Facultad 4

Sistema para la gestión automatizada de correos electrónicos de alerta de información

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Daniel Felipe Fuentes
Tutor: Dr.C. Omar Correa Madrigal

La Habana, 24 noviembre de 2023

Año 65 de la Revolución

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título *"**Sistema para la gestión automatizada de correos electrónicos de alerta de información*** "concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los 7 días del mes de 11 del año 2023.

Daniel Felipe Fuentes

_____

Firma del Autor

Omar Correa Madrigal

_____

Firma del Tutor

AGRADECIMIENTOS

Quiero expresar mi profundo agradecimiento a mi amada mamá Maribel Fuentes Ordoñes. Tu apoyo incondicional y tu amor inquebrantable han sido la fuerza impulsora detrás de mí éxito. Desde mis primeros pasos en la educación hasta este momento en el que finalizo mi tesis, siempre has estado ahí para alentarme, inspirarme y motivarme a dar lo mejor de mí. Tu paciencia y sacrificio han sido fundamentales en cada paso del camino. Este logro es tuyo tanto como mío, y no tengo palabras suficientes para expresar cuánto te amo y te agradezco por todo lo que has hecho por mí. Gracias, mamá.

A mi amado papá Fidel F. Felipe Valdez, quien, aunque ya no está físicamente a mi lado, siempre ha sido y seguirá siendo una inspiración en mi vida. Aunque no puedo expresar mis palabras de gratitud directamente a él, quiero reconocer su papel fundamental en mi desarrollo académico y personal. Papá, tus valores, tu dedicación y tu amor incondicional han dejado una huella imborrable en mi corazón y en mi determinación de alcanzar mis metas. Aunque no puedo abrazarte o escuchar tus palabras de aliento en este momento, sé que estás conmigo en espíritu, orgulloso de cada paso que doy. Te extraño profundamente y siempre serás mi guía. Gracias, papá, por todo lo que hiciste y sigues haciendo por mí.

No puede faltar agradecer a mi novia que ha estado ahí apoyándome en todo momento, tu amor, tus palabras de aliento y tu comprensión han sido un faro de luz en los momentos en que me sentía abrumado. Gracias por creer en mí, incluso cuando yo dudaba de mis propias habilidades. Tu paciencia y compasión infinitas han sido un regalo que nunca olvidaré. Mi éxito en esta tesis no habría sido posible sin tu amor y apoyo constante. Te amo más de lo que las palabras pueden expresar y estoy eternamente agradecido por tenerte a mi lado.

Además, quiero agradecer de manera especial a mis mejores amigos, el Flaco y Morales, vuestra amistad ha sido un verdadero pilar en mi vida, llenándola de risas, apoyo y momentos inolvidables. Agradezco sinceramente vuestro respaldo inquebrantable y vuestra confianza en mis capacidades. Asimismo, deseo expresar mi gratitud a todos mis amigos y profesores de la universidad, en especial a Osmany, el Javao, Josué y Elier. Vuestra amistad ha sido una fuente de inspiración y compañerismo. Juntos hemos compartido largas horas de estudio, debates académicos y valiosas experiencias en el aula y como olvidar a mi profesor Luis Manuel, quien

ha sido no solo un mentor académico, sino también como un hermano para mí. Su dedicación y su guía han sido fundamentales en mi desarrollo académico y personal. Vuestro apoyo y vuestra disposición para colaborar han sido cruciales para alcanzar nuestros objetivos académicos. Estoy sinceramente agradecido por vuestra amistad y por ser parte de mi vida universitaria.

Al final, pero no menos importante a mi padrastro Luis Manuel tu presencia en mi vida ha sido invaluable. Tu apoyo y guía han sido fundamentales en mi crecimiento personal y académico. Tus palabras de aliento han sido una luz en los momentos de incertidumbre. Agradezco sinceramente todo el amor y el apoyo que me has brindado a lo largo de los años.

DEDICATORIA

A mis amados padres Maribel Fuentes Ordoñes y Fidel F. Felipe Valdez, quienes han sido mi mayor inspiración y apoyo a lo largo de mi vida y en la realización de esta tesis. Mamá y papá, vuestra dedicación, amor incondicional y sacrificio han sido los pilares fundamentales en mi camino hacia el logro de mis metas académicas.

RESUMEN

Es crucial para las empresas, instituciones y organizaciones dedicadas a la informática, la comunicación y las tecnologías mantenerse actualizadas en un entorno en constante evolución. La investigación constante es necesaria debido a los avances e innovaciones diarios en este campo. La actualización permanente es también fundamental para garantizar un desempeño eficiente y competitivo en un entorno dinámico.

Para lograrlo, es necesario realizar investigaciones que permitan comprender la evolución y el comportamiento de estas áreas. Esto proporciona la información necesaria para adaptarse y aprovechar las oportunidades en el mercado en constante desarrollo. Mantenerse informado sobre las últimas tendencias y avances permite tomar decisiones informadas y mantener la competitividad.

El estudio del estado del arte determinó que no existe un sistema para gestionar de manera automatizada los correos electrónicos de alerta de información, el sistema que se implementó incluye entre sus principales funcionalidades: la creación de alertas de temas de interés, la obtención de contenido de fuentes de información y el envío de notificaciones de las mismas.

La metodología en la que se basó el proceso de desarrollo de software fue *extreme programming* (XP) y se concibió la documentación correspondiente a las etapas: planificación, diseño, codificación y pruebas. Para la validación del sistema se ejecutaron las pruebas aceptación y las unitarias.

PALABRAS CLAVE: Alertas, información, extreme programming (XP), notificaciones, sistema.

ABSTRACT

It is crucial for companies, institutions and organizations dedicated to IT, communication and technologies to keep updated in a constantly evolving environment. Constant research is also necessary due to the daily advances and innovations in this field. Permanent updating is essential to guarantee an efficient and competitive performance in a dynamic environment.

To achieve this, it is necessary to conduct research to understand the evolution and behavior of these areas, this provides the necessary information to adapt and take advantage of opportunities in the constantly developing market, staying informed about the latest trends and developments allows to make informed decisions and maintain competitiveness.

The study of the state of the art determined that there is no system to manage in an automated way the information alert emails, the system that was implemented includes among its main functionalities: the creation of alerts of topics of interest, the obtaining of content from information sources and the sending of notifications of the same.

The methodology on which the software development process was based was extreme programming (XP) and the documentation corresponds to the stages: planning, design, coding and testing. For the validation of the system the acceptance and unit tests were executed.

KEYWORDS: *Alerts, system, information, notifications, extreme programming (XP)*

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las Tecnologías de la Información y Comunicación (TIC) han cambiado nuestra sociedad y la adaptación a ellas direcciona nuestra evolución, la conciencia digital, global, interdependiente y conectada es una realidad [1]. La llegada de internet y los constantes avances tecnológicos actuales conllevan una serie de transformaciones que han repercutido en el mundo educativo dando lugar a nuevas formas de comunicarse, interaccionar, relacionarse, trabajar y aprender, es así, que las TIC se han convertido en una herramienta crucial para el futuro de la educación, estas pueden suponer oportunidades importantes para mejorar la educación de los estudiantes y docentes [2].

Si bien se han realizado bastantes esfuerzos de aproximación en la última década, se considera que aún hay que profundizar más, sobre todo desde la consideración de los nuevos escenarios propiciados por las TIC, que están permitiendo toda una redefinición, no solo del perfil en clave de nuevas funciones y roles profesionales, sino también de las propias competencias profesionales, competencias digitales [3].

Con el objetivo de responder a la necesidad de observación y análisis permanente, surge la vigilancia tecnológica como proceso en la toma de decisiones estratégicas para la innovación, puesto que producto de la observación y el análisis, pueden detectarse tendencias emergentes u obsoletas en una era caracterizada por la globalización de la producción y del consumo, así como por los rápidos cambios tecnológicos, cuyas repercusiones sobre las personas y el medioambiente hacen del conocimiento científico y comercial una necesidad de primer orden para el progreso de la investigación, desarrollo tecnológico e innovación (I+D+i), la competitividad y la responsabilidad social, universitaria y empresarial. En este sentido la vigilancia tecnológica se convierte en un elemento para el aprovechamiento responsable de los avances de la ciencia y la tecnología, propiciando a través de sus hallazgos la inteligencia competitiva y, con ello, más oportunidades para la apropiación social del conocimiento y para el desarrollo socio-económico, sostenible e inclusivo [4].

La Universidad de las Ciencias Informáticas (UCI) como una institución que se basa en la formación de profesionales altamente capacitados en tecnología de la información y la comunicación, ha contribuido significativamente al desarrollo del sector en Cuba. La

universidad ha asumido un papel activo en la promoción de la vigilancia tecnológica y la gestión de la información en línea como herramientas para la toma de decisiones estratégicas y ha demostrado su compromiso con el desarrollo socio-económico, sostenible e inclusivo en el país.

En el contexto de las empresas, instituciones y organizaciones involucradas en el desarrollo de la informática, la comunicación y las tecnologías destinadas a la sociedad, resulta imperativo mantenerse al tanto de su evolución y comportamiento, dicha comprensión debe ser objeto de una investigación constante, dado que diariamente se generan avances e innovaciones relevantes en este ámbito. Esta necesidad de actualización permanente constituye un elemento fundamental para garantizar el desempeño eficiente y competitivo de estas entidades en un entorno dinámico y cambiante.

Para facilitar este proceso, es importante contar con sistemas de correos de alertas eficientes que permitan estar al tanto de las novedades que surgen a diario, es necesario reconocer que el desarrollo y la adopción de estos sistemas pueden verse afectados por diversos factores, en algunos casos, se han utilizado plataformas internacionales para cubrir las necesidades, aunque esto puede limitar la flexibilidad y adaptabilidad del sistema. Además, es relevante destacar que la implementación y el alcance pueden variar entre diferentes regiones y sectores, debido a diferencias en recursos, capacidades y enfoques estratégicos.

Con el objetivo de lograr lo anteriormente planteado, el Ministerio de las Comunicaciones en colaboración con la UCI determina la necesidad de poner en práctica un sistema para la gestión automatizada de correos electrónicos de alerta de información que sea capaz de abarcar las distintas etapas del mismo.

A partir de lo anteriormente descrito se plantea el siguiente **problema de investigación**: ¿Cómo se puede mantener informado a entidades u organizaciones sobre las novedades tecnológicas con alertas específicas para mejorar la vigilancia tecnológica y la toma de decisiones?

Como **objeto de estudio** se establece los sistemas de alertas de información.

Objetivo general: Desarrollar un sistema de gestión automatizada de alerta a través de correo electrónico para la vigilancia científico-tecnológica.

Se define como **campo de acción** los sistemas de alertas de correos electrónicos de información en línea para la vigilancia científico-tecnológica.

Tareas de investigación:

1. Realizar una revisión de la literatura existente sobre herramientas de gestión automatizada de correos electrónicos de alerta de información en línea.
2. Seleccionar diferentes tecnologías y herramientas para la implementación del sistema automatizado de gestión de correos electrónicos de alerta de información en línea.
3. Generar los artefactos ingenieriles de acuerdo a la metodología seleccionada.
4. Implementar el sistema automatizado de gestión de correos electrónicos de alerta de información.
5. Realizar de las pruebas de aceptación y unitarias para validar el correcto funcionamiento del sistema.

Con el propósito de alcanzar la solución deseada y facilitar el desarrollo de investigación se utilizaron los siguientes **métodos de investigación.**

Métodos científicos:

- Investigación exploratoria: Para examinar la literatura existente y explorar las herramientas de gestión automatizada de correos electrónicos de alerta de información en línea.
- Investigación descriptiva: Para identificar los requisitos del sistema automatizado de gestión de correos electrónicos de alerta de información en línea a través de encuestas, entrevistas y análisis de casos de uso.
- Investigación experimental: Para desarrollar y evaluar el sistema automatizado de gestión de correos electrónicos de alerta de información en línea mediante pruebas de campo y comparaciones con otros métodos de gestión de información.

Técnicas e instrumentos para la recopilación de datos:

- Entrevistas: Para obtener información más detallada sobre los requisitos del sistema y cómo los usuarios interactúan con él. Ver en los anexos.
- Análisis de documentos: Para destacar y reagrupar los principales temas y desarrollar y analizar después cada tema los diferentes contenidos.

- Análisis de datos: Para explorar cómo el sistema puede ser utilizado para la detección temprana de tendencias y la identificación de oportunidades de innovación en línea, y cómo esta información puede ser utilizada en la vigilancia tecnológica para identificar oportunidades y amenazas emergentes.

Estructura de la investigación:

Capítulo 1: Fundamentos teóricos del sistema para la gestión automatizada de correos electrónicos de alerta de información, capítulo encargado de definir los elementos teóricos necesarios para el desarrollo de la investigación y los principales conceptos que se emplearán durante todo el trabajo. Se realiza un análisis de las soluciones similares y se selecciona la metodología de desarrollo de software y las herramientas y tecnologías a utilizar.

Capítulo 2: Análisis y diseño del sistema para la gestión automatizada de correos electrónicos de alerta de información, capítulo poseedor de la modelación de la solución mediante los artefactos planteados por la metodología de desarrollo a utilizar.

Capítulo 3: Validación del sistema para la gestión automatizada de correos electrónicos de alerta de información, capítulo que contiene las diferentes pruebas a las que será sometida la aplicación una vez terminada para verificar su correcto funcionamiento.

CAPÍTULO I: FUNDAMENTOS TEÓRICO DEL SISTEMA PARA LA GESTIÓN AUTOMATIZADA DE CORREOS ELECTRÓNICOS DE ALERTA DE INFORMACIÓN.

El presente capítulo comprende el estudio de varios conceptos y características generales asociados a la gestión automatizada de correos electrónicos de alertas de información. Se caracteriza el proceso de desarrollo de software, lenguajes de programación y tecnologías que se utilizan para el desarrollo de software en la web. Las tecnologías usadas en la actualidad también serán objeto de comparación especificando las ventajas y desventajas, así como sus características.

I.1 Conceptos asociados al objeto de estudio.

Las alertas tecnológicas: Proporcionan información actualizada sobre los más recientes que se estén publicando en el mundo en relación a un tema técnico concreto. Las temáticas de cada alerta serán establecidas en colaboración con las plataformas tecnológicas y otras instituciones u organismos, para que así respondan a las necesidades de información concretas de empresas y organismos públicos de investigación de los distintos sectores tecnológicos [5].

Tipos de alertas tecnológicas:

- Alertas de seguridad informática: Son alertas sobre vulnerabilidades de software, *malware*, ataques de *hackers* [6], por ejemplo, alertas de actualización de parches de seguridad para sistemas operativos.
- Alertas de novedades tecnológicas: Son anuncios de lanzamiento de nuevos productos tecnológicos, actualizaciones de software o hardware [7], por ejemplo, alertas de lanzamiento de nuevos modelos de *smartphones*, computadoras, *gadgets*.
- Alertas de tendencias tecnológicas: Son reportes o análisis de las tecnologías emergentes que van a ser tendencia [8], por ejemplo, alertas sobre la adopción de la inteligencia artificial, el 5G y la computación en la nube.

Vigilancia tecnológica (VT): Es un sistema organizado de observación y análisis del entorno, tratamiento y circulación interna de los hechos observados y posterior utilización en la empresa. De igual manera la norma UNE 166006 define la vigilancia tecnológica “como el

proceso organizado, selectivo y sistemático, para captar información del exterior y de la propia organización sobre ciencia y tecnología, seleccionarla, analizarla, difundirla y comunicarla, para convertirla en conocimiento con el fin de tomar decisiones con menor riesgo y poder anticiparse a los cambios"[9].

Ventajas

Entre las ventajas que conlleva realizar una adecuada VT en las organizaciones, se pueden citar las siguientes:

- Conocer cambios de las tecnologías y en los mercados próximos al entorno organizacional.
- Reducir riesgos en la toma de decisiones.
- Conocer nuevas necesidades de los clientes.
- Dirigir los esfuerzos de innovación hacia aquellas tendencias que lo ameriten.
- Conocer mejor la competencia.
- Buscar alianzas con nuevos socios o asesoramiento de expertos.

Ciclo de VT

A pesar de que cada proceso de VT se debe diseñar adecuándose a las necesidades y las condiciones del caso, se podría definir el siguiente ciclo de manera genérica:

1. Definición de las necesidades y objetivos de realizar un análisis de VT.
2. Selección de las fuentes de información de donde extraer los datos.
3. Extracción y captación de los datos.
4. Organización y análisis de la información.
5. Toma de decisiones y planteamiento de estrategias tecnológicas.
6. Inicio de un nuevo ciclo, adecuándose a las nuevas necesidades y objetivos trazados.

Alerta: El término alerta, del italiano *allerta*, hace referencia a una situación de vigilancia o atención. Un estado o una señal de alerta es un aviso para que se extremen las precauciones o se incremente la vigilancia [10].

Sistemas de alertas tecnológicas: Son herramientas que ayudan a identificar y monitorizar las últimas novedades, avances y tendencias en el campo de la tecnología. Estos sistemas recopilan información de diversas fuentes, como sitios web, blogs, noticias, redes sociales, patentes y publicaciones científicas, proporcionando actualizaciones periódicas a los

usuarios interesados. La finalidad de los sistemas de alertas tecnológicas es mantener a las personas y organizaciones informadas sobre los avances tecnológicos relevantes en su área de interés. Esto les permite estar al tanto de las últimas innovaciones, descubrimientos y tendencias emergentes, lo que puede ser crucial para tomar decisiones estratégicas, identificar oportunidades de negocio, anticiparse a cambios en el mercado y mantenerse competitivos en su sector.

Gestión: Acción que implica planificar, organizar, motivar, dirigir y controlar, desde un punto de vista general, y, en forma específica, prever, ordenar, atender a los objetivos, la integración de los esfuerzos y la efectividad de las aportaciones de los demás, con el fin de lograr el desarrollo de las organizaciones [11].

Web *scraping*: El *scraping*, es una importante técnica de recogida automatizada de datos en línea. Es una de las prácticas más distintivas asociadas a las formas actuales de investigación social digital, las marcadas por la aparición de Internet y la nueva omnipresencia de los datos digitales en la vida social [12].

En general, la web *data scraping* se puede definir como el proceso de extraer y combinar contenido de interés de la web de manera sistemática. En dicho proceso, un agente de software, también conocido como robot web, imita la interacción de navegación entre los servidores web y los humanos en un recorrido web convencional. Paso a paso, el robot accede a tantos sitios web como sea necesario, analiza su contenido para encontrar y extraer datos de interés, y estructura ese contenido según se desee [13].

API: Son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. API significa “interfaz de programación de aplicaciones”. En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones, este contrato define cómo se comunican entre sí mediante solicitudes y respuestas [14].

Canales RSS: Un archivo o canal RSS *(Really Simple Syndication o Rich Site Summary)* es un sublenguaje derivado del XML *(Extensible Markup Language)*; el cual está diseñado para la distribución de información y noticias albergadas en los sitios web o en los *weblogs*, siendo su característica principal proporcionar lo último o más importante de un sitio sin tener que acceder a él [15].

I.2 Soluciones informáticas para la gestión de correos electrónicos de alertas de información.

Se realiza el análisis de sistemas, que permiten apoyar el desarrollo de la investigación para lograr adaptar el sistema para la gestión automatizada de correos electrónicos de alertas de información a las necesidades de los clientes.

Al analizar y evaluar los procesos existentes se logra identificar fortalezas y debilidades de los sistemas homólogos existentes, así como identificar oportunidades de mejora y diferenciación para el sistema propuesto.

Tras realizar una encuesta a los clientes se obtuvo información valiosa sobre las características deseadas para el sistema a desarrollar. Los resultados revelaron que los clientes esperaban que el sistema presentara las siguientes características:

- Notificaciones por correo electrónico: Los clientes expresaron la importancia de recibir las alertas de información directamente en sus buzones de correo electrónico, permitiendo acceder a estas de manera conveniente y en tiempo real, sin necesidad de consultar otras plataformas o aplicaciones adicionales.
- Selección de fuentes de información: Los clientes manifestaron el deseo de poder elegir las fuentes de información relevantes para ellos, para personalizar su experiencia y recibir alertas basadas en sus áreas de interés o necesidades específicas.
- Acceso abierto: Expresaron el interés en que el sistema estuviera disponible de manera gratuita y sin restricciones excesivas, garantizando que el sistema fuera accesible para un amplio rango de usuarios.
- Selección de frecuencia de notificaciones: Destacaron la importancia de poder elegir la frecuencia con la que reciben las alertas. Algunos prefieren recibir notificaciones en tiempo real, mientras que otros pueden optar por resúmenes diarios o incluso notificaciones semanales.

Algunos de los **sistemas analizados** son:

***Change Detection*:** Su función es avisar mediante un correo electrónico de cualquier cambio que se produzca en la página web que indique el usuario, tanto de los textos que se añadan como de los que se eliminen. Su ventaja más destacable es su sencillez, tan solo será necesario introducir la URL de la web de la que se desea tener conocimiento y ella hará el

resto. Es recomendable si no se dispone de demasiado tiempo para trabajar con otras plataformas más complejas [16].

BioIsChanged: Su objetivo es muy específico, informar mediante email de cuando un contacto de Twitter hace cambios en su biografía, por ejemplo, para ser el primero en enterarse de cuando un personaje público incorpora algo relevante en su cuenta de Twitter. [17].

***Talkwalker Alerts*:** Para crear alertas en *Talkwalker*, hay que dirigirse al formulario de la página web, aquí pedirán el término de búsqueda en cuestión, las fuentes en las que buscar (medios de comunicación, Twitter, blogs y foros), el idioma, con qué frecuencia se recibiran avisos del tema y si recibiremos todos los resultados o solo los más importantes. Esta herramienta también ofrece la posibilidad de seguir las campañas y los *hashtags* que se utilizan en redes sociales de manera gratuita y en tiempo real [18].

If This Then That (IFTTT): Es una de las más completas y útiles, permite conectar diferentes plataformas y cuentas (*SMS, Youtube, Twitter, Facebook, Dropbox, Tumblr, Gmail*, etc.). Para un uso profesional puede hacer cosas tan prácticas como que cuando alguien publique en Instagram una foto bajo un *hashtag* determinado, se envíe un correo informándote. Otro ejemplo podría ser que, tras suscribirse a la RSS solicitar que se informe cuando se produzca un terremoto de gran intensidad en alguna parte determinada del mundo, estas alertas se comunican mediante SMS al móvil.

Las peticiones se hacen con la sencilla fórmula «*If-Then*». Con la primera, se selecciona lo que se desee y en qué plataforma; con la segunda, se concreta el destino y la acción final de la alerta solicitada [19].

Mention: Igual que la herramienta anterior, analiza entre millones de fuentes para saber qué están diciendo en internet, de nuestra marca y hasta de la competencia. Está enfocada a una utilización profesional y cuenta con una versión de prueba gratuita y con varios planes de pago [20].

Social Mention: La herramienta supervisa más de cien sitios de medios sociales. Está considerada como una de las mejores herramientas gratis de escucha activa en internet; ofrece análisis e informes de datos.

La característica diferenciadora es que *Social Mention* establece cuatro categorías de resultados: fuerza, sentimiento, pasión y alcance [21].

Awario: Está disponible en inglés y va más enfocada a un uso profesional y del ámbito de los negocios, se encarga de investigar todo lo que dicen en redes sociales. A diferencia de otras aplicaciones de monitorización que dependen de proveedores de datos de terceros sus servicios rastrea más de 13 mil millones de páginas webs a diario, cuenta con una versión de prueba gratis durante 14 días, sin compromiso. Transcurrido este tiempo, se elige entre uno de los planes de pago, que variarán en función del número de alertas, estos precios van desde 19 dólares al mes para diez alertas, 50.000 menciones al mes y análisis de la aplicación, hasta 299 dólares al mes para 50 alertas, 500.000 menciones, análisis e informes [22].

Google Alerts: Proporciona los medios para rastrear, así se puede estar al tanto de los temas más importantes para el éxito de estrategias de *marketing* digital sin tener que investigar cada una. La principal ventaja es su practicidad para traer noticias sobre los temas elegidos directamente al usuario [23] es de libre acceso, pero busca información de fuentes de información poco confiables.

A continuación, se muestra una tabla resumen donde se evidencia las características que cumplen los sistemas homólogos analizados:

Tabla 1 Características de los sistemas homólogos. Fuente: Elaboración propia

Sistemas de alerta	**Notificaciones por correo**	**Selección de fuente de información**	**Acceso abierto**	**Selección de frecuencia de notificaciones**
Change Detection	X	X	X	
BioIsChanged	X			
Talkwalker Alerts	X	X		X
If This Then That (IFTTT)	X			
Mention	X		X	
Social Mention	X			
Awario	X			

Google Alerts	X		X	X

Después de examinar lo anterior, se llegó a la conclusión de que ninguna de las soluciones existentes para automatizar la gestión de correos electrónicos de alertas de información satisface completamente los criterios definidos en función de las necesidades que debe abordar la solución propuesta. Sin embargo, a pesar de contar con *Change Detection*, *Talkwalker Alerts, Google Alerts,* soluciones que cumplen casi todos los criterios, no es posible seleccionarlas debido a que no son de código abierto. Esta limitación resalta la importancia de desarrollar la aplicación propuesta, la cual se basará en un enfoque de código abierto para garantizar la adaptabilidad y personalización requeridas.

I.3 Metodología, herramientas y técnicas utilizadas para el desarrollo de la solución.

I.3.1 Metodología de Programación Extrema (XP)

La metodología *eXtreme Programming-XP* seleccionada por los clientes, recibe su nombre del proceso de tomar la mejor práctica y luego llevarla al extremo, desarrollado por *Kent Beck* , manejando el problema de la complejidad de los requisitos crecientes a lo largo del desarrollo del software, estos cambios se manejan sin exceder el tiempo y el presupuesto. Consiste en una lista de procesos que pueden ser seleccionados para abordar una necesidad de mejora pertinente, siendo estos principios probados en la ingeniería de software [24].

La metodología ágil XP se enfoca en las relaciones interpersonales como un factor clave para el éxito en el desarrollo de software. Esta metodología promueve el trabajo en equipo, el aprendizaje continuo de los desarrolladores y un ambiente de trabajo agradable y productivo. Se basa en la retroalimentación constante entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes.

Algunas de las características que se tuvieron en cuenta para seleccionar esta metodología fueron las siguientes:

1. Enfoque iterativo e incremental: Al basarse en ciclos cortos permite una evolución gradual del sistema ayudando en el proyecto, pudiendo implementar funcionalidades básicas rápidamente y luego mejorarlas iterativamente en sucesivas iteraciones.

2. Retroalimentación constante: Esta metodología fomenta la retroalimentación entre el equipo de desarrollo y los usuarios/cliente, beneficiando al sistema puesto que se puede realizar ajustes y mejoras basadas en los comentarios y necesidades reales de los clientes.
3. Pruebas automatizadas: XP promueve la escritura de las pruebas automatizadas para garantizar la calidad del código y las funcionalidades del sistema, auxiliando la detención de errores y reducción de tiempo de pruebas.
4. Flexibilidad ante cambios: XP aboga por abordar los cambios de manera ágil y efectiva, para el sistema esto sería fundamental ya que los requisitos y las fuentes de información pueden cambiar con el tiempo. Esta metodología permite adaptar y ajustar el sistema de manera más rápida y eficiente a medida que surgen nuevos desafíos o requerimientos.

Artefactos que genera la metodología:

1. Historias de usuario.
2. Plan de entrega.
3. Plan de iteraciones.
4. Tarjetas CRC.
5. Tareas de ingeniería.
6. Pruebas unitarias.
7. Pruebas de aceptación.

Roles mayormente usados:

1. Programador: Escribe las pruebas unitarias, estima el tiempo de desarrollo de cada actividad y produce el código del sistema.
2. Cliente: Es el responsable de establecer y dirigir el proyecto, así como de definir sus objetivos y metas. Es la figura clave que guía el proceso de desarrollo de software y tiene la responsabilidad de asegurarse de que el producto final cumpla con las necesidades y expectativas de los usuarios finales.
3. Encargado de pruebas (*Tester*): El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales.
4. Gestor (*Big boss*): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas.

I.3.2 Lenguaje de programación y Framework

Python

Como lenguaje de programación se seleccionó Python por ser de alto nivel y de propósito general, reconocido por su potencia y facilidad de aprendizaje, se destaca por ofrecer estructuras de datos eficientes y un enfoque simple pero efectivo para la programación orientada a objetos. La sintaxis de Python es concisa y su tipado es dinámico, lo que permite un desarrollo ágil y flexible.

La naturaleza interpretada de Python lo convierte en una opción adecuada para tareas de scripting y desarrollo rápido de aplicaciones en diversos dominios y entornos, su amplia disponibilidad en diferentes plataformas lo hace altamente compatible y versátil.

Python cuenta con un intérprete que puede ser fácilmente extendido mediante la incorporación de nuevas funcionalidades y tipos de datos implementados en lenguajes como C o C++. Esto brinda la posibilidad de aprovechar bibliotecas externas y optimizaciones de rendimiento.

Además, Python se puede utilizar como un lenguaje de extensión en aplicaciones personalizables, lo que permite integrar componentes específicos o adaptar el comportamiento de programas existentes [25].

Otros beneficios de utilizar Python en este proyecto son los siguientes:

- Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.
- Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa con menos líneas de código en comparación con muchos otros lenguajes.
- Cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea, de esta manera, los desarrolladores no tienen que escribir el código desde cero.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, pueden obtenerse soporte rápido de la comunidad.

- Hay muchos recursos útiles disponibles en internet para aprender Python, por ejemplo, pueden encontrarse con facilidad videos, tutoriales, documentación y guías para desarrolladores.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix [25].

Es versátil y eficiente, demostrando ser una herramienta útil para el desarrollo de webs complejas en menos líneas de código. Permitiendo que las webs sean más ligeras y optimizadas, lo que lleva a una mejor experiencia del usuario y una mayor eficiencia en el rendimiento.

Django: Es un framework web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático. Sigue la filosofía "baterías incluidas" lo que significa que viene con muchas funcionalidades genéricas ya implementadas y fáciles de personalizar. Es muy confiable y seguro, permite crear software robusto y escalable en un tiempo relativamente corto, es gratis y de código abierto [26].

Características:

1. Django ofrece una amplia funcionalidad integrada que facilita la gestión de tareas comunes en el desarrollo web. Desde la autenticación de usuarios hasta la administración de contenidos, pasando por la generación de mapas del sitio y canales RSS, Django proporciona una amplia variedad de componentes adicionales listos para usar. Esto significa que no se tienen que desarrollar estas funcionalidades desde cero, ya que Django brinda herramientas predefinidas que agilizan el proceso de desarrollo. Se puede aprovechar el sistema de autenticación completo, la interfaz de administración intuitiva, la generación automática de formularios, el motor de consultas ORM, el sistema de enrutamiento de URL y el sistema de plantillas, entre muchas otras características.
2. Enfoque riguroso en la seguridad: Django ofrece a los desarrolladores mecanismos para prevenir errores comunes de seguridad, como inyecciones SQL, ataques de *cross-site scripting* (XSS), falsificación de solicitudes entre sitios (CSRF) y manipulación de clics (clickjacking). El sistema de autenticación de

usuarios proporciona una forma segura de gestionar las cuentas y contraseñas de los usuarios.

3. Django presenta una alta escalabilidad, lo cual implica su capacidad para adaptarse de forma rápida y flexible a las demandas de tráfico más exigentes. Este framework está diseñado para manejar un alto volumen de solicitudes concurrentes, permitiendo escalar horizontalmente agregando más servidores y distribuyendo la carga de trabajo de manera eficiente
4. Comunidad activa y amplia documentación: Cuenta con una comunidad activa de desarrolladores y una amplia documentación. Pueden encontrarse tutoriales, guías, ejemplos de código y preguntas frecuentes que ayudarán en el desarrollo de la aplicación.

Además, Django ofrece la biblioteca Django *Test*, que facilita la realización de pruebas unitarias en el desarrollo de aplicaciones. Esta biblioteca proporciona un conjunto de herramientas y funcionalidades que permiten a los desarrolladores escribir pruebas exhaustivas y automatizadas para verificar el correcto funcionamiento de su código.

I.3.3 Lenguaje del lado del cliente

HTML 5 *HyperText Markup Language:* Es la nueva versión del lenguaje de marcado que se usa para estructurar páginas web, que surge como una evolución lógica de las especificaciones anteriores con los siguientes objetivos: separar totalmente la información, y la forma de presentarla, resumir, simplificar y hacer más sencillo el código utilizado, incorporar nuevas etiquetas semánticas, páginas compatibles con todos los navegadores web incluyendo los de los teléfonos móviles y otros dispositivos utilizados en la actualidad para navegar en internet. Se puede afirmar que ofrece mayores facilidades en relación con la accesibilidad de la web, se han incorporado nuevas etiquetas que aportan fundamentalmente mayor semántica al diseño de la página y otras que permiten incorporar elementos multimedia con menor dependencia de las tecnologías externas utilizadas hasta el momento. También se han incorporado nuevos eventos independientes del dispositivo y funciones JavaScript que permiten acceder directamente a los elementos de la página, permitiendo separar el diseño de los *scripts* del código y favoreciendo la programación de estos de forma compatible con las pautas de accesibilidad.

Características de este lenguaje:

- No es necesario ningún programa especial para crear una página web, gracias a ello se ha conseguido que se puedan crear páginas con cualquier ordenador y sistema operativo.
- Es un lenguaje descriptivo.
- Describe hipertexto, texto de forma estructurada y agradable.
- Permite inserciones multimedia [27].

Hojas de estilo en cascada (CSS): Con el crecimiento de internet y la aparición del lenguaje HTML para la creación de páginas web, el W3C demostró la necesidad de un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos creados, de manera que pudieran visualizarse de igual forma en cualquier navegador web. A partir de entonces surgieron entre varias propuestas los lenguajes CHSS (*Cascading HTML Style Sheets*) y *SSP (Stream-based Style Sheet Proposal*), el primero realizado por *Hakon Wium Lie* y el segundo por *Bert Bos*, que a finales de 1994 y 1995 se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta denominado CSS (*Cascading Style Sheets*). CSS es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo utilizado para especificar el aspecto de una página web, se basa en reglas que rigen el comportamiento del estilo de los elementos [28].

JavaScript: Consiste en un lenguaje de programación interpretado, que habitualmente se utiliza en sitios webs para ejecutar acciones del lado del cliente, estando dentro del código fuente de la página web. Técnicamente, constituye un dialecto del estándar *ECMAScript,* propuesto por la entidad internacional de estándares de información y comunicación ECMA *international* y diseñado inicialmente por *Netscape* y, posteriormente, por la Fundación Mozilla. También constituye un estándar ISO [29]. Debido a su propósito y uso general, todos los navegadores webs modernos interpretan correctamente JavaScript, siendo un lenguaje universal y multiplataforma [30].

Gestor de base de datos

PostgreSQL: Es un potente sistema de base de datos objeto-relacional de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan con seguridad las cargas de trabajo de datos más complicadas. Los orígenes de PostgreSQL se remontan a 1986 como parte del proyecto POSTGRES de la

Universidad de California en *Berkeley* y cuenta con más de 35 años de desarrollo activo en la plataforma central [31].

PostgreSQL viene con muchas características destinadas a ayudar a los desarrolladores a crear aplicaciones, a los administradores a proteger la integridad de los datos y crear entornos tolerantes a fallos y ayudarles a gestionar sus datos sin importar lo grande o pequeño que sea el conjunto de datos. Además de ser gratuito y de código abierto, PostgreSQL es altamente extensible, por ejemplo, puede definir sus propios tipos de datos, crear funciones personalizadas e incluso escribir código de diferentes lenguajes de programación sin tener que recompilar la base de datos [31].

PostgreSQL intenta ajustarse al estándar SQL siempre que dicha conformidad no contradiga las características tradicionales o pueda llevar a decisiones arquitectónicas erróneas. Muchas de las características requeridas por el estándar SQL están soportadas, aunque a veces con una sintaxis o función ligeramente diferente. Es de esperar que con el tiempo se produzcan nuevos avances hacia la conformidad. A partir del lanzamiento de la versión 15 en octubre de 2022, PostgreSQL cumple con al menos 170 de las 179 características obligatorias para la conformidad con SQL: 2016 Core [31].

Entorno de desarrollo

Visual Studio Code: Es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Incluye compatibilidad integrada con JavaScript, *TypeScript* y Node.js, y cuenta con un amplio ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, Go, .NET) [32].

Algunas de sus características son:

- Es gratuito y de código abierto.
- Es muy extensible, tiene un gran ecosistema de extensiones que permiten agregar casi cualquier funcionalidad.
- Tiene soporte integrado para CSS, SASS, Less, HTML, XML, SQL, PHP y soporte de extensión para casi cualquier lenguaje.
- Incluye características como *code completion, debugging*, separación de proyectos, control de versión (Git), resaltado de sintaxis.
- Es muy liviano y rápido, no requiere mucho espacio en disco ni muchos recursos del sistema para funcionar.

- Es ampliamente utilizado, tiene una gran comunidad de desarrolladores detrás, y es el editor oficial para muchos proyectos como Visual Studio Code, *TypeScript,* etc.

Herramienta CASE:

Visual Paradigm: Es un proveedor líder y mundialmente reconocido de soluciones de software de transformación empresarial. Permite a las organizaciones mejorar la agilidad empresarial y fomentar la innovación a través de estándares abiertos populares. Cuentan con la confianza de más de 320.000 usuarios en empresas que van desde pequeñas empresas y consultores hasta organizaciones de primera línea, universidades y unidades gubernamentales de todo el mundo. Además, permite a su equipo gestionar la complejidad de la transformación empresarial para hacer frente a la rápida evolución de los mercados, las tecnologías y los requisitos normativos. Es una solución integral, ideal para la planificación de la arquitectura empresarial y la transformación del negocio, la gestión de proyectos y el desarrollo ágil de software, para que su empresa pueda mantener el control y fomentar el crecimiento [33].

Algunas características de Visual Paradigm son [34]:

- Modelado visual: Potentes herramientas de modelado visual que le ayudan a construir y gestionar sus diagramas y elementos del modelo.
- Análisis y diseño empresarial: Herramientas empresariales integrales que le ayudan a mejorar la eficacia y productividad.
- Desarrollo ágil: Conjunto completo de herramientas de gestión de procesos y *backlog* ágiles que hace que el proyecto sea más eficaz.

Conclusiones del capítulo

Con el estudio y análisis de los sistemas para la gestión automatizada de correos electrónicos de alertas de información, se logró fundamentar las bases teóricas de la investigación y se analizaron los conceptos básicos asociados para un mayor entendimiento. Además, se pudieron adquirir los conocimientos previos sobre las características principales de estos sistemas, aunque no cumplen con todas las condiciones requeridas. La caracterización de las herramientas, tecnologías, metodología y lenguajes de programación

permitió conocer sus beneficios, así como formar las bases propicias para crear una propuesta de solución, que a su vez cumpla con el objetivo de la investigación.

CAPÍTULO II: ANÁLISIS Y DISEÑO DEL SISTEMA PARA LA GESTIÓN AUTOMATIZADA DE CORREOS ELECTRÓNICOS DE ALERTA DE INFORMACIÓN

En el presente capítulo se presentan las características de la propuesta de solución; para ello se definen las reglas de negocio. Además, se especifican las técnicas de obtención de requisitos, permitiendo precisar los requisitos funcionales y no funcionales. Se describe la arquitectura con el patrón arquitectónico utilizado, así como los patrones de diseño y de base de datos. Se representan los diagramas de clases del diseño y el modelo de base de datos

II.1 Descripción del sistema a desarrollar

Después de realizar una exhaustiva investigación sobre la problemática, se determinó que la mejor solución era desarrollar una aplicación web. Esta tiene como objetivo principal la gestión automatizada de correos electrónicos de alerta de información. Permitirá a los usuarios recibir notificaciones sobre temas específicos de su interés a través del correo electrónico. Para lograrlo, se implementará un proceso de autentificación en el cual los usuarios podrán entrar al sistema proporcionando un nombre de usuario y una contraseña, o registrarse en el sistema en caso de no tener una cuenta previa.

Una vez que los usuarios se autentican, son dirigidos a la página principal de la aplicación. En dicha página, tendrán la opción de agregar temas de alerta relacionados con palabras clave, frases o conceptos específicos. El sistema utilizará técnicas como web *scraping* para buscar similitudes entre los temas de alerta y la información recuperada de las fuentes de información seleccionadas por el usuario.

Cuando el sistema encuentre información relevante relacionada con los temas de alerta, enviará notificaciones a la dirección de correo electrónico asociada a la cuenta del usuario correspondiente. De esta manera, los usuarios estarán informados sobre los eventos pertinentes en tiempo real. Además, se ha considerado la funcionalidad de un rol de administrador que permitirá la gestión de usuarios. Estos tendrán acceso para ver, modificar, agregar o eliminar cuentas de usuario según sea necesario.

II.2 Levantamiento de requisitos

Utilizando la ingeniería de requerimientos se genera un mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, definir una solución razonable y generar la propuesta de solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional [35].

Funcionalidades del sistema

A través de las reuniones con el cliente, se lograron identificar 13 requisitos funcionales que se detallan en la siguiente tabla.

Tabla 2 Funcionalidades del sistema. Fuente: Elaboración propia

Nº	Nombre	Descripción	Prioridad	Complejidad
RF1	Autenticar usuario	Permite acceder al sistema.	Alta	Media
RF2	Cerrar sesión	Posibilita salir del sistema.	Alta	Media
RF3	Listar temas de notificación	Permite listar los temas de notificación añadidos.	Alta	Baja
RF4	Agregar temas de notificación	Permite agregar temas de notificación.	Alta	Media
RF5	Modificar temas de notificación	Posibilita modificar los temas de notificación añadidos.	Alta	Media
RF6	Eliminar temas de notificación	Posibilita eliminar los temas de notificación añadidos en el sistema.	Alta	Media
RF7	Anadir usuario	Permite registrar nuevos usuarios.	Alta	Media

RF8	Listar usuarios	Posibilita listar los usuarios añadidos en el sistema.	Alta	Baja
RF9	Eliminar usuario	Posibilita eliminar los usuarios añadidos en el sistema.	Alta	Media
RF10	Modificar usuarios	Posibilita modificar los usuarios añadidos.	Alta	Media
RF11	Realizar web *scraping*	Permite realizar web *scraping* en fuentes de información.	Alta	Alta
RF12	Realizar búsqueda de semejanza	Posibilita la busque de semejanza entre los temas de notificación y lo recuperado del *scraping.*	Media	Media
RF13	Enviar notificaciones	Permite enviar notificaciones.	Alta	Alta

Requisitos no funcionales

Son “requerimientos de calidad, que representan restricciones o las cualidades que el sistema debe tener tales como: precisión, usabilidad, seguridad, rendimiento, confiabilidad, performance entre otras” [36].

Usabilidad

RNF1. La aplicación debe estar basada en la web.

RNF2. La página debe tener un diseño sencillo con una paleta de colores claros (azul, blanco) y fácil de usar para usuarios con poco conocimiento en la computación.

RNF3. El sistema debe de estar concebido para cualquier tipo de usuario mayor de 12 años.

Software

Diseño e implementación:

RNF4. El sistema será desarrollado con el framework Django en su versión 4.2

RNF5. Se utilizará Python en su versión 3.11.2

Despliegue:

RNF6. El sistema debe ser configurado en un servidor de base de datos Postgresql 9.4

RNF7. La computadora del cliente debe poseer un navegador web como (Microsoft Edge en una versión igual o superior a la 12, Google Chrome con una versión igual o superior a 30, Mozilla Firefox en una versión 27 o superior, Safari en una versión 7 o superior u Opera en una versión igual o superior a la 17.)

Seguridad

RNF8. El acceso al sistema solo será posible después de haber sido autenticado mediante un nombre de usuario y una contraseña. Los datos introducidos serán validados para evitar inconsistencias.

RNF9. Los usuarios que pertenezcan a grupos con rol Cliente solo tendrán acceso a las plantillas principales y a las funcionalidades de la interfaz de inicio.

II.3 Historias de usuario

Las "Historias de usuarios" ("*User stories*") sustituyen a los documentos de especificación funcional, y a los "casos de uso". Estas "historias" son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios. Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia [37]. Véase el resto en los anexos de la investigación.

Tabla 3 Historias de usuario#1: Autenticar usuario. Fuente: Elaboración propia

Número:1	**Nombre**: Autenticar usuario
Usuario: cliente	

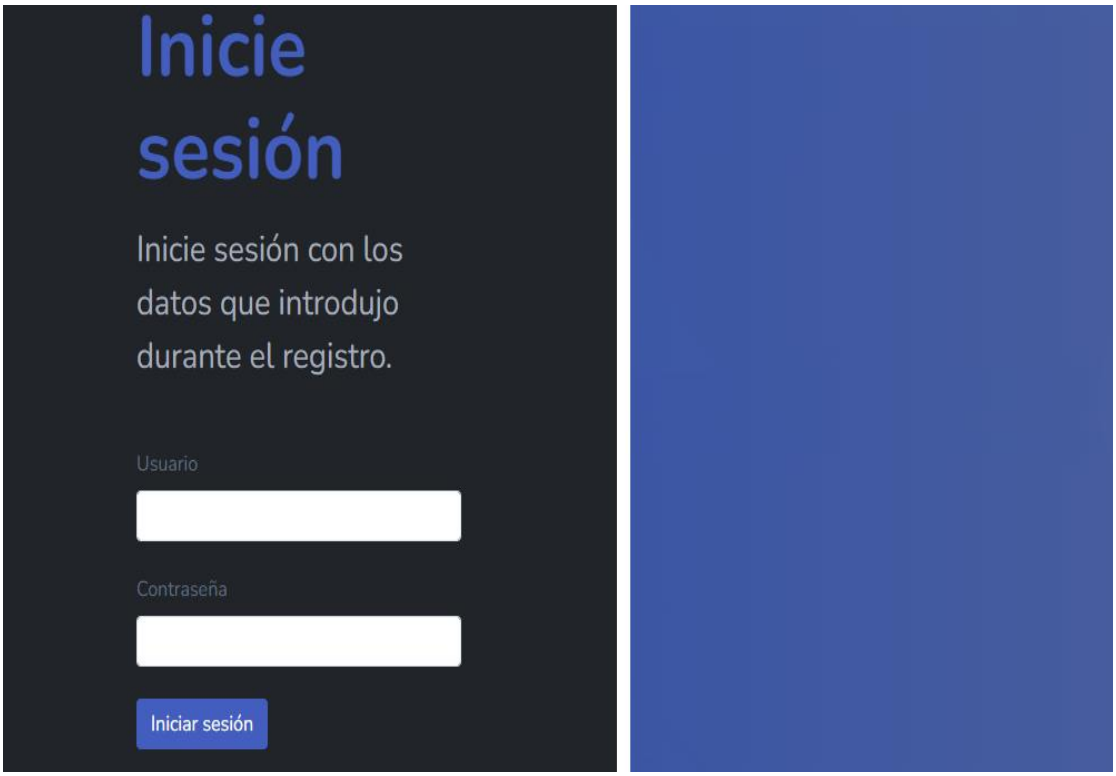
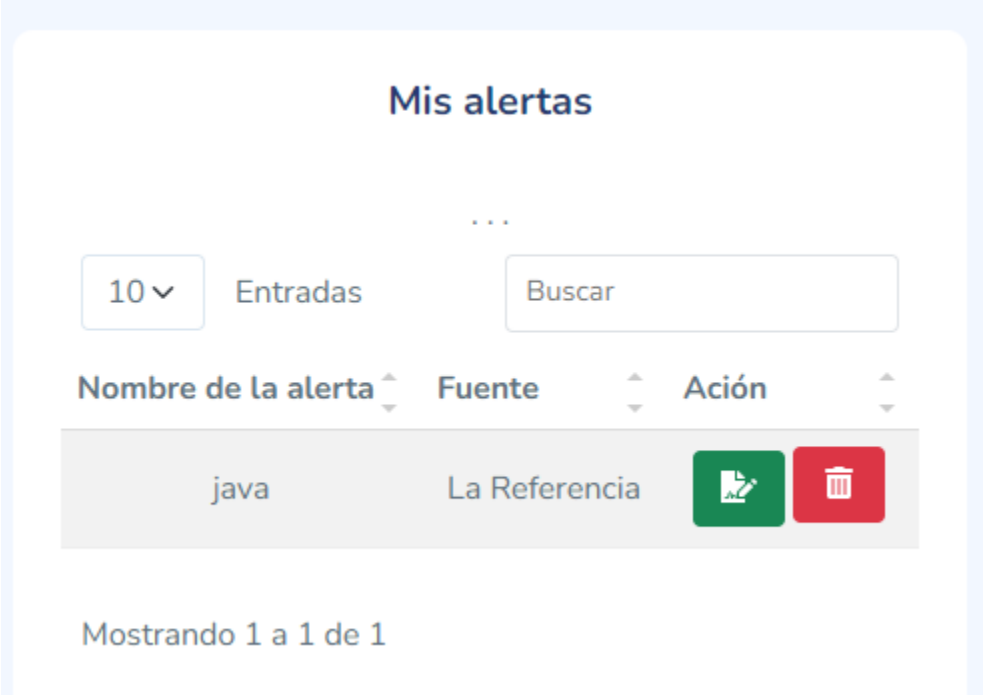
Prioridad en negocio: alta	**Riesgo de desarrollo**: medio
Puntos estimados:	**Iteraciones asignadas**:1
Programador Asignado: Daniel Felipe Fuentes	
Descripción: El sistema deberá permitir que los clientes se autentiquen en el mismo, ingresando su nombre de usuario y contraseña.	
Observaciones: No aplica	
Prototipo de Interfaz: 	

Tabla 4 Historias de usuario#2: Cerrar sesión. Fuente: Elaboración propia

Número:2	**Nombre**: Cerrar sesión
Usuario: cliente, administrador	
Prioridad en negocio: alta	**Riesgo de desarrollo**: medio
Puntos estimados:	**Iteraciones asignadas**:1
Programador Asignado: Daniel Felipe Fuentes	

Descripción: El sistema deberá permitir que los clientes cierren sesión
Observaciones: Los usuarios deben de estar ya autentificados.
Prototipo de Interfaz: No aplica

Tabla 5 Historias de usuario#3: Listar temas de notificación. Fuente: Elaboración propia

Número:3	**Nombre**: Listar temas de notificación
Usuario: cliente, administrador	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:2
Programador Asignado: Daniel Felipe Fuentes	
Descripción: Muestra un listado de todos los temas de notificación	
Observaciones: El usuario debe de estar ya autentificado.	
Prototipo de Interfaz: 	

II.4 Plan de entregas

Luego de haber establecido las Historias de Usuario se realiza la planificación de lanzamientos o entregas, como resultado de esto se establece el plan de entregas.XP se refiere a esta reunión como "Juego de planeamiento" *("Planning game"*), pero se puede utilizar cualquier denominación que sea más apropiada para el tipo de empresa y cliente, por ejemplo, "Reunión de planeamiento" (*"Planning meeting"*) o "Taller de planeamiento" (*"Planning workshop"*). El cronograma de entregas se basará en las estimaciones de tiempo de desarrollo proporcionadas por los desarrolladores, después de algunas iteraciones, se recomienda realizar otra reunión con los actores del proyecto para evaluar nuevamente el plan de entregas y realizar ajustes si es necesario [38].

Tabla 6 Plan de entrega. Fuente: Elaboración propia

Iteración	1	2	3
Cantidad de HU	6	4	4
Fecha de entrega	Abril 2023	Junio 2023	Agosto 2023

II.5 Plan de iteraciones

Es la práctica en donde el equipo establece el rumbo cada un par de semanas, las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración acuerdo al orden preestablecido [39].

Tabla 7 Plan de iteraciones. Fuente: Elaboración propia

Iteración	HU a implementar	Duración Total (Semanas)
1	1,2,7,8,9,10	8
2	3,4,5,6	8
3	11,12,13	8

II.6 Tarjetas de contenido, responsabilidad y colaboración (CRC)

Estas tarjetas ayudan a identificar y organizar fácilmente las clases más relevantes para los requerimientos de un sistema; además se dividen en tres secciones: en la parte superior se encuentra el nombre de la clase, en la parte izquierda se enlistan todas las acciones que la clase pueda realizar (responsabilidades) y en la parte derecha, otras clases que interactúen con la actual (colaboradores), ayudándola a llevar a cabo sus responsabilidades [40].

Tabla 8 Tarjeta CRC#1: Usuario. Fuente: Elaboración propia

Tarjetas CRC	
Usuario	
Responsabilidad	**Colaboración**
• Agregar usuarios • Modificar usuarios • Listar usuarios • Eliminar usuarios	Temas

Tabla 9 Tarjeta CRC#2: Temas. Fuente: Elaboración propia

Tarjetas CRC	
Temas	
Responsabilidad	**Colaboración**
• Agregar tema • Modificar tema • Listar tema • Eliminar tema	Usuario

Tabla 10 Tarjeta CRC#3: Web scraping. Fuente: Elaboración propia

Tarjetas CRC	
Web *scraping*	
Responsabilidad	**Colaboración**

• Realizar web *scraping* a la url • Notificar sobre temas de información encontrados en la web	Temas

II.7 Arquitectura del software

La arquitectura de un software o sistema de cómputo, es la estructura que comprende a los componentes del mismo, sus propiedades externas visibles y las relaciones entre ellos. Es una representación que permite analizar la efectividad del diseño para cumplir los requerimientos establecidos, considerar alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil y reducir los riesgos asociados con la construcción del software [41].

La arquitectura de software es la representación de alto nivel de la estructura de un sistema, describe las partes que la integran, las interacciones entre ellas, los patrones que supervisan su composición, y las restricciones de aplicar esos patrones. La arquitectura de software conforma la columna vertebral de cualquier sistema y constituye uno de sus principales atributos de calidad [42]. El documento de IEEE Std 1471-2000 [43] define: “La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.”

Arquitectura Cliente-Servidor

El modelo cliente-servidor como se evidencia en la figura 1 es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores donde lo habitual es que un servidor sea una máquina bastante potente con un hardware y software específico que actúa de depósito de datos y funcione como un sistema gestor de base de datos o aplicaciones y los demandantes, llamados clientes; los cuales realizan peticiones a una o varias aplicaciones servidores, que deben encontrarse en ejecución para atender dichas demandas [44].

La interacción entre un cliente y un servidor en un sistema informático se caracteriza por un patrón de solicitud y respuesta, el cual debe seguir un protocolo de comunicación establecido que defina el lenguaje, las reglas y los patrones de diálogo utilizados. Esta relación,

conocida como modelo cliente-servidor, se rige por un conjunto de protocolos de transmisión de datos basados en TCP/IP.

La correcta implementación de este modelo de comunicación es esencial para el correcto funcionamiento de sistemas informáticos complejos, ya que permite una adecuada transferencia de información entre el cliente y el servidor, asegurando la consistencia y la integridad de los datos transmitidos.

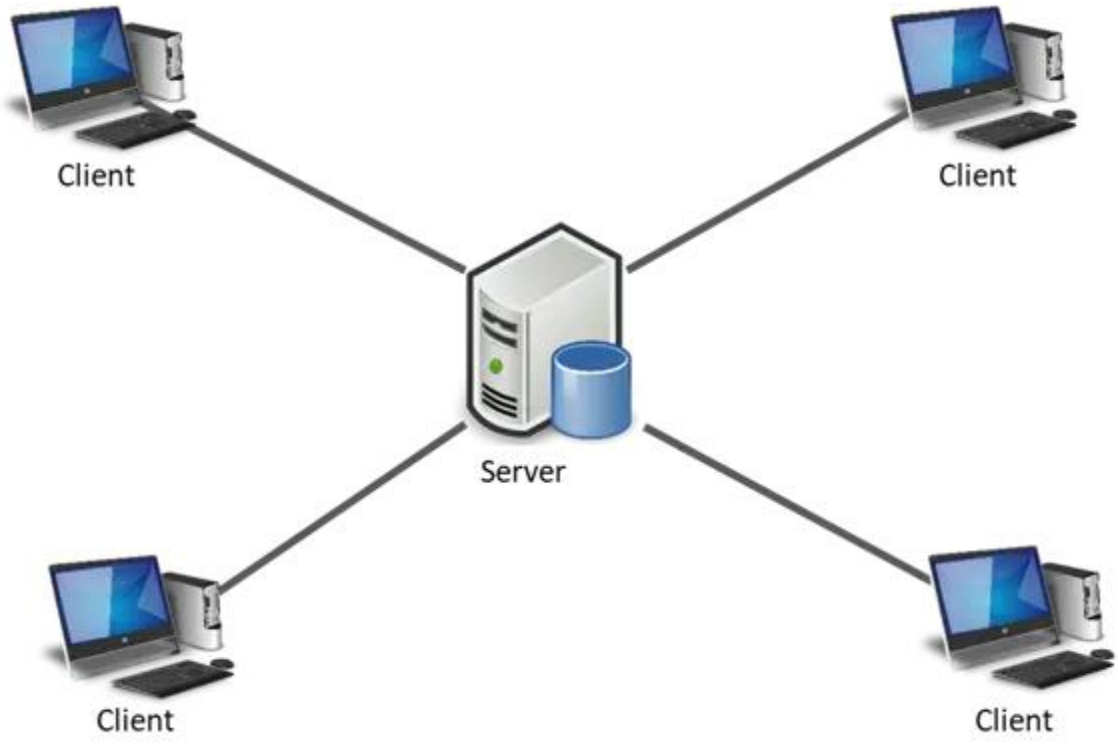


Ilustración 1 Arquitectura cliente servidor [45]

II.8 Patrones de arquitectura

El patrón de arquitectura de software Modelo-Vista-Plantilla (MVT) es ampliamente utilizado en el desarrollo de aplicaciones web, y es especialmente conocido por ser el patrón utilizado por el framework de desarrollo web Django, creado con el lenguaje de programación Python. Este patrón se divide en tres componentes principales que se encargan de tareas específicas:

- Modelo (Model): Es el componente encargado de manejar la lógica de negocio y los datos de la aplicación. Estas se pueden evidenciar en las clases Usuario,

Alerta, Correo_notificacion que se encuentra en el archivo model.py del sistema. Aquí se define la estructura de los datos, su validación y las relaciones entre ellos.

- Vista (View): Es el componente encargado de manejar la interacción con el usuario y la presentación de la información. En el caso de la aplicación se implementaron vistas basadas en función como son RegistrarCiente, LoginView, Anadir_Alerta: Aquí se define la lógica para procesar las solicitudes del usuario, se realiza la recuperación de los datos desde el modelo y se selecciona la plantilla adecuada para mostrar la información al usuario.
- Plantilla (Template): Es el componente encargado de la presentación visual de la información al usuario. En la aplicación web se utilizan las plantillas index, login, modificar_usuario. Aquí se define la estructura y el estilo de las páginas web, utilizando lenguajes de marcado como HTML, CSS y JavaScript.

El patrón MVT es muy eficiente y permite a los desarrolladores separar la lógica de negocio, la presentación y la gestión de datos en componentes independientes, lo que facilita la escalabilidad, la reutilización de código y el mantenimiento del software a largo plazo.

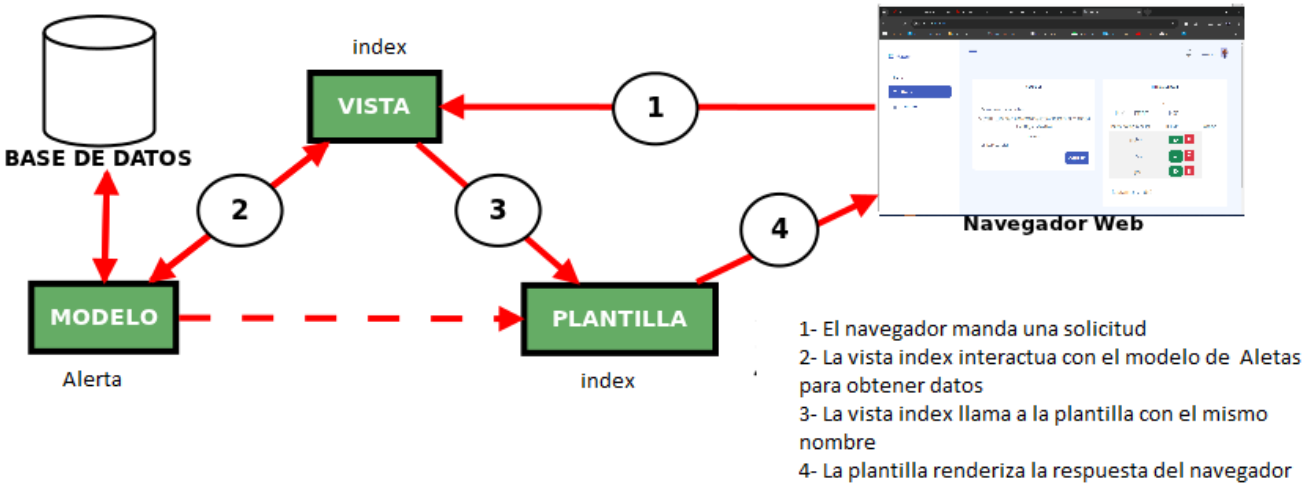


Ilustración 2 Patrón modelo vista plantilla [46]

II.9 Patrones de diseño

Son valorados debido a que imponen reglas sobre la arquitectura y expresan esquemas para solucionar problemas de un mismo tipo que pueden presentarse durante el desarrollo de la aplicación. Constituyen la base para realizar la búsqueda de soluciones a problemas que se

presentan en el desarrollo de software y otros marcos del diseño de interacción. Una solución es considerada un patrón de diseño cuando posee ciertas características, entre las cuales se encuentran: su efectividad debe haberse comprobado resolviendo problemas similares en otras ocasiones, debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias [41].

Patrones GRASP:

GRASP es el acrónimo de *General Responsibility Asignment Software Patterns* (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades [45], entre los que encontramos:

- Experto: Es el encargado de asignar una responsabilidad al experto en información, es decir la clase que cuenta con la información necesaria para cumplir la responsabilidad. Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Como se puede observar en la clase MyUserManager, a la cual se le asignó la responsabilidad de crear a los usuarios del sistema y a los administradores tomando información de la clase Usuario ya que tiene referencia a esta.

```python
class MyUserManager(BaseUserManager):
    def create_user(self, username, correo,  password, nombre, rol ):
        if not username:
            raise ValueError('El correo es obligatorio')

        user = self.model(correo=self.normalize_email(correo), username=username, nombre=nombre, rol=rol)
        user.set_password(password)
        user.save(using=self._db)
        return user


    def create_superuser(self, username, correo,  password, nombre, rol):
        user= self.create_user(username, correo,  password, nombre, rol )
        user.is_staff = True
        user.is_superuser = True
        user.save(using=self._db)
        return user

    def create_staff(self, username, correo,  password, nombre, rol):
        user= self.create_user(username, correo,  password, nombre, rol )
        user.is_staff = True
        user.save(using=self._db)
        return user
```

Ilustración 3 Patrón experto en la clase MyUserManager. Fuente: Elaboración propia

Bajo acoplamiento: Se encarga de asignar una responsabilidad para mantener bajo acoplamiento. Es decir, estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

- Al utilizar el patrón de bajo acoplamiento se soporta el diseño de clases más independientes, reduciendo el impacto de los cambios, y siendo más reutilizables. No puede considerarse en forma independiente de otros patrones como experto o alta cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades. Utilizado en las clases que no depende de muchas otras.

Alta Cohesión: Se encarga de asignar una responsabilidad de modo que la cohesión siga siendo alta. Una regla práctica de este patrón es la siguiente: una clase con mucha cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande.

- Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el

mantenimiento y los mejoramientos. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización. Aplicados en clase que realizan pocas funciones.

Patrones GoF:

Los patrones de diseño de software son un conjunto de artefactos que encapsulan el conocimiento de los problemas de diseño que ocurren en un contexto particular. Se han propuesto varios patrones de software para proporcionar soluciones a problemas de diseño recurrentes. *Gang of Four* (GoF) clasifica estos patrones en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento [46].

Los patrones estructurales existentes son los siguientes: adaptador, compuesto, decorador, fachada. A continuación, se describen los utilizados:

Decorador (Decorator): Adjunta responsabilidades adicionales a un objeto de forma dinámica. Proporciona una alternativa flexible a la creación de subclases para ampliar la funcionalidad. Se evidencia en las vistas logout_view, Anadir_Alerta, Eliminar_Usuario, Anadir_Usario en las cuales se utiliza el operador login_required. Este otorga la funcionalidad de restricción de acceso sin tener que escribir manualmente el código de verificación de inicio de sesión para cada vista.

```
@login_required
def logout_view(request):
    logout(request)
    return redirect('/')
```

Ilustración 4 Patrón decorador login_requierd. Fuente: Elaboración propia

Por último, los patrones de comportamiento existentes son los siguientes: intérprete, iterador, mediador, recuerdo, observador y visitante.

A continuación, se describen los utilizados:

Observador (Observer): Define una dependencia de uno a muchos entre objetos de modo que cuando un objeto cambia de estado, todos sus dependientes son notificados y

actualizados automáticamente[47]. Evidenciándose en la clase Alerta la cual tiene una relación de muchos a muchos con la clase Usuario.

```python
class Alerta(models.Model):
    usuario=models.ForeignKey(Usuario, on_delete=models.CASCADE)
    alerta=models.CharField(max_length=50, verbose_name="Alerta")
    fecha_alerta=models.DateTimeField(default=timezone.now)

    def __str__(self):
        fila=self.alerta
        return fila
```

Ilustración 5 Relación de muchos a muchos entre alerta y usuarios. Fuente: Elaboración propia

II.10 Modelo de datos

Un modelo de datos relacional es una estructura que representa objetos, sus propiedades y las relaciones entre ellos, con el objetivo de reflejar un fenómeno de la realidad objetiva. Este tipo de modelo permite establecer la conexión entre el mundo real y la información almacenada físicamente en una base de datos[48].

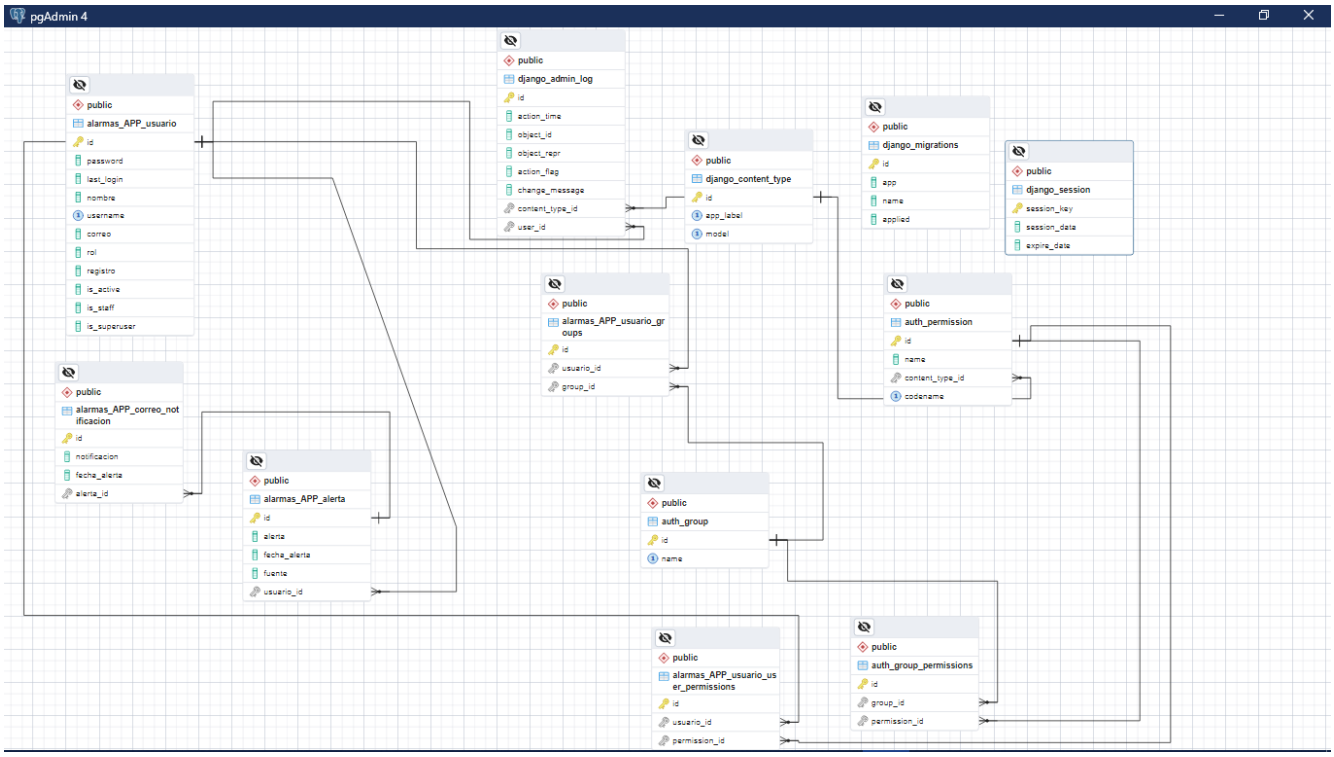


Ilustración 6 Modelo de datos del sistema. Fuente: Elaboración propia

II.11 Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado (UML) que se utiliza para modelar la disposición física de los artefactos software en nodos. Muestra la arquitectura del sistema como el despliegue de los artefactos de software a los objetivos de despliegue [49].

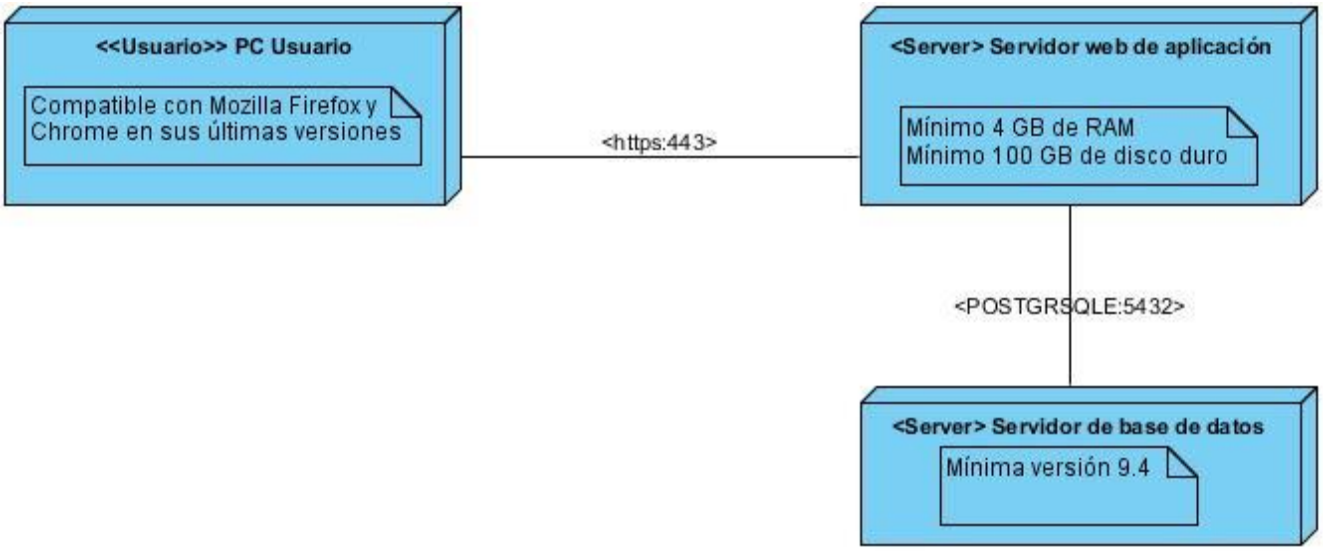


Ilustración 7 Diagrama de despliegue. Fuente: Elaboración propia

Nombre del procesador: descripción de la funcionalidad y capacidad del nodo.

Nodo: Elementos de procesamiento con al menos un procesador, memoria

Ejemplos:

- **PC_Usuario:** Computadora que es utilizada por una persona para acceder a servicios y aplicaciones proporcionados por una red o servidor. Es comúnmente utilizada para realizar tareas como navegación en internet, procesamiento de texto y correo electrónico. La computadora del cliente debe poseer un navegador web como (Microsoft Edge en una versión igual o superior a la 12, Google Chrome con una versión igual o superior a 30, Mozilla Firefox en una versión 27 o superior, Safari en una versión 7 o superior u Opera en una versión igual o superior a la 17).
- **PC_Servidora**: Computadora que proporciona servicios a otras computadoras en una red. Estos servicios pueden incluir almacenamiento compartido de

archivos, impresoras, correo electrónico, bases de datos y aplicaciones. La PC servidora es responsable de administrar y controlar el acceso a estos servicios. Es necesario contar con al menos 4GB de RAM y 100GB de disco duro para poder procesar toda la información y almacenarla.

- **PC_ServidoresBD**: Computadora que almacena y gestiona grandes cantidades de datos. Estos datos pueden ser accedidos y manipulados por otros programas o aplicaciones, como una aplicación web que utiliza una base de datos para almacenar información. Una PC base de datos también puede ser utilizada para realizar análisis de datos y generar informes. Se necesita que el servidor de base de datos sea una versión igual o superior a la 9.4 de Postgresql.

Nombre del tipo de conexión: Características físicas de la conexión.

Conectores: Expresa el tipo de conector o protocolo que relaciona dos elementos.

Ejemplos:

Https:443: Protocolo seguro de transferencia de hipertexto (HTTPS) para los servicios del World Wide Web (WWW).

PostgreSQL: 5432: Permite que las aplicaciones cliente se comuniquen con el servidor de PostgreSQL a través de la red.

Conclusiones del capítulo

En este capítulo se definieron los requisitos tanto funcionales como no funcionales que sustentan la propuesta de solución planteada. Durante el flujo de trabajo de Análisis y Diseño propuesto por la metodología de desarrollo eXtreme Programming-XP, se analizaron y realizaron las historias de usuario como principal artefacto. La definición de la arquitectura del sistema mediante el patrón arquitectónico MVT permitió delimitar la vista, en la cual se construye las interfaces gráficas de usuarios, la lógica del negocio para la definición de los controladores que gestionan las operaciones de los componentes, y el modelo para la persistencia de la información. Además se elaboró el modelo de datos con que contará la solución.

CAPÍTULO III: VALIDACIÓN DEL SISTEMA PARA LA GESTIÓN AUTOMATIZADA DE CORREOS ELECTRÓNICOS DE ALERTA DE INFORMACIÓN

En este segmento, se describe en detalle las iteraciones ejecutadas durante la fase de elaboración de la solución sugerida. Se presentan también las actividades de ingeniería establecidas para cada historia de usuario que se determinó, así como los test de aprobación que se planificaron para el sistema.

III.1 Tareas de ingeniería

Las tareas de ingeniería se pueden describir mediante un lenguaje técnico y no necesariamente entendible por el cliente. Su objetivo consiste definir cada una de las actividades que dan cumplimiento a las HU, haciendo más entendible las funciones del sistema y facilitando su construcción. Para una mayor organización, se definen en correspondencia con las iteraciones definidas como se muestra a continuación [50].

Iteración 1

En la primera iteración se encuentran las funcionalidades relacionadas con el usuario ya sea leer, crear, eliminar o modificar, así como la autentificación y el cierre de sesión. Las tareas definidas son:

- Tarea No.1: Implementar la funcionalidad que permita validar los datos ingresados.
- Tarea No.2: Implementar la funcionalidad que permita autentificar un usuario.
- Tarea No.3: Implementar la funcionalidad que permita cerrar sesión al usuario
- Tarea No.4: Implementar la funcionalidad que permita agregar usuarios
- Tarea No.5: Implementar la funcionalidad que permita listar usuarios.
- Tarea No.6: Implementar la funcionalidad que permita eliminar un usuario.
- Tarea No.7: Implementar la funcionalidad que permita modificar un usuario.

El resto puede ser consultado en los anexos de la investigación

Tabla 11 Tarea de ingeniería: funcionalidad que permita autentificar un usuario. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 1	Nombre de la HU: Autenticar usuario
Nombre de la tarea: Implementar la funcionalidad que permita validar los datos ingresados.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 3 de abril 2023	Fecha fin: 12 de abril 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que valide los datos ingresados por el usuario.	

Tabla 12 Tarea de ingeniería: Implementar funcionalidad que permita autentificar un usuario. Fuente: Elaboración propia

Tarea de ingeniería	
Número de la tarea: 2	Nombre de la HU: Autenticar usuario
Nombre de la tarea: Implementar la funcionalidad que permita autentificar un usuario.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 13 de abril 2023	Fecha fin: 23 de abril 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que identifique si el usuario y la contraseña ingresados son los almacenados en la base de datos.	

Tabla 13 Tarea de ingeniería: Implementar funcionalidad que permita cerrar sesión al usuario. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 3	Nombre de la HU: Cerrar sesión
Nombre de la tarea: Implementar la funcionalidad que permita cerrar sesión al usuario.	

Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 24 de abril 2023	Fecha fin: 3 de mayo 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita al usuario cerrar sesión.	

Iteración 2

En la segunda iteración se encuentra las funcionalidades relacionadas con los temas de notificación el cual el usuario puede modificar, eliminar, agregar o visualizar. Las tareas definidas son:

- Tarea No.8: Implementar la funcionalidad que permita listar temas de notificación.
- Tarea No.9: Implementar la funcionalidad que permita agregar temas de notificación.
- Tarea No.10: Implementar la funcionalidad que permita modificar temas de notificación.
- Tarea No.11: Implementar la funcionalidad que permita eliminar temas de notificación.

Tabla 14 Tarea de ingeniería: Implementar la funcionalidad que permita Listar temas de notificación. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 8	Nombre de la HU: Listar temas de notificación
Nombre de la tarea: Implementar la funcionalidad que permita listar los temas de notificación.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 13 de junio 2023	Fecha fin: 27 de junio 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita listar los temas de las notificaciones de correo para cada usuario.	

Tabla 15 Tarea de ingeniería: Implementar la funcionalidad que permita agregar temas de notificación. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 9	Nombre de la HU: Agregar temas de notificación
Nombre de la tarea: Implementar la funcionalidad que permita agregar temas de notificación.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 28 de junio 2023	Fecha fin: 11 de julio 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita agregar los temas de las notificaciones de correo para cada usuario.	

Tabla 16 Tarea de ingeniería: Implementar la funcionalidad que permita modificar temas de notificación. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 10	Nombre de la HU: Modificar temas de notificación
Nombre de la tarea: Implementar la funcionalidad que permita modificar temas de notificación.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 12 de julio 2023	Fecha fin: 26 de julio 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita modificar una notificación agregada.	

Iteración 3

En esta iteración se encuentran las funcionalidades que se encargaran de recolectar la información, compararla y enviar notificaciones de acuerdo a lo encontrado. Las tareas definidas son:

- Tarea No.12: Implementar la funcionalidad que permita realizar web *scraping*.
- Tarea No.13: Implementar la funcionalidad que permita búsqueda de semejanza.
- Tarea No.14: Implementar la funcionalidad que permita enviar notificaciones.

Tabla 17 Tarea de ingeniería: Implementar la funcionalidad que permita realizar web scraping. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 12	Nombre de la HU: Realizar web *scraping*
Nombre de la tarea: Implementar la funcionalidad que permita realizar web *scraping.*	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 9 de agosto 2023	Fecha fin: 23 de agosto 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que mediante un url de una fuente de información realice web *scraping.*	

Tabla 18 Tarea de ingeniería: Implementar la funcionalidad que permita búsqueda de semejanza. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 13	Nombre de la HU: Búsqueda de semejanza
Nombre de la tarea: Implementar la funcionalidad que permita búsqueda de semejanza.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 23 de agosto 2023	Fecha fin: 6 de septiembre 2023

Programador responsable: Daniel Felipe Fuentes
Descripción: Implementar un método que mediante los datos obtenidos por la web *scraping* realice una búsqueda de semejanza con los temas de notificación.

Tabla 19 Tarea de ingeniería: Implementar la funcionalidad que permita enviar notificaciones. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 14	Nombre de la HU: Enviar notificaciones
Nombre de la tarea: Implementar la funcionalidad que permita enviar notificaciones.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 7 de septiembre 2023	Fecha fin: 20 de septiembre 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que obtenga los datos del usuario y enviar notificaciones en caso de que se encuentre una nueva actualización sobre los temas.	

III.2 Pruebas de unitarias

La prueba unitaria es una prueba de caja blanca que se encarga de verificar el código del programa y es desarrollada por el programador. Cada desarrollador debe probar continuamente los resultados obtenidos durante la implementación del sistema para garantizar que la funcionalidad requerida por el cliente se implemente correctamente. Estas pruebas se realizan cada vez que se completa una función directamente en el entorno real [51].

Para la elaboración de las pruebas unitarias, se desarrolló en el entorno de desarrollo integrado Visual Studio Code la clase TestCase de Django, la cual contiene los métodos de prueba implementados. En la Figuras siguiente se ilustra un fragmento del código desarrollado con el propósito de validar el correcto funcionamiento del sistema y garantizar que la salida corresponda con los valores esperados. Consultar el resto en los anexos de la investigación.

Implementación de la clase TestCase para poder realizar las pruebas.

```python
from django.shortcuts import get_object_or_404
from django.test import TestCase, Client
from django.urls import reverse
from django.core import mail
from alarmas_APP.models import Usuario , Alerta
from .views import Scraping
# Create your tests here.

class UsuarioTestCase(TestCase):
    def setUp(self):
        self.client = Client()
        self.useradmin = Usuario.objects.create(
            nombre='Daniel',
            username='daniel',
            correo='df241441@gmail.com',
            rol='Administrator',
            is_superuser=True
        )
        self.userclient = Usuario.objects.create(
            nombre='Daniel',
            username='danielf',
            correo='df24144131@gmail.com',
            rol='Cliente',

        )
```

Ilustración 8 Clase TestCase*. Fuente: Elaboración propia*

Pruebas para verifica si se están creando correctamente los usuarios con el rol administrador.

```python
#Prueba para crear un usuario administrador
def test_user_create_admin(self):
    self.assertEqual (self.useradmin.is_superuser, True)
    self.assertEqual (self.useradmin.is_staff, False)
    self.assertEqual (self.useradmin.is_active, True)
```

Ilustración 9 Función test_user_.create_admin. Fuente: Elaboración propia

Prueba con el cual se verifica si se están creando correctamente los usuarios con el rol cliente.

```python
#Prueba para crear un usuario cliente
def test_user_create_client(self):
    self.assertEqual (self.userclient.is_superuser, False)
    self.assertEqual (self.userclient.is_staff, False)
    self.assertEqual (self.userclient.is_active, True)
```

Ilustración 10 Función test_user_.create_client. Fuente: Elaboración propia

Prueba con la cual se asegura si el sistema está autentificando de manera correcta a los usuarios con el rol de administrador.

```
#Prueba para iniciar sesion del administrador
 def test_Login_admin(self):
     self.useradmin.set_password('password')
     self.useradmin.save()
     response =self.client.login( username ='daniel', password ='password')
     self.assertEqual(response, True)
```

Ilustración 11 Función test_Login_admin. Fuente: Elaboración propia

La figura 23 exhibe los resultados satisfactorios de las pruebas ejecutadas, evidenciando que en todas las iteraciones se obtuvo el valor esperado, el cual consiste en un identificador único de la acción a ejecutar por el agente, de entre todas las acciones factibles.

```
PS C:\Users\Kira\Desktop\alarmaaaaaaaaaaaaaaaaaaaaaaaaaa\alarmas> py manage.py test
Found 14 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..............
----------------------------------------------------------------------
Ran 14 tests in 26.244s

OK
Destroying test database for alias 'default'...
```

Ilustración 12 Resultado de las pruebas Fuente: Elaboración propia

Se realizaron 14 pruebas unitarias, abarcando diferentes aspectos y funcionalidades del código implementado. Los resultados obtenidos en todas las pruebas fueron exitosos, validando así la funcionalidad correcta y la estabilidad del código. Estas pruebas demostraron una cobertura completa de los requerimientos establecidos, identificando tempranamente posibles errores y garantizando un software confiable y de calidad. Además, el éxito en las pruebas unitarias ahorra tiempo y recursos al prevenir problemas futuros, proporcionando una base sólida para iteraciones y mejoras continuas.

III.3 Pruebas de aceptación

Las pruebas de aceptación son una parte integral del desarrollo incremental. Todas las historias de usuarios están respaldadas por pruebas de aceptación, que son definidas por el cliente, significan la satisfacción del mismo con el producto desarrollado. Estas pruebas obligan

al cliente a profundizar en el conocimiento de su dominio y declarar con precisión qué debe hacer la aplicación en circunstancias específicas, por esto, el cliente es la persona adecuada para diseñarlas. En efecto, las pruebas de aceptación marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo la dirección a seguir, así como los puntos o funcionalidades en que se debe poner el mayor esfuerzo y atención [52].

A continuación, se especifican algunas de las pruebas de aceptación realizadas al sistema:

Iteración 1

Véase el resto en los anexos de la investigación.

Tabla 20 Prueba de aceptación#1: Validar credenciales. Fuente: Elaboración propia

Caso de prueba de aceptación	
Código: HUV_P1	**Historia de usuario:** Autenticar usuario
Nombre: Validar credenciales	
Descripción: Prueba para comprobar que los datos introducidos son correctos.	
Condiciones de ejecución: El usuario debe introducir sus credenciales.	
Pasos de ejecución: El usuario debe escribir su nombre de usuario y su contraseña.	
Resultados esperados: De ser correctos los datos se procede a comprobar que el usuario exista.	
Evaluación de la prueba: Satisfactorio	

Tabla 21 Prueba de aceptación#2: Validar credenciales P2. Fuente: Elaboración propia

Caso de prueba de aceptación	
Código: HUV_P2	**Historia de usuario:** Autenticar usuario
Nombre: Validar credenciales	

Descripción: Prueba para comprobar que los datos introducidos son correctos.
Condiciones de ejecución: El usuario debe introducir sus credenciales.
Pasos de ejecución: El usuario debe escribir su nombre de usuario y su contraseña.
Resultados esperados: De ser incorrectos los datos, se muestra un mensaje “Existen campos incorrectos”.
Evaluación de la prueba: Satisfactorio

Tabla 22 Prueba de aceptación#3: Autenticar usuario. Fuente: Elaboración propia

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** Autenticar usuario
Nombre: Autenticar usuario	
Descripción: Comprueba que el usuario introducido sea un usuario válido.	
Condiciones de ejecución: El usuario debe introducir sus credenciales.	
Pasos de ejecución: El usuario debe escribir su nombre de usuario y su contraseña.	
Resultados esperados: Si los datos introducidos por el usuario coinciden con la base de datos de usuarios se accede a la página principal.	
Evaluación de la prueba: Satisfactorio	

Tabla 23 Prueba de aceptación#4: Autenticar usuario P2. Fuente: Elaboración propia

Caso de prueba de aceptación	
Código: HU1_P2	**Historia de usuario:** Autenticar usuario
Nombre: Autenticar usuario	
Descripción: Comprueba que el usuario introducido sea un usuario válido.	

Condiciones de ejecución: El usuario debe introducir sus credenciales.
Pasos de ejecución: El usuario debe escribir su nombre de usuario y su contraseña.
Resultados esperados: De ser incorrectos los datos, se muestra el mensaje “El usuario no existe”.
Evaluación de la prueba: Satisfactorio

Tabla 24 Prueba de aceptación#5: Cerrar sesión. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** Cerrar sesión
Nombre: Cerrar sesión	
Descripción: Prueba para ver si el usuario puede cerrar la sesión.	
Condiciones de ejecución: El usuario debe de estar ya autentificado.	
Pasos de ejecución: El usuario debe de presionar el botón en el menú desplegable de las opciones de este.	
Resultados esperados: Una vez presionado el botón se cierra la sesión y se carga la página de autentificación.	
Evaluación de la prueba: Satisfactorio	

Iteración 2

Tabla 25 Prueba de aceptación#13: Listar temas de notificación. Fuente: Elaboración propia

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** Listar temas de notificación
Nombre: Listar temas de notificación	
Descripción: Comprueba si se puede listar todos los temas de notificación del usuario.	

Condiciones de ejecución: El usuario debe de estar ya autentificado y tener temas ya ingresados.
Pasos de ejecución: Al iniciar sesión se mostrará la página de inicio con la lista de temas.
Resultados esperados: Si se tiene temas registrados se muestra una tabla con todos los temas ingresados por el usuario ordenados por la fecha de adición.
Evaluación de la prueba: Satisfactorio

Tabla 26 Prueba de aceptación#14: Listar temas de notificación P2. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU3_P2	**Historia de usuario:** Listar temas de notificación
Nombre: Listar temas de notificación	
Descripción: Comprueba si se puede listar todos los temas de notificación del usuario.	
Condiciones de ejecución: El usuario debe de estar ya autentificado y tener temas ya ingresados.	
Pasos de ejecución: Al iniciar sesión se mostrará la página de inicio con la lista de temas.	
Resultados esperados: Si no se tienen temas registrados se muestra un mensaje en la tabla donde diga “Lista de temas vacío”.	
Evaluación de la prueba: Satisfactorio	

Tabla 27 Prueba de aceptación#15: Agregar temas de notificación

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** Agregar temas de notificación.

Nombre: Agregar temas de notificación.
Descripción: Comprueba si se puede agregar un tema del que se quiera una notificación.
Condiciones de ejecución: El usuario debe de estar ya autentificado.
Pasos de ejecución: Al iniciar sesión se mostrará la página de inicio un label donde se escribirá el tema que se guardará en la base de datos.
Resultados esperados: Se recarga la página y se agrega el tema a la lista de temas.
Evaluación de la prueba: Satisfactorio

Tabla 28 Prueba de aceptación#16: Modificar temas de notificación. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de usuario:** Modificar temas de notificación.
Nombre: Modificar temas de notificación.	
Descripción: Comprueba si se puede modificar los temas de notificación.	
Condiciones de ejecución: El usuario debe de estar ya autentificado y tener temas agregados.	
Pasos de ejecución: Al iniciar sesión se mostrará la página de inicio con una tabla con los temas y un botón para modificarlos.	
Resultados esperados: Se recarga la página y se agrega el tema modificado a la lista de temas.	
Evaluación de la prueba: Satisfactorio	

Tabla 29 Prueba de aceptación#17: Eliminar temas de notificación. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de usuario:** Eliminar temas de notificación.

Nombre: Eliminar temas de notificación.
Descripción: Comprueba si se puede eliminar los temas de notificación.
Condiciones de ejecución: El usuario debe de estar ya autentificado y tener temas agregados.
Pasos de ejecución: Al iniciar sesión se mostrará la página de inicio con una tabla con los temas y un botón para eliminarlos.
Resultados esperados: Se recarga la página y se elimina el tema seleccionado.
Evaluación de la prueba: Satisfactorio

Iteración 3

Tabla 30 Prueba de aceptación#18: Web scraping. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU11_P1	**Historia de usuario:** Realizar web scraping.
Nombre: Web scraping	
Descripción: Comprueba si se puede realizar web scraping.	
Condiciones de ejecución: Que tenga conexión.	
Pasos de ejecución: Se hace de forma automática cada cierto tiempo.	
Resultados esperados: Se extrae y recopila los datos de una fuente de información en específico.	
Evaluación de la prueba: Satisfactorio	

Tabla 31 Prueba de aceptación#19: Búsqueda de semejanza. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU12_P1	**Historia de usuario:** Búsqueda de semejanza.
Nombre: Búsqueda de semejanza	

Descripción: Comprueba si se puede realizar una búsqueda de semejanza entre los datos obtenidos por el *scraping* y los temas de notificación. .
Condiciones de ejecución: Que tenga conexión.
Pasos de ejecución: Se ejecuta después de realizar el *scraping*.
Resultados esperados: Se da una señal si se encuentra información semejante.
Evaluación de la prueba: Satisfactorio

Tabla 32 Prueba de aceptación#20: Enviar notificaciones. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU13_P1	**Historia de usuario:** Enviar notificaciones.
Nombre: Enviar notificaciones	
Descripción: Comprueba si se puede enviar notificaciones después de encontrar semejanza.	
Condiciones de ejecución: Que tenga conexión y que se encuentren semejanza entre los datos obtenidos y los temas.	
Pasos de ejecución: Se realiza en el momento de encontrar una semejanza.	
Resultados esperados: Se envía una notificación al correo proporcionado al crear un usuario.	
Evaluación de la prueba: Satisfactorio	

Resultados de las pruebas a las funcionalidades del software.

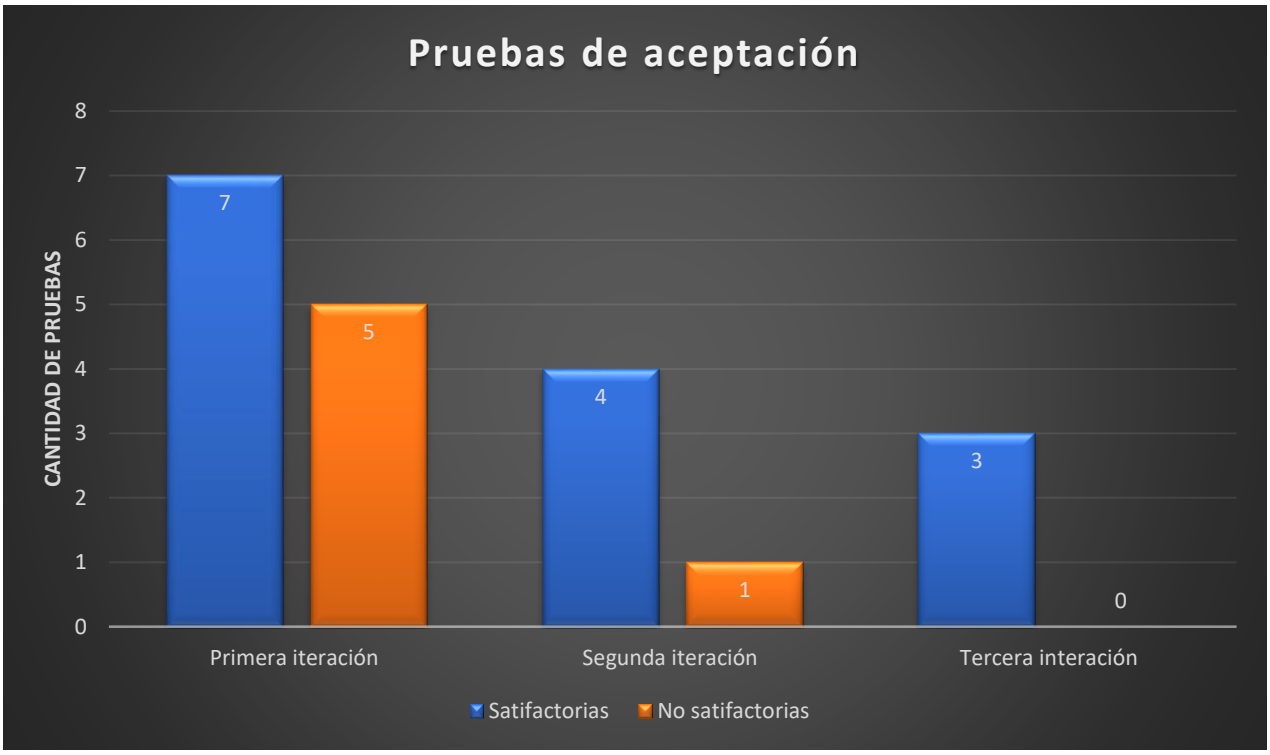


Ilustración 13 Resultado de las pruebas de aceptación Fuente: Elaboración propia

A través de tres iteraciones, se evaluaron diferentes aspectos de la aplicación, como la creación edición de alertas y el envío de correos electrónicos. En la primera iteración, se llevaron a cabo pruebas exhaustivas, y se identificaron cinco casos en los que la aplicación no cumplía completamente con los requisitos establecidos mostrando errores funcionales. Durante la segunda iteración, se enfocaron en la corrección de errores y se realizaron pruebas adicionales para asegurar la funcionalidad adecuada. Aunque se encontró un caso adicional de no conformidad en esta iteración, se trabajó en su resolución. En la última iteración, se logró solucionar todas las no conformidades detectadas, demostrando que la aplicación web era capaz de gestionar correctamente las alertas de correo electrónico.

Conclusiones del capítulo

En este capítulo se especificó el proceso de implementación a partir de la disgregación de las HU en tareas de ingeniería. Permitiendo esclarecer los procedimientos necesarios para dar cumplimiento a cada HU para definir y aplicar las pruebas de aceptación a las funcionalidades del sistema. Estas pruebas permitieron detectar errores y corregirlos, junto a

las pruebas unitarias que se realizaron a cada método de la aplicación fue posible obtener resultados concretos en cada iteración, contribuyendo a mejorar la eficiencia del sistema.

CONCLUSIONES FINALES

Con este desarrollo para la implementación de un sistema para la gestión automatizada de correos electrónicos de alerta de información se arriba a las siguientes conclusiones:

- La implementación exitosa del sistema para la gestión automatizada de correos electrónicos de alerta de información ha sido posible gracias a la cuidadosa selección y empleo de las herramientas, tecnologías y metodologías pertinentes. El uso estratégico de estas ha permitido su funcionamiento eficiente y óptimo.
- El empleo de las tecnologías como las API y el Web *scraping* permitió obtener información científica y académica relacionada con las ciencias de la computación para su proceso y posterior notificación.
- Los sistemas de gestión de alertas de correo electrónico son herramientas de vital importancia para la vigilancia tecnológica en las organizaciones. Estos permiten la detección temprana de información relevante, agilizan el monitoreo automatizado, facilitan la personalización y filtrado de contenido, y mejoran la toma de decisiones estratégicas. Gracias a estos las organizaciones pueden mantenerse al tanto de los avances tecnológicos, cambios regulatorios y eventos relevantes en su sector.

RECOMENDACIONES

Después de concluida la investigación se recomienda:

- Implementar una nueva funcionalidad que permita generar reporte de los temas de notificaciones más recurrentes y otros tipos de reportes.
- Aumentar el número de fuentes bibliográficas en el sistema para la gestión automatizada de correos electrónicos de alerta de información, a fin de ofrecer a los usuarios una mayor diversidad de alertas de información.

REFERENCIAS BIBLIOGRÁFICAS

[1] Antonio Mañas Pérez, «Las Tecnologías de la Información y la Comunicación en el ámbito educativo. Un tándem necesario en el contexto de la sociedad actual», *Jan-Abr 2019*, p. 86.

[2] Ronald M. Hernández, Isaac Sanchez Cáceres, Jesús Roberto Zarate Hermoza, Daniela Medina Coronado, Telmo Pablo Loli Poma, y Georgina Raquel Arévalo Gómez, «Tecnología de Información y Comunicación (TIC) y su práctica en la evaluación educativa», *05-21-19*, vol. Vol. 7, n.º N° 2, p. 10.

[3] José Tejada Fernándezy Katia V. Pozos Pérez, «NUEVOS ESCENARIOS Y COMPETENCIAS DIGITALES DOCENTES:HACIA LA PROFESIONALIZACIÓN DOCENTE CON TIC», *Enero-Marzo 2018*, vol. VOL.22, n.º Nº1, p. 51.

[4] R. M. T. Valdés, «Estructuras, procesos e instrumentos de vigilancia tecnológica. La vigilancia tecnológica como proceso de innovación relacional Universidad-Empresa», n.º 58, 2013.

[5] «Oficina Española de Patentes y Marcas - Información Tecnológica». Accedido: 28 de junio de 2023. [En línea]. Disponible en: https://www.oepm.es/es/informacion_tecnologica/informacion_gratuita/Alertas_Tecnologicas/index.html

[6] Martha Irene Romero Castro *et al.*, *INTRODUCCIÓN A LA SEGURIDAD INFORMÁTICA Y EL ANÁLISIS DE VULNERABILIDADES*, vol. Volumen 46 de Ingeniería y Tecnología. 3Ciencias, 2018.

[7] C. E. M. Echeverry, M. L. Trujillo, y L. I. L. Villegas, «Desarrollo de una aplicación móvil para alerta tecnológica», *Rev. Virtual Univ. Católica Norte*, n.º 48, pp. 316-330, 2016.

[8] M. Jakesch, K. Garimella, D. Eckles, y M. Naaman, «Trend Alert: How a Cross-Platform Organization Manipulated Twitter Trends in the Indian General Election», *Proc. ACM Hum.-Comput. Interact.*, vol. 5, n.º CSCW2, pp. 1-19, oct. 2021, doi: 10.1145/3479523.

[9] Yarnher Enrique Sánchez y Jheimer Julián Sepúlveda López, «Vigilancia tecnológica como mecanismo de innovación educativa», *20/11/2021*, vol. Vol. 15, n.º Núm. 4 (2021): Número Especial, p. 6.

[10] Omar Sierra Sarduy, «Componentes para la gestión de alertas y notificaciones en la herramienta informática Expediente Judicial Electrónico», Universidad de las Ciencias Informáticas, Facultad 3, 2019.

[11] B. Ropa-Carrión, M. Alama-Flores, B. Ropa-Carrión, y M. Alama-Flores, «Gestión organizacional: un análisis teórico para la acción», *Rev. Científica UCSA*, vol. 9, n.º 1, pp. 81-103, abr. 2022, doi: 10.18004/ucsa/2409-8752/2022.009.01.081.

[12] E. Brasil, «pyHDB - ferramenta heurística para a Hemeroteca Digital Brasileira: utilizando técnicas de web scraping para a pesquisa em história», *História Historiogr.*, vol. 15, n.º 40, pp. 186-217, 2022.

[13] D. Glez-Peña, A. Lourenço, H. López-Fernández, M. Reboiro-Jato, y F. Fdez-Riverola, «Web scraping technologies in an API world», *Brief. Bioinform.*, vol. 15, n.º 5, pp. 788-797, sep. 2014, doi: 10.1093/bib/bbt026.

[14] «What is an Application Programming Interface (API)? | IBM». Accedido: 3 de noviembre de 2023. [En línea]. Disponible en: https://www.ibm.com/topics/api

[15] H. E. G. Meléndez, «comunicación para las bibliotecas», *NUEVA ÉPOCA*, vol. 10, n.º 2, 2007.
[16] «Website change detection, monitoring, alerts, notifications, restock alerts | changedetection.io». Accedido: 19 de septiembre de 2023. [En línea]. Disponible en: https://changedetection.io/#features
[17] «Bio Is Changed - Company Information, Competitors, News & FAQs». Accedido: 19 de septiembre de 2023. [En línea]. Disponible en: https://6sense.com/company/bio-is-changed/www.slintel.com/bio-is-changed/5c3b020fd55ae49f1b78c788
[18] «Talkwalker: Inteligencia del consumidor para las marcas consumidor». Accedido: 20 de septiembre de 2023. [En línea]. Disponible en: https://www.talkwalker.com/es/
[19] IFTTT, «IFTTT - Automate business & home», IFTTT. Accedido: 19 de septiembre de 2023. [En línea]. Disponible en: https://ifttt.com/
[20] «Monitoreo de medios en línea y gestión de redes sociales», Mention. Accedido: 20 de septiembre de 2023. [En línea]. Disponible en: https://mention.com/es/
[21] «Influencers - Social Mention». Accedido: 20 de septiembre de 2023. [En línea]. Disponible en: https://www.socialmention.com/influencers/
[22] «Brand monitoring tool to analyze all brand mentions on the web», Awario. Accedido: 20 de septiembre de 2023. [En línea]. Disponible en: https://awario.com/
[23] «Alertas de Google: controla la Web en busca de contenido nuevo e interesante». Accedido: 20 de septiembre de 2023. [En línea]. Disponible en: https://www.google.com/alerts?hl=es
[24] E. Bautista-Villegas, «Metodologías agiles XP y Scrum, empleadas para el desarrollo de páginas web, bajo MVC, con lenguaje PHP y framework Laravel», *Rev. Amaz. Digit.*, vol. 1, p. e168, ene. 2022, doi: 10.55873/rad.v1i1.168.
[25] «What is Python? Executive Summary», Python.org. Accedido: 3 de noviembre de 2023. [En línea]. Disponible en: https://www.python.org/doc/essays/blurb/
[26] «Django», Django Project. Accedido: 3 de noviembre de 2023. [En línea]. Disponible en: https://www.djangoproject.com/
[27] J. D. Gauchat, *El gran libro de HTML5, CSS3 y JavaScript*. MARCOMBO, S.A., 2017. Accedido: 3 de noviembre de 2023. [En línea]. Disponible en: https://repositorio.uci.cu/jspui/handle/123456789/10129
[28] A. Durango, *Diseño Web con CSS: 2ª Edición*. IT Campus Academy.
[29] «Definición de JavaScript». Accedido: 2 de noviembre de 2023. [En línea]. Disponible en: https://www.definicionabc.com/tecnologia/javascript.php
[30] «JavaScript | MDN». Accedido: 2 de noviembre de 2023. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript
[31] «PostgreSQL: About». Accedido: 2 de noviembre de 2023. [En línea]. Disponible en: https://www.postgresql.org/about/
[32] «Visual Studio: IDE y Editor de código para desarrolladores de software y Teams». Accedido: 2 de noviembre de 2023. [En línea]. Disponible en: https://visualstudio.microsoft.com/es/
[33] «About Visual Paradigm». Accedido: 2 de noviembre de 2023. [En línea]. Disponible en: https://www.visual-paradigm.com/aboutus/
[34] «Visual Paradigm - UML, Agile, PMBOK, TOGAF, BPMN and More!» Accedido: 2 de noviembre de 2023. [En línea]. Disponible en: https://www.visual-paradigm.com/features/

[35] Jose Marino Enriquez Martinez, Sergio Alonso Rosales de la Vega, y Mario Martin Morales Flores, «PROPUESTA DE LEVANTAMIENTO DE REQUERIMIENTOS EN EL DESARROLLO DE SISTEMAS APLICANDO METODOLOGÍA TRIZ», Tesis, Mexico, 2018.
[36] Y. Molina Hernández, A. Granda Dihigo, A. VelázquezCintra, Y. Molina Hernández, A. Granda Dihigo, y A. VelázquezCintra, «Estrategia de desarrollo de requisitos no funcionales en aplicaciones para la salud», *Rev. Cuba. Informática Médica*, vol. 12, n.º 1, pp. 92-107, jun. 2020.
[37] J. Joskowicz, «Reglas y Prácticas en eXtreme Programming», p. 22.
[38] L. E. Urian y L. Á. Alba, «PROPUESTA DE METODOLOGIA PARA REALIZACION DE APP ANDROID PARA MANEJO DE RECURSO HUMANO, BASADO EN “SCRUM &».
[39] SINTYA MILENA MELÉNDEZ VALLADAREZ, MARIA ELIZABETH GAITAN, y . NELDIN NOEL PÉREZ REYES, «METODOLOGIA ÁGIL DE DESARROLLO DE SOFTWARE PROGRAMACION EXTREMA.», UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA, MANAGUA UNAN-MANAGUA, FACULTAD DE CIENCIAS E INGENIERÍA DEPARTAMENTO DE COMPUTACIÓN, 2016.
[40] W. Ocampo Pazos, J. Ulloa, J. Azcona Esteban, y M. Carrasco, «METODOLOGÍA HÍBRIDA DE DESARROLLO DE SOFTWARE COMBINANDO XP Y SCRUM», *Mikarimin*, vol. 5, may 2019.
[41] R. S. Pressman, «Ingenieria del Software. Un Enfoque Practico».
[42] M. G. Cambarieri, F. Difabio, y N. G. Martínez, «Implementación de una Arquitectura de Software guiada por el Dominio».
[43] «IEEE Standards Association», IEEE Standards Association. Accedido: 2 de noviembre de 2023. [En línea]. Disponible en: https://standards.ieee.org
[44] J. L. G. Hoyos, «MIGRACIÓN DEL SOFTWARE KACTUS-HCM DE UNA ARQUITECTURA CLIENTE-SERVIDOR A UNA ARQUITECTURA CLIENTE-CONTENEDOR», 2019.
[45] O. Sandoval Carlos, *INTEGRACIÓN DE PATRONES DE DISEÑO Y APLICACIONES MÓVILES EN UN SISTEMA DE GESTIÓN PARA EL CONTROL DE MANTENCIONES DE PLACAS CATÓDICAS DE LA CÍA. MINERA QUEBRADA BLANCA S.A. INTEGRATION OF DESIGN PATTERNS AND MOBILE APPLICATIONS IN A MANAGEMENT SYSTEM FOR MONITORING MAINTENANCE CATHODE PLATES OF MINING COMPANY QUEBRADA BLANCA SA*. 2017.
[46] L. E. González-Castaño, S. V. Marroquín-Soto, G. V. Maturana-González, y R. A. Manjarrés-Betancur, «Coffee Challenge: Un juego para la enseñanza de patrones de diseño de software», *Rev. Politécnica*, vol. 17, n.º 33, pp. 34-46, may 2021, doi: 10.33571/rpolitec.v17n33a3.
[47] G. Salazar Bermúdez, «USO DE PATRONES DE DISEÑO: UN CASO PRÁCTICO», 2013.
[48] J. A. LIC. ROSA MARIA MATO GARCIA, «SISTEMAS DE BASES DE DATOS», oct. 1999.
[49] ANDRES LEONARDO CARRILLO PEÑA, «PRÁCTICAS ACADÉMICAS EN ELECTROSOFTWARE S.A.S.», *Bucaramanga 2020*, p. 19.
[50] L. R. E. González, «Catálogo virtual interactivo para la visualización de contenidos bibliográficos usando Kinect 2», 2022.

[51] Esteban Bedoya Alzate, «Implementación de pruebas unitarias», Universidad de Antioquia, Facultad de Ingienería, departamento de Sistemas Medellín, Colombia, 2021.
[52] Asael Caraballo Oquendo y Javier Mancha Cabrera, «Sistema de gestión para el apoyo de la integralidad de los estudiantes de 5to», UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS, Facultad 4, 2022.

ANEXOS

Entrevista

Entrevista	
Objetivo:	Dirigida a:
1. Definir la situación problemática de la siguiente investigación 2. Establecer las reglas del negocio 3. Establecer el flujo actual del proceso del negocio 4. Definir los requisitos funcionales y no funcionales del sistema a desarrollar 5. Definir las herramientas a utilizar en el desarrollo 6. Definir la metodología con la que se estará trabajando	**MINCOM** **Director:** Rafael Luis Torralbas Ezpeleta **CITMA** **Director:** Ceferino Julio Santarén Suárez

Guía de preguntas:

1. ¿Cuál es el problema o desafío principal que enfrenta la organización?
2. ¿Qué problemas específicos han surgido al monitorear y recibir la información relevante en tiempo real?
3. ¿Cómo afecta esta situación a la toma de decisiones o la eficiencia de la organización?
4. ¿Existen regulaciones o políticas específicas que se deben cumplir en cuanto a la gestión y entrega de las alertas de información?
5. ¿Cómo es el flujo actual de obtención, proceso y distribución de la información relevante?
6. ¿Cuáles son los principales desafíos o ineficiencias identificados en el flujo actual de información?
7. ¿Qué funcionalidades específicas se esperan del sistema a desarrollar?
8. ¿Cuáles son los criterios para filtrar y personalizar las alertas según las necesidades del usuario?
9. ¿Se requiere la capacidad de enviar alertas a través de diferentes canales?

10. ¿Existen requisitos de rendimiento, escalabilidad, seguridad o confiabilidad que debe cumplir el sistema?
11. ¿Existen limitaciones tecnológicas o de infraestructura que se deban tener en cuenta?
12. ¿Existen preferencias o restricciones en cuanto a las herramientas o tecnologías a utilizar para el sistema?
13. ¿Se ha considerado alguna metodología específica para el desarrollo y seguimiento del proyecto?

Historias de usuario

Tabla 33 Historias de usuario#4: Agregar temas de notificación. Fuente: Elaboración propia

Número:4	**Nombre**: Agregar temas de notificación
Usuario: cliente	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:2
Programador Asignado: Daniel Felipe Fuentes	
Descripción: El sistema debe permitir agregar temas de notificaciones y seleccionar la fuente de información.	
Observaciones: El usuario debe de estar ya autentificado.	
Prototipo de Interfaz:	

Tabla 34 Historias de usuario#5: Modificar temas de notificación. Fuente: Elaboración propia.

Número:5	**Nombre**: Modificar temas de notificación
Usuario: cliente	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:2
Programador Asignado: Daniel Felipe Fuentes	
Descripción: El sistema debe permitir modificar los temas de notificación.	
Observaciones: El usuario debe de estar ya autentificado y tener temas de notificaciones agregadas.	
Prototipo de Interfaz:	

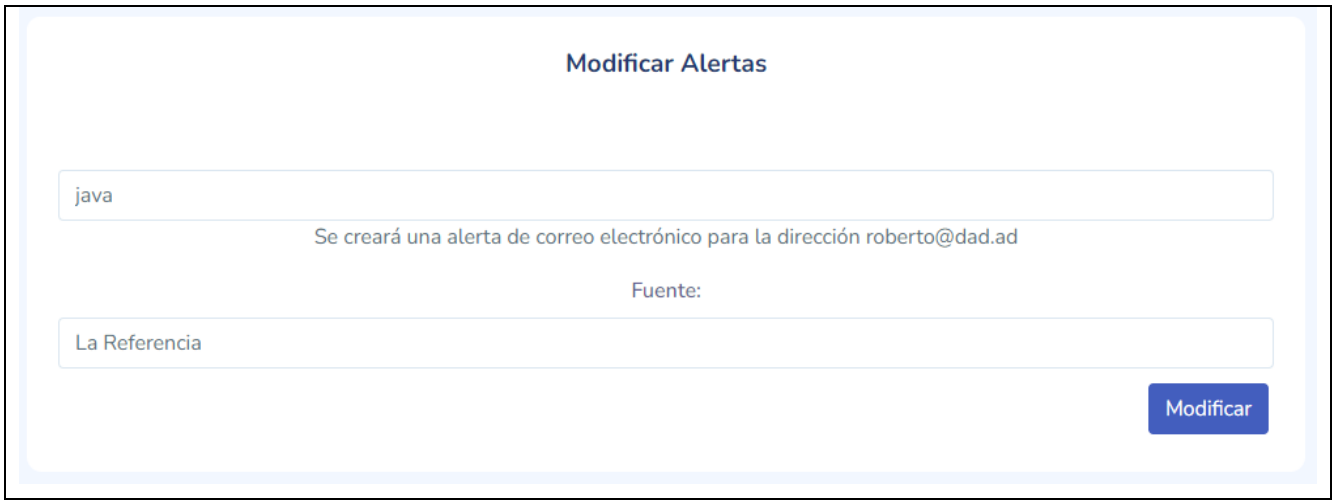


Tabla 35 Historias de usuario#6: Eliminar temas de notificación. Fuente: Elaboración propia.

Número:6	**Nombre**: Eliminar temas de notificación
Usuario: cliente	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:2
Programador Asignado: Daniel Felipe Fuentes	
Descripción: El sistema debe permitir eliminar los temas de notificación.	
Observaciones: El usuario debe de estar ya autentificado y tener temas de notificaciones agregadas.	
Prototipo de Interfaz:	

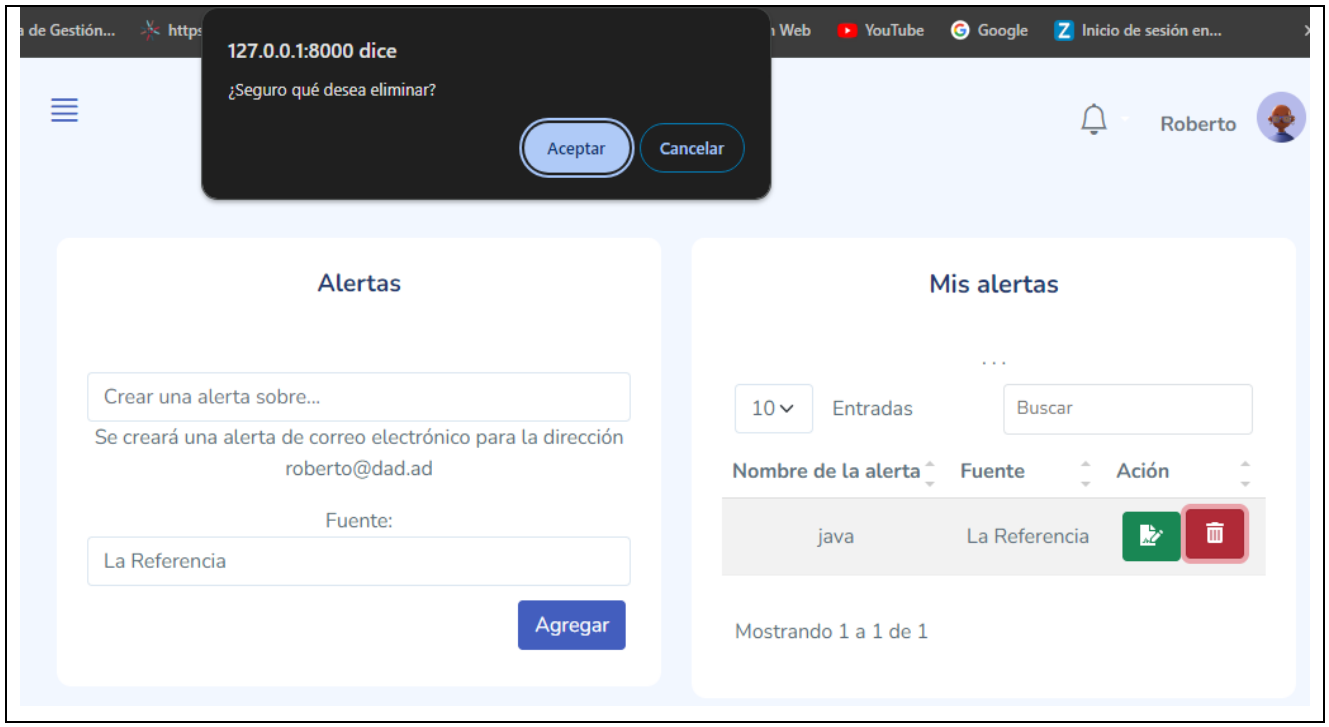


Tabla 36 Historias de usuario#7: Agregar usuario. Fuente: Elaboración propia.

Número:7	**Nombre**: Agregar usuario
Usuario: administrador	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:1
Programador Asignado: Daniel Felipe Fuentes.	
Descripción: El sistema debe permitir agregar usuarios por su nombre, usuario, correo, rol y contraseña.	
Observaciones: El usuario debe de estar ya autentificado y ser administrador.	
Prototipo de Interfaz:	

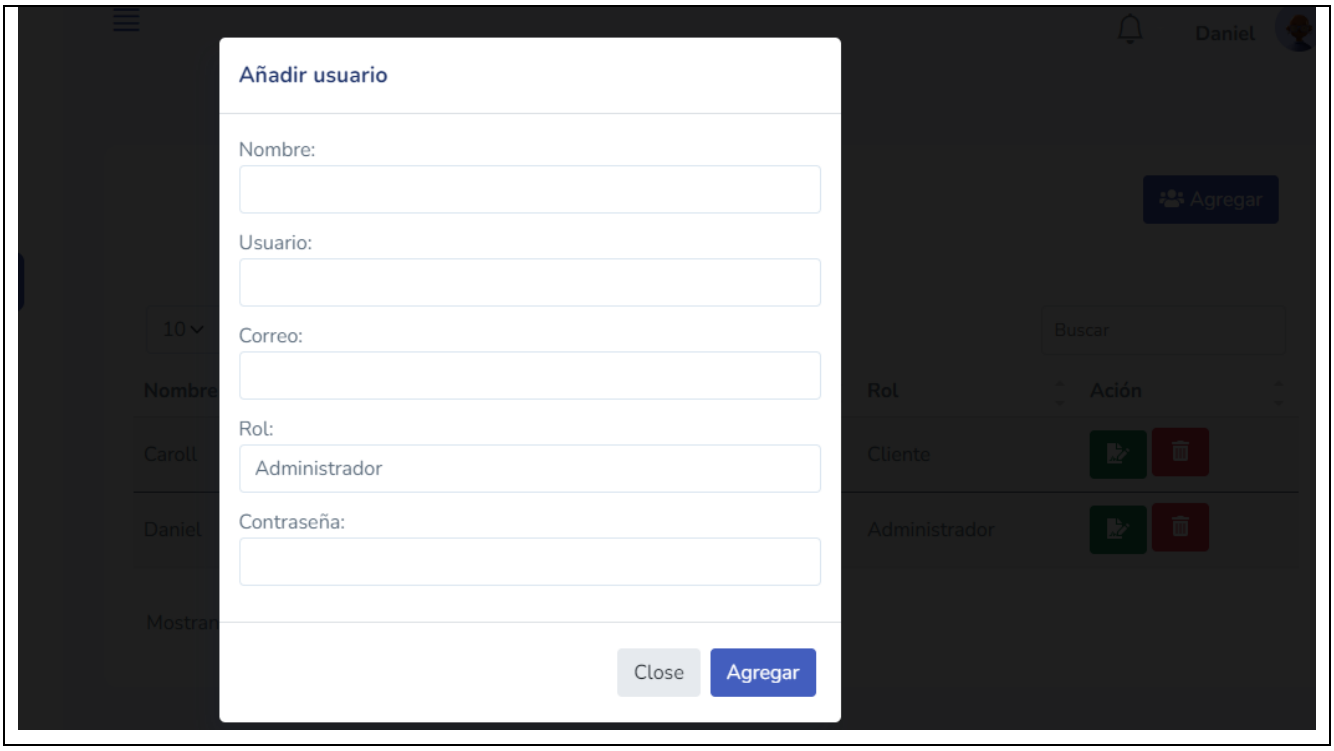


Tabla 37 Historias de usuario#8: Listar usuarios. Fuente: Elaboración propia.

Número:8	**Nombre**: Listar usuarios
Usuario: administrador	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:1
Programador Asignado: Daniel Felipe Fuentes	
Descripción: Muestra un listado de todos los usuarios	
Observaciones: El usuario debe de estar ya autentificado y ser administrador.	
Prototipo de Interfaz:	

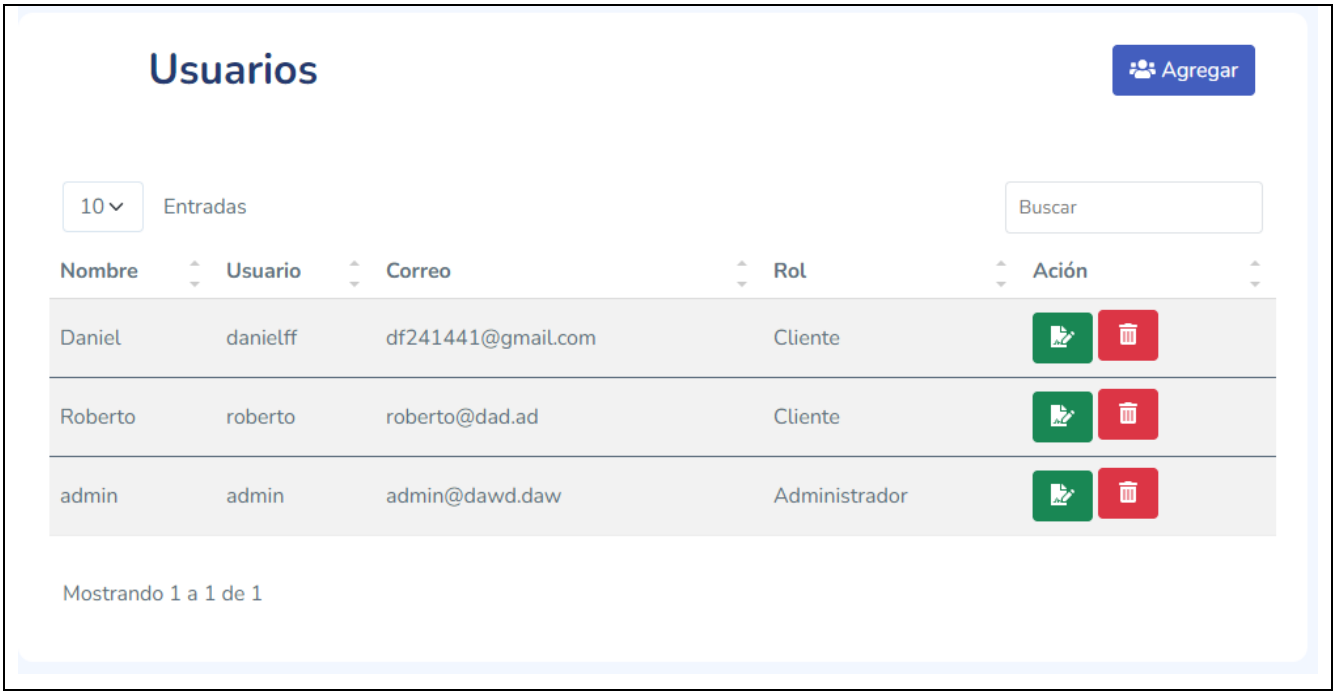


Tabla 38 Historias de usuario#9: Eliminar usuario. Fuente: Elaboración propia.

Número:9	**Nombre**: Eliminar usuario
Usuario: administrador	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:1
Programador Asignado: Daniel Felipe Fuentes	
Descripción: El sistema debe permitir eliminar los usuarios.	
Observaciones: El usuario debe de estar ya autentificado y ser administrador.	
Prototipo de Interfaz:	

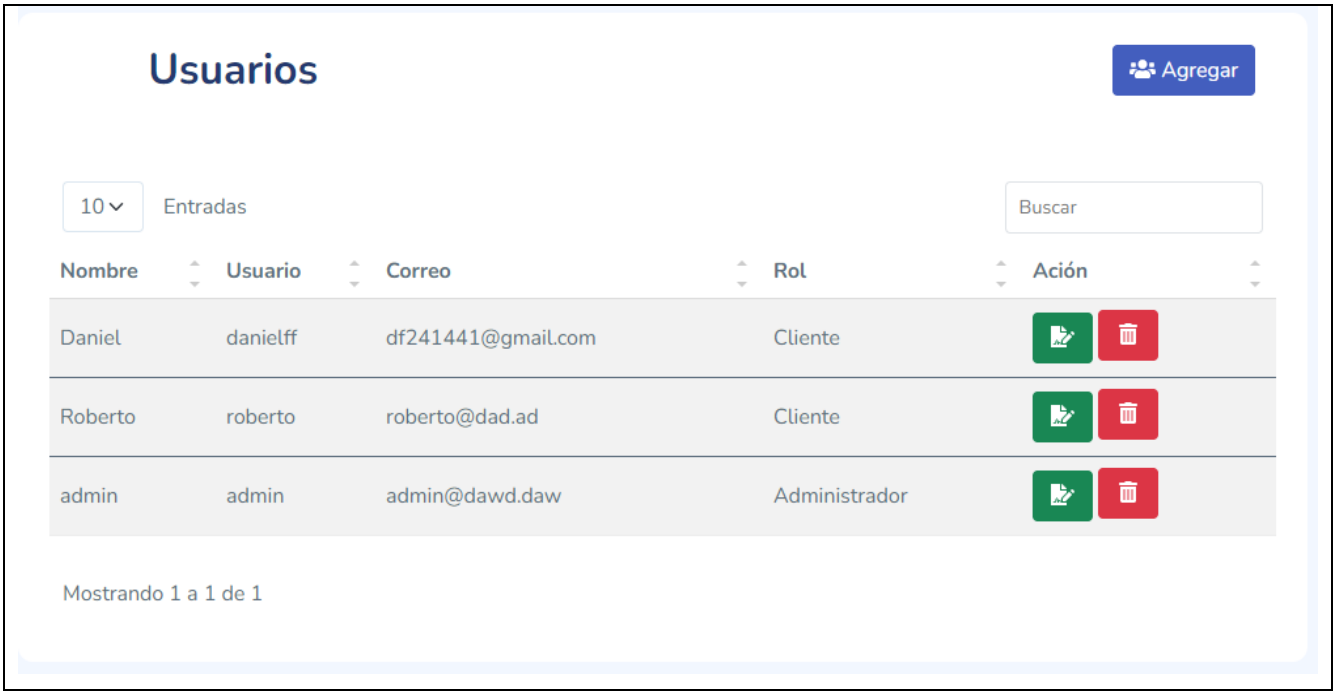


Tabla 39 Historias de usuario#10: Modificar usuarios. Fuente: Elaboración propia.

Número:10	**Nombre**: Modificar usuarios
Usuario: administrador	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:1
Programador Asignado: Daniel Felipe Fuentes	
Descripción: El sistema debe permitir modificar los campos nombre, correo y rol de los usuarios.	
Observaciones: El usuario debe de estar ya autentificado y ser administrador.	
Prototipo de Interfaz:	

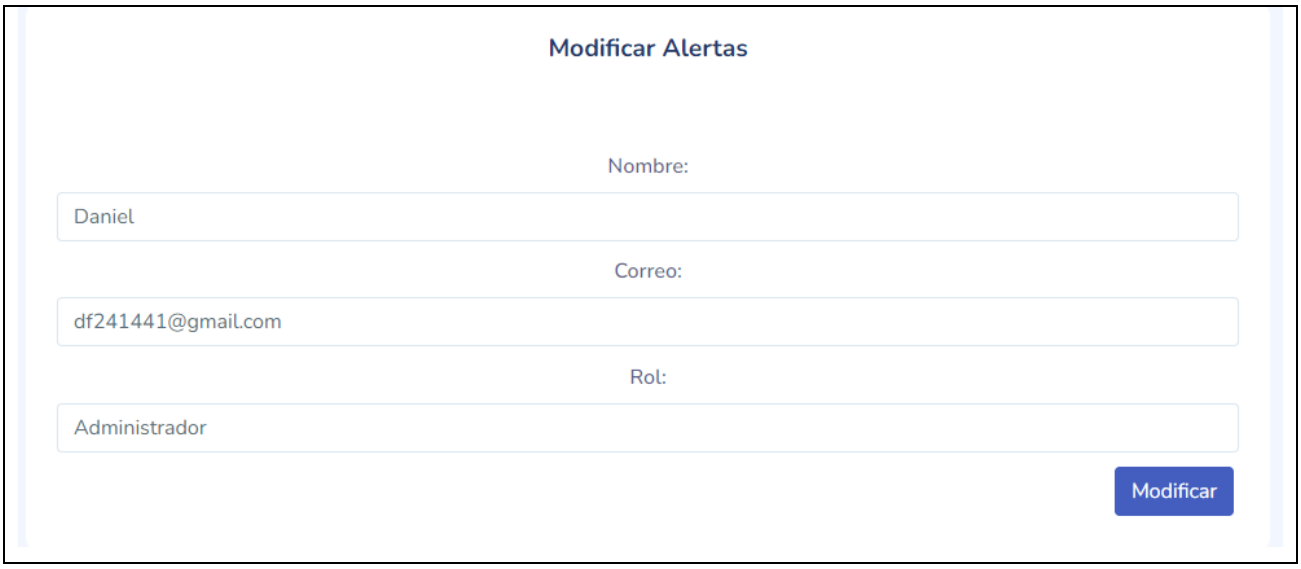


Tabla 40 Historias de usuario#11: Realizar web scraping. Fuente: Elaboración propia

Número:11	**Nombre**: Realizar web *scraping*
Usuario: usuario	
Prioridad en negocio: alta	**Riesgo de desarrollo**: alto
Puntos estimados:	**Iteraciones asignadas**:3
Programador Asignado: Daniel Felipe Fuentes	
Descripción: El sistema debe permitir realiza web *scraping* a una página determinada.	
Observaciones: No aplica	
Prototipo de Interfaz: No aplica	

Tabla 41 Historias de usuario#12: Búsqueda de semejanza. Fuente: Elaboración propia.

Número:12	**Nombre**: Búsqueda de semejanza
Usuario: especialista	
Prioridad en negocio: media	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:3
Programador Asignado: Daniel Felipe Fuentes	

Descripción: Permite realizar búsqueda de semejanza a la información obtenida mediante el web *scraping.*
Observaciones: No aplica
Prototipo de Interfaz: No aplica

Tabla 42 Historias de Usuario#13: Enviar notificaciones. Fuente: Elaboración propia.

Número:13	**Nombre**: Enviar notificaciones
Usuario: especialista	
Prioridad en negocio: alta	**Riesgo de desarrollo**: media
Puntos estimados:	**Iteraciones asignadas**:3
Programador Asignado: Daniel Felipe Fuentes	
Descripción: Envía notificaciones con lo encontrado al correo.	
Observaciones: No aplica	
Prototipo de Interfaz: No aplica	

Tareas de ingeniería

Tabla 43 Tarea de ingeniería: Implementar funcionalidad que permita agregar usuarios. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 4	Nombre de la HU: Agregar usuario
Nombre de la tarea: Implementar la funcionalidad que permita agregar usuarios.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 4 de mayo 2023	Fecha fin: 13 de mayo 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita validar y añadir los datos ingresado por el cliente para adicionar un nuevo usuario.	

Tabla 44 Tarea de ingeniería: Implementar la funcionalidad que permita listar usuarios. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 5	Nombre de la HU: Listar usuarios
Nombre de la tarea: Implementar la funcionalidad que permita listar usuarios.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 14 de mayo 2023	Fecha fin: 23 de mayo 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita listar todos los usuarios por el nombre, usuario y rol de la base de datos.	

Tabla 45 Tarea de ingeniería: Implementar la funcionalidad que permita eliminar un usuario. Fuente: Elaboración propia.

Tarea de Ingeniería	
Número de la tarea: 6	Nombre de la HU: Eliminar usuario
Nombre de la tarea: Implementar la funcionalidad que permita eliminar un usuario.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 24 de mayo 2023	Fecha fin: 2 de junio 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita eliminar un usuario del sistema.	

Tabla 46 Tarea de ingeniería: Implementar la funcionalidad que permita modificar un usuario. Fuente: Elaboración propia.

Tarea de Ingeniería	
Número de la tarea: 7	Nombre de la HU: Modificar usuarios
Nombre de la tarea: Implementar la funcionalidad que permita modificar un usuario.	
Tipo de tarea: Desarrollo	Puntos estimados:

Fecha de inicio: 3 de junio 2023	Fecha fin: 11 de junio 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita modificar cualquier atributo de la clase usuario.	

Iteración 2

Tabla 47 Tarea de ingeniería: Implementar la funcionalidad que permita eliminar temas de notificación. Fuente: Elaboración propia.

Tarea de ingeniería	
Número de la tarea: 11	Nombre de la HU: Eliminar temas de notificación
Nombre de la tarea: Implementar la funcionalidad que permita eliminar temas de notificación.	
Tipo de tarea: Desarrollo	Puntos estimados:
Fecha de inicio: 27 de julio 2023	Fecha fin: 9 de agosto 2023
Programador responsable: Daniel Felipe Fuentes	
Descripción: Implementar un método que permita eliminar una notificación agregada.	

Pruebas de unitarias

Prueba unitaria: Verifica el autentificar usuario en el sistema con campos incorrectos con el rol de administrador.

```
#Prueba para iniciar sesion del administrador con campos incorrectos
def test_Login_admin_wrong(self):
    self.useradmin.set_password('password')
    self.useradmin.save()
    response =self.client.login( username ='daniel11', password ='asdpassword')
    self.assertEqual(response, False)
```

Ilustración 14 Función test_Login_admin_wrong. Fuente: Elaboración propia.

Prueba con la cual se asegura si el sistema está autentificando de manera correcta a los usuarios con el rol de cliente.

```
#Prueba para iniciar sesion del cliente
def test_Login_client(self):
    self.userclient.set_password('password')
    self.userclient.save()
    response =self.client.login( username ='danielf', password ='password')
    self.assertEqual(response, True)
```

Ilustración 15 Función test_Login_client. Fuente: Elaboración propia

Prueba unitaria: Verifica el autentificar usuario en el sistema con campos incorrectos con el rol de cliente.

```
#Prueba para iniciar sesion del cliente con campos incorrectos
def test_Login_client_wrong(self):
    self.userclient.set_password('password')
    self.userclient.save()
    response =self.client.login( username ='dda1anielf', password ='daw2password')
    self.assertEqual(response, False)
```

Ilustración 16 Función test_Login_client_wrong. Fuente: Elaboración propia

Prueba unitaria: Verifica si se está mostrando la lista de usuario correctamente.

```
#Prueba para verificar si muestra la lista de usuarios
def test_user_list(self):
    self.useradmin.set_password('password')
    self.useradmin.save()
    login =self.client.login( username ='daniel', password ='password')
    response = self.client.get('/usuario', HTTP_X_REQUEST_WITH = 'XMLHttpRequest')
    self.assertEqual(response.status_code,200)
   # print (response.content.decode('utf-8'))
```

Ilustración 17 Función test_user_list. Fuente: Elaboración propia

Prueba unitaria: Verifica si se están agregando correctamente las alertas.

```
#Prueba para verificar si anade una alerta
def test_alert_add(self):
    self.userclient.set_password('password')
    self.userclient.save()
    login =self.client.login( username ='danielf', password ='password')
    response = self.client.post('/Anadir_Alerta', {'alerta':'python'})
    self.assertEqual(response.headers['Location'],'/')
```

Ilustración 18 Función test_alert_add. Fuente: Elaboración propia

Prueba unitaria: Verifica si se están agregando correctamente las alertas por el administrador.

```
#Prueba para verificar si anade una alerta el administrador
def test_alert_add_admin(self):
    self.useradmin.set_password('password')
    self.useradmin.save()
    login =self.client.login( username ='daniel', password ='password')
    response = self.client.post('/Anadir_Alerta', {'alerta':'python'})
    self.assertEqual(response.headers['Location'],'/')
```

Ilustración 19 Función test_alert_add_admin. Fuente: Elaboración propia

Prueba unitaria: Verifica si se está mostrando la lista de alertas correctamente.

```
#Prueba para verificar si se muestra una lista de alertas
def test_alert_list(self):
    self.userclient.set_password('password')
    self.userclient.save()
    login =self.client.login( username ='danielf', password ='password')
    Alerta = self.client.post('/Anadir_Alerta', {'alerta':'python'})
    response = self.client.get('/', HTTP_X_REQUEST_WITH = 'XMLHttpRequest')
    #print (response.content.decode('utf-8'))
    self.assertEqual(response.status_code, 200)
```

Ilustración 20 Función test_alert_list. Fuente: Elaboración propia

Prueba unitaria: Verifica si se está mostrando la lista de alertas correctamente para el administrador.

```
#Prueba para verificar si muestra la lista de alertas para el administrador
def test_alert_list_admin(self):
    self.useradmin.set_password('password')
    self.useradmin.save()
    login =self.client.login( username ='daniel', password ='password')
    Alerta = self.client.post('/Anadir_Alerta', {'alerta':'python'})
    response = self.client.get('/', HTTP_X_REQUEST_WITH = 'XMLHttpRequest')
    # print (response.content.decode('utf-8'))
    self.assertEqual(response.status_code, 200)
```

Ilustración 21 Función test_alert_list_admin. Fuente: Elaboración propia

Prueba unitaria: Verifica si se puede modificar una alerta ya ingresada por el cliente

```python
#Prueba para verificar si se puede modificar una alerta
def test_alert_edit(self):
    self.userclient.set_password('password')
    self.userclient.save()
    login =self.client.login( username ='danielf', password ='password')
    Alerta = self.client.post('/Anadir_Alerta', {'alerta':'python'})
    modificar = self.client.post('/mod_alerta', {'id' : '3', 'nombre_alerta':'java'})
    response = self.client.get('/', HTTP_X_REQUEST_WITH = 'XMLHttpRequest')
    #print (response.content.decode('utf-8'))
    self.assertEqual(response.status_code, 200)
```

Ilustración 22 Función test_alert_edit. Fuente: Elaboración propia

Prueba unitaria: Verifica si se puede modificar una alerta ya ingresada por el administrador

```python
#Prueba para verificar si se puede modificar una alerta por el administrador
def test_alert_edit_admin(self):
    self.useradmin.set_password('password')
    self.useradmin.save()
    login =self.client.login( username ='daniel', password ='password')
    Alerta = self.client.post('/Anadir_Alerta', {'alerta':'c++'})
    modificar = self.client.post('/mod_alerta', {'id' : '4', 'nombre_alerta':'C#'})
    response = self.client.get('/', HTTP_X_REQUEST_WITH = 'XMLHttpRequest')
    #print (response.content.decode('utf-8'))
    self.assertEqual(response.status_code, 200)
```

Ilustración 23 Función test_alert_edit_admin. Fuente: Elaboración propia

Prueba unitaria: Verifica si se está realizando correctamente el *scraping* y si está recuperando bien la información y enviando correo de alerta de estas.

```python
#Prueba para verificar si se puede realizar el scraping y si puede mandar la notificacion
def test_scraping(self):
    self.useradmin.set_password('password')
    self.useradmin.save()
    login =self.client.login( username ='daniel', password ='password')

    usuario= get_object_or_404(Usuario, username='daniel')
    alerta = Alerta.objects.create(usuario= usuario, alerta='python')


    request = self.client.get(reverse('scraping'))
    Scraping(request)


    self.assertGreaterEqual(len(mail.outbox), 0)  # Verificar que se envió un correo electrónico
    if len(mail.outbox)>0:
        for i in mail.outbox:
            self.assertEqual(i.subject, 'Alertas VIGITEC')  # Verificar el asunto del correo electrónico

            self.assertIn('Hay una nueva publicaión sobre ', i.body)  # Verificar el contenido del correo electrónico

    # Verificar la redirección
    self.assertRedirects(request, '/')
```

Ilustración 24 Función test_scraping. Fuente: Elaboración propia

Pruebas de aceptación

Tabla 48 Prueba de aceptación#6: Agregar usuario. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU7_P1	**Historia de usuario:** Agregar usuario.
Nombre: Agregar usuario.	
Descripción: Comprueba si se puede agregar usuario.	
Condiciones de ejecución: El usuario debe de estar ya autentificado y ser administrador.	
Pasos de ejecución: Al iniciar sesión se mostrará un menú desplegable con la opción de usuarios la cual permitirá agregar usuarios a la base de datos rellenado los datos correspondientes.	
Resultados esperados: Si los datos son correctos se agrega un nuevo usuario.	
Evaluación de la prueba: Satisfactorio	

Tabla 49 Prueba de aceptación#7: Agregar usuario P2. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU7_P2	**Historia de usuario:** Agregar usuario.
Nombre: Agregar usuario.	
Descripción: Comprueba si se puede agregar usuario.	
Condiciones de ejecución: El usuario debe de estar ya autentificado y ser administrador.	
Pasos de ejecución: Al iniciar sesión se mostrará un menú desplegable con la opción de usuarios la cual permitirá agregar usuarios a la base de datos rellenado los datos correspondientes.	
Resultados esperados: Si los datos son incorrectos no se agrega un usuario y mostrará el mensaje ‘Datos incorrectos’.	
Evaluación de la prueba: Satisfactorio	

Tabla 50 Prueba de aceptación#8: Listar usuarios. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU8_P1	**Historia de usuario:** Listar usuarios.
Nombre: Listar usuarios.	
Descripción: Comprueba si se puede listar usuarios.	
Condiciones de ejecución: El usuario debe de estar ya autentificado y ser administrador.	
Pasos de ejecución: Al iniciar sesión se mostrará un menú desplegable con la opción de usuarios en la cual se verá una tabla con los usuarios.	
Resultados esperados: Si hay algún usuario agregado se mostrará una lista.	
Evaluación de la prueba: Satisfactorio	

Tabla 51 Prueba de aceptación#9: Listar usuarios P2. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU8_P2	**Historia de usuario:** Listar usuarios.
Nombre: Listar usuarios.	
Descripción: Comprueba si se puede listar usuarios.	
Condiciones de ejecución: El usuario debe de estar ya autentificado y ser administrador.	
Pasos de ejecución: Al iniciar sesión se mostrará un menú desplegable con la opción de usuarios en la cual se verá una tabla con los usuarios.	
Resultados esperados: Si no hay ningún usuario agregado se mostrará un mensaje que diga “No hay usuario”.	
Evaluación de la prueba: Satisfactorio	

Tabla 52 Prueba de aceptación#10: Eliminar usuario. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU9_P1	**Historia de usuario:** Eliminar usuario.

Nombre: Eliminar usuario
Descripción: Comprueba si se puede eliminar los usuarios.
Condiciones de ejecución: El usuario debe de estar ya autentificado y ser administrador y que exista al menos un usuario.
Pasos de ejecución: Al iniciar sesión se mostrará un menú desplegable con la opción de usuarios en la cual se verá una tabla con los usuarios y al lado el botón de eliminar.
Resultados esperados: Se recarga la página y se elimina el usuario.
Evaluación de la prueba: Satisfactorio

Tabla 53 Prueba de aceptación#11: Modificar usuarios. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU10_P1	**Historia de usuario:** Modificar usuarios.
Nombre: Modificar usuarios	
Descripción: Comprueba si se puede modificar los usuarios.	
Condiciones de ejecución: El usuario debe de estar ya autentificado y ser administrador y que exista al menos un usuario.	
Pasos de ejecución: Al iniciar sesión se mostrará un menú desplegable con la opción de usuarios en la cual se verá una tabla con los usuarios y al lado el botón de modificar.	
Resultados esperados: Si se completan todos los datos y están correctos se modifica el usuario y se recarga la página.	
Evaluación de la prueba: Satisfactorio	

Tabla 54 Prueba de aceptación#12: Modificar usuarios P2. Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código: HU10_P2	**Historia de usuario:** Modificar usuarios.

Nombre: Modificar usuarios
Descripción: Comprueba si se puede modificar los usuarios.
Condiciones de ejecución: El usuario debe de estar ya autentificado y ser administrador y que exista al menos un usuario.
Pasos de ejecución: Al iniciar sesión se mostrará un menú desplegable con la opción de usuarios la cual se verá una tabla con los usuarios y al lado el botón de modificar.
Resultados esperados: Si no se completan todos los datos o no están correctos se mostrará un mensaje donde diga “Datos incorrectos”.
Evaluación de la prueba: Satisfactorio