UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

DISEÑO Y EJECUCIÓN DE ARQUITECTURA DE DESCARGA, MODELAMIENTO Y ANÁLISIS DE DATOS PARA AMPLIAR SERVICIOS EN UNA EMPRESA DE TECNOLOGÍA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

DANIEL ESTEBAN ESPINOZA PICCOLIS

PROFESOR GUÍA
JUAN ROMERO GODOY

MIEMBROS DE LA COMISIÓN

ALEJANDRA PUENTE CHANDÍA

FELIPE VILDOSO CASTILLO

SANTIAGO DE CHILE
2020

Resumen de la memoria para optar al título de: Ingeniero Civil Industrial
Por: Daniel Espinoza Piccolis
Año: 2020
Profesor guía: Juan Pablo Romero

Diseño y ejecución de arquitectura de descarga, modelamiento y análisis de datos para ampliar servicios en una empresa de Tecnología

El propósito de este proyecto es diseñar e implementar una arquitectura de descarga, la cual contendrá dentro de sus capas un modelo de datos que optimice el trabajo con ellos, para el posterior análisis de los datos. La finalidad es hacer que una empresa de Tecnología, que se desenvuelve en el negocio de servicio de emisión de boletas electrónicas, posea un nuevo servicio hacia sus clientes, en conjunto con poseer una base de datos actualizada diariamente para lanzar nuevos productos hacia sus clientes en el futuro, mejorando así su propuesta de valor.

Para ello se diseñó e implementó una arquitectura de descarga de cinco capas. La primera capa corresponde a la fuente de datos, siendo, es este caso, la plataforma de Servicios de Impuestos Internos (SII). Por medio de web scraping se cuantificó la cantidad de archivos disponibles para descargar dado un horizonte fijo de tiempo. La segunda corresponde a la extracción, transformación y carga de datos, en donde se utilizó web scraping por medio de Selenium para recaudar la información luego de diseñar un algoritmo de descarga; la información fue transformada en diccionarios y cargada a una base de datos en PostgreSQL. La tercera capa consistió en el modelo de datos para que el acceso a la información fuese posible por medio de consultas óptimas. Se eligió un modelo tipo estrella dado la naturaleza de los datos recolectados y se crearon tablas destinadas para el registro del proceso de descarga. La cuarta etapa consistió en el análisis de datos, en donde se procedió a realizar una plantilla de un dashboard por medio de Power BI, en donde, tras ingresar un datamart de un cliente de un año, entrega un análisis general y específico en base a su información. En este, el cliente puede filtrar por tipo y origen de documento tributario, razón social de sus clientes, ver los productos más vendidos según sus facturas electrónicas, sus clientes más importantes, entre otras cosas. La quinta capa corresponde a la propuesta de servicio, siendo, es este caso, decisiones que los clientes pueden tomar una vez tenga acceso al dashboard de la cuarta capa.

Terminado el trabajo de título, se le entregó a la empresa un proceso automatizado capaz de extraer los documentos electrónicos de sus clientes, alimentar un modelo de datos estandarizado y optimizado y construir una serie de reportes que dan visibilidad a indicadores de gestión. Esto permite a la empresa ampliar sus servicios y mejorar su oferta de valor.

Tabla de contenido

Lista de tablas

Lista de figuras

1. Introducción

1.1. Antecedentes de la empresa

La empresa para la cual se desarrolla este proyecto fue fundada el año 2014 en la ciudad de Santiago de Chile. Se desempeña en el giro de servicios integrales en tecnología e información, y cuenta con más de veinte trabajadores. Su negocio es proveer el servicio de crear boletas electrónicas por medio de un dispositivo POS.

La Máquina POS (Point Of Sale) es un dispositivo de tipo electrónico con una pantalla y un teclado. Se trata de una tecnología que se adapta a los pagos a través de tarjetas de débito y de crédito. Por otra parte, permite realizar toda una serie de funciones de gran utilidad comercial: adaptación a los planes de fidelización que realizan los bancos, lectura de códigos de barra, interacción con el stock de mercancías del comercio, impresión de recibos, datos estadísticos, etc. Estos terminales tienen un software muy parecido a un PC y, sobre todo, se adaptan a cada tipo de comercio. (Definición, 2019)

Figura 1: Ejemplo de dispositivo POS. Xeumior SM-V2.

Este servicio cuenta con la ventaja de estar conectado directamente con el SII, por lo que la empresa cliente puede desprenderse de realizar trámites tributarios relacionados con la emisión de boleta, tales como llevar libro de compra y venta, timbrar boletas, entre otros.

En el caso base, donde una empresa no tiene contratado el servicio, este debe emitir de manera manual boletas para cada venta y presentarlas al SII, además de hacer un libro de compra y libro de venta. Un problema durante la realización de este proceso podría significar una multa hacia la empresa o una clausura del negocio.

En el caso en que esta empresa se convierta en cliente, ahora sólo debe emitir sus boletas por medio del dispositivo facilitado. Luego es la empresa que provee el servicio la encargada de hacer todos los trámites pertinentes en el SII. Para poder hacerlo, la empresa cliente debe pasar por un proceso de certificación por parte de la empresa que provee el servicio. Esto trae como resultado que la empresa proveedora del servicio queda registrada como usuario autorizado por parte de la empresa cliente en el SII.

1.2. Justificación

La empresa no es la única participante en Chile dentro del negocio de boleta electrónica. En promedio tardan alrededor de seis meses en recuperar la inversión inicial necesaria para captar a un cliente nuevo, por lo que no pueden darse el lujo de competir con precio; deben hacerlo con la propuesta de valor que puedan ofrecer a sus clientes.

Dado que la empresa es un usuario autorizado por partes de sus clientes para ingresar al SII, en conjunto a que el cliente da permiso en el contrato para que la empresa pueda trabajar con sus datos con finalidades de marketing, de la propia prestación de servicios y/o con el objeto de entregar información y/o beneficios, esta puede acceder a la información correspondiente a los documentos tributarios de sus empresas clientes.

Siendo esta la situación, se desea abordar la oportunidad de crear una base de datos con la información de los documentos tributarios de las empresas clientes para entregar, en primera instancia, un dashboard personalizado a las empresas clientes de manera de mejorar el servicio a entregar. No obstante, al ser una base de dato lo que se está creando, la empresa podrá desarrollar nuevos servicios en el futuro, al igual que análisis de los datos.

1.3. Objetivos

Diseñar e implementar una arquitectura de descarga para una PYME capaz de crear una base de datos que sea actualizada cada día y que sea escalable para su posterior explotación y así sugerir nuevos servicios a la empresa con el fin de mejorar la propuesta de valor hacia los clientes.

Para llevar esto a cabo, el proyecto cumplirá con las siguientes fases:

- Estudiar y analizar herramientas disponibles para proyecto.
- Diseñar e implementar una arquitectura de descarga.
- Diseñar e implementar modelo de datos para acceder a información.
- Entregar mockup de nuevo servicio para clientes.

1.4. Metodología

Para poder realizar este proyecto, en primera instancia se realizará el proceso deseado de manera manual para estar familiarizado con el proceso que se desea automatizar. Se verá la estructura de los documentos tributarios a descargar y se identificará el segmento de información de interés.

Una vez teniendo claro el proceso, se procederá a pedir permiso a la empresa para poder utilizar sus herramientas ya contratadas. En este caso corresponde a la creación de un usuario en Amazon Web Services, dar las llaves necesarias para acceder a sus bases de datos y otorgar el certificado digital del representante legal de la empresa junto a su contraseña para poder realizar el web scraping.

Posterior a ello, se hará un estudio del estado del arte para poder familiarizarse con herramientas ya contratadas por la empresa, al igual que herramientas ajenas para evaluar si es justificable contratarla. Estas herramientas abarcarán tanto el almacenamiento de datos como el registro y análisis de estos; y se prestará atención a su eficiencia para un volumen elevado de datos.

Una vez estudiado las herramientas para carga y análisis de datos, se prestará atención a herramientas de web scraping, específicamente qué navegador web presenta una mayor atención en foros de programación, y cuál interactúa de mejor manera con certificados digitales (un factor sumamente importante para realizar extracción de datos en la página web de interés).

Se verán herramientas para poder trabajar en un servidor remoto, de manera que el programa esté en una plataforma estable y confiable.

Finalmente se verán modelo de dato que permitan una manera óptima de analizar los datos, al igual que herramientas que permitan hacer dashboard con la información recolectada.

Ya habiendo terminado con el estado del arte, se diseñará cada una de las capas de la arquitectura de descarga desde su base, además de diseñar al ambiente remoto en donde se ejecutará el producto final. Esto se hará en base a las herramientas estudiadas y de la característica de los datos a recolectar.

De igual manera, la implementación de la arquitectura se hará por capas, partiendo desde la base. Antes de comenzar a programar, se terminará de preparar el entorno de trabajo, el cual ya fue en parte tratado antes de investigar el estado del arte, al instalar programas y dar permisos a herramientas necesarias. Esto involucra tanto el ambiente local como el ambiente remoto.

Las pruebas se irán realizando progresivamente el código sea desarrollado, asegurándose que antes de pasar a una capa nueva, la anterior funcione correctamente. Se trabajará siempre con los mismos datos para extraer durante la implementación, de manera de tener resultados consistentes y comparables. Acorde se vayan implementando más capas, se podrá ir comparando los tiempos de descarga y carga a lo largo del proyecto. En base a los resultados, se irá evaluando cambios en el código que puedan hacerlo más eficaz.

Una vez lograda una arquitectura de descarga funcional y eficiente, se creará una base de datos acotada, orientada para hacer dashboards a clientes en específico para así poder desarrollar un dashboard personalizado para ellos.

En base a los dashboards creado, en conjunto con la arquitectura de descarga y los tiempos asociados a sus etapas, se procederá a realizar las conclusiones y a dar futuros pasos a seguir para la empresa.

1.5. Estructura del Documento

En este apartado se presentará la estructura del documento con el fin de entregar una visión general de este y facilitar su lectura y seguimiento.

1. Introducción: Se detalla el contexto en donde se desarrolla el proyecto, así como la justificación, los objetivos y la estructura de este.
2. Estado del arte: Se analizan las herramientas disponibles para distintos apartados necesarios para el proyecto, tales como herramientas de Amazon Web Services (AWS) y herramientas para realizar web scraping.
3. Diseño: Se describe el proceso de diseño del proceso, se explica las herramientas seleccionadas y se explica el porqué de estas elecciones.
4. Implementación: Se describe el proceso de implementación del proyecto, en donde se muestran los pasos necesarios y resultados preliminares de distintas capas de la arquitectura
5. Pruebas y resultados: Se exponen resultados obtenidos con la arquitectura completada, tanto en el proceso de descarga como en el análisis de los datos obtenidos.
6. Conclusiones: Se realiza una retrospectiva del trabajo realizado y se esbozan posibles líneas de trabajo a futuro para este proyecto.

2. Estado del arte

Antes de comenzar con el proyecto, se estudiarán las herramientas disponibles con el fin de encontrar la mejor solución en diferentes apartados para el desarrollo de este. Debido a que la empresa en la que se desarrollará el proyecto trabaja en gran parte con herramientas de Amazon Web Services, se buscará que las herramientas a utilizar formen parte de estas, aunque aquello no es un factor netamente determinante.

Sumado a esto, se buscará explicar nociones necesarias para el entendimiento del proyecto, tal como explicar la estructura de documentos tributarios y cómo acceder a su información.

2.1. PostgreSQL

PostgreSQL es un manejador de base de datos relacionales, completamente open source, el cual se puede utilizar en cualquier sistema operativo. (Código Facilito, 2019) Se puede trabajar con hasta 64 TB de datos y, para un volumen bajo de datos, no presenta diferencia de desempeño frente a MySQL. (AWS, 2019)

Al ser una herramienta de código abierto, es ampliamente utilizada alrededor del mundo, por lo que es fácil encontrar información sobre su uso en distintos casos en foros de programación.

2.2. Amazon Web Services

Amazon Web Services (AWS) es una colección de servicios de computación en la nube pública, también llamados servicios web, que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de internet por Amazon. (Wikipedia, 2019)

La empresa posee varios centros de datos esparcidos en puntos estratégicos a lo largo del mundo, generando una red de internet secundaria completamente controlada por Amazon. En el año 2019, la nube de AWS incluye 69 zonas de disponibilidad en 22 regiones geográficas alrededor del mundo. (Amazon AWS, 2019)

Figura 2: Mapa de la infraestructura global de AWS. (Amazon AWS, 2019)

La finalidad de este sistema consiste en diversificar sus recursos de manera que, en caso de poseer problemas con un servidor, ya sea por un problema directo del servidor o un tráfico muy elevado de datos durante un periodo, exista otra alternativa en un rango adecuado para poder reemplazarla mientas se soluciona el problema. Esta flexibilidad ha llevado a AWS a ser uno de los participantes más importantes del internet. (Brandom, 2019)

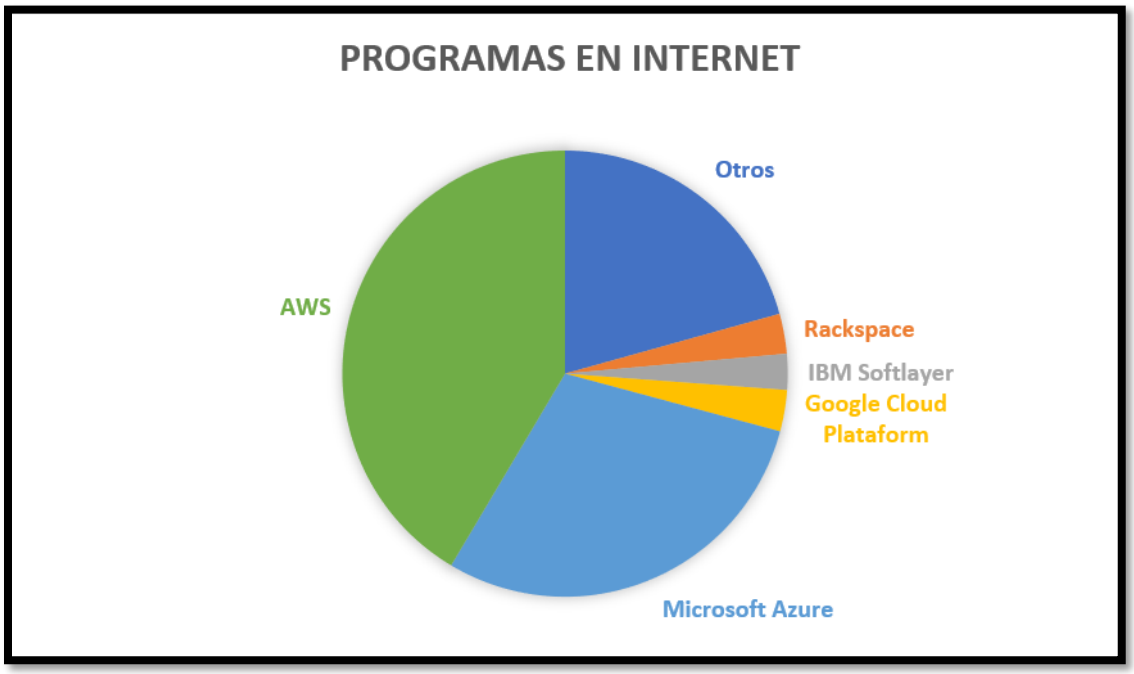

Figura 3: Programas en internet según servidores.

Durante los años noventa, el equipo web de Amazon estaba centrado en hacer que la carga de la página amazon.com fuera lo más rápido posible y que esta no se cayera. Su éxito fue tal que a comienzo de los años dos mil comenzaron a vender el sistema a otras compañías. A partir de entonces, si una compañía deseaba introducirse al e-commerce, Amazon podía construirles una tienda online usando la misma tecnología. (Brandom, 2019)

AWS fue oficialmente lanzado el 2006 sólo para almacenar datos, meses después se lanzó la habilidad de poder correr programas de manera remota, y, a partir de entonces, no hizo más que agregar más características. De esta manera llamó la atención de empresas como Netflix, Airbnb, Pinterest, Glovo, entre otras.

2.2.1. S3

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento líderes en el sector. Esto significa que clientes de todos los tamaños y sectores pueden utilizarlo para almacenar y proteger cualquier cantidad de datos para diversos casos de uso, como sitios web, aplicaciones móviles, procesos de copia de seguridad y restauración, operaciones de archivado, aplicaciones empresariales, dispositivos IoT y análisis de big data. Amazon S3 proporciona características de administración fáciles de utilizar que le permiten organizar los datos y configurar sofisticados controles de acceso con objeto de satisfacer sus requisitos empresariales, organizativos y de conformidad. Amazon S3 está diseñado para ofrecer una durabilidad del 99,999999999% por medio de 11 nubes, y almacena datos de millones de aplicaciones para empresas de todo el mundo. (Amazon AWS, 2019)

Además de tener buena sincronía con herramientas de AWS, también existen aplicaciones de terceros que pueden conectarse a S3, tales como Hadoop, un proyecto de software de código abierto utilizado para Big Data. Es una herramienta de bajo coste y escalable a largo plazo.

2.2.2. DynamoDB

Amazon DynamoDB es una base de datos de clave-valor y documentos que ofrece rendimiento en milisegundos de un solo dígito a cualquier escala. Se trata de una base de datos duradera de varias regiones y con varios maestros, completamente administrada, que cuenta con copia de seguridad, restauración y seguridad integradas, y almacenamiento de caché en memoria para aplicaciones a escala de Internet. DynamoDB puede gestionar más de 10 billones de solicitudes por día y puede admitir picos de más de 20 millones de solicitudes por segundo. (Amazon AWS, 2019)

Presenta la desventaja de que no posee un lenguaje de consulta estructurado, se debe trabajar con una estructura llave-valor en formato JSON.

2.2.3. Athena

Amazon Athena es un servicio de consulta interactivo que facilita el análisis de datos en Amazon S3 utilizando SQL estándar. Athena no tiene servidor, por lo que no hay

infraestructura para administrar y solo paga por las consultas que ejecuta. (Amazon AWS, 2019)

Athena puede leer archivos de diferente formato en S3, tales como archivos de texto, CSV, JSON, ORC, Parquet, entre otros. La decisión de qué formato utilizar puede afectar el desempeño de las consultas, siendo específicamente ORC y Parquet dos formatos que ayudan a Athena a correr más rápido. Además de ello, hay otras prácticas que uno puede utilizar para optimizar las consultas, tales como particionar los datos, comprimir datos, optimizar el tamaño de datos, entre otros. (Hocanin, 2017)

La herramienta fue lanzada al mercado a finales del 2016, por lo que, en comparación con otras alternativas, no hay tanta información disponible por ser relativamente nueva.

2.2.4. Redshift

Amazon Redshift es un servicio de almacenamiento de datos en la nube completamente administrado a escala de petabytes[1]. Se puede comenzar desde unos cientos de gigabytes de datos y, luego, ampliarlos a un petabyte o más. (Amazon AWS, 2019) Actualmente, Amazon RedShift es el almacén de datos en la nube más rápido del mundo y su rapidez aumenta radicalmente año tras año. Además, los costos de uso son los más bajos de todos los almacenes de datos en la nube. (Amazon AWS, 2019)

2.2.5. EC2

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable. Está diseñado para simplificar el uso de la informática en la nube a escala web para los desarrolladores. (Amazon AWS, 2019)

En otras palabras, es una herramienta que permite trabajar con máquinas remotas. Esto puede presentar beneficios en contraste a una máquina local, todo dependiendo del proyecto que se lleve a cabo.

Existen varios planes disponibles a contratar según la necesidad del proyecto, los cuales varían en las funciones y capacidades disponibles del sistema, tales como la cantidad de núcleos del sistema, el tipo de procesador de la máquina, la memoria, entre otros.

2.2.6. Aurora PostgreSQL

Amazon Aurora PostgreSQL es un motor de base de datos relacional completamente administrado, compatible con PostgreSQL, que combina la velocidad y la fiabilidad de las bases de datos comerciales de tecnología avanzada con la sencillez y la rentabilidad de las bases de datos de código abierto. Aurora PostgreSQL es un reemplazo instantáneo para PostgreSQL que simplifica y hace más rentable

[1] Un petabyte corresponde a 10^{15} bytes.

configurar, usar y escalar las implementaciones de PostgreSQL nuevas y ya existentes. (Amazon AWS, 2019)

En consecuencia, se tiene un motor de base de datos relacional con una fuente ampliamente utilizado en el mundo, haciendo que sea fácil acceder a información pertinente a su uso para distintos casos por medio de foros de programación.

2.3. Selenium

Las vistas de las páginas web que uno visita están escritos en HTML, un lenguaje ampliamente utilizado en el mundo para este tipo de problemas que funciona en base a etiquetas. Esto hace que cada elemento dentro de una página web adquiera una dirección única en la página, llamada xpath.

Una técnica para extraer datos de páginas web es el web scraping, un proceso automático donde una aplicación procesa el HTML de una página web para obtener información. Selenium es una librería disponible en Python que permite automatizar procesos webs al ir seleccionando elementos de la página, como, por ejemplo, xpaths. La librería es capaz de simular la interacción de una persona, logrando cosas como presionar elementos en la página, enviar datos en un formulario, navegar en internet, trabajar con certificados digitales, entre otros.

Para que esto sea posible, la librería trabaja en conjunto con un archivo específico necesario que hace posible desplegar una ventana del navegador para que el código pueda interactuar con ella. Este no necesita ser instalado, se accede al archivo directamente desde el código. De no existir el archivo, o de ser de una versión desactualizada en comparación con la versión del navegador a seleccionar, el web scraping no podrá desarrollarse de manera correcta[2].

Navegador	**Driver**
Mozilla Firefox	geckodriver.exe
Google Chrome	chromedriver.exe
Opera	operadriver.exe
PhantomJS	phantomjs.exe

Tabla 1: Archivos necesarios para distintos navegadores web en Windows.

A continuación, se muestran algunos navegadores web con sus características asociadas al web scraping.

2.3.1. Google Chrome

Google Chrome es un navegador web de software privativo o código cerrado desarrollado por Google, aunque derivado de proyectos de código abierto. Está disponible gratuitamente.

[2] Si el archivo necesario no tiene la misma versión que el navegador correspondiente, la ventana podría tanto o no desplegarse, o el código podría no identificar elementos dentro de la página para interactuar.

Cuenta con más de 750 millones de usuarios. Actualmente el número de usuarios aumentó considerablemente situándose en una cuota de mercado del 54 % convirtiéndolo en el navegador más utilizado de todo el planeta. (Wikipedia, 2019)

En virtud de este último dato, es el navegador que presenta mayor documentación en foros de web scraping, por lo que es más fácil encontrar respuestas a problemas relacionados.

2.3.2. Mozilla Firefox

Mozilla Firefox (llamado simplemente Firefox) es un navegador web libre y de código abierto desarrollado para Linux, Android, iOS, macOS y Microsoft Windows coordinado por la Corporación Mozilla y la Fundación Mozilla. Usa el motor Gecko para renderizar páginas web, el cual implementa actuales y futuros estándares web. (Wikipedia, 2019)

Al realizar web scraping, presenta la ventaja de que tiene facilidad para trabajar con perfiles de usuarios, por lo que se puede recordar decisiones tomadas previamente.

2.3.3. PhantomJS

PhantomJS es un navegador descatalogado que se utiliza para automatizar la interacción de la página web. PhantomJS proporciona una API de JavaScript que permite la navegación automatizada, capturas de pantalla, comportamiento del usuario y afirmaciones, lo que la convierte en una herramienta común utilizada para ejecutar pruebas unitarias basadas en navegador en un sistema sin cabeza como un entorno de integración continua. (Wikipedia, 2019)

Dentro del proyecto, tiene la ventaja que presenta facilidad para cargar un certificado digital al realizar web scraping, pero no permite descargar archivos de manera nativa.

2.4. Documentos tributarios

Existen distintos tipos de documentos tributarios emitidos en formato electrónico que se manejan en el país, tales como facturas electrónicas, notas de débito, notas de crédito, guía de despacho, entre otras. Estos poseen información respecto al negocio de sus participantes, ya sea como el detalle del producto transado, al igual que la cantidad y monto asociado.

Ahora, más importante que hablar sobre en qué consiste cada documento, es importante hablar de la estructura que estos poseen, lo cual será importante más adelante. Para comenzar, los documentos tributarios tienen una estructura bastante definida por parte del SII, como se puede ver en la Figura 4. Esta es una plantilla por parte del SII que muestra la base de una factura electrónica, la cual no difiere mucho con otros documentos tributarios.

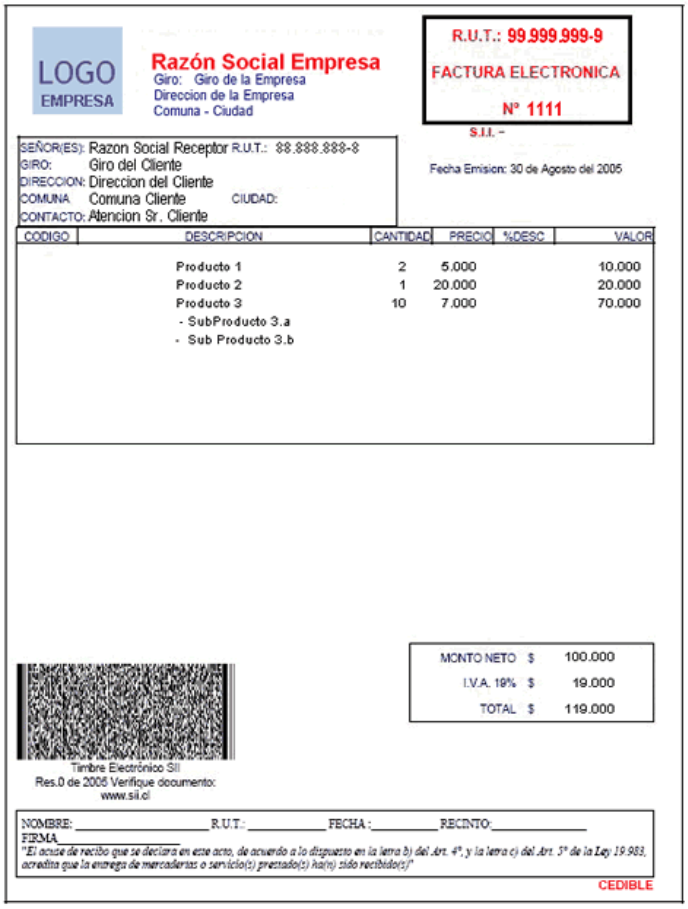

Figura 4: Factura electrónica MiPyme. (SII, 2019)

Si bien esta figura nos ayuda a tener una idea conceptual de un documento tributario, no nos aporta mucha información en cuanto a la información tangible con la cual se trabajará. Los documentos tributarios son descargados del SII en formato XML, un lenguaje de etiqueta similar a HTML.

Figura 5: Extracto de XML de factura electrónica.

En la figura anterior se ve parte del XML de una factura electrónica, en donde se pueden identificar dos partes que son de nuestro interés: el encabezado y el detalle. El encabezado entrega la información de los participantes de la transacción, es decir, del emisor y del receptor, además de mostrar información del documento en sí, tales como el tipo de documento y el folio. La segunda parte corresponde al detalle, el cual adjunta la información específica de los productos transados. Si bien, en el ejemplo mostrado, el detalle es único, este no es siempre el caso; un documento tributario puede tener múltiples detalles, tal como se puede ver en la Figura 4. Es desde estas partes específicas del XML en que podemos encontrar información de interés para hacer una base de datos.

2.5. VNC

VNC (Virtual Network Computing) es un programa de software libre basado en una estructura cliente-servidor que permite observar las acciones del ordenador servidor remotamente a través de un ordenador cliente. VNC no impone restricciones en el

sistema operativo del ordenador servidor con respecto al del cliente: es posible compartir la pantalla de una máquina con cualquier sistema operativo que admita VNC conectándose desde otro ordenador o dispositivo que disponga de un cliente VNC portado. (Wikipedia, 2019)

2.6. Power BI

Power BI es un servicio de Windows que facilita la visualización de información de manera detallada con el fin de poder tomar decisiones de manera rápida e informada. Es multiplataforma y se puede sincronizar los dashboards a crear en la cuenta del usuario. Si bien cuenta con una version gratuita, varias herramientas están reservadas para la suscripción pagada del servicio.

Se puede importar una base de datos por medio de un documento de Excel, un CSV, un servidor SQL, servicios de análisis, entre otros. Entrega la facilidad de hacer tablas dinámicas y distintos tipos de gráficos, además de controlar quiénes pueden editar y quiénes pueden sólo observar los datos.

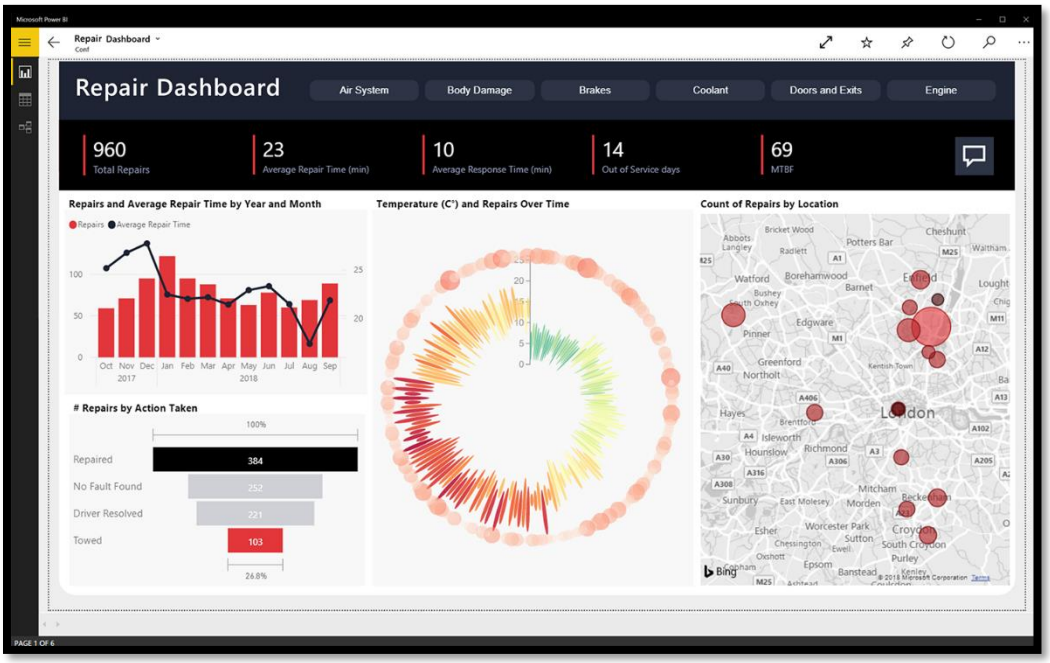

Figura 6: Ejemplo de dashboard en Power BI. (Microsoft, 2019)

2.6. Conexión a servidor remoto

2.6.1. WinSCP

WinSCP es una herramienta para SFTP (SSH File Transfer Protocol), lo que quiere decir que permite trabajar de manera local un servidor remoto. Permite ejecutar comandos en la consola del servidor remoto, al igual que trabajar con sus archivos. No permite una visualización de la interfaz gráfica, sino que sólo se pueden ver en un explorador de archivos.

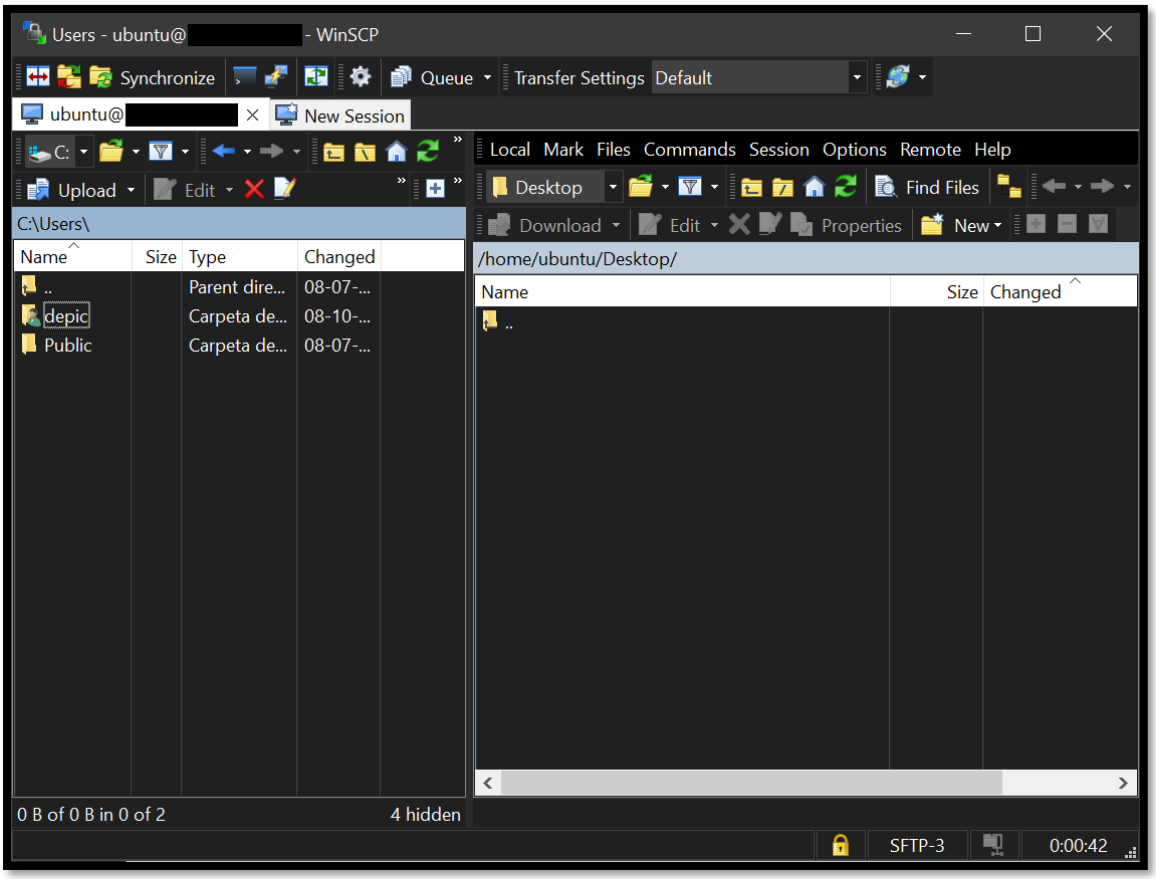

Figura 7: Interfaz de WinSCP.

2.6.2. Cygwin

Cygwin es una colección de herramientas desarrollada por Cygnus Solutions para proporcionar un comportamiento similar a los sistemas Unix en Microsoft Windows. Su objetivo es portar software que ejecuta en sistemas POSIX a Windows con una recopilación a partir de sus fuentes. (Wikipedia, 2019)

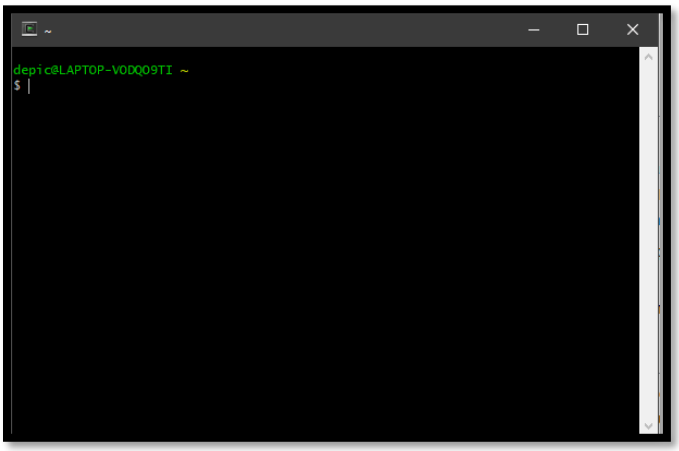

Figura 8: Terminal de Cygwin.

2.6.3. VNC Viewer

VNC Viewer es una herramienta de RealVNC, la cual permite acceder de manera gráfica a un servidor remoto, siempre que se tengan los permisos necesarios.

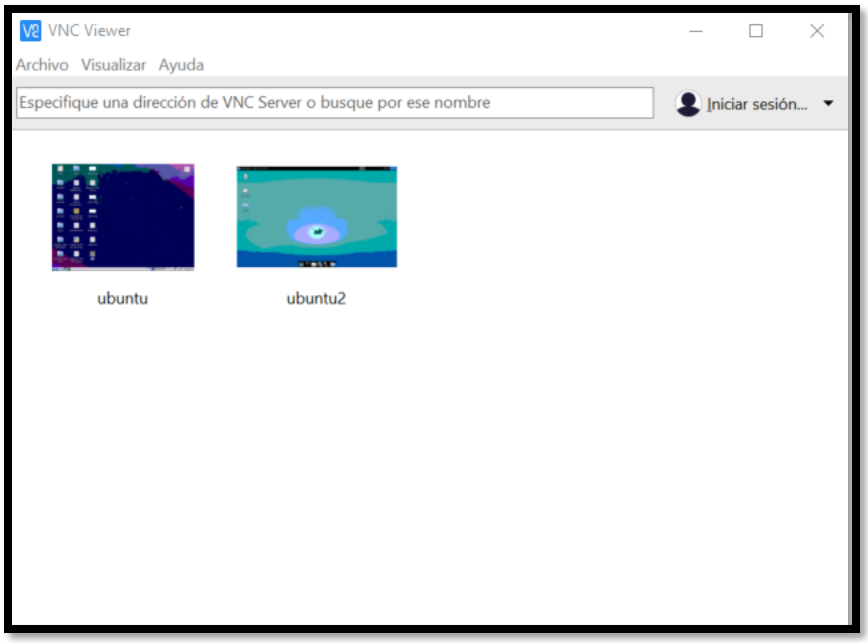

Figura 9: VNC Viewer en ejecución.

2.7. Modelo de datos

A la hora de realizar un Datawarehouse, es recomendable darles una estructura a los datos, de manera que sea posible acceder a estos de una manera más fácil y ordenada. A continuación, se muestran dos estructuras ampliamente utilizadas en el ámbito.

2.7.1. Modelo estrella

Este modelo es considerado sencillo pues consta de sólo dos tipos de tablas: una de hechos y otra de dimensiones. Se caracteriza por tener todas las tablas de dimensiones ligadas a la tabla de hechos, y por tener sólo una tabla de dimensiones para cada dimensión.

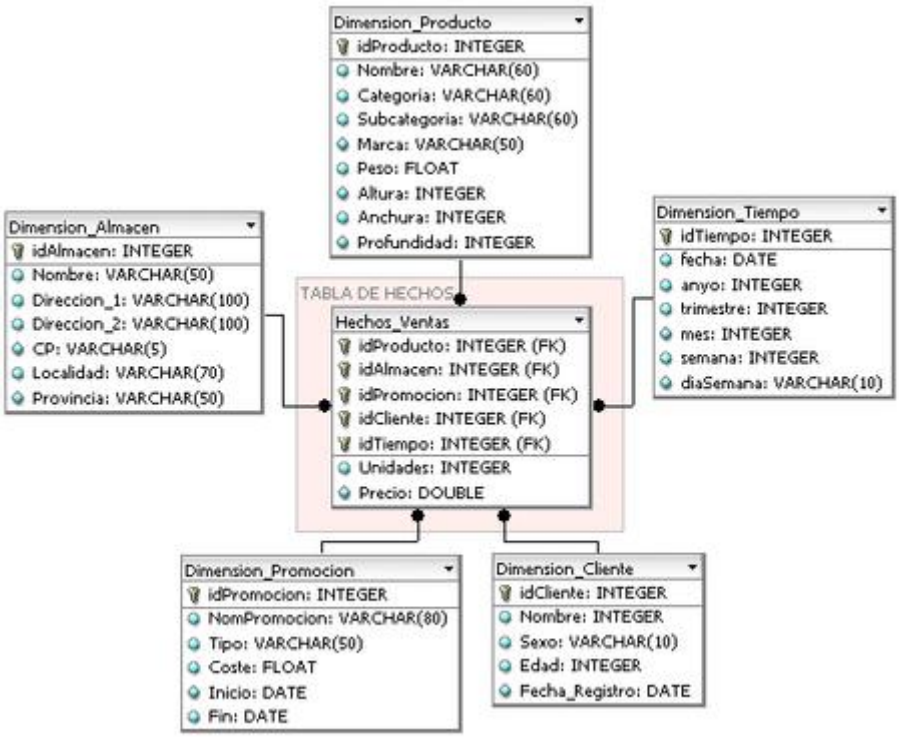

Figura 10: Ejemplo de modelo de datos en estrella. (Wikipedia, 2019)

2.7.1. Modelo copo de nieve

Este modelo es una variación del modelo estrella, en donde no todas las tablas de dimensiones se conectan directamente con la tabla de hechas, sino que algunas se relacionan con otra tabla de dimensiones. La finalidad de esto es facilitar el mantenimiento de las dimensiones, pero aquello hace más compleja la extracción de datos.

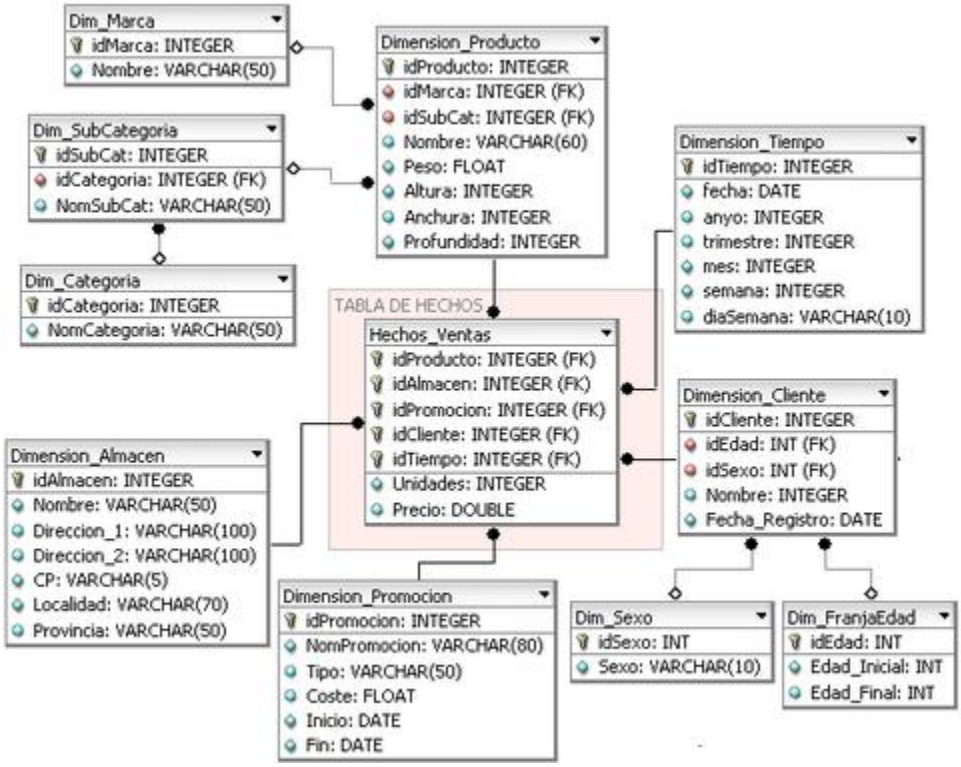

Figura 11: Ejemplo de modelo de datos en copo de nieve. (Wikipedia, 2019)

3. Diseño

3.1. Introducción

Luego de analizar las herramientas disponibles, y de dar una breve introducción a ciertos conceptos necesarios para el proyecto, se explicará el diseño de la arquitectura de descarga que, más adelante, será implementada para lidiar con la oportunidad ya explicada.

En primer lugar, se presentará una visión general del sistema en conjunto con sus unidades funcionales. Luego se explicará la arquitectura diseñada en general para luego abordar de manera más detallada cada capa de esta. La arquitectura incluye partes relacionadas con la obtención de datos, el modelamiento de datos y la visualización de datos. Finalmente se mostrará cómo este sistema podrá ser ejecutado de manera remota, además de sus ventajas frente a un entorno local.

3.2. Visión General y unidades funcionales

Tal como se dijo previamente, se desea crear una arquitectura de descarga, junto a un modelo de datos, que permita a una PYME mejorar su propuesta de valor hacia sus clientes por medio de un dashboard que muestre los datos respectivos a los documentos tributarios del cliente. Dado que se desea que el producto final se ejecute de manera automática todos los días, se trabajará en dos entornos durante el desarrollo del proyecto: uno local y otro remoto.

La ventaja de trabajar en dos entornos es que permitirá que el desarrollo sea más rápido. Editar código de manera local es más rápido, además de ser más cómodo al no tener que lidiar con problemas de latencia, en comparación con un entorno remoto. Una vez el código esté listo, tendrá que ser migrado a un entorno remoto, el cual es más estable e independiente, dando mayor facilidad para una automatización diaria.

Regresando al diseño del sistema en sí, se establecerá a un operario de la empresa como administrador del sistema. Este no necesariamente será único, aquel es un factor que dependerá de la empresa en sí, por lo que no es necesario establecer un permiso especifico en el sistema para su uso. El factor que permitirá a un usuario ser administrador del sistema es tener el permiso de la empresa para tener todos los elementos necesarios para preparar el entorno de trabajo. Estos varían dependiendo si se desea trabajar en un modo local, remoto, o sólo se desea ver la base de datos, dejando de lado el proceso de descarga de datos. Dentro de los permisos necesarios, se encuentra el certificado digital de la empresa, usuario y contraseña de la base de datos, permiso para acceder a la base de datos desde código local, permisos para visualizar el servidor remoto, entre otros.

El administrador del sistema tendrá los permisos de activar o pausar la descarga diaria de datos, realizar descargas en específico para un cliente, trabajar el

modelamiento de datos y editar los resultados a mostrar en la visualización de los datos hacia los clientes. Estas funciones forman tres unidades funcionales del sistema.

3.2.1. Obtención de datos

La primera tarea a monitorizar el administrador del sistema es la obtención de datos, lo cual estará basado en un código escrito en Python y ejecutado en un servidor remoto. La función comenzará realizando web scraping en la página del SII para descargar la información correspondiente a los documentos tributarios del cliente. El entorno estará configurado para ejecutar esta acción con todos los clientes de la empresa de manera diaria para tener la información actualizada. El administrador de sistema podrá detener el proceso de ser necesario, modificar los clientes a los cuales se realiza la descarga, elegir qué documentos tributarios se descargan, entre otros.

Los datos serán descargados como XML desde la página del SII, se transformará la información a un formato más adecuado y se insertará la información en una base de datos. No será necesario almacenar los XML originales.

3.2.2. Modelamiento de datos

Para poder analizar de manera fácil la información descargada, se debe estructurar y organizar los datos. Esto permitirá realizar consultas más eficientes a la base de datos, además de que reducirá la complejidad de estas. Dado que los datos son descargados directamente desde el SII, no será necesario realizar una limpieza a la base de datos, ya que los documentos tributarios ya fueron aceptados por el SII, así que se puede tomar el supuesto de que estos no presentan problema de formato.

3.3.3. Visualización de datos

Esta unidad es la encargada de mostrar los resultados tras el análisis de la base de datos. En este caso, es una visualización personalizada para cada empresa cliente en base a sus propios documentos tributarios, la cual contendrá indicadores, tablas y gráficos que permitirán al cliente conocer de mejor manera su propia información. El administrador del sistema tendrá la libertad de modificar la visualización disponible de la información hacia los usuarios, además de crear otros servicios basados en la información recolectada de los clientes, tanto para ellos como para la empresa en sí.

3.3. Arquitectura del sistema

A partir de la investigación de herramientas disponibles para el proyecto, en conjunto con una búsqueda de proyectos similares, se decidió realizar una arquitectura de cinco capas para cumplir con los requerimientos necesarios. La definición de estas capas está basada en el artículo “Arquitectura referencial de Big Data para la gestión de las telecomunicaciones” (Lieter Plasencia Moreno, 2017) por Lieter Plasencia Moreno y Caridad Anías Calderón, en donde proponen una arquitectura de cinco capas, pero con herramientas de Big Data que no vienen al caso para este proyecto. La arquitectura planteada se puede ver en la Figura 12.

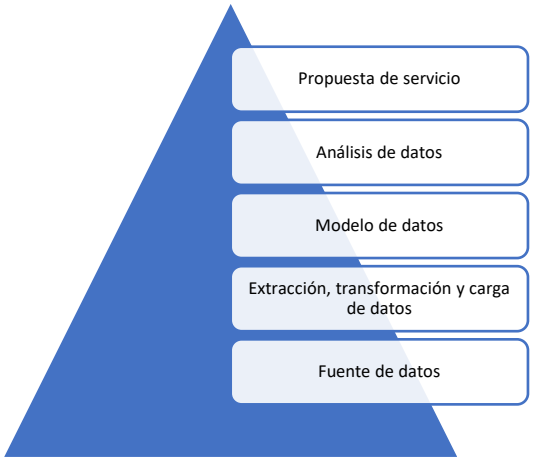

Figura 12: Arquitectura de descarga.

El nivel más bajo de la arquitectura corresponde a la fuente que genera los datos, en este caso, el sistema de respaldo de DTE del SII. El segundo nivel corresponde a la extracción, transformación y carga de datos. Más adelante se verá en detalle la elección de seguir estos tres pasos en orden, en vez de cargar los datos originales y transformarlos posteriormente al realizar el análisis de datos, como podría pasar en otra arquitectura.

El tercer nivel considera el modelo de datos, en donde se estructura y se organizan los datos para su posterior análisis. El cuarto considera el análisis por medio de consultas a los datos, además de comprobar que los datos ingresados están correctos, y una visualización de la información. Por último, se tiene la quinta capa correspondiente a la propuesta de servicio, es decir, en este caso, qué decisiones puede tomar un cliente en base a la información entregada.

3.3.1. Primera capa: Fuente de datos

La primera capa de la arquitectura planteada consiste en de dónde se sacarán los datos deseados. En este caso corresponden a documentos emitidos y recibidos por clientes[3] de la empresa, los cuales se encuentran en la página web del SII. Estos documentos sólo se pueden descargar como archivos XML, los cuales pueden tener desde uno hasta veinte documentos distintos.

Los elementos que hacen que un documento sea único son el folio, el tipo de DTE[4], el RUT del emisor y el RUT del receptor.

Para poder tener acceso a estos datos, un operario de la empresa debe ingresar a la página web del SII utilizando el certificado digital de la empresa. Los pasos necesarios para llegar a la plataforma de descarga son:

[3] Los clientes de la empresa son, en general, empresas pequeñas o de tamaño mediano, pero no está limitado a ello.
[4] Entre estos se encuentran boletas electrónicas, notas de crédito, notas de débito, guías de despacho, entre otras.

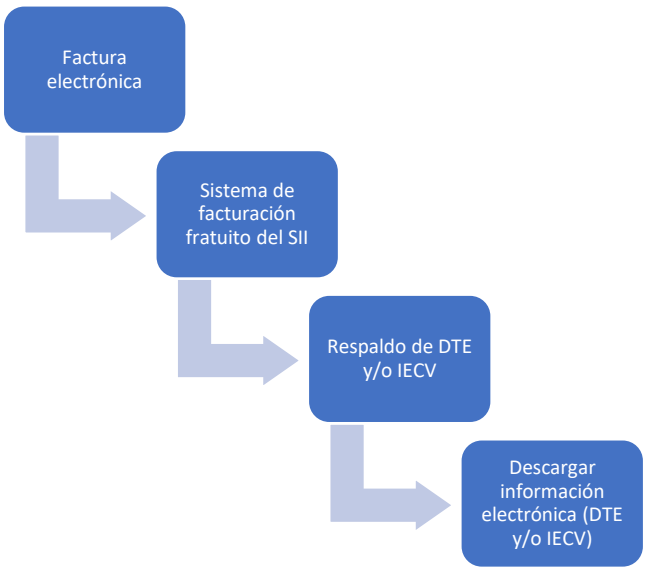

Figura 13: Pasos para ingresar a sistema de descarga de información del SII

Luego se tiene la opción de ingresar el RUT de una de las empresas clientes que ya pasó por el proceso de certificación y se selecciona la opción DTE. Esto es posible debido a que, tras terminar con el proceso de certificación de un cliente nuevo, la empresa que provee el servicio queda registrada como usuario autorizado para ingresar a la página del SII en representación de la empresa cliente.

Una vez en el sistema, se puede seleccionar el origen del documento, es decir, si es emitido o recibido. Posterior a ello se pueden aplicar filtros para reducir los resultados obtenidos, tales como:

- Tipo de documento
- Fecha desde hasta
- Folio desde hasta
- RUT
- Razón social

Estos filtros son necesarios para el desarrollo de un algoritmo de descarga de los documentos, ya que sólo podemos descargar paquetes de no más de veinte documentos, por restricción de la plataforma del SII. Es decir, se debe hacer un algoritmo de descarga eficiente para los documentos emitidos y recibidos de manera que, utilizando los filtros disponibles, armar paquetes descargables sin repetir documentos ya descargados.

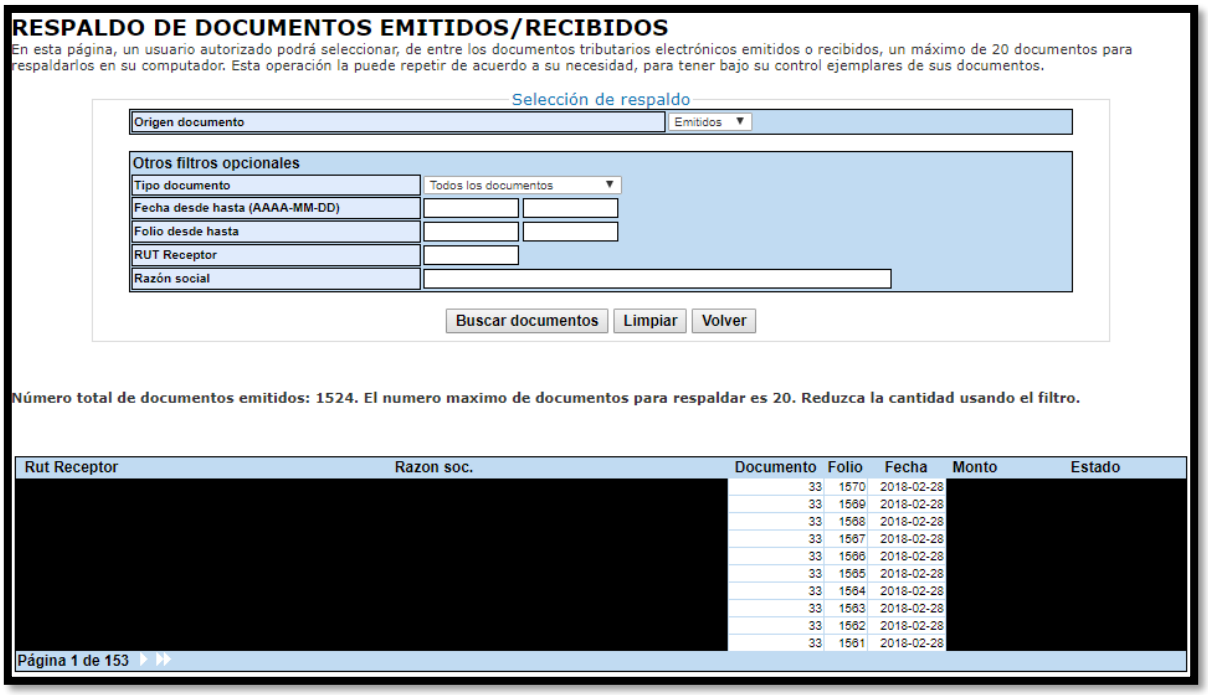

Figura 14: Plataforma de descarga de datos emitidos para un cliente sin aplicar filtros de búsqueda

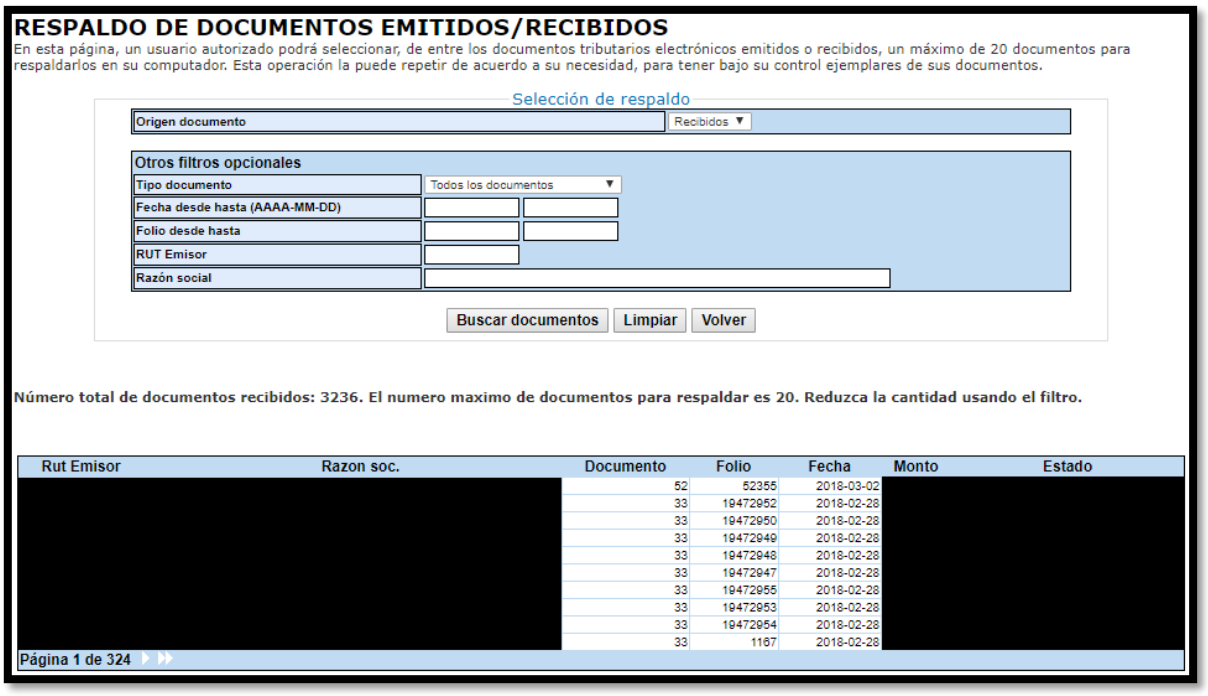

Figura 15: Plataforma de descarga de datos recibidos para un cliente sin aplicar filtros de búsqueda

Como se puede ver en la Figura 14 y Figura 15, si se desean descargar los documentos históricos de un cliente, es necesario utilizar filtros de búsqueda. En primera instancia, se deben descargar los documentos históricos de cada cliente y,

posteriormente, con el mismo algoritmo, descargar diariamente los documentos para cada uno de ellos.

Para el diseño de este proyecto, se trabajará únicamente con las facturas electrónicas de los clientes puesto a que son los documentos tributarios de mayor volumen. Se definirá una variable asignada al tipo de documento tributario, pero siempre se trabajará con la factura electrónica durante la implementación. Al finalizar la arquitectura, bastará cambiar sólo esta variable para abarcar el resto de los documentos, por lo que las pruebas a realizar más adelante considerarán a todos los documentos tributarios encontrados en la plataforma del SII.

Igualmente se trabajará con un horizonte de tiempo fijo correspondiente al año 2018 para tener resultados constantes durante las pruebas necesarias durante el desarrollo de la arquitectura.

3.3.2. Segunda capa: Extracción, transformación y carga de datos

La segunda capa consiste en obtener los documentos y poder guardarlos en un formato que facilite el acceso a los datos encontrados en estos. La descarga debe realizarse de manera remota, es decir, no en un equipo físico de la empresa, y ser un proceso totalmente automático. Los archivos tipo XML, los cuales pueden poseer hasta veinte folios distintos, deben ser separados para armar documentos únicos para ser almacenados. Debe tener un ID único que permita encontrarlo en la base de datos a crear una vez sea cargado.

3.3.2.1. Extracción

Dado las limitaciones que presenta la plataforma del SII para descargar los documentos deseados, se debe diseñar un algoritmo de descarga para los documentos emitidos y recibidos, los cuales tienen un comportamiento distinto a la hora de realizar la descarga. Una vez este sea diseñado y testeado de manera manual, se debe realizar un código que realice web scraping para automatizar el proceso, ejecutarlo de manera local en un equipo para probarlo, y, si todo sale bien, subirlo a un sistema remoto para que su ejecución sea automática y así no dependa de factores externos, como la estabilidad del equipo o de la conexión a internet.

La descarga se realizará para todos los clientes de la empresa, los cuales van variando continuamente, por lo que es necesario conectarse a la base de datos de la empresa para rescatar una lista con los RUT necesarios. No obstante, por temas de comodidad, la lista de clientes será ingresada como un dato fijo durante la realización de este proyecto. Conectarse a la base de datos para rescatar la lista de clientes para cada intento de descarga durante el desarrollo de la arquitectura no presenta mayor beneficio en comparación con una lista con valores fijos, además que significaría un costo innecesario, tanto monetario como en tiempo. Al finalizar el proyecto y tener que entregar el producto final a la empresa, se deberá cambiar la lista fija por una variable que se desprenda de la base de datos de la empresa.

3.3.2.1.1. Algoritmo de descarga: Documentos emitidos

Al ser en este caso los documentos siempre emitidos por un mismo emisor, el número asociado al folio es incrementado en uno por cada nuevo documento emitido dentro de la categoría.[5] Tal como se puede ver en la Figura 14, la búsqueda de documentos emitidos entrega por defecto el último documento con el folio más grande, además del dato de la cantidad del número total de documentos emitidos en el periodo. Siendo este el caso, se puede filtrar por el número de folio para crear paquetes de veinte folios consecutivos y poder así descargarlos hasta que el número de documentos descargados sea igual al número total de documentos emitidos.

Un detalle importante para considerar es que en ocasiones los documentos pueden saltarse folios debido a la cancelación de un documento tributario debido a, por ejemplo, un error en esta; por esta razón no se puede asumir que cada paquete de descarga contenga efectivamente la cantidad de documentos que promete. Existe la probabilidad de que se deban descargar más paquetes que los planeados inicialmente.

Otro factor importante es que, dado un periodo de tiempo para el horizonte de descarga, se deben descargar únicamente los datos correspondientes tal de no descargar documentos extras.

3.3.2.1.2. Algoritmo de descarga: Documentos recibidos

Los documentos recibidos, a diferencia de los emitidos, no poseen un orden asociado al número de sus folios. En primera instancia se debe filtrar por el periodo establecido para identificar el número total de documentos recibidos en ese periodo. Para filtrar el resultado, se debe ir acotando manualmente el periodo de descarga hasta encontrar uno con veinte documentos o menos, descargarlos, y proseguir con la búsqueda de un nuevo periodo. En caso de que un día se encuentren más de veinte documentos, se debe filtrar por el folio específico de cada documento, es decir, hacer una descarga por separado para cada uno de esos documentos. El proceso se itera hasta descargar la cantidad de documentos especificados en la primera búsqueda.

Para hacer el algoritmo óptimo, se puede establecer una diferencia de tiempo recomendada en base del periodo inicial, la cantidad de documentos encontrados y el factor de que sólo se pueden descargar en paquetes de veinte documentos.

A modo de ejemplo, si se ingresa un periodo de cien días y se encuentran cien documentos, la diferencia de tiempo recomendada de descarga sería de veinte días. Si se considera una distribución uniforme, el cual sería el mejor de los casos, la obtención de los documentos se lograría en cinco descargas. A partir de esta búsqueda, se puede ir disminuyendo el horizonte en caso de que existan más de veinte documentos en ese periodo.

[5] Los documentos tributarios tienen folios independientes entre sí. Por ello puede haber, para una misma empresa, una factura electrónica y una nota de crédito, por ejemplo, con el mismo folio asignado.

3.3.2.1.3. Descarga de documentos

Como ya se dijo previamente, el producto final se conectará a la base de datos de la empresa para rescatar una lista actualizada de sus clientes a la hora de hacer una descarga de datos para cada uno de ellos. No obstante, se trabajará con una lista con algunos clientes fijos para agilizar el desarrollo de la arquitectura.

El algoritmo se probará primero de manera manual para evaluar los tiempos asociados a las descargas de documento, luego se correrá de manera automático en un equipo local y finalmente en uno remoto. Hacer que el código sea ejecutado diariamente de manera automática se implementará más adelante luego de terminar la arquitectura y realizar pruebas que entreguen buenos resultados.

Para que este proceso se desarrolle de manera automática, se desarrollará un código en Python de manera que simule la interacción de un usuario con la página web del SII y pueda así realizar las descargas. Este utilizará la librería Selenium para realizar el web scraping en un navegador de internet. La elección del navegador dependerá de qué tan amigable sea a la hora de trabajar con un certificado digital.

3.3.2.2. Transformación

Por defecto, lo que se obtiene del proceso de descarga desde el SII corresponde a archivos tipo XML que contienen hasta veinte documentos distintos, ya sean emitidos o recibidos. Antes de considerar una transformación en cuanto al tipo de archivo que tendrá el documento que será cargado a un sistema, se debe separar cada XML en cada documento que lo compone.

Aprovechando que los XML funcionan en base a etiquetas, se crean diccionarios para guardar la información que estos contienen. Los documentos tienen una estructura fija, pero no un contenido fijo. En otras palabras, siempre se accederá a una variable por medio de las mismas etiquetas, pero esa variable no siempre estará presente. Este factor debe considerarse para no buscar elementos que no existen, lo cual haría que el código se cayera.

Un documento tributario consta de dos partes: el encabezado y el detalle. Todo documento tributario tiene un encabezado, pero puede tener múltiples detalles. Por ello se hacer un diccionario del encabezado y un diccionario de diccionarios para el detalle. Las llaves de acceso serán las etiquetas encontradas en el XML.

3.3.2.3. Carga

Los datos rescatados por medio de diccionarios serán cargados en Aurora PostgreSQL. Se eligió esta herramienta debido al volumen de datos con el que se planea trabajar. Debería tener mejores resultados que Athena en tema de eficiencia de las consultas y, por gusto personal, es más amigable que MySQL. Se elige Aurora PostgreSQL por sobre PostgreSQL para continuar con el ecosistema de AWS, además de ser una herramienta de mayor preferencia para la empresa.

3.3.3. Tercera capa: Modelo de datos

La tercera capa tiene como objetivo darle una estructura a los datos recolectados para que estos puedan ser trabajados de una manera fácil y eficiente. Siendo este el caso, y recordando la estructura presente en los XML descargados, se plantea un modelo de datos tipo estrella, donde la tabla asociada a hechos corresponde a los datos únicos del documento, mientras que las tablas asociadas a las dimensiones tendrían datos que se repetirían a lo largo de algunos documentos tributarios, tales como datos del emisor, receptor, tipo de pago, entre otros.

Debido a la manera en que están planteados los datos, un modelo tipo estrella presentará mejores resultados al momento de realizar una consulta en SQL respecto a una tabla única asociada a todos los datos encontrados en cada documento tributario. En cuento a la estructura tipo copo de nieve, no se justifica su utilización puesto que las mejores que presenta no aportan un beneficio directo al modelo; no existe problemas para el mantenimiento de las dimensiones.

Se trabajará en un ambiente de pruebas de la empresa en Aurora PostgreSQL en donde existen otras bases de datos de esta. Por ello se utilizará el prefijo “doc” para diferenciar los datos de este proyecto con otras tablas de la empresa.

Las tablas por construir serían:

- Tabla de hechos
    - doc_datos_documento
- Tablas de dimensiones
    - doc_detalle
    - doc_emisor
    - doc_fecha_emision
    - doc_receptor
    - doc_referencia
    - doc_transaccion

Los elementos individuales encontrados en cada tabla, al igual que sus dimensiones y llaves, serán detallados más adelante durante la implementación del modelo.

Cabe destacar que, si bien las tablas destinadas a los emisores y receptores son parecidos en estructura, se decidió tratarlos por separado debido a que no siempre tienen los mismos datos en los documentos tributarios. Además, más adelante harán más intuitivo el uso de filtros para ambas partes.

3.3.4. Cuarta capa: Análisis de datos

La cuarta capa de esta arquitectura tiene como finalidad analizar los datos recolectados para su posterior toma de decisiones por parte de una empresa cliente. Por ello se realizará un reporte personalizado para un cliente utilizando la herramienta Power BI de Microsoft.

Dado que cada reporte está destinado a un cliente en específico, se harán sólo dos reportes con dos clientes seleccionados manualmente de manera que cumplan con el requisito de tener varios documentos tributarios, al igual que tener una cantidad variada de tipos de documentos. Así mismo, no es necesario trabajar con toda la base de datos, sino que se puede trabajar con un conjunto de metadatos correspondiente a los relacionados con ambos clientes.

El reporte contendrá un reporte general y uno más específico, tanto para los documentos emitidos como para los recibidos del cliente, sumando en total cuatro páginas. El reporte general contendrá un gráfico correspondiente al recuento de folios por mes, un gráfico para el monto total por mes, y una tabla con el monto total de un tipo de documento tributario específico, ya sea según emisores o receptores. La finalidad de este reporte será que el cliente pueda tener una idea global de su negocio, identificando qué meses logró facturar más, además de identificar cuáles son sus clientes más importantes, al igual que los menos importantes.

El reporte específico contendrá gráficos que le permita al cliente identificar sus productos más populares, tanto en cantidad como en monto asociado. Además, se mostrará la cantidad de folios durante el periodo seleccionado para ayudar a cuantificar la cantidad asociada a cada producto. Poder identificar los productos específicos se puede lograr gracias al detalle encontrado en los documentos tributarios.

El reporte también considerará la existencia de notas de crédito que anulen una factura electrónica, haciendo que sólo se muestren en el reporte las facturas electrónicas válidas.

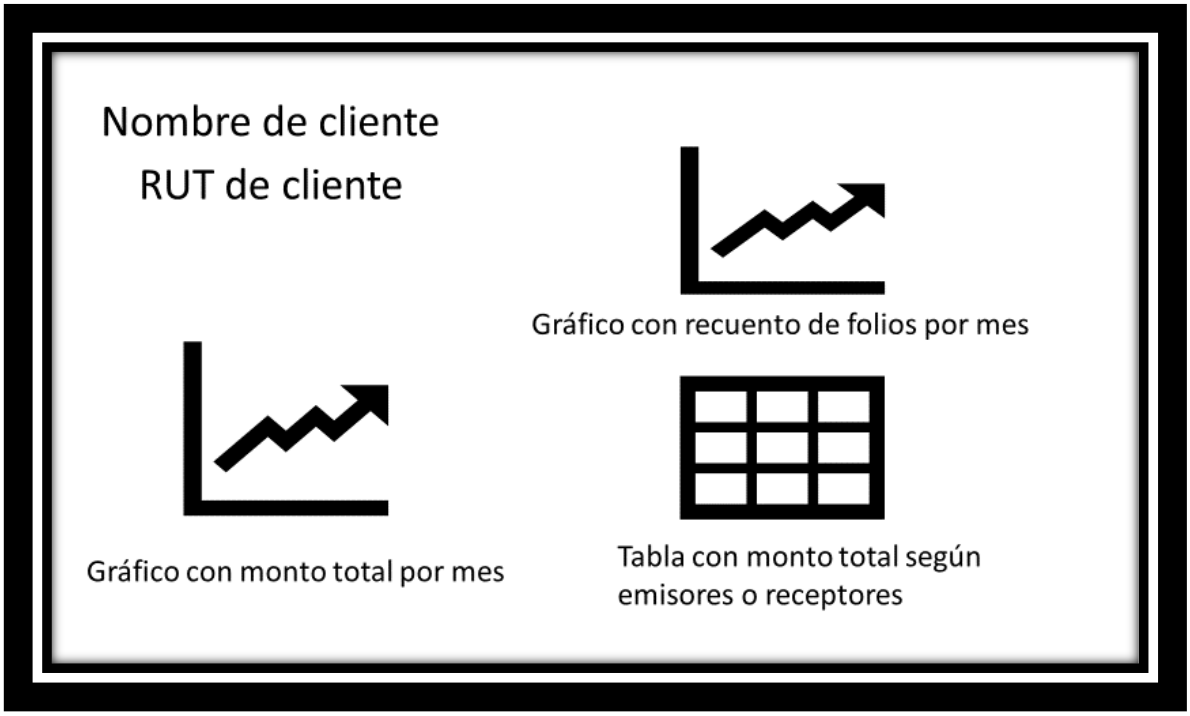

Figura 16: Mockup de reporte general

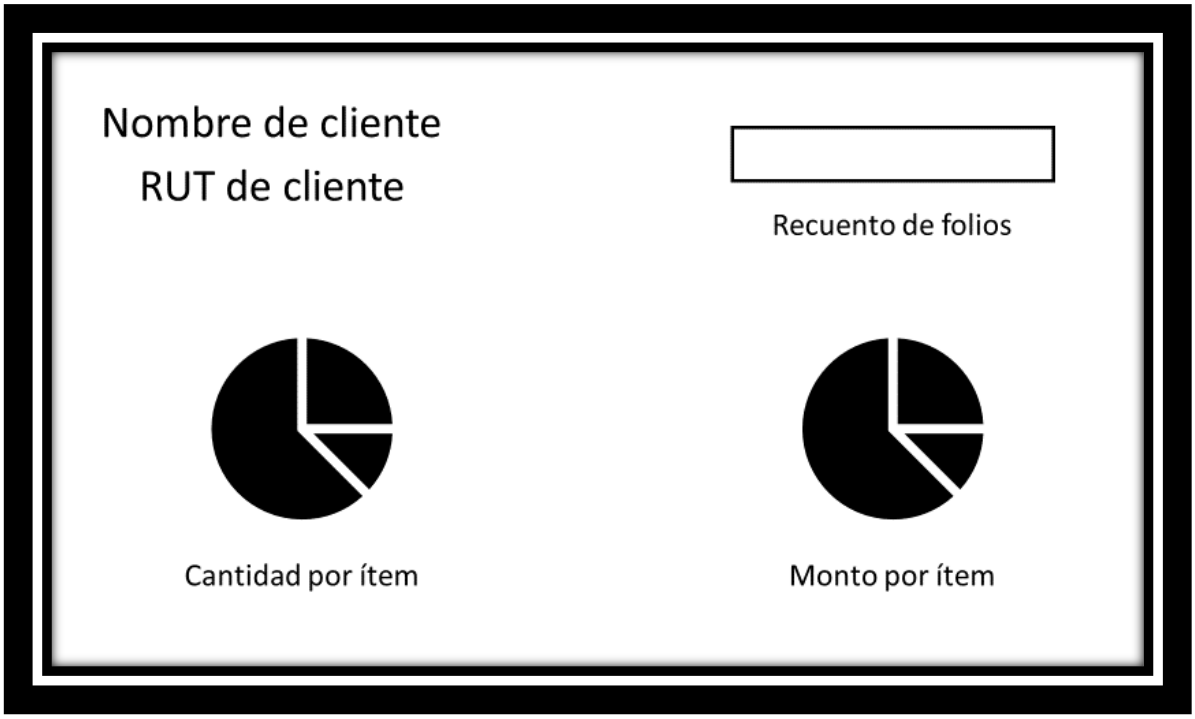

Figura 17: Mockup de reporte específico

3.3.5. Quinta capa: Propuesta de servicio

La quinta capa consiste en la toma de decisiones que el cliente hace en base a los datos observados, por esta razón queda fuera del alcance del proyecto. No obstante, en virtud de que se realizará un reporte real para dos clientes a modo de prueba utilizando sus datos, se presentarán conclusiones y posible toma de decisiones para esos casos, simulando la acción que tendría el cliente asociado tras obtener el reporte.

3.4. Servidor remoto

Una parte importante del proyecto es que, en el producto final a entregar a la empresa, todo código necesario para la creación y actualización diaria de la base de datos debe ser ejecutada de manera automática en un servidor remoto. La razón para ser automática no requiere de mucha explicación, aquello deja de lado el factor humano que se desea evitar. El caso de por qué elegir un servidor remoto frente a uno local es más interesante.

Comenzando con la base, todo código que uno programe y ejecute se desenvuelve en algún hardware en el mundo. El caso más común para la mayoría correspondería al hardware propio de la persona, es decir, un ordenador portátil o de escritorio, por ejemplo. No obstante, hoy en día existen varias alternativas basadas en la nube, las cuales son ejecutadas en alguna parte del mundo. Esto último es lo que se conoce como servidor remoto.

Esto tiene varias ventajas frente a un servidor local. Para empezar, no se requiere de un hardware físico. Aquello no sólo significa que no es necesario invertir en todos los componentes necesarios para tener un ordenador apto para la labor que se desea, sino que también no es necesario preocuparse por la mantención de este. En caso de que el hardware físico de nuestro servidor remoto, el cuales se encuentra en alguna parte del mundo, sufra algún imperfecto, dependerá de la compañía distribuidora del servicio solucionar el problema.

También tenemos el factor de que es más fácil actualizar las características de un sistema remoto que uno local. Si el tiempo pasa al punto de encontrarnos con que nuestro servidor remoto no cuenta con los requerimientos necesarios para ejecutar nuestro programa, basta con cambiar el plan contratado por uno con más CPU y memoria. En el caso de un servidor local, sería necesario cambiar el ordenador para tener un mejor desempeño del CPU.

Pero el factor más importante a considerar para este proyecto a la hora de escoger un servidor remoto es que este puede estar encendido en todo momento y que tiene una conexión estable y de alta velocidad a internet. Dado a que el proyecto se basa en hacer web scraping cada día, es muy importante contar con una conexión a internet fiable para que el código no se caiga, y poder tener la máquina siempre prendida para ejecutar el código cada día. Más allá de ello, si la base de datos está en la misma red privada que el servidor remoto, se tiene acceso directo a esta; no es

necesario un intermediario entre el cliente externo y la base de datos, lo que reducirá considerablemente los tiempos en las consultas.

3.4.1. EC-2

La empresa para la cual se desarrolla el proyecto facilitó una instancia en EC-2 para utilizarlo como servidor remoto. Este presentaba los requerimientos más bajos dentro de su categoría, denominadas instancias M5.

Las instancias M5 y M5d disponen del procesador de la serie Intel Xeon Platinum 8000 (Skylake-SP), con velocidad de reloj de CPU turbo constante para todos los núcleos de hasta 3,1 GHz. Adicionalmente, existe soporte para el nuevo conjunto de instrucciones de extensiones vectoriales avanzadas de Intel 512 (AVX-512), que ofrece hasta el doble de FLOPS por núcleo en comparación con las instancias M4 de la generación anterior. (Amazon AWS, 2019)

Modelo	**CPU virtual**	**Memoria (GiB)**	**Almacenamiento de instancias (GiB)**	**Ancho de banda de red (Gbps)**	**Ancho de banda de EBS (Mbps)**
m5.large	2	8	Solo EBS	Hasta 10	Hasta 3500
m5.xlarge	4	16	Solo EBS	Hasta 10	Hasta 3500
m5.2xlarge	8	32	Solo EBS	Hasta 10	Hasta 3500
m5.4xlarge	16	64	Solo EBS	Hasta 10	3500
m5.8xlarge	32	128	Solo EBS	10	5000
m5.12xlarge	48	192	Solo EBS	10	7000
m5.16xlarge	64	256	Solo EBS	20	10 000
m5.24xlarge	96	384	Solo EBS	25	14 000
m5.metal	96*	384	Solo EBS	25	14 000
m5d.large	2	8	1 x 75 SSD NVMe	Hasta 10	Hasta 3500
m5d.xlarge	4	16	1 x 150 SSD NVMe	Hasta 10	Hasta 3500
m5d.2xlarge	8	32	1 x 300 SSD NVMe	Hasta 10	Hasta 3500
m5d.4xlarge	16	64	2 x 300 SSD NVMe	Hasta 10	3500
m5d.8xlarge	32	128	2 × 600 SSD NVMe	10	5000
m5d.12xlarge	48	192	2 x 900 SSD NVMe	10	7000
m5d.16xlarge	64	256	4 × 600 SSD NVMe	20	10 000
m5d.24xlarge	96	384	4 x 900 SSD NVMe	25	14 000
m5d.metal	96*	384	4 x 900 SSD NVMe	25	14 000

Tabla 2: Detalles de instancias M5 (Amazon AWS, 2019)

La instancia entregada para trabajar es la m5.large, la cual cuenta con 2 CPU's virtuales, 8 GB de memoria y hasta 10 Gbps de ancho de banda de red. Más a futuro, en caso de necesitar correr más procesos en paralelo, la empresa podrá optar por crear otra instancia en EC-2, o mejorar la instancia entregada.

El servidor remoto entregado está hecho en base a Ubuntu, un sistema operativo open source. Este detalle será importante más adelante ya que las herramientas que se utilicen en el ambiente deben ser compatibles con este sistema operativo. En primera instancia, cuando se implemente el proyecto de manera local, será en

Windows, y se debe asegurar que todas las herramientas necesarias estén disponibles igualmente en Ubuntu.

3.4.2. VNC

Debido a que deseamos hacer web scraping en un navegador, necesitamos que el servidor virtual tenga una interfaz gráfica para que el código pueda acceder a elementos de una página web[6], al igual que una manera para acceder a esta interfaz. Por ello se instalará un entorno de escritorio libre para Linux que sea compatible con EC-2. La prioridad es que este sea rápido, ligero y que esté activo en cuanto a actualizaciones. Por ello se escoge XFCE, un entorno de escritorio que cumple con esas características. Si bien existen otras alternativas, las características de XFCE bastan para este proyecto, por lo que no es necesario buscar otras alternativas para realizar una comparación.

Figura 18: Logo of the Xfce project. (Xfce Team, 2008)

[6] Una interfaz gráfica no es necesario de por sí para hacer web scraping. En otros casos se podría hacer que el código trabaje con un navegador headless, es decir, sin encabezado. En este ejemplo, el código podría realizar todo el proceso de web scraping sin la necesidad de que un navegador web esté levantado y una interacción directa con ella sea efectuada. No obstante, dado que en este caso en particular se debe trabajar con un certificado digital, no se puede hacer el proceso headless, ya que en este caso el código no puede ingresar a la página web deseada al no haber compatibilidad entre headless y certificado digital. Si bien no se descarta que esta posibilidad exista de alguna manera, no se logró encontrar una manera de hacer esto posible tras buscar en varios foros de programación.

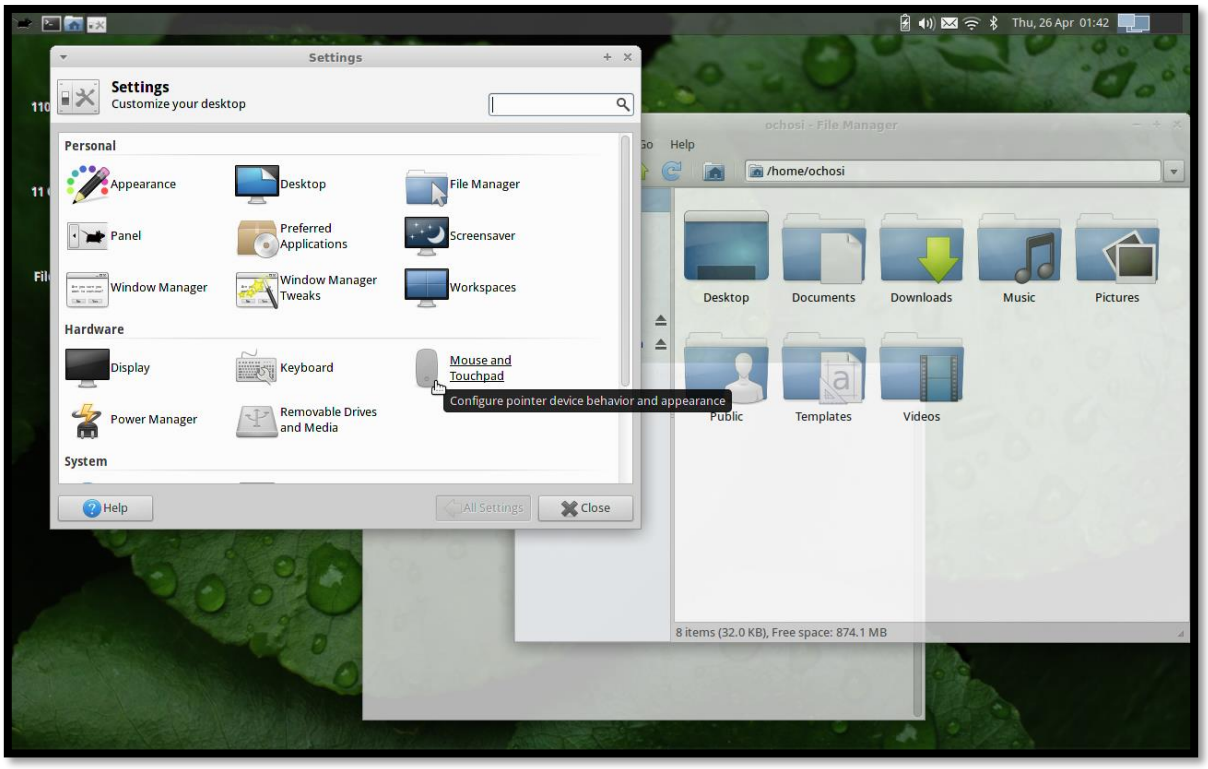

Figura 19: A screenshot of XFCE 4.10-1 with panel, Settings window, default file manager and xfce theme. (Xfce Development Team, 2012)

Una vez esté instalada la interfaz gráfica, se puede instalar de manera manual todas las herramientas necesarias en la plataforma, tales como un navegador web compatible con lo diseñado para realizar web scraping, librerías de Python, entre otras.

4. Implementación

4.1. Introducción

Tras mostrar el diseño de la arquitectura a crear, se mostrará a continuación la implementación de la solución.

Para comenzar, se mostrará cómo preparar el entorno de trabajo instalando todos los programas y librerías necesarias. Sumado a eso, se piden los permisos necesarios a la empresa para implementar la solución, lo que incluye permiso a su base de datos de clientes, permisos en AWS, acceso al certificado digital, entre otros.

Dado que ciertas decisiones dependían de la compatibilidad con certificados digitales, se explicará el cómo se llegó a elegir las alternativas seleccionadas. Esto afecta específicamente el proceso de web scraping y el cómo llevarlo a un servidor remoto.

Posteriormente se mostrará el desarrollo del modelo de datos para su análisis y la elaboración de un dashboard para la empresa cliente.

4.2. Preparación del entorno de trabajo

4.2.1. Instalación navegadores

Para poder realizar una automatización de un proceso web por medio de la simulación de un usuario en una página web, se necesita un navegador web. Debido a que estos son gratuitos y no requieren muchos recursos de un ordenador, es recomendable instalar más de uno para así hacer distintas pruebas que puedan llevar a la elección de uno para el proyecto.

Se instalan los navegadores Google Chrome, Mozilla Firefox y PhantomJS para probar la mejor alternativa para poder lidiar con un certificado digital a la hora de hacer web scraping.

4.2.2. Instalación de Python y librerías

Para el desarrollo de este proyecto, se eligió la version 3.6.8 de Python debido únicamente a una mayor familiaridad con esta versión. Presenta las herramientas necesarias para este proyecto. Esta versión fue lanzada el 24 de Diciembre del 2018.

En conjunto con ello, se actualizan las librerías que vienen por defecto en Python ingresando el siguiente comando en cmd:

```
python -m pip

python -m pip install --upgrade pip
```

4.2.2.1. Selenium

Como ya se dijo previamente, Selenium es una librería disponible en Python que permite automatizar procesos webs al ir seleccionando xpaths dentro de la página.

Es una librería esencial para realizar web scraping que se instala ingresando el siguiente comando en cmd:

```
pip install selenium
```

4.2.2.2. Requests

Requests es una librería para HTTP escrita en Python que hace que la integración con servicios web sea transparente. Es decir, no es necesario hacer consultas a URLs manualmente para hacer una petición POST. (Requests: HTTP para Humanos, 2019)

El uso más común que tiene esta librería es para trabajar con APIs, pero en este caso se utilizará para acceder a las cookies de una página web en específico. Se instala con el siguiente comando en cmd:

```
pip install requests
```

4.2.2.3. Xmltodict

Como su nombre lo indica, es una librería que permite trabajar archivos XML como si fuesen diccionarios y, por consecuencia, como JSON. Se instala con el siguiente comendo en cmd:

```
pip install xmltodict
```

4.2.2.4. Boto3

Permite crear, configurar y manejar servicios de AWS tales como EC2 y S3 desde Python, lo que permitirá guardar archivos en un bucket de S3. Se instala con el siguiente comando en cmd:

```
pip install boto3
```

4.2.2.5. Psycopg2

Es el driver más popular para acceder a PostgresSQL, lo que permite acceder a bases de datos y realizar distintas consultas. Se instala con el siguiente comando en cmd:

```
pip install psycopg2
```

4.2.3. Instalación certificado digital

Una vez se termina el proceso de certificación en el SII de un cliente por la empresa, esta se convierte en un representante legal del cliente, por lo que se puede acceder a la página del SII utilizando el certificado digital de la empresa.

Cuando uno accede a la página del SII y desea ingresar con un usuario, se da la opción de ingresar con un certificado digital. El problema para este proyecto es que le certificado digital debe ser seleccionado manualmente en una ventana que es mostrada al usuario. Esta ventana no forma parte del navegador, por lo que no puede simularse la interacción de un usuario en ella.

Una forma poco elegante de solucionar el problema durante el desarrollo del proyecto es simular la tecla Intro siendo presionada para así seleccionar el certificado digital cuando se le pregunte al usuario. Si bien esto cumple con el objetivo, existe un espacio para el error. Basta que el usuario pase a seleccionar un elemento cualquiera fuera del navegador para que la simulación de la tecla no se efectúe en este. En conjunto con esto, si existiese un segundo proceso siendo ejecutado en el mismo entorno, la simulación de la tecla siendo presionada podría interferir ese proceso.

Al investigar en foros de programación, tales como StackOverflow, se encontró la alternativa de definir un perfil de un usuario en el navegador seleccionado para acceder a este al realizar la simulación con código en Python. Si bien esta idea era prometedora, se encontraron distintos problemas que hizo imposible su implementación.

Para comenzar, al realizar una simulación por medio de Selenium, se abre un navegador con propiedades semejantes a un modo incognito. Esto significa que el navegador no recuerda perfiles creados ni preferencias de selección, por lo que no se puede simular la selección de un perfil con un certificado digital asociado. En diversos foros de StackOverflow se proponía la idea de definir directamente en el código el perfil del usuario, especificando la dirección de la carpeta asociada a los datos del usuario en el equipo.

Siguiendo las instrucciones que se encontraban en distintos foros, se intentó implementar la solución para los navegadores Google Chrome, Mozilla Firefox y PhantomJS, lográndolo únicamente en el tercer navegador luego de dos semanas intentándolo. Si bien se consideró entonces quedarse con ese resultado, se descartó debido a que PhantomJS es un navegador "headless"[7] que está descontinuado[8] (Wikipedia, 2019), por lo que podría traer problemas más adelante al proyecto, y posee una menor documentación en foros de programación.

Tras ello, se cambió la visión del problema, ahora buscando que el código acepte un único certificado por defecto en vez de cargarlo en el código. Se encontró documentación que permitía agregar opciones a la ventana de navegación que se creaba para la simulación y, si bien en algunos foros explicaban pasos para lograr lo cometido, ninguno logró evitar la aparición de una ventana para seleccionar el certificado digital de la empresa.

[7] Al cargar menor recursos, esto tiene la ventaja de que programar un proceso de automatización debería ser más rápido que el mismo navegador con una interfaz gráfica. No se puede asegurar que un navegador "headless" como PhantomJS sería más rápido que, por ejemplo, Google Chrome ya que ambos presentan características distintas. Una comparación más acertada sería comparar un mismo navegados en un modo clásico y un modo "headless", acto que se puede realizar en Google Chrome y Mozilla Firefox al realizar una simulación por medio de Selenium.

[8] El último lanzamiento estable fue lanzado el 24 de Enero del 2016. El desarrollo de PhantomJS fue suspendido en Marzo de 2018 debido a la falta de contribuciones activas.

Durante el proceso de solucionar este problema, se descubrió que la documentación disponible sobre utilizar certificados digitales con Selenium no era amplia, por lo que se terminó releyendo foros que ya habían dado paso a intentos fallidos.

Debido al fracaso que estaba significando hacer que el código de Selenium en Python solucionara el problema, se cambió una vez más la visión del problema, centrándose esta vez en el navegador. Se eligió Google Chrome debido a que tiene una gran documentación en foros de programación.

Se buscaba que el navegador tomara por defecto el primer certificado digital que estaba en sus registros, de manera de evitar que se mostrara al usuario una ventana con los certificados digitales instalados en el equipo.

En el foro de StackOverflow titulado “Chrome Certificate Selection appears multiple times”, publicado por Nelson Teixeira en Enero del 2015, se presenta el problema

```
{
   "AutoSelectCertificateForUrls": ["{\"pattern\":\"*\",\"filter\":{}}"]
}
```

planteado con su solución. Esta especifica ir a (/etc/opt/chrome/policies/managed) y agregar la siguiente configuración en el archivo de políticas:

Posteriormente se puede acceder a chorme:policy para confirmar que la configuración fue agregada. Con esta configuración, Google Chrome no pregunta por el certificado digital al usuario, sino que elige el primero por defecto, tanto en una interacción manual en un equipo por un usuario, como en una simulación por medio de código de Selenium en Python.

4.2.4. Permisos

Como ya se dijo en la sección de Diseño, la empresa facilitó una instancia en EC2 con el modelo m5.large, el cual incluye 2 CPU virtuales, 8 Gb de memoria y un ancho de banda de res de hasta 10 Gbps.

Teniendo esta instancia corriendo, debemos darnos acceso a tres puertos en la página de AWS para poder acceder a ella. A continuación, se muestra los pasos necesarios para poder agregar permisos en EC2

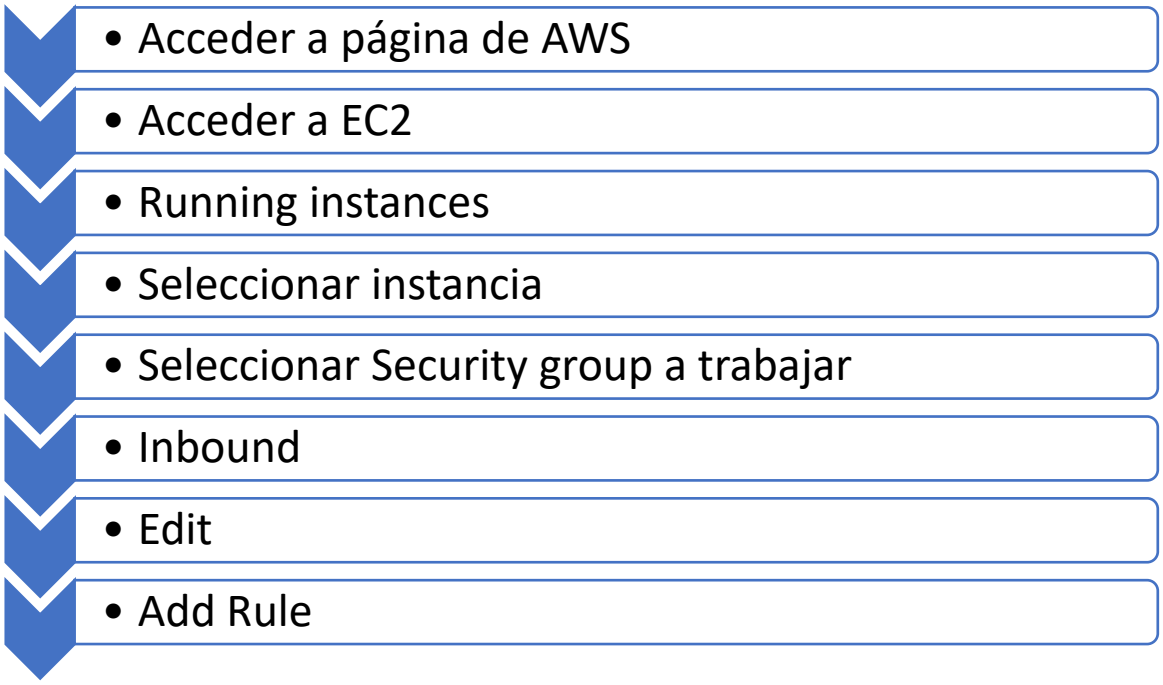

Figura 20: Pasos necesarios para dar permisos a puertos en AWS.

Una vez se llaga a la opción Add Rule, se debe dar permiso a tres puertos por medio de nuestra IP[9].

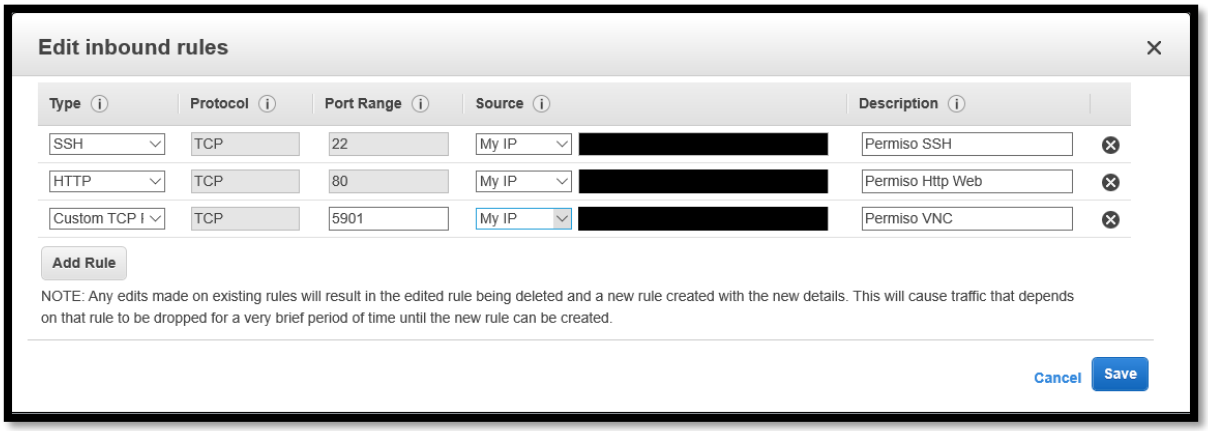

Figura 21: Acceso a puertos necesarios en AWS.

Los puertos confieren los siguientes permisos a la IP asignada:

- Puerto 22:
    - SSH (Secure Shell): Protocolo que permite el acceso remoto a un servidor por medio de un canal seguro. Nos permitirá enviar comandos en la consola del servidor remoto.

[9] Dado que la IP cambia al estar conectado a distintas redes, se debe verificar que la IP registrada en los permisos sea la nuestra antes de realizar cualquier acción en la implementación de la arquitectura.

    - SFTP (SSH File Transfer Protocol): permite serie de operaciones sobre archivos remotos. Nos permitirá trabajar archivos del servidor remoto desde el local, ya sea tanto de carga de archivos, descarga o editarlos.
- Puerto 80:
    - HTTP (Hypertext Transfer Protocol): Es el puerto desde el cual una computadora envía y recibe mensajes y comunicaciones basadas en el cliente web desde un servidor web. Nos permitirá acceder a la base de datos de la empresa en Aurora PostgreSQL, en conjunto con el usuario y contraseña necesario. También nos permitirá hacer un túnel[10] a la base de datos para que el código en el servidor local pueda comunicarse con ella.
- Puerto 5901:
    - VNC: Nos permitirá acceder al servidor remoto por medio de una emulación de su interfaz gráfica en el servidor local.

4.2.5. Herramientas para comunicación con servidor remoto

Ya teniendo la máquina virtual, necesitamos una manera de poder comunicarnos con los archivos en él. Para ello instalamos una version portable de WinSCP desde su página web. Se elige la versión portable únicamente por comodidad para evitar tener que instalarlo.

Para poder hacer uso de WinSCP, necesitamos el archivo PPK[11] de la empresa. Tras especificar su ruta e ingresar el IP del servidor remoto, podemos acceder a nuestro servidor remoto.

Igualmente se instala Cygwin para, más adelante, hacer un túnel para conectarse a la base de datos desde el código en el servidor local. Además, se instala VNC Viewer para acceder de manera visual al servidor remoto.

4.2.6. Interfaz gráfica con VNC

Ya teniendo la máquina virtual, es necesario instalar un ambiente de escritorio para poder dar el uso de web scraping deseado. Para ello, se siguen los pasos del post “How to Install and Configure VNC Server on Ubuntu 18.04 LTS” (Mutai, 2018), en donde se muestra cómo instalar XFCE, una interfaz gráfica ligera para Ubuntu.

Para comenzar, se abre un terminal en el servidor remoto por medio de WinSCP y se escriben las siguientes líneas de código en base a los pasos especificados en el post a seguir.

Paso 1.- Instalar servidor VNC en Ubuntu 18.04

```
sudo apt update
sudo apt -y install vnc4server
```

[10] Se entiende como túnel al intermediario entre un cliente externo y la base de datos.
[11] Archivos creados por PuTTYgen son conocidos como archivos PPK. Son utilizados para almacenar llaves primarias generadas por el programa.

Paso 2.- Instalar ambiente de escritorio

Paso 3.- Configurar servidor VNC

a) Establecer una contraseña de seguridad de acceso

```
$ vncpasswd
Password:
Verify:
```

b) Iniciar servidor VNC

```
$ vncserver :1

New 'ubuntu-01:1 (computingforgeeks)' desktop is ubuntu-01:1
Creating default startup script /home/computingforgeeks/.vnc/xstartup
Starting applications specified in /home/computingforgeeks/.vnc/xstartup
Log file is /home/computingforgeeks/.vnc/ubuntu-01:1.lo
```

c) Matar servidor VNC

```
$ vncserver -kill :1
Killing Xvnc4 process ID 20842
```

d) Establecer ambiente de escritorio en servidor VNC

- Abrir archivo de configuración para editar.

```
sudo vim ~/.vnc/xstartup
```

- Agregar la siguiente línea al final de archivo

```
exec /usr/bin/startxfce4 &
```

- Iniciar servidor VNC con:
  - Display number [1]
  - Screen resolution [800×600]
  - Color depth [24]

```
$ vncserver :1 -geometry 800x600 -depth 24

New 'ubuntu-01:1 (vagrant)' desktop is ubuntu-01:1
```

Paso 4.- Conectarse al escritorio de VNC

Si bien el post sugiere una serie de pasos para conectarse de manera visual al VNC, utilizamos VNC Viewer para acceder. La aplicación nos pide el IP del servidor, el cual se puede rescatar desde la página de AWS, y la contraseña creada en el Paso 3. Antes de abrir el servidor remoto, se cambia la resolución de este al mínimo para ahorrar recursos.

En caso de que todos los pasos hayan sido efectuados de manera correcta, el acceso debería ser posible y nos encontraríamos con el siguiente mensaje en el escritorio:

Figura 22: Primer mensaje mostrado en máquina virtual tras primer acceso exitoso mediante VNC Viewer.

Paso 5.- Configurar Systemd

a) Detener servidor VNC

```
vncserver -kill :1
```

b) Crer un archivo de unidad de servicio systemd para administrar el servidor VNC

```
sudo vim /etc/systemd/system/vncserver@.service
```

c) Agregar

```
[Unit]
Description=Start TightVNC server at startup
After=syslog.target network.target

[Service]
Type=forking
User=XXXXX
Group=XXXXX
WorkingDirectory=/home/XXXXX

PIDFile=/home/vagrant/.vnc/%H:%i.pid
ExecStartPre=-/usr/bin/vncserver -kill :%i > /dev/null 2>&1
ExecStart=/usr/bin/vncserver -depth 24 -geometry 1280x800 :%i
ExecStop=/usr/bin/vncserver -kill :%i

[Install]
WantedBy=multi-user.target
```

d) Recargar sistemd e iniciar VNC

```
sudo systemctl daemon-reload
sudo systemctl enable --now vncserver@1
```

e) Ver estado

```
$ systemctl status vncserver@1

* vncserver@1.service - Start TightVNC server at startup

   Loaded: loaded (/etc/systemd/system/vncserver@.service; indirect; vendor preset: enabled)

   Active: active (running) since Wed 2018-12-05 11:32:50 PST; 7s ago

  Process: 24161 ExecStart=/usr/bin/vncserver -depth 24 -geometry 1280x800 :1 (code=exited,
status=0/SUCCESS)

  Process: 24154 ExecStartPre=/usr/bin/vncserver -kill :1 > /dev/null 2>&1 (code=exited,
status=2)

 Main PID: 24168 (Xvnc4)

    Tasks: 63 (limit: 1110)

   CGroup: /system.slice/system-vncserver.slice/vncserver@1.service

           |-24168 Xvnc4 :1 -desktop ubuntu-01:1 (vagrant) -auth /home/vagrant/.Xauthority -
geometry 1280x800 -depth 24 -rfbwait 30000 -rfbauth /home/v

           |-24174 vncconfig -iconic

           |-24176 /bin/sh /etc/xdg/xfce4/xinitrc -- /etc/X11/xinit/xserverrc

           |-24186 xfce4-session

           |-24189 /usr/bin/dbus-launch --sh-syntax --exit-with-session xfce4-session

           |-24190 /usr/bin/dbus-daemon --syslog --fork --print-pid 5 --print-address 7 --session

           |-24194 /usr/lib/x86_64-linux-gnu/xfce4/xfconf/xfconfd

           |-24198 /usr/bin/ssh-agent -s

           |-24202 xfwm4

.....
```

4.2.7. Preparación de entorno de trabajo remoto

Una vez levantado el entorno remoto, se debe realizar una preparación análoga para realizar el web scrpaing. En virtud de que Chromium[12] viene instalado por defecto, sólo necesitamos instalar las librerías necesarias de Python, y dejar establecido que

[12] Chromium es un navegador web gratuito y de código abierto desarrollado por Google. Al realizar web scraping, necesitamos tener el archivo “chromedriver” necesario para trabajar en Ubuntu.

el sistema acepte el primer certificado digital para que así el proceso sea realmente automático.

4.3. Arquitectura de cinco capas

4.3.1. Primera capa: Fuente de datos

Antes de comenzar con la descarga de datos desde la plataforma del SII, se toma de parte de la empresa una lista de sus clientes actuales. Dado que se trabajará con documentos tributarios de los clientes, se desea saber si durante el periodo objetivo, estaban vigentes, independiente de si para entonces eran clientes de la empresa o no.

Para poder identificar esto, se desarrolla un código en Python que realice web scraping utilizando Selenium en el navegador Google Chrome. Este código ingresa al SII utilizando el certificado digital del representante legal de la empresa y, para cada RUT proporcionado por la empresa, rescatar la cantidad de facturas emitidas y recibidas. Si bien el producto final será capaz de trabajar con más documentos tributarios, para la implementación de la arquitectura se trabajará únicamente con las facturas electrónicas puesto a que tienen un volumen considerablemente mayor frente a otros documentos.

Por otro lado, si bien el producto final debe ser capaz de ser ejecutado todos los días para así tener una base de datos actualizada, para el desarrollo del proyecto se trabajará con un periodo de tiempo fijo correspondiente al año 2018. Este horizonte de tiempo dará un volumen de datos que permitirá más adelante diseñar una herramienta que beneficie una parte del negocio de la empresa.

Considerando estas decisiones, se corre el programa de web scraping para los 1739 clientes entregados por la empresa.

N° de clientes totales	**1739**
N° de clientes con facturas electrónicas en 2018	817
N° de clientes sin facturas electrónicas en 2018	922
Facturas emitidas	81830
Facturas recibidas	247081

Tabla 3: Análisis de resultados de web scraping para clientes de empresa

Con estos valores se puede dimensionar de manera concreta el volumen que tendrá la base de datos de facturas electrónicas de clientes de la empresa que se busca generar.

Antes de continuar con las siguientes capas de la arquitectura, se identifica a un cliente que posea varias facturas durante el periodo 2018 para así armar las capas en torno a este cliente en específico. Si posee varias facturas, tanto emitidas como recibidas, presentará una mayor cantidad de casos posibles para probar el futuro código. Este cliente será denominado cliente_1, el cual posee 183 facturas emitidas y 692 facturas recibidas.

Más adelante, cuando la arquitectura tenga más forma, se agregarán otros clientes con varias facturas y otros documentos tributarios.

4.3.2. Segunda capa: Extracción, transformación y carga de datos

4.3.2.1. Extracción

4.3.2.1.1. Manual

Se utiliza el algoritmo diseñado para la descarga de documentos emitidos y recibidos para obtener las facturas del cliente_1 durante distintos periodos del 2018 para evaluar el tiempo requerido en el caso de que un humano deba realizar el proceso, el cual es el peor de los casos y la actual situación para la empresa cuando un cliente pide que le entreguen los respaldos de sus facturas para cierto periodo a su criterio.

Origen	**Octubre**		**Noviembre**		**Diciembre**		**Octubre-Diciembre**	
	Tiempo	Facturas	Tiempo	Facturas	Tiempo	Facturas	Tiempo	Facturas
Emitido	40 seg	17	36 seg	13	32 seg	12	46 seg	44
Recibido	66 seg	65	51 seg	52	33 seg	61	2 min 40 seg	178

Tabla 4: Resultados de descarga manual de facturas

4.3.2.1.2. Automática

Se desarrolla un programa en Python utilizando la librería Selenium para hacer web scraping en Google Chrome para descargar facturas de un cliente. El programa accede a la página del SII utilizando el certificado digital del representante legal de la empresa y accede a los documentos de su cliente. Para descargarlos, se utiliza el mismo algoritmo de descarga de la sección anterior. Esta vez se realiza la descarga para un periodo de tres meses y de un año.

Origen	**Octubre-Diciembre**		**2018**	
	Tiempo	Facturas	Tiempo	Facturas
Emitido	18 seg	44	21 seg	183
Recibido	48 seg	178	2 min 7 seg	692

Tabla 5: Resultados de descarga automática de facturas

4.3.2.2. Transformación

Se aprovecha la estructura de los XML descargados para hacer diccionarios con su información. Utilizando la librería xmltodict en Python, se transforma cada archivo por separado en un diccionario que contiene toda la información del documento. Posteriormente se crean dos diccionarios con la información de interés: la del enunciado y del detalle.

Para el caso del enunciado, si bien se divide en IdDoc, Emisor, Receptor y Totales, las etiquetas internas que estas categorías tienen son únicas, tal como se puede ver en la Figura 5, por lo que no es necesario segmentar el diccionario en distintas partes.

Se crea un ID único para el documento, el cual es una combinación del tipo de DTE, folio, RUT emisor y RUT receptor; estos valores en conjunto es lo que hacen a un documento tributario único. Posteriormente se crea un diccionario que recibe como llave la ID del documento, y como valores las etiquetas encontradas en el

encabezado. Además, se incluye el valor de la ID del documento al final de este para conectarlo al diccionario de detalle. Más adelante esto será importante al ingresarlo a la base de datos.

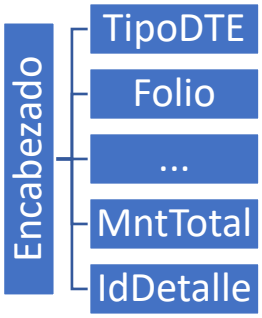

Figura 23: Estructura de diccionario para encabezado.

Dado que un documento puede tener múltiples detalles, se hace un diccionario de diccionarios para el detalle. El diccionario madre tiene como llave el ID del documento, y como valores un diccionario que tiene el número del detalle como llave, y como valores los datos del detalle y el ID del documento.

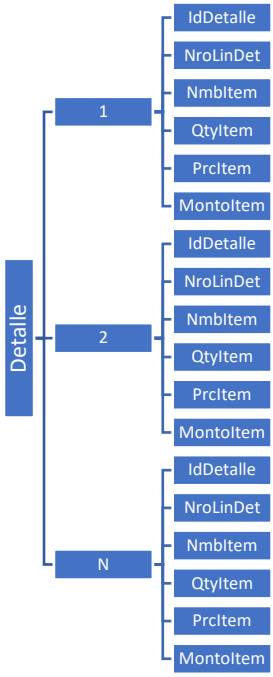

Figura 24: Estructura de diccionario de diccionarios para detalles.

Un factor importante para considerar es que los documentos tributarios no necesariamente llevan las mismas etiquetas, incluso en la misma categoría de DTE. Para evitar este problema, en caso de no encontrar la etiqueta deseada, se entrega un valor vacío en el lugar correspondiente, tanto para el encabezado como para el detalle. Los únicos valores que no pueden ser vacíos en el encabezado son las etiquetas asociadas al tipo de DTE, folio, fecha de emisión, RUT de emisor y RUT de

receptor; mientras que para el detalle es el número del detalle, nombre de ítem y el monto del ítem. No es necesario comprobar si estos valores existen, pues son requisitos para que un documento tributario sea aceptado por el SII.

4.3.2.3. Carga

Una vez obtenido los valores deseados en un formato agradable para su trabajo, se procede a cargar el registro en una base de datos. Antes de que el código pueda hacer esto, es necesario hacer un túnel que conecte el equipo con la base de datos. Para ello se corre un comando ssh en Cygwin, el cual tiene valores como el puerto de la base de datos, su URL, el archivo PEM[13] asociado al certificado digital necesario para la conexión, entre otros. Por ser valores confidenciales de la empresa, no se especifica el comando ejecutado.

Antes de ejecutar el comando ssh en Cygwin, se comprueba que el equipo local tenga acceso a los puertos necesarios desde AWS. De tener los permisos necesarios, en conjunto con haber logrado realizar el túnel, se ejecuta el código de Python. A continuación, se muestra el tiempo que tarda para el Cliente_1 realizar todo el proceso de la segunda capa.

Origen	**2018**	
	Tiempo	Facturas
Emitido	4 min 25 seg	183
Recibido	25 min	692

Tabla 6: Tiempo de ejecución para segunda capa en servidor local.

Una vez logrado realizar la segunda capa de manera local, se migra el código a una máquina virtual en EC2. Se utiliza WinSCP para copiar los archivos necesarios desde el equipo local a la máquina virtual y se edita el código lo necesario para que acceder a la base de datos sin la necesidad de hacer un túnel.[14]

Tras configurar el sistema, se ejecuta el mismo caso de un año para el Cliente_1 con los siguientes resultados.

Origen	**2018**	
	Tiempo	Facturas
Emitido	1 min 6 seg	183
Recibido	4 min 15 seg	692

Tabla 7: Tiempo de ejecución para segunda capa en servidor remoto.

4.3.3. Tercera capa: Modelo de datos

4.3.3.1. Tablas para documentos descargados

Para realizar las pruebas de carga en la segunda capa de la arquitectura, es en realidad necesario haber definido un modelo de datos para cargar la información en

[13] PEM es un formato de archivo empleado para almacenar certificados digitales.
[14] A diferencia del caso local, donde es Cygwin quien permite hacer el túnel a la base de datos, en el caso remoto es el mismo Ubuntu quien hace esta labor.

esta. En la práctica, la tercera capa es implementada junto a la parte de carga de la segunda capa.

Para estructurar los datos, y así facilitar su uso más adelante, se crean siete tablas distintas, siguiendo la estructura del modelo estrella.

Dado que no todos los documentos tributarios son exactamente iguales en estructura, tanto dentro del mismo tipo de DTE como entre distintos, varias variables tienen el valor de nulo por defecto, ya que son valores que no necesariamente se encontrarán en el documento.

4.3.3.1.1. doc_datos_documento

Columna	**Tipo**	**Descripción**
id_doc	integer *Incremento automático* [**nextval('doc_datos_documento_id_doc_seq')**]	Llave primaria asociada a un documento
id_fecha_emision	integer	Llave foránea asociada a la fecha de emisión del documento
id_transaccion	integer	Llave foránea asociada a la sección de transacción de un documento tributario
id_emisor	integer	Llave foránea asociada al emisor del documento
id_receptor	integer	Llave foránea asociada al receptor del documento
folio	integer	Folio encontrado en el documento tributario
tipo_dte	integer	Tipo de DTE de documento
monto_neto	Integer NULL	Monto neto del documento
tasa_iva	character varying(10) NULL	Tasa de IVA
iva	Integer NULL	Monto de IVA
mnt_total	Integer NULL	Monto total del documento

Tabla 8: Estructura de tabla doc_datos_documento

4.3.3.1.2. doc_detalle

Columna	Tipo	Descripción
id_doc	integer	Llave foránea asociada a un documento
nro_lin_det	integer	Número de línea del detalle
nmb_item	character varying(256)	Nombre del producto
qty_item	character varying(20) NULL	Cantidad del producto
prec_item	character varying(20) NULL	Precio individual del producto
monto_item	integer	Monto total por venta del producto

Tabla 9: Estructura de tabla doc_detalle

4.3.3.1.3. doc_emisor

Nombre	Tipo	Descripción
id_emisor	integer Incremento automático [nextval('doc_emisor_id_emisor_seq')]	Llave primaria asociada al emisor de documento
rut_emisor	character varying(20)	RUT del emisor
rzn_social	character varying(256) NULL	Razón social del emisor
giro_emisor	character varying(256) NULL	Giro del emisor
acteco	character varying(40) NULL	Acteco del emisor
cdg_sii_sucur	character varying(40) NULL	Código de sucursal del SII
dir_origen	character varying(256) NULL	Dirección de emisor
comuna_origen	character varying(80) NULL	Comuna de emisor
ciudad_origen	character varying(80) NULL	Ciudad de emisor

Tabla 10: Estructura de tabla doc_emisor

4.3.3.1.4. doc_fecha_emision

Columna	Tipo	Descripción

id_fecha_emision	integer Incremento automático [nextval('doc_fecha_emision_id_fecha_emision_seq')]	Llave primaria asociada a fecha de emisión
fecha	date	Fecha de emisión
ano	integer	Año de emisión
mes	integer	Mes de emisión
dia	integer	Día de emisión
q	integer	Cuarto del año de fecha de emisión

Tabla 11: Estructura de tabla doc_fecha_emision

4.3.3.1.5. doc_receptor

Nombre	**Tipo**	**Descripción**
id_receptor	integer Incremento automático [nextval('doc_receptor_id_receptor_seq')]	Llave primaria asociada al receptor
rut_receptor	character varying(20)	RUT del receptor
rzn_social_recep	character varying(256) NULL	Razón social del receptor
giro_recep	character varying(256) NULL	Giro del receptor
dir_recep	character varying(256) NULL	Dirección del receptor
cmna_recep	character varying(80) NULL	Comuna del receptor
ciudad_recep	character varying(80) NULL	Ciudad del receptor

Tabla 12: Estructura de tabla doc_receptor

4.3.3.1.6. doc_referencia

Nombre	**Tipo**	**Descripción**
id_doc	integer	Llave foránea asociada al documento
nro_lin_ref	integer	Número de línea de documento
tpo_doc_ref	integer	Tipo de documento de referencia
folio_ref	integer	Folio de referencia
fch_ref	date NULL	Fecha de referencia
cod_ref	Integer NULL	Código de referencia
razon_ref	character varying(256) NULL	Razón social de referencia

Tabla 13: Estructura de tabla doc_referencia

4.3.3.1.7. doc_transaccion

Columna	**Tipo**	**Descripción**

id_transaccional	integer Incremento automático [nextval('doc_transaccion_id_transaccion_seq')]	Llave primaria asociada al tipo de transacción
tpo_tran_compra	integer NULL	Tipo de compra
tpo_tran_venta	integer NULL	Tipo de venta
fma_pago	integer NULL	Forma de pago

Tabla 14: Estructura de tabla doc_transaccion

4.3.3.2. Modelo de datos para documentos descargados

Ya presentadas las tablas a utilizar para almacenar la información de los documentos tributarios, se conectan por medio del modelo estrella, siendo doc_datos_documento la tabla de hechos, y el resto correspondiendo a tablas de dimensiones.

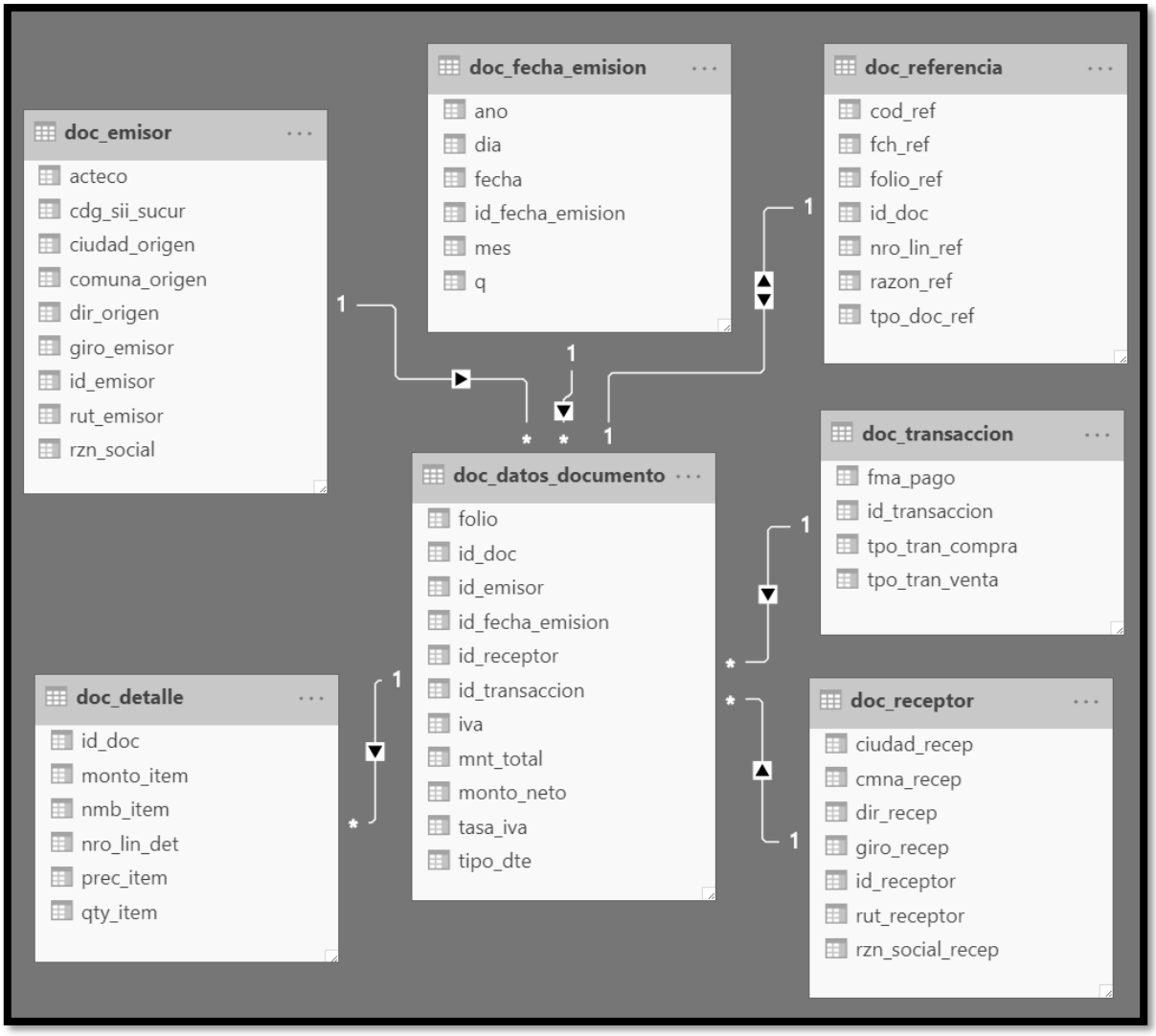

Figura 25: Modelo estrella para información de documentos descargados. Elaboración propia

4.3.3.3. Registro de descarga

El web scraping no es un método infalible, depende de la estabilidad de la página web en la que se desarrolla para su correcto funcionamiento. Por ello es necesario llevar un registro de cada descarga exitosa, al igual de las problemáticas para intentarlo nuevamente más adelante.

Se crea la tabla registro_descarga_xml, la cual contiene la siguiente estructura:

Columna	**Tipo**	**Descripción**
id	character varying(80)	Llave primaria con la estructura rut_origen_tipo
rut	character varying(20)	RUT del cliente
origen	character varying(10)	Origen de los documentos. Puede ser Emitido o recibido
tipo	character varying(2)	Tipo de DTE de documento
fecha_error	date NULL	Fecha en la que ocurrió un error de descarga
fecha_final_descarga	date	Fecha inicial de último periodo de descarga
fecha_inicial_descarga	date	Fecha final de último periodo de descarga
fecha_inicio	date	Fecha inicial de primera descarga historia para ese ID
folio_error	character varying(80)	Folio en que ocurrió un problema en una descarga. Por defecto se ingresa 0 si la descarga fue efectuada correctamente.

Tabla 15: Estructura de tabla registro_descarga_xml

Cada vez que se realiza una descarga, se identifica primero si, para ese cliente, alguna vez se ha realizado alguna descarga para ese tipo de DTE. De ser así, se evalúa si existió algún problema, lo cual se puede comprobar analizando la columna fecha_error o folio_error.[15] De haber un error, se modifica la fecha inicial de descarga para ese periodo para que se ajuste a la falla registrada.

El periodo se registra como exitoso únicamente cuando todas las descargas del periodo para un cliente fueron logradas positivamente. Si existe algún problema y debe repetirse el periodo, se intentará insertar datos a la base de datos que ya están registrados, por lo que se coloca la condición de no hacer nada en caso de encontrar conflicto con datos repetidos. Esto, además de solucionar el problema para descargas fallidas, soluciona el caso de posibles documentos duplicados en caso de que tanto

[15] La diferencia en el algoritmo de descarga para documentos emitidos y recibidos conlleva a que existan dos registros para descargas fallidas. En la descarga de documentos emitidos nos interesa el folio en el cual existió un error, mientras que para los documentos recibidos nos interesa la fecha.

se realice una descarga de datos tanto para emisor como receptor de un documento tributario.

4.3.4. Cuarta capa: Análisis de datos

Se crea un reporte en Power BI con los datos descargados para un segmento reducido de clientes. Descargar y trabajar con todos los datos de todos los clientes no aportaría al desarrollo del reporte.

Continuando con lo propuesto en el diseño de esta capa, se realizan reportes generales y específicos para los clientes, tanto para los documentos emitidos como para los recibidos. Adicionalmente, se agregan filtros dinámicos para que quien maneja el reporte pueda segmentar por mes y/o por tipo de documento tributario. Para su uso final, se agregará un filtro para que el cliente vea únicamente los datos que le corresponde, es decir, filtrar por su RUT para los documentos emitidos y recibidos. Igualmente se excluirán las facturas electrónicas que hayan sido anuladas por una nota de crédito.

Figura 26: Reporte general para clientes

Figura 27: Reporte específico para clientes

4.3.5. Quinta capa: Propuesta de servicio

Como ya se dijo antes, la quinta capa corresponde a las decisiones que tomará el cliente para su negocio en virtud de lo analizado en la cuarta capa. Una vez filtrado los datos para un cliente en específico, se tendrá una noción más clara de su negocio y de su comportamiento a lo largo del tiempo.

En base a esto, se puede tomar decisiones en base a los productos especificados en el reporte, tales como los más vendidos en ciertos meses, cuáles dan mayores ganancias en total, ver el comportamiento de los clientes más importantes, ver la evolución temporal del número de facturas electrónicas emitidas, cambios de comportamiento, entre otras cosas. Así el cliente puede anticiparse a la demanda en virtud del comportamiento que ha tenido su negocio, decidir dejar de trabajar con los productos menos vendidos y centrarse en los más demandados, ver las razones de anulaciones de facturas electrónicas, redireccionar sus recursos humanos hacia cierto tipo específico de clientes, y más.

Dado que esta capa es más práctica que las anteriores, más adelante se verán casos reales en la sección de "Pruebas y resultados", en donde se mostrará decisiones reales que un cliente podría tomar en base a la información entregada.

4.4. Automatización de proceso en servidor remoto

Si bien en este proyecto se trabaja con una base de datos correspondiente al periodo del 2018 para que los resultados durante múltiples descargas de prueba sean concisos, el producto final para la empresa se debe ejecutar diariamente para mantener la base de datos actualizada día a día. Por ello se especificará en esta

sección los pasos necesarios automatizar el proceso una vez el proyecto esté terminado.

En primer lugar, hay que considerar el periodo de descarga diario. Si se descargan los datos desde la última fecha de descarga exitosa registrada hasta el presente, existirá el problema de que se está asumiendo que los datos de ese día están listos, cuando no es siempre ese el caso. En caso de ejemplo, una descarga efectuada un 26 de diciembre a las dos de la tarde registrará ese día cómo último día de descarga exitosa; no obstante, un documento tributario podría ser emitido a las tres de la tarde, quedando fuera del registro de la empresa. Por ello se toma la decisión de descargar hasta el día previo al de descarga.

Ya solucionado ese inconveniente, viene el problema realmente importante: ¿cómo automatizar el proceso? Si estuviésemos en un Lambda de AWS, podríamos activar la ejecución por medio de CloudWatch Event, el cual permite colocar eventos como gatillantes de la aplicación. Pero, al estar en un servidor remoto, podemos utilizar un programa presente en Linux/Unix que comparte la esencia de CloudWatch Event: Cron.

Cron es el nombre de un programa que permite a usuarios de Linux/Unix ejecutar automáticamente comandos o scripts al definir una condición de ejecución con parámetros de tiempo. Al definir un Cron, se deben especificar siete parámetros específicos para gatillar una aplicación.

Campo	**Descripción**
Minuto	Controla el minuto de la hora en que el comando será ejecutado, este valor debe de estar entre 0 y 59.
Hora	Controla la hora en que el comando será ejecutado, se especifica en un formato de 24 horas, los valores deben estar entre 0 y 23, 0 es medianoche.
Día del Mes	Día del mes en que se quiere ejecutar el comando. Por ejemplo, se indicaría 20, para ejecutar el comando el día 20 del mes.
Mes	Mes en que el comando se ejecutará, puede ser indicado numéricamente (1-12), o por el nombre del mes en inglés, solo las tres primeras letras.
Día de la semana	Día en la semana en que se ejecutará el comando, puede ser numérico (0-7) o por el nombre del día en inglés, solo las tres primeras letras. (0 y 7 = domingo)
Usuario	Usuario que ejecuta el comando.
Comando	Comando, script o programa que se desea ejecutar. Este campo puede contener múltiples palabras y espacios.

Tabla 16: Campos necesarios de Cron en Linux/Unix. (González, 2019)

Normalmente bastaría con definir un Cron específico para un código en Linux para que este sea ejecutado cuando corresponda, pero en este caso en particular, debemos agregar un paso extra. Dado que nuestro código trabaja con una interfaz gráfica, debemos especificar en qué pantalla se ejecutará el proceso.

Para realizar esto, primero se identifica la pantalla de trabajo con el siguiente código:

```
echo $DISPLAY
```

Una vez obtenida esa información, se procese a definir el Cron

En este caso, &display_name corresponde al nombre de la pantalla de trabajo,

```
crontab -e

30 11 * * * DISPLAY=$display_name /usr/bin/python3
/home/ubuntu/Desktop/Daniel/download_emitidos.py
```

"/usr/bin/python3" corresponde a la dirección de python, y "/home/ubuntu/Desktop/Daniel/download_emitidos.py" corresponde a la dirección del código. La instalación para el código de los documentos recibidos es análoga. El programa se ejecutará a las 11:30 am cada día del año.

El último cambio por realizar será cambiar la lista fija de clientes por una consulta directa a la base de datos de la empresa y así tener este valor actualizado cada día.

5. Pruebas y resultados

5.1. Introducción

Tras el diseño y la implementación de la arquitectura de descarga y del modelo de datos, se procederá con la prueba del proceso completo para dos clientes, llegando al punto de colocarse en el puesto del cliente a la hora de ver la visualización de los datos para poder tomar decisiones que afecten su negocio.

La prueba se realizará para sólo dos clientes durante el periodo de 2018 que cumplan con los requisitos de que posean una gran variedad de documentos tributarios, tanto en el tipo de dte como en el número de estos. La implementación final con todos los clientes de la empresa, con los datos actualizados continuamente, será presentado a la empresa siguiendo los pasos ya planteados previamente.

Las pruebas se realizarán de manera remota. Se mostrará el volumen de datos recolectado y se visualizarán los datos en un dashboard personalizado para cada cliente.

5.2. Descarga de datos

Al estar la arquitectura de descarga ya consolidada, se tiene un código que realiza todo el proceso pertinente para obtener y almacenar los datos de los documentos tributarios. Antes de realizar la prueba para dos clientes, se selecciona de manera manual a dos clientes que posean varios documentos tributarios, tanto en tipo de dte como en volumen dentro de la categoría. Los clientes seleccionados presentan las siguientes características:

Tipo DTE	**Cliente_2**	**Cliente_3**
33	585	174
52	525	5
56	3	1
61	137	6

Tabla 17: Cantidad de documentos correspondientes a clientes

Ya teniendo los clientes de prueba, se debe limpiar la base de datos obtenida tras las pruebas de descarga durante el diseño e implementación de la arquitectura. Además de limpiar los datos y el registro de descarga, se reinicia el conteo de las llaves primarias que son autoincreméntales, siendo esto último más que nada por un tema de estética.

```
delete from doc_datos_documento;

delete from doc_detalle;

delete from doc_emisor;

delete from doc_receptor;

delete from doc_transaccion;

delete from doc_referencia;

delete from registro_descarga_xml;


ALTER SEQUENCE doc_datos_documento_id_doc_seq RESTART WITH 1;

ALTER SEQUENCE doc_emisor_id_emisor_seq RESTART WITH 1;

ALTER SEQUENCE doc_receptor_id_receptor_seq RESTART WITH 1;

ALTER SEQUENCE doc_transaccion_id_transaccion_seq RESTART WITH 1
```

Para ejecutar el programa en el servidor remoto, primero debemos dar permiso para acceder a la máquina por medio de un VNC. Esto se logra dando acceso al puerto 5901 directamente desde AWS. Tras ello podemos configurar la entrada del código en nuestro servidor remoto desde WinSCP, ingresar a nuestra máquina remota y ejecutar el código para los dos clientes seleccionados.

Origen	**Tiempo de descarga**
Emitido	6 min 19 seg
Recibido	14 min 32 seg

Tabla 18: Tiempo de ejecución de arquitectura de descarga en servidor remoto

5.3. Caso de análisis para clientes

5.3.1. Caso de análisis: cliente_2

Tabla 19: Reporte general como emisor para cliente_2

Comenzando con el reporte general como emisor, podemos ver que el mes de diciembre tuvo un ingreso mayor en comparación con el resto del año. Al ver las razones sociales de sus clientes durante ese mes, junto a su monto total, se puede apreciar que esto se debió a una venta importante a un cliente en específico.

Tabla 20: Reporte específico como emisor para cliente_2

Cuando vemos el reporte específico como emisor, podemos apreciar la basta cantidad de productos con los que trabaja la empresa, pero también podemos identificar un problema: los nombres de los productos no están estandarizados. Por ejemplo, la etiqueta asociada a dieciocho kilogramos aparece como: x18, x 18k, y x 18k. No obstante, el mismo producto con distinto nombre presenta el mismo comportamiento, por lo que no es un problema real para analizarlo.

En este caso, al ser los productos más vendidos comida para perros, asumiremos que no hay estacionalidad, o, mejor, podemos ver el comportamiento mes a mes y confirmar que no existe estacionalidad. En este caso en particular, no solo podemos aprender cuáles son los productos más vendidos, sino que también qué formato es el más preferido por sus clientes.

En este negocio en específico, podría tomar el producto estrella para comida de perro y hacer el contraste de comida para cachorro y comida para adulto, considerando que el cachorro terminará comiendo la otra categoría eventualmente. A partir de esto, podría estimar cuánta comida para adulto tendría que comprar más adelante. Por otro lado, podría filtrar sólo por comida de cachorro para dimensionar la variación en la venta de esta y así saber si el negocio está creciendo o no en cuanto a la cantidad de clientes, tomando como supuesto que el dueño del cachorro seguirá comprando la comida en este lugar.

5.3.2. Caso de análisis: cliente_3

Tabla 21: Reporte específico como emisor para cliente_3

Antes de comenzar con el análisis, nuevamente se evidencia un comportamiento esperable: duplicado de ítems por mal ingreso del producto a la hora de realizar el documento tributario. En ciertos meses aparece, por ejemplo, palta hass escrita tres veces: Palta hass, Paltas hass y polta hass. Si bien no están escritos iguales, uno puede analizar igualmente la información debido a que los tres productos de palta registrados tienen un comportamiento similar en el reporte, lo cual es entendible dado que son el mismo producto.

Tabla 22: Reporte general como emisor para cliente_3

Comenzando con el reporte general como emisor, podemos ver que en general los ingresos presentan una gran varianza, teniendo como mínimo $4.500.000 y como máximo $42.000.000. Llama la atención que durante el mes de Abril se obtuvo el ingreso más significante del año, pero por medio de sólo nueve facturas a cuatro empresas. Al indagar en el reporte específico como emisor para ese mes en particular, se puede ver que sólo se facturó por tres productos. Ordenados de manera decreciente por la cantidad del ítem sería: cosecha de nueces, proceso de despelonado y palta hass. Por otro lado, ordenado por el monto asociado, sería: proceso de despelonado, cosecha de nueces y palta hass.

Tabla 23: Reporte general como receptor para cliente_3

Si bien este parece a un gran mes, se puede ver en el reporte general como receptor que este también es el mes con mayor gasto. Esta conclusión es interesante para uno que es externo a la empresa, pero podemos creer con seguridad que este es un dato que el dueño de la empresa ya conocía, o al menos debería de conocer.

Más interesante para el dueño de la empresa sería cuantizar la cantidad de palta hass vendido durante cada mes para así poder anticiparse al próximo año. La palta hass es o el producto más vendido o el que genera un mayor ingreso en la mayoría de los meses, específicamente entre los meses de enero y marzo. Las nueces tienen su pick en abril, y los huevos en junio y julio. La plata regresa entre agosto y noviembre, y da paso nuevamente a los huevos en diciembre. Como persona externa al negocio, no puedo explicar el motivo de esta tendencia, al igual que no puedo decir con seguridad que es una tendencia; no obstante, puedo a partir de los datos cuantificar las ventas según de sus productos estrella para hacer una recomendación de qué esperar el próximo año. De estar en conjunto con el dueño del negocio, podría desentrañar de mejor manera los datos.

Por último, pero no menos importante, se pueden identificar los clientes que aportan un mayor ingreso a la empresa, de manera de tenerlos presente y hacer lo posible para no perderlos frente a la competencia.

6. Conclusiones

6.1. Resultados

El objetivo de este trabajo era el diseño e implementación de una arquitectura de descarga, el cual incluía un modelo de datos, que en conjunto permitiera a una empresa la descarga, transformación, carga, almacenamiento, procesamiento, análisis y visualización de datos desde documentos tributarios para poder entregar un reporte a sus clientes y así estos puedan tomar decisiones que afecte su negocio. Los datos serían rescatados desde la página web del SII por medio de web scraping y se consideraría un periodo fijo correspondiente al año 2018.

Se buscaba que el diseño de la arquitectura fuese escalable a medida que llegaran más clientes a la empresa, y que fuese robusto a los errores externos que pudiesen ocurrir durante el proceso de extracción de datos. También se buscaba que fuese eficiente, es decir, que hubiese mejoría en los tiempos de procesamiento en comparación con un caso sin un modelo de datos.

Teniendo estos objetivos como foco, y tras una investigación de herramientas disponibles, se diseñó una arquitectura que fuese capaz de extraer, transformar y almacenar información a partir de distintos documentos tributarios del SII, teniendo la habilidad de registrar el proceso, tanto en aciertos como en errores. Para ello, se aseguró durante el diseño de la arquitectura que las herramientas fuesen apropiadas para el volumen de datos a trabajar y así tener la confianza que el resultado final fuese eficiente en cuanto a los tiempos de procesamiento. Junto a esto, se diseñó un modelo de datos a partir del modelo tipo estrella que permitiera un uso más eficiente de los datos a la hora de realizar consultas y trabajar con ellos.

La arquitectura fue levantada en un servidor remoto que mejoraba los tiempos del proceso considerablemente, lo desligaba de posibles problemas relacionados con hardware o conexión local de internet, y daba la opción de poder hacer el sistema escalable separando el proceso de descarga en grupos de clientes y así ejecutar el programa en distintos servidores remotos de manera paralela. Esto, en conjunto con la correcta elección de herramientas y el modelo tipo estrella, daba como resultado una arquitectura eficiente a lo largo de sus capas.

Se creó un registro de descarga que podía identificar cuándo una descarga no fue lograda exitosamente, de manera que esta sección de datos era descargada en la siguiente iteración. Esto lo hacía robusto a errores externos al código, más dirigidos a complicaciones que podían nacer del proceso de web scraping en una página web.

Finalmente, la obtención de datos por medio de la arquitectura permitió realizar pruebas con clientes para crear un reporte personalizado para cada uno con el cual este pueda sacar información útil para su negocio. Se visualizó esta información para facilitar el análisis de los datos y se mostró posibles conclusiones que un cliente

podría realizar a partir de este reporte. La posibilidad para la empresa de poder sacar al mercado este servicio, en conjunto de la oportunidad de otros servicios al poseer información importante de los clientes, mejora su propuesta de valor hacia sus clientes al poder entregarles servicios que la competencia carece.

Al ver los datos, no se establece la realización del reporte para los clientes como el fin del proyecto, ya que los datos son un punto inicial para poder realizar más proyectos a futuro. Todo depende del uso que le dé la empresa con la cual se trabajó durante este proyecto.

6.2. Importancia del tamaño de datos

A lo largo de esta tesis, se ha mostrado los pasos a seguir para armar una arquitectura de cinco capas que permitiera recolectar información por medio de web scraping, transformarla y cargarla a una base de datos para finalmente poder analizarla. No obstante, este no fue la primera alternativa planteada.

Al comenzar el proyecto, la empresa dio la libertad de utilizar cualquier herramienta de AWS disponible para poder así entregar el mejor resultado posible. Esto se debía a que la empresa poseía varios de los servicios contratados y su uso para el proyecto significaría un costo despreciable para la empresa, en comparación con utilizar una herramienta externa. Además, hizo la recomendación de utilizar AWS Athena, una herramienta relativamente nueva destinada al Big Data.

Si bien el proyecto carecía del volumen de datos necesario para ser Big Data, tanto en el corto como largo plazo, era atrayente la idea de utilizar una herramienta poderosa que asegurara que el producto final mantuviera un buen rendimiento en todo momento.

Al investigar al respecto, se encontró que Athena era barata en comparación con otras herramientas similares. En comparación con otras herramientas, como Hadoop, por ser serverless no se necesita pagar por una instancia EC2 las 24 horas cada día, sino que se paga por la cantidad de TB de datos escaneados.

Al tener un buen desempeño con bases de datos de gran volumen, se asumió que no tendría problemas con el volumen de los datos deseados para el proyecto.

Para poder utilizar Athena, era necesario que la información descargada se transformara en JSON y fuese cargado a AWS S3 para su posterior análisis. Si bien existían otros formatos disponibles, se optó por JSON porque era más fácil el desarrollo del proyecto y cambiarlo más adelante no significaría un retraso mayor en caso de encontrar un formato con mayor eficiencia.

Por otro lado, utilizando el mismo argumento que se tenía libertad para utilizar cualquier herramienta de AWS, se procedió a elegir DynamoDB para llevar el registro de descarga del proceso, nuevamente por asegurar un buen desempeño al tratar con volúmenes grandes de datos.

De esta manera, la arquitectura original para el proyecto tenía la siguiente forma:

1.- Fuente de datos: Plataforma del SII

2.- Extracción, transformación y carga de datos:

- Extracción: web scraping en máquina remota
- Transformación: JSON
- Carga: S3

3.- Almacenamiento de datos: DynamoDB

4.- Análisis de datos: Athena

5.- Parte de negocio beneficiado: dashboard personalizado para la empresa

Uno de los puntos importantes de esta arquitectura, es que existen varias prácticas que se pueden implementar en Athena para hacer las consultas más eficientes, tal como se puede ver en "Top 10 Performance Tuning Tips for Amazon Athena" (Hocanin, 2017). Por ello, en un inicio la parte de optimización de las consultas tenía un rol importante en la planificación.

Al realizar las primeras pruebas de análisis de datos, se encontró que una consulta para los datos de un único cliente para el periodo de un año tardaba entre 1.5 y 2.5 segundos por consulta. Al realizar una de las prácticas encontradas, este tiempo logró ser reducido en medio segundo, lo cual sigue dando un tiempo considerable para una consulta de este tipo.

Aún en el caso más optimista, en donde una consulta tardara un segundo por cliente, significaba un resultado no consistente para el volumen de datos tratados, hacía más difícil la idea de que el proyecto fuese escalable y, no menos importante, mostraba directamente que la arquitectura no era óptima.

Al investigar más al respecto, se encontró que la hipótesis original estaba errada: el que una herramienta sea eficiente para un volumen de datos considerable, no significa que presentará el mismo rendimiento para volúmenes pequeños.

Para empezar, se encontró que el tamaño mínimo para que una consulta en Athena sea óptima debe ser de por lo menos 128 MB. Dentro de las prácticas para hacer las consultas óptimas, se encontraba comprimir archivos pequeños en un paquete de mayor tamaño y así llegar a este valor óptimo, o directamente hacer un archivo grande cuya información se constituyese de la información de los archivos pequeños. No obstante, de seguir esa práctica con los valores utilizados en el proyecto, no se lograba hacer más que 2 MB de información.

Se hubiesen podido haber implementado varias prácticas para hacer las consultas óptimas, tales como particionar la información, comprimir los archivos, trabajar en la consulta en específico, optimizar uniones de tablas en las consultas, entre otras cosas; pero incluso con todo ello, difícilmente se llegaría a tiempos razonables para las consultas.

Dado este mayúsculo percance, se debió retroceder a la etapa de diseño de la arquitectura y elegir herramientas que pudiesen trabajar de manera eficiente con los datos necesarios, esta vez asegurándose que la herramienta dará buenos resultados para el volumen de datos planteados.

Para ello, además de investigar en foros y leer papers, se aprovechó una de las ventajas que tiene tener contratado AWS: poder hablar directamente con alguien de soporte de Amazon. De esta manera se comprobó de la fuente directa que esta vez las herramientas eran adecuadas para el proyecto.

Como ya se mostró antes, se terminó escogiendo PostgreSQL para cargar, almacenar y analizar los datos, por lo que se prescindió de S3, ya que no era necesario guardar el archivo; y de DynamoDB, únicamente porque era una herramienta innecesariamente poderosa para el proyecto. Al prescindir de S3, se hizo más rápido el proceso de descarga de carga de información.

De esta manera, la arquitectura final utilizada en esa memoria es la siguiente:

1.- Fuente de datos: Plataforma del SII

2.- Extracción, transformación y carga de datos:

- Extracción: web scraping en máquina remota
- Transformación: JSON
- Carga: PostgreSQL

3.- Almacenamiento de datos: PostgreSQL

4.- Análisis de datos: Power BI

5.- Parte de negocio beneficiado: dashboard personalizado para la empresa

6.3. Futuras líneas de trabajo en empresa

Después del diseño e implementación que se hizo de este proyecto, además de las pruebas realizadas, existen futuras líneas de trabajo para desarrollar. Algunas ya fueron mencionadas en el pasado, mientras que otras nacieron a partir de los datos que lograron ser almacenados.

- Automatización del proceso de descarga de manera diaria para todos los clientes. Como ya se dijo, para el desarrollo de este proyecto, se trabajó un periodo de tiempo acotado y se descargó datos de clientes específicos que cumplieran ciertas características tal que fuesen buenos casos de prueba. No obstante, se explicó los pasos necesarios para automatizar el proceso por medio de la implementación de un Cron.
- Identificar clientes con un mayor valor futuro para la empresa. Teniendo los datos de las empresas clientes, podemos analizar cuáles presentan una mayor retención, además de ver cuáles son las más propensas a abandonar el servicio. Igualmente podemos ver qué tipo de clientes lo utiliza más y/o

contrata más máquinas durante un periodo. Si se logra identificar estos segmentos, se puede direccionar a los vendedores de la empresa para que intenten vender más en el segmento que más lo utiliza, y menos en el segmento de clientes que son más propensos a abandonar.

- Hacer el proceso headless. Si se logra que el proceso de web scraping funcione sin una interfaz gráfica, en teoría debería mejorar los tiempos. Esto no se logró hacer dado que trabajar con un certificado digital es incompatible con Google Chrome de manera headless. Se buscó mucha información al respecto y no se logró encontrar algo que solucione este problema.
- Pronóstico de ventas o análisis de tenencia a partir de una base de datos completa con los datos de los clientes. Un correcto análisis de datos por parte de la empresa le permitiría ofrecer más productos a sus clientes, aumentar su tasa de retención y captar más clientes al tener una mejor propuesta de valor. En base a esto, se pueden implementar estrategias de retención, cross selling o fidelización de las empresas cliente en base a la evolución de sus ventas u otros indicadores.
- Creación de datamart. Luego de que la empresa ejecute la arquitectura de descarga para todos sus clientes, tomando un horizonte de tiempo adecuado, se tendrá una base de datos desde la cual se pueden crear datamarts orientados a un área del negocio específico. En este caso, se podrían realizar datamarts con datos específicos de clientes para entregarles un reporte mensual con la información correspondiente del periodo anterior. Esto podría transformarse en un servicio adicional entregado por la empresa. En virtud de que la empresa ya entrega actualmente un reporte mensual con la información de la cantidad de boletas emitidas, el cual es de interés para los clientes, la idea de un reporte mensual con la información de sus documentos tributarios debería estar entre las primeras cosas que la empresa debería hacer con la información. Se recalca que un reporte se diferencia del dashboard ya planteado a que un reporte presenta información más directa y menos interactiva.
- Aplicar modelo de datos a otras bases en la empresa. La empresa en donde se realizó el proyecto tiene varias bases de datos destinadas a distintos procesos internos de la empresa. No obstante, estos carecen de un modelo de datos más allá de estar relacionadas con llaves foráneas, llaves primarias y secundarias. Si bien, dada la magnitud de los datos que maneja la empresa, es descabellado aplicar un modelo tipo estrella como el utilizado en esta arquitectura, debería ser considerada para bases de datos futuras para que tengas un acceso a esta información de una manera óptima en comparación con la estructura actual.

6.4. Alternativas a AWS

La arquitectura de descarga fue desarrollada en su totalidad con herramientas de AWS, exceptuando el análisis de datos en Power BI. Esto se debió primordialmente a que la empresa en donde se desarrolló el proyecto está comprometida con este ecosistema y trabajar con estas herramientas significaría un menor costo en

comparación con contratar otras ajenas. Como ya se vio antes en el estado del arte, se investigó distintas herramientas de AWS para poder realizar el proyecto, dejando de lado otras alternativas que igualmente hubiesen podido solucionar las distintas problemáticas a tratar.

Por lo que se vio en la empresa, esta no migrará de AWS por el costo operacional que esto significaría. No obstante, a continuación se mostrarán alternativas ajenas a este ecosistema en caso que alguien desee implementar una arquitectura de descarga similar sin utilizar AWS.

6.4.1. Google Cloud

Se trata de una agrupación de servicios de cloud computing que ofrecen soluciones unificadas para el trabajo en la nube lanzado por Google el 2008. A diferencia de AWS, presenta menos servicios que AWS. Estas son: Google App Engine, para alojamiento web con ejecución de aplicaciones; Google Cloud Storage, para almacenamiento en la nube; Google BigQuery, para el análisis de Big Data en tiempo real; Google Cloud Datastore, para almacenar datos NoSQL; Google Cloud SQL, para el uso de bases de datos relacionales; Google Prediction API, para la ejecución de algoritmos de machine learning; y Google Translate API, para el diseño de aplicaciones multilenguaje. (Antoniony, 2019)

A diferencia de AWS, no saca al mercado nuevos servicios de manera constante, sin embargo, estas presentan un alto rendimiento en Big Data, análisis y aprendizaje automático. (PowerData, 2019)

Según Brismark Antoniony en “Alternativas a Amazon Web Services”, sus ventajas frente a AWS son:

- Cobertura y seguridad: la infraestructura de Google cuenta con la mejor encriptación y está muy bien interconectada entre las distintas regiones, contando con respaldo en diferentes partes del mundo. Se trata de un servicio en la nube con un grado muy alto de seguridad, mayor que el de AWS, que puede hacer caer la balanza en su favor a la hora de elegir el proveedor cloud.
- Velocidad de acceso y respaldo: los servicios Google tienen un tiempo de arranque y reinicio muy superiores al de otras plataformas en la nube. Además, Google Cloud Storage Nearline (servicio de almacenamiento de datos en la nube) es extremadamente rápido y permite una restauración rápida de datos. Este servicio tiene un bajo costo y con tarifas definidas por GB/mes (sistema claro que permite calcular el costo por transferencia). Esto hace que muchas empresas utilicen este servicio como su único método de almacenamiento.
- Personalización del hardware: las máquinas virtuales contratadas en Google permiten configurar la combinación adecuada de CPU y memoria virtual. Este hecho da una gran ventaja a la hora de planificar el tipo de máquinas que se necesitan y cuáles serán los requisitos necesarios, ahorrando costos. Además,

en cualquier momento puede aumentarse esta contratación en caso de necesitar más capacidad de proceso, memoria, almacenaje, etc.
- Gran carga de usuarios: los servicios Google Cloud utilizan el mismo sistema que otros productos Google, tales como Google Maps o Gmail, para gestionar la conexión de los clientes con la infraestructura. Esto permite manejar grandes cantidades de tráfico de usuarios, así como altos picos de conexiones, con gran fiabilidad.

6.4.2. Microsoft Azure

Es un servicio de computación en la nube creado por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos. Azure fue anunciado en octubre de 2008, comenzó con el nombre en clave "Project Red Dog" y publicado el 1 de febrero de 2010 como "Windows Azure" antes de ser rebautizado como "Microsoft Azure" el 25 de marzo de 2014. (Wikipedia, 2020)

Según Brismark Antoniony, sus ventajas frente a AWS son:

- Microsoft: detrás de Azure hay una gran empresa como Microsoft, que apuesta por la innovación y el desarrollo. La fiabilidad de una empresa como Microsoft, con tantos años de experiencia, puede hacer que la decisión de contratar sus servicios caiga de su lado.
- Abierta: Azure está abierta a todos los lenguajes de programación ya que tiene un compromiso con el código abierto para la creación de soluciones.
- Entorno híbrido: permite trabajar tanto en local como en la nube para adaptarse a las necesidades de la empresa. Así es posible que la empresa, dadas sus necesidades, requiera de una nube híbrida donde algunos procesos se realicen en local y otros en remoto o la en la nube.
- Seguridad: Microsoft integra en su red controles de seguridad multinivel frente a amenazas que permiten una rápida reacción y protección. Los datacenter de Microsoft Azure poseen protección, tanto de los datos almacenados, como de los procesos que se llevan a cabo en ellos.
- Pago por uso: el servicio de Microsoft Azure cobra sólo los productos que se están utilizando. De esta forma, cada empresa puede adaptar los productos contratados a sus necesidades, ahorrando el costo de aquellos que no necesite y que por lo tanto no contrate.
- Continuidad y alcance: los servicios Azure se ejecutan en una gran red mundial, con centros de datos en 54 regiones y 140 países (es la nube con más regiones globales). Esto permite una continuidad en el servicio y un mayor alcance de este, que garantiza poder acceder a los servicios desde cualquier lugar y en cualquier momento.

6.4.3. Comparación con AWS

A continuación, se muestra una comparativa de distintas herramientas para un mismo servicio. Se consideran las herramientas de AWS utilizadas en la

arquitectura, además de S3 en caso de que se desee almacenar los archivos descargados durante el proceso.

AWS	**Google Cloud**	**Microsoft Azure**	**Descripción**
EC2	Compute Engine	Virtual Machines	Capacidad de cálculo segura y redimensionable en la nube.
Aurora	Google Cloud SQL	Microsoft Azure SQL Database	Servicio de base de datos relacional.
S3	Google Cloud Storage	Azure Storage	Almacenamiento de datos den la nube.

Tabla 24: Comparación de herramientas de servicios de computación en la nube

7. Bibliografía

Amazon AWS. (3 de Octubre de 2019). *¿Qué es Amazon Redshift?* Obtenido de Amazon Athena: https://docs.aws.amazon.com/es_es/redshift/latest/mgmt/welcome.html

Amazon AWS. (3 de Octubre de 2019). *Amazon Athena*. Obtenido de Amazon AWS: https://translate.google.com/translate?hl=es-419&sl=en&u=https://aws.amazon.com/athena/&prev=search

Amazon AWS. (3 de Octubre de 2019). *Amazon DynamoDB*. Obtenido de Amazon AWS: https://aws.amazon.com/es/dynamodb/

Amazon AWS. (13 de Octubre de 2019). *Amazon EC2*. Obtenido de Amazon AWS: https://aws.amazon.com/es/ec2/

Amazon AWS. (3 de Octubre de 2019). *Amazon Redshift*. Obtenido de Amazon AWS: https://aws.amazon.com/es/redshift/

Amazon AWS. (6 de Septimebre de 2019). *Amazon S3*. Obtenido de Amazon AWS Web site: https://aws.amazon.com/es/s3/

Amazon AWS. (5 de Septiembre de 2019). *Infraestructura global: La infraestructura global en la nube más amplia, fiable y segura.* Obtenido de aws.amazon.com: https://aws.amazon.com/es/about-aws/global-infrastructure/

Amazon AWS. (8 de 10 de 2019). *Instancias M5 de Amazon EC2*. Obtenido de Amazon AWS: https://aws.amazon.com/es/ec2/instance-types/m5/

Amazon AWS. (13 de Octubre de 2019). *Uso de Amazon Aurora PostgreSQL*. Obtenido de Amazon AWS: https://docs.aws.amazon.com/es_es/AmazonRDS/latest/AuroraUserGuide/Aurora.AuroraPostgreSQL.html

Antoniony, B. (5 de Diciembre de 2019). *Alternativas a Amazon Web Services*. Obtenido de Ambit: https://www.ambit-bst.com/blog/alternativas-a-amazon-web-services

AWS, S. d. (9 de Septiembre de 2019). Regarding case 6428708211. (D. Espinoza, Entrevistador)

Brandom, R. (2019 de Mayo de 2019). *Mapping out Amazon's invisible server empire*. Obtenido de The Verge Web site: https://www.theverge.com/2019/5/10/18563485/amazon-web-services-internet-location-map-data-center

Código Facilito. (3 de Octubre de 2019). *Introducción A PostrgeSQL*. Obtenido de Código Facilito: https://codigofacilito.com/articulos/tutorial-postgresql

Definición. (12 de Octubre de 2019). *Definición de Máquina POS*. Obtenido de Definición: https://definicion.mx/maquina-pos/

González, S. (26 de Diciembre de 2019). *Manual básico de CRON*. Obtenido de LINUXTOTAL: https://www.linuxtotal.com.mx/?cont=info_admon_006

Hocanin, M. C. (2017). Top 10 Performance Tuning Tips for Amazon Athena. *AWS Big Data Blog*.

Lieter Plasencia Moreno, C. A. (2017). Arquitectura referencial de Big Data para la gestión de las telecomunicaciones. *Ingeniare. Revista chilena de ingeniería, vol. 25 N° 4*, 566-577.

Microsoft. (13 de Octubre de 2019). *Power BI*. Obtenido de Microsoft: https://powerbi.microsoft.com/es-es/

Mutai, J. (5 de Diciembre de 2018). *How to Install and Configure VNC Server on Ubuntu 18.04 LTS*. Obtenido de computingforgeeks: https://computingforgeeks.com/how-to-install-vnc-server-on-ubuntu-18-04-lts/

PowerData. (25 de Junio de 2019). *Google cloud vs AWS. Comparativa, pros y contras*. Obtenido de PowerData: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/google-cloud-vs-aws.-comparativa-pros-y-contras

Requests: HTTP para Humanos. (5 de Septiembre de 2019). *Requests: HTTP para Humanos*. Obtenido de Requests: HTTP para Humanos: https://2.python-requests.org//es/latest/#

SII. (13 de Octubre de 2019). *FACTURA ELECTRONICA MIPYME*. Obtenido de SII: http://www.sii.cl/factura_electronica/facturamipyme.htm

Wikipedia. (5 de Septiembre de 2019). *Amazon Web Services*. Obtenido de wikipedia.org: https://es.wikipedia.org/wiki/Amazon_Web_Services

Wikipedia. (16 de Octubre de 2019). *Cygwin*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Cygwin

Wikipedia. (21 de Diceimbre de 2019). *Esquema de copo de nieve*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Esquema_en_copo_de_nieve

Wikipedia. (21 de Diciembre de 2019). *Esquema en estrella*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Esquema_en_estrella

Wikipedia. (5 de Septiembre de 2019). *Google Chrome*. Obtenido de wikipedia.org: https://es.wikipedia.org/wiki/Google_Chrome

Wikipedia. (5 de Septiembre de 2019). *Mozilla Firefox*. Obtenido de wikipedia.org: https://es.wikipedia.org/wiki/Mozilla_Firefox

Wikipedia. (5 de Septiembre de 2019). *PhantomJS*. Obtenido de wikipedia.org: https://en.m.wikipedia.org/wiki/PhantomJS

Wikipedia. (8 de 10 de 2019). *VNC*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/VNC

Wikipedia. (Abril de 2020). *Microsoft Azure*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Microsoft_Azure

Xfce Development Team. (2012 de Mayo de 2012). *Xfce Development Team*. Obtenido de http://cdn.xfce.org/about/screenshots/4.10-1.png

Xfce Team. (31 de Agosto de 2008). *Xfce Team*. Obtenido de http://www.xfce.org/download#artwork