Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ciencias Matemáticas
Escuela Profesional de Computación Científica

Automatización de procesos para el registro de facturas en formato XML en una base de datos mediante Python

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Licenciada en Computación Científica

AUTOR

Leyla Katiuska MARTEL SOLIS

ASESOR

Mg. Luis Javier VASQUEZ SERPA

Lima, Perú

2021

Referencia bibliográfica

Martel, L. (2021). *Automatización de procesos para el registro de facturas en formato XML en una base de datos mediante Python*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ciencias Matemáticas, Escuela Profesional de Computación Científica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Leyla Katiuska Martel Solis
Tipo de documento de identidad	DNI
Número de documento de identidad	76000628
URL de ORCID	NO APLICA
Datos de asesor	
Nombres y apellidos	Mg. Luis Javier Vásquez Serpa
Tipo de documento de identidad	DNI
Número de documento de identidad	43389380
URL de ORCID	https://orcid.org/0000-0002-5414-6764
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Dr. María Natividad Zegarra Garay
Tipo de documento	DNI
Número de documento de identidad	09206994
Miembro del jurado 1	
Nombres y apellidos	Mg. Carlos Augusto Ruiz De La Cruz Melo
Tipo de documento	DNI
Número de documento de identidad	08249640
Miembro del jurado 2	
Nombres y apellidos	Mg. Luis Javier Vásquez Serpa
Tipo de documento	DNI
Número de documento de identidad	43389380
Datos de investigación	
Línea de investigación	NO APLICA

Grupo de investigación	NO APLICA
Agencia de financiamiento	SIN FINANCIAMIENTO
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Callao Distrito: Ventanilla Latitud: -11.832008 Longitud: -77.142221
Año o rango de años en que se realizó la investigación	2020 - 2021
URL de disciplinas OCDE	Ciencias de la computación https://purl.org/pe-repo/ocde/ford#1.02.01

ACTA DE SUSTENTACIÓN DEL INFORME DE TRABAJO DE SUFICIENCIA PROFESIONAL EN LA MODALIDAD VIRTUAL PARA LA OBTENCIÓN DEL TÍTULO PROFESIONAL DE LICENCIADO(A) EN COMPUTACIÓN CIENTÍFICA (PROGRAMA DE TITULACIÓN PROFESIONAL 2021-I)

En la UNMSM – Ciudad Universitaria – Facultad de Ciencias Matemáticas, siendo las 19:20 horas del jueves 21 de octubre del 2021, se reunieron los docentes designados como Miembros del Jurado Evaluador (PROGRAMA DE TITULACIÓN PROFESIONAL 2021-I): Dra. María Natividad Zegarra Garay (PRESIDENTE), Mg. Carlos Augusto Ruiz De La Cruz Melo (MIEMBRO) y el Mg. Luis Javier Vásquez Serpa (MIEMBRO ASESOR), para la sustentación del Informe de Trabajo de Suficiencia Profesional titulado: "**AUTOMATIZACIÓN DE PROCESOS PARA EL REGISTRO DE FACTURAS EN FORMATO XML, EN UNA BASE DE DATOS MEDIANTE PYTHON**", presentado por la señorita **Bachiller Leyla Katiuska Martel Solis**, para optar el Título Profesional de Licenciada en Computación Científica.

Luego de la exposición del Informe de Trabajo de suficiencia Profesional, la Presidente invitó a la expositora a dar respuesta a las preguntas formuladas.

Realizada la evaluación correspondiente por los Miembros del Jurado Evaluador, la expositora mereció la aprobación **bueno,** con un calificativo promedio de **dieciséis (16)**.

A continuación, los Miembros del Jurado Evaluador dan manifiesto que la participante **Bachiller Leyla Katiuska Martel Solis**, en vista de haber aprobado la sustentación del Informe de Trabajo de Suficiencia Profesional, será propuesta para que se le otorgue el Título Profesional de Licenciada en Computación Científica.

Siendo las 20:00 horas se levantó la sesión firmando para constancia la presente Acta.

UNMSM
Firmado digitalmente por ZEGARRA GARAY Maria Natividad FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 03.11.2021 17:28:36 -05:00

Dra. María Natividad Zegarra Garay
PRESIDENTE

Mg. Carlos Augusto Ruiz De La Cruz Melo
MIEMBRO

Mg. Luis Javier Vásquez Serpa
MIEMBRO ASESOR

“Los grandes algoritmos son la poesía de la computación”.

Francis Sullivan

Dedicatoria

A mis padres, Barbara y Walter, por su apoyo

incondicional y amor.

A mi hermana, María Fernanda, por su gran compañía.

A mi novio, Jonatan por ser un gran compañero.

Agradecimientos

A mi alma mater, UNMSM, por haberme permitido formarme en una carrera profesional que me apasiona.

A mi Facultad de Ciencias Matemáticas, por enseñarme una nueva visión de las cosas, a mis maestros, por brindarme sus conocimientos, a mis amigos por su grata compañía y experiencias.

A mi asesor, Mg. Luis Javier Vásquez Serpa por su orientación y apoyo para poder desarrollar este trabajo.

A mi familia, por creer siempre en mí y ser mi principal motivación para llegar muy lejos en mi carrera profesional y vida.

Resumen

En el presente trabajo, se presenta una propuesta de automatización de procesos para el Registro de Facturas en una Base de Datos, orientada hacia aquellas empresas en donde reciben de manera masiva sus facturas electrónicas en formato XML a través del correo, por lo que se desarrollará un Bot que realice este proceso repetitivo con la finalidad de obtener mayor productividad, logrando de esta manera beneficios tanto en tiempo como en costos para dicha empresa. Para ello se inició con el análisis, estudio y definición del proceso en donde se determina si es posible la automatización, por lo cual se realiza un diagrama de Flujo que muestre la lógica del Bot, así como también se indica las actividades de Casos de Uso que detallan procesos internos más específicos. Finalmente, se emplea el lenguaje de programación de código abierto como Python, usando las diferentes librerías para poder manejar los archivos y aplicativos, SQLite como Base de Datos a fin de almacenar la información de las Facturas y para la implementación del Bot se utilizará el Programador de Tareas de Windows con el propósito de que se ejecute de manera automática tanto de forma diaria o cuando se requiera.

***Palabras claves*:** XML, SQLite, Bot, Python, Automatización, Casos de Uso.

Abstract

In this work, a proposal for the automation of processes for Invoices Registrations in a Database is presented, focused on those companies which receive massively electronic invoices in XML format through the mail, therefore a Bot will be developed to perform this repetitive process in order to obtain greater productivity, thus achieving benefits in both time and costs. For this, it began with the analysis, study and definition of the process where it is determined if automation is possible, for which a Flow diagram is made to show the logic of the Bot, as well as the activities of Use Cases which detail more specific internal processes. Finally, the open-source programming language such as Python is used, using the different libraries to be able to handle the files and applications, SQLite as a Database in order to store the information of the Invoices and for the implementation of the Bot the Programmer will be used Windows Tasks so that it runs automatically both on daily or as needed.

***Keywords*:** XML, SQLite, Bot, Python, Automation, Use Cases.

Tabla de Contenido

RESUMEN ..... **IV**

ABSTRACT ..... **V**

TABLA DE CONTENIDO ..... **VI**

CAPÍTULO 1 ..... **1**

INTRODUCCIÓN **1**

CAPÍTULO 2 ..... **2**

INFORMACIÓN DEL LUGAR DONDE SE DESARROLLÓ LA ACTIVIDAD **2**

2.1. INSTITUCIÓN DONDE SE DESARROLLÓ LA ACTIVIDAD 2

2.2. PERIODO DE DURACIÓN DE LA ACTIVIDAD 2

2.3. FINALIDAD Y OBJETIVOS DE LA ENTIDAD 2

2.4. RAZÓN SOCIAL 2

CAPÍTULO 3 ..... **3**

DESCRIPCIÓN DE LA ACTIVIDAD **3**

3.1. PRESENTACIÓN DE LA ACTIVIDAD 3

3.1.1. Formulación del Problema *3*

3.1.2. Objetivo General *3*

3.1.3. Objetivos Específicos *3*

3.1.4. Alcances y Limitaciones *3*

3.1.5. Motivación del Proyecto *4*

3.1.6. Metodología *4*

3.2. SUSTENTACIÓN TEÓRICA 5

3.2.1. ¿Qué es un Bot? *5*

3.2.2. Factura Electrónica *6*

3.2.3. Lenguaje de Programación: Python *10*

3.2.4. Base de datos: SQLite *12*

3.2.5. Programador de Tareas *12*

3.2.6. Archivos XML (Extensible Markup Language) *13*

3.2.7. Microsoft Office Excel *16*

3.2.8. Archivos Batch o Bat *17*

3.2.9. Diagramas de Casos de Uso *17*

3.3. PROPUESTA, ANÁLISIS Y DEFINICIÓN DEL PROCESO 18

3.3.1. Propuesta *18*

3.3.2. Análisis y Definición de Proceso *19*

3.4. DESARROLLO DEL BOT 22

3.4.1. Casos de Uso *22*

3.4.2. Actividades de Casos de Uso *22*

3.5. IMPLEMENTACIÓN DEL BOT 26

3.5.1. Crear archivo Bat *26*

3.5.2. Configuración del Programador de Tareas *26*

CAPÍTULO 4 ..... **31**
CONCLUSIONES **31**
CAPÍTULO 5 ..... **32**
RECOMENDACIONES **32**
CAPÍTULO 6 ..... **33**
BIBLIOGRAFÍA **33**
CAPÍTULO 7 ..... **35**
ANEXOS **35**
7.1. CÓDIGO FUENTE 35

Capítulo 1

INTRODUCCIÓN

Actualmente nos encontramos en la era de transformaciones digitales que proporcionan tecnologías emergentes con grandes beneficios para las empresas, en este caso nos referimos a la Automatización de procesos mediante Bots**,** herramienta ideal para aquellos procesos en donde un empleado invierte tiempo y esfuerzo en realizar tareas repetitivas en un computador.

Un Bot es un software que trabaja como si lo haría un humano, con la diferencia de que este lo haría con mayor rapidez, con un gran volumen de datos y sin errores, liberando el tiempo del personal, para que se pueda enfocar en actividades que requieran un análisis, criterio o tareas más complejas. Es por ello que, en el presente trabajo se propone realizar un Bot enfocado en almacenar Facturas en una Base de Datos, la organización del trabajo es como sigue:

En el capítulo 1, se muestra esta sección.

En el capítulo 2, se muestra la información referente del lugar donde se desarrolló la actividad como proyecto.

En el capítulo 3, se muestra la descripción de la actividad, así como la presentación de esta, el sustento teórico, el análisis y desarrollo de la actividad

En el capítulo 4 y 5, se exponen las conclusiones sobre el Bot y las recomendaciones para trabajos futuros como continuidad, respectivamente.

En el capítulo 6, se muestra la biografía utilizada para el presente trabajo.

Por último, en el capítulo 7, se encuentra el Anexo que muestra el código fuente realizado a través de Python para el Desarrollo del Bot.

Capítulo 2

INFORMACIÓN DEL LUGAR DONDE SE DESARROLLÓ LA ACTIVIDAD

2.1. Institución donde se desarrolló la actividad

Actualmente me encuentro laborando en la empresa DT-DATA como Desarrolladora en Automatización Robótica de Procesos donde tengo asignado labores como el desarrollo de robots de software en base al análisis y estudio de un determinado proceso que contiene tareas repetitivas, construidas a través de herramientas especializadas que tienen un costo de licencia de miles de dólares al año, en base a mi experiencia al desarrollar estos robots, el que tiene más demanda entre los clientes, es el proceso de Registro de Facturas en formato XML al respectivo sistema contable de su empresa, es por ello que nació esta iniciativa de poder crear un robot que pudiera realizar este proceso pero a través de un lenguaje de programación de código abierto.

2.2. Periodo de duración de la actividad

El periodo de desarrollo del presente trabajo fue de 1 mes aproximadamente, dedicándole 8 horas a la semana al final del día, iniciando con el análisis del del proyecto, para posteriormente realizar la codificación e implementación.

2.3. Finalidad y Objetivos de la entidad

DT-DATA está compuesta por un conjunto de profesionales altamente capacitados que ayuda a las organizaciones a ser más eficientes a través de la implantación, desarrollo y formación de soluciones tecnológicas líderes con el objetivo de mejorar su competitividad y productividad, agilizar la correcta toma de decisiones, incrementar las ventas y abrir nuevos mercados de nuestros clientes.

2.4. Razón Social

METATRON TECHNOLOGY PARTNERS S.A.C.

Capítulo 3

DESCRIPCIÓN DE LA ACTIVIDAD

3.1. Presentación de la actividad

3.1.1. Formulación del Problema

Actualmente en muchas empresas existen procesos que son repetitivos entre humano y computador, en este caso en particular tenemos el proceso de Registro de Facturas que se realiza de manera manual a su respectivo sistema contable, que demanda mucho tiempo y esfuerzo del personal de la empresa.

3.1.2. Objetivo General

Proponer y desarrollar un Bot que pueda mejorar el proceso de Registro de Facturas en una Base de Datos, basándose en reglas establecidas, trabajando mucho más rápido, a gran volumen y omitiendo errores humanos, teniendo como ventajas en costo y tiempo en una determinada empresa.

3.1.3. Objetivos Específicos

- Estudiar y analizar el Proceso de Registro de Facturas.
- Desarrollar un Bot mediante el lenguaje de programación de Python que pueda realizar el proceso de Registro de Facturas almacenándolas directamente a una base de datos.
- Implementar un Bot que no necesite la intervención humana.

3.1.4. Alcances y Limitaciones

Este proyecto de creación de un Bot está enfocado, para aquellas empresas digitalizadas, que empleen una Base de datos y que realicen sus compras a través de Facturas electrónicas en formato XML debido a que estos son los insumos principales para que el Bot pueda realizar el registro de estas.

Nos limitaremos debido al tiempo al emplear SQLite por su fácil configuración, uso y sencillez que nos proporciona; ya que hay que tener en cuenta que para una empresa se hace uso de un sistema gestor de base de datos más robusta.

3.1.5. Motivación del Proyecto

La idea de desarrollar un Bot que abarque el proceso de registro de facturas en formato XML surgió por experiencia personal, debido a que, como Desarrolladora en Automatización Robótica de Procesos, este proceso en particular es muy demandado por las empresas, sin embargo estos Bots son desarrollados a través de una herramienta especializada que tiene un costo de licencia de miles de dólares al año, y lo que se pretende en este trabajo es realizar e implementar un Bot a través de un lenguaje de programación de código abierto, dejando para trabajos futuros la realización de un aplicativo que se adecue a los sistemas que pueden usar diferentes empresas.

3.1.6. Metodología

La investigación del trabajo será de tipo bibliográfica porque utilizará fuentes como la documentación oficial de las herramientas a usar para el marco teórico y posteriormente para la construcción del Bot propuesto.

La investigación del trabajo tendrá una modalidad aplicada porque para la construcción del Bot de registro de facturas en una base de datos se aplicará los conocimientos adquiridos durante la carrera y el trabajo

3.2. Sustentación Teórica

3.2.1. ¿Qué es un Bot?

Según Deloitte (2017), "un Bot es un software que puede ejecutar tareas repetitivas. Se programa mediante un lenguaje de programación sencillo o bien, cuenta con una opción para grabar las acciones de un usuario, como lo son el copiar, pegar o realizar consultas a bases de datos, para luego ejecutarlas con base en un calendario establecido."

Según IONOS (2000) el termino Bot deriva de la palabra robot, estos están programados para realizar tareas concretas y repetitivas a través de comandos e instrucciones bien definidas en forma de algoritmos desarrollado mediante código en un determinado lenguaje de programación. Por lo tanto, un Bot es un programa informático que trabaja de manera automática y autónoma que no depende de la supervisión o participación humana para llevar a cabo sus funciones para el que fue programado.

Las ventajas de usar Bots son las siguientes:

- Más rápido que los humanos en tareas repetitivas.
- Eliminación de errores humanos.
- Reducen los costos laborales para las empresas.
- Están disponibles 24 horas al día, 7 días a la semana.
- Son personalizables.
- Aumenta la satisfacción en clientes y empleados.

Las desventajas son las siguientes:

- Necesitan mantenimiento y debido a esto puede ocasionar retrasos.

Aunque un Bot puede desempeñar funciones positivas también existen aquellos que son perjudiciales debido a que tienen fines ilegales y dañinos, como por ejemplo la gran cantidad de

spam que llegan a un correo, hasta Bots que roban información a través de enlaces falsos. La variedad de Bots se utilizan en diferentes áreas, como servicio al cliente, negocios, funcionalidad de búsqueda, entretenimiento y muchos más.

Ejemplos de procesos que puede realizar un Bot:

- Automatización de tareas en un entorno web para descargar archivos, extraer información, realizar registros.
- Realizar reportes en base al procesamiento de datos que se encuentran en alguna fuente como Excel, CSV, SQL u otro y enviarlos por correo.
- Chatbots en una página web, en una app, en las respuestas de Emails, en las conversaciones por Mensaje de texto al celular, Facebook Messenger, WhatsApp u otra herramienta de comunicación y mensajería.

3.2.2. Factura Electrónica

Según SUNAT (2017) menciona que "la factura electrónica es la factura regulada por el Reglamento de Comprobantes de pago (RS 007-99/SUNAT) soportada en un formato digital que cumple con las especificaciones reguladas en la R.S.097-2012/SUNAT, R.S.XXX-2017/SUNAT y modificatorias, que se encuentra firmada digitalmente."

Esta factura electrónica se encontrará en un formato XML que contendrá los datos tributarios referentes a la factura.

A continuación, se detallará en las siguientes tablas, los campos principales que se encuentran en una Factura electrónica, previo mostraremos la nomenclatura de representación del valor de cada dato en la Tabla 1.

Tabla

Nomenclatura de datos de una Factura Electrónica.

a	Carácter alfabético
n	Carácter numérico
an	. Carácter alfanumérico
a3	3 caracteres alfabéticos de longitud fija
n3	3 caracteres numéricos de longitud fija
an3	3 caracteres alfa-numéricos de longitud fija
a…3	Hasta 3 caracteres alfabéticos
n…3	Hasta 3 caracteres numéricos
an…3	Hasta 3 caracteres alfa-numéricos

Nota. Adaptado de *Guía de Elaboración de Documentos XML Factura Electrónica,* por SUNAT, 2017.

Tablas 1

Formato de los elementos de los datos de una Factura electrónica

n(12, 2)	elemento numérico hasta 12 enteros + punto decimal + dos decimales
n(2,2)	elemento numérico hasta 2 enteros + punto decimal + hasta dos decimales
F#####	elemento inicia con la letra F seguida de cinco dígitos
YYYY-MM-DD	formato fecha yyyy=año, mm=mes, dd=día

Nota. Adaptado de *Guía de Elaboración de Documentos XML Factura Electrónica,* por SUNAT, 2017.

Para un determinado campo se considerará la nomenclatura M si es Mandatorio u obligatorio o C si es Condicional u opcional.

Tabla 2

Datos generales de una Factura Electrónica

Datos de la Factura electrónica			
Dato	Condición	Tipo y Longitud	Formato
Numeración, conformada por serie y número correlativo	M	an...13	F###-NNNNNNNN
Fecha de emisión	M	an...10	YYYY-MM-DD
Hora de Emisión	M	an...11	hh:mm:ss.0z
Tipo de documento (Factura)	M	an2	
Tipo de moneda en la cual se emite la factura electrónica	M	an3	
Fecha de Vencimiento	C	an10	YYYY-MM-DD
Firma Digital	M	an...3000	

Nota. Adaptado de *Guía de Elaboración de Documentos XML Factura Electrónica,* por SUNAT, 2017.

Tabla 3

Datos del emisor de una Factura Electrónica

Datos del emisor			
Dato	Condición	Tipo y Longitud	Formato
Número de RUC	M	n11	
Nombre Comercial	C	an…100	

Nota. Adaptado de *Guía de Elaboración de Documentos XML Factura Electrónica,* por SUNAT, 2017.

Tabla 4

Datos del receptor de una Factura Electrónica

Datos del cliente o receptor			
Dato	Condición	Tipo y Longitud	Formato
Tipo y número de documento de identidad del adquirente o usuario	M	an…15	
Apellidos y nombres, denominación o razón social del adquirente o usuario	C	an…100	

Nota. Adaptado de *Guía de Elaboración de Documentos XML Factura Electrónica,* por SUNAT, 2017.

Tabla 5

Datos de ítem de una Factura Electrónica

Datos del detalle o Ítem			
Dato	Condición	Tipo y Longitud	Formato
Unidad de medida por ítem	M	an…3	
Cantidad de unidades por ítem	M	an…23	n(12,10)
Código de producto	C	an…30	
Código producto de SUNAT	C	an…20	

Nota. Adaptado de *Guía de Elaboración de Documentos XML Factura Electrónica,* por SUNAT, 2017.

Tabla 6

Datos con respecto a los importes totales de una Factura Electrónica

Totales			
Dato	Condición	Tipo y Longitud	Formato
Total valor de venta	M	n(12,2)	
Sumatoria IGV	C	n(12,2)	

Nota. Adaptado de *Guía de Elaboración de Documentos XML Factura Electrónica,* por SUNAT, 2017.

3.2.3. Lenguaje de Programación: Python

Según *Python 3.9.6 documentation* (2021) menciona que:

> Python es un lenguaje de programación interpretado, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipificación dinámica, tipos de datos dinámicos de muy alto nivel y clases. Admite múltiples paradigmas de programación más allá de la programación orientada a objetos, como la programación funcional y de procedimientos. Python combina una potencia notable con una sintaxis muy clara. Tiene interfaces para muchas llamadas del sistema y bibliotecas, así como para varios sistemas de ventanas, y es extensible en C o C ++. También se puede utilizar como lenguaje de extensión para aplicaciones que necesitan una interfaz programable. Finalmente, Python es portátil: se ejecuta en muchas variantes de Unix, incluidos Linux y macOS, y en Windows.

Figura 1

Logo oficial de Python

Nota. Tomado de *Python Software Foundation,* 2021. https://www.python.org

Las principales características de Python son las siguientes:

- ***Propósito general***

  Se puede realizar programas de todo tipo para diferentes propósitos como Desarrollo Web, Big Data, Inteligencia Artificial, Web Scraping, Programas de escritorio, entre otros.

- ***Multiplataforma***

Existe diferentes versiones disponibles y compatibles con los sistemas operativos como Windows, Linux, Unix, Mac OS.

- ***Interpretado***

  Significa que Python interpreta el código que ha realizado el programador es decir que lo traduce y lo ejecuta a la vez.

- ***Multiparadigma***

  Python admite varios tipos de paradigmas estos son modelos de desarrollo al realizar un software, entre estos tipos tenemos: Programación orientada a objetos, Programación imperativa y Programación funcional.

- ***Sintaxis clara***

  Python cuenta con una sintaxis muy clara y sencilla debido a su indentación que se tiene que hacer de manera obligatoria.

- ***De tipado dinámico***

  Al declarar una variable no es necesario decirle que tipo de dato es (si es entero, decimal, cadena u otro), Python hace que la variable se adapte a nuestro programa.

Las principales librerías a usar son las siguientes:

- ***xmltodict***

  Librería que permite trabajar con archivos en formato XML al convertir el contenido de esta en una estructura de datos tipo diccionario.

- ***smtplib***

  Librería que permite el envío de mensajes de correo electrónicos a través del SMTP (Protocolo de transferencia de correo simple).

- ***imaplib***

  Librería que permite acceder a los mensajes almacenados en un determinado correo electrónico a través del IMAP (Protocolo de acceso a mensajes de Internet).

- ***email***

  Librería que permite dar formato a texto de mensaje o adjuntar imágenes, archivos, hipervínculos u otro archivo receptivo enviado a través de correo electrónico.

- ***sqlite3***

  Librería que permite conectarse a una base de datos.

3.2.4. Base de datos: SQLite

Según Antonio, G. (2011). menciona que "SQLite es un motor de base de datos construido sobre el lenguaje de programación C, de código abierto que se caracteriza por el almacenamiento local de datos para aplicaciones como: Firefox, Chrome, Opera, Android, iOS, entre otros, y dispositivos individuales".

Figura 2

Logo oficial de SQLite

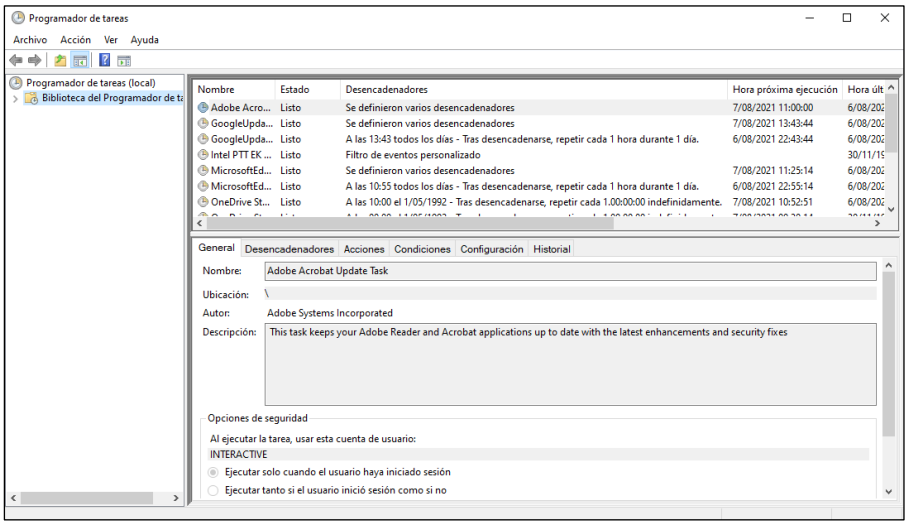
Nota. Tomado de *About SQLite*, 2021. https://www.sqlite.org/about.html

3.2.5. Programador de Tareas

Es una herramienta de Microsoft que permite ejecutar ciertas aplicaciones en un día y/u hora determinada, de manera periódica o cuando se produce algún evento como por ejemplo Crear una Tarea que se encargue de abrir el navegador Chrome luego de encender la computadora.

Figura 3

Programador de Tareas



3.2.6. Archivos XML (Extensible Markup Language)

Según Indeed Editorial Team (2020), indica que "un archivo XML es un archivo de lenguaje de marcado extensible y se utiliza para estructurar datos para su almacenamiento y transporte. En un archivo XML, hay etiquetas y texto. Las etiquetas proporcionan la estructura a los datos. El texto del archivo que desea almacenar está rodeado por estas etiquetas, que se adhieren a pautas de sintaxis específicas. En esencia, un archivo XML es un archivo de texto estándar que utiliza etiquetas personalizadas para describir la estructura del documento y cómo debe almacenarse y transportarse".

A continuación, mostramos un ejemplo simple:

Figura 4

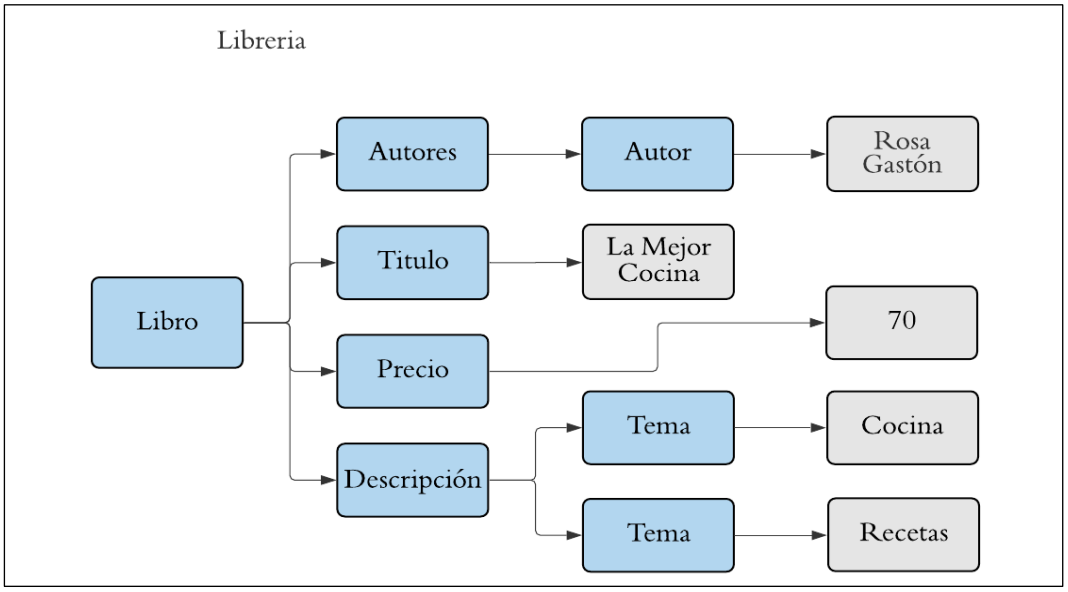
Ejemplo de la estructura de un archivo XML

```
<?xml version="1.0">
<librería>
  <libro>
    <autores>
        <autor>Rosa Gastón</autor>
    </autores>
    <titulo>La mejor Cocina</titulo>
    <precio moneda="PEN">70</precio>
    <descripcion>
        <tema>Cocina</tema>
        <tema>Recetas</tema>
    </descripcion>
   </libro>
</libreria>
```

En forma gráfica de la estructura de la Figura 3, podríamos representarlo de la siguiente manera como en la Figura 4:

Figura 5

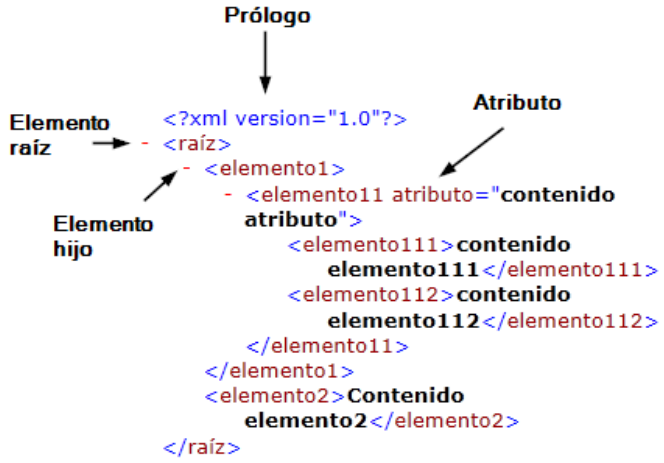
Representación gráfica equivalente de la Figura 3



La estructura de un archivo XML es la siguiente:

Figura 6

Componentes de un archivo XML



Nota. Tomado de *XML práctico - Bases esenciales, conceptos y casos prácticos, por* Thierry Boulanger*, 2015.*

- **Prólogo**.

  Contiene la versión del XML, el tipo de codificación, si es un documento autónomo y el tipo de documento que es.

  Ejemplo:

  <?xml versión= “1.0” encoding= “ISO-8859-1” standalone= “yes”?>

- **Cuerpo**.

  Es el contenido de información del archivo, organizado como la estructura de un árbol en la que hay un elemento principal o raíz en las cuales dentro de este elemento se encuentra el resto de los elementos.

  Cualquier elemento puede contener otro sub-elemento o elemento(s) hijo(s).

- **Elemento.**

  Se encuentra delimitado por una etiqueta de apertura (<tag>) y otra etiqueta de fin (</tag>) y todo lo que se encuentra dentro de estas etiquetas se denomina “contenido de etiqueta”.

  Ejemplo: <titulo>XML Guía de Aprendizaje</titulo>

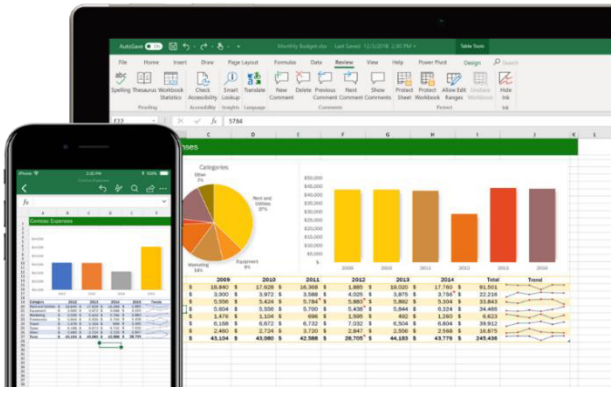
3.2.7. Microsoft Office Excel

Según Lambert, J. y Frye, C. (2015) explicaron que:

Microsoft Office Excel proporciona herramientas poderosas que permiten a los usuarios organizar, analizar, administrar y compartir información fácilmente. La base de Excel y las ubicaciones donde realiza su trabajo son filas, columnas y celdas dentro de una hoja de trabajo, y las hojas de trabajo como parte de un libro de trabajo. Muchas de las herramientas que usa mientras trabaja en Excel se encuentran en la cinta que se indica en el sector sobresaliente de la ventanilla. La cinta está organizada en pestañas de comando orientadas a tareas. Cada pestaña se divide en grupos de comandos específicos de tareas con comandos y opciones que se relacionan con el nombre del grupo. (p. 197)

Figura 7

Hoja de Cálculo Excel en diferentes dispositivos.



Nota. Tomado de *Microsoft Excel 2010*, 2021. https://www.microsoft.com/es-es/microsoft-365/previous-versions/microsoft-excel-2010

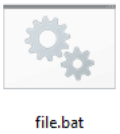
3.2.8. Archivos Batch o Bat

Según Norfi Carrodeguas. (2019) afirma que:

Los archivos Batch son aplicaciones para Windows, que pueden ser creadas por los propios usuarios, para infinidad de tareas. Son simples archivos de texto con la extensión de archivo .CMD o .BAT, que al ejecutarlas (dar dos clics en ellas), sus instrucciones son ejecutadas en la Consola de CMD o Símbolo del sistema.

Figura 8

Ejemplo de un archivo Bat



Nota. Tomado de Remontka.pro, 2021. https://remontka.pro/images/make-bat-file-windows.png

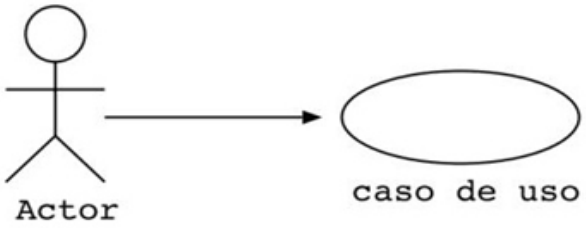
3.2.9. Diagramas de Casos de Uso

Según *IBM Docs*. (2021) mencionan que:

Un *caso de uso* es un artefacto que define una secuencia de acciones que da lugar a un resultado de valor observable. Los casos de uso proporcionan una estructura para expresar requisitos funcionales en el contexto de procesos empresariales y de sistema. Los casos de uso pueden representarse como un elemento gráfico en un diagrama y como una especificación de caso de uso en un documento textual. Un caso de uso empresarial define una secuencia de acciones que una empresa lleva a cabo y que da lugar a un resultado de valor observable (una salida de trabajo) para un actor empresarial particular o que muestra el modo en que la empresa responde a un evento empresarial.

Figura 9

Ejemplo de un Diagrama de Caso de Uso



Nota. Observamos al actor y a una elipse que representa a un Caso de Uso. Tomado de *UML: Casos de Uso*. INGENIERÍA DEL SOFTWARE, 2015. https://ingsotfwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/

3.3. Propuesta, Análisis y Definición del Proceso

3.3.1. Propuesta

En el presente trabajo, se propone el desarrollo de un Bot utilizando el lenguaje de programación Python y librerías específicas de esta, para el Registro de Facturas en una Base de Datos.

El Bot propuesto requiere de las siguientes especificaciones técnicas:

- Sistema Operativo (SO) Windows 7, o superior.
- Licencia de Microsoft Office.
- Acceso a memoria del sistema y redes.
- Conexión a internet estable.
- Acceso con usuario y contraseña para el correo electrónico para el Bot.

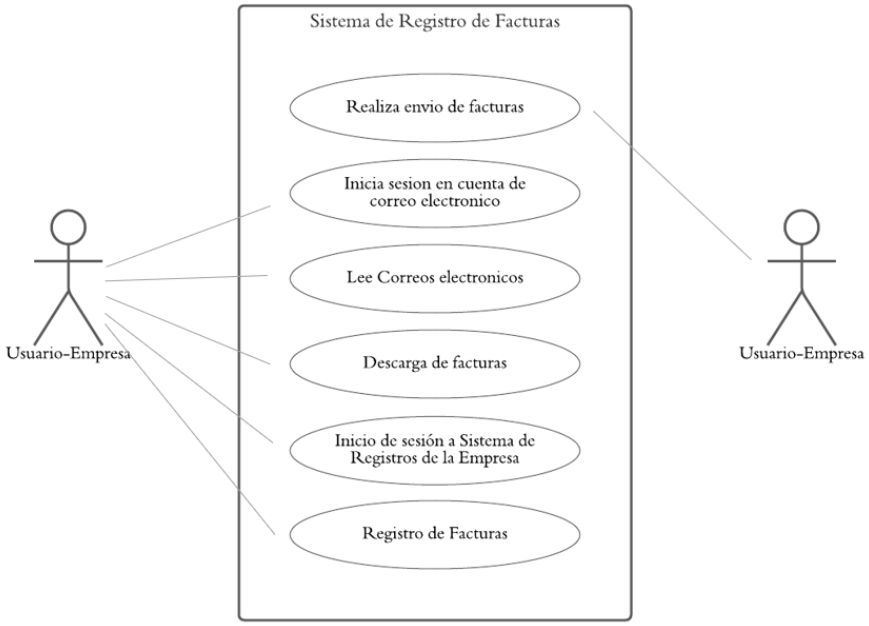
3.3.2. Análisis y Definición de Proceso

Flujo del Proceso Actual.

La Figura 8 muestra el diseño del diagrama de *Casos de Uso* para el usuario-empresa, que envía las facturas electrónicas mediante correo a otro usuario-empresa y este tiene que realizar la descarga de facturas y registrarlas a su sistema de manera manual, sin implementar algún sistema de automatización.

Figura 10

Diagrama de Caso de Uso para el Registro de Facturas en su sistema.



Flujo del Proceso para el Desarrollo del Bot

En la Figura 9, observamos el diagrama de Flujo del proceso que nos va a permitir ver la secuencia de pasos de manera detallada que realizará el Bot.

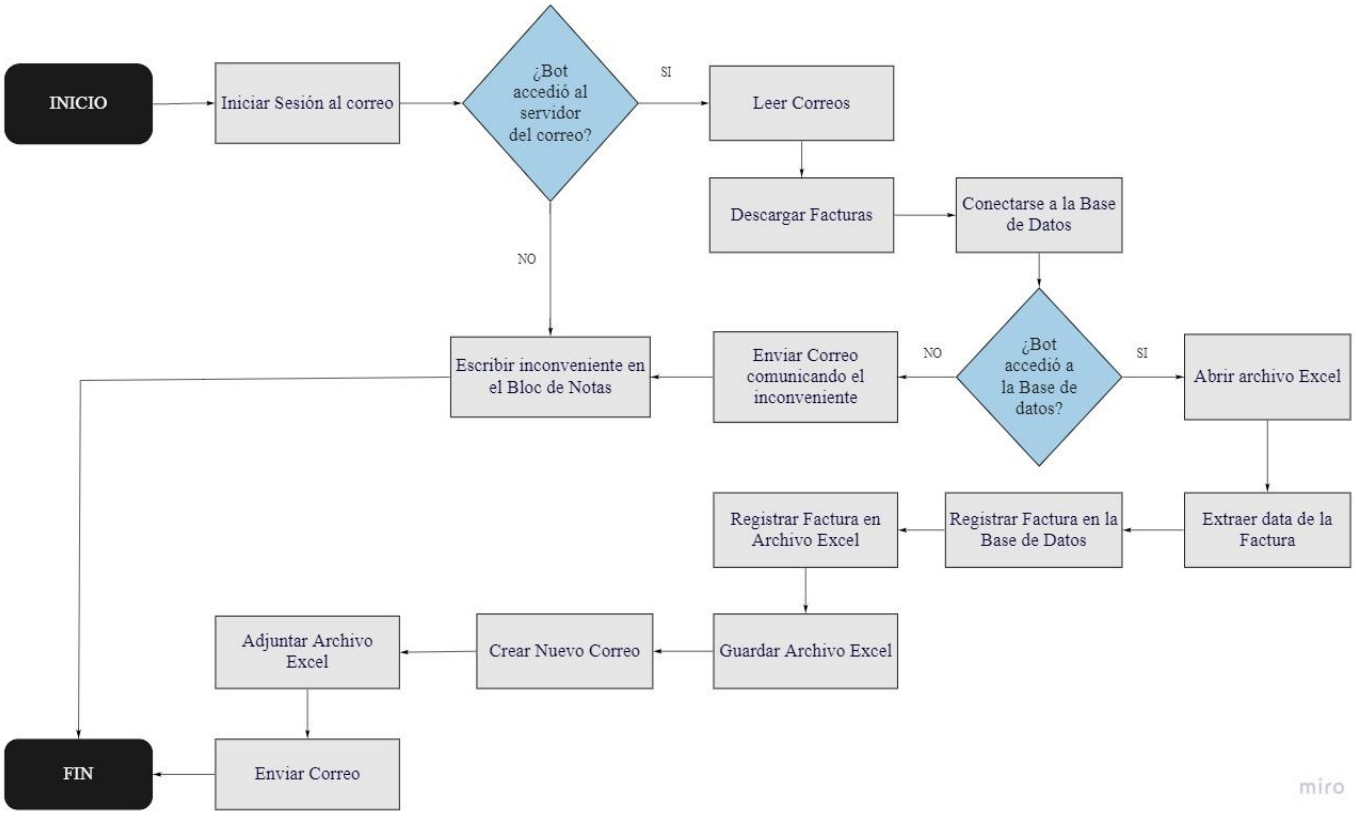
El proceso empieza iniciando sesión en el correo electrónico donde llegan las facturas, en el caso de que se conectó al servidor del correo, se descargará todas las facturas en una carpeta. Caso contrario de no haberse conectado con el servidor, este inconveniente se detallará en un Bloc de Notas donde se registra todos los posibles errores que pueda presentar el Bot.

Una vez descargadas las facturas en una carpeta, el Bot se conectará a la Base de datos, en el caso de no tener inconvenientes con la conexión, este procederá a extraer toda la información de la factura en XML y esta será validada de acuerdo a la *Guía de Elaboración de Documentos XML* elaborada por la *SUNAT*, el cual se registrará tanto en la Base de Datos y en un archivo Excel. Si hubo inconvenientes al extraer la información, solo se colocará el nombre del archivo en el archivo Excel para una posterior búsqueda de forma manual por el usuario en la bandeja del correo. En el caso de haber inconvenientes para conectarse a la Base de Datos, este se colocará en el Bloc de Notas mencionado dichos inconvenientes.

Por último, este archivo Excel donde se encuentra la información de las facturas, será adjuntado en un correo que se enviará al personal encargado del Bot.

Figura 11

Diagrama de flujo para el registro de facturas electrónicas en una base de datos



Aplicaciones utilizadas en el proceso de Automatización.

La tabla 8 muestra una lista de Aplicaciones que interactúan en el proceso de Automatización.

Tabla 7

Aplicaciones con la que interactuará el Bot

N°	Aplicación
1	Gmail
2	Excel
3	SQLite
4	Bloc de Notas

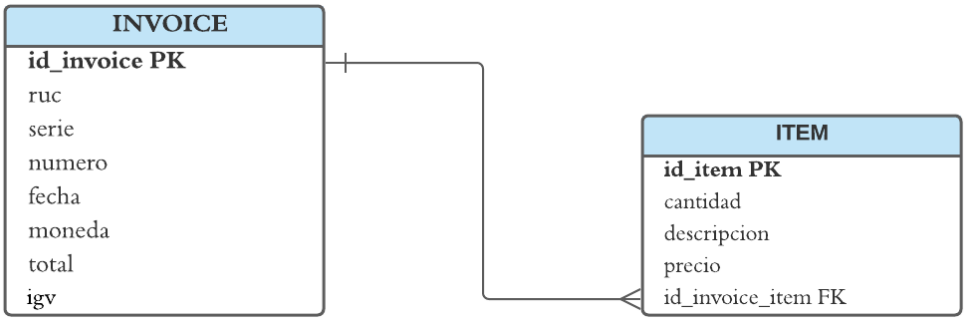
Implementación de la Base de datos.

Por último, se crea una base de datos llamada *Master*, con una tabla llamada *invoice* donde se almacenará la información general de la factura y la tabla *ítem* donde se almacenará los detalles de cada producto de las facturas.

En la Figura 10 se observa un diagrama que contiene dos clases, una clase hace referencia a una tabla que se encuentra en una Base de datos:

Figura 12

Diagrama de Clases de la Base de Datos "Master"



Nota. La primera clase *invoice* que tiene los siguientes atributos: *id_invoice, ruc, serie, numero, fecha, moneda, total, igv,* y la segunda clase *item* tiene los siguientes atributos: *id_item, cantidad, descripcion, precio, id_invoice_item*, ambas clases se encuentran ligadas a través de la clave foránea *id_invoice_item* que va a permitir relacionar una factura con sus productos.

3.4. Desarrollo del Bot

En base al análisis del proceso, se concluyó que es una secuencia repetitiva y específica, el cual demuestra ser un gran candidato para su automatización.

3.4.1. Casos de Uso

Seguidamente, se muestran los *Casos de Uso* utilizados para el desarrollo del Bot dentro del flujo propuesto para automatizar.

- CU001 – Iniciar Sesión en la cuenta de correo y descargar las facturas en formato XML.
- CU002 – Extraer información de las facturas en formato XML.
- CU003 – Realizar validaciones de la información extraída de las facturas en formato XML.
- CU004 – Almacenar los datos de la información extraída de las facturas en la Base de Datos.
- CU005 – Registrar en Excel los datos de las facturas.
- CU006 – Enviar correo a usuario con informe y archivo Excel adjunto.

3.4.2. Actividades de Casos de Uso

A continuación, se detalla las actividades que se presentan en los *Casos de Uso* mencionados en el punto anterior.

CU001 – Iniciar Sesión en la cuenta de correo y descargar las facturas en formato XML.

Tabla 8

Caso de Uso CU001

CU001	**Iniciar Sesión en la cuenta de correo y descargar las facturas en formato XML.**
Descripción	▪ Iniciar Sesión a la cuenta de correo electrónico asignado al Bot en donde llegan las Facturas en formato XML. ▪ Eliminar Carpeta donde se almacenan las Facturas en formato XML. ▪ Crear Carpeta donde se almacenan las Facturas en formato XML. ▪ Ingresar a la Bandeja de Entrada, leer los correos objetivos y descargar las Facturas. ▪ Registrar en un Bloc de notas en caso de que no se haya podido conectar al servidor de Correo,
Objetivo	Obtener todas las Facturas en formato XML.
Para quien	Insumo de entrada para el Bot y para el CU002
Participantes	Bot

CU002 – Extraer información de las facturas en formato XML.

Tabla 9

Caso de Uso CU002

CU002	**Extracción de datos de la Factura en formato XML.**
Descripción	▪ Ingresar a la carpeta donde se descargó todas las Facturas en formato XML. ▪ Tomar cada archivo XML e identificar los campos necesarios a extraer como la, serie, número, proveedor, fecha emisión, moneda, total, IGV, productos (cantidad, nombre, precio unitario por cada uno).
Objetivo	Obtener datos que pertenezcan a la factura.
Para quien	Insumo de entrada para el CU003
Participantes	Bot

CU003 – Realizar validaciones de la información extraída de las facturas en formato XML.

Tabla 10

Caso de Uso CU003

CU003	**Realizar validaciones de la información extraída de las facturas en formato XML**
Descripción	▪ Luego de extraer los datos de cada Factura estos serán validados basándose en los formatos que tienen que cumplir los campos de una factura electrónica según la ***Guía de Elaboración de Documentos Electrónicos*** de SUNAT.
Objetivo	Validar datos que fueron extraídos de la factura y que estos cumplan el formato dado por la ***Guía de Elaboración de Documentos Electrónicos XML.***
Para quien	Insumo de entrada para el CU004
Participantes	Bot

CU004 – Almacenar los datos de la información extraída de las facturas en la Base de Datos.

Tabla 11

Caso de Uso CU004

CU004	**Almacenar los datos de la información extraída de las facturas en la Base de Datos**
Descripción	▪ Después de validar la información de una Factura, estos se almacenarán en la Base de Datos ***Master***. ▪ La serie, número, proveedor, fecha emisión, moneda, total, igv, serán registradas en la tabla ***invoice***. ▪ Los productos (cantidad, nombre, precio unitario por cada uno) de cada factura serán almacenados en la tabla ***item.*** ▪ Registrar en un Bloc de notas en caso de que no se haya podido conectar al servidor de Base de Datos
Objetivo	Almacenar facturas con datos correctos en la Base de Datos ***Master***.
Para quien	Insumo de entrada para el CU005
Participantes	Bot

CU005 – Registrar en Excel los datos de las facturas.

Tabla 12

Caso de Uso CU005

CU005	***Registrar en Excel la data de las Facturas***
Descripción	▪ Crear un archivo Excel. ▪ Después de que los datos de una factura se almacenan en la Base de Datos, estos también se registran en un archivo Excel. ▪ Se registran en el archivo Excel según la cantidad de productos tenga una factura (Por ejemplo, si hay una factura con tres productos, estos se registraran en 3 filas, donde en cada fila se especificará cada producto, compartiendo la información general de la Factura). ▪ En el caso de que no se pueda extraer la información de la Factura debido a un error de estructura del archivo XML solo será registrado en el Excel el nombre del archivo. ▪ Guardar archivo Excel.
Objetivo	Registrar las facturas en el archivo Excel e indicar si fueron registradas en la Base de Datos o tuvieron algún error.
¿Para quién?	Insumo de entrada para el CU006
Participantes	Bot

CU006 – Enviar correo a usuario con reporte y archivo Excel adjunto.

Tabla 13

Caso de Uso CU006

CU006	**Enviar correo a usuario con reporte y archivo Excel adjunto.**
Descripción	▪ Se le notificará al personal encargado del Bot a través de un correo electrónico informando que el Bot ya hizo el registro de facturas adjuntando el reporte en Excel o si fuera el caso de que el Bot ha tenido problemas para conectarse con la Base de Datos.
Objetivo	Informar a personal a través de un correo adjuntando el reporte en Excel o si hubo problemas en la conexión con la Base de Datos.
¿Para quién?	Para el personal asignado encargado del Bot
Participantes	Bot

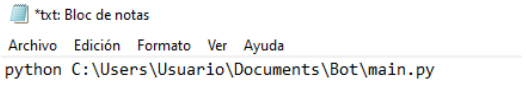
3.5. Implementación del Bot

3.5.1. Crear archivo Bat

Para crear el archivo Bat (ejecutable del Bot), abriremos un *Bloc de Notas* y colocaremos la ruta donde se encuentra nuestro programa principal, anteponiendo la palabra *python*.

Figura 13

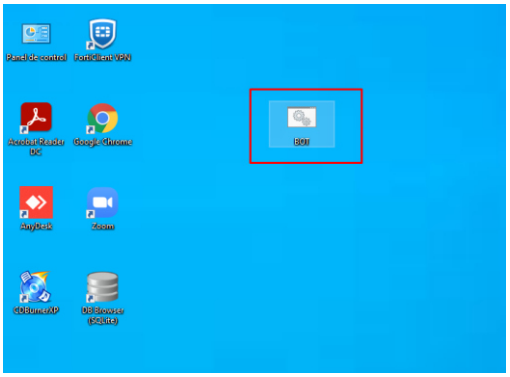
Creación de Archivo Bat



El archivo es guardado en el escritorio para un fácil acceso, colocando el nombre "BOT" seguido de la extensión ".bat", a todo esto, colocarlo entre comillas y darle clic en *Guardar*.

Figura 14

Ejecutable del Bot ubicado en el escritorio



3.5.2. Configuración del Programador de Tareas

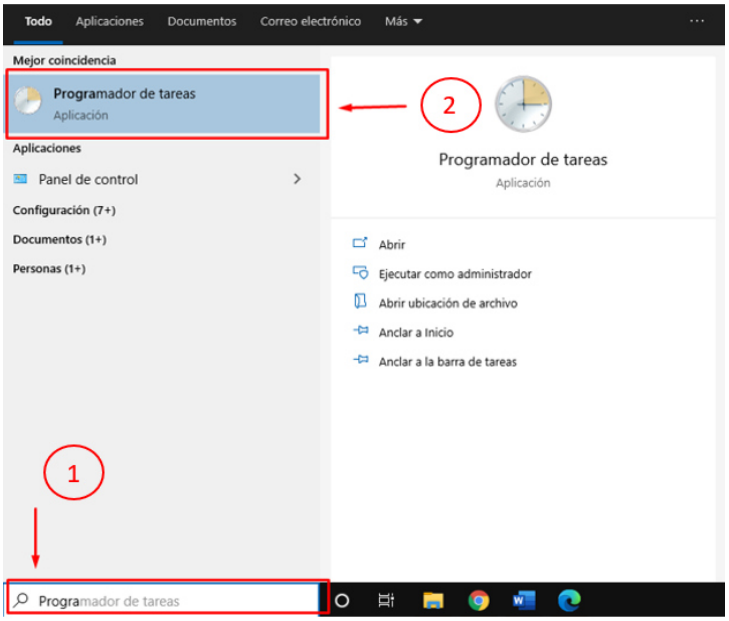
Para la ejecución del Bot de manera diaria o recurrente, este será configurado a través del Programador de Tareas todos los días a las 6:00 pm (o en la hora que se requiera).

Se muestra a continuación las configuraciones realizadas.

a. En la Barra de Tareas de la computadora, donde será ejecutado el Bot, colocamos *Programador de Tareas* para su búsqueda y damos doble clic sobre la aplicación.

Figura 15

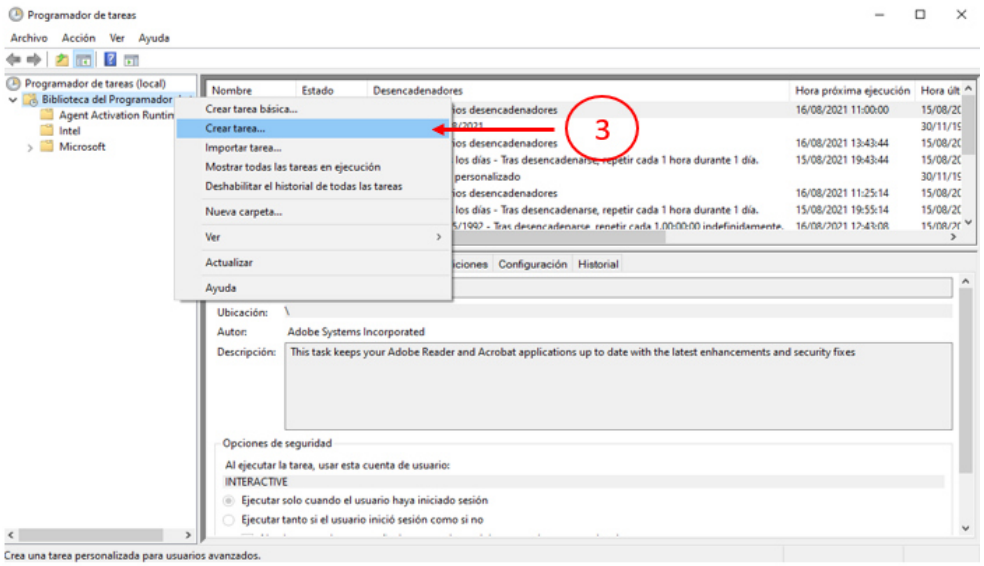
Búsqueda de Herramienta Programador de Tareas de Windows



b. A continuación, se mostrará la herramienta como en la Figura 20, hacer clic derecho a *Biblioteca del Programador de Tareas,* de tal manera que se desplegará una lista de opciones en donde seleccionaremos *"Crear Tarea"*.

Figura 16

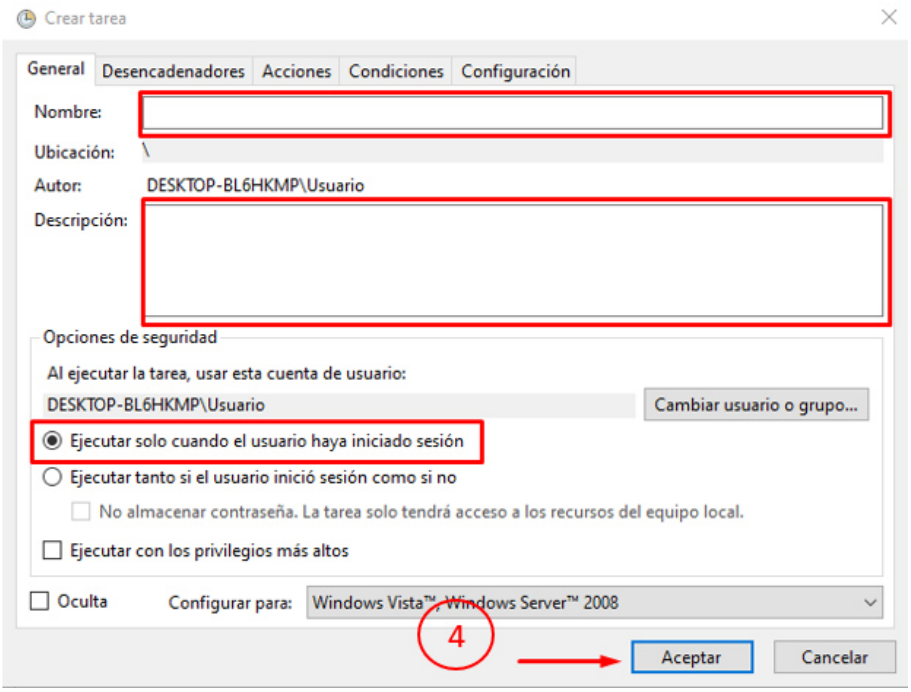
Entorno del Trabajo de Programador de Tareas



c. Se mostrará la siguiente ventana, en la pestaña "*General*", se colocará el nombre de la tarea a ejecutarse, la descripción (opcional), elegir la opción "*Ejecutar solo cuando el usuario haya iniciado sesión"* y darle clic en Aceptar.

Figura 17

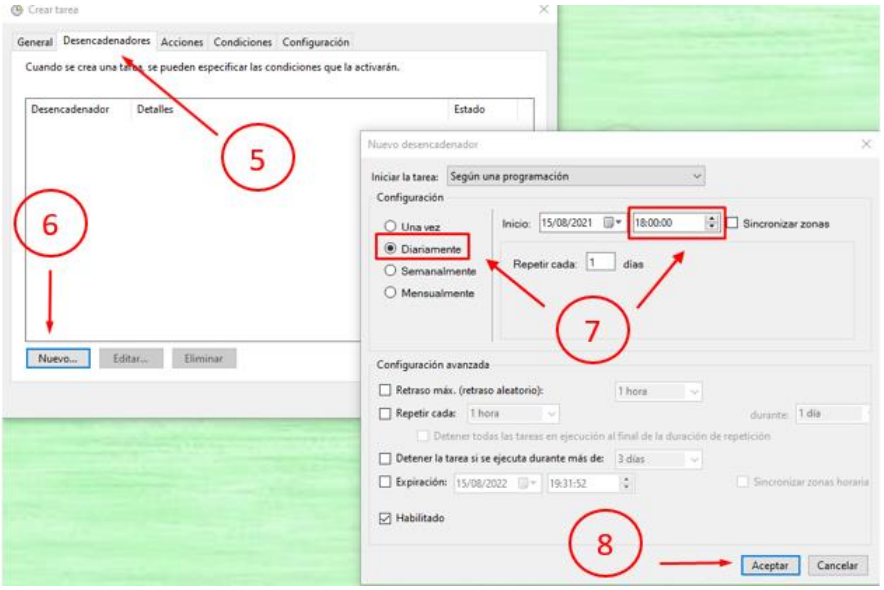
Creación de una tarea, aspecto General



d. Luego nos ubicamos en la pestaña "*Desencadenadores*" y le damos clic en *Nuevo, en donde* se abrirá una segunda ventana (Ver Figura 22), en este caso como ya definimos un horario en la que se ejecutará el Bot, seleccionamos la opción *Diariamente* y establecemos la hora (por ejemplo, a las 6:00 pm) y le damos clic en *Aceptar*.

Figura 18

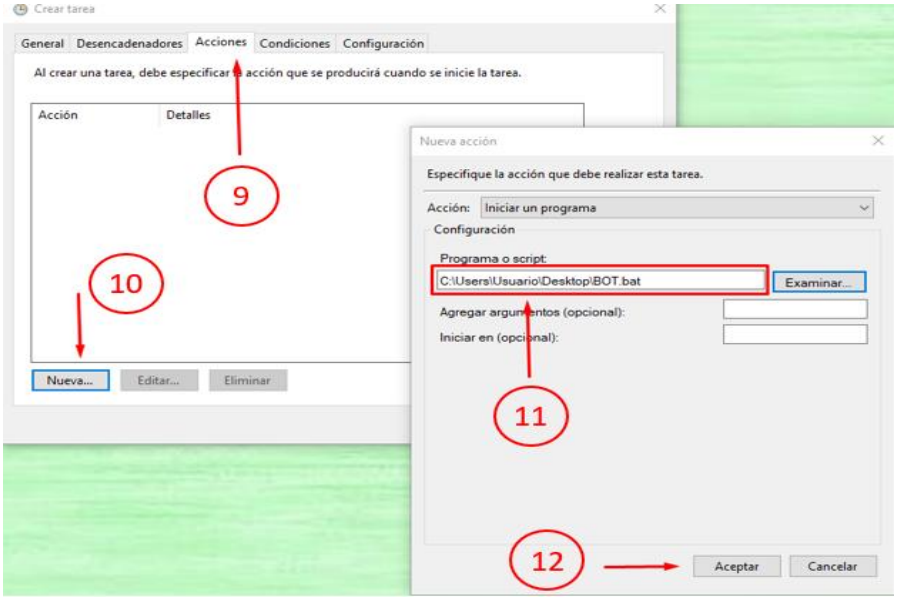
Creación de desencadenador



e. Luego nos ubicamos en la pestaña "*Acciones*" y damos clic en *Nuevo,* se abrirá la segunda ventana de la Figura 23, en este caso colocamos la ruta donde se encuentra el ejecutable del Bot y le damos clic en *Aceptar.*

Figura 19

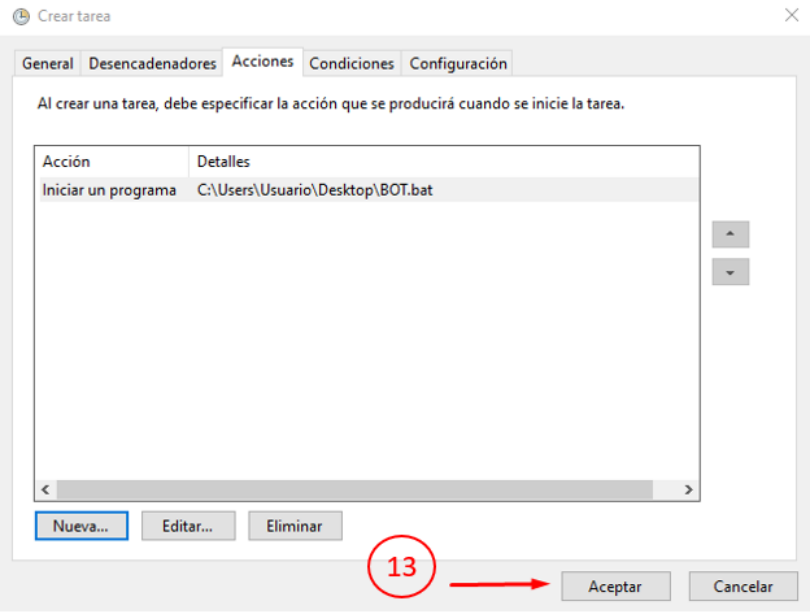
Creación de Acción



f. Por último, dar clic en Aceptar y ya estará configurado el Programador de Tareas para la ejecución de manera diaria del Bot a las 6:00 pm.

Figura 20

Paso final para Crear Tarea



Capítulo 4

CONCLUSIONES

En base a la realización de este trabajo se puede mencionar las siguientes conclusiones:

- La descripción oportuna y registro de los diferentes inconvenientes que podría tener el Bot pueden ser revisadas por el personal encargado del Bot, a través del archivo de texto Bloc de Notas específico para estos casos. Solucionando así los inconvenientes y garantizando la continuidad del proceso.
- Se cumplió el objetivo general mostrado inicialmente en el trabajo al desarrollar un Bot que registre de manera satisfactoria el proceso de Registro de Facturas en una Base de Datos.
- Reducción significativa del tiempo al emplear este Bot ya que puede registrar 50 facturas en formato XML descargadas del correo en 15 segundos aproximadamente, tomando en cuenta que un empleado puede tomarse un tiempo aproximado de 3 minutos en registrar una factura.
- Debido a la reducción de tiempo de registro de las facturas electrónicas mediante el Bot, esto supone también un ahorro significativo en costos.
- Se puede desarrollar un Bot orientando a las necesidades de cada empresa. Es decir, se puede automatizar el registro detallado de los productos de una factura a la medida de los sistemas que usa la empresa.
- La implementación de Bots en las empresas constituye una oportunidad de innovación, presupuesto, tiempo y calidad.

Capítulo 5

RECOMENDACIONES

A continuación, se muestra las recomendaciones con respecto al Bot, así como también para algunos posibles trabajos futuros para la mejora del tema:

- A diferencia de un software especializado que requiere de un hardware de instalación, los Bots pueden trabajar en un entorno virtual, el cual disminuye los gastos operativos.
- Se recomienda realizar el mantenimiento y/o actualización a los Bots cada cierto tiempo para realizar las mejoras y/o prever algún posible inconveniente.
- Mediante el uso adecuado de etiquetas en el archivo XML, los Bots también pueden registrar la información específica de: las Boletas de venta electrónica, Nota de crédito electrónica, Nota de débito electrónica entre otros.
- Se recomienda adaptar e implementar el Bot para que pueda trabajar con diferentes Sistemas de Gestor de Bases de Datos y diferentes servidores de Correo, según la empresa lo requiera.
- Como continuidad de estudio de este trabajo se recomienda, poder realizar un aplicativo de tal forma que sea de fácil uso e implementación en una empresa que permita la configuración del uso de la Base de Datos y Correo que usa la empresa para evitar la codificación.
- Se recomienda realizar un manual de usuario para el personal encargado del Bot y un manual técnico para el mantenimiento del Bot.

Capítulo 6

BIBLIOGRAFÍA

Deloitte (2017). *Automatización Robótica de Procesos (RPA).* https://www2.deloitte.com/content/dam/Deloitte/mx/Documents/strategy/Automatizacion_Rob%C3%B3tica_Procesos.pdf

IONOS (2020). *Bot: ¿Qué es? tipos y funciones*. https://www.ionos.es/digitalguide/online-marketing/vender-en-internet/que-es-un-bot/

SUNAT (2017). *Guía de Elaboración de Documentos XML Factura Electrónica.*

Python (2021). *Python 3.9.6 documentation.* https://docs.python.org/3/

Python (2021). *smtplib — SMTP protocol client*. https://docs.python.org/3/library/smtplib.html

Python (2021). *imaplib — IMAP4 protocol client*. https://docs.python.org/3/library/imaplib.html

Python (2021). *email.mime: Creating email and MIME objects from scratch* https://docs.python.org/3/library/email.mime.html

Python (2021). *email — An email and MIME handling package* https://docs.python.org/3/library/email.html#:~:text=The%20email%20package%20is%20a,such%20as%20smtplib%20and%20nntplib%20.

Python (2021) *sqlite3 — DB-API 2.0 interfaz para bases de datos SQLite¶* https://docs.python.org/es/3.10/library/sqlite3.html

SQLite (2021) *Features Of SQLite.* https://www.sqlite.org/features.html

Gerardo Antonio Cabero, Daniel Maldonad (s.f). *Sqlite: Rápido, ágil, liviano y robusto.* http://sqlite-latino.blogspot.com/

Andrés Visus (2020). *¿Para qué sirve Python? Razones para utilizar este lenguaje de programación.* https://www.esic.edu/rethink/tecnologia/para-que-sirve-python

Indeed Editorial Team (2020). *What is an XML file and How Do I Open One?* Indeed Career Guide; Indeed. https://www.indeed.com/career-advice/career-development/xml-file

World Wide Web Consortium (2015). *XML ESSENTIALS.* https://www.w3.org/standards/xml/core

Jesús Vegas (2001). *Una Introducción a XML.* https://www.infor.uva.es/~jvegas/cursos/web/xml/ixml/ixml.html#caracteristicas

Thierry Boulanger (2015). *XML práctico Bases esenciales, conceptos y casos prácticos* (2ª edición)

Microsoft (2021). *Programador de tareas para desarrolladores.* https://docs.microsoft.com/es-es/windows/win32/taskschd/task-scheduler-start-page

Excel.X (2020). *What is Excel?* https://excelx.com/what-is-excel/?fbclid=IwAR3hjRVq1HbMB1eFmV1NeIyLURsjbwrtvEkmhAbYILHiruUKiNSz3Uu_GHo

Norfi Carrodeguas. (2019). *Qué son los archivos BATCH o BAT, usos prácticos y como crearlos en Windows*. https://norfipc.com/comandos/que-son-archivos-batch-bat-usos-practicos-como-crearlos.php

IBM Docs (2021). *Diagramas de Casos de Uso.* https://www.ibm.com/docs/es/elm/6.0.3?topic=requirements-defining-use-cases

Capítulo 7

ANEXOS

7.1. Código fuente

C1. Script *conexión_email.py,* que contiene las funciones de conexión al servidor de correo, descarga de facturas y envío de correo adjuntando reporte en Excel.

```
def read_email_from_gmail():
      if os.path.isdir(d['path_invoices']):
            shutil.rmtree(d['path_invoices'])
            os.mkdir(d['path_invoices'])
      else:
            os.mkdir(d['path_invoices'])
      try:
            mail = imaplib.IMAP4_SSL('imap.gmail.com')
            mail.login(gmail_user,gmail_pass)
            mail.select()
            data = mail.search(None, 'ALL')
            if data[0] == 'OK':
                  mail_ids = data[1]
                  id_list = mail_ids[0].split()
                  first_email_id = int(id_list[0])
                  latest_email_id = int(id_list[-1])
                  for i in range(latest_email_id, first_email_id, -1):
                        data = mail.fetch(str(i), '(RFC822)' )
                        for response in data:
                              arr = response[0]
                              if isinstance(arr, tuple):
                              msg = email.message_from_string(str(arr[1],'utf-
8'))
                                    email_from = msg['from']
                                    email_subject = msg['subject']
                                    for j in msg.walk():
                                          if j.get_content_maintype() ==
'multipart':
                                                continue
                                          if j.get('Content-Disposition') is
None:
                                                continue
                                          file = j.get_filename()
                                          if bool(file):
                                          filePath =
                                                      os.path.join(d['path_
                                                      invoices'], file)
                                          if not os.path.isfile(filePath) :
                                          fp = open(filePath, 'wb')

                                          fp.write(j.get_payload(decode=True))
                                          fp.close()
            else:
                  fecha = datetime.now()
```

```
                  error = "[" + str(fecha) + ", Error al conectar con servidor
del correo]"
                  main.write_error(error)
      except Exception as e:
            fecha = datetime.now()
            error = "[" + str(fecha) + ", Error de codificacion: " + str(e) +
"]"
            main.write_error(error)

      finally:
            mail.close()
            mail.logout()


def send_email_report(): #funcion que envia un correo adjuntando un ARCHIVO
EXCEL
      msg = MIMEMultipart()
      gmail_from = main.read_correos()
      msg['From'] = gmail_user
      msg['To']   = ','.join(gmail_from)
      #msg['Date'] = formatdate(localtime = True)
      fecha = date.today()
      fecha = fecha.strftime('%d/%m/%Y')

      msg['Subject'] = 'BOT DE REGISTRO DE FACTURAS'
      msg.attach(MIMEText('Envio de registro del dia ' + fecha))
      part = MIMEBase('application', "octet-stream")
      part.set_payload(open(os.path.join(d['path_xlsx'], 'registros.xlsx'),
"rb").read())
      encoders.encode_base64(part)
      part.add_header('Content-Disposition', 'attachment; filename="Registros
de Facturas.xlsx"')
      msg.attach(part)
      server = smtplib.SMTP('smtp.gmail.com: 587')
      server.starttls()
      server.login(gmail_user, gmail_pass)
      server.sendmail(gmail_user, gmail_from, msg.as_string())
      server.quit()
      return 1
```

C2. Script *conexión_sqlite.py* que contiene las funciones que permiten la conexión y almacenamiento en la Base de Datos *Master*.

```
def sqlite_conn():
      try:
            global conn
            conn = sqlite3.connect('master.db')
            return conn
      except Error:
            print(Error)

def sqlite_insert(conn, d):
      cursorObj = conn.cursor()
```

```
      values_invoice = (d['ruc_emisor'], d['serie'], d['numero'], d['fecha'],
                  d['moneda'], d['total'],  d['igv'])
      cursorObj.execute('INSERT INTO invoice(ruc, serie, numero, fecha,
                     moneda, total, igv)     VALUES(?, ?, ?, ?, ?, ?, ?)',
                     values_invoice)
      cursorObj.execute('SELECT id_invoice FROM invoice ORDER BY id_invoice
DESC LIMIT 1')
      id_invoice = cursorObj.fetchall()[0][0] #id de la factura

      for i in d['items']:
            values_item = (i['cantidad'], i['producto'], i['precio_unit'],
id_invoice)
            cursorObj.execute('INSERT INTO item(cantidad, descripcion, precio,
                        id_invoice_item) VALUES(?, ?, ?, ?)', values_item)


def sqlite_close():
      conn.commit()
      conn.close()
```

C3. Script *main.py* que es el código principal ya que desde aquí se hace el llamado a los script *conexión_sqlite* y *conexión_correo.py.*

```
def write_error(error):
      with open(d['path_log'], 'a') as f:
            f.write('\n' + error)
            f.close()

def read_correos(): # función que lee los correos que están en un bloc
      with open(d['path_email'], 'r') as f:
            correos = f.read()
            correos = correos.split('\n')
            f.close()
            return correos

def xml_to_dict(path_xml, encoding): # función que convierte xml en
diccionario(tipo de dato estructurado)
      try:
            if not encoding:
                  encoding = "latin-1"
            with open(path_xml, encoding = encoding) as fd:
                  doc = xmltodict.parse(fd.read())
                  res = json.loads(json.dumps(doc))
      except:
            res = {}
      return res

def extract_xml(d): # funcion que extrae data de diccionario
      invoice_tmp = {   'factura' : None,  'serie' : None,   'numero' : None,
'fecha' : None,   'hora_emision' : None,
                        'moneda' : None, 'ruc_emisor'  : None, 'ruc_cliente'
: None, 'total' : None,  'igv'  : None, 'items': []  }
items_tmp   = {     'cantidad' : None, 'producto' : None, 'precio_unit' :None
}
```

```
      try:  #datos de la factura electronica
            invoice = invoice_tmp.copy()
            invoice['factura']        =
d['Invoice']['cbc:InvoiceTypeCode']['#text']
            invoice['serie']          = d['Invoice']['cbc:ID'].split('-')[0]

            invoice['numero']         = d['Invoice']['cbc:ID'].split('-')[1]

            invoice['numero']             =   ['0'*(6-len(invoice['numero']))
                        + invoice['numero'] if len(invoice['numero']) < 6
                        else invoice['numero']][0]
            invoice['fecha']                  = d['Invoice']['cbc:IssueDate']

            invoice['hora_emision']   = d['Invoice']['cbc:IssueTime']

            invoice['moneda']             =
                        d['Invoice']['cbc:DocumentCurrencyCode'] if
                        type(d['Invoice']['cbc:DocumentCurrencyCode']) ==
                        str else
                        d['Invoice']['cbc:DocumentCurrencyCode']['#text']
            # datos del emisor
            invoice['ruc_emisor']    =
d['Invoice']['cac:AccountingSupplierParty']['cac:Party']['cac:PartyIdentificat
ion']['cbc:ID']['#text'] if 'cac:PartyIdentification' in
d['Invoice']['cac:AccountingSupplierParty']['cac:Party'] else
d['Invoice']['cac:AccountingSupplierParty']['cbc:CustomerAssignedAccountID']
            #datos del cliente
            invoice['ruc_cliente']   =
d['Invoice']['cac:AccountingCustomerParty']['cac:Party']['cac:PartyIdentificat
ion']['cbc:ID']['#text'] if 'cac:PartyIdentification' in
d['Invoice']['cac:AccountingCustomerParty']['cac:Party'] else
d['Invoice']['cac:AccountingCustomerParty']['cbc:CustomerAssignedAccountID']
            #totales
            invoice['total'] =
round(float(d['Invoice']['cac:LegalMonetaryTotal']['cbc:PayableAmount']['#text
']),2)
            invoice['igv'] =
round(float(d['Invoice']['cac:TaxTotal']['cbc:TaxAmount']['#text']),2)
            #datos de items
            items = [d['Invoice']['cac:InvoiceLine']] if
                 type(d['Invoice']['cac:InvoiceLine']) != list else
                 d['Invoice']['cac:InvoiceLine']
            num_items = len(items)
            for i in items:
                  items_invoice = items_tmp.copy()
                  items_invoice['cantidad']    =
float(i['cbc:InvoicedQuantity']['#text'])
                  items_invoice['producto']    =
i['cac:Item']['cbc:Description']
                  items_invoice['precio_unit'] =
round(float(i['cac:Price']['cbc:PriceAmount']['#text']), 2)
                  invoice['items'].append(items_invoice)
      except:
            invoice = {}
      return invoice
```

```
def validations(d): # funcion que realiza las validaciones a la data de la
factura
      patters = {     'factura'    : '(01)',
                      'serie'      : '(F.{3})',
                      'numero'     : '(\d{6})',
                      'fecha'      : '(20\d{2}-\d{2}-\d{2})',
                      'moneda'     : '(PEN|USD|EUR)',
                      'ruc_emisor' : '(\d{11})',
                      'ruc_cliente': '(\d{11})'}
      if len(d) == 0:
            d = {}
            flags = False
      else: #primera validacion
            flag_1 = all([ i != None for i in d.values()])
            flag_2 = all([j != None for i in d['items'] for j in i.values()])
            flag = all([flag_1, flag_2])
            if flag: #segunda validacion
                  list_flag = list()
                  for key, value in patters.items():
                        mo = re.search(value, d[key])
                        if mo:
                              list_flag.append(True)
                              #valor = mo.group(1)
                  flags = all(list_flag)
            else:
                  flags = False
                  d = {}
      return flags, d

def rgbToInt(rgb): # función que obtiene rgb de color
      colorInt = rgb[0] + (rgb[1] * 256) + (rgb[2] * 256 * 256)
      return colorInt

def main():
      if os.path.isdir(d['path_xlsx']):
            shutil.rmtree(d['path_xlsx'])
            os.mkdir(d['path_xlsx'])
      else:
            os.mkdir(d['path_xlsx'])
      excel = win32.gencache.EnsureDispatch("Excel.Application")
      excel.Visible = True
      book = excel.Workbooks.Add()
      workSheet = book.Worksheets('Hoja1')
      etiquetas = ['Estado', 'Archivo', 'Serie', 'Numero', 'Fecha',
             'Moneda','Emisor', 'Cliente', 'Total', 'IGV', 'Cantidad',
             'Producto', 'Precio unitario']
      for c, e in enumerate(etiquetas, 1):
            workSheet.Cells(1, c).Value = e
            workSheet.Cells(1, c).Interior.ColorIndex = 45
      lastRow = workSheet.UsedRange.Rows.Count
      fila = lastRow + 1
      conexion_email.read_email_from_gmail()
      list_xml = os.listdir(d['path_invoices'])
      if len(list_xml) == 0:
            error = '[No se encontro archivos para trabajar]'
      else:
            con = conexion_sqlite.sqlite_conn()
```

```
            for i in list_xml:
                  try:
                        dict_invoice = xml_to_dict(d['path_invoices'] + i,
'utf-8')
                  except:
                        dict_invoice = xml_to_dict(d['path_invoices'] + i,
'ISO-8859-1')
                  accurate, invoice = validations(extract_xml(dict_invoice))
                  if accurate:
                        try:
                              conexion_sqlite.sqlite_insert(con, invoice)
                              for k, j in enumerate(invoice['items']):
                                    workSheet.Cells(fila, 1).Value =
'Registrado en BD'
                                    workSheet.Cells(fila, 2).Value = i
                                    workSheet.Cells(fila, 3).Value =
invoice['serie']
                                    workSheet.Cells(fila, 4).Value =
invoice['numero']
                                    workSheet.Cells(fila, 5).Value =
invoice['fecha']
                                    workSheet.Cells(fila, 6).Value =
invoice['moneda']
                                    workSheet.Cells(fila, 7).Value =
invoice['ruc_emisor']
                                    workSheet.Cells(fila, 8).Value =
invoice['ruc_cliente']
                                    workSheet.Cells(fila, 9).Value =
invoice['total']
                                    workSheet.Cells(fila, 10).Value =
invoice['igv']
                                    workSheet.Cells(fila, 11).Value =
j['cantidad']
                                    workSheet.Cells(fila, 12).Value =
j['producto']
                                    workSheet.Cells(fila, 13).Value =
j['precio_unit']
                                    if k != 0:
                                    workSheet.Range(workSheet.Cells(fila,
                                    1),workSheet.Cells(fila,13)).Font.Color =
                                    rgbToInt((127,127,127))
                                    fila += 1
                        except:
                              workSheet.Cells(fila, 1).Value = 'Error al
registrar'
                              workSheet.Cells(fila, 2).Value = i
                              fila += 1
                  else:
                        workSheet.Cells(fila, 1).Value = 'Error de estructura
XML'
                        workSheet.Cells(fila, 2).Value = i
                        fila += 1
            book.SaveAs(os.path.join(d['path_xlsx'], 'registros.xlsx'))
            book.Close()
            excel.Quit()
            conexion_sqlite.sqlite_close()
            conexion_email.send_email_report()
```