PROYECTO SISTEMA DE GESTION DE EVENTOS SIGEU

RODRIGO ANDRÉS GÓMEZ LÓPEZ. Código: 2247014

SISTEMAS OPERATIVOS I

Presentado a:

XX



UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

INGENIERÍA DE DATOS E INTELIGENCIA ARTIFICIAL

SANTIAGO DE CALI

2025

# 1. OBJETIVOS

## 1.1 OBJETIVO GENERAL

Implementar dos soluciones de almacenamiento, una relacional (SQL) y otra documental (NoSQL), para el Sistema de Gestión de Eventos Universitarios (SIGEU), con el fin de respaldar su backend inicial y comparar su eficiencia, escalabilidad y adaptabilidad.

## 1.2 OBJETIVOS ESPECIFICOS

1.2.1 Modelar e implementar la solución de almacenamiento relacional (SQL)

1.2.2 Modelar e implementar la solución de almacenamiento documental (NoSQL)

1.2.3 Desarrollar un backend mínimo con API REST

1.2.4 Definir, ejecutar y documentar pruebas comparativas entre SQL y NoSQL

# 2. METODOLOGÍA

Para el desarrollo de este proyecto se abordarán dos metodologías complementarias. La primera enfocada al trabajo en equipo, y la otra aplicada al desarrollo técnico del sistema. Adicionalmente, se definen las tecnologías que se utilizaran para el desarrollo.

En este sentido, las metodologías elegidas son:

1. SCRUM para el trabajo en equipo, ya que permite organizar el proyecto en sprints cortos, asignar responsabilidades claras y fomentar la colaboración continua entre los integrantes. Esta metodología garantiza que los avances se entreguen de manera progresiva y que los ajustes se realicen de forma ágil.
2. CRISP-DM para la metodología de desarrollo, considerando que, aunque fue creada para minería de datos, se adapta muy bien a un proyecto como SIGEU. Esto porque plantea un ciclo claro: comprender el problema, preparar los datos, modelar, evaluar y desplegar, lo cual se ajusta al flujo de trabajo que implica construir y comparar soluciones de almacenamiento SQL y NoSQL.

## 2.1 SCRUM – Trabajo en equipo

SCRUM es una metodología ágil de gestión de proyectos que divide el trabajo en ciclos cortos llamados *sprints* y promueve la comunicación constante entre los miembros del equipo.

Aplicación en SIGEU:

* Roles:
  + *Product Owner (PO):* líder del equipo (define prioridades y comunicación con el docente).
  + *Scrum Master:* miembro que guía el cumplimiento de la metodología y resuelve bloqueos.
  + *Equipo de desarrollo:* todos los integrantes, responsables de las tareas técnicas.
* Artefactos: backlog de tareas (lista completa de actividades), sprint backlog (tareas de cada ciclo) e incrementos (resultados parciales).
* Eventos:
  + *Sprint planning:* planificación semanal de tareas.
  + *Daily stand-up:* reunión breve para revisar avances y bloqueos.
  + *Sprint review:* muestra de resultados en cada entrega.
  + *Retrospectiva:* análisis de mejoras para el siguiente ciclo.

## 2.2 CRISP-DM – Desarrollo del sistema

CRISP-DM (Cross Industry Standard Process for Data Mining) es una metodología estructurada para proyectos de datos, que organiza el desarrollo en seis fases iterativas: comprensión, preparación, modelado, evaluación y despliegue.

Aplicación en SIGEU:

* Comprensión del negocio: análisis del enunciado y definición de requerimientos.
* Comprensión de los datos: identificación de entidades clave (eventos, usuarios, organizaciones, documentos).
* Preparación de datos: construcción del MER/MR para SQL y definición del modelo documental en NoSQL.
* Modelado: implementación de la base relacional en MySQL y de colecciones en MongoDB.
* Evaluación: pruebas de consultas, validación de la API y comparación SQL vs NoSQL.
* Despliegue académico: documentación, sustentación y entrega de resultados.

## 2.3. TECNOLOGÍAS UTILIZADAS.

Dentro del desarrollo del presente proyecot se hace uso de las siguientes tecnologías:

* XAMPP
  + Uso: Paquete que incluye Apache, MySQL, PHPMyAdmin y otros servicios.
  + Aporte: Provee un entorno local fácil de instalar para correr el motor MySQL y gestionar la base de datos desde PHPMyAdmin.
* MySQL Workbench
  + Uso: Herramienta oficial de diseño y administración de MySQL.
  + Aporte: Permite modelar gráficamente el MER/MR, ejecutar scripts SQL, crear consultas simples y anidadas, y administrar la base relacional.
* MySQL (SQL relacional)
  + Uso: Motor de base de datos relacional en el que se implementará el MER/MR con tablas, relaciones, restricciones, vistas y procedimientos.
  + Aporte: Ofrece integridad referencial, soporte a consultas complejas y robustez para el backend del sistema.
* MongoDB (NoSQL documental)
  + Uso: Motor NoSQL basado en documentos JSON para representar información de eventos y usuarios con flexibilidad.
  + Aporte: Escalabilidad y adaptación para consultas rápidas y estructuras de datos no rígidas.
* Swagger / OpenAPI
  + Uso: Documentación interactiva de la API REST.
  + Aporte: Prueba y validación de endpoints por parte del equipo.
* Postman
  + Uso: Ejecución de pruebas sobre la API y validación de respuestas.
  + Aporte: Facilita la verificación de casos de uso funcionales y pruebas de rendimiento.
* GitHub
  + Uso: Repositorio de código y documentación colaborativa.
  + Aporte: Control de versiones, registro de aportes y Wiki del proyecto.
* Trello / Jira
  + Uso: Organización de tareas bajo metodología ágil (SCRUM).
  + Aporte: Seguimiento de backlog, roles y sprints.
* Python (FastAPI)
  + Uso: Framework backend para exponer la API mínima conectada a SQL y NoSQL.
  + Aporte: Comunicación entre bases de datos y pruebas de endpoints.
* Visual Studio Code
* Uso: Entorno de desarrollo de código.
* Aporte: Integración con GitHub, ejecución de scripts, soporte para SQL y MongoDB.

1. PlantUML / StarUML

* Uso: Creación de diagramas UML (clases, procesos, secuencia).
* Aporte: Documentación gráfica del análisis y diseño.

# 3. REQUERIMIENTOS

En un proyecto de desarrollo de software, los requerimientos son la base para guiar el diseño, la implementación y las pruebas. En este sentido, dentro de la metodología SCRUM (trabajo en equipo), los requerimientos se reflejan en el Product Backlog, es decir, en la lista priorizada de funcionalidades a construir.

Por otra parte, en la metodología CRISP-DM (desarrollo), este punto corresponde a la fase de Comprensión del Negocio y de los Datos, donde se identifican las necesidades del sistema y se expresan en requerimientos claros.

Por lo anterior, en el proyecto SIGEU, los requerimientos se dividen en:

1. Requerimientos Funcionales (RF): describen qué debe hacer el sistema.
2. Requerimientos No Funcionales (RNF): describen cómo debe comportarse el sistema (rendimiento, seguridad, escalabilidad, etc.).
3. Reglas de Negocio (RN): condiciones que siempre deben cumplirse según la lógica institucional.
4. Actores y Roles: usuarios que interactúan con el sistema, con sus responsabilidades y permisos.

3.1 ANALIS DEL PROYECTO Y DEFINICION DE REQUERIMIENTOS

Para este ejercicio, es necesario hacer lectura del enunciado del proyecto y poder definir los requerimientos:

Párrafo 1:

*“Este proyecto tiene como objetivo principal el modelado e implementación de dos soluciones de almacenamiento destinadas a respaldar el desarrollo inicial del backend de una aplicación web para la gestión de eventos en una institución universitaria. Una de las soluciones empleará bases de datos relacionales y la otra base de datos NoSQL, lo que permitirá comparar y evaluar diferentes enfoques de almacenamiento orientados a garantizar una gestión eficiente de los datos y un funcionamiento óptimo de la futura aplicación.”*

De acuerdo al texto anterior, el proyecto busca construir dos soluciones de almacenamiento de datos para el sistema SIGEU (Gestión de Eventos Universitarios):

* Una solución con base de datos relacional (SQL).
* Otra solución con base de datos NoSQL (documental).

El propósito es respaldar el backend inicial de la aplicación y, al mismo tiempo, comparar ambos enfoques, evaluando su capacidad de:

* Manejar los datos con eficiencia.
* Garantizar escalabilidad y adaptación.
* Asegurar un rendimiento óptimo de la aplicación en el futuro.

En este sentido los requerimientos iniciales son los siguientes:

Tabla 1. Requerimientos parciales 1.

|  |  |
| --- | --- |
| REQUERIMIENTOS FUNCIONALES (RF) | |
| RF-01 | Implementar almacenamiento relacional (SQL) para SIGEU. |
| RF-02 | Implementar almacenamiento NoSQL (documental) para SIGEU. |
| RF-03 | Desarrollar un backend mínimo de prueba que conecte con ambas bases. |
| RF-04 | Permitir la ejecución de consultas y operaciones básicas sobre ambos almacenamientos. |
|  |  |
| REQUERIMIENTOS NO FUNCIONALES (RNF) | |
| RNF-01 | El sistema debe ofrecer eficiencia en las operaciones (consultas rápidas). |
| RNF-02 | Debe ser escalable, soportando crecimiento en usuarios y datos. |
| RNF-03 | Debe ser adaptable, permitiendo cambios futuros sin perder rendimiento. |
| RNF-04 | Se debe comparar y documentar SQL vs NoSQL en métricas de latencia, facilidad y escalabilidad. |
|  |  |
| MODELO DE DATOS (ALTO NIVEL, INICIAL) | |
| Entidades centrales | Evento, Usuario/Rol, Organización externa, Documentos (PDFs). |
|  |  |
| REGLAS DE NEGOCIO | |
| RN-01 | El proyecto debe entregar dos soluciones de almacenamiento (SQL y NoSQL). |
| RN-02 | La entrega debe incluir un informe comparativo entre ambos enfoques. |
|  |  |
| PROCESOS PRINCIPALES | |
| P1 | Implementar almacenamiento SQL. |
| P2 | Implementar almacenamiento NoSQL. |
| P3 | Ejecutar pruebas de validación. |
| P4 | Comparar resultados y elaborar informe. |
|  |  |
| ACTORES (MARCADORES INICIALES) | |
| Rol-1 | Equipo de desarrollo |
| Rol-2 | Docente |
| Rol-3 | Los roles de usuario final (estudiante, docente, secretario académico) |
|  |  |
| API MÍNIMA ESPERADA | |
| API-1 | Endpoint de salud: GET /health. |
| API-2 | Diagnóstico SQL: GET /diagnostico/sql. |
| API-3 | Diagnóstico NoSQL: GET /diagnostico/nosql. |
|  |  |
| PRUEBAS | |
| PRUEBA - 1 | Verificar que se crean las tablas en SQL y las colecciones en NoSQL. |
| PRUEBA – 2 | Validar consultas básicas en ambos entornos. |
| PRUEBA - 3 | Medir tiempo de respuesta en operaciones clave. |
|  |  |
| PENDIENTES / SUPUESTOS | |
| P/S -1 | Confirmar tecnologías exactas: (ej. MySQL y MongoDB) |
| P/S - 2 | Definir framework backend para pruebas (Node.js, Python FastAPI, etc.). |
|  |  |
| RIESGOS INICIALES | |
| RIESG - 1 | Subestimar el trabajo al implementar dos modelos de datos paralelos. |
| RIESG - 2 | No definir criterios claros de comparación. |

Párrafo 2.

*“Es importante destacar que el proyecto no contempla el desarrollo completo del backend, sino que se centra exclusivamente en la implementación de las soluciones de almacenamiento, garantizando que los modelos y bases de datos diseñados sean consistentes, escalables y cumplan con las necesidades de la aplicación en su etapa inicial de desarrollo.”*

Ahora bien, este parte del enunciado aclara que el proyecto no incluye la construcción del backend completo de la aplicación.  
Básicamente, el foco está en la implementación de las soluciones de almacenamiento (SQL y NoSQL), asegurando que:

* Los modelos y bases de datos sean consistentes (esquemas correctos y sin contradicciones).
* Sean escalables (puedan crecer en datos y usuarios).
* Respondan adecuadamente a las necesidades de la aplicación en su fase inicial (backend mínimo para pruebas, no despliegue completo).

Tabla 2. Requerimientos parciales 2.

|  |  |
| --- | --- |
| REQUERIMIENTOS FUNCIONALES (RF) | |
| RF-05 | Implementar únicamente las soluciones de almacenamiento (SQL y NoSQL) sin desarrollar todo el backend. |
| RF-06 | Asegurar que las bases de datos creadas soporten las operaciones necesarias del backend mínimo. |
|  |  |
| REQUERIMIENTOS NO FUNCIONALES (RNF) | |
| RNF-05 | Los modelos deben ser consistentes (sin ambigüedades en esquemas ni contradicciones en relaciones). |
| RNF-06 | Las soluciones deben ser escalables, soportando crecimiento progresivo de datos. |
| RNF-07 | Las soluciones deben estar alineadas a las necesidades iniciales de la aplicación (no se pide sistema productivo, solo pruebas académicas). |
|  |  |
| MODELO DE DATOS (ALTO NIVEL, INICIAL) | |
| Entidades centrales | Evento, Usuario/Rol, Organización externa, Documentos (PDFs).  En esta fase se reafirma que los modelos deben ser consistentes y escalables; en este sentido, se refuerza la identificación de las entidades mencioanadas |
|  |  |
| REGLAS DE NEGOCIO | |
| RN-03 | El sistema debe excluir el desarrollo del backend completo. |
| RN-04 | Los modelos deben cumplir con las necesidades iniciales de la aplicación (fase temprana). |
|  |  |
| PROCESOS PRINCIPALES (REFORZANDO PÁRRAFO 1) | |
| P5 | Limitar el alcance al almacenamiento, excluyendo backend completo |
| P6 | Validar consistencia y escalabilidad de los modelos antes de pruebas. |
|  |  |
| ACTORES (MARCADORES INICIALES) | |
| Rol-1 | Equipo de desarrollo (responsable de asegurar consistencia/escalabilidad) |
| Rol-2 | Docente (validador del cumplimiento del alcance limitado) |
| Rol-3 | Los roles de usuario final (estudiante, docente, secretario académico) |
|  |  |
| API MÍNIMA ESPERADA | |
|  | Nota: Se mantiene como API de prueba básica (health y diagnósticos). |
|  | Nota 2: Se confirma que no se debe extender hacia backend completo. |
|  |  |
| PRUEBAS | |
| PRUEBA – 4 | Validar consistencia del modelo (integridad referencial en SQL, esquema válido en NoSQL). |
| PRUEBA – 5 | Simular escalabilidad con dataset de prueba mayor a lo mínimo esperado. |
|  |  |
| PENDIENTES / SUPUESTOS | |
| P/S -3 | Confirmar tamaño de dataset para validar escalabilidad. |
| P/S – 4 | Definir criterios de consistencia para comparar SQL vs NoSQL. |
|  |  |
| RIESGOS INICIALES | |
| RIESG – 3 | Intentar desarrollar backend más allá del alcance definido (riesgo de dispersión). |
| RIESG – 4 | Diseñar modelos poco escalables que no resistan pruebas de crecimiento. |

Párrafo 3

*“En una institución universitaria se desea construir una aplicación que permita llevar a cabo el registro, aprobación y publicación de los eventos organizados por la comunidad académica (docentes y/o estudiantes) de los distintos programas adscritos a las unidades académicas de la universidad.”*

Tabla 3. Requerimientos parciales 3.

|  |  |
| --- | --- |
| REQUERIMIENTOS FUNCIONALES (RF) | |
| RF-07 | Permitir el registro de eventos por parte de docentes y estudiantes. |
| RF-08 | Permitir la aprobación de eventos por parte de la secretaría académica. |
| RF-09 | Permitir la publicación de eventos aprobados |
| REQUERIMIENTOS NO FUNCIONALES (RNF) | |
| RNF-08 | El sistema debe asegurar la trazabilidad del evento desde su registro hasta su publicación. |
|  |  |
| MODELO DE DATOS (ALTO NIVEL, INICIAL) | |
| Entidades centrales | Entidad Evento (ya identificada en P1). |
|  | Relación con Usuario (Docente/Estudiante como organizadores). |
|  | Relación con UnidadAcadémica (programa/unidad adscrita). |
|  | |
| REGLAS DE NEGOCIO | |
| RN-05 | Todo evento debe pasar por las fases de registro, aprobación, publicación. |
|  |  |
| PROCESOS PRINCIPALES (REFORZANDO PÁRRAFO 1) | |
| P7 | Registrar evento. |
| P8 | Aprobar evento |
| P9 | Publicar evento |
| ACTORES (MARCADORES INICIALES) | |
| Rol-4 | Estudiante (registrar eventos). |
| Rol-5 | Docente (registrar eventos). |
| Rol-6 | Secretaría académica (aprobar/publicar eventos). |
|  |  |
| API MÍNIMA ESPERADA | |
| API-4 | POST /eventos : registrar evento. |
| API-5 | PATCH /eventos/{id}/aprobar : aprobar evento. |
| API-6 | PATCH /eventos/{id}/publicar : publicar evento. |
| PRUEBAS | |
| PRUEBA – 6 | Validar que un estudiante pueda registrar un evento. |
| PRUEBA – 7 | Verificar que un evento no pueda publicarse sin aprobación previa. |
|  |  |
| PENDIENTES / SUPUESTOS | |
| P/S – 5 | Confirmar qué información mínima requiere un evento (ej. título, tipo, fecha, lugar). |
|  |  |
| RIESGOS INICIALES | |
| RIESG – 5 | Posibilidad de que eventos sean publicados sin pasar por aprobación (inconsistencia). |

4. MODELADO DE LA BASE DE DATOS

## 4.1 MODELO DE DATOS RELACIONAL (MER/MR)

En esta etapa modelamos el “QUE” debe persistirse y cómo se relaciona, para soportar los procesos de registro, evaluación, publicación y participación externa.

Ahora bien, en cuento a la metodología de desarrollo CRISP-DM, este punto hace referencia a la etapa de Modeling, usando los hallazgos de Business Understanding y Data Understanding (requerimientos consolidados) para construir estructuras que garanticen consistencia y trazabilidad.

Por otra parte, en cuanto a la metodología de trabajo en equipo (SCRUM): este entregable hace parte del Sprint 2 como base del desarrollo técnico y de las pruebas.

En este punto, se estructura el modelado teniendo en cuenta las entidades principales. Dentro de los pasos relevante están:

1. MER Y MR

2. Crea el esquema, tablas, PK/FK y restricciones

3. Llena tablas con datos de la UAO (correos, instalaciones, eventos, etc.).

4. agrega vistas, triggers y demás objetos para el flujo de negocio (aprobaciones, notificaciones, etc.).

4.1.1 MER

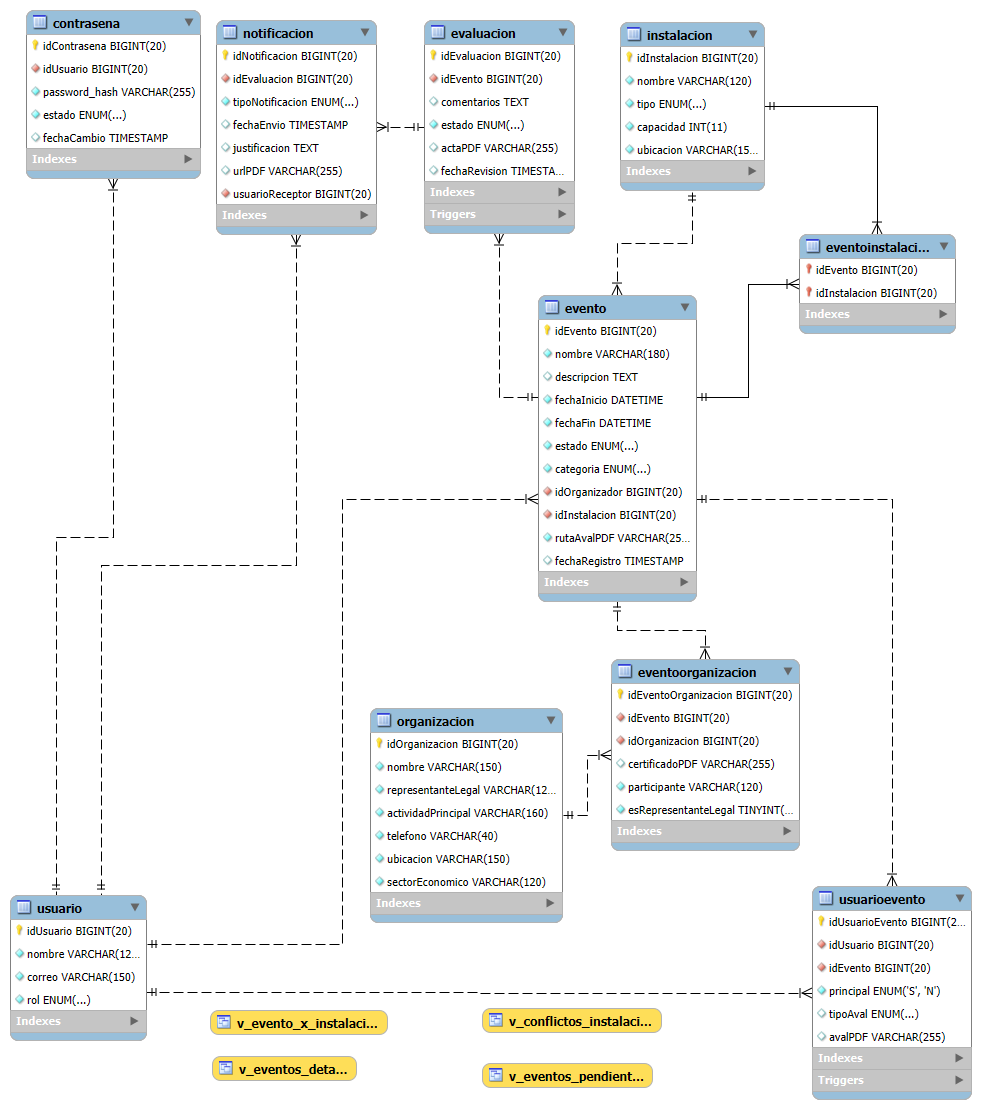
A continuación se muestra el MER ajustado según análisis de requerimientos y sugerencias del docente.

Diagrama

El contenido generado por IA puede ser incorrecto.

4.1.2 MR

A continuación se muestra el MR elaborado a partir del MER



4.1.3 TABLAS

En primer lugar, se creo una consulta con el nombre 1\_CREAR\_BASE\_D.sql. Esta consulta define el esquema uao\_eventos, tablas, claves foráneas, tipos ENUM y restricciones. Esta consulta es la base del sistema; y asegura que la estructura refleje el MR/MER final y que la integridad referencial esté garantizada desde el inicio.

Para este punto es necesario primero crear la base de datos a partir de la siguiente orden:

|  |
| --- |
| DROP DATABASE IF EXISTS uao\_eventos;  CREATE DATABASE uao\_eventos CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;  CLEAR |

Luego se procede a crear las tablas como se indica a continuación.

4.1.3.1 Tabla usuario

|  |
| --- |
| Propósito: Personas de la plataforma (docente, estudiante, secretaria).  PK: idUsuario (BIGINT, AI)  Campos principales   * nombre (VARCHAR(120), NOT NULL) * correo (VARCHAR(150), NOT NULL, UNIQUE) * rol (ENUM('docente','estudiante','secretariaAcademica'), NOT NULL)   Relaciones   * 1–N con evento (como organizador). * 1–N con contrasena. * N–N con evento a través de usuarioevento (organizadores/aval). * 1–N como receptor de notificacion.   Importancia: saber quién organiza, quién recibe notificaciones, y segmentar métricas por rol (p.ej., ¿cuántos eventos organizan los docentes vs. estudiantes?). |
| CREATE TABLE usuario (    idUsuario INT PRIMARY KEY AUTO\_INCREMENT,    nombre    VARCHAR(100) NOT NULL,    correo    VARCHAR(120) NOT NULL UNIQUE,    rol       ENUM('docente','estudiante','secretariaAcademica') NOT NULL  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idUsuario | BIGINT AI | No | PK | Identificador del usuario |
| nombre | VARCHAR(120) | No |  | Nombre completo |
| correo | VARCHAR(150) | No | UNIQUE | Correo institucional |
| rol | ENUM('docente','estudiante','secretariaAcademica') | No |  | Perfil dentro del sistema |
|  | | | | |
| Utilidad: segmentar métricas por rol; identificar organizadores y destinatarios de notificaciones | | | | |

4.1.3.2 TABLA contraseña

|  |
| --- |
| Propósito: Estado y hash de la clave del usuario (auditoría básica).  PK: idContrasena  FK: idUsuario: usuario(idUsuario)  Campos   * estado (ENUM('activa','inactiva'), DEFAULT 'activa') * clave/password\_hash (VARCHAR(255)) * fechaCambio (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)   Importancia: controles de seguridad y auditoría ( por ejemplo claves caducadas, usuarios inactivos). |
| CREATE TABLE contrasena (    idUsuario INT PRIMARY KEY,    hash      VARCHAR(255) NOT NULL,    actualizadoEn DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,    CONSTRAINT fk\_contra\_usuario FOREIGN KEY (idUsuario) REFERENCES usuario(idUsuario)  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idContrasena | BIGINT AI | No | PK | Identificador del registro de clave |
| idUsuario | BIGINT | No | FK:usuario | Usuario dueño de la clave |
| clave / password\_hash | VARCHAR(255) | No |  | Hash de la contraseña |
| estado | ENUM('activa','inactiva') | No |  | Estado de la clave |
| fechaCambio | TIMESTAMP | Sí |  | Último cambio |
|  | | | | |
| Utilidad: auditoría de seguridad básica. | | | | |

4.1.3.3 TABLA instalacion

|  |
| --- |
| Propósito: Espacios físicos para eventos.  PK: idInstalacion  Campos   * nombre (VARCHAR(120)) * tipo (ENUM('salon','laboratorio','auditorio','otro')) * capacidad (INT, CHECK (capacidad>0)) * ubicacion (VARCHAR(150))   Relaciones   * N–N con evento por eventoinstalacion.   Importancia: planeación de espacios (ocupación, top de salones usados, topes de capacidad). |
| CREATE TABLE instalacion (    idInstalacion INT PRIMARY KEY AUTO\_INCREMENT,    nombre   VARCHAR(80) NOT NULL,    tipo     ENUM('salon','auditorio','laboratorio','cancha') NOT NULL,    capacidad INT NOT NULL,    ubicacion VARCHAR(120) NOT NULL  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idInstalacion | BIGINT AI | No | PK | Identificador del espacio |
| nombre | VARCHAR(120) | No |  | Nombre del espacio |
| tipo | ENUM('salon','laboratorio','auditorio','otro') | No |  | Clasificación |
| capacidad | INT (CHECK > 0) | No |  | Aforo |
| ubicacion | VARCHAR(150) | No |  | Ubicación física |
|  | | | | |
| Utilidad: planificación de espacios y análisis de ocupación. | | | | |

4.1.3.4 TABLA organización

|  |
| --- |
| Propósito: Organizaciones externas que participan.  PK: idOrganizacion  Campos   * nombre, representanteLegal, actividadPrincipal, telefono, ubicacion, sectorEconomico   Relación   * N–N con evento por eventoorganizacion.   Importancia: alianzas más activas, sectores con mayor participación. |
| CREATE TABLE organizacion (    idOrganizacion INT PRIMARY KEY AUTO\_INCREMENT,    nombre VARCHAR(120) NOT NULL,    nit    VARCHAR(40) ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idOrganizacion | BIGINT AI | No | PK | Identificador |
| nombre | VARCHAR(150) | No |  | Razón social |
| representanteLegal | VARCHAR(120) | No |  | RL |
| actividadPrincipal | VARCHAR(160) | No |  | Objeto social |
| telefono | VARCHAR(40) | No |  | Contacto |
| ubicacion | VARCHAR(150) | No |  | Dirección |
| sectorEconomico | VARCHAR(120) | No |  | Sector |
|  | | | | |
| Utilidad: seguimiento de alianzas externas. | | | | |

4.1.3.5 TABLA evento

|  |
| --- |
| Propósito: Actividades de la gestión (unidad central del dominio).  PK: idEvento  Campos   * nombre (VARCHAR(180)) * descripcion (TEXT, NULL) * fechaInicio, fechaFin (DATETIME, CHECK fechaFin>=fechaInicio) * estado (ENUM según versión: 'registrado','enRevision','aprobado','rechazado' o 'pendiente','aprobado','rechazado') * categoria (ENUM('academico','ludico')) * idOrganizador (FK a usuario) * rutaAvalPDF (VARCHAR(255)) * fechaRegistro (TIMESTAMP, default now)   Relaciones   * 1–N con usuario (un creador). * N–N con instalacion (eventoinstalacion). * N–N con organizacion (eventoorganizacion). * 1–N con usuarioevento (organizadores/aval); la regla de negocio asegura un principal por evento. * 1–N con evaluacion (historial de revisiones).   Importancia: pipeline de eventos, % aprobación/rechazo, tiempos de aprobación, uso por categoría. |
| CREATE TABLE evento (    idEvento INT PRIMARY KEY AUTO\_INCREMENT,    nombre       VARCHAR(180) NOT NULL,    descripcion  VARCHAR(280),    idOrganizador INT NOT NULL,            *-- se mantiene para no romper el POST actual*    idInstalacion INT NOT NULL,    fechaInicio DATETIME NOT NULL,    fechaFin    DATETIME NOT NULL,    categoria   ENUM('academico','ludico') NOT NULL,    estado      ENUM('registrado','enRevision','aprobado','rechazado') NOT NULL DEFAULT 'registrado',    rutaAvalPDF VARCHAR(200),    CONSTRAINT fk\_evento\_organizador FOREIGN KEY (idOrganizador) REFERENCES usuario(idUsuario),    CONSTRAINT fk\_evento\_instalacion  FOREIGN KEY (idInstalacion) REFERENCES instalacion(idInstalacion)  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idEvento | BIGINT AI | No | PK | Identificador del evento |
| nombre | VARCHAR(180) | No |  | Título |
| descripcion | TEXT | Sí |  | Descripción |
| fechaInicio | DATETIME | No |  | Inicio |
| fechaFin | DATETIME | No |  | Fin (CHECK >= fechaInicio) |
| estado | ENUM('registrado','enRevision','aprobado','rechazado') | No |  | Estado del flujo |
| categoria | ENUM('academico','ludico') | No |  | Clasificación |
| idOrganizador | BIGINT | No | FK:usuario | Organizador principal |
| idInstalacion | BIGINT | No | FK:instalacion | Instalación por defecto |
| rutaAvalPDF | VARCHAR(255) | No |  | Evidencia de aval |
| fechaRegistro | TIMESTAMP | Sí |  | Creación |
|  | | | | |
| Utilidad: núcleo del dominio; soporta KPIs (tasa de aprobación, tiempos, categorías). | | | | |

4.1.3.6 TABLA usuarioevento

|  |
| --- |
| Propósito: vincula usuarios con eventos indicando rol dentro del evento.  PK: idUsuarioEvento  FK: idUsuario → usuario, idEvento → evento  Campos   * principal (ENUM('S','N'), default 'N') * tipoAval (ENUM('director\_programa','director\_docencia'), NULL) * avalPDF (VARCHAR(255), NULL)   Reglas (implementadas con constraint/trigger)   * Solo roles docente/estudiante pueden organizar. * Un principal='S' por cada evento.   Importancia: gobernanza (qué perfil impulsa más eventos), seguimiento de avales. |
| CREATE TABLE usuarioEvento (    idUsuarioEvento INT PRIMARY KEY AUTO\_INCREMENT,    idUsuario INT NOT NULL,    idEvento  INT NOT NULL,    principal ENUM('S','N') NOT NULL DEFAULT 'S',    tipoAval  ENUM('director\_programa','director\_docencia'),    avalPDF   VARCHAR(200),    CONSTRAINT fk\_ue\_usuario FOREIGN KEY (idUsuario) REFERENCES usuario(idUsuario),    CONSTRAINT fk\_ue\_evento  FOREIGN KEY (idEvento)  REFERENCES evento(idEvento)  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idUsuarioEvento | BIGINT AI | No | PK | Identificador |
| idUsuario | BIGINT | No | FK:usuario | Usuario vinculado |
| idEvento | BIGINT | No | FK:evento | Evento vinculado |
| principal | ENUM('S','N') | No (default 'N') |  | Si es organizador principal |
| tipoAval | ENUM('director\_programa','director\_docencia') | Sí |  | Tipo de aval |
| avalPDF | VARCHAR(255) | Sí |  | Archivo de aval |
|  | | | | |
| Reglas: Solo docente/estudiante pueden estar aquí (trigger de validación). Un principal='S' por cada idEvento (trigger/validación). | | | | |

4.1.3.7 TABLA eventoinstalacion

|  |
| --- |
| Propósito: asignación de instalaciones a eventos (N–N).  PK: (depende del script: PK compuesto o id propio).  FK: idEvento: evento, idInstalacion: instalacion  importancia: agenda de ocupación y detección de solapamientos. |
| CREATE TABLE eventoInstalacion (    idEvento INT NOT NULL,    idInstalacion INT NOT NULL,    PRIMARY KEY (idEvento, idInstalacion),    CONSTRAINT fk\_ei\_evento      FOREIGN KEY (idEvento)      REFERENCES evento(idEvento),    CONSTRAINT fk\_ei\_instalacion FOREIGN KEY (idInstalacion) REFERENCES instalacion(idInstalacion)  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idEventoInstalacion (o PK compuesta) | BIGINT/PK compuesta | No | PK | Identificador o (idEvento,idInstalacion) |
| idEvento | BIGINT | No | FK:evento | Evento |
| idInstalacion | BIGINT | No | FK:instalacion | Instalación |
|  | | | | |
| Utilidad: asignar múltiples espacios a un evento y detectar solapamientos. | | | | |

4.1.3.8 TABLA eventoorganizacion

|  |
| --- |
| Propósito: organizaciones participantes por evento (N–N).  PK: idEventoOrganizacion  FK: idEvento: evento, idOrganizacion: organizacion  Campos   * certificadoPDF (opcional), participante, esRepresentanteLegal (BOOLEAN)   Importancia: valor de alianzas y trazabilidad legal. |
| CREATE TABLE eventoOrganizacion (    idEvento INT NOT NULL,    idOrganizacion INT NOT NULL,    rol ENUM('patrocinador','aliado','coorganizador') NOT NULL,    certificadoPDF VARCHAR(200),    representanteLegal TINYINT(1) DEFAULT 0,    PRIMARY KEY (idEvento, idOrganizacion),    CONSTRAINT fk\_eo\_evento        FOREIGN KEY (idEvento)        REFERENCES evento(idEvento),    CONSTRAINT fk\_eo\_organizacion  FOREIGN KEY (idOrganizacion)  REFERENCES organizacion(idOrganizacion)  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idEventoOrganizacion | BIGINT AI | No | PK | Identificador |
| idEvento | BIGINT | No | FK:evento | Evento |
| idOrganizacion | BIGINT | No | FK:organizacion | Organización |
| certificadoPDF | VARCHAR(255) | Sí |  | Evidencia |
| participante | VARCHAR(120) | No |  | Nombre del participante |
| esRepresentanteLegal | BOOLEAN | No (default TRUE) |  | Indicador |
|  | | | | |
| Utilidad: trazabilidad con terceros. | | | | |

4.1.3.9 TABLA evaluación

|  |
| --- |
| Propósito: registro de revisión/decisión académica sobre el evento.  PK: idEvaluacion  FK: idEvento → evento *(y si tu script lo incluye, también idSecretariaAcademica)*  Campos   * estado (ENUM('aprobado','rechazado')) * comentarios (TEXT), actaPDF (VARCHAR(255)), fechaRevision (TIMESTAMP)   Regla de negocio típica   * AFTER INSERT / AFTER UPDATE: sincroniza evento.estado y genera una notificacion.   Importancia: auditoría de decisiones; medir tiempos, motivos de rechazo. |
| CREATE TABLE evaluacion (    idEvaluacion INT PRIMARY KEY AUTO\_INCREMENT,    idSecretariaAcademica INT NOT NULL,    idEvento INT NOT NULL,    comentarios VARCHAR(280),    estado ENUM('aprobado','rechazado') NOT NULL,    actaPDF VARCHAR(200),    fechaEvaluacion DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,    CONSTRAINT fk\_ev\_user   FOREIGN KEY (idSecretariaAcademica) REFERENCES usuario(idUsuario),    CONSTRAINT fk\_ev\_evento FOREIGN KEY (idEvento)              REFERENCES evento(idEvento)  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idEvaluacion | BIGINT AI | No | PK | Identificador |
| idEvento | BIGINT | No | FK:evento | Evento evaluado |
| comentarios | TEXT | Sí |  | Observaciones |
| estado | ENUM('aprobado','rechazado') | No |  | Decisión |
| actaPDF | VARCHAR(255) | Sí |  | Acta |
| fechaRevision | TIMESTAMP | Sí |  | Fecha de decisión |
|  | | | | |
| Regla: AFTER INSERT/UPDATE actualiza evento.estado y genera notificacion. | | | | |

4.1.3.10 TABLA notificación

|  |
| --- |
| Propósito: evidencia de la comunicación generada por una evaluación.  PK: idNotificacion  FK: idEvaluacion → evaluacion, usuarioReceptor → usuario  Campos   * tipoNotificacion (ENUM('aprobado','rechazado')), fechaEnvio, justificacion, urlPDF   Importancia: cumplimiento de avisos; SLA de comunicación, quién recibe qué y cuándo. |
| CREATE TABLE notificacion (    idNotificacion INT PRIMARY KEY AUTO\_INCREMENT,    idEvaluacion INT NOT NULL,    idEvento INT NOT NULL,    tipo ENUM('aprobado','rechazado') NOT NULL,    mensaje VARCHAR(220) NOT NULL,    fechaEnvio DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,    CONSTRAINT fk\_not\_ev     FOREIGN KEY (idEvaluacion) REFERENCES evaluacion(idEvaluacion),    CONSTRAINT fk\_not\_evento FOREIGN KEY (idEvento)     REFERENCES evento(idEvento)  ); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Nulo** | **Clave** | **Descripción** |
| idNotificacion | BIGINT AI | No | PK | Identificador |
| idEvaluacion | BIGINT | No | FK:evaluacion | Origen |
| tipoNotificacion | ENUM('aprobado','rechazado') | No |  | Tipo |
| fechaEnvio | TIMESTAMP | Sí |  | Fecha |
| justificacion | TEXT | Sí |  | Texto enviado |
| urlPDF | VARCHAR(255) | Sí |  | Enlace a documento |
| usuarioReceptor | BIGINT | No | FK:usuario | Destinatario |
|  | | | | |
| Utilidad: evidencia de comunicación y verificación de SLA internos. | | | | |

4.1.3.11 Vista v\_eventos\_detalle

Vistas basicas

|  |
| --- |
| Propósito: “todo en una sola vista” para reportes: evento + organizador + instalación (normalmente agregada) + estado/categoría.  Columnas: idEvento, nombre, categoria, estado, organizador, instalaciones\_listadas, fechas, etc.  Utilidad: alimenta paneles (BI) y reportes sin hacer múltiples JOIN manuales. |
| CREATE OR REPLACE VIEW v\_eventos\_detalle AS  SELECT e.idEvento, e.nombre, e.fechaInicio, e.fechaFin, e.categoria, e.estado,         u.idUsuario AS idOrganizador, u.nombre AS organizador, i.nombre AS instalacion  FROM evento e  JOIN usuario u    ON u.idUsuario = e.idOrganizador  JOIN instalacion i ON i.idInstalacion = e.idInstalacion; |

4.1.3.12 Vista v\_eventos\_pendientes

|  |
| --- |
| Propósito: lista de eventos aún pendientes/en revisión.  Utilidad: tablero operativo para la secretaría; prioriza qué evaluar. |
| CREATE OR REPLACE VIEW v\_eventos\_pendientes AS  SELECT \* FROM evento WHERE estado IN ('registrado','enRevision'); |

4.1.3.13 Relaciones Clave

|  |
| --- |
| usuario (1) ──< (N) evento *(creador)*  usuario (N) ──< (N) usuarioevento >── (N) evento *(organizadores/aval – con “un principal”)*  evento (N) ──< (N) eventoinstalacion >── (N) instalacion  evento (N) ──< (N) eventoorganizacion >── (N) organizacion  evento (1) ──< (N) evaluacion ──< (N) notificacion  usuario (1) ──< (N) notificacion *(receptor)* |

4.1.4 Inserción de datos.

En este punto, se creun la consulta “2\_insert\_uao.sql”. Esta consulta permite cargar usuarios @uao.edu.co, instalaciones, organizaciones y eventos iniciales. Además, permite probar el backend end-to-end con datos reales, reproducir escenarios y evidenciar resultados en Swagger/Postman y Workbench.

|  |
| --- |
| Tabla usuarios |
| **INSERT** **INTO** usuario (nombre, correo, rol) **VALUES**  ('Carlos Perez','carlos.perez@uao.edu.co','docente'),  ('Maria Gomez','maria.gomez@uao.edu.co','estudiante'),  ('Juan Lopez','juan.lopez@uao.edu.co','docente'),  ('Laura Moreno','laura.moreno@uao.edu.co','estudiante'),  ('Andres Torres','andres.torres@uao.edu.co','docente'),  ('Paula Ramirez','paula.ramirez@uao.edu.co','estudiante'),  ('Diego Martinez','diego.martinez@uao.edu.co','docente'),  ('Ana Rodriguez','ana.rodriguez@uao.edu.co','estudiante'),  ('Felipe Sosa','felipe.sosa@uao.edu.co','docente'),  ('Natalia Rios','natalia.rios@uao.edu.co','estudiante'),  ('Sergio Vargas','sergio.vargas@uao.edu.co','docente'),  ('Luisa Arias','luisa.arias@uao.edu.co','estudiante'),  ('David Castaño','david.castano@uao.edu.co','docente'),  ('Juliana Silva','juliana.silva@uao.edu.co','estudiante'),  ('Camilo Ortiz','camilo.ortiz@uao.edu.co','docente'),  ('Valentina Paz','valentina.paz@uao.edu.co','estudiante'),  ('Mateo Sanchez','mateo.sanchez@uao.edu.co','docente'),  ('Sofia Herrera','sofia.herrera@uao.edu.co','estudiante'),  ('Javier Niño','javier.nino@uao.edu.co','docente'),  ('Daniela Calle','daniela.calle@uao.edu.co','secretariaAcademica'); |

|  |
| --- |
| TABLA contrasena |
| *-- Contraseñas (hash simbólico)*  **INSERT** **INTO** contrasena (idUsuario, **hash**) **VALUES**  (1,  '$2b$12$hash1'), (2,'$2b$12$hash2'), (3,'$2b$12$hash3'), (4,'$2b$12$hash4'),  (5,  '$2b$12$hash5'), (6,'$2b$12$hash6'), (7,'$2b$12$hash7'), (8,'$2b$12$hash8'),  (9,  '$2b$12$hash9'), (10,'$2b$12$hash10'), (11,'$2b$12$hash11'), (12,'$2b$12$hash12'),  (13, '$2b$12$hash13'), (14,'$2b$12$hash14'), (15,'$2b$12$hash15'), (16,'$2b$12$hash16'),  (17, '$2b$12$hash17'), (18,'$2b$12$hash18'), (19,'$2b$12$hash19'), (20,'$2b$12$hash20'); |

|  |
| --- |
| TABLA instalacion |
| **INSERT** **INTO** instalacion (nombre, tipo, capacidad, ubicacion) **VALUES**  ('Auditorio Principal','auditorio',300,'Bloque A - Piso 1'),  ('Laboratorio 3','laboratorio',35,'Bloque C - Piso 2'),  ('Salón 201','salon',50,'Bloque B - Piso 2'); |

|  |
| --- |
| TABLA organización |
| **INSERT** **INTO** organizacion (nombre, nit) **VALUES**  ('Comunidad AI Cali','900111222'), ('Fundación Talentos','901333444'); |

|  |
| --- |
| TABLA evento |
| *-- Eventos (3 de muestra)*  **INSERT** **INTO** evento (nombre, descripcion, idOrganizador, idInstalacion, fechaInicio, fechaFin, categoria, estado, rutaAvalPDF) **VALUES**  ('Seminario IA UAO','Charlas sobre ML',1,2,'2025-11-05 08:00:00','2025-11-05 12:00:00','academico','registrado','/avales/aval1.pdf'),  ('Festival de Talentos UAO','Música y artes escénicas',2,1,'2025-11-10 18:00:00','2025-11-10 21:30:00','ludico','registrado','/avales/aval\_evento2.pdf'),  ('Jornada de Robótica','Exhibición interfacultades',3,2,'2025-11-15 09:00:00','2025-11-15 17:00:00','academico','enRevision','/avales/aval\_rob1.pdf'); |

|  |
| --- |
| TABLA usuarioEvento |
| *-- Organizador principal en usuarioEvento (regla de “uno solo S”)*  **INSERT** **INTO** usuarioEvento (idUsuario, idEvento, principal) **VALUES**  (1,1,'S'),(2,2,'S'),(3,3,'S'); |

|  |
| --- |
| TABLA eventoInstalacion |
| *-- Instalaciones adicionales del evento (ejemplo)*  **INSERT** **INTO** eventoInstalacion (idEvento, idInstalacion) **VALUES** (1,1),(1,3); |

|  |
| --- |
| TABLA eventoOrganizacion |
| *-- Organizaciones vinculadas al evento (ejemplo)*  **INSERT** **INTO** eventoOrganizacion (idEvento, idOrganizacion, rol, certificadoPDF, representanteLegal) **VALUES**  (1,1,'aliado','/cert/aliado1.pdf',0),  (2,2,'patrocinador','/cert/patro2.pdf',1); |

## 4.1.5 CONSULTAS DE VERIFICACIÓN Y CONTROL

Se creo un grupo de consultas en el archivo “3\_consultas\_control.sql”. Estas consultas incluyen chequeos de calidad, tales como conteos por estado, detección de huérfanos, un “principal” por evento, duplicados y solapamientos de instalaciones. En cierta manera, es el “semáforo” de salud del dato; pues si aquí aparece una alerta, se corrige antes de continuar con pruebas funcionales.

Ahora bien, cuando se habla de “huérfanos”, nos referimos a filas “hijas” que apuntan (por FK) a un registro “padre” que no existe. Por ejemplo,

* Una notificación que apunta a una evaluación inexistente.
* Una evaluación que apunta a un evento que ya fue borrado.
* Un usuarioEvento que referencia un usuario o evento que no están.

Esto es importante poque, los registros huérfanos pueden causar lo siguiente:

* Rompen la **coherencia** del dato (lo que ves no es confiable).
* Pueden causar **errores** en el backend (joins que fallan) y en análisis (indicadores inflados o vacíos).
* indican problemas en el **flujo de borrado** o en la **carga de datos**.

Por otra parte, los solapamientos se presentan cuando dos (o más) eventos reservados en la misma instalación con rangos de tiempo que se cruzan.

Criterio clásico de solapamiento entre A y B:

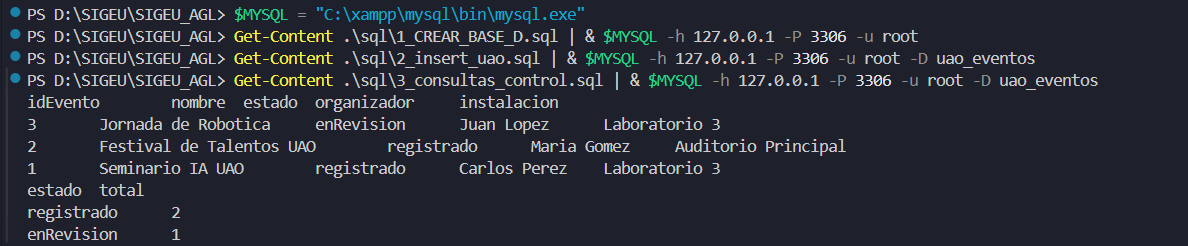
* A.inicio < B.fin AND B.inicio < A.fin

Este tipo de consulta son importantes porque un solapamiento puede:

* Genera conflictos logísticos (dos actividades en el mismo salón/auditorio).
* Impacta la experiencia de asistentes y la operación del evento
* Afecta indicadores de ocupación real vs. planificada.

Al detectar esto, se puede:

* Reprogramar el evento en otro horario o mudar de instalación.
* Detectar patrones de saturación (qué salas chocan más y cuándo), lo cual sirve como insumo para planeación de infraestructura, distribución de eventos y políticas de reserva.



4.1.5.1

4.1.5.2

4.1.5.3

4.1.5.4

4.1.5.5

## 4.2 CONSULTAS AVANZADAS PARA TOMA DE DESCIONES

Estas consultas, generan indicadores operativos y estratégicos tales como: tasas de aprobación/rechazo, tiempos de revisión por secretaría, uso de instalaciones, horas pico, anticipación de solicitudes, top de organizadores/organizaciones, etc. Su importancia, radica en que estas métricas guían decisiones de planificación (calendario, asignación de espacios y personal), mejora de procesos (SLA) y gestión de aliados externos

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

## 4.3 OBJETOS ALMACENADOS

Texto

El contenido generado por IA puede ser incorrecto.