SNE Radar - Complete System Architecture

High-Level Overview

```
graph TB
    subgraph "User Touchpoints"
        USER[("👤 Trader")]
        TELEGRAM[("📱 Telegram")]
    end

    subgraph "Frontend Layer"
        LANDING["🌐 Landing Page<br/>radar.snelabs.space<br/>(React + Vite)"]
        DESKTOP["🖥️ Desktop App<br/>SNE_Radar.exe<br/>(pywebview + Vue.js)"]
    end

    subgraph "Backend Layer"
        API["⚙️ Backend API<br/>api.snelabs.space<br/>(Flask + SocketIO)"]
        BOT["🤖 Telegram Bot<br/>(python-telegram-bot)"]
    end

    subgraph "Data Layer"
        DB[("🗄️ PostgreSQL<br/>Cloud DB")]
        SQLITE[("📁 SQLite<br/>Local (Desktop)")]
    end

    subgraph "External Services"
        BINANCE["📈 Binance API"]
        BYBIT["📊 Bybit API"]
        SCROLL["⛓️ Scroll Chain<br/>(NFT Licenses)"]
        WALLET["🦊 MetaMask"]
    end

    USER --> LANDING
    USER --> DESKTOP
    USER --> TELEGRAM

    LANDING --> WALLET
    LANDING --> API
    LANDING --> SCROLL

    DESKTOP --> API
    DESKTOP --> SQLITE
    DESKTOP --> BINANCE
    DESKTOP --> BYBIT

    BOT --> API
    BOT --> TELEGRAM

    API --> DB
    API --> BINANCE
    API --> BYBIT
```

Component Architecture

Desktop App (SNE_Radar.exe)

```
graph LR
    subgraph "Desktop Package"
        EXE["SNE_Radar.exe"]

        subgraph "Python Backend"
            FLASK["Flask Server<br/>:9999"]
            SOCKETIO["SocketIO"]
            AUTH["auth_manager.py"]
            MONITORS["monitors/"]
            SERVICES["services/"]
        end

        subgraph "Vue.js Frontend"
            DASHBOARD["Dashboard.vue"]
            RADAR["WickRadar.vue"]
            ANALYSIS["Analysis.vue"]
            TRADING["AutomatedTrading.vue"]
            LOCKSCREEN["LockScreen.vue"]
        end
    end

    EXE --> FLASK
    FLASK --> SOCKETIO
    FLASK --> AUTH
    FLASK --> MONITORS
    FLASK --> SERVICES

    FLASK -->|"HTTP/WS"| DASHBOARD
    DASHBOARD --> RADAR
    DASHBOARD --> ANALYSIS
    DASHBOARD --> TRADING
```

Landing Page (radar.snelabs.space)

```
graph TB
    subgraph "Landing Page (Vercel)"
        APP["App.tsx"]

        subgraph "Auth Flow"
            SIWE["SIWE Auth"]
            WAGMI["wagmi (Web3)"]
            WALLET_SEL["WalletSelector"]
        end

        subgraph "License Flow"
            MINT["Mint License"]
            DOWNLOAD["Download Handler"]
        end

        subgraph "UI Components"
            HERO["Hero Section"]
            PRICING["Pricing Cards"]
            FAQ["FAQ Accordion"]
        end
    end

    APP --> SIWE
    SIWE --> WAGMI
    WAGMI --> WALLET_SEL

    APP --> MINT
    MINT --> DOWNLOAD

    APP --> HERO
    APP --> PRICING
    APP --> FAQ
```

User Flow Diagrams

Flow 1: First-Time Purchase & Installation

```
sequenceDiagram
    actor User
    participant Landing as Landing Page
    participant Wallet as MetaMask
    participant Scroll as Scroll Chain
    participant API as Backend API
    participant Desktop as Desktop App

    User->>Landing: Acessa radar.snelabs.space
    User->>Landing: Clica "Comprar Licença"
    Landing->>Wallet: Conectar wallet
    Wallet-->>Landing: Wallet conectada

    User->>Landing: Seleciona plano (30D/365D)
    Landing->>Scroll: Mint NFT License
    Scroll-->>Landing: TX confirmada

    User->>Landing: Clica "Download"
    Landing->>API: POST /api/download-token
    API-->>Landing: Token one-time
    Landing->>User: Download SNE_Radar_Setup.exe
    User->>Desktop: Instala e abre app
    Desktop->>Desktop: Detecta: não autenticado
    Desktop->>User: Mostra Lock Screen
```

Flow 2: Desktop Authentication (Deep Link)

```
sequenceDiagram
    actor User
    participant Desktop as Desktop App
    participant Browser as System Browser
    participant Landing as Landing Page
    participant Wallet as MetaMask
    participant API as Backend API

    User->>Desktop: Abre app (não autenticado)
    Desktop->>Desktop: Gera state + machine_id
    Desktop->>Browser: Abre radar.snelabs.space/auth?...

    Browser->>Landing: Carrega página auth
    User->>Wallet: Conecta wallet
    Wallet-->>Landing: Assinatura SIWE
    Landing->>API: POST /api/auth/verify
    API-->>Landing: Sessão criada

    Landing->>API: POST /api/auth/desktop-link
    API-->>Landing: code (60s, single-use)

    Landing->>Desktop: sneradar://auth?code=...&state=...
    Desktop->>Desktop: Valida state
    Desktop->>API: POST /api/auth/exchange
    API-->>Desktop: access_token + refresh_token

    Desktop->>Desktop: Armazena tokens (DPAPI)
    Desktop->>User: App desbloqueado! 🎉
```

Flow 3: Daily Usage (Authenticated)

```
sequenceDiagram
    actor User
    participant Desktop as Desktop App
    participant API as Backend API
    participant Binance as Binance API

    User->>Desktop: Abre app
    Desktop->>Desktop: Carrega tokens (DPAPI)
    Desktop->>Desktop: Verifica grace period (24h)

    alt Token válido no cache
```

```
        Desktop->>User: App abre normalmente
    else Precisa validar
        Desktop->>API: GET /api/auth/validate
        API-->>Desktop: Token válido
    end

    Desktop->>Binance: Busca dados de mercado
    Binance-->>Desktop: Candles, orderbook

    Desktop->>Desktop: Análise técnica
    Desktop->>User: Dashboard com oportunidades
```

Data Flow

```
flowchart LR
    subgraph "Market Data"
        BINANCE_API[Binance API]
        BYBIT_API[Bybit API]
    end

    subgraph "Processing"
        CACHE[(Cache TTL)]
        INDICATORS[Indicators Engine]
        ML[ML Predictions]
        PATTERNS[Pattern Detection]
    end

    subgraph "Output"
        DASHBOARD[Dashboard UI]
        ALERTS[Alert System]
        TELEGRAM_OUT[Telegram Notifications]
    end

    BINANCE_API --> CACHE
    BYBIT_API --> CACHE

    CACHE --> INDICATORS
    INDICATORS --> ML
    INDICATORS --> PATTERNS

    ML --> DASHBOARD
    PATTERNS --> DASHBOARD
    ML --> ALERTS
    ALERTS --> TELEGRAM_OUT
```

Deployment Topology

```
graph TB
    subgraph "User's Machine"
        WIN[Windows 10/11]
        EXE[SNE_Radar.exe]
        LOCAL_DB[(SQLite)]
        AUTH_FILE[auth.enc]
    end

    subgraph "Vercel"
        LANDING_DEPLOY[Landing Page<br/>radar.snelabs.space]
    end

    subgraph "Render / GCP"
        API_DEPLOY[Backend API<br/>api.snelabs.space]
        POSTGRES[(PostgreSQL)]
    end

    subgraph "Blockchain"
        SCROLL_CHAIN[Scroll L2]
        NFT_CONTRACT[License NFT Contract]
    end

    EXE --> LOCAL_DB
    EXE --> AUTH_FILE
    EXE -.->|HTTPS| API_DEPLOY

    LANDING_DEPLOY -.->|HTTPS| API_DEPLOY
    LANDING_DEPLOY -.->|RPC| SCROLL_CHAIN

    API_DEPLOY --> POSTGRES
    SCROLL_CHAIN --> NFT_CONTRACT
```

Technology Stack Summary

Layer	Technology	Purpose
Desktop Wrapper	pywebview	Native window
Desktop Backend	Flask + SocketIO	API + Real-time
Desktop Frontend	Vue.js 3 + Vite	UI Components
Landing Page	React + Vite	Sales + Auth
Web3	wagmi + viem	Wallet integration
Auth	SIWE	Sign-In with Ethereum
Blockchain	Scroll L2	NFT Licenses
Backend API	Flask + PostgreSQL	Central services

Bot	python-telegram-bot	Notifications
Packaging	PyInstaller + Inno Setup	Windows distribution