Pulse

Tejera Santana Adoney

2ºDAW

Sumario

Descripción

Pulse es una aplicación de gestión de proyectos colaborativos orientada al entorno del desarrollo de software y equipos técnicos. Su objetivo es facilitar la organización y seguimiento del trabajo en equipo a través de una estructura flexible y centrada en tareas.

Con Pulse, los usuarios pueden crear proyectos, invitar a otros colaboradores y asignar tareas de forma individual o grupal. Cada tarea puede estar vinculada a varios usuarios, y cada uno mantiene su propio estado de progreso: To Do, In Progress o Done. El sistema monitoriza estos estados individualmente y, una vez que todos los usuarios han marcado su parte como Done, la tarea entra automáticamente en fase de Review. Esta fase permite a administradores o al propietario del proyecto validar y cerrar oficialmente la tarea, garantizando así un control de calidad antes de su finalización definitiva.

Desarrollo del proyecto

Análisis del proyecto:

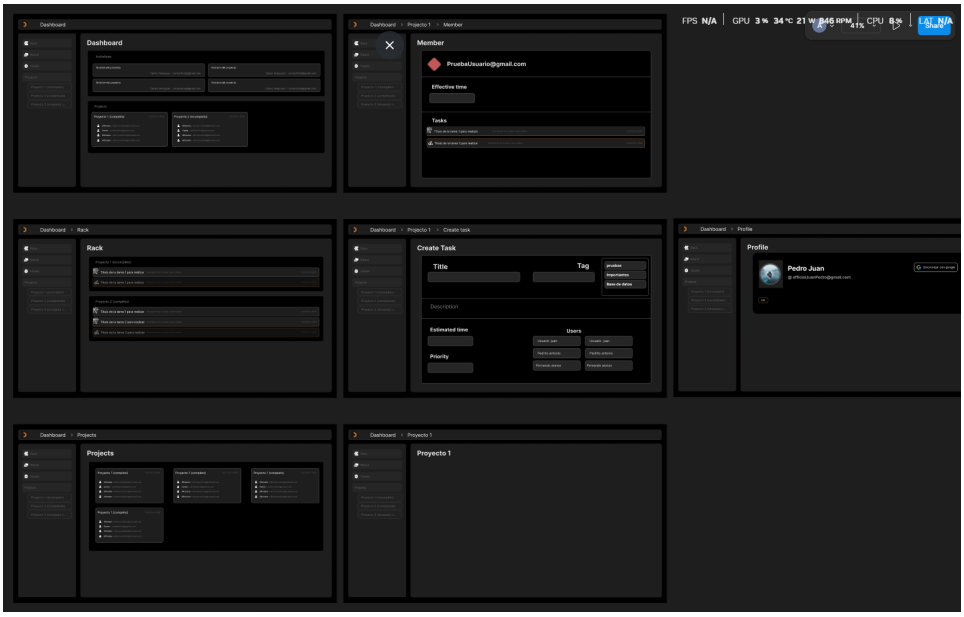
Lo primero de todo fue el planteamiento del proyecto, analizando qué acciones se podrá hacer en la aplicación, complejidad y herramientas que se utilizarán.

Diseño de la base de datos:

Se comenzó el desarrollo definiendo la estructura de la base de datos, basándose en el funcionamiento previsto del proyecto, asegurando una base sólida para las relaciones entre usuarios, proyectos, tareas y estados.

Boceto funcional de la interfaz:

Se realizó un primer boceto de la interfaz sin entrar en detalles visuales, centrado en identificar las acciones clave que tendría la aplicación y en cómo sería su flujo de uso.



Desarrollo de la API:

Se implementó la API con foco en un funcionamiento sólido, integrando autenticación, validaciones y todas las operaciones básicas necesarias para manejar usuarios, proyectos y tareas.

Desarrollo del FrontEnd:

Una vez que la API estuvo casi completa, se inició el desarrollo del FrontEnd. Se realizaron pruebas de autenticación y se integraron funcionalidades progresivamente, añadiendo nuevas rutas a la API según se requerían, procurando además optimizar las peticiones para no sobrecargarlo.

Una vez tenía la mayor parte del proyecto se integró autenticación mediante cuenta de Google.

Tecnologías empleadas

Angular: Utilizado por su capacidad para estructurar aplicaciones FrontEnd de forma eficiente y porque fue una tecnología impartida en el curso.

MySQL

Laravel: Facilita el desarrollo de APIs robustas y seguras, además de haber sido trabajado durante el curso.

Google API: Permite una autenticación rápida y segura mediante cuentas de Google, mejorando la experiencia del usuario.

Axios: Se usó para las peticiones HTTP desde el FrontEnd por su simplicidad y funciones integradas que optimizan la comunicación con la API.

Integración y coste adicional

Identificación de necesidades del sector productivo

Las empresas, especialmente en el sector tecnológico, requieren herramientas que faciliten la **gestión de proyectos, la colaboración entre equipos y el control del estado de las tareas**. Muchas soluciones existentes son complejas o tienen costes elevados, por lo que una aplicación como Pulse, centrada en la sencillez,

visibilidad del progreso y control por parte del administrador, responde directamente a esas necesidades.

Aportación y beneficios a la empresa

Pulse permite a una empresa:

- Organizar proyectos y tareas de forma clara.
- Asignar responsabilidades a varios usuarios por tarea.
- Hacer seguimiento individual y colectivo del estado del trabajo.
- Disminuir errores gracias a la revisión obligatoria por parte de administradores antes de marcar una tarea como finalizada.

Costes de hardware y software

Para el desarrollo y despliegue de Pulse se requiere lo siguiente:

- **Hardware**: Ordenador básico (mínimo 4 GB RAM, CPU de 1 núcleo)
- **Software**:
  - **Angular, Laravel, MySQL, Axios** → Open Source (sin coste).
  - **Google API para autenticación** → gratuita

Comparativa y elección de precios

Elemento	Opciones	Precio aproximado	Elección	Justificación
Dominio	GoDaddy, Namecheap	10–15 €/año	Namecheap	Precio competitivo y fiable
Hosting/VPS	Heroku, Vercel, OVH, DigitalOcean	5–20 €/mes	DigitalOcean (VPS)	Mayor control, escalabilidad
Copias de seguridad	Integradas en VPS o con servicios externos (p.ej., Backblaze)	2–5 €/mes	Incluidas en VPS	Gestión centralizada, coste menor

Elemento	Opciones	Precio aproximado	Elección	Justificación
Licencias	No necesarias	0 €	—	Todo el software es libre o gratuito

Referencias

Documentación

- Documentación de Laravel.
- Documentación de Google oAuth (Sesiones con Google)
- Documentación de Angular

¿Qué es?

Pulse es una aplicación web para la gestión de proyectos y tareas colaborativas. Permite crear proyectos, invitar a usuarios, asignar tareas y hacer seguimiento del progreso de cada miembro del equipo.

Comienzo

Para comenzar a utilizar la aplicación deberá iniciar sesión o, si no lo tiene, crear una nueva. En ambos casos se podrá utilizar una cuenta de Google.

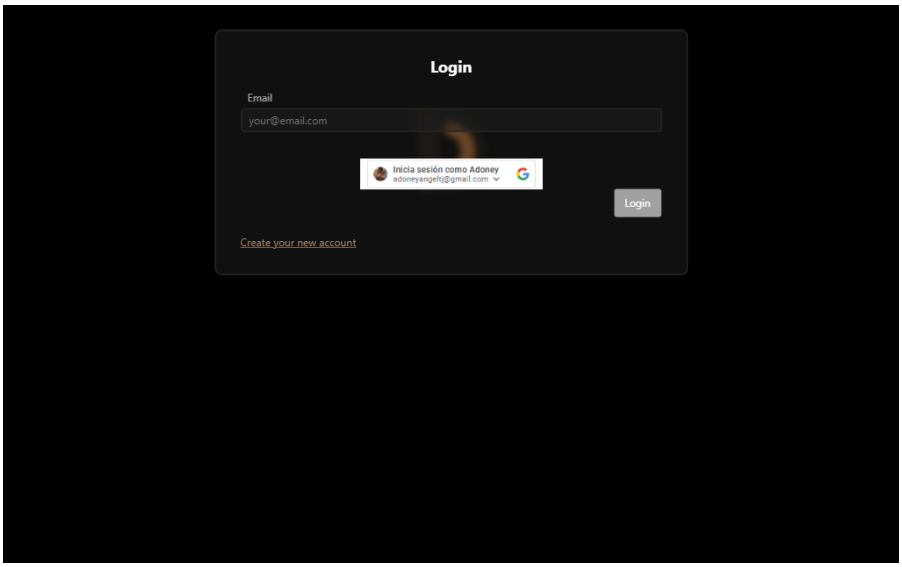


Figura 1: Login

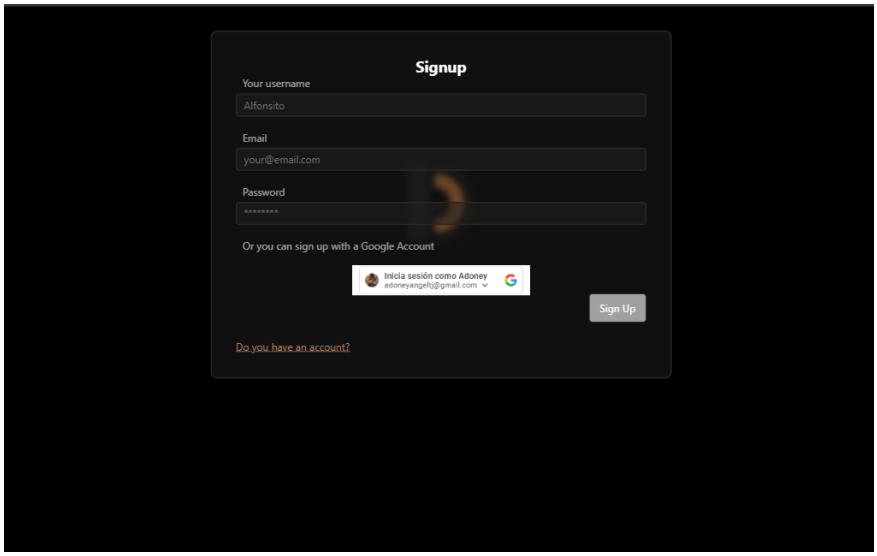
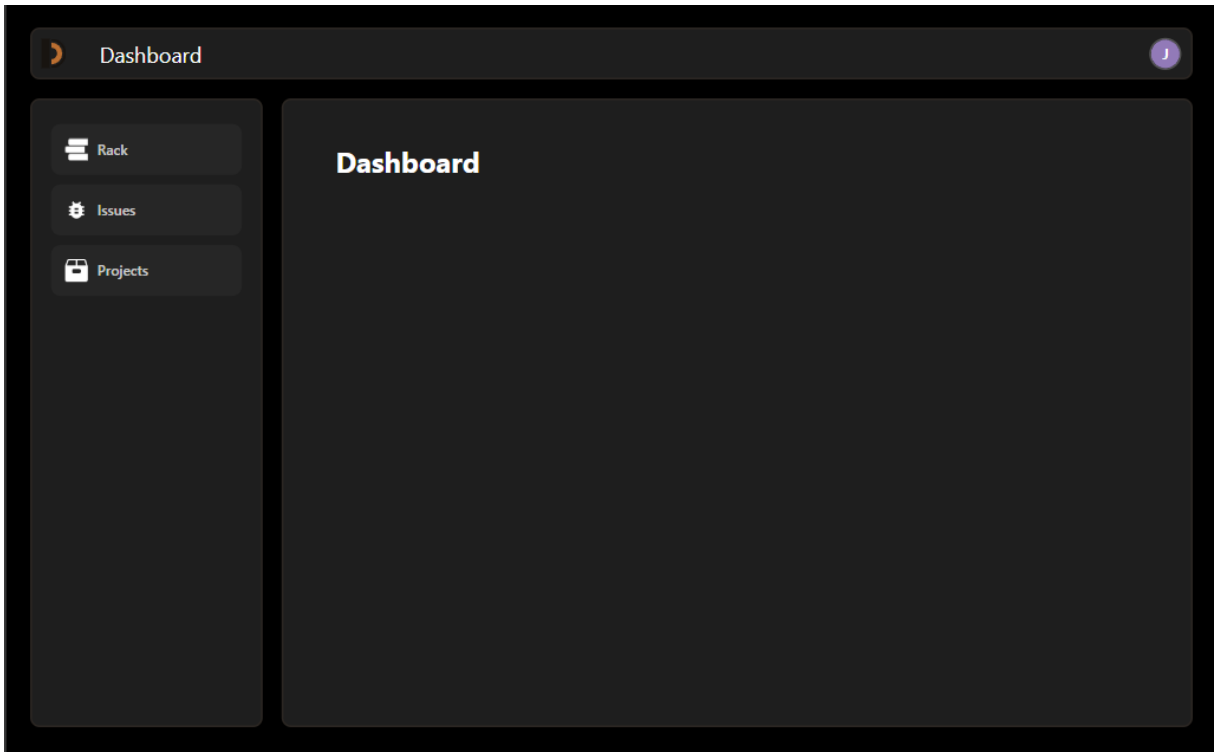


Figura 2: Creación de cuenta

Página principal

Una vez iniciado sesión, se redireccionará a la página principal (Dashboard).



En este punto puede acceder a su perfil de usuario, cerrar sesión, cambar su foto de perfil o enlazar una cuenta de Google.

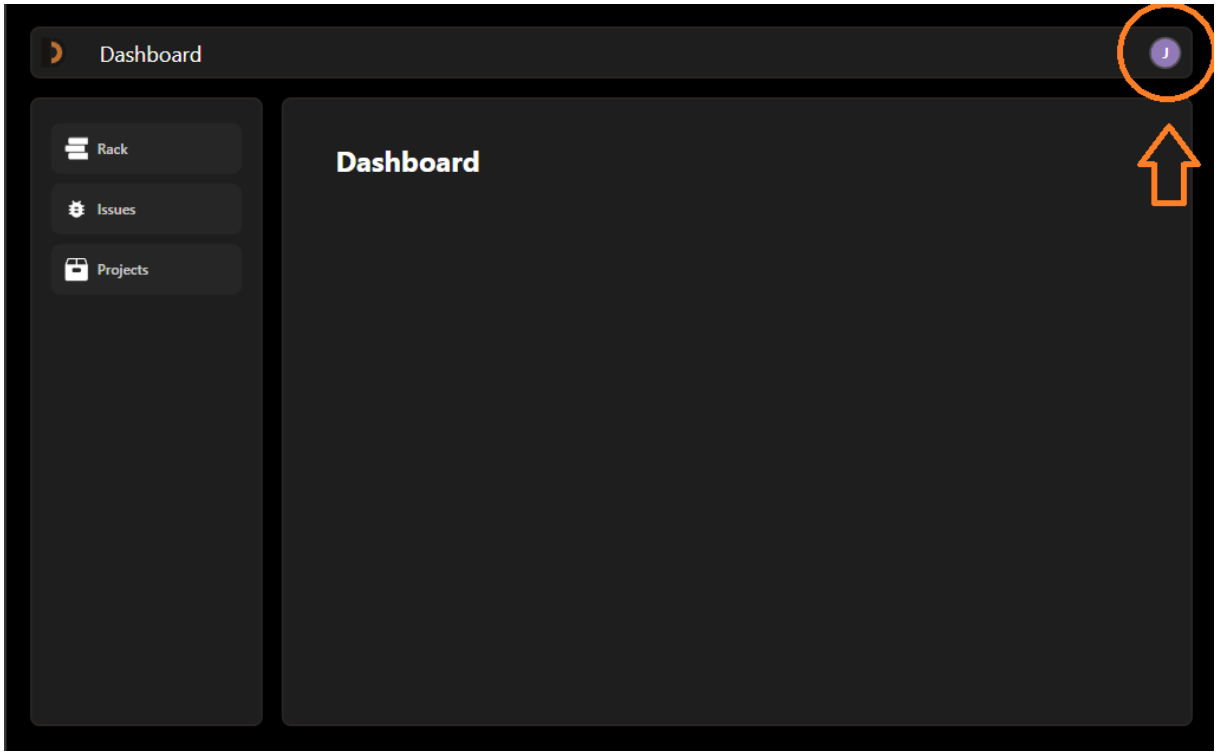


Figura 3: Acceso a su perfil

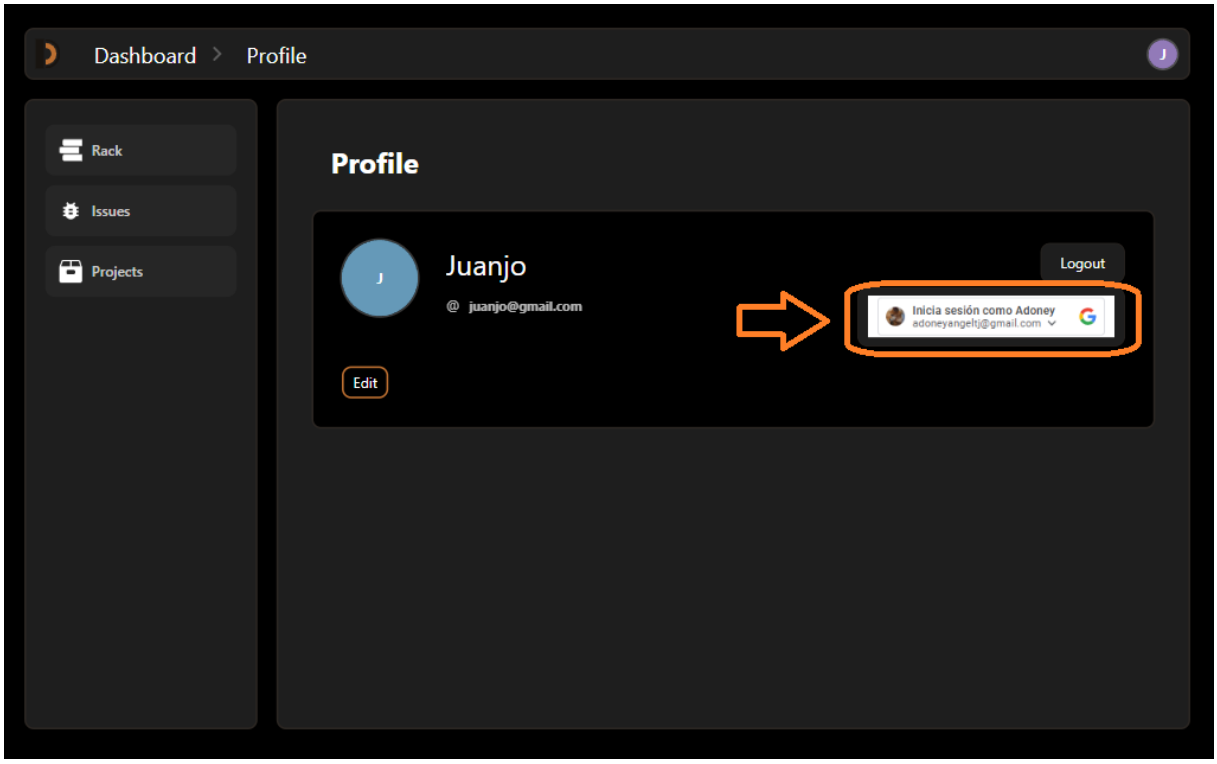


Figura 4: Enlazar cuenta de Google

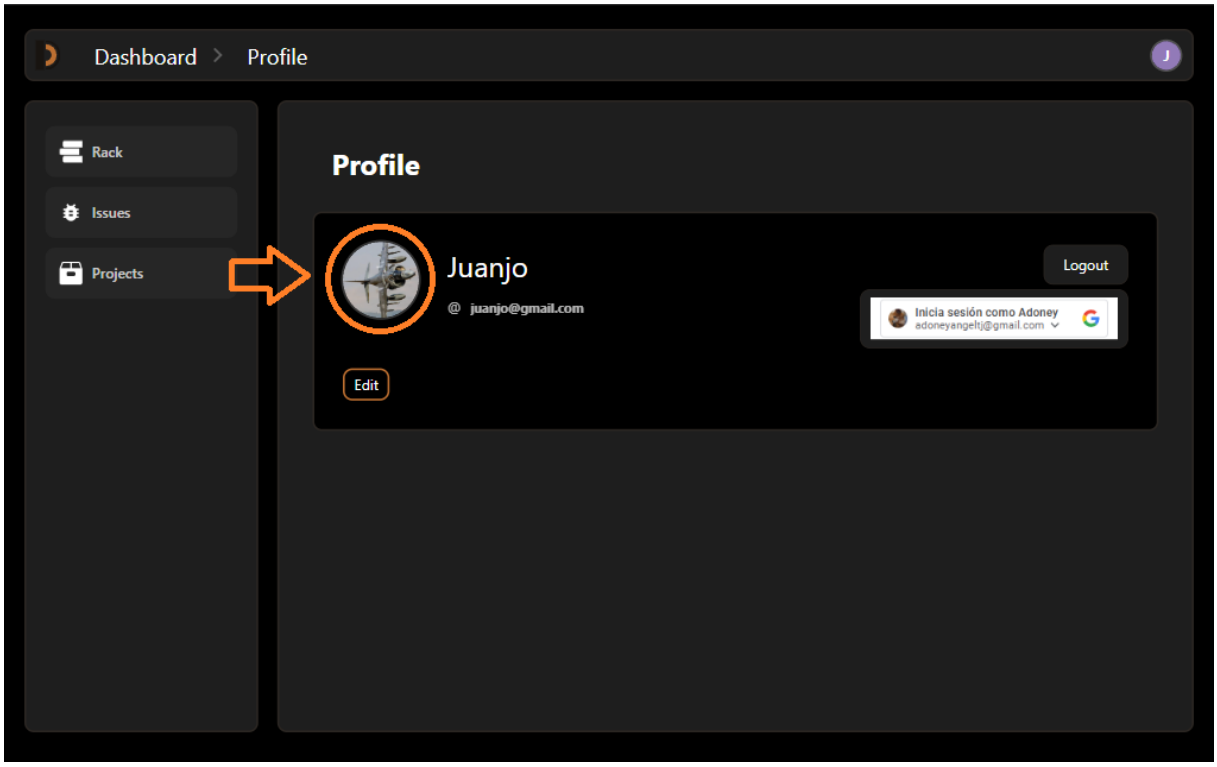


Figura 5: Cambiar foto de perfil

Creación de un nuevo proyecto

Ahora mismo no hay ningún proyecto creado ya que somos un nuevo usuario, por lo que procedemos a crear un nuevo proyecto.

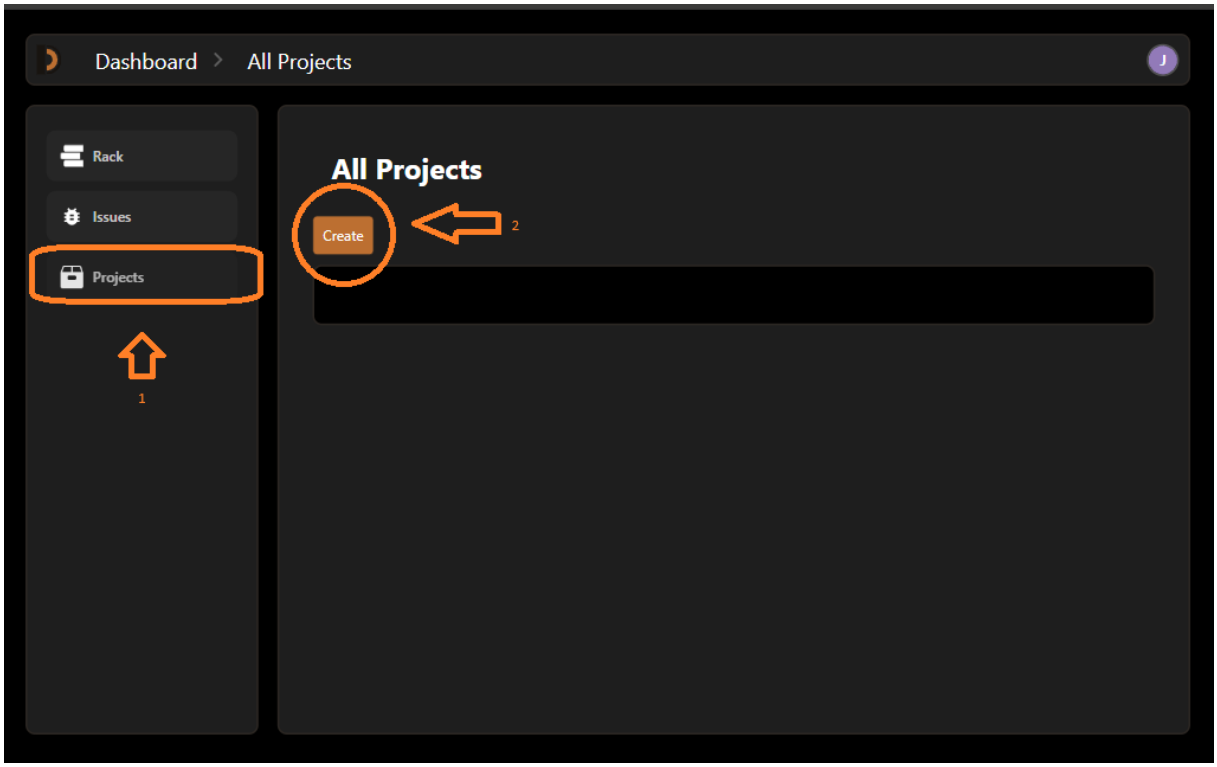


Figura 6: Creación de nuevo proyecto

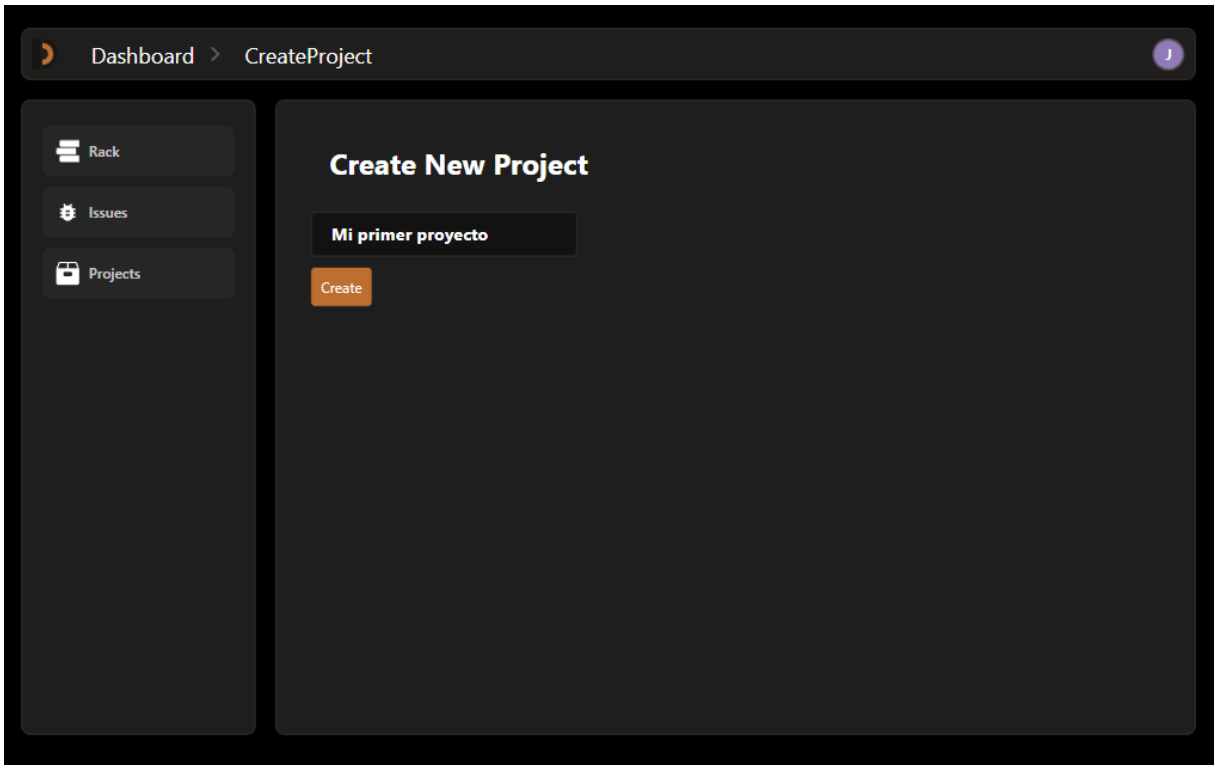


Figura 7: Nombre del proyecto

Una vez creado el proyecto, si vamos a “Dashboard”, podremos ver el nuevo proyecto creado.

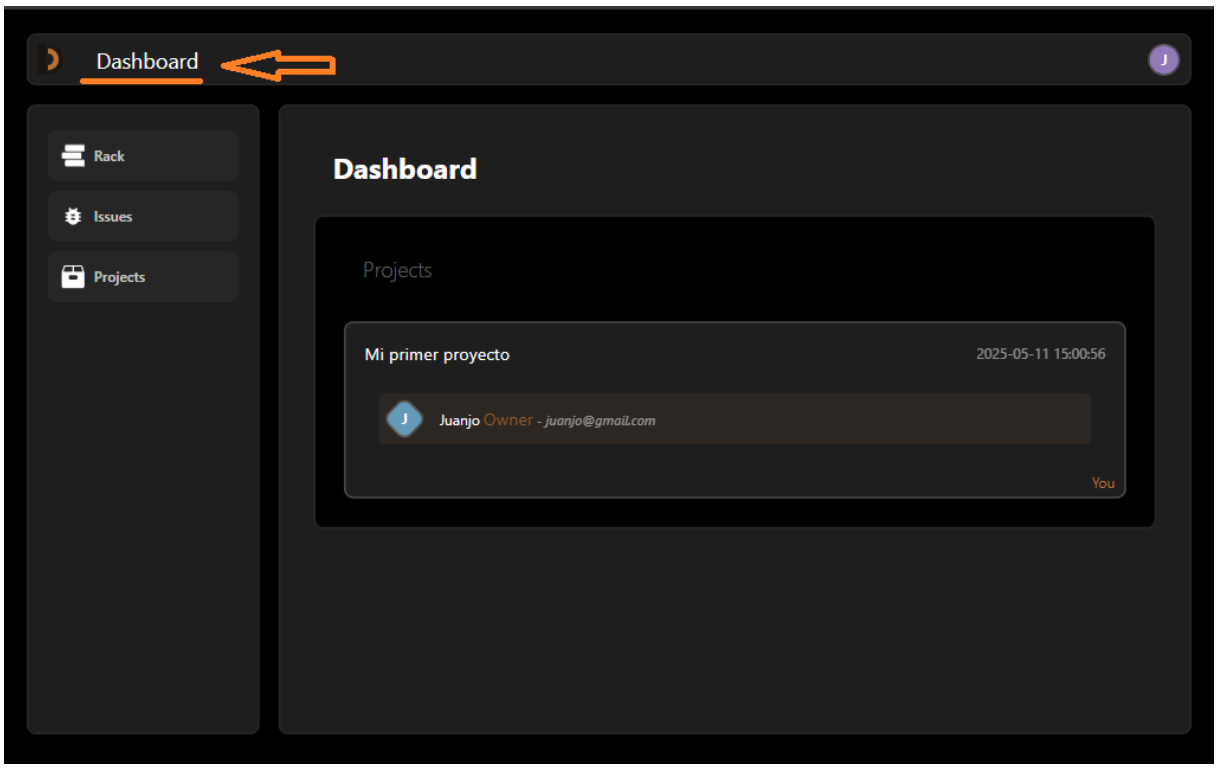


Figura 8: Nuevo proyecto creado

Adición de miembros en proyectos

Una vez creamos el proyecto, lo mejor es empezar a añadir a los miembros del mismo. Para ello accedemos al proyecto (haciendo click en él).

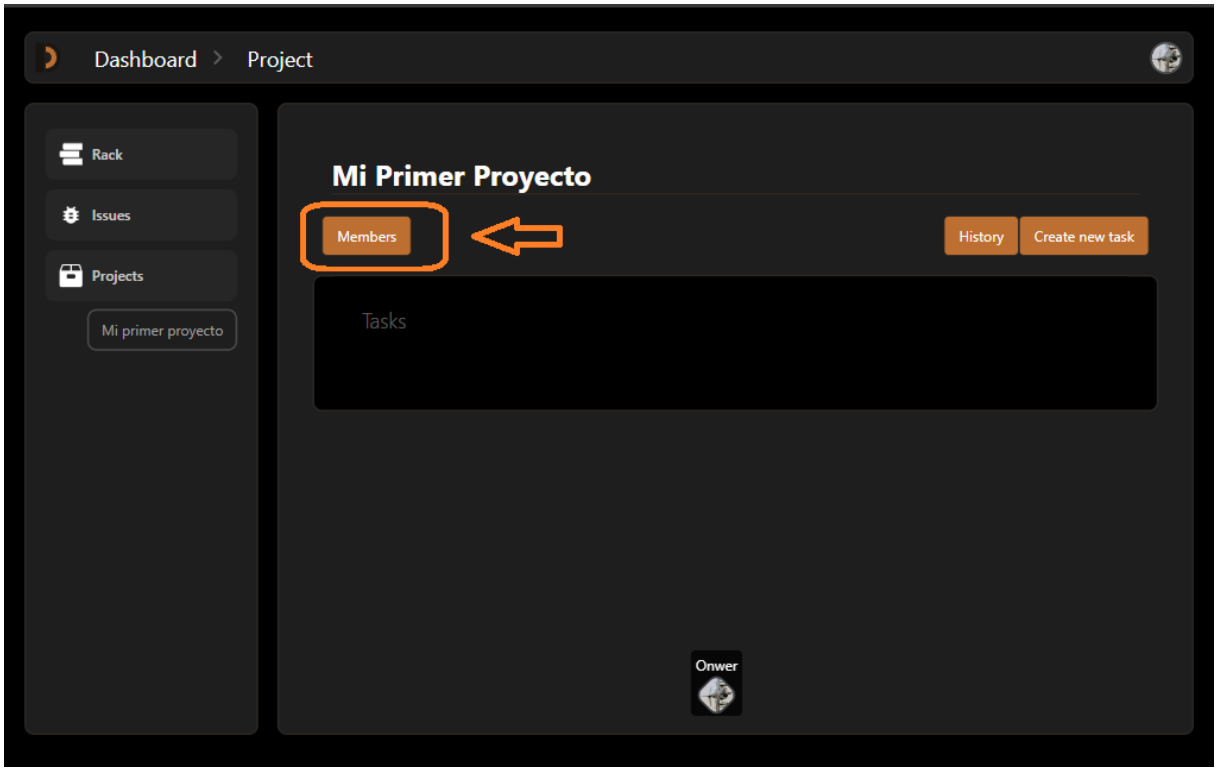


Figura 9: Vista del proyecto

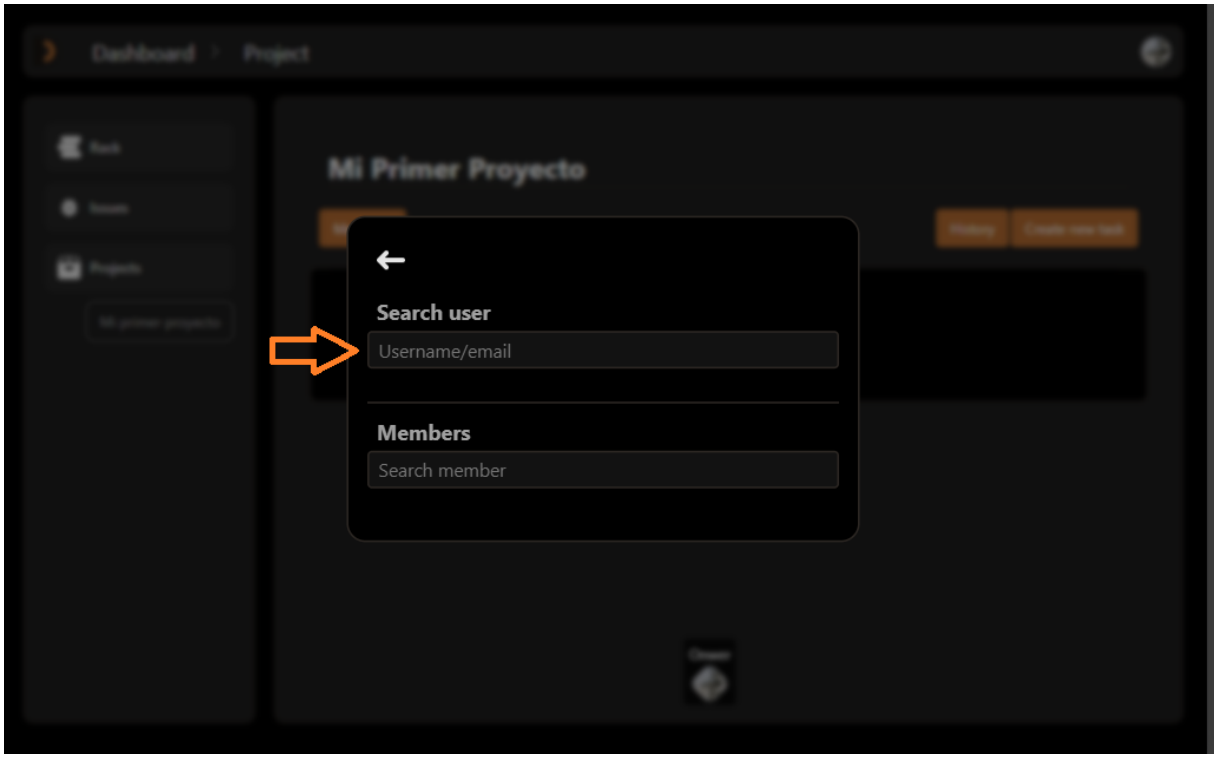


Figura 10: Buscar usuarios

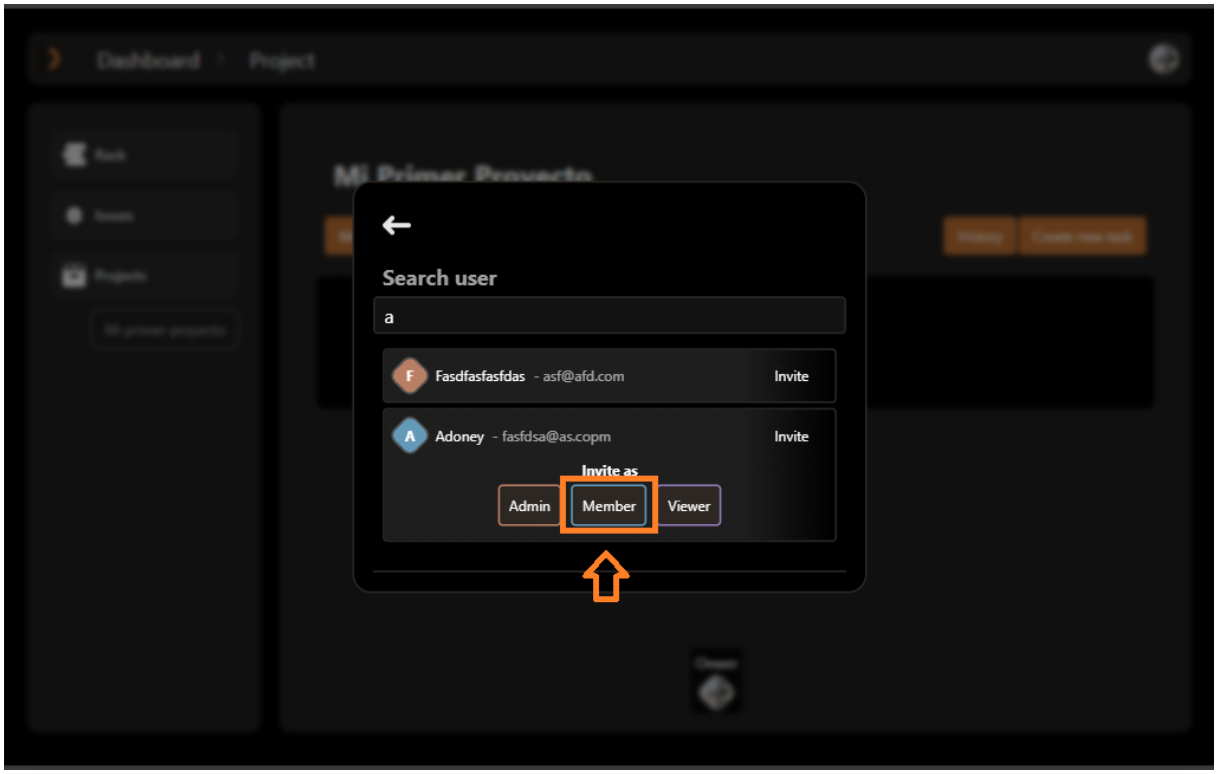


Figura 11: Agregar miembro

Una vez invitado, el usuario invitado podrá aceptarla desde su dashboard.

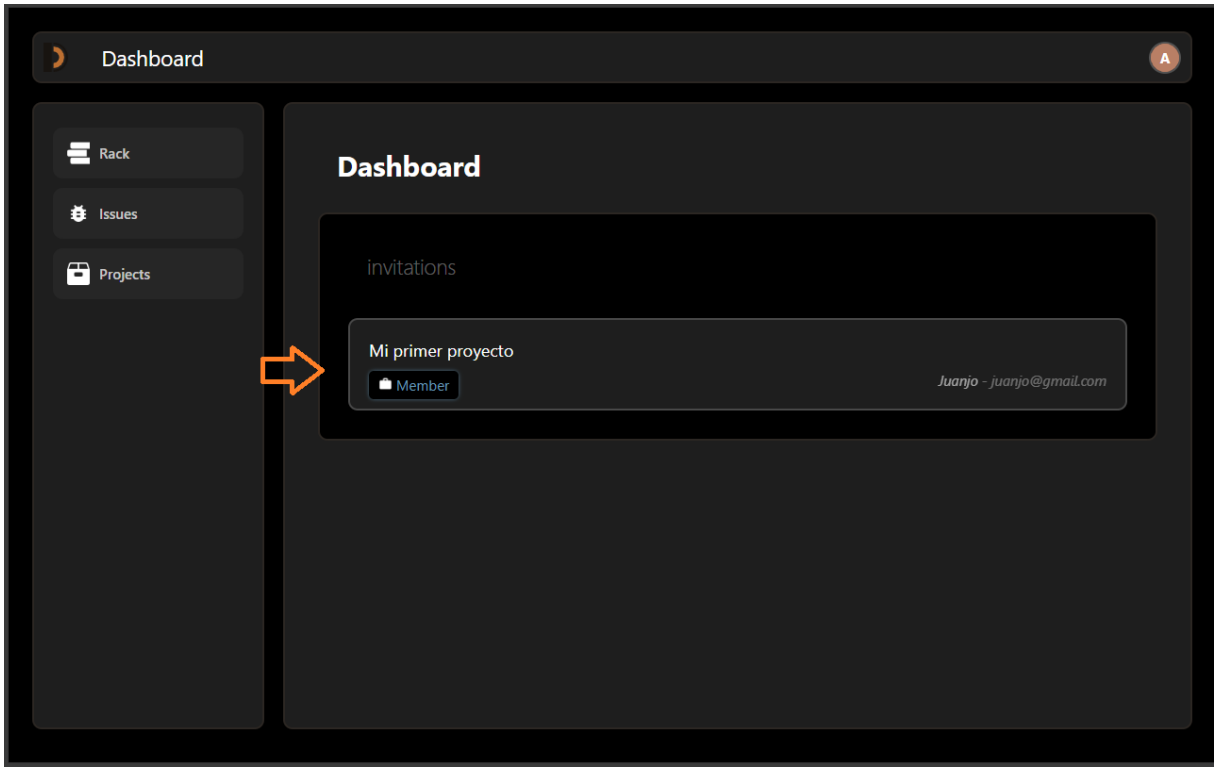


Figura 12: Dashboard desde el usuario invitado

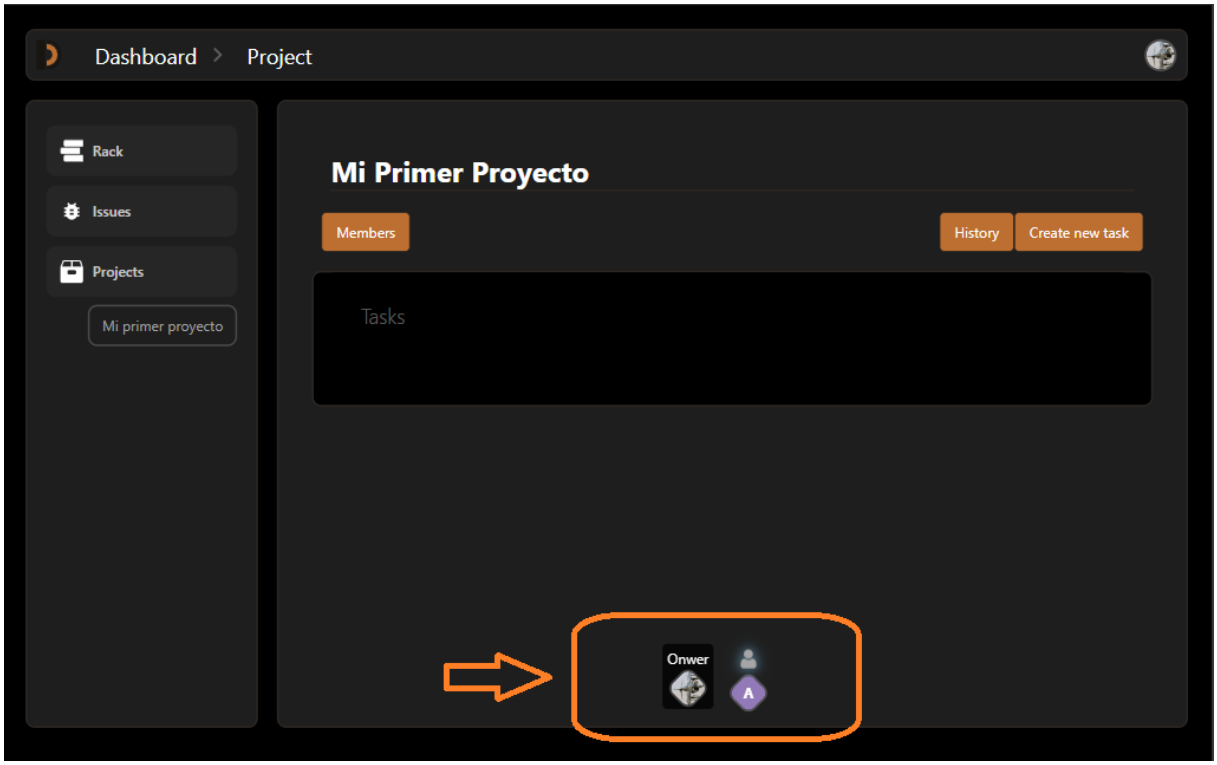


Figura 13: Miembro agregado al proyecto

Creación de tareas

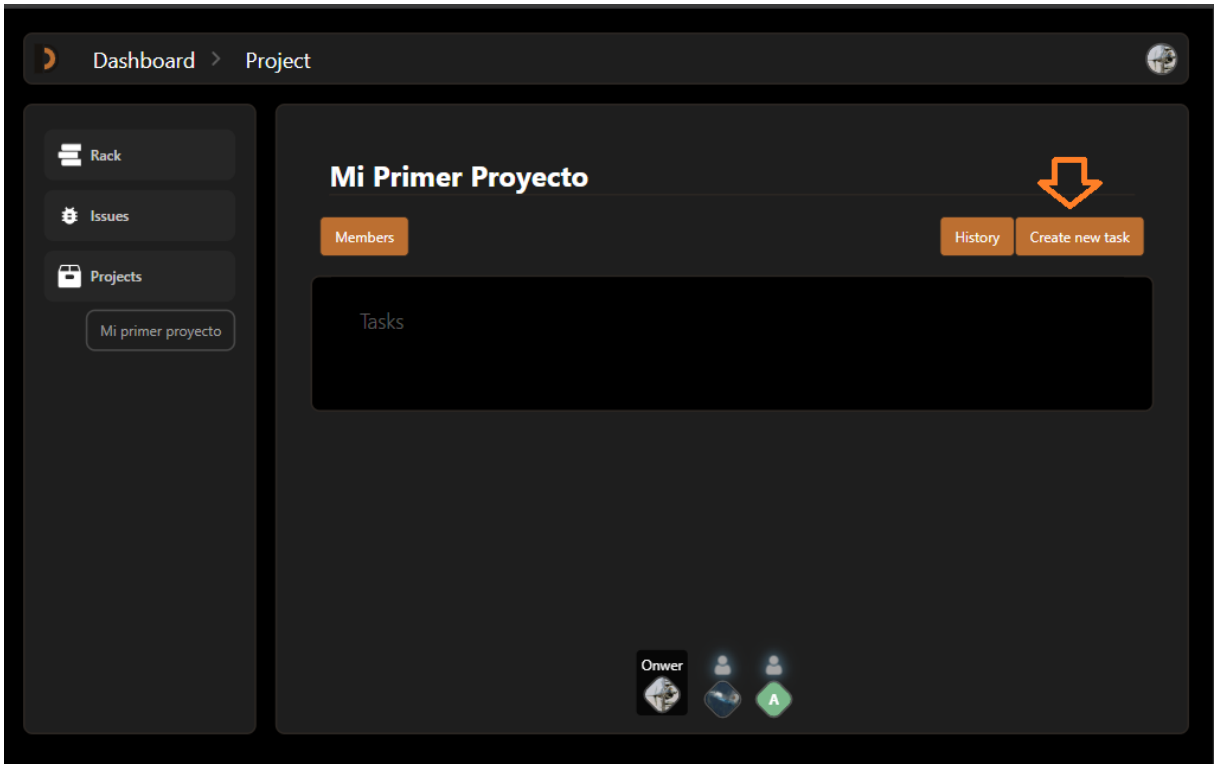


Figura 14: Creación de nueva tarea

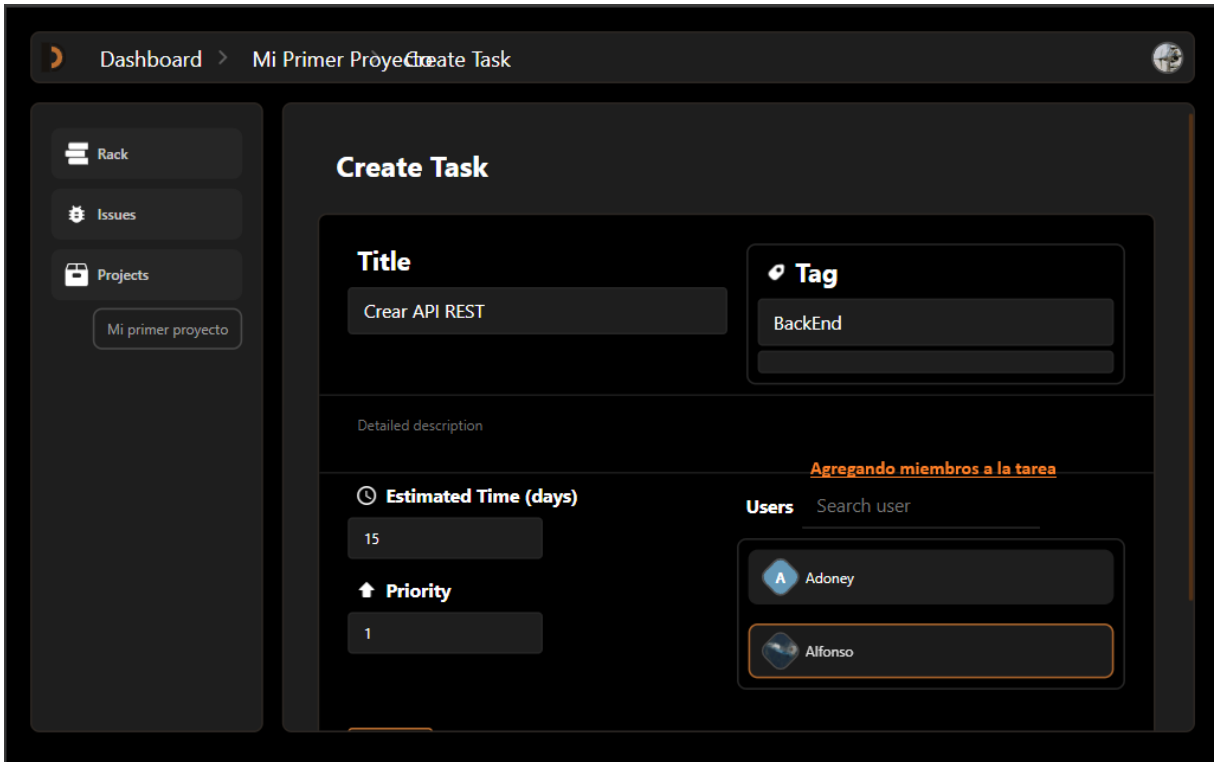


Figura 15: Creando tarea

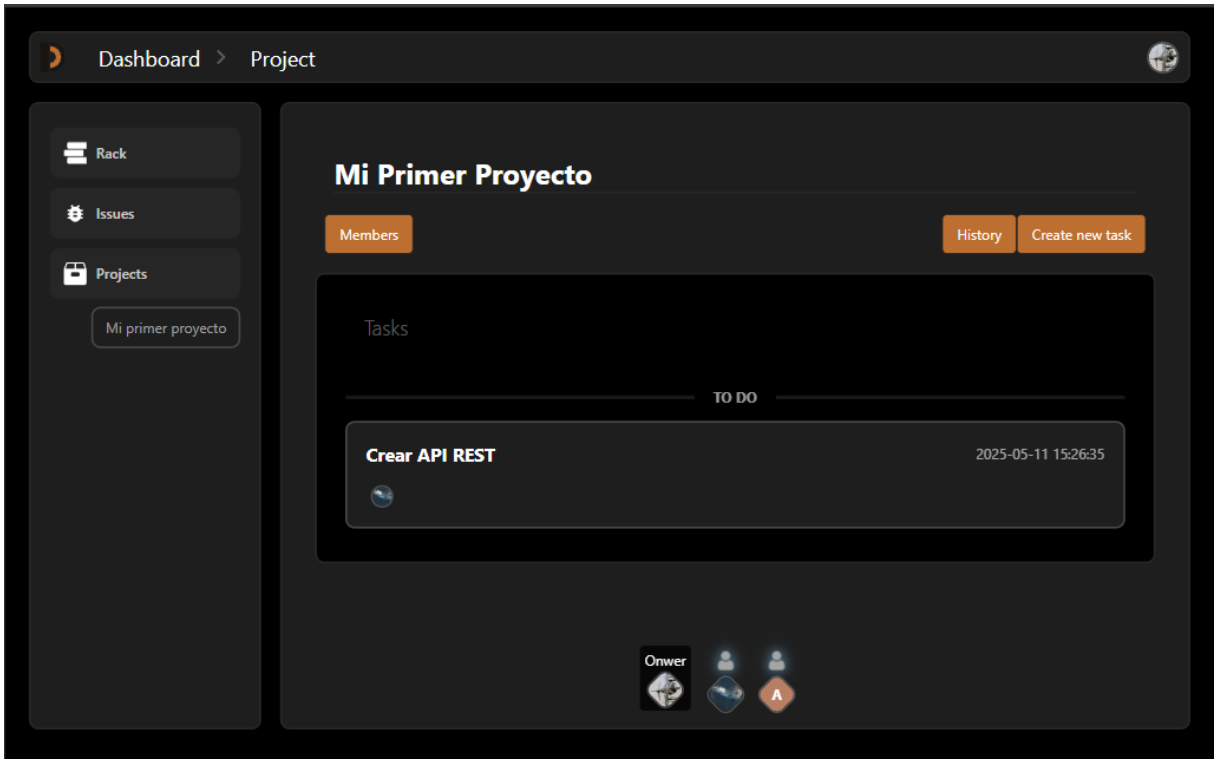


Figura 16: Tarea creada

La primera fase de las tareas es “To Do” (Por comenzar). Cada miembro incluido en la tarea podrá cambiar su estado en ella y dependiendo del estado de cada miembro, el estado de la tarea varía automáticamente:

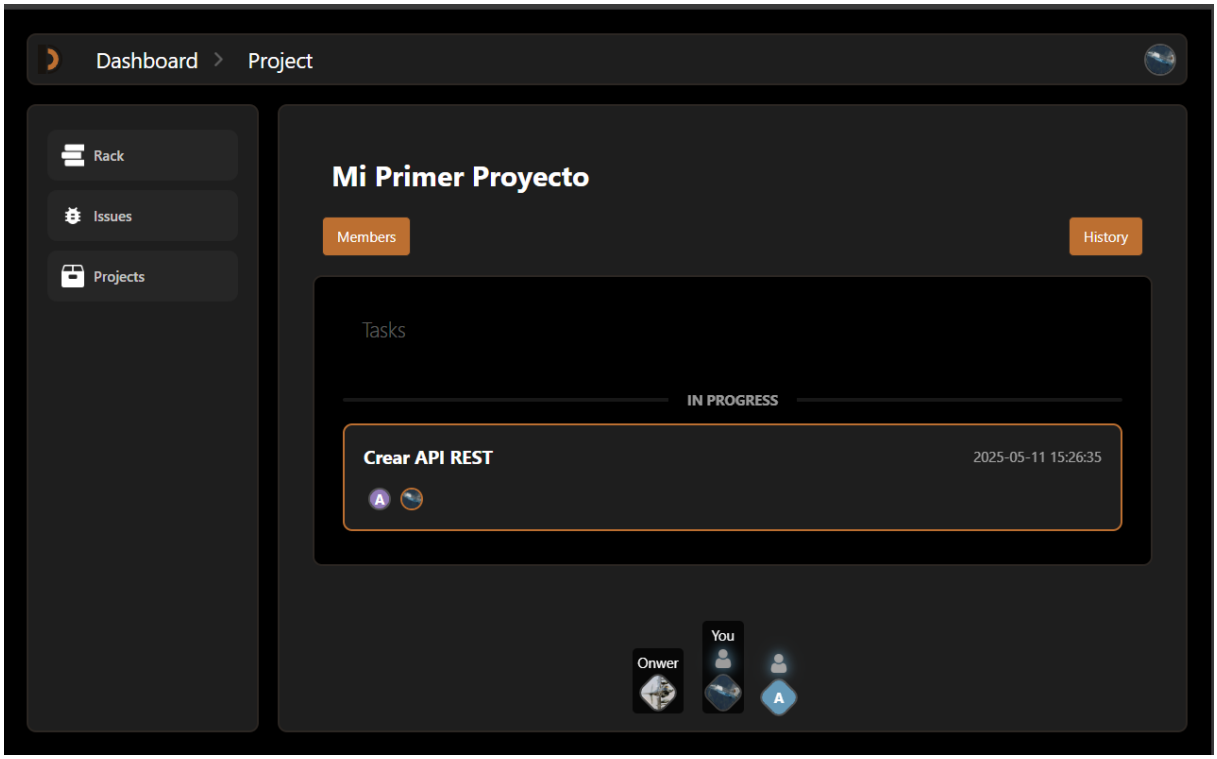


Figura 17: Tarea en progreso

Cuando todos los miembros de una tarea terminan su parte, la tarea automáticamente pasa a estado de “review” (revisión / checkeo) hasta que el propietario o unos de los administradores cambien su estado a “done” (terminado):

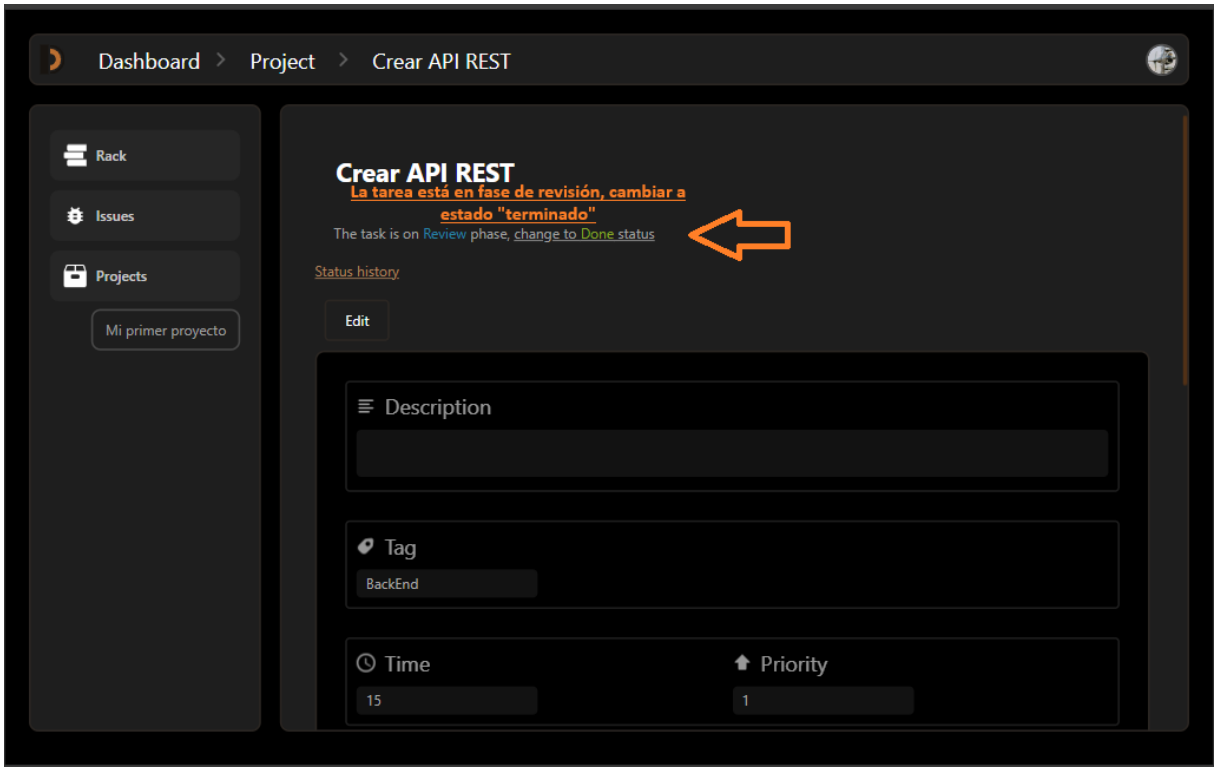


Figura 18: Cambio de fase de tarea a "done"

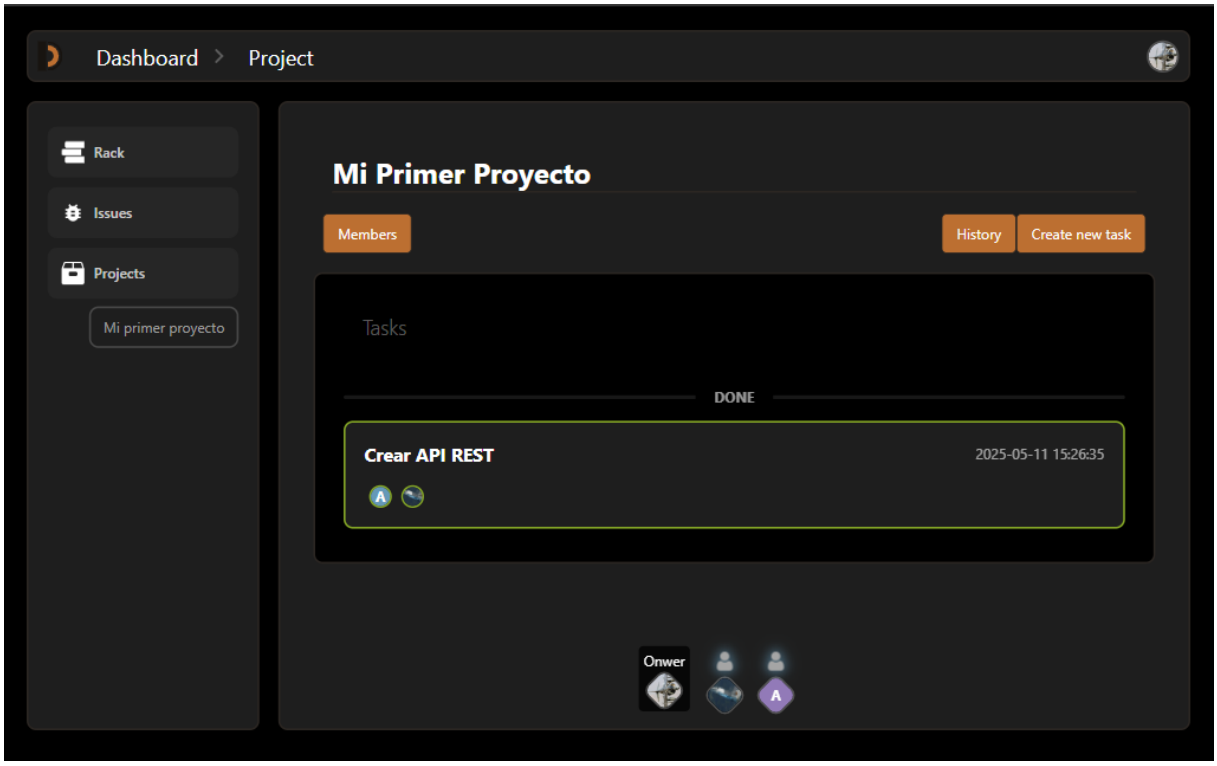


Figura 19: Tarea terminada

Agregar problemática

En un proyecto puede haber tareas pero también problemáticas. En este proyecto la creación de una “issue” o problemática es muy parecido al de una tarea:

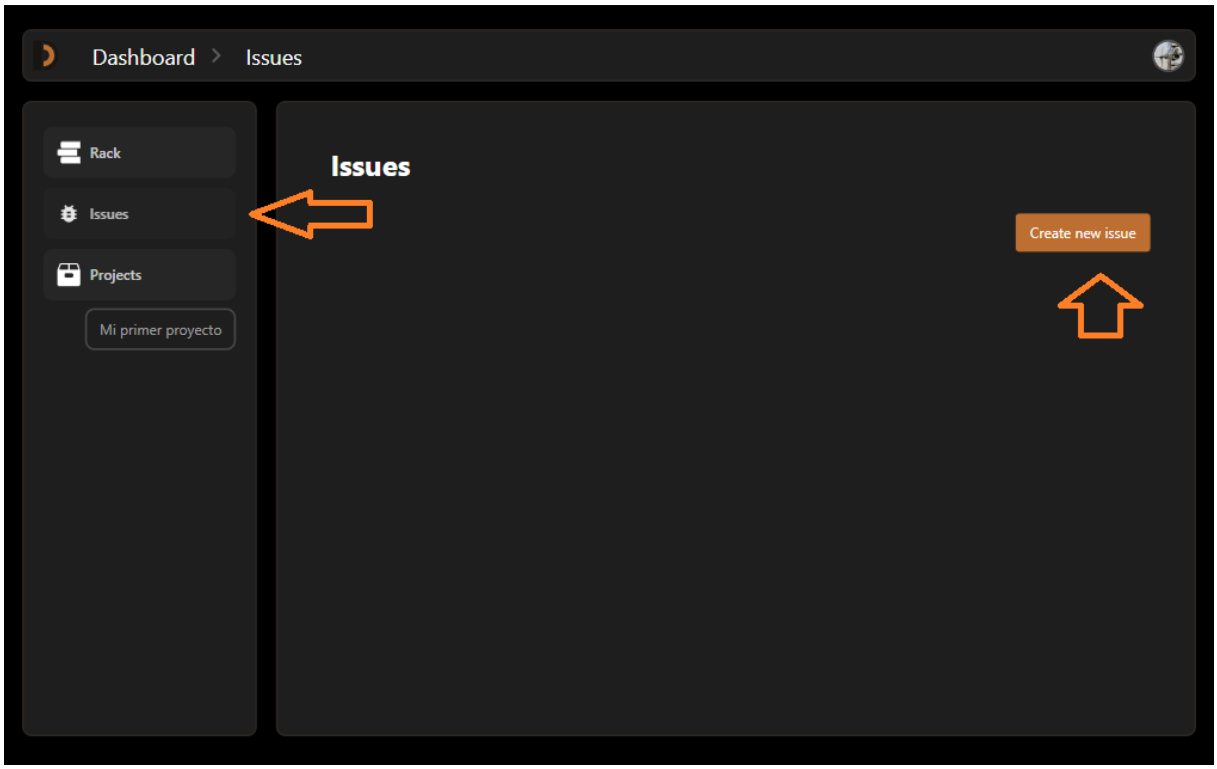


Figura 20: Vista de issues vacía

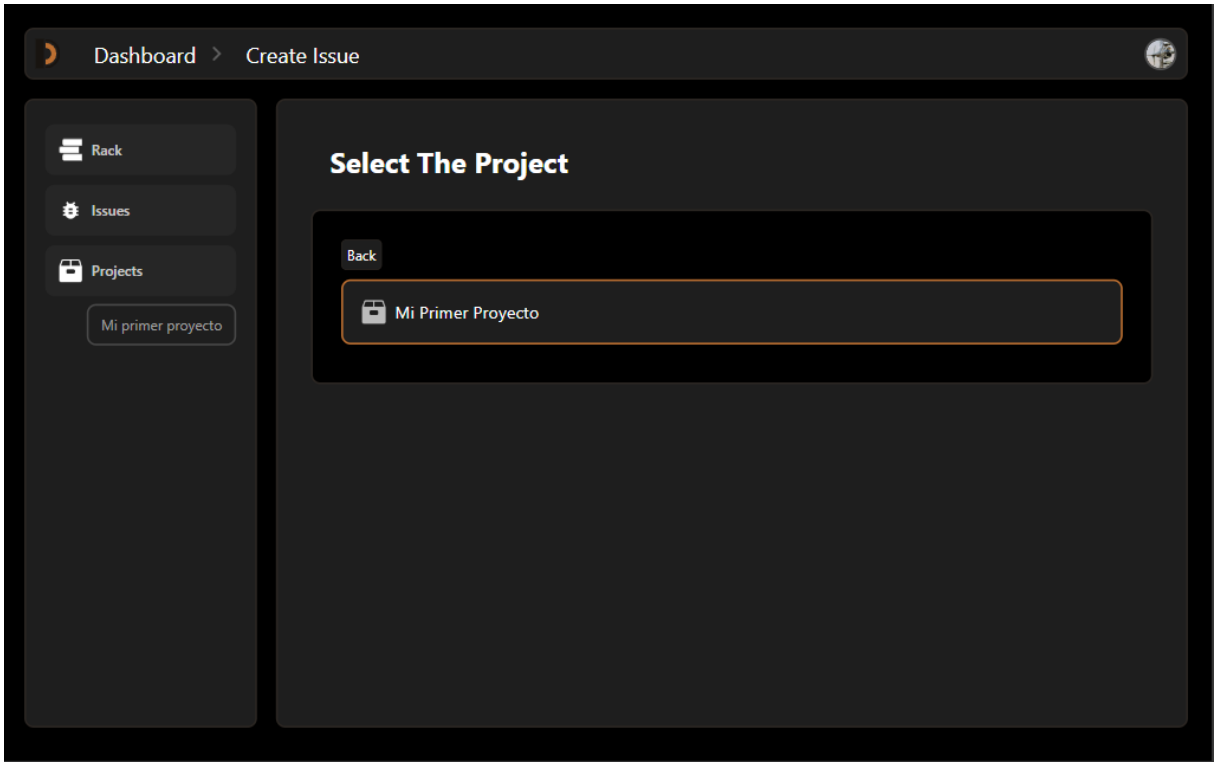


Figura 21: Selección de proyecto para la problemática

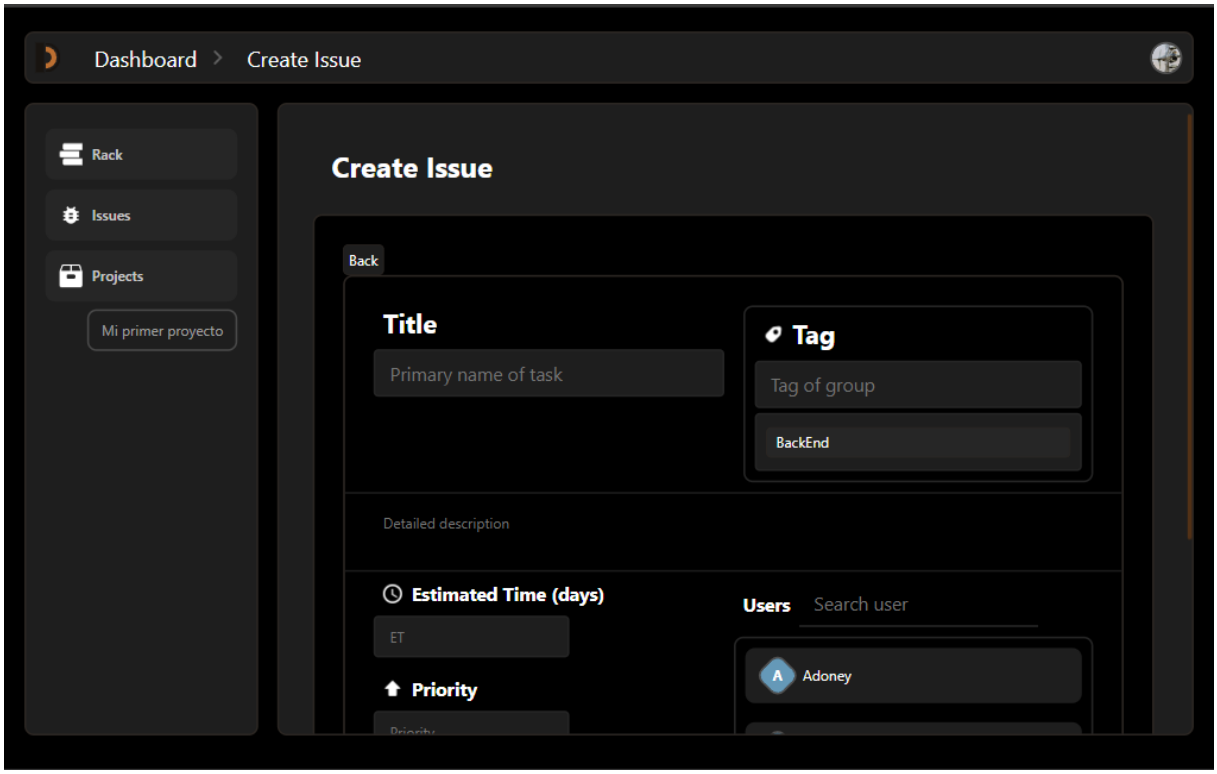


Figura 22: Formulario para la creación de una problemática (issue)

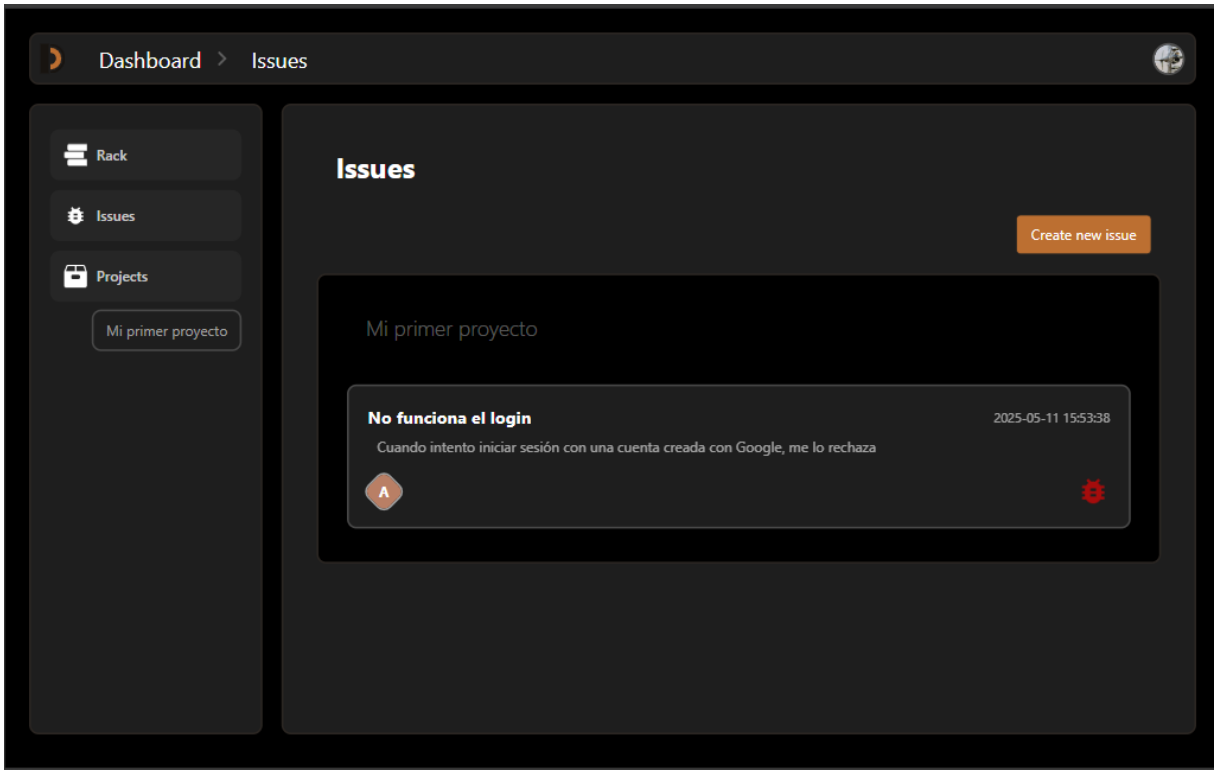


Figura 23: Vista de problemáticas

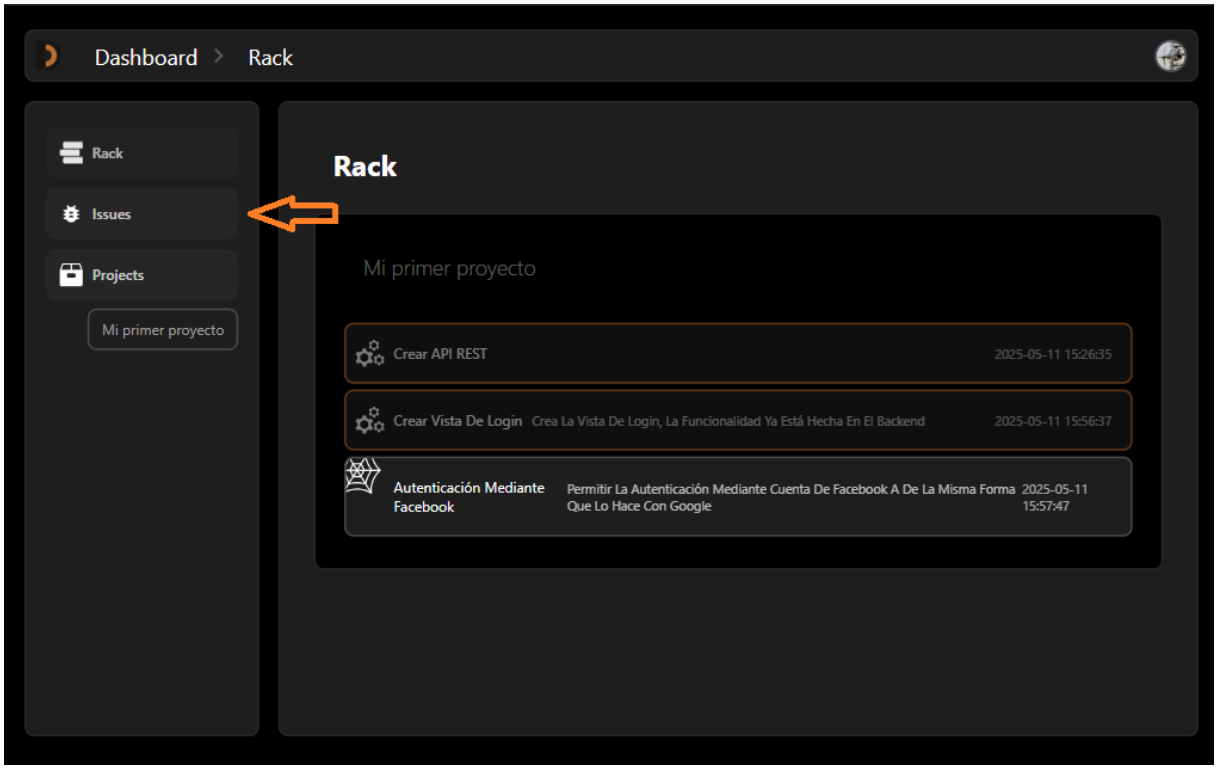


Figura 24: Rack, vista de todas las tareas de cada proyecto

DB Coding Standards y Guías de Estilo de Programación en PHP y JavaScript

DB Coding Standards

- Todas las tablas se nombran en minúscula. Las tablas con el nombre de más de 1 sujeto se separan entre “_”, por ejemplo “usuarios” y “usuarios_token”. Se cumple la misma norma para las columnas.
- Todas las acciones relacionadas con la modificación de tablas y registros ser hacen con procedimientos almacenados.
- Las acciones “SELECT” básicos se harán directamente, salvo los “SELECT” complejos como multiselect o sentencias demasiadas largas.
- Se usa el modelo Modelo Controlador (La vista no porque ya de eso se encarga el proyecto FrontEnd).

Estilos de programación

- El funcionamiento total de la aplicación está en la API.
- El FrontEnd se dedica únicamente para la interacción con la API.
- Toda verificación está hecha en la API, independientemente si el FrontEnd lo hace o no.
- Las variables que tienen más de 1 sujeto se pone en mayúscula el segundo, por ejemplo “usuario” y “tokenUsuario”.
- Todas las peticiones a la API están en el modelo “API/api.ts”, su configuración en “API/config.ts, el cual contiene la URL de la API y el prefijo (“v1”, “v2”…).

Guía del Desarrollador

Despliegue de la API (PulseServer)

Requisitos:

- Composer.
- PHP >= 8.2.
- mysql_pdo.
- Apache (Si se desea desplegar en el)
- módulo de apache “rewrite” (si se despliega en apache).
- Unzip
- phpxml

- **Despliege:**

1. Instalar dependencias

Desde la raiz del proyecto: composer install.

2. Base de datos

Como es de esperar, es requerido tener la base de datos importada para que la API funcione.

3. Archivo .env

En el proyecto hay un fichero .env.example, se debe renombrar o duplicar a “.env”.

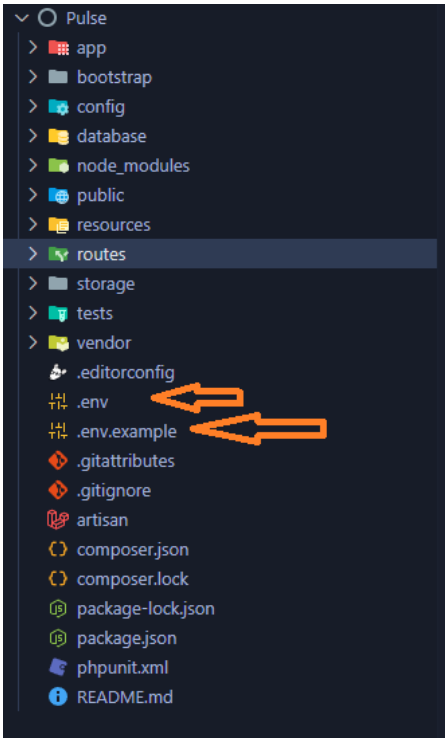


Figura 25: Sistema de directorios API

Se configura el fichero .env:

- Configuración de la base de datos.
- Configuración de la clave secreta de Google (para el inicio de sesión mediante Google), el cual se obtiene en la Google API Console

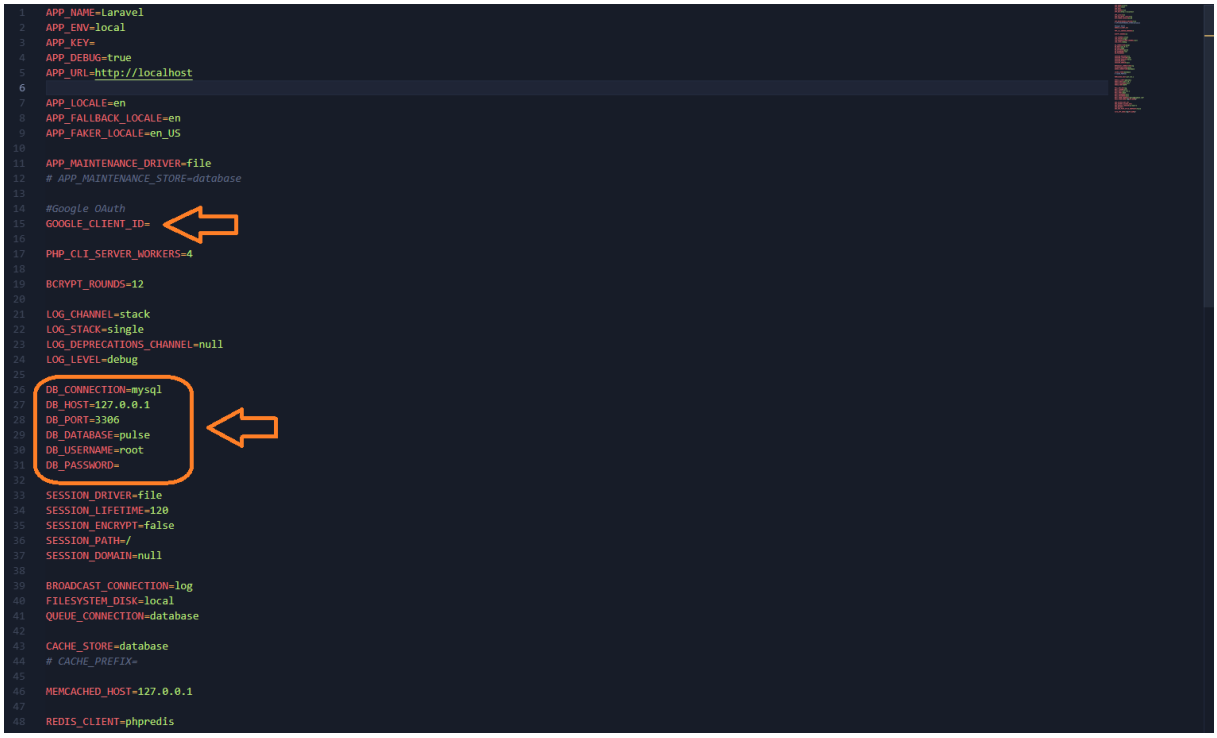


Figura 26: Fichero .env API

Para que Laravel funcione, la aplicación requiere de generar una clave. Para ello se ejecuta php artisan key:generate desde la raíz del proyecto (No funcionará si no tiene el fichero .env).

4. Despliegue sin apache

Para desplegar el proyecto sin apache solo será necesario ejecutar desde la raíz el comando php artisan serve y ya estaría totalmente funcional.

5. Despliegue con apache

Para poder desplegar con apache será necesario tenerlo instalado, tener el módulo “rewrite” habilitado, instalar, si es necesario, mysql_pdo y hacer que el dominio apunte a la carpeta “public” localizada en la raíz del proyecto. (No será necesario utilizar el comando php artisan serve).

Despliegue del proyecto React (FrontEnd)

Requisitos:

- NodeJS.
- NPM.
- Módulo rewrite de apache habilitado (Si se despliega en Apache).

Archivo environment

En la raíz hay una carpeta “environments”, el cual hay 2 ficheros, environment.ts y environment.development.ts, cada uno con el siguiente contenido:

```
export const environment = {
  production: true,
  GOOGLE_CLIENT_ID: ""
};
```

Figura 27: environments de Angular

Configurar URL de la API

Se debe configurar el fichero “/src/API/config.ts”, asignando la URL de la api y, si es necesario, el prefijo:

```
export const apiUrl = "http://localhost:8000/api"
export const prefix = "v1"
```

Figura 28: configuración de la conexión a la API de Angular

Instalar dependencias

Para instalar las dependencias del proyecto se ejecutará el comando npm install desde la raíz del proyecto. **Es necesario tener instalado NodeJS y NPM.**

Despliegue sin Apache

Para desplegar la aplicación sin necesidad de utilizar un servidor Apache se utilizará el propio servidor incluido en el Framework (o utilizar alguno alternativo). Para ello se ejecutará el comando npm run serve.

Despliegue con Apache

Lo primero que hay que hacer para desplegar el proyecto será montarlo, por lo tanto se ejecutará el comando npm run build. Este comando creará una carpeta "**dist**" listo para su despliegue.

Una vez montada la aplicación se debe crear un fichero "**.htaccess**" en el directorio "**/dist/pulse/browser/**", este fichero no se crea automáticamente pero ya se encuentra una copia en la raíz, por lo que solo deberá copiarla y pegarla en "**/dist/pulse/browser/**".

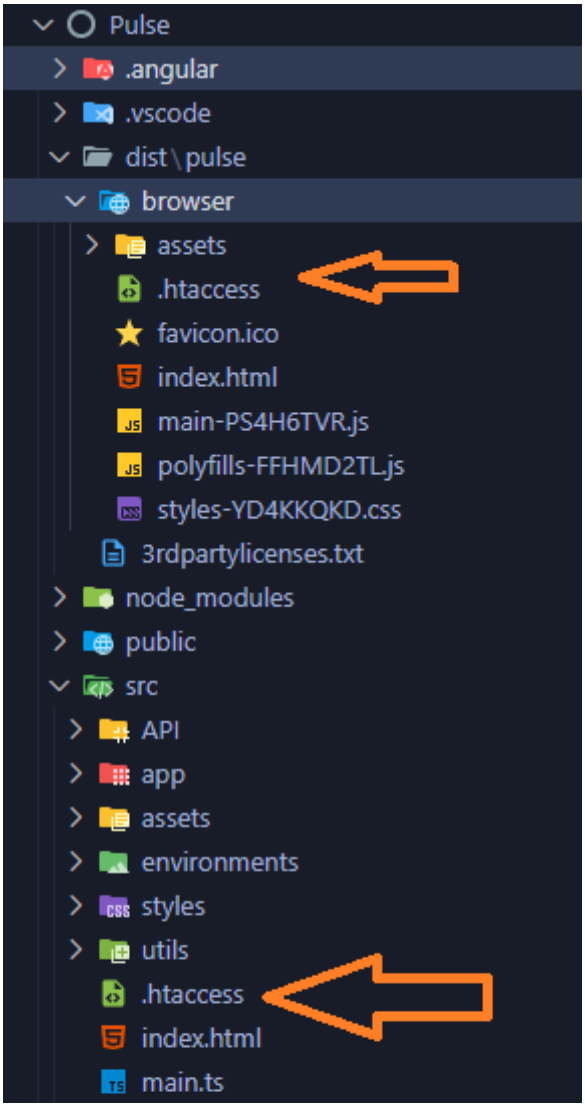


Figura 29: Sistema de directorios Angular

Después de todo esto, el servidor apache deberá apuntar al directorio donde está el fichero **.htaccess**, en "**/dist/pulse/browser/**".

Cada vez que hay un cambio en la aplicación se deberá montar (**npm run build**) y volver copiar el fichero .htaccess puesto que el comando reescribe el directorio entero.

Guía Técnica

Estructura de la base de datos

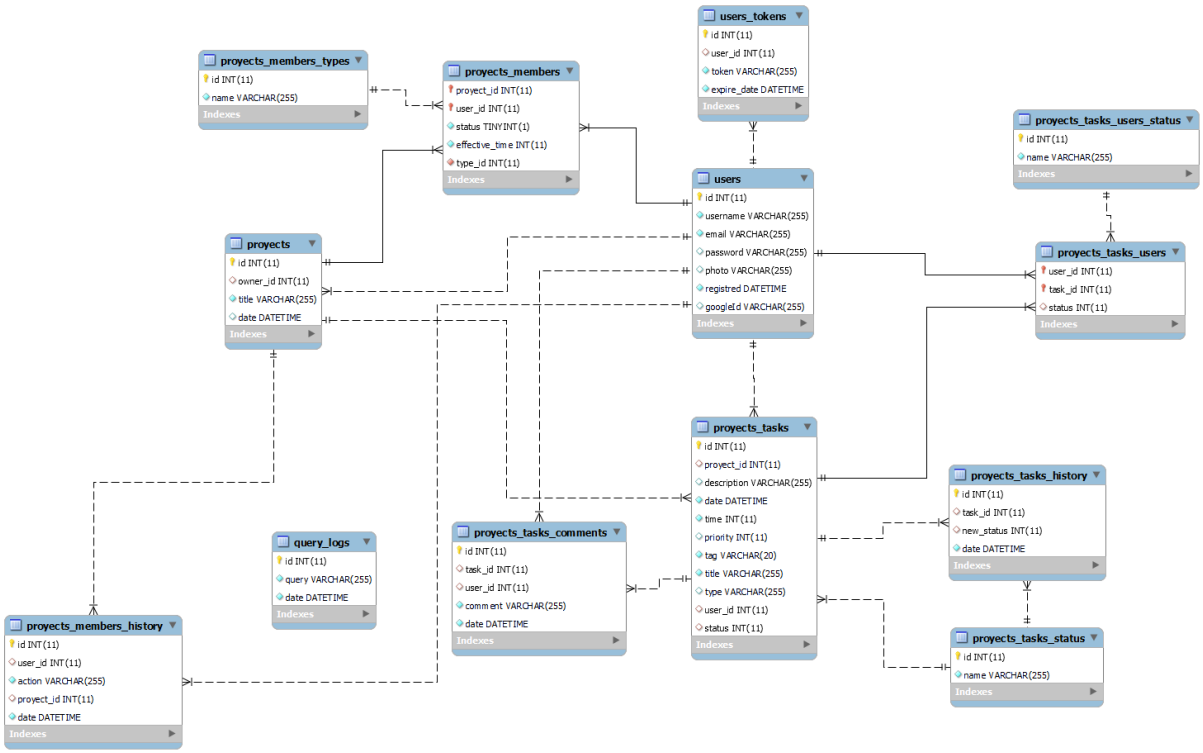


Figura 30: Estructura de la base de datos

Diagrama UML

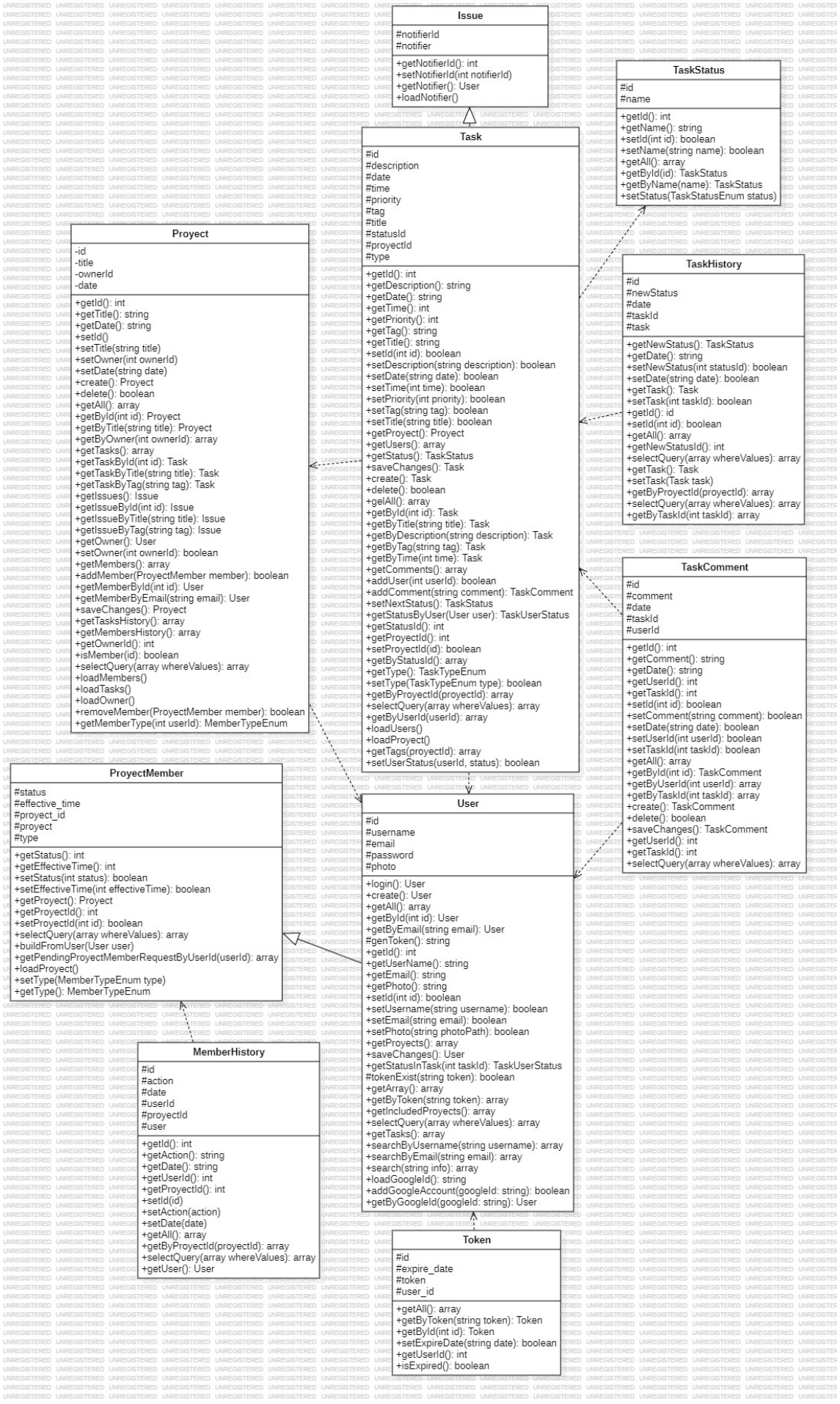


Figura 31: Diagrama UML