UNIVERSIDAD
COMPLUTENSE
MADRID

Proyecto de Innovación y Mejora de la Calidad Docente

Convocatoria 2015

Proyecto Nº 109

Aprendizaje de lenguajes de programación mediante tutoriales interactivos: diseño y aplicabilidad

Responsable: Enrique Martín Martín

Facultad de Informática

Dpto. Sistemas Informáticos y Computación

1. Objetivos propuestos en la presentación del proyecto

El proyecto propone 12 objetivos:

- O1: Revisar los sistemas existentes para el aprendizaje de la programación, tanto los basados en interacción con los usuarios como los basados en jueces automáticos y otros enfoques colaborativos aplicados en el ámbito universitario.

- O2: Estudiar las asignaturas de la UCM en las cuales puede ser beneficioso un sistema de tutoriales interactivos para el aprendizaje de la programación. Recopilar los lenguajes de programación involucrados.

- O3: Describir los puntos y contenidos básicos que debería cubrir un sistema de tutoriales interactivos para el aprendizaje de la programación desde 0 para alumnos sin ningún conocimiento previo. Estos contenidos deberían ser compatibles con los temarios de las asignaturas de introducción a la programación en los distintos planes.

- O4: Estudiar la posible aplicación de tutoriales interactivos para aprender nuevos lenguajes de programación para alumnos que ya poseen conocimiento previo.

- O5: Estudiar el formato y alcance más adecuado para ofrecer tutoriales interactivos: aplicación específica para un lenguaje o entorno genérico.

- O6: Estudiar los requisitos hardware y software necesarios para implementar un sistema de tutoriales interactivos.

- O7: Estudiar los requisitos de personal necesarios para llevar a cabo el sistema.

- O8: Estudiar la posibilidad de emitir algún tipo de certificado que acredite la superación de algunas partes de los tutoriales interactivos. Estos certificados, además de su valor motivador, podrían servir a los profesores para conceder puntos extra o para conformar la calificación por participación en clase del alumno.

- O9: Estudiar cómo integrar este sistema con el Campus Virtual.

- O10: Estudiar si es interesante (y cómo llevarla a cabo) una integración con redes sociales, que permitiría un intercambio más ágil de resultados y proporcionaría motivación adicional a los usuarios.

- O11: Estudiar la integración con sistemas móviles (tabletas y smartphones).

- O12: Estudiar las posibilidades de uso del proyecto fuera del ámbito académico.

2. Objetivos alcanzados

El presente Proyecto de Innovación y Mejora de la Calidad Educativa ha cumplido con los 12 objetivos propuestos en su solicitud.

Consideramos que la aplicación de tutoriales interactivos es **viable** y que puede reportar diversos beneficios tanto a estudiantes como a docentes. Debido a su longitud los resultados detallados de cada una de las 5 tareas han sido incluidos como anexos en la sección 6.

3. Metodología empleada en el proyecto

Tal como se planteó en la memoria de solicitud del proyecto, hemos agrupado los objetivos en 5 tareas que hemos abordado de acuerdo a lo previsto:

- Tarea 1 (Objetivo 1): Revisión del estado del arte. Hemos analizado distintas herramientas análogas a la que planteamos o relacionadas. En particular, teníamos previsto estudiar “Swirl”, “Udemy”, “Code Academy”, “Mooshak”, “CodingBat”, “Duolingo”. También nos ha sido de utilidad nuestra experiencia con “Acepta el Reto” y “FLOP”, jueces automáticos diseñados y utilizados en nuestra Universidad.

- Tarea 2 (Objetivos 2, 4, 12): Estudio de aplicabilidad. En esta tarea nos interesaba estudiar la viabilidad real de la propuesta fundamentalmente para la docencia, pero también en el ámbito empresarial. Los profesores involucrados en el proyecto impartimos o hemos impartido un amplio conjunto de asignaturas de programación en distintas titulaciones, conocemos los programas de las mismas y tenemos una amplia experiencia conjunta. Esta tarea básicamente ha consistido en discutir propuestas, posibles dificultades y plasmar ideas que necesitan contraste experimental.

- Tarea 3 (Objetivos 3, 5, 8): Enunciado de funcionalidad y objetivos concretos. En esta tarea hemos concretado la funcionalidad básica de la herramienta, pensando además en la versatilidad de la misma de a cara a futuras extensiones y/o diversidad en los ámbitos de aplicación. La herramienta soportaría tres tipos de preguntas: de opción múltiple, basadas en gramáticas y evaluadas en el lenguaje de programación destino. Hemos analizado posibles formatos y entornos que faciliten la implementación de acuerdo a los requisitos establecidos.

- Tarea 4 (Objetivos 6, 7): Captura de requisitos para la implementación. Estos requisitos incluyen hardware, software y personal para el desarrollo de la herramienta. Los dos primeros se han resuelto fácilmente. En cuanto al personal, de cara a desarrollar un primer prototipo hemos considerado la propuesta de un Trabajo de Fin de Grado.

- Tarea 5 (Objetivos 9, 10, 11): Integración con otros sistemas. Para facilitar la posible integración en plataformas existentes hemos optado por un lenguaje altamente portable como Java, utilizar el formato YAML para las baterías de lecciones por un su sencillez y estandarización y el lenguaje marcado Markdown para la representación de lecciones con gran variedad de formatos: listas, tablas, fragmentos de código, imágenes incrustadas, etc.

4. Recursos humanos

El proyecto fue realizado por un equipo de 6 doctores:

- Enrique Martín Martín
- Adrián Riesco Rodríguez
- Manuel Montenegro Montes
- Salvador Tamarit
- Jaime Sánchez Hernández
- Carlos Gregorio Rodríguez

Los miembros del proyecto fueron asignados a distintas tareas considerando dos roles principales: *responsable* de tarea y *participante* en tarea. La distribución concreta fue:

Tarea 1: Revisión del estado del arte.
Responsable: Enrique Martín
Participantes: Adrián Riesco, Salvador Tamarit

Tarea 2: Estudio de aplicabilidad
Responsable: Carlos Gregorio
Participantes: Jaime Sánchez, Manuel Montenegro

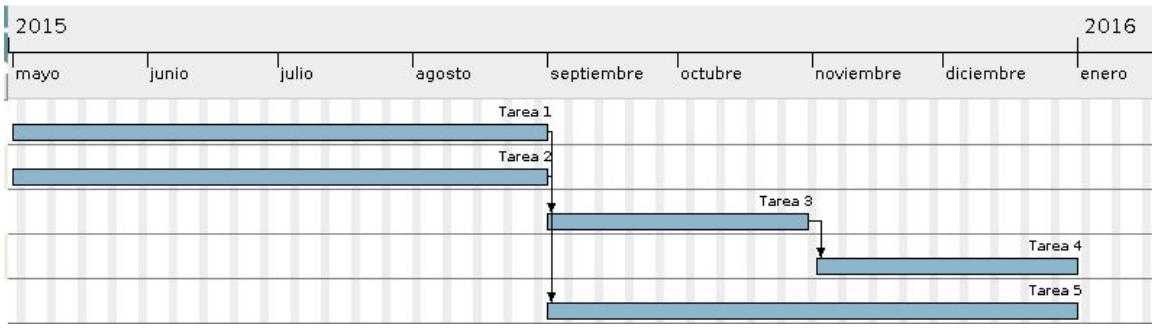
Tarea 3: Enunciado de funcionalidad y objetivos concretos
Responsable: Adrián Riesco
Participantes: Salvador Tamarit, Enrique Martín

Tarea 4: Captura de requisitos para la implementación
Responsable: Salvador Tamarit
Participantes: Adrián Riesco, Enrique Martín

Tarea 5: Integración con otros sistemas
Responsable: Manuel Montenegro
Participantes: Carlos Gregorio, Jaime Sánchez

5. Desarrollo de las actividades

En esta sección detallaremos el desarrollo de las distintas tareas del proyecto, planificadas para el periodo mayo-diciembre de 2015. La organización de las distintas tareas siguió el orden:



- Tarea 1: mayo a agosto.
- Tarea 2: mayo a agosto.
- Tarea 3: septiembre a octubre. Precondición T1, T2.
- Tarea 4: noviembre a diciembre. Precondición T3.
- Tarea 5: agosto a diciembre. Precondición T1, T2.

De esta manera pudimos formar dos grupos de trabajo que avanzaron de manera paralela, aunque sus resultados principales fueron puestos en común para el conocimiento de todos los integrantes del proyecto.

- Equipo 1 (Tareas 1, 3 y 4):
    - Enrique Martín
    - Adrián Riesco
    - Salvador Tamarit
- Equipo 2 (Tareas 2 y 5):
    - Carlos Gregorio
    - Manuel Montenegro
    - Jaime Sánchez

En la Tarea 1 localizamos los distintos sistemas a evaluar y los distribuimos en 2 grupos: herramientas basadas en tutoriales y jueces automáticos. Cada miembro del equipo inspeccionó las características de los distintos sistemas asignados y las pusimos en común. Finalmente el responsable de la tarea escribió el informe combinando el conocimiento adquirido para resumir el estado del arte (ver sección 6).

Para la Tarea 2 se revisaron las fichas docentes de los grados y posgrados en los que tiene presencia nuestro departamento (Fac. Informática, Ciencias Matemáticas y Estudios Estadísticos) para detectar aquellas que tenían un componente de programación. Una vez localizadas las asignaturas, se anotó los lenguajes de programación o librerías involucradas. En paralelo se investigó la posible aplicación de los tutoriales interactivos en ámbitos externos al académico, centrándose en su aplicación en el mundo empresarial y en la formación de personal. También se

consideró el impacto y la aplicación de tutoriales interactivos para el aprendizaje de la programación teniendo en cuenta principalmente la experiencia docente de cada miembro del grupo. Estas tres fases se realizaron de manera bastante autónoma entre los distintos integrantes, aunque los resultados y aspectos más importante se pusieron en común entre todos los miembros de la tarea. Finalmente el responsable escribió el informe de aplicabilidad (ver sección 6) resumiendo las principales cuestiones acerca de la aplicabilidad, cuya valoración fue altamente **positiva**.

Teniendo en cuenta el estado del arte y el informe positivo de aplicabilidad, en la Tarea 3 se definieron los objetivos concretos que nuestra herramienta debería cumplir. Estos objetivos incluyen la descripción de una funcionalidad básica pero también de sus posibles extensiones. Para ello los integrantes de la tarea se reunieron hasta consensuar un listado de características deseables, que quedó plasmada en el informe que se puede encontrar en la sección 6.

Considerando los resultados de las Tareas 1, 2 y 3, se realizó un boceto con las principales decisiones de diseño y tecnologías involucradas para implementar el sistema de tutoriales interactivos (Tarea 4). Se hizo hincapié en tres recursos: software, hardware y personal. El desarrollo siguió un enfoque de reuniones entre los miembros de la tarea para detectar todos los requisitos deseables y para detectar qué tecnologías se podrían utilizar. El resultado se recoge en el informe de requisitos incluido en la sección 6.

A la vez que se realizaban las Tareas 3 y 4, la Tarea 5 sirvió para detectar los beneficios de la integración de un sistema de tutoriales interactivos con otras otras herramientas y plataformas. Se dió prioridad al Campus Virtual de la UCM, ya que sería la herramienta que más impacto tendría en la mejora de la docencia de nuestras asignaturas. Sin embargo también se consideró la posibilidad de desarrollar apps para sistemas móviles (tabletas y smartphones), y la inclusión de redes sociales (Facebook, Twitter, Google+ o LinkedIn) para mejorar la visibilidad y fomentar el autoaprendizaje gracias a la gamificación. En el desarrollo de esta tarea se dividieron los sistemas a estudiar entre los distintos miembros, que pusieron en común las ventajas e inconvenientes de cada uno. Finalmente el responsable de la tarea redactó el informe de integración que se puede encontrar en la sección 6.

6. Anexos

Resultados de la tarea 1: *Revisión del estado del arte*

Esta tarea consistió en el estudio de técnicas y herramientas de aprendizaje. Las herramientas de interacción con el usuario a estudiar son principalmente *swirl*[1] para el lenguaje R, que se asemeja bastante a la idea que queremos implementar, pero también *Udemy*[2] o *Code Academy*[3]. Por otra parte, los jueces automáticos son herramientas que dado un programa en código fuente comprueban si éste devuelve los resultados adecuados para unos conjuntos de prueba (p.ej, *Mooshak*[4] o *CodingBat*[5]). También se revisaron herramientas similares para el aprendizaje colaborativo como Duolingo.

Swirl es una herramienta interactiva para el aprendizaje del lenguaje R. Su característica principal es que está desarrollada en el propio lenguaje R y por tanto se integra a la perfección en el terminal de R. De esta manera cualquier usuario puede acceder a ella y comenzar a aprender con el único requisito de tener correctamente instalado el entorno de desarrollo de R. Esto es muy conveniente de cara a usuario noveles y alumnos con escasa formación informática. Una vez seleccionada una lección, swirl muestra paso a paso las explicaciones intercaladas con las preguntas, que bloquean el progreso hasta que se contestan correctamente.

Entre las demás características de *swirl* destacaremos:

- Es código abierto con un repositorio público GitHub[6]. Cualquier persona puede utilizarlo, modificarlo y comprobar cómo funciona. Esto le proporciona visibilidad e impide que el proyecto muera de manera prematura, pues fomenta la creación de un grupo de personas interesadas.
- Las lecciones se representan en el lenguaje de marcado YAML, similar a HTML o XML pero más visual y simple para humanos. Esto permite que la creación de nuevas lecciones por parte del profesor sea sencilla, ya que solo debe modificar una plantilla añadiendo las explicaciones y preguntas adecuadas. Además esta edición se puede realizar con cualquier editor de texto.
- Las preguntas se corrigen utilizando el propio lenguaje R. Es decir, para saber si una respuesta es correcta, el texto escrito por el alumno se ejecuta en el terminal de R y el resultado obtenido se verifica. Esta verificación recae en el profesor, que debe proporcionar el código adecuado para llevar a cabo esta revisión.
- Es posible utilizarlo para la obtención de puntos adicionales en cursos on-line. Concretamente el curso *R Programming* de la Universidad Johns Hopkins

[1] http:// swirlstats.com
[2] http:// www.udemy.com
[3] https://www.codecademy.com
[4] https://mooshak.dcc.fc.up.pt
[5] http://codingbat.com
[6] https://github.com/swirldev/swirl

alojado en Coursera[7] proporciona puntos extra a los alumnos que superen distintas lecciones en *swirl*. Consideramos que este enfoque es muy interesante, ya que motiva a los alumnos a aprender más allá de los contenidos mínimos y practicar.

A pesar de ser una herramienta muy interesante, apreciamos algunos puntos débiles:

- Como *swirl* está integrado en el propio terminal, el aspecto visual es bastante pobre, pues solo soporta texto plano. Sería interesante permitir texto formateado como listas, tablas, imágenes incrustadas, etc.
- Está íntimamente ligado al lenguaje de programación R, por lo que no se puede utilizar para el aprendizaje de otros lenguajes.
- Solo permite preguntas en las que el alumno escriba código. Aunque es un enfoque flexible y potente sería interesante tener otro tipo de preguntas como las de opción múltiple o preguntas cuya respuesta se validase con respecto a una determinada gramática, para garantizar que poseen una estructura concreta.

Además de swirl, es posible encontrar tutoriales interactivos y gratuitos para el lenguaje de programación Python como por ejemplo http://www.trypython.org y http://www.learnpython.org. Estos tutoriales tienen un formato on-line, por lo que no es necesario descargarse ningún programa (salvo algunos complementos en el ordenador). Ambos proporcionan distintos apartados para aprender y practicar y muestran una explicación seguida de un ejercicio. Este ejercicio está orientado siempre a modificar o completar un programa completo para que realice una determinada acción. Aunque son dos herramientas interesantes, creemos que no encajarían en la docencia de la UCM por las siguientes razones:

- Todas las explicaciones y ejercicios están escritos en inglés.
- Sólo soportan el lenguaje de programación Python.
- Son herramientas cerradas, por lo que no es posible incluir nuevas lecciones ni obtener los resultados de los alumnos.

En esta misma categoría englobaríamos a plataformas maduras para el aprendizaje de la programación como *Udemy*, *Code Academy* o *Code School*[8]. Todas estas plataformas permiten el aprendizaje de distintos lenguajes de programación a base de pequeños vídeos y ejercicios que se resuelven a través del navegador. Sin embargo comparten algunos de los aspectos negativos mencionados anteriormente: están mayoritariamente en inglés y al ser herramientas cerradas (y en algún caso de pago) no permiten añadir nuevas lecciones con facilidad.

Dentro del aprendizaje de la programación son comunes los jueces automáticos como *Mooshak*, *CodingBat*, *UVA Online Judge*[9] o *DOM Judge*[10]. Estas herramientas web funcionan como *correctores de caja negra* para los programas enviados: reciben los programas de los usuarios, los evalúan en base a unos casos de prueba previamente

[7] https://es.coursera.org/learn/r-programming/
[8] https://www.codecademy.com/es
[9] https://uva.onlinejudge.org
[10] https://www.domjudge.org

especificados y muestran el resultado. En ningún momento muestran explicaciones para el alumno, únicamente muestran el problema a resolver de manera muy detallada y reciben la solución expresada en alguno de los lenguajes soportados. Aunque son herramientas muy maduras y ampliamente utilizadas en concursos de programación, consideramos que no se adecúan a la idea que perseguimos de entrelazar explicaciones teóricas y ejercicios o preguntas prácticas de manera fluida.

Una mención especial dentro de esta categoría merecen los jueces automáticos *Acepta el reto*[11], *FLOP*[12] y *jutge.org*[13], desarrollados en el seno de la UCM y de la UPC. Aunque siguen el mismo enfoque de caja negra mencionado anteriormente, y por tanto no serían adecuados para entrelazar explicaciones y ejercicios, son utilizados de manera integrada en algunas asignaturas de aprendizaje de la programación. Por ejemplo *FLOP* se usa actualmente en la asignatura *Informática* de los Grados de la Facultad de Ciencias Matemáticas, donde se enseña la programación en Python. Concretamente *FLOP* verifica las prácticas enviadas por los usuarios y proporciona una calificación objetiva en base a los casos de prueba correctamente pasados. Además en algunas asignaturas como *Estructuras de datos y algoritmos* o *Métodos algorítmicos en resolución de problemas* de los Grados de la Facultad de Informática se utiliza el juez *DOM Judge* o *Acepta el reto* para las sesiones de laboratorio y también para que los alumnos practiquen libremente en casa.

[11] https://www.aceptaelreto.com
[12] problem-g.estad.ucm.es/FLOP
[13] https://www.jutge.org/

Resultados de la tarea 2: *Estudio de aplicabilidad*

En la UCM existen un buen número de facultades, titulaciones y asignaturas en las que se enseñan o utilizan lenguajes de programación. Centramos nuestro estudio en las facultades en las que el departamento de Sistemas Informáticos y Computación (al que pertenecen los miembros del equipo) imparte docencia. A continuación damos un listado de las principales asignaturas en las que se utilizan lenguajes de programación, incluyendo el número de grupos que reciben dicha asignatura y el lenguaje o librería utilizados:

Fac. Informática

Asignatura	*Grupos*	*Lenguaje o librería*
Fundamentos de Programación	8	C++
Estructuras de Datos y Algoritmos	6	C++
Tecnología de la Programación	6	Java, C++
Bases de Datos	3	SQL
Ampliación de Bases de Datos	2	SQL
Programación de Sistemas y Dispositivos	1	C
Sistemas Web	1	Java, PHP, jQuery
Diseño de Algoritmos	1	C++
Técnicas algorítmicas en ingeniería del software	1	C++
Programación Concurrente	1	Java
Programación declarativa	1	Haskell, Prolog
Métodos algorítmicos en resolución de problemas	2	C++
Aplicaciones web	1	Java, PHP, jQuery
Informática gráfica	1	C++, OpenGL
Especificación, validación y testing	1	Maude
Programación con restricciones	1	MiniZinc/Gecode

Desarrollo de videojuegos mediante tecnologías web	1	JavaScript, Unity, C#, HTML5
Gestión de la información en la web	1	Python, MongoDB
Bases de Datos noSQL	1	MongoDB, HBase
Motores de Videojuegos	1	Unity
Sistemas de gestión de datos y de la información	1	Python, MongoDB
Auditoría, calidad y fiabilidad informáticas	1	Maude
Desarrollo de videojuegos	1	Unity, C#
Programación declarativa aplicada	1	Erlang

Fac. Matemáticas

Asignatura	***Grupos***	***Lenguaje o librería***
Informática	5	Python
Métodos Numéricos	5	MATLAB
Programación Paralela	1	Python
Geometría Computacional	1	Python
Estructuras de Datos	1	Python, Java
Programación Declarativa	1	Haskell, Prolog
Bases de Datos	1	SQL, MySQL
Álgebra Lineal Numérica	1	MATLAB

Fac. Estudios Estadísticos

Asignatura	***Grupos***	***Lenguaje o librería***
Programación I	3	C++
Programación II	3	C++

Software Estadístico	3	SAS
Bases de Datos	2	SQL, MySQL
Software Estadístico II	2	R
Matemáticas con ordenador	2	MATLAB
Taller de Algoritmos	1	C++

En todas las titulaciones ofrecidas en las Facultades de Informática, Estudios Estadísticos y Matemáticas, se aprecia que los alumnos tienen que utilizar distintos lenguajes de programación, módulos o paquetes informáticos en distintas asignaturas.

Es claro que un recurso docente como los tutoriales interactivos podrían ser utilizados en todas ellas, si bien hemos detectado que podríamos hacer una clasificación de los tipos de tutoriales a utilizar:

- Tutoriales Introductorios
- Tutoriales de refuerzo/recordatorio
- Tutoriales de extensión
- Tutoriales de traducción

Tutoriales Introductorios
Son los adecuados para los primeros cursos de programación. Estos tutoriales no asumen conocimientos previos de programación, tienen que ser muy progresivos e ir tratando todos los aspectos y conceptos esenciales de la programación.

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 los 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. Estos tutoriales no necesitarían ser tan progresivos y exhaustivos como los anteriores.

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. En numerosas ocasiones, estos tutoriales se utilizarían después de un tutorial introductorio o de un tutorial de refuerzo.

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. Su objetivo es distinto a los tutoriales anteriores, pues involucraría dos lenguajes de programación en lugar de uno solo.

Por último, con respecto al estudio de la aplicabilidad fuera del ámbito académico, nuestra conclusión es que el formato de tutorial interactivo es muy adecuado para dar formación online especializada. En estos casos, los tipos de tutoriales más apropiados serían los de extensión y los de traducción.

Resultados de la tarea 3: *Enunciado de funcionalidad y objetivos concretos*

Tras estudiar el estado del arte decidimos que nuestra aplicación debía tener una cierta funcionalidad mínima en una primera aproximación, además de plantearse diferentes mejoras que podrían aplicarse en el futuro. La funcionalidad mínima incluye una aplicación de escritorio que permite al usuario tanto aprender nuevos contenidos como comprobar sus conocimientos mediante preguntas relacionadas con la teoría previamente mostrada. Las preguntas pueden tener diferentes formatos, puesto que no es posible juzgar el aprendizaje de los usuarios con un solo tipo de pregunta, por lo que el formato para almacenar la información debe ser suficientemente flexible. La corrección de las respuestas debe hacerse también de manera flexible, pues unos criterios demasiado estrictos pueden hacer que los usuarios se frustren y pierdan interés en la aplicación. En cuanto a la presentación de los contenidos, se presentarán distintos temas entre los cuales se podrá seleccionar cualquiera de ellos sin restricciones.

Como futuras extensiones consideramos primero una mayor integración con la web. En primer lugar, sería interesante desarrollar una aplicación web, de tal manera que el usuario puede empezar a aprender sin necesidad de configurar su ordenador. Como segunda parte de esta integración web es interesante la integración con sistemas ya en funcionamiento, tanto a nivel más académico mediante el Campus Virtual (como explicamos en los resultados de la tarea 5) como a nivel social con la integración con redes sociales. La primera nos permitirá sincronizar las tareas de los estudiantes con el resto de sus notas en asignaturas ya virtualizadas; mientras que las redes sociales nos servirán para unificar los datos de los usuarios, mejorar la seguridad y potenciar la *gamificación*, haciendo que los estudiantes puedan compartir sus logros con otros. Otra mejora interesante es la expedición de certificados; aunque el Campus Virtual permite expedir certificados, consideramos que si el usuario utiliza además otras plataformas internacionalmente conocidas para estos propósitos, como es Mozilla Badges (https://wiki.mozilla.org/Badges/), esto potenciaría el uso fuera del ámbito académico. Desde el punto de vista de los contenidos, a largo plazo resultará importante añadir otros lenguajes de programación e incluso otros paradigmas, por lo que será necesario que nuestra plataforma tenga una estructura modular y fácilmente extensible. También puede ser interesante permitir el uso de aserciones, lo que uniría la programación con la especificación de programas, mejorando el entendimiento de ambos por parte de los estudiantes.

Teniendo estas ideas en cuenta, tenemos los siguientes objetivos:

1. Implementar un sistema de escritorio en Java. Java es un lenguaje suficientemente potente tanto para implementar una versión de escritorio como para una futura versión web.
2. La aplicación permitirá elegir entre diferentes temas y, dentro de cada tema, presentará la teoría y una serie de preguntas ordenadas. Para pasar a la siguiente pregunta será necesario contestar correctamente la actual, aunque sí será posible volver libremente a preguntas y explicaciones anteriores.

3. Desarrollar los contenidos usando YAML (http://www.yaml.org/). YAML es un lenguaje lo suficientemente flexible como para permitir definir tanto contenidos, las preguntas correspondientes y sus soluciones.
4. Definir al menos 2 tipos de preguntas que nos permitan comprobar diferentes competencias adquiridas por los usuarios. Estas preguntas serán del tipo (i) de opción múltiple y (ii) campo de texto que debe ser completado con código que cumpla una cierta funcionalidad, descrita en lenguaje natural.
5. La corrección de preguntas de tipo (ii) puede ser: (a) comprobación de sintaxis con respecto a la gramática del lenguaje, en los primeros estadios del aprendizaje o (b) comprobación de corrección, realizando varios test de caja negra usando un código correcto suministrado previamente.

Como trabajo futuro proponemos:

1. Extensión de la herramienta como aplicación web.
2. Integración de la herramienta con el Campus Virtual de la UCM.
3. Integración de la herramienta con redes sociales, en especial con Google+, dado que todos los estudiantes de la UCM tienen cuentas de Google, y con Facebook, la red social más usada.
4. Expedición de certificados.
5. Aumento de la gamificación, permitiendo acumular rachas o compartir puntuaciones para mostrar clasificaciones entre usuarios conectados usando las redes sociales indicadas arriba.

Resultados de la tarea 4: *Captura de requisitos para la implementación*

Para el desarrollo esta tarea se han tenido en cuenta los objetivos enunciados en la tarea 3, buscando estimar los requisitos necesarios para cumplirlos de la manera más óptima posible. Como se puede observar en dichos objetivos, éstos implican principalmente tareas de implementación. Por ello en esta tarea nos centramos en requisitos orientados a la implementación de un sistema que reúna toda la funcionalidad requerida.

Uno de los requisitos indispensables para la obtención del sistema será tener acceso al lenguaje de desarrollo elegido, i.e. Java. La elección de Java como lenguaje de implementación no es casual, ya que hace que este requisito se pueda cumplir fácilmente, dado que Java cuenta con una distribución gratuita (actualmente Java SE 8u71) a través de la página de Oracle[14]. En esta misma página se puede obtener, así mismo, la última versión del IDE de desarrollo NetBeans que puede ser de gran ayuda a la hora de desarrollar el sistema, dado la cantidad de herramientas de apoyo que ofrece al programador, tales como depuración, refactorización, interacción con el código, etc. Una alternativa a NetBeans sería Eclipse[15], de código libre y abierto, y de uso muy común entre los programadores de Java (y también de otros lenguajes) .

Dado que se va a utilizar documentos YAML para el desarrollo de contenidos, y estos han de ser procesados y mostrados a través de Java, necesitaremos alguna manera de conectar ambas tecnologías. La forma más sencilla sería a través de una librería para poder tanto analizar como construir (en caso que sea necesario para futuras ampliaciones del sistema) expresiones YAML. Podemos encontrar varias librerías que se podrían utilizar para cumplimentar este requisito como, por ejemplo, SnakeYAML[16].

De manera similar a YAML, el lenguaje elegido para dar formato al contenido es Markdown, y por tanto necesitaremos una manera de conectar Java con dicho lenguaje. Para ello, otra vez la mejor opción es hacer uso de librerías. En este caso buscamos librerías que sean capaces de producir HTML dada una cadena de texto en Markdown. Encontramos otra vez diversas opciones gracias a que Java es uno de los lenguajes más utilizados hoy en día. Un ejemplo de estas librerías sería *pegdown*[17], que cuenta con funciones tan apropiadas para nuestros objetivos como la función `markdownToHtml`.

Finalmente necesitaremos alguna tecnología para poder comprobar la corrección de las respuesta a preguntas con cajas de texto. Para las de caja negra, simplemente se necesitará disponer de un sistema capaz de ejecutar código del lenguaje que se esté aprendiendo (p.e. Python). Como alternativa se puede tener un servidor externo y enviar el código a este para ser compilado y ejecutado remotamente. En cuanto a las preguntas que exigen comprobación sintáctica, necesitaremos librerías o herramientas

[14] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[15] https://eclipse.org/downloads/
[16] https://bitbucket.org/asomov/snakeyaml
[17] https://github.com/sirthias/pegdown

externas que hagan este trabajo de la manera más simple posible y que ofrezcan resultados con un formato y una expresividad suficiente para poder ser utilizados por nuestro sistema. Un candidato muy interesante es el generador de analizadores sintácticos ANTLR[18]. Para ambos tipos de pregunta, estos requisitos serán necesarios de manera individual para cada lenguaje que se desee ofrecer para su aprendizaje a través del sistema.

Por lo que respecta a recursos humanos, como ya se ha mencionado en los anteriores apartados el sistema sería desarrollado como parte de un Trabajo Final de Grado. Es por ello que buscaremos un perfil de alumno que tenga soltura y comprobada habilidad en el desarrollo de aplicaciones Java. Estimamos que serán necesarios 2 alumnos para poder desarrollar el sistema en un margen de tiempo acorde a nuestras perspectivas y objetivos. En cuanto al tiempo estimado para el desarrollo del sistema, hemos calculado un tiempo de 3 horas por semana, durante un total de 8 meses, i.e. 32 semanas aproximadamente. Por tanto, tenemos que cada alumno dedicará 96 horas al desarrollo del sistema, requiriendo 192 horas en total si contamos a los 2 alumnos.

Finalmente los requisitos de hardware necesarios para el desarrollo del sistema son un ordenador por cada alumno. No se necesita ningún tipo de equipo de alto rendimiento ni hardware adicional para los objetivos inmediatos. Puede que para próximas iteraciones del proyecto sea necesario incorporar un servidor para agilizar parte del trabajo de instalación del sistema o para hacerlo disponible en web o través de dispositivos móviles. En este último caso, se incorporaría como requisito algún dispositivo móvil para poder hacer pruebas del sistema.

[18] http://www.antlr.org/

Resultados de la tarea 5: *Integración con otros sistemas*

En el apartado correspondiente a la tarea 3 se apuntó la necesidad futura de extender la herramienta como una aplicación web. Un problema de los proyectos de innovación de ámbito universitario que ofrecen una interfaz web es la dispersión de sistemas a los que se enfrenta el alumnado. Los distintos sistemas se almacenan en servidores separados, el acceso a cada uno de ellos se realiza mediante una dirección distinta, y cada uno requiere un nombre de usuario y contraseña distinto. Con el fin de no contribuir más a esta heterogeneidad de herramientas, pensamos que la ampliación futura de nuestro sistema de tutoriales a un sistema web debe realizarse teniendo en mente la integración con el Campus Virtual. Con esto se pretende conseguir, por un lado, una experiencia de usuario unificada con el resto de herramientas de aprendizaje UCM y, por otra parte, la posibilidad de integrar con otras herramientas del Campus Virtual como, por ejemplo, la gestión de calificaciones.

A este respecto, ya existe un módulo en el sistema *Moodle* que tiene un fin similar a nuestro proyecto: el módulo *Lesson*. Este módulo presenta al estudiante una serie de páginas HTML. Al final de cada una de ellas se le realiza al estudiante una pregunta de opción múltiple. Según la respuesta proporcionada por este último, el sistema saltará a otra página de la lección. En su expresión más sencilla, una lección de *Moodle* puede consistir en una secuencia lineal de páginas, con una única opción de *Continuar* que lleva a la siguiente página del tutorial. Sin embargo, el módulo *Lesson* presenta algunos inconvenientes que lo hacen inviable para nuestro proyecto:

- **Escasa variedad en las clases de preguntas disponibles**. El módulo *Lesson* soporta preguntas de elección múltiple, de respuesta larga y corta, de asociación y de respuesta numérica. Si una determinada lección requiere un programa como respuesta del estudiante, sería necesario el uso una pregunta de respuesta larga. Sin embargo, la corrección de las preguntas de respuesta larga es totalmente manual. Es decir, el estudiante envía su programa y espera a que el profesor lo evalúe para que el estudiante pueda continuar con el tutorial. Esto es inadmisible en un sistema de tutoriales interactivos, en los que se persigue una experiencia fluida en el paso del estudiante por las distintas lecciones. Para conseguir esta experiencia es necesario un mecanismo automático de corrección.
- **Escasa usabilidad de la interfaz del gestor de lecciones**. La herramienta de creación de lecciones de *Moodle* está pensada para profesores noveles que desean construir sus lecciones de manera interactiva. La interfaz de esta herramienta, aunque amigable, hace tediosa la creación de tutoriales extensos con preguntas complejas. Para este tipo de tutoriales resulta más cómoda y operativa la elección de un editor de texto plano.
- **Dificultades en la reutilización de lecciones.** Aunque *Moodle* permite utilizar la herramienta de copias de seguridad para exportar un curso entero, no es posible exportar lecciones individuales en este sistema. Puesto que nuestro objetivo es ejecutar los tutoriales interactivos en distintos sistemas (escritorio,

web, dispositivos móviles, etc.), la imposibilidad de exportar lecciones resulta una carencia significativa.

Estos inconvenientes sugieren la necesidad de implementar un nuevo módulo en el sistema *Moodle* cuya finalidad sea la ejecución de tutoriales interactivos especificados mediante YAML. Esto coincide plenamente con el enfoque de nuestra herramienta: un tutorial expresado en YAML puede ser ejecutado en distintos sistemas. Nuestra contribución a *Moodle* no sería más que otro sistema ejecutor de tutoriales. Además, el hecho de que las preguntas de los tutoriales incorporen una puntuación facilita la integración con la herramienta de gestión de calificaciones de *Moodle* (*Grades*). La puntuación obtenida por cada estudiante al ejecutar un tutorial puede verse reflejada en el libro de calificaciones de éste.

En el objetivo de integración con *Moodle*, existen otras vías alternativas al desarrollo de un módulo dentro de esta herramienta. *Moodle* proporciona una API basada en servicios web de tipo REST que permite añadir calificaciones a un estudiante. De este modo, cada vez que un estudiante ejecuta un tutorial localmente utilizando la aplicación Java creada a tal efecto, su calificación puede ser almacenada en *Moodle* a través de dicha API.