



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, 7 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 7 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: se centra en el aprendizaje y puesta en práctica de metodologías, aproximaciones, técnicas y herramientas para abordar la creciente complejidad de este tipo de aplicaciones en el marco de las metodologías ágiles.

En este Trabajo de Fin de Grado se propone el desarrollo de una gema Ruby que facilite la elaboración y despliegue de cuestionarios autoevaluables. Existe un buen número de plataformas y herramientas para la generación de este tipo de cuestionarios. Sin embargo, estas plataformas se limitan a la evaluación de cuestiones verdadero-falso, multi-elección, respuesta única y variaciones mas o menos simples de estas. Para la corrección de preguntas propias de las ramas de Ingeniería, como pueden ser la evaluación de código escrito por el alumno, es necesaria la figura del profesor para calificar dichas tareas. En nuestra propuesta, además de las típicas preguntas que proveen las plataformas mencionadas anteriormente, se cuenta con la posibilidad de añadir preguntas cuyas respuestas son evaluadas por programas escritos por el profesor.

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	1
1.1. Antecedentes y estado actual del tema	1
1.2. ¿Qué es RuQL?	2
1.2.1. Código de ejemplo para realizar un cuestionario HTML5 . . .	4
1.3. Objetivos y actividades a realizar	6
1.4. Tecnología usada	7
2. Desarrollo	9
2.1. Metodología usada	9
2.1.1. GitHub	9
2.1.2. Testing	12
2.1.3. Experiencia de usuario	12
2.2. Resultados	13
3. Mejoras del DSL original	15
3.1. Problemas encontrados y soluciones	16
3.1.1. Entender el funcionamiento del código de la gema	16
3.1.2. Corregir tests y funcionalidades de la gema	16
4. Generando HTML para el entrenamiento del alumno: HtmlForm renderer	17
4.1. Problemas encontrados y soluciones	27
4.1.1. Corrección de preguntas de Ruby en JavaScript	27
4.1.2. Alojamiento librería MathJax en un directorio de la gema	27
5. Generando un corrector de exámenes: Sinatra renderer	29
5.1. Problemas encontrados y soluciones	31
5.1.1. Timeout corto entre peticiones del navegador al servidor . . .	31
5.1.2. Lugar de almacenamiento de las respuestas de los alumnos . .	32
5.1.3. Problema de seguridad al evaluar código Ruby en el servidor .	32
6. Conclusiones y trabajos futuros	33

7. Summary and Conclusions	35
7.1. First Section	35
8. Presupuesto	37
8.1. Sección Uno	37
A. Glosario de términos	39
B. Guía de usuario	46
B.1. Instalación de la gema	46
B.2. HtmlForm renderer	48
B.2.1. Preguntas FillIn	49
B.2.2. Preguntas Drag and Drop FillIn	51
B.2.3. Preguntas Multiple Choice	51
B.2.4. Preguntas TrueFalse	51
B.2.5. Preguntas Drag and Drop Multiple Choice	51
B.2.6. Preguntas Select Multiple	51
B.2.7. Preguntas Drag and Drop Select Multiple	52
B.2.8. Preguntas Programming	52
B.2.9. Otras consideraciones	53
B.2.10. Generando el cuestionario	53
B.3. Sinatra renderer	53
B.3.1. Preguntas FillIn	53
B.3.2. Preguntas Programming	54
B.3.3. Parámetros adicionales	54
B.3.4. Dar de alta una aplicación en Google Developers Console . . .	57
B.3.5. Generando la aplicación	62
B.3.6. Ficheros y directorios generados por el renderer	62
B.3.7. Ejecutando la aplicación	63
B.3.8. Otras consideraciones	74
Bibliografía	74

Índice de figuras

1.1. Pregunta de completar simple	3
1.2. Pregunta de completar con distractor y explicación	3
1.3. Pregunta multirrespuesta con una única respuesta correcta	3
1.4. Pregunta multirrespuesta usando la opción <i>raw</i>	4
1.5. Pregunta multirrespuesta con una múltiples respuestas correctas	4
1.6. Pregunta de verdadero o falso	4
2.1. Captura del repositorio de la gema en GitHub	10
2.2. Ramas del repositorio propio	10
2.3. Pull Request aceptado y cerrado	11
2.4. Contribuciones hechas al repositorio original	11
2.5. Apartado de issues cerrados	12
4.1. Código para incluir un header personalizado	18
4.2. Código para incluir un footer personalizado	18
4.3. Ejemplo de pregunta con HTML escapado	18
4.4. Ejemplo de pregunta con código LaTeX	19
4.5. Ejemplo de pregunta con código LaTeX renderizada	19
4.6. Ejemplo de pregunta con respuesta numérica	20
4.7. Ejemplo de pregunta con respuesta de tipo JavaScript	20
4.8. Ejemplo de pregunta con respuestas de múltiples longitudes	20
4.9. Ejemplo de pregunta con respuestas de múltiples longitudes renderizada	21
4.10. Ejemplo del primer tipo de pregunta simplificada	21
4.11. Ejemplo del segundo tipo de pregunta simplificada	21
4.12. Ejemplo de pregunta de completar con Drag and Drop	22
4.13. Ejemplo de pregunta de completar con Drag and Drop renderizada	22
4.14. Ejemplo de pregunta tipo test de respuesta única con Drag and Drop	23
4.15. Ejemplo de pregunta tipo test de respuesta única con Drag and Drop renderizada	23
4.16. Ejemplo de pregunta de tipo test multirrespuesta con Drag and Drop	24
4.17. Ejemplo de pregunta de tipo test multirrespuesta con Drag and Drop renderizada	24
4.18. Ejemplo de pregunta de programación	24
4.19. Ejemplo de pregunta de programación renderizada	25

4.20. Ejemplo de pregunta corregida	25
4.21. Puntuación total	26
4.22. Ejemplo de mostrar respuesta correcta en pregunta de completar . . .	26
4.23. Ejemplo de mostrar respuesta correcta en pregunta tipo test	26
4.24. Mensaje en español de Local Storage borrado	27
5.1. Método para especificar los profesores permitidos en la aplicación . .	29
5.2. Método para especificar los alumnos permitidos en la aplicación . . .	29
5.3. Método para especificar la configuración de la aplicación	30
5.4. Listado de ficheros y directorios generado dentro del directorio <i>app</i> .	30
B.1. Ejemplo de fichero <i>config.yml</i> con la información necesaria	56
B.2. Botón para crear un nuevo proyecto	57
B.3. Elegir nombre para el proyecto	57
B.4. Apartado de APIs	58
B.5. Apartado de credenciales	59
B.6. Crear nuevo cliente ID	59
B.7. Pantalla de creación del cliente ID	60
B.8. Personalizar pantalla de permisos	61
B.9. Mensaje anunciando que el cuestionario no está abierto	63
B.10. Mensaje anunciando que el cuestionario está cerrado	63
B.11. Página de iniciar sesión	64
B.12. Página de activar cuestionario	64
B.13. Página de dar permisos a la aplicación	65
B.14. Mensaje anunciando que el cuestionario aún no está activado	65
B.15. Mensaje anunciando que el cuestionario ha activado	66
B.16. Cuestionario desplegado	66
B.17. Mensaje anunciando que ha finalizado la revisión del cuestionario . .	67
B.18. Listado de ficheros dentro de la carpeta del cuestionario en Google Drive	67
B.19. Hoja de cálculo con la información de los alumnos	68
B.20. Hoja de cálculo con la información de las preguntas	69
B.21. Hoja de cálculo con la información de las respuestas correctas	70
B.22. Mensaje anunciando que ha completado cuestionario	71
B.23. Hoja de cálculo con la puntuación por preguntas	72
B.24. Información actualizada del alumno que realizó el cuestionario	73
B.25. Listado de ficheros en Google Drive con la copia hecha por el alumno	73
B.26. Copia del cuestionario con las respuestas del alumno	74

Índice de tablas

8.1. Tabla resumen de los Tipos	37
---	----

Capítulo 1

Introducción

1.1. Antecedentes y estado actual del tema

La World Wide Web esta sujeta a un cambio continuo. La llegada de HTML5, la creciente importancia de AJAX y de la programación en el lado del cliente, las nuevas fronteras de la Web Semántica, la explosión de las redes sociales así como la llegada de las redes sociales federadas son ejemplos de esta tendencia general. Las aplicaciones web parecen evolucionar hacia entornos cada vez más ricos y flexibles en los que los usuarios pueden acceder con facilidad a los documentos, publicar contenido, escuchar música, ver vídeos, realizar dibujos e incluso jugar usando un navegador. Esta nueva clase de software ubicuo no cesa de ganar *momentum* y promueve nuevas formas de interacción y cooperación.

Dentro del ámbito educativo también se ha vivido una revolución, en forma y contenido, de impartir enseñanza. En Internet se puede encontrar, cada vez con más facilidad, contenidos de cualquier disciplina en múltiples formatos: blogs, videotutoriales, presentaciones, ejercicios resueltos, etc.

También están teniendo mucho éxito las plataformas de aprendizaje virtual ofreciendo, además de conocimiento sin la necesidad de estar físicamente presente en un aula, una serie de recompensas y medallas por ir obteniendo logros. Esta metodología se denomina **Gamificación** y está teniendo un impacto muy positivo en los usuarios de estas plataformas, ya que los anima a seguir usando estas herramientas de conocimiento.

Otro tipo de plataforma de aprendizaje virtual más orientada a universidades e institutos es **Moodle**. Está implantada en numerosos centros de todo el planeta. Es el complemento perfecto para enriquecer las enseñanzas impartidas físicamente con cuestionarios autoevaluativos y compartición de recursos adicionales. Además, facilita numerosas tareas a los docentes como la corrección y calificación de ejercicios.

Sin embargo, esta plataforma sólo se limita a la evaluación de cuestiones triviales. Para la corrección de preguntas propias de las ramas de Ingeniería, como pueden ser la implementación de código, es necesaria la figura del profesor para evaluar dichas tareas.

Otro problema que presenta es su difícil portabilidad. Estamos hablando de un tipo de plataforma que sigue un esquema **cliente-servidor**, que no es fácilmente migrable a otras máquinas.

Estas desventajas se ven resueltas, haciendo uso de **RuQL**, en la **Aplicación para la Elaboración y Despliegue de Cuestionarios** que se presentará en esta memoria. Dicha aplicación está destinada a profesores con ciertos conocimientos en el ámbito de la programación y de la informática, aunque la curva de aprendizaje no es excesiva para el resto del profesorado.

1.2. ¿Qué es RuQL?

Esta aplicación de generación y despliegue de cuestionarios hace uso 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.

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

Figura 1.1: Pregunta de completar simple

```
fill_in do
  text 'The visionary founder of Apple is ---'
  answer /^ste(ve|phen)\s+jobs$/
  distractor /^steve\s+wozniak/, :explanation => 'Almost, but not quite.'
end
```

Figura 1.2: Pregunta de completar con distractor y explicación

- **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*.

```
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
```

Figura 1.3: Pregunta multirrespuesta con una única respuesta correcta

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

```
choice_answer :raw => true do
  text %Q{What does the following code do:


```

 puts "Hello world!"

```


}
  distractor 'Throws an exception', :explanation => "Don't be an idiot."
  answer 'Prints a friendly message'
end
```

Figura 1.4: Pregunta multirrespuesta usando la opción *raw*

- **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.

```
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
```

Figura 1.5: Pregunta multirrespuesta con una múltiples respuestas correctas

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

```
truefalse 'The earth is flat.', false, :explanation => 'No, just looks that way'
```

Figura 1.6: Pregunta de verdadero o falso

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

1.2.1. Código de ejemplo para realizar un cuestionario HTML5

Instalamos la gema:

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

Creamos el fichero Ruby que contendrá las preguntas:

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

```
1 quiz 'Example_quiz' do
2
3   fill_in :points => 2 do
4     text 'The capital of California is ____'
5     answer 'sacramento'
6   end
7
8   choice_answer :randomize => true do
9     text "What is the largest US state?"
10    explanation "Not big enough." # for distractors without their own explanation
11    answer 'Alaska'
12    distractor 'Hawaii'
13    distractor 'Texas', :explanation => "That's pretty big, but think colder."
14  end
15
16  select_multiple do
17    text "Which are American political parties?"
18    answer "Democrats"
19    answer "Republicans"
20    answer "Greens", :explanation => "Yes, they're a party!"
21    distractor "Tories", :explanation => "They're British"
22    distractor "Social Democrats"
23  end
24
25  select_multiple do
26    text "Which are American political parties?"
27    answer "Democrats"
28    answer "Republicans"
29    answer "Greens", :explanation => "Yes, they're a party!"
30    distractor "Tories", :explanation => "They're British"
31    distractor "Social Democrats"
32  end
33
34  truefalse 'The week has 7 days.', true
35  truefalse 'The earth is flat.', false, :explanation => 'No, just looks that way'
36 end
```

Para generar el HTML versión imprimible, ejecutamos el siguiente comando:

```
[~/tmp]$ ruql example.rb Html5 > example.html
```

Si deseamos que nuestras preguntas se rendericen usando nuestro propio template, debemos especificarlo con la opción `-t`:

```
[~/tmp]$ ruql example.rb Html5 -t template.html.erb > example.html
```

Las especificaciones de cómo crear nuestro propio template se encuentran en el apartado de Guía de usuario (véase apartado del Apéndice B.1).

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 
- CSS3 
- JavaScript 
- Bootstrap 

- jQuery  write less, do more.
- XRegExp 
- MathJax 
- CodeMirror 
- Mocha 
- Chai 
- Karma 
- Sinatra 
- GitHub 
- Heroku 
- OAuth 2.0 
- Google Drive 

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 se introducirá a los resultados obtenidos para posteriormente describirlos por capítulos de manera detallada.

2.1. Metodología usada

Se ha llevado a cabo una metodología de trabajo *ágil*, común en el campo de la Ingeniería Informática, con reuniones semanales en las que se definían una serie de tareas u objetivos (iteración) 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.

2.1.1. GitHub

Para llevar a cabo esta metodología, se ha usado **GitHub** como herramienta de **Control de Versiones** (CVS). Todo el código implementado se alojaba en dicha herramienta, permitiendo así su cómoda modificación y actualización.

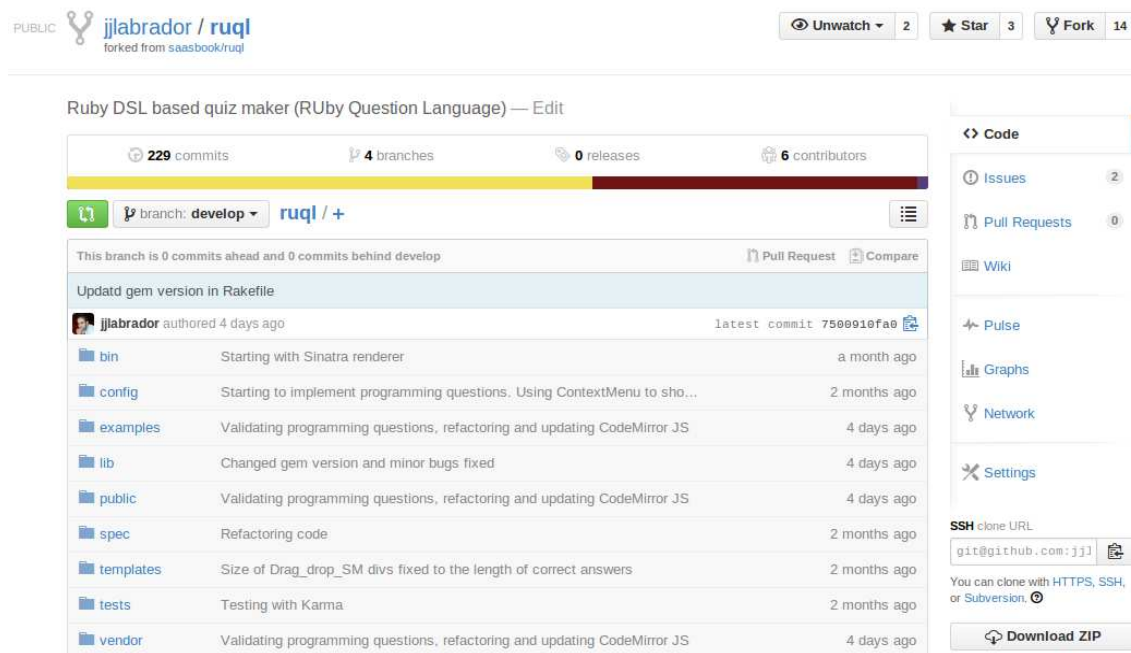


Figura 2.1: Captura del repositorio de la gema en GitHub

El trabajo se dividía en ramas, de modo que la versión estable de la aplicación (*rama master*) quedara aislada de la versión en desarrollo (*rama develop*).

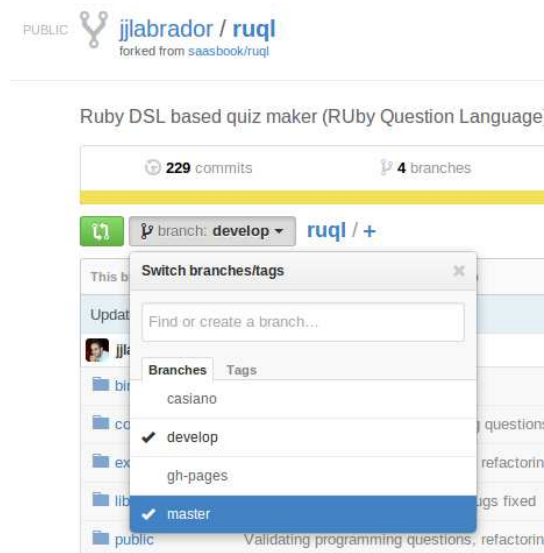


Figura 2.2: Ramas del repositorio propio

GitHub también ha servido para mantener contacto con el creador de la gema e indicarle mi intención de mejorar su gema en mi Trabajo de Fin de Grado. Él ha estado al tanto de mi progreso y tras solicitarle *Pull Requests* con mejoras y correcciones de su gema me ha convertido en colaborador de su repositorio.

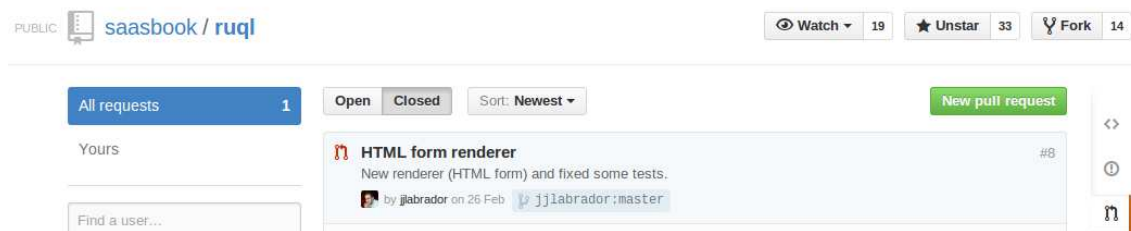


Figura 2.3: Pull Request aceptado y cerrado

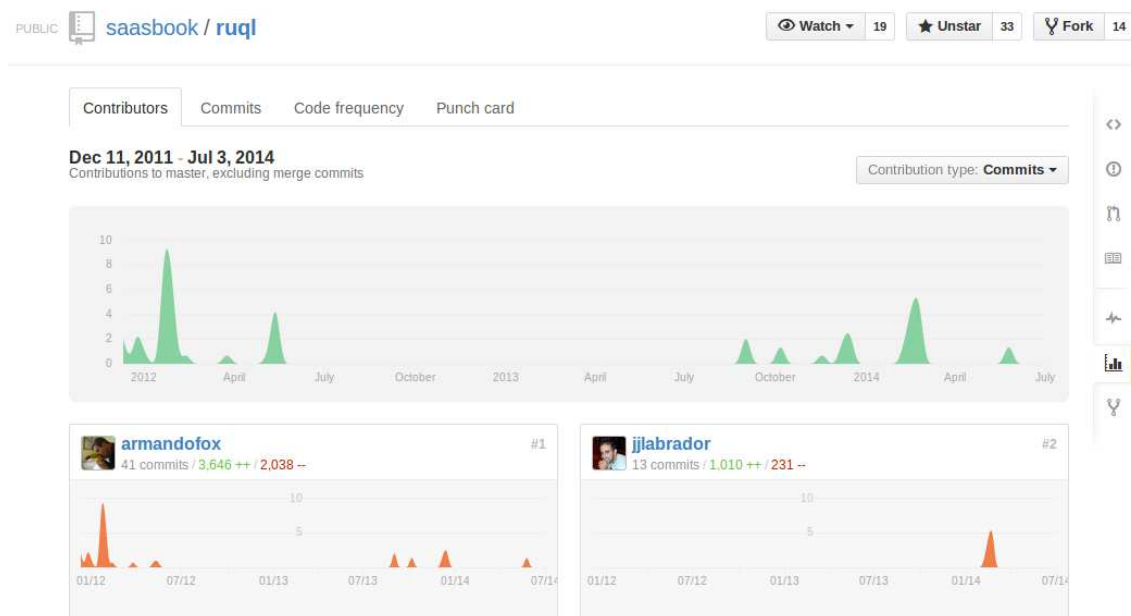


Figura 2.4: Contribuciones hechas al repositorio original

Las nuevas funcionalidades que han ido surgiendo, así como los problemas detectados, se anotaban en el apartado de *issues* con el fin de que quedara constancia de ello y se reflejara el estado en el que se encontraba cada uno.

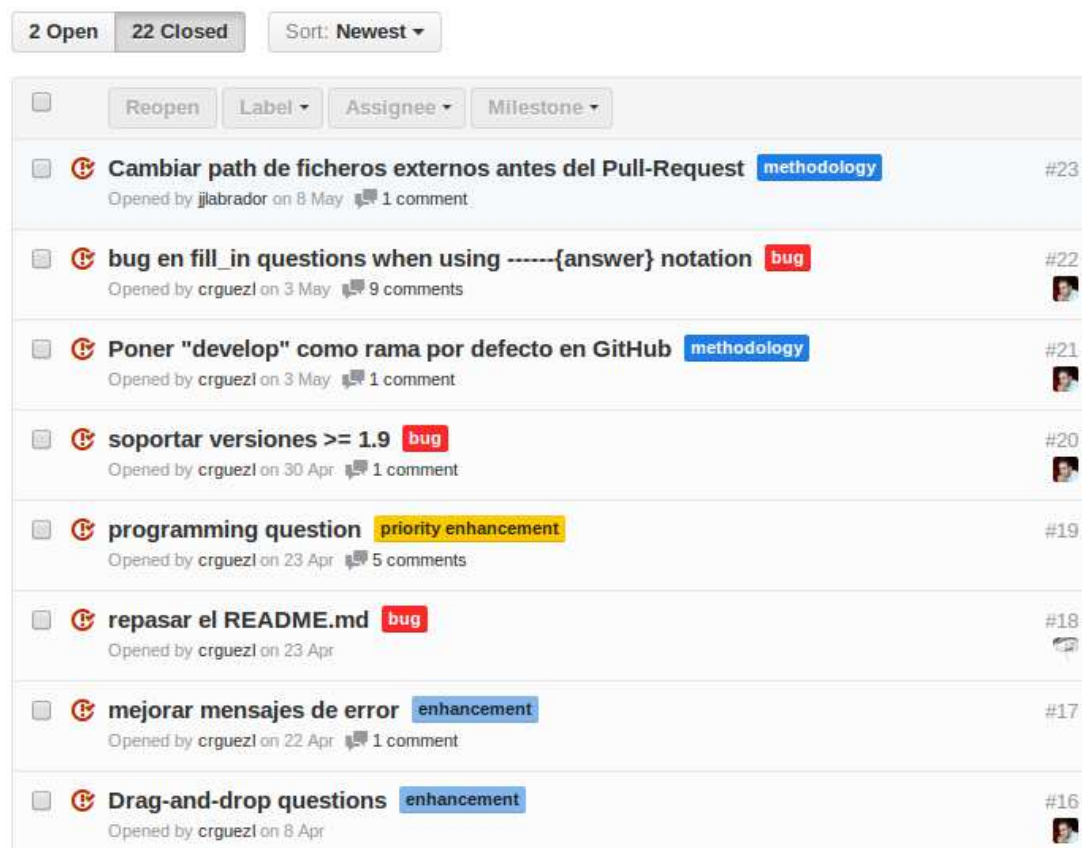


Figura 2.5: Apartado de issues cerrados

2.1.2. Testing

Dentro de la metodología también ha habido etapas de testing, haciendo uso de la metodología de Desarrollo Dirigido por Pruebas **TDD** (Test Driven Development). Se ha empleado la herramienta *Spec* para los test en Ruby y se han usado los frameworks *Mocha*, *Chai* y *Karma* para los test del HTML y el JavaScript.

2.1.3. Experiencia de usuario

Por otra parte, el tutor del Trabajo de Fin de Grado ha hecho pruebas reales usando los prototipos de la aplicación con alumnos de sus asignaturas. De este modo, se comprobaba el funcionamiento de la aplicación en un entorno real y se recibía un valioso feedback para mejorar en las siguientes iteraciones.

2.2. Resultados

Tras el desarrollo del Trabajo de Fin de Grado, se distinguen tres claros resultados: por un lado tenemos la **corrección de errores y mejoras de la gema original**. En segunda instancia, contamos con el renderer **HtmlForm**, válido para realizar una autoevaluación por parte del alumnado que le sirve, a modo de entrenamiento, para afrontar el examen final. Por último, contamos con el renderer **Sinatra**, que genera una aplicación Sinatra con todo lo necesario para su despliegue y puesta en funcionamiento.

Se explicarán cada uno de estos resultados en los siguientes capítulos de la memoria.

Capítulo 3

Mejoras del DSL original

Fruto del estudio del código y de ejecuciones sucesivas de la gema se detectaron una serie de errores de funcionamiento, por lo que antes de implementar mis mejoras propias era conveniente solucionar los problemas existentes. Además, se añadieron algunos cambios para mejorar el funcionamiento de la gema. A continuación se detallan las mejoras realizadas:

- Corrección de errores en el funcionamiento de la gema. Las opciones enumeradas a continuación 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.
- Refactorización de código, evitando la repetición del mismo en la medida de lo posible.
- 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.

3.1. Problemas encontrados y soluciones

A continuación se detallan los problemas encontrados durante la implementación de las mejoras del DSL original y las soluciones encontradas para los mismos.

3.1.1. Entender el funcionamiento del código de la gema

Solución

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

3.1.2. 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.

Capítulo 4

Generando HTML para el entrenamiento del alumno: HtmlForm renderer

Este renderer permitirá generar un documento HTML5 con un formulario en el que se encuentran todas las preguntas listas para ser completadas desde el navegador. Posee además todos los JavaScripts necesarios para la corrección automática de las mismas.

El objetivo de este renderer es proporcionar al alumno una serie de preguntas para que practique con vistas a afrontar un examen oficial. La principal ventaja que ofrece es la portabilidad: al ser un documento HTML puede alojarse en cualquier servidor o ejecutarse localmente desde un navegador sin tener que configurar nada previamente. Solo es necesario tener conexión a Internet ya que existe un JavaScript que necesita ser descargado mediante un **CDN**.

A continuación se enumerarán todas las características de este renderer:

- Permite añadir uno o más JavaScripts al cuestionario que se generará.

```
[~/tmp]$ ruql example.rb Html5 -j examples/test.js > example.html
```

- Permite añadir uno o más ficheros de fragmentos de código HTML en la cabecera del cuestionario que se generará.

```
[~/tmp]$ ruql example.rb Html5 -h examples/partial.html > example.html
```

- Permite la posibilidad de añadir más de una hoja de estilo CSS al cuestionario que se generará.

```
[~/tmp]$ ruql example.rb Html5 -c examples/style1.css
-c examples/style2.css > example.html
```

- Permite añadir un header y un footer personalizado al cuestionario que se generará. Se deberá indicar en el fichero Ruby. Se puede especificar como un *string* o indicar el path donde se encuentra el fichero que contiene dicho código HTML.

```
head : 'examples/header.html'
```

Figura 4.1: Código para incluir un header personalizado

```
foot : 'examples/footer.html'
```

Figura 4.2: Código para incluir un footer personalizado

- Los textos de las preguntas admiten ahora caracteres HTML escapados usando el método *escape*.

```
tag = '<a href="www.google.es"></a> '
fill_in do
  text "<i>Example of escaped HTML </i><br> #{escape(tag)}" + "is a ---- "
  answer /^link$/
end
```

Figura 4.3: Ejemplo de pregunta con HTML escapado

- El cuestionario es capaz de renderizar expresiones escritas en **LaTeX**.

```
fill_in do
  text %Q{
    Calculate the determinant of this matrix:
    $$\mathbf{A} = \begin{vmatrix}
    1 & 3 \\
    2 & 4
    \end{vmatrix}$$
    <br/>
    ----
  }
  answer -2
end
```

Figura 4.4: Ejemplo de pregunta con código LaTeX

[1 point] Calculate the determinant of this matrix:

$$\mathbf{A} = \begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix}$$

Figura 4.5: Ejemplo de pregunta con código LaTeX renderizada

- Las preguntas de completar permiten ahora respuestas numéricas y de código JavaScript.

```
fill_in do
  text %Q{
    Solve:
    
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

    when n = 5 and k = 2:
    ----
  }
  explanation "Not exactly"
  answer 10
  distractor 9
end
```

Figura 4.6: Ejemplo de pregunta con respuesta numérica

```
fill_in do
  text %Q{
    Diga dos números x = ---- e y = ---- que multiplicados den 100
  }
  answer JavaScript.new(%q{result = function(x,y) { return (x * y === 100); }})
end
```

Figura 4.7: Ejemplo de pregunta con respuesta de tipo JavaScript

- Las espacios para rellenar las respuestas de las preguntas de completar se ajustan al tamaño de dicha respuesta. Para ello basta especificar tantos guiones '-' como caracteres contenga la respuesta en el fichero Ruby. Es necesario un mínimo de tres guiones, de lo contrario, no se traducirán dichos guiones a etiquetas input HTML. En caso de querer mostrar mas de tres guiones sin que sean traducidos a inputs, se deben escapar usando un *backslash* ('\\')

```
fill_in do
  text "\\- \\- \\- The capital of Tenerife is ----- Cruz de ----- \\- \\- \\- "
  answer [/Santa/i, /Tenerife/i]
end
```

Figura 4.8: Ejemplo de pregunta con respuestas de múltiples longitudes

Figura 4.9: Ejemplo de pregunta con respuestas de múltiples longitudes renderizada

- Cuando en una pregunta existen numerosos huecos para escribir respuestas, en el array de respuestas podrá haber alguna de ellas que no mantiene su correspondencia con el índice que ocupa, por lo que se han añadido dos nuevas maneras simplificadas de escribir estas preguntas de completar:
 - Colocando la respuesta de la pregunta junto a sus guiones correspondiente. Para este tipo de escritura, solo se admiten respuestas de tipo *String*.

```
fill_in do
  text "The three stooges are -----{larry}, ----{moe}, and -----{curly}."
end
```

Figura 4.10: Ejemplo del primer tipo de pregunta simplificada

- Usando una notación de *Hash*: especificando una clave seguida de los guiones de la respuesta y definir su correspondencia en el método definido para escribir la respuesta.

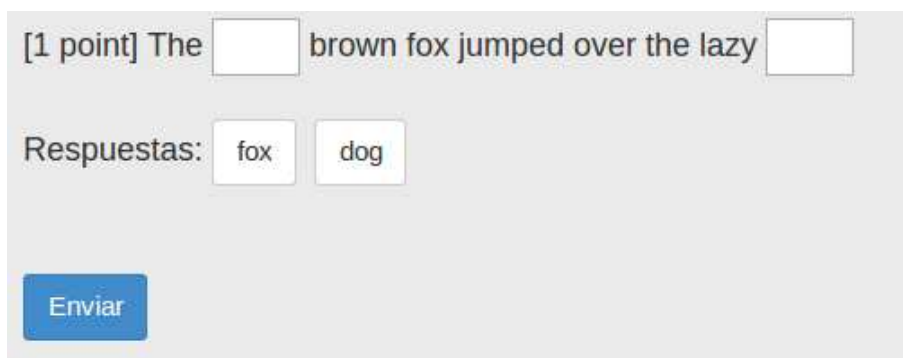
```
fill_in do
  text "The capital of Tenerife is -----{:santa} Cruz de -----{:tenerife}"
  answer :santa => /Santa/i, :tenerife => /Tenerife/i
end
```

Figura 4.11: Ejemplo del segundo tipo de pregunta simplificada

- 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).

```
drag_drop_fill_in do
  text "The ---- brown fox jumped over the lazy ----"
  answer ['fox', 'dog']
end
```

Figura 4.12: Ejemplo de pregunta de completar con Drag and Drop



[1 point] The brown fox jumped over the lazy

Respuestas:

Figura 4.13: Ejemplo de pregunta de completar con Drag and Drop renderizada

```
drag_drop_choice_answer do
  text "Relate these concepts"
  relation :Facebook => 'Mark Zuckerberg', :Twitter => 'Jack Dorsey'
end
```

Figura 4.14: Ejemplo de pregunta tipo test de respuesta única con Drag and Drop



The screenshot shows a web-based interface for a drag-and-drop question. At the top, it says "[1 point] Relate these concepts". Below this, there are two columns of draggable items. The left column contains two items: "Facebook" and "Twitter". The right column contains two items: "Mark Zuckerberg" and "Jack Dorsey". In the center, there are two empty rectangular boxes for the user to place the items. At the bottom left, there is a blue button labeled "Enviar".

Figura 4.15: Ejemplo de pregunta tipo test de respuesta única con Drag and Drop renderizada

```
drag_drop_select_multiple do
  text "Relate these concepts"
  relation :Ruby => ['Sinatra', 'Rails'], :JavaScript => 'jQuery'
end
```

Figura 4.16: Ejemplo de pregunta de tipo test multirrespuesta con Drag and Drop

Figura 4.17: Ejemplo de pregunta de tipo test multirrespuesta con Drag and Drop renderizada

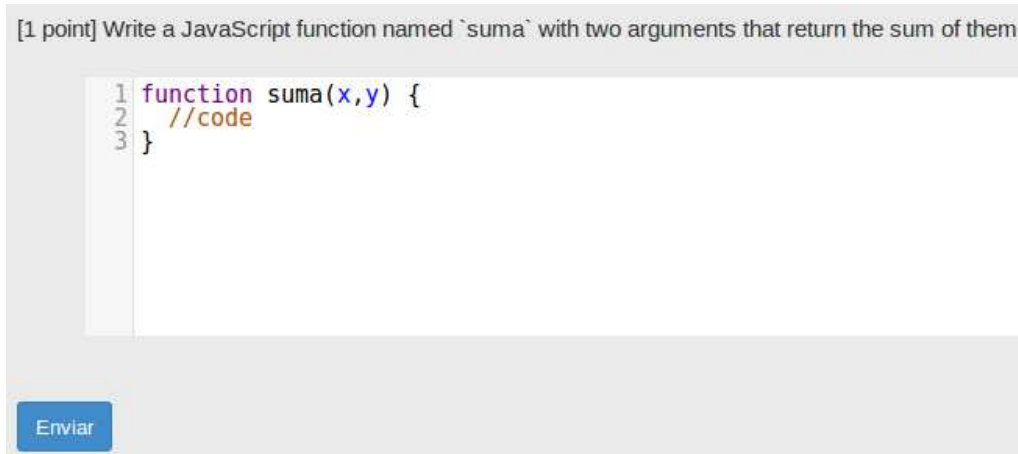
NOTA: en este último tipo de pregunta Drag and Drop, la longitud del contenedor donde se insertan las respuestas es el máximo de la suma de las longitudes de las respuestas correctas.

- 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.
 - La respuesta asignada a este tipo 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.

```
programming :language => 'JavaScript', :height => 150, :width => 800 do
  text %Q{Write a JavaScript function named `suma` with two arguments that return the sum of them}
  answer JavaScript.new(:'examples/test_suma.js')
end
```

Figura 4.18: Ejemplo de pregunta de programación

- 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.



[1 point] Write a JavaScript function named `suma` with two arguments that return the sum of them

```
1 function suma(x,y) {  
2   //code  
3 }
```

Enviar

Figura 4.19: Ejemplo de pregunta de programación renderizada

- Validación automática mediante JavaScript de las preguntas que han sido rellenas. Para corregir respuestas de tipo *Regexp* se utiliza **XRegExp**, mucho más potente que las expresiones regulares proporcionadas en JavaScript nativo. Esta validación muestra:
 1. La nota obtenida en esa pregunta.
 2. La explicación asociada a esa respuesta (si se ha especificado en el fichero Ruby).



[1 point] What is the largest US state?

☐ Alaska

☒ Texas **Incorrecto - That's pretty big, but think colder.**

☐ Hawaii

0.00/1.00 puntos

Mostrar respuesta Enviar

Figura 4.20: Ejemplo de pregunta corregida

3. La puntuación total al final de la página.



Figura 4.21: Puntuación total

- 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.



Figura 4.22: Ejemplo de mostrar respuesta correcta en pregunta de completar

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

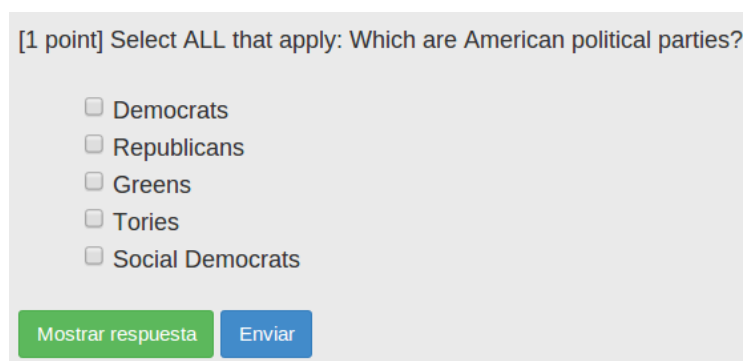


Figura 4.23: Ejemplo de mostrar respuesta correcta en pregunta tipo test

- Internacionalización: la gema comprueba el idioma del sistema para ofrecer la traducción adecuada al idioma del usuario de diversos mensajes como la corrección de cada pregunta (correcto/incorrecto) o la alerta que anuncia que se ha borrado el Local Storage. Actualmente solo soporta inglés y español. Para cualquier otro idioma, se utiliza el inglés por defecto.

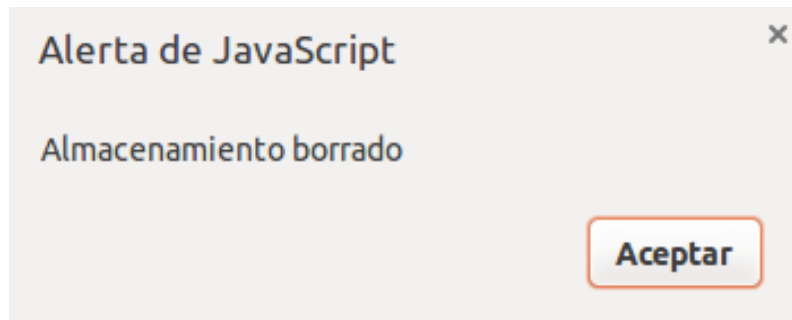


Figura 4.24: Mensaje en español de Local Storage borrado

La sintaxis para ejecutar este renderer es la siguiente:

```
[~/tmp]$ ruql example.rb HtmlForm -t templates/htmlform.html.erb > example.html
```

4.1. Problemas encontrados y soluciones

4.1.1. Corrección de preguntas de Ruby en JavaScript

Se ha tratado de usar **Opal** para traducir preguntas escritas en Ruby a JavaScript y poder validarlas desde el navegador. Se consiguió en parte, pues no fue posible obtener el código de una *Lambda* o un *Proc*, ya que Ruby los evaluaba antes de llegar al JavaScript.

Solución

Permitir solamente preguntas de código JavaScript en este renderer.

4.1.2. Alojar librería MathJax en un directorio de la gema

Todos los JavaScript y CSS de terceras partes se encuentran alojadas en un directorio denominado *vendor*. Para los JavaScripts y CSS propios existe otro directorio

llamado *public*. Estos ficheros se insertan en el HTML generado, de modo que no es necesario manejar varios ficheros junto con el HTML.

Sin embargo, el framework **MathJax**, usado para insertar texto en LaTeX en el HTML, no se ha podido insertar en el mismo. Consta de un amplio número de ficheros y cada uno de ellos es una dependencia de otro. Además, existen numerosas imágenes por lo que no ha sido posible insertarlo en el HTML de salida.

Solución

Hacer uso de un **CDN** para utilizar este framework.

Capítulo 5

Generando un corrector de exámenes: Sinatra renderer

Este otro renderer genera una aplicación Sinatra con todo lo necesario para ser desplegada en **Heroku** o ejecutar localmente. Se hace uso de **Google Drive** para almacenar una copia del cuestionario y para alojar las preguntas y respuestas de los alumnos.

Para usar este renderer es necesario añadir especificar algunos parámetros más en el fichero Ruby que contiene el cuestionario. Estos parámetros se enumeran a continuación:

1. La dirección de correo en **Gmail** del profesor. Es posible especificar más de una dirección usando una notación de *Array*.

```
teachers "jjlabradorglez@gmail.com"
```

Figura 5.1: Método para especificar los profesores permitidos en la aplicación

2. El path de un fichero **CSV** con los datos de los alumnos.

```
students : "examples/students.csv"
```

Figura 5.2: Método para especificar los alumnos permitidos en la aplicación

3. El path de un fichero de configuración denominado *config.yml* que contiene información del cuestionario.

```
config : "examples/config.yml"
```

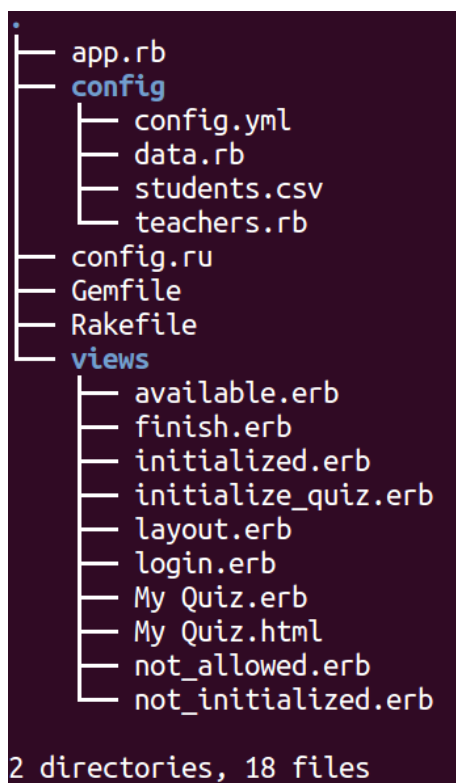
Figura 5.3: Método para especificar la configuración de la aplicación

Para saber cómo rellenar estos parámetros, véase Apéndice B.3.3.

La sintaxis para ejecutar este renderer es la siguiente:

```
[~/tmp]$ ruql example.rb Sinatra -t templates/htmlform.html.erb
```

Tras su ejecución, se creará una carpeta llamada *app* en el directorio donde nos encontremos. Este directorio contendrá lo siguiente:



```
.
├── app.rb
├── config
│   ├── config.yml
│   ├── data.rb
│   ├── students.csv
│   └── teachers.rb
├── config.ru
├── Gemfile
├── Rakefile
├── views
│   ├── available.erb
│   ├── finish.erb
│   ├── initialized.erb
│   ├── initialize_quiz.erb
│   ├── layout.erb
│   ├── login.erb
│   ├── My Quiz.erb
│   ├── My Quiz.html
│   ├── not_allowed.erb
│   └── not_initialized.erb
└── 2 directories, 18 files
```

Figura 5.4: Listado de ficheros y directorios generado dentro del directorio *app*

Para obtener información de los directorios y ficheros generados, véase Apéndice B.3.6.

La aplicación resultante funciona del siguiente modo:

1. Una vez ejecutada, el profesor deberá autenticarse con su cuenta de Gmail y activar el cuestionario. Hecho esto, se creará en su **Google Drive** una carpeta que contendrá una copia de examen y una hoja de cálculo con las preguntas y respuestas correctas del cuestionario. También guardará la información de los usuarios.
2. Una vez activado, los alumnos podrán acceder a realizar el mismo. Cuando lo finalicen, se generará una copia de su examen en la carpeta de Google Drive del profesor y se escribirá automáticamente su hoja de cálculo para añadir la nota que ha sacado el alumno individualmente en cada pregunta y de manera global. 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. Éste dejará de estar activo cuando finalice la fecha límite establecida por el profesor en el fichero de configuración. Al reintentar el cuestionario, se actualizará Google Drive con las nuevas calificaciones del alumno.

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

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. Para poder usarlo es necesario dar de alta en **Google Developers Console** una aplicación. Este proceso se explicará en el apartado del Apéndice B.3.4.

5.1. Problemas encontrados y soluciones

5.1.1. Timeout corto entre peticiones del navegador al servidor

Al inicializar el cuestionario, el servidor tarda en responder ya que es necesario escribir en Google Drive una cantidad de datos considerable. Cuando termina la escritura, nos debe mostrar una vista en la que se puede comprobar que el cuestionario ha sido inicializado correctamente. Esta vista sólo se muestra una vez y únicamente al profesor que inicializó el cuestionario. El resto de ocasiones, se nos redirige

al cuestionario en sí. Sin embargo, mientras se está escribiendo en Google Drive el navegador interpreta que el servidor tarda en responder, por lo que le manda una nueva petición y éste la despacha en cuanto finaliza la escritura en Google Drive. El problema reside en que atiende a esta nueva petición y, al ver que el cuestionario ya ha sido inicializado, nos redirige al cuestionario, saltándose la vista que debería aparecer la primera vez.

Solución

Usar como servidor **WeBrick** en lugar del que ejecuta Sinatra por defecto (**Thin**). WeBrick no permite múltiples peticiones por lo que se nos muestra la vista adecuada al inicializar el cuestionario.

5.1.2. Lugar de almacenamiento de las respuestas de los alumnos

Estaba claro que la información de los cuestionarios había que guardarla en algún lado. La idea principal era almacenar todas las preguntas y respuestas de los alumnos en una base de datos, pero preocupaba el hecho de que ésta se alojara en un servidor desconocido y que, por algún motivo, se perdiera dicha información sensible.

Solución

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. Además de ser segura, la información reside en la cuenta del profesor y éste la puede exportar donde desee cómodamente.

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

En esta primera versión del renderer, la aplicación generada evalúa el código introducido por el alumno sin ningún mecanismo de protección, por lo que introduciendo algún script comprometedor en el campo de alguna respuesta se podría comprometer la información de la aplicación.

Este es un aspecto importante a tratar en próximas mejoras del renderer.

Capítulo 6

Conclusiones y trabajos futuros

Desde hace unos años hasta ahora, ha tenido lugar un enorme crecimiento de las plataformas de aprendizaje online. Cada vez cuentan con más adeptos y las instituciones de enseñanza saben que incorporarlas a sus sistemas educativos es clave para ofrecer un servicio puntero y de calidad.

Ésto es lo que se pretende con la herramienta obtenida tras la realización de este Trabajo de Fin de Grado: que sea posible su implantación dentro del marco académico de la Universidad de La Laguna, partiendo de la premisa de que nos estamos adentrando en una época en la que estas herramientas de aprendizaje online seguirán evolucionando y teniendo un papel importante en la educación.

En este trabajo se proporciona/extiende un **Lenguaje de Dominio Específico** (*DSL*) para la elaboración y evaluación de cuestionarios que ofrece varias ventajas con respecto a otras herramientas equivalentes:

1. Respuestas que pueden ser evaluadas mediante expresiones regulares extendidas
2. Respuestas que pueden ser evaluadas mediante código arbitrario definido por el profesor
3. Generación de código para un número amplio de plataformas y formas de uso: Generación de HTML *standalone* para retroalimentación y entrenamiento del alumno, XML para EdX, ...,
4. Escalabilidad: Posibilidad de implantar soporte para cualquier plataforma o forma de uso que se desee mediante el uso de renderers
5. Generación de una aplicación de evaluación completa
6. XXXX aumentar como quieras

Los objetivos marcados al comienzo de la asignatura han sido cumplidos, además de los objetivos generales establecidos en la guía docente de la asignatura y la adquisición de las competencias generales y específicas descritas en el mismo. Sin embargo, el desarrollo no finaliza aquí. Aún queda trabajo por realizar para poder garantizar un eficaz funcionamiento y rendimiento de esta herramienta. El estado final de la misma puede ser el punto de partida para próximos Trabajos de Fin de Grado.

Así pues, las principales líneas de desarrollo a continuar serían las enumeradas a continuación:

- Resolver los problemas de seguridad relacionados con evaluar el código escrito por los alumnos.
- Dar soporte a preguntas con respuestas de código en otros lenguajes de programación.
- Ofrecer una alternativa de despliegue distinta a Heroku, como podría ser un servidor dedicado ofrecido por la Universidad de La Laguna.

Capítulo 7

Summary and Conclusions

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

7.1. First Section

Capítulo 8

Presupuesto

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

8.1. Sección Uno

Tipos	Descripcion
AAAA	BBBB
CCCC	DDDD
EEEE	FFFF
GGGG	HHHH

Tabla 8.1: Tabla resumen de los Tipos

Apéndice A

Glosario de términos

A

API: (*Application Programming Interface* o Interfaz de Programación de Aplicaciones). Conjunto de funciones y procedimientos o métodos que ofrece cierta librería para ser utilizados por otro software como una capa de abstracción.

AJAX: acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML). Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

B

Bootstrap: framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript opcionales adicionales.

C

CDN: (*Content Delivery Network* o Red de Entrega de Contenidos). Red superpuesta de computadoras que contienen copias de datos, colocados en varios puntos

de una red con el fin de maximizar el ancho de banda para el acceso a los datos de clientes por la red. Un cliente accede a una copia de la información cerca del cliente, en contraposición a todos los clientes que acceden al mismo servidor central, a fin de evitar embudos cerca de ese servidor.

Chai: librería de aserciones TDD que puede incluirse con algún framework de test JavaScript (como, por ejemplo, Mocha).

Cliente-Servidor: modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

CodeMirror: editor de texto implementado en JavaScript para usar desde el navegador.

CVS: (*Concurrent Versioning System* o Sistema de Control de Versiones). Aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.

CSS3: (*Cascading Style Sheets* o Hoja de Estilos en Cascada). Lenguaje de hojas de estilo utilizado para describir el aspecto y el formato de un documento escrito en un lenguaje de marcas, como HTML. CSS3 es la versión más reciente de este lenguaje.

CSV: (*Comma-Separated Values*). Tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

D

DSL: (*Domain Specific Language* o Lenguaje de Dominio Específico). En desarrollo de software, un lenguaje de dominio específico es un lenguaje de programación o especificación de un lenguaje dedicado a resolver un problema en particular, representar un problema específico y proveer una técnica para solucionar una situación particular.

E

ERB: (*Embedded Ruby*). Sistema de plantillas que incrusta código Ruby en un documento HTML. Está incluido en el núcleo de Ruby.

F

Footer: texto que se encuentra separado del texto principal del documento y que aparece al final de la página del mismo.

Fork: en desarrollo de software, el fork es la acción de realizar una copia exacta de un proyecto a partir de otro y contiene todo el código fuente listo para empezar un desarrollo independiente haciendo uso del mismo.

Framework: (marco de trabajo). En el desarrollo de software, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

G

Gamificación: uso de técnicas y dinámicas propias de los juegos y el ocio en actividades no recreativas, como puede ser en actividades de aprendizaje.

Gema: programa o biblioteca escrito en Ruby que proporciona una serie de métodos para extender la funcionalidad del lenguaje Ruby.

Gemfile: fichero de Ruby en el que se especifican todas las gemas de las que depende un proyecto o programa para que funcione correctamente.

Google Drive: servicio de alojamiento de archivos accesible desde su página web mediante ordenadores o mediante aplicaciones móviles.

GitHub: forja para alojar proyectos utilizando el Sistema de Control de Versiones **Git**.

H

Header: texto que se encuentra separado del texto principal del documento y que aparece al comienzo de la página del mismo.

Heroku: plataforma como servicio (*PaaS*) de computación en la Nube que permite, entre otras cosas, alojar aplicaciones web.

HTML5: (*HyperText Markup Language*). Lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web definiendo una estructura básica y un código para la definición del contenido de la misma.

J

JavaScript: lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

jQuery: librería de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

K

Karma: librería JavaScript que proporciona un entorno configurable y personalizable de testing que permite la integración continua de los mismos.

L

Lambda: objeto en Ruby que representa un bloque, es decir, un trozo de código Ruby asociado con la invocación de un método. Las lambdas también reciben el nombre de Procs.

M

MathJax: librería de JavaScript que permite mostrar signos y elementos matemáticos en los documentos HTML.

Metodología ágil: conjunto de métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizados y multidisciplinarios. Se caracterizan además por la minimización de riesgos desarrollando software en iteraciones cortas de tiempo.

Mocha: framework de test para código JavaScript.

Moodle: aplicación web de tipo Ambiente Educativo Virtual. Es un sistema de gestión de cursos, de distribución libre, que ayuda a los profesores a crear comunidades de aprendizaje en línea. Este tipo de plataformas tecnológicas también se conoce como LCMS (*Learning Content Management System*).

O

OAuth 2.0: (*Open Authorization*): protocolo abierto que permite la autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, web y móviles.

Opal: gema de Ruby que compila código escrito en Ruby a código JavaScript.

Open EdX: iniciativa de código abierto creada con el fin de crear la plataforma de aprendizaje online de nueva generación, que proporcionará una educación de mejor calidad a los estudiantes de todo el mundo.

P

Proc: objeto en Ruby que representa un bloque, es decir, un trozo de código Ruby asociado con la invocación de un método. Las procs también reciben el nombre de Lambdas.

R

Renderer: generador de cuestionarios en un determinado formato, como por ejemplo, HTML5.

Ruby: lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto. Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU.

RuQL: (*Ruby-based Quiz Generator and DSL*). Gema de Ruby que permite la elaboración de cuestionarios a partir de un fichero Ruby escrito usando un DSL propio de la gema. Este DSL se ha usado como base para elaborar este Trabajo de Fin de Grado.

S

Sinatra: DSL que permite la rápida creación de aplicaciones web en Ruby.

T

TDD: (*Test-Driven Development* o Desarrollo Dirigido por Pruebas). Práctica de programación que involucra otras dos prácticas: escribir las pruebas primero (*Test First Development*) y Refactorización de código (*Refactoring*).

Template: plantilla HTML en la cual se renderizará (incrustará) el contenido del cuestionario.

Thin: gema de Ruby que implementa un servidor web rápido y ligero.

W

WeBrick: librería de Ruby que proporciona un sencillo servidor web. Está incluido en el núcleo de Ruby.

Web semántica: idea de añadir metadatos semánticos y ontológicos a la World Wide Web. Esas informaciones adicionales, que describen el contenido, el significado y la relación de los datos, se deben proporcionar de manera formal, para que sea posible evaluarlas automáticamente por máquinas de procesamiento. El objetivo es mejorar Internet ampliando la interoperabilidad entre los sistemas informáticos usando **agentes inteligentes**, es decir, programas en las computadoras que buscan información sin necesidad de interacción humana.

World Wide Web: (WWW). Sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y navega a través de esas páginas usando hiperenlaces.

X

XRegExp: librería JavaScript que proporciona y extiende la funcionalidad de las **expresiones regulares** nativas de JavaScript.

Apéndice B

Guía de usuario

El objetivo de esta guía de usuario es proporcionar a los usuarios un ejemplo para la puesta a punto y ejecución de las funcionalidades implementadas en el gema RuQL durante el Trabajo de Fin de Grado.

B.1. Instalación de la gema

Para instalar la gema, basta con ejecutar el siguiente comando:

```
[~]$ gem install ruql
```

Si deseamos ejecutar la gema sin instalarla, debemos hacer lo siguiente:

- Descargar el código desde GitHub.
- Ejecutar el siguiente comando desde la raíz del proyecto:

```
[~]$ ruby -Ilib bin/ruql [argumentos]
```

Para consultar la ayuda sobre qué argumentos opcionales recibe cada `render`, basta con escribir lo siguiente:

```
[~]$ ruql --help
```

si tenemos la gema instalada en nuestra máquina o

```
[~]$ ruby -Ilib bin/ruql --help
```

si la ejecutamos desde el código fuente.

Entre ellos destacan las opciones de añadir hojas de estilo (*-c*), JavaScripts (*-j*) y especificar un propio template (*-t*) de **ERB**. Si se desea emplear un template

propio, éste debe contener una serie de variables obligatoriamente para que se inserten los recursos necesarios y el renderer funcione correctamente. A continuación se muestra un ejemplo de template con dichas variables necesarias. Además, será posible añadirle un header y un footer personalizado. De este modo no sería necesario elaborar un template completamente nuevo cada vez. Para ello basta con indicarlo en nuestro fichero Ruby. Los métodos *head* y *foot* admiten tanto un path (escrito como un **símbolo**) de donde se encuentre el código HTML o un **string** con el propio HTML:

```
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title><%= quiz.title %></title>
    <!-- Bootstrap CSS -->
    <%= @bootstrap_css %>
    <style type="text/css" media="all">
      /* Custom CSS */
      /* .....*/
      /* Inputs size */
      <%= @sass %>
    </style>
    <!-- jQuery ContextMenu CSS -->
    <%= @context_menu_css %>
    <!-- Any CSS included by the user -->
    <%= @css_custom %>
    <!-- Mathjax -->
    <%= @mathjax %>
    <!-- CodeMirror -->
    <%= @codemirror %>
  </head>
  <body>
    <div class="container">
      <% if (quiz.get_header)%>
        <h3><%= quiz.title %></h3>
        <%= quiz.get_header %>
        <br></br>
      <% else %>
        <!-- HTML Code -->
      <% end %>
    </div>
  </body>
</html>
```

```

    <%= yield %>
    <% if (quiz.get_footer) %>
      <%= quiz.get_footer %>
      <br></br>
    <% else %>
      <!-- HTML Code -->
    <% end %>
  </div>
  <!-- ##### JavaScripts ##### -->
  <!-- jQuery -->
  <%= @jQuery %>
  <!-- Internationalization -->
  <%= @i18n %>
  <!-- Drag and Drop -->
  <%= @dragdrop %>
  <!-- XRegExp -->
  <%= @xregexp %>
  <!-- Form validation -->
  <%= @validation_js %>
  <!-- jQuery ContextMenu -->
  <%= @context_menu %>
  <!-- Any JavaScript included by the user -->
  <%= @js_custom %>
  <!-- CodeMirror object -->
  <%= @codemirror_object %>
  <!-- JavaScript for Bootstrap -->
  <%= @bootstrap_js %>
</body>
</html>

```

Si no se especifica template, se mostrará un HTML sin apenas estilo. El profesor se deberá encargar de adecuarlo a su gusto posteriormente.

B.2. HtmlForm renderer

A continuación veremos un ejemplo de cuestionario explicando las peculiaridades de cada tipo de pregunta:

El fichero Ruby debe comenzar de esta manera:

```

quiz 'Example quiz' do
  head : 'examples/header.html' # Opcional

```



```
# Preguntas

foot : 'examples/footer.html' # Opcional
end
```

B.2.1. Preguntas FillIn

Para que el renderer genere las etiquetas input, se deben especificar tres guiones ('-') como mínimo. Si la cantidad es menor que tres, se mostrarán los guiones en el cuestionario. Si queremos que se muestren más de tres guiones seguidos, debemos escaparlos. Recuerda además que el número de guiones establecerá el largo del input.

```
tag = '<a href="www.google.es"></a> '
fill_in do
  text "<i>Example of escaped HTML and three hyphens not evaluated:</i><br>
#{escape(tag)}" + "is a \\-\\-\\- ---- " + '\\-\\-\\-'
  answer /^link$/
end
```

Ejemplo de pregunta con un *distractor* y un comentario:

```
fill_in :points => 2 do
  text 'The visionary founder of Apple is -----'
  comment 'Question too easy'
  answer /^ste(ve|phen)\s+jobs #comment $/imx
  distractor /^steve\s+wozniak/i, :explanation => 'Almost, but not quite.'
end
```

Ejemplo de pregunta con código LaTeX:

```
fill_in do
  text %q{
    When  $x = 2$ , the solution of  $\sqrt{3x+3}+(1+x)^2$  is:
    ----
  }
  answer 12
  distractor 11, :explanation => "Try again!"
end
```

La respuesta a la pregunta puede ser una cadena, una expresión regular, un número o un objeto JavaScript. En el caso de los tres primeros tipos de respuesta, se usará un *Array* si hay más de una respuesta en la pregunta.

```
fill_in do
  text 'The ---- brown fox jumped over the lazy ----'
  answer [/fox/, /dog/]
end
```

Se puede usar la opción *order* para especificar si las respuestas dadas se corresponden con el orden dado en la pregunta. Por defecto está a **true**.

```
fill_in do
  text 'The ---- brown fox jumped over the lazy ----'
  answer [/fox/, /dog/], :order => false
end
```

NOTA: en caso de especificar en el Array dos respuestas iguales para dos huecos diferentes, forzosamente es necesario especificar *order* a **true**.

También se pueden escribir este tipo de preguntas de dos formas más compactas:

Esta primera forma permite colocar la respuesta al lado de los guiones. Sólo se admiten respuestas de tipo *String*.

```
fill_in do
  text 'The three stooges are -----{larry}, ----{moe}, and -----{curly}.'

```

Esta otra forma permite asociar las respuestas con una clave, que la definiremos en un **Hash** que se pasará como respuesta.

```
fill_in do
  text "The capital of Tenerife is -----{:santa} Cruz de -----{:tenerife}"
  answer :santa => /Santa/i, :tenerife => /Tenerife/i
end
```

Por último, tenemos las respuestas de tipo **objeto JavaScript**. La opción *order* debe ser siempre **true**:

```
fill_in do
  text %q{
    Diga dos numeros x = ---- e y = ---- que multiplicados den 100
  }
  answer JavaScript.new(%q{result = function(x,y) { return (x * y === 100); }})
end
```

B.2.2. Preguntas Drag and Drop FillIn

Son muy similares a las preguntas FillIn normales. Sólo admiten respuestas de tipo *String* pero admiten los mismos parámetros opcionales.

```
drag_drop_fill_in do
  text 'The ---- brown fox jumped over the lazy ----'
  answer ['fox', 'dog']
end
```

B.2.3. Preguntas Multiple Choice

Aquí se presenta un ejemplo de este tipo de pregunta. Usando la opción *randomize*, el renderer ordenará al azar todas las respuestas y el cuestionario generado las mostrará en diferente orden:

```
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
```

B.2.4. Preguntas TrueFalse

Éste es un caso particular de las preguntas **Multiple Choice**:

```
truefalse 'The earth is flat.', false, :explanation => 'No, just looks that way'
```

B.2.5. Preguntas Drag and Drop Multiple Choice

Para las preguntas Multiple Choice de Drag and Drop no hace falta especificar la opción *randomize*, el renderer cambiará el orden de las respuestas siempre.

```
drag_drop_choice_answer do
  text "Relate these concepts"
  relation :Facebook => 'Mark Zuckerberg', :Twitter => 'Jack Dorsey'
end
```

B.2.6. Preguntas Select Multiple

Las preguntas de Select Multiple se definen del siguiente modo:

```
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
```

B.2.7. Preguntas Drag and Drop Select Multiple

Para las preguntas de Drag and Drop Select Multiple basta con especificar lo siguiente:

```
drag_drop_select_multiple do
  text "Relate these concepts"
  relation :Ruby => ['Sinatra', 'Rails'], :JavaScript => 'jQuery'
end
```

Se deberá usar una notación de **Array** cuando haya múltiples respuestas para un determinado ítem. Cuando sólo exista una respuesta por ítem, se puede indicar como un **String**.

B.2.8. Preguntas Programming

Para las preguntas de tipo **Programming** se puede especificar diversos parámetros:

- El lenguaje. Como este renderer valida las respuestas mediante JavaScript, sólo se permite este lenguaje.
- Ancho y largo. Si no se especifica, el cuadro de texto para redactar la respuesta será de 150px de largo y 800px de ancho.

NOTA: el parámetro *order* siempre debe ser **true**. Está configurado para ello por defecto.

Para la respuesta, se puede especificar:

- Un path indicando donde está el fichero que testea la respuesta introducida por el alumno: para ello, es necesario expresar ese path como un **símbolo**.
- Escribir como un **String** el código JavaScript que testea la entrada del alumno.

A continuación se muestra un ejemplo especificando un path donde se encuentra el fichero JavaScript con el que se comprueba la respuesta introducida del alumno:

```
programming :language => 'JavaScript', :height => 150, :width => 800 do
  text %q{Write a JavaScript function named 'suma' with two arguments that
  return the sum of them}
  answer JavaScript.new(:'examples/test_suma.js')
end
```

B.2.9. Otras consideraciones

- Todas las respuestas se guardarán una vez pulsado el botón de *enviar* haciendo uso de **Local Storage**. Para cada cuestionario generado, habrá un objeto guardado dentro de Local Storage. Es recomendable borrar el almacenamiento frecuentemente para que no persista en el navegador.
- EL menú contextual que muestra las respuestas sólo funciona en preguntas de tipo **FillIn** cuyas respuestas sean *Strings*, *RegExp* o numéricas. Para las preguntas de tipo test existe un botón que muestra y oculta las respuestas correctas.

B.2.10. Generando el cuestionario

Para generar el cuestionario, ejecutamos el siguiente comando:

```
[~]$ ruql [ruta_fichero_rb] HtmlForm -t [ruta_template.html.erb] > [fichero_salida]
```

B.3. Sinatra renderer

Este renderer permite todos los tipos de preguntas mencionados anteriormente con la diferencia de que en las preguntas de código (*FillIn* y *Programming*), el lenguaje permitido es **Ruby**. A continuación veremos dos ejemplos:

B.3.1. Preguntas FillIn

```
fill_in do
  text %q{
    Diga dos numeros x = ---- e y = ---- que multiplicados den 100
  }
  answer Ruby.new(%q{Proc.new do |x,y| x * y == 100 end})
end
```

B.3.2. Preguntas Programming

```
programming :language => 'Ruby', :height => 150, :width => 800 do
  text %q{Write a Ruby function named 'suma' with two arguments that
  return the sum of them}
  answer Ruby.new(:'examples/test_suma.rb')
end
```

B.3.3. Parámetros adicionales

En el fichero Ruby 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*.

```
quiz 'Example quiz' do

  teachers 'jjlabradorglez@gmail.com'

  .
  .
  .
end
```

- Por otra parte, se deberán indicar los alumnos permitidos para realizar el cuestionario. Se puede usar un *Hash* con la información necesaria de ellos.

```
quiz 'Example quiz' do

  students : 'jjlabradorglez@gmail.com' => { :surname => 'Labrador Gonzalez',
      :name => 'Juan Jose' },
      : 'tutu@gmail.com' => { :surname => 'Chuchu', :name => 'Tutu' }

  .
  .
  .
end
```

O indicar el path de un fichero **CSV** con los datos de los mismos llamado *students.csv*.

```
quiz 'Example quiz' do

  students : 'examples/students.csv'

  .
  .
  .
end
```

El formato del fichero CSV debe ser del siguiente modo:

```
casiano.rodriguez.leon@gmail.com, Rodriguez Leon, Casiano
tutu@gmail.com, Chuchu, Tutu
youwapp@gmail.com, Developers, YouWapp
```

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. En la siguiente subsección, se indicarán los pasos a realizar para dar de alta una aplicación en **Google Developers Console**.

NOTA: el nombre del subdominio de Heroku solo puede contener letras y números. Tampoco puede contener espacios. Sin embargo, puede especificarse con espacios en el fichero *config.yml*, el renderer lo modificará para que Heroku lo acepte.

```

---
quiz:
  schedule:
    date_start: '2014-06-09'
    date_finish: '2014-06-29'
    time_start: '20:00'
    time_finish: '23:00'
  heroku:
    domain: My Quiz # Must be equal to the subdomain specified in the redirect URI of the Google Developers Console
  google_drive:
    spreadsheet_name: Test
    folder: Example Quiz
    path: JuanJose/RuQL
    google_key: 350555787239-pvci2icb51hvomdos7jcde9dbg5c500h.apps.googleusercontent.com
    google_secret: uzs-5eEX05RMZfays0kpgH1U

```

Figura B.1: Ejemplo de fichero *config.yml* con la información necesaria

Para añadir este fichero al fichero del cuestionario especificamos el path del fichero en notación de **símbolo**:

```

quiz 'Example quiz' do

  .

  config : 'examples/config.yml'

  .

end

```


B.3.4. Dar de alta una aplicación en Google Developers Console

Pasos para dar de alta una aplicación en Google Developers Console:

1. Dirigirse a Google Developers Console
2. Crear un nuevo proyecto.



Figura B.2: Botón para crear un nuevo proyecto

3. Elegir un nombre para el proyecto.

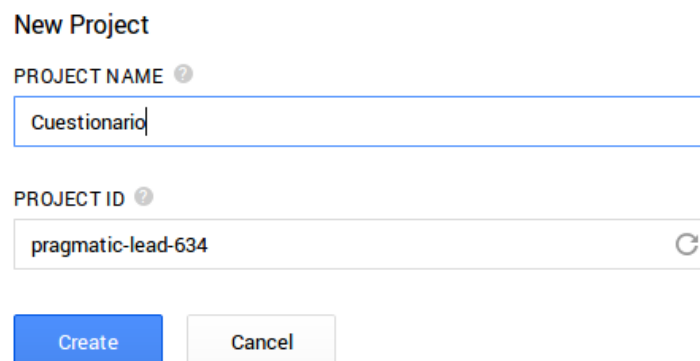
The image shows the 'New Project' form in the Google Developers Console. It has two input fields: 'PROJECT NAME' with the value 'Cuestionario' and 'PROJECT ID' with the value 'pragmatic-lead-634'. There are 'Create' and 'Cancel' buttons at the bottom.

Figura B.3: Elegir nombre para el proyecto

4. Una vez se ha creado, se nos redireccionará al mismo. Pinchamos en la sección **APIS & AUTH** y elegimos la opción *APIs*.

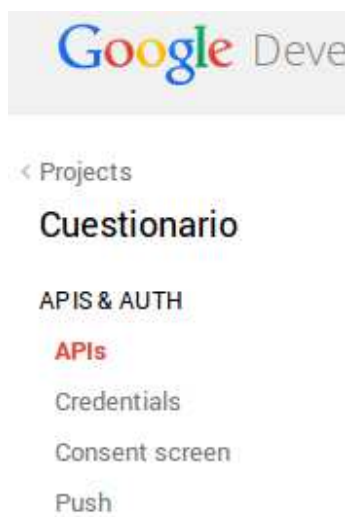


Figura B.4: Apartado de APIs

Deberemos activar las siguientes APIs:

- a) *Contacts API*
- b) *Drive API*
- c) *Drive SDK*
- d) *Google+ API*

5. Una vez hecho esto, nos iremos al apartado *Credentials* dentro de **APIS & AUTH** y seleccionaremos *Create new Client ID*.

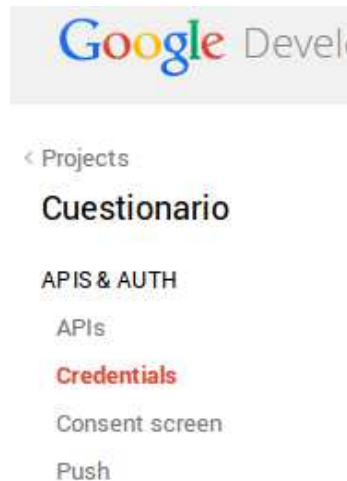


Figura B.5: Apartado de credenciales

OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

[Learn more](#)

Create new Client ID

Figura B.6: Crear nuevo cliente ID

6. Seleccionamos *Web application* y establecemos la URI a la que Google debe devolvernos los datos. La URI debe ser igual a la de la imagen salvo en el subdominio de Heroku, en el que pondremos el que nos ha dado Heroku o el que hemos especificado en nuestro *config.yml*

Create Client ID

APPLICATION TYPE

☒ **Web application**
Accessed by web browsers over a network.

☐ **Service account**
Calls Google APIs on behalf of your application instead of an end-user. [Learn more](#)

☐ **Installed application**
Runs on a desktop computer or handheld device (like Android or iPhone).

AUTHORIZED JAVASCRIPT ORIGINS
Cannot contain a wildcard (`http://*.example.com`) or a path (`http://example.com/subdir`).

AUTHORIZED REDIRECT URI
Needs to have a protocol, no URL fragment, and no relative paths

`https://subdomain.herokuapp.com/auth/google_oauth2/callback`

Create Client ID **Cancel**

Figura B.7: Pantalla de creación del cliente ID

- Opcionalmente, nos podemos dirigir al apartado *Consent screen* dentro de **APIS & AUTH** para indicar el nombre que queremos que aparezca en la pantalla de permisos cuando alguien vaya a dar permiso a nuestra aplicación para que use sus datos. Además, se pueden añadir otros campos tales como logos, URLs, etc.

Consent screen

The consent screen will be shown to users whenever you request access to their private data using your client ID.

Note: This screen will be shown for all of your applications registered in this project

EMAIL ADDRESS
jjlabradorglez@gmail.com

PRODUCT NAME
Cuestionario

HOMEPAGE URL (Optional)
https:// or http://

LOGO (Optional)
https:// or http://

This is how your logo will look to end users.
Max size: 120x120 px

Product Name -
Developer info
email:

This app would like to:

- Know your name, basic info, and list of people you're connected to on Google+
- Make your app activity and reviews available via Google, visible to you and:
- Your circles + Add more people
- Only you

Product Name and Google will use this information in accordance with their respective terms of service and privacy policies.

Cancel Accept

Figura B.8: Personalizar pantalla de permisos

B.3.5. Generando la aplicación

Para generar la aplicación, ejecutamos el siguiente comando. Para este renderer es obligatorio el uso de algún template:

```
[~]$ ruql [ruta_fichero_rb] Sinatra -t [ruta_template.html.erb]
```

B.3.6. Ficheros y directorios generados por el renderer

Finalmente, los ficheros que genera este renderer son:

- El código Ruby del servidor (**app.rb**).
- Un fichero **config.ru** para la ejecución de la aplicación.
- Las vistas necesarias de la aplicación (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.

B.3.7. Ejecutando la aplicación

En esta sección se explicará visualmente las funciones que realiza la aplicación:

1. Si el alumno visita el cuestionario antes de que esté abierto, verá la siguiente página:

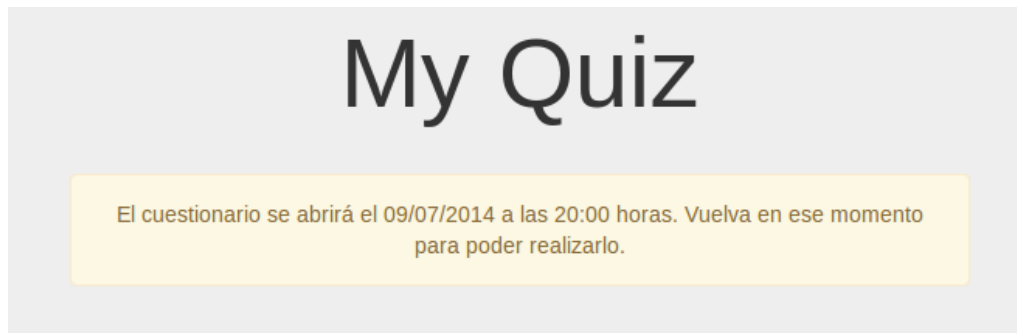


Figura B.9: Mensaje anunciando que el cuestionario no está abierto

2. Si el alumno visita el cuestionario después de que se haya cerrado, verá la siguiente página:

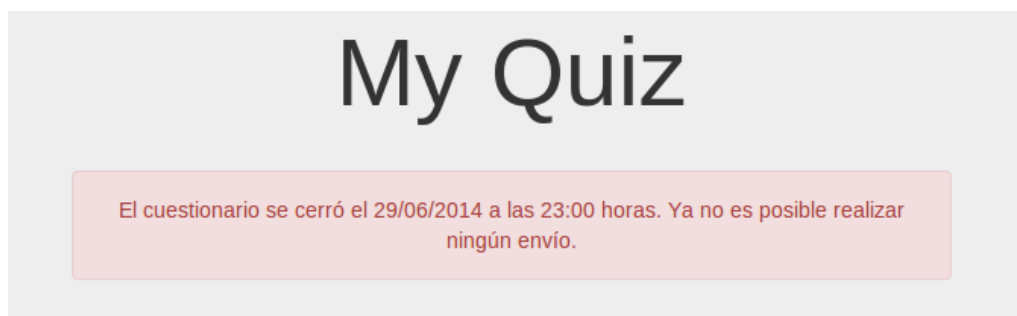


Figura B.10: Mensaje anunciando que el cuestionario está cerrado

3. Tanto los alumnos como los profesores verán esta página al iniciar sesión:

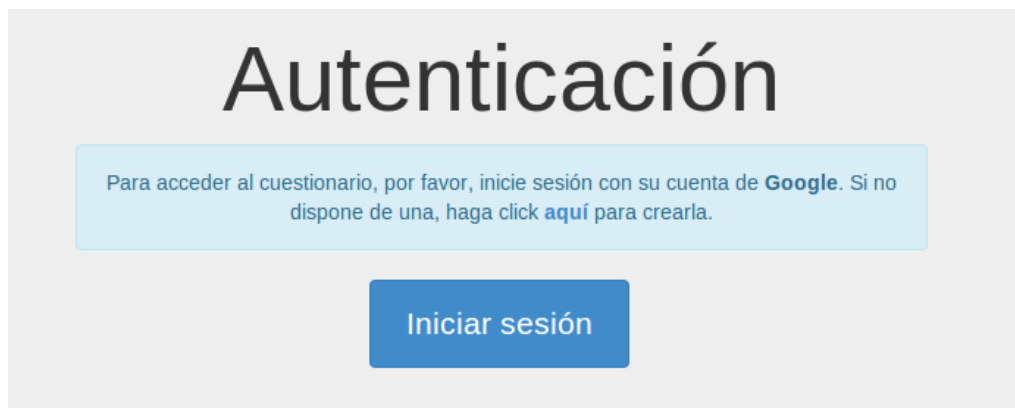


Figura B.11: Página de iniciar sesión

4. El profesor verá esta página cuando vaya a activar el cuestionario:

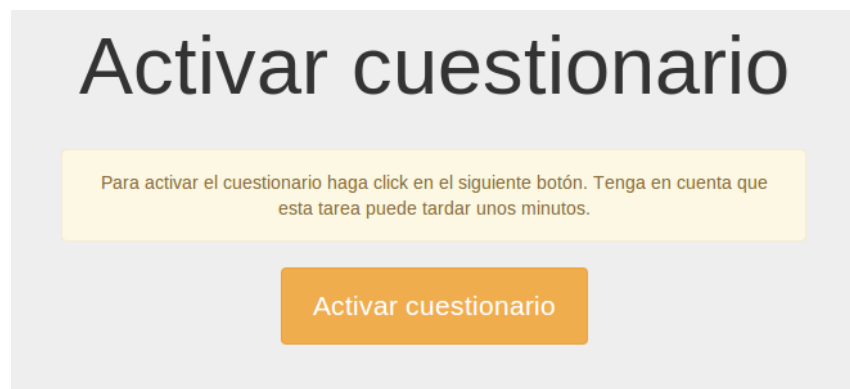


Figura B.12: Página de activar cuestionario

5. Tanto los alumnos como los profesores verán esta página para dar permisos a la aplicación al iniciar sesión con Google:

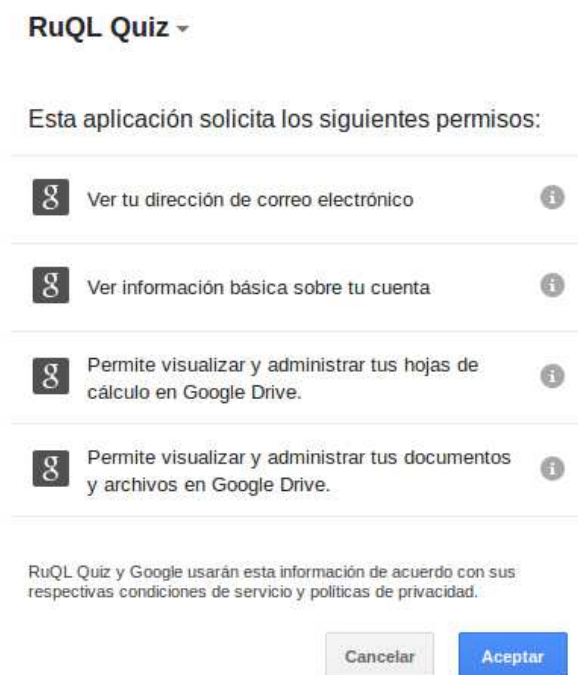


Figura B.13: Página de dar permisos a la aplicación

6. Mientras el profesor no haya activado el cuestionario, los alumnos verán esta página:

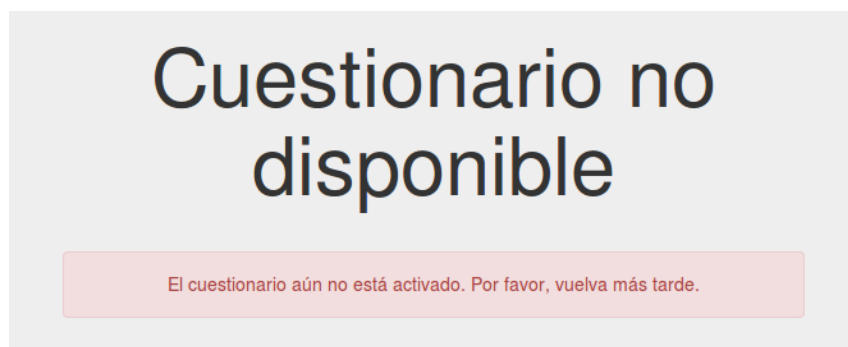


Figura B.14: Mensaje anunciando que el cuestionario aún no está activado

7. Cuando el profesor active el cuestionario verá esta página. Puede ir a Google Drive para comprobar que la hoja de cálculo se ha creado correctamente o visitar el cuestionario desplegado:



Figura B.15: Mensaje anunciando que el cuestionario ha activado

8. Así se vería el cuestionario desplegado:

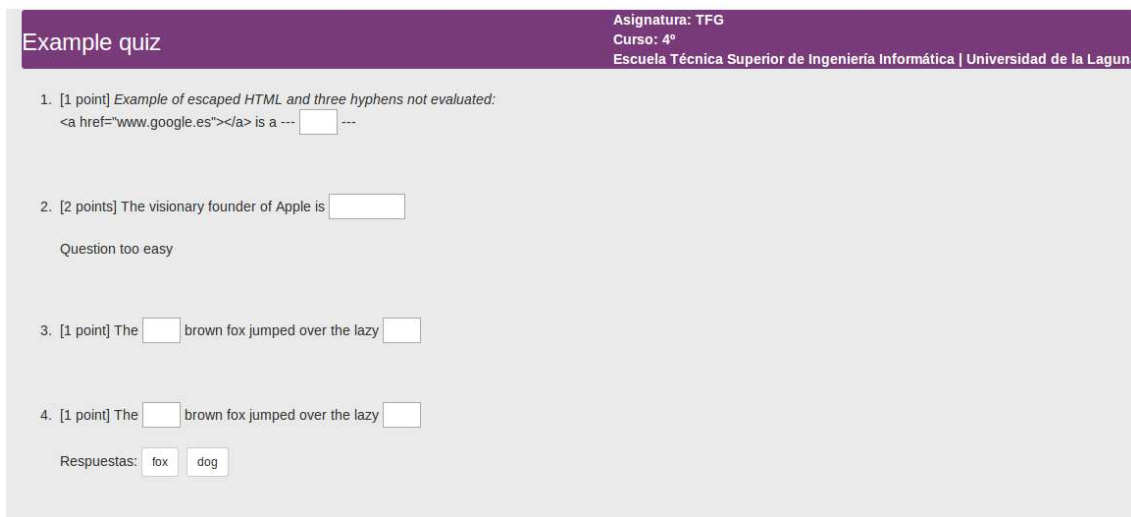


Figura B.16: Cuestionario desplegado

9. Cuando el profesor termine de comprobar el cuestionario y pulse *Enviar*, verá la siguiente pantalla, que le permitirá regresar al cuestionario o ver su carpeta de Google Drive.

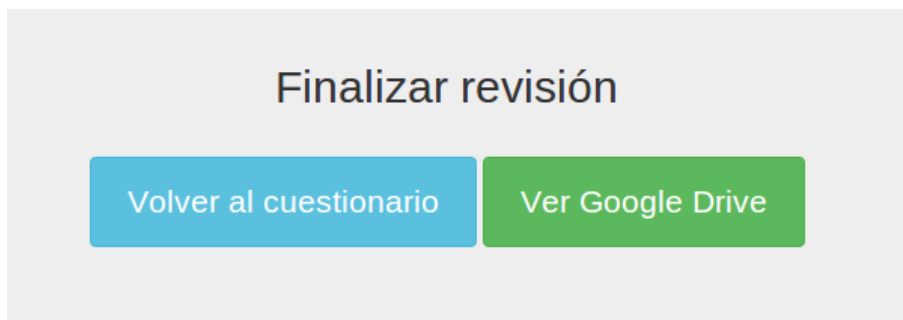


Figura B.17: Mensaje anunciando que ha finalizado la revisión del cuestionario

10. Esto es lo que verá el profesor en su carpeta de Google Drive.

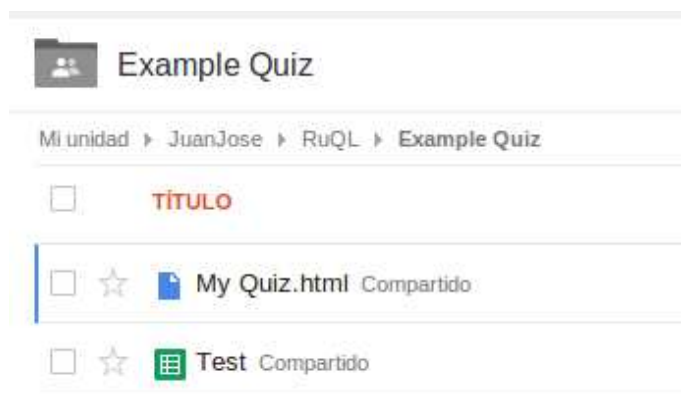
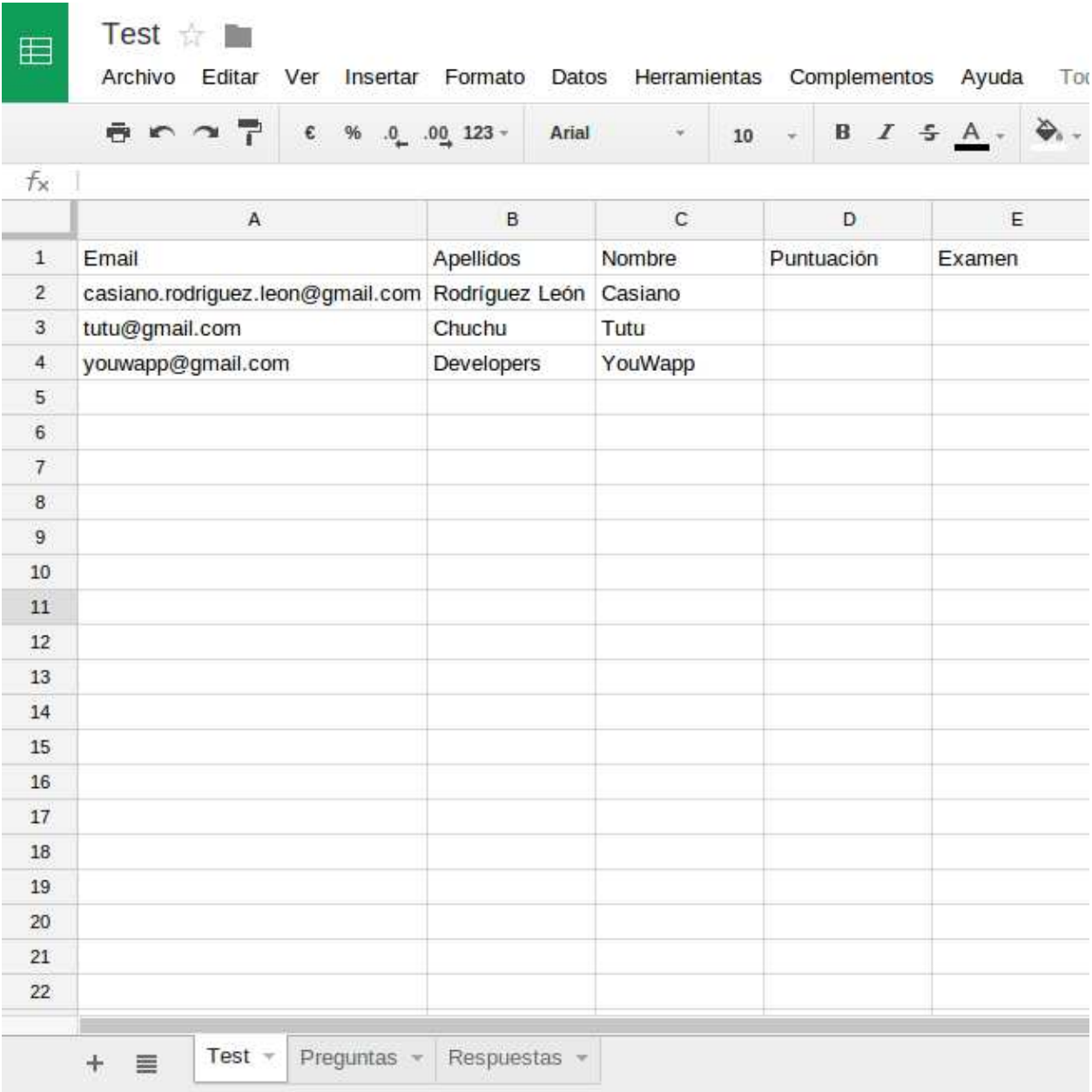


Figura B.18: Listado de ficheros dentro de la carpeta del cuestionario en Google Drive

11. Esta es la hoja de cálculo creada con toda la información. Esta es la primera hoja, que contiene la información de los alumnos.



The screenshot shows a spreadsheet application window titled "Test". The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Formato", "Datos", "Herramientas", "Complementos", "Ayuda", and "To". The toolbar contains various icons for file operations and formatting. The formula bar shows "fx". The spreadsheet grid has columns A through E and rows 1 through 22. The data is as follows:

	A	B	C	D	E
1	Email	Apellidos	Nombre	Puntuación	Examen
2	casiano.rodriguez.leon@gmail.com	Rodríguez León	Casiano		
3	tutu@gmail.com	Chuchu	Tutu		
4	youwapp@gmail.com	Developers	YouWapp		
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					

The bottom of the window shows a tab bar with a "+" icon, a menu icon, and three tabs: "Test", "Preguntas", and "Respuestas".

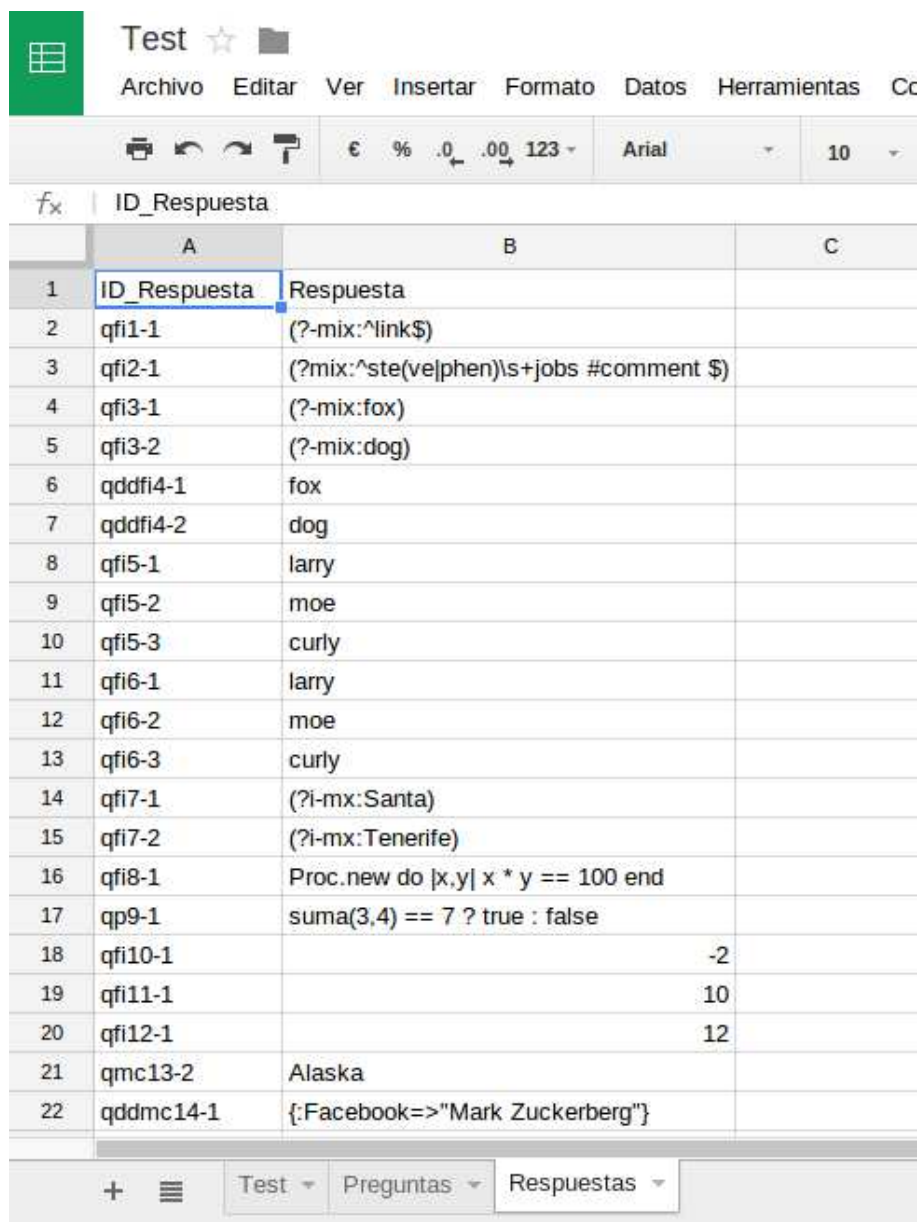
Figura B.19: Hoja de cálculo con la información de los alumnos

12. Esta es la segunda hoja, que contiene la información de las preguntas.

Test ☆			
Archivo Editar Ver Insertar Formato Datos Herramientas Complementos Ayuda Todos los cambios			
€ % .0 .00 123 Arial 10 B I A			
f_x	ID_Pregunta		
	A	B	
1	ID_Pregunta	Tipo_Pregunta	Pregunta
2	question-0	FillIn	<i>Example of escaped HTML and three hyphens not evaluated:</i>
3	question-1	FillIn	The visionary founder of Apple is -----
4	question-2	FillIn	The ---- brown fox jumped over the lazy ----
5	question-3	Drag and Drop FillIn	The ---- brown fox jumped over the lazy ----
6	question-4	FillIn	The three stooges are ----, ----, and ----.
7	question-5	FillIn	The three stooges are ----, ----, and ----.
8	question-6	FillIn	The capital of Tenerife is ---- Cruz de -----
9	question-7	FillIn	Diga dos números x = ---- e y = ---- que multiplicados den 100
10	question-8	Programming	Write a Ruby function named `suma` with two arguments that return the sum of the two arguments
11			Calculate the determinant of this matrix: $\mathbf{A} = \begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix}$ -----
	question-9	FillIn	
12			Solve: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ when n = 5 and k = 2:

Figura B.20: Hoja de cálculo con la información de las preguntas

13. Esta es la tercera hoja, que contiene la información de las respuestas correctas.



	A	B	C
1	ID_Respuesta	Respuesta	
2	qfi1-1	(?-mix:^(link\$)	
3	qfi2-1	(?mix:^(ste(ve phen)\s+jobs #comment \$)	
4	qfi3-1	(?-mix:fox)	
5	qfi3-2	(?-mix:dog)	
6	qddfi4-1	fox	
7	qddfi4-2	dog	
8	qfi5-1	larry	
9	qfi5-2	moe	
10	qfi5-3	curly	
11	qfi6-1	larry	
12	qfi6-2	moe	
13	qfi6-3	curly	
14	qfi7-1	(?i-mx:Santa)	
15	qfi7-2	(?i-mx:Tenerife)	
16	qfi8-1	Proc.new do x,y x * y == 100 end	
17	qp9-1	suma(3,4) == 7 ? true : false	
18	qfi10-1		-2
19	qfi11-1		10
20	qfi12-1		12
21	qmc13-2	Alaska	
22	qddmc14-1	{:Facebook=>"Mark Zuckerberg"}	

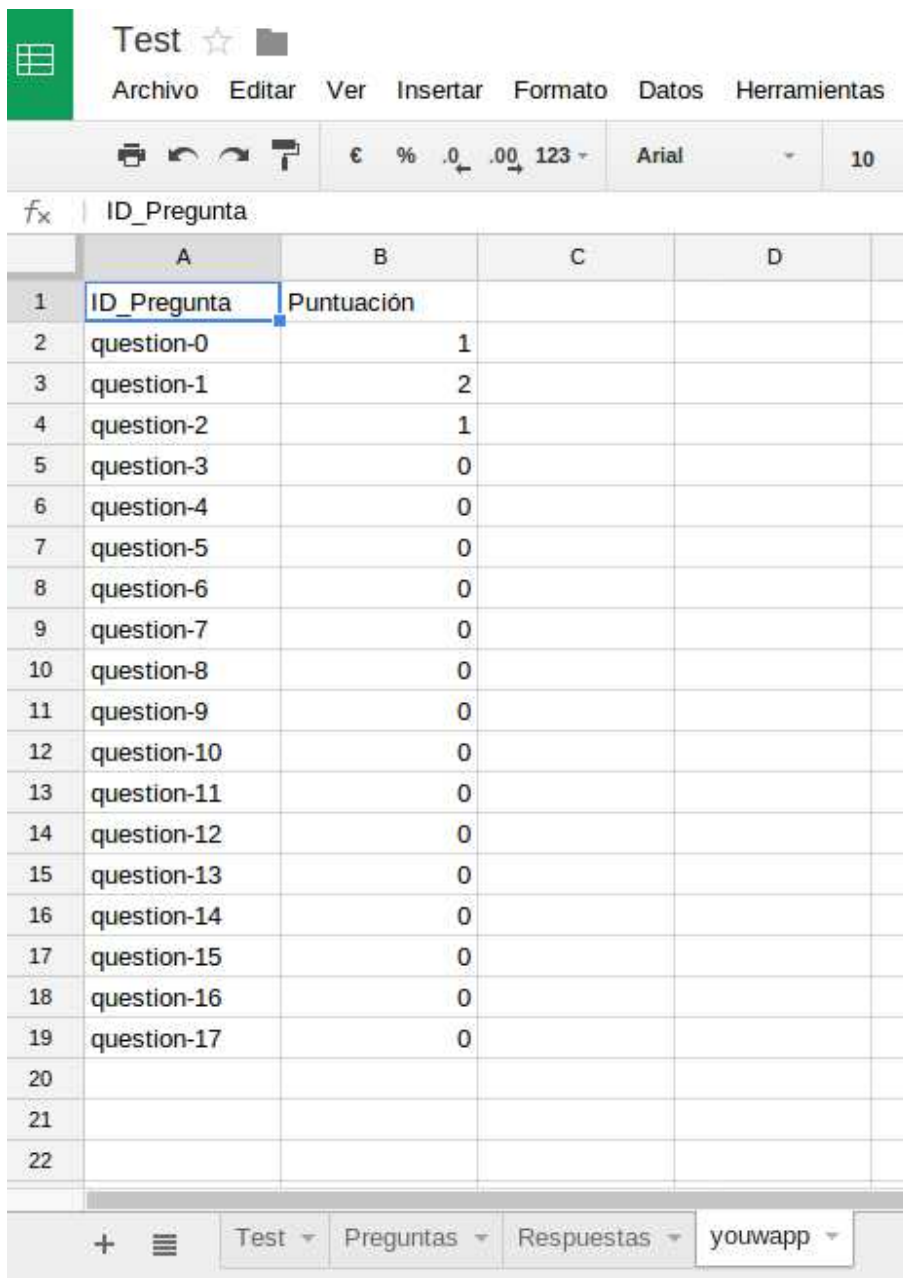
Figura B.21: Hoja de cálculo con la información de las respuestas correctas

14. Cuando un alumno complete el cuestionario, verá la siguiente pantalla. Podrá repetir el cuestionario todas las veces que desee dentro de la fecha permitida:



Figura B.22: Mensaje anunciando que ha completado cuestionario

15. Una vez completado el cuestionario, se creará una nueva hoja dentro de la hoja de cálculo del profesor con la puntuación que ha sacado el alumno en cada pregunta:

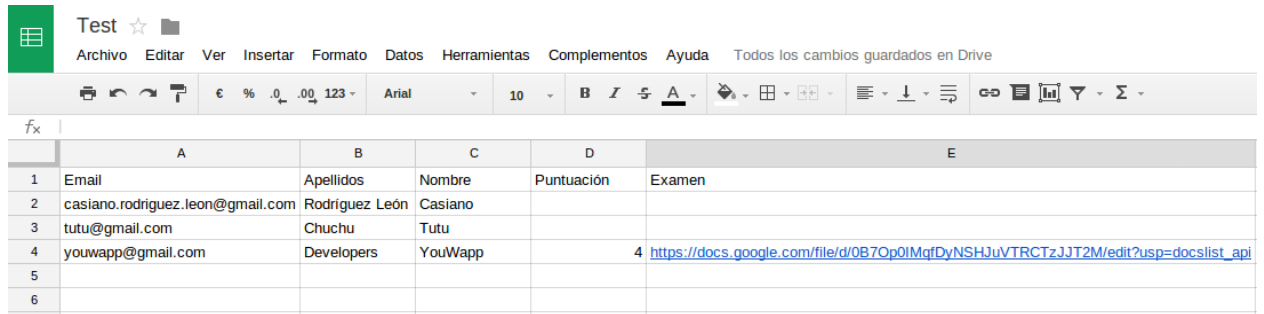


The screenshot shows a Google Sheets interface with a spreadsheet titled 'Test'. The spreadsheet has a table with two columns: 'ID_Pregunta' and 'Puntuación'. The table contains 18 rows of data, with question IDs from 'question-0' to 'question-17' and scores ranging from 0 to 2. The interface includes a menu bar with options like 'Archivo', 'Editar', 'Ver', 'Insertar', 'Formato', 'Datos', and 'Herramientas'. The bottom of the screen shows a tab labeled 'Test' and a sidebar with 'Preguntas' and 'Respuestas'.

	A	B	C	D
1	ID_Pregunta	Puntuación		
2	question-0	1		
3	question-1	2		
4	question-2	1		
5	question-3	0		
6	question-4	0		
7	question-5	0		
8	question-6	0		
9	question-7	0		
10	question-8	0		
11	question-9	0		
12	question-10	0		
13	question-11	0		
14	question-12	0		
15	question-13	0		
16	question-14	0		
17	question-15	0		
18	question-16	0		
19	question-17	0		
20				
21				
22				

Figura B.23: Hoja de cálculo con la puntuación por preguntas

16. En la primera hoja, se añadirá la información del alumno que hizo el cuestionario, su nota y un enlace a la copia del examen que realizó.



	A	B	C	D	E
1	Email	Apellidos	Nombre	Puntuación	Examen
2	casiano.rodriguez.leon@gmail.com	Rodríguez León	Casiano		
3	tutu@gmail.com	Chuchu	Tutu		
4	youwapp@gmail.com	Developers	YouWapp	4	https://docs.google.com/file/d/0B7Op0IMqfDyNSHJuVTRCTzJJT2M/edit?usp=docslist_api
5					
6					

Figura B.24: Información actualizada del alumno que realizó el cuestionario

17. Dentro de la carpeta de Google Drive del profesor se creará, además, una copia del cuestionario con las respuestas que puso el alumno.



Figura B.25: Listado de ficheros en Google Drive con la copia hecha por el alumno

18. Por último, esta sería la copia del cuestionario generada con las respuestas del alumno:

Example quiz

Asignatura: TFG
Curso: 4º
Escuela Técnica Superior

1. [1 point] Example of escaped HTML and three hyphens not evaluated:
 is a --- link ---

2. [2 points] The visionary founder of Apple is **steve jobs**
Question too easy

3. [1 point] The **fox** brown fox jumped over the lazy **dog**

4. [1 point] The **n/a** brown fox jumped over the lazy **n/a**

5. [1 point] The three stooges are **n/a**, **n/a**, and **n/a**.

Figura B.26: Copia del cuestionario con las respuestas del alumno

B.3.8. Otras consideraciones

- Este renderer no permite la opción de mostrar respuestas mientras se realiza el cuestionario.
- El Local Storage también funciona en este renderer, sin embargo, al ser un JavaScript, el evento de guardado de respuestas se dispara al perder el foco el input o textarea en el que nos encontramos.

Bibliografía

- [1] “7 ways to create and deliver online quizzes.” <http://www.freetech4teachers.com/2013/07/7-ways-to-create-and-deliver-online.html#.UwvA5vQhDtF>.
- [2] D. L. Christopher Douce and J. Orwell, “Automatic test-based assessment of programming: A review,” *ACM Journal of Educational Resources in Computing*, vol. 5, 2005.
- [3] O. Sepaälä, *Advances in Assessment of Programming Skills*. PhD thesis, Aalto University.
- [4] M. M. Pavel Jezek and T. Pop, *Automated Evaluation of Regular Lab Assignments: A Bittersweet Experience?* PhD thesis, Charles University in Prague.
- [5] Z. J. H.-F. Juan C. Rodríguez-del Pino, Enrique Rubio-Royo, *A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features*. PhD thesis, University of Las Palmas de Gran Canaria.