

# TeamFlow

## Nuestra Plataforma de Gestión de Proyectos para Equipos Remotos

### Fase 1: Planificación del proyecto

#### 1.1 Definición del alcance

- **Gestión de usuarios y equipos**
  - Registro e inicio de sesión
  - Perfiles de usuario
  - Creación y administración de equipos
  - Asignación de roles y permisos
- **Gestión de proyectos**
  - Creación y configuración de proyectos
  - Tableros Kanban personalizables
  - Seguimiento de tareas y estado
  - Fechas límite y dependencias
- **Colaboración**
  - Comentarios en tareas
  - Compartir archivos
  - Notificaciones en tiempo real
- **Seguimiento de tiempo**
  - Registro de horas trabajadas
  - Reportes de tiempo
  - Cronómetros para tareas
- **Análisis y reportes (Falta por implementar)**
  - Estadísticas de productividad
  - Informes de avance
  - Exportación de datos

#### 1.2 Arquitectura técnica

Definamos la arquitectura y stack tecnológico:

- **Frontend:**
  - Vue.js 3 (con Composition API)
  - Vuex para gestión de estado
  - Vue Router para navegación
  - Tailwind CSS para UI
- **Backend:**
  - Node.js con NestJS (framework TypeScript)
  - REST API + Websockets para tiempo real
  - JWT para autenticación
- **Base de datos:**
  - PostgreSQL para datos relacionales
  - Redis para caché y sesiones
- **Infraestructura:**
  - Docker para contenedores
  - Docker Compose para orquestar servicios
  - Nginx como proxy inverso
  - GitHub para control de versiones

## Fase 2: Configuración inicial

### 2.1 Creación del repositorio

Empecemos por crear el repositorio en GitHub:

1. Crea una cuenta en GitHub si aún no tienes una
2. Crea un nuevo repositorio (por ejemplo, "remote-team-project-manager")
3. Clona el repositorio en tu máquina local
4. Crea un archivo README.md inicial con la descripción del proyecto

### 2.2 Estructura de directorios

Organicemos el proyecto con la siguiente estructura:

## 2.3 Configuración Docker

# Fase 3: Configuración del Backend (NestJS)

## 3.1 Iniciar proyecto NestJS

Primero, configuremos el backend:

1. Crea el directorio server e inicializa un proyecto NestJS:
2. Crea un Dockerfile para el backend:

## 3.2 Estructurar módulos principales

Organizaremos el backend en módulos:

- AuthModule: Autenticación y usuarios
- DashboardModule: Extracción de data para ser presentada en métricas
- ProjectsModule: Proyectos y equipos
- TasksModule: Tareas y tableros Kanban
- Time-trackingModule: Seguimiento de tiempo
- CommentsModule: Comentarios en tareas
- AttachmentsModule: Archivos adjuntados en tareas
- NotificationsModule: Sistema de notificaciones
- (Falta el reports Module, que será para hacer reportes de rendimiento en usuarios y equipos)

# Fase 4: Diseño de la base de datos

## 5.1 Esquema de base de datos

Definamos las principales tablas:  

Y aquí debajo encontrarás todas las tablas más a detalle   

	table_name	name	lock
1	comments		
2	migrations		
3	notifications		
4	projects		
5	task_attachmen...		
6	tasks		
7	team_members		
8	teams		
9	time_entries		
10	users		

users	teams	projects	tasks
<ul style="list-style-type: none"><li>Columns (9)</li></ul> <ul style="list-style-type: none"><li>id</li><li>username</li><li>email</li><li>password_hash</li><li>full_name</li><li>role</li><li>avatar_url</li><li>created_at</li><li>updated_at</li></ul>	<ul style="list-style-type: none"><li>Columns (6)</li></ul> <ul style="list-style-type: none"><li>id</li><li>name</li><li>description</li><li>avatar_url</li><li>created_at</li><li>updated_at</li></ul>	<ul style="list-style-type: none"><li>Columns (9)</li></ul> <ul style="list-style-type: none"><li>id</li><li>name</li><li>description</li><li>status</li><li>start_date</li><li>end_date</li><li>created_at</li><li>updated_at</li><li>team_id</li></ul>	<ul style="list-style-type: none"><li>Columns (12)</li></ul> <ul style="list-style-type: none"><li>id</li><li>title</li><li>description</li><li>status</li><li>priority</li><li>due_date</li><li>estimated_hours</li><li>created_at</li><li>updated_at</li><li>created_by_id</li><li>assigned_to_id</li><li>project_id</li></ul>

team_members	time_entries	task_attachments	notifications
<ul style="list-style-type: none"><li>Columns (4)</li></ul> <ul style="list-style-type: none"><li>id</li><li>role</li><li>user_id</li><li>team_id</li></ul>	<ul style="list-style-type: none"><li>Columns (8)</li></ul> <ul style="list-style-type: none"><li>id</li><li>description</li><li>start_time</li><li>end_time</li><li>duration_minute</li><li>created_at</li><li>task_id</li><li>user_id</li></ul>	<ul style="list-style-type: none"><li>Columns (8)</li></ul> <ul style="list-style-type: none"><li>id</li><li>file_name</li><li>file_path</li><li>mime_type</li><li>file_size_kb</li><li>created_at</li><li>task_id</li><li>uploaded_by_user_id</li></ul>	<ul style="list-style-type: none"><li>Columns (8)</li></ul> <ul style="list-style-type: none"><li>id</li><li>message</li><li>is_read</li><li>created_at</li><li>recipient_id</li><li>task_id</li><li>project_id</li><li>team_id</li></ul>

comments	migrations
<ul style="list-style-type: none"><li>Columns (6)</li></ul> <ul style="list-style-type: none"><li>id</li><li>content</li><li>created_at</li><li>updated_at</li><li>taskId</li><li>userId</li></ul>	<ul style="list-style-type: none"><li>Columns (3)</li></ul> <ul style="list-style-type: none"><li>id</li><li>timestamp</li><li>name</li></ul>

# Relaciones Esenciales del Sistema

## Jerarquía Principal:

USERS → TEAM\_MEMBERS → TEAMS → PROJECTS → TASKS

## Relaciones Clave:

### 1. Users ↔ Teams (Muchos a Muchos)

- team\_members.user\_id → users.id
- team\_members.team\_id → teams.id
- Un usuario puede estar en múltiples equipos

### 2. Teams → Projects (Uno a Muchos)

- projects.team\_id → teams.id
- Cada proyecto pertenece a un equipo

### 3. Projects → Tasks (Uno a Muchos)

- tasks.project\_id → projects.id
- Cada tarea pertenece a un proyecto

### 4. Users → Tasks (Uno a Muchos)

- tasks.assigned\_to\_id → users.id
- tasks.created\_by\_id → users.id
- Cada tarea se asigna a un usuario y es creada por otro

## Relaciones de Soporte:

### 5. Tasks → Actividades

- comments.task\_id → tasks.id (comentarios)
- time\_entries.task\_id → tasks.id (tiempo trabajado)
- task\_attachments.task\_id → tasks.id (archivos)

### 6. Users → Actividades

- comments.user\_id → users.id (quien comenta)
- time\_entries.user\_id → users.id (quien registra tiempo)
- notifications.recipient\_id → users.id (quien recibe notificación)

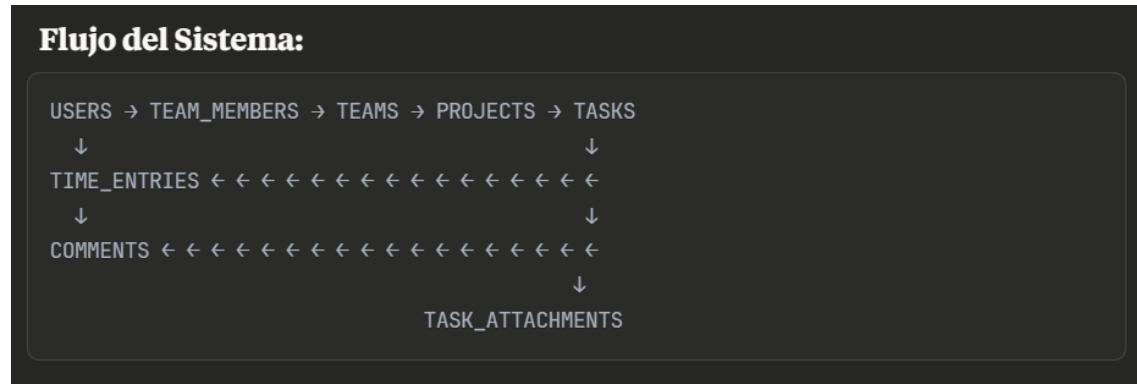
## Flujo de Trabajo:

1. **Usuarios** se unen a **equipos** (team\_members)
2. **Equipos** crean **proyectos**
3. **Proyectos** contienen **tareas**

4. Tareas se asignan a **usuarios** del equipo
5. Los usuarios pueden **comentar, adjuntar archivos y registrar tiempo** en las tareas

## Fase 6: Plan de desarrollo

Para avanzar de manera estructurada, sugiero el siguiente plan de desarrollo y flujo de sistema:



### Sprint 1: Configuración y autenticación

- Configurar entorno Docker
- Implementar registro e inicio de sesión
- Configurar autenticación JWT
- Crear perfil de usuario básico

### Sprint 2: Gestión de equipos y proyectos

- CRUD de equipos
- Gestión de miembros del equipo
- CRUD de proyectos
- Asignación de proyectos a equipos

### Sprint 3: Sistema de tareas y tableros Kanban

- Implementar tablero Kanban
- CRUD de tareas
- Arrastrar y soltar tareas entre columnas
- Filtros y búsqueda de tareas

### Sprint 4: Seguimiento de tiempo y colaboración

- Implementar cronómetro para tareas
- Registro manual de tiempo
- Comentarios en tareas
- Notificaciones básicas

## Sprint 5: Análisis y reportes

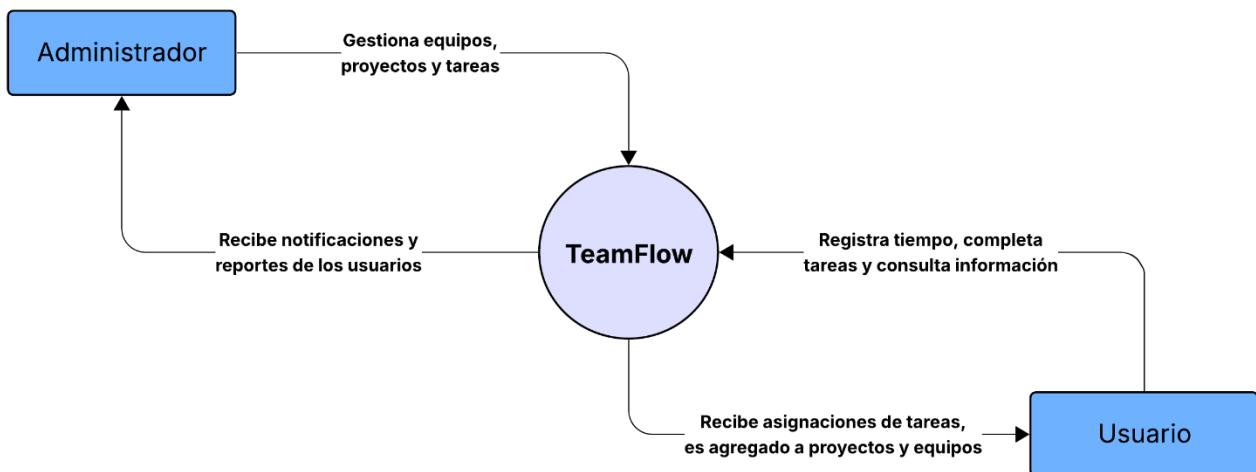
- Estadísticas de proyecto
- Informes de tiempo
- Exportación de datos
- Gráficos de progreso

## Sprint 6: Pulido y optimización

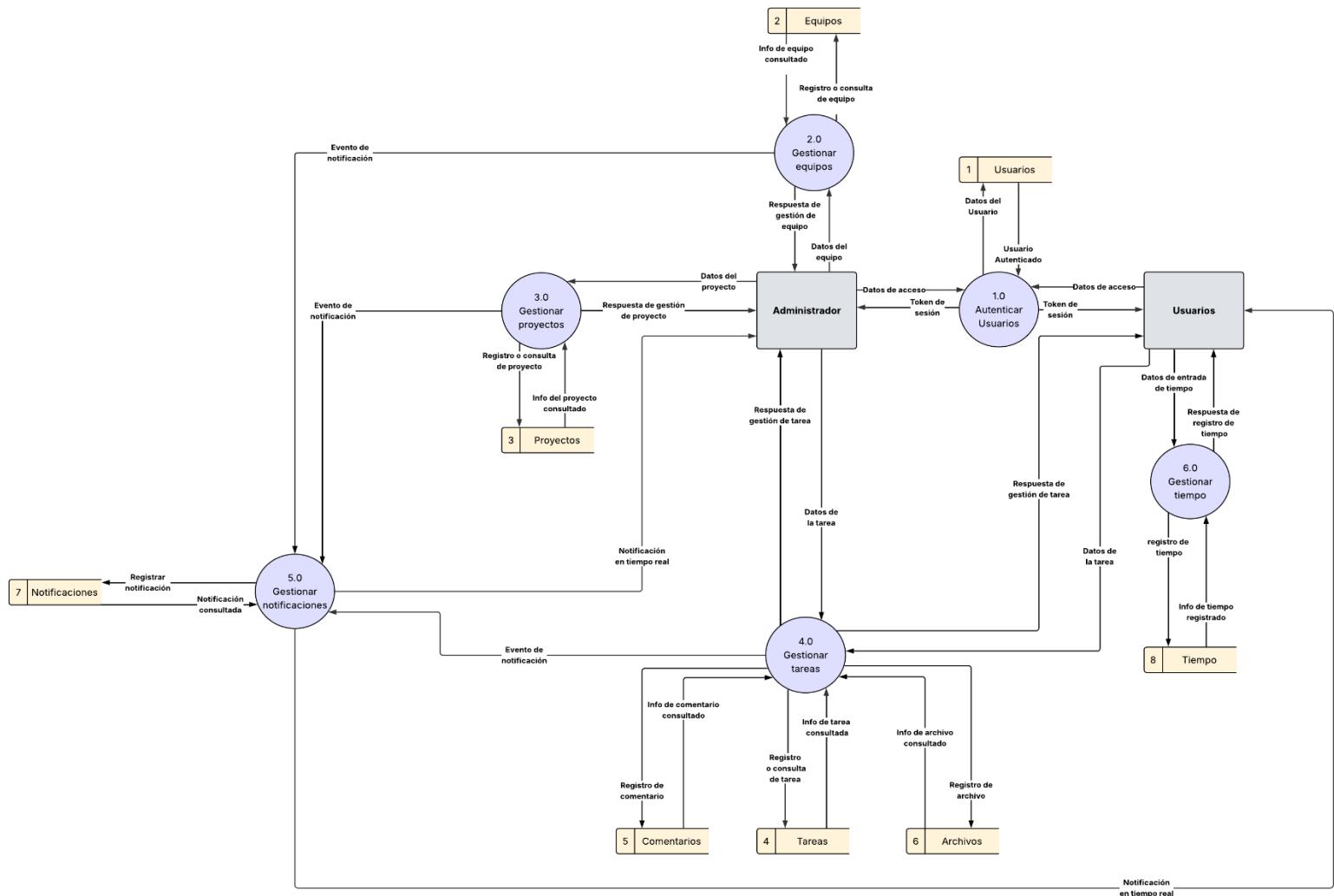
- Mejoras de UX/UI
- Optimización de rendimiento
- Pruebas de carga
- Documentación detallada

Diagramas de flujo: 

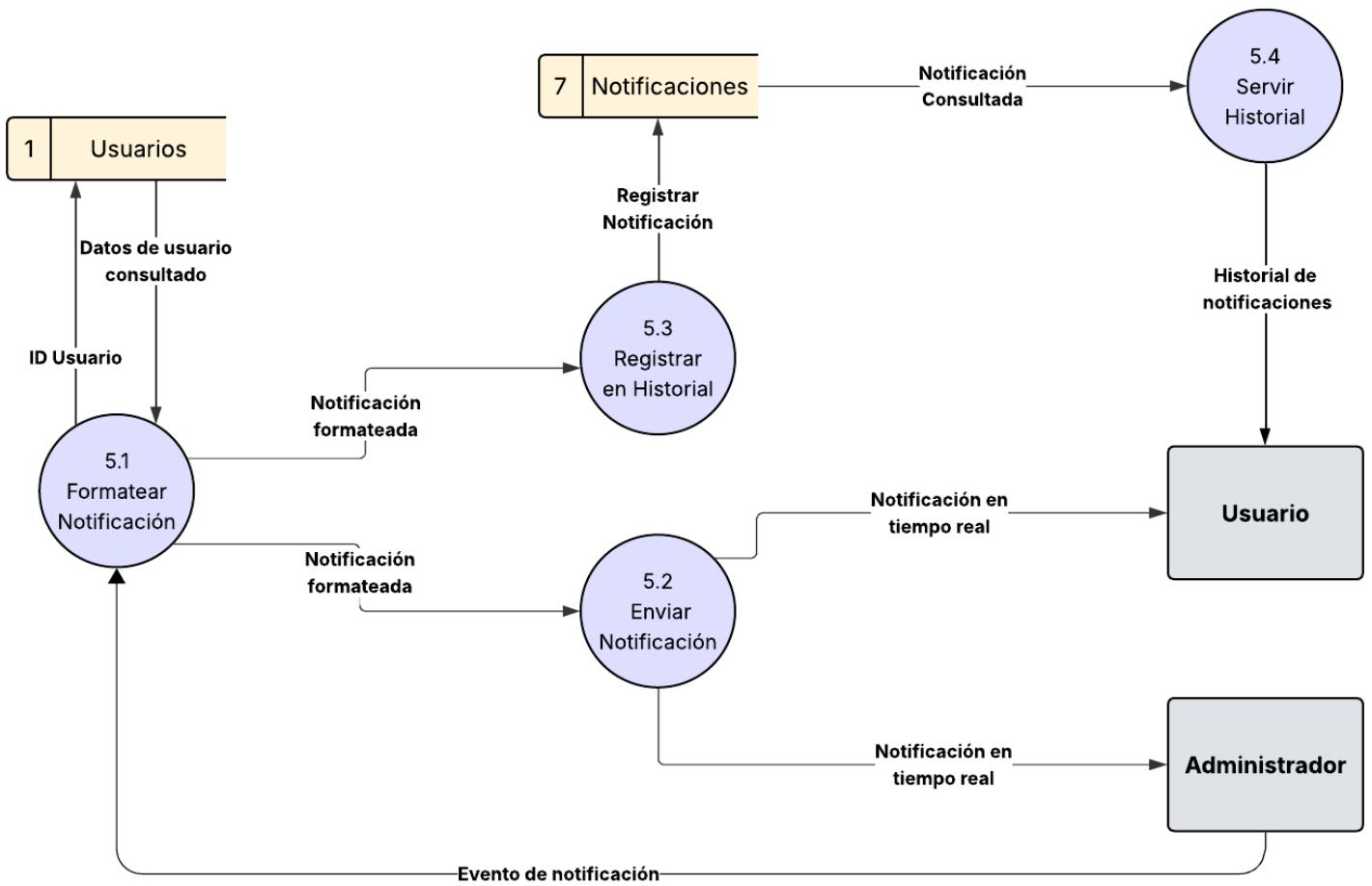
## Nivel 0



## Nivel 1



## Nivel 2: subprocesso de notificaciones



## Nivel 2: subprocesso creación de tareas

