TALLER PRÁCTICO

Análisis, Evaluación y Propuesta de Refactorización de Arquitectura Monolítica

Proyecto: Sistema de Encuestas "Espagueti"

Asignatura: Arquitectura de Software

2.1 Análisis del Backend (Java/Spring Boot)

Categoría	Anti-Patrón	Descripción	Líneas de Código
Seguridad	SQL Injection (Concatenación SQL)	Uso de concatenación SQL directamente con los valores proporcionados por el usuario, lo que permite la inyección de código malicioso en las consultas SQL.	41, 78, 101-105, 109
Seguridad	Credenciales Hardcodeadas	Las credenciales de la base de datos (usuario y contraseña) están hardcodeadas en el código fuente, exponiendo información sensible.	16-18
Arquitectura	Sin capas (Service, Repository)	El controlador maneja tanto la lógica de negocio como el acceso a la base de datos, lo que dificulta la mantenibilidad y escalabilidad del sistema.	Todo el archivo
SOLID	Violación SRP (Single Responsibility)	El controlador está realizando múltiples tareas, como la validación de entradas y la consulta a la base de datos, violando el principio de Responsabilidad Única.	Controlador hace todo
Clean Code	Código Duplicado (Validaciones)	La validación de los campos en el cuerpo de la solicitud se repite en varias partes del código, lo que aumenta la complejidad y el riesgo de errores.	33, 60, 75, 90, 96
Manejo de Errores	Uso de printStackTrace() y return null	El uso de printStackTrace() para manejar excepciones es inapropiado en producción, ya que expone detalles internos. Además, el retorno de null no maneja adecuadamente los errores.	51-53, 66-68, 82-84, 112-114
Tipado	Uso de Map genérico sin DTOs	El uso de Map para representar datos sin una estructura definida (como un DTO) dificulta la comprensión y el mantenimiento del código.	30, 46, 72, 87
Rendimiento	Conexión BD recreada cada llamada	Se abre y cierra una nueva conexión de base de datos en cada solicitud, lo que puede generar una sobrecarga de rendimiento.	20-27 (Método jdbc())

Fase 02- Análisis de Anti-Patrones y Malas Prácticas

Seguridad SQL Injection (concatenación SQL)

```
String sql = "INSERT INTO encuestas(pregunta, si_count, no_count) VALUES ('" + pregunta + "', 0, 0)";
jdbc().update(sql);
```

```
String sql = "SELECT id, pregunta, si_count, no_count FROM encuestas WHERE id = " + id;
return jdbc().queryForMap(sql);
```

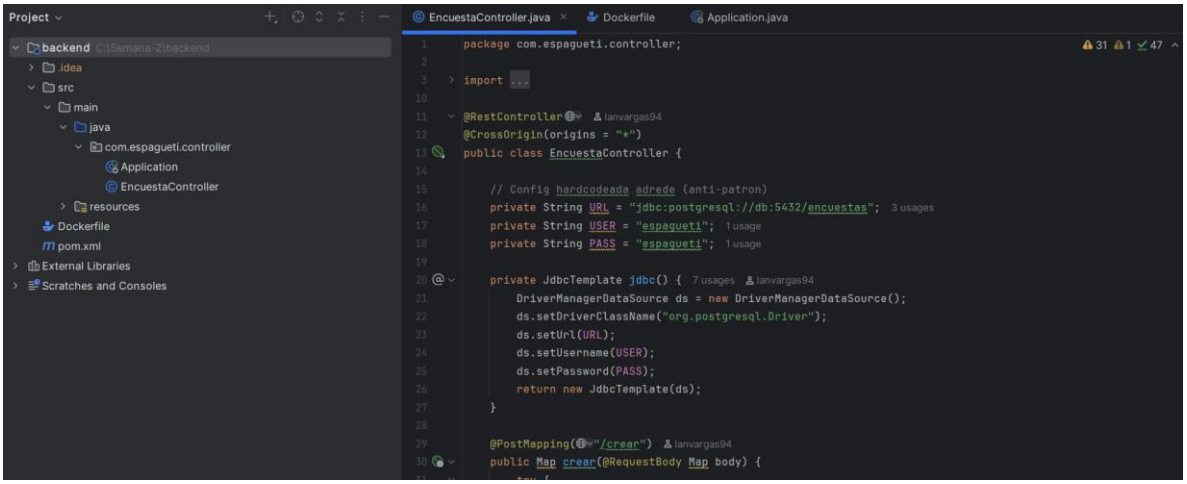
```
if ("SI".equalsIgnoreCase(voto)) {
    String sql = "UPDATE encuestas SET si_count = si_count + 1 WHERE id = " + id;
    jdbc().update(sql);
} else {
    String sql = "UPDATE encuestas SET no_count = no_count + 1 WHERE id = " + id;
    jdbc().update(sql);
```

```
String sql2 = "SELECT id, pregunta, si_count, no_count FROM encuestas WHERE id = " + id;
```

Seguridad Credenciales hardcodeadas

```
private String URL = "jdbc:postgresql://db:5432/encuestas";  3 usages
private String USER = "espagueti";  1 usage
private String PASS = "espagueti";  1 usage
```

Arquitectura Sin capas (Service, Repository)



Fase 02- Análisis de Anti-Patrones y Malas Prácticas

Clean Code Código duplicado(validaciones)

```
30  public Map crear(@RequestBody Map body) {
31      try {
32          // Validaciones hardcodeadas repetidas a propósito
33          if (body == null || body.get("pregunta") == null || body.get("pregunta").toString().trim().length() < 3) {
34              Map r = new HashMap();
35              r.put("error", "Pregunta muy corta");
36              return r;
37          }
```

```
if (URL == null || URL.length() < 5) {
    return null;
}
```

```
75              if (id == null || id <= 0) {
76                  return null;
```

```
90          if (body == null || body.get("id") == null || body.get("voto") == null) {
91              return null;
```

```
96              if (id == null || id <= 0) {
97                  return null;
98              }
```

Manejo de Errores printStackTrace() y return null

```
51              e.printStackTrace();
52          }
53          return null;
```

```
66              e.printStackTrace();
67          }
68          return null;
```

```
81              e.printStackTrace();
82          }
83          return null;
84      }
```

Fase 02- Análisis de Anti-Patrones y Malas Prácticas

```
111         } catch (Exception e) {
112             e.printStackTrace();
113         }
114         return null;
```

Tipado Uso de Map genérico sin DTOs

```
30      public Map crear(@RequestBody Map body) {
```

```
46              Map resp = new HashMap();
                resp.put("id", id);
48              resp.put("pregunta", pregunta);
49              return resp;
```

```
72      public Map encuesta(@PathVariable("id") Integer id) {
```

```
87      public Map votar(@RequestBody Map body) {
88          try {
```

Rendimiento Conexión BD recreada cada llamada

```
20  @   private JdbcTemplate jdbc() {  7 usages   lanvargas94
21          DriverManagerDataSource ds = new DriverManagerDataSource();
22          ds.setDriverClassName("org.postgresql.Driver");
23          ds.setUrl(URL);
24          ds.setUsername(USER);
25          ds.setPassword(PASS);
26          return new JdbcTemplate(ds);
27      }
```

2.2 Análisis del Frontend (Angular)

URL API hardcodeada: En cada componente (crear.component.ts:18, encuesta.component.ts:18, respuestas.component.ts:15)

```
export class RespuestasComponent implements OnInit {
  lista: any = [];
  url: any = 'http://localhost:8080';
  mostrar: any = true;
```

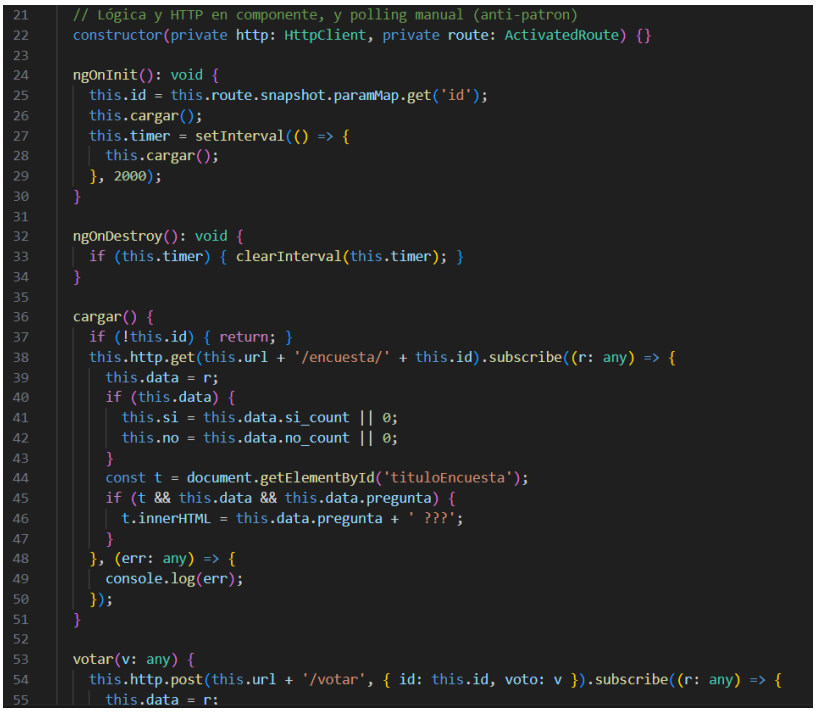
```
13  export class EncuestaComponent implements OnInit, OnDestroy {
14    id: any = 0;
15    data: any = null;
16    si: any = 0;
17    no: any = 0;
18    url: any = 'http://localhost:8080';
19    timer: any = null;
```

```
14  export class CrearComponent {
15    pregunta: any = '';
16    loading: any = false;
17    msg: any = '';
18    url: any = 'http://localhost:8080';
```

HttpClient directo: Sin servicio intermedio (violación del patrón Service)

```
20  // HttpClient directo en el componente (anti-patron)
21  constructor(private http: HttpClient, private router: Router) {}
22
23  publicar() {
24    this.loading = true;
25
26    if (!this.pregunta || this.pregunta.toString().trim().length < 3) {
27      this.msg = 'Escribe algo mas largo';
28      const x = document.getElementById('msg');
29      if (x) { x.innerHTML = this.msg; }
30      this.loading = false;
31      return;
32    }
33
34    this.http.post(this.url + '/crear', { pregunta: this.pregunta }).subscribe((r: any) => {
35      this.loading = false;
36      if (r && r.id) {
37        this.router.navigateByUrl('/encuesta/' + r.id);
38      } else {
39        this.msg = 'Error raro';
40      }
41    }, (err: any) => {
42      console.log(err);
43      this.loading = false;
44      this.msg = 'Error de servidor';
45    });
46  }
47  }
48
```

Fase 02- Análisis de Anti-Patrones y Malas Prácticas



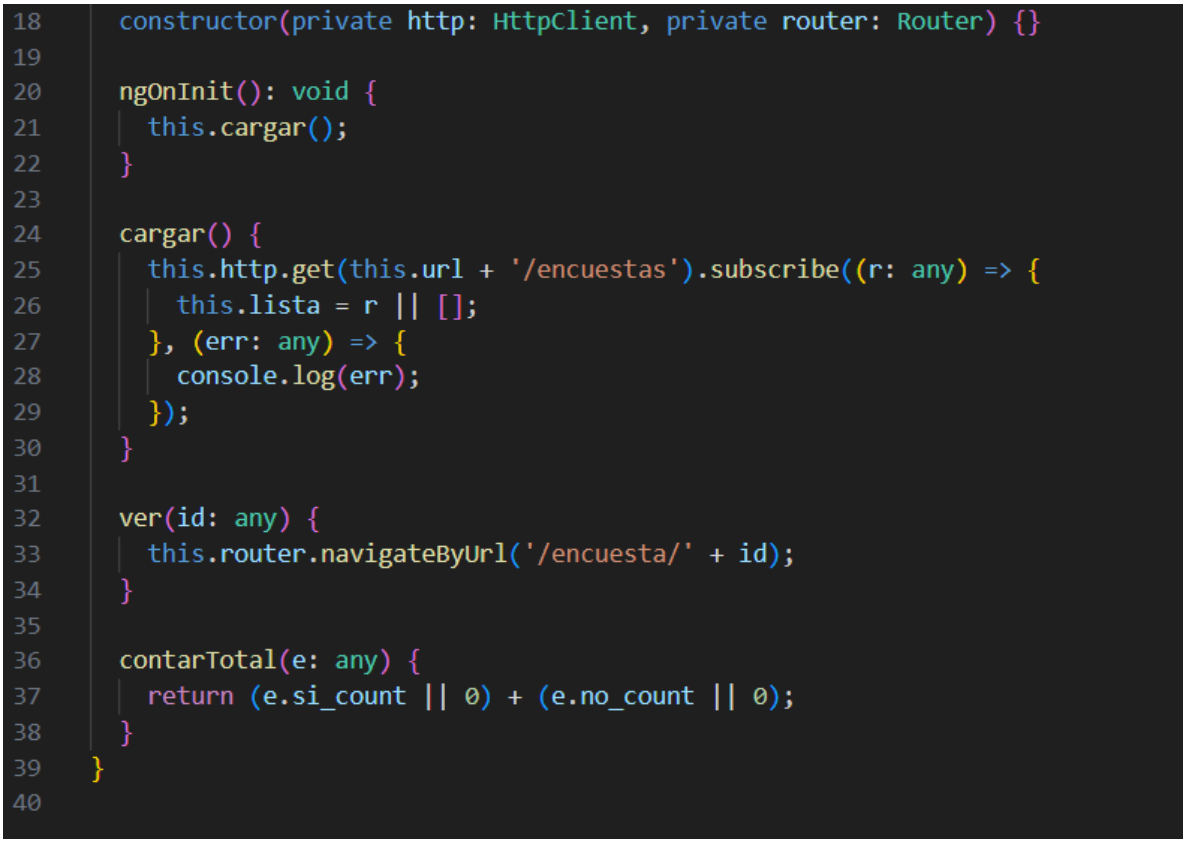
```
// Lógica y HTTP en componente, y polling manual (anti-patron)
constructor(private http: HttpClient, private route: ActivatedRoute) {}

ngOnInit(): void {
  this.id = this.route.snapshot.paramMap.get('id');
  this.cargar();
  this.timer = setInterval(() => {
    this.cargar();
  }, 2000);
}

ngOnDestroy(): void {
  if (this.timer) { clearInterval(this.timer); }
}

cargar() {
  if (!this.id) { return; }
  this.http.get(this.url + '/encuesta/' + this.id).subscribe((r: any) => {
    this.data = r;
    if (this.data) {
      this.si = this.data.si_count || 0;
      this.no = this.data.no_count || 0;
    }
    const t = document.getElementById('tituloEncuesta');
    if (t && this.data && this.data.pregunta) {
      t.innerHTML = this.data.pregunta + ' ???';
    }
  }, (err: any) => {
    console.log(err);
  });
}

votar(v: any) {
  this.http.post(this.url + '/votar', { id: this.id, voto: v }).subscribe((r: any) => {
    this.data = r;
```



```
  constructor(private http: HttpClient, private router: Router) {}

  ngOnInit(): void {
    this.cargar();
  }

  cargar() {
    this.http.get(this.url + '/encuestas').subscribe((r: any) => {
      this.lista = r || [];
    }, (err: any) => {
      console.log(err);
    });
  }

  ver(id: any) {
    this.router.navigateByUrl('/encuesta/' + id);
  }

  contarTotal(e: any) {
    return (e.si_count || 0) + (e.no_count || 0);
  }
}
```

Fase 02- Análisis de Anti-Patrones y Malas Prácticas

Manipulación DOM directa: Uso de document.getElementById (home.component.ts:18, crear.component.ts:29)

```
ngOnInit(): void {
  t.getElementById('titulo');
  ML = t.innerHTML + ' !!!'; }
}
```

Click to collapse the range.

```
26      if (!this.pregunta || this.pregunta.toString().trim().length < 3) {
27        this.msg = 'Escribe algo mas largo';
28        const x = document.getElementById('msg');
29        if (x) { x.innerHTML = this.msg; }
30        this.loading = false;
31        return;
32      }
```

Uso excesivo de 'any': Sin interfaces ni tipos definidos

```
export class CrearComponent {
  pregunta: any = '';
  loading: any = false;
  msg: any = '';
  url: any = 'http://localhost:8080';
```

```
export class EncuestaComponent implements OnInit, OnDestroy {
  id: any = 0;
  data: any = null;
  si: any = 0;
  no: any = 0;
  url: any = 'http://localhost:8080';
  timer: any = null;
```

```
13    export class RespuestasComponent implements OnInit {
14      lista: any = [];
15      url: any = 'http://localhost:8080';
16      mostrar: any = true;
17
```

Polling manual: setInterval en lugar de RxJS (encuesta.component.ts:27-29)

```
24    ngOnInit(): void {
25      this.id = this.route.snapshot.paramMap.get('id');
26      this.cargar();
27      this.timer = setInterval(() => {
28        this.cargar();
29      }, 2000);
30    }
```

Error	Ubicación	Descripción del Problema
URL API Hardcodeada	crear.component.ts:18, encuesta.component.ts:18, respuestas.component.ts:15	La URL de la API está **hardcodeada** en el código, lo que hace que sea difícil de mantener y reutilizar.
HttpClient Directo (Violación del Patrón Service)	crear.component.ts, encuesta.component.ts, respuestas.component.ts	Las llamadas HTTP están directamente en los componentes, violando el **patrón Service**.
Manipulación DOM Directa	home.component.ts:18, crear.component.ts:29	Se usa **document.getElementById** para manipular el DOM directamente, lo que va en contra de la filosofía de Angular.
Uso Excesivo de any	En varios componentes y servicios	Se usa **any** en lugar de definir tipos claros, lo que genera falta de tipado estricto y posibles errores.
Polling Manual (setInterval en lugar de RxJS)	encuesta.component.ts:27-29	Se usa **setInterval** para hacer polling manual, lo cual es un anti-patrón en Angular.