Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

SISTEMA PARA GESTIÓN DE PROBLEMAS PARA COMPETENCIAS DE PROGRAMACIÓN

Br. Rosmeli Quintero

Prof. Héctor Navarro, Tutor

Caracas, 19 de mayo del año 2017

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

SISTEMA PARA GESTIÓN DE PROBLEMAS PARA COMPETENCIAS DE PROGRAMACIÓN

Br. Rosmeli Quintero

Prof. Héctor Navarro, Tutor

Caracas, 19 de mayo del año 2017

SISTEMA PARA GESTIÓN DE PROBLEMAS PARA COMPETENCIAS DE PROGRAMACIÓN

Br. Rosmeli Quintero

Trabajo Especial de Grado presentado

ante la ilustre Universidad Central de Venezuela

como requisito parcial para optar al título de

Licenciado en Computación.

Prof. Héctor Navarro, Tutor Fecha

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Centro de Computación Gráfica

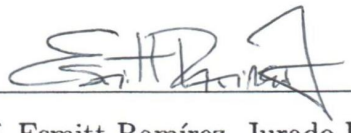
ACTA DEL VEREDICTO

Quienes suscriben miembros del Jurado, designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el **Br. Rosmeli Quintero, C.I. 21310353**, titulado: "**Sistema para gestión de problemas para Competencias de Programación**" a los fines de optar al título de Licenciado en Computación, dejan constancia lo siguiente:

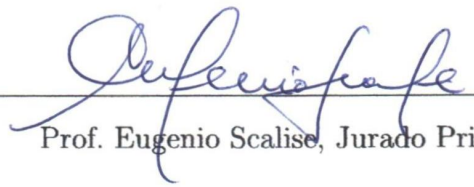
Leído como fue, dicho trabajo por cada uno de los miembros del Jurado, se fijó el día 18 de mayo del 2017 a las 8:00 a.m., para que su autor lo defendiera en forma pública, en Escuela de Computación de la Facultad de Ciencias, mediante una presentación oral de su contenido. Finalizada la defensa pública del Trabajo Especial de Grado, el Jurado decidió aprobarlo.

En fé de lo cual se levanta la presente Acta, en Caracas a los dieciocho días del mes de mayo del año dos mil diecisiete, dejándose también constancia de que actuó como Coordinador del Jurado el Prof. Héctor Navarro.

Prof. Héctor Navarro, Tutor



Prof. Esmitt Ramírez, Jurado Principal



Prof. Eugenio Scalise, Jurado Principal

Agradecimientos

A mis padres, por entre todas las cosas, enseñarme que la educación es lo más importante y motivarme desde siempre a seguir esta meta. Por su humor y su apoyo. Por ustedes soy lo que soy.

A Luis, mi amor, por ~~apoyarme~~ aguantarme todos los días de estrés incondicionalmente. Siempre con una sonrisa para mi. Por aplicarme siempre la técnica del Rubber Duck, por todos "¿Y la tesis?"

A todos los que en mi carrera me ayudaron a seguir adelante, a los profesores y preparadores. A la Facultad de Ciencias, mi Facultad y mi Universidad Central de Venezuela, por seguir venciendo la sombra hasta en los momentos más difíciles.

A Héctor, mi tutor **...to be continued.**

Resumen

Sistema para gestión de problemas para Competencias de Programación

Rosmeli Quintero

Prof. Héctor Navarro, Tutor

Universidad Central de Venezuela

El siguiente trabajo especial de grado describe la problemática de la Escuela de Computación de la Facultad de Ciencias de la UCV para gestionar la información referente al problemario en una competencia de programación.

Al ser una actividad importante en el ámbito y conociendo las necesidades e inconvenientes existentes se plantea el desarrollo de una aplicación web centralizada que permita la gestión de enunciados de problemas para competencias de programación haciendo uso del framework *Django* y tecnologías como *HTML, CSS, JavaScript*. Además se utiliza LaTeXpara la generación de enunciado en formato *PDF* y *Git* para el manejo de control de versiones.

Palabras clave: competencias de programación, problemario, aplicación web, Django, LaTeX, control de versiones.

Prof. Héctor Navarro
Tutor

Índice General

Lista de Figuras

Introducción

La tecnología y la computación, así como el estudio de las ciencias de la computación son áreas que han estado en constante crecimiento en los últimos años, y con ello, se han creado diversas actividades relacionadas a estas, entre ellas, la programación competitiva.

La programación competitiva no sólo representa un reto como competidor sino también como organizador. La faceta de organización incluye muchas tareas que deben ser realizadas con cuidado, seguridad y eficiencia; entre ellas, la creación y organización del problemario que será llevado a competencia. Esta tarea incluye: redacción de problemas, propuestas de soluciones, votación y discusión.

La Escuela de Computación de la Facultad de Ciencias de la UCV no tiene una herramienta que permita realizar estas tareas. Los profesores y estudiantes de la misma han adaptado un foro para compartir la información pero este método trae varios problemas de usabilidad, eficiencia y consistencia.

El objetivo de este Trabajo Especial de Grado es el desarrollo de este sistema web que se encargará de gestionar y centralizar ágilmente las tareas relacionadas para la creación del problemario aplicando conceptos y tecnologías actuales.

Para lograr esto, se estructuró este Trabajo Especial de Grado en siete (7) capítulos: capítulo 1, "Planteamiento del Problema", donde se explora y justifica el problema a ser resuelto. Luego siguen los capitulos 2, 3, 4 y 5, "Programación Competitiva", "LaTeX", "Desarrollo Webz "Tecnologías", respectivamente, que abarca los conceptos relevantes y necesarios para el desarrollo de esta investigación. El capítulo 6, "Diseño",

se describe la arquitectura y estructura de la solución. El capítulo 7, "Implementación", donde se explica en detalle cómo se desarrolló la aplicación y, para culminar, el capítulo 8, "Pruebas", donde se evalúa la experiencia de los usuarios con la aplicación.

Capítulo 1

El problema de investigación

1.1. Planteamiento del problema

Al momento de realizar una competencia de programación es necesario que varias personas redacten problemas y escriban su solución y casos de prueba correspondientes. Generalmente estos problemas suelen ser discutidos y analizados entre todos los organizadores para luego, a través de una votación, elegir los problemas finales que se llevarán a competencia.

Adicionalmente, es preferible que estos enunciados tengan una alta calidad tipográfica; específicamente elementos como formulas matemáticas, tablas o teoremas, sin la necesidad que los autores dediquen mucho tiempo en la presentación del documento.

Actualmente, para la *Universidad Central de Venezuela* y específicamente para la *Escuela de Computación* en la *Facultad de Ciencias*, este proceso es complicado ya que no se posee de un sistema centralizado y usable que permita organizar y llevar a cabo fácilmente una competencia de programación. Normalmente se organizan muchas competencias de programación al año y sólo se tiene un sitio donde realizar las gestiones de problemas, este sitio es un foro adaptado para compartir información, donde: los problemas pueden ser subidos en cualquier formato (*PDF*, TEX, o incluso *TXT*), lo que puede traer problemas de visualización; no existe un control de versiones, las votaciones

deben ser sumarizadas manualmente, es necesario navegar entre muchos *links* para llegar a la información deseada, entre otros factores que entorpecen el proceso y da cabida a errores. Por otro lado, existen otras universidades, como la *Universidad de Valladolid (UVa)*, que si proveen una creación y edición controlada de problemas en el mismo sitio (*on site*) pero las votaciones y elecciones de problemas a competencia deben realizarse en un sitio diferente.

1.2. Justificación

La programación competitiva constituye un componente importante para los estudiantes afines a las Ciencias de la Computación ya que permiten un crecimiento a nivel cognitivo, específicamente en las áreas de la lógica y matemática. Por estas razones, constituye una buena idea incentivar a los estudiantes en la participación de maratones de programación, lo que llevaría, naturalmente, a la creación y organización de los mismos.

Anteriormente, se plantearon las dificultades que conllevan a organizar una competencia. Por lo que es necesario un sistema que facilite el proceso. Esto traería consecuencias favorables para los organizadores, ya que podrían centrarse en los problemas de programación y no en tareas adicionales para la construcción de la competencia.

1.3. Objetivo general

Desarrollo de un sitio web que ayude a la creación y gestión de problemas para competencias de programación.

1.4. Objetivos específicos

- Analizar los datos que se manejarán en el sitio y establecer un modelo de datos.

- Permitir a los usuarios redactar un problema y añadirlo a una competencia.
- Generar automáticamente un archivo en formato TEX y a su vez un compilado en formato *PDF* por cada problema.
- Ofrecer la opción de visualizar el avance del problema en formato *PDF* mientras se redacta el mismo.
- Permitir a los usuarios enviar por cada problema el código fuente de su solución y casos de prueba.
- Proporcionar un foro de discusión por cada problema.
- Manejar control de versiones.
- Proveer un sistema de votación de problemas por cada competencia.
- Implementar módulo que permita la descarga de todos los archivos necesarios para la realización de la competencia.
- Asegurar la seguridad de los datos referente a una competencia.
- Realizar pruebas de usabilidad con usuarios finales.

1.5. Herramientas

Para lograr los objetivos descritos, es necesario hacer uso de las siguientes herramientas:

- Django (y por extensión, Python) como framework web para el desarrollo del lado del servidor.
- HTML, CSS y JavaScript para el desarrollo de interfaces y métodos de interacción del lado del cliente.

- TEX y LATEX para la generación de enunciados compilados y no compilados.
- *MySQL* como manejador de Base de Datos.

1.6. Metodología

Para la implementación de esta apliacación se utilizará una metodología ad hoc orientada a prototipos.

Capítulo 2

Programación Competitiva

La programación competitiva, clasificada como un deporte mental, consiste en competencias donde uno o varios participantes escriben un algoritmo que de solución a un problema con ciertas especificaciones dadas. El o los ganadores son los que logren resolver la mayor cantidad de problemas dentro del límite de tiempo establecido por la competencia por lo que la competitividad recae en el tiempo. Estas competencias pueden ser a través de Internet (no presencial) o locales (presenciales).

Existen competencias que restringen el lenguaje de programación a ser usado. Entre los más populares encontramos *C*, *C++* y *Java*, pero hay competencias que admiten también soluciones en *Python*, *Pascal*, entre otros.

La programación competitiva es considerada una de las actividades extra - curriculares más valoradas en el área de las Ciencias de la Computación, pues ponen a prueba la velocidad de análisis de los participantes, siendo esta una característica deseable para cualquier puesto de trabajo.

2.1. Problemas de Programación

Un problema consiste en una pregunta a ser respondida, en donde el competidor debe escribir un algoritmo donde aplique técnicas o conceptos matemáticos y/o

computacionales para llegar a una solución. Resolver el problema lo más rápido posible es donde recae la competitividad. Un problema está estructurado de la siguiente forma:

- Descripción del problema: consiste generalmente en una explicación de cuál es la pregunta a ser respondida y por qué se quiere la respuesta o qué uso se le dará a la misma. Algunas descripciones contienen muchos detalles que no son de gran ayuda al competidor y hacen parecer el problema más complicado de lo que en realidad es.
- Descripción de entrada y salida: se detalla el formato que tendrán los datos entrada y también el formato que debe tener la salida. Generalmente estos formatos son muy estrictos, y si no se respetan adecuadamente se tomará la solución como errada.
- Ejemplos de entrada y salida: el autor provee un ejemplo sencillo o caso de prueba con una entrada y salida para que el competidor pueda probar su entendimiento básico del problema y compruebe si los formatos de entrada y salida de su solución son correctas.

La solución a un problema de programación puede abarcar un sin fin de conceptos en el área de la matemática y lógica como: combinatorias, teoría de números, teoría de grafos, geometría, estructuras de datos, entre otras.

La eficiencia es un concepto muy importante en este ámbito, ya que algunos problemas establecerán límites en memoria y/o tiempo, que pueden aumentar de gran manera el nivel de dificultad del mismo.

2.2. Juez

Es un sistema automatizado para la prueba de problemas de programación. Cada problema tiene asociado un conjunto de casos de prueba con entrada y salida, estos

casos de prueba son generalmente realizados por el autor del problema y probados con su propia solución. Cuando un competidor envía el código fuente de su solución a un problema, el juez compila y ejecuta con todos los casos de prueba con la posible solución y compara con las salidas esperadas.

Entre las respuestas más comunes se encuentran:

- Accepted (AC): respuesta correcta.
- Wrong Answer (WA): respuesta incorrecta.
- Time Limit Exceeded (TLE): límite de tiempo excedido.
- Memory Limit Exceeded (ML): límite de memoria excedido.
- Presentation Error (PE): errores en el formato de la salida.
- Compile Error (CE): error de compilación.

2.3. Competencias Actuales

Existe una gran cantidad de organizaciones que apoyan la programación competitiva, estas crean plataformas y jueces para incentivar a estudiantes a competir e incluso para reclutar trabajadores.

2.3.1. Olimpiada Internacional de Informática (IOI)

Iniciada por la *UNESCO* (Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura), la *IOI*[2] es una de las 5 Olimpiadas Internacionales de la Ciencia, creada para estimular el interés de los estudiantes por las Ciencias de la Computación.

La *IOI* es organizada anualmente alrededor del mundo, cada país participante envía a 4 competidores menores de 20 años y 2 acompañantes a competir individual-

mente por dos días. Los participantes pueden ganar medallas de oro, plata o bronce según las puntuaciones obtenidas durante los 2 días de competencia.

2.3.2. Competencia Internacional Universitaria ACM de Programación (ACM-ICPC)

El *ACM-ICPC* nace en la Universidad A&M en Texas, Estados Unidos, en el año 1970 como una competencia pequeña. Con el paso de los años, y el apoyo de la Asociación de los Sistemas Informáticos (*ACM*)[1] llegó a ser una reconocida competencia mundial orientada a estudiantes universitarios.

El *ACM-ICPC* es organizado anualmente. La representación de cada país está conformada por un equipo de máximo 3 estudiantes que intentarán resolver entre 8 y 10 problemas en cada fase clasificatoria. La representación de cada país es elegida durante 2 fases: local y regional.

2.3.3. Competencias a través de Internet

Las competencias a través de Internet han tenido un gran auge en los últimos años. Existen muchas plataformas dedicadas a organizar y realizar competencias y, además, proveer de un ambiente de aprendizaje en el que un participante intenta resolver un problema sin límite de tiempo.

Muchas compañías están interesadas en encontrar talento a partir de estas competencias, y lanzan sus propios concursos, como el caso de *Google* y el *Google Code Jam* [3] y *Facebook* con el *Facebook Hacker Cup* [4] donde los ganadores no solo obtienen premios monetarios sino la oportunidad de un puesto de trabajo en la compañía.

Por otro lado, las plataformas que proveen ambientes de aprendizaje son muy populares entre los competidores, entre las más reconocidas están: *TopCoder* [5], *Hacker Rank* [6], *Codeforces* [7], *SPOJ* [8], entre otras.

Capítulo 3

TEX

TEX es un sistema de composición tipográfica de textos (del inglés, *Typesetting* [9]) creado por Donald Knuth y lanzado en su primera versión en el año 1978. TEX tiene dos metas principales: permitir que cualquiera pueda producir documentos de alta calidad usando el mínimo esfuerzo, y proveer un sistema que diera exactamente el mismo resultado en cualquier computadora en cualquier momento de tiempo.

3.1. Características principales

Se podría decir que TEX es un procesador de textos pero esta afirmación está errada. Un procesador de texto se enfoca en la entrada y preparación del texto, sin embargo, esto no está dentro de las tareas de TEX, este se enfoca en la composición del texto, es decir, la forma en que cada letra está posicionada en relación a las demás y, en la forma en la que el texto se distribuye en las líneas.

TEX da uso de sofisticadas técnicas de programación para cuidar aspectos como el espaciado en la justificación de texto, la separación correcta por guión de las palabras, evitar las líneas huérfanas, entre otros. El sistema TEX tiene un preciso conocimiento de los tamaños de todos los caracteres y símbolos, usando esa información computa la disposición optima de letras por linea y lineas por página.

TEX es muy popular para el uso de expresiones matemáticas complejas, pues el creador le prestó mucha atención a esta funcionalidad, analizando el espaciado y posicionamiento de y entre cada elemento para que fuese correcto y de agrado para la comunidad científica.

3.2. El sistema

La entrada para el sistema TEX consiste en una serie de comandos, comúnmente iniciados con una barra invertida, dipuestos en un archivo de texto plano.

El sistema compila texto plano y genera un archivo *.DVI* (del inglés **de**v**v**ice **i**ndependent, independiente del dispositivo) que contiene la posición final de todos los caracteres. Este archivo consiste en datos binarios que deben ser leídos por otro programa para generar un archivo gráfico, como por ejemplo, *dvipdf* que convierte archivos del formato *DVI* al formato *PDF*.

3.3. Uso

Actualmente TEX es subministrado en un ejecutable con todas las fuentes, formatos de documentos y utilidades necesarias para utilizar el sistema. Para los sistemas operativos basados en *UNIX* incluyendo *Mac OS X*, TEX es distribuido en la forma de *teTex* [10] y más recientemente *TeX Live* [11]. Para el sistema operativo *Microsoft Windows* es suministrada la distribución *MiKTeX* [12]

3.4. LATEX

LATEX es un conjunto de comandos de alto nivel para la producción de documentos complejos que permite que cualquier usuario (con o sin conocimientos de programación y/o composición tipográfica) sea capaz de utilizar el sistema.

LaTeX fue creado en 1984 por Leslie Lamport con la intención de facilitar el uso de TeX. Contiene facilidades para automatizar encabezados, secciones, tablas, numeración de ecuaciones, citaciones, gráficas matemáticas, entre otras. Provee comandos para definir la estructura general del documento, ya sea artículo, libro, carta o reporte.

A diferencia de procesadores de texto como *Microsoft Office Word*, LaTeX separa muy bien la presentación y el contenido para que el autor se concentre en el contenido sin tener que que atender simultáneamente la apariencia del documento, pero aún así permitiendo ajustes manuales en caso de ser necesario.

3.4.1. Paquetes

Una característica importante para LaTeX son los paquetes. Estos son extensiones de comandos LaTeX que se son cargados y proveen de funcionalidades o usos. Se pueden clasificar los paquetes por su origen:

- Núcleo: paquetes que son parte integral de la instalación básica de LaTeX.
- Herramientas: paquetes escritos por los miembros del Equipo LaTeX3 .
- Gráficos: paquetes estandarizados para incluir imágenes generadas por otros programas y para manejar colores.
- $\mathcal{AMS}$-LaTeX: paquetes publicados por la Sociedad Americana de Matemática ($\mathcal{AMS}$).
- Aportes: paquetes enviados por usuarios.

Existen más de 1000 paquetes escritos y aportados por usuarios, muchos con utilidades que no cualquier usuario necesitaría y muchos que se han vuelto prácticamente escenciales para el uso de LaTeX.

3.4.2. Funcionamiento

Al estar basado en TEX, la barra invertida y las llaves conservan el mismo uso.

Un archivo LATEX consiste en texto plano, creado en cualquier editor de texto, con etiquetas o comandos LATEX. Generalmente la extensión de estos archivos es *.TEX*. El procesador de textos interpreta las órdenes escritas en él y compila el documento, dejándolo preparado para que pueda ser enviado a la salida correspondiente, ya sea la pantalla o la impresora. Si se quiere añadir o cambiar algo en el documento, se deberán hacer los cambios en el archivo fuente y compilarlo de nuevo.

LATEX conserva las ventajas de TEX al ofrecer siempre la misma salida sin importar el dispositivo o sistema operativo que se esté usando, y también puede ser exportado de una misma fuente a numerosos formatos como *Postscript*, *PDF*, *SGML*, *HTML*, *RTF*, entre otros.

Actualmente, existen una gran variedad de ambientes de desarrollo integrados (*IDE*) que combinan un editor de texto, visor integrado, manejo de errores, completación de comandos, entre otras utilidades. Estos pueden ser como aplicaciones de escritorio y basados en WEB con el uso de Internet. Los IDEs más populares son: *TEXmaker*, *TEXstudio*, *TEXworks*, *ShareLATEX*, *Overleaf*, *Authorea*, entre muchos otros.

3.4.3. Usos y comandos

3.4.3.1. Clases de documentos

Originalmente LATEX provee de cuatro clases para un documento:

- Artículo (`article`): destinado a artículos cortos. No poseen capítulos, y el título sólo aparece centrado al principio de la primera página seguido del contenido.
- Reporte (`report`): destinado a documentos técnicos más largos. Similar a *artículo*, excepto que contiene capítulos y el título tiene una página dedicada a él.

- Libro (`book`): destinados a libros para publicación. La distribución de la página es ajustada asumiendo que eventualmente será impreso por ambas caras del papel.
- Carta (`letter`): dedicado a cartas personales. Permite producir una carta con todos sus elementos (direcciones, fecha, firmas) distribuidos correctamente.

Las clases pueden ser modificadas con opciones, como: el puntaje de la fuente a ser usada, si el contenido debe estar distribuido en dos columnas, si el formato debe ser adecuado para impresión de cada cara, entre otras.

El comando para asignar una clase y opciones a un documento es:

```
documentclass[opciones]{clase}
```

Sin embargo, los usuarios pueden desarrollar clases y opciones según los requerimientos que estos tengan, y actualmente estos desarrollos son compartidos en la comunidad de LaTeX para su uso.

3.4.3.2. Estructura básica

LaTeX provee comandos para estructurar un documento, entre las más comunes tenemos:

- `\chapter{nombre}` añade un capítulo al documento. No disponible en la clase `artículo`.
- `\section{nombre}`, `\subsection{nombre}`, `\subsubsection{nombre}` que generan un título con un puntaje y numeración según el nivel de anidamiento.
- `\begin{itemize} \item ... \item \end{itemize}` que permite generar listas con viñetas.
- `\begin{enumerate} \item ... \item \end{enumerate}` que permite generar listas enumeradas.

3.4.3.3. Tablas

Con el entorno `tabular` se definen las tablas en un documento.

Por ejemplo:

```
\begin{tabular}{l*{6}{c}r}
    Team              & P & W & D & L & F  & A & Pts \\
\hline
    Manchester United & 6 & 4 & 0 & 2 & 10 & 5 & 12  \\
    Celtic            & 6 & 3 & 0 & 3 &  8 & 9 &  9  \\
    Benfica           & 6 & 2 & 1 & 3 &  7 & 8 &  7  \\
    FC Copenhagen     & 6 & 2 & 1 & 3 &  5 & 8 &  7  \\
\end{tabular}
```

que genera:

Team	P	W	D	L	F	A	Pts
Manchester United	6	4	0	2	10	5	12
Celtic	6	3	0	3	8	9	9
Benfica	6	2	1	3	7	8	7
FC Copenhagen	6	2	1	3	5	8	7

3.4.3.4. Matemática

LaTeX ganó popularidad por sus composición tipográfica en formulas y expresiones matemáticas. Todas las expresiones y símbolos matemáticos deben ser comenzar y terminar con el caracter $.

LaTeX provee de casi la totalidad de símbolos posibles en la comunidad científica, todos deben ser escritos con el prefijo \, por ejemplo, el comando `$\forall$` genera el símbolo $\forall$.

Ejemplos:

Código

```
$$\sum_{i=1}^\infty
   \frac{1}{2^i} = 1$$
```

Salida

$$\sum_{i=1}^{\infty} \frac{1}{2^i} = 1$$

```
$\phi(t)=\frac{1}{\sqrt{2\pi}}
\int^t_0 e^{-x^2/2} dx$
```

$$\phi(t) = \tfrac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

```
$f(x) = \begin{dcases}
\int x\, \mathrm{d} x &
\text{con } x \geq 0 \\
b^2  & \text{con } x < 0
\end{dcases}$
```

$$f(x) = \begin{cases} \displaystyle\int x \, \mathrm{d}x & \text{con } x \geq 0 \\ b^2 & \text{con } x < 0 \end{cases}$$

```
$z = \overbrace{
\underbrace{x}_\text{real} +
\underbrace{iy}_\text{imaginary}
}^\text{complex number}$
```

$$z = \overbrace{\underbrace{x}_{\text{real}} + \underbrace{iy}_{\text{imaginary}}}^{\text{complex number}}$$

```
$A_{m,n} =
\begin{pmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,n}\\
a_{2,1} & a_{2,2} & \cdots & a_{2,n}\\
\vdots  & \vdots  & \ddots & \vdots \\
a_{m,1} & a_{m,2} & \cdots & a_{m,n}
\end{pmatrix}$
```

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Capítulo 4

Desarrollo de Aplicaciones Web

El desarrollo web un término amplio que define la creación de sitios web para Internet. Para conseguirlo se hace uso de tecnologías de software del lado del servidor y del cliente que involucran una combinación de procesos de base de datos con el uso de un navegador web a fin de realizar determinadas tareas o mostrar información.

4.1. Arquitectura cliente – servidor

Según Sommerville:

> El modelo arquitectónico cliente-servidor es un modelo de sistema que se organiza como un conjunto de servicios y servidores asociados, y clientes que acceden y usan dichos servicios. [14]

Como se ejemplifica en la figura 4.1, la arquitectura cliente-servidor se basa en un principio básico: el cliente realiza peticiones de servicios a otro programa, el servidor, y este le da respuesta.

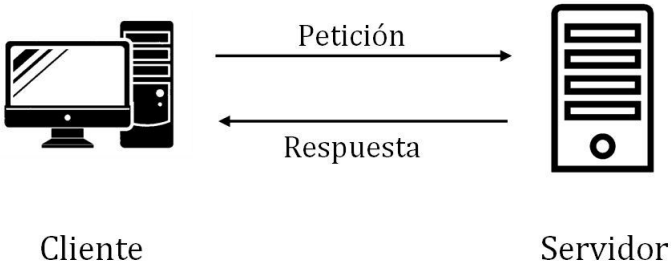


Figura 4.1: Representación gráfica de la arquitectura Cliente-Servidor.

4.2. Aplicación Web

Una aplicación web es una pieza de software desarrollada con una arquitectura cliente - servidor que se encarga de proveer herramientas a un usuario. En una aplicación web, el cliente corresponde al navegador de una computadora, que recibe y envía información del/al servidor, procesa, codifica y ejecuta funciones. El servidor, se accede a través de una red (local o remota) y se encarga de procesar información según la lógica del negocio.

4.3. Patrón Modelo – Vista – Controlador

El patrón *Modelo-Vista-Controlador (MVC)*, fue introducido en el año 1979 por *Trygve Reenskaug*, y años después en el año 1988, MCV se expresó como un concepto general en un artículo escrito por *Glenn E. Krasner and Stephen T. Pope* [15], conceptualizado como una separación triple de los componentes de un sistema, que representa un paradigma para la estructuración e implementación de aplicaciones interactivas. Los *Modelos* son los componentes del sistema encargados de los datos y la lógica de negocio, las *Vistas* se encargan de presentar los aspectos del modelo y los *Controladores* son usados para enviar mensajes al *modelo* y proveer una relación entre el *modelo* y las *vistas*.

4.4. Framework

Generalmente, para realizar este tipo de desarrollo se utiliza un *framework* que, en términos generales, ofrece una estructura para la implementación del sitio o aplicación web. Actualmente existen muchos *frameworks* que usualmente están atados a un lenguaje de programación como *Python*, *PHP*, *Java*, entre otros. Una de las ventajas del uso de *frameworks* es que estos están basados en el paradigma *MVC (Modelo-Vista-Controlador)* logrando una distribución modular de los aspectos del desarrollo.

4.5. Autorización abierta (*OAuth*)

OAuth del inglés *Open Authorization* es un protocolo que permite flujos simples de autorización para aplicaciones web, de escritorio, móviles, entre otras.

Fue propuesto por Blaine Cook y Chris Messina, y permite la autorización segura de una *Interfaz de Progrmación de Aplicaciones (API, del inglés Application Programming Interface)* [26] de modo estándar.

Es usado para que un usuario de internet pueda iniciar sesión en otros sitios web, o clientes, usando sus cuentas de Google, Facebook, Microsoft, Twitter, entre muchas otras. Esto permite que proveedores de servicios, como los mencionados anteriormente, permita a un cliente la autorización para accesar a la información del usuario.

4.5.1. Características

OAuth permite:

- Diferentes niveles de acceso: sólo lectura y lectura – escritura.
- Granularidad de acceso: el usuario decide a qué información le está dando acceso. Por ejemplo, información básica como: correo electrónico, nombre de usuario,

fecha de nacimiento hasta información sensible como la lista de amigos o calendario.

- Anulación de acceso: los proveedores de servicio ofrecen la opción de anular el acceso al cliente, en caso de que el usuario así lo desee.

4.5.2. Proceso

Antes de usar *OAuth* la aplicación cliente debe estar registrada en el proveedor de servicio y esta generará credenciales únicas y secretas para la aplicación que se usarán para identificar y validar a la misma. Contiene un *ID del Cliente* (*Client ID*) que representa la identidad del cliente y un *Secreto de Cliente* (*Client Secret*) usado para autenticar la aplicación, este se debe mantener privado. Estas credenciales son llamadas token de acceso o *access token*.

El proceso consta de varios pasos:

1. La aplicación cliente solicita la autorización al proveedor de servicio seleccionado para acceder a los recursos del usuario.
2. Un flujo de iniciar sesión – solicitud de permiso al usuario es suministrado por el proveedor de servicio.
3. La aplicación cliente solicita un token de acceso de usuario al API del proveedor de servicio presentando los token de acceso propios de la aplicación cliente, incluyendo el *ID del Cliente* y *Secreto de Cliente*.
4. Si la aplicación es autenticada correctamente, el API del proveedor de servicio envía un token de acceso de usuario a la aplicación cliente.
5. La aplicación cliente solicita el o los recursos a el API del proveedor de servicio y presenta el token de acceso de usuario para autenticación.

6. Si el token de acceso es válido, el API del proveedor de servicio provee el o los recursos a la aplicación cliente.

Capítulo 5

Tecnologías y Herramientas

5.1. Django

Django es un framework para desarrollo web basado en *Python* [16] que fue desarrollado en originalmente para gestionar varias páginas orientadas a noticias de la *World Company de Lawrence, Kansas*, y fue liberada al público bajo una licencia BSD en julio de 2005, desde entonces este framework ha estado en crecimiento. Actualmente, la Fundación de Software Django, lo define como:

> Un framework para aplicaciones Web que estimula el desarrollo rápido y un diseño limpio y pragmático. Elaborado por desarrolladores experimentados, se encarga de gran parte de las dificultades del desarrollo Web, mientras el programador puede enfocarse en escribir su aplicación sin reinventar la rueda. Es gratuito y de código abierto.

Django se describe a sí mismo como: "*El framework web para perfeccionistas con fechas topes*", ya que provee una serie de funcionalidades robustas que acortan el tiempo de desarrollo, como: mapeo objeto relacional (ORM), vistas genéricas, sistema de plantillas, un despachador de URLs basado en expresiones regulares, cacheo, compresión de la salida, normalización de URLs, protección CSRF, soporte de sesiones, soporte de internacionalización, entre otras.

Uno de los principios que definen a Django, es el "No Te Repitas"(DRY, del inglés Don't Repeat Yourself), donde las funcionalidades o aplicaciones desarrolladas con Django son altamente modulares y fáciles de integrar en cualquier otro sistema.

5.1.1. Patrón Modelo – Plantilla – Vista

Django tiene una interpretación diferente para el patrón *MVC*. En su esta, las *Vistas* describen qué datos serán presentados, y no necesariamente cómo los datos lucen.

Entonces, en su caso, una *Vista* representa la función que es llamada por una URL en específico, esta función a través del *Modelo* selecciona qué datos serán presentados. Luego, la *Vista* despacha los datos a una *Plantilla (template)* que describirá cómo los datos son presentados.

Por estas razones, Django se define a sí mismo como un *framework Modelo-Plantilla-Vista (MTV del inglés, Model – Template – View)*.

5.1.2. Mapeo Objeto – Relacional

El *Mapeo Objeto – Relacional o (ORM)* consiste en una abstracción de alto nivel, es una técnica que permite crear una capa de comunicación en la que el acceso a una base de datos está basada en un ambiente orientado a objetos.

Esto permite al programador escribir código Python (en el caso de Django) y no *SQL* (del inglés, Structured Query Language, Lenguaje de Consultas Estructuradas), permitiendo una abstracción referente al manejador de base de datos a usar.

Una consulta en lenguaje *SQL* [17] como:

```
SELECT * FROM USUARIOS WHERE codigo_zip=94107;
```

es equivalente a esta porción en código Python:

```
usuarios = USUARIOS.objects.filter(codigo_zip=94107)
```

Donde una clase es equivalente a una tabla (`usuarios`) y un atributo es equivalente a una columna de dicha tabla (`codigo_zip`).

Entre las características y funcionalidades más relevantes, tenemos:

- Todos los modelos son representados en el archivo `models.py`.
- La cantidad de tipos de datos disponibles para los atributos es extensa, desde los tipos de datos básicos hasta tipos de datos dedicados a archivos.
- Es posible hacer todas las operaciones *CRUD (crear, leer, actualizar y borrar).*
- Soporta las relaciones uno-a-uno con el método `OneToOneField()`, uno-a-muchos con el método `ForeignKey()` y las relaciones muchos-a-muchos con el método `ManyToManyField()`.
- Provee la posibilidad de escribir métodos propios por cada clase, permitiendo representar la mayoría de la lógica de negocios en el modelo.
- Provee un módulo migratorio de *SQL* al esquema orientado a objetos. Es decir, que si la definición de la estructura de la base de datos sólo la tenemos en lenguaje *SQL* este módulo migra ese código a clases escritas en *Python* y al revés.

5.1.3. URLs y vistas

Django provee un despachador de URLs basado en expresiones regulares, estas se crean y editan en el archivo `url.py`. Este archivo está basado en funciones del tipo:

```
url(<url>, <vista>)
```

Donde una `url` representa un patrón escrito en forma de expresión regular y la `vista` representa el nombre del método que se encargará de manejar la solicitud.

Los métodos que manejarán las solicitudes se encuentran en el archivo `views.py` y constituyen las *vistas*. Dentro de cada método se harán las comunicaciones perminentes con el *modelo* y luego se enviará una respuesta al cliente. La respuesta pueden ser códigos HTTP [18], un objeto JSON [19] o una página HTML [20].

5.1.4. Plantillas

Una plantilla o *template* es una cadena de texto pensada para separar la presentación de un documento de sus datos. Una plantilla define variables y varios trozos de lógica básica (etiquetas o *tags*) que regulan la manera en que debería mostrarse el documento. Esto permite la construcción de sitios con datos enteramente dinámicos de una manera sencilla.

Las plantillas están basadas en el lenguaje de plantillas de Django. Al momento de usar el sistema de plantillas de Django, estas necesitan un parámetro de entrada llamado *context* que representa toda la información necesaria, recopilada por una *vista*, que debe ser dispuesta en un documento *HTML* para su posterior respuesta al cliente.

Ya que esta información es dinámica y contiene variables, son necesarias etiquetas que separan código HTML del lenguaje de plantillas de Django, como:

- `{{ ... }}`: cualquier cosa representada entre dos llaves representa el contenido de una variable dentro de *context*.
- `{{ a | b }}`: una etiqueta de esta forma representa un filtro. Donde `a` representa una variable y `b` representa un filtro que se le aplica a `a`.
- `{% ... %}`: cualquier cosa representada entre llaves y simbolos de porcentaje representa una etiqueta de bloque, lo que indica una instrucción, estas pueden ser condicionales o de ciclicas.

5.2. Sistema de Control de Versiones

Un *Sistema de Control de Versiones* o *SCV* es un paquete de software que, al ser iniciado, monitorea y guarda etiquetas referentes a todos los cambios en todos los archivos permitiendo revisitar cualquiera de estas etapas en cualquier momento. Los *SCV* [21] son útiles para archivos que son modificados frecuentemente, por ejemplo programas informáticos, documentación, gráficos de procedimientos, monografías y cartas.

Un *SCV* tiene dos grandes beneficios:

- Versionamiento: la capacidad de guardar todos los cambios de cualquier carpeta sin necesidad de tener múltiples copias de los mismos archivos. Es decir, si existe una carpeta con un proyecto, los archivos guardados dentro de esta carpeta sólo corresponden a la última versión del mismo, y todas las versiones y variantes de cada uno de los archivos están empaquetados dentro del *SCV* así, en caso de necesitarlo, es posible solicitar cualquier versión.
- Colaboración: permite que en un equipo de trabajo, cada miembro pueda trabajar en cualquier archivo, en cualquier momento de una manera eficaz. Cada miembro puede crear y modificar archivos compartidos para luego unirlos (*merge*) en una sola versión común para todos.

5.2.1. Tipos

Existen tres tipos de *Sistema de Control de Versiones* disponibles. Éstos están clasificados basados en modo de operación.

5.2.1.1. Sistema de Control de Versión Local

Basado en el común esquema de copiar archivos con nombres diferentes para llevar un control de versiones manual, este tipo de *SCV* fue el primero en ser desarrollado. Este tipo de *SCV* mantiene los cambios guardados en una base de datos simple para un sólo archivo. El *SCV* Local más popular es el Sistema de Control de Revisión (o *RCS del inglés Revision Control System*) que incluso aún se distribuye en computadoras actuales.

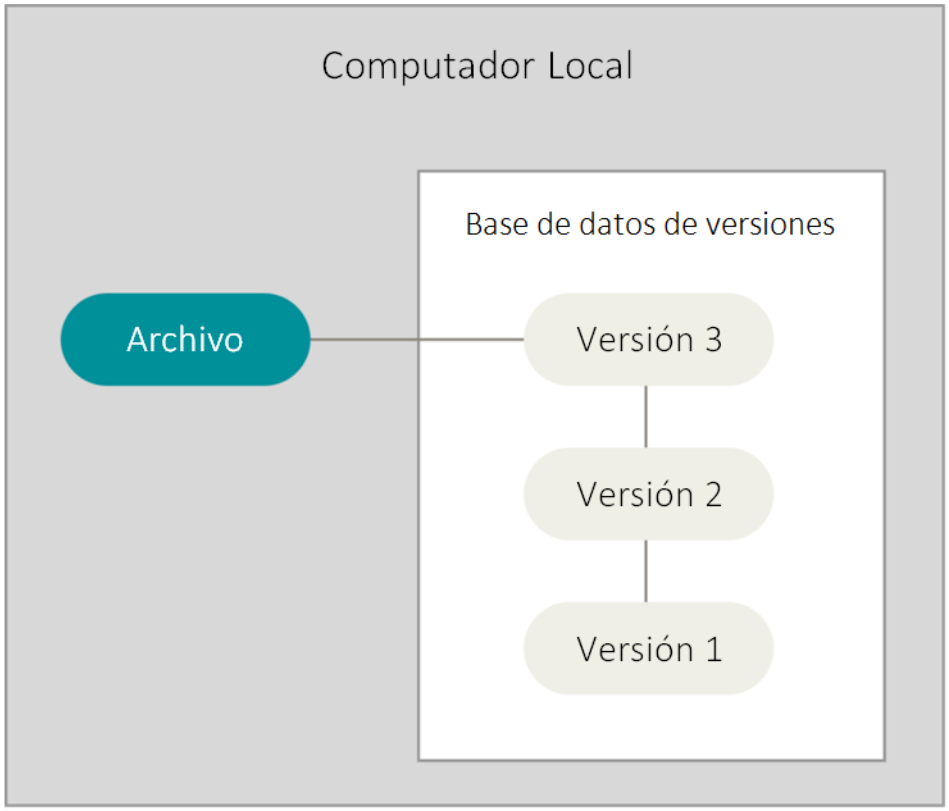


Figura 5.1: Representación gráfica del Sistema de Control de Versión Local.

5.2.1.2. Sistema de Control de Versión Centralizada

Luego de ser desarrollado el *SCV* Local se encontró que este esquema no era apropiado para ambientes colaborativos, por lo que se creó un *SCV* Centralizado. Este

tipo de sistema tiene un servidor central que contiene todos los archivos versionados y un número de clientes que recuperan estos archivos. El problema de este sistema es que al momento de haber alguna falla de software o hardware en el servidor principal todos los clientes pierden el acceso al mismo o incluso se puede correr el riesgo que los datos estén corruptos y perder toda la información, entre otros. Actualmente sistemas como CVS, Subversion y Perforce están desarrollados con esta estructura.

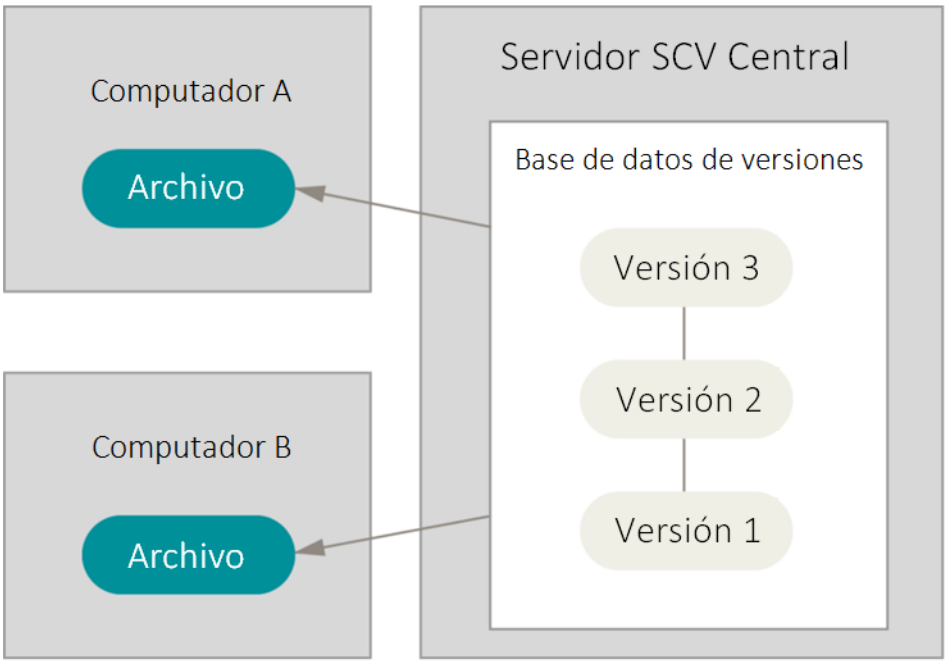


Figura 5.2: Representación gráfica del Sistema de Control de Versión Centralizado.

5.2.1.3. Sistema de Control de Versión Distribuido

En este tipo de *SCV* cada cliente contiene una copia de los archivos del servidor principal (repositorio), así, si se presenta una falla en el mismo, cualquiera de los clientes puede copiar los archivos al repositorio. Sistemas como Git, Mercurial, Bazaar o Darcs son sistemas distribuidos.

Los *SCV* distribuidos tienen las ventajas de los *SCV* locales, tales como: la capacidad de hacer cambios locales sin ninguna preocupación de la conectividad con el servidor y no depender de una sola copia de los archivos almacenados en el servidor. Estos se combinan con las ventajas de los *SCV* centralizados, como: la reutilización de trabajo y el trabajo colaborativo al no depender de la historia almacenada en máquinas individuales.

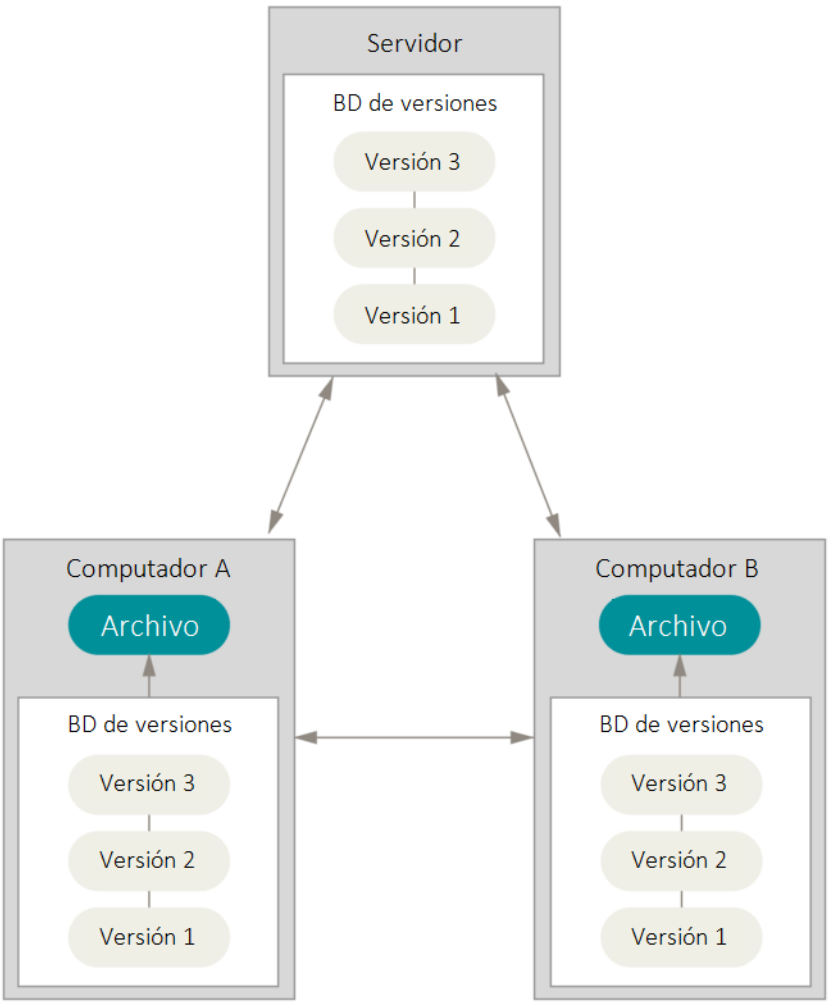


Figura 5.3: Representación gráfica del Sistema de Control de Versión Distribuido.

5.2.2. Terminología

La terminología empleada puede variar de sistema a sistema, pero a continuación se describen algunos términos de uso común:

- Repositorio (`repository`): lugar en el que se guarda la última versión de todos los archivos junto con el histórico de cambios.
- Clonar (`clone`): creación de un repositorio local basado en una copia de un repositorio ya existente.
- Revisión, versión (`revision, version`): cualquier cambio. Es una versión determinada de la información que se gestiona. Identificada en algunos sistemas con contadores y otros con código de detección de manipulaciones.
- Publicar (`commit`): escritura o integración de la copia local al repositorio.
- Integración (`merge`): operación en la que dos conjuntos de cambios son aplicados a un archivo o set de archivos.
- Conflicto (`conflict`): ocurre cuando dos partes hacen cambios al mismo documento y el sistema no puede reconciliar los cambios automáticamente.
- Descarga (`pull`): copia de versiones del repositorio principal a la copia local.
- Subida (`push`): copia de versiones desde la copia local hacia el repositorio principal.
- Maestro (`master`): rama maestra o principal.

5.2.3. Git

Git es un *Sistema de Control de Versiones Distribuido* gratis y de código abierto diseñado para manejar desde pequeños a grandes proyectos con velocidad y eficiencia.

Creado por Linus Torvalds [22] en el año 2005 y es actualmente uno de los *SCV* más populares.

Entre las características que distinguen a *Git* de otros *SCV*, se tiene: [23]

- Ramificaciones: *Git* permite e incentiva el uso de `branches` locales que pueden ser completamente independientes entre sí. Al momento de hacer `push` se pueden seleccionar todas, algunas o ningunas de los `branches`, lo que permite, por ejemplo: crear un `branch` para probar alguna idea o funcionalidad: si es exitoso, se puede hacer `push` y si no, se abandona o se elimina el `branch`.
- Rapidez: al ser un sistema distribuido, casi la totalidad de las operaciones son realizadas localmente, lo que le da una gran ventaja contra sistemas centralizados que constantemente necesitan comunicación con el servidor principal.
- Seguridad de los datos: cada archivo y `commit` es verificado a través de sumas de verificación (*checksum*). *Git* asegura que ningún archivo, fecha, mensaje de `commit` o cualquier otro dato puede ser cambiado sin cambiar el número de identificación de la operación.
- Area intermedia: area entre el repositorio local y el repositorio central en la que los `commits` pueden ser revisados antes de completarlos. Permite hacer `commit` de uno, algunos o todos los archivos; a diferencia de otros sistemas que sólo permiten hacer `commit` sobre todos los archivos en el repositorio local.

5.2.4. Subversion (*SVN*)

SVN es un *SCV* centralizado gratis y de código abierto que promete ser una mejora mayor de su predecesor *CVS, o Concurrent Versioning System*. *SVN* fue lanzado en el año 2000 por CollabNet y actualmente se encuentra bajo la licencia Apache/BSD. [24]

SVN ofrece las características básicas que puede ofrecer cualquier *SCV* centralizado, pero entre las características que lo diferencian, se tiene:

- Sistema centralizado: al ser centralizado, *SVN* necesita constante comunicación con el repositorio para realizar cualquier operación. Lo que permite al dueño del repositorio total control sobre el mismo.
- Archivos binarios: soporte nativo y robusto sobre archivos binarios.
- Versionamiento: provee versionamiento sobre directorios, metadata y renombres de archivos.
- Bloqueos: archivos pueden ser bloqueados ante `commits`.

5.3. Foro en internet

Es un sitio de discusión online asíncrono donde las personas publican mensajes alrededor de un tópico, creando un hilo de conversación jerárquico. [25]

Estos son muy usados actualmente, permitiendo a personas de cualquier parte del mundo discutir y compartir conocimientos.

Un foro está compuesto por:

- Mensaje (*Post*): un mensaje enviado por un usuario que contiene fecha y hora de publicación y detalles del usuario.
- Temas, hilos (*Thread*): es una colección de mensajes, normalmente ordenados de recientes a antiguos. Un *thread* está definido por un título, una descripción y un mensaje original que abre la discusión.

Normalmente se tienen dos tipos de usuarios: administradores, que tienen permisos especiales sobre el foro; moderadores que contribuyen a depurar el foro de *posts*

dañinos o sensibles, y usuarios comunes que sólo pueden tener permiso para publicar *posts* y/o crear nuevos *threads*.

Algunos foros online proveen características como:

- Moderación: un administrador puede decidir si el *post* puede ser publicado o no.
- Control de spam: bloquea o elimina mensajes que son considerados spam automáticamente.
- Censura de palabras: palabras sensibles o inapropiadas pueden ser automáticamente cesuradas reemplazando algunos de sus caracteres con '*'.
- Soporte de usuarios anónimos.
- Soporte de archivos adjuntos: ya sean imágenes o archivos para descarga, pueden ser adjuntados a un mensaje.

5.3.0.1. jquery-comments

jquery-comments es una librería de jQuery a código abierto desarrollada por la empresa Viima que provee las funcionalidades y facilidades para desplegar la interfaz de un ambiente de comentarios y respuestas.

La cual permite comentar, responder a un comentario (comentarios anidados), editar un comentario, borrar un comentario, opción de "me gusta" o "no me gusta" un comentario, adjuntar archivos, ordenar según varios criterios, entre otras cosas. Cabe destacar que jquery-comments sólo provee interfaz, la lógica de los métodos para cada funcionalidad deben ser escritas por el desarrollador.

5.4. ContentTools

ContentTools es una librería a código abierto desarrollada en JavaScript que provee la funcionalidad de convertir una página HTML en un editor de texto del tipo

WYSIWYG (Lo que ves es lo que obtienes, del inglés What You See Is What You Get).

ContentTools fue desarrollada por Anthony Blackshaw en la compañía Getme Limited. Creada inicialmente para su propio uso en el año 2009 y lanzada como código abierto en el año 2015. Con una gran recepción ha sido incluida en numerosas listas de recomendación e interés sobre librerías JavaScript y CSS.

Este editor permite que texto, tablas, imágenes, videos y otros contenidos puedan ser creados, editados, eliminados o movidos vía Drag and Drop directamente en la página. ContentTools provee una caja de herramientas sensitiva a contenido que le permite al usuario usar todas las funcionalidades de edición, similar a un procesador de textos.

5.5. MathJax

MathJax es una librería de JavaScript con licencia libre (licencia Apache) lanzada inicialmente en el año 2009 que permite visualizar fórmulas escritas en el lenguaje LaTeXen navegadores web. MathJax es patrocinada por Asociación Americana de Matemática y constituye la librería más popular y confiable para este tipo de visualización de formulas matemáticas.

MathJax no requiere uso ni instalación de un compilador TeX, únicamente hace usos de funciones JavaScript, HTML y CSS. MathJax busca en el contenido de la página cualquier trozo de texto, escrito bajo la notación matemática de LaTeXque comience y termine con cierta cadena de caracteres (generalmente, el caracter "$") y lo transforma en contenido HTML optimizado que permite una visualización muy similar al que daría el sistema de composición tipográfica TeX.

Capítulo 6

Diseño

6.1. Levantamiento de Requerimientos

La intención de esta sección es explorar los requerimientos que son necesarios para la aplicación, separadas en tres grandes grupos según su funcionalidad.

6.1.1. Sesiones y cuentas

Con el fin de asegurar la información que pueda estar disponible para cada usuario, es necesario que la aplicación maneje cuentas y sesiones. En las que un usuario pueda:

- Crear una cuenta: la aplicación debe permitir crear una cuenta, proveyendo información básica y contraseña.
- Iniciar sesión: luego de que el usuario cree su cuenta, puede iniciar sesión haciendo uso de las credenciales provistas anteriormente.
- Recuperar contraseña: olvidar o perder el control de una cuenta es algo común entre los usuarios, por lo que es necesario que el sistema tenga la capacidad de hacer un proceso seguro de recuperar cuenta al olvidar la contraseña.
- Iniciar sesión con sistemas terciarios (*OAuth*): con el fin de agilizar y hacer el proceso de *registro - iniciar sesión* más sencillo, se requiere usar el protocolo *OAuth* (discutido anteriormente).

6.1.2. Competencias

Por cada competencia, se requiere:

- Crear, editar y eliminar una competencia.
- Otorgar permisos de administrador o contribuyente: el creador de la competencia podrá conceder permisos a otros usuarios.
- Descargar competencia: por cada competencia, requiere descargar un archivo comprimido con toda la información y documentos necesarios.
- Votación: un usuario podrá llamar a votación para que los usuarios realicen su voto sobre problemas que prefieren vayan a competencia y su dificultad.
- Crear problema: por cada competencia un usuario puede crear cualquier cantidad de problemas según lo desee o requiera.
- Importar problema: se podrá copiar un problema de otra competencia a la competencia actual.
- Cambiar estilo: por cada competencia, se podrá cambiar el estilo y plantilla de problema para ofrecer un PDF con aspecto diferente.

6.1.3. Problemas

Por cada problema se requiere:

- Crear, editar y eliminar un problema.
- Añadir etiquetas (tags): según la(s) técnica(s) requeridas para su solución, con el fin de clasificar problemas.

- Subir casos de prueba: por cada problema se podrá subir múltiples casos de prueba individualmente.
- Subir solución: por cada problema se podrá subir múltiples soluciones individualmente.
- Generar PDF: generar y visualizar el archivo .PDF generado en base a el contenido introducido en la página.
- Descargar problema: por cada problema, requiere descargar un archivo comprimido con toda la información y documentos necesarios.
- Comentarios: por cada problema, se requiere un ambiente de comentarios donde los usuarios puedan discutir sobre el problema.

6.2. Arquitectura

Basado en un ambiente de petición y respuesta, la arquitectura del sistema es cliente-servidor. Donde la aplicación y base de datos residen en un mismo servidor y los clientes se conectan a ella a través de internet. La aplicación fue desarrollada bajo el framework web Django (versión 1.8) usando el lenguaje Python (versión 2.7).

Para el diseño y construcción de las vistas se utilizó HTML5, CSS3, JavaScript y jQuery como lenguajes que se interpretan del lado del cliente, y la tecnología AJAX para solicitudes asíncronas aumentando la interactividad.

Ilustrado en la figura 6.1, para la compilación de generación de archivos .PDF a partir de archivos .TEX se utilizó MikTeX**, para el Sistema de Control de Versiones se usó GIT** y como manejador de base de datos MySQL**.

Por cada solicitud hecha por el cliente al servidor, este la recibe, procesa y decide si hacer uso de recursos como el sistema LaTeX, control de versiones y/o base de datos.

Figura 6.1: Diagrama de arquitectura de sistema

6.3. Modelo de Datos

El siguiente modelo relacional provee la estructura para manejar la información necesaria para cumplir con todos los requerimientos mencionados anteriormente. Cabe destacar que la base de datos no está ligada a ningún Manejador de Bases de Datos gracias al ORM que el framework Django provee.

Se infiere:

- Un usuario tiene muchos intentos de recuperar contraseña.
- Una competencia puede estar asociada con muchos usuarios y un usuario puede estar relacionado con muchas competencias.
- Una competencia tiene muchos problemas.
- Un problema tiene muchas etiquetas y una etiqueta representa a muchos problemas.
- Un problema tiene muchos casos de pruebas y soluciones.

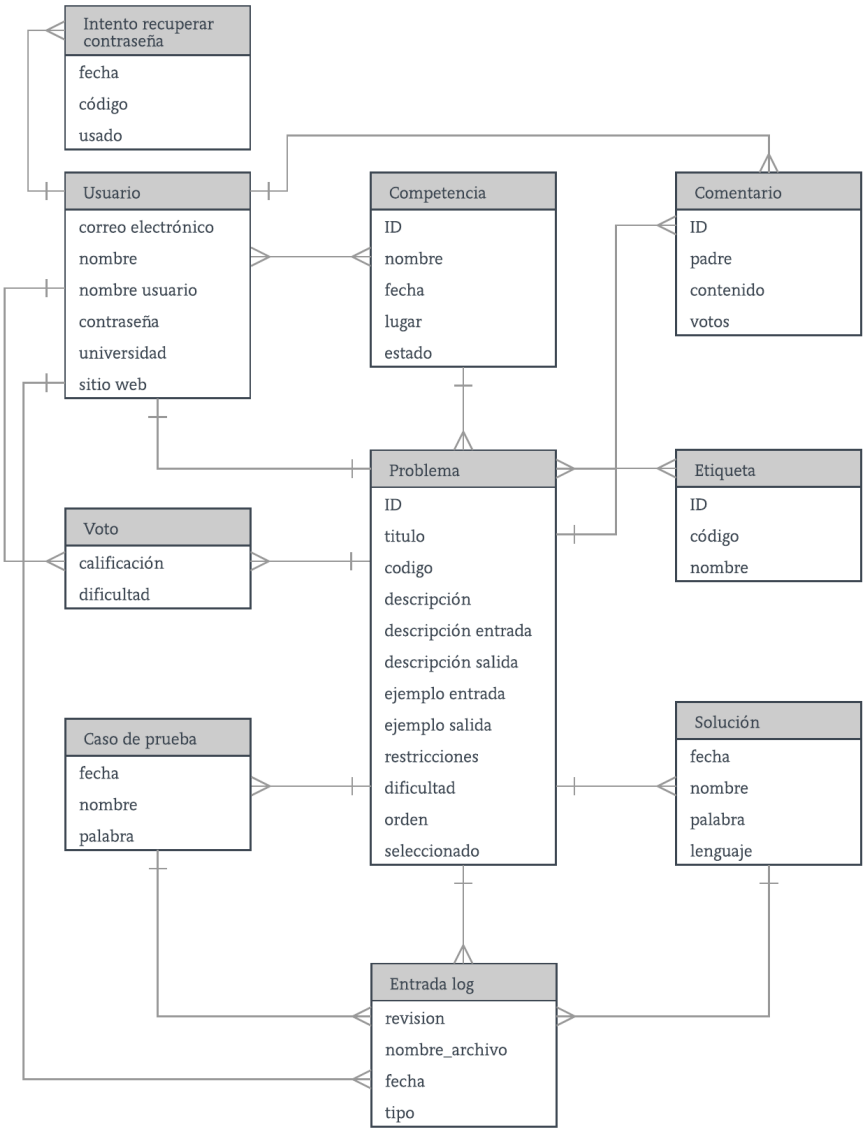


Figura 6.2: Diagrama de base de datos de la aplicación.

- Un usuario puede votar por muchos problemas y un problema puede ser votado por muchos usuarios.
- Un usuario puede hacer muchos comentarios a un problema y un problema puede tener muchos comentarios de un usuario.
- Una entrada de log tiene un usuario y un problema, caso de prueba o solución.

6.4. Interfaz

Con el fin de entregar una interfaz limpia, usable, moderna y atractiva, la aplicación se inspira en el Material Design [29] propuesto por Google en el año 2014. Intentando usar tipografía clara, casillas bien ordenadas y separadas, colores e imágenes llamativos, dejando de lado las animaciones.

La interfaz en general, tiene tres componentes:

- Una cabecera (header) con el logo de la aplicación, botones para iniciar sesión o registrarse o, en caso de que el usuario haya iniciado sesión, información sobre su perfil y la opción de terminar la sesión.
- El contenido de la página que se esté solicitando.
- Un pie de página con información sobre la aplicación.

La interacción a través de AJAX (JavaScript asíncrono y XML) es parte fundamental del lado del cliente, haciendo que el usuario tenga una experiencia eficaz evitando que la página deba recargarse en cada solicitud. Por cada solicitud AJAX se asegura el despliegue de una barra de progreso como también mensajes satisfactorios, de alerta y errores.

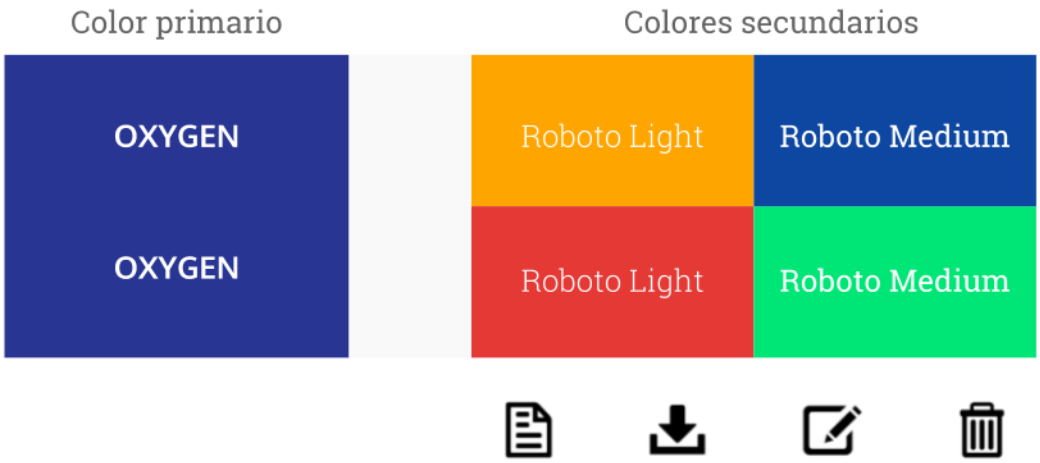


Figura 6.3: Colores y tipografía

6.4.1. Colores y tipografía

Como color principal se eligió un azul oscuro que provee seriedad a la aplicación. Los colores secundarios, inspirados en la gama de colores de Material Design [27] proveen más dinamismo y contraste a la aplicación.

Como fuente se utilizó una combinación de Roboto para el contenido y Oxygen para los títulos y otros elementos que necesitaban énfasis. Ambas fuentes proveídas por Google a través de su plataforma GoogleFonts [28].

Los íconos fueron proveídos por Ligature Kudakurage Symbols [30] y Font Awesome [31] proveyendo metáforas fáciles de reconocer para cualquier usuario. Las fuentes y gama de colores son mostrados en la figura 6.3

6.4.2. Logo

El logo es un diseño sencillo directamente con el nombre que se le dio a la aplicación "*ContestMaker*". Está presentado en dos versiones: larga como se ve en la figura 6.4 y corta como se ve en la figura 6.5, según la necesidad y espacio de donde se quiera

Figura 6.4: Logo largo

Figura 6.5: Logo corto

visualizar. Normalmente con un fondo azul, pero puede tener un fondo de cualquier color.

Capítulo 7

Implementación

7.1. Inicio

En la página de inicio, mostrada en la figura 7.1 se encuentra una barra de navegación con el logo, y las opciones de iniciar sesión o registrarse; seguido de un banner con únicamente fines estéticos y una breve explicación de las capacidades de la aplicación.

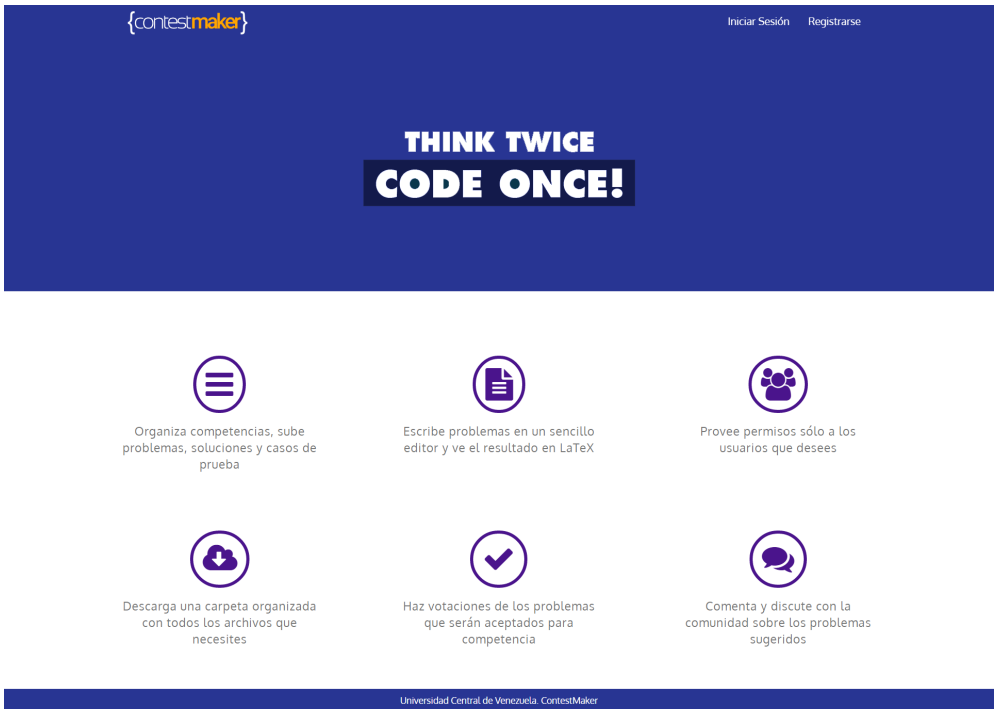


Figura 7.1: Página principal de la aplicación

7.2. Sesiones y cuentas

El entorno de sesiones de Django permite almacenar y recuperar cualquier dato basándose en la sesión del usuario. La única cookie que usa el framework de sesiones es un identificador de sesión; todos los datos de la sesiones se almacenan en la base de datos. Así, almacenando la información relevante solo en el servidor y se abstrae del problema de envío y recepción de cookies constante.

En la aplicación, por cada sesión, se guarda únicamente un identificador del usuario y por cada solicitud que necesite autenticación, es recuperada la información del usuario desde la base de datos con el identificador de sesión, asegurando siempre la validez de la autenticación del usuario evitando recargar de datos el entorno de sesiones.

7.2.1. Registro

Provee las funcionalidades necesarias para el registro de un usuario nuevo, permitiendo así el uso de la aplicación. El registro puede ser de dos formas:

- A través de la aplicación: como se ve en la figura de ser con esta opción el sistema le solicitará al usuario su nombre, un correo electrónico, una contraseña que deberá ser escrita dos veces (para evitar los errores) y opcionalmente, una imagen de perfil que funcionará como avatar para la identificación gráfica de usuarios. El campo de contraseña tiene integrado un medidor de seguridad, que se encarga de analizar la contraseña con el fin de informar si esta es débil o segura; este medidor es sólo informativo.
- A través de aplicaciones terciarias: con el uso del protocolo OAuth 2.0 es posible hacer el registro usando la información ya almacenada en los proveedores de servicio Facebook y Google, haciendo el proceso más rápido.

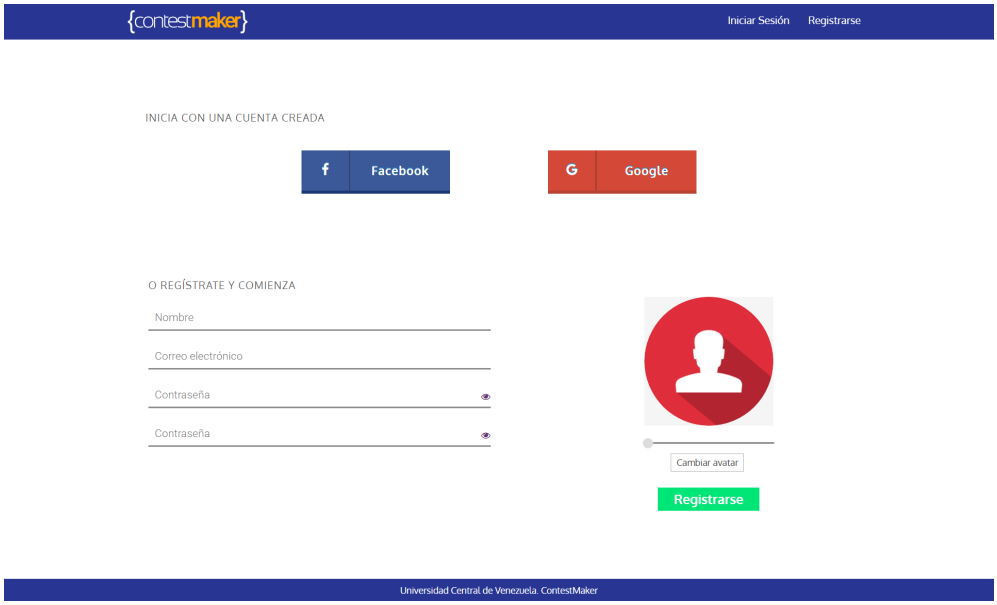


Figura 7.2: Página de registro

En la figura 7.2 se visualiza las dos formas posibles de registro. Si el registro fue a través de la aplicación, se le será enviado un correo electrónico para verificar la cuenta. Si fue a través del protocolo OAuth, la sesión del usuario es iniciada automáticamente.

Al crear una cuenta, la aplicación genera un nombre de usuario en base al nombre suministrado.

7.2.2. Inicio de sesión

Para el inicio de sesión se hizo uso de un modal, como se ve en la figura 7.3, que provee de los dos tipos de inicio de sesión, este solicita correo electrónico y contraseña, si la cuenta fue creada a través de la aplicación, y los botones de iniciar sesión con Facebook o Google.

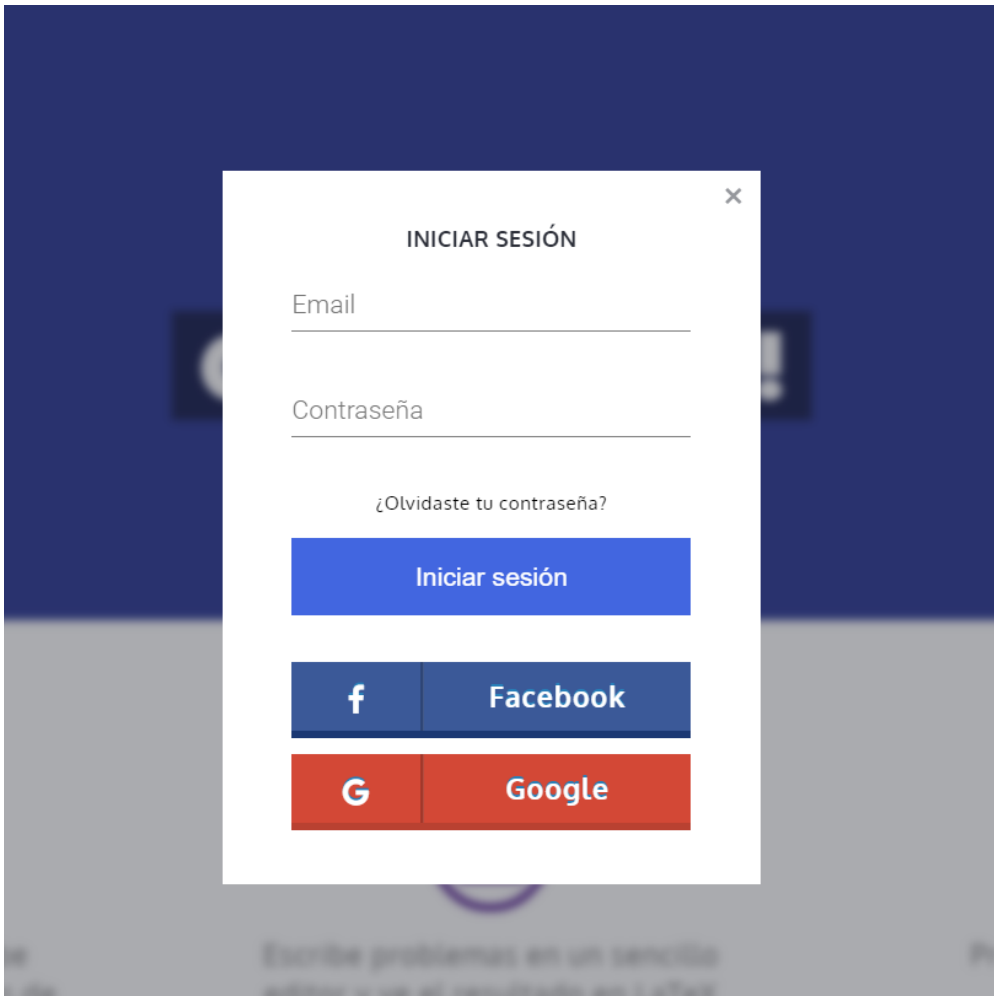


Figura 7.3: Inicio de sesión

7.2.3. Recuperación de cuenta

Toda la información que es servida por la aplicación es relativa a el usuario que la está solicitando. Con el fin de asegurar que un usuario no pierda el control de su cuenta y de que sea un proceso seguro, se implementa el siguiente proceso:

- Cuando el usuario solicita la recuperación, se le pide una dirección de correo electrónico.
- El sistema comprueba que dicha dirección esté asociada a un usuario. En caso de que exista, se genera un código UUID (Identificador Único Universal, del inglés Universally Unique Identifier) al que luego es aplicado una función hash. Este código es enviado en forma de link al correo electrónico provisto por el usuario.
- En la tabla Intento de Recuperar Contraseña es guardada la dirección de correo electrónico, código, fecha y estado del intento.
- En el momento en que el usuario le da click al link, es redirigido a una función del servidor que se encarga de comprobar la validez de la solicitud y además, el tiempo de vida del código, ya que por medidas de seguridad este tiempo de vida es limitado.
- Al ser comprobado, el sistema permite el reestablecimiento de la contraseña.

Nota: si la cuenta de la que el usuario perdió control está enlazada con Google o Facebook, el usuario deberá recuperar su contraseña directamente con esos proveedores de servicio y luego ingresar normalmente en la aplicación.

7.2.4. Perfil

Para visualizar el perfil de un usuario sólo basta con seguir el link de cualquier avatar. En esta página, como se ve en la figura 7.4 se puede ver la información básica del usuario.

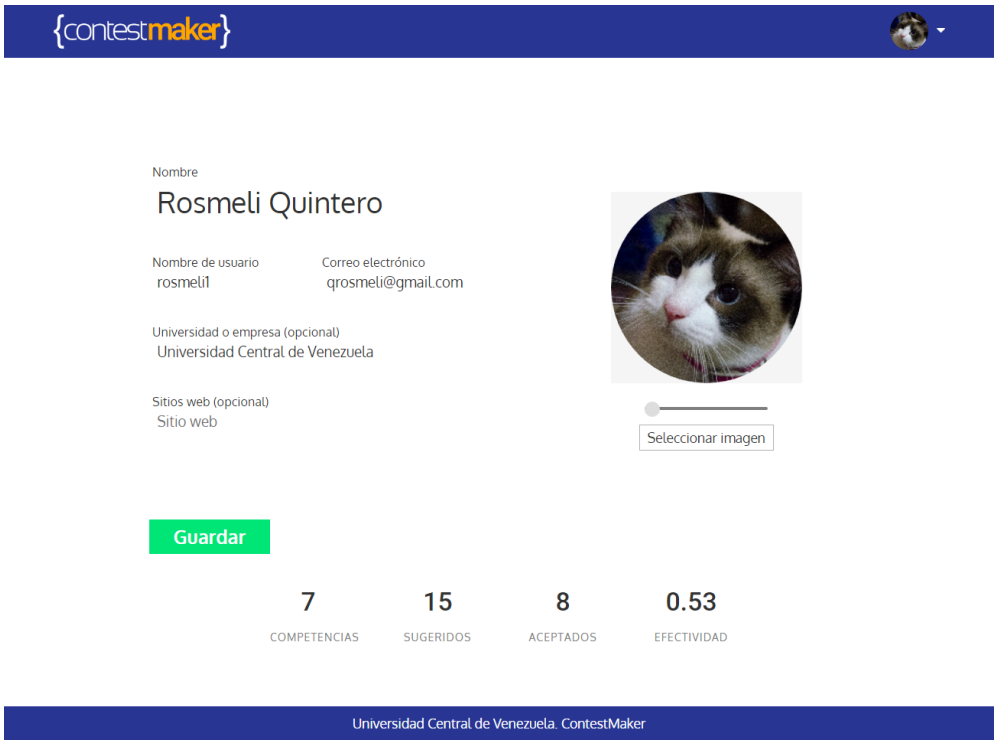


Figura 7.4: Página del perfil de usuario

Si el usuario que solicita la página es el dueño del perfil, se es habilitada la opción de editar su perfil, incluyendo su avatar o foto de perfil. Y se permiten agregar dos campos más de información pública: universidad o empresa y sitio web. Estos dos últimos pensados para que los usuarios puedan identificarse y relacionarse.

7.3. Roles de acceso

Para la aplicación se crearon dos roles de acceso:

- Administrador: un administrador de una competencia tiene todos los permisos para cualquier función. Por defecto, el usuario que crea una competencia es automáticamente administrador de la misma.

- Contribuyente: dirigido a los usuarios que pueden ver y proponer problemas, discutirlos, y votar. Tienen limitaciones en cuanto a editar la competencia.

En la competencia, cualquier administrador es capaz de otorgar y denegar permisos de administrador o contribuyente a un usuario.

7.4. Creación y edición de competencias

Para crear una competencia no es necesaria demasiada información. La aplicación solicita un nombre para la competencia, que será el nombre que aparece en el encabezado del problema, una fecha y un lugar o sitio web según aplique. Como se ve en la figura 7.5, luego de la información básica la aplicación permite agregar usuarios bajo los dos roles: administradores y contribuyentes.

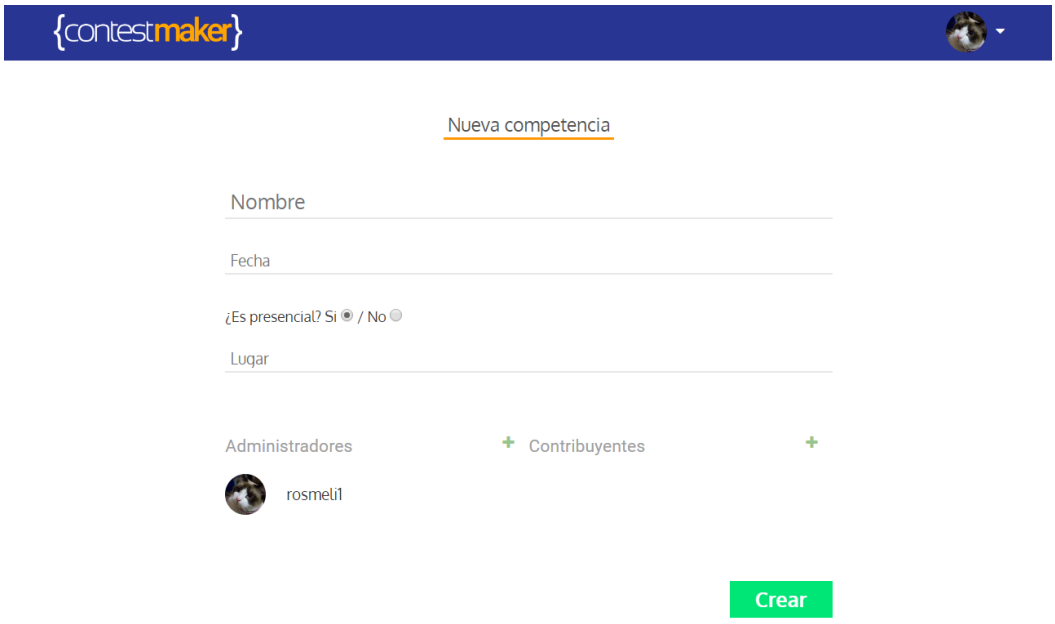


Figura 7.5: Página para crear una competencia

Los usuarios a ser agregados deben tener una cuenta creada en la aplicación, y mediante una búsqueda según nombre de usuario, nombre o correo se sirven los

usuarios que satisfagan la cadena de caracteres suministrada, excluyendo los que ya estén agregados.

Dentro de la competencia, es posible editar todos los datos suministrados anteriormente.

7.5. Creación y edición de problemas

Al ser seleccionada la opción de crear un problema, se es llevado directamente al editor de problemas, que está constituido por varias partes:

7.5.1. Acciones

Esta parte está compuesta por botones que despliegan diferentes tipos de funcionalidades. Desplegadas con íconos y metáforas que representan su acción. En la figura 7.6 se muestran e identifican, estos números corresponden a la descripción en la siguiente lista.

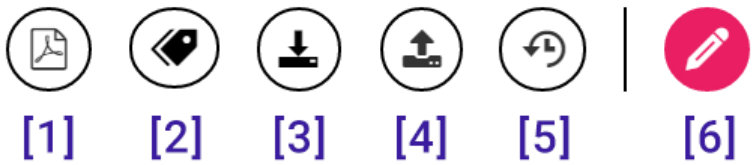


Figura 7.6: Acciones en el editor de problema

1. Ver .PDF: este botón muestra en una pestaña nueva el resultado de la compilación del problema en formato PDF.

2. Añadir tags (etiquetas): al dar click, se despliega un modal que permite relacionar el problema con diferentes tags existentes en la aplicación y también añadirlas en caso de que no existan.

3. Descargar archivos: funcionalidad que permite descargar desde la aplicación un archivo .zip con los archivos del problema.

4. Subir soluciones y casos de prueba: permite subir los archivos de soluciones y/o casos de prueba. Por cada problema se podrán subir varias soluciones y casos de prueba según se requiera. Por cada archivo subido se debe proveer una palabra informativa que describa brevemente el archivo; en caso de las soluciones, puede ser la técnica usada y en caso de los casos de prueba, puede ser el tamaño o dificultad del caso.

5. Editar: activa el editor de contenidos. Este botón luego es ocultado para mostrarse los botones de *guardar* y *cancelar*.

7.5.2. Editor

El editor de problemas está basado completamente en la librería ContentTools y forma parte fundamental de la aplicación. Este editor puede ser visualizado en su estado inactivo en la figura 7.7. Como mencionado anteriormente, esta librería convierte una estructura HTML estática en una estructura completamente editable por el usuario sin perder propiedades de estilo ni posición. De esta forma, una página estática puede ser convertida en un editor WYSIWYG.

Los problemas de programación tienen una estructura básica que debe ser respetada, esta es descrita en el capítulo 1, por tal razón el editor está restringido a sólo modificar las regiones permitidas: título, código, descripción de problema, descripción de entrada y salida, ejemplo y restricciones. Los títulos u orden de cada una de estas secciones no pueden ser modificadas.

Además de modificar texto, el editor permite la agregación de diversos elementos. Dichos elementos también fueron limitados según el sentido de un problema de programación.

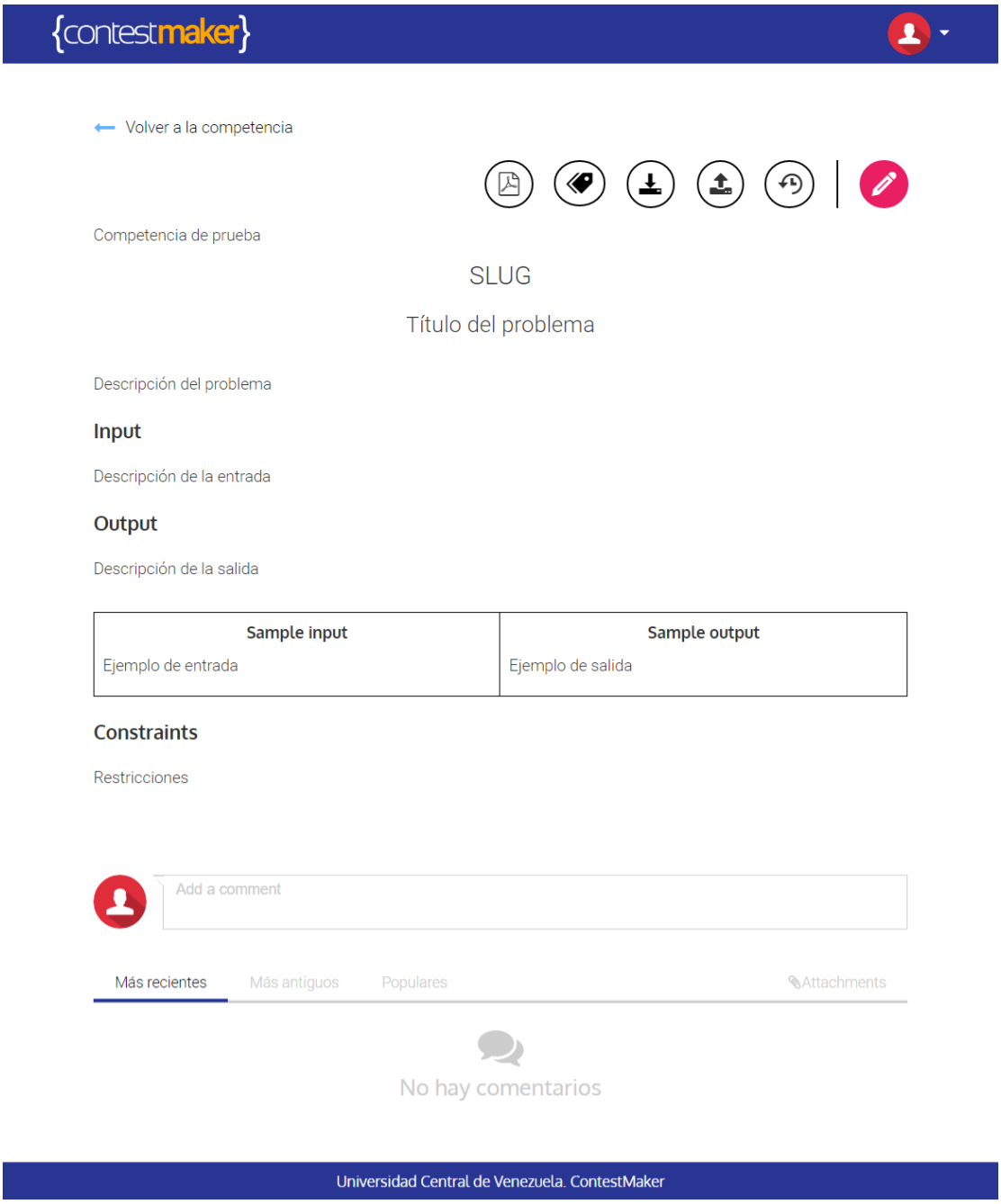


Figura 7.7: Editor de problema

Los elementos posibles de insertar están desplegados en una caja de herramientas flotante, estos son:

- Negrilla
- Itálica
- Link
- Lista con viñetas
- Lista enumerada
- Tabla
- Imagen

Donde únicamente en la descripción de problema es posible insertar una imagen. Y en la sección de ejemplo, sólo es posible usar herramientas de texto: no es posible insertar listas, ni tablas ni imágenes.

Además de restricciones la librería también fue modificada para agregar 4 herramientas más a su caja de herramientas. Estas son:

- Código LaTeX incrustado: permite escribir código LaTeX puro en la línea para luego ser renderizado al compilar. Esto con el fin de dar libertar a los usuarios más avanzados de redactar usando elementos que tal vez no provee el editor.
- Código LaTeX centrado: similar al item anterior pero este se presenta centrado como un elemento independiente.
- Formula matemática incrustada: permite escribir fórmulas matemáticas en la misma línea bajo la sintaxis LaTeX que luego serán renderizadas al compilar. Funciona para fórmulas, variables o cualquier elemento dentro del espacio matemático de LaTeX.
- Formula matemática centrada: similar al item anterior pero este se presenta centrado como un elemento independiente.

La caja de herramientas puede ser visualizada en la figura 7.8.

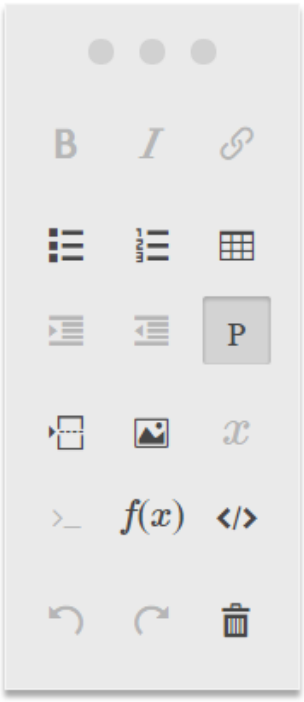


Figura 7.8: Caja de herramientas

Al escribir una formula matemática y dejar de editarla el procesador MathJax se encarga de convertir ese texto en diversos elementos HTML combinados con CSS que permite una visualización gráfica de la fórmula similar o igual a la que daría al ser compilada. Al tratar de editar la fórmula nuevamente, esta es convertida en el texto plano completamente editable.

Al darle al botón *guardar* se hace una solicitud AJAX con los elementos HTML modificados para luego ser guardado en el servidor.

7.5.3. Etiquetas (tags)

Con la finalidad de tener una clasificación que mejore la navegación de problemas, por cada problema pueden ser añadidas varias etiquetas que lo describan. Estas

etiquetas corresponden a las técnicas que son necesarias aplicar al problema para solucionarlo. Como las técnicas generalmente se pueden repetir, se implementó un banco de técnicas en el que cualquier usuario puede hacer su contribución, y a partir de este banco, se pueden asociar técnicas al problema. Esto con el fin de evitar procesos repetitivos al usuario. La interfaz para añadir etiquetas puede ser detallada en la figura 7.9.

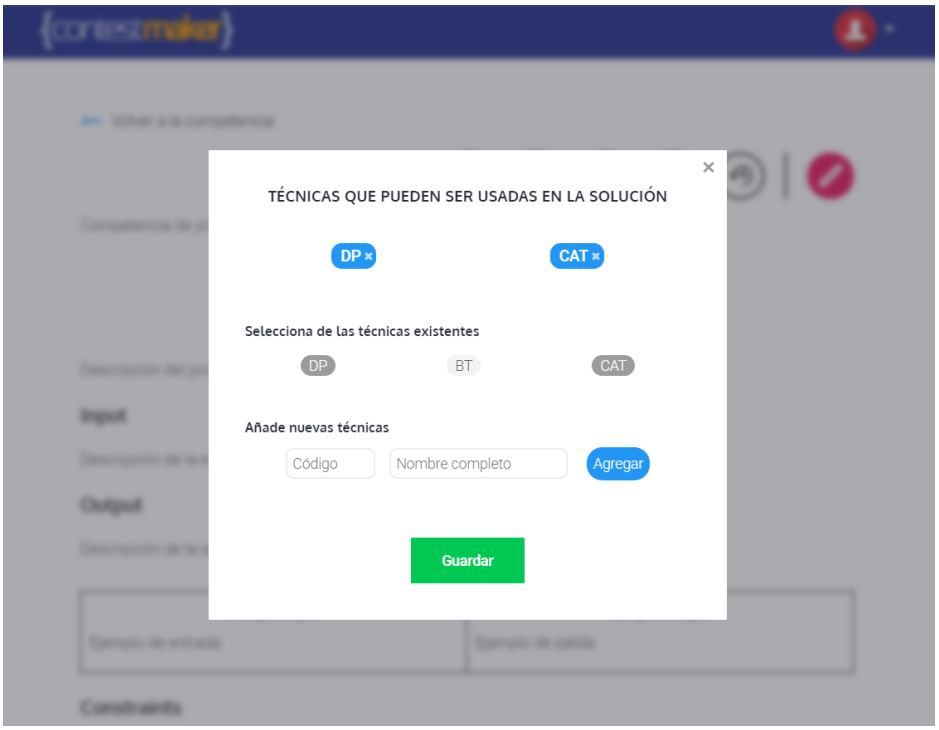


Figura 7.9: Interfaz para añadir etiquetas

7.5.4. Subir soluciones y casos de prueba

Relacionar un problema con soluciones y casos de prueba es una tarea importante y necesaria para muchos. Por cada problema se pueden subir muchos archivos correspondientes al código fuente de las soluciones y archivos en texto plano correspondientes a los casos de prueba. La interfaz para subir soluciones puede ser visualizada en la figura 7.10, la interfaz para subir casos de pruebas es muy similar.

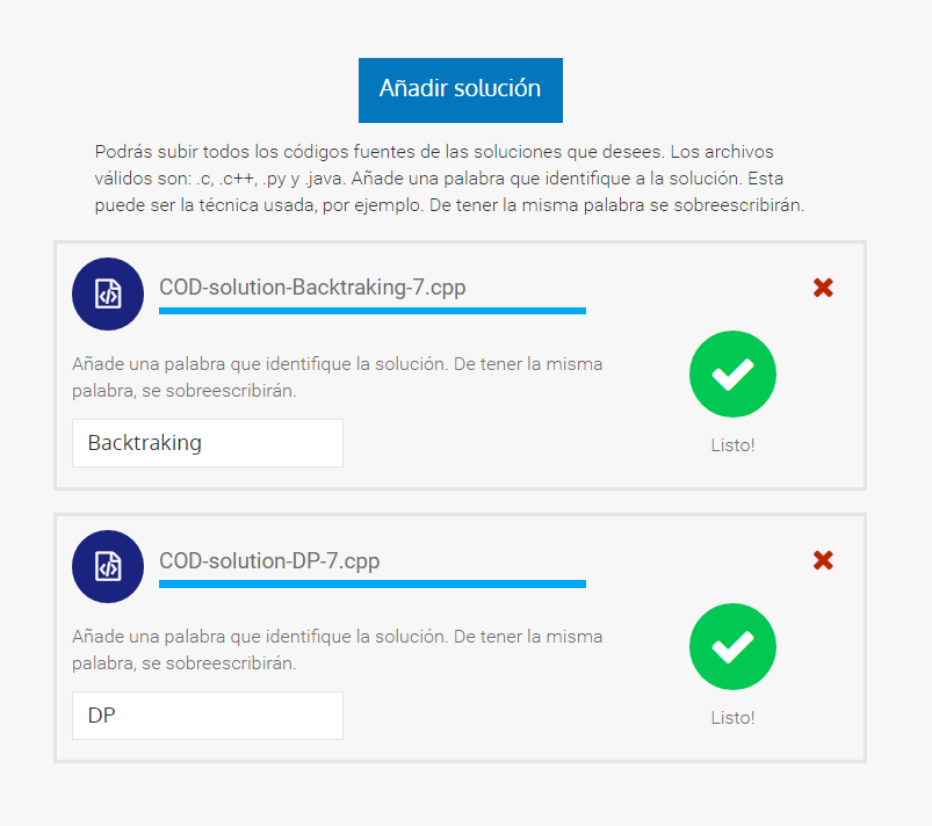


Figura 7.10: Interfaz para subir soluciones

Para subir cada archivo es solicitado al usuario una “Palabra informativa”, esta es necesaria para identificar de una manera sencilla al archivo.

Para las *soluciones* esta palabra puede ser, por ejemplo, la técnica usada, así otros usuarios sabrán identificar cada archivo, si el usuario quiere subir una solución una palabra informativa, repetida se le obliga al usuario a cambiarla, obligando así, a identificar mejor la solución.

Para los *casos de prueba*, la palabra es usada con dos fines: identificar el caso de prueba, como por ejemplo, el tamaño o dificultad (en la comunidad es común usar nombres como ”small”, ”medium”, etc.) y asociar la entrada con su salida. Esto significa que si el usuario desea subir un caso de prueba, debe subir dos archivos con la misma palabra informativa: uno con la entrada y otro con la salida esperada según

la entrada.

7.5.5. Sistema de Control de Versiones

Como fue descrito en los capítulos anteriores, el control de versiones, usando el sistema Git [23], puede usarse tanto localmente como en un repositorio. La aplicación hace uso de estas dos formas. Como se puede ver en la figura 7.11, las entidades relacionadas al control de versiones son:

- Enunciado del problema: usando LaTeX, los enunciados de los problemas están desplegados en texto plano, por lo que el versionado del mismo es posible. Sin embargo, se requiere que las versiones no estén ligadas a una plantilla LaTeX en específico, ya que estas pueden cambiar, por lo que, aparte del archivo fuente *.TEX*, también es versionada una adaptación en formato *JSON* de los campos editables en el problema. De esta manera, el formato es completamente transparente, por lo que usuarios poco experimentados pueden hacer uso sin problemas. La visualización de la versión se retorna en un modo web utilizando la misma interfaz del editor de problema en modo sólo lectura, permitiendo que cualquier usuario visualice los cambios rápidamente.

- Soluciones y casos de prueba: los códigos fuentes de las soluciones y casos de prueba son versionados cada vez que el usuario los reemplaza. Son visualizados en modo web. Como ayuda y para ayudar al usuario el código fuente es estilizado. Los elementos como palabras reservadas del lenguaje, literales y operadores tienen colores y estilos diferentes para que la lectura del archivo sea más sencilla.

El sistema de control de versiones realiza todas las operaciones localmente. Pero tiene la capacidad de asociarse con un proveedor de servicio de alojamiento como BitBucket o GitHub usando una autenticación basada en llaves SSH.

Figura 7.11: Interfaz para el Sistema de Control de Versiones

7.6. Plantillas (Template)

Como fue discutido anteriormente, Django provee plantillas que permiten establecer una estructura de elementos, generalmente son usados para generar archivos HTML como respuesta a una solicitud de un cliente. En esta aplicación también se usaron para generar la estructura de los archivos .TEX.

7.6.1. Problema

La plantilla usada para la estructura de un problema fue**:

```
\begin{document}
    \problem{ {{titulo_problema} }{prev}
        {{ descripcion_problema }}
        \subsection{Input}
            {{ descripcion_entrada }}
        \subsection{Output}
            {{ descripcion_salida }}
    \begin{minipage}[c]{\textwidth}
   \begin{center}
   \begin{tabular}{|l|l|} \hline
        \begin{minipage}[t]{0.475\textwidth}
         \bf{Sample input}
         \begin{verbatim}
             {{ejemplo_entrada}}
         \end{verbatim}
         \vskip 12pt
       \end{minipage}
       &
      \begin{minipage}[t]{0.475\textwidth}
         \bf{Output for the sample input}
         \begin{verbatim}
            {{ejemplo_salida}}
         \end{verbatim}
         \vskip 12pt
         \end{minipage} \\
       \hline
   \end{tabular}
   \end{center}
    \end{minipage}
\end{document}
```

Donde las variables `titulo_problema`, `descripcion_problema`, `descripcion_salida`, `descripcion_entrada`, `ejemplo_entrada`, `ejemplo_salida` son sustituídas por el contenido correspondiente.

7.6.2. Competencia

La plantilla usada para generar el archivo que recopila todos los problemas fue:

```
\begin{document}
    {% for problema in problemas %}
        \input{ {{ problema.codigo }} }
    {% endfor %}
\end{document}
```

Donde `problemas` es un diccionario de los problemas seleccionados para competencia y `\input` copia cada archivo .TEX cuyo nombre sea `problema.codigo`.

7.7. Generación de archivos

Por cada competencia y problema se generan archivos según cierta solicitud. A continuación se detalla este proceso.

7.7.1. Problema

Cuando la opción *Ver .PDF* es solicitada, se procede a:

1. Usando el template para la estructura de un problema referido anteriormente y la función `render_to_string` de Django, se crea una cadena de caracteres. Esta cadena constituye los datos HTML del problema dispuestas en la plantilla en formato LaTeX.
2. Los elementos HTML son convertidos en elementos LaTeX.
3. Se crea un archivo con el resultado del paso anterior.
4. El archivo de texto creado es compilado con el programa *pdflatex* que tiene como salida el archivo *PDF*.

Las imágenes son guardadas en una subcarpeta del problema llamada “images” y usadas con el macro de LaTeX `\includegraphics`

Si existe una plantilla y/o estilo personalizado para el problema, la generación del archivo *.PDF* es similar, pero en lugar de usar el template y estilo por defecto, son usados los archivos suministrados por el usuario. Todos los archivos necesarios para la generación y los archivos generados están guardados dentro de una subcarpeta llamada `custom`, así, si existe algún error en la generación, siempre estarán disponibles los arhivos originales.

7.7.2. Competencia

Al ser terminada la votación (se verá más adelante), se corre un algoritmo para generar los archivos necesarios de la competencia. De forma general, los pasos son los siguientes:

1. Se crea un directorio para la competencia.
2. Para cada problema seleccionado para competencia, se generan los archivos y directorios necesarios y se copia dicha carpeta al directorio de la competencia.
3. Es copiado el archivo *.TEX* correspondiente a cada problema sin las cabeceras `\begin{document}` y `\end{document}` y las imágenes (si existen) de cada problema en una subcarpeta llamada "images".
4. Se crea un archivo llamado `problems.tex` que corresponde a la salida de la plantilla de la sección anterior usando la función `render_to_string`.
5. Se crea un proceso para compilar el archivo problems.TEX con el programa *pdflatex*.

7.8. Descarga de archivos

7.8.1. Problema

Por cada problema es generado un archivo .ZIP con todos los archivos necesarios para ver y compilar el problema. Este archivo consta de:

- `solutions`: contiene todos los archivos de texto subidos por los usuarios correspondientes a las soluciones.
- `test-cases`: contiene todos los archivos de texto subidos por los usuarios correspondientes a los casos de prueba.
- `images`: contiene todos las imágenes que el usuario relacionó y subió con el uso del editor de problema. Estas imágenes son usadas al momento de compilar el enunciado del problema.
- `latex`: contiene todos los archivos inherentes directamente a la compilación LaTeX, tanto `statement.tex`, `contest.sty` (estilo) y generados. Permite un ambiente de compilación local (si el usuario lo desea)
- `generated_statement.pdf`: corresponde al archivo PDF resultado del compilado de `statement.tex` generado por la aplicación.
- `custom`: contiene los archivos statement.tex, statement.pdf, y estilo. (Ver sección 7.10)

7.8.2. Competencia

Al ser revisada la votación (se verá más adelante), los usuarios de la competencia podrán descargar los archivos creados en la misma. Un archivo comprimido es generado y servido por la aplicación. En este archivo está contenido todos los archivos *.TEX* y *.PDF* generados a partir del set de problemas.

La estructura de carpetas por competencia es bastante simple y consta básicamente de dos partes:

1. Competencia: esta consta de los archivos *.TEX* necesarios para generar el documento del set de problemas, así como el *.PDF* generado. Esto está constituido de los archivos *.TEX* de cada problema individualmente identificados por su código, sus imágenes y el archivo textit.TEX que se encarga de recopilar todos los problemas.

2. Problemas: por cada problema seleccionado para competencia se crea una carpeta que contiene los archivos *.TEX* y *.PDF* pertenecientes al enunciado del mismo, una carpeta con los casos de prueba, una carpeta con las soluciones y otra con las imágenes usadas en el enunciado.

En caso de existir un estilo personalizado (ver sección 7.10), un directorio `custom` es creado, que sigue la misma estructura de carpetas señalada anteriormente.

7.9. Votación

Cuando un administrador así lo decida puede llamar a votación. Esto con el fin de decidir cuáles problemas serán usados en competencia. Cuando esta acción es solicitada, se cambia el estatus de la competencia, se bloquea cualquier opción para editar los y la cantidad de problemas, es decir, crear, editar, eliminar e importar problemas y se le permite a los usuarios votar.

7.9.1. Voto

El voto consiste de dos variables, como se ejemplifica en la figura 7.12 una calificación del 1 a 5, proyectando el agrado del problema y la dificultad, entre la que el usuario podrá elegir entre “fácil”, “medio.º “difícil”. El usuario deberá votar sobre la

calificación y dificultad de cada problema que exista en la competencia. Mientras la votación siga abierta, el usuario podrá editar su voto.

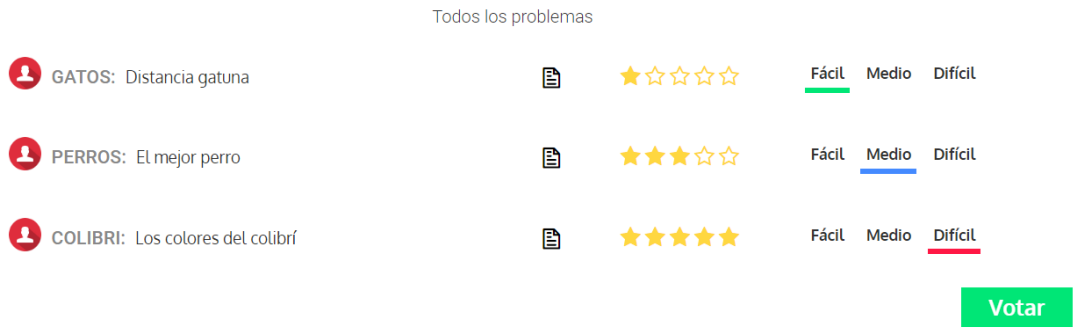


Figura 7.12: Voto para un problema

7.9.2. Cierre de votación y revisión de resultados

Existen dos maneras de cierre de votación: cuando un administrador así lo decida o cuando todos los usuarios hayan votado. Cuando una votación es cerrada, un administrador deberá revisar los resultados.

La aplicación no toma la decisión sobre qué problemas irán a competencia. Los resultados promediados son servidos al administrador en la página de revisar votación. En ella, el administrador introduce cuántos problemas de cada dificultad son necesarios y en base a estos números se genera un set de problemas tentativo según las calificaciones de los usuarios.

Para una competencia de prueba con sólo 3 problemas, en la figura 7.13, se observa la interfaz luego de introducir una meta de 1 problema fácil y 1 problema difícil y darle click al botón "Generar Problem Set".

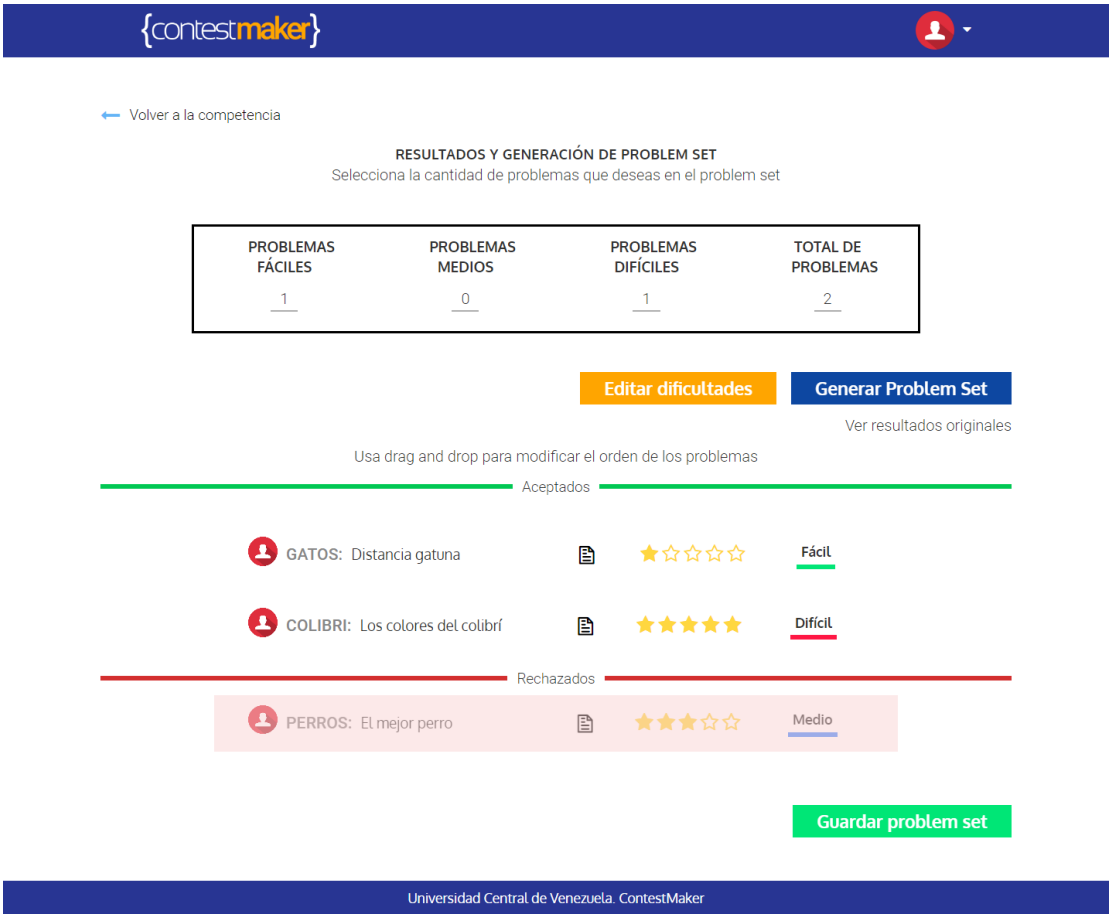


Figura 7.13: Voto para un problema

La aplicación tomará los mejores n problemas por cada dificultad, dándole el mayor peso al criterio de la dificultad. Sin embargo, todo es editable. Según los votos, el administrador puede decidir si editar las dificultades de los problemas o modificar mediante *Drag and Drop* este set de problemas según los criterios que considere y guardar esa decisión de forma definitiva. Luego de cerrar la votación, todos los usuarios tendrán acceso a los archivos y *.PDF* generado de la competencia.

7.10. Estilo personalizado

El estilo y la apariencia de los *.PDF* generados están dados bajo dos elementos: la plantilla del problema y el estilo LaTeX. La plantilla por defecto hace uso del comando personalizado `\problem` que tiene como argumento el título del mismo. El estilo de este comando está especificado en el archivo `contest.sty` que se encarga de añadir nombre de la competencia y una letra asociada al problema, por ejemplo, `Problem A`, márgenes, tamaño de fuente, entre otras cosas.

Al darse como finalizada la competencia (después de revisar los resultados), es activada la opción de cambiar estilo. Esta opción, dirigida a usuarios con experiencia en LaTeX, permite subir un nuevo archivo de estilo en formato *.STY* y una nueva plantilla para la renderización de problemas.

Para personalizar el estilo el usuario puede subir dos archivos: plantilla del problema y estilo, similares a las plantillas discutidas en la sección anterior, pero las variables el lugar de ser de tipo `{{ ... }}` son del tipo `|- ... -|`. En estos archivos el usuario puede indicar la estructura LaTeX que deberán seguir los problemas y tu estilo si así lo desea, haciendo uso de las variables suministradas del tipo `|- ... -|` que serán reemplazadas por su contenido correspondiente.

Al aplicar un estilo personalizado, se sigue el siguiente proceso:

1. Se crea un subdirectorio dentro de la competencia llamada `custom`, donde residirán todos los archivos.

2. Por cada problema dentro de la competencia, se generan los archivos necesarios haciendo uso de la nueva plantilla y/o estilo. Estos archivos residen dentro del directorio del problema dentro de la carpeta `custom`. El contenido de esta carpeta es luego copiado al subdirectorio de la competencia `custom`.

3. Es copiado el archivo .TEX correspondiente a cada problema sin las cabeceras `\begin{document}` y `\end{document}`.

4. Se crea un archivo llamado `problems.TEX` que corresponde a la salida de la plantilla de la sección anterior usando la función `render_to_string`.

5. Se crea un proceso para compilar el archivo `problems.TEX` con el programa *pdflatex*.

Capítulo 8

Resultados y pruebas

Los resultados tangibles y la utilidad más importante que ofrece la aplicación desarrollada en este TEG, es la obtención del archivo *.PDF* y los otros archivos descritos en la sección 7.8. A continuación se describirán los resultados obtenidos al momento de generar un *.PDF* correspondiente a un problema y el archivo *.ZIP* correspondiente a la descarga de la competencia.

Para validar la efectividad de este proceso, la aplicación pasa por una serie de pruebas cualitativas obtenidas mediante la realización de encuestas a un grupo voluntario de profesores y estudiantes.

8.1. Resultados

8.1.1. Enunciado de un problema

Haciendo uso de la aplicación, se crea un enunciado de prueba que haga uso de la mayor parte de las herramientas disponibles, estas son: cursiva, negrilla, inserción de imágenes, tablas, listas, formulas centradas y no centradas.

El enunciado ya redactado, se visualiza en la figura 8.1, al generar el *.PDF* y usando el comando *pdflatex*, se obtiene como resultado el archivo *.PDF* que se observa en la figura 8.2

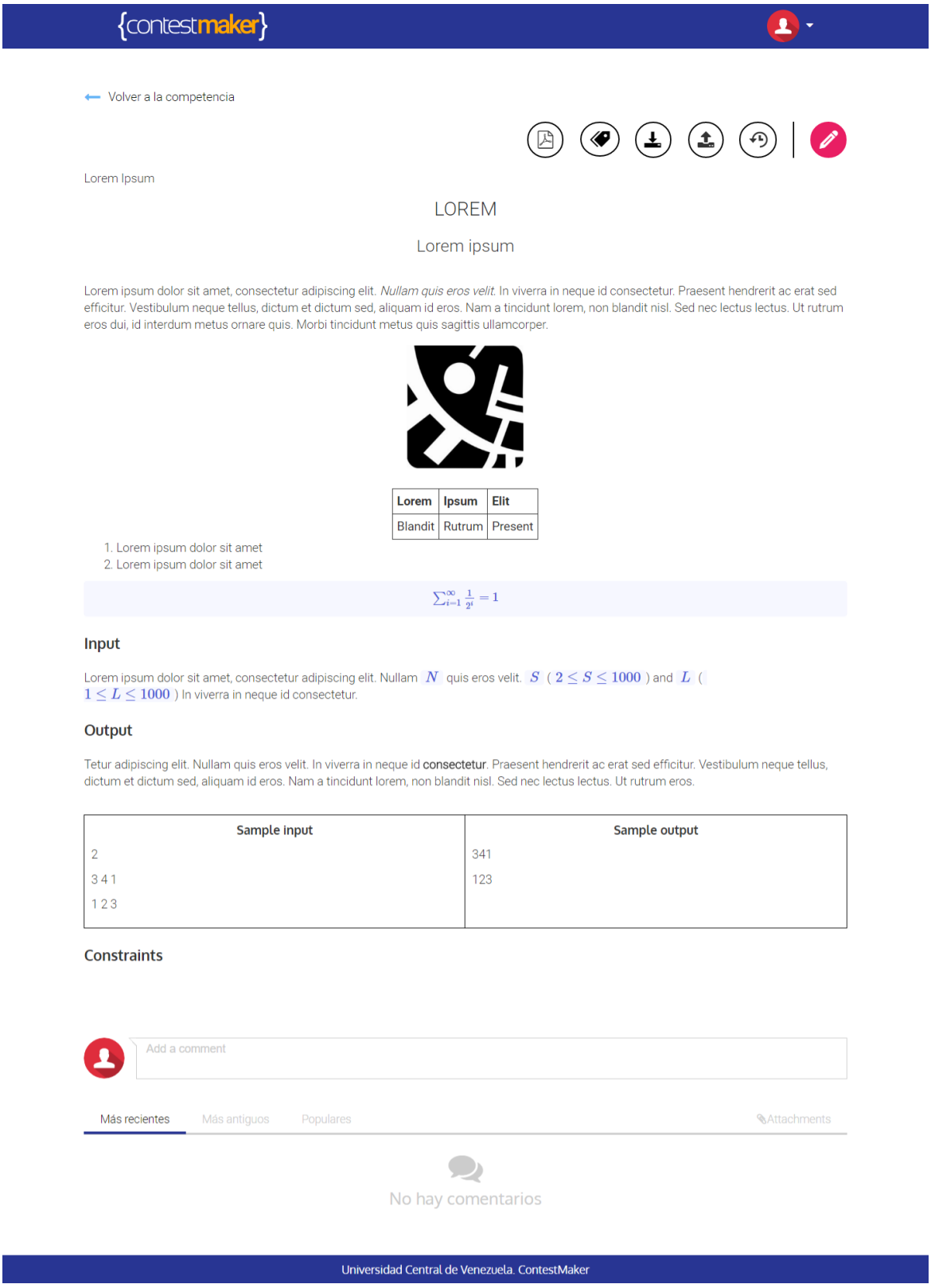


Figura 8.1: Interfaz para el enunciado redactado haciendo uso de la aplicación

Problem A
Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. *Nullam quis eros velit.* In viverra in neque id consectetur. Praesent hendrerit ac erat sed efficitur. Vestibulum neque tellus, dictum et dictum sed, aliquam id eros. Nam a tincidunt lorem, non blandit nisl. Sed nec lectus lectus. Ut rutrum eros dui, id interdum metus ornare quis. Morbi tincidunt metus quis sagittis ullamcorper.

Lorem	**Ipsum**	**Elit**
Elit	Elit	Elit

1. Lorem ipsum dolor sit amet
2. Lorem ipsum dolor sit amet

$$\sum_{i=1}^{\infty} \frac{1}{2^i} = 1$$

Input

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam N quis eros velit. S ($2 \leq S \leq 1000$) and L ($1 \leq L \leq 1000$) In viverra in neque id consectetur.

Output

Tetur adipiscing elit. Nullam quis eros velit. In viverra in neque id **consectetur**. Praesent hendrerit ac erat sed efficitur. Vestibulum neque tellus, dictum et dictum sed, aliquam id eros. Nam a tincidunt lorem, non blandit nisl. Sed nec lectus lectus. Ut rutrum eros.

Sample input	**Output for the sample input**

`2`
 	
`3 4 1`
 	
`1 2 3`
	`341`
	
	`123`

Figura 8.2: Archivo *.PDF* generado

8.1.2. Descarga de archivos

Haciendo uso de la aplicación, se sube un archivo que corresponde a la solución del problema y otros dos que corresponden a un caso de prueba: entrada y salida según la entrada. Como la palabra informativa se usa: "Palabra-Prueba". La interfaz gráfica de este proceso en la aplicación se observa en la figura 8.3. Al hacer click en la opción de descarga de problema, se descarga un archivo *.ZIP* con el nombre `Lorem` que corresponde al código del problema. En este archivo están disponibles todos los archivos y directorios descritos en la sección 7.8. En la figura 8.4 se aprecia en detalle dicho comprimido *ZIP*.

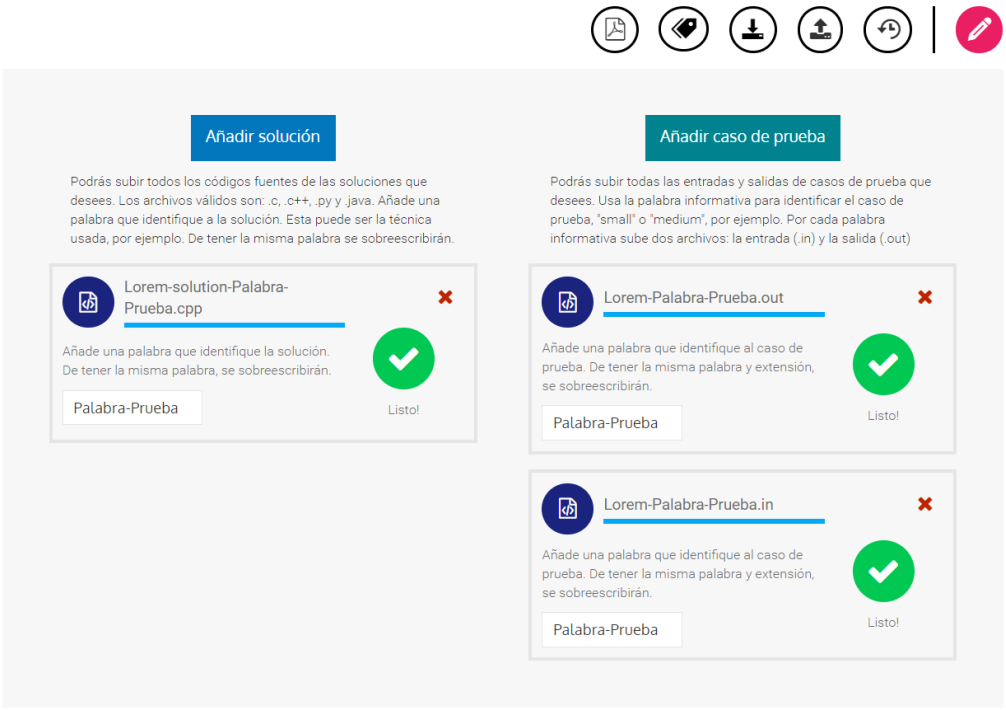


Figura 8.3: Interfaz para para la subida de archivos

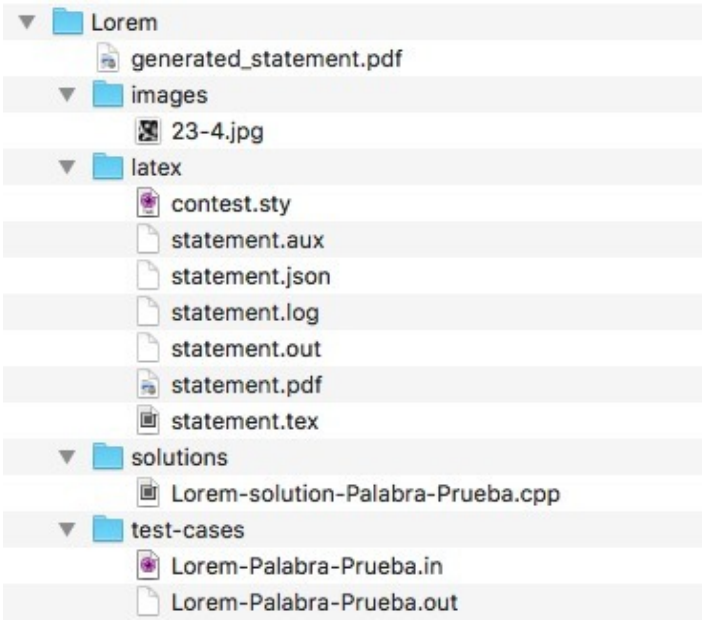


Figura 8.4: Detalle del comprimido descargado

8.2. Pruebas de aceptación

Se le pidió a una serie de estudiantes y profesores de la Facultad de Ciencias de la UCV, todos estos pertenecientes a la comunidad de Programación Competitiva. Se les pidió que utilizaran la aplicación según sus necesidades y luego tomaran un cuestionario dividido en secciones para medir las opiniones de estos sobre la aplicación. El cuestionario consiste en preguntas y afirmaciones a las que el usuario puede dar respuestas desde "Totalmente en desacuerdo.ª "Totalmente de acuerdo" o "No lo hice".

8.2.1. Interfaz

Esta sección está enfocada los aspectos grosos de la interfaz y sensaciones del usuario al usar la aplicación. Las preguntas y respuestas pueden ser visualizadas en la figura 8.5.

Figura 8.5: Prueba de aceptación: interfaz

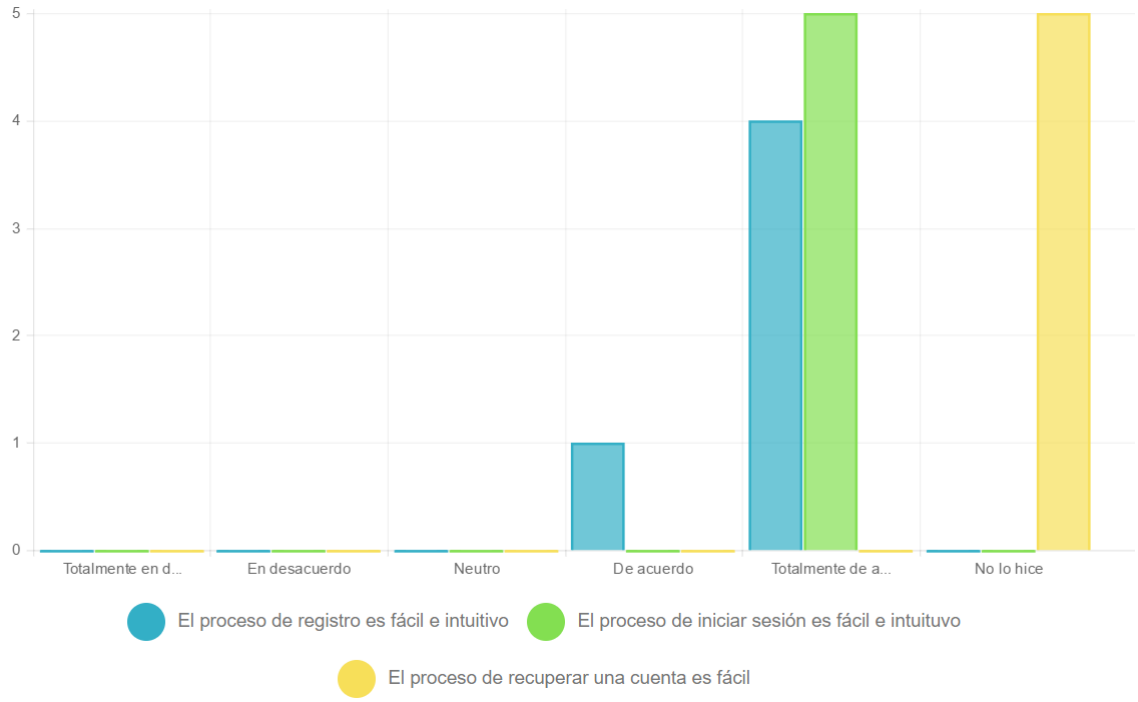


Figura 8.6: Prueba de aceptación: cuentas

Se puede observar que la mayoría de las votaciones son favorables. Sin embargo, hay un voto neutro para los items de "La distribución de los items es agradablez "Mis acciones tienen un feedback de estado".

8.2.2. Cuentas

Esta sección se enfoca en el manejo y uso de cuentas de los usuarios. Las preguntas y respuestas pueden ser visualizadas en la figura 8.6.

En donde todas las votaciones fueron favorables, con un sólo voto "De acuerdo". Sin embargo, ningún usuario necesitó usar la funcionalidad de recuperar una contraseña.

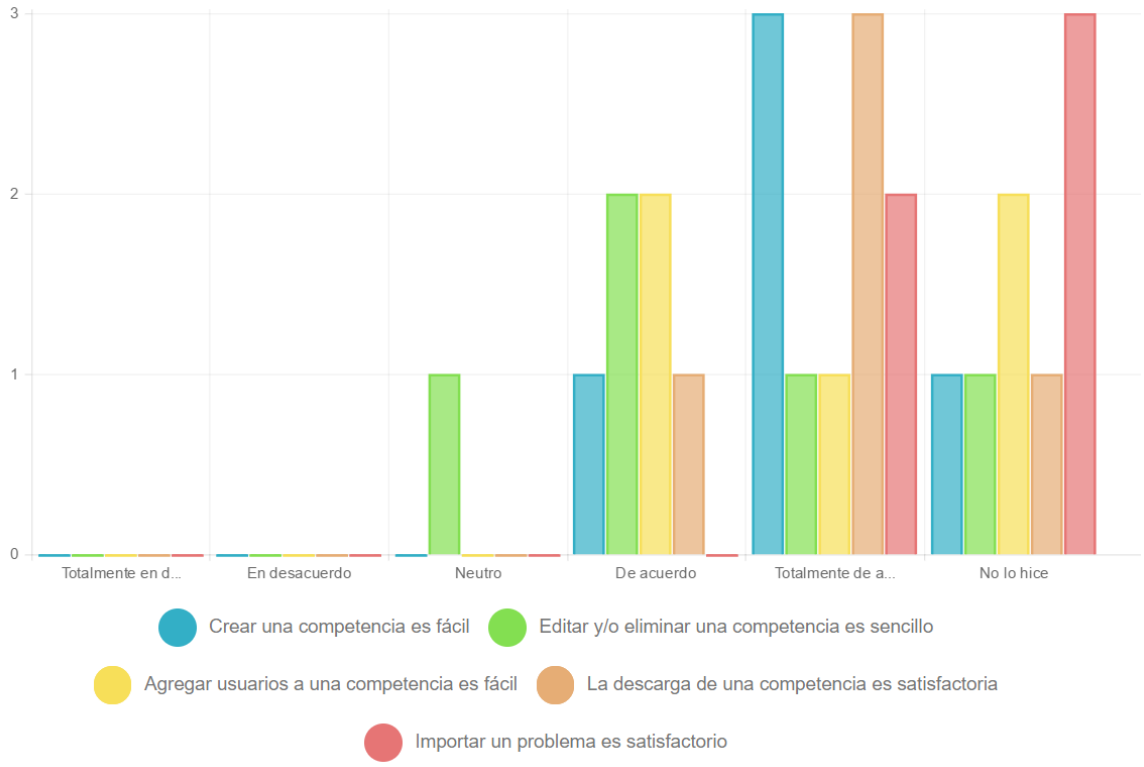


Figura 8.7: Prueba de aceptación: competencia

8.2.3. Competencia

Centrado en las funciones propias de las competencias, se mide la opinión de los usuarios. Las preguntas y respuestas pueden ser visualizadas en la figura 8.7.

En esta sección un usuario opinó que editar una competencia no era del todo sencillo, mientras las demás votaciones estuvieron distribuidas entre "De acuerdo" y "Totalmente de acuerdo". Cabe destacar que crear, editar y agregar usuarios a una competencia son acciones que sólo puede realizar el rol de administrador, por lo que en esta sección hay muchos "No lo hice". Sólo 3/5 de los usuarios importaron un problema a la competencia.

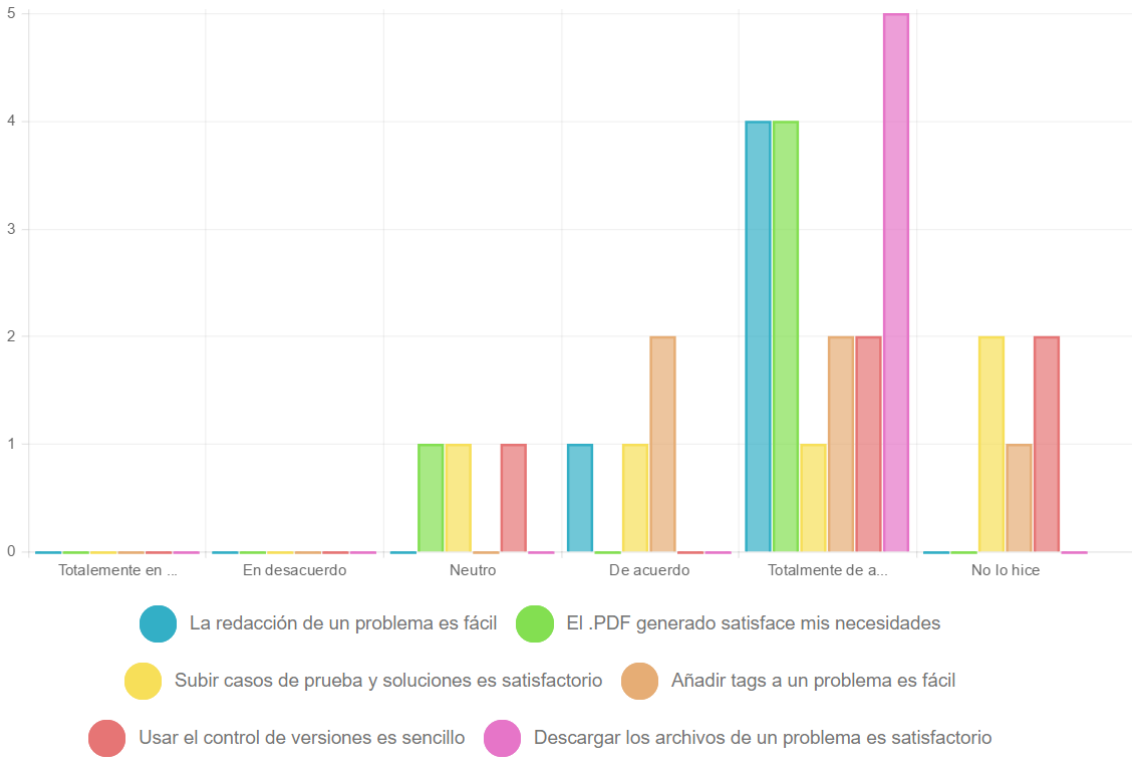


Figura 8.8: Prueba de aceptación: problema

8.2.4. Problema

Centrado en las funciones propias de los problemas, se mide la opinión de los usuarios. Las preguntas y respuestas pueden ser visualizadas en la figura 8.8.

Las acciones más complicadas se encuentran directamente relacionadas a los problemas. En los resultados se observa que la generación del *PDF*, la subida de archivos y el uso de control de versiones obtuvieron 1 voto neutro cada uno. Es posible que algún usuario haya presentado algún error, que debería ser posteriormente revisado. Del resto de las votaciones, estuvieron en "Totalmente de acuerdo" . Cabe destacar que hay usuarios que no subieron archivos, ni usaron el control de versiones ni añadieron tags al problema.

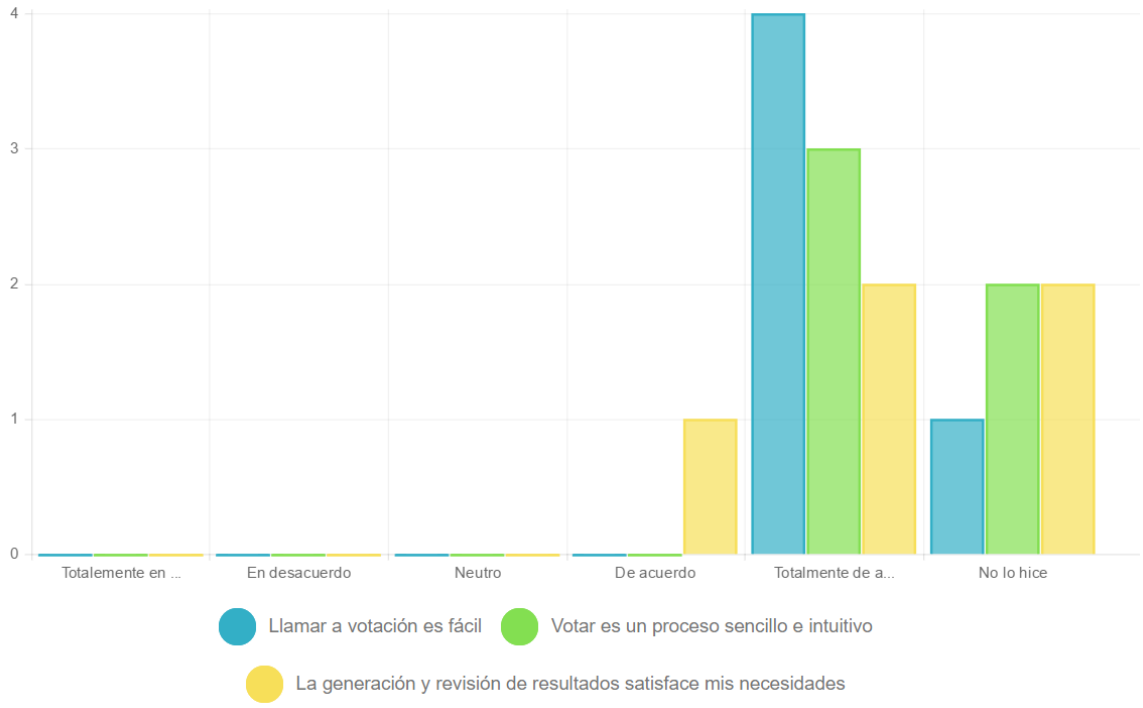


Figura 8.9: Prueba de aceptación: votación

8.2.5. Votación

Esta sección se enfoca en proceso de votación. Las preguntas y respuestas pueden ser visualizadas en la figura 8.9.

Para el proceso de votación, las opiniones de los usuarios fueron bastante favorables. Nuevamente, muchos usuarios no usaron ciertas funcionalidades.

8.2.6. Recomendación

Para finalizar, se le preguntó a los usuarios si recomendarían la aplicación desarrollada en el marco de este TEG y un 100 % de los usuarios respondieron afirmativamente.

Capítulo 9

Conclusiones y trabajos futuros

Para finalizar este trabajo especial de grado, se presentan las conclusiones tras la investigación, desarrollo y pruebas de la aplicación. Seguidamente se sugieren trabajos futuros para el crecimiento de este desarrollo.

9.1. Conclusiones

El objetivo de este Trabajo Especial de Grado se cumplió satisfactoriamente, ya que se logró desarrollar una aplicación web usable que permite gestionar eficientemente los problemas para competencias de programación.

La investigación y, la posterior, aplicación web desarrollada, cumplen satisfactoriamente con todos los objetivos específicos planteados. Se desarrolló una aplicación que permite la redacción de problemas y su asociación a una competencia, generación de archivos *.TEX* y *.PDF* correspondientes, visualización del *.PDF* generado y subida y descarga de soluciones y casos de prueba usando un sistema de control de versiones que permite la recuperación de cualquier cambio en cualquier momento. Además, proporciona a los usuarios de un foro de discusión por cada problema y competencia, un proceso votación automatizada para la selección de problemas, asegurando los datos por medio de la autorización.

El framework *Django* y por extensión el lenguaje *Python* permitieron un desarrollo ágil, ya que su estructura y facilidades integradas permitieron acortar el tiempo de implementación del lado de servidor. Asimismo, las herramientas utilizadas del lado del servidor y del lado del cliente contribuyeron notablemente al desarrollo, proveyendo utilidades ya usadas y probadas por otras comunidades.

Las opiniones de usuarios pertenecientes a la Facultad de Ciencias entregadas a través de la prueba de aceptación fueron favorables. Sin embargo, existen aspectos sencillos referentes a la interfaz que podrían ser mejorados para una mejor experiencia de usuario.

La aplicación desarrollada genera aportes importantes para la comunidad de Programación Competitiva de la Facultad de Ciencias, ya que provee de un sistema dedicado e integrado para la gestión de problemas, y no sólo una adaptación como se venía haciendo.

9.2. Trabajos Futuros

Algunos de los posibles objetivos que se plantean para continuar con la línea de desarrollo del sistema a partir de la investigación realizada son los siguientes:

- Inclusión de recursos en la interfaz que permitan al usuario a aprender a usar la aplicación rápidamente.
- Campos de búsqueda y filtros para competencias y problemas.
- Extensión de las funciones de seguridad para que el sistema tenga una protección alta ante ataques malintencionados.
- Edición de enunciado colaborativo en tiempo real.
- Integración de un sistema de notificaciones de eventos en tiempo real.

- Visualización gráfica e interactiva de las diferencias del item actual y una versión seleccionada (SCV). Esto aplica para enunciados, soluciones y casos de prueba.

- Estadísticas de los problemas con solución basada en cierta técnica que más son propuestos y/o seleccionados a competencia. Esto con el fin de ayudar a los usuarios a decidir qué tipo de problema redactar o seleccionar, permitiendo que las competencias sean más diversas.

- Implementación de un módulo que permita la inscripción y gestión de competidores a diferentes competencias e integración con un juez, permitiendo organizar todos los aspectos de una competencia desde una misma plataforma.

Bibliografía

[1] Association for Computing Machinery. Página oficial. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: http://www.acm.org/

[2] International Olympiad in Informatics. Página oficial. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: http://www.ioinformatics.org/index.shtml

[3] Google Code Jam. Google. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: https://code.google.com/codejam/

[4] Facebook Hacker Cup. Facebook. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: https://www.facebook.com/hackercup/

[5] Top Coder. Página oficial. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: https://www.topcoder.com/

[6] Hacker Rank. Página oficial. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: https://www.hackerrank.com/

[7] Codeforces. Página oficial. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: http://www.codeforces.com/

[8] Sphere Online Judge. Página oficial. [Fecha de consulta: 03 de Junio de 2016]. Disponible en: http://www.spoj.com/

[9] Typesetting. Wikipedia, the free Encyclopedia. [Fecha de consulta: 04 de Junio de 2016]. Disponible en: https://en.wikipedia.org/wiki/Typesetting

[10] The teTeX Homepage. TeX Users Group. [Fecha de consulta: 04 de Junio de 2016]. Disponible en: https://www.tug.org/tetex/

[11] TeX Live. TeX Users Group. [Fecha de consulta: 04 de Junio de 2016]. Disponible en: https://www.tug.org/texlive/

[12] MiKTeX. Página oficial. [Fecha de consulta: 04 de Junio de 2016]. Disponible en: http://miktex.org/

[13] Kopka, H., Daly, P. (2004). A Guide to LaTeX: Addison-Wesley.

[14] Sommerville, I. (2005) Ingeniería del Software: Pearson Educación.

[15] Krasner, G., Pope, S. A Cookbook for Using View-Controller User Interface Paradigm in Smalltalk-80. ParcPlace Systems. Disponible en: https://www.lri.fr/~mbl/ENS/FONDIHM/2013/papers/Krasner-JOOP88.pdf

[16] Python. Wikipedia, La Enciclopedia Libre. [Fecha de consulta: 05 de Junio de 2016]. Disponible en: https://es.wikipedia.org/wiki/Python

[17] SQL. Wikipedia, La Enciclopedia Libre. [Fecha de consulta: 05 de Junio de 2016]. Disponible en: https://es.wikipedia.org/wiki/SQL

[18] Anexo:Códigos de estado HTTP. Wikipedia, La Enciclopedia Libre. [Fecha de consulta: 05 de Junio de 2016]. Disponible en: https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP

[19] JSON. Wikipedia, La Enciclopedia Libre. [Fecha de consulta: 05 de Junio de 2016]. Disponible en: https://es.wikipedia.org/wiki/JSON

[20] HTML. Wikipedia, La Enciclopedia Libre. [Fecha de consulta: 05 de Junio de 2016]. Disponible en: https://es.wikipedia.org/wiki/HTML

[21] Somasundaram, R. (2013). Packt Publishing. Git: Version Control for Everyone.

[22] Linus Torvalds. Wikipedia, La Enciclopedia Libre. [Fecha de consulta: 28 de Julio de 2016]. Disponible en: https://es.wikipedia.org/wiki/Linus_Torvalds

[23] Git: About. Git. [Fecha de consulta: 28 de Julio de 2016]. Disponible en: https://git-scm.com/about/

[24] . Collins-Sussman, B., Fitzpatrick, B., Pilato, M. (2011). Version Control with Subversion. For Subversion 1.6.

[25] Foro (Internet). Wikipedia, la enciclopedia libre. [Fecha de consulta: 28 de Julio de 2016]. Disponible en: https://es.wikipedia.org/wiki/Foro_(Internet)

[26] Interfaz de programación de aplicaciones. Wikipedia, La Enciclopedia Libre. [Fecha de consulta: 28 de Julio de 2016]. Disponible en: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

[27] Estilo, color. Material design. [Fecha de consulta: 10 de Abril de 2017]. Disponible en: https://material.io/guidelines/style/color.html#color-color-tool

[28] Google Fonts. Google. [Fecha de consulta: 10 de Abril de 2017]. Disponible en: https://fonts.google.com/

[29] Material design. Wikipedia, la enciclopedia libre. [Fecha de consulta: 10 de Abril de 2017]. Disponible en: https://es.wikipedia.org/wiki/Material_design

[30] Ligature Kudakurage Symbols. Disponible en: http://kudakurage.com/ligature_symbols/

[31] Font Awesome. Disponible en: http://fontawesome.io/