

Módulo IV: Grafo Social e Lógica de Streak - Implementado

Visão Geral







O **Módulo IV** adiciona funcionalidades sociais complexas inspiradas no YouVersion, incluindo sistema de streak para gamificação e planos de leitura compartilhados com amigos.

1. Sistema de Streak (Gamificação)

Funcionalidades Implementadas

- **Cálculo Automático de Streak:** Dias consecutivos de conclusão do fluxo “Respira & Ore”
- **Atualização Backend:** Streak atualizado apenas após completar o fluxo devocional completo
- **Campos no User Model:**
 - `currentStreak` : Dias consecutivos atuais
 - `longestStreak` : Recorde de dias consecutivos
 - `lastActivityDate` : Data da última atividade
 - `totalDevocionais` : Total de devocionais concluídos

Arquivos Criados/Modificados

-  `/lib/streak.ts` - Lógica de cálculo e atualização de streak
-  `/app/api/streak/route.ts` - API para consultar streak do usuário
-  `/app/api/journals/route.ts` - Modificado para integrar atualização de streak
-  `/components/streak-badge.tsx` - Componente visual do streak
-  `/components/navbar.tsx` - Navbar atualizada com badge de streak
-  `prisma/schema.prisma` - Schema atualizado com campos de streak

Como Funciona

1. **Conclusão do Devocional:** Usuário completa o fluxo “Respira & Ore”
2. **Registro no Journal:** Sistema cria entrada no journal com gratidão, insight e humor
3. **Atualização Automática:** Função `updateUserStreak()` é chamada automaticamente
4. **Cálculo Inteligente:**
 - Se for o mesmo dia, não atualiza (previne múltiplas contagens)
 - Se for dia consecutivo, incrementa o streak
 - Se quebrou o streak (passou mais de 1 dia), reinicia para 1
 - Atualiza o recorde se necessário

API Endpoints

```
GET /api/streak
// Retorna: { currentStreak, longestStreak, totalDevocionais, isActive, lastActivity-
Date }
```

```
POST /api/journals
// Cria journal E atualiza streak automaticamente
// Retorna: { journal, streak: { currentStreak, longestStreak, isNewRecord } }
```

2. Sistema de Amizades





Funcionalidades Implementadas

- **Busca de Usuários:** Buscar por nome ou email
- **Pedidos de Amizade:** Enviar, aceitar e recusar
- **Status de Amizade:** Visualizar amigos, pendências e pedidos enviados
- **Visualização de Progresso:** Ver streak e total de devocionais dos amigos

Modelos do Banco de Dados

```
model Friendship {
  id          String   @id @default(cuid())
  senderId    String
  receiverId   String
  status       String   // 'pending', 'accepted', 'rejected', 'blocked'
  createdAt    DateTime @default(now())
  updatedAt    DateTime @updatedAt
}
```

Arquivos Criados

-  /app/amigos/page.tsx - Página de gerenciamento de amigos
-  /app/api/friends/route.ts - GET (listar) e POST (enviar pedido)
-  /app/api/friends/[id]/route.ts - PATCH (aceitar/rejeitar) e DELETE (remover)
-  /app/api/users/search/route.ts - Buscar usuários

API Endpoints

```
// Listar amigos
GET /api/friends?type=accepted // Amigos aceitos
GET /api/friends?type=pending  // Pedidos recebidos
GET /api/friends?type=sent      // Pedidos enviados

// Enviar pedido
POST /api/friends
Body: { receiverId: string }

// Responder pedido
PATCH /api/friends/:id
Body: { action: 'accept' | 'reject' }

// Remover amizade
DELETE /api/friends/:id

// Buscar usuários
GET /api/users/search?q=query
```

3. Planos com Amigos (Social)

Funcionalidades Implementadas

- **Criação de Planos:** Criar planos de leitura com múltiplas etapas (dias)
- **Planos Públicos e Privados:** Controle de visibilidade
- **Progresso Individual:** Cada membro marca seu próprio progresso
- **Sistema de Comentários:** Comentar em cada etapa do plano
- **Sistema de Curtidas:** Curtir comentários dos membros
- **Visualização de Progresso:** Ver quem completou cada etapa

Modelos do Banco de Dados

```

model ReadingPlan {
  id          String   @id @default(cuid())
  title       String
  description  String?
  duration    Int       // duração em dias
  isPublic    Boolean   @default(false)
  creatorId   String
  startDate   DateTime
  endDate     DateTime
}

model ReadingPlanStep {
  id          String   @id @default(cuid())
  planId      String
  dayNumber   Int       // Dia 1, 2, 3...
  book        String
  chapter     Int
  verses      String?   // "1-10" ou null
  reflection  String?   // Reflexão do dia
}

model ReadingPlanMember {
  id          String   @id @default(cuid())
  planId      String
  userId      String
  role        String   @default("member") // 'admin', 'member'
  joinedAt    DateTime @default(now())
  leftAt      DateTime?
}

model ReadingPlanProgress {
  id          String   @id @default(cuid())
  stepId      String
  userId      String
  completed    Boolean   @default(false)
  completedAt  DateTime?
}









model PlanComment {
  id          String   @id @default(cuid())
  stepId      String
  userId      String
  content     String
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
}

model PlanLike {
  id          String   @id @default(cuid())
  commentId   String
  userId      String
  createdAt   DateTime @default(now())
}

```

Arquivos Criados

- ✓ /app/planos-com-amigos/page.tsx - Lista de planos
- ✓ /app/planos-com-amigos/criar/page.tsx - Criar novo plano

-  `/app/planos-com-amigos/[id]/page.tsx` - Detalhes do plano com progresso e comentários
-  `/app/api/plans/route.ts` - GET (listar) e POST (criar)
-  `/app/api/plans/[id]/route.ts` - GET (detalhes) e DELETE (remover)
-  `/app/api/plans/[id]/join/route.ts` - Participar de um plano
-  `/app/api/plans/[id]/leave/route.ts` - Sair de um plano
-  `/app/api/plans/steps/[stepId]/progress/route.ts` - Marcar etapa como concluída
-  `/app/api/plans/steps/[stepId]/comments/route.ts` - GET e POST comentários
-  `/app/api/comments/[commentId]/like/route.ts` - Curtir/descurtir comentário

API Endpoints

```
// Planos
GET /api/plans?type=my           // Meus planos
GET /api/plans?type=created      // Planos criados
GET /api/plans?type=public      // Planos públicos

POST /api/plans
Body: { title, description, duration, isPublic, steps[] }

GET /api/plans/:id              // Detalhes do plano
DELETE /api/plans/:id           // Remover plano (só criador)

POST /api/plans/:id/join        // Entrar no plano
POST /api/plans/:id/leave       // Sair do plano

// Progresso
POST /api/plans/steps/:stepId/progress
Body: { completed: boolean }

// Comentários
GET /api/plans/steps/:stepId/comments // Listar comentários
POST /api/plans/steps/:stepId/comments // Criar comentário
Body: { content: string }

// Curtidas
POST /api/comments/:commentId/like // Toggle curtida
```

4. Privacidade e RBAC

Implementação de Segurança

Dados Privados (100% Privados)

- **Progresso de Leitura Individual:** Não compartilhado
- **Journals (Gratidão, Insight, Humor):** Estritamente privados
- **Highlights e Notas:** Visíveis apenas para o dono
- **Favoritos:** Privados por usuário

Dados Semi-Públicos (Compartilhados em Contexto)

- **Comentários em Planos:** Visíveis apenas para membros do plano
- **Progresso em Etapas:** Visível apenas para membros do plano
- **Curtidas:** Visíveis apenas no contexto do plano

✓ Dados Públicos para Amigos

- **Streak:** Visível para amigos
- **Total de Devocionais:** Visível para amigos
- **Nome e Avatar:** Visível em buscas

Verificações de Permissão

Todas as rotas implementam verificação RBAC:

```
// Exemplo de verificação em comentários
const step = await prisma.readingPlanStep.findUnique({
  where: { id: stepId },
  include: {
    plan: {
      include: {
        members: {
          where: { userId: user.id, leftAt: null },
        },
      },
    },
  },
});

// RBAC: Apenas membros podem comentar
if (step.plan.members.length === 0) {
  return NextResponse.json({ error: 'Not a member of this plan' }, { status: 403 });
}
```

5. Interface do Usuário

Componentes Criados

1. **Streak Badge** (/components/streak-badge.tsx)
 - Exibe dias consecutivos
 - Ícone de fogo animado
 - Mostra recorde pessoal
2. **Página de Amigos** (/app/amigos/page.tsx)
 - Busca de usuários
 - Gerenciamento de pedidos
 - Lista de amigos com streaks
3. **Lista de Planos** (/app/planos-com-amigos/page.tsx)
 - Meus planos com progresso
 - Planos públicos disponíveis
 - Cards com informações de membros
4. **Criar Plano** (/app/planos-com-amigos/criar/page.tsx)
 - Formulário multi-etapa
 - Adicionar/remover dias
 - Definir leituras e reflexões
5. **Detalhes do Plano** (/app/planos-com-amigos/[id]/page.tsx)
 - Progresso individual e do grupo

- Sistema de check para marcar etapas
- Comentários expandíveis por etapa
- Curtidas em tempo real
- Lista de membros com streaks

Atualizações na Home

- ☒ CTAs para “Planos com Amigos”
- ☒ CTAs para “Amigos”
- ☒ Design responsivo com gradientes atrativos

Atualização na Navbar

- ☒ Link para “Planos”
- ☒ Link para “Amigos”
- ☒ Streak Badge sempre visível
- ☒ Ícones lucide-react



Como Usar

1. Sistema de Streak

1. Complete o fluxo “Respira & Ore” diariamente
2. Preencha o journal ao final
3. Seu streak será atualizado automaticamente
4. Visualize seu progresso no badge na navbar

2. Adicionar Amigos

1. Vá para `/amigos`
2. Busque por nome ou email
3. Envie pedidos de amizade
4. Aceite pedidos recebidos
5. Veja os streaks dos seus amigos

3. Criar Plano com Amigos

1. Vá para `/planos-com-amigos`
2. Clique em “Criar Plano”
3. Defina título, descrição e tipo (público/privado)
4. Adicione as leituras de cada dia
5. (Opcional) Adicione reflexões para cada dia
6. Convide amigos ou torne público

4. Participar de Planos

1. Visualize planos públicos ou dos seus amigos
2. Clique em “Participar”
3. Marque seu progresso diário
4. Comente e curta as reflexões dos membros
5. Acompanhe o progresso do grupo



Estatísticas do Módulo IV

- **7 Novos Modelos** no Prisma Schema
- **15 Novos Endpoints de API**
- **5 Novas Páginas de UI**
- **3 Componentes Reutilizáveis**
- **100% RBAC Implementado**
- **Zero Vazamento de Dados Privados**



Checklist de Implementação

- [x] Sistema de Streak com cálculo backend
- [x] Integração do streak com fluxo “Respira & Ore”
- [x] Modelo de amizades no banco de dados
- [x] APIs de gerenciamento de amizades
- [x] Busca de usuários
- [x] Modelo de planos compartilhados
- [x] APIs de gerenciamento de planos
- [x] Sistema de progresso individual
- [x] Sistema de comentários
- [x] Sistema de curtidas
- [x] RBAC completo em todas as rotas
- [x] Privacidade de dados pessoais garantida
- [x] UI completa e responsiva
- [x] Integração com navbar e home
- [x] Streak badge visível
- [x] Testes e build bem-sucedidos



Próximos Passos Sugeridos

1. **Notificações:** Adicionar notificações para pedidos de amizade e comentários
2. **Push Notifications:** Lembrar usuários de completar o devocional diário
3. **Analytics:** Dashboard com insights sobre engajamento do grupo
4. **Conquistas:** Badges para milestones (7 dias, 30 dias, 100 dias)
5. **Compartilhamento:** Permitir compartilhar versículos e reflexões



Notas Técnicas

Performance

- Todas as queries utilizam índices apropriados

- Relações otimizadas com `include` seletivo
- Paginação implementada onde necessário

Segurança

- Autenticação em todas as rotas
- Validação de permissões em cada operação
- SQL injection prevenido pelo Prisma
- XSS protection no Next.js

Escalabilidade

- Schema preparado para adicionar features
- APIs RESTful bem estruturadas
- Componentes React modulares e reutilizáveis

Módulo IV implementado com sucesso! 🎉

Desenvolvido com Next.js 14, Prisma, PostgreSQL e TypeScript