# 📊 Análise Completa do Projeto Turion API Nav

**Data da Análise:** 18 de Fevereiro de 2026
**Repositório:** https://github.com/BollaNetwork/turion-api-nav
**Status:** Projeto clonado e analisado com sucesso

---

## 🎯 Propósito do Projeto

**Turion** é uma API de web scraping/rendering voltada para desenvolvedores que precisam de automação de navegador em escala. O projeto oferece:

- **Renderização de JavaScript** com Headless Chrome
- **Captura de screenshots** (full-page ou viewport)
- **Geração de PDFs** de páginas web
- **API REST simples** com autenticação via API keys
- **Sistema de assinaturas** com Stripe (planos Free, Starter, Growth, Scale)
- **Dashboard de usuário** para gerenciamento de API keys e monitoramento de uso

### Modelo de Negócio

- Micro-SaaS com modelo pay-per-request
- Preços em £ GBP (mercado UK)
- Compliance GDPR (servidores na Europa)
- Planos de £0 a £79/mês

---

## 🛠️ Stack Tecnológico

### Frontend (Next.js App)

| Tecnologia | Versão | Propósito |
|---|---|---|
| **Next.js** | 16.1.1 | Framework React com App Router |
| **React** | 19.0.0 | Biblioteca UI |
| **TypeScript** | 5.x | Type safety |
| **Tailwind CSS** | 4.x | Styling utility-first |
| **shadcn/ui** | - | Componentes UI (Radix UI) |
| **Framer Motion** | 12.23.2 | Animações |

## Backend & Database

| Tecnologia | Propósito |
|---|---|
| **Supabase** | Autenticação + PostgreSQL database |
| **Prisma** | ORM (configurado para SQLite local) |
| **Stripe** | Pagamentos e assinaturas |
| **NextAuth.js** | Autenticação (alternativa) |

## Mini-Service (Browser API)

| Tecnologia | Versão | Propósito |
|---|---|---|
| **FastAPI** | 0.109.2 | Framework Python para API |
| **Playwright** | 1.41.0 | Automação de navegador |
| **Uvicorn** | 0.27.1 | ASGI server |
| **Gunicorn** | 21.2.0 | Process manager |

## Ferramentas de Build

- **Bun** - Runtime JavaScript (alternativa ao Node.js)
- **ESLint** - Linting
- **PostCSS** - CSS processing

---

## 📁 Estrutura do Projeto

```
turion-api-nav/
├── .zscripts/                    # Scripts de build e deploy
│   ├── build.sh
│   ├── start.sh
│   ├── mini-services-build.sh
│   ├── mini-services-install.sh
│   └── mini-services-start.sh
│
├── mini-services/                # Microserviços Python
│   └── browser-api/              # API de automação de navegador
│       ├── app/
│       │   ├── main.py           # Entry point FastAPI
│       │   ├── routes.py         # Rotas da API
│       │   ├── models/schemas.py # Modelos Pydantic
│       │   └── services/browser.py # Lógica Playwright
│       ├── Dockerfile
│       ├── docker-compose.yml
│       ├── requirements.txt
│       └── nginx.conf
│
├── src/                          # Código fonte Next.js
│   ├── app/                      # App Router (Next.js 16)
│   │   ├── page.tsx              # Landing page
│   │   ├── layout.tsx            # Layout principal
│   │   ├── globals.css           # Estilos globais
│   │   ├── auth/                 # Páginas de autenticação
│   │   │   ├── login/page.tsx
│   │   │   ├── signup/page.tsx
│   │   │   └── callback/page.tsx
│   │   ├── dashboard/page.tsx    # Dashboard do usuário
│   │   ├── demo/page.tsx         # Página de demo
│   │   └── api/                  # API Routes
│   │       ├── route.ts          # API principal
│   │       ├── keys/route.ts     # Gerenciamento de API keys
│   │       ├── usage/route.ts    # Estatísticas de uso
│   │       └── stripe/           # Integração Stripe
│   │           ├── checkout/route.ts
│   │           ├── portal/route.ts
│   │           └── webhooks/stripe/route.ts
│   │
│   ├── components/               # Componentes React
│   │   ├── ui/                   # shadcn/ui components (100+ componentes)
│   │   ├── InteractiveDemo.tsx
│   │   └── demo/DemoModal.tsx
│   │
│   ├── contexts/                 # React Contexts
│   │   └── AuthContext.tsx       # Contexto de autenticação
│   │
│   ├── hooks/                    # Custom hooks
│   │   ├── use-mobile.ts
│   │   └── use-toast.ts
│   │
│   └── lib/                      # Utilitários
│       ├── supabase.ts           # Cliente Supabase
│       ├── db.ts                 # Database helpers
│       ├── types.ts              # TypeScript types
│       └── utils.ts              # Funções utilitárias
│
├── prisma/
│   └── schema.prisma             # Schema Prisma (SQLite)
│
├── supabase/
```

```
│   └── schema.sql                # Schema PostgreSQL
│
├── public/                       # Assets estáticos
│   ├── logo.svg
│   └── robots.txt
│
├── db/
│   └── custom.db                 # SQLite database local
│
├── examples/                     # Exemplos de código
│   └── websocket/
│       ├── frontend.tsx
│       └── server.ts
│
├── download/                     # Diretório para downloads
│
├── package.json                  # Dependências Node.js
├── bun.lock                      # Lockfile do Bun
├── tsconfig.json                 # Config TypeScript
├── next.config.ts                # Config Next.js
├── tailwind.config.ts            # Config Tailwind
├── eslint.config.mjs             # Config ESLint
├── postcss.config.mjs            # Config PostCSS
├── components.json               # Config shadcn/ui
├── Caddyfile                     # Config Caddy (reverse proxy)
│
├── README.md                     # Documentação principal
├── MVP_INTEGRATION.md            # Guia de integração Supabase/Stripe
├── SUPABASE_SETUP.md             # Setup do Supabase
├── BUSINESS_PLAN.md              # Plano de negócios
├── worklog.md                    # Log de trabalho
└── supabase-schema.sql           # Schema SQL completo
```

---

# 📦 Dependências Principais

## Frontend (package.json)

```
{
  "dependencies": {
    "next": "^16.1.1",
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "typescript": "^5",
    "@supabase/supabase-js": "^2.96.0",
    "@supabase/ssr": "^0.8.0",
    "stripe": "^20.3.1",
    "@stripe/stripe-js": "^8.7.0",
    "prisma": "^6.11.1",
    "@prisma/client": "^6.11.1",
    "tailwindcss": "^4",
    "framer-motion": "^12.23.2",
    "zod": "^4.0.2",
    "zustand": "^5.0.6",
    "@tanstack/react-query": "^5.82.0",
    "react-hook-form": "^7.60.0",
    "lucide-react": "^0.525.0"
  }
}
```

## Backend (mini-services/browser-api/requirements.txt)

```
fastapi==0.109.2
uvicorn[standard]==0.27.1
gunicorn==21.2.0
playwright==1.41.0
httpx==0.26.0
pydantic==2.6.1
pydantic-settings==2.2.1
```

---

# ⚠️ Problemas Identificados

## 🔴 CRÍTICOS (Impedem o funcionamento)

### 1. Variáveis de Ambiente Ausentes

**Problema:** Não existe arquivo `.env` ou `.env.local` no repositório.

**Variáveis necessárias:**

```
# Supabase (OBRIGATÓRIO)
NEXT_PUBLIC_SUPABASE_URL=https://xxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...
SUPABASE_SERVICE_ROLE_KEY=eyJ...

# Stripe (OBRIGATÓRIO para pagamentos)
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_...
STRIPE_SECRET_KEY=sk_live_...
STRIPE_WEBHOOK_SECRET=whsec_...
STRIPE_STARTER_PRICE_ID=price_...
STRIPE_GROWTH_PRICE_ID=price_...
STRIPE_SCALE_PRICE_ID=price_...

# App
NEXT_PUBLIC_APP_URL=http://localhost:3000
DATABASE_URL=file:./db/custom.db

# NextAuth (se usado)
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=<random-secret>
```

**Impacto:** Aplicação não inicia sem essas variáveis.

---

### 2. Dependências Não Instaladas

**Problema:** Não existe diretório `node_modules/`.

**Solução necessária:**

```
bun install
# ou
npm install
```

**Impacto:** Não é possível executar `bun dev` sem instalar dependências.

---

### 3. Playwright Browsers Não Instalados

**Problema:** O mini-service `browser-api` usa Playwright, que requer download de navegadores.

**Solução necessária:**

```
cd mini-services/browser-api
pip install -r requirements.txt
playwright install chromium
```

**Impacto:** API de scraping não funciona sem os navegadores.

---

### 4. Database Não Configurado

**Problema:** Existem dois sistemas de database conflitantes:
- **Prisma** configurado para SQLite ( `db/custom.db` )
- **Supabase** (PostgreSQL) usado no código

**Conflito:**
- `prisma/schema.prisma` aponta para SQLite
- Código em `src/` usa Supabase (PostgreSQL)
- Não há migrations aplicadas

**Solução necessária:**
1. Decidir qual database usar (recomendado: Supabase)
2. Se Supabase:
- Executar `supabase-schema.sql` no projeto Supabase
- Configurar variáveis de ambiente
3. Se Prisma local:
- Executar `bun db:push` ou `bun db:migrate`

**Impacto:** Autenticação e armazenamento de dados não funcionam.

---

## 🟡 IMPORTANTES (Afetam funcionalidades)

### 5. Stripe Não Configurado

**Problema:**
- Webhooks não configurados
- Price IDs não criados
- Produtos não criados no Stripe Dashboard

**Solução necessária:**
1. Criar conta Stripe (UK)
2. Criar produtos e preços
3. Configurar webhook endpoint
4. Adicionar Price IDs ao `.env`

**Impacto:** Sistema de pagamentos não funciona.

---

### 6. Supabase OAuth Não Configurado

**Problema:** Código usa Google OAuth via Supabase, mas não está configurado.

**Solução necessária:**
1. Configurar Google OAuth no Supabase Dashboard
2. Adicionar Client ID e Secret do Google Cloud Console
3. Configurar URLs de callback

**Impacto:** Login social não funciona (apenas email/senha).

---

### 7. Mini-Service Não Buildado

**Problema:** O serviço Python `browser-api` não está containerizado ou rodando.

**Solução necessária:**

```
cd mini-services/browser-api
docker-compose up -d
# ou
python -m uvicorn app.main:app --host 0.0.0.0 --port 8001
```

**Impacto:** Funcionalidade de scraping não está disponível.

---

## 🟢 MENORES (Melhorias recomendadas)

### 8. Falta de Documentação de API

**Problema:** Não há documentação OpenAPI/Swagger gerada.

**Recomendação:** FastAPI gera automaticamente em `/docs`.

---

### 9. Logs e Monitoramento

**Problema:** Não há sistema de logging estruturado.

**Recomendação:** Implementar logging com Winston ou Pino.

---

### 10. Testes Ausentes

**Problema:** Não há testes unitários ou de integração.

**Recomendação:** Adicionar Jest/Vitest para frontend, pytest para backend.

---

# 🔧 O Que Precisa Ser Corrigido (Checklist)

## Fase 1: Configuração Básica (Essencial)

- [ ] **Criar arquivo** `.env.local` com todas as variáveis necessárias
- [ ] **Instalar dependências Node.js**: `bun install`
- [ ] **Configurar Supabase**:
- [ ] Criar projeto no Supabase
- [ ] Executar `supabase-schema.sql` no SQL Editor
- [ ] Copiar URL e keys para `.env.local`
- [ ] **Configurar Database**:
- [ ] Decidir entre Prisma (SQLite) ou Supabase (PostgreSQL)
- [ ] Se Prisma: executar `bun db:push`
- [ ] Se Supabase: já configurado no passo anterior

## Fase 2: Mini-Service (Browser API)

- [ ] **Instalar dependências Python**:
  `bash`
  `cd mini-services/browser-api`
  `pip install -r requirements.txt`
- [ ] **Instalar navegadores Playwright**:
  `bash`
  `playwright install chromium`
- [ ] **Testar mini-service**:
  `bash`
  `uvicorn app.main:app --reload`
- [ ] **Verificar endpoint**: `http://localhost:8000/docs`

## Fase 3: Integração Stripe (Opcional para MVP)

- [ ] **Criar conta Stripe** (modo test)
- [ ] **Criar produtos e preços** no Dashboard
- [ ] **Configurar webhook** apontando para `/api/webhooks/stripe`
- [ ] **Adicionar Price IDs** ao `.env.local`
- [ ] **Testar checkout flow**

## Fase 4: Testes e Deploy

- [ ] **Testar aplicação localmente**:
  `bash`
  `bun dev`
- [ ] **Verificar páginas**:
- [ ] Landing page: `http://localhost:3000`
- [ ] Login: `http://localhost:3000/auth/login`
- [ ] Dashboard: `http://localhost:3000/dashboard`
- [ ] **Testar fluxo completo**:
- [ ] Registro de usuário
- [ ] Geração de API key
- [ ] Chamada à API de scraping
- [ ] Visualização de uso no dashboard

## Fase 5: Produção (Futuro)

- [ ] Configurar domínio (turion.network)
- [ ] Setup Caddy como reverse proxy
- [ ] Configurar SSL/TLS
- [ ] Migrar Stripe para modo live
- [ ] Configurar monitoramento (Sentry, LogRocket)
- [ ] Setup CI/CD

---

# 🚀 Como Deixar o Projeto Funcional

## Passo 1: Criar `.env.local`

```
cd /home/ubuntu/github_repos/turion_api_nav
cat > .env.local << 'EOF'
# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://seu-projeto.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=sua-anon-key-aqui
SUPABASE_SERVICE_ROLE_KEY=sua-service-role-key-aqui

# Stripe (modo test)
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_xxx
STRIPE_SECRET_KEY=sk_test_xxx
STRIPE_WEBHOOK_SECRET=whsec_xxx
STRIPE_STARTER_PRICE_ID=price_xxx
STRIPE_GROWTH_PRICE_ID=price_xxx
STRIPE_SCALE_PRICE_ID=price_xxx

# App
NEXT_PUBLIC_APP_URL=http://localhost:3000
DATABASE_URL=file:./db/custom.db

# NextAuth
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=$(openssl rand -base64 32)
EOF
```

## Passo 2: Instalar Dependências

```
# Frontend
bun install

# Backend (mini-service)
cd mini-services/browser-api
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
playwright install chromium
cd ../..
```

## Passo 3: Configurar Supabase

1. Acesse https://supabase.com
2. Crie novo projeto (região EU)
3. Vá para SQL Editor
4. Execute o conteúdo de `supabase-schema.sql`
5. Copie URL e keys para `.env.local`

### Passo 4: Iniciar Aplicação

```
# Terminal 1: Next.js
bun dev

# Terminal 2: Browser API (mini-service)
cd mini-services/browser-api
source venv/bin/activate
uvicorn app.main:app --reload --port 8001
```

### Passo 5: Testar

- Acesse: http://localhost:3000
- Registre um usuário
- Gere uma API key
- Teste a API de scraping

---

## 📊 Estimativa de Capacidade

Baseado no `BUSINESS_PLAN.md`:

| Recurso | Especificação |
|---|---|
| Servidor | 4 vCores, 8GB RAM, 150GB SSD |
| Custo | £10/mês |
| Navegadores Simultâneos | 3-4 instâncias |
| Memória por Navegador | ~1.5GB |
| Tempo Médio de Request | 3-5 segundos |
| Limite Seguro | 30 requests/min |
| **Capacidade Mensal** | **~1.3M requests/mês** |

---

## 🎯 Conclusão

### Status Atual

**✅ Código-fonte completo e bem estruturado**
**✅ Arquitetura moderna (Next.js 16 + FastAPI)**
**✅ UI profissional com shadcn/ui**
**✅ Documentação detalhada (README, MVP_INTEGRATION, BUSINESS_PLAN)**

**❌ Não funcional sem configuração**
**❌ Dependências não instaladas**

❌ **Variáveis de ambiente ausentes**
❌ **Database não configurado**
❌ **Stripe não configurado**

## Próximos Passos Recomendados

1. **Configurar ambiente** ( `.env.local` )
2. **Instalar dependências** (bun + pip)
3. **Setup Supabase** (database + auth)
4. **Testar localmente** (Next.js + FastAPI)
5. **Configurar Stripe** (opcional para MVP)
6. **Deploy em produção** (VPS + Caddy)

## Tempo Estimado para Deixar Funcional

- **Configuração básica**: 30-60 minutos
- **Setup Supabase**: 15-30 minutos
- **Testes locais**: 15-30 minutos
- **Setup Stripe**: 30-60 minutos (opcional)
- **Total**: **1.5 - 3 horas**

---

**Análise realizada por:** Z.ai Agent
**Data:** 18/02/2026
**Versão do Projeto:** 0.2.0