Proyecto:

Efectos de la desigualdad socioeconómica en la percepción de la democracia en Latinoamérica: periodo 1995-2023

Manual de Instalación

Versión 001

Versión	Causa	Responsable	Fecha
001	Versión Inicial	Patricio Mendoza	23/11/2025

Contents

1 Manual de Instalación y Documentación Técnica **3**

1.1 Arquitectura general de la plataforma 3

1.2 Requisitos previos del servidor dedicado 5

1.3 Obtención del código fuente 6

1.4 Estructura de directorios del proyecto 6

1.5 Puesta en marcha de los servicios 8

1.5.1 Instrucciones de ejecución. 8

1.5.2 Configuración de dashboards y despliegue de `gateway` y `react_app` . . . 9

1.6 Mantenimiento y operación 14

1 Manual de Instalación y Documentación Técnica

1.1 Arquitectura general de la plataforma

La plataforma *Proyecto Democracia YACHAY* se basa en una arquitectura orquestada con Docker Compose, compuesta por varios servicios:

- **`automatic_download`** (descarga y limpieza inicial) Construido desde `./automatization/download_scripts`. Al iniciarse:
  - Monta volúmenes de datos:
    - ENEMDU: `./data/raw/ENEMDU`, `./data/enemdu_persona/{unprocessed,processed}`, `./data/enemdu_vivienda/{unprocessed,processed}`.
    - V-Dem: `./data/raw/VDEM`, `./data/VDEM`.
    - Latinobarómetro: `./data/raw/latinobarometro`, `./data/latinobarometro/normalized_csv`, `./data/latinobarometro/processed`, `./data/latinobarometro/unprocessed_dta`, `./data/latinobarometro/unprocessed_csv`, diccionario `latinobarometro_glossary_candidates_BM.csv`.
  - Utiliza variables de entorno de `.env` (`ENEMDU_ROOT`, `LATINOBAROMETRO_ROOT`, `VDEM_ROOT`, `PERSONA_UNPROC`, `PERSONA_PROCESSED`, etc.).
  - Ejecuta el script `runner.sh`, que encadena la descarga de ENEMDU, Latinobarómetro y V-Dem, así como las rutinas de limpieza inicial (normalización de nombres de columnas, diccionarios, etc.).
- **`clickhouse_server`** (base de datos analítica) Contenedor basado en `clickhouse/clickhouse-server`. Sus responsabilidades principales son:
  - Esperar a que `automatic_download` termine correctamente (`depends_on: service_completed_successfully`).
  - Montar el volumen persistente `clickhouse_volume` en `/var/lib/clickhouse` y los scripts de inicialización en `/docker-entrypoint-initdb.d`.
  - Configurar usuario, contraseña y base de datos a partir de `CH_USER`, `CH_PASSWORD`, `CH_DATABASE`.
  - Exponer los puertos:
    - `8123`: interfaz HTTP.
    - `9000`: protocolo nativo.
  - Ejecutar un `healthcheck` contra `/ping`.
- **`clickhouse_client`** (cliente interactivo) Contenedor `clickhouse/clickhouse-client` que:
  - Espera a que `clickhouse_server` esté `healthy`.
  - Ejecuta un `entrypoint` que comprueba repetidamente la conexión (`SELECT 1`) y, cuando está disponible, abre el cliente de línea de comandos de ClickHouse.
  - Se usa para ejecutar consultas de diagnóstico y mantenimiento manual.

- **automatic_ingest** (ingesta y cálculo de indicadores) Construido desde `./automatization/ingest`. Su función es mover los datos ya limpios a ClickHouse y derivar los indicadores:
  - Depende de:
    - `automatic_download` (completado con éxito).
    - `clickhouse_server` (`service_healthy`).
  - Monta volúmenes para:
    - Diccionarios: `./data/diccionario`.
    - ENEMDU persona/vivienda: directorios `unprocessed` y `processed`.
    - V-Dem: `./data/VDEM`.
    - Latinobarómetro: `normalized_csv` y `processed`.
    - Logs y errores: `./automatization/ingest/logs` y `./automatization/ingest/errors`.
  - Usa variables como `CH_HOST`, `CH_PORT`, `CODIGOS_FILE`, `POVERTY_FILE`, `GEOJSON_FILE`, etc.
  - Ejecuta su `runner.sh` para:
    - Ingerir diccionarios, ENEMDU, Latinobarómetro y V-Dem.
    - Registrar el detalle de ejecución en `LOG_DIR` y errores en `ERR_DIR`.
- **superset** (BI / visualización) Contenedor de Apache Superset, construido desde `./automatization/init-scripts/superset`:
  - Depende de `clickhouse_server` (`service_healthy`).
  - Usa variables de `.env`: `SUPERSET_LOAD_EXAMPLES`, `SUPERSET_SECRET_KEY`, `MAPBOX_API_KEY`, `DATABASE_DIALECT`, `CH_HOST`, `DATABASE_PORT`, `CH_DATABASE`, `CH_USER`, `CH_PASSWORD`, etc.
  - Expone el puerto `8088` para la interfaz web.
  - Monta:
    - `./volumes/superset/home` como `SUPERSET_HOME`.
    - Scripts de inicialización y configuración (`init_superset_db.py`, `superset_config.py`).
  - Ejecuta un `healthcheck` sobre `/health`.
- **gateway** (backend Node/Express para *guest tokens*) Contenedor definido en `./web_app/backend`:
  - Lee variables desde `.env` mediante `env_file: ./.env`, incluyendo `SUPERSET_URL`, `SUPERSET_API`, `SUPERSET_USER`, `SUPERSET_PASS` y `ALLOWED_DASHBOARDS`.
  - Depende de `superset` (`service_healthy`).
  - Expone el puerto `8080`.
  - Expone la ruta `/api/superset/guest-token`, que:
    - Inicia sesión en Superset (`/api/v1/security/login`).
    - Solicita un *guest token* para un `dashboardId` específico.

    * Restringe los IDs de dashboard a una lista blanca (`ALLOWED_DASHBOARDS`) que se actualiza con los UUIDs de los tableros a embeber.

- **`react_app`** (aplicación web de usuario final) Contenedor con el *frontend* React en `./web_app/frontend`:
  - Lee variables desde `.env`: `SUPERSET_API_FRONT`, `ALLOWED_DASHBOARDS`, `GATEWAY`, `MAPBOX_API_KEY`, etc.
  - Expone el puerto `3000`, que sirve la interfaz web con menús, páginas de proyecto, metodología e indicadores.
  - Utiliza el componente `SupersetDashboard` para embeber dashboards, llamando al `gateway` para obtener *guest tokens* y conectando los IDs de Superset con las rutas/selecciones del usuario.

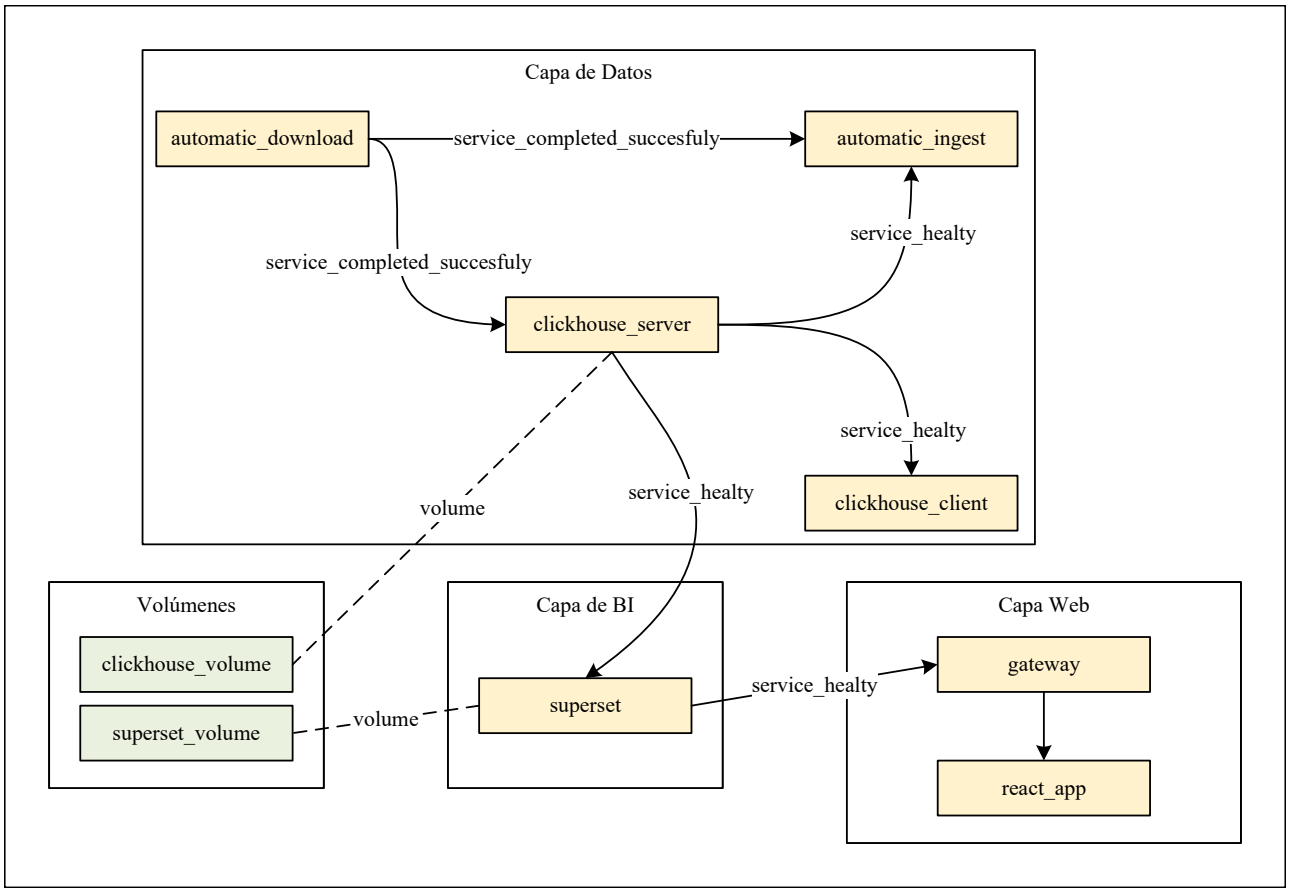

Figure 1: Esquema de los contenedores definidos en `docker-compose.yml` y sus dependencias.

1.2 Requisitos previos del servidor dedicado

- Sistema operativo tipo Linux (Ubuntu Server 20.04+ recomendado).
- Docker $\geq$ 20.10 instalado.
- Docker Compose $\geq$ 1.29 (o plugin equivalente en Docker moderno).
- Acceso a internet desde el servidor para descargar imágenes de Docker y datos de ENEMDU/Latinobarómetro/V-Dem.
- Usuario con permisos para ejecutar comandos `docker` y `docker-compose`.

1.3 Obtención del código fuente

1. Clone el repositorio desde GitHub:

   ```
   git clone https://github.com/datascienceyt/proyecto_pii25-08
   cd proyecto_pii25-08
   ```

2. Verifique la estructura básica del proyecto:

   ```
   tree -L 2
   ```

   Debería observar carpetas como `data/`, `automatization/`, `web_app/`, `docs/`, el archivo `docker-compose.yml`, etc.

3. Otorgar permisos de ejecución a los `runner.sh` que se encargan de ejcutar los códigos de descarga e ingesta.

   ```
   # En el directorio proyecto_pii25-08/ ejecutar:
   chmod +x automatization/download_scripts/runner.sh
   chmod +x automatization/ingest/runner.sh
   ```

1.4 Estructura de directorios del proyecto

El proyecto se organiza en una estructura de carpetas que separa claramente la lógica de automatización, los datos y los archivos de orquestación de contenedores.

```
.env                          # Variables de entorno usadas por docker-compose
docker-compose.yml            # Orquestación de contenedores
README.md                     # Descripción general del proyecto
automatization/               # Código de automatización (ETL)
    download_scripts/         # Descarga y limpieza de datos en Python
        enemdu_descarga.py
        latinobarometro_descarga.py
        vdem_descarga.py
        limpieza_persona.py
        limpieza_vivienda.py
        limpieza_latinobarometro.py
        limpieza_vdem.py
        runner.sh
    ingest/                   # Ingesta a ClickHouse
        ingest_persona.py
        ingest_vivienda.py
        ingest_latinobarometro.py
        ingest_vdem.py
        ingest_codigos.py
        ingest_geojson.py
        errors/               # Filas con errores de ingesta
        logs/                 # Registros de ejecución
```

```
    init-scripts/
        clickhouse/          # Scripts SQL para inicializar ClickHouse
            create_table.sql
        superset/            # Imagen y configuración inicial de Superset
            Dockerfile
            entrypoint.sh
            init_superset_db.py
            superset_config.py
data/                        # Datos utilizados por la plataforma
    diccionario/             # Tablas auxiliares (códigos, salarios, etc.)
    enemdu_persona/          # ENEMDU (personas) procesado / sin procesar
    enemdu_vivienda/         # ENEMDU (vivienda/hogar) procesado / sin procesar
    latinobarometro/         # Latinobarómetro (dta, csv, normalizados)
    raw/                     # Fuentes originales (ENEMDU cruda por año/mes)
```

La carpeta raíz contiene los archivos de configuración generales del proyecto:

- `.env`: archivo de variables de entorno usado por `docker-compose`. Define credenciales, puertos, URLs y parámetros de conexión para los servicios (ClickHouse, Superset, gateway, frontend, etc.).
- `docker-compose.yml`: orquesta los contenedores de base de datos, automatización, BI (Superset) y capa web (gateway y aplicación React).
- `README.md`: proporciona una descripción general del proyecto y las instrucciones básicas de uso.

El directorio `automatization/` agrupa todo el código de automatización (ETL):

- `download_scripts/`: contiene los scripts de descarga y limpieza de las bases ENEMDU, Latinobarómetro y V-Dem. Estos scripts se empaquetan en el contenedor de *descarga automática* y generan archivos CSV normalizados en el directorio `data/`.
- `ingest/`: incluye los scripts de ingesta hacia ClickHouse y cálculo de indicadores a partir de los CSV ya limpios. Los subdirectorios `errors/` y `logs/` almacenan, respectivamente, las filas que no pudieron ser insertadas y los registros de ejecución de los procesos.
- `init-scripts/clickhouse/`: scripts SQL iniciales para crear tablas y vistas en la base de datos.
- `init-scripts/superset/`: archivos necesarios para construir la imagen de Superset y cargar la configuración inicial (usuarios, roles, conexiones a la base de datos, etc.).

El directorio `data/` organiza las fuentes de información y los datos procesados:

- `data/diccionario/`: tablas auxiliares (códigos geográficos, líneas de pobreza, salario básico unificado, geometrías en formato GeoJSON).
- `data/enemdu_persona/` y `data/enemdu_vivienda/`: estructuras con subcarpetas de datos procesados y sin procesar por año y mes, que alimentan los indicadores socioeconómicos.
- `data/latinobarometro/`: contiene los archivos originales en formato `.dta`, las versiones CSV sin normalizar y las versiones normalizadas usadas para la ingesta.
- `data/raw/ENEMDU/`: almacena los ZIP originales descargados del INEC, organizados por año y por ronda (mes), junto con documentación metodológica asociada.

1.5 Puesta en marcha de los servicios

1.5.1 Instrucciones de ejecución.

1. **Configurar el archivo `.env` en la raíz del proyecto**. Los parámetros para las rutas de archivos están pre-configuradas, pero elementos como nombres y contraseñas deben adaptarse para coincidir con las credenciales adecuadas. En el archivo `.env` en la raíz al menos los siguientes parámetros deben ser ajustados:

```
# ClickHouse
CH_USER=admin
CH_PASSWORD=ContrasenaSegura

# Superset
DATABASE_USER=admin                    # Debe coincidir con CH_USER
DATABASE_PASSWORD=ContrasenaSegura     # Debe coincidir con CH_PASSWORD

SUPERSET_USER=pmendoza
SUPERSET_PASS=pmendoza                 # Usar una contraseña segura
SUPERSET_ADMIN_FIRSTNAME=Patricio
SUPERSET_ADMIN_LASTNAME=Mendoza
SUPERSET_ADMIN_EMAIL=pmendoza@yachaytech.edu.ec

# Web App (Nodej + React)
ALLOWED_DASHBOARDS=XXX                 # Los UIDs de los dashboards se obtendrán
                                         más adelante
```

   Estas variables son referenciadas por los servicios `clickhouse_server`, `automatic_ingest` y `superset` en el `docker-compose.yml`.

2. **Primer arranque: capa de datos y BI (sin gateway ni frontend)** Desde la raíz del proyecto, construir e iniciar únicamente los servicios de datos y visualización (usar `sudo` si es necesario):

```
docker-compose --env-file .env up --build \
  automatic_download \
  clickhouse_server \
  clickhouse_client \
  automatic_ingest \
  superset
```

3. **Verificar logs de automatización** Comprobar que los contenedores de descarga y de ingesta terminan sin errores:

   - Descarga/limpieza:

     ```
     docker logs -f automatic_download
     ```

   - Ingesta:

```
docker logs -f automatic_ingest
```

Ambos contenedores deben finalizar con código de salida `0` una vez concluida la ejecución del `runner.sh`.

4. **Validar ClickHouse y Superset**

   (a) Verificar el estado general:

   ```
   docker-compose ps
   ```

   (b) Comprobar la salud de ClickHouse:

   ```
   curl http://localhost:8123/ping
   ```

   (c) Acceder a Superset en el navegador:

   ```
   http://<IP-del-servidor>:8088
   ```

   Iniciar sesión con el usuario administrador configurado en `.env` en los campos `SUPERSET_USER` y `SUPERSET_PASS`.

Figure 2: Inicialización de los contenedores básicos (datos y BI) tras ejecutar el primer `docker-compose --env-file .env up --build`.

1.5.2 Configuración de dashboards y despliegue de `gateway` y `react_app`

Una vez que la capa de datos y Superset están operativos, se deben seguir los pasos siguientes para conectar los dashboards de Superset con la aplicación web:

1. Crear y obtener los IDs de dashboards en Superset.

1. Ingresar a Superset en `http://<IP-del-servidor>:8088` con el usuario `admin`.
2. Crear o importar los dashboards que se van a exponer en la web:
   - Un dashboard de visión general (por ejemplo, “Visión general de indicadores”).
   - Tres dashboards comparativos por base: ENEMDU, Latinobarómetro y V-Dem.

3. Para cada dashboard:

   (a) Abrir el dashboard.

   (b) Hacer clic en el menú de opciones (tres puntos) y seleccionar *Embed dashboard*.

   (c) En el campo *Allowed Domains* se debe colocar las IPs del backend (`http://<IP-del-servidor>:8080`) y del frontend (`http://<IP-del-servidor>:3000`) y hacer clic en el boton *Enable embedding*.

   (d) Copiar el valor del *UUID* (es el identificador que se utilizará en `gateway` y en `react_app`).

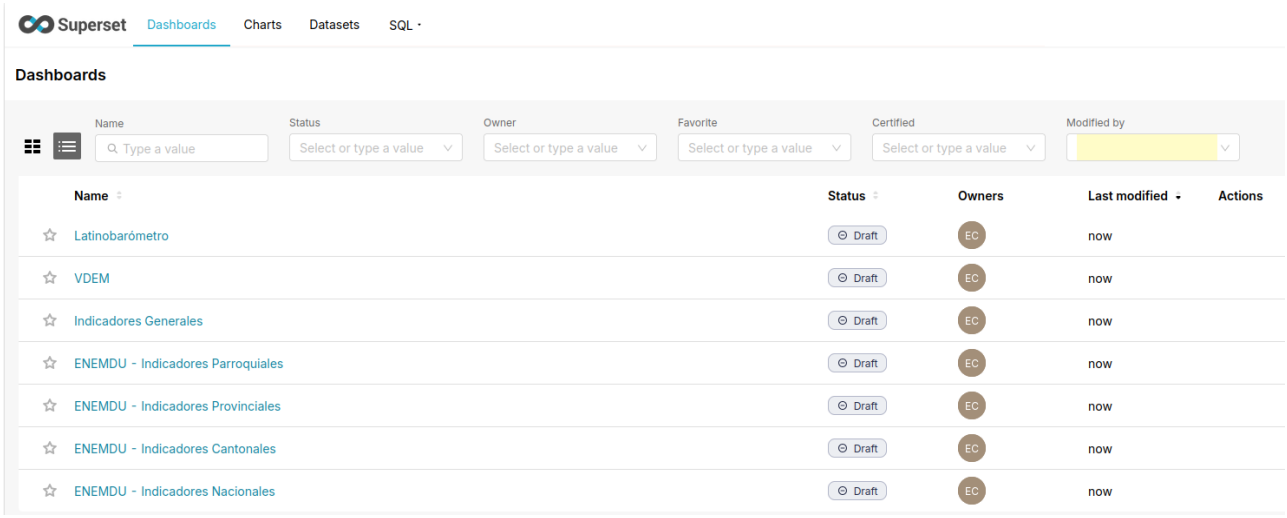

Figure 3: Lista de dashboards creados en Superset para el proyecto (vista general y por base).

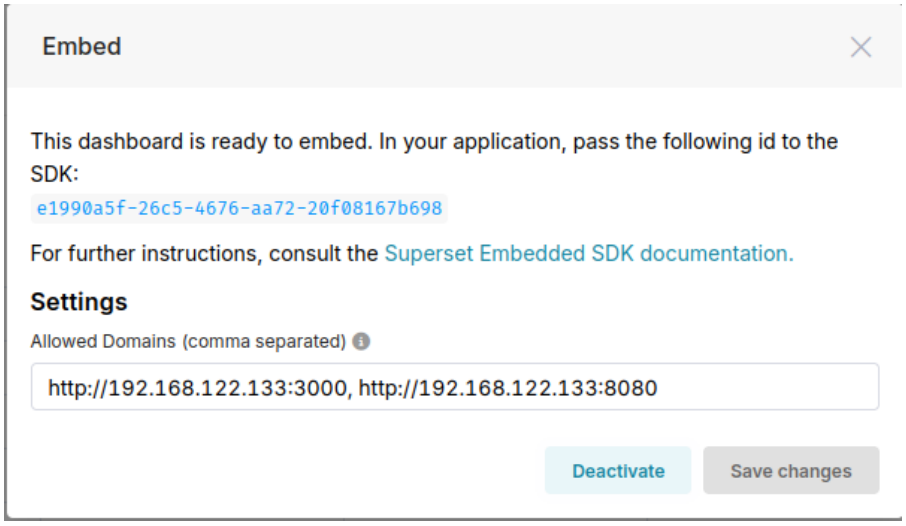

Figure 4: Propiedades avanzadas de un dashboard en Superset mostrando el UUID utilizado para el embed.

2. Registrar los IDs de dashboards en el backend `gateway`. En el backend Node/Express (`web_app/backend/server.js`) se define una lista blanca de dashboards permitidos:

```
const ALLOWED_RAW = process.env.ALLOWED_DASHBOARDS || "";
```

```
const ALLOWED_DASHBOARDS = new Set(
  ALLOWED_RAW.split(",").map((s) => s.trim()).filter(Boolean)
);
```

Antes de compilar la imagen de `gateway`, se debe actualizar el archivo `.env` con la lista de UIDs separados por comas:

```
ALLOWED_DASHBOARDS=UID1,UID2,UID3
```

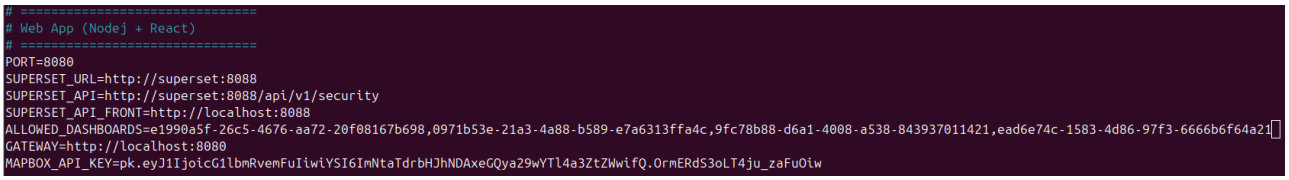

Figure 5: Edición de `.env` para actualizar la lista de dashboards permitidos (UUIDs) en el gateway.

3. Registrar los IDs de dashboards en la aplicación React (`react_app`). La página de indicadores en React (`web_app/frontend/app/src/pages/IndicatorsPage.jsx`) mapea cada "tarjeta" o modo de visualización a un `supersetId`. Dichos `supersetId` deben coincidir con los UUID de Superset:

```
// src/pages/IndicatorsPage.jsx

// 1) Dashboard general (único)
const GENERAL_DASHBOARD = {
  key: "general-overview",
  label: "Visión general de indicadores",
  supersetId: "UUID_GENERAL",
};

// 2) Dashboards comparativos por base
const BASES_DASHBOARDS = [
  {
    key: "base-vdem",
    label: "VDEM",
    description: "Indicadores comparativos construidos a partir de la base VDEM.",
    supersetId: "UUID_VDEM",
  },
  {
    key: "base-latinobarometro",
    label: "Latinobarómetro",
    description:
      "Indicadores comparativos construidos a partir de la base Latinobarómetro.",
    supersetId: "UUID_LATINO",
  },
  {
    key: "base-enemdu",
    label: "ENEMDU",
```

```
    description: "Indicadores comparativos construidos a partir de la base ENEMDU.",
    supersetId: "UUID_ENEMDU",
  },
];
```

1. Abrir `IndicatorsPage.jsx`.
2. Reemplazar `UUID_GENERAL`, `UUID_VDEM`, `UUID_LATINO` y `UUID_ENEMDU` por los IDs concretos de cada dashboard en Superset.
3. Guardar los cambios.

```
  GNU nano 7.2                                                    Indica
// ===== CONFIGURACIÓN DE DASHBOARDS =====

// 1) Dashboard general (único)
const GENERAL_DASHBOARD = {
  key: "general-overview",
  label: "Visión general de indicadores",
  supersetId: "e1990a5f-26c5-4676-aa72-20f08167b698",
};

// 2) Dashboards comparativos por base
const BASES_DASHBOARDS = [
  {
    key: "base-vdem",
    label: "VDEM",
    description: "Indicadores comparativos construidos a partir de la base VDEM.",
    supersetId: "0971b53e-21a3-4a88-b589-e7a6313ffa4c",
  },
  {
    key: "base-latinobarometro",
    label: "Latinobarómetro",
    description:
      "Indicadores comparativos construidos a partir de la base Latinobarómetro.",
    supersetId: "9fc78b88-d6a1-4008-a538-843937011421",
  },
  {
    key: "base-enemdu",
    label: "ENEMDU",
    description: "Indicadores comparativos construidos a partir de la base ENEMDU.",
```

Figure 6: Actualización de los `supersetId` en la página de indicadores de la aplicación React.

3.5. (Opcional) Configuración para entornos de VMs En caso de usar una VM se debe tener en cuenta que el localhost de la maquina virtual es diferente a la del host. En el archivo `src/components/SupersetDashboard.jsx` se hace referencia a localhost para la conexión con el backend y superset:

```
const SUPERSET_DOMAIN = "http://localhost:8088";
const BACKEND_GATEWAY = "http://localhost:8080";
```

Esto se debe cambiar para referenciar la IP de la máquina virtual:

```
const SUPERSET_DOMAIN = "http://<IP-de-la-VM>:8088";
const BACKEND_GATEWAY = "http://<IP-de-la-VM>:8080";
```

Nota: No confundir la IP del servidor con la IP de la VM.

4. Reconstruir y levantar `gateway` y `react_app`. Una vez actualizados `.env` e `IndicatorsPage.jsx`, se procede a construir e iniciar los contenedores correspondientes:

1. Construir e iniciar el backend `gateway`:

   ```
   docker-compose --env-file .env up --build --no-deps gateway
   ```

2. Verificar que el gateway responde:

   ```
   curl http://localhost:8080/health
   ```

3. Construir e iniciar la aplicación React:

   ```
   docker-compose up --build --no-deps react_app
   ```

4. Comprobar el estado global:

   ```
   docker-compose ps
   ```

5. Acceder a la aplicación en el navegador:

   ```
   http://<IP-del-servidor>:3000
   ```

   Deberían visualizarse las páginas de proyecto, metodología e indicadores, y al entrar a la sección de indicadores se embeberán los dashboards definidos en Superset.

Figure 7: Frontend React + Dashboards funcionando.

1.6 Mantenimiento y operación

El repositorio incluye un documento de operaciones y runbooks que describe procedimientos de backup, rotación de logs y troubleshooting.

Backups

- **ClickHouse**:
  - Respaldar el volumen `clickhouse_volume`, que se monta en `/var/lib/clickhouse`.
  - Opcionalmente, utilizar herramientas como `clickhouse-backup` para dumps incrementales.
- **Superset**:
  - Respaldar el volumen `./volumes/superset/home` asociado a `SUPERSET_HOME`.
  - Exportar dashboards y datasets vía CLI:

```
superset export-dashboards -f backup_dashboards.zip
```

Logs

- Los logs de ingesta se almacenan en `automatization/ingest/logs` y los errores en `automatization/ingest/errors`.

Resolución de problemas

Procedimientos básicos (usaro `sudo` si es necesario):

- Ver estado de contenedores:

```
docker-compose ps
```

- Forzar descarga y Limpieza de ENEMDU (Ecuador)/Latinobarómetro/V-Dem:

```
# ENEMDU
docker-compose exec automatic_download python enemdu_descarga.py \
     && python limpieza_persona.py \
     && python limpieza_vivienda.py

# Latinobarometro
docker-compose exec automatic_download python latinobarometro_descarga.py \
     && python limpieza_latinobarometro.py

# V-Dem
docker-compose exec automatic_download python vdem_descarga.py \
     && python limpieza_vdem.py
```

- Reprocesar únicamente descarga:

```
# ENEMDU
docker-compose exec automatic_download python enemdu_descarga.py

# Latinobarómetro
docker-compose exec automatic_download python latinobarometro_descarga.py

# V-Dem
docker-compose exec automatic_download python vdem_descarga.py
```

- Reprocesar únicamente limpieza:

```
# ENEMDU
docker-compose exec automatic_download python limpieza_persona.py
docker-compose exec automatic_download python limpieza_vivienda.py

# Latinobarómetro
docker-compose exec automatic_download python limpieza_latinobarometro.py

# V-Dem
docker-compose exec automatic_download python limpieza_vdem.py
```

- Ingresar al cliente de la base de datos y hacer cambios:

```
# Ingresar al contenedor de clickhouse_client:
docker-compose exec clickhouse_client sh

# Ingresar a la base de datos
# (Usar las credenciales reales definidas en .env):
clickhouse-client --host clickhouse_server \
     --port 9000 --user USER --password CONTRASEÑA
```

- Eliminar base de datos y contenedores:

  ```
  docker-compose down --volumes
  ```

  Nota: Al eliminar los contenedores y sus volúmenes se eliminará la base de datos. Los archivos CSV procesados de ENEMDU (Persona y Vivienda) y Latinobarómetro deben moverse a su respectiva carpeta "unprocessed" para que la ingesta se realice correctamente.

- Validar salud de ClickHouse:

  ```
  curl http://localhost:8123/ping
  ```