Sistema de Monitoreo de Pipelines DevOps

Alumno

Luiggi Rodríguez Aliaga

Curso

SQL (Flex)

Profesor

Camilo Andres Redondo

Tutor

Matias Cantora

Fecha

04/03/2025

Índice

1) Introducción

Este proyecto propone una base de datos para gestionar pipelines DevOps, artefactos, despliegues e incidentes. Al centralizar los datos de flujos CI/CD, las organizaciones pueden rastrear despliegues, identificar cuellos de botella y resolver incidentes eficientemente.

Conceptos Clave:

- **DevOps (Development and Operations)**: Conjunto de prácticas, filosofía y cultura que apunta a mejorar la colaboración y comunicación entre los equipos de Desarrollo y Operaciones. Su objetivo es acortar el ciclo de vida del software, entregar rápidamente software de alta calidad, y mejorar la confiabilidad y estabilidad de las aplicaciones en producción.
- **CI/CD (Continuous Integration/Continuous Delivery)**: Grupo de prácticas y herramientas utilizadas en el desarrollo de software para automatizar el proceso de construcción, testeo y despliegue de aplicaciones.
- **Pipeline**: Un flujo de trabajo CI/CD que automatiza la construcción, pruebas y despliegue de código desde un repositorio (ej: GitHub).
- **Artefacto**: Un paquete versionado (ej: imagen Docker, archivo JAR) generado durante la ejecución de un Pipeline. Suele ser una imagen de Docker que contiene la totalidad o componentes del código de desarrollo, archivos de configuración del mismo código y del contenedor.

2) Objetivo

Diseñar una base de datos relacional que:

- Rastree ejecuciones de pipelines CI/CD.
- Monitoree entornos de despliegue y artefactos.
- Registre incidentes relacionados con los despliegues.
- Brinde visibilidad sobre responsabilidades de equipos y configuraciones de entornos de despliegue.

3) Descripción del Problema

En DevOps, ocasionalmente los equipos enfrentan:

- Falta de trazabilidad centralizada de despliegues y estados de pipelines.
- Dificultad para vincular incidentes con despliegues específicos.
- Configuraciones inconsistentes de entornos de despliegue que causan fallos.

La propuesta de una **base de datos relacional** resuelve tales problemas brindando las siguientes ventajas:

- Almacenar datos históricos para futuras auditorías.
- Garantizar consistencia de la información con claves foráneas.
- Vincular incidentes con despliegues y sus respectivos entornos.
- Mejoras continuas de procesos de desarrollo y operaciones a partir de la información obtenida.

4) Modelo de Negocio

Organización: Empresa tecnológica con múltiples equipos DevOps gestionando microservicios.
Usuarios: Ingenieros DevOps y SRE (Site Reliability Engineering), y gerentes de proyecto.
Propuesta de Valor: Reducir fallos de despliegue y tiempo de resolución de incidentes.

5) Diagrama Entidad-Relación

Diagrama

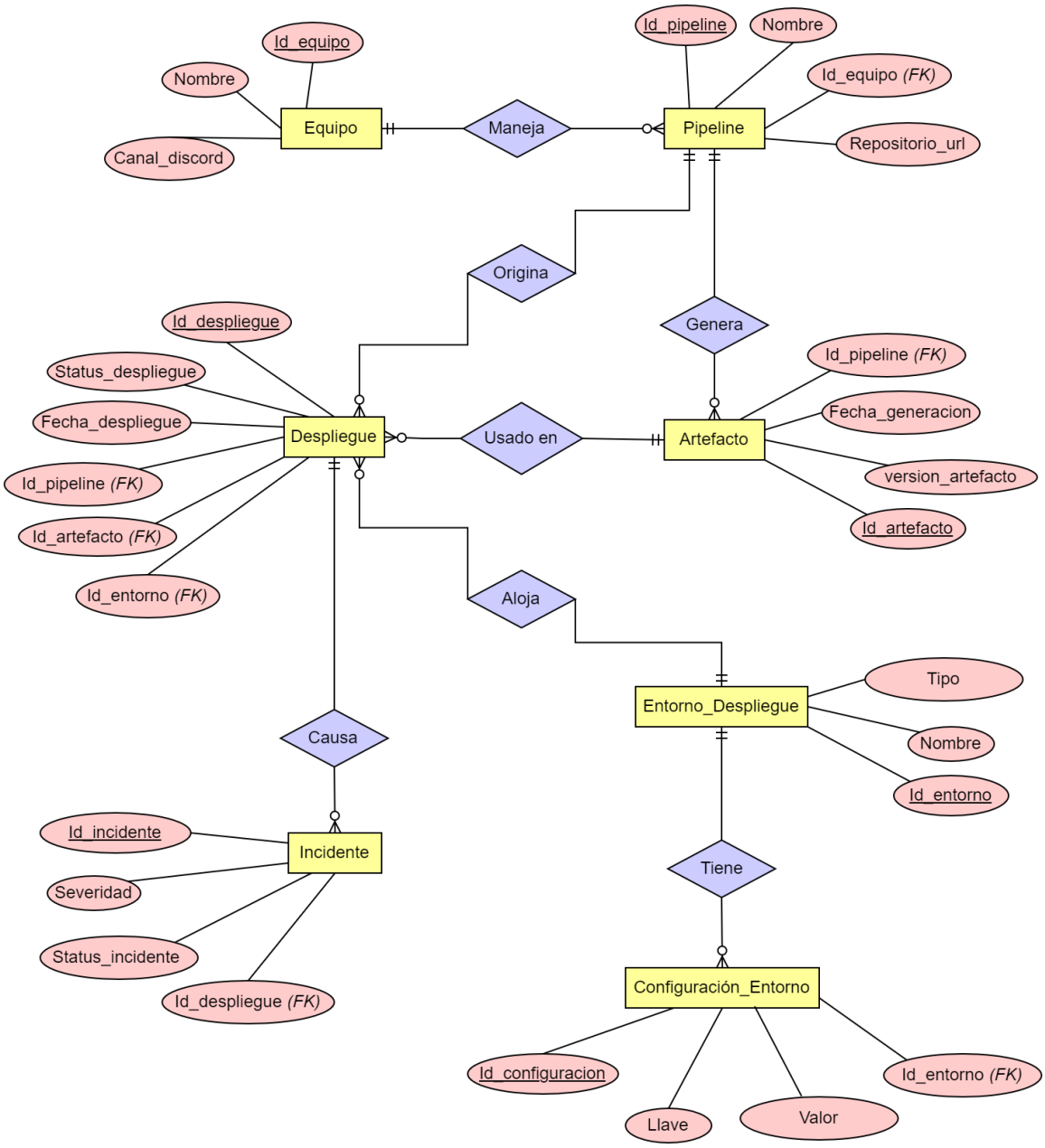

Figure 1: Diagrama Entidad-Relación

Leyenda:

- *Atributos* → Rojo
- *Entidades* → Amarillo
- *Relaciones* → Púrpura
- *Llave Primaria* → Atributo subrayado
- *Llave Foránea* → Atributo con *(FK)*

Entidades

- Equipo
- Pipeline
- Artefacto
- Despliegue
- Incidente
- Entorno_Despliegue (Entorno de Despliegue)
- Configuracion_Entorno (Configuración de Entorno de Despliegue)
- Log_Auditoria (Tabla para auditorias)

Relaciones

- Un `Equipo` puede *manejar* múltiples `Pipelines`.
- Un `Pipeline` puede *originar* múltiples `Despliegues`.
- Varios `Despliegues` pueden *usar* un mismo `Artefacto`.
- Un `Pipeline` puede *generar* múltiples `Artefactos`.
- Un `Entorno_Despliegue` puede *alojar* múltiples `Despliegues`.
- Varias `Configuraciones de Entorno` pueden ser *asignadas* a un mismo `Entorno de Despliegue`.
- Varios `Incidentes` pueden ser *causados* por un mismo `Despliegue`.

6) Listado de Tablas

Tabla "Equipo"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_equipo	INT	X		X	X		
Nombre	VARCHAR(50)			X			
Canal_discord	VARCHAR(50)		X	X			

Tabla "Pipeline"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_pipeline	INT	X		X	X		
Nombre	VARCHAR(100)			X			
Repositorio_url	VARCHAR(200)		X	X			
Id_equipo	INT			X		X	

Tabla "Artefacto"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_artefacto	INT	X		X	X		
Version_artefacto	VARCHAR(20)		X	X			
Fecha_generacion	DATETIME			X			
Id_pipeline	INT			X		X	

Tabla "Despliegue"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_despliegue	INT	X		X	X		
Status_despliegue	ENUM			X			X
Fecha_despliegue	DATETIME			X			
Id_pipeline	INT			X		X	
Id_artefacto	INT			X		X	
Id_entorno	INT			X		X	

Tabla "Incidente"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_incidente	INT	X		X	X		
Severidad	ENUM			X			
Status_incidente	ENUM			X			X
Id_despliegue	INT			X		X	

Tabla "Entorno_Despliegue"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_entorno	INT	X		X	X		
Nombre	VARCHAR(50)			X			
Tipo	ENUM			X			

Tabla "Configuracion_Entorno"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_configuracion	INT	X		X	X		
Llave	VARCHAR(50)			X			
Valor	VARCHAR(200)			X			
Id_entorno	INT			X		X	

Tabla "Log_Auditoria"

Campo	Tipo de dato	AUTO_INCREMENT	UNIQUE	NOT NULL	PK	FK	Indexado
Id_log	INT	X		X	X		
CampoNuevo_CampoAnterior	VARCHAR(255)						
accion	VARCHAR(40)						
tabla	VARCHAR(50)						
usuario	VARCHAR(100)						
fecha_modificacion	DATETIME						

7) Inserción de Datos

- La inserción de datos se realizó por medio de comandos de importación dentro del script SQL.
- Se utilizó archivos `.csv` como fuente para importar y cargar la información en cada tabla de la base de datosf. Dichos archivos se ubican en la carpeta `/csv_files` del repositorio del presente proyecto.
- Como muestra, a continuación se presenta el comando de carga para la tabla *equipo*:

```
-- Cargando data desde 'equipo.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
sistema_Monitoreo_Pipelines_DevOps/csv_files/equipo.csv'
into table equipo
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;
```

- **OJO:** Las rutas de directorio para cada caso deben ser adaptadas en función donde esté ubicado el archivo `.csv` en la máquina local donde se corre el script SQL. Se hace hincapié en el hecho que MySQL sólo acepta rutas absolutas.
- Dependiendo el caso, quizás deba modificar la configuración local del MySQL para cargar los archivos `.csv` desde cualquier directorio, de lo contrario, remover `local` del comando de carga y ubicar los archivos `.csv` en el directorio donde MySQL por defecto recupera tales archivos.
- Se utiliza `\r\n` para finales de línea en archivos `.csv` creados en Windows.
  Usar `\n` para archivos `.csv` creados en Unix/Linux.
- Tal comando se replica para cada tabla de la base de datos, modificando el nombre del archivo `.csv` y de la tabla donde se carga la información.
- En relación al contenido de los archivos `.csv`, en algunos casos introdujeron menos de 10 registros debido al contexto del proyecto:
  - `equipo.csv` → 10 equipos.
  - `pipeline.csv` → 10 pipelines - 1 por equipo.
  - `artefacto.csv` → 20 artefactos - 2 por pipeline.
  - `entorno_despliegue.csv` → 3 entornos.
  - `configuracion_entorno.csv` → 9 configuraciones - 3 por entorno de despliegue.
  - `despliegue.csv` → 10 despliegues.
  - `incidente.csv` → 5 incidentes relacionados a despliegues fallidos o pendientes.

8) Vistas

vw_despliegue_panorama

Descripción

Esta vista combina información de las siguientes tablas:

1. `despliegue:` Proveé estado y fecha del despliegue.
2. `pipeline:` Suministra nombre y repositorio de los pipelines.
3. `artefacto:` Abastece versión y fecha de generación de artefactos.
4. `entorno_despliegue:` Facilita nombre y tipo de entorno de despliegue.

La vista está diseñada para dar una vista holística de todos los despliegues, haciendo más sencilla la tarea de monitorización y análisis de los despliegues a través de los pipelines, artefactos y entornos de despliegue.

Columnas de la vista:

- `ID Despliegue`
- `Pipeline`
- `Repositorio Pipeline`
- `Versión Artefacto`
- `Fecha Generación Artefacto`
- `Estado Despliegue`
- `Fecha Despliegue`
- `Entorno Despliegue`
- `Tipo Despliegue`

Objetivo

- Suministrar una mirada centralizada sobre la actividad de los despliegues.
- Ayudar a los equipos de DevOps a identificar rápidamente:
  - Qué pipelines han sido desplegadas.
  - El estado de cada despliegue (exitoso, fallido, pendiente).
  - Las versiones de los artefactos utilizados en cada despliegue.
  - Los entornos donde los despliegues ocurren.

vw_incidente_panorama

Descripción

Esta vista combina información de las siguientes tablas:

1. `incidente:` Proveé severidad y estado de los incidentes.
2. `despliegue:` Suministra detalles sobre los despliegues.
3. `pipeline:` Abastece nombres de pipelines.
4. `entorno_despliegue:` Facilita nombre y tipo de entorno de despliegue.

La vista está diseñada para tener una vista panorámica de todos los incidentes, vinculados a despliegues, pipelines y entornos de despliegues específicos.

Columnas de la vista:

- `ID Incidente`
- `Severidad Incidente`
- `Estado Incidente`
- `Estado Despliegue`
- `Fecha Despliegue`
- `Pipeline`
- `Repositorio Pipeline`
- `Entorno Despliegue`
- `Tipo Despliegue`

Objetivo

- Obtener una mirada centralizada sobre la actividad de todos los incidentes.
- Ayudar a los equipos de DevOps a identificar rápidamente:
  - Qué despliegues causan incidentes.
  - La severidad y estado de cada incidente.
  - Los pipelines y entornos involucrados.

vw_equipo_configuracion_entorno

Descripción

Esta vista combina información de las siguientes tablas:

1. `equipo:` Proveé nombres y canales de Discord de los equipos.
2. `configuracion_entorno:` Suministra llaves y valores de los entornos de despliegue.
3. `entorno_despliegue:` Facilita nombre y tipo de entorno de despliegue.

La vista está diseñada para tener una mirada completa sobre qué equipos están asociados con configuraciones de entorno específicas, haciendo más sencilla la tarea de monitorización de la configuración de un equipo en específico.

Columnas de la vista:

- `ID Equipo`
- `Equipo`
- `Canal Discord Equipo`
- `Llave Configuración Entorno`
- `Valor Configuración Entorno`
- `Entorno Despliegue`
- `Tipo Despliegue`

Objetivo

- Abastecer una mirada centralizada sobre la configuración de equipos y su entorno de despliegue.
- Ayudar a los equipos de DevOps a identificar rápidamente:
  - Qué equipos son asociados con ciertos entornos de despliegue.
  - La configuración específica para cada entorno de despliegue.
  - El tipo de entorno de despliegue para cada configuración de entorno.

9) Funciones

fn_getStatusDespliegue

Descripción

Esta función retorna el estado de un despliegue para un ID de despliegue en específico. Analizar la tabla `despliegue` para determinar si un despliegue fue exitoso, fallido o está pendiente.

Tabla involucrada:

1. `despliegue` Cuenta con los detalles de los despliegues, incluyendo `status_despliegue`

Estructura:

- *Parámetros:* `p_id_despliegue int` → ID del despliegue.
- *Retorna:* `v_status_despliegue varchar(20)` → Estado del despliegue (Success, Failed, Pending).

Objetivo

- Abastecer una forma rápida de verificar el estado de un despliegue.
- Simplificar las consultas relacionadas al estado de los despliegues.

fn_countIncidentes

Descripción

Esta función cuenta el número de incidentes para un nivel de severidad dado (Low, Medium, High). Consulta la tabla `incidente` para contar incidentes en base de la severidad.

Tabla involucrada:

1. `incidente` Cuenta con los detalles de los incidentes, incluyendo `severidad`

Estructura:

- *Parámetros:* `p_severidad varchar(20)` → Severidad del incidente (Low, Medium, High).
- *Retorna:* `v_count int` → Cantidad de incidentes.

Objetivo

- Suministrar una forma rápida de contar incidentes a partir de la severidad.
- Ayudar a los equipos a priorizar la resolución de incidentes en función de la severidad de los mismos.

10) Stored Procedures

sp_getDespliegues

Descripción

Este procedimiento recupera todos los despliegues a partir de un estado de despliegue específico (Success, Failed, Pending). Consulta la tabla `despliegue` y devuelve los detalles de los despliegues junto con información asociada del pipeline y el entorno de despliegue.

Tablas involucradas:

1. `despliegue` Contiene detalles de los despliegues, incluyendo `status_despliegue`
2. `pipeline` Proporciona nombres de pipelines y URLs de repositorios
3. `entorno_despliegue` Proporciona nombres y tipos de entornos

Estructura:

- *Parámetros:* `in p_status_despliegue varchar(20)` → Estado de despliegue (Success, Failed, Pending).

Objetivo

- Proporcionar una forma de filtrar despliegues por estado de despliegue.
- Devolver información detallada sobre los despliegues, incluyendo detalles del pipeline y el entorno.

sp_agregarNuevoArtefacto

Descripción

Este procedimiento agrega un nuevo artefacto a la tabla `artefacto`. Toma la versión del artefacto, la fecha de generación y el ID del pipeline como parámetros de entrada e inserta un nuevo registro en la tabla `artefacto`.

Tablas involucradas:

1. `artefacto` Contiene detalles de los artefactos, incluyendo la versión y la fecha de generación.
2. `pipeline` Valida que el ID del pipeline proporcionado exista.

Estructura:

- *Parámetros:* `in p_id_pipeline int` → ID del pipeline, `in p_version_artefacto varchar(20)` → Versión del artefacto, `in p_fecha_generación datetime` → Fecha de generación del artefacto.

Objetivo

- Simplificar el proceso de agregar nuevos artefactos.
- Asegurar la consistencia de los datos validando el ID del pipeline.

11) Triggers

tr_beforeInsertarDespliegue

Descripción

Este trigger se aplica `BEFORE INSERT` en la tabla `despliegue`. Asegura que el campo `status_despliegue` se establezca en 'Pending' si no se proporciona un estado durante la inserción de un nuevo despliegue.

Tabla involucrada:

1. `despliegue:` Contiene detalles de los despliegues, incluyendo `status_despliegue`.

Objetivo

- Forzar un valor predeterminado para el campo `status_despliegue`.
- Asegurar la consistencia de los datos evitando valores nulos o inválidos.

tr_beforeUpdateIncidenteStatus

Descripción

Este trigger se aplica `BEFORE UPDATE` en la tabla `incidente`. Guarda un nuevo registro en la tabla separada `log_auditoria` cada vez que se actualiza el campo `status_incidente`. Dicho guardado de registro sucede sólo si exclusivamente se modifica el estado del incidente.

Tabla involucrada:

1. `incidente:` Contiene detalles de los incidentes, incluyendo `status_incidente`.
2. `incidente_log:` Una nueva tabla para almacenar los registros de cambios en diferentes campos de la base de datos, incluyendo el estado de incidentes. Sirve como una tabla de auditoría.

Objetivo

- Registrar automáticamente los cambios en los estados de los incidentes.
- Proporcionar un historial de auditoría para la gestión de incidentes.

12) Script SQL

```
-- ####################################
-- ##### Creación de base de datos #####
-- ####################################


DROP DATABASE IF EXISTS monitoreo_devops;
CREATE DATABASE IF NOT EXISTS monitoreo_devops;
USE monitoreo_devops;


-- ##############################
-- ##### Creación de tablas #####
-- ##############################


DROP TABLE IF EXISTS equipo;
CREATE TABLE IF NOT EXISTS equipo (
    id_equipo INT AUTO_INCREMENT,
    nombre VARCHAR(50) NOT NULL,
    canal_discord VARCHAR(50) NOT NULL UNIQUE,
    CONSTRAINT pk_equipo PRIMARY KEY (id_equipo)
);

DROP TABLE IF EXISTS pipeline;
CREATE TABLE IF NOT EXISTS pipeline (
    id_pipeline INT AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    repositorio_url VARCHAR(200) NOT NULL UNIQUE,
    id_equipo INT NOT NULL,
    CONSTRAINT pk_pipeline PRIMARY KEY (id_pipeline),
    CONSTRAINT fk_pipeline_equipo FOREIGN KEY (id_equipo)
        REFERENCES equipo (id_equipo)
);

DROP TABLE IF EXISTS artefacto;
CREATE TABLE IF NOT EXISTS artefacto (
    id_artefacto INT AUTO_INCREMENT,
    version_artefacto VARCHAR(20) NOT NULL UNIQUE,
    fecha_generacion DATETIME NOT NULL,
    id_pipeline INT NOT NULL,
    CONSTRAINT pk_artefacto PRIMARY KEY (id_artefacto),
    CONSTRAINT fk_artefacto_pipeline FOREIGN KEY (id_pipeline)
        REFERENCES pipeline (id_pipeline)
```

```
);

DROP TABLE IF EXISTS entorno_despliegue;
CREATE TABLE IF NOT EXISTS entorno_despliegue (
    id_entorno INT AUTO_INCREMENT,
    nombre VARCHAR(50) NOT NULL,
    tipo ENUM('Prod', 'Staging', 'Test') NOT NULL,
    CONSTRAINT pk_entorno PRIMARY KEY (id_entorno)
);

DROP TABLE IF EXISTS configuracion_entorno;
CREATE TABLE IF NOT EXISTS configuracion_entorno (
    id_configuracion INT AUTO_INCREMENT,
    llave VARCHAR(50) NOT NULL,
    valor VARCHAR(200) NOT NULL,
    id_entorno INT NOT NULL,
    CONSTRAINT pk_configuracion PRIMARY KEY (id_configuracion),
    CONSTRAINT fk_configuracion_entorno FOREIGN KEY (id_entorno)
        REFERENCES entorno_despliegue (id_entorno)
);

DROP TABLE IF EXISTS despliegue;
CREATE TABLE IF NOT EXISTS despliegue (
    id_despliegue INT AUTO_INCREMENT,
    status_despliegue ENUM('Success', 'Failed', 'Pending') NOT NULL,
    fecha_despliegue DATETIME NOT NULL,
    id_pipeline INT NOT NULL,
    id_artefacto INT NOT NULL,
    id_entorno INT NOT NULL,
    CONSTRAINT pk_despliegue PRIMARY KEY (id_despliegue),
    CONSTRAINT fk_despliegue_pipeline FOREIGN KEY (id_pipeline)
        REFERENCES pipeline (id_pipeline),
    CONSTRAINT fk_despliegue_artefacto FOREIGN KEY (id_artefacto)
        REFERENCES artefacto (id_artefacto),
    CONSTRAINT fk_despliegue_entorno FOREIGN KEY (id_entorno)
        REFERENCES entorno_despliegue (id_entorno),
    INDEX idx_status (status_despliegue)
);

DROP TABLE IF EXISTS incidente;
CREATE TABLE IF NOT EXISTS incidente (
    id_incidente INT AUTO_INCREMENT,
    severidad ENUM('Low', 'Medium', 'High') NOT NULL,
    status_incidente ENUM('Open', 'Closed') NOT NULL,
    id_despliegue INT NOT NULL,
    CONSTRAINT pk_incidente PRIMARY KEY (id_incidente),
    CONSTRAINT fk_incidente_despliegue FOREIGN KEY (id_despliegue)
        REFERENCES despliegue (id_despliegue),
    INDEX idx_status (status_incidente)
);

-- Tabla de auditoría
drop table if exists log_auditoria;
create table if not exists log_auditoria (
```

```
        id_log int auto_increment,
    campoNuevo_campoAnterior varchar(255),
    accion varchar(40),
    tabla varchar(50),
    usuario varchar(100),
    fecha_modificacion datetime,
    constraint pk_incidente_log
                primary key (id_log)
);


-- ##############################
-- ##### Inserción de datos #####
-- ##############################


-- OJO: Las rutas de directorio para cada caso deben ser adaptadas en función
-- donde esté ubicado el archivo .csv en la máquina local donde se corre el
-- script SQL. MySQL sólo acepta rutas absolutas.
-- Dependiendo el caso, quizás deba modificar la configuración local del MySQL
-- para cargar los archivos .csv desde cualquier directorio, de lo contrario,
-- remover 'local' de los comandos de carga y ubicar los archivos .csv en el
-- directorio donde MySQL por defecto recupera tales archivos.

-- Cargando data desde 'equipo.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
sistema_Monitoreo_Pipelines_DevOps/csv_files/equipo.csv'
into table equipo
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n' -- Se usa '\r\n' para finales de línea en archivos .csv creados
-- en Windows. Usar '\n' para archivos .csv creados en Unix/Linux.
ignore 1 rows;

-- Cargando data desde 'pipeline.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
sistema_Monitoreo_Pipelines_DevOps/csv_files/pipeline.csv'
into table pipeline
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;

-- Cargando data desde 'artefacto.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
sistema_Monitoreo_Pipelines_DevOps/csv_files/artefacto.csv'
into table artefacto
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;

-- Cargando data desde 'entorno_despliegue.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
```

```
sistema_Monitoreo_Pipelines_DevOps/csv_files/entorno_despliegue.csv'
into table entorno_despliegue
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;

-- Cargando data desde 'configuracion_entorno.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
sistema_Monitoreo_Pipelines_DevOps/csv_files/configuracion_entorno.csv'
into table configuracion_entorno
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;

-- Cargando data desde 'despliegue.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
sistema_Monitoreo_Pipelines_DevOps/csv_files/despliegue.csv'
into table despliegue
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;

-- Cargando data desde 'incidente.csv'
load data local infile 'C:/Users/Luigg/coderhouse/sql/4_[Primera_Entrega]/
sistema_Monitoreo_Pipelines_DevOps/csv_files/incidente.csv'
into table incidente
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;


-- ##################
-- ##### Vistas #####
-- ##################


create or replace view vw_despliegue_panorama as
select d.id_despliegue 'ID Despliegue', p.nombre 'Pipeline',
           p.repositorio_url 'Repositorio Pipeline',
           a.version_artefacto 'Versión Artefacto',
       a.fecha_generacion 'Fecha Generación Artefacto',
       d.status_despliegue 'Estado Despliegue',
       d.fecha_despliegue 'Fecha Despliegue',
       e.nombre 'Entorno Despliegue',
       e.tipo 'Tipo Despliegue'
from despliegue d
        join pipeline p on d.id_pipeline = p.id_pipeline
    join artefacto a on d.id_artefacto = a.id_artefacto
    join entorno_despliegue e on d.id_entorno = e.id_entorno;
```

```sql
create or replace view vw_incidente_panorama as
select i.id_incidente 'ID Incidente', i.severidad 'Severidad Incidente',
           i.status_incidente 'Estado Incidente',
       d.status_despliegue 'Estado Despliegue',
       d.fecha_despliegue 'Fecha Despliegue',
       p.nombre 'Pipeline',
       p.repositorio_url 'Repositorio Pipeline',
       e.nombre 'Entorno Despliegue',
       e.tipo 'Tipo Despliegue'
from incidente i
        join despliegue d on i.id_despliegue = d.id_despliegue
    join pipeline p on d.id_pipeline = p.id_pipeline
    join entorno_despliegue e on d.id_entorno = e.id_entorno;

create or replace view vw_equipo_configuracion_entorno as
select e.id_equipo 'ID Equipo', e.nombre 'Equipo',
           e.canal_discord 'Canal Discord Equipo',
       c.llave 'Llave Configuración Entorno',
       c.valor 'Valor Configuración Entorno',
       ed.nombre 'Entorno Despliegue',
       ed.tipo 'Tipo Despliegue'
from equipo e
        join pipeline p on e.id_equipo = p.id_equipo
    join despliegue d on d.id_pipeline = p.id_pipeline
    join entorno_despliegue ed on d.id_entorno = ed.id_entorno
    join configuracion_entorno c on ed.id_entorno = c.id_entorno;


-- ######################
-- ##### Funciones #####
-- ######################


drop function if exists fn_getStatusDespliegue;
delimiter //
create function fn_getStatusDespliegue(p_id_despliegue int)
returns varchar(20)
deterministic
begin
        declare v_status_despliegue varchar(20);

    select status_despliegue
    into v_status_despliegue
    from despliegue
    where id_despliegue = p_id_despliegue;

    return v_status_despliegue;
end //
delimiter ;

drop function if exists fn_countIncidentes;
delimiter //
create function fn_countIncidentes(p_severidad varchar(20))
returns int
```

```
deterministic
begin
        declare v_count int;

        select count(*)
    into v_count
    from incidente
    where severidad = p_severidad;

    return v_count;
end //
delimiter ;


-- ############################
-- ##### Stored Procedures #####
-- ############################


drop procedure if exists sp_getDespliegues;
delimiter //
create procedure sp_getDespliegues(in p_status_despliegue varchar(20))
begin
        select d.id_despliegue 'ID Despliegue',
               p.nombre 'Pipeline',
               p.repositorio_url 'Repositorio Pipeline',
               e.nombre 'Entorno Despliegue',
               e.tipo 'Tipo Entorno Despliegue',
               d.fecha_despliegue 'Fecha Despliegue'
        from despliegue d
                join pipeline p on d.id_pipeline = p.id_pipeline
        join entorno_despliegue e on d.id_entorno = e.id_entorno
        where d.status_despliegue = p_status_despliegue;
end //
delimiter ;

drop procedure if exists sp_agregarNuevoArtefacto;
delimiter //
create procedure sp_agregarNuevoArtefacto(in p_id_pipeline int,
                                          in p_version_artefacto varchar(20),
                                          in p_fecha_generacion datetime)
begin
    if exists (select 1 from pipeline p where p.id_pipeline = p_id_pipeline) then
        insert into artefacto(version_artefacto, fecha_generacion, id_pipeline)
        values
           (p_version_artefacto, p_fecha_generacion, p_id_pipeline);
    else
        signal sqlstate '45000'
        set message_text = 'id_pipeline inválido';
    end if;
end //
delimiter ;
```

```
-- ####################
-- ##### Triggers #####
-- ####################


drop trigger if exists tr_beforeInsertarDespliegue;
delimiter //
create trigger tr_beforeInsertarDespliegue
before insert on despliegue
for each row
begin
    if new.status_despliegue is null then
                set new.status_despliegue = 'Pending';
    end if;
end //
delimiter ;

drop trigger if exists tr_beforeUpdateIncidenteStatus;
delimiter //
create trigger tr_beforeUpdateIncidenteStatus
before update on incidente
for each row
begin
    -- Se loguea el cambio de 'status_incidente' en la tabla auditoria
    -- sólo si dicho campo es actualizado.
        if old.status_incidente <> new.status_incidente then
                insert into log_auditoria(campoNuevo_campoAnterior, accion, tabla,
                                                                    usuario, fecha_modificacion)
        values
                        (concat('STATUS_INCIDENTE NUEVO: ', new.status_incidente,
                                        ', STATUS_INCIDENTE ANTERIOR: ', old.status_incidente),
            'UPDATE',
            'INCIDENTE',
            current_user(),
            now());
    end if;
end //
delimiter ;
```