



### **Programa educativo**

INGENIERÍA EN SISTEMAS  
COMPUTACIONALES

### **Grupo y Grupo**

8 "A"

### **Nombre de la materia**

Arquitectura de Servicios

### **Nombre del alumno**

Alcalá Ríos Miguel Ángel. (21010252)

Colin Miranda Miguel Ángel. (21010272)

### **Nombre del trabajo**

Actividad: 2do Avance  
del Proyecto

### **Nombre del Profesor**

Roberto Suarez Zinzun

### **Fecha**

14/03/2025

## Tabla Persona:

```
CREATE TABLE `persona` (  
  `idPersona` int NOT NULL AUTO_INCREMENT,  
  `Nombre` varchar(45) NOT NULL,  
  `Apellido` varchar(45) NOT NULL,  
  `FechaNacimineto` date NOT NULL,  
  `Telefono` varchar(10) NOT NULL,  
  `Correo` varchar(45) NOT NULL,  
  `Sexo` varchar(1) NOT NULL,  
  PRIMARY KEY (`idPersona`)  
)  
  
INSERT INTO persona(Nombre, Apellido, FechaNacimineto, Telefono, Correo, Sexo)  
VALUES('Miguel', 'Alcalá', CONVERT('20030531', DATE), '3511234567', 'isc_malcala@accitesz.com', 'M');  
  
INSERT INTO persona(Nombre, Apellido, FechaNacimineto, Telefono, Correo, Sexo)  
VALUES('Erick', 'Gutierrez', CONVERT('20000623', DATE), '3511234568', 'erick@gmail.com', 'M');  
  
INSERT INTO persona(Nombre, Apellido, FechaNacimineto, Telefono, Correo, Sexo)  
VALUES('Yesenia', 'Flores', CONVERT('20020910', DATE), '3511234569', 'yesenia@gmail.com', 'F');  
  
INSERT INTO persona(Nombre, Apellido, FechaNacimineto, Telefono, Correo, Sexo)  
VALUES('Alan', 'Gabriel', CONVERT('20030401', DATE), '3511234570', 'alan@gmail.com', 'M');  
  
INSERT INTO persona(Nombre, Apellido, FechaNacimineto, Telefono, Correo, Sexo)  
VALUES('Viridiana', 'Garcia', CONVERT('20031111', DATE), '3511234571', 'viridiana@gmail.com', 'F');
```

## Tabla Empleados:

```
CREATE TABLE `empleados` (  
  `idEmpleados` int NOT NULL AUTO_INCREMENT,  
  `Contraseña` varchar(45) NOT NULL,  
  `Roll` varchar(45) NOT NULL DEFAULT 'Recepcionista',  
  `Estatus` varchar(45) NOT NULL DEFAULT 'Activo',  
  `id_persona` int NOT NULL,  
  PRIMARY KEY (`idEmpleados`),  
  KEY `id_persona_idx` (`id_persona`),  
  CONSTRAINT `id_persona` FOREIGN KEY (`id_persona`) REFERENCES `persona` (`idPersona`)  
)  
  
INSERT INTO gym.empleados(Contraseña, Roll, Estatus, id_persona)  
VALUES('12345678', 'Admin', 'Activo', 1);  
  
INSERT INTO gym.empleados(Contraseña, Roll, Estatus, id_persona)  
VALUES('12345678', 'Recepcionista', 'Activo', 2);  
  
INSERT INTO gym.empleados(Contraseña, Roll, Estatus, id_persona)  
VALUES('12345678', 'Recepcionista', 'Activo', 3);
```

## Tabla Clientes:

```
CREATE TABLE `clientes` (  
  `idClientes` int NOT NULL AUTO_INCREMENT,  
  `Fecha_Adquisicion` date NOT NULL,  
  `Estatus` varchar(45) NOT NULL,  
  `id_persona` int NOT NULL,  
  PRIMARY KEY (`idClientes`),  
  KEY `persona_cliente_idx` (`id_persona`),  
  CONSTRAINT `persona_cliente` FOREIGN KEY (`id_persona`) REFERENCES `persona` (`idPersona`)  
)  
  
INSERT INTO gym.clientes(Fecha_Adquisicion, Estatus, id_persona)  
VALUES(convert(NOW(), DATE), 'Activo', 5);  
  
INSERT INTO gym.clientes(Fecha_Adquisicion, Estatus, id_persona)  
VALUES(convert(NOW(), DATE), 'Activo', 4);
```

## Tabla Membresías:

```
CREATE TABLE `membresias` (  
  `idMembresias` int NOT NULL AUTO_INCREMENT,  
  `Nombre` varchar(45) NOT NULL,  
  `Descripcion` varchar(255) NOT NULL,  
  `Precio` float NOT NULL,  
  `Vigencia` int NOT NULL,  
  PRIMARY KEY (`idMembresias`)  
)  
  
INSERT INTO gym.membresias(Nombre, Descripcion, Precio, Vigencia)  
VALUES('Mensual', 'Membresía mensual', '400', 30);  
  
INSERT INTO gym.membresias(Nombre, Descripcion, Precio, Vigencia)  
VALUES('Semanal', 'Membresía semanal', '100', 7);
```

## Tabla Detalle\_Membresías:

```
CREATE TABLE `detallemembresia` (  
  `idDetalleMembresia` int NOT NULL AUTO_INCREMENT,  
  `FechaInicio` date NOT NULL,  
  `FechaFin` date NOT NULL,  
  `Estatus` bit(1) NOT NULL,  
  `id_cliente` int NOT NULL,  
  `id_membresia` int NOT NULL,  
  `id_empleado` int NOT NULL,  
  PRIMARY KEY (`idDetalleMembresia`),  
  KEY `id_empleado_idx` (`id_empleado`),  
  KEY `id_cliente_idx` (`id_cliente`),  
  KEY `id_membresia_idx` (`id_membresia`),  
  CONSTRAINT `id_cliente` FOREIGN KEY (`id_cliente`) REFERENCES `clientes` (`idClientes`),  
  CONSTRAINT `id_empleado` FOREIGN KEY (`id_empleado`) REFERENCES `empleados` (`idEmpleados`),  
  CONSTRAINT `id_membresia` FOREIGN KEY (`id_membresia`) REFERENCES `membresias`  
  (`idMembresias`)  
)
```

```
INSERT INTO gym.detallemembresia(FechaInicio, FechaFin, Estatus, id_cliente, id_membresia,  
id_empleado)
```

```
VALUES(CONVERT(NOW(), DATE), CONVERT('20250414', DATE), 1, 2, 1, 2);
```

```
INSERT INTO gym.detallemembresia(FechaInicio, FechaFin, Estatus, id_cliente, id_membresia,  
id_empleado)
```

```
VALUES(CONVERT(NOW(), DATE), CONVERT('20250321', DATE), 1, 3, 2, 3);
```

## Tabla Proveedores:

```
CREATE TABLE `proveedores` (  
  `idproveedores` int NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(150) NOT NULL,  
  `contacto` varchar(100) NOT NULL,  
  `telefono` varchar(11) NOT NULL,  
  `direccion` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`idproveedores`)  
)
```

```
INSERT INTO gym.proveedores(nombre, contacto, telefono, direccion)
```

```
VALUES('Proveedor 1', 'proveedor@gmail.com', '3511234567', 'Calle #1, Valencia 2da Secc. Zamora Mich.');
```

```
INSERT INTO gym.proveedores(nombre, contacto, telefono, direccion)
```

```
VALUES('Proveedor 2', 'proveedor2@gmail.com', '3511234568', 'Calle #2, Valencia 1ra Secc. Zamora Mich.');
```

## Tabla Productos:

```
CREATE TABLE `productos` (  
  `idproductos` int NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(100) NOT NULL,  
  `descripcion` varchar(255) DEFAULT NULL,  
  `categoria` varchar(100) DEFAULT NULL,  
  `precio` float NOT NULL,  
  `existencias` int NOT NULL,  
  `fecha_registro` date NOT NULL,  
  `id_proveedor` int NOT NULL,  
  PRIMARY KEY (`idproductos`),  
  KEY `id_proveedor_idx` (`id_proveedor`),  
  CONSTRAINT `id_proveedor` FOREIGN KEY (`id_proveedor`) REFERENCES `proveedores`  
  (`idproveedores`)  
)
```

```
INSERT INTO gym.productos(nombre, descripcion, categoria, precio, existencias, fecha_registro,  
id_proveedor)
```

```
VALUES('Agua Ciel 600ml.', 'Botella de Agua marca Ciel de 600 mililitros', 'Bebidas', 15, 20,  
CONVERT(NOW(), DATE), 1);
```

```
INSERT INTO gym.productos(nombre, descripcion, categoria, precio, existencias, fecha_registro,  
id_proveedor)
```

```
VALUES('Barra Proteina', 'Barra de proteina', 'Comida', 20, 30, CONVERT(NOW(), DATE), 2);
```



## Tabla Venta:

```
CREATE TABLE `venta` (  
  `idVenta` int NOT NULL AUTO_INCREMENT,  
  `id_usuario` int NOT NULL,  
  `fecha` date NOT NULL,  
  `total` float NOT NULL,  
  `comentarios` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`idVenta`),  
  KEY `id_empleado_idx` (`id_usuario`),  
  CONSTRAINT `id_usuario` FOREIGN KEY (`id_usuario`) REFERENCES `empleados` (`idEmpleados`)  
)
```

```
INSERT INTO gym.venta(id_usuario, fecha, total, comentarios)  
VALUES(2, CONVERT(NOW(), DATE), 45, 'Venta Exitosa');
```

```
INSERT INTO gym.venta(id_usuario, fecha, total, comentarios)  
VALUES(3, CONVERT(NOW(), DATE), 50, 'Venta Exitosa');
```

## Tabla Detalle\_Venta:

```
CREATE TABLE `gym`.`detalleventa` (  
  `iddetalleventa` INT NOT NULL,  
  `id_venta` INT NOT NULL,  
  `id_producto` INT NOT NULL,  
  `cantidad` INT NOT NULL,  
  `precio_unitario` FLOAT NOT NULL,  
  `subtotal` FLOAT NOT NULL,  
  PRIMARY KEY (`iddetalleventa`),  
  INDEX `id_venta_idx` (`id_venta` ASC) VISIBLE,  
  INDEX `id_producto_idx` (`id_producto` ASC) VISIBLE,  
  CONSTRAINT `id_venta`  
    FOREIGN KEY (`id_venta`)  
      REFERENCES `gym`.`venta` (`idVenta`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `id_producto`  
    FOREIGN KEY (`id_producto`)  
      REFERENCES `gym`.`productos` (`idproductos`)  
);  
  
INSERT INTO gym.detalleventa(id_venta, id_producto, cantidad, precio_unitario, subtotal)  
VALUES(1, 1, 3, 15, 45);  
  
INSERT INTO gym.detalleventa(id_venta, id_producto, cantidad, precio_unitario, subtotal)  
VALUES(2, 2, 2, 15, 30);
```

```
INSERT INTO gym.detalleventa(id_venta, id_producto, cantidad, precio_unitario, subtotal)
VALUES(2, 2, 1, 20, 20);
```

# 1. Servicio de Empleados

**Nombre del Servicio:** Servicio de Empleados

**Tipo de Servicio:** Entidad

Justificación: Este servicio gestiona la creación, actualización y eliminación de empleados, lo que implica la manipulación de datos persistentes relacionados con los usuarios del sistema.

**Responsable:** Miguel Ángel Alcalá Ríos

## Operaciones expuestas:

- Agregar Empleado
- Eliminar Empleado
- Actualizar Empleado
- Obtener Empleado

## Descripción de Operaciones

Agregar Empleado	
Actores	Administrador
URL	/empleados/agregar
Método HTTP	POST
Lógica de Negocio	Permite registrar un nuevo empleado en el sistema. Se validará que el nombre de usuario y el correo electrónico no estén duplicados.
Entrada	{ "idPersona": "Int", "Nombre": "string", "Apellido": "string", "FechaNacimiento": "date", "telefono": "string", "Correo": "string", "Sexo": "string", "Contraseña": "string", "Roll": "string", "Estatus": "string" }
Salida	{ "idPersona": "int", "mensaje": "Usuario creado exitosamente" }

Eliminar Empleado	
Actores	Administrador
URL	/Empleado/eliminar/{idEmpleados}
Método HTTP	DELETE
Lógica	Elimina un usuario del sistema basado en su ID.

<b>de Negocio</b>	Se validará que el empleado exista antes de proceder con la eliminación.
<b>Entrada</b>	{ "idEmpleado": "Int" }
<b>Salida</b>	{ "mensaje": "Empleado eliminado exitosamente" }

<b>Actualizar Empleado</b>	
<b>Actores</b>	Administrador
<b>URL</b>	/Empleado/Actualizar/{idEmpleados}
<b>Método HTTP</b>	PUT
<b>Lógica de Negocio</b>	Actualiza la información de un empleado existente. Se validará que el usuario exista y que los datos no estén duplicados.
<b>Entrada</b>	{ "Nombre": "string", "Apellido": "string", "FechaNacimiento": "date", "telefono": "string", "Correo": "string", "Sexo": "string", "Contraseña": "string", "Roll": "string", "Estatus": "string" }
<b>Salida</b>	{ "mensaje": "Usuario actualizado exitosamente" }

<b>Obtener Empleado</b>	
<b>Actores</b>	Administrador
<b>URL</b>	/Empleado/Obtener/{idEmpleados}
<b>Método HTTP</b>	GET
<b>Lógica de Negocio</b>	Obtiene la información de un Empleado específico basado en su ID.
<b>Entrada</b>	{ "idEmpleado": "Int" }
<b>Salida</b>	{

	<pre>         "idPersona": "Int",         "Nombre": "string",         "Apellido": "string",         "FechaNacimiento": "date",         "telefono": "string",         "Correo": "string",         "Sexo": "string",         "Contraseña": "string",         "Roll": "string",         "Estatus": "string"     } </pre>
--	---

## 2. Servicio de Clientes

**Nombre del Servicio:** Servicio de Clientes

**Tipo de Servicio:** Entidad

Justificación: Este servicio gestiona el registro de clientes y la asignación de membresías, lo que implica la manipulación de datos persistentes relacionados con los clientes del gimnasio.

**Responsable:** Miguel Ángel Alcalá Ríos

### Operaciones expuestas:

- Agregar Cliente
- Eliminar Cliente
- Actualizar Cliente
- Obtener Cliente

### Descripción de las operaciones:

Agregar Cliente	
Actores	Administrador-Recepcionista
URL	/clientes/agregar
Método HTTP	POST
Lógica de Negocio	Permite registrar un nuevo cliente en el sistema. Se validará que el correo electrónico y el teléfono no estén duplicados.
Entrada	{ "idPersona": "Int", "Nombre": "string", "Apellido": "string", "FechaNacimiento": "date", "telefono": "string", "Correo": "string", "Sexo": "string", "Fecha_Aquisicion": "string", "Estatus": "string" }
Salida	{ "idPersona": "int", "mensaje": "Cliente creado exitosamente" }

Eliminar Cliente	
Actores	Administrador
URL	/Clientes/eliminar/{idClientes}
Método HTTP	DELETE
Lógica	Elimina un cliente del sistema basado en su ID.

<b>de Negocio</b>	Se validará que el empleado exista antes de proceder con la eliminación.
<b>Entrada</b>	{ "clientId": "Int" }
<b>Salida</b>	{ "mensaje": "Cliente eliminado exitosamente" }

<b>Actualizar Cliente</b>	
<b>Actores</b>	Administrador
<b>URL</b>	/Clientes/Actualizar/{idClientes}
<b>Método HTTP</b>	PUT
<b>Lógica de Negocio</b>	Actualiza la información de un cliente existente. Se validará que el usuario exista y que los datos no estén duplicados.
<b>Entrada</b>	{ "idPersona": "Int", "Nombre": "string", "Apellido": "string", "FechaNacimiento": "date", "telefono": "string", "Correo": "string", "Sexo": "string", "Fecha_Adquisicion": "string", "Estatus": "string" }
<b>Salida</b>	{ "mensaje": "Cliente actualizado exitosamente" }

<b>Obtener Cliente</b>	
<b>Actores</b>	Administrador
<b>URL</b>	/Cliente/Obtener/{idClientes}
<b>Método HTTP</b>	GET
<b>Lógica de Negocio</b>	Obtiene la información de un Cliente específico basado en su ID.
<b>Entrada</b>	{ "clientId": "Int" }



Salida	<pre>{   "idPersona": "Int",   "Nombre": "string",   "Apellido": "string",   "FechaNacimiento": "date",   "telefono": "string",   "Correo": "string",   "Sexo": "string",   "Fecha_Adquisicion": "string",   "Estatus": "string" }</pre>
--------	--

### 3. Servicio de Membresía

**Nombre del Servicio:** Servicio de Membresía

**Tipo de Servicio:** Entidad

Justificación: Este servicio gestiona la creación y renovación de membresías, lo que implica la manipulación de datos persistentes relacionados con las membresías de los miembros del gimnasio.

**Responsable:** Miguel Ángel Colin Miranda

#### Operaciones expuestas:

- Agregar Membresía
- Eliminar Membresía
- Actualizar Membresía
- Asignar Membresía

#### Descripción de las operaciones:

Agregar Membresía	
Actores	Administrador-Recepcionista
URL	/Membresia/Agregar/
Método HTTP	POST
Lógica de Negocio	Permite registrar una nueva membresía en el sistema. Se validará que el miembro exista y que no tenga una membresía activa.
Entrada	{ "IdMembresias": "int", "Nombre": "String", "Descripcion": "String", "Precio": "Vigencia" }
Salida	{ "idMembresia": "Int", "mensaje": "Membresia creada con exito" }

Eliminar Membresia	
Actores	Administrador-Recepcionista
URL	/membresias/eliminar/{idMembresias}
Método HTTP	DELETE
Lógica de	Elimina una membresia del sistema basado en su ID. Se validará que el empleado exista antes de

<b>Negocio</b>	proceder con la eliminación.
<b>Entrada</b>	{ "idMembresia": "Int" }
<b>Salida</b>	{ "mensaje": "Cliente eliminado exitosamente" }

<b>Actualizar Membresia</b>	
<b>Actores</b>	Administrador -Recepcionista
<b>URL</b>	/Membresias/Actualizar/{idMembresias}
<b>Método HTTP</b>	PUT
<b>Lógica de Negocio</b>	Permite actualizar la información de una membresía existente.
<b>Entrada</b>	{ "idMembresias": "int", "Nombre": "String", "Descripcion": "String", "Precio": "Vigencia" }
<b>Salida</b>	{ "idMembresia": "Int", "mensaje": "Membresia creada con exito" }

<b>Asignar Membresía</b>	
<b>Actores</b>	Administrador -Recepcionista
<b>URL</b>	/Membresias/Asignar
<b>Método HTTP</b>	POST
<b>Lógica de Negocio</b>	Permite asignar una membresía existente a un miembro específico. Se validará que el miembro exista y que no tenga una membresía activa. Además, se verificará que la membresía esté disponible para ser asignada.
<b>Entrada</b>	{ "idCliente": "int", "idMembresias": "int" }

<b>Salida</b>	<pre>{   "mensaje": "Membresia asignada exitosamente"   "idMembresia": "Int",   "idCliente": "Int" }</pre>
---------------	--

## 4. Servicio de Venta

**Nombre del Servicio:** Servicio de Venta

**Tipo de Servicio:** Tarea

Justificación: Este servicio gestiona las ventas de productos, lo que implica la ejecución de tareas específicas como la creación de ventas, agregar productos a una venta, calcular el total y generar reportes. No se enfoca en la persistencia de datos, sino en la ejecución de procesos.

**Responsable:** Miguel Ángel Colin Miranda

### Operaciones expuestas:

- Crear Venta
- Agregar Producto a Venta
- Calcular Total de Venta
- Finalizar Venta
- Generar Reporte de Ventas

### Descripción de las operaciones:

Crear Venta	
Actores	Administrador -Recepcionista
URL	/Ventas/Crear
Método HTTP	POST
Lógica de Negocio	Permite crear una nueva venta en el sistema. Se validará que el usuario (recepcionista) que realiza la venta exista y tenga permisos para realizar ventas.
Entrada	{ "idEmpleado": "int", "Fecha": "date" }
Salida	{ "idVenta": "int" "mensaje": "Membresia asignada exitosamente" }

Agregar Producto a Venta	
<b>Actores</b>	Administrador -Recepcionista
<b>URL</b>	/Ventas/Agregar-Producto
<b>Método HTTP</b>	POST
<b>Lógica de Negocio</b>	Permite agregar un producto a una venta existente. Se validará que la venta y el producto existan, y que haya suficiente stock del producto.
<b>Entrada</b>	<pre>{   "idVenta": "int",   "idProductos": "int",   "Existencias": "int" }</pre>
<b>Salida</b>	<pre>{   "idVenta": "int",   "idProducto": "int"   "mensaje": "Producto Agregado" }</pre>

Generar Reporte de Ventas	
<b>Actores</b>	Administrador -Recepcionista
<b>URL</b>	/Ventas/Reporte
<b>Método HTTP</b>	Get
<b>Lógica de Negocio</b>	Genera un reporte de todas las ventas realizadas en un período específico
<b>Entrada</b>	<pre>{   "fecha": "date" }</pre>
<b>Salida</b>	<pre>{   "reporte": [     {       "id_venta": "int",       "id_producto": "int",       "cantidad": "int",       "precio_unitario": "int",       "subtotal": "float"     }   ] }</pre>