Informe de Diseño de Software - Sistema de Gestión de Proyectos "ProjectFlow"

Fecha: 2 de julio de 2025

Versión: 1.0

Autor(es): Equipo de Desarrollo ProjectFlow

---

1. Introducción

Este documento presenta el informe de diseño de software para el sistema **ProjectFlow**, una aplicación web diseñada para facilitar la gestión de proyectos, tareas, recursos y seguimiento del progreso. El objetivo principal de ProjectFlow es proporcionar una plataforma centralizada e intuitiva para que equipos de todos los tamaños puedan colaborar eficientemente y llevar sus proyectos a buen término.

---

2. Arquitectura del Sistema

ProjectFlow seguirá una arquitectura de **microservicios**, lo que permitirá un desarrollo, despliegue y escalabilidad independientes de los diferentes componentes del sistema. Esta elección arquitectónica busca maximizar la flexibilidad, la resiliencia y la mantenibilidad del sistema a largo plazo.

2.1. Vista General de Componentes:

- **Servicio de Autenticación y Autorización:** Maneja el registro de usuarios, inicio de sesión y gestión de permisos.
- **Servicio de Proyectos:** Gestiona la creación, actualización, eliminación y consulta de proyectos.
- **Servicio de Tareas:** Administra la creación, asignación, seguimiento del estado y dependencias de las tareas.
- **Servicio de Usuarios y Equipos:** Maneja la información de usuarios, roles y la formación de equipos.
- **Servicio de Notificaciones:** Envía notificaciones a los usuarios sobre cambios en el estado de las tareas, asignaciones, etc.
- **Servicio de Informes y Analíticas:** Genera informes sobre el progreso del proyecto, utilización de recursos y otros KPIs.
- **API Gateway:** Punto de entrada único para todas las solicitudes del cliente, gestionando el enrutamiento a los microservicios apropiados.

2.2. Tecnologías Clave:

- **Backend:** Spring Boot (Java) para los microservicios.
- **Base de Datos:** PostgreSQL para datos relacionales (proyectos, tareas, usuarios) y MongoDB para datos no estructurados (logs, eventos de notificación).
- **Frontend:** React.js con TypeScript para una interfaz de usuario interactiva y robusta.
- **Contenedores:** Docker para empaquetar y desplegar los microservicios.
- **Orquestación de Contenedores:** Kubernetes para la gestión y escalabilidad de los microservicios en un entorno de producción.
- **API RESTful:** Para la comunicación entre el frontend, el API Gateway y los microservicios.
- **Colas de Mensajes:** Kafka para la comunicación asíncrona entre microservicios (por ejemplo, para notificaciones).

---

3. Diseño de Bases de Datos

3.1. Base de Datos Relacional (PostgreSQL):

Se utilizará para almacenar la información estructurada que requiere transacciones y relaciones fuertes.

- **Tabla `Usuarios`:**
  - `id_usuario` (PK, UUID)
  - `nombre` (VARCHAR)
  - `email` (VARCHAR, UNIQUE)
  - `password_hash` (VARCHAR)
  - `rol` (ENUM: 'administrador', 'gerente_proyecto', 'miembro_equipo')
  - `fecha_creacion` (TIMESTAMP)
- **Tabla `Proyectos`:**
  - `id_proyecto` (PK, UUID)
  - `nombre` (VARCHAR)
  - `descripcion` (TEXT)
  - `fecha_inicio` (DATE)
  - `fecha_fin_estimada` (DATE)
  - `estado` (ENUM: 'pendiente', 'en_progreso', 'completado', 'cancelado')
  - `id_gerente_proyecto` (FK a `Usuarios.id_usuario`)
- **Tabla `Tareas`:**

        - `id_tarea` (PK, UUID)
        - `nombre` (VARCHAR)
        - `descripcion` (TEXT)
        - `fecha_vencimiento` (DATE)
        - `prioridad` (ENUM: 'baja', 'media', 'alta', 'critica')
        - `estado` (ENUM: 'por_hacer', 'en_progreso', 'en_revision', 'completado')
        - `id_proyecto` (FK a `Proyectos.id_proyecto`)
        - `id_asignado_a` (FK a `Usuarios.id_usuario`, NULLABLE)
- **Tabla `Equipos`:**
    - `id_equipo` (PK, UUID)
    - `nombre_equipo` (VARCHAR)
    - `descripcion` (TEXT)
- **Tabla `Usuarios_Equipos` (Tabla de unión):**
    - `id_usuario` (FK a `Usuarios.id_usuario`)
    - `id_equipo` (FK a `Equipos.id_equipo`)
    - (PK conjunta: `id_usuario`, `id_equipo`)

3.2. Base de Datos No Relacional (MongoDB):

Se utilizará para almacenar datos semi-estructurados o no estructurados, como logs de actividad, eventos de notificación y configuraciones de usuario.

- **Colección `ActividadLogs`:**
    - `timestamp` (TIMESTAMP)
    - `id_usuario` (UUID)
    - `tipo_evento` (VARCHAR)
    - `descripcion` (TEXT)
    - `datos_adicionales` (JSON/BSON)
- **Colección `Notificaciones`:**
    - `id_notificacion` (UUID)
    - `id_usuario_destino` (UUID)
    - `mensaje` (TEXT)

- `leido` (BOOLEAN)
- `fecha_creacion` (TIMESTAMP)
- `tipo_notificacion` (VARCHAR)

---

4. Diseño de la Interfaz de Usuario (UI/UX)

El diseño de la interfaz de usuario se centrará en la simplicidad, la eficiencia y la experiencia del usuario. Se seguirán los principios de Material Design para una apariencia moderna y consistente.

4.1. Componentes Clave:

- **Dashboard Personalizado:** Vista general de proyectos asignados, tareas pendientes y progreso.
- **Gestión de Proyectos:**
  - Lista de proyectos con filtros y búsqueda.
  - Vista detallada del proyecto (diagrama de Gantt simplificado, lista de tareas, miembros del equipo).
- **Gestión de Tareas:**
  - Tableros Kanban para visualizar el flujo de tareas.
  - Vistas de lista y calendario.
  - Formularios de creación y edición de tareas con campos para asignación, fecha de vencimiento, prioridad y descripción.
- **Gestión de Usuarios y Equipos:**
  - Listado y búsqueda de usuarios.
  - Creación y edición de equipos.
  - Asignación de roles y permisos.
- **Informes y Analíticas:**
  - Gráficos interactivos de progreso del proyecto.
  - Informes de rendimiento del equipo.

4.2. Flujos de Usuario Críticos (Ejemplos):

- **Creación de un Nuevo Proyecto:**
  1. Usuario inicia sesión.
  2. Navega a la sección "Proyectos".
  3. Hace clic en "Crear Nuevo Proyecto".

4. Rellena el formulario (nombre, descripción, fechas, gerente).
5. Hace clic en "Guardar Proyecto".
6. El sistema muestra el detalle del nuevo proyecto.

- **Asignación de una Tarea a un Miembro del Equipo:**
  1. Usuario navega al detalle de un proyecto.
  2. Selecciona una tarea.
  3. Hace clic en "Editar Tarea".
  4. Utiliza el selector para asignar la tarea a un usuario.
  5. Guarda los cambios.
  6. El sistema notifica al usuario asignado.

---

5. Consideraciones de Seguridad

La seguridad es una prioridad fundamental en ProjectFlow. Se implementarán las siguientes medidas:

- **Autenticación Basada en Tokens (JWT):** Para proteger las API RESTful.
- **Autorización Basada en Roles (RBAC):** Control de acceso detallado a las funcionalidades según el rol del usuario.
- **Cifrado de Contraseñas:** Uso de algoritmos de hashing seguros (ej., BCrypt) para almacenar las contraseñas.
- **Validación de Entradas:** Prevención de ataques de inyección (SQL Injection, XSS).
- **HTTPS:** Comunicación cifrada entre el cliente y el servidor.
- **Auditoría y Logging:** Registro de actividades críticas para la detección de anomalías.
- **Políticas de Sesión:** Gestión segura de sesiones de usuario.

---

6. Estrategia de Pruebas

Se adoptará una estrategia de pruebas integral para asegurar la calidad y fiabilidad del sistema.

- **Pruebas Unitarias:** Para cada microservicio y componente individual.
- **Pruebas de Integración:** Para verificar la comunicación y el correcto funcionamiento entre los microservicios.
- **Pruebas de Componente (Frontend):** Pruebas de los componentes de React de forma aislada.

- **Pruebas End-to-End (E2E):** Simulación de flujos de usuario completos a través de la interfaz de usuario.
- **Pruebas de Rendimiento y Carga:** Para evaluar la capacidad del sistema bajo diferentes volúmenes de usuarios y datos.
- **Pruebas de Seguridad:** Realización de pruebas de penetración y escaneo de vulnerabilidades.
- **Pruebas de Usabilidad:** Involucrando a usuarios reales para obtener retroalimentación sobre la experiencia del usuario.

---

7. Plan de Despliegue y Operaciones

7.1. Entornos:

- **Desarrollo:** Entornos locales de los desarrolladores.
- **Pruebas/QA:** Entorno que replica la producción para pruebas de integración y E2E.
- **Pre-producción/Staging:** Entorno final antes de producción para validación.
- **Producción:** Entorno de despliegue en la nube.

7.2. Herramientas de CI/CD:

- **GitHub Actions/GitLab CI/CD:** Para automatizar la construcción, prueba y despliegue continuo de los microservicios y el frontend.

7.3. Monitorización y Logging:

- **Prometheus y Grafana:** Para la monitorización de métricas de rendimiento y salud de los microservicios.
- **ELK Stack (Elasticsearch, Logstash, Kibana):** Para la agregación, búsqueda y visualización centralizada de logs.
- **Alertas:** Configuración de alertas para problemas críticos (errores, latencia, uso de recursos).

7.4. Escalabilidad:

- **Escalado Horizontal:** Posibilidad de añadir más instancias de microservicios según la demanda.
- **Escalado Vertical:** Aumento de recursos (CPU, RAM) para instancias individuales si es necesario.

---

8. Conclusiones y Próximos Pasos

Este informe de diseño detalla la arquitectura, las tecnologías y las consideraciones clave para el desarrollo de ProjectFlow. La adopción de una arquitectura de microservicios y tecnologías modernas proporcionará una base sólida para un sistema escalable, mantenible y robusto.

Los próximos pasos incluyen:

- Refinamiento de los modelos de datos detallados.
- Implementación del API Gateway y el servicio de autenticación como base.
- Desarrollo incremental de cada microservicio, comenzando por los más críticos.
- Desarrollo continuo del frontend en paralelo.
- Establecimiento del pipeline de CI/CD.