Esta es la vista general de Merlin

Diagram

Description automatically generated

Se divide en 4 partes:

1. Merlin Agent: Este agente tiene permisos “Anonymous” y accede a los Jobs que tiene asignado mediante un código (passport\_id)
2. Merlin Web: Es el administrador visual de la plataforma. Tiene 3 roles principales:
3. Admin: Este es el rol principal de Hasura y solo lo usa el backend
4. Dqroot: Solo puede ver otros admins. No tiene otros permisos, su uso está destinado a recuperar la contraseña de un administrador
5. Dqadmin: Puede administrar todos los recursos de la plataforma y otorgar permisos a otros usuarios
6. User: Solo puede ser creado por un administrador y solo tiene puede acceder a la información de agentes que un administrador le ha asignado
7. Anonymous: Este acceso es el que suelen usar los agentes para obtener información de los Jobs que tienen que hacer
8. Merlin Hasura: Aquí se encuentra la base de datos y expone un servicio api GraphQL
9. Merlin Server: Se encarga de manejar los procesos que Merlin Hasura no puede manejar como las reglas de negocio. Gestiona los tokens y los refresh tokens

**Merlin Agent**

Schema: merlin\_agent

¿Cómo funciona?

* Cada agente tiene un código **(tabla: AgentaPassport, campo: id)** que usa para identificarse en Hasura.
* En intervalos de 5 minutos pregunta a Hasura si tiene un Job **(tabla: PipelineJobQueue)** pendiente y también informa su estatus **(tabla: AgentPassportPing)**
* Cada Job tiene una definición **(tabla: Pipeline)** que está compuesta por varias unidades de trabajo **(tabla: PipelineUnit)**.
* La cadena de ejecución de un pipelineUnit se define asignando un padre **(campo: pipeline\_unit\_id)**.
* Cuando un pipelineUnit termina de ejecutarse puede pasarle la salida de ese job al siguiente. Por ejemplo, en el diagrama: Un Unit que genera archivos con SQL le da una lista de de las direcciones de los archivos al siguiente Unit que puede ser un compresor ZIP y este a su vez le da la dirección donde guardó el archivo zip al siguiente Unit que puede ser una subida SFTP.
* Al final escribe los logs de cada Unit **(tabla: PipelineJobLog)**

¿Qué necesita?

* La compatibilidad mínima es .netcore 3.1 para sistemas windows 7 SP1 en adelante.
* Podemos usar .netcore 6 para entornos de producción en nuestros servidores.
* El consumo de RAM va a depender de los jobs que vaya a correr, en las pruebas de extracción, compresión y subida apenas llega a los 100MB de RAM.

Instalación

* dotnet merlin-agent.dll. También se puede usar merlin-agent.exe directamente
* Se puede usar con PM2
* Hay que instalar y modificar el código de program.cs para que sea compatible como servicio de Linux

Diagram

Description automatically generated

**Merlin Web**

Schema: public

¿Cómo funciona?

* Se conecta a Merlin Server para recibir su token y refreshtoken. Los cuales se renuevan cada 30 minutos, tiene un mecanismo para hacerlo en segundo plano.
* Usa el token recibido para intercambiar información con Merlin Hasura que es donde se encuentra la base de datos.
* Administrar información de los Agentes y puntos de datos como:
  + La distribuidora (tabla: merlin.**DataNode**)
  + La información de la fábrica (tabla: merlin.**Fabric**)
  + La relación fábrica-distribuidora (tabla: merlin.**FabricXDataNote**)
  + El agente está relacionado con merlin.**FabricXDataNote**
* Puede ver información de los Agentes como:
  + La última vez que se conectó (tabla: merlin\_agent.**AgentPassportPing**, campo: **last\_ping\_at**)
  + Nombre de host
  + Lista de direcciones ip
  + Directorio donde está instalado
  + Versión del sistema operativo
* Administra información de los Pipelines:
  + Tipos de acciones soportadas: Zip, Unzip, Upload/download SFTP, Query (Por ahora solo SQL Server) y Command. Con el tiempo se pueden ir agregando más.
  + Horarios de ejecución de Jobs. Esto se tiene que con los eventos de Hasura directamente utilizando su API: <https://hasura.io/docs/latest/graphql/core/api-reference/metadata-api/scheduled-triggers/#metadata-create-cron-trigger>
  + Para importar desde chainer.exe. Se puede crear un asimilador XML para crear un pipeline directamente en lugar de volverlos a crear uno por uno
  + Las operaciones de DataQuality ahora serán un tipo de especial de Job

¿Qué necesita?

* Node.js 16+
* Vue 3
* TypeScript
* Vite

Instalación

* Esta web estará instalada en el servidor de despliegues.

Diagram

Description automatically generated

**Merlin Server**

Componentes

* [docker-compose] Postgres (Hasura)
* [docker-compose] Graphql-engine (Hasura)
* [docker-compose] Redis (Refresh tokens)
* servidor python (server.py). Funciona con virtual env (.venv)
* servidor de notificaciones (notificator.py). Funciona con virtual env (.venv)

Requerimientos

* Python 3.8+

Instalación

1. Clonar el repo
2. python -m venv .venv
3. source .venv/scripts/activate
4. En el archive pip\_install.txt se encuentran los comandos para instalar los paquetes necesarios para funcionar
5. docker-compose up -d
6. python server.py
7. python notificator.py

¿Qué funciones tiene?

* Aquí se crean los tokens de autenticación (JWT)
* Manejo de refreshtokens usando redis
* Si un agente tiene la opción de observar el ping. El servidor notificará que un agente está desconectado
* Nos informará sobre Jobs que hayan salido mal o cuando se haya terminado de ejecutar un Job

Diagram

Description automatically generated