🚀 Setup Completo - Turion API Nav

Data: 18 de Fevereiro de 2026
Status: ✅ Projeto funcional

📋 O Que Foi Corrigido

1. Configuração de Ambiente

- ✅ Criado arquivo `.env.local` com todas as credenciais necessárias:
- Supabase URL e keys
- Stripe keys (test mode)
- URLs da aplicação
- DATABASE_URL para Prisma (SQLite)
- ✅ Criado arquivo `.env` (cópia do .env.local) para Prisma CLI

2. Dependências

- ✅ Instaladas todas as dependências Node.js (`npm install`)
- ✅ Instalados browsers do Playwright (Chromium)
- ✅ Instaladas dependências Python do mini-service browser-api

3. Database (Prisma)

- ✅ Gerado Prisma Client (`npx prisma generate`)
- ✅ Criado database SQLite (`npx prisma db push`)
- ⚠️ **Nota:** O projeto usa principalmente Supabase via JS client, não Prisma

4. Erros de Código Corrigidos

4.1 Icon Inexistente (lucide-react)

- **Arquivo:** `src/app/dashboard/page.tsx`
- **Erro:** `UsageIcon` não existe em lucide-react
- **Solução:** Substituído por `Activity`

4.2 Supabase Auth Helpers Deprecados

- **Arquivos afetados:**
- `src/app/api/keys/route.ts`
- `src/app/api/stripe/checkout/route.ts`
- `src/app/api/stripe/portal/route.ts`
- `src/app/api/usage/route.ts`
- **Erro:** `createRouteHandlerClient` de `@supabase/auth-helpers-nextjs` está deprecado
- **Solução:**
- Criado `src/lib/supabase-server.ts` com nova implementação usando `@supabase/ssr`
- Atualizado todos os imports para usar o novo módulo

🚀 Como Iniciar o Projeto

Pré-requisitos

- Node.js 18+
- Python 3.9+ (para mini-service)

Iniciando o Frontend (Next.js)

```
cd /home/ubuntu/github_repos/turion_api_nav

# Instalar dependências (se necessário)
npm install

# Modo desenvolvimento
npm run dev

# Ou modo produção
npm run build
npm start
```

Acesse: http://localhost:3000

Iniciando o Mini-Service (Browser API)

```
cd /home/ubuntu/github_repos/turion_api_nav/mini-services/browser-api

# Ativar ambiente virtual (se não existir, crie com: python3 -m venv venv)
source venv/bin/activate  # Linux/Mac
# ou: venv\Scripts\activate  # Windows

# Instalar dependências
pip install -r requirements.txt

# Instalar navegador Playwright
playwright install chromium

# Iniciar servidor
uvicorn app.main:app --reload --port 8001
```

API Docs: http://localhost:8001/docs

⚠️ Problemas Restantes (Não Críticos)

1. STRIPE_WEBHOOK_SECRET Placeholder

- **Status:** O webhook secret ainda é `whsec_your-webhook-secret-here`
- **Impacto:** Webhooks do Stripe não funcionarão até configurar
- **Solução:** Configurar no Stripe Dashboard e atualizar .env.local

2. Supabase Schema

- **Status:** As tabelas do Supabase (profiles, api_keys, subscriptions, etc.) precisam ser criadas
- **Solução:** Execute o conteúdo de `supabase-schema.sql` no Supabase SQL Editor

3. Stripe Products/Prices

- **Status:** Os produtos e preços não estão criados no Stripe
- **Solução:** Criar produtos no Stripe Dashboard ou configurar Price IDs

4. Dependências Python Deprecadas

- **Status:** Versões fixadas podem ter conflitos com outros pacotes
- **Impacto:** Apenas warnings, não afeta funcionamento

📁 Estrutura de Arquivos Modificados

```
turion-api-nav/
├── .env.local           # CRIADO - Variáveis de ambiente
├── .env                 # CRIADO - Para Prisma CLI
├── db/custom.db         # CRIADO - Database SQLite
├── node_modules/        # INSTALADO
├── src/
│   ├── lib/
│   │   ├── supabase.ts          # MODIFICADO - Removido next/headers
│   │   └── supabase-server.ts   # CRIADO - Cliente para API routes
│   └── app/
│       ├── dashboard/page.tsx   # MODIFICADO - UsageIcon → Activity
│       └── api/
│           ├── keys/route.ts            # MODIFICADO - Novo import
│           ├── usage/route.ts           # MODIFICADO - Novo import
│           └── stripe/
│               ├── checkout/route.ts    # MODIFICADO - Novo import
│               └── portal/route.ts      # MODIFICADO - Novo import
└── mini-services/browser-api/
    └── (dependências Python instaladas)
```

🎯 Próximos Passos Recomendados

Prioridade Alta

1. **Configurar Supabase Database**
   - Executar `supabase-schema.sql` no SQL Editor do Supabase
   - Verificar se as tabelas foram criadas corretamente

2. **Configurar Stripe Webhook**
   - Criar endpoint no Stripe Dashboard
   - Obter o webhook secret real
   - Atualizar `STRIPE_WEBHOOK_SECRET` no .env.local

Prioridade Média

1. **Testar fluxo completo**
   - Registrar usuário
   - Gerar API key
   - Testar scraping endpoint
   - Verificar dashboard

2. **Deploy**
   - Configurar servidor de produção
   - Setup Caddy como reverse proxy
   - Migrar Stripe para live mode

Prioridade Baixa

1. **Otimizações**
   - Adicionar testes
   - Configurar CI/CD
   - Monitoramento (Sentry, etc.)

📊 Resumo do Status

Item	Status
Build	✅ Funcional
Dev Server	✅ Funcional
Mini-Service	✅ Instalado
Supabase	⚠️ Schema pendente
Stripe	⚠️ Config pendente
Testes	❌ Não implementado

Conclusão: O projeto está **funcional para desenvolvimento**. Para produção, é necessário configurar Supabase schema e Stripe.

Documentação gerada automaticamente durante o processo de setup.