



INSTITUTO TECNOLÓGICO SUPERIOR DE MÉXICO

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.

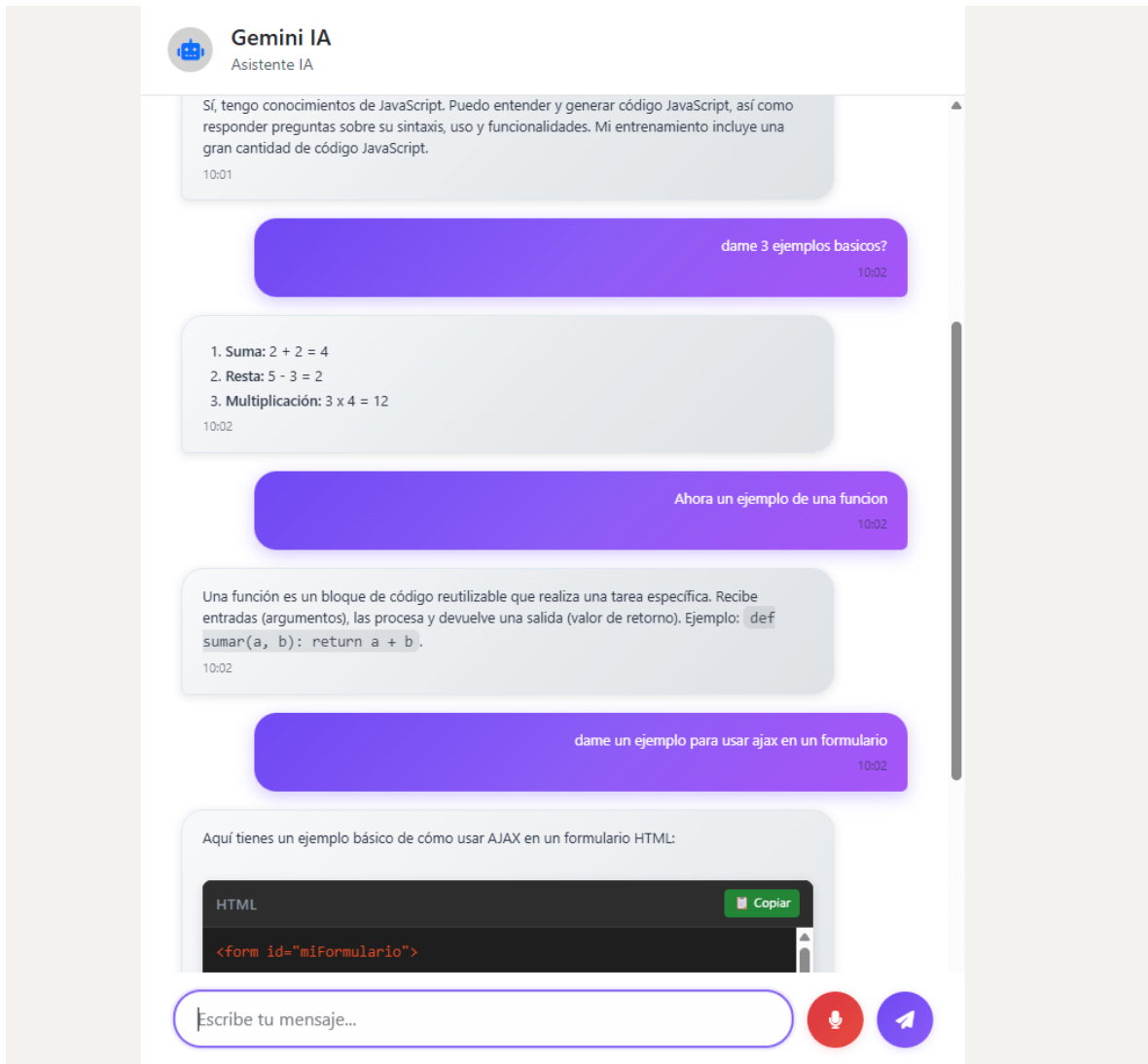
En concreto, buscamos:

- 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.

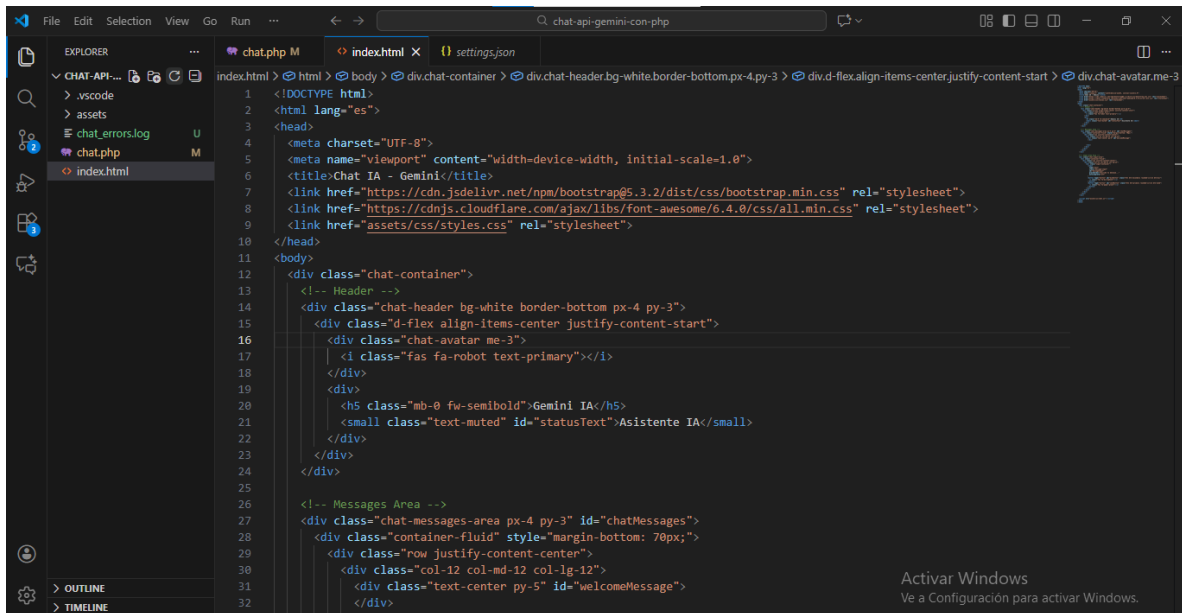
Cronograma de Actividades

| Fase | Actividades Clave | Duración Estimada | Responsable |
|-------------------------|--|-------------------|----------------------|
| 1. Planificación | <ul style="list-style-type: none">- Definir requisitos de la app.- Investigar sobre la API de Gemini.- Establecer diseño inicial. | 2 semanas | Equipo de Proyecto |
| 2. Desarrollo | <ul style="list-style-type: none">- Configurar el entorno de desarrollo.- Integrar la API de Gemini 2.0.- Programar la interfaz de usuario (UI). | 2 semanas | Equipo de Desarrollo |
| 3. Pruebas | <ul style="list-style-type: none">- Probar la aplicación (funcionalidades y errores).- Revisar que las respuestas- Corregir fallos detectados. | 2 semanas | Equipo de Calidad |
| 4. Lanzamiento | <ul style="list-style-type: none">- Preparar la app para usuarios finales. | 1 semana | Equipo de Despliegue |
| 5. Mantenimiento | <ul style="list-style-type: none">- Monitorear el funcionamiento.- Solucionar problemas. | Continuo | Equipo de Soporte |

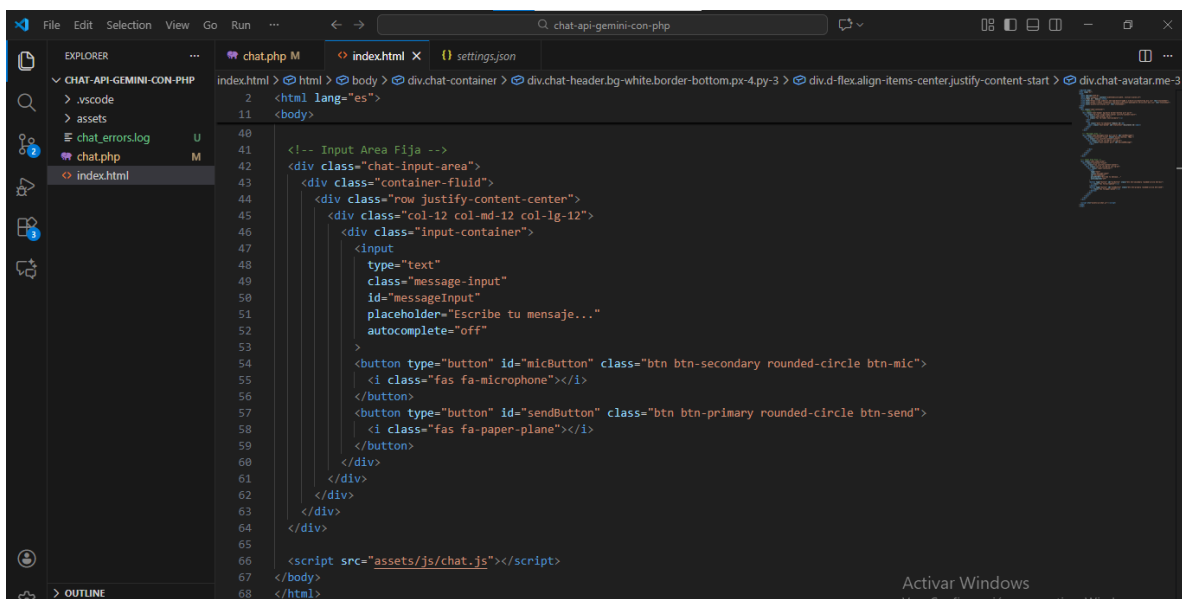
Interfaz de la aplicación



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

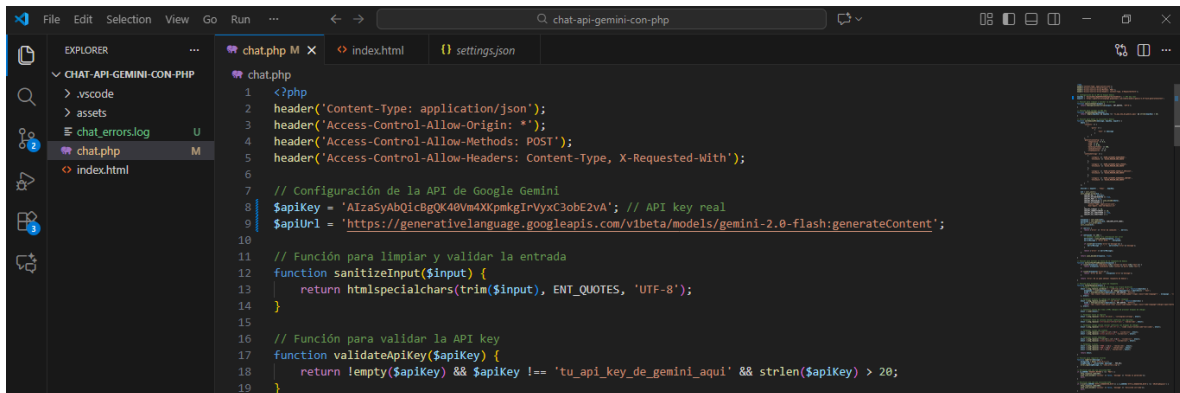
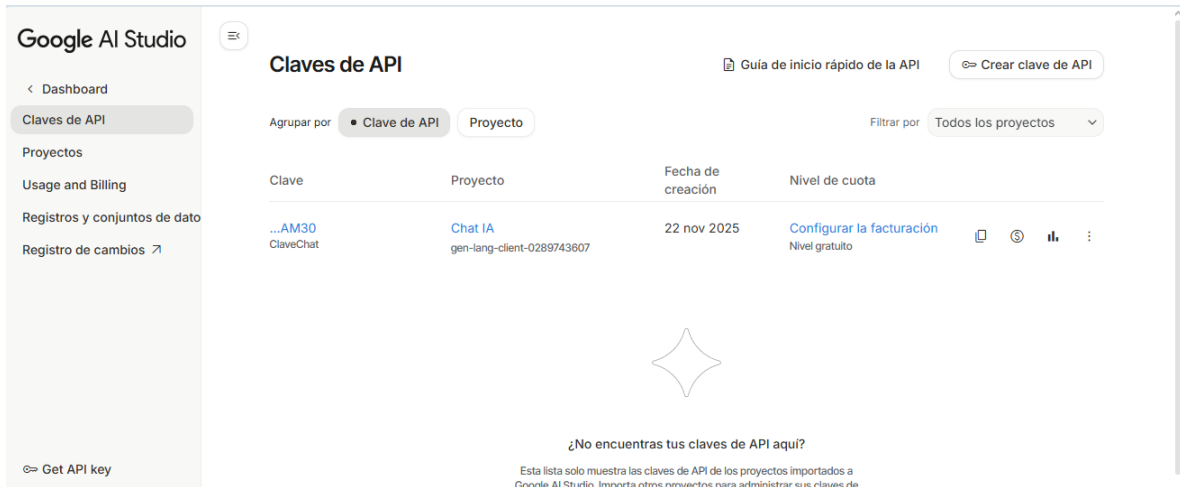


```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Chat IA - Gemini</title>
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
8   <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" rel="stylesheet">
9 </head>
10 <body>
11   <div class="chat-container">
12     <!-- Header -->
13     <div class="chat-header bg-white border-bottom px-4 py-3">
14       <div class="d-flex align-items-center justify-content-start">
15         <div class="chat-avatar me-3">
16           <i class="fas fa-robot text-primary"></i>
17         </div>
18         <div>
19           <h5 class="mb-0 fw-semibold">Gemini IA</h5>
20           <small class="text-muted" id="statusText">Asistente IA</small>
21         </div>
22       </div>
23     </div>
24     <!-- Messages Area -->
25     <div class="chat-messages-area px-4 py-3" id="chatMessages">
26       <div class="container-fluid" style="margin-bottom: 70px;">
27         <div class="row justify-content-center">
28           <div class="col-12 col-md-12 col-lg-12">
29             <div class="text-center py-5" id="welcomeMessage">
30               <div>
31                 <div>
32                   </div>
```



```
33             </div>
34           </div>
35         </div>
36       </div>
37     </div>
38     <script src="assets/js/chat.js"></script>
39   </body>
40 </html>
```

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>

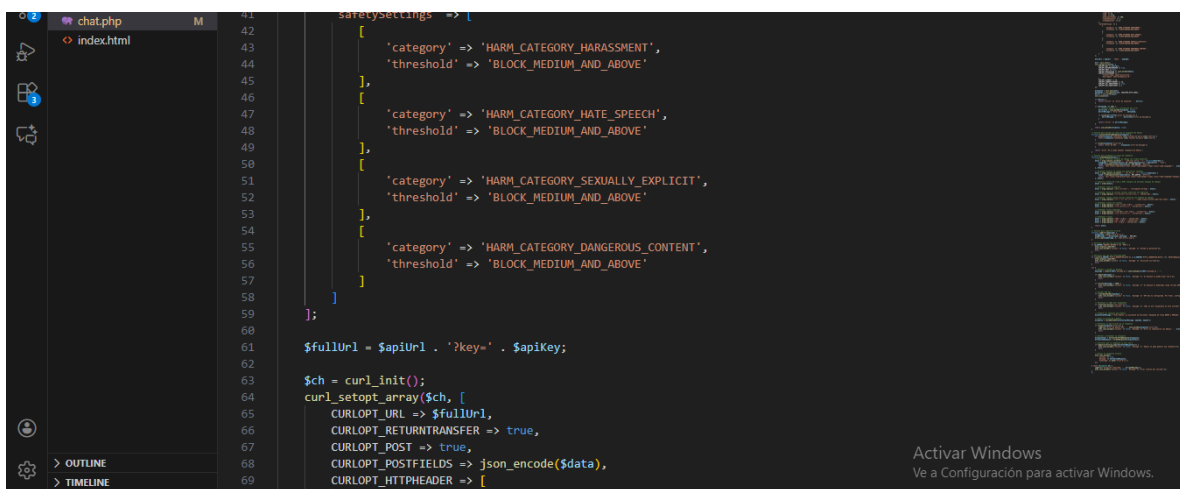


Implementamos la función para validar al API key y la función para solicitar a Gemini



The screenshot shows a code editor with a dark theme. On the left, a sidebar displays a file explorer with 'assets', 'chat_errors.log', 'chat.php', and 'index.html'. The main editor area shows PHP code. Line 16 is a comment: '// Función para validar la API key'. Line 17 is the start of a function: 'function validateApiKey(\$apiKey) {'. Line 18 is a return statement: 'return !empty(\$apiKey) && \$apiKey !== 'tu_api_key_de_gemini_aqui' && strlen(\$apiKey) > 20;'. Line 19 is the closing brace: '}'. Line 21 is a comment: '// Función para hacer la solicitud a Gemini'. Line 22 is the start of a function: 'function callGeminiAPI(\$message, \$apiKey, \$apiUrl) {'. Line 23 is the start of an array: '\$data = ['. Line 24 is the start of a 'contents' array: ' 'contents' => ['. Line 25 is the start of a 'parts' array: ' [']. Line 26 is the start of a 'text' array: ' [']. Line 27 is the assignment: ' 'text' => \$message'. Line 28 is the closing brace for 'text': ' ']. Line 29 is the closing brace for 'parts': ' ']. Line 30 is the closing brace for 'contents': ' ']. Line 31 is the closing brace for '\$data': '].'. Line 32 is the start of a 'generationConfig' array: ' 'generationConfig' => ['. Line 33 is the start of a 'temperature' array: ' 'temperature' => 0.4,. Line 34 is the start of a 'topK' array: ' 'topK' => 25,. Line 35 is the start of a 'topP' array: ' 'topP' => 0.85,. Line 36 is the start of a 'maxOutputTokens' array: ' 'maxOutputTokens' => 400,. Line 37 is the start of a 'candidateCount' array: ' 'candidateCount' => 1,. Line 38 is the start of a 'stopSequences' array: ' 'stopSequences' => []. Line 39 is the closing brace for 'generationConfig': ' ']. Line 40 is the start of a 'safetySettings' array: ' 'safetySettings' => ['. Line 41 is the start of a 'category' array: ' [']. Line 42 is the start of a 'threshold' array: ' 'category' => 'HARM_CATEGORY_HARASSMENT',. Line 43 is the start of a 'threshold' array: ' 'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'. Line 44 is the closing brace for 'category': ' ']. Line 45 is the closing brace for 'safetySettings': ' '].'. The right sidebar shows a 'TIMELINE' view. A watermark 'Activar Windows' is visible in the bottom right corner.

```
15
16 // Función para validar la API key
17 function validateApiKey($apiKey) {
18     return !empty($apiKey) && $apiKey !== 'tu_api_key_de_gemini_aqui' && strlen($apiKey) > 20;
19 }
20
21 // Función para hacer la solicitud a Gemini
22 function callGeminiAPI($message, $apiKey, $apiUrl) {
23     $data = [
24         'contents' => [
25             [
26                 'parts' => [
27                     [
28                         'text' => $message
29                     ]
30                 ]
31             ],
32         ],
33         'generationConfig' => [
34             'temperature' => 0.4,
35             'topK' => 25,
36             'topP' => 0.85,
37             'maxOutputTokens' => 400,
38             'candidateCount' => 1,
39             'stopSequences' => []
40         ],
41         'safetySettings' => [
42             [
43                 'category' => 'HARM_CATEGORY_HARASSMENT',
44                 'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
45             ],
```



The screenshot shows the same code editor as the previous one, but with the continuation of the 'callGeminiAPI' function. Line 46 is the closing brace for the first 'category': ' ']. Line 47 is the start of the second 'category': ' [']. Line 48 is the start of the second 'threshold': ' 'category' => 'HARM_CATEGORY_HATE_SPEECH',. Line 49 is the start of the second 'threshold': ' 'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'. Line 50 is the closing brace for the second 'category': ' ']. Line 51 is the start of the third 'category': ' [']. Line 52 is the start of the third 'threshold': ' 'category' => 'HARM_CATEGORY_SEXUALLY_EXPLICIT',. Line 53 is the start of the third 'threshold': ' 'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'. Line 54 is the closing brace for the third 'category': ' ']. Line 55 is the start of the fourth 'category': ' [']. Line 56 is the start of the fourth 'threshold': ' 'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',. Line 57 is the start of the fourth 'threshold': ' 'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'. Line 58 is the closing brace for the fourth 'category': ' ']. Line 59 is the closing brace for 'safetySettings': ' ']. Line 60 is the closing brace for '\$data': '].'. Line 61 is the assignment: '\$fullUrl = \$apiUrl . '?key=' . \$apiKey;'. Line 62 is the start of a curl array: '\$ch = curl_init();'. Line 63 is the start of a curl_setopt array: 'curl_setopt_array(\$ch, ['. Line 64 is the start of a 'CURLOPT_URL' array: ' CURLOPT_URL => \$fullUrl,. Line 65 is the start of a 'CURLOPT_RETURNTRANSFER' array: ' CURLOPT_RETURNTRANSFER => true,. Line 66 is the start of a 'CURLOPT_POST' array: ' CURLOPT_POST => true,. Line 67 is the start of a 'CURLOPT_POSTFIELDS' array: ' CURLOPT_POSTFIELDS => json_encode(\$data),. Line 68 is the start of a 'CURLOPT_HTTPHEADER' array: ' CURLOPT_HTTPHEADER => ['. Line 69 is the closing brace for 'curl_setopt_array': '].'. The right sidebar shows a 'TIMELINE' view. A watermark 'Activar Windows' is visible in the bottom right corner.

```
41
42
43     'category' => 'HARM_CATEGORY_HARASSMENT',
44     'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
45 ],
46 ],
47     'category' => 'HARM_CATEGORY_HATE_SPEECH',
48     'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
49 ],
50 ],
51     'category' => 'HARM_CATEGORY_SEXUALLY_EXPLICIT',
52     'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
53 ],
54 ],
55     'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
56     'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
57 ],
58 ],
59 ];
60
61 $fullUrl = $apiUrl . '?key=' . $apiKey;
62
63 $ch = curl_init();
64 curl_setopt_array($ch, [
65     CURLOPT_URL => $fullUrl,
66     CURLOPT_RETURNTRANSFER => true,
67     CURLOPT_POST => true,
68     CURLOPT_POSTFIELDS => json_encode($data),
69     CURLOPT_HTTPHEADER => [
```

chat_errors.logU

chat.phpM

index.html

OUTLINE

TIMELINE

```
69     CURLOPT_HTTPHEADER => [
70         'Content-Type: application/json',
71         'User-Agent: Chat-IA-Gemini/1.0'
72     ],
73     CURLOPT_TIMEOUT => 30,
74     CURLOPT_CONNECTTIMEOUT => 10,
75     CURLOPT_SSL_VERIFYPEER => true,
76     CURLOPT_SSL_VERIFYHOST => 2
77 ];
78
79 $response = curl_exec($ch);
80 $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
81 $error = curl_error($ch);
82 curl_close($ch);
83
84 if ($error) {
85     return ['error' => 'Error de conexión: ' . $error];
86 }
87
88 if ($httpCode !== 200) {
89     // Intentar obtener más información del error
90     $errorInfo = json_decode($response, true);
91     $errorMessage = 'Error HTTP: ' . $httpCode;
92
93     if (isset($errorInfo['error']['message'])) {
94         $errorMessage .= ' - ' . $errorInfo['error']['message'];
95     }
96
97     return ['error' => $errorMessage];
98 }
```

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

```
102 // Función para extraer el texto de la respuesta de Gemini
103 function extractTextFromResponse($response) {
104     if (isset($response['candidates'][0]['content']['parts'][0]['text'])) {
105         return $response['candidates'][0]['content']['parts'][0]['text'];
106     }
107
108     if (isset($response['error'])) {
109         return 'Error de API: ' . $response['error']['message'];
110     }
111
112     return 'Error: No se pudo obtener respuesta de Gemini';
113 }
114
115 // Función para formatear el texto de respuesta
116 function formatResponse($text) {
117     // Primero, formatear bloques de código con triple backticks
118     $text = preg_replace_callback('/```(.*?)\n```/s', function($matches) {
119         $language = isset($matches[1]) && !empty($matches[1]) ? $matches[1] : 'text';
120         $code = htmlspecialchars($matches[2], ENT_QUOTES, 'UTF-8');
121         return '<div class="code-block"><div class="code-header"><span class="code-language">' . $language . '</span><bu';
122     }, $text);
123
124     // Formatear bloques de código sin especificar lenguaje
125     $text = preg_replace_callback('/```\n```/s', function($matches) {
126         $code = htmlspecialchars($matches[1], ENT_QUOTES, 'UTF-8');
127         return '<div class="code-block"><div class="code-header"><span class="code-language">código</span><button class="';
128     }, $text);
129
130 }
```

```
131 // Convertir saltos de línea a HTML (después de procesar bloques de código)
132 $text = nl2br($text);
133
134 // Formatear texto en negrita
135 $text = preg_replace('/\*(.*?)\*/', '<strong>$1</strong>', $text);
136
137 // Formatear texto en cursiva (evitar conflicto con negritas)
138 $text = preg_replace('/(?<!\*)\*(?!\/)\*/', '<em>$1</em>', $text);
139
140 // Formatear código inline (evitar conflicto con bloques de código)
141 $text = preg_replace('/(?<!\*)\*(?!\/)/', '<code class="inline-code">$1</code>', $text);
142
143 // Formatear listas con viñetas
144 $text = preg_replace('/^\s*[-+]\s+(.*)$/m', '<li>$1</li>', $text);
145 $text = preg_replace('/<li>.*</li>/s', '<ul>$1</ul>', $text);
146
147 // Formatear listas numeradas
148 $text = preg_replace('/^\s*(\d+)\s+(.*)$/m', '<li>$2</li>', $text);
149 $text = preg_replace('/<li>.*</li>/s', '<ol>$1</ol>', $text);
150
151 // Formatear títulos
152 $text = preg_replace('/^### (.*)$/m', '<h3>$1</h3>', $text);
153 $text = preg_replace('/^## (.*)$/m', '<h2>$1</h2>', $text);
154 $text = preg_replace('/^# (.*)$/m', '<h1>$1</h1>', $text);
155
156 return $text;
157 }
```

```
159 // Función para registrar errores
160 function logError($message) {
161     $timestamp = date('Y-m-d H:i:s');
162     $logMessage = "[{$timestamp}] $message" . PHP_EOL;
163     error_log($logMessage, 3, 'chat_errors.log');
164 }
165
166 // Verificar que sea una solicitud POST
167 if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
168     http_response_code(405);
169     echo json_encode(['success' => false, 'message' => 'Método no permitido']);
170     exit;
171 }
172
173 // Verificar que sea una solicitud AJAX
174 if (!isset($_SERVER['HTTP_X_REQUESTED_WITH']) || $_SERVER['HTTP_X_REQUESTED_WITH'] !== 'XMLHttpRequest') {
175     http_response_code(400);
176     echo json_encode(['success' => false, 'message' => 'Solicitud inválida']);
177     exit;
178 }
```

```
181 // Obtener y validar el mensaje
182 $message = isset($_POST['message']) ? sanitizeInput($_POST['message']) : '';
183
184 if (empty($message)) {
185     echo json_encode(['success' => false, 'message' => 'El mensaje no puede estar vacío']);
186     exit;
187 }
188
189 if (strlen($message) > 4000) {
190     echo json_encode(['success' => false, 'message' => 'El mensaje es demasiado largo (máximo 4000 caracteres)']);
191     exit;
192 }
193
194 // Validar API key
195 if (!validateApiKey($apiKey)) {
196     echo json_encode(['success' => false, 'message' => 'API key no configurada. Por favor, configura tu API key de Gemini']);
197     exit;
198 }
199
200 // Verificar si cURL está disponible
201 if (!function_exists('curl_init')) {
202     echo json_encode(['success' => false, 'message' => 'cURL no está disponible en este servidor']);
203     exit;
204 }
205
206 // Preparar el contexto para Gemini
207 $contextualMessage = "Eres Gemini, un asistente IA eficiente. Responde de forma BREVE y PRECISA. Máximo 3-4 oraciones";
208
```

```
209 // Hacer la solicitud a Gemini
210 $response = callGeminiAPI($contextualMessage, $apiKey, $apiUrl);
211
212 // Verificar si hay errores en la respuesta
213 if (isset($response['error'])) {
214     logError('Error de Gemini API: ' . json_encode($response['error']));
215     echo json_encode(['success' => false, 'message' => 'Error al comunicarse con Gemini: ' . $response['error']]);
216     exit;
217 }
218
219 // Extraer y formatear la respuesta
220 $responseText = extractTextFromResponse($response);
221 $formattedResponse = formatResponse($responseText);
222
223 // Verificar que la respuesta no esté vacía
224 if (empty(trim(strip_tags($formattedResponse)))) {
225     echo json_encode(['success' => false, 'message' => 'Gemini no pudo generar una respuesta']);
226     exit;
227 }
228
```

```
229 // Enviar la respuesta exitosa
230 echo json_encode([
231     'success' => true,
232     'message' => $formattedResponse,
233     'timestamp' => date('Y-m-d H:i:s')
234 ]);
235
236 } catch (Exception $e) {
237     logError('Excepción capturada: ' . $e->getMessage());
238     echo json_encode(['success' => false, 'message' => 'Error interno del servidor']);
239 }
240 ?>
```

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

