



INSTITUTO TECNOLOGICO 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 de forma 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.

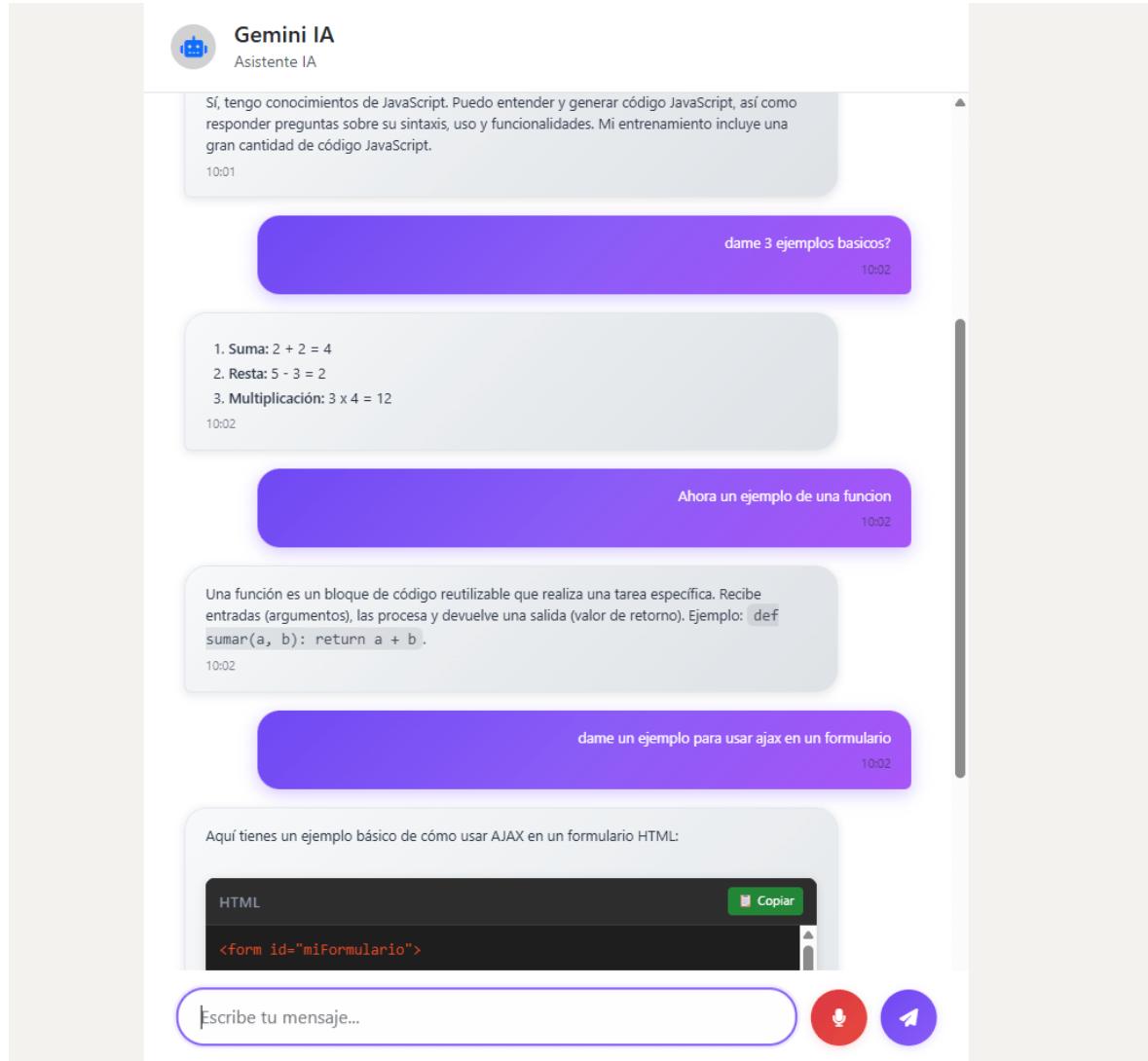
En concreto, buscamos:

- 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.

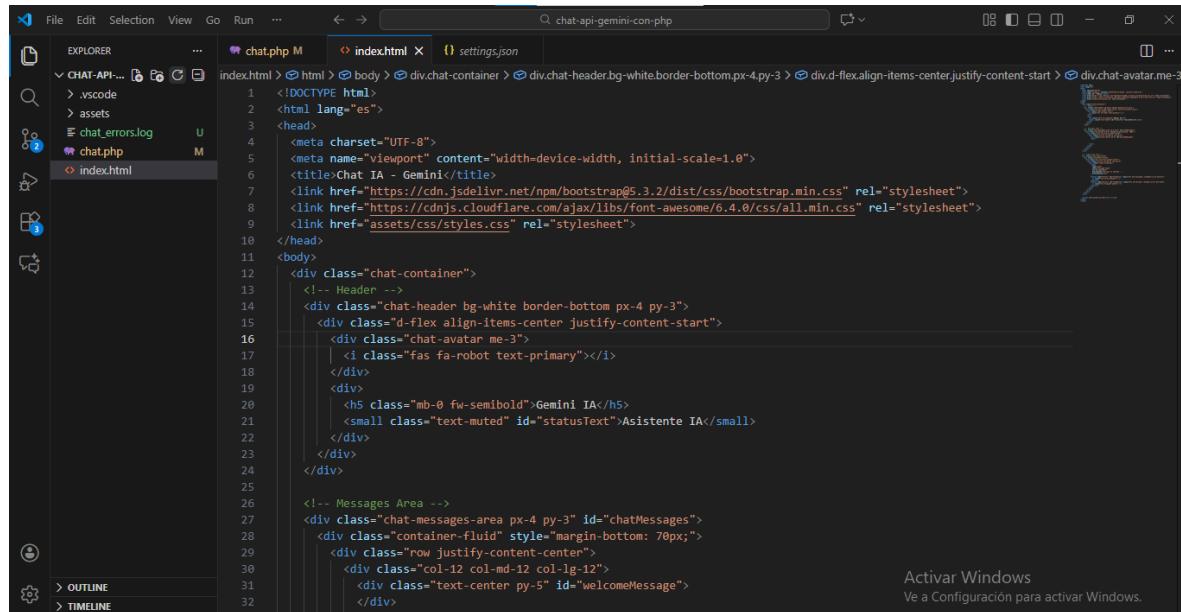
Cronograma de Actividades

Fase	Actividades Clave	Duración Estimada	Responsable
1. Planificación	- Definir requisitos de la app. - Investigar sobre la API de Gemini. - Establecer diseño inicial.	2 semanas	Equipo de Proyecto
2. Desarrollo	- 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	- Probar la aplicación (funcionalidades y errores). - Revisar que las respuestas - 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. - Solucionar problemas.	Continuo	Equipo de Soporte

Interfaz de la aplicación



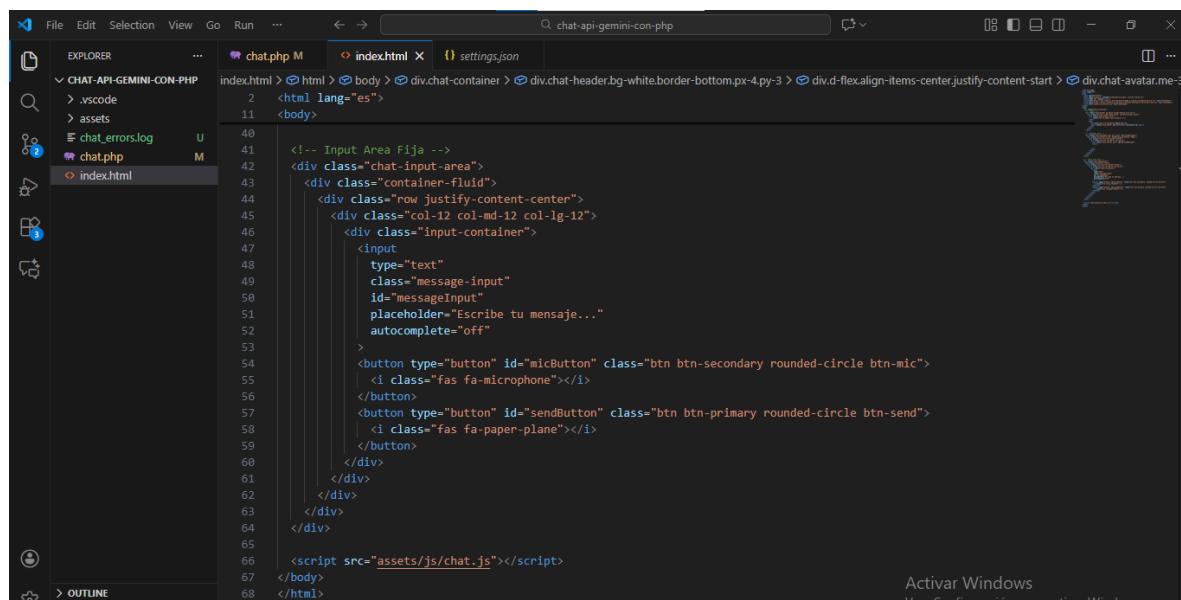
Comenzamos con la elaboración del archivo index.html y posteriormente el diseño



The screenshot shows the Visual Studio Code interface with the 'index.html' file open in the center editor tab. The code is a basic HTML structure for a chat application, including meta tags, a title, and a header section with a logo and user information. The code uses Bootstrap classes like 'd-flex', 'align-items-center', and 'justify-content-start'. The right side of the interface shows a preview of the application's appearance.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Chat IA - Gemini</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" rel="stylesheet">
  <link href="assets/css/styles.css" rel="stylesheet">
</head>
<body>
  <div class="chat-container">
    <!-- Header -->
    <div class="chat-header bg-white border-bottom px-4 py-3">
      <div class="d-flex align-items-center justify-content-start">
        <div class="chat-avatar me-3">
          <i class="fas fa-robot text-primary"></i>
        </div>
        <div>
          <h5 class="mb-0 fw-semibold">Gemini IA</h5>
          <small class="text-muted" id="statusText">Asistente IA</small>
        </div>
      </div>
    </div>
    <!-- Messages Area -->
    <div class="chat-messages-area px-4 py-3" id="chatMessages">
      <div class="container-fluid" style="margin-bottom: 70px;">
        <div class="row justify-content-center">
          <div class="col-12 col-md-12 col-lg-12">
            <div class="text-center py-5" id="welcomeMessage">
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
```

Activar Windows
Ve a Configuración para activar Windows.



The screenshot shows the Visual Studio Code interface with the 'index.html' file open in the center editor tab. The code now includes a new section for a fixed input area at the bottom of the page. This section contains an input field with placeholder text 'Escribe tu mensaje...', a microphone icon button, and a send button with a paper plane icon. The code uses Bootstrap classes like 'row', 'col-12', 'col-md-12', and 'col-lg-12' to layout the components.

```
<!-- Input Area Fija -->
<div class="chat-input-area">
  <div class="container-fluid">
    <div class="row justify-content-center">
      <div class="col-12 col-md-12 col-lg-12">
        <div class="input-container">
          <input type="text" class="message-input" id="messageInput" placeholder="Escribe tu mensaje..." autocomplete="off" />
          <button type="button" id="micButton" class="btn btn-secondary rounded-circle btn-mic">
            <i class="fas fa-microphone"></i>
          </button>
          <button type="button" id="sendButton" class="btn btn-primary rounded-circle btn-send">
            <i class="fas fa-paper-plane"></i>
          </button>
        </div>
      </div>
    </div>
  </div>
</div>
<script src="assets/js/chat.js"></script>
</body>
</html>
```

Activar Windows
Ve a Configuración para activar Windows.

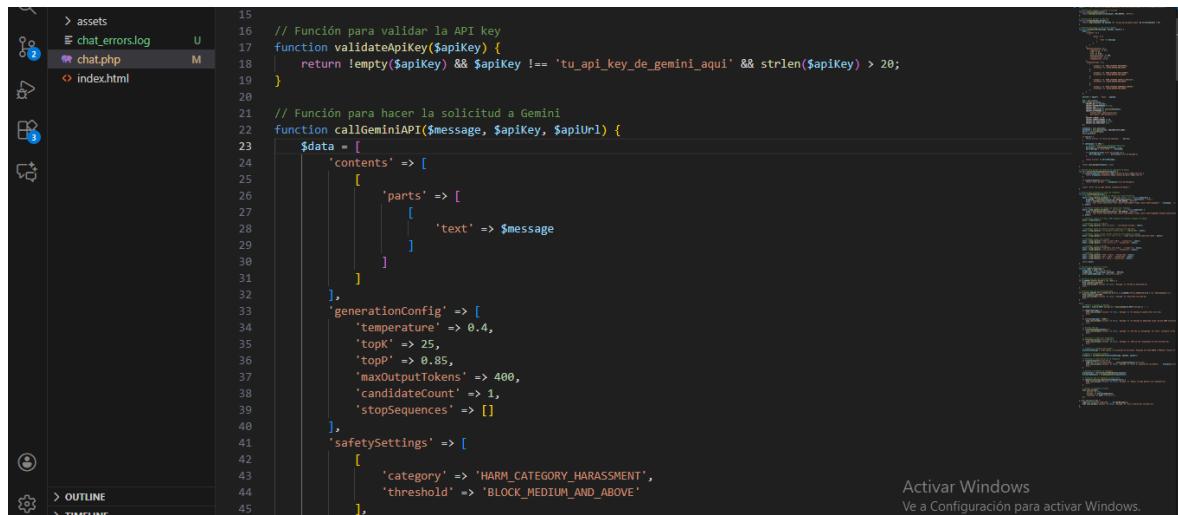
Luego la creación del archivo chat.php luego la configuración de la API de Google Gemini y usaremos la “apiKey” y apiUrl nuestra cuenta de Google IA studio esto lo encontraremos en el apartado de “Get API Key” en la página <https://aistudio.google.com>

The screenshot shows the Google AI Studio interface. On the left, there's a sidebar with options like Dashboard, Claves de API (selected), Proyectos, Usage and Billing, Registros y conjuntos de dato, and Registro de cambios. The main area is titled "Claves de API". It has a search bar with "Guía de inicio rápido de la API" and a button to "Crear clave de API". Below that, there are filters for "Agrupar por" (set to "Clave de API"), "Proyecto" (set to "Todos los proyectos"), and a "Filtrar por" dropdown. A table lists one API key: "...AM30" (ClaveChat) under the project "Chat IA" (gen-lang-client-0289743607). The table columns are "Clave", "Proyecto", "Fecha de creación", and "Nivel de cuota". The "Nivel de cuota" row contains a link to "Configurar la facturación" and the text "Nivel gratuito". At the bottom, there's a message: "¿No encuentras tus claves de API aquí? Esta lista solo muestra las claves de API de los proyectos importados a Google AI Studio. Importa otros proyectos para administrar sus claves de API".

The screenshot shows the VS Code editor with the file "chat.php" open. The code is as follows:

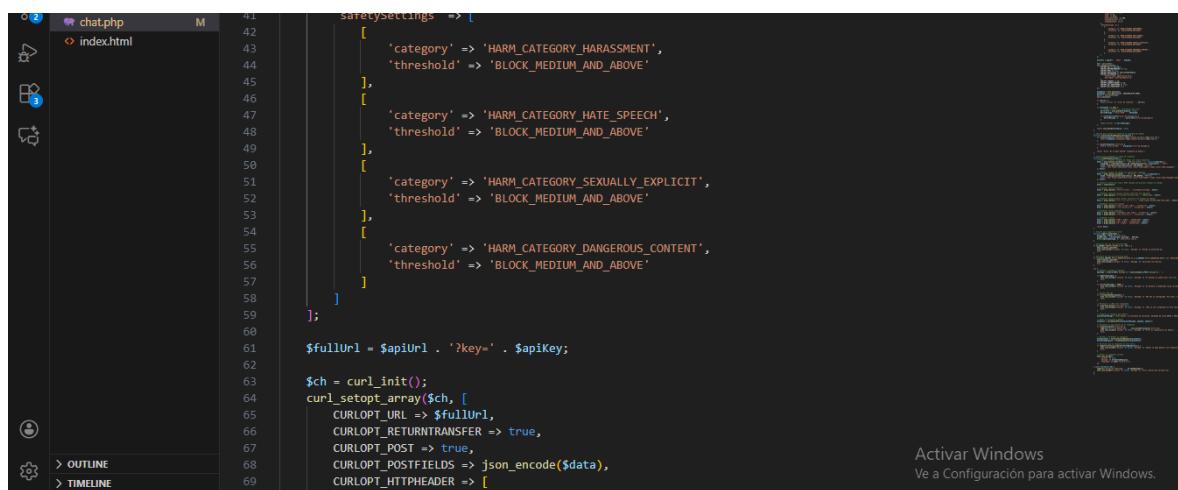
```
File Edit Selection View Go Run ... < - > chat.php M index.html settings.json
chat.php
1 <?php
2 header('Content-Type: application/json');
3 header('Access-Control-Allow-Origin: *');
4 header('Access-Control-Allow-Methods: POST');
5 header('Access-Control-Allow-Headers: Content-Type, X-Requested-With');
6
7 // Configuración de la API de Google Gemini
8 $apiKey = 'AIzaSyAbQicBgQX40Vm4XKpmkgrVyxC3obE2vA'; // API key real
9 $apiUrl = 'https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent';
10
11 // Función para limpiar y validar la entrada
12 function sanitizeInput($input) {
13     return htmlspecialchars(trim($input), ENT_QUOTES, 'UTF-8');
14 }
15
16 // Función para validar la API key
17 function validateApiKey($apiKey) {
18     return !empty($apiKey) && $apiKey !== 'tu_api_key_de_gemini_aqui' && strlen($apiKey) > 20;
19 }
```

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



```
15 // Función para validar la API key
16 function validateApiKey($apiKey) {
17     return !empty($apiKey) && $apiKey !== 'tu_api_key_de_gemini_aqui' && strlen($apiKey) > 20;
18 }
19
20 // Función para hacer la solicitud a Gemini
21 function callGeminiAPI($message, $apiKey, $apiUrl) {
22     $data = [
23         'contents' => [
24             [
25                 'parts' => [
26                     [
27                         'text' => $message
28                     ]
29                 ]
30             ],
31             'generationConfig' => [
32                 'temperature' => 0.4,
33                 'topK' => 25,
34                 'topP' => 0.85,
35                 'maxOutputTokens' => 400,
36                 'candidateCount' => 1,
37                 'stopSequences' => []
38             ],
39             'safetySettings' => [
40                 [
41                     'category' => 'HARM_CATEGORY_HARASSMENT',
42                     'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
43                 ],
44             ],
45         ],
46     ];
47
48     $fullUrl = $apiUrl . '?key=' . $apiKey;
49
50     $ch = curl_init();
51     curl_setopt_array($ch, [
52         CURLOPT_URL => $fullUrl,
53         CURLOPT_RETURNTRANSFER => true,
54         CURLOPT_POST => true,
55         CURLOPT_POSTFIELDS => json_encode($data),
56         CURLOPT_HTTPHEADER => [
57             'Content-Type: application/json'
58         ]
59     ]);
60
61     $response = curl_exec($ch);
62     curl_close($ch);
63
64     if ($response === false) {
65         throw new Exception("Error en la solicitud a Gemini: " . curl_error($ch));
66     }
67
68     return json_decode($response, true);
69 }
```

Activar Windows
Ve a Configuración para activar Windows.



```
41         'category' => 'HARM_CATEGORY_HARASSMENT',
42         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
43     ],
44     [
45         'category' => 'HARM_CATEGORY_HATE_SPEECH',
46         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
47     ],
48     [
49         'category' => 'HARM_CATEGORY_SEXUALLY_EXPLICIT',
50         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
51     ],
52     [
53         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
54         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
55     ],
56     [
57         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
58         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
59     ],
60     [
61         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
62         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
63     ],
64     [
65         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
66         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
67     ],
68     [
69         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
70         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
71     ],
72     [
73         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
74         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
75     ],
76     [
77         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
78         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
79     ],
80     [
81         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
82         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
83     ],
84     [
85         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
86         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
87     ],
88     [
89         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
90         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
91     ],
92     [
93         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
94         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
95     ],
96     [
97         'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
98         'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
99     ],
100    [
101        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
102        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
103    ],
104    [
105        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
106        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
107    ],
108    [
109        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
110        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
111    ],
112    [
113        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
114        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
115    ],
116    [
117        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
118        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
119    ],
120    [
121        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
122        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
123    ],
124    [
125        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
126        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
127    ],
128    [
129        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
130        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
131    ],
132    [
133        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
134        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
135    ],
136    [
137        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
138        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
139    ],
140    [
141        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
142        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
143    ],
144    [
145        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
146        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
147    ],
148    [
149        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
150        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
151    ],
152    [
153        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
154        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
155    ],
156    [
157        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
158        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
159    ],
160    [
161        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
162        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
163    ],
164    [
165        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
166        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
167    ],
168    [
169        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
170        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
171    ],
172    [
173        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
174        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
175    ],
176    [
177        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
178        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
179    ],
180    [
181        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
182        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
183    ],
184    [
185        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
186        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
187    ],
188    [
189        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
190        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
191    ],
192    [
193        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
194        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
195    ],
196    [
197        'category' => 'HARM_CATEGORY_DANGEROUS_CONTENT',
198        'threshold' => 'BLOCK_MEDIUM_AND_ABOVE'
199    ],
199 ];
```

Activar Windows
Ve a Configuración para activar Windows.

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with icons for file operations like Open, Save, and Close, and navigation links for 'OUTLINE' and 'TIMELINE'. The main area displays a PHP script with line numbers from 69 to 98. The code uses the cURL library to make HTTP requests. It includes setting headers, executing the request, and handling errors based on the response code. A tooltip on the right side of the editor window says 'Activar Windows' and 'Ve a Configuración para activar'.

```
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 | }
```

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

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[1], ENT_QUOTES, 'UTF-8');
121         return '<div class="code-block"><div class="code-header"><span class="code-language">' . $language . '</span><button class="copy-button" type="button"><span>Copy</span></button></div><pre>' . $code . '</pre></div>';
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="copy-button" type="button"><span>Copy</span></button></div><pre>' . $code . '</pre></div>';
128     }, $text);
129 }
130
131

```

```

> assets
 (chat_errors.log) U
chat.php M
index.html

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
157 }
158
159

```

```

> assets
 (chat_errors.log) U
chat.php M
index.html

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 }


```

```
// 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 Gemini en https://platform.openai.com/account/api-keys']);
    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 oraciones por respuesta";
```

```
// 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

