


# Proceso de Alta de Usuarios con AWS Lambda, SQS y Servicios Externos

Este documento describe el flujo de procesamiento de alta de usuarios a través de la cola `sqs_coto_superapp_alta_usuarios`, usando una función Lambda `lambda-superapp-alta-user`, y la interacción con sistemas externos como Daxia, Cognito y Comunidad Coto.

## 1. Microservicio Orquestador - `MSOnBoarding`

El microservicio `MSOnBoarding` es responsable de orquestar el alta del usuario generando **3 tipos de mensajes**, que se colocan en la **cola SQS** `sqs_coto_superapp_alta_usuarios`:

Tipo de mensaje	Descripción	Receptor
<code>daxia</code>	Contiene datos completos del usuario	<code>lambda-superapp-alta-user</code>
<code>cognito</code>	Instrucción para marcar usuario como creado	<code>cognito_service</code>
<code>coto_community</code>	Datos para registrar usuario en Comunidad Coto	<code>coto_community_service</code>

 Ejemplos de mensajes en SQS:

### ◇ Mensaje Daxia

```
{
  "meta": {
    "transaction-id": "string",
    "type": "daxia",
    "date": "string",
    "transmitter": "MsOnBoarding",
    "receptor": "lambda-superapp-alta-user",
    "action": "processing_user_creation",
    "user_id": "string"
  },
  "data": {
    "name": "string",
    "surname": "string",
    "email": "string",
    "phone_number": "string",
    "username": "string",
    "user_gender": "string",
    "government_identification": "string",
    "government_identification2": "string",
    "government_identification_type": "string",
    "public_id": "string",
    "nationality": "string",
    "additional_info": "string",
    "birth_date": "string"
  }
}
```

```
}  
}
```

#### ◇ Mensaje Cognito

```
{  
  "meta": {  
    "transaction_id": "string",  
    "type": "cognito",  
    "date": "string",  
    "transmitter": "MsOnBoarding",  
    "receptor": "cognito_service",  
    "action": "processing_user_creation",  
    "user_id": "string"  
  },  
  "data": {  
    "user_name": "string"  
  }  
}
```

#### ◇ Mensaje Comunidad Coto

```
{  
  "meta": {  
    "transaction_id": "string",  
    "type": "coto_community",  
    "date": "string",  
    "transmitter": "MsOnBoarding",  
    "receptor": "coto_community_service",  
    "action": "processing_user_creation",  
    "user_id": "string"  
  },  
  "data": {  
    "name": "string",  
    "lastName": "string",  
    "DocumentTypeId": "number",  
    "documentNumber": "string",  
    "sexId": "number",  
    "NationalityId": "number",  
    "maritalState": "number",  
    "birthDate": "string",  
    "email": "string",  
    "cellPhoneNumber": "string",  
    "areaCode": "string",  
    "OriginId": "number",  
    "ip": "string"  
  }  
}
```

## ⚙️ 2. Lambda `lambda-superapp-alta-user`

🔔 Trigger:

- Se activa cuando ocurre un `PutItem` en la cola `sqs_coto_superapp_alta_usuarios`.

🧠 Funcionalidad:

- La Lambda **lee todos los mensajes disponibles** en la cola, no solo el que disparó el evento.
- Evalúa cada mensaje y registra/controla su estado en la tabla DynamoDB `control_messages`.

📄 Tabla `control_messages`:

Campo	Descripción
<code>id</code>	Identificador
<code>user_id</code>	Usuario relacionado
<code>daxia_receipt_date</code>	Fecha recepción Daxia
<code>cognito_receipt_date</code>	Fecha recepción Cognito
<code>coto_community_receipt_date</code>	Fecha recepción Comunidad Coto

## ✏️ 3. Escenarios de Procesamiento

### ☑️ 1. `meta.type == daxia`

- Si `user_id` NO existe** en la tabla:
  - Enviar datos a `MSUser API` para creación.
  - Si la respuesta es `200 OK`, registrar en base de datos y hacer `acknowledge` del mensaje.
  - Si NO es `200`, NO registrar ni `acknowledge`.
- Si `user_id` YA existe** en la tabla:
  - Generar log técnico con `transaction_id` y `user_id`.

### ☑️ 2. `meta.type == coto_community`

- Si `user_id` NO existe** en la tabla: ❌ No hacer nada, el mensaje queda en cola.
- Si `user_id` SÍ existe** y:
  - `daxia_receipt_date`  $\neq$  `NULL` y `coto_community_receipt_date`  $==$  `NULL`:
    - Registrar usuario en Comunidad Coto.
    - Si respuesta es exitosa, actualizar `coto_community_receipt_date` y `acknowledge`.
    - Si no es exitosa, NO guardar ni `acknowledge`.
  - `daxia_receipt_date`  $\neq$  `NULL` y `coto_community_receipt_date`  $\neq$  `NULL`:
    - Generar log técnico de duplicado.

☑ 3. `meta.type == cognito`

- Si `user_id` **NO existe** en la tabla: ❌ No hacer nada, el mensaje queda en cola.
- Si `user_id` **SÍ existe** y:
  - `daxia_receipt_date ≠ NULL` y `cognito_receipt_date == NULL`:
    - Llamar a **Cognito API** para marcar `isCreated=true`.
    - Si respuesta es exitosa, actualizar `cognito_receipt_date` y `acknowledge`.
    - Si no es exitosa, NO guardar ni `acknowledge`.
  - `daxia_receipt_date ≠ NULL` y `cognito_receipt_date ≠ NULL`:
    - Generar log técnico de duplicado.

---

🧠 Consideraciones Generales

- 🧑 Cada mensaje debe ser evaluado por separado, y la ejecución depende de los **campos de control de la tabla**.
- 📄 Los mensajes de tipo `cognito` y `coto_community` **dependen** de que el mensaje `daxia` haya sido procesado y registrado previamente.
- 🗄 Se guardan los campos de `meta.user_id` y la fecha de recepción según el `meta.type`.

---

📖 Ejemplo de Logs Técnicos

- 📄 **Daxia repetido**: "Usuario `123` ya fue creado. Se recibió duplicado. `transaction_id`: abc"
- 📄 **Coto/Cognito repetido**: "Ya se había procesado `isCreated` o registro en Comunidad Coto para `user_id=123`."

---

📌 Resumen

Tipo de mensaje	Acción principal	Depende de
<code>daxia</code>	Crear usuario en MSUser	Ninguna
<code>cognito</code>	Marcar <code>isCreated=true</code>	Usuario creado ( <code>daxia_receipt_date ≠ NULL</code> )
<code>coto_community</code>	Crear en Comunidad Coto	Usuario creado ( <code>daxia_receipt_date ≠ NULL</code> )

---

👤 Autor

Documentado por Jiliar Antonio Silgado Cardona – Tech Manager

---