

Manual Técnico

Elaborado por: Angyee Marín

Índice

Contenido

Índice	2
1. Información del Sistema	3
¿Cómo funciona?.....	3
Recolección de datos.....	3
Procesamiento de Datos.....	4
Almacenamiento de datos.....	4
Programación de Tareas.....	4
Logs	5
2. Requerimientos del Sistema	5
2.1 Variables de Entorno Requeridas	5
2.2 Variables en Shell.....	5
2.3 Otras Variables de Entorno.....	5
3. Instalación.....	6
4. Comandos.....	6
4.1 Automatizaciones de Compilación y Construcción (Makefile).....	6
4.1.1 Configuración del Sistema	6
4.1.2 Ejecución del Sistema.....	6
4.1.3 Creación de Carpetas Requeridas	7
4.1.4 Recolección de Datos de SCAN	7
4.1.5 Procesamiento de Datos de SCAN.....	7
4.1.6 Recolección y Procesamiento de Datos de SCAN.....	7
4.1.7 Almacenamiento de la Data Obtenida	7
4.1.8 Limpieza de Data y Archivos Temporales.....	8
4.1.9 Actualización de Fuentes	8
4.2 Interfaz de Línea de Comandos del Sistema	8
4.2.1 Recolección de Datos Manual	8
4.2.2 Procesamiento de Datos Manual	9
4.2.3 Almacenamiento de Datos Manual	9
4.2.4 Almacenamiento Único de Data Procesada.....	10
4.2.5 Actualización de Archivos Fuentes del Sistema.....	11
4.2.6 Operaciones en la Base de Datos	11
4.2.7 Generación de Reportes	12
Anexos	15

1. Información del Sistema

ScanBackup es un sistema diseñado para la recolección de datos de tráfico de todas las interfaces de la red a través de la plataforma de Monitoreo (SCAN), y procesamiento de la misma para la generación de distintos reportes para el análisis de tráfico.

¿Cómo funciona?

El sistema se encarga de recolectar la información del tráfico de las interfaces de red en datos de cada 5 minutos de un día registradas en Monitoreo (SCAN), dando un total de 288 muestras por interfaz en red. Esta data es obtenida gracias a los *logs* que el sistema de SCAN genera por cada interfaz que monitorea; esto se visualiza al cambiar la extensión de la URL de las interfaces de “.html” a “.log”. Una vez obtenida la data de cada una de las interfaces, estas son procesadas para obtener los promedios de tráfico de cada una de las 288 muestras recolectadas en el día por cada interfaz. Todo con el objetivo de almacenar en la base de datos toda la información recolectada (las 288 muestras de data cruda) y toda la información procesada (los promedios generados).

Gracias a este proceso, el sistema tiene un orden estricto para actualizarse: recolección, procesamiento y almacenamiento de los datos. Todo con el fin de prevalecer la información de tráfico de las interfaces para generar reportes donde se necesiten muestreos de data mayor a 3 días, que es el máximo de días que el sistema de SCAN guarda información.

Para este sistema, todo esto se logra mediante el correcto orden de ejecución de las rutinas. Las rutinas se encuentran en la carpeta *routines/*.

Recolección de datos

scanbackup/routines/scanner.sh

Este script se encarga de la recolección de los datos de tráfico de SCAN del día anterior de todas las interfaces especificadas en los archivos fuentes del sistema, que se encuentran en el directorio *sources/SCAN*. Todo el tráfico obtenido está en base a intervalos de cinco minutos. La información obtenida se alojará en el directorio *data/SCAN*, dividido en carpetas según la capa del BBIP que corresponda.

Las capas del BBIP disponibles hasta el momento de esta documentación son:

- *BORDE*: Enlaces Internacionales
- *BRAS*: Agregadores
- *CACHING*: Servicio.
- *RAI*: Servicio.
- *IXP*: Servicio.
- *IPBRAS*: IP activas de los agregadores.

Archivos Fuentes del Sistema

Para poder realizar la captura de data, es necesario declarar los archivos fuentes por cada una de las capas disponibles para consultar (*BORDE*, *BRAS*, *CACHING*, *RAI*, *IXP*, *IPBRAS*). Los datos que se especifiquen en cada uno de dichos archivos son los que el sistema se va a encargar de recolectar información.

Estas fuentes deben estar declaradas en un archivo sin extensión, dentro del directorio *sources/SCAN/* y con el nombre de la capa escrito en mayúscula sin espacios o caracteres especiales.

Como ya se dijo, los archivos fuentes deben estar clasificados por las capas del sistema, siguiendo el siguiente formato para declarar las interfaces que se va a buscar información:

```
link-de-acceso1;nombre-de-la-interfaz1;capacidad-de-la-interfaz1;tipo-de-la-interfaz1  
link-de-acceso2;nombre-de-la-interfaz2;capacidad-de-la-interfaz2;tipo-de-la-interfaz2
```

Procesamiento de Datos

scanbackup/routines/daily.py

Este script se encarga de procesar los datos de SCAN recolectados para generar los resúmenes de promedios de tráfico para todas las interfaces en las que se haya obtenido la información cruda previamente. Esta nueva data promediada será almacenada en el directorio *data/SCAN/DAILY_SUMMARY* en archivos separados por la capa a la que corresponda la interfaz.

Almacenamiento de datos

Este módulo se encuentra en la carpeta *updater/*. Se encarga de cargar toda la data recolectada y procesada de las interfaces obtenidas de SCAN en la base de datos del sistema. Realizando la limpieza de toda la data que haya sido almacenada exitosamente.

Nota: De haber algún problema en la carga de la data, el sistema no borrará los archivos con la data.

Por defecto, este módulo solo almacena la data del día anterior al actual, pero cuenta con distintas opciones para especificar tanto la fecha que se desea cargar, como si se desea cargar toda la data existente sin importar la diferencia de días.

Programación de Tareas

Para la correcta ejecución del actualizador del sistema, se debe programar la ejecución automática diaria del actualizador del sistema. Para ello se debe añadir al *crontab* del sistema el siguiente comando:

```
export PWDSCANBACKUP="/home/user/ScanBackup" # Debe reemplazarse por la ruta  
del directorio del sistema  
export USERSCAN="usuario" # Debe reemplazarse por el usuario  
export PASSWORDSCAN="contraseña" # Debe reemplazarse por la contraseña  
00 04 * * * cd $PWDSCANBACKUP && /usr/bin/make run
```

O puede especificar uno a uno los comandos en el siguiente orden necesario:

```
export PWDSCANBACKUP="/home/user/ScanBackup" # Debe reemplazarse por la ruta  
del directorio del sistema  
export USERSCAN="usuario" # Debe reemplazarse por el usuario  
export PASSWORDSCAN="contraseña" # Debe reemplazarse por la contraseña  
00 04 * * * bash /home/user/ScanBackup/scanbackup/routines/scanner.sh  
00 07 * * * /home/user/ScanBackup/.venv/bin/python -m scanbackup.routines.daily  
30 07 * * * /home/user/ScanBackup/.venv/bin/python -m scanbackup.updater data
```

IMPORTANTE: Si se especifica de esta manera, es importante dejar un espacio de tiempo de mínimo 3h entre el recolector de data (*scanner*) y el procesador de data (*daily*).

Logs

El sistema lleva un registro de logs de todas las operaciones que se realizan. Esto se puede encontrar en el directorio *data/logs/*.

Nota para desarrollador: Estos logs cuentan con un formato específico para facilitar su lectura. Se insta a respetar dicho formato.

2. Requerimientos del Sistema

El sistema requiere la instalación de:

- Python 3.10 o superior.
- MongoDB 8.0 o superior.

Solo válido para sistemas Linux.

2.1 Variables de Entorno Requeridas

El sistema requiere un archivo *“.env.production”* o *“.env”* en la carpeta raíz del proyecto con las siguientes variables de entorno:

```
URI_MONGO="mongodb://localhost:27017/ScanBackupDB"
```

Nota: Para la carga de variables de entorno en modo desarrollador se debe tener el archivo *“.env.development”*, así podrá especificar la opción *“--dev”* en los comandos con la opción disponible.

2.2 Variables en Shell

El sistema requiere que se añadan las siguientes variables a nuestro archivo de shell (*.bashrc* o *.zshrc*):

```
export PWDSCANBACKUP="/home/user/ScanBackup" # Debe reemplazarse por la ruta del directorio del sistema
```

```
export USERSCAN="usuario" # Debe reemplazarse por el usuario
```

```
export PASSWORDSCAN="contraseña" # Debe reemplazarse por la contraseña
```

Nota: Esto es importante para el correcto funcionamiento de la captura de data.

2.3 Otras Variables de Entorno

Solo para el funcionamiento del actualizador de las fuentes de enlaces, se debe añadir las siguientes variables de entorno en el archivo *“.env.production”* o *“.env”*:

```
SCAN_USERNAME="username"
```

```
SCAN_PASSWORD="password"
```

```
SCAN_URL_BORDE_HUAWEI="url" # Página principal de los enlaces
```

```
SCAN_URL_BORDE_CISCO="url" # Página principal de los enlaces
```

```
SCAN_URL_BORDE_JUNIPER="url" # Página principal de los enlaces
```

SCAN_URL_BRAS_HUAWEI="url" # Página principal de los enlaces

SCAN_URL_CACHING_HUAWEI="url" # Página principal de los enlaces

SCAN_URL_RAI_HUAWEI="url" # Página principal de los enlaces

SCAN_URL_RAI_ZTE="url" # Página principal de los enlaces

SCAN_URL_IPBRAS="url" # Página principal de los enlaces

SCAN_URL_IXP="url" # Página principal de los enlaces

Nota: Esto puede ser omitido si no se espera realizar la ejecución del mismo. Por defecto, se puede omitir.

3. Instalación

Nota: Se recomienda realizar la instalación del sistema con el comando disponible en el *makefile* [3.1.1 Configuración del Sistema](#).

El proyecto cuenta con un archivo `pyproject.toml` que contiene toda la información necesaria para instalar el sistema. Para instalar se debe crear un entorno virtual y ejecutar el siguiente comando:

```
$ pip install .
```

Si se realiza de esta manera, una vez finalizada la instalación de dependencias, debe instanciar la base de datos a través del siguiente comando:

```
$ python3 -m scanbackup.database start
```

4. Comandos

4.1 Automatizaciones de Compilación y Construcción (Makefile)

El sistema cuenta con un archivo *Makefile* que contiene toda la información necesaria para ejecutar las rutinas de manera automática, y realizar las operaciones necesarias para el correcto funcionamiento del sistema.

4.1.1 Configuración del Sistema

Para instanciar correctamente el sistema, se debe ejecutar el siguiente comando:

```
$ make setup
```

Esto creará las carpetas necesarias para el funcionamiento del sistema, creará la base de datos e instalará las dependencias de Python.

4.1.2 Ejecución del Sistema

Para poder ejecutar el sistema, se debe ejecutar el siguiente comando:

```
$ make run
```

Esto capturará la data del día anterior, procesará los datos y los almacenará en la base de datos del sistema.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ make run-dev
```

IMPORTANTE: Para esto se debe creado el archivo *.env.development* con las variables de entorno requeridas.

4.1.3 Creación de Carpetas Requeridas

Si se necesita solo crear las carpetas requeridas para el funcionamiento del sistema, se debe ejecutar el siguiente comando:

```
$ make setup-files
```

4.1.4 Recolección de Datos de SCAN

Si se necesita solo recolectar los datos de SCAN, se debe ejecutar el siguiente comando:

```
$ make run-scan
```

Más información sobre las rutinas de recolección de datos se encuentra en la sección [Recolección de datos](#)

4.1.5 Procesamiento de Datos de SCAN

Si se necesita solo procesar los datos de SCAN y obtener la data correspondiente a los promedios diarios, se debe ejecutar el siguiente comando:

```
$ make run-daily
```

Más información sobre las rutinas de procesamiento de datos se encuentra en la sección [Procesamiento de Datos](#)

4.1.6 Recolección y Procesamiento de Datos de SCAN

Si se necesita solo recolectar y procesar los datos de SCAN, se debe ejecutar el siguiente comando:

```
$ make run-base
```

4.1.7 Almacenamiento de la Data Obtenida

Si se necesita solo almacenar la data existente temporal en el sistema, se debe ejecutar el siguiente comando:

```
$ make run-updater
```

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ make run-updater-dev
```

IMPORTANTE: Para esto se debe creado el archivo *.env.development* con las variables de entorno requeridas.

4.1.8 Limpieza de Data y Archivos Temporales

Si se necesita limpiar las carpetas donde se almacenan temporalmente la data obtenida de SCAN antes de guardarse en la base de datos, se debe ejecutar el siguiente comando:

```
$ make clean-data
```

Nota: Tome precaución de utilizar este comando antes de que la data se haya almacenado previamente en la base de datos.

4.1.9 Actualización de Fuentes

Si se requiere actualizar los archivos fuentes que utiliza el sistema para recolectar la información de SCAN puede ejecutar el comando:

```
$ make updater-sources
```

IMPORTANTE: Tenga en cuenta que una vez ejecutado el comando, se recomienda la revisión manual de los archivos, ya que el sistema pudiera no encontrar alguna información de las interfaces en SCAN, dejando así valores por defecto para su modificación manual.

4.2 Interfaz de Línea de Comandos del Sistema

Para la ejecución de los comandos descritos a continuación, se recuerda la activación del entorno virtual.

4.2.1 Recolección de Datos Manual

Para realizar la ejecución manual para recolectar los datos de SCAN, se debe ejecutar el siguiente comando:

```
$ bash $(pwd)/scanbackup/routines/scanner.sh
```

Esto iniciará la ejecución de consulta de tráfico de las interfaces específicas en los archivos fuentes del sistema. Para más información, revisar: [Archivos Fuentes del Sistema](#).

Recolección en Día Específico

Si se requiere especificar una fecha específica para recolectar información, se debe ejecutar el siguiente comando:

```
$ bash $(pwd)/scanbackup/routines/scanner.sh <FECHA>
```

Donde <FECHA> es la fecha del día que se desea recolectar en el formato YYYY-MM-DD. Por defecto, si no se especifica, se recolectará los datos del día anterior al actual.

Recolección en Capa Específica

Si se requiere especificar una capa para recolectar información solo de esa capa, se debe ejecutar el siguiente comando:

```
$ bash $(pwd)/scanbackup/routines/scanner.sh <FECHA> <LAYER>
```

Donde <FECHA> es la fecha del día que se desea recolectar en el formato YYYY-MM-DD y <LAYER> es la capa de la que se desea recolectar los datos.

Nota: Las capas disponibles son: BORDE, BRAS, CACHING, RAI, IXP, IPBRAS. Por defecto, si no se especifica, se recolectará los datos de todas las capas del día anterior al actual.

4.2.2 Procesamiento de Datos Manual

Para realizar la ejecución manual para procesar los datos recolectados de SCAN, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.routines.daily
```

Procesamiento de Día Específico

Si se requiere especificar una fecha solo para procesar la data recolectada de SCAN que pertenezca a ese día, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.routines.daily --date <FECHA>
```

Donde <FECHA> es la fecha del día que se desea procesar en el formato YYYY-MM-DD. Por defecto, si no se especifica, se procesará los datos del día anterior al actual.

Procesamiento de Capa Específica

Si se requiere especificar una capa solo para procesar la data recolectada de SCAN que pertenezca a esa capa, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.routines.daily --layer <LAYER>
```

Donde <LAYER> es la capa de la que se desea procesar los datos.

Nota: Las capas disponibles son: *BORDE*, *BRAS*, *CACHING*, *RAI*, *IXP*, *IPBRAS*. Por defecto, si no se especifica, se procesará los datos de todas las capas.

4.2.3 Almacenamiento de Datos Manual

Para almacenar toda la data recolectada y procesada, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.updater data
```

Esto almacenará la data recolectada y procesada de todas las capas del día anterior al actual.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup.updater data --dev
```

IMPORTANTE: Para esto se debe creado el archivo *.env.development* con las variables de entorno requeridas.

Almacenamiento de Día Específico

Para almacenar los datos de SCAN de una fecha específica, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.updater data --date <FECHA>
```

Donde <FECHA> es la fecha del día que se desea almacenar en el formato YYYY-MM-DD.

Nota: La bandera *--dev* es válida para combinación.

Almacenamiento de Capa Específica

Para almacenar los datos de SCAN de una capa específica, se debe ejecutar el siguiente comando según la capa deseada:

```
$ python3 -m scanbackup.updater data --borde
$ python3 -m scanbackup.updater data --bras
$ python3 -m scanbackup.updater data --caching
$ python3 -m scanbackup.updater data --rai
$ python3 -m scanbackup.updater data --ixp
$ python3 -m scanbackup.updater data --ipbras
```

Nota: La bandera `--date` y `--dev` es válida para combinación.

Almacenamiento de Toda la Data Existente

Para subir toda la data recolectada y procesada, sin importar las fechas, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.updater data --force
```

Esto subirá la data recolectada y procesada de todas las capas de todas las fechas que se obtenga.

Nota: La bandera `--dev` es válida para combinación.

4.2.4 Almacenamiento Único de Data Procesada

Para subir solo la data procesada de los resúmenes de promedios de tráfico diarios, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.updater daily
```

Esto subirá la data procesada de los resúmenes diarios de todas las capas del día anterior al actual.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup.updater daily --dev
```

IMPORTANTE: Para esto se debe creado el archivo ``.env.development`` con las variables de entorno requeridas.

Almacenamiento de Día Específico

Para subir solo la data procesada de los resúmenes diarios de una fecha específica, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.updater daily --date <FECHA>
```

Donde `<FECHA>` es la fecha del día que se desea almacenar en el formato `YYYY-MM-DD`.

Nota: La bandera `--dev` es válida para combinación.

Almacenamiento de Toda la Data Existente

Para subir todos los resúmenes diarios, sin importar las fechas, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.updater daily --force
```

Esto subirá los resúmenes diarios de todas las capas de todas las fechas que se obtenga.

Nota: La bandera `--dev` es válida para combinación.

4.2.5 Actualización de Archivos Fuentes del Sistema

Nota: Se deben especificar las variables de entorno opcionales para el correcto funcionamiento de este apartado. Véase: [2.3 Otras Variables de Entorno](#).

Para actualizar de forma automática los archivos fuentes del sistema (también válido para crear los archivos fuentes del sistema por primera vez), se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.updater sources
```

Esto realizará un *web scrapping* en el sistema SCAN para obtener todas las interfaces de las distintas capas del sistema para su consulta.

IMPORTANTE: Este módulo puede verse afectado con un cambio considerable en la estructura de la página de SCAN. Se recomienda ejecutar este comando con precaución. **NO SE AUTOMATICE DIARIAMENTE.** También debe tener en cuenta que, en algunos casos, pudiera faltar la capacidad de las interfaces si no se logra obtener. Se debe escribir la misma manualmente en el archivo.

SIEMPRE REALICE UNA REVISIÓN DE LOS ARCHIVOS FUENTES UNA VEZ FINALIZADA LA AUTOMATIZACIÓN.

Actualización de las Fuentes de una Capa Específica

Para actualizar las fuentes del sistema solo para una capa específica, se debe ejecutar el siguiente comando según la capa deseada:

```
$ python3 -m scanbackup.updater sources --borde  
$ python3 -m scanbackup.updater sources --bras  
$ python3 -m scanbackup.updater sources --caching  
$ python3 -m scanbackup.updater sources --rai  
$ python3 -m scanbackup.updater sources --ipbras
```

4.2.6 Operaciones en la Base de Datos

Construcción de la Base de Datos

Para construir la base de datos, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.database start
```

Esto creará la base de datos con los esquemas e índices correspondientes.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup.database start --dev
```

IMPORTANTE: Para esto se debe creado el archivo `.env.development` con las variables de entorno requeridas.

Destrucción de la Base de Datos

Para destruir la base de datos, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup.database drop
```

Esto eliminará toda la información de la base de datos de manera irreversible.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup.database drop --dev
```

IMPORTANTE: Para esto se debe creado el archivo *.env.development* con las variables de entorno requeridas.

4.2.7 Generación de Reportes

Generación de Reportes Diarios

Para generar el reporte diario, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup diario
```

Esto generará el reporte del día anterior y lo exportará en un archivo *.xlsx* llamado *Data_Diario_BBIP.xlsx*.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup diario --dev
```

IMPORTANTE: Para esto se debe creado el archivo *.env.development* con las variables de entorno requeridas.

Además, se puede especificar la fecha para el que se desea generar el reporte, con el siguiente comando:

```
$ python3 -m scanbackup diario --date <FECHA>
```

Donde *<FECHA>* es la fecha del día que se desea generar el reporte en el formato *YYYY-MM-DD*.

Nota: La bandera *--dev* es válida para combinación.

Generación de Reporte Semanal

Para generar el reporte semanal, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup semanal
```

Esto generará un reporte con el promedio de datos desde el lunes de la semana pasada hasta el domingo de la semana cursando. Este reporte será exportado un archivo *.xlsx* llamado *Data_Semanal_BBIP.xlsx*.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup semanal --dev
```

IMPORTANTE: Para esto se debe creado el archivo *.env.development* con las variables de entorno requeridas.

Además, se puede especificar la fecha para el que se desea generar el reporte, con el siguiente comando:

```
$ python3 -m scanbackup semanal --literal
```

Esto generará un reporte desde de la semana contando los 7 días hacia atrás iniciando desde el día actual.

Nota: La bandera `--dev` es válida para combinación.

Generación de Reporte Quincenal

Para generar el reporte quincenal, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup quincenal
```

Esto generará un reporte con el promedio de datos desde el primer día del mes hasta el día 15 del mes. Este reporte será exportado un archivo `.xlsx` llamado *Data_Quincenal_BBIP.xlsx*.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup quincenal --dev
```

IMPORTANTE: Para esto se debe creado el archivo `.env.development` con las variables de entorno requeridas.

Además, se puede especificar la fecha para el que se desea generar el reporte, con el siguiente comando:

```
$ python3 -m scanbackup quincenal --literal
```

Esto generará un reporte desde de la quincena contando los 15 días hacia atrás iniciando desde el día actual.

Nota: La bandera `--dev` es válida para combinación.

Generación de Reporte Mensual

Para generar el reporte mensual, se debe ejecutar el siguiente comando:

```
$ python3 -m scanbackup mensual
```

Esto generará un reporte con el promedio de datos desde el primer día del mes hasta el día 30 del mes. Este reporte será exportado un archivo `.xlsx` llamado *Data_Mensual_BBIP.xlsx*.

También se puede ejecutar el comando en modo desarrollo, ejecutando el siguiente comando:

```
$ python3 -m scanbackup mensual --dev
```

IMPORTANTE: Para esto se debe creado el archivo `.env.development` con las variables de entorno requeridas.

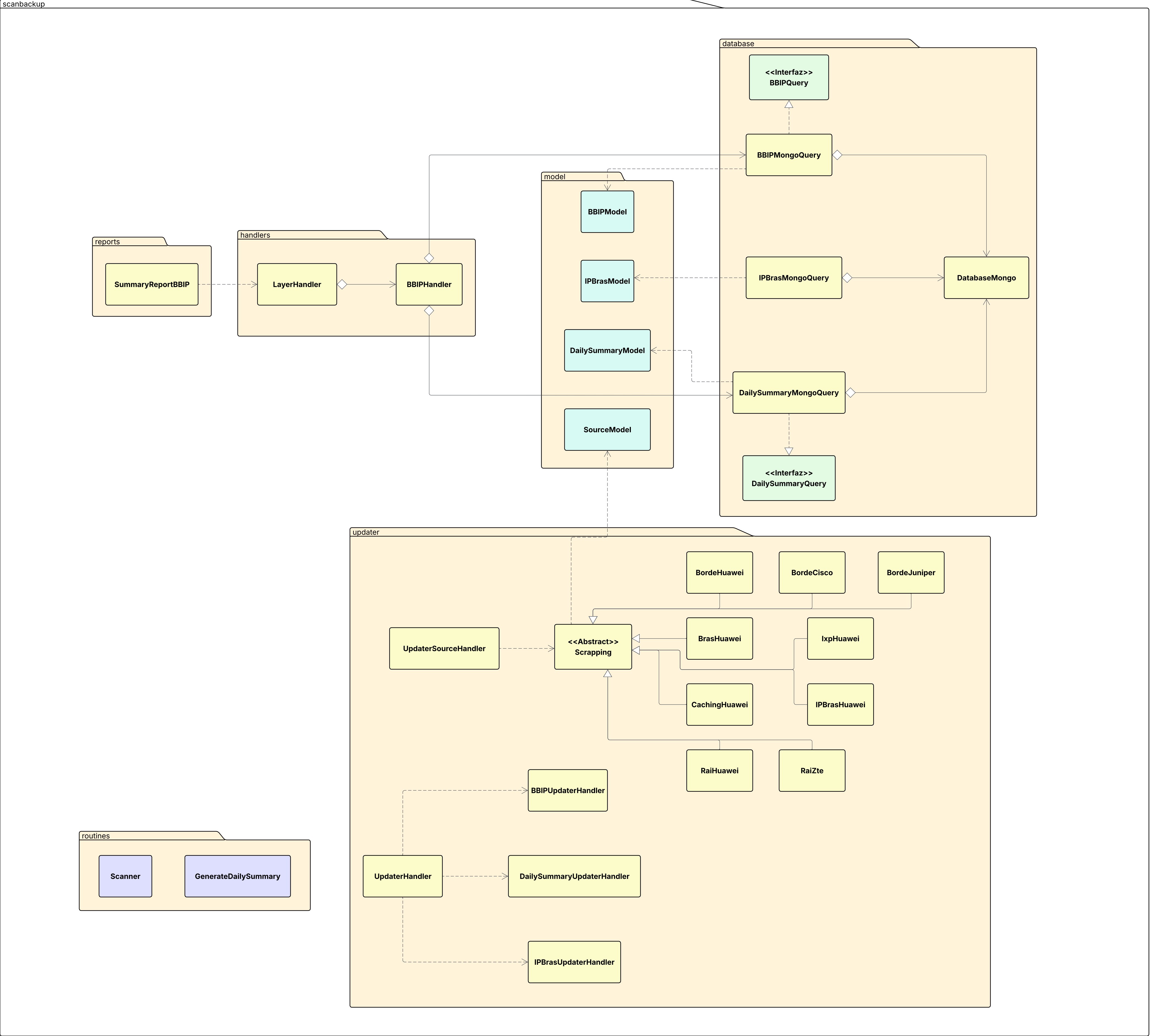
Además, se puede especificar la fecha para el que se desea generar el reporte, con el siguiente comando:

```
$ python3 -m scanbackup mensual --literal
```

Esto generará un reporte desde de la mensual contando los 30 días hacia atrás iniciando desde el día actual.

Nota: La bandera `--dev` es válida para combinación.

Anexos



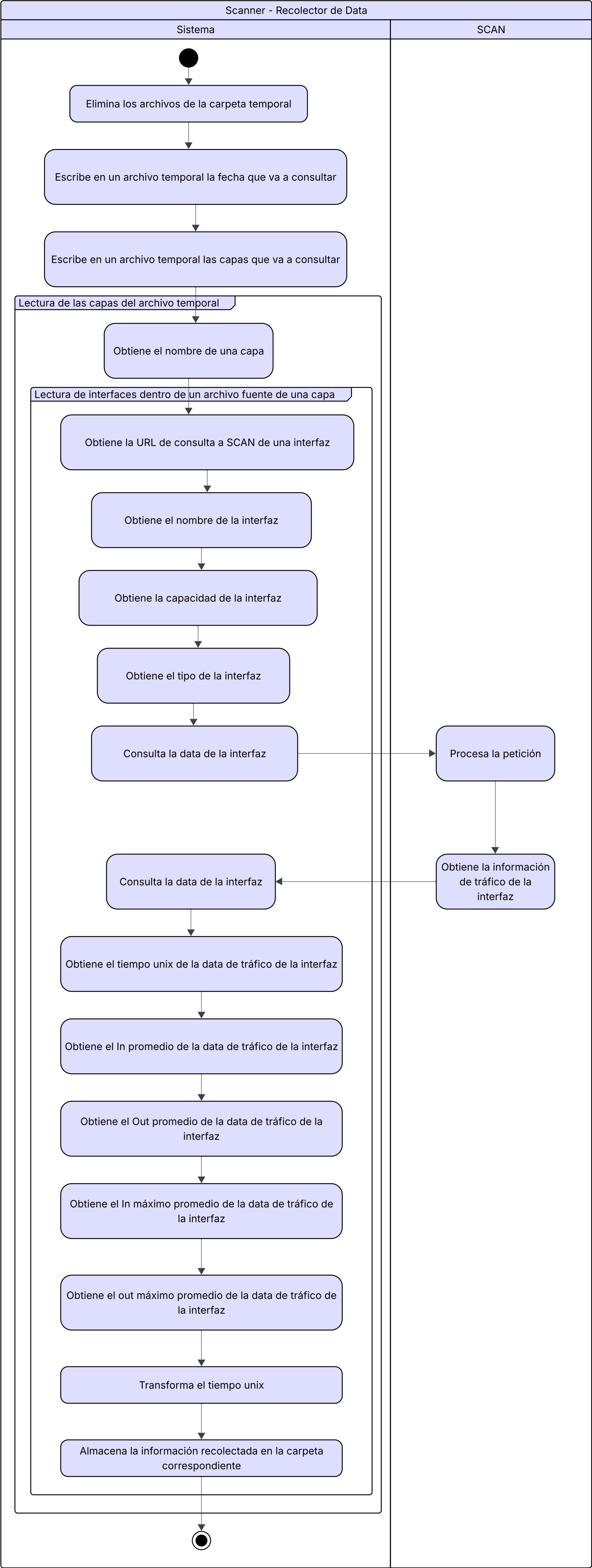
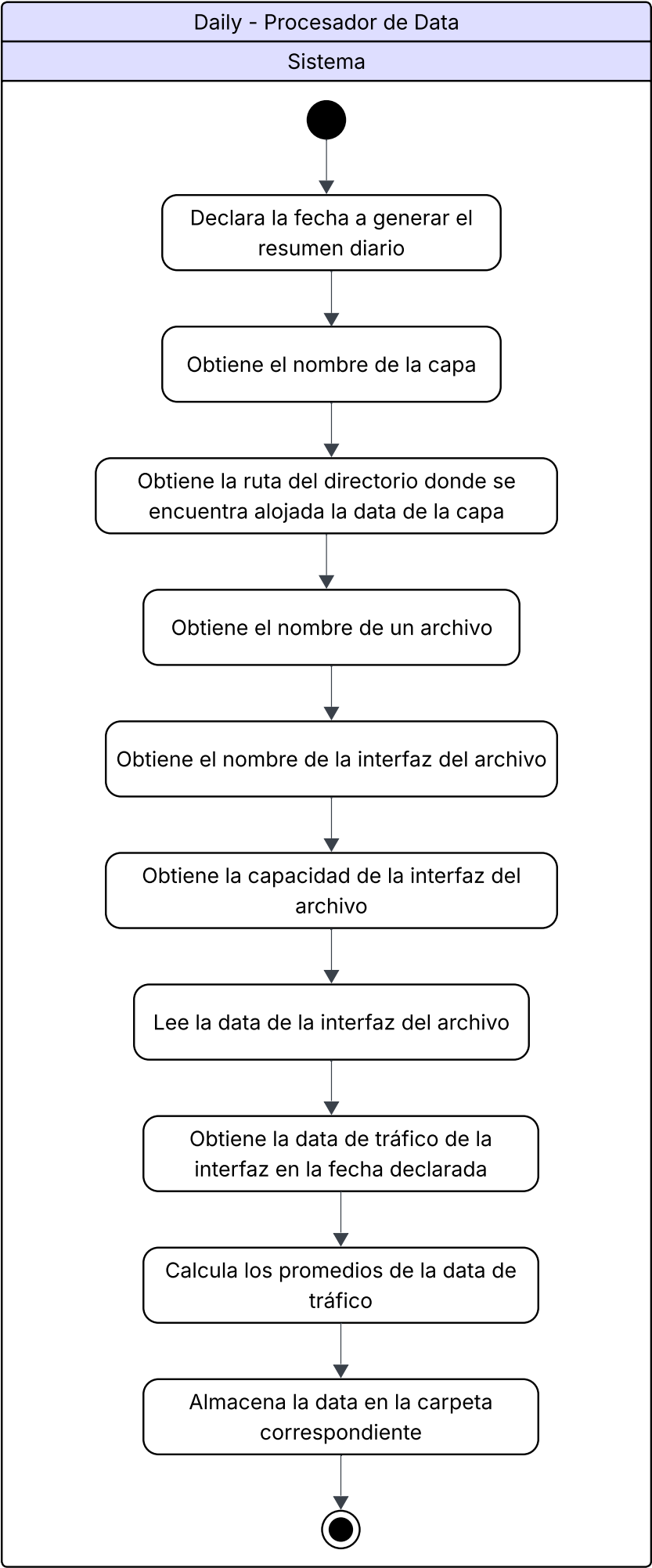


DIAGRAMA DE ACTIVIDADES - PROCESADOR DE DATA (DAILY)



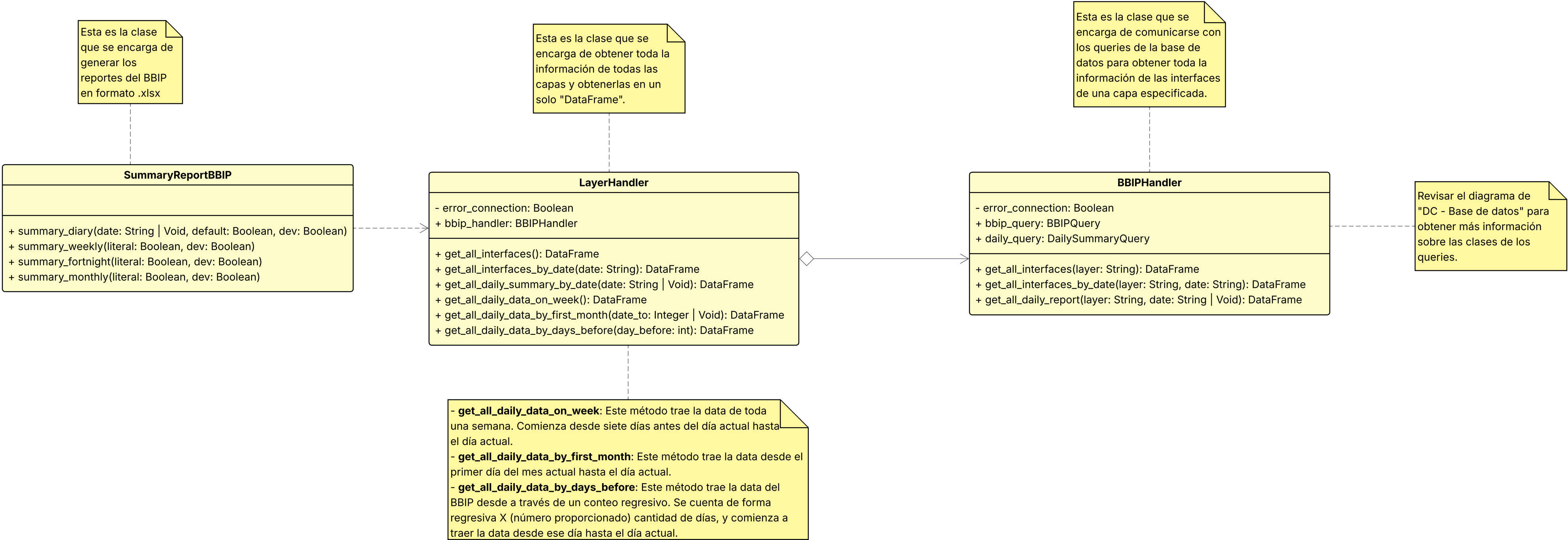
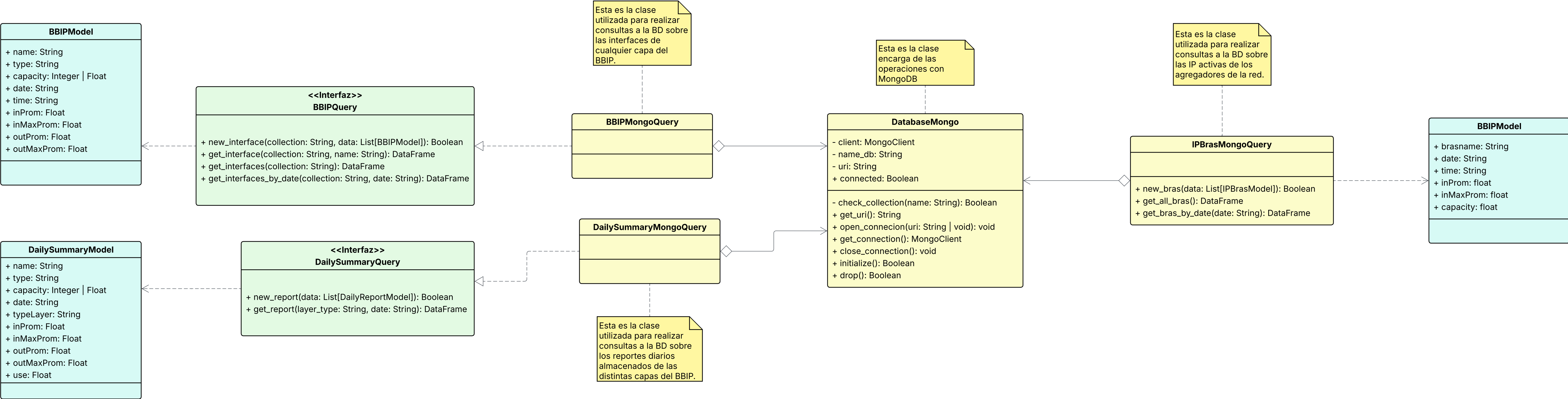
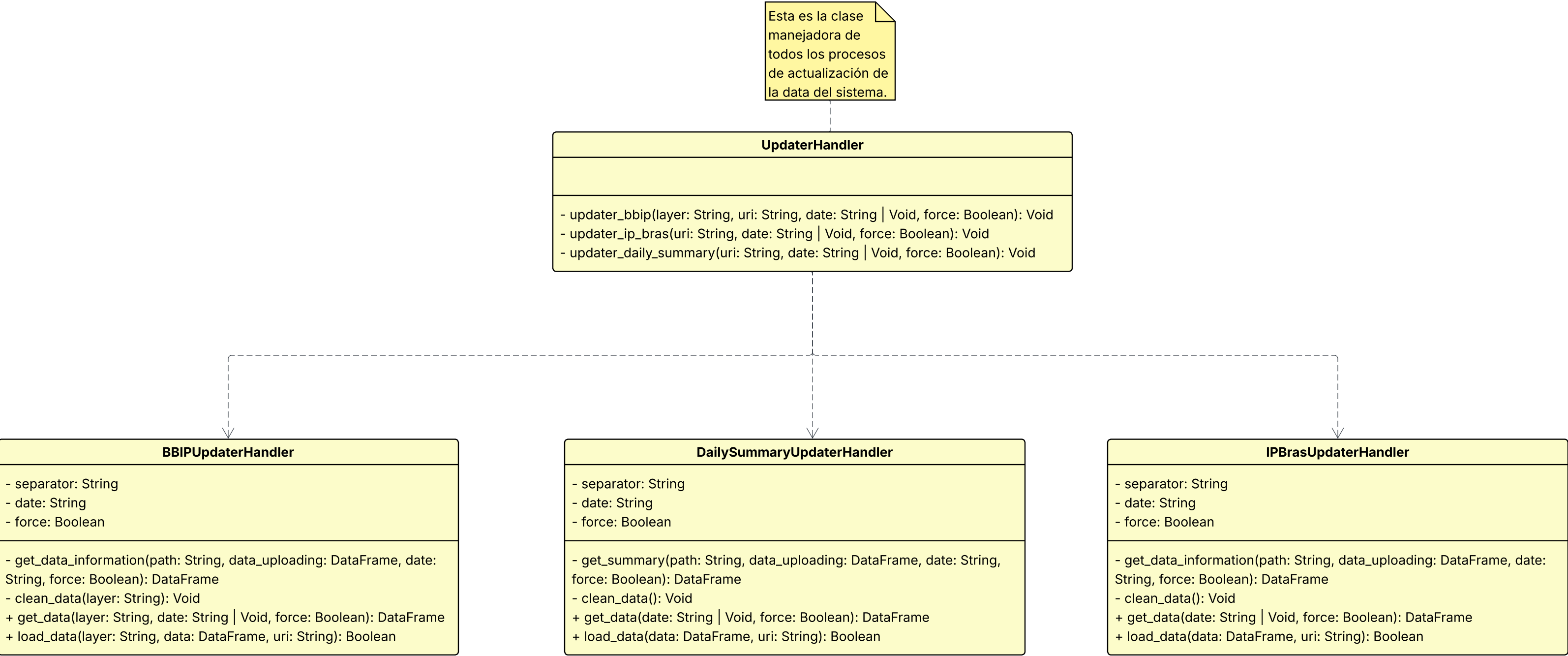
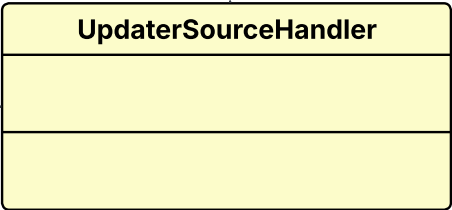
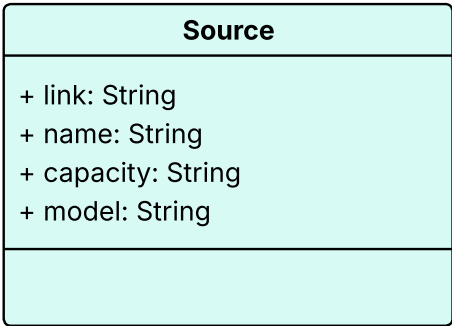


DIAGRAMA DE CLASES - QUERIES Y CONEXIÓN CON LA BASE DE DATOS

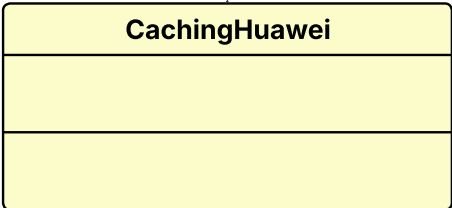
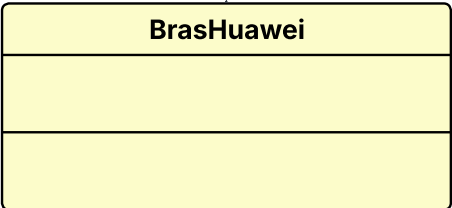
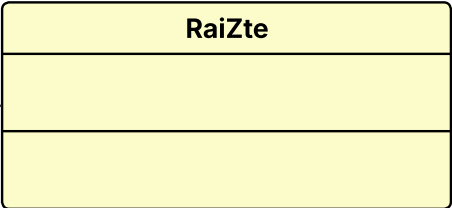
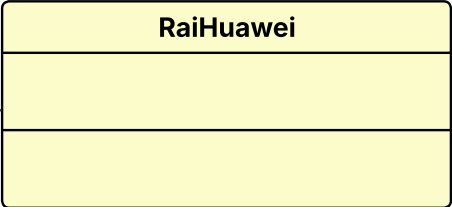
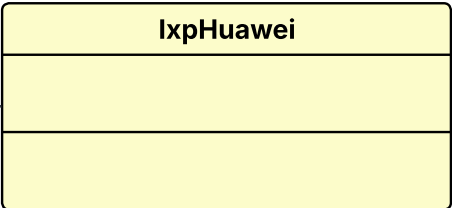
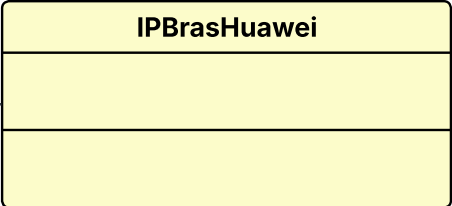
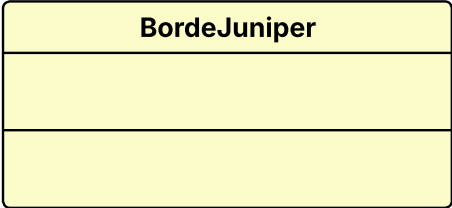
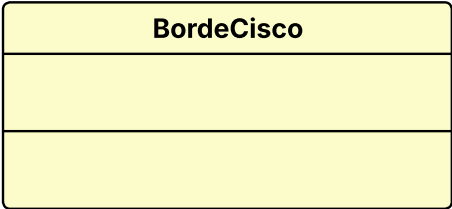
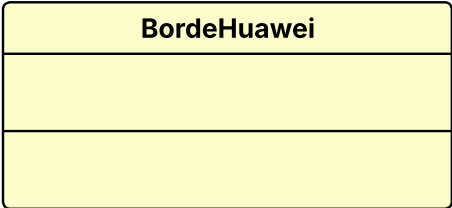
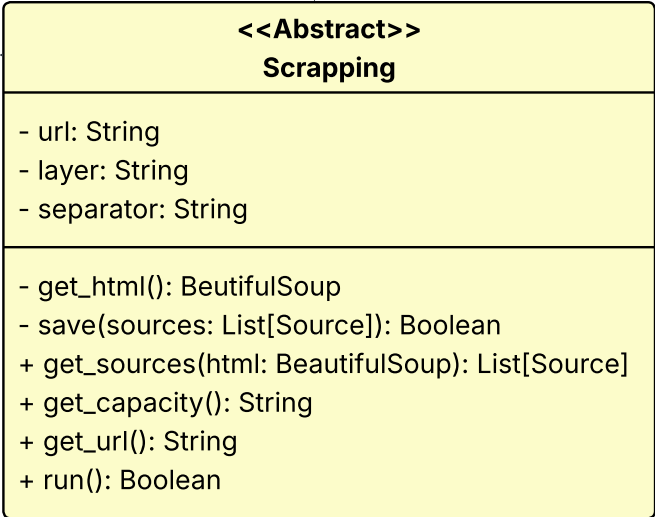




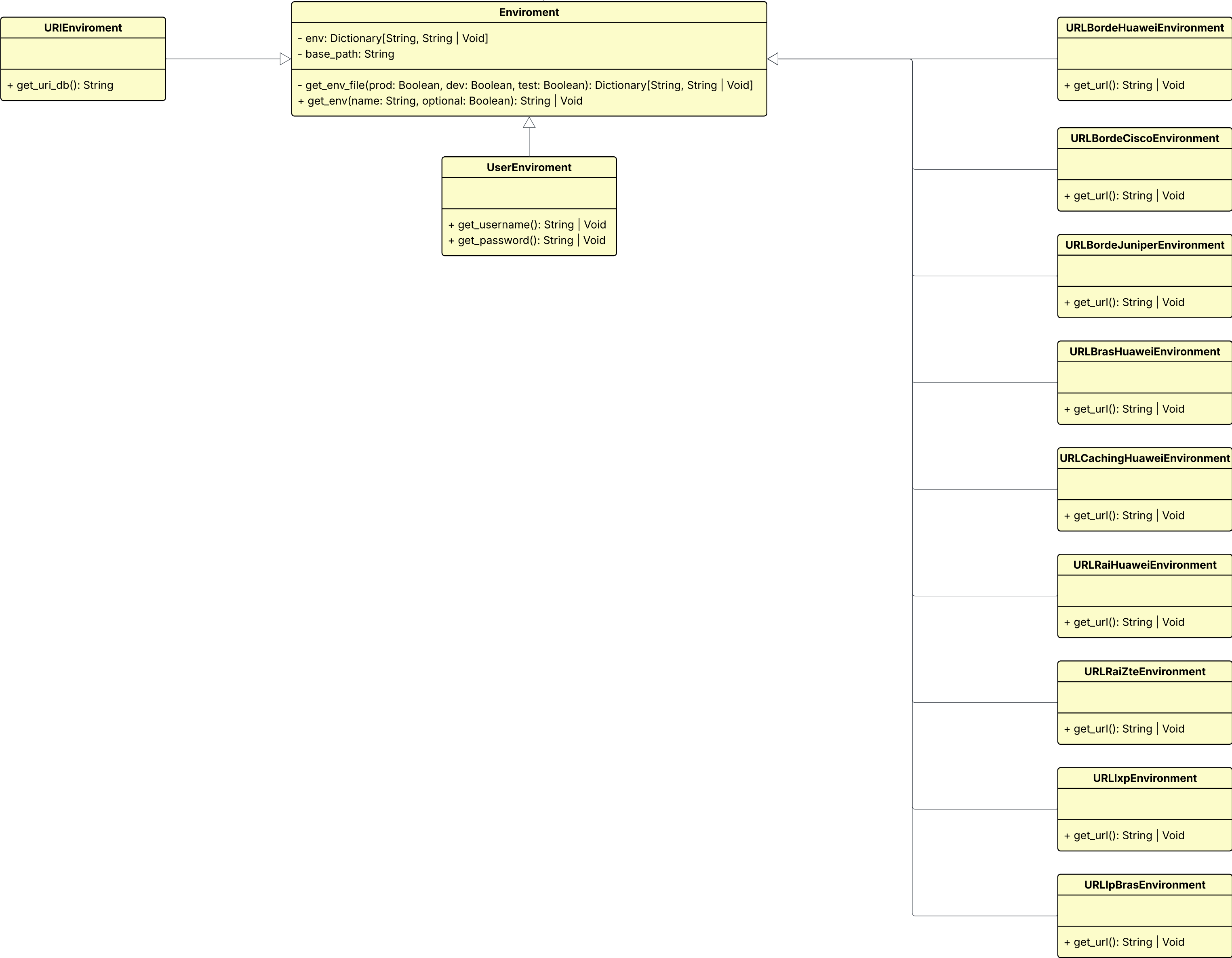
Esta es la clase
manejadora de
todos los procesos
de actualización de
las fuentes del
sistema.



Esta es la clase
que se encarga
de las operaciones
comunes
necesarias para
actualizar las
fuentes del sistema



Esta es la clase base encargada de la recolección de las variables de entorno necesarias para el sistema. No lee ninguna variable de por sí, solo se encarga de cargar la información existente en el archivo. Las clases heredadas son las que encargadas de traer los valores específicos que se han recolectado previamente con esta clase.



ESQUEMAS DE LA BASE DE DATOS

BORDE_HISTORY		
PK	name	string
PK	type	string
PK	date	string
PK	time	string
	capacity	string
	inProm	int, long, double
	inMaxProm	int, long, double
	outProm	int, long, double
	outMaxProm	int, long, double

BRAS_HISTORY		
PK	name	string
PK	type	string
PK	date	string
PK	time	string
	capacity	string
	inProm	int, long, double
	inMaxProm	int, long, double
	outProm	int, long, double
	outMaxProm	int, long, double

CACHING_HISTORY		
PK	name	string
PK	type	string
PK	date	string
PK	time	string
	capacity	string
	inProm	int, long, double
	inMaxProm	int, long, double
	outProm	int, long, double
	outMaxProm	int, long, double

RAI_HISTORY		
PK	name	string
PK	type	string
PK	date	string
PK	time	string
	capacity	string
	inProm	int, long, double
	inMaxProm	int, long, double
	outProm	int, long, double
	outMaxProm	int, long, double

IXP_HISTORY		
PK	name	string
PK	type	string
PK	date	string
PK	time	string
	capacity	string
	inProm	int, long, double
	inMaxProm	int, long, double
	outProm	int, long, double
	outMaxProm	int, long, double

IPBRAS_HISTORY		
PK	brasname	string
PK	type	string
PK	date	string
PK	time	string
	capacity	string
	inProm	int, long, double
	inMaxProm	int, long, double

DAILY_SUMMARY_HISTORY		
PK	name	string
PK	type	string
PK	date	string
PK	typeLayer	string
	capacity	string
	inProm	int, long, double
	inMaxProm	int, long, double
	outProm	int, long, double
	outMaxProm	int, long, double
	use	int, long, double