UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de un Sistema de Inteligencia Fiscal

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniera en Computación

P R E S E N T A

Ana Laura Hernández Galindo

ASESORA DE INFORME

Ing. Josefina Rosales García

Ciudad Universitaria, Cd. Mx., 2022

Índice

Índice de Anexos

1. Objetivo

El presente documento tiene la finalidad de describir cómo fue mi experiencia en la práctica profesional y la manera en la que mis aptitudes y técnicas, propias de la carrera de ingeniería en computación, fueron de utilidad como parte de la resolución del requerimiento de adquisición, almacenamiento, análisis y procesamiento de datos para la implementación del Sistema de Inteligencia Fiscal. Aunado a lo anterior, se contemplará el marco teórico, la definición del problema, la metodología empleada, mi participación profesional y los resultados obtenidos como parte de las distintas actividades que desempeñé a lo largo del tiempo que laboré para la compañía.

2. Marco teórico

2.1 Inteligencia Fiscal

En la actualidad, las organizaciones se encuentran en la necesidad de aplicar correctamente la inteligencia fiscal, esto se refiere a tener mejores prácticas fiscales para evitar riesgos al momento de rendir aclaraciones ante el sistema tributario que rige nuestro país. Esto no sólo les permite mantenerse en estado positivo como empresa, sino que también les provee de la capacidad de tener control sobre su información fiscal generada día con día. A través del uso de técnicas especializadas para el manejo correcto de los datos y el asesoramiento de administración tributaria, se logra evitar cualquier tipo de problemas relacionados con la evasión fiscal y los ilícitos.

2.2 Factura electrónica

La factura electrónica, conocida también como Comprobante Fiscal Digital por Internet (CFDI), fue concebida como un instrumento de control documental del proceso de facturación, tanto para evitar la omisión de ventas y la inclusión de compras falsas, así como el incumplimiento tributario y la evasión fiscal. Se trata de un documento digitalizado de la administración tributaria que se presenta como un servicio focalizado en la misión de brindar apoyo al ciudadano bajo criterios de eficiencia basada en la generación y difusión de datos. De forma adicional, la factura electrónica pretende cambiar la relación con el contribuyente, el sector privado en general y el propio sector público; ya que permite ofrecer servicios adicionales al contribuyente, alejándose un poco de la lógica tradicional de control y represión, así como ayudar al sector público a mejorar su función reguladora para asegurar la efectiva competencia en los mercados y a transparentar los precios de contrataciones públicas de servicios, así como poder incluir un mecanismo novedoso de colaboración con el sector privado.

Para quienes hacemos uso de las tecnologías de la información, tener a nuestra disposición la factura electrónica de nuestros clientes, nos permite brindarles apoyo en caso de requerir un mejor control sobre su información fiscal y otorgar beneficios relacionados con proyecciones a futuro a través del análisis de los datos, la Big data y el Machine Learning. Así mismo, podremos asegurarnos de almacenar y resguardar sus documentos para su consulta en cualquier momento y circunstancias.

2.3 Big Data

Nos encontramos en un mundo repleto de datos, donde los datos no sólo nos rodean sino que también son producto de nosotros mismos y a menudo sin darnos cuenta. Por

lo anterior, ha sido necesario extraer información útil y relevante a partir de todos estos datos, siendo esto, uno de los objetivos primordiales de la Big Data.

El almacenamiento y procesamiento de datos ha sido una de las tareas asociadas a las computadoras desde sus inicios. La primera computadora comercial (UNIVAC I) fue construida en 1951 y adquirida por la oficina del Censo de Estados Unidos para tratar la ingente cantidad de información obtenida en los censos que se realizaban cada 10 años, además de otra cantidad de datos que comenzaban a recopilarse a través de otras industrias. Pronto se revelaría la capacidad de la computadora para realizar cálculos y predicciones estadísticas imposibles hasta el momento. Uno de sus mayores éxitos fue la predicción del resultado de las elecciones presidenciales de 1952. La consecuencia más importante de esta anécdota fue que la población en general se hizo consciente de las posibilidades que ofrecía el manejo de datos por parte de las computadoras.

2.4 Bases de datos relacionales

Uno de los inconvenientes al momento de manejar datos es cuando se presenta la situación de información redundante, la cual, como sabemos, ocupa espacio en memoria y tiene un coste, además de que al repetir información es fácil que pueda existir algún tipo de equivocación en los registros. Para resolver el problema anterior, Edgar “Ted” Codd publicó un artículo en 1970 que revolucionaría el mundo de las bases de datos. En este artículo Cood proponía dividir los datos en relaciones (estructuras con forma de tablas) que se combinan entre sí para obtener información, de tal manera que el problema de la redundancia se solucionó dividiendo tablas grandes en varias más pequeñas. Su propuesta iba más allá, puesto que propuso que el diseño de una base de datos debía mantenerse separado de su implementación física. Es decir, el usuario no tenía que preocuparse de cómo se distribuía esta información o de si existían huecos o no. De esto se encargaría un programa especializado conocido como sistema gestor de bases de datos. Así mismo, Codd planteó separar el nivel físico (cómo se graban los

datos) del nivel lógico o conceptual que era del principal interés del diseñador de bases de datos. Muy pronto e inspirándose en las ideas de Codd se presentaría una nueva versión del lenguaje de consultas SEQUEL que ya se tenía en IBM, denominándose ahora como SQL propuesto por el fundador de Oracle Lawrence J. Ellison y que hasta la fecha seguiría siendo el estándar en los lenguajes de consultas de bases de datos relacionales.

2.5 Ciencia de datos y Machine Learning

El objetivo de la ciencia de datos es descubrir el significado de los datos mediante la combinación de varias herramientas, algoritmos y principios de aprendizaje automático. Se ocupa principalmente de conjuntos de datos y utiliza habilidades estadísticas, matemáticas y de conocimiento de dominio empresarial.

La inteligencia artificial se refiere a la manera en la que las computadoras pueden desarrollar capacidades de percepción similares a las humanas con la finalidad de sentir el entorno, comprender datos detectados, tomar acciones adecuadas y aprender de esta experiencia para el rendimiento futuro. Machine Learning, o por su traducción al español Aprendizaje automático, es una rama de la inteligencia artificial que consiste en la aplicación práctica de modelos matemáticos que aprenden de forma automática con base en datos históricos. Se divide en dos tipos de problemas, el primero es el aprendizaje supervisado (con variable objetivo) que utiliza la regresión para las variables continuas (numéricas de magnitud) y la clasificación para variables categóricas (que se dividen en etiquetas o clases distintas) y el segundo problema que es el aprendizaje no supervisado (sin variable objetivo) con reducción de la dimensionalidad para continuas y clustering o agrupamiento para categóricas.

3. Descripción de la empresa o medio en que labora

3.1 Acerca de la empresa

STO Consulting (Servicios, Tecnología y Organización S.A. de C.V.) es una empresa de consultoría especializada en Contabilidad y Facturación Electrónica fundada en el año 2002 y con sede en la Ciudad de México. Se trata de una empresa 100% mexicana, con expansión a Latinoamérica (con oficinas también en Bogotá, Colombia), con un tamaño de 51 a 200 empleados, que está Asociada a Platinum de Oracle y cuyo negocio es Proveedor Autorizado de Certificación (PAC) para el SAT, cuenta con un equipo de trabajo con más de 140 certificaciones de Oracle y con alto grado de experiencia en el diseño, implementación y soporte de soluciones que transforman y optimizan los procesos clave de la organización para la prestación de servicios de certificación en el intercambio de comunicaciones digitales del ámbito fiscal y comercial. Estuvo en primer lugar como partner de aplicaciones en México durante 6 años consecutivos, ha estado en el top 10 de las mejores consultorías de Tecnologías de Información del país y fue partner del año en Sector Público por Oracle en el 2018.

La misión de la empresa es entregar soluciones basadas en servicios de consultoría con productos propios y de terceros en Tecnologías de Información para toda Latinoamérica y diversas industrias del sector público y privado, mediante un equipo de profesionales certificados, con el propósito fundamental de agregar valor al negocio de sus clientes y de generar riqueza y satisfacción a los miembros de la cadena productiva.

La visión de la empresa es ser una de las mejores firmas de consultoría especializada en soluciones de Tecnologías de Información en el continente americano; con productos y servicios más competitivos e innovadores del mercado, respaldada por un grupo de profesionales comprometidos, responsables y en continua certificación.

3.2 Descripción del puesto de trabajo

Mi puesto de trabajo tiene el nombre de Consultor BI Jr. con enfoque en el desarrollo de programas codificados en Python. Me encuentro asignada al Departamento de Analítica y formo parte del Equipo Técnico para el desarrollo del sistema de Inteligencia Fiscal. Mi labor consiste en la implementación de procesos para adquirir, almacenar y llevar a las tablas de bases de datos la información encontrada en la factura electrónica (CFDI) de cada uno de los clientes que tenemos. También realizo otras actividades como la generación de reportes de nómina para una determinada empresa, tengo una participación activa en el Laboratorio de Ciencia de Datos para la mejora continua de nuestros procesos en general, reviso los documentos Log generados de la ejecución de la noche para asegurarme de que todo ha salido bien y corregir en caso de que haya habido algún inconveniente, ayudo a la resolución de problemas relacionados con el servicio de Elasticsearch, doy mantenimiento a una API de conversión de documentos de formato XML a Json y algunas veces hago manuales para un determinado cliente y si es necesario se lo explico a detalle en una sesión virtual. Por lo regular me encuentro a la expectativa de requerimientos nuevos y a la mejora continua y mantenimiento de mis procesos.

4. Antecedentes

Antes de que se propusiera poner en marcha la implementación del sistema de Inteligencia Fiscal, la empresa ya contaba con un Portal de Facturación, el cual se encontraba conformado por los siguientes módulos: Timbrado, Emisión, Nómina, Autofactura, Gastos y anticipos, Proveedores y Contabilidad electrónica. Muy pronto surgiría un requerimiento propuesto por el SAT, en el que se les solicitaría a las organizaciones tener 5 años históricos de todas sus facturas por si en algún momento dicho servicio tributario tuviese la necesidad de hacer una auditoría a su compañía.

Debido a lo anterior, se tuvo la necesidad de crear un nuevo módulo para el Portal de Facturación al que se le denominó inicialmente como Bóveda Fiscal.

De manera previa, el cliente ya tenía la posibilidad de introducir sus datos al portal y generar sus propias facturas electrónicas con el servicio de Timbrado. Dichas facturas se guardaban en una Base de datos. El cliente sólo tenía la necesidad de almacenar su información, pero no contaba con que dichos documentos pudiesen serle de utilidad alguna para otros propósitos, hasta que llegó el requisito del SAT.

La idea se llevó a cabo luego de una reunión y se dieron propuestas sobre el alcance que podría llegar a tener este nuevo proyecto. Uno de los primeros requisitos que debía cumplir era el de realizar el análisis de los datos a través de indicadores para lograr comparativos de resultados históricos, a este proceso se le llamó Compulsa, el cual debía comparar la cantidad de documentos obtenidos desde una Fuente A con los obtenidos desde una Fuente B.

Con el paso del tiempo comenzaron a surgir nuevos requerimientos, algunos de ellos específicos por cada cliente y otros más serían todo lo que la Bóveda Fiscal tendría que ofrecer como producto. Los nuevos requisitos tenían como objetivo utilizar los datos más relevantes de la factura electrónica y lograr darles alguna utilidad que permitiera a los clientes conocer más acerca de su negocio por medio de indicadores estadísticos y reportes en Excel. Aunado a lo anterior, también se solicitó generar un buscador para consultas de comprobantes por medio de introducir algún dato relevante como RFC o versión del documento.

Finalmente, se consideró adicionar la Bóveda Fiscal como módulo del Portal de Facturación. Meses más tarde, se decidiría que su nombre definitivo cambiaría a Inteligencia Fiscal.

5. Definición del problema

El sistema de Inteligencia Fiscal debe ser capaz de proporcionar a los interesados distintos tipos de beneficios relacionados con su información fiscal. En primer lugar, debe permitir generar y gestionar información con funcionalidad de conexión al Servicio de Administración Tributaria (SAT) para realizar la descarga de Factura electrónica (CFDI). Acto seguido, debe almacenar de forma ordenada y clasificada dichos documentos descargados para poder ser consultados periódicamente. Además, será necesario que los datos obtenidos de cada factura sean procesados para fines estadísticos, con la intención de proporcionar al cliente información relacionada con su situación fiscal frente a la entidad tributaria, indicadores de facturas (tanto emitidas como recibidas), tableros analíticos y reportes en formato .XLS. Aunado a lo anterior, se contemplará tener un proceso que logre realizar una comparación eficiente entre la información que se tiene de manera local a través de fuentes ya establecidas y la información que se obtiene por medio del SAT; de tal manera que se puedan identificar posibles diferencias existentes entre las dos fuentes de información mencionadas. Finalmente, se debe presentar al cliente una interfaz gráfica instrumentada al usuario con la que podrá interactuar mediante un navegador Web. Dicho portal debe ser visualizado desde la computadora (véase Anexo 1) o desde un celular, ya que es necesario que cuente con diseño responsivo de acuerdo a la pantalla.

6. Análisis y metodología empleada

Decidí que seguiría cada una de las etapas de desarrollo del software, ya que tenía claro que mis procesos iban a ser codificados con un lenguaje de programación.

6.1 Análisis de requerimientos

Describí cada una de las características operacionales que debía tener mi proceso, poniendo especial atención a lo que el cliente necesitaba, ya que a veces, cada cliente pedía algo adicional para su producto. Había quienes necesitaban reportes de nómina, otros querían ser capaces de obtener su información fiscal en otro formato en lugar de xml, también quienes no sólo iban a almacenar CFDI sino también Pólizas y necesitaban que sus documentos (recibos, complementos de pagos, fianzas y endosos) estuviesen relacionados entre sí y otros detalles particulares como el hecho de que algunos facturaban con nuestra empresa y había quienes no lo hacían.

Requisitos generales que aplicaban para todos los clientes:

- Creación de una estructura de carpetas (véase Anexo 2) en la que se debía considerar una para la adquisición de documentos, otra para el almacenamiento de archivos utilizados para la búsqueda y una última para guardar los procesos de ejecución.
- Adquisición y clasificación de documentos por cada RFC que tuviese el cliente y por cada versión del comprobante fiscal.
- Conversión de documentos de formato .xml a formato .json para poder ser cargados en un índice en un repositorio en la nube.
- Selección y almacenamiento de información útil para fines estadísticos en tablas de la base de datos productiva.
- Respaldo de comprobantes fiscales en una carpeta general para su consulta y descarga desde el portal.
- Procesamiento de datos para lograr la visualización de la información a través de indicadores de CFDI, tableros analíticos preventivos y proceso de compulsa.

6.2 Diseño de la solución

La codificación de los procesos se realizó con Python 3, se utilizaron librerías para la conexión con la base de datos de MySQL, con el repositorio de índices de Elasticsearch y también para trabajar con formato de archivos xml, csv y json.

Se consideró que serían un total de 6 procesos, de los cuales 3 pertenecerían a requerimientos generales del sistema de Inteligencia Fiscal y los otros 3 a requisitos extras solicitados por algún cliente en particular.

Diagramas de flujo principales:

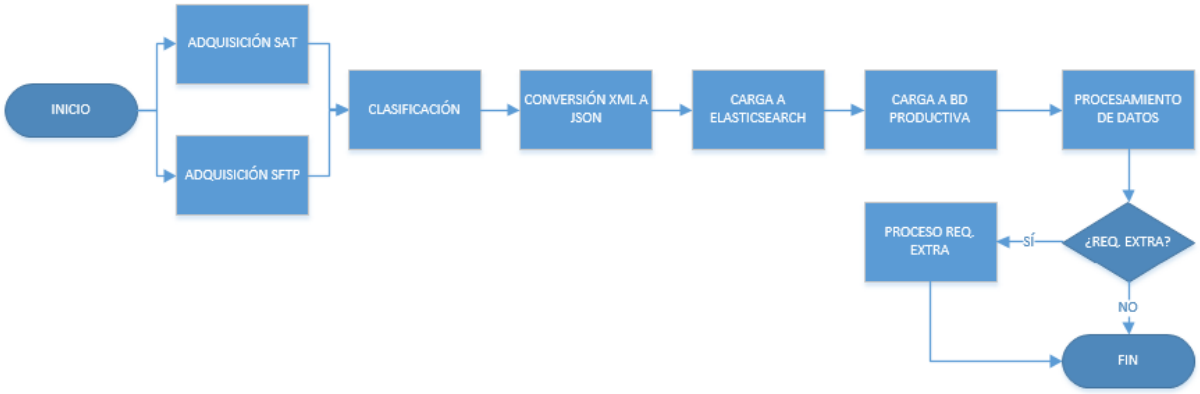


Figura 1. Diagrama de Flujo Proceso General versión 1

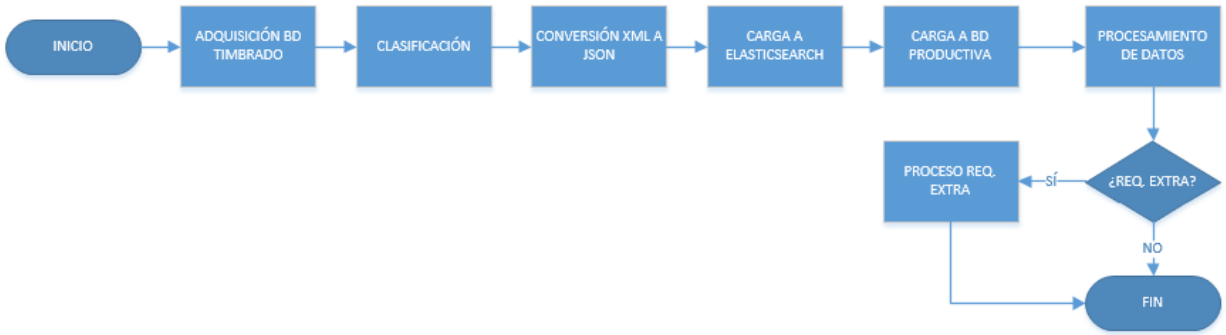


Figura 2. Diagrama de Flujo Proceso General versión 2

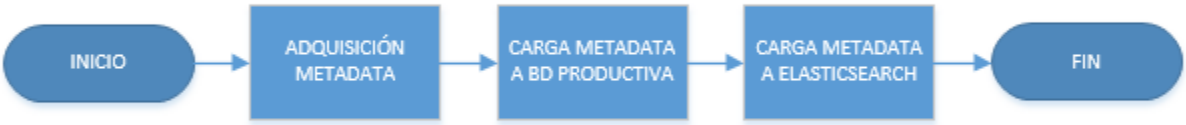


Figura 3. Diagrama de Flujo Proceso General para Metadata

Diagramas de flujo para requerimientos extras:

Figura 4. Diagrama de Flujo Proceso de Relación de documentos

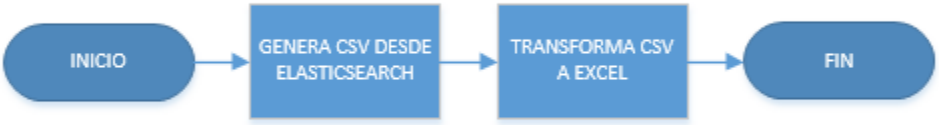


Figura 5. Diagrama de Flujo Proceso Genera reporte de nómina

Figura 6. Diagrama de Flujo Proceso Transforma XML a Excel

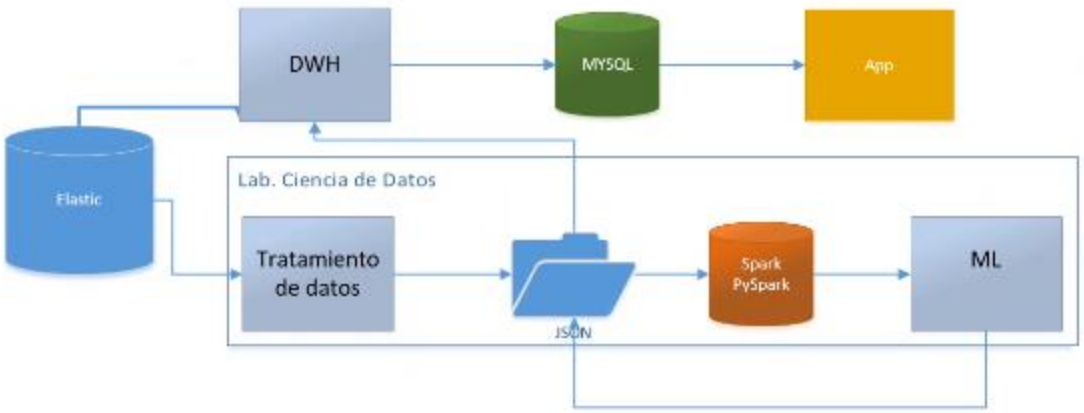


Figura 7. Diagrama del Laboratorio de Ciencia de datos

6.3 Implementación

Archivos ejecutables

Para cada cliente disponemos de tres archivos ejecutables Python, los cuales tienen la nomenclatura siguiente:

1. CICLO_CLASIFICACION_CLIENTE.py (Proceso General versión 1)
2. CICLO_XMLFROMBD_CLIENTE.py (Proceso General versión 2)
3. INGESTA_METADATA_CLIENTE.py (Proceso General para Metadata)

Cada uno de los anteriores códigos maneja un documento de propiedades al que sólo tenemos acceso los desarrolladores por motivos de seguridad y en el que se encuentran datos de conexión a bases de datos que pertenecen al cliente, así como otros parámetros necesarios para que el programa al que corresponde funcione.

1. PROPIEDADES_DOC (Para el Proceso General versión 1)
2. PROPIEDADES_BD (Para el Proceso General versión 2)

3. PROPIEDADES_METADATA (Para el Proceso General de Metadata)

El archivo *CICLO_CLASIFICACION_CLIENTE.py* está diseñado para cuando el origen de datos es Descarga Masiva (SAT) o SFTP, en el la información llega a unas determinadas carpetas. Dicho código está conformado por las funciones que se mencionan a continuación:

a) origen()

Verifica si en la ruta de adquisición de los CFDI se encuentran archivos .zip para descomprimirlos o .xml para procesarlos tal cual han llegado. Manda a llamar a la función de clasificación de acuerdo a cada RFC.

b) clasificación(x,y)

Recibe dos parámetros, el primero es la ruta inicial de los archivos de origen y el segundo corresponde al RFC. Clasifica los archivos de acuerdo a su versión. Si la clasificación resultó un éxito, dichos archivos pasan a una carpeta en la que se puede comprobar que sí se han clasificado correctamente.

c) cambiaRFC()

De acuerdo a la cantidad de RFC que tenga el cliente, va ejecutando uno a uno los procesos necesarios para la carga en el repositorio y en la base de datos. Ejecuta en primera instancia, la función xmltojson para obtener el directorio JSON y posteriormente manda a llamar a la función que se encargará de subir los datos al repositorio (Elasticsearch). Al finalizar todos los procesos, llama a la función que pasará los xml nuevos al directorio en el que ya están los archivos que fueron procesados con éxito.

d) xmltojson(x)

Recibe la ruta de la carpeta en la que se encuentran los archivos CFDI nuevos a punto de ser procesados. Crea los directorios necesarios, el primero que contendrá los archivos JSON y el segundo los archivos XML_BAD. Renombra los xml, quita los elementos no indispensables y transforma los documentos de xml a json. Retorna la ruta de los json.

e) cargaDatos(x,y,z)

Recibe tres parámetros, el primero corresponde a la ruta de los archivos json, el segundo es el nombre del índice que se encuentra en el repositorio y el tercero es el RFC del cliente. Verifica que sí existan archivos para poder subirlos, también comprueba que los archivos nuevos no estén repetidos, es decir, que no sean algunos que ya se han procesado antes y finalmente sube al índice correspondiente. Si el índice no existe, ejecuta una función que lo crea. Al finalizar la carga en el repositorio, llama a la función que genera los archivos csv con los datos necesarios para su análisis.

f) creaindice(x)

Recibe el nombre que se le dará al índice y como su nombre lo indica, se encarga de crear el índice en el repositorio en caso de no existir previamente. La creación de dicho índice mantiene el tipo de dato que corresponde a los datos numéricos, fechas y de texto. Con ayuda del nombre, verifica si pertenece a la versión 3.2 o a la versión 3.3.

g) generaCsv(v,w,x,y,z)

Recibe cinco parámetros, el primero es la ruta en la que se van a guardar los 4 archivos csv que se van a generar (general, conceptos, traslados y retenciones), el segundo es el valor desde el cual se van a empezar a tomar los datos en el índice que está en el repositorio, el tercer parámetro corresponde al valor total de los archivos que se subieron en esa carga, el cuarto es el índice que se va a consultar y el quinto es el RFC respectivo. Luego de generar los 4 csv manda a llamar a la función que cargará los datos en la base productiva.

h) cargaStage(u,v,w,x,y,z)

Es la función que más argumentos recibe, pues son un total de seis. Los primeros cuatro parámetros corresponden a la ruta y nombre de los archivos csv que contienen los registros que van a subirse a las 4 tablas que se encuentran en la base de datos productiva (stagebf). El quinto se refiere al índice del repositorio del que sólo se utiliza el nombre para renombrar los archivos que contienen la totalidad de registros de cada tabla y el último argumento es el del RFC, que también sirve de apoyo para nombrar variables necesarias para el proceso. Esta función se conecta a la base de datos productiva y se encarga de crear las 4 tablas si es que éstas no existen. Desde luego, hace la carga de los registros y genera 4 archivos en los cuales se ve reflejado el total de los datos que

fueron cargados y de los que hay hasta el momento. Lo último sirve para comprobar que la carga fue exitosa y que no hubo datos perdidos.

i) almacenaNuevos(x,y)

Recibe dos argumentos que corresponden a una ruta de origen y una ruta de destino. Básicamente, se encarga de pasar los datos xml nuevos a un directorio en el que se especifica que son archivos ya procesados, ya que ésta función se ejecuta al finalizar todo el proceso.

Lo anterior establece hasta cierto punto el orden de ejecución de todo el ciclo de la adquisición y del almacenamiento de datos, considerando que la función que crea el índice está muy relacionada con la que carga los datos al repositorio y sólo se ejecuta en caso de ser necesaria. El resto de las funciones sí son obligatorias en todo momento.

Tabla del Proceso de Adquisición de SFTP y Descarga masiva

Propiedad	**Relación**
Nombre del proceso Python	CICLO_CLASIFICACION_CLIENTE.py
Ruta donde se encuentra el archivo .py	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI
Nombre del .sh que lo ejecuta en la noche	cliente_descarga.sh
Ruta donde se encuentra el .sh	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI/SHELL
Nombre del log que guarda la ejecución de la noche	cliente_descarga_log.log
Ruta donde se encuentra el .log	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI/SHELL/LOGS
Nombre de su archivo de propiedades	PROPIEDADES_DOC.txt
Ruta donde se encuentra el archivo de propiedades	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI

Guía de llenado de PROPIEDADES_DOC.txt	#Archivo de propiedades. Complete sin espacios host_Elasticsearch:DIR_IP_CLUSTER port_Elasticsearch:PUERTO firm:CLIENTE file_path:PATH_INICIAL para este caso XML origin_db_host:HOST_BD opcional origin_db_user:USER_BD opcional origin_db_pass:CONTRASEÑA_BD opcional origin_db_name:NOMBRE_BD opcional origin_table:TABLA_BD opcional final_db_host:HOST_STAGE final_db_user:USER_STAGE final_db_pass:CONTRASEÑA_STAGE final_db_name:NOMBRE_BD

El archivo *CICLO_XMLFROMBD_CLIENTE.py* está diseñado para cuando el origen de datos es la base de Neon, por lo que el programa debe adquirir los archivos XML desde una columna de la tabla de dicha base de datos. El código se encuentra conformado por todas las funciones que han sido mencionadas con anterioridad, sólo que la función origen() tiene algunas modificaciones y se añaden dos nuevas funciones llamadas obtieneFecha() y ejecutaQuery().

a) origen()

Establece un query que hace una consulta a la tabla de origen de Neon y llama a una función para que ejecute dicho query y traiga los CFDI correspondientes a la fecha del día anterior en el que se puso en marcha el programa. Dichos archivos los coloca en la ruta del primer RFC de la lista y después manda a llamar a la función de clasificación para que continúe todo el proceso.

b) obtieneFecha()

Obtiene la fecha del día anterior a la ejecución del programa y la retorna. La función nos permite obtener el día, el mes y el año que vamos a utilizar para realizar la consulta a la base de datos de Neon.

c) ejecutaQuery(x)

Recibe un parámetro y éste es el query que se va a ejecutar para poder obtener los registros que corresponden a los archivos CFDI, aunque antes realiza una conexión a la

base de datos de Neon con las variables de origen establecidas en el archivo de PROPIEDADES.txt.

El programa anterior debe ejecutarse todas las noches por medio del Crontab.

Tabla del Proceso de Adquisición de Base de datos Neon

Propiedad	**Relación**
Nombre del proceso Python	CICLO_XMLFROMBD_CLIENTE.py
Ruta donde se encuentra el archivo .py	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI
Nombre del .sh que lo ejecuta en la noche	cliente.sh
Ruta donde se encuentra el .sh	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI/SHELL
Nombre del log que guarda la ejecución de la noche	cliente_log.log
Ruta donde se encuentra el .log	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI/SHELL/LOGS
Nombre de su archivo de propiedades	PROPIEDADES_BD.txt
Ruta donde se encuentra el archivo de propiedades	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI
Guía de llenado de PROPIEDADES_BD.txt	#Archivo de propiedades. Complete sin espacios host_Elasticsearch:DIR_IP_CLUSTER port_Elasticsearch:PUERTO firm:CLIENTE file_path:PATH_INICIAL para este caso NEON origin_db_host:HOST_BD obligatorio y según el cliente origin_db_user:USER_BD obligatorio y según el cliente origin_db_pass:CONTRASEÑA_BD obligatorio y según el cliente origin_db_name:NOMBRE_BD obligatorio y según el cliente origin_table:TABLA_BD_EMI obligatorio y según el cliente origin_table_pdf:TABLA_BD_PDF obligatorio y según el cliente final_db_host:HOST_STAGE final_db_user:USER_STAGE final_db_pass:CONTRASEÑA_STAGE final_db_name:NOMBRE_BD

El archivo *INGESTA_METADATA_CLIENTE.py* está diseñado para cuando el tipo de información que se nos hace llegar es Metadata con formato csv. Dicho código está conformado por las funciones que se mencionan a continuación:

a) **getdate()**

   Obtiene la fecha del día anterior.

b) **obtieneDirectorio(x,y)**

   Revisa si hay información de Metadata que deba procesarse. De ser así, manda a llamar a las siguientes funciones.

c) **cargaMetadata(v,w,x,y,z)**

   Recibe parámetros como la ruta del archivo, el índice en Elasticsearch, el nombre de la tabla, el origen y el rfc del cliente. Junta los archivos encontrados de Metadata en uno solo y carga el contenedor objetivo final a la base de datos productiva. Crea la tabla destino en caso de que no exista.

d) **reemplaza(x)**

   Recibe la ruta de la carpeta en donde se encuentra el archivo final y hace unas modificaciones a la forma en la que se separan cada una de los atributos. Convierte el csv en json y lo retorna.

e) **creaindicemetadata(x)**

   Recibe el nombre que se le dará al índice y como su nombre lo indica, se encarga de crear el índice en el repositorio en caso de no existir previamente. La creación de dicho índice mantiene el tipo de dato que corresponde a los datos numéricos, fechas y de texto.

f) **upload(x,y)**

   Recibe el archivo json y el nombre del índice en Elasticsearch en el que se va a subir la información de Metadata. Si el índice no existe manda a llamar a la función que lo crea.

g) **almacenaNuevos(x,y,z)**

Recibe dos argumentos que corresponden a una ruta de origen y una ruta de destino. Además del parámetro borrar el cual de ser afirmativo borra información para ahorrar consumo de memoria, pero si dichos documentos no deben borrarse porque su origen no lo permite, entonces no lo hace Básicamente, se encarga de pasar los datos Metadata nuevos a un directorio en el que se especifica que son archivos ya procesados, ya que ésta función se ejecuta al finalizar todo el proceso.

h) borraElementos(x)

Esta función recibe un argumento que es la ruta de la carpeta de archivos consolidados. Se encarga de borrar todos aquellos archivos generados de manera temporal para que trabajen el resto de los procesos. Cuando la información llega a su destino en Elasticsearch y la base de datos productiva, dichos archivos temporales ya no son de utilidad.

El código fue trabajado en conjunto con otra persona, por lo que los nombres de las funciones pueden leerse en diferente idioma.

Tabla del Proceso Metadata

Propiedad	**Relación**
Nombre del proceso Python	INGESTA_METADATA_CLIENTE.py
Ruta donde se encuentra el archivo .py	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_METADATA
Nombre del .sh que lo ejecuta en la noche	cliente_metadata.sh
Ruta donde se encuentra el .sh	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_METADATA/SHELL
Nombre del log que guarda la ejecución de la noche	cliente_metadata_log.log
Ruta donde se encuentra el .log	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_METADATA/SHELL/LOGS
Nombre de su archivo de propiedades	PROPIEDADES_METADATA.txt
Ruta donde se encuentra el archivo de propiedades	/u01/CLIENTES/CLIENTE/PROCESOS/PROCESO_CFDI

Guía de llenado de PROPIEDADES_METADATA.txt	#Archivo de propiedades. Complete sin espacios host_Elasticsearch:DIR_IP_CLUSTER port_Elasticsearch:PUERTO firm:CLIENTE file_path:PATH_INICIAL para este caso SFTP, pero es para SAT igual final_db_host:HOST_STAGE final_db_user:USER_STAGE final_db_pass:CONTRASEÑA_STAGE final_db_name:NOMBRE_BD

Seguridad

Debido al tipo de información que se está manejando dentro del Sistema de Inteligencia Fiscal, se tomaron en cuenta varias medidas de seguridad:

- Encriptación de archivo de propiedades

Cada cliente cuenta con tres archivos de propiedades, los cuales están destinados a ser leídos por un código distinto, de tal forma que almacenan atributos de conexión a las bases de datos y al Elasticsearch, así como otros parámetros importantes. Por las razones anteriores, es indispensable que dichos archivos no queden expuestos en el servidor para que cualquier usuario los vea, por lo que se tuvo que optar por la encriptación de dichos archivos. Para ello se utiliza una librería de Python llamada Cryptography, la cual contiene el módulo fernet que se basa en un algoritmo de encriptado del tipo AES (Advanced Encryption Standard), un esquema de cifrado por bloques estándar y con el que se va a generar una clave que permitirá encriptar la información que hay dentro de los archivos de propiedades. El código en Python cuenta con varias funciones como generar la clave, encriptar el archivo, desencriptar para su uso en los códigos y se ejecuta antes de cada código principal.

- Implementación de seguridad en Elasticsearch

Para dicha implementación, se siguieron varios pasos que venían en la documentación de Elasticsearch relacionada con habilitar la seguridad en el clúster. Se tomaron en cuenta tres principales fases:

1. Configuración de seguridad del protocolo TLS (Transport Layer Security o Seguridad de la capa de transporte). Se habilita para un clúster con varios nodos. Evita que los nodos no autorizados accedan a dicho clúster. Utiliza un algoritmo de cifrado que protege la transferencia de datos e información. Se genera un certificado que permite a los nodos que lo tengan, unirse a un determinado clúster para el cual el certificado fue generado.
2. Creación de usuarios y contraseñas. Definición de roles para proteger el acceso a nivel de índice y clúster. Como su nombre lo indica, permite crear usuarios, ponerles una contraseña y determinados roles para asegurar de que sólo se pueden tener los privilegios asignados desde el principio y así evitar la vulnerabilidad de los datos en los índices.
3. Configuración de seguridad del protocolo HTTPS (HyperText Transfer Protocol Secure o Protocolo seguro de transferencia de hipertexto). Permite realizar una petición de datos y recursos. Con esta capa adicional de seguridad garantiza que todas las comunicaciones hacia y desde el clúster estén protegidas. Más que nada ayuda cuando se tiene que salir al exterior. Es sabido que los ambientes de Elasticsearch que se utilizan hoy en día, se encuentran dentro de la red de la empresa y no son visibles desde el exterior, por lo que se pudo omitir esta última fase, pero aún así, se hicieron las configuraciones pertinentes.

Una vez hecho lo anterior y ejecutado los comandos necesarios, se realizaron las siguientes acciones con el fin de comprobar que en efecto se había implementado de forma correcta la seguridad en Elasticserach:

a) Alta y baja de usuarios. Creación y asignación de roles.

b) Comprobación de si existe un número de intentos al momento de iniciar sesión o qué ocurre si el usuario se equivoca al poner la contraseña.

c) Una vez que entró con su usuario y contraseña, verificación de que pueda hacer una consulta al índice según el rol que se le definió.

Acceso (Puede entrar o no).

Privilegios (Con qué índices puede trabajar).

Roles (Qué acciones puede ejecutar con respecto a esos índices).

d) Configuración en Python cuando hay seguridad en Elasticsearch.

- Asignación de usuario y contraseña de acceso al Portal de Inteligencia Fiscal

Una vez que el cliente ha recibido los entregables correspondientes a su producto. Debe asignar a una persona para capacitación. En dicha fase, la persona asignada recibe un usuario y una contraseña para el uso exclusivo del portal. Cuando desee iniciar sesión, contará con un número limitado de intentos, en caso de equivocarse muchas veces, el inicio de sesión quedará bloqueado, por lo que deberá solicitar el desbloqueo de su usuario y contraseña, proporcionando datos que confirmen que se trata de la persona que fue asignada por el cliente.

- Asignación de usuario, contraseña y grupo de pertenencia para cada equipo involucrado en el desarrollo del Sistema de Inteligencia Fiscal

Cada equipo recibe un usuario, contraseña, se le asigna a un grupo y se le dan ciertos privilegios sobre directorios dentro del servidor, por lo que nadie puede alterar el trabajo y las carpetas de otros equipos.

- Ética profesional

Todos los trabajadores están comprometidos con la empresa y con los clientes, por lo que tienen estrictamente prohibido sacar información importante que tenga que ver con el cliente, hacer difusión de la misma, así como aprovecharla con fines maliciosos. Tampoco pueden exponer documentación realizada en horas de trabajo, ni compartir códigos, diseños, propuestas, entregables, ni nada que pueda suponer un riesgo para la empresa o para los clientes. Además, deben hacerse responsables del equipo con el que trabajan, por lo que deben haber activado el antivirus proporcionado por la empresa, asegurándose de que todos los datos estén siempre protegidos y no sufran alguna vulnerabilidad.

Buenas prácticas

Con relación a los códigos de programación:

- Variables y métodos con nombres que puedan identificarse fácilmente.
- Comentarios en el código legibles.
- Documentación de los procesos: diagramas de flujo que corresponden al diseño de la solución.
- Nomenclatura de carpetas y otros objetos que utilicen los códigos para su ejecución.
- Mantenimiento de procesos.
- Reutilización de código. Si se observa un patrón definido en el comportamiento de cierto proceso, se puede volver a utilizar lo ya codificado con el fin de ahorrar tiempo y no tener que codificar desde el inicio.
- Que el código sea lo más legible posible.
- Cuando el código comience a ser demasiado grande, ya sea porque se han añadido funciones debido a nuevos requerimientos, se recomienda comenzar a dividir el código en varios subprocesos y mandarlos a llamar como librerías.

Con relación a todo lo que está relacionado con el cliente:

- Definir una sola estructura de directorios y replicarla para cada cliente.
- Procurar que cada cliente tenga su propio espacio en el servidor. Esto implica que no se revuelva su información con la de otros.
- Determinar el nombre de los índices de cada cliente de acuerdo a la cantidad de RFCs que tiene y a la versión de los documentos.
- Almacenar los nombres de los índices en un archivo para llevar el control de los mismos.
- Evitar utilizar la información que hay en cada documento con fines distintos a los ya establecidos.

6.4 Pruebas

En la empresa se tenían dos ambientes para trabajar, uno era el ambiente productivo y otro era el ambiente de desarrollo, justo en este último era donde se nos tenía permitido hacer las pruebas de nuestros programas o procesos, por lo que la estructura de carpetas, instalación del software a utilizar y cada uno de los archivos ejecutables tenían que existir ahí antes de salir a producción. Debía confirmarse que dicho programa funcionaba correctamente antes de poder pasarlo al área productiva.

Para cada programa se realizaron pruebas unitarias que consistían en probar cada función de manera independiente para ver si funcionaba como se esperaba que lo hiciera. Generalmente, se separaba en un archivo python a parte y se utilizaban tanto documentos CFDI de prueba como índices y tablas, ya que después iba a ser necesario borrarlos para ahorrar espacio. Otras veces era común que las pruebas unitarias se realizaran de manera local si los programas a utilizar en Linux, podían utilizarse también en Windows.

Además de probar funciones de forma independiente, había ocasiones en las que se hacía uso de códigos auxiliares (véase Anexo 3) que no formaban parte del proceso como tal, pero que eran necesarios para realizar ciertas tareas como por ejemplo: pruebas de conexión a la base de datos, pruebas de conexión a Elasticsearch, transferencia de archivos de una carpeta a otra, manejo de cadenas, documentos de texto, modificación de fechas, búsqueda de documentos, inserciones a índices temporales, control de tiempos, pruebas con listas, manejo de dataframes, renombramiento de archivos, entre otros, los cuales se usaban de forma rápida para satisfacer algún requerimiento esporádico.

Luego de las pruebas unitarias, se aplicaban pruebas integrales en las que se ponía en marcha todo el ciclo por el que pasaba el proceso: desde la adquisición de los documentos fiscales hasta su llegada a las tablas que eran leídas por los tableros para

presentar la información de manera visual en la web. Estas pruebas manejaban un documento de Control de errores (véase su formato, carátula e índice en el Anexo 4).

A continuación, se describe a grandes rasgos cada uno de los errores que aparecen en el documento.

Errores en el servidor

- Contraseña de acceso a servidor incorrecta

Al momento de querer ingresar al programa Bitvise SSH Client, el cual se utiliza para hacer una conexión remota desde la máquina local al servidor de desarrollo, se debe tener un usuario y una contraseña de acceso. Si ocurre error al ingresar, es necesario verificar que se está escribiendo bien la contraseña, en caso de que se compruebe que es incorrecta, se debe solicitar ayuda a soporte técnico para que genere una nueva contraseña para dicho usuario.

- Client Key de acceso a servidor incorrecta

Al momento de querer ingresar al programa Bitvise SSH Client, el cual se utiliza para hacer una conexión remota desde la máquina local al servidor productivo, se debe contar con una Client Key o clave de acceso, la cual es un archivo encriptado que se guarda en algún directorio de la máquina local. Si ocurre error al utilizarla, es necesario solicitar el apoyo de soporte técnico.

- Usuario incorrecto

Del mismo modo que los dos escenarios anteriores, si se quiere acceder al servidor y se muestra que el usuario es incorrecto, se debe verificar que se esté escribiendo bien y si no funciona pedir ayuda a soporte técnico para que revise qué ocurre con el usuario que se ha proporcionado.

- Sin permisos necesarios para crear directorios

El usuario de desarrollador asignado para trabajar dentro del servidor productivo por lo regular tiene permisos para crear directorios en determinadas rutas del servidor, pero había ocasiones en las que, por alguna razón, se le estaba negando la manipulación de ciertos directorios, dígase la creación o eliminación de los mismos. Para solucionarlo, quien utilizaba el usuario debía enviar un correo a Infraestructura solicitando ayuda para la supervisión de los permisos y privilegios del usuario.

Errores de código

- Error de lectura de archivo

Cuando el archivo no se puede leer por algún motivo, se debe comprobar, primero que todo, que dicho archivo no esté cifrado, ya sea abriéndolo con algún editor de texto para obtener su codificación. Una vez que se tenga dicho dato, añadirlo al código. Por lo regular se trata de UTF-8 o Latin-1.

- Error de archivo de propiedades no encontrado

Cuando el archivo de propiedades no está donde debería, es indispensable ir a la ruta especificada y comprobar que se llame del modo esperado, si no se llama tal como debería, se debe renombrar, si no existe, se debe crear desde cero.

- Error de conexión a la Base de Datos

Se verifica que se tengan los datos de conexión correctos como usuario, contraseña, nombre de la base de datos, nombre de la tabla o tablas que se van a ocupar y el puerto de conexión. Por lo regular, la conexión falla cuando no se tiene el puerto de acceso configurado para la máquina local, por lo cual se debe solicitar a soporte técnico que nos apoye con dicha configuración.

- Error de directorio no encontrado

Cuando no existe un directorio, lo primero que se debe hacer es verificar que al cliente se le haya creado su estructura de directorios por completo, ya que los procesos necesitan de dichas carpetas para poder ejecutarse correctamente, dado que hay

movimiento de archivos durante el proceso de clasificación y almacenamiento. Si el directorio no existe, se debe crear.

- Error de archivo no encontrado

Ocurre cuando se espera que un archivo de procesamiento esté en una determinada ruta, pero en realidad no es así, lo primero que debe hacerse es revisar el directorio en el que se indica que se encuentra el archivo y si en efecto no está se debe hacer el reporte.

Errores en los archivos originales

Metadata

- Error de formato no válido

Por lo regular la Metadata tiene un formato estándar, ya que se extrae directamente de una petición al SAT, no obstante, hay ocasiones en las que el SAT hace ciertas actualizaciones con su formato, por lo que nuestro deber es adaptar el código a lo nuevo.

- Error de salto de línea

Muchas veces la Metadata tiene saltos de línea en sus registros, por lo que de igual forma es responsabilidad de nosotros adaptar el código para corregir dicho percance.

- Error de carga a la tabla

Generalmente ocurre porque el tipo de dato no corresponde al que está definido en la tabla, por lo que es necesario asegurarse de que el valor del atributo del registro tiene el tipo de dato esperado, en este caso la tabla es la que debe adaptarse al tipo de dato que viene en el archivo de Metadata.

CFDI

- Error de estructura no válida

Ocurre cuando el XML no es un XML válido de acuerdo a las especificaciones del SAT para los archivos CFDI, por lo que nosotros sólo podemos enviar dicho documento a una carpeta de archivos malos, ya que no se puede procesar.

- Error de versión no válida

Ocurre cuando el XML no trae la versión que va de acuerdo a las especificaciones del SAT para los archivos CFDI, la cual puede ser 3.2, 3.3 o 4.0, por lo que nosotros sólo podemos enviar dicho documento a una carpeta de archivos malos, ya que no es posible procesarlo.

- Error de codificación

Ocurre cuando el archivo no tiene la codificación esperada e impide al programa leerlo, muy similar al error de cuando el archivo no se puede leer. Se debe modificar el código para añadir una excepción y esperar documentos de dos codificaciones: UTF-8 y Latin-1.

- Error de carga a las tablas

Muy similar a lo que ocurre con la carga de Metadata, el tipo de dato no corresponde al esperado, pero además también puede ocurrir que el csv no se generó correctamente o que no existe la tabla en la base de datos, por lo que debe crearse de forma inmediata. Todo tipo de excepciones, siempre se manejan dentro del código.

Errores de visualización

- Error de acceso al portal

Cuando la URL pública del portal no es la correcta o cuando el cliente no puede acceder a su portal, esto se debe notificar para que lo revise el área encargada de esta situación.

- Error de descarga de archivos

Ocurre cuando los archivos solicitados al SAT no llegan en el momento en que deberían, como lo soluciona otro equipo, tan sólo se debe notificarles.

- Error en tableros

Cuando los tableros no se ven bien o hay un error al momento de presionar los botones o elegir filtros, esto se soluciona revisando el código y haciendo pruebas.

- Error en las cifras

Cuando las cifras no coinciden con lo que se espera de ellas al momento de hacer la compulsa, lo cual podría indicar pérdida de información, lo que se hace es volver a solicitar los documentos al SAT y procesar todo desde el principio. Se debe generar un archivo log en el que se escriba cada momento por el que pasa el CFDI, así como el conteo de archivos totales adquiridos. Si después de todo, los números siguen sin coincidir, se debe hacer un seguimiento muy particular para rastrear el documento faltante, con el objetivo de dar con el error que le impide llegar hasta el final o de comprobar que quizá se haya movido a la carpeta de archivos erróneos.

- Error en las gráficas

Ya sea que se vean extrañas o que no concuerden con las cifras mostradas. Este error se soluciona revisando el código y haciendo pruebas unitarias o integrales.

Errores de procesamiento

- Error de conexión a Elasticsearch

Si desde la máquina local no es posible conectarse al repositorio de Elasticsearch que se encuentra en alguno de los servidores, ya sea productivo o de desarrollo, es probable que se deba a la falta acceso a través del puerto, por lo que debe enviarse un correo a soporte técnico para que conceda dicho acceso.

- Error de acceso a la API de Elasticsearch

Debido a la implementación de seguridad, para acceder a la API por medio de la línea de comandos o a través de Kibana, es necesario registrarse con un usuario y con una contraseña. Se debe contar con ambos para poder conectarse y hacer consultas.

- Error de mapping

Todos los índices se crean con un mapping que corresponde a la estructura del documento json y al tipo de dato que debe tener cada atributo. Había ocasiones en las que el documento no coincidía con el mapping esperado, ya fuera porque tuviera un nodo de más o algo distinto, para ello se implementó el uso de índices dinámicos, los cuales permitían la adición de documentos con nuevos nodos. Es importante recalcar que añade el documento al índice, pero no añade su mapping.

- Error de indexación

Ocurre cuando un documento no se puede indexar, ya sea por tener un mapping distinto al esperado o porque su estructura de json no es válida. Lo que se hace es revisar a detalle el archivo json y verificar que cumpla con las reglas establecidas. Si el error se debe a detalles mínimos que se pueden controlar con código, ya sea acentos o codificación, se puede intentar corregir y volver a subir, si existe algo más allá de aquellos límites, se debe tratar como archivo erróneo.

- Error de índice no existente

Siempre que se tiene un cliente nuevo, se le debe crear índices que dependen de la cantidad de RFCs que tenga y es necesario comunicar a todos los interesados acerca de la creación de dichos índices.

- Error de documento no existente

Cuando se hace una consulta al índice, hay ocasiones en las que el documento buscado no se encuentra. Primero se debe verificar que la consulta sea correcta. En caso de que sea así, pero sigue sin aparecer, es necesario dirigirse a la carpeta final que le corresponde al cliente de cuyo índice se está haciendo la consulta y comprobar que el archivo buscado se encuentra ahí, si no está, se debe revisar la carpeta de archivos malos, una vez hallado, se debe detallar la razón por la cual el documento no se indexó como debería.

- Error de bloqueo de índice

Sucede cuando el índice de un cliente específico se bloquea. Por lo regular ocurre en el servidor productivo. Si el error es por falta de memoria, se debe implementar la liberación de la misma. Si persiste, se debe ejecutar una sentencia API para desbloquear el índice, poniendo el atributo de “Solo lectura” en Falso.

6.5 Liberación a producción

Básicamente, esta etapa consistía en migrar todos los procesos python y sus archivos correspondientes a las carpetas respectivas de cada cliente en producción y apuntar a las conexiones que se tenían en ese ambiente tanto para bases de datos como para Elasticsearch. Se daba por hecho que ya no debían ocurrir errores, pero en caso de que por algún motivo llegasen a presentarse inconvenientes (algo muy poco común), siempre debía haber manera de corregirlos al instante, por lo que se registraba dentro de un archivo log, el cual que se mandaba a crear a través de un script .sh, todo lo que sucedía con el proceso durante las noches. Dicho archivo contenía la fecha y la hora en la que el proceso python se ejecutaba, así como todo lo que iba haciendo a través del ciclo y a qué hora se terminaba de ejecutar.

6.6 Mantenimiento

Los procesos en la actualidad siguen en mejora continua, ya que al ser un sistema relativamente nuevo, cada día surgen requisitos o cambios aplicables a los archivos que se ejecutan, por lo que se me pidió que tuviese un archivo Excel de Control de versiones (véase Anexo 5), el cual estaría conformado por las siguientes columnas: Proceso, Nombre de Versión más reciente, Fecha de modificación, Tipo de modificación y Clientes

adaptados a ese nuevo cambio. Este archivo se edita conforme se van aplicando cambios a cada uno de los códigos utilizados en el flujo y sirve para saber qué proceso es el más adecuado a tomar en cuenta en caso de que un cliente nuevo solicite el módulo o sistema de Inteligencia Fiscal.

Entre los cambios más importantes que se han llegado a presentar se encuentran: corrección de errores con Elasticsearch, con el Unicode o con la base de datos, cambio de directorios o subdirectorios añadidos, manejo de archivos rar o zip, juntar funciones en una sola, cambio en la nomenclatura de las tablas y de los índices, nuevos tipos de información a procesar, cambio en el modo de procesamiento de datos, consideración de archivos pdf, ajuste de detalles para optimizar tiempos y otro tipo de mejoras necesarias.

En sus inicios se procesaban los documentos haciendo una clasificación por tipo de comprobante (pago, ingreso, egreso, nómina o traslado) el cual se aplicaba tanto a los nombres de las carpetas en las que caía la información como al nombre de los índices en Elasticsearch. Esto había provocado un aumento considerable en la cantidad de índices para el cliente y por tanto resultaba laborioso al momento de hacer consultas, ya que, al ser demasiados, se tenía que estar cambiando continuamente de nombre, en un intento de tratar de acceder a todos los índices que se tenían. Esta situación fue reportada y de inmediato se tuvieron que hacer cambios en la manera de clasificar los documentos y en la nueva forma de nombrar las carpetas y los índices. En la actualidad sólo se clasifican por versión y por RFC.

Adicionalmente, se manejan distintos tipos de situaciones de atención a clientes:

- Crítica Nivel 1 (aquéllas que no permiten la operación del sistema).
- Alta Nivel 2 (aquéllas que no permiten la operación de algún proceso o módulo del sistema).
- Media Nivel 3 (aquéllas que aunque se presentan, permiten la operación funcional de los procesos o módulos del sistema).
- No Crítica (aquellas que permiten la operación del sistema y que tengan relación con la funcionalidad de algún proceso o módulo del sistema que no sea grave).

7. Participación profesional

Para participar en el proyecto de Inteligencia Fiscal, se me asignó al departamento de Analítica y, a su vez, al Equipo técnico, el cual era, junto al Equipo de Soporte y Equipo Comercial, uno de los tres equipos principales que conformaban el desarrollo del sistema que se pretendía realizar.

Mi objetivo consistía en llevar a cabo la implementación de procesos cuya finalidad radicaba en lograr la adquisición, el almacenamiento y el procesamiento de los datos que venían en la factura electrónica del cliente en general. De forma adicional, debía realizar reportes de nómina para un cliente en específico, manejar distintos formatos y otro tipo de requerimientos que pudiesen cubrir alguna necesidad extra. Además de participar activamente en el Laboratorio de ciencia de datos en el que se realizaban distintos procesos de prueba para lograr mejorías en los indicadores estadísticos que actualmente se manejaban.

Dicho lo anterior, era necesario considerar varios factores como el origen de los datos, ya que hasta el momento se tenían los que provenían de la descarga masiva (actividad realizada por el usuario en el portal), los que se encontraban en determinadas carpetas asignadas al cliente por medio de una conexión con el protocolo SFTP y a través de la Base de datos en la que se encontraba la factura electrónica que se almacenaba ahí desde antes por el proceso de Timbrado. Asimismo, el tipo de información que iba a ser procesada (ya fuese CFDI o Metadata) y también el tipo de formato de dichos documentos (xml o csv). Además, mis procesos debían ser capaces de estar automatizados, de tal forma que se ejecutaran diariamente durante las noches, trayendo y procesando datos que fuesen adquiridos desde el día anterior.

Otro aspecto importante a mencionar era el hecho de que previo a empezar a programar cada uno de los procesos, tenía que realizar la instalación del software y de las librerías a utilizar, también tenía que ser capaz de hacer que mi proceso codificado en el lenguaje de programación Python pudiese conectarse tanto al repositorio de indexación como a

las bases de datos, además de leer archivos csv, xml o json y trabajarlos de forma adecuada. Tenía que saber realizar consultas a la API de Elasticsearch y debía tener conocimiento sobre levantar dichos servicios y cerrarlos. Era necesario que supiera sobre las herramientas que estaba utilizando e intentar dominar, en la medida de lo posible, todo aquello con lo que estuviese trabajando, ya que muchas veces otras áreas de equipos como desarrollo o soporte me consultaban para poder darles apoyo, entre sus preguntas más frecuentes se encontraban las relacionadas con Elasticsearch.

8. Resultados obtenidos

Hoy en día contamos con más de 10 clientes satisfechos con el producto y cada vez surgen nuevas contrataciones. Varios de mis procesos se ejecutan todas las noches para algunos de estos clientes, ya que depende mucho si son constantes al subir su información fiscal o de si cuentan con base de datos de facturación en nuestra empresa. Dichos procesos están siempre listos para ser implementados en clientes futuros gracias al control de versiones y a que se cuenta con manuales que forman parte de la alineación de procesos de la compañía (véase Anexo 6) y que tienen como objetivo la recopilación de la experiencia, conocimiento y técnicas utilizadas para desarrollar sus funciones y establecer el marco de actuación y lineamientos para la ejecución de las mismas.

Aunado a lo anterior, también se tienen manuales de usuario (para que el cliente tenga conocimiento de cómo usar el módulo de Inteligencia Fiscal) y manuales para desarrolladores en los que se describen situaciones relacionadas con el código, consultas a Elasticsearch, configuración de conexiones, instalación de software requerido, reglas de nomenclatura, control de migración de documentos, listado de índices productivos, conexiones por cada cliente, rutas necesarias para el buscador, listado de tablas en las bases de datos utilizadas, entre otros. De igual manera, se cuenta con archivos relacionados con diagramas de flujo, de red, de implementación, de

cambios en la estructura, de requerimientos nuevos y bitácoras de cargas, este último nos permite llevar una contabilización por mes de la información almacenada por cada cliente. Toda esta documentación es necesaria ya que representa todo el trabajo logrado hasta la fecha y se mantienen almacenados de tal forma que los interesados puedan acceder a ellos sin ningún problema.

8.1 Entregables del proyecto

- **Configuración** Entregable de definición de alcance, configuración y personalización del producto (este documento deberá ser firmado por el cliente para el inicio de la configuración y personalización).
- **Pruebas** Carta liberación de ambiente de pruebas (este documento deberá ser firmado por el usuario para la ejecución de las pruebas integrales).
- **Liberación a producción** Carta liberación de ambiente productivo.
    - ✓ Plan de capacitación.
    - ✓ Manual de usuario del Portal de emisión.
- **Capacitación** Grabación de sesión de capacitación realizada.
- **Acompañamiento** Carta liberación del proyecto y pase a soporte.
- **Portal de Inteligencia Fiscal** Visualización del portal para el cliente con usuario y contraseña. Indicadores analíticos y gráficas (véase Anexos 7, 8, 9, 10, 11, 12 y 13).

9. Conclusiones

Trabajar en esta empresa ha sido toda una experiencia y aprendizaje. He puesto a prueba mis conocimientos sobre programación y bases de datos, además de que también he aprendido a manejar otras herramientas relacionadas con el procesamiento de información. La capacitación constante, la retroalimentación de mis superiores y el que se me permita tomar cursos y certificaciones de Oracle hacen de mi experiencia algo más llevadero. Contar con objetivos bien definidos al momento de trabajar, me ha permitido estar siempre organizada, medir mis tiempos para la ejecución de cada actividad que se me va asignando y comprender cuál sería la solución a cualquier inconveniente que pudiese llegar a surgir con uno de los procesos.

El hecho de que los códigos que he realizado en mi labor hayan sido realizados por mí desde cero, me ha ayudado a conocerlos a la perfección, así que no tengo ningún problema cuando se trata de hacer mejoras o añadir otras funciones, pues día con día surgen nuevos requerimientos que siempre dependen de las necesidades actuales del cliente. El proyecto por tanto, sigue en mejora continua y nos encontramos en búsqueda de nuevas soluciones y de implementación para optimizar todavía más todo lo que hacemos. También he tenido la oportunidad de capacitar o de transmitir mis conocimientos a nuevos empleados que se han estado uniendo al proyecto de Inteligencia Fiscal, por lo que para mí representa toda una responsabilidad que las nuevas personas que llegan para apoyarnos conozcan el flujo y la manera en la que se va procesando toda la información desde que se adquiere hasta que se visualiza en cada uno de los tableros e indicadores estadísticos.

Dentro de las habilidades que he mejorado durante mi estancia en esta empresa está la de trabajar en equipo, pues es vital comunicarse con todas las áreas relacionadas con el proyecto; también he mejorado la manera en la que realizo exposiciones frente a la gente que necesita saber acerca de mis funciones, incluso he tenido la oportunidad de estar cara a cara con el cliente. Dentro de otras cosas, he aprendido a ser proactiva,

pues siempre estoy adelantándome a algo que pudiese ocurrir, aunque no se me haya pedido que lo hiciera y aquello nos ha ahorrado mucho tiempo cuando el percance sucede en realidad. Siento que también he mejorado mis habilidades en la programación y en el manejo de bases de datos, pues he realizado tareas que antes no imaginaba que podría llegar a hacer.

10. Bibliografía

Arreguin, L., Arreguin, J. (2021). *Inteligencia Fiscal*. Independently Published.

Martín, E., Caballero, R. (2020). *Las bases de Big Data*. Madrid, España: Los Libros De La Catarata.

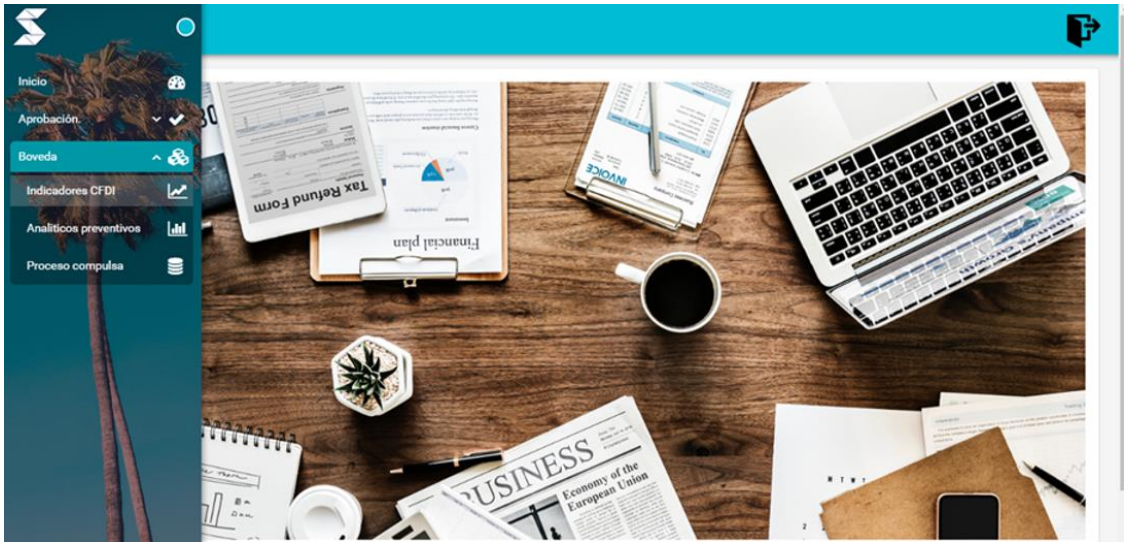
Barreix, A., Zambrano, R. (2018). Factura electrónica en América Latina. Washington, Estados Unidos: Inter-American Development Bank.

Sebe, N., Cohen, I., Garg, A., Huang, T. (2005). *Machine Learning in Computer Vision*. Dordrecht, Países Bajos: Springer Science and Business Media.

Vance, W. (2020). *Ciencia de Datos: Guía completa para principiantes para aprender los reinos de la ciencia de datos*. Joiningthedotstv.

11. Anexos

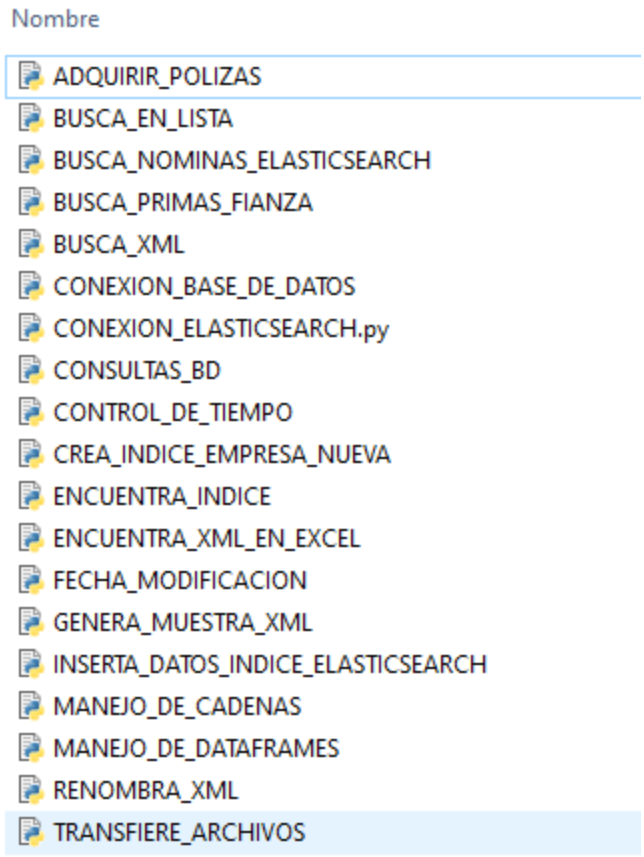
Anexo 1. Visualización del portal de Inteligencia Fiscal (antes Bóveda Fiscal)



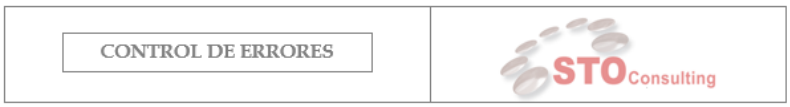
Anexo 2. Estructura de carpetas por cliente

1. home
    1.1. Analítica
        1.1.1.CLIENTE
            1.1.1.1. PROCESOS
                1.1.1.1.1. PROCESO_CFDI
                1.1.1.1.2. PROCESO_METADATA
                1.1.1.1.3. PROCESO_DWH
            1.1.1.2. ARCHIVOS
                1.1.1.2.1. RFC
                    1.1.1.2.1.1. METADATA
                        1.1.1.2.1.1.1. AÑO
                            1.1.1.2.1.1.1.1. MES
                                1.1.1.2.1.1.1.1.1. CSV
                    1.1.1.2.1.2. CFDI
                        1.1.1.2.1.2.1. AÑO
                            1.1.1.2.1.2.1.1. MES
                                1.1.1.2.1.2.1.1.1. XML
                                1.1.1.2.1.2.1.1.2. JSON
                                1.1.1.2.1.2.1.1.3. CSV

Anexo 3. Códigos auxiliares



Anexo 4. Visualización de carátula e índice de documento de Control de errores



DOCUMENTO DE CONTROL DE ERRORES

Autor:	Servicios, Tecnología y Organización S.A. de C.V.
Fecha de creación:	09/06/2020
Última Actualización	12/01/2022
Versión:	1.0

Nombre	Puesto	Fecha
Ana Hernández Galindo	Consultor BI Jr.	09-06-2020

CONTROL DE ERRORES	STO Consulting

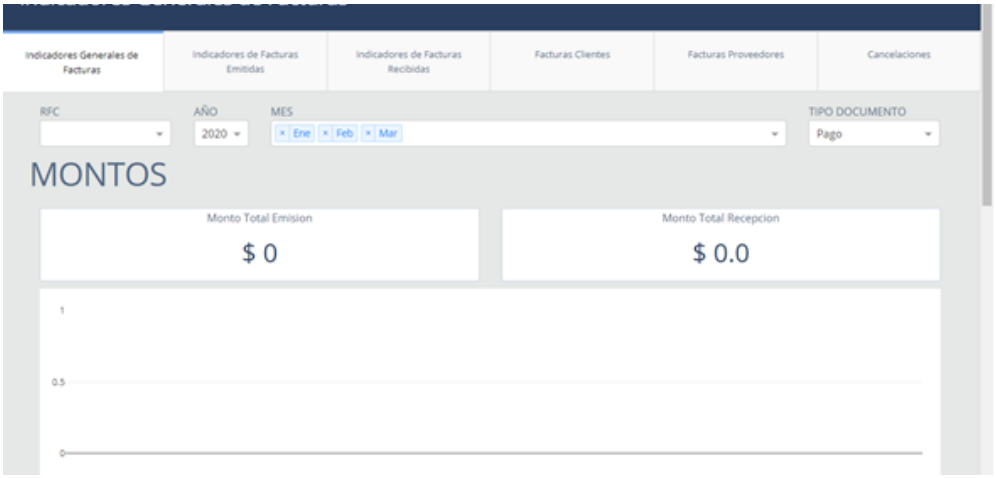
CONTENIDO

Anexo 5. Ejemplo de documento de Control de versiones

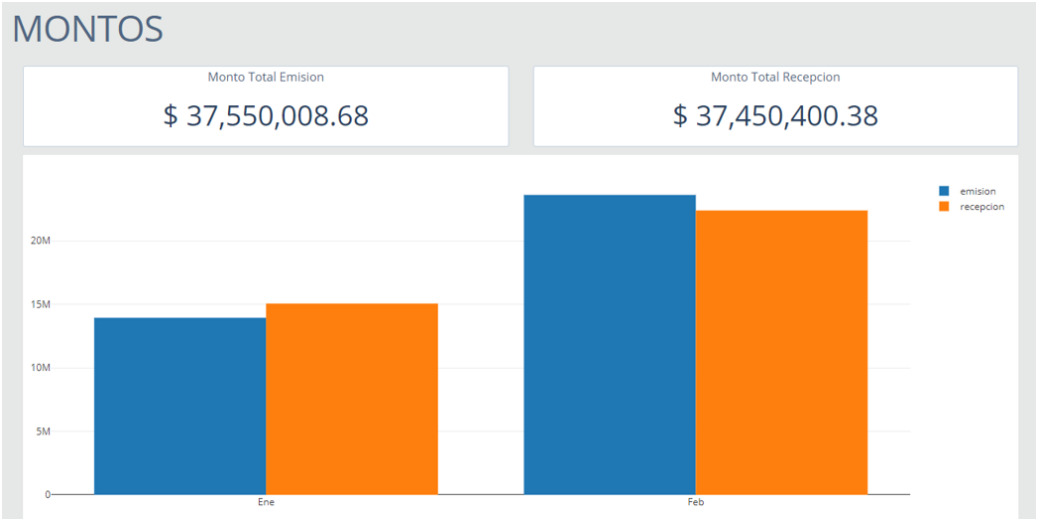
	A	B	C	D	E
1	**PROCESO**	**NOMBRE VERSIÓN MÁS RECIENTE**	**FECHA DE MODIFICACIÓN**	**TIPO DE MODIFICACIÓN**	**CLIENTES ADAPTADOS**
2	CICLO_CLASIFICACION_CLIENTE.py	CICLO_CLASIFICACION_CLIENTE_1.py	9 de febrero 2021	Apunta a nueva estructura de directorios	TODOS
3	DWH_CLIENTE_SQL_SCRIPTS_PROD.py	DWH_CLIENTE_3_SQL_SCRIPTS_PROD.py	13 de febrero 2021	Condicional para que ciertas sentencias se ejecuten 1 vez	CLIENTE_3
4	ING_MET_STAGE_ELASTIC_CLIENTE.py	ING_MET_STAGE_ELASTIC_CLIENTE_4.py	15 de febrero 2021	Listado correcto después de borrar repetidos	TODOS
5	CICLO_NOMINA_CLIENTE.py	CICLO_NOMINA_CLIENTE_2.py	18 de febrero 2021	Generación de 4 tablas con atributos de nómina	CLIENTE_2
6	GENERA_NUEVO_JSON_MOD.py	GENERA_NUEVO_JSON_MOD.py	12 de marzo 2021	Carga a los índices todas las relaciones Pag_Recibos_Fianzas	TODOS
7	CICLO_XMLFROMBD_CLIENTE.py	CICLO_XMLFROMBD_CLIENTE_5.py	24 de marzo 2021	Se añadió la extracción de datos de estacionamiento	CLIENTE_5
8	CICLO_PAGOS_CLIENTE.py	CICLO_PAGOS_CLIENTE_2.py	14 de abril 2021	Llena las tablas de pagos en stage para cada rfc	CLIENTE_2
9	CICLO_RETENCIONES_CLIENTE.py	CICLO_RETENCIONES_CLIENTE_2.py	11 de mayo 2021	Llena una tabla con todas las retenciones del índice	CLIENTE_2
10	CICLO_RELACIONA_DOCUMENTOS.py	CICLO_RELACIONA_DOCUMENTOS.py	2 de junio 2021	Ajuste de código para optimización	CLIENTE_2
11	ACTUALIZA_TABLA_CONTROL_CLIENTE.p	ACTUALIZA_TABLA_CONTROL_CLIENTE_4.p	25 de junio 2021	Según el estatus, actualiza la tabla de adquisicion	CLIENTE_4
12					

Anexo 6. Manuales de alineación de procesos de la empresa

Anexo 7. Visualización de indicadores al entrar al sistema (sin filtro RFC)



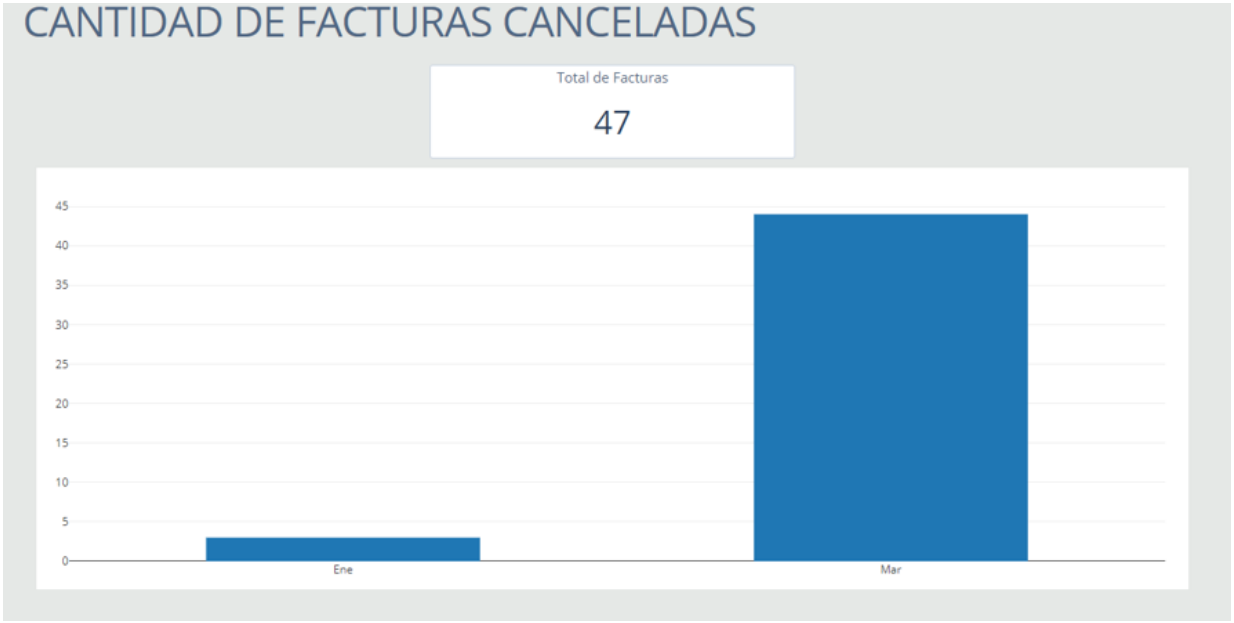
Anexo 8. Indicadores de montos generales con gráfica de barras (luego de seleccionar filtros)



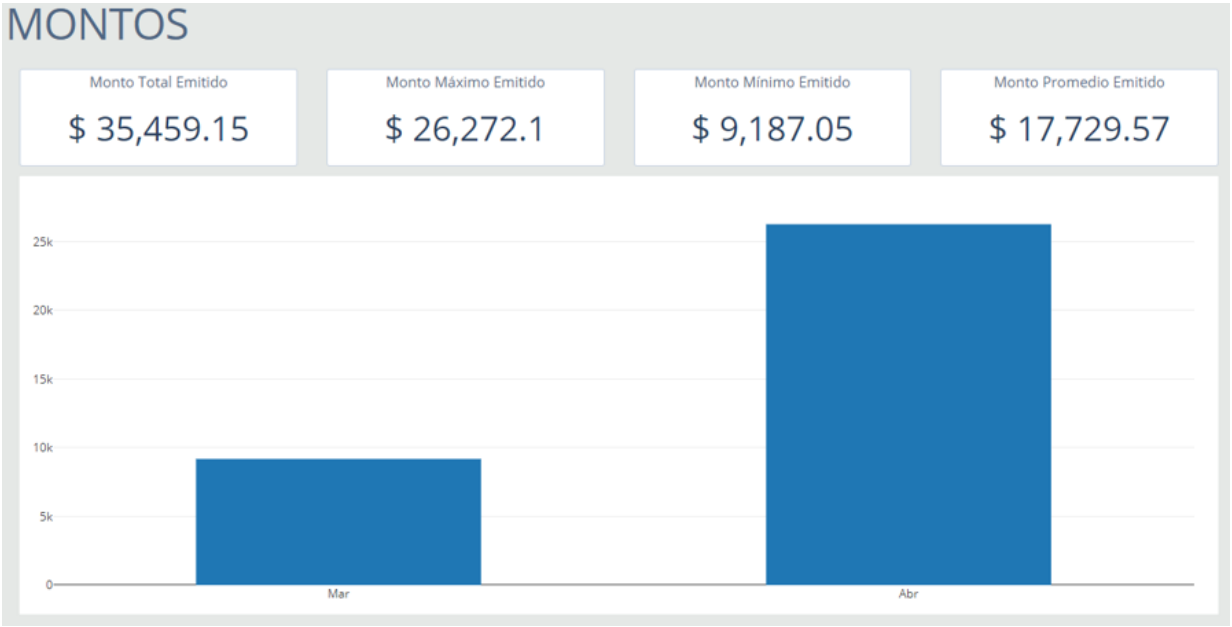
Anexo 9. Indicadores de montos a detalle con gráfica lineal (luego de seleccionar filtros)

Anexo 10. Indicadores de montos de cancelaciones con gráfica de barras (luego de seleccionar filtros)

Anexo 11. Indicador de cantidad para facturas canceladas (luego de seleccionar filtros)



Anexo 12. Indicadores de montos por emisión (luego de seleccionar filtros)



Anexo 13. Indicadores de montos por recepción (luego de seleccionar filtros)