INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ZAMORA

Maestría en Sistemas Computacionales

Materia: Arquitectura Orientada a Servicios

Maestro: Roberto *Suárez Zinzun*

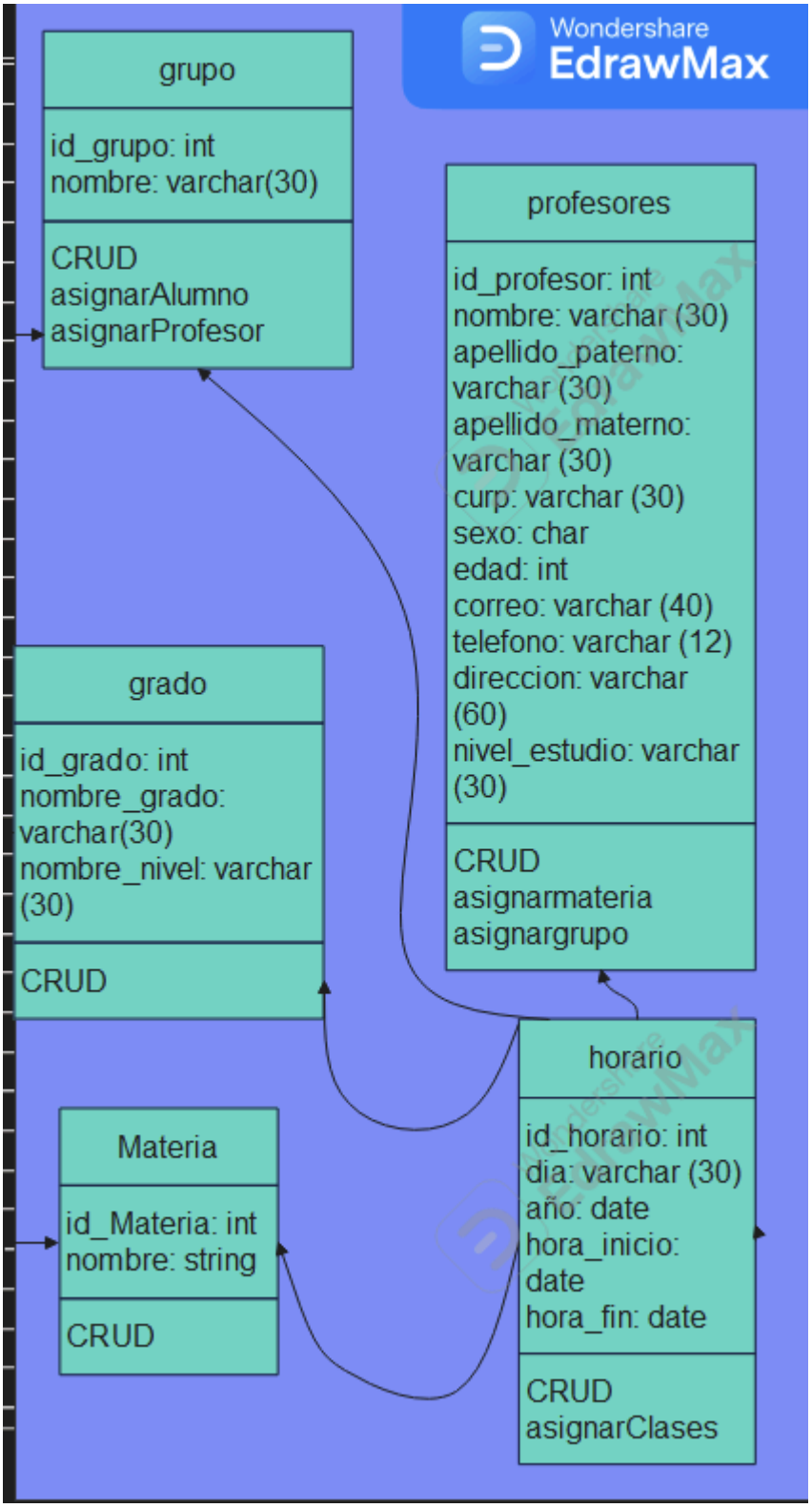
ACTIVIDAD 2: Definición de servicios del Proyecto

Alumno:

Jesús Martín Sámano López

Zamora, Mich. a 12 de Septiembre de 2025

1.- Diagrama del Contexto delimitado:



2.- Nombre del recurso:

ServicioHorario

API REST HORARIO

3.- Tipo de Servicio:

Este servicio se clasifica como de **Entidad** porque está directamente asociado a la tabla **Horario** dentro del modelo de datos, la cual representa una entidad central que relaciona a profesor, materia, grupo y grado. Su función principal es administrar los registros de horarios de forma CRUD (Crear, Consultar, Actualizar, Eliminar).

4.- Operaciones expuestas:

API REST HORARIO

- **Crear horario**
- **Consultar horario**
- **Consultar horario por ID**
- **Consultar horario por grupo**
- **Consultar horario por Profesor**
- **Consultar horario por Grado**
- **Actualizar horario**
- **Eliminar horario**
- **Asignar clases**

5.- Descripción de las operaciones.

Crear Horario

Elemento	Valor
Método de acceso	POST
Actor	Administrador
URL	/horarios

Proceso	● Validar que **idGrupo, idMateria, idGrado y idProfesor** existan en la base de datos. ● Validar que el horario no se solape con otros horarios del mismo profesor o grupo. ● Crear el registro del horario si todas las validaciones son correctas.
Entrada	{ "idHorario":" int", "idGrupo": "int", "idMateria": "int", "idGrado": "int", "idProfesor": "int", "dia": "String", "horaInicio": "date", "horaFin": "date" },

Salida	{
	"estatus":"boolean",
	"mensaje":"String"
	}

Consultar horario

Elemento	Valor
Método de acceso	GET
Actor	Administrador, Profesor
URL	/horarios
Proceso	● Validar que **idGrupo, idMateria, idGrado y idProfesor** existan en la base de datos. ● Validar que el horario no se solape con otros horarios del mismo profesor o grupo. ● Crear el registro del horario si todas las validaciones son correctas.
Entrada	N/A
Salida	[{ "idHorario":" int", "idGrupo": "int", "idMateria": "int",

	“idGrado”: “int”,
	"idProfesor": “int”,
	"dia": “String",
	"horaInicio": "date",
	"horaFin": "date"
	},
	{
	"idHorario":” int”,
	"idGrupo": “int”,
	"idMateria": “int”,
	“idGrado”: “int”,
	"idProfesor": “int”,
	"dia": “String",
	"horaInicio": "date",
	"horaFin": "date"
	}
	]

Consultar Horario por ID

Elemento	Valor
Método de acceso	GET
Actor	Administrador
URL	/horarios/{idHorario}
Proceso	● Verificar que **idHorario** exista. ● Si existe, retornar los datos completos del horario. ● Si no existe, retornar error con mensaje “Horario no encontrado”.

Entrada	{
	"idHorario": int
	}

Salida	[{ "idHorario":” int”, "idGrupo": “int”, "idMateria": “int”, “idGrado”: “int”, "idProfesor": “int”, "dia": “String", "horaInicio": "date", "horaFin": "date" },

	{
	"idHorario":” int”,
	"idGrupo": “int”,
	"idMateria": “int”,
	“idGrado”: “int”,
	"idProfesor": “int”,
	"dia": “String",
	"horaInicio": "date",
	"horaFin": "date"
	}
	]

Consultar horarios por grupo

Elemento	Valor
Mètodo de acceso	GET
Actor	Administrador, Profesor
URL	/horarios/grupo/{idGrupo}
Proceso	● Listar todos los registros de horarios existentes. ● Permitir filtros opcionales (por profesor, materia, grupo, día).

Entrada	{
	"idGrupo": int
	}

Salida	[{ "idHorario":” int”, "idGrupo": “int”, "idMateria": “int”, “idGrado”: “int”, "idProfesor": “int”, "dia": “String", "horaInicio": "date", "horaFin": "date" }, { "idHorario":” int”, "idGrupo": “int”, "idMateria": “int”, “idGrado”: “int”, "idProfesor": “int”, "dia": “String",

	"horaInicio": "date", "horaFin": "date" }]

Consultar horario por profesores

Elemento	Valor
Mètodo de acceso	GET
Actor	Administrador
URL	/horarios/profesor/{idProfesor}
Proceso	• Validar que **idProfesor** exista. • Retornar todos los horarios donde el profesor esté asignado. • Ordenar por día y hora.

Entrada	{
	"idProfesor": int
	}

Salida	[
	{
	"idHorario":” int”,
	"idGrupo": “int”,
	"idMateria": “int”,
	“idGrado”: “int”,
	"idProfesor": “int”,
	"dia": “String",
	"horaInicio": "date",
	"horaFin": "date"
	},
	{
	"idHorario":” int”,
	"idGrupo": “int”,
	"idMateria": “int”,
	“idGrado”: “int”,
	"idProfesor": “int”,
	"dia": “String",
	"horaInicio": "date",
	"horaFin": "date"
	}
	]

Consultar horario por grado

Elemento	Valor
Mètodo de acceso	GET
Actor	Administrador, Profesor
URL	/horarios/grado/{idGrado}
Proceso	• Validar que **idGrado** exista. • Obtener todos los grupos que pertenezcan al grado. • Obtener todos los horarios asociados a esos grupos. • Retornar la lista completa de horarios, ordenados por día y hora de inicio. • Si no existen horarios para el grado, retornar arreglo vacío.

Entrada	{
	"idGrado": int
	}

Salida	[
	{
	"idHorario":” int”,
	"idGrupo": “int”,
	"idMateria": “int”,
	“idGrado”: “int”,
	"idProfesor": “int”,
	"dia": “String",
	"horaInicio": "date",
	"horaFin": "date"
	},
	{
	"idHorario":” int”,
	"idGrupo": “int”,
	"idMateria": “int”,
	“idGrado”: “int”,
	"idProfesor": “int”,
	"dia": “String",
	"horaInicio": "date",
	"horaFin": "date"
	}
	]

Actualizar horario

Elemento	Valor
Mètodo de acceso	PUT

Actor	Administrador
URL	/horarios/{idHorario}
Proceso	• Validar que **idHorario** exista. • Validar que el nuevo horario no cause conflicto con otros horarios del mismo profesor o grupo. • Actualizar el registro si todas las validaciones son correctas.

Entrada	{
	"idGrupo": “int”,
	"idMateria": “int”,
	"idProfesor": “int”,
	"dia": “date",
	"horaInicio": "date",
	"horaFin": "date"
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Eliminar Horario

Elemento	**Valor**
Mètodo de acceso	DELETE
Actor	Administrador
URL	/horarios/{idHorario}
Proceso	• Validar que **idHorario** exista. • Solo permite eliminar si no tiene clases activas asignadas. • Si tiene clases asignadas, retorna el error.

Entrada	{
	"idHorario": 1
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Asignar clases

Elemento	Valor
Mètodo de acceso	POST
Actor	Administrador
URL	/horarios/{idHorario}/asignar
Proceso	● Validar que **idHorario, idMateria y idProfesor** existan. ● Verificar que el profesor y grupo estén disponibles en ese horario. ● Asignar la clase si no hay conflictos.

Entrada	{
	"idMateria": 2,
	"idProfesor": 3
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

API REST GRUPO

3.-Tipo de servicio:

Este servicio se clasifica como de **Entidad** porque está directamente asociado a la tabla **Grupo** dentro del modelo de datos, la cual representa la organización de los estudiantes en cada grado. Su función principal es administrar los registros de grupos de forma CRUD (Crear, Consultar, Actualizar, Eliminar).

4.- Operaciones expuestas

- **Crear grupo**
- **Consultar grupo por ID**
- **Consultar todos los grupos**
- **Actualizar grupo**
- **Eliminar grupo**
- **Asignar alumno**

- **Asignar profesor**

5.- Descripción de las operaciones.

Crear Grupo

Elemento	**Valor**
Mètodo de acceso	POST
Actor	Administrador
URL	/grupos
Proceso	• Validar que el **nombre** no esté vacío y sea único dentro del grado. • Validar que **idGrado** exista. • Crear el grupo si todas las validaciones pasan.

Entrada	{
	"nombre": "String",
	"idGrado": “int”
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Consultar grupo por ID

Elemento	**Valor**
Mètodo de acceso	GET
Actor	Administrador
URL	/grupos/{idGrupo}
Proceso	• Validar que **idGrupo** exista. • Retornar los datos del grupo, incluyendo los alumnos y profesores asignados si existen.
Entrada	{ "idGrupo": int

	}

Salida	{
	"idGrupo": “int”,
	"nombre": "String",
	"idGrado": “int”,
	"alumnosAsignados": “int”,
	"profesoresAsignados": “int”
	}

Consultar todos los grupos

Elemento	**Valor**
Mètodo de acceso	GET
Actor	Administrador, Profesor
URL	/grupos
Proceso	• Retornar todos los grupos activos en el sistema. • Ordenar por nombre del grupo.
Entrada	N/A

Salida	[
	{
	"idGrupo": “int”,
	"nombre": "String",
	"idGrado": “int”,
	},
	{
	"idGrupo": “int”,
	"nombre": "String",
	"idGrado": “int”,
	}
	]

Actualizar Grupo

Elemento	**Valor**
Mètodo de acceso	POST
Actor	Administrador

URL	/grupos/{idGrupo}
Proceso	• Validar que **idGrupo** exista. • Validar que el **nombre** nuevo sea único dentro del grado. • Actualizar los datos si todas las validaciones pasan.

Entrada	{
	"idGrupo": “int”,
	"nombre": "String",
	"idGrado": “int”,
	"alumnosAsignados": “int”,
	"profesoresAsignados": “int”
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Elimina Grupo

Elemento	**Valor**
Mètodo de acceso	DELETE
Actor	Administrador
URL	/grupos/{idGrupo}
Proceso	• Validar que **idGrupo** exista. • Verificar que no tenga horarios o alumnos asignados. • Si tiene dependencias activas, retornar error; si no, eliminar grupo.

Entrada	{
	"idGrupo": “int”
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Asignar alumno

Elemento	**Valor**

Actor	Administrador
URL	/grupos/{idGrupo}/asignarAlumno
Proceso	● Validar que **idGrupo y idAlumno** existan. ● Verificar que el alumno no esté asignado a otro grupo del mismo grado. ● Si pasa la validación, asignar el alumno al grupo.

Entrada	{
	"idAlumno": 1
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Asignar profesor

Elemento	**Valor**
Actor	Administrador
URL	/grupos/{idGrupo}/asignarProfesor
Proceso	● Validar que **idGrupo** y **idProfesor** existan. ● Verificar que el profesor no tenga conflicto de horarios con este grupo. ● Asignar profesor al grupo si no hay conflictos.

Entrada	{
	"idProfesor": “int”
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

API REST GRADO

3.- Tipo de servicio:

Este servicio se clasifica como de **Entidad** porque está directamente asociado a la tabla **Grado** dentro del modelo de datos, la cual representa los distintos niveles académicos en los

que se organizan los estudiantes. Su función principal es administrar los registros de grados de forma CRUD (Crear, Consultar, Actualizar, Eliminar).

4.- Operaciones expuestas:

- **Crear grado**
- **Consultar grado por ID**
- **Consultar todos los grados**
- **Actualizar grado**
- **Eliminar grado**

5.- Descripción de las operaciones.

Crear grado

Elemento	**Valor**
Mètodo de acceso	POST
Actor	Administrador
URL	/grados
Proceso	• Validar que el **nombre** no esté vacío y no se repita. • Crear el grado en la base de datos si las validaciones pasan.

Entrada	{
	“ id_grado”: “ int”
	“nombre_grado”: “varchar(30)”
	“nombre_nivel”:” varchar (30)”
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Consultar grado por ID

Elemento	**Valor**
Mètodo de acceso	GET
Actor	Administrador

URL	/grados/{idGrado}
Proceso	● Validar que **idGrado** exista. ● Retornar los datos del grado, incluyendo los grupos asociados si existen. ● Si no existe, retornar error “Grado no encontrado”.

Entrada	{
	"idGrado": “int”
	}

Salida	{
	“ id_grado”: “ int”
	“nombre_grado”: “varchar(30)”
	“nombre_nivel”:” varchar (30)”
	}

Consultar todos los grados

Elemento	**Valor**
Mètodo de acceso	GET
Actor	Administrador, profesor
URL	/grados
Proceso	● Retornar todos los grados registrados en el sistema. ● Ordenar por nombre de grado.
Entrada	N/A
Salida	[{ “ id_grado”: “ int” “nombre_grado”: “varchar(30)” “nombre_nivel”:” varchar (30)” }, { “ id_grado”: “ int” “nombre_grado”: “varchar(30)” “nombre_nivel”:” varchar (30)” }, { “ id_grado”: “ int” “nombre_grado”: “varchar(30)” “nombre_nivel”:” varchar (30)” },]

Actualizar grado

Elemento	Valor
Mètodo de acceso	PUT
Actor	Administrador
URL	/grados/{idGrado}
Proceso	• Validar que **idGrado** exista. • Validar que el nuevo nombre no esté duplicado. • Actualizar el grado si todas las validaciones pasan.

Entrada	{
	“ id_grado”: “ int”
	“nombre_grado”: “varchar(30)”
	“nombre_nivel”:” varchar (30)”
	}

Salida	{
	“estatus”:”boolean”,
	"mensaje":"String"
	}

Eliminar grado

Elemento	Valor
Mètodo de acceso	DELETE
Actor	Administrador
URL	/grados/{idGrado}
Proceso	• Validar que i**dGrado** exista. • Verificar que no tenga grupos activos asociados. • Si tiene dependencias, retornar error; si no, eliminar grado.

Entrada	{
	"idGrado": “int”
	}

Salida	{ “estatus”:”boolean”, "mensaje":"String"

	}