GENERALITAT VALENCIANA
CONSELLERIA D'EDUCACIÓ, INVESTIGACIÓ, CULTURA I ESPORT

IES JUAN DE GARAY
C /Juan de Garay, 25 46017 VALENCIA
Tel.: 961206045 Fax: 961206046
46012963@gva.es juandegaray.com

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Práctica 10

API GATEWAY

Web Socket Chat mejorado

aws academy

WS-1

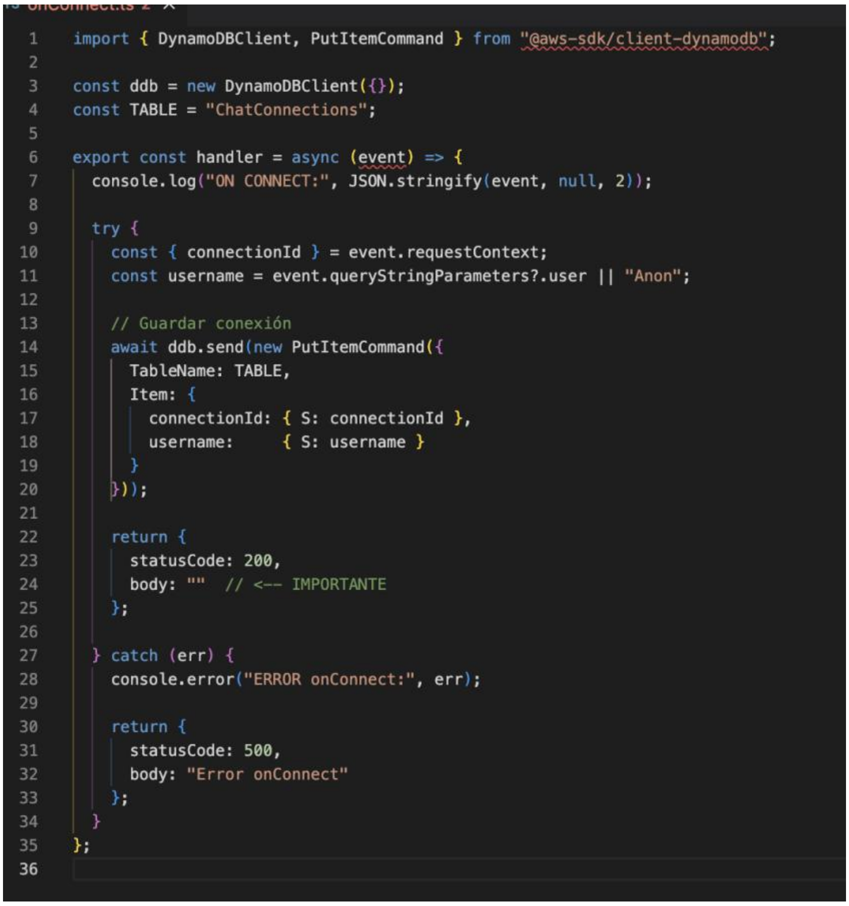
Vamos a mejorar el "chat" visto en el ejercicio anterior, para ello, crearemos una tabla de mensajes donde ir guardando éstos:

Pasos:

- Create table
- Table name: ChatMessages
- Partition key: roomId (String)
- Sort key: createdAt (String)

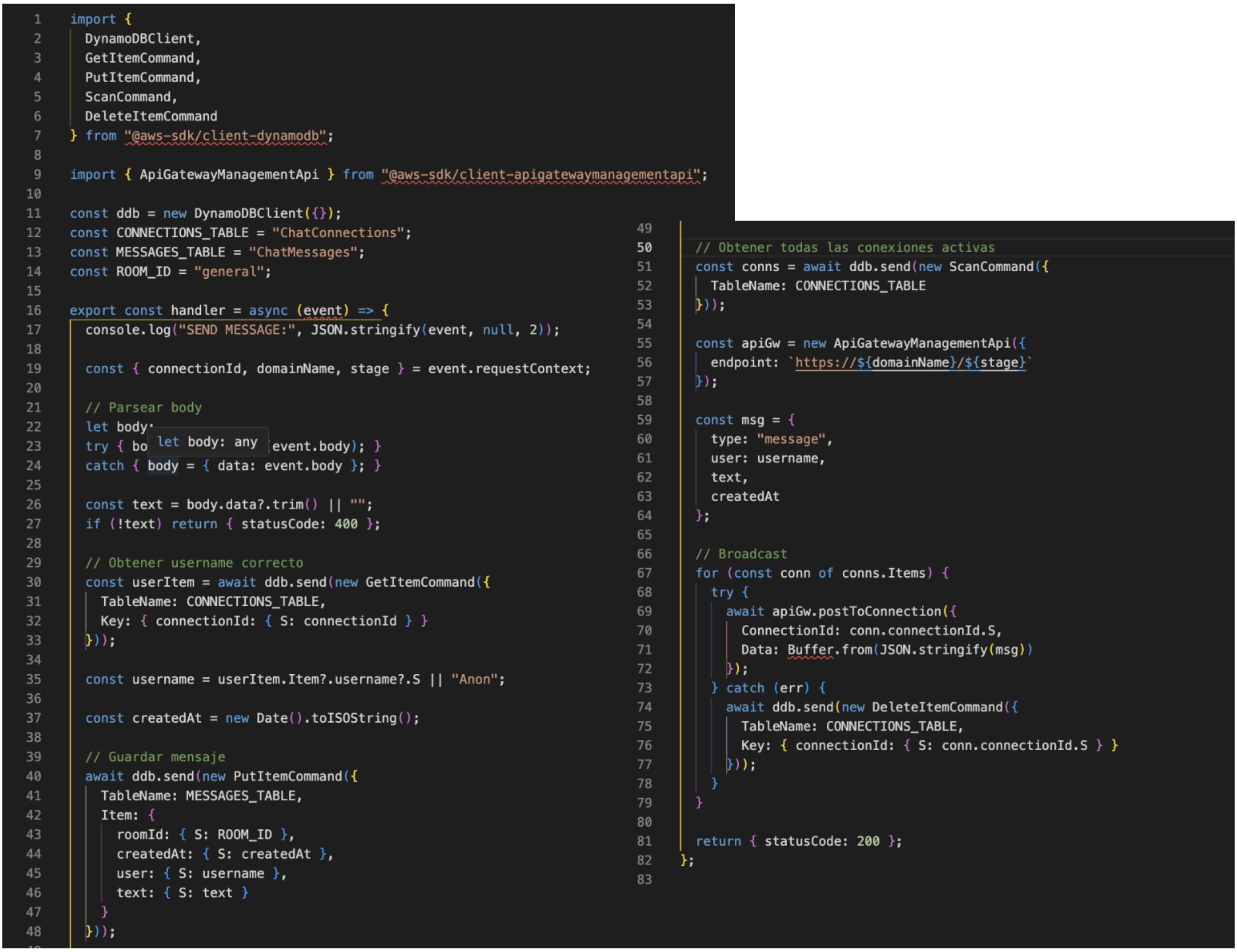
Modificaremos:

- Función onConnect



```javascript
import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";

const ddb = new DynamoDBClient({});
const TABLE = "ChatConnections";

export const handler = async (event) => {
  console.log("ON CONNECT:", JSON.stringify(event, null, 2));

  try {
    const { connectionId } = event.requestContext;
    const username = event.queryStringParameters?.user || "Anon";

    // Guardar conexión
    await ddb.send(new PutItemCommand({
      TableName: TABLE,
      Item: {
        connectionId: { S: connectionId },
        username:     { S: username }
      }
    }));

    return {
      statusCode: 200,
      body: ""  // <-- IMPORTANTE
    };

  } catch (err) {
    console.error("ERROR onConnect:", err);

    return {
      statusCode: 500,
      body: "Error onConnect"
    };
  }
};
```

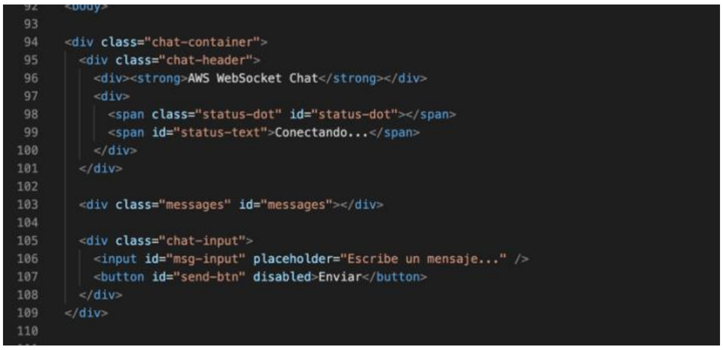
- Función sendMessage



```
import {
  DynamoDBClient,
  GetItemCommand,
  PutItemCommand,
  ScanCommand,
  DeleteItemCommand
} from "@aws-sdk/client-dynamodb";

import { ApiGatewayManagementApi } from "@aws-sdk/client-apigatewaymanagementapi";

const ddb = new DynamoDBClient({});
const CONNECTIONS_TABLE = "ChatConnections";
const MESSAGES_TABLE = "ChatMessages";
const ROOM_ID = "general";

export const handler = async (event) => {
  console.log("SEND MESSAGE:", JSON.stringify(event, null, 2));

  const { connectionId, domainName, stage } = event.requestContext;

  // Parsear body
  let body;
  try { bo let body: any event.body); }
  catch { body = { data: event.body }; }

  const text = body.data?.trim() || "";
  if (!text) return { statusCode: 400 };

  // Obtener username correcto
  const userItem = await ddb.send(new GetItemCommand({
    TableName: CONNECTIONS_TABLE,
    Key: { connectionId: { S: connectionId } }
  }));

  const username = userItem.Item?.username?.S || "Anon";

  const createdAt = new Date().toISOString();

  // Guardar mensaje
  await ddb.send(new PutItemCommand({
    TableName: MESSAGES_TABLE,
    Item: {
      roomId: { S: ROOM_ID },
      createdAt: { S: createdAt },
      user: { S: username },
      text: { S: text }
    }
  }));

  // Obtener todas las conexiones activas
  const conns = await ddb.send(new ScanCommand({
    TableName: CONNECTIONS_TABLE
  }));

  const apiGw = new ApiGatewayManagementApi({
    endpoint: `https://${domainName}/${stage}`
  });

  const msg = {
    type: "message",
    user: username,
    text,
    createdAt
  };

  // Broadcast
  for (const conn of conns.Items) {
    try {
      await apiGw.postToConnection({
        ConnectionId: conn.connectionId.S,
        Data: Buffer.from(JSON.stringify(msg))
      });
    } catch (err) {
      await ddb.send(new DeleteItemCommand({
        TableName: CONNECTIONS_TABLE,
        Key: { connectionId: { S: conn.connectionId.S } }
      }));
    }
  }

  return { statusCode: 200 };
};
```

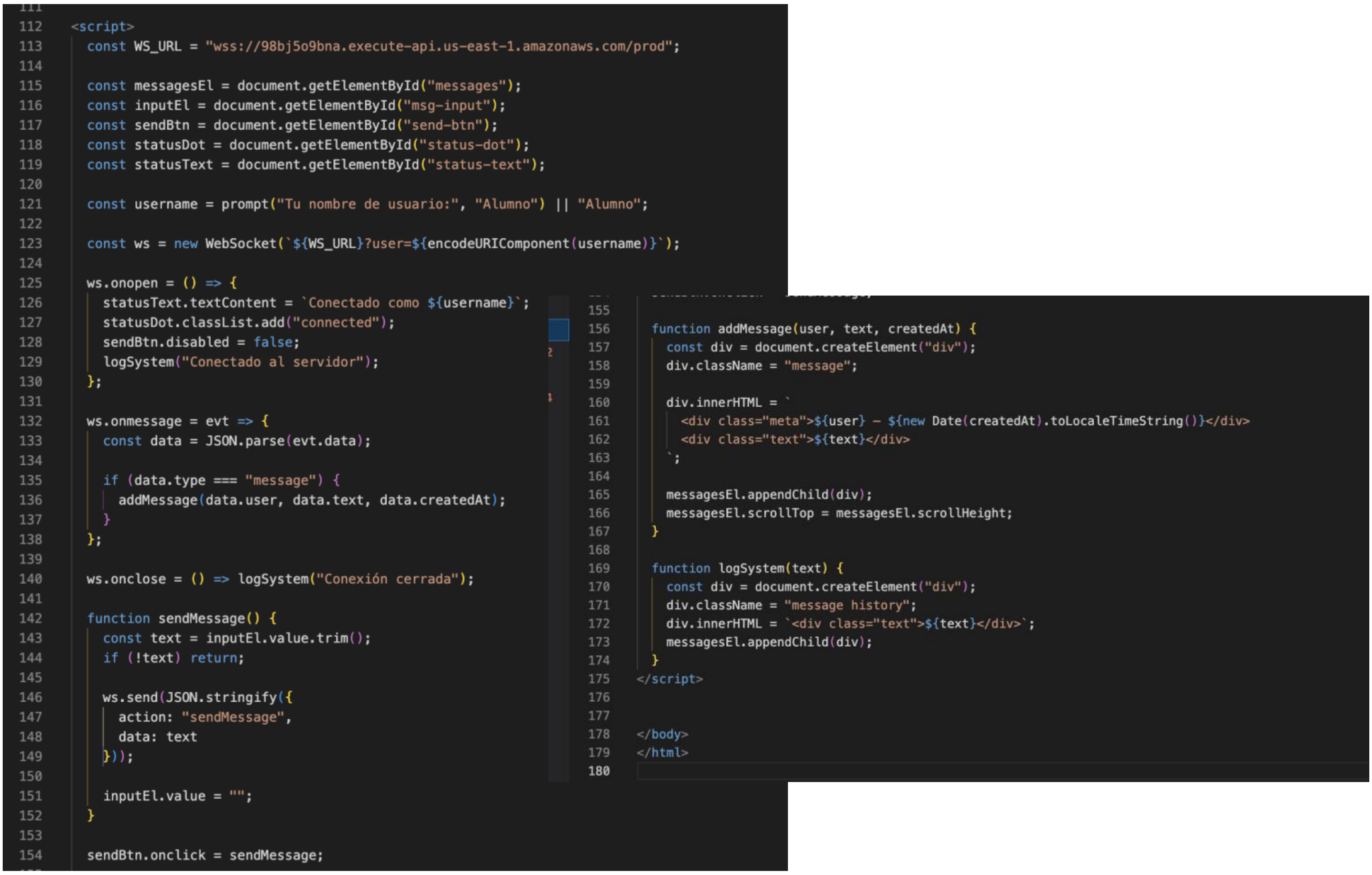
- Index.html

Primero la UI sin estilos:



```html
<body>

<div class="chat-container">
  <div class="chat-header">
    <div><strong>AWS WebSocket Chat</strong></div>
    <div>
      <span class="status-dot" id="status-dot"></span>
      <span id="status-text">Conectando...</span>
    </div>
  </div>

  <div class="messages" id="messages"></div>

  <div class="chat-input">
    <input id="msg-input" placeholder="Escribe un mensaje..." />
    <button id="send-btn" disabled>Enviar</button>
  </div>
</div>
```

Y la parte de scripting:



```javascript
<script>
  const WS_URL = "wss://98bj5o9bna.execute-api.us-east-1.amazonaws.com/prod";

  const messagesEl = document.getElementById("messages");
  const inputEl = document.getElementById("msg-input");
  const sendBtn = document.getElementById("send-btn");
  const statusDot = document.getElementById("status-dot");
  const statusText = document.getElementById("status-text");

  const username = prompt("Tu nombre de usuario:", "Alumno") || "Alumno";

  const ws = new WebSocket(`${WS_URL}?user=${encodeURIComponent(username)}`);

  ws.onopen = () => {
    statusText.textContent = `Conectado como ${username}`;
    statusDot.classList.add("connected");
    sendBtn.disabled = false;
    logSystem("Conectado al servidor");
  };

  ws.onmessage = evt => {
    const data = JSON.parse(evt.data);

    if (data.type === "message") {
      addMessage(data.user, data.text, data.createdAt);
    }
  };

  ws.onclose = () => logSystem("Conexión cerrada");

  function sendMessage() {
    const text = inputEl.value.trim();
    if (!text) return;

    ws.send(JSON.stringify({
      action: "sendMessage",
      data: text
    }));

    inputEl.value = "";
  }

  sendBtn.onclick = sendMessage;

  function addMessage(user, text, createdAt) {
    const div = document.createElement("div");
    div.className = "message";

    div.innerHTML = `
      <div class="meta">${user} — ${new Date(createdAt).toLocaleTimeString()}</div>
      <div class="text">${text}</div>
    `;

    messagesEl.appendChild(div);
    messagesEl.scrollTop = messagesEl.scrollHeight;
  }

  function logSystem(text) {
    const div = document.createElement("div");
    div.className = "message history";
    div.innerHTML = `<div class="text">${text}</div>`;
    messagesEl.appendChild(div);
  }
</script>


</body>
</html>
```

A tener en cuenta que el index.html hay que subirlo a un bucket para hacer pruebas ya que ws suele bloquearse en los navegadores.

Ahora sólo falta mejorarlo recuperando un cierto número de mensajes del historial