

PROYECTO

SeriesBuddies

CICLO FORMATIVO DE GRADO SUPERIOR:

Desarrollo de Aplicaciones Web

AUTORES:

Román Kornyeyev

Ana Isabel Pedrajas Navarro

Licencia

Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons.



Para ver una copia de esta licencia:

- Visite: <http://creativecommons.org/licenses/by-sa/3.0/es/>
- O envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

AGRADECIMIENTOS

Anabel

A mi familia y a Álvaro por el apoyo y los ánimos recibidos en estos meses y Román por las ganas y el esfuerzo que ha dedicado al realizar este proyecto juntos, gracias por ser un buen amigo y compañero.

Román

A mi familia, a los amigos que me han escuchado y apoyado estos meses y a los compañeros que contribuyeron a que este viaje fuese mucho más sencillo. Y por supuesto a Anabel por haber hecho este proyecto mucho más ameno.

RESUMEN

Con este proyecto de fin de grado (Grado Superior de Desarrollo de Aplicaciones Web) se quiere demostrar los conocimientos adquiridos a lo largo de los dos años de curso y la experiencia obtenida durante el periodo de prácticas en la empresa.

Hoy en día, en pleno 2023 existen muchos foros, blogs de opinión, etc. Pero no existe ningún sitio web especializado en compartir ideas respecto a series. Asimismo, hay mucha gente a la que le gustaría compartir ideas, opiniones y gustos sobre series en un sitio web y encontrar a otras personas con los mismos gustos (o similares) con los que poder conversar, debatir y compartir opiniones. Y es aquí donde entra SeriesBuddies.

Nuestra propuesta de proyecto consiste en realizar una aplicación web llamada SeriesBuddies. Este sitio web permitirá a los aficionados de las series, no solo llevar un registro del contenido que ven en diferentes plataformas de streaming, sino de poder interactuar con otros usuarios a partir del diálogo y el intercambio de opiniones sobre las mismas. En un primer momento, la plataforma está enfocada hacia el público joven (rango de edad entre los 18-25 años), aunque será accesible para todas las edades.

Con respecto a los stack tecnológicos utilizados para llevar a cabo SeriesBuddies, en el apartado front-end se han empleado las siguientes tecnologías: HTML, CSS (vanilla) y JavaScript (vanilla). Mientras que, en el lado back-end se ha optado por emplear PHP (vanilla) y MariaDB (MySQL). El proyecto se desplegará en un servidor Linux VPS, usando apache.

ABSTRACT

With our final degree project (Higher Degree in Web Application Development), we aim to demonstrate the knowledge acquired throughout the two-year course, as well as the new skills obtained during our internship period at the company.

At present, there are many forums, opinion blogs, etc. However, there is no specialized website for sharing ideas about TV series. There are many people who would like to do so and have the opportunity to share ideas, opinions, and preferences about different series on a website. This way they could connect with others who have similar or shared interests, opening the door to discussion on various items related. This is where SeriesBuddies comes in.

Our proposed project is to develop a web application called SeriesBuddies. This website will allow series enthusiasts not only to keep a record of the content they watch on different streaming platforms but also to interact with other users through dialogue and the exchange of opinions about series. Initially, the platform will primarily target a younger audience (aged between 18-25), although it will be accessible to all age groups.

To develop SeriesBuddies on the front-end, the following technologies have been employed: HTML, CSS (vanilla), and JavaScript (vanilla). For the back-end, we have chosen to use: PHP (vanilla) and MariaDB. The project will be deployed on a Linux VPS server, using Apache.

Índice de contenido

1	INTRODUCCIÓN	9
2	NECESIDADES DEL SECTOR PRODUCTIVO.....	12
2.1	ANÁLISIS DE LA SITUACIÓN ACTUAL	12
2.2	NECESIDADES DEL CLIENTE Y OPORTUNIDAD DE NEGOCIO	13
2.3	EL NUEVO PROYECTO: SERIESBUDDIES	15
2.3.1	TIPO DE PROYECTO.....	15
2.3.2	CARACTERÍSTICAS REQUERIDAS AL PROYECTO.....	17
2.3.3	OBLIGACIONES FISCALES, LABORALES Y DE PREVENCIÓN DE RIESGO	20
2.3.4	AYUDAS/SUBVENCIONES	21
3	DISEÑO DEL PROYECTO	23
3.1	FASES DEL PROYECTO	23
3.1.1	ANÁLISIS	23
3.1.2	DISEÑO.....	24
3.1.3	IMPLEMENTACIÓN	43
3.1.4	PRUEBAS.....	73
3.2	OBJETIVOS QUE CONSEGUIR.....	78
3.3	PREVISIÓN DE LOS RECURSOS MATERIALES Y HUMANOS NECESARIOS	80
3.4	PRESUPUESTO ECONÓMICO.....	80
4	PLANIFICACIÓN DE LA EJECUCIÓN DEL PROYECTO.....	82
4.1	FASE DE ANÁLISIS	82
4.2	FASE DE DISEÑO.....	83
4.3	FASE DE IMPLEMENTACIÓN.....	83
4.4	FASE DE PRUEBAS.....	84
5	DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN.....	85
6	FUENTES.....	86

Índice de figuras

Figura 1 - Logo de git y github	19
Figura 2 - Logotipo de Figma.....	20
Figura 3 - Logotipo de Visual Studio Code	20
Figura 4 - Estructura layout del proyecto	24
Figura 5 - Logotipo Figma.....	25
Figura 6 - Wireframe completo del proyecto.....	25
Figura 7 - Bocetos de logos del proyecto	26
Figura 8 - Logo final del proyecto (móvil)	27
Figura 9 - Isologotipo final del proyecto (PC).....	27
Figura 10 - Colores primarios	28
Figura 11 - Colores secundarios.....	28
Figura 12 - Colores de fondo	28
Figura 13 - Mockup completo del proyecto.....	29
Figura 14 – pantalla de index	30
Figura 15 - Pantalla de registro	31
Figura 16 - Pantalla de inicio de sesión.....	32
Figura 17 - Pantalla recuperación de contraseña	33
Figura 18 - Pantalla de verificación	34
Figura 19 - Pantalla de géneros	35
Figura 20 - Pantalla de series.....	36
Figura 21 - Pantalla de feed	37
Figura 22 - Pantalla de buddies (usuarios).....	38
Figura 23 - Pantalla de perfil.....	39
Figura 24 - Pantalla de contacto.....	40
Figura 25 - Estructura de correos automatizados.....	41
Figura 26 - Modelo E/R de la base de datos	42
Figura 27 - Tabla usuarios.....	43
Figura 28 - Tabla peticiones	44
Figura 29 - Tabla tokens.....	45
Figura 30 - Evento eliminar_tokens_expirados.....	46
Figura 31 - Tabla respuestas.....	46
Figura 32 - Tabla chips	47
Figura 33 - Tabla chips_usuario	47
Figura 34 - Trigger insertar_medalla_inicio	48
Figura 35 - Estructura del proyecto	51
Figura 36 - Carpeta clases	52
Figura 37 - Carpeta documentacion	52
Figura 38 - Carpeta public	52
Figura 39 - Carpeta scripts	52
Figura 40 - Carpeta src.....	53
Figura 41 - Carpeta vendor.....	53
Figura 42 - Esquema UML de la librería de formularios	54
Figura 43 - Instanciación de un objeto Formulario.....	55
Figura 44 - Formulario de login.....	55
Figura 45 - Atributos clase DWESBaseDatos.....	56
Figura 46 - Constantes públicas clase DWESBaseDatos	56
Figura 47 - Función inicializa de DWESBaseDatos	57
Figura 48 - Función ejecuta de DWESBaseDatos	57
Figura 49 - Función obtenInfoBuddieTarjeta de DWESBaseDatos	58

Figura 50 - insertarUsuario de DWEBaseDatos	58
Figura 51 - Función actualizarRespuesta de DWESBaseDatos	59
Figura 52 - Función eliminaTokensUsuario de DWESBaseDatos	59
Figura 53 - Función obtenLimitesPaginacion de DWESBaseDatos	60
Figura 54 - Función obtenPaginacion de DWESBaseDatos.....	60
Figura 55 - Constantes de la clase clase TMDB.....	61
Figura 56 - Función peticionHTTP de TMDB.....	61
Figura 57 - Funciones url de TMDB.....	62
Figura 58 - Función getSeriesID de TMDB.....	62
Figura 59 - Atributos, constantes y constructor clase Peticion	63
Figura 60 - Función pintaSesionNoIniciada de Peticion	63
Figura 61 - Card footer	64
Figura 62 - Función peticion (JS).....	64
Figura 63 - peticiones_handler.php	65
Figura 64 - Tarjeta buddy (sin petición)	65
Figura 65 - Tarjeta buddy (petición enviada)	65
Figura 66 - Tarjeta buddy (petición recibida)	66
Figura 67 – Peticiones recibidas en el perfil	66
Figura 68 - Clase Mailer	67
Figura 69 - Función pintaEmailVerificacion de Mailer.....	67
Figura 70 - Layout de las páginas	68
Figura 71 - Template	68
Figura 72 - Lógica de las páginas.....	69
Figura 73 - Buffer y llamada al template	69
Figura 74 - Archivo 000-default.conf.....	71
Figura 75 - Dominio	72
Figura 76 - Datos de registro	73
Figura 77 - Ruta enlace correo	74
Figura 78 - Datos de login	74
Figura 79 - Cookies	74
Figura 80 - Función getSeriesNombre de TMDB.....	75
Figura 81 - Resultados por búsqueda de series	75
Figura 82 - Proceso de obtención del chip bronce	76
Figura 83 - Proceso de eliminación de chip de bronce	77
Figura 84 - Coste del dominio y servidor VPS	81
Figura 85 - Diagrama de Gantt análisis	82
Figura 86 - Diagrama de Gantt diseño	83
Figura 87 - Diagrama de Gantt implementación	84
Figura 88 - Diagrama de Gantt pruebas	84

1 INTRODUCCIÓN

Este documento responde a la realización del módulo de Proyecto del CFGS en Desarrollo de Aplicaciones Web. El módulo de Proyecto complementa, la formación establecida para el resto de los módulos profesionales que integran el título en las funciones de análisis del contexto, diseño del proyecto y organización de la ejecución.

Con SeriesBuddies se pretende habilitar un espacio seguro en donde los aficionados de las series puedan dejar comentarios sobre las series que ven o han visto y conocer a otros usuarios, en este caso buddies, que tengan gustos similares. Este proyecto también aspira a convertirse en una muestra de la formación obtenida en el Ciclo Formativo que se ha cursado y de las habilidades que se han adquirido con tecnologías como PHP, MySQL o CSS y JavaScript.

La interconexión de los diferentes módulos estudiados se ha visto más claramente ejemplificada gracias al desarrollo de SeriesBuddies y en las diferentes fases de desarrollo y diseño vividos y que se desglosan a continuación:

El temario impartido en la asignatura Empresa e Iniciativa Emprendedora ha servido como base para:

- Realizar un correcto análisis y detección de la situación y las necesidades de los internautas de un determinado rango de edad (18 - 25 años).
- Definir la idea final del producto que se desea publicar, tras la generación de una lluvia de ideas con diversas propuestas viables y el estudio de los factores presentes en el macroentorno y microentorno que afectan al proyecto.
- Saber escoger el tipo de empresa que mejor se adapta a SeriesBuddies en función de las necesidades económicas y la viabilidad de desarrollo.

Las nociones de diseño aprendidas en los módulos Lenguaje de marcas y sistemas de gestión de información y Diseño de interfaces web se han podido aplicar para dotar al sitio web de:

- Una interfaz accesible e intuitiva. El objetivo principal siempre ha consistido en permitir que el usuario pueda hacer uso de SeriesBuddies de una manera sencilla, con una navegación clara y bien definida.
- Una gran capacidad de adaptación a todo tipo de dispositivos con los que poder acceder a SeriesBuddies a través del diseño responsive.

- Un diseño moderno, minimalista e innovador con el que poder diferenciar a SeriesBuddies de otras plataformas, pero guardando aquellos elementos estéticos y funcionales que permiten que sea identificada como un sitio web relacionado con el mundo de las series y, en consecuencia, de las plataformas de streaming que las alojan.

Los módulos de Programación, Desarrollo Web en entorno cliente y Desarrollo web en entorno servidor han permitido que durante el desarrollo del proyecto se tengan en cuenta las siguientes medidas:

- Establecer unas buenas prácticas a la hora de programar para obtener un código lo más estructurado, entendible y factorizado en la medida de lo posible.
- Documentar el código para una fácil lectura pensando no sólo en la supervisión de este durante las fases de desarrollo sino también para hacer más sencillo la revisión en caso de implementar mejoras o actualizaciones en un futuro.
- Diseñar y modularizar el proyecto para que las distintas partes que lo conforman puedan ser en cierta medida independiente y conserven su capacidad para fusionarse y complementarse en cualquier situación o necesidad requerida. Ej.: desarrollo de una librería especializada para la gestión de formularios, uso de una clase con sus respectivos métodos y variables para facilitar las peticiones a la API utilizada,...
- Realizar optimizaciones que agilicen los procesos de carga, mediante la utilización de peticiones asíncronas AJAX. A su vez usar dichas peticiones para evitar recargar la página entera en algunos casos y mejorar así, la experiencia del usuario.

Gracias a la asignatura de Base de Datos se han podido fortalecer y ampliar los conocimientos aprendidos de SQL e integrarlos en SeriesBuddies de la siguiente manera:

- Hacer un correcto diseño de la base de datos necesaria para poder desarrollar el sitio web, atendiendo a las necesidades que se han de cubrir y escogiendo los tipos de datos y relaciones que mejor se adaptan al proyecto.
- Preparar y configurar procedimientos y consultas optimizadas, que faciliten la obtención de la información y que puedan ser resueltas en el menor tiempo posible gracias al uso de JOINS en lugar de combinación de tablas/ subconsultas.

Finalmente, los módulos Entornos de Desarrollo, Sistemas informáticos y Despliegue de aplicaciones web han servido para:

- Hacer uso de un IDE específico (Visual Studio Code), así como de todos los complementos necesarios para la mayor comodidad posible y optimización del tiempo de desarrollo.
- Trabajar de forma colaborativa utilizando un sistema de control de versiones y ser capaces de solventar conflictos.
- Ser capaces de desplegar la aplicación configurando y usando un servidor VPS (Virtual Private Server) y un dominio propio (<https://www.seriesbuddies.es>), gracias a los conocimientos adquiridos en los módulos de Despliegue de aplicaciones web, Desarrollo web en entorno servidor y Sistemas Informáticos.

2 NECESIDADES DEL SECTOR PRODUCTIVO

A continuación, se identifican las necesidades detectadas en el sector productivo que originan la oportunidad de negocio que se detalla en los siguientes puntos.

2.1 *Análisis de la situación actual*

La tendencia de consumir contenido audiovisual (más concretamente series) ha crecido de forma constante en los últimos años, sin embargo, con la llegada de la pandemia y las medidas de aislamiento que se tuvieron que adoptar, el consumo de dicho contenido aumentó entre un 37% y un 41% en España durante los primeros meses del confinamiento.

Esta tendencia tras el paso de los años no ha disminuido. A inicios del año 2023, España ha ocupado el puesto número 28 en cuota de usuarios que consumen contenido televisivo por plataformas de streaming y la media que se emplea en plataformas como Netflix y Prime Video alcanza los 40 minutos al día en los consumidores más jóvenes. Además, otro dato importante para tener en cuenta es que el tiempo que se consume delante de una pantalla asciende aproximadamente a unas 4 y 6 horas diarias.

Tras la obtención de estos datos, se plantea las siguientes cuestiones ¿Existen sitios web o aplicaciones que permitan a estos consumidores expresar sus opiniones y compartir sus reacciones sobre las series que ven? ¿Son plataformas seguras y que estén especializadas o enfocadas en series y programas televisivos? ¿Permiten que cualquier usuario independientemente de su edad pueda hacer uso de esta aplicación de forma sencilla y se sienta integrado?

La información obtenida como respuesta a estas preguntas es la siguiente: sí, existen aplicaciones que permiten comentar episodios de series como TV Time o Letterboxd y plataformas web como FilmAffinity, Rotten Tomatoes e IMDb. Sin embargo, abarcan también la industria cinematográfica por lo que el contenido que muestran se multiplica.

Otras aplicaciones que se desarrollan en este sector son JustWatch, Trakt o Hobi, la diferencia principal con las anteriores y por las que merecen ser tratadas como grupo aparte es que la función que le ofrecen al usuario es la de llevar un registro de los episodios que tienen pendientes por ver, es decir, hacen la tarea de una agenda o calendario, pero no permiten comentar o dejar valoraciones sobre las series.

Finalmente, las opciones que existen actualmente en la web y que son capaces de responder en gran medida a las cuestiones planteadas anteriormente son NextEpisode y SeriesGuide. La primera se trata de un sitio web que ofrece información sobre los capítulos de las series disponibles, el reparto que la protagoniza y un calendario para contar con un seguimiento de las series, no obstante, al ser un servicio veterano, la interfaz y la navegación por la página no es clara y sencilla ya que muestra mucha información repartida a la vez. La segunda es una aplicación de código abierto que cuenta prácticamente con las mismas funciones que NextEpisode, pero está disponible únicamente para Android por lo que el segmento de consumidores y posibles clientes se limita.

Es importante destacar el hecho de que además de todas las plataformas mencionadas en el apartado anterior, existen redes sociales como TikTok y Twitter que permiten ser espacio de diálogo y promoción de series a través del uso de hashtags o palabras clave y en las cuales, el usuario puede encontrar y formar una comunidad más fácilmente.

2.2 *Necesidades del cliente y oportunidad de negocio*

Una vez realizado el estudio de mercado y analizado la situación actual en la que se encuentra el consumidor detectamos las siguientes necesidades:

A pesar de contar con dos espacios especializados en series de televisión, no son capaces de transmitir un diseño limpio y sencillo además de una navegación intuitiva que invite al consumidor a quedarse en la página y hacer un uso continuado de ella.

Por todo esto, las **necesidades que necesitan ser cubiertas** son las que se especifican a continuación:

- Configurar un espacio especializado únicamente en series que se pueden encontrar en el catálogo de la televisión y otras plataformas de streaming, con indiferencia del país en donde se produjo para así aportar un amplio abanico de resultados y evitar que el usuario no encuentre la serie que necesita.
- Diseñar un sitio web que contemple todas las posibilidades de acceso que pueden hacerse, esto es, que sea compatible con todos los sistemas operativos, navegadores y dispositivos y permitir que el rango de posibles usuarios sea grande.
- Desarrollar una plataforma que cuente con un mapa de navegación (menús, barras de navegación, herramientas de búsqueda y filtrados) que faciliten al usuario el uso de esta, evitar que no sepa ubicarse en todo momento en la web y que el acceso a los diferentes contenidos resulte fácil de recordar.

- Dotar al sitio de interactividad entre los usuarios mediante las peticiones de amistad con las que poder incentivar a crear comunidad ya sea por géneros o series determinadas y por la capacidad de dejar comentarios en cualquier serie.
- Permitir que los comentarios que se publiquen puedan ser modificados con posterioridad y eliminados, ofreciendo al usuario la capacidad de sentirse libre a la hora de gestionar el contenido que publica en la plataforma.
- Elaborar un diseño único, llamativo y diferenciador que permita que la aplicación pueda distinguirse de otras de tal forma que facilite que SeriesBuddies se convierta no sólo en un servicio, sino en una marca que se defina por sí misma. Esto se consigue a partir de la elección de una guía de estilos, paleta de color uniforme, creación de una marca e identidad visual, etc.

Con respecto al tipo de empresa seleccionado para dar forma a SeriesBuddies, se ha optado por elegir la sociedad de responsabilidad limitada (SL) siendo una empresa capitalista. Esta decisión se debe a las siguientes razones:

- La sociedad limitada es el tipo de responsabilidad mercantil más extendido en España debido a la limitación existente sobre la responsabilidad del capital aportado, es decir, en caso de insuficiencia económica el patrimonio personal de los socios no se ve implicado.
- El capital social mínimo es de 1 euro, no existiendo un capital máximo que aportar. Esto permite que los socios que conforman SeriesBuddies puedan aportar la cantidad de capital que deseen y del que dispongan con libertad.
- El número de socios mínimo es uno y el tipo de personalidad de los futuros socios puede ser físico o jurídico por lo que aporta mayor diversidad y flexibilidad a la hora de en un futuro, expandir SeriesBuddies.

En cuanto a la localización de la empresa, se establecen las siguientes directrices:

En primera instancia, SeriesBuddies no dispondrá de una oficina física propia por la falta de medios existentes. Las reuniones y encuentros necesarios durante el desarrollo y crecimiento de la empresa SeriesBuddies, se realizará en espacios de coworking. Sin embargo, la dirección física en donde se recibirán notificaciones y avisos físicos será la Calle Diamante 32 (vivienda habitual de uno de los socios y presidentes de la empresa).

Se prevé que, en el futuro, tras alcanzar una viabilidad financiera suficiente y tanto el tamaño, como otros eventos y necesidades de la empresa lo demande, se buscarán oficinas físicas que contengan la posibilidad de compartir espacios con empresas del mismo sector y empresas colaboradoras y que cuya ubicación se encuentre bien comunicada para poder llegar usando medios de transporte sostenibles.

2.3 *El nuevo proyecto: SeriesBuddies*

2.3.1 Tipo de proyecto

Características sobre la empresa SeriesBuddies que se crea para dar respuesta a las necesidades que se han detectado:

Tipo de empresa: sociedad limitada SL.

Socios: se establece un total de 2 socios por lo que el capital está repartido en particiones.

Responsabilidad: la responsabilidad de la empresa se limita al capital aportado.

Capital: el capital mínimo para constituir la empresa será de 1€.

Cotización: se cotiza en el RETA (Régimen Especial de Trabajadores Autónomos) y la sociedad tributa en el Impuesto de Sociedades.

Constitución del nombre: la sociedad limitada exige que se añada al nombre la expresión Sociedad de Responsabilidad Limitada (SRL) o Sociedad Limitada (SL), por lo que quedaría configurado como SeriesBuddies SL.

Funciones, tareas y responsabilidades:

- **CEO**
 - Román Konyeyev
 - Anabel Pedrajas Navarro
 - **Funciones:**
 - Dirección empresarial y toma de decisiones.
- **Departamento de Diseño**
 - Anabel Pedrajas Navarro – Directora creativa
 - Román Konyeyev – Asistente de diseño
 - **Funciones:**
 - Desarrollo de la imagen corporativa de la empresa.
 - Diseño y prototipado del sitio web.
 - Elaboración de gráficos y optimización de archivos utilizados en la plataforma

- **Departamento IT**

- Román Korneyev – Responsable de IT y Desarrollador
- Anabel Pedrajas Navarro – Desarrolladora

- **Funciones:**

- Desarrollo, mantenimiento y actualización de Aplicación web.
 - Desarrollo, mantenimiento y actualización de Aplicación móvil.
 - Desarrollo, mantenimiento y actualización de Bases de datos.
 - Desarrollo, mantenimiento y actualización de Entorno cliente.

- **Departamento de Marketing**

- Román Korneyev – Director de Marketing
- Anabel Pedrajas Navarro – Directora de ventas

- **Funciones:**

- Promoción de la aplicación y estrategia de publicidad.
 - Negociación de acuerdos, convenios y márgenes de beneficio con las principales plataformas de streaming
 - Elaborar una estrategia de branding para situar a la marca en el mercado y mejorar o afianzar su reputación.
 - Realizar estudios de mercado para conocer la situación del sector en el que la empresa se mueve.
 - Elaboración de campañas promocionales.
 - Diseño y desarrollo de acciones de marketing digital: presencia en redes sociales, campañas SEO/SEM, email marketing, campañas de afiliados, etc.
 - Conocer el producto, las diferentes opciones, los cambios y las tendencias del consumidor.
 - Comunicación con el cliente.
 - Controlar la calidad del producto.
 - Resolución de problemas.

- **Departamento de contabilidad y RRHH**

- Anabel Pedrajas Navarro

- **Funciones:**

- Elaboración del presupuesto.
 - Registro de los procesos contables.
 - Gestión de la financiación.

- Gestión de las inversiones.
- Gestión de la información financiera.
- Tesorería.
- Auditoría interna.

El **principal proyecto** desarrollado por la empresa (plataforma SeriesBuddies) consiste en:

- Desarrollo e implementación de software de un sitio web enfocado a cubrir las necesidades previamente comentadas (poder compartir opiniones sobre series e interactuar con otras personas).

2.3.2 Características requeridas al proyecto

¿Qué hace SeriesBuddies?

Ser una aplicación web que permita a los aficionados de las series, no solo llevar un registro del contenido que ven en diferentes plataformas de streaming, sino de poder interactuar con otros usuarios a partir del diálogo y el intercambio de opiniones sobre las mismas.

SeriesBuddies cuenta con los siguientes elementos diferenciadores:

1. **Estética:** SeriesBuddies dispone de una paleta de colores que se diferencia del resto de aplicaciones y plataformas con funciones similares debido al uso del amarillo como color primario. Así mismo, al no contar con un color negro puro y optar por degradados dota al sitio web de modernidad y permite que el usuario no sufra de fatiga visual.
2. **Interfaz cómoda e intuitiva:** la aplicación cuenta con secciones claramente diferenciadas posibilitando que el usuario tenga que memorizar lo mínimo posible a la hora de navegar por la web.
3. **Nivel de interacción de los usuarios:** se han habilitado ciertos logros o medallas (en este caso, "chips") que el usuario puede desbloquear tras la consecución de ciertas acciones.
4. **Velocidad de respuesta prácticamente instantánea:** por lo general, la carga de las distintas páginas del sitio web es instantánea, pero las que no son así, están optimizadas con peticiones Ajax, para que, aunque no carguen todos los datos de golpe, cargue la interfaz de usuario.

Entorno específico:

En las primeras etapas de SeriesBuddies como empresa, el proyecto que se llevará a cabo no requerirá de ningún proveedor externo, ya que se contempla la opción de adquirir un propio VPS donde alojar nuestro sitio web y disponer de un dominio propio.

Sin embargo, al desear en un futuro realizar ciertas mejoras y actualizaciones en la página, así como valorar el recibir ganancias económicas por afiliaciones y acuerdos con otras plataformas del sector, se plantea buscar otra API de pago que presente mejores funcionalidades y una mayor velocidad en el procesado de las peticiones.

Con respecto a la posibilidad poder realizar reuniones con futuros clientes o el mismo equipo de desarrollo, se buscarán oficinas de coworking u otros espacios colaborativos similares.

Competencia:

- **Productos sustitutivos:** aplicaciones como NextEpisode y SeriesGuide.
- **Productos complementarios:** plataformas como TV Time o Letterboxd ya que su catálogo incluye también películas.

Clientes y público objetivo:

El público contemplado se establece principalmente en los jóvenes de entre 18 y 25 años. El sexo de este segmento de mercado es indiferente.

La ubicación o zona de acción en la que SeriesBuddies pretende empezar a actuar es en Madrid, más concretamente en los núcleos urbanos.

Tanto la clase social como el poder adquisitivo del público objetivo no es relevante en estas primeras fases de despegue de la empresa ya que se trata de una plataforma totalmente gratuita.

Con respecto a los hábitos de consumo, el perfil de las personas que se desea captar con SeriesBuddies, son aquellas que hayan empezado o están acostumbradas a consumir series, documentales y programas de televisión disponibles en la televisión por cable, servicio de streaming y/o canales públicos. Personas que se perciban a sí mismas como amantes o grandes fans de este tipo de contenido y que deseen contribuir en el crecimiento de esta comunidad, así como servir de apoyo a los estudios y plataformas que producen este contenido.

Segmento de mercado:

A pesar de señalar que esta página está dedicada a un rango de edad específico, otros usuarios que no formen parte del mismo tramo de edad pueden hacer uso de la misma teniendo en cuenta las diferentes necesidades a cubrir que poseen.

- **Segmento #1:** Entre 18 y 25 años. Necesidad homogénea del segmento: Encontrar personas afines con las que poder compartir y debatir sobre diferentes series.
- **Segmento #2:** De 26 a 40 años. Necesidad homogénea del segmento: Encontrar nuevo contenido que visualizar y que sea similar a otras series ya vistas, sugerencias por géneros.

El conjunto de actividades informáticas dedicadas al proceso de creación, diseño y despliegue de software son las siguientes:

- Creación de una base de datos para la gestión y almacenamiento de los datos mediante consultas SQL utilizando MariaDB como gestor de bases de datos.
- Todas las comunicaciones del servidor irán por PHP (vanilla).
- Hacer el sitio web más dinámico del lado entorno cliente con JavaScript (vanilla).
- Representaremos la información del lado cliente con HTML5.
- Todos los documentos HTML estarán estilados con CSS3 (vanilla).

A continuación, exponemos y justificando las distintas herramientas empleadas en el desarrollo de SeriesBuddies:

Git + GitHub: Git es un sistema para guardar y rastrear cambios en proyectos y GitHub es una plataforma en línea que permite a los usuarios compartir y colaborar en proyectos utilizando Git.

Se han utilizado estas herramientas porque son las más populares, gratuitas y adecuadas para el desarrollo del proyecto.



Figura 1 - Logo de git y github

Figma es una herramienta especializada en el diseño y desarrollo de productos digitales. Permite que diferentes equipos de diseñadores y desarrolladores trabajen de forma colaborativa en tiempo real en una misma propuesta. Dispone de la capacidad de generar prototipos y animaciones interactivas simulando una página web real.

Se ha empleado Figma en el proceso de diseño, por lo anteriormente comentado, por ser una herramienta gratuita y por ser uno de los programas aprendidos en el módulo de Diseño de Interfaces Web.



Figura 2 - Logotipo de Figma

Visual Studio Code: se trata de un editor de codificación gratuito. Tiene un alto nivel de personalización mediante la instalación de extensiones, lo cual ayuda al desarrollador a programar más rápidamente. Se ha usado para codificar todos los lenguajes de programación utilizados en este proyecto.

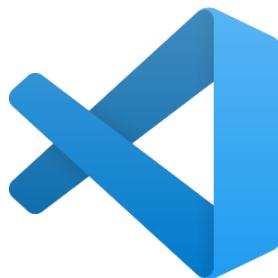


Figura 3 - Logotipo de Visual Studio Code

2.3.3 Obligaciones fiscales, laborales y de prevención de riesgo

A continuación, se resumen las principales obligaciones que presenta SeriesBuddies como empresa de sociedad limitada:

Obligación de comunicarse con la Administración pública (Hacienda y la Seguridad Social) de manera telemática a través del Servicio de Notificaciones Electrónicas y de la Sede Electrónica de la Seguridad Social. Es por tanto necesario disponer del certificado digital de SeriesBuddies.

Obligaciones contables y registrales ante el Registro Mercantil:

- Legalización de los libros sociales como el libro de actas que regula la adopción de acuerdos y el libro de registro de socios en donde se presenta la titularidad y las futuras transmisiones de las participaciones según el capital aportado.
- Seguimiento de los libros contables como el libro de inventario y el de cuentas anuales, el libro diario que presenta un registro cronológico de las operaciones de la empresa y el libro de registro del Impuesto de Valor Añadido (IVA).

Obligaciones fiscales que incluyen los procesos de:

- Registro en Hacienda de la empresa, sus empresarios y profesionales que en ella participan.
- Darse de alta en el Impuesto de Actividades Económicas por desarrollar una actividad profesional, empresarial o económica.
- Declarar el IVA soportado o repercutido.
- Pago del Impuesto sobre Sociedades.
- Retenciones del IRPF cuando se incorporen a la plantilla nuevos empleados.

Obligaciones con la Seguridad Social:

- Comunicar el alta y la baja de los futuros trabajadores en un plazo de 60 días y 3 días, respectivamente.
- Estar al día de los pagos de los seguros de los futuros trabajadores.

Obligaciones con respecto a la prevención de riesgos laborales:

- Evitar la conocida “tecno ansiedad” y aumentar la motivación de los empleados mediante la continua participación y la formación de los trabajadores con herramientas potentes e innovadoras.
- Disminuir el estrés y aumentar la flexibilidad laboral con el fin de facilitar la desconexión digital tras la jornada laboral.
- Formar a los trabajadores en materia preventiva de los riesgos laborales haciendo hincapié en enfermedades como “tecno fatiga”, “tecno adicción”, fatiga visual, cefaleas, problemas de atención, sueño y concentración y otros riesgos psicosociales.

2.3.4 Ayudas/subvenciones

A pesar de no haber encontrado ayudas y subvenciones beneficiosas para SeriesBuddies antes de poner en marcha la creación y desarrollo del proyecto, sí existen otros programas que se podrían solicitar una vez establecida la empresa de manera oficial.

Microcréditos del Ministerio de Igualdad: este programa concede hasta 25.000 euros a las mujeres que deseen crear su propia empresa o promocionar y consolidar una ya creada.

Programa ENISA del Ministerio de Industria: enfocado en ayudar a los creadores más jóvenes, ofrece una ayuda económica de entre los 25.000€ hasta los 75.000€. El primer requisito que cumplir es que la empresa haya sido constituida como máximo 24 meses antes de realizar la solicitud. El segundo requisito establece que la actividad a desarrollar y el domicilio social de la empresa se encuentren en territorio nacional. Por último, se indica que el modelo de negocio de la empresa sea innovador y posea una ventaja competitiva.

Premios de emprendimiento Joven-Carné Joven: la finalidad es premiar a los proyectos realizados por jóvenes y que sean ejemplo de innovación, compromiso medioambiental y/o social. Los destinatarios que pretendan optar por estas ayudas deben ser personas físicas de entre los 14 y los 30 años y que estén en posesión del Carné Joven de la Comunidad de Madrid. Por otro lado, los proyectos a presentar deberán haberse configurado dentro de los 5 años anteriores a la publicación del Boletín Oficial de la Comunidad de Madrid; el proyecto empresarial debe cumplir con los estándares de originalidad, creatividad e innovación y que se desarrolle la actividad empresarial en la Comunidad de Madrid. Las cuantías que se ofrecen a los ganadores de estos premios parten de los 2.000€, alcanzando la cifra máxima de los 8.000€.

3 DISEÑO DEL PROYECTO

Una vez vista la viabilidad del proyecto, en este apartado se concretarán las fases necesarias para llevarlo a cabo y cumplir con los objetivos que se establezcan, teniendo en cuenta los recursos necesarios.

3.1 Fases del proyecto

El proyecto se encuentra dividido en cuatro fases principales: análisis, diseño, implementación y pruebas. Dichas fases se detallan a continuación.

3.1.1 Análisis

En la primera fase del análisis se determinan los requisitos del proyecto. Diferenciamos entre requisitos funcionales, aquellos que tienen que ver con lo que el sitio web tiene que hacer; y los requisitos no funcionales, que responden a las cualidades que el sitio web debe cumplir y transmitir.

FUNCIONALES

- Sitio web disponible las 24 horas del día, todos los días del año, para garantizar el acceso al usuario en todo momento.
- Permitir al usuario compartir ideas sobre una serie al instante desde cualquier parte del mundo y/o debatir sobre estas ideas y opiniones con otros usuarios.
- Establecer un círculo de amigos cercanos a partir del envío de peticiones de amistad.
- Ayudar a los usuarios a hacerse una idea general sobre una serie gracias a los comentarios que otros “buddies” han publicado.

NO FUNCIONALES

- Dotar al sitio web de un diseño atractivo para atraer al público al que va dirigido.
- Implementar una interfaz intuitiva con el fin de minimizar la memorización de los buddies, asegurando un comportamiento predecible y proporcionando acceso rápido a la información práctica y útil de la web.
- Uniformidad estructural en todas las páginas que conforman el sitio web para no desorientar al usuario.
- Por otro lado, esta aplicación se debe adaptar a todo tipo de pantallas para facilitar el acceso al usuario desde cualquier plataforma y lugar.
- Además de lo comentado anteriormente, el sitio web será compatible con todos los navegadores principales: Chrome, Firefox, Internet Explorer, Safari y Opera.

3.1.2 Diseño

Antes de realizar un primer boceto de cómo se desea configurar el sitio web, se ha realizado una breve investigación de otros diseños y layouts de plataformas con funcionalidades similares a SeriesBuddies en portales como Pinterest, Behance y Figma. De esta forma conseguimos establecer una estructura básica que además de ser familiar para el usuario y evitar que se pueda desorientar, cumplirá con los principios UX/UI del diseño (estructura en bloques claramente diferenciados, diseño que facilite las tareas más comunes y evitar información innecesaria).

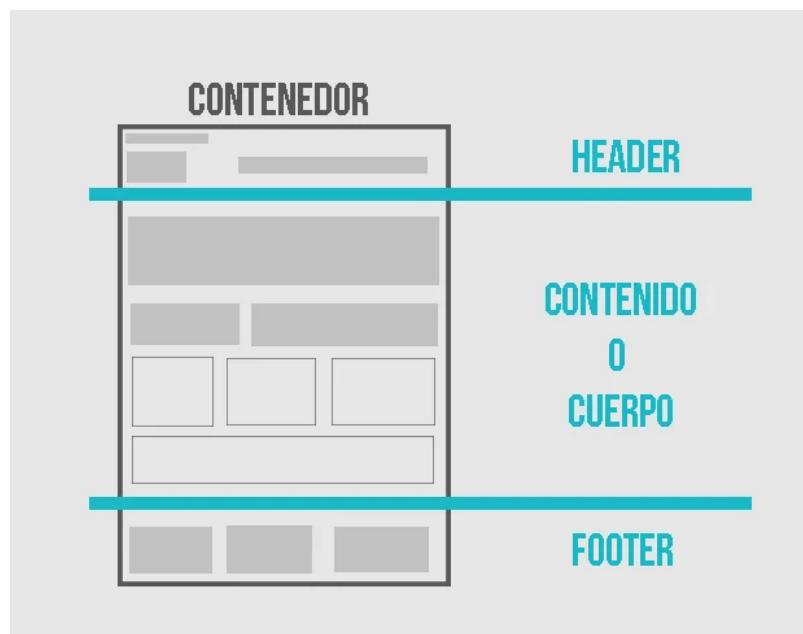


Figura 4 - Estructura layout del proyecto

Es importante mencionar que para realizar tanto el wireframe como el mockup se ha utilizado el editor gráfico y de prototipado Figma. Las razones por las que se ha escogido este programa son las siguientes:

- Figma ha sido uno de los editores empleados en el módulo de Diseño de Interfaces Web.
- Destaca en el mundo del diseño digital, posicionándose hoy en día, como el programa de referencia en el diseño de páginas web e interfaces de aplicaciones.
- Figma permite el trabajo colaborativo simultáneo, agilizando el proceso creativo y de maquetación de las diferentes vistas con las que cuenta el sitio web.
- Dispone de varias herramientas que facilitan luego el proceso de maquetación web. Por ejemplo: posibilidad de varias capas y controlar la visibilidad de estas, cuenta con distintos espectros de colores (RGB, HSL, hexadecimal...), posibilidad de crear gráficos vectoriales que puedan exportados como SVG.



Figura 5 - Logotipo Figma

Wireframe

Con respecto al diseño y la interfaz del sitio web, se elabora un wireframe a escala de grises sobre el que poder plasmar y organizar las diferentes partes que compondrán la aplicación.

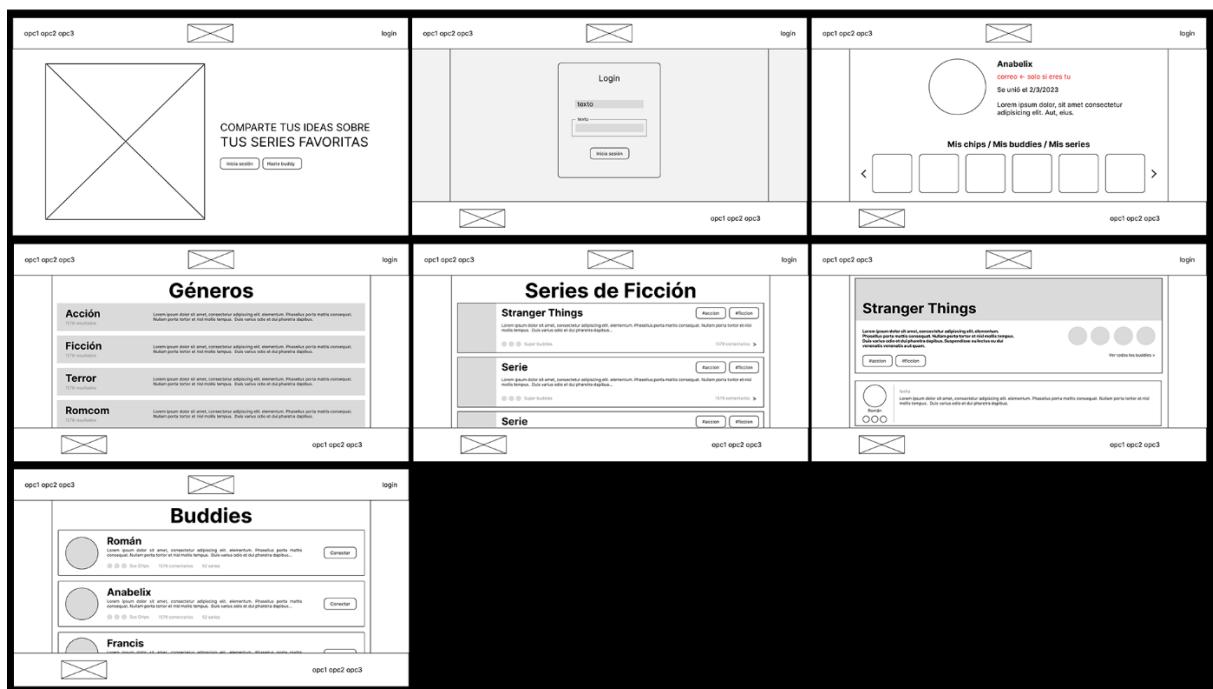


Figura 6 - Wireframe completo del proyecto

Nombre, logo y paleta de colores

¿Por qué se escogió SeriesBuddies como nombre del proyecto?

Tras una lluvia de ideas se ha decidido combinar los dos pilares fundamentales que dan forma a el sitio web: las personas y las series. Para darle un nombre más fresco, creativo e internacional se optó por combinar el inglés y el español. Hay que tener en cuenta que SeriesBuddies presenta una pronunciación sencilla y muy similar tanto en inglés como en español.

SeriesBuddies se caracteriza por ser un nombre breve y con ligero ritmo al contar las dos palabras con la misma terminación (-es) facilitando así su memorización.

Al escoger la palabra “buddies” se desea hacer llegar el mensaje de que se trata de una plataforma informal, casual, sencilla y jovial. Buddies se traduce como amigos, compañeros, colegas e incluso compinches. Esto ayuda a la imagen de la marca puesto que reafirma el hecho de que se trata no sólo de una plataforma o foro donde dejar comentarios sobre series, sino que puede llegar a convertirse en una red social donde poder conocer, interactuar y reforzar lazos de amistad con otros usuarios.

¿Qué significado posee el isotipo de SeriesBuddies y cuál fue su proceso de creación?

Se empezó diseñando una serie de bocetos que englobaran o representaran las palabras “series” y “buddies”. Destaca el uso de iconos de usuarios, el icono del play, caras sonrientes y bocadillos para representar el concepto de comentario y mensajería. En este primer estado de concepción de la imagen visual de SeriesBuddies destacaba el isotipo como representación de la marca.

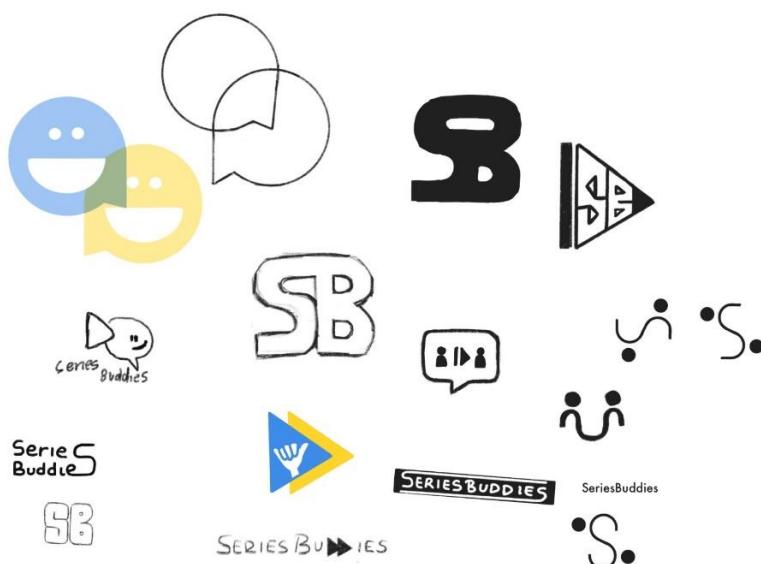


Figura 7 - Bocetos de logos del proyecto

Tras valorar las propuestas elaboradas, se decidió combinar dos iconos de “play” superpuestos entre sí y que, en la capa superior, apareciese un mano que tuviera los dedos índices y meñiques levantados. Este elemento iconográfico se escogió debido a su significado en la cultura de la música Heavy, la cual establece este gesto como una forma de saludo que sugiere “¿Qué tal? ¿Todo bien?”. Con esta sencilla ilustración se quiere reforzar la idea de que SeriesBuddies es un espacio seguro, abierto y divertido en donde poder conocer a gente con gustos similares en series.



Figura 8 - Logo final del proyecto (móvil)

Por último, se decidió combinar tipografía e imagen. A la palabra “SeriesBuddies” escrita en Nunito, se le suprimieron las dos letras d y en su lugar se posicionó el logotipo anterior configurando el isologotipo definitivo para la empresa y el proyecto.



Figura 9 - Isologotipo final del proyecto (PC)

¿Cuál es la paleta de colores elegida para SeriesBuddies?

La paleta de colores cuenta con dos colores primarios, el amarillo y el azul. El resto de los tonos son variaciones del amarillo y tonalidades cercanas al negro.

El amarillo representa la energía, la alegría y la felicidad. Al ser un color con tanta fuerza, invita al usuario a adquirir una posición activa. El amarillo es un color poco usado en el diseño de interfaces web ya que el ojo no suele estar habituado a verlo de forma excesiva o en grandes superficies, es por esto por lo que se ha optado por colocarlo en los títulos y subtítulos de las páginas web. Para el resto de los elementos como tarjetas de información y comentarios, se utilizan tonalidades más pasteles para evitar fatiga visual, cansancio y sobrecarga.



Figura 10 - Colores primarios

El azul representa seriedad y confianza. Utilizada comúnmente por compañías tecnologías y redes sociales como Facebook o Twitter. Es un color que invita al usuario a entrar en un estado de tranquilidad y frescura. El uso de un tono saturado favorece centrar focos de atención en determinados elementos y, en consecuencia, se obtiene una experiencia de usuario y navegación más clara.



Figura 11 - Colores secundarios

Por último, los tonos oscuros permiten aportar minimalismo, sencillez y seriedad a la página. Al usarse como degradados y no hacer uso del negro puro, se evita el contraste exagerado y la fatiga visual. Es importante mencionar, que al emplear varios tonos oscuros como fondo de la interfaz posicionamos a SeriesBuddies en el mismo espectro que otras plataformas y aplicaciones que utilizan esta misma paleta de colores.



Figura 12 - Colores de fondo

Mockup

Una vez realizado el wireframe y decidida la paleta de colores sobre la que trabajar, se desarrolla un mockup o prototipo, donde se puedan ver los estilos definidos a las diferentes páginas y cómo se expondría la información al usuario de manera que le fuera atractiva, sencilla e intuitiva.

Al mismo tiempo, se pensó en qué efectos o animaciones se podían incluir en el sitio web para dotarla de mayor dinamismo y modernidad.

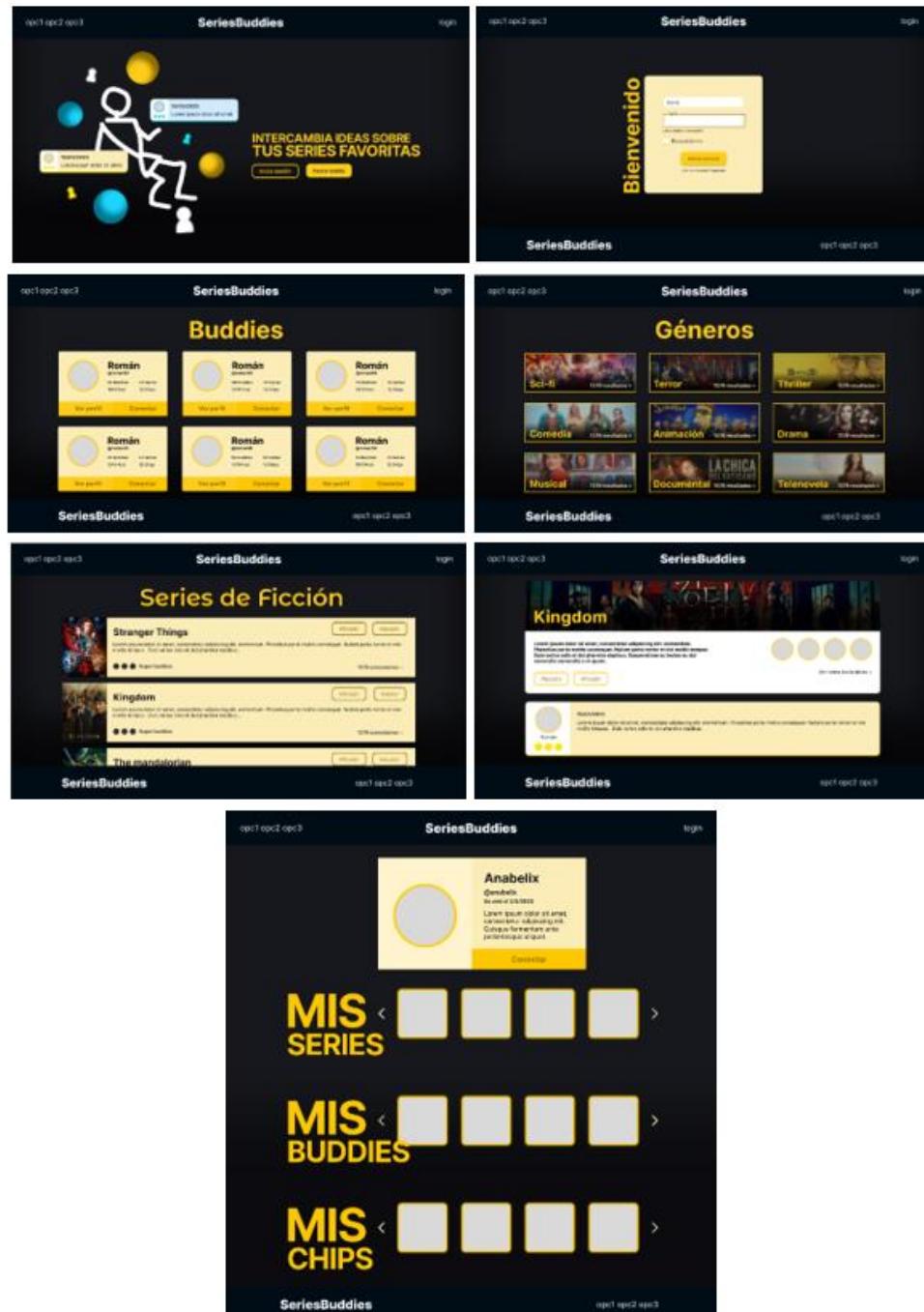


Figura 13 - Mockup completo del proyecto

Páginas y funciones

A continuación, se especifican todas las páginas que compondrán la plataforma y que son las siguientes:

- **Índice:** página de inicio o home que a su vez actúa como página de landing.

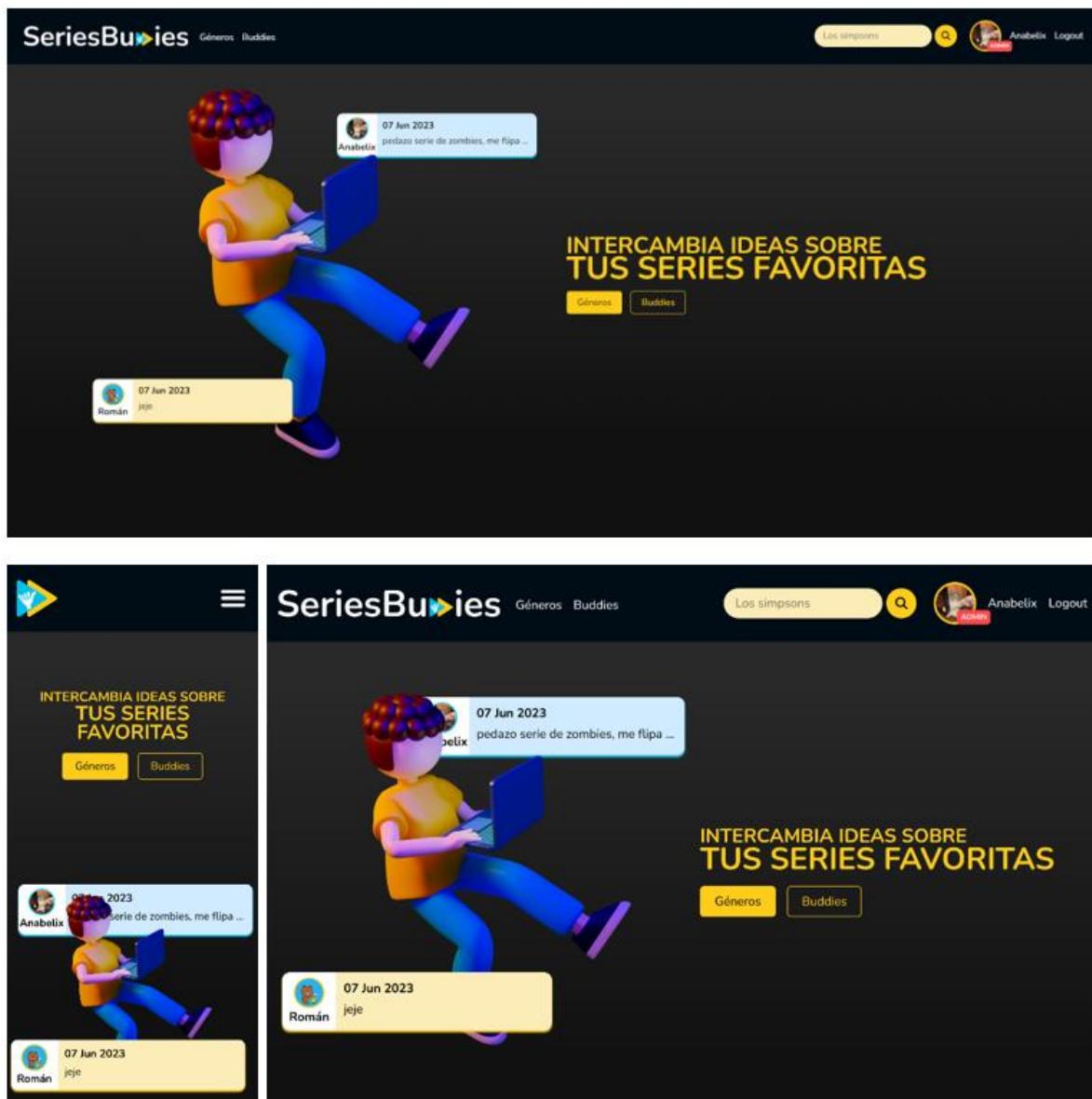


Figura 14 – pantalla de index

- **Registro:** página donde los nuevos usuarios puedan darse de alta.

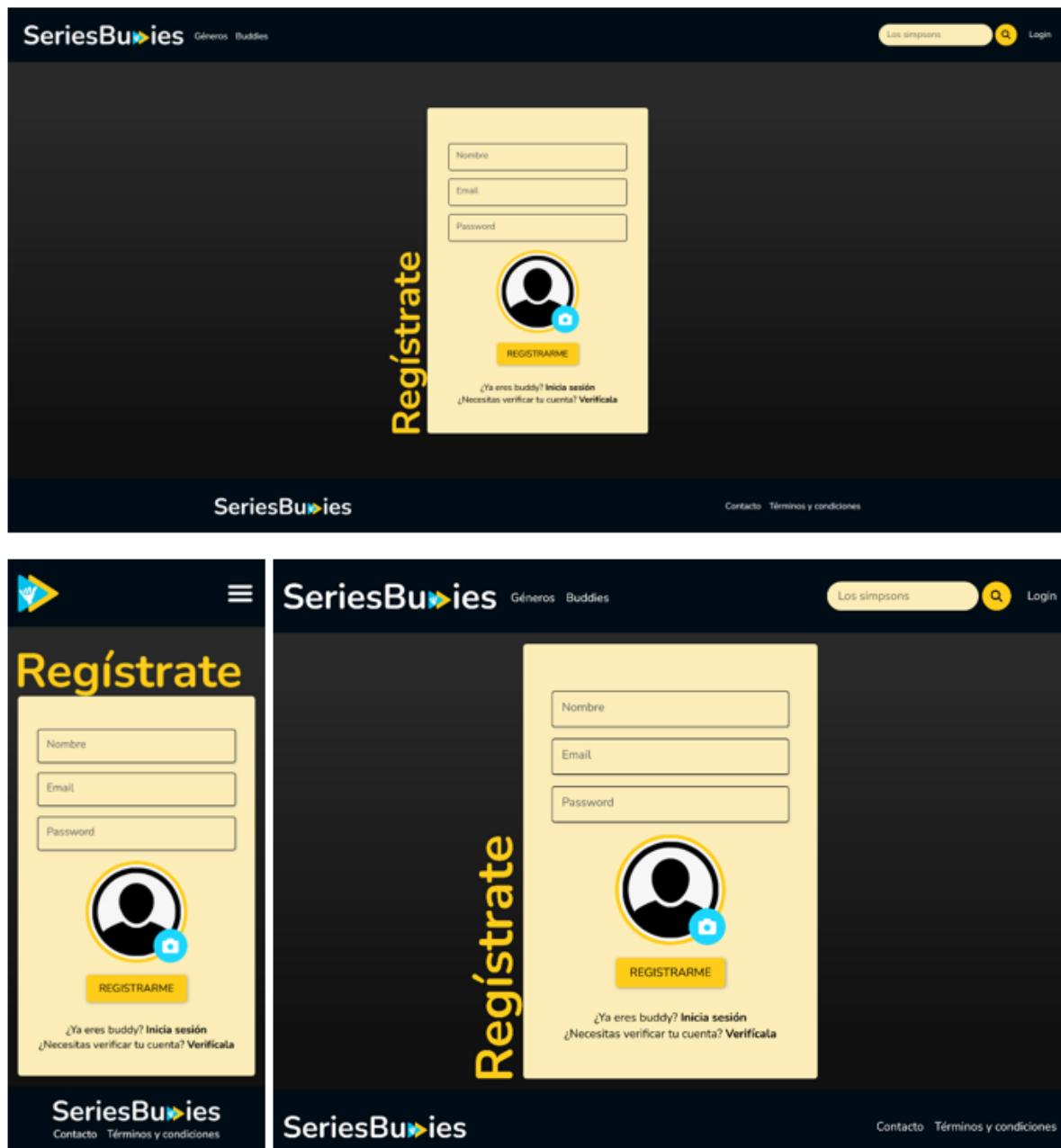


Figura 15 - Pantalla de registro

- **Inicio de sesión:** página donde los usuarios puedan iniciar sesión y tengan habilitadas otras funciones que los usuarios no registrados no posean.

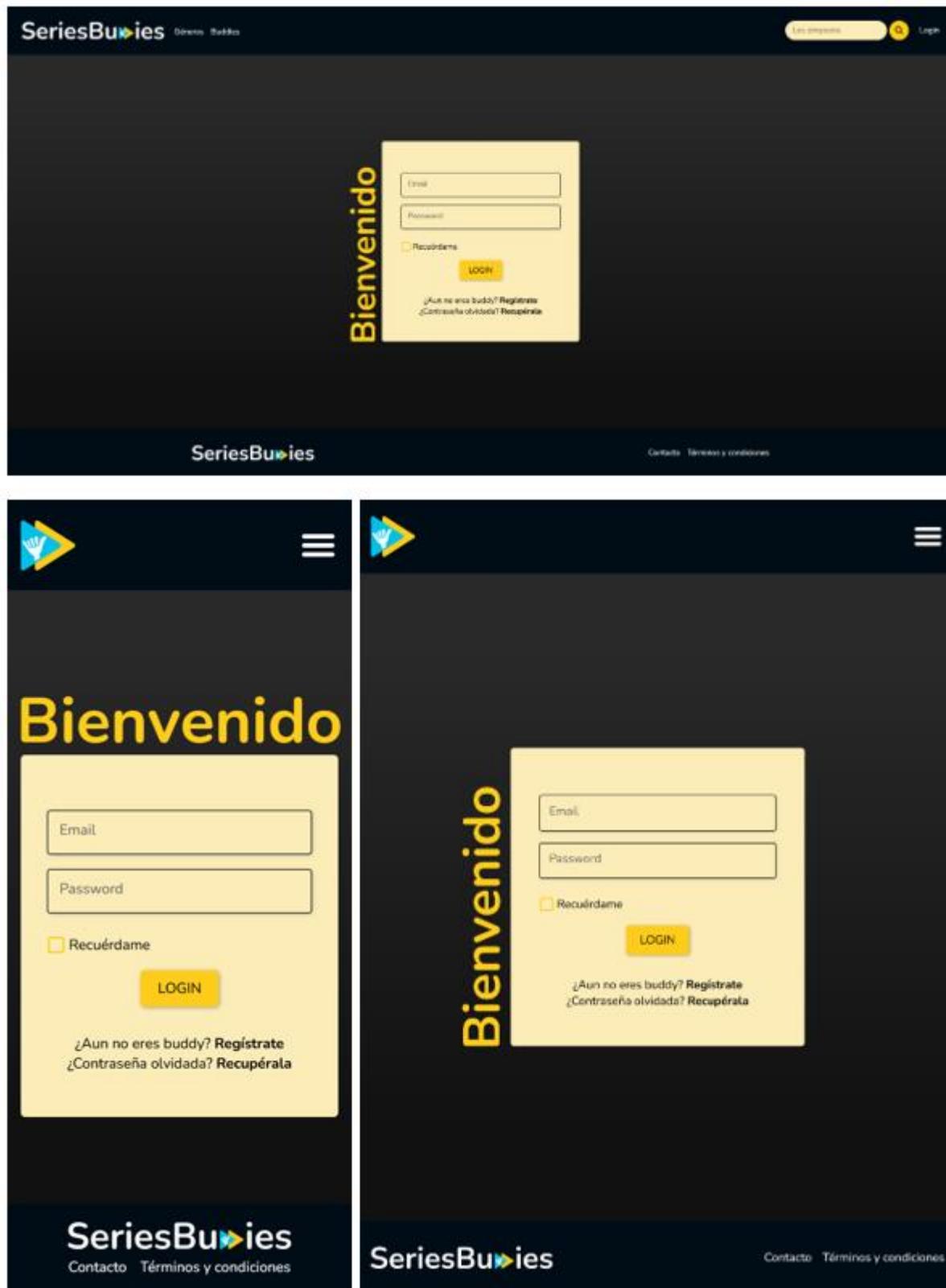


Figura 16 - Pantalla de inicio de sesión

- **Recuperación de contraseña:** página donde los usuarios que hayan olvidado o quieran restablecer la contraseña de su cuenta, pueden solicitar un email que les permita realizar esta acción a partir del envío de un token personal con una fecha de expiración para mayor seguridad.

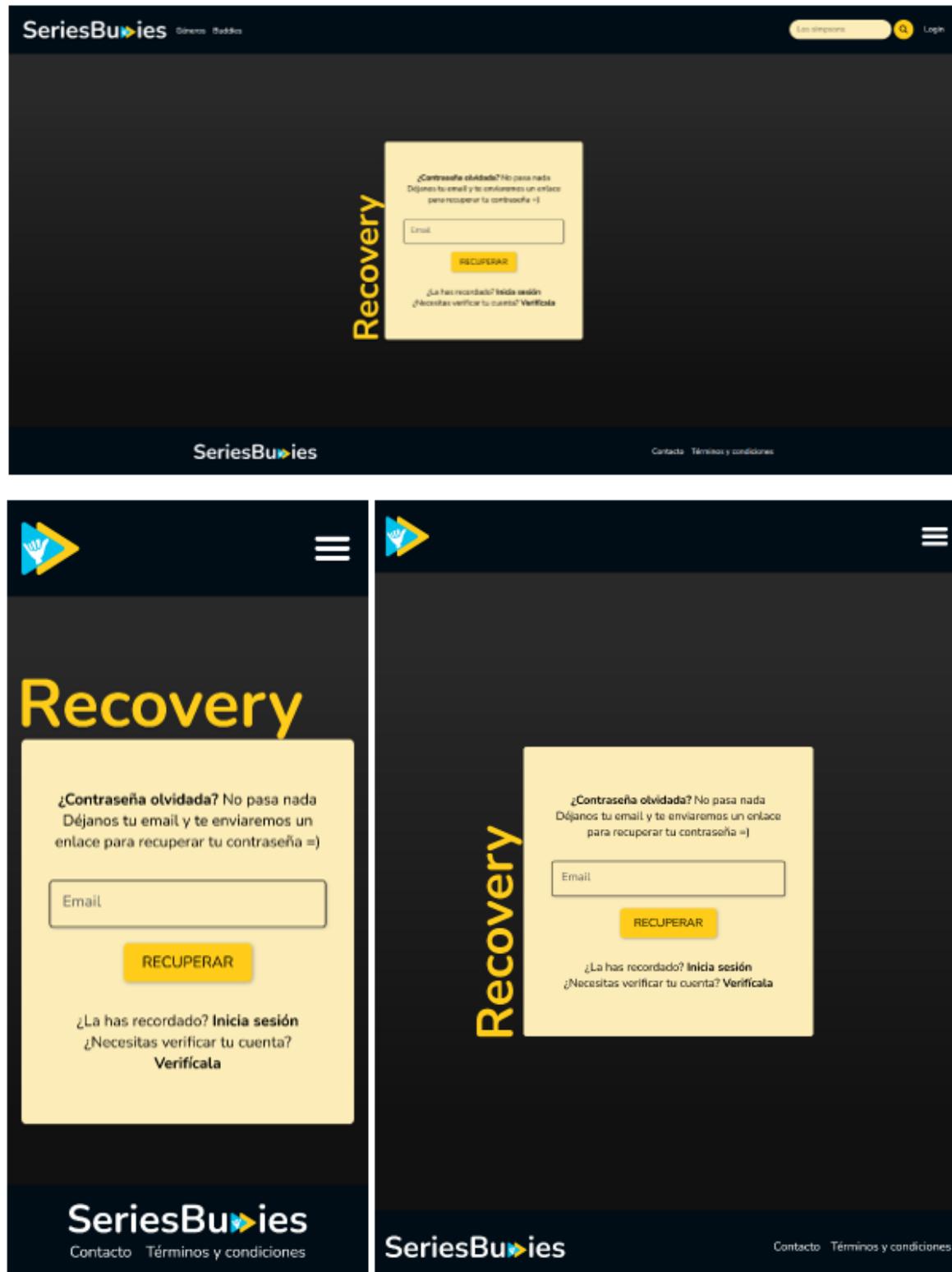


Figura 17 - Pantalla recuperación de contraseña

- **Verificación:** página donde los usuarios puedan solicitar un nuevo correo de verificación en el caso de que no les haya llegado el de registro y a su vez, verifica aquellos que posean un enlace válido reforzando así la seguridad del sitio web.

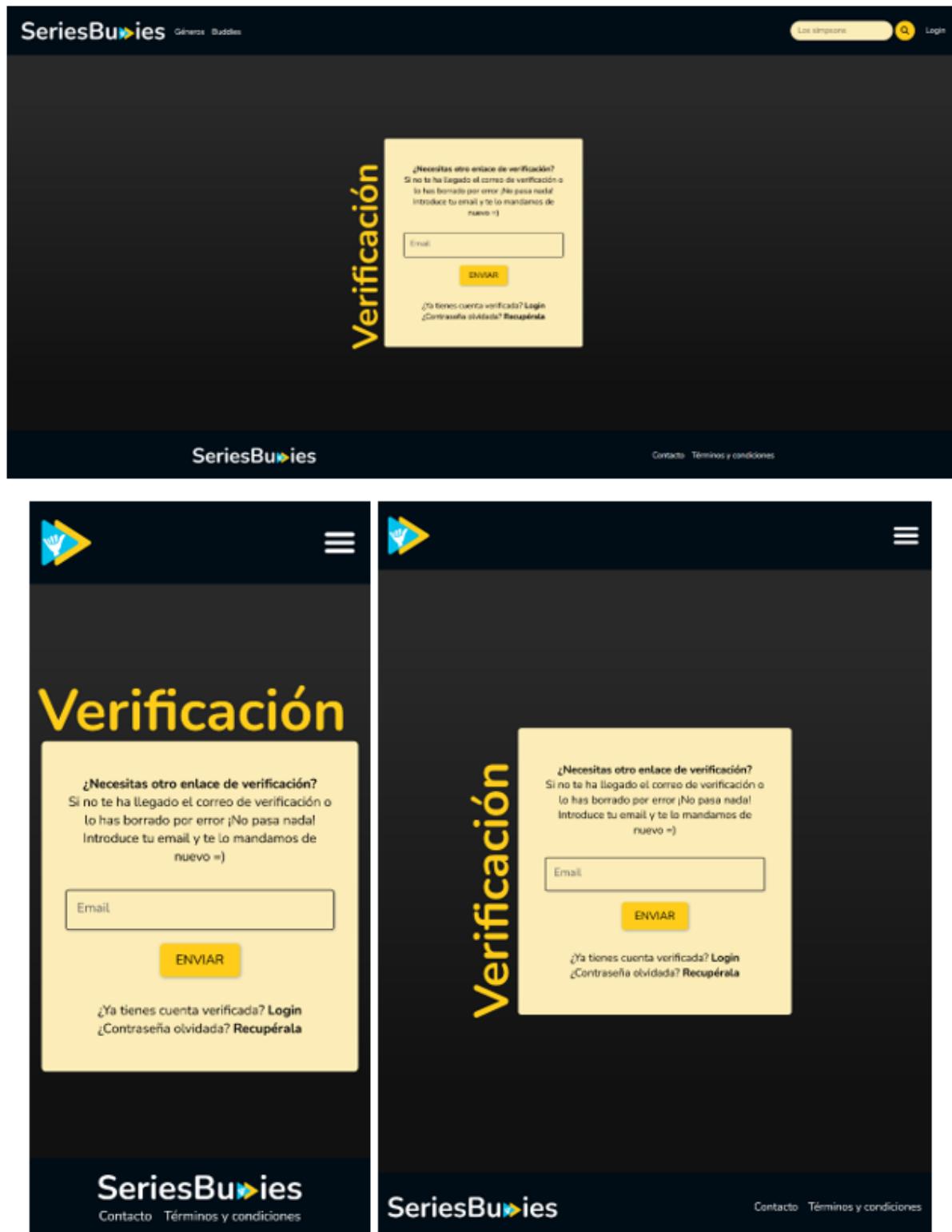


Figura 18 - Pantalla de verificación

- **Géneros:** página donde se muestre el listado de los géneros de todas las series que hay en SeriesBuddies y del total de resultados encontrados.

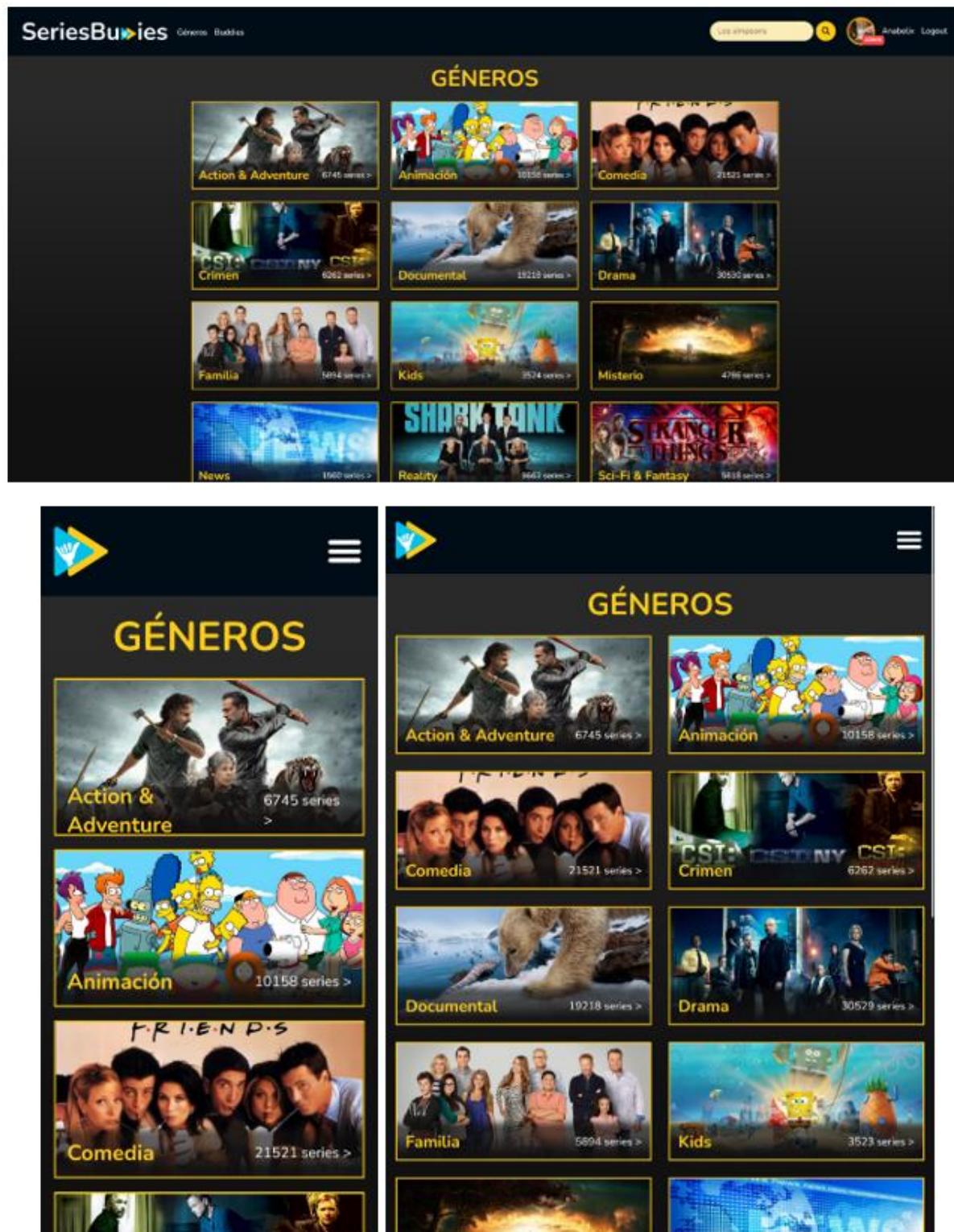


Figura 19 - Pantalla de géneros

- Serie:** página que muestra una lista de las diferentes series filtradas por un determinado género y/o una búsqueda. También indica los datos más importantes sobre la serie (imagen tipo poster, título y breve descripción), además de información adicional (las tres personas que más han comentado en dicha serie). Al hacer clic sobre una serie, se redirige al usuario al feed de la misma.

The screenshot shows the SeriesBuddies website interface. At the top, there's a navigation bar with links for 'Géneros', 'Súper buddies', 'Series', 'Citas románticas', 'Análisis', and 'Login'. Below the header, a breadcrumb navigation 'Géneros > Sci-Fi & Fantasy' is visible. The main content area is titled 'Series de Sci-Fi & Fantasy' and shows a list of three series:

- Voltus V: El Legado**: A real-life adaptation of the Japanese anime Chōdenji Machine Voltes V. It's described as a mysterious town that traps everyone who enters. The series is developed by GMA Network and produced by Toei Animation and Nippon Sunrise. It's also known as Voltes V in the Philippines and Voltres Cinco in Cuba and other parts of Latin America. There are 0 comments.
- From**: Describes the mystery of a small town in North America that traps everyone who enters. Residents fight to maintain a sense of normality and find a way out. It's developed by GMA Network and produced by Toei Animation and Nippon Sunrise. It's also known as From in the Philippines and Voltres Cinco in Cuba and other parts of Latin America. There are 0 comments.
- The Flash**: After a particle accelerator causes a strange storm, investigator Barry Allen is struck by lightning and falls into a coma. Months later, he wakes up with super-speed abilities. It's developed by GMA Network and produced by DC Comics. There are 0 comments.

Below the main content, there's a sidebar with a play button icon and the text 'Géneros > Sci-Fi & Fantasy'. The sidebar also features a large 'Series de Sci-Fi & Fantasy' heading and a 'Súper buddies' button. At the bottom of the sidebar, there's a preview of another series card for 'Voltus V: El Legado'.

Figura 20 - Pantalla de series

- **Feed:** página que indica de forma más detallada la información de una determinada serie (nombre, descripción completa, listado de géneros a los que pertenece e imagen tipo banner). La página feed permite a los usuarios dejar comentarios (que se muestran ordenados por fecha de manera descendente) sobre esa serie mostrando la fecha de publicación, el nombre del usuario, su foto de perfil y los chips que posee actualmente. Estos comentarios pueden ser editados con posterioridad y eliminados tanto por el propietario de ese comentario como por los administradores de SeriesBuddies. Por último, esta página posee un enlace que redirige al usuario al listado de todos los buddies que han participado en esa serie.

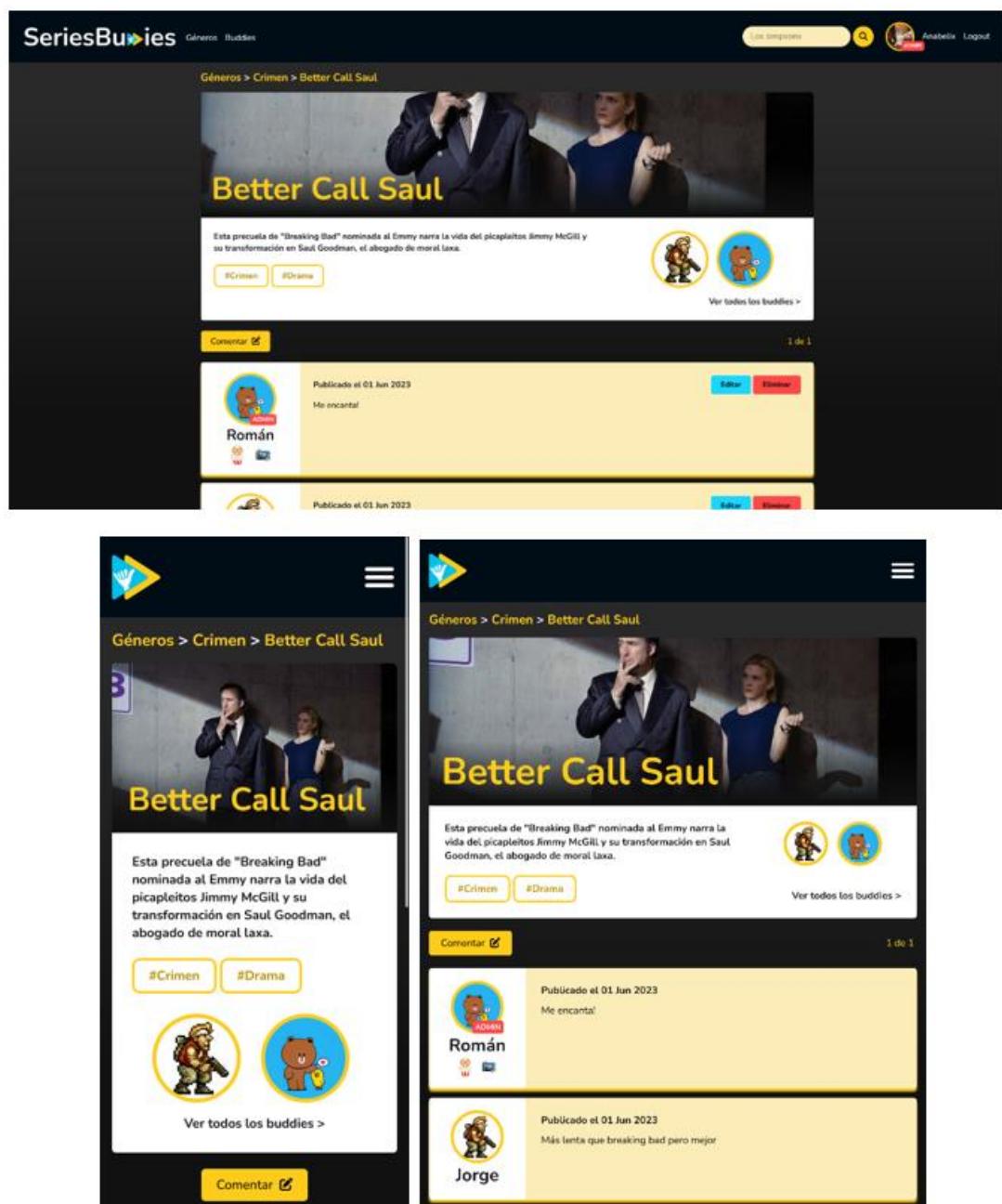


Figura 21 - Pantalla de feed

- Buddies:** en esta página se muestra un listado de todos los buddies que pueden o no estar filtrados por su participación en una serie y/o una búsqueda por nombre. Esta información se presenta en formato tarjeta en la que se indica de forma resumida el número total de amigos que posee en SeriesBuddies, el total de series en las que ha participado, el total de post que ha escrito y el total de chips que posee. Estas tarjetas cuentan con un footer que contiene acciones rápidas para ver el perfil de ese usuario, enviarle una petición de amistad o de aceptar/rechazar su invitación, por otro lado, si el usuario que está visualizando esta página tiene privilegios de administrador, podrá eliminar a cualquier usuario.

Buddy	Nombre	Handle	Buddies	Posts	Series	Chips
Jorge	@Jorge#1		0 Buddies	1 Posts	1 Series	1 Chips
Román	@Román#2	ADMIN	6 Buddies	13 Posts	10 Series	3 Chips
Esteban dido	@Esteban dido#3		1 Buddies	1 Posts	1 Series	1 Chips
PruebasXD	@PruebasXD#5		1 Buddies	0 Posts	0 Series	1 Chips
Francis	@Francis#6		1 Buddies	2 Posts	2 Series	1 Chips
Alvaro_33	@Alvaro_33#7		2 Buddies	6 Posts	6 Series	2 Chips
Almudenia	@Almudenia#8		1 Buddies	1 Posts	1 Series	1 Chips
Anabelix	@Anabelix#9	ADMIN	3 Buddies	6 Posts	6 Series	2 Chips
Natalia	@Natalia#10		0 Buddies	1 Posts	1 Series	1 Chips

Figura 22 - Pantalla de buddies (usuarios)

- Perfil del usuario:** página dedicada a visualizar y/o gestionar la información de un usuario específico. La información principal del usuario (nombre, foto de perfil, alias, fecha en la que se unió y descripción) se encuentra presentada en formato tarjeta. Esta información sólo podrá ser editada por el mismo usuario o el administrador. Si un usuario inicia sesión en el sitio web y accede a su página de perfil, podrá ver en el lado derecho las peticiones de amistad recibidas y gestionarlas (aceptar o rechazar). La información adicional del usuario (series en las que ha participado, amigos y chips) viene representada en un bonito formato carrusel.

The figure consists of two screenshots of the SeriesBuddies website, one for desktop and one for mobile, showing a user profile page for 'Román'.

Desktop Screenshot:

- User Profile:** Shows a circular profile picture of a brown bear, the name 'Román' with an 'ADMIN' badge, the handle '@Román#2', the date 'Se unió el 01 Jun 2023', and the message 'Soy admin, no os portéis mal.'
- SUS SERIES:** A horizontal row of five small thumbnail images representing TV shows: Better Call Saul, Los Simpson, Family Guy, Bob's Burgers, and a fourth show partially visible.
- SUS BUDDIES:** A horizontal row of five small thumbnail images representing users or profiles: a character from League of Legends, a man with a mustache, a young man with blonde hair, a cat, and a yellow character.

Mobile Screenshot:

- User Profile:** Shows the same profile information as the desktop version.
- SUS SERIES:** A horizontal row of five small thumbnail images representing TV shows: Better Call Saul, Los Simpson, Family Guy, Bob's Burgers, and a fourth show partially visible.
- SUS BUDDIES:** A horizontal row of five small thumbnail images representing users or profiles: a character from League of Legends, a man with a mustache, a young man with blonde hair, a cat, and a yellow character.

Figura 23 - Pantalla de perfil

- **Página de contacto:** página que permite a los usuarios registrados ponerse en contacto con los administrados de la página web con respecto a dudas, sugerencias, propuestas de mejora de SeriesBuddies y peticiones para ampliar el catálogo de series.

The figure displays three versions of the SeriesBuddies contact page:

- Desktop Version:** Shows a dark-themed contact form with fields for Name, Email, Subject, and Message, along with a yellow 'Enviar' button. The background features a sidebar with 'Contacto' and other site links like 'Los simpsons' and 'Logout'.
- Tablet Version:** A responsive version of the contact form, maintaining the same layout and styling as the desktop version but adapted for a smaller screen.
- Mobile Version:** A fully responsive version of the contact form, designed for mobile devices with a clean layout and touch-friendly buttons.

Figura 24 - Pantalla de contacto

Comunicaciones con el usuario

Todas las comunicaciones con el usuario (mensajes automatizados de completar registro, reestablecer contraseña, etc.) se harán siguiendo una estructura de correo:

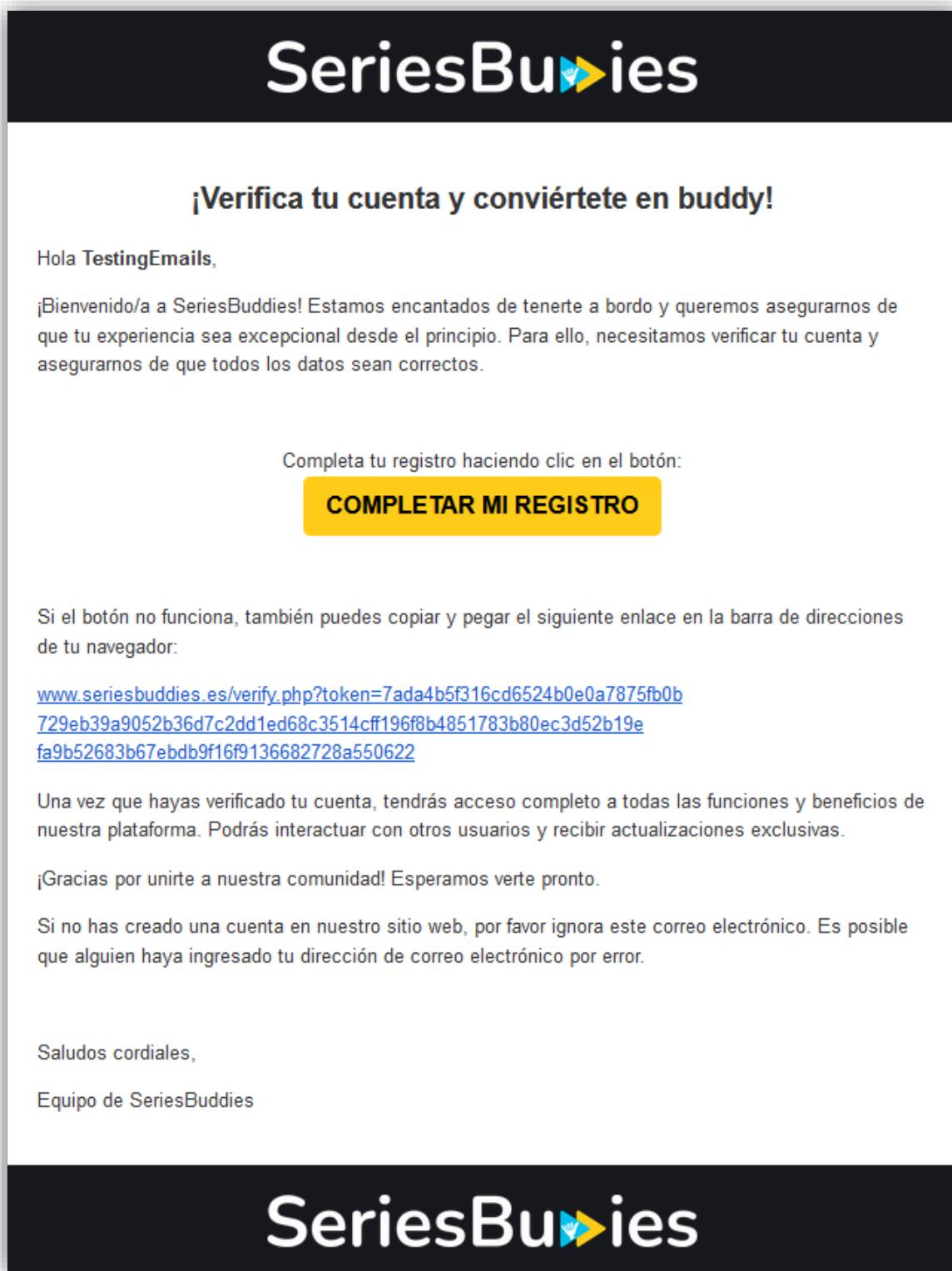


Figura 25 - Estructura de correos automatizados

Diseño de la base de datos

Para el diseño de la base de datos, se usará un gestor de base de datos relacional (MariaDB) y se diseña un modelo entidad relación para poder ver cómo interactúan las tablas entre sí.

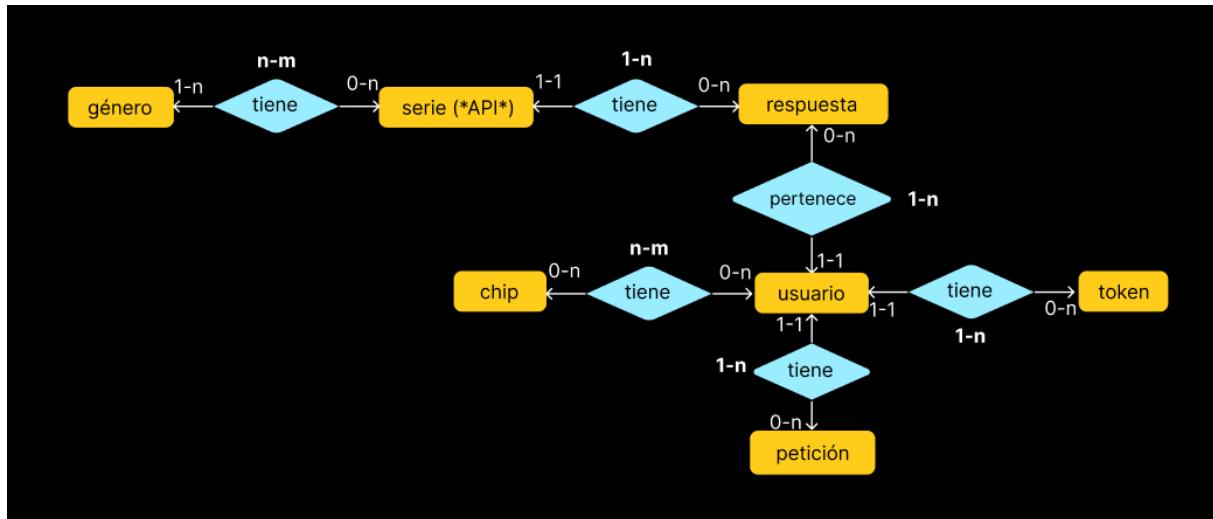


Figura 26 - Modelo E/R de la base de datos

Por último, para el desarrollo back-end tras barajar varias opciones como Python, PHP y Java, se elige PHP como lenguaje de desarrollo entorno servidor debido a que, a pesar de ser lenguaje de scripting de propósito general, es un lenguaje enfocado al desarrollo web.

Como ya se ha mencionado anteriormente, SeriesBuddies se construye a partir de la utilización de los lenguajes HTML, CSS y JavaScript para el lado front-end, mientras que para la gestión y la carga de los datos se hace uso de MariaDB.

Todo será desplegado en un servidor Linux VPS con Apache.

3.1.3 Implementación

Para llevar a cabo la implementación del diseño del proyecto se distinguen varios bloques principales:

- **Diseño de la Base de Datos**
- **Estructura del proyecto, de las páginas y programación de la lógica de estas**
- **Despliegue**

DISEÑO DE LA BASE DE DATOS

Partiendo del modelo entidad relación visto en el punto anterior, se comienza a elaborar la base de datos del proyecto.

Se empieza con la creación de las tablas, sus claves primarias, ajenas y demás restricciones.

Tabla usuarios

```

9  **** TABLA USUARIOS ****
10 DROP TABLE IF EXISTS usuarios CASCADE;
11 CREATE TABLE usuarios (
12     id          INT AUTO_INCREMENT PRIMARY KEY,
13     nombre      VARCHAR(255) NOT NULL,
14     contra      VARCHAR(255) NOT NULL,
15     img         VARCHAR(255) DEFAULT 'upload/perfiles/default.png',
16     correo      VARCHAR(255) NOT NULL UNIQUE,
17     descripcion TEXT,
18     privilegio  ENUM('admin', 'usuario') NOT NULL,
19     verificado  ENUM('si', 'no') NOT NULL,
20     fecha_alta  DATETIME DEFAULT NOW(),
21     ult_tkn_solicitado DATETIME DEFAULT (NOW() - INTERVAL 1 DAY),
22     ult_contacto DATETIME DEFAULT (NOW() - INTERVAL 1 DAY)
23 );

```

Figura 27 - Tabla usuarios

- **id:** identificador único para los usuarios. Se autoincrementa con cada nueva inserción en la tabla. Es la clave primaria.
- **nombre:** nombre del usuario. Este campo no puede ser nulo.
- **contra:** contraseña del usuario. Este campo no puede ser nulo.
- **img:** imagen de perfil del usuario. Este campo sí puede ser nulo, de ser así se le aplica una foto por defecto.
- **correo:** dirección de correo electrónico del usuario, es con la que el usuario podrá identificarse en el sitio web. Este campo no puede ser nulo.

- **descripción:** pequeña biografía sobre el usuario. Este campo sí puede ser nulo.
- **privilegio:** nivel de privilegios del usuario. Pudiendo ser usuario común o administrador. Admite valores admin o usuario. Este campo no puede ser nulo.
- **verificado:** indica si el usuario está verificado para que pueda hacer uso de su cuenta en el sitio web. Admite valores si o no. Este campo no puede ser nulo.
- **fecha_alta:** señala la fecha de alta del usuario en la plataforma. Se establece la fecha de forma automática al mismo tiempo que se realiza el insert en la tabla (al registrarse el usuario).
- **ult_tkn_solicitado:** hace referencia a la fecha de solicitud de token para restablecer la contraseña. Esta aplicación se limita a uno por día. De serie se inicializa en el día anterior, para que el usuario pueda obtener un correo para restablecer la contraseña el mismo día que se ha registrado en caso de que se haya olvidado de ella. Este campo sí puede ser nulo.
- **ult_contacto:** indica la última fecha en la que el usuario ha contactado con los administradores mediante la página de contacto. Al igual que el punto anterior, esta acción se limita a uno por día. Como en el caso anterior, también se inicializa en el día anterior, para que el usuario pueda contactar con los administradores el mismo día que se creó la cuenta. Este campo sí puede ser nulo.

Tabla peticiones

```

38 | /***** TABLA PETICIONES *****/
39 | DROP TABLE IF EXISTS peticiones CASCADE;
40 | CREATE TABLE peticiones (
41 |     id          INT AUTO_INCREMENT PRIMARY KEY,
42 |     id_emisor   INT NOT NULL,
43 |     id_receptor INT NOT NULL,
44 |     estado      ENUM('aceptada', 'pendiente', 'rechazada') NOT NULL,
45 |     CONSTRAINT fk_emisor FOREIGN KEY (id_emisor) REFERENCES usuarios(id) ON DELETE CASCADE,
46 |     CONSTRAINT fk_receptor FOREIGN KEY (id_receptor) REFERENCES usuarios(id) ON DELETE CASCADE
47 | );

```

Figura 28 - Tabla peticiones

- **id:** identificador único para las peticiones. Se autoincrementa con cada nueva inserción en la tabla. Es la clave primaria.
- **id_emisor:** identificador del usuario que envía una petición de amistad a otro usuario. Es una clave foránea de la tabla usuarios. Este campo no puede ser nulo.
- **id_receptor:** identificador del usuario que recibe una petición de amistad de otro usuario. Es una clave foránea de la tabla usuarios. Este campo no puede ser nulo.

- **estado:** especifica el estado de la petición. Puede tomar los valores: “aceptada”, “pendiente” o “rechazada”. No obstante, tal y como está configurada la aplicación únicamente podrá tomar los valores “aceptada” y “pendiente”. Esto se debe a que, tras ser rechazada una petición, el registro en la tabla se borra. Aun así, se considera conveniente definir un tercer estado (rechazada) por si en un futuro se desea contar con un registro de todas las peticiones de amistad hayan sido aceptadas, rechazadas o pendientes de aprobar. Este campo no puede ser nulo.

El sistema de gestión de las peticiones de amistad funciona de la siguiente manera:

1. Un usuario envía una petición a otro usuario.
2. Si el usuario que la recibe la rechaza, se borra el registro. Por el contrario, si la acepta, se actualiza el campo de ese registro a “aceptada” y se duplica el registro en la misma tabla, pero cambiando el id_emisor y el id_receptor en sentido inverso.

Tabla tokens

```

55 //***** TABLA TOKENS *****/
56 DROP TABLE IF EXISTS tokens CASCADE;
57 CREATE TABLE tokens (
58     id int auto_increment PRIMARY KEY,
59     id_usuario int,
60     valor VARCHAR(255),
61     expiracion DATETIME NOT NULL DEFAULT (NOW() + INTERVAL 7 DAY),
62     CONSTRAINT fk_id_usuario FOREIGN KEY (id_usuario) REFERENCES usuarios(id) ON DELETE CASCADE
63 );

```

Figura 29 - Tabla tokens

Esta tabla gestionará todos los tokens que pueda tener un usuario.

- **id:** identificador único para los tokens. Se autoincrementa con cada nueva inserción en la tabla. Es la clave primaria.
- **id_usuario:** identificador del usuario al que se le adjudica un token. Es una clave foránea de la tabla usuarios. Este campo no puede ser nulo.
- **valor:** es el valor del token. Este campo no puede ser nulo.
- **expiración:** fecha en la que el token deja de tener validez, su periodo de vida por defecto es de siete días a contar desde el momento de la creación. Este periodo de vida útil se puede modificar mediante la aplicación. Este campo no puede ser nulo.

Evento eliminar_tokens_expirados

```

66  /****** BARRIDO DE TOKENS EXPIRADOS ******/
67  /* evento que elimina los tokens expirados (barrido Ivez/día) */
68  CREATE EVENT eliminar_tokens_expirados
69  ON SCHEDULE EVERY 1 DAY
70  DO
71      DELETE FROM tokens WHERE expiracion < NOW();
72
73  /* podemos ver si se nos ha creado */
74  SHOW EVENTS \G;
75
76  /* IMPORTANTE: para que esto funcione es necesario habilitar el planificador de eventos CON PRIVILEGIOS DE ROOT */
77  SET GLOBAL event_scheduler = ON;
78
79  /* podemos ver si se nos ha habilitado correctamente y si se ha ejecutado */
80  SHOW GLOBAL VARIABLES WHERE Variable_name LIKE 'e%';
81  SHOW GLOBAL STATUS WHERE Variable_name LIKE 'E%';
82  /****** FIN BARRIDO ******/

```

Figura 30 - Evento eliminar_tokens_expirados

Evento que elimina todos los tokens caducados. Se ejecuta una vez por día.

Tabla respuestas

```

85  /****** TABLA RESPUESTAS ******/
86  DROP TABLE IF EXISTS respuestas CASCADE;
87  CREATE TABLE respuestas(
88      id INT AUTO_INCREMENT PRIMARY KEY,
89      id_serie INT NOT NULL,
90      id_usuario INT NOT NULL,
91      contenido VARCHAR(500) NOT NULL,
92      fecha DATETIME NOT NULL DEFAULT NOW(),
93      CONSTRAINT fk_res_usuario FOREIGN KEY (id_usuario) REFERENCES usuarios(id) ON DELETE CASCADE
94  );

```

Figura 31 - Tabla respuestas

- **id:** identificador único para las respuestas. Se autoincrementa con cada nueva inserción en la tabla. Es la clave primaria.
- **id_serie:** identificador de la serie comentada. Este dato proviene de la información obtenida de la consulta a la API, por tanto, no es una clave foránea, pero si lo seria si existiese una tabla series con este campo en nuestra base de datos. Este campo no puede ser nulo.
- **id_usuario:** identificador del usuario que publica el comentario. Es una clave foránea que proviene de la tabla usuarios. Este campo no puede ser nulo.
- **contenido:** es el texto publicado por el usuario. Este campo no puede ser nulo.
- **fecha:** señala la fecha de publicación del comentario. Se establece la fecha de forma automática al mismo tiempo que se realiza el insert en la tabla. Este campo no puede ser nulo.

Tabla chips

```

97 | /****** TABLA CHIPS *****/
98 | DROP TABLE IF EXISTS chips CASCADE;
99 | CREATE TABLE chips (
100|     id      INT AUTO_INCREMENT PRIMARY KEY,
101|     img     VARCHAR(255) NOT NULL DEFAULT 'upload/perfiles/default.png',
102|     nombre   VARCHAR(50) NOT NULL DEFAULT 'chip'
103| );

```

Figura 32 - Tabla chips

- **id:** identificador único para los chips (medallas). Se autoincrementa con cada nueva inserción en la tabla. Es la clave primaria.
- **img:** ruta de la imagen que pertenece a determinado chip. Por defecto se establece una imagen por defecto. Este campo no puede ser nulo.
- **nombre:** nombre del chip. Por defecto se establece 'chip'. Este campo no puede ser nulo.

Inserción de los datos en la tabla chips:

```

INSERT INTO chips (img, nombre) VALUES ('upload/chips/chip_oro.png', 'CHIP - GOLD');
INSERT INTO chips (img, nombre) VALUES ('upload/chips/chip_plata.png', 'CHIP - SILVER');
INSERT INTO chips (img, nombre) VALUES ('upload/chips/chip_bronce.png', 'CHIP - BRONZE');
INSERT INTO chips (img, nombre) VALUES ('upload/chips/chip_inicio.png', 'CHIP - INICIO');

```

Tabla chips_usuario

```

105 | /****** TABLA CHIPS_USUARIO *****/
106 | DROP TABLE IF EXISTS chips_usuario CASCADE;
107 | CREATE TABLE chips_usuario (
108|     id      INT AUTO_INCREMENT PRIMARY KEY,
109|     id_chip    INT NOT NULL,
110|     id_usuario  INT NOT NULL,
111|     CONSTRAINT fk_id_user FOREIGN KEY (id_usuario) REFERENCES usuarios(id) ON DELETE CASCADE,
112|     CONSTRAINT fk_id_chip FOREIGN KEY (id_chip) REFERENCES chips(id) ON DELETE CASCADE
113| );

```

Figura 33 - Tabla chips_usuario

- **id:** identificador único para los chips_usuarios. Se autoincrementa con cada nueva inserción en la tabla. Es la clave primaria.
- **Id_chip:** identificador del chip obtenido. Es una clave foránea que proviene de la tabla chips. Este campo no puede ser nulo.
- **Id_usuario:** identificador del usuario que obtiene el chip. Es una clave foránea que proviene de la tabla usuarios. Este campo no puede ser nulo.

Trigger insertar_medalla_inicio

```

128 | **** TRIGGER INSERTAR_MEDALLA_INICIO ****/
129 | drop trigger insertar_medalla_inicio;
130 | DELIMITER //
131 | CREATE TRIGGER insertar_medalla_inicio AFTER INSERT
132 | ON usuarios
133 | FOR EACH ROW
134 | BEGIN
135 |     INSERT INTO chips_usuario (id_chip, id_usuario) VALUES (4, NEW.id);
136 | END;
137 | //
138 | DELIMITER ;

```

Figura 34 - Trigger insertar_medalla_inicio

Este trigger se encarga de añadir el chip de bienvenida, convirtiéndose en el primer chip que puede obtener un usuario tras registrarse en SeriesBuddies.

Trigger sumar_medallas

```

drop trigger sumar_medallas;
DELIMITER //
CREATE TRIGGER sumar_medallas AFTER INSERT
ON respuestas
FOR EACH ROW
BEGIN
    -- Variables que se necesitan para controlar errores y resultados
    DECLARE total_series INT;
    DECLARE count_chips_series INT;
    DECLARE count_chip_plata INT;
    DECLARE count_chip_oro INT;

    -- Total de series en las que ha comentado
    SET total_series = (WITH cte AS (SELECT id_serie, id_usuario FROM respuestas GROUP BY id_usuario,
    id_serie) SELECT count(id_serie) as total_series FROM cte WHERE id_usuario = NEW.id_usuario GROUP BY
    id_usuario);

    -- Total de chips sin contar el de bienvenida
    SET count_chips_series = (SELECT COUNT(DISTINCT(id_chip)) FROM chips_usuario where id_usuario =
    NEW.id_usuario and id_chip != 4);
    -- Total de chips de plata que tiene
    SET count_chip_plata = (SELECT count(id_chip) FROM chips_usuario WHERE id_usuario = NEW.id_usuario
    AND id_chip = 2);
    -- Total de chips de oro que tiene
    SET count_chip_oro = (SELECT count(id_chip) FROM chips_usuario WHERE id_usuario = NEW.id_usuario
    AND id_chip = 1);

    -- Si no tiene ningun chip ademas del de bienvenida
    IF count_chips_series = 0 THEN
        -- Y el total de series comentadas esta entre 5 y 10, se le inserta el chip de bronce

```

```
IF (total_series >= 5 AND total_series<10) THEN
    INSERT INTO chips_usuario (id_chip, id_usuario) VALUES (3, NEW.id_usuario);
END IF;
-- Si tiene el chip de bronce y entonces...
ELSE
    -- El total de series comentadas (10 - 19) y NO tiene un chip de plata, se le inserta ese chip
    IF (total_series >=10 AND total_series<20 AND count_chip_plata = 0) THEN
        INSERT INTO chips_usuario (id_chip, id_usuario) VALUES (2, NEW.id_usuario);
    END IF;

    -- El total de series comentadas >= 20 y NO tiene un chip de oro, se le inserta ese chip
    IF (total_series >=20 AND count_chip_oro = 0) THEN
        INSERT INTO chips_usuario (id_chip, id_usuario) VALUES (1, NEW.id_usuario);
    END IF;
END IF;
END;
//  
DELIMITER ;
```

El trigger sumar medallas se ejecuta tras una inserción en la tabla respuestas y su principal función es la de añadir un chip más al usuario si ha obtenido el mínimo necesario para ello.

A continuación, se explica su funcionamiento de forma breve:

1. Un usuario deja un comentario en una serie, realizando la inserción en la tabla respuestas provocando la ejecución del trigger.
2. Se declaran las variables que almacenarán el total de medallas que posee ese usuario.
3. Se recoge y almacena en total_series, el total de las series en las que ha comentado ese usuario.
4. Se recoge y almacena en count_chips_series, count_chips_plata y count_chips_oro, si el usuario posee algún chip de bronce, plata u oro respectivamente.
5. Si solo posee el chip de bienvenida, se revisa si el total de series comentadas está entre los valores 5 y 9. Si es así, se le inserta el chip de bronce.
6. Si no posee el chip de plata y el total de series comentadas está entre los valores 10 y 19, se le inserta ese chip.
7. Si no posee el chip de oro y el total de series comentadas es superior o igual a 20, se le inserta ese chip.

Trigger restar_medallas

```

drop trigger restar_medallas;
DELIMITER //
CREATE TRIGGER restar_medallas AFTER DELETE
ON respuestas
FOR EACH ROW
BEGIN
    -- Variables que se necesitan para controlar errores y resultados
    DECLARE total_series INT;
    DECLARE count_chip_bronce INT;
    DECLARE count_chip_plata INT;
    DECLARE count_chip_oro INT;

    -- Total de series en las que ha comentado
    SET total_series = (WITH cte AS (SELECT id_serie, id_usuario FROM respuestas GROUP BY id_usuario,
    id_serie) SELECT count(id_serie) as total_series FROM cte WHERE id_usuario = OLD.id_usuario GROUP BY
    id_usuario);

    -- Total de chips de bronce que tiene
    SET count_chip_bronce = (SELECT COUNT(id_chip) FROM chips_usuario WHERE id_usuario = OLD.id_usuario
    AND id_chip = 3);
    -- Total de chips de plata que tiene
    SET count_chip_plata = (SELECT COUNT(id_chip) FROM chips_usuario WHERE id_usuario = OLD.id_usuario
    AND id_chip = 2);
    -- Total de chips de oro que tiene
    SET count_chip_oro = (SELECT COUNT(id_chip) FROM chips_usuario WHERE id_usuario = OLD.id_usuario
    AND id_chip = 1);

    -- Si el total de series comentadas es menos de 5 y tiene el chip de bronce, se elimina el chip
    IF (total_series < 5 AND count_chip_bronce = 1) THEN
        DELETE FROM chips_usuario WHERE id_chip=3 and id_usuario=OLD.id_usuario;
    END IF;

    -- El total de series comentadas (10 - 19) y tiene el chip de plata, se elimina el chip
    IF (total_series < 10 AND count_chip_plata = 1) THEN
        DELETE FROM chips_usuario WHERE id_chip=2 and id_usuario=OLD.id_usuario;
    END IF;

    -- El total de series comentadas >= 20 y tiene el chip de oro, se elimina el chip
    IF (total_series < 20 AND count_chip_oro = 1) THEN
        DELETE FROM chips_usuario WHERE id_chip=1 and id_usuario=OLD.id_usuario;
    END IF;
END;
// 
DELIMITER ;

```

El trigger restar medallas se ejecuta tras eliminar un registro en la tabla respuestas y su principal función es la de eliminar un chip al usuario si no se ha llegado el mínimo necesario para ello.

A continuación, se explica su funcionamiento de forma breve:

1. Un usuario elimina un comentario de una serie, realizando un delete en la tabla respuestas provocando la ejecución del trigger.
2. Se declaran las variables que almacenarán el total de medallas y comentarios que posee ese usuario.
3. Se recoge y almacena en total_series, el total de las series en las que ha comentado ese usuario.
4. Se recoge y almacena en count_chips_bronce, count_chips_plata y count_chips_oro, si el usuario posee algún chip de bronce, plata u oro respectivamente.
5. Si posee el chip de bronce y el total de series comentadas es inferior a 5, se le elimina ese chip.
6. Si posee el chip de plata y el total de series comentadas es inferior 10, se le elimina ese chip.
7. Si posee el chip de oro y el total de series comentadas es inferior a 20, se le elimina ese chip.

ESTRUCTURA DEL PROYECTO, DE LAS PÁGINAS Y PROGRAMACIÓN DE LA LÓGICA

Estructura de ficheros

La estructura que seguirá el proyecto será la siguiente:

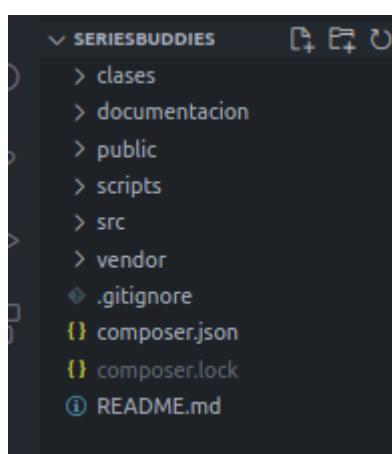


Figura 35 - Estructura del proyecto

- **clases:** va a tener todas las clases (menos una) que se usen en el proyecto.

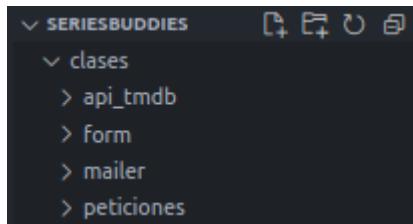


Figura 36 - Carpeta *clases*

- **documentación:** va a contener la documentación del proyecto (wireframes, mockups, metodología), así como la memoria del proyecto.

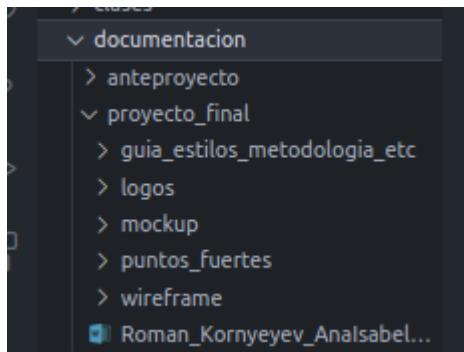


Figura 37 - Carpeta *documentacion*

- **public:** va a contener todo aquel contenido que sea accesible de cara al público, es decir, todas las páginas (archivos PHP, hojas de estilo CSS, archivos JS e imágenes).

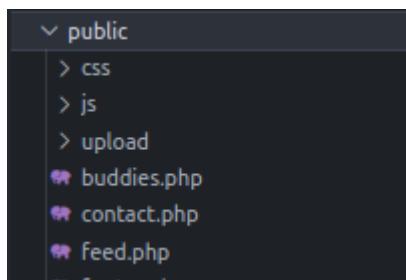


Figura 38 - Carpeta *public*

- **scripts:** tablas de bases de datos e inserts necesarios.



Figura 39 - Carpeta *scripts*

- **src:** va a contener utilidades que se usan y repiten en todas las páginas (sesion_start(), entre otros).

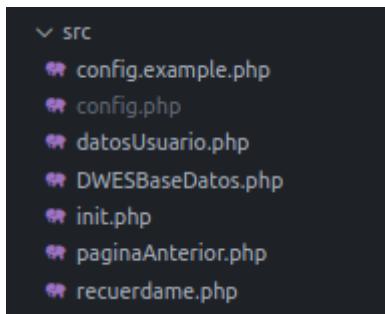


Figura 40 - Carpeta src

- **vendor:** dependencias del proyecto (en nuestro caso solamente PHPMailer).

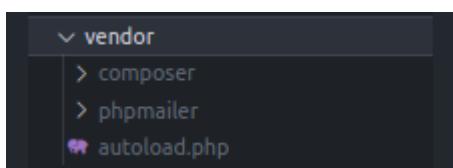


Figura 41 - Carpeta vendor

Librería de formularios PHP

Librería de formularios PHP que ayuda a validar los campos de los formularios, evitando así posibles inyecciones SQL y/o ataques cross-site scripting. La librería está orientada a objetos, teniendo una clase principal “Formulario”, seguida de clases secundarias, que corresponden a cada tipo de campo (numérico, texto, etc.):

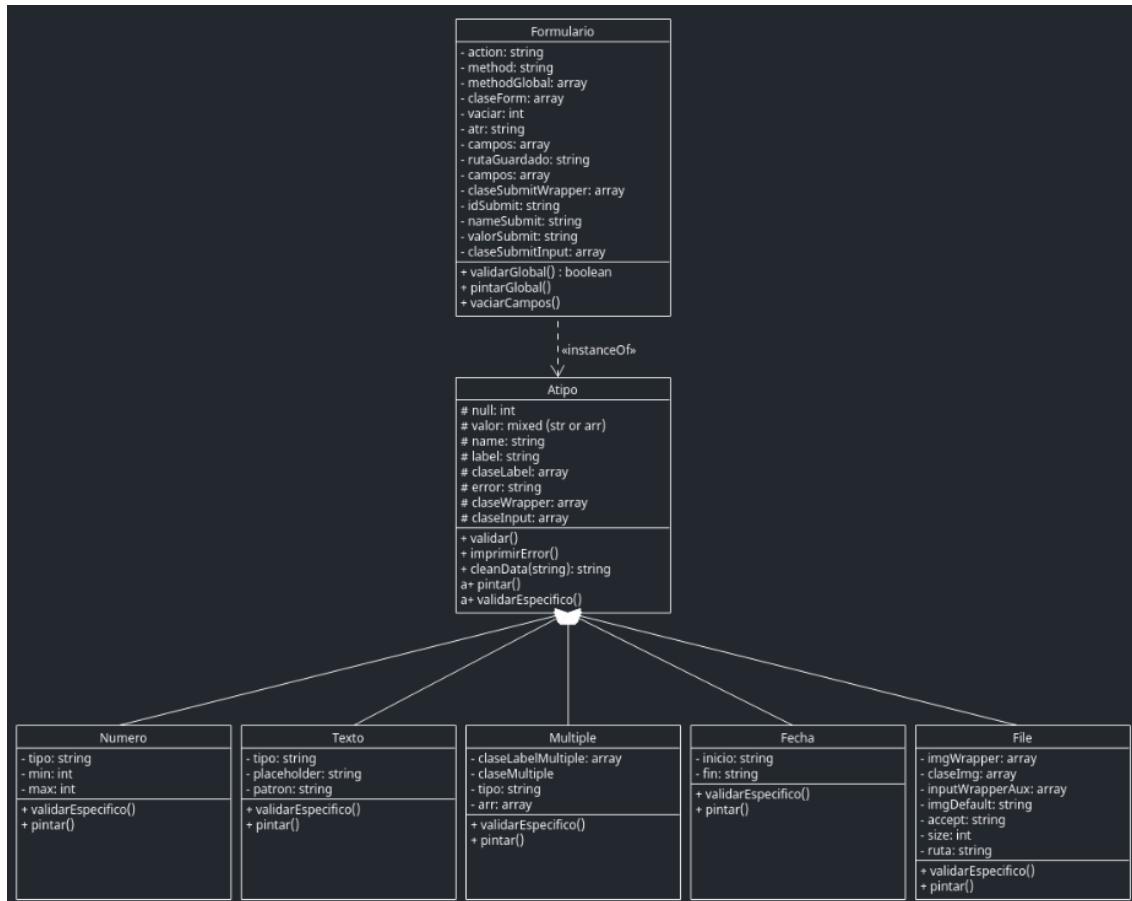


Figura 42 - Esquema UML de la librería de formularios

- **Formulario:** clase de la que se instancia un objeto que va a generar y pintar un formulario (<form>) con un botón submit, con todos los parámetros que se le pasen, tales como: action, method (GET/POST), id y clases de los elementos HTML, etc.
- **Atipo:** clase abstracta que contiene todos los datos que se van a repetir en las clases que hereden de ella (clases de cada tipo de input).
- **Numero:** clase para pintar y validar inputs de tipo number y range. Valida que los valores introducidos sean números y que estén entre un mínimo y un máximo especificado.
- **Texto:** clase para pintar y validar inputs de tipo text, textarea, password y hidden. Valida que los valores introducidos no lleven código malicioso (evitando ataques de cross-site scripting) y que no se sobrepasen un máximo de longitud especificada.

- **Multiple:** clase para pintar y validar inputs de tipo checkbox, radio y select. Valida que los valores enviados sean los correctos y evita que se envíen valores que haya podido adulterar un usuario malicioso editando el HTML desde el navegador.
- **Fecha:** clase para pintar y validar inputs de tipo date. Comprueba que la fecha esté entre un periodo de tiempo especificado.
- **File:** clase para pintar y validar inputs de tipo file. Comprueba que el archivo sea de la clase especificada y no supere el tamaño máximo especificado.

Además de todo lo anteriormente comentado, todos los campos cuentan con errores personalizados que se pintan automáticamente debajo de los propios campos en el caso de que el usuario haya introducido datos no válidos o haya dejado algún input en blanco. También se puede especificar si los inputs se pueden dejar en blanco o no.

Un ejemplo de instanciación del formulario de login:

```

15 // ===== FORM DE LOGIN =====
16 // ACTION      METHOD      clases-css-form  ¿Vacio al validar?   atr-extra(para forms con img)  CAMPOS
17 $formulario = new Formulario("login.php", Formulario::METHOD_POST, ["form"],           Formulario::VACIAR_NO, "", array( // =====
18 // ===== COMUN ===== // =====
19 // ¿Puede estar vacío? valor name  label    clases-css-label    clases-css-wrapper    clases-css-input    tipoCampo
20 $email = new Texto  (Atipo::NULL_NO, null,"email", "Email", ["label","label--text"], ["input-wrapper"], ["input","shadow-lightgray"], Texto::TYPE_TE
21 $pass = new Texto  (Atipo::NULL_NO, null,"contra", "Password",["label","label--text"], ["input-wrapper"], ["input","shadow-lightgray"], Texto::TYPE_PS
22 // claseLabel-claseInput
23 $recuerdame = new Multiple(Atipo::NULL_SI, null,"recuerdame",NULL,[{"label","label--checkbox"}, {"input-wrapper"}, [{"checkbox"}, {"label", "label--checkbox"}], [{"label","label--checkbox"}]);
24 // === SUMMIT ===
25 // claseLabel-claseInput
26 ), ["input-wrapper","input-wrapper--submit"], "enviar", "enviar", "LOGIN", ["btn", "btn--primary", "shadow-lightgray"]);
27
28 <?php $formulario->intarGlobal(); ?>

```

Figura 43 - Instanciación de un objeto Formulario

Bienvenido

Email
hola

Email no válido

Password

El campo contra no puede estar vacío

Recuérdame

LOGIN

¿Aun no eres buddy? [Regístrate](#)
¿Contraseña olvidada? [Recupérala](#)

Figura 44 - Formulario de login

Clase DWESBaseDatos

Esta clase tiene como propósito general facilitar las conexiones y consultas a la base de datos. Con DWES se puede parametrizar todas las consultas que se planteen y en consecuencia, se reduce código. En esta clase se maneja principalmente la información referente a los usuarios y sus relaciones con el resto de las tablas (peticiones de amistad y número de buddies, número de comentarios realizados, número de series que siguen y total de chips obtenidos).

Para comenzar se han declarado tres variables privadas que se encargan de la conexión y ejecución de las consultas enviadas a la base de datos:

```
private $conexion = null;
private $sentencia = null;
private $executed = false;
```

Figura 45 - Atributos clase DWESBaseDatos

Se han definido al mismo tiempo una serie de constantes que señalan datos como:

- La longitud máxima de los tokens generados.
- El tipo de nivel de privilegios que posee un usuario.
- Si el usuario ha sido verificado para poder hacer uso de su cuenta.
- El estado en el que se encuentra una petición de amistad.
- El número de registros por páginas a mostrar para definir la paginación.
- El número máximo de páginas que mostrar en la paginación.
- El número máximo de usuarios a mostrar en el enlace “Listado de buddies” y “Super Buddies” que aparecen en las páginas feed y serie respectivamente.
- La ruta del dominio.

```
const LONG_TOKEN = 64;
const ADMIN = "admin";
const USUARIO = "usuario";
const VERIFICADO_SI = "si";
const VERIFICADO_NO = "no";
const PENDIENTE = "pendiente";
const ACEPTADA = "aceptada";

const REGISTROS POR_PAGINA = 20;
const MAX_BUDDIES_FEED = 3;
const MAX_PAG_PAGINADOR = 3;

const RUTA_DOMINIO_BASE = "www.seriesbuddies.es";
```

Figura 46 - Constantes públicas clase DWESBaseDatos

Se utiliza el patrón Singleton para poder hacer uso de la clase en proyectos más grande.

Mediante la función inicializa, se establece la conexión con la base de datos.

```
function inicializa(
    $basedatos,           // Nombre debe ser especificado o el archivo si es SQLite
    $usuario = 'root',    // Ignorado si es SQLite
    $pass = '1234',       // Ignorado si es SQLite
    $motor = 'mysql',
    $serverIp = 'localhost',
    $charset = 'utf8mb4',
    $options = null
) {
    if($motor != "sqlite") {
        $cadenaConexion = "{$motor}:host={$serverIp};dbname={$basedatos};charset={$charset}";
    } else {
        $cadenaConexion = "{$motor}:{$basedatos}";
    }

    if($options == null){
        $options = [
            PDO::ATTR_EMULATE_PREPARES => false, // La preparación de las consultas no es simulada
                                                // más lento pero más seguro
            PDO::ATTR_ERRMODE          => PDO::ERRMODE_EXCEPTION, // Cuando se produce un error
                                                // salta una excepción
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC, // Cuando traemos datos lo hacemos como array asociativo
        ];
    }

    try {
        if($motor != "sqlite") {
            $this->conexion = new PDO($cadenaConexion, $usuario, $pass, $options);
        } else {
            $this->conexion = new PDO($cadenaConexion, null, null, $options);
        }
    } catch (Exception $e) {
        error_log($e->getMessage());
        exit('No ha sido posible la conexión');
    }
}
```

Figura 47 - Función inicializa de DWESBaseDatos

La función ejecuta permite ejecutar una consulta preparada con parámetros posicionales.

```
function ejecuta(string $sql, ...$parametros) {
    $this->sentencia = $this->conexion->prepare($sql);

    if($parametros == null){
        $this->executed = $this->sentencia->execute();
        return;
    }

    if($parametros != null && is_array($parametros[0])) {
        $parametros = $parametros[0]; // Si nos pasan un array lo usamos como parámetro
    }

    $this->executed = $this->sentencia->execute($parametros);
}
```

Figura 48 - Función ejecuta de DWESBaseDatos

Las siguientes funciones sirven para manejar los datos y se han dividido por grupos en función de si la consulta es un SELECT, INSERT, UPDATE o DELETE.

```
// ===== SELECTS =====

//Obtiene los datos personales de un usuario en concreto dado su id
public static function obtenInfoBuddieTarjeta ($db, $idUsuario) {
    $db->ejecuta("SELECT
        u.id, u.nombre, u.img,
        CONCAT('@',u.nombre, '#', u.id)'alias',
        u.descripcion,
        DATE_FORMAT(fecha_alta, '%d %b %Y', 'es-ES')'fecha',
        u.privilegio
        FROM usuarios u
        WHERE u.id=?",
        $idUsuario);

    return $db->obtenDatos()[0];
}
```

Figura 49 - Función obtenInfoBuddieTarjeta de DWESBaseDatos

En este ejemplo, la función obtenInfoBuddieTarjeta recibe el singleton y el identificador del usuario. Se emplea la función ejecuta para ejecutar la consulta y finalmente, se hace uso de la función obtenDatos (recogiendo el primer parámetro del array devuelto) para devolver la información.

Para el resto de las sentencias (insert, update y delete) se sigue el mismo proceso. La diferencia radica en que el parámetro que devuelven es un booleano, útil para conocer si la sentencia se ha ejecutado correctamente o no.

```
// ===== INSERTS =====

public static function insertarUsuario($db, $nombre, $contra, $correo, $privilegio, $verificado) : bool
{
    $db->ejecuta(
        "INSERT INTO usuarios (nombre, contra, correo, privilegio, verificado) VALUES (?, ?, ?, ?, ?);",
        $nombre, $contra, $correo, $privilegio, $verificado
    );
    if ($db->getExecuted()) {
        return true;
    }else{
        return false;
    }
}
```

Figura 50 - insertarUsuario de DWEBaseDatos

```
// ===== UPDATES =====

public static function actualizarRespuesta($db, $idRespuesta, $mensaje) : bool
{
    $db->ejecuta(
        "UPDATE respuestas SET contenido=? WHERE id=?",
        $mensaje, $idRespuesta
    );
    if ($db->getExecuted()) {
        return true;
    }else{
        return false;
    }
}
```

Figura 51 - Función *actualizarRespuesta* de DWESBaseDatos

```
// ===== DELETES =====

public static function eliminaTokensUsuario($db, $id) : bool
{
    $db->ejecuta(
        "DELETE FROM tokens WHERE id_usuario=?",
        $id
    );
    if ($db->getExecuted()) {
        return true;
    }else{
        return false;
    }
}
```

Figura 52 - Función *eliminaTokensUsuario* de DWESBaseDatos

En esta clase también definimos cómo será la paginación, el total de resultados a mostrar y las redirecciones que aparecen.

```
// ===== EXTRA (PAGINACION, ETC.) =====

public static function obtenLimitesPaginacion ($paginaActual, $totalPaginas) {
    $limites['primera'] = $paginaActual - ($paginaActual % self::MAX_PAG_PAGINADOR) + 1;

    if ($limites['primera'] > $paginaActual) {
        $limites['primera'] = $limites['primera'] - self::MAX_PAG_PAGINADOR;
    }

    if ($limites['primera'] + (self::MAX_PAG_PAGINADOR-1) > $totalPaginas ) {
        $limites['ultima'] = $totalPaginas;
    } else {
        $limites['ultima'] = $limites['primera'] + (self::MAX_PAG_PAGINADOR-1);
    }

    return $limites;
}
```

Figura 53 - Función `obtenLimitesPaginacion` de `DWESBaseDatos`

```
public static function obtenPaginacion($paginaBase, $paginaActual, $totalPaginas, $argumentos = array()) : String
{
    //variables a llenar
    $paginacion = "";
    $args = "";

    //si el array tiene valores, llenaremos la variable args para meterla posteriormente al link
    if(!empty($argumentos)){
        foreach ($argumentos as $key => $value) {
            if($value != ""){
                $args .= "&" . $key . "=" . $value;
            }
        }
    }

    //Botones << y >> para aquellas que no sean la primera pagina
    if ($paginaActual != 1) {
        $paginacion = "
            <a href='./$paginaBase.php?pagina=1$args' class='btn btn--pagination-size'>&lt;&lt;/a>
            <a href='./$paginaBase.php?pagina=".($paginaActual-1)."{$args}' class='btn btn--pagination-size'>&lt;&lt;/a>
        ";
    }

    //Pagina actual
    $paginacion = $paginacion . "<span class='primary-color'>&ampnbsp". $paginaActual ." de ".$totalPaginas ."&ampnbsp</span>";

    //Boton > y >>
    if ($paginaActual != $totalPaginas) {
        $paginacion = $paginacion . "
            <a href='./$paginaBase.php?pagina=".($paginaActual+1)."{$args}' class='btn btn--pagination-size'>&gt;&gt;/a>
            <a href='./$paginaBase.php?pagina=".$totalPaginas."{$args}' class='btn btn--pagination-size'>&gt;&gt;/a>
        ";
    }

    return "<div class='pages'>". $paginacion . "</div>";
}
```

Figura 54 - Función `obtenPaginacion` de `DWESBaseDatos`

Clase TMDB

Para la realización de este proyecto, se ha necesitado usar la API de The Movie Database con la que poder obtener la información de las series existentes junto con su ficha técnica y los géneros a los que pertenecen.

Esta clase posee una serie de constantes que se utilizan en la mayoría de las funciones desarrolladas y que, por tanto, al tenerlas separadas e identificadas como variables, se hace más sencillo, cómodo y rápido el poder modificar sus valores con posterioridad.

```
//VARIABLES Y CONSTANTES
const API_KEY = '';
const API_URL = 'https://api.themoviedb.org/3/';
const TOKEN = 'eyJhbGwiOiIzZmQ
dF1hwzUErQB-Ad';
const MAX_PAGINAS = 500;
const BASE_URL_IMG='https://image.tmdb.org/t/p/';
const WIDTH_BACKDROP = 'w1280';
const WIDTH_POSTER = 'w500';
const WIDTH_POSTER_MIN = 'w300';
public $adult = false;
public $lang = 'es-ES';
const LISTADO_GENEROS = [
    10759 => 'Action & Adventure',
    16 => 'Animación',
    35 => 'Comedia',
    80 => 'Crimen',
    99 => 'Documental',
    18 => 'Drama',
    10751 => 'Familia',
    10762 => 'Kids',
    9648 => 'Misterio',
    10763 => 'News',
    10764 => 'Reality',
    10765 => 'Sci-Fi & Fantasy',
    10766 => 'Soap',
    10767 => 'Talk',
    10768 => 'War & Politics',
    37 => 'Western',
    0 => 'Búsqueda'
];

```

Figura 55 - Constantes de la clase clase TMDB

La siguiente función es la que realiza una petición a la API a través de una url específica con la consulta que se pretende realizar. Devuelve dicha información en formato array mediante la librería CURL.

```
//Lanza la petición a la api con la url especificada y lo devuelve como un array
public function peticionHTTP ($url) {
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_ENCODING, '');
    curl_setopt($curl, CURLOPT_MAXREDIRS, 10);
    curl_setopt($curl, CURLOPT_TIMEOUT, 30);
    curl_setopt($curl, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_1);
    curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'GET');
    curl_setopt($curl, CURLOPT_HTTPHEADER, [
        "Authorization: ".self::TOKEN,
        "accept: application/json"
    ]);

    $response = curl_exec($curl);
    curl_close($curl);
    return (array) json_decode($response), true);
}
```

Figura 56 - Función peticionHTTP de TMDB

El resto de esta clase se divide en dos grandes bloques, generación de la url para la obtención de la información y funciones que recogen y filtran los datos para poder manejarlos en las diferentes páginas.

```
//Devuelve la url para buscar la info de la serie por su ID.
public function urlSerieVideoID ($serieID) {
    return self::API_URL.'tv/'.$serieID.'?append_to_response=videos&language='.$this->lang;
}

//Devuelve la url para buscar la info de la serie por su ID.
public function urlSerieID ($serieID) {
    return self::API_URL.'tv/'.$serieID.'?language='.$this->lang;
    //return self::API_URL.'tv/'.$serieID.'?append_to_response=videos&language='.$this->lang;
}
```

Figura 57 - Funciones url de TMDB

```
//Devuelva la info de una serie a partir de su ID
public function getSerieID ($serieID) {
    $urlSerie = $this->urlSerieID($serieID);
    $serie = $this->peticionHTTP($urlSerie);

    $serieData['serieTitle'] = $serie['name'];

    //Si no hay foto banner para la serie se pone el logo de seriesbuddies por defecto
    if ($serie['backdrop_path'] == '') {
        $serieData['backdrop'] = 'upload/logos/logo-backdrop.png';
    } else {
        $serieData['backdrop'] = self::BASE_URL_IMG.self::WIDTH_BACKDROP.$serie['backdrop_path'];
    }

    //Si no hay descripción para la serie se pone el texto por defecto
    if ($serie['overview'] == '') {
        $serieData['seriePlot'] = 'Lo sentimos, pero en estos momentos no contamos con la trama de esta serie. Estamos trabajando para disponer de ella en la máxima brevedad posible.';
    } else {
        $serieData['seriePlot'] = $serie['overview'];
    }
    $serieData['serieAirDate'] = $serie['first_air_date'];
    $serieData['serieGenres'] = $serie['genres'];
    $serieData['video'] = 'https://www.youtube.com/watch?v='.$serie['videos'][0]['results'][0]['key'];

    return $serieData;
}
```

Figura 58 - Función getSerieID de TMDB

Clase Petición

Esta clase gestiona todo lo relacionado con las peticiones de amistad. Los footers de las tarjetas de los usuarios y las peticiones que llegan al perfil del usuario. Tiene un único atributo booleano que permite saber si el usuario es administrador o no (para pintar funciones extra). Posee varias constantes para evitar el uso de “strings mágicos”, por ejemplo, las constantes de footers permiten saber qué footer pintar (los footers de buddies.php son distintos a los del profile.php):

```

5   class Peticion{
6
7     private $esAdmin;
8
9     public const FOOTER_BUDDIES = 1;
10    public const FOOTER_PROFILE = 2;
11    public const NOTIFICACION_PROFILE = 3;
12
13    public const ESTADO_PENDIENTE = "pendiente";
14    public const ESTADO_ACEPTADO = "aceptada";
15
16    public const ACCION_ENVIAR = "enviar";
17    public const ACCION_CANCELAR = "cancelar";
18    public const ACCION_ACEPTAR = "aceptar";
19    public const ACCION_RECHAZAR = "rechazar";
20    public const ACCION_ELIMINAR = "eliminar";
21
22
23    public function __construct($esAdmin = false){
24      $this->esAdmin = $esAdmin;
25    }

```

Figura 59 - Atributos, constantes y constructor clase Peticion

Esta clase contiene funciones que pintan todos los footers de las tarjetas, así como tarjetas de petición correspondientes:

```

27   public function pintaSesionNoIniciada($id) : String
28   {
29     return "
30       <div class='buddy__footer-internal-layer' id='$id'>
31         <div class='buddy__footer buddy__footer--primary grid-col-1'>
32           <button class='btn btn--card' onclick='subir(this)'>
33             <i class='fa-solid fa-id-card'></i>&nbsp;
34             <span class='primary-font'>Volver</span>
35             &nbsp;<i class='fa-solid fa-arrow-down'></i>
36           </button>
37         </div>
38         <div class='buddy__footer buddy__footer--primary grid-col-1'>
39           <a class='btn btn--card' href='./profile.php?id=$id'>Perfil</a>
40         </div>
41       </div>
42     ";
43   }

```

Figura 60 - Función pintaSesionNoIniciada de Peticion

Esto se gestiona de la siguiente manera, con condicionales en PHP. Dependiendo de si el usuario tiene sesión iniciada, es administrador o no y de su estado de amistad, se pintarán unos botones con unas funciones (de JS) u otras:

```

188     <!-- CARD FOOTER -->
189     <div class="buddy_footer-external-layer">
190         <?php
191             //si el user no tiene sesión iniciada
192             if (!$sesionIniciada || $_SESSION['id'] == $value['id']) {
193                 echo $peticionFooter->pintaSesionNoIniciada($value['id']);
194             }
195             //si el user SI TIENE La sesión iniciada
196             }else{
197
198                 //obtenemos info sobre el estado de petición de amistad
199                 $peticion = DWESEBaseDatos::obtenPeticion($db, $_SESSION['id'], $value['id']);
200
201                 //si ninguno ha mandado petición de amistad
202                 if ($peticion == "" || $peticion == null) {
203                     echo $peticionFooter->pintaAmistadNula($value['id'], $paginaActual, $totalPaginas, Peticion::FOOTER_BUDDIES);
204
205                     //si el user actual (SESIÓN) ha ENVIADO peti al user seleccionando
206                     }else if($peticion['estado'] == DWESEBaseDatos::PENDIENTE && $peticion['id_emisor'] == $_SESSION['id']) {
207                         echo $peticionFooter->pintaAmistadEnviada($value['id'], $paginaActual, $totalPaginas, Peticion::FOOTER_BUDDIES);
208
209                     //si el user actual (SESIÓN) ha RECIBIDO peti del user seleccionando
210                     }else if($peticion['estado'] == DWESEBaseDatos::PENDIENTE && $peticion['id_receptor'] == $_SESSION['id']) {
211                         echo $peticionFooter->pintaAmistadRecibida($value['id'], $paginaActual, $totalPaginas, Peticion::FOOTER_BUDDIES);
212
213                     //si son AMIGOS
214                     }else if($peticion['estado'] == DWESEBaseDatos::ACEPTADA) {
215                         echo $peticionFooter->pintaAmistadMutua($value['id'], $paginaActual, $totalPaginas, Peticion::FOOTER_BUDDIES);
216                     }
217                 }
218             ?>
219         </div>

```

Figura 61 - Card footer

Posteriormente estas funciones de peticiones se van a gestionar con AJAX, para hacer la página más fluida y agradable, evitando recargar la página entera al mandar una petición de amistad:

```

function peticion(elemento, id, accion, paginaActual=1, totalPaginas=1, tipo){
    let globalElementOverwrite = elemento.parentNode.parentNode.parentNode;

    elemento.className = "";
    elemento.onclick = null;
    elemento.innerHTML = "<i class='fa-solid fa-spinner rotate-infinite font-size-default'>/i>";
    disableSiblingClicks(elemento);
    if (accion === "enviar") {elemento.classList.add("btn-no-clickable-primary");}
    else if (accion === "aceptar") {elemento.classList.add("btn-no-clickable-success");}
    else{elemento.classList.add("btn-no-clickable-error");}

    //nuevo objeto XMLHttpRequest
    var xhttp = new XMLHttpRequest();

    //cuando hay un cambio de estado
    xhttp.onreadystatechange = async function() {

        //si todo está ok, 4 (respuesta está completa) y 200 (solicitud HTTP correcta)
        if (this.readyState === 4) {
            await delay(1000);
            if (this.status === 200) {
                // respuesta del handler, la pintamos en el elemento
                elemento.parentNode.innerHTML = "<div class='btn-no-clickable-success pos-absolute opacity-fade'><i class='fa-solid fa-check'>/i></div>";
                await delay(1000);
                globalElementOverwrite.innerHTML = this.responseText;
            }
            //Error en el handler? Pintamos un error
            else if (this.status === 500 || this.status === 400){
                elemento.parentNode.innerHTML = "<div class='btn-no-clickable-error pos-absolute'>ERROR<br><i class='fa-solid fa-xmark'>/i></div>";
            }
        }
    };

    //se abre una conexión POST al controlador
    xhttp.open("POST", "peticiones_handler.php", true);
    //establecemos el tipo de contenido de la solicitud
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    //pasamos La petición POST
    xhttp.send(`accion=${accion}&id=${id}&pagina_actual=${paginaActual}&total_paginas=${totalPaginas}&tipo=${tipo}`);
}

```

Figura 62 - Función peticion (JS)

Según los parámetros recibidos, se manda una petición POST a un handler (peticiones_handler.php) que devolverá una respuesta. Con el fin de evitar posibles errores, este handler comprueba que el usuario que manda la petición tenga la sesión iniciada.

```
1 <?php
2
3 // ***** INIT *****
4 require_once("../src/init.php");
5
6 // ***** PETICIONES *****
7 use clases\peticiones\Peticion;
8
9 $peticionFooter = new Peticion($esAdmin);
10
11 if ($sesionIniciada) {
12     if (isset($_POST['accion'])) {
13         switch (strval($_POST['accion'])) {
14             case 'enviar':
15                 $db->ejecuta(
16                     "INSERT INTO peticiones (id_emisor, id_receptor, estado) VALUES (?, ?, ?);",
17                     $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
18                 );
19                 echo $peticionFooter->pintaAmistadEnviada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
20                 break;
21
22             case 'cancelar':
23                 $db->ejecuta(
24                     "DELETE FROM peticiones WHERE id_receptor=? AND id_emisor=? AND estado=?;",
25                     $_POST['id'], $_SESSION['id'], Peticion::ESTADO_PENDIENTE
26                 );
27                 echo $peticionFooter->pintaAmistadNula($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
28                 break;
29
30             case 'aceptar':
31                 // ACEPTAR AMISTAD
32                 $db->ejecuta(
33                     "UPDATE peticiones SET estado=? WHERE id_emisor=? AND id_receptor=? AND estado=?;",
34                     Peticion::ESTADO_ACEPTADA, $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
35                 );
36                 echo $peticionFooter->pintaAmistadAceptada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
37                 break;
38
39             case 'rechazar':
40                 // RECHAZAR AMISTAD
41                 $db->ejecuta(
42                     "UPDATE peticiones SET estado=? WHERE id_emisor=? AND id_receptor=? AND estado=?;",
43                     Peticion::ESTADO_RECHAZADA, $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
44                 );
45                 echo $peticionFooter->pintaAmistadRechazada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
46                 break;
47
48             case 'eliminar':
49                 $db->ejecuta(
50                     "DELETE FROM peticiones WHERE id_emisor=? AND id_receptor=? AND estado=?;",
51                     $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
52                 );
53                 echo $peticionFooter->pintaAmistadEliminada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
54                 break;
55
56             case 'ver':
57                 // VER AMISTAD
58                 $db->ejecuta(
59                     "SELECT * FROM peticiones WHERE id_emisor=? AND id_receptor=? AND estado=?;",
60                     $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
61                 );
62                 echo $peticionFooter->pintaAmistadVerificada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
63                 break;
64
65             case 'aceptar_requerimiento':
66                 // ACEPTAR REQUERIMIENTO
67                 $db->ejecuta(
68                     "UPDATE peticiones SET estado=? WHERE id_emisor=? AND id_receptor=? AND estado=?;",
69                     Peticion::ESTADO_ACEPTADA, $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
70                 );
71                 echo $peticionFooter->pintaAmistadAceptada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
72                 break;
73
74             case 'rechazar_requerimiento':
75                 // RECHAZAR REQUERIMIENTO
76                 $db->ejecuta(
77                     "UPDATE peticiones SET estado=? WHERE id_emisor=? AND id_receptor=? AND estado=?;",
78                     Peticion::ESTADO_RECHAZADA, $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
79                 );
80                 echo $peticionFooter->pintaAmistadRechazada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
81                 break;
82
83             case 'aceptar_solicitud':
84                 // ACEPTAR SOLICITUD
85                 $db->ejecuta(
86                     "UPDATE peticiones SET estado=? WHERE id_emisor=? AND id_receptor=? AND estado=?;",
87                     Peticion::ESTADO_ACEPTADA, $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
88                 );
89                 echo $peticionFooter->pintaAmistadAceptada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
90                 break;
91
92             case 'rechazar_solicitud':
93                 // RECHAZAR SOLICITUD
94                 $db->ejecuta(
95                     "UPDATE peticiones SET estado=? WHERE id_emisor=? AND id_receptor=? AND estado=?;",
96                     Peticion::ESTADO_RECHAZADA, $_SESSION['id'], $_POST['id'], Peticion::ESTADO_PENDIENTE
97                 );
98                 echo $peticionFooter->pintaAmistadRechazada($_POST['id'], $_POST['pagina_actual'], $_POST['total_paginas'], $_POST['tipo']);
99                 break;
100
101         }
102     }
103 }
104
105 // CERRAR SESIÓN
106 session_destroy();
107
108 // CARGAR VISTA
109 $peticionFooter->cargarVista();
```

Figura 63 - peticiones_handler.php

Esta respuesta es una función del objeto Peticiones, la cual pintará un footer de tarjeta u otro. Es decir, el usuario1 no ha enviado petición a usuario2, le saldrá el siguiente footer:

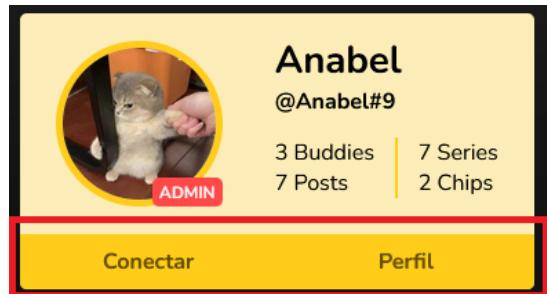


Figura 64 - Tarjeta buddy (sin petición)

Si le da a conectar, gracias a la petición AJAX se verá una breve transición que actualizará únicamente esa parte de la página, mandándole la petición al usuario y actualizando el footer a este estado (enviada), si el usuario da click al botón “Enviada”, se deslizará para abajo y nos mostrará más opciones (cancelar la petición en este caso):

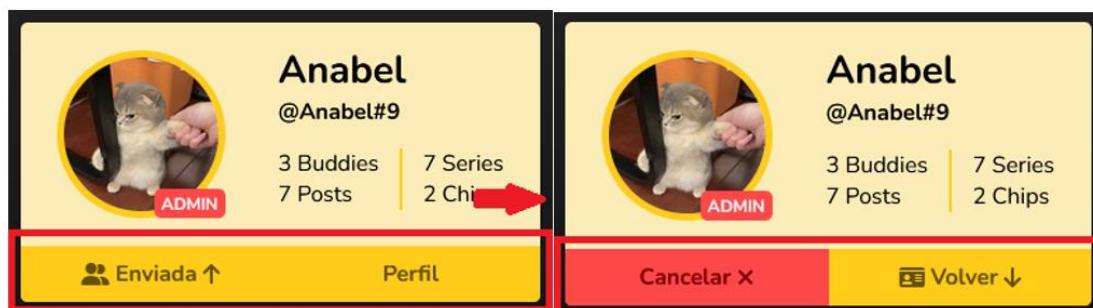


Figura 65 - Tarjeta buddy (petición enviada)

En otro posible caso, si el usuario1 recibe una petición de amistad de usuario2, saldría este otro footer. Si da click sobre el botón “Recibida”, se deslizará para abajo y mostrará más opciones (aceptar o rechazar en este caso):

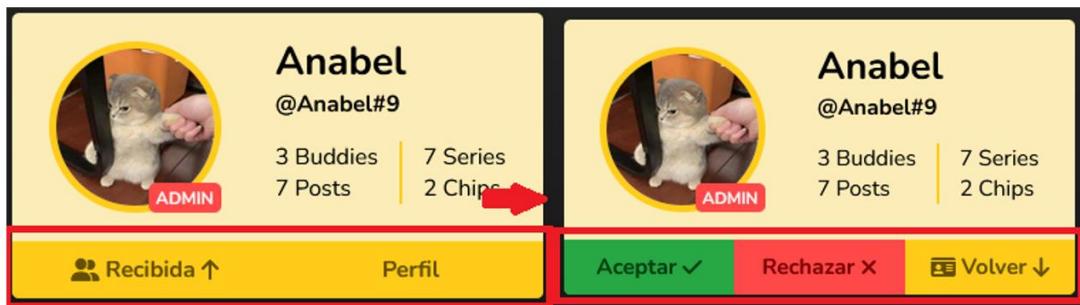


Figura 66 - Tarjeta buddy (petición recibida)

Ejemplo de peticiones que llegan al perfil que también funcionan con AJAX:

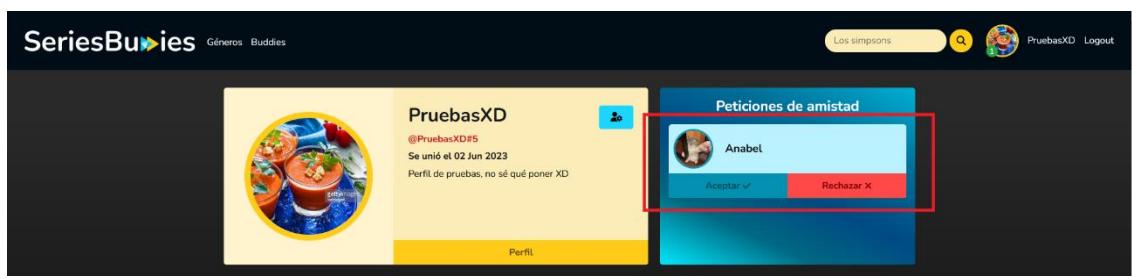


Figura 67 – Peticiones recibidas en el perfil

Clase Mailer

Clase dedicada a enviar correos automatizados a los usuarios. No tiene ni atributos, ni constructor, posee únicamente métodos estáticos que interactúan (o no) con la librería PHPMailer:

```
class Mailer{
    public static function sendEmail($correo, $asunto, $cuerpo)
    {
        //Create an instance; passing `true` enables exceptions
        global $CONFIG;
        $mail = new PHPMailer(true);

        try {
            //Server settings
            $mail->isSMTP();
            // $mail->SMTPDebug = SMTP::DEBUG_SERVER;
            $mail->SMTPAuth = true;
            $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
            $mail->Port = 465;
            $mail->Host = 'smtp.gmail.com';
            $mail->Username = $CONFIG['mail_user'];
            $mail->Password = $CONFIG['mail_pass'];

            //Recipients
            $mail->setFrom('no.reply.seriesbuddies@gmail.com', 'SeriesBuddies'); //desde donde lo envía
            $mail->addAddress($correo); //Add a recipient

            //Content
            $mail->isHTML(true); //Set email format to HTML
            $mail->Subject = $asunto;
            $mail->Body = $cuerpo;
            // $mail->AltBody = 'This is the body in plain text for non-HTML mail clients';

            $mail->send();
            //echo 'Message has been sent';
        } catch (Exception $e) {
            //echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
        }
    }
}
```

Figura 68 - Clase Mailer

Su principal método es sendEmail, este método va a enviar un correo del destinatario al remitente, con los datos indicados.

Adicionalmente, cuenta con otros métodos que pintan el cuerpo del correo:

```
// MAIL VERIFICACIÓN
public static function pintaEmailVerificacion($rutaBase, $nb, $tkn) : String
{
    return "
<!DOCTYPE html>
<html>
<head>
    <title>Verifica tu cuenta y ñómete a nuestra comunidad</title>
</head>
<body style='font-family: Arial, sans-serif; background-color: #F0F0F0; color: #333333;'>
    <header style='max-width: 600px; margin: 0 auto; padding: 20px; background-color: #f0f0f0; height: 35px; background-color: #17181D; text-align: center;'>
        <img style='height:35px; user-select: none;' src='https://i.imgur.com/pt568sh.png' alt='SeriesBuddies'>
    </header>
    <div style='max-width: 600px; margin: 0 auto; padding: 20px; background-color: #f0f0f0;'>
        <h2 style='text-align: center;'>Verifica tu cuenta y convírtete en buddy!</h2>
        <p>¡Bienvenido/a a SeriesBuddies! Estamos encantados de tenerte a bordo y queremos asegurarnos de que tu experiencia sea excepcional desde el principio. Para ello, necesitamos verificar tu cuenta.<br>
        <p>Completa tu registro haciendo clic en el botón:<br>
            <a href='{$rutaBase}verify.php?token={$tkn}'>Verificar cuenta</a>
        </p>
        <p>Si el botón no funciona, también puedes copiar y pegar el siguiente enlace en la barra de direcciones de tu navegador:</p>
        <p><a href='{$rutaBase}verify.php?token={$tkn}'>{$rutaBase}verify.php?token={$tkn}</a></p>
        <p>Una vez que hayas verificado tu cuenta, tendrás acceso completo a todas las funciones y beneficios de nuestra plataforma. Podrás interactuar con otros usuarios y recibir actualizaciones exclusivas. ¡Gracias por unirte a nuestra comunidad! Esperamos verte pronto.</p>
        <p>Si no has creado una cuenta en nuestro sitio web, por favor ignora este correo electrónico. Es posible que alguien haya ingresado tu dirección de correo electrónico por error.</p>
        <br>
        <p>Saludos cordiales,</p>
        <p>Equipo de SeriesBuddies</p>
    </div>
    <footer style='max-width: 600px; margin: 0 auto; padding: 20px; background-color: #f0f0f0; height: 35px; background-color: #17181D; text-align: center;'>
        <img style='height:35px; user-select:none;' src='https://i.imgur.com/pt568sh.png' alt='SeriesBuddies'>
    </footer>
</body>
</html>
";
```

Figura 69 - Función pintaEmailVerificacion de Mailer

Estructura de las páginas, templates y programación de la lógica

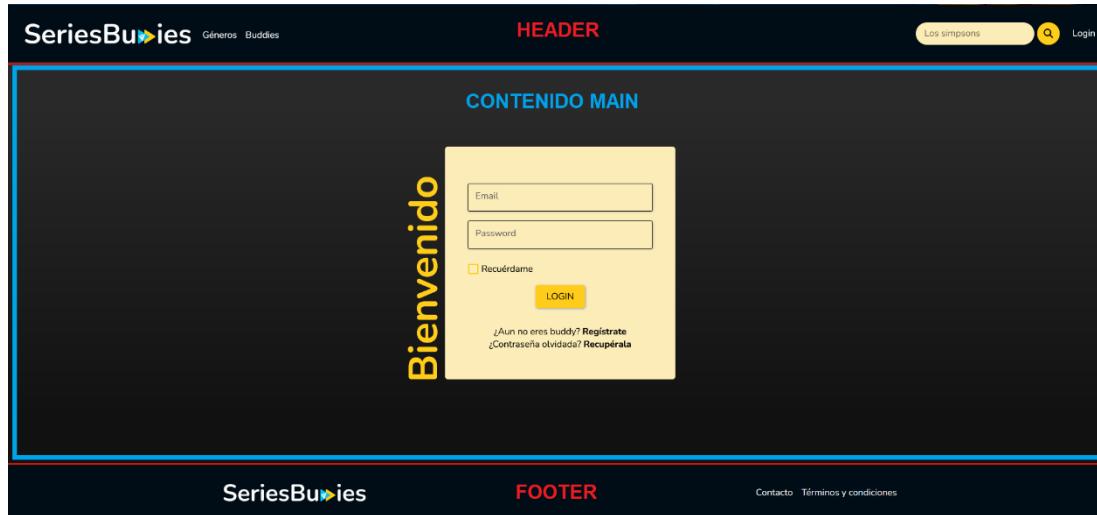


Figura 70 - Layout de las páginas

Para mantener la estructura de las páginas y la maquetación, con un código óptimo, se han utilizado templates, para evitar repetir código innecesariamente, llamando al template en cada página y pasándole el contenido central, así como también el título y otros parámetros (hojas CSS y JS específicas). A su vez el template hace una llamada al header y al footer, que se podrían considerar otros 2 templates o elementos “fijos”:

```

public > template.php
1   <!DOCTYPE html>
2   <html lang="es-ES">
3     <head>
4       <!-- META -->
5       <meta charset="UTF-8">
6       <meta http-equiv="X-UA-Compatible" content="IE=edge">
7       <meta name="viewport" content="width=device-width, initial-scale=1.0">
8       <meta name="description" content="Intercambia ideas sobre tus series favoritas">
9       <meta name="keywords" content="series, opiniones, ideas">
10      <meta name="author" content="Anabel, Román">
11      <meta name="copyright" content="Anabel, Román">
12      <!-- TITLE -->
13      <title><${titulohead}></title>
14      <link rel="icon" type="image/png" href="upload/logos/logo-mobile.png">
15      <!-- CSS -->
16      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
17      <link rel="stylesheet" href="/css/general.css">
18      <link rel="stylesheet" href=><${estiloSpecifico}>>">
19      <!-- JS -->
20      <script src=".js/header.js" defer></script>
21      <script src=><${scriptSpecifico}>><${scriptLoadMode}><></script>
22   </head>
23   <body>
24     <!-- global container -->
25     <div class="global-container">
26       <!-- header -->
27       <?php include_once('header.php'); ?>
28
29       <!-- body (central container) -->
30       <div class="container limit-width">
31         <!-- main -->
32         <main class="main" id="main">
33
34           <!-- contenido para el template -->
35           <?= ${content}>
36
37         </main>
38         <!-- bg fixed -->
39         <div class="bg-fixed"></div>
40
41       <!-- footer -->
42       <?php include_once('footer.php'); ?>
43     </div>
44     <div class="confirmation-wrapper d-none" id="confirmation-wrapper">
45       <div class="confirmation-card" id="confirmation-card">
46         <div class="confirmation-card__body" id="confirmation-body">
47
48         </div>
49         <div class="confirmation-card__footer">
50           <button class='btn btn--success' id='si'>Sí</button>
51           <button class='btn btn--error' id='no'>NO</button>
52         </div>
53       </div>
54     </div>
55   </body>
56 </html>

```

Figura 71 - Template

El procedimiento a seguir es: en las páginas se hace primero la lógica de la página:

```
<?php

    // ***** INIT *****
    require_once("../src/init.php");

    // ***** API TMDB *****
    use clases\api_tmdb\TMDB;

    $tmdb = new TMDB();

    $url = $tmdb->urlListadoGeneros();
    $response = $tmdb->peticionHTTP($url)['genres'];
```

Figura 72 - Lógica de las páginas

Y posteriormente se captura todo el contenido generado dado a la interacción con los modelos y/o la API y se manda al template, junto con otros datos adicionales (título, hojas css, etc.):

```
14     // ***** INFO PARA EL TEMPLATE *****
15     $tituloHead = "Géneros - Seriesbuddies";
16     $estiloEspecifico = "./css/genders.css";
17     $scriptEspecifico = "./js/genders.js";
18     $scriptLoadMode = "defer";
19     $content;
20
21     // ***** COMIENZO BUFFER *****
22     ob_start();
23 ?>
24     <h1 class="title title--1 text-align-center">GÉNEROS</h1>
25     <div class="main_content">
26
27         <?php
28             // *** generación de cajas ***
29             foreach ($response as $key => $value) {
30                 $url = $tmdb->urlSeriesGeneros($response[$key]['id']);
31             }
32             <!-- caja género -->
33             <div class="box">
34                 <a href=".//series.php?id=<?=$response[$key]['id']?>&pagina=1" class="box-body-wrapper">
35                     <div class="box_body box_body--gender loader">
36                         
37                     <div class="box-body__info">
38                         <h2 class="title title-gender"><?=$response[$key]['name']?></h2>
39                         <p class="text-white resultados" id="r-<?=$key?>" data-url="<?=$url?>">
40                             <i class='fa-solid fa-spinner rotate-infinite'></i> series &gt;
41                         </p>
42                     </div>
43                 </a>
44             </div>
45         <?php } ?>
46
47     </div>
48
49     </div>
50 <?php
51
52     // ***** FIN BUFFER + LLAMADA AL TEMPLATE *****
53     $content = ob_get_contents();
54     ob_end_clean();
55     require("template.php");
56
57 ?>
```

Figura 73 - Buffer y llamada al template

El buffer captura todo lo que se genera, lo que se almacena en la variable \$content, posteriormente se limpia el buffer y se hace una llamada al template, generando así páginas enteras sin necesidad de repetir código. Este método se realiza en cada una de las páginas.

DESPLIEGUE

Para el despliegue se va a utilizar un servidor VPS con Linux (Ubuntu 22.04 LTS) con Apache.

Para esto se necesitan dos cosas:

- Servidor VPS
- Dominio propio

La mecánica es tener el servidor escuchando todas las peticiones en un puerto determinado y con el dominio comprado apuntar hacia la IP de ese servidor. Así, cuando el servidor recibe una petición, lo redirige a la página index del proyecto (cosa que hay que configurar).

Antes de comenzar, se realizará la conexión mediante SSH con la IP, usuario y contraseña del servidor VPS:

- ssh usuario@IP

Primero se han de instalar todas las tecnologías que se van a utilizar (PHP, MariaDB, etc.):

- sudo apt update
- sudo apt upgrade
- sudo apt install curl
- sudo apt install apache2
- sudo apt install php
- sudo apt install php-curl
- sudo apt install mariadb-server
- sudo apt install php-mysql
- sudo apt install php-xml php-mbstring php-gd php-json php-zip
- service apache2 restart

También el sistema de gestión de paquetes para poder instalar ciertas dependencias de las que se hará uso, en este caso PHPMailer:

- ```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'55ce33d7678c5a611085589f1f3ddf8b3c52d662cd01d4ba75c0ee0459970c2200a51f
492d557530c71c15d8dba01eae') { echo 'Installer verified'; } else { echo 'Installer
corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

```
php composer-setup.php
```

```
php -r "unlink('composer-setup.php');
```

- sudo mv composer.phar /usr/local/bin/composer
- sudo reboot

Una vez con todas las tecnologías instaladas en el servidor, se procede a configurar el sitio web. Hacemos un git clone del proyecto en la carpeta Home. Posteriormente lo copiamos a /var/www:

- sudo cp -r SeriesBuddies /var/www

Se borra el .git (por seguridad) y se instalan todas las dependencias y se insertan todas las credenciales necesarias para el proyecto.

Posteriormente, se cambian los usuarios y grupos de todos los archivos de ese proyecto, a www-data, es el “usuario” de Apache que va a gestionar esos archivos:

- sudo chown -R www-data:www-data /var/www/SeriesBuddies/

Realizado lo anterior, se configura el nombre del servidor y a qué carpeta raíz se redirigirán todas las peticiones que lleguen desde ahí:

- cd /etc/apache2/sites-available

Editamos el archivo 000-default.conf:

- sudo nano 000-default.conf

```
GNU nano 6.2
<VirtualHost *:80>
 # The ServerName directive sets the request scheme, hostname and port that
 # the server uses to identify itself. This is used when creating
 # redirection URLs. In the context of virtual hosts, the ServerName
 # specifies what hostname must appear in the request's Host: header to
 # match this virtual host. For the default virtual host (this file) this
 # value is not decisive as it is used as a last resort host regardless.
 # However, you must set it for any further virtual host explicitly.
 ServerName www.seriesbuddies.es

 ServerAdmin webmaster@localhost
DocumentRoot /var/www/SeriesBuddies/public

 # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
 # error, crit, alert, emerg.
 # It is also possible to configure the loglevel for particular
 # modules, e.g.
 #LogLevel info ssl:warn

 ErrorLog ${APACHE_LOG_DIR}/error.log
 CustomLog ${APACHE_LOG_DIR}/access.log combined

 # For most configuration files from conf-available/, which are
 # enabled or disabled at a global level, it is possible to
 # include a line for only one particular virtual host. For example the
 # following line enables the CGI configuration for this host only
 # after it has been globally disabled with "a2disconf".
 #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figura 74 - Archivo 000-default.conf

Aquí conviene dejar el documento raíz directamente en public, para que el usuario no pueda escalar y acceder a otros archivos (archivos de config del src o clases). Hacemos exactamente lo mismo, pero con sites-enabled.

Hecho todo lo anterior, sacamos la IP de nuestro VPS y apuntamos con el dominio a esa IP (tardará en propagarse mínimo 5-15min, hasta un máximo de 24h):

| Domínio                             | TTL | Tipo  | Destino    |
|-------------------------------------|-----|-------|------------|
| seriesbuddies.es.                   | 0   | NS    | [REDACTED] |
| seriesbuddies.es.                   | 0   | NS    | [REDACTED] |
| seriesbuddies.es.                   | 0   | MX    | [REDACTED] |
| seriesbuddies.es.                   | 0   | MX    | [REDACTED] |
| seriesbuddies.es.                   | 0   | A     | IP         |
| www.seriesbuddies.es.               | 0   | A     | IP         |
| seriesbuddies.es.                   | 0   | TXT   | [REDACTED] |
| autoconfig.seriesbuddies.es.        | 0   | CNAME | [REDACTED] |
| autodiscover.seriesbuddies.es.      | 0   | CNAME | [REDACTED] |
| _autodiscover_tcp.seriesbuddies.es. | 0   | SRV   | [REDACTED] |

Figura 75 - Dominio

Un último retoque, es añadir un certificado SSL para poder tener HTTPS en el sitio web. Esto se hará con certbot siguiendo las instrucciones de <https://certbot.eff.org/>.

### 3.1.4 Pruebas

Son muchas pruebas que pueden realizarse en un proyecto, para eliminar los posibles errores y garantizar su correcto funcionamiento. Los casos de prueba establecen las condiciones/variables que permitirán determinar si los requisitos establecidos se cumplen o no. A continuación, se detallan algunos de los casos de prueba que se ejecutarán para comprobar la correcta construcción de este proyecto.

#### 28-05-2023 – Román Konyeyev

**Prueba:** correo de confirmación de registro

**Descripción:** comprobar que el correo se mande y visualice correctamente y que el enlace funcione.

**Pasos previos:** tener una cuenta de correo que no esté registrada en la página. Realizar un registro estándar mediante la página web ya desplegada.

**Entrada:** datos del usuario.

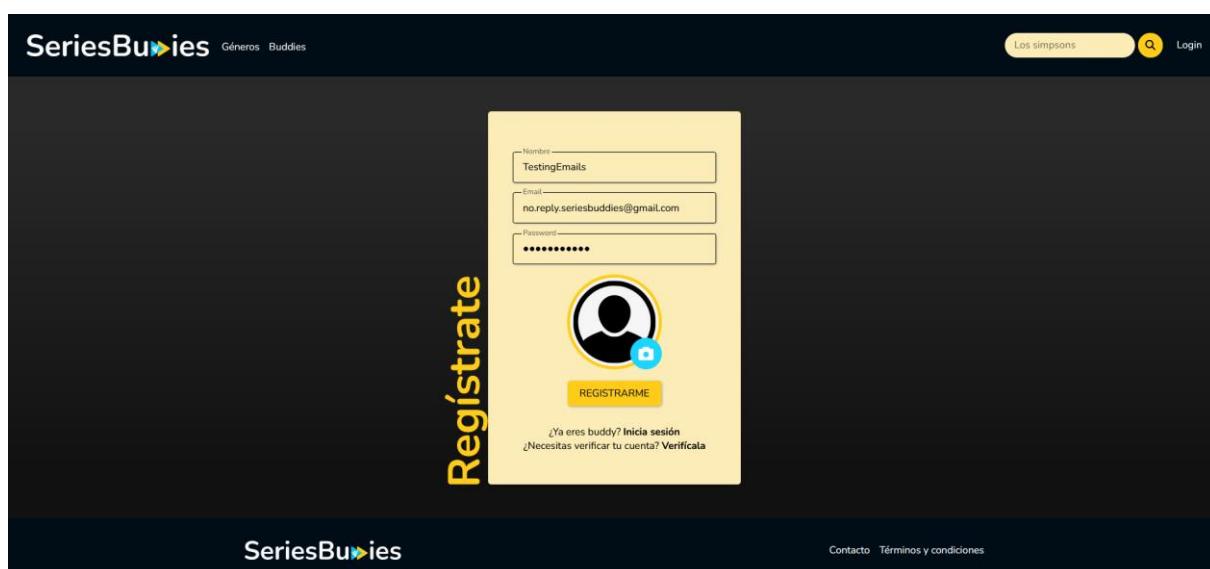


Figura 76 - Datos de registro

**Resultado esperado:** se espera que el correo si todos los datos introducidos son válidos, se envíe el correo de confirmación, se visualice correctamente y el enlace funcione.

**Resultado obtenido:** el resultado obtenido no es del todo el esperado. Parece que está todo correcto, el correo se envía, se visualiza correctamente, pero el enlace no redirige a la página adecuada.

**Evaluación:** existe un error de código, un “string mágico” que está provocando una redirección incorrecta.

```
//mandamos mail de confirmación
Mailer::sendEmail(
 $email->getValor(),
 "Completa tu registro - SeriesBuddies",
 Mailer::pintaEmailVerificador("http://localhost:8000/public/", $nuevoUsuario['nombre'], $token)
);
```

Figura 77 - Ruta enlace correo

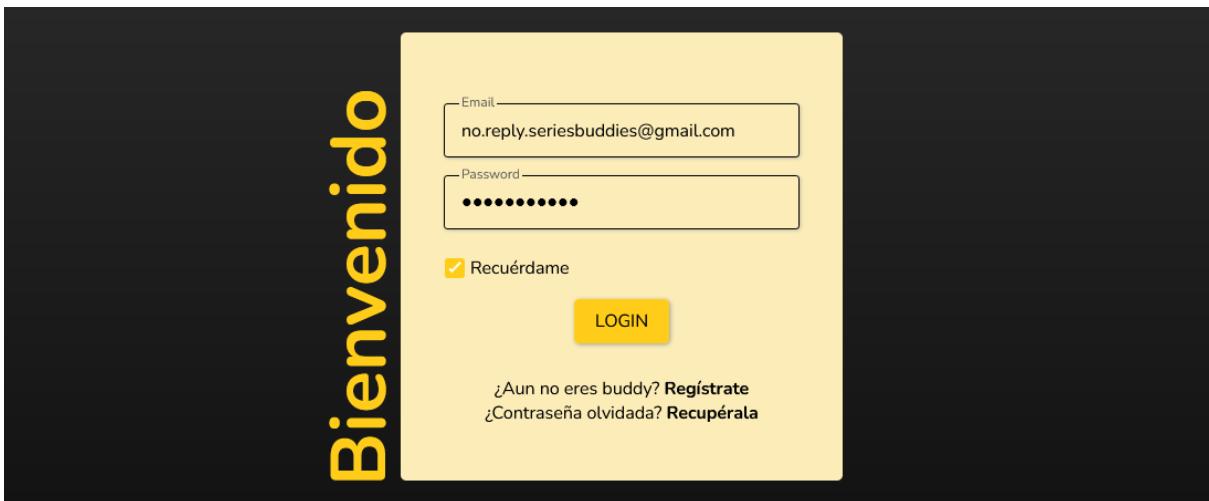
**29-05-2023 - Román Konyeyev****Prueba:** cookies**Descripción:** comprobar que el sitio web almacena las cookies de manera correcta**Pasos previos:** tener un usuario registrado en la base de datos, realizar el login con los datos correctos del usuario.**Entrada:** datos del usuario registrado.

Figura 78 - Datos de login

**Resultado esperado:** se espera que se establezca la cookie “recuerdame”.**Resultado obtenido:** se establece la cookie “recuerdame” correctamente.

| Almacenamiento |  |                                                                                                                               |                        |
|----------------|--|-------------------------------------------------------------------------------------------------------------------------------|------------------------|
| Nombre         |  | Valor                                                                                                                         | Domain                 |
| PHPSESSID      |  | 1biq175et89bi5cp0ppsqlf64g                                                                                                    | www.seriesbuddies.es / |
| recuerdame     |  | 3206f9a2735af609530c822e8f52f5191e5e2508853d976124cf932cdfde30f6ad14985ba792a1f25991d449a3290453a06cc5bd0396bb1089dbf615cc8cf | www.seriesbuddies.es / |

Figura 79 - Cookies

**Evaluación:** la prueba ha resultado satisfactoria.

## 31-05-2023 – Ana Isabel Pedrajas Navarro

**Prueba:** buscador de series con la API tmdb.

**Descripción:** comprobar que se listan todas las series cuyo nombre original, traducido o “nombre conocido” coincida con la búsqueda especificada sin tener en cuenta las mayúsculas o las minúsculas.

**Pasos previos:** para hacer uso del buscador no es necesario ser un usuario registrado. En la función de la clase TMDB se debe habilitar una función que sea no case-sensitive.

```
//Devuelve la info de las series cuyo nombre coincide
public function getSeriesNombre ($nombre, $currentPage=1) {
 $urlSerie = $this->urlSerieNombre($nombre, $currentPage);
 $resultados=$this->peticionHTTP($urlSerie);

 $serieData = [];
 $series=$resultados['results'];

 $totalResults = $resultados['total_results'];

 //Guardamos el total de paginas, si sobrepasan las 500 permitidas almacenamos ese maximo
 $totalPages = ($resultados['total_pages'] <= self::$MAX_PAGINAS)? $resultados['total_pages']: $totalPages = self::$MAX_PAGINAS;

 foreach ($series as $key => $serie) {
 foreach ($serie as $clave => $contenido) {
 if ($clave=='name' && str_contains($contenido, strtolower($nombre)) == 0) {
 $serieData[] = $this->almacenarDatosSeries($series, $key);
 }
 }
 }

 //serieData = $this->mapearGenerosID($serieData);
 array_push($serieData, $totalResults);
 array_push($serieData, $totalPages);

 return $serieData;
}
```

Figura 80 - Función getSeriesNombre de TMDB

**Entrada:** una o más palabras que sirvan como parámetro de búsqueda.

**Resultado esperado:** se espera que aparezcan todas las series que contengan en su nombre la palabra “walking”.

**Resultado obtenido:** se devuelven todas las series con la palabra “walking” en su nombre sin tener en cuenta las mayúsculas o minúsculas.

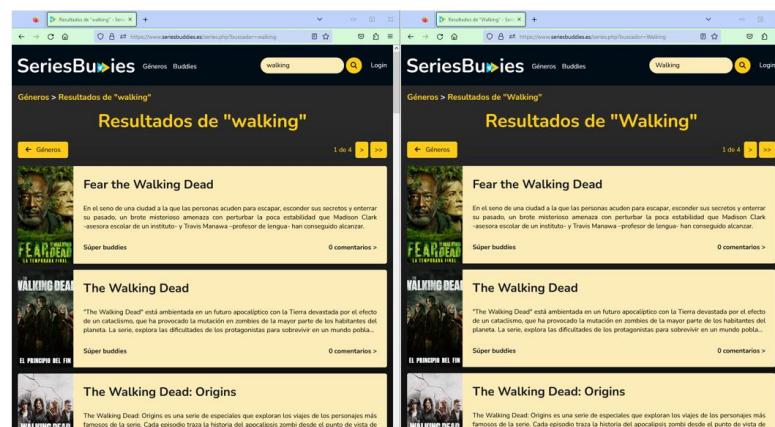


Figura 81 - Resultados por búsqueda de series

**Evaluación:** la prueba ha resultado satisfactoria.

### 25-05-2023 – Ana Isabel Pedrajas Navarro

**Prueba:** comprobar el funcionamiento de los triggers sumar\_medallas y restar\_medallas.

**Descripción:** comprobar que los triggers sumar\_medallas y restar\_medallas se ejecutan correctamente tras alterar la tabla respuestas.

**Pasos previos:** tener la sesión iniciada en el sitio web.

**Entrada:** insertar y eliminar un comentario.

**Resultado esperado:** se espera que al dejar el comentario número 5, se consiga el chip de bronce y que al eliminarlo, se elimine de forma automática el mismo chip.

**Resultado obtenido:** tras dejar el comentario número 5, se consigue el chip de bronce. Al eliminar un comentario, se elimina el chip de bronce.

*Se escribe el comentario número 5 y se obtiene el chip de bronce (trigger sumar\_medallas)*

The screenshot shows a browser window with the URL [https://www.seriesbuddies.es/feed.php?id=62286&id\\_genero=0](https://www.seriesbuddies.es/feed.php?id=62286&id_genero=0). The page is for the TV show 'Fear the Walking Dead'. It features a large banner image of the show. Below the banner, a comment from 'Rowoon' is displayed, which has been liked 5 times. At the bottom of the page, there's a summary of 'Rowoon's stats and a section titled 'MIS CHIPS' with icons for a gift and a popcorn bucket.

Figura 82 - Proceso de obtención del chip bronce

Se elimina un comentario y se pierde el chip de bronce (trigger restar\_medallas)

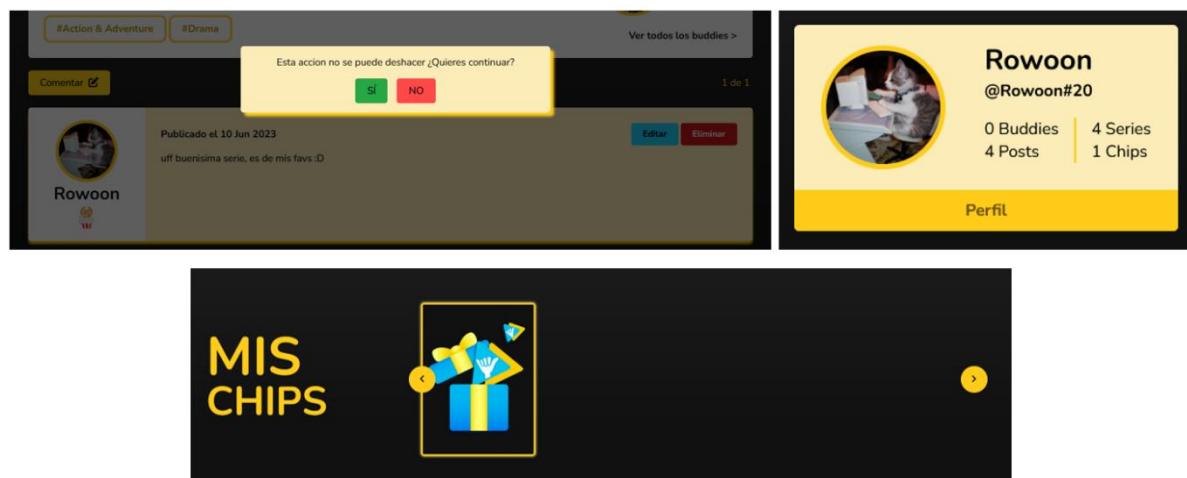


Figura 83 - Proceso de eliminación de chip de bronce

**Evaluación:** la prueba ha resultado satisfactoria.

### **3.2 *Objetivos que conseguir***

Establecemos los siguientes objetivos “Smart” con los que poder definir una estrategia de crecimiento de la plataforma web:

2023 - 2025

- Establecer afiliaciones y colaboraciones con empresas del sector cinematográfico, audiovisual y de streaming para poder aplicar sus servicios en la plataforma.
- Crear una base estable de usuarios en Madrid situados en la franja de edad de los 18 y 25 años (1.000 usuarios potenciales).
- Generar sentimiento de comunidad asociado a la marca SeriesBuddies.
- Lanzar aplicaciones móviles para Android e iOS.

2025 - 2027

- Alcanzar una rentabilidad.
- Expansión a otras comunidades autónomas del país.
- Incrementar la base de usuarios proporcionalmente al número de comunidades autónomas al que se consiga hacer llegar la plataforma.

2027 - 2030

- Expandir SeriesBuddies horizontalmente hacia la venta de merchandising propio y colaboración con plataformas de streaming para publicar contenido exclusivo (productos físicos y digitales).

Con respecto a los objetivos como el incremento del número de usuarios que utilicen SeriesBuddies, abordar nuevas líneas de negocio o mejorar la competitividad, se desarrollan las siguientes propuestas:

#### **Propuestas a corto plazo (máximo 6 meses):**

Para aumentar la interactividad de los usuarios al hacer uso de SeriesBuddies:

- Diseñar las creatividades para nuevos chips y sus correspondientes procedimientos y/o triggers. Por ejemplo:
  - Chips a los usuarios con que alcancen un total de 5 buddies, entre 10 y 19 buddies y más de 20 buddies.
  - Chips a los usuarios que alcancen más de 10 series vistas del mismo género.
  - Chips a los usuarios que comenten más de 10 veces en un mismo día.

- Chips por veteranía en función del tiempo que lleven registrados en el sitio web.
- Habilitar la posibilidad de enviar y recibir mensajes privados a otros buddies con el fin de establecer una comunicación más directa y personal.
- Responder a otros comentarios y poder darle me gusta.
- Aplicar botones para marcar las series que se están viendo actualmente y una sección de “favoritos”.
- Permitir que el usuario pueda dejar una valoración numérica de una serie para poder calcular y elaborar un ranking de las series mejor valoradas.
- Complementar la información de una serie con su respectivo tráiler de tal modo que el usuario pueda hacerse una idea de si esa serie se encuentra acorde a sus preferencias o no.

#### **Propuestas a medio y largo plazo (entre 1 y 3 años):**

Para abordar nuevas líneas de negocio con SeriesBuddies:

- Añadir publicidad en la página de forma poco intrusiva (configurar banners a modo de “asides” o cabeceras). Aplicar una membresía premium para aquellos usuarios que deseen deshabilitar la publicidad.
- Agregar botones que servirán como hipervínculos que indiquen dónde puede ver el usuario dicha serie y establecer a su vez links de afiliaciones para esas plataformas de streaming.
- Configurar una sección o apartado en donde el usuario pueda comprar merchandising de las series e integrar el mismo sistema de afiliación que el comentado en el punto anterior.
- Diseño y elaboración de propio merchandising relacionado con la marca SeriesBuddies para favorecer la expansión de la plataforma y en consecuencia de la empresa.
- Establecer acuerdos con otros portales para distribuir productos digitales y físicos exclusivos a la comunidad de SeriesBuddies.
- Invitar a otras plataformas a la creación de una API colaborativa y común en donde poder tener un registro actualizado de las series disponibles y de los próximos estrenos.

Para mejorar la competitividad con respecto a otras plataformas o sitios web que puedan surgir o ya existan en el mercado:

- Desarrollar una aplicación móvil que esté disponible en todos los dispositivos, facilitando y automatizando el acceso del usuario a la aplicación.
- Poder ofrecer varias opciones de registro (Google, Apple, Outlook, ...) para aumentar la comodidad del usuario al usar la plataforma e incrementar el número de posibles clientes.
- Implementar un calendario o sistema de recordatorio de eventos que avisen al usuario de cuándo está disponible el próximo episodio de la serie que está viendo o de la fecha de estreno de una serie.
- Habilitar la aplicación móvil y el sitio web en diferentes idiomas para que la sección de mercado no hispanohablante pueda hacer uso de ella.

### **3.3 Previsión de los recursos materiales y humanos necesarios**

Se tendrá en cuenta las herramientas y la formación necesaria para desarrollar las actividades que requiere el proyecto, así como el tiempo para llevarlo a cabo.

### **3.4 Presupuesto económico.**

Antes de poner en marcha el desarrollo de SeriesBuddies, se realizó un listado de los materiales necesarios y posibles gastos a tener en cuenta y que se detallan a continuación:

|                              |                                                                                                                                                                              |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ordenadores                  | Equipos informáticos con los que se desarrollará el proyecto.                                                                                                                |
| API                          | Fuente de información externa que permite enriquecer nuestro sitio web y mantenerlo actualizado sin necesidad de realizar cambios.                                           |
| VPS (Virtual Private Server) | Servidor remoto dónde se alojará el proyecto.                                                                                                                                |
| Dominio                      | Zona DNS que se utilizará para apuntar a la IP del servidor. Esto permite disponer de un nombre para la web propio, totalmente personalizado, más corto y fácil de recordar. |

En cuanto a los recursos humanos, al tratarse SeriesBuddies de una empresa de sociedad limitada y contar en un primer momento con solo dos socios como personal de la empresa, no se necesitan a más programadores o diseñadores dado el nivel de trabajo que requiere la aplicación. No obstante, en el futuro se puede valorar la ampliación de la plantilla si el crecimiento de SeriesBuddies es el esperado.

Finalmente, tras finalizar las fases de desarrollo se realizó la compra del VPS y del dominio tal y como se había previsto, para poder desplegar el sitio web. A continuación, se adjunta una captura de pantalla del desglose de gastos.

| Mis facturas                                                                                        |                          |                  |                  |                 |                   |        | Guías   |
|-----------------------------------------------------------------------------------------------------|--------------------------|------------------|------------------|-----------------|-------------------|--------|---------|
| Consulte y descargue sus facturas. Consulte el estado de sus pagos. Consulte su crédito disponible. |                          |                  |                  |                 |                   |        |         |
| Facturas                                                                                            | Seguimiento de los pagos | Mi crédito       |                  |                 |                   |        |         |
| Acciones masivas                                                                                    |                          |                  |                  |                 |                   |        | Filtrar |
| □                                                                                                   | Referencia               | Número de pedido | Fecha de emisión | Importe sin IVA | Importe IVA incl. | Saldo  | Estado  |
| □                                                                                                   | E5 [REDACTED]            | [REDACTED]       | 1 jun. 2023      | 7.08 €          | 8.57 €            | 0.00 € | Pagada  |
| □                                                                                                   | E5 [REDACTED]            | [REDACTED]       | 31 may. 2023     | 5.80 €          | 7.02 €            | 0.00 € | Pagada  |

Figura 84 - Coste del dominio y servidor VPS

## 4 PLANIFICACIÓN DE LA EJECUCIÓN DEL PROYECTO

A continuación, se detallan las actividades/tareas/procedimientos por cada una de las fases del proyecto previamente definidas.

### 4.1 Fase de Análisis

1. Estudio de la situación actual
2. Estudio de las necesidades a cubrir
3. Establecimiento de los requisitos del proyecto
4. Valoración comparativa de las posibles soluciones
5. Identificación de las necesidades que implica el nuevo proyecto en la empresa
6. Estudio de viabilidad de la solución elegida teniendo en cuenta no solo los beneficios económicos
7. Corrección de posibles errores

### FASE DE ANÁLISIS

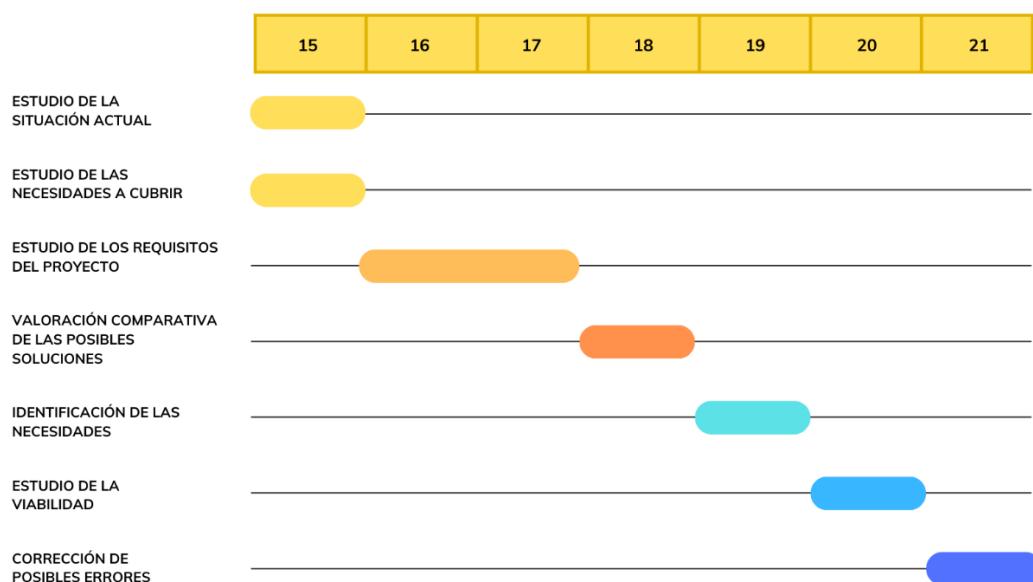


Figura 85 - Diagrama de Gantt análisis

## 4.2 Fase de diseño

1. Preparación del entorno de diseño
2. Diseño de la arquitectura
3. Diseño de los interfaces
4. Diseño de los datos
5. Diseño de los procedimientos
6. Corrección de posibles errores

### FASE DE DISEÑO

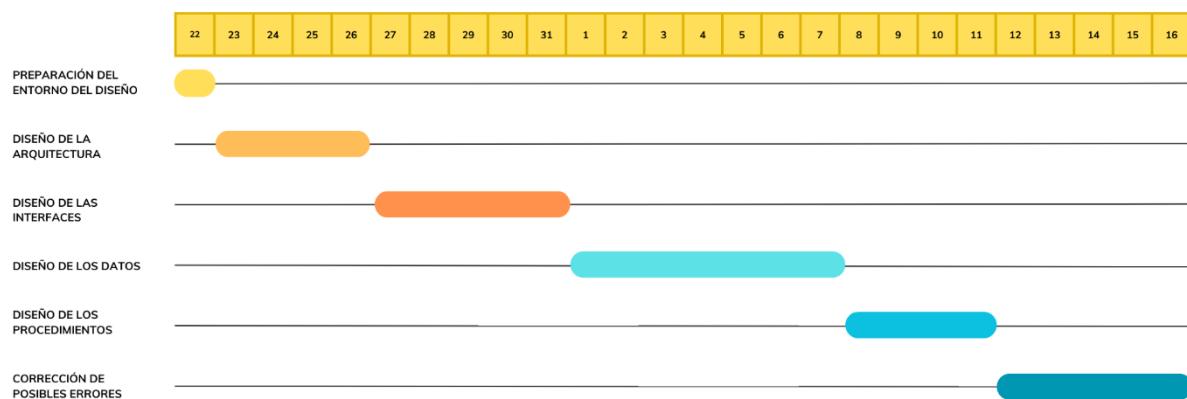


Figura 86 - Diagrama de Gantt diseño

## 4.3 Fase de Implementación

1. Preparación del entorno de implementación
2. Desarrollo de la arquitectura
3. Desarrollo de los interfaces, los datos y los procedimientos
4. Corrección de posibles errores: se debe destacar el hecho de que durante el proceso de desarrollo e integración de las distintas funcionalidades que posee cada página del proyecto, se han realizado al mismo tiempo pequeñas pruebas para controlar y subsanar errores. Por tanto, el tiempo de esta fase es relativo.

## FASE DE IMPLEMENTACIÓN

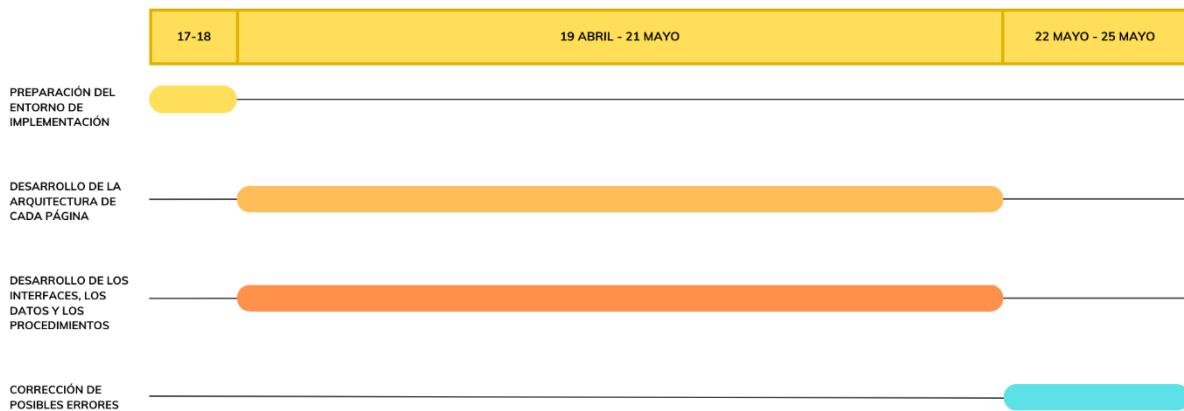


Figura 87 - Diagrama de Gantt implementación

## 4.4 Fase de pruebas

1. Preparación del entorno de pruebas
2. Ejecución de las pruebas y reporte de los errores encontrados

## FASE DE PRUEBAS



Figura 88 - Diagrama de Gantt pruebas

## 5 DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN

Tras haber dado por finalizado el desarrollo y despliegue de SeriesBuddies y aun habiendo realizado las pruebas necesarias durante el mismo proceso, se plantea realizar pruebas periódicas para controlar que todas las funciones del sitio web siga funcionando de manera correcta.

Por otro lado, se mantiene y actualiza una copia de seguridad del proyecto completo. Esta copia contiene la estructura del proyecto, los scripts de la base de datos (tablas, datos necesarios, eventos y triggers), clases DWES y API, templates y copias de todas las páginas que componen el sitio web.

## 6 FUENTES

- API *The Moovie Database*. (2023, marzo). The Moovie Database. <https://developer.themoviedb.org/docs>
- Cuesta, V. E. (2022, 29 mayo). ¿No tienes con quién comentar las series que miras? Esta es tu aplicación. *La Vanguardia*. <https://www.lavanguardia.com/series/20220529/8301509/tienes-comentar-series-miras-aplicacion-tv-time-pmv.html>
- divcode. (2022, 14 noviembre). *como HACER un CARRUSEL en HTML y CSS y JAVASCRIPT*  [Vídeo]. YouTube. <https://www.youtube.com/watch?v=wK86LvGxnJc>
- Elpuente. (2021). Innovación tecnológica en la Prevención de Riesgos Laborales. *Metacontratas*. <https://www.metacontratas.com/blog/innovacion-tecnologica-en-la-prevencion-de-riesgos-laborales/>
- Fernández, Y. (2022). Los 14 mejores servicios y apps para seguir y controlar las series y películas que ves y tener toda su. . . *Xataka*. <https://www.xataka.com/basics/14-mejores-servicios-apps-para-seguir-controlar-series-peliculas-que-ves-tener-toda-su-informacion>
- GitHub.com Documentación de la Ayuda*. (s. f.). GitHub Docs. <https://docs.github.com/es>
- González, C. (2023, 16 febrero). En 2022, los niños pasaron cuatro horas al día delante de una pantalla. *El Debate*. [https://www.eldebate.com/familia/20230216/2022-ninos-pasan-cuatro-horas-dia-delante-pantalla\\_93891.html](https://www.eldebate.com/familia/20230216/2022-ninos-pasan-cuatro-horas-dia-delante-pantalla_93891.html)
- Infoautonomos. (2022a, enero 21). *Obligaciones de una Sociedad Limitada (SL)*. <https://www.infoautonomos.com/tipos-de-sociedades/obligaciones-sociedad-limitada-sl/>
- Infoautonomos. (2022b, diciembre 1). *Sociedad Limitada: definición, características y ventajas*. <https://www.infoautonomos.com/tipos-de-sociedades/sociedad-limitada-caracteristicas-ventajas/>
- Moody, R., & Moody, R. (2023). Estadísticas sobre el tiempo de pantalla: tiempo de pantalla medio en Estados Unidos y el resto del mundo. *Comparitech*. <https://www.comparitech.com/es/transmisiones-de-video/estadisticas-tiempo-pantalla/>

*PHP: Language Reference - Manual.* (s. f.). <https://www.php.net/manual/en/langref.php>

*PHP: MySQL - Manual.* (s. f.). <https://www.php.net/manual/en/set.mysqlinfo.php>

*Principios del diseño UX/UI.* (s. f.). The Bridge. Recuperado 7 de junio de 2023, de <https://www.thebridge.tech/blog/principios-basicos-del-ui-design>

Quiñoy, L. (2023). Estas son todas las ayudas para jóvenes emprendedores. *APD España*.  
<https://www.apd.es/ayudas-para-jovenes-emprendedores/>

Roces, P. R., & Roces, P. R. (2021, 11 marzo). El Covid dispara el consumo de televisión en España: cuatro horas al día y record en marzo y abril. *ELMUNDO*. <https://www.el-mundo.es/television/2021/03/11/6049f1f3fddfffc93e8b4642.html>

Ruiz, A. (2023, 21 marzo). *Así consumimos televisión vía streaming: estas fueron las películas y series más populares en plataformas online del 2022 - Marketing 4 Ecommerce - Tu revista de marketing online para e-commerce*. Marketing 4 Ecommerce - Tu revista de marketing online para e-commerce. Recuperado 5 de junio de 2023, de <https://marketing4ecommerce.net/asi-consumimos-television-via-streaming-peliculas-y-series-mas-populares-en-plataformas-online/>

*Trigger Overview.* (s. f.). MariaDB KnowledgeBase. <https://mariadb.com/kb/en/trigger-overview/>