Sistema de Gestión de Eventos Universitarios

Sistema de Gestión de Eventos Universitarios

Ultima entrega del proyecto de Almacenamiento de Datos

Omar Yepez Ospino
Laura Fernanda Piragauta Pinzón
Jose Gonzalías Millán
Ana María Rivera
Jackeline Ramírez Sánchez

Presentado al maestro Jhon Éder Masso

Universidad Autónoma de Occidente
Almacenamiento de Datos
Santiago de Cali
Octubre de 2025

Tabla de contenido

Modelado de la base de datos 4

Script completo en SQL 6

Base de datos inicial empleada para la elaboración y prueba de consultas 9

Desarrollo del backend de la API REST 11

Evidencia de las pruebas realizadas en Postman 12

Definición de las 10 consultas con su definición clara 27

1. Eventos con su creador, unidad académica y horarios 27

2. Unidades académicas que colaboraron juntas en más de un evento 27

3. Reservas con usuario y evento 28

4. Historial de cambios con responsable y reserva 28

5. Cantidad de reservas por usuario 29

6. Reservas con historial y detalle del usuario responsable de la última modificación 29

7. Eventos con su porcentaje de reservas activas vs totales 30

8. Participación estudiantil por facultad 30

9. Usuarios que hicieron reserva en un evento, pero NO en otros eventos 31

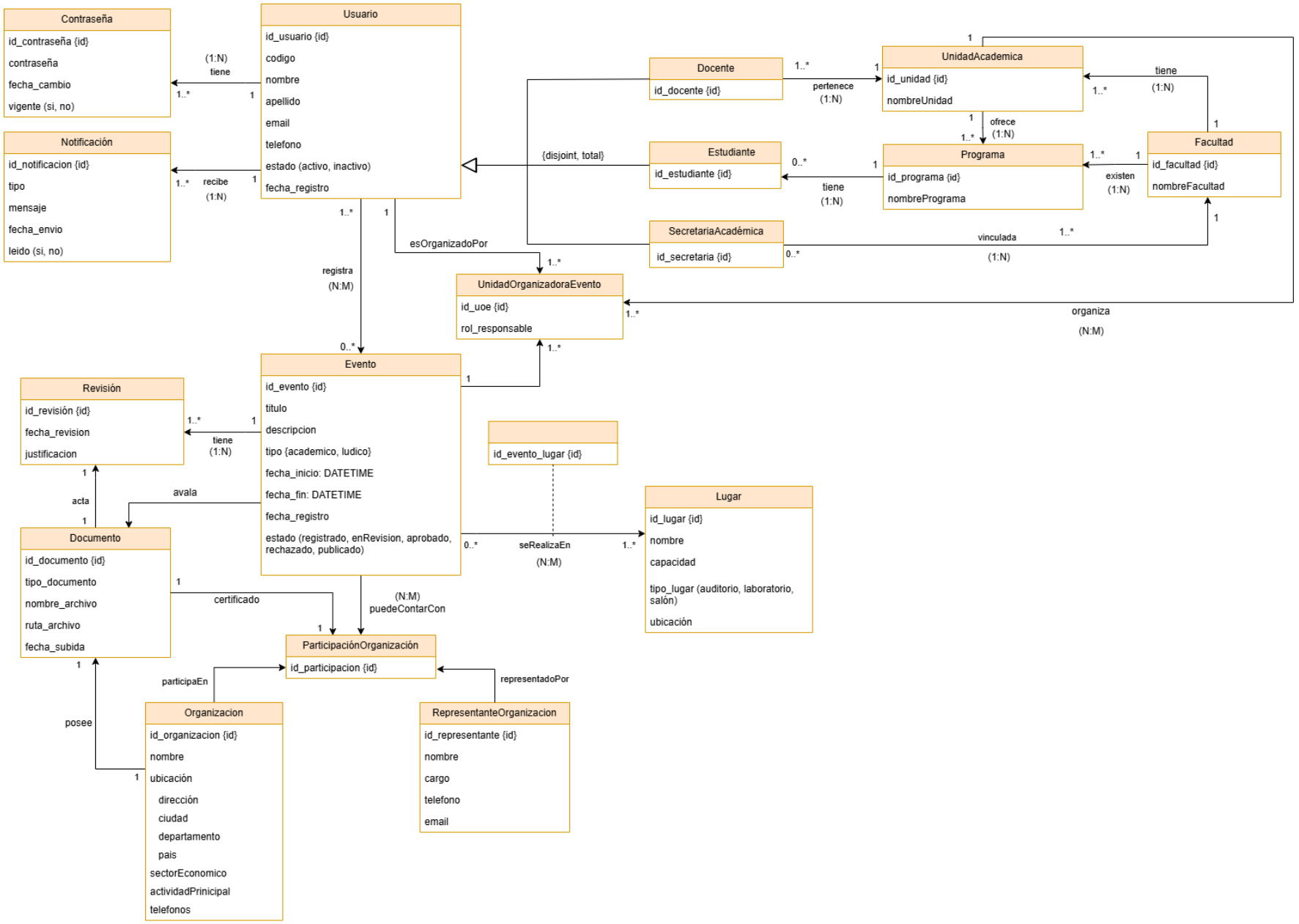
10. Ranking de unidades académicas según impacto 31

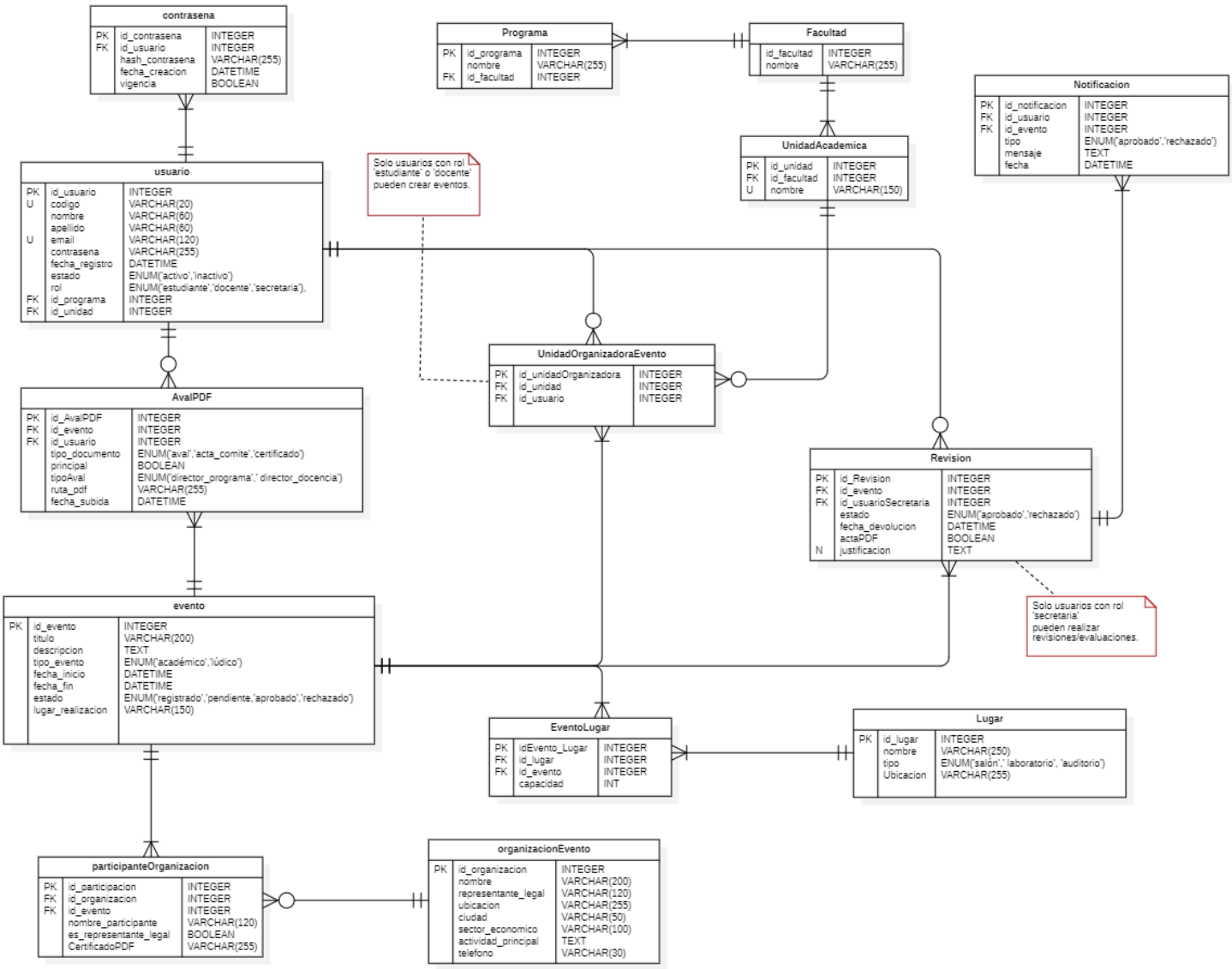
Objetos almacenados en MySQL 33

Modelado de la base de datos de documento 34

Modelado de la base de datos

Sistema de Gestión de Eventos Universitarios (SIGEU)




contrasena
PK id_contrasena INTEGER
FK id_usuario INTEGER
hash_contrasena VARCHAR(255)
fecha_creacion DATETIME
vigencia BOOLEAN
Programa
PK id_programa INTEGER
nombre VARCHAR(255)
FK id_facultad INTEGER
Facultad
id_facultad INTEGER
nombre VARCHAR(255)
Notificacion
PK id_notificacion INTEGER
FK id_usuario INTEGER
FK id_evento INTEGER
tipo ENUM('aprobado','rechazado')
mensaje TEXT
fecha DATETIME
UnidadAcademica
PK id_unidad INTEGER
FK id_facultad INTEGER
U nombre VARCHAR(150)
usuario
PK id_usuario INTEGER
U codigo VARCHAR(20)
nombre VARCHAR(60)
apellido VARCHAR(60)
U email VARCHAR(120)
contrasena VARCHAR(255)
fecha_registro DATETIME
estado ENUM('activo','inactivo')
rol ENUM('estudiante','docente','secretaria').
FK id_programa INTEGER
FK id_unidad INTEGER
Solo usuarios con rol 'estudiante' o 'docente' pueden crear eventos.
UnidadOrganizadoraEvento
PK id_unidadOrganizadora INTEGER
FK id_unidad INTEGER
FK id_usuario INTEGER
AvalPDF
PK id_AvalPDF INTEGER
FK id_evento INTEGER
FK id_usuario INTEGER
tipo_documento ENUM('aval','acta_comite','certificado')
principal BOOLEAN
tipoAval ENUM('director_programa',' director_docencia')
ruta_pdf VARCHAR(255)
fecha_subida DATETIME
Revision
PK id_Revision INTEGER
FK id_evento INTEGER
FK id_usuarioSecretaria INTEGER
estado ENUM('aprobado','rechazado')
fecha_devolucion DATETIME
actaPDF BOOLEAN
N justificacion TEXT
Solo usuarios con rol 'secretaria' pueden realizar revisiones/evaluaciones.
evento
PK id_evento INTEGER
titulo VARCHAR(200)
descripcion TEXT
tipo_evento ENUM('académico','lúdico')
fecha_inicio DATETIME
fecha_fin DATETIME
estado ENUM('registrado','pendiente,'aprobado','rechazado')
lugar_realizacion VARCHAR(150)
EventoLugar
PK idEvento_Lugar INTEGER
FK id_lugar INTEGER
FK id_evento INTEGER
capacidad INT
Lugar
PK id_lugar INTEGER
nombre VARCHAR(250)
tipo ENUM('salón', 'laboratorio', 'auditorio')
Ubicacion VARCHAR(255)
participanteOrganizacion
PK id_participacion INTEGER
FK id_organizacion INTEGER
FK id_evento INTEGER
nombre_participante VARCHAR(120)
es_representante_legal BOOLEAN
CertificadoPDF VARCHAR(255)
organizacionEvento
PK id_organizacion INTEGER
nombre VARCHAR(200)
representante_legal VARCHAR(120)
ubicacion VARCHAR(255)
ciudad VARCHAR(50)
sector_economico VARCHAR(100)
actividad_principal TEXT
telefono VARCHAR(30)

Script completo en SQL

```
CREATE DATABASE IF NOT EXISTS gestion_eventos;
USE gestion_eventos;
CREATE TABLE facultad (
id_facultad BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100) NOT NULL UNIQUE
);
CREATE TABLE unidad_academica (
id_unidad_academica BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100) NOT NULL UNIQUE,
id_facultad BIGINT UNSIGNED NOT NULL,
FOREIGN KEY (id_facultad) REFERENCES facultad(id_facultad)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE usuario (
id_usuario BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(60) NOT NULL,
apellido VARCHAR(60) NOT NULL,
correo VARCHAR(120) NOT NULL UNIQUE,
rol ENUM('Docente','Estudiante','Secretario','Administrador') NOT NULL,
estado ENUM('Activo','Inactivo') NOT NULL DEFAULT 'Activo'
);
CREATE TABLE tipo_evento (
id_tipo BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(60) NOT NULL UNIQUE
);
CREATE TABLE evento (
id_evento BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100) NOT NULL,
tipo VARCHAR(60) NOT NULL,
fecha_inicio DATE NOT NULL,
fecha_fin DATE NOT NULL,
id_usuario BIGINT UNSIGNED NOT NULL,
id_unidad_academica BIGINT UNSIGNED NOT NULL,
FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (id_unidad_academica) REFERENCES
unidad_academica(id_unidad_academica)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE evaluacion (
id_evaluacion BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
id_evento BIGINT UNSIGNED NOT NULL,
```

```
estado ENUM('Pendiente','Aprobado','Rechazado') NOT NULL,
justificacion TEXT,
FOREIGN KEY (id_evento) REFERENCES evento(id_evento)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE organizacion (
id_organizacion BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(120) NOT NULL UNIQUE,
representante_legal VARCHAR(120) NOT NULL,
telefono VARCHAR(30),
ubicacion VARCHAR(120)
);
CREATE TABLE participacion (
id_participacion BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
id_evento BIGINT UNSIGNED NOT NULL,
id_organizacion BIGINT UNSIGNED NOT NULL,
participante VARCHAR(120) NOT NULL,
es_representante_legal BOOLEAN NOT NULL DEFAULT FALSE,
FOREIGN KEY (id_evento) REFERENCES evento(id_evento)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (id_organizacion) REFERENCES organizacion(id_organizacion)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE historial_contrasenas (
id_historial BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
id_usuario BIGINT UNSIGNED NOT NULL,
contrasena VARCHAR(255) NOT NULL,
vigente BOOLEAN NOT NULL DEFAULT FALSE,
FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE notificacion (
id_notificacion BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
id_usuario BIGINT UNSIGNED NOT NULL,
mensaje VARCHAR(255) NOT NULL,
leida BOOLEAN NOT NULL DEFAULT FALSE,
FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE horario (
id_horario BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
id_evento BIGINT UNSIGNED NOT NULL,
hora_inicio TIME NOT NULL,
hora_fin TIME NOT NULL,
```

```
FOREIGN KEY (id_evento) REFERENCES evento(id_evento)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE reserva (
id_reserva BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
id_usuario BIGINT UNSIGNED NOT NULL,
id_evento BIGINT UNSIGNED NOT NULL,
fecha_reserva DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
estado ENUM('Activa','Cancelada') NOT NULL DEFAULT 'Activa',
FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (id_evento) REFERENCES evento(id_evento)
ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE historial (
id_historial BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
id_reserva BIGINT UNSIGNED NOT NULL,
accion VARCHAR(150) NOT NULL,
fecha_cambio DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
usuario_responsable BIGINT UNSIGNED NOT NULL,
FOREIGN KEY (id_reserva) REFERENCES reserva(id_reserva)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (usuario_responsable) REFERENCES usuario(id_usuario)
ON UPDATE CASCADE ON DELETE CASCADE
);
```

Base de datos inicial empleada para la elaboración y prueba de consultas

```
INSERT INTO facultad (nombre) VALUES
('Facultad de Ingeniería'),
('Facultad de Humanidades'),
('Facultad de Ciencias Económicas'),
('Facultad de Arquitectura y Diseño'),
('Facultad de Comunicación'),
('Facultad de Ciencias de la Salud'),
('Facultad de Derecho');

INSERT INTO unidad_academica (nombre, id_facultad) VALUES
('Ciencias de la Computación', 1),
('Ingeniería Industrial', 1),
('Comunicación Social', 5),
('Administración de Empresas', 3),
('Diseño Gráfico', 4),
('Psicología', 6),
('Derecho Público', 7);

INSERT INTO usuario (nombre, apellido, correo, rol, estado) VALUES
('Lorena', 'Martínez', 'lorena.martinez@uao.edu.co', 'Estudiante', 'Activo'),
('Samuel', 'Murillo', 'samuel.murillo@uao.edu.co', 'Docente', 'Activo'),
('Sofía', 'Hernández', 'sofia.hernandez@uao.edu.co', 'Administrador', 'Activo'),
('David', 'Rojas', 'david.rojas@uao.edu.co', 'Estudiante', 'Inactivo'),
('Vanesa', 'Cruz', 'vanesa.cruz@uao.edu.co', 'Docente', 'Activo'),
('Carlos', 'Mejía', 'carlos.mejia@uao.edu.co', 'Docente', 'Activo'),
('Laura', 'Gómez', 'laura.gomez@uao.edu.co', 'Estudiante', 'Activo');

INSERT INTO tipo_evento (nombre) VALUES
('Académico'),
('Cultural'),
('Deportivo'),
('Social');

INSERT INTO evento (nombre, id_tipo, fecha_inicio, fecha_fin, id_usuario,
id_unidad_academica) VALUES
('Seminario Big Data', 1, '2025-10-05', '2025-10-06', 1, 1),
('Foro de Innovación', 1, '2025-09-20', '2025-09-22', 2, 1),
('Festival de Teatro', 2, '2025-08-10', '2025-08-12', 3, 3),
('Torneo de Voleibol', 3, '2025-07-15', '2025-07-17', 4, 2),
('Simposio de Psicología', 1, '2025-05-10', '2025-05-12', 5, 6),
('Jornada de Derecho Público', 4, '2025-04-03', '2025-04-04', 6, 7),
('Concurso de Diseño', 2, '2025-03-15', '2025-03-18', 7, 5);
```

```
INSERT INTO evaluacion (id_evento, estado, justificacion) VALUES
(3, 'Aprobado', 'Cumple requisitos'),
(4, 'Pendiente', NULL),
(6, 'Rechazado', 'Documentación incompleta');

INSERT INTO reserva (id_usuario, id_evento) VALUES
(1, 3),
(3, 4),
(4, 6);
```

Aquí se encuentra de manera más organizada.
También se encuentra el Postman y los objetos almacenados.
https://github.com/LPiragauta26/Almacenamiento-de-datos---Script-base-de-datos-objetos-almacenados-y-Postman.git

Desarrollo del backend de la API REST

Para el desarrollo del backend se implementó una API REST con el framework FastAPI, utilizando el enfoque asincrónico con SQLAlchemy Async para la conexión a la base de datos MySQL.

La API permite realizar las operaciones CRUD (crear, consultar, actualizar y eliminar) sobre los eventos registrados en el sistema institucional de gestión de eventos (SIGEU).

Se definieron modelos de datos basados en el archivo base_datos_final.sql, junto con sus correspondientes esquemas Pydantic para validación, servicios para la lógica de negocio, y rutas que exponen los endpoints principales.

Además, se configuró un archivo .env para proteger las credenciales de conexión y un archivo requirements.txt con las dependencias necesarias para ejecutar el proyecto.

El backend fue probado mediante Postman, verificando la funcionalidad de cada endpoint y la correcta interacción con la base de datos.

Toda la estructura del proyecto y los archivos fuente se encuentran disponibles en el enlace compartido junto con esta entrega.

Aquí está el backend. https://github.com/LPiragauta26/Proyecto-almacenamiento-de-datos-backend.git

Evidencia de las pruebas realizadas en Postman


```
{
  "info": {
    "_postman_id": "dd6a8a7a-29c4-4c8c-bf4c-5dbec7c62b3e",
    "name": "Backend FastAPI",
    "description": "Colección de pruebas del backend",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
    "_exporter_id": "49064500",
    "_collection_link": "https://laura-fer-piragauta-
9800274.postman.co/workspace/LAURA-FERNANDA-PIRAGAUTA-PINZON~639addf0-
050d-4855-8603-8552433b6875/collection/49064500-dd6a8a7a-29c4-4c8c-bf4c-
5dbec7c62b3e?action=share&source=collection_link&creator=49064500"
  },
  "item": [
    {
      "name": "Usuarios",
      "item": [
        {
          "name": "Listar usuarios",
          "protocolProfileBehavior": {
            "disableBodyPruning": true
          },
          "request": {
            "auth": {
              "type": "noauth"
            },
            "method": "GET",
            "header": [],
            "body": {
              "mode": "raw",
              "raw": "{\r\n  \"nombre\": \"Julia\",\r\n  \"apellido\":
\"Gómez\",\r\n  \"correo\": \"laura@uao.edu.co\",\r\n  \"rol\": \"Estudiante\",\r\n  \"estado\":
\"Activo\"\r\n}",
              "options": {
                "raw": {
                  "language": "json"
                }
              }
            },
            "url": {
              "raw": "http://127.0.0.1:8000/usuarios/usuarios/",
              "protocol": "http",
              "host": [
```

```
                    "127",
                    "0",
                    "0",
                    "1"
                ],
                "port": "8000",
                "path": [
                    "usuarios",
                    "usuarios",
                    ""
                ]
            }
        },
        "response": []
    },
    {
        "name": "Crear usuarios",
        "request": {
            "auth": {
                "type": "noauth"
            },
            "method": "POST",
            "header": [],
            "body": {
                "mode": "raw",
                "raw": "{\r\n  \"nombre\": \"Vanesa\",\r\n  \"apellido\":
\"Lopez\",\r\n  \"correo\": \"vaneza.lopez@uao.edu.co\",\r\n  \"rol\":
\"Estudiante\",\r\n  \"estado\": \"Activo\"\r\n}",
                "options": {
                    "raw": {
                        "language": "json"
                    }
                }
            },
            "url": {
                "raw": "http://127.0.0.1:8000/usuarios/usuarios/",
                "protocol": "http",
                "host": [
                    "127",
                    "0",
                    "0",
                    "1"
                ],
                "port": "8000",
```

```
            "path": [
              "usuarios",
              "usuarios",
              ""
            ]
          }
        },
        "response": []
      },
      {
        "name": "Actualizar usuario",
        "request": {
          "auth": {
            "type": "noauth"
          },
          "method": "PUT",
          "header": [],
          "body": {
            "mode": "raw",
            "raw": "{\r\n  \"nombre\": \"Laura Actualizada\",\r\n  \"apellido\":
\"Gomez\",\r\n  \"correo\": \"laura_updated@example.com\",\r\n  \"rol\":
\"Docente\",\r\n  \"estado\": \"Activo\"\r\n}",
            "options": {
              "raw": {
                "language": "json"
              }
            }
          },
          "url": {
            "raw": "http://127.0.0.1:8000/usuarios/usuarios/8",
            "protocol": "http",
            "host": [
              "127",
              "0",
              "0",
              "1"
            ],
            "port": "8000",
            "path": [
              "usuarios",
              "usuarios",
              "8"
            ]
          }
```

```
    },
    "response": []
},
{
    "name": "Eliminar usuarios",
    "request": {
        "auth": {
            "type": "noauth"
        },
        "method": "DELETE",
        "header": [],
        "url": {
            "raw": "http://127.0.0.1:8000/usuarios/usuarios/2",
            "protocol": "http",
            "host": [
                "127",
                "0",
                "0",
                "1"
            ],
            "port": "8000",
            "path": [
                "usuarios",
                "usuarios",
                "2"
            ]
        }
    },
    "response": []
},
{
    "name": "Obtener usuarios",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw": "http://127.0.0.1:8000/usuarios/usuarios/3",
            "protocol": "http",
            "host": [
                "127",
                "0",
                "0",
                "1"
            ],
```

```
                "port": "8000",
                "path": [
                  "usuarios",
                  "usuarios",
                  "3"
                ]
              }
            },
            "response": []
          }
        ]
      },
      {
        "name": "Eventos",
        "item": [
          {
            "name": "Crear Evento",
            "request": {
              "method": "POST",
              "header": [],
              "body": {
                "mode": "raw",
                "raw": "{\r\n  \"nombre\": \"Conferencia de Ciencia\",\r\n  \"id_tipo\":
1,\r\n  \"fecha_inicio\": \"2025-11-20\",\r\n  \"fecha_fin\": \"2025-11-20\",\r\n  \"id_usuario\":
3,\r\n  \"id_unidad_academica\": 2\r\n}",
                "options": {
                  "raw": {
                    "language": "json"
                  }
                }
              },
              "url": {
                "raw": "http://127.0.0.1:8000/eventos/eventos/",
                "protocol": "http",
                "host": [
                  "127",
                  "0",
                  "0",
                  "1"
                ],
                "port": "8000",
                "path": [
                  "eventos",
                  "eventos",
```

```
                ""
              ]
            }
          },
          "response": []
        },
        {
          "name": "Listar Eventos",
          "request": {
            "method": "GET",
            "header": [],
            "url": {
              "raw": "http://127.0.0.1:8000/eventos/eventos/",
              "protocol": "http",
              "host": [
                "127",
                "0",
                "0",
                "1"
              ],
              "port": "8000",
              "path": [
                "eventos",
                "eventos",
                ""
              ]
            }
          },
          "response": []
        },
        {
          "name": "Actualizar Evento",
          "request": {
            "method": "PUT",
            "header": [],
            "body": {
              "mode": "raw",
              "raw": "{\r\n  \"nombre\": \"Torneo de Voleibol\",\r\n  \"id_tipo\":
3,\r\n  \"fecha_inicio\": \"2025-07-19\",\r\n  \"fecha_fin\": \"2025-07-20\",\r\n  \"id_usuario\":
4,\r\n  \"id_unidad_academica\": 2\r\n}\r\n",
              "options": {
                "raw": {
                  "language": "json"
                }
```

```
            }
        },
        "url": {
            "raw": "http://127.0.0.1:8000/eventos/eventos/4",
            "protocol": "http",
            "host": [
                "127",
                "0",
                "0",
                "1"
            ],
            "port": "8000",
            "path": [
                "eventos",
                "eventos",
                "4"
            ]
        }
    },
    "response": []
},
{
    "name": "Eliminar Evento",
    "request": {
        "method": "DELETE",
        "header": [],
        "url": {
            "raw": "http://127.0.0.1:8000/eventos/eventos/1",
            "protocol": "http",
            "host": [
                "127",
                "0",
                "0",
                "1"
            ],
            "port": "8000",
            "path": [
                "eventos",
                "eventos",
                "1"
            ]
        }
    },
    "response": []
```

```
        },
        {
            "name": "Obtener eventos",
            "request": {
                "method": "GET",
                "header": [],
                "url": {
                    "raw": "http://127.0.0.1:8000/eventos/eventos/4",
                    "protocol": "http",
                    "host": [
                        "127",
                        "0",
                        "0",
                        "1"
                    ],
                    "port": "8000",
                    "path": [
                        "eventos",
                        "eventos",
                        "4"
                    ]
                }
            },
            "response": []
        }
    ]
},
{
    "name": "Reservas",
    "item": [
        {
            "name": "Listar reserva",
            "request": {
                "method": "GET",
                "header": [],
                "url": {
                    "raw": "http://127.0.0.1:8000/reservas/reservas/",
                    "protocol": "http",
                    "host": [
                        "127",
                        "0",
                        "0",
                        "1"
                    ],
```

```
            "port": "8000",
            "path": [
                "reservas",
                "reservas",
                ""
            ]
        }
    },
    "response": []
},
{
    "name": "Crear reservas",
    "request": {
        "method": "POST",
        "header": [],
        "body": {
            "mode": "raw",
            "raw": "{\r\n  \"id_usuario\": 1,\r\n  \"id_evento\": 7\r\n}",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw": "http://127.0.0.1:8000/reservas/reservas/",
            "protocol": "http",
            "host": [
                "127",
                "0",
                "0",
                "1"
            ],
            "port": "8000",
            "path": [
                "reservas",
                "reservas",
                ""
            ]
        }
    },
    "response": []
},
{
```

```
    "name": "Obtener reserva",
    "request": {
      "method": "GET",
      "header": [],
      "url": {
        "raw": "http://127.0.0.1:8000/reservas/reservas/7",
        "protocol": "http",
        "host": [
          "127",
          "0",
          "0",
          "1"
        ],
        "port": "8000",
        "path": [
          "reservas",
          "reservas",
          "7"
        ]
      }
    },
    "response": []
  },
  {
    "name": "Eliminar reserva",
    "request": {
      "method": "DELETE",
      "header": [],
      "body": {
        "mode": "raw",
        "raw": "{\r\n  \"id_usuario\": 3,\r\n  \"id_evento\": 8\r\n}",
        "options": {
          "raw": {
            "language": "json"
          }
        }
      },
      "url": {
        "raw": "http://127.0.0.1:8000/reservas/reservas/1",
        "protocol": "http",
        "host": [
          "127",
          "0",
          "0",
```

```
            "1"
          ],
          "port": "8000",
          "path": [
            "reservas",
            "reservas",
            "1"
          ]
        }
      },
      "response": []
    },
    {
      "name": "Actualizar reserva",
      "request": {
        "method": "PUT",
        "header": [],
        "body": {
          "mode": "raw",
          "raw": "{\r\n  \"id_usuario\": 3,\r\n  \"id_evento\": 8\r\n}",
          "options": {
            "raw": {
              "language": "json"
            }
          }
        },
        "url": {
          "raw": "http://127.0.0.1:8000/reservas/reservas/1",
          "protocol": "http",
          "host": [
            "127",
            "0",
            "0",
            "1"
          ],
          "port": "8000",
          "path": [
            "reservas",
            "reservas",
            "1"
          ]
        }
      },
      "response": []
```

```
        }
      ]
    },
    {
      "name": "Evaluaciones",
      "item": [
        {
          "name": "Listar evaluaciones",
          "request": {
            "method": "GET",
            "header": [],
            "url": {
              "raw": "http://127.0.0.1:8000/evaluaciones/evaluaciones/",
              "protocol": "http",
              "host": [
                "127",
                "0",
                "0",
                "1"
              ],
              "port": "8000",
              "path": [
                "evaluaciones",
                "evaluaciones",
                ""
              ]
            }
          },
          "response": []
        },
        {
          "name": "Crear evaluaciones",
          "request": {
            "method": "POST",
            "header": [],
            "body": {
              "mode": "raw",
              "raw": "{\r\n  \"id_evento\": 7,\r\n  \"estado\":
\"Pendiente\",\r\n  \"justificacion\": \"El evento aún no ha sido revisado\"\r\n}\r\n",
              "options": {
                "raw": {
                  "language": "json"
                }
              }
```

```
      },
      "url": {
        "raw": "http://127.0.0.1:8000/evaluaciones/evaluaciones/",
        "protocol": "http",
        "host": [
          "127",
          "0",
          "0",
          "1"
        ],
        "port": "8000",
        "path": [
          "evaluaciones",
          "evaluaciones",
          ""
        ]
      }
    },
    "response": []
  },
  {
    "name": "Obtener evaluaciones",
    "request": {
      "method": "GET",
      "header": [],
      "url": {
        "raw": "http://127.0.0.1:8000/evaluaciones/evaluaciones/4",
        "protocol": "http",
        "host": [
          "127",
          "0",
          "0",
          "1"
        ],
        "port": "8000",
        "path": [
          "evaluaciones",
          "evaluaciones",
          "4"
        ]
      }
    },
    "response": []
  },
```

```
        {
            "name": "Actualizar evaluaciones",
            "request": {
                "method": "PUT",
                "header": [],
                "body": {
                    "mode": "raw",
                    "raw": "{\r\n  \"id_evento\": 3,\r\n  \"estado\":
\"Aprobado\",\r\n  \"justificacion\": \"Cumple requisitos\"\r\n}\r\n",
                    "options": {
                        "raw": {
                            "language": "json"
                        }
                    }
                },
                "url": {
                    "raw": "http://127.0.0.1:8000/evaluaciones/evaluaciones/4",
                    "protocol": "http",
                    "host": [
                        "127",
                        "0",
                        "0",
                        "1"
                    ],
                    "port": "8000",
                    "path": [
                        "evaluaciones",
                        "evaluaciones",
                        "4"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "Eliminar evaluaciones",
            "request": {
                "method": "DELETE",
                "header": [],
                "url": {
                    "raw": "http://127.0.0.1:8000/evaluaciones/evaluaciones/5",
                    "protocol": "http",
                    "host": [
                        "127",
```

```
                    "0",
                    "0",
                    "1"
                ],
                "port": "8000",
                "path": [
                    "evaluaciones",
                    "evaluaciones",
                    "5"
                ]
            }
        },
        "response": []
    }
  ]
}
```

Estos fueron las pruebas que se realizaron y todas funcionaron.

Definición de las 10 consultas con su definición clara

1. Eventos con su creador, unidad académica y horarios

```
SELECT e.id_evento, e.nombre AS evento, e.tipo,
       e.fecha_inicio, e.fecha_fin,
       CONCAT(u.nombre, ' ', u.apellido) AS creador,
       ua.nombre AS unidad_academica,
       h.hora_inicio, h.hora_fin
FROM evento e
JOIN usuario u ON e.id_usuario = u.id_usuario
JOIN unidad_academica ua ON e.id_unidad_academica = ua.id_unidad_academica
LEFT JOIN horario h ON e.id_evento = h.id_evento
ORDER BY e.id_evento, h.hora_inicio;
```

Definición: Recupera información detallada de los eventos, incluyendo quién los creó, a qué unidad académica pertenecen y los horarios asociados.

Propósito: Permite visualizar de manera completa los eventos programados, facilitando el control de responsables y horarios.

id_evento	evento	tipo	fecha_inicio	fecha_fin	creador	unidad_academica	hora_inicio	hora_fin
1	Seminario Big Data	Académico	2025-10-05	2025-10-06	Lorena Martínez	Ciencias de la Computación	08:00:00	10:00:00
1	Seminario Big Data	Académico	2025-10-05	2025-10-06	Lorena Martínez	Ciencias de la Computación	10:30:00	12:00:00
2	Foro de Innovación	Académico	2025-09-20	2025-09-22	Samuel Murillo	Ciencias de la Computación	09:00:00	11:00:00
3	Festival de Teatro	Cultural	2025-08-10	2025-08-12	Sofía Hernández	Comunicación Social	14:00:00	16:00:00
4	Torneo de Voleibol	Deportivo	2025-07-15	2025-07-17	David Rojas	Ingeniería Industrial	NULL	NULL
5	Simposio de Psicología	Académico	2025-05-10	2025-05-12	Vanesa Cruz	Psicología	NULL	NULL
6	Jornada de Derecho Público	Social	2025-04-03	2025-04-04	Carlos Mejía	Derecho Público	NULL	NULL
7	Concurso de Diseño	Cultural	2025-03-15	2025-03-18	Laura Gómez	Diseño Gráfico	NULL	NULL

2. Unidades académicas que colaboraron juntas en más de un evento

```
SELECT
   ua1.nombre AS unidad_1,
   ua2.nombre AS unidad_2,
   COUNT(*) AS coincidencias
FROM evento e1
JOIN evento e2 ON e1.id_evento < e2.id_evento
JOIN unidad_academica ua1 ON ua1.id_unidad_academica = e1.id_unidad_academica
JOIN unidad_academica ua2 ON ua2.id_unidad_academica = e2.id_unidad_academica
GROUP BY ua1.id_unidad_academica, ua2.id_unidad_academica
HAVING COUNT(*) > 1;
```

Definición: Identifica pares de unidades académicas que han participado juntas en más de un evento.

Propósito: Permite analizar la colaboración entre unidades académicas y detectar relaciones recurrentes en la organización de eventos.

unidad_1	unidad_2	coincidencias
Ciencias de la Computación	Comunicación Social	2
Ciencias de la Computación	Ingeniería Industrial	2
Ciencias de la Computación	Psicología	2
Ciencias de la Computación	Derecho Público	2
Ciencias de la Computación	Diseño Gráfico	2

3. Reservas con usuario y evento

```
SELECT DISTINCT
    r.id_reserva,
    CONCAT(u.nombre, ' ', u.apellido) AS usuario,
    e.nombre AS evento,
    r.fecha_reserva,
    r.estado
FROM reserva r
JOIN usuario u ON r.id_usuario = u.id_usuario
JOIN evento e ON r.id_evento = e.id_evento
ORDER BY r.id_reserva;
```

Definición: Muestra todas las reservas realizadas, indicando qué usuario hizo la reserva y a qué evento corresponde.

Propósito: Facilita el seguimiento de las reservas y el control de asistencia a los eventos.

id_reserva	usuario	evento	fecha_reserva	estado
1	Lorena Martínez	Seminario Big Data	2025-11-08 13:00:31	Activa
2	Samuel Murillo	Seminario Big Data	2025-11-08 13:00:31	Activa
3	Sofía Hernández	Foro de Innovación	2025-11-08 13:00:31	Activa
4	David Rojas	Festival de Teatro	2025-11-08 13:00:31	Cancelada
5	Vanesa Cruz	Torneo de Voleibol	2025-11-08 13:00:31	Activa
6	Carlos Mejía	Simposio de Psicología	2025-11-08 13:00:31	Activa
7	Laura Gómez	Jornada de Derecho Público	2025-11-08 13:00:31	Cancelada

4. Historial de cambios con responsable y reserva

```
SELECT h.id_historial,
    h.id_reserva,
    h.accion,
    h.fecha_cambio,
    CONCAT(u.nombre, ' ', u.apellido) AS responsable
FROM historial h
JOIN usuario u ON h.usuario_responsable = u.id_usuario
ORDER BY h.fecha_cambio DESC;
```

Definición: Recupera el historial de acciones realizadas sobre las reservas, indicando quién hizo cada modificación y cuándo.

Propósito: Permite auditar cambios y responsabilidades sobre reservas, importante para control administrativo y seguimiento.

id_historial	id_reserva	accion	fecha_cambio	responsable
1	1	Reserva creada	2025-11-08 13:00:31	Lorena Martínez
2	1	Reserva modificada	2025-11-08 13:00:31	Sofía Hernández
3	2	Reserva cancelada	2025-11-08 13:00:31	Samuel Murillo
4	3	Reserva creada	2025-11-08 13:00:31	Sofía Hernández
5	4	Reserva cancelada	2025-11-08 13:00:31	David Rojas
6	5	Reserva creada	2025-11-08 13:00:31	Vanesa Cruz

5. Cantidad de reservas por usuario

```
SELECT CONCAT(u.nombre, ' ', u.apellido) AS usuario,
      COUNT(r.id_reserva) AS total_reservas
FROM usuario u
LEFT JOIN reserva r ON u.id_usuario = r.id_usuario
GROUP BY u.id_usuario
ORDER BY total_reservas DESC;
```

Definición: Cuenta cuántas reservas ha hecho cada usuario.

Propósito: Permite identificar usuarios activos o frecuentes en el sistema, útil para estadísticas o incentivos.

usuario	total_reservas
Lorena Martínez	1
Samuel Murillo	1
Sofía Hernández	1
David Rojas	1
Vanesa Cruz	1
Carlos Mejía	1
Laura Gómez	1

6. Reservas con historial y detalle del usuario responsable de la última modificación

```
WITH ultimos AS (
   SELECT
      h.id_reserva,
      h.accion,
      h.usuario_responsable,
      ROW_NUMBER() OVER (PARTITION BY h.id_reserva ORDER BY h.fecha_cambio
DESC) AS rn
   FROM historial h
)
SELECT
   r.id_reserva,
   CONCAT(u.nombre, ' ', u.apellido) AS usuario_duenio_reserva,
   e.nombre AS evento,
   ult.accion AS ultima_accion,
   CONCAT(ur.nombre, ' ', ur.apellido) AS responsable_ultima_accion
FROM reserva r
JOIN usuario u ON r.id_usuario = u.id_usuario
```

```
JOIN evento e ON r.id_evento = e.id_evento
JOIN ultimos ult ON r.id_reserva = ult.id_reserva AND ult.rn = 1
JOIN usuario ur ON ult.usuario_responsable = ur.id_usuario;
```

Definición: Muestra la última acción realizada sobre cada reserva y quién fue el responsable.

Propósito: Permite auditar las modificaciones recientes sobre reservas, destacando el control de cambios y la responsabilidad administrativa.

id_reserva	usuario_duenio_reserva	evento	ultima_accion	responsable_ultima_accion
1	Lorena Martínez	Seminario Big Data	Reserva creada	Lorena Martínez
2	Samuel Murillo	Seminario Big Data	Reserva cancelada	Samuel Murillo
3	Sofía Hernández	Foro de Innovación	Reserva creada	Sofía Hernández
4	David Rojas	Festival de Teatro	Reserva cancelada	David Rojas
5	Vanesa Cruz	Torneo de Voleibol	Reserva creada	Vanesa Cruz

7. Eventos con su porcentaje de reservas activas vs totales

```
SELECT
    e.nombre AS evento,
    COUNT(r.id_reserva) AS total_reservas,
    SUM(r.estado = 'Activa') AS reservas_activas,
    ROUND(
        (SUM(r.estado = 'Activa') / COUNT(r.id_reserva)) * 100, 2
    ) AS porcentaje_activo
FROM evento e
LEFT JOIN reserva r ON e.id_evento = r.id_evento
GROUP BY e.id_evento;
```

Definición: Calcula cuántas reservas están activas en comparación con el total de reservas para cada evento.

Propósito: Permite evaluar el nivel de participación real y la efectividad de la gestión de reservas de cada evento.

evento	total_reservas	reservas_activas	porcentaje_activo
Seminario Big Data	2	2	100.00
Foro de Innovación	1	1	100.00
Festival de Teatro	1	0	0.00
Torneo de Voleibol	1	1	100.00
Simposio de Psicología	1	1	100.00
Jornada de Derecho Público	1	0	0.00
Concurso de Diseño	0	NULL	NULL

8. Participación estudiantil por facultad

```
SELECT
    f.nombre AS facultad,
    COUNT(e.id_evento) AS total_eventos_estudiantiles
FROM evento e
JOIN usuario u ON u.id_usuario = e.id_usuario
JOIN unidad_academica ua ON ua.id_unidad_academica = e.id_unidad_academica
```

```
JOIN facultad f ON f.id_facultad = ua.id_facultad
WHERE u.rol = 'Estudiante'
GROUP BY f.id_facultad;
```

Definición: Cuenta el número de eventos creados por estudiantes de cada facultad.

Propósito: Analiza la participación estudiantil por facultad, útil para medir compromiso y protagonismo en la organización de actividades.

facultad	total_eventos_estudiantiles
Facultad de Ingeniería	2
Facultad de Arquitectura y Diseño	1

9. Usuarios que hicieron reserva en un evento, pero NO en otros eventos

```
SELECT DISTINCT
   CONCAT(u.nombre, ' ', u.apellido) AS usuario
FROM usuario u
JOIN reserva r1 ON u.id_usuario = r1.id_usuario
WHERE r1.id_evento = 1
AND u.id_usuario NOT IN (
   SELECT r2.id_usuario
   FROM reserva r2
   WHERE r2.id_evento <> 1
);
```

Definición: Identifica usuarios que participaron exclusivamente en un evento específico.

Propósito: Permite detectar usuarios con participación selectiva, útil para análisis de fidelización o segmentación de asistentes.

usuario
Lorena Martínez
Samuel Murillo

10. Ranking de unidades académicas según impacto

```
SELECT
   ua.nombre AS unidad,
   (
      (SELECT COUNT(*)
       FROM evaluacion ev
       JOIN evento e2 ON e2.id_evento = ev.id_evento
```

```
        WHERE ev.estado = 'Aprobado'
          AND e2.id_unidad_academica = ua.id_unidad_academica)
        +
        (SELECT COUNT(*)
         FROM evento e3
         JOIN participacion p ON p.id_evento = e3.id_evento
         WHERE e3.id_unidad_academica = ua.id_unidad_academica)
    ) AS puntaje_total
FROM unidad_academica ua
ORDER BY puntaje_total DESC;
```

Definición: Calcula un puntaje total para cada unidad académica según la cantidad de evaluaciones aprobadas y participaciones en eventos.

Propósito: Permite generar un ranking de impacto institucional, destacando las unidades académicas más activas y efectivas en la gestión de eventos.

unidad	puntaje_total
Ciencias de la Computación	4
Ingeniería Industrial	2
Psicología	2
Comunicación Social	1
Derecho Público	1
Administración de Empresas	0
Diseño Gráfico	0

Objetos almacenados en MySQL

En esta sección se desarrollaron los objetos almacenados del sistema SIGEU (Sistema de Gestión de Eventos Universitarios), implementados sobre la base de datos MySQL. Estos objetos fueron diseñados para mejorar la eficiencia, la integridad de los datos y la automatización de procesos dentro del sistema.

Los objetos almacenados incluyen:

- Funciones: permiten realizar cálculos o retornar valores específicos para facilitar el análisis de información, como la cantidad de eventos aprobados o la duración promedio de estos.

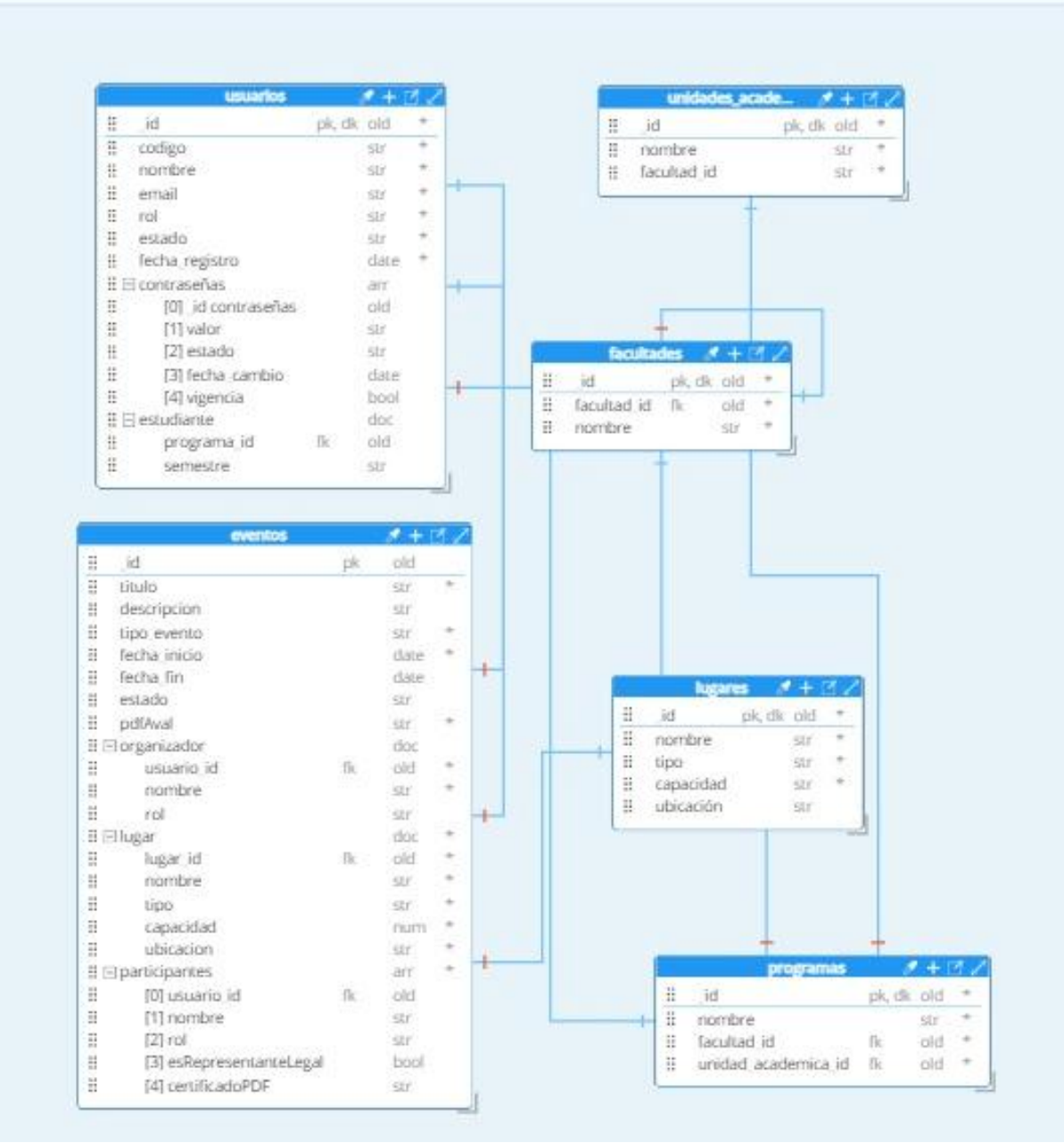
- Vistas: fueron creadas para simplificar consultas complejas y permitir el acceso rápido a información combinada de varias tablas, como el detalle de eventos con sus unidades académicas, responsables y estado actual.

- Triggers: se implementaron para garantizar la integridad de los datos y automatizar acciones, por ejemplo, actualizar el estado de un evento cuando se inserta una nueva revisión o evaluación.

Todos los scripts correspondientes a estos objetos se encuentran disponibles en el siguiente repositorio: https://github.com/LPiragauta26/Almacenamiento-de-datos---Script-base-de-datos-objetos-almacenados-y-Postman.git

Modelado de la base de datos de documento

La base de datos del sistema SIGEU fue diseñada previamente en Hackolade, donde se construyó el modelo conceptual y lógico. Una vez finalizado el modelado, el siguiente paso fue llevar la estructura a MongoDB.



Para esto se utilizó MongoDB Compass, desde cuyo *Shell* se ejecutó el script:

sigeu_mongo.txt

Validación de Datos Insertados en las Colecciones

Luego de cargar los datos de prueba y ejecutar las funciones de validación, se verificó la cantidad de documentos insertados en cada colección. Para esto se usó un ciclo en JavaScript dentro de Mongo Shell, el cual imprimió el total de registros por colección.

- Código utilizado para la verificación

```
>_MONGOSH
> use sigeu_mongo
< already on db sigeu_mongo
> [
  'eventos',
  'facultades',
  'lugares',
  'programas',
  'revisiones',
  'unidad_academicas',
  'usuarios' ].forEach(c => {
  print(c, ':', db.getCollection(c).countDocuments());
  });
< eventos
< :
< 4
< facultades
< :
< 6
< lugares
< :
< 7
< programas
< :
< 7
< revisiones
< :
< 4
< unidad_academicas
< :
< 0
< usuarios
```

- Código extra opcional para validar si alguna colección está vacía.

```
> [
    'eventos',
    'facultades',
    'lugares',
    'programas',
    'revisiones',
    'unidad_academicas',
    'usuarios'
  ].forEach(c => {
    let count = db.getCollection(c).countDocuments();
    if (count === 0) {
      print("[ADVERTENCIA] La colección", c, "no contiene registros.");
    } else {
      print("[OK]", c, "tiene", count, "documentos.");
    }
  });
< [OK]
< eventos
< tiene
< 4
< documentos.
< documentos.
< [OK]
< programas
< tiene
< 7
< documentos.
< [OK]
< revisiones
< tiene
< 4
< documentos.
< [ADVERTENCIA] La colección
```

Triggers / Funciones de SIGEU en MongoDB Atlas

Todas las funciones se cargan mediante:

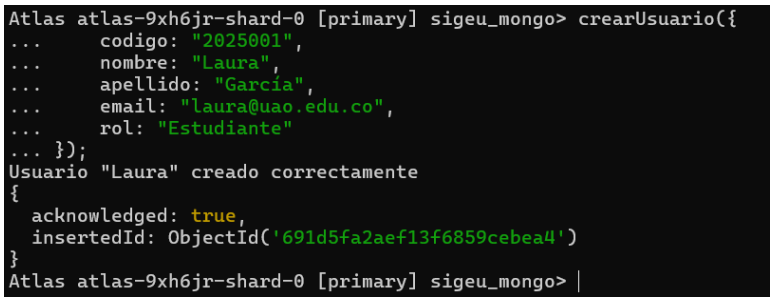
load("C:/Users/Laura/Downloads/Proyecto final almacenamiento de datos/sigeu_triggers.js")

1. crearUsuario()

Simula un trigger BEFORE INSERT de SQL, se debe validar que el correo sea del dominio @uao.edu.co. Para que funcione, también se debe verificar que el rol esté entre los permitidos: Docente, Estudiante, Secretario, Administrador.

Código:

```
crearUsuario({
   codigo: "2025001",
   nombre: "Laura",
   apellido: "García",
   email: "laura@uao.edu.co",
   rol: "Estudiante"
});
```



2. AsignarUsuarioAPrograma()

Solo permite asignar usuarios con rol "Estudiante" a un programa y verifica que tanto el usuario como el programa existan.
Código:
asignarUsuarioAPrograma("ID_DEL_USUARIO", "ID_DEL_PROGRAMA");
asignarUsuarioAPrograma("691d5fa2aef13f6859cebea4","691bd008d6d0970b26b627e5")

```
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo> asignarUsuarioAPrograma("691d5fa2aef13f6859cebea4", "691bccbbac48543cd94ac443")
Usuario Laura asignado al programa Ingeniería de Sistemas
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo>
```

3. registrarOrganizadorEvento()

Solo usuarios con rol "Docente" o "Estudiante" pueden organizar un evento. Se actualiza el documento del evento agregando el campo organizador con la información del usuario.
Código:
registrarOrganizadorEvento("ID_DEL_EVENTO", "ID_DEL_USUARIO");
registrarOrganizadorEvento("691bd009d6d0970b26b627ea","691bd008d6d0970b26b627e1")

```
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo> registrarOrganizadorEvento("691bd009d6d0970b26b627ea","691bd008d6d0970b26b627e1")
Organizador Juan Pablo Castillo registrado en evento undefined
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo>
```

4. evaluarEvento()

Solo usuarios con rol "Secretario" pueden evaluar un evento. Se verifica que el secretario pertenezca a la misma facultad que el evento (si los datos de facultad están disponibles).
Código:
evaluarEvento("ID_DEL_EVENTO", "ID_DEL_SECRETARIO", "Aprobado", "Todo correcto");
Esto sucede cuando no hay secretario:

```
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo> evaluarEvento("691bd009d6d0970b26b627ea","691bd008d6d0970b26b627e3","Aprobado","Todo correcto")
Error: Solo usuarios con rol 'Secretario' pueden evaluar eventos.
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo>
```

Esto sucede cuando si hay secretaria:

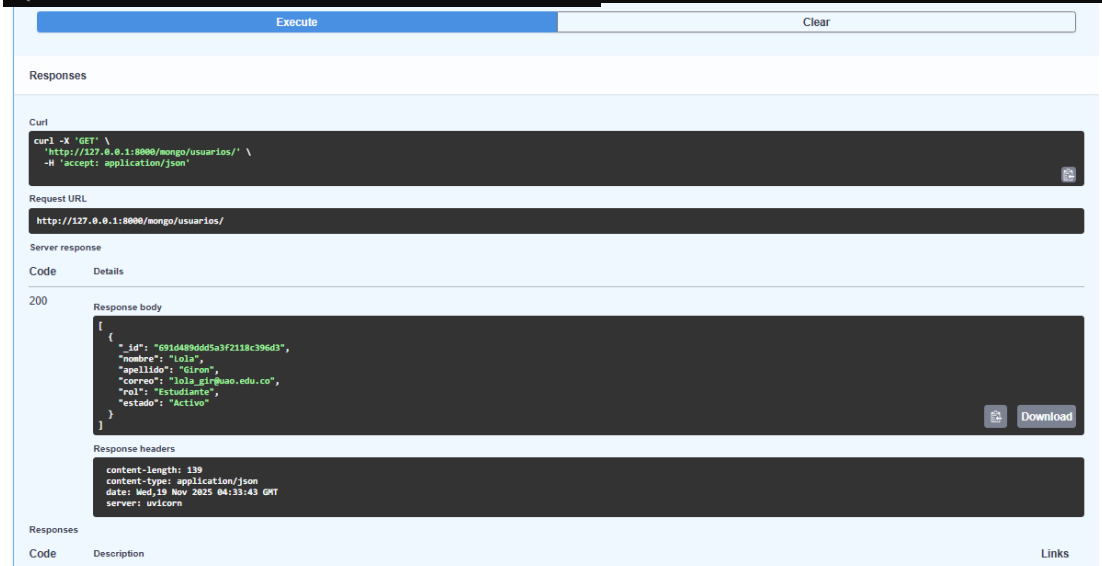
```
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo> evaluarEvento(
...   "691bd009d6d0970b26b627ea",  // ID del evento
...   "691d66dbaef13f6859cebea5",  // ID del secretario (Ana Pérez)
...   "Aprobado",
...   "Todo correcto"
... )
...
Evaluación registrada: evento undefined, estado Aprobado
{ ok: true }
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo>
```

Consejos antes de usar:
Antes de llamar las funciones, verifica los _id de los documentos:
db.usuarios.find().pretty()
db.programas.find().pretty()
db.eventos.find().pretty()

```
Atlas atlas-9xh6jr-shard-0 [primary] sigeu_mongo> db.usuarios.find().pretty()
... db.programas.find().pretty()
... db.eventos.find().pretty()
...
[
  {
    _id: ObjectId('691bcdb7d6d0970b26b627d4'),
    titulo: 'Semana de la Ingeniería',
    descripcion: 'Charlas, talleres y actividades académicas.',
    tipo_evento: 'académico',
    fecha_inicio: ISODate('2025-03-10T00:00:00.000Z'),
    fecha_fin: ISODate('2025-03-15T00:00:00.000Z'),
    estado: 'aprobado',
    pdfAval: 'aval_semana_ing.pdf',
    Organizador: {
      usuario_id: ObjectId('691bcdb6d6d0970b26b627d1'),
      nombre: 'Carlos Romero',
      rol: 'docente'
    },
    lugar: {
      lugar_id: ObjectId('691bcdb6d6d0970b26b627cd'),
      nombre: 'Auditorio Central',
      tipo: 'auditorio',
      capacidad: 200,
      ubicacion: 'Bloque A'
    },
    Participantes: [
      {
        usuario_id: ObjectId('691bcdb6d6d0970b26b627cf'),
        nombre: 'Juan Pérez',
        esRepresentanteLegal: false,
        certificadoPDF: 'cert_jp.pdf'
      }
    ]
  },
  {
    _id: ObjectId('691bd009d6d0970b26b627e9'),
    titulo: 'Semana de la Ingeniería',
    descripcion: 'Charlas, talleres y actividades académicas.',
    tipo_evento: 'académico',
    fecha_inicio: ISODate('2025-03-10T00:00:00.000Z'),
    fecha_fin: ISODate('2025-03-15T00:00:00.000Z'),
    estado: 'aprobado',
    pdfAval: 'aval_semana_ing.pdf',
    Organizador: {
      usuario_id: ObjectId('691bd008d6d0970b26b627e4'),
      nombre: 'Carlos Romero',
      rol: 'docente'
    },
    lugar: {
      lugar_id: ObjectId('691bd008d6d0970b26b627de'),
      nombre: 'Auditorio Principal',
      tipo: 'auditorio',
      capacidad: 200,
      ubicacion: 'Bloque A'
    },
    Participantes: [
      {
        usuario_id: ObjectId('691bd008d6d0970b26b627e1'),
        nombre: 'Juan Pablo Castillo',
        esRepresentanteLegal: false,
        certificadoPDF: 'cert_jp_semanaIng.pdf'
      },
      {
        usuario_id: ObjectId('691bd008d6d0970b26b627e2'),
        nombre: 'Laura Fernández',
        esRepresentanteLegal: true,
        certificadoPDF: 'cert_lf_semanaIng.pdf'
      }
    ]
  },
  {
```

```
  {
    _id: ObjectId('691bd009d6d0970b26b627ea'),
    titulo: 'Taller de Introducción a IA',
    descripcion: 'Sesión práctica de modelos de IA para estudiantes.',
    tipo_evento: 'académico',
    fecha_inicio: ISODate('2025-04-05T00:00:00.000Z'),
    fecha_fin: ISODate('2025-04-05T00:00:00.000Z'),
    estado: 'pendiente',
    pdfAval: 'aval_taller_ia.pdf',
    Organizador: {
      usuario_id: ObjectId('691bd008d6d0970b26b627e1'),
      nombre: 'Juan Pablo Castillo',
      rol: 'estudiante'
    },
    lugar: {
      lugar_id: ObjectId('691bd008d6d0970b26b627df'),
      nombre: 'Salón 301',
      tipo: 'salón',
      capacidad: 40,
      ubicacion: 'Bloque C'
    },
    Participantes: [
      {
        usuario_id: ObjectId('691bd008d6d0970b26b627e1'),
        nombre: 'Juan Pablo Castillo',
        esRepresentanteLegal: true,
        certificadoPDF: null
      },
      {
        usuario_id: ObjectId('691bd008d6d0970b26b627e3'),
        nombre: 'Andrés Rivas',
        esRepresentanteLegal: false,
        certificadoPDF: null
      }
    ]
  },
  {
    _id: ObjectId('691bd009d6d0970b26b627eb'),
    titulo: 'Campeonato Relámpago de Microfútbol',
    descripcion: 'Actividad lúdica para toda la comunidad universitaria.',
    tipo_evento: 'lúdico',
    fecha_inicio: ISODate('2025-05-20T00:00:00.000Z'),
    fecha_fin: ISODate('2025-05-20T00:00:00.000Z'),
    estado: 'rechazado',
    pdfAval: 'aval_microfutbol.pdf',
    Organizador: {
      usuario_id: ObjectId('691bd008d6d0970b26b627e2'),
      nombre: 'Laura Fernández',
      rol: 'estudiante'
    },
    lugar: {
      lugar_id: ObjectId('691bd008d6d0970b26b627e0'),
      nombre: 'Cancha Múltiple',
      tipo: 'cancha',
      capacidad: 100,
      ubicacion: 'Zona deportiva'
    },
    Participantes: [
      {
        usuario_id: ObjectId('691bd008d6d0970b26b627e2'),
        nombre: 'Laura Fernández',
        esRepresentanteLegal: true,
        certificadoPDF: null
      },
      {
        usuario_id: ObjectId('691bd008d6d0970b26b627e3'),
        nombre: 'Andrés Rivas',
        esRepresentanteLegal: false,
        certificadoPDF: null
      }
    ]
  }
```

Api del mongo funcional



Eventos Mongo

GET `/mongo/eventos/` Listar Eventos

Parameters | Cancel

No parameters

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/mongo/eventos/' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/mongo/eventos/
```

Server response

Code	Details
200	Response body

```
[
  {
    "_id": "691d492add5a3f2118c396d4",
    "nombre": "Semana de la Investigación",
    "id_tipo": 1,
    "fecha_inicio": "2025-11-20",
    "fecha_fin": "2025-11-22",
    "id_usuario": "674f41a90ffd2f40fd8a1b23",
    "id_unidad_academica": "UA-01"
  }
]
```

Download

Response headers

```
content-length: 210
content-type: application/json
date: Wed,19 Nov 2025 04:36:12 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type: application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

POST `/mongo/eventos/` Crear Evento

Parameters | Cancel | Reset

No parameters

Request body required | application/json

Edit Value | Schema

```
{
  "nombre": "Semana de la Investigación",
  "id_tipo": 1,
  "fecha_inicio": "2025-11-20",
  "fecha_fin": "2025-11-22",
  "id_usuario": "674f41a90ffd2f40fd8a1b23",
  "id_unidad_academica": "UA-01"
}
```

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/mongo/eventos/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "nombre": "Semana de la Investigación",
  "id_tipo": 1,
  "fecha_inicio": "2025-11-20",
  "fecha_fin": "2025-11-22",
  "id_usuario": "674f41a90ffd2f40fd8a1b23",
  "id_unidad_academica": "UA-01"
}'
```

Request URL

```
http://127.0.0.1:8000/mongo/eventos/
```

Server response

Code	Details
200	Response body

```
{
  "id": "691d492add5a3f2118c396d4"
}
```

Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 33
content-type: application/json
date: Wed,19 Nov 2025 04:35:53 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type: application/json

Controls Accept header.