

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

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

**CONVOCATORIA:** **ORDINARIA**

**Título del proyecto: TASK-CRAFT**

**PROFESOR TUTOR: Nombre y Apellidos**

**Alumnos:**

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

**MODALIDAD DE PROYECTO: Proyectos de gestión**

ÍNDICE

[1. JUSTIFICACIÓN 3](#_Toc192890108)

[2. INTRODUCCIÓN 3](#_Toc192890109)

[3. TOMA DE REQUISITOS 3](#_Toc192890110)

[4. OBJETIVOS 3](#_Toc192890111)

[5. DESARROLLO 3](#_Toc192890112)

[5.1. METODOLOGÍA 3](#_Toc192890113)

[5.2. TECNOLOGÍAS 3](#_Toc192890114)

[5.3. PROTOTIPADO Y USABILIDAD 3](#_Toc192890115)

[5.4. MODELADO DE DATOS 3](#_Toc192890116)

[5.5. SEGURIDAD Y AUTENTICACIÓN 3](#_Toc192890117)

[5.6. DISEÑO E IMPLEMENTACIÓN 3](#_Toc192890118)

[5.7. PRUEBAS Y CALIDAD DEL SOFTWARE 3](#_Toc192890119)

[6. DESPLIEGUE DEL PRODUCTO 3](#_Toc192890120)

[7. PRESUPUESTO, RIESGOS, VIABILIDAD 4](#_Toc192890121)

[8. DIFICULTADES ENCONTRADAS Y RESOLUCIÓN 4](#_Toc192890122)

[9. CONCLUSIONES 4](#_Toc192890123)

[10. REFERENCIAS BIBLIOGRÁFICAS 4](#_Toc192890124)

[11. ANEXOS 4](#_Toc192890125)

# 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.

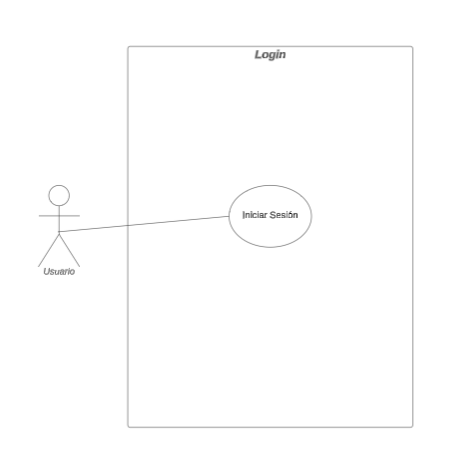
5. **Gestión de usuario:** el usuario podrá **cambiar sus credenciales, cerrar sesión o eliminar** su cuenta.

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

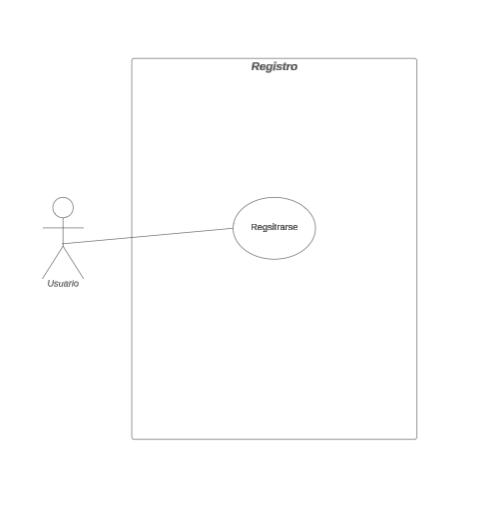
7. **Seguridad:** la aplicación tendrá las **contraseñas encriptadas** y las sesiones estarán protegidas con **JWT**, guardadas en una **cookie**.

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

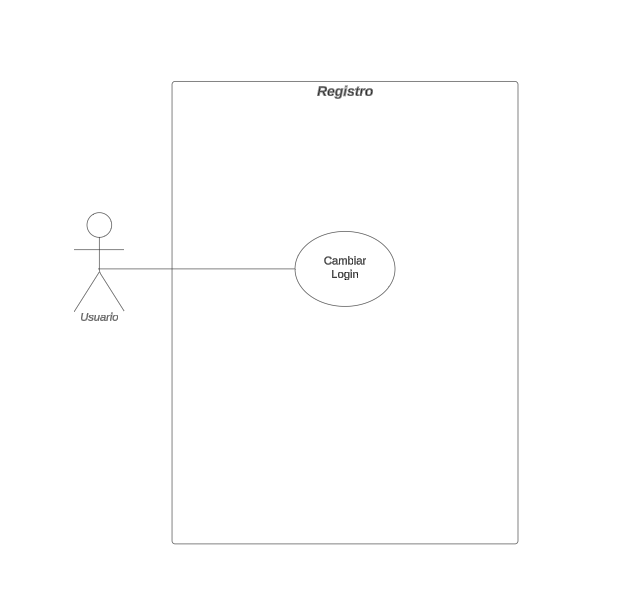
**CASOS DE USO**

****

|  |  |
| --- | --- |
| **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 |

# 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

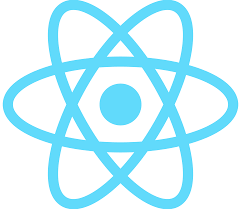
## 5.1. METODOLOGÍA

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.

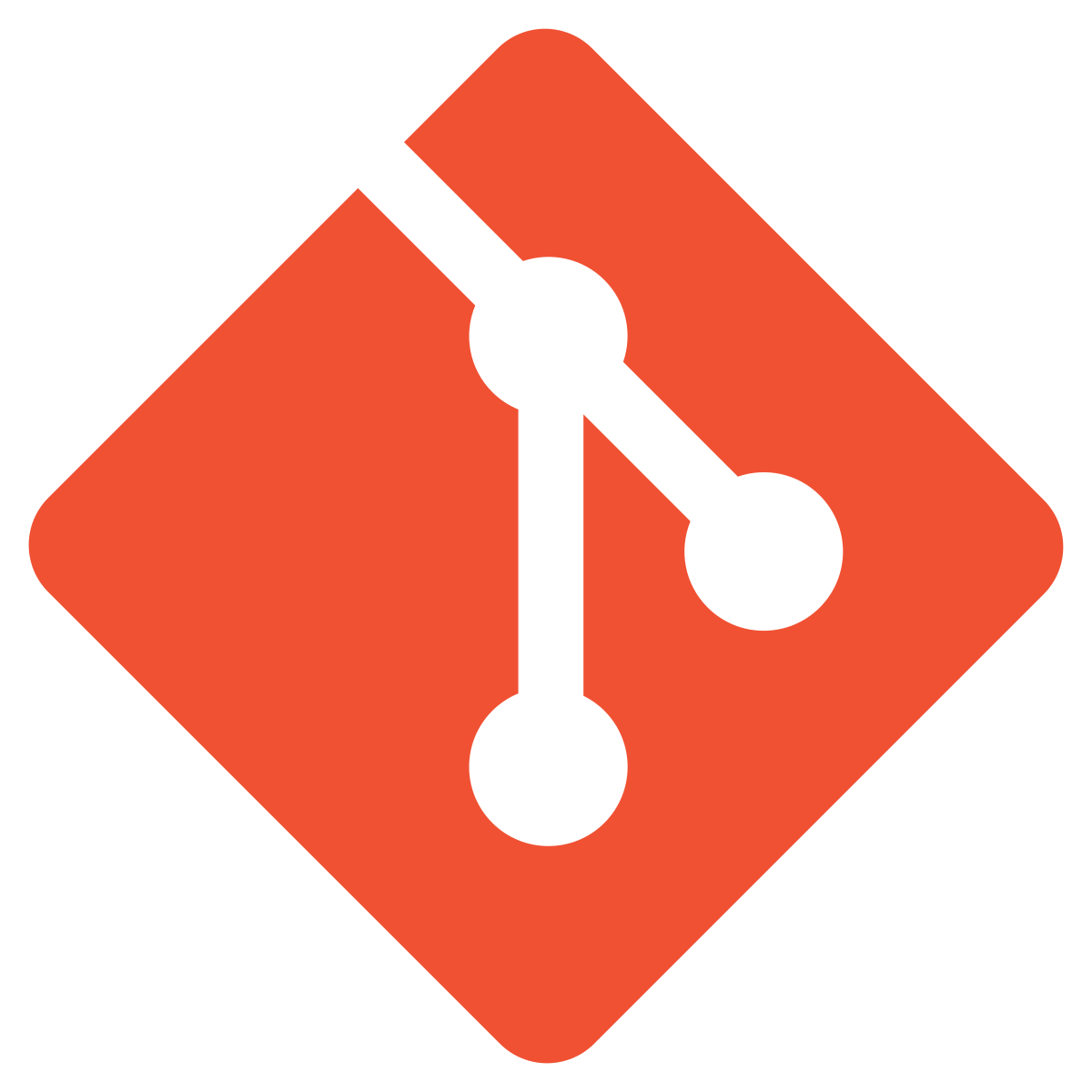
## 5.2. TECNOLOGÍAS

**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.

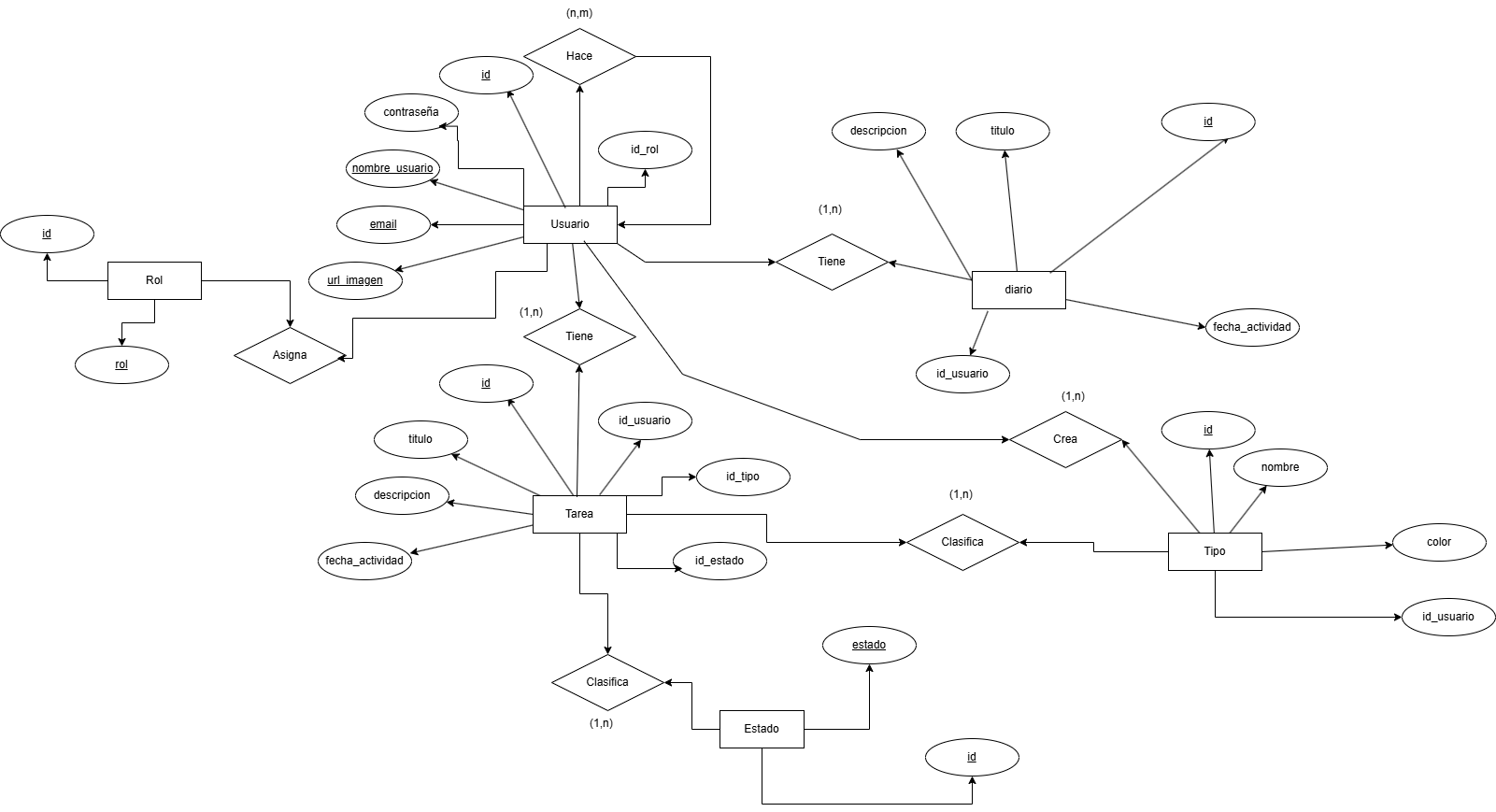
**TypeScript:** Se prefiere sobre JavaScript en el frontend por su tipado estático, lo que brinda mayor seguridad y claridad en el código, especialmente para desarrolladores acostumbrados a lenguajes tipados como Java. Esto contribuye a reducir la probabilidad de errores y a mejorar la mantenibilidad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

## 5.4. MODELADO DE DATOS

**Modelo Entidad Relación:**



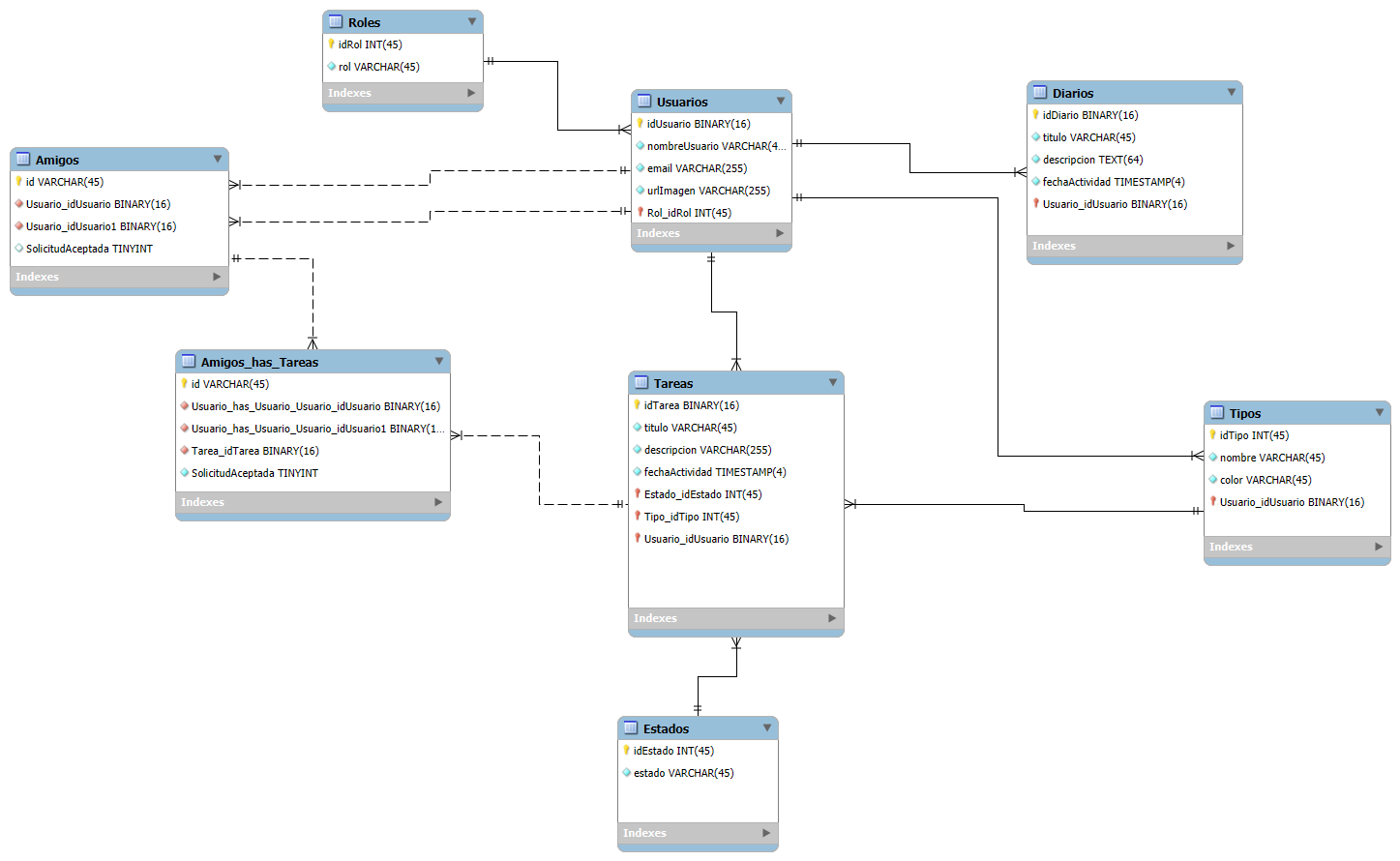
**Entidades:**

* 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

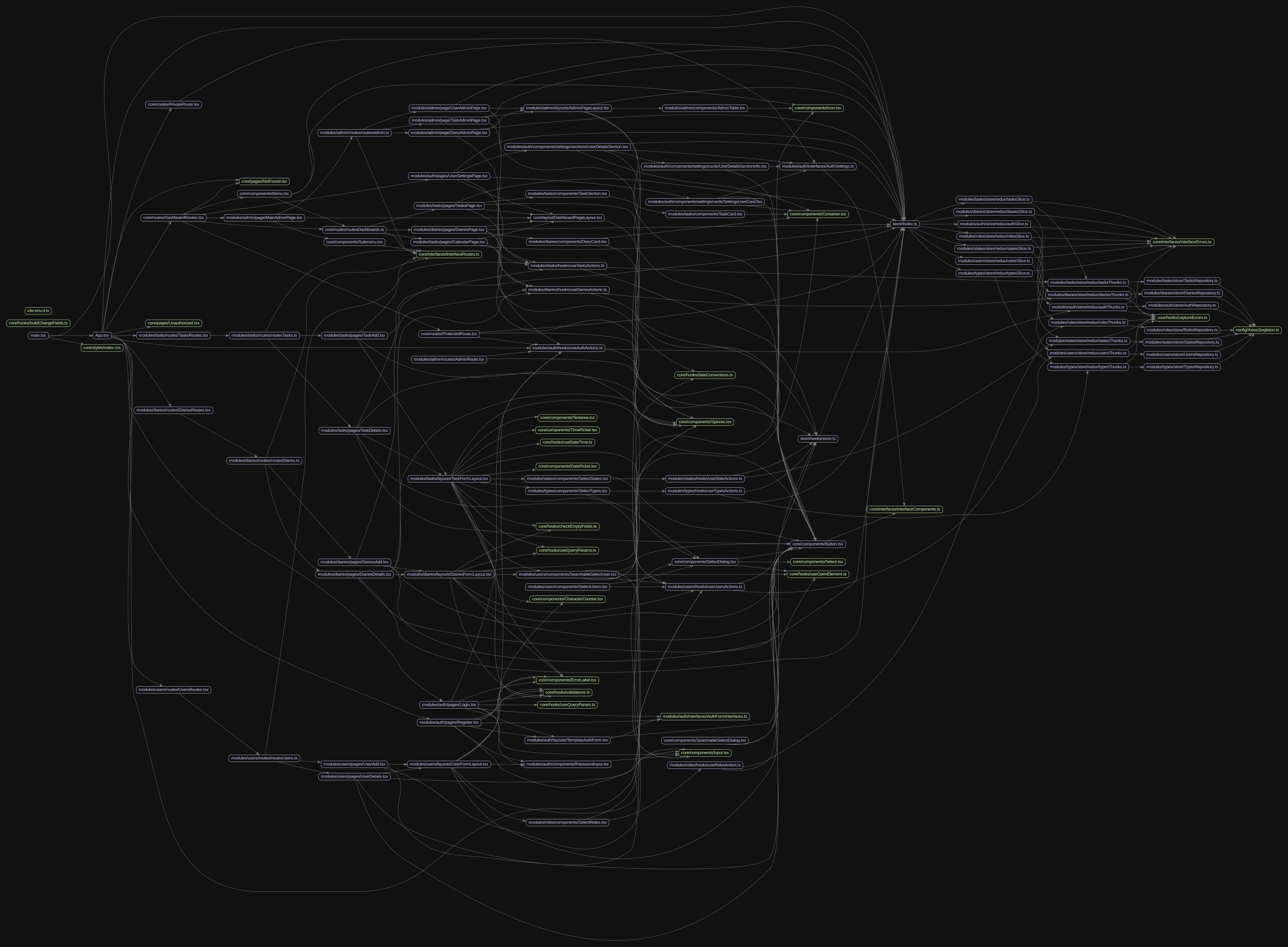
Para la seguridad de la aplicación se ha implementado una librería de **Node.js** para hashear contraseñas llamada **bcrypt,** el cual aplica un algoritmo unidireccional, lo que significa que no se puede revertir la contraseña original. Esta contraseña se guardará encriptada en la base de datos. Para 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

# PRESUPUESTO, RIESGOS, VIABILIDAD

PRESUPUESTO

Horas de trabajo estimadas

1. Diseño

• Diseño deInterfaz de Usuario(UI) y Experiencia de Usuario(UX):25h

2. Desarrollo

• Desarrollo Frontend (Interfaz del Usuario):**25h**

• Desarrollo Backend(Servidor y BasedeDatos):**15h**

• Integración y Sincronización de Datos: **15h**

3. Pruebas y Calidad

• Pruebas deFuncionalidad: **30h**

• Pruebas deCompatibilidady Rendimiento: **20h**

• CorreccióndeErrores y Optimización: **30h**

**Presupuesto Total:**

• 200hx 14,5€/h= **2288€**

# 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.

# CONCLUSIONES

# 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