**Guía detallada de lo que falta por implementar (Sprint 2 y más allá)**

El **Sprint 2** está enfocado en la gestión de roles y la creación de dashboards específicos para cada tipo de usuario. Aquí está la guía detallada de lo que falta por implementar.

**Sprint 2: Gestión de Roles y Dashboards**

**Objetivo:**

* Crear dashboards personalizados según el rol del usuario: Alumno, Profesor, Administrador.
* Proteger rutas específicas según el rol.

**Pasos a implementar:**

1. **Gestión de Roles (Backend)**

* **Ubicación**: index.js
* Durante el registro de usuario, cada usuario debe tener un rol asignado (Alumno, Profesor, Administrador).
* El token JWT debe contener el rol del usuario, lo que permitirá verificar su acceso a rutas específicas.

1. **Proteger rutas por rol (Backend)**

* Implementar un middleware adicional que verifique el rol del usuario antes de permitir el acceso a ciertas rutas.

1. **Dashboards específicos por rol (Frontend)**

* **Ubicación**: Crear tres nuevos componentes en frontend/src/components:
* AlumnoDashboard.tsx: Mostrará el progreso del alumno y sus clases programadas.
* ProfesorDashboard.tsx: Mostrará las clases asignadas al profesor y permitirá registrar las horas impartidas.
* AdminDashboard.tsx: Permitirá al administrador gestionar usuarios y ver reportes.

1. **Proteger rutas por rol en el Frontend**

* Actualiza App.tsx para proteger las rutas según el rol. Utiliza ProtectedRoute pero con la verificación adicional del rol.

**Sprint 3: Funcionalidad avanzada**

**Objetivo:**

• Implementar la gestión de clases y el manejo de paquetes de horas.

**Pasos a implementar:**

1. **Asignar alumnos a profesores (Backend)**

* Crear una ruta donde el administrador pueda asignar alumnos a profesores.
* Guardar la relación en la base de datos.

1. **Registrar clases y paquetes de horas (Backend)**

* Profesores podrán registrar las horas impartidas.
* Alumnos podrán ver sus paquetes de horas restantes.
* Crear rutas específicas para gestionar estas operaciones.

**Sprint 4: Integración con Zoom y chat en tiempo real**

**Objetivo:**

* Integrar la API de Zoom para la gestión de las clases virtuales.
* Implementar un sistema de chat en tiempo real con WebSockets para la comunicación entre profesores y alumnos.

**Pasos a implementar:**

1. **Integrar API de Zoom**

* Utilizar la API de Zoom para crear enlaces de reuniones automáticamente y enviarlos a los alumnos y profesores.

1. **Implementar chat en tiempo real**

* Usar Socket.IO para permitir la comunicación en tiempo real entre los alumnos y los profesores.

**Próximos Sprints:**

1. **Sprint 5: Reportes y estadísticas**:

* Generar reportes mensuales para el administrador.
* Mostrar estadísticas como horas impartidas y pagos a profesores.

1. **Sprint 6: Despliegue en AWS**:

* Configurar AWS para desplegar tanto el frontend como el backend.
* Configurar el dominio, SSL, y la base de datos en AWS RDS.