NeoCare Health

Plataforma Interna de Gestión

Documentación Académica del Proyecto

Equipo de desarrollo

Equipo Gamma

- Manuel Forment
- Francisco Santos
- Luis Fernando Ayala

Índice

1. Introducción

El presente documento describe el desarrollo de una plataforma web interna destinada a la gestión de proyectos y tareas del Departamento de Innovación de la empresa NeoCare Health. El proyecto se ha realizado como una entrega académica, simulando un entorno real de trabajo en una empresa tecnológica, con el objetivo de aplicar conocimientos de desarrollo Full Stack en un caso práctico y realista.

La aplicación desarrollada permite organizar tareas mediante un tablero Kanban, registrar el tiempo dedicado a cada actividad y generar informes semanales (exportandolo a archivos csv si fuese necesario) que facilitan la toma de decisiones y el seguimiento del progreso de los proyectos.

El desarrollo del sistema se ha llevado a cabo a lo largo de siete semanas, siguiendo una planificación progresiva por fases, incorporando funcionalidades de forma incremental hasta obtener una solución funcional y estable.

2. Contexto del problema y objetivos

NeoCare Health gestiona múltiples proyectos de innovación de manera simultánea. Antes del desarrollo de esta plataforma, la organización interna del trabajo se realizaba mediante herramientas dispersas como hojas de cálculo, correos electrónicos o aplicaciones externas, lo que dificultaba el seguimiento del progreso y el control del esfuerzo dedicado.

Ante esta situación, se plantea como objetivo principal el desarrollo de una herramienta interna, ligera y unificada que permita:

- Organizar tareas de forma visual y estructurada.
- Conocer el estado real de los proyectos en todo momento.
- Registrar las horas de trabajo dedicadas a cada tarea.
- Obtener informes semanales claros y exportables.
- Mejorar la coordinación del equipo de innovación.

La solución propuesta busca cubrir estas necesidades mediante una aplicación web moderna, accesible y fácil de utilizar.

3. Descripción general de la solución con el proyecto

La solución desarrollada consiste en una plataforma web que integra un frontend interactivo con un backend robusto y una base de datos relacional. El sistema permite a los usuarios autenticarse, gestionar tareas en un tablero Kanban por medio de tarjetas; las cuales se modifican segun el estado trabajo, registrar horas de trabajo en cada tarjeta y global ademas de consultar informes semanales generados automáticamente y extraerlos.

La aplicación ha sido diseñada para simular un producto interno real, priorizando la claridad de uso, la estabilidad y la correcta organización de la información individual.

4. Arquitectura general del sistema

La arquitectura del sistema se divide en tres bloques principales:

Frontend

El frontend se encarga de la interfaz de usuario y de la interacción directa con el sistema. Permite visualizar el tablero Kanban, gestionar tarjetas, mover tareas entre columnas, registrar horas y consultar informes.

Backend

El backend proporciona una API REST desarrollada con FastAPI. Se encarga de la lógica de negocio, la validación de datos, la seguridad mediante autenticación JWT y la comunicación con la base de datos.

Base de datos

La información se almacena en una base de datos relacional PostgreSQL, donde se gestionan usuarios, tarjetas, registros de horas y relaciones entre los distintos elementos del sistema.

La comunicación entre frontend y backend se realiza mediante peticiones HTTP seguras, utilizando tokens JWT para garantizar el acceso autorizado.

5. Funcionalidades implementadas

5.1 Autenticación

El sistema dispone de un sistema de autenticación segura basado en JSON Web Tokens (JWT). Cada usuario debe iniciar sesión (previamente necesita registrarse como usuario) para acceder a la plataforma, garantizando que solo los usuarios autorizados puedan gestionar la información.

5.2 Tablero Kanban

La aplicación incluye un tablero Kanban compuesto por tres columnas principales:

- Por hacer
- En curso
- Hecho

Las tareas se representan mediante tarjetas que se muestran en la columna correspondiente según su estado.

5.3 Gestión de tarjetas

El sistema permite crear, editar y visualizar tarjetas. Cada tarjeta puede incluir un título, una descripción y una fecha límite. Además, se muestran avisos visuales cuando una tarea se aproxima a su fecha de vencimiento y el numero total de horas realizadas en esa tarea.

5.4 Movimiento de tarjetas (Drag & Drop)

Las tarjetas pueden moverse entre columnas mediante un sistema de arrastrar y soltar (Drag & Drop). Este movimiento se refleja de forma inmediata tanto en la interfaz como en la base de datos, manteniendo la coherencia del estado del tablero.

5.5 Registro de horas

Cada usuario puede registrar las horas trabajadas en una tarjeta concreta, indicando la fecha y el tiempo dedicado. El sistema permite consultar las horas registradas por tarjeta y obtener una vista semanal de las horas trabajadas por cada usuario.

5.6 Informe semanal

La plataforma genera un informe semanal que muestra:

- Tareas completadas durante la semana.
- Tareas vencidas.
- Tareas nuevas.

- Horas totales trabajadas por persona.
- Horas acumuladas por tarjeta.

Este informe proporciona una visión clara del progreso y del esfuerzo invertido por el equipo.

5.7 Exportación de datos

Los informes semanales pueden exportarse en formato CSV, tanto de forma individual como agrupados por tarjetas, facilitando su análisis externo o su presentación a dirección.

5.8 Funcionalidades extra

Además de las funcionalidades principales, se han implementado los siguientes extras:

- Sistema de búsqueda de tarjetas.
- Etiquetas para clasificar tareas.
- Contabilidad detallada de horas por tarjeta y por semana.
- Avisos visuales por fecha límite.
- Exportación avanzada de informes en CSV.

6. Base de datos

La base de datos se ha diseñado siguiendo un modelo relacional, (se ha realizado con el sistema de gestión de datos PostgreSQL) con entidades principales como usuarios, tarjetas y registros de horas. Las relaciones entre tablas permiten asociar tarjetas a usuarios y vincular múltiples registros de horas a una misma tarea.

El diseño adoptado facilita la obtención de métricas agregadas y garantiza la integridad de los datos almacenados.

7. Control de versiones y repositorio

El proyecto utiliza Git como sistema de control de versiones y GitHub como plataforma de alojamiento del código fuente. El repositorio se encuentra organizado en dos partes

principales: frontend y backend, cada una con su estructura de carpetas claramente definida.

El uso de GitHub ha permitido llevar un seguimiento del progreso del proyecto, registrar cambios de forma incremental y mantener un historial claro de la evolución del sistema.

URL principal del proyecto en Git Hub : https://github.com/Pyro14/Gamma

8. Instalación y ejecución del proyecto

Para ejecutar el proyecto en un entorno local es necesario disponer de una serie de herramientas y tecnologías que permiten el correcto funcionamiento tanto del frontend como del backend.

Requisitos previos

- Node.js (entorno de ejecución para el frontend desarrollado con React y Vite).
- Python (lenguaje utilizado para el desarrollo del backend con FastAPI).
- PostgreSQL (sistema gestor de base de datos relacional).
- Gestores de paquetes:
    - npm, para la instalación de dependencias del frontend.
    - pip, para la instalación de dependencias del backend.

Adicionalmente, el proyecto hace uso de diversas librerías y herramientas, entre las que destacan React y Vite para la interfaz de usuario, FastAPI para la creación de la API, JWT para la autenticación segura, dnd-kit para la implementación del sistema de Drag & Drop y librerías como pandas y numpy para el tratamiento y agregación de datos en los informes.

Proceso general de instalación

El proceso de instalación del proyecto se compone de las siguientes fases:

1. Clonación del repositorio desde la plataforma GitHub.
2. Instalación de las dependencias del frontend mediante el gestor de paquetes npm.
3. Instalación de las dependencias del backend mediante el gestor de paquetes pip.

4. Configuración de las variables de entorno necesarias para la conexión con la base de datos y la autenticación.

5. Puesta en marcha del backend y del frontend en entornos locales independientes.

Una vez completados estos pasos, la aplicación queda accesible desde el navegador, permitiendo el uso completo de todas las funcionalidades implementadas.

La documentación incluida en el repositorio detalla de forma más específica los comandos y configuraciones necesarias para completar este proceso correctamente.

9. Decisiones técnicas

La elección del stack tecnológico se ha realizado teniendo en cuenta criterios de simplicidad, estabilidad y aprendizaje:

- FastAPI se ha seleccionado por su rapidez, claridad y validación automática de datos.
- React se ha utilizado para crear una interfaz dinámica y modular.
- Pandas, por su rapidez y sencillez de obtener y analizar datos.
- PostgreSQL ofrece fiabilidad y consistencia en el almacenamiento de datos.
- JWT garantiza un sistema de autenticación seguro.
- dnd-kit permite implementar un sistema de Drag & Drop moderno y fluido.
- Vite se ha implementado como motor de construcción por su velocidad de arranque instantánea y eficiencia en el flujo de desarrollo.

Estas decisiones permiten obtener una aplicación coherente, mantenible y alineada con estándares actuales de desarrollo web.

10. Conclusiones

El desarrollo de esta plataforma ha permitido reproducir de forma realista el ciclo completo de un proyecto de desarrollo Full Stack. Se ha conseguido implementar una solución funcional que responde a las necesidades planteadas por NeoCare Health, ofreciendo una herramienta útil para la organización interna y el seguimiento del trabajo.

A lo largo del proyecto se han afrontado retos técnicos relacionados con la sincronización de estados, la gestión de datos, visualización de tarjetas y la organización del sistema, los cuales han sido resueltos mediante una planificación progresiva y una correcta división del trabajo. A la vez hemos aprendido a usar programas nuevos como Visual Studio Code, Git y librerías nuevas como FastAPI, Pandas, dnd-kit, Numpy, React.

Como conclusión final, el proyecto ha supuesto una experiencia formativa valiosa, permitiendo aplicar conocimientos teóricos en un contexto práctico y comprender el funcionamiento global de una aplicación web profesional.