

Actividad 3_1: Inngest

Alvaro Del Valle Fernández

Para este ejercicio usaré la aplicación de CookieClicker realizada para Orto. Esta usa los elementos de ejemplo dados adaptados a esta práctica.

Primero creo la ruta de carpetas de inngest:

C:

```
| client.js  
|  
+-- functions  
    scoreWorkflow.js
```

[Client.js](#) usando el ejemplo dado:

```
import { Inngest } from 'inngest';  
  
// Cliente Inngest compartido por todas las funciones  
export const inngest = new Inngest({  
  id: 'serverCookie'  
});
```

[scoreWorkflow.js](#) con el código dado:

```
import { inngest } from '../client.js';  
import { enviarMensajeTelegram } from '../../utils/telegram.js';  
  
/**  
 * Ejemplo 4: Workflow Multi-Paso  
 *  
 * Demuestra:  
 * - Encadenamiento de múltiples steps  
 * - Uso de step.sleep para delays  
 * - Mantener estado entre steps  
 * - Flujo complejo con múltiples acciones  
 */  
export const onboardingUsuario = inngest.createFunction(  
  { id: 'onboarding-usuario' },  
  { event: 'usuario/registro' },  
  async ({ event, step }) => {  
    // Step 1: Mensaje de bienvenida  
    await step.run('enviar-bienvenida', async () => {  
      const mensaje = `👋 Bienvenido ${event.data.nombre}!*\n\n`  
    }  
  }  
);
```

```
`Gracias por registrarte con el email:  
${event.data.email}\n\n` +  
`En los próximos minutos recibirás más información.`;  
  
    return await enviarMensajeTelegram(mensaje);  
});  
  
// Step 2: Esperar 10 segundos  
await step.sleep('espera-inicial', '10s');  
  
// Step 3: Recordatorio de configuración  
await step.run('enviar-recordatorio-configuracion', async () =>  
{  
    const mensaje = `⚙️ *Configura tu Perfil*\n\n` +  
    `Hola ${event.data.nombre},\n\n` +  
    `No olvides completar tu perfil para aprovechar todas las  
funcionalidades.`;  
  
    return await enviarMensajeTelegram(mensaje);  
});  
  
// Step 4: Esperar otros 10 segundos  
await step.sleep('espera-tips', '10s');  
  
// Step 5: Enviar tips de uso  
await step.run('enviar-tips', async () => {  
    const mensaje = `💡 *Tips de Uso*\n\n` +  
    `• Explora el dashboard\n` +  
    `• Configura tus notificaciones\n` +  
    `• Invita a tus compañeros\n\n` +  
    `¡Que disfrutes la plataforma!`;  
  
    return await enviarMensajeTelegram(mensaje);  
});  
  
return {  
    usuario: event.data.nombre,  
    email: event.data.email,  
    onboardingCompletado: true,  
    pasos: 5  
};  
}  
);
```

Ahora creo units/telegram.js:

```
import axios from 'axios';

/**
 * Envía un mensaje a un chat de Telegram
 * @param {string} mensaje - Texto del mensaje a enviar
 * @returns {Promise<Object>} - Respuesta de la API de Telegram
 */
export async function enviarMensajeTelegram(mensaje) {
  const token = process.env.TELEGRAM_BOT_TOKEN; //TERMINAR
ESTO=====
=====
  const chatId = process.env.TELEGRAM_CHAT_ID;

  if (!token || !chatId) {
    console.warn('⚠️ TELEGRAM_BOT_TOKEN o TELEGRAM_CHAT_ID no
configurados');
    console.log('Mensaje que se enviaria:', mensaje);
    return { ok: true, result: { message_id: 'demo' } };
  }

  try {
    const url = `https://api.telegram.org/bot${token}/sendMessage`;
    const response = await axios.post(url, {
      chat_id: chatId,
      text: mensaje,
      parse_mode: 'Markdown'
    });

    return response.data;
  } catch (error) {
    console.error('Error enviando mensaje a Telegram:',
error.message);
    throw error;
  }
}

/**
 * Obtiene actualizaciones del bot para extraer el chat_id
 * @returns {Promise<Object>} - Últimas actualizaciones
 */
export async function obtenerActualizaciones() {
  const token = process.env.TELEGRAM_BOT_TOKEN;

  if (!token) {
    throw new Error('TELEGRAM_BOT_TOKEN no configurado');
  }
}
```

```
const url = `https://api.telegram.org/bot${token}/getUpdates`;
const response = await axios.get(url);
return response.data;
}
```

Ahora dentro de mi server actual adapto el contenido para que funcione con inngest:

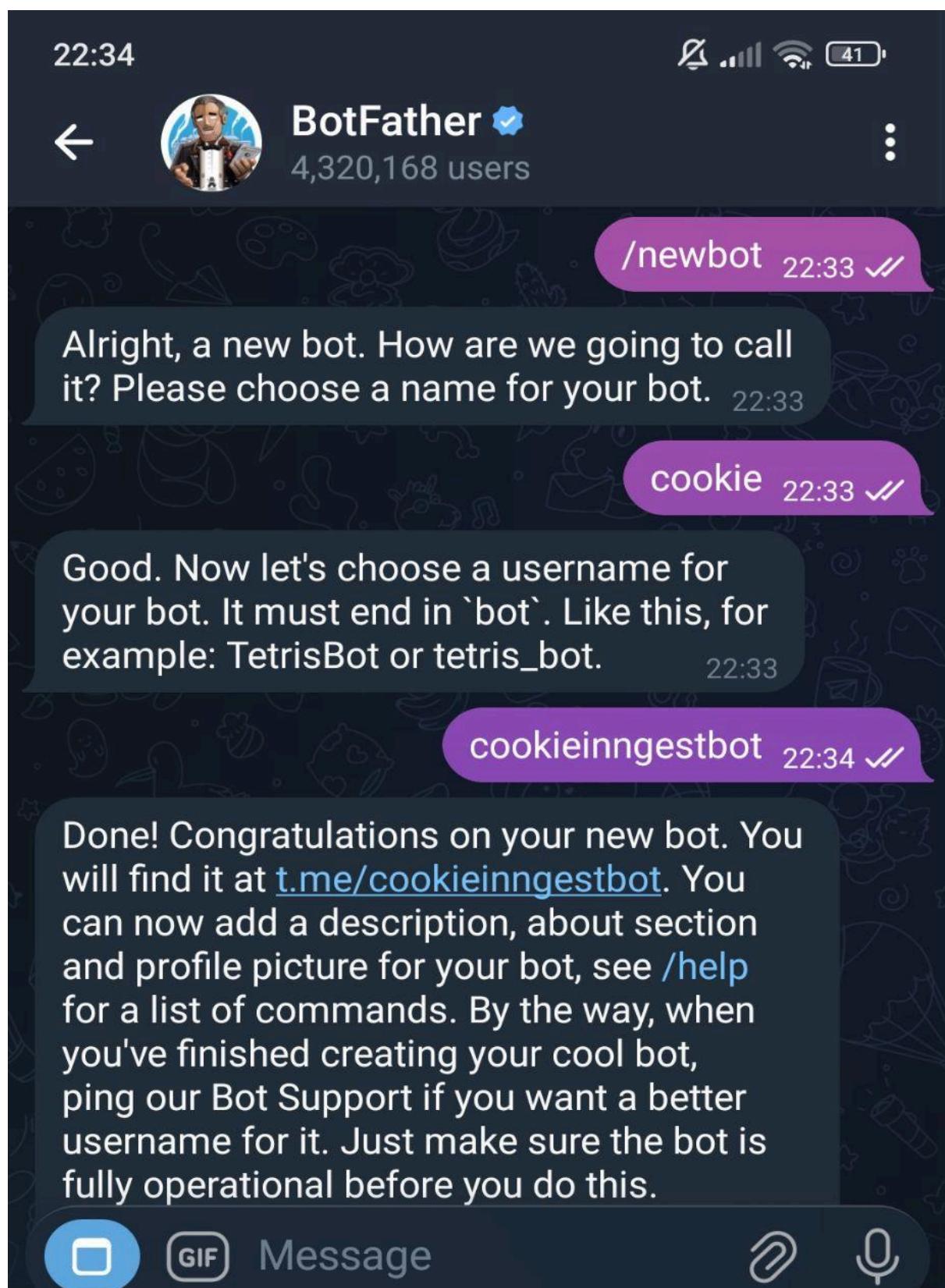
```
import {server} from"inngest/express";
import {inngest} from"inngest/client.js";
import {server} from"inngest/functions/scoreWorkflow.js";
```

```
app.use(
"/ali/inngest",
server({client: inngest,
functions: [scoreWorkflow]}));
```

```
await inngest.send({
  name: "score/created",
  data: { clicks }
});
```

Con esto adapto el documento dado a la práctica anterior.

Ahora voy a telegram a BotFather y creo el servidor:



Lo configuro:

 **BotFather** ✅
4,320,168 users

What can this bot do?

BotFather is the one bot to rule them all. Use it to create new bot accounts and manage your existing bots.

About Telegram bots:
<https://core.telegram.org/bots>

Bot API manual:
<https://core.telegram.org/bots/api>

Contact [@BotSupport](#) if you have questions about the Bot API.

6 de febrero

/start 22:33 ✓

I can help you create and manage Telegram bots. If you're new to the Bot API, please see the manual.

  Message  

Creado:

Ahora configuro el TELEGRAM_BOT_TOKEN y el TELEGRAM_CHAT_ID para que funcione correctamente con mi Bot de BotFather. Ahora hago una prueba, el Inngest es llamado cada vez que el jugador termina una partida, se guarda en mongodb y envía el mensaje en el bot creado de telegram.

```

Guardar Copiar Contraer todo Expandir todo Filtrar JSON
ok: true
▼ result:
  ▼ 0:
    update_id: 799167242
    ▼ message:
      message_id: 3
      ▼ from:

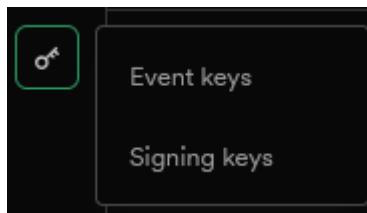
```

Funciona por lo que podemos ver

Tenemos que configurar Inngest, primero creamos una cuenta:

The screenshot shows the Inngest web interface. On the left, there's a sidebar with sections like 'Production', 'Metrics', 'Runs', 'Events', 'Insights' (Beta), 'Apps', 'Functions', 'Event Types', and 'Webhooks'. The main area is titled 'Getting started' and shows 'STEP 1 OF 4 Create an Inngest app'. It explains what an Inngest App is and provides instructions to run a CLI command: 'npx inngest-cl@latest dev'. Below this, there's a 'Dev Server' section with the URL 'http://localhost:8288'. To the right, there's an 'EXPLORE ONBOARDING GUIDE' with links to 'Create an Inngest app', 'Deploy your Inngest app', 'Sync your app to Inngest', 'Invoke your function', 'See documentation', 'Join discord community', and 'Request a demo'.

Buscamos keys:



Comprobación del test muestra que todos los elementos fucnionan excepto la key de inngest dando el error de ello:

```

PS C:\Users\delva\cookieclicker\electron-cookie-clicker> node .\serverCookie.js
Servidor Express corriendo en http://localhost:5000
Servidor WebSocket corriendo en ws://localhost:5000
{ ok: true, result: [ { update_id: 799167242, message: [Object] } ] }
Conectada a MongoDB Atlas
Error: Inngest API Error: 401 Event key not found
at Inngest.getErrorResponse (file:///C:/Users/delva/cookieclicker/electron-cookie-clicker/node_modules/inngest/components/Inngest.js:260:26)
at applyHookToOutput.result_ids.retryWithBackoff.maxAttempts (file:///C:/Users/delva/cookieclicker/electron-cookie-clicker/node_modules/inngest/components/Inngest.js:486:60)
at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
at async retryWithBackoff (file:///C:/Users/delva/cookieclicker/electron-cookie-clicker/node_modules/inngest/helpers/promises.js:161:10)
at async Inngest._send (file:///C:/Users/delva/cookieclicker/electron-cookie-clicker/node_modules/inngest/components/Inngest.js:469:52)
at async C:\Users\delva\cookieclicker\electron-cookie-clicker\serverCookie.js:78:5

```

Con esto tenemos Inngest correctamente configurado con la app cookieclicker aunque no detecte la key, probablemente limpiado el cache o esperando el programa realmente funcione.

