

Universidad de La Habana
Facultad de Matemática y Computación



**"Interfaz visual para la herramienta de
generación y evaluación de exámenes,
AutoExam."**

Autor: Héctor David Quintero Peña

Tutor: Lic. Pedro Quintero Rojas

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación



La Habana, Mayo de 2018

Mis más sinceros agradecimientos al profesor Pedro Quintero Rojas por el apoyo, colaboración y excelente guía durante la realización de este trabajo.

Agradezco además a todos los profesores de la Facultad de Matemática y Computación de la Universidad de la Habana por su ayuda durante todos estos años.

Las gracias a todos mis colegas de la Universidad de La Habana, quienes me ayudaron de manera incondicional durante toda la carrera.

Gracias a mi familia por la paciencia y el amor brindado, en especial a mi madre por ser guía y ejemplo.

A mi novia por ser compañera inseparable en los momentos más difíciles.

A todos, MUCHAS GRACIAS.



La Habana, 28 de mayo de 2018

Opinión de Tutor

Título: *Interfaz Visual para la Herramienta de Generación y Evaluación de Exámenes, AutoExam*

Diplomante: Héctor David Quintero Peña

Tutor: Lic. Pedro Quintero Rojas.

Durante los últimos meses Héctor ha venido desarrollando la tesis de diploma "*Interfaz Visual para la Herramienta de Generación y Evaluación de Exámenes, AutoExam*" que constituye uno de los requisitos para convertirse en licenciado en ciencias de la computación.

Durante ese período Héctor fue capaz de leer bibliografía actualizada, aprender tecnologías modernas, utilizar patrones de diseño y buenas prácticas de programación. Además, tuvo que vincularse con un grupo de desarrolladores e insertarse en un trabajo empezado por otras personas, tarea que resulta ser difícil e incómoda para muchos de nuestros estudiantes porque durante los 5 años de la carrera no se enfrentan con frecuencia a esta situación.

Héctor trabajó de manera muy independiente y en la mayoría de las situaciones fue capaz de resolver los problemas a los que se enfrentó con mucha destreza lo que demuestra la solidez de los conocimientos adquiridos durante la carrera. Integró conocimientos y habilidades de diferentes asignaturas y disciplinas de varios años.

Creemos que está listo para obtener su título de licenciado en ciencias de la computación y enfrentarse a la vida profesional como una persona muy capaz y con sobrados conocimientos técnicos y una elevada calidad humana.

Por todo lo antes expuesto, considerando la actualidad del trabajo realizado, su gran utilidad, así como la calidad de la exposición, propongo la calificación de excelente.

Lic. Pedro Quintero Rojas

Resumen

Introducción: AutoExam es una herramienta creada en la facultad de Matemática y Computación de la Universidad de La Habana que permite la generación y evaluación, de forma automática, de exámenes. La interacción de los usuarios con esta herramienta puede ser compleja, debido a que se realiza a través de la ejecución de sus componentes a partir de líneas de comandos en terminales de sistemas operativos.

Objetivo: El objetivo de este trabajo es crear una aplicación web capaz de brindar a los usuarios una interfaz sencilla, amigable y comprensible.

Diseño e Implementación: En este documento se exponen detalles sobre el diseño e implementación de la aplicación. La misma permite la generación y evaluación de exámenes a partir de la interacción con la herramienta AutoExam. Los comandos de esta herramienta son ejecutados por la aplicación en dependencia de las acciones realizadas por los usuarios en la interfaz gráfica. Este proceso es invisible para el usuario, por lo que no necesita de altos conocimientos informáticos o acerca de AutoExam para generar y evaluar exámenes. También pueden generarse reportes estadísticos de los resultados obtenidos y gráficas para su mejor análisis.

Conclusiones: Se logró la creación de una interfaz gráfica para la herramienta AutoExam. La misma fue probada y quedó lista para su uso.

Abstract

Introduction: AutoExam is a tool created in the faculty of Mathematics and Computation of the University of Havana that allows the generation and evaluation, automatically, of exams. The interaction of users with this tool can be complex, because it is done through the execution of its components from command lines in operating system terminals.

Objective: The objective of this work is to create a web application able to provide users with a simple, friendly and understandable interface.

Design and Implementation: This document details the design and implementation of the application. It allows the generation and evaluation of exams based on the interaction with the AutoExam tool. The commands of this tool are executed by the application depending on the actions performed by the users in the graphical interface. This process is invisible to the user, so it does not need high computer skills or about AutoExam to generate and evaluate exams. You can also generate statistical reports of the results obtained and graphs for your best analysis.

Conclusions: The creation of a graphic interface for the AutoExam tool was achieved. It was tested and was ready for use.

Índice general

Introducción	1
Objetivos	2
Objetivo General	2
Objetivos Específicos	2
1. Marco Teórico	4
1.1. Estado del Arte	4
1.2. Herramientas Utilizadas	6
2. Diseño de la aplicación	9
2.1. ¿Cómo funciona AutoExam?	9
2.1.1. Principales comandos y ficheros	9
2.1.2. Proceso de generación y evaluación de un examen . .	12
2.2. Propuesta	15
2.3. ¿Cómo funciona la aplicación?	17
2.4. Funcionalidades brindadas	22
3. Implementación	24
3.1. Implementación	24
3.2. Usuarios y Roles	25
3.3. Interacción con AutoExam y generación de ficheros	26
3.4. Evaluación de exámenes	27
3.5. Búsqueda	28
Conclusiones	30
Recomendaciones	32
Bibliografía	33

Índice de figuras

2.1. Ficheros contenidos en el directorio de un proyecto de Auto-Exam.	10
2.2. Versiones de un examen.	10
2.3. Ficheros contenidos dentro de una versión de un examen.	10
2.4. Fragmento de un fichero master.txt.	11
2.5. Fragmento de un fichero grader.txt.	12
2.6. Par de documentos pertenecientes a un examen.	14
2.7. Modelo de la aplicación.	16
2.8. Vista de modificación de etiquetas.	18
2.9. Vista de creación y edición de preguntas y opciones.	19
2.10. Primera parte del proceso de creación de un examen.	19
2.11. Segunda parte del proceso de creación de un examen.	20
2.12. Vista de una versión de un examen.	21
2.13. Evaluación de un examen a través de la <i>webcam</i>	22
3.1. Sistema autenticación de usuarios en la aplicación.	25
3.2. Algunos gráficos generados por la aplicación.	28
3.3. Resultados de búsqueda en la aplicación.	29
3.4. Vista principal de la aplicación.	30

Índice de tablas

2.1. Principales líneas de comando de AutoExam.	9
2.2. Principales ficheros de un examen generado por AutoExam.	11

Introducción

La evaluación de los conocimientos y habilidades adquiridas por las personas, generalmente estudiantes, es para los centros educativos e investigativos una actividad que impone continuos retos. Conocer si los estudiantes han adquirido los conocimientos impartidos en clases, es un problema que enfrentan a diario los profesores.

Junto a esto, determinar cuál es el sistema de evaluación idóneo es otra problemática que existe hoy en día. En los centros educativos, por ejemplo, se utilizan las preguntas orales, los seminarios, los proyectos y las evaluaciones escritas para valorar a los estudiantes. Estas últimas son las más utilizadas debido a que generalmente abarcan una gran cantidad de contenido y exigen a los alumnos dominar determinados conocimientos y habilidades.

Sin embargo, este sistema de evaluación posee algunos inconvenientes en cuanto a su aplicación. Por una parte, la existencia del fraude entre los estudiantes exige, en ocasiones, la creación de varios temarios a la hora de confeccionar el examen. Además, la revisión de las respuestas puede ser un proceso que consuma mucho tiempo. Tanto el total de educandos evaluados como el de preguntas contenidas en el examen pueden ser números grandes. Si multiplicamos el número de preguntas por el de evaluados entonces el tiempo de revisión se extiende, muchas veces, por encima del plazo acordado.

En los tiempos actuales, donde la tecnología forma gran parte de nuestras vidas, no sería descabellado pensar en que las máquinas podrían ayudarnos en el trabajo de crear y revisar evaluaciones escritas. En efecto, existen herramientas que permiten la creación y evaluación de exámenes de forma automática y rápida.

Tal es el caso de AutoExam[6], una herramienta creada en el departamento de Inteligencia Artificial y Sistemas Computacionales de la Facultad de Matemática y Computación de la Universidad de La Habana, capaz de generar un examen a partir de un conjunto de preguntas y evaluarlos de

forma automática, a través de imágenes de las respuestas tomadas por una cámara.

Los exámenes entregados a los estudiantes pueden contener distintas interrogantes. Por otra parte, tanto las preguntas que son comunes, como sus opciones pueden aparecer en distinto orden. La herramienta se encarga de generar los exámenes con la mayor aleatoriedad posible, cumpliendo con todas las restricciones proporcionadas por el usuario.

Para evaluar los exámenes no es necesario contar con ningún equipamiento especial. Este proceso se realiza con una cámara tan común como la *webcam*, integrada en cualquier computadora actual.

La interacción del usuario con esta herramienta se realiza a través de un conjunto de ficheros y programas (scripts) desarrollados en Python que se ejecutan en una terminal[6]. Los ficheros poseen una sintaxis específica, la cual debe ser conocida por el usuario para el correcto funcionamiento de la herramienta.

AutoExam lleva varios años siendo utilizada por los profesores de la Facultad de Matemática y Computación. Existe un fuerte interés por parte de profesores de otras facultades e instituciones de usar la herramienta. Sin embargo muchos plantean que su uso puede volverse complicado debido a que es necesario conocer las funciones de cada una de las líneas de comandos y la sintaxis específica de los ficheros.

El principal problema es que no se cuenta con una interfaz visual que permita interactuar con la herramienta y llevar a cabo el proceso de generación y evaluación de exámenes de forma simple e intuitiva. Esta razón dificulta el uso de AutoExam por parte de usuarios que no poseen altos conocimientos informáticos. Debido a este problema nuestro trabajo se plantea los siguientes objetivos.

Objetivos

Objetivo General

Crear una interfaz gráfica para la herramienta AutoExam que resulte sencilla y comprensible para todos los usuarios.

Objetivos Específicos

1. Revisar en profundidad la documentación e implementación de la herramienta AutoExam para conocer su funcionamiento.

2. Analizar las interfaces visuales de otras herramientas similares a AutoExam para comparar las funcionalidades que brindan con las de nuestra aplicación.
3. Implementar una interfaz gráfica web que interactúe con la herramienta AutoExam.
4. Realizar pruebas de todas las funciones implementadas en el proyecto.

Este documento está estructurado en 3 capítulos. En el capítulo 1 se habla sobre el marco teórico de nuestro proyecto y se especifican las herramientas utilizadas para la implementación de nuestra aplicación. El capítulo 2 habla sobre diseño de la aplicación, su arquitectura y funcionamiento. Además se describe el proceso de generación y evaluación de un examen mediante el uso de la herramienta AutoExam. El capítulo 3 contiene temas sobre la implementación de la aplicación.

El documento contiene también las conclusiones obtenidas al finalizar el proyecto y las recomendaciones que se proponen para el mejoramiento de la aplicación.

Capítulo 1

Marco Teórico

1.1. Estado del Arte

Utilizar herramientas informáticas para generar y evaluar exámenes es bastante común hoy en día. Las más populares combinan una interfaz visual sencilla y atractiva para el usuario con la generación de reportes y estadísticas para el análisis de los resultados obtenidos.

En el caso específico de AutoExam, se han creado algunas interfaces visuales que permiten la interacción de los usuarios con dicha herramienta[6]. Sin embargo estas interfaces no abarcan todas las funciones que proporciona AutoExam.

El objetivo de este trabajo es realizar una nueva interfaz gráfica web, capaz de interactuar con la herramienta AutoExam, que resulte sencilla y comprensible para los usuarios. Para esto se realizó una investigación sobre algunas herramientas similares a AutoExam, que si poseen interfaz gráfica, existentes en la actualidad. El objetivo de esta investigación no es comparar estas herramientas con AutoExam, sino analizar las componentes visuales que resultan más populares para los usuarios para tratar de integrarlas en nuestra aplicación.

Durante la investigación analizamos las siguientes herramientas:

- **GradeCam[5]**

Es una aplicación web en la cual se crean dos documentos, uno con preguntas de múltiple selección y otro con las opciones de respuesta. Los documentos son creados desde la aplicación, ingresando las preguntas junto con sus opciones de respuestas. El usuario debe indicarle a la aplicación cuales opciones son las correctas.

A cada estudiantes se le entrega un documento de cada tipo. Luego que éste responda las preguntas, la aplicación utiliza una cámara que identifica las respuestas proporcionadas, registrando las puntuaciones de cada estudiante en un reporte.

La interfaz visual de esta aplicación posee componentes que resultan muy agradables a la vista del usuario, como la barra de navegación lateral, que proporciona accesos directos a las distintas secciones de la aplicación. El uso de listas desplegables en algunos casos, y en otros el autocompletamiento evita errores cuando el usuario desea insertar información.

Algunas vistas, como la de seleccionar las opciones que son verdaderas, utilizan componentes gráficas para una mejor comprensión. En este caso el usuario puede cambiar una opción de verdadera a falsa o viceversa, simplemente tocando la opción. Las opciones se muestran como círculos en blanco si son verdaderas o negros si son falsas.

Los reportes generados muestran gráficos de diferentes tipos. Se pueden generar tablas con los resultados obtenidos, ordenados por dichos valores, o comparar las preguntas según las veces que han sido respondidas correctamente.

■ **Auto-Multiple-Choice[13]**

Es un software libre escrito para Linux, aunque también puede ser instalado en MAC OS X. Consiste en un set de herramientas que permiten el uso de preguntas de múltiple selección escritas en texto plano o LaTeX, para la generación, corrección y evaluación de forma automática de exámenes. Este *software* evalúa los exámenes mediante imágenes de las respuestas tomadas por una *webcam* usando *Optical Mark Recognition (OMR)*.

La edición de las preguntas y sus opciones se realiza a través de la modificación de un fichero escrito en LaTeX. En este fichero el usuario debe insertar todas las preguntas junto con sus opciones de respuestas que aparecerán en el examen. Además se indican las opciones verdaderas mediante la sintaxis propia de LaTeX.

Los elementos visuales más destacados son los botones que abren el fichero de LaTeX y los documentos generados. Además, las pestañas ubicadas en la parte superior separan las etapas del proceso, proporcionando al usuario una vista muy ordenada. Estas etapas son, creación de las preguntas, evaluación de los exámenes y generación de

reportes estadísticos. Los reportes pueden ser exportados en formato *.csv* (Hojas de cálculo).

Como limitantes podemos destacar que los usuarios deben tener conocimientos de LaTeX para poder editar el fichero que contiene las preguntas y las opciones de respuesta.

Del análisis de estas herramientas pudimos elaborar un diseño de nuestra aplicación, tomando los componentes más populares e integrándolos a nuestro proyecto.

1.2. Herramientas Utilizadas

■ Ruby:

Es un lenguaje de programación interpretado y orientado a objetos creado por Yukihiro Matsumoto en 1995 [1]. Su implementación oficial es distribuida bajo una licencia de software libre. Entre sus principales características encontramos cuatro niveles de ámbito de variables (global, clase, instancia y local), manejo de excepciones, interpretación de expresiones regulares, recolector de basura y sobrecarga de operadores.

La aplicación fue implementada utilizando la versión de Ruby 2.2.3. La decisión de usar este lenguaje de programación se basa en el *framework* de desarrollo web utilizado y en los conocimientos de Ruby por parte del autor y el tutor.

■ Ruby on Rails:

Existen numerosos *frameworks* de desarrollo web como Django, Flask o ASP.Net. Todos posibilitan la creación de una interfaz gráfica web como la de nuestra aplicación. La decisión de usar Ruby on Rails se basa fundamentalmente en el conocimiento del mismo por parte del autor y del tutor.

Ruby On Rails, comúnmente conocido por Rails[8], es un *framework* de aplicaciones web de código abierto escrito en Ruby, el cual sigue la arquitectura Modelo Vista Controlador (MVC). Su principio es la simplicidad y legibilidad en el código, haciendo uso de la metaprogramación permitida por el lenguaje de programación Ruby.

Uno de sus componentes fundamentales son las gemas, las cuales son *plugins* que se añaden a los proyectos de Ruby on Rails y permiten nuevas funcionalidades y herramientas.

Rails proporciona facilidades para implementar aplicaciones Ajax, que es una técnica que permite usar JavaScript y XML para procesar en un segundo plano peticiones enviadas a un servidor web desde un navegador sin cargar otras páginas web adicionales. Además, incluye el framework de JavaScript *jQuery*, que proporciona herramientas para la realización de llamadas Ajax.

Debido a su arquitectura Rails favorece el uso de bases de datos para el almacenamiento de información. Por defecto el *framework* soporta la biblioteca SQLite. El acceso a la base de datos es totalmente abstracto desde el punto de vista del programador y son gestionados automáticamente por el *framework*, aunque también pueden hacerse consultas directas usando el lenguaje SQL.

La aplicación fue implementada utilizando la versión de Rails 5.0.0

■ Modelo Vista Controlador (MVC):

Modelo Vista Controlador (MVC) es un patrón de arquitectura de *software* basado en las ideas de reutilización de código y la separación de conceptos [7]. En él se separan los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos: el modelo, la vista y el controlador.

Los modelos representan la información con la que el sistema opera y, por tanto, son los encargados de gestionar los accesos a dicha información que llegan a través del controlador. Además, envían a las vistas, para ser mostrada, la parte de la información solicitada por el usuario.

Los controladores responden a eventos (usualmente provocados por el usuario) e invoca peticiones a los modelos cuando se realiza alguna solicitud de información. También pueden enviar comandos a las vistas asociadas si se solicitan cambios en la forma en que se presenta los modelos.

Las vistas presentan los modelos en un formato adecuado para interactuar con los usuarios. Las mismas requieren de dichos modelos la información que debe representar como salida.

La decisión de utilizar MVC es debido a que es el patrón que utiliza Ruby on Rails[9].

- **SQLite:**

Es un proyecto de dominio público el cual consiste en un sistema de gestión de bases de datos relacional (SGBDR) contenido en una biblioteca escrita en el lenguaje de programación C [10]. En ella se implementa la mayor parte del estándar SQL-92 incluyendo consistencia, aislamiento y durabilidad, *triggers* y la mayor parte de las consultas complejas [11].

Distinto a otros sistemas de gestión de base de datos, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica, sino que se enlaza con este pasando a ser parte del mismo y comunicándose a través de llamadas simples a subrutinas y funciones.

Entre sus características fundamentales SQLite no asigna un tipo a cada columna, sino que los tipos se asignan a los valores individualmente. Además, varios procesos o hilos pueden acceder a la base de datos al mismo tiempo. Los accesos de lectura pueden ser servidos en paralelo mientras que los accesos de escritura deben ser servidos individualmente.

La base de datos utilizada en la implementación de la aplicación pertenece a la versión de SQLite 3.

- **C3.js:**

C3.js es una librería de JavaScript construida sobre D3.js que brinda una API muy sencilla de utilizar y con la que podremos crear y mostrar gráficas con pocas líneas de código [3].

En la aplicación se utiliza la versión de C3.js v0.5.4.

- **Bootstrap:**

Bootstrap es un *front-end framework* de código abierto (*open source*) que permite diseñar proyectos web rápido y fácil y hacer estos escalables a cualquier dispositivo, desde computadoras hasta teléfonos inteligentes [2].

Su inclusión en la aplicación se realizó a través de la gema *twitter-bootstrap-rails* en su versión 3.2.2.

Capítulo 2

Diseño de la aplicación

2.1. ¿Cómo funciona AutoExam?

2.1.1. Principales comandos y ficheros

AutoExam fue creado usando Python como lenguaje de programación y consiste en un conjunto de ficheros y programas que se ejecutan a través de comandos desde una terminal. Los principales comandos de la herramienta se muestran en la tabla 2.1.

Comando	Sintaxis	Descripción
<i>new</i>	\$ autoexam new [directory_name]	Crea un nuevo proyecto de AutoExam.
<i>gen</i>	\$ autoexam gen -c [exams_ammount]	Genera una nueva versión de un examen.
<i>scan</i>	\$ autoexam scan	Inicia el escaner de AutoExam para evaluar los documentos.
<i>grade</i>	\$ autoexam grade -v [version]	Genera las notas de los documentos escaneados.

Tabla 2.1: Principales líneas de comando de AutoExam.

Al ejecutarse el comando *new* se crea un nuevo proyecto de AutoExam con el nombre especificado[6]. Dentro del mismo se generan una serie de ficheros, entre ellos, el *master.txt* (Ver Figura 2.1).



Figura 2.1: Ficheros contenidos en el directorio de un proyecto de AutoExam.

El comando *gen* genera una nueva versión de un examen[6]. Los ficheros pertenecientes a una versión son guardadas en carpetas con el nombre *v-#versión*, ubicadas en el directorio *AutoExam_project/generated* (Ver Figura 2.2). Estas carpetas contienen los documentos con las preguntas y las respuestas que fueron generados y los ficheros necesarios para la evaluación.



Figura 2.2: Versiones de un examen.

En las carpetas de las versiones la herramienta AutoExam genera los directorios *images*, *log*, *pdf*, *src*, y los ficheros *grader.txt*, *order.json* y *seed* (Ver Figura 2.3). En el directorio *pdf* se encuentran los ficheros de extensión *.pdf* correspondientes a los documentos generados para la aplicación del examen.

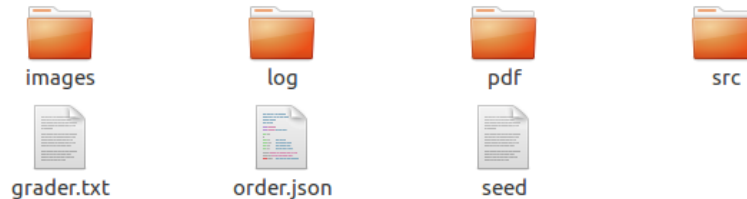


Figura 2.3: Ficheros contenidos dentro de una versión de un examen.

El comando *scan* inicia el escaner para permitir la evaluación de los documentos [6]. El comando *grade* es usado para calificar los exámenes que han sido escaneados por la herramienta [6]. En el capítulo 2.1.2 se abordarán más detalles acerca de los comandos.

Algunos comandos requieren de ficheros con sintaxis específicas durante su ejecución. Los ficheros son generados por la herramienta y pueden ser editados por los usuarios para personalizar los exámenes. Los principales ficheros se muestran en la tabla 2.2.

Fichero	Descripción
<i>master.txt</i>	Contiene las preguntas y opciones de un examen.
<i>grader.txt</i>	Contiene la clave de un examen.

Tabla 2.2: Principales ficheros de un examen generado por AutoExam.

Por cada examen se genera un fichero *master.txt*, donde se almacena el nombre del examen y todas las preguntas y sus opciones de respuestas que pueden aparecer en sus documentos [6]. Además contiene algunas restricciones como el número de preguntas por examen y el mínimo por etiquetas.

```

1 % Final Exam
2 % =====
3 % Lines starting with a % are ignored.
4 % Blank lines are ignored as well.
5
6 % the number of questions in each document
7 total: 4
8
9 % the tags that will be used in the test
10 % each tag comes with the minimum number of questions
11 @bits: 3
12
13 -----
14 (1) %%%%%%%%%%%
15 @bits
16 How many bits are in one byte?
17 _ 12
18 _ 4
19 _x 8
20

```

Figura 2.4: Fragmento de un fichero master.txt.

La Figura 2.4 muestra un fragmento de este fichero. En este ejemplo la línea 1 contiene el nombre del examen, mientras que la 7 posee el número de preguntas en el examen. La línea 11 representa una de las etiquetas seleccionadas para el examen junto con el mínimo número de preguntas asociadas a ella que pueden aparecer en el examen.

Las líneas de la 14 a la 16 pertenecen a una de las preguntas que puede aparecer en el examen. En la 14 encontramos el identificador de la pregunta

para este examen, en la 15 la etiqueta asociada (una pregunta puede tener asociadas más de una etiqueta) y en la 16 la orden de dicha pregunta.

Las líneas 17, 18 y 19 representan opciones de respuesta para la pregunta. La 17 y 18 son opciones falsas, mientras que la 19 representa una opción verdadera.

Por cada versión de un examen se genera un fichero *grader.txt*. Este contiene los puntos otorgados por seleccionar o no cada una de las opciones de respuestas de las preguntas contenidas en el fichero *master.txt* del examen [6].

```
1 23
2
3 1
4 total: 3
5 0:1 0:2 3:0
6
7 2
8 total: 2
9 1:1 2:0
10
11 3
12 total: 2
13 1:0 0:1
14
15 4
16 total: 2
17 1:0 0:1
18
19
20
```

Figura 2.5: Fragmento de un fichero *grader.txt*.

La Figura 2.5 contiene un fragmento de un fichero *grader.txt*. En este ejemplo las líneas 3, 4 y 5 representan los puntos otorgados a las opciones de una pregunta. La línea 3 posee el identificador de la pregunta, mientras que en la 4 aparece el total de opciones asociadas. La línea 5 contiene la puntuación que se obtiene por marcar o no (separados por ":") cada una de las opciones.

2.1.2. Proceso de generación y evaluación de un examen

- **Paso 1:** Lo primero que debe hacer un usuario es crear un nuevo proyecto de la herramienta AutoExam. Esto se realiza a través de la línea de comando *new*, la cual crea un nuevo directorio que contine entre otros ficheros el *master.txt*.

```
$ autoexam new -f [directory_name]
```

- **Paso 2:** Como convención, el fichero *master.txt* contiene las preguntas que pueden aparecer en el examen y el número de estudiantes a los que se les va a aplicar el examen. El usuario debe editar este fichero, siguiendo su sintaxis específica, para personalizar el examen. Puede incluir tantas preguntas como desee, pero las restricciones tienen que poder cumplirse. El número de preguntas por examen debe ser menor o igual al total de preguntas en el fichero y lo mismo debe cumplirse por cada etiqueta.
- **Paso 3:** A través de la línea de comando *gen* y el fichero *master.txt* se generan los documentos pertenecientes a una nueva versión del examen y la clave para la evaluación de los mismos. Por cada estudiante se generan dos documentos (Ver Figura 2.6). En uno aparece un subconjunto de las preguntas contenidas en el fichero *master.txt* y en el otro, recuadros para seleccionar las opciones correspondientes a las preguntas en el otro documento. Ambos documentos poseen el mismo número, que los identifica y diferencia del resto.

```
$ autoexam gen -c [exams_amount]
```

Generalmente, los estudiantes encuentran en sus documentos las mismas preguntas, sin embargo, ni estas, ni sus opciones aparecen en el mismo orden en cada documento. Además, puede darse el caso que dos estudiantes encuentran en sus documentos distintas preguntas.

Los estudiantes responden sus preguntas rellenando los recuadros de su documento que se correspondan con las opciones que consideren correctas.

- **Paso 4:** Las respuestas proporcionadas por los estudiantes se determinan de forma automática mediante imágenes de los documentos que las contienen. Este proceso puede realizarse de dos formas, ambas con el uso de la línea de comando *scan*.

La primera forma es abrir el escaner de la herramienta, el cual utiliza una *webcam* para tomar imágenes de las respuestas a los exámenes generados.

```
$ autoexam scan
```

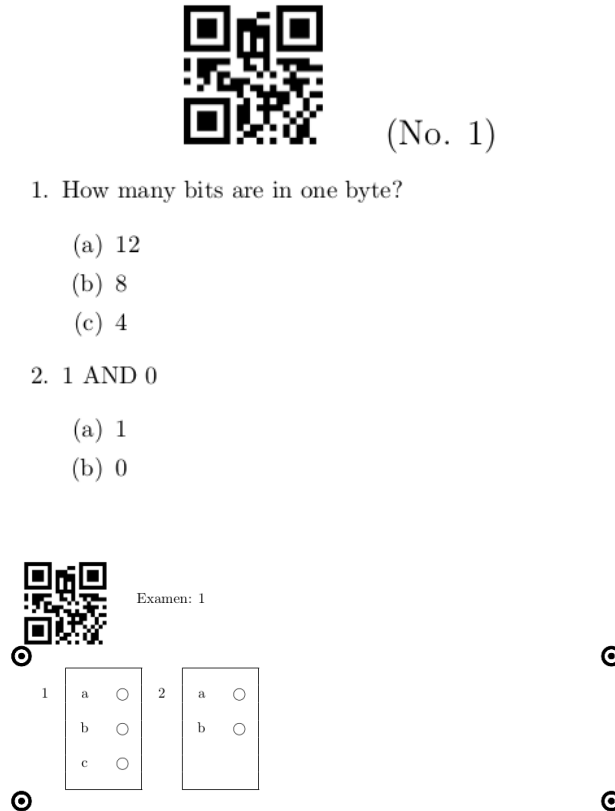


Figura 2.6: Par de documentos pertenecientes a un examen.

El usuario debe poner frente a la *webcam* todos los documentos con respuestas y esperar que la herramienta los detecte. Otra forma es tomar imágenes de estos documentos y almacenarlas en un directorio. Luego especificarle a la línea de comando *scan* la ubicación de este directorio.

```
$ autoexam scan -f [image_directory_path]
```

En ambas opciones, por cada documento con respuestas, se determinan los recuadros rellenos por el estudiante, los cuales corresponden a sus respuestas al examen.

- **Paso 5:** A partir de la clave de la versión del examen, almacenada en el fichero *grader.txt*, la herramienta determina las notas. Estas son

obtenidas mediante la suma de los puntos otorgados por seleccionar o no cada una de las opciones de respuesta a las preguntas del examen. Esto se realiza a través de la línea de comando *grade*, la cual usa el fichero (*grader.txt*) de la versión especificada. Las preguntas asociadas a las opciones se determinan a través del número identificador, es decir, son las preguntas que aparecen en el otro documento de igual identificador.

```
$ autoexam grader -v [version]
```

2.2. Propuesta

La idea con este proyecto es proporcionar una interfaz gráfica, a través de la creación de una aplicación web, que acompañe al usuario de la herramienta AutoExam durante todo el proceso descrito anteriormente. Dicha aplicación web permite la creación y edición de preguntas junto con sus opciones, la confección, generación y evaluación de exámenes, así como la recopilación de información estadística de los resultados y su muestra en tablas y gráficos para su posterior estudio.

La aplicación tiene una arquitectura basada en 7 entidades fundamentales: Asignatura, Etiqueta, Pregunta, Opción, Examen, Usuario y Rol (Ver Figura 2.7). Además, existen relaciones entre dichas entidades posibilitando la interacción entre ellas durante la ejecución de la aplicación. AutoExam no incluye el concepto de “asignatura”, sin embargo, se decidió añadir para permitir usar la aplicación como un repositorio de exámenes y preguntas, divididos según el contenido que evalúan.

Cada elemento de la entidad Usuario se relaciona con uno o varios elementos de la entidad Rol. Los usuarios son los encargados de interactuar con la aplicación y, por tanto, modificar la información contenida en ella en dependencia de los roles que tengan asignados. Estas dos entidades (Usuario y Rol) conforman un sistema de autenticación integrado en la aplicación.

La entidad Asignatura representa un área del conocimiento sobre la que se desean generar exámenes. Cada elemento de la entidad posee un nombre que lo identifica del resto. Los elementos de las entidades Pregunta y Examen tienen asociados un elemento de la entidad Asignatura.

Para categorizar el contenido de las asignaturas se usan los componentes de la entidad Etiqueta, definidos inicialmente al crear la asignatura pero que pueden ser modificados posteriormente.

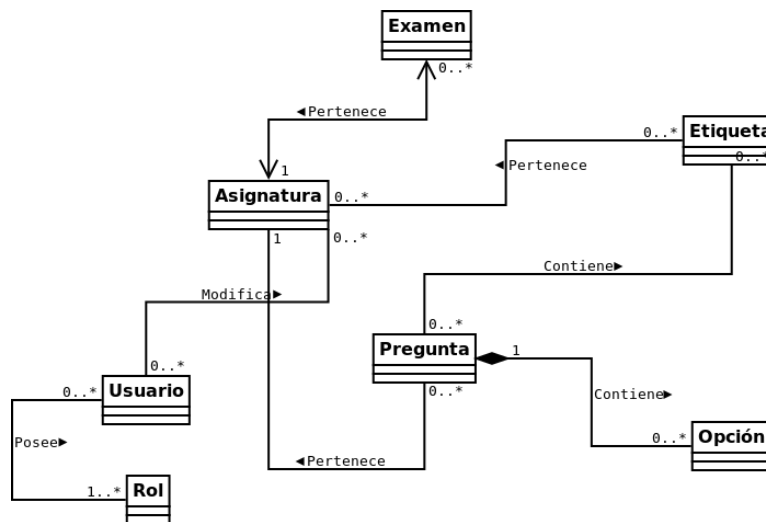


Figura 2.7: Modelo de la aplicación.

Los componentes de la entidad **Pregunta** son los encargados de evaluar el contenido de una asignatura y tienen asociados una orden o cuerpo y varios elementos de la entidad **Opción**.

Una opción contiene una posible respuesta al elemento de la entidad **Pregunta** al cual está asociado. Las opciones pueden ser verdaderas o falsas y determinan la evaluación del examen a partir de su selección o no.

Al crear o editar una pregunta se seleccionan las etiquetas de la asignatura asociada que mejor categorizan el contenido evaluado en la misma.

Examen es la entidad principal en la arquitectura de la aplicación. Cada elemento de esta entidad tiene asociados un título, un encabezado, una descripción y la información necesaria para evaluarlo posteriormente. Además, tiene un grupo de preguntas, pertenecientes a la misma asignatura, escogidas a partir de la selección de categorías del contenido a evaluar.

Durante la creación o edición de un examen se seleccionan las etiquetas que representan las categorías del contenido de la asignatura asociada que se desean evaluar y se generan los ficheros *master.txt* y *grader.txt*. El primero contiene las preguntas con sus respectivas opciones que aparecen en los documentos correspondientes al examen, mientras que el segundo la clave para evaluarlos. La clave está compuesta por los puntos otorgados por seleccionar o no cada una de las opciones de las preguntas que aparecen en el examen.

La aplicación facilita la interacción de los usuarios con la herramienta

AutoExam. A través de una interfaz gráfica los usuarios envían órdenes a la aplicación para generar los ficheros *master.txt* y *grader.txt*, crear o editar preguntas y opciones y confeccionar y evaluar los exámenes, entre otras. La aplicación interpreta estas órdenes y utiliza la herramienta AutoExam para cumplirlas, siendo este proceso invisible para el usuario.

2.3. ¿Cómo funciona la aplicación?

El proceso de generación y evaluación de un examen mediante la aplicación se resume de esta forma.

- **Paso 1:** El usuario selecciona la asignatura a la que pertenece el examen que se desea generar (el usuario debe tener permisos para modificar la información de dicha asignatura). En este punto el usuario puede modificar las etiquetas vinculadas a la asignatura (Ver Figura 2.8). Se deben incluir todas las etiquetas que representen el contenido del examen que se desea generar.
- **Paso 2:** Se crean las preguntas y opciones que pueden aparecer en el examen. Por cada pregunta se seleccionan las etiquetas que representan el contenido que evalúan. Las opciones son clasificadas en verdaderas o falsas, según la pregunta a la que dan respuesta (Ver Figura 2.9).
- **Paso 3:** El usuario crea un nuevo examen. En un primer momento se especifica el título, que lo diferencia del resto de los exámenes, y otras características como el encabezado y el total de estudiantes a evaluar. Además se seleccionan las etiquetas que representan el contenido a evaluar en el examen (Ver Figura 2.10).

Luego se especifican los puntos otorgados por marcar o no cada una de las opciones de las preguntas que pueden aparecer en el examen. Estas preguntas deben contener al menos una de las etiquetas seleccionadas para el examen. Además se establecen restricciones para la generación del examen como el número de preguntas en el examen y por etiquetas (Ver Figura 2.11).

- **Paso 4:** Después de creado el examen se genera una nueva versión. En este punto se obtienen los documentos pertenecientes a la misma, generados por la herramienta AutoExam (Ver Figura 2.12).

Paso 5: Finalmente se evalúan las respuestas mediante la webcam (Ver Figura 2.13). La aplicación muestra los resultados obtenidos en gráficos y tablas para su análisis.

The screenshot shows the 'AutoExam' application interface. At the top, there is a navigation bar with 'AutoExam' and 'Home' on the left, and 'Logged in as hd@matcom.uh.cu', 'Edit profile', and 'Logout' on the right. Below the navigation bar, on the left, is a sidebar titled 'My Signatures' with two links: 'Arquitectura de Computadoras' and 'Programacion de Maquinas'. The main content area is titled 'Editing Labels' and shows 'Signature: Arquitectura de Computadoras'. Below this, there is a section labeled 'Labels' with a text input field containing the text 'bits, easy, hard, medium'. At the bottom of this section are two buttons: 'Update Signature' and 'Back'.

Figura 2.8: Vista de modificación de etiquetas.

AutoExam

Home

Logged in as hd@matcom.uh.cu

Edit profile

Logout

My Signatures

Arquitectura de Computadoras

Programacion de Maquinas

Arquitectura de Computadoras

Labels [edit](#)

bits, easy, hard, medium

Exams

+ New Exam

Title	Header	Description	Labels	Amount	
Examen Final	Curso 2017-2018		bits, easy, hard	5	Show Edit Destroy
Examen Extraordinario	Curso 2017-2018		bits, easy, medium	10	Show Edit Destroy

Questions

+ New Question

Title	Body	Labels	Signature	
Options	How many bits are in one byte?	bits	Arquitectura de Computadoras	Show Edit Destroy
Options	1 AND 0	bits, easy	Arquitectura de Computadoras	Show Edit Destroy
Options	1 OR 0	bits	Arquitectura de Computadoras	Show Edit Destroy
Options	1 XOR 0	bits	Arquitectura de Computadoras	Show Edit Destroy

Figura 2.9: Vista de creación y edición de preguntas y opciones.

AutoExam

Home

Logged in as hd@matcom.uh.cu

Edit profile

Logout

My Signatures

Arquitectura de Computadoras

Programacion de Maquinas

New Exam

Title

Examen Final

Header

2017-2018

Description

Labels

Amount

30

Submit

Figura 2.10: Primera parte del proceso de creación de un examen.

AutoExam

Home

Logged in as [hd@matcom.uh.cu](#) [Edit profile](#) [Logout](#)

My Signatures

[Arquitectura de Computadoras](#)
[Programacion de Maquinas](#)

Examen Final

Curso 2017-2018

Arquitectura de Computadoras

Number of questions:

Labels:

☒ bits ☐ easy ☐ hard

How many questions:

Title	Body	Min Cost	Max Cost
<input checked="" type="checkbox"/> Options	1 AND 0	<input type="text" value="0"/>	<input type="text" value="1"/>

Options for: "1 AND 0"

Body	True or false	Checked	Unchecked
1	False	<input type="text" value="0"/>	<input type="text" value="1"/>
0	True	<input type="text" value="1"/>	<input type="text" value="0"/>

Submit

Figura 2.11: Segunda parte del proceso de creación de un examen.

The screenshot displays the AutoExam application interface. At the top, a navigation bar includes 'AutoExam' and 'Home' tabs, and a user status section indicating 'Logged in as hd@matcom.uh.cu' with links for 'Edit profile' and 'Logout'. On the left, a sidebar titled 'My Signatures' lists 'Arquitectura de Computadoras' and 'Programacion de Maquinas'. The main content area is titled 'Examen Final' and 'Curso 2017-2018'. It specifies the 'Signature: Arquitectura de Computadoras' and 'Version: 23'. Below this, there are three tabs: 'Tests', 'Answers' (which is active), and 'Statics'. Under the 'Answers' tab, a list of four PDF files is shown: 'Answer-0000.pdf', 'Answer-0001.pdf', 'Answer-0002.pdf', 'Answer-0003.pdf', and 'Answer-0004.pdf'. At the bottom of the list, there are three buttons: 'Download All' (green), 'Evaluate Answer' (blue), and 'Back' (blue).

Figura 2.12: Vista de una versión de un examen.

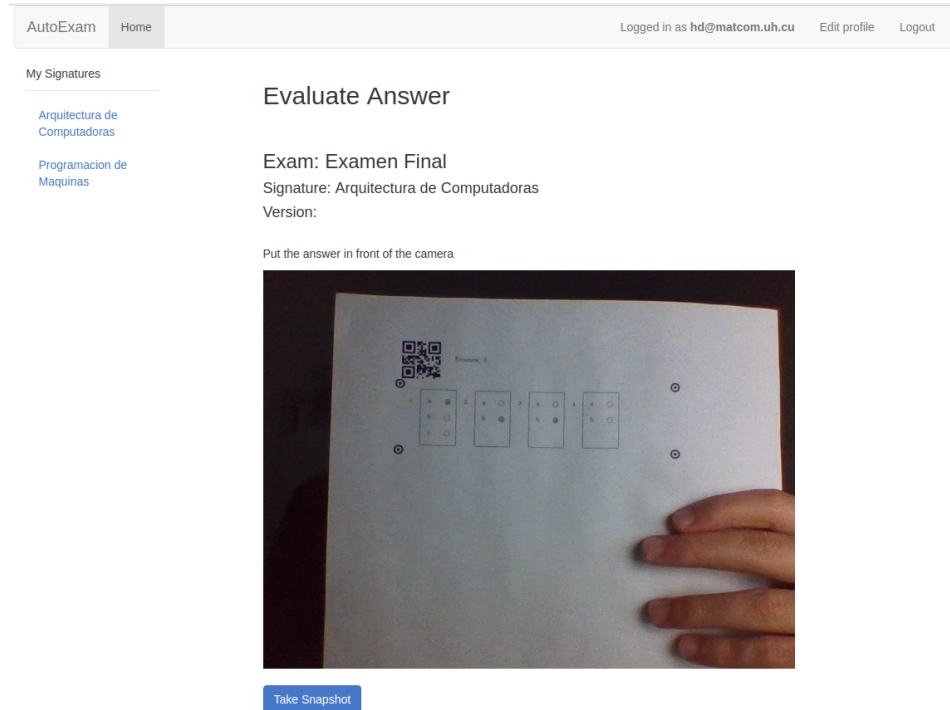


Figura 2.13: Evaluación de un examen a través de la *webcam*.

2.4. Funcionalidades brindadas

La aplicación provee a los usuarios un conjunto de funcionalidades que permiten el trabajo con la herramienta AutoExam.

1. **Creación y modificación de información:** El usuario puede crear, editar o eliminar preguntas, opciones y exámenes asociadas a las asignaturas de las que tiene permiso. Además puede modificar las etiquetas de estas asignaturas y las claves de los exámenes.
2. **Almacenamiento de información:** La aplicación guarda la información creada por los usuarios. Las asignaturas agrupan a preguntas y exámenes según su contenido. Los usuarios pueden utilizar para la creación de los exámenes preguntas asociadas a las asignaturas de las que tienen permisos, que fueron creadas por otros usuarios.

También pueden generar nuevas versiones de exámenes creados por otros, siempre que pertenezcan a asignaturas de las que tienen permisos.

3. **Generación de ficheros:** La aplicación genera diversos ficheros durante su ejecución. Los ficheros *master.txt* y *grader.txt* son generados durante la creación de exámenes. Además se generan los documentos correspondiente a los exámenes en formato .pdf. Estos documentos pueden ser descargados por el usuario directamente desde la aplicación.
4. **Generación de estadísticas:** Los usuarios tienen acceso a gráficos y tablas que contienen los resultados obtenidos durante la evaluación de los exámenes. Esto posibilita un análisis estadístico de dichos resultados para una mejor interpretación.
5. **Búsqueda de información:** La aplicación permite la búsqueda de información dentro de su base de datos. Por cada asignatura pueden obtenerse los resultados de buscar etiquetas, preguntas o exámenes que contengan dentro de su información semejanzas con patrones de textos deseados.

Capítulo 3

Implementación

3.1. Implementación

La aplicación contiene 7 modelos, ellos son Signatures, Questions, Options, Exams, Users, Roles y Teachers. Los modelos Signatures, Questions, Options y Exams se corresponden con las entidades Asignatura, Pregunta, Opción y Examen especificadas en el capítulo 2.2. Los modelos Users, Roles y Teachers se usan para la asignación de permisos a los usuarios de la aplicación. Teachers es usado específicamente para asociar los elementos de los modelos Users y Signatures, y de esta forma asignar permisos a los usuarios para modificar la información contenida en las asignaturas.

Las etiquetas son imprescindibles al momento de generar exámenes debido a que son las que determinan el conjunto de preguntas (en los documentos, solo pueden aparecer preguntas que posean alguna de las etiquetas seleccionadas para el examen). Además, por cada etiqueta se establecen restricciones en cuanto al mínimo y máximo número de preguntas pertenecientes a la misma que pueden aparecer en los documentos generados.

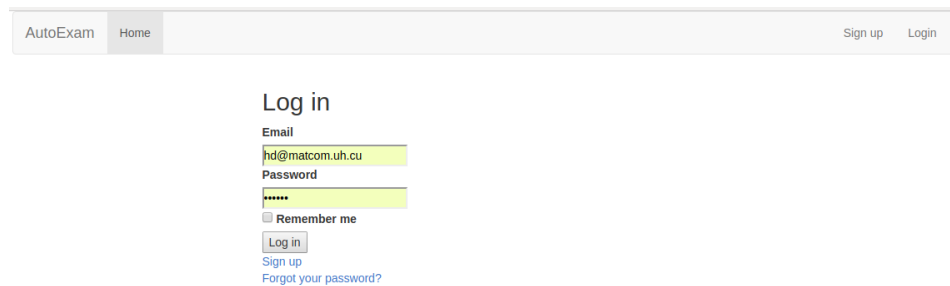
No existe un modelo para las etiquetas, estas se representan mediante un campo de Signatures y son insertadas separadas por comas. Se escoge esta representación por encima de crear un modelo y relacionarlo con Signatures y Questions para evitar consultas innecesarias a la base de datos, debido a que en ningún momento es necesario conocer una etiqueta a cuantas asignaturas pertenece. En la creación o edición de preguntas y exámenes se acceden a las etiquetas mediante la referencia a la asignatura asociada.

Estos modelos representan la información necesaria para permitir a los usuarios trabajar con la herramienta AutoExam y contenida en una base de

datos SQLite. Esta base de datos está estructurada por 7 tablas (una por modelo) y cada una contiene diversos campos para almacenar la información necesaria para la ejecución de la aplicación.

3.2. Usuarios y Roles

La aplicación posee un sistema de autenticación de usuarios implementado con el uso de la gema *devise* (Ver Figura 3.1). Esta gema proporciona una solución de autenticación flexible para Rails con Warden [4].



The screenshot shows the top navigation bar of the 'AutoExam' application with links for 'AutoExam', 'Home', 'Sign up', and 'Login'. Below the navigation bar is a 'Log in' form. The form contains an 'Email' field with the value 'hd@matcom.uh.cu', a 'Password' field with masked characters '*****', a 'Remember me' checkbox, and a 'Log in' button. Below the button are links for 'Sign up' and 'Forgot your password?'.

Figura 3.1: Sistema autenticación de usuarios en la aplicación.

Los usuarios deben tener una dirección de correo electrónico válida para poder registrarse en la aplicación. Una vez registrados, su información es guardada en la base de datos, específicamente en la tabla *Users*, encriptándose algunos campos como *password* para mayor seguridad.

Cada usuario posee roles dentro de la aplicación. Existen dos tipos de roles (Admin, Professor). Cuando un nuevo usuario se registra en el sitio tiene automáticamente el rol de Professor. Inicialmente existe un único usuario con rol de Admin, el cual es el encargado de crear las asignaturas y asignarlas a los usuarios. Además, este tipo de usuario puede cambiar los roles de los otros usuarios y/o eliminarlos de la aplicación.

Los usuarios con rol de Professor pueden acceder únicamente a las asignaturas que les han sido asignadas. Dentro de estas pueden crear, editar o eliminar preguntas, opciones, etiquetas de contenido, así como generar nuevos exámenes.

3.3. Interacción con AutoExam y generación de ficheros

La aplicación funciona como una especie de mediador entre el usuario y la herramienta AutoExam. La interfaz gráfica recibe del usuario órdenes de crear preguntas, generar exámenes, entre otras y las convierte en líneas de comandos de AutoExam (Ver capítulo 2.1.1), los cuales ejecuta a través de llamadas al sistema operativo. Es por eso que la aplicación interactúa en el *back-end* con la herramienta AutoExam.

Cuando un usuario crea un examen en la aplicación, esta a su vez, crea un nuevo proyecto de AutoExam, a través de la ejecución del comando *new*, mediante una llamada al sistema operativo. Este proyecto es creado en el directorio *generated*, ubicado en la raíz del proyecto, y con el nombre *Exam-id_del_exam*.

Cuando un examen se modifica, el fichero *master.txt* correspondiente es modificado por la aplicación, guardando en él las preguntas seleccionadas por el usuario, junto con sus opciones de respuestas y etiquetas de contenido. Además, se almacenan las restricciones usadas en la generación de los documentos, como el número de preguntas que deben aparecer en cada uno.

Otro momento es al generarse una nueva versión de un examen. La aplicación ejecuta el comando *gen* mediante una llamada al sistema operativo. Posteriormente modifica el fichero *grader.txt*, que contiene la clave del examen. En él se guardan los puntos otorgados por seleccionar o no cada una de las opciones de las preguntas que pueden aparecer en los documentos. Este fichero no es editado en ningún otro momento.

Finalmente cuando el usuario desea evaluar las respuestas a un examen, la aplicación vuelve a interactuar con la herramienta AutoExam. Esto se produce mediante la ejecución del comando *scan* mediante una llamada al sistema operativo. El proceso de evaluación de los exámenes es descrito en el capítulo 3.4.

La aplicación también genera otros ficheros que no interactúan con AutoExam. La mayoría son explicados en el capítulo 3.4, pues son utilizados durante la evaluación de los exámenes. Los usuarios pueden descargar los documentos de un examen. En ese caso la aplicación genera dos ficheros con extensión *.tar*, *Answers.tar* y *Tests.tar*, los cuales son proporcionados a los usuarios cuando estos solicitan las descargas.

3.4. Evaluación de exámenes

El proceso de evaluación de exámenes se realiza con la ayuda de una *webcam*. La integración de la *webcam* a la aplicación (Ver Figura 2.13) fue implementada usando la biblioteca de Javascript *Webcam.js*. *Webcam.js* permite capturar imágenes desde la computadora del usuario a través de su *webcam* y entregarlas al servidor de la aplicación [12].

La aplicación muestra las imágenes obtenidas desde la *webcam* del usuario en una de sus vistas, utilizando un canvas. El usuario debe tomar una imagen desde la aplicación, del documento con respuestas que se desea evaluar. Esta imagen es procesada por el servidor y almacenada en un directorio temporal *temp*. Posteriormente la aplicación ejecuta el comando *scan*, indicándole la ruta del directorio *temp*, para que AutoExam examine la imagen almacenada en este directorio. Los resultados son guardados por la herramienta en un fichero llamado *results.json*. La aplicación lee e interpreta este fichero para determinar la nota y guardar los resultados en la base de datos para posteriormente generar tablas y gráficos estadísticos. En caso de existir algún error con la imagen proporcionada durante el proceso, la aplicación notifica al usuario para que este tome una nueva imagen. Finalmente se eliminan del servidor el directorio *temp* y la imagen capturada.

Los resultados obtenidos en los exámenes generados se muestran en tablas y gráficos para poder realizar análisis estadísticos (Ver Figura 3.2). Por cada examen se genera una tabla que muestra las puntuaciones obtenidas por los estudiantes. En la primera columna se muestran los identificadores de los estudiantes evaluados. El resto de las columnas corresponden a las preguntas del examen y muestran las puntuaciones de cada estudiante en la pregunta correspondiente.

Los gráficos son generados mediante la biblioteca de Javascript C3.js. En ellos se muestran datos estadísticos por cada una de las preguntas de los exámenes (Ver Figura 3.2). Algunos de estos datos son, los porcentajes de respuestas correctas e incorrectas a dicha pregunta y el número de veces que los estudiantes han seleccionado cada una de las opciones de respuesta.

Question 1

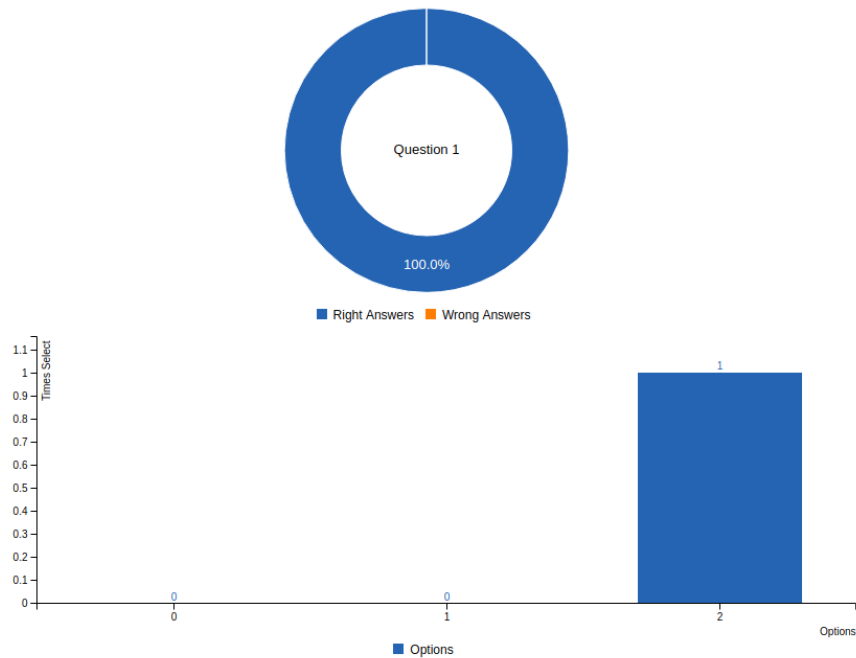


Figura 3.2: Algunos gráficos generados por la aplicación.

3.5. Búsqueda

La aplicación permite realizar búsquedas dentro de la base de datos. Un usuario puede buscar dentro de una de las asignaturas de las que tiene asignado permisos, aquellas preguntas, opciones y/o exámenes que contengan entre sus datos una cadena de texto determinada. La búsqueda puede realizarse también dentro de las etiquetas de la asignatura, o utilizar una etiqueta como patrón de texto y obtener de esta forma todas las preguntas y exámenes que la contienen.

Los resultados se muestran en una sola vista, separados según su tipo. Primeramente se muestran los resultados obtenidos en las etiquetas de la asignatura. Seguidamente se muestran en ese orden el conjunto de exámenes y de preguntas junto con sus opciones cuyos datos tienen alguna coincidencia con el patrón de texto deseado (Ver Figura 3.3).

AutoExam

Home

Logged in as [hd@matcom.uh.cu](#) [Edit profile](#) [Logout](#)

My Signatures

[Arquitectura de Computadoras](#)
[Programacion de Maquinas](#)

Arquitectura de Computadoras

Search Results for "bits" ...

Labels

bits

Exams

Title	Header	Description	Labels	Amount	
Examen Final	Curso 2017-2018		bits, easy, hard	5	Show Edit Destroy
Examen Extraordinario	Curso 2017-2018		bits, easy, medium	10	Show Edit Destroy

Questions

	Title	Body	Labels	Signature	
Options		How many bits are in one byte?	bits	Arquitectura de Computadoras	Show Edit Destroy
Options		1 AND 0	bits, easy	Arquitectura de Computadoras	Show Edit Destroy
Options		1 OR 0	bits	Arquitectura de Computadoras	Show Edit Destroy
Options		1 XOR 0	bits	Arquitectura de Computadoras	Show Edit Destroy

Figura 3.3: Resultados de búsqueda en la aplicación.

Conclusiones

Se logró la creación de una interfaz gráfica para la herramienta AutoExam. La misma brinda a los usuarios un ambiente sencillo, capaz de interactuar con los componentes principales de la herramienta. Además, permite la generación y evaluación de exámenes, así como el análisis estadístico de los resultados obtenidos.

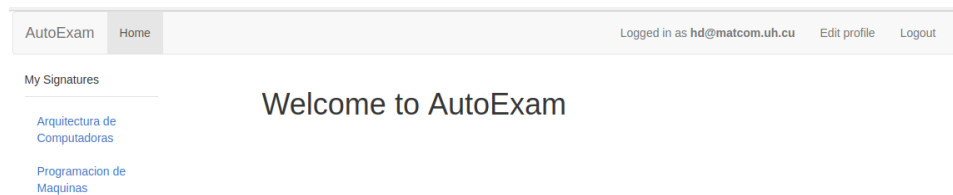


Figura 3.4: Vista principal de la aplicación.

Durante el proceso se revisó la documentación de la herramienta AutoExam, así como cada uno de las componentes de su implementación. También fueron analizadas herramientas similares, las que ayudaron a realizar un diseño de nuestra aplicación, que agrupara algunas de las componentes visuales más atractivas para los usuarios.

Al finalizar la implementación de la aplicación se realizaron pruebas de todas las funciones de nuestro proyecto, obteniéndose buenos resultados. Se crearon varios exámenes para simular los procesos de generación y evaluación con el uso de la aplicación. Además se usaron para probar la generación de reportes y gráficos estadísticos.

Durante todo el proceso se tuvieron en cuenta las opiniones de posibles usuarios de la aplicación para el perfeccionamiento de los componentes

visuales y el facilitamiento de la interacción con la herramienta AutoExam. Finalmente podemos concluir que la aplicación quedó en un estado en que se encuentra lista para su uso.

Recomendaciones

Al finalizar este trabajo se discutieron algunas recomendaciones para continuar mejorando la aplicación:

- Mejorar la seguridad de la aplicación a través de la conversión de la misma en una aplicación portable.
- Integrar *openCV* a la aplicación para agilizar el escaner de los documentos de un examen durante su evaluación.
- Actualizar la interfaz gráfica a partir de la opinión de los usuarios de la aplicación.

Bibliografía

- [1] About ruby. <http://www.ruby-lang.org/en/about>.
- [2] Bootstrap, the most popular html, css and js library in the world. <http://www.getbootstrap.com>.
- [3] C3.js | d3-based reusable chart library. <http://www.c3js.org>.
- [4] devise. <http://www.rubygems.org/gems/devise>.
- [5] Easy online grader, grading app for teachers from gradecam. <http://www.gradecam.com>.
- [6] Github - matcom/autoexam: A simple exam generator and grader written in python with opencv. <http://github.com/matcom/autoexam>.
- [7] Modelo-vista-controlador. <http://www.es.wikipedia.org/wiki/Modelo-vista-controlador>.
- [8] Ruby on rails. <http://www.rubyonrails.org>.
- [9] Ruby on rails - wikipedia. http://www.es.wikipedia.org/wiki/Ruby_on_Rails.
- [10] Sqlite. <http://www.sqlite.org>.
- [11] Sqlite - wikipedia. <http://www.en.wikipedia.org/wiki/SQLite>.
- [12] Webcamjs | pixlcore. <http://pixlcore.com/read/WebcanJS>.
- [13] Alexis Bienvenüe, Anirvan Sarkar, Hiroto Kagotani, and Frédéric Bréal. *Auto Multiple Choice*. March 2016.