



API NÓMINAS

API GATEWAY, LAMBDA, POSTMAN

EDUARDO ALCALÁ HUERTA

- 01** Resumen y Objetivo
- 02** Procesamiento XML
- 03** Tecnologías
 - API Gateway
 - Lambda
 - DynamoDB
 - Postman
- 04** Conclusiones

RESUMEN

Este documento servirá como representación de las características generales y consideraciones de la api de nóminas en desarrollo apartir de los conocimientos y aplicación práctica de bases de datos NoSQL, visualizadores de datos y procesamiento de datos.



En la actualidad la cantidad de información en el mundo crece con gran velocidad, es por esto que su almacenamiento se ha visto limitado en cuanto a características como el volumen, velocidad y escalabilidad de la información, siendo esto el motivo principal para problemas relevantes como la pérdida de información y sobre costos.

Teniendo en cuenta lo anterior, se puede afirmar que la información tiene un enorme poder dentro de muchas áreas, específicamente en la toma de decisiones en cualquier organización, puesto que proporciona una ventaja competitiva o una oportunidad de negocio.

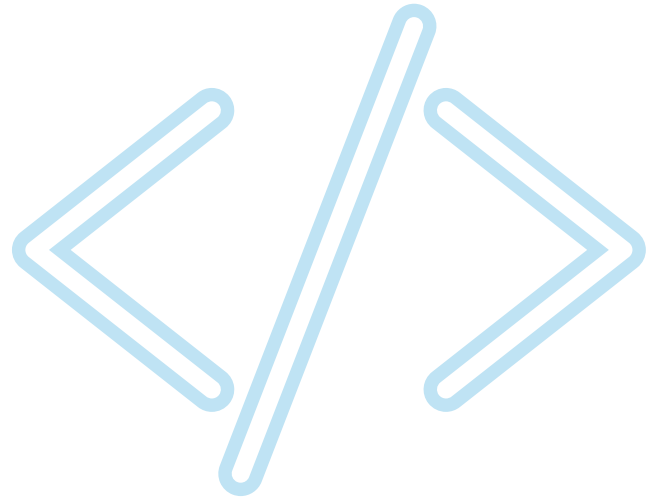
Una posible solución al problema de almacenamiento para grandes volúmenes de datos y su consulta de manera rápida y eficiente es la incorporación de nuevos enfoques tecnológicos en cuanto al almacenamiento, tal enfoque es conocido como bases de datos NoSQL,



DOCUMENTACIÓN

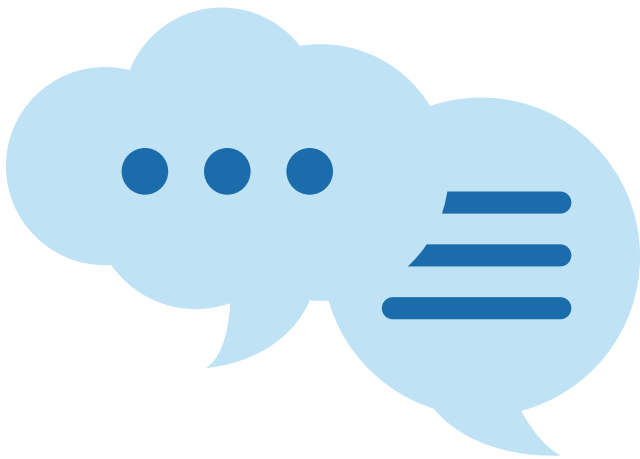
La documentación puede ser consultada en el siguiente repositorio:

<https://github.com/EduardoContpaqi/APIemployees>



API NÓMINAS

Para cualquier duda relacionada, puedes escribir a los siguientes emails:
eduardo.alcala@contpaqi.com
ramiro.arellano@contpaqi.com



LICENCIA

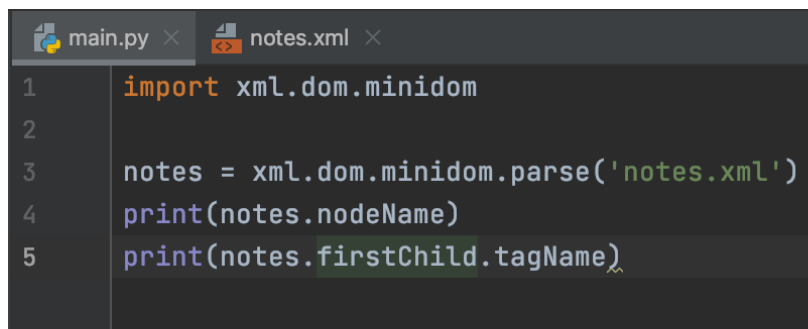
El proyecto está licenciado bajo la licencia GNU GPLv3.



PROCESAMIENTO XML

Para el procesamiento y transformación de los archivos xml, se utilizaron dos herramientas principalmente, Python y Spoon.

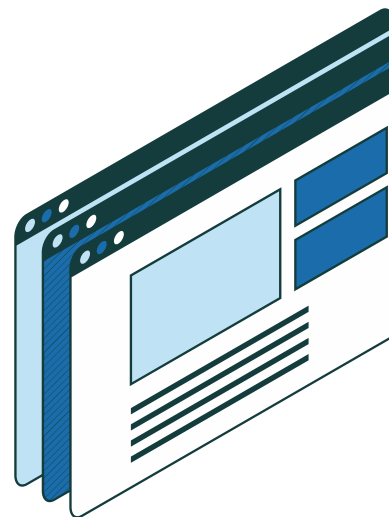
Para el procesamiento mediante Python se utilizó la librería "Minidom" la cuál es una implementación mínima de la interfaz Document Object Model (Modelo de objetos del documento), con una API similar a la de otros lenguajes, el código se puede obtener directamente del repositorio en línea mencionado en la documenteción.

A screenshot of a code editor with two tabs: 'main.py' and 'notes.xml'. The 'main.py' tab is active, showing five lines of Python code. Line 1: `import xml.dom.minidom`. Line 2: (empty). Line 3: `notes = xml.dom.minidom.parse('notes.xml')`. Line 4: `print(notes.nodeName)`. Line 5: `print(notes.firstChild.tagName)`.

```
1 import xml.dom.minidom
2
3 notes = xml.dom.minidom.parse('notes.xml')
4 print(notes.nodeName)
5 print(notes.firstChild.tagName)
```

La documentación técnica de la librería se puede consultar en el siguiente enlace:
<https://docs.python.org/es/3.9/library/xml.dom.minidom.html>.

La función principal de la librería es la función `parse()` que puede tomar un nombre de archivo como parámetro para su procesamiento, retorna un objeto Document que representa el contenido del documento xml.



PROCESAMIENTO XML

Para el procesamiento mediante Spoon se realizó una lectura mediante la herramienta "Get XML Data", por cada ubicación de "Loop XPath" que se encuentre en los archivos XML, se generará una fila de datos (atributos) correspondientes a la etiqueta.

Name	XPath
RfcEmisor	/cfdi:Comprobante/cfdi:Emisor/Rfc
NombreEmisor	/cfdi:Comprobante/cfdi:Emisor/Nombre
RegimenFiscalEmisor	/cfdi:Comprobante/cfdi:Emisor/RegimenFiscal
RfcReceptor	/cfdi:Comprobante/cfdi:Receptor/Rfc
NombreReceptor	/cfdi:Comprobante/cfdi:Receptor/Nombre
UsoCFDIReceptor	/cfdi:Comprobante/cfdi:Receptor/UsoCFDI

La documentación técnica de la herramienta se puede consultar en el siguiente enlace:
<https://wiki.pentaho.com/display/EAI/Get+Data+From+XML>

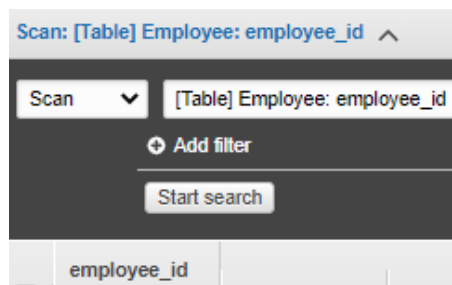
Una vez realizado el proceso, se obtienen dos salidas, una correspondiente a la data completa en formato CSV y adicional una salida en formato JSON para su posterior visualización. En la documentación del repositorio en línea, se puede encontrar un vídeo con el procedimiento a detalle de la herramienta.



DYNAMODB

DynamoDB es una base de datos de documentos y llave - valor que ofrece un gran rendimiento a cualquier escala. Es una base de datos duradera, multiregión, multiactiva y no tiene servidores que mantener, parchear o administrar, ni software que instalar, mantener u operar.

Se crea una tabla en la base de datos para poder realizar operaciones sobre ella, simplemente se elige el nombre de la tabla y se asigna una llave primaria, la tabla no necesita tener un esquema definido por lo que podemos insertar cualquier información necesaria.



En Amazon DynamoDB, se utiliza PartiQL como medio de consulta, es un lenguaje de consulta compatible con SQL, o las API clásicas de DynamoDB que es el caso para este ejercicio, se realizó la carga de información por medio de la función PutItem, que provee soporte nativo para documentos por medio del formato JSON, que puede obtener y manipular datos a través del formato.

LAMBDA

Lambda es una plataforma basada en eventos que se ejecuta sólo cuando se activa y acciona el código que se ha cargado en el sistema.

En las arquitecturas basadas en la nube, hay que ser muy conscientes de los niveles de seguridad y disponibilidad del proyecto, por lo que lambda se convierte en una gran opción para poder desplegar código de forma rápida, potente y con una fácil integración con el resto de servicios entre plataformas. Para el caso de esta prueba de caso de uso, se realizó un ejemplo de inserción de información con código python, se muestra a continuación:

```
lambda_function
1 import boto3
2
3 dynamodb = boto3.resource('dynamodb')
4 table = dynamodb.Table('Employee')
5
6 def lambda_handler(event, context):
7     table.put_item(Item=event)
8     return{ "statusCode": 200, "message": "Employee was successfully created"}
9
```

El código consiste en el uso de la librería boto3 que facilita la integración de la aplicación, biblioteca o script de Python con los servicios de AWS. Con esta herramienta, podemos hacer distintas transacciones en la base de datos como insertar, borrar o actualizar información.

API GATEWAY

API Gateway es un servicio de AWS para la creación, publicación, mantenimiento, monitoreo y protección de API REST, HTTP y WebSocket a cualquier escala. Es posible crear APIs que accedan a AWS o a otros servicios web, así como los datos almacenados en la nube de AWS, como en este caso a DynamoDB por medio una función lambda para la inserción de información a la base de datos por medio de una estructura JSON.

Request: /employee

Status: 200

Latency: 224 ms

Response Body

```
{
  "statusCode": 200,
  "message": "Employee was successfully created"
}
```

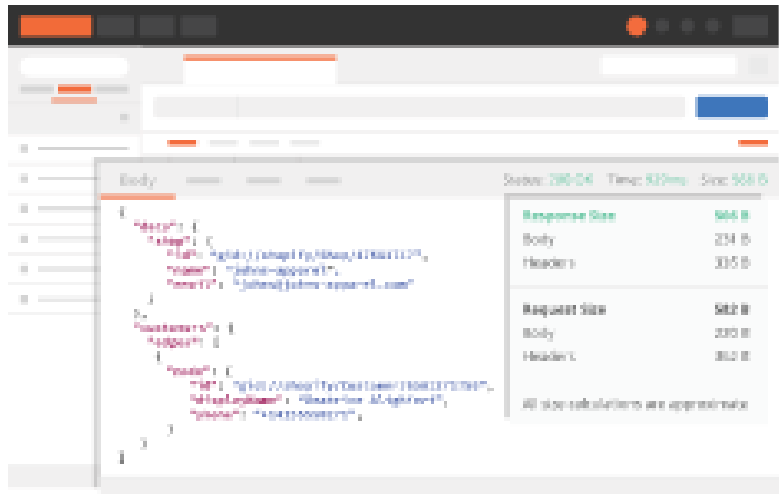
API Gateway crea una API RESTful que:

- Se basa en HTTP.
- Habilita la comunicación entre cliente y servidor sin estado.
- Implementa métodos HTTP estándar como GET, POST, PUT, PATCH y DELETE.



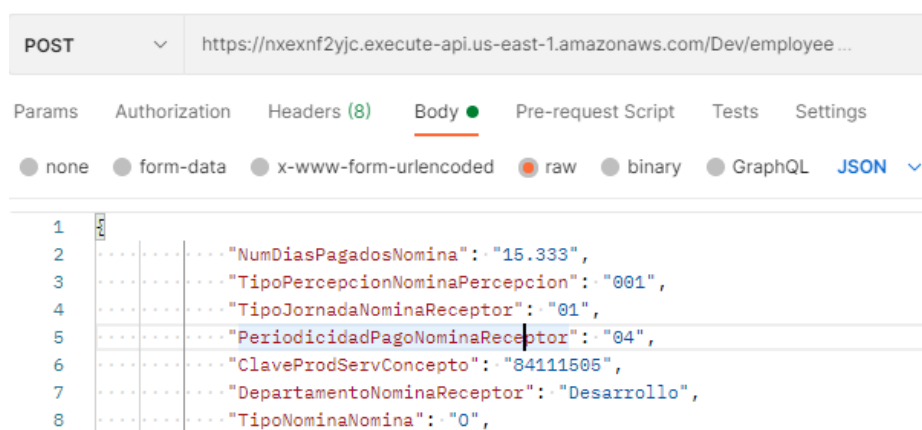
POSTMAN

Postman permite crear y ejecutar cualquier consulta REST, SOAP y GraphQL desde su interfaz.



Para este ejemplo práctico se utiliza la herramienta como medio para realizar inserción de información, por medio del método POST y un formato JSON para la información.

La documentación online se encuentra disponible en: <https://learning.postman.com/docs/getting-started/introduction/>



CONCLUSIONES

Como cualquier desarrollo, este tipo de arquitectura y diseño tiene sus beneficios e inconvenientes, pero podría destacar:

1. No es necesario realizar mantenimiento de los servidores donde se tienen instalados programas y aplicaciones. El código se ejecuta en un contenedor temporal por lo que ya no es necesario instalar software, gestionar puertos de acceso o estar pendiente de las actualizaciones.
2. Se puede realizar un escalamiento de manera horizontal tanto como se requiera. Es posible añadir todos los clusters, balanceo de cargas etc, conforme se necesite.
3. Relacionado a los costos, solamente se va a pagar por el tiempo que se estén utilizando los procesos.
4. Las funciones que se utilicen, es posible integrarlas con el resto de servicios que ofrece la plataforma, como son logging, virtualización o endpoints.