Коротко для **Alpha-2**:

1. **Парсинг X/Twitter → фильтр “мем-слов” → список ключей** за 24ч.
2. Список уходит в **Helius**: вебхук(и) на **TOKEN\_MINT / poolCreated** + DAS-проверка только по кандидатам.
3. **UI-ввод слов:** я могу вручную добавить 10–15 слов/фраз (через форму), они сразу попадают в список мониторинга Helius.
4. **Матчинг имён/метаданных:** если совпало (пример: “frogme”), система шлёт **ALERT: “создан токен по слову ‘frogme’ (mint, ссылка, время)”**.
5. **Антишум:** дедуп по mint, отсечка по программам (pump.fun), минимальные safety-флаги (LP, owner %, honeypot).
6. **Логи в UI:** последние срабатывания, статус вебхуков, включить/выключить слово одной кнопкой.

**Можно ли сделать “мини-парсер” под ваши слова → алерт “token created on pump.fun”?** — Да.  
Мини-схема:

1. Парсер X/Twitter отдаёт список **ключевых слов/имен токенов** за 24ч (как вы и делаете).
2. В Helius ставите **вебхук на TOKEN\_MINT** (Enhanced), вытаскиваете mint из transferTokens[0].mint и проверяете метаданные через **DAS** getAsset. [Helius+2Helius+2](https://www.helius.dev/blog/how-to-fetch-newly-minted-tokens-with-helius?utm_source=chatgpt.com)
3. Для pump.fun-старта можно подключить **gRPC Pump AMM stream** — даёт realtime по запускам/свапам (быстрее для снайпа). [Helius](https://www.helius.dev/docs/grpc/stream-pump-amm-data?utm_source=chatgpt.com)
4. Матчите: если имя/тег/token-name попал под ваши слова → шлёте **ALERT** “создан на pump.fun”.

**Сколько это выйдет (оценка):**

* При строгих фильтрах (после ваших слов) вебхуков немного: скажем, **до 10–200 валидных событий/день** ⇒ **300–6,000 кредитов/мес** — укладывается в **Free (1M)** с огромным запасом. Если подключите gRPC-стрим “широко” без фильтров, расход вырастет (gRPC тарифицируется по объёму), но всё равно обычно хватает Dev-плана. [Helius+1](https://helius.mintlify.app/billing/plans-and-rate-limits?utm_source=chatgpt.com)

**Как сэкономить:**

* Делайте **дедуп по mint** + предварительную отсечку по программам/парам (pump.fun), иначе будет шум. [Solana Stack Exchange](https://solana.stackexchange.com/questions/13204/helius-webhook-for-get-real-time-information-about-new-tokens-created?utm_source=chatgpt.com)
* Гоняйте **вебхук только по нужным типам** (TOKEN\_MINT) и фильтруйте на своей стороне, а DAS дергайте **только** для кандидатов. [Helius+1](https://www.helius.dev/blog/how-to-fetch-newly-minted-tokens-with-helius?utm_source=chatgpt.com)
* Начните с **Free**, при упоре — переход на **$49 Dev** (10M кредитов + $5 за каждый доп. миллион).

Отлично, Брат ✊. Фиксирую замысел и даю чёткий план Альфа-2: как собрать **ИИ Голиба + парсер (Apify+Grok) + Helius-мониторинг** с ручной подгрузкой до **20 слов/названий** и мгновенными алертами о **mint-адресе** (в т.ч. pump.fun).

# Что делаем в Альфа-2 (коротко)

* **Парсер X (Twitter)** через Apify + **Grok NLP** расширяет/очищает слова →
* **Очистка/дедуп/фильтры** (только оригиналы, последние 24ч, whitelist авторов) →
* **Отбрасываем мусор**, остаются «триггер-слова» и имена токенов →
* Список уходит в **Helius Tracker**, плюс ты **ручно** добавляешь до **20 слов/названий** через UI →
* **Helius Webhooks** слушают newTokenMint, poolCreated (pump.fun / Raydium) →
* При совпадении по слову/названию → **алерт** (TG/веб-уведомление) + краткая авто-проверка (5 red flags) →
* (Опция) сразу открываем график (dexscreener / pump.fun).

# Архитектура (MVP)

## Сервисы

1. **Parser Service (Apify + Grok)**

* Забирает твиты по whitelist аккаунтам, «последние 24ч», только **оригиналы**.
* Grok чистит и расширяет словарь (синонимы/варианты написания, эмодзи-коды).
* Результат: массив **triggerTerms** (уникальные строки).

1. **Word Hub (UI + API)**

* Вкладка **“Tracked Terms”**:
  + Поле «+ Add terms» (до **20** вручную).
  + Переключатель «Include Parser Terms» (включить/выключить автодобавление из парсера).
  + Кнопки **Start/Stop Monitoring**.
* Хранение: Postgres (таблицы ниже).

1. **Helius Tracker**

* Webhook подписки:
  + newTokenMint (все новые SPL)
  + poolCreated (Raydium/Orca)
* Воркер **MintMatcher**: матчит новые мины с нашим списком terms (смарт-алгоритм: exact + fuzzy + token name).
* Дедуп по mint (чтобы не спамить).

1. **Alert Engine**

* Если найдено совпадение:
  + Формируем карточку: **mint**, имя/символ, время, источник (pump.fun/др.), краткие проверки (5 ред-флагов), ссылки (Solscan/Dexscreener).
  + Отправляем в TG-бот + пуш в веб-UI.

# Таблицы (Postgres)

**terms**

* id (pk)
* term (text, unique, lower)
* source (enum: manual | parser)
* enabled (bool)
* created\_at

**mints**

* mint (pk)
* name, symbol
* source (pumpfun|other)
* created\_at

**term\_hits**

* id (pk)
* term\_id (fk → terms.id)
* mint (fk → mints.mint)
* score (float, FuzzyMatch 0–1)
* matched\_on (enum: name | symbol | tweet\_text)
* created\_at

**alerts**

* id (pk)
* mint (fk)
* risk\_flags (jsonb)
* sent\_to\_tg (bool), sent\_to\_ui (bool)
* created\_at

# API (быстро подключаем Dev’у)

**POST /api/terms**

{ "terms": ["doge", "frogme", "pepe ai"], "source": "manual" }

**GET /api/terms** → список, с флагом enabled.

**POST /api/monitoring/start**  
**POST /api/monitoring/stop**

**POST /webhooks/helius**

{

"type": "newTokenMint",

"mint": "XXXXXXXX",

"name": "FROGME",

"symbol": "GA",

"metadata": { "program":"pump.fun","owner":"..." },

"ts": 1699999999

}

# Логика матчинга (MintMatcher)

1. **Exact**: term == name|symbol (регистронезависимо).
2. **Fuzzy**: tokenName включает term, Jaro-Winkler ≥ 0.88, отбрасываем слишком общие («coin», «moon»).
3. **Alias-expansion**: из парсера (Grok) подставляем варианты (frogme → frog me, 🐸me).
4. **Дедуп**: один алерт на mint в первые 10 мин.
5. **Приоритет**: manual terms > parser terms.

# Быстрые проверки (5 red flags, ≤ 1 сек)

* **LP locked/burn**: неизвестно → ⚠️
* **Dev wallet %** > X% → ⚠️
* **Trading rules** (honeypot / transfer tax > Y%) → ⚠️
* **Top-10 holders** > 60% → ⚠️
* **Migrations > 0** (ранние клоны) → ⚠️  
  (Можно через RugCheck/Chainbase/GoPlus — любой доступный быстрый эндпоинт.)

# Карточка алерта (TG/веб)

🆕 Mint match: FROGME ($GA)

Mint: GHZiG8iSVN...pump

 Term hit: "frogme" (score 0.92, exact)

⏱ Detected: 2m ago | Source: pump.fun

⚑ Quick risk:

- LP lock: unknown (⚠️)

- Dev%: 7.8% (⚠️>5%)

- Top10: 22.7% (ok)

- Rules: tax 5.8% / 4.8% (ok)

- Migrations: 0 (ok)

 Solscan | Dexscreener | Open chart

# Фильтры парсера X (Apify+Grok)

* **Только whitelist авторов** (твои топ-аккаунты).
* **Только оригиналы** (без ретвитов/реплаев/рекламы).
* **Последние 24 часа**.
* **Семантика**: Grok извлекает кандидатов (имена/хэштеги/мем-фразы), чистит спам, нормализует.
* **Лимиты**: отдаём не более N новых terms за окно (например 50/сутки), чтобы не зашумлять Helius.

# UI (что важно добавить)

* Вкладка **Tracked Terms**: список, тег manual|parser, включить/выключить, счётчик «совпадений за 24ч».
* Кнопка **Upload CSV** (быстро загрузить 10–20 слов).
* Ползунок **Fuzzy threshold** (0.85–0.95).
* Тумблер **Pump.fun only / All sources**.
* Тумблер **Auto open chart** при алерте.

# Анти-шум и стабильность

* **Rate-limit/backoff** для внешних API.
* **Очередь** (BullMQ) для обработки webhooks (чтобы не терять пики).
* **Cache** для «известных плохих» шаблонов (бан-лист terms).
* **Health-метрики**: алерты в Grafana/Sentry.

# Тест-план перед продом (1 день)

1. Симуляция 50 событий newTokenMint → проверка дедуп/скорости.
2. Матчинг edge-кейсов (эмодзи/разделители/регистры).
3. Ручная загрузка 20 слов → проверка, что именно они в приоритете.
4. TG-уведомления и веб-плашки приходят < 2 сек.

# Что от тебя нужно сейчас

Ничего публиковать не будем, понял. Просто подтвержди:

* Нужна ли опция **“Pump.fun only”** по умолчанию включённой?
* Хватаем **poolCreated** сразу же после **mint** (двойной алерт) или только mint-события?

Если ок, я подготовлю:

1. JSON-схемы таблиц/эндпоинтов (ready-to-use).
2. Шаблон карточки алерта (TG + UI).
3. Список технических задач для Dev’а (спринт на 2–3 дня) — без «поломать текущее».

Идём ровно. Альфа-1 — закрываем с ИИ Голиба и деплоем; Альфа-2 — запускаем этот поток «слова → Helius → mint-алерт» с твоей ручной загрузкой до 20 терминов. Это даст тебе **ранний вход** и контроль без каналов и публичности.

Коротко: **да, можно и правильно** — делаем «дополнение-пак» как отдельную папку/репозиторий, которое подключается **оверлеем** (compose/Helm), проксируется в UI как /addons, общается с ядрами по API и не трогает исходные ZIP’ы. Ниже — готовый план и структура, чтобы фрилансер просто «подкинул одну папку» и всё завелось.

# Как сделать без поломок исходников

## Подход

* **Add-on (sidecar/overlay)** вместо «вшивания».
* Поднимаем новые сервисы **отдельно**, подключаемся к существующим через **HTTP API/Webhook**.
* Включаем их через docker-compose.override.yml (или helm values), без изменений в оригинальных архивах.

## Структура «одной папки» (repo alpha-addons/)

alpha-addons/

README.md

.env.addons.example

docker-compose.override.yml # оверлей к базовому compose

configs/

addons.yml # фичефлаги, пороги фильтров, списки

routing.nginx.conf # прокси /addons → ui-addon

migrations/

001\_addons\_init.sql # ТОЛЬКО новые схемы/таблицы

ui-addon/ # отдельный фронт для расширенных настроек

Dockerfile

dist/ or src/

ai-rules/ # «ИИ Голиба — доп. фильтры/правила»

Dockerfile

app/ (REST: /score, /rules, /explain)

parser-filters/ # доп. очистка/фильтры для парсера

Dockerfile

app/ (REST: /filterTerms, /expandTerms)

bridge/ # тонкий шлюз к ядрам (Adapter)

Dockerfile

app/ (talks to: Alpha1 API, Parser API, Helius Webhooks)

scripts/

install\_addons.sh # применить оверлей, миграции, .env

rollback\_addons.sh # откат

## Принципы интеграции (важно)

1. **Никаких правок** в трёх исходных ZIP.
2. **Соглашение об интерфейсах**:
   * UI ядра ↔ bridge через POST /addons/\*
   * Parser ядра ↔ parser-filters через POST /filterTerms (вход: список слов; выход: очищенный/расширенный список)
   * Core AI ↔ ai-rules через POST /score (политики, правила, risk-score)
   * Helius Webhook ↔ bridge (/webhooks/helius) — матчинг и доп. логика без изменения основного вебхука
3. **Reverse proxy**:
   * Nginx/Traefik добавляет маршрут /addons → ui-addon (не трогаем основной UI).
   * Кнопка «Расширенные фильтры/ИИ» в основном UI просто открывает /addons во фрейме/новой вкладке.
4. **База данных**
   * Все **новые таблицы** — в своей схеме, например addons\_\*.
   * **Не меняем** существующие таблицы ядра. Если нужно читать — читаем **read-only**.
   * Если нужно «сшить» данные, используем **view/materialized view** или bridge для кросс-запросов.
5. **Фичефлаги и пороги**
   * Все новые настройки — в configs/addons.yml и .env.addons.
   * Никаких новых ключей в старых .env.
6. **Сборка/запуск**
   * Базовый docker-compose.yml — как есть.
   * Наш docker-compose.override.yml добавляет 3–4 сервиса и Nginx-роут.
   * Запуск: ./scripts/install\_addons.sh → миграции → docker compose up -d.

## Пример docker-compose.override.yml (фрагмент)

services:

ui-addon:

build: ./ui-addon

env\_file:

- .env.addons

networks: [default]

ai-rules:

build: ./ai-rules

env\_file:

- .env.addons

networks: [default]

parser-filters:

build: ./parser-filters

env\_file:

- .env.addons

networks: [default]

bridge:

build: ./bridge

env\_file:

- .env.addons

depends\_on: [ai-rules, parser-filters]

networks: [default]

nginx:

volumes:

- ./configs/routing.nginx.conf:/etc/nginx/conf.d/addons.conf:ro

## Контракты (минимум, чтобы Dev сразу писал)

### POST /addons/filterTerms (→ parser-filters)

{ "terms": ["frogme", "pepe ai"], "mode":"expand|clean", "lang":"en|ru" }

**200 →**

{ "terms": ["frogme","frog me","pepe ai","pepe-ai"] }

### POST /addons/score (→ ai-rules)

{

"token": { "mint":"...", "name":"...", "symbol":"..." },

"metrics": { "cap":11200, "holders":35, "top10":22.7, "fees":{"b":5.8,"s":4.8} },

"security": { "lpLocked":false, "devPct":7.8, "migrations":0 },

"social": { "triggerTerm":"frogme", "from":"whitelist" }

}

**200 →**

{

"riskScore": 0.63,

"flags": ["LP\_UNLOCKED","DEV\_PCT>5%"],

"explain": ["LP не залочен","Доля dev кошелька выше порога 5%"]

}

### POST /webhooks/helius (→ bridge)

— получает newTokenMint/poolCreated, дергает ai-rules и при необходимости шлёт алерт в ваш TG/веб.

## Где может понадобиться «вшивание» (редко)

* **Менять схему БД ядра** — не нужно. Если совсем необходимо, делаем **shadow-таблицы** и отображаем через view.
* **Добавлять кнопки в старый UI** — делаем это **через конфиг-меню/рамку** (ссылка на /addons) либо через **минимальный PR**, но это необязательно.

## Деплой/откат

* Установка: ./scripts/install\_addons.sh
  1. копирует .env.addons
  2. прогоняет migrations/\*.sql в **новой схеме**
  3. docker compose -f docker-compose.yml -f docker-compose.override.yml up -d
* Откат: ./scripts/rollback\_addons.sh (останавливает сервисы add-on, возвращает старый прокси-конфиг).

## Версионирование и совместимость

* Семвер: addons v1.x совместимы с текущими API ядра.
* **Pin** версии эндпоинтов ядра в bridge (например, ALPHA1\_API=v1), чтобы апгрейд ядра не ломал add-on.
* Контроль контрактов — через простые **контракт-тесты** в bridge.

### Итог

Да, **делаем «одну папку-дополнение»** без изменения трёх уже готовых систем. Это безопасно, прозрачно, легко ставится и снимается. Когда фрилансер придёт — он:

1. положит alpha-addons/ рядом с основными,
2. выполнит install\_addons.sh,
3. включит маршруты /addons — и у тебя появятся расширенные фильтры, правила ИИ Голиба и прокачанный парсер **без риска поломать базу**.

Если хочешь, назову точные имена сервисов/портов под твой текущий compose и подготовлю install\_addons.sh прямо под твой сервер.

Брат, принял. Вот чёткий «план-апгрейд» для **Альфа-2**: усиливаем ИИ Голиба в 2 раза по фильтрам, жёстче настраиваем парсер (Apify+Grok) с экономией лимитов, и докручиваем веб-интерфейс (ручные слова→Helius→таймер-алерты + быстрые кнопки «смотреть график»). Всё — как доп-модуль без поломки Альфа-1.

# 1) ИИ Голиба — «V2 Risk & Pump Engine»

## 1.1 Фильтры (x2 к Альфа-1)

**Безопасность/контракт**

* LP: lock/burn/ре-лок, доля залоченной ликвидности, скорость снятия LP.
* Dev-владение % (по адресным кластерам), перемещения dev→CEX/DEX.
* Honeypot/transfer-tax, trading-cooldown, blacklist/whitelist funcs.
* Миграции/форки контракта; прокси-апгрейды; mint-authority, freeze-authority.
* Анти-wash: корреляция большого процента объёма между связанными адресами.
* Топ-holders: Top10/Top20 + динамика их долей (распределение/концентрация).

**Ончейн-потоки**

* Whale inflow (по проверенному списку кит-адресов), размер первой покупки, «лестница» докупок.
* Появление крупных ордеров в первые N секунд после пула (стакан/своп-суммы).
* Скорость прироста холдеров в первые 5/15/60 минут.

**Соц-сигналы (из парсера)**

* Оригинальные твиты топ-аккаунтов за 24ч, engagement-порог (лайки/ретвиты).
* «Мем-паттерны» (фразы/эмодзи/солянки слов), тональность/сарказм (Grok).
* Всплеск упоминаний + связка со словарём alias’ов (DOGE↔DogeDay, 🐸↔frog).

**Рынок/микроструктура**

* Сдвиг цены vs. приток ликвидности (disbalance), волатильность по минутам.
* Спред/просадки на первых сделках, «скользящая проскальзывание».
* Аномалии комиссии (bundle, priority fee) в момент старта.

## 1.2 Скоринг и логика (пример)

* **RiskScore** = w₁Sec + w₂LP + w₃Dev% + w₄Holders + w₅Honeypot + w₆Wash + w₇Taxes + w₈Proxy
* **PumpScore** = a₁Whales + a₂SocialBurst + a₃VolAccel + a₄HoldersΔ + a₅\*SpreadTightening
* **Гейты**: не торговать, если RiskScore > 0.7 ИЛИ (LP unlocked & Dev%>5% & Taxes>8%).
* **Режимы**:
  + Scout (без покупки) — только алерты и карточки.
  + Snipe — покупка при PumpScore≥T, RiskScore≤R, «кит вошёл», «LP OK».
* **Анти-RUG**: мгновенный выход при LP-out или dev-dump > X% от капы.

**Мини-пример конфигурации (JSON)**

{

"risk": { "lp\_lock\_min\_pct": 70, "dev\_pct\_max": 5, "tax\_max": 8, "top10\_max\_pct": 60 },

"pump": { "whale\_buy\_min\_sol": 50, "social\_burst\_min": 3, "vol\_accel\_factor": 1.8 },

"gates": { "risk\_max": 0.7, "pump\_min": 0.62 },

"modes": { "scout": true, "snipe": false }

}

# 2) Парсер (Apify+Grok) — жёстче и дешевле

## 2.1 Жёсткие фильтры ввода

* **Только whitelist аккаунтов** (твой список звёзд/аналитиков).
* **Только оригиналы** (без ретвитов/реплаев/промо).
* **Окно 24ч**, шаги 15–30 мин.
* **Engagement-пороги**: лайки/ретвиты ≥ N (по рангу автора).
* **Язык**: en/ru/es; отбрасывать смешанные спам-шаблоны.
* **Бан-слова**: generic мусор (moon, pump, go to 100x без контекста).
* **De-dup**: по тексту/линкам/ID; fuzzy 0.92+.

## 2.2 Нормализация терминов (меньше запросов наружу)

* Расширение alias’ов локально (словарь эмодзи↔слова, splitCamelCase, синонимы).
* Частотный отсев (оставляем top-N термов за окно).
* Кэширование результатов по авторам/тредам (чтобы не дёргать API повторно).
* Локальная ML-классификация (Grok оффлайн режим/тонкая модель) перед внешними вызовами.

**Пример входа актору Apify (минимум)**

{

"searchQueries": [],

"profiles": ["elonmusk","APompliano","SatoshiLite"],

"sinceHours": 24,

"onlyOriginal": true,

"minEngagement": { "likes": 50, "retweets": 10 },

"lang": ["en","ru"],

"maxItems": 300

}

**Фильтрация на выходе**

{

"allowTerms": ["doge","pepe ai","frogme"],

"denyTerms": ["moon","pump","giveaway","airdrop"],

"fuzzy": 0.9,

"limitPerWindow": 50

}

# 3) Веб-интерфейс — контроль руками + быстрые «смотреть»

## 3.1 Вкладка ****Tracked Terms****

* Поле **+ Добавить слова/названия** (до 20) → сохраняются как source=manual.
* Тумблера: **Include Parser Terms**, **Pump.fun only**, **Auto-open chart**.
* **Fuzzy threshold** слайдер (0.85–0.95).
* **Start/Stop Monitoring** (для Helius-подписок).
* Счётчики: «совпадений за 24ч», «активных термов».

## 3.2 Таймер-алерты (Helius)

* Подписки newTokenMint и poolCreated.
* Матчинг: exact + fuzzy + alias-словарь.
* Дедуп по mint (10 мин).
* Push в TG и UI.

## 3.3 Кнопки ****Смотреть****

* На карточке токена:
  + **Pump.fun**, **Birdeye**, **Dexscreener**, **Solscan** (открыть в новом окне).
  + (Опция) **CEX Radar** (отдельная вкладка: агрегация листингов/слухов).

# 4) Подключаем сервисы-усилители (как адаптеры)

Делаем «тонкие адаптеры» в нашем add-on, чтобы не менять ядро.

* **CoinMarketCap**: метаданные токена/символы/история (когда есть CMC id).
* **Chainbase**: быстрые risk-checks/labels/новые минты (API-фичи под безопасность).
* **Nansen** (если доступен): киты/лейблы/смарт-деньги (иначе — фолбэк Arkham/Flipside).
* **TokenMetrics** (если нужен): sentiment/рыночные метрики; используем как дополнительный сигнал.
* **Solscan AI**: аннотации по контракту/транзакциям, быстрые ссылки на анализ.

**Храним только необходимые поля** (id, label, riskFlags, updatedAt) — минимизируем стоимость и задержки.

# 5) Резервирование и подстраховка Альфа-1

* **Фолбэк RPC/Webhooks**: Helius → QuickNode (авто-переключение).
* Очереди: BullMQ для вебхуков (burst-устойчивость).
* Кэш: на частые справочники (alias, labels, risk).
* Health-чеки: heartbeat всех интеграций + алерты в TG при деградации.
* Режим **Independent**: Альфа-2 продолжает мониторинг и алерты даже если Альфа-1 лежит.

# 6) Что отдаём фрилансеру (готовые артефакты)

* ai\_weights\_v2.json — веса и пороги Risk/Pump.
* parser\_filters\_v2.json — входные/выходные фильтры, бан-слова, fuzzy.
* terms\_schema.sql — таблицы addons\_terms, addons\_term\_hits, addons\_alerts.
* addons\_api.md — эндпойнты: /addons/terms, /addons/monitoring, /webhooks/helius, /addons/score.
* ui\_spec.md — скрин-флоу вкладки **Tracked Terms** и карточек с кнопками «смотреть».

# 7) Мини-критерии готовности (acceptance)

* Добавление до **20** ручных слов работает, матчинг — <2 сек до алерта.
* Дедуп mint-событий и двойных алертов (mint/pool).
* Risk/Pump-скоринг возвращается за ≤500 мс.
* Кнопки «смотреть» открывают правильные страницы (pump.fun, birdeye, dexscreener, solscan).
* При падении Helius автосвитч на резервный провайдер.

Хочешь — я сразу сгенерирую **примерные файлы**:

* ai\_weights\_v2.json
* parser\_filters\_v2.json
* addons\_api.md

чтобы ты их просто передал фрилансеру и не ждал никакой «вшивки» в ядро. Если ок — сделаю их прямо сейчас.

Брат, супер-логика: идём **поэтапно** — (1) допиливаем ИИ Голиба, (2) ужимаем и усиливаем парсер, (3) докручиваем веб-интерфейс. Ниже — чёткий план работ + реальный бюджет по сервисам (актуальные цены проверил).

# Этапы и «Definition of Done»

## 1) ИИ Голиба (V2)

**Что добавляем (поверх Альфа-1):**

* Контракт/безопасность: LP lock/burn/релок, dev% c кластеризацией адресов, honeypot/налоги, proxy-upgrade/mint-authority, миграции.
* Ончейн-поведение: whale inflow (порог SOL), скорость прироста холдеров (5/15/60 мин), всплеск объёма/волатильность, спред/проскальзывание.
* Анти-манипуляции: wash-trading (доля оборота внутри кластера), «LP-инъекции» (быстрый ввод/вывод), dev-dump.
* Соц-сигналы: только whitelist авторов, оригиналы 24h, engagement-пороги, alias/эмодзи-нормализация.
* Две метрики: **RiskScore** и **PumpScore** + гейты (не торгуем при Risk>порога, входим при Pump≥T и «киты вошли»).
* Анти-RUG: мгновенный выход при LP-out/dev-dump.

**Мини-конфиг (готов для вшивания):**

{

"risk": { "lp\_lock\_min\_pct": 70, "dev\_pct\_max": 5, "tax\_max": 8, "top10\_max\_pct": 60, "proxy\_upgrade\_block": true },

"pump": { "whale\_buy\_min\_sol": 50, "social\_burst\_min": 3, "vol\_accel\_factor": 1.8, "holders\_5m\_min": 25 },

"gates": { "risk\_max": 0.70, "pump\_min": 0.62 },

"modes": { "scout": true, "snipe": false },

"weights": {

"lp": 0.18, "dev\_pct": 0.15, "honeypot\_tax": 0.12, "wash": 0.12,

"whales": 0.18, "social": 0.10, "vol\_accel": 0.10, "holders\_delta": 0.05

},

"fuzzy\_match": 0.90,

"ban\_terms": ["airdrop","giveaway","moon","to the moon","free"],

"whale\_list\_source": "data/whales.csv"

}

**DoD:** Risk/Pump возвращаются ≤500 мс; флаги (LP\_UNLOCKED, DEV\_PCT>5%, WASH) проставляются корректно; анти-RUG триггерит выход.

## 2) Парсер (Apify + Grok) — «жёстко и дёшево»

**Правила экономии CUs и качества:**

* Только **whitelist** авторов, **только оригиналы**, окно **24h**, шаг запуска 15–30 мин.
* Engagement-пороги (по рангу автора), язык en/ru/es, строгий deny-лист.
* Fuzzy-dedup ≥0.92 (по тексту/URL/ID), локальный кэш результатов.
* Локальная нормализация терминов (alias/эмодзи/слитные слова) — меньше вызовов NLP.
* Лимит «top-N terms/окно» (например, 50), чтобы не зашумлять Helius.

**Пример входа актору Apify:**

{

"profiles": ["elonmusk","APompliano","SatoshiLite"],

"sinceHours": 24,

"onlyOriginal": true,

"minEngagement": { "likes": 50, "retweets": 10 },

"lang": ["en","ru"],

"maxItems": 300

}

**DoD:** На 5-часовом окне < X CUs/сутки (снижение на ~40–60% vs сейчас), термины чистые, без спама; отдаём не >50/окно.

## 3) Веб-интерфейс (минимум клик-добавки, без поломок)

* Вкладка **Tracked Terms**: ручной ввод до **20** слов/токенов (source=manual), тумблеры: Include Parser Terms, Pump.fun only, Auto-open chart, слайдер Fuzzy 0.85–0.95, кнопки Start/Stop Monitoring.
* **Helius-таймер-алерты**: подписки newTokenMint и poolCreated, матчинг exact+fuzzy+alias, дедуп по mint (10 мин), пуш в TG и UI.
* Кнопки **Смотреть** на карточке токена: **Pump.fun / Birdeye / Dexscreener / Solscan** (открыть в новом окне).  
  **DoD:** ручные 20 слов ловятся <2 с от события, двойных алертов нет, кнопки открывают правильные страницы.

# Бюджет (ежемесячно, ориентиры)

* **GroqCloud (LLM-инференс):** напр. Llama 3.1 8B — **$0.05/M вход** и **$0.08/M выход токенов**; есть и другие модели (см. таблицу на странице). ([Groq](https://groq.com/pricing))  
  (Планы GroqCloud с Free/Developer указаны отдельно, «pay-as-you-go» с лимитами и кэшем подсказок.) ([Groq](https://groq.com/groqcloud?utm_source=chatgpt.com))
* **xAI (Grok) — если решим брать их для соц-NLP:** **grok-4-fast** от **$0.20/M вход** и **$0.50/M выход**. ([xAI](https://x.ai/news/grok-4-fast?utm_source=chatgpt.com))
* **Apify:** Starter **$39/мес** (+$0.30 за CU), Scale **$199/мес** (+$0.25 за CU). Для нашего профиля хватит Starter/Scale в зависимости от частоты сканирования. ([Apify](https://apify.com/pricing?utm_source=chatgpt.com))
* **Helius (Solana RPC/Webhooks):** Free: 1M кредитов; **Developer $49/мес** — 10M кредитов; вебхуки списывают **1 кредит за событие**. ([Helius](https://www.helius.dev/pricing?utm_source=chatgpt.com))
* **Сервер (EU):** Hetzner Cloud —  
  CX22 (2vCPU/4GB) ≈ **€3.79**, CX32 (4vCPU/8GB) ≈ **€6.80**, CX42 (8vCPU/16GB) ≈ **€16.40**/мес. Рекомендую для прод-бота минимум CX32/CX42. ([hetzner.com](https://www.hetzner.com/pressroom/new-cx-plans/?utm_source=chatgpt.com))

(Альтернативы DigitalOcean стоят дороже при схожих ресурсах, но тоже вариант.) ([DigitalOcean](https://www.digitalocean.com/products/droplets?utm_source=chatgpt.com))

**Итог по сервисам (минимум):** Helius $49 + Apify $39 + сервер €6.80 ≈ **€90–110/мес** без учёта трафика LLM (Groq/xAI — по факту токенов).

# Фрилансер и цена

Твоё **€200** — реально, если задача строго «сшить и задеплоить уже готовое» за **6–8 часов**. Чтобы уложиться:

* Зафиксировать **узкий объём**: вшить конфиг ИИ, подключить Apify+Grok, добавить вкладку Tracked Terms + 4 кнопки «смотреть», Helius-вебхук.
* Принять работу по чек-листам DoD выше (и логам).
* Разбить на **2 кратких этапа по оплате**: (1) дев-стенд и тесты, (2) прод-деплой и мониторинг.

# Как двигаемся сейчас (по твоей схеме)

1. **Этап 1 — ИИ Голиба:** беру конфиг выше как основу V2 (Risk/Pump/weights/ban\_terms).
2. **Этап 2 — Парсер:** накатываем жёсткие фильтры/лимиты и кэш; проверяем CUs.
3. **Этап 3 — UI:** добавляем Tracked Terms + Helius-алерты + «Смотреть».

Готов двигаться именно так, по очереди. Когда скажешь «поехали с Этапа 1», я сразу дам **точный список файлов/папок** для вшивания в архив ИИ (имена, куда положить, что править в .env и где включить флаги) — без лишнего кода и без риска поломок.

Отлично, Брат 🙌. Ты правильно выстроил разделение: **Альфа-1 — базовая и рабочая**, **Альфа-2 — усиленная, с новыми сервисами, другими парсерами и расширенным функционалом**. Вот тебе **10 реально «бомбовых» функций**, которых почти ни у кого нет, и которые сделают твою систему уникальной. Я разделил их по трём твоим блокам: **Киты**, **Пампы мемкоинов**, **Мемные слова/триггеры**.

# 🔱 Топ-10 уникальных функций для Альфа-2

## 🐋 Киты (2–3 функции)

1. **Whale Predictive Clustering**  
   Не просто видеть входы китов, а **группировать их по стилю торговли**:
   * один кита «спринтер» (входит и выходит за минуты),
   * другой «фермер» (заливает ликвидность и ждёт).  
     Можно через **поведенческие паттерны** + кластеризацию транзакций.  
     ➝ Так мы сможем **предсказать**, какой кит готовит новый памп.
2. **Early Wallet Shadowing**  
   Отслеживать **вторичные кошельки китов** (часто они «разминаются» перед пампом: тестовые транзакции, перекидывание SOL по мелочи).  
   Если поймали тест-сети или подготовку ликвидности — сигнал, что кит вот-вот зальёт пул.
3. **Cross-DEX Whale Radar**  
   Отслеживать, когда кит одновременно двигает ликвидность на нескольких DEX (Raydium+Orca+Pumpfun).  
   Это почти всегда означает, что токен готовят к массовому пампу (в т.ч. перед листингом на CEX).

## 🚀 Мемкоины с супер-пампами (2–3 функции)

1. **Liquidity Pulse Detector**  
   Реальное время: резкий залив LP в токен (<60 секунд от старта) — фильтр на **самые перспективные старты**.  
   Секрет: большинство пампов начинается именно с аномально быстрого вливания ликвидности.  
   ➝ В Альфе-2 можно сделать авто-флаг: «⚡ Памповый старт — LP > X SOL за 1 мин».
2. **CEX Whisper Scan**  
   Использовать **обходные API/сканеры слухов CEX** (например, утечки из Gate, MEXC, CoinEx через их листинг-страницы и фиды).  
   Как только появляется токен на тест-листинге — это почти гарант памп.  
   ➝ Можно привязать к твоей кнопке CEX Radar.
3. **Narrative Fusion (AI)**  
   Ловить токены, которые одновременно совпали по 2–3 хайповых темам (например: **frog+AI+Trump**).  
   Такие «гибридные мемы» всегда дают самые большие свечи.  
   ➝ Фишка в том, что большинство парсеров ищут одно слово, а мы будем собирать **сочетания слов**.

## 🐸 Мемные слова/триггеры (2–3 функции)

1. **Emoji Heatmap Scanner**  
   Следить не за словами, а за **эмодзи-паттернами** (🐸, 🚀, 🦍, 🍌, 🔥).  
   ➝ В комьюнити мемов сначала появляются именно эмодзи, а слова подтягиваются позже.  
   Можно собрать словарь «мемных эмодзи» и мониторить их всплески в X/Telegram.
2. **Micro-Community Detector**  
   Ловить упоминания токенов/слов в **маленьких телеграм-группах (200–500 участников)**.  
   Почему? Там часто тестят мемы перед массовым запуском.  
   ➝ Использовать парсер с фокусом на «low-audience groups» + ранний NLP-анализ.
3. **Image-Meme Recognition**  
   Использовать Groq/Grok для распознавания **картинок/мемов в X/TG** (например, жаба с каской, Илона с собакой).  
   ➝ Первые мемы появляются в картинках, а не в тексте. Если бот увидел такой мем 2–3 раза за сутки, то значит, скоро появится токен с этим названием.

## 🧩 Бонус-функция (связка всего)

1. **Pump Predictor Timeline**  
   Система строит **хронологию запуска мемкоина**:

* сначала → мем (эмодзи/картинка/фраза),
* потом → шум в маленьких группах,
* потом → твиты у средних блогеров,
* потом → вход китов / LP залив.  
  Если мы видим **2–3 первых стадии** за <24 часа, бот помечает токен как **«High Probability Pump»**.

# ⚡ Почему это убойно

* **Киты** → не только фиксируем входы, но и предугадываем их стиль и подготовку.
* **Пампы** → берём самые сильные сигналы (LP-влив, утечки CEX, гибридные нарративы).
* **Мем-слова** → ищем неочевидные триггеры (эмодзи, микро-группы, картинки).

Такой набор делает Альфа-2 **мегамашиной**, которая видит хайп **до того, как его заметили остальные**.

Брат, хочешь, я сейчас соберу эти **10 функций в ТЗ-док** (с номерами, описанием, примерами, полями API), чтобы потом прямо передать фрилансеру как часть Альфа-2?

Понял тебя, Брат ✊🔥. То, что я выдал — это действительно «верхушки айсберга». Для **ИИ** таких намётов мало: ей нужны чёткие **правила, триггеры, словари, метрики и тайминги**, чтобы работать без холостых ударов.

Предлагаю так:

* Берём каждый из 10 пунктов.
* Делаем **структуру описания** (единый формат, чтобы потом собрать всё в JSON/конфиг).
* Для каждого: **10+ подпунктов** → что искать, где искать, как фильтровать, какие параметры считать, какие слова/эмодзи/паттерны триггерят, какие цифры считать «ок» и «флаг», какие действия по минутам (от старта токена).
* В конце — **словари/списки** (например, мемные эмодзи, бан-слова, whitelist китов, паттерны LP).
* Добавим **примеры сценариев** (кейсы), и **выводы с цифрами**: что значит «пампово», что значит «слив».

## Шаблон для каждого пункта (чтобы было понятно)

1. **Название функции** (например: Whale Predictive Clustering).
2. **Зачем нужна** (почему именно так, а не иначе).
3. **Источники данных** (какие сервисы: Helius, Solscan, Groq, Apify, Nansen и т.д.).
4. **Триггеры** (что ловим: события, слова, эмодзи, LP-влив, tx-паттерны).
5. **Фильтры** (что отбрасываем: боты, спам, микротранзакции).
6. **Метрики и сетки** (например: LP > 30 SOL за 1 мин = сильный сигнал; Dev% > 5% = флаг).
7. **Тайминги (минуты)** — как разворачивается событие (0–1 мин, 1–5 мин, 5–30 мин, 1 час).
8. **Словари** (эмодзи, бан-слова, whitelist, alias-таблицы).
9. **Примеры (кейсы)** — реальный сценарий, как это работает (например, PEPE: сначала 🐸-мемы, потом LP 50 SOL за 1 мин, потом памп).
10. **Вывод/алгоритм действий ИИ** (что делать: поднять алерт, пометить как High Probability, ждать китов, или отбрасывать).

## Как это пойдёт в работу

* Мы с тобой поочерёдно берём каждый пункт → я его **расписываю на 10+ подпунктов по этой структуре**.
* Ты смотришь, корректируешь, добавляешь свои идеи.
* Потом собираем всё в один документ **«Alpha-2 AI Playbook»** (Word/JSON).
* Фрилансеру это вообще не нужно — он только «сшивает железо».
* А **ИИ Голиба** будет загружать это как «настройки поведения» — и знать каждую запятую.

👉 Предлагаю начать с блока **Киты** (три функции).  
Например, с первой: **Whale Predictive Clustering**. Я её сразу распишу по 10+ подпунктам: сетки, триггеры, примеры.

Хочешь, я начну с неё прямо сейчас и дам тебе первый **развёрнутый пример** — чтобы мы утвердили формат, а потом таким же стилем пройдём по всем остальным?

Отлично, Брат — фокусируемся на китах. Ниже разворачиваю первые три пункта целиком: каждый пункт — это подробный «инструктаж» для ИИ Голиба: почему, какие данные, какие триггеры, фильтры, метрики, минутная логика, примеры, конфиги и конечные действия. Всё предельно конкретно — можно сразу превратить в JSON-конфиги и правила для движка.

Я дам каждую функцию в одном формате:

1. цель / зачем; 2) источники данных; 3) что именно собираем; 4) триггеры/фичи; 5) фильтры от ложных срабатываний; 6) метрики и пороги; 7) временная логика (минуты); 8) алгоритмы/модели; 9) примеры сценариев; 10) итоговые действия ИИ (чёткие команды). В конце каждого блока — пример JSON-конфига и краткий псевдокод.

Погнали.

# 1) Whale Predictive Clustering — «Кластерная предсказалка китов»

### 1.1. Цель / зачем

Не просто фиксировать большие транзакции, а научиться распознавать поведенческие паттерны китов и предсказывать их следующую акцию (влив ликвидности → памп). Это даёт нам преимущество: мы знаем не только «когда» кит влетел, но и **какой тип входа** он планирует и с какой вероятностью это приведёт к пампу.

### 1.2. Источники данных

* On-chain: Helius Webhooks / QuickNode / Solana RPC (tx, token mints, token transfers).
* Analytics: Solscan, SolanaFM, Dexscreener (ликвидность/объёмы).
* Enrichment: Nansen (если подключён), Arkham/Nansen labels, Chainbase (если есть).
* Social: Apify/Grok — упоминания кошелька/адреса и реакция сообщества.
* Internal: история действий известных китов (наш whitelist whales.csv).

### 1.3. Что собираем (фичи)

* Транзакционный вектор кошелька: (time, amount, token, to/from, tx\_type [swap/add\_liq/remove\_liq, transfer, mint], target\_pool).
* Частота операций, средний чек (median tx amount), stddev суммы.
* Скорость последовательных операций (tx/sec), интервалы.
* История распределения: входы на CEX vs DEX, перекидки между кошельками.
* Перекрытие по кошелькам (address graph): какие адреса связаны (common signers, same owner patterns).
* Социальный фон вокруг адреса (упоминания, комментарии, «имя» адреса — label).

### 1.4. Триггеры / признаки подготовки к пампу

(Каждый триггер даёт баллы; суммарный Score используется в решении.)

1. Серия «тестовых» переводов в токен с увеличивающимся размером (3+ tx в течение 10–30 мин).
2. Короткая цепочка перекидываний между «пул-кошельками» (создание нескольких адресов-посредников).
3. Совпадение времени: парная активность нескольких китов на одном токене / пуле.
4. Резкий «прогон» мелких транзакций на лимит с целью проверки слippage.
5. AddLiquidity → small swap tests within 1–5 min → then big add.
6. Transfer to known market-making addresses (Jupiter aggregator / known LP providers).
7. Всплеск наблюдений в локальных микро-группах (парсер) + whale activity = усиленный триггер.
8. Переход token/LP на адреса, связанные с CEX (выводы на кефир-кошельки) — часто после пампа (анти-триггер для BUY).
9. Поведение «разогрева»: перекидывание небольших сумм (0.1–1 SOL) на новые кошельки перед крупным вливом.
10. Определённые «подготовительные» подписи в calldata (если есть) — например, одновременное выставление on-chain параметров.

### 1.5. Фильтры от ложных срабатываний

* Мелкие смесеры / краны (<0.01 SOL) — игнорировать.
* Боты-синтетики: адреса с очень высокой частотой tx (<1s между tx) и малыми суммами; маркируем как «bot».
* Аппокси-кошельки (dust collectors) — исключаем по profile features (созданы в последние N дней и имеют мало outbound pattern).
* CEX cold wallets — почти всегда не пампят напрямую (но используются для вывода) — пометим как «неинтересны для входа».

### 1.6. Метрики и пороги (пример сеток)

* TestTx pattern: 3+ test tx с ростом суммы ≥ 2× за window 30 мин → +2 балла.
* AddLiquidity volume: LP\_added ≥ 10 SOL за 1–5 min → +3 балла.
* WhaleCluster entry: 2+ «whale addresses» (наш whitelist) interact with pool within 10 min → +4 балла.
* Slippage Probe: test swap with slippage observed < threshold → +1 (означает, что пул «торгуемый»).
* Score thresholds:
  + Score ≥ 7 → HIGH PROBABILITY (готовить sniping/alert).
  + 4 ≤ Score < 7 → WATCH (следить в реальном времени).
  + Score < 4 → LOW.

### 1.7. Временная логика (таймлайн, минутный разбор)

* T0 = обнаружен первый test tx / small LP add.
  + 0–1 мин: пометить токен как «heat-1», запустить быструю проверку LP и Dev%.
  + 1–5 мин: если добавлены ещё 2 tx с ростом суммы → повысить Score.
  + 5–15 мин: если появляются мульти-whale входы → Score↑, открыть «Trade window».
  + 15–60 мин: мониторинг наблюдательных метрик (holders growth, volume spike); в случае LP\_injection > X SOL → оповестить и подготовить buy script (при соблюдении anti-rug).
  + 60 мин: если нет активов/снятия LP — снять флаг, отложить в историю.

### 1.8. Алгоритмы / модели

* Построение графа адресов (graph analytics) + кластеризация владельцев (HDBSCAN / DBSCAN по фичам: tx frequency, avg\_amount, out\_degree).
* Sequence model для паттернов (LSTM/Transformer) — обучаем распознавать «последовательности перед пампом».
* Feature importance → RandomForest/LightGBM для объяснимого скоринга (выдаёт contribution каждой фичи к Score).

### 1.9. Пример сценария (кейc)

* Кошелёк A делает: transfer 0.5 SOL → tokenA (test swap), через 2 мин transfer 2 SOL → tokenA (test), через 8 мин add\_liq 30 SOL → через 12 мин кошелёк B (из whitelist) делает add\_liq 50 SOL → Score 3+2+4=9 → HIGH → action: alert, open trade window, set preapproved buy (слегка агрессивный slippage).

### 1.10. Итоговые действия ИИ (чёткие)

* Score ≥ 7: создать alert в UI + TG, поднять «snipe mode» (готовить buy с заранее заданными параметрами: max\_slippage, max\_amount, stop\_conditions).
* 4 ≤ Score < 7: открыть live monitoring (stream charts + show holders delta).
* Score < 4: логировать события в historical DB (для ре-трейнинга модели).

#### JSON пример конфига (Whale Clustering)

{

"whale\_clustering": {

"test\_tx\_window\_min": 30,

"test\_tx\_min\_count": 3,

"lp\_add\_threshold\_sol": 10,

"whale\_list\_file": "data/whales.csv",

"score\_weights": { "test\_tx": 2, "lp\_add": 3, "multi\_whale": 4, "slippage\_probe":1 },

"score\_thresholds": { "high": 7, "watch": 4 }

}

}

#### Псевдокод (detector)

for tx in webhook\_stream:

if is\_test\_tx(tx):

update\_counter(token, "test\_tx", window=30min)

if lp\_added(tx) and amount >= config.lp\_add\_threshold\_sol:

add\_score(token, "lp\_add", config.score\_weights.lp\_add)

if multi\_whale\_interaction(token, window=10min):

add\_score(token, "multi\_whale", config.score\_weights.multi\_whale)

score = compute\_score(token)

if score >= config.score\_thresholds.high:

emit\_alert(token, "HIGH")

# 2) Early Wallet Shadowing — «Теневой шэдоуинг кошельков»

### 2.1. Цель / зачем

Авто-определение «побочных» кошельков китов — тех, которые часто используются как «тестовые» или «фронт-адреса». Shadowing позволяет ловить подготовительные операции раньше, чем основной кит проявит себя публично.

### 2.2. Источники данных

* On-chain (Helius, QuickNode) — полные tx-ХЕШи.
* Chain analytics (Nansen/Arkham/Chainbase) — для меток адресов.
* Local heuristics DB — историческая база наших shadow-связей.

### 2.3. Что собираем

* Парные сигнатуры: однофамильные pattern–owners (common signer, same memo, reuse of seed nodes).
* Временные паттерны: address A создал address B (via create tx) → далее B делает test tx → затем A делает крупный add\_liq.
* Перекидывания с «мастер» → «темп» → pool.
* Метаданные: age of wallet, first tx time, code reuse.

### 2.4. Триггеры / признаки

1. Создание нового адреса, получившего >0.5 SOL в первые 5 минут от известного whale.
2. Повторная активация «новых адресов» связана с одним мастером.
3. Переключение активности с «публичного» whale на свежесозданный адрес — часто тестовые buy.
4. Сценарий: master → split funds to N wallets (micro-probes) → micro-probes do swaps → master executes large add\_liq.

### 2.5. Фильтры ложных срабатываний

* Faucet funds / staking payouts (регулярные переводы по шаблону) — исключаем.
* Mass wallet creators (some services create lots of wallets) — если адреса создаются автоматом каждым second, помечаем как сервис и не шэдоуим.
* Fresh wallets с нулевой историей — требуют threshold before shadowing (например, >0.2 SOL moved).

### 2.6. Метрики и пороги

* Shadow confidence (0–1) = weighted sum: (funds transferred from master, temporal proximity, repeated patterns).
* threshold\_shadow = 0.6 → начать «преследование» (monitoring + social scraping).
* Probe min amount = 0.2 SOL (for test swap), Master transfer min = 1 SOL.

### 2.7. Временная логика

* T0 = master transfers to new wallet.
  + 0–2 мин: mark new wallet as «candidate shadow».
  + 2–10 мин: если candidate does probe swap → increase confidence.
  + 10–30 мин: if multiple candidates show probes → raise token Score and alert.
  + 30–120 мин: track all downstream activity; if master adds LP → fire HIGH alert.

### 2.8. Алгоритмы / методы

* Graph traversal + propagation score: from master nodes propagate credit to downstream wallets.
* Label propagation (community detection) — HITS/PageRank on transfer graph to find high-influence addresses.
* Temporal sequence modeling for detection of split→probe→add patterns.

### 2.9. Примеры

* Master M transfers 5 SOL split to 5 new addresses (A1..A5) within 3 minutes; A2 does probe swap 0.3 SOL at 4 min; M does add\_liq 40 SOL at 12 min → Shadow confidence of A2 ≥0.8 → action: prealert + preauthorized buy.

### 2.10. Действия ИИ

* Confidence ≥ 0.6: open «shadow mode» для token: start deep monitoring of candidate wallets (live tx feed), raise alert in UI with wallet list.
* Confidence ≥ 0.85 and subsequent LP add: emit HIGH alert and prepare buy script (with anti-rug checks).

#### JSON-пример конфига (Shadowing)

{

"shadowing": {

"master\_transfer\_min": 1.0,

"probe\_min": 0.2,

"shadow\_confidence\_threshold": 0.6,

"high\_confidence\_threshold": 0.85,

"propagation\_depth": 3

}

}

#### Псевдокод

if tx.from in whales\_list and tx.amount >= config.master\_transfer\_min:

mark\_candidate(tx.to)

start\_timer(candidate, 10min)

if candidate.performs\_probe(amount >= config.probe\_min):

candidate.confidence += compute\_confidence(...)

if candidate.confidence >= config.shadow\_confidence\_threshold:

open\_shadow\_mode(token, candidate)

# 3) Cross-DEX Whale Radar — «Кросс-DEX радар китов»

### 3.1. Цель / зачем

Киты перед пампом часто совершают согласованные операции на нескольких DEX (для ликвидности и распределения риска). Отслеживание одновременно в нескольких DEX даёт более высокий уровень уверенности — если один DEX показывает активность, это может быть тест/шум; если два/три — это подготовка.

### 3.2. Источники данных

* DEX feeds: Raydium, Orca, Jupiter swaps, Pump.fun, Serum orderbooks (если есть).
* Aggregators: Dexscreener, SolanaFM volume feeds.
* Webhooks Helius (poolCreated, addLiquidity, swap events).
* Off-chain: pump.fun, Birdeye, CEX listing pages (для кросспроверки).

### 3.3. Что собираем

* LP additions on DEX A, swap probes on DEX B, orderbook changes on Serum/CEX.
* Time alignment: timestamps of each DEX event (use NTP sync).
* Cross-DEX price discrepancies and arbitrage signals (indicates coordinated activity).

### 3.4. Триггеры

1. LP add on DEX1 + probe swap on DEX2 within window 5–10 min.
2. Simultaneous (within 60s) large buys on 2+ DEX → strong signal.
3. Creation of liquidity on DEX1, and at same time mint + pool on DEX2 → coordinated.
4. Price movement on DEX1 followed by DEX2 repeating the move (momentum propagation).

### 3.5. Фильтры ложных срабатываний

* Cross-posting by bots: identical small tx across DEX — ignore if amounts << threshold.
* Arbitrage bots — if pattern matches arbitrage (sells immediately after buy across DEX), mark as arb (may create temporary volume but not pump).
* Low liquidity pools on both DEX — false positive; require min liquidity threshold.

### 3.6. Метрики и пороги

* CrossEventsCount within window W (e.g., 2 events in 10 min) → +points.
* Combined LP volume across DEX ≥ X SOL (e.g., 20 SOL) → strong.
* Synchronous buy ratio: number of buys in same direction across DEX in 60s window / total txs > 0.6 → signal.
* Price spread contraction: spread decreases < Y% after LP add → pump readiness.

### 3.7. Временная логика

* T0 = LP add on DEX A.
  + 0–1 min: check DEX B/C for probe swaps.
  + 1–5 min: if >1 DEX has buy pressure → set CrossDEX Score.
  + 5–15 min: monitor for momentum propagation; if momentum observed → open high alert.
  + 15 min: if pump occurs, log and save pattern; если нет — downgrade.

### 3.8. Алгоритмы / модели

* Temporal correlation engine: align event streams from DEXs by timestamps and compute cross-correlation.
* Bayesian update: prior probability of pump updated with each DEX event.
* Rule engine: if (LP across DEXs + sync buys + low slippage) → HIGH.

### 3.9. Пример сценария

* Raydium: add\_liq 15 SOL at T0.
* Orca: 3 buys totalling 10 SOL at T0+30s.
* Pump.fun: pool created at T0+2min.
* CrossDEX Score: high → alert.

### 3.10. Действия ИИ

* CrossDEX Score ≥ threshold: publish alert with evidence: list DEX events (tx hash, amount, time), prepare snipe parameters.
* Если CrossDEX score высокий и Whale Predictive score высокий и Shadow confidence высокий → immediate HIGH alert + auto-preparation (approve tx, set slippage, set stop-loss rules).

#### JSON конфиг (Cross-DEX)

{

"cross\_dex": {

"window\_min": 10,

"min\_dex\_events": 2,

"min\_combined\_lp": 20,

"sync\_buy\_ratio\_threshold": 0.6,

"score\_weights": { "lp": 0.4, "sync\_buys": 0.3, "price\_momentum":0.3 }

}

}

#### Псевдокод

events = collect\_events(token, window=10min)

dex\_events = group\_by\_dex(events)

if count\_unique\_dex(dex\_events) >= config.min\_dex\_events and sum\_lp(events)>=config.min\_combined\_lp:

score = compute\_cross\_dex\_score(events)

if score >= threshold:

alert\_high(token, events)

# Общие замечания по реализации (инфраструктурные требования)

1. **Stream processing** — все on-chain события должны приходить через очередь (BullMQ / Kafka). Реакция должна быть near-real time (<1–2s latency on processing).
2. **Time sync** — timestamps от разных источников необходимо нормализовать (NTP).
3. **Feature store** — хранить фичи в Redis/ClickHouse для быстрых агрегатов.
4. **Explainability** — для каждого Score хранить contribution per feature (для дебага и human review).
5. **Retraining** — каждые N дней ре-тренировать sequence model на новых pattern’ах (хранить истории false/true positives).
6. **Safety gates** — если dev% > threshold OR LP unlocked → блокируй auto-buy (только alert).

# Словари / короткий список полезных артефактов (для загрузки в ИИ)

* whales.csv — известные whale addresses + labels + trustscore.
* ban\_terms.json — ["airdrop","giveaway","claim","free"]
* emoji\_dict.json — { "frog": ["🐸","frog emoji variations"], "rocket":["🚀","^rocket^"] }
* micro\_group\_list.txt — list of TG groups to prioritise parse (low audience groups)
* dex\_list.json — ["raydium","orca","pump.fun","serum","jupiter"]

# Вывод — чего добиться за первый месяц

1. Разработать Whale Predictive Clustering + Shadowing + CrossDEX Radar и включить их в конвейер.
2. Довести latency end-to-end до <5s для базовых тест-событий.
3. Достичь precision первичных high alerts ≥ 60% (в течение первого месяца, затем улучшать).
4. Протестировать систему на 50 исторических пампах (backtest) и собрать метрики false/true positives.

Если хочешь, Брат — я прямо сейчас:

* 1. конвертирую всё это в набор JSON-конфигов (whale\_clustering.json, shadowing.json, cross\_dex.json),
  2. подготовлю шаблон SQL для хранения фичей,
  3. напишу детализацию «чек-лист тестирования» (как проверить каждую функцию).

Как дальше? Хочешь, чтобы я сгенерировал эти три JSON-конфига и чек-лист прямо сейчас?