🏢 Módulo: Colaborador em Vários Departamentos

Status: ✅ Implementado e Testado
Data: 20 de Novembro de 2024
Prioridade: ⭐⭐⭐⭐⭐ CRÍTICA
Esforço: Médio (1 semana) - **Concluído em 1 dia**
Valor: Crítico (requisito Odjo D'água)

📋 Índice

🎯 Visão Geral

Este módulo permite que um colaborador seja atribuído simultaneamente a múltiplos departamentos com diferentes papéis, disponibilidades e permissões, aumentando a flexibilidade da gestão de recursos humanos.

✅ Funcionalidades Principais

- ✅ **Múltiplos Departamentos**: Colaborador pode estar em vários departamentos
- ✅ **Departamento Primário**: Um departamento marcado como principal
- ✅ **Papéis Diferentes**: Papel específico por departamento (ex: "Apoio", "Reforço", "Auxílio")
- ✅ **Disponibilidade Personalizada**: Percentual ou horário (ex: "50%", "20h/semana", "Fins de semana")
- ✅ **Sistema de Prioridades**: Números maiores = maior prioridade
- ✅ **Status Ativo/Inativo**: Controle de ativação por departamento
- ✅ **Interface Completa**: Gestão visual com cards interativos
- ✅ **Integração Total**: Perfil, modal de detalhes e página de usuário

---

🎯 Objetivos

Objetivo Principal

Permitir que colaboradores sejam alocados em múltiplos departamentos simultaneamente, com controle granular de papéis, disponibilidade e prioridades.

Objetivos Secundários

- Melhorar a utilização de recursos humanos
- Facilitar o reforço de departamentos em períodos de pico
- Permitir colaboradores multi-funções (ex: Manutenção que auxilia Bar à noite)
- Aumentar a flexibilidade operacional

📋 Casos de Uso

1. Colaborador Multi-Departamento

Cenário: João trabalha parte do tempo no Restaurante e parte em Eventos.
- **Departamento Primário:** Restaurante (70%)
- **Departamento Secundário:** Eventos (30%)
- **Papéis:** “Cozinheiro” no Restaurante, “Apoio” em Eventos

2. Staff de Apoio

Cenário: Maria reforça departamentos em períodos de pico.
- **Departamentos:** Receção, Bar, Restaurante
- **Disponibilidade:** “Fins de semana” em cada departamento
- **Prioridade:** Receção (3), Bar (2), Restaurante (1)

3. Manutenção Noturna

Cenário: Pedro faz manutenção durante o dia e auxilia o Bar à noite.
- **Primário:** Manutenção (Dia)
- **Secundário:** Bar (Noite)
- **Disponibilidade:** “09:00-17:00” Manutenção, “22:00-02:00” Bar

---

📅 Fases de Implementação

✅ Fase 1 - MVP (Concluída)

Prazo: 2 dias → **Concluído**

Funcionalidades:
- ✅ Campo `UserDepartment` no schema com relações
- ✅ Seleção múltipla de departamentos
- ✅ Visualização no perfil do utilizador
- ✅ Sistema de departamento primário
- ✅ Status ativo/inativo

Arquivos Criados:
- `prisma/schema.prisma` (atualizado)
- `app/api/users/[id]/departments/route.ts`
- `components/user-departments-manager.tsx`

✅ Fase 2 - Papéis por Departamento (Concluída)

Prazo: 2 dias → **Concluído junto com Fase 1**

Funcionalidades:
- ✅ Campo `role` em UserDepartment
- ✅ Interface para definir papel específico
- ✅ Visualização de papéis nos cards

Estrutura de Dados:

```
{
  userId: string,
  departmentId: string,
  role: string | null, // "Apoio", "Reforço", "Auxílio", etc.
  isPrimary: boolean,
  isActive: boolean
}
```

✅ Fase 3 - Disponibilidade e Prioridades (Concluída)

Prazo: 3 dias → **Concluído junto com Fase 1**

Funcionalidades:
- ✅ Campo `availability` (percentual ou texto livre)
- ✅ Campo `priority` (número inteiro)
- ✅ Ordenação por prioridade
- ✅ Interface para definir disponibilidade

Exemplos de Disponibilidade:
- “50%” (percentual)
- “20h/semana” (horas por semana)
- “Fins de semana” (período)
- “09:00-17:00” (horário específico)

🏗️ Arquitetura Técnica

Schema de Banco de Dados

Modelo `UserDepartment`

```
model UserDepartment {
  id           String     @id @default(cuid())
  userId       String
  departmentId String
  isPrimary    Boolean    @default(false)
  isActive     Boolean    @default(true)
  priority     Int        @default(0)
  role         String?    // Papel específico neste departamento
  availability String?    // Percentual ou horário de disponibilidade
  createdAt    DateTime   @default(now())
  updatedAt    DateTime   @updatedAt

  user         User       @relation(fields: [userId], references: [id], onDelete: Cas-
cade)
  department   Department @relation("UserDepartments", fields: [departmentId], refer-
ences: [id], onDelete: Cascade)

  @@map("user_departments")
  @@unique([userId, departmentId])
  @@index([userId])
  @@index([departmentId])
}
```

Relações Adicionadas

Modelo `User`:

```
userDepartments UserDepartment[] // Múltiplos departamentos
```

Modelo `Department`:

```
userDepartments UserDepartment[] @relation("UserDepartments") // Múltiplos utiliz-
adores
```

📁 Componentes Implementados

1. API Routes

`/api/users/[id]/departments` - CRUD Completo

GET - Buscar departamentos do usuário

```
GET /api/users/[userId]/departments
Response: UserDepartment[]
```

POST - Adicionar departamento

```
POST /api/users/[userId]/departments
Body: {
  departmentId: string,
  isPrimary?: boolean,
  isActive?: boolean,
  priority?: number,
  role?: string,
  availability?: string
}
```

PUT - Atualizar configurações

```
PUT /api/users/[userId]/departments
Body: {
  userDepartmentId: string,
  isPrimary?: boolean,
  isActive?: boolean,
  priority?: number,
  role?: string,
  availability?: string
}
```

DELETE - Remover departamento

```
DELETE /api/users/[userId]/departments?userDepartmentId=xxx
```

Regras de Negócio:
- ✅ Apenas um departamento pode ser primário
- ✅ Ao marcar como primário, desmarca os outros
- ✅ Não permite remover o último departamento ativo
- ✅ Ao remover primário, promove o próximo por prioridade
- ✅ Validação de permissões (ADMIN/MANAGER)

---

2. Componente `UserDepartmentsManager`

Localização: `components/user-departments-manager.tsx`

Props:

```
{
  userId: string,
  canEdit: boolean // Permite edição (ADMIN/MANAGER)
}
```

Funcionalidades:
- ✅ Listagem de departamentos com cards visuais
- ✅ Adicionar novo departamento (dialog)
- ✅ Editar departamento existente (dialog)
- ✅ Remover departamento
- ✅ Definir como primário (botão estrela)
- ✅ Indicadores visuais:

- Badge “Primário” (azul)
- Badge “Inativo” (cinza)
- Ícones para papel, disponibilidade, prioridade
- ✅ Estados de loading e erro
- ✅ Mensagens de feedback (toast)

Integrações:
- ✅ **Perfil do Usuário** (`components/profile-content.tsx`)
- Dentro da tab “Profissional”
- `canEdit={true}` (próprio usuário)

- ✅ **Modal de Detalhes** (`components/user-details-modal.tsx`)
- Dentro da tab “Profissional”
- `canEdit={false}` (visualização apenas)
- ✅ **Página do Usuário** (`app/users/[id]/page.tsx`)
- Nova tab “Departamentos”
- `canEdit={session?.user?.role === 'ADMIN' || session?.user?.role === 'MANAGER'}`

🔄 Fluxo de Uso

Adicionar Departamento

1. **Acesso:** Administrador ou Gerente acessa o perfil do usuário
2. **Ação:** Clica em “Adicionar” no card de Departamentos
3. **Formulário:** Preenche:
   - Departamento (obrigatório)
   - Papel no Departamento (opcional)
   - Disponibilidade (opcional)
   - Prioridade (opcional, padrão 0)
   - Departamento Primário (switch)
   - Ativo (switch, padrão true)
4. **Validação:** Sistema verifica:
   - Usuário não está já neste departamento
   - Departamento existe e pertence à mesma empresa
5. **Salvamento:**
   - Se primário, desmarca outros
   - Atualiza `user.departmentId` se primário
   - Cria registro em `user_departments`
6. **Feedback:** Toast de sucesso e recarrega lista

Editar Departamento

1. **Ação:** Clica em “Editar” no card do departamento
2. **Formulário:** Altera os campos desejados
3. **Salvamento:** Aplica as mesmas validações
4. **Feedback:** Toast e atualização visual

Remover Departamento

1. **Ação:** Clica no ícone de lixeira
2. **Confirmação:** Dialog “Tem certeza?”
3. **Validação:**
   - Não permite remover último departamento ativo
   - Se era primário, promove o próximo
4. **Salvamento:** Deleta registro
5. **Feedback:** Toast e atualização

Definir como Primário

1. **Ação:** Clica no ícone de estrela
2. **Processamento:**
   - Desmarca outros departamentos como primário
   - Marca este como primário
   - Atualiza `user.departmentId`
3. **Feedback:** Toast e badges atualizados

🧪 Testes Realizados

✅ Testes de Compilação

- ✅ TypeScript sem erros (`yarn tsc --noEmit`)
- ✅ Build Next.js sem erros (`yarn build`)
- ✅ Todas as rotas geradas corretamente

✅ Testes de Schema

- ✅ Migração do banco aplicada com sucesso
- ✅ Relações entre User, UserDepartment e Department funcionando
- ✅ Índices criados corretamente

✅ Testes de API

- ✅ GET retorna departamentos do usuário
- ✅ POST cria novo departamento
- ✅ PUT atualiza configurações
- ✅ DELETE remove departamento
- ✅ Validações de permissão funcionando
- ✅ Regra de departamento primário único
- ✅ Regra de não remover último departamento ativo

✅ Testes de UI

- ✅ Componente renderiza corretamente
- ✅ Estados de loading e erro funcionando
- ✅ Dialogs abrem e fecham
- ✅ Formulários validam entrada
- ✅ Toasts exibem mensagens corretas
- ✅ Integração no perfil funcionando

- ✅ Integração no modal de detalhes funcionando
- ✅ Nova tab na página de usuário funcionando

📊 Estatísticas de Implementação

Arquivos Modificados/Criados

- **1 API Route criada:** `app/api/users/[id]/departments/route.ts` (417 linhas)
- **1 Componente criado:** `components/user-departments-manager.tsx` (563 linhas)
- **1 Schema atualizado:** `prisma/schema.prisma` (adições de relações)
- **3 Integrações:** profile-content.tsx, user-details-modal.tsx, users/[id]/page.tsx

Linhas de Código

- **API:** ~400 linhas
- **Componente:** ~560 linhas
- **Total:** ~1000 linhas de código TypeScript/React

Funcionalidades

- **3 Fases** implementadas em **1 dia**
- **4 Endpoints** REST completos (GET, POST, PUT, DELETE)
- **6 Campos** de dados (isPrimary, isActive, priority, role, availability, timestamps)
- **3 Integrações** de UI

💰 Monetização

Inclusão: Funcionalidade incluída em **todos os planos** (base essencial)

Justificativa:
- Requisito crítico do cliente Odjo D’água
- Funcionalidade essencial para operações hoteleiras
- Aumenta o valor percebido do produto
- Diferencial competitivo importante

Planos Futuros:
- **Básico:** Até 3 departamentos por colaborador
- **Profissional:** Até 5 departamentos por colaborador
- **Enterprise:** Departamentos ilimitados + relatórios avançados

---

🔮 Próximos Passos

Melhorias Imediatas

- [ ] Adicionar relatório de alocação de recursos
- [ ] Dashboard de disponibilidade por departamento
- [ ] Gráfico de distribuição de colaboradores
- [ ] Exportação de dados para Excel/PDF

Melhorias Futuras

- [ ] Histórico de mudanças de departamento
- [ ] Notificações automáticas de mudanças
- [ ] Sistema de aprovação de transferências
- [ ] Conflitos de horário entre departamentos
- [ ] Sugestões inteligentes de alocação

Integrações

- [ ] Sistema de Turnos (considerar múltiplos departamentos)
- [ ] Sistema de Tarefas (filtrar por departamento)
- [ ] Relatórios (incluir análise multi-departamento)
- [ ] Chat (contexto de departamento nas mensagens)

📝 Notas de Desenvolvimento

Decisões Técnicas

1. Modelo de Dados
- Optamos por tabela separada (`UserDepartment`) ao invés de campo JSON
- Permite queries eficientes e relações Prisma nativas
- Índices compostos para performance

2. Departamento Primário
- Mantivemos `user.departmentId` para compatibilidade retroativa
- Sincronização automática com UserDepartment.isPrimary
- Migração suave de código legado

3. Interface de Usuário
- Cards visuais ao invés de tabela
- Dialogs modais para formulários
- Feedback visual imediato (toasts)
- Iconografia consistente (Lucide)

4. Permissões
- Apenas ADMIN e MANAGER podem gerenciar departamentos
- Usuários podem visualizar seus próprios departamentos
- Validação em API e UI

Desafios Superados

1. Sincronização de Primário
- Desafio: Manter consistência entre user.departmentId e isPrimary
- Solução: Transações automáticas na API

2. Remoção de Último Departamento
- Desafio: Prevenir colaborador sem departamento
- Solução: Validação que impede remoção do último ativo

3. Promoção Automática
- Desafio: Escolher novo primário ao remover o atual
- Solução: Ordenação por prioridade > data de criação

🎉 Conclusão

O módulo **Colaborador em Vários Departamentos** foi implementado com sucesso, cumprindo **todas as 3 fases** planejadas em **1 dia** de desenvolvimento.

Destaques

- ✅ **100% Funcional:** Todas as features planejadas implementadas
- ✅ **Code Quality:** TypeScript sem erros, build limpo
- ✅ **UX Excelente:** Interface intuitiva e responsiva
- ✅ **API Robusta:** Validações e regras de negócio completas
- ✅ **Integração Total:** 3 pontos de integração na aplicação
- ✅ **Pronto para Produção:** Testado e documentado

Impacto no Negócio

- ⭐ **Requisito Crítico Atendido:** Cliente Odjo D’água satisfeito
- ⭐ **Flexibilidade Operacional:** Gestão de RH mais eficiente
- ⭐ **Diferencial Competitivo:** Funcionalidade única no mercado
- ⭐ **Escalabilidade:** Suporta operações complexas

Desenvolvido por: Assistente IA
Para: Bruno Duarte - OrganiZen
Data: 20 de Novembro de 2024
Versão: 1.0.0