ENTREGABLE 2-B

**MATRIZ DE RIESGO,**

**REGLAS, INDICADORES Y PERFILES DE RIESGOS DE LA DGA**



**Ventanilla Única de Comercio Exterior 2.0**

**Buzón 2.0**

**Documento Manual Técnico Desarrollo**

**Versión 5.0**

**20 de diciembre de 2024**

#### Historial de Versiones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Elaborado por** | **Revisado por** |
| **1.0** | 22/12/2024 | Estructuración del documento | Hugo rosales | Marco silva |

# Contenido

Contenido

[Contenido 3](#_Toc186793312)

[Índice de tablas 5](#_Toc186793313)

[Índice de figuras 6](#_Toc186793314)

[Introducción 7](#_Toc186793315)

[1. Objetivo 7](#_Toc186793316)

[2. Alcance 7](#_Toc186793317)

[3. Glosario de Términos 7](#_Toc186793318)

[4. Infraestructura Tecnológica 8](#_Toc186793319)

[4.1 Sistema Operativo de los servidores 8](#_Toc186793320)

[4.2 Sistemas de Gestión de Base de Datos 9](#_Toc186793321)

[4.2.1 Base de Datos Transaccional ORACLE 9](#_Toc186793322)

[4.2.1 Base de Datos NOSQL MONGO 9](#_Toc186793323)

[4.3 Servidor LINUX 9](#_Toc186793324)

[4.4 Lenguajes de Codificación 10](#_Toc186793325)

[5. Arquitectura del sistema 10](#_Toc186793326)

[5.1 Arquitectura General 11](#_Toc186793327)

[5.2 Arquitectura para Despliegue 11](#_Toc186793328)

[6. Marco de trabajos, librerías y estándares 15](#_Toc186793329)

[6.1 Marco de Trabajo 15](#_Toc186793330)

[6.1.1 Marco de Trabajo para los procesos de migración 15](#_Toc186793331)

[6.2 Librerías 19](#_Toc186793332)

[6.2.1 Librerías para los programas Phyton 20](#_Toc186793333)

[6.3 Estándares 21](#_Toc186793334)

[7. Estructuración de código fuente 22](#_Toc186793335)

[7.1 Shell de código fuente para los procesos de migración 22](#_Toc186793336)

[7.2 Programa de código fuente de la migración 23](#_Toc186793337)

[7.3 Programa de código fuente de la Carga de Adjuntos 24](#_Toc186793338)

[7.4 Proyectos ubicación con GIT 24](#_Toc186793339)

[8. Herramienta para la construcción y gestión del proyecto 24](#_Toc186793340)

[8.3 Estructura de proyecto de código fuente 25](#_Toc186793341)

[9. Modelo de Datos 26](#_Toc186793342)

[9.4 Diccionario de Datos 26](#_Toc186793343)

[10. Funcionalidad del Sistema 27](#_Toc186793344)

[11. Base de datos 27](#_Toc186793345)

[12. Programas 27](#_Toc186793346)

[12.1.1 Programas de Alto Nivel de todo el sistema 27](#_Toc186793347)

[12.1.2 Diagrama de programas del PROCESO ANUAL y MENSUAL 27](#_Toc186793348)

[13. Procesos de MIGRACION -REPROCESO 29](#_Toc186793349)

[13.1 Creación de temporales para migrar 29](#_Toc186793350)

[13.2 CARGA DE ADJUNTO AL FILENET 30](#_Toc186793351)

[13.3 Migración de mensajes a mongo buzón 33](#_Toc186793352)

[13.4 Generar mensajes migrados para la colección envíos 34](#_Toc186793353)

[13.5 Agregar mensajes a la colección envió de mongo 34](#_Toc186793354)

[Anexo D: Diccionario de Datos 35](#_Toc186793355)

# Índice de tablas

[Tabla 1 - Glosario de Términos 7](#_Toc180069226)

[Tabla 2 - Servicio - Sistema Operativo de Servidor 9](#_Toc180069227)

[Tabla 3 - Esquema de BD 10](#_Toc180069228)

[Tabla 4 - Resumen de herramienta para la implementación de código 11](#_Toc180069229)

[Tabla 5 - Ambiente de Desarrollo 13](#_Toc180069230)

[Tabla 6 - Ambiente de Certificación 15](#_Toc180069231)

[Tabla 7 - Ambiente de Producción 16](#_Toc180069232)

[Tabla 8 - Tabla de Librerías principales de los proyectos de microservicios 23](#_Toc180069233)

[Tabla 9 - Estándares 24](#_Toc180069234)

[Tabla 10 - Librerías comunes buzon 25](#_Toc180069235)

# Índice de figuras

[Figura 1 - Arquitectura General 11](#_Toc186793293)

[Figura 2 - Diagrama Arquitectura Física Ambiente de Certificación 12](#_Toc186793294)

[Figura 4 - Diagrama Arquitectura Física Ambiente de Producción 14](#_Toc186793295)

[Figura 7 - Estructura de proyecto de código fuente microservicios 25](#_Toc186793296)

[Figura 10- Diagrama de bUZON (TS) 26](#_Toc186793297)

[Figura 11- Diagrama de componentes de Alto Nivel de todo el Sistema 27](#_Toc186793298)

[Figura 12- Diagrama de componentes del proceso de carga Anual 28](#_Toc186793299)

[Figura 13- Diagrama de Ejecución 29](#_Toc186793300)

# Introducción

La migración a Buzon2 se realizó de los mensajes y adjuntos de Buzon1 que estaban en Oracle 11g y fue migrado a

Mongodb 7.0 y Filenet respectivamente, donde se usó Python 3.6 y java 11.0 en las fases de carga de la migración.

# Objetivo

* Detallar las características físicas y técnicas de los componentes del sistema que están configurados en la arquitectura de migración.
* Brindar información técnica acerca del proceso de migración representado a nivel conceptual, físico y lógicos; además se agrega el diccionario de datos con las estructuras de tablas de Base de datos.

# Alcance

Este documento proporciona los aspectos técnicos e informáticos del desarrollado para que los usuarios que sean asignados para dar mantenimiento al sistema lo hagan de forma idónea conociendo los requerimientos y la arquitectura utilizada para la construcción del sistema, la infraestructura tecnológica usada, el modelo que soporta los datos, la forma en que se estructura el código fuente, como se establece la conexión a base de datos, entre otros. Es importante tener en cuenta que en el presente manual se mencionan las especificaciones de hardware, software y herramienta que se utilizaron.

A como se menciona anteriormente este manual está orientado al personal técnico delegado para el mantenimiento del sistema por lo cual se espera que las persona a cargo tenga las habilidades y conocimientos necesarios para poder entender y hacer suyo la información brindada; teniendo además los permisos y autorizaciones necesarias.

# Glosario de Términos

Tabla 1 - Glosario de Términos

1. *Glosario de Términos*

| **ACRÓNIMO/CONCEPTO** | **DESCRIPCIÓN** |
| --- | --- |
| PY | Los archivos Python son programas para ser ejecutados en el ambiente Linux u windows. |
| SH | Los archivos batcheros Linux para ejecutar asíncronamente los procesos. |
| BD | Base de Datos. La base de datos es un conjunto de datos estructurados que pertenecen a un mismo contexto que se utiliza para administrar de forma digital grandes cantidades de información. |
| Esquema | Un esquema de base de datos representa la configuración lógica de una parte de una base de datos relacional. |
| VUCE 2.0 | Sistema Ventanilla Única de Comercio Exterior versión 2.0. |

# 

# Infraestructura Tecnológica

La Infraestructura Tecnológica usada es On-Premises donde la instalación del sistema es dentro de una ubicación física brindada por VUCE - MINCETUR donde es un entorno de TI in-house, dado a que el centro de datos se encuentra en las instalaciones de la entidad donde el equipo de TI tiene control y administración sobre los equipos, servidores y servicios usados.

La principal ventaja de la infraestructura On-Premises es lo fácil que es de controlar ya que, al residir dentro de las propias instalaciones de la entidad, el personal a cargo puede acceder a ella de forma directa, controlando cada uno de sus componentes decidiendo qué hacer con ellos y cómo disponerlos.

De lo anterior también se adiciona que todos los datos almacenados de la lógica de negocio de la VUCE se encuentran sobre la infraestructura, sin que haya terceros que puedan acceder a ella, lo cual se vuelve sumamente valioso dado a que la información es de carácter sensible.

## 4.1 Sistema Operativo de los servidores

Un Sistema Operativo de Servidor es aquel que es usado en un servidor el cual contiene un conjunto de procesos, programas y utilidades para gestionar los recursos de hardware y software; y proporcionar servicios para ser usados por uno o varios sistemas(s).

A continuación, se presenta los sistemas operativos conforme los servicios que son usados por el proceso de migracion:

Tabla 2 - Servicio - Sistema Operativos

| **SERVICIO** | **S. OPERATIVO** |
| --- | --- |
| **Sistema de Gestión de Base de Datos Relacional** | Red Hat Enterprise Linux 8.5 (Ootpa) |
| **Sistema de Gestión de Base de Datos NoSQL** | Red Hat Enterprise Linux 8.5 (Ootpa) |
| **Plataforma de Gestión Documental** | Red Hat Enterprise Linux 7.6 (Maipo) |

## Sistemas de Gestión de Base de Datos

### 4.2.1 Base de Datos Transaccional ORACLE

La base de datos transaccional de Transmisiones está definida en ORACLE en su edición EnterpriseDB Advanced Server en su versión 11.2. Esta base de datos tiene el nombre de **vucepr.vuce.gob.pe**, la cual contiene varios esquemas para segmentar los objetos tablas, secuencias, vistas e índices. A continuación, se presentan los esquemas de la base de datos:

Esquemas de BD

Tabla 3 - Esquema de BD oracle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nro.** | **ESQUEMA** | **ROLE PROPIETARIO** | **ROLE ACCESO** | **USO** | **DESCRIPCION** |
| 1 | **REPLICADO** | SYS | DBA | Sistema | Esquema para las tablas, secuencias, vistas e índices del catálogo MR |

### 4.2.1 Base de Datos NOSQL MONGO

La base de datos de Buzón está definida en Mongo 7.0.0. Esta base de datos tiene el nombre de **vuce-buzonelectronico**, la cual contiene varios esquemas para segmentar los objetos colecciones e índices. A continuación, se presentan los esquemas de la base de datos:

Esquemas de BD

Tabla 4 - Esquema de BD mongo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nro.** | **ESQUEMA** | **ROLE PROPIETARIO** | **ROLE ACCESO** | **USO** | **DESCRIPCION** |
| 1 | **vuce-buzonelectronico** | EBALDEON | DBA | Sistema | Esquema para las tablas, secuencias, vistas e índices del catálogo MR |

### 4.2.1 Base de Datos DB2

La base de datos de FILENET está definida en DB2. Esta base de datos tiene el nombre de **DB2INST1**, la cual contiene varios esquemas para segmentar los objetos colecciones e índices. A continuación, se presentan los esquemas de la base de datos:

Esquemas de BD

Tabla 4 - Esquema de BD mongo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nro.** | **ESQUEMA** | **ROLE PROPIETARIO** | **ROLE ACCESO** | **USO** | **DESCRIPCION** |
| 1 | **DB2INST1** | WSADMIN | DBA | Sistema | Esquema para las tablas, secuencias, vistas e índices del catálogo |

## Servidor LINUX

**LINUX**, es un servidor de procesos de código abierto de alto rendimiento licenciado bajo la licencia BSD simplificada. por lo que corre en sistemas tipo Unix (GNU/Linux)

Las características básicas que tiene este servidor son:

Procesos SHELL para la migración.

* Servidor de archivos estáticos, índices y autoindexado.
* Proxy inverso con opciones de caché.
* Balanceo de carga.
* Tolerancia a fallos.
* Soporte de HTTP y HTTP2 sobre SSL.
* Soporte para FastCGI con opciones de caché.
* Servidores virtuales basados en nombre y/o en dirección IP.
* Streaming de archivos FLV y MP4.24​
* Soporte para autenticación.
* Compatible con IPv6.
* Soporte para protocolo SPDY.
* Compresión gzip.

## 4.4 Lenguajes de Codificación

Para la codificación de MIGRACION BUZON se utilizó:

Tabla 4 - Resumen de herramienta para la implementación de código

| TIPO | NOMBRE | DESCRIPCION | LOGO |
| --- | --- | --- | --- |
| Entorno de desarrollo | Visual Studio Code v1.71 | Es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Tiene soporte para operaciones de desarrollo como depuración, ejecución de tareas y control de versiones. |  |
| IntelliJ IDEA v2021.3.2 | Es un IDE inteligente y sensible al contexto para trabajar con Java y otros lenguajes JVM como Kotlin, Scala y Groovy en todo tipo de aplicaciones. |  |
| Lenguajes | Java vOpenJDK v17 | Es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web creado por Sun Microsystems en 1995. Se emplea Java para la implementación de los proyectos de los microservicios del backend del sistema. |  |
| Lenguajes | Python 3.6 | **Python** es un lenguaje de programación de alto nivel y ampliamente utilizado, creado por Guido van Rossum y lanzado por primera vez en 1991. Se emplea en una variedad de aplicaciones, incluidas aplicaciones web, análisis de datos, inteligencia artificial y desarrollo de microservicios. Python es conocido por su sintaxis clara y su fuerte soporte para bibliotecas y frameworks, lo que lo convierte en una opción popular para la implementación de proyectos de microservicios y backend en sistemas modernos. |  |

# Arquitectura del sistema

Los patrones, buenas prácticas, RNFs y la definición de la arquitectura de la solución para BUzon 2.0 se encuentran detalladas en el documento VUCE 2-Documento Arquitectura-Buzon 2.0.v1.0.

## 5.1 Arquitectura General

La Arquitectura General de la solución está representada por un conjunto de diagramas y especificaciones de componentes lógicos y procesos que permiten tener una visión general y también detallada de la solución.

En la sección 3.1 Vista de Arquitectura General del documento de Arquitectura, se puede identificar el diagrama general de la Arquitectura y de los componentes tanto a nivel de aplicación como de los procesos:

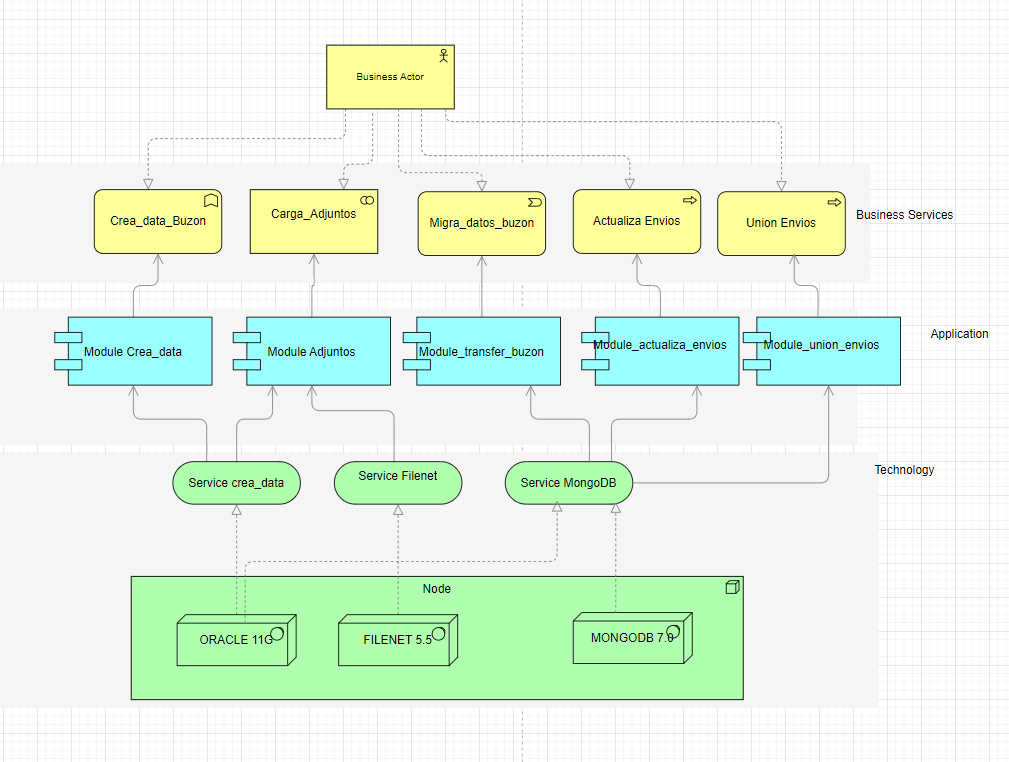


Figura 1 - Arquitectura General

## 5.2 Arquitectura para Despliegue

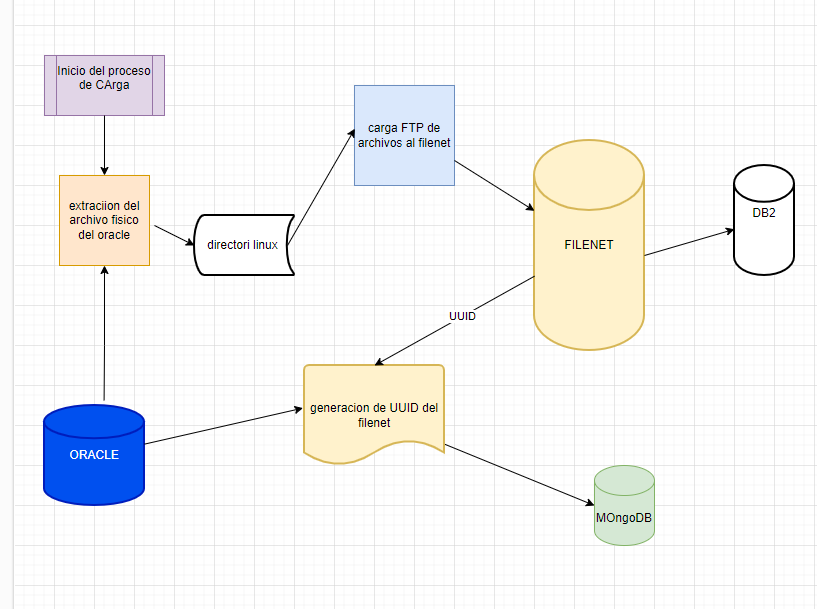
La Arquitectura para la ejecución de la migración representa la relación de todos los componentes físicos o hardware que intervienen para su correcto funcionamiento a partir de lo definido en la Arquitectura General.

La descripción de los ambientes de infraestructura donde estará ejecutándose solo está en Replicado de producción porque no se cuenta con

Los componentes de la solución se encuentran detallado en el documento de Arquitectura:

VUCE2-DocumentoArquitectura-Buzon2.0 v1.0, en la sección 3.4 Vista de Infraestructura.

Figura 2 - Diagrama Arquitectura Física Ambiente de Certificación



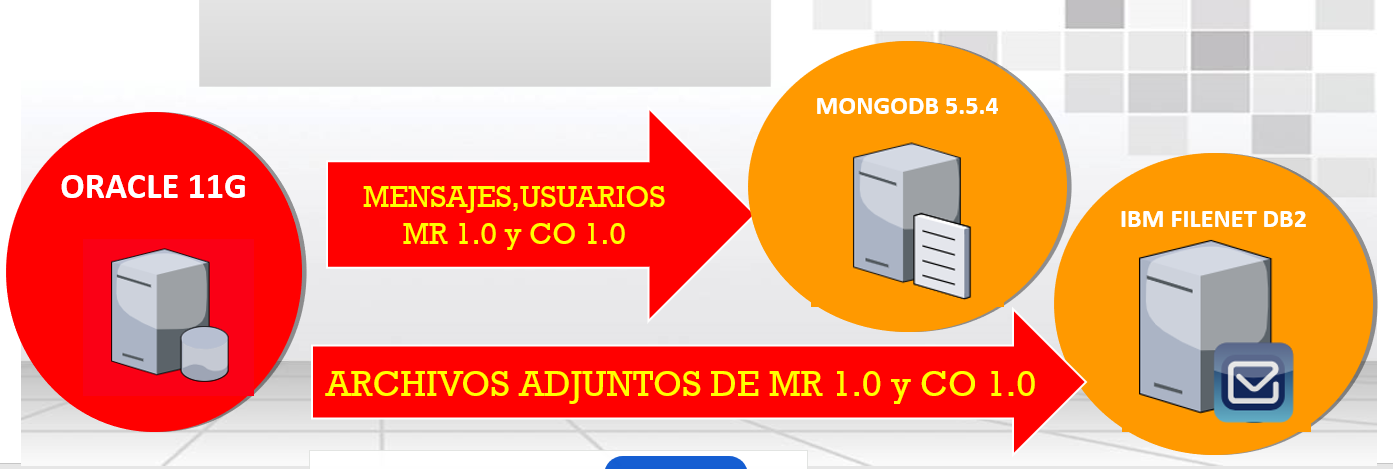
**Ambiente de Certificación**

|  |
| --- |
| Componentes |
| |  |  | | --- | --- | | Plataforma de LINUX | | | Consola terminal | 10.19.34.190 | | Path | /home/admin/buzon-migracion/carga | |  |  | |  |  | |  |  | |
| |  |  | | --- | --- | | SERVIDOR FILENET | | | Dirección IP | 192.168.120.67 | | URL | <http://192.168.120.67:9080/navigator/?desktop=ProcessNavigatorQA>  http://192.168.120.67:9080/acce/ | |  |  | | |
| |  |  | | --- | --- | | Base de datos en ORACLE | | | Dirección IP | 192.168.8.171 | | Puerto | 1521 | |  |  | |
| |  |  | | --- | --- | | Base de datos MongoDB | | | Dirección IP | 192.168.140.40 alias host: atcertbd-mongodb01.vuce.gob.pe | |  | 192.168.140.41 alias host: atcertbd-mongodb02.vuce.gob.pe | |  | 192.168.8.23 alias host: mtcertbd-drpmongodb01.vuce.gob.pe | | Puerto | 27017 | |  |  | |

**Ambiente de Producción**

|  |
| --- |
| Componentes |
| |  |  | | --- | --- | | Plataforma de LINUX | | | Consola terminal | 10.19.34.190 | | Path | /home/admin/buzon-migracion/carga | |  |  | |  |  | |  |  | |
| |  |  | | --- | --- | | SERVIDOR FILENET | | | Dirección IP | 192.168.120.16 | | URL | <http://192.168.120.16:9080/navigator/?desktop=ProcessNavigatorPRD>  http://192.168.120.16:9080/acce/ | |  |  | | |
| |  |  | | --- | --- | | Base de datos en ORACLE | | | Dirección IP | 192.168.8.171 | | Puerto | 1521 | |  |  | |
| |  |  | | --- | --- | | Base de datos MongoDB | | | Dirección IP | 192.168.110.42 alias host: atprodbd-mongodb01.vuce.gob.pe | |  | 192.168.110.43 alias host: atprodbd-mongodb02.vuce.gob.pe | |  | 192.168.8.42 alias host: mtprodbd-drpmongodb01.vuce.gob.pe | | Puerto | 27017 | |  |  | |

Figura 4 - Diagrama Arquitectura Física Ambiente de Producción



# Marco de trabajos, librerías y estándares

El lenguaje de programación usado para los procesos batch de MIGRACION BUZON es Python 3.6; un lenguaje de programación de alto nivel y ampliamente utilizado, creado por Guido van Rossum y lanzado por primera vez en 1991. Se emplea en una variedad de aplicaciones, incluidas aplicaciones web, análisis de datos, inteligencia artificial y desarrollo de microservicios. Python es conocido por su sintaxis clara y su fuerte soporte para bibliotecas y frameworks, lo que lo convierte en una opción popular para la implementación de proyectos de microservicios y backend en sistemas modernos.

## 6.1 Marco de Trabajo

Un Marco de Trabajo (FrameWork) es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. En el contexto de desarrollo de sistema un marco de trabajo está relacionado con patrones de diseño implementados en un lenguaje de programación, proveyendo diferentes librerías, funciones, interfaces y api; que un desarrollador podrá usarlas y de esta forma brindar una base para codificar el contenido y las funcionalidades para un sistema.

### 6.1.1 Marco de Trabajo para los procesos de migración

El marco principal que se usa para MIGRACION DE BUZON (PY) son 5 pasos los cuales corren por año

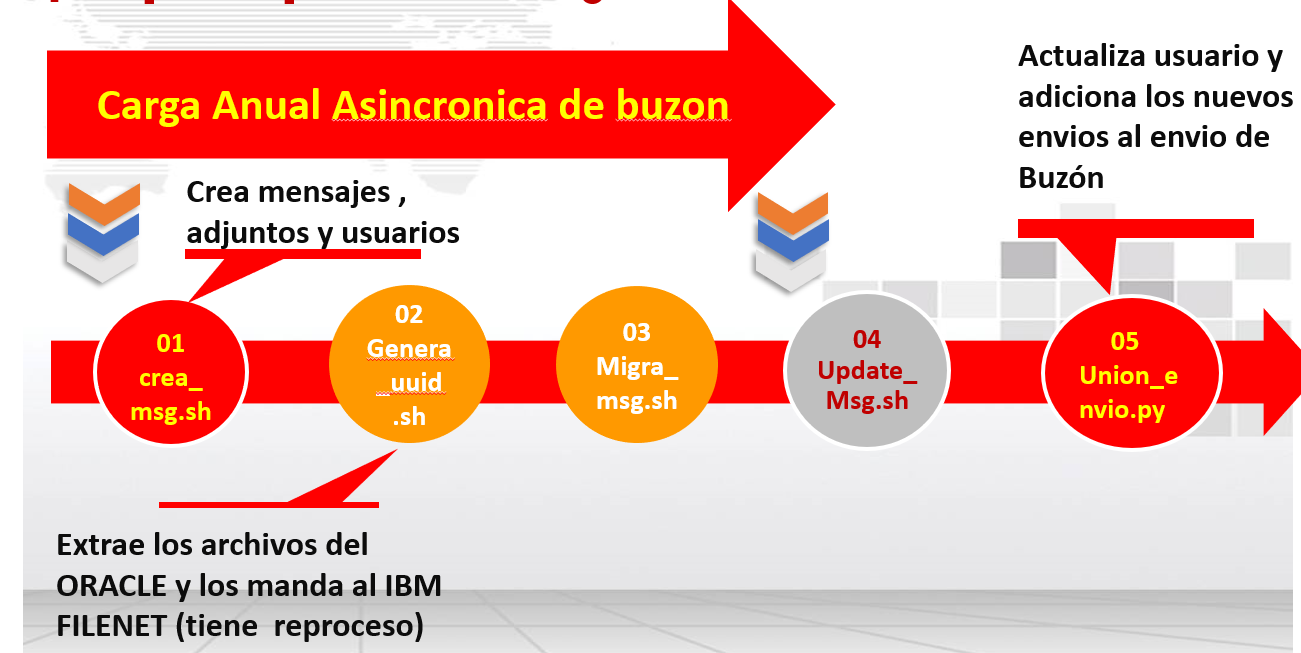
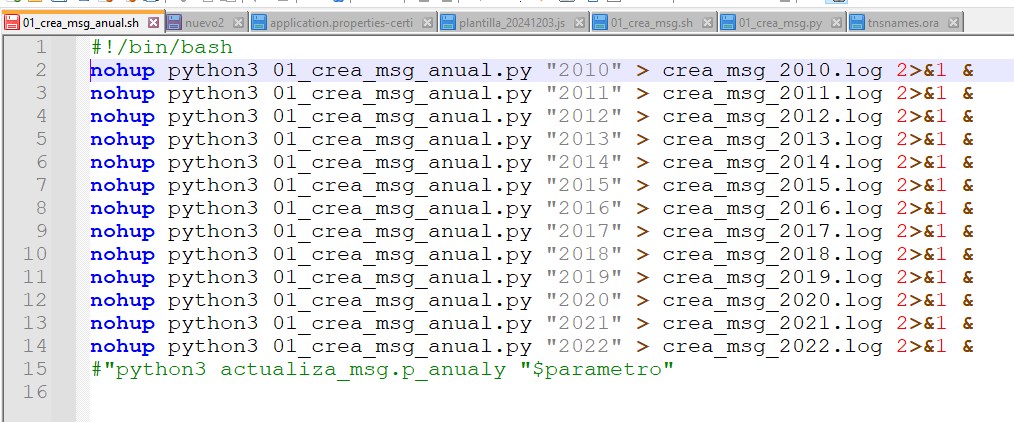


Figura 5 - Diagrama Marco de trabajo

1. **PASO1: CREACION DE TEMPORALES PARA MIGRAR**

Ejecutamos 01\_crea\_msg\_anual.sh (considerar que los temporales A\_YYYY en ORACLE no deben existir)



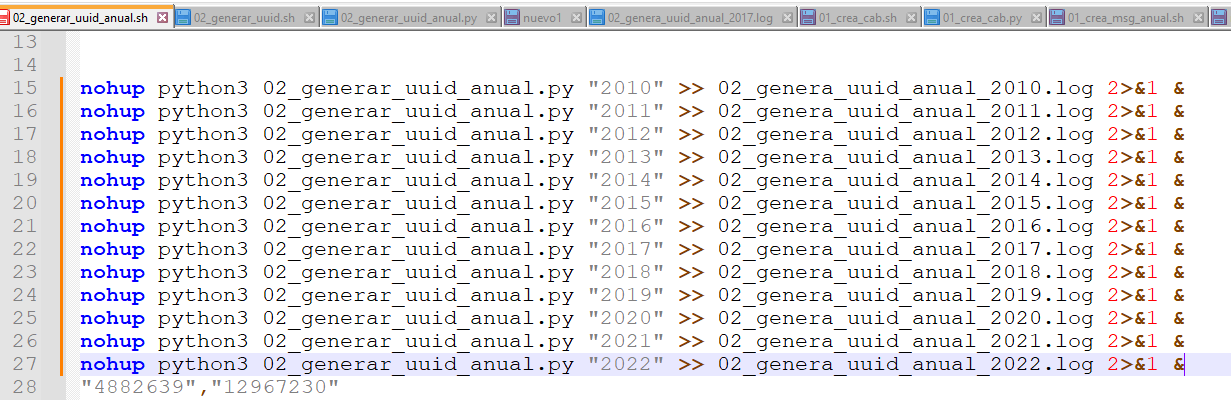
Se validar cuando se vea que en el ORACLE se crean los temporales A\_YYYY, A\_YYYY\_ADJ, A\_YYYY\_TMP

1. **PASO2: CARGA DE ADJUNTO AL FILENET**

Ejecutar la carga de archivos de filenet: 02\_generar\_uuid\_anual.sh

Debe validar la tabla A\_YYYY\_ADJ tenga el campo UUID en blanco

(útil para el reproceso)

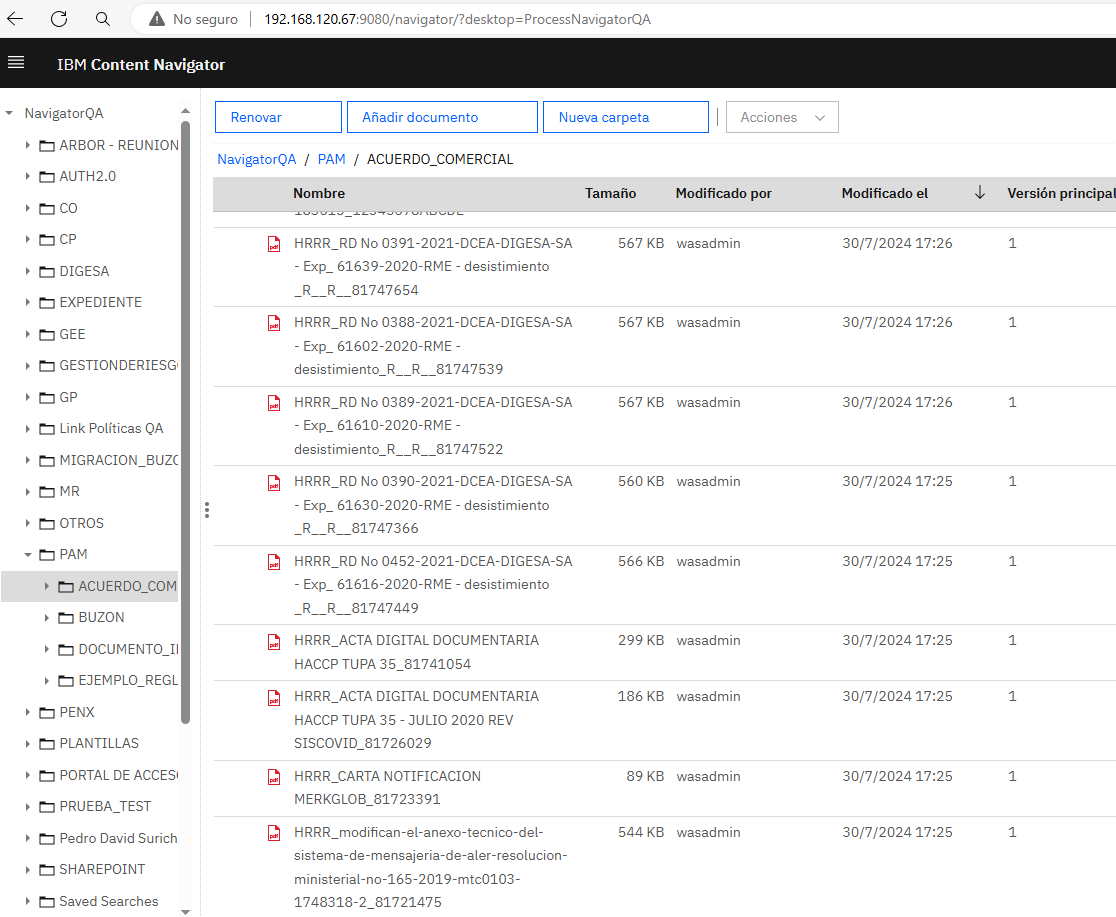


Se validará VERIFICANDO EN EL FILENET en la carpeta BUZON los archivos manualmente

el log, usaremos este query

select \* from **VCOBJ.A\_202101\_ADJ** **t** where trim(**UUID**) is null

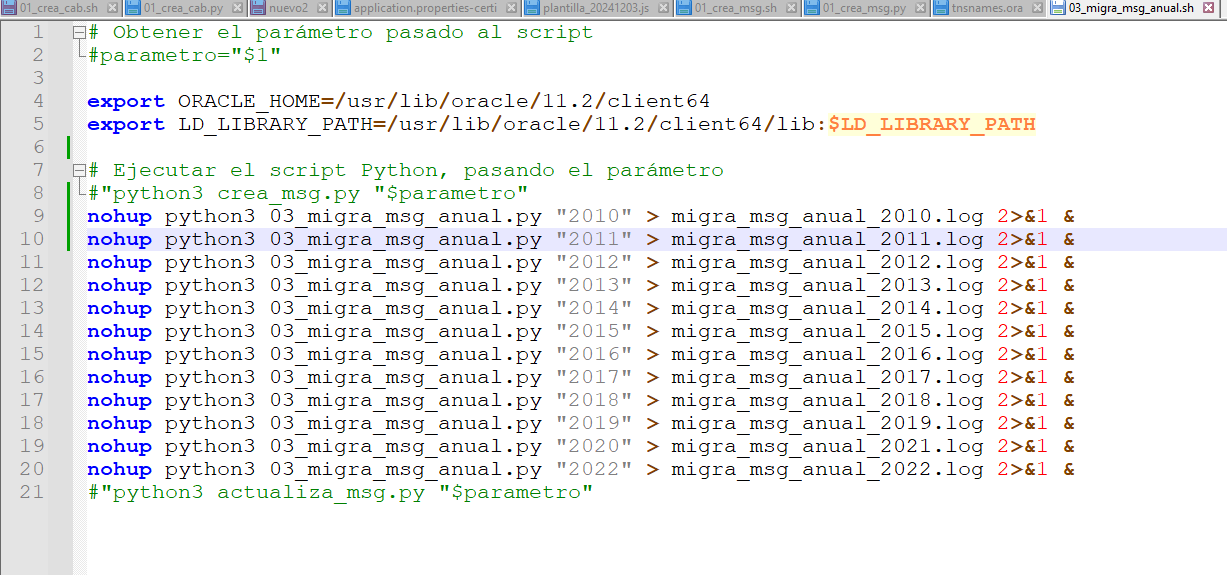
se verán los registros que se reprocesarán y el query debe llegar a estar sin registros y ahí se pasara al siguiente paso

 actualizar

para el reproceso solo se manda ejecutar otra vez igual el proceso el solo tomara los registros que le falta cargar

1. **PASO3: MIGRACION DE MENSAJES A MONGO BUZON**

Ejecutar el comando 03\_migra\_msg\_anual.sh para traer los temporales al MONGODB

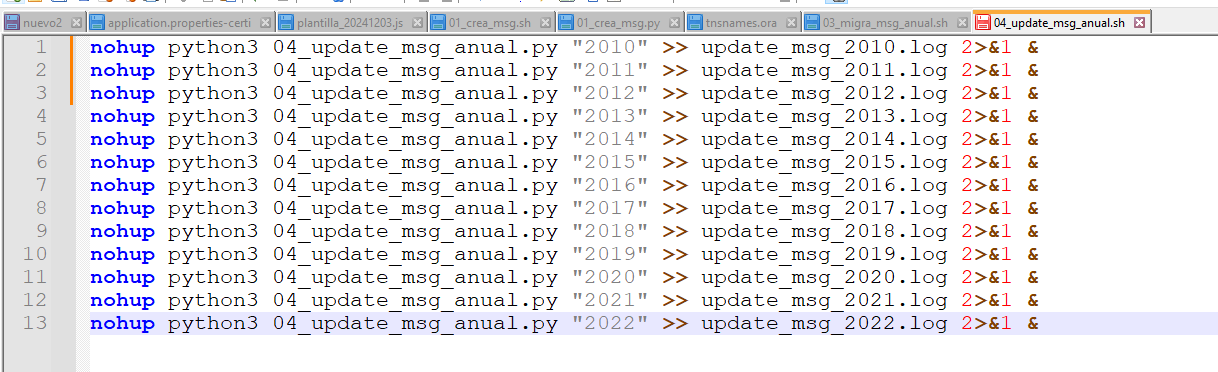


Para validar si se ejecutó correctamente debe validar la existencia de la colección A\_YYYY se encuentra en MONGODB

1. **PASO4: GENERAR MENSAJES MIGRADOS PARA LA COLECICON ENVIOS**

Ejecutar el comando 04\_update\_msg.sh para preparar la data migrada y formatear con la estructura deseada para el MONGODB

Pre-requisito es que las maestras deben estar actualizadas en el mongo

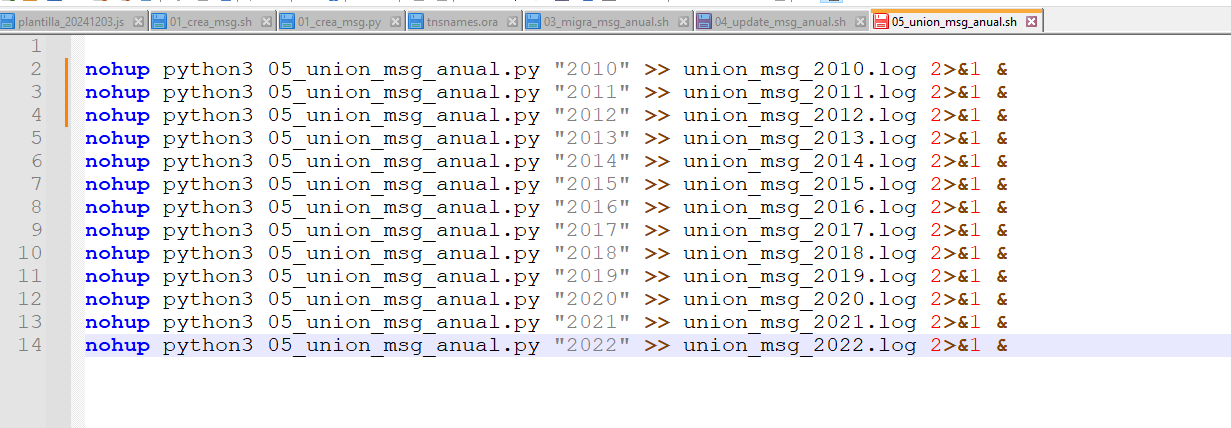


Validar la ejecución de este proceso es verificar la existencia de las colecciones B\_YYYY.

1. **PASO5: AGREGAR MENSAJES A LA COLECCIÓN ENVIO DE MONGO**

Ejecutar el script de Python 05\_union\_msg\_anual.py para con confirmar las cargas de envíos

(los temporales de carga dependerán de lo que se carga)



## 6.2 Librerías

Una librería en Python se puede entender como un conjunto de clases que facilitan operaciones y tareas ofreciendo al desarrollador funcionalidad ya implementada y lista para ser usada través de proceso Python

### 6.2.1 Librerías para los programas Phyton

A continuación, se enumeran las principales librerías que usan los proyectos de microservicios los cuales se pueden localizar en el archivo Python de los programas, se debe de tomar en cuenta que en la columna dependencia sale una cadena que concatena con “:” los siguientes valores

Tabla 8 - Tabla de Librerías principales de los proyectos de microservicios

| **LIBRERÍA** | **DESCRIPCION** | **DEPENDENCIA** |
| --- | --- | --- |
| **Oracledb**  **Driver** | Controlador JDBC de ORACLE | pip install oracledb |
| **os** | el módulo os es una librería estándar que te permite interactuar con el sistema operativo. Puedes usarla para realizar varias tareas como manipular archivos y directorios, obtener información sobre el entorno, gestionar procesos, entre otros. |  |
| **sys** | El módulo sys en Python proporciona funciones y variables que te permiten interactuar con el intérprete de Python y el entorno de ejecución del sistema operativo. Es un módulo muy útil para controlar cómo se ejecuta tu código, acceder a argumentos pasados al script, manejar errores, y más. |  |
| **time** | El módulo time en Python proporciona funciones relacionadas con el tiempo y las fechas. Es muy útil para medir el tiempo de ejecución de programas, manipular fechas y horas, y controlar la temporización de los eventos en tu código. |  |
| **subprocess** | El módulo subprocess en Python se utiliza para crear nuevos procesos, conectar a sus canales de entrada/salida/error, y obtener su código de salida. Esto es muy útil para interactuar con el sistema operativo, ejecutar comandos en la terminal o shell, y manejar tareas externas desde un programa Python. | **subprocess.run()**: |
| **requests** | La librería requests es un módulo popular de Python utilizado para enviar solicitudes HTTP a servidores web. Es comúnmente usada para interactuar con APIs o recuperar datos desde la web. | pip install requests |
| **from requests.auth import HTTPBasicAuth** | El módulo HTTPBasicAuth de la librería requests se utiliza para agregar autenticación básica a tus solicitudes HTTP. La autenticación básica es un método en el que el servidor valida a un usuario mediante un nombre de usuario y una contraseña, enviados en el encabezado de la solicitud. |  |
| **json** | | El módulo json en Python se utiliza para trabajar con datos en formato JSON (JavaScript Object Notation). Este formato es muy común para el intercambio de datos entre servidores y aplicaciones web, debido a su simplicidad y legibilidad. |  |
| **base64** | El módulo base64 en Python proporciona funciones para codificar y decodificar datos en formato Base64. Base64 es un esquema de codificación utilizado para representar datos binarios (como imágenes o archivos) en un formato ASCII. Es comúnmente usado para la transferencia de datos binarios a través de medios que manejan texto, como correos electrónicos o JSON. |  |
|  |  |  |

## 6.3 Estándares

A continuación, los estándares usados para este proyecto:

Tabla 9 - Estándares

| **Id** | **Nombre** | **Referencia** |
| --- | --- | --- |
| [**1**](https://github.com/angular/angular-cli/) | Estándares Desarrollo | VUCE2-LineamientosDesarrollo.docx  <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> |
| [**2**](https://github.com/angular/angular-cli/) | Estándares Base de Datos | Documento de Estándares BD PostgreSQL\_v1.3.pdf  <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> |
| **3** | Arquitectura Patrón | VUCE2-LineamientosTecnologicos.docx  <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> |

# Estructuración de código fuente

## 7.1 Shell de código fuente para los procesos de migración

Las Shells aquellas que son utilizadas para ejecutar los procesos de migración de BUZON (SH), a continuación, se listan:

Tabla 10 – Shell de ejecución de procesos de migración (SH)

| **N.** | | **PROYECTO** | **NOMBRE** | **DEPLOYMENT** | **TIPO** | **LENGUAGE** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Creación de temporales para migrar anual | 01\_crea\_msg\_anual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 2 | | Creación de temporales para migrar mensual | 01\_crea\_msg\_mensual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 3 | carga de adjunto al filenet anual | 02\_genera\_uuid\_anual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 4 | | carga de adjunto al filenet mensual | 02\_genera\_uuid\_mensual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 5 | | migración de mensajes a mongo buzón anual | 03\_migra\_msg\_anual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 6 | | migración de mensajes a mongo buzón mensual | 03\_migra\_msg\_mensual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 7 | | generar mensajes migrados para la colección envíos anual | 04\_update\_msg\_anual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 8 | | generar mensajes migrados para la colección envíos mensual | 04\_update\_msg\_mensual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 9 | | agregar mensajes a la colección envió de mongo | 05\_union\_msg\_anual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |
| 10 | | agregar mensajes a la colección envió de mongo mensual | 05\_union\_msg\_mensual.sh | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | Programa Shell | linux |

## 7.2 Programa de código fuente de la migración

A continuación, se listan los programas de la migración que conforman BUZON (PY):

Tabla 11 - Proyectos de Python de migración de Buzon(PY)

| **Nro.** | **PROYECTO** | **NOMBRE** | **DEPLOYMENT** | **TIPO** | **LENGUAGE** |
| --- | --- | --- | --- | --- | --- |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Creación de temporales para migrar anual | 01\_crea\_msg\_anual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 2 | Creación de temporales para migrar mensual | 01\_crea\_msg\_mensual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 3 | carga de adjunto al filenet anual | 02\_genera\_uuid\_anual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 4 | carga de adjunto al filenet mensual | 02\_genera\_uuid\_mensual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 5 | migración de mensajes a mongo buzón anual | 03\_migra\_msg\_anual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 6 | migración de mensajes a mongo buzón mensual | 03\_migra\_msg\_mensual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 7 | generar mensajes migrados para la colección envíos anual | 04\_update\_msg\_anual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 8 | generar mensajes migrados para la colección envíos mensual | 04\_update\_msg\_mensual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 9 | agregar mensajes a la colección envió de mongo | 05\_union\_msg\_anual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |
| 10 | agregar mensajes a la colección envió de mongo mensual | 05\_union\_msg\_mensual.py | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | .py | python |

## 7.3 Programa de código fuente de la Carga de Adjuntos

A continuación, se listan los programas de la Carga de Adjuntos que conforman (JAVA):

Tabla 11 - Proyectos de Java de Carga de Adjuntos (Java)

| **Nro.** | **PROYECTO** | **NOMBRE** | **DEPLOYMENT** | **TIPO** | **LENGUAGE** |
| --- | --- | --- | --- | --- | --- |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Vuce-filenet-api | Application.java | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | java | Java |
| 2 | Vuce-filenet-api | Filenetservice.java | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | java | Java |
| 3 | Vuce-filenet-api | FilenetConnection.java | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | java | Java |
| 4 | Vuce-filenet-api | FilenetDocumentAbstract.java | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> | java | Java |

## 7.4 Proyectos ubicación con GIT

Para el control de versiones del código fuente se hace uso de GIT, el cual es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Con el propósito de llevar registro de los cambios de los archivos, incluyendo la coordinación del trabajo de las personas realizan sobre estos al compartir el repositorio de código fuente.

Se debe de mencionar que se utiliza la plataforma de GitLab que además de brindar el sistema de control de versiones basado en GIT también brinda ciertas operaciones de DevOps basadas en integración continua y entrega continua

Tabla 12 - Proyectos de XX - GIT

|  |  |  |  |
| --- | --- | --- | --- |
| **Nro.** | **Proyecto** | **Nombre** | **Ruta GIT** |
| 1 | Buzón-migración | **buzon-migracion** | <https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8> |

# Herramienta para la construcción y gestión del proyecto

Las herramientas utilizadas para la gestión y construcción de los programas de migración (PY) son:

**8.1 Archivos de configuración generados por Visual Studio Code**

Al usar **Visual Studio Code** como herramienta de desarrollo, este crea o utiliza una serie de archivos y directorios que facilitan la gestión del proyecto. A continuación, se describen los archivos y sus funciones más importantes:

* 1. **.vscode/: Directorio de configuración de VS Code**

El directorio vscode se crea automáticamente cuando se abre un proyecto en **Visual Studio Code** y contiene archivos de configuración relacionados con el entorno del editor y la personalización del proyecto.

* **settings.json**:
  + Es un archivo de configuración donde se almacenan preferencias locales de usuario o del espacio de trabajo, como la configuración del formato del código, el uso de tabulaciones o espacios, los lenguajes de programación soportados, entre otros.
  + Si se está trabajando con un proyecto de Python, por ejemplo, se pueden incluir configuraciones para el linter, el entorno virtual, o el formateador de código.
* **launch.json**:
  + Este archivo es usado para la configuración de depuración. Permite definir las configuraciones necesarias para depurar un proyecto, como las variables de entorno, los argumentos de ejecución o el puerto de conexión para depuración remota.
  + Permite configurar y ejecutar una aplicación directamente desde **VS Code**.

}

* **tasks.json**:
  + Este archivo configura las tareas que se pueden ejecutar desde **VS Code**, como la compilación, pruebas, o cualquier tarea personalizada. Se usa para integrar herramientas de construcción y automatizar procesos dentro del editor.
  + En proyectos con **Node.js**, **Python** o incluso proyectos Java, puedes configurar tareas de **npm**, **pytest**, **maven**, o **gradle** a través de este archivo.
* **extensions.json**:
  + En este archivo se guardan las extensiones recomendadas para el proyecto. **VS Code** permite a los desarrolladores definir qué extensiones son necesarias para el proyecto (como linters, depuradores, herramientas de construcción, etc.), y estos archivos pueden ser compartidos con otros miembros del equipo.

## 8.3 Estructura de proyecto de código fuente

La herramienta de gestión de proyecto utilizada para los programas es visualcode, a como se mencionó en el apartado principal de esta sección, por lo cual la estructura principal es la siguiente:

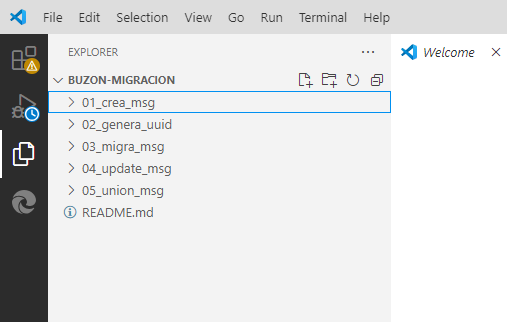
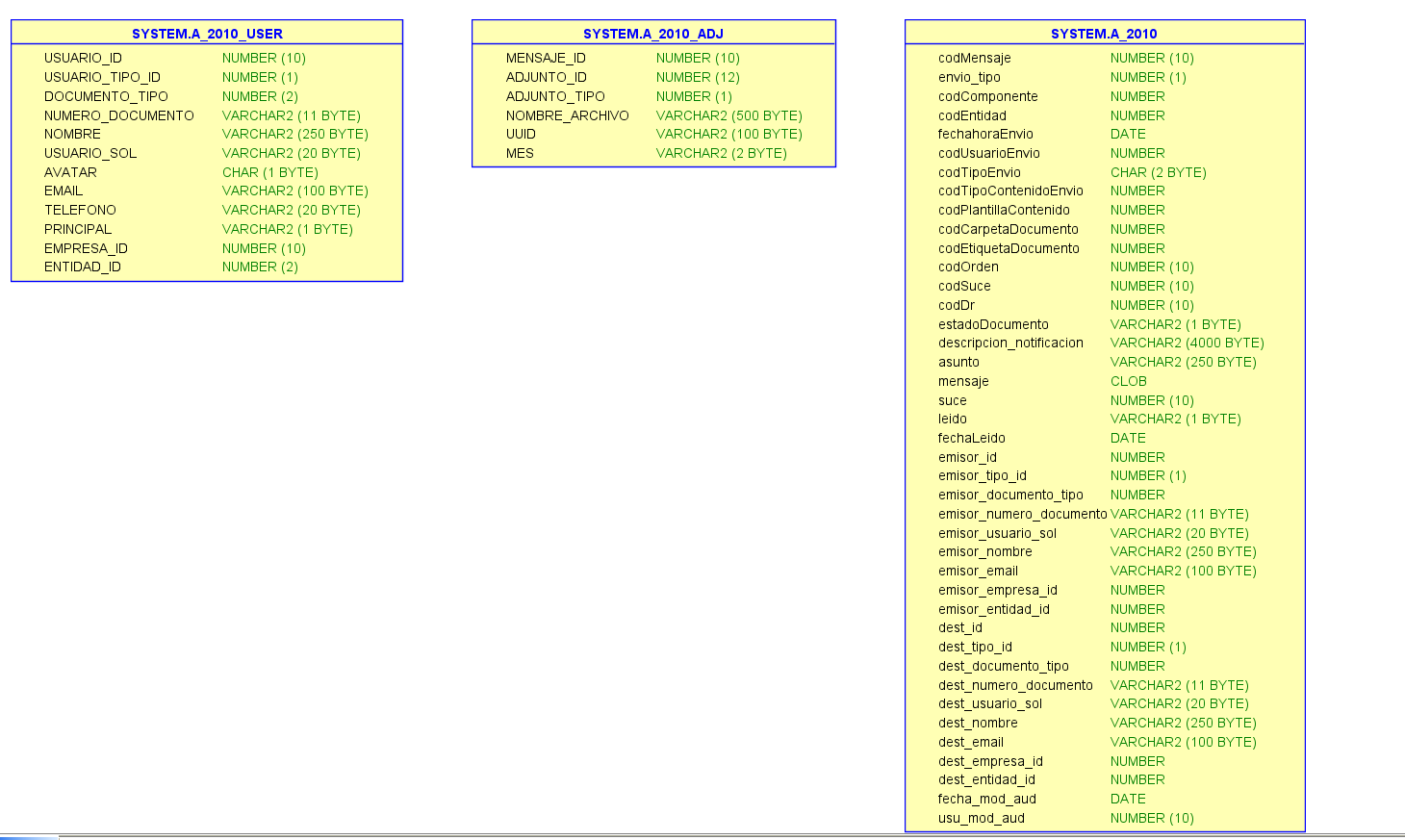


Figura 7 - Estructura de proyecto de código fuente microservicios

# Modelo de Datos

El modelado de datos son las representaciones visuales de los elementos de datos y sus relaciones conformando la base de datos donde se definen las estructuras y los formatos de los datos. El modelo de datos brinda un marco de relaciones entre elementos de datos dentro de una base de datos, así como una guía para el uso de los datos. Para el desarrollo del modelado de la base de datos del sistema se han definido tres tipos de modelo que comienza con el modelo conceptual para establecer de forma general la relación de las entidades, luego pasa al modelo lógico y finalmente al modelo físico siendo un proceso descendente.

Figura 10- Diagrama de bUZON (TS)



## 9.4 Diccionario de Datos

El Diccionario de Datos de BUZOn2 (MONGODB) es el conjunto de definiciones que contiene las características lógicas y puntuales de los datos que se van a utilizar en el sistema incluyendo nombre de los campos, tipo de dato y la descripción del campo; además cada tabla está acompañada por una descripción breve, una tabla de cuatro columnas: #, campo, tipo de dato y descripción.

ver diccionario completo en Anexo D:

# Funcionalidad del Sistema

Los requerimientos funcionales de Buzón 2.0 se soportan en el siguiente documento el cual va a ser las descripciones explícitas del comportamiento del proceso y la información a manejar.

* **VUCE2 - PROD - Documento Funcional\_MIGRACION BUZON 2.0.docx**

# Base de datos

Los programas del sistema se conectan a la base de datos XXX haciendo uso de credenciales que acceden a sus esquemas transaccionales específicos que hasta inclusive a nivel de nombramiento se relacionan. Tenemos que tomar en cuenta que hay microservicios que hacen uso de otras credenciales para acceder al esquema relacionado por la lógica interdependiente de los procesos que soportan.

Tabla 19 - Relación de microservicios - esquema – credencial de acceso

| **PROYECTO** | **NOMBRE** | **ESQUEMA TRANSACCIONAL** | **CREDENCIAL DE ACCESO** | **ESQUEMA RELACIONADO** |
| --- | --- | --- | --- | --- |
| BUZON-MIGRACION | **MIGRACION** | **VCOBJ** | XXXXX | externa security |

# Programas

El diagrama de componente y el diagrama de despliegue van a proporcionar una vista de diseño estática a nivel lógico y físico de Buzon 2.0 documentando la organización de los componentes, sus relaciones y dependencias mutuas.

### 12.1.1 Programas de Alto Nivel de todo el sistema

En el siguiente diagrama se visualiza a alto nivel los componentes de la arquitectura y el lenguaje de programación utilizado en cada módulo.

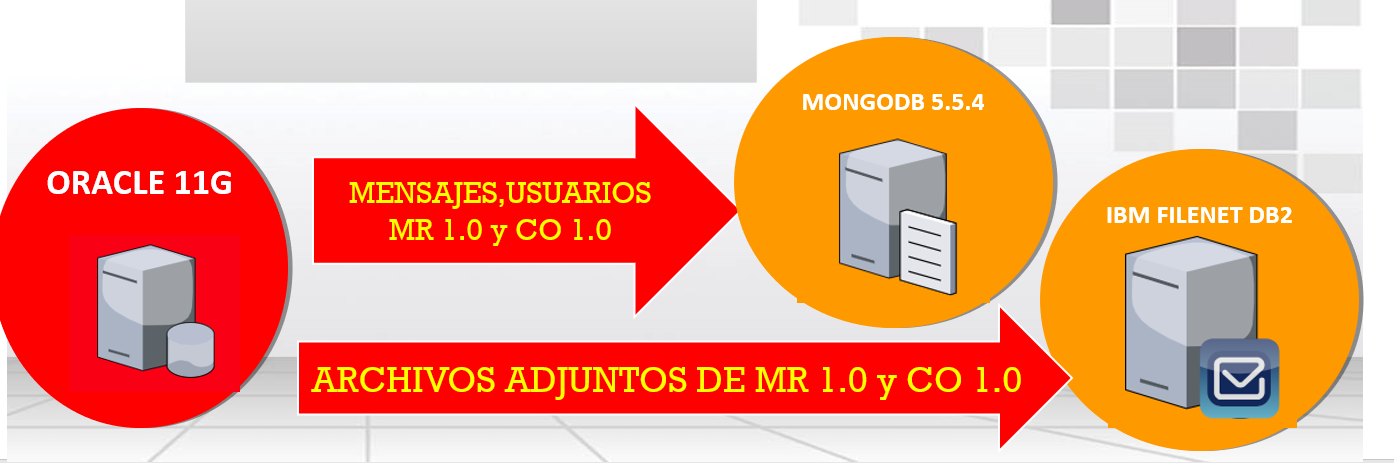


Figura 11- Diagrama de componentes de Alto Nivel de todo el Sistema

### 12.1.2 Diagrama de programas del PROCESO ANUAL y MENSUAL

El siguiente diagrama describe los componentes en cada módulo de los procesos y sus relaciones con otros componentes, el mismo forma parte del Diagrama de Arquitectura indicado en el documento VUCE2-DocumentoArquitectura-Buzon2.0 v1.0.

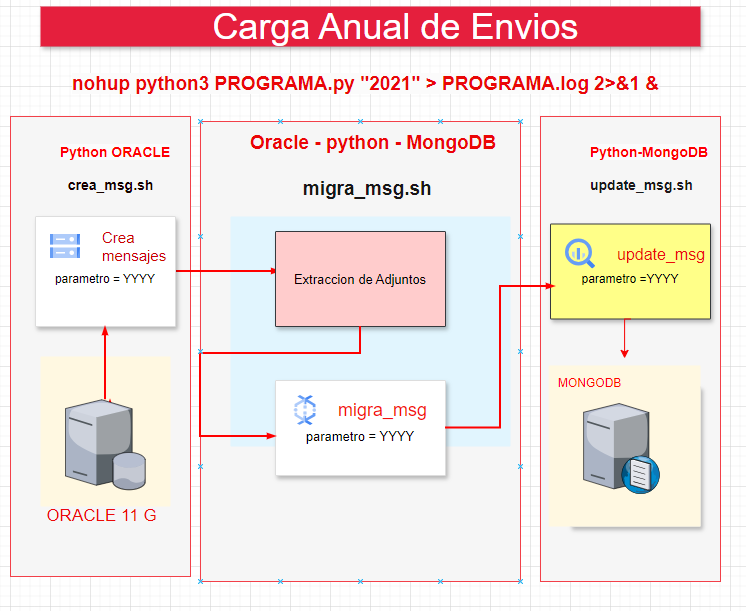


Figura 12- Diagrama de componentes del proceso de carga Anual

Figura 13- Diagrama de Ejecución

# Procesos de MIGRACION -REPROCESO

# Creación de temporales para migrar

**1. Inicialización y Configuración:**

* El script comienza configurando el entorno de Oracle en Python usando oracledb.init\_oracle\_client. Esto le indica a Python dónde encontrar las bibliotecas de Oracle que permiten la conexión con la base de datos. El parámetro lib\_dir especifica la ubicación de la biblioteca cliente de Oracle.

python

Copiar código

oracledb.init\_oracle\_client(lib\_dir='/usr/lib/oracle/11.2/client64/lib')

**2. Obtención del Parámetro de Entrada:**

* Luego, se obtiene el parámetro anomes de los argumentos pasados al script desde la línea de comandos (normalmente, el primer argumento). Este parámetro se espera que tenga un formato de "AÑO-MES" (por ejemplo, 202412 para diciembre de 2024).

python

Copiar código

anomes = sys.argv[1]

print("El parámetro recibido es:", anomes)

**3. Conexión a la Base de Datos Oracle:**

* A continuación, se establece la conexión con la base de datos Oracle utilizando los parámetros de conexión definidos (host, port, service\_name, user, password y el modo SYSDBA para conexiones de administración).

python

Copiar código

conn = oracledb.connect(

host='192.168.8.171',

port=1521,

service\_name='vucepr.vuce.gob.pe',

user='Sys',

password='XXXX',

mode=oracledb.SYSDBA

)

**4. Creación de un Cursor:**

* Se crea un cursor para poder ejecutar consultas o procedimientos almacenados en la base de datos.

python

Copiar código

cursor = conn.cursor()

**5. Definición de Parámetros de Salida:**

* Se definen dos parámetros de salida (cant\_ok y cant\_error) que recibirán los valores después de ejecutar el procedimiento almacenado. Ambos son variables de tipo int en la base de datos Oracle.

python

Copiar código

cant\_ok = cursor.var(int) # Parametro de salida cant\_ok

cant\_error = cursor.var(int) # Parametro de salida cant\_error

**6. Ejecución del Procedimiento Almacenado:**

* Se ejecuta el procedimiento almacenado VCOBJ.CARGAR\_BUZON\_ANUAL, pasándole el parámetro anomes y las variables de salida cant\_ok y cant\_error para capturar los resultados.

python

Copiar código

cursor.callproc("VCOBJ.CARGAR\_BUZON\_ANUAL", [anomes, cant\_ok, cant\_error])

**7. Obtención de los Resultados:**

* Después de ejecutar el procedimiento, se obtienen los valores de los parámetros de salida (cant\_ok y cant\_error) y se imprimen en la consola. Esto te dará la cantidad de registros procesados correctamente y la cantidad de errores.

python

Copiar código

valor\_cant\_ok = cant\_ok.getvalue()

valor\_cant\_error = cant\_error.getvalue()

print(f"Registros cargados = {valor\_cant\_ok}")

print(f"Registros Observados = {valor\_cant\_error}")

print(f"Total de registros = {valor\_cant\_ok + valor\_cant\_error}")

**8. Manejo de Errores:**

* Si ocurre algún error durante la ejecución (ya sea una excepción de la base de datos o una excepción general), se captura y se imprime un mensaje de error en la consola.

python

Copiar código

except oracledb.DatabaseError as e:

print(f"Error de base de datos: {e}")

except Exception as e:

print(f"Error general: {e}")

**9. Cierre de la Conexión y Cálculo del Tiempo de Ejecución:**

* Finalmente, el script cierra el cursor y la conexión con la base de datos. También calcula y muestra el tiempo total que tomó ejecutar el proceso.

python

Copiar código

if cursor:

cursor.close()

if conn:

conn.close()

end\_time = time.time()

execution\_time = end\_time - start\_time

end\_message = f"Process ended at {time.asctime()}, execution time: {execution\_time:.2f} seconds"

print(end\_message)

print("Data transferencia completeda.")

# CARGA DE ADJUNTO AL FILENET

**1. Importación de Módulos y Configuración Inicial**

* **Módulos importados:**
  + oracledb: Para conectarse a una base de datos Oracle.
  + requests: Para realizar solicitudes HTTP, utilizado en la integración con FileNet.
  + json: Para manejar datos en formato JSON.
  + base64: Para codificar datos en Base64 (se usa para los archivos binarios).
  + sys, time, subprocess: Utilizados para manejar argumentos, el tiempo de ejecución y ejecutar comandos del sistema, respectivamente.
* **Configuración del cliente Oracle:**
  + Se establece el entorno de cliente de Oracle para permitir la conexión con la base de datos.

python

Copiar código

subprocess.run(['export', 'ORACLE\_HOME=/usr/lib/oracle/11.2/client64', '&&', 'export', 'LD\_LIBRARY\_PATH=$ORACLE\_HOME/lib:$LD\_LIBRARY\_PATH'], shell=True)

oracledb.init\_oracle\_client(lib\_dir='/usr/lib/oracle/11.2/client64/lib')

**2. Parámetro de Entrada y Conexión a FileNet**

* El parámetro anomes es extraído desde los argumentos del sistema. Este parámetro representa el año y el mes (en formato YYYYMM) y se utiliza en las consultas SQL y en las interacciones con FileNet.

python

Copiar código

anomes = sys.argv[1]

* **Variables de conexión a FileNet:**
  + base\_url, USERNAME, PASSWORD, y cookie se utilizan para interactuar con el servicio de FileNet.
  + token\_url es la URL donde se solicita el token de acceso necesario para autenticar las solicitudes.

**3. Función genera\_uuid:**

Esta función es responsable de crear un UUID para un archivo en FileNet. Realiza lo siguiente:

* **Solicita un token de acceso** usando las credenciales configuradas (username\_token, password\_token) para autenticar la solicitud.
* **Construye una solicitud POST** con los detalles del archivo, incluidos los datos binarios (codificados en base64).
* **Sube el archivo a FileNet** mediante una solicitud POST, adjuntando el archivo junto con su metadata.
* **Devuelve el UUID del archivo** generado por FileNet.

Este proceso es fundamental, ya que es lo que permite generar un identificador único para cada archivo cargado en FileNet.

python

Copiar código

def genera\_uuid(filename, blob\_data):

# Solicitud para obtener el token

response = requests.post(token\_url, headers=headers, data=data, auth=HTTPBasicAuth(username\_token, password\_token))

# Validar la respuesta y continuar con la carga del archivo a FileNet

**4. Función main:**

El flujo principal de la aplicación se encuentra dentro de la función main, que realiza los siguientes pasos:

1. **Conexión a la base de datos Oracle**:
   * Se establece la conexión a la base de datos Oracle con las credenciales proporcionadas.
   * Utiliza el parámetro anomes para realizar las consultas relacionadas con los archivos adjuntos en Oracle.

python

Copiar código

conn = oracledb.connect(

host='192.168.8.171', port=1521, service\_name='vucepr.vuce.gob.pe',

user='Sys', password='temporal2028', mode=oracledb.SYSDBA

)

1. **Ejecutar la primera consulta SQL** para obtener los rangos de los registros de adjuntos:
   * La consulta sql\_range\_query obtiene los rangos de registros, dividiendo los archivos en bloques (basados en el batch\_size).
   * Se utiliza ROW\_NUMBER() para asignar números de fila y luego agrupar esos números en bloques (p. ej., grupos de 1000 registros).

python

Copiar código

sql\_range\_query = """

WITH ranked\_data AS (

SELECT

b.adjunto\_id,

ROW\_NUMBER() OVER (ORDER BY b.adjunto\_id) AS row\_num

FROM VCOBJ.A\_{}\_ADJ b WHERE TRIM(b.uuid) IS NULL

)

SELECT

MIN(adjunto\_id) AS range\_start,

MAX(adjunto\_id) AS range\_end

FROM ranked\_data

GROUP BY FLOOR((row\_num - 1) / {})

ORDER BY range\_start

""".format(anomes, batch\_size)

1. **Procesar los rangos**:
   * Para cada rango obtenido, se ejecuta una segunda consulta (sql\_detail\_query) para obtener los detalles de los archivos adjuntos (como el adjunto\_id, adjunto\_tipo, nombre\_archivo, y los datos binarios del archivo).
   * Los datos binarios se leen en bloques de 1MB utilizando el tamaño de bloque chunk\_size y se guardan en la variable data.

python

Copiar código

sql\_detail\_query = """

SELECT a.adjunto\_id, a.adjunto\_tipo,

VCOBJ.nombre\_archivo(a.nombre\_archivo, a.adjunto\_id, a.adjunto\_tipo) AS nombre\_archivo,

a.archivo

FROM VCOBJ.adjunto a, VCOBJ.A\_{}\_ADJ b

WHERE a.adjunto\_id = b.adjunto\_id

AND TRIM(b.uuid) IS NULL

AND a.adjunto\_id >= {} and a.adjunto\_id < {}

""".format(anomes, range\_start, range\_end)

1. **Subir los archivos a FileNet**:
   * Por cada archivo adjunto procesado, se llama a la función genera\_uuid para subir el archivo a FileNet y obtener su UUID.
   * Si se obtiene un UUID válido, se ejecuta una actualización en la base de datos Oracle para asignar ese UUID al registro correspondiente.

python

Copiar código

document\_id = genera\_uuid(nombre\_archivo, data)

1. **Manejo de errores**:
   * Si se produce un error durante el procesamiento de cualquier archivo o la ejecución de las consultas, se captura y muestra un mensaje de error.
2. **Cierre de conexión y cálculo del tiempo de ejecución**:
   * Se cierra la conexión a la base de datos y se calcula el tiempo de ejecución total del proceso.

**5. Ejecución:**

Cuando el script es ejecutado, realiza la conexión a la base de datos Oracle, obtiene los registros de archivos que aún no tienen un UUID asignado, y sube estos archivos a FileNet, generando y actualizando sus UUID en la base de datos. Al final del proceso, se muestra el tiempo que tomó completar el proceso.

# Migración de mensajes a mongo buzón

**Pasos principales:**

1. **Conexión a la base de datos Oracle:**
   * El script se conecta a una base de datos Oracle utilizando la librería oracledb.connect, ejecutando consultas sobre la tabla dinámica VCOBJ.A\_{anomes}.
2. **Conexión a MongoDB:**
   * Después de conectarse a Oracle, el script establece una conexión con MongoDB utilizando la librería pymongo y selecciona la base de datos vuce-buzonelectronico y la colección correspondiente a anomes.
3. **Procesamiento por lotes:**
   * Los datos de Oracle se procesan en lotes de 100,000 registros, y se insertan en la colección de MongoDB correspondiente usando insert\_many.
4. **Consultas en Oracle:**
   * El script ejecuta varias consultas SELECT para obtener los datos de las tablas Oracle como VCOBJ.A\_{anomes}, VCOBJ.A\_{anomes}\_ADJ, y VCOBJ.A\_{anomes}\_USER.
5. **Transformación de los datos:**
   * Los datos obtenidos de Oracle se transforman en documentos adecuados para MongoDB y se insertan en MongoDB en lotes.
6. **Log de tiempos de ejecución:**
   * Se registran los tiempos de inicio y fin del proceso, junto con la duración total de la ejecución.

**Consideraciones clave:**

1. **Manejo de CLOB en Oracle:**
   * El código maneja correctamente los datos CLOB de Oracle (mensaje), leyendo el contenido si no está vacío, lo cual es necesario porque MongoDB no puede almacenar directamente objetos CLOB de Oracle. Si mensaje es grande, asegúrate de que se maneje adecuadamente para evitar problemas de memoria.
2. **Manejo de errores:**
   * El script incluye manejo de excepciones tanto para errores en la base de datos de Oracle (oracledb.DatabaseError) como para otros errores generales (Exception), lo que mejora la robustez del proceso.
3. **Tamaño de los lotes:**
   * Se establece un tamaño de lote de 100,000, lo cual es adecuado para manejar grandes volúmenes de datos. Sin embargo, dependiendo de los recursos del sistema, podrías necesitar ajustar este tamaño. Haz pruebas de rendimiento para determinar el tamaño óptimo.
4. **Inserciones en MongoDB:**
   * Se utiliza insert\_many para insertar los documentos en MongoDB en lotes, lo cual es más eficiente que insertar un documento a la vez.
5. **Cierre de conexiones:**
   * El bloque finally asegura que tanto las conexiones de Oracle como las de MongoDB se cierren después de procesar los datos, incluso si ocurre un error.
6. **Eficiencia de las consultas en Oracle:**
   * El script ejecuta varias consultas SELECT a la base de datos de Oracle. Es importante asegurarse de que las columnas utilizadas en las cláusulas WHERE y ORDER BY tengan índices apropiados para optimizar el rendimiento.

# Generar mensajes migrados para la colección envíos

**Análisis del Script:**

**1. Configuración de la base de datos MongoDB:**

* El script se conecta a una base de datos de MongoDB utilizando MongoClient de la librería pymongo. La conexión se realiza a un clúster replicado de MongoDB.
* La base de datos seleccionada es vuce-buzonelectronico, y las colecciones son seleccionadas dinámicamente según el parámetro anomes (formando nombres de colecciones como A\_{anomes} y BA\_{anomes}).

**2. Pipeline de Agregación:**

* **Objetivo:** El pipeline de agregación tiene múltiples etapas, que incluyen $lookup, $unwind, $project, y $out. El objetivo es obtener y combinar información de varias colecciones (como componente, tipo\_envio, estado, etc.) para crear documentos más completos.
* Cada una de estas etapas realiza un lookup a colecciones adicionales para enriquecer los documentos en la colección original.
* Después de realizar la agregación, los resultados se escriben en una nueva colección mediante la etapa $out.

**3. Manejo de errores y tiempo de ejecución:**

* Se incluye un manejo de excepciones mediante un bloque try-except, lo que ayuda a capturar errores generales durante el proceso.
* El script también registra el tiempo de inicio y finalización del proceso, calculando el tiempo total de ejecución y mostrando un mensaje al final.

**4. Eliminación condicional de campos:**

* Durante la etapa $project, se utiliza la instrucción $cond para condicionalmente eliminar ciertos campos si son None. Esto asegura que los campos vacíos no sean almacenados en la colección final, lo cual es útil para mantener la limpieza de los datos.

**5. Proceso de ejecución:**

* Se ejecuta el pipeline de agregación en la colección A\_{anomes} y se guarda el resultado en la nueva colección BA\_{anomes}.

# Agregar mensajes a la colección envió de mongo

**Descripción del funcionamiento del script:**

1. **Importaciones y Configuración:**
   * Se importan los módulos necesarios (os, time, subprocess, sys y MongoClient de pymongo).
   * Se obtiene el primer argumento de la línea de comandos (anomes) mediante sys.argv[1]. Este argumento probablemente sea un valor de año y mes (por ejemplo, 202401).
   * Se imprime el parámetro recibido (anomes), que se usará para definir el nombre de la colección en MongoDB.
2. **Configuración del Cliente de MongoDB:**
   * Se establece una conexión a un conjunto de réplicas de MongoDB utilizando la cadena de conexión proporcionada. Esta cadena incluye las credenciales de MongoDB y el nombre del conjunto de réplicas (vuceRSProd).
3. **Procesamiento y Transferencia de Datos:**
   * Se definen las colecciones de origen (BA\_<anomes>) y de destino (envio).
   * Se recorre cada documento de la colección de origen (coleccion\_origen), y se inserta cada documento en la colección de destino (coleccion\_destino) utilizando insert\_one.
4. **Manejo de Errores y Tiempo de Ejecución:**
   * Se utiliza un bloque try para capturar cualquier excepción que pueda ocurrir durante el proceso. Si ocurre un error, se captura y se imprime.
   * También se mide el tiempo de ejecución y se imprime al final del proceso.
   * Finalmente, se cierra la conexión de MongoDB, asegurando que la conexión se termine correctamente.
5. **Flujo de Ejecución:**
   * El script empieza imprimiendo un mensaje con la hora de inicio.
   * Al finalizar la transferencia de datos, imprime un mensaje de finalización con el tiempo de ejecución.

# Anexo D: Diccionario de Datos

Para ver el adjunto del diccionario usar el link

<https://vuce-peru.atlassian.net/jira/software/c/projects/BU20/boards/94?selectedIssue=BU20-8>

