**Карта проекта (что где лежит)**

* Аналитика и стратегии: src/analysis/\*, src/indicators.py, src/strategy.py, src/paper.py
* Исполнение: executors/\* (Binance REST, симулятор, UI-агент-заглушка, базовый протокол)
* FastAPI роуты: routers/ohlcv.py, routers/trading.py, routers/trading\_exec.py, routers/ui.py
* Схемы/валидация: schemas/\* (Pydantic)
* БД и доступ: db/models.py, db/crud.py, db/session.py
* Риск-контроль: risk/deadman.py, risk/trailing.py, risk/position\_sizing.py (есть)
* Источники данных: sources/\* (в т.ч. alpha\_vantage.py, ohlcv.py)
* Сервисы: services/trading\_service.py, services/reconcile.py
* Приложение: src/main.py (FastAPI, middleware, endpoints)
* Тесты: tests/\* (индикаторы, анализ, paper-trader, ohlcv, исполнение и пр.)

**Сопоставление с функциональными требованиями (по разделам)**

**1) Подготовка и инфраструктура**

**Требования:** автозапуск/проверки, .env/ключи, мониторинг, self-check.  
**Есть в коде:**

* Загрузка ключей/конфигов: utils/secrets.py (используется в executors/api\_binance.py, routers/trading\_exec.py).
* FastAPI приложение и CORS/GZip/т.д.: src/main.py (инфраструктурный каркас есть).
* Конфиг и схемы: schemas/\*, services/\* под единым интерфейсом.  
  **Пробелы:**
* Нет явного self-check эндпоинта (здоровье источников, доступ к бирже, версия/таймсинк).
* Нет модуля системного мониторинга ресурсов и авто-перезапуска.  
  **Статус:** ◑ Частично.

**2) Анализ рынка**

**Требования:** индикаторы (EMA/RSI/MACD/BB/ATR), свечные паттерны, уровни, MTF, итоговый score.  
**Есть в коде:**

* Векторные индикаторы, нормализация, MTF-конфирм: src/analysis/analyze\_market.py (большой модуль ~700 строк).
* Базовые индикаторы и сигналы EMA-cross: src/indicators.py, src/strategy.py.
* Тесты индикаторов и анализа: tests/test\_indicators.py, tests/test\_analyze\_market.py.  
  **Пробелы:**
* Не увидел явной реализации MACD/RSI/BB/ATR как стабильного публичного API (часть индикаторов есть, но не весь перечень из требований).
* Свечной анализ: если и есть эвристики, их нужно связать с контрактом результатов.  
  **Статус:** ◑ Частично (основа сильная, покрытие индикаторов неполное).

**3) Генерация торговых сигналов**

**Требования:** ансамбль стратегий, confidence score, фильтры по режиму рынка/волатильности.  
**Есть в коде:**

* EMA-cross и базовый генератор сигналов: src/strategy.py, routers/trading.py (эндпоинты signals, backtest).
* В analyze\_market.py рассчитаны признаки для принятия решения (trend/volatility/signal/confidence/reasons/levels).  
  **Пробелы:**
* Ансамбль стратегий (несколько одновременно + интегральный score) — как каркас не выделен.
* Управление частотой сделок/антишумовой фильтр — частично (в анализе), но централизованного throttling-слоя нет.  
  **Статус:** ◑ Частично.

**4) Риск-менеджмент**

**Требования:** риск ≤1% на сделку, SL/TP/трейлинг, дневной стоп, лимит сделок, dead-man switch.  
**Есть в коде:**

* Paper-уровень SL/TP/Trail с учётом gap-through и комиссий: src/paper.py (подробная логика, тесты tests/test\_paper.py).
* Трейлинг-модуль: risk/trailing.py.
* Dead-man каркас: risk/deadman.py (загрузка risk-конфига, heartbeat протокол).
* Позиционирование/размер позиции: risk/position\_sizing.py (есть заготовка).  
  **Пробелы:**
* Централизованный лимитер числа сделок/дневной стоп (hook в сервисный слой) — не увидел включения в services/trading\_service.py.
* Расчёт риска ≤1% со связкой капитала и ATR/волы — есть заготовка, но требуется интеграция в исполнение.  
  **Статус:** ◑ Частично (механика SL/TP/Trail — хорошо, лимиты/1% — интегрировать).

**5) Исполнение сделок**

**Требования:** API брокера/биржи, типы ордеров, модификация SL/TP, сверка позиций.  
**Есть в коде:**

* Базовый протокол исполнителя и типы: executors/base.py (Executor, OrderResult, Position).
* Binance REST (spot) с фильтрами/подписанием/precision: executors/api\_binance.py (объёмный модуль ~1000 строк).
* Симулятор исполнителя: executors/simulated.py.
* Заглушка UI-агента для экранной автоматизации: executors/ui\_agent.py.
* Роут для исполнения: routers/trading\_exec.py (обработка ключей/режимов).  
  **Пробелы:**
* Нет явной модификации/линковки SL/TP к позиции через OCO/стоп-ордера после входа (нужно проверить методики внутри api\_binance.py и эндпоинтов).
* Сверка позиций и reconciliation — есть services/reconcile.py, но нужен связанный cron/воркер-запуск.  
  **Статус:** ◑ Частично (каркас полный, нужна довязка SL/TP через API и сверка в рантайме).

**6) Сопровождение сделок**

**Требования:** мониторинг, автоподтягивание стопа, закрытие по развороту, учёт комиссий/проскальзывания.  
**Есть в коде:**

* Трейлинг/закрытия в PaperTrader реализованы; комиссии учитываются в моделировании.
* Сервисный слой для управления исполнителем: services/trading\_service.py.  
  **Пробелы:**
* Реальный мониторинг открытых позиций + авто-подтягивание стопа на стороне биржи — нет выделенного фонового воркера.
* Проскальзывание/latency — в симуляторе отсутствует, в реале не учитывается метриками.  
  **Статус:** ◑ Частично.

**7) Мониторинг и отчёты**

**Требования:** дашборд, телеграм/почта уведомления, авто-отчёты (день/неделя), журнал JSON/CSV.  
**Есть в коде:**

* Журнал OHLCV/сделок в БД: db/models.py, db/crud.py (bulk-insert, upsert под SQLite).
* Эндпоинты работы с OHLCV: routers/ohlcv.py (загрузка/хранение/выдача).  
  **Пробелы:**
* Витрина/дашборд (FastAPI + HTMX/Tailwind) заявлена в спринтах, но UI коды/шаблоны не увидел (вероятно, в другом проекте; здесь есть routers/ui.py, но без завершённого фронта).
* Нет уведомлений и авто-отчётов (cron/apscheduler/Telegram).  
  **Статус:** ○ В зачатке (требует реализацию).

**8) Автовосстановление**

**Требования:** перезапуск терминала, reconnection, восстановление позиций, проверка консистентности.  
**Есть в коде:**

* Каркас recon/сверки: services/reconcile.py.
* Dead-man heartbeat: risk/deadman.py (основа).  
  **Пробелы:**
* Нет готового супервизора (watchdog/apscheduler/systemd) и сценариев старта/перезапуска.
* Восстановление позиций из истории брокера — нужен полный цикл в services/reconcile.py + периодический запуск.  
  **Статус:** ◑ Частично.

**9) Обучение и адаптация**

**Требования:** walk-forward, переоценка параметров, адаптивный порог confidence, «ворота новостей».  
**Есть в коде:**

* Бэктест базовой стратегии: routers/trading.py + src/paper.py.  
  **Пробелы:**
* Нет WFA, нет автопереоценки параметров по расписанию, нет адаптивного порога в рантайме.  
  **Статус:** ○ Нет/минимум.

**10) Безопасность и контроль**

**Требования:** шифрование ключей, маскирование, двойное подтверждение массовых действий, аудит.  
**Есть в коде:**

* Безопасная загрузка ключей (ошибки понятные), разделение testnet/prod: executors/api\_binance.py, routers/trading\_exec.py.  
  **Пробелы:**
* Шифрование локальных секретов (DPAPI/KeyVault) — нет.
* Двухуровневое подтверждение «закрыть всё» — не увидел.
* Маскирование чувствительных логов и скриншоты — нет.  
  **Статус:** ○ Требуется реализация.

**Чек-лист готовности (по спринтам)**

**Спринт 1 — Подготовка среды → ✅**

* Python/окружение + базовые зависимости — да.
* Источник котировок подключён (sources/alpha\_vantage.py, routers/ohlcv.py) — да.

**Спринт 2 — Расширенный анализ рынка → ◑ 70%**

* Индикаторы и нормализация данных — есть.
* Свечные паттерны/BB/ATR/MACD/RSI — частично/не в полном объёме.
* MTF-подтверждение и контракт ответов — в analyze\_market.py реализовано.
* Тесты индикаторов/аналитики — есть.

**Спринт 3 — Симулятор (paper-trading) → ◑ 80%**

* PaperTrader с SL/TP/Trail/комиссией/гепами — реализовано.
* Журнал/статистика win-rate/PnL/maxDD — есть внутри PaperTrader.
* Эндпоинт бэктеста (прогон) — routers/trading.py — есть.
* Не хватает: сохранения журнала сделок в CSV/JSON из эндпоинтов + стабильный контракт ответа.

**Спринт 4 — Исполнение (Demo / testnet) → ◑ 60–70%**

* Binance REST исполнитель — реализован (executors/api\_binance.py).
* Простейшие типы ордеров — есть; связку SL/TP после входа нужно проверить/добавить (OCO).
* Роуты для исполнения — routers/trading\_exec.py.
* Нужна сверка позиций и явный «контур» демо-торговли (сервис/воркер).

**Спринт 5 — Риск-менеджмент → ◑ 50%**

* SL/TP/Trail — готовы; dead-man — каркас есть.
* Лимит риска 1%/дневной стоп/лимит сделок — нужно внедрить в services/trading\_service.py.
* Настройка конфигов риска и телеметрии — дополнить.

**Спринт 6 — Витрина (Web UI) → ○ 20%**

* Бэкенд-роуты частично есть; UI/шаблоны/HTMX/Tailwind — отсутствуют.
* Нужны таблицы позиций, сделки, PnL, статус-панель.

**Спринт 7 — Новости и AI-фильтр → ○ 10%**

* RSS/NLP/тональность — нет.
* «Ворота» новостей (stop-trade на high-impact) — нет.

**Спринт 8 — Автовосстановление → ◑ 30–40%**

* Модули dead-man/reconcile — каркасы есть.
* Нужны воркеры/cron, сценарии восстановления и сверка с биржей по расписанию.

**Прицельные next-steps (короткие задачи, чтобы довести до «демо-боеспособно»)**

1. **Закрыть Спринт 3 окончательно**

* В routers/trading.py добавить сохранение результатов бэктеста в db и/или CSV (контролируемый путь).
* Расширить schemas/trading.py (BacktestResponse) полями: win\_rate, pnl\_sum, max\_dd, trades (список), equity\_curve.

1. **Привязать риск-лимиты**

* В services/trading\_service.py внедрить лимит риска 1% на сделку (модуль risk/position\_sizing.py + equity из биржи).
* Дневной стоп и лимит сделок/день как декоратор/прослойка вокруг вызова исполнителя.

1. **SL/TP через API**

* В executors/api\_binance.py реализовать постановку OCO (TP/SL) сразу после market/limit входа (или эквивалент с linked-orders, если OCO недоступно для пары).
* Эндпоинт модификации стопов + автоподтягивание (воркер по символу).

1. **Сверка позиций**

* Доделать services/reconcile.py: сравнение состояния (внутренний журнал vs биржа) + корректирующие действия.
* Поставить планировщик (apscheduler) для periodic reconcile.

1. **Self-check и мониторинг**

* Добавить /health эндпоинт: версия, время, доступ к источникам, ping Binance, статус БД.
* Лёгкая метрика latency/сложение PnL, счётчики попыток/ошибок.

1. **Витрина**

* routers/ui.py + простая HTMX-страница: баланс, свободные средства, активные сделки, журнал последних ордеров, PnL.

1. **Автовосстановление**

* Heartbeat-сервис (deadman) + graceful shutdown/close all.
* Скрипт запуска с рестартом (run-dev.bat/supervisor/systemd).

**Быстрый чек-лист по файлам (да/нет/надо доделать)**

* src/analysis/analyze\_market.py — **ДА** (сильный модуль; добить индикаторы/паттерны).
* src/indicators.py — **ДА** (расширить до MACD/RSI/BB/ATR под единым API).
* src/strategy.py — **ДА** (добавить ансамбль и фильтры частоты сделок).
* src/paper.py — **ДА** (хорошо: SL/TP/Trail/комиссии/gap; добавить экспорт журнала).
* routers/trading.py — **ЧАСТИЧНО** (бэктест OK; добавить сохранение, расширить ответ).
* routers/ohlcv.py — **ДА** (работа с историей/БД есть).
* routers/trading\_exec.py — **ЧАСТИЧНО** (интеграция с API есть; добавить постановку OCO/SLTP).
* executors/api\_binance.py — **ЧАСТИЧНО** (каркас полный; дописать OCO/ошибки/ретраи).
* executors/simulated.py — **ДА** (надо учесть комиссию/проскальзывание при желании).
* services/trading\_service.py — **ЧАСТИЧНО** (вкрутить риск-лимиты, дневной стоп, лимит сделок).
* services/reconcile.py — **ЧАСТИЧНО** (дописать до реальной сверки и восстановления).
* risk/deadman.py — **ЧАСТИЧНО** (каркас heartbeat; подключить к жизненному циклу).
* risk/trailing.py — **ДА** (есть логика; связать с рантаймом).
* schemas/\* — **ДА** (расширить BacktestResponse, добавить схемы для журналов/статистики).
* db/\* — **ДА** (модели/CRUD есть; нужны таблицы сделок/ордеров, если их ещё нет).
* src/main.py — **ДА** (каркас приложения готов; добавить /health, UI-страницы).
* news/\* — **НЕТ** (модуль новостей/тональности отсутствует).