🚀 Quick Start - Timming LoveU

Guia rápido para começar a trabalhar no projeto.

📦 Setup Inicial

```
# 1. Navegar para o projeto
cd /home/ubuntu/timming_loveu

# 2. Instalar dependências
npm install

# 3. Configurar variáveis de ambiente
cp .env.example .env
# Editar .env com suas credenciais

# 4. Setup do banco de dados
npx prisma generate
npx prisma migrate dev
npx prisma db seed

# 5. Iniciar desenvolvimento
npm run dev
```

Aplicação rodando em: **http://localhost:3000**

🗄️ Variáveis de Ambiente Obrigatórias

```
# Database (já configurado)
DATABASE_URL="postgresql://..."

# NextAuth
NEXTAUTH_SECRET="generate-with-openssl-rand-base64-32"
NEXTAUTH_URL="http://localhost:3000"
```

Gerar `NEXTAUTH_SECRET`:

```
openssl rand -base64 32
```

🎨 Estrutura do Projeto

```
timming_loveu/
├── app/                    # Next.js App Router
│   ├── api/                # API Routes
│   ├── dashboard/          # Dashboard do usuário
│   ├── login/              # Login
│   ├── signup/             # Cadastro
│   └── page.tsx            # Homepage
│
├── components/             # Componentes React
│   ├── dashboard/          # Componentes do dashboard
│   └── ui/                 # Componentes UI (Radix)
│
├── lib/                    # Bibliotecas e helpers
│   ├── auth/               # Autenticação
│   ├── db.ts               # Prisma client
│   └── utils.ts            # Utilitários
│
├── prisma/                 # Prisma ORM
│   └── schema.prisma       # Schema do banco
│
└── public/                 # Arquivos estáticos
    └── uploads/            # Upload de mídia
```

🔑 Comandos Principais

Desenvolvimento

```
npm run dev              # Iniciar dev server
npm run build            # Build produção
npm run start            # Iniciar produção
npm run lint             # Linter
npm run type-check       # Type checking
```

Banco de Dados

```
npm run prisma:generate  # Gerar Prisma Client
npm run prisma:migrate   # Rodar migrations
npm run prisma:studio    # Abrir Prisma Studio (UI)
npm run prisma:seed      # Seed do banco
```

Testes

```
npm test                 # Rodar testes
npm run test:watch       # Watch mode
npm run test:coverage    # Coverage
```

Docker

```
npm run docker:build     # Build imagem
npm run docker:up        # Iniciar containers
npm run docker:down      # Parar containers
npm run docker:logs      # Ver logs
```

📝 Fluxo de Desenvolvimento

1. Criar Nova Feature

```
# 1. Criar branch
git checkout -b feature/nome-da-feature

# 2. Desenvolver
# ... código ...

# 3. Testar
npm test

# 4. Commit
git add .
git commit -m "feat: descrição da feature"

# 5. Push
git push origin feature/nome-da-feature
```

2. Modificar Schema do Banco

```
# 1. Editar prisma/schema.prisma
# ... modificações ...

# 2. Criar migration
npx prisma migrate dev --name nome_da_migration

# 3. Gerar Prisma Client
npx prisma generate
```

3. Criar API Route

```
// app/api/minha-rota/route.ts
import { NextRequest, NextResponse } from 'next/server'
import { getServerSession } from 'next-auth'
import { authOptions } from '@/lib/auth/auth-options'
import { prisma } from '@/lib/db'

export async function GET(req: NextRequest) {
  const session = await getServerSession(authOptions)
  if (!session) {
    return NextResponse.json({ error: 'Unauthorized' }, { status: 401 })
  }

  // Sua lógica aqui
  const data = await prisma.user.findUnique({
    where: { id: session.user.id }
  })

  return NextResponse.json({ data })
}
```

4. Criar Componente

```
// components/meu-componente.tsx
'use client'

import { useState } from 'react'
import { Button } from '@/components/ui/button'

export function MeuComponente() {
  const [state, setState] = useState('')

  return (
    <div>
      <Button onClick={() => setState('clicked')}>
        Click me
      </Button>
    </div>
  )
}
```

🐛 Troubleshooting

Erro: Port 3000 já em uso

```
# Matar processo na porta 3000
lsof -ti:3000 | xargs kill -9

# Ou usar outra porta
PORT=3001 npm run dev
```

Erro: Database connection failed

```
# Verificar se PostgreSQL está rodando
psql $DATABASE_URL

# Testar conexão
npx prisma db pull
```

Erro: Prisma Client não encontrado

```
# Gerar Prisma Client
npx prisma generate
```

Build falha

```
# Limpar cache
rm -rf .next node_modules
npm install
npm run build
```

📚 Recursos

Documentação

- **Next.js:** https://nextjs.org/docs
- **Prisma:** https://www.prisma.io/docs
- **NextAuth:** https://next-auth.js.org
- **Tailwind:** https://tailwindcss.com/docs
- **Radix UI:** https://www.radix-ui.com

Guias Internos

- `ANALISE_ARQUITETURA.md` - Arquitetura completa
- `PAYMENT_IMPLEMENTATION_GUIDE.md` - Implementar pagamentos
- `DEPLOYMENT_GUIDE.md` - Deploy em produção
- `SECURITY_CHECKLIST.md` - Segurança

🎯 Próximos Passos

1. ✅ Setup inicial completo
2. ⏳ Implementar sistema de pagamento (Stripe)
3. ⏳ Migrar uploads para cloud storage (S3/Cloudinary)
4. ⏳ Aplicar rate limiting
5. ⏳ Configurar monitoring

💬 Suporte

- Health check: `GET http://localhost:3000/api/health`
- Logs: `tail -f logs/app.log`
- Database UI: `npm run prisma:studio`

Criado por: DeepAgent - Abacus.AI
Data: 23/10/2024