Data Workbench

Table of contents

1 Objetivos

- Unificar ingesta RAW → bronze → silver con reglas declarativas (YAML) reutilizables.
- Permitir subir insumos (Excel/CSV) desde la web, previsualizar el tidy resultante y correr QA.
- Mantener metadatos en Postgres (datasets, runs, lineage) y exponerlos vía API/UI.
- Usar Redis como cache/cola ligera para previsualizaciones y locks de jobs.
- Integrar validaciones con Great Expectations y exponer reportes accionables.

2 Roadmap

Hito	Resultado	Métricas de éxito
Fase 1 – Fundaciones	Esquema SQL (datasets, runs, lineage), config Redis, specs YAML + loader pydantic, endpoints base `/workbench`.	API responde en <300ms a `GET /workbench/datasets`, seeds sincronizados con specs.
Fase 2 – Preview interactivo	Upload multi-part + parse + sample rows + validaciones GX básicas, UI en Quarto con lista de datasets y tabla de resultados.	Preview < 5s para archivos <5 MB, registros en Postgres + cache Redis.

Hito	Resultado	Métricas de éxito
Fase 3 – Operación guiada	Historias de runs, backfills parametrizables, lineage graficado y publicación automática de Data Docs.	QA obligatoria con alertas si falla, enlaces a Data Docs por run.
Fase 4 – Automatización	Integración con pipelines (Prefect/RQ), scheduling y publicación de gold tables.	KPI: 100% pipelines críticos automatizados en Workbench.

3 TODO inmediato (iteramos con este PR)

- ☒ Añadir dependencia `redis` y `great-expectations` en `requirements/base.txt` con versiones fijas.
- ☒ Crear paquete `data_workbench/` (spec loader, parser, validation, cache helper).
- ☒ Registrar specs iniciales (`dipres_ejecucion`) y exponerlas vía API `/workbench/datasets`.
- ☒ Nueva migración Alembic para tablas `data_workbench_datasets` y `data_workbench_runs` (JSON + timestamps).
- ☒ Endpoints FastAPI (`POST /workbench/preview`, `GET /workbench/runs/{id}`, `GET /workbench/lineage/{slug}`).
- ☒ UI Quarto `frontend/pages/workbench/index.qmd` con formulario upload + vistas de QA y lineage.
- ☒ Documentar flujo en `docs/data-workbench.md` (este archivo) y enlazar en sidebar.

3.1 Checklist en curso (seguir en orden)

1. Mostrar preview RAW + timeline de transformaciones dentro del Workbench.
2. Enriquecer metadatos y lineage (nodos/edges) y exponerlos en la UI.
3. Exponer historial de runs persistente (Redis + Postgres) y permitir cargar runs previos desde la UI.
4. Publicar hub de herramientas (Workbench, API Docs, pgAdmin, GE Data Docs, visores) en `/herramientas/`.
5. Redis como bus intermediario (stream/evento) para difundir runs y coordinar otros servicios.

4 Backlog

- Sincronizar specs YAML tabla `data_workbench_datasets` (upsert automático al levantar la app).
- Persistir lineage expandido (nodos/edges) para grafo interactivo y exponerlo como JSON-LD.
- Integrar `rq worker` + Redis para jobs async (backfills, cargas completas) con locks per dataset.
- Publicar reportes de Great Expectations (Data Docs) estáticos y enlazarlos desde la UI.
- Añadir editor visual de reglas (definir `melt`, `map_categories`, `type_cast`) con previsualización incremental.

- Integrar `dbt` exposures y `openlineage` para construir grafo completo RAW→GOLD.
- Automatizar pruebas específicas (pytest) por spec usando fixtures en `tests/data/workbench/`.
- Security hardening: auth en `/workbench` (Basic/Access), rate-limit uploads y sanitizar metadata.