

**TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES WEB (IFCS03)**

**MÓDULO: PROYECTO**. **CURSO: 2024-2025**

**CONVOCATORIA:** **ORDINARIA**

**Título del proyecto: TASK-CRAFT**

**PROFESOR TUTOR: Mercedes Limón Echevarría**

**Alumnos:**

|  |  |
| --- | --- |
| **APELLIDOS Y NOMBRE: MOLINA ARELLANO NEIL** | **DNI: 54444173E** |

**MODALIDAD DE PROYECTO: Proyectos de gestión**

ÍNDICE

[1. JUSTIFICACIÓN 3](#__RefHeading___Toc3244_2974155747)

[2. INTRODUCCIÓN 4](#__RefHeading___Toc3246_2974155747)

[3. TOMA DE REQUISITOS 5](#__RefHeading___Toc3248_2974155747)

[4. OBJETIVOS 36](#__RefHeading___Toc3250_2974155747)

[5. DESARROLLO 37](#__RefHeading___Toc3252_2974155747)

[5.1. METODOLOGÍA 37](#__RefHeading___Toc3254_2974155747)

[5.2. TECNOLOGÍAS 45](#__RefHeading___Toc3256_2974155747)

[5.3. PROTOTIPADO Y USABILIDAD 46](#__RefHeading___Toc3258_2974155747)

[5.4. MODELADO DE DATOS 64](#__RefHeading___Toc3260_2974155747)

[5.5. SEGURIDAD Y AUTENTICACIÓN 71](#__RefHeading___Toc3262_2974155747)

[5.6. DISEÑO E IMPLEMENTACIÓN 72](#__RefHeading___Toc3264_2974155747)

[5.7. PRUEBAS Y CALIDAD DEL SOFTWARE 74](#__RefHeading___Toc3266_2974155747)

[6. DESPLIEGUE DEL PRODUCTO 78](#__RefHeading___Toc3268_2974155747)

[7. PRESUPUESTO, RIESGOS, VIABILIDAD 80](#__RefHeading___Toc3270_2974155747)

[8. DIFICULTADES ENCONTRADAS Y RESOLUCIÓN 82](#__RefHeading___Toc3272_2974155747)

[9. CONCLUSIONES 83](#__RefHeading___Toc3274_2974155747)

[10. REFERENCIAS BIBLIOGRÁFICAS 84](#__RefHeading___Toc3276_2974155747)

[11. ANEXOS 85](#__RefHeading___Toc3278_2974155747)

# JUSTIFICACIÓN

En un mundo cada vez más ocupado y lleno de distracciones, resulta fundamental contar con una herramienta que facilite la organización de las actividades diarias de manera eficiente y efectiva. La falta de organización puede conllevar fácilmente a la pérdida de tiempo en actividades poco productivas, desviándonos de nuestros objetivos y metas personales. Esta necesidad surge en situaciones donde se busca equilibrar la adquisición de nuevas habilidades, como aprender a programar tanto de forma académica como autodidacta, con otras actividades recreativas o sociales, como asistir al gimnasio o pasar tiempo con amigos. La inspiración proviene de obras como "Hábitos Atómicos" de James Clear, donde se destaca la importancia de desarrollar hábitos pequeños pero consistentes en el día a día para alcanzar metas personales y convertirse en la mejor versión de uno mismo. En este contexto, surge la idea de crear una aplicación que facilite la creación y seguimiento de nuevos hábitos, proporcionando una herramienta de organización más efectiva para el manejo de múltiples actividades. (Clear, 2020)

# INTRODUCCIÓN

En un contexto contemporáneo marcado por la velocidad y la constante presencia de distracciones, surge la necesidad imperante de gestionar eficazmente el tiempo y las responsabilidades. Este proyecto de fin de grado aborda esta premisa mediante la creación de una aplicación de gestión de tareas y metas, con el propósito de mejorar la productividad y la satisfacción del usuario al cumplir con sus compromisos diarios y alcanzar objetivos a largo plazo.

El objetivo de esta investigación consiste en desarrollar una herramienta intuitiva y eficiente que facilite a los usuarios la organización de sus actividades cotidianas y la planificación de sus metas de manera efectiva. La motivación principal detrás de esta indagación radica en la creciente demanda de soluciones que permitan a las personas manejar sus responsabilidades de forma más efectiva, aumentando así su eficiencia personal.

Para lograr dicho propósito, se realizó una revisión exhaustiva de estudios sobre gestión del tiempo y productividad, con el fin de identificar los principales obstáculos y desafíos que los usuarios enfrentan en su organización diaria. El desarrollo de la aplicación se llevó a cabo empleando un enfoque centrado en el usuario, lo cual incluyó el diseño de una interfaz intuitiva y la implementación de mecanismos de seguimiento y organización de tareas.

Los precedentes y prácticas en el ámbito de la gestión del tiempo demuestran que una planificación adecuada y el uso de herramientas especializadas pueden incidir significativamente en la eficiencia personal. Con los avances tecnológicos y el diseño de aplicaciones, surgen nuevas oportunidades para desarrollar herramientas más intuitivas y eficaces.

Este proyecto proporciona una solución práctica para la gestión de tareas y metas, con el propósito de ayudar a las personas a optimizar su tiempo y alcanzar sus objetivos de manera efectiva. El proceso de desarrollo de la aplicación garantiza que esta herramienta responda a las necesidades actuales de los usuarios.

# TOMA DE REQUISITOS

1. Que el usuario pueda **iniciar sesión**: Mediante ingresando unas credenciales (correo electrónico y contraseña), el usuario pueda acceder de manera segura a las funcionalidades de la aplicación. Al autenticarse correctamente, el sistema generará una sesión segura o token con JWT.

2. El programa tendrá la opción de ofrecer al usuario un formulario de **registro,** el cual el cliente podrá crear una credenciales personalizadas, con la **contraseña encriptada** para ofrecer mayor seguridad a la aplicación.

3. **Gestión de diarios:** se podrá **crear, editar, eliminar o visualizar** entradas de **diarios**, asociadas a un usuario. Cada uno de estos tendrá la fecha, el título y un contenido.

4. **Gestión de tareas:** se podrá **crear, editar, eliminar o visualizar** (tanto en un calendario como en una página aparte) **tareas,** asociadas a un usuario. Cada uno de estos contendrá el título, una breve descripción, un estado cargado desde la base de datos y una categoría, el cual podrá ser personalizada por el usuario o cargada desde la base de datos.

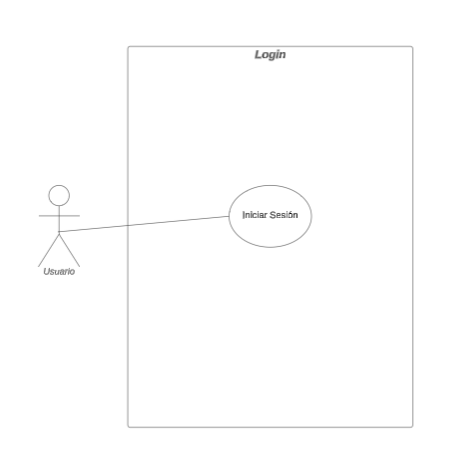
5. **Gestión de usuario:** el usuario podrá **cambiar sus credenciales, cerrar sesión o eliminar** su cuenta.

6. **La aplicación debe ser web:** debe de tener una interfaz amigable, accesible y responsiva que se pueda manejar desde un navegador.

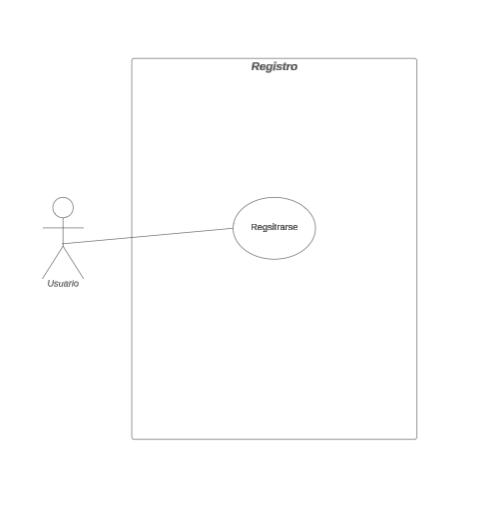
7. **Seguridad:** la aplicación tendrá las **contraseñas encriptadas** y las sesiones estarán protegidas con **JWT**, guardadas en una **cookie**.

8. **Gestión de amigos:** el usuario podrá añadir a amigos mediante solicitudes de amistad. También se podrán compartir tareas entre ellos.

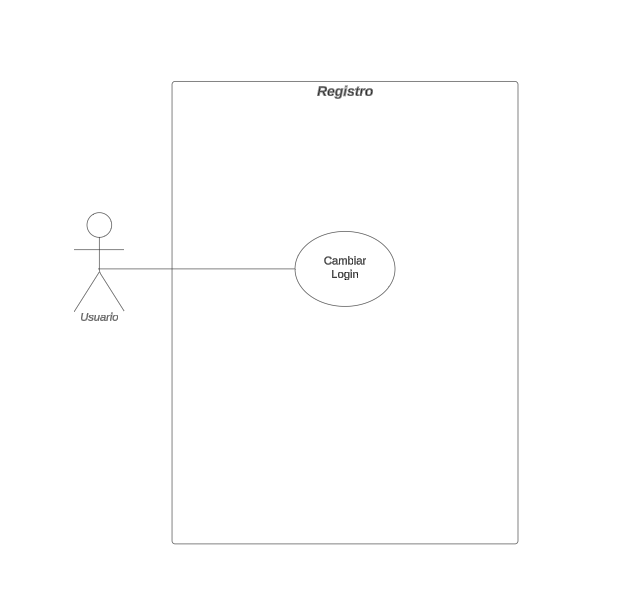
**CASOS DE USO**



|  |  |
| --- | --- |
| **Nombre** | CU01 |
| **Actor** | Usuario |
| **Descripción** | El usuario inicia sesión |
| **Evento desencadenado** | La pantalla se cambia a la de la aplicación principal si las credenciales son correctas. |
| **Pasos realizados** | 1. Pulsar el botón de cambio de pantalla 2. Se cambia de pantalla de Registro |
| **Precondiciones** | Abrir la pantalla de Login |
| **Postcondiciones** | Cambiar de pantalla de Dashboard |
| **Vista** | Login |



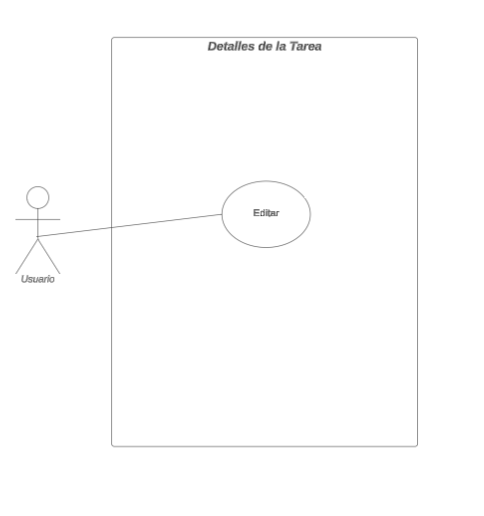
|  |  |
| --- | --- |
| **Nombre** | CU02 |
| **Actor** | Usuario |
| **Descripción** | El usuario se registra |
| **Evento desencadenado** | La pantalla se cambia a la de Login |
| **Pasos realizados** | 1. Pulsar el botón de cambio de pantalla 2. Se cambia de pantalla de Registro |
| **Precondiciones** | Abrir la pantalla de Registro |
| **Postcondiciones** | Cambiar de pantalla de Dashboard |
| **Vista** | Register |



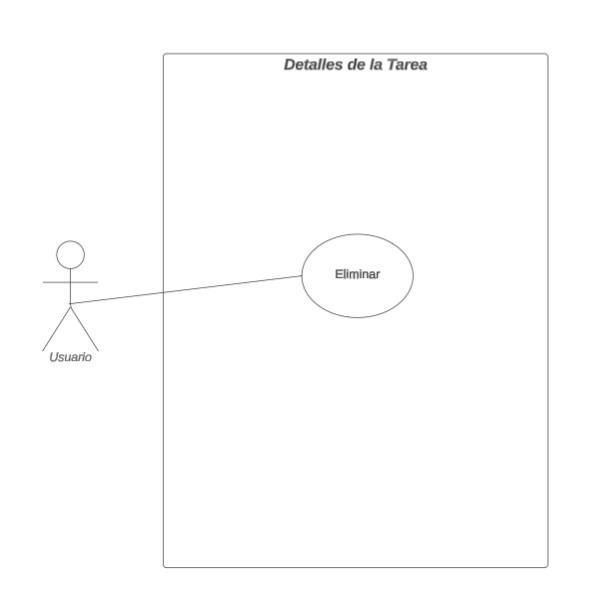
|  |  |
| --- | --- |
| **Nombre** | CU03 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia de pantalla a la de Register |
| **Evento desencadenado** | La pantalla se cambia a la de Register |
| **Pasos realizados** | 1. Pulsar el botón de cambio de pantalla 2. Se cambia de pantalla de Registro |
| **Precondiciones** | Abrir la pantalla de Registro |
| **Postcondiciones** | Cambiar de pantalla de Dashboard |
| **Vista** | Register |



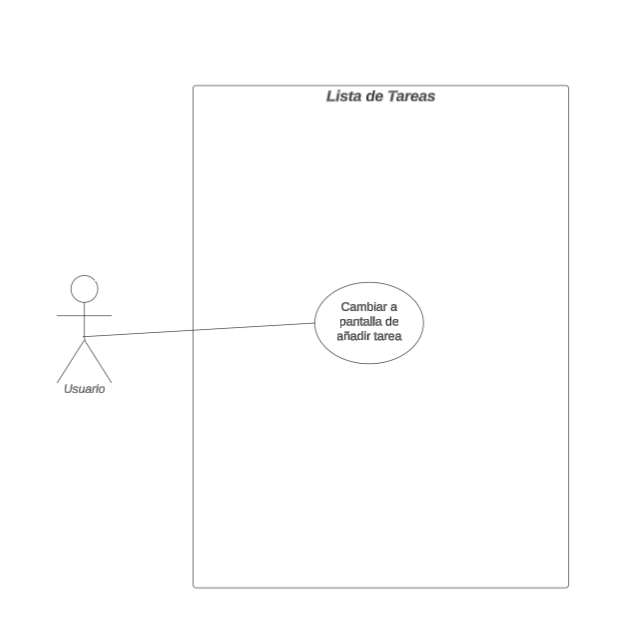
|  |  |
| --- | --- |
| **Nombre** | CU04 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia de pantalla a la de Detalles Tarea |
| **Evento desencadenado** | La pantalla se cambia a la de Detalles Tareas |
| **Pasos realizados** | 1. Pulsar la tarea 2. Se cambia de pantalla de Detalles Tareas |
| **Precondiciones** | Abrir la pantalla de Lista de Tareas |
| **Postcondiciones** | Cambiar de pantalla de Detalles Tareas |
| **Vista** | Lista de Tareas |



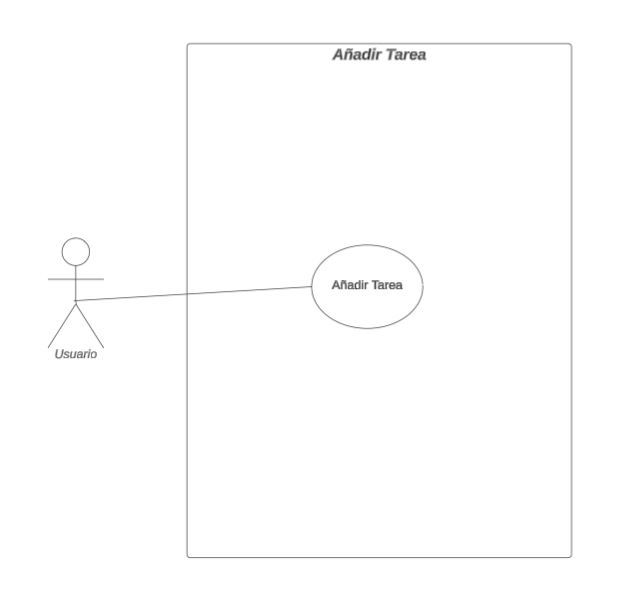
|  |  |
| --- | --- |
| **Nombre** | CU05 |
| **Actor** | Usuario |
| **Descripción** | El usuario edita la tarea en la pantalla de Detalles Tarea |
| **Evento desencadenado** | La pantalla se cambia a la de Lista de Tareas, se cambia los datos de la tarea tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar la tarea 2. Se cambian los datos de la tarea 3. Se cambia de pantalla a la Lista de Tareas |
| **Precondiciones** | Abrir la Tarea |
| **Postcondiciones** | Cambiar de pantalla de Lista de Tareas |
| **Vista** | Detalles Tarea |



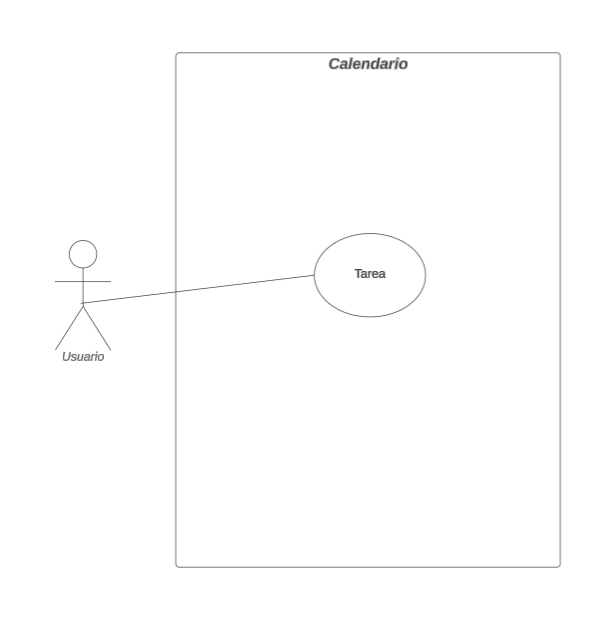
|  |  |
| --- | --- |
| **Nombre** | CU06 |
| **Actor** | Usuario |
| **Descripción** | El usuario se elimina la tarea en la pantalla de Detalles Tarea |
| **Evento desencadenado** | La pantalla se cambia a la de Lista Tareas y se elimina la tarea tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar la tarea 2. Se elimina la tarea 3. Se cambia de pantalla de Detalles Tareas |
| **Precondiciones** | Abrir la Tarea |
| **Postcondiciones** | Cambiar de pantalla de Lista de Tareas |
| **Vista** | Detalles Tarea |



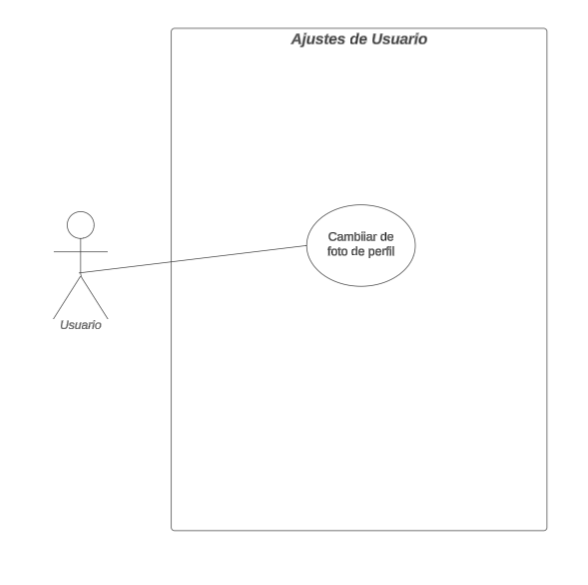
|  |  |
| --- | --- |
| **Nombre** | CU07 |
| **Actor** | Usuario |
| **Descripción** | El usuario pulsa el botón para cambiar a la pantalla de Añadir Tareas |
| **Evento desencadenado** | La pantalla se cambia a la de Añadir Tareas |
| **Pasos realizados** | 1. Pulsar el botón 2. Se cambia de pantalla de Añadir Tareas |
| **Precondiciones** | Estar en la pantalla de Lista de Tareas |
| **Postcondiciones** | Cambiar de pantalla de Añadir Tareas |
| **Vista** | Lista de Tareas |



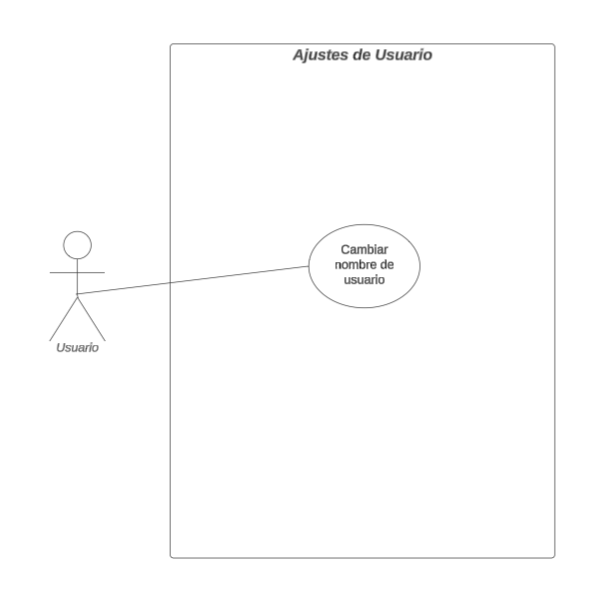
|  |  |
| --- | --- |
| **Nombre** | CU08 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia la pantalla a la de Detalles Tarea |
| **Evento desencadenado** | La pantalla se cambia a la de Añadir Tareas |
| **Pasos realizados** | 1. Pulsar el botón 2. Se cambia de pantalla de Añadir Tareas |
| **Precondiciones** | Estar en la pantalla de Lista de Tareas |
| **Postcondiciones** | Cambiar de pantalla de Añadir Tareas |
| **Vista** | Añadir de Tareas |



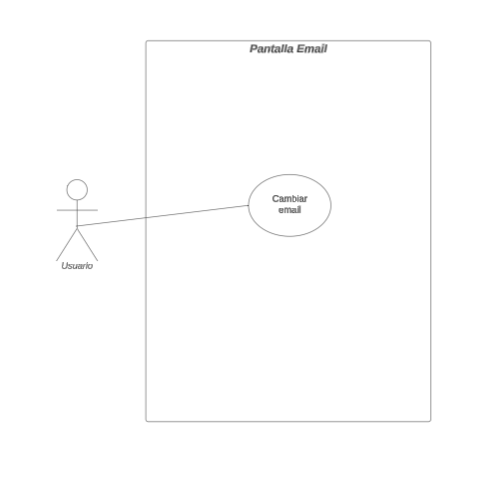
|  |  |
| --- | --- |
| **Nombre** | CU09 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia de pantalla a la de Detalles Tarea |
| **Evento desencadenado** | La pantalla se cambia a la de Detalles Tareas |
| **Pasos realizados** | 1. Pulsar la tarea o el evento en el calendario 2. Se cambia de pantalla de Detalles Tareas |
| **Precondiciones** | Abrir la pantalla del Calendario |
| **Postcondiciones** | Cambiar de pantalla de Detalles Tareas |
| **Vista** | Calendario |



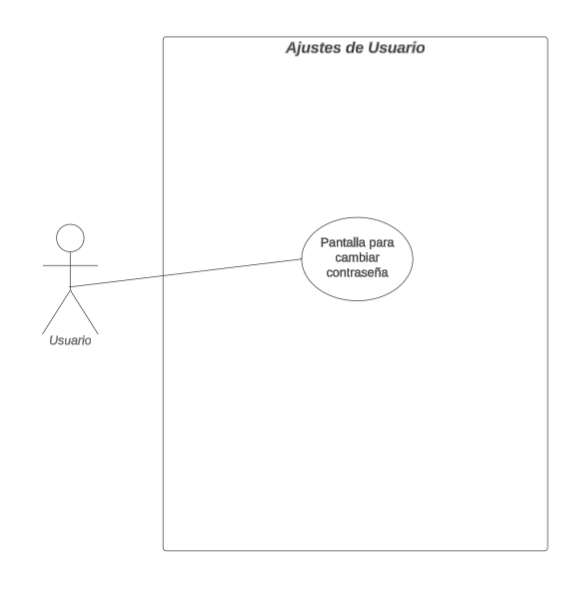
|  |  |
| --- | --- |
| **Nombre** | CU10 |
| **Actor** | Usuario |
| **Descripción** | El usuario cambia su foto de perfil |
| **Evento desencadenado** | La foto de perfil del usuario se cambia |
| Pasos realizados | 1. Pulsar botón 2. Se cambian los datos del usuario tanto en la UI como en la base de datos |
| **Precondiciones** | Estar en la pantalla de Ajustes de Usuario |
| **Postcondiciones** | Se abre un menu y el usuario cambia su foto de perfil |
| **Vista** | Ajustes de Usuario |



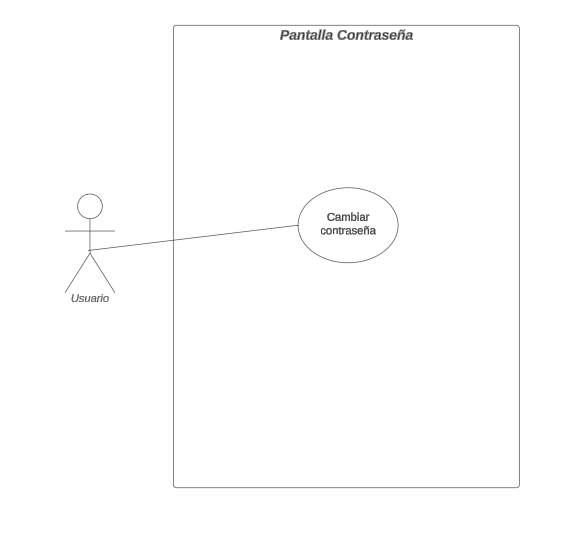
|  |  |
| --- | --- |
| **Nombre** | CU10 |
| **Actor** | Usuario |
| **Descripción** | El usuario cambia su nombre de usuario |
| **Evento desencadenado** | El nombre de usuario se cambia |
| **Pasos realizados** | 1. Pulsar botón 2. Se cambian los datos del usuario tanto en la UI como en la base de datos |
| **Precondiciones** | Estar en la pantalla de Ajustes de Usuario |
| **Postcondiciones** | Se abre un menu y el usuario cambia su nombre |
| **Vista** | Ajustes de Usuario |



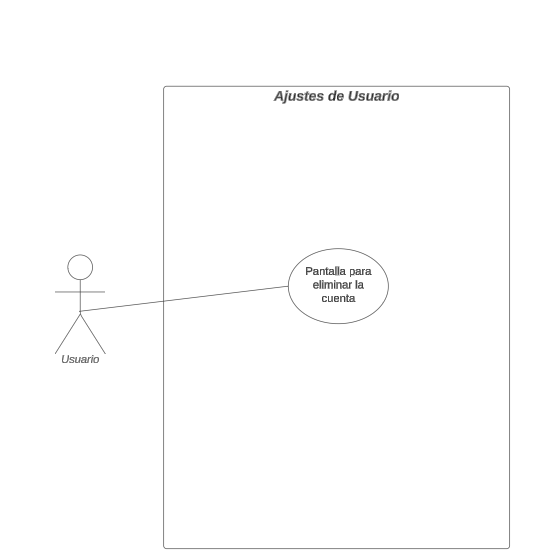
|  |  |
| --- | --- |
| **Nombre** | CU11 |
| **Actor** | Usuario |
| **Descripción** | El usuario cambia su email |
| **Evento desencadenado** | El email del usuario se cambia |
| **Pasos realizados** | 1. Pulsar botón 2. Se cambian los datos del usuario tanto en la UI como en la base de datos |
| **Precondiciones** | Estar en la pantalla de Ajustes de Usuario |
| **Postcondiciones** | Se abre un menu y el usuario cambia su nombre |
| **Vista** | Ajustes de Usuario |



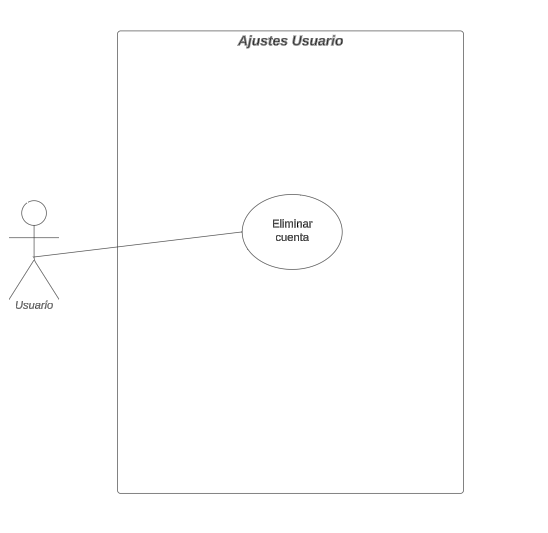
|  |  |
| --- | --- |
| **Nombre** | CU12 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia de pantalla a la de Cambiar contraseña |
| **Evento desencadenado** | La pantalla se cambia a la de Cambiar contraseña |
| **Pasos realizados** | 1. Pulsar el botón 2. Se cambia de pantalla de Cambiar contraseña |
| **Precondiciones** | Abrir la pantalla de Ajustes de Usuario |
| **Postcondiciones** | Cambiar de pantalla de Cambiar Contraseña |
| **Vista** | Ajustes de Usuario |



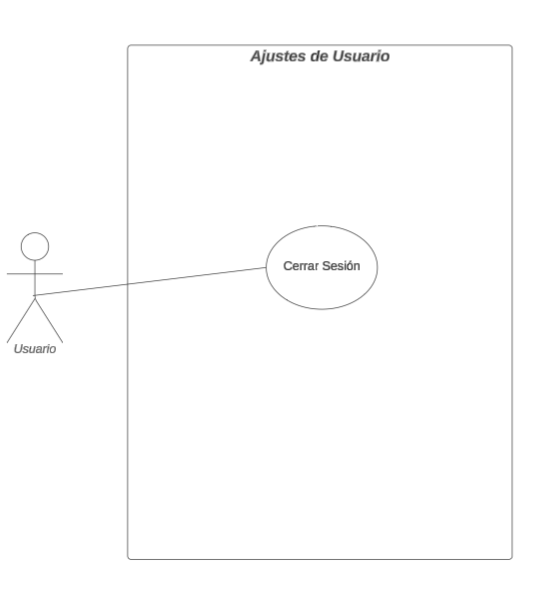
|  |  |
| --- | --- |
| **Nombre** | CU13 |
| **Actor** | Usuario |
| **Descripción** | El usuario cambia su contraseña |
| **Evento desencadenado** | La contraseña del usuario se cambia |
| **Pasos realizados** | 1. Pulsar botón 2. Se cambian los datos del usuario tanto en la UI como en la base de datos |
| **Precondiciones** | Estar en la pantalla de Cambiar Contraseña |
| **Postcondiciones** | Se abre un menu y el usuario cambia su nombre |
| **Vista** | Cambiar Contraseña |



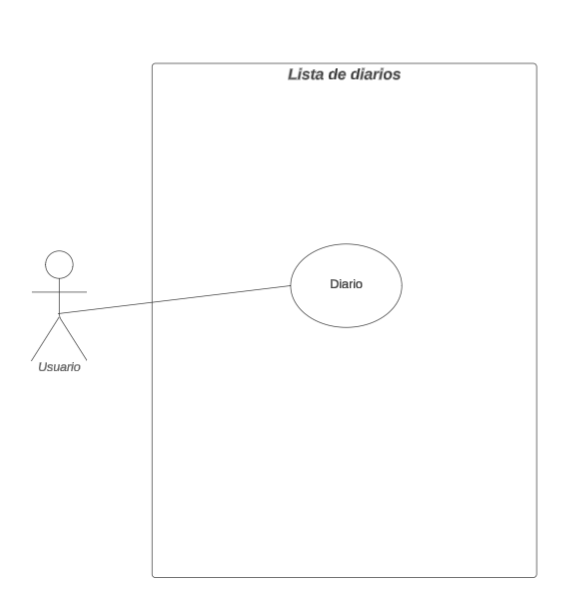
|  |  |
| --- | --- |
| **Nombre** | CU14 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia de pantalla a la de Eliminar contraseña |
| **Evento desencadenado** | La pantalla se cambia a la de Eliminar contraseña |
| **Pasos realizados** | 1. Pulsar el botón 2. Se cambia de pantalla de Eliminar contraseña |
| **Precondiciones** | Abrir la pantalla de Ajustes de Usuario |
| **Postcondiciones** | Cambiar de pantalla de Cambiar Contraseña |
| **Vista** | Ajustes de Usuario |



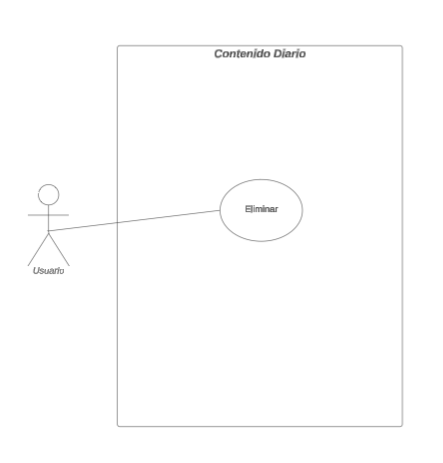
|  |  |
| --- | --- |
| **Nombre** | CU15 |
| **Actor** | Usuario |
| **Descripción** | El usuario elimina la cuenta |
| **Evento desencadenado** | Se despliega el menu para eliminar la cuenta |
| **Pasos realizados** | 1. Pulsar el botón 2. Se eliminan los datos del usuario como en la base de datos y se cierra sesión |
| **Precondiciones** | Pulsar el botón |
| **Postcondiciones** | Se elimina la cuenta |
| **Vista** | Ajustes de Usuario |



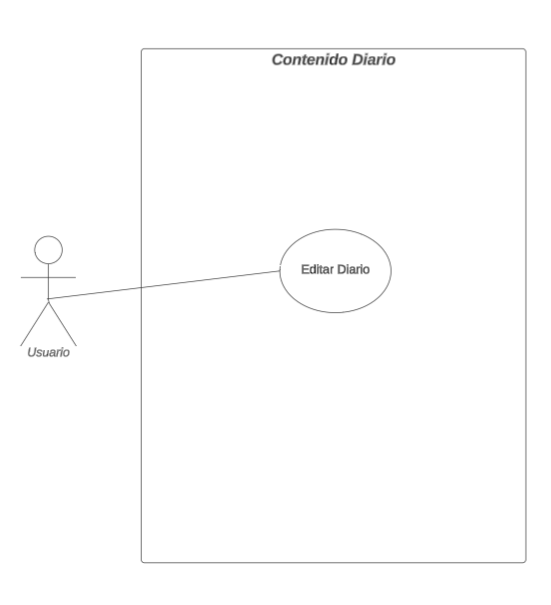
|  |  |
| --- | --- |
| **Nombre** | CU16 |
| **Actor** | Usuario |
| **Descripción** | El usuario cierra sesión |
| **Evento desencadenado** | Se despliega el menu para eliminar la cuenta |
| **Pasos realizados** | 1. Pulsar el botón 2. Se elimina el token y se cierra sesión |
| **Precondiciones** | Pulsar el botón |
| **Postcondiciones** | Se cierra sesión de la cuenta |
| **Vista** | Ajustes de Usuario |



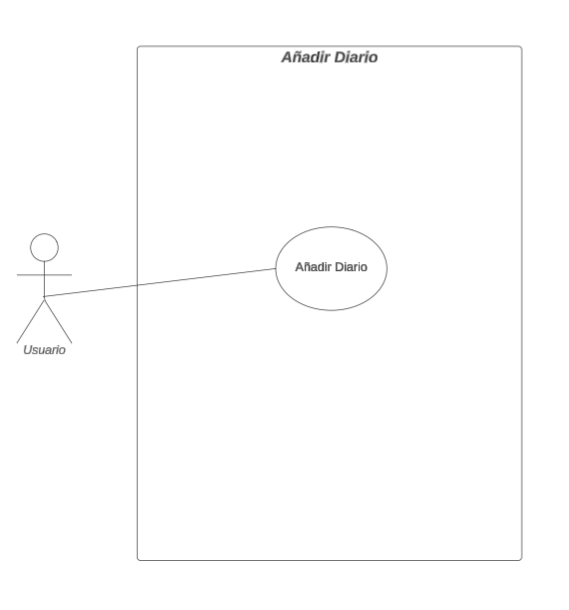
|  |  |
| --- | --- |
| **Nombre** | CU17 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia de pantalla a la de Detalles Diarios |
| **Evento desencadenado** | La pantalla se cambia a la de Detalles Diarios |
| **Pasos realizados** | 1. Pulsar la tarea 2. Se cambia de pantalla de Detalles Tareas |
| **Precondiciones** | Abrir la pantalla de Lista de Diarios |
| **Postcondiciones** | Cambiar de pantalla de Detalles Diarios |
| **Vista** | Lista de Diarios |



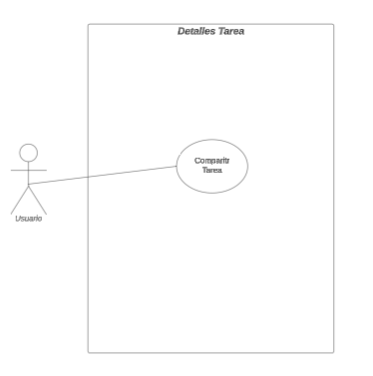
|  |  |
| --- | --- |
| **Nombre** | CU18 |
| **Actor** | Usuario |
| **Descripción** | El usuario elimina el diario en la pantalla de Detalles Diarios |
| **Evento desencadenado** | La pantalla se cambia a la de Lista Diarios y se elimina la tarea tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el diario 2. Se elimina el diario 3. Se cambia de pantalla de Lista Diarios |
| **Precondiciones** | Abrir el Diario |
| **Postcondiciones** | Cambiar de pantalla de Lista de Diarios |
| **Vista** | Detalles Diarios |



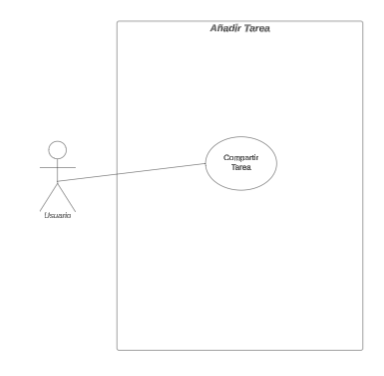
|  |  |
| --- | --- |
| **Nombre** | CU19 |
| **Actor** | Usuario |
| **Descripción** | El usuario edita el diario en la pantalla de Detalles Diarios |
| **Evento desencadenado** | La pantalla se cambia a la de Lista Diarios y se edita la tarea tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el diario 2. Se edita el diario 3. Se cambia de pantalla de Lista Diarios |
| **Precondiciones** | Abrir el Diario |
| **Postcondiciones** | Cambiar de pantalla de Lista de Diarios |
| **Vista** | Detalles Diarios |



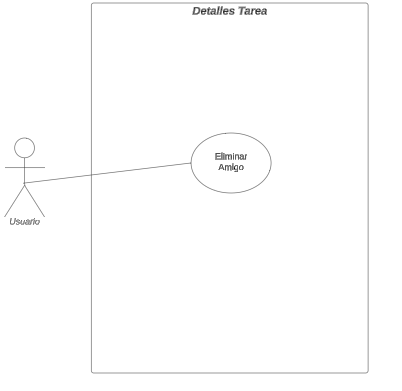
|  |  |
| --- | --- |
| **Nombre** | CU20 |
| **Actor** | Usuario |
| **Descripción** | El usuario se cambia la pantalla a la de Añadir Diarios |
| **Evento desencadenado** | La pantalla se cambia a la de Añadir Diarios |
| **Pasos realizados** | 1. Pulsar el botón 2. Se cambia de pantalla de Añadir Diarios |
| **Precondiciones** | Estar en la pantalla de Lista de Diarios |
| **Postcondiciones** | Cambiar de pantalla de Añadir Diarios |
| **Vista** | Añadir de Diarios |



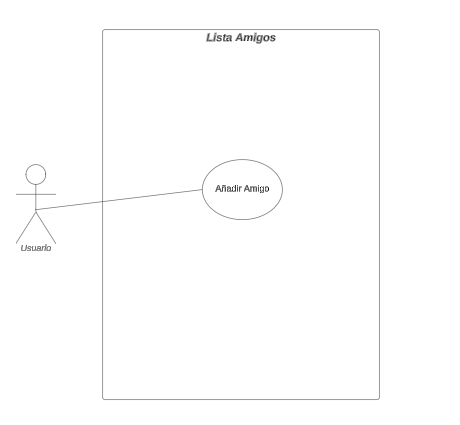
|  |  |
| --- | --- |
| **Nombre** | CU21 |
| **Actor** | Usuario |
| **Descripción** | El usuario comparte la tarea en la pantalla de Detalles Tareas |
| **Evento desencadenado** | La pantalla se cambia a la de Lista Tareas y se comparte la tarea. Esto se mostrará tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se comparte la tarea 3. Se cambia de pantalla de Lista Tareas |
| **Precondiciones** | Abrir la tarea |
| **Postcondiciones** | Cambiar de pantalla de Lista de Tareas |
| **Vista** | Detalles Tareas |



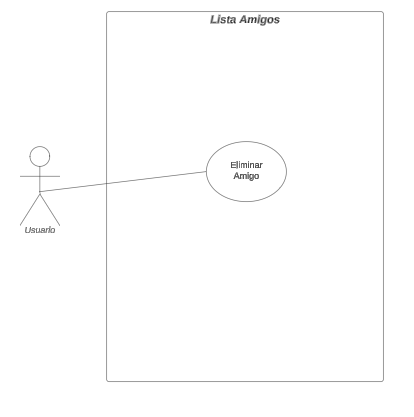
|  |  |
| --- | --- |
| **Nombre** | CU22 |
| **Actor** | Usuario |
| **Descripción** | El usuario comparte la tarea en la pantalla de Añadir Tareas |
| **Evento desencadenado** | La pantalla se cambia a la de Lista Tareas y se comparte la tarea. Esto se mostrará tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se comparte la tarea 3. Se cambia de pantalla de Lista Tareas |
| **Precondiciones** | Abrir la tarea |
| **Postcondiciones** | Cambiar de pantalla de Lista de Tareas |
| **Vista** | Detalles Tareas |



|  |  |
| --- | --- |
| **Nombre** | CU23 |
| **Actor** | Usuario |
| **Descripción** | El usuario elimina al amigo en la tarea en la pantalla de Detalles Tareas |
| **Evento desencadenado** | La pantalla se cambia a la de Lista Tareas y se elimina al amigo en la tarea. Esto se mostrará tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se elimina la tarea 3. Se cambia de pantalla de Lista Tareas |
| **Precondiciones** | Abrir la tarea |
| **Postcondiciones** | Cambiar de pantalla de Lista de Tareas |
| **Vista** | Detalles Tareas |



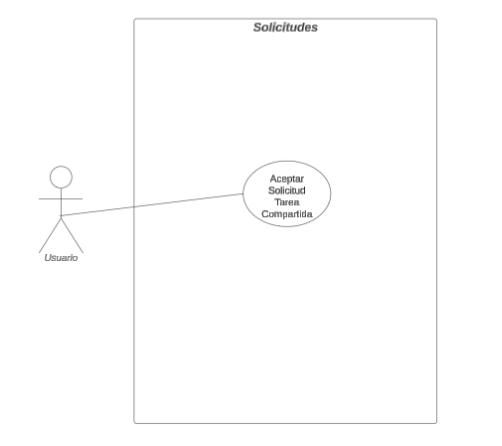
|  |  |
| --- | --- |
| **Nombre** | CU24 |
| **Actor** | Usuario |
| **Descripción** | El usuario añade una solicitud en la pantalla de Lista Amigos |
| **Evento desencadenado** | Se mandará una solicitud de amistad al usuario que se solicitará |
| **Pasos realizados** | 1. Pulsar el botón 2. Se añade la solicitud |
| **Precondiciones** | Estar en la pantalla de Lista Amigos |
| **Postcondiciones** | Se enviará una solicitud a un usuario |
| **Vista** | Lista Amigos |



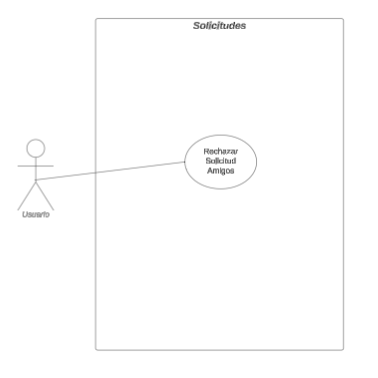
|  |  |
| --- | --- |
| **Nombre** | CU25 |
| **Actor** | Usuario |
| **Descripción** | El usuario elimina un amigo en la pantalla de Lista Amigos |
| **Evento desencadenado** | Se eliminará una amistad al usuario tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se eliminará la amistad |
| **Precondiciones** | Estar en la pantalla de Lista Amigos |
| **Postcondiciones** | Se enviará una solicitud a un usuario |
| **Vista** | Lista Amigos |



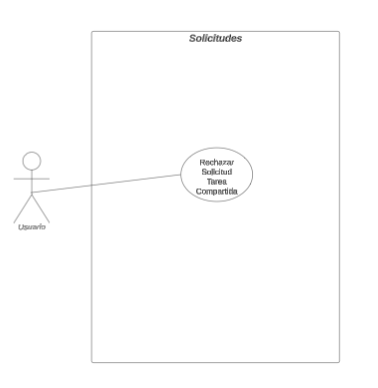
|  |  |
| --- | --- |
| **Nombre** | CU26 |
| **Actor** | Usuario |
| **Descripción** | El usuario aceptar a un amigo en la pantalla de Solicitudes |
| **Evento desencadenado** | Se añadirá una amistad al usuario tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se añadirá la amistad |
| **Precondiciones** | Estar en la pantalla de Solicitudes |
| **Postcondiciones** | Se aceptará una solicitud de un usuario |
| **Vista** | Solicitudes |



|  |  |
| --- | --- |
| **Nombre** | CU27 |
| **Actor** | Usuario |
| **Descripción** | El usuario acepta la una solicitud de compartir tarea en la pantalla de Solicitudes |
| **Evento desencadenado** | Se aceptará una solicitud de compartir tarea al usuario tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se añadirá la tarea |
| **Precondiciones** | Estar en la pantalla de Solicitudes |
| **Postcondiciones** | Se aceptará una solicitud de compartir tarea |
| **Vista** | Solicitudes |



|  |  |
| --- | --- |
| **Nombre** | CU28 |
| **Actor** | Usuario |
| **Descripción** | El usuario elimina a una solicitud en la pantalla de Solicitudes |
| **Evento desencadenado** | Se eliminará una solicitud de un usuario tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se añadirá la amistad |
| **Precondiciones** | Estar en la pantalla de Solicitudes |
| **Postcondiciones** | Se eliminará una solicitud de un usuario |
| **Vista** | Solicitudes |



|  |  |
| --- | --- |
| **Nombre** | CU29 |
| **Actor** | Usuario |
| **Descripción** | El usuario elimina una solicitud de compartir tarea en la pantalla de Solicitudes |
| **Evento desencadenado** | Se aceptará una solicitud de compartir tarea al usuario tanto en la UI como en la base de datos |
| **Pasos realizados** | 1. Pulsar el botón 2. Se eliminará la solicitud |
| **Precondiciones** | Estar en la pantalla de Solicitudes |
| **Postcondiciones** | Se eliminará una solicitud de compartir tarea |
| **Vista** | Solicitudes |

# OBJETIVOS

El objetivo general es crear una aplicación de gestión de tareas y objetivos intuitivos y eficientes para mejorar la productividad y la satisfacción del usuario al cumplir sus compromisos diarios y alcanzar sus objetivos a largo plazo.

Objetivos específicos:

1. Analizar las principales dificultades que enfrentan los usuarios en la gestión de sus tareas diarias y objetivos a largo plazo.
2. Definir los parámetros y funcionalidades necesarias para una aplicación que gestione tareas y objetivos de manera intuitiva y eficiente.
3. Optimizar la interfaz de usuario para asegurar una experiencia fluida y agradable durante la gestión de tareas y objetivos.

# DESARROLLO

## 5.1. METODOLOGÍA

La metodología empleada fue la del método cascada, en la que se estableció una serie de tareas antes de iniciar el proceso de desarrollo de la aplicación. Durante este proceso, era necesario completar cada tarea antes de avanzar a la siguiente. También se estableció una temporalización, antes de realizar el proyecto la cual es la siguiente:

**TEMPORALIZACIÓN SEMANAL DEL PROYECTO**

**Fecha: 14 de marzo – 5 de junio**

**Semana 14 - 20 marzo**

* Inicio del backend.
* Desarrollo del modelo, rutas, validaciones y lógica inicial de TAREAS.
* Conexión con la base de datos y configuración de entorno.

**Semana 21 - 27 marzo**

* Continuación de TAREAS: edición, eliminación, validaciones más complejas.
* Implementación de filtros por fecha, estado y búsqueda por nombre.

**Semana 28 marzo - 3 abril**

* Desarrollo de USUARIOS: creación y autenticación con JWT.
* Creación de DIARIOS: modelo, controladores, y rutas principales.

**Semana 4 - 10 abril**

* Continuación de USUARIOS: validaciones, edición de perfil (no se va a hacer), y gestión de datos.
* Continuación de DIARIOS: edición y eliminación, validaciones y relaciones con usuario.

**Semana 11 - 17 abril**

* Módulo de AMIGOS: modelo, controladores y rutas para solicitudes.
* Funcionalidad de enviar, aceptar, rechazar y eliminar solicitudes de amistad.

**Semana 25 abril**

* Desarrollo de TAREAS COMPARTIDAS: asignación de tareas entre usuarios, validaciones y control de acceso.
* Conexión entre módulo de amigos y tareas compartidas.

**Semana 26 abril - 2 mayo**

* Pantallas de inicio de sesión y registro con validaciones.
* Conexión al backend para autenticación y persistencia de sesión.

**Semana 3 - 9 mayo**

* UI de TAREAS: listado principal, añadir tarea, mostrar estado, filtros.
* Diseño adaptativo y conexión con API de tareas.
* Detalle y edición de TAREAS.
* Eliminación y actualización dinámica de la lista.

**Semana 10 - 16 mayo**

* Detalle y edición de TAREAS.
* Eliminación y actualización dinámica de la lista.
* Interfaz de DIARIOS: crear, editar y mostrar entradas vinculadas al usuario.

**Semana 17 - 23 mayo**

* Interfaz de DIARIOS: crear, editar y mostrar entradas vinculadas al usuario.

**Semana 24 - 30 mayo**

* Módulo de AMIGOS: visualización, solicitudes entrantes/salientes y acciones.
* Conexión completa con la lógica de backend para gestión de amistades.

**Semana 31 mayo - 5 junio**

* Ajustes del perfil de usuario: modificación de datos, cierre de sesión.
* Implementación de vista de calendario integrada con las tareas.

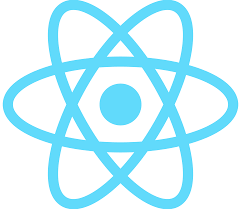
**LISTADO DE TAREAS**

**Diagrama de Gantt**

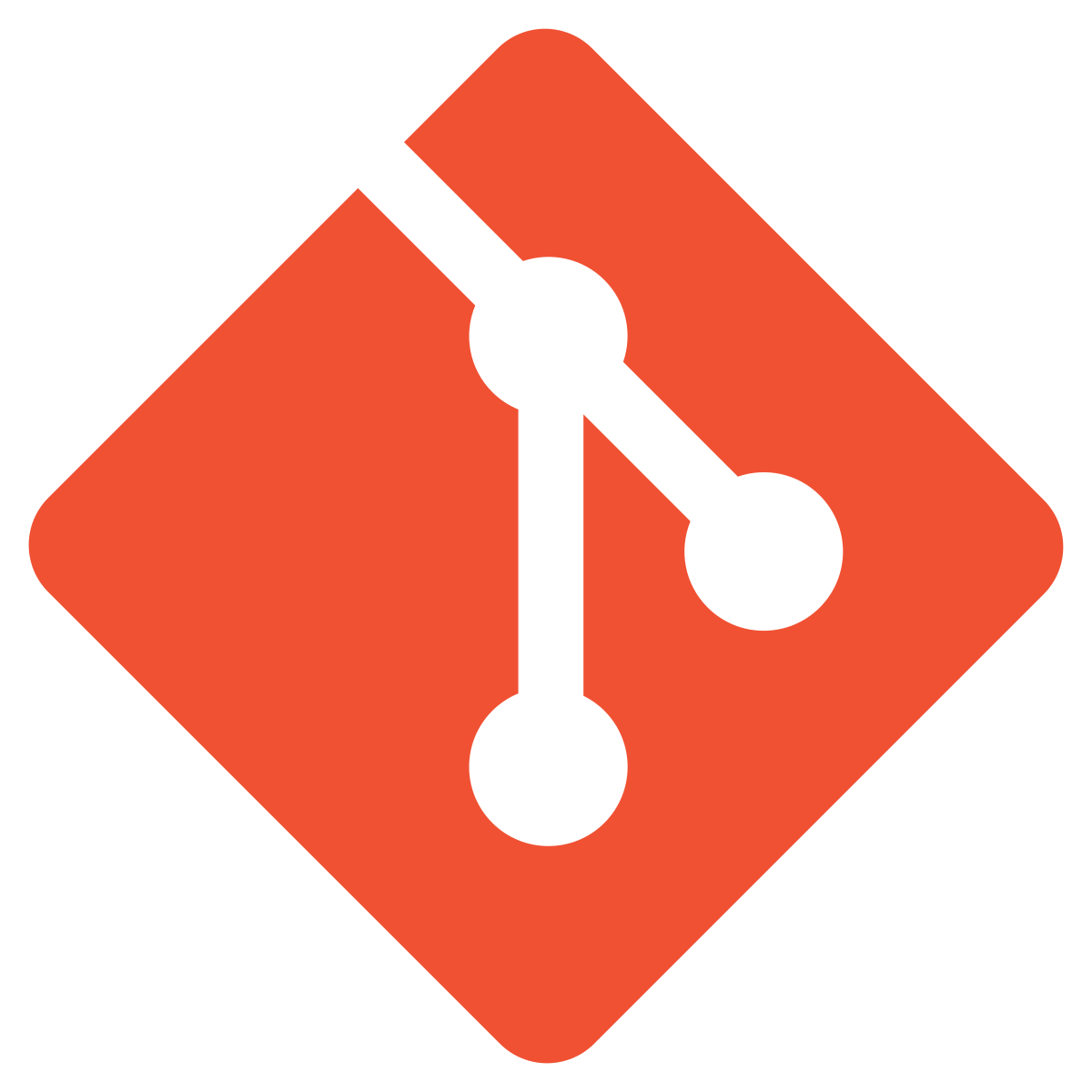
## 5.2. TECNOLOGÍAS

**AWS:** Se ha escogido por su facilidad de tener la aplicación en la nube. Con sus servicios como RDS para guardar la base de datos MySQL y EC2 para alojar el backend, facilitan el despligue de la página web.

**NodeJS:** Se utiliza como tecnología de backend por su capacidad para gestionar la transferencia de datos hacia la base de datos. Se aprovechan herramientas como Express para la creación de la API-REST, Zod para realizar validaciones de datos y Socket.io para crear los WebSocket (Midudev, s. f.)

**React**: Una de las bibliotecas de Javascript más usadas en el desarrollo frontend, por la facilidad que ofrece a la hora de programar con sus bibliotecas como **React-Router** para las rutas, **Redux** para la gestión de estados globales o **Axios** para hacer peticiones fetch al backend de manera más legible. También ofrece la creación de componentes, que hace que el código sea más escalable.

**TypeScript:** Se prefiere sobre JavaScript en el frontend por su tipado estático, lo que brinda mayor seguridad y claridad en el código, especialmente para desarrolladores acostumbrados a lenguajes tipados como Java. Esto contribuye a reducir la probabilidad de errores y a mejorar la mantenibilidad del código.

**Git y GitHub:** tecnologías para el control de versiones, he utilizado dado que son las tecnologías más utilizadas en esta área.

## 5.3. PROTOTIPADO Y USABILIDAD

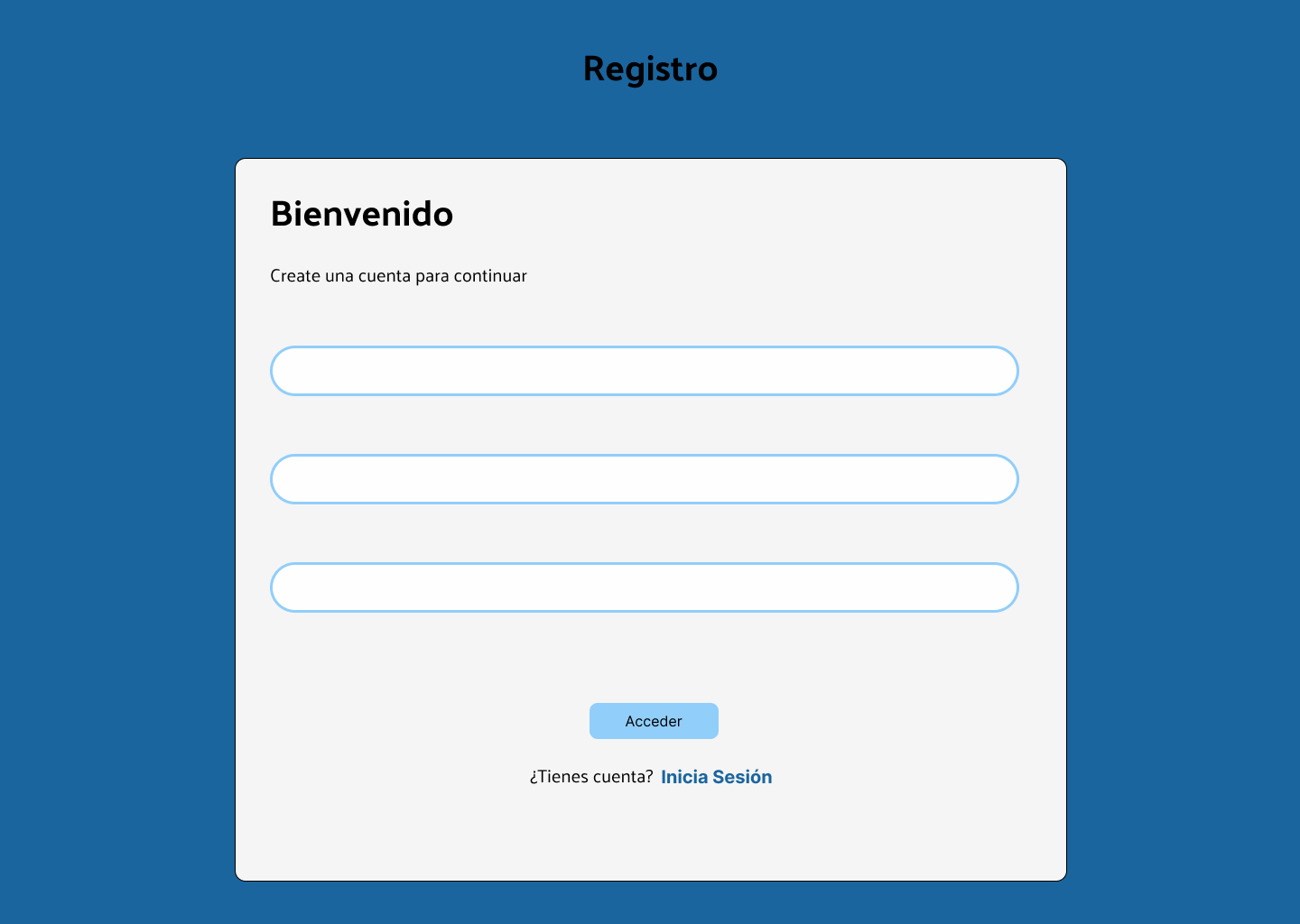
**PROTOTIPOS**

Para los prototipos se ha utilizado **Figma.**

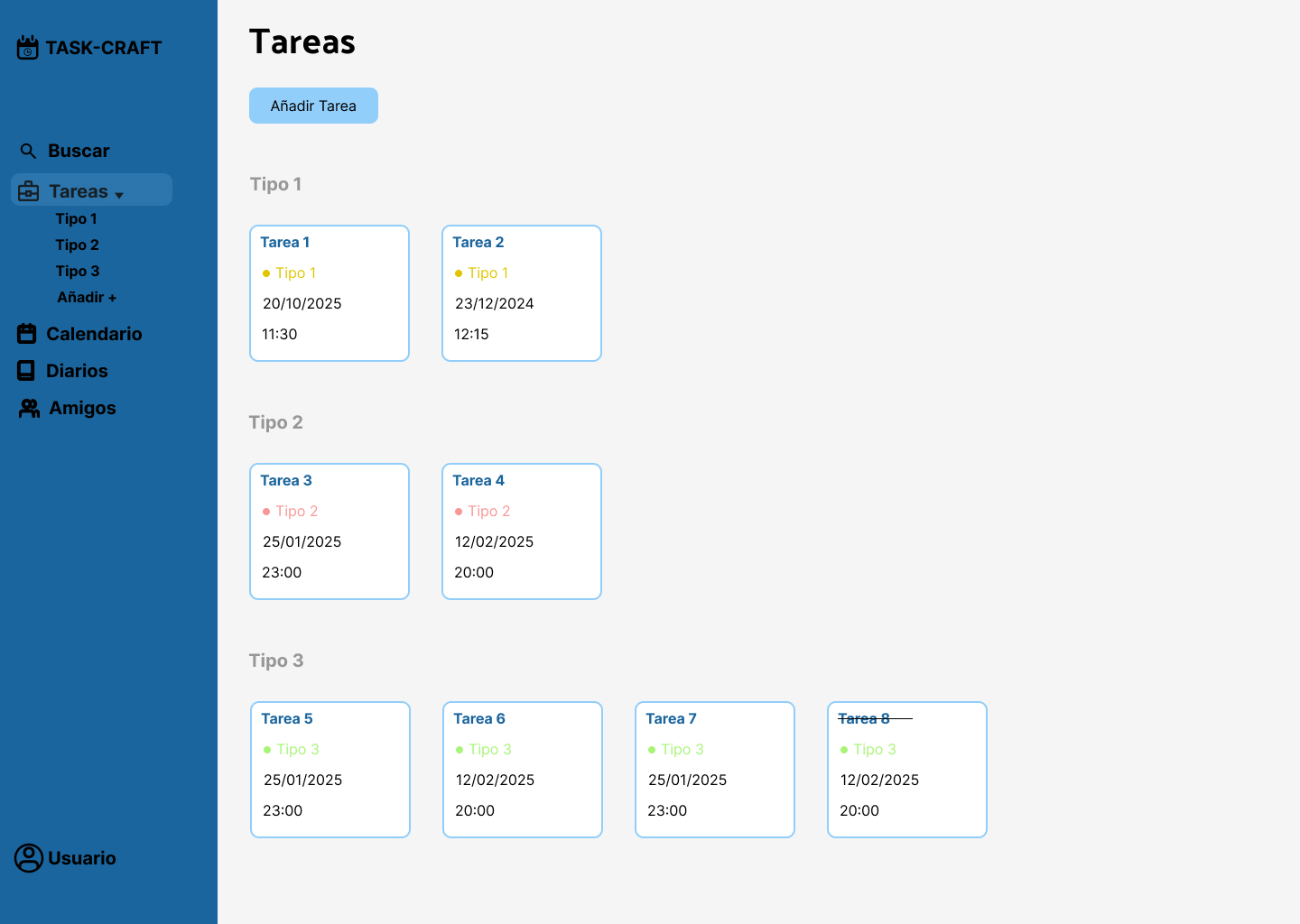
**Login:**



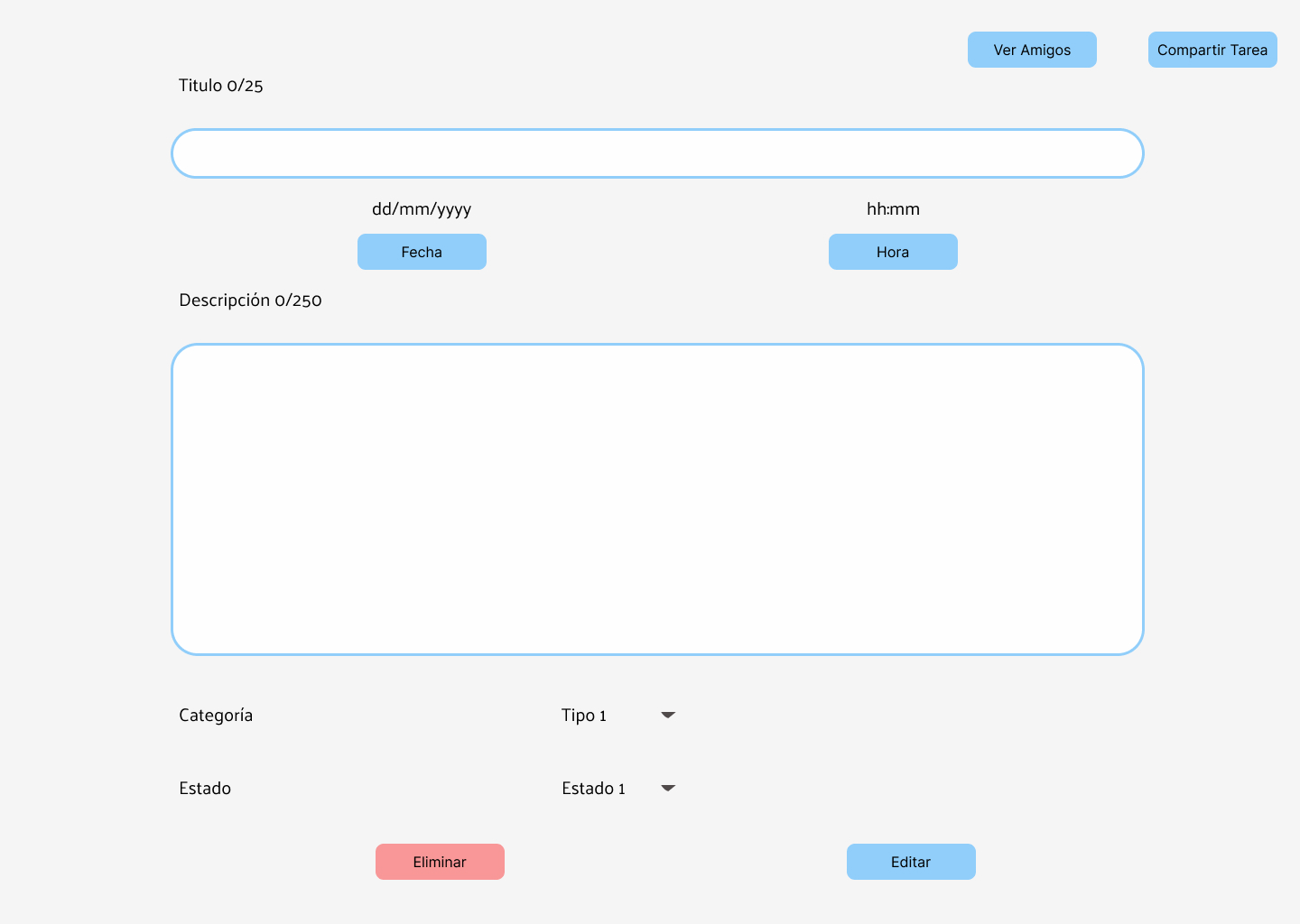
**Registro:**



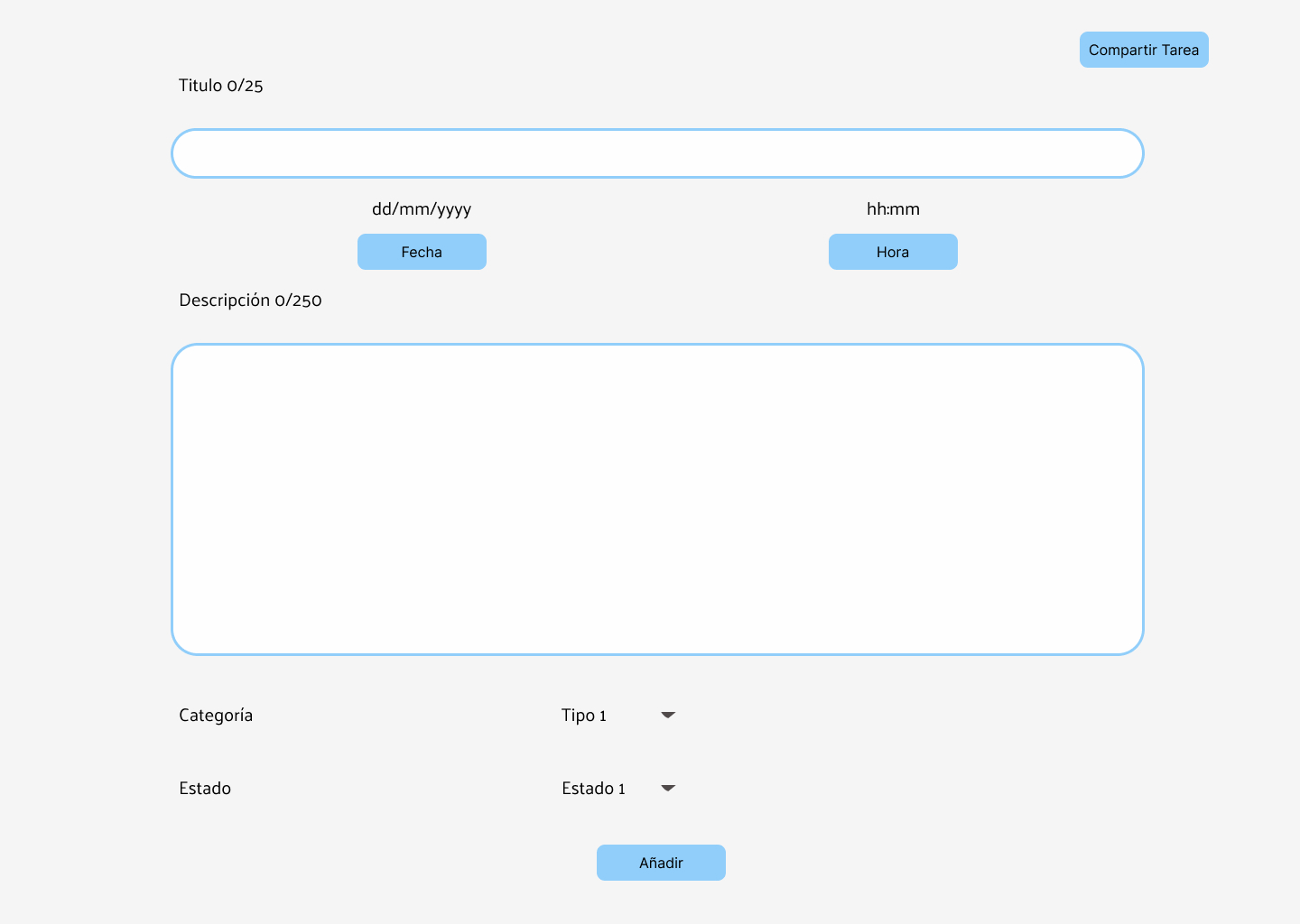
**Lista de Tareas:**



**Detalles Tareas:**



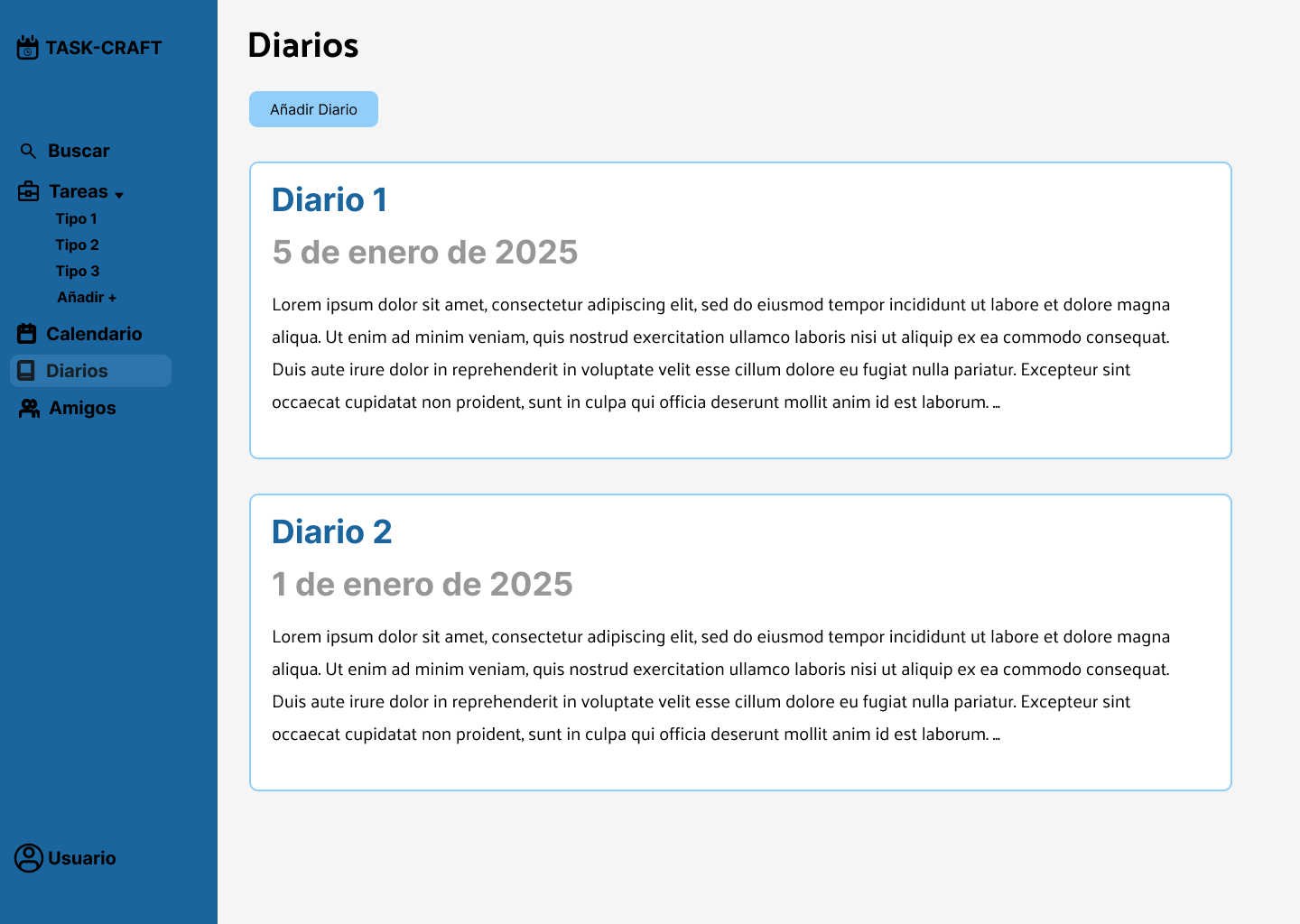
**Añadir Tareas**



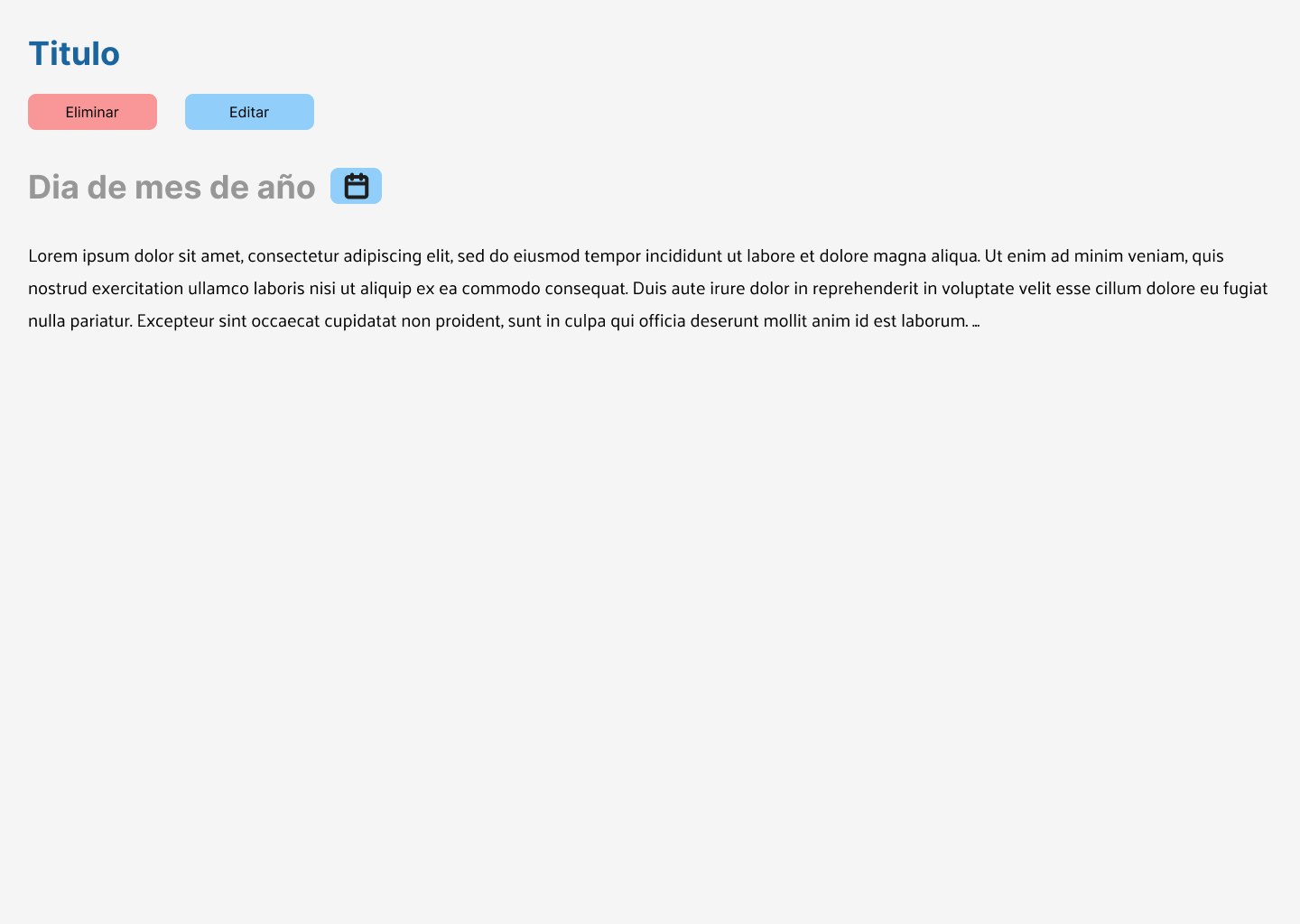
**Calendario (se utiliza componente de Javascript):**



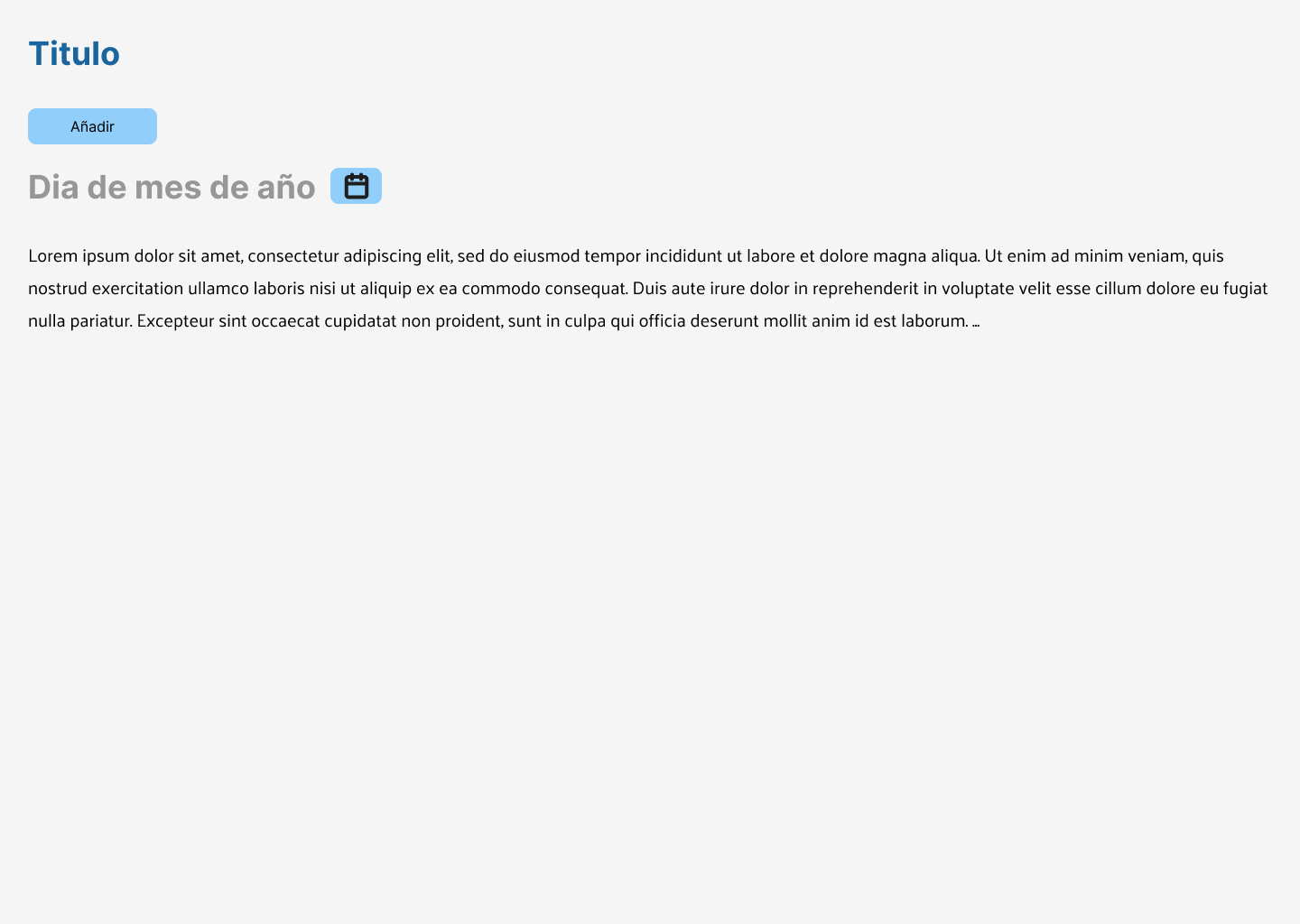
**Lista de Diarios:**



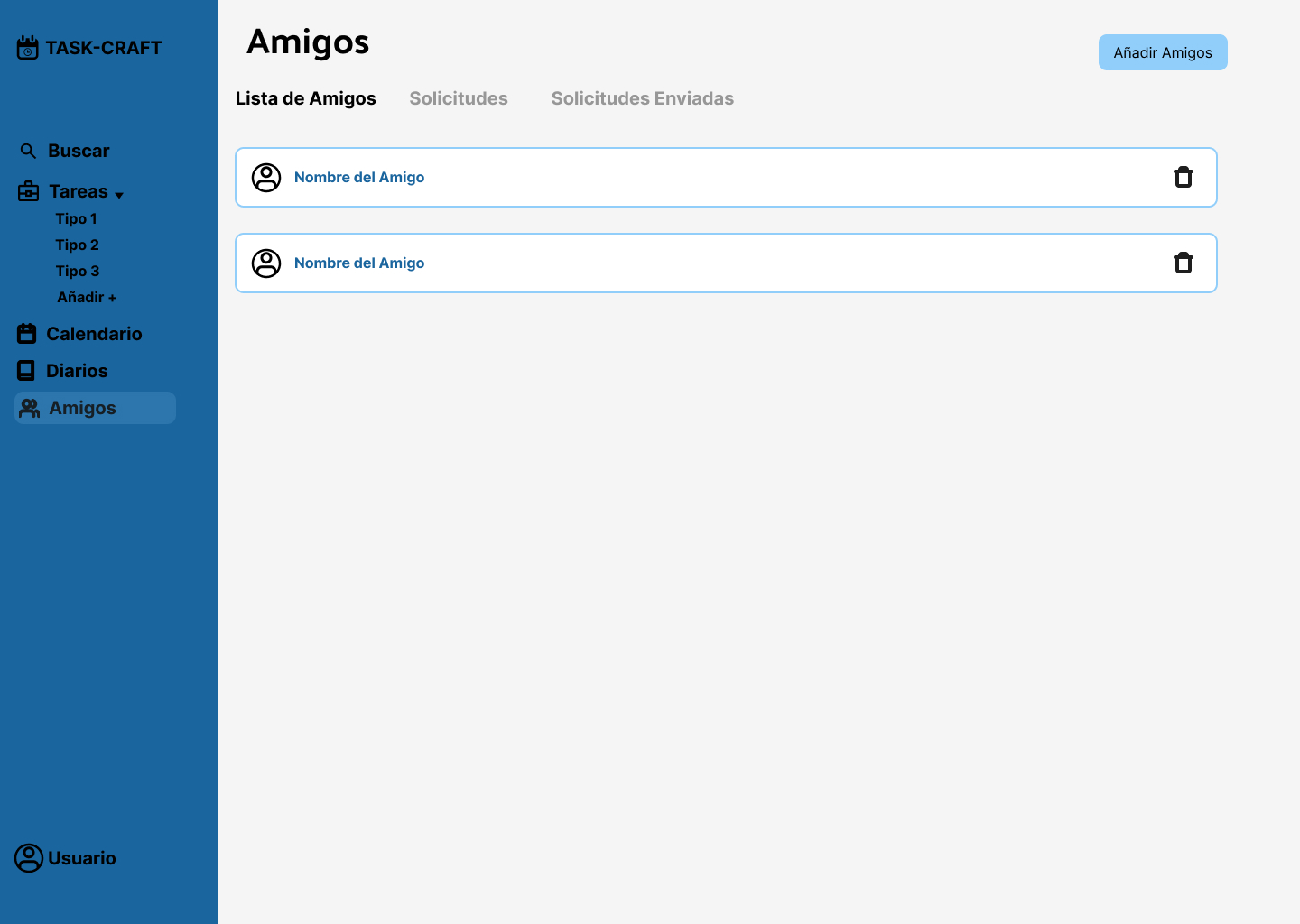
**Detalles de Diarios:**



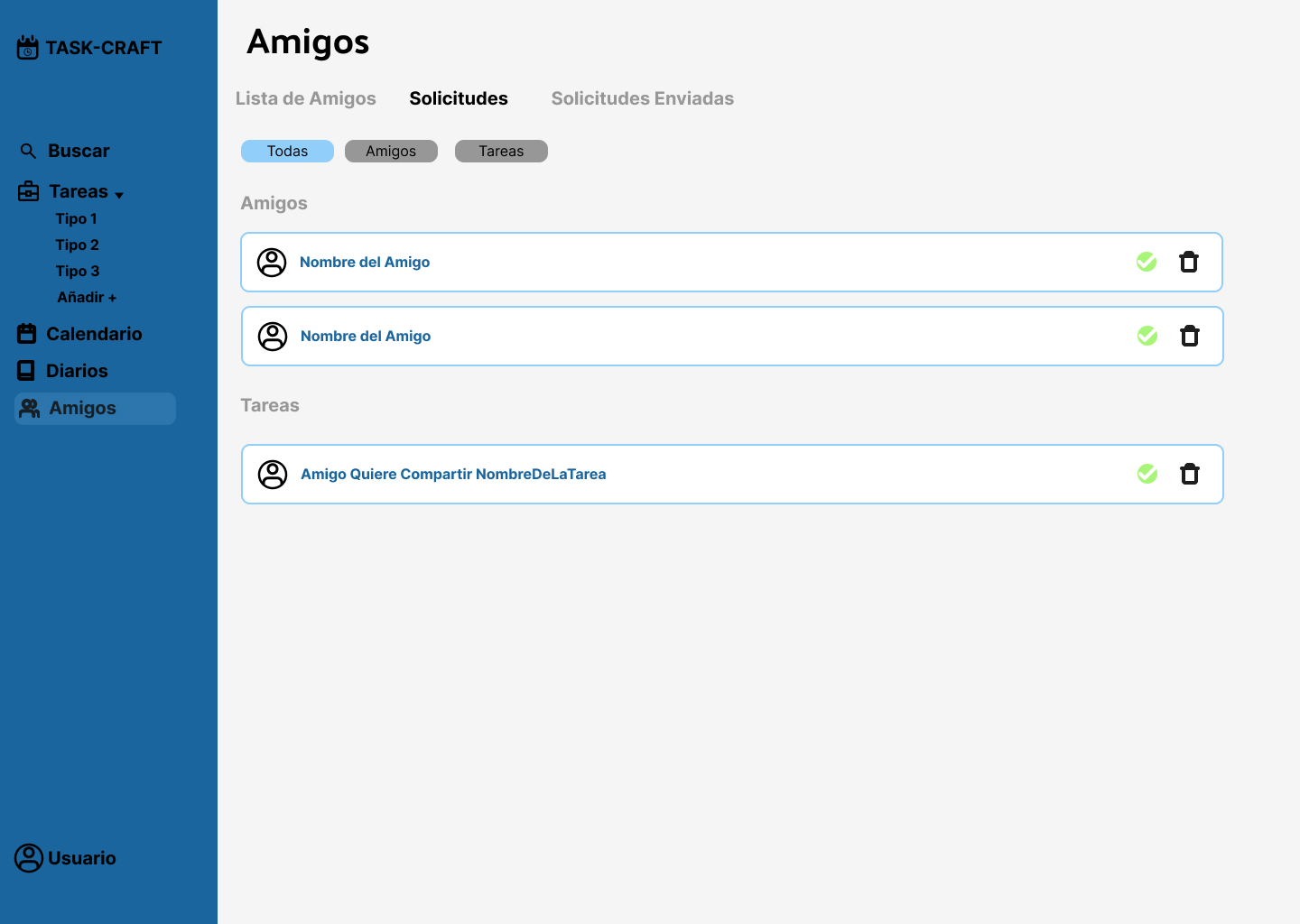
**Añadir Diario:**



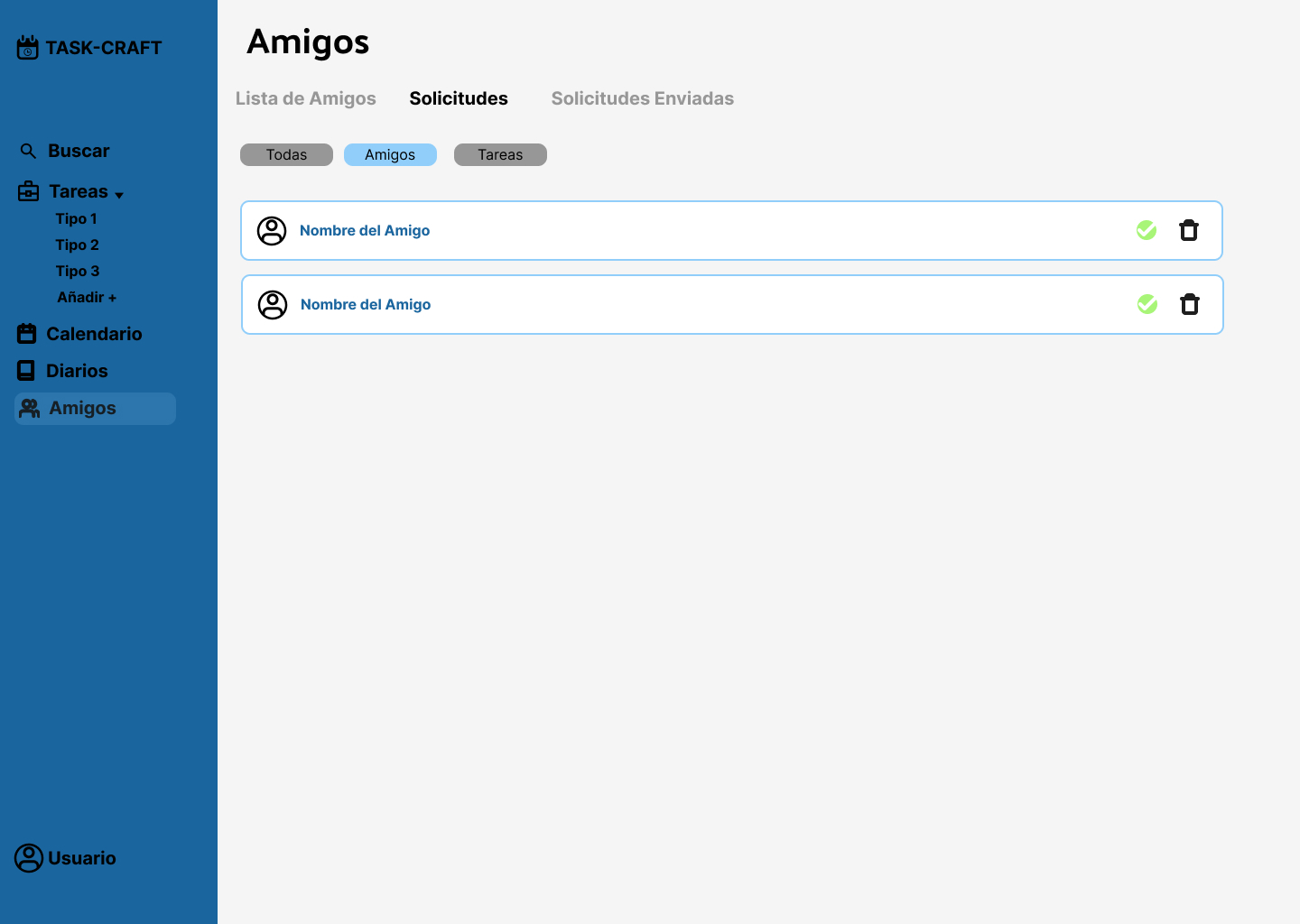
**Lista de Amigos:**



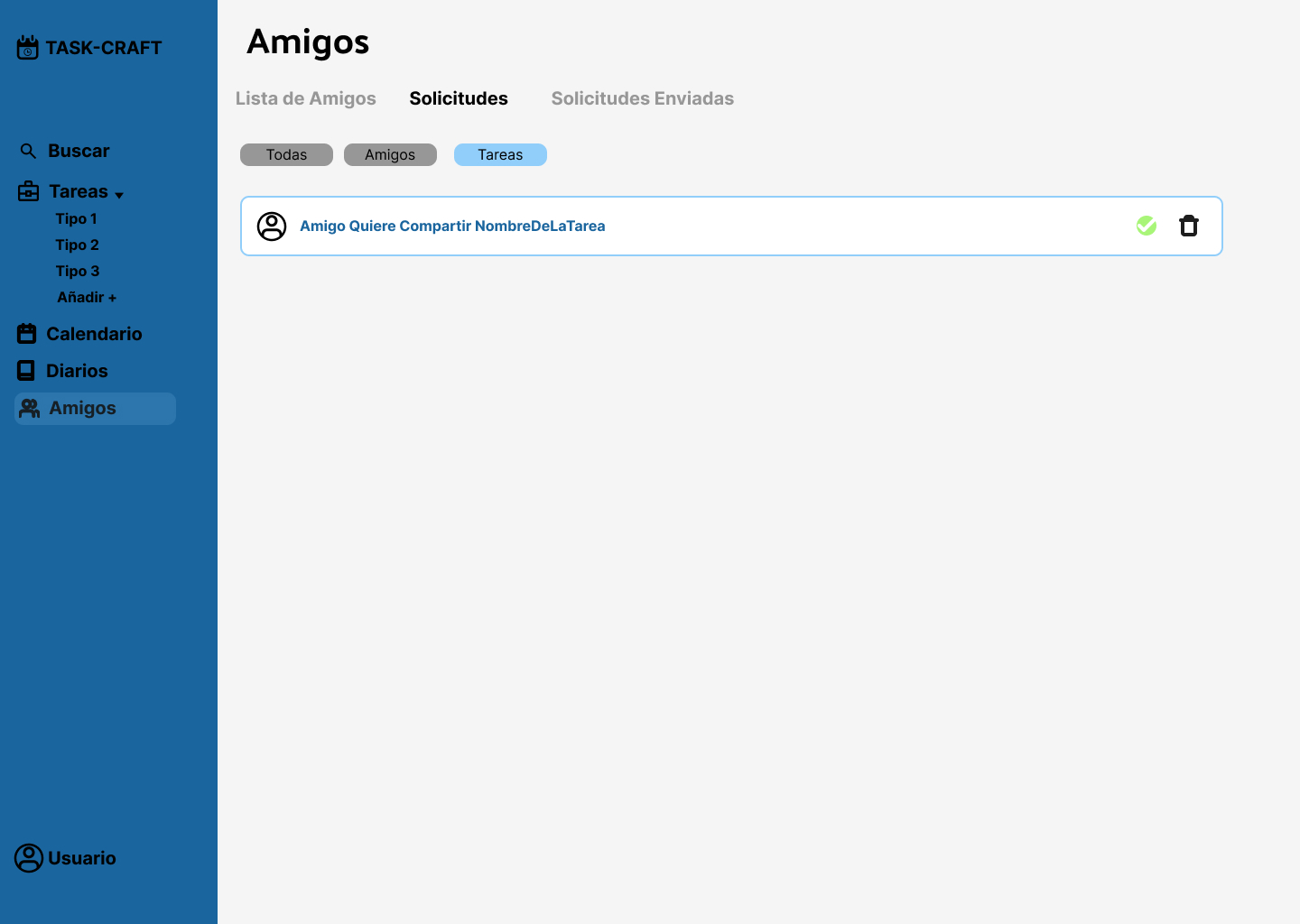
**Solicitudes Todas:**



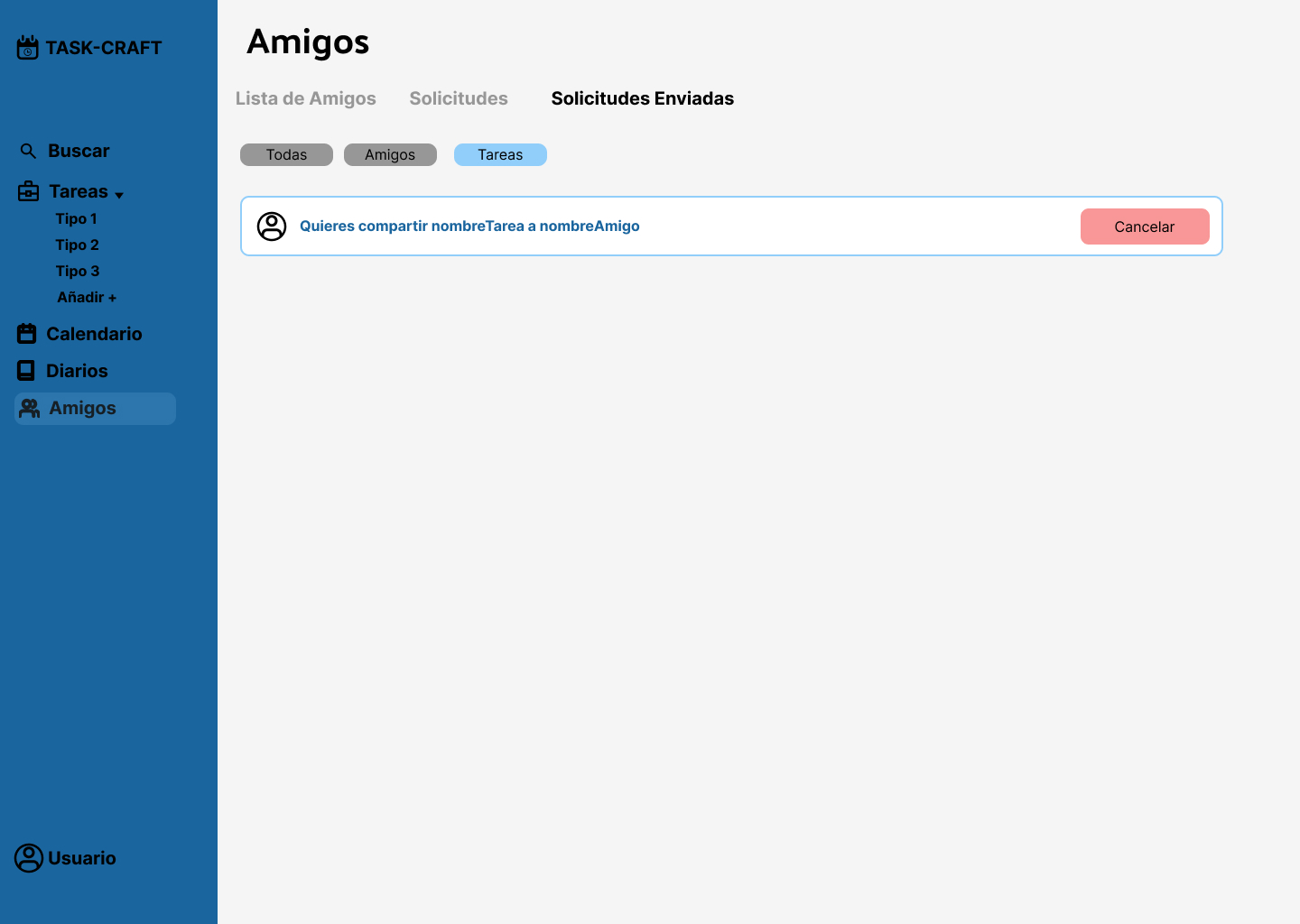
**Solicitudes de amigos:**



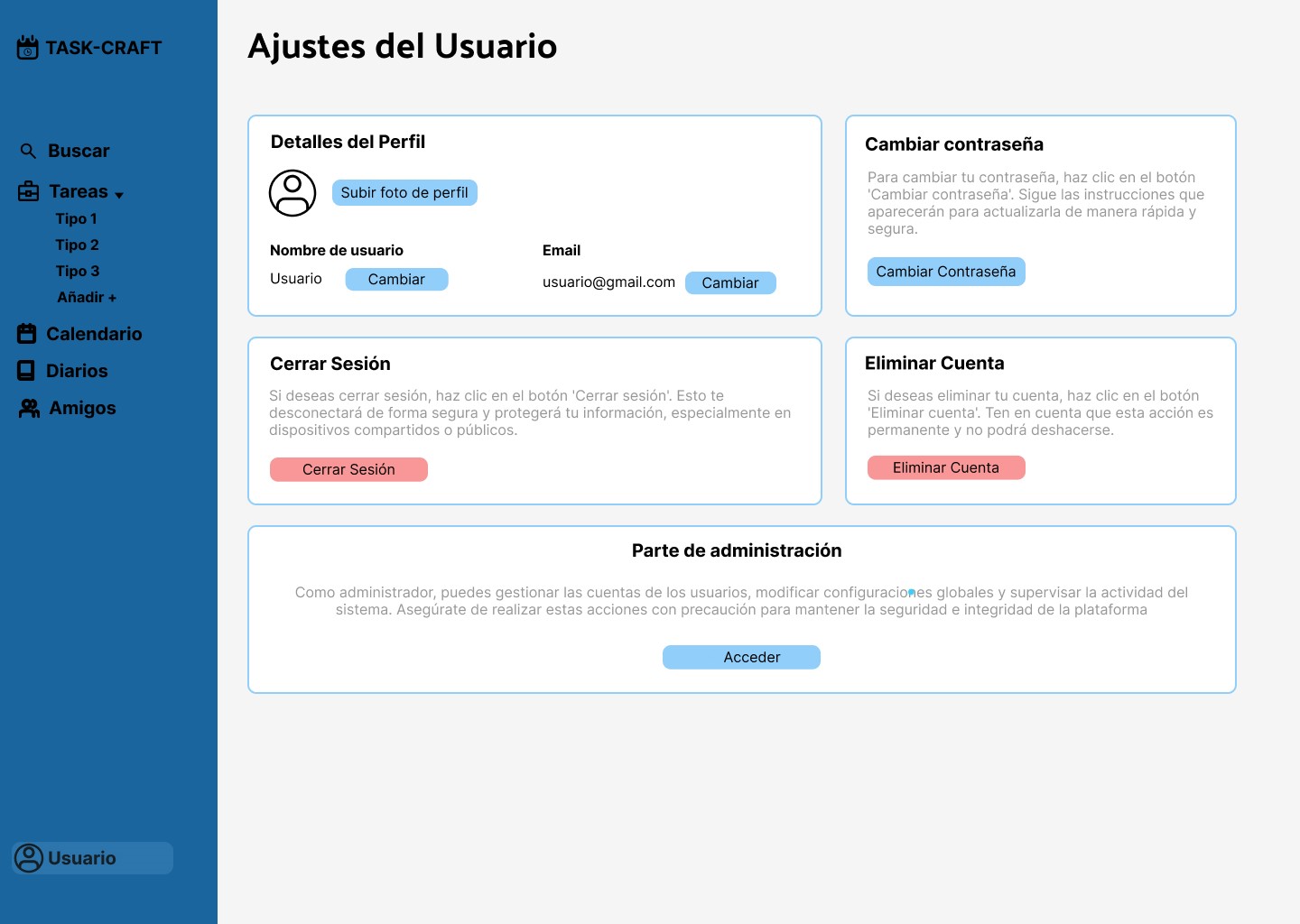
**Solicitudes de Amigos:**



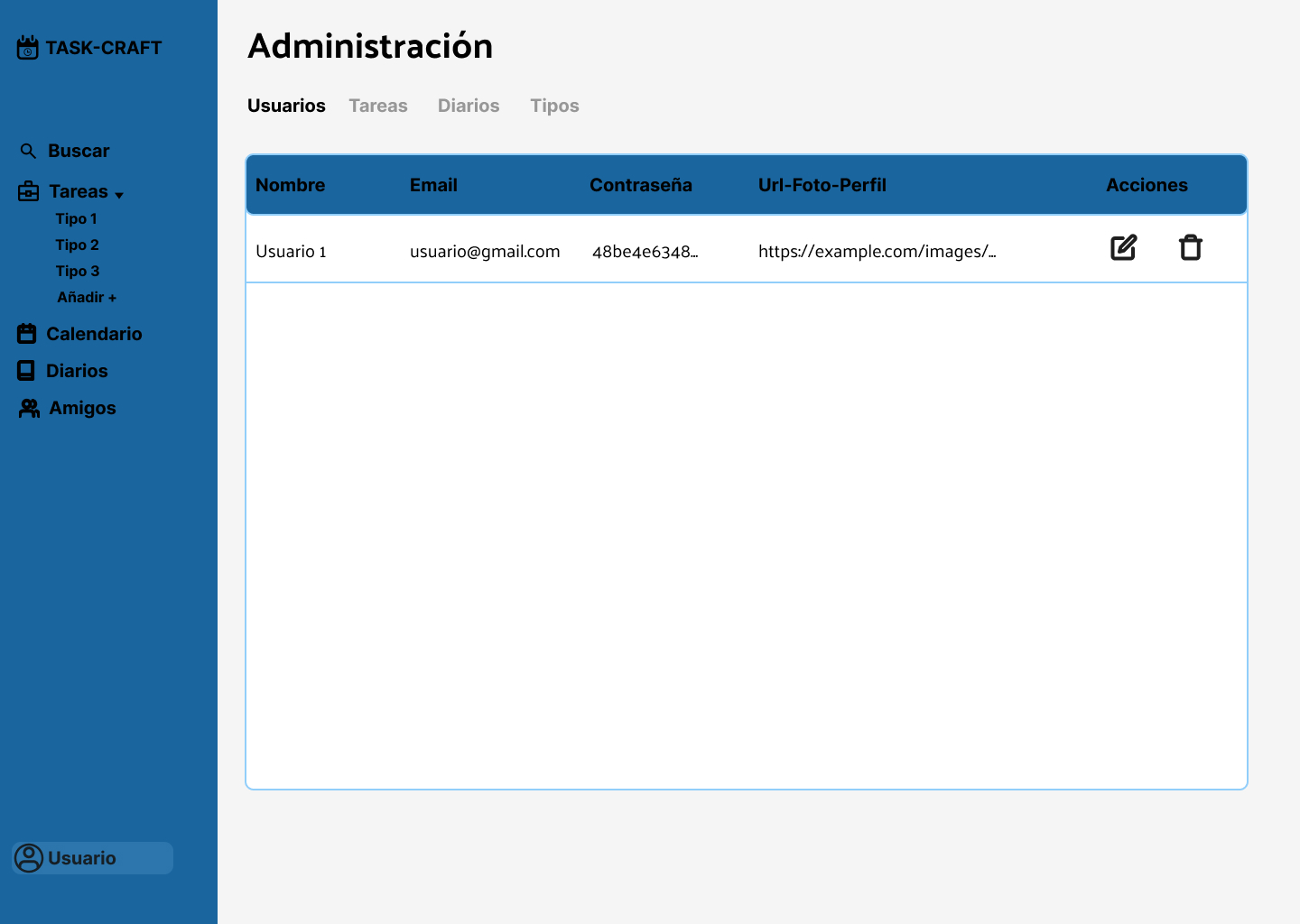
**Solicitudes Enviadas:**



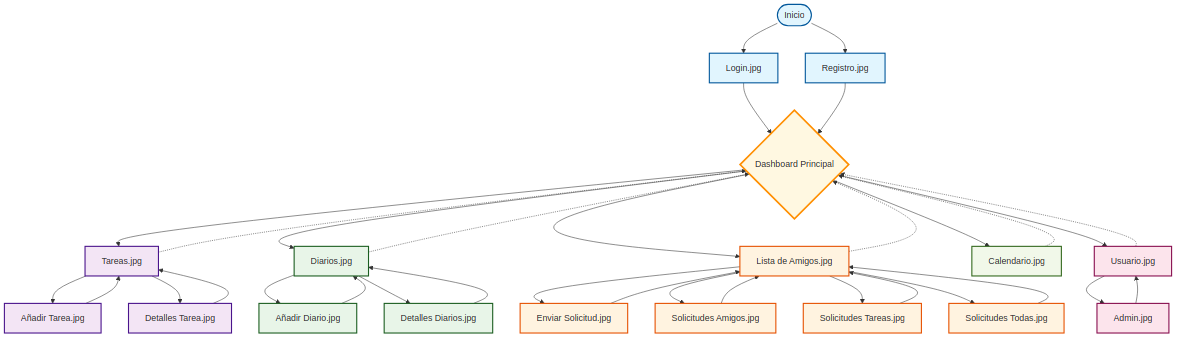
**Ajustes de Usuario:**



**Parte de Admin:**



**DIAGRAMA DE FLUJO:**



**USABILIDAD:**

**Diseño enfocado en la persona**

La disposición del sistema presenta una navegación intuitiva en la que el menú lateral, siempre visible, permite un rápido acceso a las secciones más relevantes: diario, amigos, administración, configuraciones, etc.

**Interfaz sencilla**

La combinación de colores entre azul, blanco y negro hace que resalte el contenido y evita la sobrecarga visual.

Las pantallas están dispuestas con un espacio bien configurado entre cada uno de los elementos, permitiendo a los usuarios concentrarse a la hora de asumir sus tareas.

**Flujo de tareas limpio**

Procesos tales como el **registro**, el **login**, la creación de entradas del diario y la gestión de amigos están bien separados en formularios simples y accesibles.

Las acciones como **"Enviar"**, **"Aceptar"**, **"Rechazar"**, **"Eliminar**" están bien posicionadas, utilizando gráficos y colores (verde, rojo) que facilitan su comprensión.

**Feedback inmediato y control del usuario**

En el apartado de amigos se proporciona un feedback evidente y rápido al enviar una solicitud de amistad o al aceptar una solicitud de amistad.

En los ajustes de usuario y de administración se proporciona confirmación e independencia entre las opciones (por ejemplo, cambio de contraseña o eliminación de cuenta).

**Accesibilidad y homogeneidad**

La utilización homogénea de botones, campos de introducción y tipografía, ayuda al usuario en la adaptación rápida a su entorno.

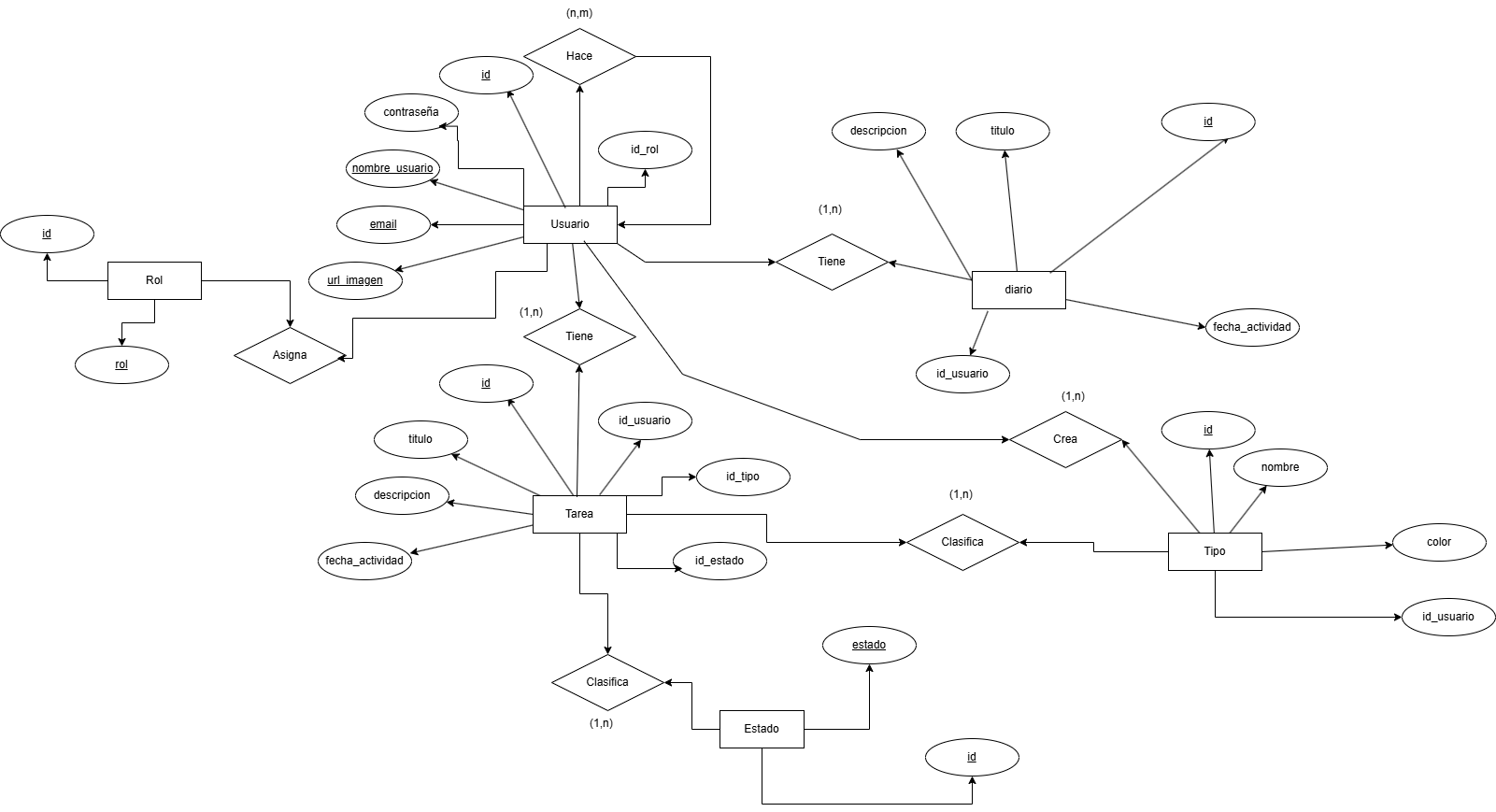
**Optimización para los diferentes niveles de usuario**

Los usuarios principiantes y avanzados pueden navegar sin una curva de aprendizaje pronunciada.

La interfaz busca la simplicidad y minimalismo, permitiendo una experiencia de usuario fluida y eficiente desde el primer uso.

## 5.4. MODELADO DE DATOS

**Modelo Entidad Relación:**



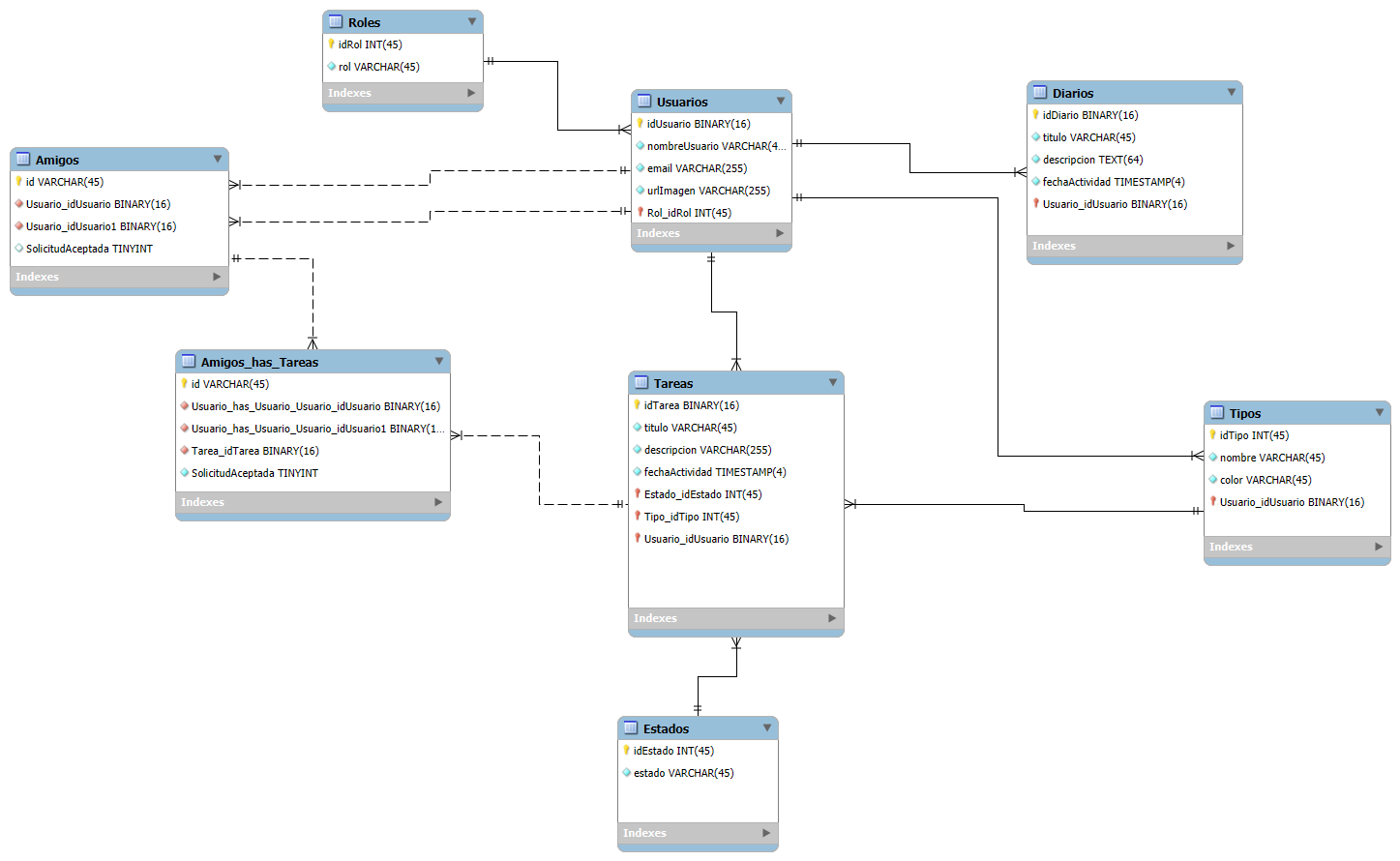
**Entidades:**

* Entidad usuario:
  + id: Clave primaria (Identificador único del usuario).
  + nombre: Nombre del usuario.
  + email: Dirección de correo electrónico del usuario. Atributo único.
  + url\_imagen: URL que lleva a la imagen del usuario. Atributo único.
  + contraseña: Contraseña del usuario.
  + idRol: Clave foránea que hacer referencia al id de la tabla rol. Esto establece el rol del usuario (admin, user, etc.).
* Entidad rol:
  + id: Clave primaria (Identificador único del rol).
  + rol: Nombre del rol.
* Entidad tarea:
  + id: Clave primaria (Identificador único de la tarea).
  + título: Título de la tarea.
  + descripción: Descripción de la tarea.
  + fecha\_actividad: Fecha para realizar la tarea.
  + categoría: Categoría de la tarea (por ejemplo, trabajo, personal, etc.).
  + id\_estado: Clave foránea que hace referencia al id de la tabla estado. Esto establece que un estado puede tener varias tareas, pero cada tarea está asignada a un solo estado.
  + id\_tipo: Clave foránea que hace referencia al id de la tabla tipo. Esto establece que un tipo puede tener varias tareas, pero cada tarea está asignada a un solo tipo.
  + id\_usuario: Clave foránea que hace referencia al id de la tabla usuario. Esto establece que un usuario puede tener varias tareas, pero cada tarea está asignada a un solo usuario.
* Entidad tipo:
  + id: Clave primaria (Identificador único del tipo).
  + nombre: Nombre del tipo (por ejemplo, "Urgente", "Personal").
  + color: Color asociado al tipo.
  + id\_usuario: Clave foránea que hace referencia al id de la tabla usuario. Esto establece que un tipo puede ser creado por un usuario, y cada usuario puede crear varios tipos.
* Entidad estado:
  + id: Clave primaria (Identificador único del estado).
  + estado: Nombre del tipo (por ejemplo, "Completo", "Incompleto").
* Entidad diario:
  + id: Clave primaria (Identificador único del diario).
  + descripción: Descripción del diario.
  + fecha: Fecha en la que se creó el diario.
  + id\_usuario: Clave foránea que hace referencia al id de la tabla usuario. Esto establece que un usuario puede tener varios diarios, pero un diario solo puede pertenecer a un único usuario.
* **Relaciones:**
  + Usuario - Tarea: Un usuario puede tener varias tareas (relación de 1 a N). Cada tarea pertenece a un solo usuario.
  + Usuario - Diario: Un usuario puede tener varios diarios (relación de 1 a N). Cada diario pertenece a un solo usuario.
  + Usuario - Usuario (supervisor): Un usuario puede hacer amigos (otros usuarios) (relación autorreferencial). No todos los usuarios pueden tener amigos, por lo que esta relación es opcional.
  + Usuario - Tipo: Un usuario puede crear varios tipos (relación de 1 a N). Cada tipo pertenece a un solo usuario.
  + Rol - Usuario: Un rol puede asignar a varios usuarios (relación de 1 a N). Cada usuario pertenece a un solo rol.
  + Tipo - Tarea: Un tipo puede estar clasificada por varias tareas (relación de 1 a N). Cada tarea pertenece a un solo tipo.
  + Estado - Tarea: Un estado puede estar clasificada por varias tareas (relación de 1 a N). Cada tarea pertenece a un solo tipo.
* **Indices:**

1. Claves primarias
   * Tabla rol: Índice en id.
   * Tabla usuario: Índice en id.
   * Tabla tarea: Índice en id.
   * Tabla tipo: Índice en id.
   * Tabla estado: Índice en id.
   * Tabla diario: Índice en id.
2. Claves foráneas

* Tabla usuario:
* Índice en id\_rol (relación con rol).
* Tabla tarea:
* Índice en id\_usuario (relación con usuario).
* Índice en id\_estado (relación con estado).
* Índice en id\_tipo (relación con tipo).
* Tabla tipo:
* Índice en id\_usuario (relación con usuario).
* Tabla diario:
* Índice en id\_usuario (relación con usuario).

**Modelo Relacional**



**Tablas**

* **Roles**
  + id\_rol → id\_rol. Identificador único para cada rol.
  + nombre\_rol → nombre del rol. (ejemplo: Administrador, Usuario)
* **Usuarios**
  + id\_usuario → id\_usuario. Identificador único del usuario
  + nombre → nombre. Nombre del usuario
  + email → email. Correo del usuario
  + contraseña → Contraseña cifrada del usuario para su autenticación.
  + id\_rol → Relación con la tabla Roles, indicando el tipo de usuario.
* **Diarios**
  + id\_diario → id\_diario. Identificador único del diario
  + titulo → titulo del diario
  + descripcion → Contenido o descripción del diario.
  + fecha\_creacion → Fecha donde se creó.
  + id\_usuario → Relación con Usuarios.
* **Tareas**
  + id\_tarea → Identificador único de la tarea.
  + titulo → Nombre de la tarea.
  + descripcion → Detalles sobre la tarea.
  + fecha\_vencimiento → Fecha límite para completar la tarea.
  + id\_usuario → Relación con Usuarios, indicando quién la creó.
  + id\_estado → Relación con Estados, indicando el estado actual de la tarea.
  + id\_tipo → Relación con Tipos, indicando la clasificación de la tarea.
  + id\_usuario → Relación con Usuarios.
* **Estados**
  + id\_estado → Identificador único del estado.
  + nombre\_estado → Nombre del estado (pendiente, en progreso, completada).
* **Tipos**
  + id\_tipo → Identificador único del tipo de tarea.
  + nombre\_tipo → Categoría de la tarea (trabajo, personal, urgente).
  + color\_tipo → Color representativo del tipo de tarea en la interfaz.
  + id\_usuario → Relación con Usuarios.
* **Amigos**
  + id\_amigo → Identificador único de la relación de amistad.
  + id\_usuario\_1 → Usuario que envió la solicitud de amistad.
  + id\_usuario\_2 → Usuario que recibió la solicitud de amistad.
  + solicitudAceptada → Estado de la relación (booleano).
* **Amigos\_has\_Tareas**
  + id\_amigo → Relación con Amigos, indicando qué amistad comparte tareas.
  + id\_usuario\_1 → Usuario que envió la solicitud de compartir tarea.
  + id\_usuario\_2 → Usuario que recibió la solicitud de compartir tarea.
  + id\_tarea → Relación con Tareas, indicando qué tarea está compartida.
  + solicitudAceptada → Estado de la relación (booleano).

**Relaciones**

* **Usuarios ↔ Roles:** Cada usuario tiene un rol asignado.
* **Usuarios ↔ Diarios:** Cada usuario puede tener varios diarios.
* **Usuarios ↔ Amigos:** Gestiona las solicitudes de amistad entre los usuarios.
* **Usuarios ↔ Tareas:** Un usuario puede crear múltiples tareas.
* **Tareas ↔ Estados y Tipos:** Cada tarea tiene un estado y una clasificación.

## 5.5. SEGURIDAD Y AUTENTICACIÓN

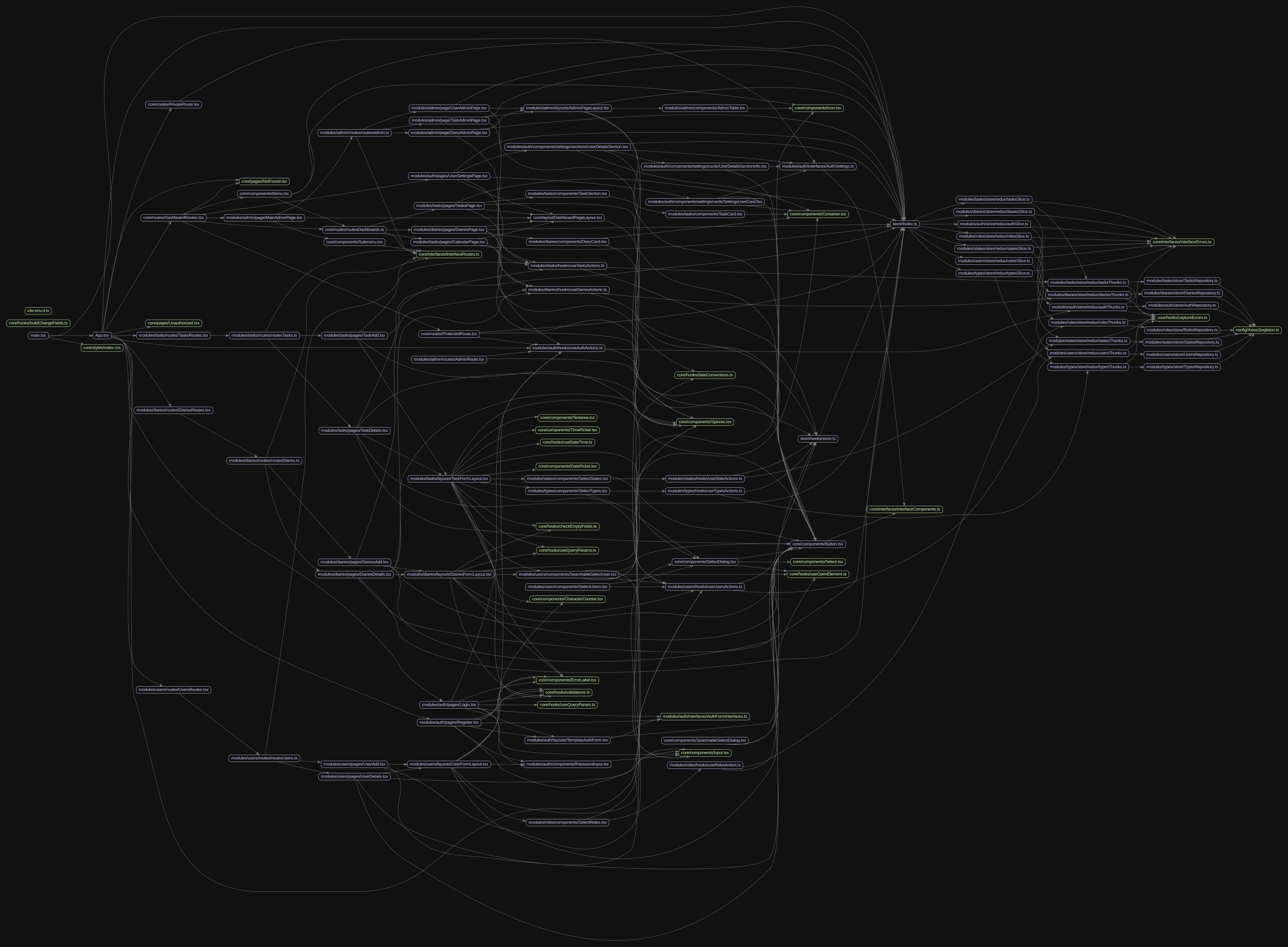
Para la seguridad de la aplicación se ha implementado una librería de **Node.js** para hashear contraseñas llamada **bcrypt,** el cual aplica un algoritmo unidireccional, lo que significa que no se puede revertir la contraseña original. Esta contraseña se guardará encriptada en la base de datos. Para la autenticación se ha utilizado **JWT (JSON Web Tokens),** el cual permite crear y verificar tokens, para mantener sesiones. Este token se guarda en una **cookie** con el plugin **cookie-parser,** el cual solo el backend puede modificar. La combinación de ambas tecnologías es la siguiente:

* El usuario se registra → **hasheas** su contraseña con **bcrypt** y la guardas.
* El usuario inicia sesión → **comparas** la contraseña con **bcrypt.compare.**
* Generas un **JWT** y lo envías al cliente **dentro de una cookie HTTP-only y Secure**, para que **solo el backend pueda gestionarla** (ni JavaScript del cliente puede accederla).
* Luego, en cada petición protegida, el usuario envía el **token** en los headers (ej: Authorization: Bearer <token>), y tú lo verificas con **jwt.verify**.

## 5.6. DISEÑO E IMPLEMENTACIÓN

**DIAGRAMAS DE CLASES:**

**Frontend:**



**Backend:**

****EXPLICACIÓN DE LA ARQUITECTURA DE LA APLICACIÓN****

**La arquitectura de la aplicación sigue el modelo cliente/servidor, donde un cliente (frontend) se comunica con el servidor a través de una **API RESTful**. El servidor ha sido implementado mediante el uso de un patrón de diseño **MVC** **(Modelo-Vista-Controlador)**, el cual promueve una mejor separación de las responsabilidades y el mantenimiento del código fuente.**

**Cada entidad de la base de datos es gestionada a través de una **DAO (Data Access Object)**, el cual encapsula toda la lógica de acceso a datos, al igual que permite mantener separada la lógica de negocio del acceso directo a la base de datos. Cada carpeta del backend es estructurada siguiendo los principios de la **scream architecture**, donde cada funcionalidad tiene su propio módulo separado, por ejemplo usuarios, amigos, tareas...**

****IMPLEMENTACIÓN****

**El cliente realiza peticiones al servidor mediante el uso de **fetch**. En este caso para realizar esas peticiones de forma legible, se ha utilizado **Axios**, una librería de JavaScript/TypeScript, donde se envia datos o se solicitan recursos desde la interfaz de usuario. Estas peticiones son procesadas por el backend, el cual ejecuta la lógica correspondiente, consulta a la base de datos a través de toda la lógica de acceso a datos (DAO) y devuelve los datos necesarios.**

****HERRAMIENTAS:****

* ****Visual Studio Code (VSCode)**** como entorno de desarrollo.
* ****Postman**** para probar y validar las rutas de la API.
* ****GitHub**** para el control de versiones del proyecto.
* ****MySQL Workbench**** como interfaz gráfica para gestionar la base de datos **MySQL**, conectada a **RDS de AWS**.

## 5.7. PRUEBAS Y CALIDAD DEL SOFTWARE

**R1 – Inicio de sesión del usuario**

* Se ha ingresado una dirección de correo y contraseña incorrectos y, después de ello, el sistema nos ha mostrado el mensaje de error. Prueba superada.
* Se ha ingresado una combinación que corresponde a unas credenciales correctas, las cuales se encuentran en la base de datos y, dado este hecho, se ha conseguido acceder al sistema de forma exitosa. Prueba superada.

**R2 – Registro de usuario**

* Se ha probado el formulario de registro, se han utilizado credenciales inválidas y se ha visualizado un mensaje informando del error. Prueba superada.
* Con credenciales válidas, el usuario se almacena correctamente en la base de datos. Prueba superada.
* Intentando guardar usuarios con datos inválidos se producen errores y, por lo tanto, no se registran. Prueba superada.

**R3 – Gestión de Diarios.**

* La petición Get devuelve la lista de diarios correctamente.
* La petición Post crea un diario en la base de datos con éxito.
* La petición Put actualiza correctamente un diario.
* La petición Delete elimina correctamente un diario.
* Al acceder, los datos del diario se cargan desde la base de datos y se muestran en pantalla (tanto para un nuevo diario como para uno que ya existía).
* Modificaciones realizadas se reflejan en la vista de la lista de diarios.
* La base de datos ha sido actualizada correctamente tras editar.

**R4 – Gestión de Tareas**

* La vista de la lista de tareas refleja correctamente los cambios.
* La vista de diarios refleja los cambios correctamente.
* La base de datos muestra correctamente la modificación.
* GET de tareas funciona correctamente.
* POST de tareas se crean correctamente.
* PUT modifica correctamente una tarea.
* DELETE elimina correctamente la tarea.
* Los datos de tareas se cargan correctamente en la pantalla.
* La información se muestra correctamente en pantalla.
* La base de datos refleja los cambios de edición.
* El calendario muestra correctamente la tarea modificada.
* La vista de tareas refleja correctamente los cambios de modificaciones.
* El calendario muestra correctamente las tareas editadas.
* La vista de tareas refleja correctamente los cambios de modificaciones.
* El calendario se actualiza correctamente con el dato modificado.
* Los cambios se reflejan correctamente en la vista de la lista de tareas.
* La base de datos refleja correctamente el cambio que esperamos.
* El calendario refleja correctamente la modificación.
* GET de tareas funciona correctamente.
* POST de tareas añade datos correctamente.
* PUT modifica correctamente los datos de tareas.
* DELETE elimina correctamente la tarea.
* El botón de acción en tareas funciona correctamente.
* El cambio se refleja en la base de datos tras la acción.
* El botón se comporta como se esperaba en la interfaz.
* La base de datos reflejará el cambio.

**R5 – Gestión de Usuario**

* GET de usuarios responde correctamente.
* POST crea nuevos usuarios sin errores.
* PUT actualiza correctamente los datos del usuario.
* DELETE elimina usuarios según lo esperado.
* La base de datos refleja el cambio de nombre.
* La interfaz para editar el nombre funciona correctamente.
* El cambio de contraseña se realiza correctamente.
* La interfaz de cambio de contraseña es funcional.
* El formulario de subida de imagen funciona correctamente.
* La imagen se guarda correctamente en el servidor.
* La URL de la imagen se ha establecido como foto de perfil.
* Los datos del usuario persisten tras recargar.
* La UI para cambiar email funciona correctamente.
* El email se actualiza correctamente en la base de datos.
* El botón 'Cerrar sesión' cierra correctamente la sesión.
* Se ha verificado que el usuario ha cerrado sesión.
* El botón 'Eliminar cuenta' responde como se espera.
* El usuario se elimina correctamente de la base de datos.

### **R6 – Aplicación web responsive**

* **R6F1T1P1**: La aplicación se ha visualizado correctamente en diferentes tamaños de pantalla y dispositivos.

# DESPLIEGUE DEL PRODUCTO

Se ha decidido utilizar servicios de Amazon Web Services (AWS) para desplegar el producto final. Esta decisión se ha tomado aprovechando aspectos como la fiabilidad, la escalabilidad y la disponibilidad de la solución. La arquitectura de despliegue queda repartida en:

**Uso de EC2:**

Se ha utilizado una **instancia EC2** ejecutando **Amazon Linux**, donde se aloja tanto el backend (API REST desarrollada en Node.js) como el frontend (aplicación SPA creada con Vite y React). Para garantizar una arquitectura estable y escalable, se ha utilizado el servidor web **Apache (httpd)** como **proxy inverso** y servidor de archivos estáticos.

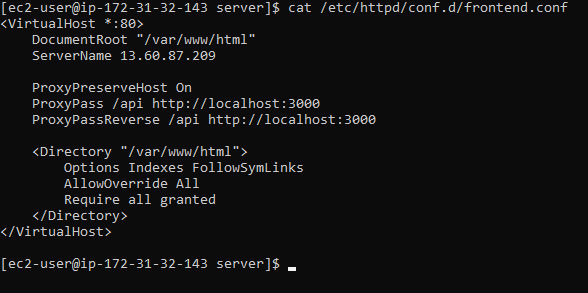
El **backend** se ejecuta mediante un servidor **Express** que escucha por defecto en el puerto 3000. Para evitar que el proceso se detenga al cerrar la sesión SSH y garantizar su persistencia, se ha utilizado **PM2** como gestor de procesos:

El **frontend** ha sido construido con **Vite**, el cual te generará un directorio ***/dist.*** El contenido del directorio generado se ha copiado a la carpeta servida por Apache, el ***/var/www/html/*.**

La configuración de Apache cumple dos funciones:

1. **Servir los archivos estáticos del frontend**.
2. **Redirigir las llamadas API a través de un proxy hacia el backend en** localhost:3000.

Se ha creado y configurado un archivo ***/etc/httpd/conf.d/frontend.conf:***



Al final la aplicación está alojada en esta url: <http://13.60.87.209/>.

**Base de datos en RDS (MySQL):**

El sistema de almacenamiento será Amazon RDS con motor MySQL. Esta base de datos relacional es totalmente gestionada por AWS lo que simplifica tareas como las copias de seguridad automáticas, la recuperación ante fallos, la actualización del software, la escalabilidad vertical... La instancia RDS quedará asociada de forma segura con el backend desplegado en EC2 con lo que se asegura baja latencia y que los datos quedan protegidos mediante configuraciones de red privada (VPC).

# PRESUPUESTO, RIESGOS, VIABILIDAD

**PRESUPUESTO**

Horas de trabajo estimadas

1. Diseño
   * Diseño de Interfaz de Usuario(UI) y Experiencia de Usuario(UX): **28h**
2. Desarrollo
   * Desarrollo Frontend (Interfaz del Usuario): **20h**
   * Desarrollo Backend(Servidor y Base de Datos): **39h**
   * Integración y Sincronización de Datos: **10h**
3. Pruebas y Calidad
   * Pruebas de Funcionalidad: **15h**
   * Corrección de Errores y Optimización: **15h**
4. Despliegue **: 16h**

**Presupuesto total:**

• 145h x 14,5€/h= **2102,5€** (El precio no incluye el servicio de AWS)

**RIESGOS**

Durante el desarrollo, han sido considerados los siguientes riesgos:

* Pérdida de información por fallo en el equipo o faltas de backups.
* Fallo en la integración de servicios externos (ejemplo APIs o servicios en la nube).
* Cambios de requisitos o alcance que podrían aparecer durante el desarrollo y que podrían influir en el cronograma.
* Saturación del servidor en el caso de un elevado crecimiento de usuarios.
* Problemas de Seguridad como inyecciones SQL, XSS y otras vulnerabilidades en caso que no se sigan buenas prácticas.

**VIABILIDAD**

La aplicación es viable ya que:

* Se utilizan tecnologías muy extendidas (Node.js, Express, bases de datos relacionales, etc.).
* Es fácilmente escalable gracias a la posibilidad de utilizar servicios como el que ofrece AWS.
* La interfaz está pensada para una buena usabilidad, basándose en pruebas y prototipos.
* Puede durar en el tiempo debido a su estructura modular y a su documentación suficiente.

# DIFICULTADES ENCONTRADAS Y RESOLUCIÓN

Los desafíos que se han enfrentado durante el desarrollo del proyecto son:

1. **Test unitarios en el back-end:** la adaptación a los tests sumado a la inexperiencia en realizar testing hizo costosa la implementación de las pruebas
2. **Implementar los JWT**: al ser una tecnología que no se había visto, implementarlo en el proyecto fue una experiencia que tardó más de lo esperado.
3. **Implementar Compartir Tarea y Amigos:** se quiso implementar los Websockets en la aplicación para realizar solicitudes de amistad o de compartir tareas. En el backend se ha testeado y es funcional, pero falta implementar esa lógica en el frontend y testearlo ya que se priorizó otras funcionalidades para realizar la entrega del proyecto. Esto es una **mejora a futuro.**
4. **Subida de imágenes:** Al ser una implementación en la cual hay que configurar muchos servicios, se ha decido no hacer para el día de la defensa para enfocarse en entregar una versión funcional del proyecto. Se tiene pensado también como **mejora a futuro.**

# CONCLUSIONES

Llevar a cabo este proyecto ha permitido practicar los conocimientos adquiridos a lo largo del ciclo formativo de Desarrollo de Aplicaciones Web, atendiendo así todas las fases de la realización del software, desde la planificación inicial hasta el despliegue final en un entorno real de producción en la nube.

Con la culminación de este proyecto se ha conseguido crear una aplicación web útil, bien estructurada y funcional, que responde a una necesidad real: la gestión de tareas, diarios y relaciones personales a través de una herramienta digital de fácil comprensión. La elección de tecnologías como Node.JS, React y TypeScript y los servicios de AWS también se ha demostrado acertada y útil, para que se pueda garantizar un desarrollo de la aplicación seguro, escalable y eficiente.

A lo largo del proceso se han abordado varios retos técnicos, como la implementación de la autenticación JWT, la gestión de tareas compartidas o la estructuración modular del backend, todos ellos alcanzados con éxito. También se ha priorizado la usabilidad y accesibilidad de la aplicación, desarrollando una interfaz clara y adaptable a diferentes dispositivos mediante un diseño responsive.

A nivel personal, este proyecto ha supuesto un crecimiento significativo en cuanto a las habilidades técnicas, metodológicas y organizativas. Asimismo, ha evidenciado la relevancia de la planificación, la validación continua, el control de versiones y la documentación.

En conclusión, **TASK-CRAFT** no solo satisface los objetivos marcados, sino que también constituye una base robusta en la que seguir construyendo las mejoras futuras, tanto en lo técnico como en la experiencia de usuario final.

# REFERENCIAS BIBLIOGRÁFICAS

Clear, J. (2020). Hábitos atómicos: cambios pequeños, resultados extraordinarios. adminAlex, & adminAlex. (2023, 13 marzo).

La gestión del tiempo y la productividad - Pro Optim Blog. Pro Optim Blog -. <https://blog.prooptim.com/mejorar-productividad/la-gestion-del-tiempo-y-la-productividad/>

Midudev. (s. f.). *GitHub - midudev/curso-node-js: Curso de Node.js desde cero*. GitHub. <https://github.com/midudev/curso-node-js>

Midudev. (s. f.). aprendiendo-react/projects/10-crud-redux at master · midudev/aprendiendo-react. GitHub. <https://github.com/midudev/aprendiendoreact/tree/master/projects/10-crud-redux>

# ANEXOS

**MANUAL DE USUARIO**

El presente manual contiene indicaciones para el uso adecuado de la aplicación. Las funcionalidades se organizan en secciones, e incluyen pasos a seguir para facilitar su uso.

**Registro e inicio de sesión**

* Para registrarse requiere:
  + Acceder a la pantalla principal.
  + Hacer clic en "Acceder".
  + Completar los siguientes campos:
  + Nombre de usuario
  + e-mail
  + Contraseña
  + Hacer clic en "crear una nueva cuenta".
* Para iniciar sesión:
  + Introducir el e-mail y la contraseña.
  + Pulsar "Acceder".
  + Si las credenciales son correctas se accede al panel principal.

**Gestión de diario**

* Para ver las entradas:
  + Acceder al menú lateral y seleccionar "Diario".
  + Aparece una lista con las entradas guardadas.
  + Para crear una nueva entrada:
* Pulsar "Añadir Diario".
  + Escribir el título y el contenido.
* Pulsar "guardar".
  + Para editar o eliminar una entrada:
  + Pulsar el botón de "editar" o "eliminar" al lado de la entrada deseada.

**Gestión de tareas**

* Para ver tus tareas:
  + Dirígete a la sección "Tareas" en el menú lateral
  + Revisa la lista con el estado de las tareas
  + Para crear una tarea nueva:
  + Pulsa “Nueva tarea”
  + Completa los siguientes campos:
  + Título
  + Descripción (opcional)
  + Fecha límite
  + Estado (pendiente, en curso, o completada)
  + Pulsa “Guardar”.
  + Actualizar una tarea:
* Para actualizar una tarea, utiliza los botones de “Editar” para posteriormente modificarla.
  + Cambia el estado según el avance.
  + Pulsa el botón de eliminar para borrarla.
* Para filtrar tareas (no implementado):
  + Utiliza los filtros según el estado para ver tareas concretas.

**Gestión de amigos (no implementado)**

* Para buscar amigos y añadirlos:
  + Ve a “Amigos”.
  + Introduce el nombre del usuario en la barra de búsqueda de amigos.
  + Pulsa “Enviar solicitud”.
* Gestionar las solicitudes:
  + Puedes aceptar, rechazar solicitudes pendientes.
  + Ver amigos:
  + En la lista de amigos, selecciona "Ver perfil" o "Eliminar amigo".

**Ajustes de cuenta**

* Modificar el perfil:
  + Accede a "Ajustes" y modifica tu nombre, correo electrónico y contraseña.
  + Pulsa “Guardar cambios”.
* Eliminar cuenta:
  + Ve a "Ajustes" y pulsa “Eliminar cuenta”. Confirma la acción cuando aparezca.

**Panel de administración (solamente para administradores de la plataforma)**

* Acceder a la lista de usuarios registrados:
  + Acceder al “Panel de administración”.
    - Visualizar la lista de usuarios, que incluye:
    - ID de usuario
    - Nombre del usuario
    - Rol que tiene asignado
    - Acciones posibles:
  + Cambiar el rol de un usuario
  + Eliminar cuentas de usuario

**Navegación general**

* El menú lateral permite navegar entre las distintas secciones: Diario, Tareas, Amigos, Ajustes.
* Los botones permiten identificar de manera clara el texto en cada uno de los, al menos, dos botones de los formularios.
* Los formularios validan los campos de los formularios antes de lanzar el formulario.
* Los formularios lanzan mensajes de error o confirmación para guiar a los usuarios.

**LOGO DE LA APLICACIÓN:**

