



INFORME DE TAREA ACADÉMICA 1

Sistemas de Información Transaccionales
Carrera de Ingeniería de Computación y Sistemas

Sección: NRC - 16705

Docente: Henry Antonio Mendoza Puerta

Startup: ProfPost - Plataforma de publicaciones

Integrantes:

Código	Apellidos	Nombres
000260691	Berrospi Reyes	Hernan Mauricchio Gaston
000264553	Llaure Calipuy	Oscar Segundo
000259687	Oliden Agreda	Alejandro del Piero
000125431	Ramos Cotrina	Vania Melissa
000260664	Velásquez Reyes	Sergio Emanuel

Índice de Contenido

Registro de Versiones del Informe	2
Capítulo I: Introducción	3
1.1. Startup Profile	4
1.2. Antecedentes y Problemática	5
1.2.1. Antecedentes	5
1.2.2. Problemática Identificada.....	7
1.3. Segmentos objetivo	8
1.3.1. Creadores de Contenido.....	8
1.3.2. Lectores de Contenido.....	8
Capítulo II: Requirements Elicitation & Analysis.....	10
2.1. Diseño de entrevistas.....	10
Preguntas para Lectores:.....	11
2.2. Registro de entrevistas	12
2.3. Análisis de entrevistas	13
2.3.1. Creadores	13
2.3.2. Lectores	13
Hallazgo 1: Preferencia por recomendaciones personalizadas	13
Hallazgo 3: Preferencia por donaciones únicas con interacción inmediata.....	13
Hallazgo 4: Interacción directa con los creadores	14
Hallazgo 5: Preferencia por métodos de pago rápidos y seguros.....	14
Capítulo III: Requirements Specification	14
3.1. EPICs	14
3.2. User Stories & Criterios de Aceptación	15
3.3. Product Backlog	27
Capítulo IV: Product Design.....	29
4.1. Style Guidelines: Imágenes, tipografías, colores y estilos	29
4.2. Web Applications Prototyping	36
4.3. Software Object-Oriented Design	37
4.3.1 Class Diagram	37
4.3.2. Database Design	37
Capítulo V: Product Implementation, Validation & Deployment	37
5.1. Software Configuration Management	37
5.2 Web Service API Restful Implementation	37
Conclusiones	38
Recomendaciones	38
Bibliografía	39

Registro de Versiones del Informe

Versión	Fecha	Autor	Descripción
1.0	28/08/2024	Vania Ramos	Creación del perfil de la startup y definición del problema.
1.1	03/09/2024	Oscar Llaure Alejandro Oliden	Realización y análisis de entrevistas, identificación de necesidades.
1.2	06/09/2024	Sergio Velásquez Hernan Berrospi Vania Ramos	Elaboración de User Stories, EPICs y Product Backlog.
1.3	07/09/2024	Oscar Llaure	Realización en las pautas de estilo y en la creación de prototipos para aplicaciones web.
1.4	08/09/2024	Sergio Velásquez Hernan Berrospi	Elaboración de diagrama de clases y diagrama de base de datos
1.5	16/09/2024	Sergio Velásquez	Gestión de la configuración del software y desarrollando APIs RESTful para facilitar la integración y comunicación entre servicios web.
1.6	18/09/2024	Alejandro Oliden	Revisión y actualización de conclusiones y recomendaciones.

Capítulo I: Introducción

En la actualidad, el entorno académico universitario se encuentra en constante evolución, impulsado por la digitalización y el acceso a nuevas tecnologías. Estudiantes y docentes participan en un ciclo continuo de creación y consumo de conocimientos, lo que ha dado lugar a una creciente demanda de recursos educativos accesibles y personalizados. Sin embargo, a pesar de los avances en tecnología educativa, la comunidad universitaria sigue enfrentando desafíos significativos para acceder a contenidos de calidad que sean adecuados a sus necesidades curriculares y, al mismo tiempo, carece de espacios adecuados donde los creadores de contenido académico puedan monetizar su trabajo de manera eficiente.

Frente a este escenario, ProfPost surge como una plataforma innovadora diseñada para aprovechar esta oportunidad única. Nuestro proyecto no solo facilita la publicación de contenidos educativos especializados, sino que también ofrece a los creadores la posibilidad de monetizar sus contribuciones a través de suscripciones, donaciones y otros mecanismos de ingresos. Esta combinación de funcionalidad y sostenibilidad económica busca resolver un vacío crítico en el ámbito académico: el de generar ingresos directos a partir de la creación de conocimiento académico de valor.

Sin embargo, el desarrollo de este proyecto implica afrontar ciertas restricciones. Uno de los desafíos más importantes es la monetización de contenido, ya que involucra una serie de factores como es el caso de la protección de datos personales, protección de pagos, el cumplimiento de normativas y regulaciones. Además, debemos asegurar que la plataforma mantenga una experiencia de usuario robusta, simple y confiable, capaz de adaptarse a las expectativas tanto de estudiantes como de docentes.

Nuestra misión es fortalecer a los creadores de contenido académico, proporcionándoles una plataforma intuitiva, segura y accesible donde puedan compartir y monetizar sus conocimientos de manera efectiva. Así como, fomentar la colaboración académica y la difusión de contenidos de alta calidad, facilitando el acceso al conocimiento y recompensando a quienes lo generan.

Nuestra visión es convertirnos en una plataforma reconocida dentro del entorno universitario, donde el intercambio de conocimientos no solo sea una herramienta de aprendizaje, sino también una fuente de ingresos para aquellos que contribuyen al progreso académico.

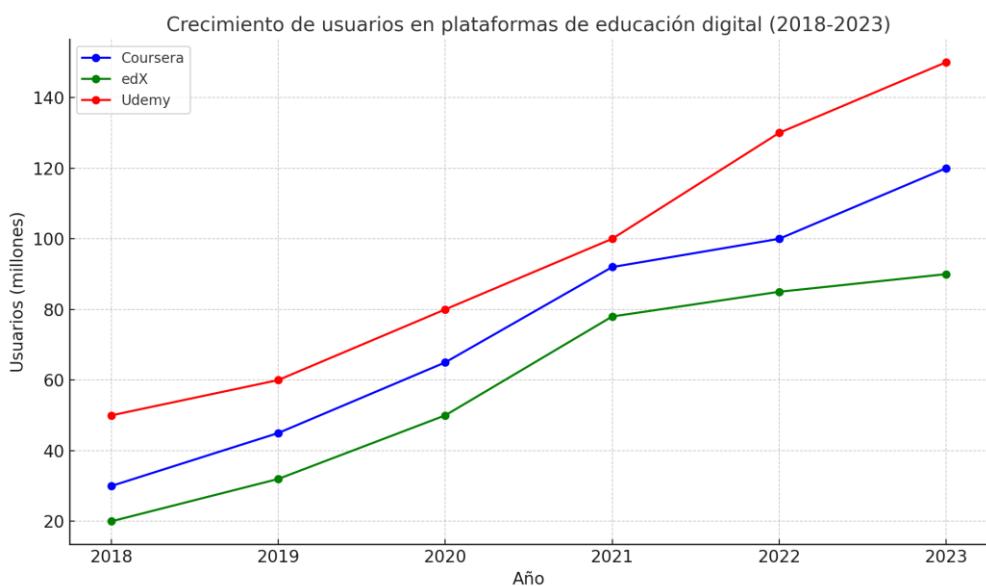
1.1. Startup Profile

	Soy Mauricio Berrospi, estudiante de Ingeniería de Computación y Sistemas. En este proyecto me enfocaré como Diseñador Web, Programador. Además de la tecnología, ofrezco habilidades colaborativas y un enfoque centrado en la solución creativa de problemas.
	Soy Oscar Llaure, estudiante de Ingeniería de Computación y Sistemas. En este proyecto me especializaré como Entrevistador de Contenidos, Gerente de Relaciones con clientes, Redactor de contenidos. Mi actitud colaborativa y creativa complementa mi habilidad técnica.
	Soy Alejandro Oliden, estudiante de Ingeniería de Computación y Sistemas. En este proyecto me especializaré como Entrevistador de Contenidos, Gerente de Relaciones con clientes, Redactor de contenidos. Mi actitud colaborativa y creativa complementa mi habilidad técnica.
	Soy Vania Ramos, estudiante de Ingeniería de Computación y Sistemas. En este proyecto me especializaré como Especificadora de Requisitos, Analista de Datos. Contribuyo con una capacidad para innovar y una disposición para adaptarme a nuevos desafíos, trabajando de manera colaborativa.
	Soy Sergio Velasquez, estudiante apasionado de Ingeniería de Computación y Sistemas. Me especializaré en este proyecto como Diseñador Web, Programador, Gestor de Base de datos. Más allá de la tecnología, aportó habilidades colaborativas y un enfoque creativo.

1.2. Antecedentes y Problemática

1.2.1. Antecedentes

En los últimos años y con la llegada de la pandemia, el aumento de la demanda de contenido especializado en educación, investigación y divulgación científica ha experimentado una transformación impulsada por la digitalización, esto ha llevado al creciente movimiento hacia la publicación de investigaciones académicas en plataformas logrando que el acceso a contenidos académicos sea más sencillo. Asimismo, ha permitido la creación de contenido específico que se puede monetizar a través de plataformas especializadas.



A nivel local se ha podido observar que, si bien el contenido académico en línea ha crecido, el acceso sigue siendo desigual debido a los altos costos, generando un aumento en la demanda de plataformas que ofrezcan contenido accesible y de calidad. Además, son los mismos universitarios y docentes los que están comenzando a generar y compartir sus propios contenidos.

Este fenómeno de creación de contenido por parte de los propios estudiantes y docentes universitarios no solo responde a la necesidad de suplir la falta de acceso a recursos educativos de pago, sino también a la creciente demanda de materiales adaptados a las realidades locales y curriculares específicas. Ya que a menudo el contenido encontrado en línea está enfocado a mercados internacionales, por lo que no logran cubrir todas las particularidades que demandan las universidades locales. Esto ha generado un vacío que los propios universitarios están empezando a llenar con materiales como apuntes, resúmenes, investigaciones y tutoriales, dirigidos directamente a un público universitario.

- **Udemy** (<https://www.udemy.com/es/>) Es una plataforma de cursos en línea que permite a cualquier persona crear y publicar cursos sobre una amplia variedad de temas, desde tecnología y negocios hasta artes y desarrollo personal. Los creadores de contenido (instructores) pueden subir videos, presentaciones, materiales adicionales y evaluaciones interactivas para sus cursos.

Objetivo: Proporcionar una plataforma flexible que se adapte a diferentes estilos de aprendizaje, donde tanto los instructores como los estudiantes se beneficien: los estudiantes obtienen acceso a cursos de alta calidad a precios accesibles, y los creadores reciben una compensación justa por su trabajo.

- **EdX** (<https://www.edx.org/>) Es una plataforma de MOOCs (Cursos Masivos Abiertos en Línea), fundada por el MIT y Harvard. Ofrece cursos universitarios en colaboración con instituciones educativas de renombre. Los creadores de contenido son universidades y centros de investigación que suben sus cursos a la plataforma.

Objetivo: Permitir que las personas puedan aprender de manera gratuita o a bajo costo y que los creadores de contenido reciban compensación por su trabajo y, al mismo tiempo, financien nuevos cursos y programas.

- **Coursera** (<https://www.coursera.org/>) Es una plataforma de MOOCs que colabora con universidades y empresas para ofrecer cursos en línea.

Objetivo: ampliar el acceso a la educación de alta calidad, permitiendo que los estudiantes aprendan de los mejores expertos del mundo. Asimismo, busca que la educación en línea sea sostenible mediante la monetización

- **Joomag** (<https://www.joomag.com/es>) Una plataforma de publicación en línea que permite a los usuarios crear y monetizar contenido digital como revistas, catálogos y libros electrónicos. Ofrece herramientas integradas para la distribución y análisis del contenido.

Objetivo: Busca facilitar la monetización para creadores de contenido que no quieren saturar sus sitios con anuncios intrusivos, pero que aún desean generar ingresos a través de su audiencia. La idea es ofrecer publicidad relevante y no disruptiva, lo que mejora la experiencia del usuario y, a la vez, ayuda a los creadores a monetizar sus plataformas.

- **Adsterra** (<https://adsterra.com/>) Es una red publicitaria que conecta editores con anunciantes, permitiendo monetizar blogs y sitios web

mediante la colocación de anuncios. Ofrece múltiples formatos publicitarios y métodos de pago.

Objetivo: Ayudar a los creadores de contenido a monetizar el tráfico de sus sitios de manera eficiente y personalizada, adaptando los anuncios según la audiencia y el tipo de contenido que ofrecen.

- **Cafecito (<https://cafecito.app/>)** Permite a creadores de contenido recibir apoyo económico de su audiencia a través de un modelo de suscripciones. Los creadores pueden personalizar su perfil en la plataforma y compartirlo en redes sociales o sitios web para recibir aportes de su comunidad.

Objetivo: Ofrecer una manera sencilla y accesible para que los creadores moneticen su trabajo sin la necesidad de depender de suscripciones o publicidad.

1.2.2. Problemática Identificada

La problemática principal que los usuarios podrían enfrentar es la falta de una solución unificada que ofrezca una experiencia fluida y completa para la creación, publicación, monetización, y análisis de contenido en un solo lugar.

En muchas plataformas, los creadores de contenido deben lidiar con diferentes herramientas o servicios para gestionar aspectos como la monetización, publicación, y análisis de rendimiento. Esta fragmentación puede resultar en una experiencia menos eficiente y más costosa, ya que los creadores deben gestionar múltiples cuentas, interfaces y pagos, lo que a su vez dificulta el enfoque en la creación de contenido de calidad y el engagement con su audiencia. Además, la falta de personalización y las limitaciones en la flexibilidad de monetización en algunas de estas plataformas pueden restringir el potencial de ingresos de los creadores.

De acuerdo a un censo del 2023 podemos ver los principales desafíos que tienen los usuarios que usan aplicaciones como Joomag, Adsterra, y Cafecito:

Problemáticas que enfrentan los usuarios	
Problemática	Usuarios
Falta de herramientas integradas	428
Dificultad para atraer audiencia	669
Bajas tasas de suscripción/donación	425
Falta de soporte técnico	280
Limitaciones en la personalización	677
Monetización	1264

Estos números indican que la monetización es la preocupación principal, seguida por las bajas tasas de suscripción/donación y las limitaciones en la personalización.

[Fuente: Cómo monetizar en una pagina web | Mailchimp](#)

1.3. Segmentos objetivo

1.3.1. Creadores de Contenido

Estudiantes entre 4to al 9no ciclo de la facultad de Ingeniería que pertenezcan al Programa de Estudiantes de Tutoría y Docentes de la Universidad Privada Antenor Orrego entre 20 - 45 años que busquen:

- Compartir conocimientos relacionados con sus estudios, como resúmenes de clases, proyectos académicos, o investigaciones.
- Publicar contenido que pueda ayudar a otros estudiantes, como guías de estudio, tutoriales o explicaciones de temas complejos.
- Utilizar la plataforma como un portafolio digital que muestre sus trabajos y logros académicos.

Necesidades y Comportamientos:

- Plataforma que facilite la publicación de contenido académico, como apuntes, trabajos, y proyectos.
- Buscan formas de monetizar su contenido, como a través de micropagos, donaciones, o intercambios de servicios académicos, para cubrir gastos educativos u obtener ingresos adicionales.
- Quieren estadísticas sobre cómo su contenido ayuda a otros estudiantes, como la cantidad de descargas o lecturas.
- Su contenido está enfocado en temas específicos y de interés directo para sus compañeros de clase o facultad.

1.3.2. Lectores de Contenido

Estudiantes entre 18 - 25 años de la Universidad Privada Antenor Orrego que busquen mejorar su rendimiento académico utilizando contenido relevante

(resúmenes, ejemplos de trabajos o explicaciones detalladas de temas específicos), seguir a creadores que publican material útil para sus cursos o áreas de estudio, interactuar con otros estudiantes para intercambiar ideas, materiales y experiencias.

Necesidades y Comportamientos:

- Buscan contenido que les ayude a preparar exámenes o completar tareas de manera eficiente.
- Necesitan una interfaz intuitiva que les permita encontrar el material adecuado para sus estudios.
- Interactuar con los creadores para hacer preguntas o pedir aclaraciones.
- Están dispuestos a apoyar a los creadores que les proporcionan contenido útil, ya sea con recomendaciones o pequeñas donaciones.
- Revisan regularmente la plataforma para buscar nuevos recursos que les ayuden en sus estudios.
- Utilizan el contenido para maximizar su tiempo de estudio y mejorar su rendimiento académico.

Capítulo II: Requirements Elicitation & Analysis

2.1. Diseño de entrevistas

Objetivos de la Entrevista

1. Comprender las expectativas y deseos de los usuarios respecto a una plataforma de Publicación y Monetización de Contenidos.
2. Evaluar la disposición de los usuarios a adoptar nuevas soluciones tecnológicas.
3. Evaluar la disposición de los usuarios a pagar por contenido y qué incentivos o características los motivarían a hacerlo.
4. Recoger feedback sobre características y funcionalidades deseadas.

Estructura de la Entrevista

Introducción:

- ¿Cual es tu nombre?

- ¿Qué edad tienes?
- ¿A qué te dedicas?

Preguntas para Creadores de Contenido:**De Contexto:**

- ¿Qué información consideras clave para incluir en tu perfil, tanto personal como profesionalmente?
- ¿Cuán importante es para ti la posibilidad de programar publicaciones futuras? ¿En qué situaciones lo utilizarías?
- ¿Qué te parece más atractivo: ofrecer suscripciones mensuales o recibir donaciones únicas? ¿Por qué?

Sobre Problemas y Desafíos:

- ¿Qué tan fácil te resulta editar y gestionar tu perfil en las plataformas que utilizas actualmente?
- ¿Qué funcionalidades debería incluir un editor de texto enriquecido para hacer tu proceso de creación más eficiente?
- ¿Cómo sueles ajustar tu contenido basado en las estadísticas que recibes? ¿Qué datos adicionales te ayudarán en este proceso?
- ¿Qué mecanismos de interacción entre la comunidad crees que pueden aumentar el compromiso de tus lectores?

Sobre Expectativas y Deseos:

- ¿Qué opciones adicionales de personalización te gustaría tener en tu perfil para destacar tus intereses y experiencia?
- ¿Te gustaría tener la opción de previsualizar tu contenido antes de publicarlo? ¿Qué otros pasos previos consideramos importantes?
- ¿Con qué frecuencia te gustaría recibir reportes de tus publicaciones y su rendimiento?
- ¿Cómo te gustaría visualizar los reportes (gráficos, tablas, resúmenes)? ¿Qué tipo de formato te parece más claro y útil?
- ¿Cómo valoras la posibilidad de incluir elementos multimedia (imágenes, videos, enlaces) en tus publicaciones?

Sobre Adopción y Usabilidad:

- ¿Qué tan importante es para ti interactuar directamente con tus lectores a través de comentarios u otros medios?

Sobre Pagos y Modelos de Negocio:

- ¿Cómo prefieres gestionar tus ingresos obtenidos de la monetización de tu contenido?
- ¿Qué métodos de retiro te resultan más convenientes?

Preguntas para Lectores:**De Contexto:**

- ¿Qué información te gustaría ver en el perfil de los creadores de contenido para decidir si sigues o apoyas a alguien?
- ¿Qué características buscas en un artículo o publicación para que te mantenga interesado?
- ¿Qué te motivaría a suscribirte a contenido exclusivo en lugar de consumir contenido gratuito?
- ¿Qué beneficios esperas al apoyar económicamente a un creador? ¿Cómo influye esto en tu decisión de apoyar a alguien?
- ¿Te gustaría formar parte de una comunidad activa en torno a los creadores que sigues? ¿Qué funcionalidades crees que fortalecerán ese sentido de comunidad?

Sobre Problemas y Desafíos:

- ¿Te parece más cómodo realizar donaciones únicas y suscribirte mensualmente? ¿Por qué?

Sobre Expectativas y Deseos

- ¿Te gustaría recibir recomendaciones personalizadas de contenido basado en tus intereses? ¿Cómo prefieres que se muestren estas recomendaciones?
- ¿Qué tan relevante es para ti que los artículos incluyan multimedia (imágenes, videos, enlaces)? ¿Mejora tu experiencia de lectura?
- ¿Qué tan importante es para ti poder interactuar directamente con los creadores (comentarios, mensajes)?

Sobre Adopción y Usabilidad:

- ¿Te gustaría tener la opción de guardar artículos para leer más tarde o programar recordatorios?
- ¿Prefieres compartir contenido que lees en redes sociales? ¿Qué te facilita o dificulta hacerlo?

Sobre Pagos y Modelos de Negocio:

- ¿Qué métodos de pago prefieres utilizar para realizar donaciones o suscripciones en plataformas de contenido?

2.2. Registro de entrevistas

Datos del Entrevistado	Método de la Entrevista	Enlace de YouTube
Nombre: Arnold Mucha Castillo. Edad: 22. Ocupación: Estudiante Universitario. Fecha de la entrevista: 04/09/2024	Videoconferencia	Enlace
Nombre: Claudia Mercedes Yzquierdo Monzon. Edad: 21. Ocupación: Estudiante Universitario. Fecha de la entrevista: 04/09/2024	Videoconferencia	Enlace
Nombre: Fernando Castillo Robles. Edad: 45. Ocupación: Docente de la Facultad de Ingeniería. Fecha de la entrevista: 04/09/2024	Videoconferencia	Enlace
Nombre: Alessandra Agredo Vega. Edad: 20. Ocupación: Estudiante Universitario. Fecha de la entrevista: 03/09/2024	Videoconferencia	Enlace
Nombre: Rodrigo Agreda Iparraguirre. Edad: 20. Ocupación: Estudiante Universitario. Fecha de la entrevista: 11/09/2024	Videoconferencia	Enlace
Nombre: Daniel Campos. Edad: 20. Ocupación: Estudiante Universitario. Fecha de la entrevista: 12/09/2024	Videoconferencia	Enlace
Nombre: Faviana Cruz Baltazar. Edad: 20. Ocupación: Estudiante Universitario. Fecha de la entrevista: 15/09/2024	Videoconferencia	Enlace

2.3. Análisis de entrevistas

2.3.1. Creadores

Hallazgo 1: Falta de interacción significativa en plataformas para creadores de contenido

- La mayoría de los creadores entrevistados mencionaron que la interacción directa con el público es esencial, y que las plataformas actuales no ofrecen suficientes mecanismos para que los creadores y su comunidad se sientan conectados.
- **Insight:** Las plataformas deberían contar con una sección de comentarios para que la interacción entre creadores y su audiencia puedan mejorar significativamente la lealtad y el compromiso de los usuarios. Esto generará una comunidad más sólida y ayudará a los creadores a mantener a sus seguidores comprometidos.

Hallazgo 2: Preferencia por suscripciones debido a beneficios

- La mayoría de los creadores entrevistados expresó que, aunque prefieren las donaciones porque el dinero es más directo, consideran que para el público en general las suscripciones son más atractivas debido a los beneficios adicionales que ofrecen, como destacarse entre otros seguidores.
- **Insight:** Incluir un sistema de suscripciones para destacar en la comunidad, incentiva a más usuarios a suscribirse. Esto no solo generará ingresos constantes para los creadores, sino que también aumentará el sentido de pertenencia de la comunidad.

2.3.2. Lectores

Hallazgo 1: Preferencia por recomendaciones personalizadas

- La mayoría de los entrevistados indicaron que les gustaría recibir recomendaciones de contenido basadas en sus intereses y comportamiento en la plataforma. Mencionaron que prefieren recibir este tipo de contenido de forma automática cuando su creador favorito no está activo.
- **Insight:** La plataforma debería implementar un sistema de recomendaciones personalizadas que sugiera contenido basado en el historial y las preferencias de los usuarios. Esto ayudaría a retener a los usuarios y ofrecerles contenido relevante en todo momento, mejorando su experiencia de navegación y evitando la pérdida de interés en la plataforma.

Hallazgo 2: Necesidad de guardar y agrupar contenido multimedia

- Varios entrevistados mencionaron que les gustaría poder agrupar y guardar sus videos a modo de listas personalizadas, ya que a menudo encuentran contenido interesante pero no tienen tiempo en el momento para revisarlo por lo que les facilitaría el poder agruparlos.
- **Insight:** Implementar una opción para guardar y agrupar contenido sugiere una oportunidad importante para mejorar la experiencia de los

usuarios. Implementar una función que permita almacenar y organizar contenido podría aumentar el tiempo de permanencia en la plataforma y fomentar una mayor interacción con los creadores. Además, esto le proporcionaría a los usuarios una herramienta para gestionar su consumo de contenido de manera más eficiente, manteniendo su interés a largo plazo.

Hallazgo 3: Preferencia por donaciones únicas con interacción inmediata

- Muchos usuarios mencionaron que prefieren realizar donaciones únicas, ya que estas suelen estar acompañadas de animaciones o sonidos que les permiten sentir una mayor interacción con el creador. En comparación, las suscripciones mensuales parecen menos dinámicas.
- **Insight:** La plataforma debería incentivar el uso de donaciones únicas mediante la implementación de animaciones o sonidos personalizados. Esto mejorará la experiencia del donante, haciendo que su contribución sea más visible e interactiva, lo cual puede motivar a más usuarios a realizar donaciones de forma recurrente.

Hallazgo 4: Interacción directa con los creadores

- Los usuarios valoran la posibilidad de interactuar directamente con los creadores mediante comentarios, notificaciones. Consideran que esto fortalece el sentido de comunidad y los hace sentir más cercanos a los creadores.
- **Insight:** Para fomentar la participación y el sentido de pertenencia en la comunidad, la plataforma debería facilitar herramientas que permitan la comunicación entre creadores y lectores. Esto incluye la opción de comentarios en cada publicación de videos o blogs.

Hallazgo 5: Preferencia por métodos de pago rápidos y seguros

- Varios usuarios prefieren métodos de pago rápidos y seguros como PayPal, billeteras digitales (Yape, Plin), que permitan realizar donaciones o suscripciones sin complicaciones. También mencionan la importancia de garantizar la seguridad de la información.
- **Insight:** Es necesario implementar métodos de pago ágiles y seguros en la plataforma para facilitar las donaciones y suscripciones. Además, brindar confianza a los usuarios mediante políticas de seguridad robustas es crucial para fomentar la adopción de las funcionalidades de monetización.

Capítulo III: Requirements Specification

3.1. EPICs

Nro EPIC	Título	Descripción
EPIC 01	Gestión de Usuario	Implementar un sistema de gestión de usuarios para que las personas puedan registrar, editar, configurar sus contraseñas y visualizar perfiles de usuarios para conocer más de ellos.
EPIC 02	Gestión de Contenido	Desarrollar e implementar un sistema donde se pueda acceder a un editor multimedia donde se pueda insertar la misma previsualizando el contenido dándole la opción de programar la subida y también de una vez subida eliminar dicho contenido.
EPIC 03	Gestión de Ingresos y Monetización	Establecer un sistema donde los creadores generen planes de suscripción para su comunidad y los usuarios puedan donarles a distintos creadores de contenido, a su vez los creadores podrán ver sus ingresos, retirar sus fondos mediante la elección de métodos de pago.
EPIC 04	Gestión de Reporte	Construir un sistema de estadísticas para que los creadores puedan ver las estadísticas de sus publicaciones, de sus interacciones generando reportes y para ayudar a su análisis, generar comparativa del rendimiento y dándole consejos para que optimice su contenido.
EPIC 05	Gestión de Comunidad	Crear un sistema donde los usuarios podrán comentar el contenido, reaccionar al mismo, haciendo que el creador reciba una notificación cuando esto ocurra, además los usuarios podrán crear una lista de reproducción y podrán suscribirse a estos creadores, si ya no desean continuar podrán cancelar dicha suscripción.

3.2. User Stories & Criterios de Aceptación

Nro EPIC	Nro. US	Título	Descripción	Criterios de Aceptación
EPIC 01	US 01	Registro de Usuarios	Como usuario, quiero poder registrarme en la plataforma para acceder a las funcionalidades disponibles. El formulario de registro debe contener campos obligatorios como nombre, correo electrónico, contraseña, confirmación de contraseña y rol.	<p>Escenario 1: Registro de un nuevo usuario</p> <p>Dado que un usuario no registrado accede al formulario de registro,</p> <p>Cuando envía una solicitud con su nombre, correo electrónico, contraseña válida y selecciona su rol.</p> <p>Entonces el sistema crea una nueva cuenta de usuario en la base de datos y otorga acceso a la plataforma.</p>
EPIC 01	US 02	Visualización de Perfil	Como usuario, quiero ver mi perfil para ver mi información personal. Se podrá acceder a la página de su perfil a través de un enlace en la foto de su perfil en la barra de navegación. Se mostrará información personal del usuario, como nombre, correo electrónico y foto de perfil.	<p>Escenario 1: Acceso a perfil del usuario</p> <p>Dado que el usuario ha iniciado sesión,</p> <p>Cuando accede a la sección de su perfil,</p> <p>Entonces el sistema muestra su información como nombre, correo electrónico, rol y foto de perfil.</p>
EPIC 01	US 03	Editar Perfil	Como usuario, quiero editar mi perfil para mantener actualizada mi información personal y preferencias. Se podrá acceder a la opción de edición de perfil desde un botón visible en la página de su perfil, esta opción debe permitir modificar campos como nombre, correo electrónico, contraseña, biografía, foto de perfil y cambiar de rol "CREATOR"(si desea el usuario, no se puede viceversa).	<p>Escenario 1: Modificación de información personal</p> <p>Dado que un usuario ha iniciado sesión y accede a su perfil,</p> <p>Cuando edita su nombre, biografía , correo electrónico, password, o cambia de rol y envía la solicitud,</p> <p>Entonces el sistema actualiza correctamente la información en la base de datos.</p>
EPIC 01	US 04	Inicio de sesión	Se debe permitir que los usuarios registrados inicien sesión en el sistema	Escenario 1: Inicio de sesión exitoso

			<p>proporcionando su correo electrónico y contraseña. Si las credenciales son correctas, se debe autenticar al usuario y permitir el acceso. Si son incorrectas, se debe rechazar la solicitud con un mensaje de error adecuado.</p>	<p>Dado que un usuario registrado ingresa su email y contraseña correctamente Cuando envía la solicitud de inicio de sesión Entonces el sistema autentica al usuario y devuelve un mensaje de éxito.</p> <p>Escenario 2: Fallo en la autenticación por credenciales incorrectas</p> <p>Dado que un usuario ingresa un email o contraseña incorrectos Cuando envía la solicitud de inicio de sesión Entonces el sistema debe devolver un error 401 indicando "Credenciales incorrectas".</p>
EPIC 01	US 05	Automatizar rol "Lector" en la creación del usuario	<p>Como usuario que se registra en la plataforma, quiero que mi rol se asigne automáticamente como "Lector" durante el proceso de creación de cuenta, para que pueda comenzar a usar la plataforma sin tener que seleccionar un rol manualmente.</p>	<p>Escenario 1: Asignación automática del rol "Lector"</p> <p>Dado que un usuario no registrado accede al formulario de registro, Cuando el usuario completa el formulario de registro y lo envía, entonces el sistema debe asignar automáticamente el rol de "Lector" al usuario sin que sea necesario seleccionarlo manualmente.</p> <p>Escenario 2: Confirmación del rol "Lector" en la base de datos</p> <p>Dado que un usuario ha completado el proceso de registro, Cuando el sistema crea la cuenta del usuario, entonces el rol de "Lector" debe estar correctamente almacenado en la base de datos y asociado al usuario.</p>

EPIC 01	US 06	Registro de usuario con Google	<p>Como usuario, quiero poder registrarme en la plataforma utilizando mi cuenta de Google, para agilizar el proceso de registro sin tener que ingresar manualmente mis datos.</p>	<p>Escenario 1: Opción de registro con Google visible</p> <p>Dado que un usuario no registrado accede a la página de registro, Cuando el usuario visualiza el formulario de registro, entonces el sistema debe mostrar un botón visible de "Registrarse con Google" junto al formulario estándar de registro.</p> <p>Escenario 2: Autenticación y permisos de Google</p> <p>Dado que un usuario ha hecho clic en el botón de "Registrarse con Google", Cuando el sistema redirige al usuario a la autenticación de Google, entonces el sistema debe solicitar los permisos necesarios para obtener la información básica del perfil (nombre, correo electrónico).</p> <p>Escenario 3: Registro exitoso con Google</p> <p>Dado que el usuario ha autorizado los permisos solicitados por Google, Cuando el sistema recibe la información del perfil de Google, Entonces el sistema debe crear una cuenta en la plataforma utilizando el correo electrónico, nombre y otros datos obtenidos, asignando automáticamente el rol de "Lector" al usuario.</p> <p>Escenario 4: Error en la autenticación de Google</p> <p>Dado que un usuario ha iniciado el proceso de registro con Google, Cuando hay un error en la autenticación o el usuario deniega los permisos, entonces el sistema debe</p>
---------	-------	--------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

				mostrar un mensaje de error indicando que el registro no se pudo completar debido a un problema con la autenticación.
EPIC 02	US 07	Crear categoría	Como creador, quiero poder crear categorías para organizar mejor el contenido de la plataforma, debe tener acceso a una opción para "Crear Categoría" desde el panel de administración de contenido.	<p>Escenario 1: Creación de una nueva categoría</p> <p>Dado que un creador ha iniciado sesión en la plataforma, Cuando accede al formulario de creación de categorías e ingresa el nombre de una nueva categoría, entonces el sistema guarda la nueva categoría en la base de datos y esta se muestra en la lista de categorías disponibles para organizar contenido.</p>
EPIC 02	US 08	Visualizar categoría	Como Usuario, quiero poder visualizar las categorías disponibles en la plataforma para explorar el contenido de acuerdo a mis intereses.	<p>Escenario 1: Visualización de categorías disponibles</p> <p>Dado que un usuario ha iniciado sesión o está navegando como invitado, cuando accede a la sección de categorías, entonces el sistema muestra una lista de todas las categorías disponibles.</p>
EPIC 02	US 09	Editar Categoría	Se debe permitir que un creador de contenido pueda editar los detalles de una Categoría ya creada. Solo los usuarios con el rol de CREADOR pueden actualizar una categoría. Se deben poder actualizar campos como nombre y descripción siempre y cuando existan en la base de datos.	<p>Escenario 1: Actualizar una categoría</p> <p>Dado que el creador es un usuario con rol CREADOR Cuando envía una solicitud con los nuevos datos de la categoría Entonces el sistema debe actualizar nombre y la descripción de la categoría.</p>
EPIC 02	US 10	Crear Blog	Como creador, quiero poder eliminar mis publicaciones si ya no deseo que estén disponibles, el creador debe tener acceso a una	<p>Escenario 1: Eliminación de la publicación</p> <p>Dado que un creador desea eliminar una publicación</p>

			<p>opción visible para eliminar una publicación desde su perfil.</p> <p>Cuando el creador accede a su perfil y selecciona una publicación específica que desea eliminar. Entonces el sistema debe mostrar una ventana de confirmación para evitar eliminaciones accidentales. Y el creador confirma que desea eliminar la publicación y el sistema elimina el video.</p> <p>Escenario 2: Eliminación de un video</p> <p>Dado que un creador desea eliminar un video de una publicación. Cuando el creador accede a la vista de edición de su publicación y selecciona el video que desea eliminar. Entonces el sistema debe mostrar una ventana de confirmación para evitar eliminaciones accidentales del video. Y el creador confirma que desea eliminar el video. El sistema elimina el video de la publicación, tanto de la base de datos como del servidor de almacenamiento de archivos. Y muestra un mensaje de éxito: "El video ha sido eliminado correctamente."</p> <p>Escenarios 3: Cancelar eliminación</p> <p>Dado que el creador selecciona la opción "Eliminar" en una publicación. Cuando el sistema muestra la ventana de confirmación. El creador decide no proceder con la eliminación. Entonces el creador cancela el proceso de eliminación, y la publicación permanece en la plataforma sin cambios.</p>
--	--	--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

			<p>Y el sistema no realiza ninguna acción adicional.</p> <p>Escenario 4: Eliminación de una imagen</p> <p>Dado que un creador desea eliminar una imagen de una publicación. Cuando el creador accede a la vista de edición de su publicación y selecciona la imagen que desea eliminar. Entonces el sistema debe mostrar una ventana de confirmación para evitar eliminaciones accidentales de la imagen. Y el creador confirma que desea eliminar la imagen. El sistema elimina la imagen de la publicación, tanto de la base de datos como del servidor de almacenamiento de archivos. Y muestra un mensaje de éxito: "La imagen ha sido eliminada correctamente."</p> <p>Escenario 5: Error en la conexión durante la eliminación</p> <p>Dado que el creador confirma la eliminación de la publicación. Cuando ocurre un error de conexión mientras el sistema está procesando la eliminación. Entonces el sistema muestra un mensaje de error: "Error en la conexión. No se pudo eliminar la publicación. Por favor, inténtalo nuevamente." Y la publicación permanece en la plataforma hasta que el proceso sea completado exitosamente.</p>
EPIC 02	US 11	Editar Blog	<p>Se debe permitir que un creador de contenido pueda actualizar los</p> <p>Escenario 1: Actualizar un blog</p>

			<p>detalles de un blog ya creado. Solo los usuarios con el rol de CREATOR pueden actualizar un blog. Se deben poder actualizar campos como el título, contenido, y categoría del blog, siempre y cuando existan en la base de datos.</p>	<p>Dado que el creador es un usuario con rol CREATOR Cuando envía una solicitud con los nuevos datos del blog Entonces el sistema debe actualizar el título, contenido, y categoría del blog.</p> <p>Escenario 2: Verificar categoría existente</p> <p>Dado que el creador envía un category_id Cuando la categoría no existe en la base de datos Entonces el sistema devuelve un error indicando que la categoría no ha sido encontrada.</p>
EPIC 02	US 12	Programación de Blog	<p>Como creador, quiero poder programar mis publicaciones para que se publiquen automáticamente en una fecha y hora específicas, por esto debe haber una opción para "Programar publicación" en lugar de publicarla inmediatamente.</p>	<p>Escenario 1: Programar publicación de blogs</p> <p>Dado que un creador de contenido quiere programar la publicación de su blog Cuando el creador introduce una fecha y hora futuras en el campo schedulePublishAt Entonces el sistema debe guardar el blog como "no publicado" y programar la publicación para la hora indicada.</p> <p>Escenario 2: Actualizar un blog con programación de publicación</p> <p>Dado que un creador de contenido quiere actualizar un blog programado para una futura publicación Cuando el creador envía nuevos datos como título, contenido, o una nueva fecha de publicación en el campo schedulePublishAt Entonces el sistema debe actualizar el blog y mantener o cambiar la programación de</p>

				<p>publicación según los nuevos datos proporcionados.</p> <p>Escenario 3: Validación de categoría en la creación y actualización de blogs</p> <p>Dado que un creador de contenido está creando o actualizando un blog</p> <p>Cuando el creador asigna una categoría que no existe en la base de datos</p> <p>Entonces el sistema debe devolver un mensaje de error indicando que la categoría no ha sido encontrada.</p> <p>Escenario 4: Publicación automática de blogs programados</p> <p>Dado que un blog ha sido programado para publicarse en una fecha futura</p> <p>Cuando la fecha y hora actuales alcanzan el valor de schedulePublishAt</p> <p>Entonces el sistema debe publicar automáticamente el blog y marcarlo como "publicado".</p>
EPIC 02	US 13	Crear Video	Como creador, quiero poder insertar imágenes y videos a mis publicaciones para hacerlas más atractivas e informativas. El creador debe poder seleccionar la opción para agregar imágenes o videos desde el editor de publicación.	<p>Escenario 1: Creación exitosa del video</p> <p>Dado que el creador desea insertar un video en su publicación.</p> <p>Cuando el creador accede al editor de publicaciones y selecciona la opción para agregar una imagen desde su dispositivo.</p> <p>Entonces el sistema carga el video correctamente en la publicación.</p>
EPIC 02	US 14	Registro de Video	Como creador, quiero registrar videos para compartir contenido	Escenario 1: Registro exitoso del video

			<p>multimedia con la comunidad. El sistema debe permitir la carga y el registro de videos con formatos y tamaños específicos, asegurando que el contenido esté disponible en mi perfil o publicaciones.</p>	<p>Dado que un creador desea registrar un video en la plataforma. Cuando el creador accede a su perfil o al editor de publicaciones y selecciona la opción para cargar un video. Entonces el sistema solicita al creador que seleccione un archivo de video desde su dispositivo. Y el creador selecciona un video en formato avi, mp4. El sistema registra el video, lo almacena en el servidor y lo asocia con el perfil o publicación del creador. Se muestra un mensaje de éxito: "El video ha sido registrado exitosamente."</p>
EPIC 02	US 15	Editar Video	<p>Como creador, quiero poder editar un video ya registrado, o reemplazar el video con una nueva versión, asegurándome de que las modificaciones sean actualizadas correctamente.</p>	<p>Escenario 1: Edición exitosa del video</p> <p>Dado que un creador desea editar los metadatos (título y descripción) de un video registrado. Cuando el creador accede a la opción de "Editar video" desde su perfil o la publicación donde se encuentra el video. Entonces el sistema permite al creador modificar el título y la descripción del video. Y el creador guarda los cambios.</p>
EPIC 02	US 16	Programación de Video	<p>Como creador, quiero poder programar mis publicaciones para que se publiquen automáticamente en una fecha y hora específicas, por esto debe haber una opción para "Programar publicación" en lugar de publicarla inmediatamente.</p>	<p>Escenario 1: Programar publicación de videos</p> <p>Dado que un creador de contenido quiere programar la publicación de su video. Cuando el creador introduce una fecha y hora futuras en el campo schedulePublishAt. Entonces el sistema debe guardar el video como "no publicado" y programar la</p>

				<p>publicación para la hora indicada.</p> <p>Escenario 2: Actualizar un video con programación de publicación</p> <p>Dado que un creador de contenido quiere actualizar un video programado para una futura publicación</p> <p>Cuando el creador envía nuevos datos en el campo schedulePublishAt</p> <p>Entonces el sistema debe actualizar el video y mantener o cambiar la programación de publicación según los nuevos datos proporcionados.</p> <p>Escenario 3: Validación de categoría en la creación y actualización de videos</p> <p>Dado que un creador de contenido está creando o actualizando un video</p> <p>Cuando el creador asigna una categoría que no existe en la base de datos</p> <p>Entonces el sistema debe devolver un mensaje de error indicando que la categoría no ha sido encontrada.</p> <p>Escenario 4: Publicación automática de videos programados</p> <p>Dado que un video ha sido programado para publicarse en una fecha futura</p> <p>Cuando la fecha y hora actuales alcanzan el valor de schedulePublishAt</p> <p>Entonces el sistema debe publicar automáticamente el video y marcarlo como "publicado".</p>
--	--	--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

EPIC 02	US 17	Eliminar Video	Como creador, quiero poder eliminar mis publicaciones si ya no deseo que estén disponibles, el creador debe tener acceso a una opción visible para eliminar una publicación desde su perfil.	Escenario 1: Eliminación del video Dado que un creador desea eliminar un video de una publicación. Cuando el creador accede a la vista de edición de su publicación y selecciona el video que desea eliminar. Entonces el sistema debe mostrar una ventana de confirmación para evitar eliminaciones accidentales del video. Y el creador confirma que desea eliminar el video. El sistema elimina el video de la publicación, tanto de la base de datos como del servidor de almacenamiento de archivos. Y muestra un mensaje de éxito: " <i>El video ha sido eliminado correctamente.</i> "
EPIC 02	US 18	Crear Playlist	Como usuario, quiero poder crear una playlist para organizar y guardar videos de mi interés, facilitando el acceso a contenido seleccionado en cualquier momento.	Escenario 1: Creación de una nueva playlist Dado que un usuario ha iniciado sesión, Cuando accede a la sección de creación de playlist y proporciona un nombre válido, Entonces el sistema crea una nueva playlist y la asocia a su cuenta.
EPIC 02	US 19	Visualizar playlist	Como usuario, quiero poder acceder a una lista de todas las playlists que he creado y ver el contenido de una playlist específica, para poder gestionar y visualizar los elementos que he añadido a mis playlists de forma fácil y organizada.	Escenario 1: Listado de playlists del usuario Dado que un usuario ha iniciado sesión, Cuando accede a la sección de playlists, Entonces el sistema muestra una lista de todas las playlists asociadas a su cuenta.
EPIC 02	US 20	Editar playlist	Como usuario, quiero poder modificar las playlists que han creado previamente, lo	Escenario 1: Actualización del nombre de una playlist

			que incluye la posibilidad de actualizar el nombre de la playlist para permitir mi playlist organizada y ajustada a mis gustos	Dado que un usuario tiene una playlist existente, Cuando edita el nombre de la playlist y envía la solicitud, Entonces el sistema actualiza el nombre de la playlist correctamente en la base de datos.
EPIC 02	US 21	Eliminar playlist	Como usuario, quiero tener la opción de eliminar una playlist que ya no deseo, para mantener en mi cuenta solo las playlist relevantes	<p>Escenario 1: Eliminación exitosa de una playlist</p> <p>Dado que un usuario tiene una playlist creada, Cuando selecciona la opción de eliminar la playlist y confirma la acción, Entonces el sistema borra la playlist de la base de datos y la elimina de su cuenta.</p>
EPIC 02	US 22	Añadir video a la lista de reproducción del usuario	Como usuario, quiero poder añadir videos a mi lista de reproducción para acceder fácilmente a ellas más tarde, habrá un botón visible en cada video para agregarla a su lista de reproducción personal.	<p>Escenario 1: Visualización del botón "Añadir a la lista de reproducción"</p> <p>Dado que un usuario ha iniciado sesión en la plataforma y está navegando por las publicaciones, Cuando el usuario visualiza una publicación que le interesa, entonces el sistema debe mostrar un botón visible de "Añadir a la lista de reproducción" en cada publicación.</p> <p>Escenario 2: Añadir una publicación a la lista de reproducción personal</p> <p>Dado que un usuario ha hecho clic en el botón de "Añadir a la lista de reproducción" en una publicación, Cuando el sistema procesa la solicitud, Entonces la publicación debe ser añadida a la lista de reproducción personal del usuario, y el sistema debe mostrar una confirmación de que la</p>

				publicación ha sido añadida correctamente.
EPIC 02	US 23	Ver la lista de videos en la lista de reproducción del usuario	Como usuario, quiero poder ver todas los videos en mi lista de reproducción para gestionar mi contenido guardado teniendo acceso a su lista de reproducción desde mi perfil. La lista de reproducción debe mostrar todos los videos que el usuario ha guardado, con el título, categoría, y la fecha en que fue añadida.	<p>Escenario 1: Acceso a la lista de reproducción</p> <p>Dado que un usuario ha iniciado sesión en la plataforma y está en su página de perfil, Cuando el usuario hace clic en la opción de "Lista de reproducción", Entonces el sistema debe redirigir al usuario a una página que muestra todas las playlists creadas</p> <p>Escenario 2: Visualización de publicaciones en la lista de reproducción</p> <p>Dado que un usuario selecciona una "Lista de reproducción", Cuando se carga la página, Entonces el sistema debe mostrar una lista de todos los videos que el usuario ha guardado, incluyendo la siguiente información Título Categoría Fecha en que fue añadida.</p>
EPIC 02	US 24	Eliminar video de la lista de reproducción del usuario	Como usuario, quiero poder eliminar videos de mi lista de reproducción cuando ya no me interesen, desde la lista de reproducción mediante un botón de "Eliminar" en cada video guardado. Al seleccionar "Eliminar", el sistema procesa la solicitud de eliminación y lo elimina de la lista de reproducción	<p>Escenario 1: Eliminación exitosa de video de la Lista de Reproducción Dado que un usuario ha iniciado sesión en la plataforma y está en su lista de reproducción Cuando el usuario visualiza los videos y selecciona "Eliminar" Entonces el sistema procesa la solicitud de eliminación y el video seleccionada debería desaparecer de la lista de reproducción del usuario.</p>
EPIC 02	US 25	Visualizar el historial de publicaciones	Como usuario, quiero poder visualizar el historial del contenido que he consumido en la plataforma en orden cronológico, para	Escenario 1: Acceso al historial de publicaciones consumidas

			así poder saber que tipo de contenido he consumido	Dado que un usuario ha iniciado sesión en la plataforma, Cuando el usuario accede a su perfil y selecciona la opción de "Historial de publicaciones", Entonces el sistema debe redirigir al usuario a una página que muestra todas las publicaciones (blog o video) que ha consumido, ordenadas cronológicamente con las más recientes al inicio.
EPIC 02	US 26	Recomendaciones personalizadas	Como usuario, quiero recibir recomendaciones personalizadas de contenido, para descubrir publicaciones (blogs o videos) que sean de mi interés, basadas en mi historial de consumo.	Escenario 1: Generación de recomendaciones basadas en el historial Dado que un usuario ha consumido contenido en la plataforma, Cuando accede a la página de inicio o una sección dedicada a recomendaciones, Entonces, el sistema debe mostrar publicaciones recomendadas (blogs o videos) basadas en las preferencias y el historial de consumo del usuario.
EPIC 02	US 27	Registro de publicaciones	Como creador, quiero tener un registro de todas las publicaciones realizadas en la plataforma para fines de moderación y análisis.	El creador debe tener acceso en su perfil donde se muestren todas las publicaciones que ha realizado. El registro debe incluir el título, fecha de creación, categoría y estado (publicado, borrador, eliminado). Cada entrada en el registro debe ser clicable para permitir al creador ver, editar o eliminar la publicación. El sistema debe mostrar un mensaje si el creador no ha realizado ninguna publicación.
EPIC 02	US 28	Previsualización de publicación	Como creador de contenido, quiero poder previsualizar mi publicación antes de compartirla, para asegurarme de que el	Escenario 1: Visualización de la publicación en vista previa Dado que un creador está redactando o editando una publicación,

			<p>contenido se vea como deseo antes de que sea publicado.</p>	<p>Cuando selecciona la opción de "Previsualizar", Entonces el sistema debe mostrar una vista previa de la publicación con el formato exacto que tendrá una vez publicada</p> <p>Escenario 2: Cierre de la previsualización</p> <p>Dado que un creador está revisando la vista previa de una publicación, Cuando el creador cierra la vista previa, Entonces el sistema debe devolverlo al editor de la publicación, permitiéndole continuar editando o proceder con la publicación.</p>
EPIC 02	US 29	Filtrar categorías por criterios específicos	Como usuario, quiero poder filtrar las publicaciones por categorías específicas, para encontrar el contenido que me interesa más fácilmente desde la página de inicio.	<p>Escenario 1: Filtrado por categoría</p> <p>Dado que un usuario desea encontrar publicaciones de una categoría específica, Cuando el usuario selecciona una o más categorías desde la lista desplegable en la página de búsqueda, Entonces el sistema debe mostrar todas las publicaciones que pertenezcan a esas categorías de manera ordenada.</p>
EPIC 03	US 30	Suscripción a creadores	Como usuario, quiero suscribirme a un creador de contenido para brindarle apoyo monetario y recibir los beneficios que el creador ofrece a sus suscriptores.	<p>Escenario 1: Suscripción exitosa</p> <p>Dado que un usuario con el rol READER desea apoyar monetariamente a un creador, Cuando el usuario envía una solicitud de suscripción, entonces el sistema debe verificar que no haya una suscripción activa y crear una nueva suscripción con una duración de 30 días.</p>

				Escenario 2: Suscripción rechazada por duplicado Dado que un usuario intenta suscribirse a un creador con el que ya tiene una suscripción activa, Cuando el usuario envía la solicitud de suscripción, entonces el sistema debe rechazar la solicitud indicando que ya existe una suscripción activa con ese creador.
EPIC 03	US 31	Cancelar Suscripción	Como usuario, quiero cancelar mi suscripción a un creador para dejar de apoyar monetariamente al finalizar el ciclo de pago.	Escenario 1: Cancelación de suscripción exitosa Dado que un usuario tiene una suscripción activa con un creador, Cuando el usuario cancela su suscripción, Entonces el sistema debe marcar la suscripción como inactiva, pero permitir que continúe hasta el final del ciclo de pago de 30 días. Escenario 2: Intento de cancelación de una suscripción inactiva Dado que un usuario intenta cancelar una suscripción que ya está inactiva, Cuando el usuario envía la solicitud de cancelación, Entonces el sistema debe rechazar la solicitud indicando que la suscripción ya ha sido cancelada.
EPIC 03	US 32	Editar Planes de Suscripciones	Se debe permitir que un usuario pueda ver los detalles de los planes ya creados, para esto el sistema debe tener acceso a un panel donde pueda ver, crear o editar los	Escenario 1: Edición de planes de suscripciones Dado que un usuario desea ver los detalles de los planes de suscripción. Cuando el usuario accede al panel de suscripciones

			planes de suscripción estandarizados.	Entonces el sistema muestra los planes disponibles con los detalles actuales.
EPIC 03	US 33	Registro de compras	Como usuario, quiero poder realizar y registrar una compra de un plan en la plataforma. El sistema debe permitir el registro de la compra, almacenar los detalles del pago y generar una confirmación para el usuario.	<p>Escenario 1: Registro exitoso de compras</p> <p>Dado que un usuario desea realizar una compra. Cuando el usuario selecciona un plan, ingresa la cantidad deseada, elige el método de pago, y completa los detalles de la dirección de envío. Entonces el sistema valida que toda la información esté completa y correcta. Y el sistema procesa el pago y registra la compra en la base de datos del sistema. El sistema muestra un mensaje de éxito: "La compra ha sido registrada exitosamente."</p>
EPIC 03	US 34	Enviar Donaciones	Como usuario, quiero poder enviar donaciones a los creadores de contenido que aprecio para apoyar su trabajo teniendo la opción de donar visible en todas sus publicaciones.	<p>El sistema debe permitir que el usuario seleccione el monto de la donación y el método de pago preferido.</p> <p>El sistema debe proporcionar una confirmación del envío de la donación y notificar tanto al usuario como al creador.</p> <p>El usuario debe recibir un recibo o comprobante de la donación realizada.</p>
EPIC 03	US 35	Visualizar lista de donaciones	Como creador, quiero poder visualizar una lista de las donaciones que he recibido para conocer quiénes me han apoyado y cuánto han donado.	Dado que un creador quiere revisar sus donaciones, Cuando el creador solicita ver la lista de donaciones, Entonces el sistema debe mostrar una lista de todas las donaciones recibidas, incluyendo detalles relevantes como monto, donante y fecha

EPIC 03	US 36	Visualizar monto total de las donaciones	Como creador, quiero poder visualizar el monto total de todas las donaciones que he recibido para tener claro el apoyo económico que he recibido.	Dado que un creador quiere saber el total de donaciones recibidas, Cuando el creador solicita el monto total, Entonces el sistema debe calcular y mostrar el monto total de todas las donaciones recibidas.
EPIC 03	US 37	Integrar funcionalidad completa de pagos con PayPal	Como usuario, quiero poder utilizar PayPal para realizar y recibir pagos de forma segura y confiable, para tener una opción de pago que sea ampliamente aceptada y sencilla de usar.	<p>Escenario 1: Selección de PayPal como método de pago</p> <p>Dado que un usuario desea realizar un pago, cuando el usuario selecciona PayPal como método de pago durante el proceso de suscripción o donación, entonces el sistema debe redirigir al usuario a PayPal para la autenticación y confirmación del pago.</p>
EPCI 03	US 38	Retiro de fondos	Como creador, quiero poder retirar los fondos generados por suscripciones y donaciones a mi cuenta bancaria o método de pago preferido, para acceder a mis ingresos de manera sencilla y segura.	<p>Escenario 1: Retiro de fondos para creadores mediante PayPal</p> <p>Dado que un creador de contenido desea retirar sus fondos generados, Cuando selecciona PayPal como método de retiro, entonces el sistema debe permitirle retirar sus fondos a su cuenta de PayPal, siguiendo el proceso de confirmación de identidad.</p>
EPIC 04	US 39	Generación de reportes básicos de publicaciones por categoría	Como creador, quiero generar reportes básicos de las publicaciones por categoría, para analizar el rendimiento y la popularidad de mis publicaciones.	<p>Escenario 1: Generación de reporte por categoría</p> <p>Dado que un creador desea analizar sus publicaciones, Cuando selecciona la opción de "Generar Reporte" y elige una categoría específica, Entonces el sistema debe generar un reporte que muestre todas las publicaciones de esa categoría, con información sobre el título, la fecha de publicación, y el número de reacciones</p>

EPIC 04	US 40	Generación de reporte de donaciones	Como creador, quiero generar reportes de las donaciones recibidas para llevar un seguimiento de mis ingresos y el apoyo de mi audiencia.	Escenario 1: Visualización de detalles en el reporte de donaciones Dado que un creador desea analizar las donaciones recibidas, Cuando genera un reporte de donaciones, Entonces el sistema debe mostrar detalles como el monto donado, la fecha de donación.
EPIC 04	US 41	Generar reportes avanzados de publicaciones por estadística	Como creador, quiero generar reportes avanzados que incluyan métricas detalladas como, likes, comentarios para analizar el rendimiento de mis publicaciones y tomar decisiones informadas para mejorar mi contenido.	Escenario 1: Inclusión de métricas avanzadas en el reporte Dado que un creador desea evaluar el rendimiento de sus publicaciones, Cuando genera un reporte avanzado, Entonces el reporte debe incluir métricas como el número de likes, comentarios y el tiempo promedio de visualización para cada publicación.
EPIC 05	US 42	Implementar comentarios en videos	Como usuario, quiero poder dejar comentarios en las publicaciones para interactuar con el creador y otros usuarios, expresando mis opiniones o haciendo preguntas sobre el contenido.	Escenario 1: Opción de comentar visible Dado que un usuario está visualizando una publicación (video), Cuando llega al final de la publicación, Entonces el sistema debe mostrar una opción visible de "Comentar" que permita al usuario escribir un comentario.
EPIC 05	US 43	Implementar reacciones en videos	Como usuario, quiero agregar un "me gusta" en las publicaciones de los creadores para demostrar mi apoyo en su contenido.	Escenario 1: Dar "me gusta" a una publicación Dado que un lector ha accedido a una publicación de un creador, Cuando el lector quiere dar "me gusta" a esa publicación, entonces el sistema debería registrar el "me gusta" y asociarlo al lector y la publicación.

EPIC 05	US 44	Notificaciones en publicaciones	Como usuario suscrito, quiero recibir notificaciones cuando el creador al que estoy suscrito publique un video o un blog, para estar al tanto de sus novedades y contenidos.	<p>Escenario 1: Notificación exitosa al subir contenido</p> <p>Dado que un creador al que estoy suscrito sube un nuevo blog o video, Cuando el creador publique el contenido, Entonces el sistema debe enviar un correo electrónico a todos los usuarios suscritos notificándoles la nueva publicación.</p> <p>Escenario 2: No hay notificación si no estoy suscrito</p> <p>Dado que un creador al que no estoy suscrito sube contenido, cuando el creador publique un nuevo video o blog, entonces no debo recibir ninguna notificación por correo electrónico.</p>
---------	-------	---------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3.3. Product Backlog

NRO	EPIC	US	Título	Story Point	Prioridad
1	EPIC 01	US 01	Registro de Usuarios	1	Alta
2	EPIC 01	US 02	Visualización de Perfil	2	Alta
3	EPIC 01	US 03	Editar perfil	2	Alta
4	EPIC 01	US 04	Inicio de sesión	1	Alta
5	EPIC 01	US 05	Automatizar Rol “Lector” en la creación del usuario	2	Media
6	EPIC 01	US 06	Registro de usuario con Google	5	Media
7	EPIC 02	US 07	Crear categoría	2	Alta
8	EPIC 02	US 08	Visualizar categoría	2	Alta
9	EPIC 02	US 09	Editar Categoría	2	Alta
10	EPIC 02	US 10	Crear Blog	2	Alta
11	EPIC 02	US 11	Editar Blog	3	Alta
12	EPIC 02	US 12	Programación de Blog	5	Alta
13	EPIC 02	US 13	Crear Video	3	Alta

14	EPIC 02	US 14	Registro de Video	2	Alta
15	EPIC 02	US 15	Editar Video	2	Alta
16	EPIC 02	US 16	Programación de video	8	Alta
17	EPIC 02	US 17	Eliminar Video	1	Alta
18	EPIC 02	US 18	Crear playlist	3	Media
19	EPIC 02	US 19	Visualizar playlist	2	Media
20	EPIC 02	US 20	Editar playlist	2	Media
21	EPIC 02	US 21	Eliminar playlist	1	Media
22	EPIC 02	US 22	Añadir video a la lista de reproducción del usuario	3	Media
23	EPIC 02	US 23	Ver la lista de videos en la lista de reproducción del usuario	2	Media
24	EPIC 02	US 24	Eliminar video de la lista de reproducción del usuario	2	Media
25	EPIC 02	US 25	Visualizar el historial de publicaciones	2	Media
26	EPIC 02	US 26	Recomendaciones personalizadas	3	Media
27	EPIC 02	US 27	Registro de publicaciones	2	Media
28	EPIC 02	US 28	Previsualización de publicación	1	Media
29	EPIC 02	US 29	Filtrar categorías por criterios específicos	3	Media
30	EPIC 03	US 30	Suscripción a creadores	8	Alta
31	EPIC 03	US 31	Cancelar suscripción	3	Alta
32	EPIC 03	US 32	Editar planes de suscripciones	2	Alta
33	EPIC 03	US 33	Registro de compras	8	Alta
34	EPIC 03	US 34	Enviar donaciones	3	Alta
35	EPIC 03	US 35	Visualizar lista de donaciones	2	Alta
36	EPIC 03	US 36	Visualizar monto total de las donaciones	2	Alta
37	EPIC 03	US 37	Integrar funcionalidad completa de pagos con PayPal	5	Media
38	EPIC 03	US 38	Retiro de fondos	3	Media
39	EPIC 04	US 39	Generación de reportes básicos de publicaciones por categoría	2	Baja
40	EPIC 04	US 40	Generación de reporte de donaciones	2	Baja
41	EPIC 04	US 41	Generar reportes avanzados de publicaciones por estadística	3	Baja
42	EPIC 05	US 42	Implementar comentarios en videos	2	Media
43	EPIC 05	US 43	Implementar reacciones en videos	2	Media
44	EPIC 05	US 44	Notificaciones en publicaciones	3	Baja

Capítulo IV: Product Design

4.1. Style Guidelines: Imágenes, tipografías, colores y estilos

LOGO



COLORES



TIPOGRAFÍA

Source Code Pro**H1 – ProfitPost, 60px****H2 – ProfitPost, 36px****H3 – ProfitPost, 28px****H4 – ProfitPost, 24px****H5 – ProfitPost, 20px****H6 – ProfitPost, 18px****P – ProfitPost, 14px****P – ProfitPost, 11px****BUTTONS**

Contained Buttons				Outlined Buttons	
Primary		Secondary		Primary	
Default		Disabled		Secondary	

SELECTORS

Dropdowns			
Dropdown ▾	Dropdown ▾	Dropdown ▾	Dropdown ▾
<input checked="" type="radio"/> Disabled he endures pains to avoid worse pains	Unread <input checked="" type="button"/> Following All Activity Source Release	Unread Following <input checked="" type="button"/> All Activity Source Release	Find... Master Dev Stage <input checked="" type="button"/> Dev-api Dev-search
<input type="radio"/> Restricted he endures pains to avoid worse pains	Edit Delete	Edit Delete	

Select Option

Pick a region ▾

Select Country ▾

Domisili ▾

Customized Select

Age ▾

Age ▾

Age ▾

SIZE

Rounded Size
NO Rounded
SM Rounded
MD Rounded
LG Rounded

Size
Small
Medium
Large

Small
Medium
Large

Line Width
1 px
2 px
3 px
4 px
5 px

BOX SHADOW**SIMPLE ALERT****INPUT****Basic input**

Placeholder	Placeholder
Placeholder	Only letters, no special characters
Placeholder	

Search box

Input search text	
Input search text	

<p>Format tooltip input</p> <p>Input group</p> <p>Size of input</p>	<p>Basic form</p> <p>Basic form</p>
------------------------------------------------------------------------------------------	---------------------------------------------------

DATA ENTRY

<p>Checkbox</p>	<p>Radio Button</p>	<p>Customized Switches</p>
<p>Switch With Form Control Label</p>		

Switch Sizes

Sliders

Simple Transfer List

Upload images

The diagram illustrates two distinct file upload scenarios:

Top Scenario: Image Upload

- A large central box contains a grid of five image thumbnails, each showing a burger and fries.
- An "Upload" button is located at the top left of the grid.
- To the right of the grid, a progress bar indicates "Uploading....".
- Below the grid, a thumbnail labeled "images.jpg" is shown with a red border, indicating it is currently being processed.
- An "Upload" button is located at the bottom right of the grid area.

Bottom Scenario: Document Upload

- The interface is divided into two columns of five boxes each.
- Left Column (Image Uploads):** Each box contains a camera icon and the text "Drag & drop your image here, OR" followed by a "Browse" button.
- Right Column (Document Uploads):** Each box contains a cloud icon and the text "Drag & drop your files here, OR" followed by a "Browse Files" button.
- Progress Indicators:** Each box includes a progress bar at the bottom. The first three boxes in the left column show a progress of 70% (red). The last two boxes in the left column show a progress of 100% (green).
- Completed Status:** The bottom-left box in the left column has a "Completed" label with a green checkmark.

4.2. Web Applications Prototyping

Creador/IniciarSesion



Usuario o Correo

Contraseña

Iniciar Sesión

OR

Iniciar Sesión con Google

Registrarse

Creador/Registrar



Usuario

Correo Electronico

Contraseña

Confirma contraseña

Registrar

OR

Iniciar Sesión con Google

Creador/Inicio1



Inicio **Suscripto**

Filtrar 

 **MariaCode**
Lo que es saber poco de programación.
Podes llegar a algo como esto con chatGPT?

```
var coinChange = function(C, A) {
  C = Uint32Array.from(C).sort()
  let ans = Infinity
  const rc = (amt, num, cix) => {
    if (!amt) ans = Math.min(num, ans)
    else if (amt > num) ans = ans
    else if (amt < num) {
      let n = Math.ceil(amt / C[cix])
      if (n + num >= ans) return
      while (~n) C[amt - n * C[cix], num + n--, cix - 1]
    }
  }
  rc(A, 0, C.length-1)
  return ans < Infinity ? ans : -1
};
```

Like  Comment  Share 

 **EduardoRD**
Visual Studio Code
Turn this on today with...
"chat.experimental.detectParticipant.enabled": true

Creador/Inicio2



Inicio **Suscripto**

Filtrar 

 **EduardoSalas**
Contenido sobre programacion

```
public Response post(UserRequest userRequest) {
  try {
    if (userRequest.name == null || userRequest.name.trim().isEmpty()
        || userRequest.surname == null || userRequest.surname.trim().isEmpty()
        || userRequest.email == null || userRequest.email.contains('@')
        || userRequest.password == null || userRequest.password.length() < 8
        || !userRequest.password.equals(userRequest.passwordConfirmation))) {
      throw new RuntimeException("Invalid Request");
    }

    if (this.userRepository.findByEmail(userRequest.email)) != null) {
      throw new RuntimeException("User already exists");
    }
  }
```

Like  Comment  Share 

 **MariaCode**
Turn this on today with...
"chat.experimental.detectParticipant.enabled": true

Creador/Inicio-Filtrar

The screenshot shows a user profile for 'MariaCode'. On the left sidebar, there are navigation links: Inicio, Notificaciones, Publicar, Historial, and Playlist. The main content area displays a post by 'MariaCode' with the title 'Lo que es saber poco de programación. Podes llegar a algo como esto con chatGPT?'. Below the title is a block of JavaScript code:

```
var coinChange = function(C, A) {
  C = Uint32Array.from(C).sort()
  let ans = Infinity
  const rc = (amt, num, cix) => {
    if (amt) ans = Math.min(num, ans)
    else if (amt > 0 && ~cix) {
      let n = Math.ceil(amt / C[cix])
      if (n * num >= ans) return
      while (~n) rc(amt - n * C[cix], num + n--, cix - 1)
    }
  }
  rc(A, 0, C.length-1)
  return ans < Infinity ? ans : -1
};
```

Below the code, there are like, dislike, and share icons. To the right of the post is a sidebar titled 'Mi Contenido' with a 'Filtrar' dropdown menu containing categories: Categoría1, Categoría2, Categoría3, Categoría4, Categoría5, Eliminar, and Filtrar. At the bottom of the main content area is a 'Visual Studio Code' card with the text 'Turn this on today with...' and the JSON configuration: "chat.experimental.detectParticipant.enabled": true.

Creador/MiPerfil1

The screenshot shows a user profile for 'MariaCode'. On the left sidebar, there are navigation links: Inicio, Notificaciones, Publicar, Historial, and Playlist. The main content area displays a profile picture of a woman with blue hair and the name 'MariaCode' above it. Below the profile picture, the text 'Hola soy Maria y este es mi perfil' is displayed. There are two buttons: 'GenerarReporte' and 'Editar Perfil'. Below these buttons are tabs: Videos, Blog, Donaciones, and Comunidad. The 'Blog' tab is active. The main content area contains a post by 'MariaCode' with the title 'Lo que es saber poco de programación. Podes llegar a algo como esto con chatGPT?'. Below the title is the same block of JavaScript code as in the previous screenshot. Like, dislike, and share icons are present at the bottom of the post.

Creador/MiPerfil2

MariaCode
Rol: Creador
Hola soy Maria y este es mi perfil

GenerarReporte
Editar Perfil

Videos Blog Donaciones Comunidad

MariaCode
Turn this on today with...
"chat.experimental.detectParticipant.enabled": true

MariaCode
Turn this on today with...
"chat.experimental.detectParticipant.enabled": true

Creador/MiPerfil3

MariaCode
Rol: Creador
Hola soy Maria y este es mi perfil

GenerarReporte
Editar Perfil

Videos Blog Donaciones Comunidad

Donaciones :	\$23
★ EduardoPR	-\$2
★ EstabanM	-\$5
★ LeoEdu	-\$10
★ JuanAndrade	-\$4
★ GisellaArg	-\$2

Retirar Donaciones

Creador/publicar



← Publicar

- Inicio
- Notificaciones
- + Publicar
- Historial

Video

Blog



MariaCode

Creador/Publicar/video



← Video

- Inicio
- Notificaciones
- + Publicar
- Historial
- Playlist

Titulo

Inserta tu imagen aquí

Previsualizar

Programar
2024-09-16T21:55

publicar



MariaCode

Creador/Publicar/Blog

The screenshot shows a user interface for creating a blog post. On the left, there's a sidebar with icons for Inicio, Notificaciones, Publicar (which is highlighted in blue), Historial, and Playlist. Below the sidebar is a profile picture of a woman with blue hair and the name 'MariaCode'. The main area has a title '← Blog' and fields for 'Titulo' (Title), 'Contenido' (Content), and 'Description'. There's also a placeholder for an image with the text 'Inserta tu imagen aquí' (Insert your image here). A 'Previsualizar' (Preview) button is below the image field. A 'Programar' (Schedule) button with the date '2024-09-16T21:55' is also present. At the bottom is a large green 'publicar' (Publish) button.

Creador/MiPerfil4

The screenshot shows a user profile page for 'MariaCode'. The sidebar on the left is identical to the previous screenshot. The main area features a circular profile picture of MariaCode, her name 'MariaCode', and her role 'Rol: Creador'. Below this is a bio: 'Hola soy Maria y este es mi perfil'. There are two green buttons: 'GenerarReporte' (Generate Report) and 'Editar Perfil' (Edit Profile). Below these buttons are tabs for 'Videos', 'Blog' (which is selected), 'Donaciones', and 'Comunidad'. At the bottom, there are six small cards representing different subscribers or premium users, each with a name like 'Maria', 'nuria', and 'name'.

Creador/Playlist1

The screenshot shows a mobile application interface. On the left, there is a sidebar with icons for Inicio, Notificaciones, Publicar, Historial, and Playlist, with 'Playlist' being the active tab. Below the sidebar is a user profile picture of a woman with blue hair and the name 'MariaCode'. The main content area is titled 'Playlist1' and displays a grid of six items, each with a small thumbnail and text: 'Introducción 11/01/2024', 'Sistemas 24/04/2024', 'Bases de datos 13/06/2024', 'Introducción 01/01/2024', 'Programación 28/01/2024', and 'Probabilidad 07/01/2024'.

Creador/Playlist

The screenshot shows a mobile application interface. On the left, there is a sidebar with icons for Inicio, Notificaciones, Publicar, Historial, and Playlist, with 'Playlist' being the active tab. Below the sidebar is a user profile picture of a woman with blue hair and the name 'MariaCode'. The main content area is titled 'Mis Playlist' and lists five entries, each with a small thumbnail and text: 'Playlist1', 'Playlist1', 'Playlist1', 'Playlist1', and 'Playlist1'. At the bottom right of the main content area is a green button labeled 'Crear Playlist'.

Creador/CrearPlaylist

← Crear Playlist

Título

Crear Playlist

MariaCode

Creador/EditarPerfil

← Editar Perfil

Name
Value

Contraseña
Value

Email
Value

Descripción
Value

Guardar

MariaCode

Creador/Historial/Video

The screenshot shows a user interface for a digital platform. On the left, there is a sidebar with icons for Home, Notifications, Publish, History, and Playlist. The 'History' icon is highlighted with a blue box. The main area has tabs for 'Blog' and 'Videos'. A 'Filtrar' button with a filter icon is at the top right. Below the tabs, there is a search bar and a 'Contenido sobre programacion' section. A video player window displays a video by 'EduardoSalas' titled 'Contenido sobre programacion'. The video content is a snippet of Java code from a GitHub repository. Below the video are like, dislike, and share buttons. On the left side of the main area, there is a profile picture of 'MariaCode'.

Creador/Historial/Blog

The screenshot shows a user interface for a digital platform. On the left, there is a sidebar with icons for Home, Notifications, Publish, History, and Playlist. The 'History' icon is highlighted with a blue box. The main area has tabs for 'Blog' and 'Videos'. A 'Filtrar' button with a filter icon is at the top right. Below the tabs, there is a search bar and a blog post by 'EduardoSalas' titled 'Estoy editando el nuevo video que publicare pronto ¡No te lo pierdas!'. Below the post, there is a snippet of Visual Studio Code code. On the left side of the main area, there is a profile picture of 'MariaCode'.

Lector/IniciarSesion



Usuario o Correo

Contraseña

Iniciar Sesión

OR

Iniciar Sesión con Google

[Registrarse](#)

Lector/IniciarSesion



Usuario

Correo Electronico

Contraseña

Confirma contraseña

Registrar

OR

Iniciar Sesión con Google

Lector/Inicio

Inicio Suscripto

MariaCode
Lo que es saber poco de programación. Podes llegar a algo como esto con chatGPT?

```
var coinChange = function(C, A) {  
    C = Uint32Array.from(C).sort()  
    let ans = Infinity  
    const rc = (amt, num, cix) => {  
        if (!amt) ans = Math.min(num, ans)  
        else if (amt > 0 && num >= cix) {  
            let n = Math.ceil(amt / C[cix])  
            if (n + num >= ans) return  
            while (~n) rc(amt - n * C[cix], num + n--, cix - 1)  
        }  
    }  
    rc(A, 0, C.length-1)  
    return ans < Infinity ? ans : -1  
};
```

EduardoRD

Lector/Suscripto

Inicio Suscripto

MariaCode
Lo que es saber poco de programación. Podes llegar a algo como esto con chatGPT?

```
var coinChange = function(C, A) {  
    C = Uint32Array.from(C).sort()  
    let ans = Infinity  
    const rc = (amt, num, cix) => {  
        if (!amt) ans = Math.min(num, ans)  
        else if (amt > 0 && num >= cix) {  
            let n = Math.ceil(amt / C[cix])  
            if (n + num >= ans) return  
            while (~n) rc(amt - n * C[cix], num + n--, cix - 1)  
        }  
    }  
    rc(A, 0, C.length-1)  
    return ans < Infinity ? ans : -1  
};
```

EduardoPR

Lector/Filtrar

The screenshot shows a mobile application interface. At the top, there are two tabs: "Inicio" (Home) and "Suscripto" (Subscriptions). On the left, a sidebar menu includes "Inicio" (selected), "Notificaciones", "Playlist", and "Historial". The main content area displays a news feed. The first item is from "MariaCode" with the text: "Lo que es saber poco de programación. Podes llegar a algo como esto con chatGPT?". Below the text is a code snippet:

```
var coinChange = function(C, A) {
    C = Uint32Array.from(C).sort();
    let ans = Infinity;
    const rc = (amt, num, cix) => {
        if (!amt) ans = Math.min(num, ans);
        else if (amt > 0 && ~cix) {
            let n = Math.ceil(amt / C[cix]);
            if (n + num >= ans) return;
            while (~n) {
                if (amt - n * C[cix] <= num) {
                    ans = n;
                    num -= n * C[cix];
                    cix--;
                }
            }
        }
        rc(A, 0, C.length-1);
    };
    return ans < Infinity ? ans : -1;
};
```

Below this is another post from "MariaCode" with the text: "Turn this on today with...". The code snippet is partially visible: "chat.experimental.detectParticipant.enabled": true. On the right side of the screen, there is a filter menu with categories: "Categoría1" (selected), "Categoría2", "Categoría3", "Categoría4", "Categoría5", "Eliminar", and "Filtrar".

Lector/Playlist

The screenshot shows a mobile application interface. At the top, there is a back arrow and the text "Playlist". On the left, a sidebar menu includes "Inicio" (selected), "Notificaciones", "Playlist" (selected), and "Historial". The main content area displays a section titled "Mis Playlist" (My Playlists) with three items, each labeled "Playlist1". At the bottom right, there is a green button labeled "CrearPlaylist" (Create Playlist).

Lector/CrearPlaylist

The screenshot shows the application's main navigation bar at the top. On the left is a logo consisting of three blue arrows pointing right. To its right are four menu items: "Inicio" (Home), "Notificaciones" (Notifications), "Playlist" (highlighted with a blue background), and "Historial" (History). Below the menu is a user profile picture of a man and the name "EduardoPR". The main content area has a light green header with the text "← CrearPlaylist". It contains a form with a title input field labeled "Titulo" containing "Nuevo Playlist" and a green "CrearPlaylist" button below it.

Lector/Playlist/Playlist1

The screenshot shows the application's main navigation bar at the top. On the left is a logo consisting of three blue arrows pointing right. To its right are four menu items: "Inicio" (Home), "Notificaciones" (Notifications), "Playlist" (highlighted with a blue background), and "Historial". Below the menu is a user profile picture of a man and the name "EduardoPR". The main content area has a light green header with the text "← Playlist1". It displays a grid of six cards representing different tracks or items in the playlist:

Playlist1		
Integrante Nuevo Playlist 11/12/24	Santa Musicalistica 01/04/24	Resto Musicalistica 13/10/24
derriadas Nuevo Playlist 05/03/24	Producto Musicalistica 28/01/24	Probabilidad Nuevo Playlist 07/12/24

Lector/MiPerfil

The screenshot shows a mobile application interface. On the left, there is a sidebar with icons for Home, Notifications, Playlist, and History. Below the sidebar is a user profile picture and the name "EduardoRD". The main content area displays a circular profile picture of a man wearing sunglasses and a hoodie. Above the picture, the name "EduardoRD" and the role "Rol: Lector" are shown. A message "Hola soy Eduardo y este es mi perfil" is displayed. A green "Editar Perfil" button is located in the bottom right corner of the main content area.

Lector/EditarPerfil

The screenshot shows the profile editing screen. The left sidebar remains the same. The main content area has a header "Editar Perfil" with a back arrow. It features a circular profile picture of the same man. To the right of the picture are four input fields: "Name" (with placeholder "Value"), "Contraseña" (with placeholder "Value"), "Email" (with placeholder "Value"), and "Descripcion" (with placeholder "Value"). A large green "Guardar" button is at the bottom right of the form area.

Lector/VerPerfilDeCreadores1

The screenshot shows a user profile for 'MariaCode' (Rol: Creador). The profile picture is a woman with blue hair. The bio reads: 'Hola soy maria y este es mi perfil'. Below the bio is a green 'Donar' button. At the bottom, there are tabs for 'Videos', 'Blog', and 'Suscribirse'. A post from 'EduardoRD' is displayed, showing a snippet of code for a coin change algorithm:

```
var coinChange = function(C, A) {
  C = Uint32Array.from(C).sort();
  let ans = Infinity;
  const rc = (amt, num, cix) => {
    if (amt) ans = Math.min(num, ans);
    else if (amt > 0 && ~cix) {
      let n = Math.floor(amt / C[cix]);
      if (n + num >= ans) return;
      while (~n) (camt - n * C[cix], num + n--, cix - 1);
    }
  };
  rc(A, 0, C.length-1);
  return ans < Infinity ? ans : -1;
};
```

Below the post are like, dislike, and share icons.

Lector/VerPerfilDeCreadores2

The screenshot shows a user profile for 'MariaCode' (Rol: Creador). The profile picture is a woman with blue hair. The bio reads: 'Hola soy maria y este es mi perfil'. Below the bio is a green 'Donar' button. At the bottom, there are tabs for 'Videos', 'Blog', and 'Suscribirse'. Two subscription options are shown:

Suscriptor	SuscriptorPremium
\$5 / month	\$10 / month
• Contenido Exclusivo • Saludo Personalizado	• Contenido Exclusivo • Saludos Personalizados • Grupo de Whatsapp • Reuniones con subs • Sorteos mensuales
Seleccionar	Seleccionar

Below the post are like, dislike, and share icons.

Lector/Historial/Videos

The screenshot shows a user interface for a digital platform. On the left, there is a sidebar with the following navigation links:

- Inicio
- Notificaciones
- Publicar
- Historial** (highlighted in blue)
- Playlist

The main content area has tabs for "Blog" and "Videos". A post by "EduardoSalas" is displayed, titled "Contenido sobre programacion". The post contains a code snippet in Java:

```
public Response pedir(LoginRequest userRequest) {  
    try {  
        if (userRequest.name() == null || userRequest.name().trim().isEmpty()  
            || userRequest.surname() == null || userRequest.surname().trim().isEmpty()  
            || userRequest.email() == null || userRequest.email().contains("@")  
            || userRequest.password() == null || userRequest.password().length() < 8  
            || userRequest.passwordConfirmation() != userRequest.password().passwordConfirmation()) {  
            throw new RuntimeException("Invalid request");  
        }  
  
        if (this.userRepository.findByEmail(userRequest.email()) != null) {  
            throw new RuntimeException("User already exists");  
        }  
    } catch (Exception e) {  
        throw new RuntimeException("Error while saving user");  
    }  
}
```

Below the post, there are three small icons: a thumbs up, a reply, and a share.

Lector/Historial/Blog

The screenshot shows a user interface for a digital platform. On the left, there is a sidebar with the following navigation links:

- Inicio
- Notificaciones
- Publicar
- Historial** (highlighted in blue)
- Playlist

The main content area has tabs for "Blog" and "Videos". A post by "EduardoPR" is displayed, titled "Estoy editando el nuevo video que publicare pronto ¡No te lo pierdas!". Below the post, there is a snippet of code for "Visual Studio Code":

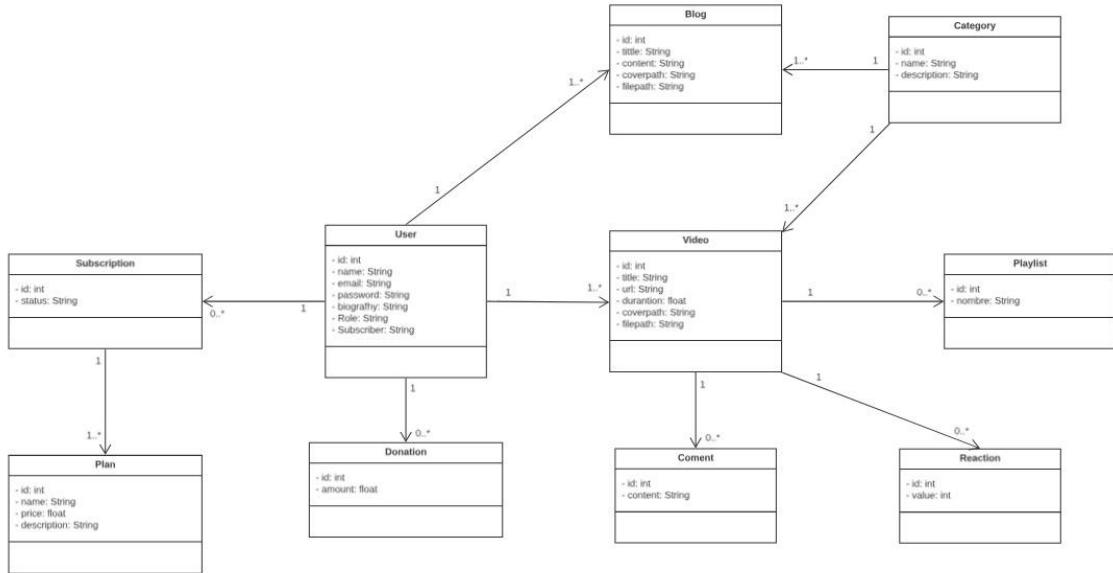
```
chat.experimental.detectParticipant.enabled": true
```

At the bottom of the post, there is a link to "GitHub":

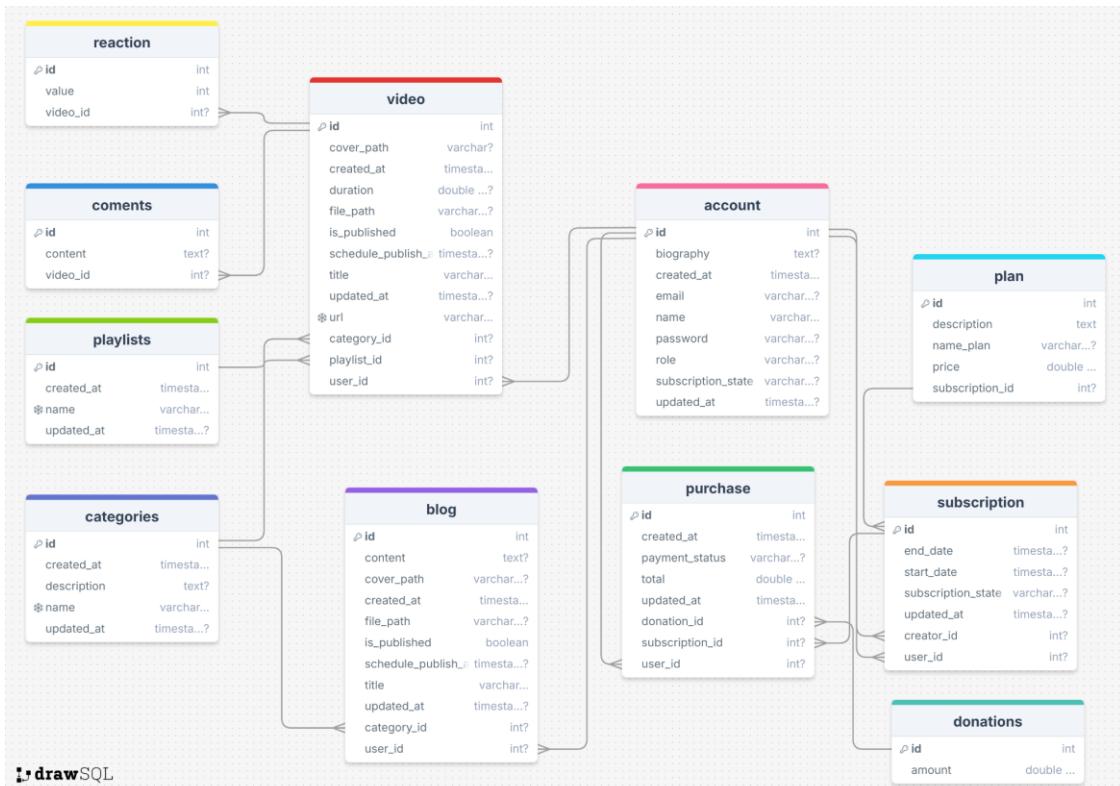
GitHub estrena una herramienta que corrige vulnerabilidades de seguridad mediante IA

4.3. Software Object-Oriented Design

4.3.1 Class Diagram



4.3.2. Database Design



Capítulo V: Product Implementation, Validation & Deployment

5.1. Software Configuration Management

5.1.1 Software Development Environment

- **Lenguaje de programación:** Java (versión 17)
- **Framework principal:** Spring Boot 3.x
- **Propósito:** Backend API
- **Base de Datos:** PostgreSQL 14
- **Conexión:** JDBC con spring.datasource
- **IDE:** IntelliJ IDEA
- **Plugins:** Spring Boot, Lombok, Maven
- **Repositorio de código (Control de versiones):** GitHub
- **URL:** <https://github.com/ProfPost/profpost-store-api.git>
- **Sistema de Dependencias:** Maven
 - Usado para gestionar librerías como spring-boot-starter-web, spring-boot-starter-data-jpa, lombok.
- **Entorno de Ejecución:** Localhost con Spring Boot embebido y gestor de base de datos PostgreSQL.
- **Entorno de Pruebas:** Postman

5.1.2 Software Code Management

- **Sistema de control de versiones:** Git, hospedado en GitHub.
- **Ramas:**
 - **main:** rama principal aún no implementada.
 - **develop:** rama de integración continua para features finalizadas hasta el Sprint 1.
 - **feature/playlistvideo:** rama de integración entre playlist y videos, permitiendo agregar, modificar y eliminar videos de la playlist.
 - **feature/subscription:** rama de implementación de suscripciones permitiendo que un usuario rector pueda suscribirse a un plan.
 - **feature/crud-plan:** rama de implementación de planes, en esta rama se añaden los planes permitiendo modificar precios y tipos de planes.
 - **feature/crud-video:** rama de implementación de videos, esta rama permite al usuario creador subir videos.
 - **feature/crud-user:** rama de implementación de usuarios, en esta se le permite al usuario registrarse y luego iniciar sesión, además de poder editar su perfil.
 - **feature/crud-blog:** rama de implementación de blogs, en esta rama se le permite al creador publicar blogs.

- **feature/crud-category:** rama de implementación de categorías, en esta rama se permite crear, ver, actualizar y eliminar categorías para las publicaciones.
- **feature/crud-playlist:** rama de implementación de playlist, en esta rama se puede crear, actualizar y eliminar playlist.

Flujo de trabajo (Workflow):

Creación de ramas feature para cada funcionalidad

Branch	Updated	Check status	Behind	Ahead	Pull request
feature/playlistvideo	2 hours ago		6	0	#14
feature/reaction	4 hours ago		3	1	...
feature/purchase	5 hours ago		3	4	...
feature/subscription	8 hours ago		21	5	...
feature/crud-plan	9 hours ago		21	3	#13
feature/notifications	13 hours ago		3	3	...
feature/crud-video	yesterday		8	0	#11
feature/donation	2 days ago		26	1	...
feature/crud-user	2 days ago		30	0	#10
feature/crud-blog	2 days ago		22	0	#9
feature/crud-category	3 days ago		27	0	#6
feature/crud-playlist	4 days ago		32	0	#2
main	4 days ago		33	0	#5

Pull requests: para revisión de código antes de fusionar en develop

12 Pull requests merged by 3 people	
feat(backend-playlistvideo) implementación de agregar, listar, eliminar...	#14 merged 2 hours ago
Feature/crud plan	#13 merged 9 hours ago
Feature/subscription	#12 merged yesterday
Feature/crud video	#11 merged yesterday
Feature/crud user	#10 merged yesterday
Feature/crud blog	#9 merged yesterday
feat(backend) implementacion de suscripciones a creadores	#8 merged yesterday
Implementación de la historia de usuario para gestión de publicaciones en el blog	#7 merged 3 days ago
feat(backend) Implementacion del CRUD de category	#6 merged 3 days ago
Edit README.md	#5 merged 3 days ago
feat(backend) implementacion del CRUD de playlist	#2 merged 4 days ago
feat(backend) implementacion del CRUD de user	#1 merged 4 days ago

Revisión y aprobación:

feature/crud-user

feat(backend) implementacion del CRUD de user #1

Merged SergioVelasquezR merged 1 commit into develop from feature/crud-user 4 days ago

Conversation 0 Commits 1 Checks 0 Files changed 4

Berrospireyes commented 4 days ago

Este pull request añade la implementación completa de las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para la entidad User.

Cambios Realizados

- Se añadió el repositorio UserRepository con métodos básicos de CRUD.
- Se creó el servicio UserService y UserServiceImpl con la lógica de negocio para gestionar usuarios.
- Se implementó el controlador UserController con endpoints Rest para las operaciones CRUD.
- Se realizaron las pruebas para verificar la funcionalidad del CRUD mediante Postman

feat(backend) implementacion del CRUD de user

SergioVelasquez merged commit 469cc87 into develop 4 days ago

Revert

feat(backend) implementacion del CRUD de user #1

Merged SergioVelasquezR merged 1 commit into develop from feature/crud-user 4 days ago

Conversation 0 Commits 1 Checks 0 Files changed 4

Changes from all commits File filter Conversations 4 / 4 files viewed Review in codespace Review changes

src/main/java/com/propost

- src/main/java/com/propost/api UserController.java
- src/main/java/com/propost/repository UserRepository.java
- src/main/java/com/propost/service UserService.java
- src/main/java/com/propost/service/impl UserServiceImpl.java

src/main/java/com/propost/repository

src/main/java/com/propost/service

src/main/java/com/propost/service/impl

feature/crud-playlist

feat(backend) implementacion del CRUD de playlist #2

Merged SergioVelasquezR merged 1 commit into develop from feature/crud-playlist 4 days ago

Conversation 0 Commits 1 Checks 0 Files changed 5

Berrospireyes commented 4 days ago

Este pull request añade la implementación completa de las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para la entidad Playlist.

Cambios Realizados

- Se añadió el repositorio PlaylistRepository con métodos básicos de CRUD.
- Se creó el servicio PlaylistService y PlaylistServiceImpl con la lógica de negocio para gestionar categorías.
- Se implementó el controlador PlaylistController con endpoints REST para las operaciones CRUD.
- Se realizaron las pruebas para verificar la funcionalidad del CRUD mediante Postman.

feat(backend) implementacion del CRUD de playlist

SergioVelasquez merged commit cb3eacb into develop 4 days ago

Revert

Pull request successfully merged and closed

You're all set—the feature/crud-playlist branch can be safely deleted.

Delete branch

feat(backend) implementacion del CRUD de playlist #2

Merged SergioVelasquezR merged 1 commit into develop from feature/crud-playlist 4 days ago

Conversation 0 Commits 1 Checks 0 Files changed 5

Changes from all commits File filter Conversations 5 / 5 files viewed Review in codespace Review changes

src/main/java/com/propost

- src/main/java/com/propost/api PlaylistController.java
- src/main/java/com/propost/repository CategoryRepository.java
- src/main/java/com/propost/repository PlaylistRepository.java
- src/main/java/com/propost/service PlaylistService.java
- src/main/java/com/propost/service/impl PlaylistServiceImpl.java

Edit Readme.md

Edit README.md #5

Merged SergioVelasquezR merged 1 commit into feature/crud-category from main 3 days ago

Conversation 0 Commits 1 Checks 0 Files changed 1

oscarLaureC commented 3 days ago
README: agregar detalles del tablero de Jira y colaboradores, diagrama de clases, diagrama de base de datos

o Edit README.md ... Verified 982c928

SergioVelasquezR merged commit 464a96e into feature/crud-category 3 days ago Revert

Pull request successfully merged and closed You're all set—the main branch can be safely deleted. Delete branch

Edit README.md #5

Merged SergioVelasquezR merged 1 commit into feature/crud-category from main 3 days ago

Conversation 0 Commits 1 Checks 0 Files changed 1 +92 -0

Changes from all commits File filter Conversations Jump to 1 / 1 files viewed Review changes

92 README .md Viewed ...

feature/crud-category

feat(backend) Implementacion del CRUD de category #6

Merged SergioVelasquezR merged 1 commit into develop from feature/crud-category 3 days ago

Conversation 1 Commits 1 Checks 0 Files changed 9

oscarLaureC commented 3 days ago
Este pull request añade la implementación completa de las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para la entidad Category.

Cambios Realizados

Se añadió el repositorio CategoryRepository con métodos básicos de CRUD.
Se creó el servicio CategoryService y CategoryServiceImpl con la lógica de negocio para gestionar categorías.
Se implementó el controlador CategoryController con endpoints REST para las operaciones CRUD.
Se realizaron las pruebas para verificar la funcionalidad del CRUD mediante Postman.

SergioVelasquezR commented 3 days ago
Revisado.

SergioVelasquezR merged commit a2d6a82 into develop 3 days ago Revert

feat(backend) Implementacion del CRUD de category #6

Merged SergioVelasquezR merged 1 commit into develop from feature/crud-category 3 days ago

Conversation 1 Commits 1 Checks 0 Files changed 9 +126 -25

Changes from all commits File filter Conversations 9 / 9 files viewed Review in codebase Review changes

Filter changed files

- src/main/java/com/Propost/api
- src/main/java/com/Propost/api/CategoryController.java
- src/main/java/com/Propost/api/CreatorCategoryController.java
- src/main/java/com/Propost/repository
- src/main/java/com/Propost/repository/CategoryRepository.java
- src/main/java/com/Propost/service
- src/main/java/com/Propost/service/CategoryService.java
- src/main/java/com/Propost/service/CreatorCategoryService.java
- src/main/java/com/Propost/service/ServiceCategory.java
- src/main/java/com/Propost/service/impl
- src/main/java/com/Propost/service/impl/CategoryServiceImpl.java
- src/main/java/com/Propost/service/impl/CreatorCategoryServiceImpl.java
- src/main/java/com/Propost/service/impl/ServiceCategoryImpl.java

Implementación de historia de usuario Gestión de publicaciones en el blog

Implementación de suscripción a creadores

feat(backend) implementacion de suscripciones a creadores #8

Merged SergioVelasquezR merged 2 commits into develop from feature/subscription yesterday

Conversation 0 Commits 2 Checks 0 Files changed 9

SergioVelasquez commented yesterday Member ...

Este pull request añade la implementación completa de las suscripciones donde los usuarios lectores podrán suscribirse a usuarios creadores.

Cambios Realizados

Se añadió el repositorio SubscriptionRepository con métodos básicos de CRUD.
Se creó el servicio SubscriptionService y SubscriptionServiceImpl con la lógica para suscribirse y dejar de suscribirse.
Se implementó el controlador SuscriptionController con endpoints REST para las operaciones Post y Delete.
Se implementó un DTO SubscriptionDTO y SubscriptionResponseDTO donde se unen los id's del usuario y del creador y devuelve la suscripción.
Se realizaron las pruebas para verificar la funcionalidad de las suscripciones mediante Postman.

Sergio and others added 2 commits 3 days ago

feat(backend) implementacion de suscripciones a creadores a47329b
Merge branch 'develop' into feature/subscription eb7e4c8

Verified

SergioVelasquezR merged commit 392cb52 into develop yesterday Revert

feat(backend) implementacion de suscripciones a creadores #8

Merged SergioVelasquezR merged 2 commits into develop from feature/subscription yesterday

Conversation 0 Commits 2 Checks 0 Files changed 9

Changes from all commits File filter Conversations 9 / 9 files viewed Review in codespace Review changes

Filter changed files

- src/main/java/com/ProPost/api/SubscriptionController.java
- src/main/java/com/ProPost/dto/SubscriptionDTO.java
- src/main/java/com/ProPost/dto/SubscriptionResponseDTO.java
- src/main/java/com/ProPost/model/entity/Plan.java
- src/main/java/com/ProPost/model/entity/Purchase.java
- src/main/java/com/ProPost/model/entity/Subscription.java
- src/main/java/com/ProPost/repository/SubscriptionRepository.java
- src/main/java/com/ProPost/service/SubscriptionService.java
- src/main/java/com/ProPost/service/impl/SubscriptionServiceImpl.java

feature/crud-blog**Feature/crud blog #9**

Merged SergioVelasquezR merged 4 commits into develop from feature/crud-blog yesterday

Conversation 0 Commits 4 Checks 0 Files changed 8

SergioVelasquezR commented yesterday Member ...

Este pull request añade la implementación completa de las suscripciones donde los usuarios lectores podrán suscribirse a usuarios creadores.

Cambios Realizados

Se añadió el repositorio `BlogRepository` con métodos básicos de CRUD.
 Se creó el servicio `UserBlogService` y `UserBlogServiceImpl` con la lógica básica de un CRUD para blogs.
 Se implementó el `BlogScheduler` con la lógica para programar la publicación del blog.
 Se implementó el controlador `BlogController` con endpoints REST para las operaciones CRUD.
 Se realizaron las pruebas para verificar la funcionalidad de las suscripciones mediante Postman.

Sergio and others added 4 commits 3 days ago

- feat(backend): implementacion del CRUD de Blog 736f224
- Merge pull request #7 from ProPost/develop ... Verified 27e7ca2
- feat(backend): finalizacion de la implementacion del CRUD de blogs 03b0f3b
- feat(backend): agregar funcionalidad de publicación programada y actu... 71e6b76

SergioVelasquezR merged commit b7b01cb into develop yesterday Revert

Feature/crud blog #9

Merged SergioVelasquezR merged 4 commits into develop from feature/crud-blog yesterday

Conversation 0 Commits 4 Checks 0 Files changed 8

Changes from all commits File filter Conversations 8 / 8 files viewed Review in codespace Review changes

Filter changed files

- src/main/java/com/ProPost/api/UserBlogController.java
- src/main/java/com/ProPost/config/SchedulingConfig.java
- src/main/java/com/ProPost/dto/BlogDTO.java
- src/main/java/com/ProPost/model/entity/Blog.java
- src/main/java/com/ProPost/repository/BlogRepository.java
- src/main/java/com/ProPost/scheduler/BlogScheduler.java
- src/main/java/com/ProPost/service/UserBlogService.java
- src/main/java/com/ProPost/service/impl/UserBlogServiceImpl.java

modificación del feature/crud-user

Feature/crud user #10

Merged SergioVelasquezR merged 2 commits into `develop` from `feature/crud-user` yesterday

Conversation 0 · Commits 2 · Checks 0 · Files changed 5

SergioVelasquezR commented yesterday

Este pull request añade la implementación completa del manejo de los usuarios, permitiéndoles crear cuentas, un login, editar su cuenta y eliminar su cuenta.

Cambios Realizados

Se añadió el repositorio `UserRepository` con métodos básicos de CRUD.
 Se creó el servicio `UserService` y `UserServiceimpl` con la lógica básica de un CRUD para usuarios.
 Se implementó el controlador `UserController` con endpoints REST para las operaciones de listado, actualizar y eliminar.
 Se implementó un authController que permite registrar usuarios y loguearse si sus credenciales son correctas.
 Se realizaron las pruebas para verificar la funcionalidad mediante Postman.

Sergio added 2 commits 2 days ago

- o- feat(backend) se agrego correctamente el registro del usuario 344e78a
- o- feat(backend) se agrego correctamente el inicio de sesion del usuario... 551e2b6

SergioVelasquezR merged commit `bdfcef` into `develop` yesterday [Revert](#)

Pull request successfully merged and closed You're all set—the `feature/crud-user` branch can be safely deleted. [Delete branch](#)

Feature/crud user #10

Merged SergioVelasquezR merged 2 commits into `develop` from `feature/crud-user` yesterday

Conversation 0 · Commits 2 · Checks 0 · Files changed 5

Changes from all commits · File filter · Conversations · S / 5 files viewed · Review in codespace · Review changes

Filter changed files

- > 37 src/main/java/com/Proffpost/api/AuthController.java
- > 8 src/main/java/com/Proffpost/api/UserController.java
- > 4 src/main/java/com/Proffpost/repository/UserRepository.java
- > 3 src/main/java/com/Proffpost/service/UserService.java
- > 23 src/main/java/com/Proffpost/service/impl/UserServiceimpl.java

[Viewed](#) [Unviewed](#) [...](#)

feature/subscription

Feature/subscription #12

Merged SergioVelasquezR merged 3 commits into `develop` from `feature/subscription` yesterday

Conversation 0 · Commits 3 · Checks 0 · Files changed 6

SergioVelasquezR commented yesterday

Este pull request soluciona el manejo de suscripciones entre usuarios y creadores

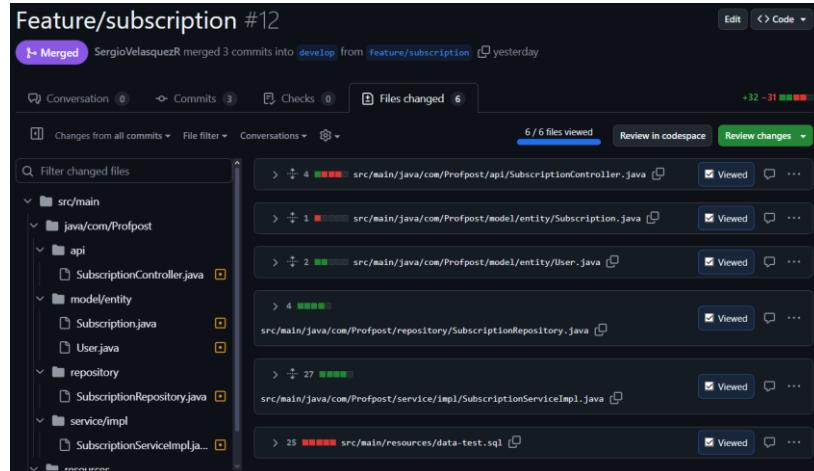
Cambios Realizados

Se añadió una verificación para evitar que un usuario se suscriba a un creador si ya existe una suscripción activa.
 La suscripción se puede cancelar, pero el período pagado (30 días) se respeta antes de finalizar.
 Se agregó la lógica para permitir que el usuario se suscriba nuevamente después de que la suscripción anterior haya finalizado.
 Se realizaron pruebas para validar el comportamiento con diferentes escenarios de suscripción usando Postman.

Sergio added 3 commits yesterday

- o- fix(backend) se corrigió el nombre de las tablas subscription -> subsc... 0b211db
- o- Merge remote-tracking branch 'origin/feature/subscription' into featu... 8ced436
- o- fix(backend-suscription) evitar múltiples suscripciones activas y per... 5787a8b

SergioVelasquezR merged commit `f69f3bb` into `develop` yesterday [Revert](#)



feature/crud-video

Feature/crud video #11

Merged SergioVelasquezR merged 7 commits into develop from feature/crud-video yesterday

Conversation 0 Commits 7 Checks 0 Files changed 13

SergioVelasquezR commented yesterday Member ...

Este pull request añade la implementación completa de la publicación de videos donde los usuarios creadores podrán subir videos a la plataforma.

Cambios Realizados

Se añadió el repositorio VideoRepository con métodos básicos de CRUD.
Se creó el servicio VideoService y VideoServiceImpl con la lógica básica de un CRUD para videos.
Se implementó el VideoScheduler con la lógica para programar la publicación del video.
Se implementó el controlador VideoController con endpoints REST para las operaciones CRUD.
Se realizaron las pruebas para verificar la funcionalidad mediante Postman.

RamosVania and others added 7 commits 4 days ago

- feat(backend): implementación del CRUD de video b4672c2
- feat(backend): implementación del CRUD de video 6817482
- Se hizo merge a Feature/crud-user con feature/crud-video para integrarlos f19dcc8
- Se hizo merge a feature/crud-category into feature/crud-video para integrarlos fe7674b
- feat(backend): implementación y corrección del crud de video 50eebf4
- feat(backend): implementación de programación de video y actualización 0ecedad
- Merge branch 'develop' into feature/crud-video 50eed83

SergioVelasquezR merged commit 7aeb3c9 into develop yesterday Verified Revert

Pull request successfully merged and closed Delete branch

Feature/crud video #11

Merged SergioVelasquezR merged 7 commits into develop from feature/crud-video yesterday

Conversation 0 Commits 7 Checks 0 Files changed 13

Changes from all commits File filter Conversations 13 / 13 files viewed Review in codespace Review changes

Filter changed files src/main java/com/Propost PropostApplication.java src/main/java/com/Propost/api/CategoryController.java src/main/java/com/Propost/api/VideoController.java src/main/java/com/Propost/config/SchedulingConfig.java src/main/java/com/Propost/dto/VideoDTO.java src/main/java/com/Propost/model/entity/Video.java src/main/java/com/Propost/repository/VideoRepository.java src/main/java/com/Propost/scheduler/VideoScheduler.java src/main/java/com/Propost/service/CategoryService.java src/main/java/com/Propost/service/VideoService.java

feature/crud-plan

Feature/crud plan #13

Merged SergioVelasquezR merged 3 commits into `feature/subscription` from `feature/crud-plan` 9 hours ago

Conversation 0 · Commits 3 · Checks 0 · Files changed 6

SergioVelasquezR commented 9 hours ago

En este pull request se hace porque se quiere integrar los planes con las suscripciones para reutilizar código.

RamosVania and others added 3 commits 3 days ago

- feat(backend): implementación del CRUD de plan 19bbb1f
- feat(backend): corrección en la relación de entidades y la implementación b752d03
- Merge remote-tracking branch 'origin/feature/subscription' into featu... 4964b28

SergioVelasquezR merged commit `4c63c29` into `feature/subscription` 9 hours ago · Revert

Pull request successfully merged and closed

You're all set—the `feature/crud-plan` branch can be safely deleted. · Delete branch

Feature/crud plan #13

Merged SergioVelasquezR merged 3 commits into `feature/subscription` from `feature/crud-plan` 9 hours ago

Conversation 0 · Commits 3 · Checks 0 · Files changed 6

Changes from all commits · File filter · Conversations · 6 / 6 files viewed · Review in codespace · Review changes

Filter changed files

- src/main
 - java/com/Propost
 - api
 - PlanController.java
 - model/entity
 - Plan.java
 - repository
 - PlanRepository.java
 - service
 - PlanService.java
 - impl
 - PlanServiceImpl.java
 - resources

Viewed · Viewed · Viewed · Viewed · Viewed · Viewed

implementación de playlist video

feat(backend-playlistvideo) implementación de agregar, listar, eliminar

Merged SergioVelasquezR merged 1 commit into `develop` from `feature/playlistvideo` 2 hours ago

Conversation 0 · Commits 1 · Checks 0 · Files changed 4

BerrospiReyes commented 2 hours ago

Este pull request añade la implementación completa de las operaciones para la entidad `Playlist`.

Cambios Realizados

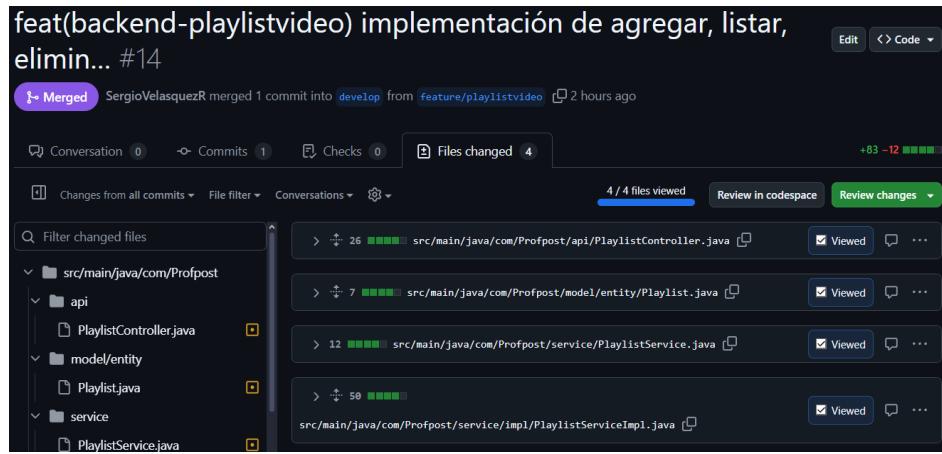
Se añadió el método `agregar un video a una playlist`.
 Se añadió el método `listar videos de una playlist`.
 Se añadió el método `eliminar un video de una playlist`.
 Se implementó en el controlador `PlaylistController` con endpoints REST para las operaciones.
 Se realizaron las pruebas para verificar la funcionalidad del CRUD mediante Postman.

feat(backend-playlistvideo) implementación de agregar, listar, eliminar ... 9e9cf1c

SergioVelasquezR merged commit `06f5820` into `develop` 2 hours ago · Revert

Pull request successfully merged and closed

You're all set—the `feature/playlistvi...` branch can be safely deleted. · Delete branch



5.1.3. Source Code Style Guide & Conventions

De acuerdo con lo que hemos programado, se ha seguido una serie de convenciones de codificación para asegurar la legibilidad, mantenibilidad y consistencia del código fuente en todo el proyecto. A continuación, se detallan las principales convenciones utilizadas:

- Nombres de Clases:

Los nombres de clases se definen utilizando PascalCase.

Deben ser sustantivos o frases que describan claramente su propósito, ejemplo: UserService, BlogRepository, VideoController.

- Nombres de Métodos:

Los métodos se nombran utilizando camelCase.

Los nombres deben ser verbos que describan claramente lo que hacen, por ejemplo: findById(), createBlog(), deleteUser().

Los métodos suelen ser cortos, con una única responsabilidad, lo que facilita su comprensión y mantenimiento.

- Nombres de Variables:

Las variables también siguen el formato camelCase.

Los nombres de las variables son descriptivos y claros, evitando abreviaturas a menos que sean comunes y ampliamente entendidas. Ejemplo: userRepository, blogService, subscriptionId.

- Nombres de Paquetes:

Los nombres de los paquetes están en minúsculas y siguen un formato estructurado de acuerdo con las capas de la aplicación. Ejemplo: com.propost.api, com.propost.model.entity.

- Uso de DTOs:

Se han implementado Data Transfer Objects (DTOs) para desacoplar la lógica del negocio de la API y mantener un diseño limpio, evitando exponer directamente las entidades. Ejemplo: BlogDTO, SubscriptionDTO.

- Formato de Código:

Se utiliza sangría de 4 espacios para mejorar la legibilidad del código.

Se añade una línea en blanco entre los bloques de código, como las definiciones de métodos y secciones lógicas dentro de ellos, para mejorar la organización visual.

- Convenciones de Acceso a Datos:

Para las entidades de datos, se sigue la convención de definir relaciones con anotaciones como @ManyToOne y @OneToMany, con llaves foráneas bien definidas en las clases de entidad. Ejemplo: la relación entre Blog y User, o Subscription y User.

- Anotaciones Estándar

Se utilizan anotaciones de Spring para manejar las dependencias (@Autowired, @RequiredArgsConstructor) y la definición de endpoints (@RestController, @GetMapping, @PostMapping), lo que facilita la comprensión rápida de cómo está estructurada la API y cómo se conectan los servicios.

- Manejo de Excepciones:

Se siguen buenas prácticas de manejo de errores y excepciones con try-catch y lanzando excepciones personalizadas para casos de validación y errores de negocio, por ejemplo, RuntimeException en los casos de entidades no encontradas.

- Documentación:

El código incluye anotaciones `@Transactional` y `@Override` cuando es necesario para indicar comportamiento específico de transacciones y sobrescritura de métodos.

5.1.4 Software Deployment Configuration

- **Lenguaje de Programación:** Java 17
- **BlogDTO:** Transforma los datos del blog (como título, contenido, categoría, y usuario) en un objeto manejable para crear o actualizar blogs.
 - **SubscriptionDTO:** Transfiere la información entre un usuario y un creador al suscribirse.
 - **SubscriptionResponseDTO:** Proporciona una respuesta estructurada después de realizar acciones relacionadas con suscripciones, incluyendo detalles del estado y mensajes.
 - **VideoDTO:** Facilita la transferencia de información entre el controlador y los servicios al crear o actualizar videos.
- **Convecciones de Nombres:**
 - **Clases:** Las clases deben tener nombres en PascalCase, ejemplos: `UserServiceImpl`, `CategoryService`, `BlogController`
 - **Métodos y variables:** Los métodos deben tener nombres en camelCase, usos: `findUserById()`, `createBlog()`, `updateCategory()`, las variables deben seguir el formato camelCase, usos: `userId`, `blogTitle`, `isPublished`
- **Buenas prácticas:**
 - **Uso de DTO (Data Transfer Objects):** Para separar la lógica de negocio de la presentación de datos, mejorando la claridad y evitando la exposición directa de las entidades de base de datos.
 - **Inyección de dependencias con @Autowired y RequiredArgsConstructor:** Para garantizar un código más modular y fácilmente testeable sin depender de instancias explícitas.
 - **Uso de Anotaciones de Transacción (@Transactional):** Para garantizar que las operaciones con bases de datos se ejecuten correctamente con control de transacciones y rollback en caso de errores.
 - **Manejo adecuado de excepciones:** Con excepciones personalizadas para casos específicos como cuando un recurso no se encuentra, facilitando el mantenimiento y la lectura.
 - **Uso de servicios y controladores bien definidos:** Separando la lógica de negocio (en servicios) de las operaciones HTTP (en

controladores), lo que mejora la organización y facilita el testing unitario y de integración.

- **Configuración clara de las propiedades del proyecto:** Para facilitar la personalización de las configuraciones, como la base de datos y los servidores de correo.
- **Uso de repositorios (Repository Pattern):** Para centralizar el acceso a la base de datos y facilitar la reutilización del código.
- **Validaciones de datos:** Validando las entradas en controladores para asegurar que los datos pasen correctamente antes de ser procesados.
- **Uso de Schedulers:** Para automatizar la publicación de contenido a una hora programada, lo que añade valor a la experiencia del usuario.
- **Testing a través de Postman:** Validamos las funcionalidades del CRUD y las operaciones de las APIs.

5.2 Web Service API Restful Implementation

5.2.1. Sprint Backlog 1

NRO	EPIC	US	Título	Story Point	Prioridad
1	EPIC 01	US 01	Registro de Usuarios	1	Alta
2	EPIC 01	US 02	Visualización de Perfil	2	Alta
3	EPIC 02	US 07	Crear categoría	2	Alta
4	EPIC 02	US 08	Visualizar categoría	2	Alta
5	EPIC 02	US 09	Editar Categoría	2	Alta
6	EPIC 02	US 10	Crear Blog	2	Alta
7	EPIC 02	US 11	Editar Blog	3	Alta
8	EPIC 02	US 12	Programación de Blog	5	Alta
9	EPIC 02	US 13	Crear Video	3	Alta
10	EPIC 02	US 14	Registro de Video	2	Alta

11	EPIC 02	US 15	Editar Video	2	Alta
12	EPIC 02	US 17	Eliminar Video	1	Alta
13	EPIC 02	US 18	Crear playlist	3	Media
14	EPIC 02	US 19	Visualizar playlist	2	Media
15	EPIC 02	US 22	Añadir video a la lista de reproducción del usuario	3	Media
16	EPIC 02	US 23	Ver la lista de videos en la lista de reproducción del usuario	2	Media
17	EPIC 02	US 24	Eliminar video de la lista de reproducción del usuario	2	Media

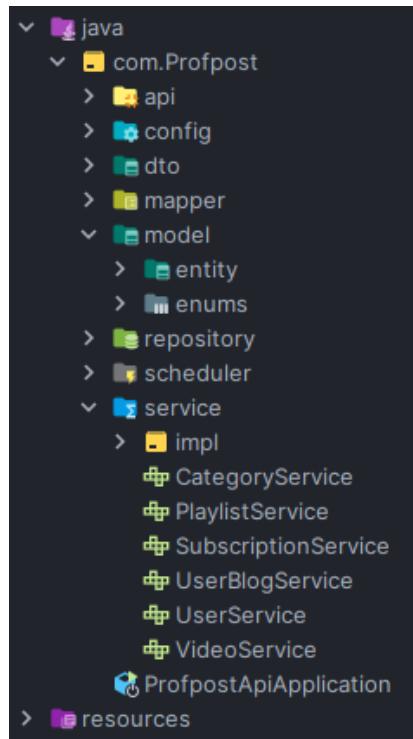
PROPOST-18 US01 - Registro de Usuario
 PROPOST-20 US02 - Visualización de Perfil
 PROPOST-7 US11 - Editar Blog
 PROPOST-11 US13 - Crear Video
 PROPOST-16 US12 - Programación de Blog
 PROPOST-36 US22 - Añadir video a la lista de reproducción del usuario
 PROPOST-43 US10 - Crear Blog
 PROPOST-44 US17 - Eliminar Video
 PROPOST-45 US07 - Crear Categoría
 PROPOST-47 US24 - Eliminar video de la lista de reproducción del usuario
 PROPOST-48 US23 - Ver la lista de videos en la lista de reproducción del usuario
 PROPOST-102 US15 - Editar Video
 PROPOST-103 US14 - Registro de Video
 PROPOST-105 US09 - Editar Categoría
 PROPOST-106 US08 - Visualizar Categoría
 PROPOST-109 US19 - Visualizar playlist
 PROPOST-107 US18 - Crear playlist

Wed, Sep 18 2024, 10:57pm Sprint completado 39

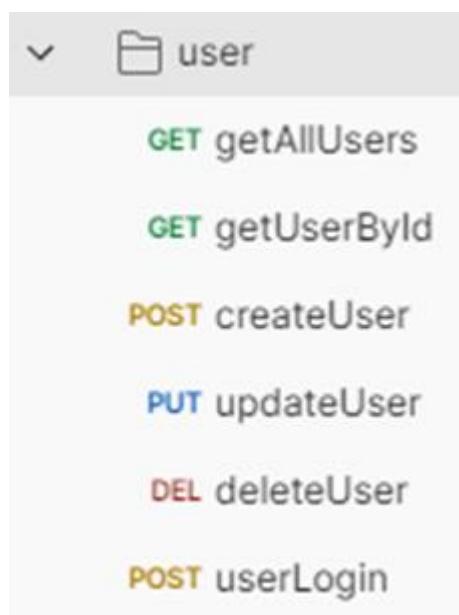
5.2.2. Development Evidence for Sprint Review

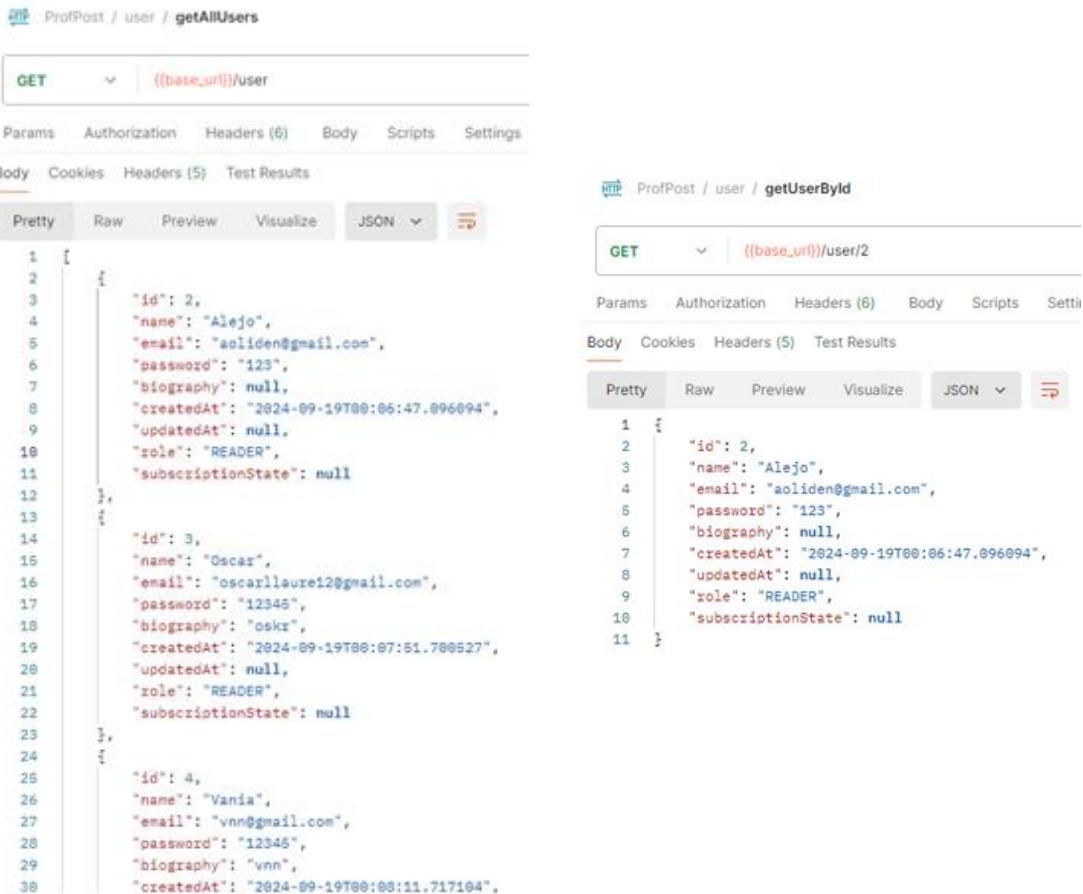
api	Controladores REST que gestionan las solicitudes y respuestas de la API
config	Configuraciones de la aplicación, incluyendo Schedulers
dto	Data Transfer Objects para la transferencia de datos entre frontend y backend
model	Entidades mapeadas a tablas de la base de datos
entity	Enumeraciones utilizadas en la aplicación para roles y estados
enums	Repositorios para la gestión de la persistencia de datos

repository	Repositorios para la gestión de la persistencia de datos
scheduler	Tareas programadas para la publicación automática de contenido
service	Interfaces de servicios que contienen la lógica de negocio
impl	Implementaciones de servicios que interactúan con los repositorios



5.2.3. Execution Evidence for Sprint Review



The screenshot shows the Postman application interface with two separate requests side-by-side.

Request 1: getAllUsers

- Method: GET
- URL: `((base_url))/user`
- Body tab is selected.
- JSON response (Pretty printed):

```
1  [
2    {
3      "id": 2,
4      "name": "Alejo",
5      "email": "aoliden@gmail.com",
6      "password": "123",
7      "biography": null,
8      "createdAt": "2024-09-19T00:06:47.096094",
9      "updatedAt": null,
10     "role": "READER",
11     "subscriptionState": null
12   },
13   {
14     "id": 3,
15     "name": "Oscar",
16     "email": "oscarlaurei2@gmail.com",
17     "password": "12345",
18     "biography": "oskr",
19     "createdAt": "2024-09-19T00:07:51.700527",
20     "updatedAt": null,
21     "role": "READER",
22     "subscriptionState": null
23   },
24   {
25     "id": 4,
26     "name": "Vania",
27     "email": "vnn@gmail.com",
28     "password": "12345",
29     "biography": "vnn",
30     "createdAt": "2024-09-19T00:08:11.717104",
31   }
]
```

Request 2: getUserById

- Method: GET
- URL: `((base_url))/user/2`
- Body tab is selected.
- JSON response (Pretty printed):

```
1  {
2    "id": 2,
3    "name": "Alejo",
4    "email": "aoliden@gmail.com",
5    "password": "123",
6    "biography": null,
7    "createdAt": "2024-09-19T00:06:47.096094",
8    "updatedAt": null,
9    "role": "READER",
10   "subscriptionState": null
11 }
```

HTTP ProfPost / user / createUser

POST | [\(\(base_url\)\)/auth/register](((base_url))/auth/register)

Params	Authorization	Headers (8)	Body	Scripts	Settings
<input type="radio"/> none	<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw	<input type="radio"/> binary	<input type="radio"/> GraphQL

```

1  {
2    "name": "Juan",
3    "biography": "Soy juan, lector",
4    "email": "jumanit@gmail.com",
5    "password": "12345",
6    "role": "READER"
7  }
8

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2    "id": 7,
3    "name": "Juan",
4    "email": "jumanit@gmail.com",
5    "password": "12345",
6    "biography": "Soy juan, lector",
7    "createdAt": "2024-09-19T03:17:56.3138614",
8    "updatedAt": null,
9    "role": "READER",
10   "subscriptionState": null
11 }

```

HTTP ProfPost / user / updateUser

PUT | [\(\(base_url\)\)/user/4](((base_url))/user/4)

Params	Authorization	Headers (8)	Body	Scripts	Settings
<input type="radio"/> none	<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw	<input type="radio"/> binary	<input type="radio"/> GraphQL

```

1  {
2    "name": "Sergio",
3    "email": "sevelasquezro@gmail.com",
4    "password": "12345",
5    "role": "READER"
6  }

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2    "id": 4,
3    "name": "Sergio",
4    "email": "sevelasquezro@gmail.com",
5    "password": "12345",
6    "biography": null,
7    "createdAt": "2024-09-19T00:08:11.717104",
8    "updatedAt": "2024-09-19T03:18:35.6951415",
9    "role": "CREATOR",
10   "subscriptionState": null
11 }

```

HTTP ProfPost / user / deleteUser

DELETE | [\(\(base_url\)\)/user/4](((base_url))/user/4)

HTTP ProfPost / user / userLogin

POST | [\(\(base_url\)\)/auth](((base_url))/auth)

Params	Authorization	Headers (8)	Body	Scripts	Settings
<input type="radio"/> none	<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw	<input type="radio"/> binary	<input type="radio"/> GraphQL

```

1  {
2    "email": "mauriccioreyes@gmail.com",
3    "password": "12345"
4  }

```

Body Cookies Headers (5) Test Results

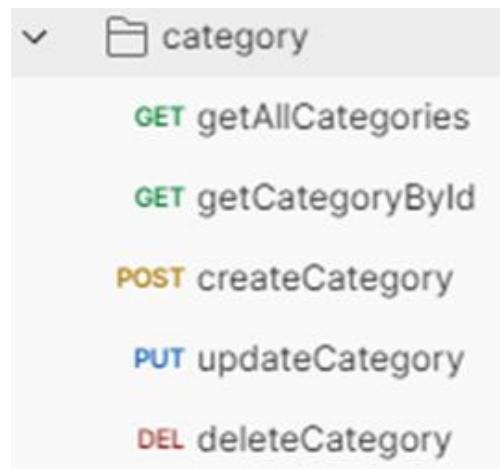
Pretty Raw Preview Visualize Text

```

1  Login successful

```

200 OK



HTTP ProfPost / category / getAllCategories

GET `((base_url))/category`

Params	Authorization	Headers (6)	Body	Scripts	Settings
Body	Cookies	Headers (5)	Test Results		
Pretty	Raw	Preview	Visualize	JSON	

```

1  [
2    {
3      "id": 1,
4      "name": "Blogs",
5      "description": "Mis blogs",
6      "createdAt": "2024-09-19T00:13:50.244858",
7      "updatedAt": null
8    },
9    {
10       "id": 2,
11       "name": "Videos",
12       "description": "Mis videos",
13       "createdAt": "2024-09-19T00:14:09.975003",
14       "updatedAt": null
15     }
16   ]

```

HTTP ProfPost / category / getCategoryById

GET `((base_url))/category/1`

Params	Authorization	Headers (6)	Body	Scripts	Settings
Body	Cookies	Headers (5)	Test Results		
Pretty	Raw	Preview	Visualize	JSON	

```

1  {
2    "id": 1,
3    "name": "Blogs",
4    "description": "Mis blogs",
5    "createdAt": "2024-09-19T00:13:50.244858",
6    "updatedAt": null
7  }

```

POST {{base_url}}/category

```

1 {
2   "name": "prueba",
3   "description": "Mis pruebas"
4 }
    
```

PUT {{base_url}}/category/1

```

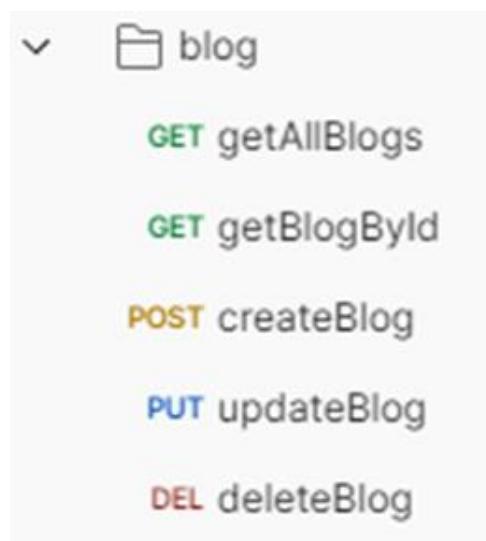
1 {
2   "name": "leer",
3   "description": "descripcion de lectura"
4 }
    
```

HTTP `ProPost / category / deleteCategory`

DELETE {{base_url}}/category/4

Body `Text`

```
1
```



ProfPost / blog / getAllBlogs

GET http://{{base_url}}/blogs

Params	Authorization	Headers (6)	Body	Scripts	Settings
body	Cookies	Headers (5)	Test Results		
Pretty	Raw	Preview	Visualize	JSON	

```

1 [
2   {
3     "id": 2,
4     "title": "Mi primer blog programado",
5     "content": "Este es el contenido de mi primer blog programado.",
6     "coverPath": null,
7     "filePath": null,
8     "createdAt": "2024-09-19T00:26:36.718431",
9     "updatedAt": null,
10    "schedulePublishAt": "2024-09-19T00:30:00",
11    "category": {
12      "id": 1,
13      "name": "leer",
14      "description": "descripcion de lectura",
15      "createdAt": "2024-09-19T00:13:58.244858",
16      "updatedAt": "2024-09-19T03:23:27.188878"
17    },
18    "user": {
19      "id": 4,
20      "name": "Sergio",
21      "email": "sevelasquezro@gmail.com",
22      "password": "12345",
23      "biography": null,
24      "createdAt": "2024-09-19T00:08:11.717104",
25      "updatedAt": "2024-09-19T03:18:35.695142",
26      "role": "CREATOR",
27      "subscriptionState": null
28    },
29    "published": true
30  }

```

ProfPost / blog / getBlogById

GET http://{{base_url}}/blogs/2

Params	Authorization	Headers (6)	Body	Scripts	Settings
body	Cookies	Headers (5)	Test Results		
Pretty	Raw	Preview	Visualize	JSON	

```

1 {
2   "id": 2,
3   "title": "Mi primer blog programado",
4   "content": "Este es el contenido de mi primer blog programado.",
5   "coverPath": null,
6   "filePath": null,
7   "createdAt": "2024-09-19T00:26:36.718431",
8   "updatedAt": null,
9   "schedulePublishAt": "2024-09-19T00:30:00",
10  "category": {
11    "id": 1,
12    "name": "leer",
13    "description": "descripcion de lectura",
14    "createdAt": "2024-09-19T00:13:58.244858",
15    "updatedAt": "2024-09-19T03:23:27.188878"
16  },
17  "user": {
18    "id": 4,
19    "name": "Sergio",
20    "email": "sevelasquezro@gmail.com",
21    "password": "12345",
22    "biography": null,
23    "createdAt": "2024-09-19T00:08:11.717104",
24    "updatedAt": "2024-09-19T03:18:35.695142",
25    "role": "CREATOR",
26    "subscriptionState": null
27  },
28  "published": true
29

```

ProfPost / blog / createBlog

POST http://{{base_url}}/blogs/creators

Params	Authorization	Headers (8)	Body	Scripts	Settings
<input type="radio"/> none	<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw	<input type="radio"/> binary	<input type="radio"/> GraphQL
Body					

```

1 {
2   "title": "Mi primer blog programado",
3   "content": "Este es el contenido de mi primer blog programado.",
4   "category_id": 1,
5   "user_id": 4
6 }

```

ProfPost / blog / updateBlog

PUT http://{{base_url}}/blogs/creators/2

Params	Authorization	Headers (8)	Body	Scripts	Settings
<input type="radio"/> none	<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw	<input type="radio"/> binary	
Body					

```

1 {
2   "title": "Mi séptimo blog actualizado",
3   "content": "Este es el contenido actualizado de mi séptimo blog",
4   "category_id": 1,
5   "user_id": 4
6 }

```

ProfPost / blog / getAllBlogs

GET http://{{base_url}}/blogs

Params	Authorization	Headers (6)	Body	Scripts	Settings
body	Cookies	Headers (5)	Test Results		
Pretty	Raw	Preview	Visualize	JSON	

```

1 [
2   {
3     "id": 3,
4     "title": "Mi primer blog programado",
5     "content": "Este es el contenido de mi primer blog programado.",
6     "coverPath": null,
7     "filePath": null,
8     "createdAt": "2024-09-19T03:25:59.8983322",
9     "updatedAt": null,
10    "schedulePublishAt": null,
11    "category": {
12      "id": 1,
13      "name": "leer",
14      "description": "descripcion de lectura",
15      "createdAt": "2024-09-19T00:13:58.244858",
16      "updatedAt": "2024-09-19T03:23:27.188878"
17    },
18    "user": {
19      "id": 4,
20      "name": "Sergio",
21      "email": "sevelasquezro@gmail.com".

```

HTTP ProfPost / blog / deleteBlog

DELETE {{base_url}}/blogs/creators/2

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (3) Test Results 204 No Content

Pretty Raw Preview Visualize Text ▾

1

The screenshot shows a REST API testing interface. At the top, it displays the URL as 'HTTP ProfPost / blog / deleteBlog'. Below that, a 'DELETE' button and the endpoint '{{base_url}}/blogs/creators/2' are shown. A navigation bar includes 'Params', 'Authorization', 'Headers (6)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is selected, showing 'Cookies', 'Headers (3)', and 'Test Results' options. To the right, a green box indicates a '204 No Content' response. Below the tabs are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'Text' with a dropdown arrow. The main body area contains the number '1'. At the bottom, there's a large button labeled 'subscription' with a folder icon, and below it are two links: 'POST subscribe' in orange and 'DEL unsubscribe' in red.

ProPost / subscription / subscribe

POST | {{base_url}}/subscription

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary

```
1 {  
2   "user_id": 3,  
3   "creator_id": 4  
4 }  
5
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▾

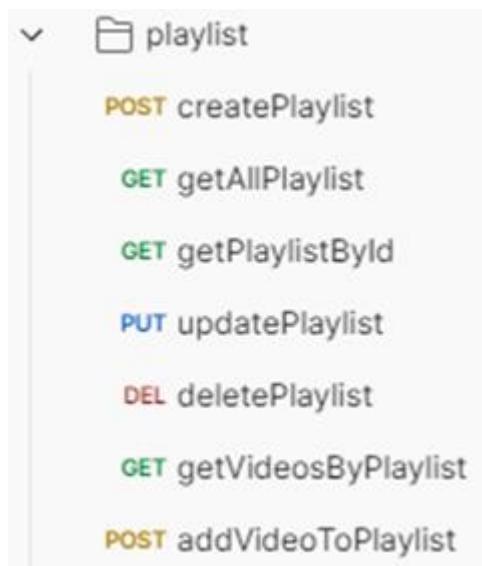
DELETE | {{base_url}}/subscription/3

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 {  
2   "subscriptionId": 3,  
3   "userId": null,  
4   "creatorId": null,  
5   "status": "Success",  
6   "message": "Subscription will end after 30 days!"  
7 }
```



[ProfPost / playlist / createPlaylist](#)

POST {{base_url}}/playlist

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw bin

```
1 {  
2 |   "name": "prueba"  
3 }  
Body Cookies Headers (5) Test Results
```

Pretty Raw Preview Visualize JSON

```
1 {  
2 |   "id": 4,  
3 |   "name": "prueba",  
4 |   "created_at": "2024-09-19T03:30:39.5293641",  
5 |   "updated_at": null  
6 }  
Body Cookies Headers (5) Test Results
```

[ProfPost / playlist / getAllPlaylist](#)

GET {{base_url}}/playlist

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [  
2 | {  
3 | |   "id": 1,  
4 | |   "name": "playlist actualizado",  
5 | |   "created_at": "2024-09-19T00:34:46.666898",  
6 | |   "updated_at": "2024-09-19T00:47:30.298491"  
7 | },  
8 | {  
9 | |   "id": 2,  
10 | |   "name": "Playlist",  
11 | |   "created_at": "2024-09-19T00:47:43.977531",  
12 | |   "updated_at": null  
13 | },  
14 ]  
Body Cookies Headers (5) Test Results
```

HTTP ProPost / playlist / `getPlaylistById`

GET `((base_url))/playlist/1`

Params	Authorization	Headers (6)	Body	Scripts	Settings
Body	Cookies	Headers (5)	Test Results		
Pretty	Raw	Preview	Visualize	JSON	

```

1 {
2   "id": 1,
3   "name": "playlist actualizado",
4   "created_at": "2024-09-19T00:34:46.666898",
5   "updated_at": "2024-09-19T00:47:30.298491"
6 }

```


HTTP ProPost / playlist / `updatePlaylist`

PUT `((base_url))/playlist/1`

Params	Authorization	Headers (8)	Body	Scripts	Settings
<input type="radio"/> none	<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw	<input type="radio"/>	

```

1 {
2   "name": "playlist actualizado"
3 }

```


Body	Cookies	Headers (5)	Test Results	
Pretty	Raw	Preview	Visualize	JSON

```

1 {
2   "id": 1,
3   "name": "playlist actualizado",
4   "created_at": "2024-09-19T00:34:46.666898",
5   "updated_at": "2024-09-19T03:32:00.6879147"
6 }

```


HTTP ProPost / playlist / `getVideosByPlaylist`

GET `((base_url))/playlist/1/videos`

Params	Authorization	Headers (6)	Body	Scripts	Settings
Body	Cookies	Headers (5)	Test Results	200 OK	
Pretty	Raw	Preview	Visualize	JSON	

```

1 [
2   {
3     "id": 3,
4     "title": "Mi hole mundo en video",
5     "url": "https://youtu.be/XccPGAAizZA7si=h5QosSLTfgd328ifg832321wXgt",
6     "duration": 0.0,
7     "coverPath": null,
8     "filePath": null,
9     "createdAt": "2024-09-19T00:22:40.86937",
10    "updatedAt": "2024-09-19T00:52:48.267816",
11    "schedulePublishAt": null,
12    "category": {
13      "id": 2,
14      "name": "Videos",
15      "description": "Mis videos",
16      "createdAt": "2024-09-19T00:14:09.975003",
17      "updatedAt": null
18    },
19    "user": {
20      "id": 4,
21      "name": "Sergio",
22      "email": "sevelasquezro@gmail.com",
23      "password": "12345",
24      "biography": null,
25      "createdAt": "2024-09-19T00:00:11.717104",
26      "updatedAt": "2024-09-19T03:18:35.695142",
27      "role": "CREATOR",
28      "subscriptionState": null
29    },
30    "playlist": f

```


HTTP ProPost / playlist / `addVideoToPlaylist`

POST `((base_url))/playlist/add/video?playlistId=1&videoId=5`

Params	Authorization	Headers (7)	Body	Scripts	Settings
Body	Cookies	Headers (5)	Test Results	200 OK	
Pretty	Raw	Preview	Visualize	JSON	

```

1 {
2   "id": 8,
3   "title": "Mi hola mundo en video programado",
4   "url": "https://youtu.be/XccPGAAizZA7si=h5QosSLTfgd328ifg832321wXgt",
5   "duration": 0.0,
6   "coverPath": null,
7   "filePath": null,
8   "createdAt": "2024-09-19T00:27:38.265929",
9   "updatedAt": "2024-09-19T03:33:19.321577",
10  "schedulePublishAt": "2024-09-19T00:38:00",
11  "category": {
12    "id": 2,
13    "name": "Videos",
14    "description": "Mis videos",
15    "createdAt": "2024-09-19T00:14:09.975003",
16    "updatedAt": null
17  },
18  "user": {
19    "id": 4,
20    "name": "Sergio",
21    "email": "sevelasquezro@gmail.com",
22    "password": "12345",
23    "biography": null,
24    "createdAt": "2024-09-19T00:00:11.717104",
25    "updatedAt": "2024-09-19T03:18:35.695142",
26    "role": "CREATOR",
27    "subscriptionState": null
28  },
29  "playlist": {
30    "id": 1

```

HTTP ProfPost / playlist / deletePlaylist

DELETE `{{base_url}}/playlist/2`

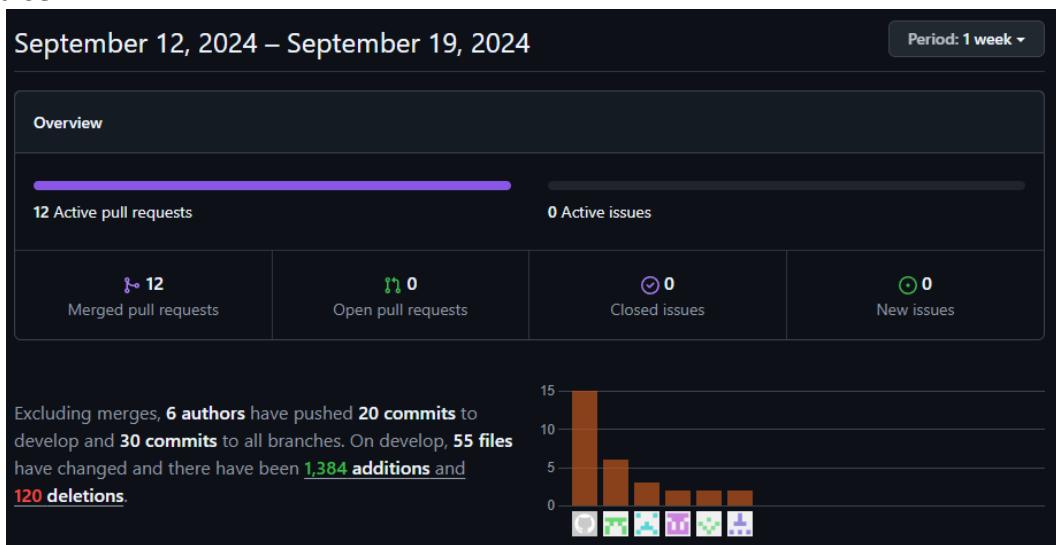
Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (3) Test Results **204 No Content**

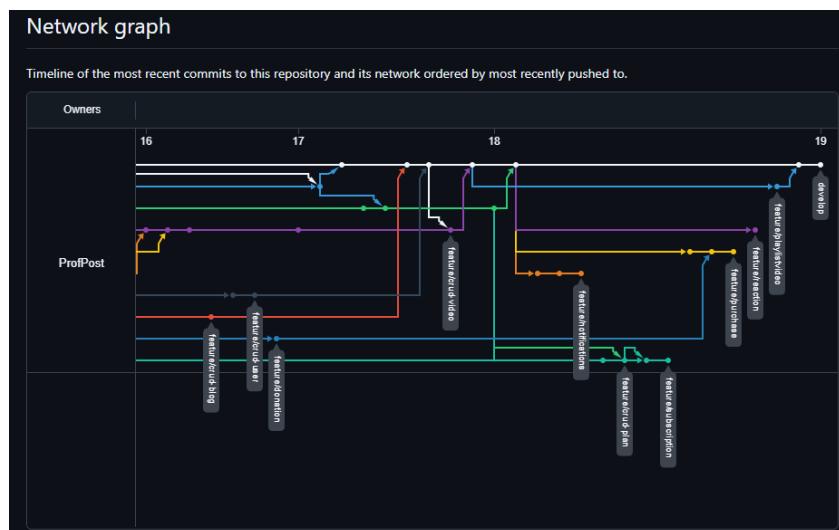
Pretty Raw Preview Visualize Text **1**

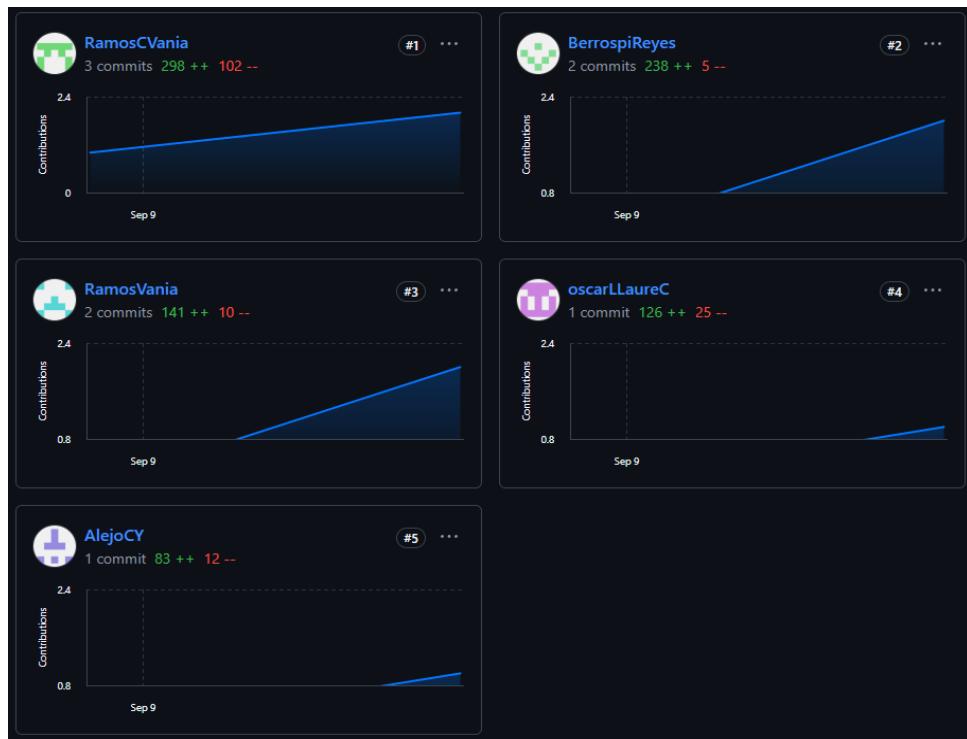
5.2.6. Team Collaboration Insights during Sprint

Pulse:



Network:



Commits:**Contributors:**

Conclusiones

Durante el desarrollo de este proyecto, se implementaron diversas tecnologías y buenas prácticas para garantizar un código limpio, eficiente y mantenible. Se utilizó Spring Boot como el framework base para desarrollar el backend, aprovechando sus características de inyección de dependencias, configuración automática, y el robusto soporte para JPA/Hibernate para la interacción con la base de datos PostgreSQL. Cada entidad fue cuidadosamente diseñada con relaciones y mapeos adecuados, mientras que las tareas programadas (schedulers) se implementaron para automatizar la publicación de blogs y videos.

Uso de Git

Git fue la herramienta de control de versiones elegida, permitiendo una gestión eficiente del código fuente a través de ramas organizadas por funcionalidades (feature branches). Se realizaron commits constantes para reflejar el progreso, y se usaron pull requests para integrar nuevas características a la rama principal, garantizando revisiones de código y evitando errores. El flujo de trabajo basado en Git permitió una colaboración fluida y una implementación ágil de nuevas funcionalidades, además de facilitar la resolución de conflictos en los merges y asegurar un historial limpio de cambios.

Codificación

Se aplicaron convenciones estrictas de codificación para garantizar la claridad y legibilidad del código, utilizando nombres descriptivos para clases, métodos y variables. Los controladores REST gestionan las operaciones CRUD de las entidades, mientras que los DTOs facilitaron la transferencia de datos entre el cliente y el servidor. La estructura modular del proyecto permite una fácil expansión y mantenimiento a largo plazo.

En resumen, el proyecto logró un balance eficiente entre el uso de herramientas modernas, buenas prácticas de codificación y una gestión eficiente del ciclo de desarrollo con Git, lo que garantiza una base sólida y escalable para futuras mejoras y expansiones del sistema.

Recomendaciones

Iniciar un proyecto con una planificación detallada es crucial para el éxito del mismo. La planificación permite definir claramente los objetivos, los recursos necesarios y los tiempos estimados de desarrollo, lo que ayuda a evitar contratiempos y malentendidos durante el ciclo de vida del proyecto. Al estructurar las tareas y priorizarlas desde el inicio, se asegura que todo el equipo esté alineado y se pueda enfocar en cumplir los entregables de manera eficiente. Además, la planificación permite anticipar riesgos y posibles bloqueos, lo que facilita la toma de decisiones proactivas.

Las entrevistas son fundamentales para captar los requisitos de los usuarios y asegurarse de que el desarrollo esté alineado con sus necesidades reales. A través de entrevistas, se pueden obtener insights valiosos directamente de las personas que usarán el sistema, lo que garantiza que las funcionalidades desarrolladas sean realmente útiles y resuelvan problemas concretos. Las entrevistas también ayudan a aclarar expectativas, identificar prioridades y ajustar el alcance

del proyecto según lo que los usuarios valoran más. Esto no solo mejora la calidad del producto, sino que también reduce la cantidad de cambios y correcciones posteriores.

Bibliografía

Spring Framework Documentation (2023). *Spring Framework Documentation*
<https://spring.io/projects/spring-framework>.

Documentación oficial de Spring Framework, que cubre las funcionalidades de este marco para construir aplicaciones robustas y escalables.

PostgreSQL Global Development Group (2023). *PostgreSQL Documentation*
<https://www.postgresql.org/docs/>.

Documentación oficial de PostgreSQL, utilizada como referencia en la configuración y uso de la base de datos en este proyecto.