Начну сразу с сути: **takeyourtoken.app (TYT)** — это не просто «копия GoMining». Это трёхслойная система:

1. **Платформа NFT-майнеров и BTC-наградами** (аналог GoMining, но с совиным/рыцарским брендом и своей экономикой).
2. **Токен TYT** (мем-/utility-токен: скидки, burn-циклы, ve-локи, governance, геймификация).
3. **Образовательный и благотворительный слой**:
    - *TYT Digital-Interactive-Technology-Blockchain Crypto Academia* (академия внутри экосистемы).
    - *TYT Brain Cancer Children's Research & Support Foundation* — фонд поддержки и исследований опухолей мозга у детей, встроенный в токеномику и UI.

Ниже — разбор «с самого начала»: что это за проект, как он работает для пользователя, какие есть модули, как всё реализовать технически.

---

1. Идея и позиционирование TYT

1.1. Что мы копируем у GoMining

По сути, GoMining делает следующее:

- Продаёт **NFT-майнеры** (цифровые майнеры с параметрами TH/s и W/TH).
- Начисляет **ежедневные BTC-награды** в кастодиальный кошелёк.
- Взимает **плату за обслуживание (maintenance)**: электричество + сервис.
- Даёт **скидки**, если платить их токеном платформы.
- Имеет **маркетплейс**, VIP-статусы, геймификацию, реферальную программу и пр.

TYT берёт **всю модель как категории**, но:

- Меняет цепочки и архитектуру (Polygon + Solana, опционально TRON).
- Меняет **числа** и форму кривых (скидки, burn-каденс).
- Делает **совершенно другой бренд и UI** (шлем рыцаря/филин/щит/меч, Owlverse, ранги и совиная мифология).

1.2. Три столпа TYT

1. **TYT NFT Mining Protocol**
    - NFT-майнеры → ежедневные BTC-награды → кастодиальный кошелёк.
    - Оплата обслуживания в BTC/стейблах/токене TYT.

    - NFT-маркетплейс и апгрейды мощности/эффективности.

2. **TYT Token & Owlverse**
    - TYT как utility: скидки, апгрейды, комиссии, governance.
    - veTYT (локи) → голос, повышенный дисконт, участие в распределениях.
    - Ранги «рабочая сова / академик / дипломат / миротворец / воин» как VIP-уровни.
3. **TYT Academia + Brain Cancer Fund**
    - Академия: уроки про крипту, майнинг, безопасность, налоги, DeFi.
    - За выполнение — бейджи, сертификаты-NFT, очки ранга.
    - **Детский фонд опухолей мозга**:
        - фиксированный % от maintenance/комиссий;
        - отдельные “Charity-Miners”, доход которых частично направляется в фонд;
        - видимый Charity Dashboard с on-chain / off-chain отчётами.

---

2. Как TYT выглядит для пользователя

2.1. Предтеча (landing takeyourtoken.app)

До полноценного кабинета у тебя есть **лендинг-страница** на домене:

- Hero-блок: бренд TYT (логотип, сова/шлем/щит/меч), слоган, CTA «Launch App».
- Объяснение: *«NFT-майнеры → ежедневные BTC → скидки за TYT → прозрачные burn-циклы»*.
- Простой **калькулятор доходности**: вводишь TH/s, W/TH, цену BTC, цену kWh → получаешь дневной/месячный/годовой чистый доход с учётом обслуживания.
- Блок **Tokenomics** (TYT, burn, veTYT).
- Блок **How it works (4 шага)**:
    1. Зарегистрируйся, пройди KYC, создай кошелёк.
    2. Купи NFT-майнер.
    3. Получай BTC ежедневно (минус обслуживание).
    4. Торгуй майнером, голосуй параметрами протокола и поддерживай детский фонд.
- Ссылки на соцсети, Telegram-канал, будущий dApp.

Это уже готовый, статический фронтенд (HTML+CSS+JS) под Hostinger, без серверной части.

2.2. Личный кабинет (Web + Mobile)

Основные разделы (feature-паритет с GoMining):

1. **Dashboard**
   - Балансы: BTC, TYT, стейблы.
   - Дневные начисления BTC.
   - Незаплаченные счета за обслуживание.
   - Твой VIP-уровень/ранг совы.
   - (Позже) виджет live-стримов дата-центров.
2. **My Miners (Моя ферма)**
   - Список всех NFT-майнеров: TH/s, W/TH, регион/ферма, статус (Active/Delinquent).
   - Кнопки **Upgrade Power** и **Upgrade Efficiency**.
   - Привязка майнера к Charity-режиму (часть дохода → фонд).
3. **Rewards**
   - История ежедневных BTC-начислений.
   - Графики, суммарные показатели.
   - Кнопка Withdraw (BTC / L2 / wBTC и др.).
   - Опция Auto-Reinvest: часть награды автоматически конвертируется в увеличение TH.
4. **Maintenance & Discounts**
   - Показывается формула:
dailyCost = kWhPrice * (efficiency * TH) + serviceFee.
   - Выбор валюты оплаты: BTC / стейбл / TYT.
   - При оплате в TYT — динамическая скидка до ~18–20% по собственной Discount Curve.
   - Кнопка Service (extra discount) и влияние VIP/veTYT.
5. **Marketplace**
   - P2P-торговля NFT-майнерами: листинги, покупки, отмена ордеров.
   - Комиссии платформы, часть из которых:
     - сжигается в TYT;
     - переводится в Charity-кошелёк фонда.
6. **Wallet**
   - Кастодиальный кошелёк внутри TYT.

    - Депозиты/выводы BTC, стейблов, TYT, мульти-чейн маршруты.

7. **Governance / veTYT**
    - Локация TYT (1 неделя – 4 года) → veTYT.
    - Голосование по: kWh тарифам, discount curve, burn-каденсу, доле поступлений в фонд и т.д.
8. **VIP & Owlverse**
    - Ранги Worker / Academic / Diplomat / Peacekeeper / Warrior.
    - Привязка к: суммарному TH, объёму veTYT, истории оплат maintenance, прогрессу в Академии.
9. **Referrals & Ambassadors**
    - Базовая реф-программа (5% и т.п., с возможным расширением до 5-5-5).
10. **Academy**

- Курсы, задания, тесты, сертификаты.
- Байджи и NFT-сертификаты → встраиваются в ранги и скидки.

11. **Charity / Foundation**

- Отдельная вкладка:
    - какие суммы ушли в фонд;
    - отчёты об использовании средств (off-chain документы + on-chain tx).
- Переключатели у майнеров: «% дохода → фонд» и «одноразовое пожертвование TYT/BTC».

---

3. Ончейн-слой: сети, контракты, сущности

3.1. Сети

По текущему черновику:

- **Polygon PoS (EVM)** — основной слой для NFT-майнеров и части логики veTYT.
- **Solana** — сеть, где уже существует мем-токен TYT (pump.fun), используется как utility токен и источник ликвидности.
- Дополнительно рассматривается **TRON (TRC-20/721)** как альтернатива/расширение, но в v2 это может быть отдельной фазой.

BTC остаётся «базовой наградой», хранящейся кастодиально off-chain, но с возможностью вывода на разные сети (BTC L1, Lightning, wBTC/ETH, и т.д.).

3.2. Основные контракты

1. **MinerNFT (Polygon, ERC-721)**
    - Параметры:
        - power_th (TH/s, масштабированный uint256);
        - efficiency_w_per_th (W/TH);
        - maintenanceRate (базовая ставка);
        - farmId (идентификатор дата-центра/региона).
    - Функции:
        - mintMiner(to, params) — минт нового майнера (админ/launchpad).
        - upgradeHashrate(tokenId, newPower) — апгрейд мощности.
        - upgradeEfficiency(tokenId, newEffTier) — улучшение эффективности (за TYT).
        - setStatus(tokenId, status) — Active/Locked/Delinquent.
    - События:
        - RewardAccrued (для индексатора);
        - MaintenancePaid (для прозрачности и foundation-логики).
2. **TYT Token (Solana, SPL)**
    - Уже создан на pump.fun; в архитектуре используется как:
        - средство оплаты обслуживания/апгрейдов;
        - источник скидок;
        - токен для локов (через мост и veTYT).
3. **veTYT (Polygon, EVM)**
    - Модель: **локи на время → veTYT → голос + буст скидок/награды**.
    - Реализуется либо:
        - как отдельные «лок-позиции» в виде NFT;
        - либо как простая запись в контракте с параметрами amount, unlockTime.
4. **RewardsTreasury**
    - Содержит BTC (или wrapped-аналог).

- Ежедневно:
    - получает данные от Rewards Engine;
    - фиксирует net-начисления;
    - публикует Merkle root по всем NFT и дату.

5. **BurnScheduler (TYT Burn & Mint)**
    - Собирает все TYT, уплаченные за обслуживание/апгрейды/комиссии.
    - Раз в N (например, каждые 2 недели, вторник 00:00 UTC):
        - вызывает burnCollected();
        - пишет событие TokensBurned;
        - опционально выпускает новую порцию токенов < X% от burnt (в Rewards/Promo пул).
6. **Marketplace**
    - Функции:
        - listMiner(tokenId, price, asset);
        - buyNow(orderId);
        - cancel(orderId).
    - Поддержка роялти (ERC-2981-like) и блокировки делинквентных майнеров.

---

4. Off-chain слой: движки, сервисы, БД

4.1. Бэкенд-ядро

Рекомендуемый стек: NestJS (Node.js) + Postgres + Redis + очередь (Kafka/Cloud Tasks).

Основные сервисы:

- **auth-service**: пользователи, сессии, 2FA, KYC-статусы.
- **wallet-service**:
    - кастодиальные балансы;
    - депозиты/выводы;
    - интеграция с MPC/HSM (Fireblocks-подобная модель).
- **rewards-engine**:
    - каждый день считает gross BTC-доход по пулу;
    - вычисляет per-NFT net после maintenance и скидок;

    - записывает результаты в rewards и кошельки;
    - строит Merkle-дерево и публикует root в контракт.
- **maintenance-engine**:
    - хранит таблицу тарифов kWh и сервис-комиссии по farmId;
    - считает сумму инвойсов;
    - применяет Discount Curve, Service-бонус, VIP/veTYT.
- **marketplace-service**:
    - ордербук, фильтры;
    - связи с контрактом Marketplace;
    - учёт комиссий и charity-долей.
- **reporting-service**:
    - отчёты по burn;
    - отчёты для детского фонда (специфический Charity Report API);
    - выгрузка CSV/экспорт для пользователей.

4.2. База данных (core таблицы)

Из blueprint:

- users(id, email, kyc_status, rank_score, ve_tyt_power, created_at)
- miners(id, owner_id, power_th, eff_tier, region_id, status, reinvest_pct, created_at)
- fees(region_id, kwh_usd, service_bps, updated_at)
- rewards(date, miner_id, gross_btc, elec_usd, service_usd, discount_pct, net_btc, proof_leaf)
- wallet_accounts(id, user_id, asset)
- ledger_entries(entry_id, account_id, debit, credit, ref_type, ref_id, ts)
- burn_events(id, amount_tyt, window_id, tx_hash, report_uri)
- **дополнительно под фонд:**
    - charity_flows(id, source_type, source_id, asset, amount, tx_hash, created_at)

---

5. TYT Tokenomics & Foundation

5.1. Роли токена TYT

Согласно blueprint:

- Оплата maintenance (со скидкой).
- Оплата апгрейдов эффективности.
- Комиссии на маркетплейсе.
- Governance голос.
- Геймификация (ранги, академия, бонусы).

5.2. Сжигание и эмиссия

- **Все TYT**, уплаченные за обслуживание и апгрейды, **собираются и сжигаются** по расписанию.
- Для стимулов (reinvest-бонусы, промо) можно **mint ≤ 35% от количества сожжённых** в отдельный пул.

5.3. Discount Curve

Задаётся **своими числами**, не совпадающими с GoMining, например:

- Bronze — 2% при 30 днях покрытия;
- Silver — 5% при 90 днях;
- Gold — 9% при 180;
- Platinum — 13% при 270;
- Diamond — 18% при 360;
- Cap = 18%.

5.4. Интеграция фонда в токеномику

Добавляем явный слой:

- % от:
    - maintenance-fee (service часть);
    - marketplace комиссий;
    - части burn-window (например, 1–2% от ETH/USDT эквивалента сожжённых TYT);
    - добровольные donations пользователей (в BTC/TYT/стейблах).
- На уровне governance:
    - veTYT голосует за **долю протокольного дохода в фонд** (например, с 5% до 15%).
    - Параметр регулируем через стандартный governance-процесс.

---

6. Право, комплаенс и сторовые ограничения

- NFT = **доступ к сервису** (право на BTC-индексированные награды нетто от обслуживания), а не доля в компании/оборудовании.
- Нет фиксированных APR/ROI-обещаний, только калькуляторы и исторические графики с дисклеймерами.
- Полный KYC/AML для кастодиальных кошельков и фиат-онрамп.
- Ни на iOS, ни на Android **нельзя майнить на устройстве** — только облачный сценарий, что чётко соблюдено.

---

7. TYT Academia (цифровая академия)

Из blueprint это отдельный, но интегрированный модуль:

- Треки:
  - Основы кошельков, ключей, комиссий.
  - Экономика BTC-майнинга.
  - NFT-майнеры и reinvest.
  - Мультичейн и безопасность.
  - Налоги и комплаенс (на базовом уровне).
- Формат lesson unit:
  - 5–7 минут чтения;
  - 2–3 интерактивных виджета;
  - небольшой квиз.
- Тех реализация:
  - Next.js, MDX контент, SBT-сертификаты (Soulbound NFT) в Polygon.
  - Метрики прохождения и защита от «фарма» тестов.

Связь с остальной системой:

- Завершённые курсы → очки в Rank Score → повышенные перки и, опционально, микроскидки.

---

8. Roadmap TYT (как это раскатывать)

По blueprint:

1. **Phase 0 — Sandbox**
    - Контракты MinerNFT + базовый Market на Polygon testnet.
    - Rewards-симулятор (без реального BTC).
    - Простой Web-кабинет (Create Miner → Rewards → Wallet).
    - Базовый лендинг (то, что уже обозначено как «предтеча»).
2. **Phase 1 — MVP**
    - Реальная кастодия BTC.
    - Ежедневный Rewards Engine + публикация Merkle-root.
    - Интеграция с TYT на Solana (скидки, burn).
    - Выводы в BTC/wBTC, базовый маркетплейс, Auto-Reinvest beta.
    - KYC/Legal/ToS + прозрачные отчёты.
3. **Phase 2 — Multi-chain & Academy**
    - Добавление маршрутов выплат на SOL/TRX/TON/XRP, Lightning.
    - Запуск Академии v1, реферальной программы, совиных рангов.
    - Старт Charity Foundation интерфейса и первых перечислений.
4. **Phase 3 — Full Owlverse**
    - Игровые режимы, аватары, расширенная VIP-система.
    - Углублённая токеномика veTYT, расширенные governance-сценарии.
    - Расширение фонда, партнерства с клиниками/NGO.

---

9. Кратко: takeyourtoken.app — что это за проект

Если сжать всё в одну фразу:

> **TYT — это Web3-платформа, где ты покупаешь NFT-майнеры, каждый день получаешь BTC, платишь за обслуживание токеном TYT со скидкой, голосуешь параметрами протокола, учишься в собственной крипто-академии и автоматически поддерживаешь фонд помощи детям с опухолями мозга.**
>
> Вся текущая документация уже разложена так, чтобы её можно было прямо «скармливать» AI-агентам (bolt.new / v0 / Lovable / Replit) для автогенерации:

- контрактов (MinerNFT, veTYT, Marketplace, BurnScheduler),
- бекенда (NestJS/FastAPI + Postgres/Redis),
- фронта (Next.js + мобильные apps),
- и сервисов для академии/фонда.

Окей, собираем TYT v2 «как схему для сборки» 🦉⚔️

---

1. Архитектура TYT v2: список микросервисов

Думаем так: у тебя есть один общий **API-Gateway / BFF**, а за ним — пачка микросервисов, каждый отвечает за свой кусок логики. Всё общается по gRPC/REST + события (Kafka/RabbitMQ).

1.1. Core-сервисы

1. **auth-service**
   Задача: аутентификация, авторизация, сессии.

- Фичи:
  - Email/телефон + пароль, OAuth (Google/Apple), 2FA.
  - JWT токены для фронта.
  - Роли: user, support, admin.
- Таблицы:
  - users(id, email, password_hash, status, created_at)
  - user_sessions(id, user_id, ua, ip, expired_at)

2. **kyc-service**
   Задача: интеграция с KYC-провайдером (Sumsub/Onfido и т.п.).

- Фичи:
  - Старт KYC-процесса.
  - Webhook от провайдера → статус (approved/rejected/pending).
- Таблицы:
  - kyc_cases(id, user_id, provider_case_id, status, updated_at)

3. **user-profile-service**
   Задача: профиль пользователя и настройки.

- Фичи:
  - Ник, язык, валюта по умолчанию, уведомления.
  - Связь с Telegram / e-mail / пуши.
- Таблицы:
  - user_profiles(user_id, nickname, lang, tz, notif_prefs_json)

4. **wallet-service** (кастодиальные балансы)
   Задача: внутренние счета для BTC, стейблов, TYT.

- Фичи:
    - Создание аккаунтов для assets: BTC, USDT, USDC, TYT, и т.д.
    - Депозиты/выводы через blockchain-gateway.
    - Внутренний ledger (двойная запись).
- Таблицы:
    - wallet_accounts(id, user_id, asset, external_address, created_at)
    - ledger_entries(id, account_id, debit, credit, ref_type, ref_id, ts)

5. **blockchain-gateway**
   Задача: разговор с блокчейнами (BTC, Polygon, Solana, TRON…).

- Фичи:
    - Мониторинг входящих tx (депозиты).
    - Отправка исходящих tx (выводы, burn).
    - Мост TYT: учёт TYT на Solana и отображение во фронте.
- Это может быть один сервис с адаптерами по сетям.

6. **miner-registry-service**
   Задача: зеркало ончейн MinerNFT + off-chain метаданные.

- Фичи:
    - Синхронизация с контрактом MinerNFT (mint, transfer, upgrade).
    - Таблица майнеров с параметрами: TH/s, W/TH, регион, статус.
- Таблицы:
    - miners(id, nft_token_id, owner_id, power_th, eff_tier, region_id, status, reinvest_pct, created_at)
    - regions(id, name, country, tz, note)

7. **maintenance-engine-service**
   Задача: расчёт стоимости обслуживания.

- Фичи:
    - kWh тариф по региону.
    - serviceFee в bps.
    - Формула: elec_cost + service_fee с учётом Discount Curve, TYT, VIP.
- Таблицы:
    - fees(region_id, kwh_usd, service_bps, updated_at)

    - maintenance_invoices(id, miner_id, period_start, period_end, amount_usd, discount_pct, asset, status)

8. **rewards-engine-service**
   Задача: ежедневные BTC-начисления.

- Фичи:
    - Получает общий BTC-пул за сутки.
    - Делит по майнерам по доле TH, вычитает maintenance.
    - Начисляет net BTC в кастодиальные кошельки.
    - Выдаёт Merkle root по распределению для ончейн-валидации.
- Таблицы:
    - daily_pool(date, gross_btc, price_btc_usd)
    - rewards(date, miner_id, gross_btc, elec_usd, service_usd, discount_pct, net_btc, proof_leaf)

9. **marketplace-service**
   Задача: P2P-торговля NFT-майнерами.

- Фичи:
    - Листинг майнера по цене и валюте (USDT/TYT/другое).
    - Покупка, отмена, комиссия в пользу протокола.
    - Передача части комиссии в burn-пул и в фонд.
- Таблицы:
    - orders(id, miner_id, seller_id, price, asset, status, created_at)
    - trade_events(id, order_id, buyer_id, amount, fee_protocol, fee_charity, ts)

10. **governance-service**
    Задача: veTYT + голосования.

- Фичи:
    - Учёт локированных TYT (через ве-NFT / лок-позиции).
    - Создание proposals: изменение скидок, доли фонда, и т.д.
    - Подсчёт голосов → push параметров в другие сервисы.
- Таблицы:
    - locks(id, user_id, amount_tyt, locked_until, voting_power)
    - proposals(id, title, description, param_key, status, created_at)
    - votes(id, proposal_id, user_id, voting_power, choice)

11. **rank-and-gamification-service (Owlverse)**
    Задача: ранги сов, уровни, бейджи.

- Фичи:
    - RankScore = функция от: суммарного TH, veTYT, истории платежей, активности в Академии.
    - Ранги: Worker / Academic / Diplomat / Peacekeeper / Warrior.
- Таблицы:
    - user_rank_state(user_id, rank, rank_score, updated_at)
    - user_badges(id, user_id, badge_code, earned_at, source)

12. **academy-service**
    Задача: TYT-академия (курсы, квизы, сертификаты).

- Фичи:
    - Курсы из markdown/MDX.
    - Трекинг прогресса, квизы.
    - Выдача SBT/NFT сертификатов (через blockchain-gateway).
- Таблицы:
    - courses(id, slug, title, level, meta_json)
    - lessons(id, course_id, order, slug, title)
    - user_course_progress(user_id, course_id, status, score)
    - user_quiz_attempts(user_id, lesson_id, result, ts)

13. **charity-service (Foundation Core)**
    Задача: всё, что связано с **фондом изучения и борьбы с раком мозга у детей**.

- Подробнее см. блок 4 ниже (там прям отдельная схема).
- Таблицы:
    - charity_flows(...)
    - charity_reports(...)
    - charity_campaigns(...)

14. **notification-service**
    Задача: e-mail, Telegram, push.

- Фичи:
    - Шаблоны писем.
    - События: начисление наград, истечение оплаты, новый отчёт фонда.

- Таблицы:
    - notifications(id, user_id, channel, template, payload_json, status, ts)

15. **admin-panel-service**
    Задача: внутренний «BackOffice» для команды.

- Просмотр пользователей, майнеров, инвойсов, жалоб.
- Мануальные корректировки, запуск отчётов, управление content/academy.

---

2. Набор задач для AI-агентов

2.1. Контракты (Solidity / EVM + Solana)

2.1.1. MinerNFT (Polygon, ERC-721)
Задача для агента (пример prompt):

Напиши Solidity-контракт MinerNFT (Solidity 0.8.x, OpenZeppelin ERC721), который:

- хранит для каждого tokenId:
  - powerTH (uint96, TH/s x 1e6),
  - efficiencyWPerTH (uint96, W/TH x 1e6),
  - regionId (uint16),
  - maintenanceRateBps (uint16),
  - status (uint8: 0=Active,1=Delinquent,2=Locked).
- имеет функции:
  - mintMiner(address to, MinerParams calldata params) только для MINTER_ROLE,
  - upgradeHashrate(uint256 tokenId, uint96 newPowerTH) только для UPGRADER_ROLE,
  - upgradeEfficiency(uint256 tokenId, uint96 newEff) только для UPGRADER_ROLE,
  - setStatus(uint256 tokenId, uint8 newStatus) только для OPERATOR_ROLE.
- эмитит события MinerMinted, MinerUpgraded, MinerStatusChanged.
- поддерживает EIP-2981 (роялти для маркетплейса).

2.1.2. RewardsMerkleRegistry
Напиши контракт RewardsMerkleRegistry, который:

- хранит mapping(uint256 day => bytes32 merkleRoot),
- позволяет REWARDS_ORACLE раз в сутки устанавливать root,
- не даёт переписывать root за прошедший день,

- эмитит RewardsRootSet(day, merkleRoot).

2.1.3. veTYT (EVM-слой для governance)

Напиши контракт veTYT:

- принимает мостированные токены TYT (ERC20),
- создаёт «лок-позиции»:
  - struct Lock { uint256 amount; uint256 start; uint256 end; }
- рассчитывает votingPower = amount * f(duration) (линейно: max при 4 года),
- поддерживает:
  - createLock(amount, lockDuration),
  - increaseAmount(lockId, addedAmount),
  - increaseDuration(lockId, addedDuration),
  - withdraw(lockId) после end.

2.1.4. Marketplace (Polygon, ERC-721 торговля)

Напиши контракт MinerMarketplace:

- поддержка листинга MinerNFT,
- ордер: struct Order { address seller; uint256 tokenId; uint256 price; address asset; bool active; },
- команды:
  - list(tokenId, price, asset),
  - buy(orderId) с переводом токена и средств,
  - cancel(orderId) от продавца,
- комиссия:
  - % в протоколный кошелёк,
  - % в charity-кошелёк (адрес задаётся в константах или storage),
- события OrderCreated/Executed/Cancelled.

2.1.5. CharityVault / CharitySplit

Напиши контракт CharityVault:

- принимает ETH/USDC/TYT от протокола и пользователей,
- ведёт totalReceived[asset],
- позволяет только FOUNDATION_MULTISIG делать withdraw с указанием назначения (string memo),

- эмитит DonationReceived(asset, amount, sourceType, sourceId) и CharityWithdrawal(asset, amount, to, memo).

2.1.6. Solana: TYT-SBT для Академии
Для агента Anchor/Rust:

Создай Anchor-программу tyt_academy_sbt, которая:

- минтит непередаваемые NFT (SBT) при завершении курса,
- хранит метаданные курса course_id, level, issue_timestamp,
- запрещает передачу токенов (override transfer hooks / использовать token-2022 с transfer-hook).

2.2. Backend-задачи для агентов (NestJS / FastAPI)

Пример формулировок:

2.2.1. rewards-engine-service
Создай сервис rewards-engine (NestJS + Postgres), который:

- читает из miners (через miner-registry-service или отдельную БД),
- раз в сутки:
  - принимает gross_btc (через admin API или из конфиг-сервиса),
  - считает по каждому майнеру gross_share, maintenance_cost (через gRPC-вызов maintenance-engine),
  - считает net_btc,
  - создаёт записи в таблице rewards и ledger_entries,
  - строит Merkle-дерево по массиву [minerId, net_btc, date],
  - отправляет root в RewardsMerkleRegistry через blockchain-gateway.
- предоставляет REST:
  - GET /rewards?user_id=... — историю,
  - GET /rewards/summary — агрегаты.

2.2.2. maintenance-engine-service
Создай сервис maintenance-engine:

- эндпоинт POST /calculate с input:
  - power_th, efficiency_w_per_th, region_id, date_range, user_discounts (vip_level, has_tyt_discount, prepay_days),
- возвращает:

    - elec_usd, service_usd, total_usd, discount_pct, applied_curve_tier.

И так далее для:

- auth-service
- wallet-service
- marketplace-service
- governance-service
- academy-service
- rank-and-gamification-service
- charity-service

(Каждому агенту — чёткий список эндпоинтов, схемы БД, интеграции.)

——

2.3. Frontend-задачи для агентов (Next.js + мобильные)

2.3.1. Web dApp (Next.js)

Создай фронтенд tyt-web на Next.js 14 (App Router + TypeScript + Tailwind):

- страницы:
  - / — лендинг:
    - hero-секция с логотипом TYT (сова/шлем/щит/меч),
    - CTA «Launch App» → /app,
    - блок «How it works» (4 шага),
    - калькулятор доходности (form + live расчёт),
    - блок «TYT Token & Burning» (график с burned vs minted),
    - блок о фонде (ссылка на /foundation).
  - /app/dashboard — баланс, награды, VIP-ранг.
  - /app/miners — список майнеров, детали с графиками, кнопки Upgrade.
  - /app/marketplace — фильтр/поиск майнеров.
  - /app/academy — курсы/прогресс.
  - /foundation — отчётность фонда.
- интеграция:

- Auth (JWT), API-Gateway,
- onchain-действия через web3 провайдеры (WalletConnect/Phantom).

2.3.2. Mobile App (React Native / Flutter)
Собери мобильное приложение TYT:

- Страницы: Dashboard, My Miners, Rewards, Wallet, Marketplace, Foundation, Profile.
- Offline-кеш базовых данных.
- Уведомления (push), deep-links на конкретных майнеров и кампании фонда.

---

3. Отдельный блок: механика фонда и отчётность

Это критически важная часть — **фатическая и репутационная опора проекта**. Схему лучше делать максимально прозрачной.

3.1. Источники поступлений в фонд

1. **Протокольная доля от maintenance**
   - maintenance-engine отдаёт breakdown:
     - elec_usd, service_usd, charity_share_usd.
   - charity_share_usd конвертируется в выбранный asset (например, USDT или BTC) и отправляется в CharityVault.
2. **Доля от marketplace-комиссий**
   - При исполнении сделки:
     - fee_protocol и fee_charity.
   - fee_charity → сразу в CharityVault.
3. **Burn-окно**
   - В окне, когда BurnScheduler сжигает TYT, можно заложить механику:
     - после продажи TYT за стейблы/ETH часть выручки (1–2%) идёт в CharityVault.
4. **Добровольные донаты пользователей**
   - В интерфейсе:
     - «Отправить X TYT / USDT / BTC в фонд».
   - wallet-service создаёт ledger-запись с sourceType = userDonation.
5. **Charity-майнеры**
   - Особый тип NFT:

    - Характеристика: charity_pct — % дохода, который идёт в фонд.
- rewards-engine при начислении:
    - делит net BTC → часть пользователю, часть фонду.

3.2. Модель данных для фонда

Таблица charity_flows:

- id
- source_type (enum: MAINTENANCE_FEE, MARKETPLACE_FEE, BURN_WINDOW, USER_DONATION, CHARITY_MINER)
- source_id (invoice_id / trade_id / burn_id / tx_id / miner_id)
- user_id (nullable, если относится к конкретному пользователю)
- asset (BTC, USDT, TYT, …)
- amount
- tx_hash (если onchain)
- created_at

Таблица charity_reports:

- id
- period_start, period_end
- total_in_by_asset_json (сумма входящих по asset)
- total_out_by_asset_json
- summary_markdown (человекочитаемый отчёт)
- external_links_json (ссылки на pdf, сканы, отчёты клиник)
- created_at

Таблица charity_campaigns (конкретные проекты):

- id
- title
- description
- target_amount_usd
- collected_amount_usd
- status (planning/active/completed)

- attachments_json (фото/видео/ссылки)
- created_at

Связующая таблица:

- charity_allocations(id, report_id, campaign_id, asset, amount, note)

3.3. Логика работы charity-service

1. **Приём средств**
   - REST/gRPC от других сервисов:
     - /internal/notify-income
       Body: {source_type, source_id, asset, amount}.
   - Заводит запись в charity_flows, триггерит обновление агрегатов.
2. **Вывод средств**
   - Только через FOUNDATION_MULTISIG (onchain).
   - Admin-панель:
     - выбирает кампанию,
     - заполняет сумму, получателя, назначение,
     - вызывает blockchain-gateway → CharityVault.withdraw(...).
   - После подтверждения tx:
     - charity_flows (out),
     - charity_allocations (раскидывание по кампаниям).
3. **Генерация отчётов**
   - Периодически (месяц/квартал):
     - агрегирует charity_flows за период,
     - рассчитывает totals,
     - создаёт charity_reports с summary_markdown (который можно редактировать через admin).
   - API:
     - GET /public/reports — список отчётов,
     - GET /public/reports/{id} — детализация + ссылки.

3.4. Отображение фонда во фронте

Страница /foundation:

- Хедер: логотип фонда + манифест (что мы делаем, кого поддерживаем).
- Cards:
    - **Total donated** (по каждому asset, эквивалент в USD).
    - **Last 5 donations** (тип, сумма, источник).
    - **Active campaigns** (название, прогресс bar).
- Блок «Latest report»:
    - дата,
    - ключевые цифры (сумма, количество детей/кейсов, клиники),
    - ссылка на pdf/подробности.
- Кнопки:
    - «Пожертвовать» (popup с выбором asset и суммы),
    - «Привязать майнер к постоянным пожертвованиям» (% slider).

Ладно, вот то самое «копипаст-оружие» для агентов 🚀
Сделаю 4 отдельных, самодостаточных промпта на английском (агентам так проще), ты просто копируешь каждый блок целиком в bolt.new / v0 / Lovable.

---

0. Общий контекст (можно вставлять в начало любого промпта)

Если хочешь, можешь предварять любой из промптов этим блоком:

You are an expert senior engineer working on a Web3 project called **TakeYourToken.app (TYT)**.

High-level concept:
- TYT is a legally clean competitor to GoMining.
- Users buy **NFT miners** (tokenized hashrate contracts, NOT hardware shares).
- Platform distributes **daily BTC-indexed rewards** to a custodial wallet, AFTER electricity + service maintenance fees.
- Users get **discounts** when paying maintenance in **TYT token**.
- There is a **P2P marketplace** for NFT miners, **VIP ranks**, **gamification (Owlverse)** and **TYT Digital Crypto Academy**.
- The third, most important pillar: **TYT Brain Cancer Children’s Research & Support Foundation**:
  - A fixed and/or governance-adjustable % of protocol revenue flows into a **foundation treasury**.
  - Users can optionally donate parts of their rewards or have special “charity miners”.
  - There is a public **Charity Dashboard** and periodic transparent reports.

Key chains:
- NFT miners & governance & some utility live on **Polygon PoS (EVM)**.

- TYT token lives on **Solana (SPL)** (already launched as a mem token), used as a utility token and bridged where needed.
- BTC rewards are held custodially off-chain, with on-chain verifiability via Merkle roots.

Core modules:
- On-chain: MinerNFT, veTYT, Marketplace, RewardsMerkleRegistry, CharityVault.
- Off-chain: auth/KYC, wallets, rewards-engine, maintenance-engine, marketplace service, governance, academy, charity-service.
- Frontend: Next.js web dApp + later mobile app.

——

1) PROMPT: Contracts Agent

Скопируй этот блок в агента, который пишет смарт-контракты:

You are **Contracts-Agent**, a senior smart-contract engineer.

Goal:
Design and implement the on-chain layer for **TakeYourToken.app (TYT)** on **Polygon (EVM)** and **Solana (SPL/Anchor)**. Focus on security, upgradability (where needed), and clear separation of concerns.

### Tech constraints

- EVM:
  - Solidity ^0.8.20
  - Use OpenZeppelin libraries (ERC-721, ERC-20, AccessControl, EIP-2981 where needed).
  - Use UUPS or Transparent upgradeable pattern only when strictly required.
- Solana:
  - Use **Anchor** framework for SBT/NFT program for the Academy.
- Follow best practices: checks-effects-interactions, reentrancy guards, minimal public surface.

## Deliverables

Design and implement the following contracts with clear interfaces, events, and NatSpec documentation.

### 1. MinerNFT (Polygon, ERC-721)

Purpose:
- Represents a user's **NFT miner** with attached parameters:
  - hashrate
  - efficiency
  - region
  - maintenance parameters
  - status flags

Requirements:
- Inherit from ERC721 + AccessControl.
- Per-token state:
  - `powerTH` (uint96) — TH/s scaled by 1e6.
  - `efficiencyWPerTH` (uint96) — W/TH scaled by 1e6.
  - `regionId` (uint16) — ID of data-center region.
  - `maintenanceRateBps` (uint16) — maintenance overhead in basis points.

  - `status` (uint8) — 0=Active,1=Delinquent,2=Locked/Paused.
- Roles:
  - `DEFAULT_ADMIN_ROLE`
  - `MINTER_ROLE`
  - `UPGRADER_ROLE`
  - `OPERATOR_ROLE`
- Functions:
  - `mintMiner(address to, MinerParams calldata params)` — only `MINTER_ROLE`.
  - `upgradeHashrate(uint256 tokenId, uint96 newPowerTH)` — only `UPGRADER_ROLE`.
  - `upgradeEfficiency(uint256 tokenId, uint96 newEffWPerTH)` — only `UPGRADER_ROLE`.
  - `setStatus(uint256 tokenId, uint8 newStatus)` — only `OPERATOR_ROLE`.
- Events:
  - `MinerMinted(uint256 indexed tokenId, address indexed to, MinerParams params)`
  - `MinerUpgraded(uint256 indexed tokenId, uint96 powerTH, uint96 effWPerTH)`
  - `MinerStatusChanged(uint256 indexed tokenId, uint8 status)`
- Optional:
  - Implement EIP-2981 royalties so that protocol and/or seller share can be configured for marketplace trades.
- Provide a minimal `IMinerNFT` interface.

### 2. RewardsMerkleRegistry (Polygon)

Purpose:
- Store a daily Merkle root of rewards distribution for all miners for on-chain verifiability.

Requirements:
- Mapping `day => bytes32 merkleRoot`.
- Role: `REWARDS_ORACLE_ROLE`.
- Functions:
  - `setRewardsRoot(uint256 day, bytes32 root)` — only REWARDS_ORACLE_ROLE, cannot overwrite existing day.
  - `getRewardsRoot(uint256 day)` view.
- Events:
  - `RewardsRootSet(uint256 indexed day, bytes32 root)`.
- Prevent re-setting root for a given day once set.

### 3. veTYT (Polygon)

Purpose:
- Lock bridged TYT tokens (ERC-20) for a defined period and get voting power for governance and boosted discounts.

Requirements:
- Takes in a specific ERC-20 token address (wrapped / bridged TYT).
- Lock model:
  - `struct Lock { uint256 amount; uint256 start; uint256 end; }`
  - `votingPower = amount * f(lockDuration)` where `lockDuration` max is e.g. 4 years, and f is linear or similar.
- Functions:
  - `createLock(uint256 amount, uint256 lockDuration)` — transfers tokens from user into contract.
  - `increaseAmount(uint256 lockId, uint256 addedAmount)`.
  - `increaseDuration(uint256 lockId, uint256 addedDuration)` (up to max).
  - `withdraw(uint256 lockId)` — only after `end`.
  - `getVotingPower(address user)` view.
- Emit events for each action (Created/AmountIncreased/DurationIncreased/Withdrawn).

### 4. MinerMarketplace (Polygon)

Purpose:
- P2P marketplace for MinerNFT with protocol + charity fees.

Requirements:
- Integrate with MinerNFT contract.
- Basic order structure:
```solidity
struct Order {
    address seller;
    uint256 tokenId;
    uint256 price;
    address asset; // ERC-20 address
    bool active;
}
```

- State:
- mapping(uint256 => Order) public orders;
- incremental orderId.
- Parameters:
- protocolFeeBps
- charityFeeBps
- protocolFeeRecipient
- charityVault (address of CharityVault contract).
- Functions:
- list(uint256 tokenId, uint256 price, address asset):
- transfers NFT from seller to marketplace (escrow).
- buy(uint256 orderId):
- transfers asset from buyer to seller + protocol + charity.
- transfers NFT from marketplace to buyer.
- cancel(uint256 orderId):
- only seller, if active, returns NFT.
- Events:
- OrderCreated(orderId, seller, tokenId, price, asset)
- OrderExecuted(orderId, buyer, tokenId, price, asset, protocolFee, charityFee)
- OrderCancelled(orderId)
- Include reentrancy protection.

5. CharityVault (Polygon)

Purpose:
- Central on-chain vault for all protocol-level donations to the Brain Cancer Children's Research & Support Foundation.

Requirements:
- Accept ETH and multiple ERC-20 tokens (including TYT).
- Track totals per asset.
- Role: FOUNDATION_MULTISIG_ROLE allowed to withdraw funds.
- State:
- mapping(address => uint256) public totalReceived;
- mapping(address => uint256) public totalWithdrawn;
- Functions:
- donate(address asset, uint256 amount, uint8 sourceType, uint256 sourceId):
- For ERC-20, transferFrom from msg.sender.
- For ETH, use payable function variant.
- withdraw(address asset, uint256 amount, address to, string calldata memo) only FOUNDATION_MULTISIG_ROLE.
- Events:

- DonationReceived(address indexed asset, uint256 amount, uint8 sourceType, uint256 sourceId, address indexed from)
- CharityWithdrawal(address indexed asset, uint256 amount, address indexed to, string memo)
- sourceType is an enum-like uint8 (MAINTENANCE_FEE, MARKETPLACE_FEE, BURN_WINDOW, USER_DONATION, CHARITY_MINER).

6. Solana / Anchor: TYT Academy SBT

Purpose:
- Non-transferable NFT-like Soulbound tokens for course completion in the TYT Academy.

Requirements:
- Anchor program tyt_academy_sbt:
- Mint 1 SBT per completed course.
- Prevent transfers (by design using token-2022 transfer hook or program-logic restrictions).
- Metadata:
- course_id, level, issued_at.
- Provide clear Rust / Anchor code structure and IDL.

⸻

Output format
- For each contract:
- Final Solidity / Rust code.
- Brief README or comments describing usage and roles.
- Include minimal test examples or pseudo-tests where helpful.

## 2) PROMPT: **Backend Agent**

Скопируй в бэкенд-агента:

```text
You are **Backend-Agent**, a senior backend architect & engineer.

Goal:
Design and implement the **backend microservice architecture** for TakeYourToken.app (TYT).
Focus on:
- clean separation of concerns,
- security and auditability,
- easy integration with smart contracts and frontends,
- strong support for the TYT Brain Cancer Children's Research & Support Foundation.

### Tech stack

- Language: TypeScript
- Framework: **NestJS**
- Databases: **PostgreSQL** (primary), **Redis** (caching, sessions)
- Messaging: **Kafka** or **RabbitMQ** (events between services)
- API style: REST + optional gRPC for internal services
- ORM: Prisma or TypeORM (choose one and configure properly)
- All services containerized (Docker) and ready for CI/CD.

---
```

## Required microservices (MVP set)

Implement the following services as separate NestJS apps within a monorepo:

1. `auth-service`
2. `kyc-service`
3. `user-profile-service`
4. `wallet-service`
5. `blockchain-gateway-service`
6. `miner-registry-service`
7. `maintenance-engine-service`
8. `rewards-engine-service`
9. `marketplace-service`
10. `governance-service`
11. `rank-and-gamification-service`
12. `academy-service`
13. `charity-service`
14. `notification-service`
15. `admin-panel-service` (API only, frontend separate)

You do NOT need to fully implement business logic for every edge case; focus on clear, extensible structure, data models, and the main API flows.

Below are the key responsibilities and minimal API contracts per service.

### 1. auth-service

Responsibilities:
- User registration / login.
- JWT issuing and refresh.
- 2FA support (TOTP).

API:
- `POST /auth/register`
- `POST /auth/login`
- `POST /auth/refresh`
- `POST /auth/enable-2fa`
- `POST /auth/verify-2fa`

DB tables:
- `users(id, email, password_hash, status, created_at)`
- `user_sessions(id, user_id, ua, ip, expired_at)`

### 2. kyc-service

Responsibilities:
- Integrate with external KYC provider (e.g. Sumsub / Onfido).
- Track KYC case status.

API:
- `POST /kyc/start` — initiate KYC for user.
- `GET /kyc/status` — get current KYC status for user.
- `POST /kyc/webhook` — callback from provider.

DB:
- `kyc_cases(id, user_id, provider_case_id, status, updated_at)`

### 3. user-profile-service

Responsibilities:
- User profile and preferences.

API:
- `GET /profile/me`
- `PATCH /profile/me`
- `GET /profile/settings`
- `PATCH /profile/settings`

DB:
- `user_profiles(user_id, nickname, lang, tz, notif_prefs_json)`

### 4. wallet-service

Responsibilities:
- Internal custodial wallet accounts for BTC, stablecoins, TYT, etc.
- Ledger of debits and credits.
- Initiating deposits / withdrawals via blockchain-gateway.

API:
- `GET /wallet/accounts`
- `GET /wallet/balance?asset=...`
- `POST /wallet/withdraw` (after KYC + checks)
- Internal:
  - `POST /wallet/internal-transfer`
  - `POST /wallet/apply-reward` (called by rewards-engine)
  - `POST /wallet/apply-donation` (called by charity-service or foundation UI)

DB:
- `wallet_accounts(id, user_id, asset, external_address, created_at)`
- `ledger_entries(id, account_id, debit, credit, ref_type, ref_id, ts)`

### 5. blockchain-gateway-service

Responsibilities:
- Interact with BTC, Polygon, Solana, etc.
- Listen for deposits.
- Send withdrawals and protocol transactions (burn, charity, etc).

API:
- `POST /blockchain/send-tx` — generic call distribution by chain.
- `POST /blockchain/register-deposit-address`
- `POST /blockchain/notify-tx` — internal callback when new tx is detected.

It should provide typed clients for:
- Polygon RPC (MinerNFT, Marketplace, RewardsMerkleRegistry, CharityVault, veTYT).
- Solana RPC / Anchor IDL for Academy SBT + TYT SPL token.

- BTC API provider.

### 6. miner-registry-service

Responsibilities:
- Off-chain mirror of on-chain MinerNFT.
- Ownership, parameters, status, reinvest percentage.

API:
- `GET /miners/my`
- `GET /miners/:id`
- `POST /miners/:id/reinvest-config` (percentage of rewards auto-reinvested, or auto-donated)
- Internal sync jobs to:
  - read events from MinerNFT contract,
  - update database accordingly.

DB:
- `miners(id, nft_token_id, owner_id, power_th, eff_tier, region_id, status, reinvest_pct, created_at)`
- `regions(id, name, country, tz, note)`

### 7. maintenance-engine-service

Responsibilities:
- Calculate per-period maintenance costs (electricity + service fee).
- Apply discount curve based on TYT usage, VIP rank, prepayment days, etc.

API:
- `POST /maintenance/calculate` with input:
  - `power_th`
  - `efficiency_w_per_th`
  - `region_id`
  - `date_range`
  - `discount_profile` (vip_level, has_tyt_discount, prepay_days, etc.)
- Returns:
  - `elec_usd`, `service_usd`, `total_usd`, `discount_pct`, `curve_tier`.

DB:
- `fees(region_id, kwh_usd, service_bps, updated_at)`
- `maintenance_invoices(id, miner_id, period_start, period_end, amount_usd, discount_pct, asset, status)`

### 8. rewards-engine-service

Responsibilities:
- Daily BTC rewards distribution.
- Integration with maintenance-engine for net calculations.
- Writing Merkle tree to on-chain RewardsMerkleRegistry.

Flow:
1. Get daily BTC pool (`gross_btc`) for date D.
2. For all active miners:
   - Calculate gross share by TH.

   - Ask maintenance-engine for maintenance cost.
   - Calculate `net_btc` per miner.
3. Write entries:
   - Into `rewards` table.
   - Into `wallet-service` via internal API.
4. Build Merkle tree of `[minerId, net_btc, date]`.
5. Send root to RewardsMerkleRegistry via blockchain-gateway.

API:
- `GET /rewards?user_id=...`
- `GET /rewards/summary?user_id=...`
- Internal endpoint (trigger or cron) to run daily job.

DB:
- `daily_pool(date, gross_btc, price_btc_usd)`
- `rewards(date, miner_id, gross_btc, elec_usd, service_usd, discount_pct, net_btc, proof_leaf)`

### 9. marketplace-service

Responsibilities:
- Off-chain business layer for MinerMarketplace.
- Index on-chain orders and trades.
- Filter, search, analytics.

API:
- `GET /marketplace/orders`
- `GET /marketplace/orders/:id`
- `POST /marketplace/orders` — optionally prepare tx data for frontend to call on-chain marketplace contract.
- `GET /marketplace/my-trades`

DB:
- `orders(id, miner_id, seller_id, price, asset, status, created_at)`
- `trade_events(id, order_id, buyer_id, amount, fee_protocol, fee_charity, ts)`

### 10. governance-service

Responsibilities:
- Manage proposals and votes (off-chain structured, on-chain anchored via veTYT / governance contracts).
- Parameters include: discount curve, charity share, burn cadence, etc.

API:
- `GET /governance/proposals`
- `GET /governance/proposals/:id`
- `POST /governance/proposals` (admin / multi-sig).
- `POST /governance/proposals/:id/vote`
- `GET /governance/user-voting-power`

DB:
- `locks(id, user_id, amount_tyt, locked_until, voting_power)` or mirrored from veTYT.
- `proposals(id, title, description, param_key, status, created_at)`
- `votes(id, proposal_id, user_id, voting_power, choice)`

### 11. rank-and-gamification-service

Responsibilities:
- Calculate and store Owlverse ranks and badges.

Inputs:
- sum TH
- veTYT voting power
- payment history (maintenance)
- academy progress
- referral metrics

API:
- `GET /ranks/me`
- `GET /ranks/leaderboard`

DB:
- `user_rank_state(user_id, rank, rank_score, updated_at)`
- `user_badges(id, user_id, badge_code, earned_at, source)`

### 12. academy-service

Responsibilities:
- TYT Digital Crypto Academy: courses, lessons, quizzes, certification.

API:
- `GET /academy/courses`
- `GET /academy/courses/:id`
- `GET /academy/courses/:id/progress`
- `POST /academy/courses/:id/complete-lesson`
- `POST /academy/courses/:id/finish` — issues SBT via blockchain-gateway.

DB:
- `courses(id, slug, title, level, meta_json)`
- `lessons(id, course_id, order, slug, title)`
- `user_course_progress(user_id, course_id, status, score)`
- `user_quiz_attempts(user_id, lesson_id, result, ts)`

### 13. charity-service (Foundation core)

Responsibilities:
- Track all flows into and out of the TYT Brain Cancer Children's Research & Support Foundation.
- Generate public reports and per-campaign breakdowns.

Sources of income:
- Share of maintenance fees.
- Share of marketplace fees.
- Cut of burn-window conversions.
- Explicit user donations.
- Charity miners' share of rewards.

API:

- Internal:
  - `POST /charity/income` with `{source_type, source_id, user_id?, asset, amount}`.
  - `POST /charity/allocate` (allocate amount to specific campaign & report).
- Public:
  - `GET /charity/summary`
  - `GET /charity/reports`
  - `GET /charity/reports/:id`
  - `GET /charity/campaigns`
- Admin:
  - `POST /charity/campaigns`
  - `PATCH /charity/campaigns/:id`
  - `POST /charity/withdraw` (calls blockchain-gateway -> CharityVault.withdraw).

DB:
- `charity_flows(id, source_type, source_id, user_id, asset, amount, tx_hash, created_at)`
- `charity_reports(id, period_start, period_end, total_in_by_asset_json, total_out_by_asset_json, summary_markdown, external_links_json, created_at)`
- `charity_campaigns(id, title, description, target_amount_usd, collected_amount_usd, status, attachments_json, created_at)`
- `charity_allocations(id, report_id, campaign_id, asset, amount, note)`

### 14. notification-service

Responsibilities:
- Unified notifications: email, Telegram, push.

API:
- `POST /notifications/send` (internal)
- `GET /notifications/history`

DB:
- `notifications(id, user_id, channel, template, payload_json, status, ts)`

Use templating engine (e.g. Handlebars) and provider integrations (SendGrid / SMTP, Telegram bot API, FCM/APNs).

### 15. admin-panel-service

Responsibilities:
- Provide admin APIs for all moderation and configuration tasks.

API (examples):
- `GET /admin/users`
- `GET /admin/users/:id`
- `GET /admin/miners`
- `GET /admin/charity/flows`
- `POST /admin/fees/update`

## Output

- Monorepo structure suggestion.
- Dockerfiles for each service.

- Detailed NestJS modules, DTOs, entities, and example controllers for each service.
- SQL or Prisma schema for all tables above.

3) PROMPT: Frontend Agent

Скопируй в фронтенд-агента:

You are **Frontend-Agent**, a senior web frontend engineer and UX designer.

Goal:
Build the **web dApp** for TakeYourToken.app (TYT) using Next.js, with a landing page and an authenticated app area (`/app/*`).

### Tech stack

- Framework: **Next.js 14+ (App Router)**
- Language: TypeScript
- Styling: TailwindCSS
- State: React Query / TanStack Query + Context where needed
- UI:
  - Dark-first design with metallic, subtle neon accents (owl/knight/shield/sword theme).
  - Minimal but premium, Web3-crypto style.
- Integrations:
  - REST APIs from backend-gateway.
  - Web3: EVM provider (WalletConnect / MetaMask) and Solana (Phantom) where needed.

## Pages and flows

Implement the following main routes:

### 1. `/` — Landing page (pre-launch / marketing)

Sections:
1. **Hero**
   - Logo (owl/knight/shield/sword motif).
   - Headline: “Own NFT Miners. Earn BTC Each Day. Support Children’s Brain Cancer Research.”
   - CTA buttons: “Launch App” -> `/app`, “Learn More” -> scroll to “How it works”.

2. **How it works**
   - 4 steps:
     1) Sign up & pass KYC.
     2) Buy NFT miners on Polygon.
     3) Receive daily BTC rewards (after maintenance).
     4) Upgrade, trade miners, and support the Foundation.

3. **Income calculator**
   - Form fields:
     - TH/s (slider + input)
     - Efficiency (W/TH)
     - BTC price
     - kWh price
   - Live calculation:
     - daily/weekly/monthly net BTC (using a mocked formula initially).

4. **TYT Token & Burning**
   - Explain that paying maintenance in TYT gets discounts.
   - Show a dummy chart “Burned vs Issued TYT” (static data, to be wired later).

5. **Fund section**
   - Brief description of the TYT Brain Cancer Children’s Research & Support Foundation.
   - Key stats (hardcoded placeholders):
     - total donated
     - number of campaigns
   - CTA “View Foundation Dashboard” -> `/foundation`.

6. **Footer**
   - Links to docs, Terms of Use, Privacy, Telegram, etc.

### 2. `/app` — Auth wrapper

- If not authenticated → redirect to `/app/login`.
- Contains main layout: sidebar navigation + top bar + content area.

Navigation entries:
- Dashboard
- My Miners
- Rewards
- Wallet
- Marketplace
- Academy
- Foundation
- Profile / Settings

### 3. `/app/login` & `/app/register`

- Simple forms with:
  - email/password
  - 2FA code (if enabled)
- Call backend `auth-service` endpoints.
- Handle errors gracefully.

### 4. `/app/dashboard`

Show user’s high-level summary:

- Total BTC balance, TYT balance, total NFT miners.
- Estimated daily BTC reward.
- Unpaid maintenance invoices.
- Current Owlverse rank (Worker / Academic / Diplomat / Peacekeeper / Warrior) with visual badge.
- Shortcut cards:
  - “Buy Miners”
  - “Pay Maintenance”
  - “Donate to Foundation”
  - “Start Academy Course”

Use responsive grid, charts for history (mocked initial data).

### 5. `/app/miners`

- Table/list of all user miners with:
  - Miner ID, region, power TH/s, efficiency, status (Active/Delinquent).
  - Reinvest/charity configuration (small badge or toggle).
- Detail page `/app/miners/[id]`:
  - Miner info.
  - Reward history for this miner.
  - Maintenance cost breakdown (electricity + service + discounts).
  - Actions:
    - “Upgrade Hashrate”
    - “Upgrade Efficiency”
    - “Configure Reinvest %”
    - “Donate % of rewards to Foundation” (slider)

For now, use mock data and define TypeScript interfaces matching backend models.

### 6. `/app/rewards`

- Rewards history table:
  - Date, total gross BTC, maintenance, discount %, net BTC.
- Filters by date range.
- Summary cards:
  - Total earned (lifetime).
  - Last 30 days.
  - Last 7 days.
- Export button (CSV) calling backend when ready.

### 7. `/app/wallet`

- Show balances per asset: BTC, USDT, USDC, TYT, etc.
- Actions:
  - “Deposit” (show addresses / deposit instructions).
  - “Withdraw” (form with amount, address).
- TX history list:
  - Type (deposit, withdraw, reward, donation, trade, maintenance).
  - Asset, amount, date, status.

### 8. `/app/marketplace`

- List all miner orders:
  - Miner preview, seller rank, price, asset (USDT/TYT).
  - Filters: by power range, region, price range, asset.
- Detail view:
  - Full miner params, ROI-style info (without fixed APR promises).
  - “Buy now” button → prepare tx data for on-chain marketplace.
- “My listings” tab with list/cancel options.

### 9. `/app/academy`

- List of Academy courses:
  - Title, difficulty, estimated time, progress.
- Course detail:
  - Lessons list.
  - Lesson view with content (Markdown/MDX rendered).
  - Simple quiz at end.
- On “Finish course”:
  - Call backend to issue SBT via blockchain-gateway (async), then show success state.

### 10. `/foundation` (public) + `/app/foundation` (in-app)

Public `/foundation`:
- Overview of the Foundation:
  - Mission, focus on children’s brain cancer research and family support.
- Key metrics (from backend when available, mocked otherwise):
  - Total donated by asset.
  - Latest report (card with date and short summary).
  - List of active campaigns with progress bars.
- CTA “Donate now”:
  - Modal with choice of asset and amount; for now, send to `/app/wallet` or show placeholder.

In-app `/app/foundation`:
- Same as above plus:
  - User-specific stats:
    - Total donated by this user.
    - % of miner rewards configured as donation.
  - Quick toggles:
    - “Donate 1%/3%/5% of all future rewards”.
  - Link to view transaction history of donations.

### 11. `/app/profile`

- User profile and settings:
  - Email, nickname, language, timezone, notification preferences.
  - KYC status indicator.
  - 2FA settings.
- Integrate with `user-profile-service` & `kyc-service`.

## Integration assumptions

- There will be an API gateway endpoint (base URL, e.g. `/api`) aggregating all backend services.
- For now, you can define type-safe `apiClient` wrappers (e.g. using Axios + Zod).
- Use mocked data where APIs are not yet implemented, but:
  - Keep TypeScript models aligned with the backend schemas described.

## Output

- Next.js app structure (App Router) with:
  - `app/(public)/page.tsx` for landing.
  - `app/(auth)/app/*` for protected routes.
- Reusable components:
  - Layout, Sidebar, TopBar, Cards, Charts, Tables, Modal, RankBadges, CampaignCards.
- Example API hooks:
  - `useUser()`, `useMiners()`, `useRewards()`, `useWallet()`, `useFoundationSummary()`, etc.
- Tailwind-based styling consistent with the owl/knight/shield/sword & crypto-mining brand.

---

4) PROMPT: Infra Agent

И наконец промпт для infra/DevOps-агента:

You are **Infra-Agent**, a senior DevOps / platform engineer.

Goal:
Design the **infrastructure, deployment, and CI/CD** for the TYT (TakeYourToken.app) platform so that all services (backend microservices + web frontend) can be reliably developed, tested, and deployed.

### Tech assumptions

- Monorepo (e.g. using pnpm / TurboRepo / Nx) containing:
  - `contracts/` (Solidity + Hardhat/Foundry)
  - `backend/` (multiple NestJS services)
  - `frontend/` (Next.js app)
- Containerization: **Docker** for every service.
- Orchestration:
  - Start with **docker-compose** for local/dev,
  - Prepare manifests for **Kubernetes** (K8s) for staging/production.
- CI/CD: GitHub Actions (or GitLab CI) pipelines.

## Tasks

### 1. Monorepo structure

Define a folder structure, e.g.:

- `/contracts` — Hardhat or Foundry setup for EVM contracts + Anchor for Solana.
- `/backend/services/<service-name>` — each NestJS microservice.
- `/frontend/web` — Next.js app.

Within your output:
- Explain the exact structure and the main config files (tsconfig, package.json, etc.).
- Ensure easy local setup: `pnpm install` and `pnpm dev` (or similar) to run multiple services.

### 2. Dockerization

For each type of component, create a Dockerfile template:

- **Backend service (NestJS):**
  - Multi-stage build:
    - Stage 1: node:20-alpine for building.
    - Stage 2: node:20-alpine for running, copy built dist + node_modules (prod).
  - Environment variables via `.env` or K8s ConfigMaps.
- **Frontend (Next.js):**
  - Multi-stage build:
    - Build stage with dependencies.
    - Run stage using `node:20-alpine`, `next start`.
- **Contracts tooling:**
  - Optional: a small image to run tests / deployments (Hardhat or Foundry) in CI.

Also provide a `docker-compose.yml` example for local:
- Services:
  - `postgres`
  - `redis`
  - `kafka` (or `rabbitmq`)
  - `auth-service`
  - `wallet-service`
  - `rewards-engine-service`
  - `charity-service`
  - `api-gateway` (optionally)
  - `frontend-web`

Include volumes and healthchecks where relevant.

### 3. Environment configuration

Define environment variable sets for:

- `LOCAL`
- `STAGING`
- `PRODUCTION`

Examples:
- DB:
  - `POSTGRES_HOST`, `POSTGRES_PORT`, `POSTGRES_DB`, `POSTGRES_USER`, `POSTGRES_PASSWORD`
- Redis:
  - `REDIS_HOST`, `REDIS_PORT`
- Kafka/RabbitMQ:
  - `MESSAGE_BROKER_URL`
- Auth:
  - `JWT_SECRET`, `JWT_EXPIRES_IN`
- Blockchain:
  - `POLYGON_RPC_URL`, `SOLANA_RPC_URL`, `BTC_PROVIDER_URL`
  - `MINER_NFT_ADDRESS`, `MARKETPLACE_ADDRESS`, `REWARDS_REGISTRY_ADDRESS`, `CHARITY_VAULT_ADDRESS`, `VETYT_ADDRESS`
- Foundation:
  - `FOUNDATION_MULTISIG_ADDRESS`

Output should include example `.env.local` / `.env.staging` / `.env.prod` templates.

### 4. CI/CD pipelines

Use **GitHub Actions** as an example CI provider (but keep it generic enough to adapt).

Pipelines:

1. **Contracts pipeline**
   - Trigger: PRs touching `/contracts`.
   - Steps:
     - Install dependencies.
     - Run `npm test` or `forge test`.
     - Static analysis (e.g. Slither if feasible).
   - Output: test reports, artifacts.

2. **Backend pipeline**
   - Trigger: PRs touching `/backend` or shared libs.
   - Steps:
     - Install dependencies.
     - Run unit tests for each service.
     - Run lint (ESLint).
     - Build Docker images and push to registry on main/staging branch merges.

3. **Frontend pipeline**
   - Trigger: PRs touching `/frontend`.
   - Steps:
     - Install deps.
     - Run `lint`, `typecheck`, `test`, `build`.
     - Build and push Docker image.

4. **Deploy pipeline**
   - Trigger: tag or manual dispatch.
   - Steps:
     - Pull latest images from registry.
     - Apply K8s manifests (staging/prod).
     - Run DB migrations (Prisma/TypeORM migrations).
     - Notify on Slack/Telegram.

Provide example GitHub Actions YAML files.

### 5. Kubernetes manifests (conceptual)

Define K8s manifests templates for:

- `Deployment` per service (auth, wallet, rewards, charity, frontend).
- `Service` (ClusterIP) per backend, LoadBalancer/Ingress for frontend & API gateway.
- ConfigMaps for non-secret configuration.
- Secrets for sensitive env variables (DB passwords, JWT secret, RPC keys).
- Optionally:
  - Horizontal Pod Autoscaler for critical services (auth, rewards-engine, wallet-service).
  - PodDisruptionBudgets.

You don't have to write full verbose YAML for every service, but provide at least one complete example (e.g. for `auth-service` and `frontend-web`) and show how it generalizes.

### 6. Observability

Design a minimal observability stack:

- Logging:
  - Structured JSON logs from all services.
  - Central log aggregation (e.g. ELK stack or Loki).
- Metrics:
  - Prometheus scraping.
  - Grafana dashboards for:
    - Reward distribution success.
    - Maintenance invoice generation.
    - Charity flows and vault withdrawals.
- Alerts:
  - Simple examples of alert rules (e.g. no rewards job run in 24h, error rate spike, etc).

## Output

- Proposed monorepo tree.
- Dockerfiles templates.
- docker-compose for local dev.
- Skeleton GitHub Actions workflows.
- Example Kubernetes manifests.
- Explanation of how to run everything locally and how to promote to staging/prod.

———

Общий контекст (опционально к любому микропромпту)

You are an expert senior engineer working on a Web3 project called **TakeYourToken.app (TYT)**.

High-level concept:

> Users buy **NFT miners** (tokenized hashrate contracts, NOT hardware shares).
> Platform distributes **daily BTC-indexed rewards** to a custodial wallet, AFTER electricity + service maintenance fees.
> Paying maintenance in **TYT token** gives discounts and feeds burning + utility loops.
> There is a **P2P marketplace**, **VIP Owlverse ranks**, a **TYT Crypto Academy**, and a **Brain Cancer Children's Research & Support Foundation** funded by protocol fees and user donations.

Key stacks:

> **Polygon (EVM)** for NFT miners & governance.
> **Solana (SPL)** for TYT token, bridged if needed.
> Backend: NestJS microservices + Postgres.
> Frontend: Next.js web dApp.

Микро-промпты для контрактов
1.1. MinerNFT (ядро майнеров)

Task: Implement a secure, gas-efficient **MinerNFT** contract on Polygon.

Requirements:

- Solidity ^0.8.20, OpenZeppelin ERC721 + AccessControl.
- Per-token data: struct Miner { uint96 powerTH; // TH/s * 1e6 uint96 efficiencyWPerTH; // W/TH * 1e6 uint16 regionId; // data-center region uint16 maintenanceRateBps; // service overhead, basis points uint8 status; // 0=Active, 1=Delinquent, 2=Locked }
- Roles: DEFAULT_ADMIN_ROLE, MINTER_ROLE, UPGRADER_ROLE, OPERATOR_ROLE.
- Functions:
  - mintMiner(address to, Miner calldata m) only MINTER_ROLE
  - upgradeHashrate(tokenId, newPowerTH) only UPGRADER_ROLE
  - upgradeEfficiency(tokenId, newEff) only UPGRADER_ROLE
  - setStatus(tokenId, newStatus) only OPERATOR_ROLE
- Events for mint/upgrade/status change.
- Include minimal IMinerNFT interface. Provide full Solidity code + brief explanation.

1.2. RewardsMerkleRegistry (корень наград)

Task: Implement **RewardsMerkleRegistry** contract.

Requirements:

- Store daily Merkle roots for rewards: mapping(uint256 day => bytes32 merkleRoot)
- Role: REWARDS_ORACLE_ROLE (AccessControl).
- Functions:
  - setRewardsRoot(uint256 day, bytes32 root) external onlyOracle:
    - cannot overwrite non-zero root for that day.
  - getRewardsRoot(uint256 day) view returns (bytes32)
- Event: RewardsRootSet(day, root). Provide Solidity code with NatSpec comments.

1.3. veTYT (локи & голос)

Task: Implement **veTYT** contract for time-locked voting power.

Requirements:

- Accepts an ERC20 token address (wrapped TYT).
- Lock model: struct Lock { uint256 amount; uint256 start; uint256 end; }
- Functions:
  - createLock(amount, lockDuration) -> lockId
  - increaseAmount(lockId, addedAmount)
  - increaseDuration(lockId, addedDuration) up to MAX_LOCK
  - withdraw(lockId) after end: return tokens to user.
- Voting power:
  - linear function of duration (0..MAX_LOCK, e.g. 4 years).
  - getVotingPower(address user) returns sum of active locks.
- Emit events for each operation. Write full Solidity code and briefly describe how governance-service can query it.

1.4. MinerMarketplace (маркет)

Task: Implement **MinerMarketplace** contract for trading MinerNFTs.

Requirements:

- Uses IMinerNFT interface.
- Order struct: struct Order { address seller; uint256 tokenId; uint256 price; address asset; // ERC20 bool active; }
- State:
  - mapping(uint256 => Order) public orders;
  - uint256 public nextOrderId;
- Config:
  - uint16 protocolFeeBps;
  - uint16 charityFeeBps;
  - address protocolFeeRecipient;
  - address charityVault; // CharityVault contract
- Functions:
  - list(tokenId, price, asset)

    - buy(orderId)
    - cancel(orderId)
- Use ReentrancyGuard.
- Events for created/executed/cancelled orders with fees. Provide Solidity code with proper access control and checks.

1.5. CharityVault (кошелёк фонда)

Task: Implement **CharityVault** contract as the on-chain treasury for the Foundation.

Requirements:

- Accept ETH and ERC20 tokens.
- Track per-asset totals: mapping(address asset => uint256 totalReceived); mapping(address asset => uint256 totalWithdrawn);
- Role: FOUNDATION_MULTISIG_ROLE.
- donate functions:
    - donateERC20(asset, amount, uint8 sourceType, uint256 sourceId)
    - donateETH(uint8 sourceType, uint256 sourceId) payable
- withdraw:
    - withdraw(asset, amount, to, string memo) only FOUNDATION_MULTISIG_ROLE
- Events:
    - DonationReceived(asset, amount, sourceType, sourceId, from)
    - CharityWithdrawal(asset, amount, to, memo) Implement in Solidity with AccessControl, safe ERC20 operations.

1.6. DiscountCurve library (кривая скидок)

Task: Implement a pure **DiscountCurve** library for maintenance discounts.

Requirements:

- Input:
    - uint16 vipLevel (0..5)
    - uint16 prepayDays (0..365)
- Output:
    - uint16 discountBps
- Hardcode a simple curve, e.g.:

        - base discount by vipLevel (0..1500 bps)
        - additional discount up to a cap by prepayDays.
    - Provide:
        - function computeDiscount(vipLevel, prepayDays) external pure returns (uint16) Use as a library that maintenance-engine or contracts can call.

1.7. Solana / Anchor SBT для Академии

Task: Implement an Anchor program **tyt_academy_sbt** for non-transferable course certificates (SBT).

Requirements:

- For each course completion, mint one SBT to user.
- Each SBT stores:
    - course_id (u64)
    - level (u8)
    - issued_at (i64)
- Tokens must be non-transferable:
    - override transfer hooks or use a custom program constraint to reject transfers.
- Provide:
    - main.rs with instructions:
        - issue_certificate(user, course_id, level)
    - account structs
    - IDL-compatible definitions. Return full Rust/Anchor code skeleton with comments.

- Микро-промпты для backend-сервисов

2.1. auth-service базовый

Task: Implement NestJS **auth-service**.

Requirements:

- Endpoints:
    - POST /auth/register {email, password}
    - POST /auth/login {email, password}
    - POST /auth/refresh {refreshToken}

- JWT access + refresh tokens.
- Hash passwords with bcrypt.
- PostgreSQL schema:
  - users(id UUID, email text unique, password_hash text, status text, created_at timestamptz)
  - user_sessions(id UUID, user_id UUID, ua text, ip text, expires_at timestamptz) Provide:
- NestJS module structure
- DTOs, controllers, services
- Prisma or TypeORM models.

2.2. wallet-service с двойной записью

Task: Implement NestJS **wallet-service** with double-entry ledger.

Requirements:

- Entities:
  - wallet_accounts(id, user_id, asset, external_address, created_at)
  - ledger_entries(id, account_id, debit, credit, ref_type, ref_id, ts)
- API:
  - GET /wallet/accounts
  - GET /wallet/balance?asset=...
  - POST /wallet/withdraw
  - Internal: POST /wallet/internal-transfer, POST /wallet/apply-reward
- Implement a helper service:
  - createLedgerEntry(accountId, debit, credit, refType, refId)
  - getBalance(userId, asset) Return NestJS code skeleton, including example implementation of internal transfer.

2.3. blockchain-gateway-service каркас

Task: Implement **blockchain-gateway-service** (NestJS) to proxy blockchain interactions.

Requirements:

- Config support for:
  - POLYGON_RPC_URL

        SOLANA_RPC_URL
        BTC_PROVIDER_URL
    Modules:
        polygon-client (ethers.js)
        solana-client (@solana/web3.js + Anchor)
        btc-client (REST provider)
    API:
        POST /blockchain/send-tx {chain, method, params}
        POST /blockchain/register-deposit-address {asset, userId}
    Provide abstract client interfaces and one example method for each chain.
    Return NestJS structure with stub implementations and proper DI.

2.4. miner-registry-service sync с контрактом

Task: Implement **miner-registry-service** to mirror MinerNFT on-chain state.

Requirements:

- DB:
    - miners(id, nft_token_id, owner_id, power_th, eff_tier, region_id, status, reinvest_pct, created_at)
- Features:
    - A background job that subscribes to MinerNFT events via blockchain-gateway:
        - MinerMinted, MinerUpgraded, MinerStatusChanged, Transfer.
    - Update DB accordingly (owner, params, status).
- API:
    - GET /miners/my
    - GET /miners/:id
    - POST /miners/:id/reinvest-config {reinvest_pct, charity_pct} Return NestJS service, controller, and a job/scheduler to sync events.

2.5. maintenance-engine-service расчёт

Task: Implement **maintenance-engine-service**.

Requirements:

- DB:

    - fees(region_id, kwh_usd, service_bps, updated_at)
    - maintenance_invoices(id, miner_id, period_start, period_end, amount_usd, discount_pct, asset, status)
- API:
    - POST /maintenance/calculate Input: {power_th, efficiency_w_per_th, region_id, date_range, discount_profile{vip_level, has_tyt_discount, prepay_days}} Output: {elec_usd, service_usd, total_usd, discount_pct, curve_tier}
- Use DiscountCurve library logic in TS. Return NestJS controller + service + sample calculation.

2.6. rewards-engine-service дневной джоб

Task: Implement **rewards-engine-service** daily job flow.

Requirements:

- DB:
    - daily_pool(date, gross_btc, price_btc_usd)
    - rewards(date, miner_id, gross_btc, elec_usd, service_usd, discount_pct, net_btc, proof_leaf)
- Flow:
    - Cron job once per day:
        - Load active miners from miner-registry-service.
        - Get gross_btc pool for the day.
        - Compute share per miner by TH.
        - Call maintenance-engine /calculate per miner or batched.
        - Compute net_btc, write to rewards.
        - Call wallet-service /wallet/apply-reward.
        - Build Merkle tree leaves [minerId, net_btc, date], compute root.
        - Call blockchain-gateway to setRewardsRoot on RewardsMerkleRegistry. Provide NestJS code skeleton with pseudo-implementation for the cron job.

2.7. marketplace-service индексация

Task: Implement **marketplace-service** for MinerMarketplace indexing.

Requirements:

- DB:
  - orders(id, miner_id, seller_id, price, asset, status, created_at)
  - trade_events(id, order_id, buyer_id, amount, fee_protocol, fee_charity, ts)
- Features:
  - Subscribe to MinerMarketplace on-chain events via blockchain-gateway.
  - Update orders and trade_events.
- API:
  - GET /marketplace/orders (filters: region, power range, price range, asset)
  - GET /marketplace/orders/:id
  - GET /marketplace/my-trades Return NestJS controller + service + event-consumer.

2.8. governance-service (proposal & voting)

Task: Implement **governance-service**.

Requirements:

- DB:
  - proposals(id, title, description, param_key, status, created_at)
  - votes(id, proposal_id, user_id, voting_power, choice, created_at)
- Integration:
  - Read voting power from veTYT via blockchain-gateway.
- API:
  - GET /governance/proposals
  - GET /governance/proposals/:id
  - POST /governance/proposals (admin)
  - POST /governance/proposals/:id/vote {choice}
  - GET /governance/user-voting-power Return NestJS controller + service, focusing on computing total votes and winning choice.

2.9. rank-and-gamification-service

Task: Implement **rank-and-gamification-service**.

Requirements:

- DB:

    - user_rank_state(user_id, rank, rank_score, updated_at)
    - user_badges(id, user_id, badge_code, earned_at, source)
- RankScore formula:
    - function of: total TH (from miner-registry), veTYT power, academy progress, payment discipline.
- API:
    - GET /ranks/me
    - GET /ranks/leaderboard
- Background job:
    - Periodically recompute rank_score and rank for all active users. Return NestJS skeleton, including pseudo formula for RankScore.

2.10. academy-service

Task: Implement **academy-service**.

Requirements:

- DB:
    - courses(id, slug, title, level, meta_json)
    - lessons(id, course_id, order, slug, title)
    - user_course_progress(user_id, course_id, status, score)
    - user_quiz_attempts(user_id, lesson_id, result, ts)
- API:
    - GET /academy/courses
    - GET /academy/courses/:id
    - GET /academy/courses/:id/progress
    - POST /academy/courses/:id/complete-lesson
    - POST /academy/courses/:id/finish
        - On finish: call blockchain-gateway to issue SBT certificate. Return NestJS code structure and DTOs.

2.11. charity-service (детальный)

Task: Implement **charity-service** for tracking all foundation-related flows.

Requirements:

- DB:

        charity_flows(id, source_type, source_id, user_id, asset, amount, tx_hash, created_at)
        charity_reports(id, period_start, period_end, total_in_by_asset_json, total_out_by_asset_json, summary_markdown, external_links_json, created_at)
        charity_campaigns(id, title, description, target_amount_usd, collected_amount_usd, status, attachments_json, created_at)
        charity_allocations(id, report_id, campaign_id, asset, amount, note)
    API:
        internal:
            POST /charity/income {source_type, source_id, user_id?, asset, amount, tx_hash?}
            POST /charity/allocate {report_id, campaign_id, asset, amount, note}
        public:
            GET /charity/summary
            GET /charity/reports
            GET /charity/reports/:id
            GET /charity/campaigns
        admin:
            POST /charity/campaigns
            PATCH /charity/campaigns/:id
            POST /charity/withdraw (calls blockchain-gateway -> CharityVault.withdraw) Return NestJS controller + service stubs and SQL/Prisma schema definitions.

Микро-промпты для frontend

3.1. Landing Hero + How It Works

Task: Implement the landing page hero + “How it works” sections in Next.js (App Router) with Tailwind.

Requirements:

/app/(public)/page.tsx

Hero:

Title: “Own NFT Miners. Earn BTC Daily. Support Children’s Brain Cancer Research.”

Subtitle: short 2-line explanation.

    - Buttons: “Launch App” (link /app/dashboard), “Learn More” (scroll to how-it-works).
- How-it-works: 4 step cards with icons, describing the flow: Sign up → Buy NFT miners → Get BTC → Support Foundation. Use a dark theme with owl/knight shield visual hints and responsive layout.

3.2. Income Calculator компонент

Task: Implement a reusable **IncomeCalculator** React component.

Props:

- Optional defaults for:
    - defaultTH
    - defaultEfficiency
    - defaultBTCPrice
    - defaultKwhPrice

UI:

- Inputs: TH/s (slider + number), W/TH, BTC price, kWh price.
- Real-time calculated outputs:
    - estimated daily/weekly/monthly net BTC (use a placeholder formula).
- Export it and integrate into the landing page “Income calculator” section. Use TypeScript + Tailwind.

3.3. Dashboard Page

Task: Implement `/app/dashboard` page.

Requirements:

- Fetch:
    - user balances (BTC, TYT, total miner count)
    - estimated daily BTC rewards
    - unpaid maintenance invoices
    - rank info (rank name + score)
- Layout:
    - top cards for balances & daily reward
    - section with a small chart of rewards over last 30 days (mock data)

card showing current Owlverse rank and progress bar to next rank
four CTAs: Buy Miners / Pay Maintenance / Donate / Start Course Use React Query hooks (e.g. useDashboardData) and Tailwind.

3.4. Miners List + Detail

Task: Implement `/app/miners` list and `/app/miners/[id]` detail pages.

List page:

- Table with columns: Miner ID, Region, Power (TH/s), Efficiency, Status, Reinvest/Charity flags.
- Row click navigates to detail page.

Detail page:

- Show full miner parameters.
- Chart of rewards for this miner (mock).
- Maintenance breakdown (electricity + service + discounts).
- Controls:
  - Upgrade Power (button)
  - Upgrade Efficiency
  - Reinvest % slider
  - Donate % slider (for Foundation) Use mocked APIs with types aligned to backend models.

3.5. Rewards History

Task: Implement `/app/rewards` page.

Requirements:

- Table:
  - date, gross_btc, elec_usd, service_usd, discount_pct, net_btc
- Filters: date from/to.
- Stats cards:
  - lifetime total
  - last 30 days
  - last 7 days

Export button that calls backend endpoint to download CSV. Use React Query + Tailwind table components.

3.6. Wallet Page

Task: Implement `/app/wallet` page.

Requirements:

- Balances section:
  - grid of asset cards (BTC, USDT, USDC, TYT).
- Actions:
  - Deposit: show deposit address / QR (mock).
  - Withdraw: open modal with form {asset, amount, address}.
- History list:
  - type (deposit/reward/withdraw/donation/trade),
  - asset, amount, date, status. Implement as React components fetching from mock API hooks.

3.7. Marketplace List & Detail

Task: Implement `/app/marketplace` (list) and `/app/marketplace/[id]` (detail).

List:

- Cards or table with: miner preview (power, efficiency, region), seller rank, price & asset.
- Filters: min/max power, region, asset, price range.

Detail:

- Shows all miner params and estimated income metrics (placeholder).
- "Buy now" button which:
  - prepares signed tx data (mock) and shows JSON in modal (for MVP).
  - Integrate with a simple `useMarketplaceOrders` hook.

3.8. Academy UI

Task: Implement `/app/academy` section.

Requirements:

- /app/academy: list of courses with title, difficulty, estimated time, progress bar.
- /app/academy/[courseId]:
    - Lesson list sidebar.
    - Main area showing lesson content (Markdown/MDX rendered).
    - Simple quiz at bottom with multiple-choice questions (mock data).
- “Finish course” button:
    - Calls a fake API endpoint, then shows success and “SBT issued” badge.
    - Implement with TS & Tailwind; separate CourseCard, LessonSidebar, Quiz components.

3.9. Foundation public + in-app

Task: Implement `/foundation` (public) and `/app/foundation` (in-app) pages.

Public `/foundation`:

- Hero section describing the Brain Cancer Children’s Research & Support Foundation.
- Cards with:
    - total donated (mock numbers)
    - number of campaigns
- List of active campaigns with progress (title, goal, raised).
- “View latest report” button linking to a report detail section.

In-app `/app/foundation`:

- Same metrics + user stats:
    - total donated by this user (mock)
    - % of rewards donated.
- Toggles:
    - quick buttons for 1% / 3% / 5% donation of future rewards.
- Donation modal to send additional donation (mock). Use shared components for campaigns & reports.

3.10. Profile & Settings

Task: Implement `/app/profile` page.

Requirements:

- Sections:
  - Account: email, nickname (editable).
  - Localization: language, timezone.
  - Security: 2FA status, KYC status indicator.
  - Notifications: toggles for email/Telegram/push.
- Integrate with mock APIs `useProfile`, `updateProfile`, `useKycStatus`. Use Tailwind forms and show validation errors.

Микро-промпты для Infra

4.1. Monorepo skeleton

Task: Design a monorepo folder structure for TYT.

Requirements:

- Use pnpm + TurboRepo (or Nx).
- Folders:
  - /contracts (Hardhat or Foundry + Anchor for Solana)
  - /backend/services/<service-name>
  - /frontend/web
  - /shared (shared libs)
- Provide:
  - root package.json
  - turbo.json or nx.json
  - basic tsconfig setup
  - instructions to run dev for multiple services. Return the directory tree and key config files content.

4.2. docker-compose local dev

Task: Create a `docker-compose.yml` for local dev.

Requirements:

- Services:
  - postgres
  - redis

        - kafka (or rabbitmq)
        - auth-service
        - wallet-service
        - rewards-engine-service
        - charity-service
        - api-gateway (optional)
        - frontend-web
    - Define volumes for DB.
    - Add healthchecks.
    - Use environment variables via .env file. Return full docker-compose.yaml with comments.

4.3. Dockerfile шаблон для NestJS сервиса

Task: Provide a reusable Dockerfile template for NestJS microservices.

Requirements:

- Multi-stage build:
    - builder: node:20-alpine, install deps, build
    - runner: node:20-alpine, copy dist + prod deps
- Include:
    - NODE_ENV handling
    - non-root user Return the Dockerfile and explain how to parameterize it for different services.

4.4. GitHub Actions: backend CI

Task: Create a GitHub Actions workflow for backend CI.

Requirements:

- Trigger: push/PR on paths under /backend/**.
- Steps:
    - checkout
    - setup pnpm & Node
    - install dependencies
    - run tests (e.g. pnpm test)
    - run lint

        - build
    - On main branch: build Docker images for services and push to registry. Return a full yaml workflow file `backend-ci.yml`.

4.5. K8s манифест для auth-service

Task: Provide Kubernetes manifests for **auth-service**.

Requirements:

- Deployment with:
    - image placeholder
    - env from ConfigMap + Secret
    - liveness/readiness probes
- Service ClusterIP.
- Example ConfigMap and Secret definitions. Return YAML with comments.

4.6. Observability stack (Prometheus + Grafana + Loki)

Task: Design minimal observability setup.

Requirements:

- Explain:
    - how services will expose Prometheus metrics (/metrics).
    - how logs will be exported in JSON to Loki.
- Provide:
    - example Prometheus scrape config for auth-service and rewards-engine-service
    - example Grafana dashboard outline for:
        - rewards job success/failure
        - charity flows per day Return config snippets and explanation, no need for full Helm charts.

полный обновлённый пакет всех блоков, промптов и микро-промптов **в правильной очередности**, чтобы ты мог просто копировать подряд в bolt.new / v0 / Lovable **без потерь и без расхождений**.

Мы полностью обновили модель комиссий:

✔ 60% protocol

✔ 30% charity (детский фонд рака мозга)

✔ 10% academy

И теперь я собираю всё **с нуля, в чистом виде**, без старых значений.

📦 TYT — FULL UPDATED PROMPT PACK (v6/3/1)

(Готов для передачи bolt.new / v0 / Lovable / Replit Agents)

🔥 БЛОК 0 — Глобальная модель комиссий (вставлять в начало всех агентов)

```
GLOBAL FEE MODEL (TYT)

Basis points:
  1%   = 100 bps
  0.1% = 10 bps
  0.01% = 1 bps

Default deposit fee:
  deposit.stables.fee_bps_total = 100    // 1.00%

Fee split inside fee_total:
  protocol_pct = 60      // 60%
  charity_pct  = 30      // 30%
  academy_pct  = 10      // 10%

Total must always = 100.

Fee formula:
  fee_total    = amount * fee_bps_total / 10_000
  amount_user = amount - fee_total

Split:
  fee_protocol = fee_total * protocol_pct / 100
  fee_charity  = fee_total * charity_pct  / 100
  fee_academy  = fee_total * academy_pct  / 100

Validation:
  - fee_bps_total <= 200
  - protocol_pct + charity_pct + academy_pct == 100
  - No component may bypass this split.

Governance:
  All values are controlled by veTYT governance.
```

🟦 БЛОК 1 — CONTRACTS (Solidity & Anchor)

Все обновлённые микро-промпты подряд.

1.1 MinerNFT — ядро майнеров

```
Task: Implement MinerNFT (Polygon, Solidity ^0.8.20).

Use:
- ERC721 OpenZeppelin
- AccessControl

struct Miner {
  uint96 powerTH;
  uint96 efficiencyWPerTH;
  uint16 regionId;
  uint16 maintenanceRateBps;
  uint8  status; // 0 Active, 1 Delinquent, 2 Locked
}

Roles:
  DEFAULT_ADMIN_ROLE
  MINTER_ROLE
  UPGRADER_ROLE
  OPERATOR_ROLE

Required functions:
  mintMiner(address to, Miner m) only MINTER_ROLE
  upgradeHashrate(tokenId, newPowerTH) only UPGRADER_ROLE
  upgradeEfficiency(tokenId, newEff) only UPGRADER_ROLE
  setStatus(tokenId, newStatus) only OPERATOR_ROLE

Emit events:
  MinerMinted, MinerUpgraded, MinerStatusChanged

Provide full Solidity code + IMinerNFT interface.
```

1.2 RewardsMerkleRegistry

```
Task: Implement RewardsMerkleRegistry.

Store daily Merkle roots:
  mapping(uint256 day => bytes32 merkleRoot)

Roles:
  REWARDS_ORACLE_ROLE

Functions:
  setRewardsRoot(day, root):
      - only oracle
      - cannot overwrite existing root
  getRewardsRoot(day)

Event:
  RewardsRootSet(day, root)

Use NatSpec comments.
```

1.3 veTYT (локи + голосование)

```
Task: Implement veTYT time-lock contract.

Lock = {
  amount,
  start,
  end
}

Functions:
  createLock(amount, duration) -> lockId
  increaseAmount(lockId, added)
  increaseDuration(lockId, added)
  withdraw(lockId) after end

Voting power = linear with duration.
getVotingPower(user) = sum over active locks.
Emit events.

Governance-service will call getVotingPower(...) for voting.
```

1.4 MinerMarketplace (учёт 60/30/10)

```
Task: Implement MinerMarketplace with fee split via FeeConfig.

Order:
  seller
  tokenId
  price
  asset
  active

Fee model:
  FeeProfile fp = feeConfig.get("marketplace.primary")

  feeTotal    = price * fp.feeBpsTotal / 10_000
  amountToSeller = price - feeTotal

  feeProtocol = feeTotal * fp.protocolPct / 100   // 60%
  feeCharity  = feeTotal * fp.charityPct / 100    // 30%
  feeAcademy  = feeTotal * fp.academyPct / 100    // 10%

Transfers:
  - seller gets amountToSeller
  - protocolFeeRecipient gets feeProtocol
  - charityVault gets feeCharity
  - academyVault gets feeAcademy

Event OrderExecuted:
  (..., feeTotal, feeProtocol, feeCharity, feeAcademy)
```

1.5 CharityVault (расщепление fee_charity + fee_academy)

```
Task: Implement CharityVault (ETH + ERC20).

Store:
  totalReceived[asset]
  totalWithdrawn[asset]

Role:
  FOUNDATION_MULTISIG_ROLE

donateERC20(asset, amount, sourceType, sourceId)
donateETH(sourceType, sourceId)

Withdraw:
  withdraw(asset, amount, to, memo) only multisig

sourceType must include:
  1 USER_DIRECT
  2 REWARDS_PERCENT
  3 MARKETPLACE_FEE_CHARITY
  4 DEPOSIT_FEE_CHARITY
  5 MARKETPLACE_FEE_ACADEMY
  6 DEPOSIT_FEE_ACADEMY

Emit events for all donations & withdrawals.
```

1.6 FeeConfig (новый контракт — основа split 60/30/10)

```
Task: Implement FeeConfig.

FeeProfile {
  uint16 feeBpsTotal;    // 1% = 100
  uint8  protocolPct;    // default 60
  uint8  charityPct;     // default 30
  uint8  academyPct;     // default 10
}
```

```
mapping(bytes32 => FeeProfile) profiles;

Roles:
  DEFAULT_ADMIN_ROLE
  GOVERNANCE_ROLE

setFeeProfile(key, totalBps, p, c, a):
  require(totalBps <= 200)
  require(p + c + a == 100)
  save profile
  emit FeeProfileUpdated(key,...)

Use keys:
  keccak256("deposit.stables")
  keccak256("deposit.usdt")
  keccak256("marketplace.primary")
  keccak256("marketplace.secondary")
```

1.7 Solana Anchor SBT (Академия)

```
Task: Anchor program tyt_academy_sbt.

SBT:
  course_id: u64
  level: u8
  issued_at: i64

Instruction:
  issue_certificate(user, course_id, level)

Non-transferable:
  reject transfer attempts via custom constraint.

Provide IDL-compatible code skeleton.
```

🟩 БЛОК 2 — BACKEND MICRO-SERVICES (NestJS)

2.1 auth-service

```
Task: NestJS auth-service.

Endpoints:
  POST /auth/register
  POST /auth/login
  POST /auth/refresh
```

```
DB:
  users(id, email, password_hash, status)
  sessions(id, user_id, ua, ip, expires_at)

Use bcrypt + JWT + refresh tokens.

Provide full module/controller/service skeleton.
```

2.2 wallet-service (обновлено под 60/30/10)

```
Task: wallet-service with double-entry ledger.

Deposit flow must use FeeConfig:

fee_total   = amount * fee_bps_total / 10_000
amount_user = amount - fee_total

fee_protocol = fee_total * 0.60
fee_charity  = fee_total * 0.30
fee_academy  = fee_total * 0.10

Ledger entries:
  credit user: amount_user
  credit protocol_revenue: fee_protocol
  credit charity_fund:     fee_charity
  credit academy_fund:     fee_academy

Notify charity-service:
  POST /charity/income (charity)
  POST /charity/income (academy)

Provide NestJS code skeleton.
```

2.3 blockchain-gateway-service

```
Task: NestJS gateway for Polygon / Solana / BTC.

API:
  POST /blockchain/send-tx
  POST /blockchain/register-deposit-address

Implement polygon-client, solana-client, btc-client stubs.
Use DI.
```

2.4 miner-registry-service

```
Task: Sync MinerNFT on-chain to DB.

DB:
  miners(id, tokenId, ownerId, powerTH, effTier, regionId, status,
reinvest_pct)

Subscribe to events:
  MinerMinted
  MinerUpgraded
  MinerStatusChanged
  Transfer

API:
  GET /miners/my
  GET /miners/:id
  POST /miners/:id/reinvest-config
```

2.5 maintenance-engine-service

```
Task: maintenance calculation.

Input:
  {powerTH, efficiencyWPerTH, region_id, date_range, discount_profile}

Output:
  {elec_usd, service_usd, total_usd, discount_pct}

Use DiscountCurve and service_bps from DB.

Provide example calculation.
```

2.6 rewards-engine-service

```
Task: Cron job (daily).

Steps:
  1) Load all active miners
  2) Compute gross BTC share
  3) Call maintenance-engine
  4) Compute net BTC
  5) Write rewards table
  6) Apply wallet-service /apply-reward
  7) Build Merkle tree → setRewardsRoot

NestJS skeleton required.
```

2.7 marketplace-service (учёт 60/30/10)

```
Task: index on-chain marketplace events.

DB:
  orders(...)
  trade_events(..., fee_protocol, fee_charity, fee_academy)

On OrderExecuted event:
  store:
    fee_protocol = 60% of fee_total
    fee_charity  = 30%
    fee_academy  = 10%

API:
  GET /marketplace/orders
  GET /marketplace/orders/:id
  GET /marketplace/my-trades
```

2.8 governance-service

```
Task: veTYT voting on fee parameters.

param_key examples:
  "deposit.stables.fee_bps_total"
  "deposit.stables.fee_protocol_pct"
  "deposit.stables.fee_charity_pct"
  "deposit.stables.fee_academy_pct"

proposals(...)
votes(...)

Winning proposal must update FeeConfig.
```

2.9 rank-and-gamification-service

```
Ranks depend on:
  - total TH
  - veTYT
  - academy progress
  - payment discipline

Recompute periodically.
Expose:
  GET /ranks/me
  GET /ranks/leaderboard
```

2.10 academy-service

```
Task: course system.

DB:
  courses, lessons, progress, attempts

Issue SBT on course finish via blockchain-gateway.

Provide NestJS structure.
```

2.11 charity-service (учёт charity + academy потоков)

```
Task: Track all flows.

DB:
  charity_flows(id, source_type, source_id, user_id, asset, amount)
  charity_reports(...)
  charity_campaigns(...)
  charity_allocations(...)

source_type must support:
  - DEPOSIT_FEE_CHARITY
  - DEPOSIT_FEE_ACADEMY
  - MARKETPLACE_FEE_CHARITY
  - MARKETPLACE_FEE_ACADEMY
  - USER_DIRECT
  - REWARDS_PERCENT

API:
  /charity/income
  /charity/summary
  /charity/reports
  /charity/campaigns
```

🟧 БЛОК 3 — FRONTEND (Next.js)

3.1 Landing

```
Hero:
  "Own NFT miners. Earn BTC daily. Support children's brain cancer
research."

How-it-works:
  1 Sign up
  2 Buy NFT miners
  3 Get BTC
  4 Support Foundation
```

```
Include fee model tooltips referencing 60/30/10.
```

3.2 IncomeCalculator

```
Inputs:
  TH/s, W/TH, BTC price, kWh
Outputs:
  daily/weekly/monthly net BTC

React + Tailwind.
```

3.3 Dashboard

```
Cards:
  balances, daily BTC, unpaid maintenance, rank

Buttons:
  Buy Miners / Pay Maintenance / Donate / Start Course
```

3.4 Miners List + Detail

```
Show:
  powerTH, efficiency, region, reinvest%, donation%

Show maintenance breakdown + daily income.
```

3.5 Rewards History

```
Table:
  date, gross, elec, service, discount, net
Filters:
  date range
```

3.6 Wallet Page (обновлено под 60/30/10)

```
Deposit flow:
  call /fees/calc-deposit
  show:
    - fee_total
    - amount_user

Breakdown:
  Protocol (60%) = fee_protocol
  Charity (30%)  = fee_charity
  Academy (10%)  = fee_academy
```

3.7 Marketplace

List + Detail.
Show estimated income + fee breakdown on purchase.

3.8 Academy

Course list, lesson pages, quizzes, "Finish" button triggers SBT.

3.9 Foundation

Public & in-app pages show:

- Total donated
- Fee-based charity (30%)
- Fee-based academy (10%)
- User donations
- Campaigns + reports

3.10 Profile

Account settings
KYC
Notifications
Language & timezone

🟥 БЛОК 4 — INFRA

4.1 Monorepo

```
/contracts
/backend/services/...
/frontend/web
/shared
pnpm + turborepo
```

4.2 docker-compose

```
Services:
  postgres, redis, kafka
  auth, wallet, rewards, charity
  frontend-web

Use env + healthchecks.
```

4.3 Dockerfile (NestJS)

```
Multi-stage build:
  builder (npm install + build)
  runner (copy dist)
user node
```

4.4 GitHub Actions — backend CI

```
Install deps, test, lint, build.
On main: build docker images and push.
```

4.5 K8s manifest

```
Deployment + Service
ConfigMap + Secrets
Liveness/Readiness probes
```

4.6 Observability

```
Prometheus:
  scrape /metrics for:
    auth-service
    rewards-engine-service

Loki:
  JSON logs

Grafana dashboards:
  - rewards job success/failure
  - charity flows per day
```

🔄 PHASE 2 IN PROGRESS (Advanced Features)

Priority 1: Rewards Engine (CRITICAL) 🚨
Status: Schema ready, implementation needed Impact: Core product functionality

Tasks:

Implement daily BTC reward calculation
Gross BTC formula (TH/s × Network difficulty)
Electricity cost calculation (W/TH × region rates)
Service fee application
Discount curve implementation (Bronze → Diamond)
Service Button mechanic (daily -3%)
Auto-reinvest logic
Merkle proof generation
Reward claim API endpoints
Maintenance payment processing

Estimated Time: 2 weeks Priority: ⭐⭐⭐⭐⭐ (Highest)

Priority 2: Marketplace Functionality 🏪
Status: UI ready, backend needed Impact: Secondary market liquidity

Tasks:

NFT miner listing API
Buy/sell transaction flow
Escrow system implementation
Price discovery mechanism
Royalty distribution (5% to creator)
Search & filter backend
Auction system (optional)
Transfer ownership logic
Marketplace fee collection (2% in TYT)
Activity feed

Estimated Time: 2 weeks Priority: ⭐⭐⭐⭐ (High)

Priority 3: Smart Contracts (EVM) 📜
Status: Not started Impact: Decentralization & trust

Tasks:

MinerNFT ERC-721 contract
Metadata structure (TH/s, efficiency, region)
Marketplace escrow contract
veTYT governance token contract
Time-weighted voting logic
FundSplitter contract (charity distribution)
BurnScheduler contract
Contract testing (Hardhat/Foundry)
Audit preparation
Deployment to Polygon/BSC

Estimated Time: 3-4 weeks Priority: ⭐⭐⭐⭐ (High)

Priority 4: Admin Panel 👨‍💼
Status: Not started Impact: Operational efficiency

Tasks:

KYC document review interface
User management dashboard
Tier upgrade controls
NFT miner creation tool
Reward distribution controls
Fee configuration panel
Analytics dashboard
Transaction monitoring
Support ticket system
Foundation grant approval flow

Estimated Time: 2 weeks Priority: ⭐⭐⭐ (Medium-High)

Priority 5: Staking Implementation 🔒
Status: UI ready, backend needed Impact: Token utility & TVL

Tasks:

TYT staking pools (30/90/180/365 days)
Lock period enforcement
APY calculation engine
Daily reward distribution
veTYT conversion logic
Early unstaking penalties
Staking statistics
Reward claim mechanism
Pool capacity management
Compound interest option

Estimated Time: 1-2 weeks Priority: ⭐⭐⭐ (Medium)

📋 PHASE 3 PLANNED (Expansion)

Mobile Applications 📱
React Native setup
Unified codebase (iOS/Android)
Push notifications (Firebase)
Biometric authentication
Deep linking
App Store / Play Store deployment

Estimated Time: 4-6 weeks Priority: ⭐⭐ (Medium)

Governance (DAO) 🗳️
veTYT staking for governance power
Proposal creation system
Voting mechanism
Quorum requirements
Timelock for execution
Governance rewards
Parameter voting (fees, discount curves, etc.)

Estimated Time: 3-4 weeks Priority: ⭐⭐ (Medium)

Academy Content 🎓
Course curriculum design
Video production
Interactive quizzes
Soulbound NFT certificates

Owl Warrior rank progression
Gamification mechanics
Progress tracking
Community forums

Estimated Time: 6-8 weeks Priority: ⭐⭐ (Medium)

Foundation Portal ❤️

Grant application system
Clinic partnership onboarding
Transparent fund tracking
Impact reporting dashboard
Donation widget
Charity staking pools
Monthly/annual reports
Beneficiary stories

Estimated Time: 3-4 weeks Priority: ⭐⭐ (Medium)

# TYT v2 - Security & Deployment Strategy

## 🎯 Цель
Защитить критические компоненты экосистемы от несанкционированного доступа, сохранив открытость для легитимных пользователей.

## 📋 Архитектура Безопасности

### Уровень 1: Public (Open Source)
Репозиторий: `github.com/takeyourtokenapp/tyt.app` (Public)

✅ **Включает:**
- Frontend код (React/TypeScript)
- UI компоненты
- Публичные типы и интерфейсы
- Документация для пользователей
- Contribution guidelines

❌ **НЕ включает:**
- `.env` файлы
- Приватные ключи
- API секреты
- Deployment конфигурации
- Admin скрипты

### Уровень 2: Private Infrastructure
Репозиторий: `github.com/takeyourtokenapp/tyt-infrastructure` (Private)

✅ **Включает:**
- Supabase миграции (уже защищены RLS)
- Edge Functions
- Deployment scripts
- CI/CD конфигурации
- Monitoring setup
- Backup strategies

### Уровень 3: Blockchain Smart Contracts

Репозиторий: `github.com/takeyourtokenapp/tyt-contracts` (Private → Public после аудита)

✅ **Включает:**
- Solidity/Rust контракты
- Тесты
- Deployment scripts (без ключей)
- Audit reports

❌ **НЕ включает:**
- Private keys
- Mnemonic phrases
- Admin wallet addresses (до launch)

## 🔐 Защита Критических Компонентов

### 1. Supabase Security

Row Level Security (RLS) - УЖЕ РЕАЛИЗОВАНО:
```sql
-- Пример из миграций:
CREATE POLICY "Users can only view own data"
  ON users FOR SELECT
  TO authenticated
  USING (auth.uid() = id);
```

Защищено:
- ✅ Users могут видеть только свои данные
- ✅ Miners защищены ownership checks
- ✅ Transactions protected
- ✅ Foundation funds read-only для публики

Edge Functions:
- ✅ JWT verification
- ✅ Rate limiting
- ✅ Input validation
- ✅ Error handling без утечки данных

### 2. Blockchain Security

Smart Contracts:
```solidity
// Защита admin функций
modifier onlyOwner() {
    require(msg.sender == owner, "Not authorized");
    _;
}

// Pause mechanism
modifier whenNotPaused() {
    require(!paused, "Contract paused");
    _;
}

// Reentrancy protection
modifier nonReentrant() {
    require(!locked, "No reentrancy");
    locked = true;
    _;
    locked = false;
}
```

Защита:
- ✅ Multi-sig wallets для критических операций
- ✅ Timelock для governance
- ✅ Circuit breakers
- ✅ Upgrade patterns (proxy)

### 3. API Keys & Secrets

Хранение:
```bash
# Локально (НЕ коммитится)
.env

# Production (зашифровано)
Vercel Environment Variables
GitHub Secrets (для CI/CD)
Supabase Vault
```

Ротация:
- API keys: каждые 90 дней
- JWT secrets: каждые 180 дней
- Admin keys: после каждого использования

## 🌐 Публикация в Сеть

### Phase 1: Private Beta (2-4 недели)

Доступ:
- Закрытая группа тестеров (50-100 человек)
- Whitelist адресов
- Invite-only

Deployment:
```bash
# Vercel Preview
vercel --prod --scope takeyourtokenapp

# Supabase Production
supabase db push
supabase functions deploy --project-ref <ref>
```

Monitoring:
- Sentry для ошибок

- Mixpanel для аналитики
- Custom alerts

### Phase 2: Public Beta (1-2 месяца)

Доступ:
- Открыт для всех
- KYC required для выводов >$1000
- Rate limiting

Security Measures:
- WAF (Web Application Firewall)
- DDoS protection (Cloudflare)
- Bot detection
- Suspicious activity alerts

### Phase 3: Full Launch

Доступ:
- Полностью публичный
- Multi-chain support
- Decentralized governance

Security Measures:
- Bug bounty program
- Continuous audits
- Incident response plan
- Insurance coverage

## 🛡️ Защита от Конкретных Угроз

### 1. Злоумышленники (Hackers)

Frontend:
- ✅ Input validation
- ✅ XSS protection
- ✅ CSRF tokens
- ✅ Content Security Policy

Backend:
- ✅ SQL injection protection (Supabase RLS)
- ✅ Rate limiting
- ✅ IP whitelisting для admin
- ✅ 2FA для критических операций

Smart Contracts:
- ✅ Professional audit (CertiK/OpenZeppelin)
- ✅ Bug bounty ($50k+)
- ✅ Formal verification
- ✅ Time-delayed upgrades

### 2. Скрейперы (Data Scrapers)

Protection:
```typescript
// Rate limiting
app.use(rateLimit({
  windowMs: 15 * 60 * 1000, // 15 min
  max: 100 // requests
}));

// Bot detection
if (req.headers['user-agent'].includes('bot')) {
  return res.status(403).json({ error: 'Forbidden' });
}
```

Supabase:
- RLS блокирует mass queries
- Pagination limits
- Query timeouts

### 3. Фишеры (Phishing)

Domain Security:
- ✅ HTTPS only
- ✅ HSTS headers
- ✅ CAA records
- ✅ Verified socials

User Education:
- ✅ Official domains list
- ✅ Wallet verification
- ✅ Phishing warnings
- ✅ Community moderation

### 4. Инсайдеры (Insider Threats)

Access Control:
- ✅ Principle of least privilege
- ✅ Audit logs (все действия)
- ✅ Multi-sig для критических операций
- ✅ Code review required

Monitoring:
- ✅ Unusual access patterns
- ✅ Large fund movements
- ✅ Contract parameter changes
- ✅ Admin wallet activity

## 📊 Что Видят Разные Пользователи

### Regular Users (Public)

Видят:
- ✅ Свои miners
- ✅ Свои rewards
- ✅ Свои transactions
- ✅ Marketplace listings
- ✅ Foundation transparency

НЕ видят:
- ❌ Данные других пользователей
- ❌ Internal balances
- ❌ Admin operations
- ❌ System architecture
- ❌ API endpoints структуру

### VIP Users

Дополнительно видят:
- ✅ Advanced analytics
- ✅ Priority support
- ✅ Beta features
- ✅ Governance proposals

### Admins (Private)

Видят:
- ✅ System metrics
- ✅ User statistics (aggregated)
- ✅ Financial reports
- ✅ Suspicious activity
- ✅ System health

Требования:
- 🔐 2FA mandatory
- 🔐 IP whitelist
- 🔐 Hardware key (YubiKey)
- 🔐 Audit trail

## 🚀 Deployment Checklist

### Pre-Launch Security Audit

- [ ] Smart contracts audited (2+ firms)
- [ ] Penetration testing
- [ ] Load testing
- [ ] Security review (OWASP Top 10)
- [ ] Legal compliance check
- [ ] Insurance coverage
- [ ] Incident response plan
- [ ] Bug bounty program ready

### Infrastructure

- [ ] CDN configured (Cloudflare)
- [ ] WAF rules active
- [ ] DDoS protection
- [ ] Backup strategy tested
- [ ] Disaster recovery plan
- [ ] Monitoring alerts
- [ ] Logging infrastructure
- [ ] Secrets management

### Compliance

- [ ] Privacy policy
- [ ] Terms of service
- [ ] KYC/AML procedures
- [ ] GDPR compliance (if EU users)
- [ ] Cookie consent
- [ ] Data retention policy
- [ ] Right to erasure procedure

## 🔍 Continuous Monitoring

### Automated Alerts

Critical (Immediate):
- Large fund movements
- Contract pause triggered
- Database breach attempt
- Admin access from new IP
- Unusual withdrawal patterns

High (1 hour):
- Failed login spikes
- API rate limit hits
- Error rate increase
- Slow query alerts

Medium (24 hours):
- Daily metrics summary
- User growth report
- Revenue report
- System health

### Manual Reviews

Daily:
- User reports
- Suspicious transactions

- Error logs

Weekly:
- Security scan results
- Dependency updates
- Access logs review

Monthly:
- Full security audit
- Compliance review
- Disaster recovery drill
- Team access review

## 📝 Incident Response Plan

### Phase 1: Detection (0-15 min)
1. Alert triggered
2. On-call engineer notified
3. Initial assessment

### Phase 2: Containment (15-60 min)
1. Identify scope
2. Activate circuit breakers if needed
3. Pause affected contracts
4. Block malicious IPs

### Phase 3: Investigation (1-4 hours)
1. Root cause analysis
2. Impact assessment
3. Evidence collection
4. Notify affected users

### Phase 4: Recovery (4-24 hours)
1. Fix vulnerability
2. Restore service
3. Verify integrity
4. Resume operations

### Phase 5: Post-Mortem (1-3 days)
1. Full report
2. Compensation plan (if needed)
3. Preventive measures
4. Update procedures

## 💡 Best Practices

### Development

```bash
# Никогда не коммитить
.env
.env.*
*.key
*.pem
secrets/
```

```
# Всегда review
Pull requests (2+ approvers)
Dependency updates
Config changes
```

### Deployment

```bash
# Staging сначала
npm run build
npm run test:e2e
vercel deploy --prod

# Потом production
git tag v2.x.x
npm run deploy:production
```

### Operations

- ✅ Principle of least privilege
- ✅ Change management process
- ✅ Rollback procedures ready
- ✅ Communication plan (users/team)

## 🎯 Success Metrics

### Security KPIs

- 0 critical vulnerabilities
- <0.01% fraud rate
- 99.9% uptime
- <1s response time p95
- 100% audit coverage

### User Trust

- Security badge visible
- Audit reports published
- Transparent operations
- Active bug bounty
- Responsive support

## 📞 Emergency Contacts

```
Security Lead: [ENCRYPTED]
Infrastructure: [ENCRYPTED]
Legal: [ENCRYPTED]
Insurance: [ENCRYPTED]
```

Public: security@takeyourtoken.app
```

---

## 🔗 Resources

- [OWASP Top 10](https://owasp.org/Top10/)
- [Smart Contract Security](https://consensys.github.io/smart-contract-best-practices/)
- [Supabase Security](https://supabase.com/docs/guides/auth)
- [Web3 Security Tools](https://github.com/Consensys/smart-contract-best-practices)

---

**Last Updated:** 2025-12-10
**Version:** 2.0.0
**Status:** Production Ready ✅

# TYT v2 — МАСТЕР-СПЕЦИФИКАЦИЯ
## Take Your Token v2 — Полная техническая документация

**Версия:** 2.0
**Дата:** Декабрь 2024
**Статус:** Phase 0 → Phase 1 (MVP Development)

---

## 📋 СОДЕРЖАНИЕ

---

<a name="миссия"></a>
## 1. МИССИЯ И КОНЦЕПЦИЯ

### Что такое TYT?

**Take Your Token (TYT)** — Web3-платформа нового поколения, объединяющая:

1. **NFT-майнинг** — цифровые майнеры с реальными BTC-вознаграждениями
2. **Token Economy** — TYT токен с burn-механикой и veTYT governance
3. **Blockchain Academy** — образовательная платформа с сертификатами (Soulbound NFTs)
4. **Children's Brain Cancer Foundation** — медицинский фонд, финансируемый через Web3

### Уникальное Ценностное Предложение (UVP)
```

> **TYT — первый в мире mining-NFT проект, где майнинг → медицина → детские жизни.**

Каждая транзакция в экосистеме автоматически поддерживает исследования опухолей мозга у детей.

### Три Столпа TYT

```
┌──────────────────────────────────────────────────────────────────────
──────────────────────────────────────┐
│                 TYT ECOSYSTEM                    │
├──────────────────────────────────────────────────────────────────────
──────────────────────────────────────┤
│                                   │
│  ┌──────────────────────────────┐  ┌──────────────────────────────┐
┌─────────────────────────┐  │
│  │  MINING & TOKEN  │  │  DIGITAL ACADEMY  │  │  FOUNDATION  │  │
│  │    ECONOMY     │  │               │  │          │  │
│  ├──────────────────────────────┤  ├──────────────────────────────┤
├─────────────────────────┤  │
│  │ • NFT Miners    │  │ • Web3 Education │  │ • Research   │  │
│  │ • BTC Rewards    │  │ • Certifications │  │ • Grants    │  │
│  │ • Marketplace    │  │ • Owl Ranks     │  │ • Support    │  │
│  │ • Governance     │  │ • Gamification   │  │ • Hospitals   │  │
│  │ • Burn/Mint     │  │ • Quests       │  │ • Reports   │  │
│  └──────────────────────────────┘  └──────────────────────────────┘
└─────────────────────────┘  │
│                                   │
│       1% от КАЖДОЙ транзакции → Foundation              │
└──────────────────────────────────────────────────────────────────────
──────────────────────────────────────┘
```

<a name="gomining-parity"></a>
## 2. FEATURE-ПАРИТЕТ С GOMINING

### Что повторяем из GoMining

Модуль	Описание	Статус
Digital Miners (NFT)	Токенизированные майнеры с TH/s и W/TH параметрами	✅
LBH Protocol	Liquid Bitcoin Hashrate — конвертация физического хешрейта в on-chain NFT	🔄
Daily BTC Rewards	Ежедневные начисления в кастодиальный кошелёк	✅
Maintenance System	Плата за электроэнергию (kWh × W/TH × TH/s) + сервис	✅
Token Discounts	Оплата TYT даёт скидку до -20%	✅
Service Button	Дополнительная скидка -3% ежедневно	✅
VIP Program	11 уровней (0-10), на основе hashrate или voting power	✅
Marketplace	P2P торговля NFT-майнерами (валюта: TYT)	✅
Miner Upgrades	20 уровней мощности (100 TH/s → 5000 TH/s max)	✅
Efficiency Upgrades	Улучшение W/TH (5-20% improvement)	✅

| **Miner Wars** | Clan battles за BTC/TYT призы | ✅ |

| **Avatars (GoMiners)** | Бонусные NFT с игровыми бустами | ✅ |

| **Burn & Mint Cycle** | Еженедельное сжигание токенов (вторник 12:00 UTC) | ✅ |

| **veToken Locks** | Блокировка токенов на 1 неделя → 4 года для governance | ✅ |

| **Live Streams** | 24/7 видео с дата-центров (США: WA/SC/TX) | 📋 |

| **Referral Program** | 5-5-5 модель (5% за покупки, апгрейды, игровые бусты) | ✅ |

| **Multi-chain Wallet** | BTC, Lightning, Liquid, wBTC, USDT, TRX, SOL, TON, XRP | ✅ |

| **Mobile Apps** | iOS/Android с паритетом веб-функционала | 📋 |

Легенда: ✅ Реализовано | 🔄 В разработке | 📋 Запланировано

<a name="architecture"></a>
## 3. АРХИТЕКТУРА СИСТЕМЫ

### High-Level Architecture

```
┌──────────────────────────────────────────────────────────┐
│                    CLIENT LAYER                           │
├─────────────────────┬──────────────────┬──────────────────┤
│  Web (Next.js)  │  iOS (Native)   │  Android (Native)       │
│  • SSR/ISR      │  • Swift/SwiftUI │  • Kotlin/Jetpack       │
│  • React Query  │  • Push Notify  │  • Deep Links           │
│  • Web3Modal    │  • Biometrics   │  • In-App Purchase      │
└─────────────────────┴──────────────────┴──────────────────┘
                    ↓ HTTPS/WSS
┌──────────────────────────────────────────────────────────┐
│                    API GATEWAY                            │
│              (Kong / AWS API Gateway)                     │
│  • Rate Limiting  • Auth Middleware  • Request Logging    │
└──────────────────────────────────────────────────────────┘
                    ↓
┌──────────────────────────────────────────────────────────┐
│              BACKEND SERVICES (NestJS)                    │
├─────────────────────┬──────────────────┬──────────────────┤
│  Auth Service    │  Miner Service   │  Rewards Engine       │
│  • Login/Signup  │  • NFT Mgmt      │  • Daily BTC Calc     │
│  • KYC/AML       │  • Upgrades      │  • Merkle Proofs      │
│  • Sessions      │  • Marketplace   │  • Distribution       │
├─────────────────────┼──────────────────┼──────────────────┤
│  Payment Service │  Governance Svc  │  Foundation Service   │
│  • Deposits      │  • veTYT Locks   │  • Donations          │
│  • Withdrawals    │  • Proposals     │  • Grants              │
│  • On-Ramp        │  • Voting       │  • Reports             │
└────────────────────────────────────┴──────────────────────────────────┴────────
───────────────────────────────────────────────┘
                  ↓
┌──────────────────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────┐
│              DATA & STORAGE LAYER                          │
├────────────────────────────────────┬──────────────────────────────────┬────────
───────────────────────────────────────────────┤
│  PostgreSQL       │  Redis           │  IPFS/S3                │
│  (Supabase)       │  (Cache/Queue)   │  (Media/NFT Metadata)     │
│  • Users          │  • Sessions       │  • Images               │
│  • Miners         │  • Rates          │  • Certificates          │
│  • Transactions   │  • Leaderboards   │  • Documents              │
└────────────────────────────────────┴──────────────────────────────────┴────────
───────────────────────────────────────────────┘
                  ↓
┌──────────────────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────┐
│              BLOCKCHAIN LAYER                           │
├────────────────────────────────────┬──────────────────────────────────┬────────
───────────────────────────────────────────────┤
│  TRON (Primary)   │  Solana          │  EVM (Bridges)            │
│  • TYT (TRC-20)   │  • TYT (SPL)      │  • Polygon               │
│  • Miners (721)   │  • Charity Mint  │  • BSC                  │
│  • Marketplace    │  • Compressed     │  • Optimism               │
│  • veTYT Locks    │    NFTs          │                         │
└────────────────────────────────────┴──────────────────────────────────┴────────
───────────────────────────────────────────────┘
                  ↓
┌──────────────────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────┐
│              EXTERNAL INTEGRATIONS                        │
├────────────────────────────────────┬──────────────────────────────────┬────────
───────────────────────────────────────────────┤
│  Payment Gateways │  Price Oracles    │  KYC/AML Providers        │
│  • Stripe         │  • Chainlink      │  • Sumsub                │
│  • Ramp Network   │  • Pyth           │  • Onfido                │
│  • Transak        │  • API3           │  • Jumio                 │
└────────────────────────────────────┴──────────────────────────────────┴────────
───────────────────────────────────────────────┘
```

### Service Architecture

```typescript
// Backend Services Structure (NestJS)

src/
├── auth/                    // Authentication & Authorization
│   ├── auth.service.ts
│   ├── jwt.strategy.ts
│   ├── kyc.service.ts
│   └── 2fa.service.ts
│
├── miners/              // NFT Miner Management
│   ├── miners.service.ts
│   ├── upgrades.service.ts
│   ├── efficiency.service.ts
│   └── metadata.service.ts
│
├── rewards/             // BTC Rewards Engine
│   ├── rewards.service.ts   // Daily calculations
│   ├── distribution.service.ts
│   ├── merkle.service.ts
│   └── proofs.service.ts
│
├── maintenance/         // Maintenance & Discounts
│   ├── maintenance.service.ts
│   ├── discounts.service.ts
│   ├── service-button.service.ts
│   └── invoices.service.ts
│
├── marketplace/         // P2P Marketplace
│   ├── listings.service.ts
│   ├── offers.service.ts
│   ├── sales.service.ts
│   └── escrow.service.ts
│
├── governance/          // veTYT & DAO
│   ├── locks.service.ts
│   ├── proposals.service.ts
│   ├── voting.service.ts
│   └── treasury.service.ts
│
├── game/                // Miner Wars
│   ├── clans.service.ts
│   ├── tournaments.service.ts
│   ├── boosts.service.ts
│   └── leaderboards.service.ts
│
├── payments/            // Payments & Withdrawals
│   ├── deposits.service.ts
│   ├── withdrawals.service.ts
│   ├── on-ramp.service.ts
│   └── custodial.service.ts
│
├── foundation/          // Children's Cancer Foundation
```

| ├─── campaigns.service.ts
| ├─── donations.service.ts
| ├─── grants.service.ts
| └─── reports.service.ts
|
├─── academy/   // Educational Platform
| ├─── courses.service.ts
| ├─── lessons.service.ts
| ├─── certificates.service.ts
| └─── xp.service.ts
|
└─── blockchain/   // Blockchain Integration
    ├─── tron.service.ts
    ├─── solana.service.ts
    ├─── indexer.service.ts
    └─── events.service.ts
~~~

<a name="blockchain"></a>
4. БЛОКЧЕЙН И КОНТРАКТЫ

Выбор блокчейна

Основная сеть: TRON (TRC-20/721)

Причины:
- ✅ Низкие комиссии (0.1-1 TRX)
- ✅ Высокая пропускная способность
- ✅ Быстрое время подтверждения (~3 сек)
- ✅ Устоявшаяся экосистема
- ✅ TRC-20 USDT — самый популярный стейблкоин

Опциональные мосты:
- Polygon (EVM-совместимость)
- Solana (TYT токен origin — pump.fun)
- BSC (дополнительная ликвидность)

Smart Contracts Architecture

1. TYT Token (TRC-20)

```solidity
// TYT.sol - Main Token Contract

contract TYT is TRC20, Ownable, Burnable {
    // Roles
    address public treasury;
    address public burner;
    address public discountController;

    // Permit (EIP-2612-like)
~~~

```
    mapping(address => uint256) public nonces;

    // Functions
    function mint(address to, uint256 amount) external onlyRole(MINTER_ROLE);
    function burn(uint256 amount) public;
    function burnFrom(address account, uint256 amount) public;
    function permit(...) external; // Gasless approvals

    // Events
    event Burned(uint256 amount, uint256 timestamp);
    event MaintenancePaid(address user, uint256 amount, uint256 discount);
}
```

Address: `T...` (to be deployed)

#### 2. Miner NFT (TRC-721)

```solidity
// MinerNFT.sol - Digital Miners

contract MinerNFT is TRC721, Ownable {
    struct Miner {
        uint256 hashrate;        // TH/s
        uint256 efficiency;      // W/TH
        uint8 powerLevel;        // 1-20
        uint256 maintenanceRate; // Daily cost in USD
        string farmId;           // Data center ID
        bool isListed;           // Marketplace status
        uint256 listedPrice;
        uint256 totalRewardsBtc;
        uint256 lastRewardAt;
    }

    mapping(uint256 => Miner) public miners;

    // Functions
    function mint(address to, MinerParams memory params) external;
    function upgrade(uint256 tokenId, uint8 newPowerLevel) external;
    function improveEfficiency(uint256 tokenId, uint256 percent) external;
    function list(uint256 tokenId, uint256 price) external;
    function unlist(uint256 tokenId) external;

    // Events
    event MinerMinted(uint256 indexed tokenId, address owner, uint256 hashrate);
    event MinerUpgraded(uint256 indexed tokenId, uint8 fromLevel, uint8 toLevel);
    event RewardAccrued(uint256 indexed tokenId, uint256 btcAmount);
}
```

Address: `T...` (to be deployed)

#### 3. RewardsTreasury

```solidity
// RewardsTreasury.sol - BTC Reward Distribution

contract RewardsTreasury is Ownable {
    // Wrapped BTC on TRON (or custodial system)
    ITRC20 public wbtc;
    // Daily snapshot system
    mapping(uint256 => DailySnapshot) public snapshots;

    struct DailySnapshot {
        uint256 totalHashrate;
        uint256 btcRewardPool;
        uint256 networkDifficulty;
        bytes32 merkleRoot;
    }

    // Functions
    function createSnapshot(uint256 date, ...) external onlyOracle;
    function claimReward(uint256 date, bytes32[] memory proof) external;
    function calculateDailyReward(uint256 hashrate, uint256 efficiency) public view returns (uint256);
}
```

#### 4. Maintenance & Discounts

```solidity
// Maintenance.sol - Maintenance Fee Controller

contract Maintenance is Ownable {
    // Fee parameters (adjustable via Governance)
    uint256 public kwhRate = 0.05e6; // $0.05 per kWh (6 decimals)
    uint256 public serviceFeePercent = 10; // 10%

    // Discount system
    mapping(address => uint256) public userDiscountPercent; // 0-20% for token payment
    mapping(address => uint256) public serviceButtonLastUsed; // Daily -3%
    mapping(address => uint8) public vipLevel; // 0-10

    // VIP discounts
    uint8[11] public VIP_DISCOUNTS = [0, 1, 2, 3, 4, 5, 7, 9, 11, 13, 15];

    // Functions
    function calculateDailyCost(
        uint256 hashrate,
        uint256 efficiency,
        uint256 btcPrice
    ) public view returns (uint256);

    function payMaintenance(uint256 minerId, address paymentToken) external;
    function activateServiceButton() external;
    function calculateTotalDiscount(address user) public view returns (uint256);
}
```

#### 5. Marketplace

```solidity
// Marketplace.sol - P2P Trading

contract Marketplace is Ownable {
    ITRC20 public tytToken;
    IMinerNFT public minerNFT;

    struct Listing {
        address seller;
        uint256 price; // in TYT
        bool active;
    }

    mapping(uint256 => Listing) public listings;

    uint256 public platformFeePercent = 3; // 3%
    address public foundationWallet;

    // Functions
    function createListing(uint256 tokenId, uint256 price) external;
    function cancelListing(uint256 tokenId) external;
    function buy(uint256 tokenId) external;
    function makeOffer(uint256 tokenId, uint256 price) external;

    // Events
    event Listed(uint256 indexed tokenId, address seller, uint256 price);
    event Sold(uint256 indexed tokenId, address seller, address buyer, uint256 price);
}
```

#### 6. veTYT (Vote-Escrowed TYT)

```solidity
// veTYT.sol - Governance Locks

contract veTYT is Ownable {
    ITRC20 public tytToken;

    struct Lock {
        uint256 amount;
        uint256 unlockTime;
        uint256 votingPower;
        bool isActive;
    }

    mapping(address => Lock) public locks;
    uint256 public constant MIN_LOCK_TIME = 1 weeks;
    uint256 public constant MAX_LOCK_TIME = 4 * 365 days;

    // Voting power formula: amount * (lockTime / MAX_LOCK_TIME)
    // Example: 1000 TYT locked for 4 years = 1000 voting power
    //          1000 TYT locked for 1 year = 250 voting power
    function lock(uint256 amount, uint256 lockTime) external;
    function extendLock(uint256 newUnlockTime) external;
    function increaseAmount(uint256 additionalAmount) external;
    function unlock() external;
    function getVotingPower(address user) public view returns (uint256);

    // Events
    event Locked(address indexed user, uint256 amount, uint256 unlockTime, uint256
votingPower);
    event Unlocked(address indexed user, uint256 amount);
}
```

Address: `T...` (to be deployed)

#### 7. BurnScheduler (Weekly Automation)

```solidity
// BurnScheduler.sol - Weekly Burn Events

contract BurnScheduler is Ownable {
    ITRC20 public tytToken;
    address public charityMint; // Solana program address

    uint256 public constant BURN_DAY = 2; // Tuesday
    uint256 public constant BURN_HOUR = 12; // 12:00 UTC

    struct BurnEvent {
        uint256 id;
        uint256 timestamp;
        uint256 collectedAmount;
        uint256 burnedAmount;
        uint256 charityMintAmount;
        bytes32 txHash;
        bool executed;
    }

    BurnEvent[] public burnHistory;

    // Distribution percentages (adjustable via governance)
    uint256 public hashrateProvidersPercent = 40;
    uint256 public veLockersPercent = 30;
    uint256 public treasuryPercent = 20;
    uint256 public charityPercent = 10; // Charity gets up to 25% via Charity Mint

    // Functions
    function executeBurn() external;
    function calculateDistribution(uint256 burnedAmount) public view returns (
        uint256 toHashrateProviders,
        uint256 toVeLockers,
        uint256 toTreasury,
        uint256 toCharity
    );
    function setDistributionPercents(...) external onlyGovernance;

    // Events
    event BurnExecuted(uint256 indexed eventId, uint256 burned, uint256 charityMinted);
    event DistributionSent(address indexed recipient, uint256 amount);
}
```

#### 8. FundSplitter (1% Auto-Donation)

```solidity
// FundSplitter.sol - Automatic Foundation Contributions

contract FundSplitter {
    address public foundationWallet;

    // 1% of all transactions
    uint256 public constant FOUNDATION_PERCENT = 100; // 1% = 100 basis points

    mapping(bytes32 => uint256) public contributionsByTxType;
    uint256 public totalContributed;

    // Functions
    function splitPayment(uint256 amount) internal returns (uint256 netAmount, uint256
foundationAmount) {
        foundationAmount = amount * FOUNDATION_PERCENT / 10000;
        netAmount = amount - foundationAmount;

        // Send to foundation
        payable(foundationWallet).transfer(foundationAmount);
        totalContributed += foundationAmount;

        emit DonationMade(msg.sender, foundationAmount);
    }

    // Events
    event DonationMade(address indexed from, uint256 amount);
}
```

### Contract Deployment Checklist

- [ ] Deploy TYT Token (TRC-20) on TRON mainnet
- [ ] Deploy MinerNFT (TRC-721) on TRON mainnet
- [ ] Deploy RewardsTreasury contract
- [ ] Deploy Maintenance contract
- [ ] Deploy Marketplace contract
- [ ] Deploy veTYT contract
- [ ] Deploy BurnScheduler contract
- [ ] Deploy FundSplitter contract
- [ ] Verify all contracts on TronScan
- [ ] Set up multi-sig for contract ownership
- [ ] Configure Oracle for BTC price feeds
- [ ] Set up automated cron jobs for burn events

<a name="database"></a>
## 5. БАЗА ДАННЫХ (PostgreSQL/Supabase)

### Database Overview

Total Tables: 50+
Migrations: 8 applied
Security: Row Level Security (RLS) enabled on all tables

### Core Schema

#### User Management

```sql
-- profiles: Extended user data
CREATE TABLE profiles (
  id uuid PRIMARY KEY REFERENCES auth.users(id),
  email text UNIQUE NOT NULL,
  username text UNIQUE,
  kyc_status text DEFAULT 'pending',
  kyc_level integer DEFAULT 0,
  vip_level integer DEFAULT 0,
  total_hashrate numeric DEFAULT 0,
  rank_score numeric DEFAULT 0,
  owl_rank text DEFAULT 'worker',
  avatar_id uuid REFERENCES avatars(id),
  referred_by uuid REFERENCES profiles(id),
  referral_code text UNIQUE NOT NULL,
  created_at timestamptz DEFAULT now(),
  updated_at timestamptz DEFAULT now()
);

-- custodial_wallets: Multi-asset balances
CREATE TABLE custodial_wallets (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  asset text NOT NULL, -- BTC, USDT, TYT, ETH, SOL, TRX, XRP
  balance numeric DEFAULT 0,
  locked_balance numeric DEFAULT 0,
  created_at timestamptz DEFAULT now(),
  UNIQUE(user_id, asset)
);

-- wallet_transactions: All financial activity
CREATE TABLE wallet_transactions (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  transaction_type text NOT NULL,
  asset text NOT NULL,
  amount numeric NOT NULL,
  fee numeric DEFAULT 0,
  status text DEFAULT 'pending',
  blockchain_tx_hash text,
  metadata jsonb DEFAULT '{}',
  created_at timestamptz DEFAULT now()
);
```

#### NFT Miners

```sql
-- nft_collections: Miner series
CREATE TABLE nft_collections (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  name text NOT NULL,
  symbol text NOT NULL,
  contract_address text UNIQUE,
  total_supply integer DEFAULT 0,
  base_hashrate numeric NOT NULL,
  base_efficiency numeric NOT NULL,
  floor_price numeric,
  is_active boolean DEFAULT true,
  image_url text,
  created_at timestamptz DEFAULT now()
);

-- nft_miners: Individual miners
CREATE TABLE nft_miners (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  token_id text UNIQUE NOT NULL,
  owner_id uuid NOT NULL REFERENCES profiles(id),
  collection_id uuid NOT NULL REFERENCES nft_collections(id),
  name text NOT NULL,
  hashrate numeric NOT NULL CHECK (hashrate > 0),
  efficiency numeric NOT NULL CHECK (efficiency > 0),
  power_level integer DEFAULT 1 CHECK (power_level BETWEEN 1 AND 20),
  maintenance_rate numeric DEFAULT 0,
  farm_id text REFERENCES data_centers(id),
  status text DEFAULT 'active',
  is_listed boolean DEFAULT false,
  total_rewards_btc numeric DEFAULT 0,
  created_at timestamptz DEFAULT now()
);

-- miner_upgrades: Upgrade history
CREATE TABLE miner_upgrades (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  miner_id uuid NOT NULL REFERENCES nft_miners(id),
  user_id uuid NOT NULL REFERENCES profiles(id),
  upgrade_type text NOT NULL, -- hashrate, efficiency, power_level
  from_value numeric NOT NULL,
  to_value numeric NOT NULL,
  cost numeric NOT NULL,
  cost_currency text DEFAULT 'TYT',
  created_at timestamptz DEFAULT now()
);

-- miner_upgrade_tiers: Cost schedule for 20 levels
CREATE TABLE miner_upgrade_tiers (
  level integer PRIMARY KEY CHECK (level BETWEEN 1 AND 20),
  max_hashrate numeric NOT NULL,
  upgrade_cost_tyt numeric NOT NULL,
  upgrade_cost_btc numeric,
  efficiency_improvement_percent numeric DEFAULT 5,
  description text
);
```

#### Rewards System

```sql
-- daily_rewards: BTC distribution tracking
CREATE TABLE daily_rewards (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  miner_id uuid NOT NULL REFERENCES nft_miners(id),
  reward_date date NOT NULL,
  gross_btc numeric NOT NULL,
  maintenance_cost_btc numeric NOT NULL,
  net_btc numeric NOT NULL,
  hashrate_snapshot_th numeric NOT NULL,
  global_difficulty numeric,
  merkle_leaf text,
  created_at timestamptz DEFAULT now(),
  UNIQUE(miner_id, reward_date)
);

-- maintenance_invoices: Payment tracking
CREATE TABLE maintenance_invoices (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  miner_id uuid NOT NULL REFERENCES nft_miners(id),
  invoice_date date NOT NULL,
  electricity_cost_usd numeric NOT NULL,
  service_fee_usd numeric NOT NULL,
  total_usd numeric NOT NULL,
  discount_applied_pct numeric DEFAULT 0,
  final_amount_usd numeric NOT NULL,
  paid_in_asset text,
  status text DEFAULT 'pending',
  due_date date NOT NULL,
  paid_at timestamptz,
  created_at timestamptz DEFAULT now()
);

-- service_button_activations: Daily -3% discount
CREATE TABLE service_button_activations (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  activation_date date NOT NULL DEFAULT CURRENT_DATE,
  discount_percent numeric DEFAULT 3,
  activated_at timestamptz DEFAULT now(),
  expires_at timestamptz NOT NULL,
  maintenance_saved numeric DEFAULT 0,
  UNIQUE(user_id, activation_date)
);
```

#### Marketplace

```sql
-- marketplace_listings: P2P sales
CREATE TABLE marketplace_listings (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  miner_id uuid NOT NULL REFERENCES nft_miners(id),
  seller_id uuid NOT NULL REFERENCES profiles(id),
  list_price_amount numeric NOT NULL,
  list_price_asset text DEFAULT 'TYT',
  listing_type text DEFAULT 'fixed_price',
  status text DEFAULT 'active',
  created_at timestamptz DEFAULT now()
);

-- marketplace_offers: Buyer bids
CREATE TABLE marketplace_offers (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  listing_id uuid NOT NULL REFERENCES marketplace_listings(id),
  buyer_id uuid NOT NULL REFERENCES profiles(id),
  offer_amount numeric NOT NULL,
  offer_currency text DEFAULT 'TYT',
  status text DEFAULT 'pending',
  expires_at timestamptz,
  created_at timestamptz DEFAULT now()
);

-- marketplace_sales: Completed transactions
CREATE TABLE marketplace_sales (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  listing_id uuid NOT NULL REFERENCES marketplace_listings(id),
  miner_id uuid NOT NULL REFERENCES nft_miners(id),
  seller_id uuid NOT NULL REFERENCES profiles(id),
  buyer_id uuid NOT NULL REFERENCES profiles(id),
  sale_amount numeric NOT NULL,
  platform_fee_amount numeric NOT NULL,
  seller_net_amount numeric NOT NULL,
  referrer_commission_amount numeric DEFAULT 0,
  blockchain_tx_hash text,
  completed_at timestamptz DEFAULT now()
);
```

#### Game Wars (Miner Wars)

```sql
-- game_clans: Clan system (max 50 members)
CREATE TABLE game_clans (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  name text UNIQUE NOT NULL,
  tag text UNIQUE NOT NULL CHECK (length(tag) <= 6),
  leader_id uuid NOT NULL REFERENCES profiles(id),
  description text,
  total_members integer DEFAULT 1,
  total_hashrate numeric DEFAULT 0,
  total_btc_won numeric DEFAULT 0,
  global_rank integer,
  win_count integer DEFAULT 0,
  is_recruiting boolean DEFAULT true,
  min_hashrate_requirement numeric DEFAULT 0,
  created_at timestamptz DEFAULT now()
);

-- game_clan_members: Membership
CREATE TABLE game_clan_members (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  clan_id uuid NOT NULL REFERENCES game_clans(id),
  user_id uuid NOT NULL REFERENCES profiles(id),
  rank text DEFAULT 'private', -- private, corporal, sergeant, lieutenant, captain, leader
  hashrate_contribution numeric DEFAULT 0,
  battles_participated integer DEFAULT 0,
  points_earned numeric DEFAULT 0,
  joined_at timestamptz DEFAULT now(),
  UNIQUE(clan_id, user_id)
);

-- game_tournaments: Competitions
CREATE TABLE game_tournaments (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  name text NOT NULL,
  tournament_type text NOT NULL,
  status text DEFAULT 'upcoming',
  prize_pool_btc numeric DEFAULT 0,
  prize_pool_tyt numeric DEFAULT 0,
  winning_clan_id uuid REFERENCES game_clans(id),
  starts_at timestamptz NOT NULL,
  ends_at timestamptz NOT NULL,
  min_hashrate numeric DEFAULT 0,
  entry_fee_tyt numeric DEFAULT 0,
  created_at timestamptz DEFAULT now()
);

-- game_boosts: Purchasable power-ups
CREATE TABLE game_boosts (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  boost_type text NOT NULL,
  boost_name text NOT NULL,
  boost_value numeric NOT NULL,
  duration_hours integer NOT NULL,
  cost_tyt numeric NOT NULL,
  activated_at timestamptz DEFAULT now(),
  expires_at timestamptz NOT NULL,
  is_active boolean DEFAULT true
);
```

#### VIP & Loyalty

```sql
-- vip_tiers: 11 levels (0-10)
CREATE TABLE vip_tiers (
  level integer PRIMARY KEY CHECK (level BETWEEN 0 AND 10),
  name text NOT NULL,
  min_hashrate numeric,
  min_voting_power numeric,
  requirement_type text DEFAULT 'either',
  maintenance_discount_percent numeric DEFAULT 0,
  marketplace_fee_discount_percent numeric DEFAULT 0,
  priority_support boolean DEFAULT false,
  early_access boolean DEFAULT false,
  exclusive_avatars boolean DEFAULT false,
  custom_badge text
);

-- avatars: Bonus NFTs
CREATE TABLE avatars (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  token_id text UNIQUE NOT NULL,
  owner_id uuid NOT NULL REFERENCES profiles(id),
  name text NOT NULL,
  rarity text NOT NULL, -- common, uncommon, rare, epic, legendary
  image_url text,
  boost_type text,
  boost_value numeric DEFAULT 0,
  is_equipped boolean DEFAULT false,
  is_tradeable boolean DEFAULT true,
  created_at timestamptz DEFAULT now()
);

-- goboxes: VIP upgrade rewards
CREATE TABLE goboxes (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  rarity text NOT NULL,
  vip_level_achieved integer NOT NULL,
  tyt_reward numeric DEFAULT 0,
  btc_reward numeric DEFAULT 0,
  avatar_id uuid REFERENCES avatars(id),
  boost_duration_hours integer DEFAULT 0,
  is_opened boolean DEFAULT false,
  opened_at timestamptz,
  created_at timestamptz DEFAULT now()
);
```

#### Referral System (5-5-5)

```sql
-- referral_earnings: Commission tracking
```

```sql
CREATE TABLE referral_earnings (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  referrer_id uuid NOT NULL REFERENCES profiles(id),
  referred_user_id uuid NOT NULL REFERENCES profiles(id),
  event_type text NOT NULL,
  base_amount numeric NOT NULL,
  commission_percent numeric NOT NULL,
  commission_amount numeric NOT NULL,
  status text DEFAULT 'pending',
  paid_at timestamptz,
  created_at timestamptz DEFAULT now()
);

-- ambassadors: High-tier referrers
CREATE TABLE ambassadors (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  tier text DEFAULT 'bronze',
  status text DEFAULT 'active',
  purchase_commission_percent numeric DEFAULT 5,
  upgrade_commission_percent numeric DEFAULT 5,
  marketplace_commission_percent numeric DEFAULT 5,
  total_referrals integer DEFAULT 0,
  total_earnings_tyt numeric DEFAULT 0,
  custom_code text UNIQUE,
  assigned_at timestamptz DEFAULT now()
);
```

#### Token Economics

```sql
-- ve_tyt_locks: Governance voting power
CREATE TABLE ve_tyt_locks (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  amount numeric NOT NULL,
  lock_duration_seconds bigint NOT NULL,
  voting_power numeric NOT NULL,
  locked_at timestamptz DEFAULT now(),
  unlock_at timestamptz NOT NULL,
  is_active boolean DEFAULT true,
  withdrawn_at timestamptz
);

-- governance_proposals: DAO voting
CREATE TABLE governance_proposals (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  proposer_id uuid NOT NULL REFERENCES profiles(id),
  title text NOT NULL,
  description text NOT NULL,
  proposal_type text NOT NULL,
  status text DEFAULT 'active',
  votes_for numeric DEFAULT 0,
  votes_against numeric DEFAULT 0,
  quorum_required numeric NOT NULL,
  created_at timestamptz DEFAULT now(),
  voting_ends_at timestamptz NOT NULL,
  executed_at timestamptz
);
```

```sql
-- token_burn_events: Weekly schedule (Tuesdays 12:00 UTC)
CREATE TABLE token_burn_events (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  scheduled_at timestamptz NOT NULL,
  collected_amount numeric NOT NULL,
  burned_amount numeric NOT NULL,
  charity_mint_amount numeric NOT NULL,
  burn_tx_hash text,
  mint_tx_hash text,
  status text DEFAULT 'scheduled',
  executed_at timestamptz,
  created_at timestamptz DEFAULT now()
);

-- burn_mint_distributions: Stakeholder payouts
CREATE TABLE burn_mint_distributions (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  burn_event_id uuid NOT NULL REFERENCES token_burn_events(id),
  hashrate_providers_amount numeric NOT NULL,
  ve_lockers_amount numeric NOT NULL,
  community_treasury_amount numeric NOT NULL,
  charity_foundation_amount numeric NOT NULL,
  created_at timestamptz DEFAULT now()
);
```

#### Academy

```sql
-- academy_tracks: Learning paths
CREATE TABLE academy_tracks (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  slug text UNIQUE NOT NULL,
  title text NOT NULL,
  description text,
  difficulty_level integer DEFAULT 1,
  estimated_hours integer,
  is_published boolean DEFAULT false,
  display_order integer DEFAULT 0,
  created_at timestamptz DEFAULT now()
);

-- academy_lessons: Course content
CREATE TABLE academy_lessons (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  track_id uuid NOT NULL REFERENCES academy_tracks(id),
  slug text UNIQUE NOT NULL,
  title text NOT NULL,
  content_mdx text NOT NULL,
  lesson_type text DEFAULT 'theory',
  xp_reward integer DEFAULT 0,
  is_published boolean DEFAULT false,
  display_order integer DEFAULT 0,
  created_at timestamptz DEFAULT now()
);

-- user_lesson_progress: Completion tracking
CREATE TABLE user_lesson_progress (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id uuid NOT NULL REFERENCES profiles(id),
  lesson_id uuid NOT NULL REFERENCES academy_lessons(id),
  status text DEFAULT 'not_started',
  xp_earned integer DEFAULT 0,
  completed_at timestamptz,
  UNIQUE(user_id, lesson_id)
);
```

#### Foundation

```sql
-- foundation_campaigns: Research funding
CREATE TABLE foundation_campaigns (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  slug text UNIQUE NOT NULL,
  title text NOT NULL,
  description text NOT NULL,
  goal_usd numeric NOT NULL,
  raised_usd numeric DEFAULT 0,
  campaign_type text NOT NULL,
  status text DEFAULT 'active',
  start_date timestamptz NOT NULL,
  end_date timestamptz,
  featured_image_url text,
  created_at timestamptz DEFAULT now()
);

-- foundation_donations: Transparent tracking
CREATE TABLE foundation_donations (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  campaign_id uuid REFERENCES foundation_campaigns(id),
  donor_id uuid REFERENCES profiles(id),
  amount_usd numeric NOT NULL,
  source_transaction_id uuid REFERENCES wallet_transactions(id),
  is_anonymous boolean DEFAULT false,
  donated_at timestamptz DEFAULT now()
);

-- foundation_fund_distributions: Grant allocations
CREATE TABLE foundation_fund_distributions (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  campaign_id uuid NOT NULL REFERENCES foundation_campaigns(id),
  recipient_name text NOT NULL,
  amount_usd numeric NOT NULL,
  purpose text NOT NULL,
  report_url text,
  distributed_at timestamptz DEFAULT now()
);
```

### Database Functions

```sql
-- Calculate daily BTC reward for a miner
CREATE OR REPLACE FUNCTION calculate_daily_btc_reward(
  p_hashrate numeric,
  p_efficiency numeric,
  p_btc_network_hashrate numeric DEFAULT 400000000,
  p_btc_block_reward numeric DEFAULT 3.125,
  p_blocks_per_day numeric DEFAULT 144
) RETURNS numeric AS $$
DECLARE
  v_daily_btc numeric;
BEGIN
  v_daily_btc := (p_hashrate / p_btc_network_hashrate) * (p_btc_block_reward *
p_blocks_per_day);
  v_daily_btc := v_daily_btc * (25.0 / GREATEST(p_efficiency, 1));
  RETURN v_daily_btc;
END;
$$ LANGUAGE plpgsql IMMUTABLE;

-- Calculate daily maintenance cost
CREATE OR REPLACE FUNCTION calculate_daily_maintenance(
  p_hashrate numeric,
  p_efficiency numeric,
  p_kwh_rate numeric DEFAULT 0.05,
  p_btc_price_usd numeric DEFAULT 40000
) RETURNS numeric AS $$
DECLARE
  v_daily_kwh numeric;
  v_daily_cost_usd numeric;
BEGIN
  v_daily_kwh := (p_efficiency * p_hashrate * 24) / 1000.0;
  v_daily_cost_usd := v_daily_kwh * p_kwh_rate;
  RETURN v_daily_cost_usd;
END;
$$ LANGUAGE plpgsql IMMUTABLE;

-- Get user's total discount percentage
CREATE OR REPLACE FUNCTION get_user_total_discount(
  p_user_id uuid,
  p_paying_with_tyt boolean,
  p_service_button_active boolean
) RETURNS numeric AS $$
DECLARE
  v_vip_level integer;
  v_vip_discount numeric;
  v_token_discount numeric := 0;
  v_service_discount numeric := 0;
  v_total_discount numeric;
BEGIN
  SELECT vip_level INTO v_vip_level FROM profiles WHERE id = p_user_id;
  SELECT maintenance_discount_percent INTO v_vip_discount FROM vip_tiers WHERE level =
v_vip_level;

  IF p_paying_with_tyt THEN
    v_token_discount := 20;
  END IF;

  IF p_service_button_active THEN
    v_service_discount := 3;
  END IF;

  v_total_discount := LEAST(v_vip_discount + v_token_discount + v_service_discount, 50);
  RETURN v_total_discount;
END;
$$ LANGUAGE plpgsql;
```

<a name="modules"></a>
## 6. ФУНКЦИОНАЛЬНЫЕ МОДУЛИ

### 6.1 Authentication & KYC

Components:
- Email/password authentication with Supabase Auth
- 2FA (TOTP) for enhanced security
- Social login (Google, Apple) - optional
- KYC integration with Sumsub/Onfido
- 3-tier KYC system:
  - Level 0: Basic registration (no KYC) - $1,000 limit
  - Level 1: ID verification - $10,000 limit
  - Level 2: Enhanced verification - unlimited

Flow:
1. User signs up with email
2. Email verification required
3. Optional 2FA setup
4. KYC prompted at first deposit/withdrawal
5. Gradual level progression based on transaction volume

### 6.2 NFT Miner Management

Features:
- Browse miner collections
- View miner stats (hashrate, efficiency, power level, ROI)
- Upgrade miners (20 power levels: 100 TH/s → 5000 TH/s)
- Improve efficiency (reduce W/TH)
- Assign miners to data centers
- Track historical performance
- Enable/disable reinvestment mode

Upgrade Economics:
```
Level 1:  100 TH/s  →  Cost: 100 TYT (0.0025 BTC)
Level 5:  500 TH/s  →  Cost: 500 TYT (0.0125 BTC)
Level 10: 1500 TH/s →  Cost: 2000 TYT (0.05 BTC)
Level 15: 3000 TH/s →  Cost: 5000 TYT (0.125 BTC)
Level 20: 5000 TH/s →  Cost: 8000 TYT (0.2 BTC) [MAX]
```

### 6.3 Rewards & Maintenance

Daily Reward Calculation:
```
Gross BTC = (Miner Hashrate / Network Hashrate) × Daily Block Rewards
Efficiency Multiplier = 25 / Miner Efficiency (W/TH)
Final BTC = Gross BTC × Efficiency Multiplier
```

Maintenance Cost:
```
Daily kWh = (Efficiency × Hashrate × 24) / 1000
Electricity Cost = Daily kWh × kWh Rate ($0.05 default)
Service Fee = Electricity Cost × 10%
Gross Maintenance = Electricity Cost + Service Fee
```

Discount Stacking (Max 50%):
1. Token Payment Discount: -20% (pay in TYT)
2. Service Button: -3% (daily activation)
3. VIP Tier: 0-15% (levels 0-10)
4. Balance Coverage: 2-18% (20-360 days of maintenance pre-paid)

Example:
- Gross maintenance: $10/day
- Paying with TYT: -$2 (20%)
- Service button active: -$0.30 (3%)
- VIP level 5: -$0.50 (5%)
- Total discount: 28% → Net cost: $7.20/day

### 6.4 Marketplace

Listing Types:
- Fixed price (TYT only)
- Auction (coming soon)
- Make offer system

Fees:
- Platform fee: 3%
- Referrer commission: 5% (if applicable)
- Foundation donation: 1% (automatic)

Security:
- Escrow system via smart contract
- NFT locked during listing
- Instant settlement on purchase
- Dispute resolution via support

### 6.5 Game Wars (Miner Wars)

Clan System:
- Max 50 members per clan
- Ranks: Private → Corporal → Sergeant → Lieutenant → Captain → Leader
- Combined hashrate determines clan power
- Weekly clan battles

Tournaments:
- Weekly tournaments (every Saturday)
- Monthly championships
- Entry fee: 100-1000 TYT
- Prize pools: 1-10 BTC + 10,000-100,000 TYT

Boosts:
- Hashrate Multiplier: +10-50% for 1-24 hours
- Efficiency Boost: -5-20% W/TH improvement
- Maintenance Discount: Additional -5-15%
- Reward Multiplier: +10-30% BTC rewards

### 6.6 VIP Program

11 Tiers (0-10):

Level	Name	Hashrate OR Voting Power	Maintenance Discount	Marketplace Fee
0	Worker	0	0%	3%
1	Apprentice	100 TH/s or 1,000 veTYT	1%	2.9%

2	Skilled	250 TH/s or 2,500 veTYT	2%	2.8%
3	Expert	500 TH/s or 5,000 veTYT	3%	2.7%
4	Master	1,000 TH/s or 10,000 veTYT	4%	2.6%
5	Elite	2,500 TH/s or 25,000 veTYT	5%	2.5%
6	Champion	5,000 TH/s or 50,000 veTYT	7%	2.3%
7	Legend	10,000 TH/s or 100,000 veTYT	9%	2.1%
8	Mythic	25,000 TH/s or 250,000 veTYT	11%	1.9%
9	Immortal	50,000 TH/s or 500,000 veTYT	13%	1.7%
10	Eternal Owl	100,000 TH/s or 1M veTYT	15%	1.5%

Benefits:
- Maintenance discounts (cumulative with other discounts)
- Reduced marketplace fees
- Priority customer support
- Early access to new features
- Exclusive avatars and badges
- GoBox rewards on level-up

### 6.7 Referral Program (5-5-5)

Commission Structure:
- Miner purchases: 5% of purchase price
- Miner upgrades: 5% of upgrade cost
- Marketplace sales: 5% of sale price
- Game boosts: 5% of boost cost

Ambassador Program:
- Bronze: 5 active referrals → 5% commission
- Silver: 25 active referrals → 6% commission
- Gold: 100 active referrals → 7% commission
- Platinum: 500 active referrals → 8% commission
- Diamond: 2,000 active referrals → 10% commission

Tracking:
- Custom referral codes
- Dashboard with earnings breakdown
- Real-time commission updates
- Monthly payouts in TYT

### 6.8 Academy (Educational Platform)

Learning Tracks:
1. Bitcoin Basics (5 lessons, 50 XP each)
2. Mining 101 (10 lessons, 100 XP each)
3. Blockchain Technology (15 lessons, 150 XP each)
4. DeFi & Trading (20 lessons, 200 XP each)
5. Advanced Mining Economics (25 lessons, 250 XP each)

Features:
- Interactive lessons with quizzes
- Video content and infographics
- Soulbound NFT certificates (non-transferable)
- XP progression system
- Owl Rank advancement

Owl Ranks:
- Worker: 0-1,000 XP
- Academic: 1,001-5,000 XP
- Diplomat: 5,001-15,000 XP
- Peacekeeper: 15,001-40,000 XP

- Warrior: 40,001+ XP

### 6.9 Foundation (Children's Brain Cancer)

Funding Sources:
- 1% of all platform transactions (automatic)
- Direct donations from users
- 10-25% of weekly token burns (Charity Mint)
- Corporate partnerships

Transparency:
- Public campaign tracking
- Real-time donation dashboard
- Quarterly impact reports
- Grant recipient updates
- Hospital partnerships showcase

Current Focus:
- Research grants for pediatric brain cancer
- Family support programs
- Medical equipment funding
- Clinical trial support

<a name="tokenomics"></a>
## 7. ТОКЕНОМИКА

### TYT Token Overview

Token Information:
- Name: Take Your Token
- Symbol: TYT
- Origin: Solana (pump.fun)
- Bridge: TRON (TRC-20) primary
- Max Supply: TBD (inflationary with burn mechanism)

### Utility

1. **Maintenance Payments** → 100% burned
2. **Miner Upgrades** → 100% burned
3. **Marketplace Currency** → 3% fee burned
4. **Game Boosts** → 100% burned
5. **Governance Voting** (via veTYT locks)
6. **VIP Level Qualification** (via veTYT)
7. **Academy Access** (premium courses)

### Burn & Mint Mechanism

Weekly Burn Schedule:
- Day: Tuesday
- Time: 12:00 UTC
- Frequency: Every week

Burn Sources:
- All maintenance payments
- All upgrade costs
- Marketplace fees (3%)
- Game boost purchases
- Manual burns from treasury

Distribution After Burn:
```
40% → Hashrate Providers (proportional to TH/s)
30% → veTYT Lockers (proportional to voting power)
20% → Community Treasury (governance-controlled)
10% → Foundation (+ up to 25% Charity Mint)
```

Charity Mint:
- Minted on Solana as new TYT
- Amount: 0-25% of burned TYT
- Destination: Foundation wallet
- Governance vote can adjust percentage

### veTYT (Vote-Escrowed TYT)

Lock Periods:
- Minimum: 1 week
- Maximum: 4 years (208 weeks)

Voting Power Formula:
```
Voting Power = TYT Amount × (Lock Time / Max Lock Time)

Examples:
- 1,000 TYT locked for 4 years = 1,000 veTYT
- 1,000 TYT locked for 2 years = 500 veTYT
- 1,000 TYT locked for 1 year = 250 veTYT
- 1,000 TYT locked for 1 month = ~21 veTYT
```

Benefits:
- Governance voting rights
- VIP level qualification (alternative to hashrate)
- Weekly distribution from burn events (30% share)
- Priority in platform decisions

Decay:
- Voting power decays linearly over time
- Users can extend lock or add more TYT
- No early unlock (except via governance proposal)

### Token Flow Diagram

```
┌──────────────────────────────────────────────────────────
──────────────────────────────┐
│                    USER ACTIONS                    │
├───────────────────────┬───────────────────────┬─────────
───┬──────────────────────────┤

│ Buy Miners    │ Upgrade      │ Marketplace  │ Game Boosts    │
│ Pay in TYT   │ Pay in TYT   │ 3% Fee       │ Pay in TYT     │
└──────────┬───────────┴──────────┬───────────┴──────────┬──────────
───┴──────────┬────────────────┘
     │          │          │            │
```

```
└──────────────────────────────┴──────────────────────────────┴──────────────────────────
─────────┘
                    ↓
          ┌──────────────────────────────────────────────────────────┐
          │  BURN ACCUMULATOR WALLET   │
          │  Collects all TYT to burn  │
          └─────────────────────────────┬────────────────────────────┘
                    ↓
          Every Tuesday 12:00 UTC
                    ↓
          ┌──────────────────────────────────────────────────────────┐
          │     BURN EVENT EXECUTED    │
          │  X TYT burned on TRON      │
          │  Y TYT minted on Solana    │
          │     (Charity Mint)         │
          └─────────────────────────────┬────────────────────────────┘

                    ↓
┌─────────────────────────────────────────┴──────────────────────────────────────────
─────┐
      │                          │
      ↓                          ↓
┌──────────────────────────────┐          ┌──────────────────────────────────┐
│  DISTRIBUTION  │                        │  CHARITY MINT   │
│   (Burned)     │              │  (New TYT)     │
├──────────────────────────────┤          ├──────────────────────────────────┤
│ 40% Hashrate   │               │ → Foundation    │
│ 30% veTYT      │               │ → Medical       │
│ 20% Treasury   │               │ → Research      │
│ 10% Foundation │               │ → Families      │
└──────────────────────────────┘          └──────────────────────────────────┘
```

### Economic Sustainability

Revenue Streams:
1. Marketplace fees (3%)
2. Miner sale commissions
3. Upgrade revenue (partially funds operations)
4. Game boost sales
5. Partnership revenue

Operational Costs:
- BTC reward pool (must be maintained)
- Data center operations
- Platform development
- Marketing and growth
- Customer support

Deflationary Pressure:
- All maintenance → burned
- All upgrades → burned
- Marketplace fees → burned
- Game boosts → burned

Inflationary Pressure:
- Weekly Charity Mint (0-25% of burn)
- Governance can adjust mint percentage

Net Effect:
- Likely deflationary long-term
- Dependent on Charity Mint governance decisions
- Higher usage = more burn = stronger deflation

<a name="ux-flows"></a>
## 8. UX ПОТОКИ (User Flows)

### 8.1 Onboarding Flow

```
1. Landing Page
   ↓
2. Sign Up (email + password)
   ↓
3. Email Verification
   ↓
4. Welcome Dashboard (Tutorial overlay)
   ↓
5. Connect Wallet OR Use Custodial
   ↓
6. Browse Miner Collections
   ↓
7. Purchase First Miner (with on-ramp guidance)
   ↓
8. View Daily Rewards
```

### 8.2 Daily User Flow

```
1. Login (Email or Social)
   ↓
2. Dashboard Overview
   ├──→ Total Hashrate
   ├──→ Daily BTC Earned
   ├──→ Maintenance Due
   └──→ VIP Progress
   ↓
3. Check Notifications
   ├──→ Maintenance reminders
   ├──→ Reward notifications
   ├──→ Tournament invites
   └──→ Foundation updates
   ↓
4. Service Button (Press for -3% discount)
   ↓
5. Pay Maintenance (if due)
   ↓
```

```
6. Claim Daily Rewards
   ↓
7. Optional: Upgrade Miners, Join Battles, Complete Lessons
```

### 8.3 Miner Purchase Flow

```
1. Browse Marketplace
   ├──→ Filter by hashrate
   ├──→ Filter by price
   └──→ Sort by ROI
   ↓
2. Select Miner
   ↓
3. View Details
   ├──→ Current stats
   ├──→ Historical performance
   ├──→ Data center location
   └──→ ROI calculation
   ↓
4. Add Funds (if needed)
   ├──→ Crypto deposit
   ├──→ Card payment
   └──→ Third-party on-ramp
   ↓
5. Confirm Purchase (Pay in TYT for 20% discount)
   ↓
6. Receive NFT (TRON wallet or custodial)
   ↓
7. Start Earning BTC (next day)
```

### 8.4 Upgrade Flow

```
1. My Miners Page
   ↓
2. Select Miner to Upgrade
   ↓
3. View Upgrade Options
   ├──→ Hashrate upgrade (next power level)
   ├──→ Efficiency improvement
   └──→ Combined upgrade deal
   ↓
4. Cost Calculator
   ├──→ TYT cost
   ├──→ BTC cost (alternative)
   └──→ ROI improvement estimate
   ↓
5. Confirm Upgrade
   ↓
```

6. Transaction Processing
   ↓
7. Updated Stats (immediate)
   ↓
8. Higher Rewards (next day)
```

8.5 Marketplace Selling Flow

```
1. My Miners Page
   ↓
2. Select Miner to Sell
   ↓
3. Set List Price (TYT only)
   ├──→ View floor price
   ├──→ See recent sales
   └──→ Auto-suggest optimal price
   ↓
4. Create Listing
   ↓
5. Miner Locked (cannot earn while listed)
   ↓
6. Buyer Makes Offer OR Purchases
   ↓
7. Accept Offer (if applicable)
   ↓
8. Instant Settlement
   ├──→ Seller receives TYT (minus 3% fee)
   ├──→ Buyer receives NFT
   ├──→ Foundation receives 1%
   └──→ Referrer receives 5% (if applicable)
```

### 8.6 Governance Participation

```
1. Governance Page
   ↓
2. Lock TYT for veTYT
   ├──→ Choose amount
   ├──→ Choose duration (1 week - 4 years)
   └──→ See voting power calculation
   ↓
3. Confirm Lock
   ↓
4. Browse Active Proposals
   ├──→ Fee adjustments
   ├──→ Burn distribution changes
   ├──→ New features
   └──→ Treasury allocations
   ↓
5. Vote on Proposals

    ├──→ For / Against
    ├──→ Voting power applied
    └──→ See current results
   ↓
6. Earn from Burns (30% share)
   ↓
7. Extend Lock or Unlock (after expiry)
```

<a name="tech-stack"></a>
9. ТЕХНИЧЕСКИЙ СТЕК

Frontend

Web Application:
- Framework: React 18 + TypeScript
- Routing: React Router v7
- Styling: Tailwind CSS 3.4
- Icons: Lucide React
- State Management: React Context + Hooks
- API Client: Supabase JS SDK
- Build Tool: Vite 5

Mobile Apps (Planned):
- iOS: Swift + SwiftUI
- Android: Kotlin + Jetpack Compose
- Cross-platform alternative: React Native

Backend

Database:
- PostgreSQL 15 (Supabase)
- Real-time subscriptions
- Row Level Security (RLS)
- 50+ tables with full audit trails

API Layer:
- REST API via Supabase
- Supabase Edge Functions (Deno)
- Real-time WebSocket connections
- Serverless architecture

Backend Services (Future: NestJS):
- Auth Service (Supabase Auth + KYC)
- Rewards Engine (Daily BTC distribution)
- Marketplace Service (Escrow + Settlement)
- Game Engine (Tournaments + Leaderboards)
- Foundation Service (Donations + Reports)

Blockchain

Primary Chain: TRON
- TYT Token: TRC-20
- Miner NFTs: TRC-721
- veTYT Locks: Custom contract
- Marketplace: Escrow contract
```

**Secondary Chains:**
- Solana: TYT origin (pump.fun) + Charity Mint
- Polygon: Optional bridge for wBTC miners
- BSC: Optional bridge for liquidity

**Web3 Integration:**
- TronWeb.js for TRON
- Solana Web3.js
- WalletConnect for multi-chain
- Custodial wallet option (no blockchain knowledge required)

### Infrastructure

**Hosting:**
- Frontend: Vercel / Netlify
- Database: Supabase Cloud
- Edge Functions: Supabase Edge Runtime
- Media Storage: S3 / IPFS

**DevOps:**
- CI/CD: GitHub Actions
- Monitoring: Sentry (errors) + PostHog (analytics)
- Logging: Supabase Logs
- Alerts: Discord webhooks

**Security:**
- SSL/TLS encryption
- Rate limiting (Kong API Gateway)
- DDoS protection (Cloudflare)
- KYC/AML (Sumsub/Onfido)
- Penetration testing (annual)
- Bug bounty program

### External Integrations

**Payment Gateways:**
- Stripe (cards)
- Ramp Network (crypto on-ramp)
- Transak (fiat on-ramp)
- Binance Pay
- Coinbase Pay

**Price Oracles:**
- Chainlink (BTC/USD, TYT/USD)
- Pyth Network (backup)
- CoinGecko API (fallback)

**KYC/AML:**
- Sumsub (primary)
- Onfido (backup)

**Communications:**
- SendGrid (emails)
- Twilio (SMS for 2FA)
- Discord (community + alerts)

---

<a name="api-spec"></a>
## 10. API СПЕЦИФИКАЦИЯ

### Authentication Endpoints

```typescript
POST /auth/signup
Body: { email, password, referralCode? }
Response: { user, session }

POST /auth/login
Body: { email, password }
Response: { user, session, accessToken }

POST /auth/logout
Headers: { Authorization: Bearer <token> }
Response: { success: boolean }

POST /auth/verify-email
Body: { token }
Response: { success: boolean }

POST /auth/2fa/enable
Headers: { Authorization: Bearer <token> }
Response: { secret, qrCode }

POST /auth/2fa/verify
Body: { code }
Response: { success: boolean }
```

### User Profile Endpoints

```typescript
GET /api/profile
Headers: { Authorization: Bearer <token> }
Response: Profile

PATCH /api/profile
Body: { username?, avatar_id? }
Response: Profile

GET /api/profile/stats
Response: {
  totalHashrate, totalRewardsBtc, vipLevel,
  owlRank, xpPoints, referralCount
}
```

### Miner Endpoints

```typescript
GET /api/miners
Query: { owner_id?, collection_id?, is_listed?, limit, offset }
Response: NFTMiner[]

GET /api/miners/:id
Response: NFTMiner & { collection, rewards[], upgrades[] }

POST /api/miners/:id/upgrade
Body: { upgradeType: 'hashrate' | 'efficiency', targetLevel }
Response: { miner: NFTMiner, transaction }
```

```
POST /api/miners/:id/assign-farm
Body: { farmId }
Response: NFTMiner

GET /api/miners/:id/performance
Query: { startDate, endDate }
Response: {
  dailyRewards: DailyReward[],
  totalBtc, avgHashrate, uptimePercent
}
```

### Rewards Endpoints

```typescript
GET /api/rewards
Query: { user_id?, miner_id?, startDate, endDate }
Response: DailyReward[]

GET /api/rewards/summary
Response: {
  todayBtc, weekBtc, monthBtc, allTimeBtc,
  pendingClaim, nextDistribution
}

POST /api/rewards/claim
Body: { rewardIds: uuid[] }
Response: { claimedAmount, txHash }
```

### Maintenance Endpoints

```typescript
GET /api/maintenance/invoices
Query: { status?, limit, offset }
Response: MaintenanceInvoice[]

POST /api/maintenance/pay
Body: { invoiceId, paymentAsset: 'TYT' | 'BTC' | 'USDT' }
Response: { invoice, transaction, appliedDiscounts }

POST /api/maintenance/service-button
Response: {
  activated: boolean,
  discount: number,
  expiresAt, maintenanceSaved
}

GET /api/maintenance/calculate
Query: { minerId, paymentAsset? }
Response: {
  grossCost, discounts[], netCost, breakdown
}
```

### Marketplace Endpoints

```typescript
GET /api/marketplace/listings
```

```typescript
Query: {
  minHashrate?, maxPrice?, sortBy?, limit, offset
}
Response: MarketplaceListing[]

POST /api/marketplace/list
Body: {
  minerId, listPrice, listingType: 'fixed_price'
}
Response: MarketplaceListing

DELETE /api/marketplace/listings/:id
Response: { success: boolean }

POST /api/marketplace/offers
Body: { listingId, offerAmount, message? }
Response: MarketplaceOffer

POST /api/marketplace/offers/:id/accept
Response: {
  sale: MarketplaceSale,
  txHash, sellerAmount, buyerMiner
}

POST /api/marketplace/buy/:listingId
Response: {
  sale: MarketplaceSale,
  miner: NFTMiner, transaction
}
```

### Game Wars Endpoints

```typescript
GET /api/game/clans
Query: { isRecruiting?, minHashrate?, limit, offset }
Response: GameClan[]

POST /api/game/clans
Body: { name, tag, description, minHashrate }
Response: GameClan

POST /api/game/clans/:id/join
Response: GameClanMember

GET /api/game/tournaments
Query: { status?, type? }
Response: GameTournament[]

POST /api/game/tournaments/:id/enter
Body: { clanId? }
Response: { participant, entryFee, receipt }

GET /api/game/boosts
Response: {
  available: GameBoost[],
  active: GameBoost[]
}

POST /api/game/boosts/activate
```

```
Body: {
  boostType, duration, minerId?
}
Response: GameBoost & { transaction }
```

### Governance Endpoints

```typescript
POST /api/governance/lock
Body: { amount, duration }
Response: {
  lock: VeTytLock,
  votingPower, unlockDate, transaction
}

POST /api/governance/extend
Body: { lockId, newUnlockDate }
Response: VeTytLock

POST /api/governance/unlock
Body: { lockId }
Response: { amount, transaction }

GET /api/governance/proposals
Query: { status?, limit, offset }
Response: GovernanceProposal[]

POST /api/governance/vote
Body: { proposalId, vote: 'for' | 'against' }
Response: { votingPower, currentTally }
```

### Foundation Endpoints

```typescript
GET /api/foundation/campaigns
Query: { status?, type? }
Response: FoundationCampaign[]

GET /api/foundation/campaigns/:id
Response: FoundationCampaign & {
  donations[], distributions[], progress
}

POST /api/foundation/donate
Body: {
  campaignId?, amount, asset, isAnonymous?
}
Response: { donation, receipt, taxDocument? }

GET /api/foundation/impact
Response: {
  totalDonated, activeCampaigns,
  familiesSupported, researchGrants
}
```

### Academy Endpoints

```typescript
GET /api/academy/tracks
Response: AcademyTrack[]

GET /api/academy/tracks/:id/lessons
Response: AcademyLesson[]

POST /api/academy/lessons/:id/complete
Body: { quizAnswers? }
Response: {
  xpEarned, newOwlRank?, certificate?, progress
}

GET /api/academy/progress
Response: {
  completedLessons, totalXp, owlRank,
  certificates[], nextMilestone
}
```

<a name="roadmap"></a>
## 11. ДОРОЖНАЯ КАРТА

### Phase 0: Foundation (COMPLETE ✅)

Duration: December 2024
Status: Done

- [x] Project specification (PDF + Word docs)
- [x] Database schema design (50+ tables)
- [x] Migration scripts (8 migrations)
- [x] TypeScript type definitions
- [x] Utility functions (maintenance, upgrades, rewards)
- [x] TRON blockchain integration setup
- [x] Authentication flow (Supabase)
- [x] Basic frontend structure (React + Tailwind)

### Phase 1: MVP Launch (Months 1-3)

Target: Q1 2025

Core Features:
- [ ] Complete UI for all main pages
- [ ] Smart contract deployment (TRON mainnet)
  - TYT Token (TRC-20)
  - MinerNFT (TRC-721)
  - Marketplace Escrow
  - veTYT Locks
- [ ] Custodial wallet integration
- [ ] Miner purchase flow
- [ ] Daily rewards engine
- [ ] Maintenance payment system
- [ ] Basic marketplace (fixed-price listings)
- [ ] KYC integration (Sumsub Level 1)
- [ ] Payment on-ramp (Stripe + Ramp Network)

Deliverables:
- Web app (desktop + mobile responsive)

- Smart contracts (audited)
- Admin dashboard
- Documentation site

### Phase 2: Gaming & Social (Months 4-6)

Target: Q2 2025

New Features:
- [ ] Miner Wars (clan system)
- [ ] Tournament engine
- [ ] Game boosts marketplace
- [ ] Service button (daily discount)
- [ ] VIP tier system
- [ ] GoBox distribution
- [ ] Referral dashboard
- [ ] Ambassador program
- [ ] Social features (chat, leaderboards)
- [ ] Mobile app beta (iOS + Android)

Marketing:
- Influencer partnerships
- Community building (Discord/Telegram)
- First tournament (10 BTC prize pool)
- Ambassador recruitment

### Phase 3: Education & Foundation (Months 7-9)

Target: Q3 2025

New Modules:
- [ ] Academy platform
  - 5 learning tracks
  - 75+ lessons
  - Interactive quizzes
  - Soulbound certificates
- [ ] Foundation transparency dashboard
- [ ] Campaign creation tools
- [ ] Direct donation portal
- [ ] Impact reports (quarterly)
- [ ] Hospital partnership showcase

Content:
- Bitcoin/Blockchain education
- Mining economics courses
- DeFi trading guides
- Video tutorials

### Phase 4: Governance & Decentralization (Months 10-12)

Target: Q4 2025

DAO Launch:
- [ ] veTYT full implementation
- [ ] Proposal creation interface
- [ ] Voting system
- [ ] Treasury management
- [ ] Parameter adjustment (fees, burn %)
- [ ] Weekly burn automation
- [ ] Charity Mint governance

- [ ] Multi-sig treasury

Decentralization:
- Transfer platform control to DAO
- Community-elected moderators
- Open-source selected components
- Bug bounty program

### Phase 5: Expansion (Year 2)

Target: 2026

New Chains:
- [ ] Polygon bridge (wBTC miners)
- [ ] Solana integration (native TYT)
- [ ] TON blockchain support
- [ ] BSC liquidity pools

Advanced Features:
- [ ] Miner derivatives (hashrate futures)
- [ ] Lending protocol (borrow against NFTs)
- [ ] Insurance products
- [ ] Institutional mining (large-scale)
- [ ] Live stream integration (24/7 data center cams)
- [ ] API for third-party developers

Global Growth:
- Multi-language support (10+ languages)
- Regional partnerships
- Physical mining facilities
- Regulatory compliance (EU, US, APAC)

### Phase 6: Maturity (Year 3+)

Long-term Goals:
- 100,000+ active miners (NFTs minted)
- 10,000+ daily active users
- $1B+ in total value locked
- Top 3 Bitcoin mining platform by hashrate
- #1 charity-backed crypto project
- $10M+ raised for children's brain cancer research

<a name="compliance"></a>
## 12. КОМПЛАЕНС И ЮРИДИЧЕСКИЕ АСПЕКТЫ

### Legal Structure

Platform Entity:
- Jurisdiction: Delaware, USA (or Cayman Islands)
- Structure: LLC or Foundation
- Purpose: Technology platform operator

Foundation Entity:
- Jurisdiction: Israel or EU (Netherlands)
- Structure: Non-profit 501(c)(3) equivalent
- Purpose: Charitable activities

### Regulatory Compliance

Securities Law:
- NFT miners = utility tokens, NOT securities
- No promises of guaranteed returns
- Dynamic reward formulas (based on BTC network)
- Educational disclaimers on all pages
- No ROI marketing claims

AML/KYC:
- KYC required for withdrawals > $1,000
- Enhanced KYC for > $10,000
- Transaction monitoring
- Suspicious activity reporting
- Sanctions screening (OFAC compliance)

Consumer Protection:
- Clear Terms of Service
- Privacy Policy (GDPR compliant)
- Cookie consent
- Right to be forgotten
- Data export functionality

Tax Compliance:
- User tax reporting tools
- 1099 forms (US users)
- Transaction history exports
- Partnership with tax software (CoinTracker)

### Risk Disclosures

User Agreements Must Include:
1. Cryptocurrency volatility risks
2. Mining profitability not guaranteed
3. Maintenance costs may exceed rewards
4. Platform operational risks
5. Smart contract risks
6. Regulatory changes
7. No FDIC/SIPC insurance

Platform Safeguards:
- Reserve fund for operational continuity
- Insurance coverage (cyber, D&O)
- Multi-sig wallets
- Regular security audits
- Incident response plan

### Intellectual Property

Trademarks:
- "Take Your Token" ® (pending)
- "TYT" ® (pending)
- Logo and brand assets

Patents:
- Consider utility patents for:
  - Charity Mint mechanism
  - Service Button discount system
  - veTYT governance model

Open Source:

- Smart contracts (audited and verified)
- API documentation
- SDK/libraries

### Data Protection (GDPR)

User Rights:
- Right to access data
- Right to rectification
- Right to erasure
- Right to data portability
- Right to object

Implementation:
- Data encryption (at rest and in transit)
- Minimal data collection
- Pseudonymization where possible
- Regular security assessments
- DPO appointed

### Foundation Governance

Board Structure:
- 5-7 board members
- Medical professionals (2)
- Crypto/tech experts (2)
- Community representatives (2-3)
- Independent auditor

Transparency:
- Public financial statements (annual)
- Donation tracking (real-time)
- Grant decisions (documented)
- Impact metrics (quarterly)

Audit:
- Annual financial audit (Big 4 firm)
- Smart contract audit (Trail of Bits, etc.)
- Compliance audit (KYC/AML)

# TAKE YOUR TOKEN v2 — MASTER BLUEPRINT

Version: 2.0.0
Date: December 10, 2024
Status: Agent-Ready Technical Specification
Purpose: Complete architecture for AI agents, developers, legal, and investors

---

## 🎯 EXECUTIVE SUMMARY

TakeYourToken (TYT) is a Web3 platform combining:

1. **NFT Bitcoin Mining** - Digital miners earning daily BTC rewards
2. **Crypto Academia** - Educational platform with certification
3. **Children's Brain Cancer Foundation** - Transparent charity funded by platform fees

Unique Value: Every transaction supports pediatric brain tumor research while users earn BTC through NFT miners and learn Web3 fundamentals.

Branding: Owl Warrior ecosystem with knight/shield/sword motif symbolizing wisdom, protection, and power.

## 🏗️ THREE PILLARS ARCHITECTURE

```
┌──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────┐
│                    TYT ECOSYSTEM v2                         │
├──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────┤
│                                                             │
│  ┌──────────────────────────────┐  ┌──────────────────────────────┐
┌──────────────────────────────┐   │
│  │  PILLAR 1    │  │  PILLAR 2    │  │  PILLAR 3    │      │
│  │              │  │              │  │              │
│  │  NFT Mining  │  │  Academy     │  │  Foundation  │      │
│  │  & Economy   │  │  Education   │  │  Charity     │      │
│  │              │  │              │  │              │
│  └─────────────┬────────────────┘  └─────────────┬────────────────┘
└────────────┬─────────────────┘   │
│         │              │              │
│         └────────────────────────────────────┴────────────────────────────────────┘
│
│                  │                    │
│               ┌────────────▼───────────┐                │
│               │  TYT TOKEN  │           │
│               │  (Solana)   │          │
│               └────────────────────────┘                │
│                                         │
└──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────┘
```

## 📊 PILLAR 1: NFT MINING & TOKEN ECONOMY

### 1.1 Core Concept

Users purchase **NFT Digital Miners** that generate daily BTC rewards. Unlike physical mining:
- No hardware management
- No electricity bills (handled by platform)
- No noise/heat/maintenance
- Tradeable on marketplace
- Upgradeable hashrate and efficiency

### 1.2 NFT Miner Specification

Contract: ERC-721 (Polygon) or TRC-721 (TRON)

Metadata Structure:
```json
{
  "tokenId": "uint256",
  "powerTH": "120.5",
  "efficiencyWTH": "28.5",
  "region": "USA_DC_01",
  "maintenanceRate": "0.065",
  "createdAt": "timestamp",
  "lastUpgradeAt": "timestamp",
  "tier": "ASIC_S19_XP",
  "ownerDiscountRef": "address"
}
```

Key Parameters:
- `powerTH` - Hashrate in TH/s (e.g., 120.5 TH/s)
- `efficiencyWTH` - Power consumption per TH (e.g., 28.5 W/TH)
- `maintenanceRate` - Daily service fee (USD/TH/day)
- `region` - Data center location
- `tier` - Miner model (impacts efficiency upgrade ceiling)

### 1.3 Rewards Calculation Engine

Daily Rewards Formula:

```javascript
// Step 1: Calculate gross BTC
const networkDifficulty = getNetworkDifficulty(); // from blockchain
const btcPrice = getBTCPrice(); // from oracle
const blockReward = 6.25; // halving adjusted

const grossBTC = (powerTH * 86400 * blockReward) /
                (networkDifficulty * 2^32);

// Step 2: Calculate electricity cost
const dailyKWH = (powerTH * efficiencyWTH * 24) / 1000;
const electricityCostUSD = dailyKWH * electricityRateUSD;

// Step 3: Calculate service fee
const serviceFeeUSD = powerTH * maintenanceRate;

// Step 4: Apply discounts
const discountPercent = calculateDiscount(user);
const totalCostUSD = (electricityCostUSD + serviceFeeUSD) *
                    (1 - discountPercent/100);

// Step 5: Convert to BTC
const costBTC = totalCostUSD / btcPrice;

// Step 6: Net reward
const netBTC = grossBTC - costBTC;

// Step 7: Apply reinvest
const reinvestPercent = user.reinvestPercent;
const creditedBTC = netBTC * (1 - reinvestPercent/100);
const reinvestBTC = netBTC * reinvestPercent/100;
```

Reward Distribution Flow:
1. Daily cron job (00:00 UTC)
2. Snapshot all active miners
3. Calculate rewards per user

4. Deduct maintenance (if auto-pay enabled)
5. Credit net BTC to custodial wallet
6. Log transaction with Merkle proof
7. Emit `RewardDistributed` event

### 1.4 Maintenance & Discount System

Payment Options:
- **USDT** - Base price, no discount
- **TYT Token** - Up to 20% discount + tokens burned
- **BTC** - Base price (converted from rewards)

Discount Tiers (based on TYT balance + coverage days):

Tier	TYT Balance	Days Coverage	Discount
Bronze	1,000+	30+	2%
Silver	5,000+	60+	5%
Gold	20,000+	90+	9%
Platinum	50,000+	180+	13%
Diamond	200,000+	365+	18%

Service Button Bonus: Daily click gives additional 3% discount for 24h.

Auto-Pay Logic:
```javascript
async function processMaintenanceFee(userId, minerId) {
  const user = await getUser(userId);
  const miner = await getMiner(minerId);

  // Calculate daily fee
  const feeUSD = calculateMaintenanceFee(miner);

  // Check payment preference
  if (user.preferredPayment === 'TYT') {
    const tytAmount = feeUSD / getTYTPrice();
    const discount = calculateDiscount(user);
    const finalTYT = tytAmount * (1 - discount/100);

    // Deduct and burn
    await deductTYT(user, finalTYT);
    await burnTYT(finalTYT);
    await emitBurnEvent(finalTYT);
  } else {
    // Pay in USDT/BTC
    await deductBalance(user, feeUSD, currency);
  }

  // Update miner status
  miner.lastMaintenancePaid = Date.now();
  await updateMiner(miner);
}
```

### 1.5 Marketplace

Features:
- List miner for sale (fixed price or auction)
- Buy instantly
- Place bids

- Cancel listing
- Creator royalties (2%)
- Platform fee (3% - goes to foundation)

Smart Contract:
```solidity
contract TYTMarketplace {
    struct Listing {
        uint256 tokenId;
        address seller;
        uint256 price;
        uint256 expiresAt;
        bool isAuction;
    }

    mapping(uint256 => Listing) public listings;

    function list(uint256 tokenId, uint256 price, bool auction) external;
    function buy(uint256 tokenId) external payable;
    function bid(uint256 tokenId) external payable;
    function cancelListing(uint256 tokenId) external;
}
```

### 1.6 Upgrade System

Hashrate Upgrade:
- Add TH/s to existing miner
- Cost: market rate + 10% premium
- Instant activation

Efficiency Upgrade:
- Reduce W/TH (lowers electricity cost)
- Tiers: Standard → Pro → Elite → Ultimate
- Cost: paid in TYT (burned)
- Limits based on miner tier

Reinvest Automation:
- User sets % of daily rewards to auto-buy TH
- Compounds over time
- Bonus: +5% extra TH on reinvest

### 1.7 Governance (veTYT)

Vote-Escrowed TYT:
- Lock TYT for 1 week to 4 years
- Receive veTYT (non-transferable)
- Voting power = amount × lock duration
- Decay over time

Governance Proposals:
- Adjust discount curve
- Change maintenance rates
- Burn schedule modifications
- Foundation allocation percentage
- New miner tier introductions

Proposal Flow:
1. Create proposal (requires 10,000 veTYT)
2. 3-day discussion period

3. 7-day voting period
4. 4% quorum required
5. 60% approval threshold
6. 2-day timelock
7. Execution by multisig

## 🎓 PILLAR 2: DIGITAL-INTERACTIVE-TECHNOLOGY-BLOCKCHAIN ACADEMIA

### 2.1 Mission

Goal: Educate 1,000,000+ users on Web3, crypto, mining, NFTs, security, and blockchain technology through gamified, interactive learning.

Differentiation: First mining platform with integrated academy - users learn while earning.

### 2.2 Course Structure

Tracks:

1. **Blockchain Fundamentals**
   - What is blockchain
   - Bitcoin basics
   - Ethereum and smart contracts
   - Consensus mechanisms
   - Mining economics

2. **Wallet Security**
   - Public/private keys
   - Seed phrases
   - Hardware wallets
   - Phishing protection
   - 2FA and security best practices

3. **NFT & Digital Assets**
   - NFT standards (721, 1155)
   - Metadata and IPFS
   - Marketplaces
   - Royalties
   - Utility NFTs

4. **DeFi & Trading**
   - DEX vs CEX
   - Liquidity pools
   - Staking and farming
   - Risk management
   - Tax implications

5. **Mining Deep Dive**
   - PoW vs PoS
   - Hashrate and difficulty
   - Pool mining
   - Profitability calculations
   - Energy efficiency

6. **Smart Contract Development**
   - Solidity basics
   - Contract deployment
   - Security audits

  - Testing frameworks
  - Real-world dApps

### 2.3 Gamification System

Owl Warrior Ranks:

Rank	XP Range	Icon	Benefits
Worker	0-99	🦉	Academy access
Academic	100-299	📚	+2% discount bonus
Diplomat	300-699	🤝	Priority support
Peacekeeper	700-1499	🛡️	Early feature access
Warrior	1500+	⚔️	Governance bonus, VIP status

XP Earning:
- Complete lesson: 10 XP
- Pass quiz: 20 XP
- Earn certificate: 50 XP
- Refer friend to academy: 30 XP
- Contribute content: 100 XP

Achievements:
- First miner purchased
- 100 days streak
- Marketplace veteran (10 trades)
- Governance participant
- Foundation donor

### 2.4 Certification System

Certificates (Soulbound NFTs):
- Non-transferable
- Stored on-chain
- Verifiable by employers/platforms
- Linked to wallet address

Certificate Tiers:
- Bronze (complete 1 track)
- Silver (complete 3 tracks)
- Gold (complete all 6 tracks)
- Platinum (Gold + contribute content)

### 2.5 Content Delivery

Formats:
- Video lessons (5-10 min each)
- Interactive quizzes
- Hands-on exercises
- Simulations (e.g., mining calculator)
- Live webinars
- Community discussions

Languages: EN, ES, FR, DE, PT, RU, ZH, JP, KO, AR, HE

Accessibility:
- Mobile-optimized
- Offline mode

- Subtitles
- Transcripts
- Adjustable playback speed

### 2.6 Tech Stack (Academy)

Backend:
```
AcademyService (NestJS)
├── CourseController
├── ProgressTracker
├── QuizEngine
├── CertificateMinter (SBT)
├── XPCalculator
└── ContentCMS (Strapi)
```

Database Schema:
```sql
CREATE TABLE courses (
  id UUID PRIMARY KEY,
  title VARCHAR(255),
  description TEXT,
  track VARCHAR(50),
  order INT,
  xp_reward INT,
  content_url TEXT,
  quiz_id UUID
);

CREATE TABLE user_progress (
  user_id UUID,
  course_id UUID,
  started_at TIMESTAMP,
  completed_at TIMESTAMP,
  quiz_score INT,
  xp_earned INT,
  PRIMARY KEY (user_id, course_id)
);

CREATE TABLE certificates (
  id UUID PRIMARY KEY,
  user_id UUID,
  token_id VARCHAR(100),
  tier VARCHAR(20),
  issued_at TIMESTAMP,
  blockchain VARCHAR(20),
  tx_hash VARCHAR(100)
);
```

## ❤️ PILLAR 3: CHILDREN'S BRAIN CANCER FOUNDATION

### 3.1 Mission Statement

TYT Children's Brain Cancer Research & Support Foundation is a transparent, crypto-native charity dedicated to:
- Funding breakthrough research in pediatric neuro-oncology
- Supporting families with travel, housing, and care costs
- Advancing early detection and treatment technologies
- Building partnerships with leading medical institutions

Symbolism: The Owl-Knight shield with inverted sword forms the gold childhood cancer awareness ribbon.

### 3.2 Revenue Streams to Foundation

Automatic Allocations:

Source	Allocation	Annual Estimate (at scale)
NFT miner sales	1%	$500K
Marketplace fees	3%	$300K
Maintenance payments	1%	$800K
Reinvest operations	1%	$200K
Charity Mint (from burns)	25% of burned TYT	$400K
Direct donations	100%	$300K
TOTAL		**$2.5M+/year**

Charity Mint Mechanism:
- When TYT is burned, 25% of USD-equivalent is minted back as "Charity TYT"
- Charity TYT goes directly to Foundation wallet
- Cannot be sold, only used for grants/expenses
- Creates deflationary + charitable loop

### 3.3 Foundation Structure

Legal Entity: 501(c)(3) Non-Profit (USA) or EU Foundation equivalent

Governance:
- Board of Directors (5-7 members)
- Scientific Advisory Board (3-5 pediatric oncologists)
- Community Council (veTYT holders)

Policies:
- Grantmaking
- Conflict of Interest
- Whistleblower
- Sanctions/Geo-blocks
- Data Privacy (HIPAA-aligned)

### 3.4 Grant Programs

Research Grants:
- Imaging technologies (MRI, fMRI)
- Genomics and targeted therapies
- Immunotherapy trials
- Clinical outcome studies

Family Support:
- Travel assistance
- Housing near treatment centers
- Rehabilitation costs
- Caregiver stipends

Equipment Grants:
- Hospital equipment purchases
- Lab instrument funding
- Telemedicine infrastructure

Grant Lifecycle:
1. RFP announcement
2. Application submission
3. Scientific review
4. Board approval
5. Milestone-based disbursement
6. Quarterly reporting
7. Impact audit

### 3.5 Transparency & Reporting

On-Chain Transparency:
- Public wallet addresses
- All transactions visible
- Proof-of-Use for grants (IPFS receipts)
- Merkle proofs for disbursements

Reports:
- Monthly donation feed
- Quarterly impact summary
- Annual comprehensive report
- Patient impact stories (anonymized)

Metrics Tracked:
- Total raised
- Total disbursed
- Number of grants
- Number of families helped
- Research publications funded
- Clinical trials supported

### 3.6 Tech Architecture (Foundation)

Smart Contracts:
```solidity
contract FundSplitter {
    address public foundationWallet;

    function splitFees(
        uint256 amount,
        string memory source
    ) external {
        uint256 foundationShare;

        if (source == "NFT_SALE") {
            foundationShare = amount * 1 / 100;
        } else if (source == "MARKETPLACE") {
            foundationShare = amount * 3 / 100;
        } else if (source == "MAINTENANCE") {
            foundationShare = amount * 1 / 100;
        }

        safeTransfer(foundationWallet, foundationShare);
        emit FundAllocated(source, foundationShare);
    }
```

}

contract CharityMint {
    function mintFromBurn(uint256 burnedAmount) external {
        uint256 charityAmount = burnedAmount * 25 / 100;
        TYTToken.mint(foundationWallet, charityAmount);
        emit CharityMinted(charityAmount);
    }
}
```

Backend Services:
```
FundService
├── DonationTracker
├── GrantManager
├── WalletMonitor
├── ReportGenerator
└── PartnerIntegration
```

Public Dashboard (`/foundation`):
- Live treasury balance
- Recent donations
- Active grants
- Impact metrics
- Partner hospitals
- Donate widget

3.7 Partnership Strategy

Target Partners:
- Children's hospitals (USA, Israel, EU)
- Research universities (MIT, Stanford, Technion, Weizmann)
- Cancer research foundations
- Patient advocacy groups
- Medical device companies

Partnership Benefits:
- Direct funding
- Data collaboration (privacy-compliant)
- Equipment grants
- Co-branded campaigns
- Community engagement

🔧 TECHNICAL ARCHITECTURE

4.1 System Overview

```
┌─────────────────────────────────────────────────────────────
─────────────────────────────────┐
│                    CLIENT LAYER                    │
├──────────────────────────────┬─────────────────────────────┬──────────────────
────┬──────────────────────┤

```
│  Web App     │  iOS App     │ Android App  │  Admin     │
│  (Next.js)   │  (RN)        │  (RN)        │  Panel     │
└──────────────┬──────────────┴──────────────┬──────────────┴──────────────┬───────────
──────┴──────────────┬──────────────┘
       │              │              │              │
└─────────────────────────────┴─────────────────────────────┴───────────────────────────
───┘
                   │
       ┌──────────────────────────────▼──────────────────────────────┐
       │      API GATEWAY              │
       │      (GraphQL + REST)         │
       └──────────────────────────────┬──────────────────────────────┘
                   │
┌──────────────────────────────▼───────────────────────────────────────────────┐
       │          BACKEND SERVICES                  │
├──────────────────────────────────────────────────────────────────────────────┤
       │  AuthService                          │
       │  MinerService                         │
       │  RewardsEngine                          │
       │  MaintenanceService                       │
       │  MarketplaceService                      │
       │  WalletService                        │
       │  AcademyService                         │
       │  FundService                          │
       │  GovernanceService                        │
       │  NotificationService                    │
└──────────────────────────────┬───────────────────────────────────────────────┘
                   │
       ┌──────────────────────────────▼──────────────────────────────┐
       │      DATA LAYER               │
       ├──────────────────────────────┬───────────────────────────────┤
       │  PostgreSQL  │   Redis         │
       │  TimescaleDB │   S3/IPFS        │
       └──────────────────────────────┴───────────────────────────────┘
                   │
       ┌──────────────────────────────▼──────────────────────────────────────────┐
       │      BLOCKCHAIN LAYER                │
       ├─────────────────────────────────────────────────────────────────────────┤
       │  Polygon (NFTs, veTYT)                │
       │  Solana (TYT Token)                 │
       │  Bitcoin (Rewards)                 │
       │  Indexer (The Graph)                │
       └─────────────────────────────────────────────────────────────────────────┘
```

### 4.2 Backend Services Detail

Tech Stack: NestJS (Node.js), TypeScript, PostgreSQL, Redis, Kafka

Core Services:

1. **AuthService**
   - JWT authentication
   - OAuth2 (Google, Twitter)
   - Web3 wallet connect
   - 2FA (TOTP)
   - Session management

2. **MinerService**
   - NFT minting
   - Metadata management
   - Upgrade processing
   - Transfer tracking
   - Performance analytics

3. **RewardsEngine**
   - Daily reward calculation
   - Network difficulty tracking
   - BTC price oracle
   - Merkle tree generation
   - Proof verification

4. **MaintenanceService**
   - Fee calculation
   - Auto-pay processing
   - Discount application
   - TYT burning
   - Payment history

5. **MarketplaceService**
   - Listing management
   - Order matching
   - Escrow handling
   - Fee distribution
   - Royalty calculation

6. **WalletService**
   - Balance management
   - Transaction ledger
   - Multi-sig operations (Fireblocks/Qredo)
   - Withdrawal processing
   - Deposit detection

7. **AcademyService**
   - Course management
   - Progress tracking
   - Quiz engine
   - XP calculation
   - Certificate minting

8. **FundService**
   - Donation tracking
   - Grant management
   - Wallet monitoring
   - Report generation
   - Partner integration

9. **GovernanceService**
   - Proposal creation
   - Voting logic
   - veTYT calculation

   - Timelock execution
   - Snapshot integration

10. **NotificationService**
    - Email (SendGrid)
    - Push (FCM)
    - In-app notifications
    - Webhook triggers

### 4.3 Database Schema (PostgreSQL)

Key Tables:

```sql
-- Users
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email VARCHAR(255) UNIQUE NOT NULL,
  wallet_address VARCHAR(42),
  kyc_status VARCHAR(20),
  vip_tier VARCHAR(20),
  referral_code VARCHAR(20),
  created_at TIMESTAMPTZ DEFAULT now()
);

-- NFT Miners
CREATE TABLE miners (
  id UUID PRIMARY KEY,
  token_id VARCHAR(100) UNIQUE,
  owner_id UUID REFERENCES users(id),
  power_th DECIMAL(10,2),
  efficiency_w_th DECIMAL(10,2),
  region VARCHAR(50),
  maintenance_rate DECIMAL(10,6),
  tier VARCHAR(50),
  created_at TIMESTAMPTZ,
  last_upgrade_at TIMESTAMPTZ
);

-- Rewards
CREATE TABLE rewards (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  miner_id UUID REFERENCES miners(id),
  date DATE,
  gross_btc DECIMAL(18,8),
  cost_btc DECIMAL(18,8),
  net_btc DECIMAL(18,8),
  reinvest_btc DECIMAL(18,8),
  credited_btc DECIMAL(18,8),
  proof_hash VARCHAR(66)
);

-- Maintenance Payments
CREATE TABLE maintenance_payments (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  miner_id UUID REFERENCES miners(id),
  amount DECIMAL(18,8),
  currency VARCHAR(10),
  discount_percent DECIMAL(5,2),
  tyt_burned DECIMAL(18,8),
  paid_at TIMESTAMPTZ
);

-- Marketplace Listings
CREATE TABLE marketplace_listings (
  id UUID PRIMARY KEY,
  miner_id UUID REFERENCES miners(id),
  seller_id UUID REFERENCES users(id),
  price DECIMAL(18,8),
  currency VARCHAR(10),
  is_auction BOOLEAN,
  expires_at TIMESTAMPTZ,
  status VARCHAR(20),
  created_at TIMESTAMPTZ
);

-- Academy Progress
CREATE TABLE academy_progress (
  user_id UUID REFERENCES users(id),
  course_id UUID REFERENCES courses(id),
  started_at TIMESTAMPTZ,
  completed_at TIMESTAMPTZ,
  quiz_score INT,
  xp_earned INT,
  PRIMARY KEY (user_id, course_id)
);

-- Foundation Donations
CREATE TABLE foundation_donations (
  id UUID PRIMARY KEY,
  user_id UUID,
  amount DECIMAL(18,8),
  currency VARCHAR(10),
  source VARCHAR(50), -- NFT_SALE, MARKETPLACE, DIRECT
  tx_hash VARCHAR(100),
  created_at TIMESTAMPTZ
);

-- Governance Proposals
CREATE TABLE governance_proposals (
  id UUID PRIMARY KEY,
  proposer_id UUID REFERENCES users(id),
  title VARCHAR(255),
  description TEXT,
  proposal_type VARCHAR(50),
  voting_starts_at TIMESTAMPTZ,
  voting_ends_at TIMESTAMPTZ,
  quorum_required DECIMAL(5,2),
  status VARCHAR(20),
  votes_for BIGINT,
  votes_against BIGINT
);
```

### 4.4 Smart Contracts

Polygon Contracts:

1. **MinerNFT.sol** (ERC-721)
```solidity
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/AccessControl.sol";

contract TYTMinerNFT is ERC721, AccessControl {
    bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");

    struct MinerData {
        uint256 powerTH;
        uint256 efficiencyWTH;
        string region;
        uint256 maintenanceRate;
        string tier;
    }

    mapping(uint256 => MinerData) public miners;
    uint256 private _tokenIdCounter;

    function mintMiner(
        address to,
        uint256 powerTH,
        uint256 efficiencyWTH,
        string memory region,
        uint256 maintenanceRate,
        string memory tier
    ) external onlyRole(MINTER_ROLE) returns (uint256) {
        uint256 tokenId = _tokenIdCounter++;
        _safeMint(to, tokenId);

        miners[tokenId] = MinerData({
            powerTH: powerTH,
            efficiencyWTH: efficiencyWTH,
            region: region,
            maintenanceRate: maintenanceRate,
            tier: tier
        });

        emit MinerMinted(tokenId, to, powerTH);
        return tokenId;
    }

    function upgradeHashrate(
        uint256 tokenId,
        uint256 additionalTH
    ) external onlyRole(MINTER_ROLE) {
        miners[tokenId].powerTH += additionalTH;
        emit HashrateUpgraded(tokenId, miners[tokenId].powerTH);
    }

    function upgradeEfficiency(
        uint256 tokenId,
        uint256 newEfficiency
    ) external onlyRole(MINTER_ROLE) {
        require(newEfficiency < miners[tokenId].efficiencyWTH, "Must improve");
        miners[tokenId].efficiencyWTH = newEfficiency;
        emit EfficiencyUpgraded(tokenId, newEfficiency);
    }
}
```

2. **TYTMarketplace.sol**
```solidity
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

contract TYTMarketplace is ReentrancyGuard {
    struct Listing {
        uint256 tokenId;
        address seller;
        uint256 price;
        uint256 expiresAt;
        bool active;
    }

    mapping(uint256 => Listing) public listings;
    address public foundationWallet;
    uint256 public platformFeePercent = 3;

    event Listed(uint256 indexed tokenId, address seller, uint256 price);
    event Sold(uint256 indexed tokenId, address buyer, uint256 price);

    function list(
        uint256 tokenId,
        uint256 price,
        uint256 duration
    ) external {
        require(minerNFT.ownerOf(tokenId) == msg.sender, "Not owner");

        listings[tokenId] = Listing({
            tokenId: tokenId,
            seller: msg.sender,
            price: price,
            expiresAt: block.timestamp + duration,
            active: true
        });

        minerNFT.transferFrom(msg.sender, address(this), tokenId);
        emit Listed(tokenId, msg.sender, price);
    }

    function buy(uint256 tokenId) external payable nonReentrant {
        Listing memory listing = listings[tokenId];
        require(listing.active, "Not active");
        require(block.timestamp < listing.expiresAt, "Expired");
        require(msg.value >= listing.price, "Insufficient payment");

        // Calculate fees
        uint256 platformFee = (listing.price * platformFeePercent) / 100;
        uint256 sellerAmount = listing.price - platformFee;

        // Transfer fees to foundation
        (bool foundationSuccess,) = foundationWallet.call{value: platformFee}("");
        require(foundationSuccess, "Foundation transfer failed");

        // Pay seller
        (bool sellerSuccess,) = listing.seller.call{value: sellerAmount}("");
        require(sellerSuccess, "Seller transfer failed");

        // Transfer NFT
        minerNFT.transferFrom(address(this), msg.sender, tokenId);

        listings[tokenId].active = false;
        emit Sold(tokenId, msg.sender, listing.price);
    }
}
```

3. **veTYT.sol** (Vote-Escrowed TYT)
```solidity
pragma solidity ^0.8.20;

contract VotingEscrowTYT {
    struct LockedBalance {
        uint256 amount;
        uint256 end;
    }

    mapping(address => LockedBalance) public locked;

    uint256 public constant MAXTIME = 4 * 365 days;
    uint256 public constant WEEK = 7 days;

    function createLock(uint256 value, uint256 unlockTime) external {
        require(value > 0, "Zero value");
        require(unlockTime > block.timestamp, "Past time");
        require(unlockTime <= block.timestamp + MAXTIME, "Too long");

        locked[msg.sender] = LockedBalance({
            amount: value,
            end: (unlockTime / WEEK) * WEEK
        });

        tytToken.transferFrom(msg.sender, address(this), value);
        emit LockCreated(msg.sender, value, unlockTime);
    }

    function balanceOf(address user) external view returns (uint256) {
        LockedBalance memory lock = locked[user];
        if (lock.end <= block.timestamp) return 0;

        uint256 timeLeft = lock.end - block.timestamp;
        return (lock.amount * timeLeft) / MAXTIME;
    }
}
```

Solana Programs:

1. **TYT Token** (SPL Token) - Already deployed on pump.fun
2. **CharityMint Program** - Mints charity tokens from burns
3. **BurnScheduler Program** - Coordinates weekly burns

### 4.5 Infrastructure

Hosting: AWS/GCP
- EKS/GKE for Kubernetes

- RDS for PostgreSQL
- ElastiCache for Redis
- S3/GCS for storage
- CloudFront/Cloud CDN

DevOps:
- Terraform for IaC
- GitHub Actions for CI/CD
- ArgoCD for GitOps
- Datadog/Grafana for monitoring
- Sentry for error tracking

Security:
- Fireblocks/Qredo for custody
- Certik audit for smart contracts
- SOC 2 Type II compliance
- Bug bounty program (Immunefi)

## 💰 TOKENOMICS

### 5.1 TYT Token (Solana SPL)

Already Deployed: https://pump.fun/APadkPpjonaLBpLYDzKB6753QQU3s8VZhEtkvLgrpump

Initial Supply: TBD (from pump.fun)
Max Supply: Fixed (deflationary through burns)

### 5.2 Utility

1. **Maintenance Payments** - Up to 20% discount
2. **Marketplace Currency** - Only accepted currency
3. **Upgrades** - Pay for efficiency upgrades
4. **Governance** - Lock for veTYT voting power
5. **Academy Rewards** - Earn for completing courses
6. **VIP Tiers** - Holding requirements
7. **Charity Staking** - Stake to donate yield to foundation

### 5.3 Burn Mechanisms

Sources of Burns:
- 100% of maintenance paid in TYT
- 100% of upgrade fees paid in TYT
- 50% of marketplace fees
- Governance proposal deposits (if rejected)

Burn Schedule: Weekly on Sundays 00:00 UTC

Transparency: Public transaction + detailed report

### 5.4 Charity Mint

Mechanism: 25% of burned USD-equivalent minted as "Charity TYT"

Example:
```
Week 1 Burns: 100,000 TYT × $0.50 = $50,000
Charity Mint: $50,000 × 25% = $12,500 worth of TYT
Minted to: Foundation wallet
```

Result: 75,000 TYT removed from circulation permanently
        25,000 TYT created for charitable use
```

5.5 Distribution (Suggested)

- 30% - Public sale (pump.fun)
- 20% - Ecosystem rewards (academy, referrals)
- 15% - Team (4-year vest)
- 15% - Treasury (DAO-controlled)
- 10% - Liquidity
- 10% - Partners & advisors (2-year vest)

📱 USER EXPERIENCE

6.1 Web Application Screens

Public Pages:
1. Landing - Hero, features, tokenomics, academy, foundation
2. About - Team, mission, roadmap
3. Foundation - Live stats, grants, donate
4. Academy - Course catalog, free preview
5. Docs - Whitepaper, FAQs, API docs

Authenticated Pages:
1. Dashboard - Portfolio overview, daily rewards, news
2. My Miners - Grid view, filters, performance charts
3. Marketplace - Browse, filter, buy, sell
4. Rewards - History, pending, withdraw
5. Wallet - Balances, deposit, withdraw
6. Maintenance - Auto-pay settings, history, discounts
7. Academy - My courses, progress, certificates
8. Governance - Proposals, voting, veTYT
9. Referrals - Code, stats, earnings
10. Foundation - My donations, impact
11. Settings - Profile, security, preferences

6.2 Mobile Applications (iOS/Android)

React Native with shared codebase

Key Features:
- Push notifications (rewards, maintenance due, governance)
- Biometric authentication
- QR code scanner (for deposits)
- Deep links
- Offline mode (cached data)

Screens: Same as web with mobile-optimized UI

6.3 Design System

Already Implemented: See `DESIGN_SYSTEM.md`

Key Elements:
- Gold/Navy/Neon color palette
- Owl Warrior branding
- Glassmorphic cards
```

- Gradient buttons
- Rank badges
- Shield progress meters

---

## 🗺️ DEVELOPMENT ROADMAP

### Phase 0: Sandbox (Weeks 1-3) ✅ IN PROGRESS

**Goal**: Proof of concept with test data

**Deliverables**:
- [x] Landing page with branding
- [x] Design system
- [x] Database schema
- [ ] Test NFT contracts (Polygon Mumbai)
- [ ] Reward simulator (fake BTC data)
- [ ] Basic dashboard UI
- [ ] Foundation page

**Team**: 2 full-stack developers

### Phase 1: MVP (Weeks 4-11)

**Goal**: Launch with real custody and rewards

**Deliverables**:
- [ ] Production smart contracts (audited)
- [ ] Custody integration (Fireblocks/Qredo)
- [ ] Real BTC reward engine
- [ ] Maintenance autopay
- [ ] Marketplace (list/buy)
- [ ] Wallet (deposit/withdraw)
- [ ] Weekly burn automation
- [ ] KYC/AML integration
- [ ] Foundation splitter contract
- [ ] Admin panel
- [ ] Mobile apps (MVP)

**Team**: 4 developers, 1 designer, 1 DevOps, 1 legal

**Budget**: $150K-$200K

### Phase 2: Full Platform (Weeks 12-23)

**Goal**: Complete all three pillars

**Deliverables**:
- [ ] Academy (all 6 tracks)
- [ ] Certificate minting (SBT)
- [ ] veTYT governance
- [ ] Multi-chain withdrawals (Lightning, Liquid, TON, etc.)
- [ ] VIP tiers
- [ ] Referral program
- [ ] Advanced analytics
- [ ] Foundation dashboard
- [ ] Grant management portal
- [ ] Charity staking

**Team**: 6 developers, 2 designers, 1 DevOps, 1 legal, 1 community manager

**Budget**: $300K-$400K

### Phase 3: Scale & Partnerships (Months 6-12)

**Goal**: 10,000+ users, $2M+ to foundation

**Deliverables**:
- [ ] Hospital partnerships (3-5 institutions)
- [ ] First research grant awarded
- [ ] Mobile app v2 (advanced features)
- [ ] Expanded academy content
- [ ] Multiple languages
- [ ] White-label solution for partners
- [ ] Insurance product integration
- [ ] Miner avatars (cosmetic NFTs)

**Team**: 10+ employees

**Budget**: $1M+

---

## ⚖️ LEGAL & COMPLIANCE

### 7.1 Key Principles

1. **NFT = Service Access**: Miners are NOT securities, they represent access to mining services
2. **No ROI Promises**: Rewards are variable and not guaranteed
3. **Dynamic Pricing**: Market-based, no fixed returns
4. **Full Disclosure**: All risks documented
5. **Geo-Restrictions**: Block high-risk jurisdictions

### 7.2 Required Documents

- [ ] Terms of Service
- [ ] Privacy Policy
- [ ] Risk Disclosures
- [ ] Whitepaper (technical)
- [ ] Foundation Bylaws
- [ ] Grant Policy
- [ ] AML/KYC Policy
- [ ] Data Protection Policy (GDPR)

### 7.3 KYC/AML

**Provider**: Sumsub, Onfido, or Jumio

**Triggers**:
- Withdrawals > $1,000/day
- Marketplace sales > $5,000/month
- VIP tier upgrades
- Foundation donations > $10,000

**Data Collected**:
- Full name
- Date of birth
- Government ID

- Proof of address
- Selfie verification

### 7.4 Restricted Countries

Initial restrictions:
- USA (until legal clarity)
- China
- North Korea
- Iran
- Syria
- Cuba

### 7.5 Foundation Legal

**Structure**: 501(c)(3) or equivalent

**Jurisdiction**: USA (Delaware) or Israel

**Tax Benefits**: Donations tax-deductible in applicable jurisdictions

**Transparency**: Annual Form 990 (USA) or equivalent

---

## 📊 BUSINESS MODEL

### 8.1 Revenue Streams

1. **Marketplace Fees**: 3% of sales
2. **Miner Sales**: 10% margin on initial sales
3. **Upgrade Fees**: 15% margin
4. **VIP Memberships**: $50-500/month tiers
5. **White-label Licensing**: Future revenue

### 8.2 Cost Structure

1. **BTC Rewards**: 70% of user payments
2. **Electricity**: 15%
3. **Service/Maintenance**: 5%
4. **Development**: 5%
5. **Operations**: 3%
6. **Marketing**: 2%

### 8.3 Unit Economics (Example)

**Average Miner**:
- Hashrate: 100 TH/s
- Sale Price: $5,000
- Daily BTC Reward: ~0.0005 BTC ($15)
- Daily Maintenance: $6.50
- User Net: $8.50/day
- Platform Take: $0.65/day (10% of maintenance)
- Foundation: $0.20/day

**At 1,000 Miners**:
- Daily Foundation: $200
- Annual Foundation: $73,000

**At 10,000 Miners**:

- Daily Foundation: $2,000
- Annual Foundation: $730,000

**At 100,000 Miners** (Target Year 3):
- Daily Foundation: $20,000
- Annual Foundation: $7,300,000

---

## 🎯 GO-TO-MARKET STRATEGY

### 9.1 Launch Phases

**Phase 1: Soft Launch** (100 beta users)
- Invite-only
- Feedback collection
- Bug fixes

**Phase 2: Public Launch** (10,000 users target)
- PR campaign
- Influencer partnerships
- Social media ads

**Phase 3: Scale** (100,000 users target)
- Strategic partnerships
- Academy certifications promoted
- Foundation impact stories

### 9.2 Marketing Channels

1. **Crypto Twitter**: Engage with mining/NFT communities
2. **YouTube**: Educational content about mining
3. **TikTok**: Short-form academy lessons
4. **Reddit**: r/CryptoMining, r/NFT, r/ethereum
5. **Telegram**: Community channel (already created)
6. **Discord**: Support and governance discussions
7. **Email**: Newsletter with market updates

### 9.3 Partnerships

**Target Partners**:
- Crypto exchanges (for TYT listing)
- Mining pools (for data integration)
- Hardware manufacturers (for branding)
- Educational platforms (for academy content)
- Hospitals/foundations (for charitable work)

### 9.4 Community Building

- Weekly AMAs
- Monthly governance calls
- Quarterly impact reports
- Annual charity gala (Night of the Owls)
- Bug bounties
- Content creator grants

---

## 🔒 SECURITY & RISK MANAGEMENT

### 10.1 Smart Contract Security

- [ ] Formal verification
- [ ] Certik audit
- [ ] OpenZeppelin contracts
- [ ] Multisig admin
- [ ] Timelock on upgrades
- [ ] Bug bounty ($100K pool)

### 10.2 Custody Security

- Fireblocks/Qredo MPC
- Hardware security modules (HSM)
- Multi-approval policies
- Cold storage for majority of funds
- Hot wallet limits
- Daily reconciliation

### 10.3 Operational Security

- SOC 2 Type II compliance
- Penetration testing (quarterly)
- Employee background checks
- 2FA mandatory for all staff
- Incident response plan
- Insurance coverage ($5M+ cyber)

### 10.4 Risk Register

| Risk | Likelihood | Impact | Mitigation |
|------|-----------|--------|------------|
| Smart contract exploit | Low | Critical | Audits, bug bounty, insurance |
| Custody hack | Low | Critical | MPC, insurance, cold storage |
| Regulatory action | Medium | High | Legal counsel, compliance |
| BTC price crash | High | Medium | Dynamic formulas, disclaimers |
| Low user adoption | Medium | Medium | Marketing, partnerships |
| Foundation mismanagement | Low | High | Board oversight, transparency |

---

## 📈 SUCCESS METRICS

### 11.1 Platform KPIs

**Year 1 Targets**:
- 10,000 registered users
- 5,000 active miners
- $5M in NFT sales
- $1M to foundation
- 50,000 academy enrollments
- 5,000 certificates issued

**Year 2 Targets**:
- 50,000 registered users
- 25,000 active miners
- $25M in NFT sales
- $5M to foundation
- 250,000 academy enrollments
- 25,000 certificates issued

**Year 3 Targets**:
- 200,000 registered users
- 100,000 active miners
- $100M in NFT sales
- $20M to foundation
- 1,000,000 academy enrollments
- 100,000 certificates issued

### 11.2 Foundation KPIs

**Year 1**:
- 3 research grants awarded
- 50 families supported
- 1 equipment grant
- 10 partner hospitals

**Year 2**:
- 10 research grants
- 200 families supported
- 5 equipment grants
- 25 partner hospitals

**Year 3**:
- 30 research grants
- 1,000 families supported
- 20 equipment grants
- 50 partner hospitals

---

## 🚀 DEPLOYMENT CHECKLIST

### Pre-Launch

- [ ] Smart contracts audited
- [ ] Contracts deployed to mainnet
- [ ] Backend deployed to production
- [ ] Database migrations run
- [ ] Web app deployed
- [ ] Mobile apps submitted to stores
- [ ] Custody wallets configured
- [ ] KYC provider integrated
- [ ] Legal documents finalized
- [ ] Foundation entity registered
- [ ] Team training completed

### Launch Day

- [ ] Announce on social media
- [ ] Press release distributed
- [ ] Influencer posts go live
- [ ] Telegram/Discord announcements
- [ ] Email to waitlist
- [ ] Monitor systems
- [ ] Support team ready

### Post-Launch (Week 1)

- [ ] Collect feedback

- [ ] Fix critical bugs
- [ ] Optimize performance
- [ ] First burn event
- [ ] Weekly metrics report
- [ ] AMA with community

---

## 📞 SUPPORT & RESOURCES

### For Developers

- **GitHub**: [Repository URL]
- **Docs**: https://docs.takeyourtoken.app
- **API Reference**: https://api.takeyourtoken.app/docs
- **Discord**: Developer channel

### For Users

- **Help Center**: https://help.takeyourtoken.app
- **Email**: support@takeyourtoken.app
- **Telegram**: https://t.me/takeyourtoken
- **Twitter**: @takeyourtoken

### For Partners

- **Email**: partnerships@takeyourtoken.app
- **Press**: press@takeyourtoken.app

### For Foundation

- **Email**: foundation@takeyourtoken.app
- **Grant Applications**: https://foundation.takeyourtoken.app/grants

---

## 🦉 CONCLUSION

TYT is more than a mining platform - it's a complete Web3 ecosystem that:

1. **Generates value** through Bitcoin mining NFTs
2. **Educates users** through comprehensive academy
3. **Creates impact** by funding children's brain cancer research

**Unique Position**: First platform to combine mining, education, and charity in one transparent ecosystem.

**Mission**: Use Web3 technology to save children's lives while building a sustainable, profitable business.

**Vision**: Become the #1 platform for ethical Web3 mining and medical research funding.

---

**Built by the Owl Warriors. Protected by the Shield. Powered by the Sword.**

**For the children. For the future. For Web3.**

---

*This document is a living blueprint. Version 2.0.0 - Last updated: December 10, 2024*

*Prepared for AI agents, developers, legal counsel, investors, and partners.*

*All trademarks and logos are property of their respective owners.*

Ниже — **4 “agent-задания”** Каждый блок уже включает: scope, DoD, структуру папок, env, тесты, миграции, локальный запуск, staging deploy.

———

## 1) PROMPT — contracts-agent (EVM ядро v3.0)

```
ROLE: contracts-agent (Senior Solidity / Foundry)
REPO: https://github.com/takeyourtokenapp/tyt.app
GOAL: TYT v3.0 — заменить “заглушки” реальными on-chain контрактами
(Polygon Amoy -> Polygon mainnet)
CHAIN: Polygon (EVM)
FEE CANON (MUST):
- deposit_fee_total_bps = 1000 (10%)
- split inside fee_total: protocol=60%, charity=30%, academy=10%  (== 6% /
3% / 1% of amount)
All fee splits MUST be routed through FeeConfig profile keys.
NEVER copy GoMining branding/design/text 1:1.

SCOPE (DELIVERABLE CONTRACTS):
1) FeeConfig.sol
   - stores fee profiles by bytes32 key (e.g.,
keccak256("deposit.default"), "marketplace.default")
   - each profile: totalBps (0..2000), recipients[] (protocol, charity,
academy), splitBps[] summing to 10_000
   - supports role-based updates (DEFAULT_ADMIN_ROLE, FEE_SETTER_ROLE)
   - emits FeeProfileUpdated(key, totalBps, recipients, splitBps)

2) CharityVault.sol
   - receives ERC20 and native; tracks totals per token; categorized
sources (bytes32)
   - withdraw only TREASURY_ROLE (multisig address set at deploy)
   - emits DonationReceived(token, from, amount, sourceKey),
DonationWithdrawn(token, to, amount, reason)

3) MinerNFT.sol (ERC-721)
   - mint (admin or sale module later)
   - metadata: minerTypeId, powerHashrate (uint), level, isActive
   - upgrade function (owner or approved; emits MinerUpgraded)
   - emits MinerMinted(tokenId, owner, minerTypeId, initialPower)

4) RewardsMerkleRegistry.sol
   - stores daily merkle root (dateKey = uint32 YYYYMMDD or uint64 epoch-
day)
   - addRoot(dateKey, root) only REWARDS_PUBLISHER_ROLE
   - once set, cannot overwrite (immutability)
   - emits RootPublished(dateKey, root, uriOptional)
```

```
5) MinerMarketplace.sol
   - listing & buy for MinerNFT
   - fee taken from sale price using FeeConfig key "marketplace.default"
   - feeTotal = price * totalBps / 10_000
   - distribute feeTotal by splitBps to protocol/charity/academy
recipients
   - seller receives price - feeTotal
   - emits OrderCreated, OrderCancelled, OrderFilled with fee breakdown

OPTIONAL v3.0:
- Minimal veTYT stub is NOT required for v3.0 launch; can be v3.2

TECH STACK:
- Foundry preferred (forge)
- OpenZeppelin contracts
- Solidity 0.8.x

REPO CHANGES:
- Create /contracts/evm (or align with existing repo layout)
- Add:
  - forge config
  - scripts/deploy_amoy.s.sol
  - scripts/deploy_mainnet.s.sol (prepared but not executed)
  - test suite (unit + invariants for FeeConfig)
  - addresses registry: /contracts/evm/deployments/{amoy,polygon}.json

DEPLOY SEQUENCE:
1) FeeConfig
2) CharityVault (set treasury multisig)
3) MinerNFT
4) RewardsMerkleRegistry
5) MinerMarketplace (wire FeeConfig + MinerNFT + recipients)

ENV VARS:
- PRIVATE_KEY
- RPC_URL_AMOY
- RPC_URL_POLYGON
- TREASURY_MULTISIG
- PROTOCOL_TREASURY (ops wallet)
- CHARITY_TREASURY (CharityVault recipient or same vault)
- ACADEMY_TREASURY
- FEE_SETTER_ADMIN

DEFINITION OF DONE (DoD):
- forge test passes
- deploy script works on Amoy and outputs deployments json
- marketplace fee split exactly matches: protocol 60%, charity 30%,
academy 10% of feeTotal
- roots immutable (no overwrite)
- all critical state changes emit events
- README_contracts.md: how to deploy + verify + run tests

OUTPUT FORMAT:
- commit changes in a new branch: feat/v3-contracts-core
- provide list of files changed + how to run:
  - forge test
  - forge script deploy_amoy
```

---

2) PROMPT — backend-agent (Wallet + Gateway + Indexers + Rewards engine)

```
ROLE: backend-agent (Senior Node/NestJS, blockchain integrations)
REPO: https://github.com/takeyourtokenapp/tyt.app
GOAL: TYT v3.0 — заменить демо-API и заглушки на реальные: ledger,
депозиты, индексаторы, rewards-merkle, marketplace sync
CANON FEES (MUST):
- deposit_fee_total_bps = 1000 (10%)
- fee split inside fee_total: protocol=60%, charity=30%, academy=10%  (==
6%/3%/1% of deposit amount)
All money-moving flows must be recorded via double-entry ledger (journal
entries).

SERVICES (NestJS microservices; can be mono-repo apps/ packages):
A) wallet-service (SOURCE OF TRUTH for balances)
   - DB: Postgres
   - Tables:
     - accounts (user, protocol, charity, academy)
     - assets (USDT, USDC, etc.)
     - journal_entries (id, ts, reference, type, status)
     - journal_lines (entry_id, account_id, asset_id, debit, credit)
     - balances (materialized view or computed)
     - withdrawals (status workflow)
   - Endpoints:
     - GET /wallet/balance
     - GET /wallet/history?asset=...
     - POST /wallet/withdraw (request)
     - Internal: POST /wallet/internal/deposit-credit (from gateway/
indexer) - idempotent
   - Apply fee profile:
     - amount_user = amount - fee_total
     - credit user (amount_user)
     - credit protocol (amount*0.06)
     - credit charity (amount*0.03)
     - credit academy (amount*0.01)

B) blockchain-gateway-service
   - Purpose: chain adapters + sending tx + verifying confirmations
   - MVP v3.0: Polygon deposits for USDC/USDT (ERC20 Transfer)
   - Functions:
     - allocate deposit address per user (custodial or derived) [MVP can
be custodial single address + memo mapping if needed]
     - index incoming transfers (or consume indexer output)
     - submit withdrawals (Polygon ERC20 transfer) with signer key
(staging only)
   - Idempotency keys mandatory.

C) indexer-service
   - Uses ethers + RPC logs to sync:
     - MinerNFT events (mint/transfer/upgrade/status)
     - Marketplace events (order created/filled/cancelled)
     - RewardsMerkleRegistry RootPublished
   - Stores in DB for frontend read-model.
   - Reorg-safe strategy:
     - store lastProcessedBlock + confirmations threshold
     - allow backfill from N

D) rewards-engine-service
   - Daily cron:
     - read miners state (from indexer DB)
     - compute gross rewards share (MVP param dailyPool set in config)
     - apply maintenance fee / discounts (stub formula acceptable but
deterministic)
     - post credits into wallet-service (BTC-like “reward asset” ledger
entry)
     - build Merkle tree of (user, dateKey, amount, asset)
     - publish root on-chain via RewardsMerkleRegistry
     - store proofs JSON in storage (local/S3) and expose via API

E) charity-service
   - Reads wallet credits for charity & academy
   - Generates public reports:
     - totals by period
     - recent allocations
     - campaign objects (off-chain first)
   - Integrate CharityVault events (DonationReceived) into report feed.

DATABASE:
- Use Postgres + Prisma or TypeORM (pick one consistently)
- Provide migrations

SECURITY (MVP but real):
- Auth: JWT
- Admin roles
- Rate limit for withdraw endpoints
- Basic fraud controls:
  - min confirmations
  - withdrawal cooldown
  - max daily withdraw per user (config)

ENV VARS:
- DATABASE_URL
- JWT_SECRET
- POLYGON_RPC_URL (amoy + mainnet configs)
- CONTRACT_ADDRESSES (FeeConfig, CharityVault, MinerNFT, Marketplace,
RewardsRegistry)
- SIGNER_PRIVATE_KEY (staging only; prod placeholder)
- CONFIRMATIONS_REQUIRED
- DAILY_POOL_AMOUNT (MVP)
- STORAGE_PATH or S3_* (optional)

DoD:
- End-to-end flow on staging (Polygon Amoy):
  1) simulate deposit event -> wallet credits apply 6/3/1 + user net
  2) rewards cron generates merkle + publishes root on-chain
  3) indexer sees RootPublished
  4) API returns proof for user/date
- Unit tests for:
  - fee application
  - idempotency
  - merkle builder deterministic
- Docker compose for local services
- Branch: feat/v3-backend-rails
- Provide commands:
  - pnpm install
  - pnpm db:migrate
  - pnpm start:dev
  - pnpm test
```

⸻

3) PROMPT — frontend-agent (Next.js: мок → реальные endpoints модульно)

```
ROLE: frontend-agent (Senior Next.js / TypeScript)
REPO: https://github.com/takeyourtokenapp/tyt.app
GOAL: TYT v3.0 — заменить mock/stub данные на реальные запросы к backend,
без ломания UI.
RULE: migrate page-by-page, keeping existing design and brand.

PAGES TO “PUT ON WHEELS” (ORDER):
1) Auth + Profile
   - login/signup -> JWT
   - persist session

2) Wallet
   - show balances from GET /wallet/balance
   - history from GET /wallet/history
   - withdraw form -> POST /wallet/withdraw
   - show fee explanation (10% total; breakdown 6/3/1)

3) Miners
   - list user miners from indexer read-model endpoint
   - miner detail page: level, power, status, last rewards

4) Rewards
   - show daily accruals from wallet ledger
   - “Verify” panel:
     - fetch proof JSON from backend
     - verify merkle inclusion locally (use merkle lib)
     - show green check

5) Marketplace
   - list orders from indexer endpoint
   - buy/list actions:
     - call backend to prepare tx data (optional) OR directly interact
with contract via wagmi/viem
     - show fee breakdown
```

```
6) Foundation (Charity)
   - totals, recent donations, allocations feed
   - campaign cards (off-chain first)
   - link to on-chain tx hashes when available

TECH:
- Next.js App Router (if repo uses it), TypeScript
- Data fetching: react-query or SWR (choose one)
- Web3: wagmi + viem (Polygon)
- Environment config:
  - NEXT_PUBLIC_API_BASE
  - NEXT_PUBLIC_CHAIN_ID
  - NEXT_PUBLIC_CONTRACTS_JSON_URL (or embed addresses by env)

REQUIREMENTS:
- Create typed API client layer: /src/lib/api/*
- Add consistent loading/error states
- No tables with huge density; keep cards/blocks consistent with design
system
- Keep all “demo” fallback behind feature flag:
  - NEXT_PUBLIC_USE_MOCKS=false in staging/prod

DoD:
- Wallet flow works against staging backend
- Rewards page verifies merkle proof for a demo user (staging dataset)
- Marketplace list renders from indexer, not static JSON
- Build passes (pnpm build)
- Branch: feat/v3-frontend-real-api
- Provide short checklist of changed routes/components and how to run
locally:
  - pnpm dev
  - set envs
```

---

4) PROMPT — infra-agent (CI/CD, docker, staging/prod, GitHub discipline)

```
ROLE: infra-agent (Senior DevOps)
REPO: https://github.com/takeyourtokenapp/tyt.app
GOAL: TYT v3.0 — обеспечить “рельсы” разработки: reproducible local,
staging, CI/CD, secrets policy.

TARGETS:
- Local: docker compose runs backend services + postgres + redis (if used)
- Staging: deploy backend + frontend (Hostinger/VPS or alternative) + run
migrations
- Prod: prepared config (no private keys in repo)

DELIVERABLES:
1) /infra/docker-compose.yml
   - postgres
   - optional redis
   - backend services with env files
```

```
2) /infra/.env.example for each service
3) GitHub Actions:
   - ci.yml:
     - install
     - lint
     - typecheck
     - unit tests
     - contracts: forge test
   - deploy-staging.yml (on push to staging):
     - build artifacts
     - run db migrations
     - restart services
   (If Hostinger doesn't support actions deploy directly, implement SSH
deploy to VPS.)

4) Secrets model:
   - .env in server only
   - GitHub secrets for staging deploy (SSH_KEY, HOST, USER)
   - never commit keys

5) Observability (MVP):
   - request logging
   - health endpoints
   - optional: simple uptime check script

6) Branching rules doc:
   - main = prod
   - staging = testnet
   - feature branches
   - PR template includes DoD checklist

DoD:
- One-command local boot:
  - make up  (or pnpm infra:up)
  - make migrate
- CI passes on PR
- Staging deploy pipeline documented in README_DEPLOY.md
- Branch: feat/v3-infra-rails
- Provide explicit run commands for macOS
```

———

- **5-й блок**: “Integrator / release-manager prompt”

  ROLE: Integrator / Release-Manager (You control the build)

REPO: https://github.com/takeyourtokenapp/tyt.app

GOAL (plain language):

Take outputs from 4 agents (contracts, backend, frontend, infra), merge them safely into one working system, run staging, and confirm “the project drives” (end-to-end works). If something breaks — create clear fix-tasks for the right agent.

NON-NEGOTIABLE CANON (MUST):

- deposit_fee_total_bps = 1000 (10% deposit fee)
- split inside fee_total: protocol=60%, charity=30%, academy=10% (== 6% / 3% / 1% of deposit amount)
- All money movement must be double-entry recorded in wallet-service (journal entries)
- Staging uses Polygon Amoy (testnet) and fake funds only
- Never copy GoMining branding/UI/text 1:1

INPUTS (expected branches from other agents):

- feat/v3-contracts-core
- feat/v3-backend-rails
- feat/v3-frontend-real-api
- feat/v3-infra-rails

OUTPUTS (what you must produce):

1) A single merged staging branch with everything working together:
   - target branch: staging
2) A STAGING “Runbook” file in repo:
   - /docs/STAGING_RUNBOOK.md
3) A Release Checklist file:
   - /docs/RELEASE_CHECKLIST_V3.md
4) If any part fails, create “Fix tickets” as text in:
   - /docs/FIX_TASKS.md

   Each fix ticket must specify:
   - owner agent (contracts/backend/frontend/infra)
   - exact bug symptoms
   - exact file paths to inspect
   - clear acceptance criteria

WORKFLOW (step-by-step you must follow):

STEP 0 — Pull latest

- Ensure local repo is clean
- Fetch all branches

STEP 1 — Merge order (important)

Merge into staging in this exact order:

```
1) feat/v3-infra-rails        -> staging
2) feat/v3-contracts-core     -> staging
3) feat/v3-backend-rails      -> staging
4) feat/v3-frontend-real-api  -> staging
```

After each merge:

- run tests/build (commands below)
- if fail: STOP, write a fix ticket, revert the merge or open a hotfix branch

STEP 2 — Configure STAGING env (safe)

- Create staging env templates:
  - /infra/env/staging.backend.env.example
  - /infra/env/staging.frontend.env.example
- Ensure no secrets are committed
- Provide placeholders for:

```
  DATABASE_URL
  JWT_SECRET
  POLYGON_RPC_URL_AMOY
  SIGNER_PRIVATE_KEY_STAGING
  CONTRACT_ADDRESSES_JSON (or explicit addresses)
  NEXT_PUBLIC_API_BASE
  NEXT_PUBLIC_CHAIN_ID=80002 (Amoy)
```

  NEXT_PUBLIC_USE_MOCKS=false

STEP 3 — Deploy contracts to Amoy (staging)

- Use contracts deploy scripts from contracts-agent
- Save addresses to:

  /contracts/evm/deployments/amoy.json

- Confirm by reading chain:
  - FeeConfig exists
  - CharityVault exists
  - MinerNFT exists
  - RewardsMerkleRegistry exists
  - MinerMarketplace exists

STEP 4 — Bring up STAGING locally first (must)

Run locally with docker compose:

- start infra (postgres etc.)
- run backend migrations
- start backend services
- start frontend

STEP 5 — End-to-end checks (THE 5 POINTS)

You must verify these 5 items and record results in STAGING_RUNBOOK.md:

E2E-1: Login works

- Create a user
- Receive a JWT
- Session persists after refresh

E2E-2: Deposit credits apply fee split correctly (6/3/1)

- Simulate or trigger a deposit of 1000 USDC/USDT in staging mode

- Expected:
  - user credited: 900
  - protocol credited: 60
  - charity credited: 30
  - academy credited: 10
- Confirm via wallet history endpoint
- If mismatch: log exact numbers and create FIX ticket for backend-agent

E2E-3: Miner appears from on-chain ownership

- Mint a MinerNFT to the user (staging admin action is fine)
- Indexer must sync it
- Frontend “Miners” page shows it

E2E-4: Rewards merkle root publish + verify

- Run rewards cron (manual trigger allowed)
- Root is published on-chain
- Backend exposes proof for user/date
- Frontend verifies proof locally and shows ✅

E2E-5: Marketplace list renders from indexer

- Create a listing on-chain
- Indexer pulls it
- Marketplace page shows the listing
- Buy flow shows fee breakdown (10% total on sale if configured for marketplace.default)

STEP 6 — Staging deployment (server)

If infra supports SSH deploy:

- deploy backend + frontend to staging host
- run migrations
- confirm health endpoints

- confirm the same E2E checks (at least E2E-1..E2E-3)

COMMANDS YOU MUST SUPPORT (document in runbook):

- Install:
```
  pnpm install
```
- Contracts:
```
  cd contracts/evm && forge test
  forge script ...deploy_amoy...
```
- Backend:
```
  pnpm db:migrate
  pnpm test
  pnpm start:dev
```
- Frontend:
```
  pnpm build
  pnpm dev
```
- Infra:
```
  docker compose up -d
```

FILES TO CREATE/UPDATE:

1) /docs/STAGING_RUNBOOK.md

   Must include:

   - how to set env
   - how to deploy contracts to Amoy
   - how to start services
   - the 5 E2E checks with expected results

2) /docs/RELEASE_CHECKLIST_V3.md

   Must include:

   - “Before merge” checks
   - “After merge” checks

  - “Before staging deploy” checks
  - “Before production” checks (placeholders)

3) /docs/FIX_TASKS.md

  Format each task like:
  - Title
  - Owner agent
  - Steps to reproduce
  - Expected vs actual
  - Suspected area (files)
  - Acceptance criteria

DEFINITION OF DONE (DoD):

- staging branch builds successfully
- local docker + services run
- E2E-1..E2E-5 pass locally (documented)
- contracts are deployed on Amoy and addresses stored in repo
- runbook + checklist committed
- if any failures: FIX_TASKS.md exists with clear tickets

BRANCH / PR:

- create branch: feat/v3-integration-runbook
- open PR into staging
- include in PR description: E2E results + contract addresses + screenshots (optional)

OUTPUT REQUIREMENT:

Return:

- list of new/changed files
- exact commands to reproduce E2E locally
- final “Staging is ready” statement only if E2E checks pass