**Arquitectura del Sistema – DocuFileFlow PT**

**Curso:** Elaboración de Proyecto de Innovación y/o Mejora  
**Tema:** Del Documento al Prototipo Funcional – Clase 3  
**Versión:** 1.0

**Índice**

1. Introducción
2. Arquitectura Elegida
3. Componentes del Sistema
4. Diagrama de Arquitectura
5. Consideraciones Técnicas

**1. Introducción**

**Problema:**  
En la empresa donde realizo mis prácticas, la gestión de documentos se mantiene en formato físico, lo que ocasiona pérdida de tiempo, duplicidad de registros, riesgo de extravío y dificultad para mantener la trazabilidad.

**Objetivo del documento:**  
Definir la arquitectura técnica del sistema **DocuFlow PT**, que permitirá digitalizar, registrar y gestionar documentos de prácticas profesionales de forma segura y accesible. Este documento servirá de base para el desarrollo del prototipo funcional.

**Revisión de requisitos:**

* **Requisitos técnicos:** Soporte para documentos en PDF, Word y Excel. Validación de tipo y tamaño de archivo (≤ 20 MB).
* **Usuarios estimados:** Entre 10 y 30 simultáneos, principalmente estudiantes, tutores y administradores.
* **Seguridad:** Autenticación con Firebase, validación de tokens JWT, control de roles y registro de auditoría.

**2. Arquitectura Elegida**

**Patrón seleccionado:**  
Arquitectura **Cliente–Servidor en 3 capas** (Presentación, Lógica de Negocio y Datos).

**Justificación:**

* Permite separación clara de responsabilidades entre frontend, backend y base de datos.
* Sencilla de implementar y mantener en una primera versión.
* Escalable a futuro: se puede migrar a PostgreSQL y AWS S3 sin rediseñar toda la solución.
* Facilita la interoperabilidad: aplicaciones móviles y web consumen la misma API REST.

**Ventajas principales:**

* Modularidad y claridad en el diseño.
* Seguridad centralizada en el backend.
* Posibilidad de crecimiento hacia microservicios o integración con otros sistemas.

**3. Componentes del Sistema**

| **CAPA** | **RESPONSABILIDAD** | **TECNOLOGÍA A USAR** | **MÓDULOS INCLUIDOS** |
| --- | --- | --- | --- |
| **Frontend** | Interfaz de usuario para login, carga, búsqueda y descarga de docs | Android (Kotlin, Jetpack Compose) / Web (Bootstrap) | Login, Dashboard, Upload, Search |
| **Backend** | API REST, validación, permisos, gestión de documentos y auditoría | Spring Boot (Kotlin), Retrofit, JWT (Firebase) | Auth, Gestión Docs, Auditoría |
| **Base Datos** | Persistencia de metadatos, usuarios y logs; almacenamiento de archivos | MongoDB (metadatos), Firebase/AWS S3 (archivos) | Users, Files, AuditLogs |

**Frontend:**  
Diseñado en Kotlin con Jetpack Compose para Android. Se conecta al backend mediante Retrofit, permite login, subida de archivos, búsqueda de documentos y descarga segura.

**Backend:**  
Implementado en Spring Boot (Kotlin), gestiona autenticación, validaciones, permisos y reglas de negocio. Expone endpoints REST para CRUD de documentos y auditoría de acciones.

**Base de Datos:**  
MongoDB para almacenar metadatos (usuarios, archivos, logs). Firebase Storage/AWS S3 para almacenar documentos en la nube de forma segura.

**4. Diagrama de Arquitectura**

**Flujo típico:** Login → Subir documento → Listar/Buscar → Descargar

[ Usuario/Cliente (Android/Web) ]

|

HTTPS + JSON

v

[ Backend/API (Spring Boot + JWT) ]

| |

| Queries | Archivos

v v

[ Base de Datos (MongoDB) ] [ Storage (Firebase/AWS S3) ]

**5. Consideraciones Técnicas**

**Seguridad:**

* Autenticación con Firebase y validación de tokens JWT.
* Roles diferenciados (estudiante, tutor, administrador).
* Validación de tipo y tamaño de archivo.
* Registro de auditoría para trazabilidad.

**Escalabilidad:**

* Infraestructura inicial sencilla con MongoDB y Firebase.
* Escalable hacia PostgreSQL y AWS S3 en el futuro.
* Soporte para balanceadores y contenedores (Docker) en etapas posteriores.

**Gestión ante fallos:**

* Respaldos automáticos de base de datos y almacenamiento.
* Manejo centralizado de errores y logs en backend.
* Alertas básicas en caso de fallos críticos.