Aplicación de Gestión de tareas tipo Trello

Esneider Cadavid David

Agosto de 2025

Consideraciones

El proyecto TrainIT se desarrolló en el marco de una simulación laboral con una duración de cinco meses, dentro de un entorno colaborativo altamente estructurado. La experiencia se destacó por la solidez del cuerpo organizativo, conformado por profesionales con diferentes niveles de seniority y áreas de especialización, lo que permitió una dinámica de trabajo similar a la de un entorno real de desarrollo tecnológico.

La simulación incluyó equipos de Project Management, QA, UX/UI Design, y Development, coordinados mediante metodologías ágiles y herramientas de comunicación internas que garantizaban el flujo constante de información y feedback. Este enfoque integral permitió no solo el crecimiento técnico de los participantes, sino también el fortalecimiento de habilidades de colaboración, planificación y resolución de problemas en entornos multidisciplinarios.

Planteamiento del Software

TrainIT es una aplicación web orientada a la **gestión y administración de tareas**, inspirada en la dinámica de tableros interactivos como **Trell** o los **b ards de GitHub**. Su diseño permite crear espacios de trabajo donde los usuarios pueden registrar tareas, asignar responsables, definir prioridades y mover ítems entre columnas mediante una interfaz *drag and drop*, reflejando los distintos estados o fases de un proyecto.

La aplicación cuenta con un sistema de **autenticación y registr de usuari s**, panel de **creación y administración de tabler s**, y manejo de **miembr s y permis s**. Cada tarea incluye campos de descripción, estado, fecha, nivel de importancia y responsables. Además, se integran funcionalidades de **n tificaci nes y mensajería**, orientadas a mantener una comunicación fluida entre los miembros del equipo ante cualquier cambio o actualización dentro de los tableros.

El desarrollo se apoyó en un equipo de diseño especializado que definió la maquetación, los prototipos visuales y los modelos de interacción, permitiendo que la implementación técnica siguiera lineamientos claros desde el inicio. La estructura del sistema, su enfoque modular y la conexión con el backend garantizan la persistencia y sincronización de los datos entre las distintas acciones del usuario.

Stack Tecnológico

El desarrollo de TrainIT se estructuró bajo una arquitectura **MVC** (*Model-View-Controller*), dividiendo claramente las responsabilidades entre la interfaz, la lógica de negocio y la gestión de datos. La solución integró un ecosistema moderno de herramientas y tecnologías orientadas tanto a la eficiencia en el desarrollo como a la escalabilidad del producto.

En el Frontend, se utilizó **Next.js** como framework principal, aprovechando su renderizado híbrido y enrutamiento automatizado. Sobre esta base se construyó una interfaz dinámica con **React**, implementando un sistema de tipado estático mediante **TypeScript**. Para la gestión de estilos se adoptó **Tailwind** CSS, con una arquitectura de diseño basada en variables de color y un sistema de temas derivados de la librería System Color, lo que permitió mantener la coherencia visual en todo el entorno.

La interacción del usuario se enriqueció con librerías como **DndKit**, responsable del sistema *drag and drop* en los tableros, y **React Date Picker** junto a **date-fns**, utilizadas para la selección y manipulación de fechas. La gestión global del estado se resolvió mediante **Zustand**, una librería ligera que optimizó el rendimiento de la aplicación.

En el Backend, el proyecto se implementó en Python, utilizando **Flask** como framework principal debido a su flexibilidad y ligereza. Se integraron extensiones como **Flask-Migrate** y **Alembic** para el control de migraciones, **Flask-JWT-Extended** y **Bcrypt** para la autenticación y cifrado seguro de contraseñas, **Flask-CORS** para la comunicación entre dominios, y Flasker para la documentación automática de la API. La base de datos se gestionó con PostgreSQL, garantizando integridad y robustez en el manejo de la información.

Además, el sistema incorporó servicios externos para ampliar sus funcionalidades: Socket.IO y **Pusher** para comunicación en tiempo real y actualización de eventos, **SendGrid** para el envío automatizado de correos electrónicos, y **Cloudinary** como servicio de almacenamiento y entrega optimizada de imágenes.

En cuanto a la colaboración y flujo de trabajo, el proyecto se apoyó en **Discord** como canal principal de coordinación, **GitHub** para la gestión de repositorios remotos y control de versiones con **Git**, y **Figma** como herramienta central de diseño y maquetación, en estrecha comunicación con los equipos de UX/UI y Frontend.

Contribución Personal

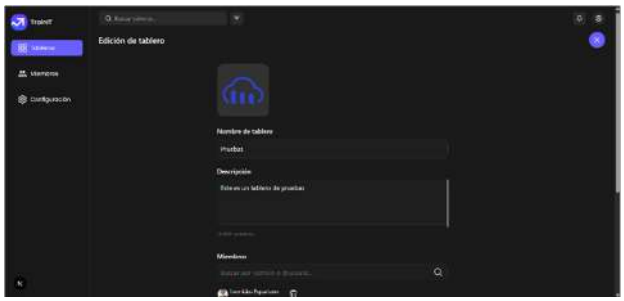
Mi participación en **TrainIT** se desarrolló bajo el rol de **Full Stack Developer**, con un enfoque predominante en el **Frontend**, aunque también realicé contribuciones sustanciales en el **Backend** durante las etapas iniciales e intermedias del proyecto.

En las primeras fases, colaboré en la definición y aplicación de la **arquitectura de estilos basada en System Color**, propuesta por el equipo de UX/UI. Esta estructura establecía una jerarquía de colores primitivos y variables globales integradas con **Tailwind CSS**, lo que permitió mantener la coherencia visual y la escalabilidad del diseño en el desarrollo de los componentes. También participé en la

organización inicial de la arquitectura frontend y en la estructuración de carpetas y patrones de diseño, siguiendo el enfoque **MVC** definido para el proyecto.

Posteriormente, mi labor se centró en la construcción de **componentes reutilizables** y secciones específicas del sistema, como el *Home* y elementos visuales complementarios (por ejemplo, el *footer*). Durante esta etapa, mi trabajo consistió en implementar componentes dinámicos, manejar la lógica de estado, y conectar las interfaces con el backend mediante **fetching de datos** y tipado en **TypeScript**, manteniendo la compatibilidad con la lógica de autenticación y gestión de usuarios.

En una fase más avanzada, asumí la responsabilidad de desarrollar y optimizar el **módulo de edición de tableros y tarjetas**, implementando la funcionalidad de actualización de datos desde el backend y su sincronización con la interfaz. Esta tarea implicó trabajar con endpoints, formularios dinámicos y control de estados, integrando tanto la lógica del servidor como la del cliente.



Finalmente, participé en el desarrollo del **sistema de arrastre y soltado (drag & drop)**, uno de los componentes más interactivos del proyecto. La implementación se realizó con la librería **DndKit**, cuidando aspectos de rendimiento como la prevención de re-renderizados innecesarios, el control de bucles infinitos y la optimización de estados locales y globales. Este componente se conectó al backend para mantener la persistencia de la posición y el estado de las tareas, consolidando una funcionalidad clave dentro de la experiencia de usuario.