Sistema de Gestión de Proyectos - Requerimientos Esenciales

Objetivo

Desarrollar un sistema web para gestión de proyectos colaborativo con funcionalidades básicas de Trello/Monday, actualizaciones en tiempo real y notificaciones por email.

Stack Tecnológico

- **Backend**: Node.js + Express.js
- **Frontend**: EJS (plantillas) + HTML/CSS/JavaScript
- **Base de Datos**: MSSQL + Sequelize (ORM y migraciones)
- **Tiempo Real**: Socket.IO
- **Email**: Nodemailer con Gmail
- **Despliegue**: Render
- **Control de Versiones**: GitHub

Funcionalidades Esenciales

1. Autenticación y Usuarios

- **Registro/Login** con email y contraseña
- **Hash de contraseñas** (bcrypt)
- **Sesiones** (express-session)

2. Gestión de Proyectos

- **Crear proyecto** (solo Admin)
- **Listar proyectos** del usuario
- **Invitar usuarios** a proyectos por email
- **Estados básicos**: Activo, Completado

3. Gestión de Tareas

- **CRUD de tareas** dentro de proyectos
- **Estados**: Por hacer, En progreso, Completado
- **Asignación** a usuarios del proyecto
- **Campos básicos**: título, descripción, fecha límite, prioridad
- **Comentarios** en tareas

4. Actualizaciones en Tiempo Real

- **Notificaciones** cuando se crea/modifica/asigna tarea
- **Actualización automática** de la interfaz
- **Indicadores** de usuarios conectados

5. Notificaciones por Email

- **Asignación de tarea** → email al usuario asignado
- **Invitación a proyecto** → email de invitación
- **Comentarios** → email al propietario de la tarea

Estructura de Base de Datos (Normalizada)

Tablas Principales

Users (id, name, email, password, created_at)

Projects (id, name, description, owner_id, status, created_at)

ProjectMembers (project_id, user_id, joined_at)

Tasks (id, title, description, project_id, assigned_to, status, priority, due_date, created_at)

Comments (id, task_id, user_id, content, created_at)

Relaciones

- User → Projects (1:N - propietario)

- Projects ↔ Users (N:M - miembros)
- Project → Tasks (1:N)
- User → Tasks (1:N - asignado)
- Task → Comments (1:N)
- User → Comments (1:N)

API RESTful Esencial

Autenticación

- `POST /auth/register` - Registro
- `POST /auth/login` - Login
- `POST /auth/logout` - Logout

Proyectos

- `GET /projects` - Listar proyectos del usuario
- `POST /projects` - Crear proyecto
- `GET /projects/:id` - Ver proyecto específico
- `POST /projects/:id/invite` - Invitar usuario

Tareas

- `GET /projects/:id/tasks` - Listar tareas del proyecto
- `POST /projects/:id/tasks` - Crear tarea
- `PUT /tasks/:id` - Actualizar tarea
- `DELETE /tasks/:id` - Eliminar tarea
- `POST /tasks/:id/comments` - Agregar comentario

Vistas EJS Principales

- `login.ejs` - Página de login
- `dashboard.ejs` - Lista de proyectos
- `project.ejs` - Vista del proyecto (estilo Kanban básico)
- `task-modal.ejs` - Modal para crear/editar tarea

Configuración de Seguridad

- **Validación de entrada** (express-validator)
- **Sanitización** de datos
- **CORS** configurado
- **Rate limiting** básico
- **Variables de entorno** para credenciales

Configuración de Email (Gmail)

```
// Usar App Password de Gmail

GMAIL_USER=tu-email@gmail.com

GMAIL_PASS=tu-app-password
```

Despliegue en Render

- **Variables de entorno** para producción
- **Base de datos MSSQL** remota
- **Build script** automatizado

Cronograma Sugerido (Tiempo Limitado)

1. **Día 1-2**: Setup inicial + Base de datos + Autenticación
2. **Día 3-4**: CRUD Proyectos y Tareas + EJS views
3. **Día 5**: Socket.IO + Notificaciones email
4. **Día 6**: Deploy + Testing básico

Simplificaciones para Ahorrar Tiempo

- **Sin drag & drop** (solo cambio de estado por select)
- **Sin upload de archivos**
- **Sin notificaciones push**
- **CSS básico** (Bootstrap CDN)
- **Sin paginación** (límite de resultados)
- **Comentarios simples** (solo texto)

Estructura de Archivos Sugerida

project-manager/

├── config/

│ ├── database.js

│ └── email.js

├── models/

├── routes/

├── views/

├── public/

│ ├── css/

│ └── js/

├── middlewares/

├── migrations/

└── app.js