Clase 4

Manejo avanzado de roles y permisos

¿Qué haremos hoy?

- Entender qué es RBAC y por qué es importante.
- Implementar un sistema de permisos en el backend.
- Proteger rutas y recursos según roles y permisos.
- Extender el frontend para verificar permisos.
- Mostrar/ocultar elementos de UI según permisos.
- Construir una experiencia contextual por rol.

¿Qué es RBAC?

Role-Based Access Control = Control de acceso basado en roles

- 👤 Cada usuario tiene un **rol** (`reporter`, `agent`, `admin`)
- 🔐 Cada rol tiene **permisos** específicos
- ✅ El sistema verifica permisos antes de permitir acciones
- 🚫 Si no tiene permiso → 403 Forbidden

Objetivo: Que cada usuario vea y haga solo lo que debe.

Los tres roles de nuestro sistema

Rol	Descripción	Permisos principales
Reporter	Reporta incidentes	Crear y ver propios
Agent	Resuelve incidentes	Ver todos, actualizar asignados
Admin	Administra el sistema	Acceso completo

Matriz de Permisos

Operación	Reporter	Agent	Admin
Crear incidente	✅	❌	✅
Leer incidentes propios	✅	❌	✅
Leer todos los incidentes	❌	✅	✅
Actualizar inc. asignados	❌	✅	N/A
Actualizar cualquier inc.	❌	❌	✅
Cancelar incidente	❌	❌	✅
Eliminar incidente	❌	❌	✅
Leer usuarios	❌	❌	✅

Backend: Sistema de Permisos

Archivo: `src/core/permissions.ts`

```
export const PERMISSIONS = {
  INCIDENTS_CREATE: 'incidents:create',
  INCIDENTS_READ_SELF: 'incidents:read:self',
  INCIDENTS_READ_ALL: 'incidents:read:all',
  INCIDENTS_UPDATE_ASSIGNED: 'incidents:update:assigned',
  INCIDENTS_UPDATE_ALL: 'incidents:update:all',
  INCIDENTS_CANCEL: 'incidents:cancel',
  INCIDENTS_DELETE: 'incidents:delete',
  USERS_READ_ALL: 'users:read:all',
}
```

Backend: Matriz de Roles

```
export const ROLE_PERMISSIONS: Record<UserRole, Permission[]> = {
  reporter: [PERMISSIONS.INCIDENTS_CREATE, PERMISSIONS.INCIDENTS_READ_SELF],
  agent: [
    PERMISSIONS.INCIDENTS_READ_ALL,
    PERMISSIONS.INCIDENTS_UPDATE_ASSIGNED,
  ],
  admin: [
    // Todos los permisos
  ],
}
```

Backend: Middleware `authorize()`

```
export function authorize(...requiredPermissions: Permission[]) {
  return (req: Request, res: Response, next: NextFunction) => {
    if (!req.user) {
      return res.status(401).json({ message: 'Unauthorized' })
    }
    if (!hasAllPermissions(req.user.role, requiredPermissions)) {
      return res.status(403).json({
        message: 'Forbidden',
        code: 'FORBIDDEN',
        required: requiredPermissions,
      })
    }
    return next()
  }
}
```

Backend: Protección de Rutas

Nivel de ruta - Permisos generales:

```
incidentsRouter.post(
  '/',
  authorize(PERMISSIONS.INCIDENTS_CREATE),
  createIncident
)

incidentsRouter.delete(
  '/:id',
  authorize(PERMISSIONS.INCIDENTS_DELETE),
  deleteIncident
)
```