# Техническое задание (сводка по проекту)

## Архитектура

[Apify Actors] ──► [Parser API] ──► [Event Bus/Queue (Redis/BullMQ)] ──► [Worker (Node)]

│ │ │

│ ├──► [FastAPI (REST)] ◄── UI (React)|

│ │ │

└──(Twitter/Telegram data) ├──► [Groq LLM] │

├──► [Helius/QuickNode RPC+Webhooks]│

├──► [RugCheck/Chainbase (optional)]│

└──► [PostgreSQL (metrics, logs)] │

User ──► UI (React+TS) ──► FastAPI ──► Queue ──► Worker ──► DEX/Alerts ──► UI/Logs

**Компоненты:**

* **UI:** React + TypeScript (Vite), Tailwind, shadcn/ui.
* **API:** Python **FastAPI** (оркестрация, REST, webhooks).
* **Worker/бот:** Node.js (интеграции Jupiter/Jito, анти-rug, триггеры).
* **Очередь:** Redis (BullMQ).
* **БД:** PostgreSQL (метрики/логи/кандидаты).
* **Интеграции:** Groq (LLM), Apify (акторы), Helius/QuickNode (Solana RPC, webhooks).
* **Обвязка:** Docker Compose, Nginx/Caddy, auto-deploy (GitHub Actions).

## Диаграмма потоков (основной сценарий «от слова до сделки»)

[Apify: Twitter/Telegram]

│ emits JSON (cleaned posts)

▼

[Parser API]

│ filters + normalize

▼

[Groq LLM]

│ score: relevance, hype, risk hints

▼

[FastAPI]

│ create Candidate + send to Queue

▼

[Redis/BullMQ Queue]

│ job: watch/mint/poolCreated

▼

[Worker (Node)]

├─► Helius Webhook subscribe (mint, poolCreated, addLiq, largeTx)

├─► QuickNode RPC (price/tx fallback)

├─► (optional) RugCheck/Chainbase quick safety

├─► generate Alert (UI + Telegram hook)

└─► (if manual confirm) Jupiter/Jito trade flow (sim/test)

## Диаграмма взаимодействия (sequence)

UI → FastAPI: POST /parser/run (terms | sources)

FastAPI → Apify: runActor(...)

Apify → FastAPI: callback/webhook with items

FastAPI → Groq: POST /chat (rank terms)

FastAPI → Redis: enqueue(job={mint\_watch, token})

Worker → Helius: subscribe(webhooks)

Helius → Worker: event(mint or poolCreated)

Worker → (RugCheck/Chainbase): safety checks (optional)

Worker → FastAPI: POST /alerts (store & push)

UI → FastAPI: GET /candidates, /alerts

(Manual) UI → FastAPI: POST /trade/confirm → Queue → Worker → Jupiter/Jito (sim/prod)

## Основные эндпоинты (предварительно)

* POST /api/parser/run — запустить актор Apify с входными словами/источниками.
* POST /api/apify/callback — вебхук с результатами парсинга.
* POST /api/helius/webhook — вебхук ончейн-событий (mint/pool/liq).
* GET /api/candidates — список кандидатов + скоринг.
* GET /api/alerts — алерты (LLM hints, whale, liq).
* POST /api/trade/confirm — подтверждение ручной сделки (симуляция/прод).
* GET /api/health — здоровье сервисов.

## ENV (пример)

# LLM / Parsers

GROQ\_API\_KEY=\*\*\*

APIFY\_TOKEN=\*\*\*

# Solana

HELIUS\_API\_KEY=\*\*\*

QUICKNODE\_HTTP=\*\*\*

JUPITER\_BASE\_URL=https://quote-api.jup.ag

JITO\_TIP=0.0001

# Infra

REDIS\_URL=redis://redis:6379

DATABASE\_URL=postgresql://alpha:alpha@postgres:5432/alpha2

BASE\_URL=https://app.my-domain.com

SNAPSHOT\_INTERVAL\_SECONDS=60

NODE\_ENV=production

## Docker Compose (скелет)

version: "3.9"

services:

ui:

build: ./ui

ports: ["3000:3000"]

env\_file: [.env]

depends\_on: [api]

api:

build: ./api

ports: ["8000:8000"]

env\_file: [.env]

depends\_on: [postgres, redis]

worker:

build: ./worker

env\_file: [.env]

depends\_on: [redis, api]

redis:

image: redis:7-alpine

ports: ["6379:6379"]

postgres:

image: postgres:15-alpine

environment:

POSTGRES\_USER: alpha

POSTGRES\_PASSWORD: alpha

POSTGRES\_DB: alpha2

volumes: [pgdata:/var/lib/postgresql/data]

ports: ["5432:5432"]

proxy:

image: caddy:2-alpine

volumes:

- ./deploy/Caddyfile:/etc/caddy/Caddyfile

ports: ["80:80","443:443"]

depends\_on: [ui, api]

volumes:

pgdata:

## Автодеплой (GitHub Actions — outline)

* Триггер: push в main.
* Джобы:
  1. **Build & push** контейнеров в GHCR/Hub.
  2. **SSH deploy** на VPS: docker compose pull && docker compose up -d.
* Артефакты: .env.example, deploy/Caddyfile или deploy/nginx.conf.

## Что нужно сделать фрилансеру (спринт «завершение»)

1. **Свести модули** (UI ↔ FastAPI ↔ Worker) по указанным REST/очередям.
2. **Подключить** Groq, Apify, Helius/QuickNode (под тестовыми ключами фрилансера).
3. Настроить **Helius Webhooks** (mint, poolCreated, addLiq, largeTx).
4. Добавить быстрые **security-чеки** (опционально RugCheck/Chainbase — stub/adapter).
5. Собрать **Docker** и развернуть на нашем VPS (Ubuntu 22.04, 4vCPU/8GB).
6. Настроить **TLS** (Caddy/Nginx), базовые логи и health-checks.
7. Включить **auto-deploy** (GitHub Actions) и отдать README.md + .env.example.
8. Провести **E2E-прогон**: парсинг → скоринг → webhook → алерт → ручной confirm (симуляция трейда) → логи.

## Критерии приёмки

* docker compose up -d поднимает все сервисы без ошибок.
* UI показывает **Candidates/Alerts** с актуальными событиями.
* Webhooks Helius приходят и отражаются в алертах.
* Кнопка **Buy (симуляция)** проходит полный цикл до лога ордера.
* Все ключи в проде — через .env (без хардкода), ключи заказчика не запрашиваются.

Если ей понадобится, могу выслать **минимальные контракты интерфейсов** (JSON-схемы для /parser/run, /apify/callback, /helius/webhook)