Corporación Universitaria del Huila – CORHUILA

Arquitectura de Software

Taller - Fase 2

Análisis y de Arquitectura Anti-Patrones y Malas Practicas

Docente:

Luis Ángel Vargas Narvaez

Integrantes:

Angie Valentina Flórez Vargas - Avflorez-2023a@corhuila.edu.co

Sebastián Puentes Gonzales – Spuentes-2022b@corhuila.edu.co

Sergio Alejandro Muñoz Cabrera - Samunoz-2023a@corhuila.edu.co

Miguel Ángel Rivera Lozano – marivera-2023ª@corhuila.edu.co

Neiva, Huila

17 de febrero 2026

INTRODUCCIÓN

Durante esta fase se efectuó un análisis profundo del sistema monolítico "Espagueti de Encuestas", cuyo objetivo fue identificar los anti-patrones y fallas estructurales que afectan la seguridad, mantenibilidad, extensibilidad y rendimiento del software.
El diagnóstico cubrió tanto el **backend** (Spring Boot) como el **frontend** (Angular), enfocándose en cómo las decisiones de diseño, la ausencia de capas y los errores recurrentes impactan la calidad global del producto.

El análisis se basó en la inspección del archivo principal del backend (EncuestaController.java), así como de los componentes claves en el frontend (crear, encuesta, respuestas, home), permitiendo una clasificación detallada de los problemas detectados en múltiples dimensiones.

ACTIVIDAD 2.1 - ANÁLISIS DEL BACKEND (SPRING BOOT)

El backend concentra la mayor parte de la complejidad del monolito. La ausencia de capas, la implementación artesanal del acceso a datos y la falta de principios de diseño generan un código difícil de evolucionar y altamente vulnerable.

A continuación se presenta un nuevo análisis con un enfoque narrativo y técnico más profundo

Categoría	Anti-Patrón	Archivo	Líneas(ref.)	Justificación
Seguridad	SQL Injection por concatenación	backend/.../ EncuestaController.java	41, 78, 101–105, 109	Se construyen consultas SQL concatenando entrada del usuario en el controlador, lo que permite inyección (p. ej., ' OR '1'='1). Debe migrarse a consultas parametrizadas (PreparedStatement/ JdbcTemplate) o a Spring Data JPA.

Categoría	Anti-Patrón	Archivo	Líneas(ref.)	Justificación
Seguridad	Credenciales hardcodeadas	backend/.../ EncuestaController.java	16–18	Usuario/contraseña embebidos en código exponen secretos y dificultan su rotación por ambiente. Externalizar en application.yml/ variables de entorno o gestor de secretos.
Arquitectura	Sin capas (Controller hace todo)	backend/.../ EncuestaController.java	Todo el archivo	El controlador mezcla validación, lógica de dominio y acceso a datos, aumentando el acoplamiento y reduciendo la testabilidad. Adoptar **Controller → Service → Repository**.
SOLID	Violación SRP	backend/.../ EncuestaController.java	Todo el archivo	Una sola clase asume múltiples responsabilidades (validar, orquestar, persistir, manejar errores), violando el Principio de Responsabilidad Única. Extraer servicios y validadores.
Clean Code	Código duplicado (validaciones)	backend/.../ EncuestaController.java	33, 60, 75, 90, 96	Validaciones repetidas generan inconsistencias y deuda técnica. Centralizar en métodos utilitarios o clase Validator.
Manejo de errores	printStackTrace() y return null	backend/.../ EncuestaController.java	51–53, 66–68, 82–84, 112–114	Respuestas ambiguas y poca trazabilidad. Implementar ***@ControllerAdvice***, excepciones personalizadas y ResponseEntity con códigos HTTP adecuados.

Categoría	Anti-Patrón	Archivo	Líneas(ref.)	Justificación
Tipado	Uso de Map genérico (sin DTOs)	backend/.../ EncuestaController.java	30, 46, 72, 87	Se pierde contrato de tipos y validación declarativa. Definir DTOs (EncuestaDTO, VotoRequest, EncuestaResponse) con Bean Validation.
Rendimiento	Conexión a BD por request (sin pool)	backend/.../ EncuestaController.java (método jdbc())	20–27	Abrir/cerrar conexión en cada solicitud degrada el rendimiento y arriesga agotamiento de recursos. Usar DataSource con HikariCP (por defecto en Spring Boot).

ACTIVIDAD 2.2 - ANÁLISIS DEL FRONTEND (ANGULAR)

El frontend presenta anti-patrones que afectan mantenibilidad, tipado y eficiencia.

A continuación se presenta un nuevo análisis con un enfoque narrativo y técnico más profundo

Categoría	Anti-Patrón	Archivo	Líneas (ref.)	Justificación
Configuración / Seguridad	URL de API hardcodeada	src/app/crear/ crear.component.ts	18	La baseURL del backend queda fija en el componente, dificultando despliegues multi-ambiente (dev/qa/prod) y duplicando configuración. Mover a environment.ts y consumir desde un servicio.
Configuración / Seguridad	URL de API hardcodeada	src/app/encuesta/ encuesta.component. ts	18	Repetición de la misma URL aumenta probabilidades de error y mantenimiento costoso. Centralizar configuración.
Configuración / Seguridad	URL de API hardcodeada	src/app/respuestas/ respuestas.compone nt.ts	15	La configuración sensible no debe residir en la UI; externalizar y reutilizar vía servicio.

Categoría	Anti-Patrón	Archivo	Líneas (ref.)	Justificación
Arquitectura	HttpClient directo en componentes (sin Service)	crear.component.ts	~21	Mezcla presentación y lógica de acceso a datos, rompiendo separación de responsabilidades y complicando pruebas. Introducir EncuestaService.
Arquitectura	HttpClient directo en componentes (sin Service)	encuesta.component.ts	~22	Los componentes deben delegar la comunicación HTTP a una capa de servicio reutilizable.
Arquitectura	HttpClient directo en componentes (sin Service)	respuestas.component.ts	~18	Duplicación de lógica de consumo de API y manejo de errores. Centralizar en servicio.
Clean Code	Manipulación directa del DOM(document.getElementById)	home.component.ts	18	Angular promueve data binding, ***@ViewChild*** o Renderer2. El acceso directo al DOM es frágil y difícil de testear.
Clean Code	Manipulación directa del DOM(document.getElementById)	crear.component.ts	29	Rompe el enfoque declarativo y puede afectar SSR y pruebas. Reemplazar por bindings/ directivas.
Tipado	Uso excesivo de any (sin interfaces)	Varios (modelos UI)	Varias	Se pierden contratos de datos, autocompletado y validación de tipos. Definir interfaces (Encuesta, Respuesta, ApiResponse) y activar strict.
Rendimiento / UX	Polling con setInterval	encuesta.component.ts	27–29	Redibujo innecesario, consumo de recursos y fugas si no se cancela. Migrar a ***RxJS*** (interval/switchMap/takeUntil) o ***WebSocket/SSE***.

CONCLUSIÓN GENERAL DE LA FASE - 2

El sistema “Espagueti de Encuestas”, aunque funcional, presenta una acumulación significativa de anti-patrones que comprometen su calidad en múltiples niveles.
Las deficiencias más graves se concentran en el backend, donde el riesgo de inyección SQL y la arquitectura sin capas conforman problemas de alto impacto. El frontend, por su parte, muestra carencias en organización, tipado, rendimiento y utilización correcta del framework.

Este diagnóstico demuestra que el monolito requiere una **refactorización estructural**, no solo correcciones puntuales. Las mejoras deben iniciarse en áreas de seguridad y arquitectura, seguidas por una limpieza del código y la incorporación de patrones adecuados para modernizar y escalar el sistema hacia una arquitectura modular o incluso microservicios en fases posteriores.