Выжимаем 400% из Opus

или как сжечь все токены на max подписке

Misha Druzhinin — https://www.druzhinin.co/

Обо мне

- ex-AWS, ex-Datadog
- CEO of Finsi — AI Agentic Growth Engine

Finsi: AI-движок для маркетинга. Объединяет все данные (реклама, email, retention, юнит-экономика), анализирует их и действует автоматически.

- 📝 76 спек в 15 категориях за 3 месяца.
- 💻 ~350k строк кода боевого проекта, которым пользуются клиенты.
- 🚀 Не сложный функционал делается за 2-4 часа от идеи до прода.

ANALYTICS

Profit Intelligence
Real-time P&L, unit economics, and contribution margin tracking

Retention Intelligence
Cohort analysis, churn prediction, and lifecycle insights

Predictive LTV
4-tier confidence LTV modeling with milestone tracking

Smart Segmentation
Natural language customer segmentation and targeting

Attribution
Multi-touch attribution and customer quality scoring

Support Insights
Support-to-churn correlation and ticket intelligence

AUTOMATION

Ads Autopilot
AI-managed Meta & Google ad campaigns with creative generation

Email Intelligence
Email autopilot with AI campaign planning and brand voice

Creative Studio
AI ad creative generation with hook-angle matrix

Competitor Intelligence
Monitor competitor ads and get counter-strategy recommendations

AI Recommendations
Ranked list of what matters now and what to do next

Customer Research
AI-powered customer surveys and feedback analysis


TAB
PROMPTS
SPEC
MULTI AGENT
imgflip.com

План

Будем выжимать из агентов максимум. Для этого будем использовать OpenSpec.

Что влияет на качество и скорость:

- 🎯 Цели
- 🔍 Фокус
- 🏗️ Датамодель

Часть 1: Greenfield

Что такое Derrick

- 🚀 Self-contained data pipeline engine
- 🔄 Достаёт из SaaS-API (Klaviyo, Shopify, FB)
- 💾 Кладёт в вашу инфраструктуру (Postgres, S3, Kafka)
- 📦 Single binary, ноль зависимостей

▒▒▒ Статистика

- Active dev days: 11
- Backend Rust LoC: ~12,750
- Frontend TS/CSS LoC: ~12,625

▒▒▒ Объём

- UI screens: 11
- Database tables: 12
- API routes: 69

🧱 Manual Engineering Estimate by Category

Category	LoC	Est. Hours	Est. Weeks
🔄 Sync Engine & Scheduler	2,067	80–120	2–3
🔌 Source Connectors (7)	2,308	100–140	2.5–3.5
💾 Destination Writers (3)	1,376	60–80	1.5–2
🌐 API Server (69 routes)	3,070	60–80	1.5–2
🗄️ Database Layer (12 tables)	2,640	40–60	1–1.5
🖥️ Frontend UI (11 screens)	12,625	120–160	3–4
🧪 Testing & Tooling	1,500	30–40	1
🚀 DevOps & Infra	823	16–24	0.5
Total	**~26,400**	**506–704**	**13–18**

Входные документы

Вместо абстрактного PRD — три реальных документа из проекта `derrick`:

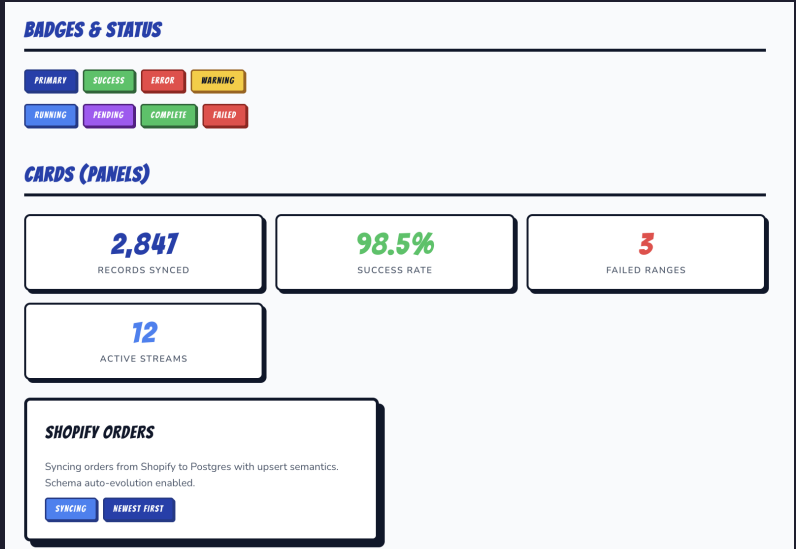
- 🎯 `concept.md` — бизнес-логика (зачем это нужно, как работают пайплайны и коннекторы)
- 🏗️ `architecture.md` — технические рамки (single-binary, async I/O, streaming)
- 💾 `datamodel.md` — структура базы (сущности Source, Destination, Stream и их связи)
- 🎨 `seed` — Стартовая обвязка проекта

UI UX Pro Max

База знаний для AI дизайна: ui-ux-pro-max-skill.nextlevelbuilder.io (https://ui-ux-pro-max-skill.nextlevelbuilder.io/)

- 🎨 UI-стили и цветовые палитры
- 🔤 Шрифтовые пары
- 📉 Типы графиков и дашбордов
- 🧠 UX-гайдлайны для компонентов

Превращает дефолтный Tailwind-код в продуманный дизайн.


BADGES & STATUS
PRIMARY
SUCCESS
ERROR
WARNING
RUNNING
PENDING
COMPLETE
FAILED
CARDS (PANELS)
2,847
RECORDS SYNCED
98.5%
SUCCESS RATE
3
FAILED RANGES
12
ACTIVE STREAMS
SHOPIFY ORDERS
Syncing orders from Shopify to Postgres with upsert semantics.
Schema auto-evolution enabled.
SYNCING
NEWEST FIRST

Tessl.io + SDX Project

Что если я не люблю промпты?

Tessl.io — реестр скилов и платформа для AI агентов.

Используем подготовленный скил: `spec-driven-magic/sdx-project`

- 📥 Берёт наши `concept.md`, `architecture.md`, `datamodel.md`
- 🏗️ Разворачивает базовый репозиторий с нужным стеком
- ⚙️ Автоматически генерирует конфиги OpenSpec
- 🗺️ Создаёт Roadmap проекта
- 📝 Пишет начальные спеки для всех фаз

Вместо того чтобы делать это руками — один запуск скила.

Пока агент работает...

Что такое OpenSpec

```
project/
  openspec/
    config.yaml        ← стек, правила, ограничения
    schemas/
      schema.yaml      ← pipeline артефактов
    specs/             ← библиотека спецификаций
    changes/           ← текущие и архивные изменения
```

Change

Change - это единица работы в OpenSpec.

```
changes/task-board/
  proposal.md     ← зачем это нужно
  datamodel.md    ← какие сущности затронуты
  specs/          ← что система должна делать
  design.md       ← как реализовать
  tasks.md        ← конкретный чеклист задач
```

Каждый файл - один уровень детализации. Proposal = зачем | Specs = что | Design = как | Tasks = делай

Жизненный цикл

```
proposal → design → specs → tasks → apply
```

Порядок

Каждый артефакт читает предыдущие. Но можно обновлять все в любой последовательности

Агент не может делать tasks, пока не готовы specs и design.

Уровни

Артефакт	Вопрос
Proposal	Зачем?
Specs	Что делает?
Design	Как делает?
Tasks	Что конкретно?

Config = контракт

```
schema: spec-driven
context: |
  Project: Derrick — Lightweight Data Pipeline Engine
  Tech stack:
    Backend: Rust, Rocket, sqlx, tokio
    Frontend: React, Vite, TypeScript
  Architecture:
    - Single-process deployment (no Docker orchestration)
    - Multi-crate Cargo workspace
  Key concepts:
    - Stream is the atomic unit
    - Range-based sync tracking (not cursor-based)
rules:
  tasks:
    - Break tasks into chunks completable in one session
    - Order tasks so the first few produce something runnable
```

Workflow

Workflow можно переопределять.

▒▒▒ По умолчанию

Schema задаёт pipeline: какие артефакты, в каком порядке, какие зависимости.

▒▒▒ Можно менять

- Свои шаблоны для артефактов
- Свои шаги в pipeline
- Свои правила в config
- Свои скилы для каждого шага

OpenSpec - фреймворк, не жёсткая структура. Подстраиваете под свой процесс.

Что я делаю пока они работают

- 🔍 Не слежу за кодом (может быть иногда)
- ⏸️ Вмешиваюсь только если агент явно отклонился
- ☕ Запустить всех, заняться другим, вернуться проверить

Все повторяемые действия — в скилы

Например, скил `next`:

- 🎯 Автоматически берёт следующую фазу из Roadmap
- 🚀 Инициирует создание спек и переход к реализации
- ⏱️ Написан за минуту, экономит мой контекст

```
> make a skill named 'next' which will take next spec
> from roadmap into implementation
```

Часть 2: Brownfield

На что обращать внимание

- Главное правило: задачи не должны пересекаться
- Если начали пересекаться - нужно разбираться: хорошо это или плохо
- Чёткие зоны ответственности для каждого агента
- Config фиксирует правила - все агенты их соблюдают
- Маленькие задачи - меньше шансов на конфликт

Как с этим жить

Что работает

- Tasks разбиты по зонам
- Каждый агент трогает свои файлы
- Общий контракт через specs
- Коммиты после каждой группы

Где ломается

- Агенты трогают одни и те же файлы
- Зависимости между задачами не учтены
- Слишком крупные задачи
- Нет config - каждый решает по-своему

Забираем домой

Что вынести

- OpenSpec - практический инструмент, пользоваться можно сразу. Changes, specs, lifecycle, workflow - всё настраивается
- Параллельные агенты работают если задачи не пересекаются. Чёткие зоны, маленькие задачи, общий контракт
- Скилы пишутся за 10 минут. Markdown-файл с инструкциями, не код
- Config фиксирует правила один раз. Дальше все агенты их соблюдают

Что попробовать

1. `openspec init` + первый change
2. Зафиксировать стек и правила в config
3. Запустить два агента параллельно на разных задачах
4. Написать свой скил

Спасибо! Вопросы?