República Bolivariana de Venezuela

Ministerio del Poder Popular para la Educación

Universidad Nacional de las Telecomunicaciones e Informática



Unidad Curricular: Proyecto Sociotecnológico II

Documento de Diseño

Tutora: Autores:

Yuly Delgado Orlando López V-31.332.501

Pedro Castro V-32.736.571

## 

## **Introducción**

**Propósito del proyecto**

Worklyst es una plataforma web diseñada para facilitar la gestión colaborativa de proyectos mediante inteligencia artificial. Su objetivo principal es optimizar la organización de tareas, mejorar la asignación de responsabilidades y ofrecer una experiencia visual clara y adaptable para equipos multidisciplinarios.

**Visión**

Worklyst busca convertirse en un asistente inteligente para equipos creativos y técnicos, capaz de entender el contexto de cada proyecto, proponer acciones estratégicas y facilitar la toma de decisiones. La plataforma combina diseño emocional, modularidad técnica y razonamiento automatizado para ofrecer una experiencia fluida, intuitiva y escalable.

**Enfoque basado en IA y colaboración**

El núcleo de Worklyst está impulsado por un agente IA que interpreta comandos naturales, divide proyectos en tareas, asigna miembros según habilidades y mueve tareas entre columnas. Este enfoque permite que los equipos trabajen de forma más autónoma, con menos fricción y mayor claridad. La colaboración se potencia mediante interfaces compartidas, trazabilidad de decisiones y sugerencias inteligentes que respetan el estilo de trabajo de cada usuario.

## 

## 

## **Arquitectura del Software**

La arquitectura de Worklyst está diseñada para ser modular, escalable y adaptable a equipos multidisciplinarios. Se compone de cuatro capas principales que se comunican entre sí de forma eficiente:

🔹 **Componentes principales**

**Frontend**: desarrollado con Next.js, Tailwind CSS y TypeScript, lo que permite una experiencia rápida, responsiva y altamente personalizable.

**Backend**: construido con Node.js y Express, encargado de manejar la lógica de negocio, autenticación, validaciones y conexión con la base de datos.

**Agente IA**: implementado como microservicio en Python usando LangChain y Groq. Este agente interpreta comandos, divide proyectos, asigna tareas y razona sobre el estado del sistema.

**Base de datos**: gestionada con Supabase (PostgreSQL), que almacena usuarios, proyectos, tareas, historial y relaciones entre entidades.

🔹 **Comunicación entre componentes**

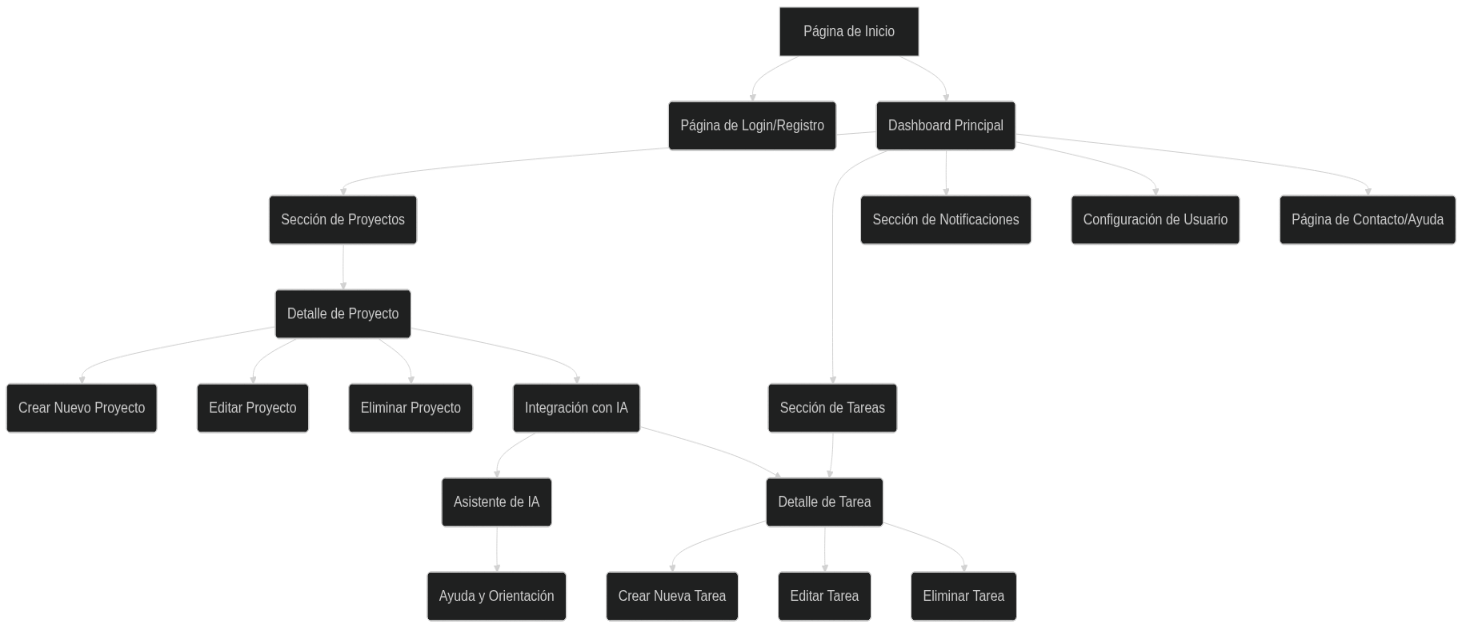
El frontend se comunica con el backend mediante API REST, enviando comandos, consultas y actualizaciones.

El agente IA se integra como microservicio separado, con posibilidad de ser embebido directamente en el backend si la carga lo requiere.

Las respuestas del agente se presentan en el frontend mediante un modal conversacional y un chat de proyecto, permitiendo interacción directa y contextual.

La base de datos se consulta desde el backend y el agente IA, permitiendo razonamiento basado en datos reales.

🔹 **Diagrama de arquitectura**

****

**Diseño de la Escalabilidad**

Worklyst está pensado para crecer con el equipo que lo utiliza. Aunque el entorno inicial es de prueba con aproximadamente 50 usuarios activos, la arquitectura permite escalar sin comprometer rendimiento ni claridad visual.

🔹 **Estrategias de escalabilidad**

**Microservicios desacoplados**: el agente IA puede escalar de forma independiente, permitiendo balanceo de carga y procesamiento paralelo.

**Escalabilidad horizontal**: el backend y el agente IA pueden replicarse en múltiples instancias si la demanda crece.

**Componentes desacoplados en frontend**: cada vista (Dashboard, Proyecto, Modal IA) puede cargarse de forma independiente, facilitando lazy loading.

**Renderizado eficiente**: se prioriza la carga de componentes clave y se minimiza el uso de efectos innecesarios.

🔹 **Optimización visual y técnica**

Lazy loading se aplicará en componentes e imágenes, especialmente en vistas como Dashboard y Modal IA.

Paginación se implementará en secciones con alto volumen de datos como proyectos e historial, evitando saturación de la interfaz.

**Cache selectivo**: se utilizará Supabase para cachear consultas frecuentes y localStorage para persistencia ligera en el navegador (por ejemplo, preferencias de usuario o estado de sesión).

## **Diseño de la Interfaz de Usuario**

**Principios de diseño**

La interfaz de Worklyst se basa en tres pilares: claridad visual, accesibilidad funcional y coherencia emocional. Cada vista está pensada para facilitar la lectura, reducir la carga cognitiva y adaptarse al modo de trabajo del usuario.

**Jerarquía visual y navegación**

La navegación se organiza en torno a cuatro vistas principales: Home, Dashboard, Proyecto y Modal IA.

* **Inicio** presenta una bienvenida contextual y accesos rápidos.
* **Dashboard** muestra el estado general de los proyectos y tareas.
* **Proyecto** permite visualizar y editar tareas, miembros y columnas.
* **Chat** Permite a los usuarios comunicarse mediante canales de proyecto
* **Modal IA** actúa como interfaz conversacional con el agente inteligente.

La jerarquía visual prioriza la información relevante, con componentes desacoplados que permiten personalización y escalabilidad.

## **Estilo Visual**

La interfaz de Worklyst utiliza una estética funcional y emocionalmente coherente, pensada para facilitar la lectura, destacar prioridades y adaptarse a distintos contextos de uso. El sistema de diseño se apoya en una paleta de colores suaves con acentos estratégicos, y una tipografía moderna que refuerza la claridad y la personalidad de la plataforma.

**Tipografía**

La fuente principal utilizada en Worklyst es **Poppins**, una tipografía sans-serif moderna que combina legibilidad con personalidad. Su estructura geométrica y versatilidad la hacen ideal para interfaces limpias, jerarquizadas y adaptables a distintos tamaños de pantalla.

### 

### **Paleta de colores**

La paleta se compone de tonos neutros para estructura y fondo, y colores de acento para representar estados, prioridades y acciones clave. A continuación, se detallan los colores principales junto con sus equivalentes en Tailwind CSS:

|  |  |  |
| --- | --- | --- |
| Color Base | Hexadecimal | Clase Tailwind CSS |
| Blanco | #FFFFFF | white |
| Gris Claro | #F5F5F5 | gray-100 |
| Gris Oscuro | #374151 | gray-700 |
| Azul | #3B82F6 | blue-500 |
| Verde | #10B9B1 | green-500 |
| Rojo | #EF4444 | red-500 |
| Amarillo | #FBBF24 | yellow-400 |