ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"OPEN SOURCE SURVEY COMMUNITY"

INFORME DE PROYECTO INTEGRADOR

Previo a la obtención del título de:

INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

KEVIN ANDRÉS ORTIZ MERCHÁN

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTOS

Mis más sinceros agradecimientos a Dios, en primer lugar, por haberme dado el conocimiento y sabiduría para terminar con éxito este periodo en mi vida. En segundo lugar, agradezco a mis padres por su infinito apoyo emocional y económico en estos 5 años de estudio.

DEDICATORIA

A Dios por ser la fuente de mi conocimiento y fuerza durante mi trayectoria académica, a mis padres que fueron mi sustento emocional, económico en estos 5 años de estudio y al Msc Rafael Bonilla por ser un instrumento de Dios, para compartir sus conocimientos, y ser de guía para el desarrollo de este proyecto.

TRIBUNAL DE EVALUACIÓN

.....	.....
Nombre del Profesor	**Nombre del Profesor**
PROFESOR DE MATERIA INTEGRADOR	TUTOR ACADÉMICO

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me(nos) corresponde exclusivamente; y doy(damos) mi(nuestro) consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

Kevin Andrés Ortiz Merchán

RESUMEN

La elaboración de encuesta puede ser una tarea tediosa y su diseño puede presentar retos como la claridad de las preguntas, cantidad de preguntas, la descripción de la encuesta, grupo de usuarios al que va dirigido, etc. Problemas como los descritos ocasionan que los resultados no se consideren válidos, haciendo difícil llegar a una conclusión o tomar decisiones. Una solución fue desarrollar una aplicación web, que permite a los usuarios crear encuestas basados en preguntas que ya han sido validadas, discutidas y aceptadas por una comunidad de usuarios. Adicional a eso, los usuarios pueden compartir encuestas con otros usuarios de la plataforma para que puedan comentar o editar. Los resultados que se obtuvieron fue la aplicación web terminada y se desarrollaron pruebas de rendimiento sobre la plataforma, para un determinado grupo de usuarios.

Palabras clave: encuestas, preguntas, aplicación web

Abstract

Create surveys can be a tedious task and it's design can present challenges such as the clarity of the questions, the number of questions, the descriptions of the survey, the group of users to which it are addressed, etc. Problems such as those described cause the results are not considered valid, making it difficult to reach a conclusion or take decisions. One solution was develop a web application, which allows users to create surveys based on questions that have already been validated, discussed and accepted by a community of users. In addition, users can share surveys with other users of the platform so they can comment or edit. The results that were obtained was the web application finished and performance test were developed on the platform, for a certain group of users.

Keywords: survey, question, web application

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN.

1.1 Análisis del Problema.

Una encuesta es un instrumento de medición que permite evaluar a un grupo de personas sobre un determinado tópico, por ejemplo la alfabetización, disponibilidad de tiempo, rendimiento de diferentes grupos de personas que pertenecen a una institución, etc.

El diseño de una encuesta resulta ser una tarea complicada para personas que no trabajan en el área de la educación o en el campo de las ciencias sociales. Además, cuando se desarrolla una encuesta esta puede presentar errores de forma y de fondo. Errores de forma se producen por ejemplo cuando el contenido de las preguntas de una encuesta contiene palabras con más de un significado, se repite la misma palabras en una pregunta, etc. Errores de fondo incluyen el contexto en que se desarrolla la encuesta, al grupo de personas que va dirigida la encuesta, etc.

También se suma al problema que muchos investigadores frecuentemente escriben similares encuestas para resolver el mismo problema, presentando diferentes resultados, siendo muy difícil comparar resultados y llegar a una buena conclusión.

Los principales problemas que existen al diseñar una encuesta son los siguientes:

- Preguntas confusas o erróneas

- Preguntas demasiadas largas
- Errores de manejo de términos por parte del encuestador
- Errores de diseño de la encuesta

Preguntas confusas o erróneas.- Toda pregunta debe enfocarse en un punto determinado, no añadir palabras ambiguas o con un significado difícil de entender**.**

Preguntas demasiadas largas.- Las preguntas que son extensas son difíciles de entender, debido que existen palabras que se repiten en una pregunta o se usan sinónimos de una palabra en la pregunta.

Errores de manejo de términos por parte del encuestado.- En una encuesta es importante cuidar que los términos técnicos y el contenido general de la pregunta estén dirigidos al público que se desea encuestar .

Errores de diseño de la encuesta.- Las encuestas pueden presentar una cantidad grande de preguntas, haciendo que el encuestado pierda el interés en responder la encuesta, ya que consumiría demasiado tiempo.

El último punto importante es la pérdida de recursos y tiempo que tendría que invertir la persona para poder depurar la encuesta y reducir al mínimo la cantidad de errores.

Todos estos problemas antes mencionados presentan las siguientes consecuencias:

- Resultados erróneamente medidos.
- Falsos Positivos en el procesamiento de resultados
- Conclusiones erróneas

Una solución a considerar es tener un **panel de expertos** que retroalimentan a los creadores de encuestas sobre la elaboración de preguntas, uso de

términos o expresiones, o al conjunto de usuarios a quienes van dirigido la encuesta.

Una segunda solución **es compartir las encuestas** entre diferentes instituciones para que los usuarios puedan reutilizar encuestas que han sido corregidas, depuradas y probadas.

Existen estudios acerca de cómo diseñar correctamente encuestas y realizar entrevistas a otras personas. Estos estudios se presentan en la **sección 1.2 Revisión de literatura.**

1.2 Revisión de literatura.

Improving survey quality through pretesting

En este estudio [1] propuso cinco maneras diferentes que una persona debía seguir para mejorar la calidad de sus encuestas: entrevista cognitiva, entrevista informativa, observar el comportamiento del encuestado, panel de expertos y entrevista del entrevistador. Estas técnicas presentan una relación directa con los entrevistados o personas que respondieron la encuesta, permitiendo hacer el papel de guía durante todo el proceso y así poder recibir retroalimentación de los entrevistados de la calidad del diseño de la herramienta de evaluación.

Using departmental surveys to assess computing culture: quantifying gender differences in the classroom

En este estudio [2] elaboró un conjunto de encuestas que trataban de la cultura computacional sobre los diferentes departamentos en que trabaja tantos hombres y mujeres y elaboró encuestas que le permitió conocer qué

problemas sociales existen en los diferentes departamentos de computación. También encontró que los **términos o palabras** usados en el desarrollo de las encuestas son fundamentales para identificar los problemas que existían en los departamentos.

Study abroad survey instruments: A comparison of survey types and experiences

[3] desarrolló diferentes encuestas para evaluar las experiencias de los estudiantes de EE.UU. en el extranjero. El estudio se realizó en 20 universidades extranjeras utilizando diferentes tipos de encuestas: encuestas con opciones múltiples, preguntas abiertas, formato de respuesta de hoja de cálculo de opción múltiple, formulario escaneado y un sitio web de encuesta. La recopilación de los datos se hizo durante 19 años y basado en este estudio se pudo comprender los problemas o beneficios que pudieron obtener los estudiantes americanos en esas universidades.

Measures of Student Engagement in Computer Science

[4] reunió datos que se relacionan con la carrera de Ciencia de la Computación a través de encuestas que miden la comprensión y el agrado que tienen los diferentes estudiantes a esa carrera mostrando que la tasa de abandono es superior, debido que existen áreas de la ciencia de computación donde la comprensión es baja mostrando un nivel de preocupación y plantea medidas para poder resolver todos esos problemas.

Improving survey questions: Design and evaluation

[5] muestra que existen errores potenciales al desarrollar una encuesta, como son el uso de palabras, el contexto en que se desarrolla una encuesta, también muestra que la validez de una encuesta es dependiente del diseño de las preguntas. Este libro enseña a los investigadores a escribir buenas preguntas que están dirigidas sobre eventos objetivos y poder evaluar la calidad de las preguntas y el nivel de comprensión que ofrecen.

How unclear terms affect survey data

[6] expone los problemas del uso de términos claves que usan los investigadores al desarrollar una encuesta, mostrando márgenes de ambigüedad por lo que podría obtenerse resultados con un alto nivel de ruido, siendo completamente inútiles de usarlos para un estudio o comparación con otros resultados. La validación de esa hipótesis se realizó elaborando tres encuestas de salud a nivel nacional utilizando procedimientos “pretesting” y encontraron que más de un término usado en las preguntas que conformaban las encuestas estaban pobremente definidas con un alto grado de ambigüedad.

A Case for Open-Source Surveys (for Assessing Security Literacy)

[7] propone el desarrollo de una herramienta open source que permita unir a los diferentes usuarios que trabajen en la academia a compartir, corregir y validar las diferentes preguntas que elaboran otros usuario que pertenecen a la plataforma, con el objetivo de mejorar la calidad con que se desarrollan las

encuestas y poder evaluar los resultados de una manera efectiva con un pequeño margen de error.

1.3 Solución Propuesta

Estos estudios muestran lo fundamental que es diseñar una buena herramienta de evaluación ya que nos permiten identificar potenciales problemas que pueden existir en una institución, área o departamento, así como poder identificar grupos de usuarios que cumplen diferentes roles o características dentro de un entorno.

Todos estos estudios presentan una problemática en común que es el esfuerzo humano que debe hacer una persona o grupo de personas para eliminar términos ambiguos en las preguntas y encuestas, respuestas que ofrece el encuestado y despertar el interés en el encuestado al momento de responder la encuesta.

El enfoque de este estudio es desarrollar una herramienta Open Source para el diseño de encuestas donde las preguntas serán debatidas por una comunidad de personas en el área académica, ya sea investigadores o profesores, y aceptadas por miembros de un comité.

A continuación se muestran las contribuciones que hará este proyecto para la elaboración de encuestas.

- Elaborar una herramienta Web Open Source para la elaboración de preguntas que serán revisados por una comunidad de académicos ya sean investigadores o profesores y aceptadas por un comité.
- Permitir que el usuario pueda elaborar encuestas, escogiendo preguntas que ya han sido discutidas por la comunidad y aceptadas por miembros de un comité

- Permitir que el usuario puede añadir colaboradores que ayudarán a editar sus encuesta u opinar acerca de la calidad de la encuesta que creo el usuario principal.

CAPÍTULO 2

2. DESARROLLO DE LA SOLUCIÓN.

Como se mencionó en el **capítulo 1 sección 1.3**, nuestra solución consiste en desarrollar una aplicación Web llamada Open Source Survey que permite al usuario crear encuestas basadas en un conjunto de preguntas que han sido discutidas por la comunidad y validadas por un comité antes de ser añadidas a un repositorio de preguntas que maneja la aplicación.

2.1 Antecedentes

A continuación se presenta el enfoque de algunos proyectos colaborativos que existen en la actualidad, como son: Wikipedia, Stack Overflow, Open Source Course Ware y Open Source Textbooks.

Wikipedia.- Es un proyecto Open Source que permite construir una enciclopedia de contenido libre, donde permite que una comunidad pueda compartir el conocimiento que poseen acerca de algo. Comenzando el 20 de mayo del 2001, actualmente poseen **1370970 artículos**.

Stack Overflow.- Un sitio web desarrollado por Jeff Atwood, utilizado por una comunidad de personas que trabajan en el área de Ciencias de la Computación. La plataforma permite a los usuarios presentar sus dudas o preguntas acerca de una herramienta o tópico (**sistemas distribuidos, comunicación por paso de mensajes, etc.**) y otros usuarios responden aquellas dudas, todos los usuarios pueden votar por la pregunta y por sus respuestas, cuando un usuario vota por una pregunta la puede calificar **como**

relevante o menos relevante, sin embargo cuando se vota por respuestas **estas pueden ser acertadas o menos acertadas.**

Open Source Course Ware .- Son publicaciones docentes de contenidos abiertos como son libros o artículos, permitiendo la reutilización libre y respetando la cita del autor original. Los materiales pueden corresponder a asignaturas de educación superior universitaria, tanto de grado como postgrado. El objetivo de este proyecto es potenciar la sociedad del conocimiento y fomentar proyectos entre instituciones y docentes relacionados con los contenidos abiertos.

Open Source TextBooks.- Son un conjuntos de libros que tienen licencia OCL (Open Copyright Licence), y significa que pueden ser distribuidos de manera gratuita y usados por estudiantes, profesores y público en general, lo que permite sus descargas sin ningún costo.

Todos estos proyectos presentan un enfoque colaborativo que permiten que los usuarios puedan compartir sus conocimientos con otros usuarios, y también que puedan debatir y resolver dudas acerca de un tópico en particular, con la finalidad que se desarrolle una comunidad con libre acceso al conocimiento.

2.2 Beneficios en el uso de la aplicación Open Source Survey

Open Source Survey que es el nombre del instrumento que se desarrolla en este estudio sigue el mismo enfoque de los proyectos que se mencionaron en la sección 2.1**,** y los beneficios que se espera lograr son los siguientes.

- Intercambio abierto
- Participación colaborativa
- Meritocracia
- Rápido Prototipado
- Desarrollo de la comunidad

A continuación se describen los beneficios en desarrollar una aplicación Open Source Survey.

- **Intercambio abierto**.- Todo el contenido de las encuestas y preguntas que se elaboren estarán disponibles para todos los usuarios registrados en la plataforma.
- **Participación Colaborativa**.- Los usuarios pueden crear discusiones y comentarios acerca del contenido que generen otros usuarios dentro de la plataforma; retroalimentando, corrigiendo y presentando sus dudas sobre la calidad del contenido (preguntas o encuestas) que generan.
- **Meritocracia**.- Existen un conjunto de usuarios que han ganado méritos en la plataforma y son seleccionados para que formen parte de un comité, permitiéndoles aprobar o rechazar los cambios hechos sobre una pregunta, después de una revisión cuidadosa y discutida por parte de la comunidad.
- **Prototipado Rápido**.- Facilita la rápida creación de encuestas a partir de preguntas discutidas y aprobadas por la comunidad y miembros del comité.

- **Desarrollo de la comunidad**.- Permite reunir usuarios con similares intereses y metas, permitiendo que trabajen en conjunto, con el fin de

facilitar contenido para la creación de encuestas, permitiendo comparar resultados y tomar decisiones.

2.3 Descripción de los módulos que se desarrollaran en la aplicación.

2.3.1 Actores

Comenzamos mencionando a los actores de la aplicación que son:

Usuario.- Tiene el rol de poder crear, editar y eliminar preguntas y encuestas. También pueden crear discusiones de preguntas y encuestas, observar los resultados de sus encuestas y realizar comentarios sobre preguntas, encuestas o discusiones de otros usuarios**.**

Colaboradores.- Tienen el rol de poder ver y comentar las encuestas que otros usuarios han compartido, también tienen el rol de poder editar las encuestas de otros usuarios.

Miembros del comité.- Tienen el rol de poder aceptar o rechazar las diferentes preguntas creadas por otros usuarios, también pueden aceptar o rechazar las discusiones que crean otros usuarios acerca de preguntas que han sido validadas por la comunidad y aceptadas por los miembros del comité.

2.3.2 Módulos

En la ***Figura 2.1 Diagrama de Casos de Uso de Preguntas*** podemos observar el diagrama de casos de uso con respecto a la sección de preguntas y a continuación se presenta la descripción de cada módulo que corresponde a dicha sección.

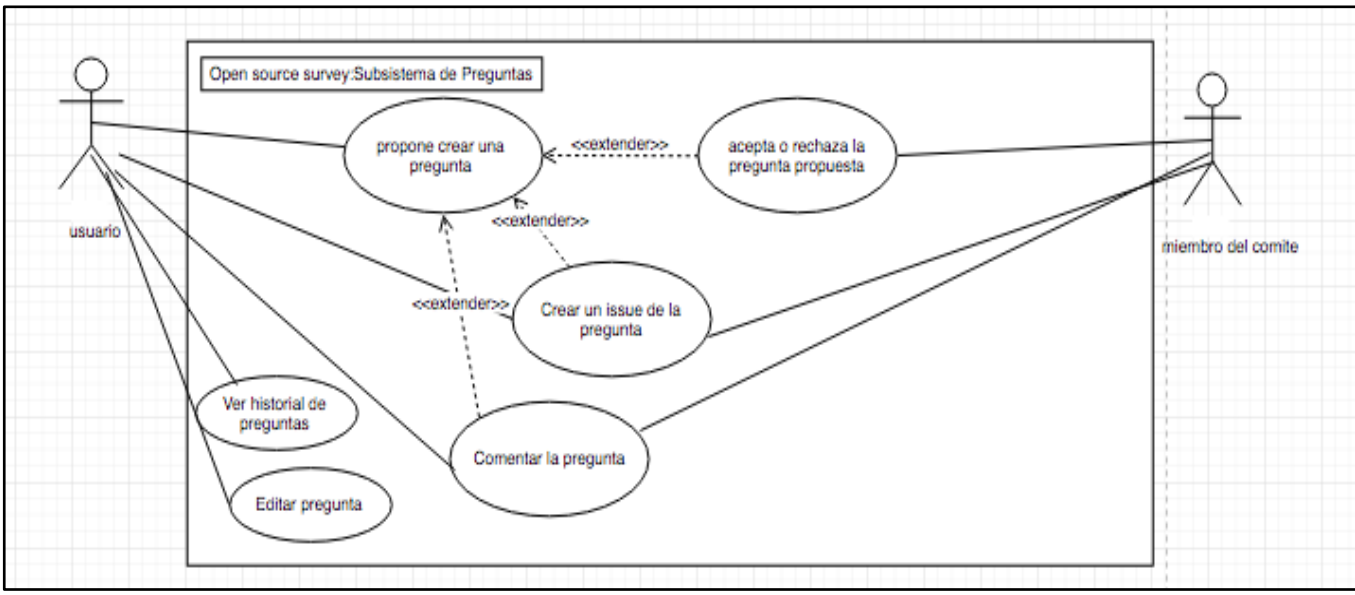

Figura 2.1 Diagrama de Casos de Uso de Preguntas

Módulo de autenticación

Un usuario deberá autenticarse en la aplicación ya sea usando su cuenta de Google, Twitter y Github.

Módulo de Preguntas

Este módulo presenta los siguientes submódulos como:

Crear una pregunta.- Un usuario puede crear una pregunta, que será enviada a los miembros del comité quienes aceptan, rechazan o definen cambios que se deban hacer a la pregunta.

Editar una pregunta.- Un usuario puede editar el contenido de una pregunta que ha creado.

Eliminar una pregunta.- Un usuario puede eliminar una pregunta que ha creado y se encuentra almacenada en la base de datos.

Proponer corrección a una pregunta.- Un usuario puede proponer correcciones sobre una pregunta aceptada por la comunidad, incluyendo los

cambios que debe efectuarse sobre la misma. La propuesta será aceptada o rechazada por los miembros del comité.

Comentar una pregunta.- Un usuario puede comentar sobre una pregunta en particular, creada por el mismo o por otro usuario.

Comentar sobre una propuesta de corrección de una pregunta,- Un usuario puede comentar sobre una propuesta de corrección planteada para una pregunta.

Historial de preguntas.- Un usuario puede ver la lista de preguntas que ha creado durante a lo largo del tiempo.

Módulo de Encuestas y Resultados

En la ***Figura 2.2 Diagrama de Casos de Uso de Encuestas*** podemos observar el diagrama de casos de uso con respecto a la sección de encuesta y a continuación se presenta la descripción de cada módulo que corresponde a dicha sección.

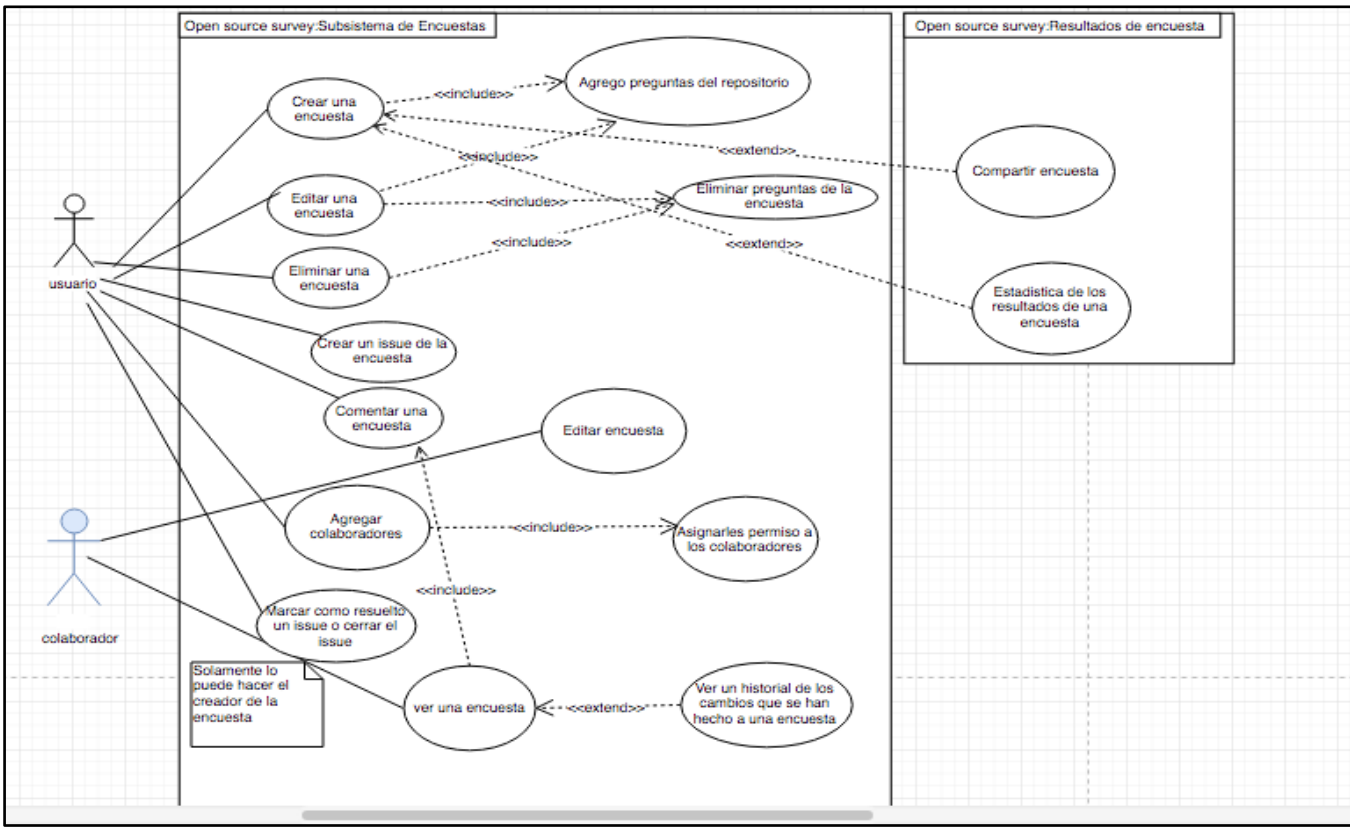

Figura 2.2 Diagrama de Casos de Uso de Encuestas

Crear encuesta.- Un usuario puede crear una encuesta, tomando diferentes preguntas que han desarrollados otros usuarios.

Editar una encuesta.- Un usuario puede editar la descripción de la encuesta y cambiar las preguntas que se encuentran en la encuesta.

Eliminar una encuesta.- Un usuario puede eliminar una encuesta que ha creado en la base de datos.

Proponer cambios a una encuesta.- Un usuario puede proponer cambios a una encuesta de un usuario, con la finalidad que la comunidad pueda comentar, discutir y retroalimentar al creador de la encuesta.

Comentar una encuesta.- Un usuario puede comentar una encuesta de otro usuarios.

Comentar cambio propuesto a una encuesta.- Un usuario puede comentar sobre los cambios propuestos a una determinada encuesta.

Añadir colaboradores.- Un usuario puede añadir colaboradores a una encuesta asignándoles permisos de lectura o edición.

Aceptar o rechazar cambios propuestos a una encuesta.- El creador de la encuesta puede aceptar o rechazar los cambios propuestos sobre una encuesta.

Ver historial de cambios sobre una encuesta.- El creador de una encuesta puede ver el historial de cambios de una encuesta a lo largo del tiempo.

Compartir encuesta.- Un usuario puede compartir una encuesta que ha creado con usuarios externos de la aplicación, con la finalidad que puedan responder la encuesta.

Ver resultado de una encuesta.- Un usuario puede ver los resultados de un encuesta determinada.

2.4.- Estados de una pregunta

En la ***Figura 2.3 Diagrama de Estados sobre una Pregunta***, se muestra los diferentes estados que puede tener una pregunta. Comienza cuando un usuario crea una nueva pregunta, automáticamente la pregunta pasa a un estado de revisión, y los miembros del comité deciden si la pregunta propuesta necesita ser cambiada o es aceptada. Si la pregunta pasa a estado de revisión, el creador de la pregunta puede realizar los cambios respectivos y automáticamente se notificará a los miembros del comité de los cambios hechos a la pregunta, los miembros del comité revisan la pregunta y deciden

si la pregunta debe pasar a un estado aceptada o rechazada. El estado rechazado indica que la pregunta no será usada para la creación de encuestas. Si la pregunta pasa a un estado aceptada, la pregunta se enviará automáticamente al repositorio central en Github donde otros usuarios pueden usar la pregunta para la elaboración de sus encuestas.

Una pregunta puede pasar de un estado aceptada a un estado de revisión si un usuario propone un cambio de dicha pregunta. Los miembros del comité son los encargados de validar si los cambios propuestos de la pregunta se deben aceptar o rechazar. Si los cambios propuestos sobre una pregunta es aprobada, la pregunta pasa a un estado de revisión donde se notifica al creador de la pregunta de los cambios que debe efectuar y nuevamente la pregunta pasa a estado de aceptada.

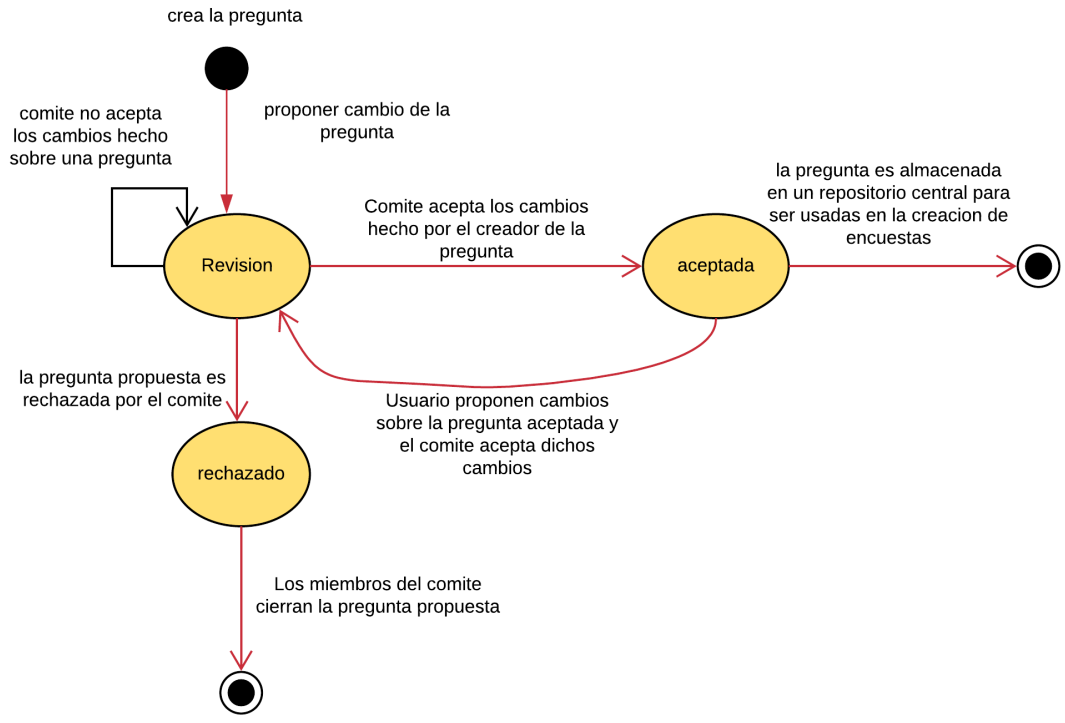

Figura 2.3 Diagrama de Estados sobre una Pregunta

En la ***Figura 2.4 Diagrama de Flujo para la creación de cambios de una pregunta***, se muestra el conjunto de pasos que se ejecutan al momento de proponer cambios a una pregunta.

Un usuario propone cambios para una pregunta, los cambios son asignados al creador de la pregunta, dichos cambios propuestos son discutidos por la comunidad. Los comentarios o retroalimentaciones ayudan al creador de la pregunta poder mejorar el contenido o estructura de la pregunta, si la comunidad considera que la pregunta ya ha sido corregida, esta pasa al comité para que sea aprobada o rechazada; caso contrario se seguirán discutiendo las correcciones. Si los miembros del comité consideran que los cambios hechos sobre una pregunta son válidos, aceptan la pregunta propuesta y la pregunta es subida a un repositorio central en GitHub, caso contrario la pregunta es rechazada y la discusión se cierra.

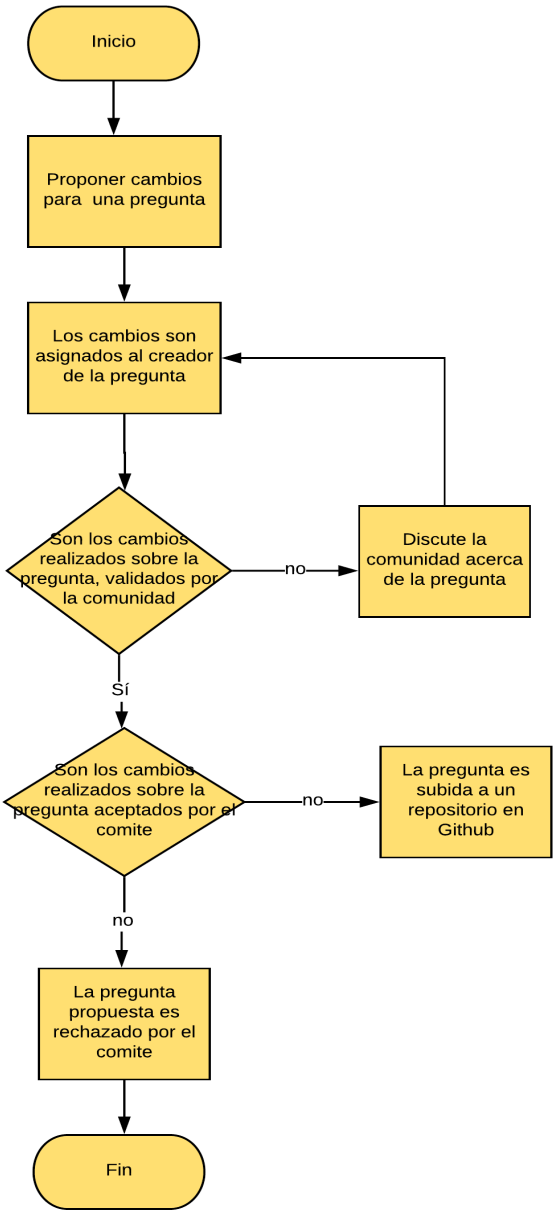

Figura 2.4 Diagrama de Flujo para la creación de cambios de una pregunta

2.6 Arquitectura de la aplicación

La ***Figura 2.5 Arquitectura de la aplicación***, muestra los módulos que conforman la aplicación web, y en las siguientes subsecciones se explica en detalle la descripción de cada módulo y la manera como interactúan.

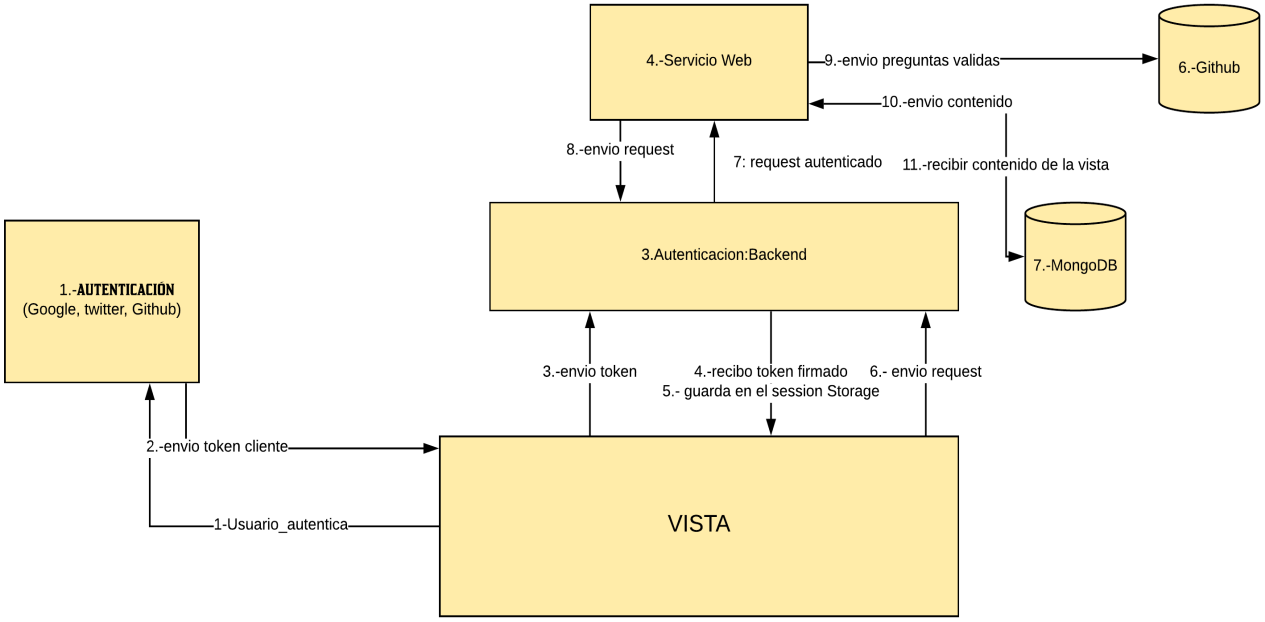

Figura 2.5 Arquitectura de la aplicación

2.6.1 Descripción de cada módulo de la aplicación web

Existen tres componentes principales de la aplicación web:

- Vista de la aplicación
- Servicio de autenticación
- Servicio de Backend

Cliente o vista de la aplicación

En el lado del cliente se desarrollará las vistas con que el usuario final interactúa. La herramienta escogida para el desarrollo de las vistas es **Angular.**

Elección de la herramienta a utilizar en el desarrollo de la vista de una aplicación web.

Los principales candidatos a utilizar en el desarrollo de la vista fueron los siguientes:

- REACT.JS
- EMBER
- iNCREMENTALDOM
- VIRTUALDOM
- EMBER2
- ANGULAR 1
- ANGULAR 2
- CITOJS

Se consideró tres aspectos para la elección del framework: el tiempo de carga de cada componente de la aplicación web, el manejo del DOM y el conjunto de herramientas que ofrece para desarrollar una aplicación web.
En la ***Figura 2.6 Comparativa en el manejo del DOM*** podemos observar los rendimientos de cada framework en el desarrollo de una aplicación web, este estudio (**https://auth0.com/blog/more-benchmarks-virtual-dom-vs-angular-12-vs-mithril-js-vs-the-rest/)** lo realizó la empresa Auth0 que media el rendimiento en los diferentes aspectos que maneja un framework como son:

- **Manejo del DOM.-**En la ***Figura 2.6 Comparativa en el manejo del DOM*** podemos observar el rendimiento de los diferentes frameworks al cargar todos los elementos de una aplicación web en el DOM. El tiempo se

mide en microsegundos y podemos observar que angular 2 posee un mejor rendimiento en comparación a angular 1.x.

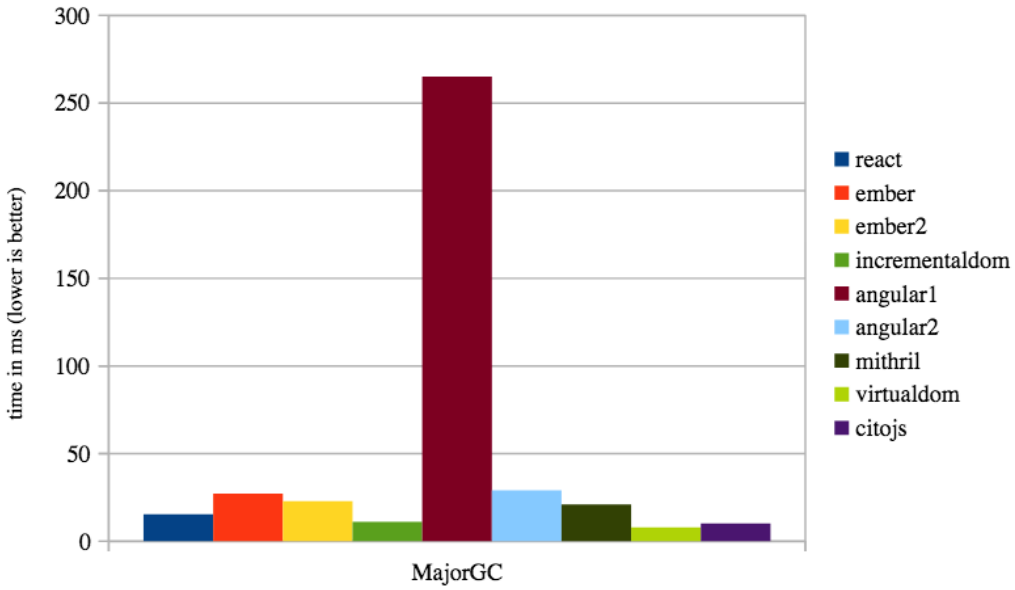

Figura 2.6 Comparativa en el manejo del DOM

- **Recarga de todos los elementos en el DOM**.-En la ***Figura 2.7 Tiempo que se demora el navegador en actualizar los cambios en el DOM***, podemos observar el tiempo que se demora en recargar todos los elementos de una página en el DOM. REACTJS, VIRTUALDOM, EMBER1, EMBER2, ANGULAR2 son los framework que presentan un mejor rendimiento a la recreación de elementos en el árbol del DOM, siendo los tiempos de carga menores a 50 ms.

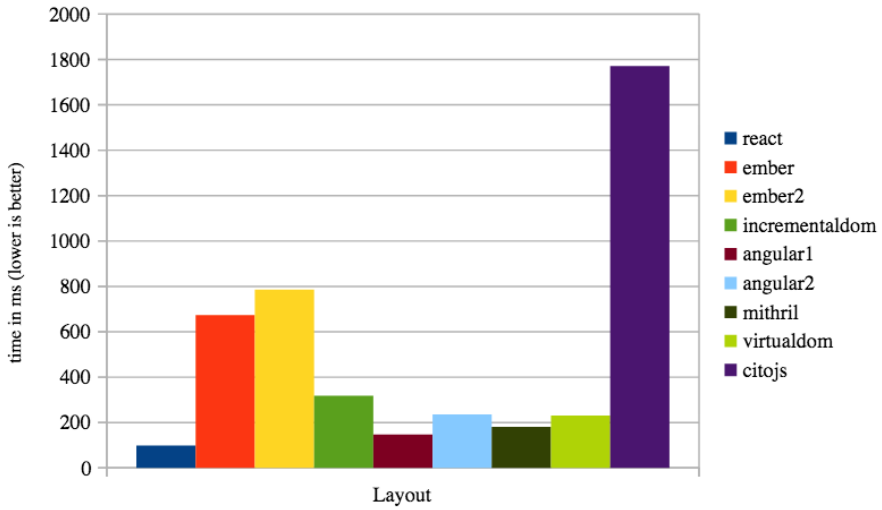

Figura 2.7 Tiempo que se demora el navegador en actualizar los cambios en el DOM

Tiempo de ejecución de código JavaScript.- En la ***Figura 2.8 Tiempo que se demora el navegador en ejecutar código JavaScript***, podemos observar que REACTJS, VIRTUALDOM, incremental DOM, ANGULAR2, ejecutan de manera rápida el código JavaScript. También hay que enfatizar que angular 2 trabaja sobre Typescript, una capa superior de JavaScript, por lo cual en una aplicación Angular 2, todo el código desarrollado con Typescript se compila a JavaScript antes de su ejecución.

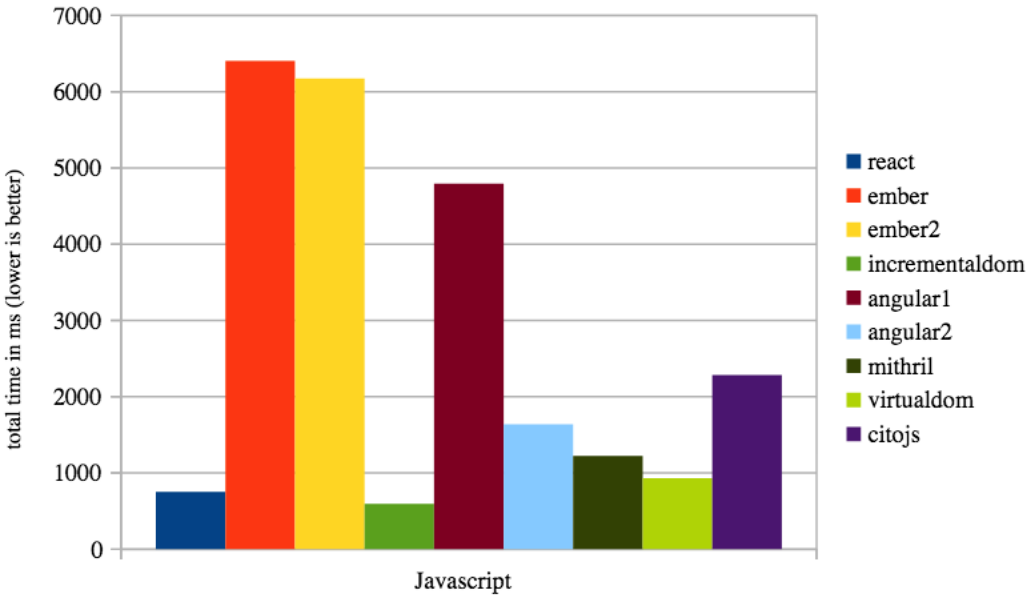

Figura 2.8 Tiempo que se demora el navegador en ejecutar código JavaScript

Características de los frameworks de estudio

A continuación se mostrará la diferencia desde el punto de vista de desarrollo de cada una de las herramientas nombras al inicio de esta subsección.

Angular 2.- Framework para aplicaciones web de código abierto, desarrollado y mantenido por Google que se utiliza para crear y mantener aplicaciones web.

Características del uso de Angular 2 para el desarrollo de vistas en una aplicación web.

Velocidad y rendimiento

Todas las características que se mencionarán fueron tomadas de la pagina oficial de Angular (https://angular.io/features**) desarrollada por la Empresa Google**

Generación de código.- Convierte las plantillas en código altamente optimizado para las máquinas virtuales donde se ejecutan JavaScript**.**

Universal.- Son independientes el manejo de las vistas, lógica y Backend, permitiendo trabajar con diferentes tecnologías como son nodejs, PHP, etc.

División del código .- Las aplicaciones de angular se cargan rápidamente gracias al nuevo enrutador de componentes, ofreciendo una división automática de código para que el navegador solo cargue el código necesario para procesar la vista solicitada.

Productividad

Plantillas.- Permite crear interfaces de manera rápida simples y potentes a través de las directivas que ofrece para el manejo del DOM de la aplicación.

Angular-cli.- Herramienta de líneas de comando que permite añadir componentes, realizar pruebas, así como pre visualizar de forma instantánea la aplicación.

React.js.- Es una librería JavaScript que permite construir interfaces de usuarios

Servicio de autenticación

Se usará los servicios de Google, Twitter y Github para que los usuarios puedan autenticarse en la aplicación web. A continuación se describe las características de cada servicio.

Firebase auth (Google).- Proporciona un conjunto de tecnologías como servicios de backend, SDK, bibliotecas UI ya desarrolladas para que los usuarios se puedan autenticar. Admite autenticación con contraseñas con proveedores de entidades federales como, Google, Facebook y Twitter.

Como podemos observar en la ***Figura 2.9 Servicios que ofrece Firebase***, el costo por usar el servicio de autenticación de Google es gratuito.

Figura 2.9 Servicios que ofrece Firebase

Autenticación para usar el servicio web de la aplicación.- La herramienta que se usará para manejar la capa de seguridad entre la comunicación del servicio web con la vista es **JWT.**

JWT (JSON WEB TOKEN).- Es un estándar abierto (RFC 7519) que define una forma compacta y autónoma para transmitir de forma segura información entre las partes (cliente y servidor) como un objeto JSON, dicha información puede ser verificada porque está firmada digitalmente. Los JWT se pueden firmar usando un secreto que el desarrollador definió al momento de establecer el canal de autenticación entre el cliente y servidor (usando el algoritmo HMAC) o usando claves públicas y privadas RSA.

Servicio Web.- Se desarrolla un servicio web que interactúa con la base de datos MONGODB y el repositorio central de GITHUB. A continuación se muestra las características que presenta MONGODB con respecto a otros gestores de base de datos, también se mostrará en la **Error! Reference source not found.** las características que ofrece GITHUB como sistema de

versionamiento en comparación a otras herramientas de control de versión como GITLAB, BITBUCKET.

MongoDB.- Es una base de datos ágil que permite diseñar esquemas dinámicos que pueden cambiar con la evolución de la aplicación, proporcionando las funcionalidades estándares tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.

Características

- Es una tecnología Open Source
- Gestor de Base de Datos No SQL orientado a documentos
- Flexibilidad en agregar datos a una determinada colección.
- Posee un esquema dinámico que permite crear colecciones sin definir ninguna estructura
- Escrito en su totalidad en C++
- Escritura de documentos atómicos y lecturas consistentes.
- Permite embeber subdocumentos dentro de otros documentos
- MongoDB trata los documentos como si estuvieran en memoria
- Sharding es un método utilizado para almacenar documentos a través de diferentes computadoras

Principales diferencias entre plataformas de control de versionamiento

GitHub	GitLab	BitBucket
Rest Api	Rest Api	No posee Rest Api
Pull Request, gestor incidencias, One click Branch	Permite instalarlo de forma local en una máquina	Notificación HipChat

Extensión con otras plataformas de desarrollo	Protecciones de las ramificaciones con diferentes niveles de autorización	Repositorios privados de hasta 5 usuarios en su versión gratuita
Uso gratuito para repositorios públicos	Sistemas de push notification para notificar a usuarios acerca de incidencias creadas de un proyecto	Permitir instalar aplicaciones de terceros
Repositorios privados con modalidades de pago	Permite controlar el flujo de trabajo general de un proyecto	Pull Request, gestor de incidencias, One click Branch
maneja 5000 request por hora	Consume 4GB de RAM en la máquina donde se está ejecutando	

Tabla 2.1 Diferencias entre las plataformas de versionamiento

2.7. Interacción de cada módulo de la aplicación web

A continuación se explica en detalle la comunicación de cada módulo que se puede observar en la ***Figura 2.5 Arquitectura de la aplicación***, que describe la arquitectura de la aplicación web.

Módulo Autenticación con el servidor.- El usuario podrá registrarse en la aplicación usando cualquiera de los tres servicios que se les ofrece como son google, twitter, Github. Todos esos tres servicios son manejados por Firebase, que se encarga de manera la capa de seguridad entre los servicios federados (Github, Twitter, Google) y si el usuario es correctamente autenticado envía un token indicando la sesión que ha abierto usando dicho servicio. Ese token junto con la información adicional obtenida como son nombre, apellido, correo, es enviada al servidor de la aplicación, que se encarga de validar si determinado usuario ya existe en la aplicación web, o es un nuevo usuario. Ya sea que el usuario es nuevo o se encuentre registrado en el servidor, crea una firma con los datos que se envían al servidor como son nombre, correo y los firma

usando la opción HS256 (HMAC con SHA-256). Con esta firma, el usuario puede realizar todo tipo de acciones en la plataforma.

Si el token de autenticación de la plataforma web ha expirado automáticamente será redirigido a la pagina de login.

En la ***Figura 2.10 Diagrama de secuencia de autenticación en la aplicación web***, podemos visualizar el conjunto de pasos que se llevan a cabo para que un usuario se autentique en la plataforma

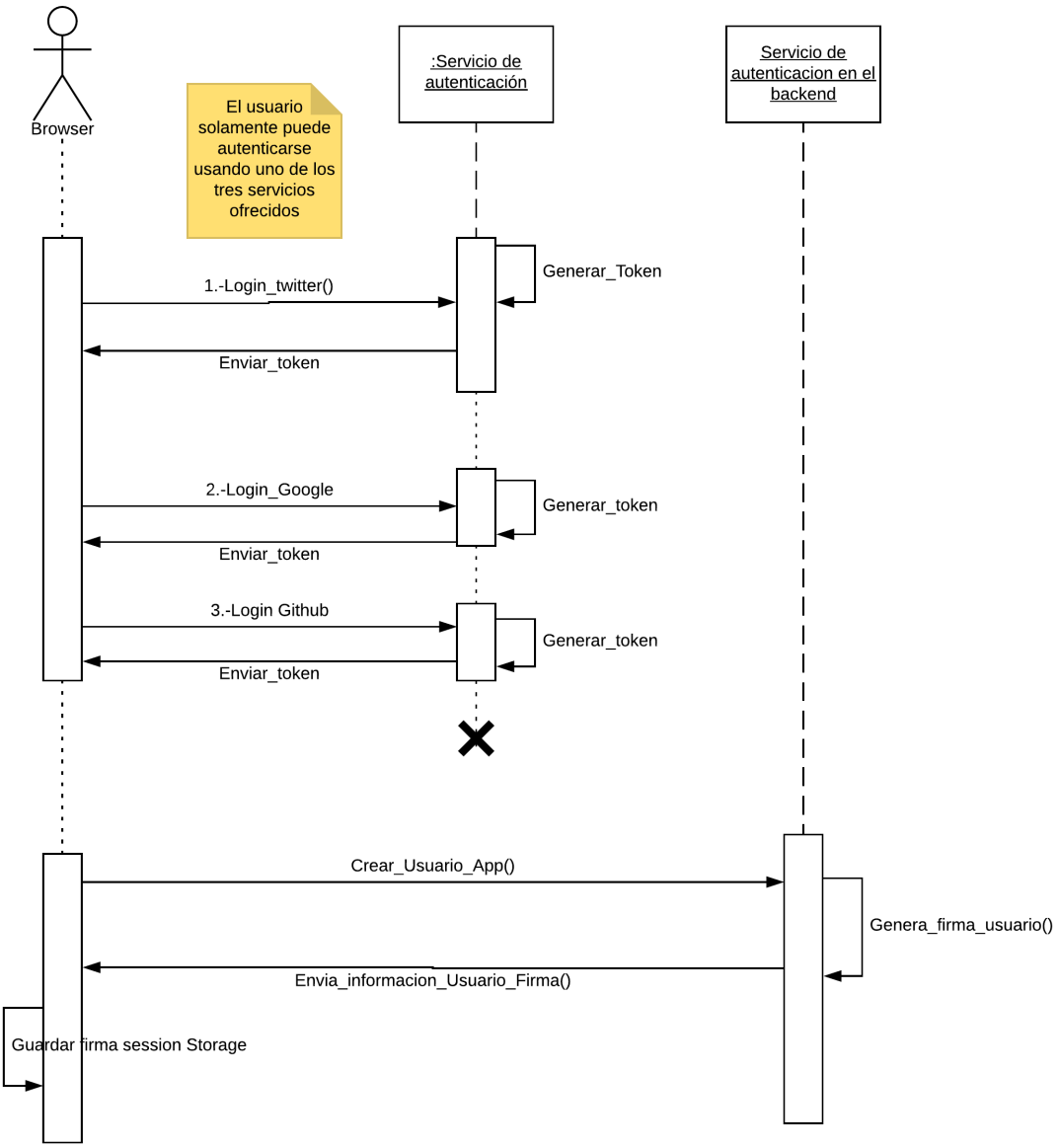

Figura 2.10 Diagrama de secuencia de autenticación en la aplicación web

Módulo de Creación de Pregunta.- Un usuario ya autenticado en la aplicación, puede crear preguntas, eliminar preguntas, crear encuestas, proponer corrección de encuestas, etc. Uno de los escenarios que vamos a mostrar en la ***Figura 2.11 Diagrama de secuencia de creación de una pregunta***,

es la creación de preguntas. Cuando un usuario envía a guardar una pregunta, se genera automáticamente una discusión indicando el estado de la pregunta, y dicha información se guarda en la base de datos

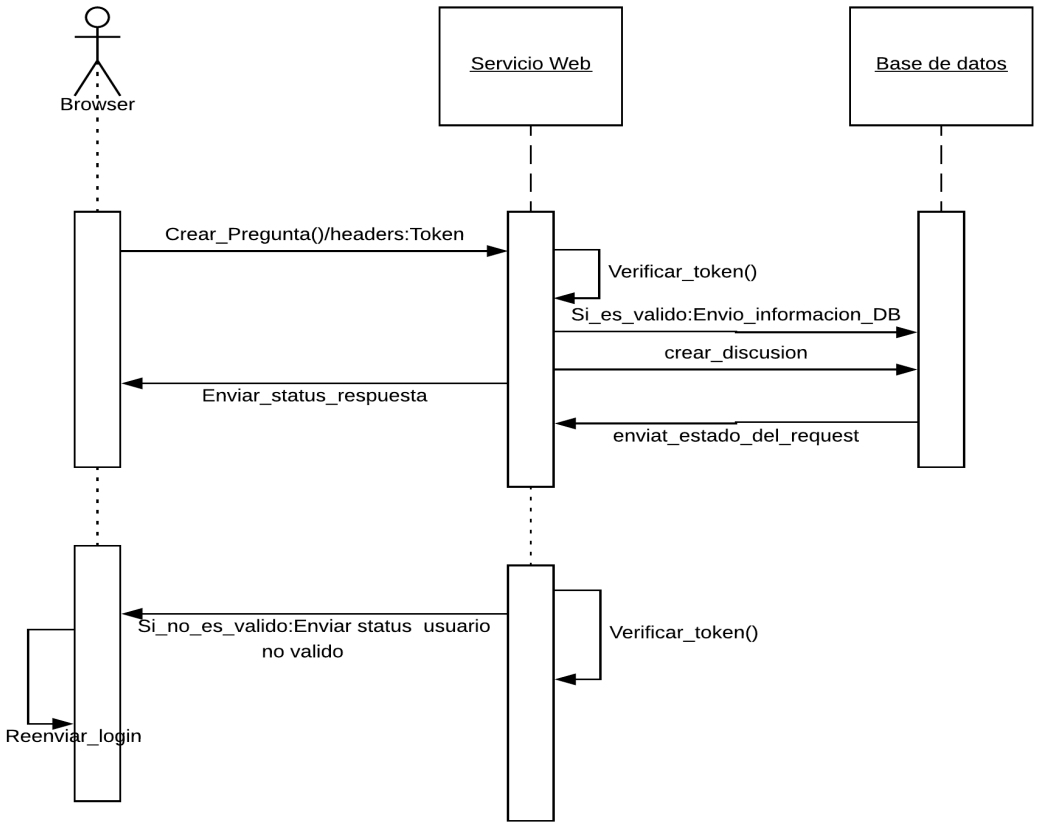

Figura 2.11 Diagrama de secuencia de creación de una pregunta

Módulo de Validación de Pregunta.- Los miembros del comité tienen la autoridad de validar si una pregunta pasa a un estado de rechazada o aceptada. Si un miembro del comité válida una pregunta, el estado de dicha pregunta se actualiza en la base de datos y se guarda automáticamente en el repositorio central. La operación de envío al repositorio se realiza de forma asíncrona, para no disminuir el rendimiento del servidor, todos estos pasos

descritos se los puede visualizar en la ***Figura 2.12 Diagrama de Secuencia de validar Pregunta***

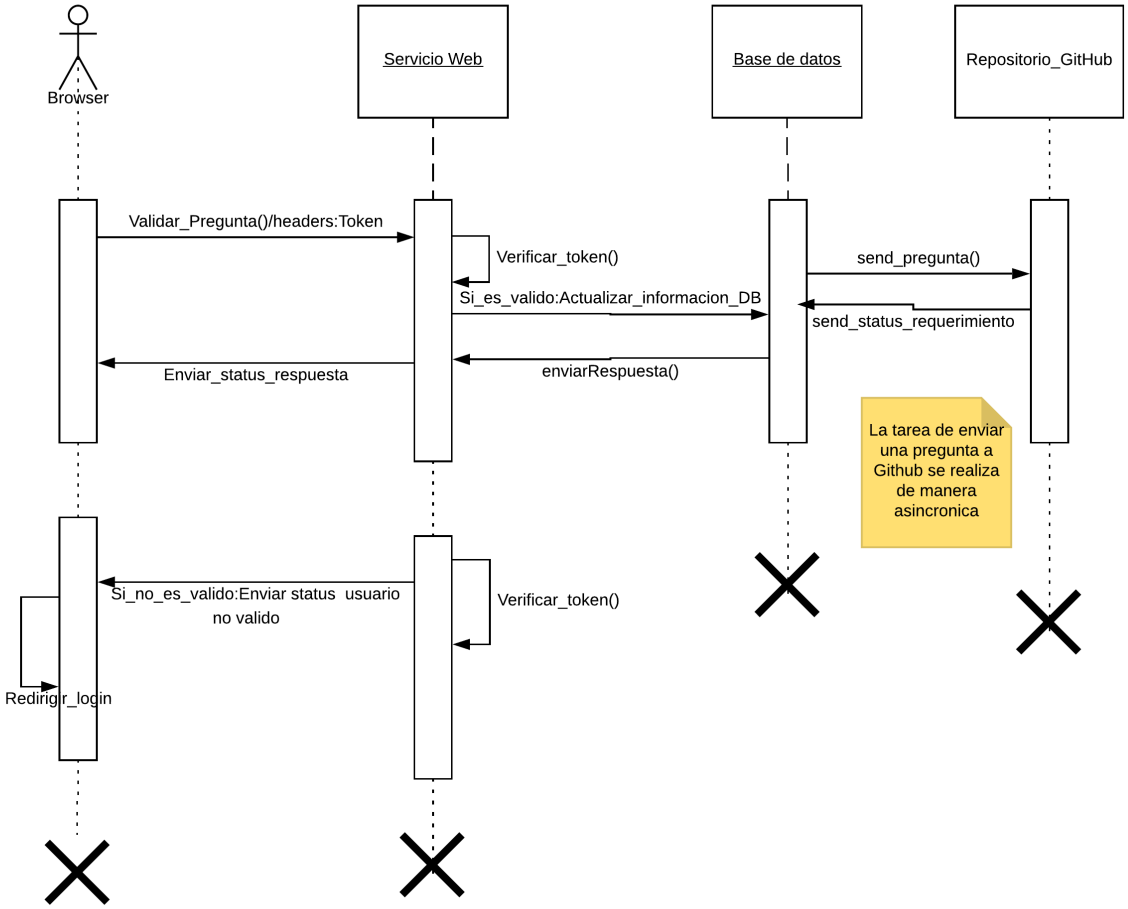

Figura 2.12 Diagrama de Secuencia de validar Pregunta

CAPÍTULO 3

3. IMPLEMENTACIÓN DEL SISTEMA.

3.1 Patrones de diseño y paradigmas de desarrollo de software usados en el desarrollo de la aplicación.

Patrón de diseño MVVM .-Se usó el patrón MVVM (modelo - vista—vista-modelo) en la vista de la aplicación para manejar y procesar los datos que traemos de un servicio Rest.

Se decidió usar este patrón (MVVM) a diferencias de otros patrones de diseño por las siguientes motivaciones

- Una separación entre la lógica de la aplicación, la interfaz gráfica y el manejo de los datos hace fácil mantener, probar y evolucionar una aplicación a largo plazo
- Los modelos de vista actúan como adaptadores para las clases que realizan la manipulación de los datos que vienen del servidor**.**

Programación Funcional Reactiva.- La programación basada en eventos se encarga de manejar los datos de un programa entorno a un flujo de información donde interactúa la vista y el servidor. Permitiendo suscribirse a eventos que actualizarán la vista dependiendo de los nuevos datos que lleguen desde el servidor.

Una de las ventajas de usar un paradigma basado en eventos, es que la aplicación maneja distintos estados y estos se subscriben a eventos que controlan la información que envía el servidor. El DOM se actualiza automáticamente cada vez que un evento recibe nuevos datos.

La actualización de cada componente (porciones de HTML de la aplicación web) se realiza de manera asíncrona mediante la inyección de servicios. Cada servicio maneja el flujo de información (stream) que se recolectan desde el servidor.

Uso de Redis en el manejo de notificaciones en tiempo Real

Se usó Redis para manejar el estado de la conexión de cada cliente con la aplicación web.

El procedimiento que se llevó a cabo para lograr esta tarea fue el siguiente.

- Almacenar el objeto Socket, que es un identificador que el sistema le otorga a cada usuario cuando se conecta y se autentica en la aplicación web.
- Cada identificador se almacena en una lista en memoria usando Redis como gestor de Base de datos.

Un enfoque tradicional es haber definido una lista en memoria e ir almacenando cada identificador y cuando se requiere enviar un evento de notificación a un usuario determinado, recorrer la lista de forma secuencial hasta encontrar el elemento y si dicho elemento no se encuentra realizar una acción distinta como mostrar un mensaje de alerta y notificar al usuario acerca de ese evento, esto presenta los siguientes problemas.

- Sobrecargar el hilo principal del programa donde está ejecutándose el servidor web, disminuyendo el rendimiento de la aplicación
- El tiempo de búsqueda de un usuario sobre una lista de usuarios tomaría O(n), a este tiempo se le suma el número de veces que un usuario se conecta y se desconecta de la aplicación o se decide realizar una notificación.
  En total debería realizarse dos búsquedas, una búsqueda para saber si el ítem existe en la lista al momento de realizar una notificación, otra búsqueda para eliminar un usuario de la lista cuando un usuario se desconecta de la aplicación, en total obtendremos un tiempo computacional de $O(n^2)$.

En la **Error! Reference source not found.**, podemos observar el rendimiento de Redis en las operaciones SET y GET, mostrando que el tiempo que se toma para almacenar 100000 objetos en memoria fueron 0.88 segundos, en comparación con el tiempo de obtener 100000 objeto en memoria que es de 1.23 segundos.

Operacion SET	Operación GET
100000 request completados en 0.88 sec	100000 request completados en 1.23 sec
se puede mantener 50 clientes en paralelo	se puede mantener 50 clientes en paralelo
53648.38 request per second	45497.73 request per second

Tabla 3.2 Resultados Benchmarks Redis

Las operaciones usadas en las notificaciones en tiempo real fueron SET y GET.

La operación **SET** permite almacenar en una tupla la información que se obtiene de un usuario al conectarse a la aplicación, esa información es un identificador que posee cada usuario, junto con el objeto **socket.**

La operación **GET** permite recuperar el valor almacenado en la tupla a través de la clave que se consulta en Redis.

3.2 Módulos en que se usaron cada uno de los paradigmas y patrones de diseño explicados en el punto anterior

En la ***Figura 3.13 Módulos de la aplicación web***, se muestra los módulos que conforman la aplicación web y la comunicación con cada módulo, cada módulo tiene

inyectado un servicio donde permite manejar, manipular junto con la vista la información que viaja desde un servicio web.

Todos estos eventos se manejan de manera asincrónica a través del patrón publish-subscribe, donde cada módulo se encuentra suscrito a distintos eventos que notifican cuando existan nuevos datos que envía el servidor web. Esta implementación muestra las siguientes ventajas:

- No existen llamadas bloqueantes en el cual los demás procesos en cola deban esperar que el proceso principal que se encuentra al inicio de la cola termine de realizar la tarea que estaba ejecutando.
- Permite que un módulo pueda realizar hasta tres llamadas de forma paralela al servicio web, sin necesidad de que ninguno de los tres procesos se bloquee.

Un punto adicional a mencionar es que el servidor web solo envía información tipo JSON, lo que permite aligerar la carga (**overhead**) del servidor al evitar que recupere todo el contenido HTML junto con la información extraída de la base de datos.

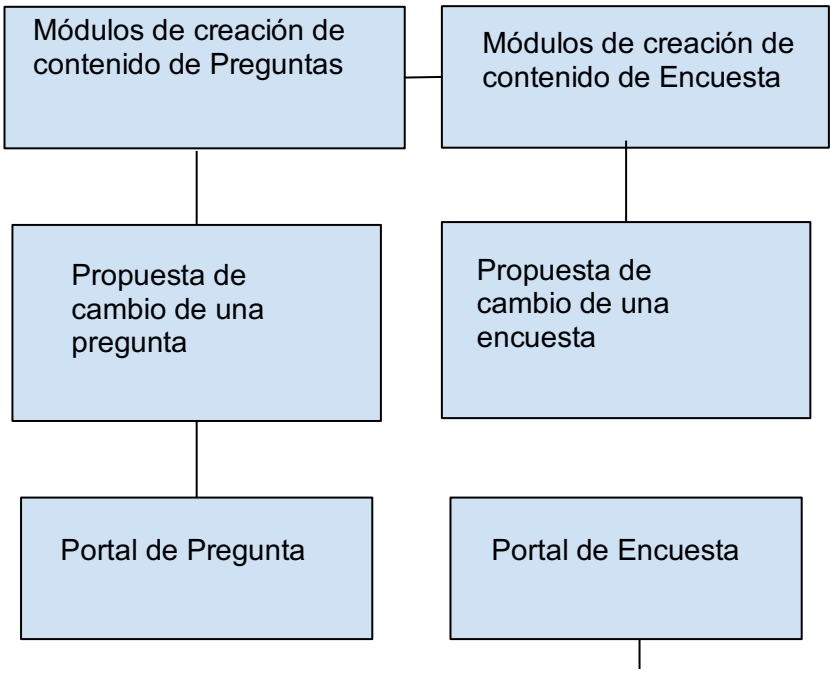

Figura 3.13 Módulos de la aplicación web

3.3 Problemas que se presentaron durante el desarrollo de la aplicación

Manejo de datos asíncronos.- Debido a que los procesos no son bloqueantes , se anidaron callbacks para poder manipular toda la información que se recibe del servidor web .

Prototipo de la aplicación

En esta sección se mostrará el prototipo de la aplicación, indicando el proceso que se lleva a cabo para crear una encuesta.

Proceso de creación de encuesta.- En la ***Figura 3.14 Autenticación en la plataforma*****,** se puede observar que un usuario puede autenticarse en la plataforma usando cualquiera de los tres servicios respectivos como son Google, Twitter, Github. Un punto importante a considerar es que la plataforma necesita una dirección de correo electrónico para enviar las notificaciones de las distintas acciones, como proponer corrección de preguntas y encuestas, validar estado de la pregunta etc**.**

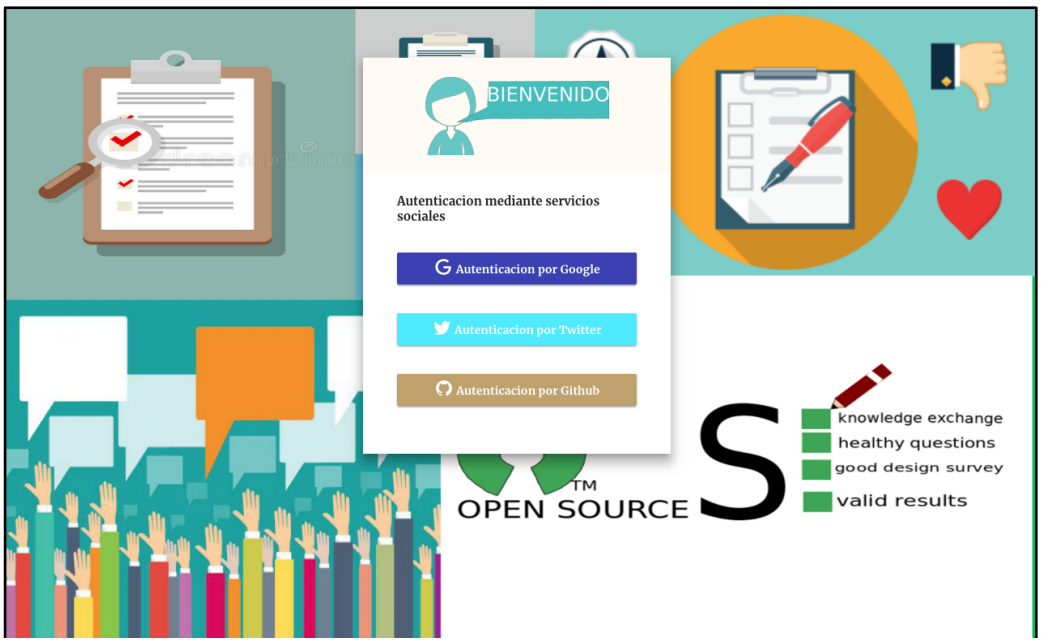

Figura 3.14 Autenticación en la plataforma

En la ***Figura 3.15 Tablero de Actividades***, se muestra un resumen del contenido que el usuario ha elaborado en la plataforma, como son: encuestas que ha comentado, preguntas que ha elaborado, comentarios positivos que otros usuarios han señalado al contenido que el usuario ha elaborado, etc.

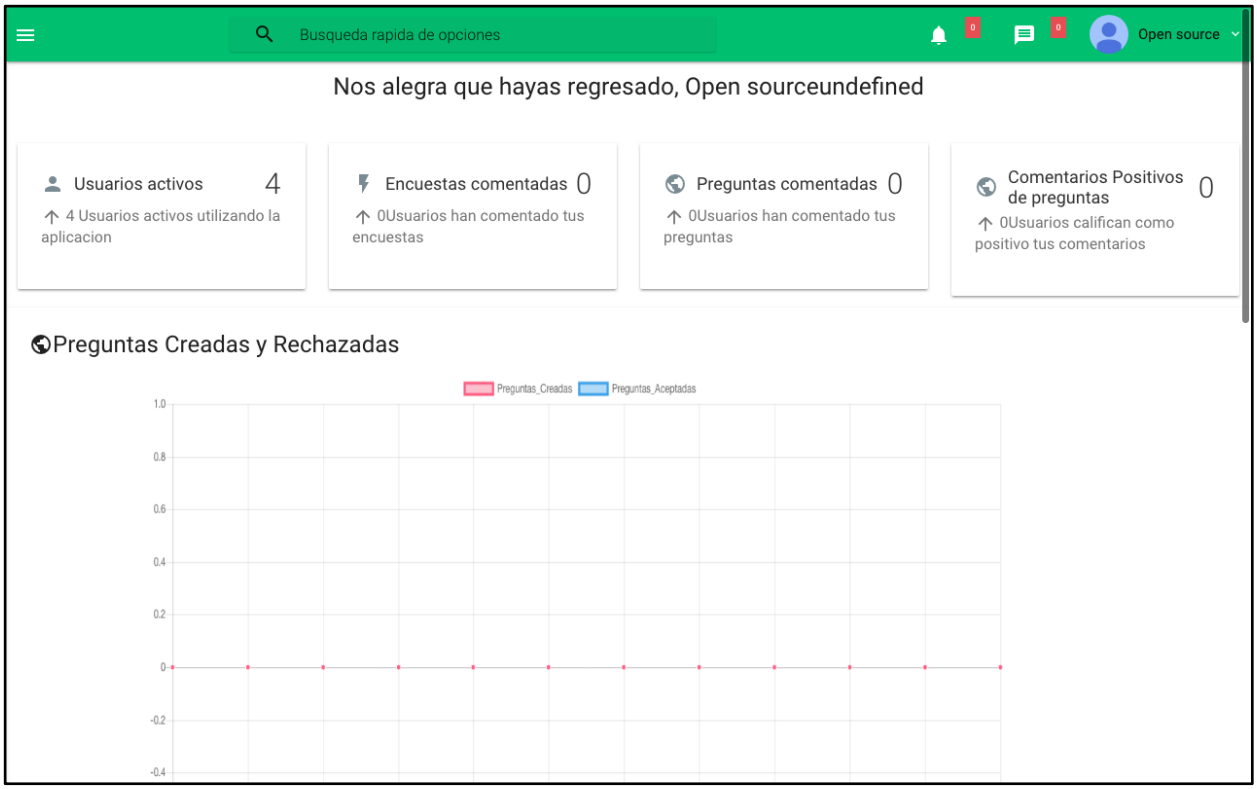

Figura 3.15 Tablero de Actividades

En la ***Figura 3.16 Crear Preguntas***, podemos observar los componentes que posee una determinada vista. Se pueden desarrollar preguntas con 6 diferentes tipos de respuestas.

Los tipos de respuestas que posee la plataforma son los siguientes:

1. Check list
2. Radio button
3. Lista desplegable
4. Rating
5. Pregunta abierta

6. Sí o no

Adicional a esto, el usuario puede añadir imágenes a una pregunta si necesita ser más descriptivo con la pregunta que está elaborando.

Es necesario que el usuario defina una categoría a dicha pregunta, esto permitirá a los demás usuarios poder reconocer qué tipo de pregunta se ha creado sin necesidad de ver toda la pregunta.

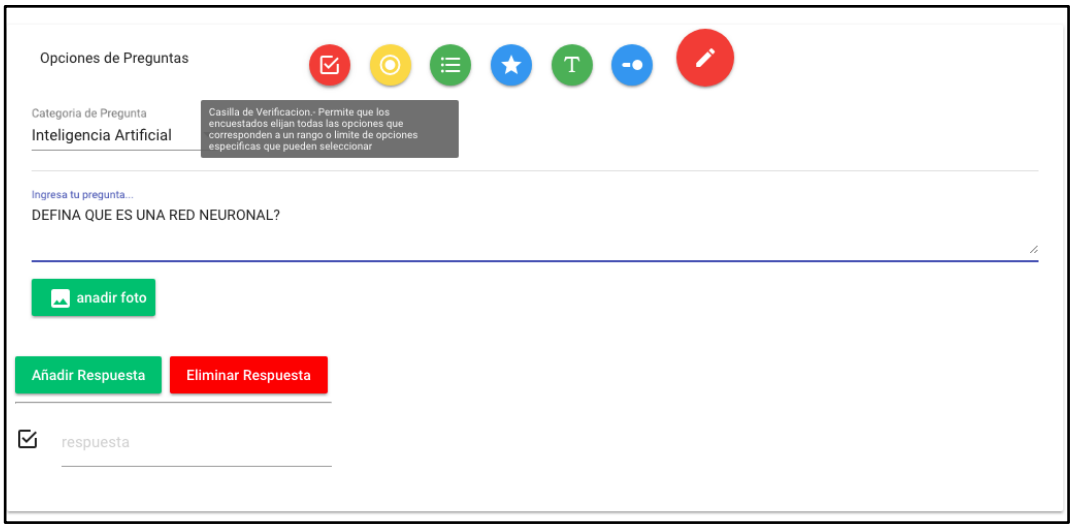

Figura 3.16 Crear Preguntas

Todas las preguntas que son creadas, se muestran en un foro donde todos los usuarios pueden comentar, proponer cambios y comentar aquellas cambios propuestos, como se muestra en la ***Figura 3.17 Foro de Preguntas***.

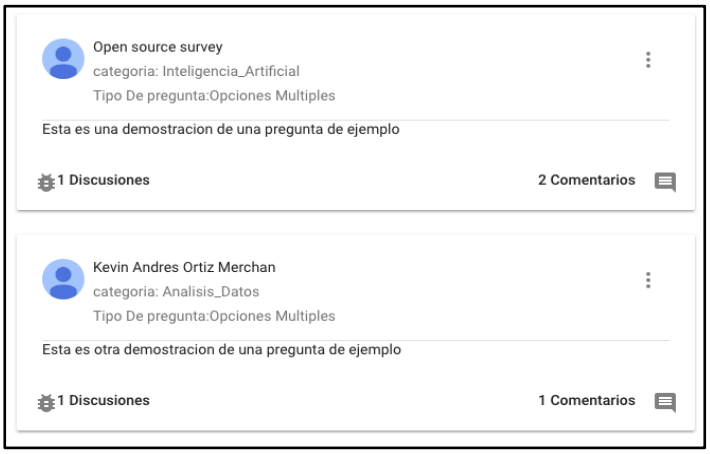

Figura 3.17 Foro de Preguntas

Las acciones que puede hacer un usuario al comentar una pregunta o proponer un cambio a una pregunta o encuesta son:

- Me encanta este comentario.
- Me gusta este comentario.
- No me gusta este comentario.

También los usuarios pueden opinar sobre comentarios hechos por otros usuarios. Esta función permite retroalimentar al creador de la pregunta o encuesta sobre el tipo de contenido que ha elaborado y recibir correcciones de otros usuarios que usan la plataforma, tal como lo muestra la **Figura 3.18 Acciones que se pueden realizar al comentar una pregunta**.

Figura 3.18 Acciones que se pueden realizar al comentar una pregunta

En la ***Figura 3.19 Proponer cambios sobre una pregunta*** , los usuarios pueden proponer cambios sobre una pregunta o encuesta, mostrando desde su punto de vista que partes de la pregunta o encuesta no tienen sentido o posee faltas ortográficas; añadiendo etiquetas o una breve descripción a los cambios que deben realizarse. Dichos cambios serán notificados al creador de la pregunta o de la encuesta.

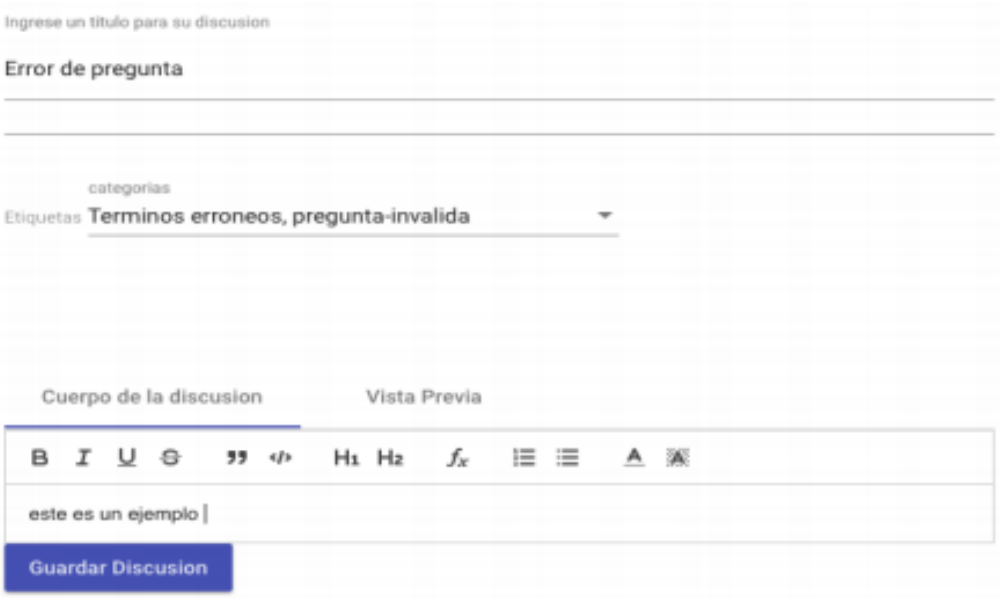

Figura 3.19 Proponer cambios sobre una pregunta

Un punto importante en nuestro proyecto es la meritocracia, lo que indica que un grupo de usuarios debe elegir si una pregunta debe ser aceptada o rechazada tal como lo indica la ***Figura 3.20 Validar estado de la pregunta***. La selección de usuarios que conformarán el comité se la realiza de forma manual, eligiendo qué usuarios han tenido una participación más activa en la plataforma. Esto se los puede descomponer en los siguientes puntos:

- Cantidad de preguntas aceptadas
- Cantidad de comentarios positivos
- Cantidad de comentarios negativos
- Cantidad de encuestas desarrolladas

Los usuarios que realizan esta elección son los administradores de la aplicación.

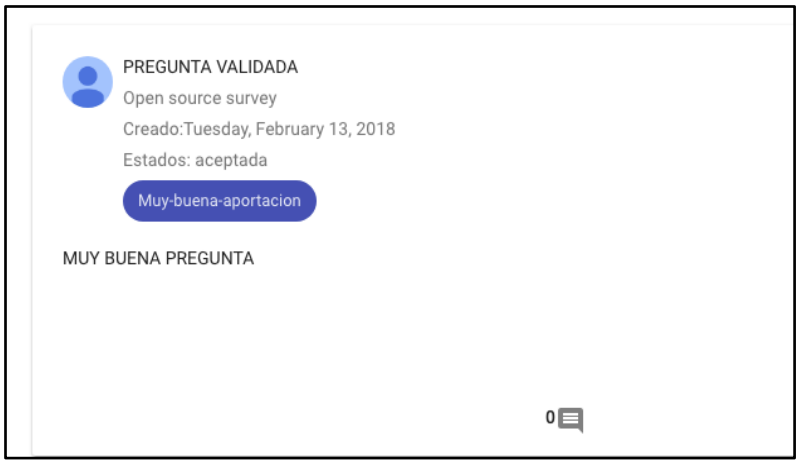

Figura 3.20 Validar estado de la pregunta

Todos los usuarios tienen la libertad de crear encuestas basadas en preguntas que han pasado por todo el proceso de validación y depuración tal como lo indica la ***Figura 3.21 Crear Encuesta***.

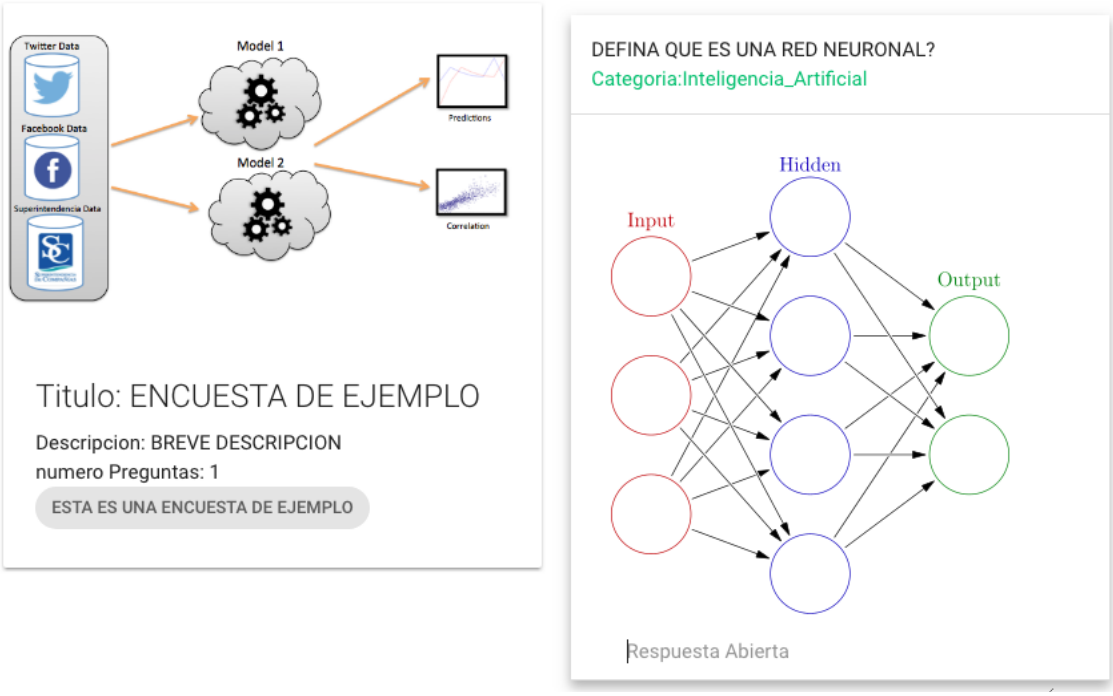

Figura 3.21 Crear Encuesta

CAPÍTULO 4

1. RESULTADOS DEL SISTEMA.

En este capítulo se mostrará los resultados que se han obtenido con el sistema desarrollado.

Las pruebas que se realizaron fueron de rendimiento y stress para un determinado grupo de usuarios que ingresan de manera concurrente a la aplicación web.

Estas pruebas de rendimiento se las efectuó usando una herramienta llamada Apache JMeter.

¿Qué es Apache JMeter?

Es un software de código abierto, escrito en Java y diseñado para pruebas de cargas y funcionales que sirven para medir el rendimiento en aplicaciones web.

Se lo puede usar para simular carga de usuarios en un servidor o grupo de servidores.

Se realizaron tres tipos diferentes de cargas en el servidor que maneja las peticiones de los recursos que solicitan los usuarios al momento de autenticarse e ingresar a la plataforma.

1. Una carga con 100 usuarios que ingresan de forma concurrente a la aplicación web en intervalos de 5 segundos, por cada requerimiento.
2. Una carga con 200 usuarios que ingresan de forma concurrente a la aplicación web en intervalos de 5 segundos, por cada requerimiento.
3. Una carga de 300 usuarios que ingresan de forma concurrente a la aplicación web en intervalos de 5 segundos, por cada requerimiento.

Las pruebas se realizaron en una computadora Intel Core I5 2.4 GHZ, con una memoria RAM DDR3 de 16 GB a 1333 MHz.

Las pruebas que se realizaron corresponden a la acción de un usuario que se autentica en la plataforma y automáticamente es redirigido al tablero de actividades.

En la ***Figura 4.22 Tiempo promedio en responder un requerimiento***, podemos observar que existe un comportamiento lineal donde el tiempo crece a medidas que más usuarios acceden a la aplicación web, siendo el tiempo en promedio máximo en microsegundos de 57 ms y el mínimo de 51 ms.

Figura 4.22 Tiempo promedio en responder un requerimiento

En cambio comparado con la latencia que podemos observar en la ***Figura 4.23 Tiempo Promedio que se demora el servidor en responder un requerimiento***, existe una similitud en los tiempos de respuesta del servidor, esto es debido a que los recursos en primera instancia no se encuentran almacenado en cache, y además el sistema verifica si el usuario ya existe o es un usuario nuevo que usa la plataforma.

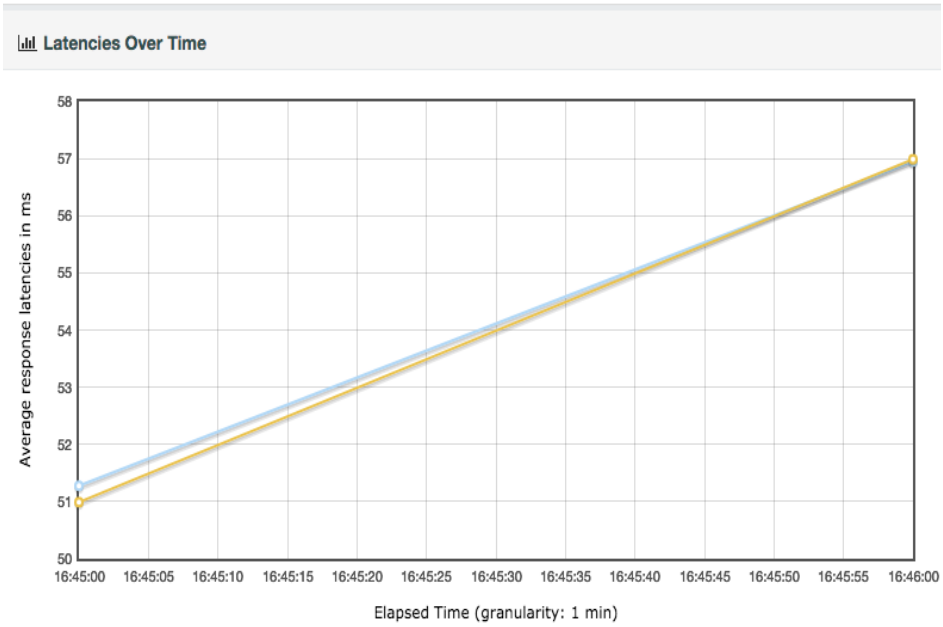

Figura 4.23 Tiempo Promedio que se demora el servidor en responder un requerimiento

En la ***Figura 4.24 Comparación del número de usuarios concurrentes en la plataforma vs tiempo de respuestas en promedio del servidor***, podemos observar que los puntos donde el servidor se tomó más tiempo en responder a un requerimiento fue cuando estaban activos 75, 85 y 95 usuarios.

Los tiempos de respuesta fueron mayor a 100 ms, al llegar a 100 usuarios concurrentes, después los tiempos de respuestas de ambas páginas (login y tablero) disminuyen en comparación al primer tiempo; siendo una de las causas que todos los recursos ya se encuentran en caché.

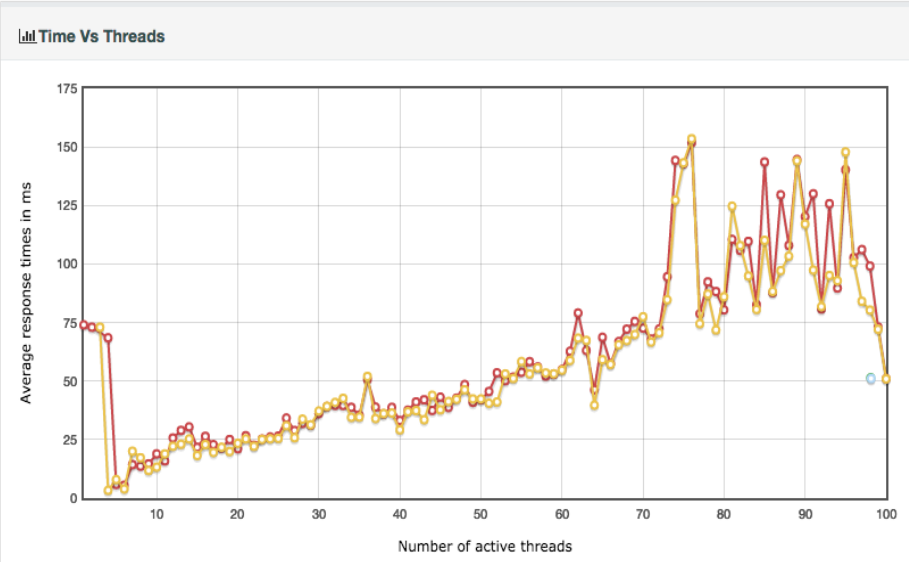

Figura 4.24 Comparación del número de usuarios concurrentes en la plataforma vs tiempo de respuestas en promedio del servidor

En la ***Figura 4.25 Tiempo promedio en responder un requerimiento para una carga de 200 usuarios concurrentes***, podemos observar que existe un comportamiento lineal con respecto al tiempo que se demora un servidor en responder un requerimiento. A medida que incrementa el número de usuarios que acceden a la aplicación web, siendo el número de usuarios concurrentes 200, el tiempo máximo fue de 240 ms y el mínimo de 221 ms

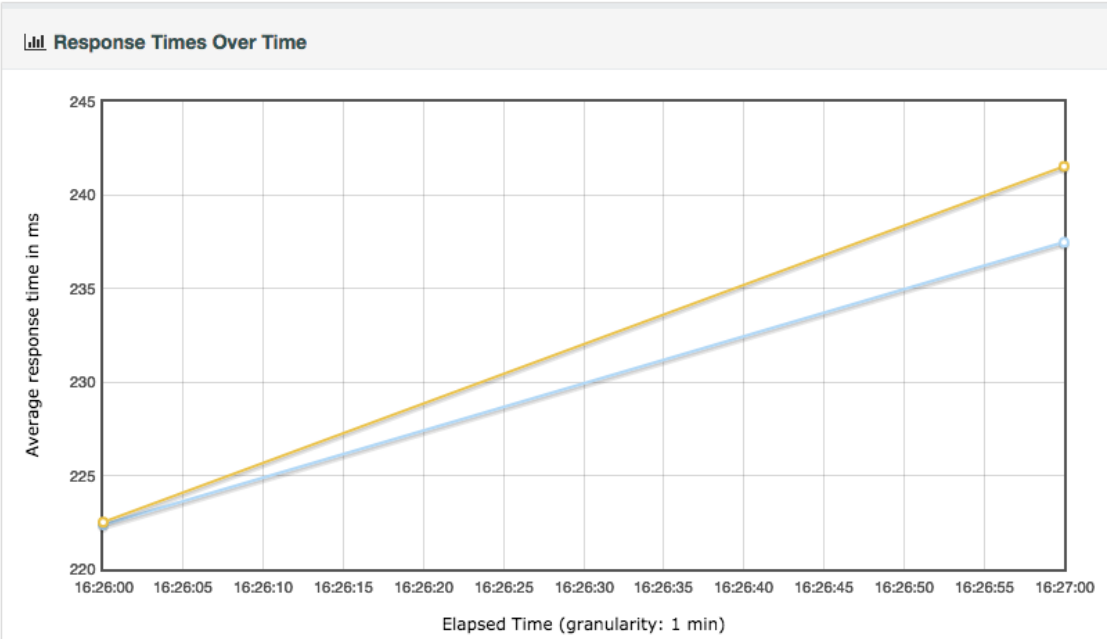

Figura 4.25 Tiempo promedio en responder un requerimiento para una carga de 200 usuarios concurrentes

En cambio comparado con el tiempo que se demora el servidor en responder un requerimiento, podemos observar en la ***Figura 4.26 Latencia promedio que se demora el servidor en responder un requerimiento***, que existe una similitud en los tiempos que se demora el servidor en responder un requerimiento. Esto se puede dar debido a que los recursos en primera instancia no se encuentran en la cache del servidor y el número de usuarios incrementa.

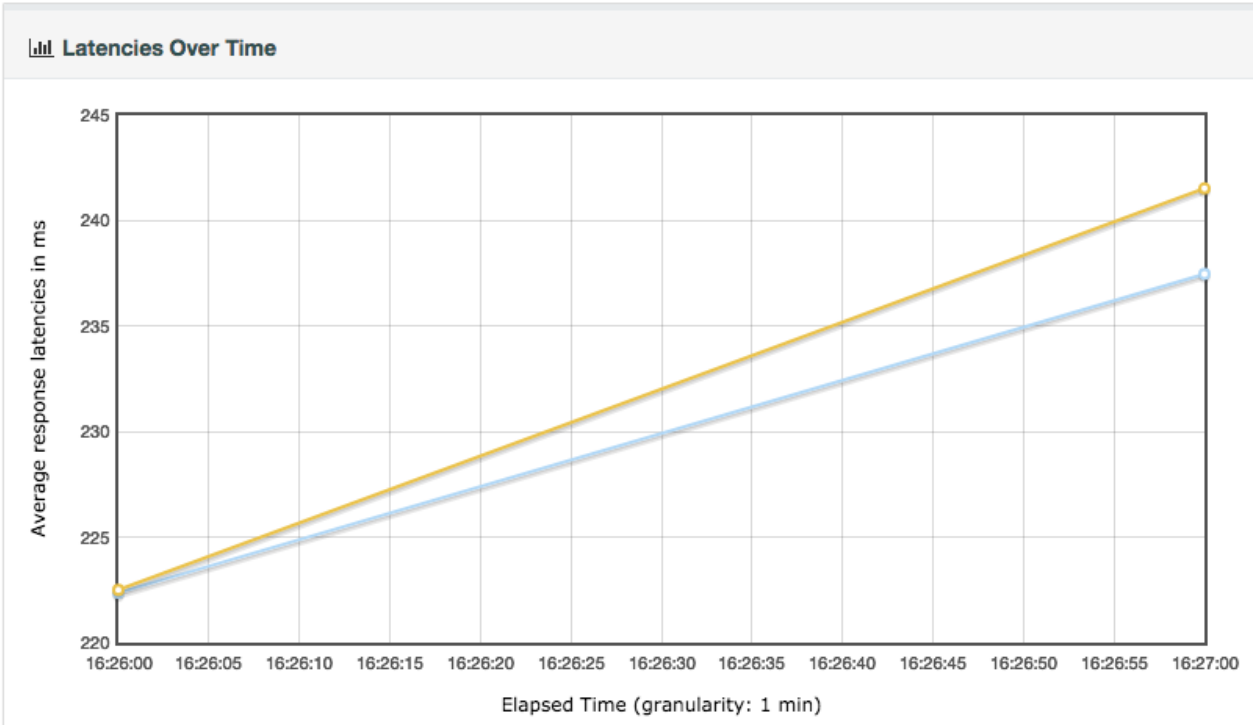

Figura 4.26 Latencia promedio que se demora el servidor en responder un requerimiento

En la ***Figura 4.27 Comparación del número de usuarios concurrentes en la plataforma vs Tiempo de respuesta en promedio del servidor para 200 usuarios***, podemos observar que mientras el número de usuarios aumenta, el tiempo que se demora el servidor en responder las peticiones de los usuarios disminuye. Si bien podemos observar que en cantidad de usuarios menores a 100 el tiempo en microsegundos que se demora el servidor en responder es de 650 ms, entre 100 y 200 usuarios, que es la carga que se quiere probar aquí, los tiempos oscilan entre 500 y 150 ms. Se enfatiza que en todos los tres escenarios realizados no hubo caídas, ni requerimientos fallidos que no se efectuaron.

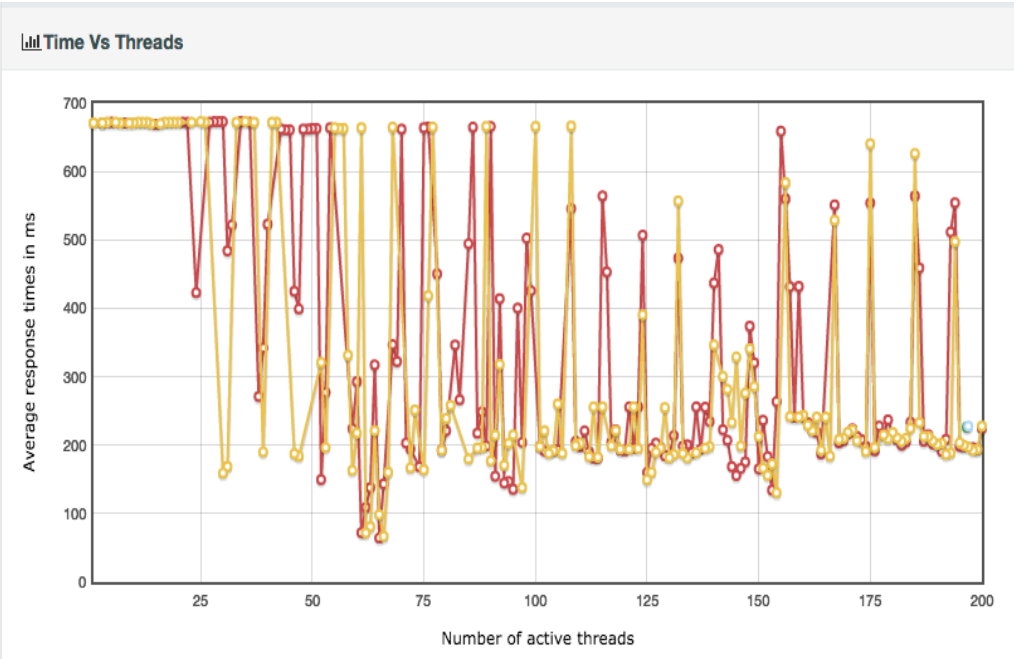

Figura 4.27 Comparación del número de usuarios concurrentes en la plataforma vs Tiempo de respuesta en promedio del servidor para 200 usuarios

En la ***Figura 4.28 Tiempo promedio en responder un requerimiento para una carga de 300 usuarios concurrentes***, podemos observar que existe un comportamiento lineal descendente en comparación con los escenarios anteriores analizados.

Como se puede observar al inicio de la ejecución de las pruebas de cargas, los tiempos están alrededor de los 350 ms.

Como se puede ver el tiempo en promedio máximo en microsegundos es de 350 ms y el mínimo de 310 ms.

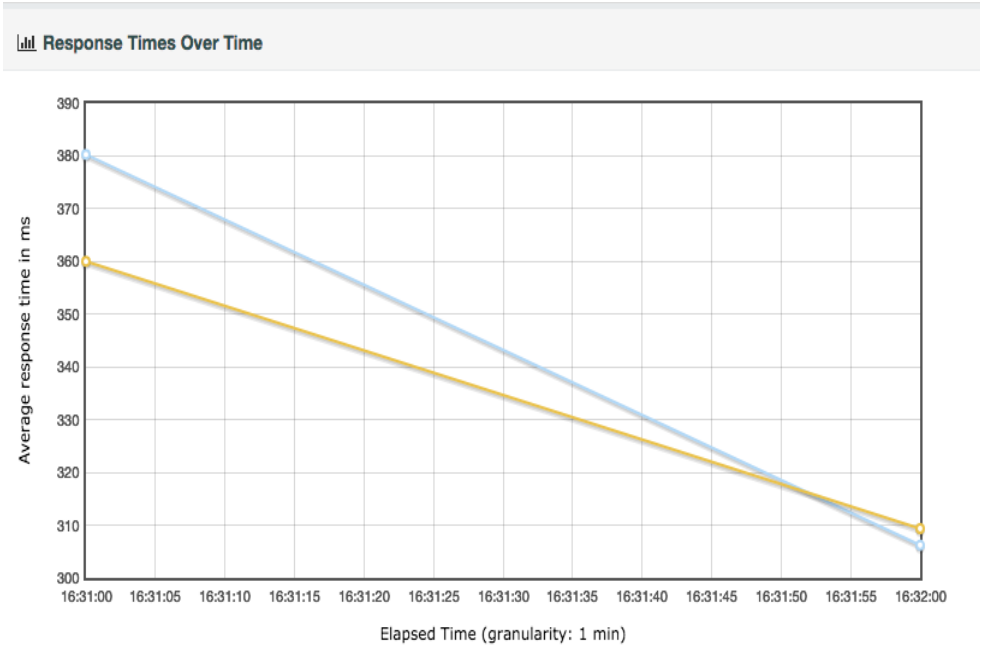

Figura 4.28 Tiempo promedio en responder un requerimiento para una carga de 300 usuarios concurrentes

En cambio comparado con la latencia, podemos observar en la ***Figura 4.29 Tiempo promedio que se demora el servidor en responder un requerimiento para 300 usuarios***, que existe un comportamiento lineal descendente con los tiempos de demora que tiene un servidor al responder un requerimiento.

Podemos observar que a medida que el tiempo transcurre, los tiempos de espera están alrededor de los 310 ms que aun siendo un número alto en comparación con otros escenarios de pruebas, tiene un buen comportamiento considerando el tiempo que se tomó al inicio de la ejecución con 300 usuarios.

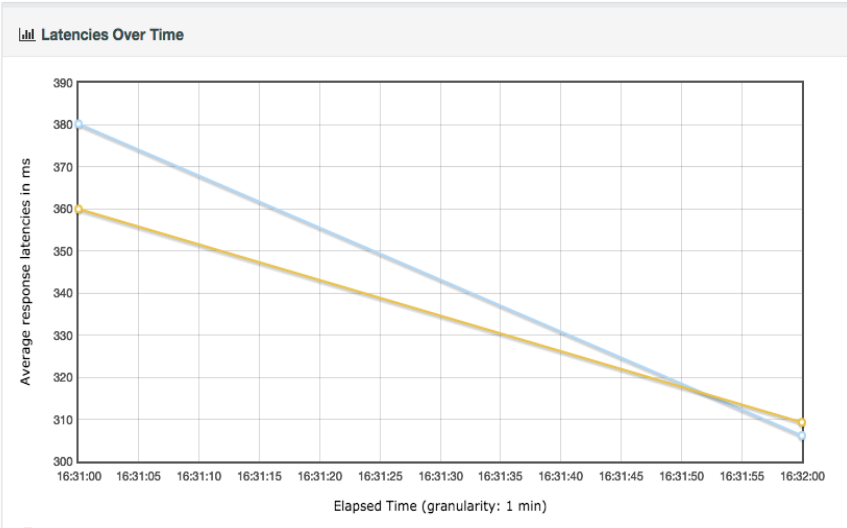

Figura 4.29 Tiempo promedio que se demora el servidor en responder un requerimiento para 300 usuarios

En la ***Figura 4.30 Comparación del número de usuarios concurrentes en la plataforma vs Tiempo de respuesta en promedio del servidor para 300 usuarios***, podemos observar que mientras el número de usuarios aumenta, el tiempo que se demora el servidor en responder las peticiones de los usuarios aumenta.

También podemos observar que hasta 175 usuarios concurrentes, el tiempo que se demora el servidor es de 900 ms, pero de 175 hasta 300 usuarios presenta otro comportamiento. De 200 a 250 usuarios el tiempo de respuesta del servidor es de 1 segundo, pero de 250 hasta 300 el tiempo de respuesta es de 500 ms.

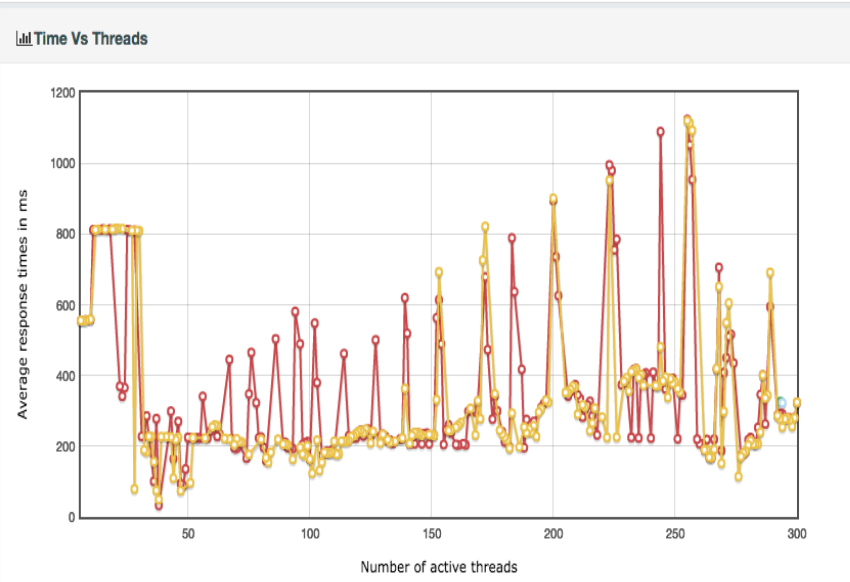

Figura 4.30 Comparación del número de usuarios concurrentes en la plataforma vs Tiempo de respuesta en promedio del servidor para 300 usuarios

CONCLUSIONES

Se desarrolló la aplicación web basado en el análisis del Capítulo 1 que mostraban los problemas que presentan los usuarios al diseñar una encuesta. La aplicación web muestra una buena escalabilidad para un determinado número de usuarios, midiendo la latencia, rendimiento, tiempo de respuesta de cada requerimiento. Los lineamiento del desarrollo de la aplicación web se cumplieron basados en las historias de usuarios tomados de los problemas analizados.

RECOMENDACIONES

Si bien la elección de usuarios que validan las preguntas que son aceptadas o rechazadas se hace de manera manual, podría en el futuro crearse un proceso que pueda analizar de forma automática los comportamientos de los usuarios y designe los roles a cada uno de los usuarios, dependiendo de las actividades que desarrolle en el sistema. Si bien todas los eventos que se manejan en la aplicación se manejan de forma asincrónica, en un futuro se podría considerar dividir la aplicación en módulos que trabajen de forma síncrona y en módulos que trabajen de manera asíncrona.

BIBLIOGRAFÍA

[1]T. J. DeMaio, J. Rothgeb, J. Hess et al., “Improving survey quality through pretesting,” Proceedings of the survey research methods section. American Statistical Association, Alexandria, 1998

[2]L. Meeden, T. Newhall, D. Blank, and D. Kumar, “Using departmental surveys to assess computing culture: Quantifying gender differences in the classroom,” SIGCSE Bull., vol. 35, no. 3, pp. 188–192, Jun. 2003.

[3]M. B. Durrant and C. R. Dorius, “Study abroad survey instruments: A comparison of survey types and experiences,” Journal of Studies in International Education, vol. 11, no. 1, pp. 33–53, 2007.

[4]J. Sinclair, M. Butler, M. Morgan, and S. Kalvala, “Measures of student engagement in computer science,” in Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ser. ITiCSE ’15. ACM, 2015, pp. 242–247.

[5]F. J. Fowler, Improving survey questions: Design and evaluation. Sage, 1995, vol. 38.

[6]F. J. Fowler, “How unclear terms affect survey data,” Public Opinion Quarterly, vol. 56, no. 2, pp. 218–231, 1992.

[7]Rafael I. Bonilla et al., ” A Case for Open-Source Surveys(for Assessing Security Literacy),”

[8] Lab, Redis How fast is Redis: https://redis.io/topics/benchmarks

.