



Sistemas y Tecnologías Web (SyTW)

Aplicación para la elaboración y despliegue de cuestionarios.

Application for the generation and deployment of questionnaires.

Juan José Labrador González

Ingeniería Informática y de Sistemas

Escuela Superior de Ingeniería y Tecnología

Trabajo de Fin de Grado

La Laguna, 2 de julio de 2014

D. **Casiano Rodríguez León**, con N.I.F. 42.020.072-S profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna

C E R T I F I C A

Que la presente memoria titulada:

“Sistemas y Tecnologías Web. Aplicación para la elaboración y despliegue de cuestionarios.”

ha sido realizada bajo su dirección por D. **Juan José Labrador González**, con N.I.F. 78.729.778-L.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 2 de julio de 2014

Agradecimientos

La realización de esta asignatura de Trabajo de Fin de Grado no hubiera sido posible sin la ayuda de la Sección de Ingeniería Informática de la Escuela Superior de Ingeniería y Tecnología que ha llevado a cabo todos los trámites necesarios.

Por otra parte, agradecer a Gara Miranda Valladares su labor, como Coordinadora de la asignatura de Trabajo de Fin de Grado, en el asesoramiento a los tutores y alumnos sobre los trámites y documentos a realizar, así como las fechas límite para sus entregas.

Mención especial para mi familia y amigos, que han caminado junto a mí durante todo este tiempo y me han alentado para no rendirme y lograr mis objetivos.

Y por último, especialmente agradecer a Casiano Rodríguez León su labor como tutor del Trabajo de Fin de Grado. Junto a él he aprendido nuevas tecnologías, conceptos y procedimientos a la hora de implementar aplicaciones. Me ha aconsejado, animado, motivado y resuelto mis dudas de manera incansable en la realización del Trabajo de Fin de Grado. Ha sido placer gozar de su conocimiento y experiencia. Estoy seguro de que la experiencia adquirida me ayudará en mis próximas etapas profesionales.

Resumen

El objetivo de este trabajo ha sido integrar los conocimientos adquiridos durante los estudios del Grado y, en especial, del itinerario de Tecnologías de la Información, aproximando al alumno a la resolución de problemas de aplicaciones Web y favoreciendo el desarrollo de destrezas propias de la Ingeniería Web.

En este Trabajo de Fin de Grado se propone el desarrollo de una gema de Ruby que facilite la elaboración y despliegue de cuestionarios autoevaluables. Éstos, además de poseer las típicas preguntas tipo test de respuesta única y multirespuesta, cuentan con la posibilidad de añadir preguntas de programación. Los resultados de la evaluación de los cuestionarios serán visibles para los profesores.

Para su desarrollo se ha partido de un Lenguaje de Dominio Específico (DSL) implementado en Ruby por Armando Fox denominado 'Ruby-based Quiz Generator and DSL' (RuQL).

Palabras clave: Generación de cuestionarios, Lenguaje de Dominio Específico, DSL, Ruby, RuQL, Sinatra.

Abstract

Here should be the abstract in a foreing language...

Keywords: *Keyword1, Keyword2, Keyword2, ...*

Índice general

1. Introducción a RuQL	1
1.1. Estado del arte	1
1.2. ¿Qué es RuQL?	1
1.3. Objetivos y actividades a realizar	3
1.4. Tecnología usada	4
2. Desarrollo	7
2.1. Metodología usada	7
2.2. Problemas encontrados y soluciones	8
2.2.1. Entender el funcionamiento del código de la gema	8
2.2.2. Corregir tests y funcionalidades de la gema	8
2.2.3. Corrección de preguntas de Ruby en JavaScript	8
2.2.4. Corrección de preguntas de JavaScript en Ruby	8
2.2.5. Timeout corto entre peticiones del navegador al servidor	9
2.2.6. Problema de seguridad al evaluar código Ruby en el servidor	9
2.2.7. Lugar de almacenamiento de las respuestas de los alumnos	9
3. Resultados	11
3.1. Mejoras del DSL original	11
3.2. Creación del renderer HtmlForm	11
3.3. Creación del renderer Sinatra	13
4. Conclusiones y trabajos futuros	17
5. Summary and Conclusions	19
5.1. First Section	19
6. Presupuesto	21
6.1. Sección Uno	21
A. Glosario de términos	23
A.1. T	23

B. Guía de usuario final	25
B.1. Guía de usuario: Seccion 1	25
B.2. Guía de usuario: Seccion 2	25
C. Guía del desarrollador	26
C.1. Guía de desarrollador: Seccion 1	26
C.2. Guía de desarrollador: Seccion 2	26
D. Enlaces	27
D.1. Enlaces	27
Bibliografía	27

Índice de figuras

1.1. Ejemplo	6
------------------------	---

Índice de tablas

6.1. Tabla resumen de los Tipos	21
---	----

Capítulo 1

Introducción a RuQL

1.1. Estado del arte

1.2. ¿Qué es RuQL?

Esta aplicación de generación y despliegue de cuestionarios forma parte de una gema de Ruby creada por Armando Fox denominada 'Ruby-based Quiz Generator and DSL' (RuQL).

Inicialmente, esta gema permitía generar un cuestionario partiendo de un fichero **Ruby**, donde se redactaban las preguntas y respuestas haciendo uso de un **DSL**.

Poseía una serie de *renderers* que permitían generar los cuestionario en los siguientes formatos:

- Open EdX: formato *open source* listo para importar en plataformas de aprendizaje online como **EdX**.
- Versión HTML 5 imprimible: lista para ser impresa y rellenada por los usuarios. Se le podía pasar como argumento el path de una hoja de estilo para incorporarla al HTML de salida. Del mismo modo, se podía especificar el path de un template predefinido por el profesor de modo que las preguntas se renderizaran en el mismo.
- AutoQCM: formato listo para importar a **AMC** (*Auto Multiple Choice*), software libre que permite elaborar cuestionarios multirrespuesta.

Los tipos de preguntas que se podían especificar eran:

- **Preguntas de completar:** en las cuales los usuarios deben rellenar los espacios en blanco. Admitía respuestas de tipo *string* o *regexp*. Si existían múltiples espacios para rellenar, se especificaban las respuestas en forma de *array*, indicando además si el orden de las mismas influía. Permitía además especificar

respuestas falsas (*distractors*) con una explicación de la misma, de modo que si el alumno escribía dicho *distractor*, le apareciera la explicación de por qué esa respuesta era incorrecta.

[foto]

- **Preguntas multirrespuesta con una única respuesta correcta:** las clásicas preguntas tipo test. Se podía aleatorizar el orden de las respuestas definido en el fichero de preguntas y asignarles explicaciones a los *distractors* de manera individual o asignar una explicación general para todos los *distractors*.

[foto]

Especificando además la opción *raw* a la pregunta, permitía incrustar dicho texto entre etiquetas `<pre>` HTML.

[foto]

- **Preguntas multirrespuesta con una múltiples respuestas correctas:** iguales a las preguntas multirrespuesta de opción única con la diferencia de que existe más de una respuesta correcta.

[foto]

- **Preguntas de verdadero o falso:** caso particular de las preguntas multirrespuesta de opción única.

[foto]

Para todos los tipos de preguntas era posible especificar un comentario opcional que acompañaría al texto de la pregunta.

XXXXX PONER EL CÓDIGO DE EJEMPLO QUE USABA ARMANDO Y COMO SE GENERA EL HTML5

```
[~]$ cd tmp
[~/tmp]$ gem install ruql
Fetching: ruql-0.0.4.gem (100%)
Successfully installed ruql-0.0.4
1 gem installed
```

```
[~/tmp]$ mkdir example
[~/tmp]$ vi example.rb
[~/tmp]$ cat example.rb
```

```

quiz 'Example_quiz' do

  fill_in :points => 2 do
    text 'The capital of California is ____'
    answer 'sacramento'
  end

  choice_answer :randomize => true do
    text "What is the largest US state?"
    explanation "Not big enough." # for distractors without their own explanation
    answer 'Alaska'
    distractor 'Hawaii'
    distractor 'Texas', :explanation => "That's pretty big, but think colder."
  end

  select_multiple do
    text "Which are American political parties?"
    answer "Democrats"
    answer "Republicans"
    answer "Greens", :explanation => "Yes, they're a party!"
    distractor "Tories", :explanation => "They're British"
    distractor "Social Democrats"
  end

  select_multiple do
    text "Which are American political parties?"
    answer "Democrats"
    answer "Republicans"
    answer "Greens", :explanation => "Yes, they're a party!"
    distractor "Tories", :explanation => "They're British"
    distractor "Social Democrats"
  end

  truefalse 'The week has 7 days.', true
  truefalse 'The earth is flat.', false, :explanation => 'No, just looks that way'

```

1.3. Objetivos y actividades a realizar

Los objetivos propuestos para alcanzar en este Trabajo de Fin de Grado ha sido los siguientes:

- Conocer, dominar y practicar con lenguajes y herramientas de desarrollo de aplicaciones web en el **servidor**.
- Conocer, dominar y practicar con diferentes lenguajes y librerías en el **cliente**.

- Conocer, practicar y dominar de herramientas de **desarrollo dirigido por pruebas** (*TDD*) en entornos web.
- Conocer, practicar y dominar diferentes lenguajes de marcas y de estilo.
- Conocer, practicar y dominar diferentes mecanismos de despliegue.
- Conocer, practicar y familiarizarse con diferentes mecanismos de seguridad, autenticación y autorización.
- Conocer, practicar y dominar diferentes herramientas colaborativas y de **control de versiones** (*CVS*).
- Conocer, practicar y dominar **metodologías ágiles** de desarrollo de software.
- Desarrollar una aplicación web para la elaboración y despliegue de cuestionarios.

Y las actividades a realizar en el mismo, tal cual están descritas en la propuesta de **Proyecto de Trabajo de Fin de Grado** firmada por el director y el alumno en la actividad 2 de la asignatura, son las que se describen a continuación:

- Revisión bibliográfica.
- Realización de una aplicación web en la que:
 - Se proporciona soporte mediante una aplicación web a los procesos de evaluación.
 - Se proporciona/extiende un **Lenguaje de Dominio Específico** (*DSL*) para la elaboración de cuestionarios.
 - Se deberá considerar cómo resolver los problemas de seguridad asociados.
 - Redacción de la memoria.
- Preparación de las presentaciones.

1.4. Tecnología usada

Debido a que este Trabajo de Fin de Grado es una extensión de una gema de **Ruby**, se ha utilizado éste como lenguaje de programación.

Además, se ha hecho uso de un numeroso conjunto de gemas y de otras tecnologías enumeradas a continuación:

- HTML5: haciendo uso de funcionalidades como **Local Storage** y **Drag and Drop**.
- CSS3
- JavaScript
- Bootstrap
- jQuery
- XRegExp
- MathJax
- CodeMirror
- Mocha
- Chai
- Karma
- Sinatra
- GitHub
- Heroku
- OAuth 2.0
- Google Drive

[Iconos]

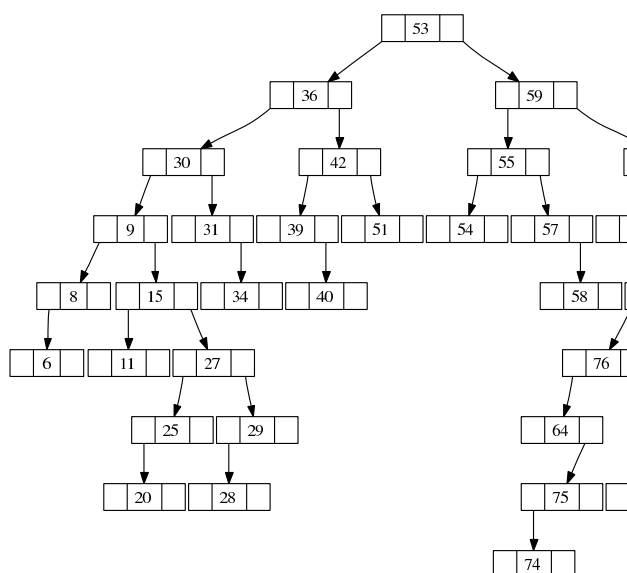


Figura 1.1: Ejemplo

Capítulo 2

Desarrollo

En el capítulo anterior se ha definido el Trabajo de Fin de Grado, especificado los objetivos y actividades a desarrollar y mencionado las tecnologías empleadas para su desarrollo. A continuación, se describirá la metodología de trabajo seguida y los problemas hallados durante el desarrollo junto con las soluciones alcanzadas.

2.1. Metodología usada

Se ha llevado a cabo una metodología de trabajo *ágil*, común en el campo de la Ingeniería Informática, con iteraciones semanales en las que se definían una serie de tareas u objetivos y que se presentaban la siguiente semana. De este modo, con la entrega de prototipos funcionales de la aplicación, se han ido testeando, corrigiendo y mejorando las funcionalidades, al mismo tiempo que detectando problemas no contemplados en las fases previas de diseño.

Esta metodología, además, ha propiciado la generación de ideas que se han traducido en nuevas características.

- git
- issues
- ramas
- experiencia en asignaturas
- reuniones
- ...

- objetivos
- intro a objetivos
- capitulo por renderer con el objetivo, desarrollo y resultados, problemas

2.2. Problemas encontrados y soluciones

A continuación se detallan los problemas encontrados durante la implementación del Trabajo de Fin de Grado y las soluciones encontradas para los mismos.

2.2.1. Entender el funcionamiento del código de la gema

Solución

Leer la documentación de la gema, generar cuestionarios de pruebas y estudiar el código fuente.

2.2.2. Corregir tests y funcionalidades de la gema

Solución

Tras realizar el correspondiente *fork* en GitHub para empezar a implementar mis modificaciones, ejecuté los tests de la gema original para comprobar la ausencia de fallos. Al finalizar, algunos tests fallaron por lo que decidí corregirlos. Del mismo modo, algunas gemas de testing existentes en el Gemfile presentaban incompatibilidades con las nuevas versiones de Ruby, por lo que también se corrigió.

Del mismo modo, las siguientes funcionalidades de la gema fueron corregidas ya que no funcionaban correctamente:

- La opción que permite indicar si el orden de las respuestas en las preguntas de completar espacios en blanco importa o no.
- La opción de añadir comentarios opcionales a los textos de las preguntas.
- La opción *raw* que permite incrustar el texto de las preguntas entre etiquetas `<pre>` HTML.
- La opción de explicación global para todos los *distractors* no funcionaba.

2.2.3. Corrección de preguntas de Ruby en JavaScript

bla, bla, bla

2.2.4. Corrección de preguntas de JavaScript en Ruby

bla, bla, bla

2.2.5. Timeout corto entre peticiones del navegador al servidor

bla, bla, bla

2.2.6. Problema de seguridad al evaluar código Ruby en el servidor

bla, bla, bla

2.2.7. Lugar de almacenamiento de las respuestas de los alumnos

La idea principal era almacenar todas las preguntas y respuestas de los alumnos en una base de datos pero, viendo la evolución que ha tenido la herramienta Google Drive y el aumento considerable de su uso por parte de docentes, decidí sustituir las tradicionales y siempre monótonas consultas a bases de datos por esta herramienta de almacenamiento que permite visualizar y administrar fácilmente toda la información.

Capítulo 3

Resultados

Tras explicar en el capítulo anterior la metodología empleada, los problemas encontrados durante la fase de desarrollo e implementación y las soluciones halladas, a continuación se detallarán todos los resultados obtenidos durante la fase de desarrollo del Trabajo de Fin de Grado: por un lado se encuentran las mejoras realizadas al DSL original de la gema y por otro, el enriquecimiento de la misma con dos nuevos *renderers*.

3.1. Mejoras del DSL original

Fruto del estudio del código y de ejecuciones sucesivas de la gema, se ha mejorado el funcionamiento de la misma efectuando diversos cambios:

- Corrección de errores en el funcionamiento de la gema (enumerados en el capítulo 2).
- Refactorización de código.
- Añadido manejo de excepciones tras errores de ejecución.
- Añadida la opción en línea de comandos *-version* para comprobar la versión de la gema.
- Añadida la opción en línea de comandos *-help* para ver la ayuda.

3.2. Creación del renderer HtmlForm

Este renderer permite generar un documento HTML5 con un formulario en el que se encuentran todas las preguntas listas para ser completadas desde el navegador. A continuación se enumerarán todas sus características:

- Añadida la opción de añadir uno o más JavaScripts al cuestionario que se generará.
- Añadida la opción de añadir uno o más ficheros de fragmentos de código HTML en la cabecera del cuestionario que se generará.
- Añadida la opción de añadir más de una hoja de estilo CSS al cuestionario que se generará.
- Añadida la opción de añadir un header y un footer personalizado al cuestionario que se generará. Se puede especificar como un *string* o indicar el path donde se encuentra el fichero que contiene dicho código HTML.
- Los textos de las preguntas admiten ahora caracteres HTML escapados.
- El cuestionario es capaz de renderizar expresiones escritas en **LaTeX**.
[foto]
- Las preguntas de completar permiten ahora respuestas numéricas y de código JavaScript.
[foto]
- Las espacios para rellenar las respuestas de las preguntas de completar se ajustan al tamaño de dicha respuesta.
[foto]
- Dos nuevas maneras simplificadas de escribir preguntas de completar.
[foto]
- Se han añadido dos nuevos tipos de preguntas:
 - Preguntas de **Drag and Drop**: para preguntas de completar y preguntas tipo test (de respuesta única y multirrespuesta)
[foto]
 - Preguntas de **programación**:
 - Este tipo de preguntas sólo admite código JavaScript debido a que la corrección de preguntas también tiene lugar en el navegador cliente.
 - Se creará un *textarea* de unas dimensiones definidas por defecto que también se pueden personalizar y se coloreará el código escrito para facilitar su lectura.

- La respuesta asignada a este tiempo de preguntas debe ser un código JavaScript que valide la respuesta introducida por el alumno. Este código puede escribirse en notación de *string* o especificar el path donde se encuentra el fichero que contiene dicho código.

[foto]

- Validación automática de respuestas mediante JavaScript que muestra la nota obtenida al instante.
- Almacenamiento local de las respuestas introducidas usando **Local Storage** de HTML5
- Menú contextual al hacer click derecho sobre el campo de respuesta para ver la respuesta correcta de dicha pregunta. Esta funcionalidad sólo está disponible para preguntas de completar, cuyas respuestas sean *strings*, *regexps* o numéricas.

[foto]

Para las preguntas tipo test existe un botón que marca las respuestas correctas.

[foto]

- Internacionalización: la gema comprueba el idioma del sistema para ofrecer la traducción adecuada al idioma del usuario Actualmente solo soporta inglés y español. Para cualquier otro idioma, se utiliza el inglés por defecto.

[foto]

Además, se han creado test para comprobar el funcionamiento del renderer (usando Spec) y el cuestionario generado, usando Mocha, Chai y Karma.

3.3. Creación del renderer Sinatra

Este otro renderer genera una aplicación Sinatra con todo lo necesario para ser desplegada en **Heroku** o ejecutar localmente.

1. Esta aplicación guardará todos los datos del cuestionario y de los alumnos permitidos para realizarlo en una hoja de cálculo de **Google Drive** en la cuenta del profesor.

2. Posteriormente, servirá el cuestionario a los alumnos especificados (durante un periodo de tiempo establecido previamente). Una vez que los alumnos hayan completado el cuestionario, sus respuestas se guardarán en una copia del cuestionario solo visible para los profesores a través de Google Drive y su nota se guardará además en la hoja de cálculo de Google Drive. De este modo, quedará constancia de la realización del mismo.
3. Los alumnos podrán reintentar el cuestionario todas las veces que deseen mientras se encuentre activo.

NOTA: Es responsabilidad del profesor facilitar la nota a los alumnos en el momento que estime oportuno.

En el fichero del examen que recibe como entrada la gema, además de definir las preguntas, se debe especificar a los usuarios que harán uso de la aplicación:

- Por un lado, se deben indicar a los profesores que podrán desplegar el examen o consultar las notas de los alumnos. Se indicará su email de Google en forma de *string*. En caso de ser múltiples profesores, se usará una notación de *array*.
[foto]
- Por otra parte, se deberán indicar los alumnos permitidos para realizar el cuestionario. Se puede usar un *Hash* con la información necesaria de ellos, o indicar el path de un fichero **CSV** con los datos de los mismos.
[foto]

El formato del fichero CSV debe ser del siguiente modo:

[tabla]

Para resolver el gran problema de la autenticación de usuarios, se hace uso de OAuth. De este modo, delegamos todo el servicio a Google y evitamos, por tanto, posibles brechas de seguridad que den lugar a suplantaciones de identidad o exposición de datos sensibles de los usuarios a terceras personas.

Además, es necesario especificar un fichero *config.yml* que contiene la ventana temporal en la cual estará disponible el cuestionario, el nombre del subdominio de Heroku que se desea usar para desplegar el cuestionario y la información relativa a Google Drive:

- Nombre de la hoja de cálculo donde se guardarán los datos de alumnos y preguntas y respuestas.

- Nombre de la carpeta que contendrá dicha hoja de cálculo.
- Path donde queremos que se cree la carpeta y la hoja de cálculo (si no existe alguna carpeta del path, se creará también).
- API keys necesarias para poder usar los servicios de Google, tanto la autenticación con OAuth como la escritura en Google Drive.

[foto de config.yml]

Finalmente, los ficheros que genera este renderer son:

- El código Ruby del servidor.
- Las vistas necesarias (incluyendo el cuestionario generado en HTML y un template ERB que se usará para crear las copias de los cuestionarios realizados por los alumnos).
- Un Gemfile con las dependencias necesarias.
- Un Rakefile para automatizar tareas (de ejecución y despliegue de la aplicación).
- Una carpeta denominada *config* con los datos de alumnos, profesores, las preguntas y respuestas y una copia del fichero *config.yml* del cual se hará una lectura de los parámetros. De este modo, evitamos que las variables existentes en el código contengan la información sensible.

[foto del árbol de directorios y ficheros]

Capítulo 4

Conclusiones y trabajos futuros

Este capítulo es obligatorio. Toda memoria de Trabajo de Fin de Grado debe incluir unas conclusiones y unas líneas de trabajo futuro

Capítulo 5

Summary and Conclusions

This chapter is compulsory. The memory should include an extended summary and conclusions in english.

5.1. First Section

Capítulo 6

Presupuesto

Este capítulo es obligatorio. Toda memoria de Trabajo de Fin de Grado debe incluir un presupuesto.

6.1. Sección Uno

Tipos	Descripcion
AAAA	BBBB
CCCC	DDDD
EEEE	FFFF
GGGG	HHHH

Tabla 6.1: Tabla resumen de los Tipos

Apéndice A

Glosario de términos

A.1. T

Términos

Lenguaje de Dominio Específico (DSL)

Ruby

Sinatra

RuQL

Bootstrap

Open EdX

Renderer

Fork

Gema

HTML5

CSS3

Desarrollo Dirigido por Test (TDD)

Control de Versiones (CVS)

Metodología ágil

GitHub

Template

Gemfile

Header

Footer

CSV

ERB

Apéndice B

Guía de usuario final

B.1. Guía de usuario: Seccion 1

Texto

B.2. Guía de usuario: Seccion 2

Texto

Apéndice C

Guía del desarrollador

C.1. Guía de desarrollador: Seccion 1

Texto

C.2. Guía de desarrollador: Seccion 2

Texto

Apéndice D

Enlaces

D.1. Enlaces

Enlaces

Bibliografía

- [1] C. Darwin, *The Origin Of Species*. November 1859.
- [2] “ACM LaTeX Style.” http://www.acm.org/publications/latex_style/.
- [3] D. H. Bailey and P. Swarztrauber, “The fractional Fourier transform and applications,” *SIAM Rev.*, vol. 33, no. 3, pp. 389–404, 1991.
- [4] A. Bayliss, C. I. Goldstein, and E. Turkel, “An iterative method for the Helmholtz equation,” *J. Comp. Phys.*, vol. 49, pp. 443–457, 1983.
- [5] “FACOM OS IV SSL II USER’S GUIDE, 99SP0050E5,” tech. rep., 1990.
- [6] C. Goldstein, “Multigrid methods for elliptic problems in unbounded domains,” *SIAM J. Numer. Anal.*, vol. 30, pp. 159–183, 1993.
- [7] P. Swarztrauber, *Vectorizing the FFTs*. Academic Press, New York, 1982.
- [8] S. Taásan, *Multigrid Methods for Highly Oscillatory Problems*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1984.