



UNIVERSIDAD DE LAS FUERZAS ARMADAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

APLICACIONES DISTRIBUIDAS

PARCIAL 2

TEMA: EXAMEN

HECHO POR:

DOCENTE: ING. DARIO MORALES

- Torres Chávez Marlon

Sangolquí,
28 de enero del 2025

Contenido

Diseño e implementación de un sistema de microservicios utilizando Java Spring Boot.	2
Objetivo	2
Descripción de la Actividad	2
Análisis	2
Diseño	3
Equipo	3
Equipo y Miembros	4
Miembros	5
Desarrollo	5
Equipos	6
Equipos	7
MiembroClienteRest	9
Pruebas	9
Crud de miembros	9
Crud de Equipos	11
Repositorio de git	18

Diseño e implementación de un sistema de microservicios utilizando Java Spring Boot.

Objetivo

Diseñar, desarrollar y probar un sistema de microservicios que implemente la gestión de dos entidades principales relacionadas mediante una relación muchos a muchos, utilizando una tabla intermedia para romper dicha relación. Este sistema deberá incluir las operaciones necesarias para manejar la interacción entre las entidades, desde el análisis hasta las pruebas funcionales.

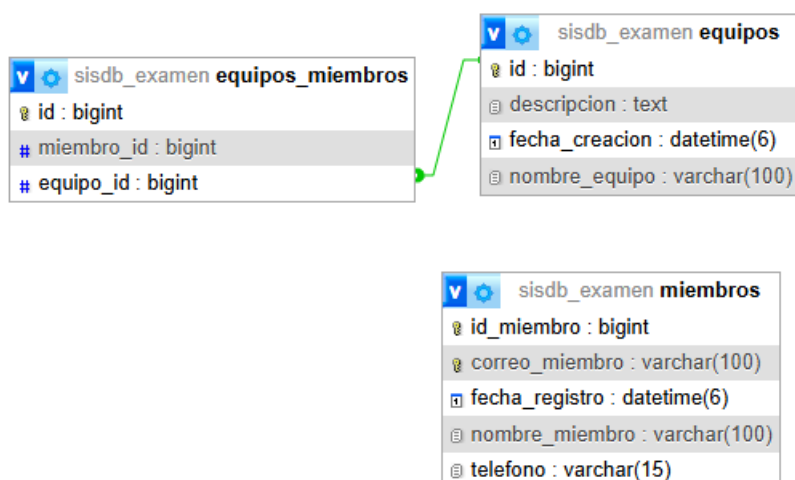
Descripción de la Actividad

Análisis

Identificar las entidades principales y la relación entre ellas.

Diseñar la tabla intermedia para romper la relación muchos a muchos.

Definir las operaciones necesarias (CRUD, listados, etc.).



Diseño

Crear los modelos de datos con JPA.

Configurar relaciones.

Equipo

```

11 @Entity
12 @Table(name = "equipos")
13 public class Equipo {
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Long id;
17
18     @NotEmpty(message = "El nombre del equipo no puede estar vacío")
19     @Size(min = 3, max = 100, message = "El nombre debe tener entre 3 y 100 caracteres")
20     @Column(name = "nombreEquipo", nullable = false, length = 100)
21     private String nombre;
22
23     @Size(max = 500, message = "La descripción no puede exceder los 500 caracteres")
24     @Column(name = "descripcion", columnDefinition = "TEXT")
25     private String descripcion;
26
27     @Column(name = "fechaCreacion", updatable = false)
28     @Temporal(TemporalType.TIMESTAMP)
29     private Date fechaCreacion;
30
31     @PrePersist
32     public void prePersist() {
33         this.fechaCreacion = new Date();
34     }
  
```

```

35
36 @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
37 @JoinColumn(name = "equipo_id")
38 private List<EquipoMiembro> equipoMiembros;
39
40 @Transient
41 private List<Miembro> miembros;
42
43 public Equipo() {
44     equipoMiembros = new ArrayList<>();
45     miembros = new ArrayList<>();
46 }
47
48 public void addEquipoMiembro(EquipoMiembro equipoMiembro) {
49     this.equipoMiembros.add(equipoMiembro);
50 }
51
52 public void removeEquipoMiembro(EquipoMiembro equipoMiembro) {
53     this.equipoMiembros.remove(equipoMiembro);
54 }
55
56 public Long getId() {
57     return id;
58 }

```

Equipo y Miembros

```

package com.espe.micro_equipos.models.entities;

import jakarta.persistence.*;

@Entity
@Table(name = "equipos_miembros")
public class EquipoMiembro {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "miembro_id")
    private Long miembroId;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Long getMiembroId() {
        return miembroId;
    }

    public void setMiembroId(Long miembroId) {
        this.miembroId = miembroId;
    }
}

```

Miembros

```

1 package com.espe.micro_miembros.models.entities;
2
3 import jakarta.persistence.*;
4 import jakarta.validation.constraints.*;
5 import java.util.Date;
6
7 @Entity
8 @Table(name = "miembros")
9 public class Miembro {
10
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     @Column(name = "idMiembro")
14     private Long idMiembro;
15
16     @NotEmpty(message = "El nombre del miembro no puede estar vacío")
17     @Size(min = 3, max = 100, message = "El nombre debe tener entre 3 y 100 caracteres")
18     @Column(name = "nombreMiembro", nullable = false, length = 100)
19     private String nombreMiembro;
20
21     @NotEmpty(message = "El correo no puede estar vacío")
22     @Email(message = "El formato del correo no es válido")
23     @Column(name = "correoMiembro", nullable = false, unique = true, length = 100)
24     private String correoMiembro;
25
26     @Pattern(regexp = "\\d{10}", message = "El teléfono debe tener 10 dígitos")
27     @Column(name = "telefono", length = 15)
28     private String telefono;
29
30     @Column(name = "fechaRegistro", updatable = false)
31     @Temporal(TemporalType.TIMESTAMP)
32     private Date fechaRegistro;
33
34     @PrePersist
35     public void prePersist() {
36         this.fechaRegistro = new Date();
37     }
38 }

```

Desarrollo

Crear microservicios separados para cada entidad o dominio.

Exponer los endpoints para las operaciones requeridas.

Implementar la lógica de negocio en los servicios.

Equipos

The screenshot shows the IntelliJ IDEA IDE with the 'application.properties' file open. The file contains configuration for a Spring Boot application named 'micro_equipos'. The configuration includes the application name, server port (8804), database connection details (JDBC URL, username, password, driver), JPA and Hibernate settings, and logging configuration.

```

1  spring.application.name=micro_equipos
2
3  #Puerto del servidor
4  server.port=8804
5
6  #Configuración de la base de datos
7  spring.datasource.url=jdbc:mysql://localhost:3306/sisdb_examen
8  spring.datasource.username=root123
9  spring.datasource.password=abcd
10 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
11
12 #Configuración de JPA e Hibernate
13 spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
14 spring.jpa.hibernate.ddl-auto=update
15
16 #Nivel de log para ver los comando SQL que se ejecutan
17 logging.level.org.hibernate.SQL=debug
18 logging.level.org.hibernate.type.descriptor.sql.BasicBinder=trace

```

The screenshot shows the IntelliJ IDEA IDE with the 'MiembroController.java' file open. The code defines a REST controller for managing members. It includes package declarations, imports, and two REST endpoints: one for creating a new member and another for updating an existing member. The controller uses Spring's annotations like @RestController, @RequestMapping, @Autowired, @Valid, @RequestBody, @PathVariable, and @PostMapping.

```

1  package com.espe.micro_miembros.controller;
2
3  import com.espe.micro_miembros.models.entities.Miembro;
4
5  @RestController
6  @RequestMapping("/api/miembros")
7  public class MiembroController {
8
9      @Autowired
10     private MiembroService service;
11
12     @PostMapping
13     public ResponseEntity<> create(@Valid @RequestBody Miembro miembro, BindingResult result) {
14         if (result.hasErrors()) {
15             return validar(result);
16         }
17         try {
18             return ResponseEntity.status(HttpStatus.CREATED).body(service.save(miembro));
19         } catch (Exception e) {
20             return ResponseEntity.status(HttpStatus.BAD_REQUEST)
21                 .body(Map.of(k1:"mensaje", "Error al crear el miembro: " + e.getMessage()));
22         }
23     }
24
25     @PutMapping("/{id}")
26     public ResponseEntity<> update(@PathVariable Long id, @Valid @RequestBody Miembro miembro, BindingResult result) {
27         if (result.hasErrors()) {
28             return validar(result);
29         }
30         try {
31             miembro.setIdMiembro(id);
32             return ResponseEntity.status(HttpStatus.OK).body(service.save(miembro));
33         } catch (Exception e) {
34             return ResponseEntity.status(HttpStatus.BAD_REQUEST)
35                 .body(Map.of(k1:"mensaje", "Error al actualizar el miembro: " + e.getMessage()));
36         }
37     }
38 }

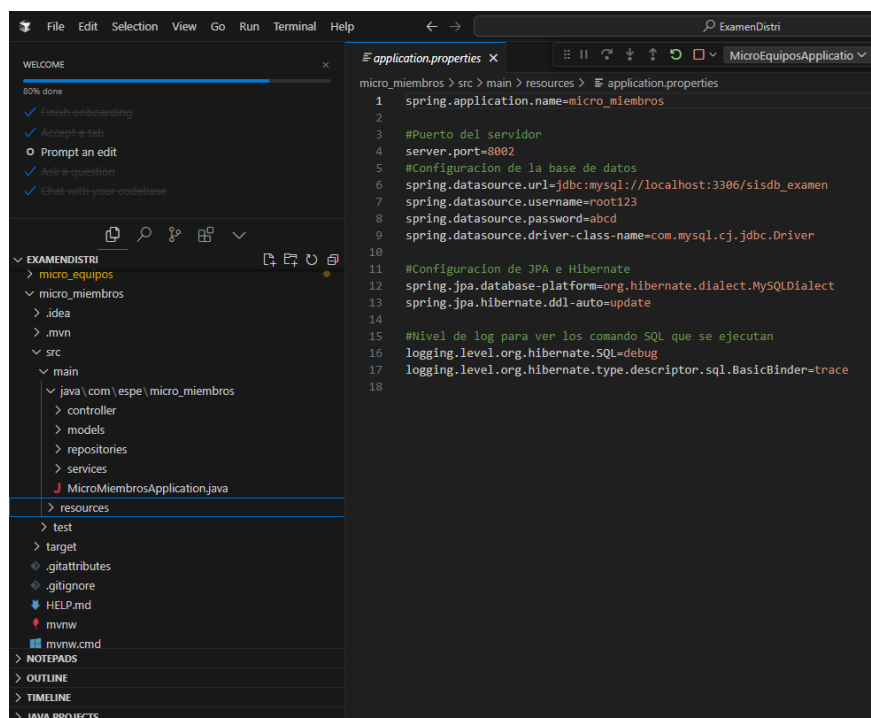
```

```

47     }
48
49     @GetMapping
50     public ResponseEntity<?> findAll() {
51         try {
52             return ResponseEntity.ok().body(service.findAll());
53         } catch (Exception e) {
54             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
55                 .body(Map.of("mensaje", "Error al obtener los miembros: " + e.getMessage()));
56         }
57     }
58
59     @GetMapping("/{id}")
60     public ResponseEntity<?> findById(@PathVariable Long id) {
61         try {
62             return ResponseEntity.ok().body(service.findById(id)
63                 .orElseThrow(() -> new RuntimeException("Miembro no encontrado con ID: " + id)));
64         } catch (RuntimeException e) {
65             return ResponseEntity.status(HttpStatus.NOT_FOUND)
66                 .body(Map.of("mensaje", e.getMessage()));
67         } catch (Exception e) {
68             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
69                 .body(Map.of("mensaje", "Error al obtener el miembro: " + e.getMessage()));
70         }
71     }
72
73     @DeleteMapping("/{id}")
74     public ResponseEntity<?> delete(@PathVariable Long id) {
75         try {
76             service.deleteById(id);
77             return ResponseEntity.noContent().build();
78         } catch (Exception e) {
79             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
80                 .body(Map.of("mensaje", "Error al eliminar el miembro: " + e.getMessage()));
81         }
82     }
83
84     private ResponseEntity<?> validar(BindingResult result) {
85         Map<String, String> errores = new HashMap<>();
86         result.getFieldErrors().forEach(err ->
87             errores.put(err.getField(), err.getDefaultMessage())
88         );
89         return ResponseEntity.badRequest().body(errores);
90     }
91 }
92

```

Equipos



```

micro_equipos > src > main > java > com > espe > micro_equipos > controller > EquipoController.java > ...
1 package com.espe.micro_equipos.controller;
2
3 import com.espe.micro_equipos.models.Miembro;
4
5 @RestController
6 @RequestMapping("/api/equipos")
7 public class EquipoController {
8
9     @Autowired
10     private EquipoService service;
11
12     @PostMapping
13     public ResponseEntity<> create(@Valid @RequestBody Equipo equipo, BindingResult result) {
14         if (result.hasErrors()) {
15             return validar(result);
16         }
17         try {
18             return ResponseEntity.status(HttpStatus.CREATED).body(service.save(equipo));
19         } catch (Exception e) {
20             return ResponseEntity.status(HttpStatus.BAD_REQUEST)
21                 .body(Map.of(k1:"mensaje", "Error al crear el equipo: " + e.getMessage()));
22         }
23     }
24
25     @PutMapping("/{id}")
26     public ResponseEntity<> update(@PathVariable Long id, @Valid @RequestBody Equipo equipo, BindingResult result) {
27         if (result.hasErrors()) {
28             return validar(result);
29         }
30         try {
31             equipo.setId(id);
32             return ResponseEntity.status(HttpStatus.OK).body(service.save(equipo));
33         } catch (Exception e) {
34             return ResponseEntity.status(HttpStatus.BAD_REQUEST)
35                 .body(Map.of(k1:"mensaje", "Error al actualizar el equipo: " + e.getMessage()));
36         }
37     }
38
39     @GetMapping
40     public ResponseEntity<> findAll() {
41         try {
42             return ResponseEntity.ok().body(service.findAll());
43         } catch (Exception e) {
44             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
45                 .body(Map.of(k1:"mensaje", "Error al obtener los equipos: " + e.getMessage()));
46         }
47     }
48
49     @GetMapping("/{id}")
50     public ResponseEntity<> findById(@PathVariable Long id) {
51         try {
52             return ResponseEntity.ok().body(service.findById(id)
53                 .orElseThrow(() -> new RuntimeException("Equipo no encontrado con ID: " + id)));
54         } catch (RuntimeException e) {
55             return ResponseEntity.status(HttpStatus.NOT_FOUND)
56                 .body(Map.of(k1:"mensaje", e.getMessage()));
57         } catch (Exception e) {
58             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
59                 .body(Map.of(k1:"mensaje", "Error al obtener el equipo: " + e.getMessage()));
60         }
61     }
62
63     @DeleteMapping("/{id}")
64     public ResponseEntity<> delete(@PathVariable Long id) {
65         try {
66             service.deleteById(id);
67             return ResponseEntity.noContent().build();
68         } catch (Exception e) {
69             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
70                 .body(Map.of(k1:"mensaje", "Error al eliminar el equipo: " + e.getMessage()));
71         }
72     }
73
74     @PostMapping("/{id}/miembros")
75     public ResponseEntity<?> agregarMiembro(@Valid @RequestBody Miembro miembro, @PathVariable Long id) {
76         try {
77             Optional<Miembro> optional = service.addMember(miembro, id);
78             if (optional.isPresent()) {
79                 return ResponseEntity.status(HttpStatus.CREATED).body(optional.get());
80             } else {
81                 return ResponseEntity.status(HttpStatus.NOT_FOUND)
82                     .body(Map.of(k1:"mensaje", "No se encontró el equipo con ID: " + id));
83             }
84         } catch (FeignException e) {
85             return ResponseEntity.status(HttpStatus.NOT_FOUND)
86                 .body(Map.of(k1:"mensaje", "Miembro no encontrado: " + e.getMessage()));
87         } catch (Exception e) {
88             return ResponseEntity.status(HttpStatus.BAD_REQUEST)
89                 .body(Map.of(k1:"mensaje", "Error al agregar el miembro: " + e.getMessage()));
90         }
91     }
92 }

```



```

108 @DeleteMapping("/{equipoId}/miembros/{miembroId}")
109 public ResponseEntity<> eliminarMiembro(@PathVariable Long equipoId, @PathVariable Long miembroId) {
110     try {
111         service.removeMember(equipoId, miembroId);
112         return ResponseEntity.noContent().build();
113     } catch (RuntimeException e) {
114         // Para errores específicos como "miembro no encontrado" o "último miembro"
115         return ResponseEntity.status(HttpStatus.BAD_REQUEST)
116             .body(Map.of(k1:"mensaje", e.getMessage()));
117     } catch (Exception e) {
118         return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
119             .body(Map.of(k1:"mensaje", "Error al eliminar el miembro del equipo: " + e.getMessage()));
120     }
121 }
122
123 private ResponseEntity<> validar(BindingResult result) {
124     Map<String, String> errores = new HashMap<>();
125     result.getFieldErrors().forEach(err ->
126         errores.put(err.getField(), err.getDefaultMessage())
127     );
128     return ResponseEntity.badRequest().body(errores);
129 }
130 }
131

```

MiembroClienteRest

```

J MiembroController.java X J MiembroClientRest.java X
micro_equipos > src > main > java > com > espe > micro_equipos > clients > J MiembroClientRest.java > {} com.espe.micro_equipos.clients
1 package com.espe.micro_equipos.clients;
2
3 import com.espe.micro_equipos.models.Miembro;
4 import org.springframework.cloud.openfeign.FeignClient;
5 import org.springframework.web.bind.annotation.*;
6
7 @FeignClient(name = "micro-miembros", url = "http://localhost:8002/api/miembros")
8 public interface MiembroClientRest {
9     @GetMapping("/{id}")
10     Miembro findById(@PathVariable Long id);
11 }
12

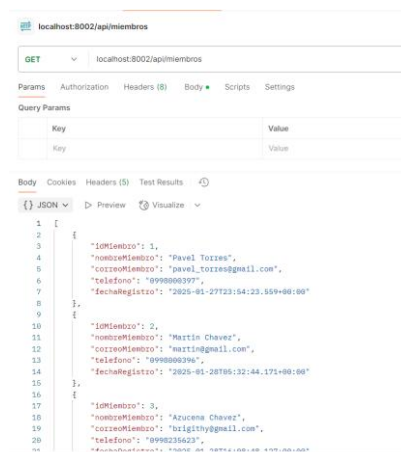
```

Pruebas

Validar cada operación (CRUD, listados, eliminaciones).

Crud de miembros

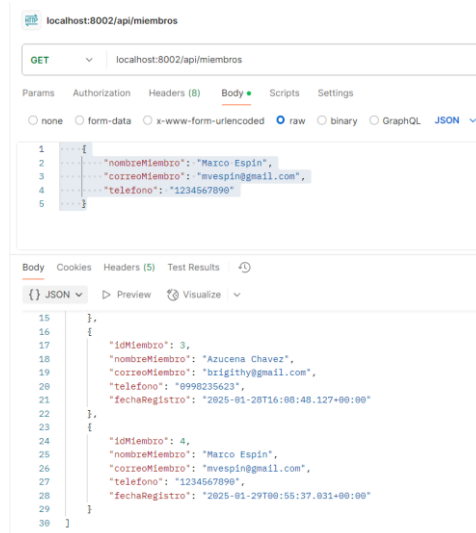
Petición Get: localhost:8002/api/miembros



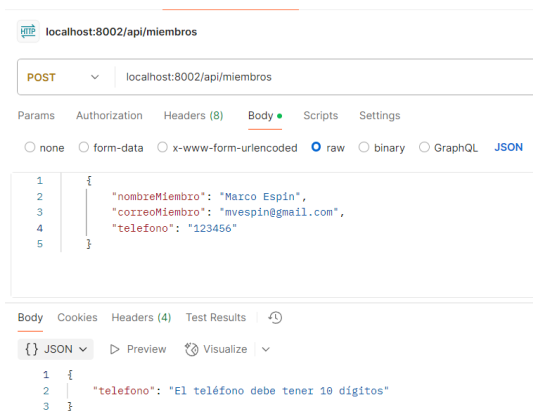
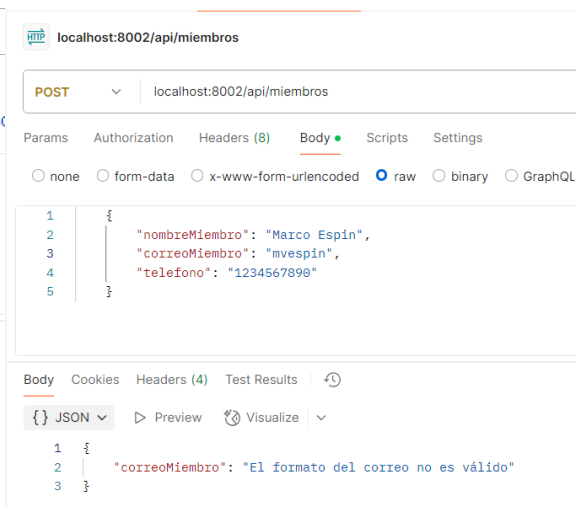
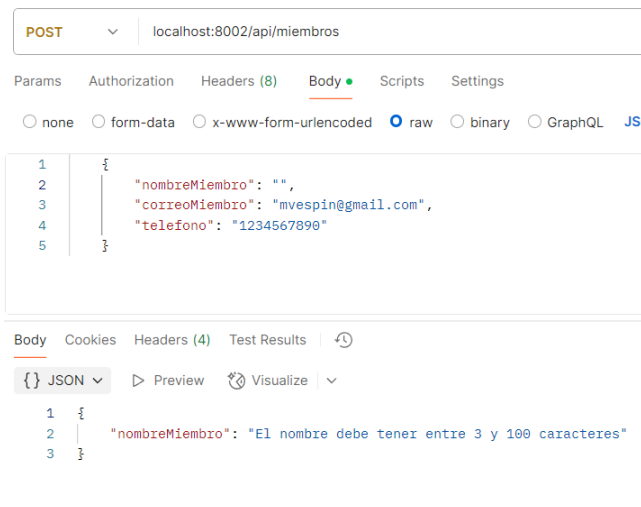
Petición Post: localhost:8002/api/miembros

Json:

```
{
  "nombreMiembro": "Marco Espin",
  "correoMiembro": "mvespin@gmail.com",
  "telefono": "1234567890"
}
```



Validaciones de Miembros



Petición Put: localhost:8002/api/miembros/1

Json:

```
{
  "idMiembro": 1,
  "nombreMiembro": "Marlon Torres",
  "correoMiembro": "mptorres4@espe.edu.ec",
  "telefono": "0998000396"
}
```

The screenshot displays two instances of a REST client. The left instance shows a PUT request to `localhost:8002/api/miembros/1` with a raw JSON body. The right instance shows the same request with the JSON body formatted for readability.

Left Interface (Raw JSON):

```
1 {
2   "idMiembro": 1,
3   "nombreMiembro": "Marlon Torres",
4   "correoMiembro": "mptorres4@espe.edu.ec",
5   "telefono": "0998000396"
6 }
```

Right Interface (Formatted JSON):

```
1 {
2   "idMiembro": 1,
3   "nombreMiembro": "Marlon Torres",
4   "correoMiembro": "mptorres4@espe.edu.ec",
5   "telefono": "0998000396"
6 }
```

Petición Delet: localhost:8002/api/miembros/4

The screenshot displays two instances of a REST client. The left instance shows a DELETE request to `localhost:8002/api/miembros/4` with a raw JSON body. The right instance shows the same request with the JSON body formatted for readability.

Left Interface (Raw JSON):

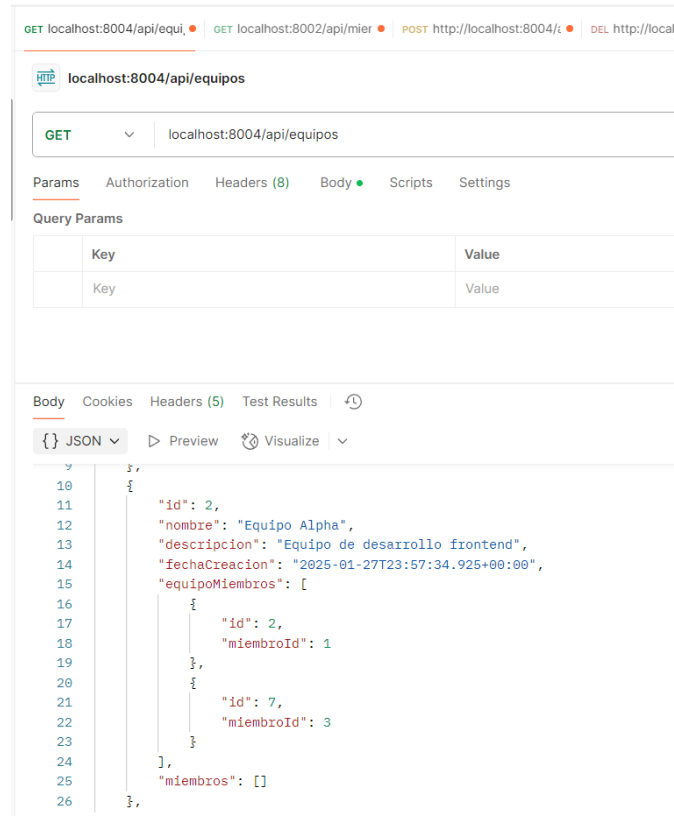
```
1 {
2   "idMiembro": 2,
3   "nombreMiembro": "Martin Chavez",
4   "correoMiembro": "martin@espe.edu.ec",
5   "telefono": "0998000396",
6   "fechaRegistro": "2025-01-27T23:54:23.559+00:00"
7 }
```

Right Interface (Formatted JSON):

```
1 {
2   "idMiembro": 2,
3   "nombreMiembro": "Martin Chavez",
4   "correoMiembro": "martin@espe.edu.ec",
5   "telefono": "0998000396",
6   "fechaRegistro": "2025-01-27T23:54:23.559+00:00"
7 }
```

Crud de Equipos

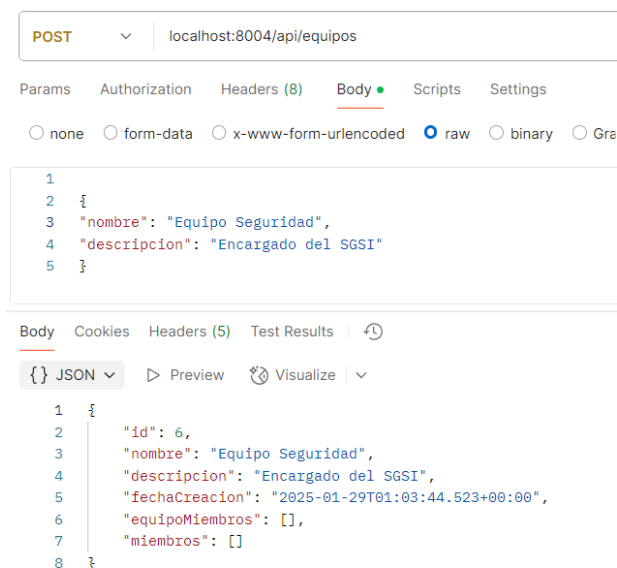
Petición Get: localhost:8004/api/equipos



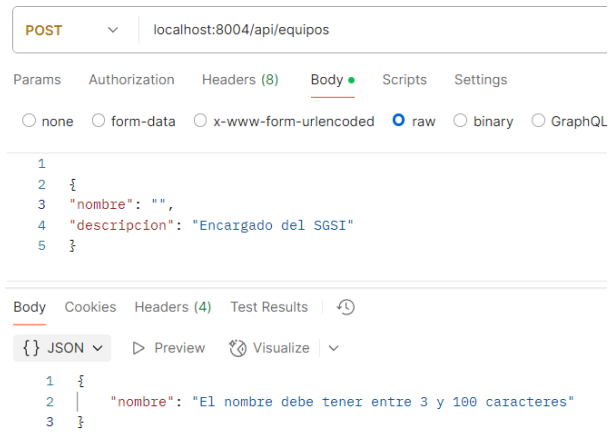
Petición Post: localhost:8004/api/equipos

Json:

```
{
  "nombre": "Equipo Seguridad",
  "descripcion": "Encargado del SGSI"
}
```



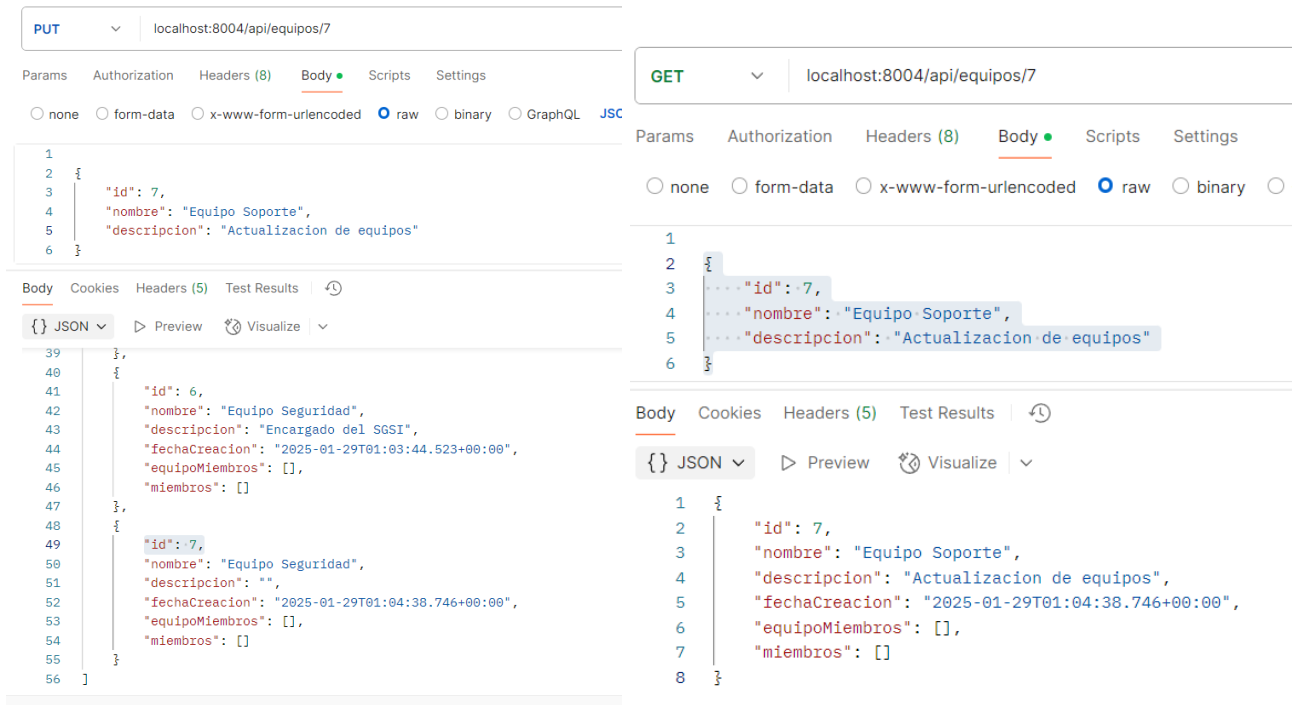
Validaciones de Equipos



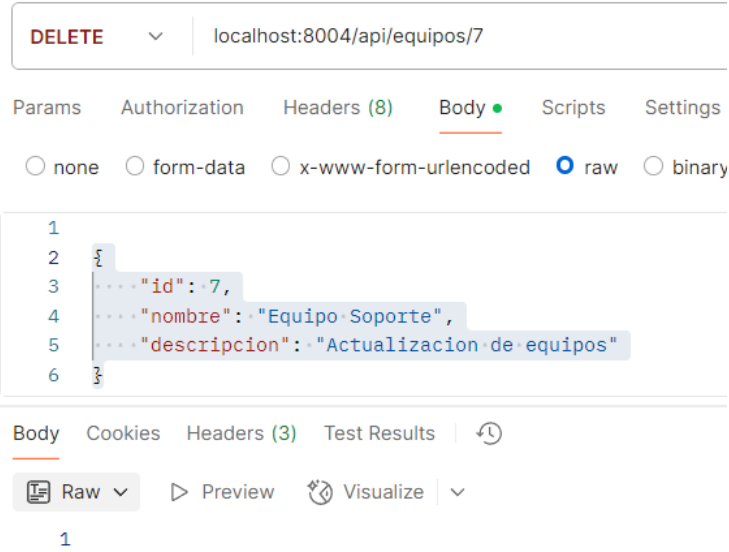
Petición Put: localhost:8004/api/equipos/7

Json:

```
{
  "id": 7,
  "nombre": "Equipo Soporte",
  "descripcion": "Actualizacion de equipos"
}
```



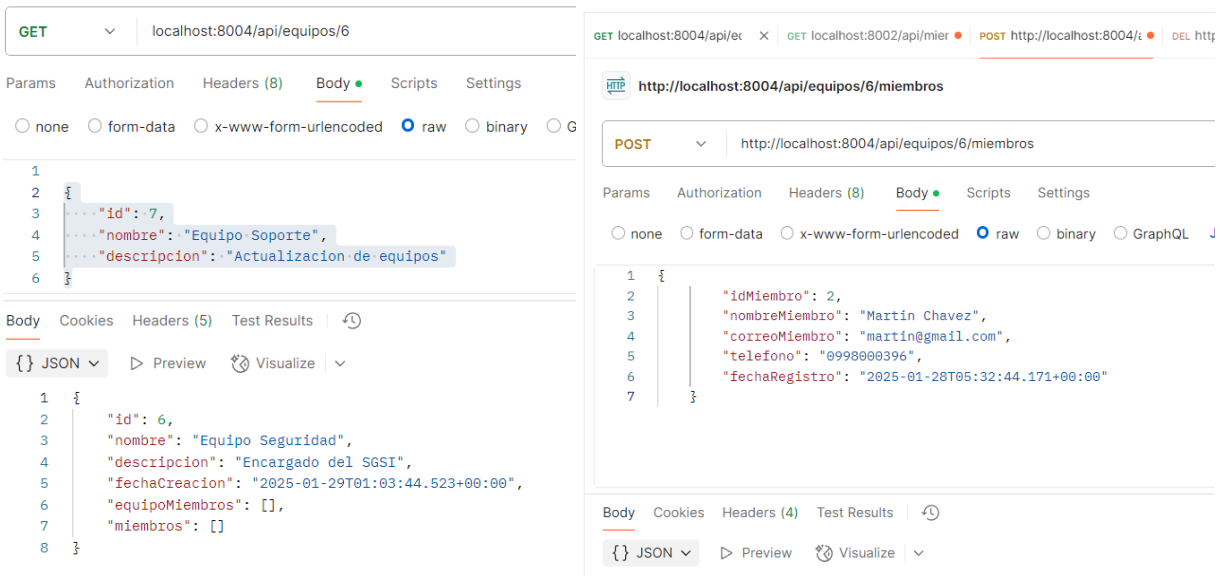
Petición Delet: localhost:8004/api/equipos/7

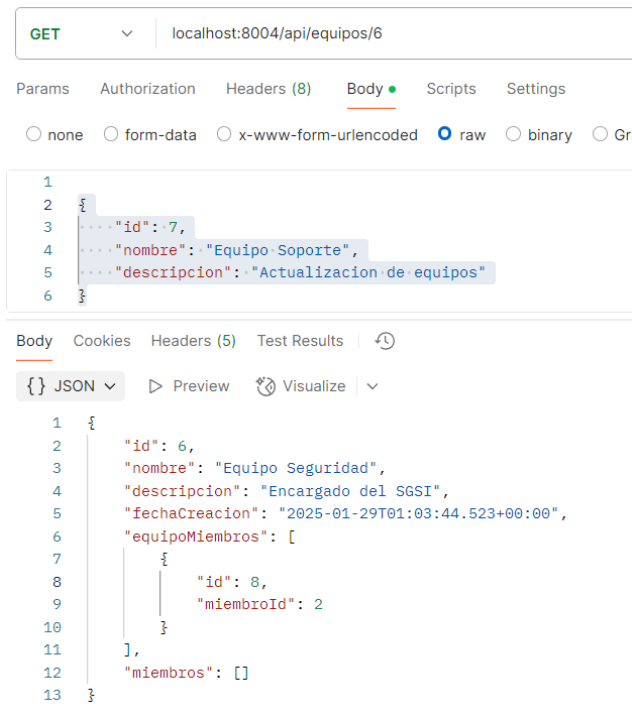


Petición para Añadir miembros a un equipo: <http://localhost:8004/api/equipos/6/miembros>

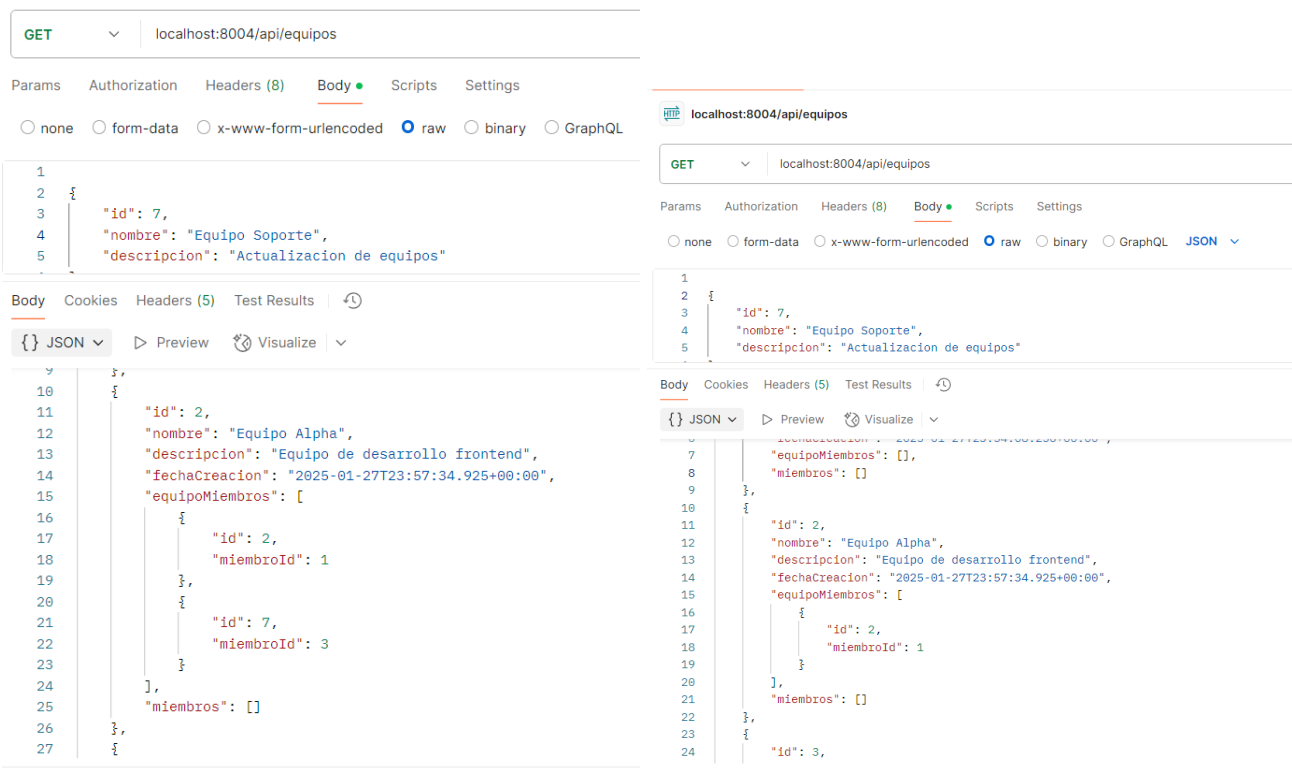
Json:

```
{
  "idMiembro": 2,
  "nombreMiembro": "Martin Chavez",
  "correoMiembro": "martin@gmail.com",
  "telefono": "0998000396",
  "fechaRegistro": "2025-01-28T05:32:44.171+00:00"
}
```

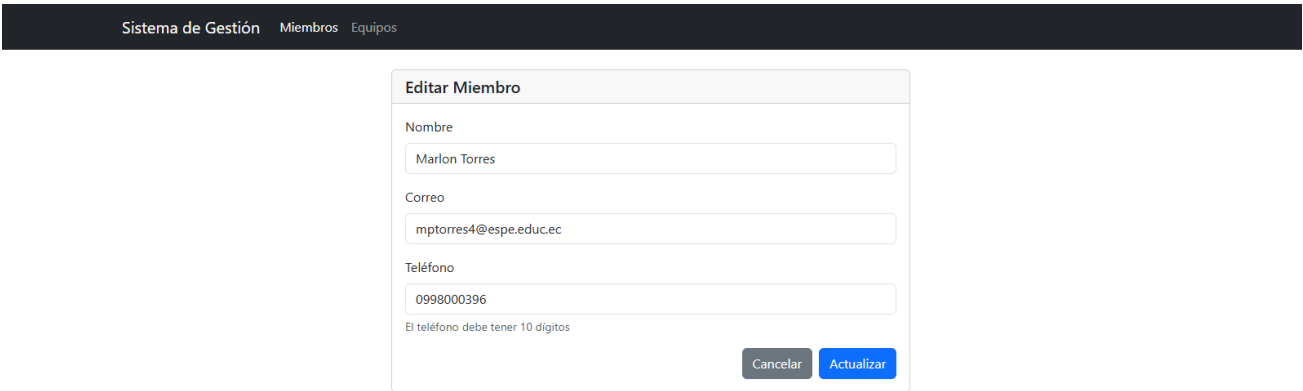
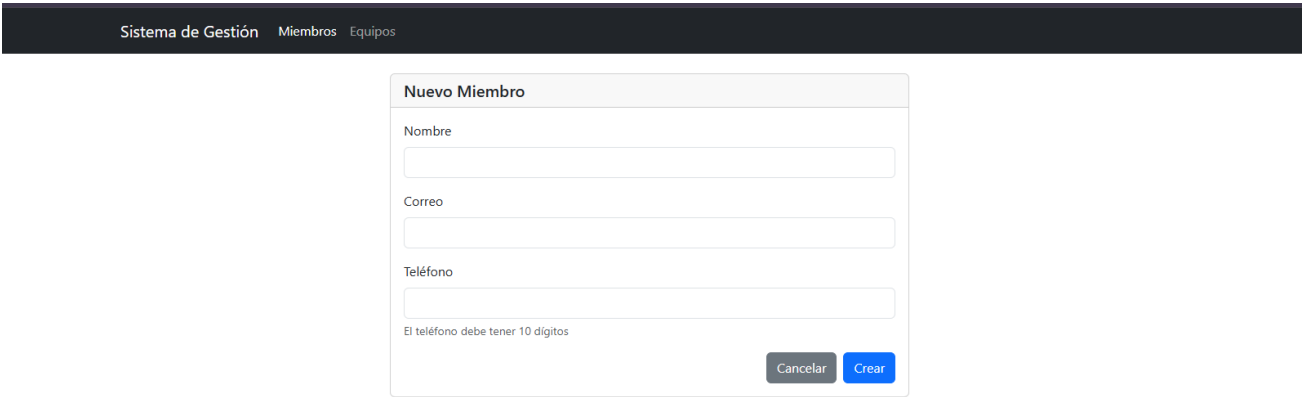
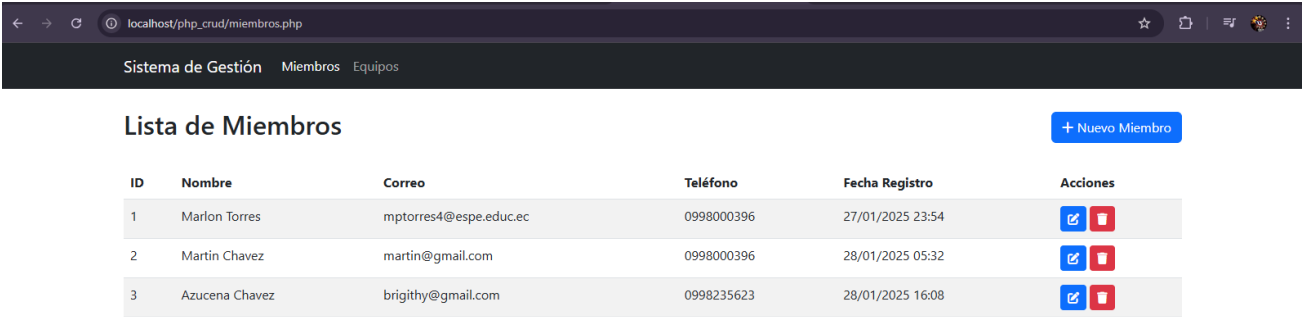
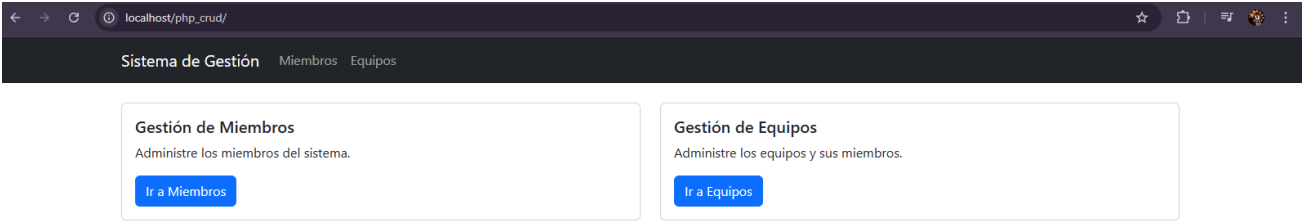




Petición para eliminar miembros de un equipo: <http://localhost:8004/api/equipos/2/miembros/3>



Parte Visual con el uso de Apache:



Sistema de Gestión Miembros Equipos

Lista de Equipos

+ Nuevo Equipo

ID	Nombre	Descripción	Fecha Creación	Miembros	Acciones
1	Equipo Delta	Gestion de pruebas	27/01/2025 23:54	0	  
2	Equipo Alpha	Equipo de desarrollo frontend	27/01/2025 23:57	1	  
3	Equipo Beta	Equipo de desarrollo backend	28/01/2025 05:34	1	  
6	Equipo Seguridad	Encargado del SGI	29/01/2025 01:03	1	  

Sistema de Gestión Miembros Equipos

Nuevo Equipo

Nombre

Descripción

Máximo 500 caracteres

Cancelar Crear

Sistema de Gestión Miembros Equipos

Editar Equipo

Nombre

Equipo Delta

Descripción

Gestion de pruebas

Máximo 500 caracteres

Cancelar Actualizar

Sistema de Gestión Miembros Equipos

Miembros del Equipo: Equipo Seguridad

Agregar Miembro

Azucena Chavez (brigithy@gmail.com)


Selecione un miembro

Marlon Torres (mptorres4@espe.educ.ec)

Azucena Chavez (brigithy@gmail.com)

+ Agregar

Acciones



Volver a Equipos

Repositorio de git

<https://github.com/MarlonTo/ExamenDistribuidas.git>