**Концепт приложения**

**Цель**  
Сделать игровой симулятор криптотрейдинга, который учит основам биржевой торговли и аналитики без реальных денег, удерживает пользователя геймификацией и собирает метрики для развития продукта.

**Платформа**  
Android. Фронт - React (React Native/Expo). Бэкенд - Node + PostgreSQL.

**Главные модули**

1. **Торговый симулятор**
   * Виртуальные валюты: «доллары» и «криптовалюта».
   * Реальные рыночные котировки (API) + мультипликатор x1-x50.
   * Параметры сделки: сумма, TP, SL, направление, левередж.
   * Операции: открыть, редактировать, закрыть. Всё обновляется в реальном времени.
2. **Обучение**
   * Чёткая пошаговая программа: текст, интерактив, квизы.
   * Виртуальный наставник «Панда» объясняет термины и функции.
3. **Геймификация**
   * 50+ достижений, уровни и рейтинги.
   * Ежедневные награды, рулетка из 9 секторов, боксы на 30 призов.
   * Редкие события: отключение рекламы 7 дн, «золотой» приз $10 000 (0,5 %).
   * Pro-режим на 3/5/10 дней с расширенным анализом графиков.
4. **Аналитика и метрики**
   * Полный набор UA, Engagement, Retention, Monetization, Ad Performance.
   * Хранение на сервере, экспорт в BI.
   * Дашборд в приложении: баланс, P/L, лучшие/худшие сделки, макс / средние показатели.
5. **Реклама и монетизация**
   * Вознаграждение за просмотр видео-ads (бонусы, кейсы).
   * Покупка виртуальных валют за реальные деньги (optionally).
   * Продажа подписки на Pro-режим.

**Интерфейс (4 языка)**  
Главный экран: баланс, выбор пары, график, кнопки «Купить/Продать».  
Нижняя навигация: Дашборд • Сделки • Рукинг/кейсы • Профиль.  
График: свечи/бары/линия, интервалы 1 м-1 д.  
Все данные обновляются WS или SSE без перезагрузок.

**Техническое ядро**

* React Native Expo → TypeScript.
* Redux Toolkit (клиентская стейт-машина).
* Socket.io для стрима котировок.
* Node + Express + PostgreSQL; Redis для сессий и очередей заданий.
* Firebase / Amplitude для метрик и push-уведомлений.

**Подробное описание функционала приложения**

**1. Онбординг и авторизация**

1. Варианты входа: Google, Facebook, Apple ID, GitHub.
2. После авторизации — краткое обучение с «Пандой»: 3–5 интерактивных экранов, объясняющих интерфейс, торговлю и геймификацию.
3. Пропуск обучения фиксируется в метрике tutorial\_skip\_rate.

**2. Баланс и валюты**

1. Две виртуальные валюты  
   • Доллары — базовая единица.  
   • Криптовалюта — для симуляции покупки/продажи.
2. Экраны показывают:  
   • Общий баланс «Всего».  
   • Доступные средства «Свободно».
3. Конвертация внутри симулятора по текущему курсу пары.

**3. Рынок и котировки**

1. Источник котировок — внешний API, обновление по WebSocket ≤ 1 сек.
2. Пользователь выбирает торговую пару из выпадающего списка.
3. График: свечи, бары или линия; интервалы — 1 м, 5 м, 15 м, 1 ч, 4 ч, 1 д.
4. В Pro-режиме доступны индикаторы (SMA, EMA, RSI) и аннотации.

**4. Сделки**

1. **Создание ордера**  
   • Параметры: сумма, направление (рост/падение), мультипликатор x1-x50.  
   • Слайдер: процент от свободного баланса.  
   • Поля TP и SL с валидацией (TP > цена, SL < цена).  
   • Видео-кнопка «Video Bonus»: просмотр рекламы ⟶ +5 % к доступной сумме.  
   • Подтверждение отдельным модальным окном.
2. **Редактирование**  
   • Можно менять TP, SL, направление до закрытия.  
   • Все обновления отражаются мгновенно.
3. **Закрытие**  
   • Итоговая прибыль/убыток показывается цветом: зелёный/красный.  
   • Подтверждение кнопкой «Забрать».

**5. Личный кабинет**

1. **Дашборд**  
   • Кол-во сделок, максимальная/средняя сумма, лучший P/L, худший P/L.  
   • Карточки-виджеты обновляются в реальном времени.
2. **Открытые сделки**  
   • Таблица активных позиций: тикер, цена входа, текущий P/L, мультипликатор.  
   • Кнопка закрытия прямо в строке.
3. **Аналитика успешных сделок**  
   • Топ-5 сделок за 30 дней по прибыли.
4. **Макс/средние показатели**  
   • Графики (bar или line) по максимальному профиту и убытку, средней сумме сделки.

**6. Геймификация**

1. **Достижения** — 50 событий (пополнение, обучение, торговля).
2. **Ежедневные бонусы** — таблица на 24 дня, награда растёт от $100 до $500 000, условия: визит или число просмотренных роликов.
3. **Рулетка** — 9 секторов, призы $100-$10 000 + «Pro-режим на 3 дня», шанс золотого приза $10 000 — 0.5 %.
4. **Боксы** — таблица из 30 призов $100-$3 000, шанс отключения рекламы — 0.5 %.
5. **Pro-режим** (3/5/10 дней) — расширенные индикаторы, цветовые схемы графика.
6. **Виртуальный помощник «Панда»** — статические советы + анимированная реакция на события.

**7. Реклама и монетизация**

1. Форматы: Rewarded Video (бонус к балансу, кейсы, рулетка), Interstitial (между экранами).
2. Метрики: ads\_watched\_per\_user, click\_to\_install\_rate, ad\_engagement\_rate, ROAS.
3. Отключение рекламы: редкий дроп из бокса или покупка за виртуальную валюту.

**8. Метрики (backend)**

* **User Acquisition**: installs, first\_open, signup\_rate, trade\_open\_rate.
* **Engagement**: sessions, screens\_opened, trades\_per\_user, avg\_virtual\_balance\_used.
* **Retention**: D1, D3, D7, D30, churn\_rate.
* **Monetization**: conversion\_to\_paid, ARPU, ARPPU, total\_revenue.
* **Ad Performance**: CPI, CPA, ROAS.  
  Все события отправляются в PostgreSQL + Amplitude; дашборды BI строятся автоматически.

**9. Сетевые и серверные функции**

1. Node + Express API: аутентификация, хранение балансов, награды, сделок.
2. Redis: сессии, очереди рефрешей метрик.
3. Планировщик: перерасчёт рейтинга раз в час.

**10. Локализация**

Полная поддержка EN, ES, FR, PT. Строки хранятся в JSON, выбираются по языку системы.

**11. Безопасность и устойчивость**

1. JWT-токены, refresh-механизм.
2. Количество открытых сделок ограничено параметром в настройках сервера.
3. Резервное копирование базы раз в 6 часов.

**Ключевые процессы приложения**

**1. Онбординг**

1. **Запуск**
   * Проверка JWT-токена. Нет токена → экран авторизации.
2. **Авторизация**
   * OAuth (Google / Facebook / Apple / GitHub) → получение id, e-mail, аватара.
   * Создание записи users при первом входе.
3. **Первичное обучение**
   * Отдаётся JSON-сценарий «Панды» (5 шагов).
   * Событие tutorial\_start; завершение → tutorial\_complete.
4. **Выбор языка**
   * Если язык телефона не EN, ES, FR, PT → UI сразу локализован.

**2. Поток «Создать сделку»**

1. **Выбор торговой пары**
   * Выпадающий список → сохранение выбранной pair в Redux и backend.
2. **Настройка ордера**
   * Поле суммы (инлайн валидация ≤ free\_balance).
   * Слайдер процента баланса, мгновенно меняет поле суммы.
   * Выпадающий мультипликатор x1-x50.
   * Поля TP / SL, проверка: TP > price, SL < price.
3. **Видео-бонус (опция)**
   * Нажатие «Watch Ad» → SDK рекламы.
   * Успешный просмотр → плюс 5 % к доступной сумме (signal от SDK).
4. **Подтверждение**
   * Модалка с итогами: сумма, мультипликатор, комиссия 0.
   * POST /order/open → запись в orders, вычитание из free\_balance.
   * Сокет-ивент order\_open для UI.

**3. Реальное время и котировки**

1. Сервис котировок открывает WS-канал /prices/{pair}.
2. Backend пробрасывает биржевой поток без хранения.
3. Клиент обновляет график и пересчитывает плавающий P/L всех открытых ордеров.
4. Достижение «Первая живая цена» фиксируется событием price\_stream\_start.

**4. Редактирование сделки**

* PUT /order/{id} с новыми TP / SL или направлением.
* Backend валидация и пересчёт потенциального P/L.
* Сокет-ивент order\_update возвращает свежие значения.

**5. Автоматическое закрытие TP / SL**

1. Процесс-воркер в Redis-очереди слушает поток цен.
2. Условие срабатывает → PATCH /order/{id} статус closed.
3. Прибыль возвращается в free\_balance, запись добавляется в trades\_history.
4. Событие order\_close\_auto уходит в аналитику + уведомление in-app.

**6. Ручное закрытие**

* Пользователь нажимает «Закрыть».
* PATCH /order/{id} → статус closed\_manual.
* Баланс обновляется, модалка результата.

**7. Ежедневный бонус**

1. Cron 00:00 UTC → всем пользователям выставляется daily\_claim\_available = true.
2. При заходе клиент делает GET /rewards/daily.
3. Нажатие «Забрать» → POST /rewards/claim → сумма + запись reward\_history.
4. Таблица прогресса 24 дня обновляется; событие daily\_reward\_claimed.

**8. Рулетка / Боксы**

1. Пользователь тратит билет или смотрит рекламу → POST /loot/start.
2. Backend выбирает приз по весовой таблице, пишет в loot\_history.
3. Клиент показывает анимацию вращения с уже известным результатом.
4. После анимации → GET /loot/result/{id}, баланс/статус (Pro, no-ads) обновляется.

**9. Pro-режим**

1. Активация:
   * Покупка за виртуальную валюту или приз из лута → POST /pro/activate.
2. Backend пишет срок окончания в users.pro\_until.
3. Ключ в JWT-payload (isPro = true).
4. По истечении срока фронт получает 401 pro\_expired → UI скрывает индикаторы.

**10. Метрики и аналитика**

* Клиент шлёт события пачками (batch\_size 20) в /analytics/collect.
* Сервер складывает в ClickHouse.
* Nightly ETL обновляет дашборды Retention / ARPU / ROAS.
* Redis job раз в час пересчитывает рейтинг пользователей по сумме виртуальных активов.

**11. Реклама**

1. SDK запускается после ads\_consent = true.
2. Rewarded Video — строго перед бонусами; Interstitial — не чаще одной за 2 мин.
3. Callback от SDK → ad\_watched событие; если rewarded — выдаётся бонус.

**12. Безопасность и устойчивость**

* Все запросы JWT + HMAC-подпись тела.
* Rate-limit 60 req/min IP+user.
* PostgreSQL горячая реплика; бэкап S3 каждые 6 ч.
* Redis — кластер 3 нод, Persistence AOF.
* Downtime фронта — локальный кеш котировок на 30 с для плавности.

**13. Локализация**

* Файл i18n/{lang}.json.
* При смене языка → reload статики, состояние Redux сохраняется.

**Техническая часть**

**1. Архитектура (Layered, контейнеры)**

* **Client (React Native / Expo)**
  + JS/TS + Redux Toolkit, RTK-Query.
  + WebSocket ↔ ws-gateway для котировок и сигналов сделок.
  + REST ↔ API-gateway для CRUD, наград, аналитики.
* **API-gateway (Node 18 + Express)**
  + Auth, rate-limit, routing в сервисы.
* **Core Services (Docker)**
  + **Auth-service** – OAuth, JWT, refresh-pair.
  + **Market-service** – прокси котировок, нормализация, кэш 1 с.
  + **Trading-sim** – логика ордеров, расчёт P/L, триггеры TP/SL.
  + **Rewards-service** – рулетка, боксы, ежедневка, Pro-таймер.
  + **Ad-service** – callbacks AdMob/Meta, выдача бонусов.
  + **Metrics-collector** – прием batch-event’ов, запись в ClickHouse.
* **Data layer**
  + **PostgreSQL 15**: users, balances, orders, trades, achievements, rewards, ad\_events.
  + **Redis 7**: sessions, WS-pubsub, delayed-jobs (BullMQ).
  + **ClickHouse**: event-stream (≈ 100 M+/сутки) для BI.
* **External**
  + Crypto feed (Binance/Bybit REST+WS).
  + Firebase FCM (push).
  + S3 backup (pg\_dump, CH-snapshots).

**2. API (key endpoints)**

| **Method** | **Path** | **Purpose** |
| --- | --- | --- |
| POST | /auth/oauth | social login → JWT |
| GET ws | /prices/{pair} | котировки real-time |
| POST | /order/open | создать ордер |
| PATCH | /order/{id} | TP/SL/dir update |
| PATCH | /order/{id}/close | ручное закрытие |
| GET | /balance | общий + свободный |
| POST | /loot/spin | рулетка/бокс |
| POST | /reward/daily | забрать ежедневку |
| POST | /pro/activate | включить Pro |
| POST | /analytics/collect | batch events |

All calls HMAC-signed; JWT в Authorization.

**3. Схема БД (основное)**

users(id PK, email, name, avatar, lang, pro\_until timestamp)

balances(user\_id PK, total numeric, free numeric)

orders(id PK, user\_id, pair, dir, leverage, sum, tp, sl, open\_ts, status, close\_ts, pl)

trades\_history(id PK, order\_id, pl, dur\_ms)

achievements(id PK, code, title, desc)

user\_achv(user\_id, achv\_id, ts)

rewards\_history(id PK, user\_id, type, payload, ts)

ad\_events(id PK, user\_id, ad\_id, kind, ts)

JSONB поля под гибкие payload'ы (loot).

**4. Реальное время**

* Gateway WS ⇒ Redis pubsub ⇒ clients.
* Триггеры TP/SL в trading-sim воркере, подписанном на поток цен.
* Закрытие ордера пушится через WS-event order\_close.

**5. Безопасность**

* OAuth 2.0 PKCE, refresh-token rotation.
* JWT 15 min, refresh 30 d; blacklist Redis.
* Rate-limit 60 req/min IP+user (express-rate-limit + redis-store).
* CSP, HSTS, SRI на статику.
* ПД шифруется at-rest (AES-256, pgcrypto).

**6. DevOps**

* **CI/CD**: GitHub Actions → Docker build → Helm charts → K8s.
* **Observability**: Prometheus + Grafana, Loki logs, Sentry JS/Node.
* **Deploy**: blue-green, migrations through Flyway.
* **Backups**: pg\_dump + WAL-G (6 h); CH snapshots daily.

**7. Масштабирование**

* Node-сервисы stateless; HPA по CPU/latency.
* WS-gateway sticky-sessions via NGINX + Redis cluster.
* Market-service кэшируется в Redis, TTL 1 s, снижаем API cost >80 %.
* ClickHouse 3-node shard-replica; event-load >50 k EPS.

**8. Тесты**

* **Unit** (Jest), **integration** (Supertest + Testcontainers).
* **E2E mobile** – Detox on CI (Android emulator).
* **Load** – k6 scripts, целевой RPS 10 k.
* **Security** – OWASP ZAP nightly.

**9. Локализация**

* i18n JSON, lazy load; plural rules ICU.