Contenido

Selección de Tecnologías y Herramientas

Para el desarrollo del sistema, se ha optado por una arquitectura cliente-servidor basada en servicios RESTful. Esta arquitectura desacopla la lógica de negocio (Backend) de las interfaces de usuario (Web, Móvil y Escritorio), permitiendo una escalabilidad modular.

A continuación, se detalla y justifica la pila tecnológica seleccionada:

1. Sistema de Gestión de Base de Datos (Persistencia)

- **Tecnología:** SQL (Implementación: MySQL 8.0 o MariaDB 10.5)
- **Justificación:**
    - Se requiere un modelo relacional robusto para mantener la integridad de las relaciones entre entidades clave (USUARIO, POST, SEGUIR), garantizando la consistencia de datos mediante claves foráneas.
    - MySQL es el estándar de la industria para aplicaciones web, ofreciendo una integración nativa y eficiente con JAVA.

2. Desarrollo Backend (API REST)

- **Lenguaje: Java 11 (LTS)**
    - *Justificación:* Se ha seleccionado una versión LTS (Long Term Support) por su estabilidad, tipado fuerte y gestión de memoria eficiente. Java 11 introduce mejoras en la sintaxis (como var) manteniendo la compatibilidad con servidores empresariales.
- **Framework: Spring Boot 2.7.x**
    - *Justificación:* Es el estándar más usado para microservicios y APIs en Java.
        - Spring Web: Facilita la creación de endpoints RESTful.
        - Spring Data JPA (Hibernate): Permite mapear las tablas SQL a objetos Java (ORM), simplificando las consultas de base de datos sin escribir SQL manual repetitivo.
        - Spring Security: Para la gestión de autenticación (JWT) y autorización.

3. Diseño y Documentación de API

- Estándar: OpenAPI 3.0.3
- Herramientas: Swagger UI / Swagger Editor.
- Justificación:
    - Permite un enfoque *API First*. Definimos el openapi.yaml antes de programar, lo que sirve como guía estricta tanto para el equipo de backend como para los desarrolladores de los clientes (Web, Móvil, Escritorio).
    - Genera documentación interactiva automática para probar los endpoints sin necesidad de programar una interfaz gráfica previa.

4. Desarrollo de Clientes (Frontends)

El sistema contará con tres clientes diferentes que consumirán la misma AP, demostrando la interoperabilidad del sistema.

A. Cliente Web

- Framework: Laravel 9/10 (Lenguaje: PHP)
- Justificación:
    - Laravel permite un desarrollo rápido de interfaces web elegantes usando el motor de plantillas Blade.
    - Actuará como un cliente HTTP (usando el *HTTP Client* nativo de Laravel) que consumirá la API de Java. Esto demuestra la capacidad de integración entre tecnologías dispares (PHP consumiendo Java).

B. Cliente Móvil

- Plataforma: Android Nativo
- Versión Objetivo: Android 8.0 (Oreo) - API Level 26.
- Lenguaje: Java.
- Justificación:
    - Seleccionar la API 26 garantiza una cobertura de mercado de más del 80% de los dispositivos activos.
    - Permite el uso de componentes nativos como RecyclerView para el feed de noticias y Retrofit para las peticiones de red asíncronas hacia el backend.

C. Cliente de Escritorio

- Lenguaje: C# (.NET)
- Framework UI: Windows Forms.
- Justificación:
    - C# ofrece un entorno de desarrollo visual rápido (RAD) en Visual Studio para aplicaciones de escritorio Windows.
    - Se utilizará la librería HttpClient y Newtonsoft.Json para serializar y deserializar los datos provenientes del Backend Java, ideal para un panel de administración o gestión de usuarios avanzada.

5. Resumen de Arquitectura

Capa	Tecnología Seleccionada	Función Principal
Datos	MySQL 8.0	Almacenamiento relacional.
Backend	Java 11 + Spring Boot	Lógica de negocio y seguridad.
API Spec	OpenAPI 3.0.3	Contrato de comunicación.
Web	Laravel (PHP)	Interfaz de usuario en navegador.
Móvil	Android SDK (API 26)	App nativa para usuarios finales.
Desktop	C# (.NET)	App de gestión/administración.

6. Carácter Provisional de la Selección Tecnológica

La selección de tecnologías y herramientas descrita en este apartado constituye un resultado **provisional**, derivado de un análisis inicial de los requisitos funcionales y no funcionales del sistema, así como del alcance definido para la versión actual del proyecto.

La arquitectura y la pila tecnológica propuestas responden a criterios de robustez, escalabilidad, interoperabilidad y adecuación al contexto académico y profesional, pero no deben considerarse definitivas. A medida que el proyecto avance y se disponga de un mayor conocimiento sobre el comportamiento real del sistema, el volumen de datos, las necesidades de rendimiento, seguridad o despliegue, podrán introducirse modificaciones o sustituciones tecnológicas que optimicen la solución final.

Este enfoque iterativo y evolutivo es coherente con las buenas prácticas de ingeniería del software, permitiendo adaptar la solución a nuevos requisitos, restricciones técnicas y/o mejoras detectadas durante las fases posteriores de desarrollo, pruebas y mantenimiento.