

# Especificación y Plan de Desarrollo de Proyecto

Gestor de Tickets Colaborativo

13 de noviembre de 2025

## 1 Especificación del Proyecto: TicketFlow

### 1.1 Nombre Tentativo

- **TicketFlow:** Refleja la idea de un flujo de trabajo para la gestión de tickets.

### 1.2 Descripción General

TicketFlow es una aplicación web diseñada para la gestión interna de solicitudes, incidencias o tareas mediante un sistema de tickets. Su objetivo es:

- Facilitar la comunicación y el flujo de trabajo dentro de la organización.
- Permitir el seguimiento detallado de tareas.
- Medir los tiempos de respuesta y cierre de tickets.

Cada compañía registrada puede establecer áreas de trabajo, equipos y asignar tickets a usuarios o equipos específicos.

### 1.3 Funcionalidades Principales

#### 1.3.1 Compañías y Usuarios

- Registro individual de compañías en la plataforma.
- Un usuario puede pertenecer a una sola compañía (modelo inicial).
- **Funciones del Administrador de la Compañía:**
  1. Invitar usuarios.
  2. Crear áreas (Departamentos).
  3. Crear equipos (*Teams*).
  4. Definir etiquetas personalizadas.
- **Mejora Sugerida:** Permitir que un usuario pueda ser parte de múltiples equipos dentro de la misma compañía.

#### 1.3.2 Roles de Usuario

- **Administrador:** Puede ver todos los tickets de la compañía. Gestiona etiquetas, áreas, equipos y usuarios.
- **Usuario Normal:** Solo puede ver los tickets que ha enviado o recibido.
- **Extra:** Implementar un rol intermedio: **Team Leader**, con capacidad de ver todos los tickets asociados a su equipo.

#### 1.3.3 Estructura del Ticket

Cada ticket debe contener la siguiente información esencial:

- Título y Cuerpo (descripción del problema o tarea).
- Fecha y hora de creación.
- **Etiqueta Personalizada (Opcional):** Ejemplos: "Bug", "Datos", "Error", "Actualizar".
- **Etiqueta de Prioridad:** Baja, Media o Alta.
- **Emisor:** Usuario que crea el ticket.

- **Receptor:** Usuario o equipo al que se asigna el ticket.
- **Estado:** Abierto / En Progreso / Cerrado.
- Fecha y hora de cierre.
- Tiempo total transcurrido entre la creación y el cierre.
- **Extra:** Permitir comentarios o un hilo de seguimiento interno dentro de cada ticket.

#### 1.3.4 Equipos (*Teams*)

- Los usuarios pueden pertenecer a uno o más equipos (según mejora sugerida).
- Los tickets pueden asignarse a **Usuarios Individuales** o a **Equipos Completos**.
- Cada ticket asignado a un equipo debe mostrar claramente la etiqueta de dicho equipo.
- **Extra:** Utilizar los equipos como agrupadores de proyectos, permitiendo que un equipo gestione múltiples tickets relacionados bajo un mismo proyecto.

#### 1.3.5 Filtrado y Visualización

- **Criterios de Filtrado:** Estado, Prioridad, Fecha, Etiqueta personalizada y Equipo.
- **Vistas Disponibles:** Tickets enviados, Tickets recibidos, Tickets por equipo, Historial de tickets cerrados.
- **Extra:** Agregar un panel de estadísticas con gráficos de rendimiento (e.g., tiempo promedio de cierre por usuario o equipo).

#### 1.3.6 Autenticación

- Registro de usuarios mediante correo electrónico y contraseña.
- Implementación de Login seguro utilizando JWT (*JSON Web Token*).
- Funcionalidad de recuperación de contraseña.
- Perfil de usuario personalizable (foto, rol, descripción).
- **Extra:** Implementar inicio de sesión con proveedores externos (Google o Microsoft).

#### 1.3.7 Interfaz del Usuario (*UI/UX*)

- Diseño limpio, moderno e inspirado en la simplicidad (similar a Google Stitch).
- Panel lateral de navegación.
- Vista principal de tablero (*Dashboard*) con tarjetas para visualizar los tickets.
- Uso de un formulario modal para la creación de nuevos tickets.
- Sistema de notificaciones de asignación de tickets.
- **Extra:** Integrar una barra de búsqueda global para la localización rápida de tickets.

## 2 Plan de Desarrollo Paso a Paso

Este plan establece las seis fases de desarrollo para la implementación del proyecto TicketFlow.

### 2.1 Fase 1: Diseño Base (*Frontend*)

Objetivo: Establecer la estructura visual y los componentes reutilizables de la aplicación.

- Creación de la estructura HTML para las páginas principales:
  - `index.html` (Landing o Login)
  - `dashboard.html` (Home del usuario)
  - `tickets.html`, `teams.html`, `admin.html`, `profile.html`
- Aplicación de estilos mediante CSS puro o un *framework* (TailwindCSS o Bootstrap).
- Creación de componentes reutilizables: barra de navegación, barra lateral (*sidebar*) y modales.

### 2.2 Fase 2: Lógica *Frontend* (JavaScript)

Objetivo: Implementar la interactividad y la lógica de visualización sin conexión al *backend*.

- Conexión y manejo de eventos de botones y formularios.
- Simulación de datos con objetos JavaScript (datos *mock*) para pruebas.
- Renderización dinámica de tickets en tarjetas (manipulación del DOM).
- Implementación de la lógica de filtros por prioridad, estado y otros criterios.

### 2.3 Fase 3: *Backend* con Node.js y Express

Objetivo: Construir la infraestructura del servidor y la conexión a la base de datos.

- Creación del servidor con el *framework* Express.
- Configuración de rutas principales:
  - `/auth` (login, register, logout)
  - `/tickets` (CRUD completo)
  - `/teams` (creación, listado, asignación de usuarios)
  - `/users` (obtener y actualizar perfiles)
- Conexión y configuración de la base de datos (MongoDB o PostgreSQL).
- **Extra:** Uso de ORM/ODM (Prisma para SQL o Mongoose para MongoDB).

### 2.4 Fase 4: Autenticación

Objetivo: Asegurar la aplicación y controlar el acceso a los recursos.

- Implementación del sistema JWT en el *backend* para la gestión de sesiones.
- Protección de rutas sensibles mediante *middleware* de autenticación.
- Conexión final del *frontend* con las rutas de autenticación del *backend* (usando `fetch` o `axios`).

### 2.5 Fase 5: Funcionalidades Avanzadas

Objetivo: Implementar las características de valor añadido.

- Cálculo automático del tiempo transcurrido entre la creación y el cierre del ticket.
- Desarrollo del sistema de comentarios o notas dentro de cada ticket.
- Creación del módulo de estadísticas para el administrador (utilizando librerías de gráficos como Chart.js o Recharts).
- Implementación de notificaciones visuales o por correo electrónico (*Nodemailer*).

### 2.6 Fase 6: *Deploy*

Objetivo: Poner la aplicación a disposición de los usuarios.

- **Frontend Deployment:** En plataformas como Vercel o Netlify.

- **Backend Deployment:** En servicios como Render o Railway.
- **Base de Datos:** Configuración final en MongoDB Atlas o Supabase (para SQL).
- Configuración y vinculación de un dominio personalizado.