Bounded Context

Ограниченный контекст: разделяй и властвуй

>_ Ограниченный контекст — помогает структурировать сложные системы и организовать взаимодействие между различными частями программного обеспечения

по Сергей Стародубцев

> 7 урок курса

Что нас ждет сегодня

Проблема

Война смыслов и двусмысленность в коде

Решение

Bounded Context как «лингвистическая граница»

Открытие

Микросервисы = Физическое воплощение контекстов

Преимущества

Автономность, гибкость и скорость

Практика

Карта Контекстов и роль Админ-панели

Проблема: Война смыслов

Одно слово «Товар» — три разных значения

Для Маркетинга (Каталог)

Обложка, описание, промо-цена.

Для Логистики (Заказы)

Артикул, вес, остаток на складе.

Для Бухгалтерии (Финансы)

Себестоимость, НДС, цена закупки.

Результат: Код становится монстром Франкенштейна — сложным, запутанным и хрупким. Различные команды борются за право определять, что такое "товар", создавая конфликты и технический долг.

Решение: Bounded Context

Устанавливаем четкие границы, внутри которых слова однозначны

> **Bounded Context** — это граница, внутри которой каждый термин из Единого Языка имеет только один, строго определенный смысл.

В DDD мы принимаем реальность: невозможно создать единый язык для всей системы. Вместо этого мы создаем "контексты" — явно определенные области, где каждое понятие имеет четкое значение.

Больше никакой войны — в каждом «государстве» свои законы и свой язык. Каждый контекст имеет свою модель данных, свои бизнес-правила и даже свою техническую реализацию, оптимизированную под конкретные задачи.

Открытие: Микросервисы = Физические границы

Наши микросервисы — это и есть техническое воплощение Ограниченных Контекстов

Микросервис catalog

Контекст «Каталог»

Управляет ассортиментом, описаниями и маркетинговыми атрибутами товаров

Микросервис orders

Контекст «Заказы»

Обрабатывает корзину, оформление и статусы заказов

Микросервис auth

Контекст «Аутентификация»

Отвечает за регистрацию, авторизацию и управление пользователями

Каждый микросервис — это автономное «здание» со своими стенами, правилами и даже собственной базой данных.

Микросервисная архитектура — это практическое воплощение идеи ограниченных контекстов.

Преимущества четких контекстов

Границы — это не ограничение, а свобода

Автономность команд

Разные команды могут работать над своими контекстами параллельно, не мешая друг другу и быстрее доставляя ценность.

Ясность и простота

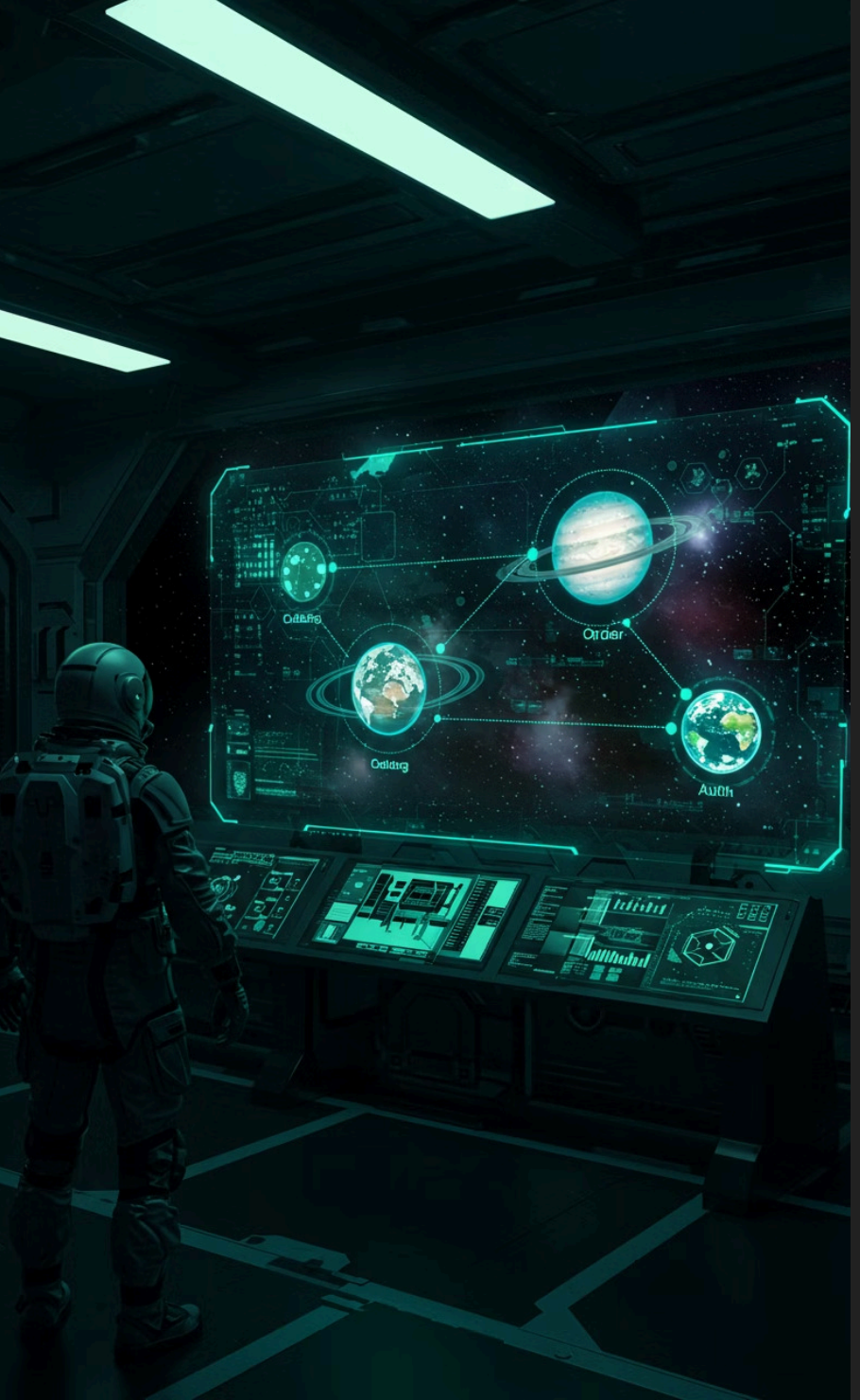
Код внутри каждого контекста намного проще и сфокусирован на одной задаче, что уменьшает сложность и повышает надежность.

Гибкость технологий

Можно использовать Python для одного сервиса и Go/Rust для другого — того, который требует высокой производительности.

Независимое масштабирование

Можно увеличить мощность только для сервиса каталога, если на него идет основной трафик, экономя ресурсы.



Карта Контекстов (Context Map)

Как независимые «государства» общаются друг с другом?

> **Карта Контекстов** — это высокоуровневая диаграмма, которая показывает все контексты в системе и типы отношений между ними.

Стратегический взгляд

Помогает увидеть всю систему целиком, понять ее структуру и выявить узкие места

Типы отношений

- Customer-Supplier (Заказчик-Поставщик)
- Conformist (Конформист)
- Anti-corruption Layer (Антикоррупционный слой)

Коммуникационные протоколы

- REST API
- gRPC
- Очереди сообщений

Это наш стратегический взгляд на всю архитектуру, наши «торговые пути» между независимыми сервисами.

Админ-панель: Контекст-Координатор

Наш «Центр управления полетами»

Админ-панель — это особый тип клиента, который не имеет своей бизнес-логики. Ее задача — управлять другими контекстами через их публичные, официальные API.

Она — директор, который общается с начальниками отделов, не вмешиваясь в их внутреннюю работу:

Нужны товары?

Запрос в контекст Catalog через API

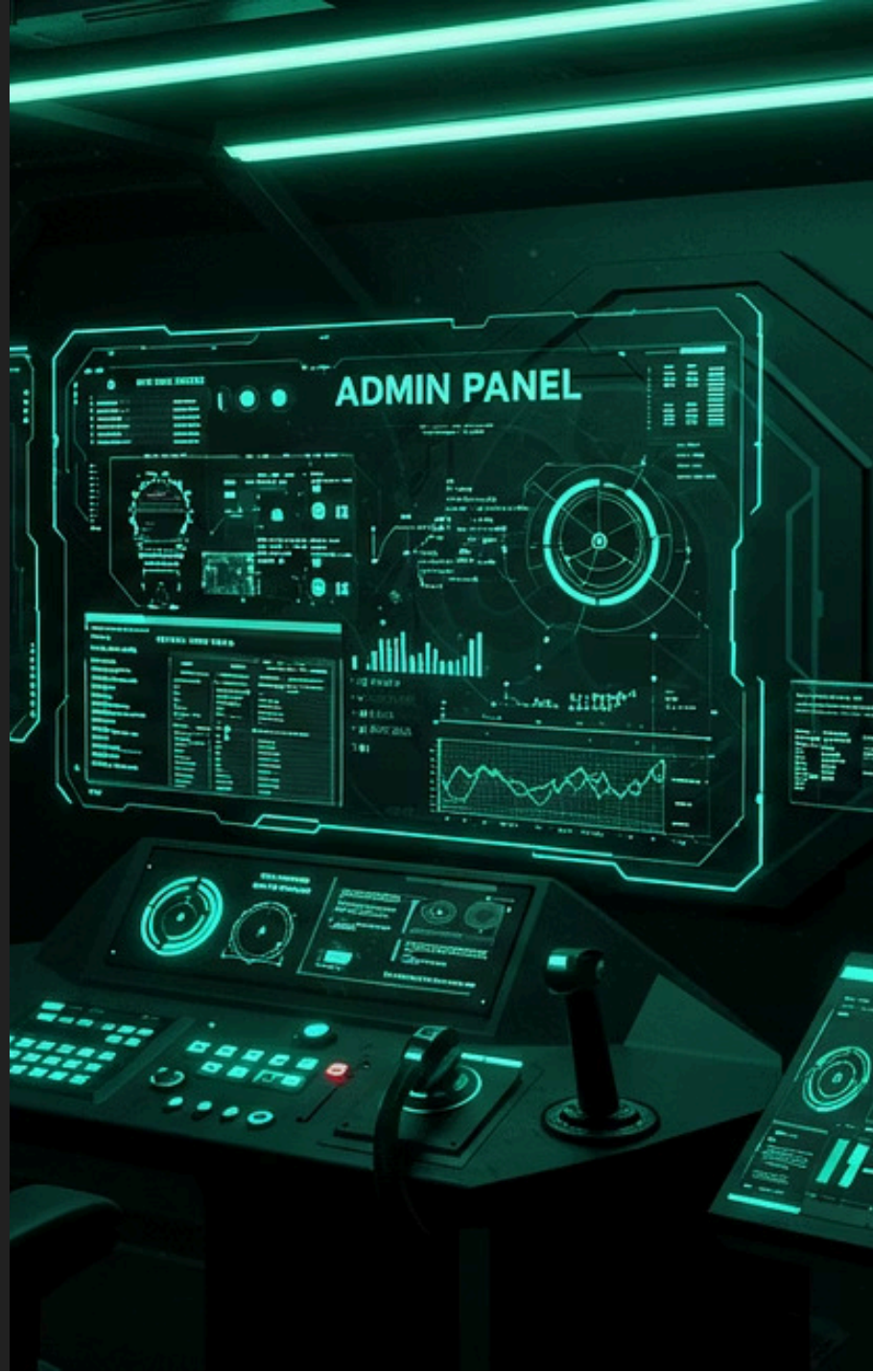
Нужны заказы?

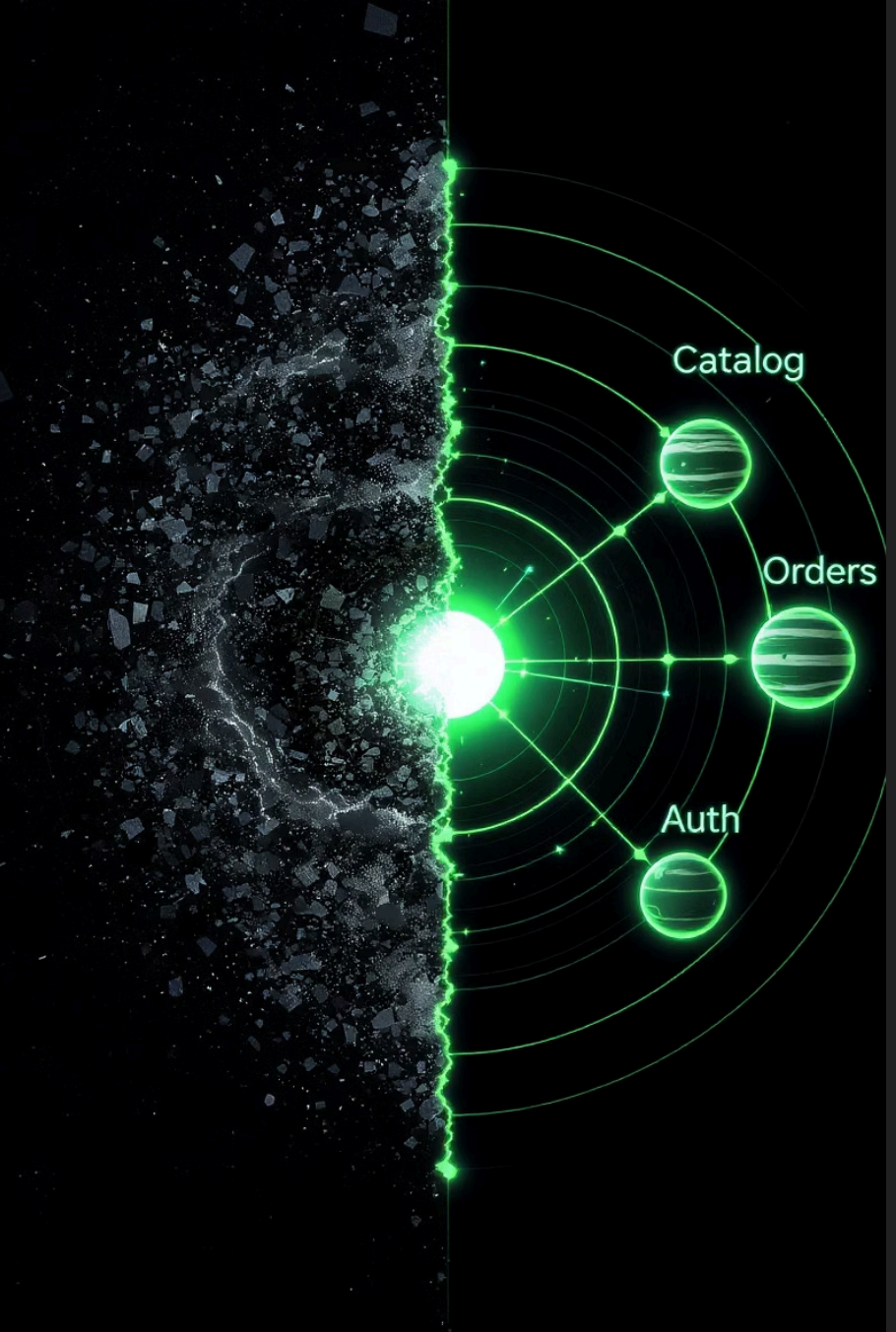
Запрос в контекст Orders через API

Управление пользователями?

Запрос в контекст Auth через API

Админ-панель — это наглядный пример того, как различные контексты могут быть объединены на уровне пользовательского интерфейса без смешивания их внутренней логики.





Ключевые выводы

1 **Bounded Context наводит порядок**

Устраняет двусмысленность и разделяет систему на логические части с четкими границами ответственности

2 **Микросервисы — идеальные «стены» для контекстов**

Обеспечивают автономность, изоляцию и независимое развитие каждого контекста

3 **Контексты общаются через API**

Сохраняя независимость и уважая границы друг друга, следуя четко определенным протоколам взаимодействия

4 **Админ-панель — идеальный пример координатора**

Демонстрирует, как объединить функциональность разных контекстов, не нарушая их границы

Готовы к практике?

Увидим магию взаимодействия контекстов вживую!

На практическом занятии мы:

Создадим с нуля админ-панель, которая будет взаимодействовать с разными контекстами

Расширим API контекста «Каталог» (CRUD-операции)

Настроим взаимодействие между контекстами через REST API

Практический результат:

- ▢ Увидим, как товар, добавленный в админке, тут же появляется на сайте — наглядная демонстрация правильной интеграции контекстов!
- ▢ Убедимся, что каждый контекст сохраняет свою автономность, но при этом все вместе они создают целостную систему.