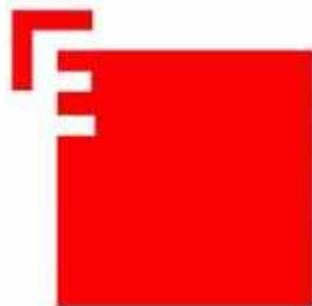


# **COMPODEV**



**escuelaartegranada**

**Alejandro Aguayo Chinchilla**

**2º DAW**

**Escuela Arte Granada**

## **1. Descripción general del proyecto**

- a.** Detalle del problema planteado indicando punto por punto todos los detalles a gestionar.
- b.** Indicar las áreas generales en las que se va a centrar el proyecto

## **2. Perfiles de usuario**

- a.** Descripción de todos los perfiles de usuario que habrá en la aplicación

## **3. Diagrama Entidad/Relación**

- a.** Diseñar el diagrama Entidad/Relación
- b.** Explicar origen de los atributos
- c.** Explicar por qué se han elegido esas claves primarias

## **4. Paso a tablas**

- a.** Realizar el paso a tablas del diagrama Entidad/Relación

## **5. Implementación de la base de datos**

- a.** Código SQL que genera las tablas

## **6. Diseño de la web**

- a.** Estudio de la competencia
- b.** Diagramación: mapa web y menús de navegación
- c.** Prototipos de bajo nivel: Wireframes
- d.** Referentes estéticos y colores
- e.** Tipografía
- f.** Imágenes: Logotipo, iconos, botones y formularios
- g.** Diseño gráfico de la interfaz

## **7. Programación de la web**

- a.** Funcionalidades programadas con PHP
- b.** Funcionalidades programadas con AJAX
- c.** APIs utilizadas

## **8. Manuales de usuario**

# 1. DESCRIPCIÓN DEL PROYECTO

El Proyecto a tratar se basa en una aplicación web, cuyo cometido es centralizar y facilitar el trabajo del programador dedicado tanto al frontend, como al backend. En un mundo sumamente complejo en el que el trabajo técnico no es valorado en base a su complejidad, aún se dificulta un extra el desempeño del desarrollador web cuando, además de solicitar la implementación de una aplicación web, se le pide que sea capaz de hacer los diseños para la misma. De esta forma, se desperdicia mucho más tiempo en el desarrollo creativo de una idea que realizando la misma, ya que un desarrollador no tiene por qué saber de diseño, lo que puede introducir errores que tendrán que solucionarse. Sumado a esto, está el hecho de que los plazos de tiempo requeridos se ajustan a solo una de las dos partes.

En ocasiones se utilizan componentes o código de proyectos antiguos implementando pequeñas modificaciones adaptadas a las necesidades del proyecto actual, lo que termina siendo un proceso tedioso. Entrar en GitHub o en el disco duro donde se tiene almacenado el proyecto donde utilizaste ese componente, luego una vez lo encuentra, si es que consigue encontrarlo, tiene que revisar las cientos de líneas de código probablemente spaghetti y sin comentar, para poder localizar ese pequeño fragmento de código.

Esto será cosa del pasado, la desorbitada cantidad de tiempo invertido en búsquedas de un único componente y frustraciones por no tener a mano eso que tanto ansía justo cuando más lo necesitas, se acabará para siempre, a partir de ahora será sustituido por un enorme catálogo donde podrás encontrar lo que requiere además de futuras ideas.

La aplicación funcionará de la siguiente manera, se podrá entrar como usuario invitado de forma que, sin necesidad de identificarse, obtendrá acceso limitado a muchos componentes o fragmentos de código completamente gratuitos. En caso de querer el acceso completo a todo el contenido, deberá identificarse, al igual de obtendrá acceso a participar en muchas otras actividades contenidas en la web para poder ayudar a mantener una comunidad activa, como lo será la sección de desafíos, donde será propuesto un desafío semanal o en su defecto, mensual, en función de la complejidad del reto. Los usuarios registrados competirán entre ellos de manera completamente gratis y libre de toxicidad para poder finalizar con el mejor resultado posible en base a la temática presentada. Sumado a todo lo anterior, el usuario registrado puede optar voluntariamente a la posibilidad de mejorar su plan de usuario, pudiendo acceder a pagos mensuales donde tendrá acceso a un plan PRO, donde accede al foro de la web, lugar

donde si desea puede iniciar debates y otro usuario PRO con más experiencia y conocimientos dará respuesta a su duda. Lo importante de hacerlo con un plan de suscripción es asegurar un foro que no acabe saturado de preguntas absurdas que no tengan ninguna relación con el tema principal de la web, al ser de pago el acceso será más exclusivo y en consecuencia, será más fácil de moderar.

## ÁREAS DE LA WEB

**El área de creación de componentes** constituye uno de los pilares fundamentales de la aplicación web, específicamente concebida para simplificar, optimizar y acelerar el trabajo de los desarrolladores web, independientemente de si su especialización es en frontend o backend. La propuesta se centra en la generación, gestión y reutilización eficiente de componentes modulares que integran las tecnologías más relevantes del entorno web, tales como HTML, CSS y JavaScript.

La creación de componentes tiene como objetivo principal dotar a los usuarios de una herramienta integrada que permita desarrollar bloques de código independientes y reutilizables, con características esenciales como la claridad, modularidad, escalabilidad y facilidad de integración. Estos componentes, concebidos desde el principio bajo las mejores prácticas y estándares técnicos actuales, contribuirán de manera determinante a evitar la repetición innecesaria de código, reduciendo así significativamente el tiempo empleado en tareas recurrentes y tediosas.

La plataforma proporcionará un editor especializado, intuitivo y robusto, donde el desarrollador podrá trabajar cómodamente, contar con soporte visual inmediato y la posibilidad de depurar y optimizar el código desde un mismo entorno integrado. Este editor no solo facilitará la codificación en tiempo real, sino que además contará con funciones avanzadas como autocompletado, validación instantánea, resaltado de sintaxis, integración directa con sistemas de control de versiones (por ejemplo, Git) y soporte para múltiples frameworks y bibliotecas populares (React, Angular, Vue.js, entre otros).

Asimismo, la plataforma permitirá a los usuarios gestionar adecuadamente su biblioteca personal de componentes, mediante herramientas que facilitan la documentación interna, etiquetado para búsqueda rápida y control de versiones claras y transparentes. De esta forma, la recuperación posterior de componentes específicos o variantes de los mismos será un

proceso ágil y sencillo, evitando de manera efectiva la confusión generada por múltiples versiones almacenadas en diferentes proyectos o repositorios externos.

Finalmente, la plataforma promoverá una cultura de compartición y mejora continua del código, ofreciendo facilidades para publicar componentes desarrollados que, previa validación técnica, podrán ser puestos a disposición del resto de la comunidad. Con ello, se establece un círculo virtuoso de retroalimentación técnica, favoreciendo la calidad del desarrollo general y generando un amplio repositorio de componentes validados y eficientes, contribuyendo decisivamente a elevar el estándar de calidad en el desarrollo web.

**La publicación de proyectos** constituye una dimensión esencial dentro del contexto de la aplicación web, al cumplir con la doble función de fomentar la visibilidad profesional del desarrollador y contribuir a la consolidación activa de una comunidad técnica de alto valor añadido.

Esta área se plantea con el objetivo fundamental de proporcionar a los desarrolladores un canal eficiente y organizado para dar a conocer sus creaciones. Dicho espacio servirá tanto para mostrar sus habilidades técnicas y creativas como para destacar ante posibles colaboradores, empleadores o clientes potenciales. En este sentido, la plataforma actuará simultáneamente como escaparate digital y red profesional, permitiendo una visualización clara, ordenada y atractiva de cada proyecto publicado, con la posibilidad adicional de incluir detalles técnicos, objetivos, tecnologías utilizadas, y funcionalidades específicas de cada trabajo.

La estructura organizativa del área de publicación facilitará considerablemente la creación y gestión del portafolio personal o profesional del usuario registrado. Los desarrolladores dispondrán de herramientas sencillas e intuitivas para añadir nuevos proyectos, describirlos adecuadamente mediante una ficha técnica completa, adjuntar fragmentos específicos de código, imágenes o vídeos explicativos, e incluso vincular el proyecto directamente a repositorios externos, tales como GitHub, GitLab o Bitbucket, promoviendo la transparencia, la colaboración y la verificación técnica inmediata por parte de otros miembros de la comunidad.

Además de potenciar el aspecto profesional individual, la publicación de proyectos favorecerá activamente la interacción constructiva entre usuarios. Mediante funcionalidades tales como comentarios, valoraciones o recomendaciones, la plataforma promoverá el intercambio de conocimientos y la generación de retroalimentación técnica entre pares. Este entorno

colaborativo incentivará la mejora constante y permitirá a los desarrolladores obtener opiniones calificadas que les servirán para perfeccionar futuros trabajos, adquirir nuevas perspectivas técnicas y detectar áreas de mejora continua en sus procesos de desarrollo.

Otra característica clave de esta área será la integración con el sistema de desafíos propuesto por la plataforma. Los usuarios podrán publicar proyectos específicos que den respuesta a los desafíos semanales o mensuales planteados por la comunidad, lo que incrementará su visibilidad dentro del ecosistema de usuarios, fomentará la sana competencia técnica, e impulsará a los desarrolladores a mantener un proceso constante de aprendizaje y mejora de sus habilidades.

Finalmente, el resultado global perseguido mediante la publicación organizada de proyectos es consolidar un espacio virtual profesional que represente un referente reconocido para empresas, reclutadores y la comunidad técnica internacional. Con ello, se busca favorecer una comunidad dinámica, respetuosa y altamente especializada, en la que el desarrollador pueda ver reflejada su evolución profesional de manera tangible, práctica y accesible.

**El área de descubrimiento** constituye un componente esencial y estratégico dentro de la aplicación web, al ofrecer a los desarrolladores una amplia biblioteca organizada y categorizada, destinada a optimizar la búsqueda y reutilización eficiente de componentes y fragmentos de código previamente desarrollados por usuarios alrededor del mundo.

Este espacio tiene como principal objetivo facilitar significativamente el proceso de búsqueda y adquisición de elementos específicos necesarios para distintos proyectos web, reduciendo al mínimo la pérdida de tiempo generada por búsquedas desorganizadas y fragmentadas en múltiples fuentes o plataformas externas. Así, mediante una cuidada organización, clasificación y etiquetado de los componentes disponibles, los desarrolladores podrán localizar rápidamente exactamente lo que necesitan.

La plataforma implementará un sistema avanzado de búsqueda basado en filtros inteligentes, categorización temática, tecnologías utilizadas, compatibilidad con frameworks específicos, niveles de dificultad técnica, y valoración de la comunidad. Esta herramienta permitirá al usuario realizar búsquedas precisas y efectivas, tanto si conoce con exactitud el componente que necesita, como si está explorando nuevas alternativas o soluciones técnicas para inspirarse y complementar sus proyectos actuales.

Además, el área de descubrimiento incluirá recomendaciones automáticas personalizadas, basadas en patrones de uso, proyectos previos realizados por el usuario, e intereses específicos registrados por cada desarrollador. Estas recomendaciones estarán orientadas a impulsar un flujo constante de aprendizaje técnico y creatividad, promoviendo la adopción de nuevas soluciones o enfoques innovadores utilizados por la comunidad global de desarrolladores.

Como mecanismo de control y aseguramiento de calidad, cada componente disponible en la plataforma será sometido a un proceso de validación técnica inicial y posteriormente evaluado por la comunidad mediante valoraciones y reseñas detalladas, permitiendo identificar claramente aquellos componentes más confiables, efectivos y robustos. Esta retroalimentación continua fomentará un ecosistema dinámico y autorregulado, donde la calidad técnica y la eficiencia estarán garantizadas, proporcionando confianza y seguridad al momento de reutilizar recursos existentes.

Finalmente, esta área de descubrimiento potenciará el intercambio global de conocimientos, fomentando una cultura colaborativa que traspase barreras geográficas y culturales. Los desarrolladores no solo podrán acceder al trabajo y la experiencia técnica de otros profesionales, sino que además tendrán la oportunidad de contribuir activamente enriqueciendo este amplio repositorio, generando así un impacto positivo en toda la comunidad tecnológica y elevando el estándar general del desarrollo web profesional.

A continuación, se desarrollan en profundidad y formalmente los perfiles o roles de usuario previstos para la plataforma web del proyecto:

## 2. PERFILES Y ROLES DE USUARIO

### 1. Usuario Invitado (No Registrado)

El **usuario invitado** representa el nivel más básico de acceso y está orientado a aquellos visitantes que exploran la plataforma de manera ocasional o inicial, sin haber formalizado un registro previo. Este tipo de usuario cuenta con acceso limitado y esencialmente exploratorio.

#### Permisos y Accesos:

- **Visualización de proyectos:**
  - Puede consultar una selección pública y limitada de componentes, fragmentos de código y proyectos publicados por usuarios registrados.
  - No dispone de opciones avanzadas de búsqueda, clasificación o personalización en los resultados.
- **Limitaciones específicas:**
  - No puede crear, modificar ni publicar componentes o proyectos.
  - No tiene acceso a funciones avanzadas, como votaciones, comentarios o interacción directa con los creadores de contenido.
  - No puede participar en desafíos o actividades comunitarias de la plataforma.
  - No tiene acceso al foro exclusivo para usuarios registrados o Premium.

---

## 2. Usuario Registrado

El **usuario registrado** conforma el núcleo principal de la comunidad y se caracteriza por disponer de un acceso más completo, orientado al uso frecuente y activo de la plataforma, habiendo formalizado previamente su registro mediante una cuenta verificada.

### Permisos y Accesos:

- **Creación y publicación:**

- Puede crear, editar, gestionar y publicar componentes y proyectos en la plataforma.
  - Acceso completo al editor integrado para desarrollar componentes utilizando tecnologías como HTML, CSS y JavaScript.
- **Interacción comunitaria:**
- Puede participar plenamente en desafíos técnicos semanales o mensuales.
  - Puede valorar, comentar y dar retroalimentación técnica sobre proyectos y componentes publicados por otros usuarios.
  - Dispone de opciones avanzadas para personalizar la búsqueda y acceso a todo el catálogo de componentes publicados públicamente por otros usuarios.
- **Gestión personal:**
- Acceso a un panel personal para administrar su portafolio, componentes, historial de actividades y configuración de cuenta.
  - Posibilidad de enlazar sus proyectos con repositorios externos como GitHub o GitLab.
- **Limitaciones específicas:**
- No dispone de acceso al foro especializado y exclusivo reservado para usuarios Premium.
  - No tiene privilegios de moderación ni administración.

### **3. Usuario Premium (PRO)**

El **usuario Premium** representa el máximo nivel de usuario no administrativo, obteniendo acceso adicional mediante una suscripción mensual o anual. Este perfil está especialmente orientado a aquellos usuarios que buscan profundizar en conocimientos avanzados y aprovechar plenamente los recursos comunitarios más especializados.

#### **Permisos y Accesos:**

- **Todas las funcionalidades del usuario registrado**, incluyendo creación, gestión, publicación e interacción completa con el catálogo y comunidad general.
- **Foro exclusivo de la plataforma:**
  - Acceso privilegiado a un foro privado y moderado, exclusivo para miembros Premium.
  - Posibilidad de iniciar y participar activamente en debates técnicos avanzados, recibir respuestas detalladas de otros usuarios Premium con experiencia destacada y solucionar dudas concretas mediante asesoramiento personalizado.

---

### **4. Administrador (Admin)**

El perfil del **administrador** está reservado para los gestores técnicos y operativos responsables del control, supervisión y mantenimiento del funcionamiento general de la plataforma. Sus privilegios son máximos y comprenden funciones críticas destinadas a asegurar la calidad, seguridad y eficiencia operativa de todo el sistema.

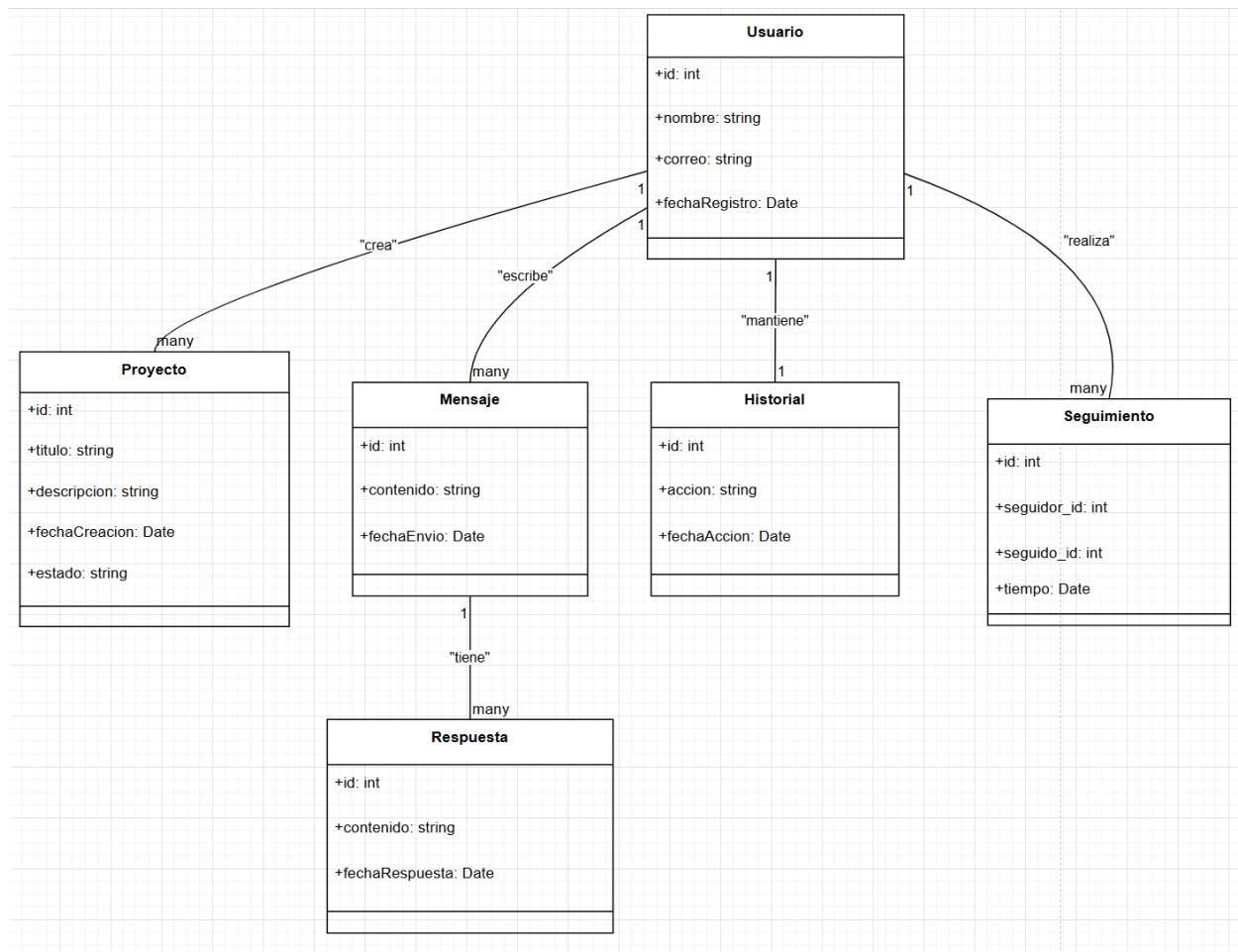
#### **Permisos y Accesos:**

- **Acceso completo a todas las áreas del sistema:**
  - Visualización y modificación integral de todos los componentes, proyectos y usuarios registrados en la plataforma.
- **Gestión de usuarios:**
  - Capacidad total para administrar perfiles de usuarios, incluyendo la creación, edición y eliminación completa de cuentas de usuarios registrados o Premium.
  - Puede suspender temporalmente o permanentemente usuarios en caso de infracciones o comportamientos inadecuados.
- **Moderación general de contenido:**
  - Autoridad para eliminar o modificar proyectos, componentes, comentarios o cualquier contenido inapropiado o técnicamente incorrecto publicado por usuarios.
  - Gestión total del foro exclusivo Premium, incluyendo aprobación de publicaciones, eliminación de mensajes inapropiados o irrelevantes, y organización de temas relevantes.
- **Control y supervisión técnica:**
  - Acceso completo a herramientas administrativas para monitorear rendimiento, seguridad y uso general de la plataforma.
  - Implementación y supervisión de actualizaciones técnicas, mantenimiento preventivo y correctivo, garantizando la estabilidad y seguridad continuas del sistema.

- **Privilegios adicionales:**

- Administración directa del sistema de desafíos, incluyendo creación, evaluación y reconocimiento de ganadores y participantes destacados.
- Gestión integral del modelo de monetización, supervisando pagos, suscripciones y atención directa a usuarios Premium en aspectos administrativos.

### 3. DIAGRAMA ENTIDAD/RELACIÓN



### Explicación del Origen de Atributos

## 1. Entidad Usuario

- **id**: Identificador único del usuario, generado automáticamente.
- **user**: Nombre del usuario, elegido por él mismo.
- **email**: Dirección electrónica para autentificación y comunicación.
- **password**: Contraseña de acceso.
- **urlFoto**: Dirección web de la imagen del perfil.
- **descripcion**: Breve descripción o biografía del usuario.
- **verificado**: Estado que indica si el usuario ha validado su cuenta.
- **fecha\_registro**: Fecha en que el usuario creó su cuenta.
- **rol**: Define el nivel de acceso (Desarrollador, Admin).

## 2. Entidad Proyecto

- **id**: Identificador único generado automáticamente.
- **titulo**: Nombre descriptivo del proyecto.
- **html, css, js**: Código del proyecto en los respectivos lenguajes.
- **descripcion\_proyecto**: Breve explicación del proyecto.
- **categoria**: Clasificación temática del proyecto.

- **fecha\_subido**: Momento exacto de publicación del proyecto.
- **id\_usuario**: Identifica al creador del proyecto (clave foránea a Usuario).

### 3. Entidad Mensaje

- **id**: Identificador único del mensaje, generado automáticamente.
- **texto**: Contenido escrito del mensaje.
- **id\_usuario**: Usuario que publicó el mensaje (clave foránea).
- **fecha**: Fecha de publicación del mensaje.
- **recurso**: Archivo adjunto o referencia adicional al mensaje.

### 4. Entidad Respuesta

- **id**: Identificador único generado automáticamente.
- **texto**: Contenido de la respuesta.
- **id\_mensaje**: Referencia al mensaje original al que responde.
- **fecha**: Fecha de publicación.
- **recurso**: Archivo o contenido adjunto adicional.
- **id\_usuario**: Usuario autor de la respuesta (clave foránea).

### 5. Entidad Actividad reciente

- **id**: Identificador único generado automáticamente.
- **id\_usuario**: Usuario que realizó la acción registrada.
- **accion**: Descripción breve de la actividad realizada (por ejemplo: "creó proyecto", "editó perfil").
- **fecha**: Fecha exacta de la acción.

## 6. Entidad Usuario-Usuario (Relación Seguimiento)

- **id**: Identificador único de la relación.
  - **id\_usu1 e id\_usu2**: Usuarios involucrados en la relación (seguido y seguidor, o amigos mutuamente).
  - **tiempo**: Fecha en que se estableció la relación.
- 

## Justificación de Claves Primarias

En todas las entidades se ha elegido un campo **id** como clave primaria porque:

- **Unicidad**: Cada registro debe poder ser identificado inequívocamente.
- **Consistencia**: Facilita el mantenimiento y la gestión relacional entre tablas mediante claves foráneas.
- **Eficiencia**: Las claves numéricas autoincrementales permiten búsquedas rápidas y operaciones eficientes en la base de datos.

Las claves primarias generadas automáticamente mediante autoincremento (AUTO\_INCREMENT) proporcionan estabilidad y simplicidad en la gestión de las relaciones, evitando problemas derivados de la modificación manual de identificadores.

---

## Relaciones clave-foránea

- **Usuario-Proyecto:** Se enlazan mediante el atributo **id\_usuario** en la entidad Proyecto.
- **Usuario-Mensaje/Respuesta:** A través del atributo **id\_usuario**, registrando qué usuario realizó cada acción comunicativa.
- **Mensaje-Respuesta:** Mediante **id\_mensaje**, lo que permite una estructura jerárquica de conversación.
- **Usuario-Historial:** Vinculación mediante **id\_usuario**, registrando todas las acciones relevantes realizadas por un usuario específico.
- **Usuario-Usuario (Amistad/Seguimiento):** Mediante **id\_usu1** y **id\_usu2**, que identifican claramente quién establece relación con quién.

## 4. PASO A TABLA

Usuario
<b>id</b> (INT, clave primaria, AUTO_INCREMENT)
nombre (VARCHAR)

correo (VARCHAR, único)

fechaRegistro (DATE)

## Proyecto

**id** (INT, clave primaria, AUTO\_INCREMENT)

titulo (VARCHAR)

descripcion (TEXT)

fechaCreacion (DATE)

estado (VARCHAR)

**id\_usuario** (INT, clave foránea referenciada a  
Usuario.id)

Mensaje
<b>id</b> (INT, clave primaria, AUTO_INCREMENT)
contenido (TEXT)
fechaEnvio (DATE)
<b>id_usuario</b> (INT, clave foránea referenciada a Usuario.id)

Comentario
<b>id</b> (INT, clave primaria, AUTO_INCREMENT)
contenido (TEXT)
fechaRespuesta (DATE)
<b>id_mensaje</b> (INT, clave foránea referenciada a Mensaje.id)

## **Actividad Reciente**

**id** (INT, clave primaria, AUTO\_INCREMENT)

accion (VARCHAR)

fechaAccion (DATE)

**id\_usuario** (INT, clave foránea referenciada a  
Usuario.id)

## **Seguidores**

**id** (INT, clave primaria, AUTO\_INCREMENT)

**id\_usu1** (INT, clave foránea referenciada a Usuario.id)

**id\_usu2** (INT, clave foránea referenciada a  
Usuario.id)

tiempo (DATE)

## 5. IMPLANTACIÓN A LA BASE DE DATOS

```
CREATE TABLE `usuarios` (
    `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
    `user` varchar(20) NOT NULL,
    `email` varchar(50) NOT NULL,
    `password` varchar(100) NOT NULL,
    `urlFoto` varchar(50) NOT NULL,
    `descripcion` varchar(500) NOT NULL,
    `verificado` int(11) DEFAULT 0,
    `fecha_registro` date NOT NULL DEFAULT current_timestamp(),
    `rol` varchar(14) NOT NULL DEFAULT 'desarrollador',
    PRIMARY KEY (`id`)
);
```

```
CREATE TABLE `proyectos` (
    `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
    `titulo` varchar(30) NOT NULL,
```

```
`html` longtext DEFAULT NULL,  
 `css` longtext DEFAULT NULL,  
 `js` longtext DEFAULT NULL,  
 `descripcion_proyecto` varchar(500) NOT NULL,  
 `categoria` varchar(30) NOT NULL,  
 `fecha_subido` timestamp NOT NULL DEFAULT current_timestamp(),  
 `id_usuario` bigint(20) UNSIGNED NOT NULL,  
 PRIMARY KEY (`id`),  
 KEY `fk_pro_usu` (`id_usuario`),  
 CONSTRAINT `fk_pro_usu` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios` (`id`) ON  
 DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE `mensajes_usuarios` (  
 `id` int(11) NOT NULL AUTO_INCREMENT,  
 `texto` text NOT NULL,  
 `id_usuario` bigint(20) UNSIGNED NOT NULL,  
 `fecha` date DEFAULT current_timestamp(),  
 `recurso` varchar(50) NOT NULL,  
 PRIMARY KEY (`id`),  
 KEY `fk_mes_usu` (`id_usuario`),
```

```
CONSTRAINT `fk_mes_usu` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
```

```
);
```

```
CREATE TABLE `comentarios_usuarios` (
```

```
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `texto` varchar(200) NOT NULL,
 `id_mensaje` int(20) NOT NULL,
 `fecha` date NOT NULL DEFAULT current_timestamp(),
 `recurso` varchar(50) NOT NULL,
```

```
 `id_usuario` bigint(20) UNSIGNED NOT NULL,
PRIMARY KEY (`id`),
KEY `fk_com_mes` (`id_mensaje`),
KEY `fk_com_usu` (`id_usuario`),
```

```
CONSTRAINT `fk_com_mes` FOREIGN KEY (`id_mensaje`) REFERENCES `mensajes_usuarios`(`id`)
ON DELETE CASCADE ON UPDATE CASCADE,
```

```
CONSTRAINT `fk_com_usu` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
```

```
);
```

```
CREATE TABLE `actividad_usuarios` (
```

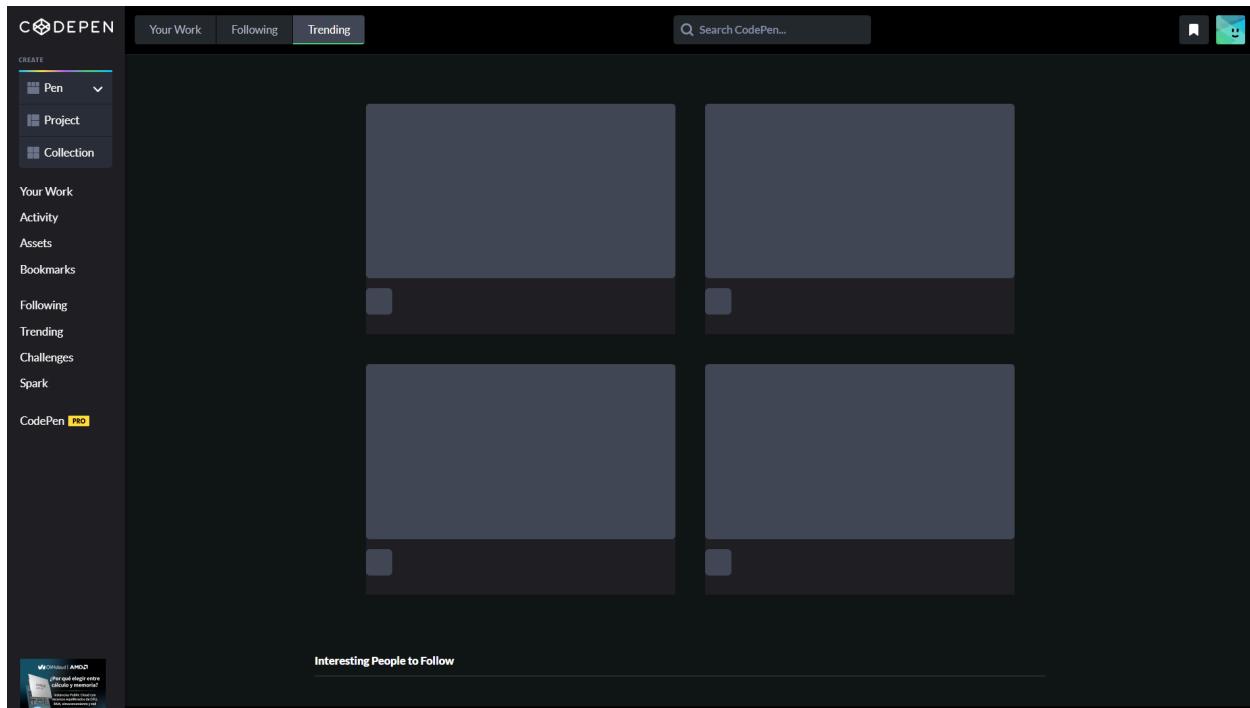
```
 `id` int(11) NOT NULL AUTO_INCREMENT,
```

```
`id_usuario` bigint(11) UNSIGNED NOT NULL,  
 `accion` varchar(30) NOT NULL,  
 `fecha` date NOT NULL DEFAULT current_timestamp(),  
 PRIMARY KEY (`id`),  
 KEY `fk_act_pusu` (`id_usuario`),  
 CONSTRAINT `fk_act_pusu` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios` (`id`) ON  
 DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE `seguidres_usuarios` (  
 `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,  
 `id_usu1` bigint(11) UNSIGNED NOT NULL,  
 `id_usu2` bigint(11) UNSIGNED NOT NULL,  
 `tiempo` date NOT NULL,  
 PRIMARY KEY (`id`),  
 KEY `fk_usu_susu1` (`id_usu1`),  
 KEY `fk_usu_susu2` (`id_usu2`),  
 CONSTRAINT `fk_usu_susu1` FOREIGN KEY (`id_usu1`) REFERENCES `usuarios` (`id`) ON  
 DELETE CASCADE ON UPDATE CASCADE,  
 CONSTRAINT `fk_usu_susu2` FOREIGN KEY (`id_usu2`) REFERENCES `usuarios` (`id`) ON  
 DELETE CASCADE ON UPDATE CASCADE  
);
```

# 6. DISEÑO WEB

## Estudio de la competencia: CodePen



### 1. Introducción a la plataforma

CodePen es una plataforma en línea lanzada en el año 2012, diseñada para la edición, prueba y exhibición rápida de fragmentos de código web compuestos principalmente por HTML, CSS y JavaScript. Se caracteriza por facilitar la creación de prototipos, la experimentación visual y la publicación abierta para la comunidad técnica. Actualmente, se ha consolidado como una referencia en su ámbito debido a su amplia base de usuarios activos y su capacidad para promover la creatividad en el desarrollo web.

### 2. Análisis comparativo del entorno competitivo

#### a) Competidores directos

- **CodeSandbox:** Plataforma orientada a la creación de proyectos web completos, que ofrece herramientas avanzadas de colaboración en tiempo real, integración completa

con sistemas como GitHub y gestión eficiente de dependencias mediante paquetes NPM.

- **JSFiddle**: Herramienta centrada en facilitar pruebas rápidas y colaborativas, con un enfoque ligero, aunque limitado en términos de gestión avanzada de archivos y colaboración extendida.
- **Glitch, Replit, StackBlitz**: Plataformas más orientadas al desarrollo completo de aplicaciones, incluyendo soporte para backend, integración con servicios externos y funcionalidades de despliegue automatizado.

#### b) Competidores enfocados en educación e iniciación

- **JS Bin y PlayCode**: Ofrecen interfaces simples y accesibles, dirigidas a desarrolladores principiantes o para tareas de prototipado rápido, siendo plataformas menos complejas pero igualmente relevantes en contextos educativos o de experimentación inicial.

#### c) Competencia indirecta

- **Editores locales (Visual Studio Code, Sublime Text, Brackets, entre otros)**: Aunque su enfoque es esencialmente diferente, compiten indirectamente ofreciendo mayor control local, capacidad avanzada de personalización y gestión de proyectos extensos con un flujo de trabajo profesional consolidado.

### 3. Fortalezas y debilidades específicas de CodePen

#### Fortalezas destacadas:

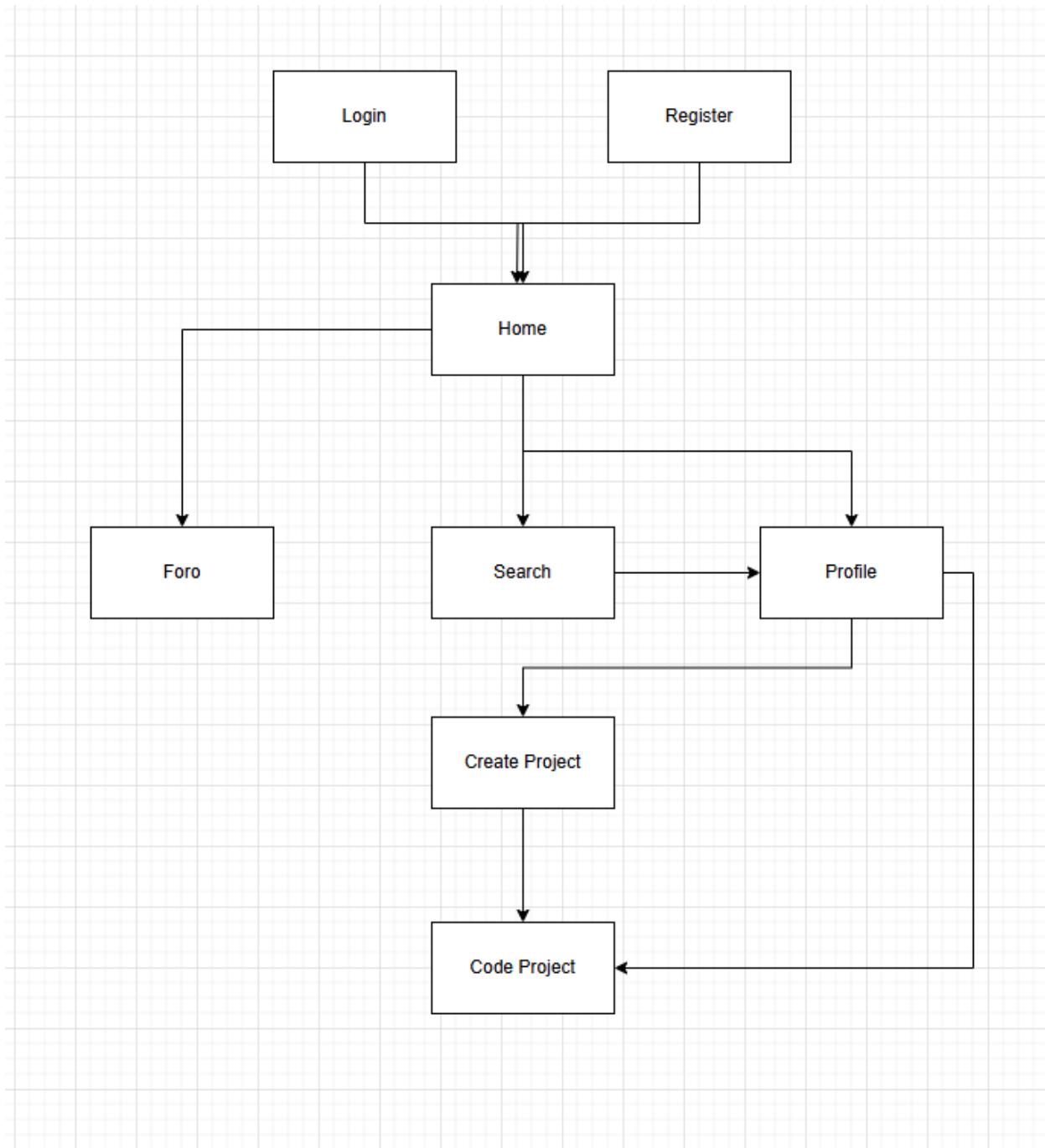
- Comunidad grande y activa que fomenta la retroalimentación constructiva.
- Facilidad para realizar prototipos visuales con previsualización instantánea.

- Sección "Trending" como mecanismo para destacar creaciones innovadoras y populares.

**Debilidades observadas:**

- Ausencia de foro de comunidad.
- Una visita agradable a los perfiles.
- La búsqueda e interfaz es muy poco intuitiva.

**Diagramación: mapa web y menús de navegación**



### Prototipos de bajo nivel: Wireframes

[https://www.figma.com/design/0fuXPidqB7pl6OqLwg7vbc/TFG?node-id=4-69&t=gAdDZbt42IA\\_NyYrt-1](https://www.figma.com/design/0fuXPidqB7pl6OqLwg7vbc/TFG?node-id=4-69&t=gAdDZbt42IA_NyYrt-1)

The wireframe illustrates the layout of the CompoDev website, featuring a header, a central content area with service cards, and a footer.

**Header:**

- Top navigation bar with links: Inicio, Explorar, Crear, Premium.
- User authentication buttons: Iniciar sesión, Registrarse.

**Section 1: Home Overview**

- Section Header:** CompoDev
- Text:** Crea, comparte y descubre componentes web
- Statistics:** 1,000+ Componentes, 90% Rendimiento
- Buttons:** CREAR COMPONENTE, EXPLORAR
- Placeholder:** A large empty rectangular box for displaying components.
- Decorative Buttons:** Six small rounded buttons labeled "lorem".

**Section 2: Nuestro Servicios**

- Section Header:** NUESTRO SERVICIOS
- Text:** Lorem Impsun aquí va una de texto increíble pero no se que va ahora mismo, porque esto se hace antes
- Service Cards:** Three identical service cards, each with:
  - Icon:** Cross icon.
  - Count:** 00
  - Title:** LOREM IMPSUN  
LOREM IMPSUN
  - Description:** Lorem Impsun aquí va una de texto increíble pero no se que va ahora mismo, porque esto se hace antes
  - Button:** LOREM

**Section 3: Footer**

- Section Header:** COMPODEV
- Section:** [Sobre Nosotros]
- Section:** [Plataforma]
- Section:** [Cuenta]
- Links:** Inicio, Explorar, Crear, Mi Perfil, Configuración, Cerrar Sesión
- Image:** Three small circular icons at the bottom left.
- Page Footer:** /\* © 2025 CompoDev Todos los derechos reservados \*/

En esta primera imagen que se habla de la Landing se puede observar cómo habrá un pequeño banner principal, con unos textos llamativos, el título de la aplicación y unas estadísticas sobre la misma.

Más abajo se pondrán los servicios que correspondan con la aplicación, en el momento en el que desarrollo el wireframe, no se realmente de qué servicios constará, es por eso que no queda especificado.

Es probable que añada alguna sección más pero mantendrá la estructura de la anterior sección.

The wireframe illustrates the user interface for the CompoDev search feature. At the top, there is a navigation bar with links for 'Inicio', 'Explorar', 'Crear', and 'Premium'. To the right of these are buttons for 'Iniciar sesión' (Sign In) and 'Registrarse' (Register). Below the navigation is the main title 'Buscador CompoDev' and a subtitle 'Encuentra componentes y desarrolladores para impulsar tus proyectos web'. There are two tabs: 'Proyectos' and 'Usuarios', with 'Usuarios' being the active tab. A search input field and a 'Buscar' button are located below the tabs. The main content area is titled 'USUARIOS ENCONTRADOS' and displays a table with columns for 'USUARIO', 'DESCRIPCIÓN', and 'ACCIONES'. Each row contains a placeholder profile picture and a 'Ver Perfil' button. At the bottom of the page, there is a section titled 'COMPODEV' and a footer with links for 'Sobre Nosotros', 'Plataforma', 'Cuenta', and copyright information.

USUARIO	DESCRIPCIÓN	ACCIONES
		<button>Ver Perfil</button>

**COMPODEV**

[Sobre Nosotros] [Plataforma] [Cuenta]

Inicio  Mi Perfil  
 Explorar  Configuración  
 Crear  Cerrar Sesión

/\* c 2025 CompoDev Todos los derechos reservados \*/

En esta segunda imagen que corresponde a la página del **Buscador**, se establece una estructura centrada en la funcionalidad de búsqueda, permitiendo al usuario encontrar tanto proyectos como desarrolladores dentro de la plataforma.

La cabecera presenta un título principal y un subtítulo que define el propósito de la sección. Justo debajo se sitúa un selector de categoría con dos opciones principales: "Proyectos" y "Usuarios". A continuación, se dispone de una barra de búsqueda y un botón para ejecutar la consulta.

La sección inferior muestra los **resultados**: en este caso, una tabla de usuarios con columnas para nombre, descripción y acciones disponibles, como ver el perfil. La estructura está claramente orientada a la exploración eficiente y a la interacción rápida.

El pie de página replica el formato general de la plataforma, asegurando coherencia visual y funcional.

Puedes enviar la siguiente imagen cuando lo deseas.

Inicio Explorar Crear Premium

Iniciar sesión Registrarse

### Acceso CompoDev

Este es un texto de relleno y no se que poner

Nombre de usuario

Nombre de usuario

Contraseña

Contraseña

Iniciar Sesión

Recordar? Regístrate

# COMPODEV

[Sobre Nosotros]

● ● ●

[Plataforma]

Inicio

Explorar

Crear

[Cuenta]

Mi Perfil

Configuración

Cerrar Sesión

/\* c 2025 CompoDev Todos los derechos reservados \*/

En esta tercera imagen correspondiente a la sección de **Inicio de Sesión**, se observa una estructura centrada en una tarjeta de acceso ubicada en el centro de la vista. Dicha tarjeta contiene los campos clásicos de autenticación: nombre de usuario, contraseña y un botón de acceso. Se incluye también un enlace alternativo para registro, dirigido a quienes aún no poseen una cuenta.

El diseño es minimalista, focalizado exclusivamente en la funcionalidad de ingreso a la plataforma, sin elementos adicionales que puedan distraer al usuario. El pie de página mantiene coherencia estructural con las demás secciones del sitio.

Puedes proceder con la siguiente imagen cuando lo deseas.

The image shows a wireframe of a registration page. At the top, there is a navigation bar with links for 'Inicio', 'Explorar', 'Crear', and 'Premium'. To the right of these are buttons for 'Iniciar sesión' and 'Registrarse'. The main content area is titled 'Registrarse' and contains placeholder text: 'Esto es un texto de relleno y no se que poner'. Below this is a large circular placeholder for a profile picture. The registration form consists of several input fields: 'Nombre de usuario' (two fields), 'Correo electrónico' (two fields), 'Contraseña' (two fields), 'Confirmar Contraseña' (two fields), and 'Descripción' (one field). At the bottom of the form is a large button labeled 'Registrarse'. Below this button is a link 'Recordar? Regístrate'.

**COMPODEV**

[Sobre Nosotros]

• • •

[Plataforma]

Inicio  
 Explorar  
 Crear

[Cuenta]

Mi Perfil  
 Configuración  
 Cerrar Sesión

/\* c 2025 CompoDev Todos los derechos reservados \*/

En esta cuarta imagen se representa la sección de **Registro de usuario**. La estructura sigue una lógica vertical, centrada en una tarjeta de formulario que solicita todos los datos necesarios para la creación de una cuenta en la plataforma.

El formulario incluye los siguientes campos: nombre de usuario, correo electrónico, contraseña y confirmación de contraseña, así como una descripción breve del perfil. También se prevé la posibilidad de añadir una imagen de usuario en la parte superior del formulario.

La disposición mantiene el diseño coherente con el resto de páginas, destacando la simplicidad, la claridad funcional y la continuidad estética. El pie de página conserva el formato corporativo de la aplicación.

Puedes enviar la siguiente imagen cuando lo estimes oportuno.

**EVOLUCION A PREMIUM**

LOREM IMPSUN NO SE QUE MAS PONER

**ACCESO PREMIUM**

- ◆ Lorem impsun que loco
- ◆ Lorem impsun que loco

Mensual      Anual

Activar ahora

**COMPODEV**

[Sobre Nosotros]

[Plataforma]

[Cuenta]

/\* c 2025 CompoDev Todos los derechos reservados \*/

Esta quinta imagen corresponde a la sección de **Evolución a Premium** dentro de la plataforma. El diseño presenta un enfoque claro y directo hacia la conversión del usuario registrado a un plan de suscripción superior.

En la parte central se expone un título destacado y un subtítulo orientativo. A continuación, se encuentra una tarjeta que enumera las ventajas o beneficios del acceso Premium, acompañada de una opción para seleccionar entre modalidad **mensual** o **anual**. El llamado a la acción se concreta con un botón para activar la suscripción.

La estructura se mantiene sobria, con jerarquía visual clara, centrada en persuadir mediante simplicidad. El pie de página repite la disposición estandarizada de las demás secciones.

Puedes remitir la siguiente imagen cuando lo consideres oportuno.

The wireframe illustrates a user interface for a platform named CompoDev. At the top, there's a header bar with a logo, navigation links (Inicio, Explorar, Crear, Premium), and buttons for Iniciar sesión and Registrarse. Below the header is a large profile card featuring a placeholder image, the name "NOMBRE", an email address "correo@correo.com", and three metrics: 0 Componentes, 0 Seguidores, and 0 Seguidos. The main content area is titled "COMPONENTES" and displays two component cards, each with a title, category, name, and two small icons. The bottom section is a dark footer with the CompoDev logo, links for Sobre Nosotros, Plataforma, and Cuenta, and a footer note indicating copyright from 2025.

Inicio   Explorar   Crear   Premium

Iniciar sesión   Registrarse

NOMBRE

correo@correo.com

0 Componentes   0 Seguidores   0 Seguidos

COMPONENTES

TITULO

Categoría

Nombre

TITULO

Categoría

Nombre

COMPODEV

[Sobre Nosotros]

[Plataforma]

[Cuenta]

Inicio

Explorar

Crear

Mi Perfil

Configuración

Cerrar Sesión

/\* © 2025 CompoDev Todos los derechos reservados \*/

En esta sexta imagen se representa la **vista de perfil de usuario**, organizada en dos bloques claramente diferenciados.

En la parte superior se dispone una tarjeta con los datos principales del usuario: nombre, correo electrónico y contadores numéricos relativos a su actividad en la plataforma (número de componentes publicados, seguidores y seguidos). También se incluye una fotografía de perfil y un botón de configuración o edición.

En la sección inferior se muestra un área titulada “Componentes”, donde se visualizan los proyectos o fragmentos desarrollados por ese usuario. Cada uno incluye su título, categoría, autor y botones de acción.

El diseño ofrece una visión sintética del historial público del usuario dentro del ecosistema, favoreciendo la navegación por su contenido y el seguimiento de su actividad.

Puedes continuar con la siguiente imagen cuando deseas.



Inicio Explorar Crear Premium

Iniciar sesión

Registrarse

## Panel de Administración

Loremp impsun no se que poner por aqui

 **LoREM IMPsUN**

Últimas 24 horas:

0

Últimas semana:

0

Últimas mes:

0

Últimas año:

0

 **LoREM IMPsUN**

Últimas 24 horas:

0

Últimas semana:

0

Últimas mes:

0

Últimas año:

0

 **LoREM IMPsUN**

Últimas 24 horas:

0

Últimas semana:

0

Últimas mes:

0

Últimas año:

0

# COMPODEV

[Sobre Nosotros]



[ Plataforma ]

- Inicio
- Explorar
- Crear

[ Cuenta ]

- Mi Perfil
- Configuración
- Cerrar Sesión

Esta séptima imagen representa la sección correspondiente al **Panel de Administración**, concebida para el uso exclusivo de perfiles con privilegios elevados (administradores del sistema).

El contenido principal se organiza en tarjetas estadísticas, cada una destinada al seguimiento de métricas específicas (como pueden ser publicaciones, nuevos registros, reportes o acciones de moderación). Cada tarjeta incluye un resumen por rangos temporales (últimas 24 horas, semana, mes y año), lo que permite una visión general del estado de la plataforma y sus variaciones recientes.

La interfaz está diseñada con un enfoque analítico, destinada a facilitar la supervisión operativa y la toma de decisiones rápidas por parte del equipo de administración. El pie de página se mantiene en consonancia con el resto del sistema.

Puedes continuar con la siguiente imagen cuando lo deseas.

The screenshot shows a web application interface with the following components:

- Header:** A dark header bar with a logo on the left, followed by navigation links: Inicio, Explorar, Crear, Premium, and two buttons on the right: Iniciar sesión and Registrarse.
- Title:** A large, bold title "Admin Dato" centered above the main content area.
- Text Area:** A section containing the text "Texto plano explicando que tipo de dato se modera".
- Search Bar:** A search input field with a placeholder and a "Buscar" button.
- Data Table:** A table with the following structure:

dato1	dato2	dato3	dato4	ACCIONES
	dato2	dato3	dato4	<button>accion1</button> <button>accion2</button>
	dato2	dato3	dato4	<button>accion1</button> <button>accion2</button>
	dato2	dato3	dato4	<button>accion1</button> <button>accion2</button>
	dato2	dato3	dato4	<button>accion1</button> <button>accion2</button>
	dato2	dato3	dato4	<button>accion1</button> <button>accion2</button>
	dato2	dato3	dato4	<button>accion1</button> <button>accion2</button>
- Footer:** A dark footer section with the text "COMPODEV" in white, followed by three circular icons at the bottom left.
- Footer Links:** Three columns of links:
  - [Sobre Nosotros]
  - [Plataforma]
  - [Cuenta]

Under [Plataforma]:  Inicio,  Explorar,  Crear.

Under [Cuenta]:  Mi Perfil,  Configuración,  Cerrar Sesión.
- Page Bottom:** A small copyright notice at the very bottom: "/\* c 2025 CompoDev Todos los derechos reservados \*/"

Esta octava imagen muestra una sección de **gestión administrativa específica**, referida genéricamente como "Admin Dato". Está diseñada para la administración, revisión o moderación de una categoría concreta de datos dentro del sistema (por ejemplo, usuarios, componentes, mensajes, etc.).

La estructura presenta un encabezado con título, descripción explicativa del tipo de dato gestionado y un campo de búsqueda con botón de acción. A continuación, se dispone una tabla donde se listan los registros existentes, organizados en varias columnas que representan atributos o propiedades del elemento, y una columna final de acciones posibles (por ejemplo, editar, eliminar, aprobar, bloquear, etc.).

Esta vista permite al administrador actuar de forma rápida sobre cada fila del sistema de datos, garantizando un control exhaustivo y personalizado sobre los elementos moderables. El pie de página mantiene su formato unificado respecto al resto del sitio.

Puedes remitir la siguiente imagen cuando lo estimes conveniente.

**Paleta de colores y estilo**

## **PALETA DE COLORES**

**#0D0D1A**

**#1A1E33**

**#BEC1E6**

**#C63EFF**

- **Tonalidades predominantes:**
  - Fondo oscuro en tonos azulados (#0d0d1a o similar), que genera alto contraste y enfoque visual.
  - Elementos destacados (botones, etiquetas, íconos activos) en tonos **magenta/violeta** (#c63eff - #b13fff), lo que aporta un carácter moderno y tecnológico.
  - Tipografía blanca o gris claro para asegurar legibilidad sobre fondo oscuro.
- **Estilo visual:**
  - Uso de diseño plano y minimalista con bordes limpios, sin sombras excesivas ni relieves.
  - Íconos funcionales simples y vectorizados (editar y eliminar).
  - Resaltado claro de secciones mediante líneas de separación y encabezados con jerarquía tipográfica.
  - Efectos sutiles (como iluminación o desenfoques) en el fondo para crear dinamismo sin sobrecargar.

## Aplicación del diseño

Este estilo es coherente con la línea gráfica ya mostrada en los wireframes anteriores y puede ser considerado como **referente estético principal** de toda la interfaz de administración (y, con variantes, también del entorno público).

Se sugiere que toda la identidad visual de CompoDev mantenga esta cohesión cromática y estructural, ya que transmite profesionalidad, modernidad y claridad operativa, cualidades fundamentales en un entorno dirigido a desarrolladores web.

## Tipografía

La tipografía principal utilizada en este proyecto es Orbitron, una fuente futurista y geométrica que refuerza la estética cyberpunk y tecnológica de la interfaz. Además, en los estilos globales y componentes (por ejemplo, en los archivos Home.css y Subscription.css), se especifica el uso de font-family: 'Orbitron', sans-serif; para los títulos, encabezados y muchos textos destacados. Para el cuerpo y textos secundarios, se suele usar una fuente de respaldo como sans-serif.

The screenshot shows the homepage of CompoDev, a platform for creating and sharing web components. The top navigation bar includes links for Inicio, Explorar, Crear, Premium, Foro, and a user profile icon. The main header features the CompoDev logo and the tagline "Crea, comparte y descubre componentes web". A large purple circular graphic is centered above a section titled "NUESTROS SERVICIOS". Below this, three numbered boxes describe the platform's services: "CREACIÓN DE COMPONENTES", "PUBLICACIÓN DE PROYECTOS", and "BÚSQUEDA INTELIGENTE". Each service box includes a brief description and a list of supported technologies or features. The "BÚSQUEDA INTELIGENTE" box highlights an AI algorithm for finding specific components. Below these boxes is a section titled "Lo que dicen nuestros usuarios" (What our users say) featuring three reviews from users NeuroGamer, Kira\_Render, and VoicEncoder. At the bottom, key statistics are displayed: 7.5M+ Usuarios, 25K+ Componentes, 89% Retención, and 12.3% Crecimiento. The footer contains sections for "Sobre Nosotros", "Plataforma", and "Cuenta", along with social media links and a copyright notice.

**CompoDev**

Crea, comparte y descubre componentes web

**1,240+** **93%**

Componentes Rendimiento

**CREAR COMPONENTE** **EXPLORAR**

Interfaces Animaciones Formularios Gráficos Navegación Cards

**NUESTROS SERVICIOS**

Soluciones de vanguardia para desarrolladores del futuro

**01** **02**

**CREACIÓN DE COMPONENTES**

Desarrolla componentes web con nuestro sistema de edición neuronal con tecnología de sincronización en tiempo real

HTML CSS JS

**02**

**PUBLICACIÓN DE PROYECTOS**

Comparte tus creaciones en nuestra red neural distribuida con verificación de autenticidad mediante blockchain

NTF Cloud Sync

**03**

**BÚSQUEDA INTELIGENTE**

Algoritmo de búsqueda cuántica que encuentra exactamente lo que necesitas antes de que lo sepas

AI Neural Quantum

**Lo que dicen nuestros usuarios**

**NeuroGamer** **Kira\_Render** **VoicEncoder**

"La plataforma revolucionó mi forma de programar. Los componentes son impresionantes y ahorran horas de trabajo. La cibernetica es su máxima expresión."

"Como diseñadora de interfaces neurales, encontré exactamente lo que necesitaba. La integración es perfecta y el soporte técnico responde en segundos."

"Después de digitalizar mi conciencia, CompoNet fue mi primera elección para desarrollar mi plataforma. Eficiencia cósmica y diseño de oro nítido."

18 mayo, 2025 3 abril, 2025 27 marzo, 2025

**7.5M+** **25K+** **89%** **12.3%**

Usuarios Componentes Retención Crecimiento

**COMPODEV**

**Sobre Nosotros** **Plataforma** **Cuenta**

El futuro del desarrollo web comienza. Una plataforma para desarrolladores que buscan crear y compartir componentes reutilizables.

[Inicio](#) [Mi Perfil](#)  
[Explorar](#) [Configuración](#)  
[Crear](#) [Cerrar Sesión](#)

© 2025 CompoDev • Todos los derechos reservados

CompoDev

Inicio Explorar Crear Premium Foro

Buscador CompoDev

Encuentra componentes y desarrolladores para impulsar tus proyectos web

Proyectos Usuarios

Buscar por título o categoría... BUSCAR

## COMPONENTES ENCONTRADOS

TÍTULO	CATEGORÍA	DESCRIPCIÓN	ACCIONES
asdfasfasdf	Animación	asdfasfasdf	<a href="#">Ver Proyecto</a>
qwerqwerqwer	Navegación	qwerqwerqwer	<a href="#">Ver Proyecto</a>
hey	Cards	asdfasdfs	<a href="#">Ver Proyecto</a>
asdfasdfs	asdfasdfs	Hey	<a href="#">Ver Proyecto</a>
rtyjrtjy4356jergje	wtherh	swfdgsh	<a href="#">Ver Proyecto</a>
rtyjrtjyty	jrtjyrtjy	jrtjyrtjyty	<a href="#">Ver Proyecto</a>
jtyjtyjt	j456jj	jyrtjyrtj	<a href="#">Ver Proyecto</a>
retyjr5657567	TjrtbjrTj	jSryhrjt546	<a href="#">Ver Proyecto</a>
rutkgg	hjkghjk	hjkghjk	<a href="#">Ver Proyecto</a>
erytertyertye	ertyerty	riyertyerty	<a href="#">Ver Proyecto</a>

« Anterior 1 2 Siguiente »

# COMPODEV

**[ Sobre Nosotros ]**

El futuro del desarrollo web compartido. Una plataforma para desarrolladores que buscan crear y compartir componentes reutilizables.

[Síguenos en GitHub](#) [Síguenos en Twitter](#) [Síguenos en LinkedIn](#)

**[ Plataforma ]**

- [Inicio](#)
- [Explorar](#)
- [Crear](#)

**[ Cuenta ]**

- [Mi Perfil](#)
- [Configuración](#)
- [Cerrar Sesión](#)

/\* © 2025 CompoDev • Todos los derechos reservados \*/



## Foro de CompoDev

¿Quéquieres compartir?

PUBLICAR MENSAJE

### ÚLTIMAS PUBLICACIONES

admin  
8 jun 2025, 02:00  
dwdwdwd

Comentar

Escribe un comentario...

Enviar

admin  
8 jun 2025, 02:00  
asdffffffwd

admin  
8 jun 2025, 02:00  
dwdwdw

sanches  
8 jun 2025, 02:00  
wdwdwdwd

agch  
7 jun 2025, 02:00  
Que guaywd

Comentar

agch  
6 jun 2025, 02:00  
asdf

Comentar

agch  
6 jun 2025, 02:00  
asdfasfasdf

Comentar

agch  
6 jun 2025, 02:00  
sdasdff

Comentar



# Crear Componente

Comparte tu creatividad con el mundo. Crea un nuevo componente para la comunidad CompoDev.

## Título del Componente

Ej: Animated Dropdown Menu

## Descripción

Describe tu componente, sus características y casos de uso...

## Categoría

UI/UX

Animación

Navegación

Formularios

Cards

Botones

Gráficos

Tablas

Layouts

Efectos

**Crear Componente**

CompoDev

≡



• AGCH

[Editar Perfil](#)

a@a.com

Hola, esto es una descripción 777

8 Componentes    0 Seguidores    0 Siguiendo

[Crear Nuevo](#)

### MIS COMPONENTES

<p>rutkgg</p> <p>Categoría: hjkghjk</p> <p>hjkghjkg</p> <p><a href="#">Editar</a> <a href="#">Borrar</a></p>	<p>retyjr5657567</p> <p>Categoría: 7jrthjr7j</p> <p>j5ryhrjt546</p> <p><a href="#">Editar</a> <a href="#">Borrar</a></p>
<p>jtyjrtyjrt</p> <p>Categoría: j456jj</p> <p>yjrtyjrt</p> <p><a href="#">Editar</a> <a href="#">Borrar</a></p>	<p>rtyjrtyjrt</p> <p>Categoría: jrtjrtyj</p> <p>jrtjrtyjrt</p> <p><a href="#">Editar</a> <a href="#">Borrar</a></p>

## 7. PROGRAMACIÓN DE LA WEB

---

## **backend/**

-  **admin\_panel/**
  -  **use\_cases/**
    - GetMessagesStats.php
    - GetProjectsStats.php
    - GetUsersStats.php
  - AdminController.php
-  **comments\_users/**
  -  **entities/**
    - CommentsUser.php
  -  **use\_cases/**
    - CreateCommentsUser.php
    - DeleteCommentsUser.php
    - GetAllCommentsUser.php
    - GetCommentsByMessage.php
    - GetCommentsUserById.php
    - UpdateCommentsUser.php
    - CommentsUserController.php
-  **config/**
  - database.php
-  **follow\_users/**
  -  **entities/**

- FollowUsers.php
-  **use\_cases/**
  - CountFollowers.php
  - FollowUser.php
  - GetFollowedUsers.php
  - UnfollowUser.php
  - FollowUsersController.php
-  **messages\_users/**
  -  **entities/**
    - MessagesUser.php
  -  **use\_cases/**
    - CreateMessagesUser.php
    - DeleteMessagesUser.php
    - GetAllMessagesUser.php
    - GetMessagesUserById.php
    - UpdateMessagesUser.php
    - MessagesUserController.php
-  **projects/**
  -  **entities/**
    - Project.php
  -  **use\_cases/**
    - CheckOwner.php

- CreateProject.php
  - DeleteProject.php
  - GetAllProjects.php
  - GetProjectById.php
  - GetProjectByUserId.php
  - UpdateCodeProject.php
  - UpdateProject.php
  - ProjectController.php
-  **users/**
    -  **entities/**
      - User.php
    -  **use\_cases/**
      - CompSession.php
      - CreateUser.php
      - DeleteUser.php
      - GetAllUsers.php
      - GetTopUsers.php
      - GetUserById.php
      - LoginUser.php
      - LogoutUser.php
      - UpdateUser.php

- VerifyUser.php
- UserController.php
- setup\_followers.php

```

2 references | 0 implementations
class GetUsersStats {
    5 references
    private $conn;

    2 references | 0 overrides
    public function __construct() {
        $db = new DB();
        $this->conn = $db->getConn();
    }

    2 references | 0 overrides
    public function execute(): void {
        try {
            $stats = [
                'last_24h' => $this->getUsersCountLast24Hours(),
                'last_week' => $this->getUsersCountLastWeek(),
                'last_month' => $this->getUsersCountLastMonth(),
                'last_year' => $this->getUsersCountLastYear()
            ];

            header(header: 'Content-Type: application/json');
            echo json_encode(value: $stats);
        } catch (Exception $e) {
            http_response_code(response_code: 500);
            echo json_encode(value: ['error' => 'Error al obtener estadísticas de usuarios: ' . $e->getMessage()]);
        }
    }

    1 reference
    private function getUsersCountLast24Hours(): int {
        $query = "SELECT COUNT(*) as total FROM usuarios WHERE fecha_registro >= DATE_SUB(NOW(), INTERVAL 24 HOUR)";
        $stmt = $this->conn->prepare(query: $query);
        $stmt->execute();
        $result = $stmt->get_result();
        $data = $result->fetch_assoc();
        return (int)$data['total'];
    }

    1 reference
    private function getUsersCountLastWeek(): int {
        $query = "SELECT COUNT(*) as total FROM usuarios WHERE fecha_registro >= DATE_SUB(NOW(), INTERVAL 7 DAY)";
        $stmt = $this->conn->prepare(query: $query);
        $stmt->execute();
        $result = $stmt->get_result();
        $data = $result->fetch_assoc();
        return (int)$data['total'];
    }

    1 reference
    private function getUsersCountLastMonth(): int {
        $query = "SELECT COUNT(*) as total FROM usuarios WHERE fecha_registro >= DATE_SUB(NOW(), INTERVAL 30 DAY)";
        $stmt = $this->conn->prepare(query: $query);
        $stmt->execute();
        $result = $stmt->get_result();
        $data = $result->fetch_assoc();
        return (int)$data['total'];
    }

    1 reference
    private function getUsersCountLastYear(): int {
        $query = "SELECT COUNT(*) as total FROM usuarios WHERE fecha_registro >= DATE_SUB(NOW(), INTERVAL 1 YEAR)";
        $stmt = $this->conn->prepare(query: $query);
        $stmt->execute();
        $result = $stmt->get_result();
        $data = $result->fetch_assoc();
        return (int)$data['total'];
    }
}

```

El fichero **GetUsersStats.php** contiene una clase denominada **GetUsersStats**, cuya finalidad es obtener estadísticas sobre los registros de usuarios en distintos intervalos de tiempo: últimas 24 horas, última semana, último mes y último año.

La clase mantiene una propiedad privada `$conn` que almacena la conexión a la base de datos. En el constructor se instancia un objeto de la clase `DB` y se obtiene dicha conexión mediante el método `getConn()`.

El método principal de la clase es `execute()`, que llama internamente a cuatro métodos privados (`getUsersCountLast24Hours`, `getUsersCountLastWeek`, `getUsersCountLastMonth` y `getUsersCountLastYear`). Cada uno de estos métodos ejecuta una consulta SQL que cuenta el número de usuarios registrados en la tabla `usuarios` a partir de una fecha calculada con la función `DATE_SUB(NOW(), INTERVAL ...)`, ajustando el intervalo correspondiente (24 horas, 7 días, 30 días y 1 año, respectivamente).

Los resultados se agrupan en un array asociativo y se devuelven como respuesta en formato JSON con cabecera de contenido `application/json`. En caso de que ocurra alguna excepción durante la ejecución, se devuelve un error HTTP 500 con un mensaje descriptivo.

Este fichero puede utilizarse, por ejemplo, para alimentar un panel de administración o un sistema de análisis de actividad de usuarios.

```
document: /commentary/insert/execute.php -> CreateCommentsUser.php -> CreateCommentsUser.php -> execute() -> [Edit] -> [Select] -> [New]
9  class CreateCommentsUser {
15     /*
16      1 reference | 0 overrides
17      public function execute($data): void {
18          try {
19              // Crear una instancia de la conexión a la BD
20              $db = new DB();
21              $conn = $db->getConn();
22
23              // Crear una nueva instancia de CommentsUser con los datos proporcionados
24              $commentsUser = new CommentsUser(
25                  texto: $data['texto'],
26                  id_mensaje: $data['id_mensaje'],
27                  id_usuario: $data['id_usuario'],
28                  fecha: isset($data['fecha']) ? $data['fecha'] : date(format: 'Y-m-d H:i:s'),
29                  recurso: $data['recurso']
30              );
31
32              // Preparar la consulta SQL para insertar el comentario
33              $sql = "INSERT INTO comentarios_usuarios (texto, id_mensaje, id_usuario, fecha, recurso)
34                  VALUES (?, ?, ?, ?, ?)";
35
36              $stmt = $conn->prepare(query: $sql);
37
38              // Asegurar que el texto sea tratado como string
39              $texto = (string) $commentsUser->texto;
40              $id_mensaje = (int) $commentsUser->id_mensaje;
41              $id_usuario = (int) $commentsUser->id_usuario;
42              $fecha = (string) $commentsUser->fecha;
43              $recurso = (string) $commentsUser->recurso;
44
45              // Registro para depuración
46              error_log(message: "Texto a insertar: '{$texto}'");
47              error_log(message: "ID mensaje: {$id_mensaje}");
48              error_log(message: "ID usuario: {$id_usuario}");
49
50              // Vincular los parámetros
51              $stmt->bind_param(
52                  types: "siiss",
53                  var: &$texto,
54                  vars: &$id_mensaje,
55                  $id_usuario,
56                  $fecha,
57                  $recurso
58              );
59
60              // Registrar los datos recibidos para diagnóstico
61              error_log(message: 'Intentando crear comentario con datos: ' . print_r(value: $data, return: true));
62
63          try {
64              // Ejecutar la consulta
65              if ($stmt->execute()) {
66                  $commentsUser->id = $conn->insert_id;
67                  // Cerrar la conexión
68                  $stmt->close();
69                  $conn->close();
70
71                  // Devolver respuesta exitosa con el ID generado
72                  http_response_code(response_code: 201);
73                  echo json_encode(value: [
74                      'message' => 'Comentario creado con éxito',
75                      'id' => $commentsUser->id
76                  ]);
77
78              } else {
79                  error_log(message: "Error en la consulta SQL: " . $stmt->error);
80                  throw new Exception(message: "Error al crear el comentario: " . $stmt->error);
81              }
82          } catch (Exception $queryEx) {
83              error_log(message: 'Excepción al ejecutar la consulta: ' . $queryEx->getMessage());
84              throw $queryEx;
85          }
86
87      } catch (Exception $e) {
88          http_response_code(response_code: 500);
89          echo json_encode(value: ['error' => $e->getMessage()]);
90      }
91  }
92 ?>
```

El fichero **CreateCommentsUser.php** define una clase llamada **CreateCommentsUser** que contiene un único método público **execute(\$data)**. Su función es insertar un nuevo comentario de usuario en la base de datos utilizando los datos proporcionados.

Primero, el método obtiene una conexión a la base de datos mediante una instancia de la clase DB. A continuación, crea un objeto **CommentsUser** y le asigna los valores recibidos en el array \$data, como el texto del comentario, el ID del mensaje al que responde, el ID del usuario que lo emite, la fecha del comentario (usando la actual si no se proporciona una) y el recurso asociado.

Luego se construye una consulta SQL de inserción con cinco campos (texto, **id\_mensaje**, id\_usuario, fecha, recurso) y se preparan los datos para su ejecución, convirtiendo los valores a los tipos adecuados (string o int) y **vinculándolos** con bind\_param().

Antes de ejecutar la consulta, se registran mensajes de depuración con error\_log() para facilitar el diagnóstico en caso de error. Si la ejecución es exitosa, se obtiene el ID generado automáticamente (insert\_id), se cierra la conexión y se devuelve una respuesta con código HTTP 201 y un mensaje de éxito en formato JSON.

Si ocurre algún error durante la preparación o ejecución de la consulta, se captura la excepción correspondiente, se registra un mensaje de error y se devuelve una respuesta de error con código HTTP 500 y un mensaje descriptivo.

Este fichero permite crear comentarios de usuarios de forma segura y controlada, con validación básica, manejo de errores y trazabilidad para depuración.

```
Scritto su /comentarios_usuarios /use_cases / ... DeleteCommentsUser.php
```

```
1 <?php
2
3 require_once __DIR__ . '/../../config/database.php';
4
5 /**
6  * Caso de uso para eliminar un comentario existente
7  */
8 class DeleteCommentsUser {
9
10    /**
11     * Ejecuta el caso de uso para eliminar un comentario
12     *
13     * @param int $id ID del comentario a eliminar
14     * @return void
15     */
16
17    public function execute($id): void {
18        try {
19            // Crear una instancia de la conexión a la BD
20            $db = new DB();
21            $conn = $db->getConn();
22
23            // Verificar primero si el comentario existe
24            $checkSql = "SELECT id FROM comentarios_usuarios WHERE id = ?";
25            $checkStmt = $conn->prepare(query: $checkSql);
26            $checkStmt->bind_param(types: "i", var: &$id);
27            $checkStmt->execute();
28            $checkResult = $checkStmt->get_result();
29
30            if ($checkResult->num_rows === 0) {
31                $checkStmt->close();
32                $conn->close();
33                http_response_code(response_code: 404);
34                echo json_encode(value: ['error' => 'Comentario no encontrado']);
35                return;
36            }
37
38            $checkStmt->close();
39
40            // Preparar la consulta SQL para eliminar el comentario
41            $sql = "DELETE FROM comentarios_usuarios WHERE id = ?";
42
43            $stmt = $conn->prepare(query: $sql);
44
45            // Vincular el parámetro
46            $stmt->bind_param(types: "i", var: &$id);
47
48            // Ejecutar la consulta
49            if ($stmt->execute()) {
50                // Cerrar la conexión
51                $stmt->close();
52                $conn->close();
53
54                // Devolver respuesta exitosa
55                http_response_code(response_code: 200);
56                echo json_encode(value: ['message' => 'Comentario eliminado con éxito']);
57            } else {
58                throw new Exception(message: "Error al eliminar el comentario: " . $stmt->error);
59            }
60
61            } catch (Exception $e) {
62                http_response_code(response_code: 500);
63                echo json_encode(value: ['error' => $e->getMessage()]);
64            }
65        }
66    ?>
```

El fichero **DeleteCommentsUser.php** define una clase llamada **DeleteCommentsUser**, cuyo objetivo es eliminar un comentario existente en la base de datos a partir de su identificador (ID).

El método principal **execute(\$id)** comienza estableciendo una conexión con la base de datos mediante la clase DB. Antes de intentar eliminar el comentario, se realiza una verificación para comprobar si el comentario con ese ID realmente existe. Para ello, se ejecuta una consulta SELECT filtrando por el campo id en la tabla **comentarios\_usuarios**.

Si no se encuentra ningún comentario con ese ID, el método cierra la conexión y devuelve una respuesta HTTP con código 404 y un mensaje en formato JSON indicando que el comentario no fue encontrado.

Si el comentario existe, se construye una segunda consulta SQL de tipo DELETE para eliminar el registro correspondiente. Se prepara la sentencia, se vincula el parámetro ID y se ejecuta la consulta. En caso de éxito, se cierra la conexión y se devuelve una respuesta HTTP 200 con un mensaje de confirmación.

Si ocurre un error durante la ejecución de la consulta, se lanza una excepción y se devuelve una respuesta con código HTTP 500, junto con un mensaje de error descriptivo.

Este fichero garantiza que solo se eliminan comentarios válidos y proporciona un manejo de errores adecuado, devolviendo respuestas claras según el resultado de la operación.

```
backend / comments_users / use_cases / 16 GetAllCommentsUser.php > FMP > 16 GetAllCommentsUser > [ Executed ] > Explain | Refactor
1  <?php
2
3  require_once __DIR__ . '/../../config/database.php';
4  require_once __DIR__ . '/../../entities/CommentsUser.php';
5
6  /**
7   * Caso de uso para obtener todos los comentarios de usuarios
8   */
9  1 reference | 0 implementations
10 class GetAllCommentsUser {
11     /**
12      * Ejecuta el caso de uso para obtener todos los comentarios
13      * @param int $page Número de página para paginación
14      * @param string|null $search Término de búsqueda opcional
15      * @param int $itemsPerPage Número de elementos por página
16      * @return array Resultado con los comentarios y metadatos de paginación
17     */
18     1 reference | 0 overrides
19     public function execute($page = 1, $search = null, $itemsPerPage = 10): array {
20         try {
21             // Crear una instancia de la conexión a la BD
22             $db = new DB();
23             $conn = $db->getConn();
24
25             // Calcular el offset para la paginación
26             $offset = ($page - 1) * $itemsPerPage;
27
28             // Preparar la consulta SQL base con JOIN a usuarios para obtener información del autor
29             $sql = "SELECT c.*, u.user AS nombre, u.email, u.urlFoto AS avatar
30                   FROM comentarios_usuarios c
31                   LEFT JOIN usuarios u ON c.id_usuario = u.id";
32             $countSql = "SELECT COUNT(*) AS total FROM comentarios_usuarios c";
33
34             $params = [];
35             $types = "";
36
37             // Si hay término de búsqueda, añadir cláusula WHERE
38             if ($search) {
39                 $sql .= " WHERE texto LIKE ? OR recurso LIKE ?";
40                 $countSql .= " WHERE texto LIKE ? OR recurso LIKE ?";
41                 $searchParam = "%$search%";
42                 $params = [$searchParam, $searchParam];
43                 $types = "ss";
44             }
45
46             // Añadir ordenamiento y límites para paginación
47             $sql .= " ORDER BY fecha DESC LIMIT ?, ?";
48             $params[0] = $offset;
49             $params[1] = $itemsPerPage;
50             $types .= "ii";
51
52             // Preparar y ejecutar la consulta para obtener el total
53             $countStmt = $conn->prepare(query: $countSql);
54             if ($search) {
55                 $countStmt->bind_param(types: "ss", var: &$searchParam, vars: &$searchParam);
56             }
57             $countStmt->execute();
58             $countResult = $countStmt->get_result();
59             $totalRow = $countResult->fetch_assoc();
60             $total = $totalRow['total'];
61             $countStmt->close();
62
63             // Preparar y ejecutar la consulta principal
64             $stmt = $conn->prepare(query: $sql);
65             if (!empty($types)) {
66                 $stmt->bind_param(types: $types, ...var: &$params);
67             }
68             $stmt->execute();
69             $result = $stmt->get_result();
70
71             // Procesar los resultados
72             $comments = [];
73             while ($row = $result->fetch_assoc()) {
74                 $comments[] = $row;
75             }
76
77             // Cerrar la conexión
78             $stmt->close();
79             $conn->close();
80
81             // Calcular metadatos de paginación
82             $totalPages = ceil(num: $total / $itemsPerPage);
83
84             // Devolver resultado
85             return [
86                 'data' => $comments,
87                 'pagination' => [
88                     'total' => $total,
89                     'totalPages' => $totalPages,
90                     'currentPage' => $page,
91                     'itemsPerPage' => $itemsPerPage
92                 ]
93             ];
94         } catch (Exception $e) {
95             return ['error' => $e->getMessage()];
96         }
97     }
98 }
```

El fichero **GetAllCommentsUser.php** define la clase **GetAllCommentsUser**, encargada de obtener todos los comentarios de usuarios registrados en la base de datos. Este caso de uso incluye soporte para paginación y búsqueda por texto.

El método principal **execute(\$page = 1, \$search = null, \$itemsPerPage = 10)** acepta como parámetros el número de página, un término de búsqueda opcional y el número de elementos por página.

Primero, se calcula el desplazamiento (**offset**) en función de la página actual. Luego, se construye una consulta SQL que une la tabla **comentarios\_usuarios** con la tabla **usuarios** para obtener información adicional del autor de cada comentario (como nombre de usuario, correo y avatar).

Si se proporciona un término de búsqueda, se añade una cláusula **WHERE** que filtra por coincidencias en los campos **texto** y **recurso**. La búsqueda se aplica tanto a la consulta principal como a la consulta que calcula el número total de registros coincidentes.

Las consultas utilizan **prepare()** y **bind\_param()** para evitar inyecciones SQL y asegurar el uso correcto de los parámetros. Se ejecuta primero la consulta de conteo (**SELECT COUNT(\*)**) para obtener el total de resultados y calcular el número de páginas. A continuación, se ejecuta la consulta principal con los resultados paginados y ordenados por fecha descendente.

Los resultados se recorren con un bucle **while** y se almacenan en un array asociativo. Finalmente, se cierra la conexión y se devuelve un array con los datos de los comentarios y los metadatos de paginación, incluyendo el total de elementos, páginas, página actual y elementos por página.

En caso de error, se captura la excepción y se devuelve un mensaje con la descripción del fallo. Esta clase permite mostrar comentarios de forma paginada, filtrarlos por contenido y asociarlos visualmente con sus autores.

```
/* Reference(s) implementation(s)
class GetCommentsByMessage {
    /**
     * Ejecuta el caso de uso para obtener comentarios por ID de mensaje
     *
     * @param int $messageId ID del mensaje del que queremos obtener los comentarios
     * @param int $page Número de página para paginación
     * @param int $itemsPerPage Número de elementos por página
     * @return void
     */
    1 reference | 0 overrides
    public function execute($messageId, $page = 1, $itemsPerPage = 10): void {
        try {
            // Crear una instancia de la conexión a la BD
            $db = new DB();
            $conn = $db->getConn();

            // Calcular el offset para la paginación
            $offset = ($page - 1) * $itemsPerPage;

            // Preparar la consulta SQL para obtener los comentarios del mensaje con datos completos del usuario
            // Usamos el campo id_mensaje para relacionar con el ID del mensaje
            $sql = "SELECT c.*, u.user, u.urlFoto, u.email, u.descripcion, u.fecha_registro
                    FROM comentarios_usuarios c
                    LEFT JOIN usuarios u ON c.id_usuario = u.id
                    WHERE c.id_mensaje = ?
                    ORDER BY c.fecha DESC
                    LIMIT ?, ?";
            $stmt = $conn->prepare($sql);
            // Consulta para obtener el total de comentarios
            $countSql = "SELECT COUNT(*) as total FROM comentarios_usuarios WHERE id_mensaje = ?";

            // Ejecutar consulta para obtener el total
            $countStmt = $conn->prepare(query: $countSql);
            $countStmt->bind_param(types: "i", var: &$messageId);
            $countStmt->execute();
            $countResult = $countStmt->get_result();
            $totalRow = $countResult->fetch_assoc();
            $total = $totalRow['total'];
            $countStmt->close();

            // Ejecutar la consulta principal
            $stmt = $conn->prepare(query: $sql);
            $stmt->bind_param(types: "iii", var: &$messageId, vars: &$offset, $itemsPerPage);
            $stmt->execute();
            $result = $stmt->get_result();

            // Procesar los resultados
            $comments = [];
            while ($row = $result->fetch_assoc()) {
                $comments[] = $row;
            }

            // Cerrar la conexión
            $stmt->close();
            $conn->close();

            // Calcular metadatos de paginación
            $totalPages = ceil(num: $total / $itemsPerPage);

            // Preparar respuesta
            $response = [
                'data' => $comments,
                'pagination' => [
                    'total' => $total,
                    'totalPages' => $totalPages,
                    'currentPage' => $page,
                    'itemsPerPage' => $itemsPerPage
                ]
            ];

            // Devolver respuesta en formato JSON
            http_response_code(response_code: 200);
            echo json_encode(value: $response, flags: JSON_UNESCAPED_UNICODE);

        } catch (Exception $e) {
            http_response_code(response_code: 500);
            echo json_encode(value: ['error' => $e->getMessage()]);
        }
    }
}

```

El fichero **GetCommentsByMessage.php** contiene la clase **GetCommentsByMessage**, cuyo propósito es obtener los comentarios asociados a un mensaje concreto, identificándolo mediante su ID. El método principal **execute(\$messagId, \$page = 1, \$itemsPerPage = 10)** permite también aplicar paginación.

En primer lugar, el método establece una conexión con la base de datos utilizando una instancia de la clase **DB**. Después, calcula el desplazamiento (**offset**) necesario en función de la página solicitada.

A continuación, se prepara una consulta SQL que selecciona los comentarios (**comentarios\_usuarios**) relacionados con el ID de mensaje proporcionado, realizando un **LEFT JOIN** con la tabla **usuarios** para obtener información adicional del autor del comentario (nombre de usuario, email, avatar, descripción y fecha de registro). Los comentarios se ordenan por fecha descendente y se limita el número de resultados según el valor de **itemsPerPage**.

Antes de ejecutar esta consulta principal, se realiza una consulta adicional para contar el número total de comentarios que pertenecen al mensaje indicado. Este valor se utiliza para calcular el número total de páginas que existirían con esa cantidad de resultados y el tamaño de página dado.

Una vez ejecutada la consulta principal, los resultados se recorren en un bucle **while** y se almacenan en un array. Finalmente, se devuelve una respuesta JSON con los datos de los comentarios y los metadatos de paginación (total de elementos, total de páginas, página actual y elementos por página). En caso de error, se lanza una excepción y se responde con código HTTP 500 y un mensaje descriptivo.

Este caso de uso está pensado para mostrar, por ejemplo, los comentarios que recibe un mensaje o publicación en un sistema de foro, blog o red social.

```
1 <?php
2
3 require_once __DIR__ . '/../../config/database.php';
4 require_once __DIR__ . '/../entities/CommentsUser.php';
5
6 /**
7  * Caso de uso para obtener un comentario específico por su ID
8 */
9 1 reference | 0 implementations
10 class GetCommentsUserId {
11     /**
12      * Ejecuta el caso de uso para obtener un comentario por ID
13      *
14      * @param int $id ID del comentario a obtener
15      * @return void
16      */
17     1 reference | 0 overrides
18     public function execute($id): void {
19         try {
20             // Crear una instancia de la conexión a la BD
21             $db = new DB();
22             $conn = $db->getConn();
23
24             // Preparar la consulta SQL para obtener el comentario
25             $sql = "SELECT * FROM comentarios_usuarios WHERE id = ?";
26
27             // Vincular el parámetro
28             $stmt = $conn->prepare(query: $sql);
29
30             // Ejecutar la consulta
31             $stmt->execute();
32             $result = $stmt->get_result();
33
34             if ($result->num_rows > 0) {
35                 $comment = $result->fetch_assoc();
36
37                 // Cerrar la conexión
38                 $stmt->close();
39                 $conn->close();
40
41                 // Devolver el comentario encontrado
42                 http_response_code(response_code: 200);
43                 echo json_encode(value: $comment);
44             } else {
45                 // No se encontró el comentario
46                 http_response_code(response_code: 404);
47                 echo json_encode(value: ['error' => 'Comentario no encontrado']);
48             }
49
50         } catch (Exception $e) {
51             http_response_code(response_code: 500);
52             echo json_encode(value: ['error' => $e->getMessage()]);
53         }
54     }
55 }
56 ?>
```

El fichero **GetCommentsUserById.php** contiene la clase **GetCommentsUserById**, cuyo objetivo es recuperar un comentario específico de la base de datos a partir de su identificador único.

El método **execute(\$id)** recibe como parámetro el ID del comentario. Primero, establece una conexión a la base de datos utilizando la clase **DB**. A continuación, prepara una consulta SQL **SELECT \* FROM comentarios\_usuarios WHERE id = ?**, utilizando una sentencia preparada para evitar inyecciones SQL, y vincula el parámetro correspondiente.

La consulta se ejecuta y se analiza el resultado. Si se encuentra un comentario con ese ID, se recupera con **fetch\_assoc()**, se cierra la conexión y se devuelve una respuesta con código HTTP 200 y el comentario en formato JSON.

Si no se encuentra ningún resultado, se devuelve una respuesta con código HTTP 404 y un mensaje de error indicando que el comentario no fue encontrado.

En caso de que ocurra una excepción durante la ejecución de la consulta, se devuelve una respuesta con código HTTP 500 y un mensaje de error detallado.

Este fichero se utiliza para obtener los datos completos de un comentario concreto, por ejemplo, cuando se desea mostrar su contenido en una vista de detalle o cargarlo para su edición.

```

9   class UpdateCommentsUser {
10
11     * Ejecuta el caso de uso para actualizar un comentario
12     *
13     * @param int $id ID del comentario a actualizar
14     * @param array $data Los nuevos datos para el comentario
15     * @return void
16   */
17   1 reference | 0 overrides
18   public function execute($id, $data): void {
19     try {
20       // Crear una instancia de la conexión a la BD
21       $db = new DB();
22       $conn = $db->getConn();
23
24       // Verificar primero si el comentario existe
25       $checkSql = "SELECT * FROM comentarios_usuarios WHERE id = ?";
26       $checkStmt = $conn->prepare(query: $checkSql);
27       $checkStmt->bind_param(types: "i", var: &$id);
28       $checkStmt->execute();
29       $checkResult = $checkStmt->get_result();
30
31       if ($checkResult->num_rows === 0) {
32         $checkStmt->close();
33         $conn->close();
34         http_response_code(response_code: 404);
35         echo json_encode(value: ['error' => 'Comentario no encontrado']);
36         return;
37       }
38
39       $checkStmt->close();
40
41       // Construir la consulta SQL dinámica para actualizar solo los campos proporcionados
42       $updateFields = [];
43       $params = [];
44       $types = "";
45
46       if (isset($data['texto'])) {
47         $updateFields[] = "texto = ?";
48         $params[] = $data['texto'];
49         $types .= "s";
50       }
51
52       if (isset($data['id_mensaje'])) {
53         $updateFields[] = "id_mensaje = ?";
54         $params[] = $data['id_mensaje'];
55         $types .= "i";
56       }
57
58       if (isset($data['id_usuario'])) {
59         $updateFields[] = "id_usuario = ?";
60         $params[] = $data['id_usuario'];
61         $types .= "i";
62       }
63
64       if (isset($data['fecha'])) {
65         $updateFields[] = "fecha = ?";
66         $params[] = $data['fecha'];
67         $types .= "s";
68       }
69
70       if (isset($data['recurso'])) {
71         $updateFields[] = "recurso = ?";
72         $params[] = $data['recurso'];
73         $types .= "s";
74       }
75
76       // Si no hay campos para actualizar, salir
77       if (empty($updateFields)) {
78         http_response_code(response_code: 400);
79         echo json_encode(value: ['error' => 'No se proporcionaron campos para actualizar']);
80         return;
81       }
82
83       // Construir la consulta SQL completa
84       $sql = "UPDATE comentarios_usuarios SET " . implode(separator: ", ", array: $updateFields) . " WHERE id = ?";
85
86       // Añadir el ID al final de los parámetros
87       $params[] = $id;
88       $types .= "i";
89
90       // Preparar y ejecutar la consulta
91       $stmt = $conn->prepare(query: $sql);
92       $stmt->bind_param(types: $types, ...var: &$params);
93
94       if ($stmt->execute()) {
95         $stmt->close();
96         $conn->close();
97
98         http_response_code(response_code: 200);
99         echo json_encode(value: ['message' => 'Comentario actualizado con éxito']);
100      } else {
101        throw new Exception(message: "Error al actualizar el comentario: " . $stmt->error);
102      }
103
104    } catch (Exception $e) {
105      http_response_code(response_code: 500);
106      echo json_encode(value: ['error' => $e->getMessage()]);
107    }
108  }

```

El fichero **UpdateCommentsUser.php** define la clase **UpdateCommentsUser**, cuyo objetivo es actualizar los datos de un comentario existente en la base de datos a partir de su identificador.

El método principal **execute(\$id, \$data)** recibe como argumentos el ID del comentario y un array asociativo con los nuevos valores a actualizar.

Primero, se establece una conexión a la base de datos mediante una instancia de la clase DB. Seguidamente, se comprueba si el comentario con el ID proporcionado existe mediante una consulta **SELECT**. Si no se encuentra, se devuelve un error HTTP 404 con un mensaje en formato JSON y se detiene la ejecución.

Si el comentario existe, se construye dinámicamente una consulta SQL de tipo **UPDATE** en función de los campos presentes en el array **\$data**. Solo se incluyen en la actualización aquellos campos que han sido definidos, como texto, **id\_mensaje**, **id\_usuario**, **fecha** o **recurso**. Los valores correspondientes se añaden a un array de parámetros (**\$params**) y se especifican sus tipos (**\$types**) para la función **bind\_param**.

Si no se ha proporcionado ningún campo para actualizar, se devuelve un error HTTP 400 con un mensaje correspondiente.

Una vez construida la sentencia **UPDATE**, se ejecuta con los parámetros adecuados. Si la ejecución es exitosa, se cierra la conexión y se devuelve una respuesta HTTP 200 con un mensaje de éxito. En caso contrario, se lanza una excepción y se informa del error con un código HTTP 500.

Este fichero permite realizar actualizaciones parciales sobre los comentarios, garantizando la integridad mediante validación previa y evitando consultas innecesarias.

```
dekerd / follow_users / use_cases / ↵ GetFollowedUsers.php / PHP Intelephense / ↵ GetFollowedUsers / ↵ execute / ↵ Execute
1  <?php
2
3  require_once __DIR__ . '/../../config/database.php';
4
5  class GetFollowedUsers {
6      2 references | 0 implementations
7      2 references | 0 overrides
8      public function execute($id_usuario): array {
9          $db = new DB();
10         $conn = $db->getConn();
11
12         // Obtener todos los datos de las personas que sigue un usuario (id_usu1 es quien
13         $stmt = $conn->prepare(query: "
14             SELECT u.*
15             FROM users u
16             INNER JOIN seguidores_usuarios f ON u.id = f.id_usu2
17             WHERE f.id_usu1 = ?
18         ");
19
20         if (!$stmt) {
21             http_response_code(response_code: 500);
22             echo json_encode(value: ["error" => "Error al preparar la consulta"]);
23             return [];
24         }
25
26         $stmt->bind_param(types: "i", var: &$id_usuario);
27         $stmt->execute();
28         $result = $stmt->get_result();
29
30         $users = [];
31         while ($user = $result->fetch_assoc()) {
32             $users[] = $user;
33         }
34
35     }
36 }
37
```

Este fichero define la clase **GetFollowedUsers**, cuyo propósito es obtener una lista de usuarios que son seguidos por un usuario concreto, identificado por su ID.

El método **execute(\$id\_usuario)** realiza una consulta SQL que une las tablas **usuarios** y **seguidores\_usuarios**, filtrando por **id\_usu1** (usuario que sigue) para obtener los datos completos (**SELECT u.\***) de los usuarios seguidos (**id\_usu2**).

Los resultados se devuelven en forma de array asociativo. En caso de error al preparar la consulta, se devuelve una respuesta HTTP 500 y un array vacío.

```
<?php

require_once __DIR__ . '/../../config/database.php';

2 references | 0 implementations
class FollowUser {
    2 references | 0 overrides
    public function execute($id_usu1, $id_usu2): bool {
        try {
            $db = new DB();
            $conn = $db->getConn();

            // Verificar si ya existe la relación
            $checkStmt = $conn->prepare(query: "SELECT id FROM seguidores_usuarios WHERE id_usu1 = ? AND id_usu2 = ?");
            if (!$checkStmt) {
                return false; // Error al preparar la consulta
            }

            $checkStmt->bind_param(types: "ii", var: &$id_usu1, vars: &$id_usu2);
            $checkStmt->execute();
            $checkResult = $checkStmt->get_result();

            if ($checkResult->num_rows > 0) {
                $checkStmt->close();
                // Ya existe la relación
                return false;
            }
            $checkStmt->close();

            // Insertar la relación
            $stmt = $conn->prepare(query: "INSERT INTO seguidores_usuarios (id_usu1, id_usu2, tiempo) VALUES (?, ?, CURDATE())");
            if (!$stmt) {
                return false; // Error al preparar la consulta
            }

            $stmt->bind_param(types: "ii", var: &$id_usu1, vars: &$id_usu2);
            $result = $stmt->execute();
            $stmt->close();

            return $result;
        } catch (Exception $e) {
            // Registra el error pero devuelve false
            error_log(message: "Error en FollowUser: " . $e->getMessage());
            return false;
        }
    }
}
```

Este fichero contiene la clase **FollowUser**, responsable de crear una relación de seguimiento entre dos usuarios.

El método **execute(\$id\_usu1, \$id\_usu2)** primero verifica si ya existe una relación en la tabla **seguidores\_usuarios** entre los dos IDs proporcionados.

Si no existe, inserta una nueva fila indicando que el usuario **\$id\_usu1** está siguiendo al usuario **\$id\_usu2**, registrando también la fecha actual (**CURDATE()**).

Devuelve **true** si la operación es exitosa y **false** si falla o si la relación ya existe.

```
<?php

require_once __DIR__ . '/../../config/database.php';

2 references | 0 implementations
class CountFollowers {
    2 references | 0 overrides
    public function execute($id_usuario): mixed {
        $db = new DB();
        $conn = $db->getConn();

        // Contar el número de seguidores de un usuario (usuarios que le siguen a él)
        $stmt = $conn->prepare(query: "SELECT COUNT(*) as total FROM seguidores_usuarios
        WHERE id_usu2 = ?");
        if (!$stmt) {
            http_response_code(response_code: 500);
            echo json_encode(value: ["error" => "Error al preparar la consulta"]);
            return 0;
        }

        $stmt->bind_param(types: "i", var: &$id_usuario);
        $stmt->execute();
        $result = $stmt->get_result();
        $data = $result->fetch_assoc();
        $stmt->close();

        return $data['total'];
    }
}
```

Este fichero define la clase **CountFollowers**, que tiene como objetivo contar cuántos seguidores tiene un determinado usuario.

El método **execute(\$id\_usuario)** realiza una consulta SQL **SELECT COUNT(\*)** sobre la tabla **seguidores\_usuarios**, donde **id\_usu2** coincide con el ID del usuario objetivo.

El resultado devuelto es un número entero correspondiente al total de seguidores.

Si hay un error al preparar la consulta, se responde con un error HTTP 500 y se devuelve el valor **0**.



```
<?php

require_once __DIR__ . '/../../config/database.php';

class UnfollowUser {
    public function execute($id_usu1, $id_usu2): bool {
        try {
            $db = new DB();
            $conn = $db->getConn();

            // Eliminar la relación de seguimiento entre usuarios
            $stmt = $conn->prepare(query: "DELETE FROM seguidores_usuarios WHERE id_usu1 = ? AND id_usu2 = ?");
            if (!$stmt) {
                return false; // Error al preparar la consulta
            }

            $stmt->bind_param(types: "ii", var: &$id_usu1, vars: &$id_usu2);
            $result = $stmt->execute();
            $stmt->close();

            return $result;
        } catch (Exception $e) {
            // Registra el error pero devuelve false
            error_log(message: "Error en UnfollowUser: " . $e->getMessage());
            return false;
        }
    }
}
```

Este fichero implementa la clase **UnfollowUser**, encargada de eliminar una relación de seguimiento entre dos usuarios.

El método **execute(\$id\_usu1, \$id\_usu2)** prepara y ejecuta una consulta **DELETE** sobre la tabla **seguidores\_usuarios** para eliminar la fila que representa que **\$id\_usu1** sigue a **\$id\_usu2**.

Devuelve **true** si la eliminación se realiza correctamente y **false** en caso de fallo. También registra errores en el log mediante **error\_log**.

```

1  <?php
2
3  /**
4   * Controlador para gestionar operaciones relacionadas con el panel de administración
5   *
6   * Este controlador utiliza el parámetro 'action' en la URL para determinar qué acción ejecutar.
7   * Ejemplo: AdminController.php?action=projectStats
8   */
9
10 require_once __DIR__ . '/../config/database.php';
11 require_once __DIR__ . '/use_cases/GetProjectsStats.php';
12 require_once __DIR__ . '/use_cases/GetMessagesStats.php';
13 require_once __DIR__ . '/use_cases/GetUsersStats.php';
14
15 // Obtener método HTTP
16 $method = $_SERVER['REQUEST_METHOD'];
17
18 // Configuración de CORS para desarrollo
19 header(header: "Access-Control-Allow-Origin: http://localhost:5173");
20 header(header: "Access-Control-Allow-Credentials: true");
21 header(header: "Access-Control-Allow-Methods: GET, POST, OPTIONS");
22 header(header: "Access-Control-Allow-Headers: Content-Type, Authorization");
23
24 // Manejar solicitudes preflight OPTIONS
25 if ($method === 'OPTIONS') {
26     http_response_code(response_code: 200);
27     exit;
28 }
29
30 // Para solicitudes normales, establecer el tipo de contenido JSON
31 header(header: "Content-Type: application/json; charset=UTF-8");
32
33 // Verificar que se haya proporcionado una acción
34 if (!isset($_GET['action'])) {
35     http_response_code(response_code: 400);
36     echo json_encode(value: ['error' => 'No se ha especificado ninguna acción']);
37     exit;
38 }
39
40 $action = $_GET['action'];
41
42 // Mapeo de acciones a métodos
43 switch ($action) {
44     case 'projectStats':
45         if ($method === 'GET') {
46             $useCase = new GetProjectsStats();
47             $useCase->execute();
48         } else {
49             http_response_code(response_code: 405);
50             echo json_encode(value: ['error' => 'Método no permitido']);
51         }
52         break;
53
54     case 'messageStats':
55         if ($method === 'GET') {
56             $useCase = new GetMessagesStats();
57             $useCase->execute();
58         } else {
59             http_response_code(response_code: 405);
60             echo json_encode(value: ['error' => 'Método no permitido']);
61         }
62         break;
63
64     case 'userStats':
65         if ($method === 'GET') {
66             $useCase = new GetUsersStats();
67             $useCase->execute();
68         } else {
69             http_response_code(response_code: 405);
70             echo json_encode(value: ['error' => 'Método no permitido']);
71         }
72         break;
73
74 // En caso de querer obtener todas las estadísticas en una sola llamada
75 case 'dashboardStats':
76     if ($method === 'GET') {
77         try {
78             // Obtener estadísticas de proyectos
79             $projectsUseCase = new GetProjectsStats();
80             $projectsUseCase->execute();
81             $projectsStats = json_decode(json: ob_get_clean(), associative: true);
82
83             // Obtener estadísticas de mensajes
84             $messagesUseCase = new GetMessagesStats();
85             $messagesUseCase->execute();
86             $messagesStats = json_decode(json: ob_get_clean(), associative: true);
87
88             // Obtener estadísticas de usuarios
89             $usersUseCase = new GetUsersStats();
90             $usersUseCase->execute();
91             $usersStats = json_decode(json: ob_get_clean(), associative: true);
92
93             // Combinar todas las estadísticas
94             $dashboardStats = [
95                 'projects' => $projectsStats,
96                 'messages' => $messagesStats,
97                 'users' => $usersStats
98             ];
99
100             header(header: 'Content-Type: application/json');
101             echo json_encode(value: $dashboardStats);
102         } catch (Exception $e) {
103             http_response_code(response_code: 500);
104             echo json_encode(value: ['error' => 'Error al obtener estadísticas del dashboard: ' .
105             $e->getMessage()]);
106         }
107     } else {
108         http_response_code(response_code: 405);
109         echo json_encode(value: ['error' => 'Método no permitido']);
110     }
111     break;
112
113 default:
114     http_response_code(response_code: 404);
115     echo json_encode(value: ['error' => 'Acción no encontrada: ' . $action]);
116     break;
117 }
118
119 }

```

## Fichero: **MessagesUserController.php**

Este fichero actúa como controlador principal para la gestión de mensajes de usuario. Su función es dirigir las peticiones entrantes HTTP hacia los casos de uso correspondientes, permitiendo operaciones de creación, lectura, actualización y eliminación (CRUD) sobre mensajes de usuario.

El script comienza importando las dependencias necesarias desde el directorio **use\_cases**, donde se definen las clases que implementan la lógica de negocio: **CreateMessagesUser**, **GetAllMessagesUser**, **GetMessagesUserById**, **UpdateMessagesUser** y **DeleteMessagesUser**.

Se maneja el método HTTP de la solicitud (**GET**, **POST**, **PUT**, **DELETE**, **OPTIONS**) y se configuran los encabezados CORS para permitir el acceso desde el frontend en desarrollo. También se gestiona la respuesta automática a solicitudes **OPTIONS** (preflight) con un código 200.

A continuación, el controlador verifica que se haya especificado una acción a través del parámetro **action** en la URL, y ejecuta el bloque correspondiente dentro de un **switch**:

- **create (POST)**: Permite crear un nuevo mensaje. Valida los datos recibidos (texto, id\_usuario y recurso), añade la fecha actual si no se proporciona, y llama al caso de uso **CreateMessagesUser**.
- **all (GET)**: Recupera todos los mensajes paginados, con opción de búsqueda mediante el parámetro **search**. Llama al caso de uso **GetAllMessagesUser** y devuelve un JSON con los mensajes y metadatos de paginación.
- **get (GET)**: Devuelve un mensaje concreto por su ID, utilizando la clase **GetMessagesUserById**.
- **update (PUT o POST)**: Permite actualizar un mensaje existente. Requiere el ID del mensaje como parámetro y un cuerpo JSON con los campos a modificar. Usa **UpdateMessagesUser**.
- **delete (DELETE o POST)**: Elimina un mensaje identificado por su ID, mediante el caso de uso **DeleteMessagesUser**.

En todos los casos, se devuelven respuestas JSON con códigos HTTP apropiados (**200**, **201**, **400**, **404**, **500**) y mensajes de error cuando corresponde. Además, se hace uso de **error\_log** para registrar incidencias en tiempo de ejecución.

Este controlador centraliza el acceso a la funcionalidad relacionada con los mensajes de los usuarios, ofreciendo una interfaz REST sencilla y extensible, adecuada para su consumo desde aplicaciones frontend modernas.

A continuación, te presento el contenido que has facilitado estructurado y maquetado de forma adecuada para su inclusión en un documento Word, como podría ser un TFG o manual de usuario. Está organizado por títulos y subtítulos jerarquizados y con viñetas y numeraciones cuando corresponde.

Aquí tienes la sección sobre **APIs utilizadas en CompoDev**, estructurada y redactada formalmente para su inclusión en un documento Word (TFG, manual técnico, documentación de desarrollo, etc.). La redacción es clara, técnica y organizada con títulos jerárquicos y viñetas:

---

## EXPLICACIÓN DE LAS APIs UTILIZADAS EN COMPODEV

### 1 APIs Externas

#### 1.1 Google Generative AI (Gemini)

##### Descripción

La API de Google Generative AI, concretamente el modelo Gemini, permite interactuar con modelos de lenguaje generativo desarrollados por Google. En CompoDev, esta API se utiliza para generar fragmentos de código (HTML, CSS y JavaScript) a partir de descripciones textuales introducidas por el usuario.

Implementación en el proyecto:

- Se integra a través del componente **AICodeGenerator.jsx**.
- Utiliza el modelo **gemini-1.5-flash**, especializado en respuestas rápidas.
- El prompt se construye a partir de instrucciones y texto del usuario.
- Se espera como salida un objeto JSON con tres claves: **html**, **css** y **js**.
- La clave de API se almacena en variables de entorno: **VITE\_GOOGLE\_API\_KEY**.

Ejemplo de uso:

```
const genAI = new GoogleGenerativeAI(apiKey);

const model = genAI.getGenerativeModel({ model: 'gemini-1.5-flash' });

const finalPrompt = `${instrucciones}${prompt}`;

const result = await model.generateContent(finalPrompt);
```

## Ventajas:

- Permite generar código funcional a partir de lenguaje natural.
- Acelera el desarrollo proporcionando componentes base reutilizables.
- La estructura JSON facilita la integración con el editor de código.

---

## 2 APIs Internas (Backend)

### 2.1 API de Proyectos

#### Descripción

Es una API REST desarrollada en PHP que gestiona todas las operaciones relacionadas con los proyectos de código creados por los usuarios.

#### Principales endpoints:

- **GET /projects/ProjectController.php?action=getProjectCode&id={id}**  
→ Obtiene el código fuente (HTML, CSS, JS) de un proyecto específico.
- **PUT /projects/ProjectController.php?action=updateCode&id={id}**  
→ Actualiza el código de un proyecto existente.
- **GET /projects/ProjectController.php?action=checkOwner&id={id}&id\_usuario={userId}**  
→ Verifica si un usuario es el propietario del proyecto.
- **GET /projects/ProjectController.php?action=all**  
→ Lista todos los proyectos, con opciones de paginación y búsqueda.

- **GET /projects/ProjectController.php?action=getByIdUser&id={userId}**  
→ Obtiene todos los proyectos creados por un usuario específico.

#### **Arquitectura:**

- Se implementa un enfoque modular basado en el patrón de Clean Architecture.
  - Cada operación está encapsulada en su propia clase de "caso de uso".
  - Las respuestas siguen el estándar REST, en formato JSON.
- 

## **2.2 API de Usuarios**

#### **Descripción**

API destinada a la autenticación, gestión de sesiones y administración de datos de usuario.

#### **Principales endpoints:**

- **GET /users/UserController.php?action=session**  
→ Valida si el usuario tiene una sesión activa.
- Además, incluye endpoints para:
  - Registro de nuevos usuarios
  - Inicio y cierre de sesión
  - Actualización de perfil y contraseña
  - Verificación y eliminación de cuentas

---

## **3 Integración entre Frontend y Backend**

La comunicación entre el frontend (React) y el backend (PHP) se realiza mediante la librería Axios como cliente HTTP.

#### **Características:**

- Se utilizan variables de entorno (**VITE\_BACKEND\_URL**, **VITE\_GOOGLE\_API\_KEY**) para configurar rutas y claves de forma dinámica.
- Las peticiones incluyen credenciales (**withCredentials: true**) para mantener la sesión del usuario.

#### Ejemplo de integración:

```
const loadProjectData = async () => {  
  const response = await axios.get(  
    `${backendUrl}/projects/ProjectController.php?action=getProjectCode&id=${idParam.id}`,  
    { withCredentials: true }  
  );  
  // Procesamiento de datos  
};
```

---

## 4 Consideraciones de Seguridad

- Gestión de claves API:  
Las claves de servicios externos (como Gemini) se almacenan en variables de entorno, evitando su exposición directa en el código fuente del frontend.
- CORS:  
Se configura el middleware de CORS para restringir qué dominios pueden realizar peticiones al backend.
- Control de sesiones:  
Antes de ejecutar acciones sensibles (como modificar código de un proyecto), se valida si el usuario tiene una sesión activa y si es propietario del recurso.

---

## 5 Conclusión técnica

El diseño de APIs en CompoDev permite una arquitectura modular, escalable y segura, que facilita tanto el mantenimiento como la incorporación de nuevas funcionalidades. Esta separación clara entre frontend, backend y servicios externos garantiza una buena práctica de desarrollo y una experiencia de usuario fluida.

## 8. MANUAL DE USUARIO

### 1. PRIMEROS PASOS

#### 1.1 Registro de cuenta

Para registrarse en CompoDev, sigue estos pasos:

1. Accede a la página principal de **CompoDev**.
2. Haz clic en el botón "**Registrarse**" ubicado en la esquina superior derecha.
3. Completa el formulario de registro con los siguientes campos:
  - Nombre de usuario (único en la plataforma)
  - Dirección de correo electrónico válida
  - Contraseña (mínimo 8 caracteres, al menos un número y un carácter especial)
  - Confirmación de contraseña
4. Opcionalmente, puedes añadir:
  - Nombre completo
  - Biografía
  - Enlaces a tus perfiles en otras plataformas
5. Acepta los términos y condiciones marcando la casilla correspondiente.

- 
6. Haz clic en "**Crear cuenta**".
  7. Recibirás un correo electrónico de verificación. Haz clic en el enlace para activar tu cuenta.
- 

## 1.2 Inicio de sesión

1. Accede a la página principal de CompoDev.
  2. Haz clic en "**Iniciar sesión**" en la esquina superior derecha.
  3. Introduce tu nombre de usuario o correo electrónico.
  4. Introduce tu contraseña.
  5. (Opcional) Marca la casilla "**Recordarme**" si deseas mantener la sesión iniciada.
  6. Haz clic en "**Entrar**".
- 

## 1.3 Recuperación de contraseña

1. En la pantalla de inicio de sesión, haz clic en "**¿Olvidaste tu contraseña?**".
  2. Introduce el correo electrónico asociado a tu cuenta.
  3. Haz clic en "**Enviar enlace de recuperación**".
  4. Recibirás un correo con un enlace para restablecer tu contraseña.
  5. Accede al enlace e introduce la nueva contraseña (dos veces).
  6. Haz clic en "**Guardar nueva contraseña**".
- 

## 1.4 Navegación por la interfaz

La interfaz de CompoDev está compuesta por varias secciones:

- **Barra de navegación superior:** Acceso a Inicio, Búsqueda, Crear componente, Mensajes y Perfil.
- **Página de inicio:** Muestra componentes destacados, usuarios populares y últimas novedades.
- **Panel lateral** (tras iniciar sesión): Accesos rápidos a tus componentes, mensajes y notificaciones.
- **Pie de página:** Contiene enlaces a información legal, contacto y ayuda.

La interfaz es responsive y se adapta a distintos dispositivos (ordenadores, móviles, tablets).

---

## 2. TU PERFIL DE USUARIO

### 2.1 Visualización de perfil

Para acceder a tu perfil:

1. Inicia sesión.
2. Haz clic en tu nombre de usuario o avatar (esquina superior derecha).
3. Selecciona "Mi perfil".

En tu perfil puedes consultar:

- Información personal: nombre de usuario, nombre completo, biografía.
  - Estadísticas: número de componentes publicados, seguidores, seguidos.
  - Lista de componentes publicados.
  - Actividad reciente.
-

## **2.2 Edición de información personal**

1. Accede a tu perfil.
  2. Haz clic en "**Editar perfil**".
  3. Modifica los campos que deseas:
    - Nombre completo
    - Biografía (hasta 500 caracteres)
    - Enlaces a redes sociales
    - Correo electrónico (requiere verificación)
    - Contraseña (requiere la contraseña actual)
  4. Haz clic en "**Guardar cambios**".
- 

## **2.3 Gestión de avatar**

1. En tu perfil, pasa el cursor sobre tu imagen.
  2. Haz clic en el icono de cámara.
  3. Selecciona una nueva imagen (formatos: JPG, PNG, GIF; máximo 2MB).
  4. Ajusta el recorte si es necesario.
  5. Haz clic en "**Guardar avatar**".
- 

## **2.4 Estadísticas de seguidores**

En el perfil puedes consultar:

- **Seguidores:** personas que te siguen.
- **Seguidos:** personas a las que sigues.

Para cada usuario listado puedes:

- Acceder a su perfil.
  - Seguir o dejar de seguir.
  - Enviar un mensaje directo.
- 

## 3. GESTIÓN DE COMPONENTES

### 3.1 Crear un nuevo componente

1. Haz clic en "**+ Crear componente**" en la barra de navegación.
  2. Completa el formulario con:
    - Título (obligatorio)
    - Categoría (obligatoria)
    - Descripción (obligatoria)
    - Etiquetas
    - Código (obligatorio)
    - Imagen de vista previa (opcional)
  3. Haz clic en "**Vista previa**".
  4. Haz clic en "**Publicar componente**".
-

### **3.2 Editar componentes existentes**

1. Ve a tu perfil o a la página del componente.
  2. Haz clic en el ícono **Editar** (lápiz).
  3. Modifica los campos deseados.
  4. Haz clic en "**Guardar cambios**".
- 

### **3.3 Eliminar componentes**

1. Ve a tu perfil o a la página del componente.
2. Haz clic en el ícono **Eliminar** (papelera).
3. Confirma en el cuadro de diálogo haciendo clic en "**Eliminar**".

**Nota:** Esta acción es permanente e irreversible. Los comentarios asociados también se eliminarán.

---

### **3.4 Visualizar componentes**

Haz clic en el título o imagen de un componente para ver su detalle. En la página verás:

- Información completa
- Código fuente con resaltado
- Vista previa (si aplica)
- Estadísticas
- Comentarios

Puedes copiar el código con el botón "**Copiar código**".

---

### **3.5 Categorización de componentes**

Los componentes están organizados en las siguientes categorías:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** PHP, Node.js, etc.
- **Utilidades:** Funciones generales
- **Animaciones:** Efectos visuales
- **Layouts:** Estructuras de diseño
- **Formularios:** Validaciones, entradas
- **Gráficos:** Visualización de datos
- **API:** Conectores con APIs externas
- **Seguridad:** Autenticación, protección
- **Otros:** Contenidos no clasificados en las anteriores

A continuación tienes el contenido que has proporcionado, organizado de forma clara y estructurada para ser incluido en un documento Word. Está maquetado con encabezados jerárquicos, listas y viñetas, listo para copiar y pegar directamente en Microsoft Word, Google Docs u otro editor similar.

---

## **4. BÚSQUEDA Y EXPLORACIÓN**

### **4.1 Buscar usuarios**

Para encontrar usuarios específicos:

1. Haz clic en "**Búsqueda**" en la barra de navegación superior.
2. Selecciona la pestaña "**Usuarios**".

3. Introduce el nombre de usuario o términos relacionados en el campo de búsqueda.
4. Opcionalmente, puedes ordenar los resultados por:
  - Popularidad (número de seguidores)
  - Fecha de registro (más reciente / más antiguo)
  - Actividad reciente
5. Haz clic en "**Buscar**".

Los resultados mostrarán tarjetas con información básica:

- Nombre y avatar
- Número de seguidores
- Componentes publicados

Desde cada tarjeta puedes:

- Ver el perfil completo
- Seguir o dejar de seguir
- Enviar un mensaje directo

---

## 4.2 Filtros avanzados

**Para componentes:**

- **Lenguaje de programación:** JavaScript, PHP, Python, etc.
- **Fecha de publicación:** Últimas 24h, semana, mes o año.
- **Popularidad:** Más visitados, comentados o descargados.

- **Estado:** Estable, beta, en desarrollo.
- **Licencia:** MIT, GPL, Apache, etc.

**Para usuarios:**

- Número mínimo de componentes publicados
  - Fecha de registro
  - Especialidad (Frontend, Backend, Fullstack)
- 

### 4.3 Ordenación de resultados

**Para componentes:**

- Más recientes
- Más populares
- Mejor valorados
- Alfabéticamente (A-Z / Z-A)

**Para usuarios:**

- Más seguidores
  - Más componentes
  - Fecha de registro
  - Actividad reciente
- 

## 5. INTERACCIÓN SOCIAL

## **5.1 Seguir a otros usuarios**

**Para seguir:**

1. Accede al perfil del usuario o encuéntralo en una búsqueda.
2. Haz clic en el botón "**Seguir**".

El botón cambiará a "**Siguiendo**" y verás su actividad en tu feed.

**Para dejar de seguir:**

1. Accede a su perfil o a la lista de seguidos.
  2. Haz clic en "**Siguiendo**".
  3. Confirma en el cuadro de diálogo.
- 

## **5.2 Ver perfiles de otros usuarios**

Para ver un perfil:

- Haz clic en el nombre o avatar del usuario.

Desde su perfil puedes:

- Ver su biografía e información pública
- Consultar estadísticas (componentes, seguidores)
- Acceder a su lista de componentes
- Ver su actividad reciente
- Seguir o dejar de seguir
- Enviar mensajes directos

- Compartir su perfil
- 

### **5.3 Sistema de mensajería**

#### **Para enviar un mensaje:**

1. Visita el perfil del destinatario.
2. Haz clic en "**Enviar mensaje**".
3. Escribe el mensaje y (opcional) adjunta código o enlaces.
4. Haz clic en "**Enviar**".

#### **Para acceder a tus mensajes:**

1. Haz clic en el ícono de mensajes en la barra superior.
  2. Consulta las conversaciones ordenadas por actividad.
  3. Haz clic para ver y responder.
- 

### **5.4 Comentarios en componentes**

#### **Para comentar:**

1. Ve a la página del componente.
2. Dirígete a la sección de comentarios.
3. Escribe tu comentario (puedes usar Markdown).
4. Haz clic en "**Publicar comentario**".

#### **Para responder:**

1. Haz clic en "**Responder**" debajo del comentario.
  2. Escribe tu respuesta y haz clic en "**Publicar respuesta**".
- 

## 6. FUNCIONES AVANZADAS

### 6.1 Editor de código integrado

El editor de CompoDev incluye:

- Resaltado de sintaxis
- Autocompletado inteligente
- Temas (claro, oscuro, cyberpunk)
- Numeración de líneas
- Plegado de secciones
- Vista previa en tiempo real (para frontend)

**Para usarlo:**

- Al crear o editar un componente, el editor se abre automáticamente.
  - Usa la barra superior para configurar el lenguaje o el tema.
  - Escribe tu código en el área principal.
  - Haz clic en "**Vista previa**" para ver el resultado.
- 

### 6.2 Vista previa de componentes

Para componentes con interfaz visual:

1. Accede a la página del componente.
  2. Haz clic en la pestaña "**Vista previa**".
  3. Interactúa con el componente en tiempo real.
  4. Si hay opciones configurables, usa el panel lateral.
- 

### 6.3 Compartir en redes sociales

1. En la página del componente o perfil, haz clic en "**Compartir**".
  2. Selecciona una red social (Twitter, LinkedIn, etc.).
  3. Personaliza el mensaje y publica.
  4. También puedes copiar el enlace directo.
- 

### 6.4 Estadísticas de uso

Para ver estadísticas:

1. Accede a tu perfil.
2. Haz clic en la pestaña "**Estadísticas**".

Podrás consultar:

- Visitas
- Copias/descargas del código
- Comentarios recibidos
- Guardados por otros usuarios

- Tendencias (diarias, semanales, mensuales)
- 

## 7. PANEL DE ADMINISTRACIÓN

### 7.1 Acceso al panel

Solo disponible para administradores:

1. Inicia sesión como administrador.
  2. Haz clic en tu nombre/avatar.
  3. Selecciona "**Panel de administración**".
- 

### 7.2 Gestión de usuarios

- Ver todos los usuarios (con filtros)
  - Editar perfiles
  - Asignar roles (usuario, moderador, admin)
  - Suspender cuentas temporalmente
  - Eliminar cuentas permanentemente
- 

### 7.3 Gestión de componentes

- Listado con filtros avanzados
- Aprobar o rechazar componentes
- Editar código e información

- Administrar categorías
  - Marcar como destacados
- 

## 7.4 Estadísticas del sistema

Indicadores disponibles:

- Total de usuarios, nuevos registros, activos
  - Total de componentes, nuevas publicaciones
  - Interacciones (comentarios, mensajes, seguimientos)
  - Rendimiento (carga, recursos del servidor)
  - Gráficos y filtros temporales
- 

# 8. SOLUCIÓN DE PROBLEMAS

## 8.1 Problemas comunes y soluciones

### Inicio de sesión

- Revisa si activaste la cuenta.
- Prueba restablecer contraseña.
- Comprueba bloqueo de mayúsculas.
- Borra cookies del navegador.

### Correo de recuperación no recibido

- Revisa la carpeta de spam.

- Asegúrate del correo correcto.
- Contacta con soporte.

### **Publicación de componentes**

- Verifica que todos los campos estén completos.
- Asegura que el código no excede el límite.
- Revisa tu conexión.

### **Visualización del código**

- Verifica sintaxis y formato.
- Selecciona el lenguaje correcto.
- Limpia el formato copiado.

### **Problemas técnicos generales**

- Limpia caché, prueba otro navegador.
- Cierra pestañas que consuman ancho de banda.
- Revisa actualizaciones del navegador.

---

## **8.2 Contacto con soporte**

1. Haz clic en "**Ayuda**" en el pie de página.
2. Selecciona "**Contactar con soporte**".
3. Completa el formulario con:
  - Descripción del problema

- Pasos que seguiste
- Capturas (si aplica)
- Tu email de contacto

4. Haz clic en "**Enviar**".

El equipo de soporte responderá en un plazo de hasta 48 horas laborables.



.env files

```
VITE_GOOGLE_API_KEY="AIzaSyAAreASQyE7bXQLh-MGeKZQmkw3RGUJdmQ"

VITE_INSTRUCCIONES = 'Eres un asistente que genera código. Devuelve
únicamente un objeto JSON con exactamente estas tres claves: html, css y
js.

No incluyas ningún otro texto adicional. No utilices saltos de línea (\n),
ni escapes adicionales en el resultado.'
```

NO encierres tu respuesta dentro de comillas triples ``` ni dentro de bloques de código markdown.

Quiero directamente y únicamente el objeto JSON de esta forma exacta:

```
{"html":"aquí el código HTML","css":"aquí el código CSS","js":"aquí el código JS"}
```

Lo que necesito:

```
VITE_BACKEND_URL="http://localhost/compodev2/backend"
```

TODAS LAS CONTRASEÑAS DE LA BASE DE DATOS SON:

**Alejandro2002**