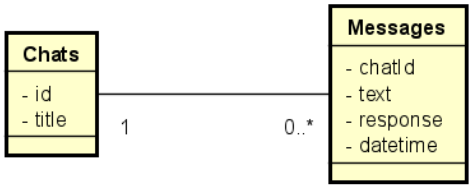
Documentació Chat GPT

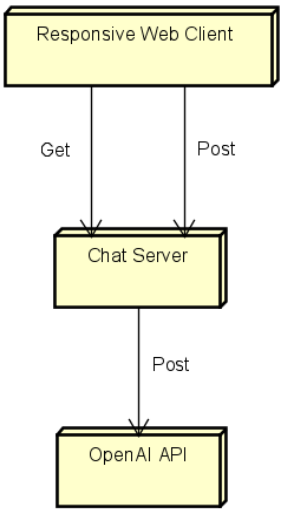
XAVIER BERMEJO SOTILLO 47192442F

UML



UML senzill, no s'han ficat usuaris per no ser molt semblant a l'exemple d'Atena. Pàgina web on podem tenir diversos xats i per cada xat tindrem diversos missatges que consten d'una pregunta que fem nosaltres i la resposta que rebem de l'intel·ligència artificial d'OpenAI.

Diagrama



En el meu cas he fet un client que interactua amb el Chat Server que es l'encarregat d'emmagatzemar xats i missatges. El servidor també es encarregat de ser intermediari entre l'API d'OpenAI i el client.

El client pot fer una pregunta enviant la pregunta al servidor, el qual s'encarrega

d'enviarla a l'API d'OpenaAI i agafar la resposta que ens retornen, guardarla i retornarla al client.

Per si es necesari la key per a OpenAI que he utilitzat es:
`OPENAI_API_KEY="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"`
ficada en un fitxer .env

Rutes Servidor

app.get ('/messages/:id', getAllMessagesController);

- Paràmetres: Identificació del chat seleccionat (chatId).
- Resposta: Llista amb tots el missatges del chat seleccionat.
- Comentari: Cada cop que es selecciona un chat es sol·licita tots els missatges d'aquest chat per mostrar les preguntes i respostes fetes.

app.post ('/openAI', AIquestion);

- Paràmetres: Identificació del chat seleccionat , Pregunta que s'enviarà a la IA
- Resposta: Resposta rebuda per OpenAI
- Comentari: Aqui el servidor actua com a intermediari entre el client i OpenAI enviant la pregunta i retornant la resposta rebuda.

app.get ('/chats', getAllChatsController);

- Paràmetres: No parametres
- Resposta: Llista amb tots el chats creats.
- Comentari: Cada cop que s'inicia la pagina es crida aquesta funció per tal de veure tots el chats disponibles.

app.post ('/chats', createController);

- Paràmetres: Títol que tindrà el nostre chat
- Resposta: Missatge confirmant que s'ha afegit un nou chat.
- Comentari: Els chats que s'afegeixen obtindran el id corresponent a la posició de la llista en que s'afegeixen (començant per 0).

app.delete ('/chats/:id', deleteChatController);

- Paràmetres: Identificació del chat seleccionat.

- Resposta: Confirmació de l'eliminació del chat.

- Comentari: Cada vegada que s'elimina un chat es canvien els ids per que tinguin el mateix id que la posició en la llista. Evitant tenir col·lisions en ids al afegir nous chats.

  Exemple (Eliminem Chat 1):
  {"**chats**":
  [{"id":0,"title":"Google"},
  {"id":1,"title":"Hola"},
  {"id":2,"title":"aloo"}],

  "**messages**":[
  {"chatId":0, "text":"Hola", …},
  {"chatId":0, "text":"Adeu", …},

  {"chatId":1, "text":"Bon Dia", …},
  {"chatId":1, "text":"Bona Nit", …},

  {"chatId":2, "text":"Per que si", …}]

  {"**chats**":
  [{"id":0,"title":"Google"},
  {"id":1,"title":"aloo"}],

  "**messages**":[
  {"chatId":0, "text":"Hola", …},
  {"chatId":0, "text":"Adeu", …},

  {"chatId":1, "text":"Per que si", …}]

  Si ho faig així a l'hora de fer un nou chat tal i com vaig fer des d'un principi, al insertar un nou chat coincidirien els ids.
  En aquest exemple si no es reordena i afegim un nou chat hi haurien 2 chats amb id 2.

Connexió amb OpenAI

```
try {
    const prompt = req.body.prompt;
    const chatId = req.body.id;
    const response = await openai.createCompletion({
      model: "text-davinci-003",
      prompt: `${prompt}`,
      temperature: 0,
      max_tokens: 3000,
      top_p: 1,
      frequency_penalty: 0.5,
      presence_penalty: 0,
    });

    res.status(200).send({
      message: response.data.choices[0].text
    });
```

La **temperatura** de Davinci és un paràmetre que controla el nivell d'aleatorietat a la selecció de paraules. Com més gran sigui el valor, més aleatori serà el text generat.
El paràmetre **max_tokens** controla el nombre màxim de tokens (paraules) que es generaran en una sola frase.
El paràmetre **top_p** controla la probabilitat que un token sigui elegit per completar una frase.
El paràmetre **frequency_penalty** penalitza els tokens amb més freqüència per augmentar la diversitat del text generat.
Finalment, el paràmetre **presence_penalty** penalitza els tokens amb menys presència per augmentar la fluïdesa del text generat.

Únic petit canvi desde la presentació

L'única cosa que ha canviat desde la presentació és com guardar els missatges.
Abans el que feia era demanar desde el client que es guardin les preguntes fetes i les respostes retornades.
Això no es una bona pràctica, llavors s'ha canviat i ara es guarda directamente desde el servidor.

Canvis

CLIENT

```
const response = await fetch(this.url + '/openAI', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        prompt: question,
        id: chatId
    })
})

if (response.ok) {
  const res = await response.json();
  respuesta = res.message.trim();
  chatContainer.innerHTML += this.message_view(true,respuesta);
  $('#text').val('');
  /*
  $.ajax({
    dataType: "json",
    url: this.url + "/message",
    method: "POST",
    data: {chatId, question, respuesta}
  })
  .then(r => {
    //this.messages_Controll(chatId);
    $('#text').val('');})
  .catch(error => {console.error(error.status, error.responseText);});
  */
```

Al client seria treure la part comentada que solicita guardar la pregunta y la resposta obtinguda.

SERVIDOR

```
/*
const saveMessage = (req, res, next) => {
  let {chatId, question, respuesta} = req.body;
  chat_model.add_message(chatId, question, respuesta)
  .then(() => {
    res.status(201).send({
      success: 'true',
      message: 'message added successfully',
    });
  })
  .catch(error => {next(Error(`message not created:\n${error}`));});
};
*/
//app.post    ('/message',          saveMessage);
```

```
// Obtenemos la respuesta de OpenAI y la enviamos al cliente
const AIquestion = async (req, res) => {
  try {
    const prompt = req.body.prompt;
    const chatId = req.body.id;
    const response = await openai.createCompletion({
      model: "text-davinci-003",
      prompt: `${prompt}`,
      temperature: 0,
      max_tokens: 3000,
      top_p: 1,
      frequency_penalty: 0.5,
      presence_penalty: 0,
    });

    res.status(200).send({
      message: response.data.choices[0].text
    });
    chat_model.add_message(chatId, prompt, response.data.choices[0].text)
  } catch (error) {
    console.error(error)
    res.status(500).send(error || 'Something went wrong');
  }
};
```

Al servidor seria treure la ruta per guardar els missatges que utilitza el "chat_model.add_message" i fer la crida de "chat_model.add_message" quan ens retornin la resposta.