Universidad Cristóbal Colón

Arquitectura Orientada a Servicios (SOA)

Análisis y diseño de la base de datos y arquitectura unificada basada en servicios SOAP y REST

Proyecto: Actividad de clase 21/11/2025

Nombre de la alumna: Isis Villanueva
Carrera: Ingeniería en Sistemas Computacionales
Profesor: Mtro. Leonardo

Fecha: 21 de noviembre de 2025

1. Introducción

La presente sección documenta el análisis y diseño de la base de datos, así como la arquitectura unificada de servicios propuesta para la plataforma académica basada en SOA. El objetivo es integrar distintos módulos académicos (alumnos, calificaciones y matrículas) mediante servicios REST y SOAP que se comunican con un mismo esquema de datos en MySQL.

2. Contexto del sistema

La plataforma tiene como finalidad centralizar los servicios académicos de una universidad, permitiendo a estudiantes, profesores y administradores consultar y administrar información desde una interfaz web unificada. Para esta primera fase se modelan tres procesos clave:

- Gestión de alumnos.
- Registro de calificaciones.
- Administración de matrículas por periodo y estatus.

Cada uno de estos procesos se implementa como un servicio independiente, pero todos comparten la misma base de datos MySQL para garantizar la consistencia de la información.

3. Análisis y diseño de la base de datos

El modelo lógico de la base de datos se construyó alrededor de tres entidades principales: ALUMNOS, CALIFICACIONES y MATRICULAS. A continuación se describe cada tabla y sus relaciones.

3.1 Tabla ALUMNOS

Campos propuestos:

- id (INT, PK, AUTO_INCREMENT): Identificador único del alumno.
- nombre (VARCHAR): Nombre completo del alumno.
- email (VARCHAR, único): Correo institucional.
- carrera (VARCHAR): Programa académico al que pertenece.

Esta tabla es la base para relacionar calificaciones y matrículas.

3.2 Tabla CALIFICACIONES

Campos propuestos:

- id (INT, PK, AUTO_INCREMENT): Identificador de la calificación.
- alumno_id (INT, FK → ALUMNOS.id): Alumno al que pertenece la calificación.
- materia (VARCHAR): Nombre de la asignatura.
- calificacion (DECIMAL): Calificación numérica obtenida.
- fecha (DATE): Fecha de registro.

Relación: Un alumno puede tener muchas calificaciones (relación 1 : N entre ALUMNOS y CALIFICACIONES).

3.3 Tabla MATRICULAS

Campos propuestos:
• id (INT, PK, AUTO_INCREMENT): Identificador de la matrícula.
• alumno_id (INT, FK → ALUMNOS.id): Alumno al que corresponde la matrícula.
• periodo (VARCHAR): Periodo académico (por ejemplo, '2025-1').
• estatus (VARCHAR): Estado de la matrícula (Activo, Inactivo, Baja, etc.).

Relación: Un alumno puede contar con varias matrículas a lo largo del tiempo (relación 1 : N entre ALUMNOS y MATRICULAS).

3.4 Resumen del modelo entidad–relación

El modelo entidad–relación resultante puede resumirse de la siguiente forma:
• Entidades: ALUMNO, CALIFICACION, MATRICULA.
• Relación ALUMNO–CALIFICACION: 1 : N, mediante la clave foránea alumno_id.
• Relación ALUMNO–MATRICULA: 1 : N, mediante la clave foránea alumno_id.

Este diseño facilita futuras ampliaciones (por ejemplo, pagos, grupos o historial académico) sin modificar la estructura básica de los servicios existentes.

4. Diseño de la arquitectura unificada basada en servicios SOAP y REST

La arquitectura propuesta sigue un enfoque de microservicios ligeros, donde cada módulo académico expone sus operaciones a través de un tipo de servicio específico, pero todos comparten el mismo modelo de datos. La unificación se consigue mediante una capa de orquestación y un cliente común (por ejemplo, una interfaz web o colecciones de POSTMAN).

4.1 Capas de la arquitectura

La solución se organiza en tres capas principales:
• Capa de presentación: Clientes web o herramientas como POSTMAN que consumen las APIs.
• Capa de servicios: Conformada por la API REST en Java y el servicio SOAP en Python.
• Capa de datos: Base de datos MySQL compartida (schema soa_universidad).

4.2 Servicio REST en Java (Spring Boot)

El módulo REST se implementa con Spring Boot y expone endpoints basados en JSON para la gestión de alumnos y calificaciones. Sus características clave son:

• Controladores REST: AlumnoController y CalificacionController.
• Repositorios: AlumnoRepository y CalificacionRepository basados en Spring Data JPA.
• Mapeo objeto–relacional hacia las tablas ALUMNOS y CALIFICACIONES.
• Uso de HTTP verbs estándar (GET, POST, PUT, DELETE) y códigos de respuesta adecuados.

4.3 Servicio SOAP en Python (Spyne)

El módulo SOAP se desarrolla en Python utilizando el framework Spyne, publicando el servicio MatriculaService a través de un WsgiApplication. Características principales:
• Operaciones getAllMatriculas, getMatricula y createMatricula.
• Intercambio de mensajes en formato XML a través de envelopes SOAP 1.1.
• Conexión directa a la tabla MATRICULAS en la base de datos MySQL.
• Validación de tipos a través de las definiciones de Spyne (Integer, Unicode, Iterable).

4.4 Interoperabilidad y arquitectura unificada

Aunque cada servicio utiliza protocolos y formatos distintos (JSON/HTTP REST frente a XML/SOAP), se consideran parte de una misma arquitectura unificada porque:
• Comparten el mismo modelo de datos y reglas de negocio.
• Pueden ser consumidos desde la misma aplicación cliente.
• Permiten evolucionar gradualmente hacia una arquitectura de microservicios completa, donde nuevos módulos pueden implementarse en el lenguaje y estilo más conveniente.

En un futuro, ambos servicios podrían exponerse detrás de un API Gateway o un bus de servicios, facilitando la autenticación, el balanceo de carga y el monitoreo centralizado.

5. Flujo de interacción entre componentes

De forma resumida, el flujo de interacción en la arquitectura propuesta es el siguiente:
1) El cliente (por ejemplo, POSTMAN o una aplicación web) envía una solicitud a la API REST o al servicio SOAP según el tipo de operación que requiera.
2) El servicio correspondiente procesa la petición, aplica las reglas de negocio y se comunica con la base de datos MySQL.
3) MySQL devuelve los datos solicitados o confirma la operación de inserción/actualización.
4) El servicio construye la respuesta en el formato adecuado (JSON o XML) y la regresa al cliente.

Este patrón asegura un bajo acoplamiento entre cliente y base de datos, concentrando la lógica de acceso a datos en los servicios y facilitando la evolución futura del sistema.

6. Conclusiones

El análisis y diseño realizados permiten contar con una base sólida para la integración de servicios académicos bajo una arquitectura orientada a servicios. El modelo de datos propuesto es simple pero extensible, y la coexistencia de una API REST en Java con un servicio SOAP en Python demuestra la interoperabilidad entre tecnologías siempre que se comparta un modelo de información bien definido.

Este enfoque facilita la ampliación de la plataforma con nuevos módulos (pagos, historial, grupos, etc.) y abre la puerta a la creación de un frontend unificado para estudiantes, profesores y administradores.