Ты прав — структуры не хватало. Я сделал **полноценный комплект** с расширенной файловой и конфигурационной базой, добавил **готовые пайплайны и джобы**, подтянул **правила/аккаунты/площадки** из твоих документов (где это возможно извлечь автоматически), и упаковал всё в один архив.

**Скачать готовый супер-парсер:**  
**➡️** [**super-parser-ultimate-pro.zip**](sandbox:/mnt/data/super-parser-ultimate-pro.zip)

# Что внутри (полный набор папок)

configs/ # все настройки

ai/prompts/ # тексты правил/советов/тонких фильтров ИИ (из твоих .docx)

derived/

accounts\_x.json # извлечённые @аккаунты X (без дублей)

hashtags.json # извлечённые хештеги

platforms\_extra.json# площадки/биржи из «площадки для листингов 30.docx»

rules.json # риск-правила, фильтры входа, безопасность, сессии, алерты

platforms.json # CEX/Launchpads (базовый список)

accounts.json # базовые офиц. аккаунты CEX + Tier1 инфлюенсеры

docs/

sources/\*.txt # распарсенные из .docx материалы (чтобы ничего не потерять)

reports/ # под отчёты

scripts/

send\_candles\_example.sh

samples/

candles.sample.json

data/

logs/

src/

index.js # запускает сервер, метрики, пайплайны и джобы

server/sse.js # /stream — поток сигналов для фронта

monitoring/

prometheus.js # /metrics Prometheus

sentry.js # интеграция Sentry

adapters/

queues/ (BullMQ + RabbitMQ)

chain/

solana.js # @solana/web3.js (QuickNode/Helius RPC)

evm.js # viem (EVM)

price/cex.js # место для твоего CEX API (Binance/OKX/Bybit/MEXC/Gate/HTX)

social/twitter.js # заглушка под X-интеграцию

storage/kv.js

providers/

helius.js # провайдер Helius

pipelines/

ingest\_onchain.js # Helius/QuickNode → свечи/события (сторедж в памяти)

ingest\_cex.js # листинги/цены CEX (ожидает твой API в adapters/price/cex.js)

jobs/

detect\_listings.js # сигнал 'listing'/'scheduled' → в /stream

whales.js # сигнал 'whale\_in' (спайк объёма)

signals/

engine.js # pump/dump/entry/exit/BB bounce (RSI/EMA/SMA/BB)

schema.js # описание формата аннотаций

tests/

test\_engine.js # юнит на ловлю пампа/дампа

test\_smoke.js # смоук-проверка

.env.sample

package.json

README.md

docker-compose.yml # Redis + (опц.) RabbitMQ

# Что я добавил поверх прошлой версии

* ✅ **Пайплайны** ingest\_onchain.js, ingest\_cex.js — сразу подключены в src/index.js.
* ✅ **Джобы** detect\_listings.js, whales.js — формируют аннотации listing/whale\_in для графика.
* ✅ **Конфиги из твоих документов**: извлёк @аккаунты, хештеги, площадки → положил в configs/derived/\*.json; сами тексты — в docs/sources/.
* ✅ **Полная структура** (scripts/reports/data/logs/samples/docs/…): ничего не потеряется при деплое/CI.
* ✅ **docker-compose.yml** — быстро поднять Redis и (опционально) RabbitMQ.

# Как запускать

cd super-parser-ultimate-pro

npm i

cp .env.sample .env # пропиши ключи/URL

docker compose up -d redis # или redis локально

# если нужен RabbitMQ:

# docker compose up -d rabbitmq && set QUEUE\_BACKEND=rabbitmq

node src/index.js

# Метрики: http://localhost:9110/metrics

# Поток SSE: http://localhost:8811/stream

# Анализ свечей: POST http://localhost:8811/signals/analyze

# Куда подставить твои реальные API

* **CEX**: src/adapters/price/cex.js  
  реализуй getListings(ticker) и getPrice(exchange, ticker) (я оставил точки подключения).
* **Helius/QuickNode**: src/pipelines/ingest\_onchain.js  
  дергай свои эндпоинты вместо заглушки toCandle() и наполняй store реальными свечами.
* **Голиб (3 модели ИИ)**: POST /advice проксируется на твой Golib-бэк (GOLIB\_URL, GOLIB\_KEY).  
  Фронт шлёт в /advice, парсер перенаправляет и возвращает ответ.

# Важно

* Некоторые файлы у тебя были в формате **.doc** (не .docx) — их автоматом в JSON не распарсить. Я **не потерял** их: положил в docs/sources/ (как есть). Если нужно — конвертируй в .docx, я прогоню повторный парс и пополню configs/derived/\*.
* Пороговые значения/правила можно тонко подстроить в configs/rules.json. Эти же правила может читать фронт и Голиб, чтобы говорить «одним языком».

Если хочешь, я:

* подключу прямо сейчас твой **публичный CEX API** в adapters/price/cex.js и **Helius** в ingest\_onchain.js;
* добавлю **ClickHouse/Postgres** для хранения свечей/сигналов;
* соберу **Dockerfile/CI** под прод.

Скажи только, какой порядок источников (Binance/OKX/Bybit/… и какие методы Helius/QuickNode) — и я прошью код внутрь пайплайнов.

Теперь всё выглядит **правильно и полноценно** 👌  
Структура на скрине совпадает с нужной для супер-парсера (без веба):

* configs/ (включая ai/prompts, derived/\*, rules.json, accounts.json, platforms.json)
* data/, logs/, reports/, samples/, scripts/
* src/ (pipelines, jobs, adapters, providers, monitoring, server, signals)
* tests/
* корень: .env.sample, docker-compose.yml, package.json, README.md

### Быстрый чек-лист (на всякий случай)

1. Скопируй .env.sample → .env и заполни ключи (HELIUS\_API\_KEY, RPC\_URL, GOLIB\_URL, GOLIB\_KEY, REDIS\_URL и т.д.).
2. Подними Redis (и, если нужно, RabbitMQ):
   * docker compose up -d redis (и при желании rabbitmq, затем QUEUE\_BACKEND=rabbitmq).
3. Запуск ядра: npm i && node src/index.js.
4. Проверка:
   * Метрики: http://localhost:9110/metrics
   * SSE-стрим сигналов: http://localhost:8811/stream
   * Юнит-тесты: npm test
   * Быстрый REST: POST /signals/analyze с samples/candles.sample.json.

### Где подключать твои реальные источники

* **CEX цены/листинги:** src/adapters/price/cex.js (реализуй getListings() и getPrice() под твой публичный API).
* **On-chain (Helius/QuickNode):** src/pipelines/ingest\_onchain.js (замени заглушку toCandle() на реальный маппинг свечей/событий).
* **ИИ Голиба ×3:** уже прокси через POST /advice (пропиши GOLIB\_URL/GOLIB\_KEY).

Если хочешь, прямо сейчас прошью твои реальные эндпоинты (Helius/QuickNode + Binance/OKX/Bybit/MEXC/Gate/HTX), добавлю хранилище (Postgres/ClickHouse) и CI/Dockerfile. Скажи, какие ключи/эндпоинты использовать — подключу.