n = 72
9          cont = 0
10         while n%@@@CODE@@@ == 0:
11             cont = cont + 1
12             n = n // 3
13         #@@@SNIPPET@@@ #Fin del fragmento de código para mostrar
14         dicc = {}
15         things = locals()
16         if 'cont' in things:
17             if things['cont'] == 2:
18                 dicc = {'isCorrect':True}
```

```
19              else:
20                  dicc = {'isCorrect':False, 'typeError':"Valor incorrecto de la
                        ↪ variable 'cont'", 'Hints':['lee el enunciado de nuevo','
                        ↪ siempre ayuda']}
21          else:
22              dicc = {'isCorrect':False, 'typeError':"Variable 'c' no asignada", '
                    ↪ Hints':['prueba a leer el enunciado otra vez']}
23          with open(filename, 'w') as outfile:
24              json.dump(dicc, outfile)
25          sys.exit(0);
26      except Exception as e:
27          e.print_exc()
28          sys.exit(1)
29          #return False
30
31  def main():
32    pregunta11(sys.argv[1])
33
34
35  if __name__ == "__main__":
36    main()
```

6.3.2. C#

Listado 6.11: *Bucles.yml*

```
Subject: 1
Title: Bucles
Intro: Bucles while en C#
Lessons:
 - Title: Bucles while en C#
   Elements:
    - Elem: Text
      Content: |
       La estructura de un bucle while es:
       ```

       while (<condición>) {
         <cuerpo del bucle>
       }
       ```

       * <**condición**> es cualquier expresión de tipo booleano

       * <**cuerpo del bucle**> es la secuencia de instrucciones que se repetirá en cada vuelta del bucle. Si hay una sola instrucción pueden omitirse las llaves *{* y *}*

       El cuerpo del bucle se ejecutará *mientras* la *condición* sea cierta (*true*)


    - Elem: Text
      Content: |
       Nuestro primer bucle:
       ```

       while (true) ;
       ```
       Por primera vez, podemos *colgar* un programa!! Este bucle se ejecuta indefinidamente ejecutando... nada!


    - Elem: Text
      Content: |
       Escribir los números de 1 a 10:
       ```

       int i = 1;   // inicialización
       while (i<=10) {   // condición de parada
          Console.WriteLine(i);
          ++i;   //incremento
       }
       ```

       Este es un esquema típico para el *while*. Observemos:
       * la *i* es la *variable de control* del bucle
       * la *condición de parada* se establece en función de esta *i*
```

```
        * dentro del bucle se actualiza el valor de *i* (en este caso se incrementa )

        ?Se puede garantizar la **terminacion** del este bucle?


    - Elem: Code
      Content: |
       Escribe un bucle que escriba en pantalla los numeros del 0 al 9, uno por lí
           nea.

      File: correctores/while2.cs



    - Elem: Code
      Content: |
       Escribe un bucle que calcule en *k* la suma de los numeros del 1 al 10.

      File: correctores/while1.cs



    - Elem: Code
      Content: |
       Escribe un bucle que escriba en pantalla la suma de los numeros pares del
           intervalo [0,100].

      File: correctores/while3.cs



    - Elem: Code
      Content: |
       Escribe un bucle que, dado un número *n* invierta el orden de sus dígitos y
           calcule el numero resultante en *m*.

      File: correctores/while4.cs
```

Listado 6.12: *while1.cs*

```
/*
 * Copyright 2017 Enrique Martin <emartinm@ucm.es>
 *
 * SPDX-License-Identifier: MIT
 */

using System;

public class Corrector {

  static void escribeJSON(String filename, bool isCorrect) {
    String[] lines = new String[5];
    lines [0] = "{";
    if (isCorrect) {
      lines [1] = "␣␣\"isCorrect\":true,";
    } else {
```

```
      lines [1] = "␣␣\"isCorrect\":false,";
    }
    lines[2] = "␣␣\"typeError\":\"Mira␣mira\",";
    lines[3] = "␣␣\"Hints\":[\"Primera␣pista\",␣\"Segunda␣pista\"]";
    lines[4] = "}";
    System.IO.File.WriteAllLines(filename, lines);
  }

  static void Main(String[] args) {
    bool ok = true;


    @@@CODE@@@
    // int k = 0, i = 1;
    // while (i<=10) {
    //   k += i;
    //   ++i;
    // }



    ok = (k == 55);
    //ok = (o == "12345");

    escribeJSON (args [0], ok);
  }
}
```

Listado 6.13: *while2.cs*

```
/* Copyright 2017 Enrique Martín <emartinm@ucm.es>
 *
 * SPDX-License-Identifier: MIT
 */

using System;

public class Corrector {

  static void escribeJSON(String filename, bool isCorrect) {
    String[] lines = new String[5];
    lines [0] = "{";
    if (isCorrect) {
      lines [1] = "␣␣\"isCorrect\":true,";
    } else {
      lines [1] = "␣␣\"isCorrect\":false,";
    }
    lines[2] = "␣␣\"typeError\":\"Mira␣mira\",";
    lines[3] = "␣␣\"Hints\":[\"Primera␣pista\",␣\"Segunda␣pista\"]";
    lines[4] = "}";
    System.IO.File.WriteAllLines(filename, lines);
  }

  static string user_output = "";

  static void Write(object s){ user_output += s; }
  static void WriteLine(object s){ user_output += s + "\n"; }
```

```
28
29    static void Main(String[] args) {
30      bool ok = true;
31
32
33      @@@CODE@@@
34      /*
35      int i = 0;
36      while (i<10) {
37       WriteLine(i);
38        ++i;
39       }
40      */
41
42
43      string my_res ="";
44      int my_i=0;
45      while (my_i<10){
46            my_res += my_i+"\n";
47            ++my_i;
48      }
49
50      ok = (user_output == my_res);
51
52      escribeJSON (args [0], ok);
53    }
54  }
```

Listado 6.14: *while3.cs*

```
1   /* Copyright 2017 Enrique Martín <emartinm@ucm.es>
2    *
3    * SPDX-License-Identifier: MIT
4    */
5
6   using System;
7
8   public class Corrector {
9
10    static void escribeJSON(String filename, bool isCorrect) {
11      String[] lines = new String[5];
12      lines [0] = "{";
13      if (isCorrect) {
14        lines [1] = "␣␣\"isCorrect\":true,";
15      } else {
16        lines [1] = "␣␣\"isCorrect\":false,";
17      }
18      lines[2] = "␣␣\"typeError\":\"Mira␣mira\",";
19      lines[3] = "␣␣\"Hints\":[\"Primera␣pista\",␣\"Segunda␣pista\"]";
20      lines[4] = "}";
21      System.IO.File.WriteAllLines(filename, lines);
22    }
23
24    static string user_output = "";
25
26    static void Write(object s){ user_output += s; }
27    static void WriteLine(object s){ user_output += s + "\n"; }
```

```
28
29    static void Main(String[] args) {
30      bool ok = true;
31
32
33      @@@CODE@@@
34      // int k = 0, i = 1;
35        // while (i<=10) {
36            // k += i;
37            // ++i;
38            //  }
39
40
41      int my_i=2, my_ac=0;
42      while (my_i<=100){
43            my_ac += my_i;
44            my_i = my_i+2;
45      }
46
47
48      ok = (int.Parse(user_output) == 2550);
49
50      escribeJSON (args [0], ok);
51    }
52  }
```

Listado 6.15: *while4.cs*

```
1   /* Copyright 2017 Enrique Martín <emartinm@ucm.es>
2    *
3    * SPDX-License-Identifier: MIT
4    */
5
6   using System;
7
8   public class Corrector {
9
10    static void escribeJSON(String filename, bool isCorrect) {
11      String[] lines = new String[5];
12      lines [0] = "{";
13      if (isCorrect) {
14        lines [1] = "␣␣\"isCorrect\":true,";
15      } else {
16        lines [1] = "␣␣\"isCorrect\":false,";
17      }
18      lines[2] = "␣␣\"typeError\":\"Mira␣mira\",";
19      lines[3] = "␣␣\"Hints\":[\"Primera␣pista\",␣\"Segunda␣pista\"]";
20      lines[4] = "}";
21      System.IO.File.WriteAllLines(filename, lines);
22    }
23
24    static string user_output = "";
25
26    static void Write(object s){ user_output += s; }
27    static void WriteLine(object s){ user_output += s + "\n"; }
28
29
```

```
30    static int n, m;
31    static void invierte(){
32      @@@CODE@@@
33    }
34
35    static void Main(String[] args) {
36      bool ok = true;
37
38
39      n = 12345;
40      m=-1;
41      invierte();
42
43      ok = (m==54321);
44
45
46      n = 4675;
47      invierte();
48
49      ok = ok && (m==5764);
50
51      escribeJSON (args [0], ok);
52    }
53  }
```

6.4. Manuales de uso

Tutoriales Interactivos - Manual de usuario

Salvador Tamarit[a] (stamarit@dsic.upv.es)
Enrique Martín Martín[b] (emartinm@ucm.es)
[a] *Dep. Sistemes Informàtics i Computació*
Universitat Politècnica de València
[b] *Dpto. de Sistemas Informáticos y Computación*
Fac. Informática, Universidad Complutense de Madrid

18 de mayo de 2017

Índice

1. Instalación y ejecución

Para poder ejecutar la herramienta *Tutoriales Interactivos* debéis tener instalada en vuestro sistema la última versión de Java (en el momento de escribir este manual es la *versión 8 update 131)*. Para ello debéis acceder a la página `https://www.java.com/es/download/` y seguir las instrucciones.

Una vez disponemos de la última versión de Java, la instalación de la herramienta *Tutoriales Interactivos* es muy sencilla: únicamente es necesario descargar el programa junto con las lecciones y situarlo en alguna carpeta del sistema (por ejemplo el directorio personal o el escritorio). Si vais a utilizar *Tutoriales Interactivos* es una asignatura, lo más probable es que el profesor os proporcione un fichero comprimido con todos los elementos incorporados. En el caso de no disponer de este fichero comprimido proporcionado por el profesor, podéis descargar la herramienta de su repositorio oficial: `https://github.com/emartinm/TutorialesInteractivos/archive/master.zip`.

Para ejecutar la herramienta, únicamente hay que hacer doble clic en el fichero `TutorialesInteractivos-jar-with-dependencies.jar` situado en la carpeta `target` dentro del directorio principal de la herramienta. En caso de no poder iniciar la herramienta de esta manera consulta la sección 5.1.

2. Configuración de la aplicación

La primera vez que se inicia la herramienta se acceder directamente a la ventana de configuración (ver figura 1). En esta ventana lo primero que hay que hacer es configurar el **directorio de temas** pulsando el botón *Buscar...* que aparece en la parte superior derecha. El directorio de temas normalmente se encuentra dentro del directorio principal de la herramienta con el nombre `temas`, y contiene carpetas por cada uno de los lenguajes disponibles: Python 3.x, C++, Java, etc.

Tras configurar el directorio de temas, nos aparecerán varias entradas en el listado central de la ventana de configuración para establecer los compiladores o intérpretes para cada lenguaje disponible (ver figura 2).

- **Python 3.x**: debes establecer la ruta del **intérprete** de Python versión 3.x. En entornos Linux suele estar en `/usr/bin/python3`, aunque también se puede utilizar el binario instalado por *Anaconda*. En entornos Windows el binario suele llamarse `python.exe` o `python3.exe`, puediendo estar en la instalación estándar de Python o en *Anaconda*.

- **C++**: debes establecer la ruta del compilador de C++. En entornos Linux se utilizará el compilador GNU C++ que suele estar en la ruta `/usr/bin/g++`. En entornos Windows se puede utilizar GNU C++, pero también se soporta el compilador de C++ incluido en *Visual Studio*. En este caso debes establecer la ruta del *script* `vcvars32.bat` o `vcvars64.bat` (dependiendo de si el sistema es de 32 o 64 bits) que establecen las variables de entorno necesarias para compilar con *Visual Studio*. La herramien-

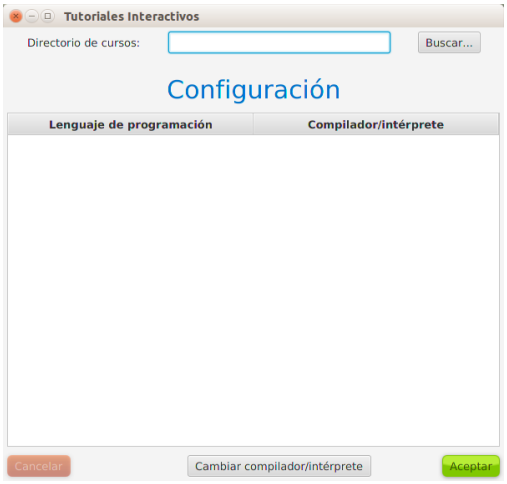


Figura 1: Ventana inicial de configuración

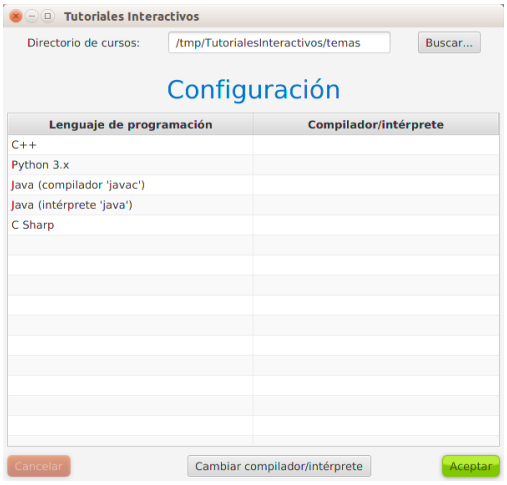


Figura 2: Ventana inicial de configuración con directorio de temas

ta *Tutoriales Interactivos* usará estas variables de entorno para encontrar el compilador de C++ (`cl.exe`) adecuado. La ruta de estos *scripts* puede variar entre distintas versiones de *Visual Studio* o entre versiones de Windows, por lo que recomendamos usar el buscador de ficheros para encontrarlos. Como ejemplos, en un sistema *Windows 10* y para la versión *Visual Studio 2013* estos ficheros están en la carpeta:

`C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin`

mientras que para la versión *Visual Studio 2017* están en:

`C:\Program Files (x86)\Microsoft Visual Studio\2017\Professional\VC\Auxiliary\Build`.

- **C#**: debes establecer la ruta del compilador de C#. En el caso de usar *Mono* es la ruta del fichero `mcs` o `mcs.exe`, que en sistemas Linux suele estar en `/usr/bin/mcs`. En sistemas Windows también se puede utilizar el compilador `csc.exe` de Visual Studio. En este último caso la ruta que hay que configurar es la del *script* `vcvars32.bat` o `vcvars64.bat`, al igual que en el caso de C++.

- **Java**: en este caso hay que configurar dos entradas: la ruta del compilador de `javac` y la ruta del intérprete `java`. En entornos Linux ambos suelen estar accesibles en `/usr/bin`, mientras que en Windows están alojados en `C:\Program Files\Java\jdkX.Y.Z_NNN\bin`. Tened en cuenta que para disponer del compilador de Java es necesario haber instalado la JDK de Java y no únicamente el JRE.

Al finalizar la configuración de la herramienta, la ventana debe tener un aspecto similar al mostrado en la figura 3. Ten en cuenta que **no es obligatorio configurar los compiladores o intérpretes de todos los lenguajes de programación disponibles**, únicamente de aquellos que vayas a utilizar para realizar tutoriales.

3. Navegación general

Tras configurar la herramienta, como se describe en la sección 2, ya estará lista para ser utilizada. La siguiente ventana que se cargará será la de selección de lenguaje, como se muestra en la figura 4. En esta ventana se mostrarán todos los lenguajes disponibles, en este caso cuatro. Primero, deberemos seleccionar el lenguaje que queramos aprender. Tras la selección deberemos pulsar el botón *Comenzar* para acceder a los temas disponibles para el lenguaje seleccionado. Desde la ventana de selección de lenguajes también se puede volver a los ajustes, descritos en la sección 2, mediante el botón *Ajustes*.

La ventana con los temas es como la que se muestra en la figura 5. Aquí deberemos de seleccionar alguno de los temas disponibles, en este caso solo hay uno, y darle al botón *Comenzar*. Podremos también volver a la ventana de selección de lenguajes mediante el botón *Volver*. Al lado de cada tema podemos observar

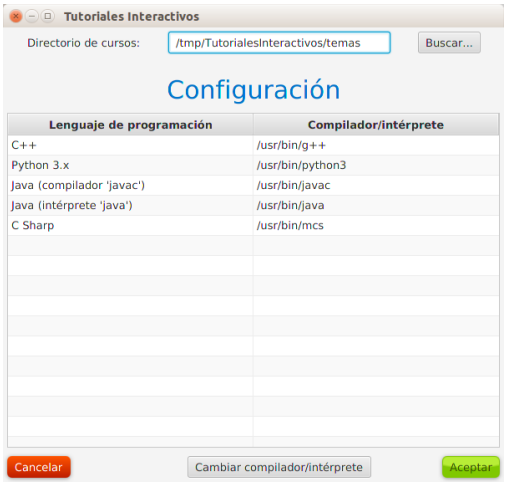


Figura 3: Ventana final de configuración, con las rutas usuales en entornos Linux

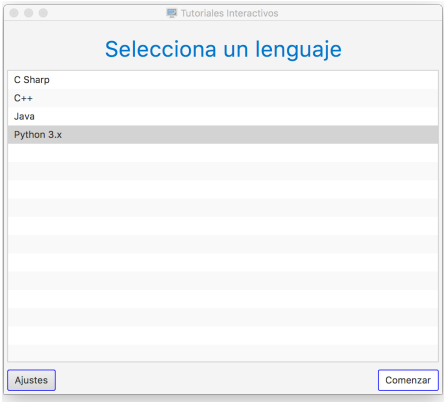


Figura 4: Ventana de selección de lenguaje

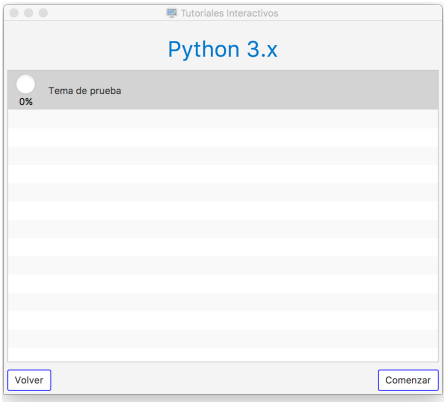


Figura 5: Ventana de selección de temas

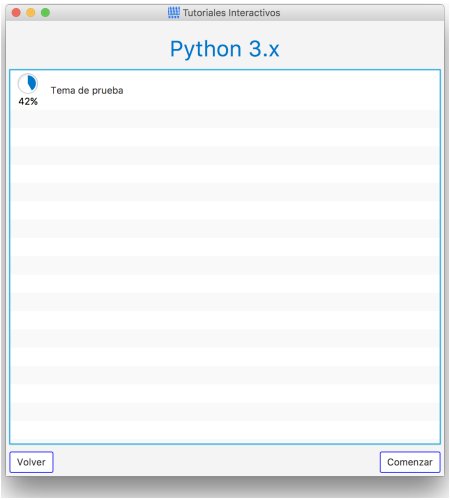


Figura 6: Ventana de selección de temas tras completar parte de un tema

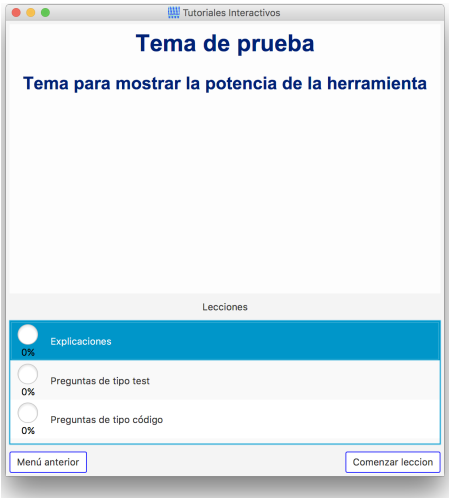


Figura 7: Ventana de selección de lecciones

el porcentaje del tema que se ha completado hasta el momento. Un ejemplo de cómo se mostraría un tema que se ha avanzado parcialmente se puede ver en la figura 6.

4. Lecciones

Una vez seleccionado un tema, como se describe en la sección 3, entraremos en la selección de lecciones. Esta ventana tiene el aspecto que muestra la figura 7. En este caso se dispone de tres lecciones, y para poder acceder a una de ellas la seleccionaremos y pulsaremos sobre el botón *Comenzar lección*. También tendremos acceso a la ventana de selección de temas mediante el botón *Menú anterior*. De manera similar a como ocurría para los temas, aquí también podremos observar el progreso en cada lección mediante el tanto por ciento mostrado en la parte izquierda de cada lección. Podemos ver un ejemplo de como se muestra este progreso en la figura 8.

Una vez dentro de una lección, veremos una ventana como la mostrada en la figura 9. En esta ventana podemos observar varios elementos. Por una parte se nos mostrará en la parte superior toda la información sobre la lección cargada, i.e. el lenguaje, el tema y el nombre de la lección. El siguiente elemento es una descripción del problema, una pregunta o incluso una explicación. En caso de ser uno de los dos primeros, el usuario deberá resolver el ejercicio, y para ello

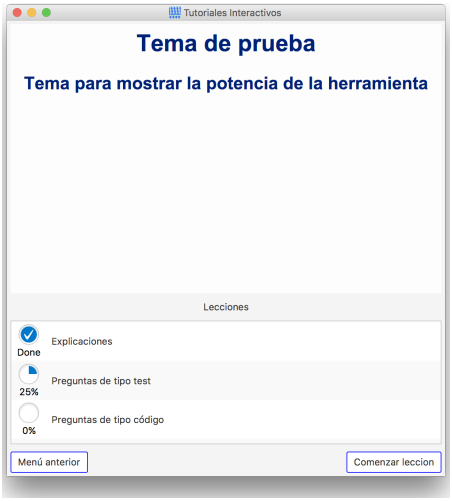


Figura 8: Ventana de selección de lecciones con algunos avances

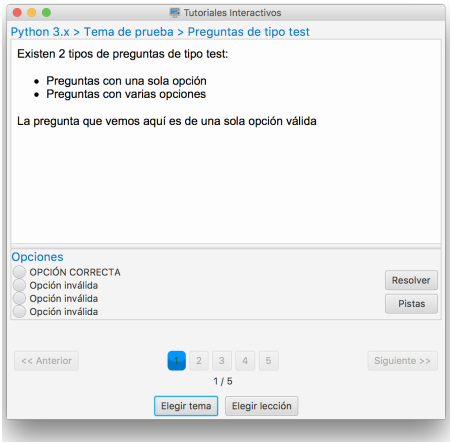


Figura 9: Ventana mostrando un ejercicio de una lección

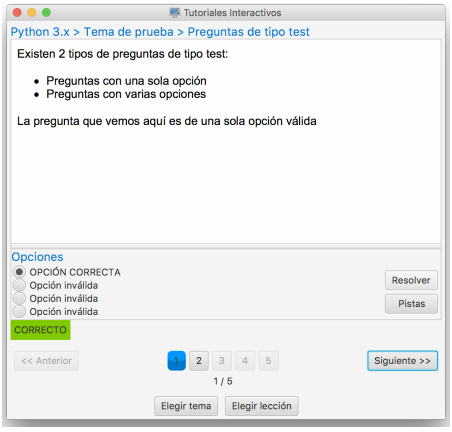


Figura 10: Ventana mostrando un ejercicio correctamente resuelto

tendrá un campo para incluir la respuesta. Este campo incluye un botón *Resolver* para comprobar si la respuesta es correcta y, a veces, un botón *Pistas*, para obtener una serie de pistas que ayuden al alumno a resolver el ejercicio. Finalmente se podrá navegar por las lecciones mediante los botones << *Anterior*, siempre que no sea el primer ejercicio, o el botón *Siguiente* >>, siempre que se haya resuelto el ejercicio que se muestre en el momento y éste no sea el último. Una manera alternativa de navegar por los ejercicios es mediante los números que identifican a cada ejercicio dentro de la lección. En este caso la restricción de haber resuelto los ejercicios también es necesaria para poder ir a ejercicios posteriores al actual. Finalmente, desde cualquier ejercicio podremos cambiar de tema o de lección mediante los botones *Elegir tema* y *Elegir lección* respectivamente.

La introducción de las respuestas depende del tipo de pregunta. Por ejemplo, en el caso del ejercicio mostrado en la figura 9, solo hay una opción posible. Tras seleccionar la que creamos que sea la opción correcta, pulsaremos sobre el botón *Resolver*, para comprobar si es correcta. En caso de ser incorrecta, se nos mostrará *RESPUESTA INCORRECTA* sobre un fondo rojo, y deberemos, por tanto, cambiar la respuesta y volver a pulsar sobre el botón *Resolver*. Cuando la respuesta sea correcta se mostrará *CORRECTO* sobre un fondo verde, como podemos ver en la figura 10, y se habilitará el botón *Siguiente* >>, siempre que no sea el último ejercicio de la lección. De manera similar, se habilitará el botón correspondiente al siguiente ejercicio, en este caso el número 2.

Otros tipos de respuestas que nos podemos encontrar son en las que existen varias opciones seleccionables, como se muestra en la figura 11. En este caso

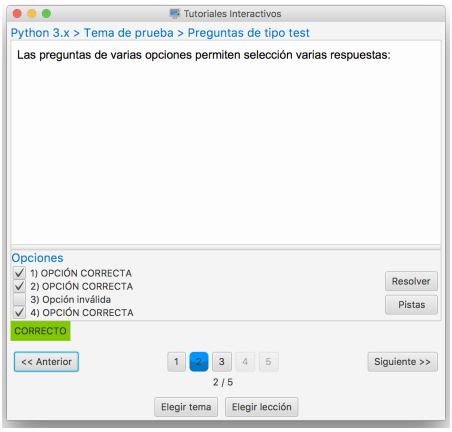


Figura 11: Ventana mostrando un ejercicio con múltiples opciones seleccionables

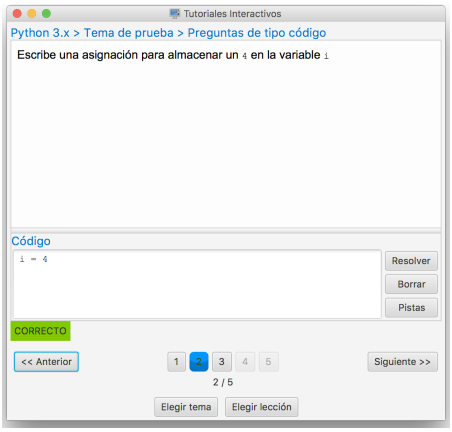


Figura 12: Ventana mostrando un ejercicio de desarrollo

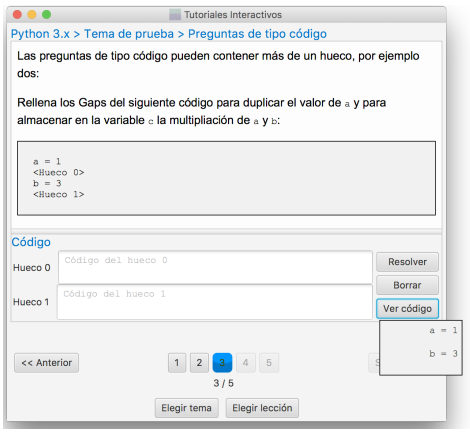


Figura 13: Ventana mostrando un ejercicio de desarrollo con varios huecos a rellenar

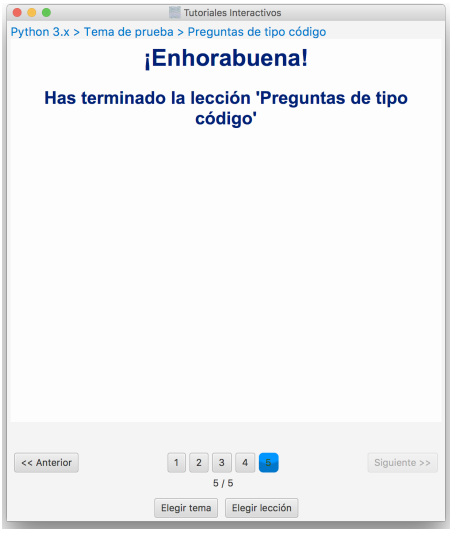


Figura 14: Última ventana correspondiente a una lección

se seleccionarán las opciones que se consideren correctas (puede que no sea ninguna) y se pulsará sobre el botón *Resolver* para conocer el resultado. También existen ejercicios de desarrollo, donde el alumno deberá introducir su respuesta mediante un campo de texto. Un ejemplo de estos ejercicios el que se muestra en la figura 12. En este tipos de ejercicios, además, se incluye el botón *Borrar*, para borrar los campos de texto de la respuesta. Tras introducir la respuesta el alumno deberá pulsar el botón *Resolver* para conocer si es correcta o no. A veces estos ejercicios de desarrollo vienen con varios huecos a completar por el alumno, como, por ejemplo, el mostrado en la figura 13. El modo de proceder es exactamente igual que el anterior. En estos casos, además se incluye el botón *Ver código*, que permite ver el aspecto del código con la respuesta introducida hasta el momento.

Finalmente, al llegar al final de la lección se nos mostrará una ventana como la de la figura 14 desde la cual podremos revisar los ejercicios de la lección, acceder a una nueva mediante el botón *Elegir lección* o cambiar de tema con el botón *Elegir tema*.

5. Solución de problemas

5.1. No puedo iniciar la herramienta

Si al hacer doble clic sobre el fichero `.jar` no se arranca la aplicación, comprueba que dicho fichero es **ejecutable**. Para ello abre el menú contextual con el botón derecho y explora las distintas opciones disponibles para cambiar los permisos del fichero.

Si aún así los problemas persisten, prueba alguna de las siguientes opciones:

- **En sistemas Windows**:
  - Haz doble clic en el fichero `TutorialesInteractivos.bat` que está en el directorio **principal** de la herramienta.
  - Abre un teminal (*Símbolo del sistema*), dirígete al directorio principal de la herramienta y ejecuta alguno de estos comandos:
    ```
    > TutorialesInteractivos.bat
    > java -jar target/TutorialesInteractivos-jar-with-dependencies.jar
    ```

- **En sistemas Linux y Mac**:
  - Haz doble clic en el fichero `TutorialesInteractivos.sh` que está en el directorio **principal** de la herramienta.
  - Abre un teminal, dirígete al directorio principal de la herramienta y ejecuta alguno de estos comandos:
    ```
    $ TutorialesInteractivos.sh
    $ java -jar target/TutorialesInteractivos-jar-with-dependencies.jar
    ```

5.2. La herramienta se inicia con errores

Si la herramienta se inicia con errores o tiene un comportamiento anómalo, la mejor opción es limpiar todas las configuraciones y empezar desde cero. Para ello existe una opción `--reset` que se puede utilizar al lanzar la aplicación. Para ello abre un terminal y, dentro del directorio principal de la aplicación, ejecuta el siguiente comando:

```
java -jar target/TutorialesInteractivos-jar-with-dependencies.jar --reset
```

La herramienta eliminará todos los valores almacenados en ejecuciones previas (directorio de temas, rutas de compiladores/intérpretes y progreso dentro de cada lección) y se iniciará como si fuese la primera vez.

5.3. No puedo corregir ejercicios C++/C# con Visual Studio

Si al corregir ejercicios C++ o C# usando Visual Studio no aparece ningún resultado, es posible que el fichero `vcvars32.bat` o `vcvars64.bat` no esté funcionando correctamente. Para comprobar el funcionamiento de estos *scripts* abre un terminal (*Símbolo del sistema*) y escribe la **ruta completa** del *script* para cargarlo. Por ejemplo, para la versión 2013 sería algo así:

```
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\vcvars32.bat
```

Esto debería mostrar una serie de líneas indicando las versiones de los distintos componentes de Visual Studio que se han cargado. A continuación, comprueba que tienes acceso a los compiladores de C++ y C# escribiendo los siguientes comandos:

```
>cl.exe
>csc.exe
```

Si alguno de estos comando falla o no se encuentra el compilador indicado, es muy probable que haya algún problema con la instalación de Visual Studio, por lo que sería recomendable reinstalarlo.

Tutoriales Interactivos – Creación de lecciones

Enrique Martín Martín[a] (emartinm@ucm.es)
Revisor: Salvador Tamarit[b] (stamarit@dsic.upv.es)

[a] *Dpto. de Sistemas Informáticos y Computación*
Fac. Informática, Universidad Complutense de Madrid
[b] *Dep. Sistemes Informàtics i Computació*
Universitat Politècnica de València

9 de marzo de 2017

Resumen

La herramienta *Tutoriales Interactivos* organiza su contenido en distintas carpetas dentro del *directorio de temas*. Estas carpetas corresponden a cada uno de los lenguajes de programación soportados, y contienen un fichero YAML por cada tema. A su vez, los temas están definidos como una secuencia de lecciones, que contiene distintos elementos. En este documento se explicará la organización de carpetas y el formato concreto de los ficheros de temas.

Índice

1. Organización de carpetas

Todo el contenido de la aplicación se almacena en la *carpeta de temas*, cuya localización debe establecer el usuario desde la ventana de configuración. Esta carpeta contendrá un directorio por cada lenguaje para el que se quiera proporcionar temas, aunque pueden existir varios directorios para versiones alternativas del mismo lenguaje de programación (por ejemplo *Python 2.x* y *Python 3.x*, o *OpenJDK 6* y *Oracle Java 8*). Para determinar cuál es el lenguaje de programación asociado a un directorio la herramienta *Tutoriales Interactivos* inspecciona su nombre, comprobando si contiene alguna subcadena. Actualmente se soportan 4 lenguajes de programación, y la comprobación de nombre se realiza en el siguiente orden:

1. **Python**: si el nombre del directorio contiene[1] la subcadena «**python**». Para cada uno de estos directorios, la ventana de configuración mostrará una entrada para establecer la ruta del intérprete `python`.

2. **Java**: si contienen la subcadena «**java**» en su nombre. Para cada uno de estos directorios, la ventana de configuración mostrará dos entradas: una para establecer la ruta del compilador `javac` y otra para establecer la ruta del entorno de ejecución `java`.

3. **C++**: si contienen la subcadena «**c++**». Para cada uno de estos directorios, la ventana de configuración mostrará una entrada para establecer la ruta del compilador *GNU C++* (`g++`) o la del fichero de entorno de *Visual Studio* (`vcvars*.bat`).

4. **C#**: si el nombre contiene la subcadena «**c sharp**». Para estos directorios aparecerá una entrada en la ventana de configuración para establecer la ruta del compilador *Mono* (`mcs`) o el fichero de entorno de *Visual Studio* (`vcvars*.bat`).

2. Formato de los temas

Cada directorio que aparece en la carpeta de temas puede contener varios temas. Estos temas se definirán en ficheros con formato YAML[2] con extensión **obligatoria** `.yml`. El formato YAML sirve para representar información como texto plano en diccionarios clave-valor, listas y tipos de datos básicos como booleanos, números, cadenas, etc. Recomendamos consultar la sección «2.1 Collections» de la documentación de YAML (`http://www.yaml.org/spec/1.2/spec.html#id2759963`) para conocer las distintas maneras de representar diccionarios (*mappings*) y listas (*sequences*), puesto que admiten representaciones en una línea o en varias y ambas formas son utilizadas en los ejemplos de este manual.

[1] No importa si las subcadenas aparecen en mayúsculas o minúsculas en el nombre del directorio.

[2] `http://www.yaml.org/spec/1.2/spec.html`

```yaml
Subject: 1
Title: Titulo del tema
Intro: Breve explicación del tema
Lessons:
    - Title: Explicaciones #Primera leccion
      Elements: #Lista de elementos de la leccion
          - Elem: Text #Elemento de tipo explicacion
            Content: | #Texto de varias lineas
              Las explicaciones sirven para mostrar información al alumno.

              Estas esplicaciones se pueden dividir en varios párrafos, y con
              Markdown se pueden incluir **negritas**, *italicas* y ‘código‘.
          - Elem: Text
            Content: |
                **Imagen desde internet**
                ![texto alternativo](http://URL/imagen.png)

                **Imagen desde el directorio de temas con ruta relativa**
                ![triangulo](file://img/triangulo.jpg)

                **Se pueden incluir enlaces en las explicaciones:**
                [Texto del enlace](https://www.ucm.es)
    - Title: Preguntas de tipo test #Segunda leccion
      Elements:
          - Elem: Options #Pregunta de varias opciones
            Content: Pregunta de 3 opciones y una correcta #Texto de la pregunta
            Hint: Esto es la pista general de la pregunta
            Solution: [1] #Opciones correctas, empezando en 1
            Multiple: no #Hay varias opcinones correctas?
            Options: # Lista de opciones para mostrar
                - OPCIÓN CORRECTA
                - Opción incorrecta
                - Opción incorrecta
    - Title: Preguntas de tipo código #Tercera leccion
      Elements:
          - Elem: Code #Pregunta de codigo
            Content: | #Texto de la pregunta, varias lineas
                Estas preguntas solicitan al usuario framentos de código que
                son insertados en una plantilla correctora, que es evaluada.
            Gaps: 2 #Numero de huecos a rellenar
            Prompt: ["Codigo hueco 1","Codigo hueco 2"] #Informacion de cada hueco
            Hint: Esto es la pista general.
            File: correctores/plantilla.py #Plantilla correctora con 'huecos'
```

Figura 1: Código YAML de un tema con tres lecciones.

Un ejemplo de fichero de tema se puede observar en la Figura 1, donde se han añadido comentarios de línea (que comienzan con `#`) para aclarar algunos apartados. El formato concreto es un diccionario YAML con las siguientes claves y valores:

- `Subject`: (OBLIGATORIO) **número** de tema. Sirve para ordenar los distintos temas de un lenguaje a la hora de mostrarlos.
- `Title`: (OBLIGATORIO) **cadena de texto** con el título del tema.
- `Intro`: (OBLIGATORIO) **cadena de texto** que explica brevemente el objetivo del tema. Este texto puede incluir código *Markdown* o incluso código HMTL, como veremos en la Sección 3.
- `Lessons`: (OBLIGATORIO) lista de **lecciones** dentro del tema.

Cada una de las lecciones del tema se representa a su vez como diccionarios con dos claves:

- `Title`: (OBLIGATORIO) **cadena de texto** con el título de la lección.
- `Elements`: (OBLIGATORIO) lista de **elementos** que conforman la lección. Existen tres tipos de elementos: **explicaciones**, **preguntas de varias opciones** y **preguntas de código**. Los distintos elementos se tratarán con detalle en la Sección 4.

El título de tema, su introducción y los títulos de temas son los que *Tutoriales Interactivos* utiliza para mostrar las pantallas de selección de tema y lecciones. En la Figura 2 se puede ver cómo se mostraría la pantalla de selección para elegir entre las tres lecciones del tema definido en la Figura 1.

3. Formatos de texto aceptados

Los distintos elementos de una lección permiten mostrar contenido al usuario. Este contenido puede ser texto plano, pero también se puede formatear y añadir elementos visuales utilizando Markdown[3] o HMTL[4]. La opción preferida es código Markdown por ser el más sencillo y ser suficientemente flexible.

3.1. Markdown

Markdown proporciona muchas características para dar formato a un texto. La herramienta *Tutoriales Interactivos* soporta como mínimo las siguientes:

- **Negrita, itálica y código integrado en la línea**:

```
**negrita**, *italica*, ‘codigo de una linea‘
```

[3] `https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet`
[4] `https://www.w3schools.com/html`

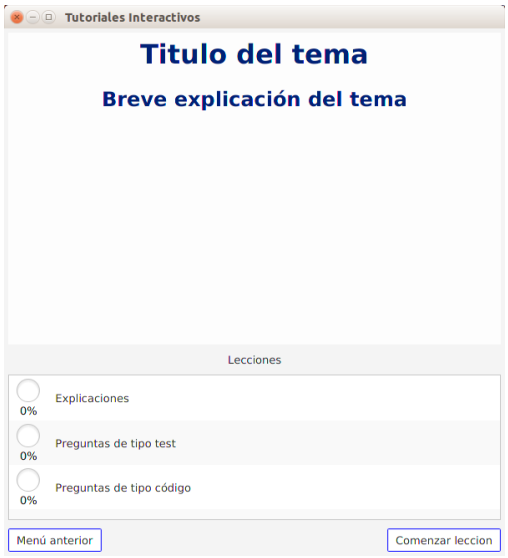


Figura 2: Pantalla de selección de lecciones

- **Bloques de código de varias lineas**:

```
def f(n):
return n + 1
```

- **Cabeceras de sección**:

```
# Cabecera 1
bla bla bla

## Cabecera 2
bla bla bla

### Cabecera 3
bla bla bla
```

- Listas numeradas y no numeradas:

```
* Elemento no numerado 1
1. elemento numerado anidado 1
1. elemento numerado anidado 2
1. elemento numerado anidada 3
* Elemento no numerado 2
* elemento no numerado anidado 1
* elemento no numerado anidado 2
```

- **Imágenes**: pueden ser imágenes remotas o almacenadas en el directorio del tema.

```
**Imagen remota**
![texto alternativo](https://URL/imagen.png)

**Imagen desde el directorio de temas, con ruta relativa**
![triangulo](file://img/triangulo.jpg)
```

  Todas las imágenes cuya ruta comience con `file://` serán consideradas imágenes locales cuya ruta es relativa al directorio donde reside el tema actual. Por ejemplo, si el tema actual reside en «`/opt/temas/Python 3.x`», la imagen «`file://img/triangulo.jpg`» se referirá al fichero situado en «`/opt/temas/Python 3.x/img/triangulo.jpg`».

- **Enlaces** que se abren en el navegador por defecto:

```
[Texto del enlace](https://www.ucm.es)
```

- **Notación matemática LaTeX**. Gracias el entorno MathJax[5] es posible incrustar código LaTeX en el texto. El código LaTeX debe encerrarse entre los símbolos `@@` para fórmulas en la misma línea y `@@@` para fórmulas en párrafos nuevos:

```
Consideremos el polinomio @@ax^2 + bx + c = 0@@
Cuando @@a \ne 0@@, existen dos soluciones:
@@@x = {-b \pm \sqrt{b^2-4ac} \over 2a}.@@@
```

- **Tablas**:

```
Cabecera|Cabecera|Cabecera
:-------|:------:|-------:   #Distinta alineacion horizontal
 1      | 2      | 3
 1      | 2      | 3
```

Recomendamos consultar el «Tema de prueba» que se puede encontrar en `https://github.com/emartinm/TutorialesInteractivos/blob/master/temas/Python%203.x/Tema0.yml` para ver en detalle las distintas características Markdown que se han presentado en esta sección.

[5] `https://www.mathjax.org/`

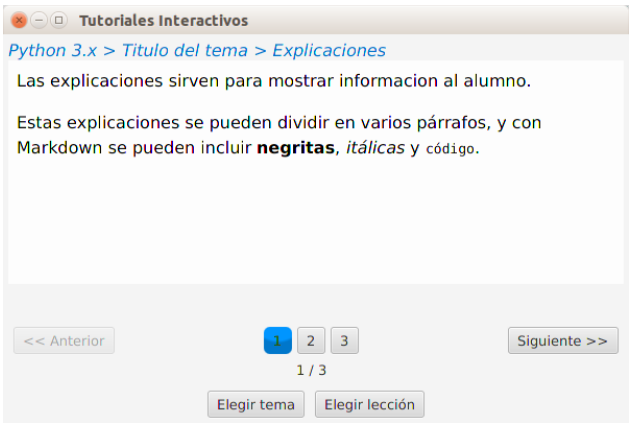


Figura 3: Pantalla mostrando la primera explicación de la lección «Explicaciones»

3.2. HMTL

En aquellas situaciones donde Markdown no es suficiente, se puede incluir código HMTL directamente. Esto puede servir para incrustar vídeos y otros elementos interactivos.[6] Por ejemplo el siguiente código incrustaría el vídeo `https://www.youtube.com/embed/PDpMgx7avzA` como un `iframe`:

```
<iframe width="640" height="360" src="https://www.youtube.com/
  embed/PDpMgx7avzA" frameborder="0" allowfullscreen target="
  _self"></iframe>
```

4. Formato de los elementos de una lección

4.1. Explicaciones

Los elementos de tipo *explicación* sirven para mostrar al alumno texto y otros contenidos gráficos. Se representan como un diccionario de dos claves:

- `Elem: Text` (OBLIGATORIO)

[6] La herramienta *Tutoriales Interactivos* utiliza internamente la componente *WebView* de JavaFX, que está basada en WebKit (`https://webkit.org/`). Por lo tanto, la capacidad para procesar HTML será algo menor al de los navegadores de escritorio usuales y habrá algunas características que no funcionen correctamente o directamente no estén soportadas.

- `Content`: (OBLIGATORIO) **cadena de texto** con el contenido que se quiere mostrar. Puede ser texto plano, código Markdowno HTML. En la Figura 1 se pueden ver ejemplos de contenidos Markdown en las líneas 9–12 y 15–22

4.2. Preguntas de varias opciones

Este tipo de elementos sirve para preguntas en las que el alumno debe elegir la opción u opciones correctas entre varias disponibles. En el fichero YAML las preguntas de varias opciones se representan como un diccionario con las siguientes claves y valores:

- `Elem: Options` (OBLIGATORIO)
- `Content:` (OBLIGATORIO) **cadena de texto** con el enunciado de la pregunta. Al igual que en las explicaciones, este texto puede contener código Markdown o HTML.
- `Hint:` (OPCIONAL) **cadena de texto** con una pista sobre la solución a la pregunta que el alumno puede visualizar pulsando el botón «Pistas».
- `Options:` (OBLIGATORIO) **lista de cadenas** con las distintas opciones que se mostrarán al alumno. Nótese que en YAML las listas se pueden representar en una línea con la notación `[e_1, e_2, ..., e_n]` o en varias líneas usando guiones (`-`) para cada elemento.
- `Multiple:` (OPCIONAL) **booleano** que indica cómo han de mostrarse las distintas opciones de la pregunta. En caso de elegir `no` las opciones se muestras mediante *radio buttons* y únicamente se permite elegir una opción, en caso de elegir `yes` las opciones se muestran mediante *checkboxes* y se pueden elegir una o más opciones. En caso de no incluir esta clave, el valor por defecto es `no`.
- `Solution:` (OBLIGATORIO) **lista de números** con las posiciones de las opciones correctas, donde la primera opción tiene posición `1`. Este campo puede contener la lista vacía (`[]`) si ninguna opción es correcta. Para superar una pregunta es necesario marcar exactamente las opciones que se indican en este campo.

Las lineas 25–33 de la Figura 1 contienen un ejemplo de pregunta de tres opciones donde únicamente la primera es correcta. A la hora de mostrar esta pregunta, *Tutoriales Interactivos* generaría la pantalla de la Figura 4.

4.3. Preguntas de código

Las preguntas de tipo código permiten solicitar al alumno uno o varios fragmentos de código y evaluar su resultado. Para dicha evaluación los fragmentos de código introducidos por el alumno se insertan en los *huecos* definidos en una *plantilla correctora* (ver más adelante), que está escrita en el mismo lenguaje

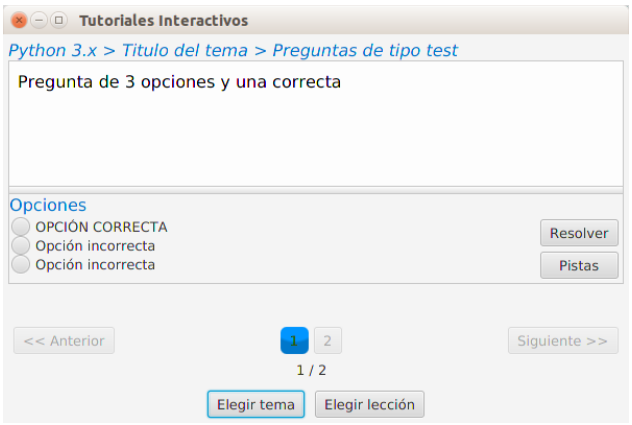


Figura 4: Pantalla mostrando una pregunta de tres opciones con solución única.

de programación que está aprendiendo el alumno. Esta plantilla se rellena y se ejecuta para obtener su veredicto, lo que involucra distintas fases dependiendo del lenguaje de programación: invocar directamente al intérprete (Python), compilar la plantilla rellena y luego invocar el binario generado (C++ y C#), compilar la plantilla rellena y luego interpretar el fichero generado (Java). La estructura de las plantillas correctoras se detallará en la Sección 4.3.1.

En el fichero YAML las preguntas de código se representan como un diccionario con las siguientes claves y valores:

- `Elem: Code` (OBLIGATORIO)
- `Content:` (OBLIGATORIO) **cadena de texto** con el enunciado de la pregunta. Como es usual, este texto puede contener código Markdown o HTML.
- `Gaps:` (OPCIONAL) **número entero** (> 0) que indica el número de huecos que debe rellenar el alumno. Si no se incluye este campo, el valor por defecto es 1.
- `Prompt:` (OPCIONAL) **lista de cadenas de texto** con tantos elementos como se indique en el campo `Gaps`. Indica el texto por defecto que debe aparecer en cada campo de texto cuando está vacío. Sirve para dar indicaciones al alumno sobre lo que debe introducir en cada hueco.
- `Hint:` (OPCIONAL) **cadena de texto** con una pista sobre la solución a la pregunta que el alumno puede visualizar pulsando el botón «Pistas».

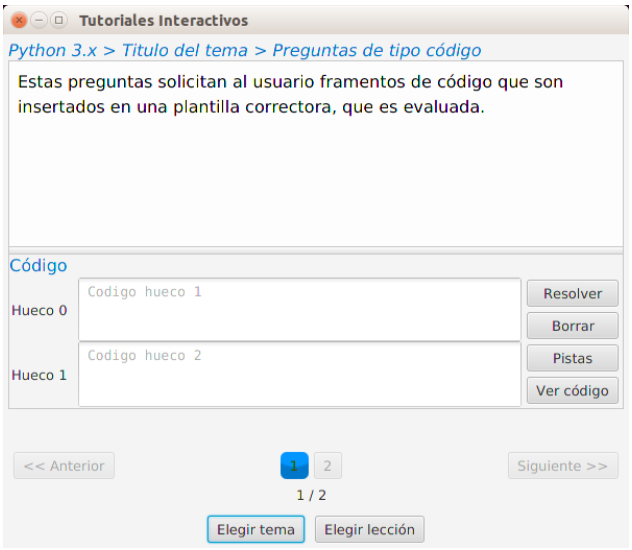


Figura 5: Pantalla mostrando una pregunta de código con dos huecos.

- `File:` (OBLIGATORIO) **cadena de texto** con la ruta de la plantilla correctora *relativa al directorio en el que está el tema actual*. Por ejemplo, si el tema actual está en el directorio «`/opt/temas/Python 3.x`» y la ruta de la plantilla correctar es «`correctores/tema1_p3.py`», la ruta completa de la plantilla será «`/opt/temas/Python 3.x/correctores/tema1_p3.py`».

Las líneas 36–43 contienen una pregunta de tipo código con dos huecos y que se corrige usando la plantilla. `correctores/plantilla.py`. A la hora de mostrar esta pregunta, *Tutoriales Interactivos* generaría la pantalla que se puede ver en la Figura 5.

4.3.1. Plantillas correctoras

Las plantillas correctoras son programas en los que se ha insertado uno o más *huecos*. Para indicar en qué punto de la plantilla hay un hueco, se utiliza el marcador `@@@CODE@@@`. En la plantilla correctora deben aparecer tantas marcas `@@@CODE@@@` como huecos se han definido en la clave `Gaps` del elemento. La Figura 6 muestra un ejemplo de plantilla de un solo hueco para Python, considerando una pregunta en la que se debe asignar el valor 4 a la variable `i`. Como se puede observar, existe un único hueco en la línea 6.

A la hora de corregir una pregunta de tipo código, la herramienta *Tutoriales*

```python
# -*- coding: UTF-8 -*-
import sys
import json

def corrige(filename):
    @@@CODE@@@
    dicc = {}
    things = locals()
    if 'i' in things:
        if things['i'] == 4:
            dicc = {'isCorrect': True}
        else:
            dicc = {'isCorrect': False,
                    'typeError': "Valor incorrecto",
                    'Hints': ['La variable tiene valor' + str(i),
                              'Deberia tener valor 4.'] }
    else:
        dicc = {'isCorrect': False,
                'typeError':"Variable 'i' no asignada",
                'Hints':["Asigna algun valor a la variable 'i'"]}

    with open(filename, 'w') as outfile:
        json.dump(dicc, outfile) #Vuelca 'dicc' como JSON
        sys.exit(0);

if __name__ == "__main__":
    corrige(sys.argv[1]) #sys.argv[1] es la ruta del fichero JSON
```

Figura 6: Ejemplo de plantilla correctora en Python

Interactivos inserta en orden los fragmentos de código del alumno en los distintos huecos y ejecuta la plantilla. Para rellenar estos huecos se tiene en cuenta el *espaciado antes de la marca de hueco*. Volviendo al ejemplo, el hueco de la Figura 6 está a 3 espacios del inicio de fichero. Eso significa que a la hora de reemplazar el hueco con código del alumno, **se insertarán 3 espacios al inicio de cada línea**. Imaginemos que el código del alumno es:

```
a = 1
b = a * 4
i = b
```

A la hora de reemplazar el hueco en la plantilla esta quedaría como se ve en la Figura 7, donde todas las líneas han sido precedidas por exactamente 3 espacios.

Las plantillas correctoras no imponen ninguna restricción a la hora de dar nombre a sus distintas componentes: variables, métodos, etc. La única restric-

```
# -*- coding: UTF-8 -*-
import sys
import json

def corrige(filename):
    a = 1
    b = a * 4
    i = b
    dicc = {}
    ...
```

Figura 7: Plantilla correctora rellena con código del alumno

ción general es que deben estar definidas en un solo fichero y que deben contener una función principal que acepte un parámetro, realice las pruebas necesarias y vuelque el veredicto en un fichero JSON (ver siguiente sección). El caso de **Java** tiene una **restricción adicional**: el método `main(String[] args)` debe estar definido en una clase con nombre `Corrector`, puesto que será la clase que usará *Tutoriales Interactivos* para lanzar la corrección. Recomendamos revisar los correctores de prueba que están en la carpeta `https://github.com/emartinm/TutorialesInteractivos/tree/master/temas` para ver ejemplos de correctores en los distintos lenguajes soportados.

4.3.2. Corrección de plantillas: JSON

Cuando se ejecuta una plantilla rellena, la herramienta *Tutoriales Interactivos* pasa como parámetro una ruta de fichero absoluta donde se debe almacenar el veredicto de la corrección en formato JSON[7]. Este fichero debe contener un diccionario con las siguientes claves:

- `'isCorrect'`: (OBLIGATORIO) **booleano** (`true`/`false`) que indica si el código es correcto.
- `'typeError'`: (OPCIONAL) **cadena de texto** con un mensaje sobre la naturaleza del error.
- `'Hints'`: (OPCIONAL) **lista de cadenas de texto** con pistas detalladas sobre lo que ha fallado o sobre cómo se podría arreglar el código enviado. Cada elemento de la lista se mostrará como una línea de texto cuando el alumno pulse el botón «Más pistas».

La Figura 6 muestra cómo se construyen 3 diccionarios para volcar el resultado en formato JSON: en la línea 11 el código es correcto, por lo que sólo se crea la clave `'isCorrect'` con valor `True`; por otro lado, en las líneas 13 y 18 la clave

[7] `http://www.json.org/`

`'isCorrect'` toma valor `False` y se añaden las otras dos claves opcionales con mensajes concretos para el alumno.

Como hemos comentado anteriormente, cuando *Tutoriales Interactivos* corrige una pregunta de código primero rellena la plantilla y luego la ejecuta. Si esta ejecución lanza algún error (por ejemplo errores de sintaxis durante la compilación, excepciones durante la ejecución...) el alumno recibirá un mensaje indicando esta situación. Además, si la ejecución tarda más de 2 segundos será abortada y se mostrará un mensaje al usuario indicando esta situación, para evitar posibles bucles infinitos. Únicamente en el caso de que la ejecución termine dentro del límite y tenga éxito, la herramienta leerá el fichero JSON generado y mostrará los resultados al alumno.

4.3.3. Visualización de plantillas rellenas

Por último, *Tutoriales Interactivos* permite que el alumno visualice cómo serán rellenados los huecos con su código sin llegar a ejecutarlo. Para ello es necesario que en la plantilla correctora se haya marcado qué fragmento de código se debe mostrar al alumno mediante las marcas `@@@SNIPPET@@@`. Consideremos una pregunta de ćodigo en la que hay que escribir el cuerpo de una función Python que duplica el argumento `n`. La plantilla correctora contendrá el siguiente código:

```
#@@@SNIPPET@@@
def duplica(n):
    @@@CODE@@@
#@@@SNIPPET@@@
```

Como se puede observar, toda la función `duplica` aparece encerrada entre marcas `@@@SNIPPET@@@`. Si el usuario rellena el código «`return n * 2`» en el campo de texto y pulsa el botón «Ver Código» la herramienta reemplazará el hueco y posteriormente mostrará las líneas que están entre las marcas `@@@SNIPPET@@@`. El resultado que vería el alumno sería:

```
def duplica(n):
    return n + 2
```

Obsérvese que las líneas donde aparecen las marcas nunca se mostrarán cuando el alumno visualice el código, sino únicamente las líneas que están **entre las marcas**. Normalmente las marcas `@@@SNIPPET@@@` aparecerán dentro de comentarios, para no afectar a la compilación/ejecución de la plantilla.

5. Crear una lección nueva

Para crear una lección nueva los pasos a seguir serían:

1. Crear un subdirectorio dentro de la carpeta de temas para el lenguaje de programación (o versión) concreto, si no existe ya.

2. Crear un fichero YAML con extensión `.yml` dentro de dicho subdirectorio. En lugar de escribirlo desde cero recomendamos tomar el «Tema de prueba» que se puede encontrar en `https://github.com/emartinm/TutorialesInteractivos/blob/master/temas/Python%203.x/Tema0.yml` y realizar modificaciones sobre él. El formato YAML es sensible al sangrado, así que es importante tenerlo en cuenta a la hora de anidar los distintos elementos. Es por ello que recomendamos utilizar el «Tema de pruebas» como punto de partida.

3. Para crear las plantillas correctoras, recomendamos tomar las usadas en las lecciones de prueba de cada uno de los lenguajes de programación (`https://github.com/emartinm/TutorialesInteractivos/tree/master/temas`) y realizar las modificaciones necesarias respetando el esqueleto general.

6.5. Comunicación en la Jornada *Las TIC en la Enseñanza*

Sistema para el aprendizaje de la programación mediante tutoriales interactivos

Enrique Martín Martín, Adrián Riesco
emartinm@ucm.es, ariesco@fdi.ucm.es
Dpto. Sistemas Informáticos y Computación, Facultad de Informática
Universidad Complutense de Madrid

Resumen

El aprendizaje de lenguajes de programación es imprescindible en las enseñanzas de Informática, pero cada vez tiene más presencia en otras ramas científicas como Matemática, Estadística, Física, Química, Biología, Sociología, etc. Una característica del aprendizaje de programación es que requiere una gran carga práctica, carga que es inviable tratar de manera íntegra en el aula. Además, las primeras tomas de contacto con la programación para alumnos sin ningún conocimiento, al contener una alta cantidad de conceptos y técnicas nuevas, puede generar en ellos una sensación abrumadora que les desmotiva y les hace más proclives a abandonar la asignatura. Por todo ello es importante proporcionar a los alumnos herramientas de apoyo que les permitan practicar la programación fuera del horario lectivo, en pequeñas porciones y de manera amena, y que les muestre retroalimentación inmediata para motivarse y autoevaluarse. En esta comunicación presentamos una herramienta para el aprendizaje de la programación que estamos desarrollando actualmente dentro del proyecto nº 35 de la convocatoria Innova-Docentia 2016/2017, el cual continúa el proyecto nº 109 de la convocatorio 2015 de Proyectos de Innovación y Mejora de la Calidad Docente. El sistema es código abierto y está disponible bajo licencia MIT en `https://github.com/emartinm/TutorialesInteractivos`.

1. Introducción y motivación

El aprendizaje de la programación es una actividad imprescindible en enseñanzas de Ingeniería como la Informática, y su presencia ha ido aumentando en otras ramas técnicas como la Matemática y la Estadística, además de aparecer en otros campos como la Sociología y la Medicina donde es necesario realizar y automatizar análisis sobre conjuntos de datos. Independientemente de la rama, los primeros contactos con la programación pueden ser frustrantes para los alumnos, ya que incluyen una carga teórica y práctica muy alta sobre una temática que en muchos casos es completamente desconocida. Por todo ello, se han desarrollado distintas herramientas para facilitar la práctica de la programación, cuya principal finalidad es corregir automáticamente distintos ejercicios de programación propuestos a los alumnos (Ala-Mutka, 2005; Douce, Livingstone, Orwell, 2005).

Entre estas herramientas automáticas para la corrección de ejercicios de programación cabe destacar los *jueces on-line* como "Mooshak" (Leal, Silva, 2003) o "jutge.org" (Giménez, Petit, Roura, 2012). Este tipo de sistemas, que son utilizados en concursos de programación, están diseñados como herramientas web con una arquitectura cliente-servidor donde solo es necesario un navegador web

en el lado del alumno para poder interactuar. Los jueces on-line toman como entrada un programa escrito por el alumno y generan un veredicto. Para producir este veredicto no se suelen tener en cuenta aspectos internos del código como puede ser su estructura o el seguimiento de una determinada guía de estilo, sino que únicamente se considera su corrección. Para ello se lanza el programa del alumno usando una serie de casos de prueba elaborados previamente por el instructor y se comprueba que las salidas generadas son iguales a las esperadas. En caso de encontrar alguna discrepancia los sistemas muestran un escueto mensaje (pues están diseñados para concursos de programación) indicando la naturaleza del problema (p. ej. "error al compilar", "excepción al ejecutar el programa" o "salida generada incorrecta") pero no incluyen detalles adicionales que ayudarían al alumno a corregir el problema. Para mejorar la capacidad formativa de estos sistemas, hace unos años se desarrolló FLOP (Llana, Martin-Martin, Pareja-Flores, Velázquez-Iturbide, 2014), un nuevo juez on-line abierto para cualquier usuario y cuyos mensajes generados incluyen toda la información necesaria para que el usuario comprenda el error y pueda resolverlo.

Los jueces on-line han sido aplicados a la docencia de la programación durante los últimos años, y en el Dpto. de Sistemas Informáticos y Computación tenemos experiencias positivas tanto en la Fac. de Informática –asignaturas de Fundamentos de Programación (1º), Estructuras de Datos y Algoritmos (2º), Diseño de Algoritmos (3º), etc.– como en la Fac. de Ciencias Matemáticas –asignatura de Informática (1º)–. Sin embargo, los jueces on-line requieren que el alumno tenga un nivel de programación mínimo para poder sacar provecho a su corrección automática. No es raro encontrar alumnos noveles en la materia que encuentran problemas (tanto técnicos como conceptuales) en distintas etapas durante la elaboración de las programas y que nunca llegan a obtener la retroalimentación del sistema, o que sí la reciben pero no saben interpretarla y usar esa información para aprender. Para aliviar este tipo de situaciones en los alumnos que se inician en la programación estamos desarrollando, dentro del proyecto nº 35 de la convocatoria Innova-Docentia 2016/2017, un sistema integrado con distintos tutoriales interactivos en los cuales el alumno adquiere los conocimientos de manera incremental y en porciones muy pequeñas. De esta manera se consigue que no sea necesario crear un programa completamente funcional sino que se puedan practicar pequeñas porciones más básicas: asignaciones, tipos de datos, estructuras de control, etc. Los tutoriales muestran paso a paso los puntos teóricos principales, pero además exigen que para poder avanzar el alumno conteste alguna pregunta o deba escribir algún pequeño fragmento de código que ejercita el contenido teórico directamente involucrado. De esta manera el aprendizaje teórico y la práctica van de la mano y todo el proceso fluye de manera muy guiada. Además, este entorno permite que los alumnos puedan avanzar a su propio ritmo y desde cualquier lugar, y los pequeños logros obtenidos realimentan su motivación y les permiten autoevaluarse.

Aunque el destinatario principal de los tutoriales interactivos son los alumnos que inician el aprendizaje de la programación, hemos detectado otras situaciones en las que resultarían beneficiosos:

- **Tutoriales de refuerzo/recordatorio**: en muchas asignaturas se utilizan lenguajes aprendidos en otras, bien porque se quiere continuar avanzando en el conocimiento del lenguaje, bien porque se utiliza de forma instrumental para implementar soluciones a problemas propios de la asignatura. En estos casos, los tutoriales se utilizarían a modo de refuerzo o recordatorio de los conocimientos ya adquiridos en el pasado, podrían servir para establecer un nivel común entre todos los participantes.

- **Tutoriales de extensión**: estos tutoriales son los que se encargan de explorar nuevos aspectos de lenguajes ya conocidos o la utilización de nuevos módulos o librerías. Son tutoriales específicos que ya asumen un conocimiento básico.
- **Tutoriales de traducción**: en otras asignaturas, sobre todo en cursos avanzados, ocurre que se utiliza un lenguaje nuevo pero la asignatura en sí misma no se centra en estudio de dicho lenguaje sino en su utilización en un determinado ámbito. Para estas asignaturas, los tutoriales más interesantes son los que hemos denominado de traducción. Estos tutoriales se encargarían de mostrar los detalles propios de un nuevo lenguaje a partir de otro conocido.

2. El sistema de tutoriales interactivos

Tomando como base el Trabajo de Fin de Grado de Rafael Caturla y Carlos Congosto (2016), hemos diseñado un sistema para manejar tutoriales interactivos multilenguaje. Todo el sistema es código abierto y está disponible bajo licencia abierta MIT en la dirección `https://github.com/emartinm/TutorialesInteractivos`. Actualmente, el sistema está en sus últimas fases de desarrollo, aunque se siguen incorporando algunas mejoras y corrigiendo los *bugs* detectados.

El sistema de tutoriales interactivos es una aplicación de escritorio desarrollada en Java (concretamente la versión 1.8.0_121) y que está siendo probada en Windows 10, Windows 8, MacOS 10.11.6, Ubuntu 14.04 y Ubuntu 16.04. Como únicamente es necesario disponer de una máquina virtual de Java para poder ejecutar la herramienta la instalación y ejecución es trivial en cualquiera de los sistemas operativos, lo que simplifica al máximo el proceso por parte del alumno. Para la interfaz gráfica se ha decidido utilizar JavaFX por ser el componente gráfico más novedoso de Java y el que más facilidades tiene para incorporar contenido HTML y hojas de estilo. Con respecto a la arquitectura interna, se ha seguido el patrón Modelo-Vista-Controlador para mantener separadas y aisladas las distintas componentes de la herramienta.

El sistema de tutoriales ha sido diseñado para ser fácil de usar por los alumnos y para que sea fácil de ampliar con nuevos temas y lecciones por parte de los instructores. Entre las principales características podemos destacar:

- El contenido de los temas se define en un documento de texto plano con formato YAML (`http://www.yaml.org`), por lo que es muy sencillo modificar y añadir nuevo contenido.
- Las explicaciones y el enunciado de las preguntas se define usando el lenguaje de formateado Markdown, por lo que se puede incluir enlaces, imágenes, tablas, listas, texto en negrita, etc. También se puede incorporar código HTML directamente, lo que permite incorporar otros elementos como vídeos, animaciones o sonido.
- El sistema usa el motor JavaScript MathJax (`https://www.mathjax.org`) para la inclusión de fórmulas matemáticas a partir de código LaTeX.
- Soporta internacionalización, por lo que puede ser adaptada a cualquier otro idioma de manera sencilla. Actualmente está disponible en español e inglés.

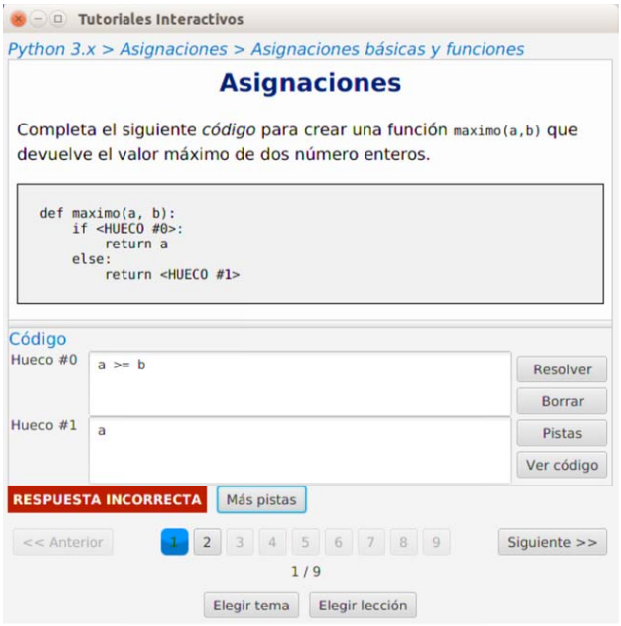


Figura 1. Pregunta de tipo código

- Permite la inclusión de temas para distintos lenguajes: Python, C# (compilador Mono y Visual Studio), C++ (GNU C++ y Visual Studio) y Java. Además el diseño de la aplicación permite la inclusión de nuevos lenguajes de manera sencilla, independientemente de que sean lenguajes compilados, interpretados o que se basen en una representación intermedia.
- El sistema organiza el contenido en temas, cada uno con una o más lecciones. A su vez cada lección puede constar de varios elementos. Por un lado se pueden mostrar **explicaciones**, que es un texto Markdown que explica un determinado concepto de programación. Por otro lado existen las **preguntas de tipo test**, que muestran un enunciado Markdown y una serie de casillas para marcar la respuesta correcta. Estas preguntas pueden ser de respuesta única y múltiple, además de incluir una pista para el alumno. Por último, contamos con las **preguntas de tipo código**, que muestran un enunciado Markdown y uno o más campos de texto en los que el alumno debe introducir código (ver figura 1). Este código se inserta en un programa corrector (escrito en el mismo lenguajes que se está aprendiendo) que evalúa si la respuesta del alumno es correcta. Además, la ejecución del programa corrector se realiza usando el mismo entorno que tiene instalado el usuario en su equipo, por lo que este puede comprobar fácilmente el comportamiento de su código y depurarlo en caso de ser necesario. Para evaluar la corrección de los fragmentos de código el programa corrector puede utilizar técnicas estándar como tests de unidad o realizar comprobaciones manuales. Las preguntas de tipo código pueden incluir una pista general para el alumno, además de permitir que el programa corrector genere pistas adicionales dependiendo del código introducido por el alumno. Esto es útil para detectar errores típicos que suelen cometer los alumnos.

- Los temas y lecciones muestran el progreso del alumno, por lo que en todo momento sabe qué elementos ha completado y cuáles están por realizar.

3. Conclusiones y trabajo futuro

La programación es una disciplina cada vez más presente en las enseñanzas de Grado de distintas ramas, y su aprendizaje requiere de una alta carga práctica. Para facilitar este aprendizaje se están aplicando jueces on-line y otros correctores automáticos; sin embargo, todos ellos requieren de unos conocimientos mínimos que en algunos casos el alumno tarda en adquirir. Por ello proponemos una herramienta para la realización de tutoriales interactivos en los cuales el alumno aprende conceptos de manera incremental y los pone en práctica. De esta manera el aprendizaje teórico y la práctica avanzan a la vez, y no se puede progresar hasta tener afianzados los conceptos previos. Consideramos que de esta manera el alumno podrá avanzar a su propio ritmo, autoevaluarse y que sus propios logros al superar lecciones y temas le animarán a seguir aprendiendo.

La aplicación está alojada en `https://github.com/emartinm/TutorialesInteractivos`. Es una aplicación de escritorio multiplataforma escrita en Java que soporta tanto preguntas de tipo test como preguntas en las que hay que rellenar código. Las lecciones pueden contener texto con formato enriquecido como listas, imágenes, enlaces, animaciones, etc.

Como trabajo futuro tenemos planificado escribir lecciones de iniciación para los lenguajes Python y C#. También queremos presentar la herramienta al profesorado involucrado en asignaturas de programación y recopilar su retroalimentación. Relacionado con el Campus Virtual, nos interesaría estudiar las posibilidades de conectar los tutoriales interactivos con las calificaciones de los cursos. Por último, nos gustaría utilizar los tutoriales interactivos en el curso 2017/18 en asignaturas de introducción a la programación y así poder medir su impacto real en el aprendizaje de los alumnos.

Referencias

Ala-Mutka K. M. (2005). A survey of automatic assessment approaches for programming assignments. Computer Science Education 15(2), 83-102.

Caturla, R., Congosto, C. (2016). Sistema de creación de tutoriales interactivos para el aprendizaje de lenguajes de programación. Trabajo Fin de Grado, Fac. Informática, Universidad Complutense de Madrid. http://eprints.ucm.es/38408

Douce, C., Livingstone, D., Orwell, J. (2005). Automatic test-based assessment of programming: A review. Journal of Educational Resources in Computing, 5(3)

Giménez, O., Petit, J., Roura, S. (2012). Jutge.org: An educational programming judge. In Proc. 17th Annual Conf. Innovation and Technology in Computer Science Education (ITiCSE 2012), 445-450.

Llana L, Martin-Martin E., Pareja-Flores, C., Velázquez-Iturbide, J. A. (2014). FLOP: A User-Friendly System for Automated Program Assessment. J. UCS 20(9): 1304-1326.

Leal, J. P., Silva, F. (2003). Mooshak: a web-based multi-site programming contest system. Software Practice & Experience 33, 567-581.

6.6. Transparencias del taller *Tutoriales Interactivos en el Aprendizaje de la Programación*

UNIVERSIDAD COMPLUTENSE MADRID

Tutoriales Interactivos en el Aprendizaje de la Programación

Enrique Martín - emartinm@ucm.es
Adrián Riesco - ariesco@ucm.es
Fac. Informática UCM – 30 de mayo de 2017

Introducción

Motivación

- **Experiencia personal**: los alumnos de 1º de programación suelen tener dificultades con la materia (*al menos en la Fac. de Ciencias Matemáticas*).
- Cualquier herramienta o técnica que pueda complementar la docencia tradicional es bienvenida.
- Los ***jueces*** revisan automáticamente los programas de los alumnos para muchos casos → rápida retroalimentación.
  - DOMjudge (https://www.domjudge.org, que p.ej. es usado en la FDI en TAIS http://tais.fdi.ucm.es)
  - FLOP (http://problem-g.estad.ucm.es, usado en *Informática* de los Grados de la FCM).

Motivación

- Sin embargo los jueces automáticos tienen alguna *limitación:*
  - Usualmente requieren programas completos.
  - Sus respuestas no son muy informativas.
  - Se ejecutan en remoto, quizá usando sistemas operativos, compiladores y versiones diferentes al alumno.
- *(Al fin y al cabo se diseñaron principalmente para competiciones, no para docencia)*

Idea

- Combinar la **evaluación automática** de código del alumno con el **aprendizaje paso a paso** de conceptos teóricos estilo tutorial.
- Origen de la idea: **swirl** para aprender R (http://swirlstats.com)
  - Modo consola
  - Paso a paso, distintas lecciones
  - Integrable en MOOCs

Nuestra herramienta

- Resultado de 2 proyectos incrementales:
  - PIMCD 2015/109 (http://eprints.ucm.es/35367/)
  - **INNOVA-Docencia UCM (ref. 2016/35)**
- Construido sobre el prototipo realizado en el TFG de Rafael Caturla y Carlos Congosto (http://eprints.ucm.es/38408/), dirigido por Manuel Montenegro y Enrique Martín.

Equipo proyecto 2015/109

- Carlos Gregorio DSIC, Matemáticas UCM
- Enrique Martín DSIC, Informática UCM
- Manuel Montenegro DSIC, Informática UCM
- Adrián Riesco DSIC, Informática UCM
- Jaime Sánchez DSIC, Informática UCM
- Salvador Tamarit DSIC, Informática UPV

Equipo proyecto 2016/35

- Rafael Caturla antiguo alumno
- Carlos Congosto alumno Máster
- Carlos Gregorio DSIC, Matemáticas UCM
- Francisco J. López DSIC, Informática UCM
- Enrique Martín DSIC, Informática UCM
- Adrián Riesco DSIC, Informática UCM
- Jaime Sánchez DSIC, Informática UCM
- Salvador Tamarit DSIC, Informática UPV

Nuestra herramienta

- Aplicación de **escritorio** para aprender lenguajes de programación.
- Muestra de manera **secuencial** distintos bloques:
  - Explicaciones teóricas
  - Preguntas de varias opciones
  - Pequeños ejercicios de codificación
- Para continuar con el siguiente bloque es necesario **completar** las preguntas/ejercicios.

Nuestra herramienta

- Soporta 4 lenguajes de programación:
  - Python
  - C++ (GNU C++)
  - C# (Mono)
  - Java
- Para C++ y C# hay un soporte *experimental* para Visual Studio (usando `vcvarsXX.bat`).
- Es sencillo incorporar nuevos lenguajes, pero requiere que modificar el código de la aplicación.

Nuestra herramienta

- Multiplataforma: mínimo Java 8, JavaFX para la interfaz gráfica.
- Gestión mediante **Maven**.
- Licencia de código abierto **MIT**.
- Disponible en GitHub: https://github.com/emartinm/TutorialesInteractivos

Demostración

Instalación y ejecución

- Instalación sencilla: descargar de GitHub o de un ZIP que prepare el profesor de la asignatura. → **descomprimir y listo**.
- Ejecución: todo el código agrupado en un único fichero JAR:
  - *Script* **.sh** (Linux/Mac) y **.bat** (Windows).
  - Directamente desde la consola con `java`.

Configuración

- Antes de empezar a utilizar la herramienta hay que configurar dos aspectos:
  - Directorio de temas (una carpeta por lenguaje de programación soportado).
  - Ruta al compilador y/o intérprete por cada lenguaje soportado.
- Esta configuración se realiza de manera gráfica.

Explicaciones

- Contienen texto representado mediante **Markdown** (también soporta HMTL directamente):
  - Negrita, monoespaciado, itálica, etc.
  - Listas numeradas y no numeradas, anidamiento
  - Tablas
  - Enlaces que se abren en el navegador
  - Imágenes locales y remotas (incluyendo GIFs animados)
  - Soporte integrado para fórmulas LaTeX
  - Vídeos incrustados

Preguntas de varias opciones

- De dos tipos:
  - Una sola opción válida *(radio buttons)*
  - Varias opciones válidas *(checkboxes)*
- Permiten mostrar **pistas generales** al alumno.
- Hasta que no se resuelven no se puede avanzar.

Preguntas de codificación

- Solicitan al usuario 1 o varios fragmentos de código.
- Permiten mostrar **pistas generales** al alumno.
- Si la solución no es correcta, puede mostrar **pistas particulares** sobre el código del alumno (p.ej. para avisar de errores comunes).
- Hasta que no se resuelven no se puede avanzar.

Preguntas de codificación

- Las preguntas de codificación se corrigen usando un **programa corrector**, que es un programa con uno o más *"huecos"*.
- El código del alumno se incrusta en esos huecos → **programa completado**.
- El programa completo se compila/interpreta, comprobando la validez del código del alumno y generando un fichero **JSON con 3 campos:**
  - `'isCorrect'`: true/false
  - `'typeError'`: mensaje sobre la naturaleza del error
  - `'Hints'`: lista de pistas

Definición de temas

- Fichero YAML por cada tema *(ojo que YAML tiende a ser un poco molesto con espacios y tabuladores)*.
- Cada tema contiene una lista de lecciones.
- Cada lección contiene una lista de elementos.
- Cada elemento está definido como un diccionario clave-valor de campos.

Más información

- Hay varios manuales en la carpeta de documentación de la herramienta:
  - Manual del usuario https://github.com/emartinm/TutorialesInteractivos/raw/master/doc/Manual_usuario.pdf
  - Manual para crear lecciones https://github.com/emartinm/TutorialesInteractivos/raw/master/doc/Manual_crear_lecciones.pdf

Conclusiones

Aspectos interesantes

- Orientado a lecciones cortas, pequeños pasos → **motivación**.
- Usa el mismo sistema (sistema operativo y compilador) que tiene el alumno en su ordenador.
- Contenido y soluciones incluidas en el fichero descargado → no necesita conexión a Internet.
- Puede complementar a los jueces automáticos.

Aspectos interesantes

- Aplicable a **distintas situaciones**:
  - **Introducción** a la programación desde 0: asignaciones, operaciones aritméticas, condicionales, bucles, listas, etc.
  - **Refuerzo/recordatorio**: para asignaturas que utilizan lenguajes aprendidos en otras.
  - **Extensión**: para explorar nuevos aspectos de lenguajes ya conocidos o la utilización de nuevos módulos o librerías.
  - **Adaptación**: para mostrar los detalles propios de un nuevo lenguaje a partir de otro conocido.
- Aplicable a **distintos niveles educativos**: Universidad, Formación Profesional, Bachillerato o incluso ESO.

Ideas para el futuro

- Gestión de distintos usuarios, posiblemente mediante un servidor central que almacena sus progresos.
- Integración con el Campus Virtual → puntos adicionales a los alumnos según superen lecciones.
- Soportar más lenguajes de programación (Maude, Haskell, Scala) e incluso otros paradigmas (ensamblador, VHDL).

¿Mejora el aprendizaje?

- Queremos pensar que sí, pero hay que medirlo.
- Hemos solicitado un nuevo proyecto INNOVA-Docencia para el curso 2017/18:
  - Aplicar a grupos reducidos en FCM y FDI
  - Obtener indicadores objetivos y subjetivos
  - Analizar la magnitud de la mejora obtenida

Discusión

- ¿Consideráis que la herramienta puede mejorar el aprendizaje de la programación?
- ¿La herramienta se podría aplicar en vuestras clases? ¿Lo haríais?
- ¿Qué aspectos se podrían mejorar para hacerla más útil?

Encuesta

- Sería muy interesante recabar vuestra opinión sobre la herramienta.
- Podéis encontrar una breve encuesta en: https://goo.gl/forms/5erVztzgdOPPJigw1

  (se tarda unos 4 minutos, son pocas preguntas)

¡Gracias!

6.7. Comentarios de retroalimentación

Durante la realización del taller *Tutoriales Interactivos en el Aprendizaje de la Programación*, celebrado el día 30 de mayo de 2017 en la Fac. de Informática con una duración de 2 horas, recibimos distintos comentarios y propuestas para mejorar la herramienta. En esta sección detallamos las principales ideas que anotamos durante el taller:

- Sería interesante que las preguntas de respuesta múltiple no fueran siempre las mismas dentro de una lección, sino que se extrajesen de un conjunto de preguntas de manera similar al funcionamiento de Moodle.
- Es necesario detallar qué ocurre en las preguntas de código con varios huecos a rellenar cuando se deja algún hueco vacío. De cara al alumno puede no quedar claro.
- El sistema actual está diseñado para que el programa corrector de preguntas de tipo código esté escrito exactamente en el mismo lenguaje de programación sobre el que se está realizando el tutorial. Puede ser interesante soportar correctores externos escritos en lenguajes de programación diferentes.
- Dada la popularidad de los jueces automáticos en la docencia de la programación (DomJudge, *Acepta el reto*, FLOP, Mooshak, etc.) sería interesante desarrollar un proceso automático para traducir los casos de prueba usados en dichos jueces a los tutoriales soportados en la herramienta.
- Dependiendo del lenguaje de programación, es posible utilizar LeX y Yacc (herramientas para en análisis léxico y sintáctico) para obtener el mensaje de error presentado al alumno y adaptar su número de línea.
- Dado que el sistema está basado en los compiladores/intérpretes que el alumno tiene instalados en su equipo, existe la posibilidad de que el programa corrector desarrollado por el profesor no funcione correctamente con dichas versiones. La moraleja que extraemos es que a la hora de implementar programas correctores es *imprescindible* ceñirse al estándar del lenguaje de programación, evitando en lo posible cualquier extensión.
- La Facultad de Informática está organizando un proyecto piloto de tutorías académicas para las asignaturas de Fundamentos de Programación y Estructuras de Datos y Algoritmos de las 4 titulaciones de grado. En este proyecto piloto los alumnos de cursos avanzados colaborarán a cambio de créditos para ayudar a compañeros a resolver ejercicios y dudas. Nos comentaron que, dado que la herramienta desarrollada encaja en la asignatura de Fundamentos de Programación, sería interesante proponer a los alumnos *mentores* la elaboración de temas y lecciones para sus compañeros.

6.8. Encuesta de retroalimentación

Opinión tutoriales interactivos

Tu dirección de correo electrónico (**ariesco@ucm.es**) se registrará cuando envíes este formulario.
¿No es tuya la cuenta **ariesco**? Cerrar sesión

1. **Tipo usuario**
   Marca solo un óvalo.
   - ◯ Profesor
   - ◯ Alumno

2. **Facilidad de instalación y configuración**
   Marca solo un óvalo.

1	2	3	4	5
◯	◯	◯	◯	◯

3. **Facilidad de ejecución**
   Marca solo un óvalo.

1	2	3	4	5
◯	◯	◯	◯	◯

4. **Facilidad de navegación entre apartados**
   Marca solo un óvalo.

1	2	3	4	5
◯	◯	◯	◯	◯

5. **"Amigabilidad" de la interfaz**
   Marca solo un óvalo.

1	2	3	4	5
◯	◯	◯	◯	◯

6. **Utilidad de la herramienta**
   Marca solo un óvalo.

1	2	3	4	5
◯	◯	◯	◯	◯

7. **¿Usarías los tutoriales interactivos en tu docencia?**
Marca solo un óvalo.

- Sí
- No
- NS/NC

8. **¿Qué aspectos de la herramienta mejorarías?**

9. **Por favor, coméntanos cualquier cosa que consideres relevante sobre la herramienta o el taller**

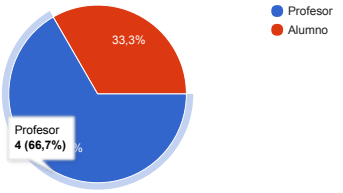
6.9. Resultados de la encuesta

Opinión tutoriales interactivos

6 respuestas

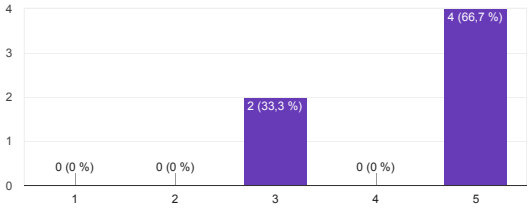
Tipo usuario

6 respuestas



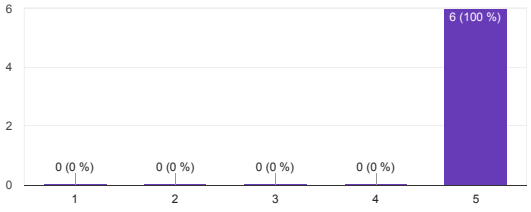
Facilidad de instalación y configuración

6 respuestas



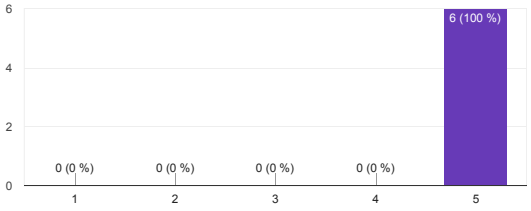
Facilidad de ejecución

6 respuestas



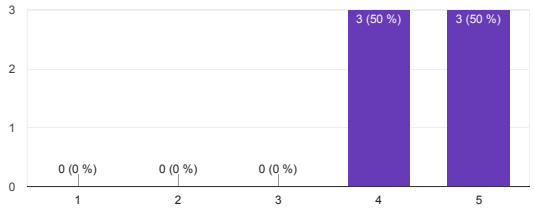
Facilidad de navegación entre apartados

6 respuestas



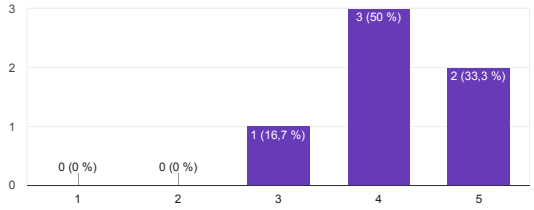
"Amigabilidad" de la interfaz

6 respuestas



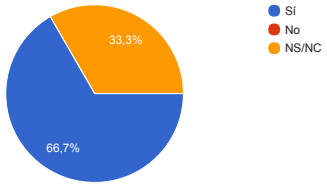
Utilidad de la herramienta

6 respuestas



¿Usarías los tutoriales interactivos en tu docencia?

6 respuestas



¿Qué aspectos de la herramienta mejorarías?

3 respuestas

Crearía un editor de preguntas y de validadores para hacerlo más sencillo y rápido

Para permitir una rápida adaptación de algunos de los problemas de los jueces a vuestra herramienta, podríais montar una plantilla del corrector/validador con las funciones ya preparadas para leer casos de prueba (cambiando el formato de lectura de los datos y poco más), la función principal para rellenar, y la exportación/comprobación de los resultados obtenidos.

La creación de correctores parece algo incómoda.

¿Y hacer que la herramienta funcione online?

En los casos en que la respuesta del alumno tenga el efecto pedido pero no sea sintácticamente igual a la esperada por el profesor, podría mostrarse la respuesta estándar. Por ejemplo, si el alumno escribe i=i=4 en C, habría que decirle que funciona, pero que es mejor i=4 (creo que tienen el mismo efecto). Sospecho que esas respuestas exóticas no son raras en alumnos principiantes.

Por favor, coméntanos cualquier cosa que consideres relevante sobre la herramienta o el taller

1 respuesta

Nada que no se haya comentado ya... Enhorabuena por el trabajo, a ver si tenéis suerte y os dan más fondos en el próximo PIMCD

Número de respuestas diarias