

DOCUMENTACION “MENTES FUERTES, REDES CONSCIENTES”

Erik Mora

UTA 02/07/2025

Contenido

Objetivo General.....	3
1. Caso de Estudio Implementado	3
✓ Requerimientos Funcionales.....	3
🔒 Requerimientos No Funcionales.....	4
🎯 Valor Agregado.....	4
2. Planificación Ágil.....	5
⌚ Metodología Aplicada: Scrum	5
📅 Ciclo de Vida del Proyecto	5
📋 Product Backlog (Historias de Usuario)	6
🏃 Detalle de Sprints	6
🛠 Herramientas utilizadas para planificación	7
3. Arquitectura y Patrones de Diseño.....	8
🧱 Arquitectura Seleccionada: MVC Modular Client-Side	8
✳️ Componentes del Modelo MVC:	8
🧠 Patrones de Diseño Aplicados.....	9
2. .Singleton	9
3. 🔔 Observer (Publicador/Suscriptor).....	9
4. 🏭 Factory	9
⚙️ Ventajas de la Arquitectura y Patrones Aplicados	10
4. Frameworks y Herramientas Utilizadas.....	10
🌐 Frontend: HTML5, CSS3, JavaScript (Vanilla).....	10
🎨 Estilos y Recursos Visuales	11
⌚ Control de Versiones: Git + GitHub	11
5. Mecanismos de Seguridad Integrados	12
1 Autenticación de Administrador mediante Contraseña.....	12
2 Protección contra Ataques XSS (Cross-Site Scripting) en el Chat.....	13
3 Control de Roles y Permisos.....	14

4	Validación de Entradas en Formularios.....	14
5	Advertencia Visual de Seguridad en el Chat.....	15
6.	Integración de Servicios Web	16
1	Chat de Apoyo Emocional Simulado (Servicio Tipo Conversacional)	16
2	CRUD Dinámico de Publicaciones Motivacionales (Servicio Local).....	17
🌐	Servicios Web de Terceros.....	17
1	Redes Sociales: Instagram y TikTok.....	17
2	Google Fonts	18
3	FontAwesome	18
⌚	Consideraciones para Escalabilidad.....	19
7.	Pruebas y Validación	19
💡	1. Pruebas Funcionales	20
📲	2. Pruebas de Interfaz y Responsividad.....	20
🛡️	3. Pruebas de Seguridad (XSS y Control de Acceso)	21
🔍	4. Pruebas de Flujo Completo (Publicación y Administración)	22
8.	Repositorio y Entrega	23
9.	Diagramas Visuales	23

“Mentes Fuertes, Redes Conscientes”

Objetivo General

Desarrollar una aplicación web funcional enfocada en el bienestar emocional y el desarrollo personal de jóvenes, incorporando principios de la metodología ágil, arquitectura de software moderna, patrones de diseño, mecanismos de seguridad, servicios web propios y de terceros, así como prácticas de pruebas y validación.

1. Caso de Estudio Implementado

“Mentes Fuertes, Redes Conscientes” es una plataforma web creada para atender una necesidad social actual: el fortalecimiento del bienestar emocional y el desarrollo personal de los jóvenes en un entorno digital seguro. Esta necesidad surge ante el incremento de problemáticas como la ansiedad, el aislamiento y el consumo de contenido tóxico en redes sociales.

El sitio se plantea como una **respuesta tecnológica y empática**, ofreciendo un espacio accesible que promueve el crecimiento personal mediante contenido motivador, guías prácticas, ejercicios mentales, recursos descargables, y una funcionalidad interactiva de chat para contención emocional inicial.

A través de una **navegación intuitiva** y diseño atractivo, se busca facilitar la interacción con los usuarios, brindar experiencias positivas y reforzar hábitos saludables en línea. Además, el sistema se adapta a diferentes dispositivos para garantizar acceso universal, sin importar la condición técnica del usuario.

Requerimientos Funcionales

Los siguientes componentes y funcionalidades forman la base operativa del sitio:

Módulo	Descripción
Sección Misión y Visión	Presenta los valores y objetivos del proyecto, destacando el compromiso social y educativo.
Repositorio de Recursos	Acceso a guías, ejercicios y materiales descargables organizados por tipo (guías, ejercicios, videos).

Buscador y Filtro Dinámico	Permite localizar rápidamente los recursos mediante un motor de búsqueda en tiempo real y botones filtradores por categoría.
Sistema de Publicaciones Motivacionales	Área dinámica donde se presentan mensajes motivadores con posibilidad de incluir imágenes. Se genera desde el panel administrativo.
Chat de Apoyo Emocional	Funcionalidad interactiva que simula un chat con especialistas (mediante lógica de respuestas automáticas). Cuenta con validación de contenido y protección de seguridad.
Panel de Administración Protegido	Accesible sólo mediante contraseña. Permite crear, editar y eliminar contenido de publicaciones.

Requerimientos No Funcionales

Aspecto	Implementación
Diseño Responsivo y Accesible	Adaptado para funcionar en resoluciones móviles, tablets y escritorio. Utiliza media queries y fuentes legibles.
Seguridad del Sistema	Validación de entradas en el chat y en formularios; protección contra ataques XSS mediante filtros de contenido y mensajes de advertencia. Autenticación de acceso al panel de administración.
Experiencia de Usuario (UX)	Interfaz minimalista, mensajes claros, botones accesibles, retroalimentación visual inmediata, y navegación rápida.
Rendimiento y Optimización	Imágenes con loading="lazy", uso eficiente del DOM, y carga asíncrona de componentes como el carrusel y publicaciones.
Persistencia en el Navegador	Uso de localStorage para conservar publicaciones y mensajes del chat sin requerir backend persistente.
Compatibilidad de Navegador	Compatible con navegadores modernos como Chrome, Firefox, Edge y Safari. Utiliza tecnologías estándar sin dependencias críticas.

Valor Agregado

El sistema se construye no solo como una herramienta técnica, sino como una **iniciativa educativa, psicológica y social**, que refleja empatía, ética digital y responsabilidad

tecnológica. Permite además una futura escalabilidad para integrar psicólogos reales, base de datos en tiempo real, o integración con servicios externos de salud mental.

2. Planificación Ágil

Metodología Aplicada: Scrum

Para la gestión y organización del desarrollo del proyecto se adoptó la metodología ágil **Scrum**, debido a su enfoque iterativo, colaborativo y adaptable, ideal para proyectos educativos con cambios constantes y entregas parciales. El equipo simuló el rol de **Product Owner** (cliente), **Scrum Master** (coordinador de tareas) y **Desarrollador Full Stack** (ejecutor técnico), cumpliendo con los roles básicos del marco Scrum.

Se trabajó con un ciclo de **3 sprints semanales**, cada uno con objetivos definidos y entregables parciales que permitieron avanzar en incrementos funcionales.

Ciclo de Vida del Proyecto

Elemento	Aplicación en el Proyecto
Scrum	
Product Backlog	Lista completa de historias de usuario, priorizadas según impacto funcional.
Sprint Backlog	Subconjunto de tareas concretas tomadas del backlog general y definidas por sprint.
Daily Meeting	Revisión individual de avances y bloqueos (simulada mediante checklist personal).
Incremento	Producto funcional entregable al final de cada sprint.
Retrospectiva	Evaluación interna de lo logrado, errores y mejoras al cierre de cada iteración.

Product Backlog (Historias de Usuario)

ID	Historia de Usuario	Prioridad	Sprint asignado
HU01	Como usuario, quiero visualizar contenido motivacional actualizado	Alta	Sprint 1
HU02	Como usuario, quiero descargar recursos para mejorar mi bienestar	Alta	Sprint 1
HU03	Como usuario, quiero filtrar contenido por tipo (guías, videos, ejercicios)	Media	Sprint 2
HU04	Como visitante, quiero chatear con un profesional de forma anónima y segura	Alta	Sprint 2
HU05	Como administrador, quiero publicar mensajes desde un panel protegido	Alta	Sprint 3
HU06	Como administrador, quiero editar o eliminar contenido publicado	Alta	Sprint 3
HU07	Como usuario, quiero navegar desde celular o tablet sin perder funcionalidad	Alta	Sprint 1 y 2
HU08	Como desarrollador, quiero validar la seguridad del sitio (autenticación y XSS)	Alta	Sprint 3

Detalle de Sprints

Sprint 1: Fundamentos del Sitio y Primeros Contenidos

- Objetivo: Construcción de estructura visual, diseño inicial y contenido estático.
- Tareas realizadas:
 - Maquetado de la landing page (HTML/CSS).
 - Diseño de misión, visión y hero con doble sección.
 - Creación del carrusel y primeros recursos descargables.
 - Configuración del layout responsivo.
 - Integración de iconografía con FontAwesome y Google Fonts.

- Entregable: Versión estática funcional y responsive de la plataforma.

Sprint 2: Funcionalidades Interactivas

- Objetivo: Incorporar lógica de interacción y módulos dinámicos para usuarios.
- Tareas realizadas:
 - Desarrollo del módulo de publicaciones dinámicas.
 - Implementación del sistema de filtros y buscador de recursos.
 - Programación del chat de apoyo emocional.
 - Simulación de profesionales con auto respuestas aleatorias.
 - Almacenamiento de mensajes y publicaciones en localStorage.
- Entregable: Plataforma con interacción básica funcional y simulación de asistencia.

Sprint 3: Seguridad y Administración

- Objetivo: Consolidación del sitio con funciones de control, seguridad y pruebas.
- Tareas realizadas:
 - Implementación de modal seguro con autenticación para acceso admin.
 - Desarrollo del panel de administración para publicar, editar y eliminar contenido.
 - Validación de entradas y protección contra inyección XSS en el chat.
 - Mensajes de advertencia ante contenido sospechoso.
 - Pruebas de flujo completo y optimización de interfaz.
- Entregable: Sistema administrable, protegido y validado para su despliegue.

Herramientas utilizadas para planificación

- **Trello:** Para simulación de tablero Kanban.
 - **Google Docs:** Para bitácoras y documentación colaborativa.
 - **Git + GitHub:** Para control de versiones y seguimiento de tareas técnicas.
-

Resultados de la planificación

Gracias al enfoque ágil, fue posible **entregar versiones funcionales al final de cada sprint**, permitir ajustes progresivos, validar componentes individuales antes de su integración y facilitar la documentación continua del proyecto.

3. Arquitectura y Patrones de Diseño

Arquitectura Seleccionada: MVC Modular Client-Side

El desarrollo de la aplicación web “*Mentes Fuertes, Redes Conscientes*” se estructura bajo un enfoque **MVC (Modelo-Vista-Controlador) modular del lado del cliente**, adaptado a entornos sin backend persistente.

Este tipo de arquitectura permite **separar responsabilidades**, mejorar la organización del código y facilitar su mantenibilidad. Se optó por una implementación ligera usando **JavaScript Vanilla**, simulando la separación de capas mediante estructuras modulares, y utilizando localStorage como modelo de persistencia temporal.

Componentes del Modelo MVC:

Componente	Descripción	Implementación
Modelo (Model)	Encargado de almacenar, recuperar y actualizar los datos de publicaciones y chat.	Funciones que gestionan localStorage, como guardarPublicacion(), cargarContenidoAdmin() y saveMessage().
Vista (View)	Representación visual de los datos y la interfaz interactiva del usuario.	Archivos HTML y manipulación del DOM para mostrar publicaciones, recursos y mensajes en pantalla.
Controlador (Controller)	Contiene la lógica que responde a eventos del usuario y coordina las acciones entre el modelo y la vista.	Funciones de eventos JS como publicarContenido(), editarPublicacion(), verificarPassword() y sendMessage().

Esta arquitectura está integrada directamente en el navegador del usuario (client-side), utilizando localStorage como base de datos local y sin requerimientos de servidor o frameworks externos.

Patrones de Diseño Aplicados

Durante la implementación del sistema, se aplicaron varios patrones de diseño de software para mejorar la estructura, escalabilidad y claridad del código.

1. Modular JavaScript

Objetivo: Reutilizar lógica mediante funciones independientes y agrupadas.

Aplicación:

Se utilizaron funciones modulares para encapsular lógica como:

- Filtrado de tarjetas por categoría
- Búsqueda en tiempo real
- Control del chat (enviar, recibir, validar)
- CRUD de publicaciones
Esto permite que cada bloque sea mantenido y probado individualmente, facilitando su extensión.

2. Singleton

Objetivo: Asegurar que solo exista una instancia controlada de un módulo específico.

Aplicación:

El **panel de administración** está protegido por contraseña, y sólo puede activarse una instancia a la vez (simulada con la variable `administradorActivo`). Esto evita conflictos de sesión y controla el acceso a funciones críticas como publicar, editar o eliminar contenido.

3. Observer (Publicador/Suscriptor)

Objetivo: Actualizar automáticamente la vista cada vez que el modelo cambia.

Aplicación:

Cuando se publica o edita contenido, el sistema **actualiza automáticamente la sección dinámica** usando `mostrarPublicaciones()`. Esta función actúa como observador de los cambios en `localStorage`, regenerando los elementos del DOM para reflejar el estado actual sin necesidad de recargar la página.

4. Factory

Objetivo: Crear objetos o elementos DOM dinámicamente a partir de plantillas base.

Aplicación:

La función mostrarPublicaciones() actúa como **fábrica de tarjetas de contenido**, generando cada bloque visual a partir de los datos almacenados (texto, categoría, imagen). Esto garantiza consistencia visual y reduce la duplicación de código.



Ventajas de la Arquitectura y Patrones Aplicados

- **Escalabilidad:** la separación de lógica y presentación permite crecer el proyecto sin reestructurarlo completamente.
- **Reutilización de código:** gracias al uso de funciones modulares y factorías de elementos.
- **Seguridad:** centralizar el control del estado de administrador mediante patrón Singleton ayuda a evitar accesos indeseados.
- **Mantenibilidad:** el uso de patrones estándar facilita la lectura del código por otros desarrolladores.
- **Desempeño:** todo se ejecuta en el navegador sin necesidad de recursos externos, ideal para cargas rápidas.

4. Frameworks y Herramientas Utilizadas

La construcción de la aplicación web “*Mentes Fuertes, Redes Conscientes*” se realizó con tecnologías modernas del lado del cliente, utilizando un stack ligero que favorece el aprendizaje, la portabilidad y el despliegue ágil. A continuación, se detallan las herramientas, librerías y plataformas utilizadas en cada etapa del desarrollo:



Frontend: HTML5, CSS3, JavaScript (Vanilla)



HTML5

- Estructura semántica optimizada para SEO y accesibilidad (<header>, <section>, <main>, etc.).
- Uso de atributos loading="lazy" en imágenes para mejorar el rendimiento.
- Inclusión de metaetiquetas para dispositivos móviles y accesibilidad (<meta name="viewport">).



CSS3

- Implementación de diseño responsive con media queries para distintas resoluciones (480px, 768px, 1200px).
- Variables CSS personalizadas (--primary-color, --dark-color, etc.) para facilitar la gestión de colores y estilos.
- Animaciones sutiles en botones, cards y elementos interactivos (hover, transitions).
- Efectos visuales como box-shadow, border-radius y overlays para mejorar la experiencia de usuario.

JavaScript Vanilla

- Manipulación del DOM para generar contenido dinámico (tarjetas, publicaciones, chat).
 - Modularización de funciones para mantenimiento y escalabilidad.
 - Control de eventos (click, scroll, input) en todas las interacciones.
 - Uso de localStorage como persistencia local para contenido y mensajes del chat.
 - Validaciones de seguridad, manejo de errores y lógica condicional para chat automático y control de administrador.
-

Estilos y Recursos Visuales

FontAwesome

- Iconografía moderna para mejorar la interfaz visual y semántica.
- Aplicación de íconos en menú de navegación, botones, recursos, chat y panel admin.

Google Fonts

- Integración de la familia tipográfica **Montserrat** en diferentes grosores.
 - Mejora de legibilidad, estética y uniformidad visual en todas las secciones del sitio.
-

Control de Versiones: Git + GitHub

Git

- Sistema de control de versiones distribuido para llevar un historial detallado del proyecto.

- Uso de ramas (main, dev) para mantener control de versiones funcionales vs. pruebas.

GitHub

- Repositorio público donde se aloja el código fuente, documentación y recursos.
 - Uso del README con descripción del proyecto, instrucciones de instalación, uso y despliegue.
 - GitHub Pages como opción alternativa de hosting estático en desarrollo inicial.
-
-

Otras Herramientas Complementarias

Herramienta Uso

VS Code	Editor de código fuente con extensiones para HTML, JS, y validación en tiempo real.
Trello / Notion	Organización ágil de tareas y documentación del flujo de trabajo.
Google Docs	Elaboración colaborativa de reportes, bitácoras y manuales.
PlantUML	Creación de diagramas de arquitectura (opcional).

5. Mecanismos de Seguridad Integrados

La aplicación web “*Mentes Fuertes, Redes Conscientes*” incorpora múltiples **medidas de seguridad** orientadas a proteger la integridad del sistema, la privacidad del usuario y la estabilidad de los datos, incluso al ejecutarse completamente en el cliente (frontend).

A continuación, se describen con detalle los mecanismos de seguridad integrados:

1 Autenticación de Administrador mediante Contraseña

- **Objetivo:** Restringir el acceso al panel de administración y operaciones críticas (publicar, editar, eliminar contenido).
- **Implementación:**

- Se utiliza un **modal seguro** que solicita una contraseña al hacer clic en el enlace de "Admin".
 - La clave se valida localmente comparándola con una constante (ADMIN_PASSWORD = "fortaleza2025").
 - Si la autenticación es exitosa:
 - Se activa el modo admin-activo.
 - Se revela el panel de administración.
 - Se habilitan botones ocultos de edición/eliminación en cada publicación.
 - Si la contraseña es incorrecta, se muestra una alerta y no se habilita el modo administrador.
 - **Seguridad adicional:** El acceso sólo está disponible desde el navegador del usuario, y los cambios sólo afectan su instancia local (con localStorage).
-

2 Protección contra Ataques XSS (Cross-Site Scripting) en el Chat

- **Objetivo:** Prevenir que usuarios maliciosos injeten código JavaScript dentro del sistema mediante el chat de apoyo.
- **Implementación:**
 - Antes de guardar o renderizar cualquier mensaje, se valida su contenido utilizando expresiones regulares que detectan:
 - Palabras clave como <script>, javascript:, onerror=, onload=, etc.
 - Si se detecta contenido sospechoso:
 - **El mensaje no se guarda** ni se muestra.
 - Se activa una advertencia visible:

“⚠ Por seguridad, evita incluir enlaces o código en tus mensajes.”

- **Beneficios:**
 - Se evita la ejecución de scripts maliciosos en el navegador.
 - Se protege la interfaz y experiencia de otros usuarios.
-

3 Control de Roles y Permisos

- **Objetivo:** Asegurar que las funciones sensibles del sistema sólo estén disponibles para usuarios autenticados como administradores.
 - **Implementación:**
 - La interfaz de usuario incluye botones Editar y Eliminar en cada publicación, pero están ocultos (opacity: 0) por defecto.
 - Estos botones sólo se activan cuando el usuario ingresa la contraseña correcta y se habilita la clase admin-activo en el <body>.
 - De esta manera, un visitante normal no puede acceder al panel ni ver funciones administrativas.
 - **Ventajas:**
 - Gestión sencilla y efectiva del control de acceso.
 - Visualmente no intrusivo para el usuario común.
 - Se mantiene la lógica separada entre contenido público y privilegios de edición.
-

4 Validación de Entradas en Formularios

- **Objetivo:** Asegurar que la información ingresada por los usuarios sea válida, segura y coherente.
- **Aplicaciones:**
 - **Publicaciones:**
 - Se valida que el mensaje no esté vacío ni compuesto solo de espacios.
 - Se limita el número de caracteres por entrada.
 - **Chat:**
 - Requiere nombre y mensaje no vacíos.
 - Se limita la longitud del mensaje (hasta 500 caracteres).
 - Se bloquea el uso de caracteres especiales peligrosos.
 - **Formulario Admin:**
 - Verificación previa a cada operación de publicación o edición.

- **Resultado:**

- Se evita la corrupción de datos.
 - Se minimiza el riesgo de mal uso o spam.
-

5 Advertencia Visual de Seguridad en el Chat

- **Objetivo:** Informar de manera proactiva a los usuarios sobre prácticas seguras de comunicación.

- **Implementación:**

- En el área de escritura del chat, se despliega un mensaje visual al detectar contenido riesgoso.
- El mensaje aparece en un bloque tipo alerta, con ícono de escudo de seguridad:

“ Por seguridad, evita incluir enlaces o código en tus mensajes.”

- **Impacto:**

- Educa al usuario sobre los límites del sistema.
 - Previene errores o intentos maliciosos por parte de personas no técnicas.
 - Mejora la percepción de confiabilidad de la plataforma.
-

🔒 Resumen de Seguridad Aplicada

Mecanismo de Seguridad	Protección Contra	Implementación
Autenticación de Admin	Acceso no autorizado	Modal con contraseña local
Filtro XSS en el chat	Inyección de scripts	Expresiones regulares en entradas
Control de roles (admin)	Modificación de contenido	Condición admin-activo
Validación de entradas	Datos maliciosos o vacíos	JS + validaciones DOM

Mecanismo de Seguridad	Protección Contra	Implementación
Advertencia de seguridad	Uso indebido por parte del usuario	Indicador visual bajo el textarea

6. Integración de Servicios Web

En este proyecto, se desarrollaron e integraron dos servicios web del lado cliente, completamente funcionales, que simulan comportamiento dinámico utilizando localStorage como sistema de persistencia. Aunque no se implementó un backend REST real por motivos pedagógicos, la lógica simula perfectamente su estructura:

1 Chat de Apoyo Emocional Simulado (Servicio Tipo Conversacional)

Objetivo:

Brindar al usuario un canal interactivo donde pueda comunicarse con un asistente emocional automatizado, en un entorno seguro y empático.

Características técnicas:

- El chat actúa como un servicio **simulado de mensajería**, basado en eventos del usuario.
- Se implementó una **lógica de respuestas automáticas** usando un arreglo de respuestas predefinidas, seleccionadas aleatoriamente.
- Utiliza localStorage para conservar el historial de mensajes por sesión.
- Incluye lógica para mostrar indicadores como:
 - “escribiendo...” por parte del asistente.
 - notificación de nuevo mensaje.
 - validación de contenido con bloqueo de XSS.

Simulación de un servicio real:

- La estructura del sistema está preparada para ser reemplazada por una API real en el futuro (por ejemplo, con un backend en Node.js, Firebase o una conexión a un chatbot como Dialogflow o GPT API).

- La modularidad permite adaptar fácilmente las funciones a peticiones fetch() o axios.
-

CRUD Dinámico de Publicaciones Motivacionales (Servicio Local)

Objetivo:

Permitir al administrador gestionar mensajes motivacionales publicados en el sitio (crear, editar y eliminar contenido).

Características técnicas:

- Implementación del ciclo **CRUD** (Create, Read, Update, Delete) completo en el frontend:
 - Create: publicar contenido con texto, categoría e imagen.
 - Read: visualización automática del contenido publicado en una sección dinámica.
 - Update: edición de contenido vía prompt y re-renderizado en tiempo real.
 - Delete: eliminación segura con confirmación.
- Los datos se guardan en localStorage en formato JSON.
- El panel de administración requiere autenticación previa para habilitar estas funciones.

Ventajas del enfoque:

- Totalmente funcional sin requerir backend.
 - Fácil de migrar hacia una API REST real conectando las mismas funciones a peticiones HTTP.
-

Servicios Web de Terceros

Se integraron servicios y recursos de terceros que enriquecen la presentación del sitio, ofrecen conectividad con redes sociales y mejoran la experiencia visual:

Redes Sociales: Instagram y TikTok

Objetivo:

Fortalecer la conexión con la comunidad juvenil a través de redes donde se comparte contenido adicional.

Implementación:

- Íconos de redes sociales colocados en la sección de contacto.
- Enlaces directos a los perfiles reales del creador en:
 - [Instagram](#)
 - [TikTok](#)
- Los enlaces se abren en una nueva pestaña con target="_blank" para mantener la sesión de la app.

Valor funcional:

- Proporciona canales alternativos de interacción.
 - Fomenta la continuidad del contenido fuera del sitio.
 - Refuerza la presencia del proyecto en entornos sociales reales.
-

[2] Google Fonts

Objetivo:

Incorporar una tipografía moderna y profesional que mejore la estética, accesibilidad y legibilidad.

Detalles técnicos:

- Se importó la familia tipográfica **Montserrat** con diferentes grosores desde fonts.googleapis.com.
- Se aplicó globalmente mediante font-family: 'Montserrat', sans-serif; en CSS.

Ventajas:

- Carga rápida y optimizada.
 - Compatibilidad garantizada en la mayoría de dispositivos.
 - Mejora el diseño visual sin necesidad de archivos locales.
-

[3] FontAwesome

Objetivo:

Enriquecer visualmente el sitio con íconos descriptivos y accesibles.

Implementación:

- Se integró la librería desde CDN (cdnjs) para asegurar compatibilidad.
- Íconos utilizados en:
 - Navegación (Inicio, Visión, Contacto...)
 - Botones (Enviar, Descargar, Editar...)
 - Categorías de recursos (videos, guías, ejercicios)
 - Panel de administrador y secciones protegidas.
 - Chat de ayuda (íconos de escudo, seguridad, papel avión...)

Beneficios:

- Mejora la comunicación visual de las funciones.
 - Brinda una experiencia de usuario más intuitiva y estética.
 - Ligero y fácil de mantener.
-

Consideraciones para Escalabilidad

En una futura evolución del proyecto, estos servicios podrían extenderse con:

Propuesta	Tecnología Sugerida
Backend real para publicaciones y chat	Node.js + Express + MongoDB
Integración con chatbot real	Dialogflow, GPT API o Watson Assistant
Autenticación avanzada	Firebase Authentication
API de clima o noticias emocionales	OpenWeather, NewsAPI

7. Pruebas y Validación

Para garantizar el correcto funcionamiento, seguridad y experiencia del usuario final, se aplicó un proceso sistemático de **pruebas y validación**, alineado con las funcionalidades clave del sistema.

Las pruebas se dividieron en tres tipos principales:

- **Pruebas funcionales**
- **Pruebas de interfaz y responsividad**

- Pruebas de seguridad (validación XSS y control de acceso)
-

1. Pruebas Funcionales

Estas pruebas verificaron que **cada componente del sistema cumpliera con su propósito funcional**. Se simularon interacciones reales por parte del usuario y del administrador.

Casos de prueba incluidos:

Prueba	Descripción	Resultado Esperado	Resultado
Botón "Enviar" en el chat	Simulación de mensaje entre usuario y sistema	Mensaje aparece con respuesta automática	<input checked="" type="checkbox"/> Exitoso
Botón "Descargar recurso"	Activación de enlace de descarga	Se inicia descarga del archivo vinculado	<input checked="" type="checkbox"/> Exitoso
Botón "Iniciar sesión Admin"	Acceso con contraseña correcta	Se habilita el modo administrador	<input checked="" type="checkbox"/> Exitoso
Edición de publicación	Cambiar texto desde el panel admin	Texto actualizado en pantalla y almacenamiento	<input checked="" type="checkbox"/> Exitoso
Eliminación de publicación	Borrar mensaje desde panel admin	Tarjeta eliminada del DOM y del localStorage	<input checked="" type="checkbox"/> Exitoso
Botón "Publicar mensaje" vacío	Enviar sin contenido	Aparece mensaje de error o se bloquea	<input checked="" type="checkbox"/> Bloqueado correctamente

2. Pruebas de Interfaz y Responsividad

Se verificó que la aplicación fuera **completamente responsive** y que la interfaz se adaptara a diferentes dispositivos y tamaños de pantalla.

Dispositivos simulados:

- Smartphone: 375px (iPhone SE)
- Tablet: 768px (iPad vertical)
- Laptop estándar: 1366px
- Monitor grande: 1920px

Elementos validados:

Elemento	Verificación	Estado
Menú de navegación	Se adapta a columna vertical en móviles	✓ Correcto
Carrusel motivacional	Oculta/ajusta elementos según resolución	✓ Correcto
Panel de admin	Oculto por defecto, visible sólo en escritorio con acceso válido	✓ Correcto
Chat	Mantiene el scroll y formato en pantallas pequeñas	✓ Correcto
Botones e íconos	Escalado y legibilidad adecuada	✓ Correcto
Imágenes y recursos	Se adaptan con max-width: 100% y lazy loading	✓ Correcto

3. Pruebas de Seguridad (XSS y Control de Acceso)

Se realizaron pruebas para evaluar la **resistencia del sistema ante entradas maliciosas** y el **cumplimiento de las restricciones de rol** (usuario vs. administrador).

Pruebas XSS:

Entrada Simulada	Resultado Esperado	Resultado
<script>alert("Hacked")</script>	Entrada bloqueada	✓ Detectada y rechazada
javascript:alert("XSS")	Entrada bloqueada	✓ Detectada y rechazada
	Entrada bloqueada	✓ Detectada y rechazada
Texto normal (Hola mundo)	Entrada aceptada	✓ Permitido correctamente

Validación de roles:

Escenario	Acción Intentada	Resultado Esperado	Resultado

Usuario sin contraseña	Acceder a panel admin	Denegado	<input checked="" type="checkbox"/> Acceso restringido
Usuario como admin	Editar publicación	Permitido	<input checked="" type="checkbox"/> Permitido
Usuario como admin	Eliminar publicación	Permitido	<input checked="" type="checkbox"/> Permitido
Usuario común	Ver botones admin	Ocultos	<input checked="" type="checkbox"/> Ocultos correctamente

4. Pruebas de Flujo Completo (Publicación y Administración)

Se simuló un flujo completo desde el punto de vista del administrador y del visitante para garantizar la integridad de los ciclos:

Escenario 1: Publicación

1. Ingreso como administrador.
2. Redacción de mensaje motivacional.
3. Selección de categoría e imagen.
4. Publicación.
5. Verificación inmediata en la sección de contenido.
6. Confirmación en localStorage.

Resultado: Flujo funcional sin errores.

Escenario 2: Edición/Eliminación

1. Selección de tarjeta publicada.
2. Edición mediante prompt.
3. Eliminación con confirm().
4. Verificación de actualización visual y persistencia.

Resultado: Edición y eliminación reflejadas en tiempo real.

Resumen de Validación

Tipo de Prueba	Casos Exitosos	Casos Fallidos	Observaciones
Funcionales	9	0	Sistema estable
Interfaz responsiva	6	0	Buena experiencia UX
Seguridad (XSS/Admin)	8	0	Seguridad básica efectiva
Flujo completo CRUD	2	0	Flujo validado con localStorage

8. Repositorio y Entrega

Repositorio: <https://github.com/ErikMora-sbg/MentesFuertesRedesConscientesOfficial>

Github: <https://erikmora.github.io/mentesfredesc/>

Incluye:

- Código fuente
- Documentación técnica
- README con instrucciones

9. Diagramas Visuales

A continuación se presenta el diagrama de arquitectura de componentes de la aplicación:

