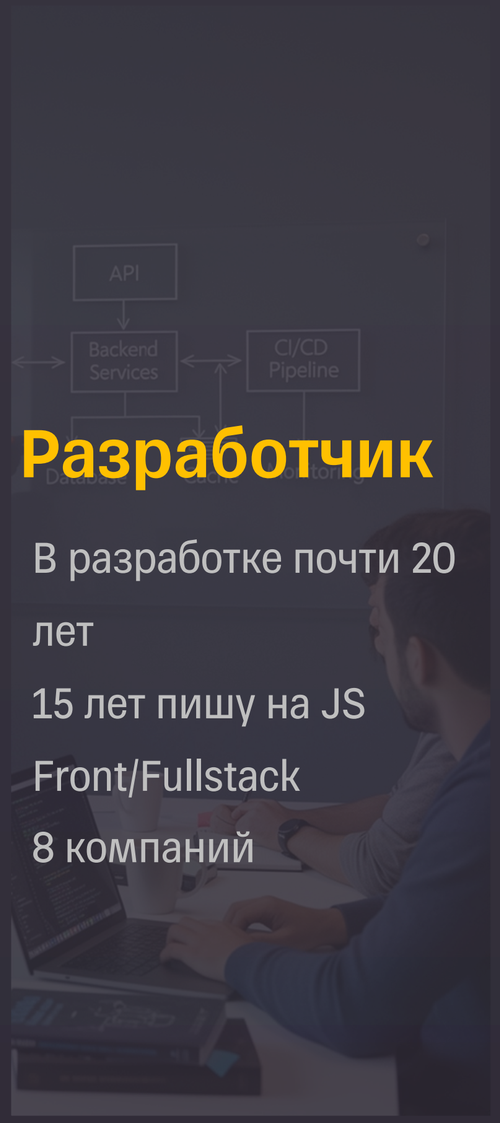


System Design шавермы

Василий Ванчук, ведущий эксперт Т-Банк



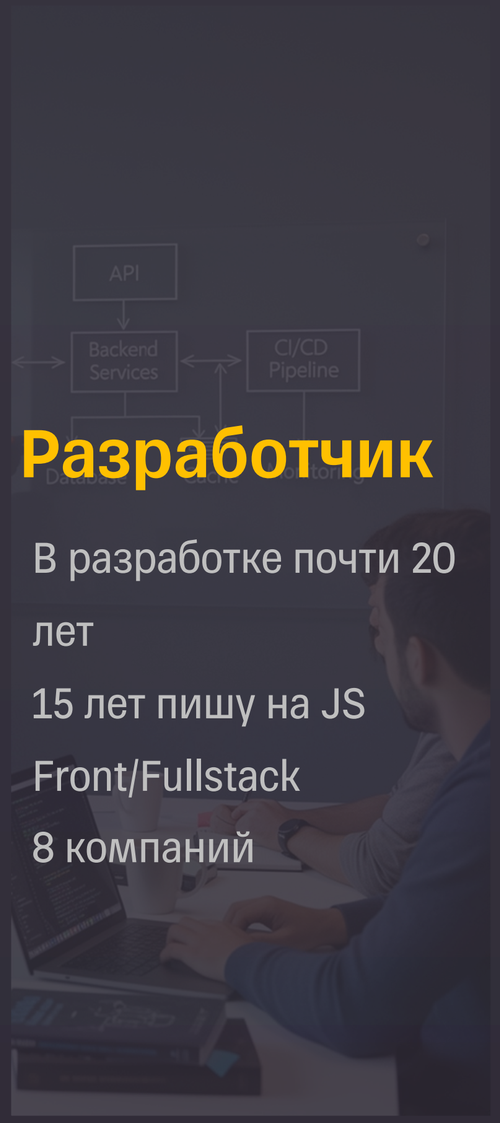
Разработчик

В разработке почти 20 лет

15 лет пишу на JS

Front/Fullstack

8 компаний



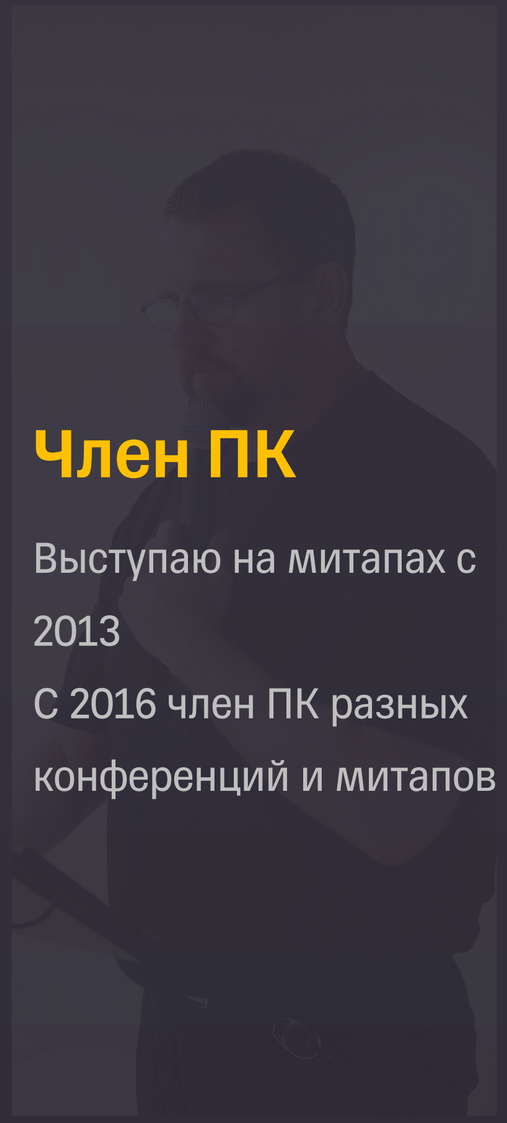
Разработчик

В разработке почти 20 лет

15 лет пишу на JS

Front/Fullstack

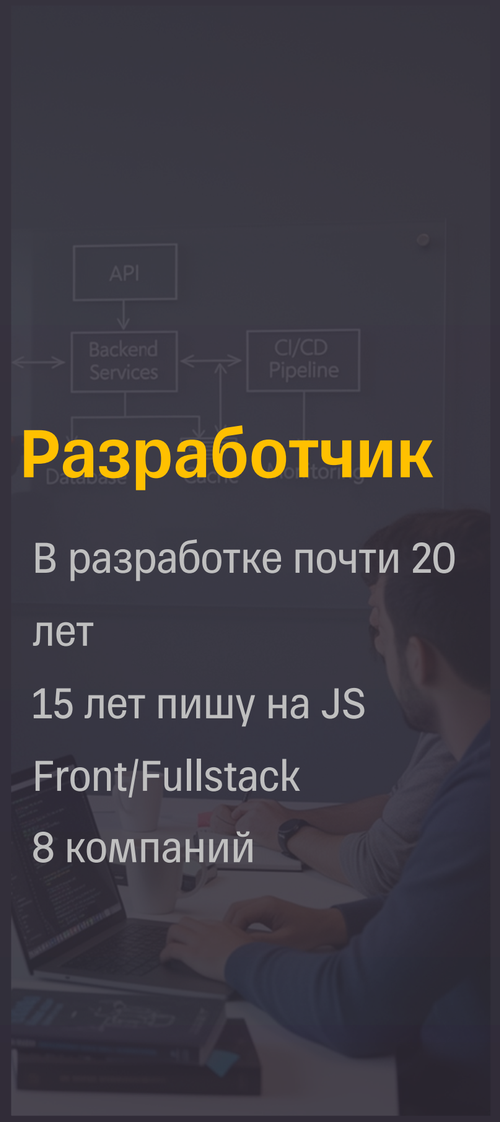
8 компаний



Член ПК

Выступаю на митапах с 2013

С 2016 член ПК разных конференций и митапов



Разработчик

В разработке почти 20 лет

15 лет пишу на JS Front/Fullstack

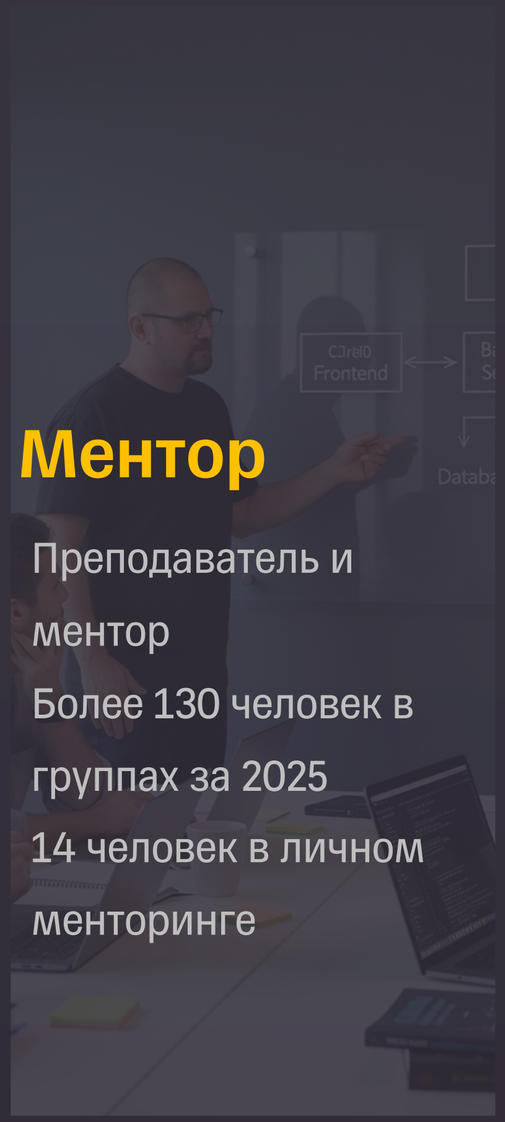
8 компаний



Член ПК

Выступаю на митапах с 2013

С 2016 член ПК разных конференций и митапов

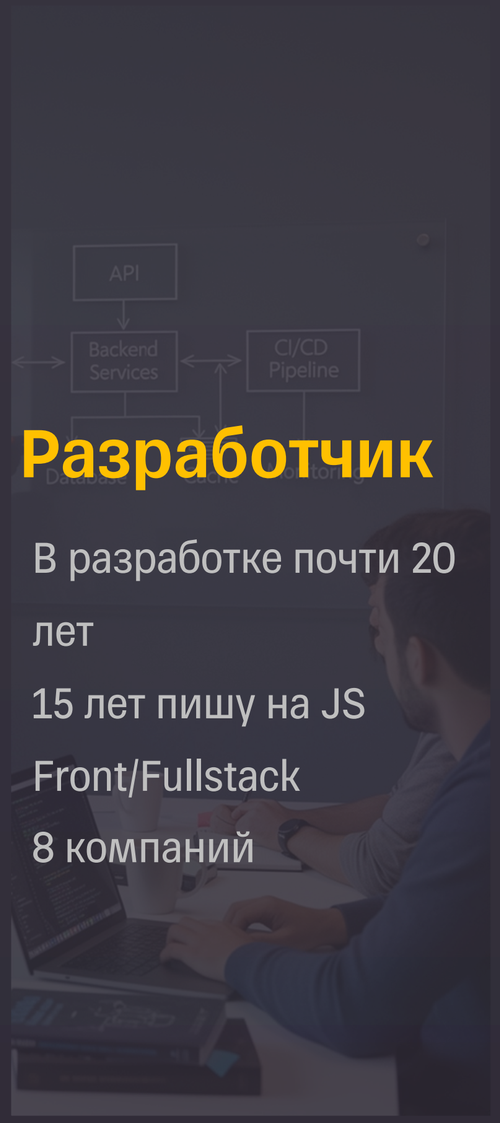


Ментор

Преподаватель и ментор

Более 130 человек в группах за 2025

14 человек в личном менторинге

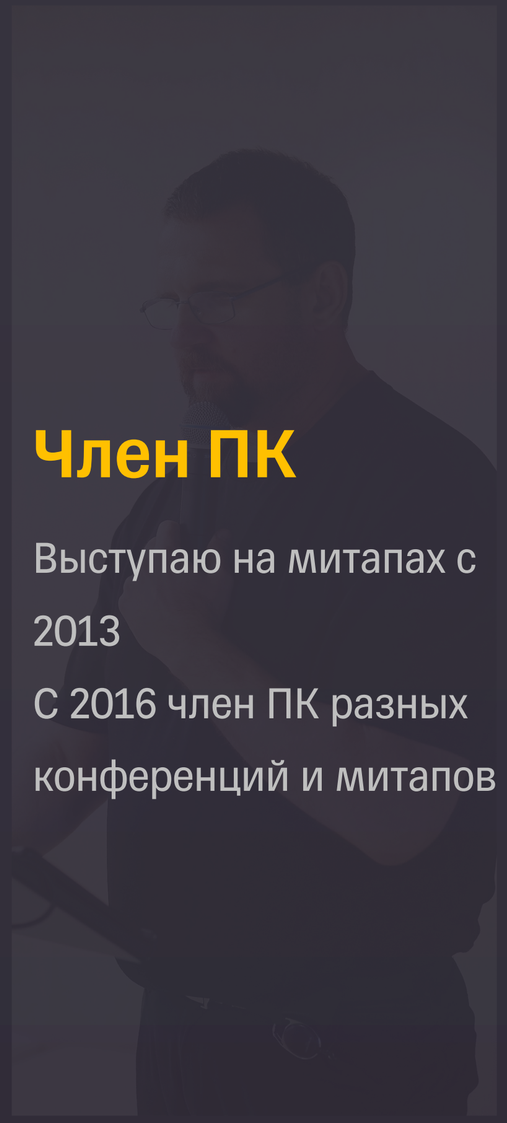


Разработчик

В разработке почти 20 лет

15 лет пишу на JS Front/Fullstack

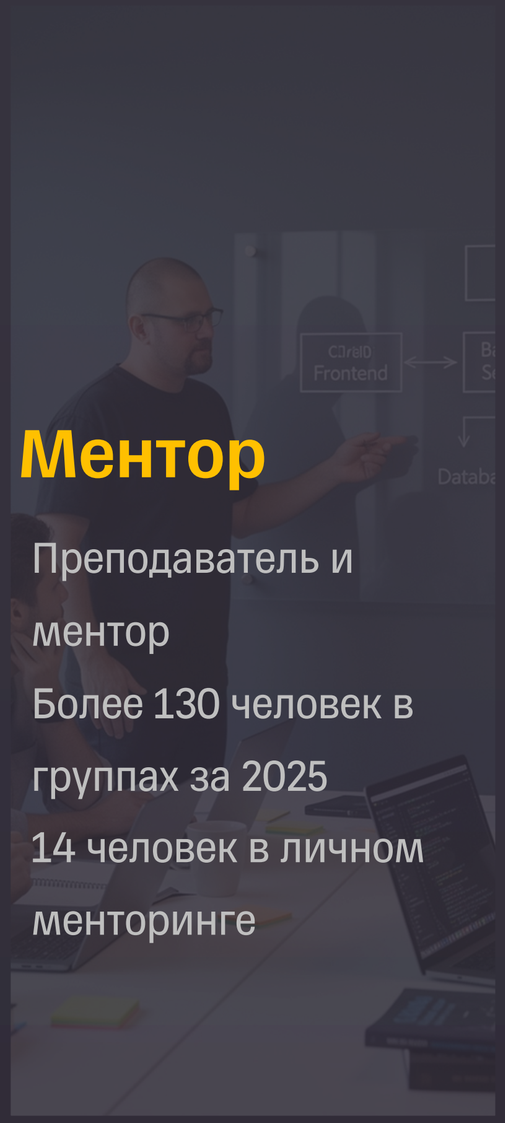
8 компаний



Член ПК

Выступаю на митапах с 2013

С 2016 член ПК разных конференций и митапов



Ментор

Преподаватель и ментор

Более 130 человек в группах за 2025

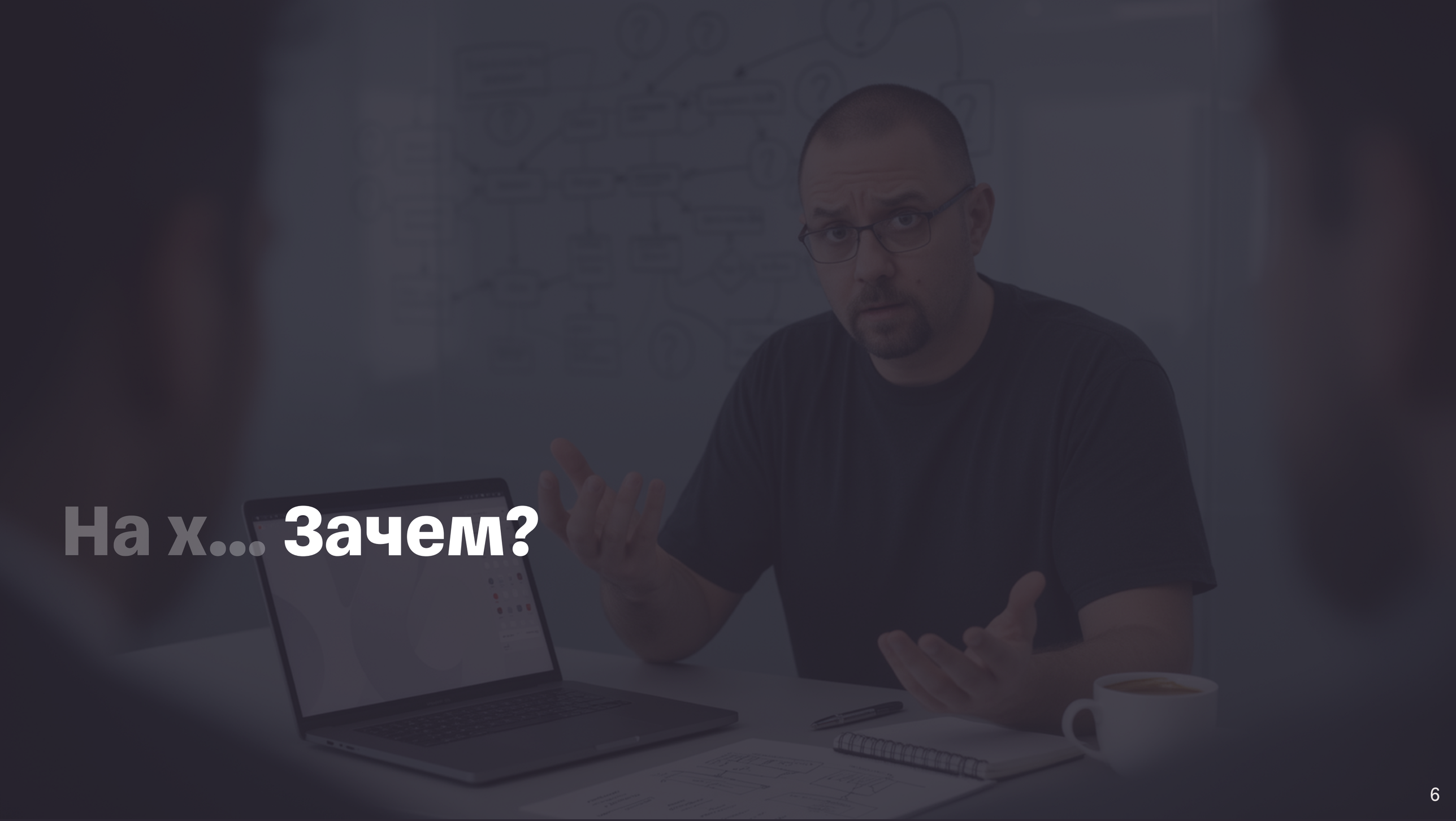
14 человек в личном менторинге



Интервьюер

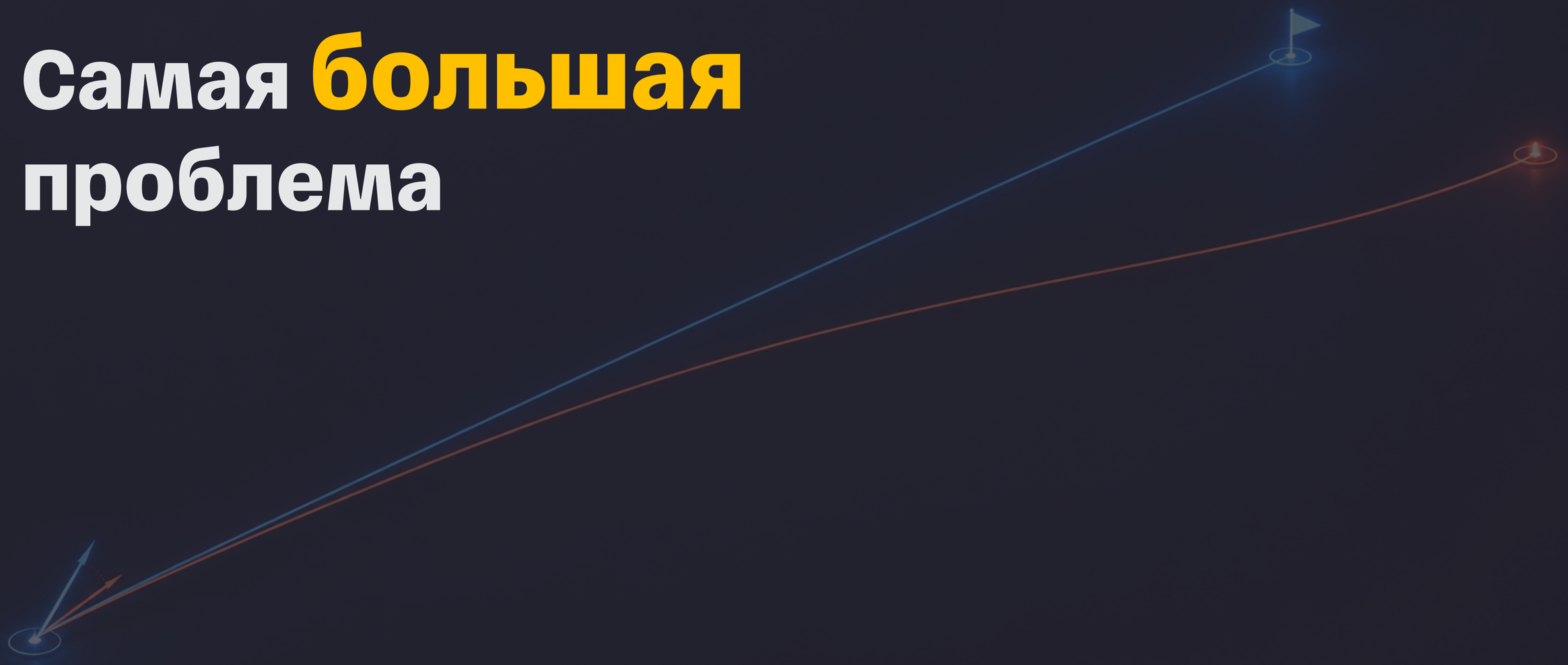
150+ собеседований за последние 2 года

В прошлом лид WA трека для FE/FS



На х... Зачем?

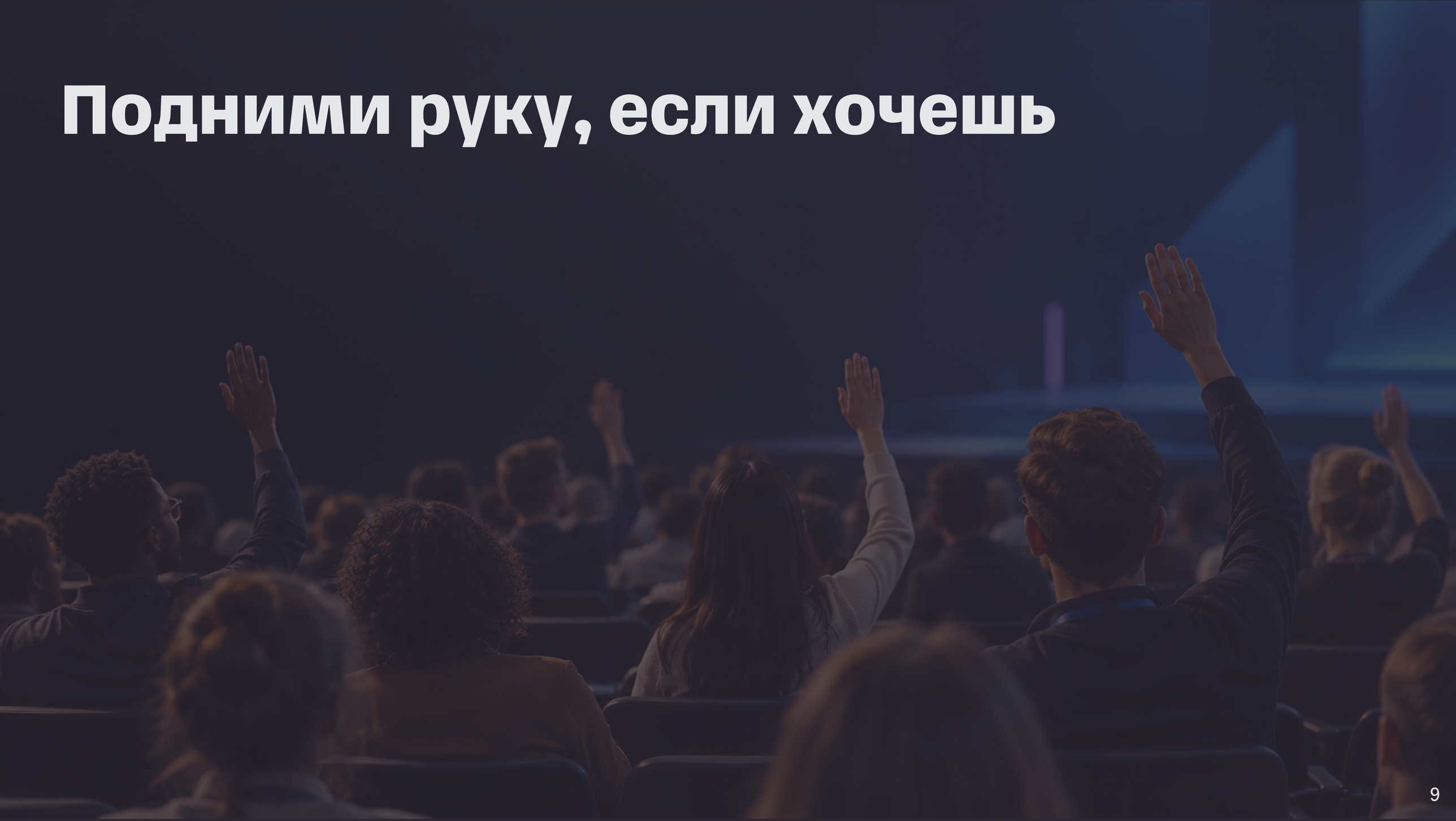
Самая **большая** проблема



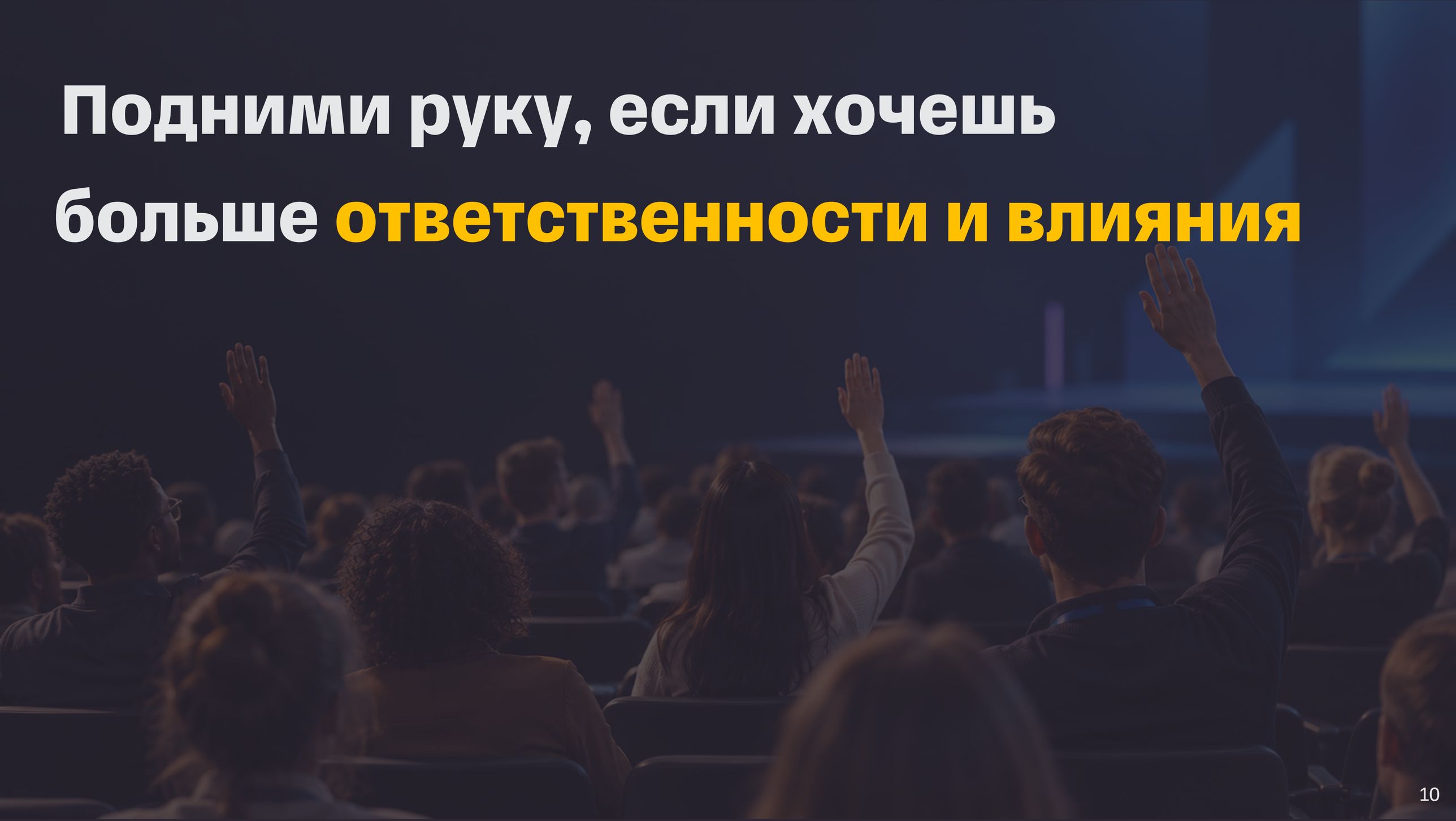
Будем проще



Подними руку, если хочешь



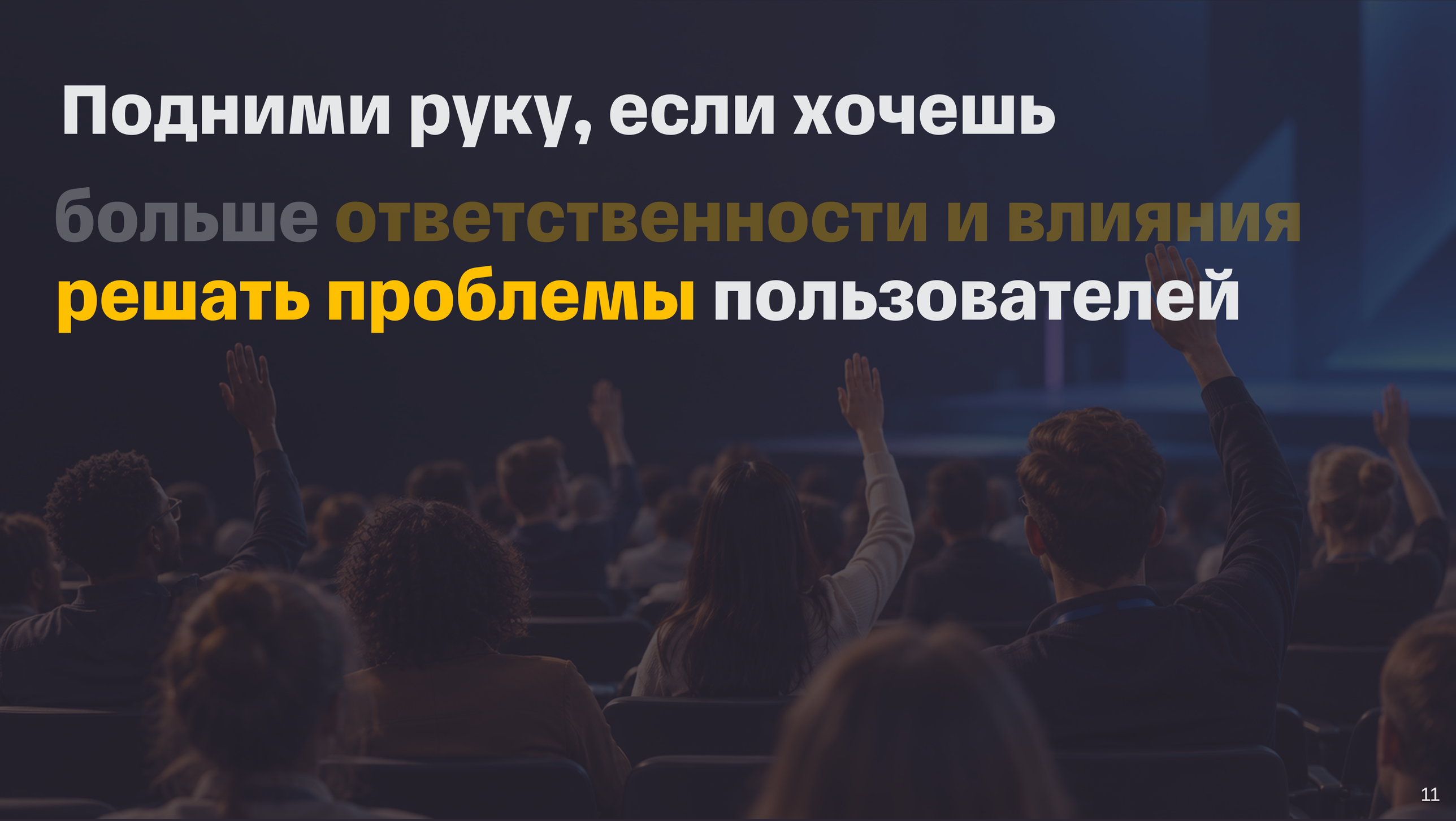
Подними руку, если хочешь больше ответственности и влияния



Подними руку, если хочешь

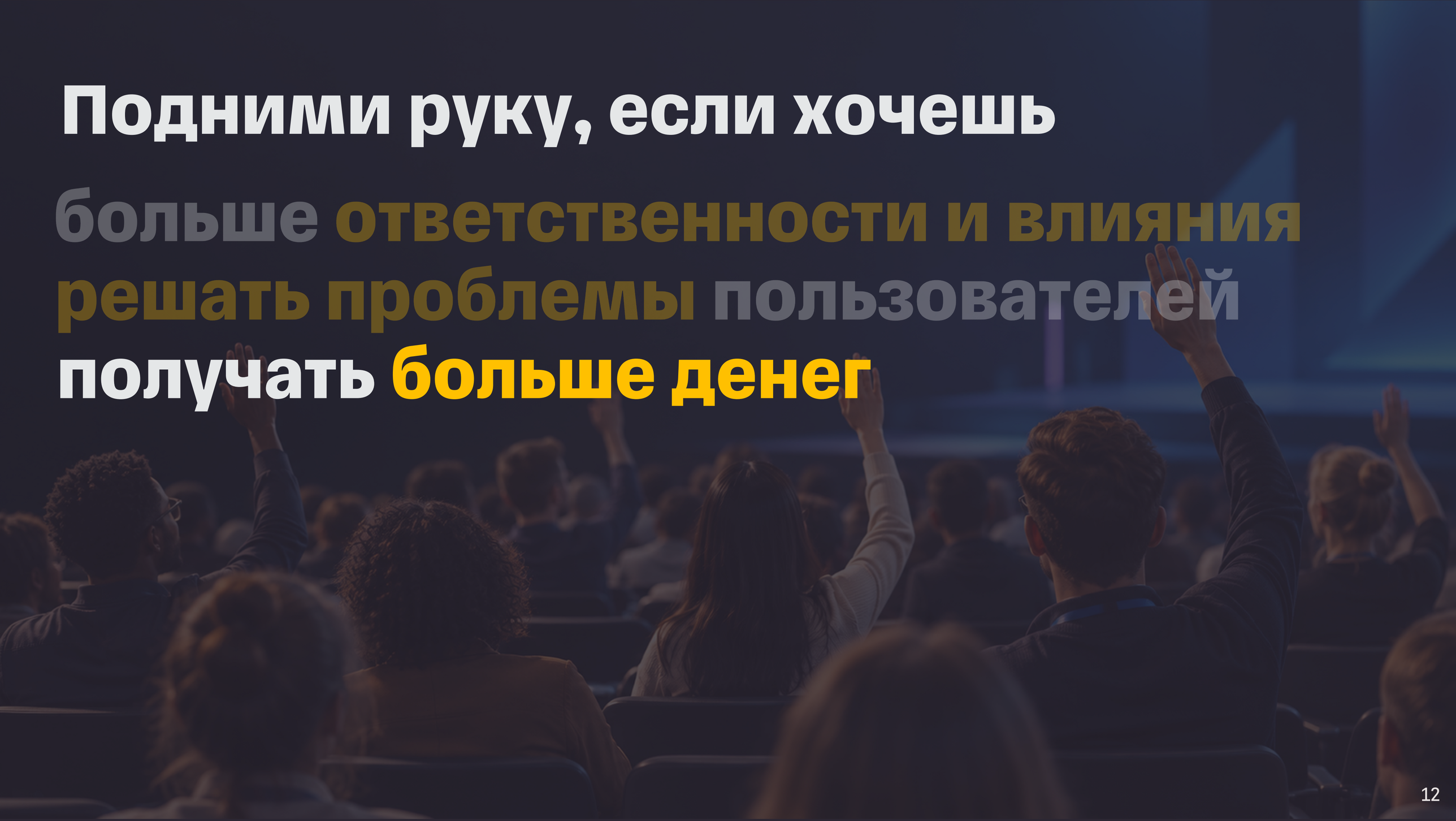
больше ответственности и влияния

решать проблемы пользователей



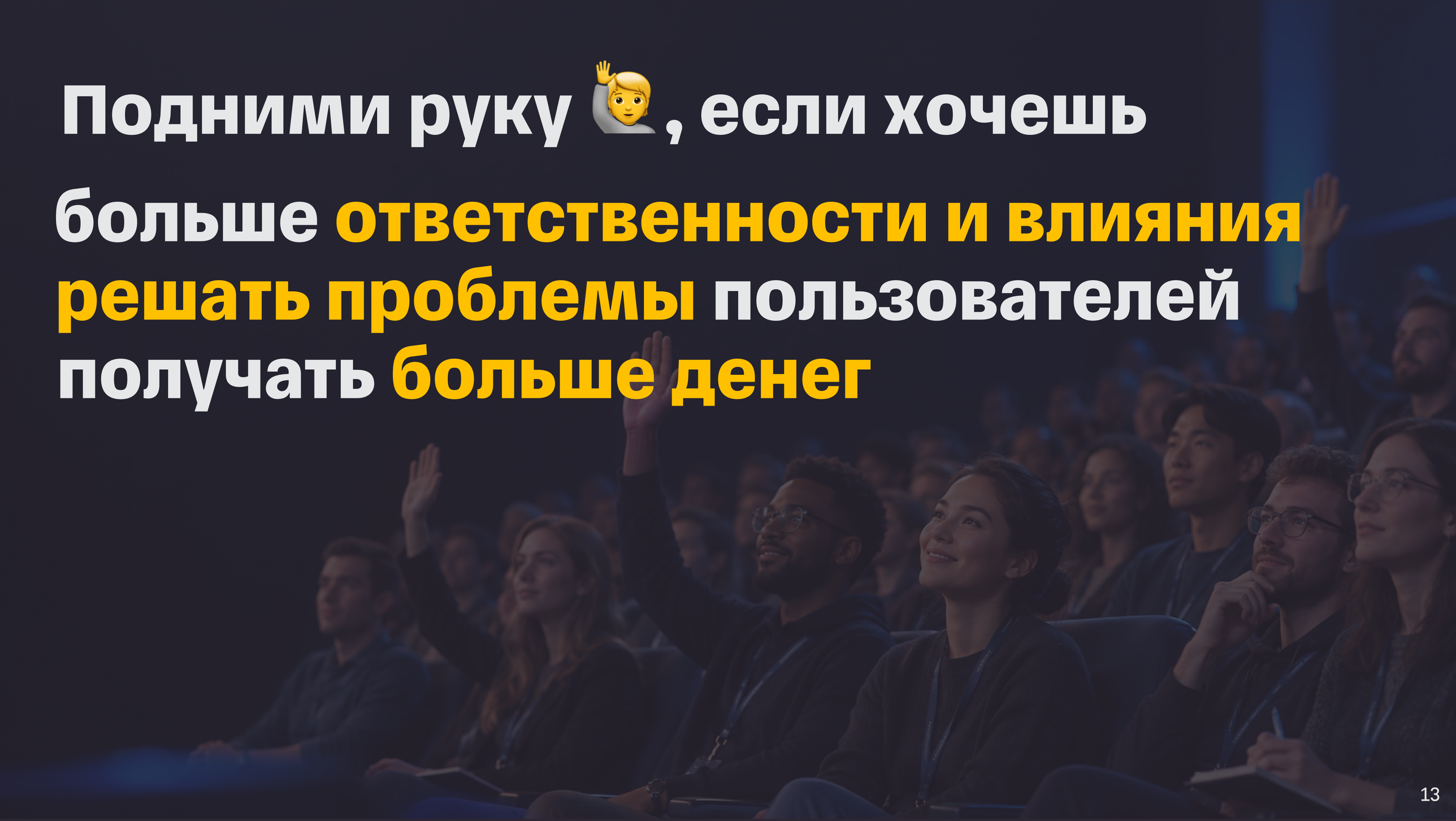
Подними руку, если хочешь

больше ответственности и влияния
решать проблемы пользователей
получать больше денег



Подними руку 🙋, если хочешь

больше **ответственности и влияния**
решать проблемы пользователей
получать **больше денег**



Выстроить границы



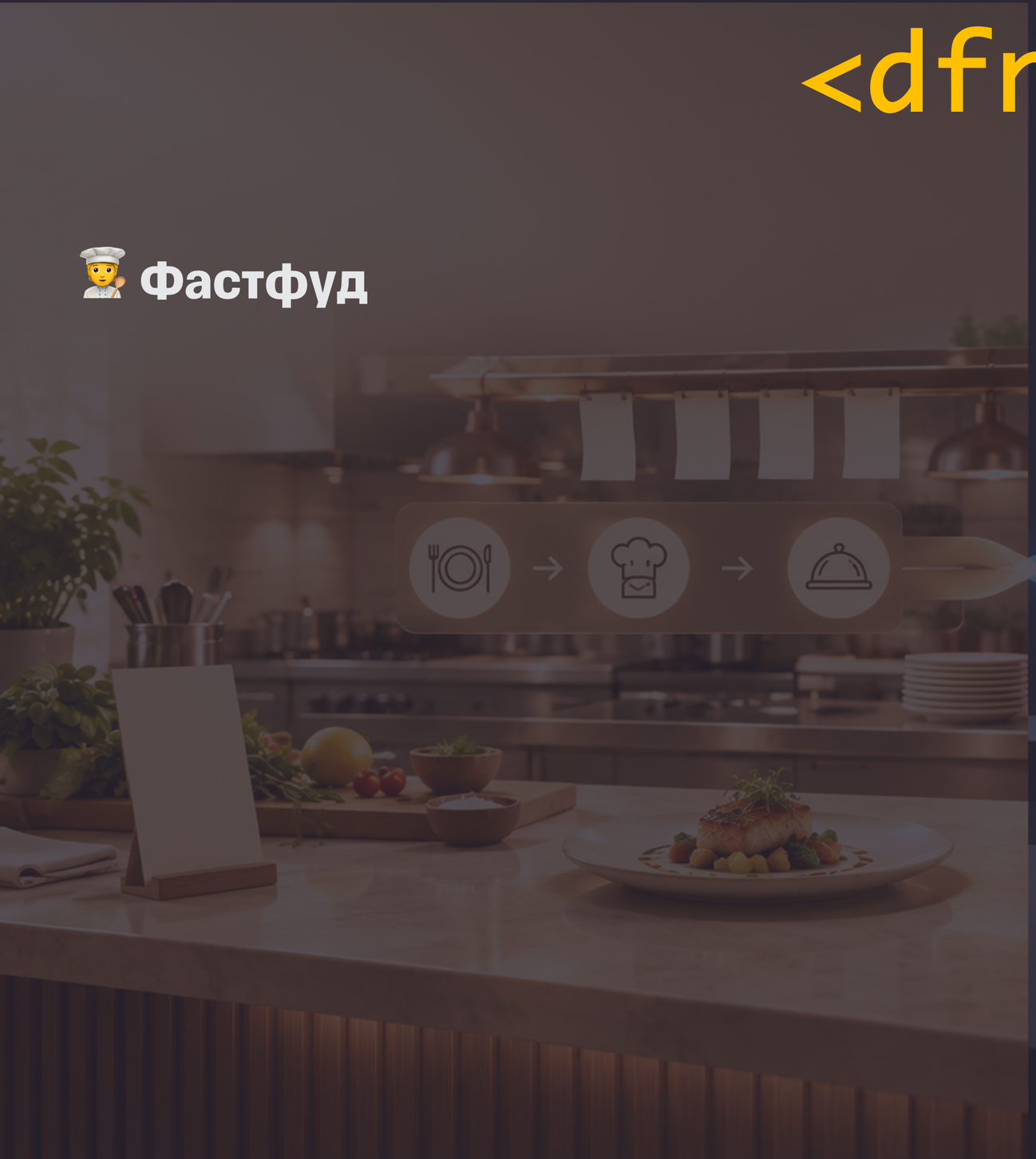
<dfn />



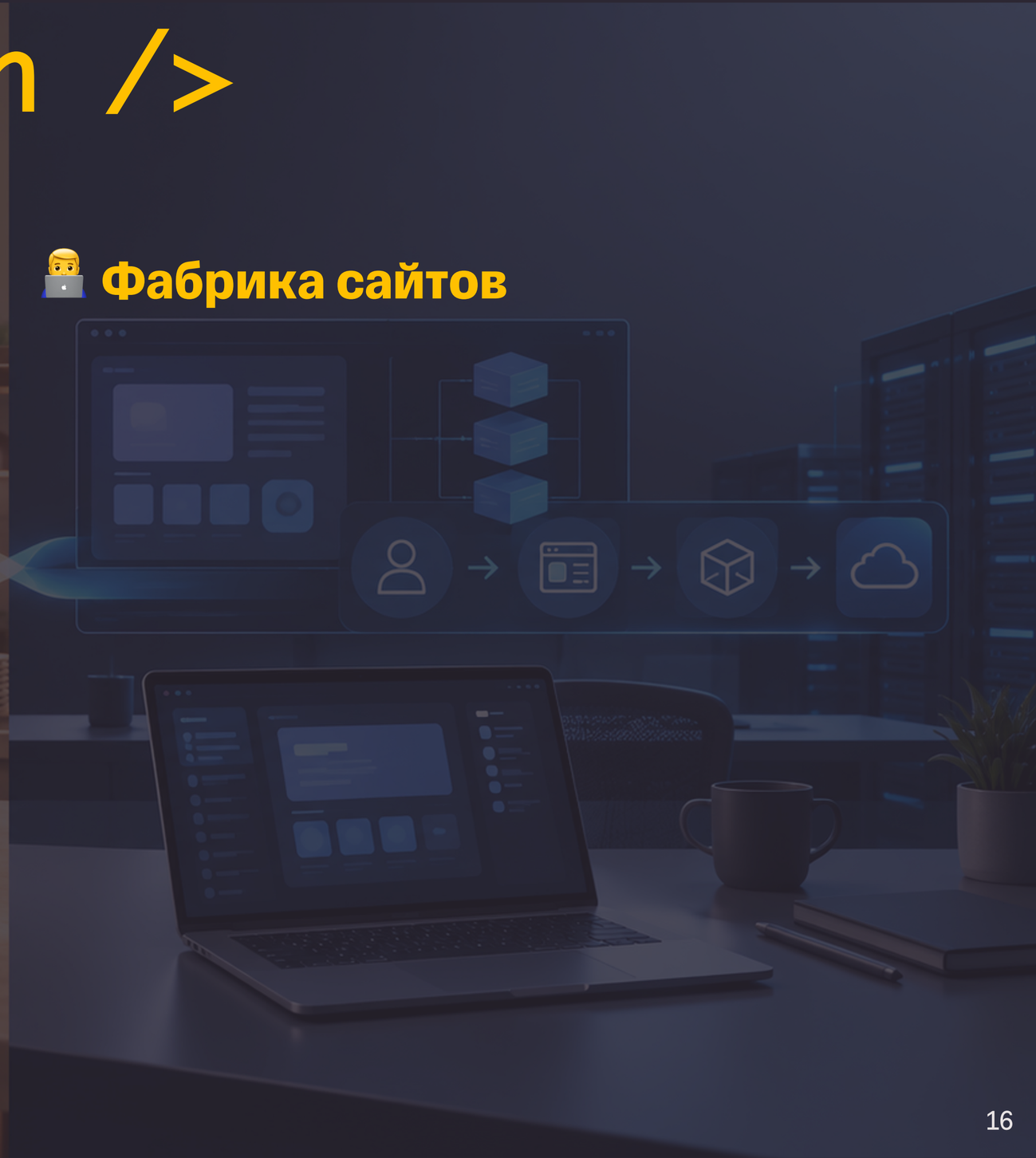


<dfn />

👨‍🍳 Фастфуд



👨‍💻 Фабрика сайтов



<dfn />

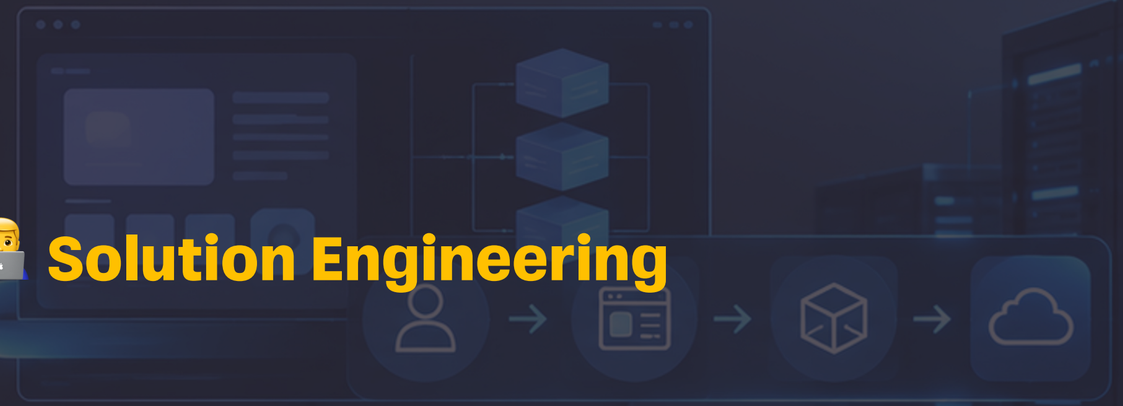
👨‍🍳 Фастфуд

👨‍🍳 Fine-dining



👨‍💻 Фабрика сайтов

👨‍💻 Solution Engineering

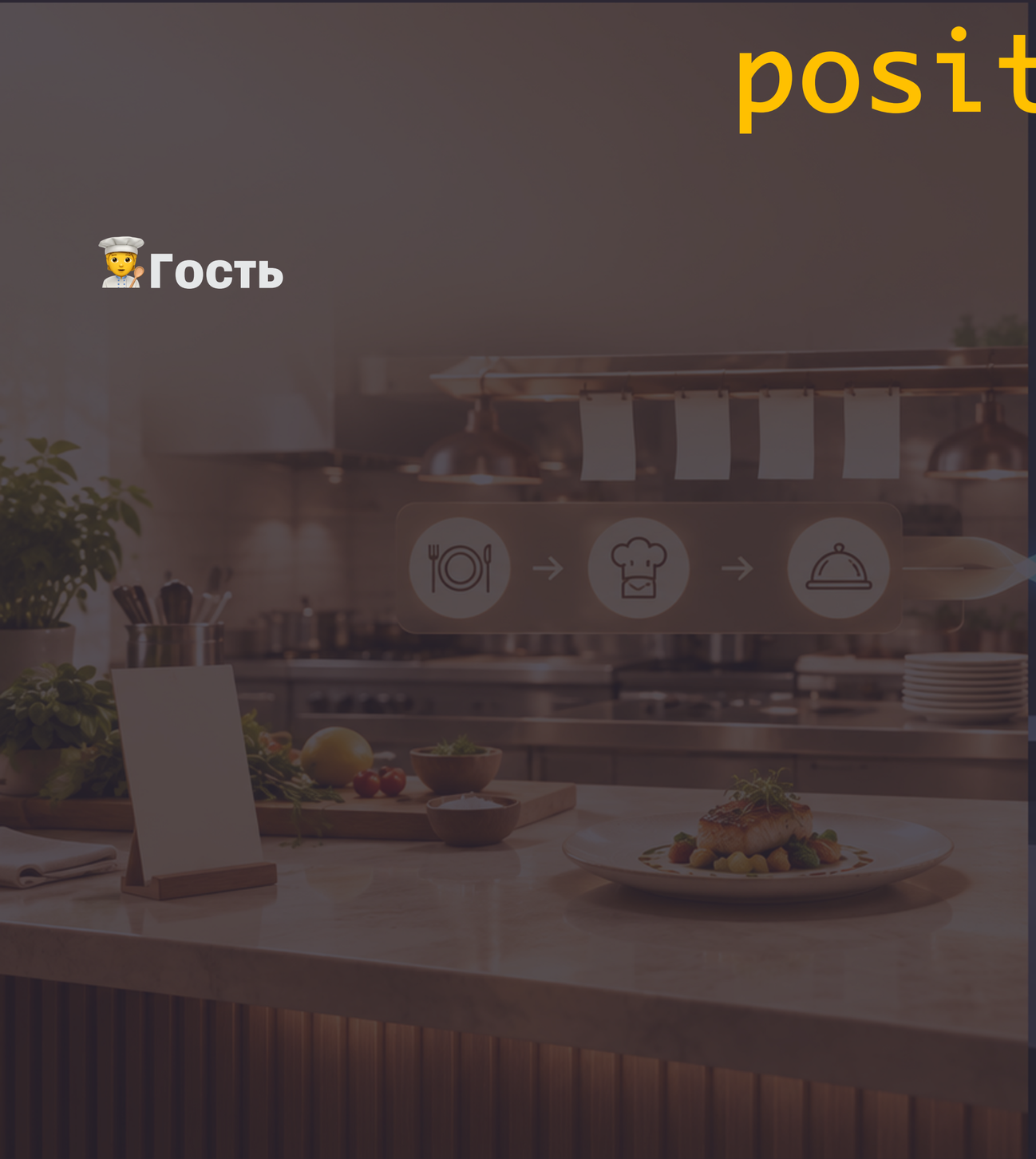


position:



position:

👨‍🍳 Гость



👨‍💻 Клиент



position:

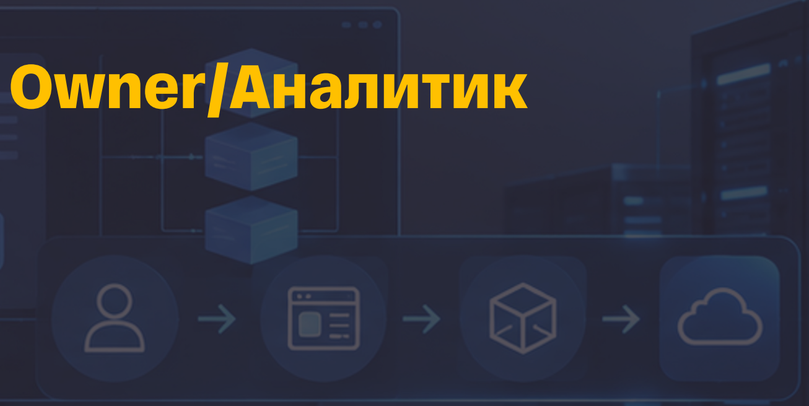
👨‍🍳 **Гость**

👨‍🍳 **Официант**



👨‍💻 **Клиент**

👨‍💻 **Product Owner/Аналитик**



position:

👨‍🍳 Гость

👨‍🍳 Официант

👨‍🍳 Шеф-повар/технолог

👨‍💻 Клиент

👨‍💻 Product Owner/Аналитик

👨‍💻 Архитектор

position:

- 👨‍🍳 Гость
- 👨‍🍳 Официант
- 👨‍🍳 Шеф-повар/технолог
- 👨‍🍳 Повар

- 👨‍💻 Клиент
- 👨‍💻 Product Owner/Аналитик
- 👨‍💻 Архитектор
- 👨‍💻 Разработчик

position:

- 👨‍🍳 Гость
- 👨‍🍳 Официант
- 👨‍🍳 Шеф-повар/технолог
- 👨‍🍳 Повар
- 👨‍🍳 Ресторан

- 👨‍💻 Клиент
- 👨‍💻 Product Owner/Аналитик
- 👨‍💻 Архитектор
- 👨‍💻 Разработчик
- 👨‍💻 Команда

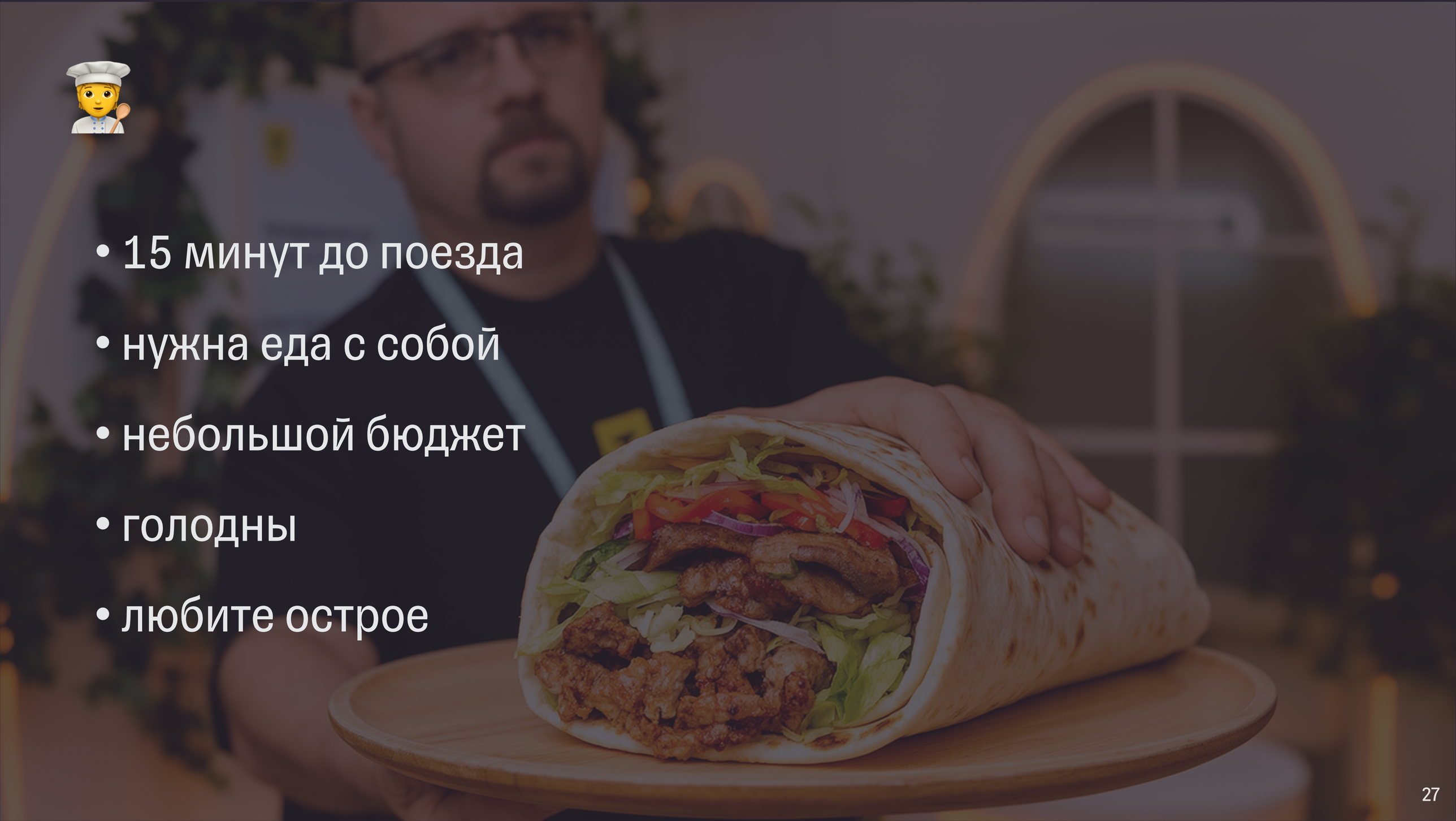


👩‍🍳 Выбора нет

👩‍🍳 Беседа с официантом

- Ваши планы на вечер?
- Сколько у вас времени?
- Вы один?
- Есть аллергии?
- Бюджет?
- Быстро перекусить, отметить событие или спокойно посидеть?

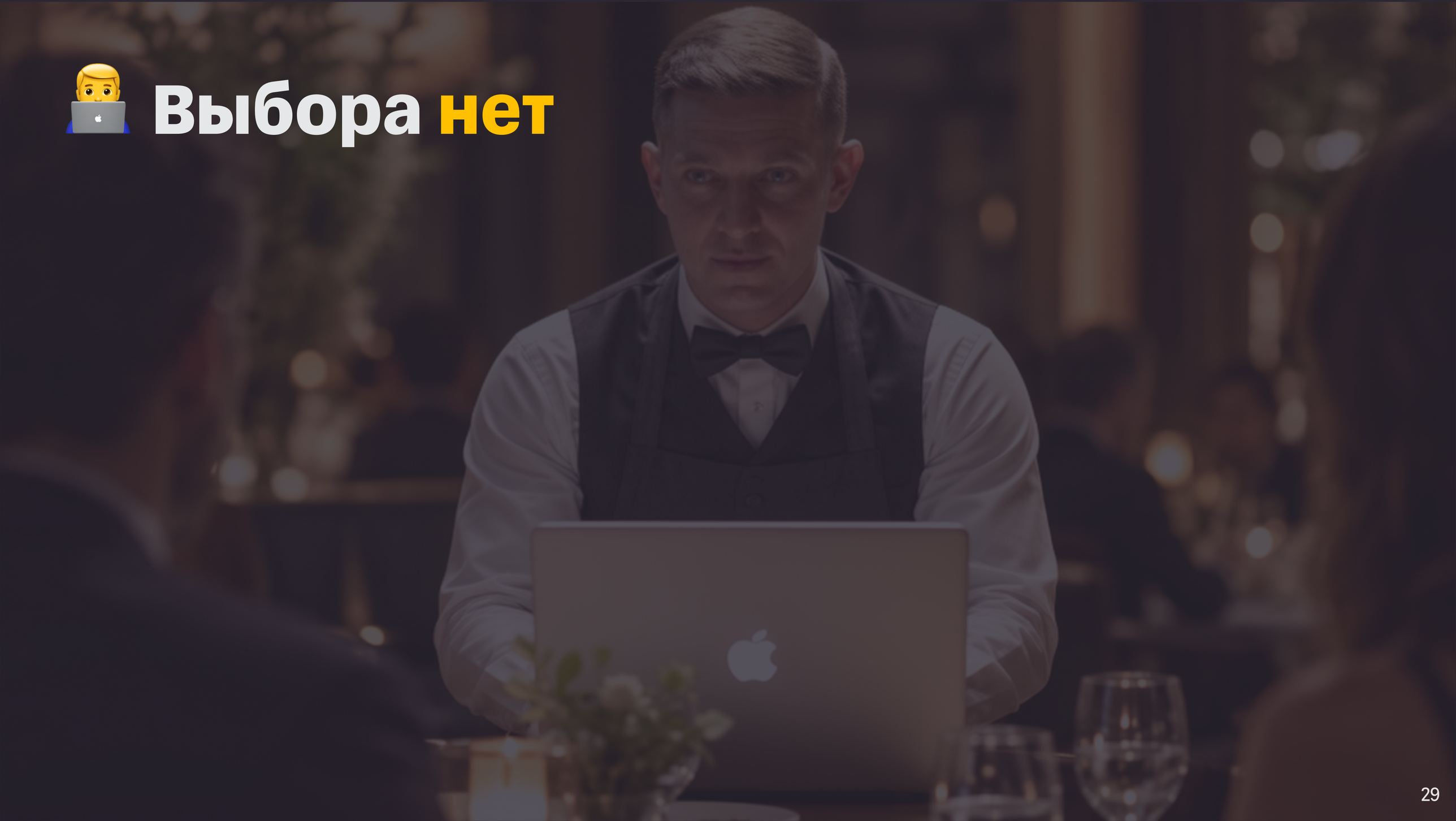


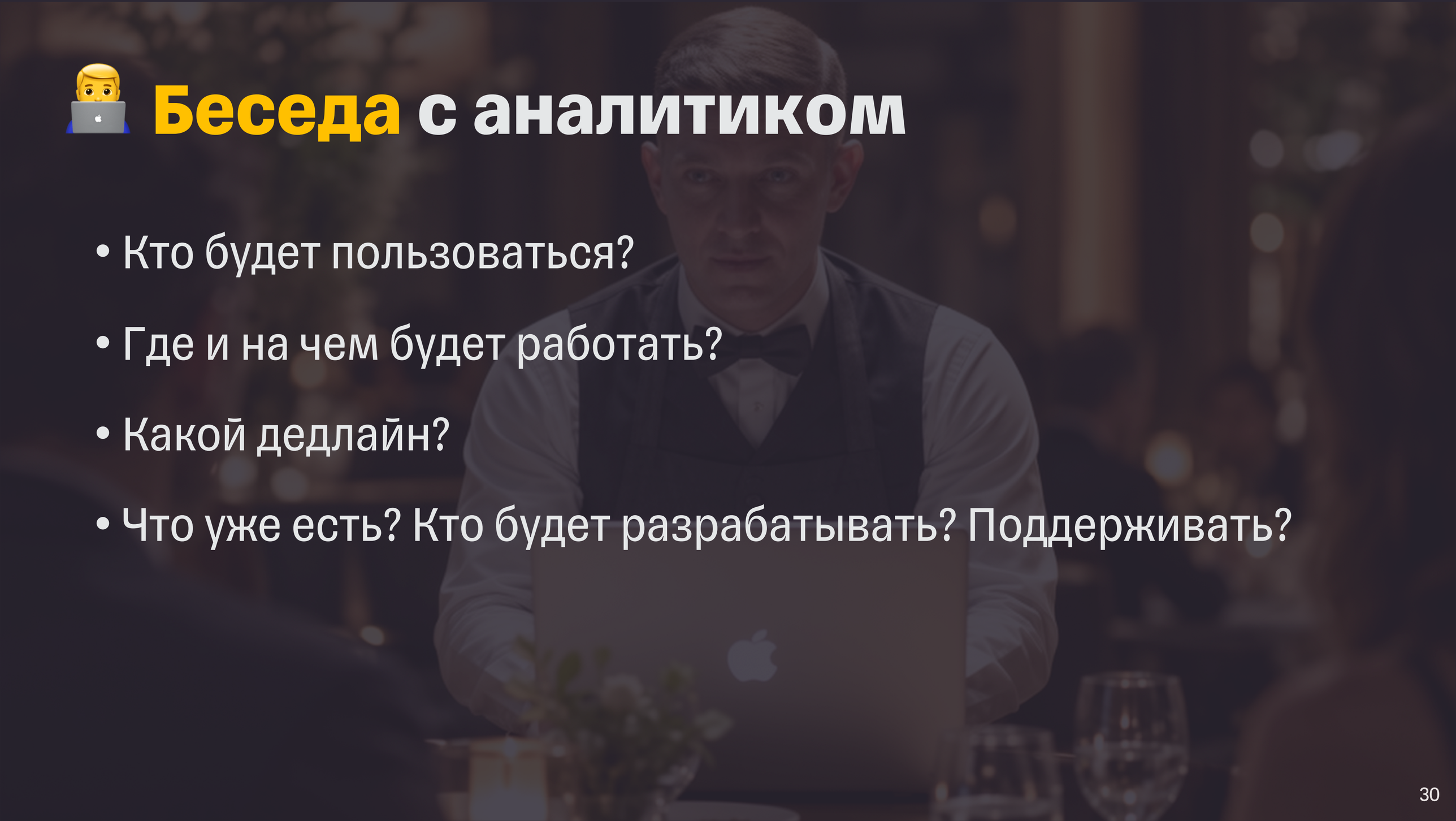


- 15 минут до поезда
- нужна еда с собой
- небольшой бюджет
- голодны
- любите острое

👩‍🍳 Тот же человек. Другой вечер

- Свидание
- 2 часа времени
- У партнера аллергия на укроп
- Хочется спокойно поговорить


Выбора нет



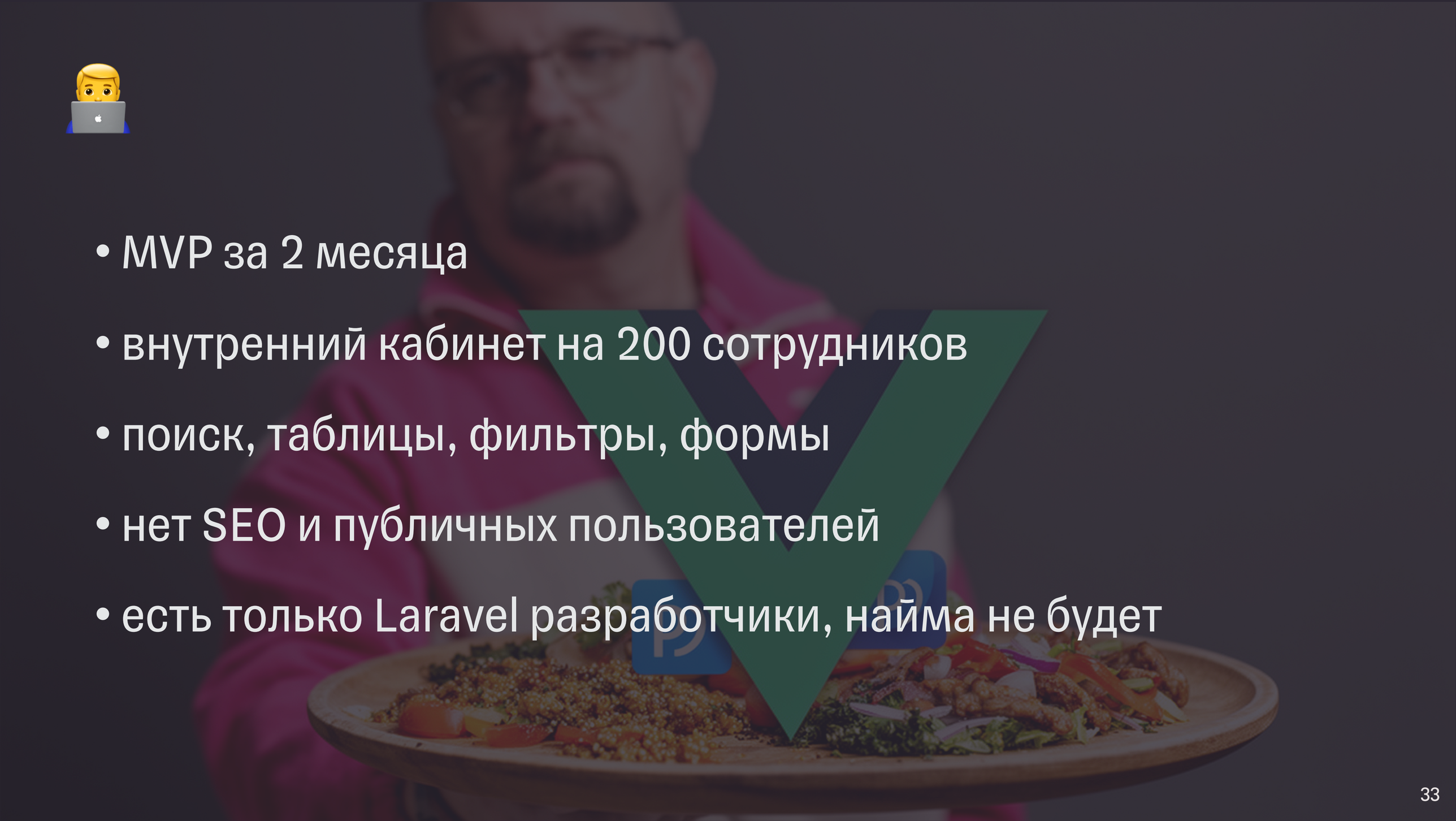
👨‍💻 Беседа с аналитиком

- Кто будет пользоваться?
- Где и на чем будет работать?
- Какой дедлайн?
- Что уже есть? Кто будет разрабатывать? Поддерживать?

👨‍💻 Беседа с аналитиком

- Это публичный сайт, внутренний кабинет или рабочее место оператора?
- У нас тысяча пользователей в месяц или десять тысяч одновременных?
- Люди работают с десктопа в офисе или с телефона в дороге?
- Нужны ли слабые сети, офлайн, локализация, WCAG-доступность?
- Есть ли существующий backend, дизайн-система, команда на Angular или React?
- Нужно ли выпустить MVP через шесть недель?
- Главная проблема — скорость первой загрузки, скорость работы оператора или скорость поставки новых фич?





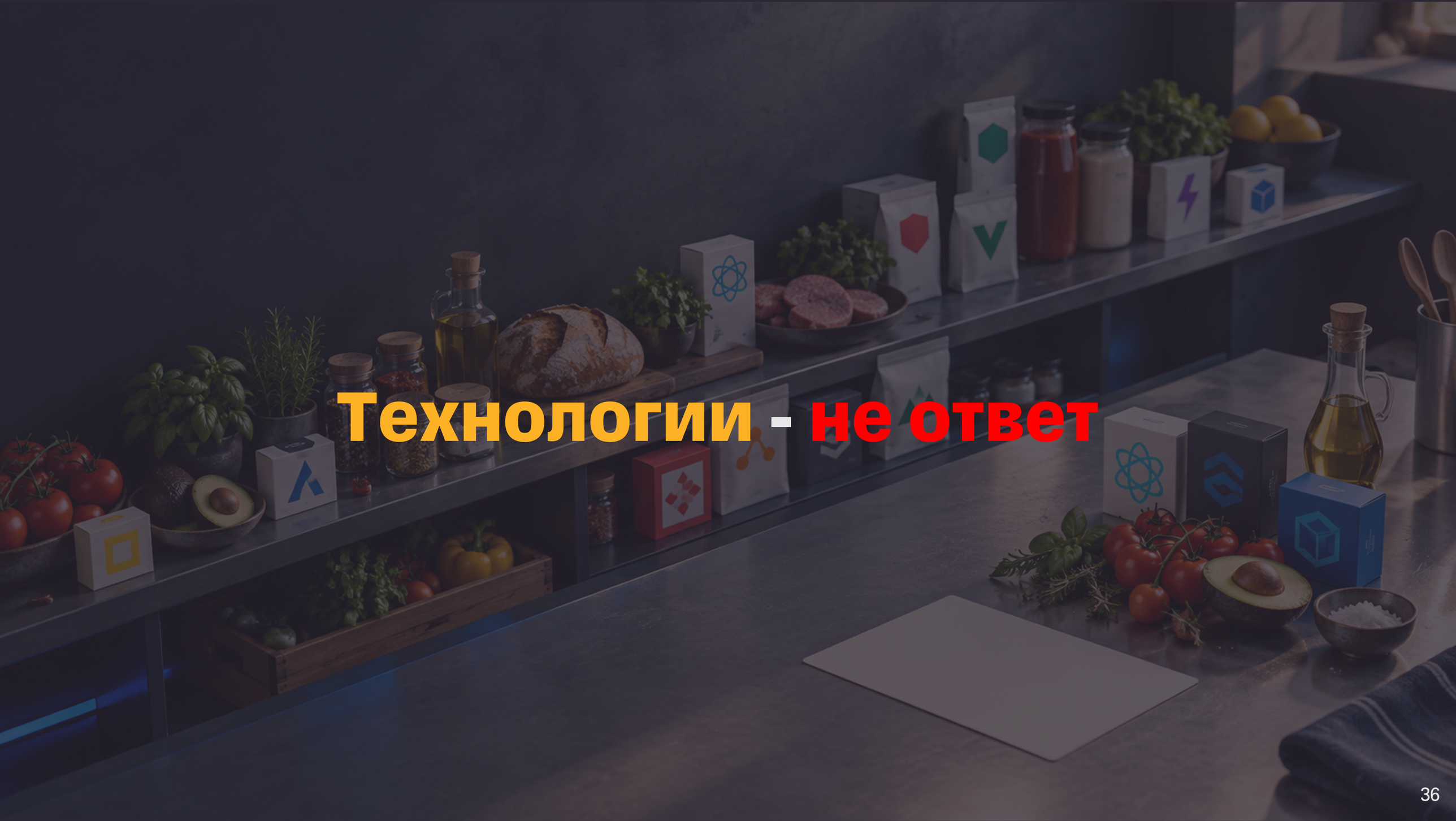
- MVP за 2 месяца
- внутренний кабинет на 200 сотрудников
- поиск, таблицы, фильтры, формы
- нет SEO и публичных пользователей
- есть только Laravel разработчики, найма не будет

👨‍💻 Тот же клиент. Другой продукт

- публичный продукт — витрина, личный кабинет
- SEO — канал привлечения
- миллионы посетителей
- маркетинг меняет страницы каждый день
- нужен быстрый первый рендер по всему миру

Хорошее решение ≠ популярное решение

Хорошее решение = вариант, подходящий контексту

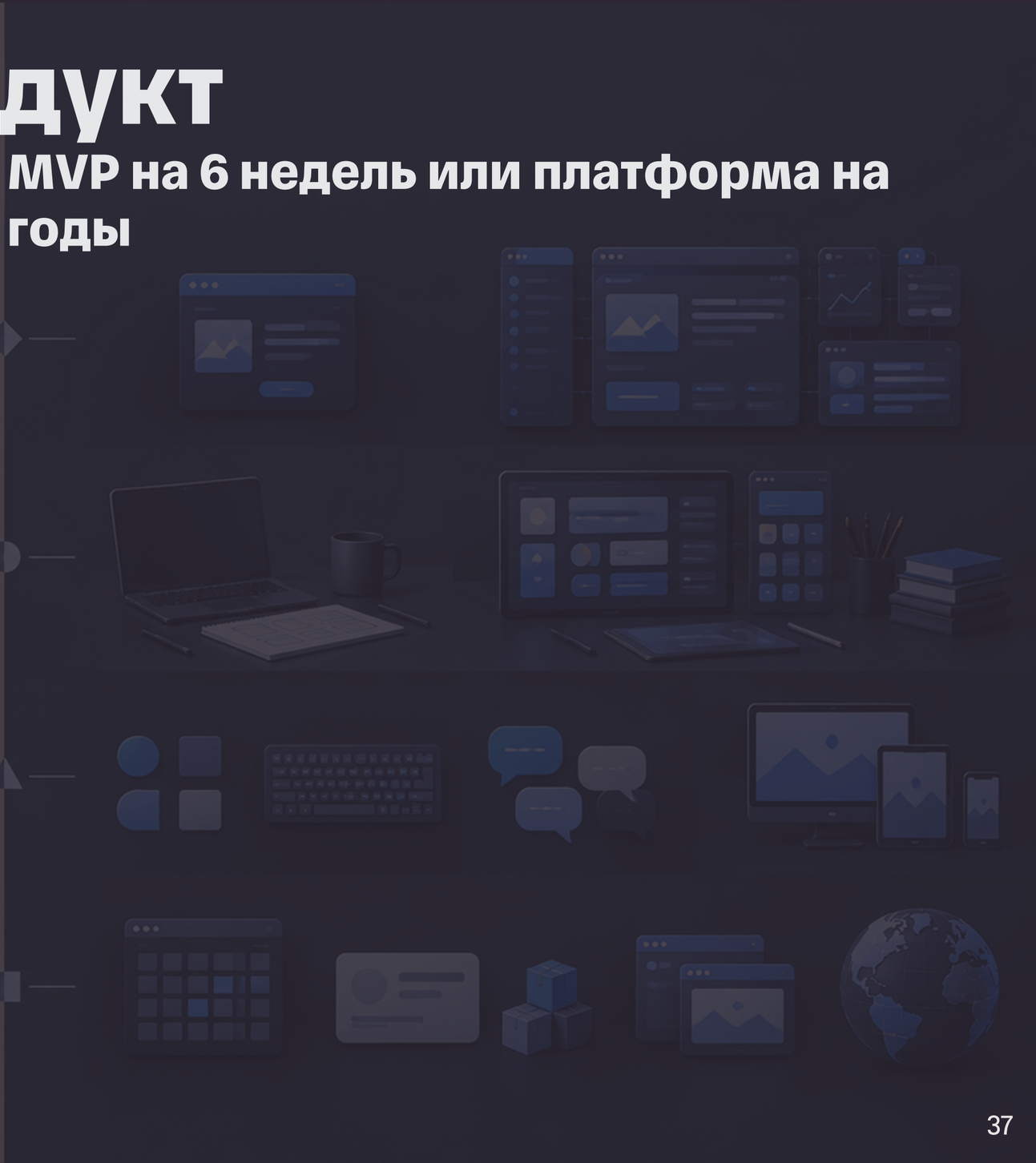

Технологии - не ответ

Продукт

Перекус на бегу или важный ужин



MVP на 6 недель или платформа на годы



Продукт

Перекус на бегу или важный ужин



MVP на 6 недель или платформа на годы

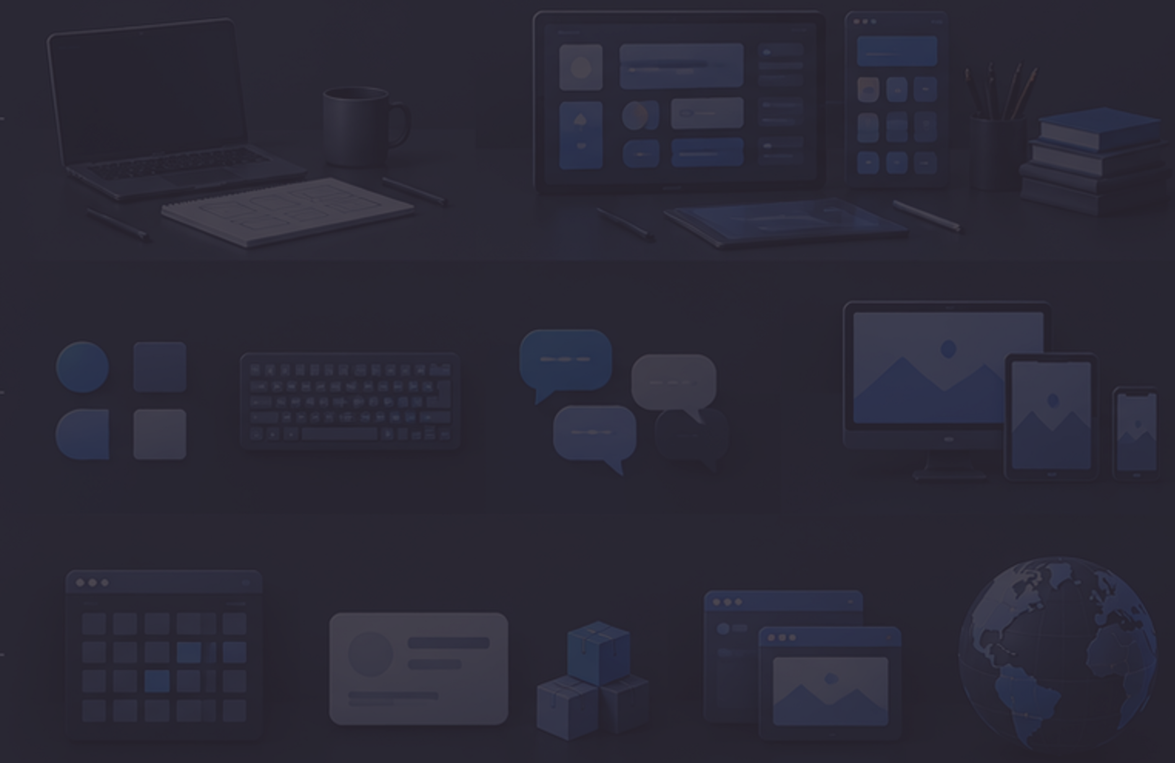


Команда

Шеф или повар на замене

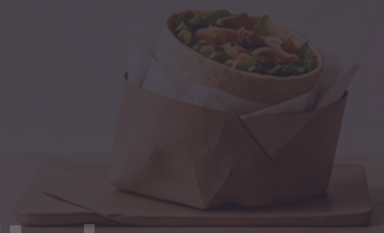


Кто владеет UI



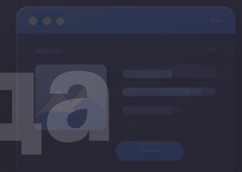
Продукт

Перекус на бегу или важный ужин





MVP на 6 недель или платформа на годы





Команда

Шеф или повар на замене





Кто владеет UI





Качество

Учтены аллергены и темп подачи





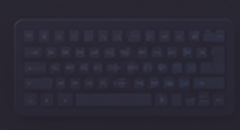






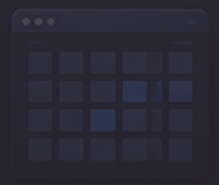
a11y, i18n, адаптив, сети

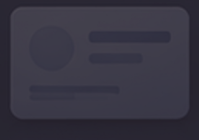


















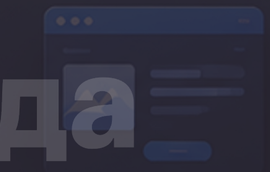
Продукт

Перекус на бегу или важный ужин





MVP на 6 недель или платформа на годы





Команда

Шеф или повар на замене





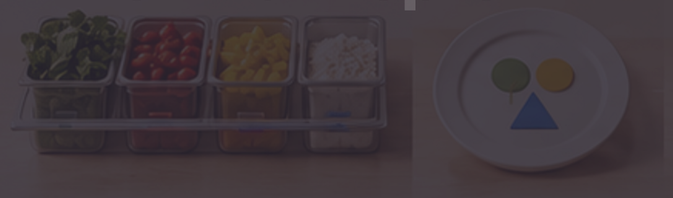
Кто владеет UI





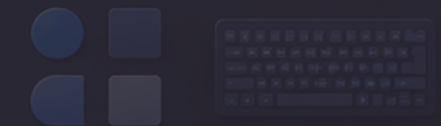
Качество

Учтены аллергены и темп подачи





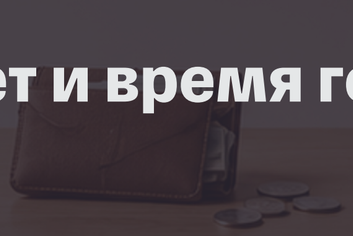
a11y, i18n, адаптив, сети





Ограничения

Бюджет и время гостя





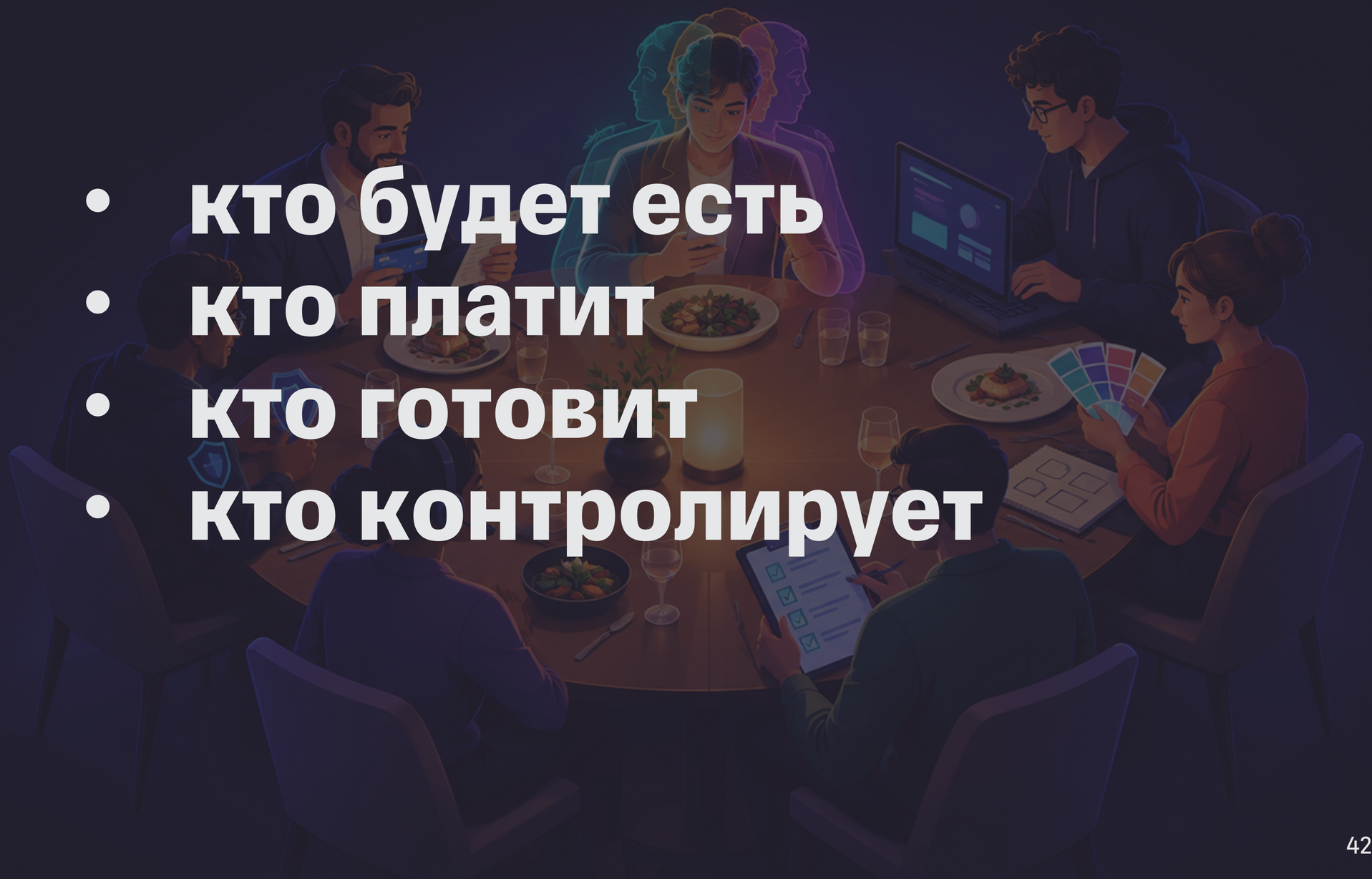
Сроки, лицензии, браузеры, регионы



Кто влияет на заказ

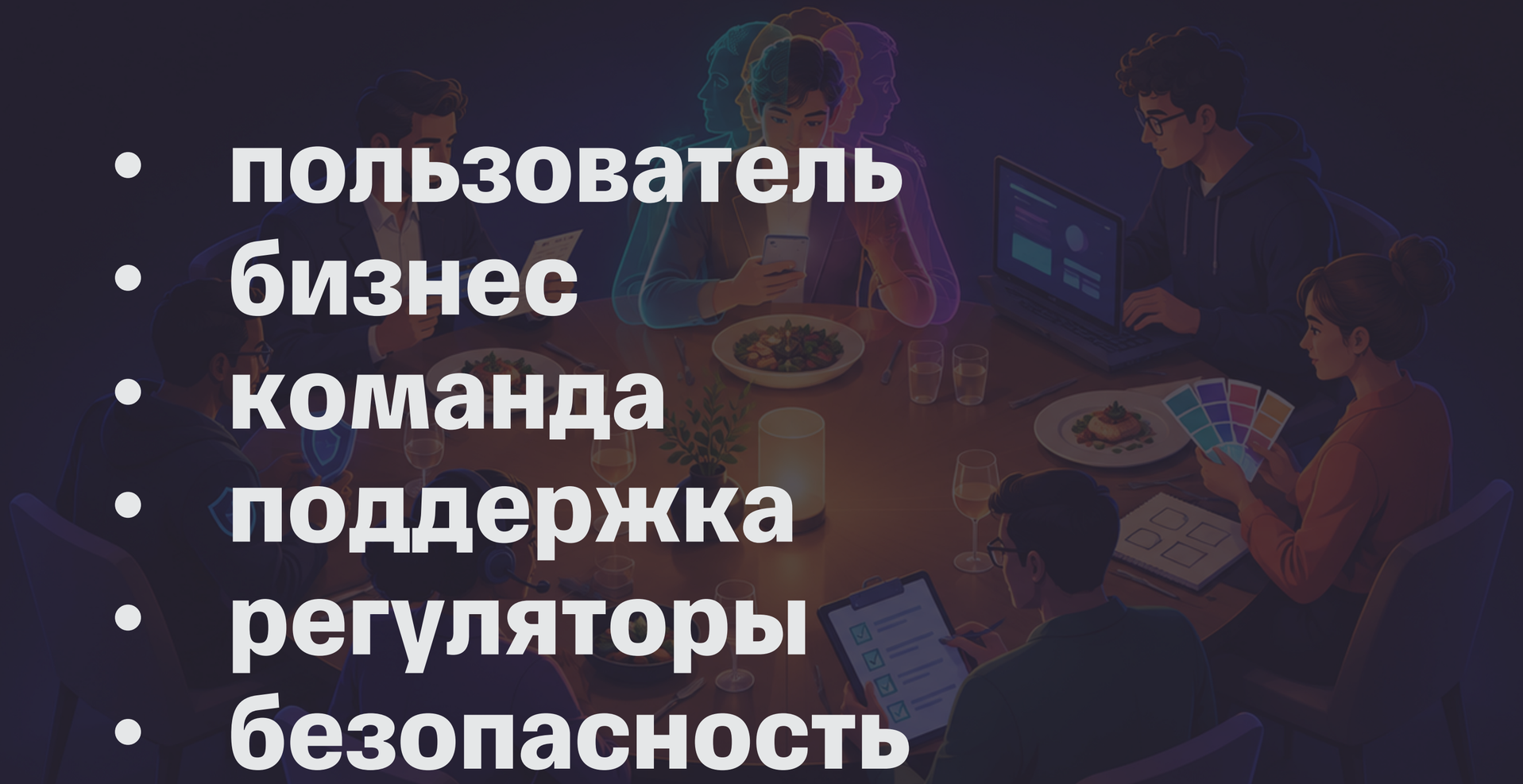


👨‍🍳 Кто **влияет** на заказ



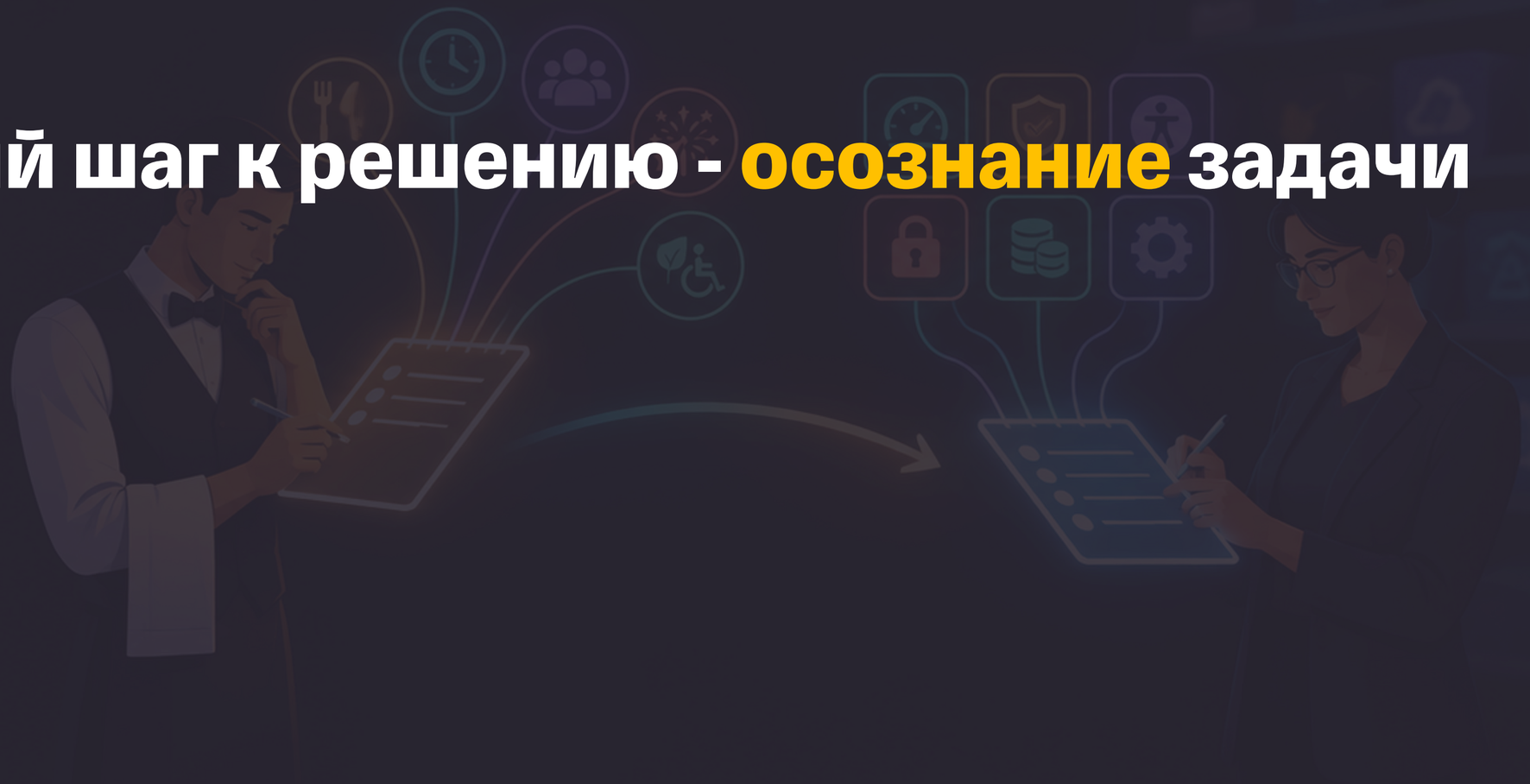
- кто будет есть
- кто платит
- кто готовит
- кто контролирует

👨‍💻 Кто **влияет** на заказ



- пользователь
- бизнес
- команда
- поддержка
- регуляторы
- безопасность

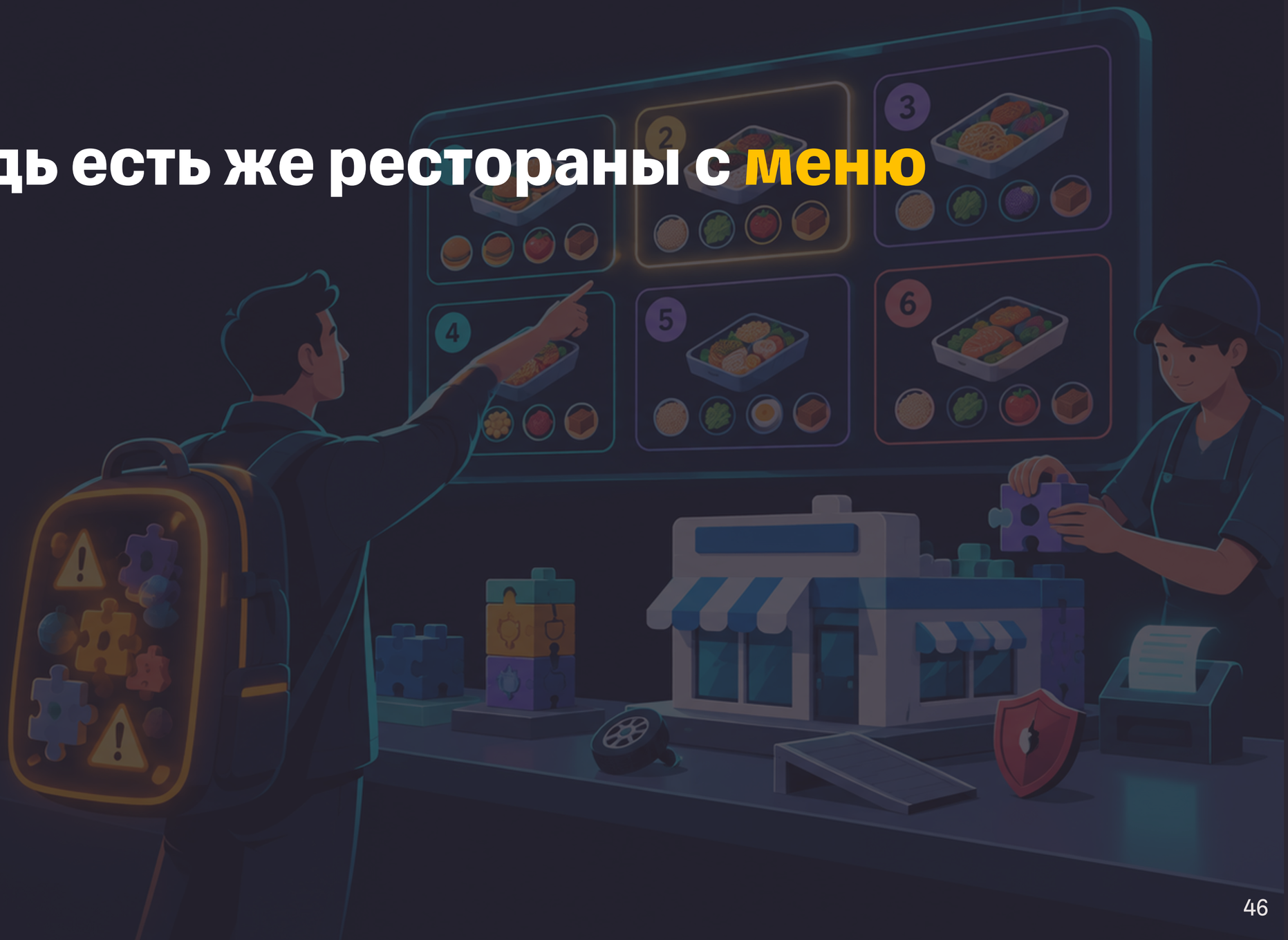
Первый шаг к решению - осознание задачи



Почему в меню нет ответа



Но ведь есть же рестораны с меню





Первый закон архитектуры

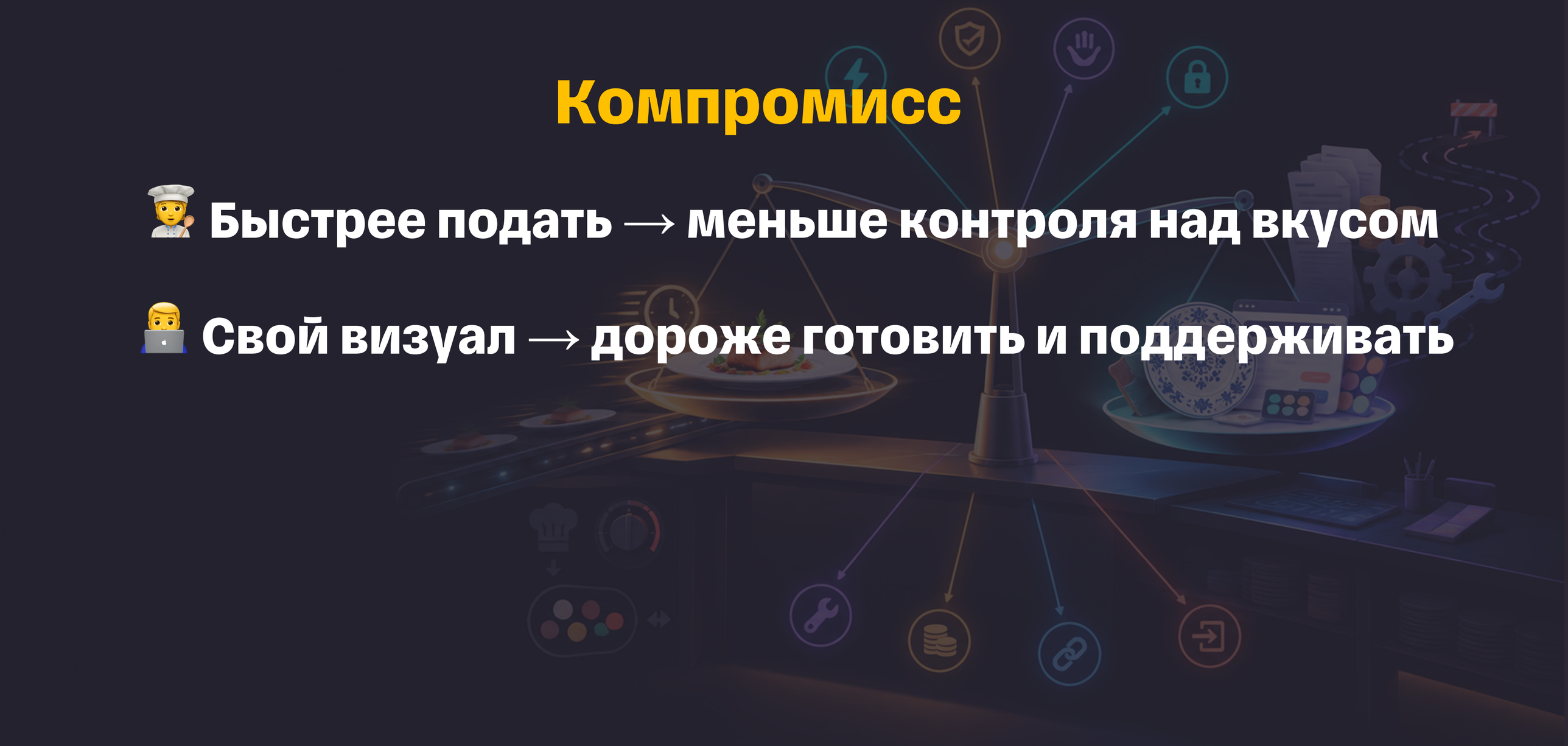
Всё в архитектуре ПО - компромисс

Если trade-off не виден — вы его ещё не нашли

Компромисс



Компромисс



👨‍🍳 Быстрее подать → меньше контроля над вкусом

👨‍💻 Свой визуал → дороже готовить и поддерживать

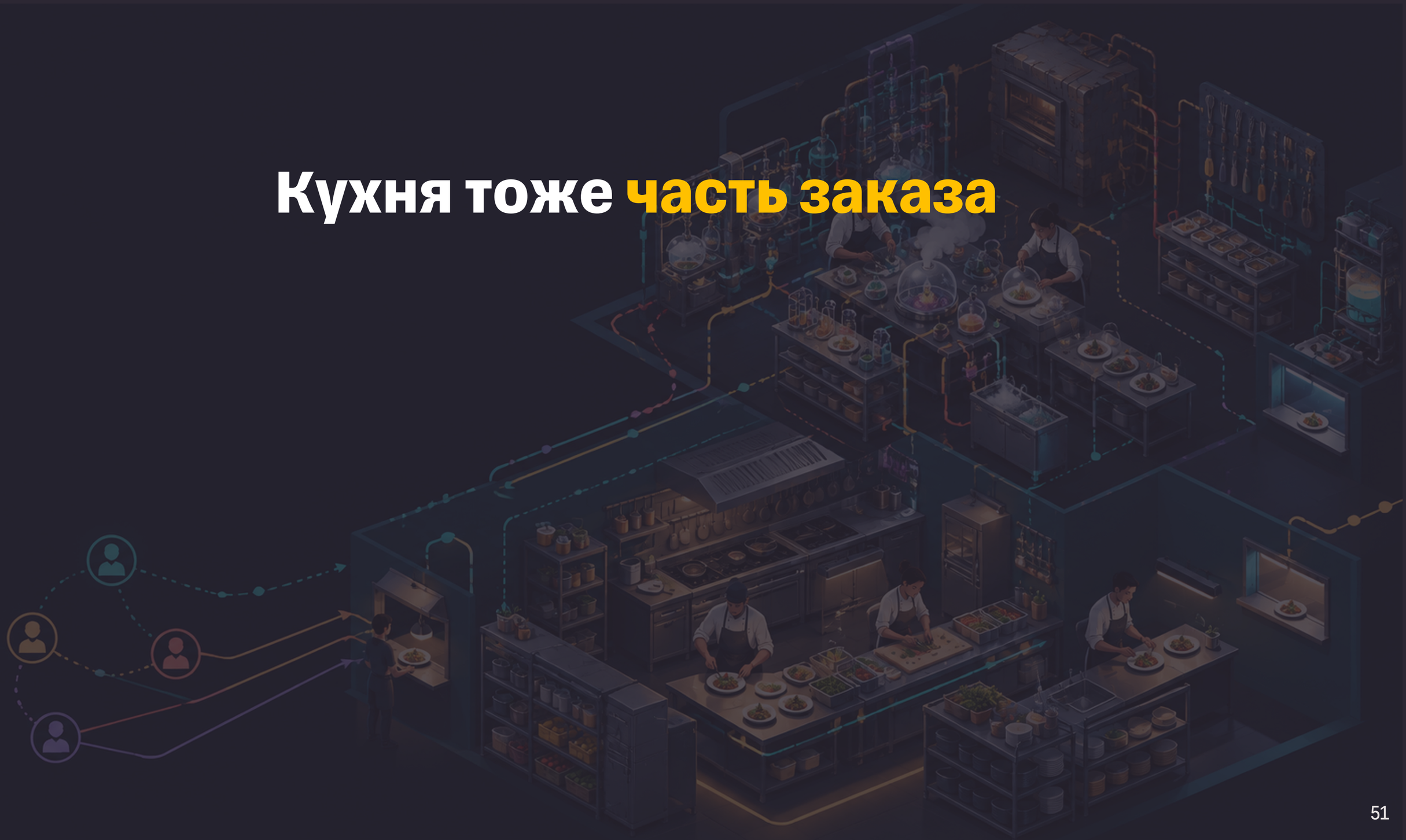
Легко не просто



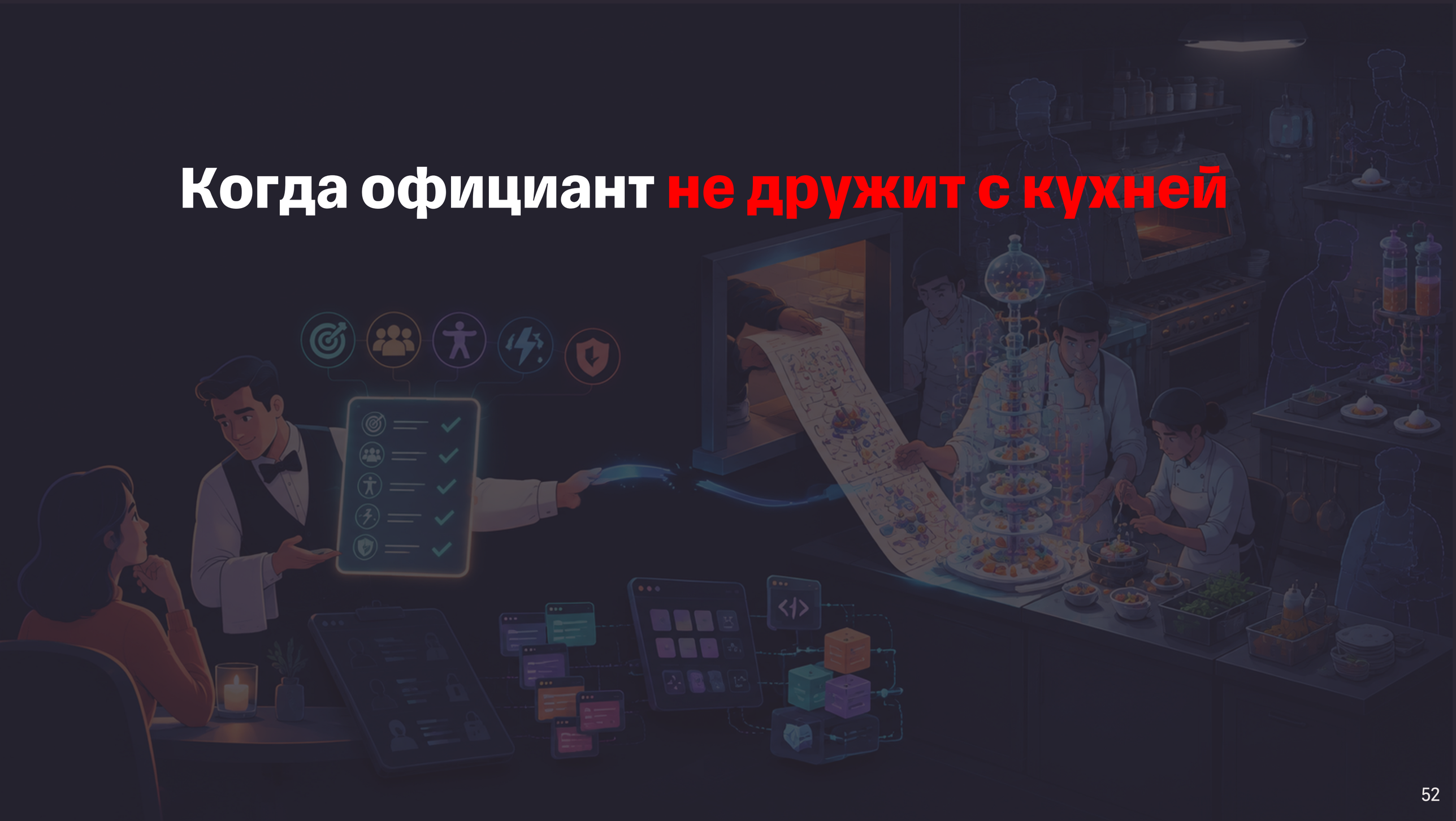
Easy for whom?

Rich Hickey

Кухня тоже часть заказа

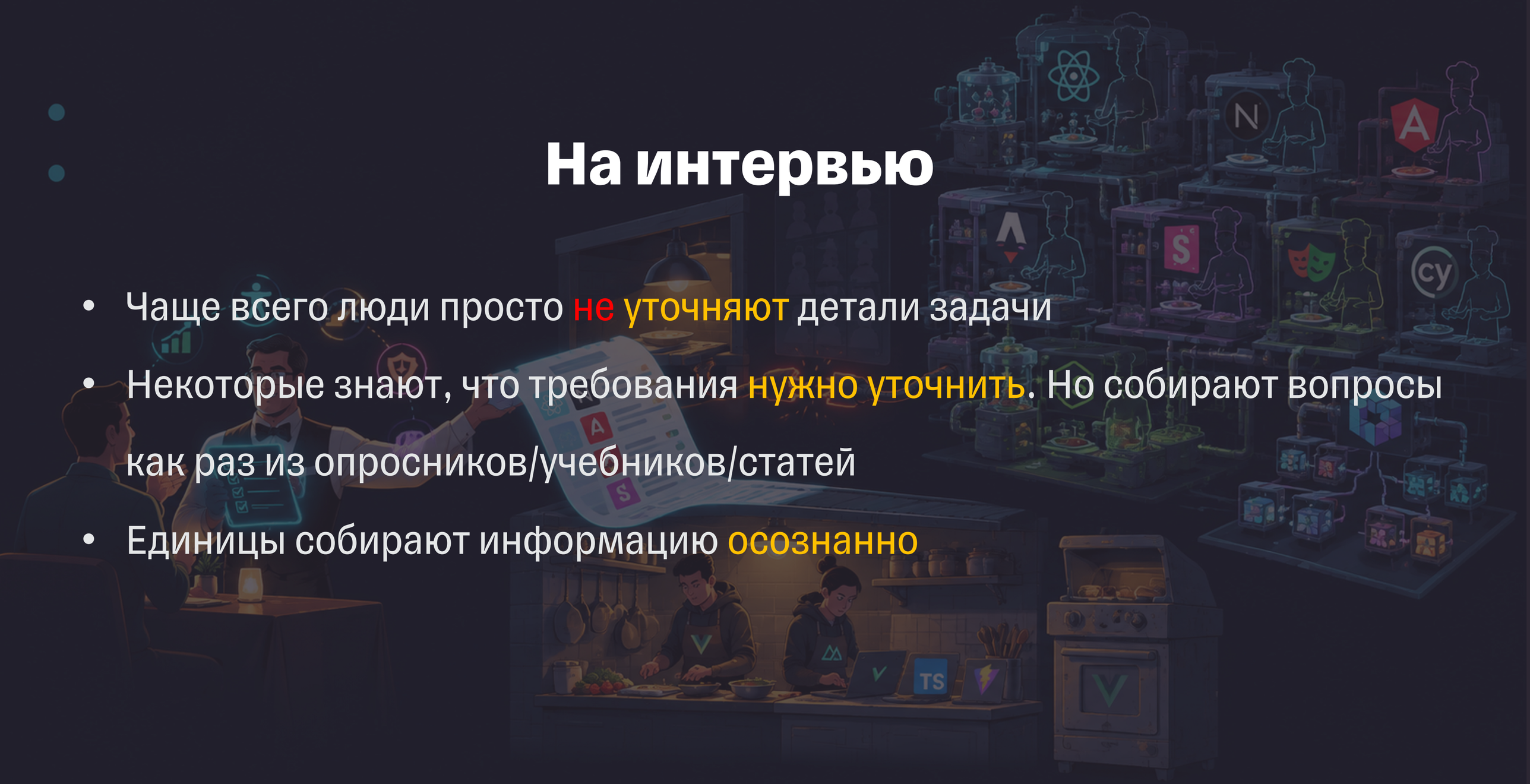


Когда официант не дружит с кухней



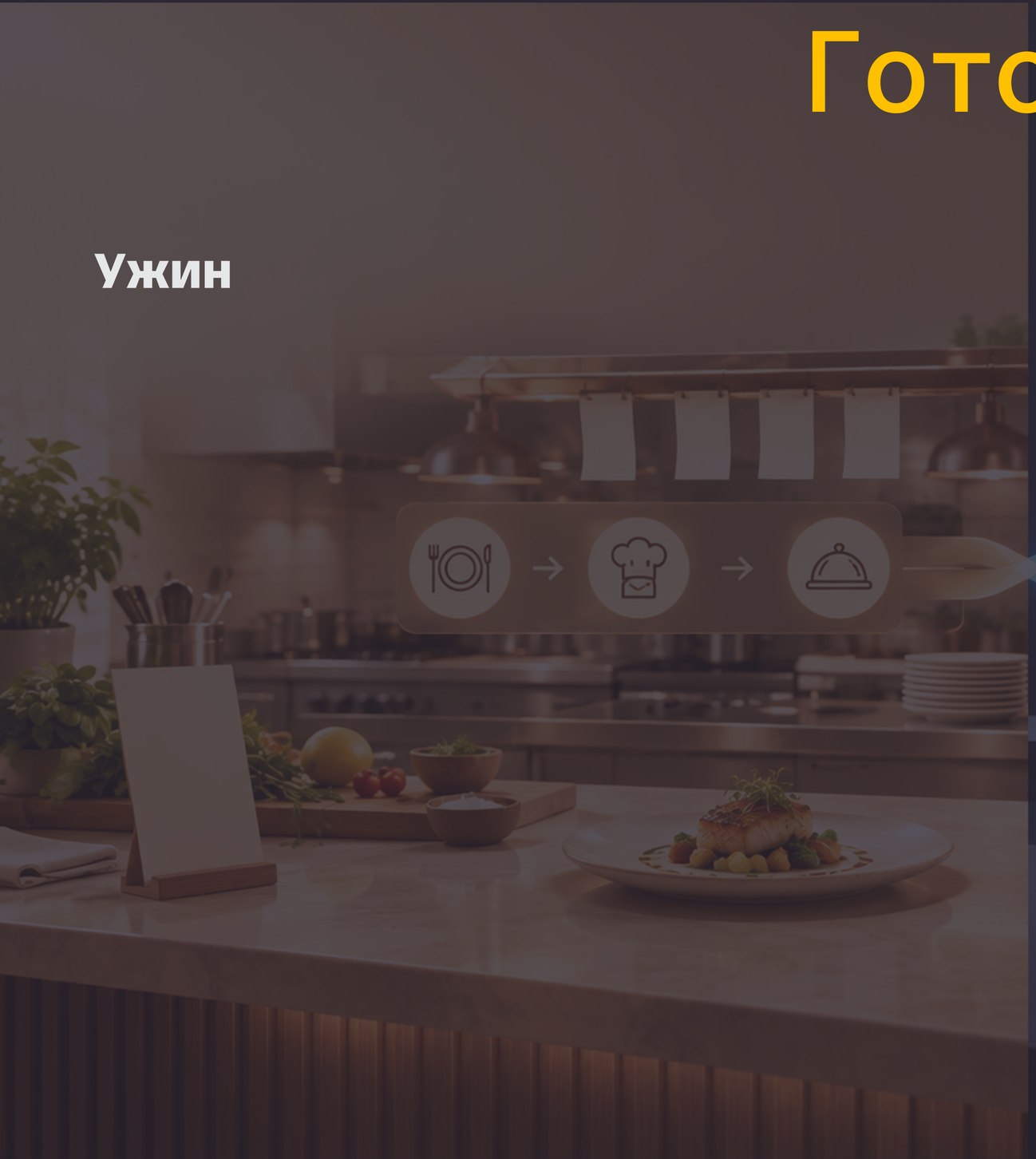
На интервью

- Чаще всего люди просто не уточняют детали задачи
- Некоторые знают, что требования нужно уточнить. Но собирают вопросы как раз из опросников/учебников/статей
- Единицы собирают информацию осознанно

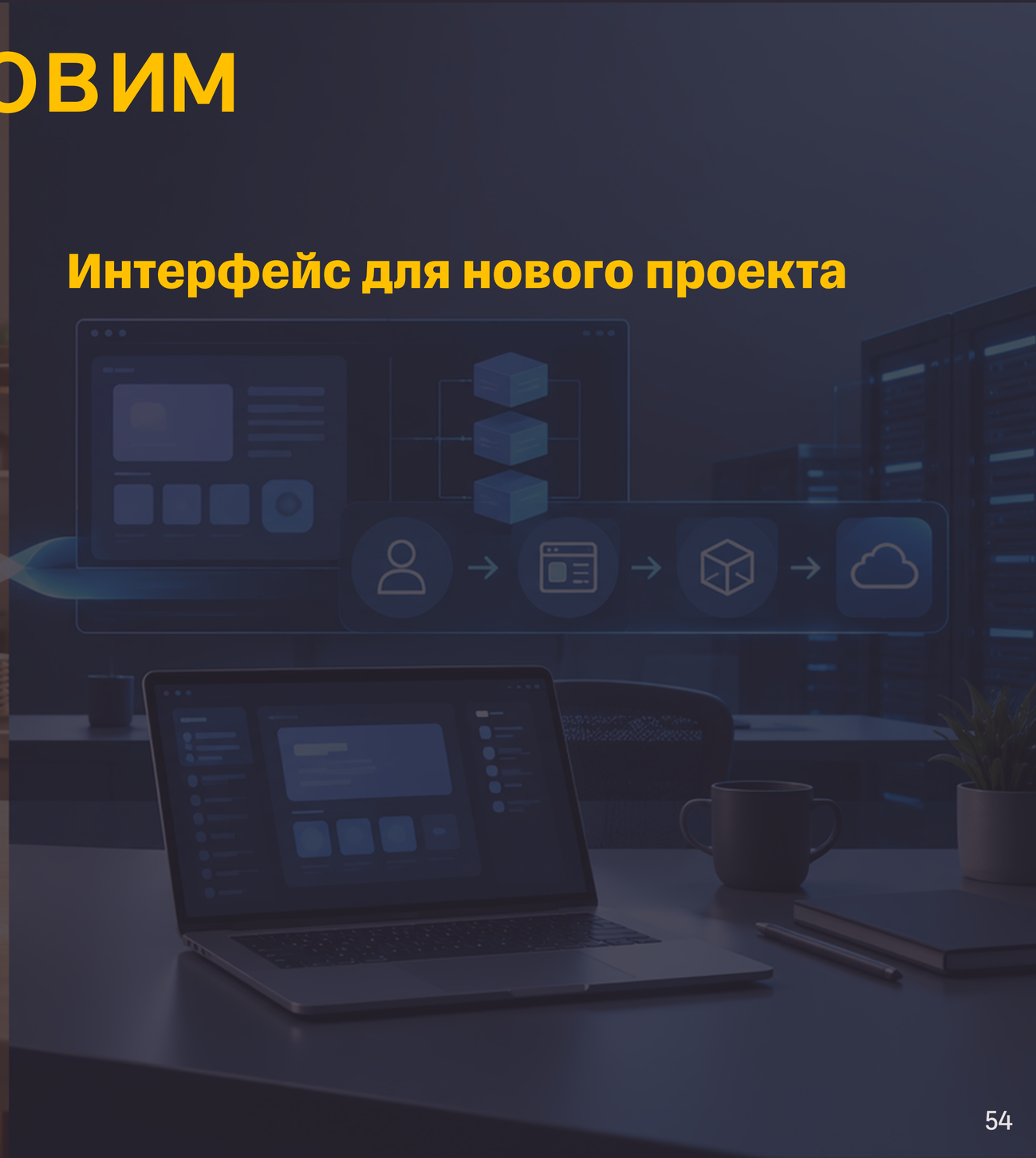


Готовим

Ужин



Интерфейс для нового проекта

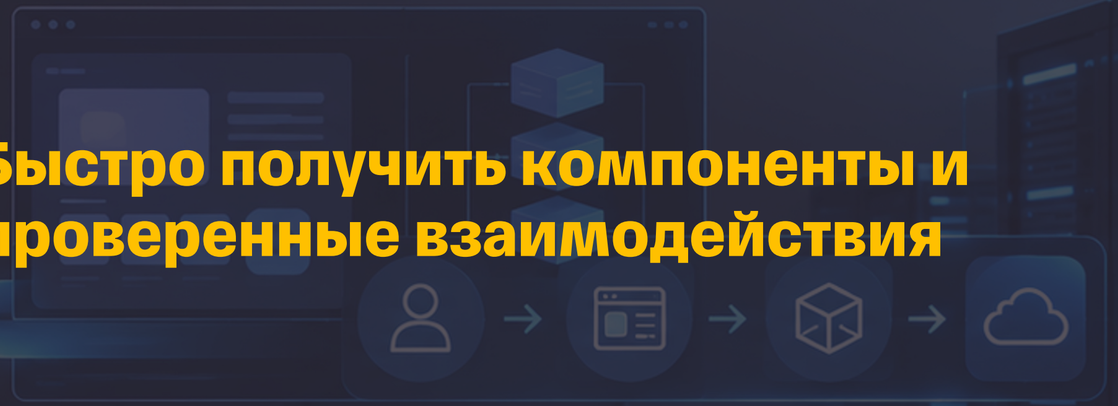


Готовое решение

Уже готовая еда, только подать к столу



Быстро получить компоненты и проверенные взаимодействия

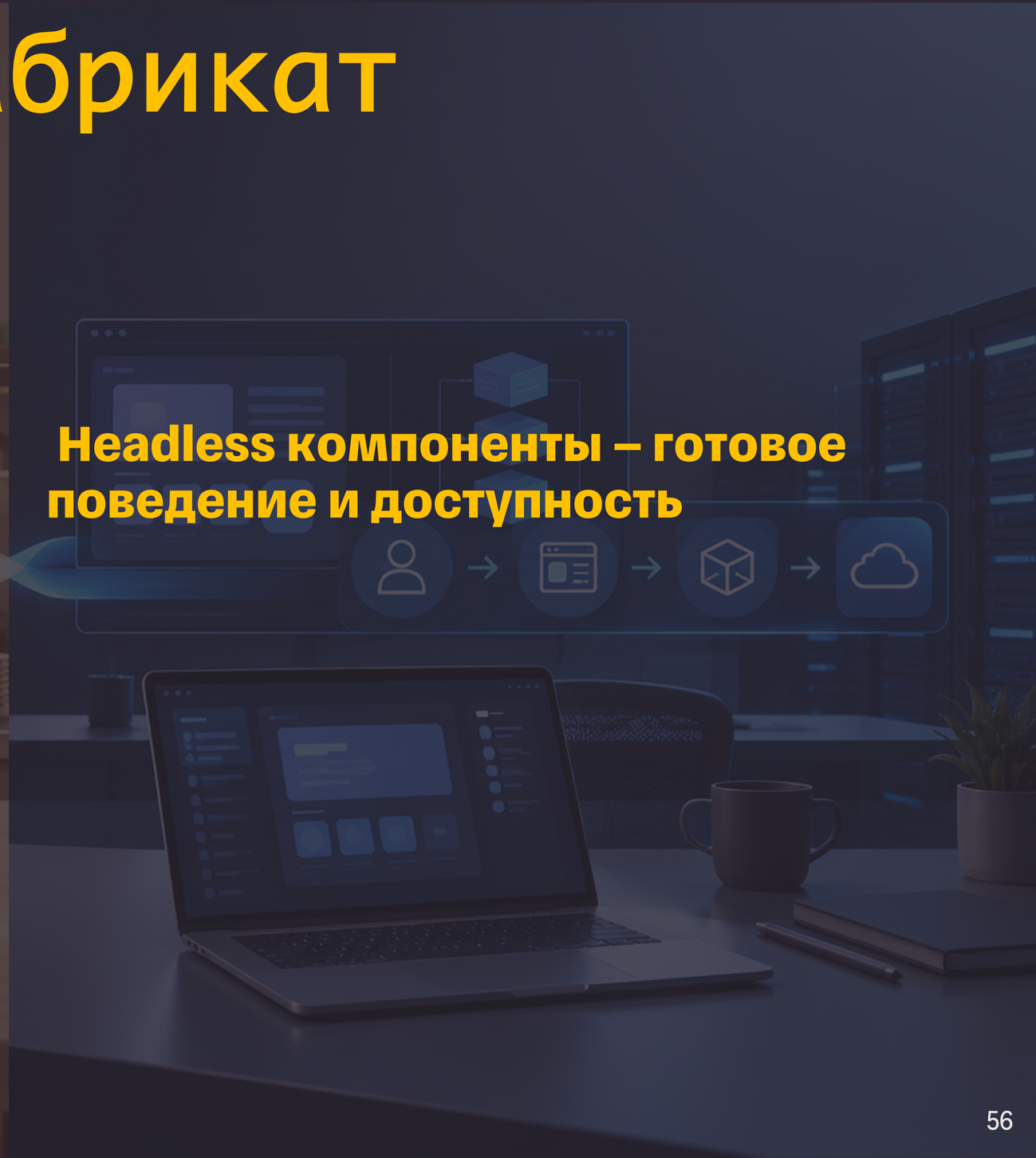


Полуфабрикат

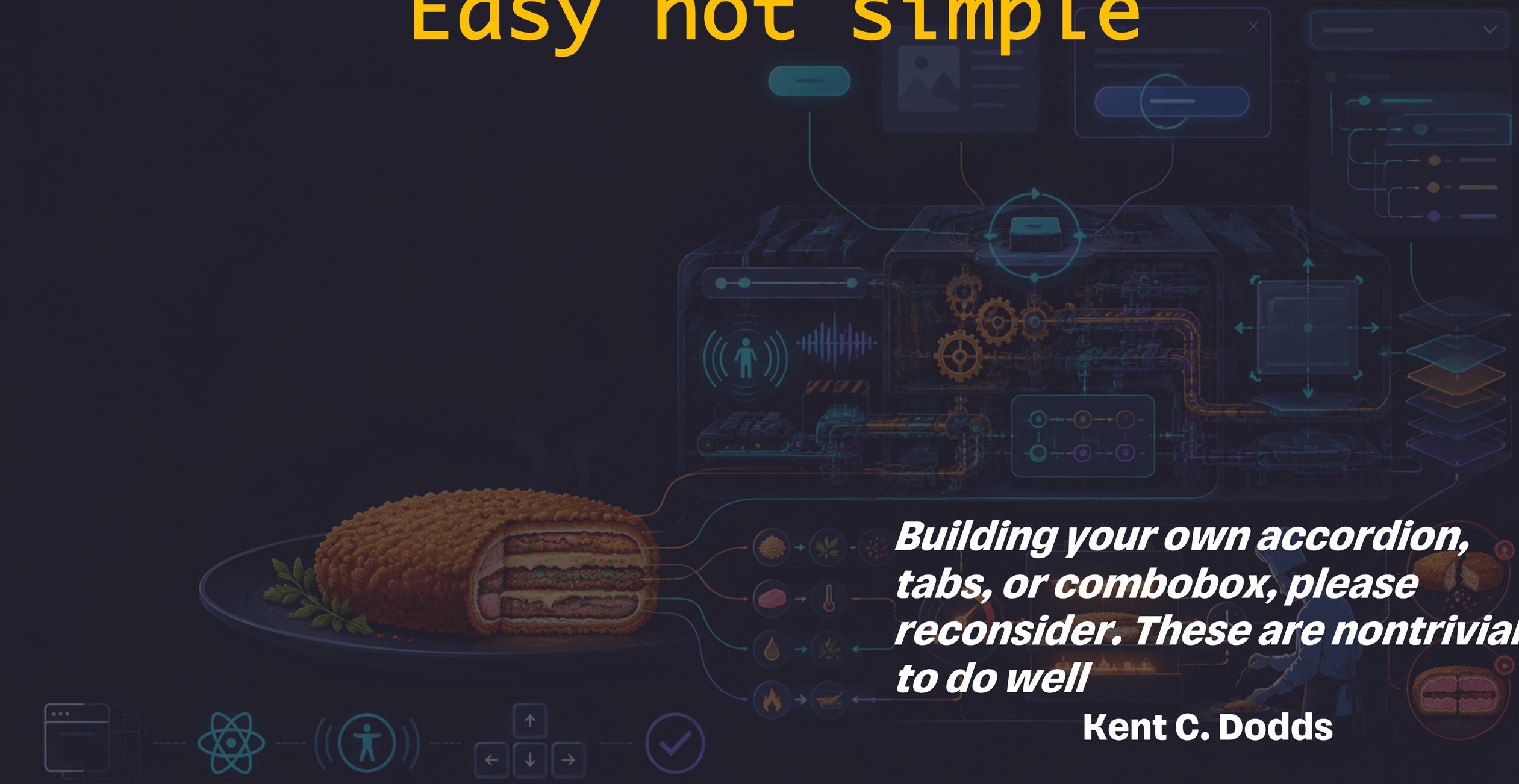
Заготовки и полуфабрикаты – монотонная работа уже сделана



Headless компоненты – готовое поведение и доступность



Easy not simple



Building your own accordion, tabs, or combobox, please reconsider. These are nontrivial to do well

Kent C. Dodds

На заказ

Приготовить блюдо с нуля специально под гостя, которого нет ни в одном меню ресторана — потому что именно это блюдо и есть причина, по которой гость сюда пришёл.

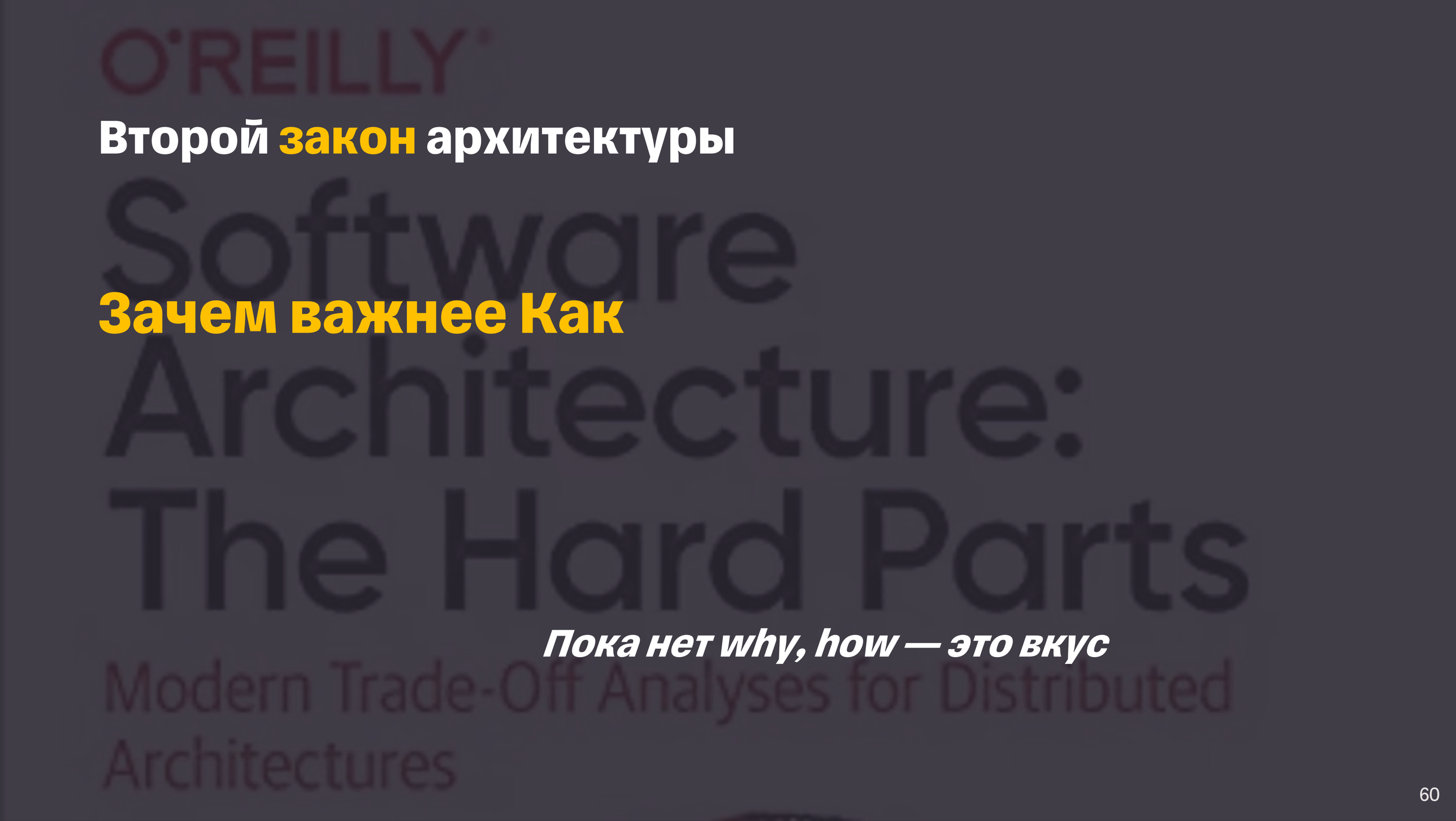
Писать компоненты самим — не потому что «так правильно», а потому что взаимодействие само и есть продукт: редактор, канвас, карта

Частая ошибка



👨‍🍳 Мы принесем вам пасту, это сегодня хит

👨‍💻 Возьмем tailwind, его все знают



Второй закон архитектуры

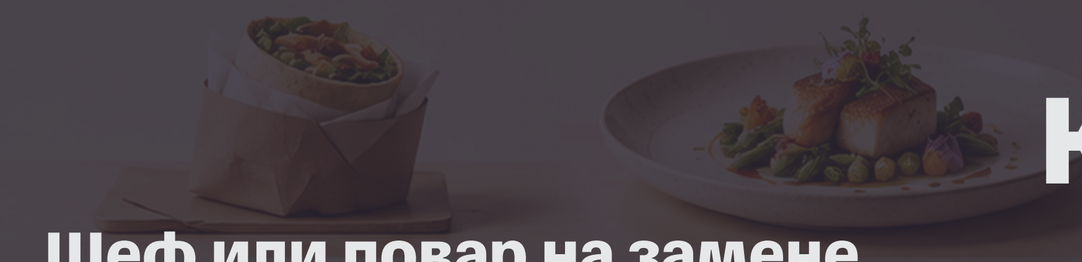
Зачем важнее Как

Пока нет why, how — это вкус

Продукт

Перекус на бегу или важный ужин

MVP на 6 недель или платформа на годы





Команда

Шеф или повар на замене

Кто владеет UI

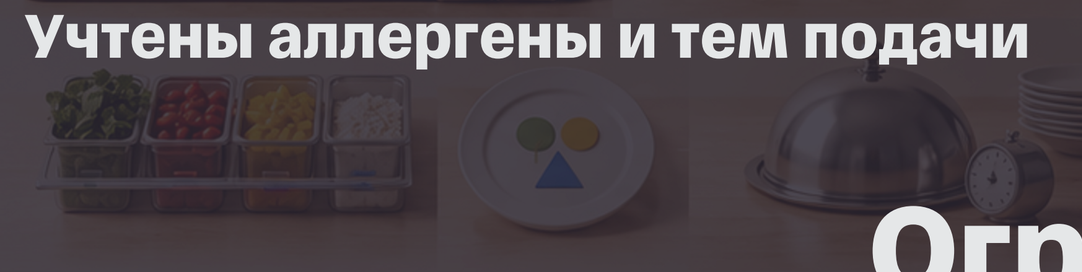


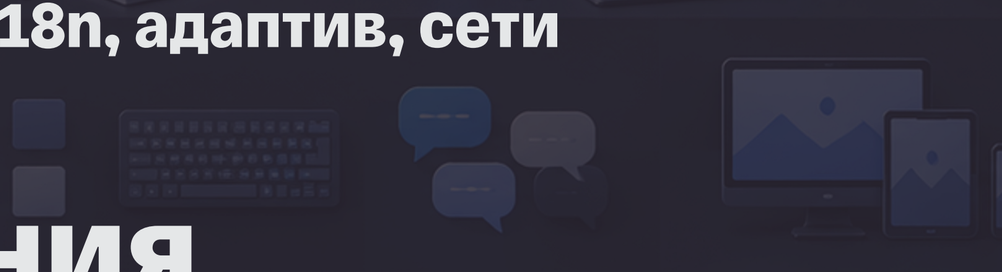


Качество

Учтены аллергены и тем подачи

a11y, i18n, адаптив, сети

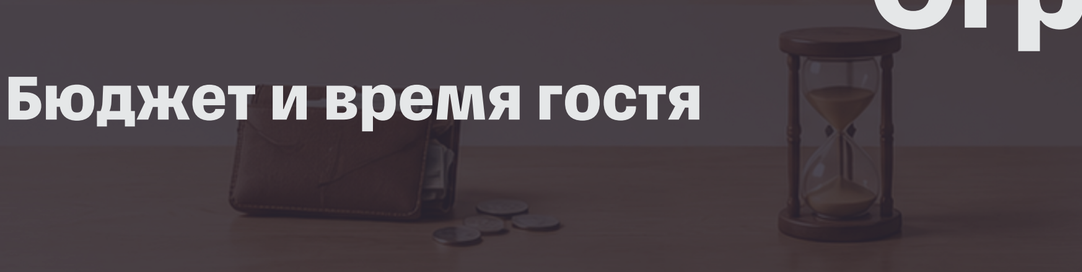


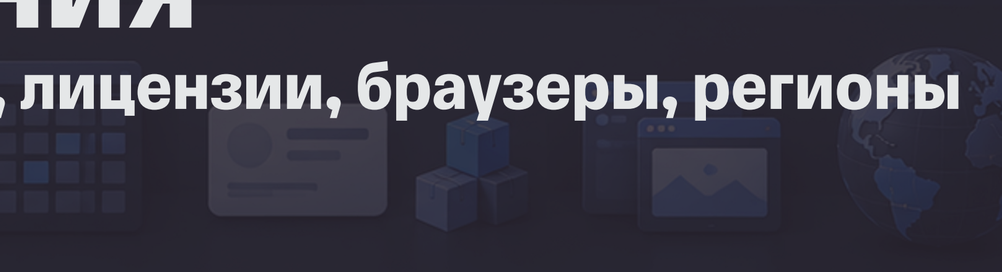


Ограничения

Бюджет и время гостя

Сроки, лицензии, браузеры, регионы





Каждому свое



Case 1



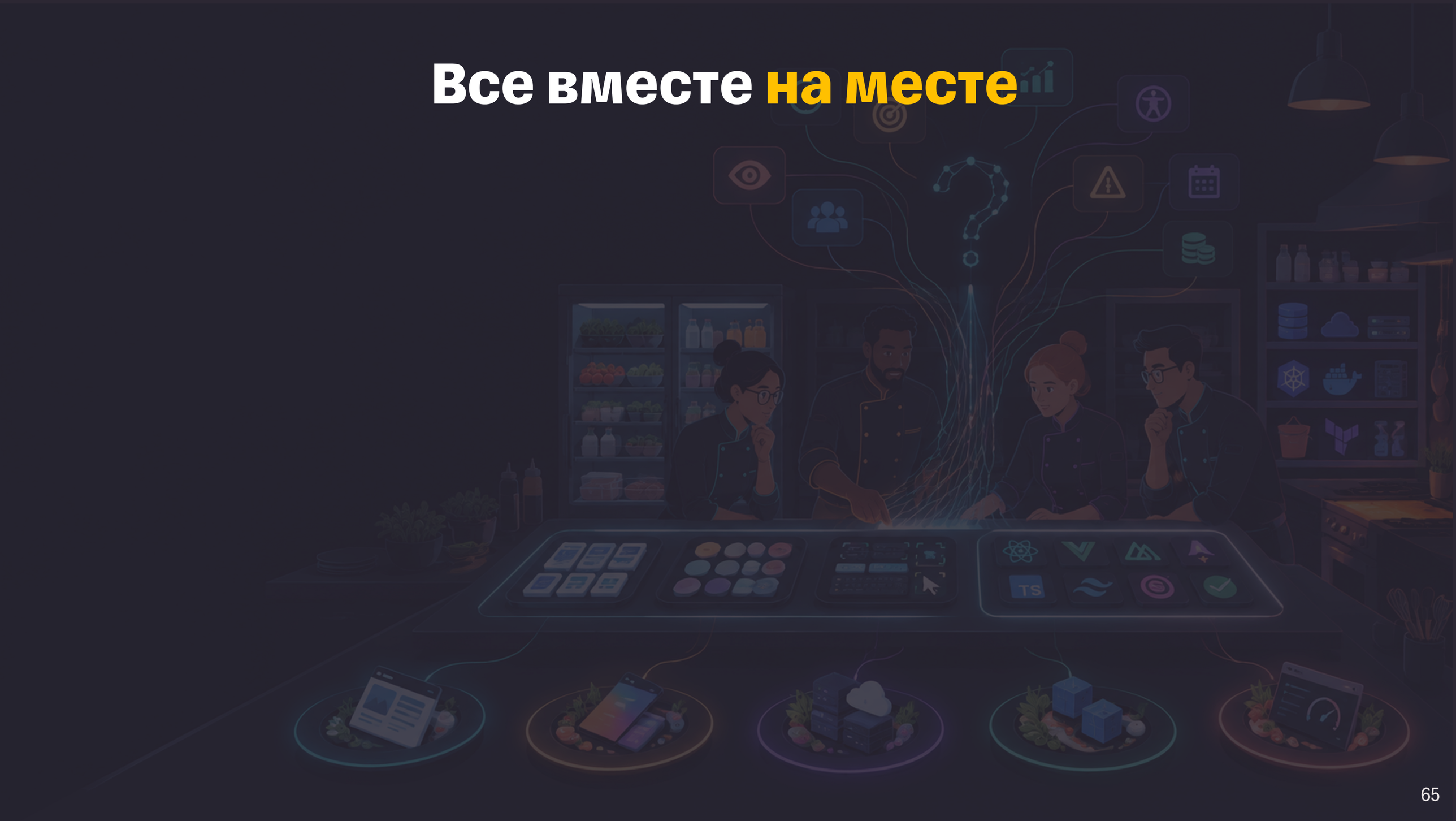
👨‍💻 Команда из двух человек. Админка за 6 недель. Дизайн не важен

Case 2

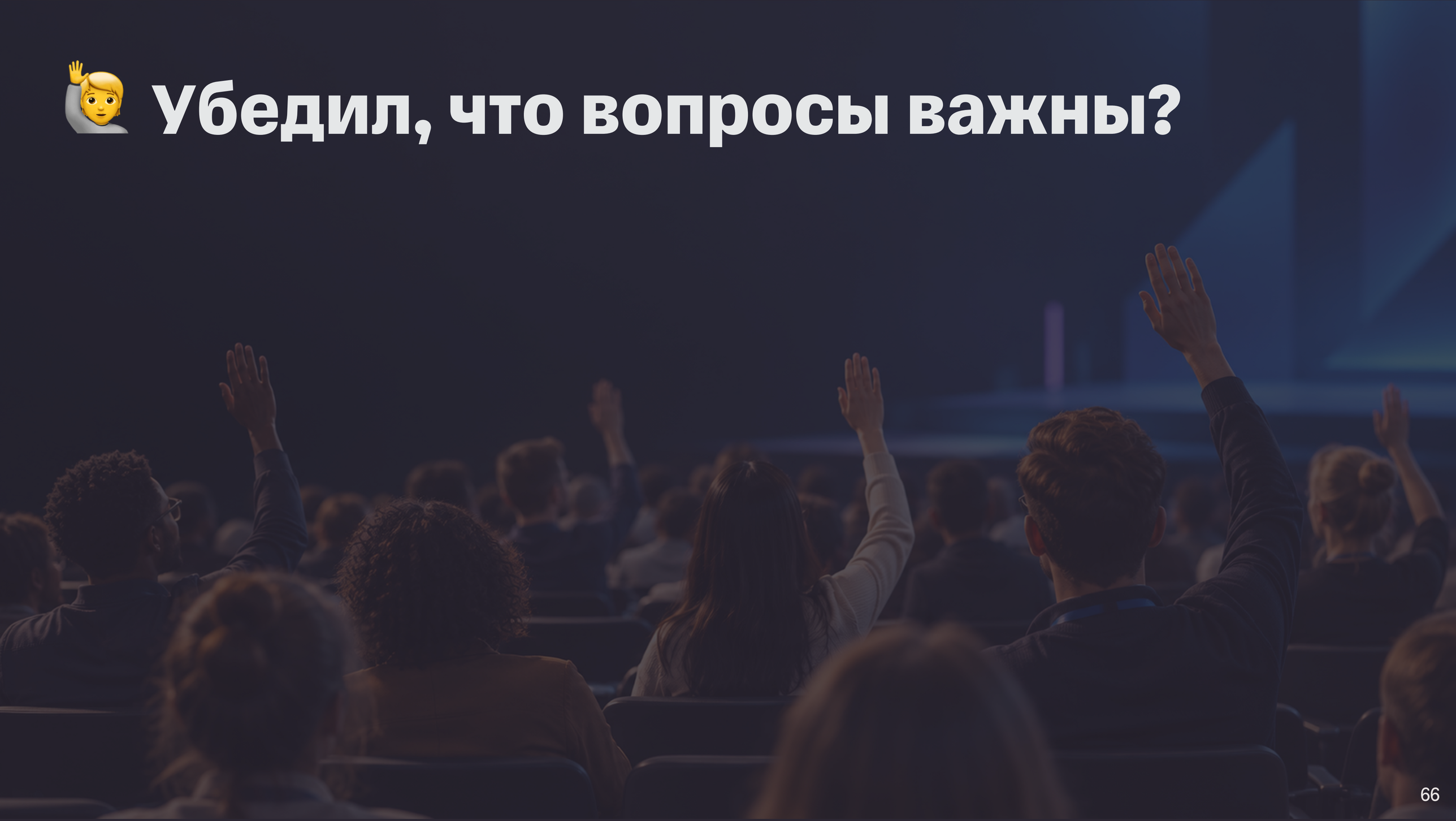
👨‍💻 Несколько продуктов. Единый и узнаваемый бренд.
Дизайнеры развивают токены



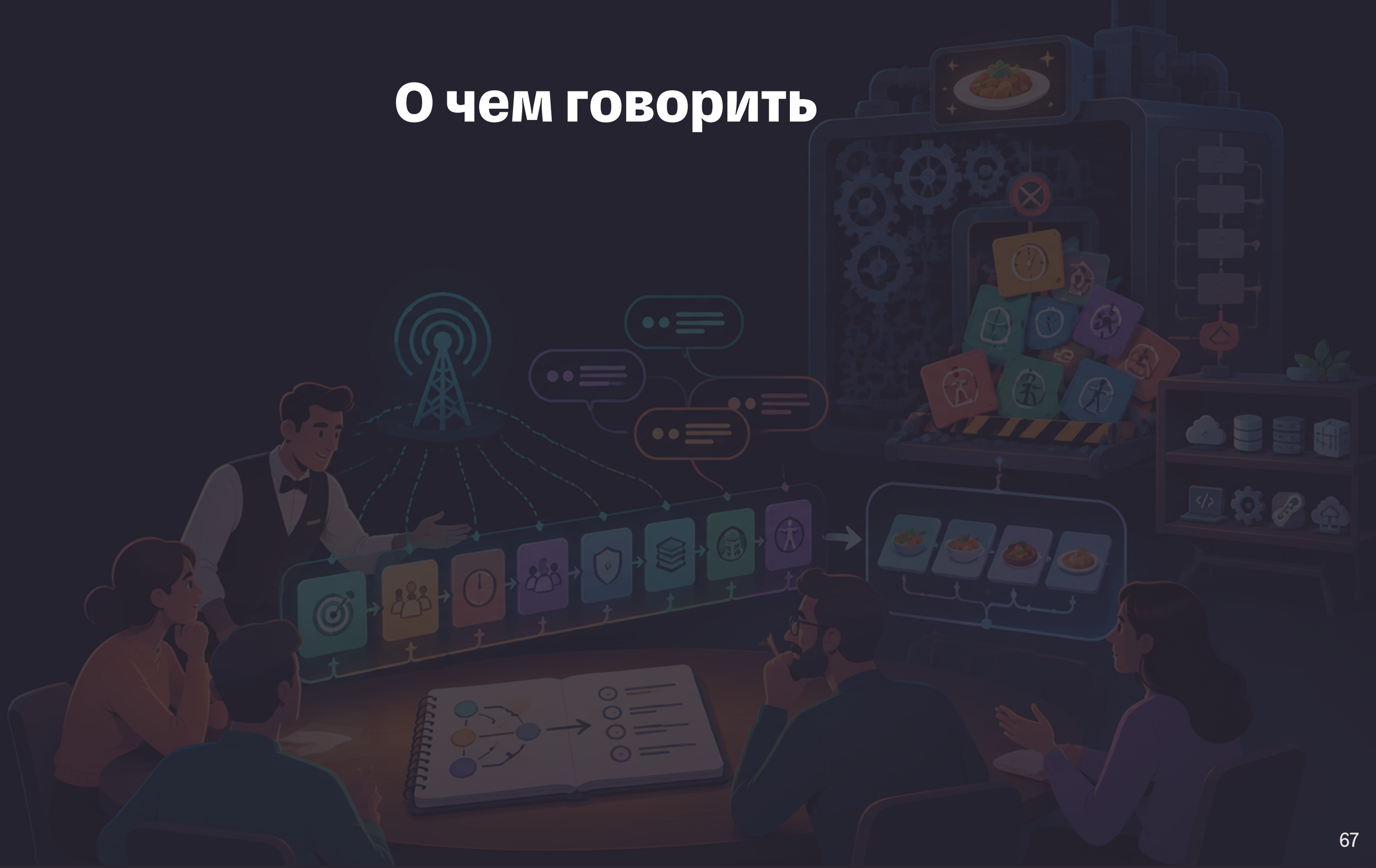
Все вместе на месте



🙋‍♂️ Убедил, что вопросы важны?



О чем говорить

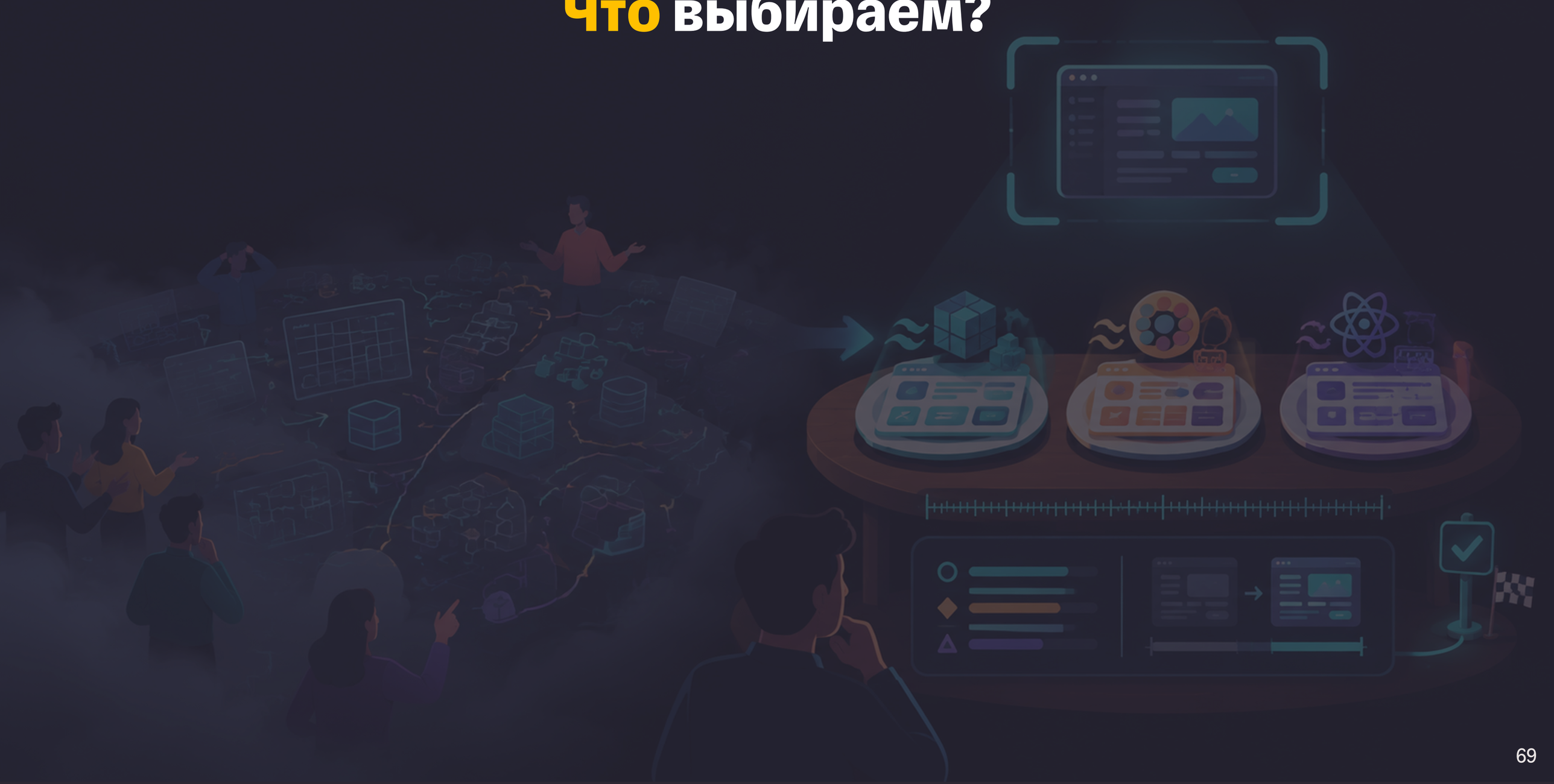


О чем говорить

Алгоритма не будет



Что выбираем?



Кто заказчик?



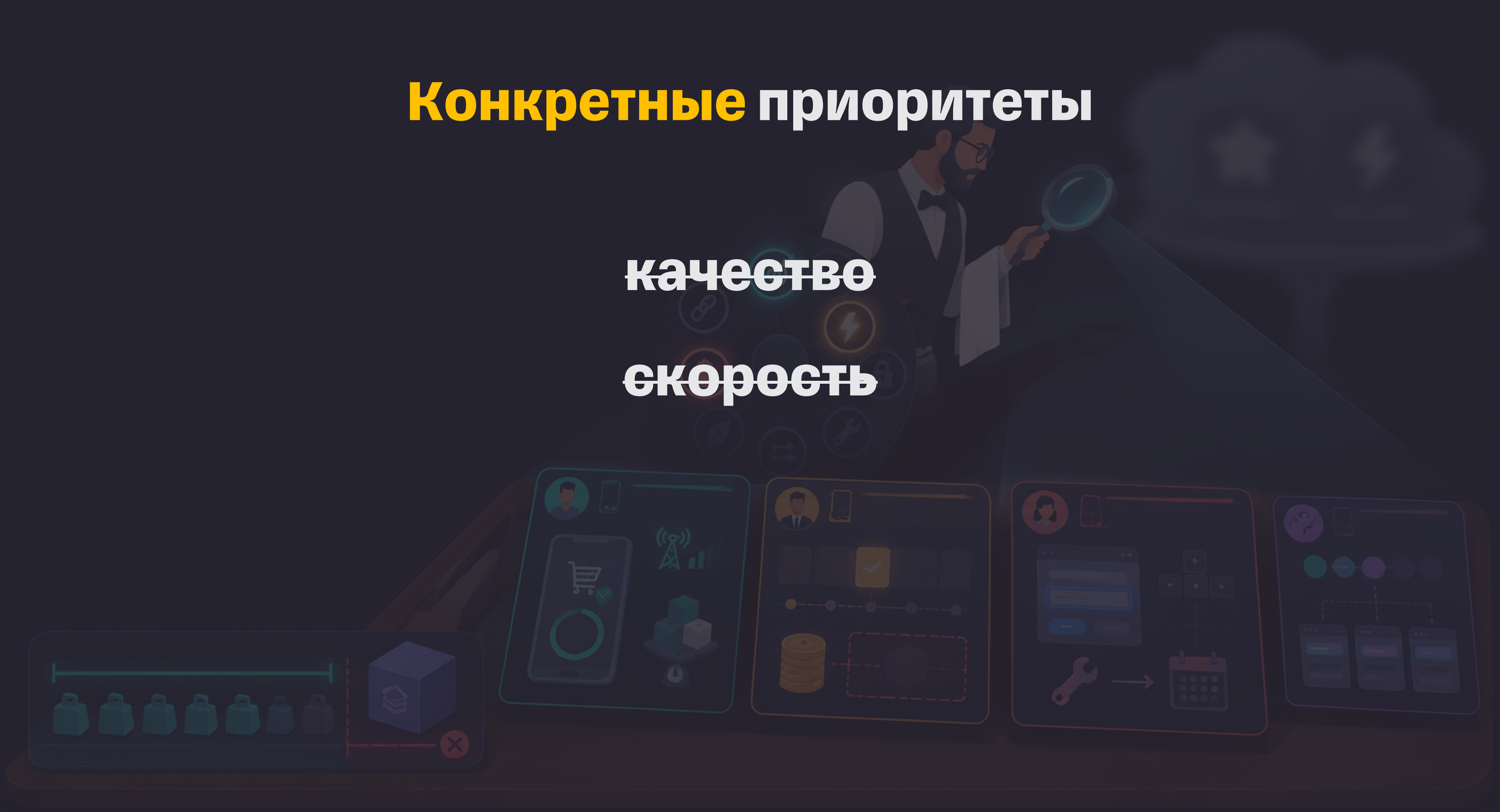
Приоритеты?



Конкретные приоритеты

~~качество~~

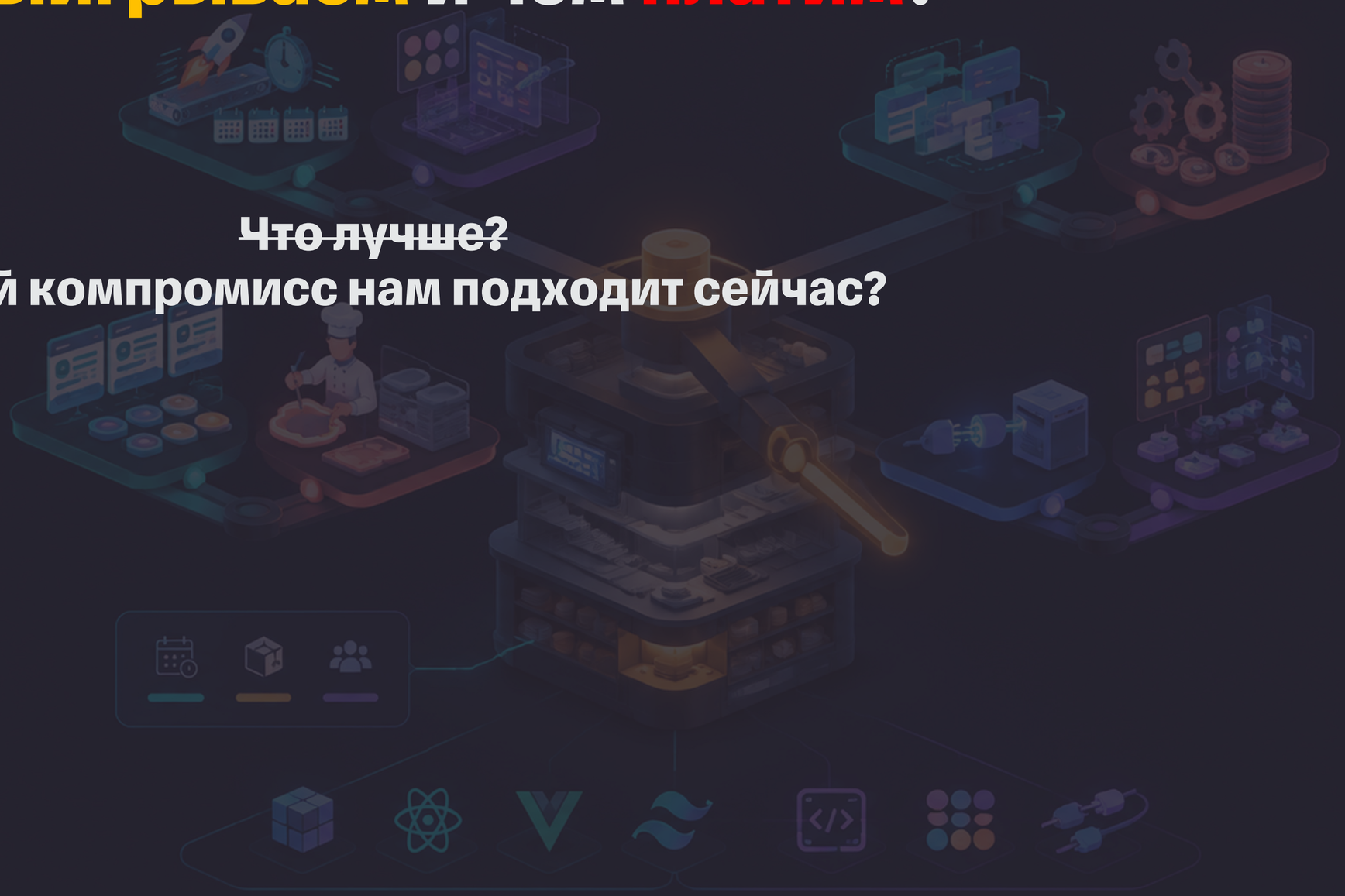
~~скорость~~



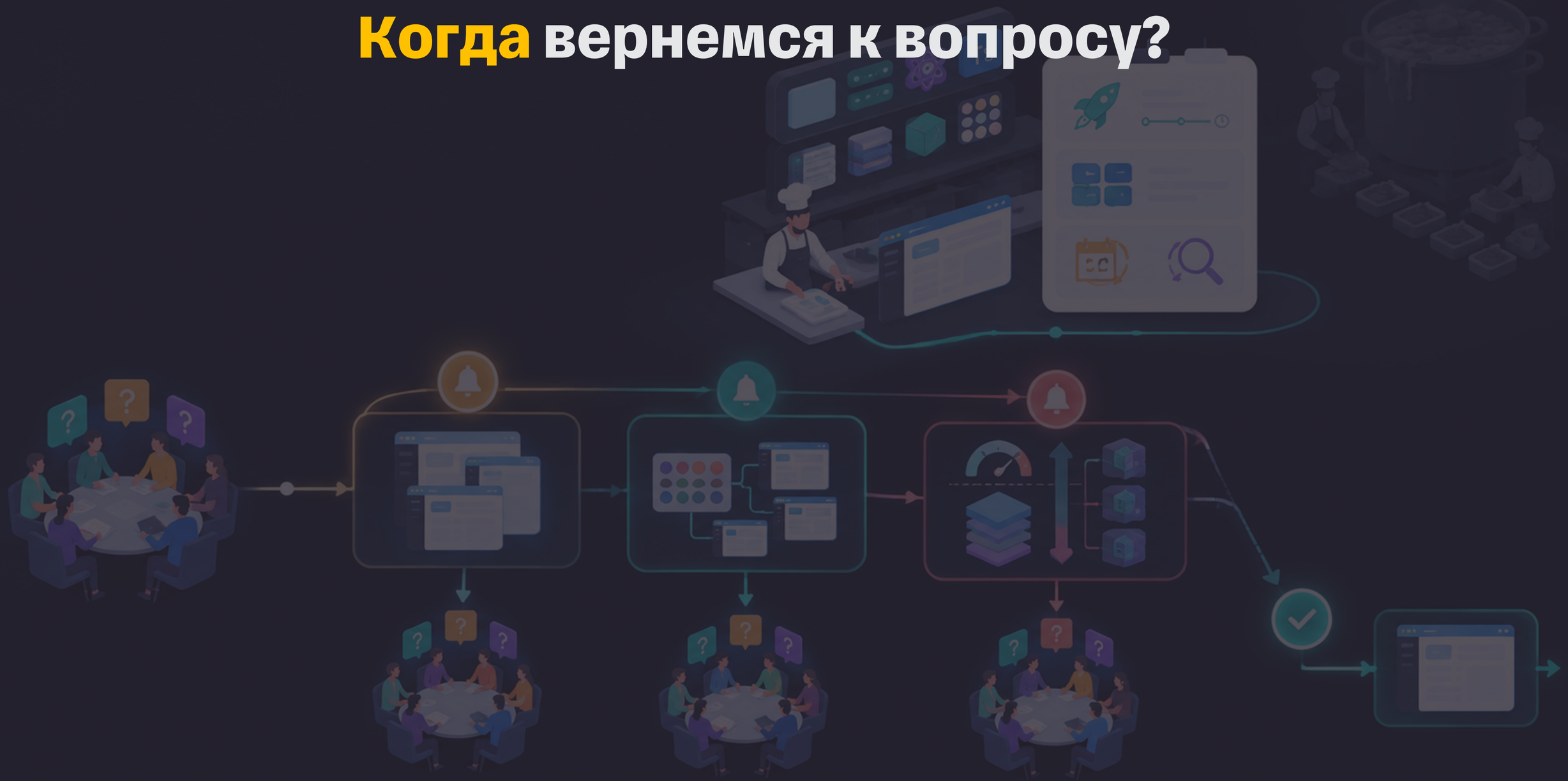
Что выигрываем и чем платим?

~~Что лучше?~~

Какой компромисс нам подходит сейчас?



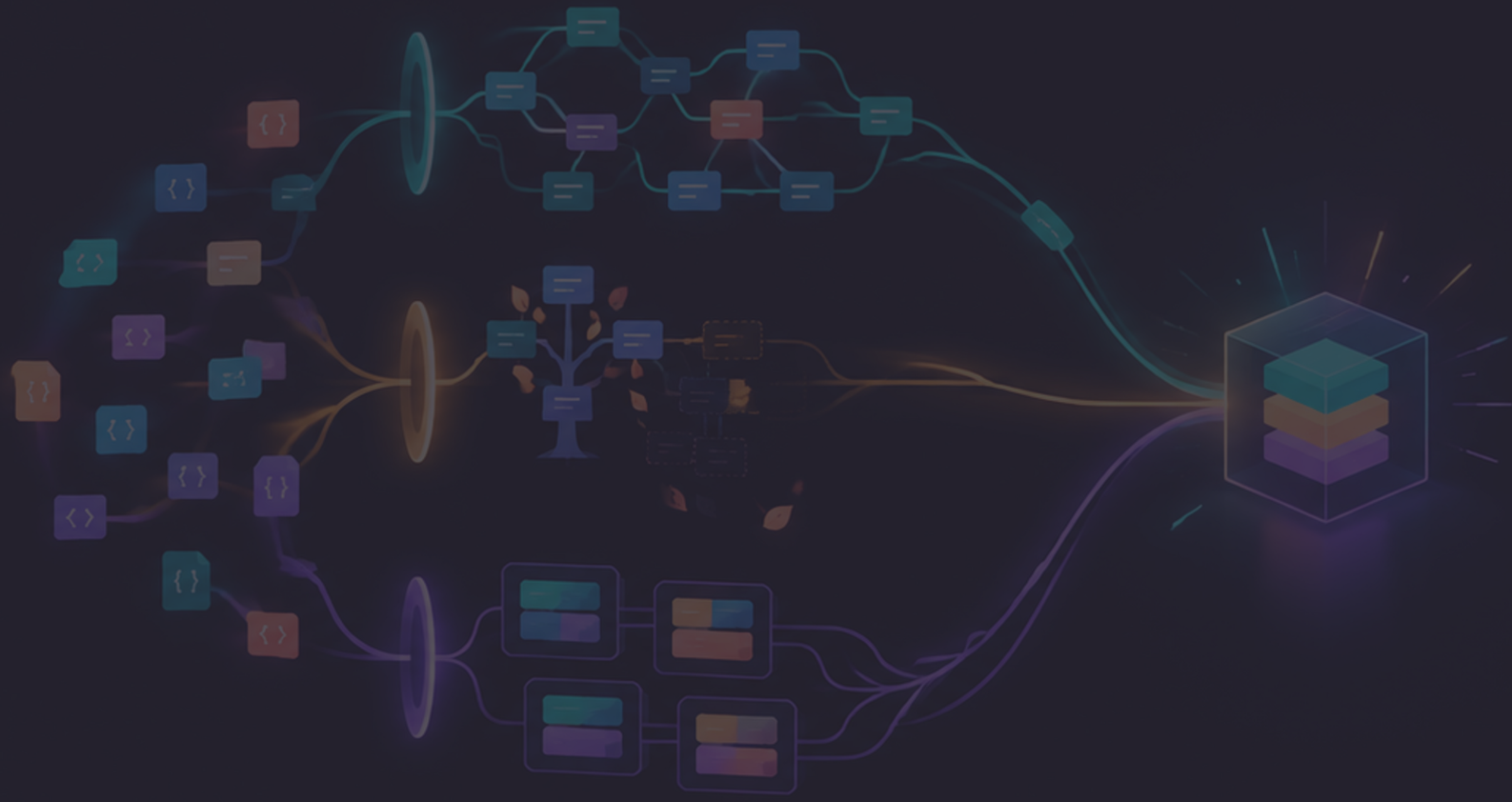
Когда вернемся к вопросу?



А делать то что?



Выбираем бандлер



Что выбираем?

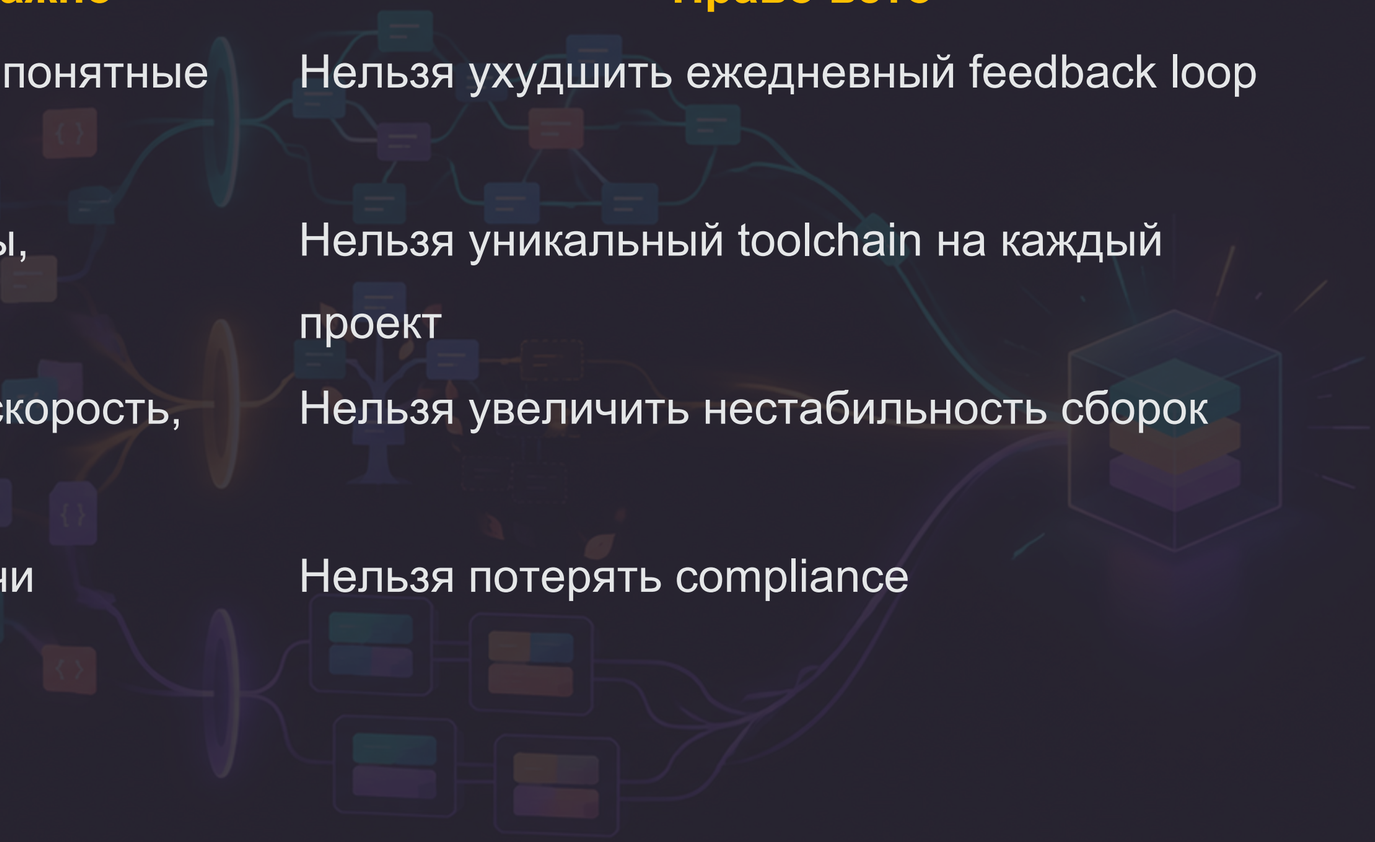
Вопрос	Примеры ответов	На что влияет
Что именно выбираем?	SPA-сборка; pipeline для React-приложения и npm-пакетов; быстрые задачи монорепо	SPA → Vite/Rsbuild/Rspack/webpack. Пакеты → Rollup/tsup. Монорепо → +Turborepo/Nx
Greenfield или миграция?	Новый проект; Webpack 5, 1400 модулей, много loaders	При миграции Rspack обычно уменьшает переписывание, т.к. ориентируется на webpack-конфигурацию
Какие типы артефактов?	SPA; SSR; SSG; web components; npm-библиотека в ESM/CJS	Библиотекам важны форматы экспорта. Приложению — HMR, code splitting, assets, оптимизация bundle

Кто заказчик

Вопрос	Примеры ответов	На что влияет
Кто работает с конфигом и чинит инциденты?	Платформенная команда; каждый product squad; один FrontOps	Мало сильных владельцев → можно низкоуровневую конфигурацию. Десятки команд → шаблоны и мало исключений
Кто главный потребитель выигрыша?	Разработчики; CI; продукт; конечные пользователи	Локальная разработка → cold start/HMR. CI → время pipeline и параллелизм. Пользователь → размер bundle и runtime
Кто платит стоимость поломки?	DevEx; дежурные; команда релиза; продукт	Чем дороже сорванный релиз, тем выше вес обратимости миграции и зрелости экосистемы
Какие навыки уже есть у команды?	«Все знают webpack»; «Привыкли к Vite»	Знакомый стек снижает стоимость внедрения; Rspack рассчитан на совместимость с webpack

Карта интересов

Роль	Что для нее важно	Право вето
Product/frontend-команды	Быстрый старт, HMR, понятные ошибки	Нельзя ухудшить ежедневный feedback loop
Platform/DevEx	Унификация, шаблоны, поддерживаемость	Нельзя уникальный toolchain на каждый проект
CI/release	Воспроизводимость, скорость, стабильность	Нельзя увеличить нестабильность сборок
Security/legal	SBOM, лицензии, патчи	Нельзя потерять compliance

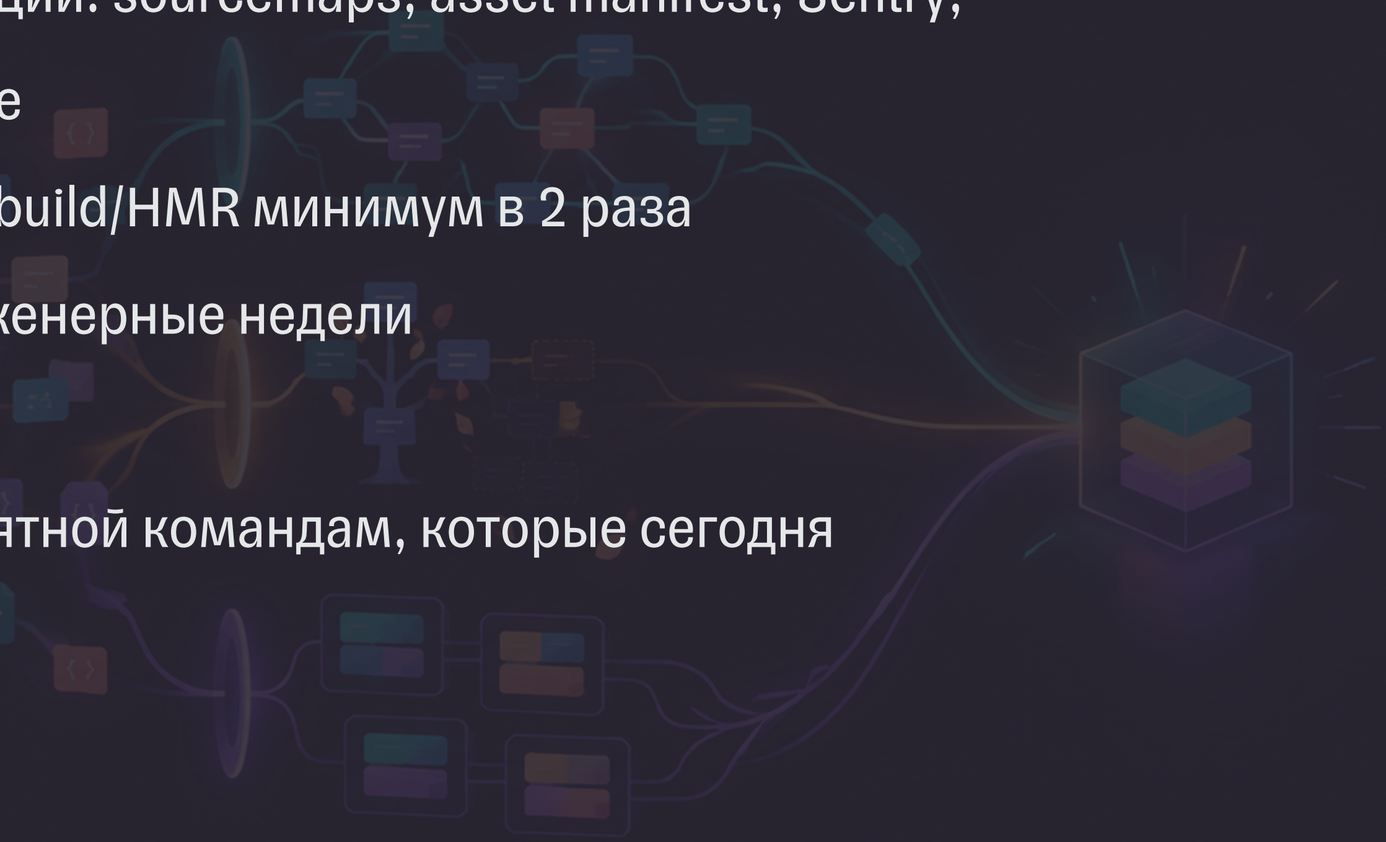


Что критично сейчас?

Вопрос	Что измерить или выяснить	Как ответ влияет на выбор
Где именно теряется время: install, cold start, HMR, production build, typecheck, tests или CI queue?	p50/p95 по каждому этапу; локально и в CI	Не лечите медленный typecheck заменой bundler. Возможно, сборщик не главный bottleneck: нужно вынести `tsc --noEmit`, lint и тесты в независимые кэшируемые задачи.
Какой performance target нужен?	«Dev server ≤ 5 с cold start»; «HMR p95 ≤ 1 с»; «production build ≤ 3 мин»; «CI ≤ 12 мин»	Если текущая webpack-сборка функционально подходит, а проблема — скорость, в shortlist попадёт Rspack/Rsbuild или точечная оптимизация webpack. Rspack использует SWC для преобразований и предлагает встроенный SWC loader вместо `babel-loader`; это может менять миграционные и performance-решения
Нужна ли поддержка старых браузеров?	…	…
Какие пакеты и плагины являются блокерами?	…	…

Пример приоритетов

1. Не потерять production-функции: sourcemaps, asset manifest, Sentry, CSS pipeline, env-переменные
2. Сократить p95 локального rebuild/HMR минимум в 2 раза
3. Уложить миграцию в 2–3 инженерные недели
4. Не повысить flaky rate CI
5. Оставить конфигурацию понятной командам, которые сегодня поддерживают webpack

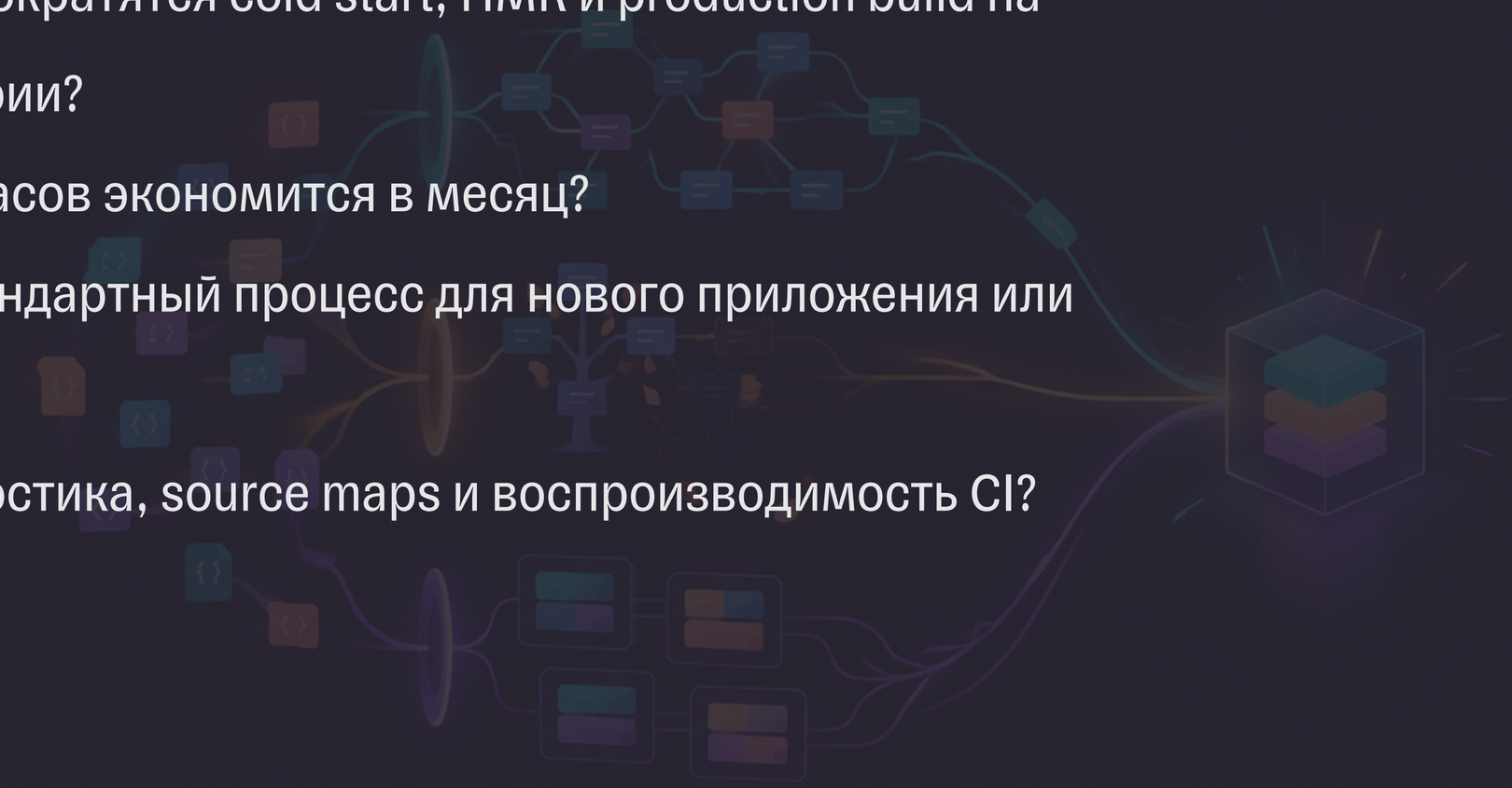


Что выигрываем и чем платим



Вопросы о выгоде

- На сколько секунд сократятся cold start, HMR и production build на реальном репозитории?
- Сколько человеко-часов экономится в месяц?
- Станет ли проще стандартный процесс для нового приложения или пакета?
- Улучшатся ли диагностика, source maps и воспроизводимость CI?



Вопросы о цене и риске

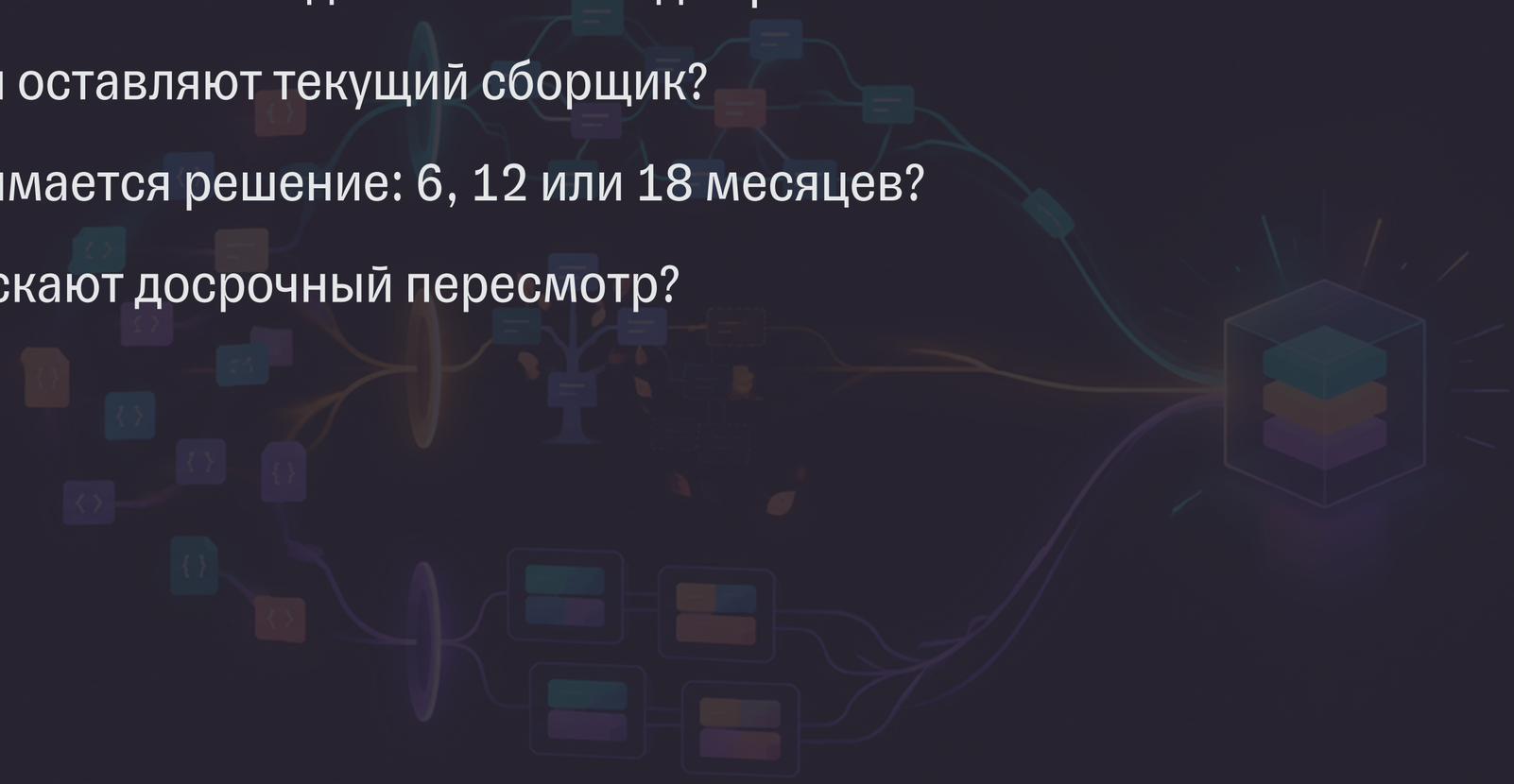
Вопрос	Ответ	Что значит ответ
Есть нестандартные webpack plugins/loaders или плагины, использующие недокументированные internals?	Да	Риск миграции высокий: Rspack поддерживает много webpack-плагинов, но несовместимость возможна, особенно при опоре на internals. Требуется отдельный spike/adapter или сохранение webpack. rspack+1
Есть собственные Babel transforms?	Да	Нужна проверка: можно ли перенести логику в SWC/plugin API или оставить Babel в цепочке без потери ожидаемого выигрыша.
Есть критичный Module Federation/микрофронтенды?	Да	Сделайте federation обязательным сценарием PoC: dev, production, shared singleton dependencies, обновление remote, error handling, sourcemaps.
Понадобится поддерживать два сборщика одновременно?	Да	Закладывайте затраты на двойные CI pipelines, расхождение конфигов, debugging и срок удаления старого пути.
Нужно менять lockfile, Node.js, менеджер пакетов или Docker image?	Да	Учитывайте security approvals, кэш CI, onboarding и rollback
Можно ли легко откатиться?	Нет	Увеличьте планку доказательств до переключения по умолчанию; оставьте feature flag или параллельную сборку до стабилизации.

Матрица решения

Критерий	Вес	Vite	Rspack/Rsbuild	webpack 5	Rollup/tsup
Совместимость с текущим webpack pipeline	25%	Оценка после PoC	Оценка после PoC	5	Обычно не основной кандидат для app
Скорость HMR на реальном проекте	20%	Замер	Замер	Baseline	Не основной сценарий
Production build и размер артефактов	15%	Замер	Замер	Baseline	Сильнее релевантно библиотекам
Сложность миграции	15%	Оценка	Оценка	5, если не мигрировать	Оценка
Надёжность CI и кэширование	10%	Замер	Замер	Baseline	Зависит от оркестратора
Поддерживаемость командой	10%	Оценка	Оценка	Оценка	Оценка
Экосистемные блокеры	5%	PoC	Проверить compatibility list	5	PoC

Когда вернемся к ответу

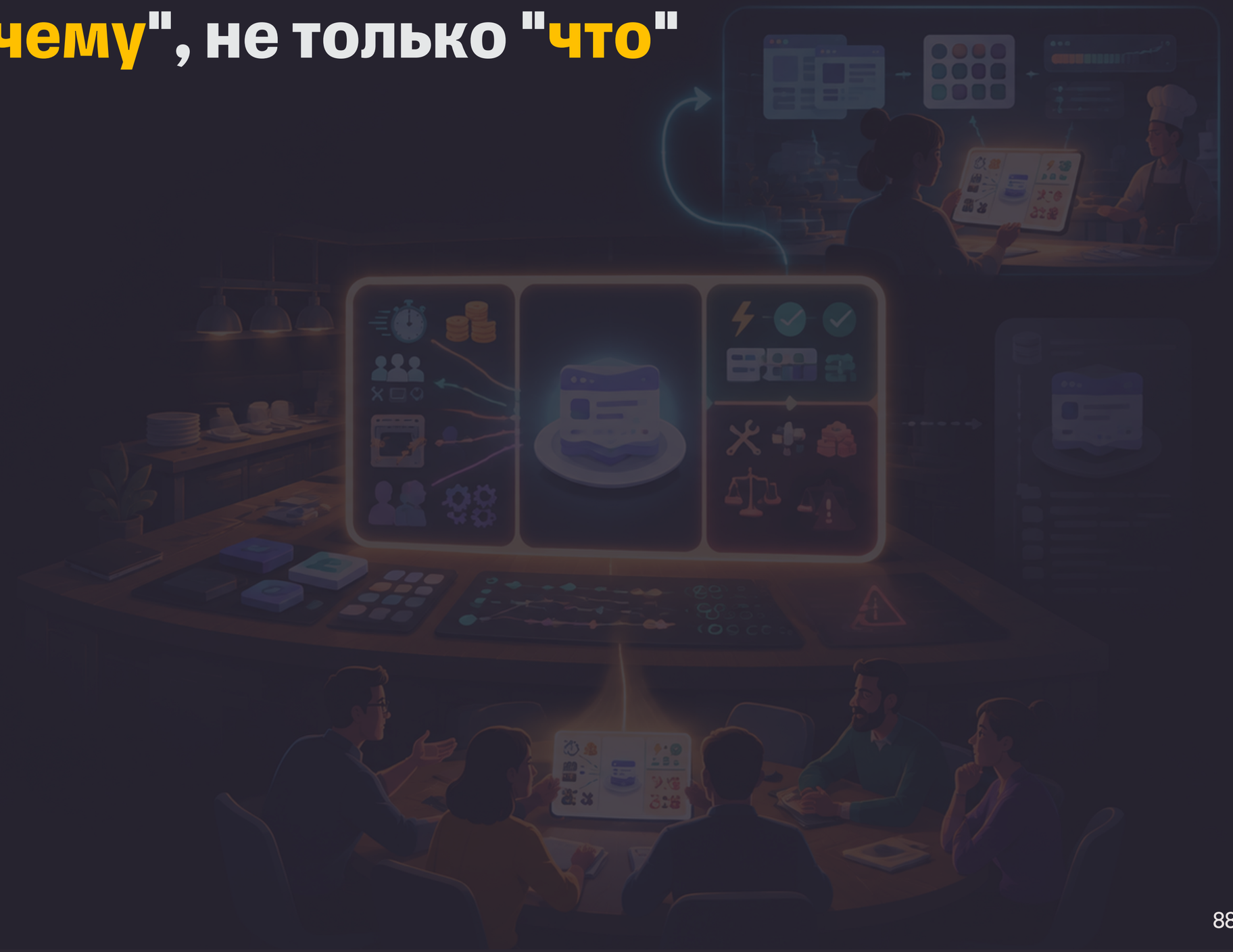
- Какие результаты PoC считаются достаточными для решения?
- Какие стоп-факторы оставляют текущий сборщик?
- На какой срок принимается решение: 6, 12 или 18 месяцев?
- Какие события запускают досрочный пересмотр?



Принцип вне контекста – тоже меню без гостя



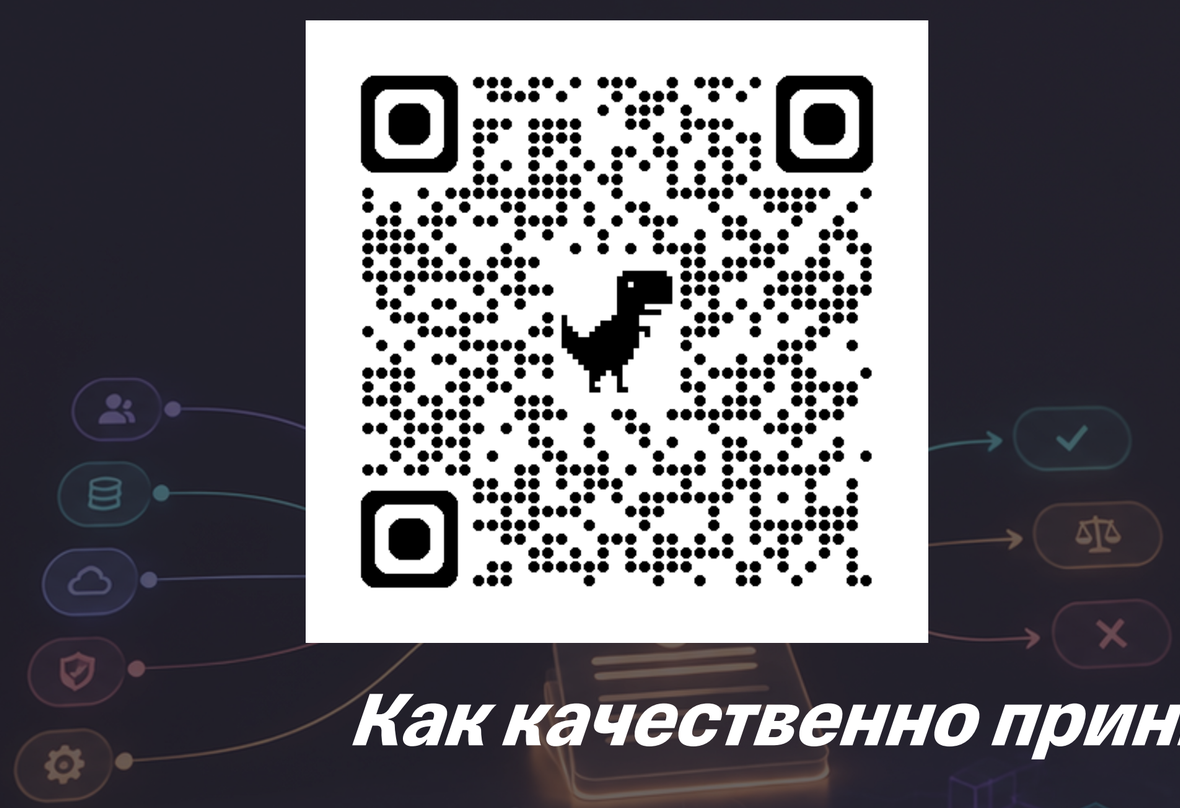
Запишите "почему", не только "что"



Нет привычке



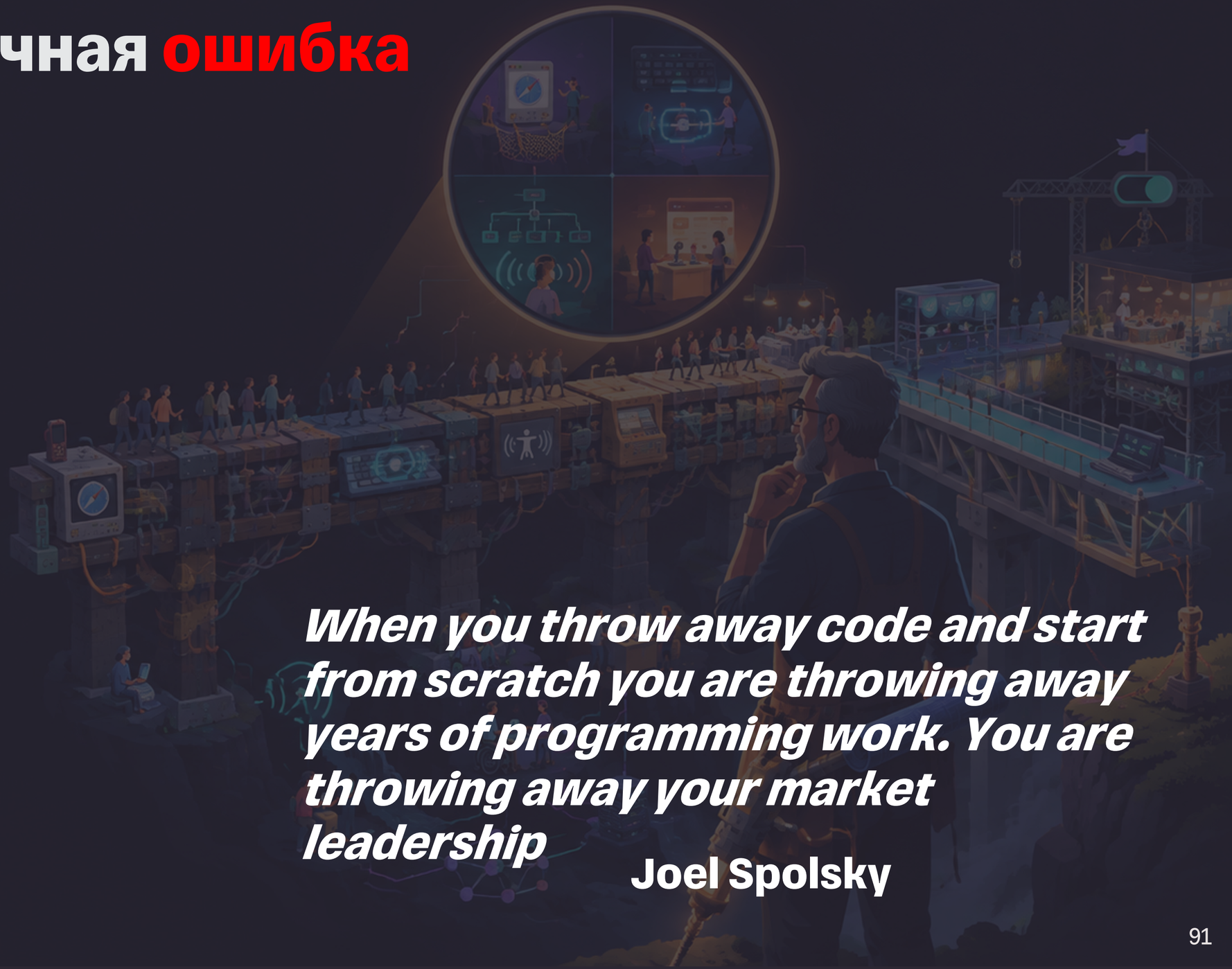
Рекомендую посмотреть



Как качественно принимать архитектурные решения

Павел Востриков

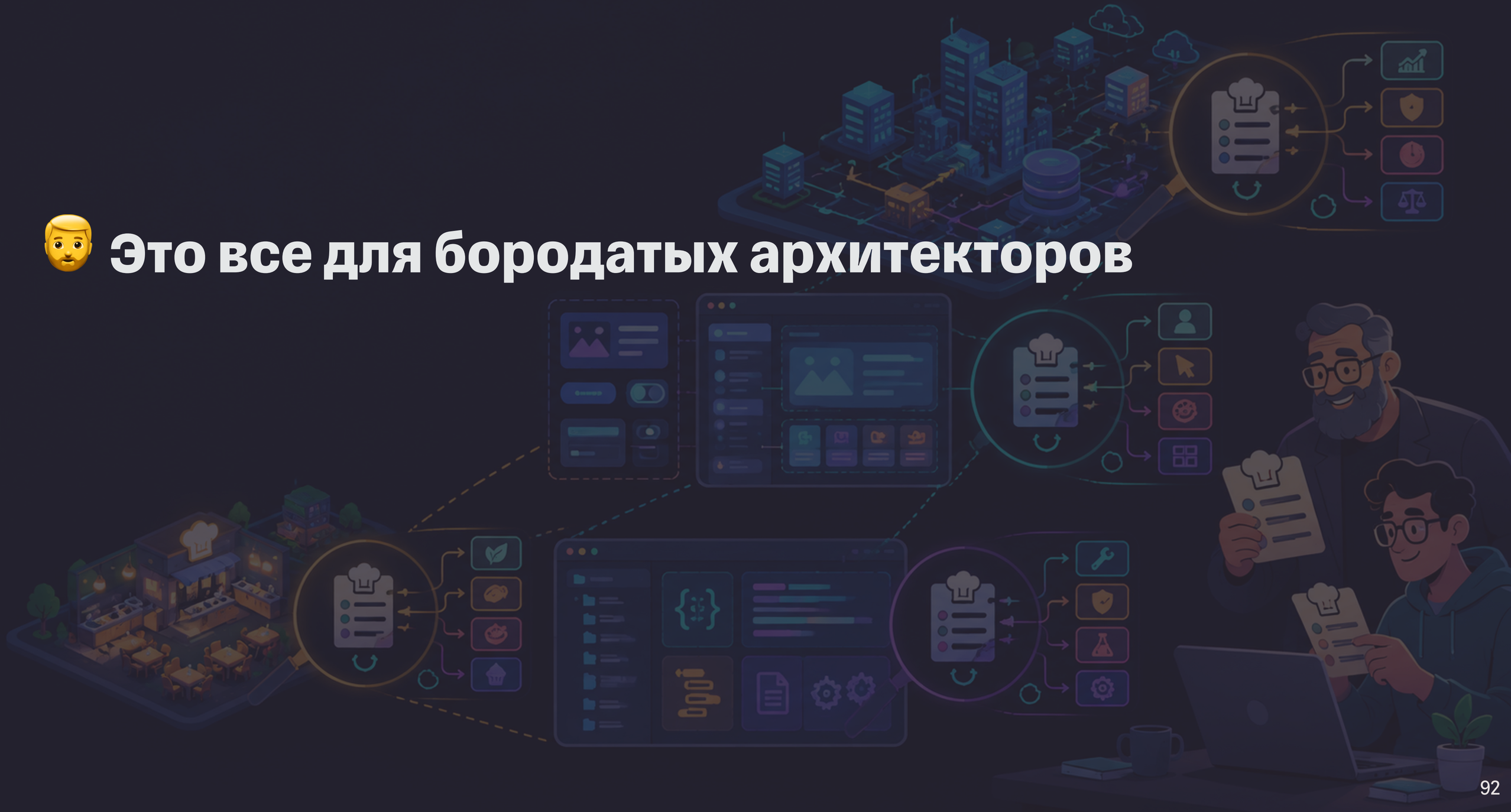
Симметричная ошибка



When you throw away code and start from scratch you are throwing away years of programming work. You are throwing away your market leadership

Joel Spolsky

👱 Это все для бородатых архитекторов



Играть с самим собой ✊



Зайти с другой стороны BACK



Мораль

- Ценится умение решать задачи заказчика
- Разговаривать и проявлять интерес полезно
- Перед тем как сделать – спроси
- Собери контекст, а не расскажи свой опыт



