INSTTITUTO TEGNOLOGICO SUPERIOR DE MEXICO

# Aplicación Chat IA con Google Gemini

Alumno: Alan Geovannie Morales Salazar

Matricula: I19050458

Carrera: Ingeniería Informática

Materia: Inteligencia Artificial

Semestre: 9no

## Introducción

"Chat IA con Google Gemini" es una aplicación de inteligencia artificial diseñada para servir como un asistente virtual integral. Desarrollada sobre la arquitectura de vanguardia de Google Gemini, esta plataforma representa una solución confiable y de alto rendimiento para usuarios que requieren procesamiento avanzado de lenguaje natural, ya sea para tareas personales, académicas o profesionales. Su objetivo central es facilitar el acceso a información y herramientas de generación de contenido, optimizando la productividad y la toma de decisiones.

## Justificación

Chat IA con Google Gemini  nace para hacer tu vida más fácil. Hoy en día, todos necesitamos respuestas rápidas, ayuda para escribir o alguien que nos explique cosas complicadas en simple. Esta aplicación es como tener un asistente inteligente siempre a la mano.

En lugar de buscar en mil lugares o perder tiempo con tareas repetitivas, puedes:

- Preguntar lo que necesites en español y obtener una respuesta clara al instante.
- Crear textos, ideas o resúmenes sin esfuerzo.
- Entender temas difíciles de forma sencilla.

La creamos porque creemos que la tecnología debe servir para simplificar, no para complicar. Usar la potencia de Google Gemini en una app fácil de usar te ahorra tiempo y te ayuda a ser más productivo, sin necesidad de ser un experto

## Objetivo

El objetivo principal de Chat IA con Google Gemini es ser tu compañero de ayuda diario, ofreciéndote respuestas rápidas, ideas creativas y soluciones prácticas en español para hacer tus tareas más fáciles y ahorrarte tiempo.

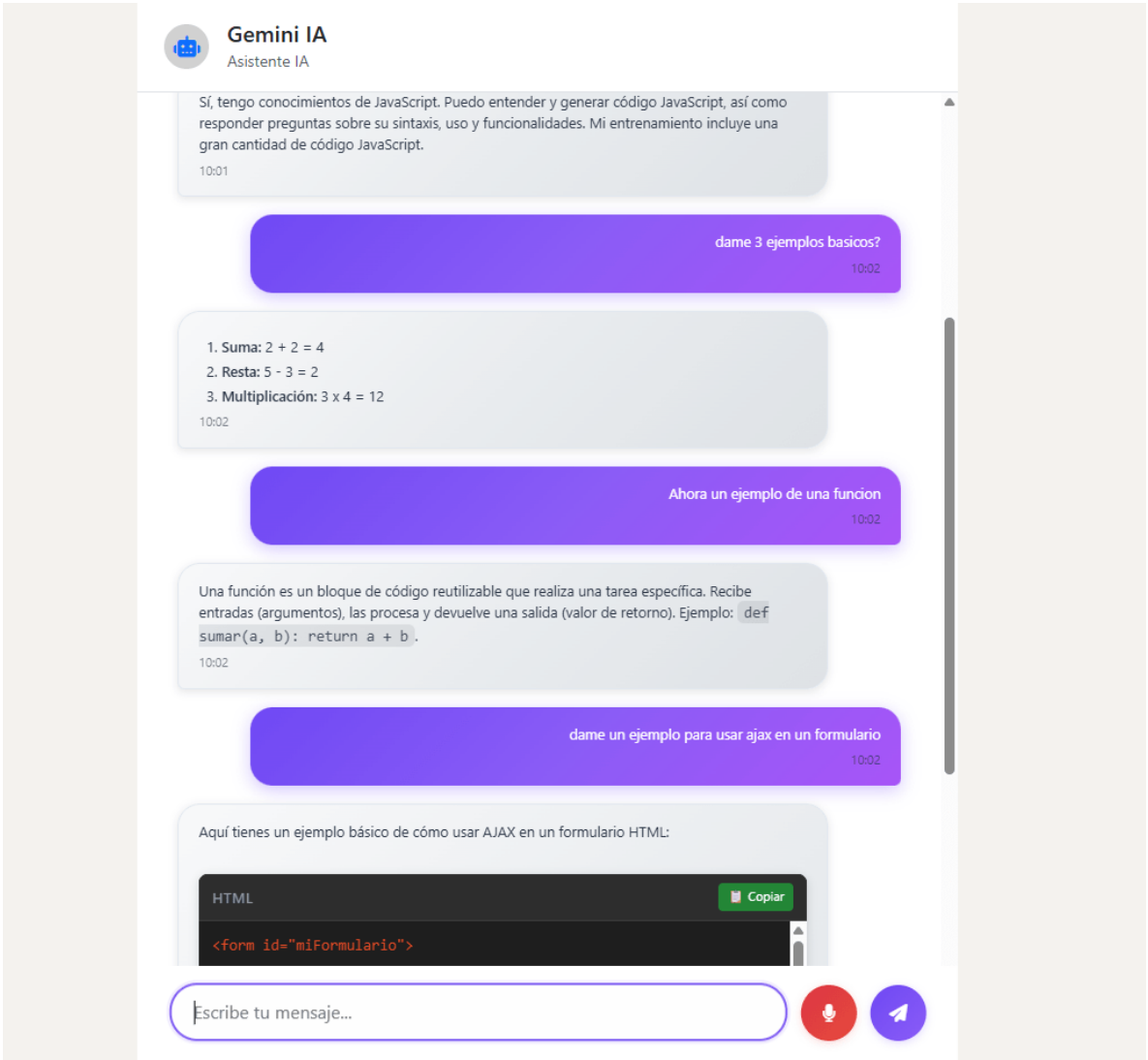
En concreto, buscamos:

- Ayudarte en lo que necesites: Desde resolver una duda puntual hasta escribir un texto o organizar tus ideas.
- Ser simples y directos: Que cualquier persona pueda usar la aplicación sin complicaciones.
- Hacer la tecnología accesible: Llevar el poder de una IA avanzada a tu día a día de forma gratuita y sencilla.

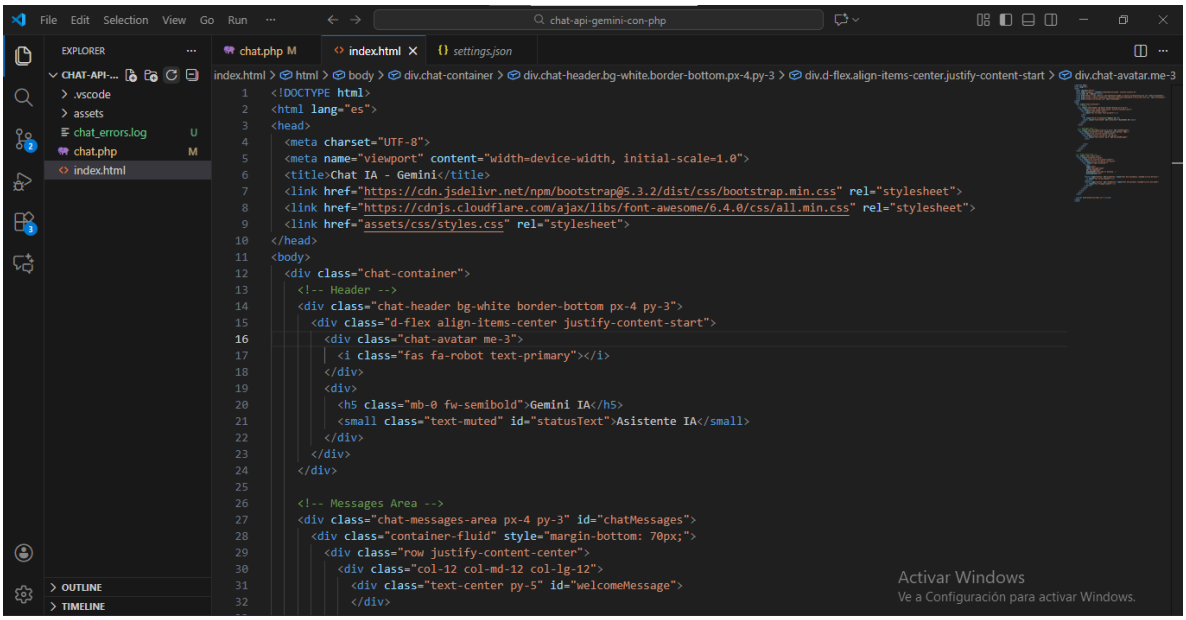
## Cronograma de Actividades

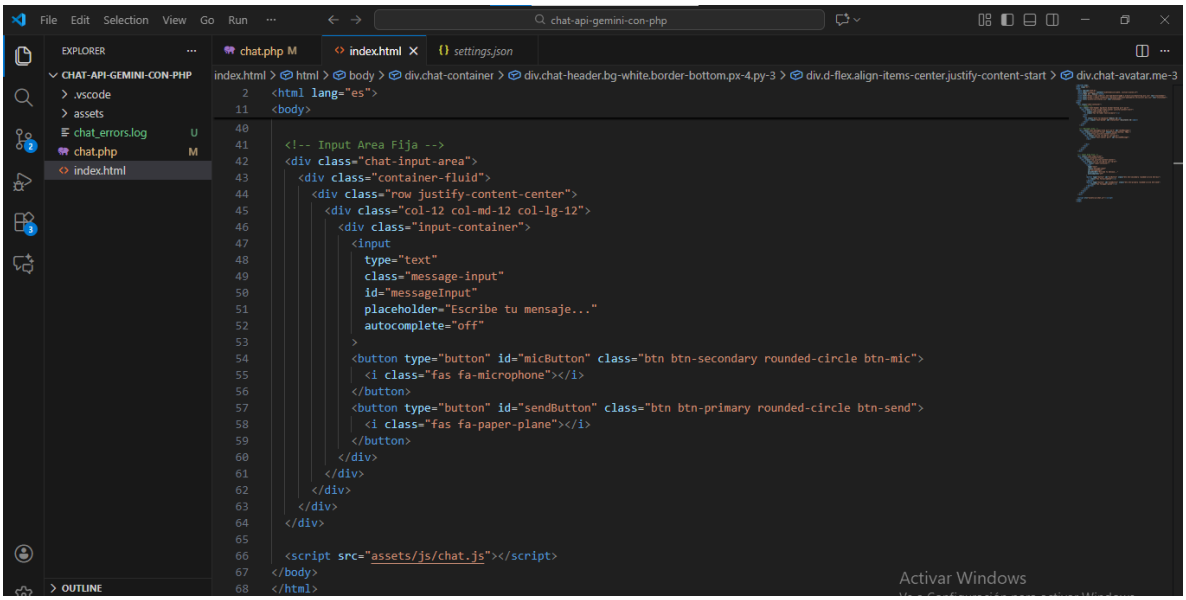
| Fase | Actividades Clave | Duración Estimada | Responsable |
|---|---|---|---|
| **1. Planificación** | - Definir requisitos de la app.<br>- Investigar sobre la API de Gemini.<br>- Establecer diseño inicial. | 2 semanas | Equipo de Proyecto |
| **2. Desarrollo** | - Configurar el entorno de desarrollo.<br>- Integrar la API de Gemini 2.0.<br>- Programar la interfaz de usuario (UI). | 2 semanas | Equipo de Desarrollo |
| **3. Pruebas** | - Probar la aplicación (funcionalidades y errores).<br>- Revisar que las respuestas<br>- Corregir fallos detectados. | 2 semanas | Equipo de Calidad |
| **4. Lanzamiento** | - Preparar la app para usuarios finales. | 1 semana | Equipo de Despliegue |
| **5. Mantenimiento** | - Monitorear el funcionamiento.<br>- Solucionar problemas. | Continuo | Equipo de Soporte |

## Interfaz de la aplicación

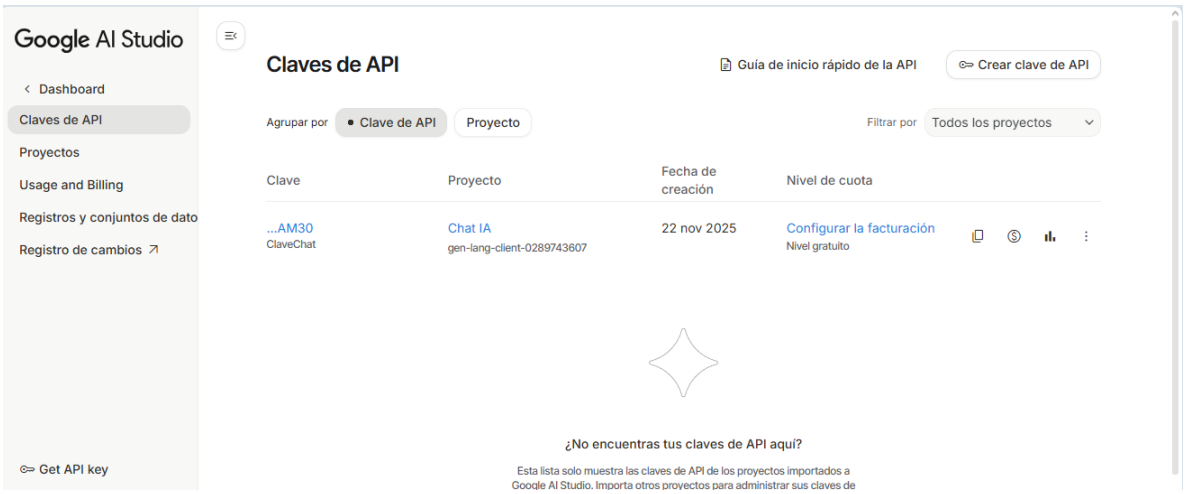


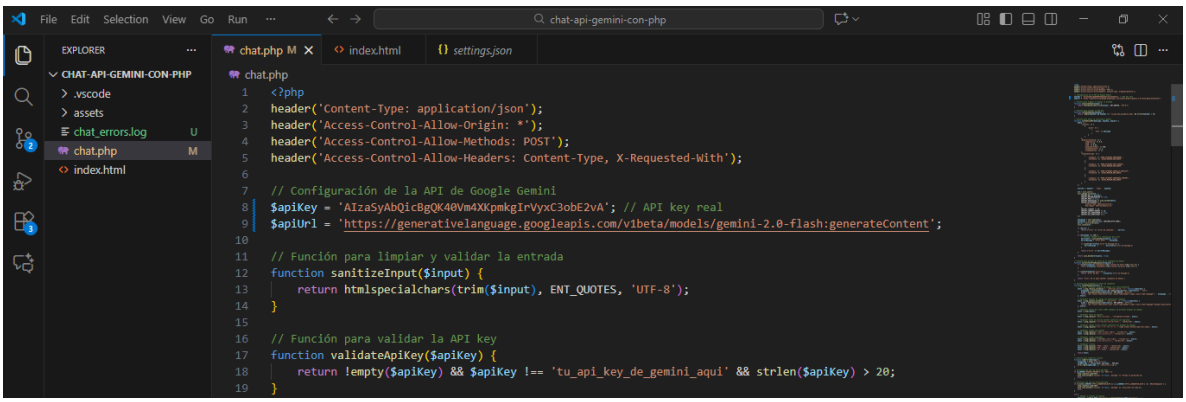
Comenzamos con la elaboración del archivo index.html y posteriormente el diseño



Luego la creación del archivo chat.php luego la configuración de la API de Google Gemini y usaremos la “apiKey” y apiUrl nuestra cuenta de Google IA studio esto lo encontraremos en el apartado de “Get API Key” en la página [https://aistudio.google.com](https://aistudio.google.com)



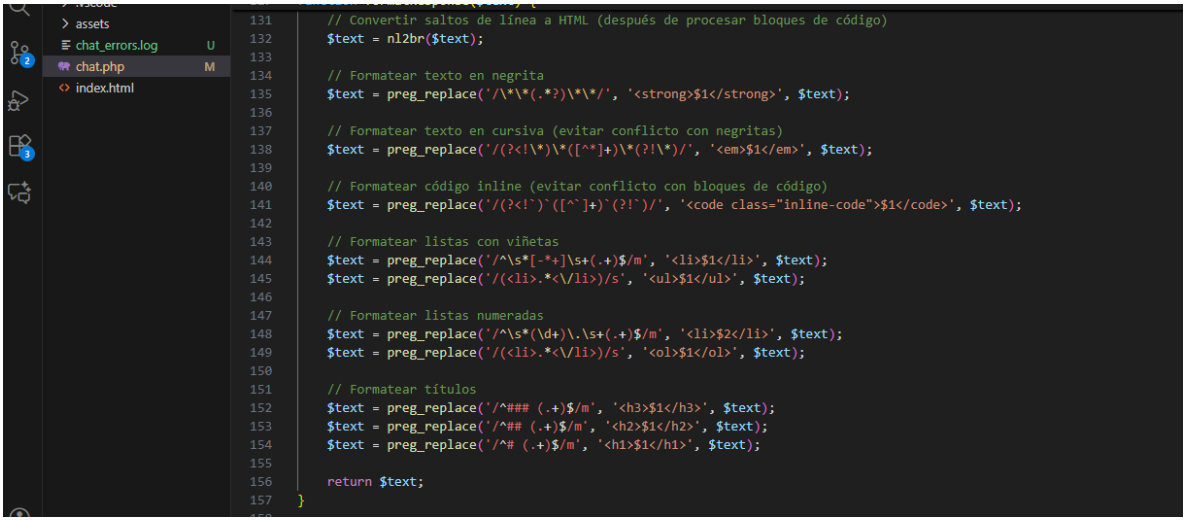



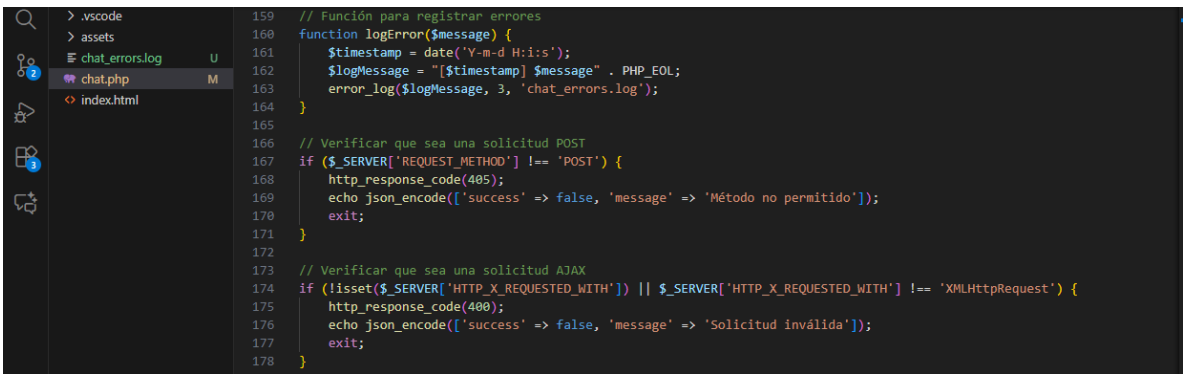
Implementamos la función para validar al API key y la función para solicitar a Gemini



chat_errors.log U
chat.php M
index.html
OUTLINE
TIMELINE
CURLOPT_HTTPHEADER => [
'Content-Type: application/json',
'User-Agent: Chat-IA-Gemini/1.0'
],
CURLOPT_TIMEOUT => 30,
CURLOPT_CONNECTTIMEOUT => 10,
CURLOPT_SSL_VERIFYPEER => true,
CURLOPT_SSL_VERIFYHOST => 2
]);
$response = curl_exec($ch);
$httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
$error = curl_error($ch);
curl_close($ch);
if ($error) {
return ['error' => 'Error de conexión: ' . $error];
}
if ($httpCode !== 200) {
// Intentar obtener más información del error
$errorInfo = json_decode($response, true);
$errorMessage = 'Error HTTP: ' . $httpCode;
if (isset($errorInfo['error']['message'])) {
$errorMessage .= ' - ' . $errorInfo['error']['message'];
}
return ['error' => $errorMessage];
}
Activar Windows
Ve a Configuración para activar

Luego la función para extraer el texto de la respuesta de Gemini, también para formatear el texto de respuesta y también formatear bloques de código sin especificar lenguaje


assets
chat_errors.log U
chat.php M
index.html
// Obtener y validar el mensaje
$message = isset($_POST['message']) ? sanitizeInput($_POST['message']) : '';
if (empty($message)) {
echo json_encode(['success' => false, 'message' => 'El mensaje no puede estar vacío']);
exit;
}
if (strlen($message) > 4000) {
echo json_encode(['success' => false, 'message' => 'El mensaje es demasiado largo (máximo 4000 caracteres)']);
exit;
}
// Validar API key
if (!validateApiKey($apiKey)) {
echo json_encode(['success' => false, 'message' => 'API key no configurada. Por favor, configura tu API key de G
exit;
}
// Verificar si cURL está disponible
if (!function_exists('curl_init')) {
echo json_encode(['success' => false, 'message' => 'cURL no está disponible en este servidor']);
exit;
}
// Preparar el contexto para Gemini
$contextualMessage = "Eres Gemini, un asistente IA eficiente. Responde de forma BREVE y PRECISA. Máximo 3-4 oracione


chat.php M
index.html
// Hacer la solicitud a Gemini
$response = callGeminiAPI($contextualMessage, $apiKey, $apiUrl);
// Verificar si hay errores en la respuesta
if (isset($response['error'])) {
logError('Error de Gemini API: ' . json_encode($response['error']));
echo json_encode(['success' => false, 'message' => 'Error al comunicarse con Gemini: ' . $response['error']]);
exit;
}
// Extraer y formatear la respuesta
$responseText = extractTextFromResponse($response);
$formattedResponse = formatResponse($responseText);
// Verificar que la respuesta no esté vacía
if (empty(trim(strip_tags($formattedResponse)))) {
echo json_encode(['success' => false, 'message' => 'Gemini no pudo generar una respuesta']);
exit;
}


// Enviar la respuesta exitosa
echo json_encode([
'success' => true,
'message' => $formattedResponse,
'timestamp' => date('Y-m-d H:i:s')
]);
} catch (Exception $e) {
logError('Excepción capturada: ' . $e->getMessage());
echo json_encode(['success' => false, 'message' => 'Error interno del servidor']);
}
?>

Finalmente corremos nuestra aplicación en el servidor de apache usando XAMPP y funciona correctamente