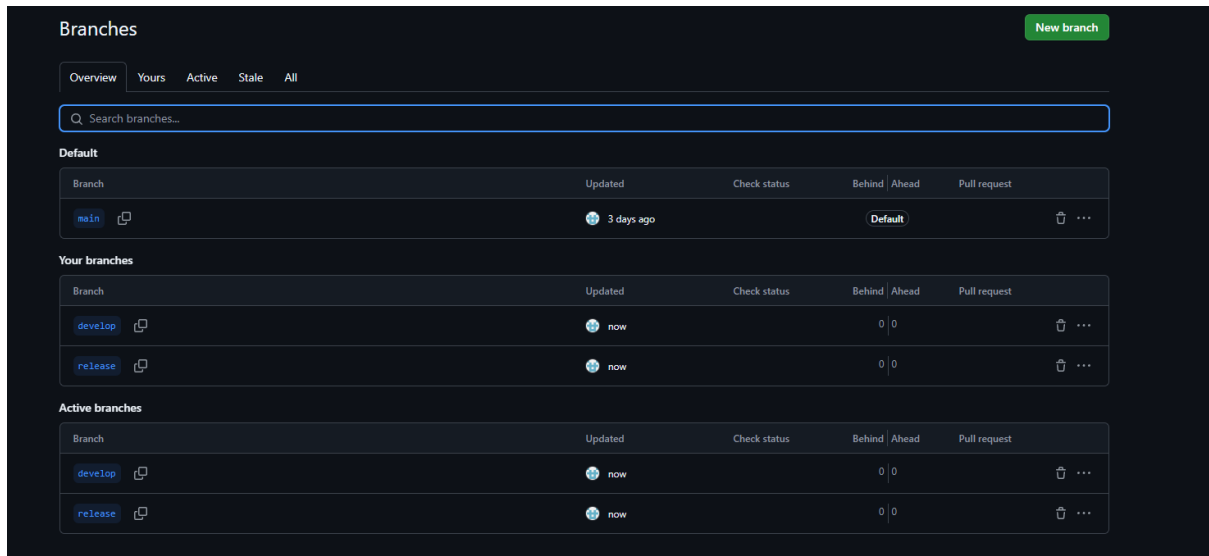


## Implementación en github:



Se crearon 3 ramas ya que es la forma ideal de llevar un proyecto estas 3 son :

Main (ambiente productivo)

Release (ambiente de pruebas)

Develop (ambiente desarrollo)

Las 3 ramas están protegidas para que solo se pueda subir cambios con pull request.

## Implementación microservicios:

### Descripción del Microservicio de Ingesta

El microservicio de ingesta tiene como objetivo principal recibir, procesar y almacenar datos provenientes de diferentes fuentes. Este es el primer punto de entrada para los datos en el sistema

Esta lambda está programada para que se ejecute todos los lunes a las 00:00

```
services > ingestion-app > ! serverless.yml
3 provider:
4
5 runtime: nodejs18.x
6 region: us-east-2
7 stage: dev
8 iamRoleStatements:
9   - Effect: "Allow"
10     Action:
11       - "s3:PutObject"
12       - "s3:GetObject"
13       - "s3:ListBucket"
14     Resource:
15       - "arn:aws:s3:::data-models-jl/*"
16
17 plugins:
18   - serverless-offline
19
20 functions:
21   ingestion:
22     handler: src/app/functions/ingestion.handler
23     events:
24       - http:
25         path: /ingestion
26         method: get
27       - schedule:
28         rate: cron(0 0 ? * 2 *)
29
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Server ready: http://localhost:3000

## Tecnologías Utilizadas

- **AWS Lambda:** Permite ejecutar el microservicio de manera serverless, manejando automáticamente la escalabilidad.
- **API Gateway:** Expone el endpoint público seguro para recibir las solicitudes de datos.
- **CloudWatch:** Monitorea las métricas y logs generados por el microservicio.
- **Almacenamiento:** S3

## Ejecución

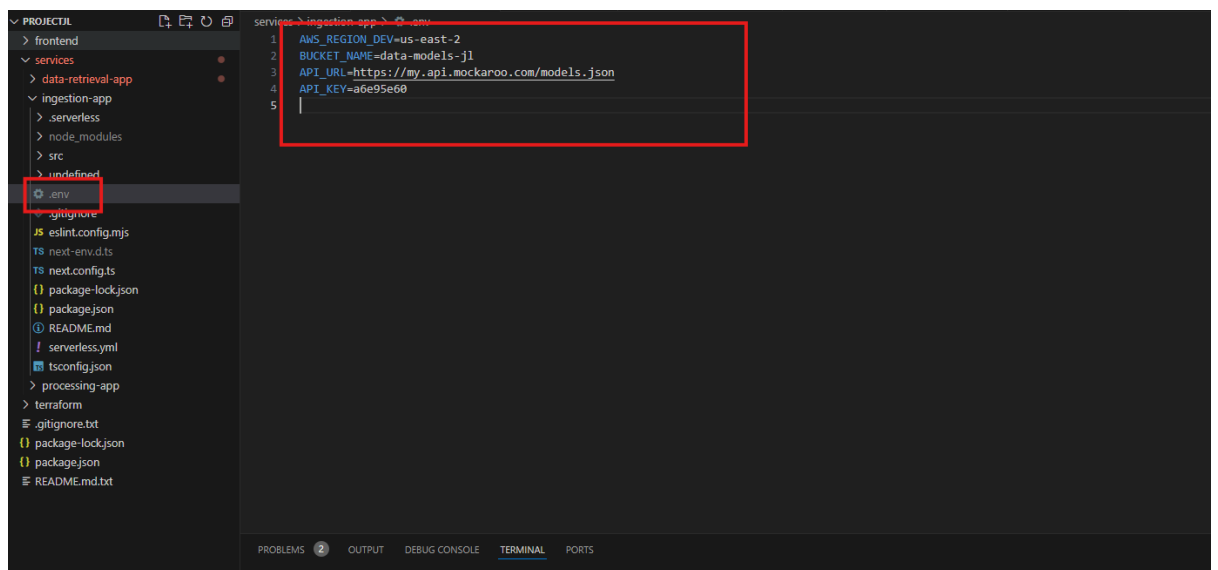
Para probar este servicio local mente es necesario ejecutar el comando **npm i** y posterior a ello usar el comando **npm serverless offline** de esta manera se ejecutará nuestra lambda localmente, adicional se tiene que crear un archivo llamado **.env** que debería de tener lo siguiente valores :

**AWS\_REGION\_DEV=us-east-2**

BUCKET\_NAME=data-models-jl

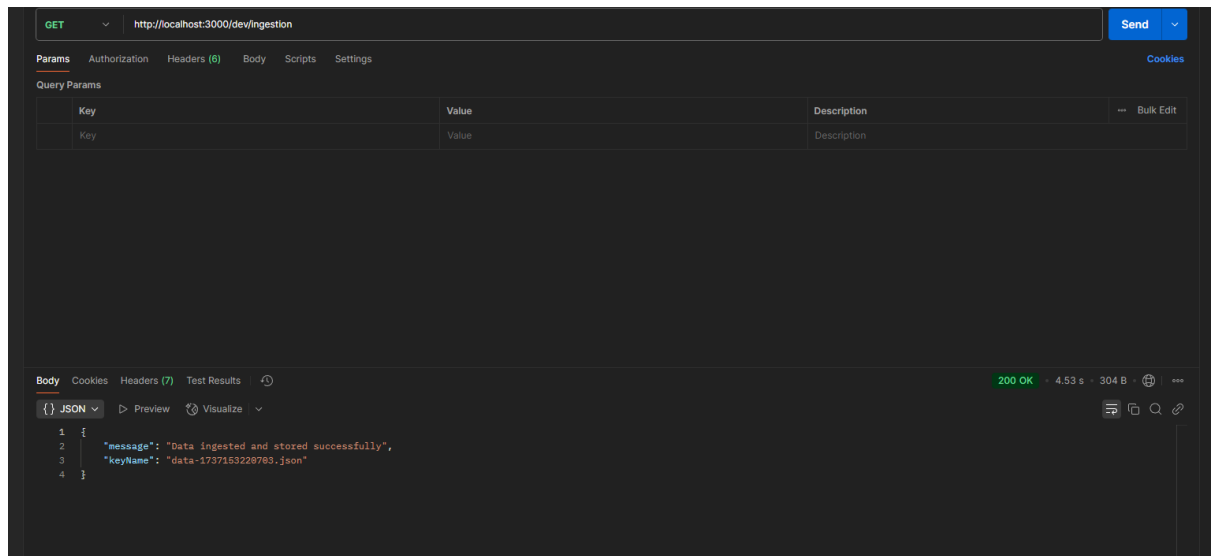
API\_URL=https://my.api.mockaroo.com/models.json

API\_KEY=a6e95e60

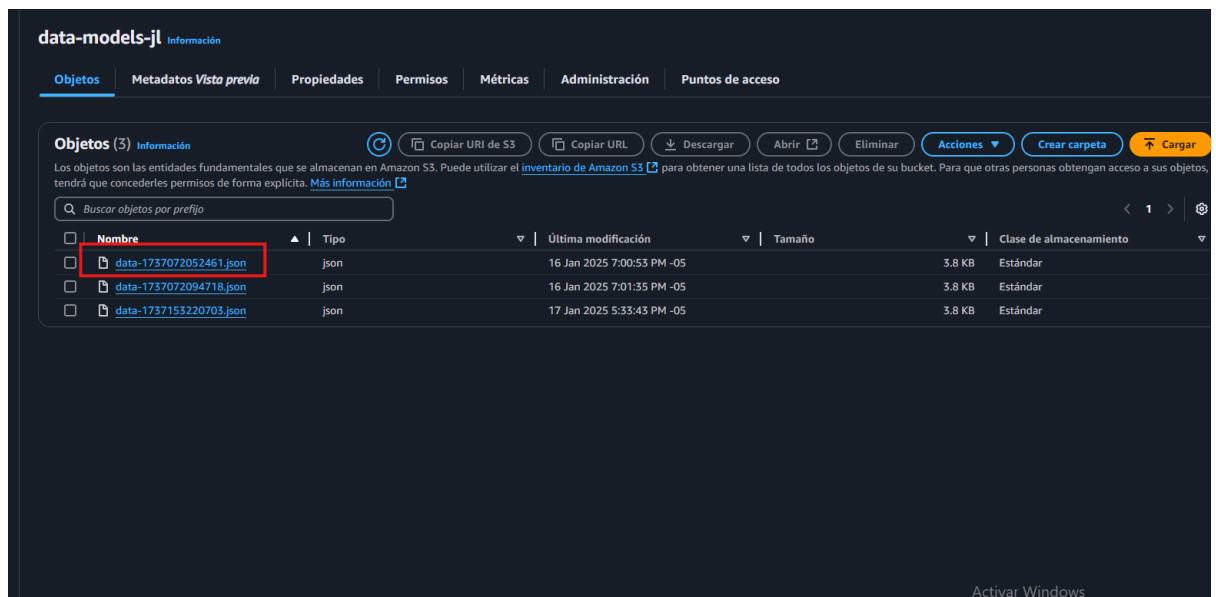


El siguiente paso es poner la ruta <http://localhost:3000/dev/ingestion> en nuestro navegador o aplicación de consumo como postman

Prueba de consumo local :



Al ejecutar esta lambda subira un archivo al buckets en el s3 el archivo con extencion .json contiene la data cruda de todos los datos que se sacaron de la api publica creada en la pagina <https://mockaroo.com/schemas/671143>,



La api publica a la cual se le realizo el consumo es:

<https://my.api.mockaroo.com/models.jsonkey=a6e95e60>

La lambda esta desplegada en aws y se puede usar ejecutando en el navegador la siguiente ruta <https://01eezx0ota.execute-api.us-east-2.amazonaws.com/dev/ingestion>

Para desplegar el micro servicio en aws nuevamente usamos el comando `serverless deploy`

## Descripción del Microservicio de procesamiento de datos

El microservicio de procesamiento se encarga de realizar análisis, cálculos y transformaciones más complejas sobre los datos recibidos, convirtiendo la información bruta en resultados procesados y útiles. Este microservicio es el encargado de transformar los datos ingresados en métricas, estadísticas o formatos preparados para ser consumidos por otros servicios, como reportes o visualizaciones.

Ejecución :

Para probar este micro servicio es necesario ejecutar el comando `npm i` en la consola y posterior a ello ejecutar el comando `npx serverless invoke local --function process --path s3-event.json`, se debe configurar el archivo que se encuentra en el proyecto llamado `s3-event` para que active el evento en la aplicación ya que esta funciona por Eventos.

Se debe crear un archivo `.env` que contenga las siguientes variables de entorno

```
AWS_REGION_DEV=us-east-2  
BUCKET_NAME=data-models-jl  
TABLE_NAME=processedData
```

```
PROJECTIL
├── frontend
├── services
│   ├── data-retrieval-app
│   ├── ingestion-app
│   └── processing-app
│       ├── .serverless
│       ├── node_modules
│       ├── public
│       ├── src
│       └── undefined
├── .env
├── .gitignore
├── JS eslint.config.mjs
├── TS next-env.d.ts
├── TS next.config.ts
├── package-lock.json
├── package.json
├── README.md
├── s3-event.json
├── serverless.yml
├── tsconfig.json
├── terraform
├── .gitignore.txt
└── package-lock.json

services > processing-app > .env
1  AWS_REGION_DEV=us-east-2
2  BUCKET_NAME=data-models-jl
3  TABLE_NAME=processedData
4
5
```

Este debería contener el nombre de uno de los archivos que estan en el bucket del s3 dentro del valor key “data-1737060334441.json”

Ejemplo de uno de los valores que debería traer es

data-models-jl

Información

Objetos

Metadatos

Vista previa

Propiedades

Permisos

Métricas

Administración

Puntos de acceso

Objetos (4)

Información

Copiar URI de S3

Copiar URL

Descargar

Abrir

Eliminar

Acciones

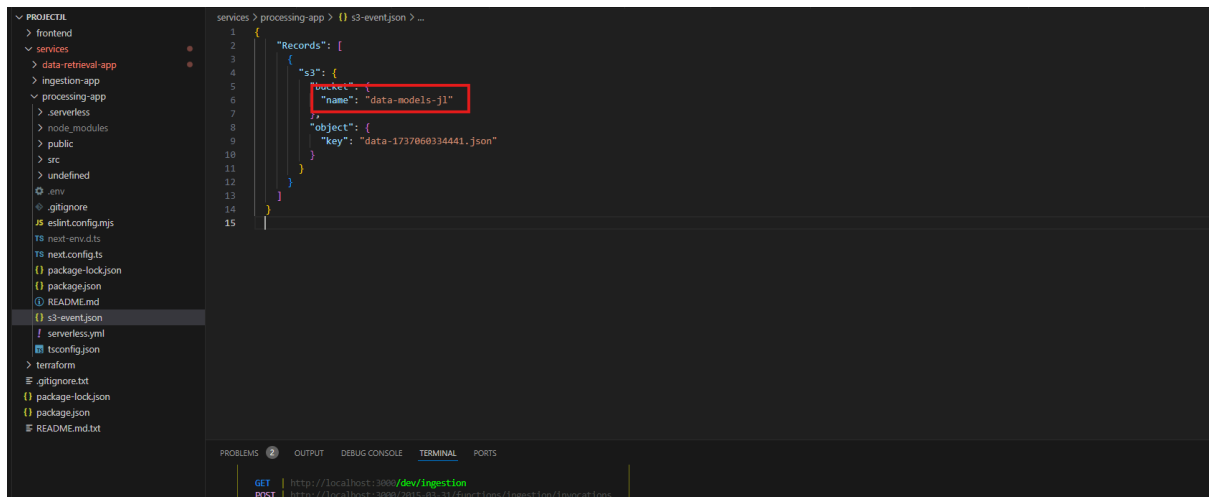
Crear carpeta

Cargar

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

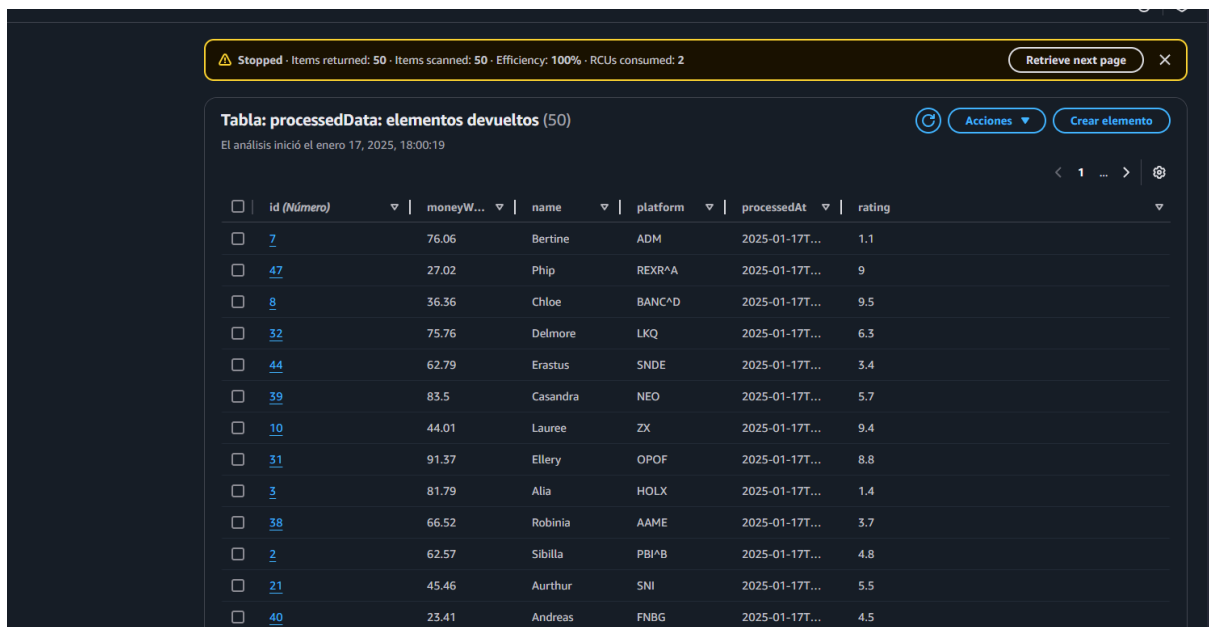
Buscar objetos por prefijo

Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<a href="#">data-1737072052461.json</a>	json	16 Jan 2025 7:00:53 PM -05	3.8 KB	Estándar
<a href="#">data-1737072094718.json</a>	json	16 Jan 2025 7:01:35 PM -05	3.8 KB	Estándar
<a href="#">data-1737153220703.json</a>	json	17 Jan 2025 5:33:43 PM -05	3.8 KB	Estándar
<a href="#">data-1737153384861.json</a>	json	17 Jan 2025 5:36:27 PM -05	3.8 KB	Estándar



```
1 {
2   "Records": [
3     {
4       "s3": {
5         "bucket": {
6           "name": "data-models-jl"
7         },
8         "object": {
9           "key": "data-1737860334441.json"
10        }
11      }
12    }
13  ]
14 }
15 }
```

Una vez ejecutado el comando podemos observar como en nuestra base de datos Dynamo db



	id (Número)	moneyW...	name	platform	processedAt	rating
<input type="checkbox"/>	<a href="#">7</a>	76.06	Bertine	ADM	2025-01-17T...	1.1
<input type="checkbox"/>	<a href="#">47</a>	27.02	Phip	REXR^A	2025-01-17T...	9
<input type="checkbox"/>	<a href="#">8</a>	36.36	Chloe	BANC^D	2025-01-17T...	9.5
<input type="checkbox"/>	<a href="#">32</a>	75.76	Delmore	LKQ	2025-01-17T...	6.3
<input type="checkbox"/>	<a href="#">44</a>	62.79	Erastus	SNDE	2025-01-17T...	3.4
<input type="checkbox"/>	<a href="#">39</a>	83.5	Cassandra	NEO	2025-01-17T...	5.7
<input type="checkbox"/>	<a href="#">10</a>	44.01	Lauree	ZX	2025-01-17T...	9.4
<input type="checkbox"/>	<a href="#">31</a>	91.37	Ellery	OPOF	2025-01-17T...	8.8
<input type="checkbox"/>	<a href="#">3</a>	81.79	Alia	HOLX	2025-01-17T...	1.4
<input type="checkbox"/>	<a href="#">38</a>	66.52	Robinia	AAME	2025-01-17T...	3.7
<input type="checkbox"/>	<a href="#">2</a>	62.57	Sibilla	PBI^B	2025-01-17T...	4.8
<input type="checkbox"/>	<a href="#">21</a>	45.46	Aurthur	SNI	2025-01-17T...	5.5
<input type="checkbox"/>	<a href="#">40</a>	23.41	Andreas	FNBG	2025-01-17T...	4.5

Los datos que quedaron guardados se encuentran limpios y listos para poder visualizar en cualquier reporte

Para desplegar el micro servicio en aws nuevamente usamos el comando **serverless deploy**

## Descripción del Microservicio de Data-retrieval-app

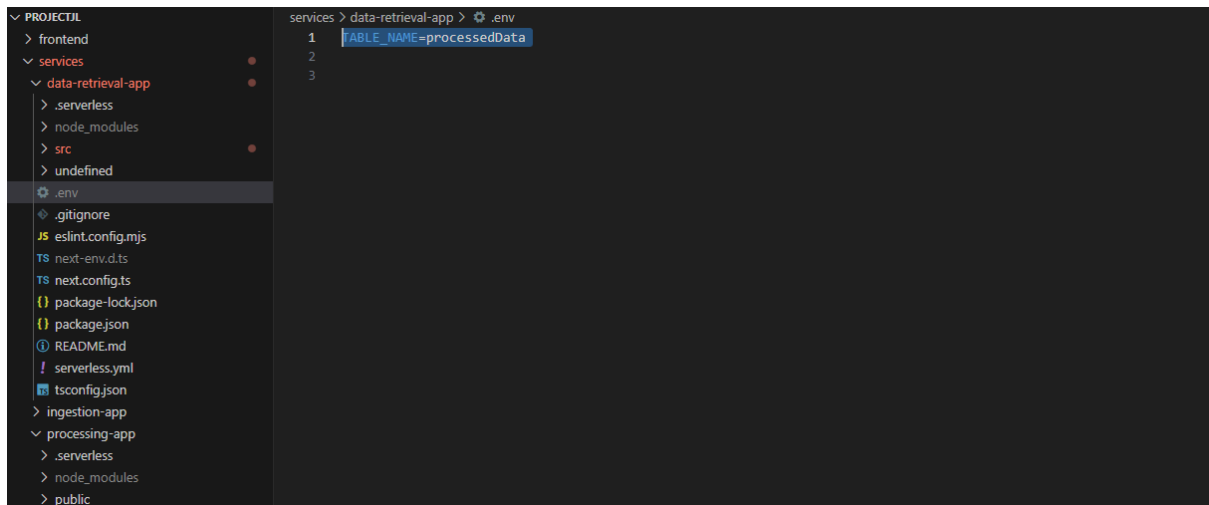
Este micro Servio cumple con la funcionalidad de extraer los datos limpios que se encuentran en nuestra base de datos Dynamobd y los expone como resulta en la ejecución de la lambda.

# Ejecución

Para probar esta lambda ejecutamos el comando en la consola npx serverless offline

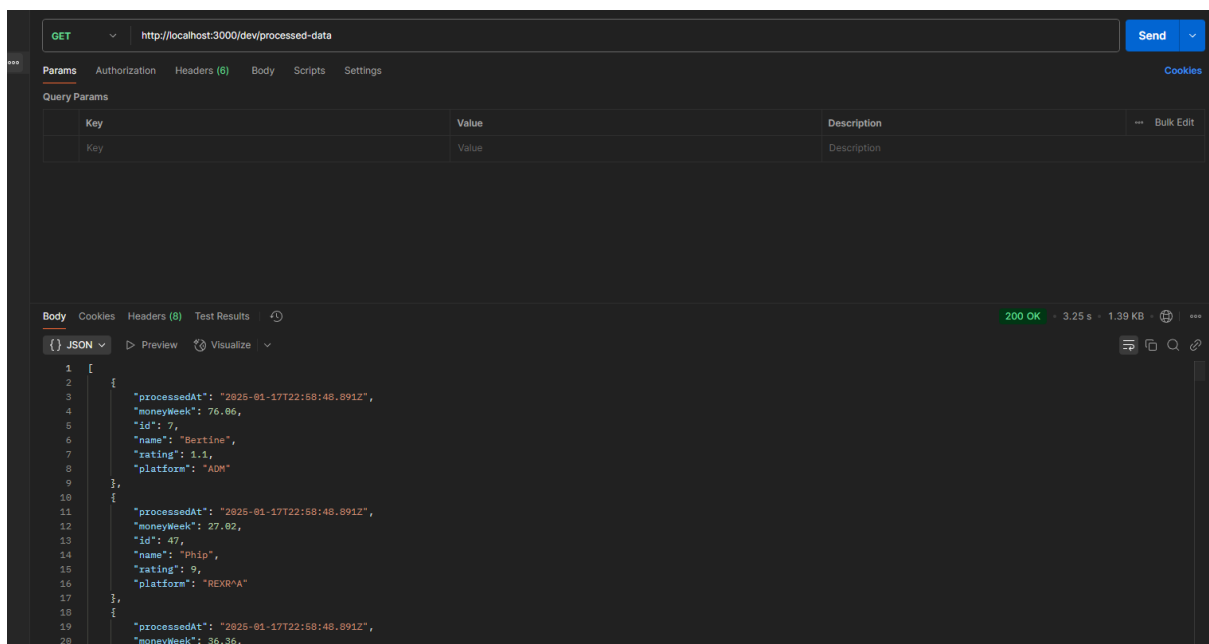
Y adicona creamos un archivo que contenga las siguientes variables de entorno

TABLE\_NAME=processedData



Una vez ejecutado el comando en nuestro navegador o aplicación de consumo web

ejecutamos la url





Para probar esta lambda la podemos usar en el navegador con el siguiente enlace <https://chp7rfq8li.execute-api.us-east-2.amazonaws.com/dev/processed-data>

Para desplegar el micro servicio en aws nuevamente usamos el comando `serverless deploy`

## Descripción aplicación frontend

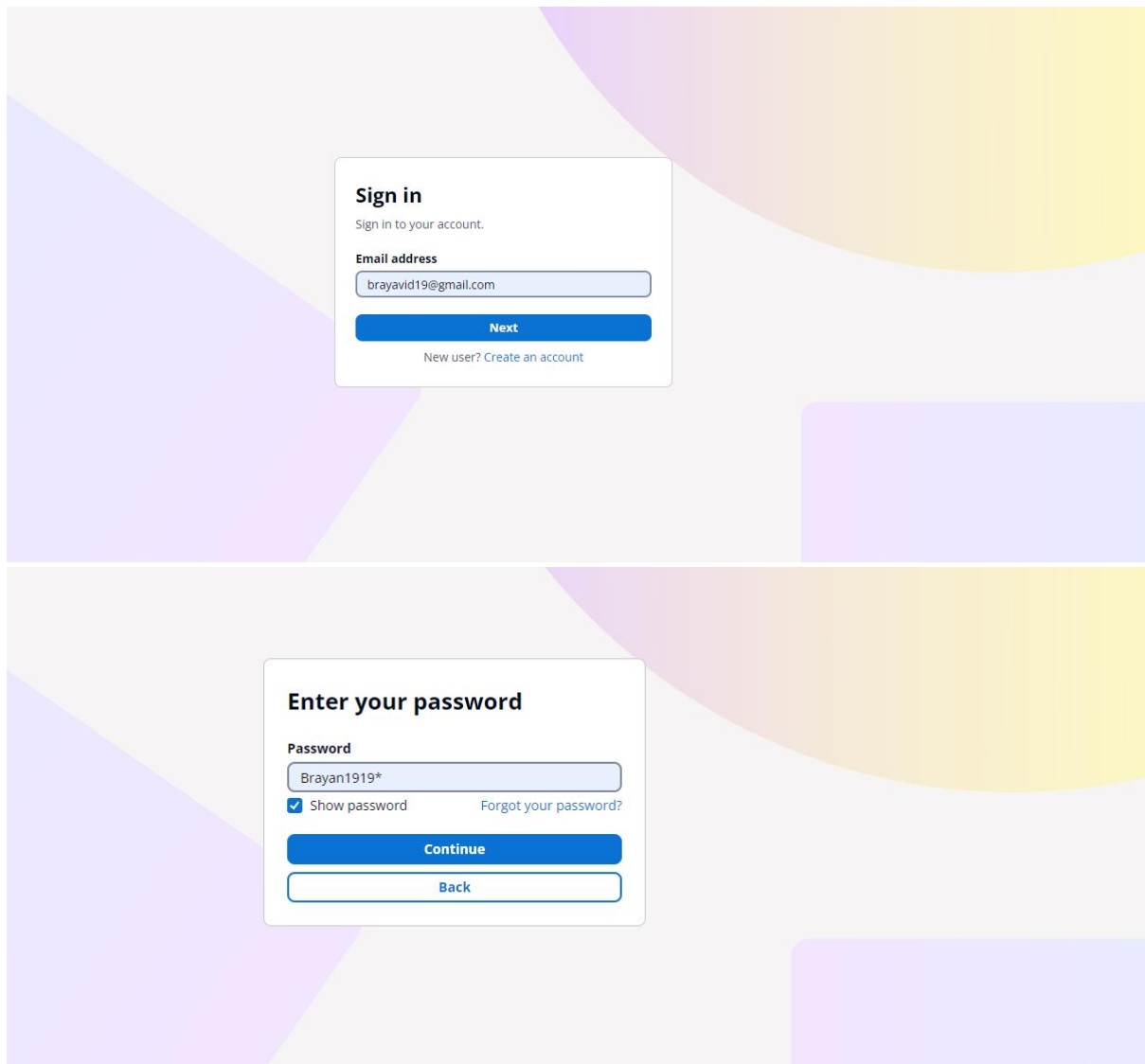
Esta aplicación crea con react cuenta con un login integrado en cognito de aws

Y se encarga de mostrar un pequeño reporte de los datos limpios que extraimos de DynamoDB

Para probar este frontend ejecutamos el comando `npm run dev` y iniciamos session con el usuario :

User : [brayavid19@gmail.com](mailto:brayavid19@gmail.com)

Contraseña : brayan1919\*



Una vez iniciada session nos mostrara un pequeño informe de la información extraída  
De nuestra base de datos dynamoDB

## Weekly Reports

Total Money Generated This Week: \$2989.29

ID	Name	Platform	Rating	Money Week	Processed At
7	Bertine	ADM	1.1	\$76.06	17/1/2025
47	Phip	REXR*A	9	\$27.02	17/1/2025
8	Chloe	BANC*D	9.5	\$36.36	17/1/2025
32	Dolmore	LKQ	6.3	\$75.76	17/1/2025
44	Erastus	SNDE	3.4	\$62.79	17/1/2025
39	Cassandra	NEO	5.7	\$83.50	17/1/2025
10	Lauree	ZX	9.4	\$44.01	17/1/2025
31	Ellery	OPOF	8.8	\$91.37	17/1/2025
3	Alia	HOLX	1.4	\$81.79	17/1/2025
38	Robinia	AAME	3.7	\$66.52	17/1/2025
2	Sibilla	PBI*B	4.8	\$62.57	17/1/2025
21	Aurthur	SNI	5.5	\$45.46	17/1/2025
40	Andreas	FNBG	4.5	\$23.41	17/1/2025
14	Sauveur	CLRBW	4.8	\$16.07	17/1/2025
18	Gunner	OTIC	4.3	\$79.00	17/1/2025