# PLANTILLA DE DOCUMENTO TÉCNICO INGENIERÍA DE SOFTWARE II

## 1. Introducción

## En este documento se presenta la propuesta técnica desarrollada en el marco de la actividad de retos empresariales para la asignatura Ingeniería de Software II. El equipo seleccionó el reto de Comfachocó con el fin de diseñar, desarrollar y validar una solución tecnológica innovadora para la autogestión de ausencias de los empleados.

## 2. Información del Equipo

## Nombre del Integrante Andrés Gómez Sepúlveda | Rol (Scrum Master / Dev / QA / UI/UX) | Correo Institucional agomez70@unilasallista.edu.co

## 3. Reto Seleccionado

## Nombre del reto: Sistema de Autogestión del Empleado

## Empresa retadora: Comfachocó

## Descripción breve del problema:

## La gestión manual de solicitudes de vacaciones, permisos y licencias en Comfachocó genera una carga administrativa significativa para el departamento de Recursos Humanos, dificulta la planificación de los equipos y carece de transparencia para los empleados. Se necesita una solución digital que automatice y centralice este proceso.

## 4. Requerimientos del Sistema

## • Requerimientos funcionales:

## 1. El sistema debe permitir a los empleados solicitar vacaciones, permisos y licencias a través de un portal web.

## 2. El sistema debe mostrar al empleado el saldo de días disponibles en tiempo real.

## 3. El sistema debe implementar un flujo de aprobación donde los supervisores puedan aprobar o rechazar solicitudes.

## 4. El sistema debe notificar a las partes interesadas (empleado, supervisor) sobre los cambios en el estado de una solicitud.

## 5. El sistema debe proporcionar un calendario para visualizar las ausencias del equipo y evitar solapamientos.

## • Requerimientos no funcionales:

## 1. La interfaz de usuario debe ser intuitiva, moderna y fácil de navegar.

## 2. El sistema debe ser responsivo, adaptándose a dispositivos móviles, tabletas y computadoras.

## 3. La aplicación debe tener un alto rendimiento, con tiempos de carga rápidos.

## 4. El sistema debe garantizar la seguridad y privacidad de los datos de los empleados.

## 5. Arquitectura Propuesta

## La arquitectura de la aplicación sigue un enfoque por capas, utilizando tecnologías modernas para el desarrollo web:

## - Capa de Presentación (Frontend): Construida con Next.js (React) y TypeScript, utilizando Tailwind CSS y componentes de shadcn/ui para una interfaz de usuario moderna y responsiva.

## - Capa de Lógica de Negocio (Backend): Gestionada por el entorno de Node.js que provee Next.js. La lógica de negocio, flujos de aprobación y validaciones se manejan en el servidor.

## - Capa de Datos (Persistencia): Se utiliza Prisma como ORM para interactuar con una base de datos SQLite, la cual gestiona los modelos de datos como Usuarios, Solicitudes y Políticas de Ausencia.

## 6. Diagramas UML

## • Diagrama de casos de uso:

## | Actor: Empleado |

## | Casos de Uso: |

## | - Iniciar Sesión |

## | - Enviar Solicitud de Permiso |

## | - Consultar Saldo de Días |

## | - Ver Estado de Solicitudes |

## | Actor: Supervisor |

## | Casos de Uso: |

## | - Aprobar/Rechazar Solicitud |

## | - Ver Calendario de Equipo |

## • Diagrama de clases:

## [User] { id, name, role } -> [LeaveRequest] { id, type, startDate, endDate, status }

## [User] -> [LeaveBalance] { id, availableDays, usedDays }

## [LeaveRequest] -> [LeaveType] { id, name }

## • Diagrama de secuencia (Solicitud de Permiso):

## Empleado -> Sistema: solicitarPermiso(fechas, tipo)

## Sistema -> BaseDeDatos: verificarSaldo(empleadoId)

## BaseDeDatos -> Sistema: saldoDisponible

## Sistema -> Sistema: crearSolicitud()

## Sistema -> Supervisor: notificarSolicitud(solicitud)

## Supervisor -> Sistema: aprobarSolicitud(solicitudId)

## Sistema -> BaseDeDatos: actualizarSaldo(empleadoId)

## Sistema -> Empleado: notificarAprobacion()

## 7. Mockups / Interfaces de Usuario

## La interfaz de usuario es limpia y centrada en la experiencia del usuario. El dashboard principal presenta un resumen del saldo de días y el estado de las solicitudes. Un formulario guiado permite crear nuevas solicitudes, y un calendario interactivo muestra las ausencias del equipo para facilitar la planificación. El diseño es minimalista, utilizando tarjetas y notificaciones para comunicar información de forma clara.

## 8. Plan de Pruebas

## Tipo de pruebas realizadas:

## - Pruebas Unitarias: Se verifican funciones individuales, como el cálculo de días de un permiso.

## - Pruebas de Integración: Se prueba la interacción entre el frontend y el backend, como el envío de un formulario de solicitud y su registro en la base de datos.

## - Pruebas de Usabilidad: Se realizan pruebas con usuarios finales para evaluar la facilidad de uso y la claridad de la interfaz.

## Resultados esperados y observados:

## Se espera que todos los flujos de usuario se completen sin errores y que la interfaz sea percibida como intuitiva. Las pruebas iniciales confirman que los componentes se renderizan correctamente y la lógica de negocio principal funciona según lo diseñado.

## 9. Evidencias del Desarrollo

## El código fuente, los commits y la configuración del proyecto están disponibles en el repositorio. Las capturas de pantalla del proceso de desarrollo, la estructura de carpetas y la ejecución de la aplicación en el entorno local sirven como evidencia del trabajo realizado.

## Enlace al repositorio GitHub: https://github.com/andressepu18/Reto-comfachoco/tree/main

## 10. Conclusiones

## El desarrollo de este reto permitió aplicar los principios de la ingeniería de software en un caso práctico, desde la definición de requerimientos hasta la implementación de una solución funcional. El uso de tecnologías como Next.js y Prisma facilitó un desarrollo rápido y robusto. La principal dificultad fue la interpretación de las reglas de negocio específicas de la empresa para asegurar que el sistema se ajustara a sus necesidades.

## 11. Bibliografía

## - Documentación oficial de Next.js (nextjs.org)

## - Documentación oficial de Prisma (prisma.io)

## - Documentación oficial de Tailwind CSS (tailwindcss.com)

## - Guía de Scrum (scrum.org)12. Herramientas propuestas

* Para documentar, prototipar y presentar:

| **NOMBRE DE LA HERRAMIENTA** | **DESCRIPCIÓN / USO RECOMENDADO** | **ENLACE** |
| --- | --- | --- |
| **Manus** | Plataforma para crear y publicar páginas web, portfolios o proyectos con asistencia de IA. Ideal para documentar o presentar resultados de software. | <https://manus.im> |
| **Z.ai Chat (GLM-4.6 / 4.5)** | Chat impulsado por modelos de lenguaje GLM de última generación. Permite generar documentación técnica, ideas y resúmenes de proyectos y prototipos Full-Stack. | <https://z.ai> |
| **Google AI Studio** | Entorno de desarrollo experimental de Google para probar prompts, crear modelos de IA y generar contenido técnico. | <https://aistudio.google.com> |
| **Google NotebookLM** | Herramienta de investigación con IA. Permite subir documentos, hacer resúmenes automáticos, generar ideas y conectar conceptos entre archivos. | <https://notebooklm.google> |
| **Elicit** | Asistente de investigación académica que busca, analiza y resume artículos científicos. Ideal para justificar propuestas o revisar antecedentes. | [https://elicit.com](https://elicit.com/?redirected=true) |
| **SciSpace AI Research Agent** | Plataforma con más de 150 herramientas de IA y 280 millones de artículos para búsqueda científica, resúmenes y citas automáticas. | <https://scispace.com/> |
| **Skywork AI** | Agente de IA todo-en-uno especializado en investigación profunda y generación de informes técnicos o académicos. | <https://skywork.ai> |
| **Qwen (Alibaba Cloud)** | Modelo de lenguaje multilingüe enfocado en desarrollo, documentación y generación de código con IA. | <https://qwen.ai> |
| **Claude.ai** | Asistente de IA avanzado para análisis de textos largos, generación de documentación, revisión de código y creación de resúmenes ejecutivos. | <https://claude.ai> |