UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"EVALUÓMETRO: DISEÑO E IMPLEMENTACIÓN DE UNA RED SOCIAL PARA EVALUAR PRODUCTOS Y SERVICIOS"

TRABAJO FIN DE GRADO

Julio - 2023

AUTOR: Jesús Luna García
DIRECTOR/ES: Jesús Javier Rodríguez Sala

RESUMEN

Este proyecto de trabajo fin de grado, denominado "Evaluómetro", podría innovar en el ámbito de las redes sociales mediante la creación de una plataforma digital, diferencial en muchos sentidos. El principio fundamental de la plataforma Evaluómetro es ofrecer un espacio interactivo en el que los usuarios puedan contribuir y solicitar evaluaciones sobre cualquier ámbito, que incluye productos, servicios, experiencias y obras de arte, entre otros.

Este proyecto exige que cada usuario cree una cuenta, un requisito que fomenta la participación activa y comprometida. Este requisito previo se aplica con el objetivo de estimular la participación de los usuarios desde el principio, mitigando cualquier posible obstrucción que pueda surgir debido al retraso en el registro.

La plataforma Evaluómetro hace especial hincapié en los contenidos generados por los usuarios, un potente catalizador para la retroalimentación y la difusión del conocimiento. Cada Post en la plataforma, requiere la inclusión de un título y una calificación, lo que proporciona una estructura coherente y susceptible de múltiples aplicaciones prácticas, dando pie a evaluar sobre cualquier cosa. Por ejemplo, una persona que dude sobre la elección de una película podría realizar una búsqueda específica en Evaluómetro y obtener así información oportuna y valiosa sobre las opciones más pertinentes y aplaudidas dentro de su género de interés.

De forma paralela, Evaluómetro puede servir de ayuda indispensable para viajeros que busquen recomendaciones localizadas, como establecimientos gastronómicos en una región concreta, o para particulares que busquen opiniones sobre productos y servicios específicos.

Es esencial reconocer que la eficacia y utilidad de Evaluómetro están íntimamente ligadas al calibre y volumen de su comunidad de usuarios. Mientras que una base de usuarios escasa puede impedir la eficacia de la plataforma para consultas especializadas, una comunidad de usuarios sólida y comprometida tiene el potencial de elevar Evaluómetro a una reserva inestimable de información y perspectivas de utilidad.

Además, para el desarrollador del proyecto, la realización de este, representa una expedición académica y profesional. A través de la implementación de los diversos aspectos y funcionalidades de Evaluómetro, se han adquirido y aplicado conocimientos cruciales en desarrollo web, administración de bases de datos y una profunda comprensión de la dinámica de las redes sociales. Esta experiencia ha aumentado significativamente la formación y competencia del desarrollador, en el campo de la ingeniería de software y en el desarrollo web.

AGRADECIMIENTOS

Primero y antes de nada, quiero expresar mi más sincero agradecimiento a todas las personas que han formado parte de este viaje académico, que finaliza con la presentación de este Trabajo de Fin de Grado.

En primer lugar, a mis compañeros de carrera, que han compartido conmigo las alegrías y desafíos de estos años de aprendizaje y desarrollo personal. Sin la colaboración, el apoyo mutuo y los momentos de compañerismo, este viaje no habría sido tan enriquecedor.

Quiero hacer mención especial a Adrián Garrido Pantaleón, un compañero de estudios insustituible con el que he compartido innumerables horas de estudio, preparación de asignaturas y exámenes. Adrián, tus esfuerzos, tu dedicación y tu amistad han sido inestimables durante toda la carrera.

Por otro lado, deseo expresar mi gratitud a los profesores, muchos de los cuales han constituido una gran inspiración personal y me llevo esos recuerdos conmigo. Quisiera hacer especial mención al profesor Jesús Javier Rodríguez Sala, quien ha supervisado y asistido en la elaboración de este proyecto final. Su guía, su apoyo y su compromiso han sido cruciales en la realización de este trabajo.

Por último, pero no menos importante, mi agradecimiento más profundo a mi familia, amigos y a Alba Vivó Serrano. Su apoyo constante, su cariño y su ánimo han sido el sostén que me ha permitido mantenerme firme en los momentos más difíciles. Han estado a mi lado, proporcionándome la fuerza necesaria para superar todos los obstáculos y animándome en cada paso del camino.

La culminación de este Trabajo de Fin de Grado es una reflexión de todos los esfuerzos, la dedicación y el cariño que proceso a todas estas personas maravillosas. A todas ellas, mi más sincero agradecimiento.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Capítulo 1
Introducción

1.1.- LA VINCULACIÓN DE LA INFORMACIÓN LIVIANA A LA COMODIDAD DE VIDA

1.1.1.- CRECIMIENTO DE LAS REDES SOCIALES

Desde la llegada de los teléfonos móviles inteligentes, las redes sociales han experimentado una gran inclusión en la vida diaria de las personas, tanto es así que nos hemos acostumbrado a vivir con ellas de manera cotidiana. A día de hoy, las redes sociales abarcan al 59% de la población mundial, manteniendo a millones de personas conectadas cada día [1].

Aunque principalmente fueron inventadas para el entretenimiento de los usuarios, permitiendo cosas como el reencuentro con viejos amigos o poder conocer a personas con

gustos, aficiones o ideas similares, la realidad es que hoy en día las redes sociales pueden tener muchas funciones, por lo que, según un artículo de www.rdstation.com podemos dividirlas en las siguientes categorías [2]:

- Redes sociales de relaciones: este es el tipo de red social más utilizada, se centran en los contactos y relaciones interpersonales entre individuos. Algunos ejemplos son Facebook, Instagram, Twitter, etc.

- Redes sociales de entretenimiento: el objetivo principal de estas redes es el de consumir todo tipo de contenidos. Youtube es quizá el ejemplo más conocido. Otros ejemplos serían redes como Pinterest, donde las personas publican y consumen imágenes (recetas, diseño, fotografía, moda, etc.), o Twitch desarrollada para la retransmisión de vídeo en vivo.

- Redes sociales profesionales: en este caso, la finalidad principal es que los usuarios generen relaciones profesionales, presentando sus currículums, habilidades y experiencia profesional. LinkedIn es la más conocida y utilizada, aunque hay más, como Bebee, Bayt, etc.

- Redes sociales de nicho: este es un grupo muy amplio de redes, están dirigidas a públicos específicos con interés en un tema concreto, ya sean el arte, el alquiler o alojamiento, la gastronomía, etc. Algunas de las más conocidas son TripAdvisor, DeviantArt o Goodreads.

Aunque estas son las funciones principales de las redes sociales, la realidad es que clasificarlas para un uso en concreto es muy complicado, ya que prácticamente todas ellas se pueden emplear para funciones muy diversas, dependiendo de como decidan utilizarlas sus usuarios.

1.1.2.- EL CONSUMO EN LA SOCIEDAD

Es innegable que en la sociedad actual, pasamos la mayoría de nuestro tiempo consumiendo o pensando en consumir comida, experiencias, cine, y todo tipo de productos o servicios. Nos hemos convertido en una sociedad consumista y cómoda, en la que buscamos conseguir productos fácilmente desechables, en la que el individuo se preocupa del presente y de satisfacer sus necesidades inmediatas. Esto se puede ver reflejado en que la mayoría de productos que utilizamos no entendemos cómo funcionan, dando lugar a que cuando se rompen, son directamente reemplazados por otro nuevo, ya que es más cómodo conseguir uno que funciona, que entender el funcionamiento del anterior y tratar de repararlo.

1.2.- JUSTIFICACIÓN DEL PROYECTO

Para este proyecto he decidido hacer una red social, puesto que me motivaría mucho crear un sitio de Internet independiente, dirigido únicamente por sus usuarios, en el que las cosas se evalúen de manera neutra, sin que haya una empresa o marca de por medio que puedan manipular o influir en las valoraciones.

Por otro lado, me pareció muy interesante poder crear un sitio donde hubiese mucha información concentrada, de manera que se pudiera saber lo básico e importante sobre algo, antes de consumirlo, para así poder hacer mejor uso del tiempo y el dinero.

Por este motivo decidí llamar a esta red social Evaluómetro. Me inspiré en una red social ya existente llamada Chollómetro, cuyo enfoque principal son las ofertas, los usuarios van colgando ofertas sobre productos, de manera que cuando se desea comprar algo rápidamente y que resulte económico, es muy práctico ir a esa red social a buscarlo, ya que, se encuentran precios muchos más económicos, sin necesidad de que el usuario tenga que estar pendiente del precio de los productos. Por esto, dado que mi red social tiene un enfoque parecido, pero dando más importancia a la evaluación de cosas, decidí llamarla Evaluómetro.

Por último, pretendo que Evaluómetro me sirva como vía de esfuerzo para desarrollarme con algunas tecnologías de back end, de manera que me motiva a conseguir hacerlo funcional.

1.2.1.- AUMENTAR LA EFICACIA DEL TIEMPO Y DINERO DE LOS USUARIOS

Dado el crecimiento de las redes sociales y las necesidades que tienen los usuarios rutinariamente, a causa de la gran cantidad de inputs que tienen cada minuto, tener un sitio web donde poder acceder a la mayor cantidad de información de interés posible, de una manera rápida y eficaz, es una propuesta que cobra sentido.

A la hora de la verdad, las personas buscan la mejor manera de invertir su tiempo para encontrar el precio más asequible posible, por lo que siempre es interesante tener la opinión de alguien con experiencia previa que pueda servir de guía con aquello que se desea obtener, pudiendo invertir así tu dinero de la forma más eficiente posible.

Además, al ser una red social donde el contenido se iría creando y renovando, puede convertirse en un punto de interés de los usuarios, creando contenido con el fin de atraer la

atención de las personas y no tan solo acceder a la red social, únicamente cuando haya una necesidad de por medio. De esta forma, Evaluómetro se convertiría en un dominio al que se acudiría frecuentemente, ya que, surgen muchas necesidades relacionadas con el consumo diario. De este modo se generaría una atracción por recibir información acerca de lo que están evaluando los usuarios actualmente y sus opiniones sobre un producto o servicio en concreto.

1.2.2.- MOTIVACIÓN PERSONAL

Personalmente es un proyecto que me motiva, ya que, frecuentemente pienso en cosas que mejorar o adquirir, y aunque no tengo idea de todo, intento informarme lo máximo posible para aprender lo suficiente sobre aquello que quiero conseguir o mejorar, de manera que pueda apostar siempre por lo más óptimo. Por este motivo, Evaluómetro es un sitio que usaría con frecuencia, por lo que yo mismo me haría usuario de una red social como esta.

En cuanto al ámbito profesional, me gustaría desarrollar mis habilidades como programador Python, Django, IA, GIT, etc., es por ello que he decidido realizar este proyecto con estas herramientas, ya que así también me formaré en su uso.

1.3.- OBJETIVOS

1.3.1.- OBJETIVOS GENERALES

Este proyecto tiene como objetivo principal la creación de una página web que funcione como una red social para evaluar los diferentes productos y servicios, que están presentes en nuestra rutina diaria. Aparte de buscar ganarse la confianza de los usuarios mediante su utilidad, Evaluómetro pretende tener un inicio entretenido y que sea acorde a los gustos de cada usuario, mediante algoritmos de IA, de manera que no se convierta únicamente en una red social a la que acudir cuando surja una necesidad, sino que también haya una motivación de usarla de forma diaria.

Los aspectos técnicos que se buscan implementar son:

- Registrarse, iniciar y cerrar sesión.
- Modificar perfil de usuario.
- Comentar/crear posts.
- Buscar usuarios/posts de interés.
- Seguir usuarios/posts de interés.

- Evaluar posts para crear opiniones globales.
- Resolver dudas a los usuarios.

1.3.2.- OBJETIVOS PERSONALES

En lo personal, el objetivo principal de este proyecto es adquirir conocimientos y práctica en la implementación de aplicaciones web, de manera que me puedan ser útiles y ponerlos en práctica como desarrollador en un futuro cercano. Además, dado que en el proyecto se abarcan conceptos back end, es un proceso que me puede servir para esa especialización.

Entre las tecnologías que se van usar, destacan:

- Python.
- Django
- HTML
- CSS
- INTELIGENCIA ARTIFICIAL
- GIT

1.4.- LÍMITES DEL PROYECTO

Como toda red social, Evaluómetro tiene un límite muy claro y es el de los usuarios. Toda red social, depende de un número mínimo de usuarios activos, sin los que la web carecería de sentido, ya que perdería toda la utilidad al no haber gente que sepa o pueda evaluar. Sin embargo, si se convirtiese en una red social con un número estable de usuarios activos, los límites serían mucho menores, ya que podría servir para conocer o saber prácticamente todo. Únicamente estaría limitada en búsquedas muy concretas, sobre las que no hubiese posts aún o sobre las que no hubiese usuarios que pudieran aconsejar o evaluar.

Por otro lado, sobre las tecnologías utilizadas en este proyecto, hay varias que están fuera de las competencias que se imparten en el grado, por lo que si se comercializara, su desarrollo sería muy costoso en tiempo y dinero.

Otro punto a tener en cuenta, sería la seguridad de la web, dado que, a parte de las herramientas que proporciona Django, como encriptación de claves, no se ha implementado nada más, ya que no tengo un conocimiento más profundo en esta área.

Capítulo 2 Antecedentes y estado de la cuestión

En la actualidad, la evaluación de productos comerciales está a la orden del día, por eso existen diferentes portales que se dedican a representar la opinión de los usuarios sobre distintos productos o servicios (hoteles, películas, coches, etc.).

Entre los servicios más comunes que podemos encontrar, hay webs que se dedican exclusivamente a evaluar productos, como algunas páginas web que se dedican a evaluar cine (IMDB, ROTTEN TOMATOES, METACRITIC, etc.), aunque también hay webs más diversas, que se dedican a evaluar tecnología o varios tipos de productos o servicios a la vez (Consumer Reports, Yelp, Angie's List, CNET, etc.). Sin embargo, lo más habitual es encontrar webs que ayudan a encontrar servicios o productos y a su vez tienen un sistema de evaluación incorporado en su web, de forma que se dedican a vender y evaluar a la vez (TripAdvisor, Amazon, Aliexpress, Blablacar, Airbnb, etc).

De entre los servicios que acabamos de mencionar, vamos a explicar algunos de cada categoría, sacando la información de sus funciones principales de sus respectivas webs,

para finalmente compararlos entre sí y con la propuesta que se presenta en este trabajo, y ver las diferencias y similitudes que tendrían con Evaluómetro.

2.1.- SITIOS DE EVALUACIÓN DE SECTORES ESPECÍFICOS (CINE)

2.1.1.- IMDB

IMDb (Internet Movie Database) es una base de datos en línea que contiene información sobre películas, programas de televisión, actores, directores, guionistas y otros profesionales de la industria del entretenimiento. Fundada en 1990, IMDb se convirtió en propiedad de Amazon en 1998 y se ha expandido para incluir noticias de entretenimiento, críticas y calificaciones de usuarios [3].

Las principales funciones de IMDb son:

- Búsqueda y exploración de películas, programas de televisión, actores, directores, guionistas y otros profesionales de la industria del entretenimiento.
- Ver reseñas y calificaciones de usuarios para películas y programas de televisión.
- Leer noticias y reseñas de entretenimiento.

2.1.2- ROTTEN TOMATOES

Rotten Tomatoes es un sitio web de reseñas y recomendaciones de películas y programas de televisión. Fue fundado en 1998 y se convirtió en propiedad de Fandango en 2016. El sitio web recopila críticas de películas y programas de televisión de varios críticos y publicaciones, calcula una calificación promedio basada en estas críticas y presenta un porcentaje de "fresco" o "podrido" [4].

Las principales funciones de Rotten Tomatoes son:

- Búsqueda y exploración de películas y programas de televisión.
- Ver reseñas y calificaciones de críticos de cine para películas y programas de televisión.
- Ver el porcentaje de "fresco" o "podrido" de una película o programa de televisión basado en críticas.

2.2.- SITIOS DE EVALUACIÓN DE VARIOS SECTORES

2.2.1.- CONSUMER REPORTS

Consumer Reports es una organización sin fines de lucro que brinda a los consumidores información independiente y confiable sobre productos y servicios en una variedad de categorías, desde automóviles hasta electrodomésticos y dispositivos electrónicos [5].

Las principales funciones de Consumer Reports son las siguientes:

- Pruebas y evaluaciones de productos: Consumer Reports realiza pruebas y evaluaciones de productos para evaluar su calidad y rendimiento. Estas pruebas se realizan en un laboratorio y cubren una amplia gama de categorías, desde automóviles hasta dispositivos electrónicos, electrodomésticos y productos para el hogar.

- Investigaciones y estudios de mercado: Consumer Reports lleva a cabo investigaciones y estudios de mercado para analizar las tendencias de consumo y brindar información sobre los mejores productos y servicios en cada categoría.

- Consejos y guías de compra: Consumer Reports proporciona consejos y guías de compra para ayudar a los consumidores a tomar decisiones informadas al comprar productos y servicios. Estos consejos incluyen información sobre características a considerar, precios y marcas recomendadas.

- Información sobre seguridad del consumidor: Consumer Reports también proporciona información sobre seguridad del consumidor, incluidas las últimas noticias sobre retiros de productos y problemas de seguridad.

2.2.2.- CNET

CNET es un sitio web de tecnología que ofrece noticias, reseñas y análisis de productos electrónicos de consumo, software y servicios en línea [6].

Las principales funciones de CNET son las siguientes:

- Noticias y análisis de productos: CNET publica noticias y análisis de productos electrónicos de consumo, software y servicios en línea. Estos análisis cubren una

amplia gama de categorías, desde teléfonos inteligentes y computadoras portátiles hasta dispositivos domésticos inteligentes y servicios en línea.

- Guías de compra: CNET proporciona guías de compra para ayudar a los consumidores a tomar decisiones informadas al comprar productos electrónicos de consumo, software y servicios en línea. Estas guías incluyen información sobre características a considerar, precios y marcas recomendadas.

- Consejos y trucos: CNET también ofrece consejos y trucos para ayudar a los consumidores a aprovechar al máximo sus dispositivos electrónicos de consumo y servicios en línea.

- Videos y podcasts: CNET produce videos y podcasts que brindan información sobre productos electrónicos de consumo y servicios en línea, así como noticias y análisis del mundo de la tecnología.

2.3.- SITIOS DE VENTA Y EVALUACIÓN DE PRODUCTOS

2.3.1.- TRIPADVISOR

TripAdvisor es un sitio web que brinda a los viajeros información sobre destinos turísticos, hoteles, restaurantes y atracciones [7]. Las principales funciones de TripAdvisor son:

- Opiniones y valoraciones: TripAdvisor permite a los usuarios dejar opiniones y valoraciones sobre los hoteles, restaurantes, atracciones y destinos turísticos que han visitado. Estas opiniones y valoraciones se utilizan para clasificar los establecimientos y ayudar a otros viajeros a tomar decisiones informadas.

- Reservas de hoteles y vuelos: TripAdvisor permite a los usuarios buscar y reservar hoteles y vuelos directamente desde el sitio web.

- Guías de viaje: TripAdvisor proporciona guías de viaje detalladas para ciudades y destinos turísticos de todo el mundo. Estas guías incluyen información sobre atracciones, restaurantes, hoteles y cosas que hacer en cada lugar.

- Foros de viaje: TripAdvisor tiene foros de viaje donde los usuarios pueden hacer preguntas y obtener respuestas de otros viajeros y expertos en viajes.

2.3.2.- AMAZON

Amazon es un sitio web de comercio electrónico que ofrece una amplia variedad de productos y servicios a los consumidores [8]. Las principales funciones de Amazon son las siguientes:

- Compras en línea: Amazon permite a los usuarios comprar una amplia variedad de productos en línea, desde electrónica hasta ropa y productos para el hogar.

- Servicios en línea: Amazon ofrece servicios en línea, como Amazon Prime Video y Amazon Music, almacenamiento en la nube y servicios de publicidad en línea.

- Reseñas y valoraciones: Amazon permite a los usuarios dejar reseñas y valoraciones sobre los productos que han comprado. Estas reseñas y valoraciones se utilizan para clasificar los productos y ayudar a otros compradores a tomar decisiones informadas.

- Marketplace: Amazon tiene un marketplace donde los vendedores pueden vender sus productos directamente a los consumidores.

2.4.- COMPARACIÓN Y CONCLUSIÓN

Como se ha podido ver, cada sitio web tiene funcionalidades diferentes, y aunque todos ellos comparten la funcionalidad de poder valorar productos o servicios, todos ellos comparten algunas similitudes que se explican a continuación. IMDb, Rotten Tomatoes, CNET, Consumer Reports, TripAdvisor y Amazon son sitios web muy populares que ofrecen una amplia variedad de servicios y productos a los consumidores. A continuación se presenta una comparativa de estas plataformas en términos de sus similitudes y diferencias generales:

- Contenido: Todos los sitios web ofrecen contenido en línea a los usuarios, pero cada uno se enfoca en un tipo específico de contenido. Por ejemplo, IMDb se centra en películas y programas de televisión, Rotten Tomatoes en críticas de películas, CNET en tecnología, Consumer Reports en revisiones de productos y servicios, TripAdvisor en viajes y turismo, y Amazon en productos y servicios en línea.

- Valoraciones y reseñas: La mayoría de los sitios web permiten a los usuarios dejar valoraciones y reseñas sobre los productos o servicios que han utilizado o adquirido. Sin embargo, cada sitio tiene su propio sistema de valoración y reseñas,

y algunos sitios utilizan la opinión de los usuarios como una parte importante de su modelo de negocio, como es el caso de Amazon y Consumer Reports.

- Funciones de búsqueda: Todos los sitios web tienen funciones de búsqueda para ayudar a los usuarios a encontrar el contenido, producto o servicio que están buscando. Sin embargo, la funcionalidad y la precisión de las funciones de búsqueda pueden variar según el sitio.

- Interacción social: Algunos sitios web, como TripAdvisor y Rotten Tomatoes, ofrecen foros de discusión o comunidades en línea donde los usuarios pueden interactuar y compartir información.

- Compras en línea: Amazon es el único sitio web de esta lista que se centra en las compras en línea. Sin embargo, algunos otros sitios web, como TripAdvisor, permiten a los usuarios reservar hoteles y vuelos directamente desde el sitio web.

- Fuentes de ingresos: Cada sitio web tiene sus propias fuentes de ingresos. Por ejemplo, IMDb gana dinero a través de la publicidad y la suscripción premium, mientras que Amazon gana dinero a través de las ventas en línea y el almacenamiento en la nube.

En resumen, aunque todos estos sitios web ofrecen contenido en línea y funciones de búsqueda, cada uno se enfoca en un tipo específico de contenido y tiene sus propias fuentes de ingresos y modelos de negocio. Sin embargo, los sitios web que permiten valoraciones y reseñas suelen utilizar esta información como una parte importante de su modelo de negocio, mientras que otros sitios web se centran en las compras en línea o en proporcionar información de calidad sobre productos y servicios, pero no hay ningún sitio que utilice las reseñas y valoraciones de productos y servicios generales como función principal de negocio, por lo que el proyecto Evaluómetro cobra sentido, ya que se encargaría de unificar la mayoría de las funciones que ofrecen las webs que hemos comentado, pero con un enfoque más utilitario.

Como conclusión, Evaluómetro trataría de implementar las valoraciones y reseñas, junto con la interacción social como función principal, pero también implementaría funciones de generación de contenido, funciones de búsqueda para que el usuario pueda encontrar rápidamente lo que busca, funciones que permitan promocionar links, de manera que se podrían promocionar compras de productos o servicios que se estén valorando y además, tendría un modelo de negocio basado en anuncios, que mediante IA se podrían ajustar al tema de cada foro, de manera que se podría enfocar mucho mejor el sector de público al que va dirigido el anuncio, haciendo un modelo de negocio viable y gratuito para los usuarios, permitiendo unificar criterios y utilidades que muchos sitios webs comparten, pero sacándole una utilidad y sencillez extra que atraiga a un público más amplio.

En la tabla 2.1, los ticks (✔) indican las funciones que sí cumplen los sitios web, mientras que las cruces (✕) indican las funciones que no. Como se puede ver, cada sitio web tiene un enfoque y una función distintos, y se diferencian principalmente en los temas que cubren, los servicios que ofrecen y los productos que venden.

Tabla 2.1: Comparativa entre dominios web con opciones para evaluar

Sitios Web	Información sobre películas y series	Críticas de cine y televisión	Noticias y reseñas de productos tecnológicos	Pruebas y reseñas de productos de consumo	Información y reseñas de destinos turísticos	Promoción de venta de productos y servicios
IMDB	✔	✔	✕	✕	✕	✔
Rotten Tomatoes	✔	✔	✕	✕	✕	✔
CNET	✕	✕	✔	✔	✕	✔
Consumer Reports	✕	✕	✔	✔	✕	✔
TripAdvisor	✕	✕	✕	✕	✔	✔
Amazon	✕	✕	✕	✔	✕	✔
Evaluómetro	✔	✔	✔	✔	✔	✔

Por estos motivos es por lo que Evaluómetro tiene ventaja respecto a los demás sitios webs, porque como se ve en la tabla, todos los dominios promocionan la compra o venta de productos o servicios, tratando de una manera u otra de ayudar a los usuarios a tomar la mejor decisión a la hora de adquirir dichos productos o servicios, pero dado que su función principal no es esa, lo hacen de manera más limitada, por lo que ahí es donde Evaluómetro sacaría partido al resto de webs. Al tener como función principal la de ayudar a los usuarios a tomar buenas decisiones económicas, ofrece la oportunidad de desconocer totalmente un tema y ser capaz de, con una búsqueda rápida, adquirir la máxima información de utilidad posible, pudiendo entender mejor por qué un viaje, hotel, producto o servicio es mejor que otros.

Capítulo 3 Hipótesis de trabajo

Este proyecto Fin de Grado de ingeniería informática, se centra en la creación e implementación de una aplicación web completa, concebida como una red social. La aplicación en cuestión es un proyecto de desarrollo full-stack, ya que incluye componentes de front-end y back-end. En esencia, comprende responsabilidades de back-end, como la implementación de la lógica empresarial, la manipulación de datos y la arquitectura general de la aplicación, además del diseño front-end (interfaz de usuario e interacción).

Aunque se han utilizado varias tecnologías para hacer que el front-end sea visualmente atractivo y fácil de usar, el proyecto se concentra principalmente en el back-end. Una cantidad significativa de tiempo del proyecto se ha dedicado a desarrollar la lógica comercial, mantener los datos y crear la infraestructura de la aplicación, de esta manera, se ve reflejada la inclinación hacia la programación de back-end por parte del desarrollador.

El patrón de diseño Modelo-Vista-Controlador (MVC) se utiliza en la estructura del proyecto, que utiliza Python y Django. Asimismo, el entorno de desarrollo integrado (IDE)

de Visual Studio Code facilitó todo el proceso de desarrollo. Es crucial enfatizar que todo el software y el hardware utilizados para este estudio académico fue de código abierto y, por tanto, de uso libre y gratuito. Cada una de estas herramientas se ha personalizado especialmente para satisfacer las necesidades concretas de este proyecto.

3.1.- MODELO VISTA CONTROLADOR (MVC)

Modelo-Vista-Controlador (MVC) es un patrón de diseño de software adoptado en este proyecto para organizar y estructurar el código de la aplicación. Este modelo divide las responsabilidades de la aplicación en tres componentes diferentes, cada uno con su rol correspondiente [9].

- Modelo: este componente encapsula los datos y las reglas de negocio de la aplicación. Es responsable de acceder a la base de datos, ejecutar consultas y almacenar y recuperar datos. En otras palabras, el modelo gestiona toda la información y las operaciones sobre ella.

- Vista: la vista es una representación visual de los datos del modelo. Es la interfaz con la que interactúa el usuario. La función de la vista es presentar los datos al usuario, recopilar sus acciones y enviarlas al controlador.

- Controlador: el controlador actúa como intermediario entre la vista y el modelo. Responde a las interacciones del usuario capturadas por la vista y realiza cambios en el modelo según sea necesario. El controlador también solicita actualizaciones de la vista cuando cambian los datos en el modelo.

La principal ventaja del modelo MVC es que admite la mantenibilidad y la extensibilidad de la aplicación al separar la lógica comercial de la interfaz de usuario. También permite a los desarrolladores trabajar simultáneamente en diferentes partes de la aplicación (modelo, vista y controlador) sin interferir entre sí.

3.2.- ENTORNO DE DESARROLLO INTEGRADO (IDE): VISUAL STUDIO CODE

El IDE seleccionado para el desarrollo de este proyecto es Visual Studio Code (VS Code). Es un editor de código fuente creado por Microsoft que incluye herramientas como resaltado de sintaxis, finalización de código inteligente, fragmentos de código, soporte para numerosos lenguajes y plataformas, y una variedad de extensiones para permitir que los

desarrolladores personalicen su flujo de trabajo. Es bien conocido por su interfaz de usuario ordenada y sin complicaciones, que libera a los desarrolladores para que se concentren en su código. Es una opción popular entre los desarrolladores web, ya que admite una amplia variedad de marcos y lenguajes de programación y ofrece una integración efectiva con sistemas de control de versiones como Git [10].

La versatilidad, adaptabilidad y efectividad de VS Code hizo que fuera el mejor IDE para este proyecto porque simplifica mucho la creación y depuración de la aplicación. Además, ofrece una gran cantidad de extensiones y complementos que facilitan el trabajo del desarrollador. Las principales extensiones de VS Code utilizadas en este proyecto se describen brevemente en las secciones a continuación:

3.2.1.- DJANGO

El marco Django es compatible con VS Code a través de esta extensión. Django es un marco de desarrollo web basado en Python de alto nivel que fomenta la iteración rápida y un diseño práctico y sencillo.

El complemento de Django mejora la productividad del desarrollo y la legibilidad del código al proporcionar resaltado de sintaxis, fragmentos de código y soporte para plantillas de Django [11].

3.2.2.- GITHUB PULL REQUESTS AND ISSUES

Con la ayuda de esta extensión, los desarrolladores pueden controlar las solicitudes de extracción y los problemas de GitHub directamente desde Visual Studio Code. Tiene capacidades como código en línea, observar cómo navegar por los problemas y las solicitudes de extracción, confirmar cambios y enviarlos directamente desde el editor.

Este complemento agiliza el proceso de control de versiones y es especialmente útil para proyectos alojados en GitHub. [12].

3.2.3.- PYTHON

El complemento de Python para VS Code ofrece una funcionalidad mejorada de este lenguaje de programación. Tiene capacidades para ejecutar pruebas unitarias, así como IntelliSense, linting, formato de código, depuración y navegación de código. Este

complemento ha demostrado ser esencial para aumentar la productividad y elevar el calibre del código para este proyecto, el cual ha sido desarrollado con Python y Django. [13].

3.2.4.- SQLITE3 EDITOR

SQLite es un sistema de gestión de bases de datos que se utiliza con frecuencia en el desarrollo web. La extensión SQLite3 Editor para VS Code proporciona una interfaz gráfica fácil de usar para interactuar con bases de datos SQLite. Esta herramienta permite crear, modificar y eliminar tablas de forma rápida e intuitiva, así como insertar, actualizar y eliminar registros. Esta extensión ha sido fundamental en la gestión de la base de datos creada por Django [14].

3.3.- BACK-END

El componente back-end de la aplicación es responsable de la lógica de negocio, el procesamiento de datos, el acceso a la base de datos y la integración con otros sistemas o servicios. Esta capa de la aplicación no suele ser vista por los usuarios finales, pero es fundamental para el correcto funcionamiento de la red social. En el siguiente apartado se describen las diferentes tecnologías back-end empleadas en el desarrollo del proyecto:

3.3.1.- PYTHON

Python, un lenguaje de programación interpretado, interactivo y orientado a objetos, fue desarrollado por Guido van Rossum en los años 80 y desde entonces se ha convertido en uno de los lenguajes de programación más populares [15]. Algunas de las principales características de Python son:

- Lenguaje de alto nivel: Python es un lenguaje de alto nivel, es decir, más cercano al lenguaje humano que al lenguaje máquina, lo que facilita su lectura, escritura y mantenimiento.

- Interpretado: Python es un lenguaje interpretado, lo que significa que el código se ejecuta línea a línea, lo que es beneficioso para el desarrollo y la depuración, pero puede ser más lento que los lenguajes compilados.

- Multiparadigma: Python soporta múltiples paradigmas de programación, incluyendo programación orientada a objetos, imperativa y, en menor medida, funcional.

- Tipado dinámico: Python es de tipado dinámico, lo que significa que no es necesario declarar los tipos de variables antes de utilizarlas. Python determina automáticamente los tipos de datos durante la ejecución del programa.

- Portabilidad: Python es portable, con intérpretes disponibles para muchos sistemas operativos, lo que permite ejecutar programas Python en cualquiera de ellos.

- Potentes bibliotecas estándar: Python cuenta con una completa biblioteca estándar disponible sin coste alguno, que incluye bibliotecas para el manejo de archivos, navegación por Internet, acceso a bases de datos, pruebas unitarias, etc.

- Compatibilidad con diferentes plataformas y sistemas: Python es compatible con varios sistemas operativos, incluyendo Windows, MacOS, Linux, Unix, entre otros.

Este lenguaje también cuenta con un vasto ecosistema de librerías y frameworks, lo que lo convierte en una opción robusta para el desarrollo web.

3.3.2.- DJANGO

Django es un framework de desarrollo web basado en Python que sigue el patrón de diseño Modelo Vista Controlador (MVC). Este patrón ayuda a segregar la lógica de negocio, la interfaz de usuario y el acceso a los datos, facilitando la escalabilidad y el mantenimiento de la aplicación. Django ofrece numerosas funcionalidades integradas, incluyendo un sistema de plantillas, un Object-Relational Mapper (ORM) para la interacción con bases de datos, y soporte para la autenticación de usuarios y la gestión de contenidos. Algunas de sus principales características son:

- Escrito en Python: Al estar escrito en Python, Django se beneficia de la simplicidad y legibilidad del lenguaje.

- Patrón MVC: Django sigue el patrón de diseño MVC, que promueve la separación de la lógica de negocio, la interfaz de usuario y el acceso a los datos, facilitando el mantenimiento y la escalabilidad de la aplicación.

- DRY (No te repitas): Django se adhiere al principio DRY, centrándose en minimizar la duplicación de código y promoviendo la reutilización.

- ORM (Mapeo Objeto-Relacional): Django viene con un ORM incorporado que simplifica la interacción con la base de datos. Los desarrolladores pueden trabajar con bases de datos utilizando el lenguaje Python, sin escribir consultas SQL.

- Seguridad: Django ayuda a los desarrolladores a evitar muchos errores de seguridad comunes proporcionando un sistema de autenticación de usuarios y protección contra ataques como Cross-site Scripting (XSS), Cross-site Request Forgery (CSRF) y SQL Injection.

- Sistema de plantillas: Django viene con un potente y versátil sistema de plantillas, simplificando la generación de HTML dinámico.

- Compatibilidad con middleware: Django soporta el uso de middleware, que son componentes de software que pueden engancharse al proceso de petición/respuesta para procesar peticiones y respuestas antes de que lleguen al usuario o a la vista.

- Administración de aplicaciones: Django incluye una interfaz de administración generada automáticamente que se puede utilizar para gestionar aplicaciones Django.

Esto ha facilitado la aceleración del desarrollo de proyectos y el enfoque en aplicaciones específicas [16].

3.3.3.- DB SQLITE3

SQLite3 es un sistema compacto de gestión de bases de datos SQL conocido por su alta fiabilidad y su completa funcionalidad. Integrado como una biblioteca C, SQLite3 ofrece un motor de base de datos ligero que no necesita un servidor independiente, por lo que es una opción óptima para proyectos de desarrollo web más pequeños como éste y para fines de prueba.

SQLite3 es el sistema de gestión de bases de datos por defecto configurado por Django, simplificando la tarea de almacenar y recuperar información de usuarios y publicaciones en redes sociales de forma eficiente. Es importante destacar que SQLite3 se adhiere a los principios ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), asegurando la integridad de los datos incluso en caso de fallos del sistema o del programa [17].

3.4.- FRONT-END

La interfaz de usuario de una aplicación, a menudo conocida como front-end, interactúa directamente con el usuario proporcionando una interfaz gráfica de usuario para la navegación y la interacción. El front-end es un componente clave de este proyecto ya que,

para mantener el compromiso del usuario, las plataformas de redes sociales deben ser visualmente atractivas, intuitivas y fáciles de usar. Cabe destacar que, aunque el front-end es necesario, el proyecto no se ha centrado principalmente en él debido a que el desarrollador presenta preferencias por el back-end, como ya se ha comentado anteriormente.

Las tecnologías front-end utilizadas en este proyecto se describen en los siguientes apartados:

3.4.1.- HTML

Un lenguaje de marcado habitual para crear páginas web es HTML (HyperText Markup Language). El World Wide Web Consortium (W3C) lanzó en octubre de 2014 HTML5, la versión más utilizada del invento de Tim Berners-Lee de 1991 [18].

A continuación se ofrece un esbozo de sus principales rasgos:

- Lenguaje de marcado: HTML emplea etiquetas para distinguir entre varios tipos de texto, lo que facilita a los navegadores web la comprensión y visualización del contenido.

- Estructura web: HTML esboza características como encabezados, párrafos, listas, enlaces, fotos, formularios, etc. y proporciona el marco fundamental para las páginas y aplicaciones web.

- Semántica: Las etiquetas HTML5 también aportan significado semántico, mejorando la accesibilidad y los resultados SEO. Estas etiquetas incluyen elementos como "artículo", "sección", "nav", "encabezado" y "pie", entre otros.

- Multimedia: HTML5 ha añadido etiquetas para contenidos multimedia, como "vídeo", "audio", "canvas" y "svg", que permiten integrar y gestionar directamente este tipo de contenidos en las páginas web.

- Aunque JavaScript se utiliza para crear la mayoría de los elementos interactivos de las páginas web, HTML5 también cuenta con elementos interactivos como "detalles", "resumen", "lista de datos" y "salida".

- Integración con CSS y JavaScript: JavaScript se utiliza junto con HTML para añadir interactividad y funcionalidad dinámica. CSS se utiliza para aplicar estilos.

- Todos los navegadores actuales son compatibles con HTML, lo que permite a la mayoría de los internautas acceder a sitios en línea.

- Curva de aprendizaje: Incluso para los principiantes en el desarrollo web, la sintaxis de HTML es razonablemente sencilla y fácil de aprender, a pesar de que cuenta con un gran número de etiquetas y propiedades.

Todas las páginas web de este proyecto se construyeron con HTML, lo que permitió estructurar el contenido de forma lógica y fácil de usar.

3.4.2.- CSS

Para aplicar y gestionar el estilo de la aplicación web se utilizó el lenguaje de hojas de estilo en cascada (CSS). Esto abarca aspectos como la disposición de los elementos, los colores, las fuentes, los tamaños y otros. [19]

Algunas de sus principales características son:

- Definición: CSS es un lenguaje de hojas de estilo utilizado para describir el aspecto o presentación de un documento escrito en HTML o XML. Es una de las tecnologías clave utilizadas en la web, junto con HTML y JavaScript.

- Estilo: CSS permite dar estilo a los elementos HTML definiendo características como colores, fuentes, tamaños, márgenes y alineación, entre otras.

- Separación entre contenido y diseño: CSS ayuda a separar el contenido (HTML) del diseño, mejorando la accesibilidad y flexibilidad del diseño.

- Hojas de estilo en cascada: El término "cascada" se refiere a la prioridad que se da a ciertas reglas de estilo sobre otras. Esto permite a diseñadores y desarrolladores controlar con precisión cómo se aplica el estilo.

- Selectores: Los selectores CSS permiten aplicar estilos a elementos específicos en función de su nombre, id, clase, atributos, estado o posición en la estructura del documento HTML.

- Capacidad de respuesta: CSS facilita la creación de diseños responsivos, que se adaptan automáticamente al tamaño y orientación de la pantalla del usuario.

- Animaciones y transiciones: CSS3, la última versión de CSS, incluye soporte para animaciones y transiciones, lo que permite crear interfaces de usuario dinámicas sin necesidad de JavaScript.

- Flexbox y Grid: CSS proporciona dos módulos, Flexbox y Grid, que facilitan la creación de diseños complejos y con capacidad de respuesta.

En situaciones en las que Tailwind no era suficiente, CSS proporcionaba un mayor control sobre los estilos, permitiendo la aplicación de diseños más complejos.

3.4.3.- TAILWIND

Tailwind es un framework CSS de utilidad que ofrece un enfoque más flexible y eficiente para el diseño web. A diferencia de otros frameworks CSS, Tailwind permite un alto grado de personalización sin desviarse de su propia sintaxis.

Casi toda la web se estiliza mediante Tailwind, lo que facilita la creación de un diseño personalizado y coherente en toda la aplicación. Cabe mencionar que a pesar de su flexibilidad, en parte se debe al limitado conocimiento del creador sobre este framework, la implementación de CSS fue necesaria para cosas más complejas, como se mencionó anteriormente. [20]

3.4.4.- JAVASCRIPT

JavaScript es un lenguaje de programación de alto nivel, dinámico e interpretado. Fue desarrollado originalmente para mejorar la interactividad y funcionalidad de las páginas web. A lo largo de los años, ha evolucionado y se ha expandido a otras áreas, incluyendo el desarrollo de servidores y aplicaciones móviles [21]. Algunas de las principales características de JavaScript son:

- Interpretado: JavaScript es un lenguaje interpretado, lo que significa que el código es ejecutado línea a línea por un intérprete. No es necesario compilar el código antes de su ejecución.

- Dinámico: JavaScript es un lenguaje dinámico. Esto significa que los tipos de datos y las estructuras pueden modificarse en tiempo de ejecución.

- Tipado débil: JavaScript es un lenguaje de tipado débil, lo que significa que no es necesario declarar explícitamente el tipo de datos de una variable. El tipo de datos de una variable puede cambiar durante la ejecución del código.

- Orientado a objetos: JavaScript admite la programación orientada a objetos. Los objetos en JavaScript son entidades que tienen propiedades y comportamientos. JavaScript utiliza un modelo de objetos basado en prototipos en lugar de clases.

- Controlado por eventos: JavaScript es un lenguaje de programación orientado a eventos. Esto significa que permite crear funcionalidades que responden a las acciones del usuario, como pulsar un botón o mover el ratón.

- Compatibilidad con navegadores: Casi todos los navegadores modernos son compatibles con JavaScript. Esto lo hace universalmente aplicable para el desarrollo web.

- Funcionalidad del lado del cliente y del lado del servidor: Aunque originalmente se diseñó para ejecutarse en el navegador (client-side), JavaScript también puede ejecutarse en el servidor gracias a entornos como Node.js.

En este proyecto, JavaScript ha permitido crear funciones dinámicas de cliente, como mostrar u ocultar elementos, cambiar el contenido de la página sin recargarla o validar formularios, entre otras.

3.4.5.- AJAX

AJAX (Asynchronous JavaScript And XML) es una técnica de desarrollo web que permite a las páginas web enviar y recibir datos de un servidor de forma asíncrona, sin interrumpir el comportamiento de la página. Esto significa que una página web puede actualizar parte de su contenido sin tener que recargar toda la página. AJAX combina varias tecnologías, como JavaScript, XML, HTML y CSS. [22, 23]

Las principales características de AJAX son:

- Asíncrono: AJAX permite que la página web solicite datos al servidor y continúe trabajando en otras tareas mientras espera la respuesta. Esto mejora la eficiencia y la experiencia del usuario.

- Interactividad: AJAX puede actualizar el contenido de una página web en respuesta a las acciones del usuario, lo que aumenta la interactividad de la página.

- Basado en estándares: AJAX utiliza tecnologías web estándar como JavaScript y XML, por lo que es compatible con una amplia gama de plataformas y navegadores.

- Sin recarga de la página: Con AJAX, se pueden cargar nuevos datos en la página sin necesidad de volver a recargarla por completo, lo que puede mejorar la velocidad y la experiencia del usuario.

- Comunicación con el servidor: AJAX puede enviar datos al servidor y recibir datos de él, lo que permite a la página web interactuar con bases de datos y otros recursos del servidor.

En este proyecto, AJAX se ha utilizado para cargar dinámicamente los posts a medida que el usuario se desplaza por la página principal, mejorando la eficiencia y la experiencia de usuario.

3.4.6.- JQUERY

jQuery es una librería JavaScript rápida, pequeña y rica en funciones. Fue diseñada para simplificar la forma de interactuar con páginas web HTML utilizando JavaScript. Las principales características de jQuery incluyen: [24]

- Manipulación del Modelo de Objetos del Documento (DOM): jQuery facilita la manipulación del DOM, permitiendo a los desarrolladores cambiar el contenido, la estructura y el estilo de su página web con facilidad.

- Manejo de eventos: jQuery proporciona una interfaz elegante para capturar y manejar eventos de usuario, como clics, movimientos del ratón, cambios de formulario, etc.

- Animación y efectos: jQuery incorpora varios métodos para aplicar efectos animados a los elementos del DOM. Esto incluye deslizar, desvanecer, ocultar/mostrar y otros efectos personalizados.

- AJAX: jQuery simplifica enormemente el uso de AJAX, permitiendo a los desarrolladores cargar datos en segundo plano, actualizar partes de la página web y enviar formularios sin tener que recargar toda la página.

- Compatibilidad con navegadores: jQuery gestiona muchas de las incoherencias entre los distintos navegadores y sus versiones, lo que permite a los desarrolladores escribir código que funciona de manera uniforme para todos ellos.

- Extensibilidad: jQuery puede ampliarse con plugins, lo que permite a los desarrolladores crear piezas de código reutilizables que añaden funcionalidades adicionales a la biblioteca.

En este proyecto, JQuery se ha utilizado para simplificar el uso de AJAX y otros métodos de JavaScript, permitiendo un código más limpio y legible.

La combinación de estas tecnologías ha permitido crear una aplicación web completa, con una experiencia de usuario fluida y una gestión eficaz de los datos y la lógica empresarial.

Capítulo 4 Metodología y resultados

4.1.- PLANIFICACIÓN DEL PROYECTO

El desarrollo de software es un proceso complejo que requiere una planificación y una estrategia cuidadosa. El modelo de ciclo de vida iterativo e incremental destaca por ser especialmente eficaz entre los numerosos modelos de desarrollo de software.

Este Trabajo Fin de Grado (TFG), trata sobre una red social centrada en la evaluación de Posts basada en Django y Python, que se ha beneficiado enormemente del uso de esta metodología.

4.1.1.- CICLO DE VIDA ITERATIVO

El paradigma del ciclo de vida iterativo e incremental hace mucho hincapié en la creación rápida y fiable de incrementos de software funcionales. Este modelo plantea una sucesión

de ciclos iterativos para construir y distribuir software. Cada ciclo termina con una pieza de software funcional que puede probarse, evaluarse y mejorarse en ciclos sucesivos. Este método tiene la ventaja de poder identificar problemas o fallos en una fase temprana del proceso de desarrollo, lo que reduce el riesgo y el coste del fracaso. Además, como los requisitos pueden cambiar con el tiempo, el equipo de desarrollo sigue teniendo la capacidad de ajustarse a las necesidades cambiantes del cliente o del mercado. En el siguiente diagrama se muestra una representación visual de este modelo:

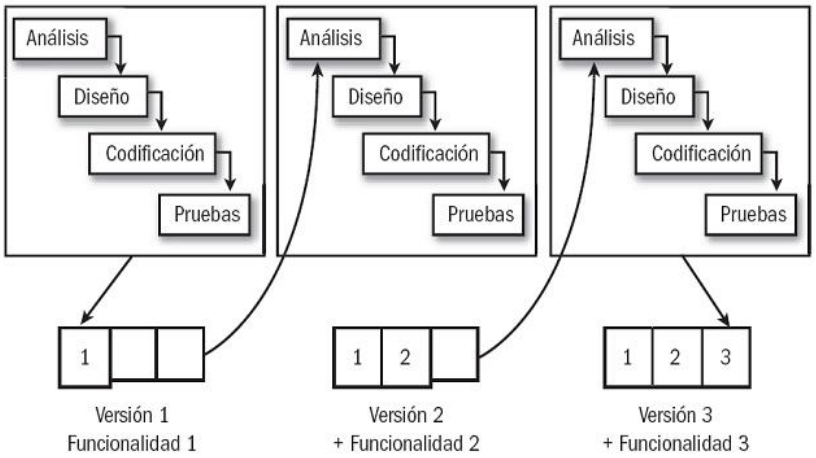

Figura 4.1. - Ciclo de vida iterativo [25]

4.1.2.- DIAGRAMA DE GANTT

El análisis de requisitos, el diseño, la implementación y el mantenimiento son las partes fundamentales del proceso de planificación de un proyecto.

- Análisis de requisitos: Esta fase presentó más incidencias de lo previsto. Uno de los objetivos principales del proyecto fue adquirir conocimientos sobre Python y Django, por ello, esta aplicación web se llevó a cabo aún sin conocimientos previos sobre estas tecnologías (ya que estos contenidos no se imparten en el grado). Dadas estas circunstancias, se dedicó una gran cantidad de estudio inicialmente, algo fundamental para entender qué necesitaba el proyecto. Durante este proceso fue recopilada información como por ejemplo, cómo construir el proyecto, cómo utilizar Django para aplicar la arquitectura Modelo-Vista-Controlador, y cómo combinar las ideas que se tenían en mente, utilizando las tecnologías que se habían elegido inicialmente y las que fueron añadidas más tarde.

- Diseño: Dado que el diseño de la web no es de principal interés en el proyecto, este se tornó como el proceso más tedioso de realizar. A pesar de ello, se tuvo que dedicar mucho tiempo al diseño puesto que, una red social tiene que ser fácilmente navegable y atractiva a la vista. Haciendo hincapié en la facilidad de uso para los

nuevos usuarios de la red social, se optó por un diseño sencillo y atractivo para el contenido y los botones del sitio web.

- Implementación: Esta fase consiste en poner en práctica toda la información que fue aprendida en la investigación. Dada la falta de experiencia del desarrollador, surgieron dificultades y problemas imprevistos. Con el tiempo, se logró configurar el IDE de Visual Studio Code, la base de datos necesaria, junto con su implementación a partir de los diseños realizados, y una distribución productiva del patrón Modelo-Vista-Controlador. Además, se crearon un conjunto de microservicios y funcionalidades, los cuales fueron posibles de implementar gracias a los conocimientos adquiridos a medida que se desarrolló el proyecto.

- Mantenimiento: Es la fase más importante actualmente. Evaluómetro tiene el potencial de ser una herramienta útil, aunque por ahora sólo sea funcionalmente básica. En particular, una herramienta no tiene por qué ser sofisticada o novedosa para ser valiosa. La finalidad de esta etapa consiste en mantener los servicios ya implementados, al mismo tiempo que se introducen otros nuevos para mejorar la funcionalidad e intuitividad de la web.

Tabla 4.1. - Fases y tareas del proyecto.

Fase	Tarea	Duración	Dependencia
Análisis de Requisitos (F1)	Investigación de Implementación del proyecto	3 semanas	-
	Definición de Funcionalidades y afinamiento de requisitos	1 semana	Investigación de Implementación del proyecto
Diseño (F2)	Diseño de Interfaz	2 semanas	Definición de Funcionalidades y afinamiento de requisitos
	Diseño de Base de Datos	1 semana	Definición de Funcionalidades y afinamiento de requisitos
Implementación (F3)	Configuración del Entorno de Desarrollo (IDE) y de la estructura del proyecto.	1 semana	Diseño de Interfaz, Diseño de Base de Datos
	Desarrollo de Funcionalidades	6 semanas	Configuración del Entorno de Desarrollo (IDE) y de la estructura del proyecto.
Pruebas (F4)	Pruebas Unitarias	1 semanas	Definición de Funcionalidades y afinamiento de requisitos
	Pruebas de Integración	1 semanas	Pruebas Unitarias
	Pruebas de Sistema	1 semanas	Pruebas de Integración
Mantenimiento (F5)	Soporte y Mantenimiento	2 semanas	Comprobación de funcionalidades ya implementadas y desarrollo de nuevas.

Cabe destacar que la medición de los tiempos se ha hecho por semanas, dado que muchos pasos han sido duraderos, pero no todos los pasos fueron exactamente de 1 semana.

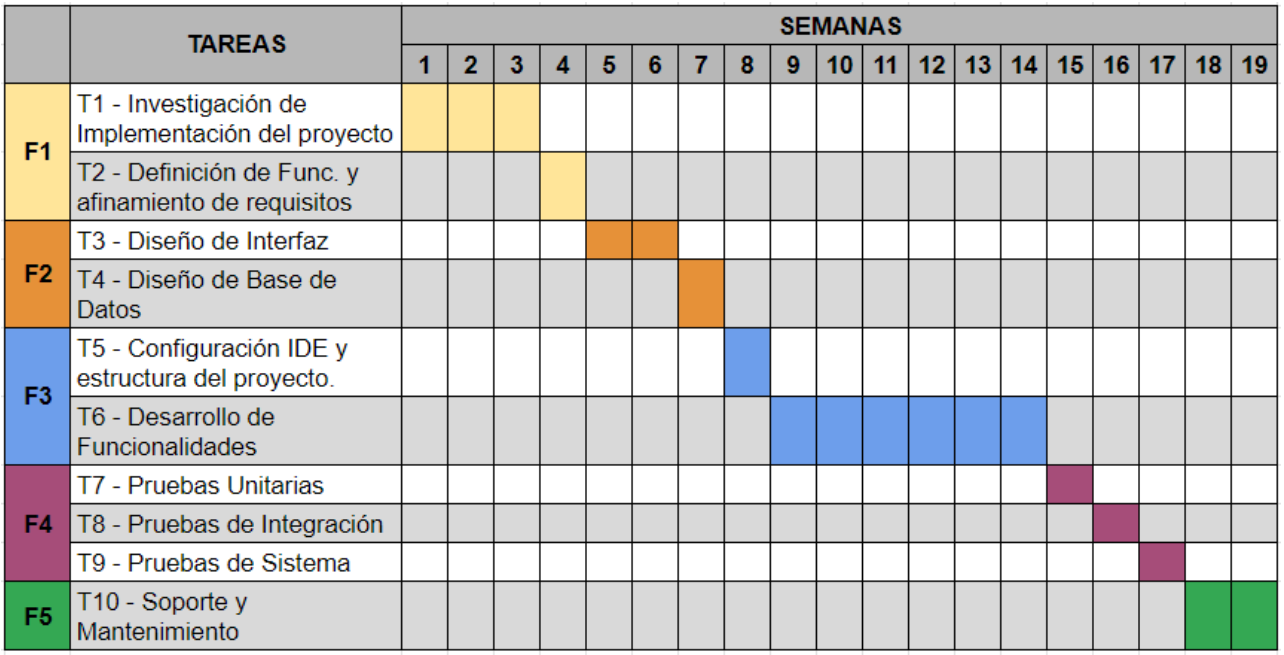

Figura 4.2.- Diagrama de Gantt del proyecto

4.2.- Captura de requisitos

4.2.1.- REQUISITOS FUNCIONALES DE LA APLICACIÓN

Los requisitos funcionales son operaciones que el usuario puede desempeñar dentro de una aplicación. A continuación vamos a ver los requisitos funcionales del presente proyecto:

Tabla 4.2. - Requisitos funcionales. Registro usuario.

RF-1	Registro Usuario
Descripción	El usuario debe poder darse de alta en la red social introduciendo sus datos.

Tabla 4.3. - Requisitos funcionales. Inicio Sesión Usuario.

RF-2	Inicio Sesión Usuario
Descripción	El usuario debe poder iniciar sesión con su cuenta en la red social introduciendo sus datos.

Tabla 4.4. - Requisitos funcionales. Cierre Sesión Usuario.

RF-3	Cierre Sesión Usuario
Descripción	El usuario debe poder cerrar sesión con su cuenta en la red social.

Tabla 4.5. - Requisitos funcionales. Creación de posts.

RF-4	Creación de posts
Descripción	El usuario debe poder crear posts para evaluar cosas en la red social.

Tabla 4.6. - Requisitos funcionales. Comentar y evaluar posts.

RF-5	Comentar y evaluar posts
Descripción	El usuario debe poder comentar y evaluar posts existentes en la red social.

Tabla 4.7. - Requisitos funcionales. Resolver dudas a los usuarios.

RF-6	Resolver dudas a los usuarios
Descripción	Los usuarios podrán resolverse dudas entre ellos en los comentarios de los posts de la red social.

Tabla 4.8. - Requisitos funcionales. Búsqueda de posts.

RF-7	Búsqueda de posts
Descripción	El usuario podrá buscar posts en la red social.

Tabla 4.9. - Requisitos funcionales. Búsqueda de usuarios.

RF-8	Búsqueda de usuarios
Descripción	El usuario podrá buscar otros usuarios.

4.2.2.- ROLES DE USUARIO

En Evaluómetro se distinguen dos tipos de usuarios: el usuario administrador y el registrado. El diseño y las características de estos tipos de usuario han ido cambiando con el tiempo. Sin embargo, el principal objetivo del proyecto, que ha sido construir una red social sin más filtros que los criterios morales y la opinión de los propios usuarios, se ha mantenido constante.

Se busca garantizar que las opiniones de los usuarios reflejen fielmente lo que piensan, por lo que se decidió que nadie pudiera acceder a Evaluómetro sin haber iniciado sesión previamente. Esto crea una barrera de registro inicial, que impide que un usuario sin cuenta pueda acceder a la web, evitando así situaciones recurrentes como cuando cualquier entidad interacciona en una web sin tener cuenta de esta, y dicha plataforma le sobresalta con una notificación de registro. Al poseer cuenta previamente, se le facilitará la participación al usuario.

Como resultado, se decidió eliminar al que hubiese sido el tercer usuario (usuario no registrado), ya que de esta manera se asegura que Evaluómetro tenga una tasa de participación más alta y realista, dejando al sitio web con los dos tipos de usuarios mencionados anteriormente:

- El usuario principal es el usuario registrado. Este aporta material y crea un entorno dinámico e interactivo que fomenta el diálogo y la retroalimentación de contenido con el resto de usuarios.

- El administrador tiene como propósito principal encargarse de los casos éticos o morales, es decir, será el encargado de gestionar (crear, editar y eliminar) la web con una interfaz amigable, sin tener que ajustar código de la aplicación para lograr algo. Esto significa que podrá administrar los blogs, usuarios, comentarios y evaluaciones en caso de que sea necesario por algún motivo de grado mayor, como por ejemplo, un post con expresiones racistas. Este usuario también puede desempeñar un papel comercial, como planificar promociones para respaldar la web en caso de que se convierta en un negocio. Los posts comerciales sólo se utilizarán para publicitar un bien o servicio, no para influir en las opiniones de las personas. Por ejemplo, los usuarios siempre tendrán la libertad de expresar su desaprobación ante un post promocional si así lo desean.

En las tablas 4.10 y 4.11 se describen esquemáticamente ambos usuarios, enumerando los casos de uso correspondientes a cada uno de ellos, que se especificarán más adelante en esta memoria.

Tabla 4.10. - Roles de usuario. Administrador.

Usuario	Administrador
Descripción	El administrador es capaz de manejar cualquier tabla de la base de datos. Puede editar, añadir y borrar cualquier cosa de ellas.
Casos de uso	CU-1, CU-2, CU-3, CU-4, CU-5, CU-6, CU-7, CU-8, CU-9, CU-10, CU-11, CU-12, CU-13, CU-14, CU-15, CU-16, CU-17, CU-18, CU-19, CU-20, CU-31, CU-32, CU-33

Tabla 4.11. - Roles de usuario. Usuario registrado.

Usuario	Usuario registrado
Descripción	El usuario registrado es el que ha iniciado sesión. Puede visualizar, crear, evaluar, seguir y comentar posts.
Casos de uso	CU-1, CU-2, CU-3, CU-4, CU-21, CU-22, CU-23, CU-24, CU-25, CU-26, CU-27, CU-28, CU-29, CU-30, CU-31, CU-32, CU-33, CU-34

Igualmente, las figuras 4.3 y 4.4 muestran los diagramas de casos de uso de cada usuario, con su identificador y denominación.

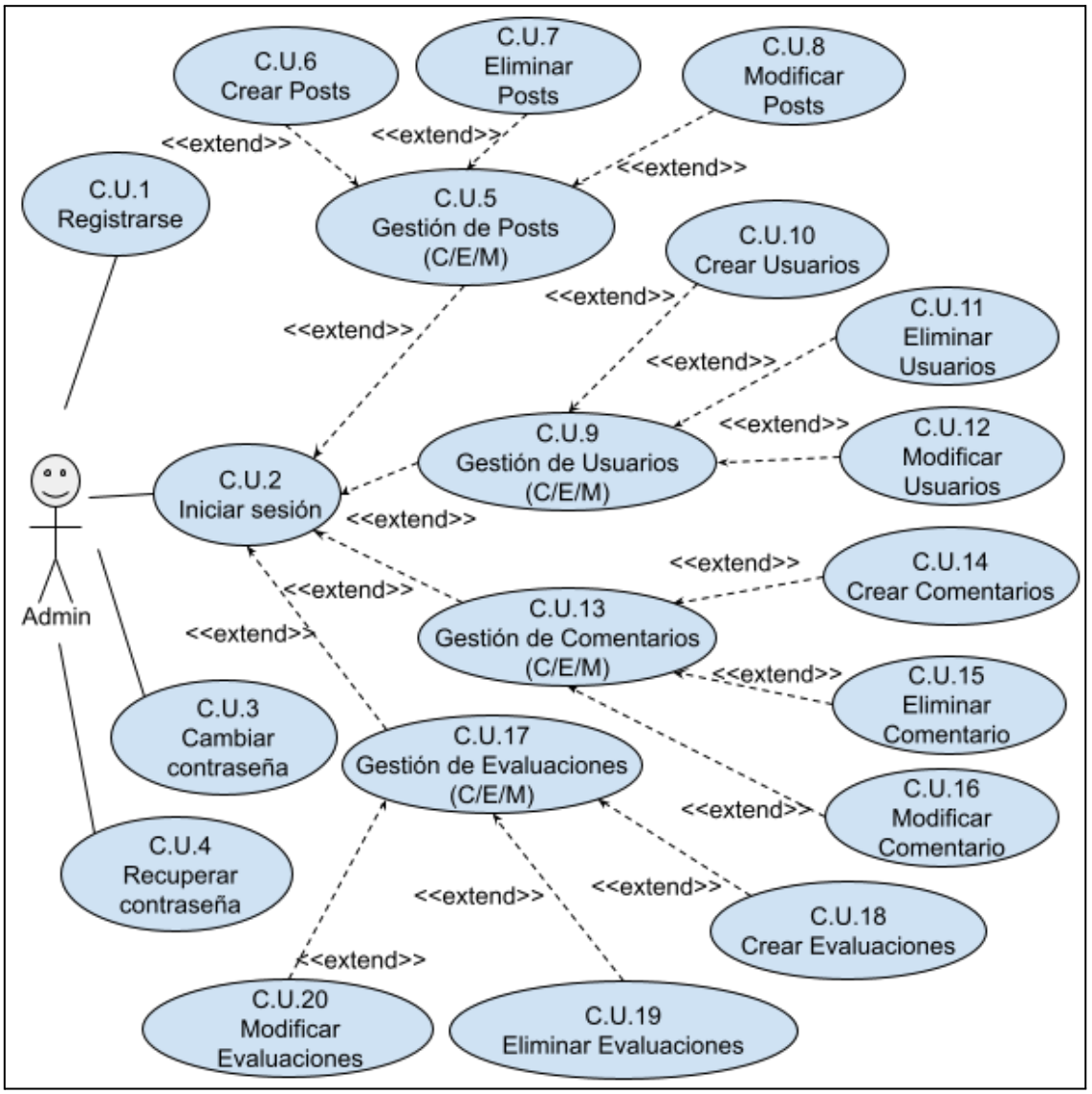

Figura 4.3. - Roles de usuario. Diagrama de casos de uso de Admin.

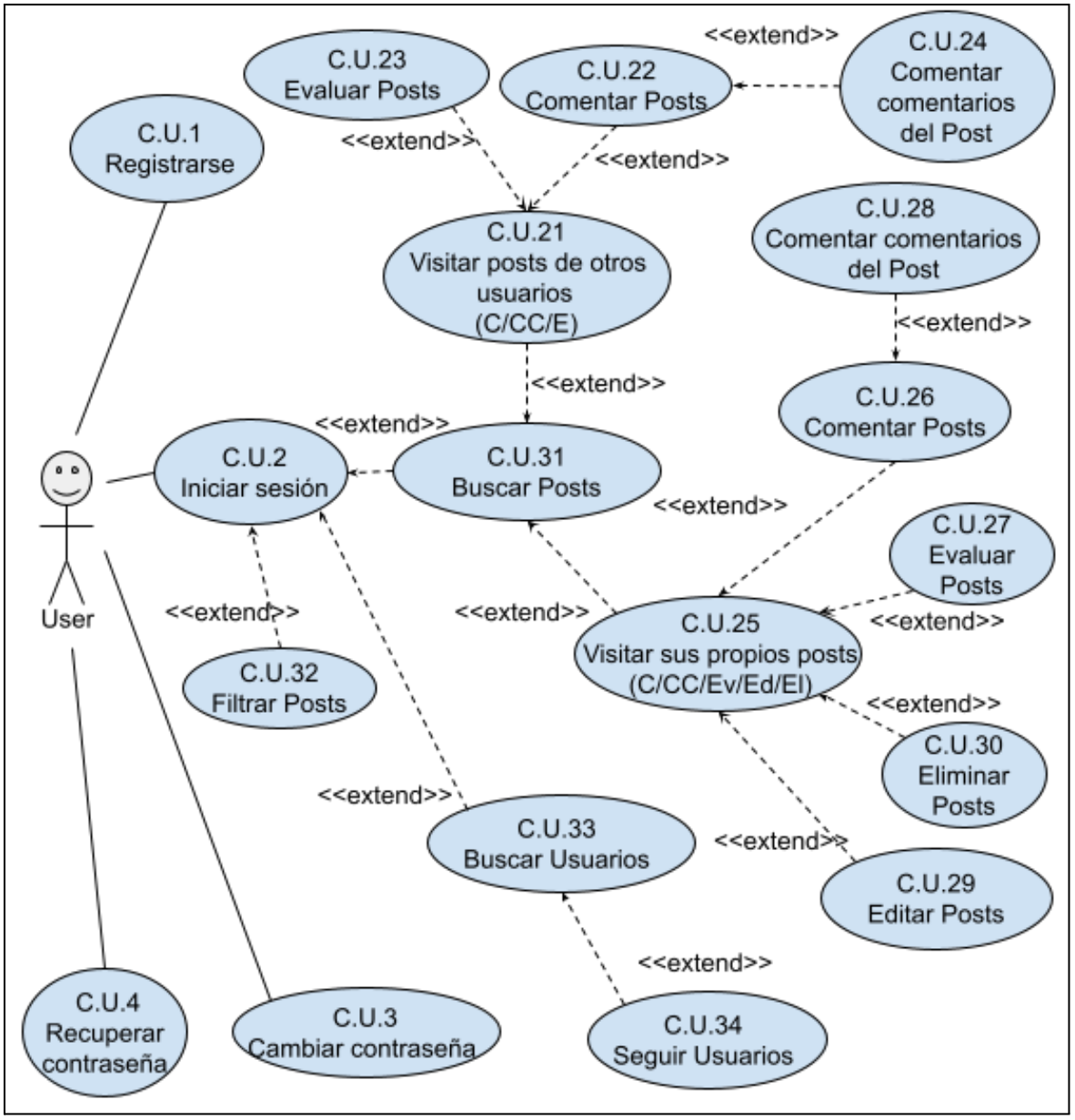

Figura 4.4. - Roles de usuario. Diagrama de casos de uso de Usuario.

4.2.3.- ESPECIFICACIÓN DE CASOS DE USO

En el ámbito de la ingeniería de software, el concepto de casos de uso se emplea con frecuencia para definir el comportamiento esperado de un sistema en respuesta a interacciones específicas del usuario o del sistema. Los casos de uso reflejan el proceso a través del cual un sistema interactúa con sus actores (generalmente usuarios u otros sistemas) para lograr objetivos específicos.

Son una herramienta vital para comprender el funcionamiento de un sistema desde la perspectiva del usuario y se utilizan principalmente en la fase de análisis de los requisitos

del sistema. Las filas de las tablas que se proporcionan constituyen elementos típicos de un caso de uso, que contribuyen a una comprensión detallada de lo que expresa cada uno:

- Actores: Se refiere a las entidades que interactúan con el sistema, incluidos usuarios como administradores o registrados, u otros sistemas.

- Descripción: Ofrece un breve resumen de la finalidad del caso de uso y el objetivo que pretende alcanzar.

- Dependencias: Este segmento aclara si el caso de uso depende de otros casos de uso, o si otros casos de uso deben completarse antes que éste.

- Precondición: Son las condiciones que deben cumplirse antes de la ejecución del caso de uso.

- Secuencia normal: En esta sección se exponen los pasos que suelen suceder cuando el caso de uso se ejecuta en condiciones normales, representando así el flujo de trabajo principal del caso de uso.

- Postcondición: Estas condiciones deben cumplirse tras la ejecución satisfactoria del caso de uso.

- Excepciones: Esta parte describe la respuesta del sistema en caso de error o condición inesperada durante la ejecución del caso de uso.

- Rendimiento: Este parámetro se refiere a los estándares de rendimiento previstos del caso de uso, en relación con factores como la velocidad, la eficiencia, etc.

- Frecuencia: Este apartado denota la frecuencia prevista de ocurrencia del caso de uso, lo que puede ayudar en la priorización de casos de uso y en la planificación de la asignación de recursos.

- Importancia: Esta sección describe la importancia del caso de uso para alcanzar los objetivos del sistema.

- Urgencia: Para indicar el nivel de prioridad para la implementación del caso de uso.

- Estado: Refleja el estado actual del caso de uso en la cadena de desarrollo, ya sea en diseño, desarrollo, pruebas, finalizado, etc.

- Estabilidad: Indica el grado de estabilidad del caso de uso frente a los cambios. Un caso de uso altamente estable es probable que sufra cambios mínimos a lo largo del

tiempo, mientras que uno con baja estabilidad puede experimentar numerosas modificaciones.

- Comentarios: Esta sección proporciona un espacio para la inclusión de cualquier información pertinente o notas que no encajan claramente en las categorías anteriores, pero son vitales para el caso de uso.

Estas secciones facilitan la comprensión global de la forma en que un sistema debe interactuar con sus usuarios y los resultados que se espera que ofrezca en cada momento. En la tabla 4.12 se muestra a modo de ejemplo el detalle de como quedaría definido el caso de uso 5, Gestionar posts, el resto se pueden encontrar en el anexo I de esta memoria.

Tabla 4.12. - Casos de uso. Gestionar posts.

C.U. 5	**Gestionar posts**
Actores	Administrador (Principal)
Descripción	El administrador gestiona los posts en la red social.
Dependencias	C.U. 2
Precondición	El administrador ha iniciado sesión en el sistema.
Secuencia normal	P1 - El administrador selecciona la opción "Gestionar posts". P2 - El sistema muestra una lista de todos los posts. P3 - El administrador puede seleccionar un post para ver detalles o realizar operaciones (crear, modificar, eliminar).
Poscondición	Posts gestionados según las operaciones seleccionadas.
Excepciones	- Si en el paso P2 no hay posts, el sistema muestra un mensaje de "No hay posts disponibles". - Si en el paso P3 el administrador selecciona un post que ya ha sido eliminado, el sistema muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

4.3.- DISEÑO

4.3.1.- LA BASE DE DATOS

Esta descripción define el modelo entidad-relación de la base de datos, que incluye numerosas tablas y cómo se relacionan entre sí.

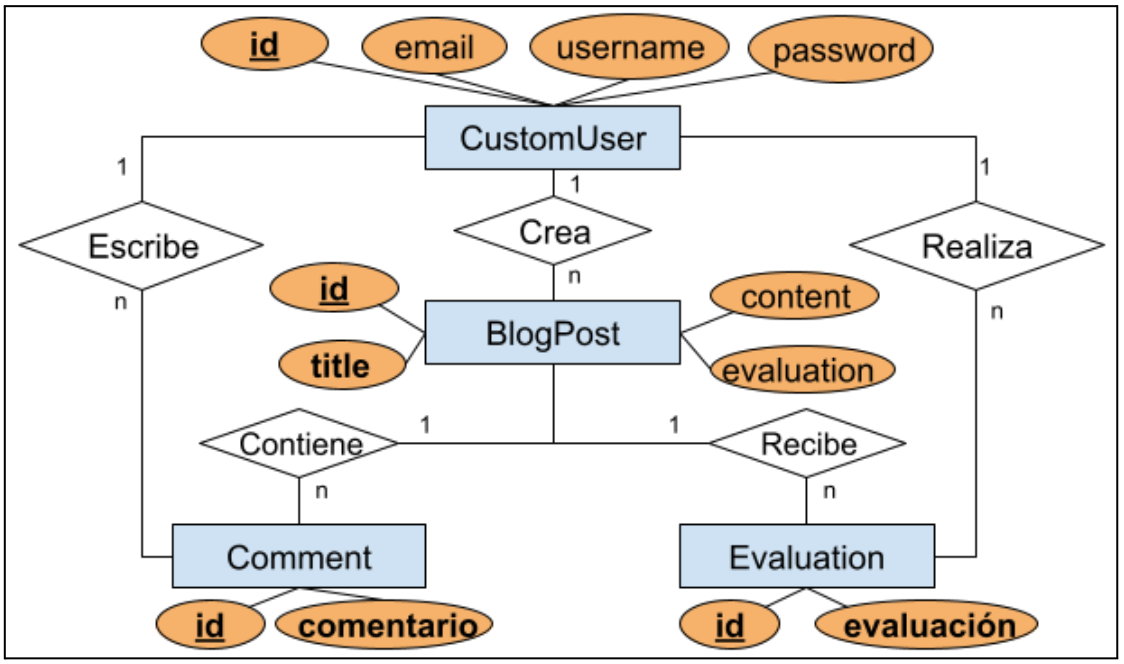

Figura 4.5. - Diseño. Diagrama Entidad/Relación de la BDD.

A continuación se resumen brevemente las entidades (tablas) y sus atributos:

- BlogPost: Esta tabla representa una publicación o un post de Evaluómetro. Los campos de esta tabla son los siguientes:

  » author: El usuario que creó la publicación.
  » user_participation: Un número entero que por defecto es 0. Indica la cantidad de interacciones de los usuarios con la publicación. Esta aumenta cuando los usuarios evalúan un post.
  » title: El título de la publicación.
  » content: El contenido de la publicación.
  » evaluation: Una puntuación decimal que varía entre 0 y 5 para la publicación. Esta puntuación es la media de todas las evaluaciones de los usuarios sobre este post.
  » image: Una imagen asociada con la publicación.

» price: Un campo opcional que podría ser usado para almacenar un precio asociado con la publicación.
» where_to_find: Campo de texto opcional, podría ser usado para almacenar información sobre dónde encontrar algo relacionado con la publicación.
» created_at: La fecha y hora en que se creó la publicación.

- Comment: Esta tabla representa los comentarios realizados a una publicación. Contiene los siguientes campos:

» author: El usuario que realizó el comentario. Referencia a la tabla de usuarios.
» post: Publicación a la que se refiere el comentario. Referencia a la tabla BlogPost.
» comment_text: El texto del comentario.
» created_at: La fecha y hora en que se creó el comentario.

- Evaluation: Esta tabla se utiliza para almacenar las calificaciones que los usuarios dan a las publicaciones.

» author: El usuario que dio la calificación. Referencia a la tabla de usuarios.
» post: La publicación que fue calificada. Referencia a la tabla BlogPost.
» rating: Es la calificación que da el usuario a la publicación, consiste en un valor decimal entre 1 y 5.

- CustomUser: Esta tabla representa a los usuarios en el sistema. Los campos de esta tabla son:

» email: El correo electrónico del usuario. Este campo es único, es decir, no puede haber dos usuarios con el mismo correo electrónico.
» username: El nombre de usuario. Este campo también es único.
» is_active: Un indicador de si el usuario está activo.
» is_admin: Un indicador de si el usuario es administrador.

- AdminLog: Esta tabla no forma parte del diseño original, la crea el propio Django para realizar un seguimiento de los movimientos del administrador, por tanto, no influye en el funcionamiento directo de la aplicación.

En cuanto a las relaciones entre las entidades de la base de datos, se explican del siguiente modo:

- Crear: Un CustomUser crea un post, la relación muchos-a-uno entre BlogPost y CustomUser representa que un usuario puede crear muchos posts pero cada post solo ha podido ser creado por un usuario.

- Contiene: Relación de muchos a uno de Comentario con BlogPost, ya que un post puede contener muchos comentarios, mientras que cada comentario individual solo pertenecen a un post.

- Escribe: Esta relación de muchos a uno de Comentario con un CustomUser (el autor), puesto que un usuario puede tener muchos comentarios y cada comentario solo puede ser escrito por un único usuario.

- Recibe: Existe una relación de muchos a uno entre una Evaluación y un post, ya que un post puede recibir varias evaluaciones, y cada evaluación solo pertenece a un post.

- Realiza: Una relación de muchos a uno también entre Evaluación y CustomUser, debido a que una evaluación solo pertenece a un usuario y un usuario puede realizar muchas evaluaciones distintas.

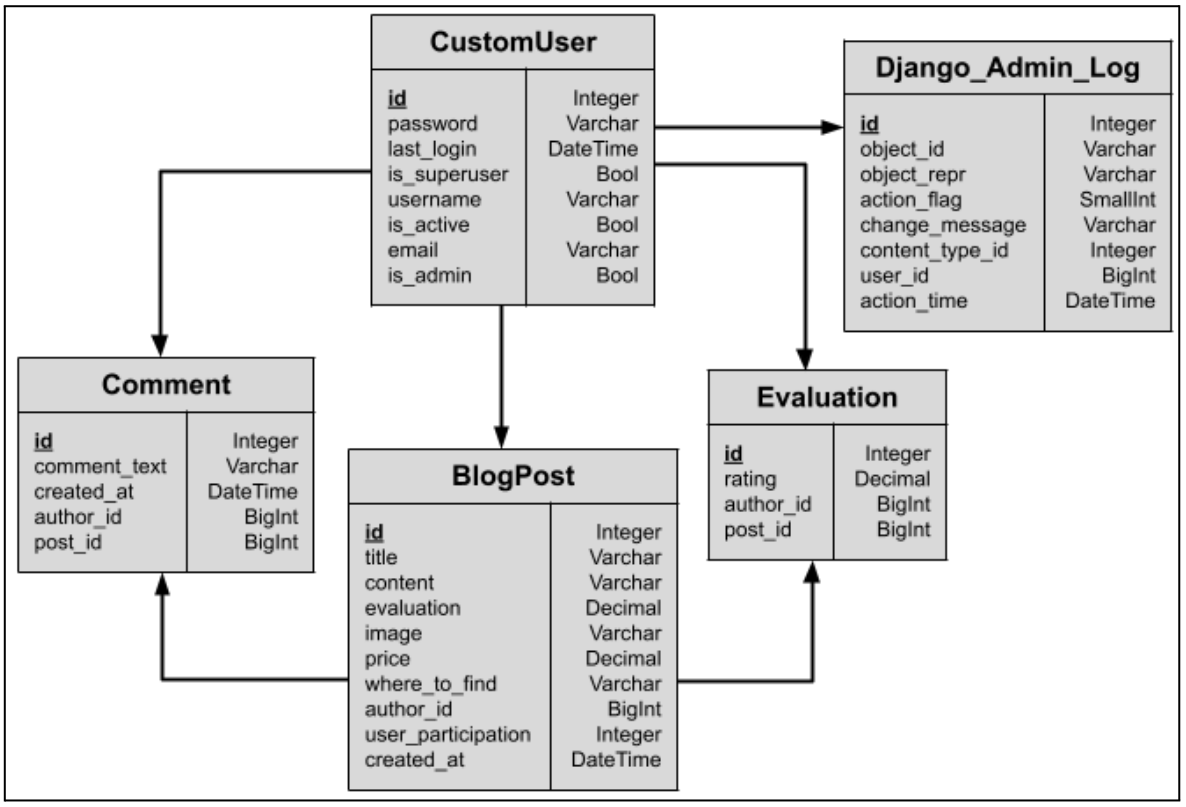

Figura 4.6. -Diseño relacional de la BDD.

4.3.2.- DIAGRAMAS DE SECUENCIA

El Modelo-Vista-Controlador (MVC) es la base sobre la que se construye la arquitectura de software de este proyecto. Esto implica que, para una mayoría de casos de uso, el patrón de interacción entre los distintos componentes del sistema será muy similar. Los casos de uso

principales se pueden agrupar en dos clases o tipos según su forma de interactuar con la aplicación.

El principal método de interacción de los usuarios con el sistema son los formularios. El sistema utiliza el mismo tipo de proceso para todas las acciones, incluido el registro, el inicio de sesión, la creación de posts, la evaluación y los comentarios.

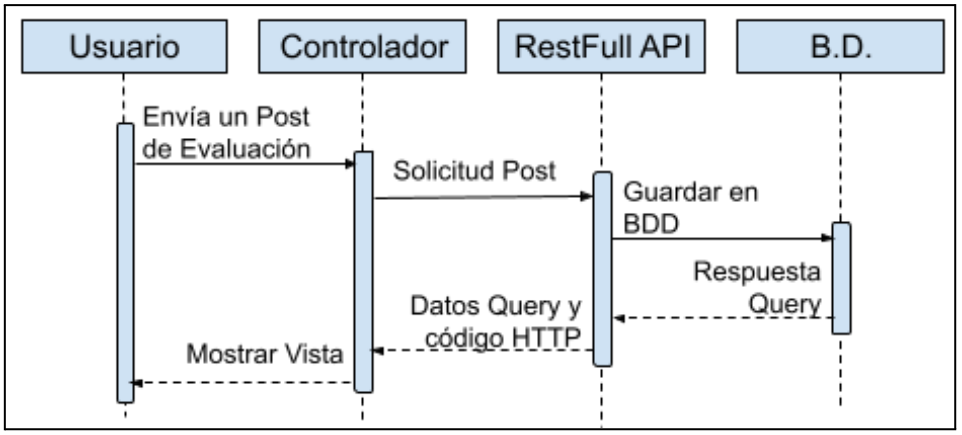

Figura 4.7. - Diagrama de Secuencia. Crear un post.

La figura 4.7 muestra lo que ocurre cuando un usuario envía un formulario pulsando el botón "Enviar". El flujo de llamadas sigue el siguiente orden:

1. El usuario, interactuando con la vista, una solicitud al controlador al pulsar el botón “Enviar”
2. El controlador manda una petición POST, GET o PUT a la API RESTful de la aplicación, implementada en el modelo.
3. A continuación, la API lanza una consulta a la base de datos para recuperar o proporcionar los datos necesarios.
4. Los datos y el código de respuesta HTTP se devuelven al controlador una vez que la consulta ha terminado de ejecutarse.
5. En función de los datos y del código de respuesta, el controlador elige la vista o el mensaje adecuado que se mostrará al usuario.
6. El usuario es informado instantáneamente del resultado o de si hay algún error en la solicitud.

Esto capta esencialmente cómo funciona el sistema basado en MVC, independientemente de si el usuario ejecuta una o más acciones en el sistema, esto se repite en la mayoría de situaciones.

En la figura 4.8 se ilustra cómo sería el flujo de acciones al pulsar los diferentes enlaces y botones de la aplicación para lo que se podrían denominar “*interacciones menores*” como cambiar de pantalla, abrir/cerrar ventanas, etc. Dado que en estos casos no es necesario

consultar la base de datos o la API para obtener información, este enfoque es mucho más sencillo. Utilizando el contexto, el controlador determina si el usuario ha iniciado sesión y si tiene los permisos necesarios para acceder al recurso solicitado.

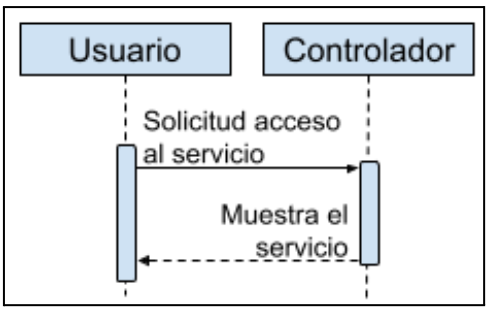

Figura 4.8. - Diagrama de Secuencia. El usuario interactúa con un botón.

4.4.- Implementación

4.4.1.- CREAR/MODIFICAR POSTS

Evaluómetro es una plataforma social interactiva dedicada a la evaluación y discusión de una variedad de temas. En su esencia, permite a los usuarios generar y modificar publicaciones, a través de las cuales pueden compartir sus evaluaciones y comentarios, promoviendo un intercambio de opiniones enriquecedor y constructivo. Estas funciones son el corazón de Evaluómetro, ya que fomentan la creación de contenido y la interacción continua entre los usuarios.

Para comprender mejor el funcionamiento de Evaluómetro, se va a explorar el código de las vistas. Las vistas, que forman parte del patrón de diseño Modelo-Vista-Controlador (MVC) utilizado en Django, son las responsables de manejar las principales funciones de nuestra plataforma. A través de este análisis, se podrá entender la lógica del código que hace posible la creación, modificación, evaluación y comentario de las publicaciones en Evaluómetro.

Se comienza con la clase ***BlogCreateView*** (figura 4.9). La gestión de los nuevos posts del blog es competencia de esta clase. Para ello es necesario iniciar sesión, de ahí el parámetro ***LoginRequiredMixin***. Para redirigir a los usuarios que no han iniciado sesión, se proporciona la URL de inicio de sesión.

Se crea un ***PostCreateForm*** en blanco, el cual es un formulario creado para rellenar unos campos que se enviarán al servidor para guardar un post en la base de datos. Tras esto, se dan a los campos del formulario algunas clases CSS en el método ***Get***. A la plantilla

'blog_create.html', que es donde esta el formulario de creación de un post, se le da esto para que lo muestre.

Tras esto se tiene el método ***Post***, que es donde los datos de la plantilla ***'blog_create.html'*** que fueron enviados a través del formulario son procesados. El usuario es redirigido a la página de inicio si los datos del formulario son correctos, después de lo cual se crea y guarda un nuevo post. Los datos no válidos del formulario se imprimen si existen.

```python
class BlogCreateView(LoginRequiredMixin, View):
    login_url = 'usuario:login'  # Actualiza este valor según el nombre de

    def get(self, request, *args, **kwargs):
        form = PostCreateForm()
        form.fields['title'].widget.attrs.update({'class': '...'})
        form.fields['content'].widget.attrs.update({'class': '...'})
        form.fields['evaluation'].widget.attrs.update({'class': '...'})
        form.fields['image'].widget.attrs.update({'class': '...'})
        form.fields['price'].widget.attrs.update({'class': '...'})
        form.fields['where_to_find'].widget.attrs.update({'class': '...'})

        context = {
            'form': form
        }
        return render(request, 'blog_create.html', context)

    def post(self, request, *args, **kwargs):
        form = PostCreateForm(request.POST, request.FILES)
        if form.is_valid():
            new_post = form.save(commit=False)
            new_post.author = request.user
            new_post.save()
            return redirect('blog:home')
        else:
            print("Formulario no válido:", form.errors)

        context = {
            'form': form
        }
        return render(request, 'blog_create.html', context)
```

Figura 4.9. - Código de la vista ***BlogCreateView***.

En la vista ***BlogUpdateView***, además de recuperar el objeto de post que tiene que ser modificado, el método ***Get*** es similar al método ***Get*** de ***BlogCreateView***. Los nuevos datos del formulario son procesados por el método ***Post***. Si el formulario es legítimo, edita el

contenido, reemplaza cualquier imagen que falte y envía al usuario de vuelta a la página principal.

```python
class BlogUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    login_url = 'usuario:login'  # Actualiza este valor según el nombre de la U

    model = Post
    form_class = PostUpdateForm
    template_name = 'blog_update.html'

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False

    def get(self, request, *args, **kwargs):
        self.object = self.get_object()
        form = self.get_form()
        form.fields['title'].widget.attrs.update({'class': '...'})
        form.fields['content'].widget.attrs.update({'class': '...'})
        form.fields['evaluation'].widget.attrs.update({'class': '...'})
        form.fields['price'].widget.attrs.update({'class': '...'})
        form.fields['where_to_find'].widget.attrs.update({'class': '...'})

        context = {
        'form': form,
        'object': self.object,
        'form_errors': form.errors,
        }
        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        print("Método post llamado")
        return super().post(request, *args, **kwargs)

    def get_success_url(self):
        pk = self.kwargs['pk']
        return reverse_lazy('blog:home')

    def form_valid(self, form):
        post = form.save(commit=False)
        delete_image = self.request.POST.get('delete_image', '')
        if delete_image == 'true':
            post.image.delete()

        post.save()
        messages.success(self.request, 'Post actualizado con éxito.')
        return HttpResponseRedirect(self.get_success_url())
```

Figura 4.10. - Código de la vista ***BlogUpdateView***.

Los detalles de un post del blog se muestran utilizando la clase ***BlogDetailView***. El contenido de un post, sus comentarios y evaluaciones se obtienen utilizando el método ***Get*** y se envían a la plantilla ***'blog_detail.html'***, que es donde se presentan los campos de un post al usuarios para su visualización.

```
class BlogDetailView(LoginRequiredMixin, View):
    login_url = 'usuario:login'

    def get(self, request, pk, *args, **kwargs):
        post = get_object_or_404(Post, pk=pk)
        comments = Comment.objects.filter(post=post).order_by('-created_at')
        evaluations = Evaluation.objects.filter(post=post)
        context = {
            'post': post,
            'comments': comments,
            'evaluations': evaluations,
        }
        return render(request, 'blog_detail.html', context)
```

Figura 4.11. - Código de la vista ***BlogDetailView***.

Las clases ***AddCommentView*** y ***AddEvaluationView*** se encargan de añadir comentarios y evaluaciones a un post. El método ***Post*** de ambas clases recupera un post y añade un nuevo comentario o evaluación a dicho post.

```python
class AddCommentView(LoginRequiredMixin, View):
    def post(self, request, pk, *args, **kwargs):
        post = get_object_or_404(Post, pk=pk)
        comment_text = request.POST.get('comment_text')
        Comment.objects.create(post=post, author=request.user, comment_text=comment_text)
        return redirect('blog:detail', pk=post.pk)

class AddEvaluationView(LoginRequiredMixin, View):
    def post(self, request, pk, *args, **kwargs):
        post = get_object_or_404(Post, pk=pk)
        rating = request.POST.get('rating')
        # Intenta obtener la evaluación existente
        evaluation, created = Evaluation.objects.get_or_create(
            post=post,
            author=request.user,
            defaults={'rating': rating},
        )
        # Si la evaluación es nueva, incrementa la participación del usuario
        if created:
            post.user_participation += 1
            post.save()
        # Si la evaluación ya existía, actualiza su calificación
        if not created:
            evaluation.rating = rating
            evaluation.save()
        # Recalcula la evaluación del post
        ratings = [evaluation.rating for evaluation in post.evaluations.all()]
        total_evaluations = len(ratings)
        total_score = sum(ratings)
        # Solo cuenta la evaluación inicial si no hay evaluaciones de usuarios
        if total_evaluations == 0:
            total_evaluations += 1
            total_score += post.evaluation
        post.evaluation = total_score / total_evaluations
        post.save()
        return redirect('blog:detail', pk=post.pk)
```

Figura 4.12. - Código de las vistas ***AddCommentView*** y ***AddEvaluationView***.

Finalmente, se va a explicar una funcionalidad del proyecto algo más compleja, pero a su vez importante. Esta funcionalidad es el Infinite Scroll, la cual consiste en ir cargando y mostrando posts a medida que baje el usuario.

En el lado del servidor, se cuenta con la vista ***BlogListView*** que es responsable de manejar las solicitudes de la lista de posts:

```python
class BlogListView(LoginRequiredMixin, View):
    login_url = 'usuario:login'

    def get(self, request, *args, **kwargs):
        start = int(request.GET.get('start', 0))
        end = start + 6
        filter = request.GET.get('filter', 'all')
        query = request.GET.get('q')

        order_param = request.GET.get('order', 'newest')  # Defecto a 'newest'
        order_mapping = {
            'newest': 'created_at',
            'oldest': '-created_at',
            'most_stars': '-evaluation',
            'most_users': '-user_participation'
        }
        order = order_mapping.get(order_param, '-created_at')  # Fallback a '-created_at' si el order_param no es válido

        if query:  # Si es una búsqueda
            if filter == 'my_posts':
                posts = Post.objects.filter(author=request.user, title__icontains=query).order_by(order)[start:end]
            else:
                posts = Post.objects.filter(title__icontains=query).order_by(order)[start:end]
        else:  # Si es filtrado y ordenamiento
            if filter == 'my_posts':
                posts = Post.objects.filter(author=request.user).order_by(order)[start:end]
            else:
                posts = Post.objects.all().order_by(order)[start:end]

        data = []
        for post in posts:
            data.append({
                "id": post.id,
                "title": post.title,
                "content": post.content,
                "image": post.image.url if post.image else None,
                "evaluation": post.evaluation,
                "user_participation": post.user_participation,
                "is_author": post.author == request.user,
                "detail_url": request.build_absolute_uri(reverse('blog:detail', args=[post.id])),
            })

        if request.headers.get('x-requested-with') == 'XMLHttpRequest':
            return JsonResponse(data, safe=False)

        return render(request, 'blog_list.html', {'posts': data[:6]})
```

Figura 4.13. - Código de la vista ***BlogListView***.

Esta vista toma un parámetro de inicio, que se utiliza para determinar el punto de partida de los posts que se mostrarán en la pantalla de inicio. También toma un parámetro de orden para determinar el orden de los posts, y un parámetro de filtro para aplicar un filtro a estos. A continuación, consulta la base de datos en busca de los mensajes que coincidan con estos parámetros y los envía de vuelta al cliente en formato JSON si la solicitud es una solicitud AJAX, o los muestra en la plantilla ***blog_list.html*** si se trata de una solicitud GET normal.

En el lado del cliente, el código JavaScript se encarga del desplazamiento infinito:

```
function loadPosts() {
    // Si no hay más posts para cargar, sal del método temprano
    if (!hasMorePosts && start > 0 || isLoading) {
        return;
    }
    isLoading = true;
    // Muestra el div de carga antes de iniciar la llamada AJAX
    $("#loading").removeClass("hidden");

    let loadMoreUrl = "{% url 'blog:home' %}" + '?start=' + start + '&order=' + order + '&filter=' + filter + '&search=' + searchQuery;
    console.log("load: ", loadMoreUrl, start, order, filter, searchQuery);
    $.ajax({
        url: loadMoreUrl,
        type: 'GET',
        success: function(data) {
            let posts = data; // Parsea la respuesta JSON
            if (posts.length === 0) { // Si la API devuelve un array vacío, no hay más posts para cargar
                hasMorePosts = false;
            } else {
                posts.forEach(post => {
                let editDeleteButtons = '';
                if (post.is_author) {
                    editDeleteButtons = `
                        <a class="..." href="/blog/${post.id}/update">Editar</a>
                        <a class="..." href="/blog/${post.id}/delete">Eliminar</a>
                    `;
                }
                $('#post-list').append(`
                <div class="post" style="...">
                    <div class="post-image" style="height: 256px; width: 100%; border-radius: 10px; overflow: hidden;">
                        ${post.image ? `<img src="${post.image}" alt="${post.title}" style="width: 100%; height: 100%; object-fit: cover;">` :
                        `<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 576 512" style="width: 100%; height: auto; max-height: 100%;">
                            <path fill="white" d="..."></path>
```

Figura 4.14. - Código JavaScript Ajax del cliente.

```
                            path>
                        </svg>`}
                    </div>
                    <div class="w-full">
                        <h2 class="text-3xl font-bold text-white py-2 dark:text-gray-200">
                            <a class="dark:text-slate-400 py-6 mt-2 text-4xl" href="${post.detail_url}">
                                ${post.title}
                            </a>
                            <hr style="border-top: 1px solid white; margin: 10px 0;">
                        </h2>
                        <p class="text-white dark:text-gray-200 color: #ddd;">${ post.content }</p>
                        <p class="text-white dark:text-gray-200 py-2">Evaluación: ${ post.evaluation }(${ post.user_participation })</p>
                        ${editDeleteButtons}
                    </div>
                </div>
            `);
        });
            start += posts.length;
            checkIfMoreContentNeeded();
            }
        // Añade un pequeño retraso antes de ocultar el div de carga
        setTimeout(() => {
            $("#loading").addClass("hidden");
        }, 500);
    },
```

Figura 4.15. - Código JavaScript Ajax del cliente.

```
let scrollTimeout;
$(window).scroll(function() {
    resetCalled = false;
    clearTimeout(scrollTimeout);

    if ($(window).scrollTop() == $(document).height() - $(window).height()) {
        scrollTimeout = setTimeout(loadPosts, 500);
    }
});
```

Figura 4.16. - Código JQuery del cliente.

Este código JavaScript establece inicialmente ***start*** en 0, order en "***oldest***" y filter en "***all***". Cuando se llama a la función ***loadPosts***, se envía una petición AJAX al servidor con estos parámetros. El servidor responde con los posts que el cliente añade al final de la lista de posts. La variable de inicio se incrementa con el número de posts recibidos.

La parte de desplazamiento es manejada por la función $(window).scroll. Cada vez que el usuario se desplaza hasta el final de la página, se llama a la función ***loadPosts*** después de un retardo de 500 milisegundos (para asegurar que no se hacen múltiples llamadas si el usuario se desplaza rápidamente). Esto recupera el siguiente conjunto de posts y los añade al final de la lista, creando el efecto de desplazamiento infinito.

La parte del servidor de esta funcionalidad se implementa en Django, utilizando la clase ***BlogListView***. Esta vista gestiona las peticiones GET para obtener los posts del blog. Espera ciertos parámetros en la petición:

- ***start***: Este parámetro especifica el índice de el primer post a obtener. Por defecto es 0 si no se proporciona.
- ***end***: Se calcula como el parámetro de inicio más 6. Por lo tanto, cada petición obtiene 6 posts.
- ***filtro***: Este parámetro se puede utilizar para filtrar los mensajes. Si no se indica, el valor predeterminado es "all".
- ***orden***: Este parámetro se utiliza para ordenar los mensajes. Por defecto es 'newest' si no se indica.
- ***q***: Este parámetro se utiliza para buscar en los títulos de los posts.

El parámetro order se asigna a un diccionario ***order_mapping***, que determina el campo de ordenación en la base de datos. Si el parámetro order no es válido, se vuelve a '***-created_at'***, ordenando los posts de los más recientes a los más antiguos.

A continuación, la vista filtra y ordena los posts en consecuencia, crea una lista de diccionarios para los posts a devolver, cada uno con la información relevante (id, título,

contenido, etc.) y comprueba si la petición se ha realizado con AJAX. En caso afirmativo, devuelve un ***JsonResponse*** con los datos. En caso contrario, muestra la plantilla ***'blog_list.html'*** con los 6 primeros posts.

La parte cliente del scroll infinito se implementa con JavaScript, jQuery y AJAX.

Cuando el documento está listo, se llama a la función ***loadPosts***, que envía una petición GET al servidor con los valores actuales de inicio, orden y filtro. Si la respuesta contiene posts, los añade al elemento ***#post-list***. Si no, establece ***hasMorePosts*** a false, deteniendo futuras peticiones.

Por último, la función ***resetPosts*** se utiliza para borrar los mensajes actuales y restablecer las variables ***start*** y ***hasMorePosts***. Esta función se ejecuta cada vez que el usuario cambia el orden o los parámetros de filtrado, lo que se gestiona mediante escuchadores de eventos añadidos a los elementos del menú desplegable.

En resumen, esta implementación de desplazamiento infinito funciona obteniendo un conjunto de mensajes del servidor cada vez que el usuario se desplaza hasta el final de la página, y añadiendo estos mensajes al final de la lista. El servidor mantiene un registro de los mensajes que debe enviar a continuación utilizando el parámetro ***start***. El cliente lleva la cuenta de hasta dónde ha llegado incrementando ***start*** cada vez que recibe un conjunto de mensajes. Esta característica mantiene la carga de nuevos contenidos a medida que el usuario se desplaza por la página, proporcionando una experiencia sin fisuras, especialmente cuando se ve una gran lista de mensajes como en un blog.

4.4.2.- DISEÑO DE LA APLICACIÓN WEB

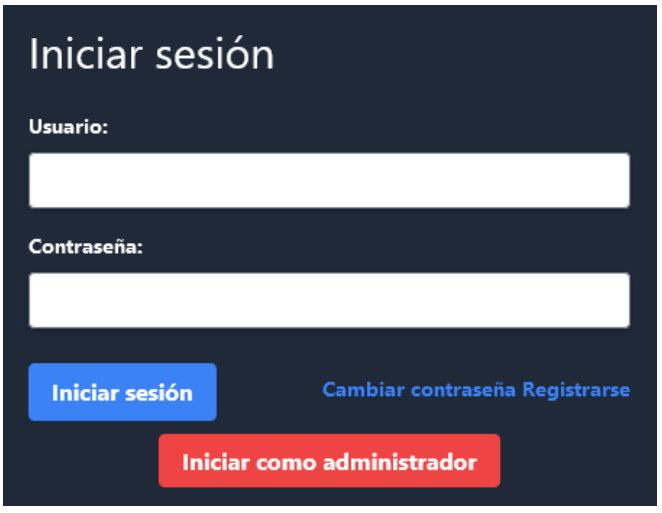

Figura 4.17. - Inicio de sesión de la Web.

Lo primero que deberá hacer el usuario al entrar en la web será iniciar sesión. En el formulario de la figura 4.17 el usuario podrá iniciar sesión, cambiar su contraseña si la ha olvidado o registrarse. También podrá ir al inicio de sesión del administrador.

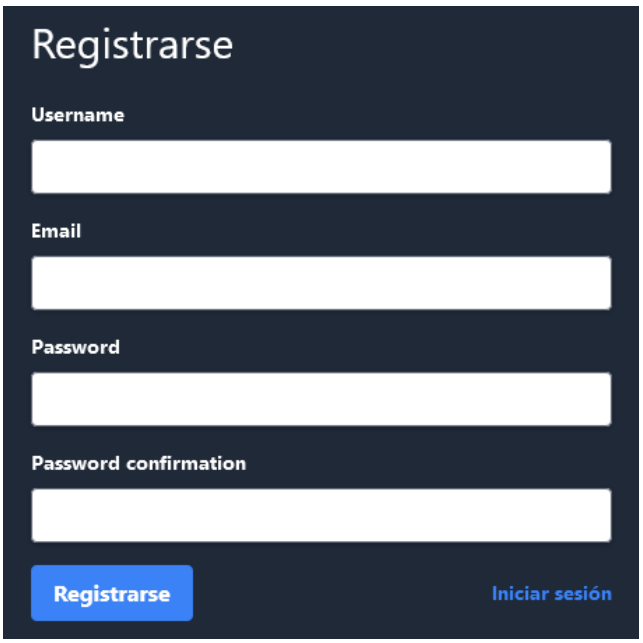

Figura 4.18. - Inicio del registro de la Web.

En esta página el usuario podrá registrarse introduciendo sus credenciales o ir al inicio de sesión.

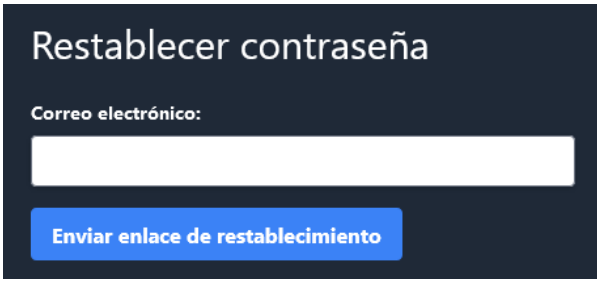

Figura 4.19. - Restablecimiento de contraseña de la Web.

En esta recuperar su contraseña introduciendo su correo electrónico o ir al inicio de sesión.

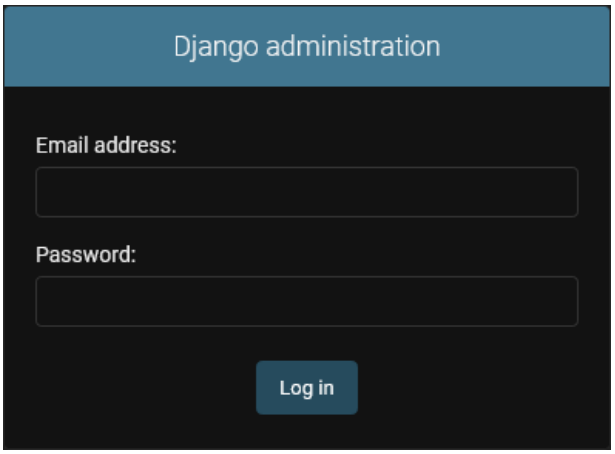

Figura 4.20. - Inicio de sesión del administrador de la Web.

En esta web el usuario podrá iniciar sesión como administrador o volver al inicio de sesión.

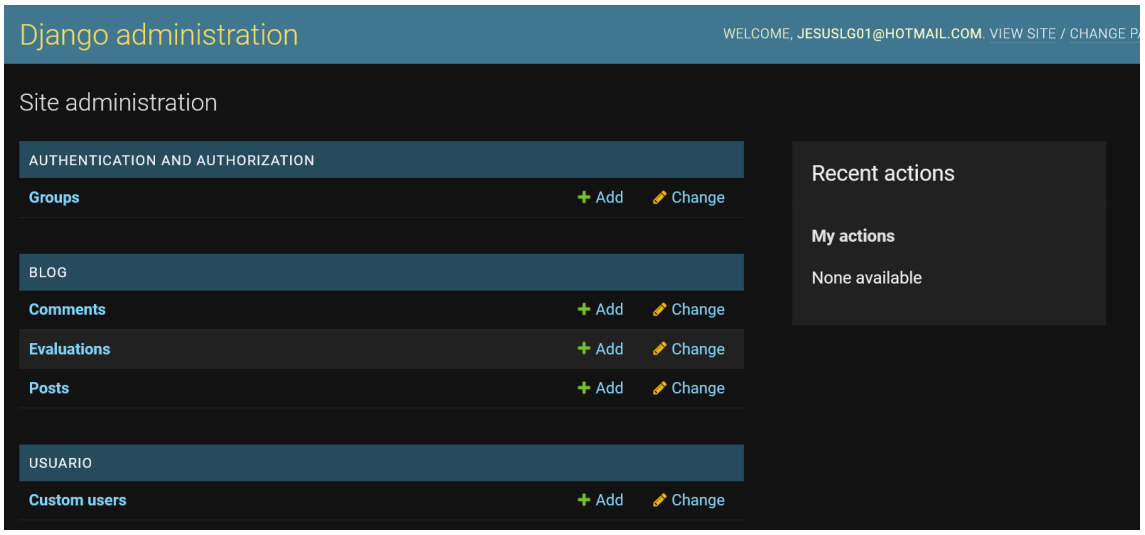

Figura 4.21. - Gestión de las tablas del administrador en la web.

Aquí el usuario administrador podrá gestionar las tablas de la base de datos, creando, editando o eliminando posts, usuarios, comentarios o evaluaciones.

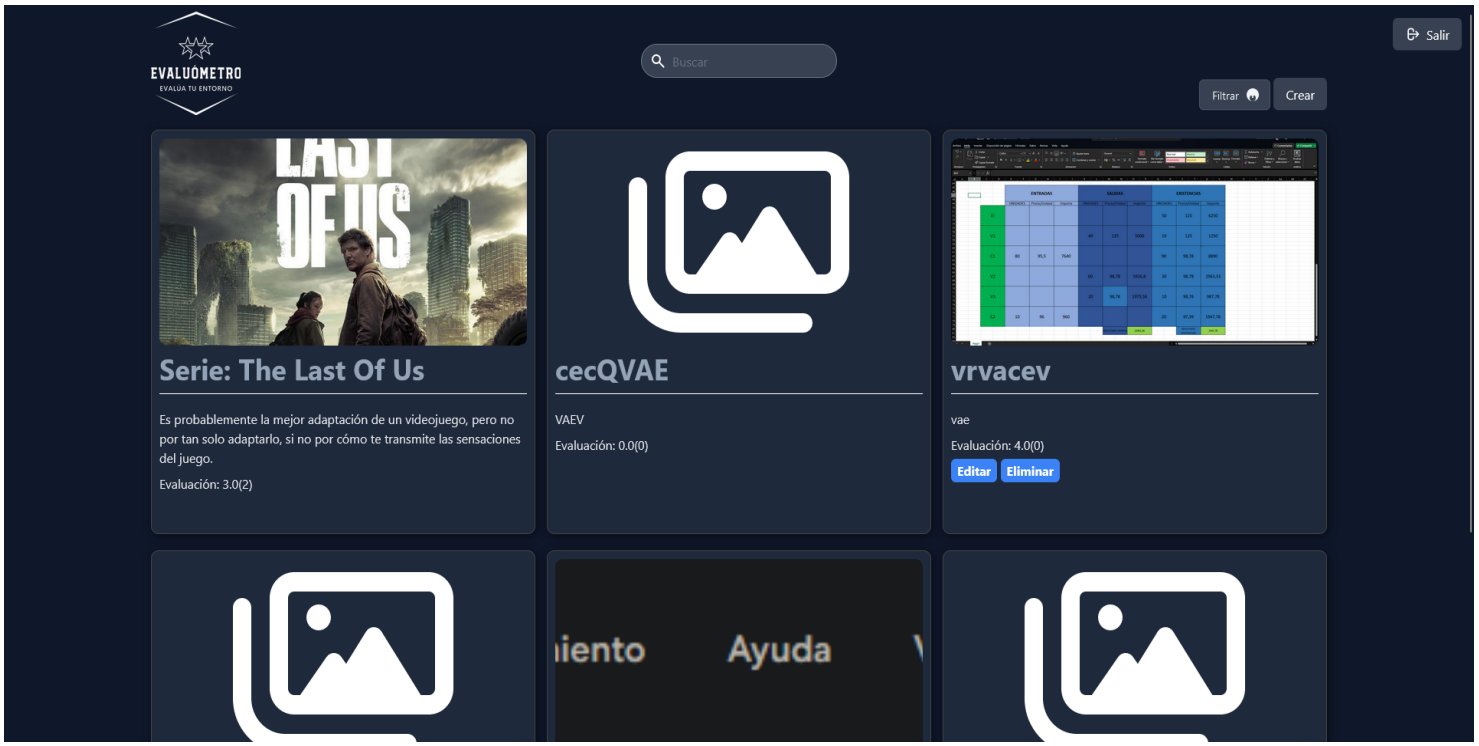

Figura 4.22. - Imagen de la página principal de la Web.

Esta es la página principal de Evaluómetro, donde el usuario registrado podrá ver todos los posts, buscar posts y usuarios, filtrar posts, editar o los posts que sean suyos, crear posts nuevos o cerrar sesión.

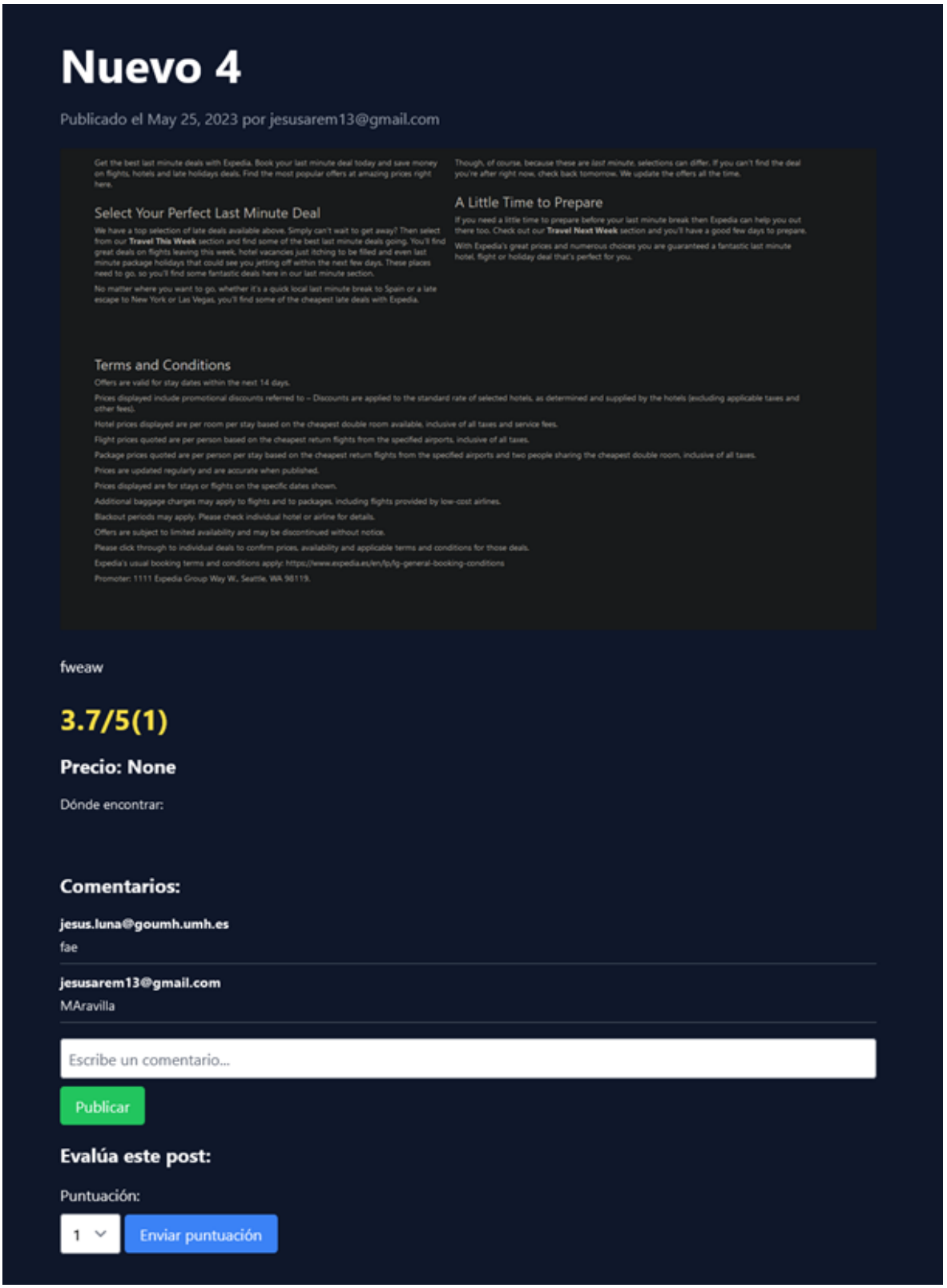

Figura 4.23. - Imagen de la visualización de un post de la Web.

En esta página el usuario podrá visualizar posts de otros usuarios y comentarlos o evaluarlos.

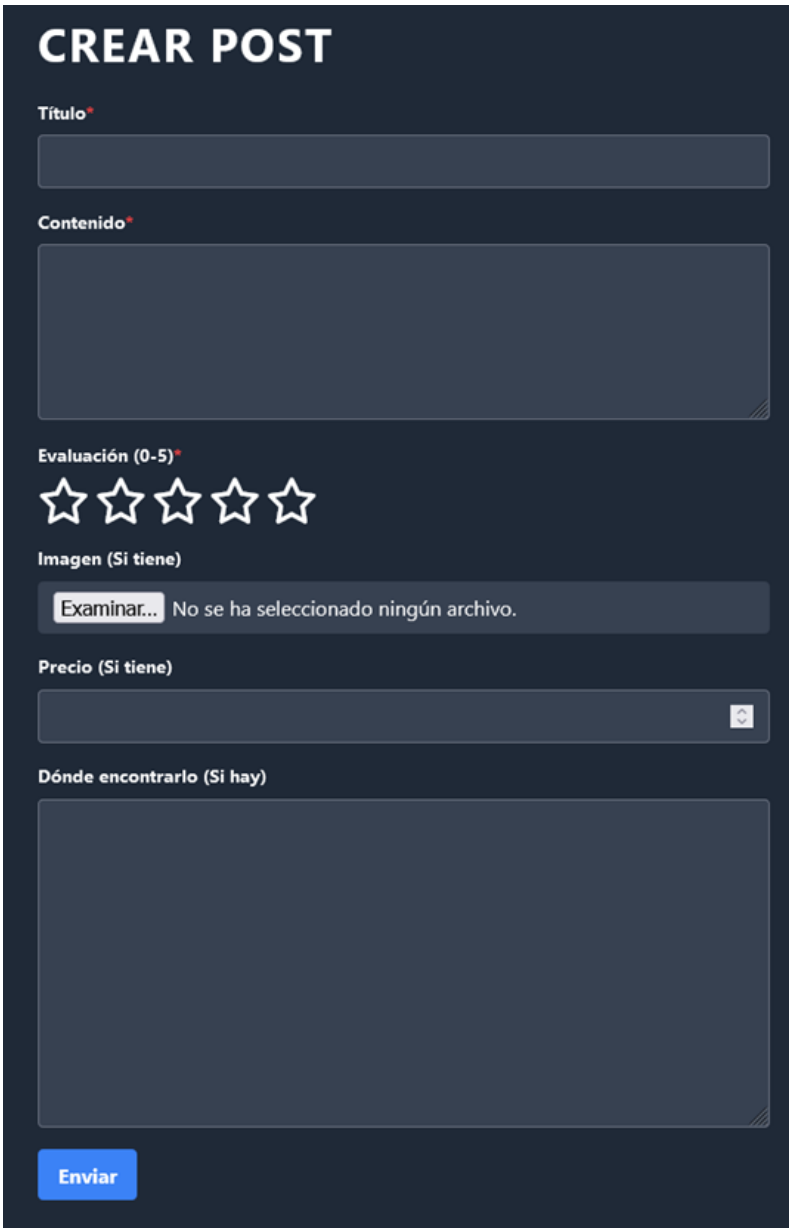

Figura 4.24. - Imagen de la creación de un post en la web.

El usuario también podrá crear posts nuevos, rellenando los siguientes apartados. Los apartados marcados con una estrellita roja son obligatorios.

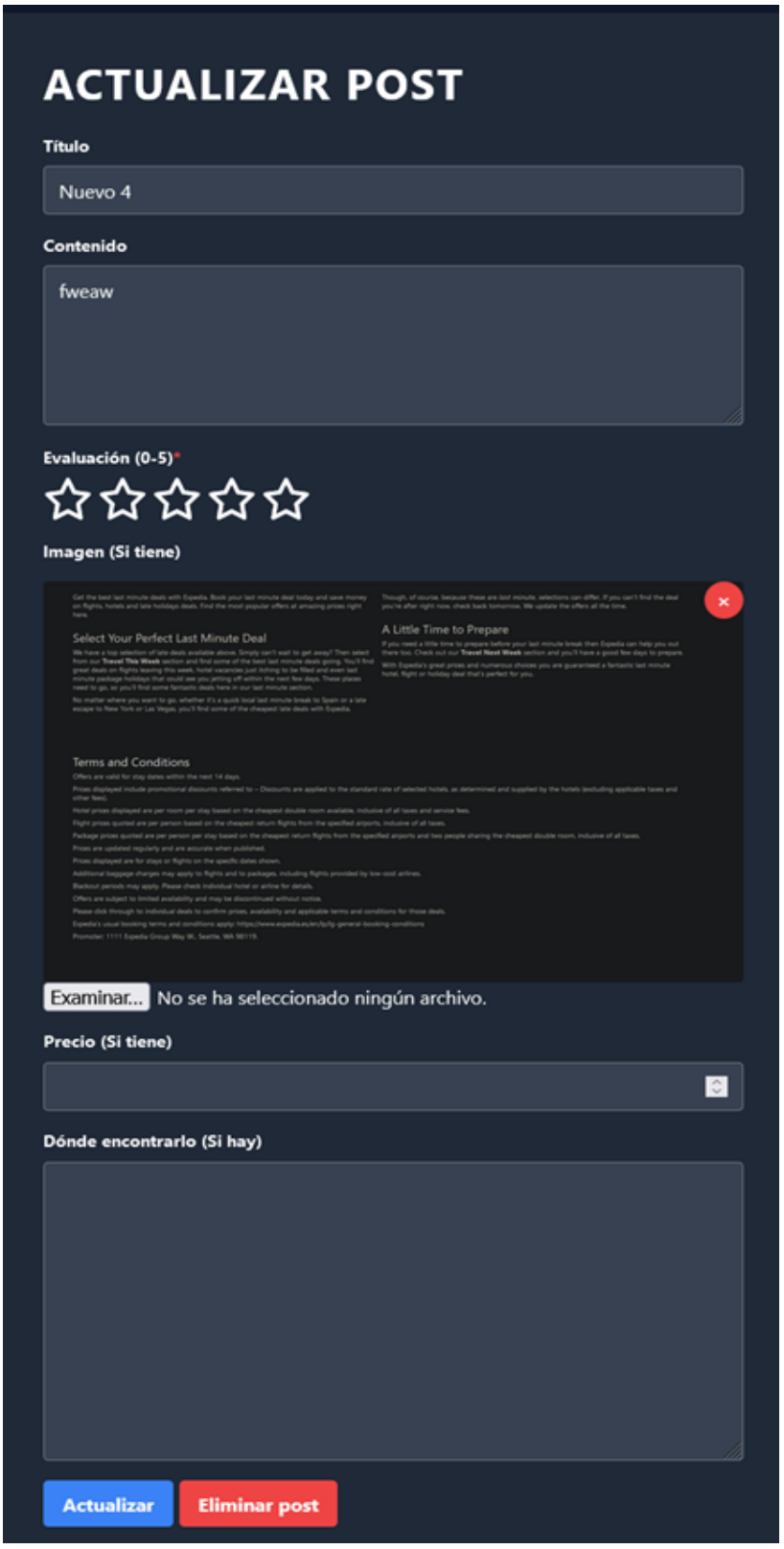

Figura 4.25. - Imagen de la actualización de posts de la Web.

Por último, en esta página el usuario podrá editar o eliminar sus propios posts.

4.4.3.- Pruebas/Implantación

El proceso de implementación y prueba en este proyecto ha progresado de forma simultánea, con cada componente funcional desarrollado y evaluado a la vez.

En el ámbito del backend, el desarrollo de microservicios se llevó a cabo según las necesidades de las funciones específicas para las que fueron diseñados. Posteriormente, estos servicios se sometieron a rigurosas pruebas para garantizar su eficacia operativa.

En resumen, el proceso de implantación y verificación de los sistemas ha sido intensivo y ha exigido un compromiso de tiempo considerable para garantizar la validación exhaustiva de los distintos componentes. Este meticuloso planteamiento ha sido esencial para garantizar el óptimo funcionamiento del producto final, en este caso Evaluómetro.

Capítulo 5 Conclusiones y trabajo futuro

5.1.- CONCLUSIONES

La realización del Evaluómetro como Proyecto Fin de Carrera ha resultado ser una experiencia valiosa y exigente, facilitando la consecución de numerosos objetivos al inicio del proyecto.

El objetivo principal de crear una red social para evaluar productos, servicios y experiencias cotidianas se ha cumplido con éxito. Los usuarios tienen la posibilidad de registrarse, iniciar y cerrar sesión, generar y comentar mensajes, buscar usuarios y mensajes que despierten su interés, seguir a sus compañeros y evaluar mensajes para contribuir al sentimiento colectivo. Esto resume una gran parte de las funcionalidades técnicas que se propusieron, lo que sirve como recompensa por el esfuerzo y el compromiso en el proyecto.

No obstante, es importante reconocer que algunos objetivos quedaron sin cumplir. La posibilidad de modificar los perfiles de los usuarios y la incorporación de inteligencia artificial no se hicieron realidad debido a restricciones temporales. Es fundamental recalcar que este proyecto ha sido creado por un único desarrollador, de manera que no ha sido posible implementar todas las funcionalidades previstas, sin embargo estos elementos podrían considerarse posibles vías de mejora y futuros avances.

En cuanto a los objetivos personales, este proyecto ha servido para adquirir importantes conocimientos en el desarrollo de aplicaciones web, haciendo especial hincapié en las tecnologías back-end. Se ha adquirido dominio y familiaridad con lenguajes y herramientas como Python, Django, HTML, CSS, JavaScript, Git, etc. Aunque la inteligencia artificial no se terminó de integrar, la experiencia adquirida por el proyecto proporciona una base sólida para su posible inclusión en el futuro.

Es pertinente destacar que, aparte de su ejecución técnica, Evaluómetro alberga el potencial de influir positivamente en los procesos de toma de decisiones de los consumidores al ofrecer una plataforma para acceder a valoraciones y puntos de vista sobre bienes y servicios cotidianos. La aspiración de crear no sólo una herramienta útil, sino también un centro de interacción y entretenimiento para el usuario, la distingue.

Como conclusión, este proyecto de fin de carrera, no significa sólo el final de una era académica, sino también un nuevo comienzo. Los conocimientos y habilidades perfeccionados durante este proyecto constituyen la base de un crecimiento profesional. Los elementos que quedaron sin explorar en esta fase, se plantean como próximos retos y oportunidades de expansión.

5.2.- POSIBLES DESARROLLOS FUTUROS

Al prever la trayectoria de avance de Evaluómetro, fueron identificadas multitud de posibles ampliaciones. Estas incluyen funcionalidades originalmente previstas pero no realizadas, y características novedosas que podrían enriquecer la interacción con el usuario y facilitar la difusión del conocimiento.

- Personalización del perfil de usuario: Una limitación notable del diseño actual es la falta de capacidades de modificación del perfil de usuario. Es vital proporcionar a los usuarios autonomía para personalizar sus perfiles, incorporando elementos como una descripción concisa y, posiblemente, enlaces a otras plataformas de medios sociales. Esta mejora fomentaría el sentido de comunidad, permitiendo a los usuarios articular su identidad de forma más genuina.

- Integración de la Inteligencia Artificial: La IA podría contribuir a enriquecer la experiencia del usuario. Por ejemplo, los sistemas de recomendación podrían proponer publicaciones y otros usuarios a los que seguir, basándose en los patrones de navegación y las preferencias individuales. También, la IA podría ayudar a moderar los contenidos, garantizando el cumplimiento de las directrices de la comunidad en comentarios y publicaciones. Por último, dentro de cada post, la IA podría contribuir notoriamente al entendimiento de este, haciendo un resumen del mismo al inicio, debajo de la presentación del post o con un glosario de las palabras o términos más relevantes en los comentarios del post.

- Mecanismo de notificaciones en tiempo real: la implantación de un sistema de notificaciones en tiempo real mantendría informados a los usuarios sobre interacciones como comentarios a sus publicaciones, nuevos seguidores o menciones en debates. Esta función motivaría la participación e interacción de los usuarios en la plataforma.

- Grupos temáticos o foros de debate: Permitir la formación de grupos o foros temáticos específicos permitiría a los usuarios congregarse en función de intereses compartidos. Esto facilitaría el intercambio específico de opiniones y conocimientos, fomentando subcomunidades dentro de la red social más amplia.

- Integración de los medios sociales: Facilitar el intercambio de contenidos de Evaluómetro en otras plataformas de medios sociales, y viceversa, podría aumentar significativamente la visibilidad y el alcance de la plataforma.

- Programa de reconocimiento: Instaurar un sistema de recompensas o insignias por la participación activa, por ejemplo, por publicar comentarios útiles, alcanzar un determinado número de seguidores o realizar contribuciones constantes, podría incentivar la participación de los usuarios y mejorar la calidad de los contenidos.

- Análisis de sentimientos: Incorporar el análisis de sentimiento a los comentarios y valoraciones y mostrarlo en el desglose de los Posts o cuando se busca uno en concreto, proporcionaría una visión rápida del sentimiento dominante (positivo, negativo, neutro) en las opiniones sobre un producto o servicio.

- Función de preguntas y respuestas: Introducir una sección en la que los usuarios puedan plantear preguntas específicas sobre un producto o servicio y recibir respuestas de la comunidad fomentaría el intercambio de conocimientos y ayudaría a los usuarios a tomar decisiones bien informadas.

- Mejoras de accesibilidad y diseño adaptativo: Garantizar que la plataforma sea universalmente accesible, incluso para las personas con discapacidad, y que

responda óptimamente a una serie de dispositivos y tamaños de pantalla, sería una consideración crucial para el futuro.

Contemplando estas mejoras y evoluciones posteriores, Evaluómetro no sólo puede superar las limitaciones iniciales, sino también convertirse en una red social más extensa, dinámica y valiosa. Con un énfasis en la creación de comunidades, el intercambio de conocimientos y la mejora continua, Evaluómetro tiene el potencial de posicionarse como una plataforma líder para la evaluación de productos y servicios.

Bibliografía

[1] ¡La audiencia publicitaria de TikTok llega a 1,002 millones! (Y otras estadísticas impactantes).
https://blog.hootsuite.com/es/informe-digital-estadisticas-de-redes-sociales/
Simon Kemp (agosto 2022)

[2] Todo Lo Que Necesitas Saber Sobre Las Redes Sociales.
https://www.rdstation.com/es/redes-sociales/
RD Station / 2023

[3] What is IMDb?
https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref_=helpsect_cons_1_1#
Marzo de 2023

[4] About Rotten Tomatoes
https://www.rottentomatoes.com/about
Marzo de 2023

[5] Our Mission
https://www.consumerreports.org/cro/about-us/what-we-do/index.htm
Marzo de 2023

[6] CNET tells you what's new and why it matters.
https://www.cnet.com/about/
Marzo de 2023

[7] Información sobre Tripadvisor
https://tripadvisor.mediaroom.com/es-about-us
Marzo de 2023

[8] Who We Are
https://www.aboutamazon.com/about-us
Marzo de 2023

[9] Overview of ASP.NET Core MVC
https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0
Mayo de 2023

[10] Why VS Code?
https://code.visualstudio.com/learn
Marzo de 2023

[11] Extensión VS Code: Django
https://marketplace.visualstudio.com/items?itemName=batisteo.vscode-django
Febrero de 2022

[12] Extensión VS Code: GitHub Pull Requests and Issues
https://marketplace.visualstudio.com/items?itemName=GitHub.vscode-pull-request-github
Mayo de 2023

[13] Extensión VS Code: Python
https://marketplace.visualstudio.com/items?itemName=ms-python.python
Mayo de 2023

[14] Extensión VS Code: SQLite3 Editor
https://marketplace.visualstudio.com/items?itemName=alexcvzz.vscode-sqlite
Junio de 2022

[15] The Python Tutorial
https://docs.python.org/3/tutorial/index.html
Mayo de 2023

[16] Documentation
https://docs.djangoproject.com/en/3.1/intro/overview/
Marzo de 2023

[17] About SQLite
https://www.sqlite.org/about.html
Marzo de 2023

[18] HTML: HyperText Markup Language
https://developer.mozilla.org/en-US/docs/Web/HTML
Mayo de 2023

[19] CSS: Cascading Style Sheets
https://developer.mozilla.org/en-US/docs/Web/CSS
Abril de 2023

[20] Get started with Tailwind CSS
https://tailwindcss.com/docs/installation
Junio de 2023

[21] JavaScript Guide
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
Mayo de 2023

[22] Ajax
https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX
Mayo de 2023

[23] AJAX Introduction
https://www.w3schools.com/xml/ajax_intro.asp
Mayo de 2023

[24] What is jQuery?
https://jquery.com/
Marzo de 2023

[25] Ciclo de vida iterativo
https://alredsa.blogspot.com/2017/07/ciclo-de-vida-de-proyectos-clasico.html
Julio de 2017

Anexo I
Casos de Uso

En este anexo se presentan las tablas/plantillas descriptivas de todos los casos de la aplicación.

Tabla AI.1. - Casos de uso. Registrarse.

C.U. 1	**Registrarse**
Actores	Usuario registrado
Descripción	El usuario se registra para poder acceder a la web
Dependencias	Acceder a la web.
Precondición	El usuario tiene un correo electrónico
Secuencia normal	P1 - Accede a la web. P2 - Le da al botón de registrarse. P3 - Introduce sus credenciales y se registra.
Poscondición	El correo debe existir y el nombre de usuario no debe existir ya.
Excepciones	- Si introduce mal un parámetro no le dejará registrarse y le dará un aviso de lo que debe cambiar.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.2. - Casos de uso. Iniciar Sesión.

C.U. 2	**Iniciar Sesión**
Actores	Usuario registrado
Descripción	El usuario se inicia sesión para poder acceder a la web
Dependencias	C.U. 1
Precondición	El usuario tiene una cuenta.
Secuencia normal	P1 - Accede a la web. P2 - Pone sus credenciales. P3 - Inicia sesión.
Poscondición	El usuario accede a la web.
Excepciones	- Si introduce mal un parámetro no le dejará iniciar sesión y le dará un aviso de lo que debe cambiar.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.3. - Casos de uso. Cambiar contraseña.

C.U. 3	**Cambiar contraseña**
Actores	Administrador (Principal)
Descripción	El administrador gestiona los posts en la red social.
Dependencias	C.U. 1
Precondición	El administrador ha iniciado sesión en el sistema.
Secuencia normal	P1 - El administrador selecciona la opción "Gestionar posts". P2 - El sistema muestra una lista de todos los posts. P3 - El administrador puede seleccionar un post para ver detalles o realizar operaciones (crear, modificar, eliminar).
Poscondición	Posts gestionados según las operaciones seleccionadas.
Excepciones	- Si en el paso P2 no hay posts, el sistema muestra un mensaje de "No hay posts disponibles". - Si en el paso P3 el administrador selecciona un post que ya ha sido eliminado, el sistema muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.4. - Casos de uso. Recuperar contraseña.

C.U. 4	**Recuperar contraseña**
Actores	Administrador (Principal)
Descripción	El administrador gestiona los posts en la red social.
Dependencias	C.U. 1
Precondición	El administrador ha iniciado sesión en el sistema.
Secuencia normal	P1 - El administrador selecciona la opción "Gestionar posts". P2 - El sistema muestra una lista de todos los posts. P3 - El administrador puede seleccionar un post para ver detalles o realizar operaciones (crear, modificar, eliminar).
Poscondición	Posts gestionados según las operaciones seleccionadas.
Excepciones	- Si en el paso P2 no hay posts, el sistema muestra un mensaje de "No hay posts disponibles". - Si en el paso P3 el administrador selecciona un post que ya ha sido eliminado, el sistema muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.5. - Casos de uso. Gestionar Posts.

C.U. 5	**Gestionar posts**
Actores	Administrador (Principal)
Descripción	El administrador gestiona los posts en la red social.
Dependencias	C.U. 2
Precondición	El administrador ha iniciado sesión en el sistema.
Secuencia normal	P1 - El administrador selecciona la opción "Gestionar posts". P2 - El sistema muestra una lista de todos los posts. P3 - El administrador puede seleccionar un post para ver detalles o realizar operaciones (crear, modificar, eliminar).
Poscondición	Posts gestionados según las operaciones seleccionadas.
Excepciones	- Si en el paso P2 no hay posts, el sistema muestra un mensaje de "No hay posts disponibles". - Si en el paso P3 el administrador selecciona un post que ya ha sido eliminado, el sistema muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.6 - Roles de usuario. Crear posts.

C.U. 6	**Crear posts**
Actores	Administrador (Principal)
Descripción	El administrador crea un nuevo post en la red social.
Dependencias	C.U. 2, C.U. 5
Precondición	El administrador ha seleccionado la opción "Crear post" en el caso de uso "Gestionar posts".
Secuencia normal	P1 - El sistema presenta un formulario para la creación del post. P2 - El administrador llena la información requerida y presiona "Crear". P3 - El sistema valida la información y crea el post.
Poscondición	El nuevo post es visible en la red social.
Excepciones	- Si en el paso P2 el administrador omite algún campo requerido, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede crear el post debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.7. - Roles de usuario. Eliminar posts.

C.U. 7	**Eliminar posts**
Actores	Administrador (Principal)
Descripción	El administrador elimina un post existente en la red social.
Dependencias	C.U. 2, C.U. 5
Precondición	El administrador ha seleccionado un post y la opción "Eliminar" en el caso de uso "Gestionar posts".
Secuencia normal	P1 - El sistema muestra una ventana de confirmación. P2 - El administrador confirma la eliminación. P3 - El sistema elimina el post.
Poscondición	El post ya no es visible en la red social.
Excepciones	- Si en el paso P3 el sistema no puede eliminar el post debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Baja
Importancia	Alta
Urgencia	Baja
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.8. - Roles de usuario. Modificar posts.

C.U. 8	**Modificar posts**
Actores	Administrador (Principal)
Descripción	El administrador modifica un post existente en la red social.
Dependencias	C.U. 2, C.U. 5
Precondición	El administrador ha seleccionado un post y la opción "Modificar" en el caso de uso "Gestionar posts".
Secuencia normal	P1 - El sistema presenta un formulario con la información actual del post. P2 - El administrador modifica la información y presiona "Guardar cambios". P3 - El sistema valida la información y actualiza el post.
Poscondición	La información modificada del post es visible en la red social.
Excepciones	- Si en el paso P2 el administrador omite algún campo requerido, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede actualizar el post debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.9. - Roles de usuario. Gestionar usuarios.

C.U. 9	**Gestionar usuarios**
Actores	Administrador (Principal)
Descripción	El administrador gestiona los usuarios de la red social.
Dependencias	Ninguna
Precondición	El administrador ha seleccionado la opción "Gestionar usuarios" en el panel de administración.
Secuencia normal	P1 - El sistema muestra una lista de usuarios. P2 - El administrador selecciona un usuario. P3 - El sistema muestra las opciones de gestión para el usuario seleccionado.
Poscondición	Se muestra la información del usuario seleccionado y las opciones de gestión disponibles.
Excepciones	- Si en el paso P2 el sistema no puede cargar la lista de usuarios, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.10. - Roles de usuario. Crear usuarios.

C.U. 10	**Crear usuarios**
Actores	Administrador (Principal)
Descripción	El administrador crea un nuevo usuario en la red social.
Dependencias	C.U. 9
Precondición	El administrador ha seleccionado la opción "Crear usuario" en el caso de uso "Gestionar usuarios".
Secuencia normal	P1 - El sistema presenta un formulario para ingresar la información del nuevo usuario. P2 - El administrador introduce la información y presiona "Crear". P3 - El sistema valida la información y crea el nuevo usuario.
Poscondición	El nuevo usuario es visible en la lista de usuarios.
Excepciones	- Si en el paso P2 el administrador omite algún campo requerido, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede crear el usuario debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.11. - Roles de usuario. Eliminar usuarios.

C.U. 11	**Eliminar usuarios**
Actores	Administrador (Principal)
Descripción	El administrador elimina un usuario de la red social.
Dependencias	C.U. 9
Precondición	El administrador ha seleccionado la opción "Eliminar usuario" en el caso de uso "Gestionar usuarios".
Secuencia normal	P1 - El sistema pide confirmación para eliminar el usuario seleccionado. P2 - El administrador confirma. P3 - El sistema elimina el usuario.
Poscondición	El usuario ya no es visible en la lista de usuarios.
Excepciones	- Si en el paso P3 el sistema no puede eliminar el usuario debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Baja
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.12. - Roles de usuario. Modificar usuarios.

C.U. 12	**Modificar usuarios**
Actores	Administrador (Principal)
Descripción	El administrador modifica la información de un usuario en la red social.
Dependencias	C.U. 9
Precondición	El administrador ha seleccionado la opción "Modificar usuario" en el caso de uso "Gestionar usuarios".
Secuencia normal	P1 - El sistema presenta un formulario con la información actual del usuario. P2 - El administrador modifica la información y presiona "Guardar cambios". P3 - El sistema valida la información y actualiza el usuario.
Poscondición	La información del usuario se actualiza en la lista de usuarios.
Excepciones	- Si en el paso P2 el administrador omite algún campo requerido, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede actualizar el usuario debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.13. - Roles de usuario. Gestionar comentarios de posts.

C.U. 13	**Gestionar comentarios de posts**
Actores	Administrador (Principal)
Descripción	El administrador gestiona los comentarios de los posts en la red social.
Dependencias	C.U. 2, C.U. 5
Precondición	El administrador ha seleccionado la opción "Gestionar comentarios" en el caso de uso "Gestionar posts".
Secuencia normal	P1 - El sistema muestra una lista de comentarios. P2 - El administrador selecciona un comentario. P3 - El sistema muestra las opciones de gestión para el comentario seleccionado.
Poscondición	Se muestra la información del comentario seleccionado y las opciones de gestión disponibles.
Excepciones	- Si en el paso P2 el sistema no puede cargar la lista de comentarios, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.14. - Roles de usuario. Crear comentarios de posts.

C.U. 14	**Crear comentarios de posts**
Actores	Administrador (Principal)
Descripción	El administrador crea un nuevo comentario en un post de la red social.
Dependencias	C.U. 2, C.U. 13
Precondición	El administrador ha seleccionado la opción "Crear comentario" en el caso de uso "Gestionar comentarios de posts".
Secuencia normal	P1 - El sistema presenta un formulario para ingresar el nuevo comentario. P2 - El administrador introduce el comentario y presiona "Crear". P3 - El sistema valida el comentario y lo publica.
Poscondición	El nuevo comentario es visible en la lista de comentarios del post.
Excepciones	- Si en el paso P2 el administrador omite el campo del comentario, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede publicar el comentario debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.15. - Roles de usuario. Eliminar comentarios de posts.

C.U. 15	**Eliminar comentarios de posts**
Actores	Administrador (Principal)
Descripción	El administrador elimina un comentario de un post de la red social.
Dependencias	C.U. 2, C.U. 13
Precondición	El administrador ha seleccionado la opción "Eliminar comentario" en el caso de uso "Gestionar comentarios de posts".
Secuencia normal	P1 - El sistema pide confirmación para eliminar el comentario seleccionado. P2 - El administrador confirma. P3 - El sistema elimina el comentario.
Poscondición	El comentario ya no es visible en la lista de comentarios del post.
Excepciones	- Si en el paso P3 el sistema no puede eliminar el comentario debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Baja
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.16. - Roles de usuario. Modificar comentarios de posts.

C.U. 16	**Modificar comentarios de posts**
Actores	Administrador (Principal)
Descripción	El administrador modifica un comentario de un post de la red social.
Dependencias	C.U. 2, C.U. 13
Precondición	El administrador ha seleccionado la opción "Modificar comentario" en el caso de uso "Gestionar comentarios de posts".
Secuencia normal	P1 - El sistema presenta un formulario con el comentario actual. P2 - El administrador modifica el comentario y presiona "Guardar cambios". P3 - El sistema valida el comentario y lo actualiza.
Poscondición	El comentario se actualiza en la lista de comentarios del post.
Excepciones	- Si en el paso P2 el administrador omite el campo del comentario, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede actualizar el comentario debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.17. - Roles de usuario. Gestionar evaluaciones de posts.

C.U. 17	**Gestionar evaluaciones de posts**
Actores	Administrador (Principal)
Descripción	El administrador gestiona las evaluaciones de los posts en la red social.
Dependencias	C.U. 1
Precondición	El administrador ha seleccionado la opción "Gestionar evaluaciones" en el caso de uso "Gestionar posts".
Secuencia normal	P1 - El sistema muestra una lista de evaluaciones. P2 - El administrador selecciona una evaluación. P3 - El sistema muestra las opciones de gestión para la evaluación seleccionada.
Poscondición	Se muestra la información de la evaluación seleccionada y las opciones de gestión disponibles.
Excepciones	- Si en el paso P2 el sistema no puede cargar la lista de evaluaciones, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.18. - Roles de usuario. Crear evaluaciones de posts.

C.U. 18	**Crear evaluaciones de posts**
Actores	Administrador (Principal)
Descripción	El administrador crea una nueva evaluación para un post en la red social.
Dependencias	C.U. 2, C.U. 13
Precondición	El administrador ha seleccionado la opción "Crear evaluación" en el caso de uso "Gestionar evaluaciones de posts".
Secuencia normal	P1 - El sistema presenta un formulario para ingresar la nueva evaluación. P2 - El administrador introduce la evaluación y presiona "Crear". P3 - El sistema valida la evaluación y la publica.
Poscondición	La nueva evaluación es visible en la lista de evaluaciones del post.
Excepciones	- Si en el paso P2 el administrador omite algún campo requerido, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede publicar la evaluación debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.19. - Roles de usuario. Eliminar evaluaciones de posts.

C.U. 19	**Eliminar evaluaciones de posts**
Actores	Administrador (Principal)
Descripción	El administrador elimina una evaluación de un post de la red social.
Dependencias	C.U. 2, C.U. 13
Precondición	El administrador ha seleccionado la opción "Eliminar evaluación" en el caso de uso "Gestionar evaluaciones de posts".
Secuencia normal	P1 - El sistema pide confirmación para eliminar la evaluación seleccionada. P2 - El administrador confirma. P3 - El sistema elimina la evaluación.
Poscondición	La evaluación ya no es visible en la lista de evaluaciones del post.
Excepciones	- Si en el paso P3 el sistema no puede eliminar la evaluación debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Baja
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.20. - Roles de usuario. Modificar evaluaciones de posts.

C.U. 20	**Modificar evaluaciones de posts**
Actores	Administrador (Principal)
Descripción	El administrador modifica una evaluación de un post de la red social.
Dependencias	C.U. 2, C.U. 13
Precondición	El administrador ha seleccionado la opción "Modificar evaluación" en el caso de uso "Gestionar evaluaciones de posts".
Secuencia normal	P1 - El sistema presenta un formulario con la evaluación actual. P2 - El administrador modifica la evaluación y presiona "Guardar cambios". P3 - El sistema valida la evaluación y la actualiza.
Poscondición	La evaluación se actualiza en la lista de evaluaciones del post.
Excepciones	- Si en el paso P2 el administrador omite algún campo requerido, el sistema muestra un mensaje de error. - Si en el paso P3 el sistema no puede actualizar la evaluación debido a un error, muestra un mensaje de error.
Rendimiento	N/A
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	-

Tabla AI.21. - Roles de usuario. Visitar posts de otros usuarios.

C.U.21	**Visitar posts de otros usuarios.**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario visita posts de otros usuarios.
Dependencias	C.U. 2
Precondición	Los posts deben existir.
Secuencia normal	P1 - El usuario selecciona el post a visitar. P2 - El sistema muestra el post seleccionado.
Poscondición	El post ha sido visualizado.
Excepciones	- Si en el paso P1 el post no existe, el sistema muestra un mensaje de error.
Rendimiento	No aplicable.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.22. - Roles de usuario. Comentar en un post.

C.U. 22	**Comentar en un post**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario comenta en el post de otro usuario.
Dependencias	C.U. 2, C.U. 21
Precondición	El post debe permitir comentarios.
Secuencia normal	P1 - El usuario escribe un comentario. P2 - El usuario envía el comentario. P3 - El sistema muestra el comentario en el post.
Poscondición	El comentario es visible en el post.
Excepciones	- Si en el paso P2 el comentario es inapropiado, el sistema muestra un mensaje de error y no lo publica.
Rendimiento	No aplicable.
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.23. - Roles de usuario. Evaluar un post.

C.U. 23	**Evaluar un post**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario evalúa un post de otro usuario.
Dependencias	C.U. 2, C.U. 21
Precondición	El post debe permitir evaluaciones.
Secuencia normal	P1 - El usuario selecciona la evaluación. P2 - El usuario envía la evaluación. P3 - El sistema muestra la evaluación en el post.
Poscondición	La evaluación es visible en el post.
Excepciones	- Si en el paso P2 el usuario intenta enviar múltiples evaluaciones, el sistema sólo permite la última evaluación.
Rendimiento	No aplicable.
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.24. - Roles de usuario. Comentar un comentario en un post.

C.U. 24	**Comentar un comentario en un post**
Actores	Usuario Registrado (Actor principal)
Descripción	C.U. 2, C.U. 21
Dependencias	El usuario debe estar registrado y autenticado, y debe haber visitado un post.
Precondición	El comentario debe permitir respuestas.
Secuencia normal	P1 - El usuario selecciona un comentario. P2 - El usuario escribe su respuesta. P3 - El usuario envía su respuesta. P4 - El sistema muestra la respuesta en el comentario.
Poscondición	La respuesta es visible en el comentario.
Excepciones	- Si en el paso P2 la respuesta es inapropiada, el sistema muestra un mensaje de error y no la publica.
Rendimiento	No aplicable.
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.25. - Roles de usuario. Visitar sus propios posts.

C.U. 25	**Visitar sus propios posts**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario visita sus propios posts.
Dependencias	C.U. 2
Precondición	El usuario debe tener al menos un post publicado.
Secuencia normal	P1 - El usuario selecciona ver sus propios posts. P2 - El sistema muestra los posts del usuario.
Poscondición	Los posts del usuario son visibles para él.
Excepciones	- Si en el paso P1 el usuario no tiene posts, el sistema muestra un mensaje indicando que no hay posts.
Rendimiento	No aplicable.
Frecuencia	Alta
Importancia	Alta
Urgencia	Baja
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.26. - Roles de usuario. Comentar su propio post.

C.U. 26	**Comentar su propio post**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario comenta en su propio post.
Dependencias	C.U. 2, C.U. 22
Precondición	No aplicable.
Secuencia normal	P1 - El usuario escribe su comentario. P2 - El usuario envía su comentario. P3 - El sistema muestra el comentario en el post.
Poscondición	El comentario es visible en el post.
Excepciones	- Si en el paso P2 el comentario es inapropiado, el sistema muestra un mensaje de error y no lo publica.
Rendimiento	No aplicable.
Frecuencia	Alta
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.27. - Roles de usuario. Evaluar su propio post.

C.U. 27	**Evaluar su propio post**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario evalúa su propio post.
Dependencias	C.U. 2, C.U. 22
Precondición	No aplicable.
Secuencia normal	P1 - El usuario selecciona una calificación para su post. P2 - El sistema registra la calificación.
Poscondición	La calificación del post se actualiza.
Excepciones	- Si en el paso P2 ocurre un error, el sistema muestra un mensaje de error y la calificación no se actualiza.
Rendimiento	No aplicable.
Frecuencia	Media
Importancia	Media
Urgencia	Baja
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.28. - Roles de usuario. Comentar comentarios de su propio post.

C.U. 28	**Comentar comentarios de su propio post**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario comenta en los comentarios de su propio post.
Dependencias	C.U. 2, C.U. 22
Precondición	Debe haber al menos un comentario en el post.
Secuencia normal	P1 - El usuario selecciona un comentario. P2 - El usuario escribe su respuesta al comentario. P3 - El usuario envía su comentario. P4 - El sistema muestra el comentario en respuesta al comentario seleccionado.
Poscondición	El comentario es visible en el post.
Excepciones	- Si en el paso P3 el comentario es inapropiado, el sistema muestra un mensaje de error y no lo publica.
Rendimiento	No aplicable.
Frecuencia	Media
Importancia	Media
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.29. - Roles de usuario. Editar su propio post.

C.U. 29	**Editar su propio post**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario edita su propio post.
Dependencias	C.U. 2, C.U. 22
Precondición	No aplicable.
Secuencia normal	P1 - El usuario selecciona la opción para editar el post. P2 - El usuario realiza los cambios en el contenido del post. P3 - El usuario guarda los cambios. P4 - El sistema muestra el post actualizado.
Poscondición	El post se actualiza con los nuevos cambios.
Excepciones	- Si en el paso P3 el sistema no puede guardar los cambios, muestra un mensaje de error y no se realizan cambios en el post.
Rendimiento	No aplicable.
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.30. - Roles de usuario. Eliminar su propio post.

C.U. 30	**Eliminar su propio post**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario elimina su propio post.
Dependencias	C.U. 2, C.U. 22
Precondición	No aplicable.
Secuencia normal	P1 - El usuario selecciona la opción para eliminar el post. P2 - El sistema pide confirmación para eliminar el post. P3 - El usuario confirma la eliminación del post. P4 - El sistema elimina el post.
Poscondición	El post se elimina del sistema.
Excepciones	- Si en el paso P3 el usuario cancela la eliminación, el sistema no elimina el post.
Rendimiento	No aplicable.
Frecuencia	Baja
Importancia	Alta
Urgencia	Baja
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.31. - Roles de usuario. Buscar posts.

C.U. 31	**Buscar posts**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario busca posts.
Dependencias	C.U. 2
Precondición	No aplicable.
Secuencia normal	P1 - El usuario ingresa el término de búsqueda en el campo de búsqueda. P2 - El sistema muestra los posts que coinciden con el término de búsqueda.
Poscondición	No aplicable.
Excepciones	- Si en el paso P2 no se encuentran posts que coincidan con el término de búsqueda, el sistema muestra un mensaje informando que no se encontraron resultados.
Rendimiento	No aplicable.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.32. - Roles de usuario. Filtrar posts.

C.U. 32	**Filtrar posts**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario filtra posts.
Dependencias	C.U. 2
Precondición	No aplicable.
Secuencia normal	P1 - El usuario selecciona los filtros que desea aplicar. P2 - El sistema muestra los posts que cumplen con los filtros seleccionados.
Poscondición	No aplicable.
Excepciones	- Si en el paso P2 no se encuentran posts que cumplen con los filtros seleccionados, el sistema muestra un mensaje informando que no se encontraron resultados.
Rendimiento	No aplicable.
Frecuencia	Alta
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.33. - Roles de usuario. Buscar Usuarios.

C.U. 33	**Buscar Usuarios**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario busca otros usuarios.
Dependencias	C.U. 2
Precondición	No aplicable.
Secuencia normal	P1 - El usuario ingresa el nombre del usuario en el campo de búsqueda. P2 - El sistema muestra los usuarios que coinciden con el nombre ingresado.
Poscondición	No aplicable.
Excepciones	- Si en el paso P2 no se encuentran usuarios que coinciden con el nombre ingresado, el sistema muestra un mensaje informando que no se encontraron resultados.
Rendimiento	No aplicable.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Hecho
Estabilidad	Alta
Comentarios	

Tabla AI.34. - Roles de usuario. Seguir Usuarios.

C.U. 34	**Seguir Usuarios**
Actores	Usuario Registrado (Actor principal)
Descripción	El usuario sigue a otro usuario.
Dependencias	C.U. 2
Precondición	No aplicable.
Secuencia normal	P1 - El usuario selecciona el botón de "seguir" en el perfil del usuario que desea seguir. P2 - El sistema confirma la acción y comienza a mostrar en el feed del usuario los posts del usuario seguido.
Poscondición	El usuario ahora sigue al usuario seleccionado y ve sus posts en su feed.
Excepciones	- Si en el paso P2 hay algún error con el sistema, se muestra un mensaje de error y la acción no se completa.
Rendimiento	No aplicable.
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Estado	Hecho
Estabilidad	Alta
Comentarios	