TEAMFLOW – GESTOR DE TAREAS PARA EQUIPOS

Desarrollo de interfaces ricos para internet
24/25

Mónica Garrigós Sánchez
Máster Desarrollo de Aplicaciones y Servicios Web - UA

ÍNDICE

Índice de ilustraciones 2
1. Introducción 3
2. objetivos del proyecto 3
2.1. Objetivos específicos 3
3. Diseño de interfaz y prototipado 3
4. Tecnologías 5
4.1. Frontend (parte visual de la app) 5
3.2. Backend (gestión de datos y usuarios) 6
5. Diseño de la solución 6
6. Funcionalidades 7
1.1 Autenticación de usuarios 7
1.2 Gestión de proyectos 9
1.3 Gestión de tareas 11
1.4 Panel de administrador 13
1.5 Internacionalización (i18n) 14
1.6 Logging 16
1.7 Pruebas unitarias 17
7. Conclusión 18

ÍNDICE DE ILUSTRACIONES

Ilustración 1 - Prototipado de la pantalla de login diseñado en Figma 4
Ilustración 2 - Prototipado de la pantalla principal diseñada en Figma 4
Ilustración 3 - Prototipado del modal de creación de proyecto diseñado en Figma ... 5
Ilustración 4 - Prototipado de la pantalla de tareas de un proyecto diseñada en Figma 5
Ilustración 5 - Interfaz del formulario de login 7
Ilustración 6 - Interfaz del formulario registro 8
Ilustración 7 - Fragmento de código donde se llama a signInWithEmailAndPassword 8
Ilustración 8 - Fragmento del código donde se llama a createUserWithEmailAndPassword 9
Ilustración 9 - Página principal de un miembro de la app 10
Ilustración 10 - Modal para crear un nuevo proyecto 10
Ilustración 11 - Opciones en el proyecto 10
Ilustración 12 - Editar información del proyecto 11
Ilustración 13 - Filtrar proyectos archivados 11
Ilustración 14 - Tablero de tareas de un proyecto 12
Ilustración 15 - Creación de una nueva tarea 12
Ilustración 16 - Modal de edición de una tarea 13
Ilustración 17 - Cambiar el estado de una tarea arrastrando de una columna a otra 13
Ilustración 18 - Panel de administración 14
Ilustración 19 - Cambiar usuario luis a administrador 14
Ilustración 20 - Selector para cambiar de idioma en login 15
Ilustración 21 - Selector de idioma en el topbar 15
Ilustración 22 - Configuración del sistema de idiomas 16
Ilustración 23 - Lógica principal del módulo de logs 16
Ilustración 24 - Ejemplo de logs de tipo [INFO] 17
Ilustración 25 - Pruebas unitarias ejecutadas 17
Ilustración 26 - Fragmento de código del test AuthForm 18

1. INTRODUCCIÓN

TeamFlow es una aplicación web desarrollada como parte del trabajo final de la asignatura de Desarrollo de Interfaces Ricas en Internet. El objetivo principal ha sido diseñar y construir una aplicación colaborativa que permita a varios usuarios crear y gestionar proyectos de forma visual y estructurada, aplicando tecnologías modernas del ecosistema React.

La aplicación ha sido pensada para ser fácil de usar, visualmente clara y útil para distintos tipos de usuarios. Además, está preparada para usarse en diferentes idiomas, tiene control de acceso con inicio de sesión y ofrece distintas funciones según el tipo de usuario.

2. OBJETIVOS DEL PROYECTO

El objetivo principal de TeamFlow es facilitar el trabajo en equipo, mejorando la comunicación, el control de tareas y la productividad. La aplicación busca convertirse en una herramienta útil para cualquier grupo que necesite coordinar tareas y proyectos de forma sencilla.

2.1. OBJETIVOS ESPECÍFICOS

- Permitir la creación y gestión de proyectos, donde se puede añadir nombre, descripción y miembros del equipo.
- Ofrecer la opción de crear tareas detalladas, asignarlas a usuarios, y clasificarlas por prioridad o estado.
- Hacer visible el avance de cada proyecto, con listas de tareas organizadas por columnas.
- Favorecer la colaboración a través de comentarios en las tareas.
- Garantizar la seguridad mediante registro e inicio de sesión con control de acceso.
- Ofrecer una interfaz clara y moderna, adaptable a distintos idiomas.
- Aumentar la productividad del equipo ayudando a que todos sepan qué deben hacer y cuándo.

3. DISEÑO DE INTERFAZ Y PROTOTIPADO

Antes de comenzar el desarrollo técnico, se diseñó un prototipo interactivo utilizando la herramienta Figma. Este prototipo permitió planificar la estructura visual de la aplicación, validar la experiencia de usuario (UX) y establecer una guía coherente para el diseño de los componentes.

Objetivos del prototipo:

- Visualizar el flujo de navegación de la aplicación.
- Definir el estilo visual, uso de colores, botones y tipografías.
- Simular la interacción del usuario con vistas clave (login, dashboard, vista de proyecto...).
- Detectar mejoras antes de escribir código.

El prototipo se centró especialmente en las siguientes pantallas:

- **Pantalla de login**: diseño limpio, campos centrados, selector de idioma.
- **Panel principal**: listado de proyectos, botón para crear nuevo, buscador y filtros.
- **Vista de proyecto**: columnas dinámicas, tareas con prioridad, miembros y botón de creación rápida.
- **Modal de creación de proyectos**: formulario para introducir y editar datos de forma clara y rápida.

Este diseño facilitó una implementación más fluida y consistente del frontend, respetando los principios de accesibilidad y simplicidad.

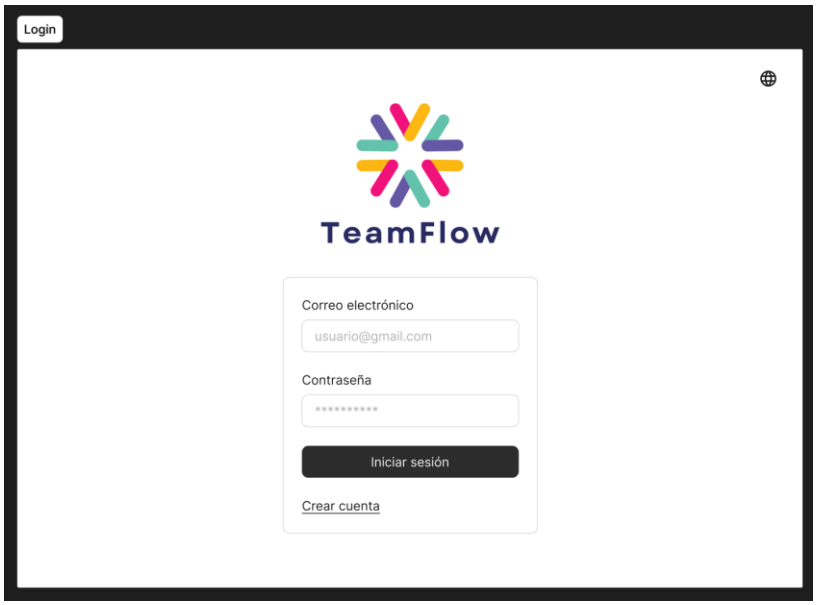

Ilustración 1 - Prototipado de la pantalla de login diseñado en Figma

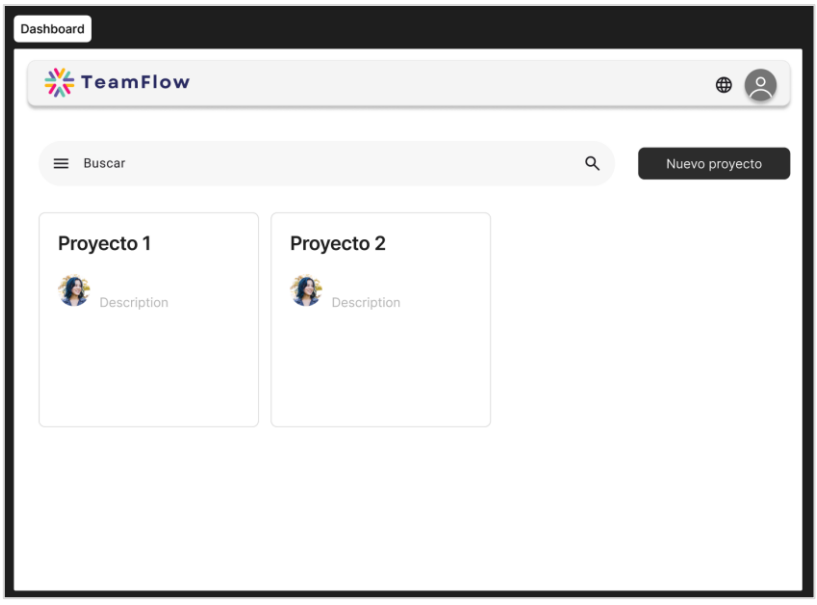

Ilustración 2 - Prototipado de la pantalla principal diseñada en Figma

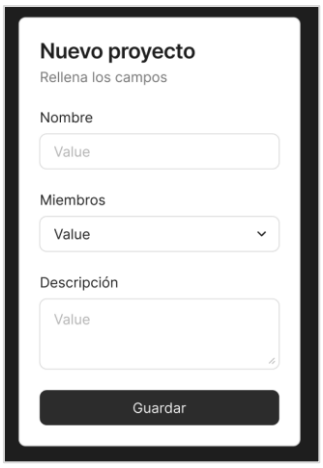

Ilustración 3 - Prototipado del modal de creación de proyecto diseñado en Figma

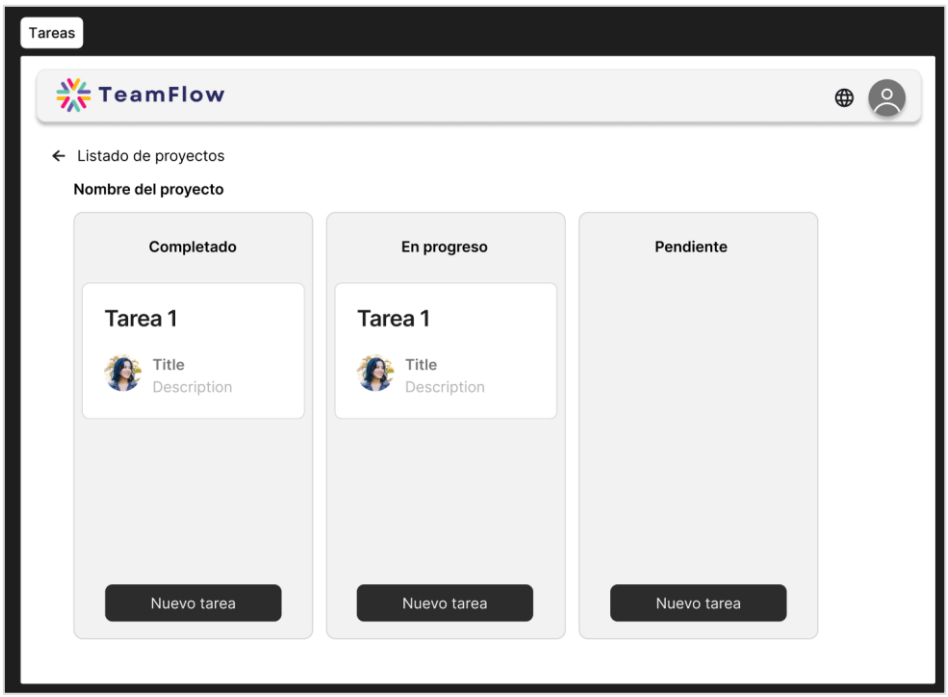

Ilustración 4 - Prototipado de la pantalla de tareas de un proyecto diseñada en Figma

4. TECNOLOGÍAS

Para el desarrollo de TeamFlow se han empleado tecnologías modernas que permiten crear una aplicación web funcional, rápida y fácil de mantener. Se han usado herramientas tanto para la parte visual (frontend) como para la gestión de datos y usuarios (backend).

4.1. FRONTEND (PARTE VISUAL DE LA APP)

- **React.js:** Es la biblioteca principal usada para construir la interfaz de usuario. Permite crear componentes reutilizables y una navegación fluida.

- **React Router**: Se usa para gestionar las distintas páginas de la aplicación (inicio, panel, proyectos...).
- **Redux Toolkit**: Facilita la gestión del estado global, es decir, la información que necesita estar accesible desde diferentes partes de la aplicación (como los datos del usuario).
- **Material UI (MUI):** Es la librería utilizada para los estilos y componentes visuales (botones, formularios, diálogos…), dando a la aplicación un diseño moderno y profesional.
- **i18n**: Herramienta de internacionalización que permite que la aplicación esté disponible en varios idiomas.
- **Vitest y Testing Library**: Para realizar pruebas automáticas y asegurarse de que las partes más importantes de la aplicación funcionan correctamente.

3.2. BACKEND (GESTIÓN DE DATOS Y USUARIOS)

- **Firebase Authentication:** Se encarga del registro e inicio de sesión de los usuarios. Asegura que solo las personas autorizadas puedan acceder a la aplicación.
- **Firebase Realtime Database**: Se usa para guardar la información de los proyectos, tareas y usuarios. Es una base de datos que actualiza los datos en tiempo real para todos los usuarios conectados.

5. DISEÑO DE LA SOLUCIÓN

Para organizar el desarrollo y facilitar el mantenimiento del proyecto TeamFlow, la aplicación se ha estructurado en varios módulos o carpetas, cada uno con una función específica:

- components/: Aquí se encuentran los componentes reutilizables, como formularios, tarjetas de proyectos, columnas de tareas, botones, modales, etc. Estos elementos se pueden usar en diferentes partes de la aplicación sin tener que volver a escribir el mismo código.
- pages/: Este módulo contiene las vistas principales que ve el usuario, como la página de inicio de sesión, el panel de proyectos (Dashboard), o el detalle de un proyecto. Cada página representa una pantalla completa dentro de la app.
- redux/: Incluye todo lo relacionado con la gestión del estado global de la aplicación usando Redux Toolkit. Aquí están los "slices" (fragmentos de estado como el usuario o los proyectos), y la configuración del store, que es donde se guarda y organiza la información.
- utils/: Esta carpeta guarda funciones auxiliares que ayudan al funcionamiento general de la app. Por ejemplo, se incluye aquí el logger, que registra ciertas acciones, o funciones para trabajar con la base de datos.
- firebase/: Contiene la configuración de Firebase, la plataforma en la nube usada tanto para almacenar los datos como para autenticar a los usuarios. Aquí se inicializan los servicios de autenticación y base de datos.
- i18n/: Aquí está la configuración para el soporte multilenguaje. Gracias a este módulo, la aplicación puede mostrarse en distintos idiomas según la preferencia del usuario (por ejemplo, español e inglés).

6. FUNCIONALIDADES

1.1 AUTENTICACIÓN DE USUARIOS

Se ha implementado un sistema de autenticación utilizando Firebase Authentication, que permite tanto el registro de nuevos usuarios como el inicio de sesión. El formulario de autenticación permite alternar entre las vistas de login y registro, y valida los datos introducidos. Al autenticarse, los datos del usuario se almacenan en el estado global (Redux) y también en Firebase Realtime Database.

Los archivos principales que han sido modificados para conseguir esta funcionalidad han sido:

- AuthForm.tsx: contiene el formulario para login/registro, validaciones y llamadas a Firebase.
- AuthSlice.ts: slice de Redux que gestiona el estado del usuario (autenticado, error, loading).
- Firebase.ts – configuración de Firebase.
- LoginPage.tsx – página principal de acceso
- saveUserToDatabase.ts – guarda los datos del usuario nuevo en la base de datos

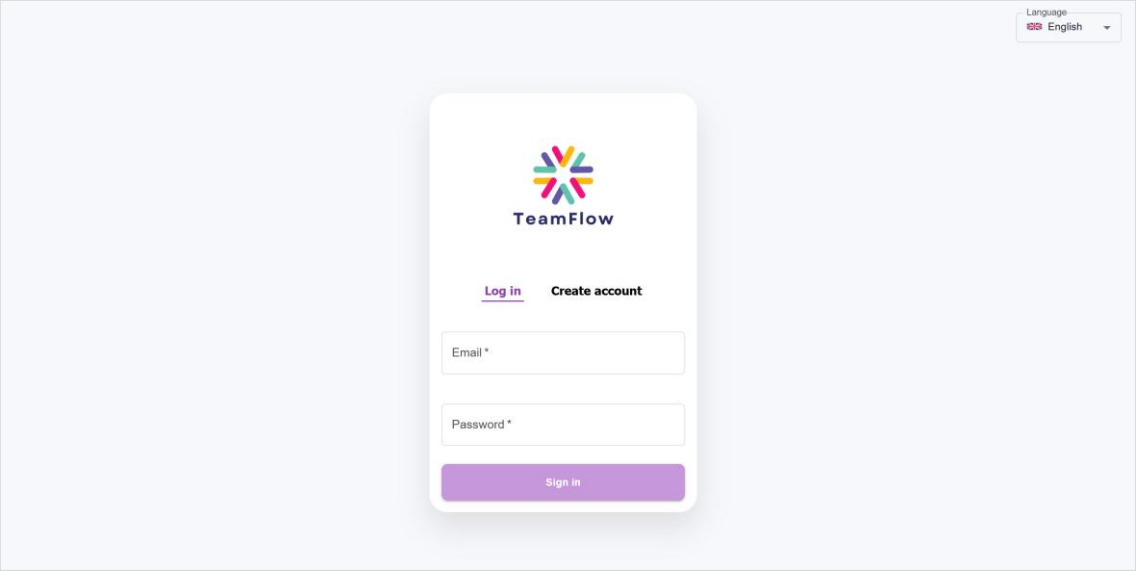

Ilustración 5 - Interfaz del formulario de login

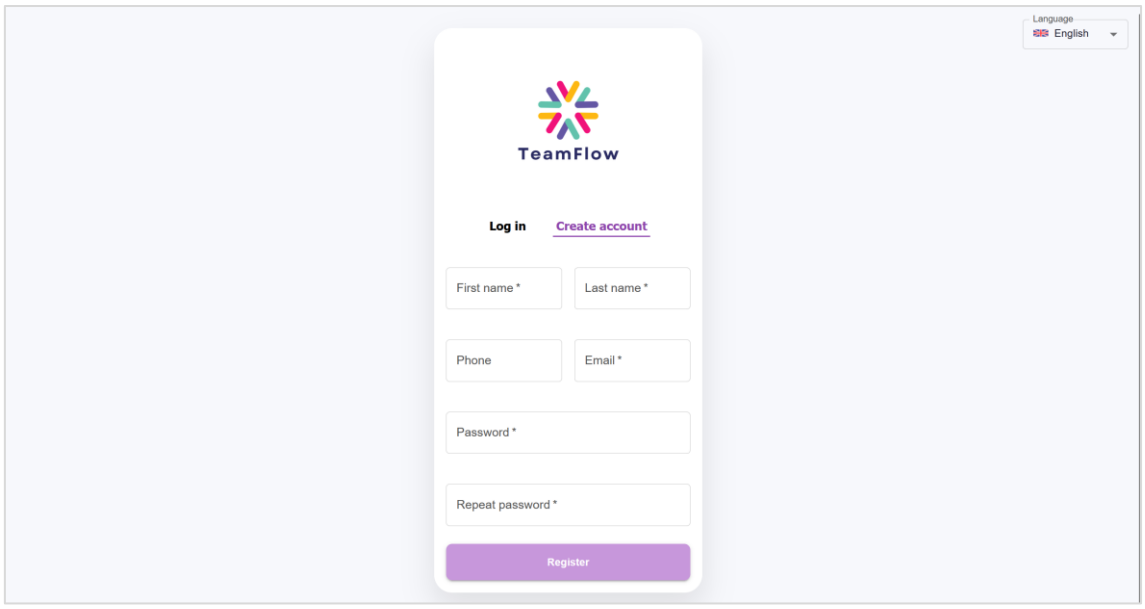

Ilustración 6 - Interfaz del formulario registro

```
dispatch(setLoading(true));
try {
  let firebaseUser;

  if (isLogin) {
    const userCredential = await signInWithEmailAndPassword(auth, email, password);
    firebaseUser = userCredential.user;

    const snap = await get(dbRef(db, `users/${firebaseUser.uid}`));
    const userDB = snap.val();

    if (userDB.active === false) {
      setSnackbar({
        open: true,
        message: t("authErrors.disabled-account"),
        severity: "error",
      });
      logWarn(`Login attempt blocked for disabled user: ${email}`);
      return;
    }

    dispatch(setUser({
      uid: firebaseUser.uid,
      email: firebaseUser.email,
      displayName: `${userDB.firstName} ${userDB.lastName}`,
      photoURL: firebaseUser.photoURL || null,
      role: userDB.role,
      active: userDB.active !== false
    }));
```

Ilustración 7 - Fragmento de código donde se llama a signInWithEmailAndPassword

```
} else {
  const userCredential = await createUserWithEmailAndPassword(auth, email, password);
  firebaseUser = userCredential.user;

  await saveUserToDatabase(firebaseUser, { firstName, lastName, phone });

  dispatch(setUser({
    uid: firebaseUser.uid,
    email: firebaseUser.email,
    displayName: `${firstName} ${lastName}`,
    photoURL: firebaseUser.photoURL || null,
    role: "member"
  }));

  setSnackbar({
    open: true,
    message: t("authSuccess.register"),
    severity: "success",
  });

  logInfo(`New user registered: ${email} (${firebaseUser.uid})`);
}

dispatch(setError(null));
```

Ilustración 8 - Fragmento del código donde se llama a createUserWithEmailAndPassword

1.2 GESTIÓN DE PROYECTOS

Una vez dentro, el usuario accede al Dashboard, donde puede gestionar sus proyectos. Cada proyecto incluye título, descripción, miembros, y estado (activo/archivado). También puede buscar proyectos por nombre o filtrarlos según su estado.

Las funciones principales que podemos encontrar en la pantalla principal:

- Ver todos los proyectos a los que pertenece.
- Crear un nuevo proyecto mediante un modal con formulario.
- Editar o archivar proyectos existentes.
- Buscar por nombre y filtrar por estado (activos o archivados).
- Invitar a otros usuarios activos como miembros del proyecto.

Toda esta información se almacena y sincroniza en tiempo real con Firebase Realtime Database.

Archivos principales que influyen en esta funcionalidad:

- Dashboard.tsx: vista principal del usuario, incluye listado, filtros y botón de nuevo proyecto.
- ProjectFormModal.tsx: modal con formulario para crear/editar proyectos.
- ProjectCard.tsx: componente visual para cada proyecto.

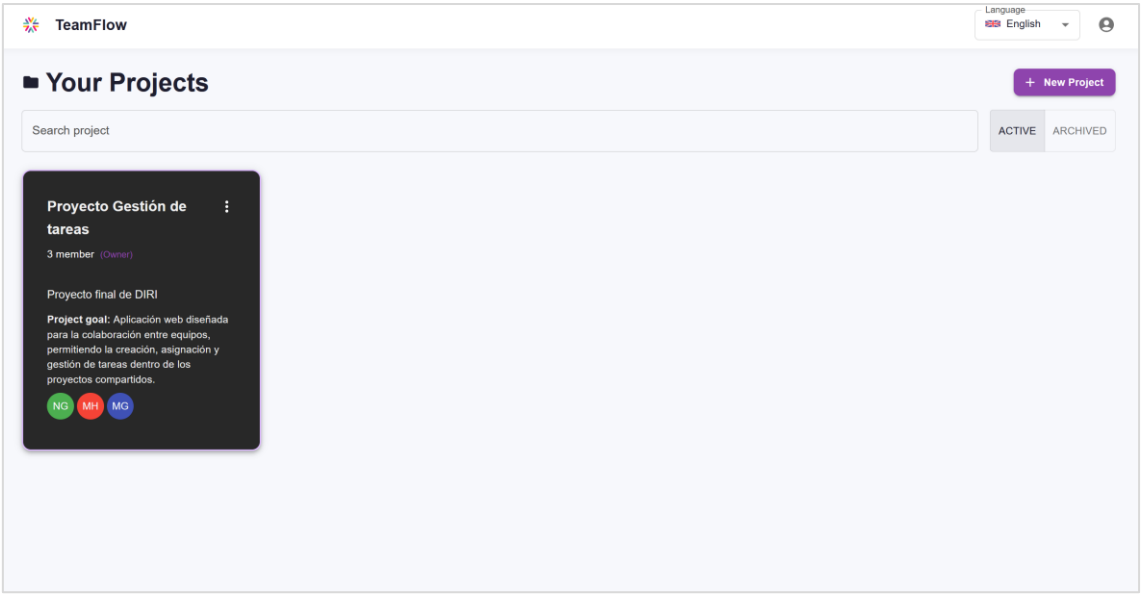

Ilustración 9 - Página principal de un miembro de la app

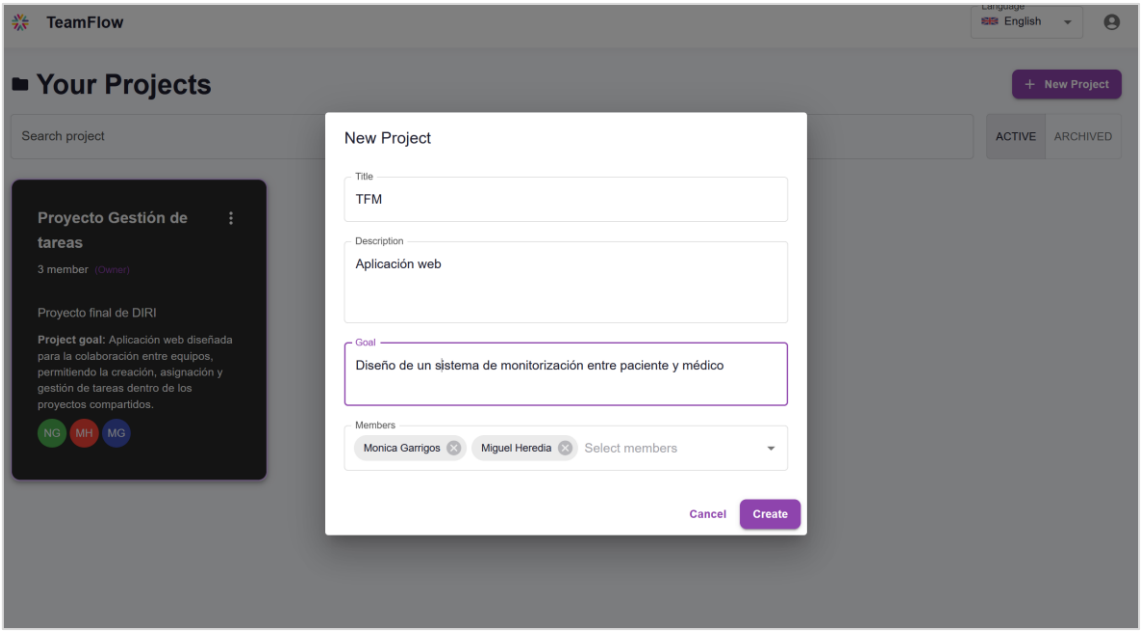

Ilustración 10 - Modal para crear un nuevo proyecto

Ilustración 11 - Opciones en el proyecto

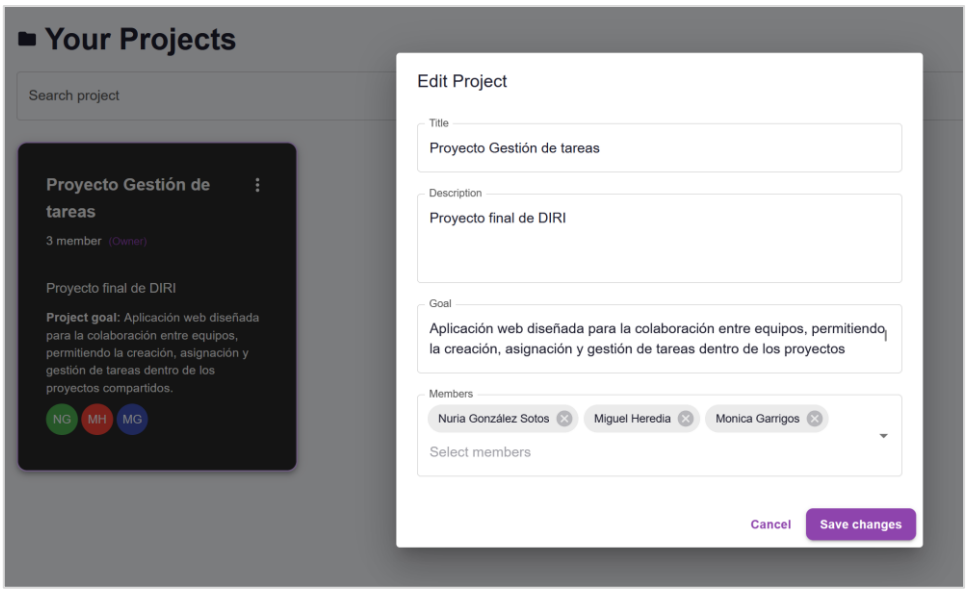

Ilustración 12 - Editar información del proyecto

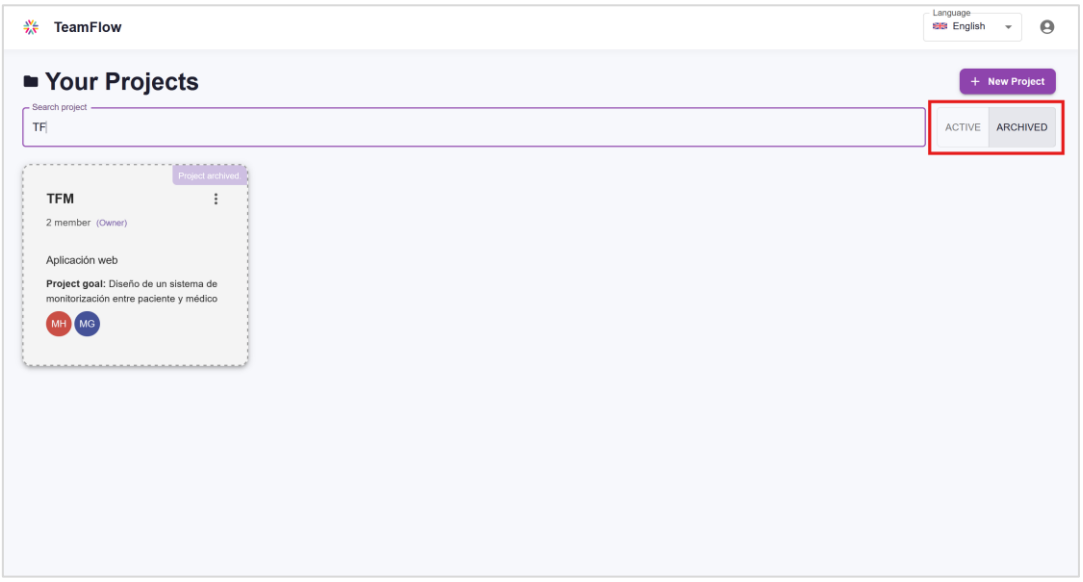

Ilustración 13 - Filtrar proyectos archivados

1.3 GESTIÓN DE TAREAS

Cada proyecto incluye un panel tipo Kanban, con columnas dinámicas (por defecto: "Por hacer", "En progreso", "Hecho"). En este panel se encuentran las siguientes funcionalidades:

La vista de detalle de proyecto ofrece un resumen visual de las tareas categorizadas por estado.

- **Añadir tareas:** Botón "+ Añadir tarea" en cada columna, que abre un modal con formulario para introducir título, descripción, prioridad, responsable, etc.
- **Editar tareas**: Al hacer clic en una tarea, se abre el mismo modal con los datos precargados, permitiendo actualizarlos.

- **Eliminar tareas**: Botón de eliminar dentro del modal o desde la propia tarjeta (si lo implementaste).
- **Drag & Drop**: Las tareas pueden moverse entre columnas gracias a react-beautiful-dnd. Esto actualiza su estado automáticamente en Firebase.
- **Vista resumen por estado**: Las columnas muestran claramente cuántas tareas hay por estado, mejorando la visibilidad del progreso.

Archivos principales implicados:

- *ProjectDetail.tsx*: vista principal del tablero Kanban de un proyecto
- *KanbanColumn*.tsx: componente de columna que agrupa tareas
- *TaskModal*.tsx: modal de creación/edición de tareas
- *Types/task*.ts: tipo de dato Task (id, titulo, estado, etc)

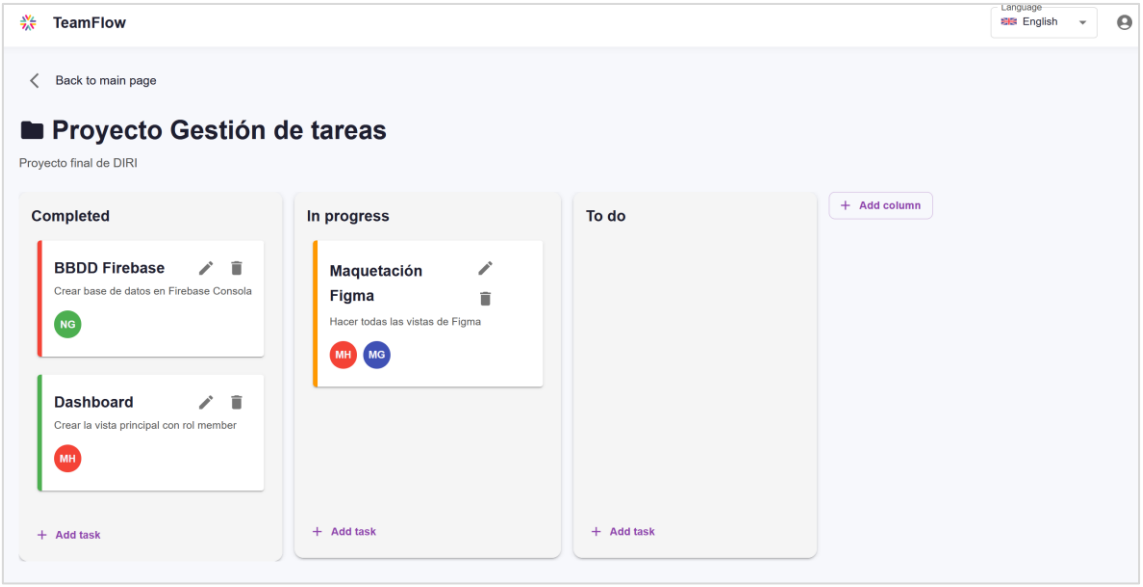

Ilustración 14 - Tablero de tareas de un proyecto

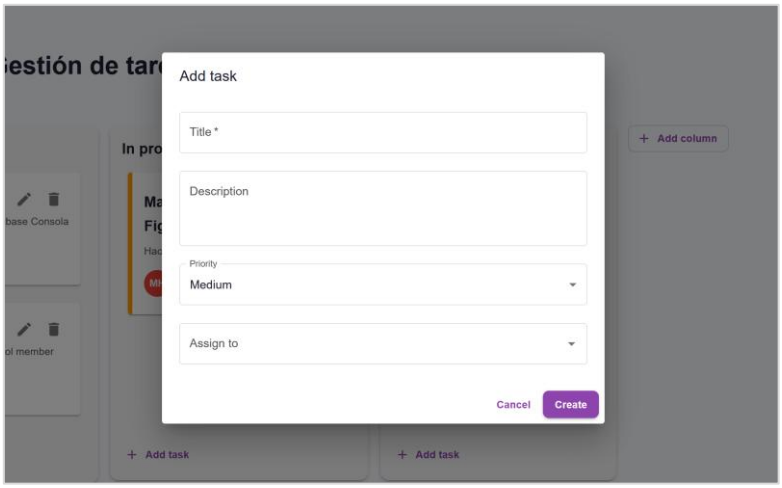

Ilustración 15 - Creación de una nueva tarea

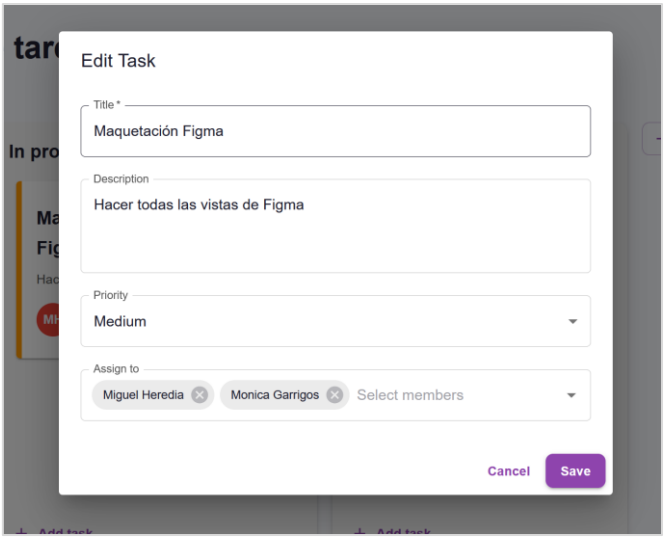

Ilustración 16 - Modal de edición de una tarea

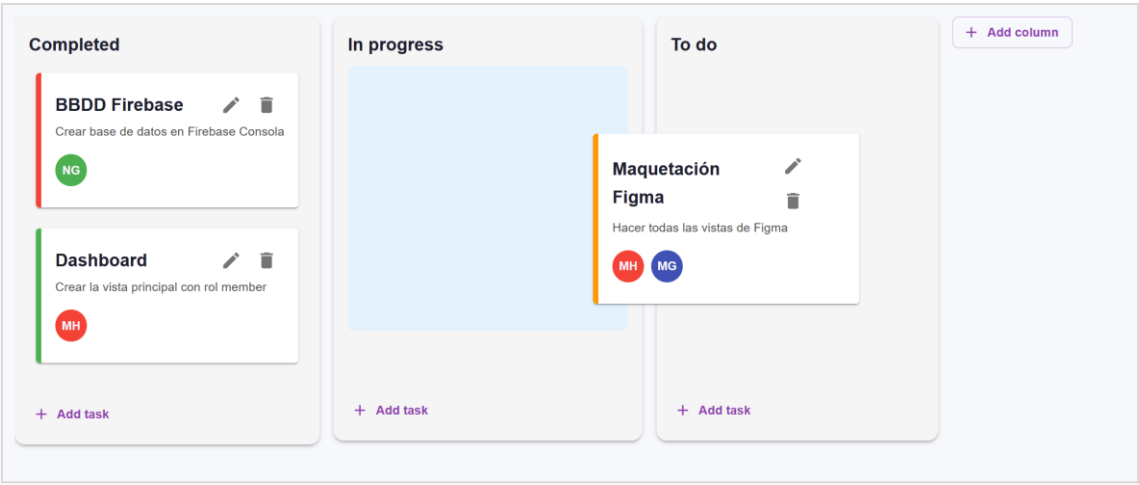

Ilustración 17 - Cambiar el estado de una tarea arrastrando de una columna a otra

1.4 PANEL DE ADMINISTRADOR

Cuando un usuario inicia sesión y tiene el rol admin, accede automáticamente a una vista especial de administración de usuarios. Esta vista permite tener un control centralizado del sistema, especialmente útil en entornos colaborativos donde es necesario gestionar accesos y roles de forma sencilla. Este módulo también se conecta a Firebase para reflejar los cambios en tiempo real y centralizar el control de accesos.

Funcionalidades implementadas:

- **Visualización de usuarios**: Se listan todos los usuarios registrados en la plataforma, mostrando su nombre completo, email, rol y estado (activo o inactivo).
- **Activar / Desactivar usuarios**: Con un solo clic, el administrador puede bloquear (desactivar) o reactivar usuarios. Esta acción modifica el campo active en Firebase y se refleja en tiempo real en toda la aplicación.
- **Gestión de roles**: Se permite cambiar el rol del usuario (por ejemplo, de "miembro" a "admin"). Esto afecta al acceso a funciones dentro de la app.

- **Sincronización en tiempo real**: Todos los cambios se reflejan al momento gracias a Firebase Realtime Database, lo que garantiza una administración eficaz.

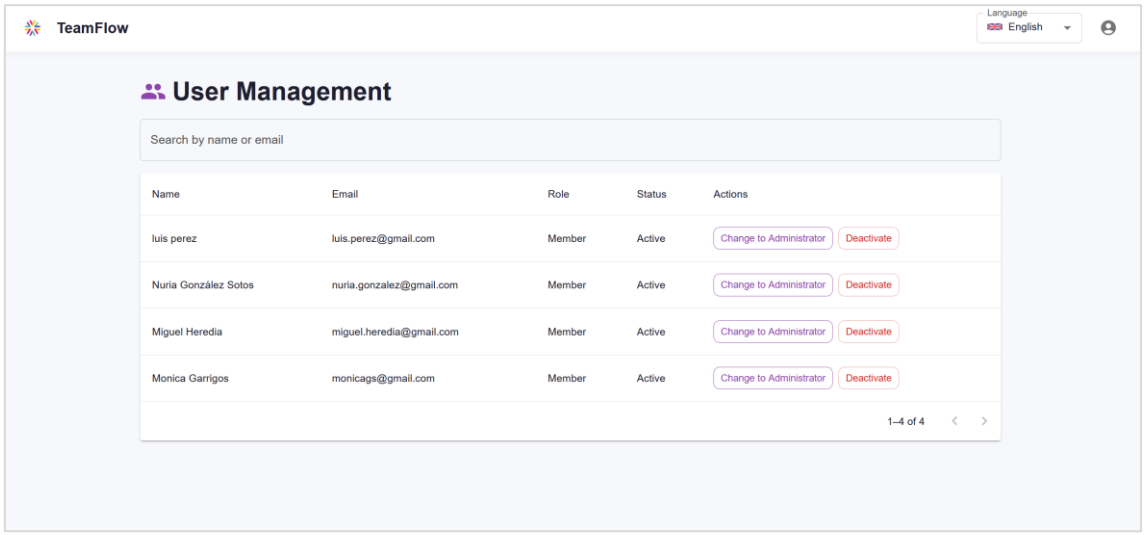

Ilustración 18 - Panel de administración

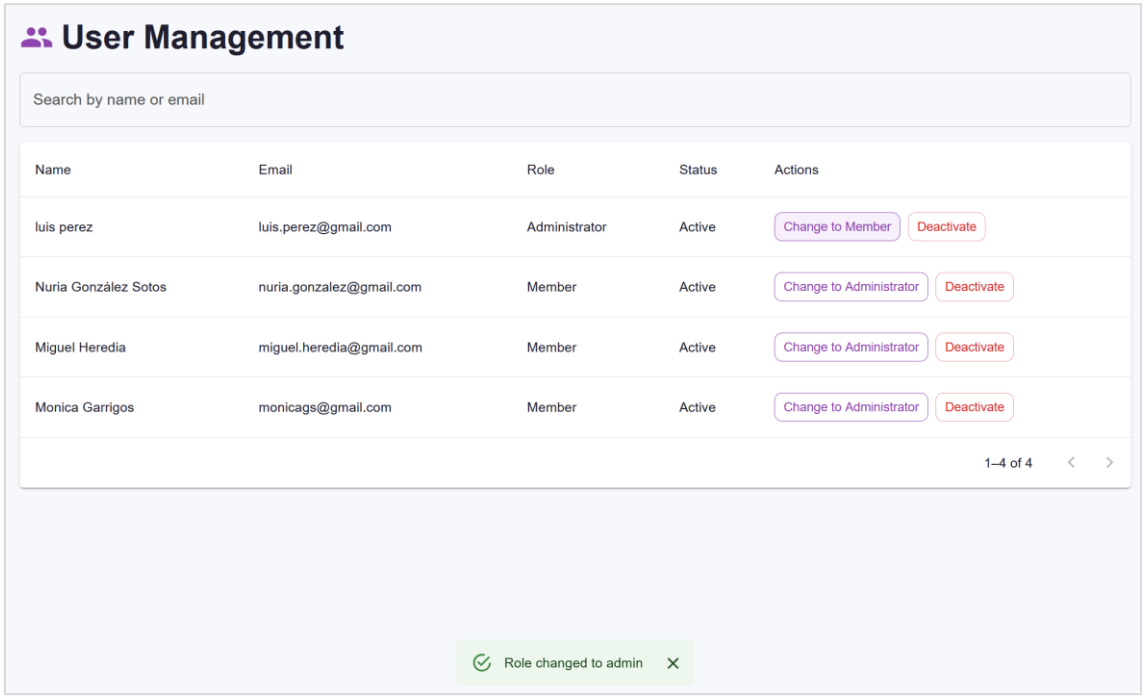

Ilustración 19 - Cambiar usuario luis a administrador

1.5 INTERNACIONALIZACIÓN (I18N)

Se ha implementado un sistema de internacionalización utilizando la herramienta i18next, que permite mostrar todos los textos de la aplicación en español o inglés, según la preferencia del usuario. Desde la pantalla de inicio de sesión, el usuario puede elegir su idioma preferido mediante un selector de idioma situado en la parte superior. Este cambio es inmediato y se aplica a toda la interfaz de la aplicación, incluidos los formularios, botones, mensajes de error y notificaciones.

Los archivos utilizados para realizar esta implementación han sido:

- Index.ts: configuración del sistema de idiomas
- Es.json y en.json: traducciones al español y al inglés
- languageSwitcher.tsx: componente del selector de idioma

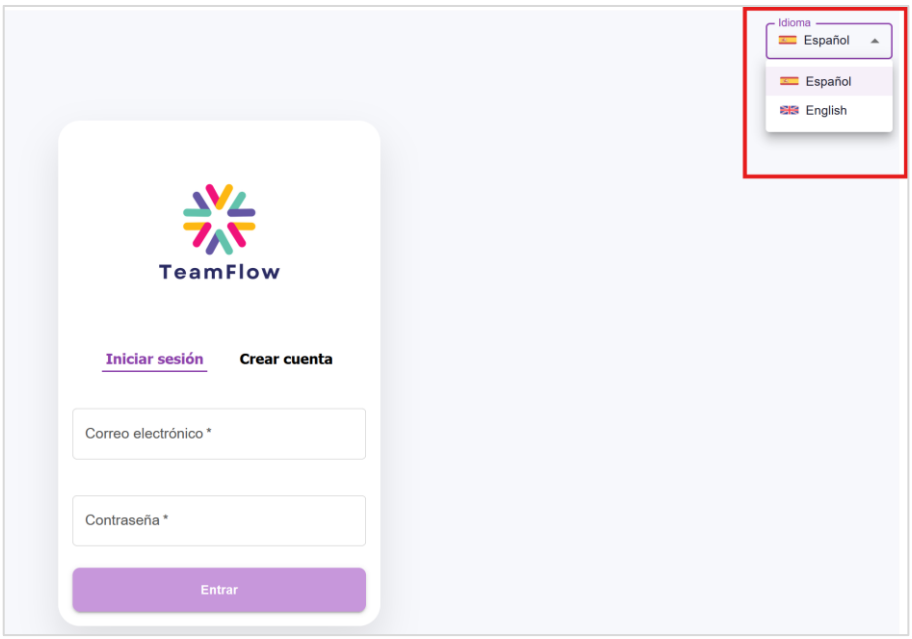

Ilustración 20 - Selector para cambiar de idioma en login

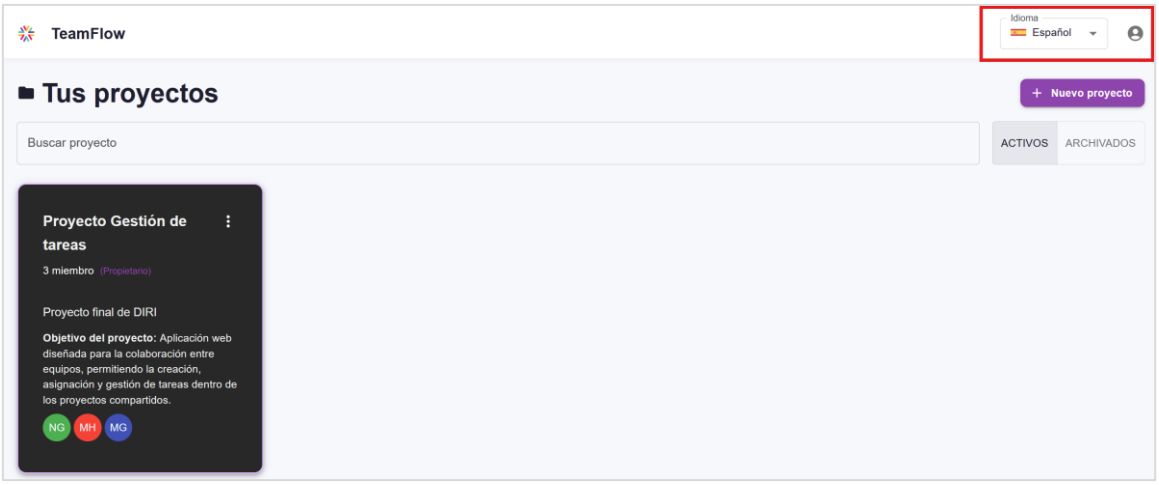

Ilustración 21 - Selector de idioma en el topbar

```
18n > index.ts > ...
import i18n from "i18next";
import { initReactI18next } from "react-i18next";

import en from "./en.json";
import es from "./es.json";

const savedLang = localStorage.getItem("appLang") || "es";

i18n.use(initReactI18next).init({
  resources: {
    en: { translation: en },
    es: { translation: es }
  },
  lng: savedLang,
  fallbackLng: "en",
  interpolation: {
    escapeValue: false
  }
});

export default i18n;
```

Ilustración 22 - Configuración del sistema de idiomas

1.6 LOGGING

Este módulo permite registrar en consola los eventos más importantes que ocurren durante el uso de TeamFlow. El objetivo es ayudar durante el desarrollo, facilitar la depuración de errores y en el futuro permitir auditar el uso del sistema.

Actualmente, se registran eventos como:

- Inicios de sesión exitosos o fallidos.
- Registro de nuevos usuarios.
- Creación o edición de proyectos y tareas.
- Errores importantes detectados tanto por el usuario como por el sistema.

El módulo está centralizado en el archivo logger.ts, donde se definen funciones como logInfo, logWarn y logError.

Archivos involucrados:

- *utils/logger.ts:* contiene la lógica principal del módulo de logs.

```
utils > logger.ts > ...
const isProd = import.meta.env.MODE === 'production';

export const logInfo = (...args: any[]) => {
  if (!isProd) console.info('[INFO]', ...args);
};

export const logWarn = (...args: any[]) => {
  console.warn('[WARN]', ...args);
};

export const logError = (...args: any[]) => {
  console.error('[ERROR]', ...args);
};
```

Ilustración 23 - Lógica principal del módulo de logs

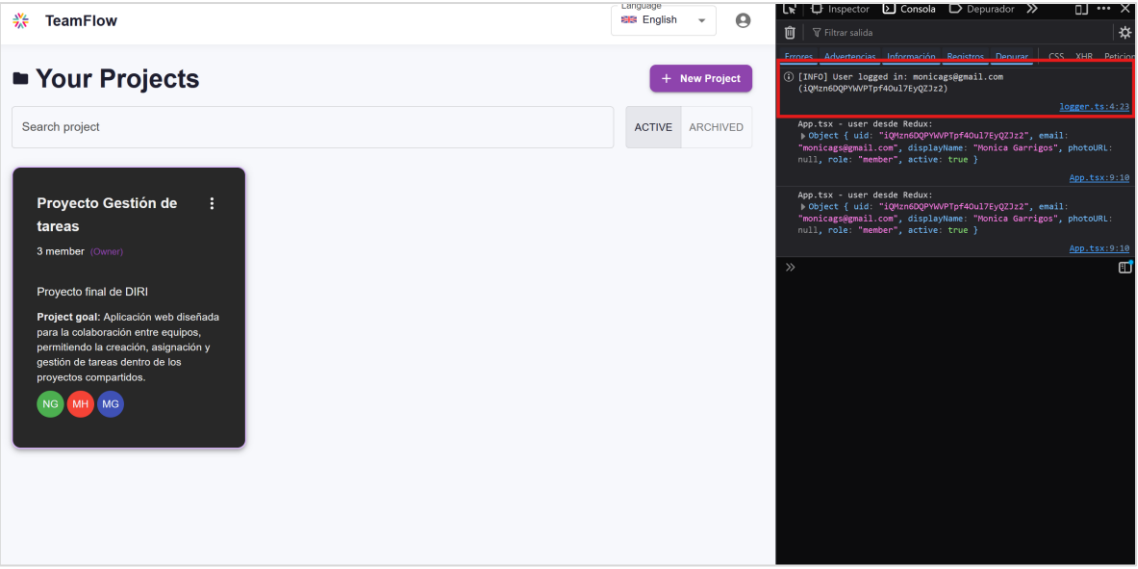

Ilustración 24 - Ejemplo de logs de tipo [INFO]

1.7 PRUEBAS UNITARIAS

Se han creado pruebas unitarias para componentes clave usando Vitest y @testing-library/react. Se ha testeado:

- El correcto renderizado del formulario de autenticación.
- Comportamiento de los inputs al escribir.
- Visualización de botones y mensajes de error.

Estas pruebas garantizan el correcto funcionamiento de la UI en situaciones críticas y facilitan futuras refactorizaciones.

```
PS C:\Users\monic\Desktop\Master UA\DIRI\Proyecto final\teamflow> npx vitest
>>

 DEV  v3.2.2 C:/Users/monic/Desktop/Master UA/DIRI/Proyecto final/teamflow

 ✓ src/components/AuthForm.test.tsx (3 tests) 954ms
   ✓ AuthForm UI tests > Should render login and register buttons 129ms
   ✓ AuthForm UI tests > Should toggle between login and register forms 278ms
   ✓ AuthForm UI tests > Should update input values as user types  544ms

 Test Files  1 passed (1)
      Tests  3 passed (3)
   Start at  22:51:22
   Duration  4.82s (transform 229ms, setup 337ms, collect 1.91s, tests 954ms, environment 1.18s, prepare 116ms)

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Ilustración 25 - Pruebas unitarias ejecutadas

```
describe('AuthForm UI tests', () => {
  afterEach(() => {
    cleanup();
  });

  const renderAuthForm = () =>
    render(
      <Provider store={store}>
        <I18nextProvider i18n={i18n}>
          <AuthForm />
        </I18nextProvider>
      </Provider>
    );

  it('Should render login and register buttons', async () => {
    renderAuthForm();

    // Adaptado al texto visible en el botón
    const loginButton = await screen.findByText('Iniciar sesión');
    const registerButton = await screen.findByText('Crear cuenta');

    expect(loginButton).toBeInTheDocument();
    expect(registerButton).toBeInTheDocument();
  });
```

Ilustración 26 - Fragmento de código del test AuthForm

7. CONCLUSIÓN

El desarrollo de TeamFlow ha sido una experiencia muy completa que ha permitido aplicar de forma práctica muchos conceptos clave del desarrollo web moderno. A lo largo del proyecto se han utilizado tecnologías como React, Redux Toolkit, Firebase o Material UI, integrando además funcionalidades avanzadas como autenticación de usuarios, gestión en tiempo real, internacionalización, control de roles y pruebas unitarias. Gracias a esta combinación de herramientas, la aplicación resultante es modular, funcional y fácilmente ampliable. La estructura del código favorece el mantenimiento y futuras mejoras, y la experiencia de usuario es clara, rápida y fluida.

Además, se han seguido buenas prácticas como el uso de TypeScript, la división del código en módulos reutilizables, la gestión centralizada del estado o el registro de eventos importantes mediante un sistema de logging personalizado.

En resumen, se ha cumplido con éxito el objetivo principal: entregar una aplicación web profesional que facilite la colaboración en equipo, centralizando toda la gestión de proyectos y tareas en una sola plataforma clara, accesible y adaptable.