

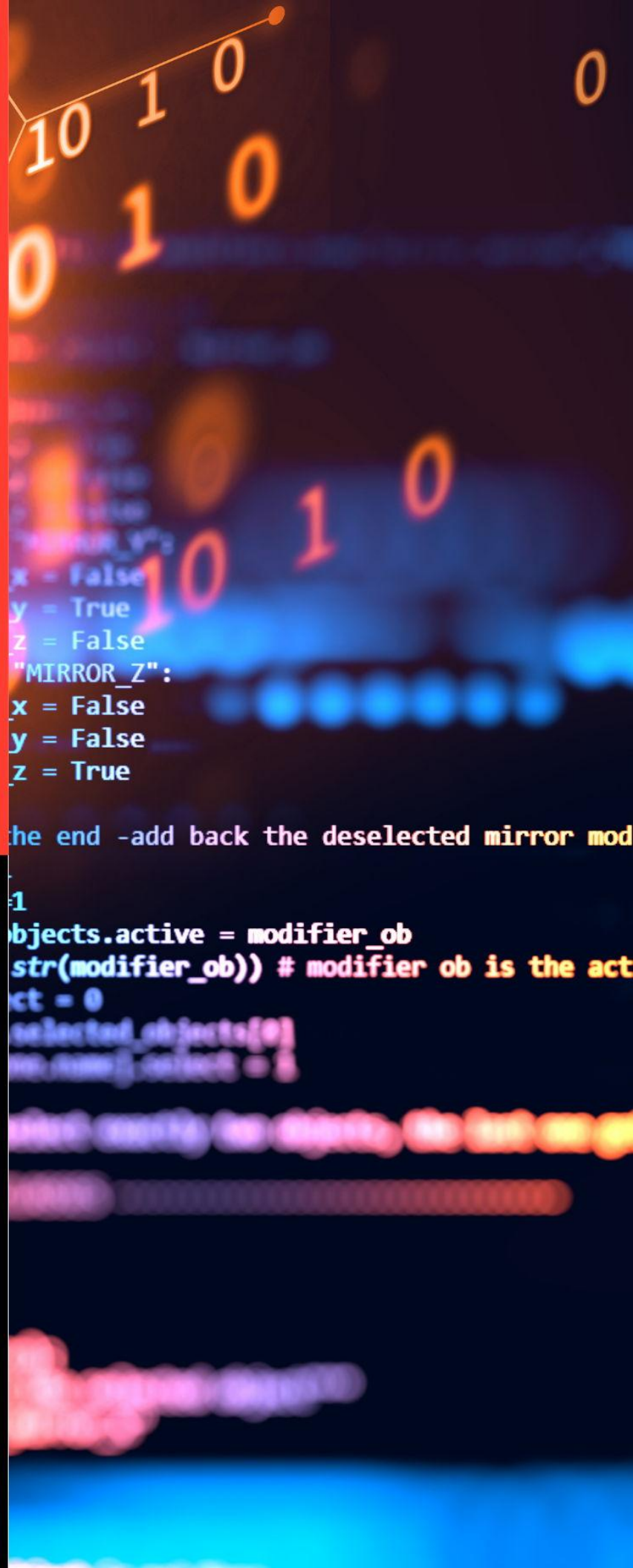
# Proyecto Final

Gestor de Tareas  
(MERN)

Desarrollo de  
Aplicaciones Web

Daniel Pintado Várez

Fecha: 28 de Mayo de 2025



## 1. Descripción General

Este proyecto es una aplicación web fullstack para la gestión de tareas, desarrollada con React en el frontend y Node.js/Express en el backend, utilizando MongoDB como base de datos. Incluye autenticación segura, protección CSRF, validación robusta y una interfaz moderna y responsiva.

---

## 2. Manual de Usuario

### 2.1 Frontend (Usuario Final)

**Registro de Usuario** - Accede a la página de registro. - Completa el formulario con tu nombre de usuario, correo y contraseña. - Recibirás un mensaje de confirmación y podrás iniciar sesión.

**Inicio de Sesión** - Ingresa tu usuario o correo y contraseña en la página de login. - Si los datos son correctos, accederás a tu panel de tareas.

**Gestión de Tareas** - Puedes crear nuevas tareas desde el panel principal. - Edita el título, descripción o marca como completada cualquier tarea. - Elimina tareas que ya no necesites. - Todas las acciones muestran notificaciones de éxito o error.

**Navegación y Temas** - Usa la barra de navegación para moverte entre secciones. - Cambia entre modo claro y oscuro desde el menú. - El menú es responsivo y se adapta a móviles y escritorio.

**Cerrar Sesión** - Haz clic en el botón de cerrar sesión en el Navbar para salir de forma segura.

## 2.2 Backend (Usuario Administrador)

**Inicio y Configuración** - Instala las dependencias con `npm install`. - Configura el archivo `.env` con las variables necesarias. - Inicia el servidor con `npm start`.

**Gestión de Usuarios y Tareas** - Los usuarios se gestionan automáticamente mediante los endpoints de registro y login. - Las tareas se almacenan y gestionan en MongoDB. - Puedes consultar, editar o eliminar usuarios/tareas directamente en la base de datos si tienes permisos de administrador.

**Seguridad y Logs** - El backend registra todas las peticiones importantes. - Los errores y accesos no autorizados quedan registrados en consola. - El sistema está protegido contra CSRF y requiere autenticación para acceder a datos sensibles.

---

## 3. Manual para el Programador

### 3.1 Frontend

- **Estructura:** Un componente por archivo, imports organizados, uso de Context API para estado global.
- **Estilos:** Utiliza Tailwind CSS, prioriza utilidades sobre CSS personalizado.
- **Buenas prácticas:** Nombres consistentes, documentación en componentes reutilizables.
- **Extensión:** Puedes añadir nuevas páginas, componentes o hooks siguiendo la estructura existente.
- **Validación:** Los formularios usan validación en frontend y backend.
- **Notificaciones:** Usa React Toastify para feedback visual.
- **Autenticación:** El contexto de autenticación gestiona el estado global y la persistencia de sesión.

### 3.2 Backend

- **Estructura:** Código modular, rutas separadas, controladores claros y modelos Mongoose bien definidos.
- **Seguridad:** Implementa CSRF, CORS, JWT y validación de datos con Zod.
- **Extensión:** Puedes añadir nuevos endpoints, modelos o middlewares siguiendo la arquitectura actual.
- **Manejo de errores:** Usa respuestas claras y consistentes, captura errores en controladores y middlewares.
- **Documentación:** Comenta funciones complejas y mantén el README actualizado.
- **Pruebas:** Se recomienda añadir tests para endpoints críticos y lógica de negocio.

## 4. Backend

### 4.1 Tecnologías y Dependencias

- **Node.js:** Entorno de ejecución para JavaScript en el servidor.
- **Express:** Framework web para Node.js.
- **MongoDB:** Base de datos NoSQL.
- **Mongoose:** ODM para MongoDB.
- **Morgan:** Middleware de logging para peticiones HTTP.
- **JWT (JsonWebToken):** Autenticación basada en tokens.
- **Bcrypt:** Hasheo de contraseñas (opcional, preparado para producción).
- **Cookie Parser:** Manejo de cookies en Express.
- **CSURF:** Protección contra ataques CSRF.

### 4.2 Instalación y Configuración

#### 1. Instalación de dependencias

```
npm install
```

#### 2. Variables de entorno

Crear un archivo `.env` en la raíz del backend con:

```
PORT=4000  
MONGO_URI=tu_mongo_uri  
JWT_SECRET=tu_jwt_secret
```

#### 3. Ejecución del servidor

```
npm start
```

### 4.3 Arquitectura y Estructura

- **src/app.js:** Configuración principal de Express, middlewares y rutas.
- **src/routes/:** Rutas de autenticación (`auth.routes.js`) y tareas (`task.routes.js`).
- **src/controllers/:** Lógica de negocio para autenticación y tareas.
- **src/models/:** Modelos de datos Mongoose (`user.model.js`, `task.model.js`).
- **src/middlewares/:** Validación de esquemas y autenticación de tokens.
- **src/libs/jwt.js:** Utilidades para generación y verificación de JWT.
- **src/schemas/:** Esquemas de validación con Zod.

#### 4.4 Seguridad y Buenas Prácticas

- **CSRF:** Implementado con CSURF y cookies.
- **CORS:** Solo permite origen del frontend (`http://localhost:5173`).
- **JWT:** Tokens seguros, almacenados en cookies `httpOnly`.
- **Validación:** Uso de Zod para validar y sanear entradas.
- **Manejo de errores:** Respuestas claras y consistentes.

#### 4.5 Endpoints Principales

- **/api/register:** Registro de usuario (POST)
- **/api/login:** Inicio de sesión (POST)
- **/api/logout:** Cierre de sesión (POST)
- **/api/profile:** Perfil de usuario autenticado (GET, requiere token)
- **/api/tasks:** CRUD de tareas (GET, POST, PUT, DELETE, requiere token)

#### 4.6 Flujo de Autenticación

1. El usuario se registra y recibe un token JWT en una cookie segura.
  2. El login valida credenciales y renueva el token.
  3. El token se verifica en cada petición protegida.
  4. Logout elimina la cookie del token.
-

## 5. Frontend

### 5.1 Tecnologías y Dependencias

- **React:** Librería para construir interfaces de usuario.
- **Tailwind CSS:** Utilidades para estilos rápidos y responsivos.
- **Axios:** Cliente HTTP con interceptores personalizados.
- **React Router DOM:** Enrutamiento SPA.
- **React Toastify:** Notificaciones.
- **Lucide React:** Iconografía moderna.
- **Context API:** Estado global para autenticación, tareas y tema.

### 5.2 Instalación y Configuración

#### 1. Instalación de dependencias

```
npm install
```

#### 2. Variables de entorno

Crear un archivo `.env` en la raíz del frontend con:

```
VITE_API_URL=http://localhost:4000/api
```

#### 3. Ejecución de la aplicación

```
npm start
```

### 5.3 Estructura de Carpetas

```
src/
├── api/           # Servicios y llamadas API
├── components/    # Componentes reutilizables
├── context/       # Contextos globales
├── pages/         # Componentes de página
├── services/      # Servicios utilitarios (hash, etc)
└── styles/       # Estilos globales y personalizados
```

## 5.4 Componentes y Contextos Clave

- **AuthContext:** Maneja autenticación, persistencia y verificación de sesión.
- **ThemeContext:** Alternancia y persistencia de tema claro/oscuro.
- **TasksContext:** Estado global de tareas del usuario.
- **Navbar:** Navegación responsiva, muestra usuario y logout.
- **TaskCard:** Tarjeta editable para cada tarea.
- **ProtectedRoute:** Protege rutas privadas según autenticación.

## 5.5 Funcionalidades Principales

- **Registro/Login:** Formularios validados, feedback visual, hash de contraseña en frontend.
- **Gestión de Tareas:** CRUD completo, edición inline, feedback inmediato.
- **Notificaciones:** Toasts para éxito/error en todas las acciones.
- **Responsive:** Layouts adaptativos, menú móvil, dark mode.
- **Seguridad:** CSRF automático, manejo de tokens, validación robusta.

## 5.6 Ejemplo de Uso de la API

```
// Registro
await register({ user, email, password });
// Login
await login({ user, password });
// Obtener tareas
await fetchUserTasks();
```

## 6. Seguridad

### 6.1 Medidas Implementadas

- **CSRF:** Protección automática en todas las rutas sensibles.
- **CORS:** Solo permite origen del frontend.
- **JWT:** Tokens httpOnly, verificación en cada request.
- **Validación:** Esquemas Zod en backend, validación en frontend.
- **Hash de Contraseña:** SHA-256 en frontend, preparado para bcrypt en backend.

### 6.2 Buenas Prácticas

- Sanitización de entradas
  - Manejo de errores consistente
  - Renovación automática de tokens
  - Logout Seguro
-



## 7. Responsive Design

### 7.1 Breakpoints y Adaptabilidad

- **sm:** 640px (móviles grandes)
- **md:** 768px (tablets)
- **lg:** 1024px (desktop)
- **xl:** 1280px (desktop grande)

### 7.2 Características

- Menú hamburguesa en móvil
  - Layouts fluidos y tarjetas adaptativas
  - Tipografía y botones escalables
  - Soporte completo para dark mode
- 

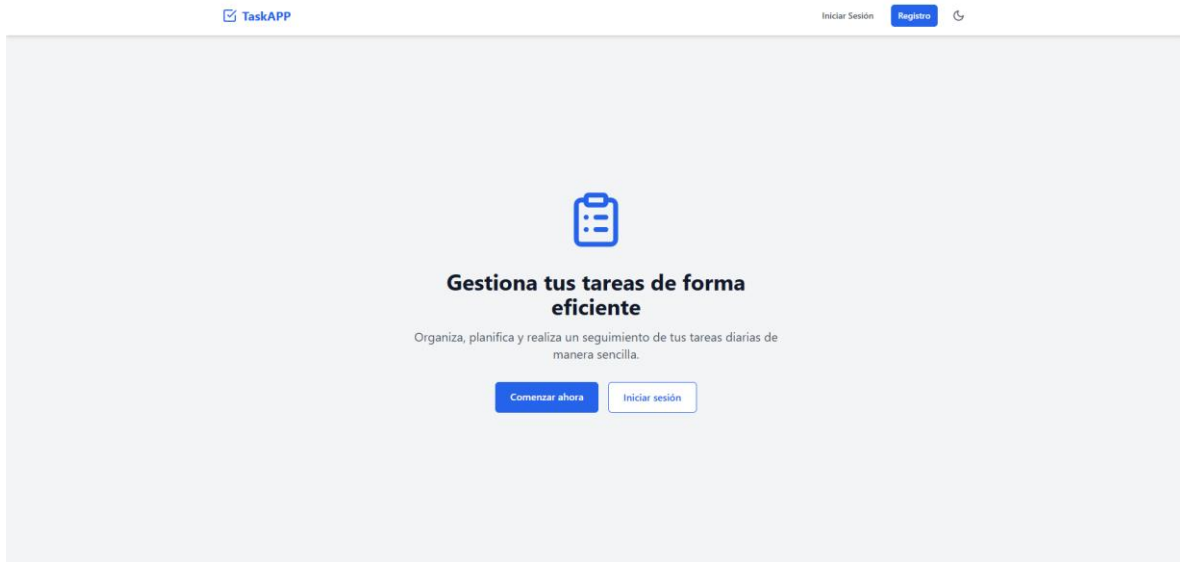
## 8. Cambios y Mejoras Recientes

- Navbar muestra el usuario autenticado y permite logout seguro.
  - Unificación de perfil y tareas en TasksPage.
  - Edición inline de tareas y feedback visual inmediato.
  - Mejoras en validación y feedback de formularios.
  - Refactorización de la arquitectura para mayor escalabilidad.
-

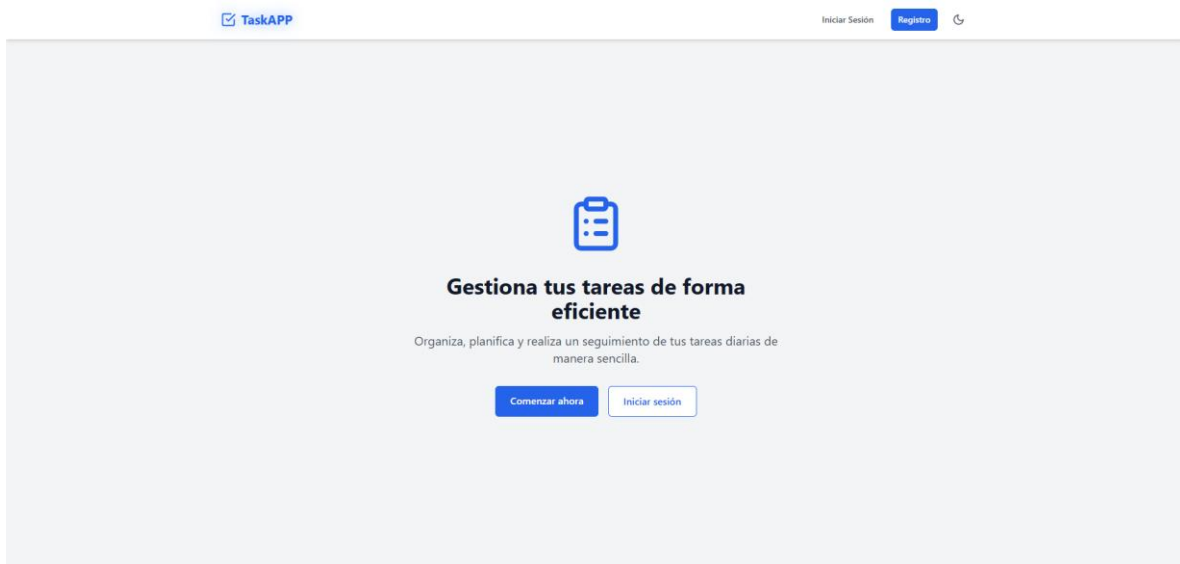
## 9. Demostración de Uso.

### 9.1 Efectos Visuales

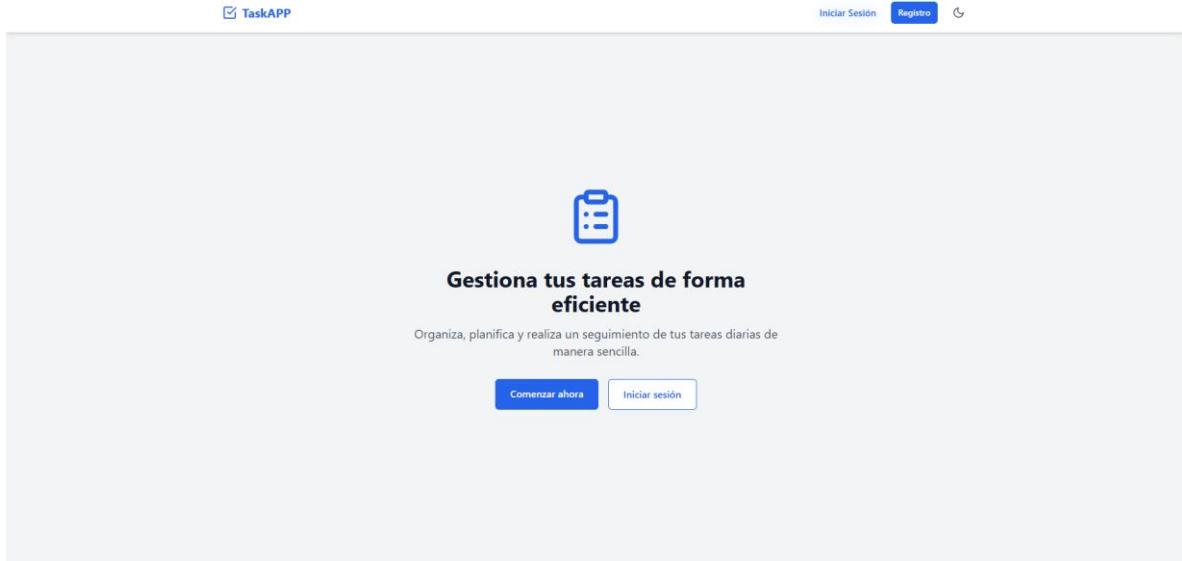
- 1



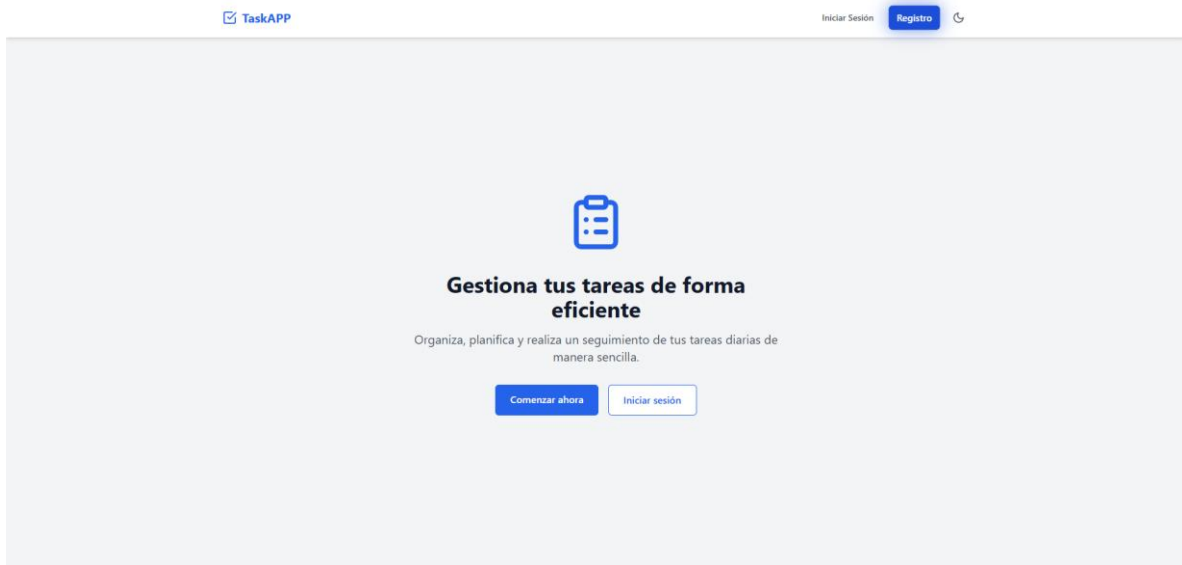
- 2



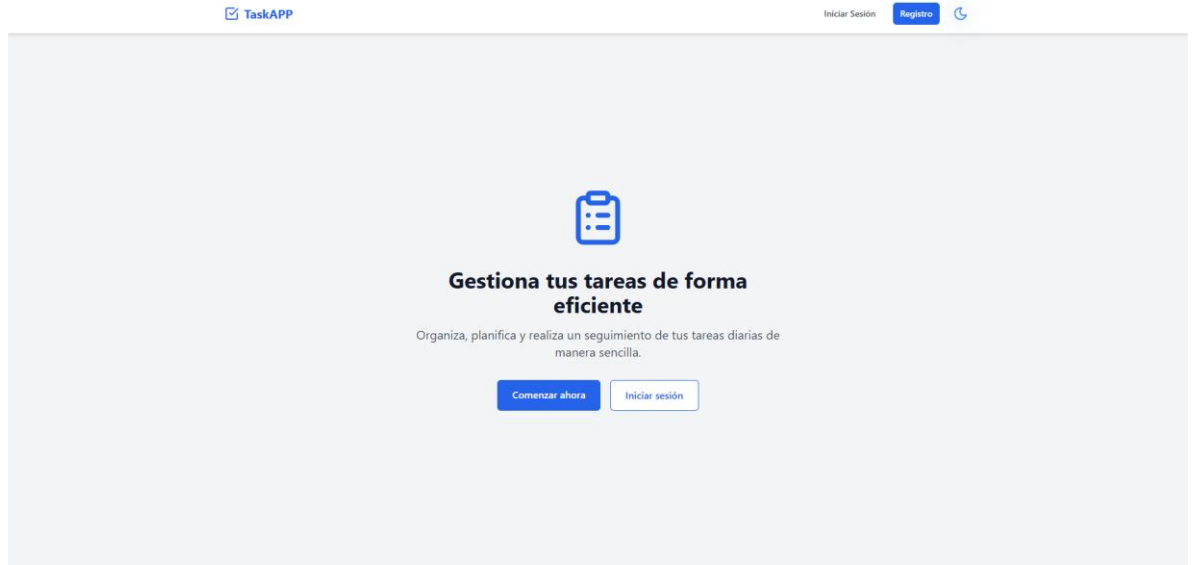
- 3



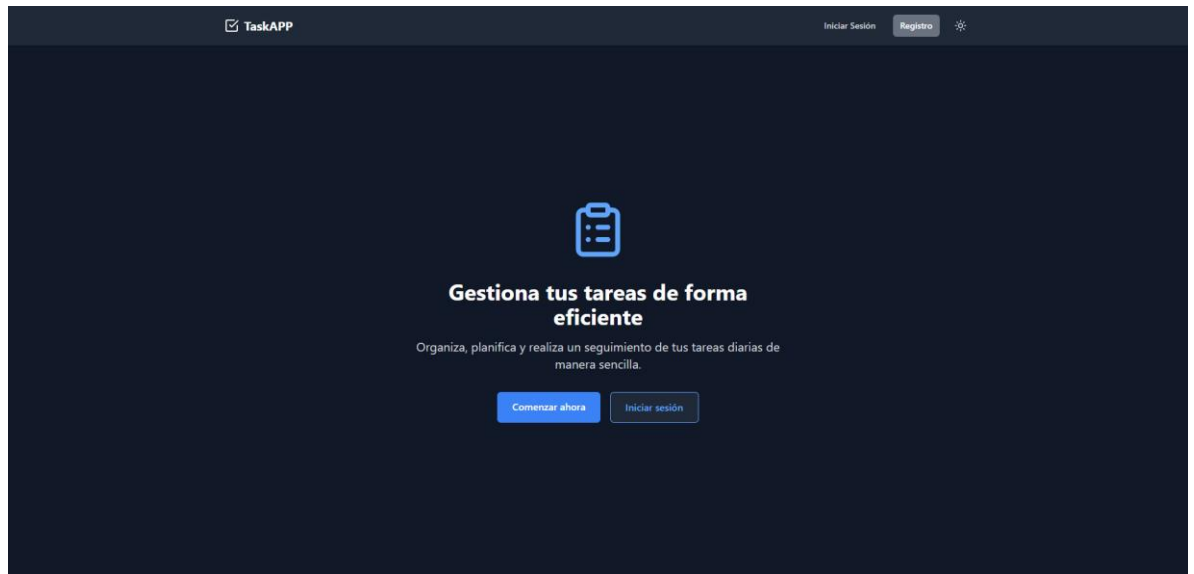
- 4



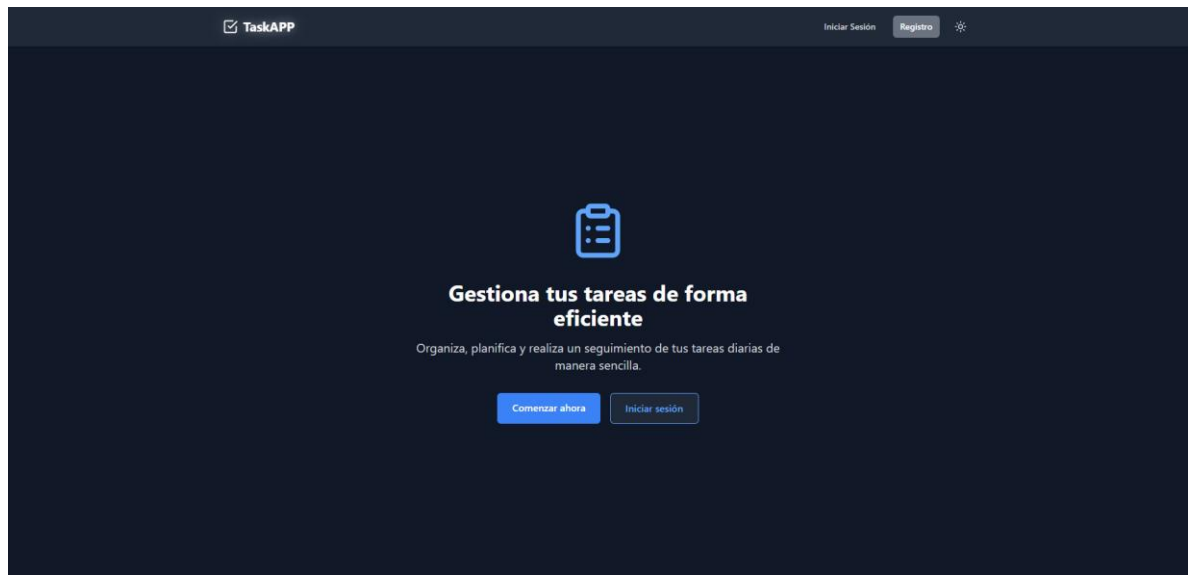
- 5



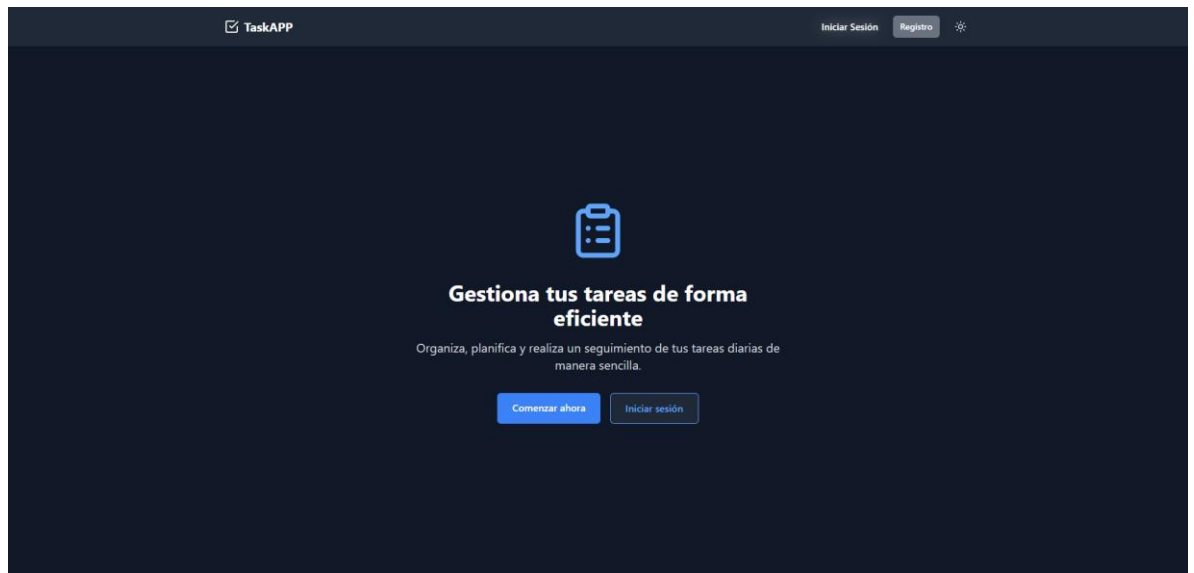
- 6



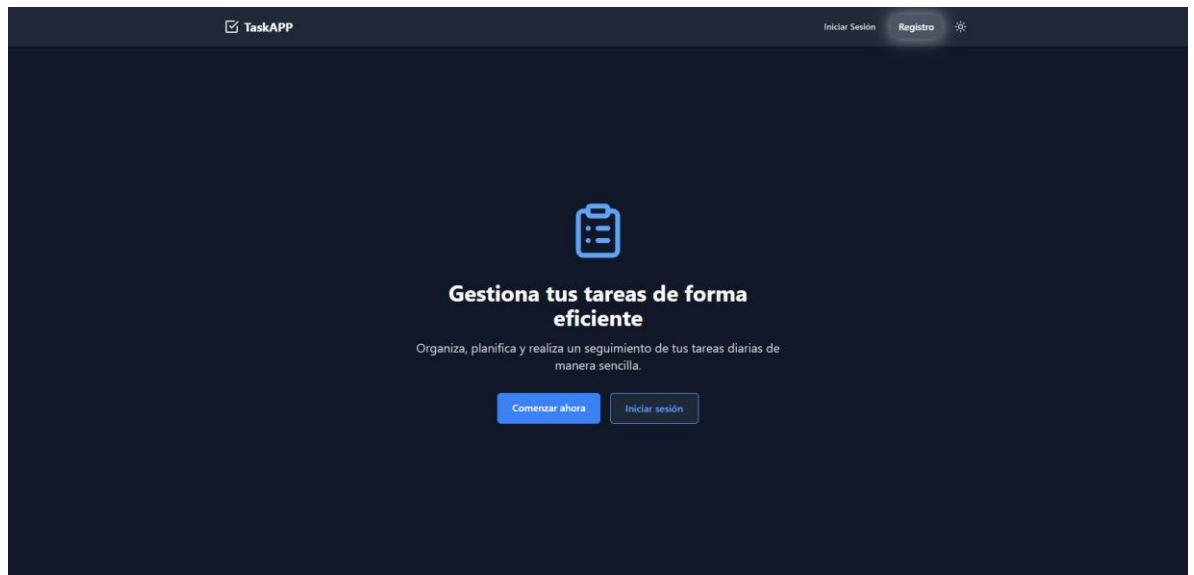
- 7



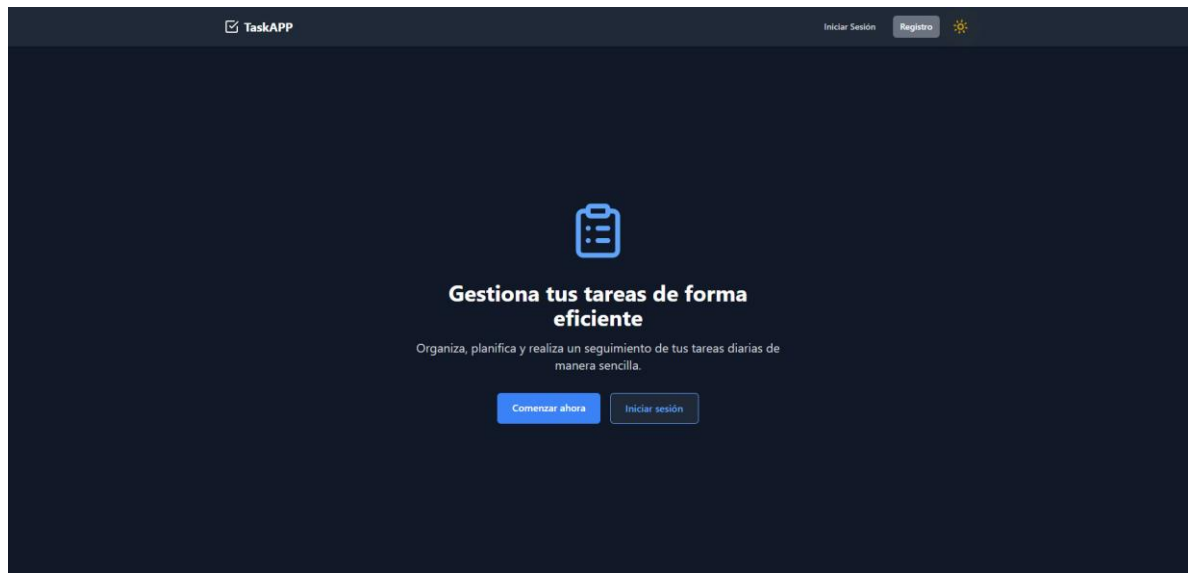
- 8



- 9

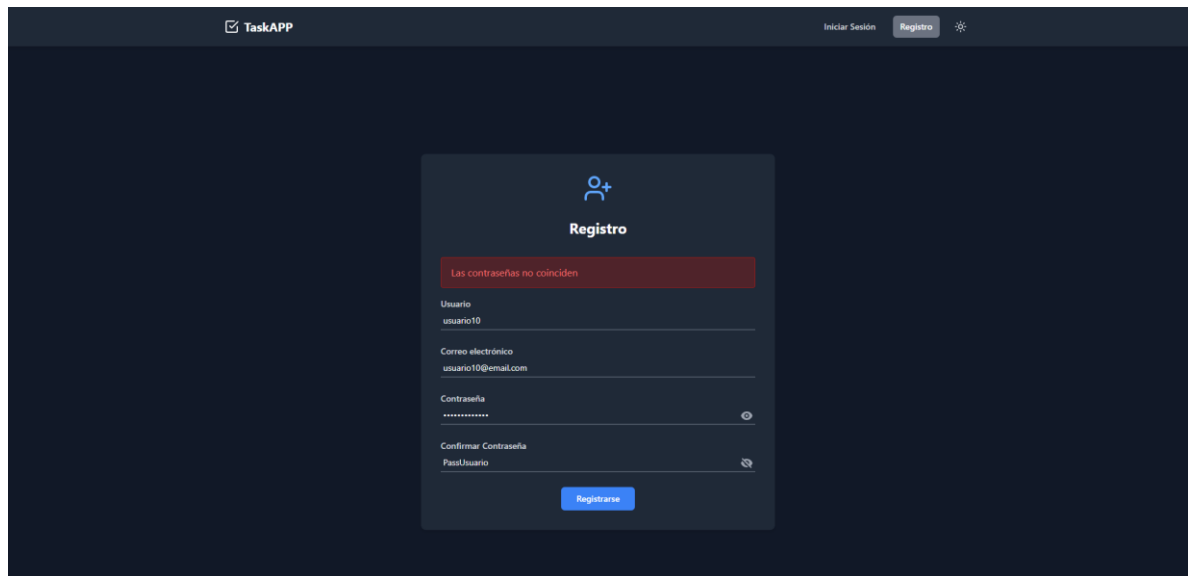


- 10



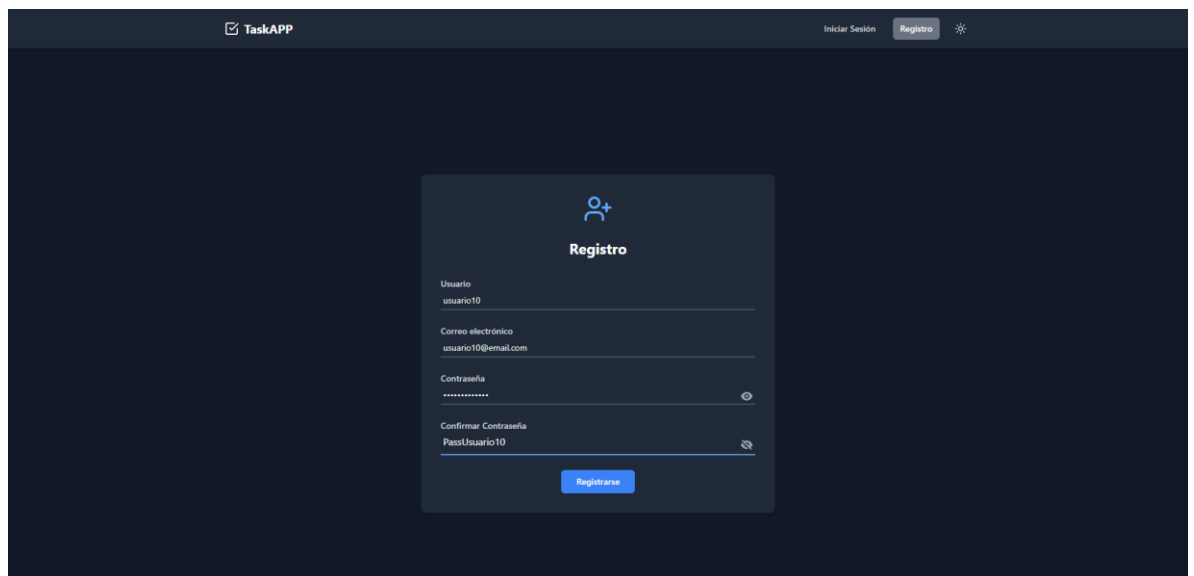
## 9.2 Registro y Login

- 11



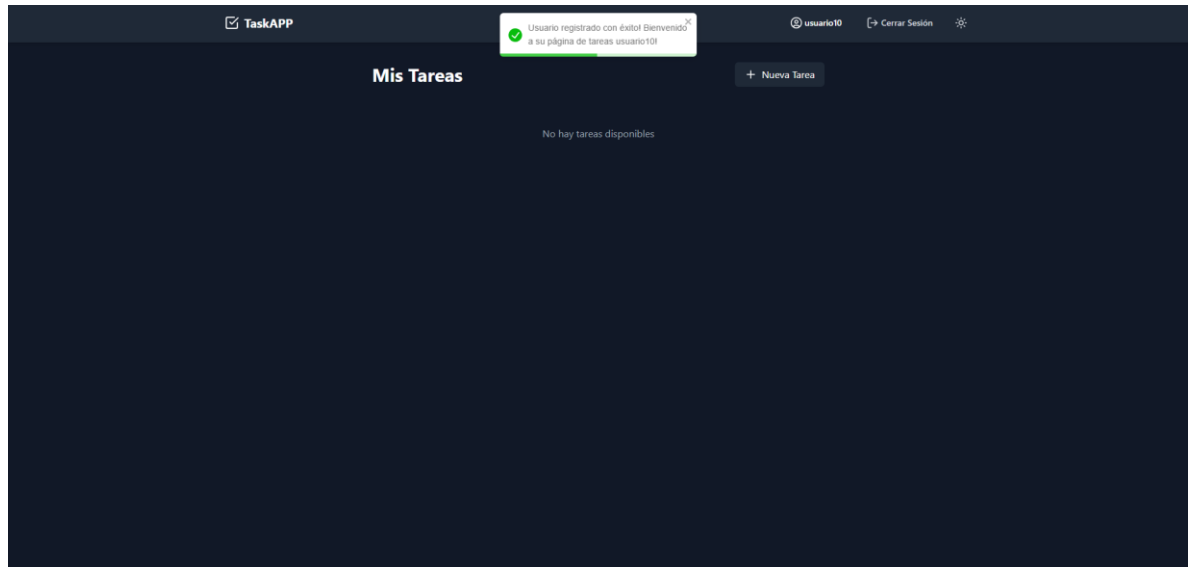
The screenshot shows the 'Registro' (Registration) page of the TaskAPP. The page has a dark blue header with the TaskAPP logo and navigation links for 'Iniciar Sesión' and 'Registro'. The main content area is a dark blue card with a user icon and the title 'Registro'. A red error message at the top of the form states 'Las contraseñas no coinciden' (Passwords do not match). The form fields are: 'Usuario' (username) with the value 'usuario10', 'Correo electrónico' (email) with the value 'usuario10@email.com', 'Contraseña' (password) with masked characters, and 'Confirmar Contraseña' (confirm password) with the value 'PassUsuario'. A blue 'Registrarse' button is at the bottom of the form.

- 12

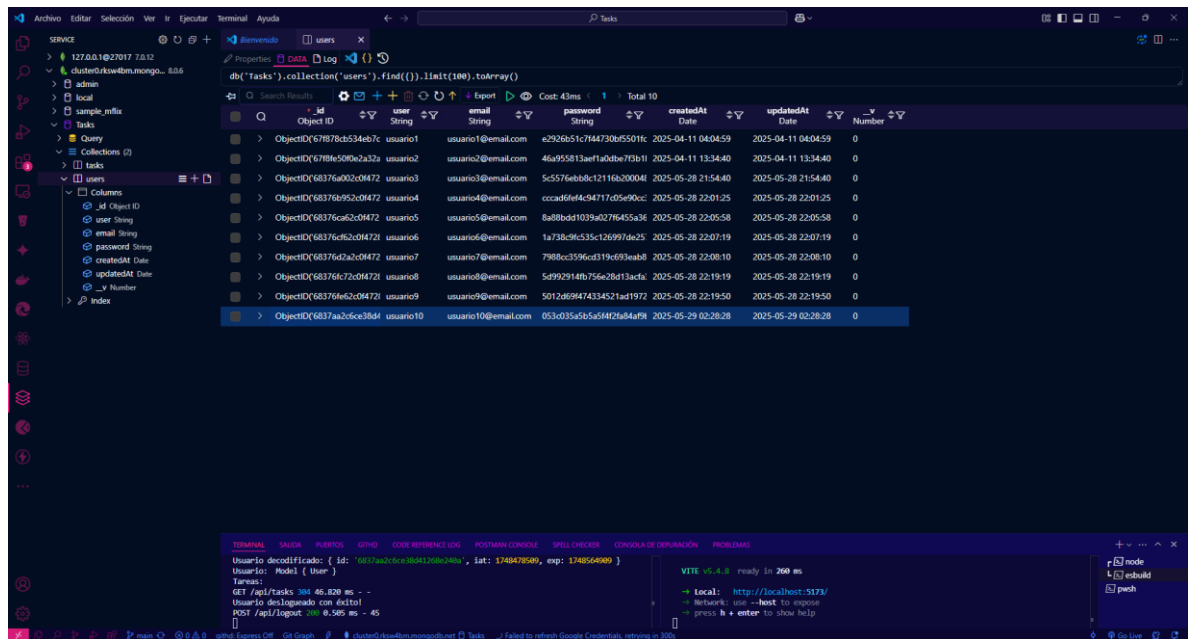


The screenshot shows the 'Registro' (Registration) page of the TaskAPP. The page has a dark blue header with the TaskAPP logo and navigation links for 'Iniciar Sesión' and 'Registro'. The main content area is a dark blue card with a user icon and the title 'Registro'. The form fields are: 'Usuario' (username) with the value 'usuario10', 'Correo electrónico' (email) with the value 'usuario10@email.com', 'Contraseña' (password) with masked characters, and 'Confirmar Contraseña' (confirm password) with the value 'PassUsuario10'. A blue 'Registrarse' button is at the bottom of the form.

- 13



- 14





- 15

TaskAPP

Iniciar SesiónRegistro

→]

Iniciar Sesión

Usuario o Correo Electrónico

usuario1

Contraseña

\*\*\*\*\*

Iniciar Sesión

- 16

TaskAPP

Iniciar SesiónRegistro

→]

Iniciar Sesión

Usuario o Correo Electrónico

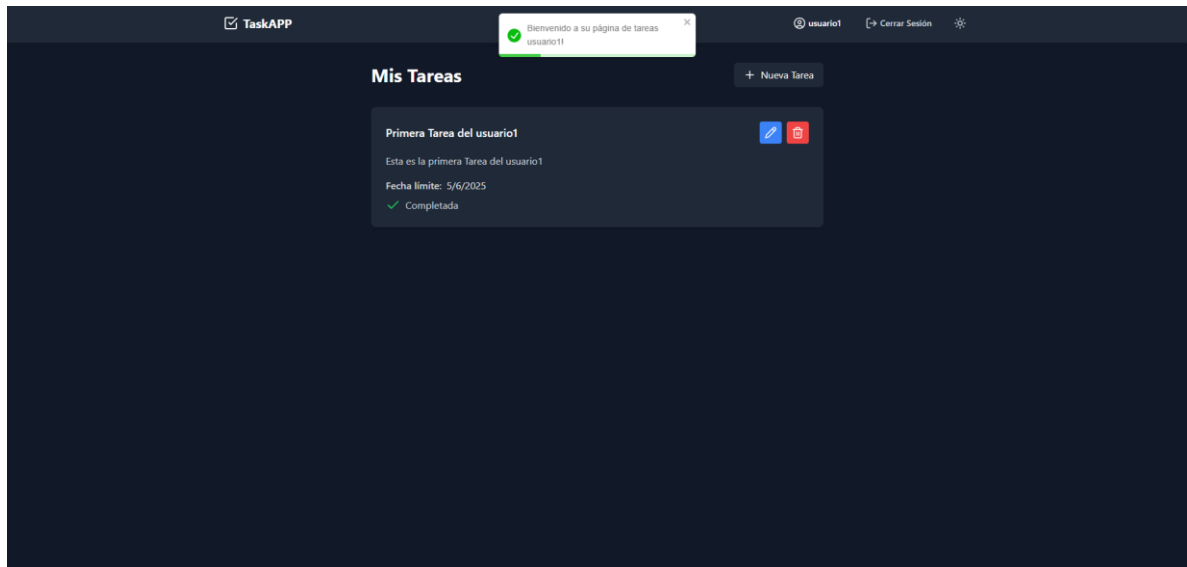
usuario1

Contraseña

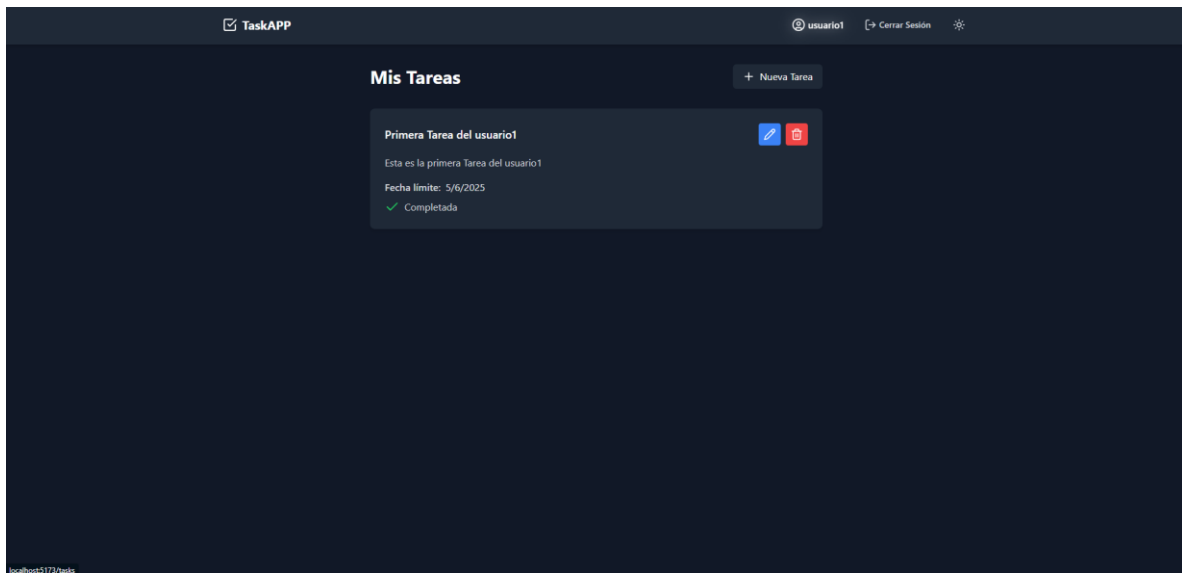
PassUsuario1

Iniciar Sesión

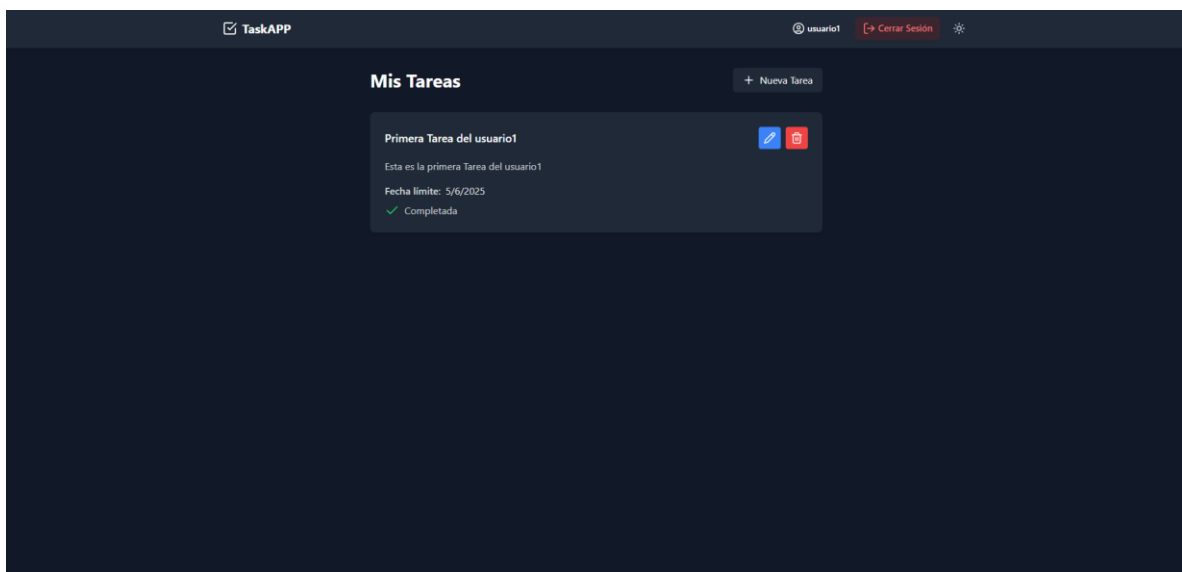
- 17



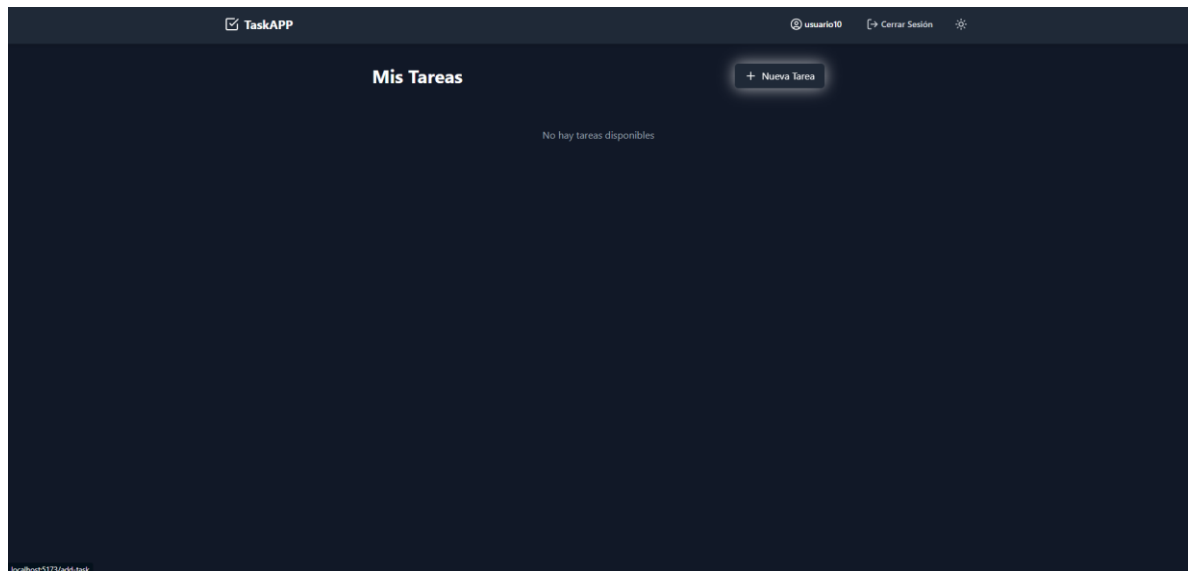
- 18



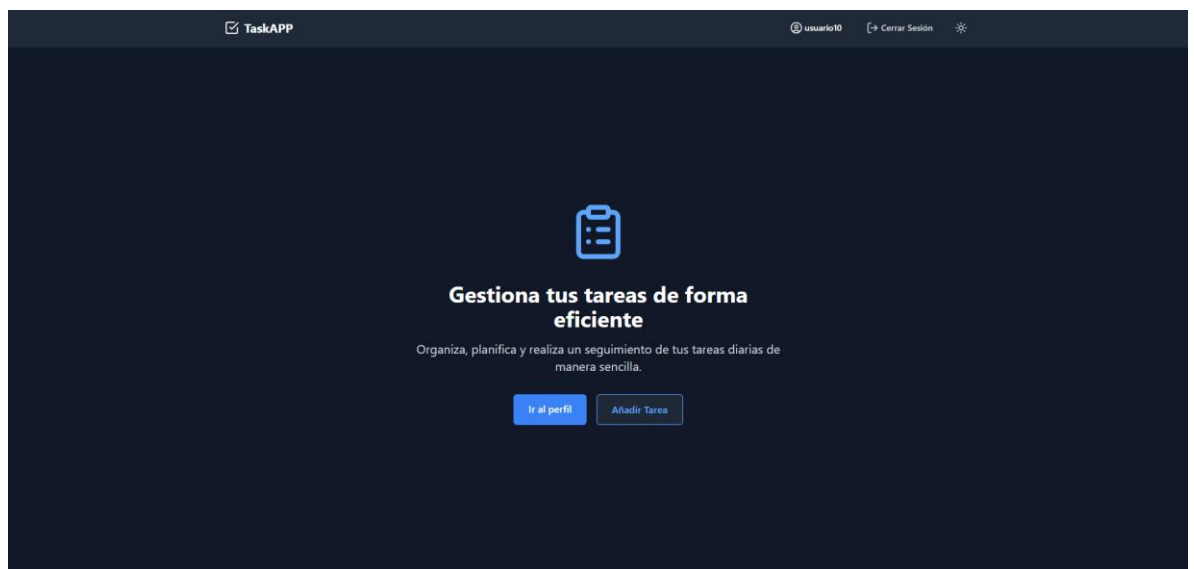
- 19



- 20



- 21

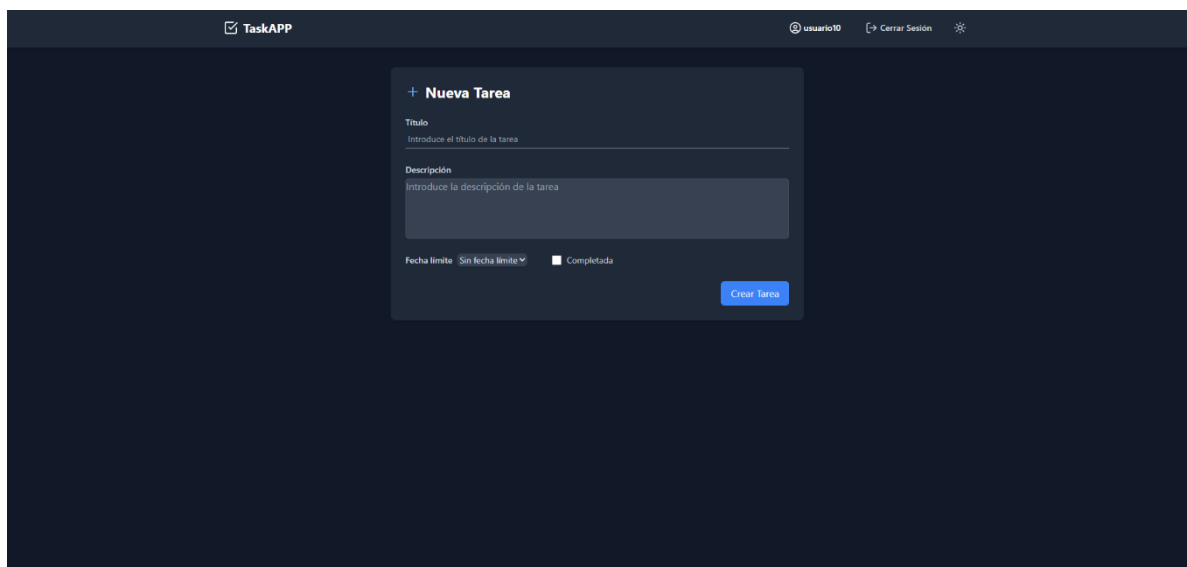


- 22



## 9.3 CRUD de Tareas

- 23



- 24

TaskAPP

usuario10Cerrar Sesión

+ Nueva Tarea

Título

Introduce el título de la tarea

Descripción

Introduce la descripción de la tarea

Fecha límite

Sin fecha límite

Completada

Crear Tarea

- 25

TaskAPP

usuario10Cerrar Sesión

+ Nueva Tarea

Título

Introduce el título de la tarea

Descripción

Introduce la descripción de la tarea

Fecha límite

Sin fecha límite

Completada

Crear Tarea

- 26

TaskAPP

usuario10Cerrar Sesión

+ Nueva Tarea

Título

Introduce el título de la tarea

Descripción

Introduce la descripción de la tarea

Fecha límite

Sin fecha límite

Sin fecha límite

Hoy

Mañana

En 1 semana

En 1 mes

☐ Completada

Crear Tarea

- 27

TaskAPP

usuario10Cerrar Sesión

+ Nueva Tarea

Título

Primera tarea Usuario10

Descripción

Esto es la primera tarea del usuario10

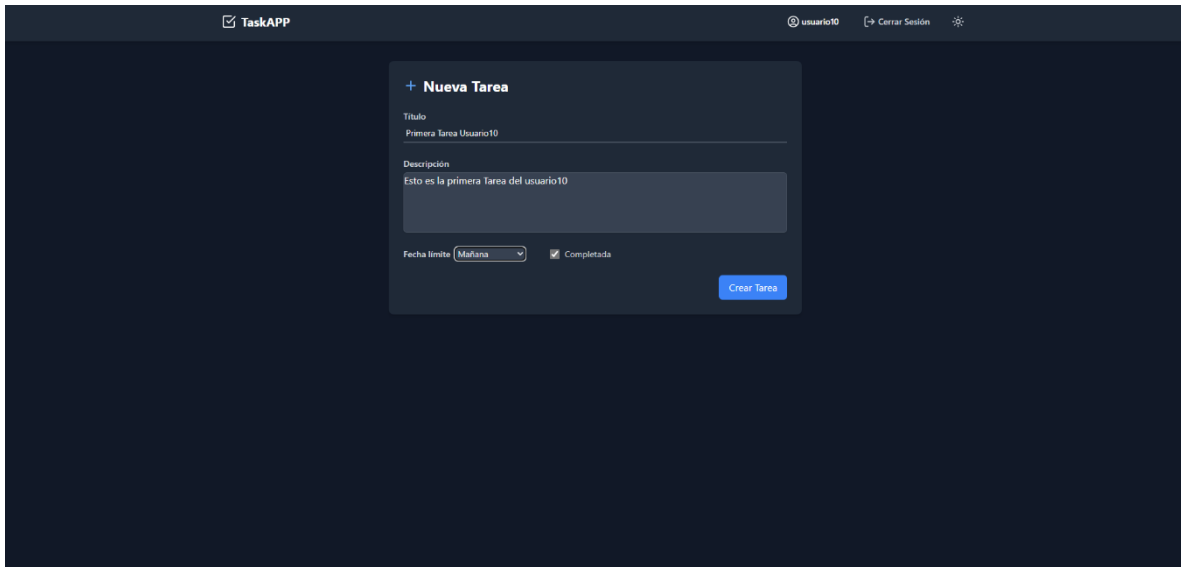
Fecha límite

Mañana

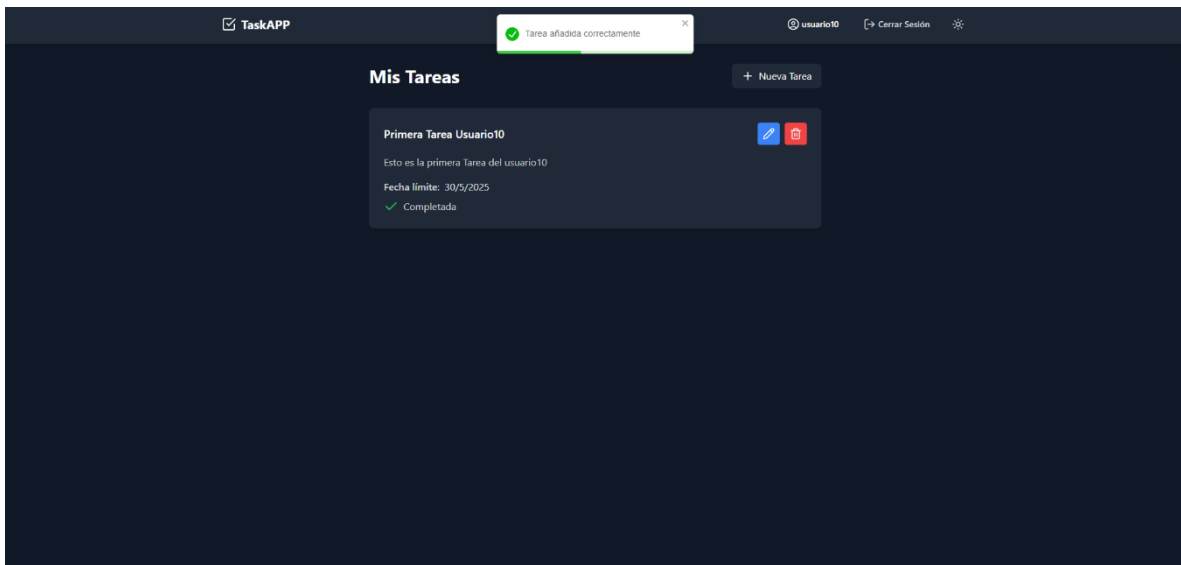
☒ Completada

Crear Tarea

- 28

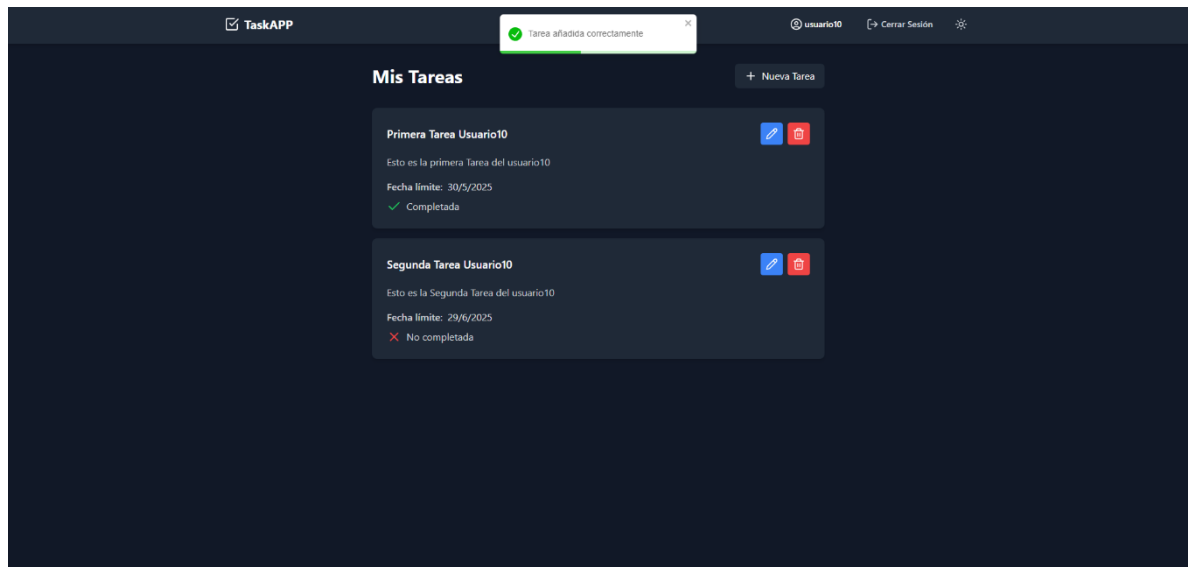


- 29

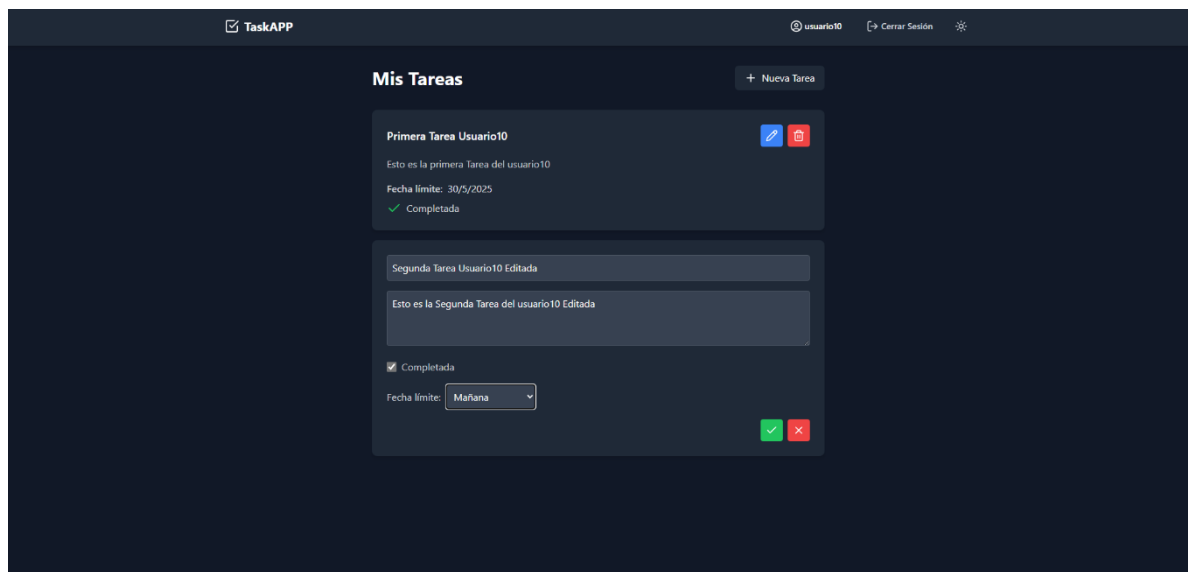




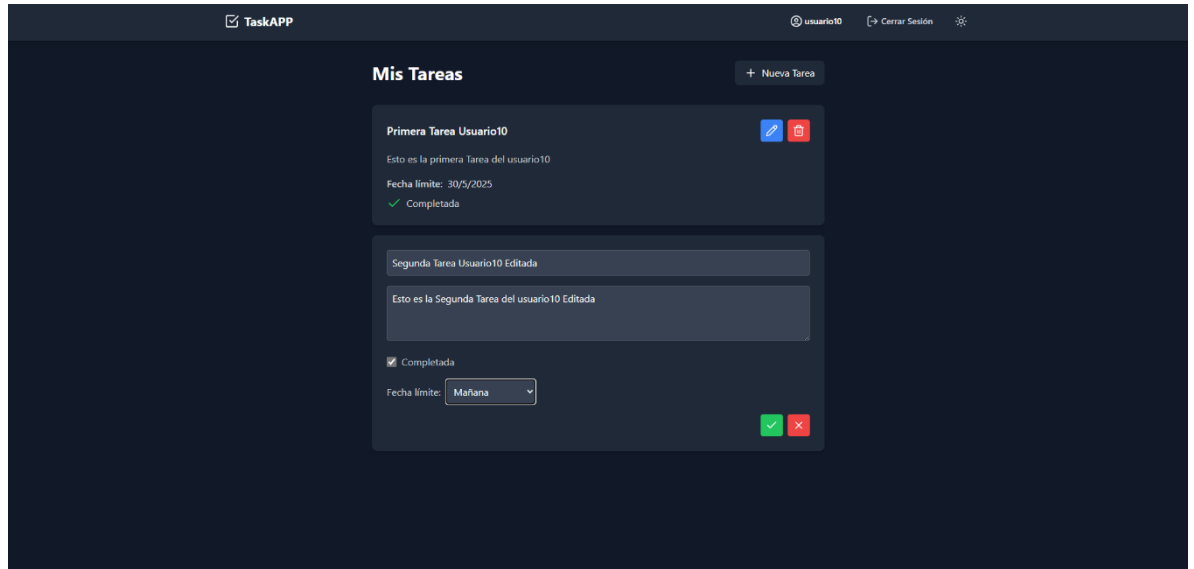
- 30



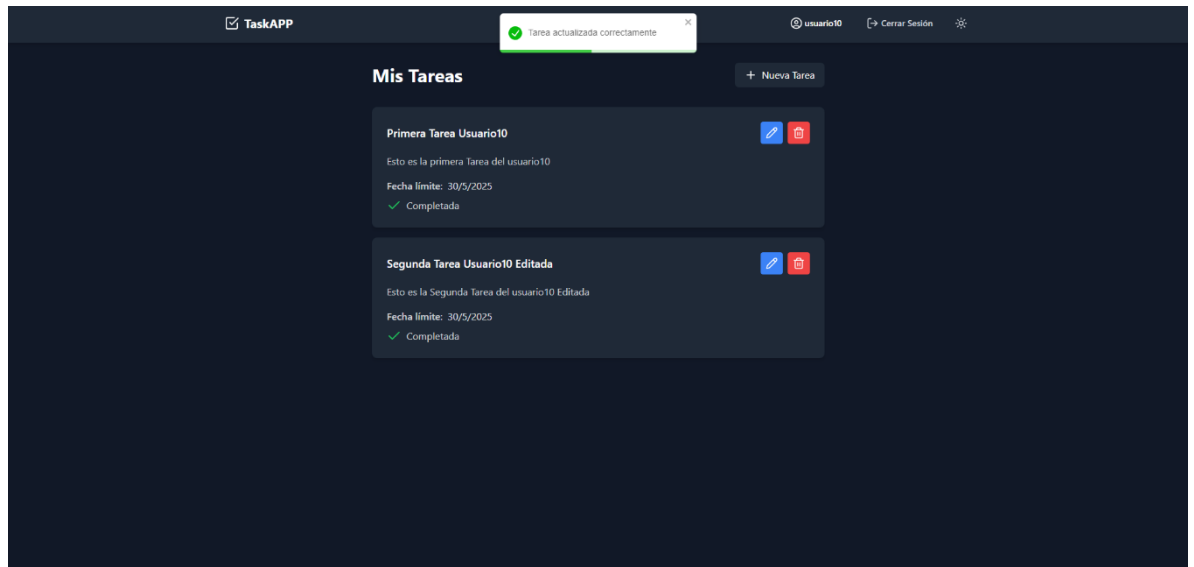
- 31



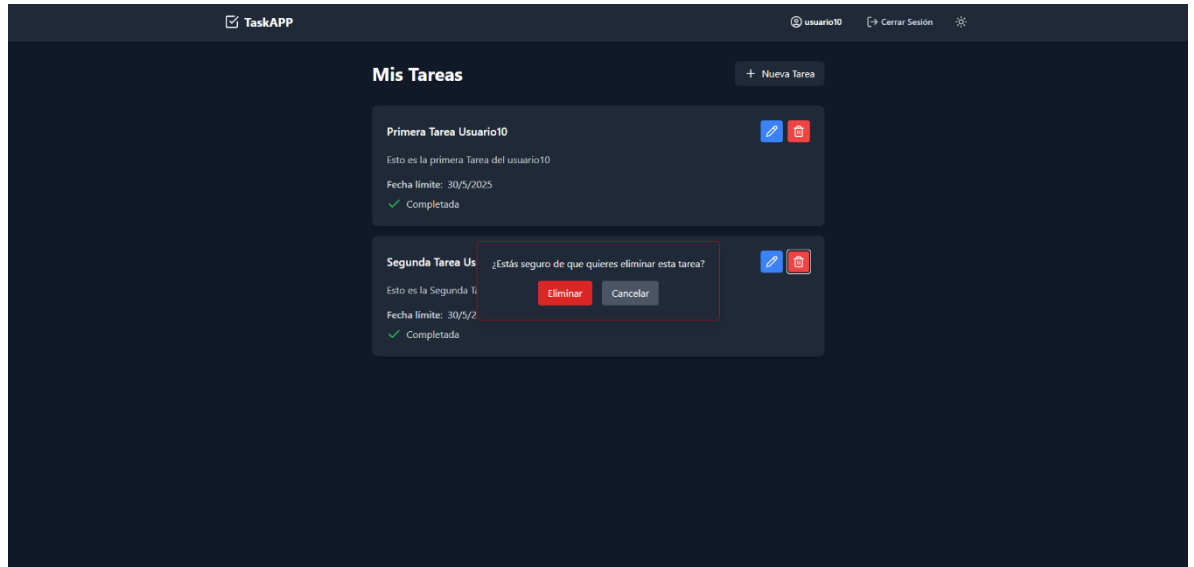
- 32



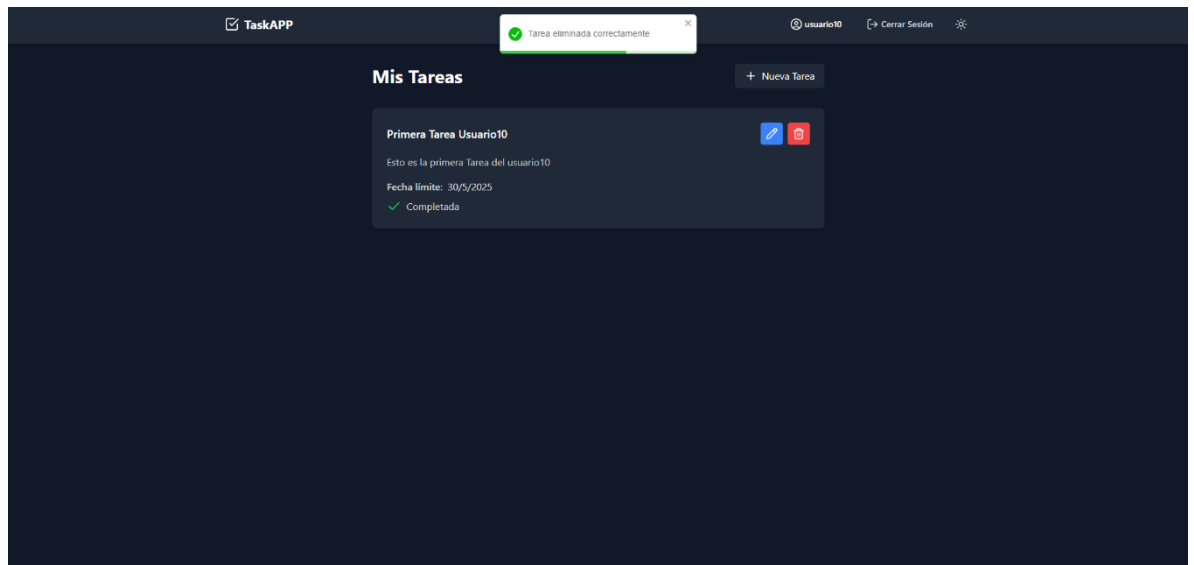
- 33



- 34



- 35



## 10. Notas Finales

- El servicio de hasheado en frontend utiliza la API Web Crypto para SHA-256.
- El backend está preparado para usar bcrypt en producción.
- El sistema es fácilmente extensible para nuevas funcionalidades (roles, adjuntos, etc).
- El código está listo para ser convertido a formato DOC o PDF para entrega o documentación formal.