

Микросервисная архитектура + DDD

Наш подход к созданию сложных систем

> Красота и гибкость в простоте



по Сергей Стародубцев

8 урок курса



Что нас ждет сегодня

Проблема

Монолит vs. Микросервисы — выбор архитектуры

Формула успеха

Наш подход — Микросервисы + DDD

Компоненты архитектуры

Контексты, API, Домены

Прагматизм

Почему мы избегаем лишней сложности

Цель

Создание платформы для нашего Core Domain



Монолит vs. Микросервисы

Одна гигантская фабрика или специализированный технопарк?



Монолит

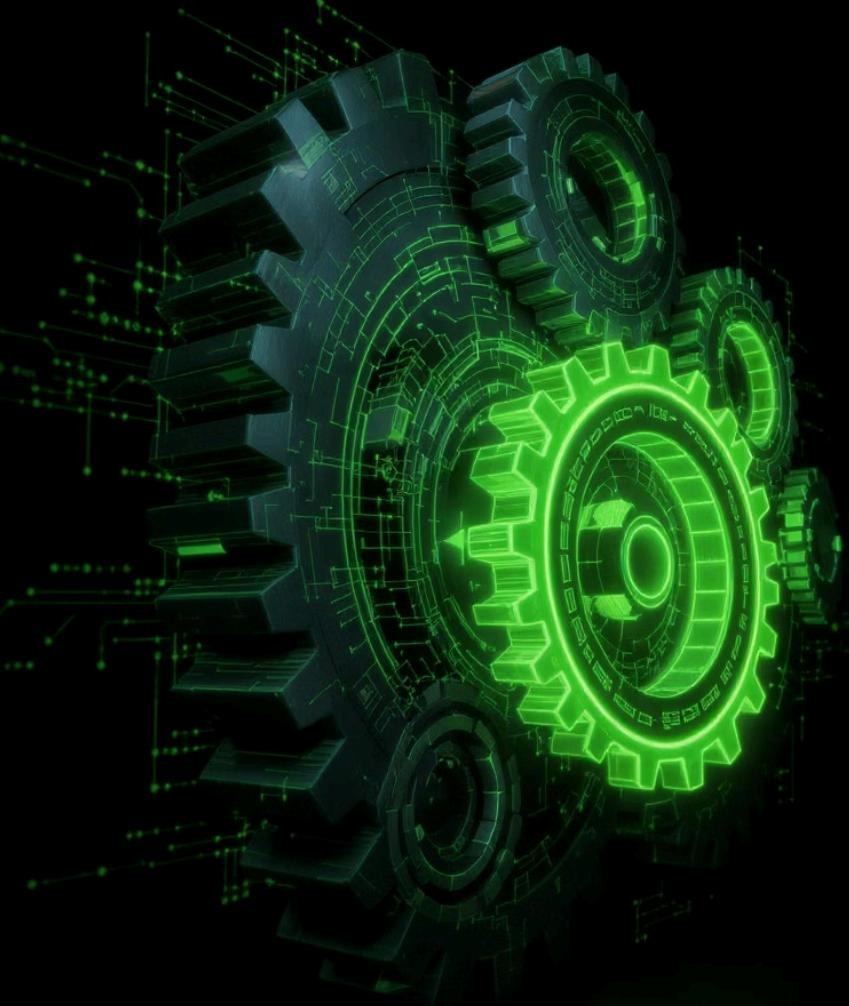
Легко начать, но сложно поддерживать и масштабировать. Ошибка в одной части останавливает всё.



Микросервисы

Сложнее в начале, но обеспечивают гибкость, независимость и масштабируемость. Ошибка в одном сервисе не затрагивает другие.

Наш выбор для серьезного проекта — [Микросервисы](#).



Наш подход: Микросервисы + DDD

Организованная Автономность

Микросервисы

→ дают нам **Автономность**

DDD

→ дает нам **Организованность**

Вместе они создают архитектуру, где каждый компонент имеет четкую бизнес-ценность, ясные границы и может развиваться независимо.

Шаг 1: Контекст → Микросервис (Стены)

Каждый Bounded Context становится отдельным микросервисом

Микросервис — это наши "стены". Он защищает целостность языка и логики внутри контекста.

 Контекст "Каталог"

Микросервис *catalog*

 Контекст "Заказы"

Микросервис *orders*

 Контекст "AI Рекомендации"

Микросервис *recommender*



Шаг 2: Единый язык → API (Двери)

API микросервиса — это его официальный язык общения

Микросервисы не лезут во "внутреннюю кухню" друг друга. Они общаются через **публичные API**, которые являются четким и формальным **контрактом**.

- ⓘ GET /api/v1/products/{product_id} — это официальное "письмо", отправленное в "посольство" сервиса catalog.



Шаг 3: Домены → План застройки

Что именно мы строим? Ответ дает стратегическое проектирование DDD

Generic Domain

Интегрируем готовое решение
(Платежи)

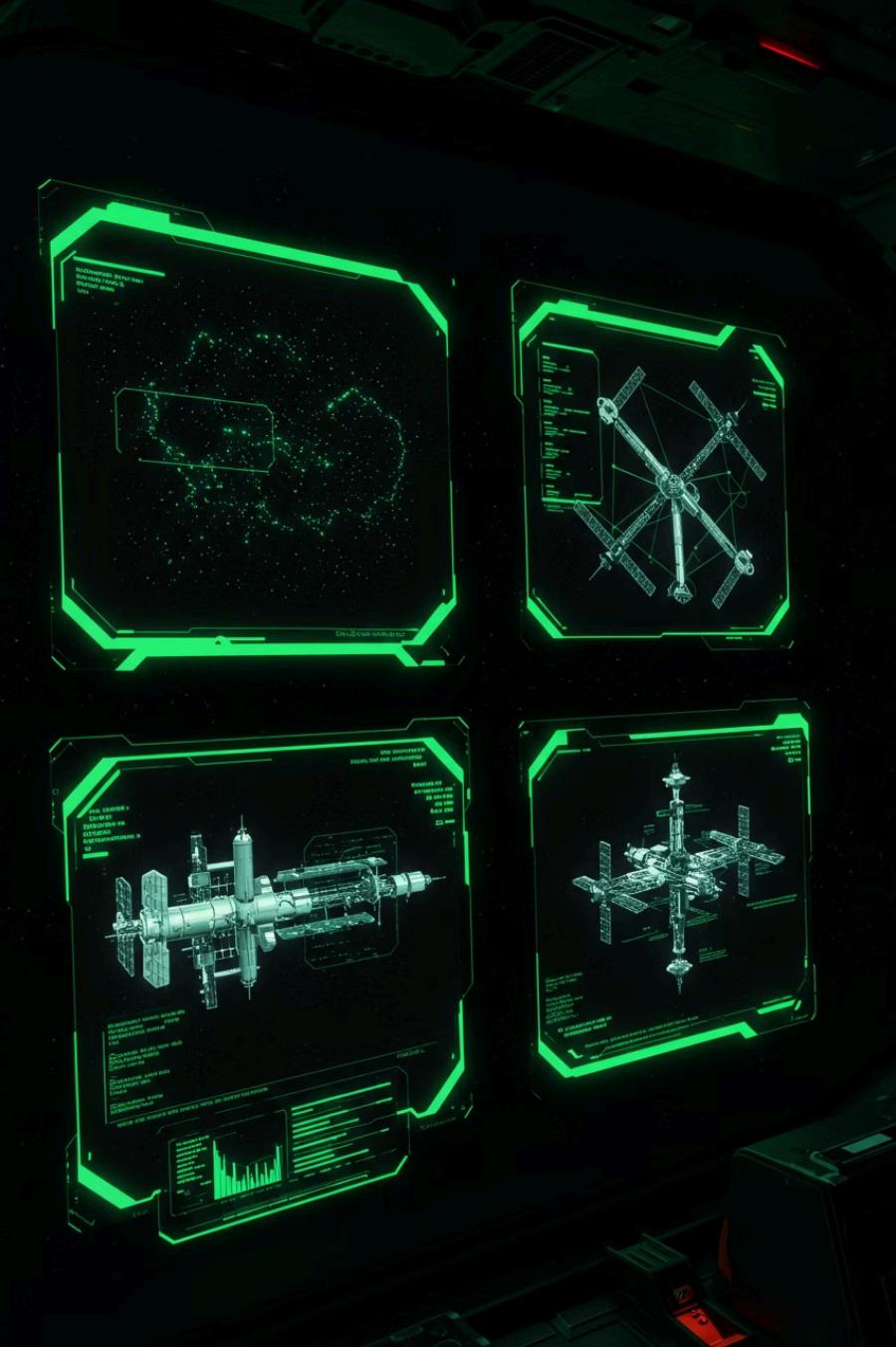
Supporting Domain

Строим стандартный, надежный
сервис (Пользователи)

Core Domain

Инвестируем все силы! Строим выделенный, самый мощный микросервис
(AI Рекомендации)





Наша архитектура в сборе

Простая, мощная и осмысленная система

Стратегия

Определяем **Домены**

Границы

Очерчиваем **Ограниченные Контексты**

Реализация

Воплощаем Контексты в **Микросервисы**

Коммуникация

Организуем общение через **API**

Наш прагматичный выбор

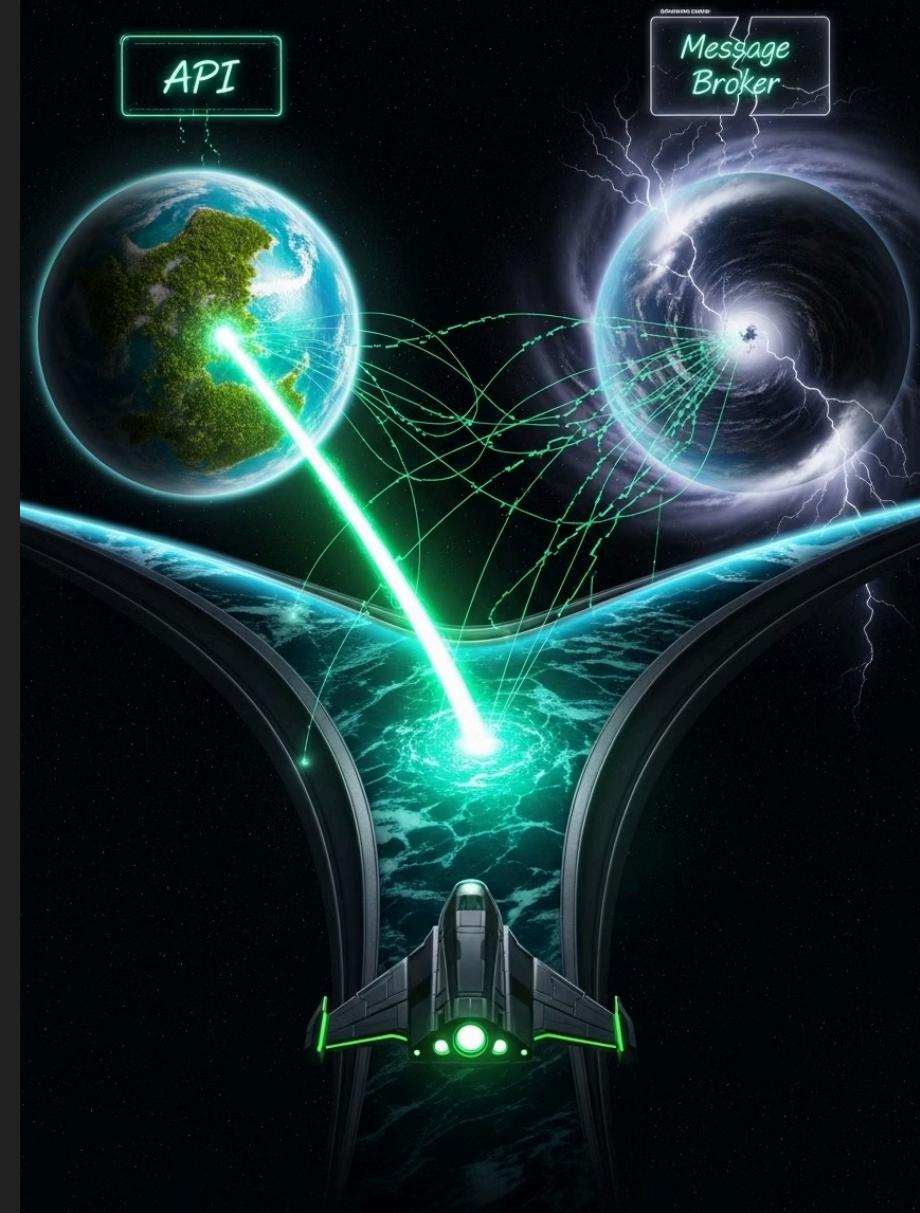
Принцип: избегать случайной сложности

Мы сознательно **не используем** брокеры сообщений (Kafka, RabbitMQ) в курсе.

Почему?

- **Прямые API-вызовы** намного проще для понимания, реализации и отладки
- Это самый прагматичный и эффективный подход для **95%** проектов, включая наш

Мы учим вас строить надежно и просто!



Ключевые выводы

- 1** Микросервисы + DDD — наш путь к организованной автономности
- 2** **Один контекст — один микросервис.** Это наш главный принцип управления сложностью
- 3** **Общение только через API.** Уважаем границы и обеспечиваем слабую связанность
- 4** **Простота — это добродетель.** Мы выбираем самые прямые и понятные решения





Готовы к практике?

Время построить наш Core Domain!

На практическом занятии мы:



Объявим AI-рекомендации
нашим Core Domain



Создадим для него новый,
изолированный
микросервис recommender



Напишем код, который
сделает наш магазин по-
настоящему уникальным