TYME

Equipo de trabajo:

* Cristian Urqueta (Líder)
* Juan Contreras (Desarrollador)
* Esteban Escudero (Desarrollador)

Fecha de inicio: 26/03

Fecha Estimada de Finalización: 27/06

1. Descripción del problema:

Los dueños del gimnasio TYME han identificado la necesidad de contar con un sistema de gestión integral que abarque desde la administración de usuarios (usuarios, profesores y administradores) hasta el control de clases, rutinas personalizadas, asistencia y comunicación interna. Actualmente, los procesos se realizan de manera manual o con sistemas fragmentados, lo que genera ineficiencias y dificulta el monitoreo en tiempo real de las operaciones del gimnasio.

**Necesidades del Cliente:**

* **Administración Eficiente:** Registro y gestión de perfiles de usuarios, profesores y administradores, con diferentes niveles de acceso.
* **Gestión de Clases y Horarios:** Creación y seguimiento de clases con control de cupos, inscripción de usuarios y alertas cuando queden pocos cupos.
* **Rutinas Personalizadas:** Posibilidad para que entrenadores creen y actualicen rutinas de entrenamiento adaptadas a cada cliente.
* **Control de Asistencia:** Registro de asistencia a clases mediante métodos manuales (actualmente sin el uso de códigos QR) y generación de estadísticas.
* **Pagos y Membresías:** Registro de pagos, envío de recordatorios y generación de reportes sobre el estado de las mensualidades.
* **Comunicación Interna:** Implementar un módulo de mensajes (o muro de mensajes) que facilite la comunicación entre los usuarios, con notificaciones de nuevos mensajes.

**Problema central:**

La falta de un sistema centralizado y automatizado en TYME dificulta la gestión integral de las operaciones del gimnasio, lo que se traduce en ineficiencias administrativas, problemas en la coordinación de clases y dificultades para mantener una comunicación efectiva entre usuarios. El proyecto TYME busca resolver estas problemáticas mediante el desarrollo de una aplicación que integre todos estos procesos en una única plataforma.

1. Objetivo del Proyecto:

Desarrollar una aplicación web integral para el gimnasio TYME que permita gestionar de manera centralizada los usuarios, clases, rutinas, asistencia, pagos y comunicación, optimizando los procesos administrativos y mejorando la experiencia de los usuarios.

Los objetivos específicos de este proyecto son:

* **Gestión de Usuarios:** Implementar módulos para el registro, inicio de sesión, recuperación de contraseña y administración de perfiles, con diferenciación de roles (usuarios, profesores y administradores).
* **Clases y Horarios:** Permitir a los profesores crear clases con horarios definidos y cupos limitados, así como habilitar la inscripción de usuarios y el envío de notificaciones cuando se acerque el límite de cupos.
* **Rutinas Personalizadas:** Facilitar a los entrenadores la creación y actualización de rutinas de entrenamiento específicas para cada usuario.
* **Control de Asistencia:** Registrar manualmente la asistencia a las clases y generar estadísticas relacionadas con la participación y puntualidad de los usuarios.
* **Pagos y Membresías:** Registrar los pagos de mensualidades y membresías, enviar recordatorios oportunos y generar reportes de pagos realizados y pendientes.
* **Comunicación Interna:** Crear un módulo de mensajes o muro de publicaciones para facilitar la interacción y comunicación entre los usuarios del sistema.

1. ***Requisitos del sistema:***

Estos requisitos describen las funcionalidades que el sistema TYME debe ofrecer para cumplir con las necesidades del gimnasio:

**Administración de Usuarios:**

* **Registro e inicio de sesión:** El sistema debe permitir el registro de nuevos usuarios (miembros, profesores y administradores) y ofrecer un mecanismo seguro de autenticación.
* **Recuperación de contraseña:** Se debe implementar una funcionalidad para la recuperación y cambio de contraseñas en caso de olvido.
* **Gestión de perfiles:** Cada usuario tendrá un perfil personalizado donde se almacene información básica (nombre, correo electrónico, rol, etc.), y se podrán modificar dichos datos según corresponda.
* **Control de accesos:** El sistema diferenciará niveles de permisos, asegurando que los administradores cuenten con privilegios superiores respecto a profesores y usuarios.

**Gestión de Clases y Horarios:**

* **Creación y administración de clases:** Los profesores podrán definir clases, asignar horarios y establecer un cupo máximo de participantes. Además, podrán saber la cantidad de personas inscritas para la clase.
* **Inscripción de usuarios:** Los usuarios tendrán la opción de inscribirse en las clases, recibiendo confirmaciones automáticas de su registro. Esto se verá afectado en caso de que el cliente no haya realizado la renovación de su plan o este mismo se encuentre vencido, con la no inscripción del ramo a causa de un pago pendiente.
* **Alertas de cupos:** Se enviarán notificaciones cuando el número de cupos disponibles en una clase se reduzca a un nivel crítico. Una vez la clase se encuentre en su máxima capacidad, no se aceptarán más inscripciones a dicha clase.
* **Calendario interactivo:** Disponibilidad de una vista de calendario que muestre de manera clara todas las clases programadas, esto con el fin que los clientes estén al tanto de futuras clases, su fecha de realización y la hora en la cual se impartirá. También se podrán recibir notificaciones cuando se agregue una nueva clase.

**Rutinas Personalizadas:**

* **Asignación y actualización de rutinas:** Los entrenadores podrán crear rutinas de entrenamiento individualizadas para cada usuario y modificarlas en función de sus necesidades. Esto busca facilitar y agilizar la realización de rutinas personales de cada cliente a causa del gasto de tiempo que significa la creación de una rutina nueva para cada persona.

**Control de Asistencia:**

* **Registro manual de asistencia:** Los profesores podrán registrar manualmente la asistencia a cada clase como también ver las personas inscritas a dicha clase. Esto con el fin de evitar inconvenientes entre la cantidad de personas inscritas y el tamaño del espacio donde se realizará la clase.
* **Generación de estadísticas:** El sistema compilará información sobre la asistencia y puntualidad, facilitando la toma de decisiones y análisis de desempeño.

**Gestión de Pagos y Membresías:**

* **Registro de pagos:** Se dispondrá de un módulo para el registro manual de los pagos de mensualidades y membresías.
* **Recordatorios de pago:** El sistema enviará notificaciones anticipadas cuando se acerque la fecha de vencimiento de un pago.
* **Reportes financieros:** Los administradores podrán acceder a reportes detallados que indiquen los pagos realizados y los pendientes.

**Comunicación Interna:**

* **Módulo de mensajes:** Implementación de un muro de mensajes o sistema de mensajería interna para facilitar la comunicación entre usuarios.
* **Notificaciones de mensajes:** Los usuarios recibirán alertas al ingresar un nuevo mensaje o comentario.

**Requisitos No Funcionales:**

Estos requisitos garantizan la calidad del sistema en términos de rendimiento, seguridad y usabilidad:

* **Rendimiento:** El sistema debe responder de manera ágil y mantener una experiencia fluida en distintos dispositivos (móviles, tabletas y computadoras).
* **Seguridad:** Implementación de mecanismos de autenticación robustos y protocolos de encriptación para proteger la información sensible de los usuarios.
* **Usabilidad:** Interfaz intuitiva y fácil de navegar, asegurando que usuarios con distintos niveles de experiencia tecnológica puedan interactuar sin complicaciones.
* **Mantenibilidad y Escalabilidad:** Diseño modular y documentación adecuada que facilite futuras actualizaciones, integraciones y el mantenimiento continuo del sistema.

**Priorización de Requisitos:**

Para orientar el desarrollo y la asignación de recursos, se ha clasificado la prioridad de los requisitos de la siguiente manera:

|  |  |
| --- | --- |
| *Funcionalidades* | *Prioridad* |
| Administración de Usuarios | Alta |
| Gestión de Clases y Horarios | Alta |
| Control de Asistencias | Media |
| Registros de Pagos | Alta |
| Rutinas personalizadas | Alta |
| Comunicación Interna | Baja |

***5. Stakeholders y Usuarios***

**Stakeholders:**Se identifican las partes interesadas en el proyecto, quienes influirán en los requerimientos y en el éxito del sistema TYME:

* ***Dueños del Gimnasio TYME:***
  + Intereses: Optimizar la gestión administrativa y operativa del gimnasio, mejorar la experiencia del usuario y centralizar procesos.
  + Involucramiento: Solicitan el desarrollo del sistema y serán los principales receptores de los reportes y análisis generados.
* ***Equipo Administrativo:***
  + Intereses: Monitorear en tiempo real las operaciones del gimnasio, gestionar la base de usuarios y obtener reportes financieros y de asistencia.
  + Involucramiento: Utilizarán el sistema para la toma de decisiones y la supervisión de los procesos internos.
* ***Profesores y Entrenadores:***
  + Intereses: Facilitar la creación y administración de clases, gestionar las rutinas de los usuarios y registrar la asistencia de manera eficiente.
  + Involucramiento: Interactuarán directamente con los módulos de clases, rutinas y asistencia.
* ***Usuario / Clientes:***
  + Intereses: Acceder de forma sencilla a información sobre clases, inscribirse a entrenamientos, consultar sus rutinas, verificar sus pagos y recibir notificaciones relevantes.
  + Involucramiento: Serán los usuarios finales que harán uso de la interfaz de inscripción y seguimiento de sus actividades. En este contexto, los usuario son también considerados clientes, ya que son quienes adquieren los servicios del gimnasio y, por tanto, deben disfrutar de una experiencia de usuario óptima y un servicio de calidad.
* ***Equipo de Desarrollo (Universitarios):***
  + Intereses: Desarrollar un sistema funcional y robusto que cumpla con los requerimientos planteados, obteniendo experiencia en el desarrollo de proyectos para clientes externos a la universidad, aplicando metodologías ágiles (Scrum) y técnicas de ingeniería de software aprendidas.
  + Involucramiento: Responsables del diseño, implementación, pruebas y mantenimiento del sistema TYME.

Perfiles de Usuario:

Se identifican los perfiles principales que interactuarán con el sistema:

* ***Administradores:***
  + Responsabilidades: Gestión completa del sistema, incluyendo la administración de usuarios, supervisión de clases, manejo de pagos y generación de reportes.
  + Permisos: Acceso total a todas las funcionalidades y datos, con capacidad para realizar modificaciones de alto nivel.
* ***Profesores/Entrenadores:***
  + Responsabilidades: Creación y administración de clases, asignación y modificación de rutinas, y registro de asistencia.
  + Permisos: Acceso a funcionalidades operativas específicas, sin autorización para cambiar datos administrativos globales.
* ***Usuario / Clientes:***
  + Responsabilidades: Inscripción en clases, consulta de rutinas personalizadas, verificación de pagos y recepción de notificaciones.
  + Permisos: Acceso restringido a funcionalidades de consulta e inscripción, sin capacidad de alterar información crítica del sistema.
    1. ***Arquitectura del Sistema***

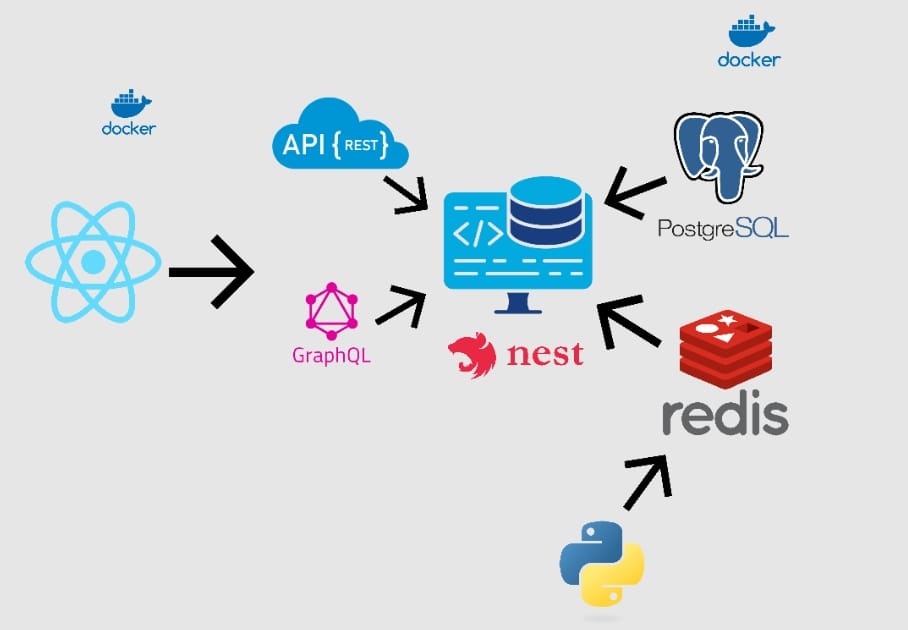


Ilustración 1: Diagrama frameworks

Este sistema está compuesto por varias tecnologías organizadas en Frontend, Backend, Base de Datos, Caché y Servicios Adicionales.

Tecnologías por Categoría:

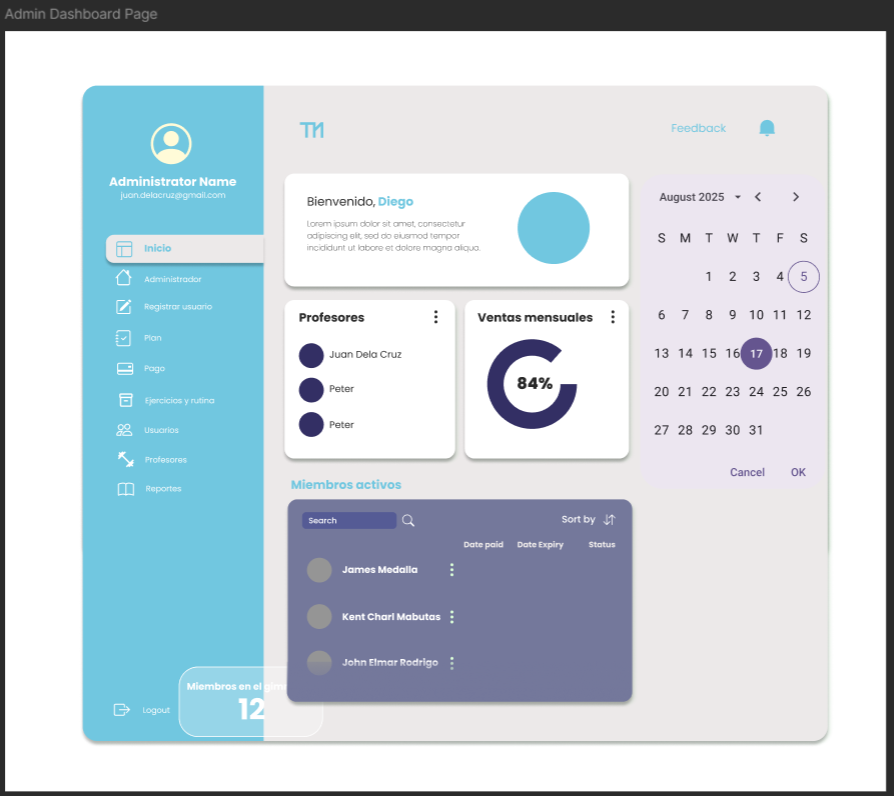
1. Frontend (Interfaz de Usuario):
   1. React.js → Para construir una interfaz de usuario dinámica y modular.
   2. GraphQL → Para optimizar la consulta y manipulación de datos desde el frontend al backend.
2. Backend (Servidor de Aplicaciones):
   1. NestJS (Node.js) → Framework basado en TypeScript para la gestión del backend.
   2. API REST → Para manejar solicitudes HTTP tradicionales.
   3. GraphQL → Para optimizar consultas de datos en el backend.
3. Base de Datos:
   1. PostgreSQL → Base de datos relacional para el almacenamiento estructurado de datos.
4. Caché y procesamiento adicional:
   1. Redis → Para almacenamiento en caché y mejora de rendimiento.
   2. Python → Para procesamiento de datos y posibles integraciones con Machine Learning o análisis avanzado.
5. Infraestructura y Contenedores:
   1. Docker → Para la contenedorización y despliegue eficiente de la aplicación.

***7. Diseño de interfaz de usuario***



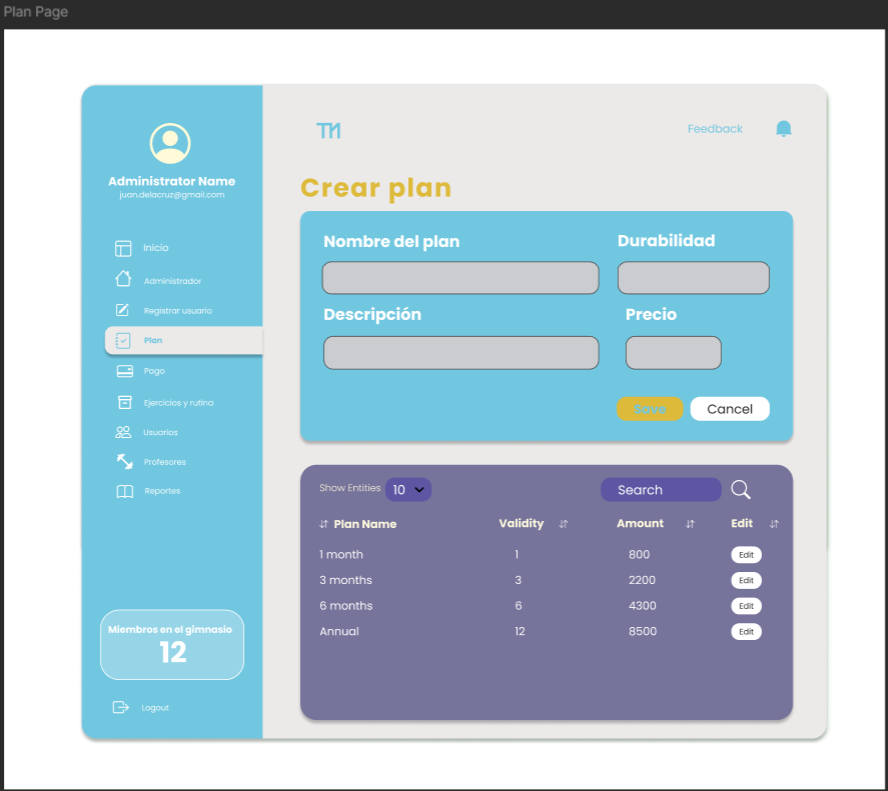
Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.



Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.



Interfaz de usuario gráfica, Texto, Aplicación

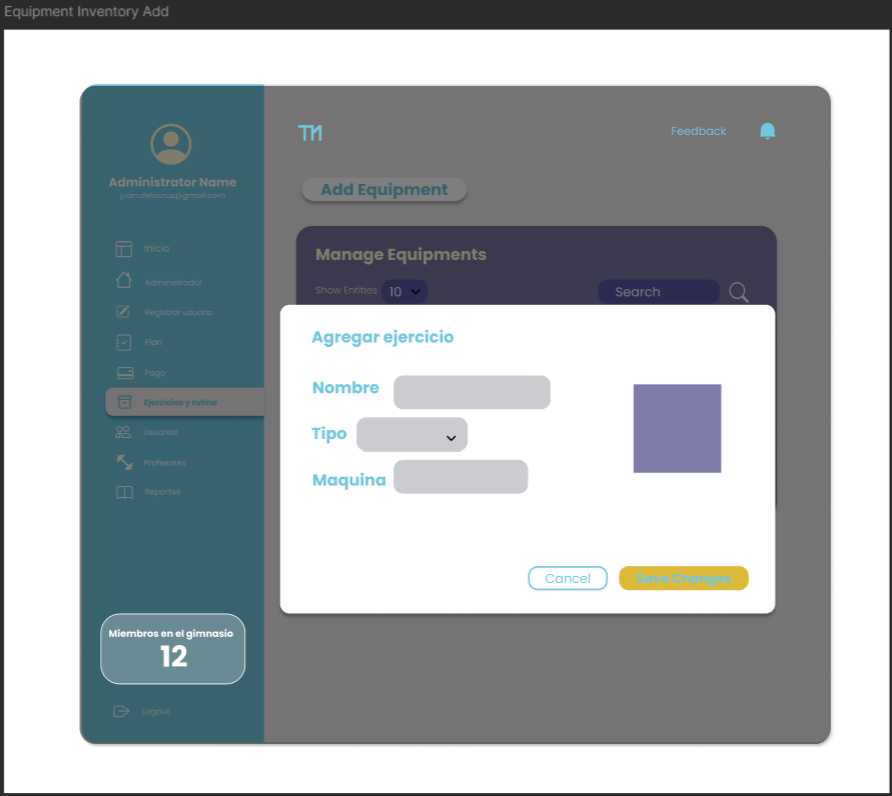
El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.



Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

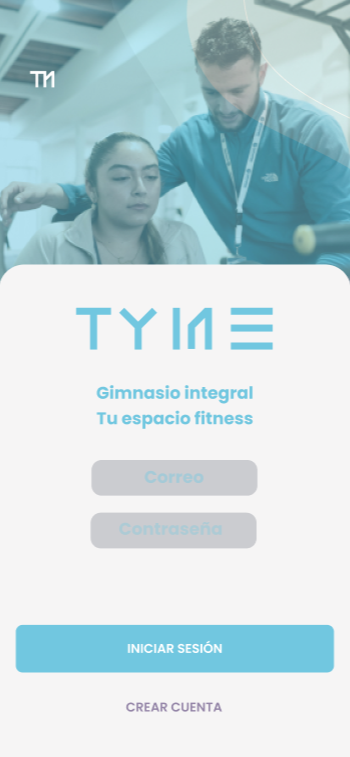
El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

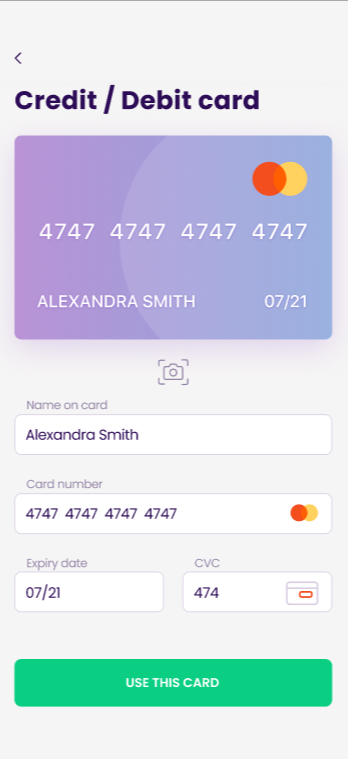
Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

**Flujo de Navegación del Sistema**

El sistema sigue un flujo lógico de navegación entre las diferentes pantallas para garantizar una experiencia intuitiva y funcional. A continuación, se describe el flujo de navegación basado en el diseño:

**1. Pantalla de Inicio**

* Primera pantalla que el usuario ve al abrir la aplicación.
* Opciones disponibles:
  + **Iniciar Sesión** → Redirige a la pantalla principal después de la validación de credenciales.
  + **Crear Cuenta** → Redirige a un formulario de registro para nuevos usuarios.

**2. Pantalla Principal**

* Muestra las diferentes secciones disponibles en el sistema.
* Opciones de navegación:
  + **Rutina** → Redirige a la pantalla de Rutina.
  + **Clases** → Redirige a una pantalla con detalles sobre clases disponibles.
  + **Calendario y Clases** → Muestra un calendario con clases y eventos programados.
  + **Perfil** → Redirige a la pantalla de perfil del usuario.

**3. Pantalla de Rutina**

* Muestra la rutina de ejercicios del usuario con detalles de cada ejercicio.
* Opciones disponibles:
  + **Buscar un ejercicio** mediante el campo de búsqueda.
  + **Seleccionar Día** para visualizar los ejercicios correspondientes.
  + **Regresar a Categorías** con el botón de navegación superior.

**4. Pantalla de Perfil**

* Muestra la información personal del usuario.
* Opciones disponibles:
  + **Modificar perfil** → Permite actualizar datos personales.
  + **Cerrar sesión** → Regresa a la pantalla de inicio.

**5. Pantalla de Pago**

* Permite la gestión de pagos mediante tarjeta de crédito o débito.
* Opciones disponibles:
  + **Ingresar o modificar datos de la tarjeta**.
  + **Confirmar el pago**.
  + **Regresar a Categorías** en caso de cancelar la operación.

***8. Plan de trabajo***

Fases del proyecto:

**Fase de Investigación y Análisis.**

En esta etapa se investigarán las tecnologías a utilizar, incluyendo React para el frontend, Nest.js para el backend, PostgreSQL como base de datos. Se revisarán tutoriales, documentación oficial y ejemplos de implementación para comprender su funcionamiento.

* Revisión de documentación y recursos de aprendizaje.
* Definición de los requisitos del sistema y objetivos del proyecto.

**Fase de Configuración del Entorno de Desarrollo**

* Instalación y configuración de Docker para estandarizar los entornos de desarrollo.
* Creación de contenedores para cada componente del sistema.
* Establecimiento de flujos de trabajo.

**Fase de Diseño**

* Diseño de la estructura de la base de datos en PostgreSQL.
* Definición de la estructura del backend y los endpoints de la API.
* Diseño de prototipos de la interfaz de usuario utilizando herramientas como Figma.

**Fase de Implementación**

Durante esta fase, se trabajará en el desarrollo del sistema, aplicando los conocimientos adquiridos en la fase de investigación y aprendiendo sobre la marcha.

* Desarrollo del frontend.
* Implementación del backend.
* Integración de la base de datos PostgreSQL.
* Implementación de autenticación y control de acceso.

**Fase de Pruebas y Corrección de Errores**

* Ejecución de pruebas unitarias e integración.
* Depuración de errores y mejoras en el código.
* Evaluación del rendimiento de la base de datos y el backend.

Cronograma detallado con las actividades y fechas estimadas para cada fase:



Asignación de Tareas: Distribución de tareas entre los miembros del equipo.

La asignación de tareas se llevará a cabo mediante herramientas de gestión de proyectos como **Notion** o **Trello**, lo que permitirá una organización eficiente del trabajo en equipo. Estas plataformas facilitarán la asignación de responsabilidades, el seguimiento del progreso y la colaboración en tiempo real, asegurando que cada miembro del equipo tenga claridad sobre sus tareas y plazos.

***Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.***Se adjunta calendarización de las actividades realizadas y por realizar:

https://trello.com/b/lGkPO5aY/proyecto-integrador-de-software

***9.*** ***Planificación del Proyecto: Fases, Tiempos y Recursos***

**Distribución por Fases**

A continuación, se muestro la distribución estimada en una tabla, considerando complejidad, dependencias y lógica secuencial entre fases:



**Calendarización Semana a Semana**



***10. Gestión de Riesgos***

Identificación de Riesgos:

* ***Retrasos en el Cronograma:*** Dada la carga académica y el tiempo limitado, existe el riesgo de no cumplir con las fechas establecidas.
* ***Fallas Técnicas o de Integración:*** Problemas en la integración de módulos (por ejemplo, entre la interfaz y el backend) o dificultades técnicas al implementar funcionalidades.
* ***Deficiencias en la Comunicación:*** Inadecuada coordinación entre los miembros del equipo que podría afectar la calidad y la coherencia del proyecto.
* ***Cambios en los Requerimientos:*** Solicitudes adicionales o modificaciones de requisitos por parte de los dueños del gimnasio durante el desarrollo.
* ***Limitaciones de Conocimiento:*** La posible carencia de experiencia en alguna tecnología o metodología puede afectar el desempeño del proyecto.

Plan de Mitigación:

* ***Planificación y Seguimiento:*** Establecer un cronograma detallado con hitos y reuniones periódicas (sprints) para evaluar avances y ajustar tiempos.
* ***Capacitación y Soporte:*** Aprovechar recursos de aprendizaje (tutoriales, documentación oficial) y solicitar asesoría en áreas de menor dominio.
* ***Comunicación Efectiva:*** Utilizar herramientas colaborativas y reuniones regulares para asegurar que todos los miembros estén alineados con los objetivos y cambios del proyecto.
* ***Flexibilidad en Requerimientos***: Establecer un proceso de gestión de cambios que permita evaluar el impacto de nuevas solicitudes y ajustar el alcance en función de la prioridad.
* ***Pruebas y Validación Continua:*** Realizar pruebas frecuentes de integración y funcionalidad para detectar y corregir fallas de manera temprana.

***11. Criterios de Aceptación***

**Criterios Generales para la Aceptación del Proyecto:**

**Cumplimiento Funcional:**

* Todas las funcionalidades definidas (administración de usuarios, clases, rutinas, asistencias, pagos y comunicación) deben estar implementadas y operativas según las especificaciones.
* La aplicación debe permitir la diferenciación de roles (administrador, profesor/entrenador y usuario/cliente) con los respectivos permisos.

**Rendimiento y Usabilidad:**

* La interfaz debe ser intuitiva y adaptarse correctamente a distintos dispositivos (implementación de diseño responsive/PWA).
* El tiempo de respuesta de la aplicación debe ser adecuado, garantizando una experiencia fluida para el usuario.

**Seguridad y Confiabilidad:**

* Implementación de mecanismos de autenticación y autorización robustos.
* Protección de la información sensible mediante encriptación y protocolos de seguridad.

**Integración y Estabilidad:**

* Los diferentes módulos (gestión de usuarios, clases, rutinas, asistencia, pagos y comunicación) deben integrarse sin generar errores críticos.
* La aplicación debe operar de forma estable durante un periodo prolongado sin caídas o fallas significativas.

**Documentación y Soporte:**

* Disponibilidad de documentación técnica que respalde la implementación y facilite el mantenimiento futuro.
* Informe de pruebas que demuestre la funcionalidad y resolución de incidencias durante el proceso de desarrollo.

***12. Evidencias Técnicas***

**Documentación de Requisitos:**

**Requisitos Funcionales:**

1. **Módulo de Usuarios:**
   * Registro de usuarios (clientes, entrenadores).
   * Inicio de sesión con autenticación básica.
   * Gestión básica de usuarios (ver detalles, actualizar datos).
2. **Módulo de Clases:**
   * Creación de clases (definición de nombre, horario, entrenador).
   * Inscripción de clientes a clases disponibles.
   * Listado de clases disponibles.
3. **Módulo de Rutinas:**
   * Creación de rutinas de ejercicio.
   * Visualización de rutinas asignadas a los clientes.
4. **Módulo de Asistencias:**
   * Registro manual de asistencia a clases por parte de los clientes.
5. **Módulo de Pagos:**
   * Registro manual de pagos de clientes por membresías o servicios adicionales.
6. **Reportes:**
   * Estadísticas de asistencia a clases (por día, por clase).
   * Estadísticas de pagos.
   * Reporte de cuántas personas ingresaron al gimnasio en un día determinado.
   * Registro de asistencia a clases por hora.

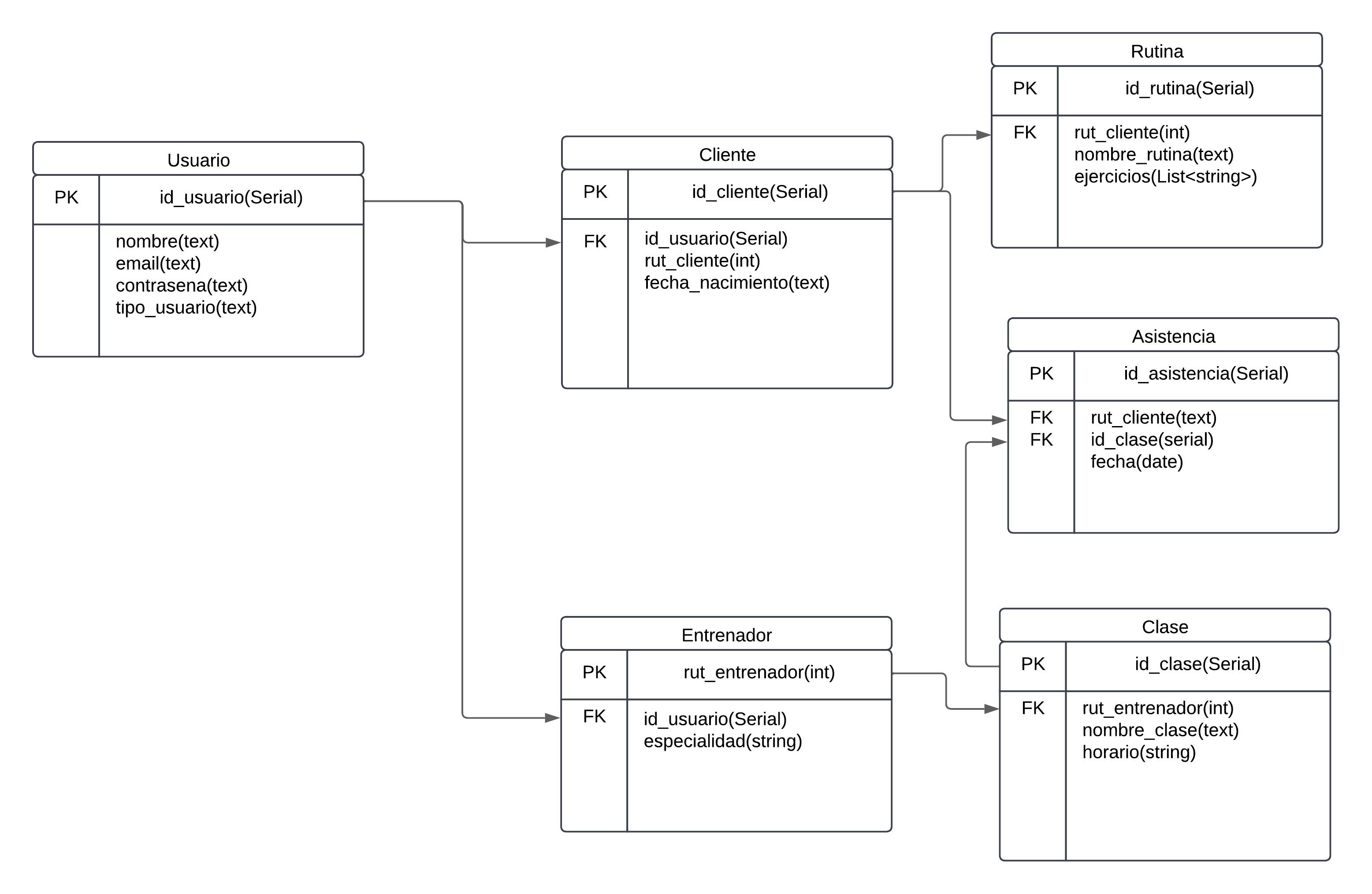
**Requisitos No Funcionales:**

1. **Rendimiento:**
   * El sistema debe ser capaz de manejar hasta 500 usuarios concurrentes sin degradación significativa en el rendimiento.
2. **Seguridad:**
   * Protección de datos personales de los usuarios mediante encriptación y medidas de seguridad estándar.
3. **Usabilidad:**
   * Interfaz intuitiva y accesible tanto para clientes como entrenadores.
4. **Escalabilidad:**
   * El sistema debe ser capaz de ser ampliado para soportar más usuarios y funcionalidades a futuro.

Diagramas UML:

Diagrama

El contenido generado por IA puede ser incorrecto.



Prototipos Iniciales:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

***13. Justificación de decisiones técnicas***

**Selección de Tecnologías**

1. **Frontend (Interfaz de Usuario)**
   * **React.js**
     + **Razón de elección**: **React.js** es un framework de JavaScript altamente eficiente y flexible para la creación de interfaces de usuario interactivas y dinámicas. Se seleccionó debido a su capacidad para construir aplicaciones modulares y reutilizables, lo cual facilita el desarrollo y mantenimiento a largo plazo del sistema. Además, React es ampliamente utilizado y soportado por una gran comunidad, lo que asegura disponibilidad de recursos, tutoriales y paquetes adicionales.
   * **GraphQL**
     + **Razón de elección**: **GraphQL** se eligió para optimizar la consulta de datos entre el frontend y el backend. A diferencia de REST, GraphQL permite a los clientes solicitar solo los datos que necesitan, lo que mejora el rendimiento y reduce el uso de ancho de banda. Esto es especialmente útil en sistemas donde la estructura de los datos es compleja y puede cambiar a lo largo del tiempo, como en el caso de **TYME**, donde se gestionan datos de usuarios, clases, rutinas, pagos y asistencia.
2. **Backend (Servidor de Aplicaciones)**
   * **NestJS (Node.js)**
     + **Razón de elección**: **NestJS** es un framework robusto y escalable para construir aplicaciones del lado del servidor con **Node.js**. Está basado en **TypeScript**, lo que proporciona una estructura clara y segura en el desarrollo del backend. **NestJS** utiliza principios de programación modular, lo que permite una fácil extensión y mantenimiento del sistema a medida que se agregan nuevas funcionalidades. Además, su integración con otras tecnologías como **GraphQL** lo hace ideal para la gestión eficiente de datos en aplicaciones modernas.
   * **API REST y GraphQL**
     + **Razón de elección**: Se opta por una combinación de **API REST** y **GraphQL** para ofrecer flexibilidad. **API REST** se utiliza para manejar las solicitudes tradicionales HTTP, mientras que **GraphQL** se emplea para mejorar la eficiencia en las consultas de datos complejos. Esta combinación permite a los desarrolladores aprovechar lo mejor de ambos enfoques, dependiendo del tipo de interacción y consulta que se necesite realizar.
3. **Base de Datos**
   * **PostgreSQL**
     + **Razón de elección**: **PostgreSQL** es una base de datos relacional de código abierto conocida por su robustez, escalabilidad y fiabilidad. Se eligió debido a su capacidad para manejar transacciones complejas y su soporte para consultas SQL avanzadas. En el caso de **TYME**, donde se gestionan múltiples entidades relacionadas (usuarios, pagos, clases, rutinas, etc.), **PostgreSQL** es la opción ideal para garantizar la integridad y consistencia de los datos. Además, su capacidad para escalar y soportar grandes volúmenes de datos lo convierte en una opción a largo plazo.
4. **Caché y Procesamiento Adicional**
   * **Redis**
     + **Razón de elección**: **Redis** es una base de datos en memoria que se utiliza como sistema de caché. Se eligió para mejorar el rendimiento del sistema, reduciendo la carga en la base de datos principal al almacenar en caché los resultados de consultas frecuentes y operaciones de lectura intensiva. Esto mejora la velocidad de respuesta de la aplicación, lo que es crucial para la experiencia del usuario. En el caso de **TYME**, Redis se utilizará para almacenar datos temporales, como información de sesiones de usuarios y consultas repetidas sobre clases o rutinas.
   * **Python**
     + **Razón de elección**: **Python** se elige para procesamiento adicional, especialmente en áreas como análisis de datos y posibles integraciones con Machine Learning en el futuro. La facilidad de integración con otras tecnologías y su popularidad en la comunidad científica y de análisis de datos hacen de Python una opción flexible y eficiente para estas tareas adicionales.
5. **Infraestructura y Contenedores**
   * **Docker**
     + **Razón de elección**: **Docker** se seleccionó para la contenedorización de la aplicación. Esto permite que el sistema se despliegue de manera eficiente en diferentes entornos, asegurando que funcione de manera consistente tanto en desarrollo como en producción. La contenedorización facilita el manejo de dependencias y la implementación continua (CI/CD), lo que mejora la agilidad en el desarrollo y facilita el escalado del sistema.

**Decisiones de Diseño**

1. **Arquitectura del Sistema: Arquitectura Modular y Escalable**
   * **Razón de elección**: La arquitectura del sistema está basada en un enfoque modular, lo que significa que cada módulo (frontend, backend, base de datos, caché) tiene una responsabilidad claramente definida. Esta arquitectura facilita el mantenimiento, la extensión y el escalado de la aplicación
2. **Interfaz de Usuario: Interfaz Reactiva y Responsiva**

* **Razón de elección**: La interfaz de usuario se ha diseñado para ser reactiva y adaptativa, utilizando **React.js** para garantizar una experiencia de usuario fluida en dispositivos móviles y de escritorio.

**3. Seguridad y Gestión de Datos**

* **Razón de elección**: La seguridad es un aspecto clave del diseño, especialmente en un sistema que maneja datos sensibles de usuarios. Se implementará un enfoque basado en **JWT** para la autenticación de usuarios, garantizando que las sesiones sean seguras y los datos estén protegidos. Además, el uso de **HTTPS** asegura que todas las comunicaciones sean cifradas. Los datos de los usuarios y transacciones se almacenarán de manera segura en **PostgreSQL**, y se emplearán medidas como la encriptación de contraseñas y la validación de entradas para prevenir accesos no autorizados.

**4. Optimización del Rendimiento: Uso de Redis para Caché**

* **Razón de elección**: El uso de **Redis** como caché mejora el rendimiento general del sistema al reducir la latencia en las consultas más frecuentes.

**5. Escalabilidad: Despliegue Contenerizado con Docker**

* **Razón de elección**: **Docker** permite que el sistema sea fácilmente escalable al garantizar que todas las dependencias y servicios se empaqueten en contenedores. La escalabilidad es esencial para el crecimiento a largo plazo de **TYME**, ya que se espera que aumenten tanto el número de usuarios como las funcionalidades del sistema.

***14. Historias de Usuario.***

**Administración de Usuarios**

1. Registro e Inicio de Sesión

* Como nuevo usuario, quiero registrarme en la aplicación para poder acceder a los servicios del gimnasio.
* Como usuario registrado, quiero iniciar sesión con mis credenciales para acceder a mi perfil y funcionalidades.
* Como usuario, quiero poder recuperar mi contraseña si la olvido para no perder acceso a mi cuenta.

1. Gestión de Perfiles

* Como usuario, quiero ver y editar la información de mi perfil para mantener mis datos actualizados.

1. Control de Accesos

* Como administrador, quiero tener permisos especiales para gestionar usuarios y clases.
* Como profesor, quiero acceder únicamente a funcionalidades relacionadas con mis clases y rutinas.
* Como cliente, quiero tener acceso solo a mi perfil, clases y rutinas asignadas.

**Gestión de Clases y Horarios**

1. Creación y Administración de Clases

* Como profesor, quiero crear clases con horarios y cupos para organizar mi calendario de entrenamientos.
* Como profesor, quiero ver cuántas personas se han inscrito en mis clases para tener un control del aforo.

1. Inscripción de Usuarios

* Como usuario, quiero inscribirme en clases disponibles para poder participar en ellas.
* Como usuario, quiero recibir una confirmación al inscribirme en una clase para saber que estoy registrado.
* Como usuario con membresía vencida, quiero recibir un aviso de que no puedo inscribirme hasta renovar mi plan.

1. Alertas de Cupos

* Como usuario, quiero recibir notificaciones cuando queden pocos cupos en una clase para poder decidir a tiempo.
* Como sistema, quiero impedir nuevas inscripciones cuando la clase alcance su capacidad máxima para evitar sobrecupo.

1. Calendario Interactivo

* Como usuario, quiero ver un calendario con todas las clases programadas para planificar mi asistencia.
* Como usuario, quiero recibir notificaciones cuando se agreguen nuevas clases al calendario.

**Rutinas Personalizadas**

1. Asignación de Rutinas

* Como entrenador, quiero asignar rutinas personalizadas a los usuarios para optimizar sus entrenamientos.
* Como entrenador, quiero poder actualizar las rutinas según el progreso o necesidades del usuario.

**Control de Asistencia**

1. Registro Manual de Asistencia

* Como profesor, quiero registrar la asistencia de los participantes en cada clase para llevar un control preciso.
* Como profesor, quiero ver la lista de inscritos antes de la clase para organizar el espacio adecuadamente.

1. Generación de Estadísticas

* Como administrador, quiero acceder a estadísticas de asistencia y puntualidad para evaluar el desempeño general.

Gestión de Pagos y Membresías

1. Registro de Pagos

* Como administrador, quiero registrar manualmente los pagos de los usuarios para llevar el control de membresías activas.

1. Recordatorios de Pago

* Como usuario, quiero recibir recordatorios antes de que venza mi membresía para evitar la suspensión de mis clases.

1. Reportes Financieros

* Como administrador, quiero generar reportes de pagos realizados y pendientes para tener un control financiero claro.

**Comunicación Interna**

1. Muro de Mensajes / Mensajería

* Como usuario, quiero poder enviar y recibir mensajes dentro del sistema para mantenerme comunicado con profesores y administradores.

1. Notificaciones de Mensajes

* Como usuario, quiero recibir alertas cuando alguien me envíe un mensaje para no perderme información importante.

**Requisitos No Funcionales (expresados como historias técnicas o de calidad)**

1. Rendimiento Multiplataforma

* Como usuario, quiero que la aplicación funcione correctamente en cualquier dispositivo para usarla donde me convenga.

1. Seguridad de Datos

* Como usuario, quiero que mi información esté protegida mediante autenticación segura y cifrado para sentirme seguro al usar la plataforma.

1. Interfaz Intuitiva

* Como usuario, quiero una interfaz fácil de usar para navegar sin complicaciones, incluso si no soy experto en tecnología.

1. Mantenibilidad y Escalabilidad

* Como desarrollador, quiero que el sistema esté bien documentado y modular para facilitar su mantenimiento y evolución futura.

***15. Referencias***

Recursos Adicionales:

- <https://www.figma.com/design/PbkVtYmvFmBwZuq8VdrFk6/Gym-Client-Based-Website-and-Gym-Management-System-UI-Design-(Community)?node-id=0-1&p=f&t=hqbOoTYLhVoBIgKs-0>

- <https://lucid.app>

***16. Firma de Aprobación Inicial***

Firma de los miembros del equipo, Cristian Urqueta y Juan Contreras

Firma del profesor encargado del proyecto, Profesor Eric Ross