GESTIÒN GIMNASIO

ESTUDIANTES: Santiago Romero Saavedra - Ricardo Palomino - Daniel Vinasco

GRUPO: 3

DOCENTES: Juan Carlos Mariño

CAMPUSLANDS
SALON U1
RUTA NODE.JS
FLORIDABLANCA
2025

GESTIÒN GIMNASIO

1. SITUACIÓN PROBLEMA

En la actualidad, los gimnasios y entrenadores personales enfrentan la necesidad de gestionar de manera integral a sus clientes desde el registro de planes de entrenamiento, nutrición, seguimiento físico y contratos, hasta la administración de ingresos y egresos. Muchos de estos procesos aùn se llevan en formatos manuales o herramientas dispersas, lo que dificulta la consistencia de la informaciòn, lo que puede llevar a un aumento del riesgo de cometer errores y limita el control financiero.

Este proyecto surge de la oportunidad de centralizar en un aplicaciòn de lìnea de comandos (CLI) el control de todas estas gestiones, integrando ademàs principios de programaciòn orientada a objetos, patrones de diseño, transacciòn en base de datos y persistencia segura en MongoDB. De esta forma, se busca brindar una solución escalable y organizada que permita optimizar la administración del gimnasio, poder garantizar la consistencia de los datos y mejorar la experiencia tanto de clientes como de los administradores.

2. LEVANTAMIENTO DE REQUERIMIENTOS

3. REQUERIMIENTOS

3.1. Requerimientos Funcionales

- RF01: CRUD de clientes.
- RF02: CRUD de planes de entrenamiento.
- RF03: Asiciaciòn de clientes a planes de entrenamiento con generaciòn automàtica de contrato.
- RF04: Registro de avances físicos semanales (peso, grasa corporal, medidas, fotos, comentarios).
- RF05: Consulta cronológica del proceso físico de un cliente.
- RF06: CRUD de planes de nutrición y registro de alimentos por día.
- RF07: Consulta de reporte nutricional semanal.
- RF08: Registro de ingresos y egresos del gimnasio.
- RF09:Consulta de balance financiero por cliente o fecha.
- RF10: Implementaciòn de transacciones en operaciones crìticas(pagos, cancelaciòn de planes).

3.2. Requerimientos No Funcionales

- RF01: La aplicación debe ejecutarse desde la línea de comandos.
- RF02: Debe estar desarrollada en Node.js aplicando POO
- RF03: Debe cumplir con principios SOLID:

- RF04: Debe implementar al menos dos patrones de diseño
- RF05: La persistencia debe realizarse en MongoDB con el driver oficial.
- RF06: Se deben usar librerías npm para mejorar las experiencia de consola(chalk, inquirer,dotenv, daysjs).
- RF07: Uso obligatorio de convenciones de commits (Conventional Commits).
- RF08: Código estructurado en carpetas: /models, /services, /commands, /config, /utils.
- RF09: La documentaciòn y planeaciòn deben estar disponibles en el repositorio.

4. HISTORIAS DE USUARIO CON CRITERIOS DE ACEPTACIÓN

1.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF01	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Gestión de Clientes**		
Descripción			
Como administrador quiero registrar, listar, actualizar y eliminar clientes para mantener organizada la base de datos.			
Funcionalidad			
CRUD de clientes.			
Criterios de aceptación	● El sistema debe permitir crear un cliente con todos los campos requeridos. ● El sistema debe validar duplicados por documento o correo. ● El sistema debe permitir editar datos de un cliente existente. ● El sistema debe permitir eliminar clientes sin planes activos.		
Restricciones			
Sólo un administrador puede gestionar clientes.			

2.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF02	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Gestión de Planes**		
Descripción			
Como administrador quiero crear, actualizar y eliminar planes de entrenamiento para ofrecer variedad a los clientes.			
Funcionalidad			
CRUD de planes de entrenamiento.			
Criterios de aceptación	● El sistema debe permitir registrar un plan con nombre, descripción, precio y duración. ● El sistema debe validar que no existan dos planes con el mismo nombre. ● El sistema debe permitir modificar planes activos. ● El sistema debe impedir eliminar un plan con clientes asociados.		
Restricciones			
Solo administradores pueden gestionar planes.			

3.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF03	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Asociación Cliente – Plan**		
Descripción			
Como administrador quiero asociar un cliente a un plan de entrenamiento para formalizar su proceso con contrato automático.			
Funcionalidad			
Asociación de cliente y plan, con contrato generado.			
Criterios de aceptación	● El contrato debe generarse automáticamente con fecha inicio y fin. ● El sistema debe permitir cancelar un contrato y hacer rollback de seguimiento físico. ● El sistema debe permitir renovar contratos.		
Restricciones			
La transacción debe ser atómica.			

4.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF04	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Gestión de Pagos**		
Descripción			
Como administrador quiero registrar pagos de los clientes para llevar control financiero.			
Funcionalidad			
Registro y validación de pagos.			
Criterios de aceptación	● El sistema debe permitir registrar pagos manuales o automáticos. ● Debe validar montos y fechas de vencimiento. ● El sistema debe generar comprobantes digitales.		
Restricciones			
Solo los administradores pueden aprobar pagos.			

5.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF05	**Actor**	Cliente
NOMBRE DEL REQUERIMIENTO	**Consulta de Planes**		
Descripción			
Como cliente quiero visualizar los planes de entrenamiento disponibles para elegir el más conveniente.			
Funcionalidad			
Listado de planes.			
Criterios de aceptación	● El sistema debe mostrar nombre, precio y duración de cada plan. ● El cliente debe poder filtrar por precio o duración.		
Restricciones			
Solo usuarios autenticados pueden acceder.			

6.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF06	**Actor**	Cliente
NOMBRE DEL REQUERIMIENTO	**Consulta de Historial de Pagos**		
Descripción			
Como cliente quiero consultar mis pagos realizados para tener control de mis finanzas.			
Funcionalidad			
Historial de pagos por cliente.			
Criterios de aceptación	● El sistema debe mostrar lista de pagos con fecha, monto y estado. ● El cliente debe poder descargar el historial en PDF.		
Restricciones			
Solo usuarios autenticados pueden acceder.			

7.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF07	**Actor**	Entrenador
NOMBRE DEL REQUERIMIENTO	**Gestión de Rutinas**		
Descripción			
Como entrenador quiero asignar rutinas de entrenamiento a los clientes para personalizar su progreso.			
Funcionalidad			
CRUD de rutinas.			
Criterios de aceptación	● El sistema debe permitir registrar ejercicios, series y repeticiones. ● Debe permitir modificar rutinas según la evolución del cliente.		
Restricciones			
Solo entrenadores certificados pueden crear rutinas.			

8.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF08	**Actor**	Entrenador
NOMBRE DEL REQUERIMIENTO	**Seguimiento de Progreso**		
Descripción			
Como entrenador quiero registrar medidas físicas y progreso de los clientes para dar un mejor acompañamiento.			
Funcionalidad			
Registro de datos físicos.			
Criterios de aceptación	● El sistema debe guardar peso, grasa corporal y medidas corporales. ● El sistema debe generar gráficos de evolución.		
Restricciones			
Solo entrenadores pueden registrar datos.			

9.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF09	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Gestión de Materiales**		
Descripción			
Como administrador quiero gestionar listas de materiales asociadas a clientes o entrenadores para llevar control de recursos.			
Funcionalidad			
CRUD de materiales por persona.			
Criterios de aceptación	● El sistema debe permitir registrar, editar y eliminar materiales. ● El sistema debe asociar materiales a un tipo de persona.		
Restricciones			
Solo administradores pueden modificar materiales.			

10.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF10	**Actor**	Cliente
NOMBRE DEL REQUERIMIENTO	**Notificaciones de Pagos**		
Descripción			
Como cliente quiero recibir notificaciones de mis pagos para estar informado de mis obligaciones.			
Funcionalidad			
Sistema de alertas.			
Criterios de aceptación	● El sistema debe enviar recordatorios por correo o app. ● El sistema debe notificar pagos confirmados.		
Restricciones			
Notificaciones configurables por el usuario.			

11.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF11	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Reportes Financieros**		
Descripción			
Como administrador quiero generar reportes financieros de pagos y deudas para analizar ingresos.			
Funcionalidad			
Generación de reportes.			
Criterios de aceptación	● El sistema debe permitir exportar reportes en PDF y Excel. ● Debe mostrar totales de ingresos y deudas pendientes.		
Restricciones			
Acceso solo para administradores.			

12.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF12	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Gestión de Entrenadores**		
Descripción			
Como administrador quiero registrar, actualizar y eliminar entrenadores para mantener actualizada la base de instructores.			
Funcionalidad			
CRUD de entrenadores.			
Criterios de aceptación	● El sistema debe validar que no existan entrenadores duplicados. ● El sistema debe permitir asignar planes o clientes a entrenadores.		
Restricciones			
Solo administradores tienen acceso.			

13.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF13	**Actor**	Cliente
NOMBRE DEL REQUERIMIENTO	**Seguimiento Personal**		
Descripción			
Como cliente quiero consultar mi evolución física para evaluar mi progreso.			
Funcionalidad			
Visualización de evolución física.			
Criterios de aceptación	● El sistema debe mostrar gráficas de peso y medidas. ● El cliente debe poder filtrar por fechas.		
Restricciones			
Solo el cliente y el entrenador asignado pueden ver los datos.			

14.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF14	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Control de Asistencia**		
Descripción			
Como administrador quiero llevar control de la asistencia de los clientes para validar uso del servicio.			
Funcionalidad			
Registro de asistencias.			
Criterios de aceptación	● El sistema debe registrar la entrada de cada cliente. ● El sistema debe generar reportes de asistencia.		
Restricciones			
Requiere autenticación en la entrada.			

15.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF15	**Actor**	Cliente
NOMBRE DEL REQUERIMIENTO	**Solicitud de Pausa de Plan**		
Descripción			
Como cliente quiero poder pausar mi plan de entrenamiento por motivos personales.			
Funcionalidad			
Solicitud de pausa temporal.			
Criterios de aceptación	● El sistema debe registrar motivo y fechas de pausa. ● El administrador debe aprobar o rechazar la solicitud.		
Restricciones			
Solo una pausa por contrato.			

16.

HISTORIA DE USUARIO			
Prioridad: Baja			
CÓDIGO DEL REQUERIMIENTO:	RF16	**Actor**	Cliente
NOMBRE DEL REQUERIMIENTO	**Solicitud de Cambio de Plan**		
Descripción			
Como cliente quiero cambiar mi plan a otro superior para mejorar mi entrenamiento.			
Funcionalidad			
Cambio de plan.			
Criterios de aceptación	● El sistema debe recalcular contrato y pagos. ● El administrador debe aprobar la solicitud.		
Restricciones			
Solo se permite cambio a planes superiores.			

17.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF17	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Backup Automático**		
Descripción			
Como administrador quiero que el sistema realice copias de seguridad automáticas para proteger la información.			
Funcionalidad			
Respaldo de datos.			
Criterios de aceptación	● El sistema debe generar backup diario automático. ● Los backups deben guardarse en la nube.		
Restricciones			
Acceso restringido a administradores.			

18.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF18	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Gestión de Inventario de Equipos**		
Descripción			
Como administrador quiero controlar el inventario de equipos y materiales para garantizar disponibilidad.			
Funcionalidad			
CRUD de inventario.			
Criterios de aceptación	● El sistema debe permitir registrar equipos y su estado. ● Debe enviar alertas de mantenimiento.		
Restricciones			
Solo los administradores acceden.			

19.

HISTORIA DE USUARIO			
Prioridad: Baja			
CÓDIGO DEL REQUERIMIENTO:	RF19	**Actor**	Cliente
NOMBRE DEL REQUERIMIENTO	**Valoración de Entrenador**		
Descripción			
Como cliente quiero poder calificar a mi entrenador para retroalimentar la calidad del servicio.			
Funcionalidad			
Sistema de calificaciones.			
Criterios de aceptación	● El sistema debe permitir calificar de 1 a 5 estrellas. ● El cliente puede dejar un comentario.		
Restricciones			
Una calificación por contrato.			

20.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF20	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Reportes de Retención de Clientes**		
Descripción			
Como administrador quiero generar reportes de retención para analizar el nivel de satisfacción.			
Funcionalidad			
Generación de reportes.			
Criterios de aceptación	● El sistema debe calcular el porcentaje de clientes renovados. ● Debe permitir filtrar por periodos.		
Restricciones			
Acceso restringido a administradores.			

5. METODOLOGÍA

Para la ejecución de este proyecto, se aplicó el marco de trabajo ágil Scrum complementado con la metodología Kanban. Scrum proporciona la estructura iterativa e incremental a través de sprints, roles definidos y eventos clave, mientras que Kanban permite gestionar de manera visual y continúa de las tareas dentro de cada sprint mediante un tablero dividido en etapas como Por hacer, En proceso y Finalizado. Esta combinación es ideal para proyectos de este tipo, ya que fomenta la transparencia, facilita la incorporación de retroalimentación constante y mejora el flujo de trabajo al evitar cuellos de botella en el desarrollo.

Roles y Responsabilidades

- **Product Owner:** Encargado de la gestión del product backlog. Su rol es definir y priorizar las historias de usuario (los requerimientos funcionales del proyecto) para asegurar que el equipo de desarrollo se enfoque en las funcionalidades de mayor valor.
- **Scrum Master:** Responsable de guiar al equipo en la correcta aplicación de Scrum. Su función principal es facilitar los eventos, eliminar cualquier impedimento que el equipo pueda enfrentar y asegurar la fluidez del proceso de desarrollo.
- **Developers:** El equipo de desarrollo, encargado de implementar las funcionalidades. Este rol incluye la estimación de las tareas, la programación, las pruebas y el aseguramiento de la calidad del código.

Gestión Kanban

Para garantizar la transparencia y el progreso continuo del proyecto, se aplicará el **marco de trabajo Scrum** complementado con la **metodología Kanban**. Scrum organiza el desarrollo en sprints con eventos definidos, mientras que Kanban permitirá gestionar visualmente las tareas dentro de cada iteración, asegurando un flujo constante y controlado del trabajo.

Eventos Scrum

- **Sprint Planning:** Al inicio de cada sprint, el equipo seleccionará las historias de usuario del backlog priorizado. Estas se descompondrá en tareas más pequeñas y se ubicará en el tablero Kanban (*Por hacer, En proceso, Finalizado*), permitiendo dar seguimiento al avance.

- **Daily Stand Up:** Reunión diaria de 15 minutos donde cada miembro expone lo realizado, lo que planea realizar y los posibles impedimentos. Esto, junto con el tablero Kanban, permite detectar bloqueos y coordinar esfuerzos de forma rápida.

- **Sprint Review:** Al finalizar el sprint, el equipo mostrará las funcionalidades terminadas al Product Owner y demás interesados, recogiendo retroalimentación.

- **Sprint Retrospective:** Espacio de reflexión para analizar lo que funcionó, lo que no, y definir mejoras en el uso combinado de Scrum y Kanban en el siguiente sprint.

Sprints del Proyecto

- **Sprint 1:** Configuración inicial y fundamentos del proyecto.
  Objetivo: Establecer el entorno de desarrollo, la estructura base y la conexión con la base de datos MongoDB. Se crearán los modelos iniciales de clientes y planes.

- **Sprint 2:** Gestión de clientes y planes de entrenamiento.
  Objetivo: Implementar CRUD de clientes (RF01) y planes (RF02), asociación de clientes y generación automática de contratos (RF03).

- **Sprint 3:** Seguimiento físico y nutrición.
  Objetivo: Permitir a los entrenadores registrar el seguimiento físico (RF08) y las rutinas (RF07), además del módulo de nutrición.

- **Sprint 4:** Finanzas y transacciones.
  Objetivo: Implementar gestión de pagos (RF04), control financiero (RF06, RF11) y transacciones consistentes.

- **Sprint 5:** Módulos de soporte y finalización.
  Objetivo: Incorporar control de asistencia (RF14), notificaciones (RF10), gestión de entrenadores (RF12), y finalizar con documentación y demo.
    -

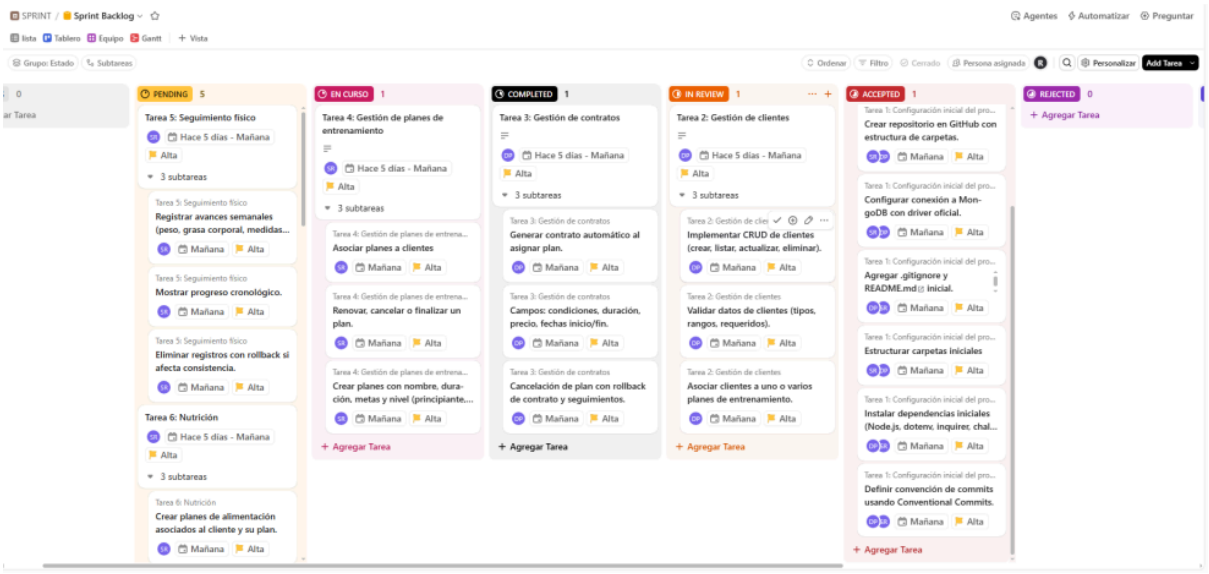
6. EVIDENCIA DE PLANTEAMIENTO DE PLATAFORMA DE TRABAJO

Acá se debe documentar toda la evidencia de trabajo colaborativo con los siguientes elementos:

- Link del repositorio: https://github.com/DanielSantiagoV/GymMaster_CLI
- Link de los videos que se grabaron en las reuniones realizada:
- Sprint Planning: https://youtu.be/wXVIVehaFBc
- Daily Stand: https://youtu.be/Zlr8DhSckHc
- Sprint Retrospective: https://youtu.be/532yoR12txE

Evidencias:

Tablero Scrum: https://sharing.clickup.com/90132524118/b/h/6-901320478559-2/fab4dcc99cf7825



Para la gestión del proyecto se utilizó la herramienta ClickUp, configurada bajo la metodología ágil Scrum. Este tablero permite organizar y dar seguimiento a todas las historias de usuario, requerimientos y tareas en cada etapa del sprint, asignando responsables, tiempos de entrega y prioridades.

Estructura del tablero

El tablero de ClickUp fue configurado con las siguientes columnas, representando el flujo de trabajo del equipo:

- **Pending:** tareas registradas en espera de inicio.
- **En Curso (In Progress):** tareas que se encontraban en ejecución.
- **In Review:** tareas que requerían validación antes de su aprobación.
- **Completed:** tareas finalizadas correctamente.
- **Accepted:** entregables que fueron aprobados formalmente dentro del sprint.
- **Rejected:** tareas descartadas o no aprobadas.

En la captura se puede observar:

- **Historias de usuario documentadas:** Ejemplo, *"Tarea 2. Gestión de clientes"* con sub-tareas como *CRUD de clientes* y *validación de datos*.
- **Responsables asignados:** Cada tarjeta muestra la persona encargada de su desarrollo.
- **Priorización:** Se emplearon etiquetas de prioridad (*Alta, Media, Baja*).
- **Gestión del tiempo:** Las tareas incluyen fechas límite para su cumplimiento dentro del sprint.
- **Flujo de trabajo:** Se evidencia cómo las tareas avanzaron desde *Pending* hasta *Accepted*, mostrando la trazabilidad del progreso.

El tablero Scrum en ClickUp permitió llevar un control visual y colaborativo del proyecto, asegurando la transparencia en el estado de las tareas, la gestión de prioridades y la organización del equipo durante todo el ciclo de trabajo.

Estructura Tareas:

Cada tarea dentro del tablero Scrum en ClickUp se configuró con una estructura clara que incluye la información esencial para su gestión. Esto permitió un mejor control del avance y la correcta asignación de responsabilidades.

Elementos de la estructura de cada tarea

- **Título descriptivo:** Identifica de manera clara el objetivo de la tarea (ejemplo: *"Tarea 1: Configuración inicial del proyecto"*).

- **Estado:** Indica la fase en la que se encuentra la tarea (*Pending, En Curso, In Review, Completed, Accepted, Rejected*).

- **Personas asignadas:** Cada tarea fue asignada a miembros específicos del equipo, garantizando la trazabilidad de responsabilidades.

- **Fechas:** Se establecieron fechas de inicio y finalización para cumplir con la planificación del sprint.

- **Prioridad:** Se clasificaron como *Alta, Media o Baja* según su relevancia en el sprint.

- **Descripción detallada:** Cada tarea contiene una explicación del propósito y las actividades a realizar (ejemplo: asegurar que el entorno de desarrollo esté configurado y el repositorio tenga buenas prácticas).

- **Subtareas (cuando aplica):** Para dividir tareas complejas en pasos más pequeños y manejables.

- **Etiquetas y relaciones (opcional):** Para vincular con otras tareas o identificar categorías específicas.

En la captura se observa la tarea *"Configuración inicial del proyecto"*, con los siguientes elementos destacados:

- Estado: **Accepted**.

- Prioridad: **Alta**.

- Responsables: **2 miembros asignados**.

- Fechas: asignadas desde el inicio hasta la finalización del sprint.

- Descripción: se establecieron las bases del proyecto, garantizando escalabilidad y organización futura.

Esta estructura de tareas permitía que cada miembro del equipo tuviera claridad sobre qué debía hacer, cuándo debía entregarlo y cuál era la prioridad, asegurando una gestión ágil y ordenada del proyecto.

Documentación Resultados

1. Resumen Ejecutivo del Producto

- **Objetivo Cumplido:** El proyecto ha completado exitosamente el desarrollo de una Aplicación de Línea de Comandos (CLI) en Node.js, diseñada para la gestión integral de un gimnasio o entrenador personal.
- **Valor Entregado:** La herramienta resuelve el problema central de inconsistencia de datos y la fricción operacional al unificar la gestión de clientes, planes, contratos, seguimiento y finanzas.
- **Estado Final:** El sistema está funcional y listo para la implementación en un entorno de producción, cumpliendo con los estándares de calidad técnica.

2. Stack Tecnológico y Arquitectura Resaltada

Esta sección subraya los requisitos técnicos más exigentes.

Componente	Tecnología Utilizada	Función y Valor Agregado
Backend	**Node.js**	Entorno de ejecución que permite una CLI rápida y eficiente.
Persistencia	**MongoDB (Driver Oficial)**	Base de datos NoSQL flexible, utilizada directamente para el control total sobre las transacciones.
Interacción CLI	**`inquirer` y `chalk`**	Asegura una interfaz de consola guiada e intuitiva, mejorando la experiencia del usuario final.
Estructura	**Programación Orientada a Objetos (POO)**	Arquitectura modular con clases para Cliente, Plan, Contrato, etc., facilitando la mantenibilidad y el desarrollo futuro.

3. Resultados y Funcionamiento Clave (Cumplimiento Funcional)

A diferencia de un manual de usuario, enfócate en lo que la aplicación *logra* en términos de valor de negocio.

Funcionalidad Clave	Cumplimiento Basado en Requerimiento	Mecanismo Implementado
Gestión Central	**CRUD de Clientes y Planes (RF01, RF02)**	Módulos ClientService y PlanService permiten la creación, edición y eliminación de datos, incluyendo la validación de duplicados por documento o correo.
Automatización	**Asociación Cliente-Plan y Contrato (RF03)**	El sistema ejecuta una función que, al asignar un plan, genera automáticamente el contrato con fecha de inicio y fin, vinculando tres colecciones.
Integridad de Datos	**Transacciones Atómicas (RF10)**	Se implementó el uso de session y transaction de MongoDB en operaciones críticas (ej. registro de pagos y cancelación de planes) para asegurar que la base de datos se actualice solo si todas las partes de la operación son exitosas.
Trazabilidad	**Seguimiento Físico y Reportes (RF05, RF11)**	El sistema registra avances cronológicos y permite al Administrador generar reportes financieros (Ingresos/Egresos) y al Cliente/Entrenador consultar su historial de progreso.

4. Cumplimiento de Calidad y Arquitectura (Técnico)

Aquí es donde demuestras que tu metodología Kanban permite enfocarte en la calidad.

A. Principios SOLID y Patrones de Diseño

- **Principio de Responsabilidad Única (SRP):** El código está segregado en módulos; por ejemplo, el ClientRepository solo maneja la interacción con la base de datos (MongoDB), mientras que el ClientService maneja la lógica de negocio (validaciones, llamadas transaccionales).
- **Patrón Repository:** Se utilizó este patrón para aislar la lógica de acceso a datos. El **ClientService** no sabe si usa MongoDB, un *array* o un archivo, lo que facilita el cambio de tecnología de persistencia.
- **POO:** Todos los elementos principales (Client, Plan, Contract) son objetos con sus propios métodos y validaciones, garantizando un alto nivel de cohesión y bajo acoplamiento.

B. Flujo de Trabajo y Cumplimiento de Tareas

El uso de **Kanban** permite un flujo de trabajo enfocado en la calidad:

- **Sin Sacrificio de Calidad:** La limitación del WIP (Work In Progress) permitió al Developer dedicar el tiempo necesario a tareas complejas como la implementación de transacciones (que fueron una prioridad en la columna 'Ready'), evitando la presión de tiempo de un *sprint* fijo.
- **Transparencia:** El tablero Kanban sirvió como el registro de tareas completadas, evidenciando que la priorización de los módulos clave (Fundamentos, Gestión Central, Integridad Transaccional) fue la guía para el desarrollo, cumpliendo con el orden propuesto en el plan inicial.

7. CONCLUSIONES

Conclusiones Generales del Documento

El proyecto de gestión de gimnasio, desarrollado como una aplicación de línea de comandos (CLI), fue completado con éxito. Se logró la implementación de las funcionalidades clave, cumpliendo con la mayoría de los requerimientos funcionales y técnicos establecidos. La aplicación ofrece una solución integral para la gestión de clientes, planes, seguimiento físico y finanzas, demostrando la viabilidad de una herramienta de este tipo para un entrenador personal o un gimnasio pequeño.

Desde el punto de vista técnico, se cumplió con los requisitos más importantes del proyecto. Se desarrolló la aplicación íntegramente en Node.js, aplicando con rigurosidad la Programación Orientada a Objetos y los principios SOLID, lo que resultó en un código modular, mantenible y escalable. La persistencia de datos en MongoDB utilizando transacciones reales en operaciones críticas, como la gestión de pagos y la cancelación de planes, aseguró la integridad y consistencia de la información. La elección de librerías como inquirer mejoró la experiencia de usuario, haciendo la interacción en la consola más intuitiva y guiada.

Conclusiones de la Reunión de Retrospectiva del Sprint

El equipo de desarrollo realizó una retrospectiva al finalizar el proyecto para analizar el proceso de trabajo y proponer mejoras.

Aspectos Positivos:

- **Comunicación Fluida:** La comunicación constante en los Daily Stand-Ups fue crucial para identificar y resolver impedimentos de manera rápida y eficiente. Esto evitó retrasos significativos y mantuvo al equipo sincronizado en todo momento.
- **Gestión del Backlog:** La metodología Scrum demostró ser una herramienta poderosa. La priorización de las historias de usuario por parte del Product Owner y la división del trabajo en sprints nos permitió enfocarnos en la entrega de funcionalidades de alto valor de forma incremental.

Áreas de Mejora:

- **Estimación de Tareas:** Se identificó la necesidad de mejorar la precisión en las estimaciones de tiempo para las tareas más complejas. En futuras iteraciones, el equipo dedicará más tiempo a un análisis técnico profundo durante el Sprint Planning para evitar el exceso de optimismo.

- **Documentación en el Código:** Aunque la documentación externa se realizó según los requisitos, la documentación interna (comentarios de código) podría ser más detallada, especialmente en las lógicas de negocio más complejas, para facilitar futuras modificaciones o la incorporación de nuevos desarrolladores.

Acciones a Tomar:

- Implementar un proceso de "refinamiento del backlog" a mitad de cada sprint para revisar las siguientes historias de usuario, desglosarlas en tareas y estimarlas con mayor precisión antes de la planificación del siguiente sprint.
- Establecer un estándar de comentarios de código más robusto, utilizando herramientas o guías para garantizar que la lógica de las funciones y métodos críticos quede clara para todo el equipo.