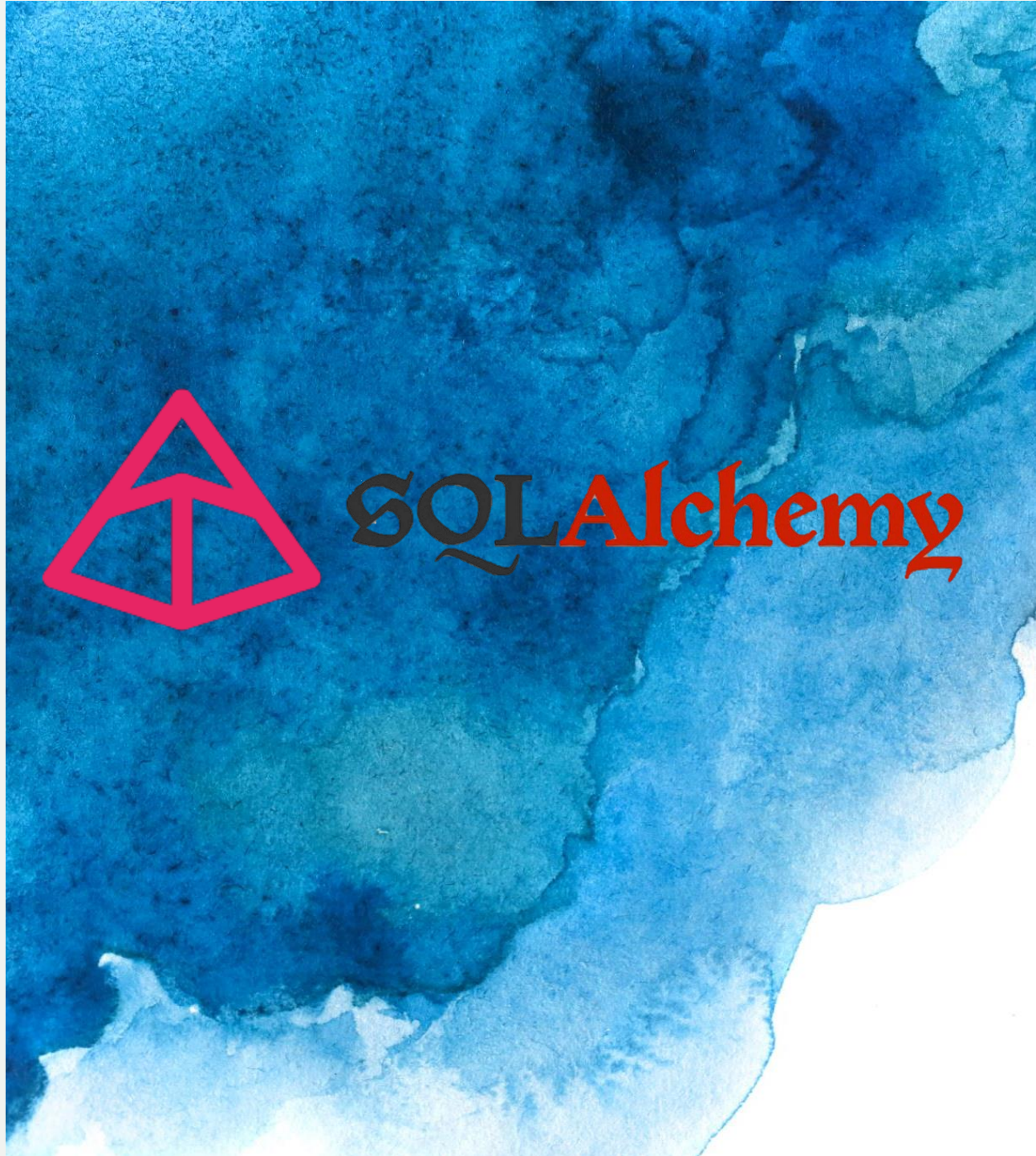
Dominando la validación de datos y el ORM en Python con Pydantic y SQLAlchemy

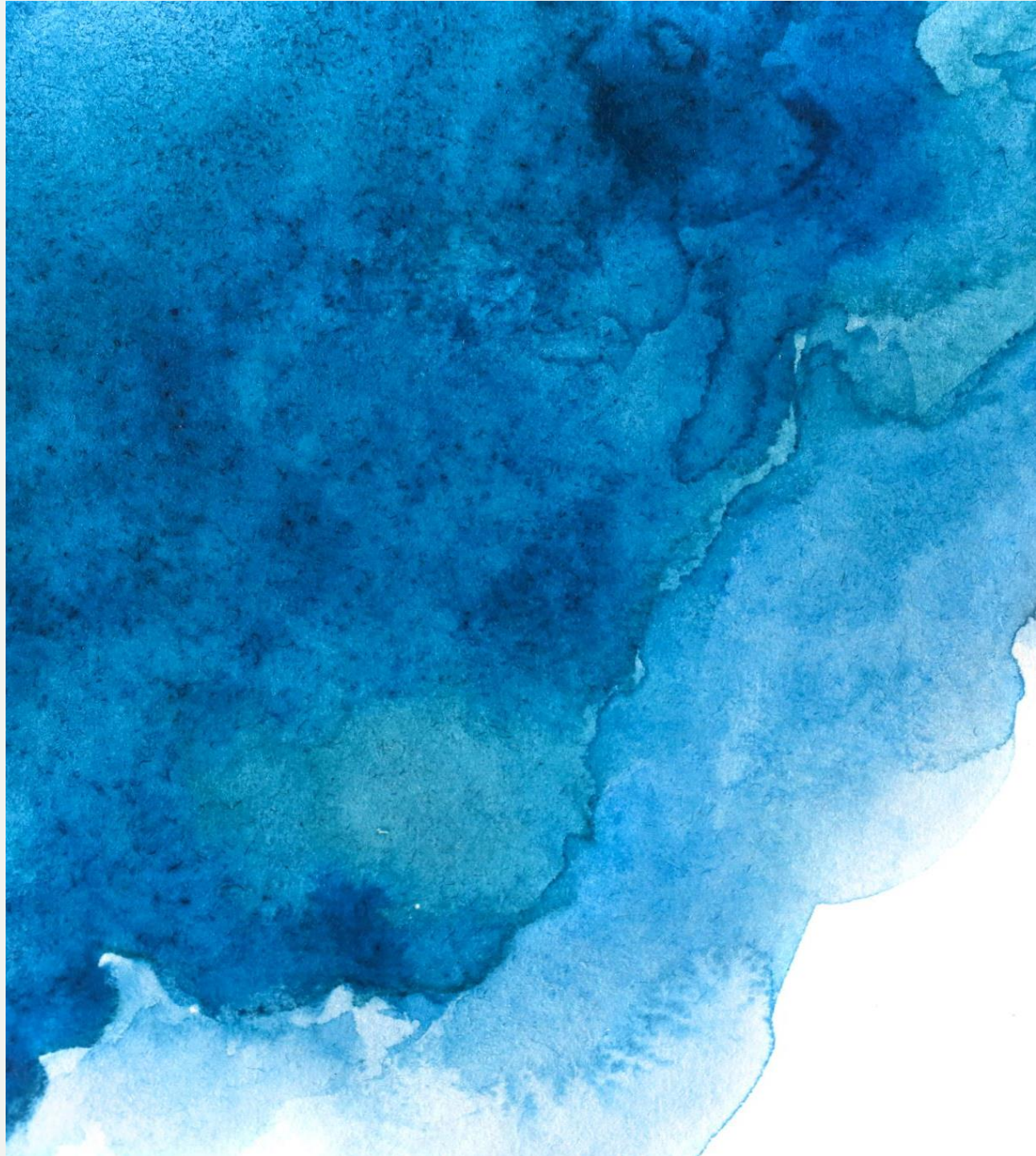
Una introducción a herramientas esenciales para el desarrollo en Python

Juan Pablo Cadena Aguilar



Agenda

- Introducción a Pydantic y SQLAlchemy.
- Uso de Pydantic para validación de datos.
- Uso de SQLAlchemy – ORM, Core y SQL directo.
- Caso de uso: Backend con FastAPI.
- Caso: Ingeniería de datos.

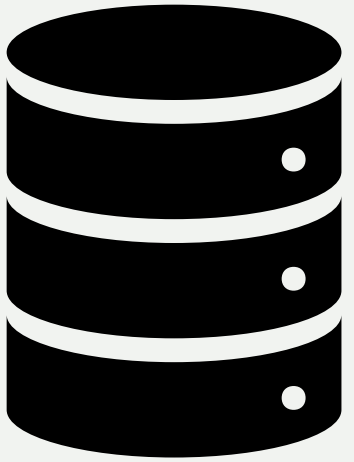


¿Qué es Pydantic?

Pydantic es una biblioteca de validación de datos que utiliza anotaciones de tipo de Python modernas. Su principal función es asegurar que los datos entrantes a nuestros programas sean correctos, reduciendo los errores comunes en el manejo de datos no verificados.

¿Qué es SQLAlchemy?

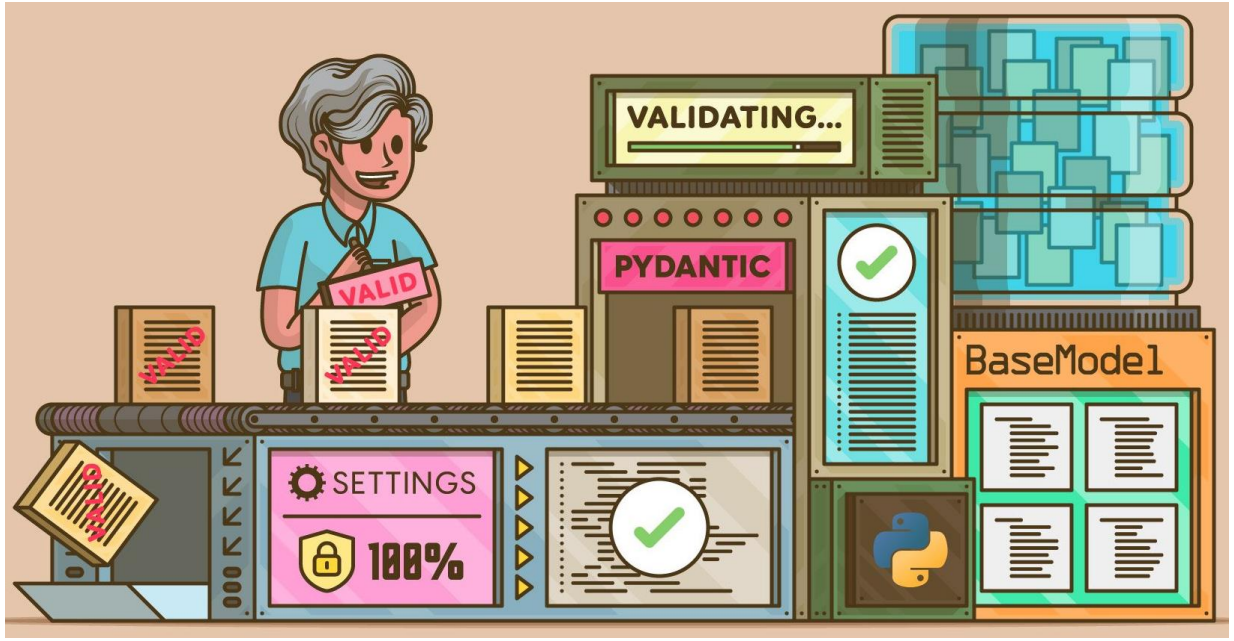
SQLAlchemy es una herramienta poderosa para trabajar con bases de datos en Python. Nos permite comunicarnos con la base de datos de manera intuitiva y segura, mediante el mapeo de objetos a tablas (ORM) o mediante sentencias SQL directas, adecuándose a las necesidades del proyecto.



SQLAlchemy

Pydantic en detalle

Utilizar Pydantic permite una definición clara de modelos de datos con tipos esperados, validaciones automáticas al instanciar objetos, y un sistema robusto de manejo de errores que facilita la depuración y el mantenimiento del código.



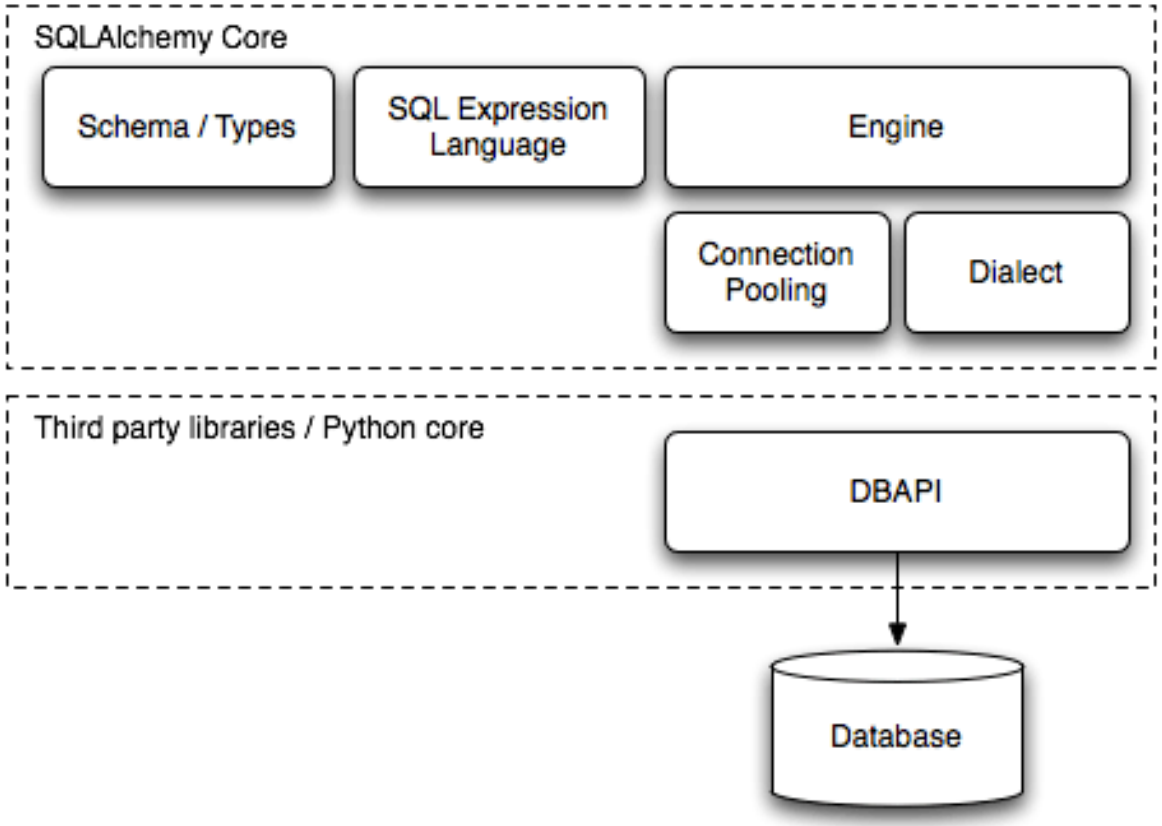
SQLAlchemy - Dialecto SQL

A veces necesitamos el control total que solo el SQL directo puede ofrecer, especialmente para consultas muy específicas o altamente optimizadas. SQLAlchemy nos permite ejecutar este tipo de consultas directamente, combinando la eficiencia del SQL con la seguridad de un toolkit moderno.

```
SELECT * FROM
   (SELECT id, name, max(phone_numbers.number) FROM customers
        INNER JOIN phone_numbers ON customers.id= phone_numbers.cid
        GROUP BY customers.name, customers.id) AS customer_details
   INNER JOIN
   (SELECT cid, count(http_req) AS total_requests FROM web_events
        INNER JOIN http_events ON http_events.reqid= web_events.reqid
        GROUP BY cid) AS customer_events
   ON customer_events.cid= customer_details.id
   WHERE total_requests > 1000;
```

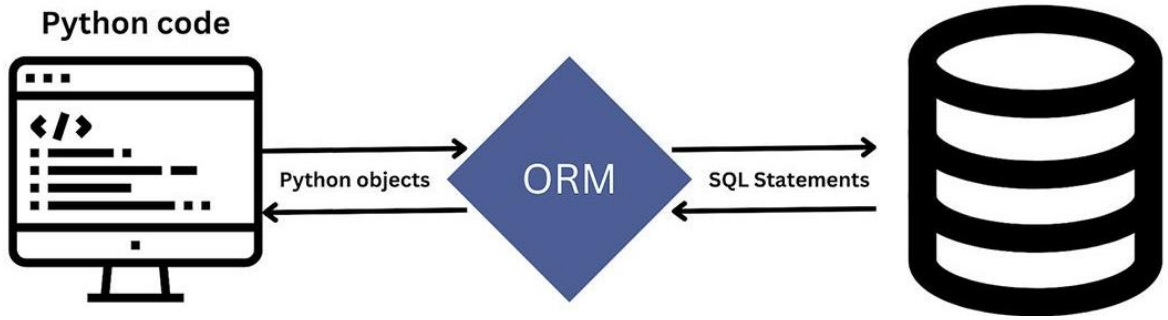
SQLAlchemy - Core

SQLAlchemy Core ofrece un nivel de abstracción que no llega a ser tan alto como el ORM, pero que proporciona una gran flexibilidad para construir consultas SQL de forma segura y eficiente utilizando construcciones de Python.



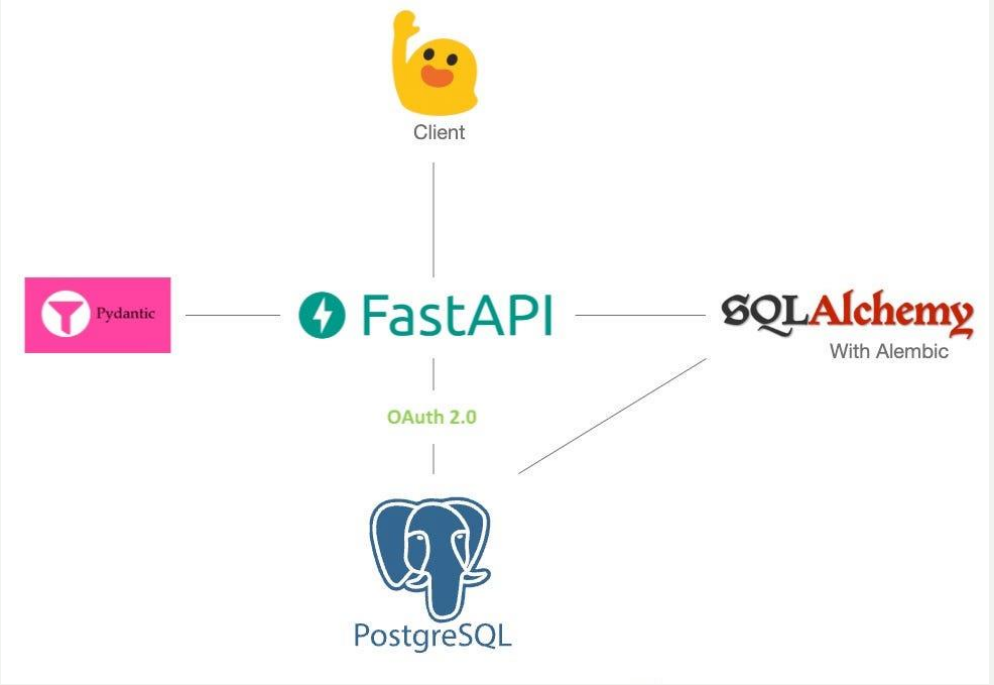
SQLAlchemy - ORM

El ORM de SQLAlchemy transforma las tablas de nuestras bases de datos en clases Python, lo que nos permite interactuar con ellas como si fueran objetos normales. Esto abstrae muchas de las operaciones SQL complejas y mejora la mantenibilidad del código.



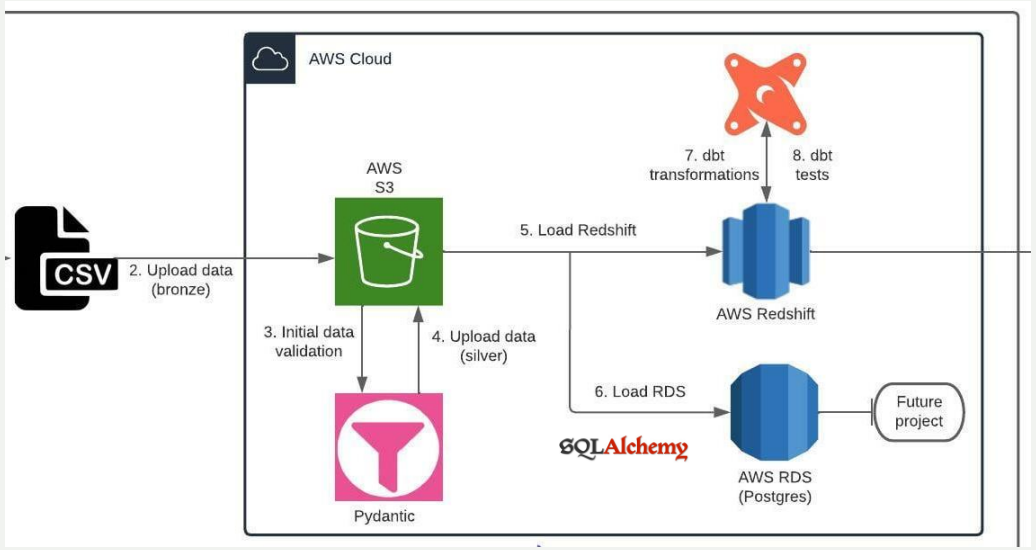
Caso de uso – Backend con FastAPI

Integrar Pydantic y SQLAlchemy en FastAPI es sumamente eficiente para el desarrollo de APIs modernas. Pydantic valida las entradas de la API automáticamente, mientras que SQLAlchemy gestiona las operaciones de la base de datos, haciendo que el desarrollo sea más rápido y menos propenso a errores.



Caso de uso – Ingeniería de datos

En los procesos de ETL, Pydantic y SQLAlchemy juegan roles cruciales. Pydantic asegura que los datos manipulados sean del tipo y formato correcto, mientras que SQLAlchemy facilita la extracción y carga de estos datos en diferentes tipos de bases de datos, optimizando los flujos de trabajo de datos



Recursos adicionales

Documentación:

- https://docs.pydantic.dev/latest/
- https://docs.sqlalchemy.org/en/20/

Repositorio:

- https://github.com/jpcadena/pydantic-sqlalchemy-tutorial

Gracias

!Continúen explorando y experimentando con estas poderosas herramientas en sus proyectos de Python!

https://github.com/jpcadena

https://www.linkedin.com/in/juanpablocadenaaguilar/

Juan Pablo Cadena Aguilar