Educación y Desarrollo Cultural de Monterrey

Actividad Integradora: Entrega Proyecto Final

Topico: Desarrollo de aplicaciones web

Erick Hernandez Abrego 31945

Campus Miguel Aleman

Diseño grafico

A 07 de agosto de 2025

1. Índice

2. Introducción

El "Generador de Contenido Estratégico" es una aplicación web diseñada para **automatizar y simplificar la creación de borradores de contenido** profesional, como campañas de marketing e informes, utilizando inteligencia artificial generativa. Su propósito es facilitar a equipos de marketing, comunicación y estrategia la producción frecuente de textos de valor, ofreciendo una plataforma rápida, personalizable y eficiente.

- **ENLACE A LA APLICACIÓN WEB PUBLICADA:** https://gestion-de-contenidos.wuaze.com/
- **ENLACE AL REPOSITORIO DE CÓDIGO FUENTE:** https://github.com/Erickhdz1/Generador-de-Contenido-Estrategico-.git

3. Fase 1: Análisis

2.1. Descripción del Problema

Las empresas y profesionales a menudo enfrentan el desafío de **generar contenido estratégico de manera constante y eficiente**. Este proceso puede ser laborioso y requerir conocimientos especializados en redacción y marketing, lo que limita la capacidad de producir textos de valor con la frecuencia deseada.

2.2. Solución Propuesta

Se propone una **aplicación web** que permita a empresas, profesionales y personas no técnicas generar automáticamente borradores de contenido estratégico (campañas de marketing, resúmenes ejecutivos, informes) utilizando **inteligencia artificial generativa**. La

herramienta busca facilitar este proceso al ofrecer una plataforma sencilla, rápida y personalizable.

2.3. Requisitos Funcionales

- El sistema debe generar contenido automáticamente basado en las entradas proporcionadas por el usuario.
- El usuario debe poder personalizar el tono del contenido generado (formal, informal, emocional, etc.).
- El sistema debe permitir la visualización y edición básica de los borradores generados.
- El usuario debe poder eliminar borradores.
- El usuario debe poder exportar el contenido en formato texto o copiar al portapapeles.
- El sistema debe contar con una interfaz web funcional.

2.4. Alcance del Proyecto (Dentro/Fuera del Curso)

- **Dentro del curso:** Generación automática de texto, personalización de tono, visualización y edición básica del borrador, interfaz web funcional.
- **Fuera de alcance (planeadas a futuro):** Autenticación de usuarios y perfiles, historial y versiones de contenido, descarga del contenido en PDF, inserción de logo empresarial en PDF, inserción de gráficos basados en datos, publicación automática en redes sociales, recomendaciones de palabras clave/títulos por IA, ampliación a otros idiomas.

2.5. Entidades Principales

Se identificaron 4 entidades clave:

- **Usuario:** La persona que utiliza el sistema.
- **Borrador:** El documento generado por la IA, que contiene el texto creado según los parámetros seleccionados.
- **Tipo de Contenido:** Determina la estructura del contenido solicitado (ej. campaña de marketing, informe).
- **Tono:** Configura el estilo del contenido (ej. formal, casual, creativo).

4. Fase 2: Diseño

3.1. Diseño de Base de Datos

La base de datos se diseñó para almacenar los borradores de contenido. Se utiliza una única tabla llamada borradores.

Código SQL (CREATE TABLE):

```
CREATE TABLE borradores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(255) NOT NULL,
  contenido TEXT,
  tipo_contenido VARCHAR(100) NOT NULL,
  tono VARCHAR(100) NOT NULL,
  fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Decisiones de Diseño Clave:

- **id (PRIMARY KEY, AUTO_INCREMENT):** Identificador único para cada borrador, generado automáticamente.
- **titulo (VARCHAR):** Almacena el título del borrador, con una longitud limitada.
- **contenido (TEXT):** Almacena el texto generado por la IA, permitiendo grandes volúmenes de texto.
- **tipo_contenido (VARCHAR):** Almacena el tipo de contenido seleccionado por el usuario.
- **tono (VARCHAR):** Almacena el tono seleccionado por el usuario.
- **fecha_creacion (TIMESTAMP):** Registra automáticamente la fecha y hora de creación del borrador.

3.2. Diseño de Interfaz de Usuario

La interfaz de usuario se diseñó para ser **intuitiva y responsiva**, adaptándose a diferentes tamaños de pantalla. Se conceptualizaron dos secciones principales:

- **Formulario de Generación/Edición:**
    - Ubicado en la parte superior.
    - Campos de entrada para **Título del Borrador**, selectores para **Tipo de Contenido** y **Tono**, y un área de texto para la **Idea o tema principal**.
    - Un botón principal que cambia entre "Generar y Guardar Borrador" (para crear nuevos) y "Actualizar Borrador" (cuando se está editando).
    - Un botón "Cancelar Edición" aparece al editar para limpiar el formulario.

- **Lista de Borradores Guardados:**
    - Ubicada debajo del formulario.
    - Cada borrador se muestra como una tarjeta, con su título, un extracto del contenido, el tipo, el tono, la fecha de creación y el número de palabras.
    - Botones de acción (Editar, Eliminar, Copiar) en cada tarjeta para la gestión individual del borrador.
    - Un botón "Ver más" para expandir/contraer el contenido completo.

El flujo de usuario es lineal: generar un borrador, visualizarlo en la lista, y desde ahí, editarlo o eliminarlo.

3.3. Diseño de API Backend

La API RESTful en PHP facilita la comunicación entre el frontend y la base de datos.

- **GET /api/borradores/leer.php**
    - **Propósito:** Recuperar todos los borradores guardados.
    - **Método HTTP:** GET
    - **Espera:** Nada.
    - **Devuelve:** Un array de objetos JSON, donde cada objeto es un borrador (ej. [{id: "1", titulo: "...", contenido: "...", tipo_contenido: "...", tono: "...", fecha_creacion: "..."}, ...]).
- **POST /api/borradores/crear_test.php**
    - **Propósito:** Crear un nuevo borrador.
    - **Método HTTP:** POST
    - **Espera:** JSON con {"titulo": "...", "contenido": "...", "tipo_contenido": "...", "tono": "..."}.
    - **Devuelve:** JSON con {"message": "Borrador creado exitosamente."} o un mensaje de error.
- **POST /api/borradores/actualizar.php**
    - **Propósito:** Actualizar un borrador existente.
    - **Método HTTP:** POST (con *workaround* _method='PUT')
    - **Espera:** JSON con {"id": "...", "titulo": "...", "contenido": "...", "tipo_contenido": "...", "tono": "...", "_method": "PUT"}.
    - **Devuelve:** JSON con {"message": "Borrador actualizado exitosamente."} o un mensaje de error.

- **POST /api/borradores/eliminar.php**
    - **Propósito:** Eliminar un borrador.
    - **Método HTTP:** POST (con *workaround* _method='DELETE')
    - **Espera:** JSON con {"id": "...", "_method": "DELETE"}.
    - **Devuelve:** JSON con {"message": "Borrador eliminado correctamente."} o un mensaje de error.

5. Fase 3: Codificación (Implementación)

4.1. Arquitectura y Tecnologías

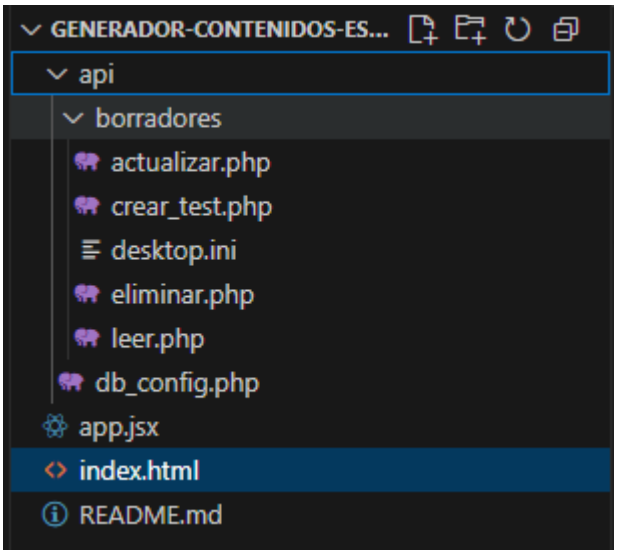
La aplicación implementa una **arquitectura full-stack**:

- **Frontend:** Desarrollado con **React** (integrado directamente en index.html), aprovechando useState y useEffect para el manejo de estado y el ciclo de vida de los componentes. El estilo se gestiona con **Tailwind CSS**. **Babel Standalone** se utiliza para la transpilación de JSX en el navegador.
- **Backend:** Construido con **PHP** para exponer una API RESTful. Se encarga de la lógica de negocio y la interacción con la base de datos.
- **Base de Datos: MySQL** almacena los datos de los borradores.
- **Inteligencia Artificial:** Se integra con la **API de Google Gemini (modelo gemini-2.5-flash-preview-05-20)** para la generación de texto.

4.2. Estructura del Proyecto

El proyecto se organiza en un servidor web local (XAMPP/Apache) de la siguiente manera:

/htdocs/generador-contenidos-estrategico/



4.3. Desafíos Técnicos

1. **Compatibilidad de Métodos HTTP con Hosting Gratuito (InfinityFree):**
    - **Desafío:** Servicios como InfinityFree a menudo restringen los métodos HTTP PUT y DELETE directamente.
    - **Resolución:** Se implementó un *workaround* donde el frontend envía todas las solicitudes de actualización y eliminación como POST requests. En el cuerpo de la solicitud JSON, se incluyó un campo _method ("PUT" o "DELETE") que el backend PHP interpreta. Los scripts PHP (actualizar.php, eliminar.php) fueron modificados para leer este _method y ejecutar la operación de base de datos correspondiente.
2. **Regeneración Dinámica de IA en Edición:**
    - **Desafío:** Originalmente, al editar un borrador y cambiar el tono/tipo, el contenido del texto no se actualizaba porque la IA no se volvía a invocar.
    - **Resolución:** La lógica de la función handleSaveDraft en React fue refactorizada para que la llamada a la API de Gemini se ejecute **siempre** que se envía el formulario, ya sea para crear un nuevo borrador o para actualizar uno existente. Esto asegura que si el usuario ajusta el tono o el tipo de contenido al editar, la IA regenere el texto basándose en la idea principal y los nuevos parámetros, manteniendo la coherencia.
3. **Prevención de Inyección SQL:**
    - **Desafío:** La manipulación directa de cadenas en consultas SQL es una vulnerabilidad común.
    - **Resolución:** Se implementaron **sentencias preparadas ($mysqli->prepare()) con bind_param()** en todos los scripts PHP de la API (crear_test.php, leer.php, actualizar.php, eliminar.php). Esto parametiza los valores de entrada, separándolos de la lógica SQL y neutralizando la inyección de código malicioso.

6. Fase 4: Implantación (Despliegue)

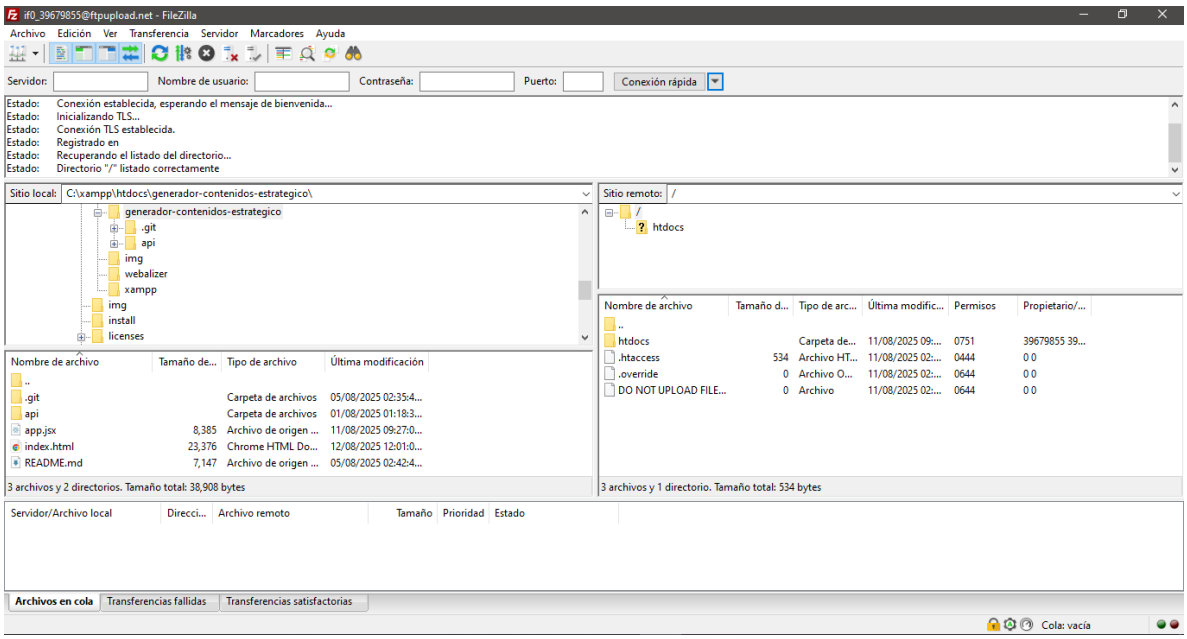
5.1. Proveedor de Hosting

La aplicación está desplegada en **InfinityFree**, un proveedor de hosting web gratuito.

5.2. Proceso de Despliegue

1. Creación de Cuenta de Hosting: Se creó una nueva cuenta de hosting en InfinityFree, obteniendo un subdominio y credenciales FTP.

2. Subida de Archivos del Proyecto: Todos los archivos del proyecto (HTML, PHP, CSS) se subieron al directorio htdocs del servidor de InfinityFree utilizando un cliente FTP (FileZilla). Se mantuvo la estructura de carpetas (api/borradores/).



3. Configuración de Base de Datos Remota: Se creó una nueva base de datos MySQL a través del panel de control de InfinityFree.Se obtuvieron las credenciales de la base de datos remota (nombre del host, nombre de la base de datos, usuario, contraseña).

4. Se accedió a phpMyAdmin de InfinityFree y se ejecutó el script SQL CREATE TABLE borradores para crear la tabla con la estructura correcta (incluyendo tipo_contenido y tono).

5. Actualización de Credenciales en db_config.php: El archivo db_config.php local se modificó con las nuevas credenciales de la base de datos de InfinityFree y se subió nuevamente al servidor vía FTP, sobrescribiendo el archivo antiguo.

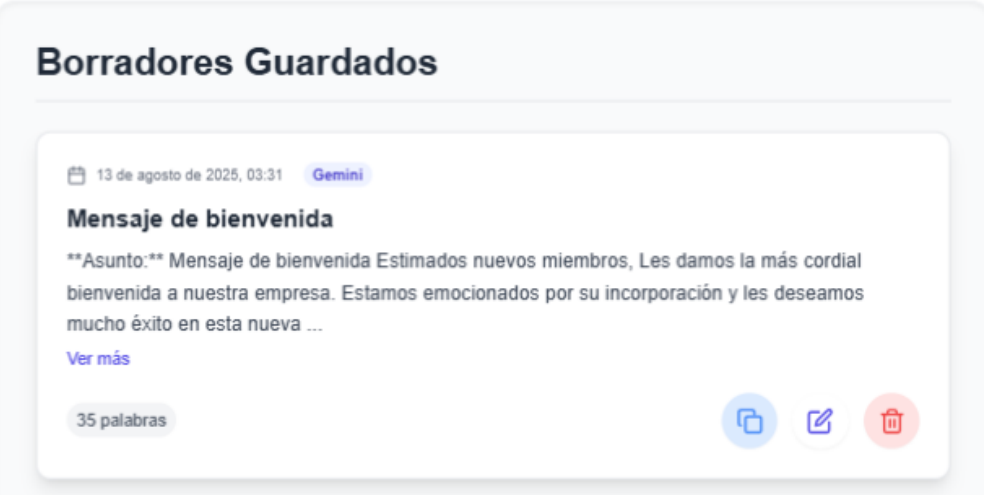
6. Verificación de API Key de Gemini: Se confirmó que la clave de API de Gemini en index.html fuera la correcta y estuviera activa.
7. **Ajuste de API_BASE_URL:** La URL base para las llamadas a la API en el frontend (index.html) se cambió de http://localhost/... a la URL pública del dominio de InfinityFree con HTTPS (https://gestion-de-contenidos.wuaze.com).

5.3. Instrucciones de Uso/Prueba

La aplicación es de uso directo y no requiere autenticación.

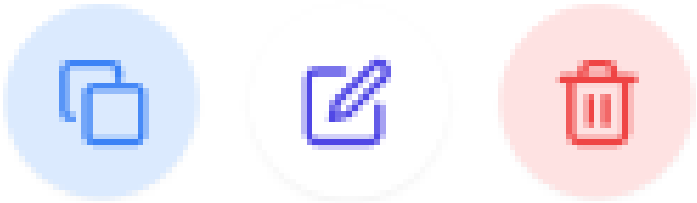
1. Accede a la aplicación desde cualquier navegador en: https://gestion-de-contenidos.wuaze.com/
2. **Generar Contenido:** Rellena los campos "Título del Borrador", "Tipo de Contenido", "Tono" e "Idea o tema principal" y haz clic en "Generar y Guardar Borrador".

3. **Ver Borradores:** Los borradores aparecerán en la lista inferior. Puedes hacer clic en "Ver más" para expandir su contenido completo.

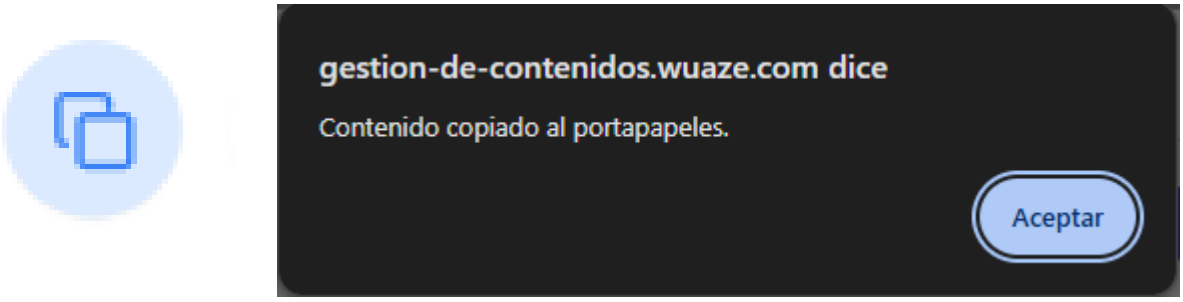


4. **Editar Borrador:** Haz clic en el icono de **lápiz** en una tarjeta de borrador. Modifica el título, el tipo de contenido, el tono, o el texto en el área "Idea o tema principal" (que ahora es el contenido editable). Haz clic en "Actualizar Borrador". Observa cómo el contenido se regenera si cambias el tipo/tono.

5. **Eliminar Borrador:** Haz clic en el icono de **bote de basura** en una tarjeta de borrador para eliminarlo permanentemente.



6. **Copiar Contenido:** Haz clic en el icono de **copiar** en una tarjeta de borrador para copiar su contenido al portapapeles.



7. Conclusiones

Reflexión Personal

El desarrollo de este "Generador de Contenido Estratégico" ha sido una experiencia de aprendizaje sin igual. La verdad es que no tengo un vasto conocimiento en el desarrollo de aplicaciones web ni siquiera de paginas, se muy poco de html. Realizar este proyecto fue todo un reto para y me abrió todo un nuevo mundo, me hizo entender la complejidad con la que se realiza una pagina web o un proyecto de este tipo. Probablemente para gente con mas experiencia es muy sencillo realizar todo el proceso, pero yo si tuve muchos obstáculos por que al principio no entendía muy bien que estaba realizando y el proceso se quedo estancado

Entender que uno debe de trabajar bajo cierta lógica y orden por que si no la aplicación no funciona, fue muy formativo para mi pues ahora en cosas personales mias ya estoy aplicando estructura y organización que antes no hacia y ahora me siento mas eficiente al realizar estas actividades.

Es un gran reto el desarrollo web, pero actualmente existen muchas herramientas que te facilitan el proceso, entendí que debo explotar todas estas áreas de oportunidad y supe sacarle provecho de una manera diferente a todas estas herramientas. La conclusion de este proyecto me parece muy gratifacante a nivel personal

¿Qué haría diferente la próxima vez?

- **Herramientas de Build:** Para un proyecto real, implementaría una herramienta de *build* como **Vite** o **Webpack** desde el principio. Esto eliminaría la necesidad de Babel Standalone en el navegador, mejoraría el rendimiento de carga y simplificaría la gestión de módulos y dependencias.
- **Estructura de Carpetas:** Organizaria el proyecto React en una estructura de carpetas más estándar (src/components, src/pages, src/api) y utilizaría un sistema de módulos adecuado en lugar de tener todo el código React en index.html.
- **Gestión de Notificaciones:** En lugar de alert() o mensajes simples, integraría una librería de notificaciones (ej. Toastify) para una UX más fluida.

Siguientes Pasos

Si el desarrollo continuara, las prioridades serían:

- **Autenticación y Perfiles de Usuario:** Permitir a cada usuario tener su propio espacio de trabajo privado para almacenar borradores.
- **Opciones Avanzadas de IA:** Integrar más parámetros de la API de Gemini, como control de longitud del texto, ajuste de temperatura, o generación de múltiples borradores por solicitud.
- **Exportación a Otros Formatos:** Añadir funcionalidad para descargar borradores en formatos como PDF o DOCX.
- **Historial de Ediciones:** Implementar un sistema de versiones para los borradores, permitiendo revertir a estados anteriores.