Игорь Антонов

Т-Банк, HolyJS 2026

Spec-Driven Backend с Claude

Node.js, Clean Architecture
и OpenSpec на практике

Игорь Антонов

Team Lead, Т-Банк



@antonovjs

- Директор ПК «Podlodka React Crew»
- Директор ПК «Podlodka AI Crew»
- Участник ПК «Podlodka TechLead Crew»
- Участник ПК «Podlodka Java Crew»
- Лектор (HTML Academy, ИТМО)
- Веду канал в YouTube и Telegram

Проблематика

Зачем всё это, если агент и так может?



01 **Зрелые процессы**

Всё работает слаженно
и тут приходит AI

02 **У каждого свой опыт**

Получим N реализацией
одного подхода

03 **Постепенная раскатка**

Нельзя разом внедрить
во все проекты команды

04 **Работа с сопротивлением**

Не все готовы быстро меняться
и это неплохо

«Claude, реализуй Order Checkout»

37 файлов

+1847 строк

Этот объём ещё нужно проверить

Один запрос агенту

Реализуй оформление заказа

Node.js + TypeScript + Express
PostgreSQL + Clean Architecture

Создай заказ, проверь товары и остатки, рассчитай сумму, добавь тесты

Приёмка результата

Теперь сделайте Code Review

На складе 3 единицы

В заказе 2 + 2 одного товара

Суммарно нужно 4.
Какое правило?

Что агент решит самостоятельно?

Один productId повторяется

product-42 quantity: 2
product-42 quantity: 2



Не начинайте с промпта

Сначала сделайте знания проекта явными

Что должен знать агент

Явные знания проекта задают границы работы



Проект

Стек, команды и структура

Архитектура

Слои и допустимые зависимости

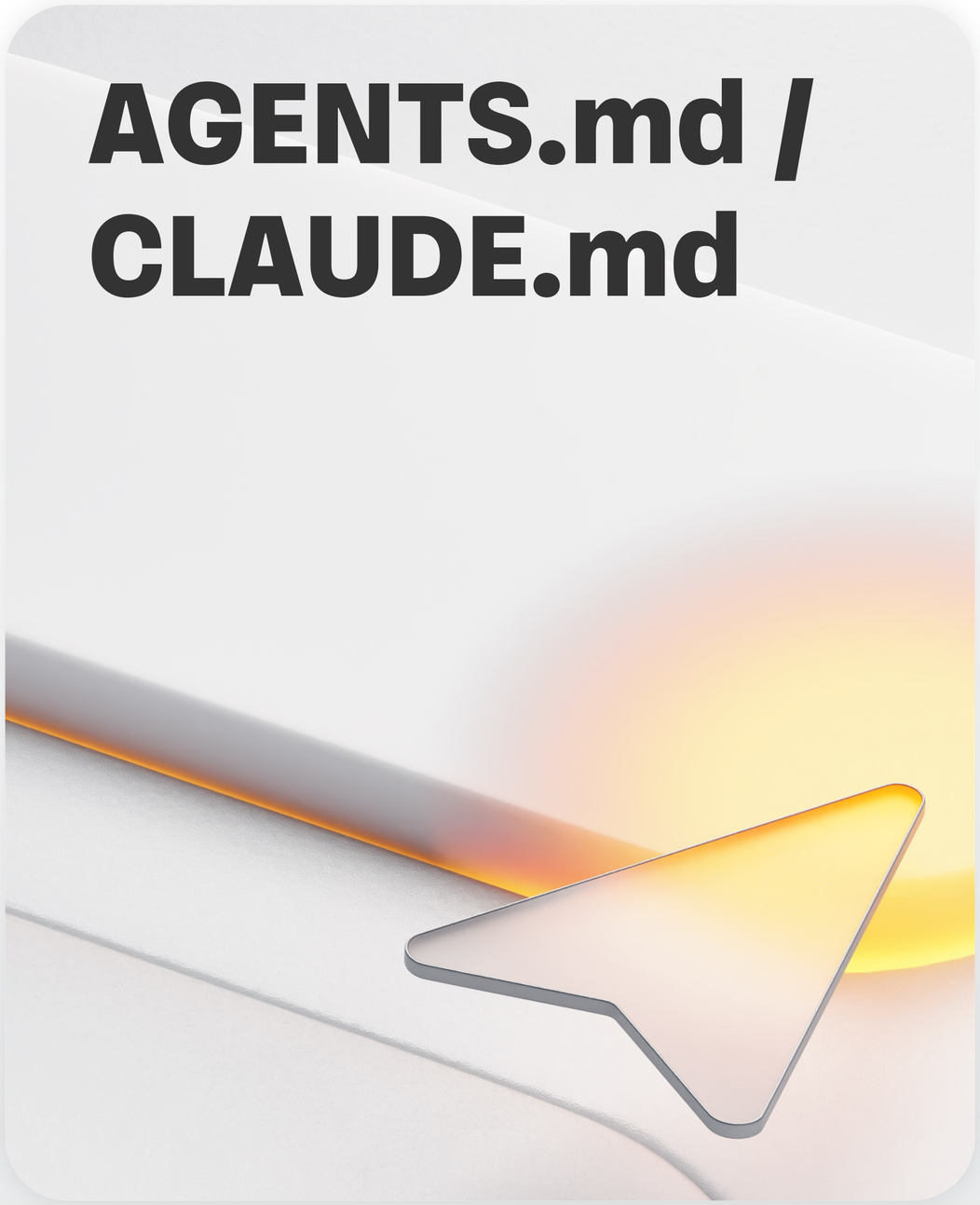
Инженерные правила

Ошибки, тесты и валидация

Процесс

Как работать и проверять результат

AGENTS.md / CLAUDE.md



presentation → application → domain

Domain не зависит от Express и PostgreSQL
Use cases не обращаются к БД напрямую
В контроллерах нет бизнес-правил

Перед завершением: typecheck, tests, lint
Сверить результат со спецификацией и Change

По реальному AGENTS.md. Формулировки сокращены

Где всё это живёт

```
holyjs-sdd-backend/
├── AGENTS.md
├── docs/order-checkout/
│   ├── spec.md
│   └── slices/
│       ├── 001-order-contracts.md
│       ├── 002-checkout-validation.md
│       ├── 003-order-pricing.md
│       ├── 004-atomic-order-write.md
│       └── 006-get-order.md
├── openspec/changes/
├── .agents/skills/
├── src/
└── tests/
```

Целевая структура. Тип работы
и зависимости указываем внутри файла

Один файл на единицу плана

005 — резерв

добавим после
изменения требований



Планирование

Dev Spec + slices

- FR-5: цена заказа сохраняется
- 003: ограниченная работа
- Явные зависимости

Planning contract

Бизнес-правило и границы

Исполнение

OpenSpec Change

- Текущий код и точки изменения
- План реализации
- Проверка снимка цены

Execution contract

Детали текущего исполнения

Откуда берётся Dev Spec?

Пробелы в постановке легко становятся предположениями AI



Brainstorming перед спецификацией

Superpowers Brainstorming
Сначала решения, затем документ

1

Идея

Задача и контекст
Order Checkout

2

Вопросы

Кто задаёт цену?
Когда резервируем?

3

Варианты

Обсуждаем ошибки
и ограничения

4

Решения

Фиксируем scope
и договорённости

5

Dev Spec

Описываем прове-
ряемое поведение



Dev Spec

FR-5. Цена фиксируется при оформлении

Изменение цены каталога
не меняет сохранённый заказ

Цена 1500 × количество 2 = сумма 3000

FR-6: итог рассчитывает сервер

FR-5/FR-6 из Dev Spec. Числа —
данные предлагаемой проверки

Спецификацию удобно ревьюить

50 страниц не делают требования понятнее

- Ограниченный scope
- Проверяемое поведение
- Однозначные формулировки
- Стабильные IDs



Dev Spec Review

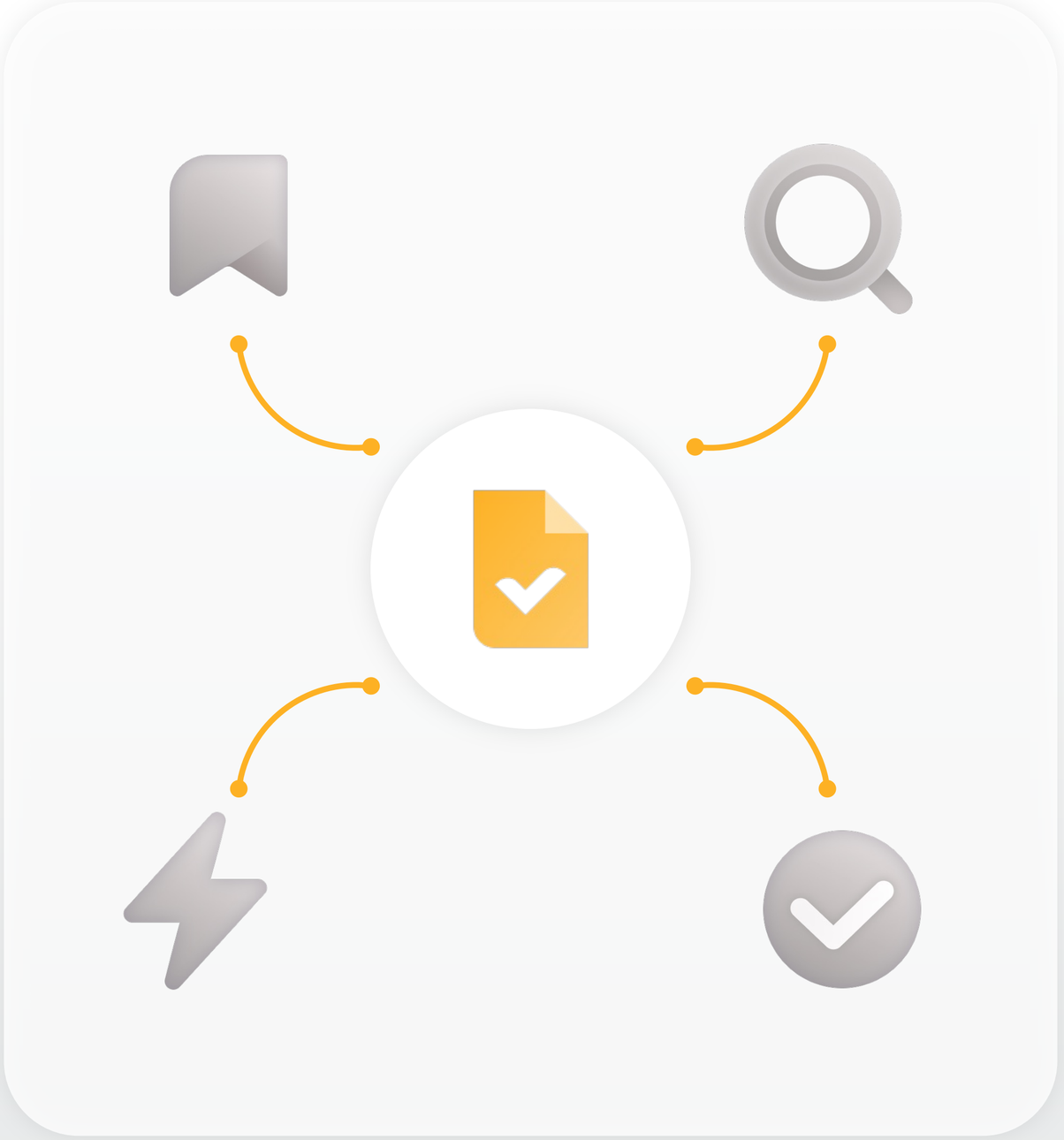
Вопрос из review

Один productId указан дважды

Выбранное правило

Заказ не создан.
Остатки неизменны.

Решение принято автором.
В Dev Spec ещё предстоит внести.



Теперь можно писать код?

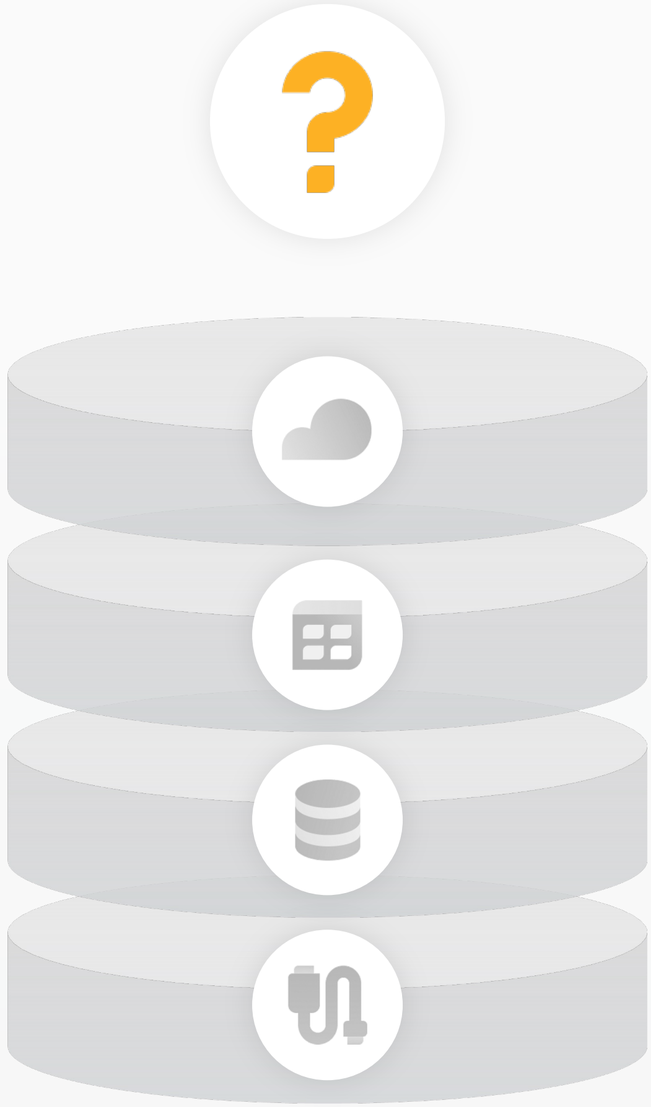
Нет

Сначала нужна маленькая единица реализации

Горизонтальная декомпозиция

Какое поведение уже работает?

- Entities
- Repositories
- Use cases
- Controllers
- Tests



Вертикальные слайсы

Наблюдаемое поведение и необходимая подготовка

002 **Отказ без записи**

Валидация покупателя,
товаров и количества

001 **Контракты**

порты, ошибки, fixtures

006 **Чтение заказа**

Сохранённые данные
на готовых fixtures

004 **Атомарная запись**

Подготовка:
заказ и остатки вместе

003 **Заказ со снимком цены**

Полный путь оформления
после зависимостей

005 **Резерв на 15 минут**

Будущее изменение
после нового требования

Skill: Dev Spec → Slices

Предлагаемый план реализации

01

Dev Spec

Поведение, scope
и инварианты
согласованы

02

spec-to-slices

Тип работы
Границы
Зависимости

03

Файлы плана

001, 002, 003...
Требования
и проверка

04

Границы Skill

Не меняет правила
Не пишет код
План идёт на review

Нумерация идентифицирует работу. Порядок задают зависимости

Анатомия одного slice

003-order-pricing.md

Результат	Заказ со снимком цены
Основание	FR-1, FR-5, FR-6
Зависимости	001, 002, 004, 006
Приемка	Каталог: 1500, затем 1700 В заказе: 1500 × 2 = 3000
Границы	Без скидок, оплаты и резерва

Предлагаемый slice. 003 берём
после готовности его зависимостей

Проверьте план до исполнения

review-slices

Что проверяем в slices

Полнота
FR/NFR покрыты проверками

Готовность slice
Выполнены AC, инварианты сохранены

Ответственность
Ясно, кто реализует и кто проверяет

Готовность feature
Общий сценарий и конкуренция проверены

Зависимости
Приёмка опирается на готовые контракты

Параллелизм
Нет скрытых пересечений и общего состояния

Тип работы
Поведение или необходимая подготовка

Размер
Работу можно отдельно выполнить и принять

Review отдельного slice

```
003-order-pricing.md

Критерий приёмки:
создать заказ, изменить каталог,
прочитать сохранённую цену.

Зависимости:
001, 002, 004, 006
```



Что нашёл review

Приёмка требует
записи и чтения заказа

**003 выполняется
после зависимостей**

Human checkpoint №2

01

Slices

План небольших наблюдаемых результатов

02

AI-review

Полнота, размер, зависимости и пересечения

03

Human review

Команда проверяет план и устраняет неоднозначности

04

Approval

Человек фиксирует решение:
план утверждён

Review ищет проблемы. Approval фиксирует решение

Планирование

- FR-5 в Dev Spec
- 003 и его зависимости
- Готовность к работе

Правило в Dev Spec

Исполнение

- Change для 003
- Точки изменения в коде
- Проверка сохранённой цены

Детали исполнения в Change

Execution contract

Что такое OpenSpec

Зачем меняем proposal.md

На какие требования опираемся

Какие задачи и проверки

003 и FR-5/FR-6 связаны с Change

Change: order-pricing

```
openspec/
└── changes/
    └── order-pricing/
        ├── proposal.md
        ├── specs/
        └── tasks.md
```

Концептуальный пример.

Состав зависит от схемы OpenSpec

Skill: Slice → OpenSpec Change

FR-5: сохраняем цену заказа



Создать по 1500. Изменить каталог. Прочитать прежнюю цену.

OpenSpec Just-in-Time

01

003 в плане

Зависимости
001, 002, 004, 006
ещё проверяются

02

Готово к работе

Контракты готовы.
Проверки зависи-
мостей пройдены

03

Change JIT

Читаем текущий код.
Уточняем исполнение
выбранного slice

04

В работе

Реализация
по Change
и проверка FR-5

Показан целевой workflow. Это не статус выполненных работ

Было

Создать заказ

Сразу уменьшить доступный остаток

Оплата и резерв вне исходного scope

Расширение scope требует решений

Стало

Резерв на 15 минут

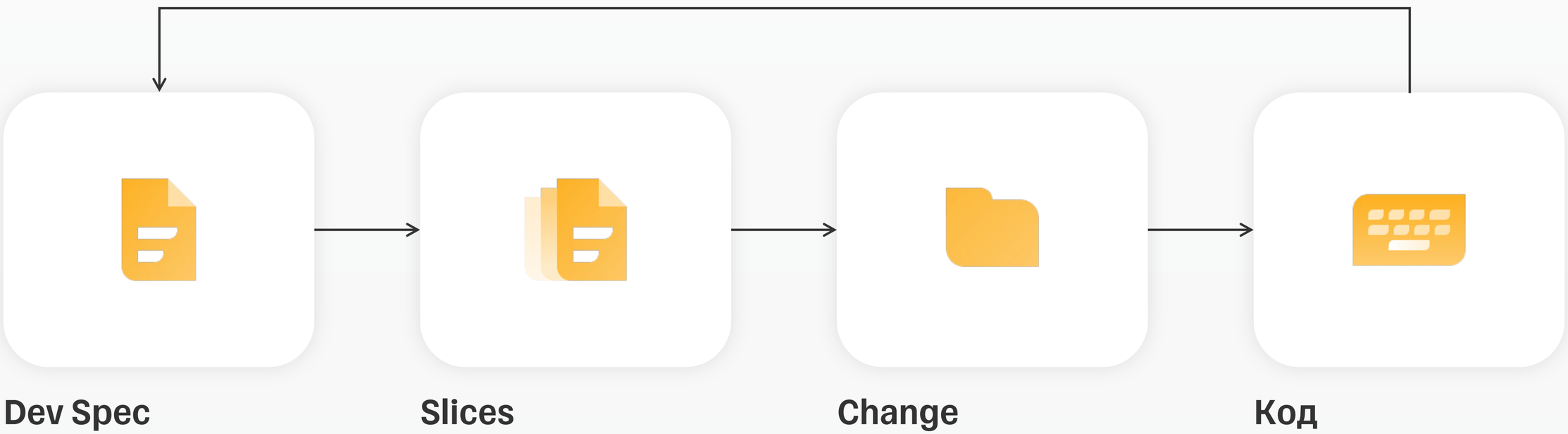
Освободить, если не оплачен

Когда начинается отсчёт?

Что при оплате на границе срока?

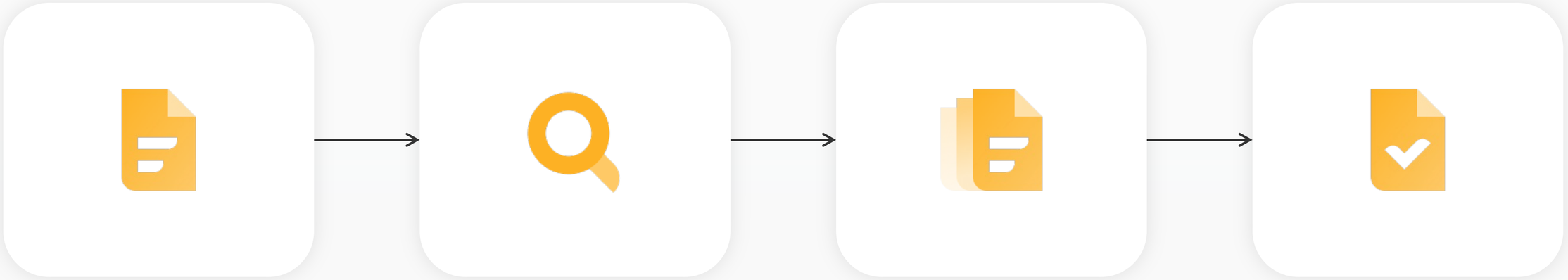
Куда возвращаться?

Сначала согласуем новое правило



Время резерва и подтверждение не должны появиться как догадка агента.

Как проходит изменение требований



Dev Spec

Анализ влияния

Затронутые работы

Review

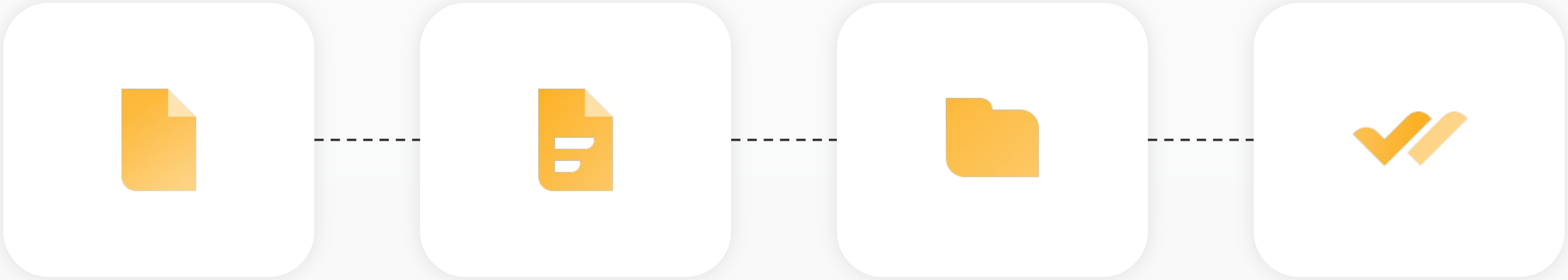
003 оформление
Создание заказа и резерв

004 атомарность
Остатки и переходы состояния

006 чтение
Нужные клиенту данные

005 — новая работа по резерву после согласования. Историю сохраняем.

Traceability



FR-5

003-order-pricing

Change

Проверка снимка

Каталог: 1500, затем 1700

Количество в заказе: 2

Сохранённый заказ

Цена 1500. Сумма 3000.

Ожидаемый результат сценария. Автоматическая проверка ещё не запускалась.

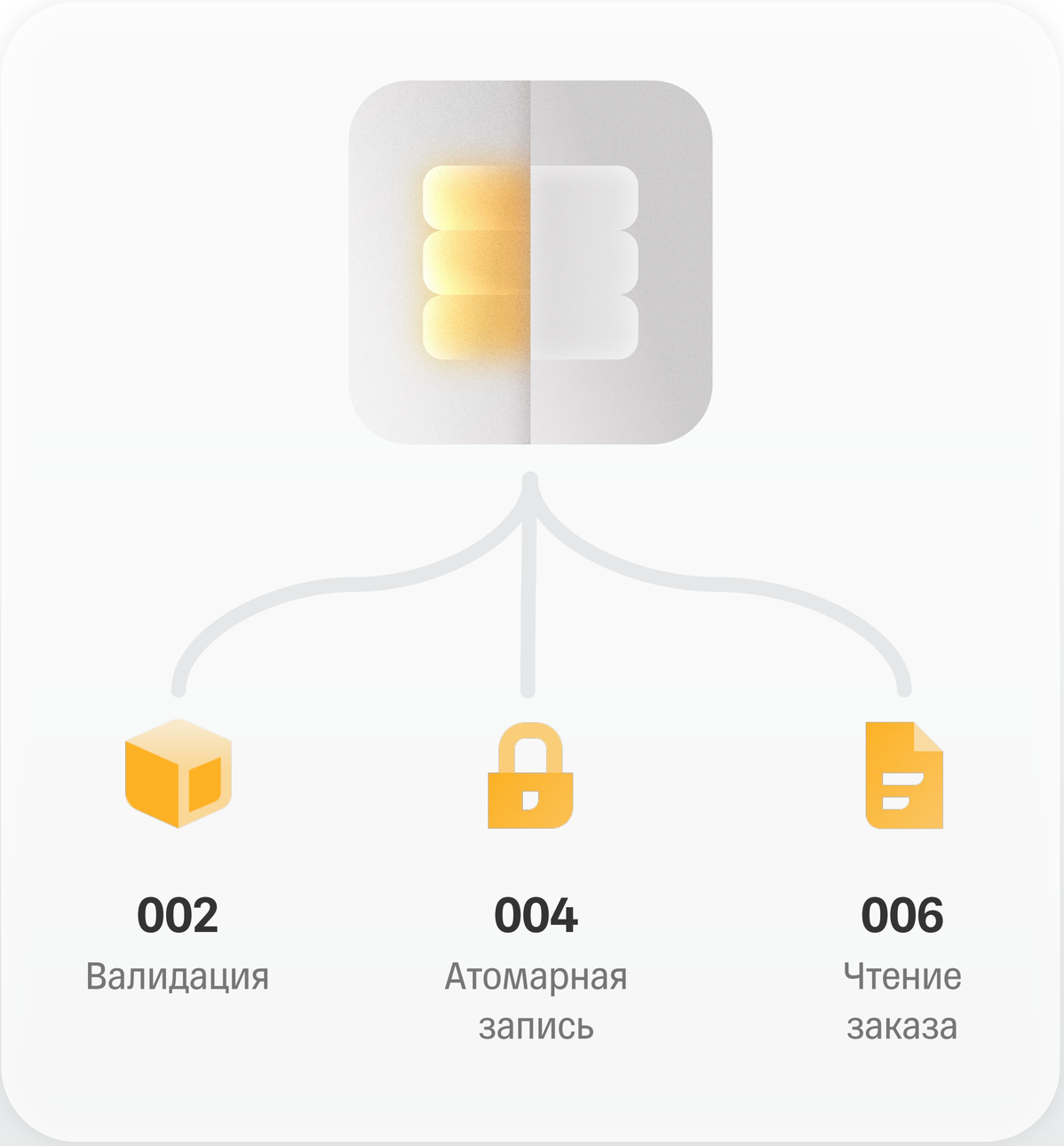
Slice как единица параллелизма

001. Общие контракты

002, 004 и 006 можно разделить после проверки границ

---

003 запускается после них



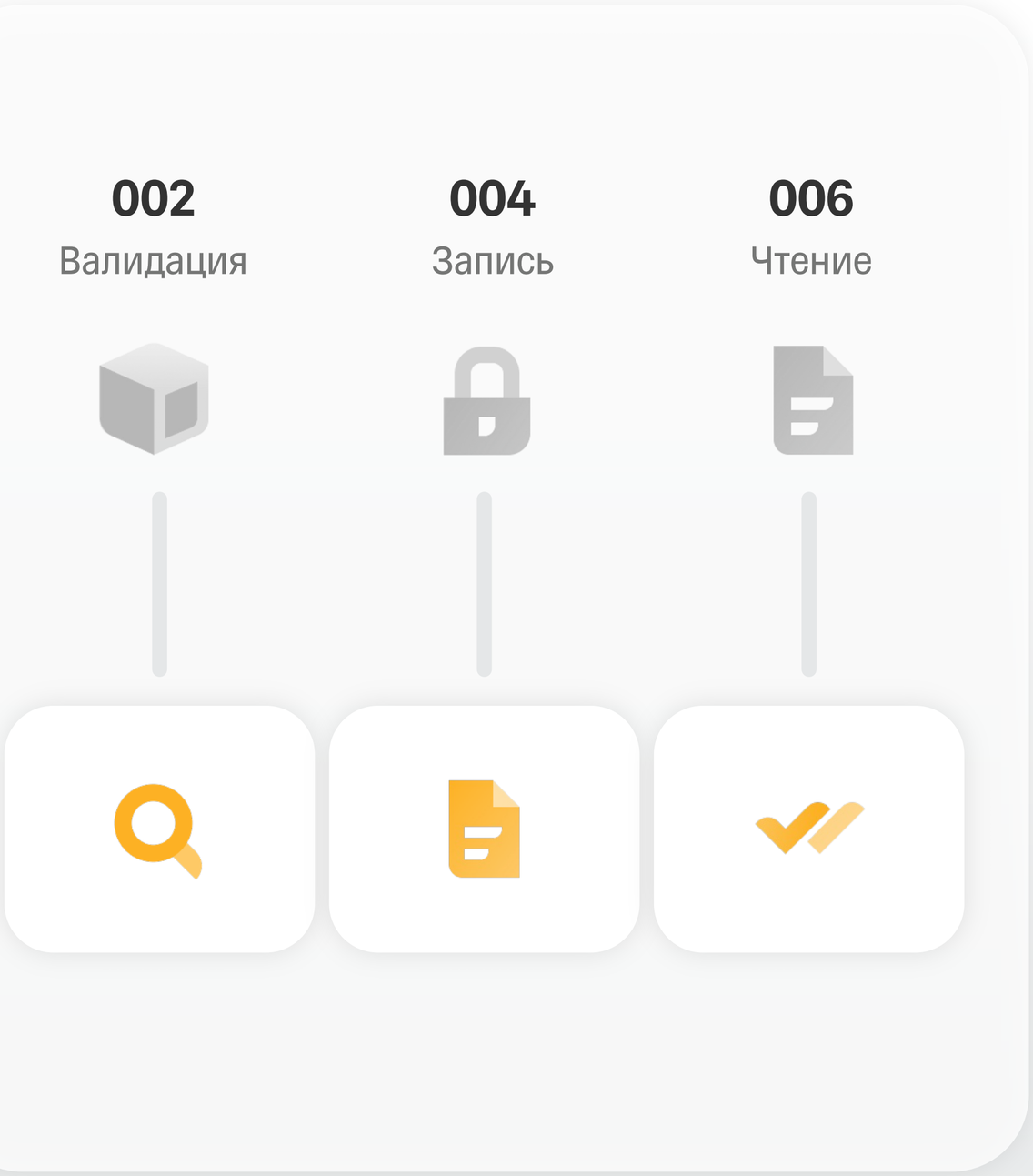
Slice как единица параллелизма

Общая проверка после сборки

Остаток 3
Два заказа по 2

Один заказ
Один отказ
Остаток 1

Ожидаемый сценарий FR-8/NFR-2.
Проверяем через собранный endpoint.





Git Worktrees

Change: order-pricing

```
repository
├── worktree/validation      002 · Agent A
├── worktree/atomic-write    004 · Agent B
└── worktree/get-order       006 · Agent C
```

Базы, порты и fixtures изолируем отдельно

Общая ветка: интеграция, затем 003 и общие проверки

Предлагаемая организация. Эти worktrees ещё не созданы

Agent Teams

Lead Agent

- Распределяет работу
- Следит за контрактами
- Собирает результат

После интеграции:
003 и общие проверки



Не параллельте хаос

Сначала устойчивый цикл одного изменения



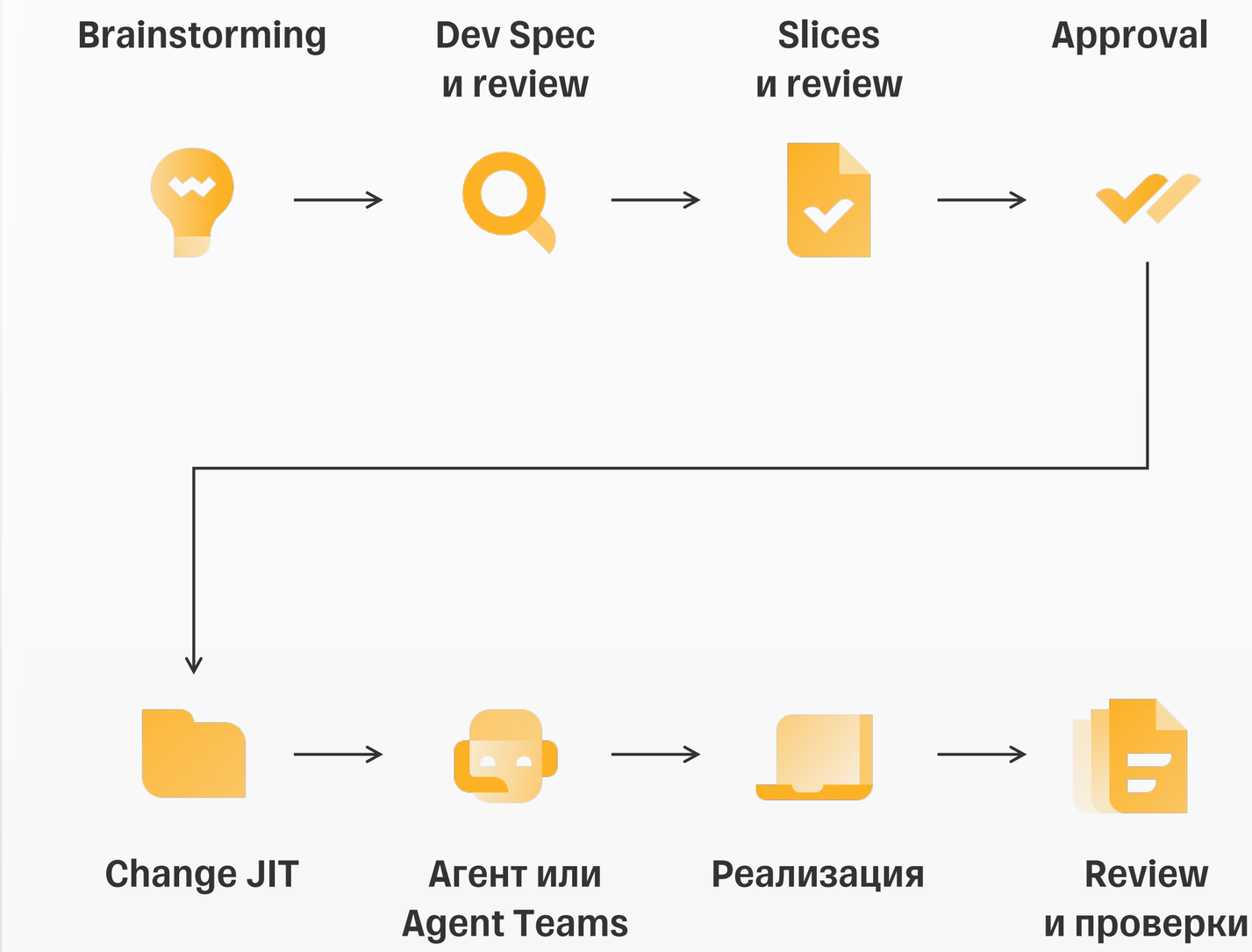
Затем несколько агентов

Весь процесс

Проверяем поведение

FR-5
Цена сохранена

NFR-2
Нет overselling



Что где хранится

AGENTS.md	Постоянные правила
docs/<feature>/spec.md	Бизнес-поведение
docs/<feature>/slices/	Работы и зависимости
openspec/changes/	Текущее исполнение
.agents/skills/	Повторяемые операции
src/ + tests/	Код и проверки поведения

Глубина описания зависит от размера и риска изменения

Целевая структура репозитория

Уровни зрелости

1. Промпт и код
2. Правила проекта
3. Dev Spec
4. Slices и Changes
5. Параллельные агенты

**Чем больше работы отдаём AI,
тем важнее процесс, который
ограничивает и проверяет его работ**

Попробуйте завтра

- Одна небольшая feature
- Brainstorming и Dev Spec
- Review и vertical slices
- Один Change JIT
- Реализация и проверка



Три главные мысли

Вместо длительных итогов

Контекст — главный ресурс

Принимайте решение, что попадёт в контекст: CLAUDE.md, скилл, субагент

Спецификация дешевле кода

Проектируйте, выравнивайтесь по требованиям с AI до имплементации

Стройте систему, не диалог

Скиллы, субагенты и хуки превращают агента в инженерную платформу

Спасибо за внимание!



No-code
Python
JavaScript
Node.js
Swift
Mobile App
Development
Innovation
Golang
Java
Analysis
Planning