UNIVERSIDAD NACIONAL DE LA PLATA

TESINA DE LICENCIATURA

Programa de apoyo para alumnos con experiencia profesional

TÍTULO: Procesamiento eficaz de archivos de empleados utilizando tecnología serverless
AUTOR: Alejo Finocchi
DIRECTOR ACADÉMICO: Armando de Giusti
DIRECTOR PROFESIONAL: Sebastián Bachmann
CODIRECTOR ACADÉMICO:
CARRERA: Licenciatura en Sistemas

Resumen

Se actualizo un sistema ya existente realizando un nuevo procesamiento de archivos donde se utilizan distintos recursos de tecnologías serverless para obtener un sistema más estable, eficaz y eficiente. Este sistema es utilizado por distintas empresas de un cliente que provee este servicio de procesamiento de archivos para cargar los empleados a una plataforma que les permite obtener acceso a cursos educacionales a los que no tendrían acceso por medios externos.

Palabras Clave

- AWS
- Step Function
- Lambda
- SNS
- SQS
- JavaScript
- NodeJS

Conclusiones

El nuevo sistema desarrollado demostró ser mas rápido que el sistema anterior (se redujo de 29 minutos a 5 minutos el tiempo de procesamiento), tener mayor estabilidad gracias a un nuevo sistema de manejo de errores, una estructura de datos bien definida donde un cambio en un estado no afecta a otro no relacionado y una división de responsabilidades entre Lambdas de procesamiento de archivos y Lambdas de creación o actualización de empleados.

Trabajos Realizados

Se crearon dos Stacks de AWS para realizar el sistema. Un stack se ocupa solamente del procesamiento del archivo utilizando una Step Function donde se realiza las validaciones sobre el mismo y se envían distintos eventos en base a la información del archivo. Los eventos de crear o actualizar empleados son escuchados por el segundo stack que se encarga de crear o actualizar los registros y una vez que los empleados ingresen al sistema, se encarga de manejar sus estados de elegibilidad y Approval

Trabajos Futuros

1. Cambiar la base de datos no relacional por una base de datos relacional.
2. Unificar los registros de empleados con su historial de Elegibilidad.
3. Mejorar la integración con la capa de encriptación de registros.
4. Agregar la habilidad de una resubida automática de archivo en el caso de una falla externa al sistema.

Fecha de la presentación: 11/2021

Glosario:

- Cloud Computing: Es un conjunto de recursos que se guardan en la red (puede ser internet o una red interna) que son accedidos por demanda de los usuarios o sistemas que lo necesiten.

- AWS: Es un proveedor de servicios de Cloud Computing donde se encuentra alojado el sistema desarrollado.

- Lambda: Es un servicio de cómputo de AWS que se ejecuta en respuesta a eventos de AWS.

- Step Function: Es un orquestador de servicios de AWS para manejar flujos e interacciones entre distintos servicios.

- SNS: Es un servicio de envío de mensajes para envío de mensajes entre aplicaciones o hacia usuarios.

- SQS: Es un servicio de mensajes de colas donde se guardan mensajes y se disparan eventos en base a los tópicos configurados.

- S3 Bucket: Es un servicio de almacenaje que provee acceso y almacenamiento seguro a cualquier objeto que el cliente desee.

- JavaScript: Es un lenguaje de programación que cumple con la especificación ECMAScript. Es de alto nivel, a menudo compilado en tiempo de ejecución, débilmente tipado y orientado a objetos.

- Node JS: Es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome.

- Core API: Término por el que se refiere al sistema desarrollado en NestJS que contiene el acceso a la base de datos y los distintos métodos de creación y actualización de empleados.

- Corporate Partner: Término por el que se refiere a las empresas que utilizan el sistema.

Motivación:

Esta propuesta surgió como respuesta a un problema recurrente en el sistema que veníamos trabajando para un cliente que nos contrató para desarrollar un sistema específico. El sistema necesitaba proveer acceso a distintos cursos de universidades estadounidenses a empleados de distintas empresas que contrataron sus servicios.

El cliente llevaba varios años trabajando como intermediario para estas empresas estadounidenses para conseguir acceso a distintos cursos, títulos intermedios y títulos universitarios sus empleados. A estas empresas se las denomina “Corporate Partners”.

El cliente comenzó con un trabajo manual, llevando un registro manual de cada empleado, sus cursos y solicitudes. Pero a medida que su empresa comenzó a crecer junto con la demanda, se dieron cuenta que dificultaba la escalabilidad del sistema el trabajo que estaba realizando en ese momento. Por lo que decidieron automatizar el proceso.

Se desarrolló un primer sistema bastante simple y estático. Los empleados podían ingresar a una página web donde todos los programas o cursos disponibles para su empresa estaban disponibles. Podía seleccionar cualquiera de esos programas y solicitar el acceso.

Para poder acceder a la página web los empleados debían estar registrados en el sistema. La forma de registrarlos se realizaba cargando un archivo de formato “CSV” a un servidor SFTP conectado a un bucket de S3. El archivo debía contener las cabeceras requeridas por el sistema y no aceptaba cabeceras extras o de valores diferentes a los esperados.

Se realizaba el procesamiento utilizando una step function que leía el archivo, lo guardaba en una base de datos temporal si pasaba las validaciones iniciales, luego sacaba el archivo de memoria para evitar consumir muchos recursos y utilizando varios steps, iba leyendo de la base de datos temporal y agregaba uno por uno los empleados en la base de datos.

Al cargar los empleados al sistema, la empresa debía marcar cuáles eran elegibles para acceder a los cursos y cuáles no, utilizando una columna del archivo con letras “Y” o “N”.

Los mayores problemas que encontrábamos eran la lentitud del procesamiento, errores que causaban que falle todo el procesamiento del archivo y que los Corporate Partners inscritos al servicio del cliente debían acomodarse a las restricciones del procesamiento de archivos en vez del procesamiento acomodarse a las demandas del cliente. El mayor inconveniente era que la elegibilidad de los empleados a los cursos y la aprobación de sus managers estaban ligados.

Durante el desarrollo del primer sistema, surgieron nuevos requisitos por parte de los Corporate Partners que requerían que los empleados pasen por un proceso de revisión de los managers antes de acceder a los programas.

Por lo que un empleado podía acceder a la página, pedir acceso a un curso, pero si el manager se lo negaba iba a dejar de acceder al mismo. Si un manager denegaba el acceso a un curso a un empleado, esto se vería reflejado en la elegibilidad del mismo. Algo erróneo ya que un empleado cuyo acceso a un curso fue denegado, no debería perder su elegibilidad y posibilidad de acceder a otros cursos. Simplemente debía negarle el acceso al curso específico al que fue denegado por parte de su manager, pero permitirle el acceso a otros cursos.

Por lo tanto, quisimos construir un sistema más robusto, dinámico y estable para garantizar la efectividad y eficacia del sistema:

- Que supere en velocidad al anterior sin perder ningún registro en el proceso y que pueda tener un buen sistema de “Error Handling” para que cualquier fallo que pueda ocurrir durante el procesamiento sea registrado, notificado y que no cause el fallo abrupto de todo el procesamiento (en el caso que sea posible continuar con el procesamiento).
- Que fuera más versátil y dinámico, dando la oportunidad a cada Corporate Partner a subir archivos con los campos que ellos querían o necesitaban, en vez de tener que subir archivos con los campos que el sistema necesitaba. Si una de las líneas del archivo era errónea, o si las cabeceras del mismo no eran las que el sistema esperaba para ese Corporate Partner en particular, el sistema debe notificar de forma correcta cuáles eran los campos faltantes o erróneos. Pero el resto de las líneas correctas deberían ser cargadas en el sistema. Al dar la oportunidad a cada empresa para que el sistema se acomode a sus requisitos, aumenta el atractivo del mismo para potenciales clientes haciendo que aumente la demanda hacia la empresa.
- Y, por último, pero más importante, el sistema debería separar en entidades diferentes los términos de elegibilidad y aprobación para que la denegación de un curso a un empleado no afecte su elegibilidad y viceversa. Es muy importante que la experiencia del usuario sea la adecuada, por lo que los conceptos de elegibilidad al sistema y de aprobación de los cursos deben estar separados para poder representar correctamente el estado del empleado en el sistema y reflejar en su experiencia con el sistema.

Casos de uso que llevaron al pedido de un nuevo sistema:

A lo largo del periodo de vida del sistema viejo, hubo varios casos que causaron problemas para el cliente:

- Se cargaron archivos grandes de empleados con varias líneas o filas incorrectas, pero como en el procesamiento no hubo ningún error el sistema devuelve una notificación de que el archivo se había procesado correctamente. Cuando en realidad hubo varias filas que no se procesaron porque tenían valores incorrectos. Por lo que se consideraba que de un archivo de 50.000 empleados se había cargado correctamente al sistema cuando solo se había cargado 30.000. Por lo tanto, a la hora de recibir la cantidad de empleados registrados en el sistema, los números que retornaban no eran los esperados.
- Archivos con más de 100.000 registros de empleados tardaban varias horas en procesarse. Esto no solo generaba que se tarde en cargar todos los empleados en el sistema, si no que generaba un costo extra de los recursos de AWS utilizados para procesar los archivos.
- Al tener el sistema de elegibilidad y aprobación de empleados tan ligado al procesamiento de archivos, hizo que el sistema fuera poco escalable y limitando las posibilidades de hacerlo crecer o crear nuevos flujos dentro del mismo.

Objetivo:

General:

El objetivo del sistema es realizar el procesamiento de archivos de distintos Corporate Partners para que puedan cargar sus empleados a un sistema que provea servicios de educación para dichos empleados. Luego el sistema computará si los empleados son elegibles para los cursos disponibles en la plataforma.

Para cada Corporate Partner, existe una configuración dinámica para cada uno. Cada empresa va a poder personalizar los headers de cada archivo, el tipo de dato de cada columna y cuales son requeridos.

Con los campos del archivo se crean los registros de los empleados en la base de datos, con los campos del archivo guardados en los campos dinámicos del empleado. Y en base a los campos dinámicos del empleado, se podrá calcular su elegibilidad en base a ciertas reglas configuradas por cada empresa.

Los empleados que sean elegibles podrán enviar solicitudes para inscribirse a los distintos cursos, aunque dependiendo de las características de los Corporate Partners, existe la posibilidad de que estos empleados deban ser aprobados por sus managers o personal de recursos humanos. Por lo que mantener un historial de los estados de los empleados es crucial para poder saber el estado del mismo y cuando ocurrió dicho cambio. Este sistema debe ser on-demand, por lo que debe iniciarse sólo cuando se reciba el archivo de empleados y finalizar su proceso una vez que termine el procesamiento del archivo.

Una vez que se hayan cargado en el sistema, los empleados podrán utilizar una plataforma ya existente para registrarse y solicitar accesos a distintos cursos académicos. Cada empresa va a tener la opción de que todos sus empleados puedan acceder a los cursos o tener que esperar una respuesta por parte del manager o del sector de recursos humanos. Una vez que hayan sido aprobados, recién en ese momento los empleados van a poder acceder a los cursos del sistema.

Específicos:

Desarrollar un sistema que sea capaz de soportar la carga y procesamiento de archivos, utilizando tecnología Serverless. Conectar múltiples lambdas utilizando Step Functions, SQS y SNS events y buckets de S3, utilizando una API construida en Node JS utilizando el Framework NestJS y una base de datos no relacional (MongoDB)

La idea de construir el sistema que procese los archivos aprovechando todos los recursos de la tecnología base de manera eficaz y eficientemente y utilizando Lambdas es que al ser solamente ejecutadas al momento de subir un file, no es necesario tener una aplicación corriendo las 24 horas ahorrando recursos. Por lo que las Lambdas de AWS cumplen los requerimientos necesarios para el sistema.

Cada Lambda de la Step Function se ocupa de un paso del procesamiento. La validación del nombre del archivo, la integridad de los datos, validar que no haya campos faltantes y si cada row es un empleado nuevo o si es uno ya existen que debe ser actualizado. Dependiendo si es un empleado nuevo o no, se disparará un evento de creación o uno de actualización.

Luego, utilizando las queues de AWS, se pueden encolar varios eventos de creación o actualización de empleados y ejecutar paralelamente varias lambdas de creación o actualización, aumentando la velocidad del procesamiento de todo el archivo procesando paralelamente múltiples filas del archivo.

Como capa de repositorio, se tiene una REST API en NestJS donde están las conexiones a la base de datos de MongoDB que corre en un contenedor de AWS donde se realizan las consultas para ver las configuraciones de los Corporate Partners, ver si un empleado existe o si es necesario realizar una actualización sobre el mismo.

También realiza las consultas para ver si la empresa requiere un registro de "Approvals" para el acceso del empleado. En caso de ser necesario, cuando el empleado pida el acceso y llegue ese evento, se creará el registro de Approval y se esperará a la respuesta del encargado de la empresa.

Para la respuesta de los managers o referentes de recursos humanos, existen dos herramientas. Una es un email donde se tendrán 2 links (uno para aprobar el acceso y otro para denegarlo) y otra es una página desarrollada solo para administradores donde podrán ver un listado de todos los empleados, seleccionar uno o más empleados y aprobarlos (o negarlos).

Análisis del estado del Sistema:

El sistema se encuentra alojado en un sistema serverless de AWS. AWS permite guardar en la nube recursos de computación sin la necesidad de tener un servidor dedicado para el mismo. Estos recursos pueden ser archivos, información e incluso código que responde a distintos eventos.

Una Lamba es el recurso que guarda el código a ejecutarse ante cierto evento especificado. Una Lambda puede escuchar a uno o múltiples eventos y ejecutar el código como respuesta al mismo. Por lo que se creó una que escucha a eventos de subida a un bucket de S3, estos buckets se encargan de almacenar archivos en la cuenta de AWS. Una vez que un archivo es subido al bucket, una Lambda escucha por ese evento y utilizando la información del mismo, dispara una Step Function para procesar el archivo.

Una Step Function es un servicio de AWS para organizar varios recursos y procesos para que se ejecuten en serie y en relación entre ellos en vez de ser recursos aislados sin conocimientos entre el recurso anterior y el próximo. Un claro ejemplo es la comunicación entre Lambdas. Las Lambdas contienen un código que se ejecuta al disparar y que retorna o ejecuta una acción, una vez que termina el código la Lambda termina. No tiene conocimiento de cualquier otra Lambda que se esté ejecutando o el contexto, solo conoce la información que le provee el evento que la disparó.

La Step Function se ocupa de conectar cada Lambda para que el resultado de una se envíe como parte del evento que dispare a la siguiente, manteniendo información o contexto del proceso general que se está llevando a cabo. No solo sirven para conectar Lambdas, sino que pueden disparar distintos eventos y decidir con qué paso continuar dependiendo de los resultados de los pasos anteriores.

Es por eso mismo que se decidió utilizar una Step Function para procesar un archivo de empleados. En vez de tener todo el procesamiento en una función, se separó en distintas Lambdas para evitar que un fallo en un paso genere el fallo total de todo el archivo. Por lo

que el paso de validación del archivo, la carga del mismo en una base de datos temporal, la lectura y procesamiento de cada línea y el encriptado del archivo por cuestiones de seguridad se dividieron en Lambdas conectadas por Step Functions.

La Base de datos en la que se cargaba el archivo temporalmente y donde se guardaban los registros de empleados y sus solicitudes era DynamoDB. Como cada archivo de empleados podía ser dinámico, las búsquedas en Dynamo se deben realizar si o si utilizando scans a la tabla completa ya que para realizar queries Dynamo requiere que se declaren Keys para poder realizar las búsquedas sin necesidad de escanear toda la tabla.

Por lo que las búsquedas de empleados por sus campos podían tardar demasiado tiempo, realizando el proceso de búsquedas del mismo para actualizarlo más lento.

Ante cada tabla nueva, había que configurarles las Keys necesarias para poder realizar las queries. Pero no podían crearse múltiples Keys ya que a partir de cierta cantidad AWS comenzaba a cobrar por cada Key extra. Haciendo que DynamoDB se convierta en un recurso muy caro por cuestiones de tiempo y monetarias.

Las Lambdas de las Step Functions también se convirtieron en un recurso muy caro. Cada Lambda de lectura y escritura de empleados se realizaba de a batches de a 500 empleados. Y cargaba o actualizaba de a un registro en la base de datos, cargaba secuencialmente los 500 registros en vez de hacer un solo request a la base de datos para escribir los 500 registros con un solo request. Las ejecuciones de una Lambda se cobran por tiempo de ejecución y no cantidad de ejecuciones.

Por lo que una instancia de Lambda ejecutándose por 10 minutos, sale más caro que 10 instancias de Lambdas ejecutándose por 1 minuto cada una. Entonces tener 3 instancias de una Lambdas procesando 1500 empleados resultaba más caro que tener 150 instancias procesando esos 1500 empleados.

Ventajas:

- Proceso separado en distintas funciones para aislar los procesos y no causar conflictos entre ellos.
- El archivo era validado en los pasos iniciales y el procesamiento de cada línea era realizado luego de las validaciones. Por lo que no podía ocurrir que un archivo completamente invalido se intente procesar ya que era detectado previamente.
- La carga del archivo en una base temporal hace que se pueda persistir en un recurso aparte y no necesariamente en memoria.
- Al estar en una Step Function, el sistema no está constantemente corriendo y no gasta recursos de manera innecesaria.

Desventajas:

- El proceso es lento. La carga del archivo en la base temporal y la lectura del mismo tarda mucho tiempo y son los 2 pasos más lentos de todo proceso, llegando a tardar horas.
- La base de datos era ineficiente para realizar búsquedas amplias de empleados.
- La falla en una de los pasos causaba que todo el proceso falle, haciendo que quede gran parte o todo el archivo sin procesar, la base de datos temporal no era borrada y el archivo se guardaba sin encriptar.
- No había un buen sistema de notificación ante las fallas, por lo que resultaba muy difícil encontrar las causas de los problemas del procesamiento.
- Los conceptos de Elegibilidad de un empleado y Approval no fueron concretos a la hora de planificar el sistema, por lo que la experiencia de usuario de los empleados no era la esperada.

Si el archivo fallaba en procesar algunas líneas no por cuestiones del sistema, sino por incompatibilidad de los tipos de datos dentro de las líneas, se generaba un archivo con el número de la línea que no se pudo procesar, el mensaje de error y el tipo de atributo esperado.

Este archivo se genera en el mismo Bucket de S3 del Corporate Partner, en una carpeta especial donde se guardaban estos archivos de errores. El archivo de errores de nombre tenía la fecha y nombre del archivo del archivo procesado. Por cuestiones de seguridad, este archivo no contiene el valor original de la línea que falló ya que podría contener información personal de un empleado.

Una vez que el archivo era cargado en el sistema, los empleados Elegibles pueden registrarse al sistema y solicitar el acceso a distintos cursos o programas disponibles. En el caso que el Corporate Partner requiera confirmar el acceso del empleado al curso, el manager del mismo deberá confirmar si el empleado puede acceder al curso.

Existe una aplicación creada con React desarrollada por otro equipo, donde los usuarios de cada Corporate Partner deben registrarse. Existe una página por cada Corporate Partner y un empleado solo va a poder registrarse en la página de su Corporate Partner. No va a poder registrarse en una página errónea. Una vez registrados, van a tener acceso a un catálogo de cursos y programas disponibles para su Corporate Partner. Cuando seleccione un programa, un formulario se va a cargar y va a tener que llenarlo para que su solicitud sea enviada.

Esa solicitud se lo denomina “Request For Information” (RFI), se guarda en la base de datos y se asocia al registro del empleado.

Un empleado puede enviar múltiples RFI y estos se asocian al registro de empleado guardando el id del mismo. Si el Corporate Partner no requiere que la solicitud se revise, se enviará automáticamente al Academic Partner.

En caso de que requiera revisar la solicitud y esperar al Approval del manager, se pondrá el RFI en un estado “PENDING” hasta que sea revisado y se le asocia un registro de Approval. El Approval tiene 3 estados:

- PENDING
- APPROVED
- DENIED

PENDING es su estado inicial. Al crearse el registro y mandarse la solicitud al manager, el estado se inicializará en PENDING a la espera de la respuesta. Cuando se crea el registro, se envía un evento a una plataforma llamada Iterable. Iterable se utiliza para el envío y comunicación de mensajes y una de sus herramientas más prácticas es el envío automático de emails.

Cuando recibe un evento de APPROVAL_REQUESTED, enviará un email al manager del empleado con dos links, uno para aprobar y otro para denegar el acceso del curso al empleado. En base de que link seleccione el manager, el registro de Approval se actualizará a APPROVED o DENIED y se actualizará el registro de RFI.

Si el manager aprobó el acceso, la solicitud de RFI se enviará al Academic Partner y se actualizará su estado a SENT y la elegibilidad del mismo se actualizará a ELEGIBLE. En cambio, si el manager denegaba el acceso, la solicitud no era enviada y quedaba en estado DENIED.

Además de no enviar la solicitud al Academic Partner, si el manager denegaba el acceso al curso, el empleado pasaba a ser No Elegible por lo que se le denegaba el acceso a todo el sistema. Esto a nivel de negocio resultó ser algo erróneo. La elegibilidad de un empleado que le permite acceder al sistema no debería verse afectada por la decisión del manager de permitirle acceder a un curso u a otro. Un empleado elegible cuyo acceso al curso X fue denegado, debería poder acceder al sistema para poder requerir el acceso al curso Y.

La mayoría de los Corporate Partners maneja solo 3 estados de elegibilidad. Elegible, No Elegible y Revoked (Cuando un empleado deja la compañía). Pero para los Corporate Partners que requieren aprobar los accesos existen 4 estados:

- PENDING: Estado inicial de un empleado.
- PENDING_APPROVAL: Estado del empleado luego de que mande un RFI.
- ELEGIBLE: Estado de un empleado luego de ser aprobado por su manager.
- NOT_ELIGIBLE: Estado de un empleado luego de no ser aprobado por su manager.
- REVOKED: Estado de un empleado que no trabaja más para el Corporate Partner.

Esto generó varios problemas ya que resultó que para los Corporate Partners que requerían el flow de Manager Approval consideraban sus empleados en estado PENDING y PENDING_APPROVAL como empleados elegibles. Entonces no debían perder el acceso al sistema si su solicitud era denegada y pasaban a ser NOT_ELIGIBLE.

La elegibilidad de un empleado sólo debería verse afectada por el procesamiento de un archivo y nada más.

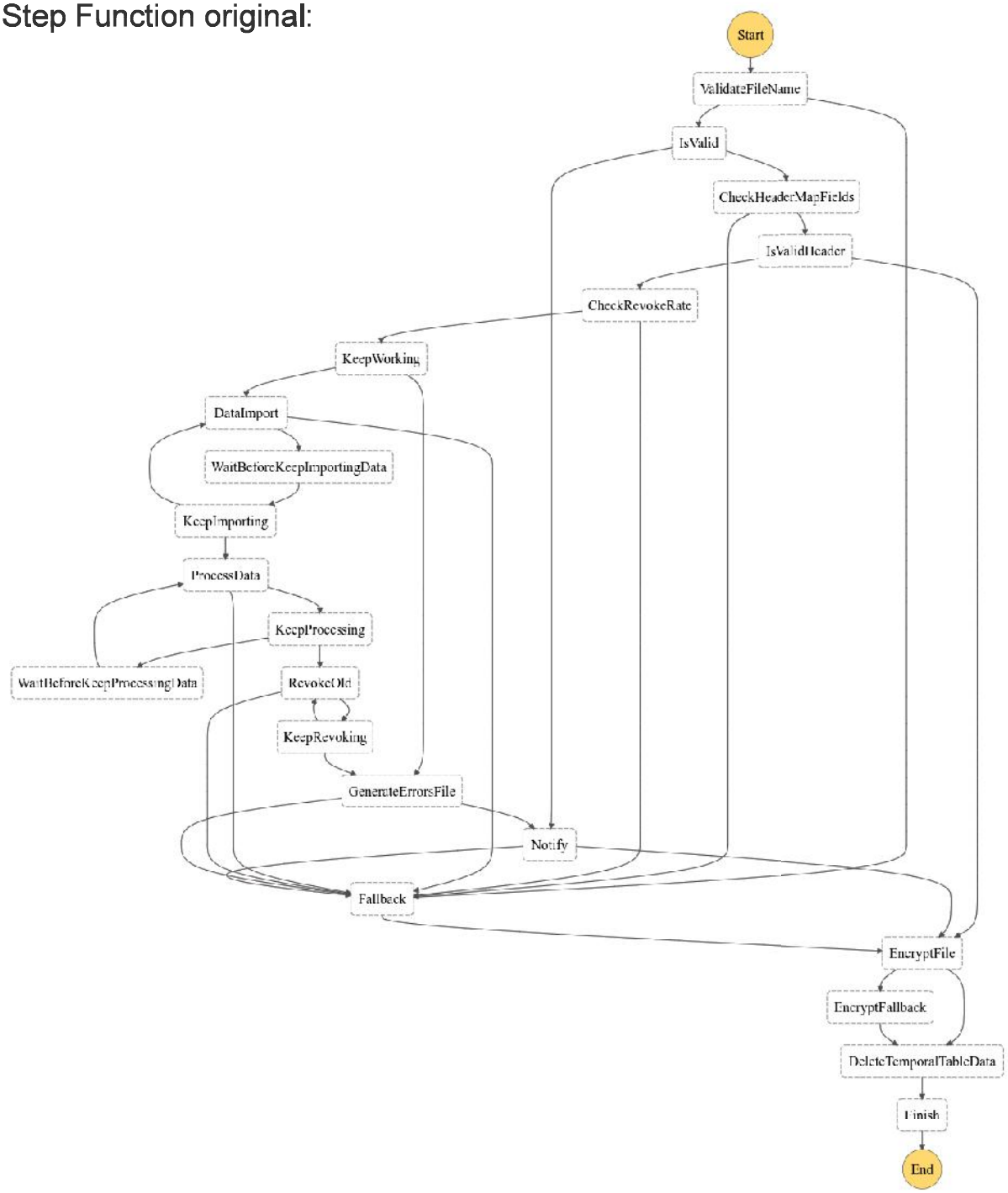
En el transcurso del tiempo, si un archivo era subido cambiando la Elegibilidad de un empleado a No Elegible, el mismo perdía el acceso al sistema, pero podía seguir accediendo a los cursos a los que ya había sido aprobado.

Esto es posible ya que los cursos provistos son cursos externos por los Academic Partners, por lo que una vez que dan el acceso el empleado tendrá acceso al mismo hasta que el curso o programa termine. A causa de eso, para corregir el estado de distintos empleados hubo que crear scripts o acceder manualmente a la base de datos para actualizar los estados de los empleados que se vieron afectados por esta falla en la lógica.

Definiciones y recursos:

Originalmente, se había desarrollado una sola Step Function para el procesamiento de archivos, pero luego al surgir el requerimiento de un flujo de Approvals de cursos y la necesidad de nuevos estados de Elegibilidad, se creó una segunda Step Function que daba soporte al flujo nuevo de Approval. Algunos de los steps de cada Step Function utilizan las mismas Lambdas, pero otros steps (en especial los de procesamiento de empleados) difieren ya que no soportan los mismos flows.

Step Function original:



Lambdas utilizadas y conexiones:

Esta step function utiliza 11 Lambdas para el procesamiento del archivo:

1. EmployeeImportValidateFileNameFunction: Esta Lambda se ocupa de validar el nombre del archivo. Los archivos deben llamarse “eligibility_{crmSystem}” donde crmSystem puede ser cualquier palabra sin caracteres especiales (“.”, “\”, “-”, “_”, etc).

2. CheckHeaderMapFields: Esta Lambda se ocupa de validar que todos los headers requeridos del archivo existan. Si un header requerido no está, no se procesa el archivo ya que automáticamente todas las líneas serían inválidas.

3. CheckRevokeRate: Esta Lambda válida la cantidad de empleados del archivo. Si la cantidad de empleados faltantes es mayor al 10% se toma como que no es un archivo válido porque estaría “revocando” a una gran cantidad de empleados. El proceso de revocar es marcar cuáles empleados dejaron de trabajar para el Corporate Partner.

4. DataImport: Esta Lamba carga en la base de datos el archivo para guardarlo temporalmente.

5. ProcessData: Esta Lambda lee un batch del archivo cargado en la base de datos y lo procesa. Si la línea es válida, crea o actualiza el empleado, pero si es invalida marca la línea como inválida.

6. RevokeOld: Esta Lambda marca como revocados a los empleados que no hayan venido en el archivo.

7. GenerateErrosFile: Esta Lambda crea un archivo con las líneas inválidas, marcando cuál fue el error que tuvo esa línea.

8. Notify: Esta Lambda manda un email usando SNS para notificar a los encargados de que terminó el procesamiento del archivo.

9. EncryptFile: Esta Lambda encripta el archivo por cuestiones de seguridad, así si alguna persona que no debe acceder al bucket de S3 no puede leer la información privada de los empleados.

10. DeleteTemporalTableData: Esta Lambda borra la información que se cargó en la tabla temporal para que no quede data duplicada ahí.

11. FallBack: Está Lambda funciona como un método de error Handling, por si falla algunos de los pasos se ocupa de que siempre se encripta el archivo.

Los steps de validación de nombre, headers y cantidad de empleados a revocar están conectadas con steps de tipo “Choice”. Los steps de tipo Choice deciden cual es el próximo paso a correr de la step function en base a los resultados obtenidos del paso anterior. Por lo que, si el archivo no es válido por alguna de las 3 razones anteriores, se pasa derecho al paso de encriptar el archivo y no intenta procesarse.

Los steps de DataImport, ProcessData y RevokeOld van a un paso de Choice para ver si se tiene que volver a llamar el paso anterior, ya que los archivos pueden ser muy grandes y los accesos a la base de datos se hacen de a batches. Por lo que se lleva un conteo y si el batch que se cargó no fue el último, se sigue loopeando hasta que todos los batches se hayan cargado en la base de datos o se hayan procesado.

El paso de ProcessData tiene un step extra que es WaitBeforeKeepProcessingData. Ese paso se creó para que haya un delay de 2 minutos entre procesamiento de batches ya que el sistema no podía procesar grandes cantidades de empleados muy seguido y necesitaba ese tiempo de espera para que no se genere cuello de botella.

Step Function para flujo de Approval:

Lambdas utilizadas y conexiones:

Esta step function utiliza 10 Lambdas para el procesamiento del archivo:

1. ValidateFileName: Este step utiliza la misma Lambda para validar el nombre que la Step Function anterior.

2. CheckHeaderMapFields: Este step utiliza la misma Lambda para validar los headers que la Step Function anterior.

3. DataImport: Este step utiliza la misma Lambda para cargar en la tabla temporal el archivo que la Step Function anterior.

4. ProcessData: Esta Lambda se encarga de procesar los empleados. Utiliza una Lambda diferente a la anterior ya que este flujo tiene nuevos estados de Elegibilidad (PENDING y PENDING_APPROVAL).

5. TerminateOldEmployees: Esta Lamba marca los empleados como “terminados”. Reemplaza la Lambda que marca los empleados como REVOKED. Esta Lamba los marca como NOT_ELIGIBLE y les pone el estado de terminated en “true”.

6. GenerateErrorsFile: Este step utiliza la misma Lambda para crear el archivo de errores que la Step Function anterior.

7. Notify: Este step utiliza la misma Lambda para notificar que se terminó de procesar el archivo que la Step Function anterior.

8. EncryptFile: Este step utiliza la misma Lambda para encriptar el archivo que la Step Function anterior.

9. DeleteTemporalTableData: Este step utiliza la misma Lambda para borrar la tabla temporal que la Step Function anterior.

10. FallBack: Este step utiliza la misma Lambda para corregir los errores que la Step Function anterior.

Esta step function funciona del mismo método que la step anterior. La mayor diferencia entre una y otra es las nuevas reglas de “Approval” de empleado por lo que no se podían reutilizar las Lambdas de procesamiento de la step anterior. Esas Lambdas no procesaban estados diferentes a ELIGIBLE o NOT ELIGIBLE, por lo que para diferenciar de un flow u otro de crearon dos Step Functions diferentes donde el procesamiento inicial (validación de nombre y formato, lectura del archivo, carga del mismo en la base de datos temporal) se mantienen igual.

Los pasos que difieren son el procesamiento de los estados de los empleados, aceptar un sistema de Approval y desechar el estado de “REVOKE” del empleado para un estado de terminación que se podía activar y desactivar para cada Corporate Partner a elección.

Por lo que se construyeron recursos en AWS bastantes similares solamente para diferenciar un sistema de otro que no eran tan diferentes, pero al ser construidos en base a una serie de reglas de negocios que no estuvieron bien definidas y que ya era utilizado en ambientes productivos no quisieron afectar cualquier Corporate Partner que ya estaba activo en la plataforma.

Configuración de Corporate Partners

Entre los distintos flows, la base de la configuración de los Corporate Partners no se modificó. Cada Corporate Partner provee la información que enviará en el archivo, el formato y las validaciones necesarias. Veamos el siguiente ejemplo:

```
{
   "canonicalName": "oldflowtest",
   "id": "23dssad-49b2-11ea-8d67-41f492c824bb",
   "dynamicFlow": true,
   "name": "Oldflow Test",
   "fullFileFlow": true,
   "configuration": {
      "uniqueIdentifier": {
         "employeeId": {
            "@type": "string",
            "required": true,
            "max": 256
         }
      },
      "fileFields": {
         "firstName": {
            "@type": "string",
            "required": true,
            "max": 50
         },
         "lastName": {
            "@type": "string",
            "required": true,
            "max": 50
         },
         "jobTitle": {
            "@type": "string",
            "required": true,
            "max": 30
         },
         "dateOfBirth": {
            "@type": "date",
            "required": true
         },
```

```
            "employeeId": {
                "@type": "string",
                "required": true,
                "max": 256
            },
            "employmentStatus": {
                "@type": "string",
                "required": true,
                "max": 30
            },
            "email": {
                "@type": "email",
                "max": 256
            },
            "managerEmail": {
                "@type": "email",
                "required": true
            },
            "managerFirstName": {
                "@type": "string",
                "required": true
            },
            "location": {
                "@type": "string"
            },
            "managerLastName": {
                "@type": "string",
                "required": true
            }
        },
        "headerMapFields": {
            "firstName": "firstName",
            "lastName": "lastName",
            "managerEmail": "managerEmail",
            "jobTitle": "jobTitle",
            "managerFirstName": "managerFirstName",
            "dateOfBirth": "dateOfBirth",
            "employeeId": "employeeId",
            "location": "location",
            "employmentStatus": "employmentStatus",
            "managerLastName": "managerLastName",
            "email": "email",
            "eligible": "eligible"
        }
    }
}
```

Se tienen 3 atributos claves para el procesamiento del archivo:

- uniqueIdentifier: Este atributo es único para cada empleado y no puede repetirse. Los Corporate Partner necesitan una forma de identificar a sus empleados entre ellos sin utilizar atributos como el nombre o apellido ya que pueden repetirse. Por lo que se puede elegir un atributo del empleado como el "employeeId" o el email. En la base de datos, este atributo se guarda como un objeto ya que tiene sus propias validaciones a seguir. Debe ser requerido, tener un tipo ("string", "email", "number", etc) y tener el nombre del atributo de los fileFields a utilizar como Unique Identifier. También se le puede agregar validaciones extras, como un máximo de caracteres, números válidos o regex con los que validen.

- fileFields: Este objeto contiene todos los atributos que van a venir en un archivo de empleados. Con este objeto se genera el esquema de validación de cada línea, por eso cada atributo además de tener un nombre, va a tener un tipo, si es requerido o no y validaciones extras que puede llegar a tener (maximo de caracteres, validaciones con regex, etc). Si alguno de los atributos no cumple con sus validaciones, esa línea se marca como inválida.

- headerMapFields: Este Objeto guarda como vienen los atributos en las columnas del archivo y cómo se van a guardar en la base de datos. En la base de datos los atributos deben guardarse con el mismo nombre con el objetivo de tener la data normalizada, pero en los archivos los Corporate Partners pueden enviarnos los atributos con distintos nombres (Ex: firstName, First Name, Name, etc). Por lo que se mapea la llave del objeto (como viene en el archivo) con el valor (como se guarda en la base de datos y en los fileFields)

Mientras que el atributo "fullFileFlow: true" es el atributo que marca que va a utilizar el procesamiento de archivos viejo.

Control y tipo de errores:

Estos sistemas viejos no tenían un sistema ordenado de control de errores, solo se reconocen errores de nombres de archivos inválidos, columnas faltantes o si se supera el número límite de empleados a terminar. Pero los mails no daban la suficiente información para mostrar cuántos empleados llegaron a procesarse, si todo el procesamiento fue correcto o si fue parcialmente exitoso y cualquier otro tipo de información que pueda ayudar a obtener una respuesta de cuál fue el problema si el archivo fallaba por alguna causa no esperada.

Proyecto realizado:

Para evitar repetir los errores del sistema anterior, se planteó un sistema completamente nuevo que permite a cada Corporate Partner subir un archivo que se acomode a sus necesidades (que permita headers customizables para cada CP), manejando solamente 2 estados de Elegibilidad. El estado de elegibilidad se podrá obtener de dos maneras, una utilizando un campo del archivo donde el CP especifique si el empleado es Elegible o no y la segunda opción es computando la Elegibilidad.

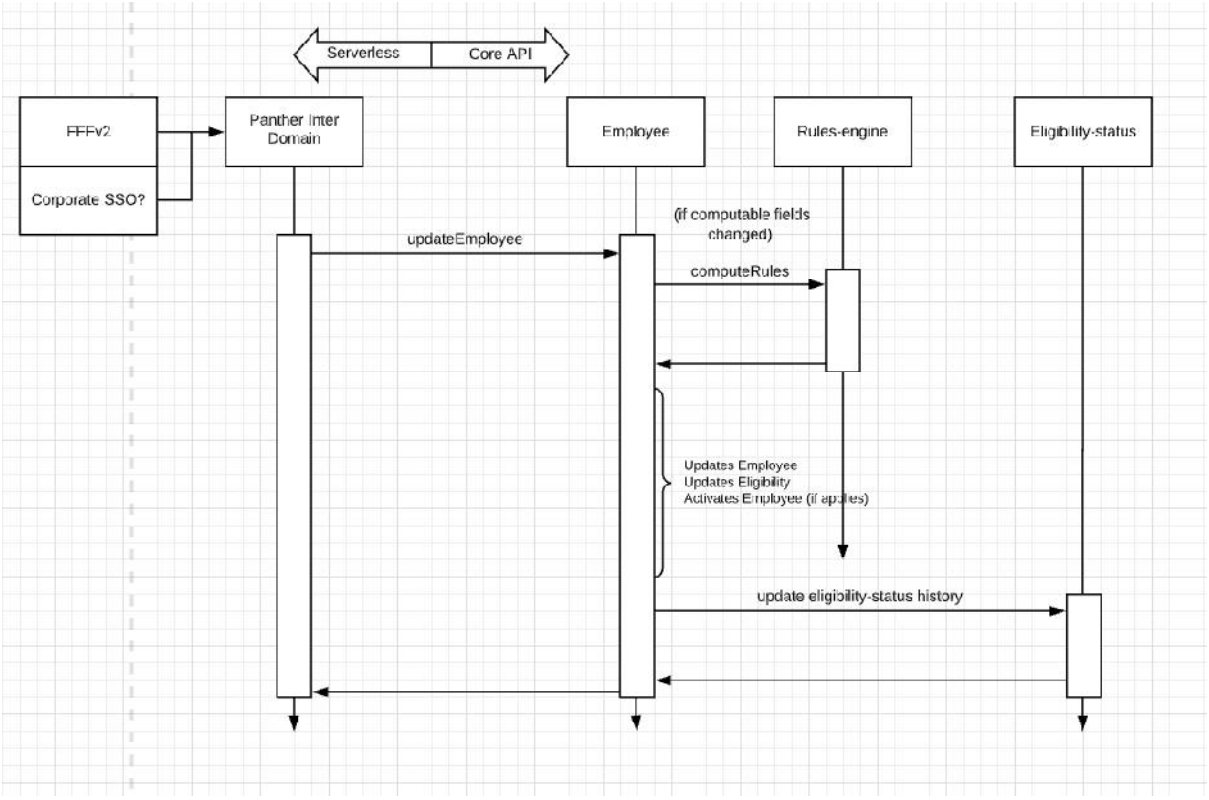
Se plantean reglas de Elegibilidad customizables para cada Corporate Partner donde cada uno podría elegir qué atributos del empleado se utilizarán para computar la elegibilidad. Por ejemplo, si se requiere que un empleado esté hace más de 6 meses en la empresa para ser elegible, se utilizará el campo “hireDate” para calcularla.

Si se requiere que esté dentro de una división en particular, se utilizará el campo “división". Y así sucesivamente. Con esto, se le da la oportunidad a los Corporate Partners que dejen de calcular ellos cuáles de sus empleados son elegibles y cuáles no y el sistema puede encargarse de ello.

Se plantearon 2 stacks especiales en AWS. Uno dedicado específicamente para el procesamiento de archivos, donde existe la Step Function y sus distintas Lambdas y otro substack que se encarga de una capa intermedia donde se realizaran la creación y update de empleados, actualización de registros de approvals y cualquier otra entidad que se requiera en el futuro.

El motivo por el que se decidió tener 2 stacks separados para eso, fue para no atar la creación o update de un empleado con el procesamiento del archivo. Se espera que el sistema crezca en el futuro y que un empleado se pueda crear o actualizar a través de medios diferentes al procesamiento de archivos. Por eso las lambdas de creación y update deberían estar aisladas y solo crear o updatear empleados en base a los registros recibidos.

A continuación, se puede ver el diagrama de secuencia del nuevo proyecto, donde se diferencia claramente la división por capas:



Luego de que se realice el procesamiento del archivo, la capa de Inter-Domain recibe la información necesaria para poder realizar la creación o actualización de un empleado. Nótese que se menciona una forma diferente de creación de empleados llamada Corporate SSO. Aunque no exista esa funcionalidad actualmente, la capa de Inter-Domain se desarrolló con la idea de múltiples puntos de entrada para la creación de empleados y que pudiera funcionar sin importar de donde provenga la información.

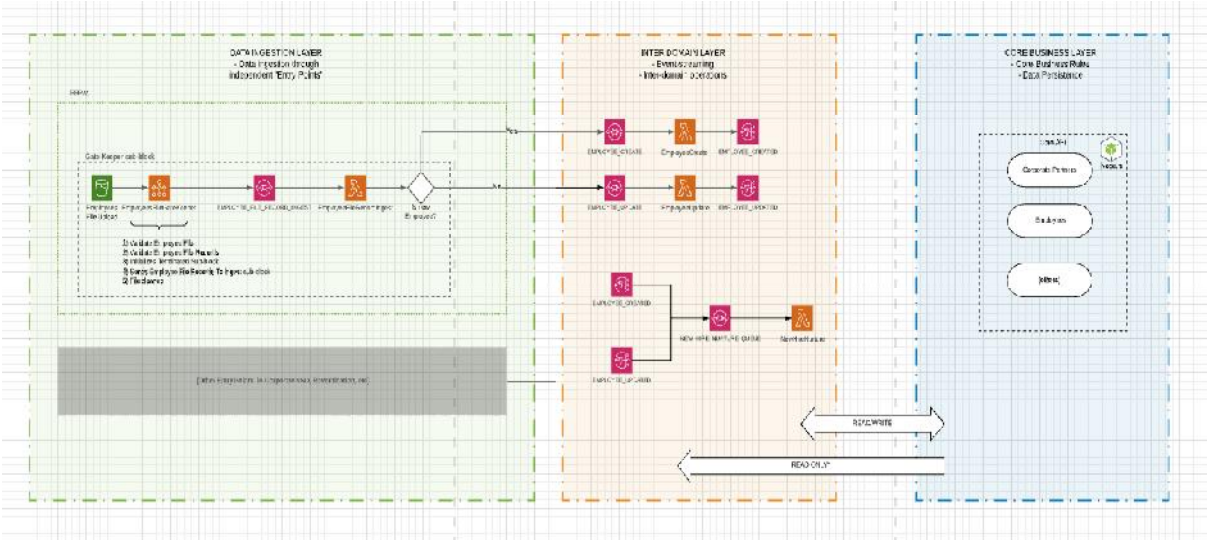
Distintos puntos de creación de empleados pueden tener distintas validaciones que no sean compatibles entre ellos. Por lo que estas validaciones están fuera de la capa Inter-Domain, la capa espera que la información recibida ya haya sido validada y que esté lista para realizar la creación o actualización.

La capa de Inter-Domain llama a los endpoints de la Core API que crean o actualizan a los empleados y dispara los eventos avisando de la creación o actualización del mismo. En los endpoints de creación/actualización, se dividen las responsabilidades entre distintos servicios.

El servicio de empleados se encarga de crear/actualizar el mismo, el servicio de Rules-Engine se encarga de computar la elegibilidad del empleado si sus atributos cambiaron y el servicio de Eligibility-Status se encarga de actualizar el historial de elegibilidad del empleado en el caso que sea necesario.

También se cambió la base de datos utilizada. DynamoDB se reemplazó por MongoDB, una base de datos no relacional. DynamoDB resultó ser muy cara y consumía demasiado tiempo y recursos a comparación de MongoDB. Y como la estructura de varias entidades todavía no estaban definidas a la hora de plantear el nuevo sistema, se decidió que una base de datos no relacional era la mejor solución.

Diagrama completo de eventos:



Definiciones y recursos:

Recursos de sub stack de procesamiento de archivo:

Step Function:

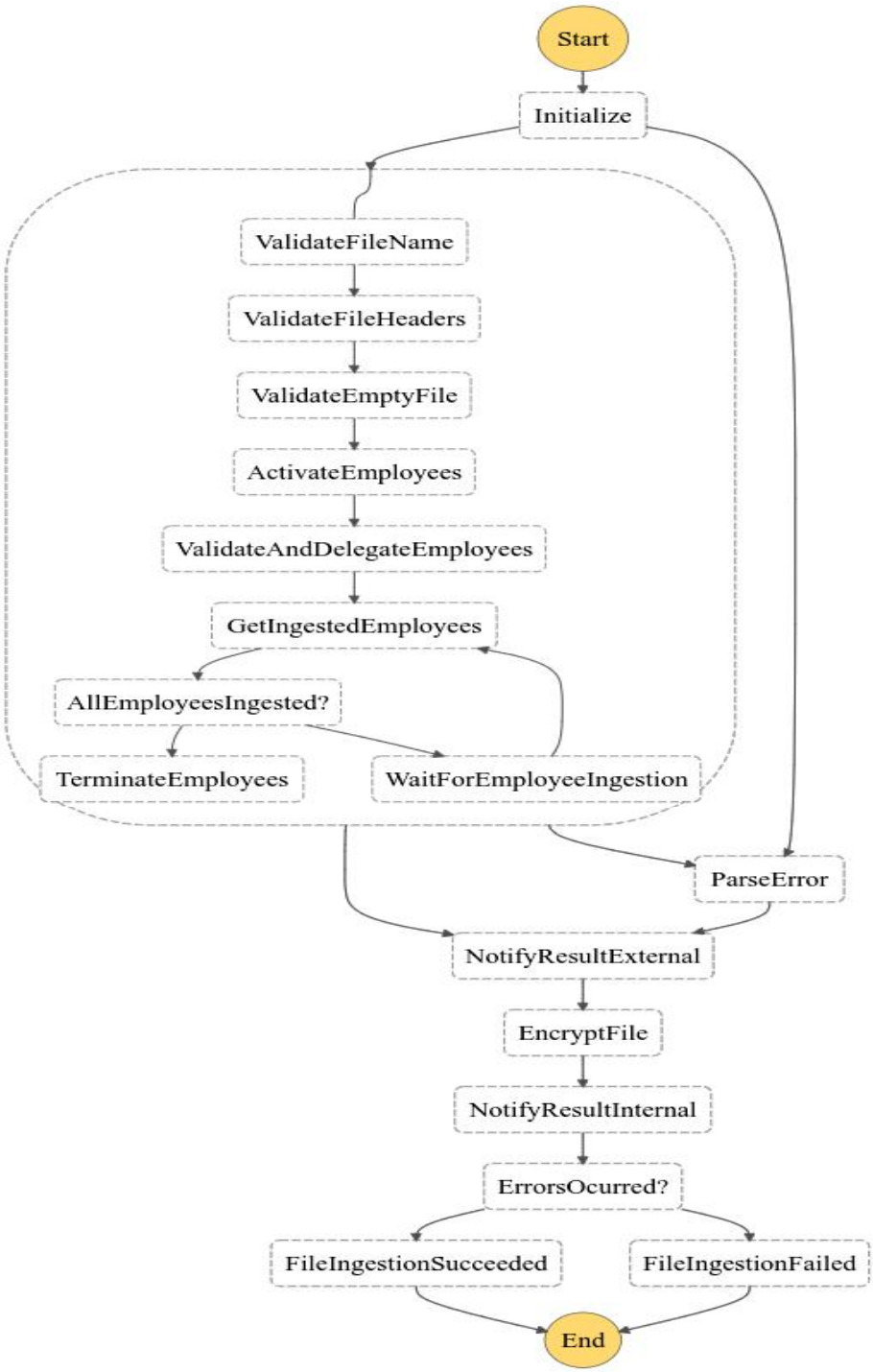
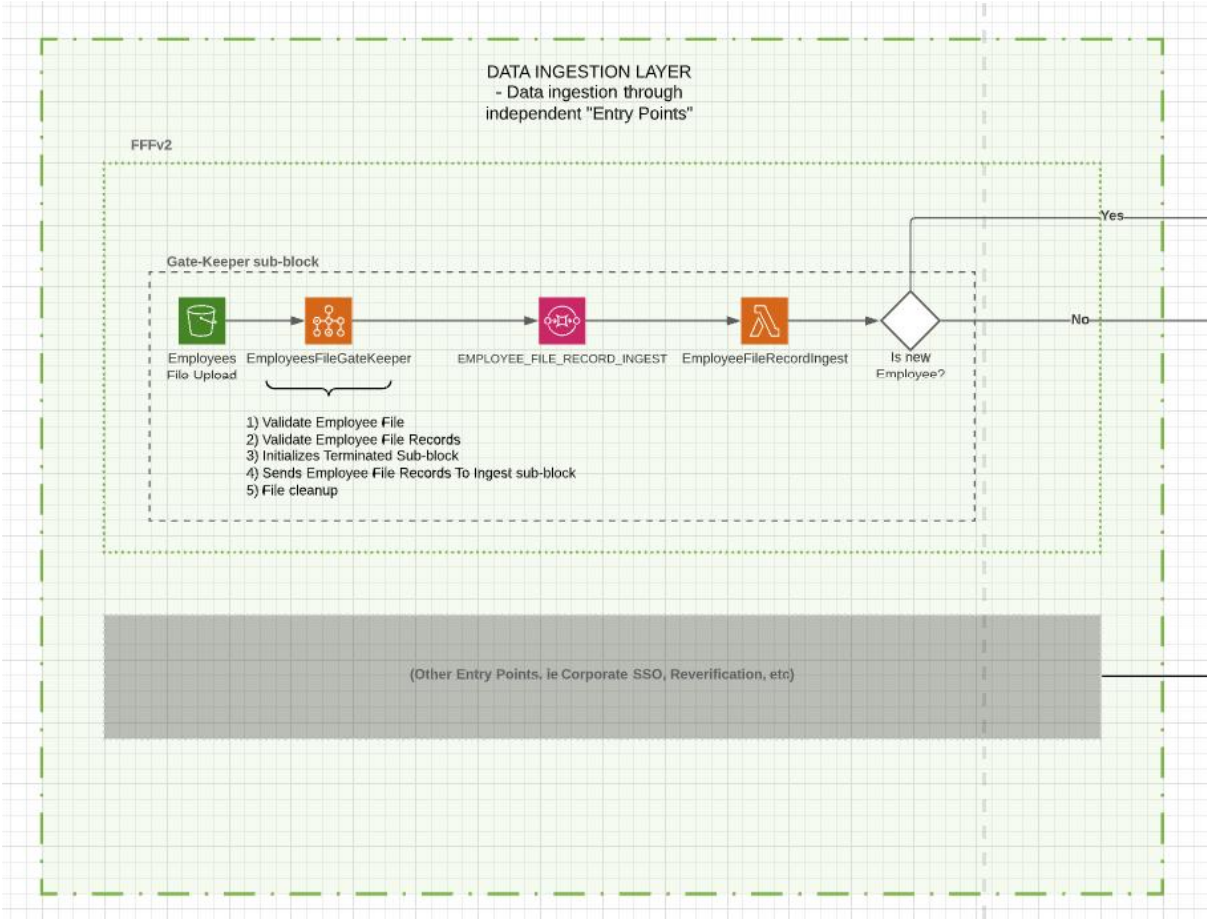


Diagrama de eventos de la capa de procesamiento de archivos:



Al subirse un archivo en el bucket de S3, se invoca la Step Function de procesamiento de archivo. Esa Step Function va a disparar 2 tipos de eventos, CREATE_EMPLOYEE y UPDATE_EMPLOYEE que son escuchados por la capa Inter-Domain (o cualquier otra Lamba o capa que lo necesite).

Lambdas utilizadas y conexiones:

1. **Initialize**: Esta Lambda inicializa el procesamiento buscando la configuración del Corporate Partner, cargando sus reglas de elegibilidad y validaciones de archivos.

2. **ValidateFileName**: Está lambda válida el nombre del file. Cumple la misma función que la Lambda utilizada en las Step Function anteriores, pero no se utilizó esa Lambda ya que no era compatible sus parámetros de entrada y salida con los de esta Step Function.

3. **ValidateEmptyFile**: Esta Lambda valida que el archivo no esté vacío.

4. **ActivateEmployees**: Esta Lambda marca como “activos” a los empleados que vinieron en el archivo. El concepto de revocado ya no existe en este nuevo flujo para mantener solo 2 estados de Elegibilidad. Por lo que además se ser ELEGIBLE o NOT_ELIGIBLE el empleado puede estar marcado como terminado o activo.

5. **ValidateAndDelegateEmployees**: Esta Lambda busca si el empleado existe en la base de datos o no. Si no existe, dispara un evento de SNS llamado CREATE_EMPLOYEE o UPDATE_EMPLOYEE en el caso de que el empleado exista enviando la información del archivo como parte del evento.

6. **GetIngestedEmployees:** Esta Lambda busca en la base de datos si todos los empleados que vinieron en el archivo fueron creados o actualizados. Si entre un llamado y otro no aumentó el número de empleados cargados, se suma en 1 un contador. Cuando este contador llega a 4 es porque hubo un problema en las Lambdas de creación o update y se dispara un error de TIME OUT.

7. **TerminateEmployees:** Esta Lambda termina todos los empleados que no vinieron en el archivo.

8. **ParseError:** Si alguna de los steps anteriores lanza un error, esta Lambda lo agarra y parsea el error del sistema a un error que pueda ser interpretado por un administrador del sistema.

9. **NotifyResultExternal:** Esta Lambda envía un mensaje a Iterable (un sistema externo de envíos de notificaciones) para que envíe un email a los administradores de los Corporate Partners para que sepan del resultado del procesamiento del archivo.

10. **EncryptFile:** Esta Lambda encripta el archivo, al igual que las Step Functions anteriores.

11. **NotifyResultInternal:** Esta Lambda dispara un evento SNS que envía notificaciones a los administradores del cliente, con el resultado del procesamiento, cantidad de empleados procesados, tiempo de procesamiento y el listado de errores que pasaron.

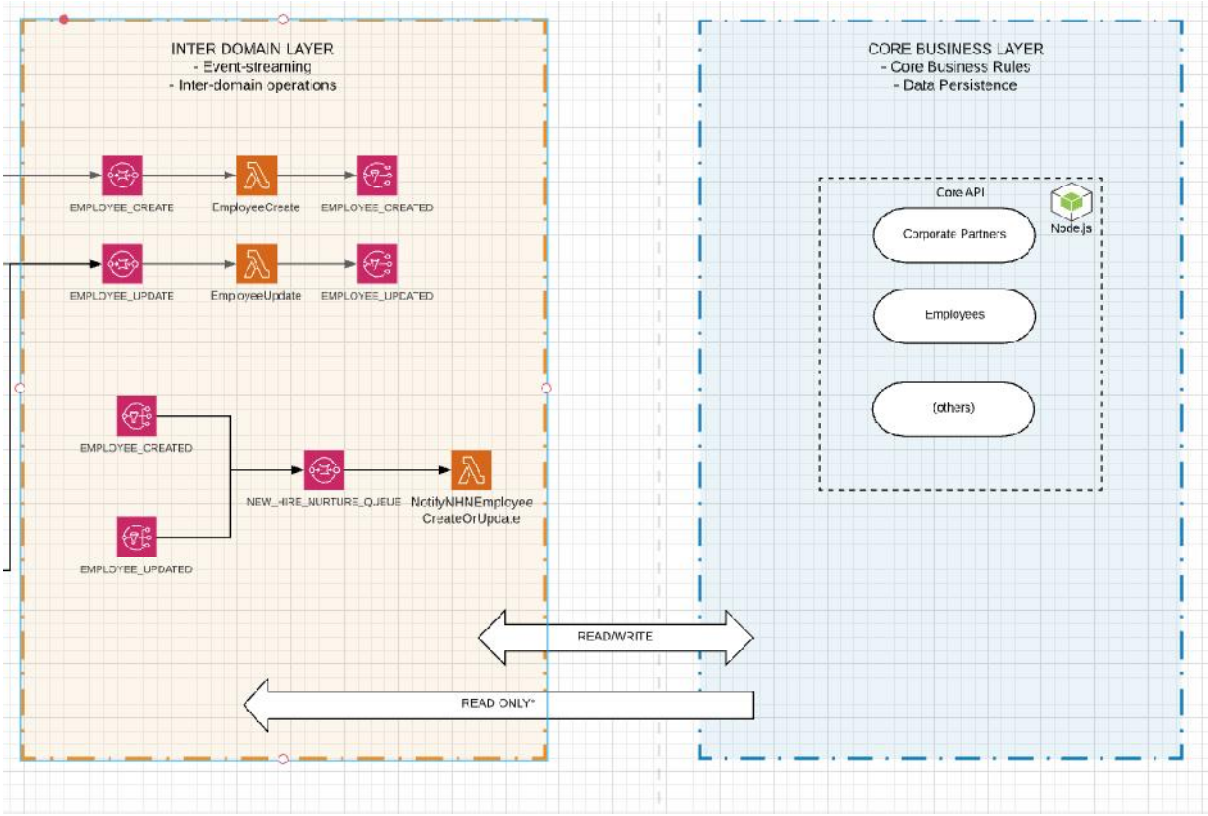
Esta Step Function encapsula los primeros 7 steps, por lo que, si uno de esos steps falla, se puede captar ese error y notificarlo de manera correcta a los administradores y encriptar el archivo correctamente.

Recursos de Sub Stack de inter-domain

Este Sub Stack se ocupa de la creación y actualización de empleados, registros de aprobación y del envío de notificaciones de Elegibilidad para los empleados. A diferencia del Sub Stack de File Processing, este stack no comunica las Lambdas utilizando Step Functions. La comunicación y disparo de las Lambdas ocurren en base a Eventos de SNS disparados por Lambdas externas.

Para cada acción se dispara un evento con el payload y nombre correspondiente. Existe una queue en SQS para cada evento, escuchando por un evento en particular cada SQS. La llegada de un evento a una SQS hace que se dispare una Lambda que es la que se va a ocupar de ejecutar la acción necesaria.

A continuación, se puede ver el diagrama de eventos de la capa de Inter-Domain:



Un ejemplo de caso de uso es el siguiente:

- La capa de procesamiento de archivos envía un evento de SNS de CREATE_EMPLOYEE.
- Existe en el inter-domain stack una SQS (EmployeeCreateQueue) que escucha por los eventos de CREATE_EMPLOYEE
- Cuando se agrega un elemento a esta SQS, automáticamente una Lambda es disparada por ese evento (agregar un elemento a la SQS EmployeeCreateQueue) y se ocupa de crear el registro de empleado en la Base de Datos con su Elegibilidad correspondiente. Para eso llama al endpoint de POST /employee de la Core API.
- Una vez que se crea el registro y se recibe la respuesta del endpoint de la Core API, la Lambda dispara un evento de EMPLOYEE_CREATED y termina su ejecución.
- Este evento de EMPLOYEE_CREATED es escuchado por la Step Function cuando se pone a contar cuántos empleados fueron procesados, pero también es escuchado por múltiples Lambdas que esperan que un empleado se cree para poder realizar una acción en particular.

Este proceso y forma de relacionar Lambdas y eventos hace que se puedan procesar múltiples registros de empleados en paralelo. Ya que si llegan X cantidad de eventos de CREATE_EMPLOYEE, se agregarán X empleados en la SQS y eso generará X instancias de la Lambda de creación de empleados que se ejecutarán en paralelo.

Recursos creados para cada acción:

Creación de empleados:

- EmployeeCreateFunction: Lambda que se ocupa de la creación de un registro de empleado y la notificación de la creación del mismo.
- EmployeeCreateFunctionQueueEvent: Evento que mapea la Lambda de creación de empleados con la SQS de creación de empleados.
- EmployeeCreateQueue: SQS donde se agregan los payloads de empleados a crear.
- EmployeeCreateFunctionRole: Rol que permite conectar la SQS de creación de empleados con los eventos de CREATE_EMPLOYEE.
- EmployeeCreatedTopic: Tópico que se dispara luego de que se cree correctamente un empleado. Este topic es escuchado por otras Lambdas a la espera de que se complete la creación de un empleado.

Actualización de empleados:

- EmployeeUpdateFunction: Lambda que se ocupa de la actualización de un registro de empleado y la notificación de la creación del mismo.
- EmployeeUpdateFunctionQueueEvent: Evento que mapea la Lambda de actualización de empleados con la SQS de creación de empleados.
- EmployeeUpdateQueue: SQS donde se agregan los payloads de empleados a actualizar.
- EmployeeUpdateFunctionRole: Rol que permite conectar la SQS de actualización de empleados con los eventos de UPDATE_EMPLOYEE.
- EmployeeUpdatedTopic: Tópico que se dispara luego de que se actualice correctamente un empleado. Este topic es escuchado por otras Lambdas a la espera de que se complete la actualización de un empleado.

Creación de registros de APPROVAL:

- SendApprovalRequestFunction: Lambda que se ocupa de enviar la petición de un empleado a su manager y la creación del registro de “Approval”.
- SendApprovalRequestFunctionQueueEvent: Evento que relaciona la Lambda de envío de peticiones de aprobación con la SQS de envío de peticiones.
- SendApprovalRequestFunctionRole: Rol que permite conectar la SQS de envío de peticiones de aprobación con los eventos de APPROVAL_REQUEST_RECEIVED.
- SendApprovalRequestPolicy: Atributo que permite a un tópico realizar ciertas acciones o tener permisos para la comunicación entre eventos de SNS y colas de SQS. Se utiliza para dar los permisos necesarios para la suscripción entre SendApprovalRequestFunctionQueue y ApprovalRequestSentTopic.
- SendApprovalRequestQueue: Cola de SQS donde se guardarán los eventos de peticiones de aprobación.
- SendApprovalRequestSubscription: Suscripción al evento de SNS de petición de aprobación para que la cola de SQS pueda recibir dichos eventos.
- ApprovalRequestSentTopic: Tópico de SNS para los eventos de petición de aprobación de un empleado.

Actualización de registros de APPROVAL:

- AppprovalAnswerReceivedSubscription: Suscripción de SNS para que queues de SQS se puedan suscribir al evento de respuesta del pedido de aprobación.
- ApprovalAnswerReceivedFunction: Lambda que se ocupa de recibir la respuesta de un manager y actualizar el registro de “Approval” del empleado.
- ApprovalAnswerReceivedFunctionQueueEvent: Evento que relaciona la Lambda de actualización de peticiones de aprobación con la SQS de respuestas de los managers de los empleados.
- ApprovalAnswerReceivedFunctionRole: Rol que permite conectar la SQS de recibimiento de respuestas de manager con respecto a las peticiones de aprobación con los eventos de APPROVAL_ANSWER_RECEIVED.
- ApprovalAnswerReceivedPolicy: Atributo que permite a un tópico realizar ciertas acciones o tener permisos para la comunicación entre eventos de SNS y colas de SQS. Se utiliza para dar los permisos necesarios para la suscripción entre ApprovalAnswerReceivedFunctionQueue y ApprovalAnswerReceivedTopic.
- ApprovalAnswerReceivedQueue: Cola de SQS donde se guardarán los eventos de respuestas de manager sobre las peticiones de aprobación.
- ApprovalAnswerReceivedTopic: Tópico de SNS para los eventos de recepción de respuesta de los manager sobre las peticiones de los empleados.

Configuración de Corporate Partners:

En el nuevo flujo de procesamiento de archivos, la configuración de Corporate Partners se mantiene igual que en el flujo anterior solamente con dos nuevos atributos, “eligibilityRules” y “fullFileFlow2: true”. El primer atributo es el que se utiliza para calcular la elegibilidad de un empleado. La elegibilidad se va a calcular en base a los atributos de un empleado. Mientras que el segundo atributo marca que el Corporate Partner va a estar utilizando el nuevo sistema de procesamiento de archivos.

En el caso de que el Corporate Partner no quiera computar la elegibilidad de un empleado, se tiene la siguiente regla de elegibilidad por defecto. Esta regla calcula la elegibilidad del empleado en base al atributo “eligible” del empleado que vino en el archivo.

```
    "eligibilityRules": [{
        "name": "Eligible",
        "priority": 100,
        "conditions": {
            "all": [{
                "fact": "eligible",
                "operator": "equal",
                "params": {
                    "message": ""
                },
                "value": "y"
            }]
        },
```

```
        "event": {
            "params": {
                "value": "ELIGIBLE"
            },
            "type": "eligible"
        }
    }]
```

Pero si el Corporate Partner decide utilizar reglas de elegibilidad, se pueden agregar de la siguiente forma:

```
    "eligibilityRules": [{
        "name": "Eligibility",
        "conditions": {
            "all": [{
                "fact": "hireDate",
                "params": {
                    "message": "must have 3 month of service"
                },
                "value": "3m",
                "operator": "moreThanInTime"
            }]
        },
        "event": {
            "params": {
                "value": "ELIGIBLE"
            },
            "type": "eligible"
        },
        "priority": 1
    }]
```

Esta regla especifica que solamente los empleados cuya “hireDate” sea hace más de 3 meses. Cualquier empleado que haya sido contratado en el periodo de los últimos 3 meses no será Elegible. Pero puede utilizarse cualquier atributo del empleado para realizar estos cálculos.

Las reglas de elegibilidad tienen 4 atributos clave:

- Name: Nombre de la regla de elegibilidad. Debe ser único, no pueden existir 2 o más reglas con el mismo nombre.
- Conditions: Condiciones que debe cumplir la regla de elegibilidad para cumplirse. Las condiciones pueden ser varias y tienen varios atributos:
    - “all”: Se deben cumplir todas las condiciones dentro del arreglo de condiciones.
    - “or”: Solo alguna de las condiciones debe cumplirse para que se cumpla la regla.

    - Fact: Atributo del empleado que debe utilizarse para el cálculo.
    - Params: Objeto que contiene el mensaje de la regla a cumplir.
    - Value: Valor que debe tener el atributo para que se cumpla la regla.
    - Operator: Operación que se debe hacer contra el atributo y el valor para ver si se cumple la regla.
- Event: Evento que se emite si se cumple la regla. En el caso de los Corporate Partners el evento de “ELIGIBLE” dicta la elegibilidad del empleado.
- Priority: En el caso de que el Corporate Partner quiera tener múltiples reglas de elegibilidad, las reglas tienen prioridades para los casos donde dos o más se cumplan.

Las operaciones que se pueden realizar utilizando el Rules Engine son varias, estas son las utilizadas por los Corporate Partners:

- equal *fact* must equal *value*
- notEqual *fact* must not equal *value*
- lessThan *fact* must be less than *value*
- lessThanInclusive *fact* must be less than or equal to *value*
- greaterThan *fact* must be greater than *value*
- greaterThanInclusive *fact* must be greater than or equal to *value*
- in *fact* must be included in *value* (an array)
- notIn *fact* must not be included in *value* (an array)
- contains *fact* (an array) must include *value*
- doesNotContain *fact* (an array) must not include *value*
- lessThanInclusiveInTime *fact* (a Date) must be less than or equal to *value* (a time exp*)
- lessThanInTime *fact* (a Date) must be less than *value* (a time exp*)
- moreThanInclusiveInTime *fact* (a timestamp) must be more than or equal to *value* (a time exp*)
- moreThanInTime *fact* (a Date) must be more than *value* (a time exp*)

Time exp: Es un string que representa una unidad y una cantidad de tiempo. Por ejemplo, 3m representa 3 meses, 5y representa 5 años.

Control y tipo de errores:

El encapsulamiento de los steps iniciales hacen que la Step Function considere todos esos Steps como un steps singular.

Estos 7 pasos se decidieron encapsular ya que se los considera el Step de lectura y validación del archivo. Este Step no realiza la carga o actualización de los mismos, simplemente manda eventos para delegar la creación o actualización a distintas Lambdas. Por lo que un fallo en el procesamiento del archivo no causa que el resto de los empleados que se debían procesar fallen.

Además, se dividió y crearon distintos errores por cada paso de la validación y procesamiento del archivo:

1. **EMPTY_FILE_ERROR:** Si el archivo viene vacío, se lanza este error para evitar que se llamen al resto de los Steps cuando no es necesario.
2. **INVALID_ROWS_ERROR:** Este error no causa que falle completamente el procesamiento del archivo. Se pueden tener líneas inválidas pero que no limiten el procesamiento del resto del archivo. En este error se guardan las líneas inválidas para poder notificar cuales fueron.
3. **TERMINATION_THRESHOLD_EXCEEDED_ERROR:** Este error evita que se realice el proceso de terminar empleados. Si por algún motivo se sube un archivo de empleados con más de un porcentaje X (definido por el Corporate Partner) faltantes del archivo, no se termina a ninguno. Se realizó esto para evitar que sin querer se terminen múltiples empleados de un Corporate Partner al subir un archivo erróneo con pocos empleados.
4. **INVALID_FILE_NAME_ERROR:** Si el archivo no tiene un nombre correcto, se lanza este error para evitar que se llamen al resto de los Steps cuando no es necesario.
5. **INVALID_FILE_HEADERS_ERROR:** Si el archivo no contiene todos los headers requeridos, se lanza este error para evitar que se llamen al resto de los Steps cuando no es necesario. Si uno de los headers requeridos no se encuentra en el archivo, todas las líneas van a estar mal y se intentará procesar todo un archivo innecesariamente.
6. **DUPLICATED_UID_ERROR:** Este error existe de forma similar al **INVALID_ROWS_ERROR,** se puede seguir procesando el archivo, pero si se encuentra que dos líneas tienen el mismo identificador para empleados diferentes no se procesan y se notifica de las líneas inválidas.
7. **SYSTEM_ERROR:** Este error existe como error por defecto. Si el error que ocurre no es uno de los errores esperados, se guarda el error y se notifica que hubo un error en el sistema.

Esos son los errores que se consideran errores de procesamiento de archivos, ya que solo pueden ocurrir durante el procesamiento del mismo. Fuera del procesamiento del archivo pueden ocurrir los siguientes errores:

8. **ENCRYPTION_ERROR:** Este error se lanza cuando falla la encriptación del archivo.
9. **EXTERNAL_NOTIFICATION_ERROR:** Este error se lanza cuando falla el envío de la notificación a los Corporate Partners. Si el envío del resultado del procesamiento del archivo a los Corporate Partners, se debe notificar de manera manual utilizando este error para saber si es necesario realizarlo o no.
10. **INTERNAL_NOTIFICATION_ERROR:** Este error se lanza cuando falla la notificación interna a los administradores del cliente. La notificación interna es la que notifica el resultado con todas las características del procesamiento. Si esta notificación falla y no se envía, los administradores deberán acceder a la ejecución de la Step Function para ver los logs y encontrar cuál fue el problema y el resultado.
11. **INGESTION_LOOP_TIMEOUT:** Este error surge en base a las respuestas de las Lambdas de creación y de actualización de los empleados. Si luego de 10 minutos no se recibe ningún evento de EMPLOYEE_CREATED o EMPLOYEE_UPDATED y todavía no se llegó a la misma cantidad que la de columnas válidas en el archivo se lanza este error para evitar que la Step Function quede esperando a eventos que nunca lleguen (pueden no llegar por cuello de botella o eventos que se pierden si hay mucho tráfico en el sistema).
12. **UNKNOWN_ERROR:** Si alguno de los Steps anteriores que no es parte de los pasos de procesamiento de archivos fallan, se lanza este error para que se dispare una alarma que los administradores podrán ver y tendrán que entrar a la ejecución de la Step Function para ver cuál es el problema.

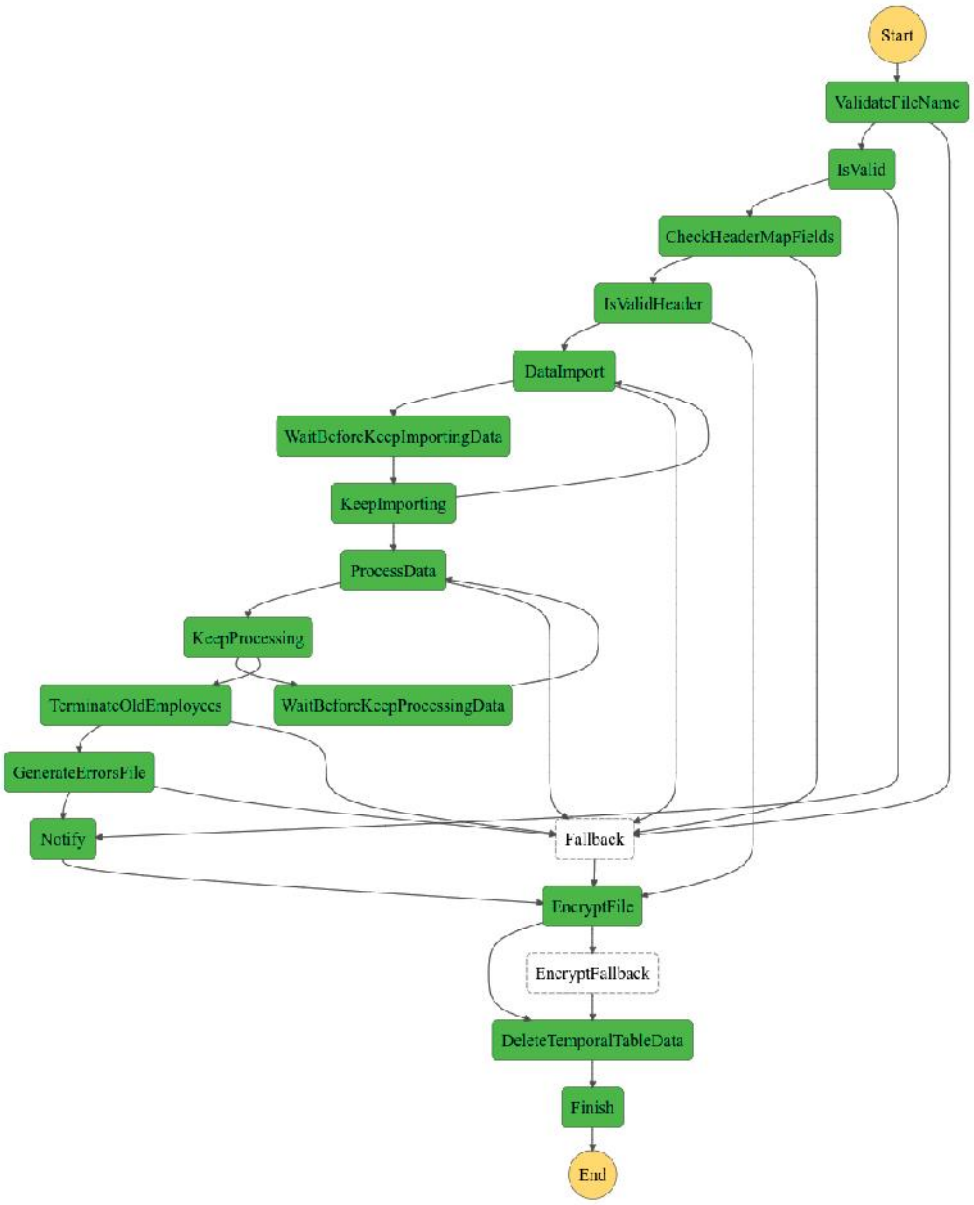
Comparación de rendimiento entre sistemas

Aclaración:

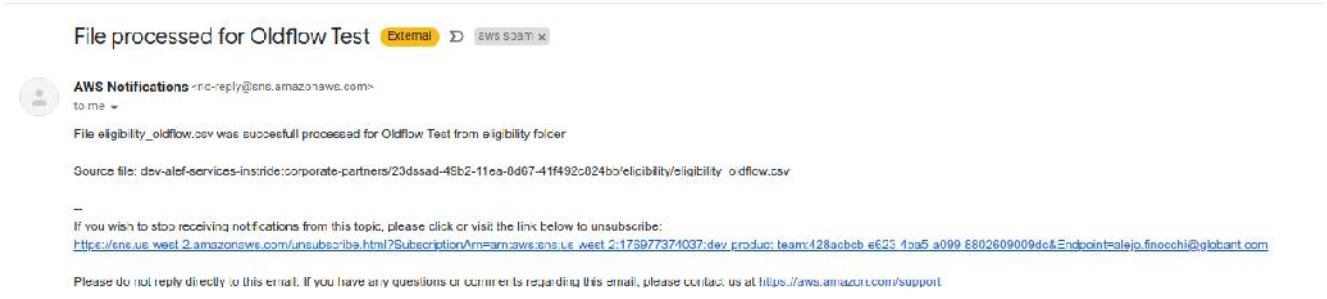
Para comparar los dos sistemas se va a utilizar el mismo archivo, la diferencia es que ese archivo se va a subir a una carpeta de un Corporate Partner en el sistema viejo y luego se va a subir en otro Corporate Partner configurado para que utilice el sistema nuevo

Sistema Viejo:

Se subió un archivo con 9964 empleados, todos válidos y que no existían en el sistema:



Cuando el procesamiento se termina, se recibe el siguiente email para los administradores del cliente como mensaje de que se procesó el archivo cargado. Pero no provee información de la cantidad de empleados, tiempo de ejecución o problemas que pudieron haber ocurrido:



File processed for Oldflow Test External aws sdam

AWS Notifications <no-reply@sns.amazonaws.com>
to me

File eligibility_oldflow.csv was succesfull processed for Oldflow Test from eligibility folder

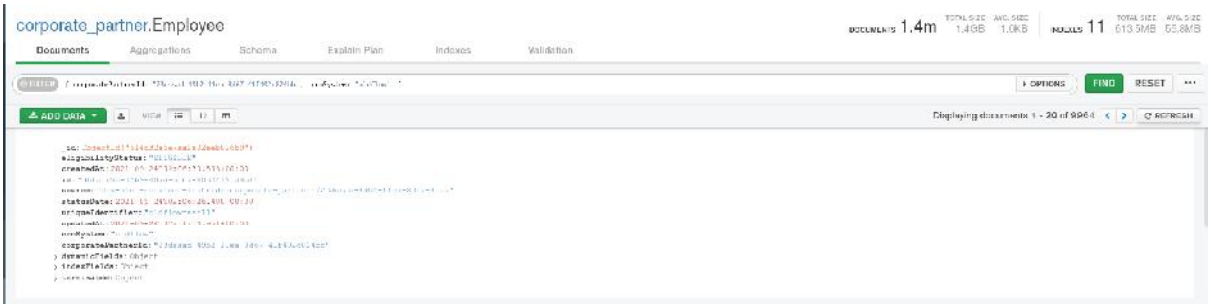
Source file: dev-alef-services-insride:corporate-partners/23dssad-45b2-11ea-0d67-41f492c024bb/eligibility/eligibility_oldflow.csv

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-west-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-west-2:178977374037:dev_product_team:428acbcb-e623-4ba5-a099-8802609009dc&Endpoint=alejo.finocchi@globant.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

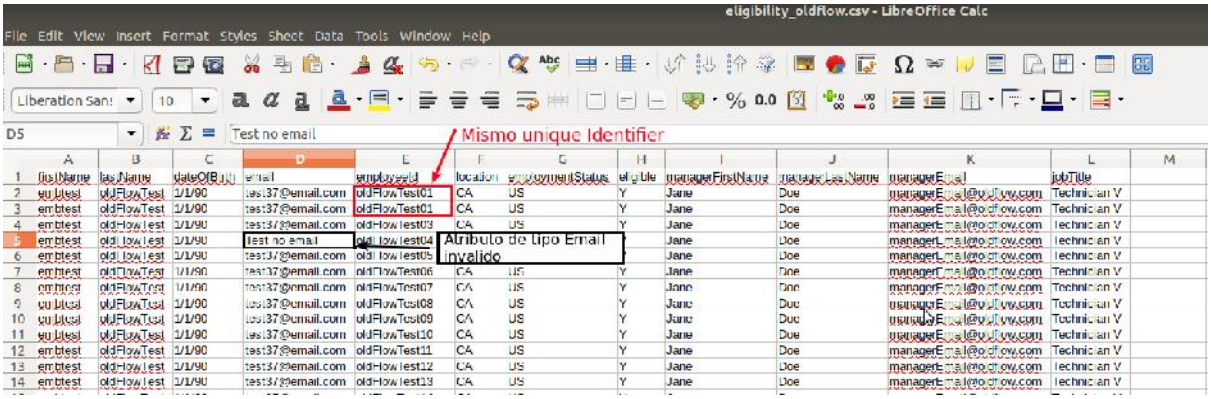
Para poder ver el tiempo de ejecución de la Step Function se debe ingresar a los logs de la ejecución y calcular el tiempo de ejecución. Viendo los logs se puede ver que la ejecución duró 29 minutos.

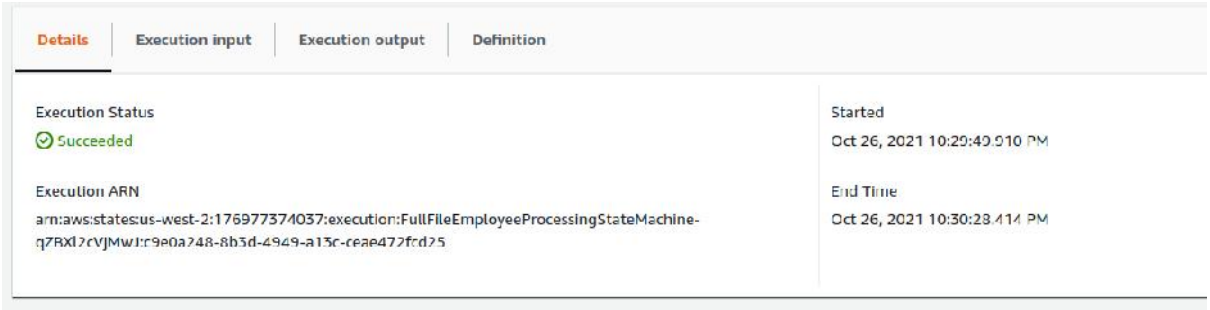
Y para ver la cantidad de empleados procesados hay que buscar en la base de datos a los empleados de ese Corporate Partner que pertenecían a ese archivo:



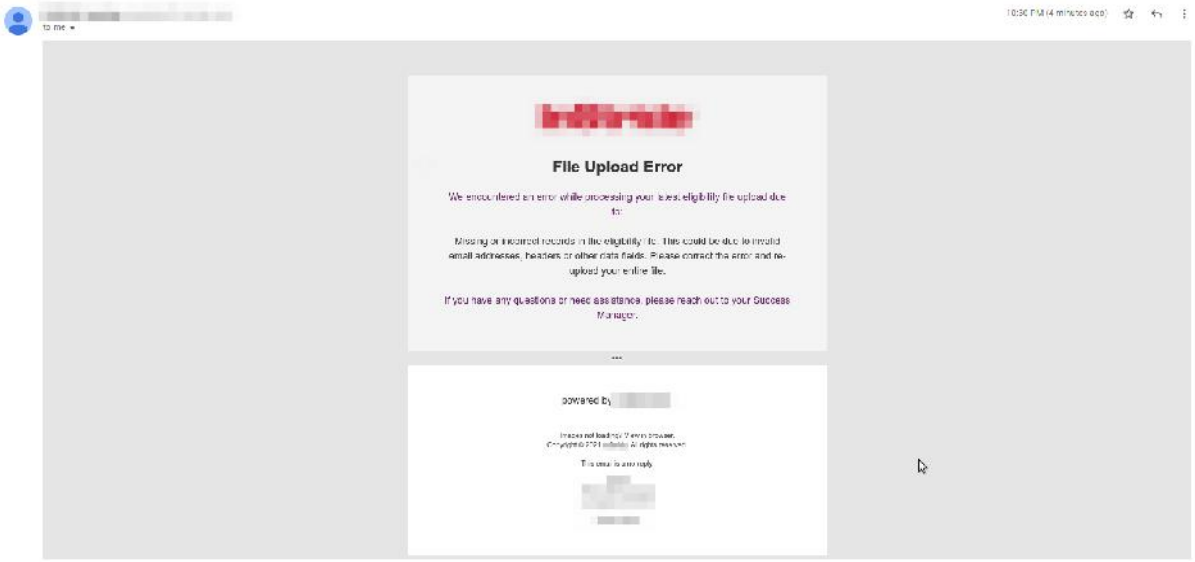
Notificaciones de errores:

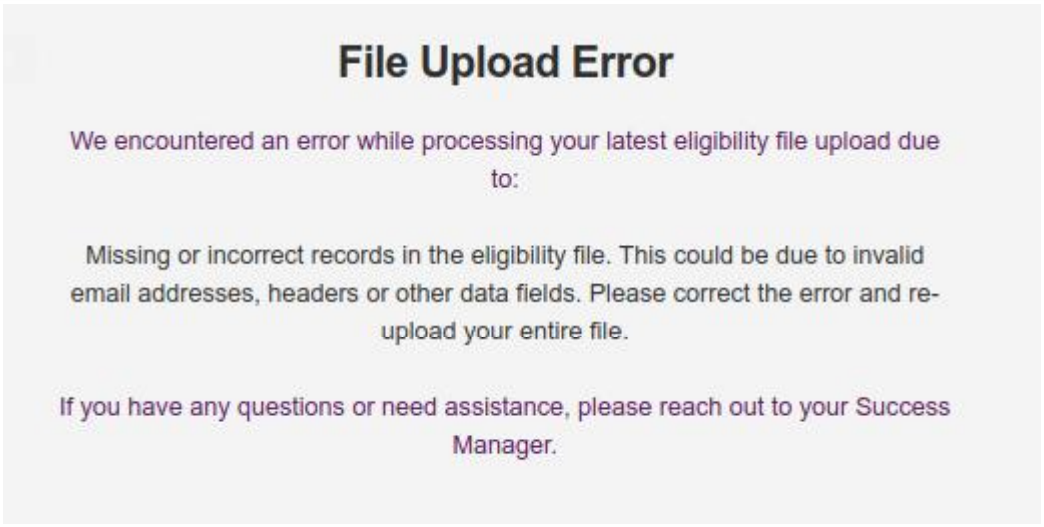
Subiendo un archivo con líneas inválidas (atributos de tipo “email” sin formato válido y empleados con Unique Identifiers repetidos) estos fueron los resultados:

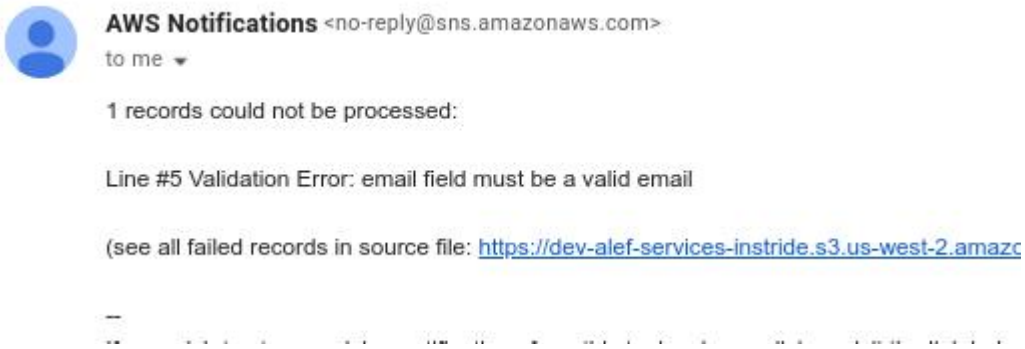




El sistema terminó con un mensaje de Success como si no hubiera ocurrido ningún error, pero estas fueron las notificaciones de Emails recibidas:

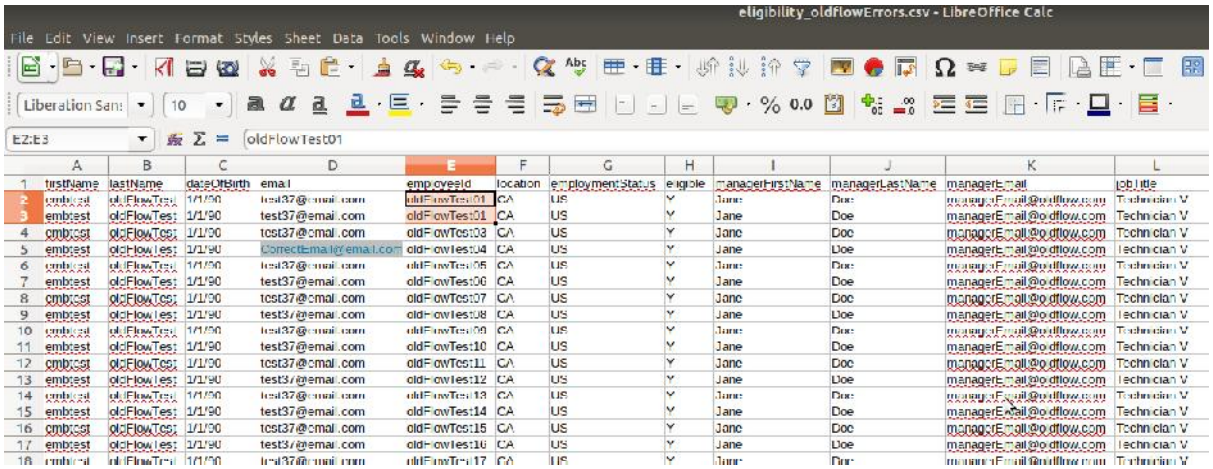






El email que se envía a los Corporate Partners contiene información sobre la causa principal del error y el mail interno solo menciona uno de los errores. No menciona cuántos empleados llegaron a procesarse, cuanto tardo y la localización del archivo encriptado en el caso que sea necesario recuperarlo. Solo la carpeta en la que se subió.

Si se corrige la línea invalida del email y se vuelve a subir el archivo, esta es la notificación recibida:

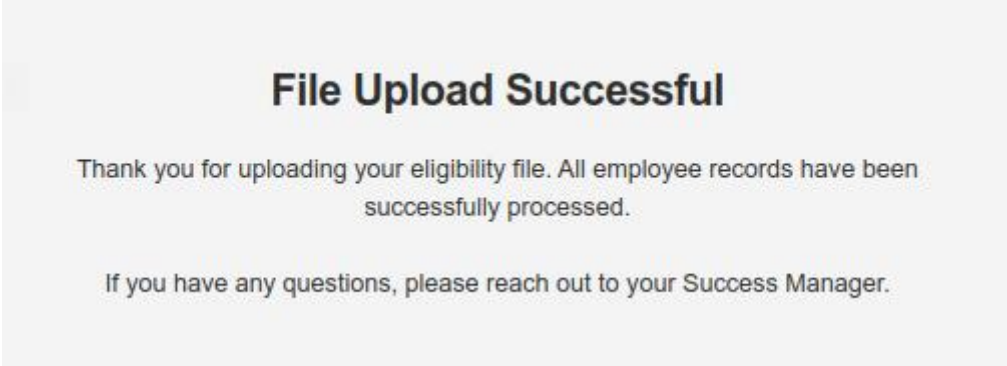


File processed for Oldflow Test External aws spam ×

AWS Notifications <no-reply@sns.amazonaws.com>
to me

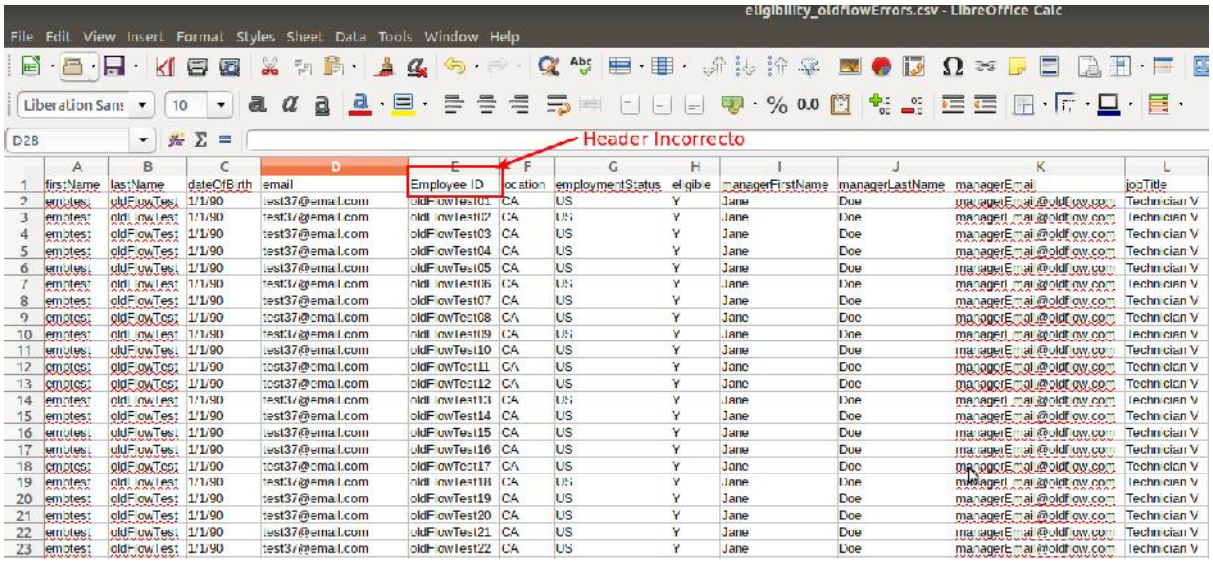
File eligibility_oldflowErrors.csv was succesfull processed for Oldflow Test from eligibility folder

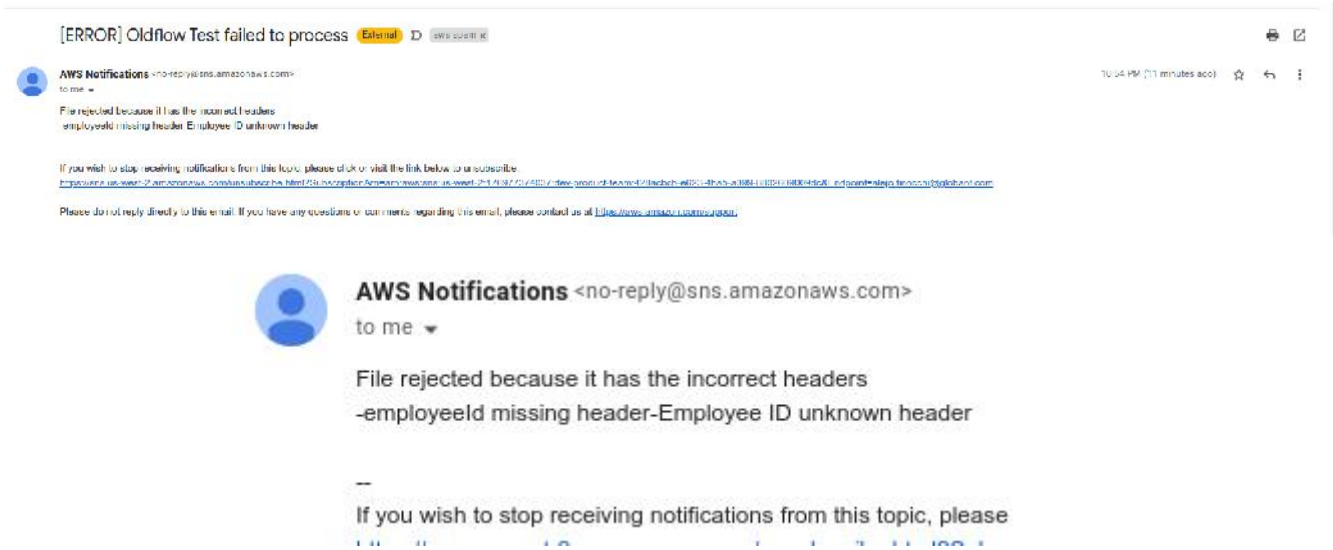
Source file: dev-alef-services-instride:corporate-partners/23dssad-49b2-11ea-8d67-41f492c824bb/eligibility/eligibility_oldflowErrors.csv



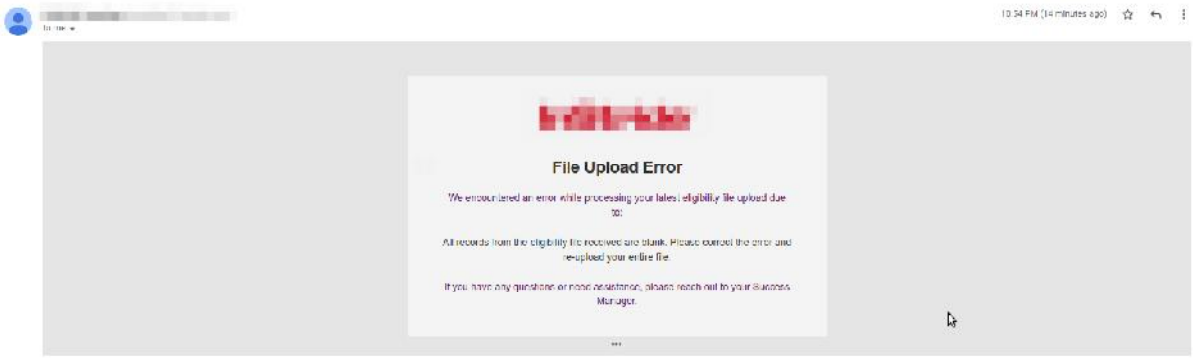
Toma las líneas con el mismo Unique Identifier como válidas, aunque eso genere que uno de los dos empleados no se cree o actualice en el sistema. Por lo que existía la posibilidad de notificar a los administradores de un procesamiento exitoso cuando no lo fue.

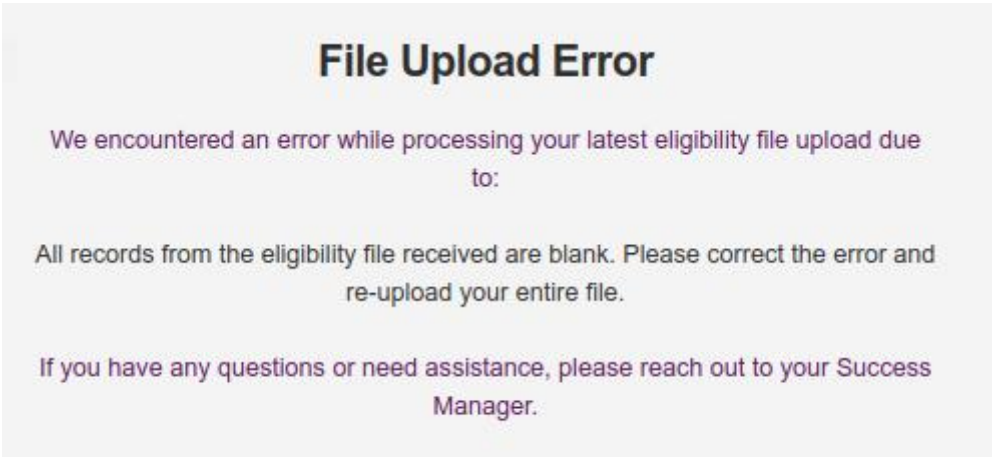
Si se sube un archivo donde directamente uno de los headers del mismo está mal, corta el procesamiento del archivo y manda la siguiente notificación:





El mensaje es poco claro, ya que habla de un header faltante (employeeId) y de un header desconocido (Employee ID), pero no aclara mucho más y el formato no es amigable. Incluso los Emails enviados a los administradores de los Corporate Partners no son claros. Este es el Email que ellos reciben:





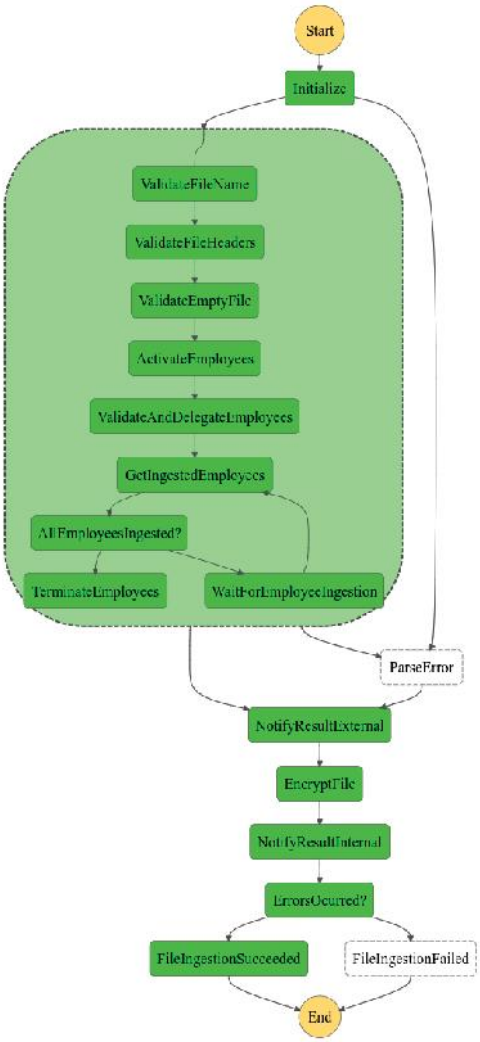
El email notifica de un error a causa de que el archivo estaba vacío o que todos los registros estaban y no se pudo procesar. Cuando el verdadero error estaba en los headers del mismo.

Y, por último, cuando se intenta subir un archivo con un nombre invalido, se recibe el siguiente mail interno:

Pero no existe un email para los administradores de los CPs notificando de ese tipo de error. Por lo que si los Corporate Partners subían un archivo con un nombre invalido, no se iban a enterar de que no se procesó por culpa del nombre. Los administradores del cliente debían ocuparse de notificar a los Corporate Partners de los problemas en el procesamiento.

Sistema Nuevo:

Se subió el mismo archivo con 9964 empleados, todos válidos y que no existían en el sistema, pero para un Corporate Partner configurado en el sistema nuevo:



Cuando termina el procesamiento del archivo, se recibe la siguiente notificación por email donde no solo muestra el resultado del procesamiento, sino que también muestra la cantidad de empleados procesados, cantidad de empleados terminados, tiempo de ejecución y localización del archivo encriptado:

[FFF2][Newflow Test][SUCCESS] Employee file ingested succesfully External aws spam ×

AWS Notifications <no-reply@sns.amazonaws.com>
to me

RESULT: SUCCESS

Corporate Partner: Newflow Test (33dssad-49b2-11ea-8d67-41f492c824bb)
CRM system: newflow

File name: eligibility_newflow.csv
File location: corporate-partners/33dssad-49b2-11ea-8d67-41f492c824bb/eligibility/eligibility_newflow.csv

Encrypted file name: eligibility_newflow-2021-09-24T01:58:21.602Z.csv.encrypted
Encrypted file location: corporate-partners/33dssad-49b2-11ea-8d67-41f492c824bb/eligibility/eligibility_newflow-2021-09-24T01:58:21.602Z.csv.encrypted

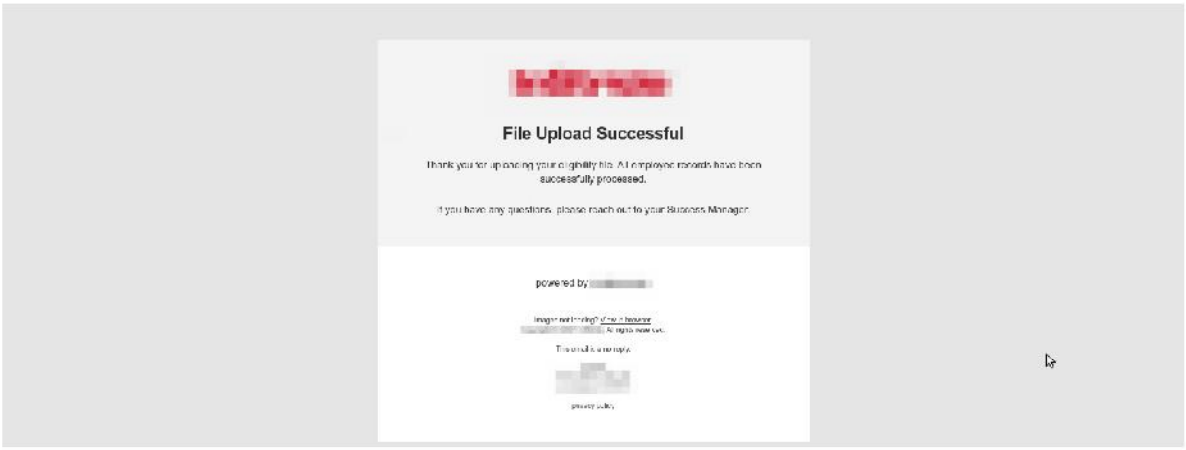
Execution start: 2021-09-24T01:58:21.602Z
Execution end: 2021-09-24T02:03:11.725Z
Execution duration: 0h 4m 50s

Employees rows total: 9964
Employees activated: 9964
Employees delegated: 9964
Employees ingested: 9964
Employees terminated: 0

Errors: []

Se puede ver que se cargaron 9964 empleados en 4 minutos y 50 segundos.

También se envía el siguiente email a los administradores de los Corporate Partners con el resultado del procesamiento:



File Upload Successful

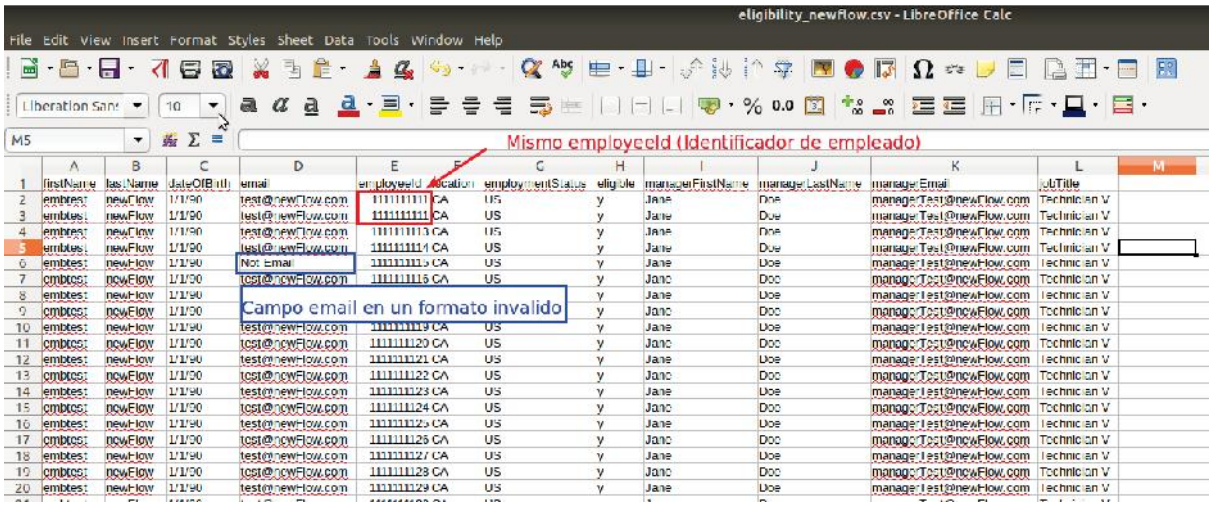
Thank you for uploading your eligibility file. All employee records have been successfully processed.

If you have any questions, please reach out to your Success Manager.

Notificaciones de errores:

Durante el procesamiento del archivo pueden ocurrir múltiples errores (líneas con datos inválidos, empleados duplicados, errores internos del sistema), por lo que generamos una jerarquía interna de errores, proveyendo mayor importancia a unos con respecto a otros a la hora de notificarlos.

Por ejemplo, tomando en cuenta el archivo que se subió para las pruebas anteriores, se le modificó las primeras dos líneas para que esos empleados tengan el mismo employeeId (El identificador de empleado) y a la línea 6 se le modificó el atributo email por un string que no posee el formato de un email:



Al subirse el archivo y procesarse, la Step Function termina en un estado de Error a diferencia del Flow anterior:

En los emails de notificación el error que se muestra como principal en el email de notificación es el error de la línea inválida por un formato de email incorrecto. Tanto como para el email de notificación interno como el email que se envía a los administradores de los Corporate Partners:

[FFF2][Newflow Test][ERROR] Employee file ingested w/ errors ["ERROR_02_INVALID_ROWS_ERROR"]

AWS Notifications <no-reply@sns.amazonaws.com>

RESULT: FAILURE

Corporate Partner: Newflow Test (33dssac-49b2-11ea-9d07-41f492c824bb)

File location: corporate-partners/33dssed-49b2-11ea-9d07-41f492c824bb/eligibility/eligibility_newflow.csv

Encrypted file name: eligibility_newflow.2021-10-25T15:25:15.568Z.csv.encrypted

Execution start: 2021-10-25T15:25:15.568Z
Execution end: 2021-10-25T15:27:49.718Z

Employees activated: 9963
Employees ingested: 9961

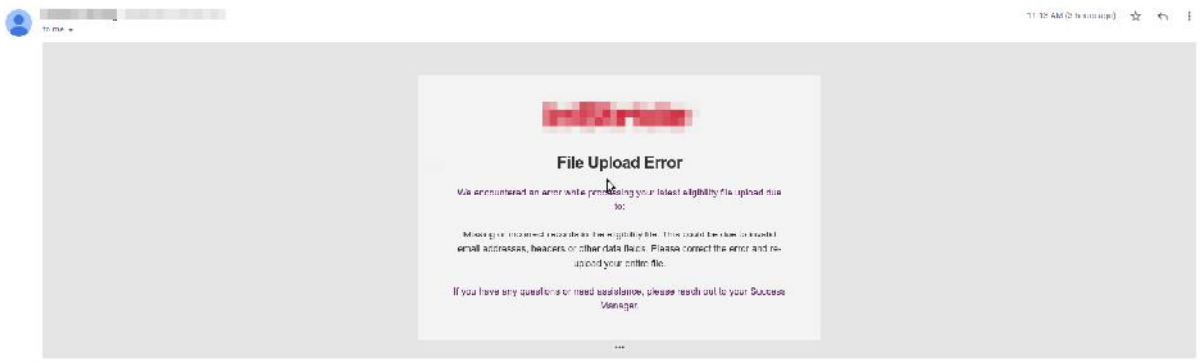
Errors: [
{
"type": "ERROR_02_INVALID_ROWS_ERROR",
"metadata": {
"firstTenRowErrors": [
{
"reason": "email field must be a valid email",
"row": 6
}
],
"totalRowErrors": 1
}
}
]

Execution start: 2021-10-25T15:25:15.568Z
Execution end: 2021-10-25T15:27:49.718Z
Execution duration: 0h 2m 34s

Employees rows total: 9964
Employees activated: 9963
Employees delegated: 9961
Employees ingested: 9961
Employees terminated: 1

Errors: [


```
{
  "type": "ERROR_02_INVALID_ROWS_ERROR",
  "metadata": {
    "firstTenRowErrors": [
      {
        "reason": "email field must be a valid email",
        "row": 6
      }
    ],
    "totalRowErrors": 1
  }
}
]
```



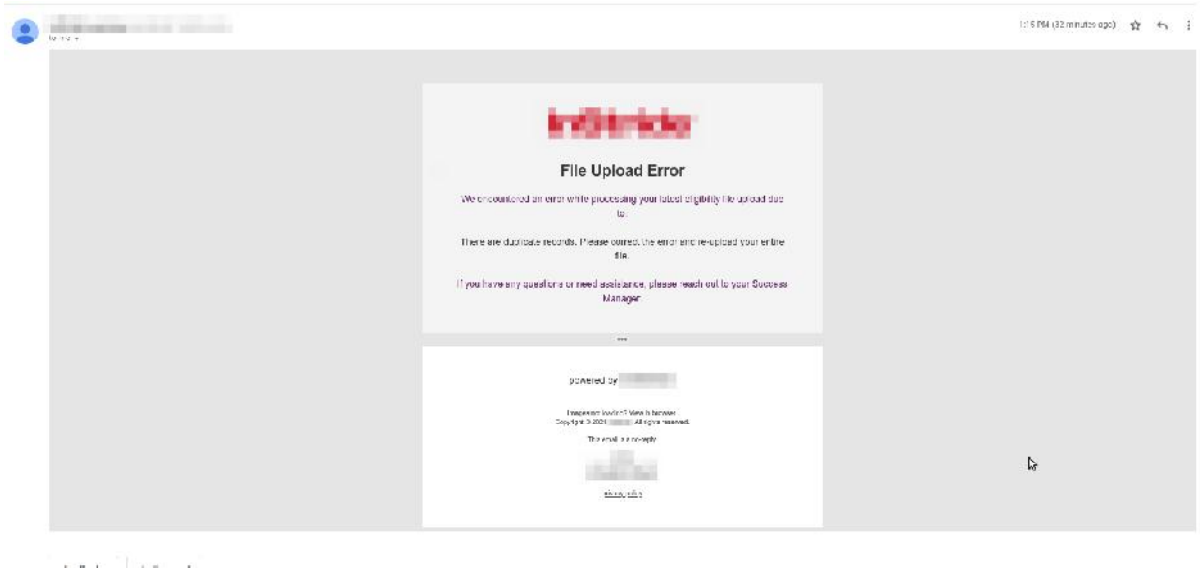
File Upload Error

We encountered an error while processing your latest eligibility file upload due to:

Missing or incorrect records in the eligibility file. This could be due to invalid email addresses, headers or other data fields. Please correct the error and re-upload your entire file.

If you have any questions or need assistance, please reach out to your Success Manager.

Cuando se corrige el error de la línea invalida y se vuelve a procesar el archivo, se va a recibir el siguiente error:



File Upload Error

We encountered an error while processing your latest eligibility file upload due to:

There are duplicate records. Please correct the error and re-upload your entire file.

If you have any questions or need assistance, please reach out to your Success Manager.

File Upload Error

We encountered an error while processing your latest eligibility file upload due to:

There are duplicate records. Please correct the error and re-upload your entire file.

If you have any questions or need assistance, please reach out to your Success Manager.

[FFF2][Newflow Test][ERROR] Employee file ingested w/ errors ["ERROR_13_DUPLICATED_UID"]

AWS Notifications <no-reply@sns.amazonaws.com>
to me

RESULT: FAILURE

Corporate Partner: Newflow Test ([illegible])
[illegible]

File name: eligibility_newflow.csv
File location: [illegible]

[illegible]
[illegible]

Execution start: 2021-10-25T16:11:20.188Z
Execution end: [illegible]
Execution duration: 0h 3m 42s

Employees rows total: 9954
Employees activated: 9965
Employees delegated: 9902
Employees rejected: [illegible]
Employees terminated: 0

Errors: [
{
"type": "ERROR_13_DUPLICATED_UID",
"includeId": [
"firstTenRowErrors": [
{
"reason": "Duplicated UID in line 2 for CP: [illegible]. Amounts of appearances: 2",
"row": 2
},
{
"reason": "Duplicated UID in line 3 for CP: [illegible]. Amounts of appearances: 2",
"row": 3
}
],
"totalRowsDuplicated": 2
}
}
]

```
Execution start: 2021-10-25T16:11:20.188Z
Execution end: 2021-10-25T16:15:02.743Z
Execution duration: 0h 3m 42s

Employees rows total: 9964
Employees activated: 9963
Employees delegated: 9962
Employees ingested: 9962
Employees terminated: 0

Errors: [
{
 "type": "ERROR_13_DUPLICATED_UID",
 "metadata": {
  "firstTenRowErrors": [
   {
    "reason": "Duplicated UID in line 2 for CP: 33dssad-49b2-11ea-8d67-41f492c824bb. Amounts of appearances: 2",
    "row": 2
   },
   {
    "reason": "Duplicated UID in line 3 for CP: 33dssad-49b2-11ea-8d67-41f492c824bb. Amounts of appearances: 2",
    "row": 3
   }
  ],
  "totalRowsDuplicated": 2
 }
}
]
```

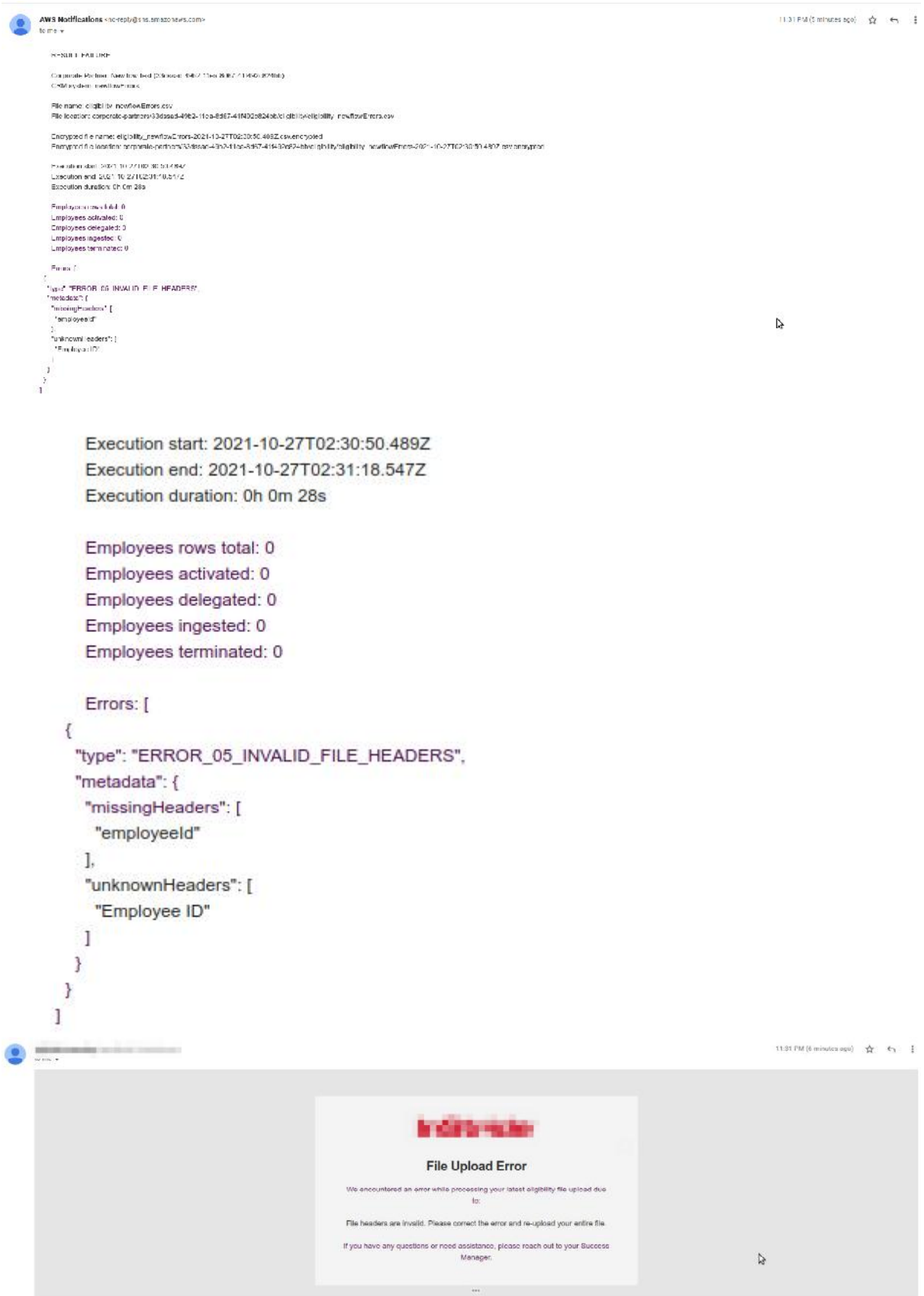
La decisión de tener una jerarquía de errores y de no mostrar múltiples errores en un email y solo de a uno, fue una decisión que se tomó en base a los requisitos del cliente. Creían que un email notificando varios errores a la vez podía confundir a los administradores de las empresas y revisando casos de usos anteriores, era muy raro que dos o más errores surgieran durante el mismo procesamiento. Por lo que se decidió enviar un solo error principal por email y para eso se decidió una jerarquía.

La jerarquía es:

1. Error interno del sistema.
2. Nombre de archivo invalido.
3. Columnas faltantes en el archivo.
4. Líneas con columnas inválidas.
5. Líneas con identificadores de empleados repetidos.
6. Porcentaje de empleados a terminar supera el límite.

El error interno del sistema es el más prioritario porque significa que la causa del error no es un problema del archivo o de los registros si no que es algo interno que debe ser notificado y arreglado en el menor tiempo posible.

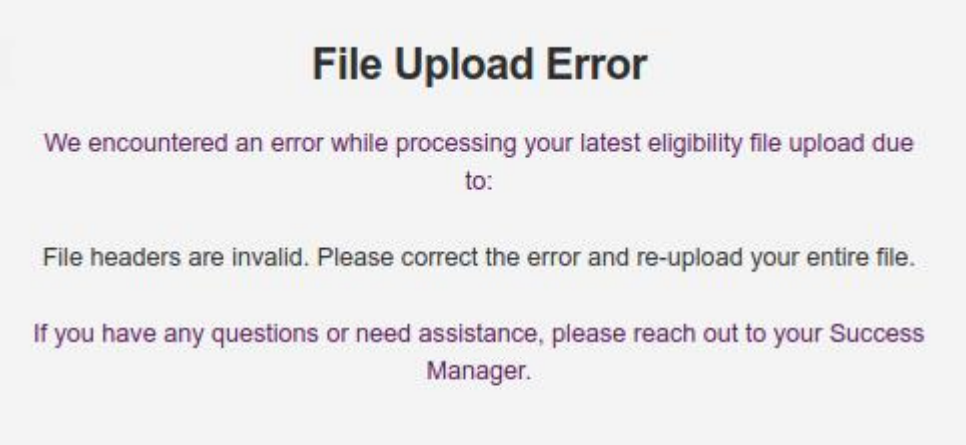
En el caso de headers incorrectos en el archivo, esta es la notificación recibida:



Execution start: 2021-10-27T02:30:50.489Z
Execution end: 2021-10-27T02:31:18.547Z
Execution duration: 0h 0m 28s

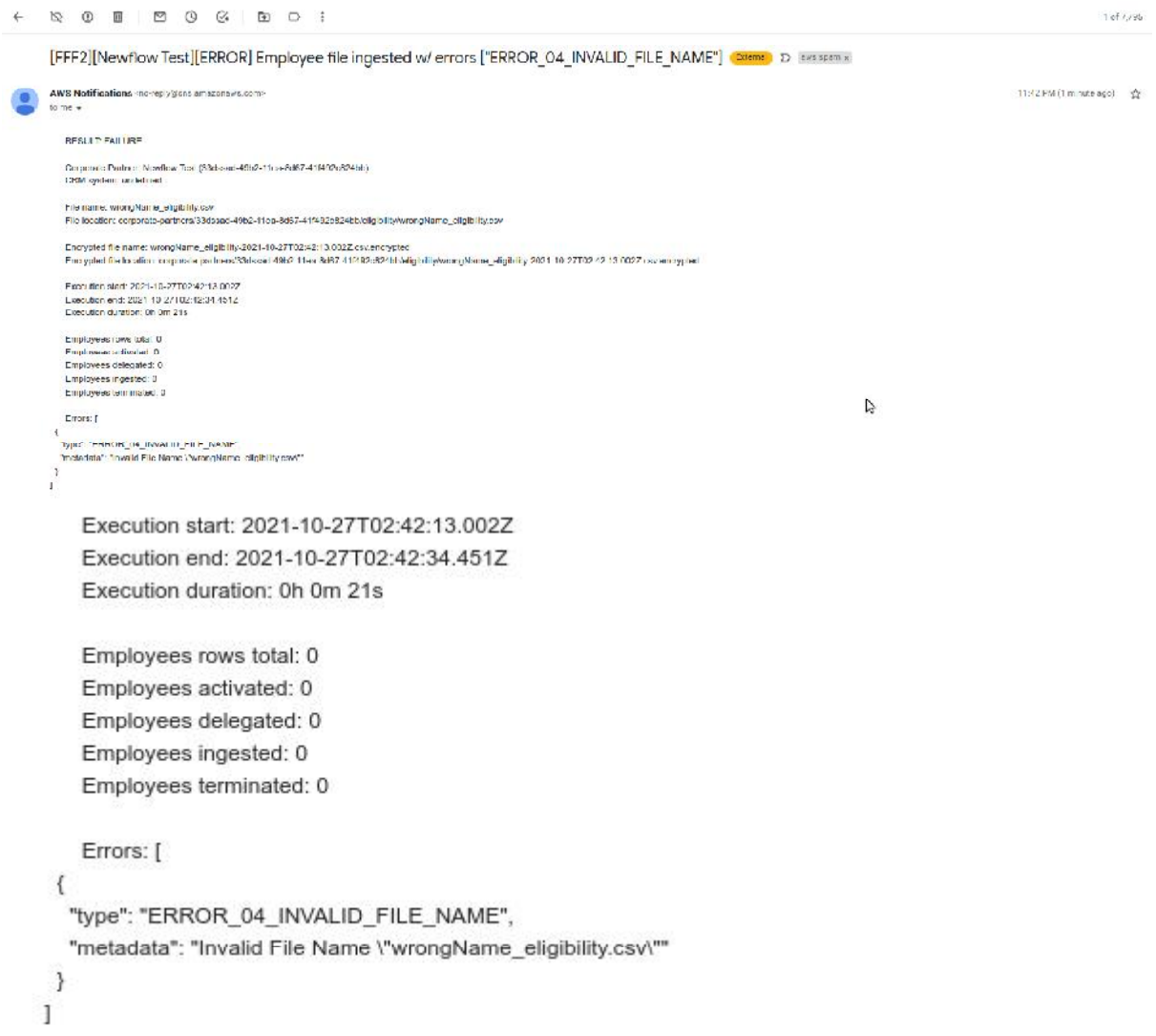
Employees rows total: 0
Employees activated: 0
Employees delegated: 0
Employees ingested: 0
Employees terminated: 0

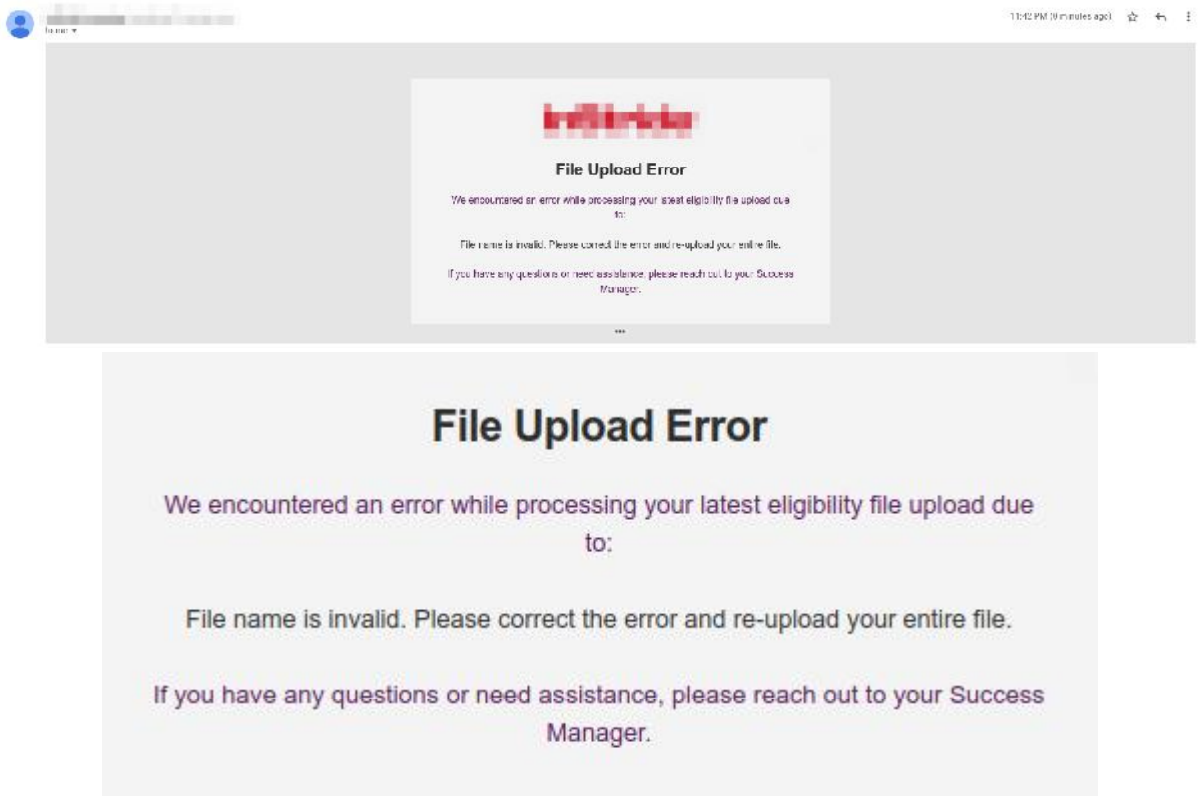
Errors: [
{
"type": "ERROR_05_INVALID_FILE_HEADERS",
"metadata": {
"missingHeaders": [
"employeeId"
],
"unknownHeaders": [
"Employee ID"
]
}
}
]



En la notificación interna se puede ver exactamente cuál fue el error, los headers incorrectos están detallados en un mejor formato y muestra que no se procesó ningún empleado.

Y si se sube un archivo de empleados con un nombre incorrecto, estas son las notificaciones recibidas:





La notificación interna sigue el mismo formato que las anteriores y la notificación al cliente detalla que el error fue a causa de un nombre invalido de archivo. Por lo que mejoró la notificación errónea del flujo anterior que marcaba que el archivo se recibió vacío cuando en realidad los headers del mismo eran incorrectos.

Mejoras posibles:

A pesar de las grandes mejoras realizadas en el nuevo sistema, todavía hay varios puntos donde se puede mejorar el performance o el mantenimiento del sistema. Los puntos que se encontraron y se planean actualizar son:

1. **Cambiar la base de datos por una BD relacional:** Actualmente el sistema utiliza MongoDB como base de datos. MongoDB es una base de datos no relacional, por lo tanto, la información no está guardada de forma estructurada y no utiliza SQL para realizar consultas. Esto afecta a la hora de realizar consultas entre distintas entidades ya que requiere realizar distintas consultas a cada colección para obtener una entidad y luego realizar otra consulta a otra colección para obtener la segunda entidad. Caso de ejemplo:

   Si se requiere saber el estado del registro de Approval de un empleado y lo único que se tiene del mismo es su Identificador (Un email por ejemplo), se busca al empleado en la colección de empleados en base al identificador y luego cuando se obtiene el registro en base al “id” se busca el registro de Approval en la colección de Approvals que esté relacionado con ese id de empleado. En una base de datos Relacional, con una sola consulta SQL se puede realizar esa operación reduciendo el número de llamadas a realizar.

En un primer momento se optó por MongoDB ya que no había una estructura clara de las entidades y sus relaciones. Más avanzado el sistema, las entidades fueron definidas junto con sus relaciones por lo que las ventajas de una base de datos relacional empezaron a ser más evidentes. Una base de datos relacional no solo va a ayudar a tener un sistema mejor estructurado si no que va a evitar tener que realizar múltiples consultas a la base de datos ya que consultas de entidades relacionadas se podrían realizar en una sola consulta.

2. **Unificar los registros de empleados con su historial de Elegibilidad:** Actualmente un registro de empleado tiene también un registro en una colección aparte con el historial histórico de elegibilidad del mismo. Cada cambio en la elegibilidad de un empleado se guarda en dicho registro para mantener un historial de cambio en los empleados. Para evitar tener que actualizar dos colecciones diferentes ante cada cambio en la elegibilidad, la mejor solución es que el registro histórico del empleado se guarde dentro del mismo empleado. Esto requiere un cambio en el procesamiento del empleado para que, en vez de actualizar otra colección, modifique el documento de empleado en el caso de que cambie la elegibilidad. Y luego un script para actualizar todos los empleados ya existentes para unificar las colecciones de empleados con sus registros de historial.

3. **Mejorar el sistema de encriptación de registros:** El equipo de plataforma por cuestiones de seguridad agregó un servicio de encriptación para mantener toda la data privada de un empleado encriptada. El problema es que la capa de encriptación no posee la capacidad de procesar registros a la velocidad que lo realiza el sistema de procesamiento de archivos. Por lo que se agregó un delay entre los eventos de creación y actualización así la capa de encriptación puede encriptar todos los registros correctamente.

   Se debería actualizar la capa de encriptación para que soporte la misma cantidad de registros por minuto que el procesamiento de archivos, así ambos servicios pueden estar sincronizados e ir a la misma velocidad.

4. **Habilidad de volver a procesar un archivo automáticamente:** Como se mencionó anteriormente, existe una jerarquía de errores con respecto al procesamiento de un archivo. En el caso de que el error que surgió durante el procesamiento no sea un error derivado del archivo si no que fue un error externo o un error del sistema y el archivo deba ser procesado de nuevo actualmente se debe des-encriptar el archivo y volver a subir manualmente al bucket de S3.
   Se busca tener la forma de automatizar ese proceso y que exista una Lambda o servicio que des-encripta el archivo y automáticamente lo suba al bucket de S3 para que se vuelva a procesar sin necesidad de intervención manual.

Resultados:

El primer resultado que salta a la vista es la velocidad del sistema. El sistema anterior tardaba 29 minutos en procesar un archivo contra los 5 minutos que tarda el sistema nuevo. Esta velocidad generó que los nuevos Corporate Partners estuvieran muy feliz con el resultado ya que permite una integración de los nuevos empleados al sistema más rápida y no deben esperar horas (en el caso de archivos muy grandes) para que termine el procesamiento de los empleados.

También el nuevo sistema de notificaciones y jerarquía de errores ayudó a evitar tantos problemas con los archivos por parte de los Corporate Partners. Anteriormente ante cada error en el procesamiento, los administradores de los Corporate Partners no podían saber claramente en donde estaba el problema ya que los errores del sistema y del archivo estaban poco claros en las notificaciones. Errores del sistema se marcaban como si fueran del archivo o viceversa, por lo que administradores del cliente debian mirar los logs del sistema y encontrar la causa del error para notificar a los Corporate Partners.

Ahora los Corporate Partners pueden saber específicamente el error, si fue proveniente del archivo, del sistema o un error desconocido. Como la mayoría de las veces el error es algo proveniente del archivo, los administradores de los Corporate Partners son capaces de solucionar el problema y volver a subir el archivo sin la necesidad de que los administradores del cliente intercedan y tengan que encontrar el problema. Y en el caso de que resulte ser un error del sistema, ahora los administradores pueden ver cuál de todos los errores fue, si el archivo pudo procesarse y cuántos empleados fueron procesados en el caso de que se haya procesado parcialmente.

Y, por último, además de la mejora en la velocidad y notificaciones de errores la última gran mejora fue la separación en entidades diferentes los conceptos de elegibilidad de un empleado y la aprobación del mismo. Esto generó que empleados que conceptualmente eran elegibles para ingresar al sistema puedan pasar por un proceso de selección y aprobación de sus manager para ver a qué cursos puedan acceder o no. El sistema anterior tenía un concepto de aprobación que estaba ligado a la elegibilidad de un empleado. Esto hacía que un empleado elegible cuyo acceso a un curso fuera denegado pasará a ser no elegible. Algo erróneo ya que debería poder seguir accediendo al sistema y acceder a otros cursos. El nuevo sistema al tener los conceptos separados hace que la experiencia del usuario sea más certera a lo diseñado y ayuda a darle a los Corporate Partners y sus empleados un sistema más estable y eficaz.

Todos los ítems mencionados anteriormente llevaron a obtener un feedback positivo por parte de los Corporate Partners, los usuarios y administradores. Los sistemas viejos pasaron a llamarse sistemas Legacy y no se dan como sistemas disponibles a los nuevos Corporate Partners, si no que el único sistema disponible para ellos es el sistema nuevo. Y los Corporate Partners que están utilizando los sistemas Legacy están siendo promovidos al nuevo sistema.

Esto llevó a que el cliente quede muy satisfecho con el sistema nuevo, que la plataforma crezca en tamaño y que provea confianza antes nuevos Corporate Partners que busquen entrar en el sistema.

Referencias bibliográficas:

- https://docs.aws.amazon.com/
- https://aws.amazon.com/event-driven-architecture/
- https://aws.amazon.com/blogs/compute/operating-lambda-understanding-event-driven-architecture-part-1/
- https://event-driven-architecture.workshop.aws/
- https://docs.nestjs.com/
- https://docs.nestjs.com/faq/serverless
- https://www.serverless.com/examples/aws-node-typescript-nest
- https://docs.mongodb.com/
- https://github.com/labs42io/clean-code-typescript