

🔒 OrganizeZen - Plano de Hierarquia e Sistema de Permissões (RBAC)

Projeto: OrganizeZen

Autor: Bruno (Cabo Verde)

Assistente: DeepAgent

Data: 15 de Outubro de 2025

Versão: 1.0 - Planejamento



Índice

1. Visão Geral
2. Estrutura Hierárquica Atual
3. Matriz Completa de Permissões
4. Permissões Detalhadas por Módulo
5. Estrutura Técnica de Implementação
6. Fluxo de Validação
7. Exemplos Práticos
8. Casos de Uso
9. Segurança e Validação
10. Migração e Implementação

🎯 Visão Geral

Objetivo

Implementar um **sistema de controle de acesso baseado em funções (RBAC - Role-Based Access Control)** que:

- Restringe funcionalidades conforme o cargo do usuário
- Protege dados sensíveis de visualização não autorizada
- Controla ações de criação, edição e exclusão
- Mantém auditoria de quem fez o quê
- Facilita gestão de permissões por grupos

Princípios

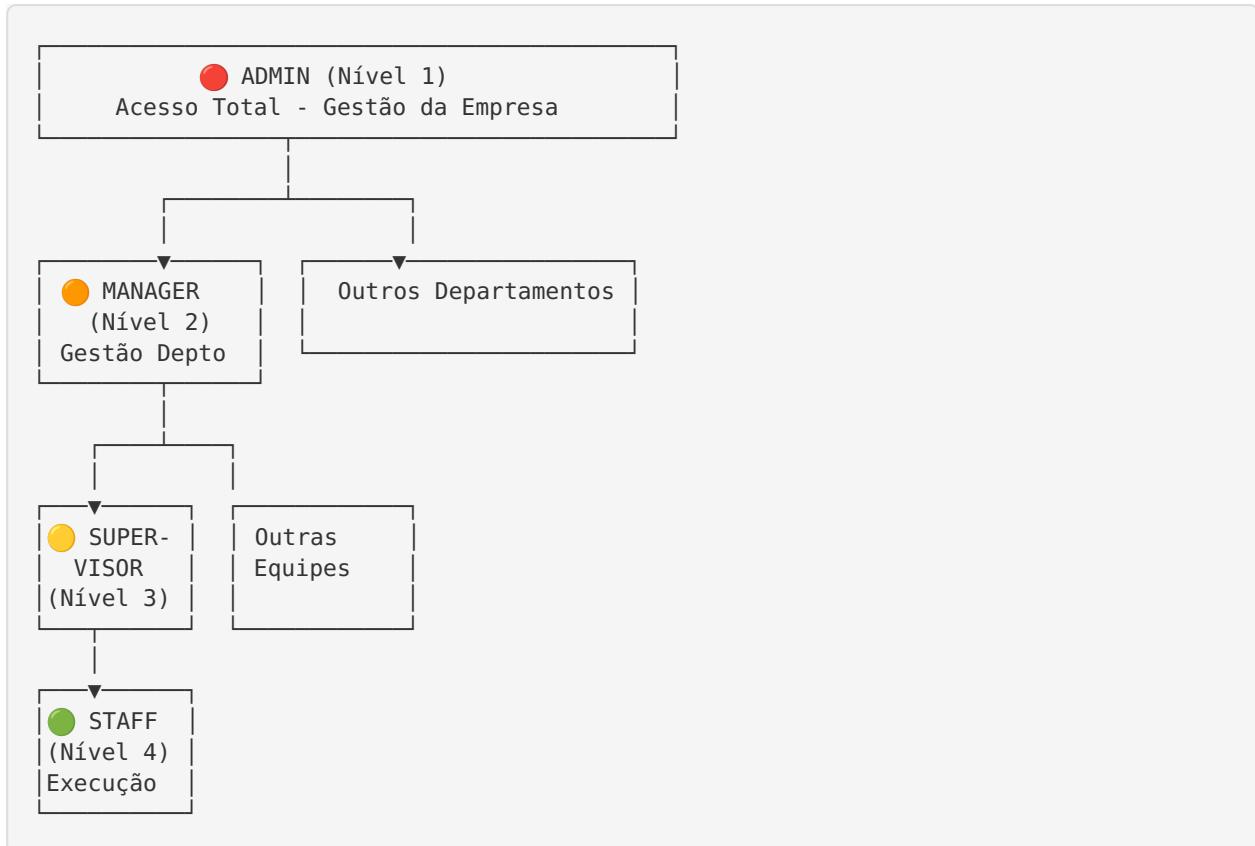
1. **Princípio do Menor Privilégio:** Usuários só têm acesso ao mínimo necessário
2. **Segurança em Camadas:** Validação no frontend E backend
3. **Auditoria:** Todas as ações são registradas
4. **Transparência:** Usuário sabe claramente o que pode/não pode fazer
5. **Hierarquia Clara:** Cargos superiores herdam permissões dos inferiores

XX Estrutura Hierárquica Atual

Cargos Existentes no Sistema

```
enum UserRole {
    ADMIN      // Nível 1 - Administrador da Empresa
    MANAGER    // Nível 2 - Gerente de Departamento
    SUPERVISOR // Nível 3 - Supervisor de Equipe
    STAFF      // Nível 4 - Funcionário/Colaborador
}
```

Hierarquia Visual



Descrição dos Cargos

● ADMIN (Administrador)

Função: Proprietário ou Diretor da Empresa

Escopo: Toda a empresa

Poder: Controle total

Exemplos: CEO, Diretor, Proprietário

● MANAGER (Gerente)

Função: Gestor de Departamento

Escopo: Seu departamento e equipes subordinadas

Poder: Gestão completa do departamento

Exemplos: Gerente de RH, Gerente de Vendas, Gerente de TI

SUPERVISOR (Supervisor)

Função: Líder de Equipe

Escopo: Sua equipe específica

Poder: Coordenação e acompanhamento da equipe

Exemplos: Supervisor de Turno, Líder de Projeto, Coordenador de Equipe

STAFF (Funcionário/Colaborador)

Função: Executor de tarefas

Escopo: Próprio trabalho e interações básicas

Poder: Execução e comunicação limitada

Exemplos: Funcionário, Assistente, Operador

Matriz Completa de Permissões

Legenda

-  **Permitido:** Usuário pode realizar a ação
 -  **Restrito:** Usuário NÃO pode realizar a ação
 -  **Parcial:** Pode realizar apenas em seu escopo (ex: apenas seu departamento)
 -  **Visualizar:** Apenas leitura, sem edição
 -  **Editar:** Pode modificar
 -  **Excluir:** Pode deletar
-

 **MÓDULO: Dashboard**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Ver estatísticas gerais da empresa	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Limitado
Ver gráficos de desempenho	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Pessoal
Ver total de usuários	✓	♦ Só seu depto	♦ Só sua equipe	🔒
Ver total de tarefas	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Próprias
Ver total de turnos	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Próprios
Ver total de departamentos	✓	👁️ Todos	👁️ Todos	👁️ Todos
Quick Actions (criar tarefa/turno/evento)	✓	✓	♦ Limitado	🔒
Exportar relatórios	✓	♦ Só seu depto	🔒	🔒

 **MÓDULO: Usuários (Users)**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Visualizar lista de usuários				
Ver todos os usuários da empresa	✓	♦ Só seu depto	♦ Só sua equipe	👁 Colegas de equipe
Ver detalhes completos de perfil	✓	♦ Só seu depto	♦ Só sua equipe	👁 Básico
Ver informações sensíveis (salário, histórico)	✓	🔒	🔒	🔒
Criar usuários				
Criar usuário ADMIN	✓	🔒	🔒	🔒
Criar usuário MANAGER	✓	🔒	🔒	🔒
Criar usuário SUPERVISOR	✓	✓	🔒	🔒
Criar usuário STAFF	✓	✓	♦ Só sua equipe	🔒
Editar usuários				
Editar ADMIN	✓	🔒	🔒	🔒
Editar MANAGER	✓	🔒	🔒	🔒
Editar SUPERVISOR	✓	📝 Do seu depto	🔒	🔒
Editar STAFF	✓	📝 Do seu depto	📝 Da sua equipe	🔒
Editar próprio perfil	✓	✓	✓	✓
Deletar usuários				

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Deletar ADMIN	Só outro ADMIN			
Deletar MANAGER				
Deletar SUPERVISOR		Do seu depto		
Deletar STAFF		Do seu depto	Da sua equipe	
Outras ações				
Alterar cargo (role) de usuário				
Resetar senha de usuário		Do seu depto	Da sua equipe	
Atribuir usuário a departamento				
Atribuir usuário a equipe		Do seu depto		

 **MÓDULO: Departamentos (Departments)**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Visualizar				
Ver lista de departamentos	✓	👁️ Todos	👁️ Todos	👁️ Só o próprio
Ver detalhes de departamento	✓	👁️ Todos	👁️ Todos	👁️ Só o próprio
Ver estatísticas do departamento	✓	◆ Só o próprio	🔒	🔒
Criar				
Criar novo departamento	✓	🔒	🔒	🔒
Editar				
Editar departamento	✓	📝 Só o próprio	🔒	🔒
Renomear departamento	✓	🔒	🔒	🔒
Deletar				
Deletar departamento	✓	🔒	🔒	🔒
Equipes (Teams)				
Ver equipes do departamento	✓	✓ Do próprio	👁️	👁️ Só a própria
Criar equipe	✓	✓ No próprio depto	🔒	🔒
Editar equipe	✓	📝 Do próprio depto	📝 Só a própria	🔒
Deletar equipe	✓	✗ Do próprio depto	🔒	🔒

 **MÓDULO: Tarefas (Tasks)**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Visualizar				
Ver todas as tarefas da empresa	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Próprias + atribuídas
Ver detalhes de tarefa	✓	♦ Do seu depto	♦ Da sua equipe	👁️ Se for atribuída
Ver tarefas de outros usuários	✓	♦ Do seu depto	♦ Da sua equipe	🔒
Criar				
Criar tarefa para qualquer usuário	✓	♦ Do seu depto	♦ Da sua equipe	🔒
Criar subtarefa	✓	♦ Do seu depto	♦ Da sua equipe	📝 Nas próprias
Editar				
Editar qualquer tarefa	✓	♦ Do seu depto	♦ Da sua equipe	📝 Próprias
Alterar status de tarefa	✓	✓ Do seu depto	✓ Da sua equipe	📝 Próprias
Alterar prioridade	✓	📝 Do seu depto	📝 Da sua equipe	🔒
Alterar prazo (due date)	✓	📝 Do seu depto	📝 Da sua equipe	🔒
Reatribuir tarefa	✓	📝 Do seu depto	📝 Da sua equipe	🔒
Deletar				
Deletar tarefa	✓	✗ Do seu depto	🔒	🔒
Comentários				
Adicionar comentário	✓	✓	✓	✓ Nas atribuídas
Ver comentários	✓	✓ Do seu depto		

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
			Da sua equipe	Nas atribuídas
Deletar comentário		Do seu depto	Próprios	Próprios
Anexos				
Adicionar anexo		Do seu depto	Da sua equipe	Nas atribuídas
Baixar anexo		Do seu depto	Da sua equipe	Nas atribuídas
Deletar anexo		Do seu depto	Próprios	Próprios
Tags				
Criar/editar tags		Do seu depto	Da sua equipe	Nas próprias
Check Items				
Criar/editar check items		Do seu depto	Da sua equipe	Nas próprias
Marcar como concluído		Do seu depto	Da sua equipe	Nas próprias

 **MÓDULO: Turnos (Shifts)**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Visualizar				
Ver todos os turnos da empresa	✓	◆ Só seu depto	◆ Só sua equipe	👁️ Próprios
Ver turnos de outros usuários	✓	◆ Do seu depto	◆ Da sua equipe	🔒
Ver calendário de turnos	✓	◆ Do seu depto	◆ Da sua equipe	👁️ Próprios
Criar				
Criar turno para qualquer usuário	✓	◆ Do seu depto	◆ Da sua equipe	🔒
Criar turno para si mesmo	✓	✓	✓	🔒
Editar				
Editar qualquer turno	✓	◆ Do seu depto	◆ Da sua equipe	🔒
Editar próprio turno	✓	✓	✓	🔒 Apenas solicitar
Alterar horário do turno	✓	📝 Do seu depto	📝 Da sua equipe	🔒
Reatribuir turno	✓	📝 Do seu depto	📝 Da sua equipe	🔒
Deletar				
Deletar turno	✓	✖️ Do seu depto	🔒	🔒
Relatórios				
Ver horas trabalhadas por usuário	✓	◆ Do seu depto	◆ Da sua equipe	👁️ Próprias
Exportar relatório de turnos	✓	◆ Do seu depto	🔒	🔒

 **MÓDULO: Calendário / Eventos (Calendar / Events)**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Visualizar				
Ver todos os eventos da empresa		♦ Só seu depto	♦ Só sua equipe	Públicos + próprios
Ver calendário completo		♦ Do seu depto	♦ Da sua equipe	Próprio
Criar				
Criar evento para empresa		♦ Do seu depto		
Criar evento pessoal				
Criar evento de reunião		Do seu depto	Da sua equipe	
Editar				
Editar qualquer evento		♦ Do seu depto		
Editar próprio evento				
Deletar				
Deletar qualquer evento		♦ Do seu depto		
Deletar próprio evento				
Lembretes				
Configurar lembretes				

💬 MÓDULO: Mensagens (Messages)

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Enviar				
Enviar para qualquer usuário	✓	✓ Do seu depto	✓ Da sua equipe	✓ Colegas
Enviar para ADMIN	✓	✓	✓	✓
Enviar para usuários de outros deptos	✓	✓	♦ Com aprovação	🔒
Enviar mensagens em massa	✓	♦ Do seu depto	🔒	🔒
Visualizar				
Ver mensagens recebidas	✓	✓	✓	✓
Ver mensagens enviadas	✓	✓	✓	✓
Ver mensagens de outros usuários	✓	🔒	🔒	🔒
Gerenciar				
Arquivar mensagem	✓	✓	✓	✓
Deletar mensagem	✓	✓ Próprias	✓ Próprias	✓ Próprias
Criar pastas	✓	✓	✓	✓
Mover para pasta	✓	✓	✓	✓
Anexos				
Enviar anexos	✓	✓	✓	✓
Baixar anexos	✓	✓	✓	✓ Próprias

 **MÓDULO: Chat em Tempo Real**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Conversas				
Iniciar chat com qualquer usuário	✓	✓ Do seu depto	✓ Da sua equipe	♦ Colegas
Ver lista de conversas	✓	✓	✓	✓
Ver status online/offline	✓	✓	✓	✓
Ver “digitando...”	✓	✓	✓	✓
Mensagens				
Enviar mensagem	✓	✓	✓	✓
Deletar própria mensagem	✓	✓	✓	✓
Deletar mensagem de outros	✓	🔒	🔒	🔒
Privacidade				
Bloquear usuário	🔒	🔒	🔒	🔒
Aparecer como offline	✓	✓	✓	✓

 **MÓDULO: Relatórios (Reports)**

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Visualizar				
Ver relatórios gerais da empresa	✓	♦ Só seu depto	🔒	🔒
Ver relatórios de tarefas	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Próprias
Ver relatórios de turnos	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Próprios
Ver produtividade de usuários	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Própria
Ver timeline de tarefas	✓	♦ Só seu depto	♦ Só sua equipe	👁️ Próprias
Exportar				
Exportar PDF de relatórios	✓	♦ Só seu depto	🔒	🔒
Exportar Excel/CSV	✓	♦ Só seu depto	🔒	🔒
Análises				
Ver gráficos avançados	✓	♦ Só seu depto	🔒	🔒
Comparar períodos	✓	♦ Só seu depto	🔒	🔒

MÓDULO: Perfil (Profile)

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Ver próprio perfil	✓	✓	✓	✓
Editar próprio perfil	✓	✓	✓	✓
Alterar própria senha	✓	✓	✓	✓
Alterar foto de perfil	✓	✓	✓	✓
Alterar idioma	✓	✓	✓	✓
Ver histórico de atividades	✓	✓	✓	✓ Limitado
Alterar próprio cargo	🔒	🔒	🔒	🔒
Alterar próprio departamento	🔒	🔒	🔒	🔒

MÓDULO: Notificações (Notifications)

Funcionalidade	ADMIN	MANAGER	SUPERVISOR	STAFF
Receber notificações	✓	✓	✓	✓
Marcar como lida	✓	✓	✓	✓
Deletar notificações	✓	✓	✓	✓
Configurar preferências	✓	✓	✓	✓
Desativar notificações	✓	✓	✓	✓
Ver notificações de outros	✓	🔒	🔒	🔒

Estrutura Técnica de Implementação

1. Camadas de Validação

Frontend (React/Next.js)

Componentes com Controle de Permissões:

```
// lib/permissions.ts

export type Permission = {
  action: 'view' | 'create' | 'edit' | 'delete' | 'manage';
  resource: 'users' | 'departments' | 'tasks' | 'shifts' | 'events' | 'messages' | 'reports';
  scope?: 'own' | 'team' | 'department' | 'company';
};

export type UserRole = 'ADMIN' | 'MANAGER' | 'SUPERVISOR' | 'STAFF';

// Verificar se usuário tem permissão
export function hasPermission(
  userRole: UserRole,
  permission: Permission,
  context?: {
    userId: string;
    targetUserId?: string;
    userDepartmentId?: string;
    targetDepartmentId?: string;
    userTeamId?: string;
    targetTeamId?: string;
  }
): boolean {
  // Implementação da lógica de permissões
  return checkPermissionLogic(userRole, permission, context);
}

// Hook para usar em componentes
export function usePermission(permission: Permission) {
  const { data: session } = useSession();
  const user = session?.user;

  return hasPermission(user?.role as UserRole, permission, {
    userId: user?.id,
    userDepartmentId: user?.departmentId,
    userTeamId: user?.teamId,
  });
}
```

Componente de Exemplo:

```
// components/tasks-content.tsx

export function TasksContent() {
  const canCreateTask = usePermission({ action: 'create', resource: 'tasks' });
  const canDeleteTask = usePermission({ action: 'delete', resource: 'tasks' });

  return (
    <div>
      {canCreateTask && (
        <Button onClick={handleCreateTask}>
          Nova Tarefa
        </Button>
      )}

      {tasks.map(task => (
        <TaskCard key={task.id}>
          {/* ... */}
          {canDeleteTask && (
            <Button onClick={() => handleDelete(task.id)}>
              Deletar
            </Button>
          )}
        </TaskCard>
      ))}
    </div>
  );
}
```

Backend (API Routes)

Middleware de Autorização:

```
// lib/auth-middleware.ts

import { NextRequest, NextResponse } from 'next/server';
import { getServerSession } from 'next-auth';
import { authOptions } from '@/lib/auth';

export async function requireAuth(req: NextRequest) {
  const session = await getServerSession(authOptions);

  if (!session) {
    return NextResponse.json(
      { error: 'Não autorizado' },
      { status: 401 }
    );
  }

  return session;
}

export async function requireRole(
  req: NextRequest,
  allowedRoles: UserRole[]
) {
  const session = await requireAuth(req);

  if (session instanceof NextResponse) return session; // Error response

  if (!allowedRoles.includes(session.user.role)) {
    return NextResponse.json(
      { error: 'Permissão negada' },
      { status: 403 }
    );
  }

  return session;
}

export async function requirePermission(
  req: NextRequest,
  permission: Permission,
  context?: any
) {
  const session = await requireAuth(req);

  if (session instanceof NextResponse) return session;

  if (!hasPermission(session.user.role, permission, context)) {
    return NextResponse.json(
      { error: 'Permissão negada para esta ação' },
      { status: 403 }
    );
  }

  return session;
}
```

API Route Protegida:

```
// app/api/tasks/route.ts

import { NextRequest, NextResponse } from 'next/server';
import { requirePermission } from '@/lib/auth-middleware';
import { prisma } from '@/lib/db';

export async function GET(req: NextRequest) {
  const session = await requirePermission(req, {
    action: 'view',
    resource: 'tasks'
  });

  if (session instanceof NextResponse) return session;

  const user = session.user;

  // Filtrar tarefas baseado no role
  let tasks;

  if (user.role === 'ADMIN') {
    // Admin vê todas as tarefas
    tasks = await prisma.task.findMany({
      where: { companyId: user.companyId }
    });
  } else if (user.role === 'MANAGER') {
    // Manager vê tarefas do seu departamento
    tasks = await prisma.task.findMany({
      where: {
        companyId: user.companyId,
        user: { departmentId: user.departmentId }
      }
    });
  } else if (user.role === 'SUPERVISOR') {
    // Supervisor vê tarefas da sua equipe
    tasks = await prisma.task.findMany({
      where: {
        companyId: user.companyId,
        user: { teamId: user.teamId }
      }
    });
  } else {
    // Staff vê apenas suas próprias tarefas
    tasks = await prisma.task.findMany({
      where: {
        companyId: user.companyId,
        userId: user.id
      }
    });
  }

  return NextResponse.json(tasks);
}

export async function POST(req: NextRequest) {
  const session = await requirePermission(req, {
    action: 'create',
    resource: 'tasks'
  });

  if (session instanceof NextResponse) return session;

  const user = session.user;
```

```

const body = await req.json();

// Validar se pode criar tarefa para o usuário alvo
if (user.role === 'STAFF') {
  return NextResponse.json(
    { error: 'Staff não pode criar tarefas' },
    { status: 403 }
  );
}

// Validar escopo (departamento/equipe)
if (user.role === 'MANAGER') {
  const targetUser = await prisma.user.findUnique({
    where: { id: body.userId }
  });

  if (targetUser?.departmentId !== user.departmentId) {
    return NextResponse.json(
      { error: 'Você só pode criar tarefas para seu departamento' },
      { status: 403 }
    );
  }
}

// Criar tarefa
const task = await prisma.task.create({
  data: {
    ...body,
    companyId: user.companyId
  }
});

return NextResponse.json(task, { status: 201 });
}

```

2. Estrutura de Arquivos

lib/	
└── permissions.ts	# Lógica central de permissões
└── auth-middleware.ts	# Middlewares de autenticação
└── permission-utils.ts	# Utilitários auxiliares
components/	
└── permission-guard.tsx	# Componente HOC para proteger UI
└── role-badge.tsx	# Badge visual do cargo
app/api/	
└── [resource]/	
└── route.ts	# Todas as rotas validam permissões

3. Componentes Auxiliares

Permission Guard (HOC)

```
// components/permission-guard.tsx

import { usePermission } from '@/lib/permissions';

interface PermissionGuardProps {
  permission: Permission;
  fallback?: React.ReactNode;
  children: React.ReactNode;
}

export function PermissionGuard({
  permission,
  fallback = null,
  children
}: PermissionGuardProps) {
  const hasAccess = usePermission(permission);

  if (!hasAccess) {
    return <>{fallback}</>;
  }

  return <>{children}</>;
}

// Uso:
<PermissionGuard
  permission={{ action: 'create', resource: 'users' }}
  fallback=<p>Você não tem permissão para criar usuários</p>
>
  <Button>Criar Usuário</Button>
</PermissionGuard>
```

Role Badge

```
// components/role-badge.tsx

const roleConfig = {
  ADMIN: { label: 'Administrador', color: 'red', icon: '🔴' },
  MANAGER: { label: 'Gerente', color: 'orange', icon: '🟡' },
  SUPERVISOR: { label: 'Supervisor', color: 'yellow', icon: '🟡' },
  STAFF: { label: 'Funcionário', color: 'green', icon: '🟢' },
};

export function RoleBadge({ role }: { role: UserRole }) {
  const config = roleConfig[role];

  return (
    <Badge variant={config.color}>
      {config.icon} {config.label}
    </Badge>
  );
}
```

 **Fluxo de Validação****Fluxo Completo de uma Ação**





Exemplos Práticos

Exemplo 1: Staff Tentando Criar Tarefa

Cenário: João é STAFF e tenta clicar no botão “Criar Tarefa”

Fluxo:

1. **Frontend:** Hook `usePermission({ action: 'create', resource: 'tasks' })` retorna `false` para STAFF
 2. **UI:** Botão “Criar Tarefa” não aparece na tela de João
 3. **Resultado:** João nem vê a opção de criar tarefa
-

Exemplo 2: Manager Tentando Editar Usuário de Outro Departamento

Cenário: Maria é MANAGER do Departamento de Vendas e tenta editar João do Departamento de TI

Fluxo:

1. **Frontend:** Hook permite (Manager pode editar usuários)
 2. **API:** Maria envia `PUT /api/users/joao-id`
 3. **Backend Middleware:** Valida que Maria é MANAGER ✓
 4. **Backend Validação de Escopo:**
 - Maria está no Depto de Vendas
 - João está no Depto de TI
 - ✗ Departamentos diferentes!
 5. **Resposta:** `403 Forbidden` - "Você só pode editar usuários do seu departamento"
 6. **UI:** Mostra mensagem de erro para Maria
-

Exemplo 3: Supervisor Visualizando Relatórios

Cenário: Carlos é SUPERVISOR da Equipe A e acessa a página de Relatórios

Fluxo:

1. **Frontend:** Página de relatórios carrega
 2. **API:** `GET /api/reports/overview`
 3. **Backend:** Detecta que Carlos é SUPERVISOR
 4. **Filtro Aplicado:**
 - Admin veria: Toda a empresa
 - Manager veria: Todo o departamento
 - Supervisor vê: **Apenas Equipe A**
 - Staff: Sem acesso
 5. **Resposta:** Relatórios filtrados apenas da Equipe A
 6. **UI:** Carlos vê dados apenas da sua equipe
-

Exemplo 4: Admin com Acesso Total

Cenário: Ana é ADMIN e acessa a lista de usuários

Fluxo:

1. **API:** GET /api/users
 2. **Backend:** Detecta role ADMIN
 3. **Sem Filtros:** Admin vê **todos** os usuários da empresa
 4. **UI:** Ana vê lista completa com opções de:
 - Criar qualquer cargo
 - Editar qualquer usuário
 - Deletar usuários
 - Alterar cargos
 - Resetar senhas
-

🎯 Casos de Uso Detalhados

Caso de Uso 1: Criação de Tarefa

Regras:

- **ADMIN:** Pode criar tarefa para qualquer usuário da empresa
- **MANAGER:** Pode criar tarefa apenas para usuários do seu departamento
- **SUPERVISOR:** Pode criar tarefa apenas para usuários da sua equipe
- **STAFF:** NÃO pode criar tarefas

Fluxo de Validação:

```
// Backend: POST /api/tasks

async function createTask(session, body) {
  const { role, id: userId, departmentId, teamId, companyId } = session.user;
  const { assignedUserId } = body;

  // 1. Validar se tem permissão básica de criar
  if (role === 'STAFF') {
    throw new Error('Staff não pode criar tarefas');
  }

  // 2. Buscar usuário alvo
  const targetUser = await prisma.user.findUnique({
    where: { id: assignedUserId }
  });

  if (!targetUser) {
    throw new Error('Usuário não encontrado');
  }

  // 3. Validar escopo conforme role
  if (role === 'MANAGER') {
    if (targetUser.departmentId !== departmentId) {
      throw new Error('Você só pode criar tarefas para seu departamento');
    }
  }

  if (role === 'SUPERVISOR') {
    if (targetUser.teamId !== teamId) {
      throw new Error('Você só pode criar tarefas para sua equipe');
    }
  }

  // 4. Criar tarefa
  const task = await prisma.task.create({
    data: {
      ...body,
      userId: assignedUserId,
      companyId: companyId
    }
  });

  // 5. Criar notificação para o usuário atribuído
  await prisma.notification.create({
    data: {
      userId: assignedUserId,
      type: 'TASK',
      title: 'Nova tarefa atribuída',
      message: `Você recebeu uma nova tarefa: ${body.title}`,
      relatedId: task.id
    }
  });

  return task;
}
```

Caso de Uso 2: Visualização de Dashboard

Regras:

- **ADMIN:** Vê estatísticas de toda a empresa
- **MANAGER:** Vê estatísticas apenas do seu departamento
- **SUPERVISOR:** Vê estatísticas apenas da sua equipe
- **STAFF:** Vê estatísticas pessoais limitadas

Filtros Aplicados:

```
// Backend: GET /api/dashboard/stats

async function getDashboardStats(session) {
  const { role, departmentId, teamId, id: userId, companyId } = session.user;

  let filters = { companyId };

  // Aplicar filtros conforme role
  switch (role) {
    case 'ADMIN':
      // Sem filtro adicional - vê tudo
      break;

    case 'MANAGER':
      filters.departmentId = departmentId;
      break;

    case 'SUPERVISOR':
      filters.teamId = teamId;
      break;

    case 'STAFF':
      filters.userId = userId;
      break;
  }

  // Buscar estatísticas com filtros aplicados
  const stats = {
    totalUsers: await prisma.user.count({ where: filters }),
    totalTasks: await prisma.task.count({
      where: {
        companyId,
        user: filters.departmentId ? { departmentId: filters.departmentId } :
          filters.teamId ? { teamId: filters.teamId } :
            filters.userId ? { id: filters.userId } : undefined
      }
    }),
    completedTasks: await prisma.task.count({
      where: {
        companyId,
        status: 'COMPLETED',
        user: /* mesma lógica */
      }
    }),
    // ... outras estatísticas
  };

  return stats;
}
```

Caso de Uso 3: Exclusão de Usuário

Regras:

- **ADMIN:** Pode deletar qualquer usuário (exceto outro ADMIN sem ser autorizado)
- **MANAGER:** Pode deletar SUPERVISOR e STAFF do seu departamento
- **SUPERVISOR:** Pode deletar STAFF da sua equipe
- **STAFF:** NÃO pode deletar ninguém

Validação:

```
// Backend: DELETE /api/users/[id]

async function deleteUser(session, targetUserId) {
  const { role, id: userId, departmentId, teamId } = session.user;

  // 1. Buscar usuário alvo
  const targetUser = await prisma.user.findUnique({
    where: { id: targetUserId }
  });

  if (!targetUser) {
    throw new Error('Usuário não encontrado');
  }

  // 2. Validações por role
  if (role === 'STAFF') {
    throw new Error('Staff não pode deletar usuários');
  }

  if (role === 'SUPERVISOR') {
    // Só pode deletar STAFF da sua equipe
    if (targetUser.role !== 'STAFF') {
      throw new Error('Supervisor só pode deletar Staff');
    }
    if (targetUser.teamId !== teamId) {
      throw new Error('Usuário não pertence à sua equipe');
    }
  }

  if (role === 'MANAGER') {
    // Só pode deletar SUPERVISOR e STAFF do seu departamento
    if (targetUser.role === 'ADMIN' || targetUser.role === 'MANAGER') {
      throw new Error('Manager não pode deletar Admin ou Manager');
    }
    if (targetUser.departmentId !== departmentId) {
      throw new Error('Usuário não pertence ao seu departamento');
    }
  }

  if (role === 'ADMIN') {
    // Admin pode deletar, mas deve validar se não está deletando ele mesmo
    if (targetUser.id === userId) {
      throw new Error('Você não pode deletar sua própria conta');
    }

    // Se tentar deletar outro ADMIN, validar permissão especial
    if (targetUser.role === 'ADMIN') {
      // Implementar lógica de permissão especial ou segundo fator
    }
  }

  // 3. Deletar usuário
  await prisma.user.delete({
    where: { id: targetUserId }
  });

  return { success: true };
}
```

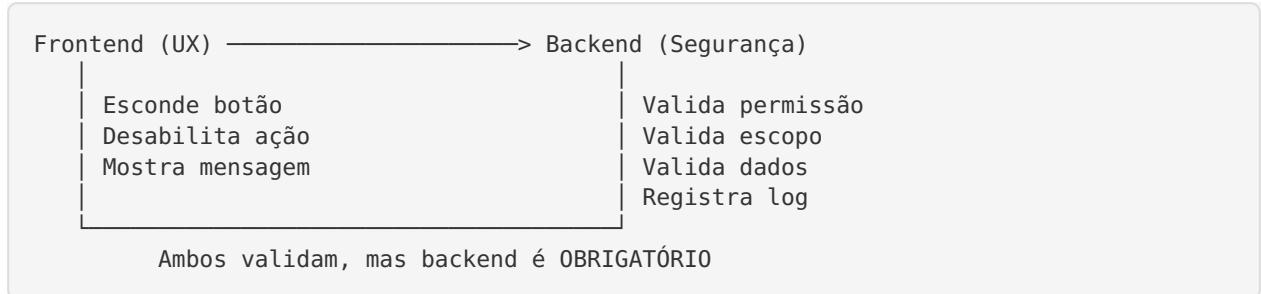
Segurança e Validação

Princípios de Segurança

1. Nunca Confiar no Frontend

- Sempre validar no backend
- Frontend é apenas para UX (esconder botões)
- Usuário pode manipular requisições

2. Validação em Camadas



3. Logs de Auditoria

```
// Registrar todas as ações sensíveis

await prisma.auditLog.create({
  data: {
    userId: session.user.id,
    action: 'DELETE_USER',
    resource: 'users',
    resourceId: targetUserId,
    metadata: {
      targetUserEmail: targetUser.email,
      targetUserRole: targetUser.role
    },
    timestamp: new Date()
  }
});
```

4. Mensagens de Erro Seguras

 Ruim:

```
{
  "error": "Usuário 'joao@teste.com' não encontrado no departamento 'TI'"
}
```

 Bom:

```
{
  "error": "Você não tem permissão para realizar esta ação"
}
```

5. Rate Limiting

```
// Limitar tentativas de ações sensíveis
import rateLimit from 'express-rate-limit';

const deleteUserLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutos
  max: 5, // Máximo 5 delegações por janela
  message: 'Muitas tentativas. Tente novamente mais tarde.'
});
```



Migração e Implementação

Fases de Implementação

Fase 1: Preparação (1-2 dias)

1. Criar arquivo `lib/permissions.ts` com lógica base
2. Criar `lib/auth-middleware.ts` com middlewares
3. Criar componentes auxiliares (`PermissionGuard`, `RoleBadge`)
4. Documentar permissões (este documento)

Fase 2: Backend (3-4 dias)

1. Adicionar validações em todas as API routes
2. Implementar filtros de escopo por role
3. Criar testes automatizados para permissões
4. Adicionar logs de auditoria

Fase 3: Frontend (2-3 dias)

1. Atualizar componentes com `usePermission()`
2. Adicionar `PermissionGuard` onde necessário
3. Esconder/desabilitar elementos conforme permissões
4. Adicionar mensagens informativas

Fase 4: Testes (2-3 dias)

1. Testar cada cargo em cada módulo
2. Tentar burlar permissões (security testing)
3. Validar mensagens de erro
4. Validar UX de cada role

Fase 5: Deploy e Monitoramento (1 dia)

1. Deploy para produção
2. Monitorar logs de erro
3. Ajustar permissões conforme feedback
4. Documentar para usuários finais

Checklist de Validação

Antes de considerar implementação concluída:

Backend:

- [] Todas as rotas validam autenticação
- [] Todas as rotas validam permissões
- [] Filtros de escopo funcionando
- [] Logs de auditoria implementados
- [] Testes automatizados passando
- [] Mensagens de erro padronizadas

Frontend:

- [] Botões escondidos quando sem permissão
- [] Campos desabilitados quando sem permissão
- [] Mensagens informativas claras
- [] Loading states adequados
- [] Tratamento de erros 403
- [] Tooltips explicativos

UX:

- [] Admin consegue fazer tudo
- [] Manager limitado a seu departamento
- [] Supervisor limitado a sua equipe
- [] Staff vê apenas informações relevantes
- [] Mensagens de erro amigáveis
- [] Sem opções “quebradas” na UI

Segurança:

- [] Validação dupla (frontend + backend)
 - [] Tentativas de bypass falham
 - [] Dados sensíveis protegidos
 - [] Rate limiting implementado
 - [] Logs de auditoria funcionando
-

Resumo Executivo

Para Implementação

O que você PRECISA fazer:

1. **Criar arquivo de permissões** (`lib/permissions.ts`)
 - Função `hasPermission()`
 - Hook `usePermission()`
 - Tipos TypeScript

2. **Criar middlewares de backend** (`lib/auth-middleware.ts`)
 - `requireAuth()`
 - `requireRole()`
 - `requirePermission()`

3. **Adicionar validações em TODAS as API routes**

- Exemplo: `POST /api/tasks` , `DELETE /api/users` , etc.
- Validar role + escopo

4. **Atualizar componentes do frontend**

- Usar `usePermission()` para mostrar/esconder elementos
- Adicionar `PermissionGuard` onde apropriado

5. **Testar exaustivamente**

- Cada role em cada módulo
 - Tentar burlar permissões
-

Permissões Rápidas por Cargo

 **ADMIN:**

- Vê e gerencia TUDO na empresa
- Sem restrições
- Controle total

 **MANAGER:**

- Vê e gerencia SEU DEPARTAMENTO
- Pode criar/editar/deletar usuários do departamento
- Pode criar tarefas/turnos para o departamento
- Acessa relatórios do departamento

 **SUPERVISOR:**

- Vê e coordena SUA EQUIPE
- Pode criar tarefas/turnos para a equipe
- Pode editar STAFF da equipe
- Acessa estatísticas da equipe

 **STAFF:**

- Vê APENAS suas próprias informações
 - Executa tarefas atribuídas
 - Muda status das próprias tarefas
 - Conversa com colegas via mensagens/chat
 - SEM poder de criar/deletar nada
-



Próximos Passos

Depois de Aprovação deste Plano:

1. **Você aprova este plano?**

- Alguma permissão precisa ser ajustada?
- Algum cargo precisa ter mais/menos acesso?

2. **Eu implemento tudo:**

- Criar arquivos de permissões
- Adicionar validações em todas as APIs

- Atualizar todos os componentes
- Testar tudo

3.  Você testa:

- Login como cada cargo
- Tentar acessar áreas restritas
- Validar se comportamento está correto

4.  Deploy:

- Checkpoint com sistema de permissões completo
 - Documentação para usuários finais
-

?

Perguntas para Você

Antes de implementar, preciso que você confirme:

1. As permissões definidas fazem sentido para seu negócio?

- Exemplo: STAFF realmente não deve criar tarefas?
- Ou alguns STAFF específicos deveriam poder?

2. Você quer hierarquia rígida ou flexível?

- Rígida: Regras fixas por cargo
- Flexível: Permissões customizáveis por usuário

3. Você quer permissões de “visualização parcial”?

- Exemplo: STAFF ver lista de tarefas da equipe (só visualizar, não editar)
- Ou STAFF só vê suas próprias tarefas?

4. Departamentos e Equipes são obrigatórios?

- Todo usuário deve estar em um departamento/equipe?
- Ou pode existir usuário “solto”?

5. Você quer sistema de “solicitações”?

- Exemplo: STAFF solicita troca de turno → SUPERVISOR aprova
 - Ou isso fica para uma feature futura?
-

Glossário

- **RBAC:** Role-Based Access Control (Controle de Acesso Baseado em Função)
 - **Permissão:** Direito de realizar uma ação específica
 - **Escopo:** Limite de atuação (próprio, equipe, departamento, empresa)
 - **Middleware:** Camada de validação antes de executar ação
 - **Auditoria:** Registro de quem fez o quê e quando
 - **Rate Limiting:** Limitação de número de tentativas em período de tempo
-

 **Versão:** 1.0 - Planejamento Inicial

 **Próxima Revisão:** Após feedback e aprovação

 **Pronto para implementar assim que você aprovar!**