

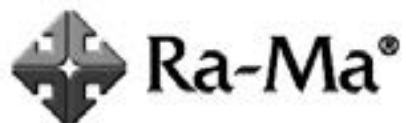
Open AI API

Modelo GPT personalizado

Open AI API

Modelo GPT personalizado

Claudio Bottini





ÍNDICE

PRÓLOGO	7
SOBRE ESTA OBRA	9
SOBRE EL AUTOR	11
CAPÍTULO 1. API DE OPENAI	13
1.1 ¿QUÉ ES LA API DE OPENAI?	14
1.1.1 ¿Cómo se usa la API de OpenAI?	16
1.1.2 Usar lenguaje de marcado de chat (ChatML)	18
1.1.3 Experimentar con la API de OpenAI	18
1.1.4 ¿Y la API de GPT-4?	19

1.1.5	ChatGPT Plus: acceso GPT-4 con un límite de uso	20
1.2	CONCEPTOS IMPORTANTES: PROMPTS, TOKENS, INCRUSTACIONES Y MODELOS	21
1.2.1	Prompts	21
1.2.2	Tokens	21
1.2.3	Incrustaciones (embeddings)	22
1.2.4	Modelos	22
1.3	EL COSTO DE USAR LA API	24
1.3.1	¿Puedo usar la API de OpenAI con la suscripción de ChatGPT Plus? ..	27
1.4	ACTIVIDADES	29
1.4.1	Test de autoevaluación	29
6	OPEN AI API. MODELO GPT PERSONALIZADO	© RA-MA

CAPÍTULO 2. MODOS DE USO.....	31
2.1 ACCESIBILIDAD.....	31
2.2 DISPONIBILIDAD	32
2.3 PRECIOS	33
2.4 USAR LA API DE CHATGPT EN JAVASCRIPT	34
2.4.1 Obtener la API Key	35
2.4.2 El código fuente	39
2.5 USAR LA API DE OPENAI DESDE PHP.....	54
2.6 ACTIVIDADES.....	58
2.6.1 Test de autoevaluación	58
2.6.2 Ejercicios prácticos	59
CAPÍTULO 3. USO AVANZADO CON LIBRERÍAS	61
3.1 LIBRERÍAS.....	61
3.2 COMPLETAR TAREAS CON MODELOS GPT DESDE OPENAI - PHP.....	71
3.2.1 Tarea de finalización de chat (chats completions).....	77
3.3 ACTIVIDADES.....	83
3.3.1 Test de autoevaluación	83
3.3.2 Ejercicios prácticos	84
CAPÍTULO 4. PERSONALIZAR UN MODELO PARA TU USO	85
4.1 AJUSTES FINOS	85
4.1.1 Fine-tuning (la “sintonía fina” de GPT)	86
4.1.2 ¿Qué modelos de GPT se pueden ajustar?.....	87
4.1.3 Parte 1. Generar el archivo de entrenamiento.....	89
4.1.4 Parte 2. Subir el archivo de entrenamiento a OpenAI	93
4.1.5 Parte 3. Iniciar el entrenamiento.....	94
4.1.6 Parte 4. Validación del modelo y pruebas de uso	97
4.1.7 El costo del entrenamiento	102
4.2 ACTIVIDADES	104
4.2.1 Test de autoevaluación	104
4.2.2 Ejercicios prácticos	104

GLOSARIO..... 105

.....

PRÓLOGO

SOBRE ESTA OBRA

SOBRE EL AUTOR

1

API DE OPENAI

1.1 ¿QUÉ ES LA API DE OPENAI?

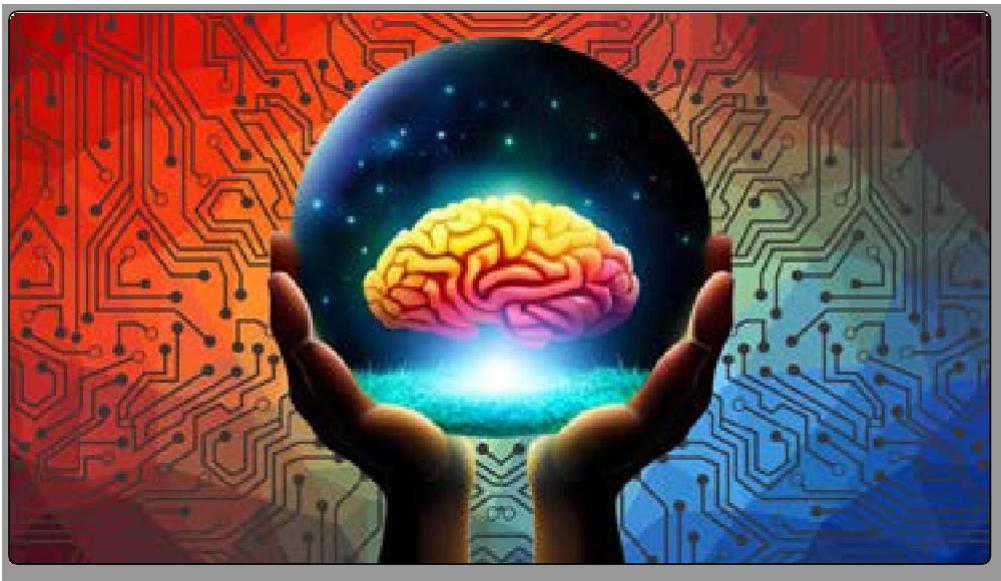


Figura 1.1. Integrar la IA mediante su API, usando uno de varios lenguajes de programación compatibles, es una solución muy simple para entregar nuevas funcionalidades con modelos GPT a usuarios o clientes. Pero, además, es mucho más rápido y económico de lo que la mayoría podría imaginar.

ChatGPT modelos OpenAI prompts API

Figura 1.2. Si bien ambos conceptos se usan de manera indistinta, y se entiende a qué se hace referencia, técnicamente la API es de los modelos de OpenAI, no es la API de ChatGPT, ya que ChatGPT es una implementación en formato de chat conversacional de estos modelos.

endpoint**1.1.1 ¿Cómo se usa la API de OpenAI?****PASO 1****PASO 2****SDK**

PASO 3**PASO 4****PASO 5**

1.1.2 Usar lenguaje de marcado de chat (ChatML)

1.1.3 Experimentar con la API de OpenAI



Figura 1.3. La integración de tecnología de IA mediante una API permite hacer uso de nuevas funcionalidades y potenciar un software existente con las características de los modelos GPT. Es una tarea relativamente simple para un programador en la actualidad.

1.1.4 ¿Y la API de GPT-4?

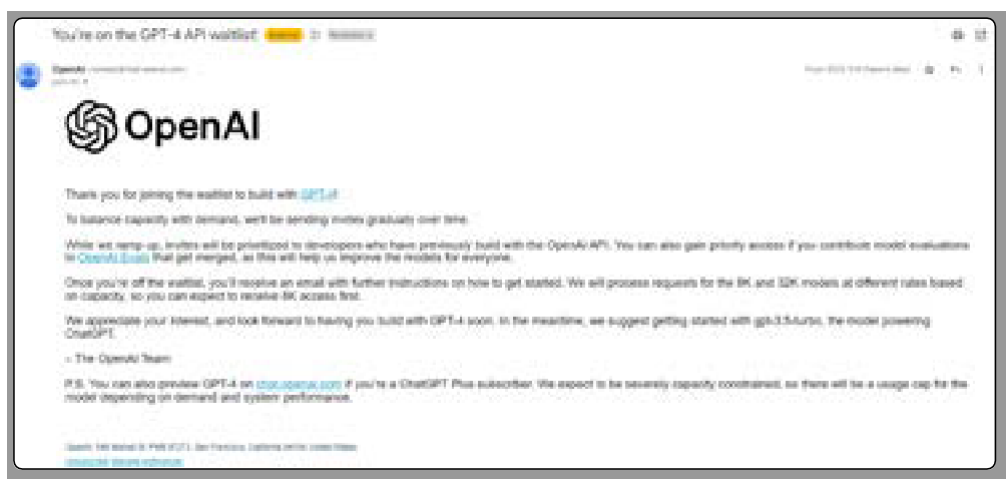


Figura 1.4. Al unirse a la lista de espera, es crucial proporcionar detalles correctos y precisos. OpenAI se basa en esta información para priorizar el acceso y garantizar un proceso de incorporación fluido para todos los usuarios.

1.1.5 ChatGPT Plus: acceso GPT-4 con un límite de uso

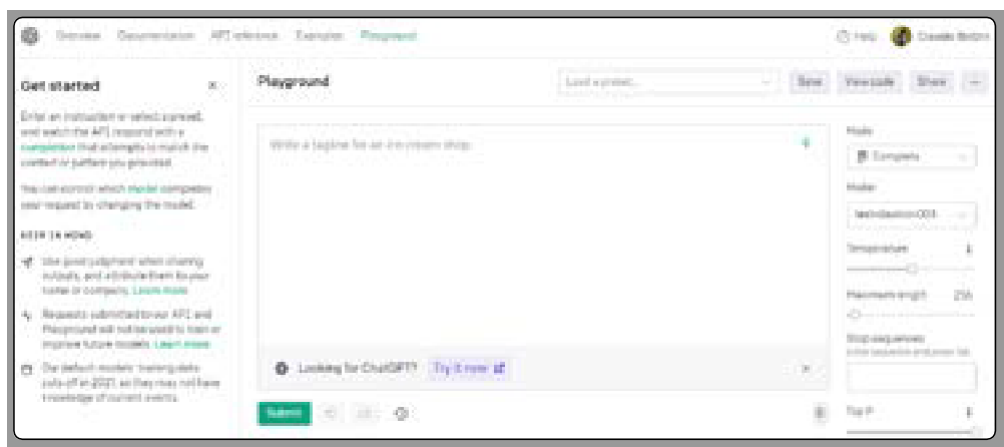


Figura 1.5. Para acceder a GPT-4 Playground, visita el sitio web de OpenAI y ve a la sección de juegos. Allí puedes ingresar indicaciones y observar las respuestas de GPT-4. Aunque GPT-4 Playground no ofrece el mismo nivel de personalización e integración que la API, es una herramienta valiosa para echar un vistazo a las capacidades de GPT-4 y generar texto para uso personal.

1.2 CONCEPTOS IMPORTANTES: PROMPTS, TOKENS, INCRUSTACIONES Y MODELOS

1.2.1 Prompts

1.2.2 Tokens

tokens

1.2.3 Incrustaciones (embeddings)

incrustaciones

1.2.4 Modelos

1.2.4.1 ELEGIR EL MODELO DE LENGUAJE ADECUADO

Serie del modelo	Características
GPT-4 (en beta limitada)	Conjunto de modelos que mejoran GPT-3.5 y pueden comprender y generar lenguaje natural o código.
GPT-3.5	Conjunto de modelos que mejoran GPT-3 y pueden comprender y generar lenguaje natural o código.
DALL·E (en beta libre)	Modelo que puede generar y editar imágenes con un mensaje de lenguaje natural como entrada.
Whisper (en beta libre)	Modelo que puede convertir un audio en su transcripción en texto.
Embeddings	Conjunto de modelos que pueden convertir texto en una forma numérica.
Moderation	Modelo perfeccionado que puede detectar si el texto puede ser confidencial o indebido de acuerdo con las condiciones de uso de la API.
GPT-3	Conjunto de modelos que pueden comprender y generar lenguaje natural.

Modelo	Características	Máximo de tokens
gpt-3.5-turbo	El modelo GPT-3.5 más capaz y optimizado para chat a 1/10 del costo de text-davinci-003.	4096
gpt-3.5-turbo-16k	Mismas capacidades que el gpt-3.5-turbomodelo estándar pero con 4 veces el contexto.	16384
texto-davinci-003	Puede realizar cualquier tarea de lenguaje con mejor calidad, resultados más prolongados y seguimiento de instrucciones constante que los modelos curie, babbage o ada. También admite algunas funciones adicionales, como la inserción de texto.	4097
texto-davinci-002	Capacidades similares a text-davinci-003pero entrenadas con ajuste fino supervisado en lugar de aprendizaje por refuerzo.	4097
código-davinci-002	Optimizado para tareas de finalización de código.	8001

text-davinci-002	Este es el modelo GPT-3 más capaz, pero también el más caro y tiene la latencia más alta. Es mejor para aplicaciones que requieren comprensión y rendimiento avanzados.
text-curie-002	Este modelo GPT-3 es un buen equilibrio entre capacidad y costo. Es adecuado para la mayoría de los casos de uso.
text-babbage-002	Este modelo GPT-3 es menos costoso que curie y davinci pero aun así ofrece un buen rendimiento. Es una buena opción para proyectos con presupuesto limitado.
text-ada-002	Este modelo GPT-3 es el menos costoso y tiene la latencia más baja, pero también es el menos capaz. Es mejor para aplicaciones que no requieren una comprensión del lenguaje muy sofisticada.

1.3 EL COSTO DE USAR LA API



Figura 1.6. La estructura de precios de la API de ChatGPT está diseñada para adaptarse a una amplia gama de necesidades de los desarrolladores, lo que garantiza que siga siendo accesible y rentable. A continuación se revisan algunos conceptos importantes sobre los precios de la API de ChatGPT.



Usage limits

Manage your spending by configuring usage limits. Notification emails triggered by reaching these limits will be sent to members of your organization with the **Owner** role.

There may be a delay in enforcing any limits, and you are responsible for any overage incurred. We recommend checking your usage tracking dashboard regularly to monitor your spend.

Approved usage limit
The maximum usage OpenAI allows for your organization each month: [Request increase](#)
\$120.00

Current usage
Your total usage so far in June (UTC). Note that this may include usage covered by a free trial or other credits, so your monthly bill might be less than the value shown here. [View usage records](#)
\$0.00

Hard limit
When your organization reaches this usage threshold each month, subsequent requests will be rejected.

Figura 1.7. Límites de uso.





Figura 1.8. Detalle de consumos diarios.

1.3.1 ¿Puedo usar la API de OpenAI con la suscripción de ChatGPT Plus?

1.3.1.1 ÚLTIMAS ACTUALIZACIONES DE LA API PARA DESARROLLADORES ventana de contexto

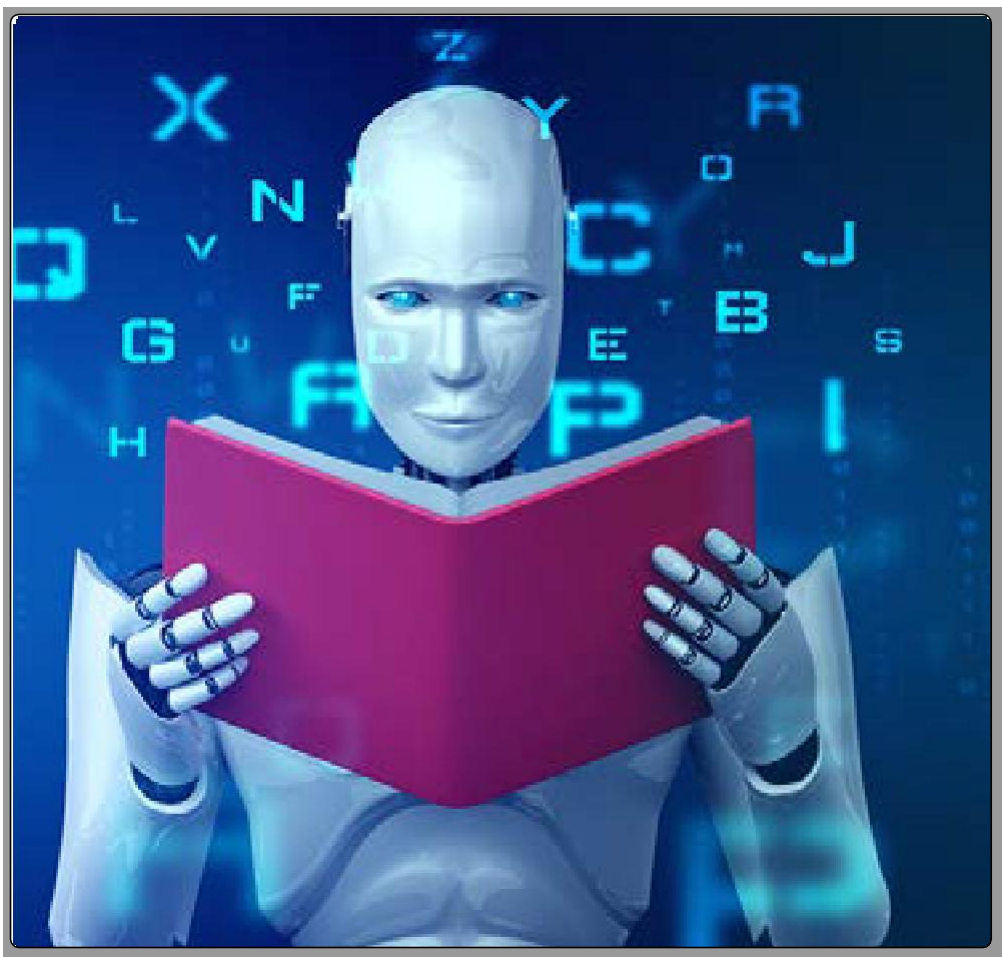


Figura 1.9. En los modelos de lenguaje grande (LLM), la “ventana de contexto” es como una memoria a corto plazo que almacena el contenido del prompt original o, en el caso de un chatbot, todo el contenido de la conversación en curso. En los modelos de lenguaje, aumentar el tamaño del contexto se ha convertido en una carrera tecnológica. OpenAI ha desarrollado una versión de 32.000 tokens de GPT-4, pero aún no está disponible públicamente.

JSON

1.4 ACTIVIDADES

1.4.1 Test de autoevaluación

2

MODOS DE USO

2.1 ACCESIBILIDAD

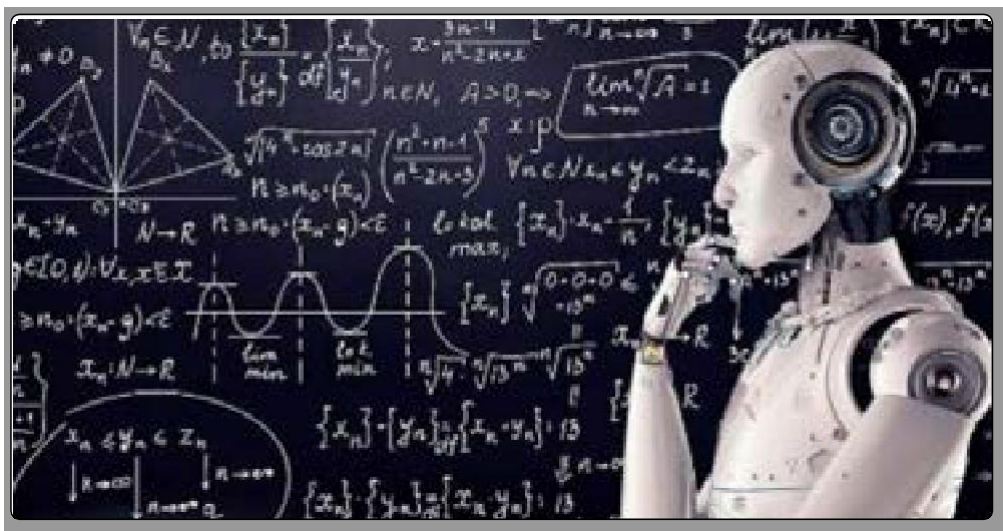


Figura 2.1. OpenAI API es una interfaz de programación de aplicaciones (API) proporcionada por OpenAI que permite a los desarrolladores acceder a las capacidades de procesamiento de lenguaje natural de los modelos GPT. Es una poderosa herramienta que permite crear aplicaciones conversacionales con capacidades de procesamiento de lenguaje natural.



Figura 2.2. En muchos casos, se hace a un lado la posibilidad de usar la API pensando en que es algo caro y complejo, y en otras no se avanza con el pago de la versión Plus del Chat Web pensando que, con usar la API, automáticamente tendrás solucionados todos tus problemas.

2.2 DISPONIBILIDAD

.....

Rate limits		
OpenAI enforces rate limits on the requests you can make to the API. These are applied over requests-per-minute, tokens-per-minute, or in the case of image models, images-per-minute.		
Learn more in the rate limits documentation , or reference the default rate limits for our models. Your organization's rate limits are listed below.		
MODEL	RPM	TPM
CHAT		
gpt-3.5-turbo	3,500	90,000
gpt-3.5-turbo-0301	3,500	90,000
gpt-3.5-turbo-0613	3,000	250,000
gpt-3.5-turbo-16k	3,500	180,000
gpt-3.5-turbo-16k-0613	3,500	180,000
TEXT		
ada	3,000	250,000
ada-code-search-code	3,000	250,000

Figura 2.3. Límites.

2.3 PRECIOS

	Desarrollo de aplicaciones	Personalización y control	Planes y estructura de precios	Público objetivo
ChatGPT Plus	Diseñado para uso individual a través de la plataforma OpenAI.	Personalización limitada, centrada principalmente en la experiencia del usuario.	La cuota de suscripción plana, más adecuada para usuarios individuales.	Individuos que quieren usar GPT para realizar ciertas tareas.
API de ChatGPT	Diseñado para la integración con aplicaciones externas.	Ofrece más opciones de control y personalización.	Precios basados en el uso, adecuados para desarrolladores y empresas.	Desarrolladores de aplicaciones/software y empresas.

2.4 USAR LA API DE CHATGPT EN JAVASCRIPT

Figura 2.4. La API de ChatGPT se puede integrar con una amplia gama de frameworks y lenguajes



de programación, incluido JavaScript. En este capítulo nos introduciremos en lo esencial para usar la API de ChatGPT desde JavaScript para luego hacer uso desde otros lenguajes de servidor.

2.4.1 Obtener la API Key

API key

PASO 1

Create your account

Please note that phone verification is required for signup. Your number will only be used to verify your identity for security purposes.

Email address

Continue

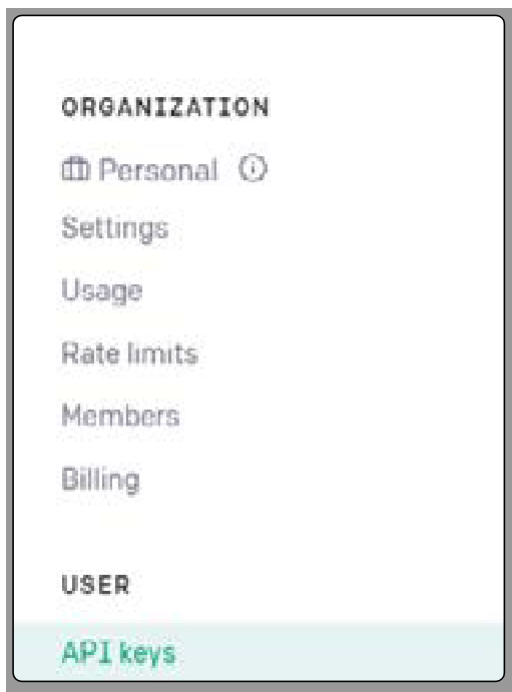
Already have an account? [Log in](#)

OR

 Continue with Google

 Continue with Microsoft Account

 Continue with Apple

PASO 2**PASO 3**

API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically rotate any API key that we've found has leaked publicly.

You currently do not have any API keys. Please create one below.

[+ Create new secret key](#)

Default organization

If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.

Personal

Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.

PASO 4

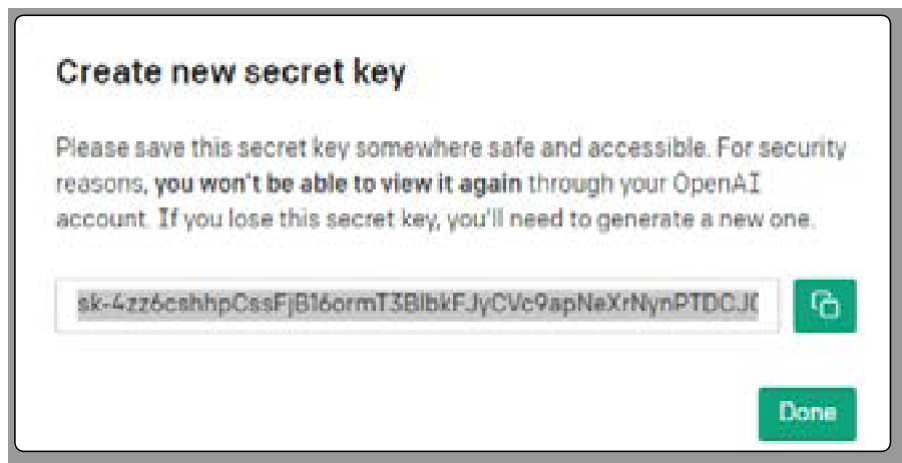
Create new secret key

Name Optional

KeyPHP

Cancel

Create secret key

PASO 5**PASO 6**

NAME	KEY	CREATED	LAST USED	
Key PHP	sk-...DCJB	27 de may de 2023	Never	 
+ Create new secret key				

2.4.2 El código fuente

```
<html>  
<head>  
  <title>JS-GPT</title>  
  <style>  
    body {  
      font-family: Arial, sans-serif;  
      margin: 0;  
      padding: 0;  
      background-color: #f2f2f2;  
    }  
  
    #idContainer {  
      max-width: 800px;  
      margin: 0 auto;  
      padding: 20px;  
      background-color: #fff;
```

```
        border-radius: 5px;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }

    textarea {
        border: 1px solid #ccc;
        border-radius: 5px;
        padding: 10px;
        resize: none;
        width: 100%;
        box-sizing: border-box;
    }

    button {
        padding: 8px 12px;
        background-color: #007bff;
        color: #fff;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }

    label {
        margin-left: 10px;
    }

    select {
        padding: 5px;
        border-radius: 5px;
    }

    #idText {
        margin-top: 10px;
        color: #999;
    }
</style>
</head>
```

OnLoad()

```
<body onload="OnLoad()">
```

```
<div id="idContainer">
  <!-- Área de salida de mensajes -->
  <textarea id="txtOutput" rows="10" placeholder="Chat..."></textarea>

  <div>
    <!-- Botón de enviar -->
    <button type="button" onclick="Send()" id="btnSend">Enviar</button>

    <!-- Opciones de escucha y silencio -->
    <label id="lblSpeak">
      <input id="chkSpeak" type="checkbox" onclick="SpeechToText()" />
      Escuchar
    </label>
    <label id="lblMute">
      <input id="chkMute" type="checkbox" onclick="Mute(this.checked)"
/>
      Mutear
    </label>

    <!-- Selección del modelo de lenguaje -->
    <select id="selModel">
      <option value="gpt-3.5-turbo">gpt-3.5-turbo</option>
      <option value="gpt-3.5-turbo-0301">gpt-3.5-turbo-0301</option>
      <option value="gpt-3.5-turbo-16k">gpt-3.5-turbo-16k</option>
      <option value="text-davinci-003">text-davinci-003</option>
      <option value="text-davinci-002">text-davinci-002</option>
      <option value="text-ada-001">text-ada-001</option>

    </select>

    <!-- Selección del idioma -->
    <select id="selLang" onchange="ChangeLang(this)">
      <option value="en-US">Ingles (EEUU)</option>
      <option value="fr-FR">Frances </option>
      <option value="es-ES">Español (España)</option>
      <option value="it-IT">Italiano </option>

    </select>
```

```

        <select id="selVoices"></select>
        <span id="spMsg"></span>
    </div>

    <!-- Área de entrada de mensajes -->
    <textarea id="txtMsg" rows="5" placeholder="Escribe aquí"></textarea>

    <!-- Historial de mensajes -->
    <div id="idText"></div>
</div>

```

```
<script src="openAiApi.js"></script>
```



Figura 2.5. La interfaz ya está armada y tiene este aspecto en cualquier navegador actual.

▀ **webkitSpeechRecognition speechSynthesis lb1Speak**
 // Función que se ejecuta cuando se carga la página

```
function OnLoad() {
    // Verificar si el navegador admite el reconocimiento de voz
    (speech-to-text)
    if ("webkitSpeechRecognition" in window) {
        // El navegador admite el reconocimiento de voz
    } else {
        // El reconocimiento de voz no es compatible
        // Ocultar elementos relacionados con el reconocimiento de voz
        lblSpeak.style.display = "none";
    }

    // Verificar si el navegador admite la síntesis de voz (text-to-
    speech)
    if ('speechSynthesis' in window) {
        // El navegador admite la síntesis de voz
        bTextToSpeechSupported = true;

        // Obtener las voces disponibles cuando cambia el evento on-
        voiceschanged
        speechSynthesis.onvoiceschanged = function () {
            oVoices = window.speechSynthesis.getVoices();
            for (var i = 0; i < oVoices.length; i++) {
                selVoices[selVoices.length] = new Option(oVoices[i].
name, i);
            }
        };
    }
}

// Función para cambiar el idioma del reconocimiento de voz
function ChangeLang(o) {
    if (oSpeechRecognizer) {
        oSpeechRecognizer.lang = selLang.value;
        // Llamar a la función SpeechToText() para aplicar el cambio
        de idioma
    }
}
```

▼ **selLang oSpeechRecognizer**

```
// Función para convertir texto a voz
function TextToSpeech(s) {
    if (bTextToSpeechSupported == false) {
        return;
    }
    if (chkMute.checked) {
        return;
    }

    oSpeechSynthesisUtterance = new SpeechSynthesisUtterance();

    if (oVoices) {
        var sVoice = selVoices.value;
        if (sVoice != "") {
            oSpeechSynthesisUtterance.voice =
oVoices[parseInt(sVoice)];
        }
    }

    oSpeechSynthesisUtterance.onend = function () {
        if (oSpeechRecognizer && chkSpeak.checked) {
            oSpeechRecognizer.start();
        }
    };

    if (oSpeechRecognizer && chkSpeak.checked) {
        oSpeechRecognizer.stop();
    }
}
```

▀ txtMsg ▀ chkMute


```
oSpeechSynthesisUtterance.lang = sellLang.value;
oSpeechSynthesisUtterance.text = s;

window.speechSynthesis.speak(oSpeechSynthesisUtterance);
}
```

```
// Función para activar o desactivar el modo silencio (mute)
function Mute(b) {
    if (b) {
        selVoices.style.display = "none";
    } else {
        selVoices.style.display = "";
    }
}
```

▀ chkMute

```
// Función para realizar el reconocimiento de voz (speech-to-text)
function SpeechToText() {
    if (oSpeechRecognizer) {
        if (chkSpeak.checked) {
            oSpeechRecognizer.start();
        } else {
            oSpeechRecognizer.stop();
        }
    }
}
```

▀ webkitSpeechRecognition

```
// Función para iniciar el reconocimiento de voz
function StartSpeechRecognition() {
    if (bTextToSpeechSupported) {
        oSpeechRecognizer = new webkitSpeechRecognition();
        oSpeechRecognizer.continuous = true;
        oSpeechRecognizer.interimResults = false;
        oSpeechRecognizer.lang = selLang.value;

        oSpeechRecognizer.onstart = function () {
            bSpeechInProgress = true;
        };

        oSpeechRecognizer.onend = function () {
            bSpeechInProgress = false;
            if (chkSpeak.checked) {
                oSpeechRecognizer.start();
            }
        };

        oSpeechRecognizer.onerror = function (event) {
            alert("Error en la funcion de comprension: " + event.error);
        };

        oSpeechRecognizer.onresult = function (event) {
            var i = event.resultIndex;
            var sTranscript = event.results[i][0].transcript;
            txtMsg.value += sTranscript;
        };
    }
}
```

```
// Función para enviar la pregunta y obtener una respuesta de la API
function Send() {
    // Obtener la pregunta ingresada por el usuario
    var sQuestion = txtMsg.value;
    if (sQuestion == "") {
        alert("Ingresa tu mensaje");
    }
}
```

Send()

```
txtMsg.focus();  
return;  
}
```

```
snMsg.innerHTML = "GPT esta pensando...":  
var sUrl = "https://api.openai.com/v1/completions";  
var sModel = selModel.value;
```

```
// Verificar si se está utilizando el modelo GPT-3.5-turbo para ajustar la  
var oHttp = new XMLHttpRequest();  
oHttp.open("POST", sUrl);  
oHttp.setRequestHeader("Accept", "application/json");  
oHttp.setRequestHeader("Content-Type", "application/json");  
oHttp.setRequestHeader("Authorization", "Bearer " + OPENAI_API_KEY);
```

```
oHttp.onreadystatechange = function () {  
    // Esta función se ejecuta cuando se recibe una respuesta del servidor  
    //El valor 4 representa el estado "DONE" (realizado), lo que significa que la  
    solicitud ha sido completada y la respuesta del servidor ha sido recibida co-
```

```

rrectamente
    if (oHttp.readyState === 4) {
        spMsg.innerHTML = "";

        var oJson = {};
        if (txtOutput.value !== "") {
            txtOutput.value += "\n";
        }

        try {
            oJson = JSON.parse(oHttp.responseText);
        } catch (ex) {
            // Si ocurre un error al analizar la respuesta JSON, mostrar el
error en el área de salida
            txtOutput.value += "Error: " + ex.message;
        }

        if (oJson.error && oJson.error.message) {
            // Si la respuesta contiene un mensaje de error, mostrarlo en el
área de salida
            txtOutput.value += "Error: " + oJson.error.message;
        } else if (oJson.choices) {
            // Si la respuesta contiene opciones de respuesta

            var s = "";

            // Si el texto de la respuesta está disponible, asignarlo a
la variable "s"
            s = oJson.choices[0].text;
        } else if (oJson.choices[0].message) {
            // Si el mensaje de la respuesta está disponible, asignar su
contenido a la variable "s"
            s = oJson.choices[0].message.content;
        }

        if (sellang.value !== "en-US") {
            // Verificar si el idioma seleccionado no es inglés (Estados
Unidos)

            var a = s.split("?\\n");
            if (a.length == 2) {
                // Si el texto de la respuesta contiene un separador
"?\\n", asignar la segunda parte a la variable "s"
                s = a[1];
            }
        }
    }
}

```

```

        }
    }

    if (s == "") {
        // Si no hay respuesta, asignar "Sin respuesta" a la variable "s"
        s = "Sin respuesta";
    } else {
        // Si hay una respuesta, agregarla al área de salida y convertirla a voz
        txtOutput.value += "GPT: " + s;
        TextToSpeech(s);
    }
}
}
};

```

```

{
  "id": "chatcmpl-abc123",
  "object": "chat.completion",
  "created": 1677858242,
  "model": "gpt-3.5-turbo-0301",
  "usage": {
    "prompt_tokens": 13,
    "completion_tokens": 7,
    "total_tokens": 20
  },
  "choices": [
    {
      "message": {
        "role": "assistant",
        "content": "\n\nEsta es una respuesta"
      },
      "finish_reason": "stop",
      "index": 0
    }
  ]
}

```

▀ ▀ ▀ ▀ ▀ **prompt_tokens completion_tokens total_tokens**

```
[choices] => Array
  (
    [0] => Array
      (
        [index] => 0
        [message] => Array
          (
            [role] => assistant
            [content] => MENSAJE
          )
      )
  )
```

total_tokens total_tokens

```

        [finish_reason] => stop
    )
)
rol
}

try {
    oJson = JSON.parse(oHttp.responseText);
    console.log(JSON.stringify(oJson, null, 4));
} catch (ex) {
    // Si ocurre un error al analizar la respuesta JSON, mostrar el
    error en el área de salida
    txtOutput.value += "Error: " + ex.message;
    console.log("Error: " + ex.message);
}

```



Figura 2.6. Respuesta recibida.

```

var iMaxTokens = 30;
var sUserId = "1";
var dTemperature = 0.5;

var data = {
  model: sModel,
  prompt: sQuestion,
  max_tokens: iMaxTokens,
  user: sUserId,
  temperature: dTemperature,
  frequency_penalty: 0.0,
  presence_penalty: 0.0,
  stop: ["#", ";"]
};

```

Send()     **sModel sQuestion**

```

// Verificar si se está utilizando el modelo GPT-3.5-turbo para ajustar la es-
oHttp.send(JSON.stringify(data));

if (txtOutput.value != "") {
  txtOutput.value += "\n";
}
txtOutput.value += "Me: " + sQuestion;
      "role": "user",
      "content": sQuestion
    }
  ]
};
}

```



```
txtMsg.value = "";
```

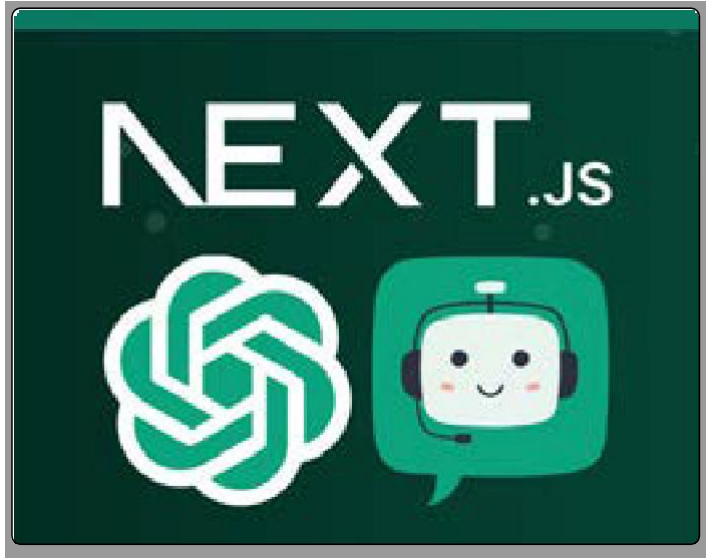


Figura 2.7. Alternativamente, si manejas un entorno con Node, puedes usar los clientes de Node.js disponibles para la API oficial de ChatGPT. Estos clientes proporcionan un framework ya creado alrededor de la API de OpenAI usando Next.js.

2.5 USAR LA API DE OPENAI DESDE PHP

.....



cURL

PASO 1

PASO 2

```

cbottin@ESCRITCHID-CLAUDIO:~$
cbottin@ESCRITCHID-CLAUDIO:~$ curl --help
Usage: curl [options...] <url>
  -d, --data <data>           HTTP POST data
  -f, --fail                   Fail silently (no output at all) on HTTP errors
  -h, --help <category>       Get help for commands
  -i, --include                 Include protocol response headers in the output
  -o, --output <file>          Write to file instead of stdout
  -O, --remote-name             Write output to a file named as the remote file
  -s, --silent                  Silent mode
  -T, --upload-file <file>     Transfer local FILE to destination
  -u, --user <user:password>   Server user and password
  -A, --user-agent <name>      Send User-Agent <name> to server
  -v, --verbose                 Make the operation more talkative
  -V, --version                 Show version number and quit

This is not the full help, this menu is stripped into categories.
Use '--help category' to get an overview of all categories.
For all options use the manual or '--help all'.
cbottin@ESCRITCHID-CLAUDIO:~$

```

```
sudo apt install curl
```

```
$curl = curl_init(); // iniciamos una sesion de CURL
```

```

$APT KEY = "sk-xxxxxxxxxxxxxxxx":
$url = "https://api.openai.com/v1/completions";
curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_POST, true);

```

```

$headers = array(
    "Content-Type: application/json",
    "Authorization: Bearer $API_KEY"
);
curl_setopt($curl, CURLOPT_HTTPHEADER, $headers);

```

Content-Type Authorization

```

$data = array(
    "model" => "text-davinci-003",
    "prompt" => "Como esta el clima en Mexico hoy?",
    "temperature" => 0.5,
    "max_tokens" => 100 );
curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($data));

```

temperatura
echo

```
$response = curl_exec($curl);  
$response = json_decode($response, true)  
  
$generated_text = $response['choices'][0]['text'];  
echo $generated_text;  
  
curl_close($curl);
```

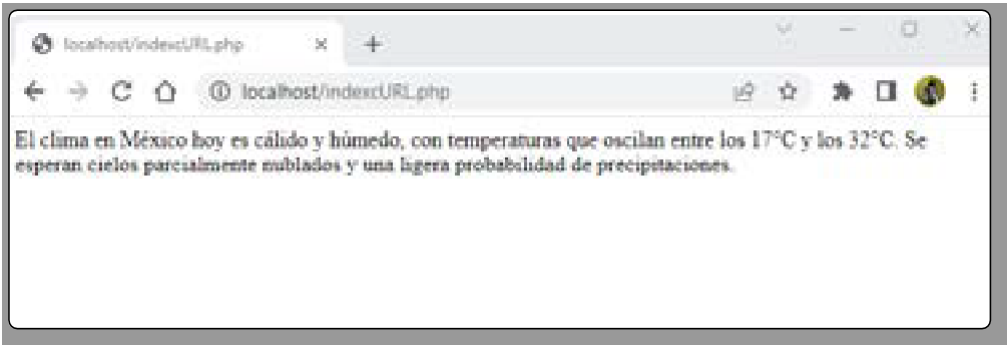


Figura 2.8. Ya has logrado usar la API de OpenAI desde un lenguaje de servidor como PHP, con lo cual tu API Key no quedará expuesta. Pero además puedes hacer uso de la potencia del lenguaje para dar más funcionalidades y complejidad a su uso.

2.6 ACTIVIDADES

.....

2.6.1 Test de autoevaluación

2.6.2 Ejercicios prácticos

3

USO AVANZADO CON LIBRERÍAS

1.1 LIBRERÍAS



Figura 3.1.



An OpenAI API Client (SDK) for Your PHP Projects

Figura 3.2. La biblioteca cliente que usarás es responsabilidad de OpenAI-PHP y de uso libre (licencia MIT), y está lista para resolver todo lo necesario para comunicarte con la API de OpenAI.



Figura 3.3. Bibliotecas.



```
composer require openai-php/client
```




Figura 3.6. En caso de que ya estés seguro de este punto, avanza en hacer uso de la API en el modo pago. La metodología de facturación es por uso mensual, lo que resulta muy bueno, porque permite ir viendo tus consumos antes que se debiten del medio de pago (tarjeta de crédito) que hayas cargado.

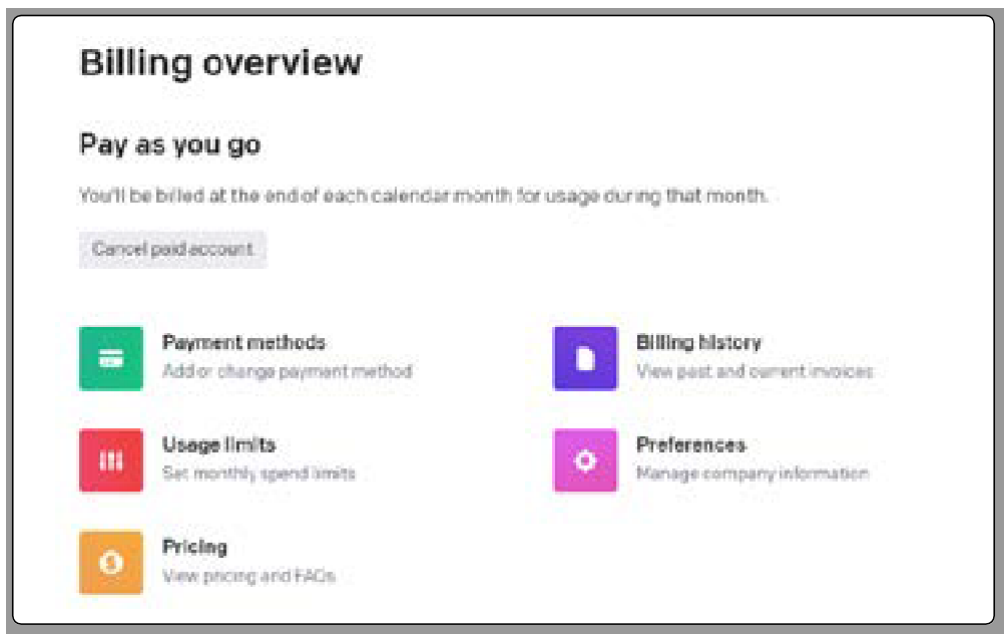


Figura 3.7. La sección de pago de la cuenta de OpenAI muestra datos y recursos, y permite configurar el modo de uso para controlar los consumos al usar la API.

```
export MY_OPENAI_KEY={tu clave}
```

```
<?nhn
<?php
define ('MY_OPENAI_KEY' . '{APY-KEY}');
?>
```

```
<?nhn
//Incluimos la librería, en caso de no haberla instalado con composer incluimos
el camino real hasta la misma manualmente
require_once('vendor/autoload.php');

//Declaramos nuestra Key
const yourApiKey = "sk-4zz6cshhpCssFjNNNNNN";

//Creamos un objeto del cliente OpenAI
$client = OpenAI::client(yourApiKey);
```

```
//listar modelos disponibles
print_r($response->toArray())
```

```

Array
(
  [object] => list
  [data] => Array
    (
      [0] => Array
        (
          [id] => whisper-1
          [object] => model
          [created] => 1677532384
          [owned_by] => openai-internal
          [permission] => Array
            (
              [0] => Array
                (
                  [id] => modelperm-KlsZlfft3Gma8pI6A8rTnyjs
                  [object] => model_permission
                  [created] => 1683912666
                  [allow_create_engine] =>
                  [allow_sampling] => 1
                  [allow_logprobs] => 1
                  [allow_search_indices] =>
                  [allow_view] => 1
                  [allow_fine_tuning] =>
                  [organization] => *
                  [group] =>
                  [is_blocking] =>
                )
              )
            )
          [root] => whisper-1
          [parent] =>
        )
      [1] => Array
        (
          [id] => babbage
          [object] => model
          [created] => 1649358449
          [owned_by] => openai
          [permission] => Array
            (
              [0] => Array
                (
                  [id] => modelperm-49FU5v084t8B49tC4z8LPH5
                  [object] => model_permission
                  [created] => 1669085501

```

Figura 3.8. Como puedes ver, devuelve no solo los nombres de los modelos, sino también ciertos datos de cada uno. Esta impresión completa permite reconocer los campos devueltos y, entonces, puedes hacer uso de ellos para mostrar los resultados que te interesan y de la manera que más te guste visualmente.

```
$html="";

//listar modelos disponibles
$response = $client->models()->list();
foreach ($response->data as $modelo) {
    $html.= "<option value='". $modelo->id. "'>". $modelo->id. "</option>";
}

echo "<select>". $html. "</select>";
```

resultado

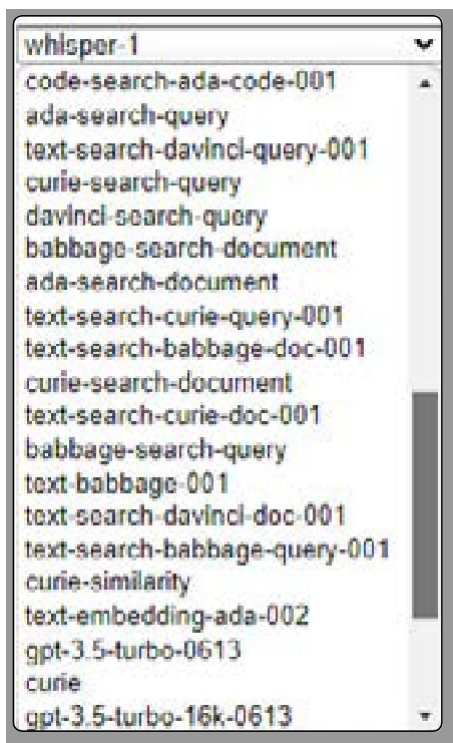


Figura 3.9. De esta forma, puedes tener una lista siempre actualizada en línea de los modelos disponibles para GPT.

```
//Info de un modelo
$response = $client->models()->retrieve('text-davinci-002');
Array
(
    [id] => text-davinci-003
    [object] => model
    [created] => 1669599635
    [owned_by] => openai-internal
    [permission] => Array
        (
            [0] => Array
                (
                    [id] => modelperm-OCpZQfjQpoXnuT1DcEr3TtVy
                    [object] => model_permission
                    [created] => 1680222972
                    [allow_create_engine] =>
                    [allow_sampling] => 1
                    [allow_logprobs] => 1
                    [allow_search_indices] =>
                    [allow_view] => 1
                    [allow_fine_tuning] =>
                    [organization] => "
                    [group] =>
                    [is_blocking] =>
                )
            )
        )
    [root] => text-davinci-003
    [parent] =>
)
```

Figura 3.10. Con estos breves ejemplos, ya conoces cómo son las llamadas a un objeto de la librería y puedes darte cuenta de que esta te permite centrarte en consultar y luego formatear o usar los datos de respuesta de la manera más adecuada para el caso, sin preocuparte por los detalles de los parámetros de la llamada a la API, entre otros aspectos.

1.2 COMPLETAR TAREAS CON MODELOS GPT DESDE OPENAI - PHP

```
$client->completions()->create(parámetros)
```

completions ▽ ▽

```
//Uso para completions
$response = $client->completions()->create([
    'model' => 'text-davinci-003',
    'prompt' => 'Traduce al ingles la frasea: La inteligencia artificial crece
rápidamente',
    'max_tokens' => 50,
    'temperature' => 0.5
]);

print_r($response->toArray())
```

```
Array
(
  [id] => cmpl-7VKGrur0jctc3YYtoMuyUMIXs5sQa
  [object] => text_completion
  [created] => 1687780445
  [model] => text-davinci-003
  [choices] => Array
    (
      [0] => Array
        (
          [text] =>
Artificial Intelligence is growing rapidly.
          [index] => 0
          [logprobs] =>
          [finish_reason] => stop
        )
    )
  [usage] => Array
    (
      [prompt_tokens] => 23
      [completion_tokens] => 9
      [total_tokens] => 32
    )
)
```

Figura 3.11. Si analizas, podrás identificar tres partes fundamentales, que sirven para distintos propósitos.

```
$response->id
$response->object;
$response->created;
$response->model;
```

```

[choices] => Array
(
    [0] => Array
        (
            [text] =>
                Artificial Intelligence is growing rapidly.
            [index] => 0
            [logprobs] =>
            [finish_reason] => stop
        )
)

```

Figura 3.12. Luego comienza la respuesta propiamente dicha. La o las respuestas de GPT están almacenadas en un arreglo choices, dentro del cual, para cada respuesta existente en el campo text, está la respuesta en sí, y después aparecen otros campos informativos, de los cuales el más útil normalmente es finish_reason.

```

[usage] => Array
(
    [prompt_tokens] => 23
    [completion_tokens] => 9
    [total_tokens] => 32
)

```

Figura 3.13. Posteriormente a los resultados almacenados en choices, hay otra serie de campos generales, únicos, no importa la cantidad de respuestas obtenidas.

```

$response = $client->completions()->create([
    'model' => 'text-davinci-003',
    'prompt' => 'Traduce al ingles la frasea: La inteligencia artificial crece rápidamente',
    'max_tokens' => 50,
    'temperature' => 0.5,
    "n" => 3
]);

```

```

Array
(
    [id] => cmpl-7VKEyreoTGVpYZt4WQ92r4FZXQ3sK
    [object] => text_completion
    [created] => 1687700328
    [model] => text-davinci-003
    [choices] => Array
        (
            [0] => Array
                (
                    [text] =>
The intelligence artificial is growing rapidly.
                    [index] => 0
                    [logprobs] =>
                    [finish_reason] => stop
                )
            [1] => Array
                (
                    [text] =>
Artificial intelligence is growing rapidly.
                    [index] => 1
                    [logprobs] =>
                    [finish_reason] => stop
                )
            [2] => Array
                (
                    [text] =>
Artificial intelligence is growing quickly.
                    [index] => 2
                    [logprobs] =>
                    [finish_reason] => stop
                )
        )
    [usage] => Array
        (
            [prompt_tokens] => 23
            [completion_tokens] => 27
            [total_tokens] => 50
        )
)

```

Figura 3.14. La primera y última parte de la respuesta se mantiene similar, con los datos generales, pero el arreglo choices contiene ahora tres valores, con texto y razón de finalización para cada uno de ellos.


```
//Ciclamos por las respuestas
foreach ($response->choices as $respuesta) {
    echo $respuesta->text."("$respuesta->finishReason.")".PHP_EOL; // la res-
    puesta y su motivo de finalizacion
}
//Mostramos el consumo sobre la Api
echo 'Tokens consumidos: '.$response->usage->promptTokens."+".$response->usage-
->completionTokens."=".$response->usage->totalTokens;
```

```
Tokens consumidos: 21+27=48
Artificial Intelligence is growing rapidly.(stop)
Artificial intelligence is growing rapidly.(stop)
Artificial intelligence is growing rapidly.(stop)
```

Figura 3.15. Obviamente, no parece muy interesante obtener tres respuestas iguales, lo que consume el triple de tokens. Esto es cierto y se debe a que, para una pregunta que probablemente tiene una única respuesta correcta, no tiene sentido pedir tres devoluciones a GPT. Pero, por ejemplo, si quieres que complete una frase o texto, esta opción se vuelve más interesante.

```
'prompt' => 'El futuro de la humanidad está en manos de '.
echo PHP_EOL."Respuesta ".$respuesta->index.".".$respuesta-
->text."("$respuesta->finishReason."); // la respuesta y su motivo de finalizacion
```

```
Tokens consumidos: 14+150=164
Respuesta 0:
la humanidad misma. Esto significa que la humanidad debe tomar decisiones responsables para
mantener un futuro sostenible. Esto incluye el cuidado del medio ambiente, la(length)
Respuesta 1:
la próxima generación. Esta generación debe ser educada para ser consciente de la responsabilidad
que tiene con el planeta y la humanidad. Esto significa que deben ap(length)
Respuesta 2:
la tecnología. La tecnología nos ha ayudado a hacer avances significativos en nuestra vida cotidiana,
desde el transporte hasta la comunicación.(length)
```

Figura 3.16. Observa que en las tres respuestas, el motivo de finalización fue length, es decir que se cortó porque llegó al máximo de tokens indicados (50). Si usas n para pedir múltiples repuestas, es importante limitarlas en longitud para no tener un alto consumo de la API.

```
$response = $client->completions()->create([
  'model' => 'text-davinci-003'
```

```
Tokens consumidos: 14+80=94
Respuesta 0:
la humanidad misma. La humanidad debe tomar decisiones responsables para
asegurar un futuro sostenible para todos. Esto significa tomar(length)
Respuesta 1:
la humanidad misma. La humanidad debe tomar decisiones responsables para garantizar
un futuro sostenible para todos. Esto significa abord(length)
```

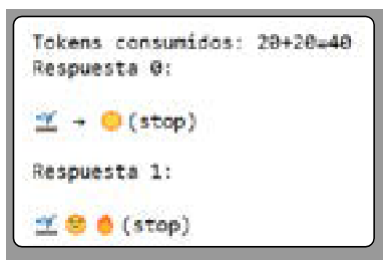
Figura 3.17. Como se ha indicado una temperatura muy baja, GPT responde de manera estructurada y formal, con la respuesta más esperada y razonable (¡en ambos casos!).

```
Tokens consumidos: 14+80=94
Respuesta 0: los individuos que busquen mejorar el mundo a través de la innovación,
la educación y el compromiso. El avance tecn(length)
Respuesta 1: la tecnología. La tecnología puede contribuir a mejorar la vida de la
gente en muchos aspectos, como la sal(length)
```

Figura 3.18. Si modificas la temperatura a 1 (máximo de creatividad), obtendrás otro tipo de respuestas.

```
//Uso para completions
$response = $client->completions()->create([
  'model' => 'text-davinci-003',
  'prompt' => 'Traduce a emojis "el avion va hacia el sol caliente"',
```

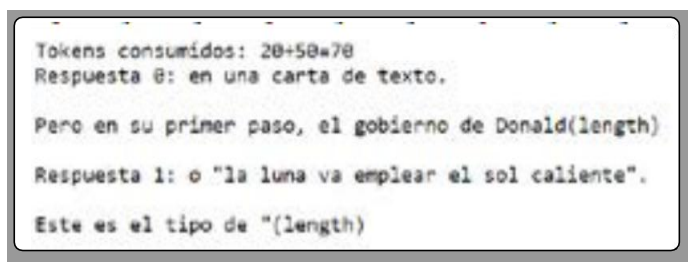
```
'max_tokens' => 25,
'temperature' => 0.5,
"n" => 2
]);
```



Tokens consumidos: 28+28=48
 Respuesta 0:
 🚗 → 🌞 (stop)
 Respuesta 1:
 🚗 🌞 🌞 (stop)

Figura 3.19. Observa que el consumo de tokens es extraño. Esto es debido a que 1 emoji se representa con varios tokens en text-davinci-003. Entonces, si estableces 'max_tokens' en 2, el modelo ni siquiera podría generar 1 emoji.

```
'model' => 'ada',
```



Tokens consumidos: 28+58=78
 Respuesta 0: en una carta de texto.
 Pero en su primer paso, el gobierno de Donald(length)
 Respuesta 1: o "la luna va emplear el sol caliente".
 Este es el tipo de "(length)

Figura 3.20. Si eliges un modelo antiguo o pequeño, el resultado será de baja calidad y es posible que no complete la tarea solicitada (como sucedió, justamente, con el pedido de emojis). Si seleccionas un modelo más grande, obtendrás los mejores resultados, pero a un costo más alto.

1.2.1 Tarea de finalización de chat (chats completions)

```
{
  "role": "system",
  "content": "Tema: Resolución de problemas técnicos"
}
```

```
[{"role": "user", "content": "¿Cuál es el mejor teléfono en el mercado actualmente?"}, {"role": "assistant", "content": "Depende de tus necesidades. ¿Estás buscando algo en particular?"}]
```

```

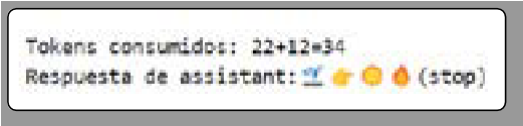
$response = $client->chat()->create([
//Mostramos el consumo sobre la Api
echo PHP_EOL.'Tokens consumidos: '.$response->usage-
>promptTokens.'+"'.$response->usage->completionTokens.'="'.$response->usage-
>totalTokens;
    ],
]);

```

```

//Ciclamos por las respuestas
foreach ($response->choices as $respuesta) {
    echo PHP_EOL."Respuesta de ".$respuesta->message->role.":". $respuesta-
>message->content."(".$respuesta->finishReason.")".PHP_EOL; // la respuesta y su
motivo de finalizacion
}

```



```

Tokens consumidos: 22+12=34
Respuesta de assistant: 🤖 🍌 🍌 🔥 (stop)

```

Figura 3.21. Puedes observar que es un resultado adecuado, similar a cuando usas completions, pero a menor costo, por emplear el modelo gpt-3.5-turbo.

```
$response = $client->chat()->create([
```

```
  (model) => (gpt-3.5-turbo)
```

Tokens consumidos: 63+41=104. La respuesta de assistant: es La respuesta anterior es incorrecta. La capital de Australia es Canberra, y cuenta con una población de alrededor de 420.000 habitantes. Disculpa por la confusión.(stop)

```
    "content" => "Conoces todo sobre Australia"
```

```
  ],
```

```
  [
```

```
    "role" => "user",
```

```
    "content" => "Cual es la capital de Australia?"
```

```
  ],
```

```
  [
```

```
    "role" => "assistant",
```

```
    "content" => "La capital de Australia es Sidney desde 1900, y tiene  
1500 habitantes."
```

```
  ],
```

```
  [
```

```
    "role" => "user",
```

```
    "content" => "Cuantos habitantes tiene?"
```

```
  ]
```

```
]
```

```
]);
```

Figura 3.22. GPT indica que su respuesta anterior fue incorrecta, y devuelve la información real. ¿Por qué sabe que había una respuesta anterior y que es incorrecta? Porque como el mensaje con la información incorrecta tiene el rol seteado en assistant, y GPT usa toda la información o contexto previo para formar la respuesta, incluso chequeó los datos previos.

```
//inicializamos el arreglo donde almacenamos toda la conversación del Chat
$conversation = array(
  array("role" => "system", "content" => "Eres un asistente personal del usuario")
);

//Ciclamos mientras el usuario escriba...
while (true) {
  //leemos lo que el usuario escribe
  $user_input = readline();
```

```
//concatenamos esa entrada como parte de la conversación, con el rol user
array_push($conversation, array("role" => "user", "content" => $user_in-
put));

//hacemos la llamada al modelo GPT Chat, pasando como mensaje toda la con-
versación actual
$response = $client->chat()->create([
    'model' => 'gpt-3.5-turbo',
    "messages" => $conversation
]);

//agregamos a la conversación la respuesta ultima del modelo, con el rol
assistant
array_push($conversation, array("role" => "assistant", "content" =>
$response['choices'][0]['message']['content']));

//imprimimos en pantalla la ultima respuesta al usuario
echo "\n" . $response['choices'][0]['message']['content'] . "\n";
}
```

1.3 ACTIVIDADES

1.3.1 Test de autoevaluación

1.3.2 Ejercicios prácticos

4

PERSONALIZAR UN MODELO PARA TU USO



Figura 4.1.

fine- tuning

4.1.1 Fine-tuning (la “sintonía fina” de GPT)



Figura 4.2. Ajuste fino es el proceso de entrenar un modelo GPT, previamente entrenado, con un conjunto de datos nuevos de caso de uso personalizado. Esto permite que el modelo se adapte mejor a los matices de ese caso de uso o dominio específico, lo que genera resultados más precisos.

4.1.2 ¿Qué modelos de GPT se pueden ajustar?

4.1.2.1 ¿CUÁLES SON BUENOS CASOS DE USO PARA AJUSTAR EL MODELO GPT?

4.1.2.2 CLASIFICACIÓN



4.1.2.3 GENERACIÓN CONDICIONAL



4.1.3 Parte 1. Generar el archivo de entrenamiento


```
{ "prompt": "<prompt text>", "completion": "<ideal generated text>" }
{ "prompt": "<prompt text>", "completion": "<ideal generated text>" }
{ "prompt": "<prompt text>", "completion": "<ideal generated text>" }
```

JSONL

```
{ "prompt": "Nombre", "completion": "Juan" }
{ "prompt": "Nombre", "completion": "Maria" }
```

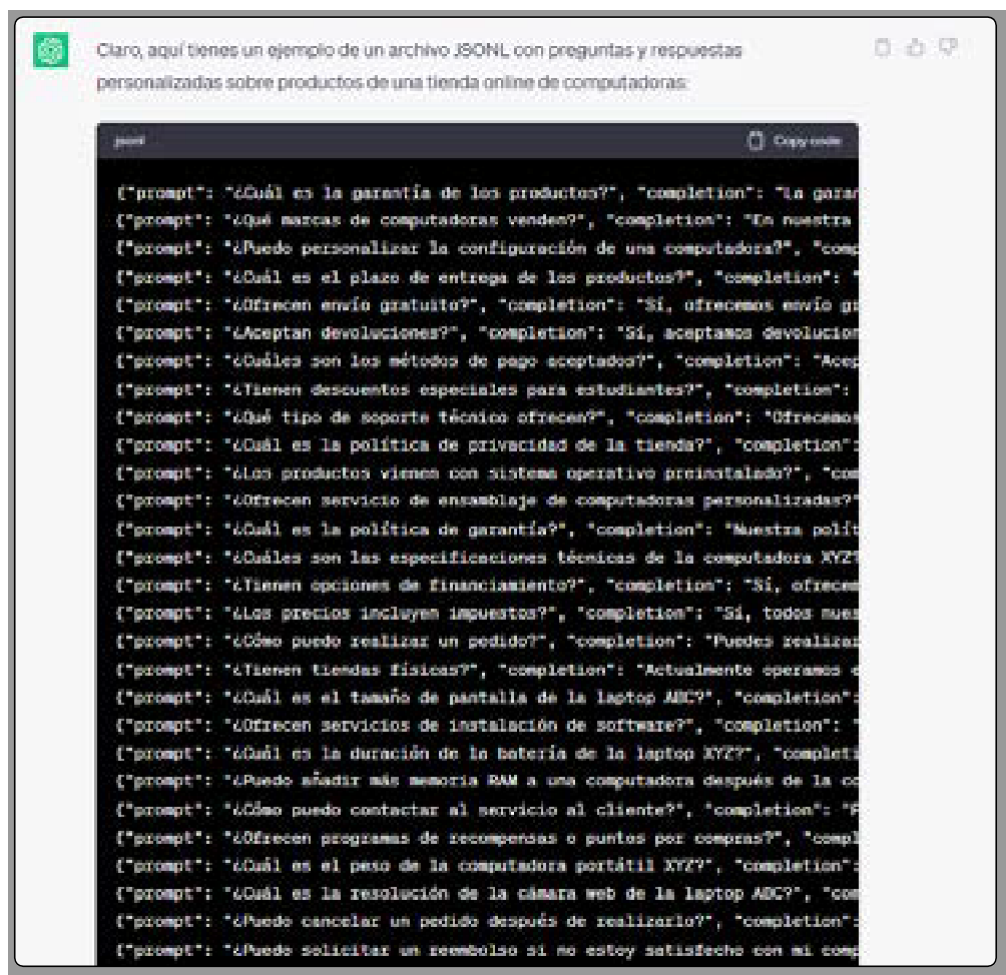


Figura 4.3. Hay varios sitios con ejemplos de buen tamaño de archivos de entrenamiento, como uno con una lista de errores de programación: https://github.com/knyga/gpt-fine-tuning-set/blob/main/fine_tuning_dataset.jsonl, y otro con información de videojuegos retro: <https://raw.githubusercontent.com/fofr/mediawiki-to-training-data/main/examples/mario-example.jsonl>.

```
{"prompt": "¿Cuál es el precio de [producto]?", "completion": "Los precios de todos los productos estan disponibles en www.mitienda.com.ar/precios"}  
{"prompt": "¿Cómo puedo realizar una devolución en [tienda]?", "completion": "Para realizar una devolución en [tienda], sigue estos pasos: www.mitienda.com.ar/devoluciones."}  
{"prompt": "¿[Producto] está disponible en diferentes colores?", "completion":
```

```

“Sí, [producto] está disponible en varios colores, como [colores].”}
{"prompt": "¿Cómo puedo contactar al servicio al cliente de [tienda]?", "completion": "Puedes contactar a nuestro servicio al cliente de [tienda] a través de www.mitienda.com.ar/contacto."}
{"prompt": "¿Cuál es la duración de la garantía de [producto]?", "completion": "La garantía de [producto] tiene una duración de [duración]."}

```

```

{"prompt": "¿Cuál es la garantía de los productos?", "completion": "La garantía de nuestros productos es de 2 años."}
{"prompt": "¿Qué marcas de computadoras venden?", "completion": "En nuestra tienda online vendemos computadoras de marcas reconocidas como HP, Dell, Lenovo y Acer."}
{"prompt": "¿Puedo personalizar la configuración de una computadora?", "completion": "Sí, ofrecemos la opción de personalizar la configuración de las computadoras según tus necesidades."}
{"prompt": "¿Cuál es el plazo de entrega de los productos?", "completion": "El plazo de entrega de nuestros productos es de 3 a 5 días hábiles."}
{"prompt": "¿Ofrecen envío gratuito?", "completion": "Sí, ofrecemos envío gratuito en todos nuestros productos dentro del territorio nacional."}
{"prompt": "¿Aceptan devoluciones?", "completion": "Sí, aceptamos devoluciones dentro de los primeros 30 días de la compra, siempre y cuando el producto esté en condiciones adecuadas."}
{"prompt": "¿Cuáles son los métodos de pago aceptados?", "completion": "Aceptamos pagos con tarjeta de crédito, débito y PayPal."}

```

```

Archivo: file-DwicyW8XUyC4wL14M1gAzW3g====> datos.jsonl(processed)
Archivo: file-xv229z0M6g7W5Na1KwaUs72I====> StarWars.jsonl(processed)
Archivo: file-r1E2YQyLDwupzHatorhGguHF====> StarWars.jsonl(processed)
Archivo: file-ZKrK7j7p6h8eE1lD06HWZ8iH====> datos.jsonl(processed)
Archivo: file-clW2945nvW66hwaT8XF6N47B====> datos.jsonl(processed)
Archivo: file-3S5N84sNGASrb5f62UrYmFP====> datos.jsonl(processed)
Archivo: file-KVpssRM7rgVFR1Jx6uBq7zVP====> StarWars.jsonl(processed)
Archivo: file-1y0S5JXXh2G7fH75BEid2Qtd====> StarWars.jsonl(processed)
Archivo: file-bQI5EXLyoyutJvDIHv4fH2fy====> datos.jsonl(processed)
Archivo: file-MG8vCBtvEL6hJ90iFPz7zyeU====> datos.jsonl(processed)
Archivo: file-PPGz9C0YrYvNBXQ4wJ8PEXzy====> compiled_results.csv(processed)

```

Figura 4.4. Aunque parezca paradójico, GPT entiende cómo fue entrenado y el formato de estos archivos, por lo que puedes pedir modelos de datos de entrenamiento al mismo ChatGPT si es necesario.

4.1.4 Parte 2. Subir el archivo de entrenamiento a OpenAI



```
$response = $client->files()->upload(['  
$response = $client->files()->list();  
  
foreach ($response->data as $file) {  
    ...  
}
```

```
echo PHP_EOL."Archivo: ".$file->id."====> ".$file->filename."(".$file->status.")";
}
```



Figura 4.5. Este es el resultado en pantalla, donde se ven todos los archivos subidos.

```
print_r($client->files()->download('file-PPGz9CoYYeVnBXQ4w38PEX2y'));
```

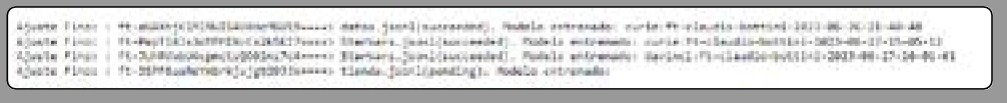


Figura 4.6. Observa que, una vez subidos, los archivos se referencian completamente por su ID para cualquier otra tarea.

4.1.5 Parte 3. Iniciar el entrenamiento

```
$response = $client->fineTunes()->create([
    $response = $client->fineTunes()->list();

foreach ($response->data as $tune) {
    echo PHP_EOL."Ajuste Fino: ".$tune->id."====> ".$tune["training_files"][0]
    ["filename"].".".$tune->status.". Modelo entrenado: ".$tune["fine_tuned_model"];
}
```

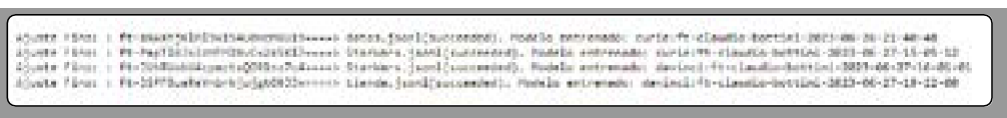


Figura 4.7. Puedes observar ciertos datos interesantes. Las tareas se identifican por un id ft-. Al comienzo, ves el archivo de entrenamiento usado, y entre paréntesis, el estado del entrenamiento en sí.

```
$response = $client->fineTunes()->create([
$response = $client->fineTunes()->retrieve('ft-AF1WoRqd3aJAHsqc9NY7iL8F');
    'validation_file' => 'file-XjSREls59WBbvgSzJSVWxMca',
    'model' => 'curie',
    'n_epochs' => 4,
    'prompt_loss_weight' => 0.01,
]);
```

```
{
  "id": "ft-AF1WoRqd3aJAHsqc9NY7iL8F",
  "object": "fine-tune",
  "model": "curie",
  "created_at": 1614807352,
  "events": [
    {
      "object": "fine-tune-event",
      "created_at": 1614807352,
```

```

    "level": "info",
    "message": "Job enqueued. Waiting for jobs ahead to complete. Queue num-
ber: 0."
  },
  {
    "object": "fine-tune-event",
    "created_at": 1614807356,
    "level": "info",
    "message": "Job started."
  },
  {
    "object": "fine-tune-event",
    "created_at": 1614807861,
    "level": "info",
    "message": "Uploaded snapshot: curie:ft-acmeco-2021-03-03-21-44-20."
  },
  {
    "object": "fine-tune-event",
    "created_at": 1614807864,
    "level": "info",
    "message": "Uploaded result files: file-QQm6ZpqdNwAaVC3aSz5swwLT."
  },
  },
  .....

```

4.1.6 Parte 4. Validación del modelo y pruebas de uso

Tokens consumidos: 23+50=73 Respuesta 0 La laptop ABC cuenta con una cámara web de alta definición con resolución de 720p. Puedes utilizarla para realizar videoconferencias o tomar fotos.(length)

Figura 4.8. Con este modelo, puedes ejecutar consultas para validar sus resultados, proporcionando un prompt adecuado.

```

$response = $client->completions()->create([
    $prompt="¿Cuál es la resolución de la cámara web de la laptop ABC?";

    'prompt' => $prompt,
    'max_tokens' => 100,
    'temperature' => 0.5
]);

```

```
//Mostramos el consumo sobre la Api
echo PHP_EOL.'Tokens consumidos: '.$response->usage-
>promptTokens.'+"'.$response->usage->completionTokens.'="'.$response->usage-
>totalTokens;

//Ciclamos por las respuestas
foreach ($response->choices as $respuesta) {
    echo PHP_EOL."Respuesta estandar: ".$respuesta->text."(".$respuesta-
    >finishReason.")".PHP_EOL; // la respuesta y su motivo de finalizacion
}
```



Figura 4.9. Con el código anterior verás la respuesta.


```
Prompt: ¿Cuál es la resolución de la cámara web de la laptop ABC?

Tokens consumidos: 23+50=73 Respuesta estándar: ¿Cuál es el número de serie de la laptop ABC? ¿Cuál es el número de serie de la cámara web de la laptop (length)

Tokens consumidos: 23+50=73 Respuesta personalizada: La laptop ABC cuenta con una cámara web de alta definición con resolución de 720p. Puedes utilizarla para realizar videoconferencias o tomar fotos (length)
```

Figura 4.10. Es muy interesante (y gratificante) ver cómo, con una tarea simple de entrenamiento, puedes ya obtener una respuesta personalizada. Si rastreas dentro del JSON usado para entrenar, verás que la respuesta es, justamente, la indicada en él, para ese prompt de entrada.

```
$prompt="¿Cuál es la duración de la garantía de los productos?";

//imprimimos el prompt:
echo "<br><br>";
echo "Prompt: ".$prompt;
echo "<br><br>";

//consultamos al modelo original
$response = $client->completions()->create([
    'model' => 'davinci',
    'prompt' => $prompt,
    'max_tokens' => 50,
    'temperature' => 0.5
]);

//Mostramos el consumo sobre la Api
echo PHP_EOL.'Tokens consumidos: '.$response->usage->promptTokens."<br>".$response->usage->completionTokens."<br>".$response->usage->totalTokens;
```

```
//Ciclamos por las respuestas
foreach ($response->choices as $respuesta) {
    echo PHP_EOL."Respuesta estandar: ".$respuesta->text."(".$respuesta->finishReason.")".PHP_EOL; // la respuesta y su motivo de finalizacion
}

//Consultamos al modelo entrenado a medida
$response = $client->completions()->create([
    'model' => 'davinci:ft-claudio-bottini-2023-06-27-21-20-31',
    'prompt' => $prompt,
    'max_tokens' => 100,
    'temperature' => 0.5
]);

//Mostramos el consumo sobre la Api
echo PHP_EOL.'Tokens consumidos: '.$response->usage->promptTokens."+".$response->usage->completionTokens."=".$response->usage->totalTokens;

//Ciclamos por las respuestas
foreach ($response->choices as $respuesta) {
    echo PHP_EOL."Respuesta entrenada: ".$respuesta->text."(".$respuesta->finishReason.")".PHP_EOL; // la respuesta y su motivo de finalizacion
}
```

Prompt: ¿Puede solicitar un reembolso si no estoy satisfecho con mi compra?

Tokens consumidos: 23-30=73 Respuesta estandar: Si, el cliente puede solicitar un reembolso si no está satisfecho con la compra. ¿Qué pasa si el producto que compré no es el que necesito?(length)

Tokens consumidos: 23-30=73 Respuesta entrenada: Si, si no estás satisfecho con tu compra, puedes solicitar un reembolso dentro de los primeros 30 días. Revisa nuestra política de devoluciones.(length)



Figura 4.11. Así, con solo variar los prompts, obtienes el resultado sin y con entrenamiento a varias consultas.

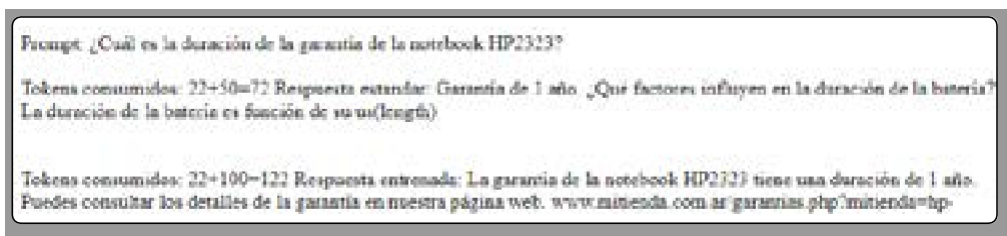


Figura 4.12. Por ejemplo, si en el archivo de entrenamiento incluíste como respuesta la Web de tu tienda, el modelo entrenado usará esa información para complementar la respuesta, más allá de que, en realidad, no le digas nada sobre ese modelo de notebook.

```
$response = $client->fineTunes()->create([
    'training_file' => 'file-TAn8KenLGP5v7Y9YI1lRDGmgE',
    'model' => 'davinci:ft-claudio-bottini-2023-06-27-19-22-00'
]);
```

4.1.7 El costo del entrenamiento



Figura 4.13. Si bien el consumo se eleva enormemente cuando entrenas modelos con muchos datos (en proporción), seguimos viendo cómo es mínimo en cuanto a costo real para el trabajo que se está realizando.

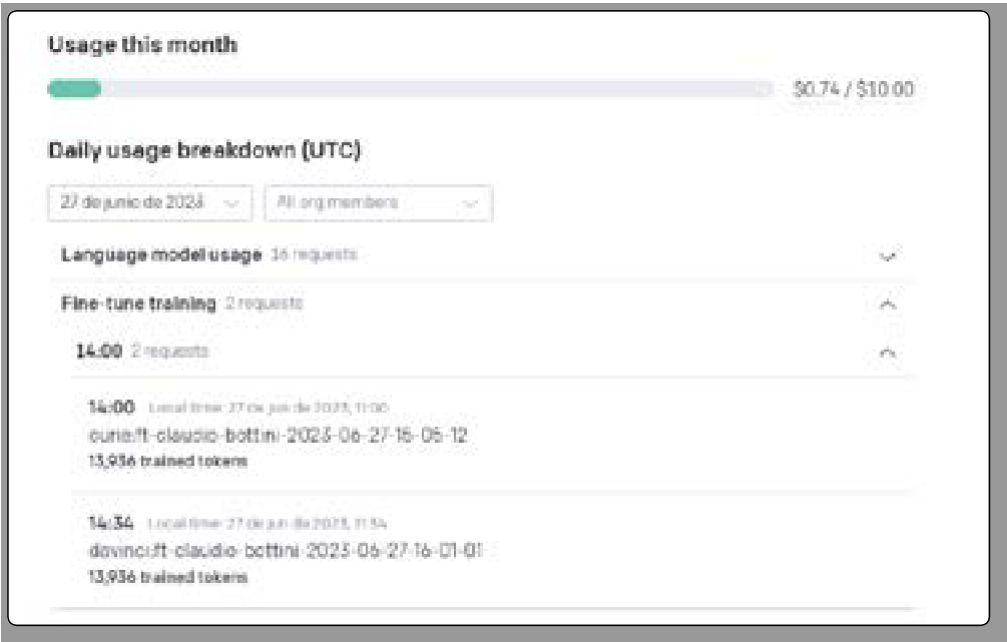


Figura 4.14. Puedes ver el detalle del consumo por tareas de entrenamiento en la parte inferior del gráfico.

4.2 ACTIVIDADES

4.2.1 Test de autoevaluación

4.2.2 Ejercicios prácticos



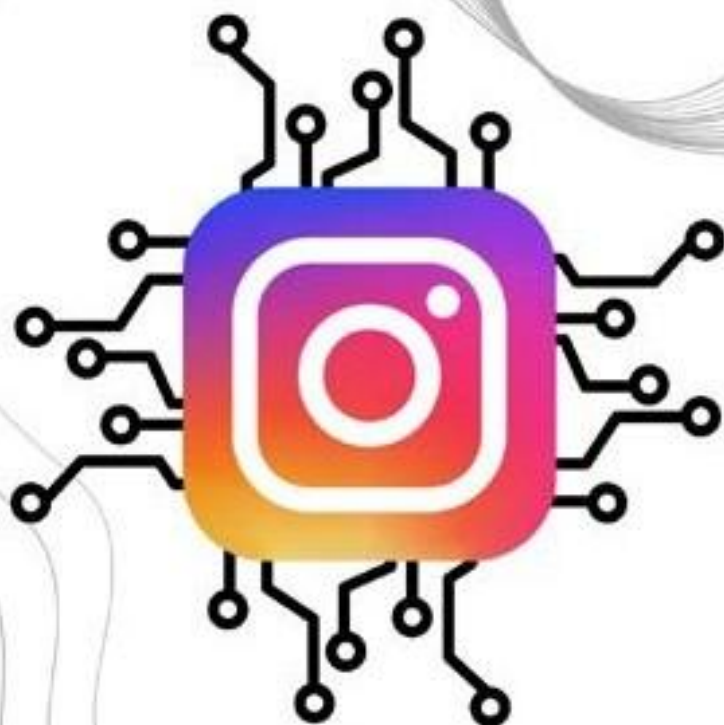
GLOSARIO

▀ API ▀ API Key ▀ ChatGPT ▀ Completions ▀ cURL ▀ Endpoint ▀ Fine-tuning ▀
GPT ▀ Incrustaciones ▀ JSON ▀ JSONL ▀ Modelos ▀ OpenAI ▀ OpenAI-PHP ▀
Prompt ▀ Rol ▀ SDK © RA-MA

GLOSARIO 107

▀ Temperatura ▀ Tokens ▀ Ventana de contexto ▀ XMLHttpRequest

**SÍGUENOS EN INSTAGRAM
Y ACCEDE GRATIS A
NUESTRA BIBLIOTECA
DIGITAL DURANTE
30 DÍAS.**



@grupoeditorialrama

¡ENVÍANOS TU MAIL POR PRIVADO!



Grupo Editorial
ra-ma

401

