Eurlquid API Guide

Dokumen ini menjelaskan integrasi dan cara uji endpoint untuk:

- Groq (LLM orchestration)
- The Graph (Subgraph via ID)
- DeFiLlama (harga publik)
- Alchemy (RPC/WS, contoh latest block)

Semua endpoint berjalan pada Next.js App Router. Gunakan hanya dari server atau API Routes (jangan expose secrets ke client).

1) Prasyarat & Konfigurasi Environment

Buat file `.env.local` dari `.env.example` dan isi placeholder berikut:

- Groq
  - GROQ_API_KEY=...
  - (opsional) GROQ_API_URL=https://api.groq.com/openai/v1/chat/completions
- Alchemy
  - ALCHEMY_API_KEY=...
  - (opsional, override penuh)
    - ALCHEMY_HTTP_ETHEREUM=...
    - ALCHEMY_HTTP_ARBITRUM=...
    - ALCHEMY_HTTP_BASE=...
    - ALCHEMY_WS_ETHEREUM=...
    - ALCHEMY_WS_ARBITRUM=...
    - ALCHEMY_WS_BASE=...
- The Graph
  - (opsional, rekomendasi) THEGRAPH_API_KEY=... → gunakan gateway
  - (opsional, override ID subgraph)
    - THEGRAPH_SUBGRAPH_ID_UNISWAP_V3_ETHEREUM
    - THEGRAPH_SUBGRAPH_ID_UNISWAP_V2_ETHEREUM
    - THEGRAPH_SUBGRAPH_ID_CURVE_ETHEREUM
    - THEGRAPH_SUBGRAPH_ID_CURVE_ARBITRUM
    - THEGRAPH_SUBGRAPH_ID_BALANCER_ETHEREUM
- Wallet (publik)
  - NEXT_PUBLIC_WALLETCONNECT_PROJECT_ID=...

Jalankan dev server:

```
npm run dev
# atau
yarn dev
```

Akses: http://localhost:3000

Terkait keamanan: simpan secrets hanya di `.env.local`. Jangan commit ke repo.

---

2) Ringkasan Endpoint

- Groq
  - GET /api/groq → panggil Groq dengan mock data untuk uji cepat
  - POST /api/groq → panggil Groq dengan body JSON dari client/server
- The Graph
  - GET /api/thegraph/uniswap-v3-pools?first=5
  - GET /api/thegraph/balancer-pools?first=5
- DeFiLlama
  - GET /api/defillama/price?chain=ethereum&address=0x...
  - GET /api/defillama/1inch?chain=ethereum
- Analysis (Smart DEX Comparison)
  - POST /api/analysis → Compare all 4 DEXs (Uniswap V3, Curve, Balancer, 1inch)
- Alchemy
  - GET /api/alchemy/latest-block?chain=ethereum|arbitrum|base

Detail tiap endpoint ada di bawah.

---

3) Groq API

Sumber kode:

- API route: src/app/api/groq/route.ts
- Client: src/lib/groq.ts

Fungsi:

- GET: men-load mock JSON (src/mock/liquidity_oracle_input.json), mengirim ke Groq dengan SYSTEM_PROMPT, dan mengembalikan hasil.
- POST: menerima JSON arbitrer (data pasar/liquidity) dan meneruskannya ke Groq.

Contoh uji (GET):

```
curl -s http://localhost:3000/api/groq | jq .
```

Contoh uji (POST):

```
curl -s -X POST http://localhost:3000/api/groq \
  -H 'Content-Type: application/json' \
  -d '{
    "pair": "USDC/ETH",
    "chain": "ethereum",
    "window": "1h",
    "features": {
      "tvlUSD": 12000000,
      "volumeUSD_5m": 350000,
      "lpNetDelta_15m": -0.07,
      "whaleTransfersUSD_10m": 800000
    }
  }' | jq .
```

Respons berisi:

- parsed: hasil parsing JSON dari Groq (jika valid)
- raw: string mentah dari Groq
- groqResponse: objek response SDK Groq
- error: pesan error parse (jika ada)
- timestamp

Catatan:

- Model default: llama-3.3-70b-versatile
- Jangan gunakan `src/lib/groq.ts` di komponen client.

---

4) The Graph (Subgraph by ID)

Sumber kode:

- Lib: src/lib/thegraph.ts
- API uji: src/app/api/thegraph/uniswap-v3-pools/route.ts

Konstruksi URL (otomatis oleh lib):

- Dengan API key (disarankan, lebih andal):
  - https://gateway.thegraph.com/api/{THEGRAPH_API_KEY}/subgraphs/id/{SUBGRAPH_ID}
- Tanpa API key (fallback hosted):
  - https://api.thegraph.com/subgraphs/id/{SUBGRAPH_ID}

Subgraph IDs default (dapat di-override lewat env):

- Uniswap V3 (Ethereum): 5zvR82QoaXYFyDEKLZ9t6v9adgnptxYpKpSbxtgVENFV
- Uniswap V2 (Ethereum): A3Np3RQbaBA6oKJgiwDJeo5T3zrYfGHPWFYayMwtNDum
- Curve (Ethereum): 3fy93eAT56UJsRCEht8iFhfi6wjHWXtZ9dnnbQmvFopF
- Curve (Arbitrum): Gv6NJRut2zrm79ef4QHyKAm41YHqaLF392sM3cz9wywc
- Balancer (Ethereum): C4ayEZP2yTXRAB8vSaTrgN4m9anTe9Mdm2ViyiAuV9TV

Endpoint uji:

```
# Uniswap V3 pools
curl -s "http://localhost:3000/api/thegraph/uniswap-v3-pools?first=5" |
jq .

# Balancer pools
curl -s "http://localhost:3000/api/thegraph/balancer-pools?first=5" | jq
.
```

Parameter:

- first (opsional, default 5, maksimum 50)

Catatan:

- Saat ini route menggunakan subgraph Uniswap V3 Ethereum. Mapping chain lain dapat ditambah mudah di route.
- Jika mendapat error 4xx/5xx, pastikan THEGRAPH_API_KEY valid (atau fallback hosted siap dipakai), dan ID subgraph benar.

---

5) DeFiLlama (Harga Publik)

Sumber kode:

- Lib: src/lib/defillama.ts — jika belum ada, endpoint API route tetap berfungsi sebagai klien ringan
- API uji: src/app/api/defillama/price/route.ts

Endpoint:

```
# Token price from DeFiLlama
curl -s "http://localhost:3000/api/defillama/price?
chain=ethereum&address=0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48" | jq
.

# 1inch protocol data from DeFiLlama
curl -s "http://localhost:3000/api/defillama/1inch?chain=ethereum" | jq
.
```

Parameter:

- chain: ethereum | arbitrum | base
- address: alamat token ERC-20 (checksum atau lowercased) - untuk price endpoint

Keterangan:

- Tidak memerlukan API key.
- Price menggunakan coins.llama.fi/...
- 1inch menggunakan api.llama.fi/protocols untuk data protokol (TVL, volume, dll.)
- 1inch tidak memiliki subgraph publik, hanya mengandalkan DeFiLlama.

---

6) Smart DEX Analysis API

Sumber kode:

- API route: src/app/api/analysis/route.ts
- Store: src/store/userprompt-store.ts

Endpoint untuk analisis cerdas yang membandingkan 4 DEX sekaligus:

```
curl -s -X POST http://localhost:3000/api/analysis \
  -H 'Content-Type: application/json' \
  -d '{
    "fromToken": "USDC",
    "toToken": "ETH",
    "amount": "1000"
  }' | jq .
```

Parameter:

- fromToken: USDC | USDT | ETH | WETH (dapat diperluas)
- toToken: USDC | USDT | ETH | WETH
- amount: jumlah swap dalam string

DEX yang dianalisis:

1. **Uniswap V3**: Data dari TheGraph + DeFiLlama
2. **Curve**: Data dari TheGraph + DeFiLlama
3. **Balancer**: Data dari TheGraph + DeFiLlama
4. **1inch**: Data dari DeFiLlama saja (tidak ada subgraph)

Output:

- Analisis AI komprehensif menggunakan Groq LLM
- Rekomendasi route optimal dengan alokasi per-DEX
- Risk assessment dan timing predictions
- Expected slippage dan savings estimations

---

7) Alchemy (RPC/WS) – Latest Block

Sumber kode:

- Lib: src/lib/alchemy.ts
- API uji: src/app/api/alchemy/latest-block/route.ts

Endpoint:

```
curl -s "http://localhost:3000/api/alchemy/latest-block?chain=ethereum"
| jq .
```

Parameter:

- chain: ethereum | arbitrum | base

Keterangan:

- Jika tidak override endpoint HTTP/WS per-chain, lib akan membangun URL default berbasis `ALCHEMY_API_KEY`.
- WS client tersedia di lib untuk real-time (subscription), walau belum diekspos via route.

---

7) Wagmi/RainbowKit Chains (UI Wallet)

Konfigurasi:

- File: src/config/wagmi.ts
- Chains aktif: Ethereum mainnet, Arbitrum, Base, Sonic Blaze Testnet (demo)

Tidak berpengaruh ke server API di atas, namun diperlukan untuk koneksi wallet pada UI.

---

8) Troubleshooting Umum

- The Graph:
  - 400/403: pastikan THEGRAPH_API_KEY benar (jika pakai gateway) atau gunakan fallback hosted tanpa key.
  - 404: verifikasi SUBGRAPH_ID. Anda dapat override via `.env.local`.
  - Rate limit: gunakan `first` kecil (≤ 50); caching internal disarankan.
- Alchemy:
  - 401/403: cek ALCHEMY_API_KEY atau override URL HTTP/WS.
  - CORS tidak relevan untuk server-side; gunakan API route untuk klien.
- DeFiLlama:
  - Token tidak ditemukan: cek chain/address; gunakan checksum address resmi.

    - Rate limit: cache hasil 5–15 detik.

- Groq:
    - 401: cek GROQ_API_KEY.
    - Response bukan JSON valid: periksa `raw` dan `error`; sesuaikan SYSTEM_PROMPT atau lakukan postprocessing.

---

9) Rencana Lanjutan

- Tambahkan endpoints:
    - /api/thegraph/uniswap-v2-pairs
    - /api/thegraph/curve-pools?chain=ethereum|arbitrum
    - /api/defillama/balancer (dedicated endpoint)
    - /api/analysis/historical (time-series analysis)
- Integrasikan 1inch Aggregation API untuk real-time quotes
- Tambahkan Balancer V2 weighted pools analysis
- Tambah collector WS (Alchemy) untuk Swap/Mint/Burn real-time + rule-based alerts.
- Timeseries storage untuk swaps/snapshots/prices (Postgres/Timescale) untuk analitik
- Cross-chain DEX analysis (Arbitrum, Base, Polygon)