Do modular ao microsserviço com PHP

Leonardo Tumadjian

PHPConference 2025

Sobre mim

PHP +15 anos

ZCPE(Zend Certified PHP Engineer)

Arquiteto de Soluções no Grupo Fênix

Professor de cursos livres / Mentoria

Gamer quando dá tempo

Harleyro nas horas vagas

Conheça a DifferDev
Site: differdev.com
Youtube: DifferDev

Nossa grade

1. Conceitos de Monolito Modular em PHP
2. Diferenças entre modularização tradicional e microsserviços
3. Estratégias de separação de responsabilidades (Repository, Strategy Pattern, Domain Isolation)
4. Organização de código PHP por "módulos isolados"
5. Comunicação entre serviços (HTTP, fila de mensagens com RabbitMQ)

Alguns Quotes:

Arquitetura é sobre Trade-offs

“Não existe uma arquitetura melhor em absoluto. A escolha de um padrão arquitetural (microserviços, monolito, event-driven) envolve fazer concessões entre atributos de qualidade (desempenho, escalabilidade, disponibilidade, custo, etc.).”

Mark Richards e Neal Ford

Alguns Quotes:

Arquitetura é sobre “Coisas” Importantes

“A arquitetura de software trata de decisões que são difíceis de reverter, que têm um grande impacto no sistema e que afetam a organização da equipe.”

- Martin Fowler

O terreno é importante

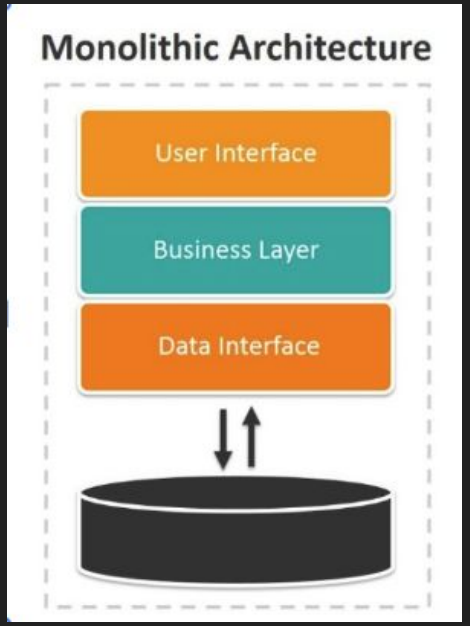
Assim como na engenharia civil, precisamos antes de mais nada avaliar toda a solução, seja ela a nível de infraestrutura até sobre manter a evolução do sistema.

Isso nos leva a seguintes questões:

O que é um monolito?

Uma arquitetura onde todos os componentes da aplicação (interface, lógica de negócios, banco de dados) estão interligados e são implantados como uma única unidade.

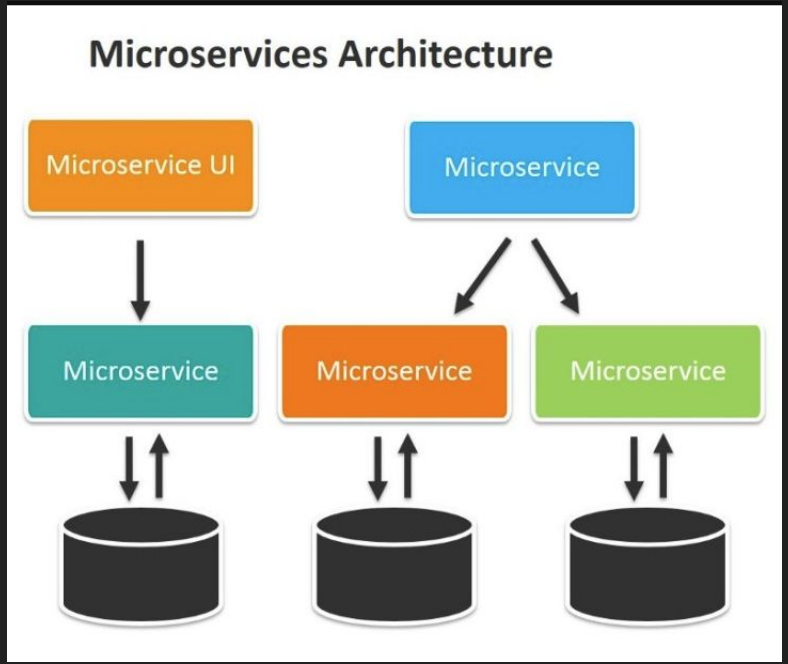
99% dos projetos que nascem monolito, morrem monolito, devido a falta de estratégia inicial.



O que é um microsserviço?

O microsserviço opera de forma independente, com sua própria base de código e, frequentemente, seu próprio banco de dados. Isso significa que a falha em um serviço não derruba o aplicativo inteiro.

Nem sempre o microsserviços será a solução, pelo contrário, há casos de rollback, como o recente caso da Amazon Prime Video.



[Why amazon reverted to a monolith?]

Casos de uso não funcionais

1. Quantos usuários devemos atender inicialmente?
2. Quais partes serão as mais frágeis?
3. Quais as comunicações externas?
4. Quais partes podem ser reaproveitadas?
5. Pontos sensíveis a falhas de segurança?

Custo operacional vs viabilidade

É aqui onde muitos erram, porque há alguns pontos a se considerar que vão além de código, como:

1. Nível da equipe
2. Tamanho da empresa
3. Possíveis outros produtos!
4. Mudança de negócio
5. Custo de infraestrutura
6. Monitoramento e instrumentação

Uma opção

O monolito modular, para que possa atender futuras evoluções:

Prós:

1. Isolamento lógico
2. Organização
3. Zero latência entre os módulos
4. Contratos diretos

Contras:

1. Aumento da complexidade
2. Cuidado à violação de módulo
3. Tempo de entrega maior(inicial)
4. DB relacional dará trabalho

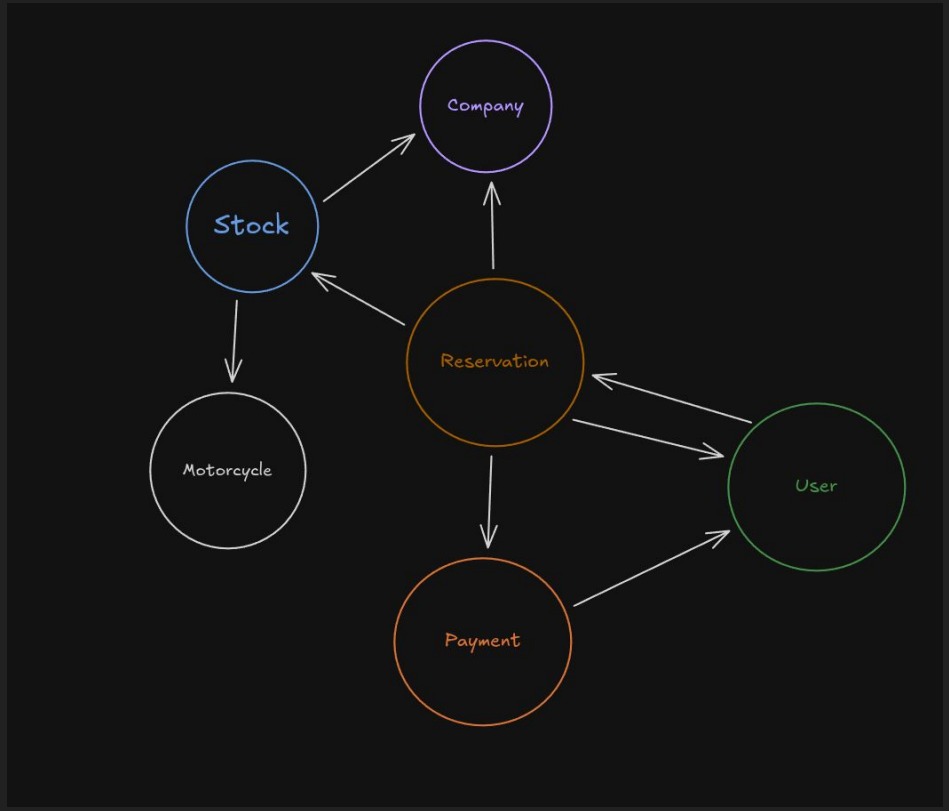
Um caso de uso

Locação de motos elétricas:

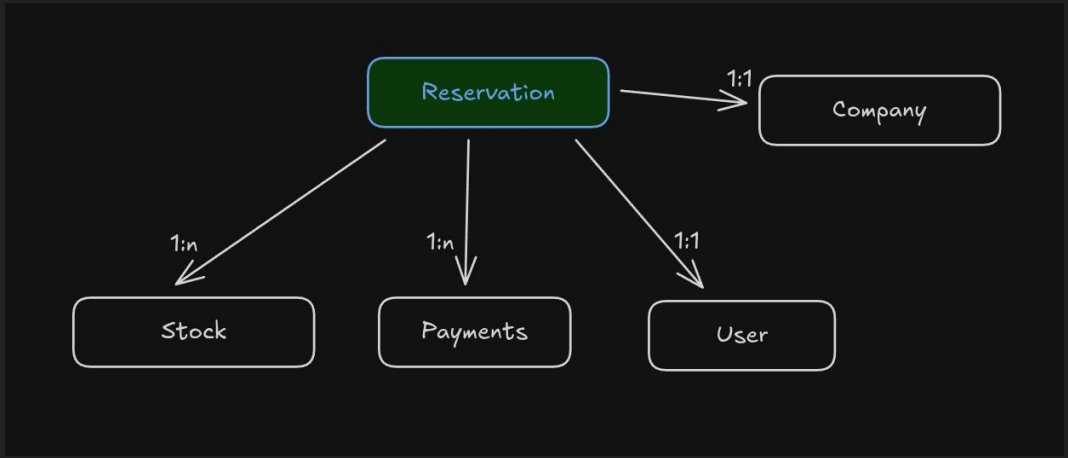
O sistema deve fornecer locação temporária para motocicletas elétricas na cidades de São Paulo.

1. Users/Authe
2. Motorcycles
3. Stock
4. Reservation
5. Payment
6. Messages
7. Shared (Módulo global compartilhado entre todos domínios)

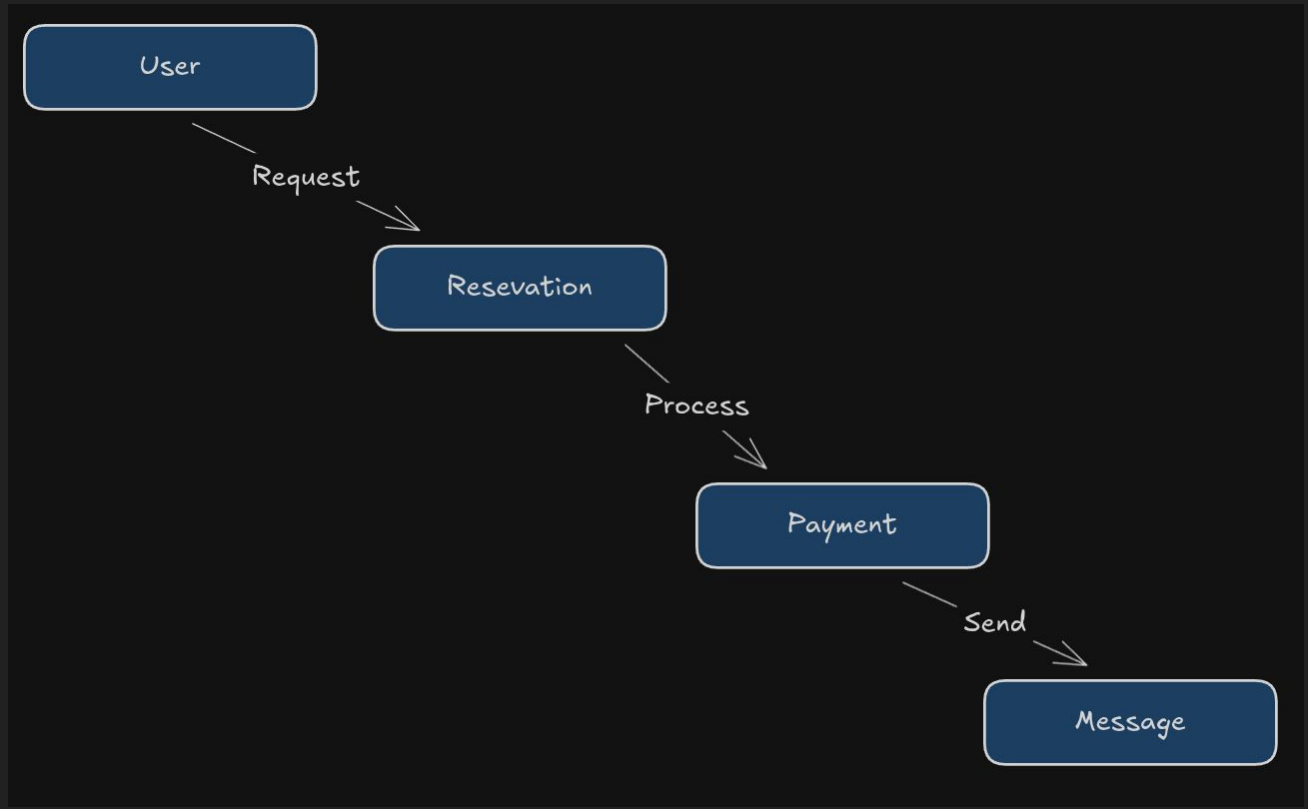
Achando as relações



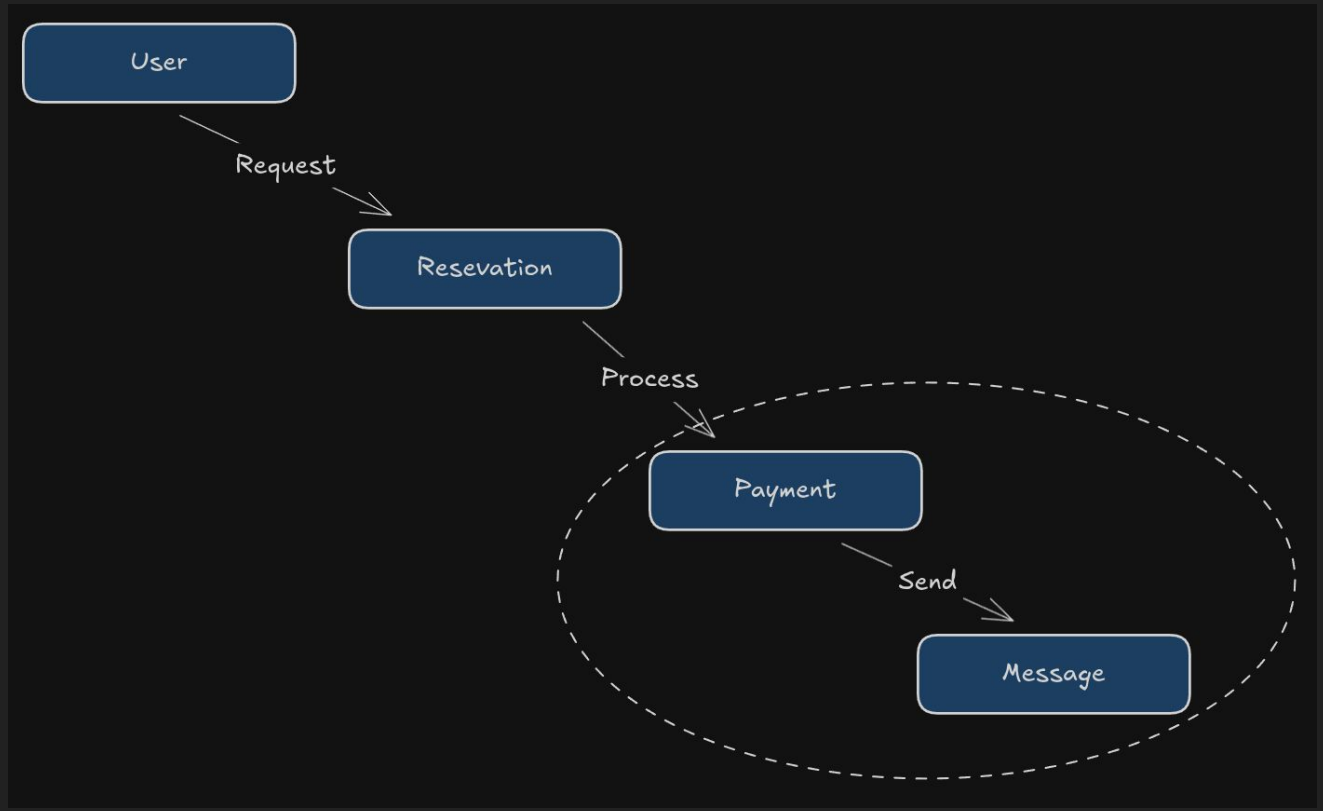
Achando as Root Aggregates



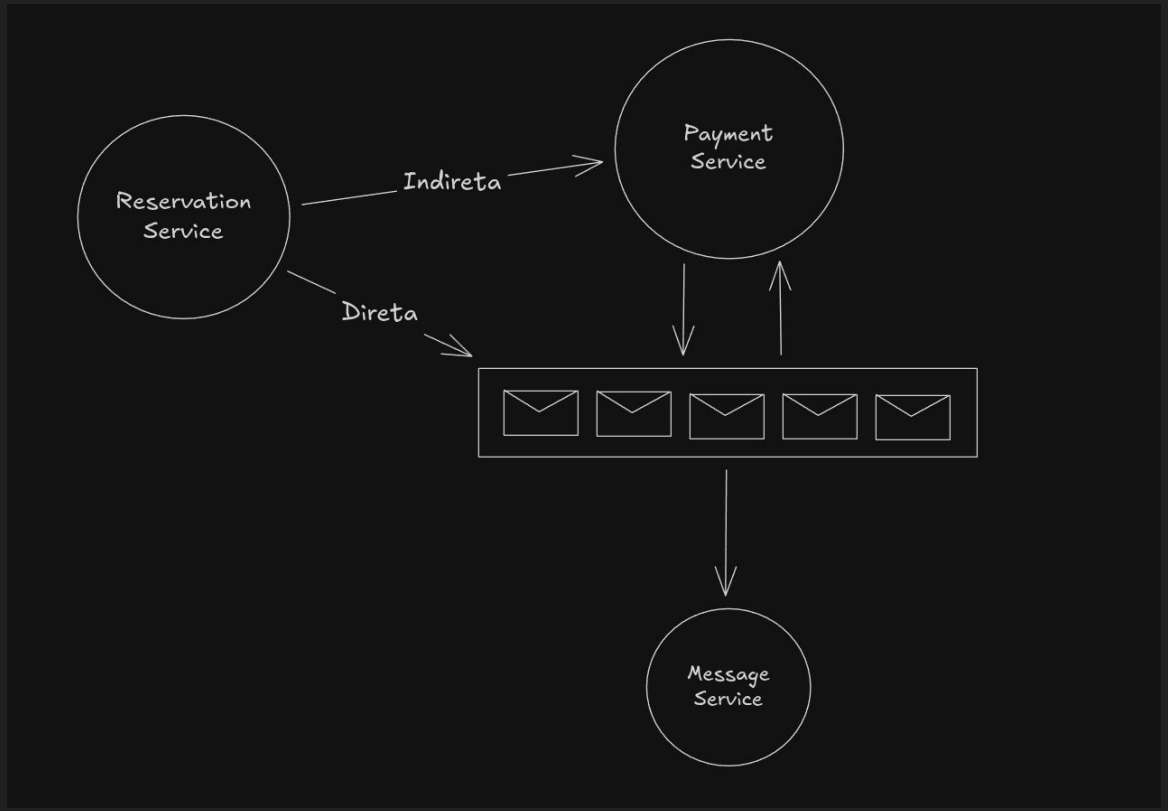
Caso de uso principal



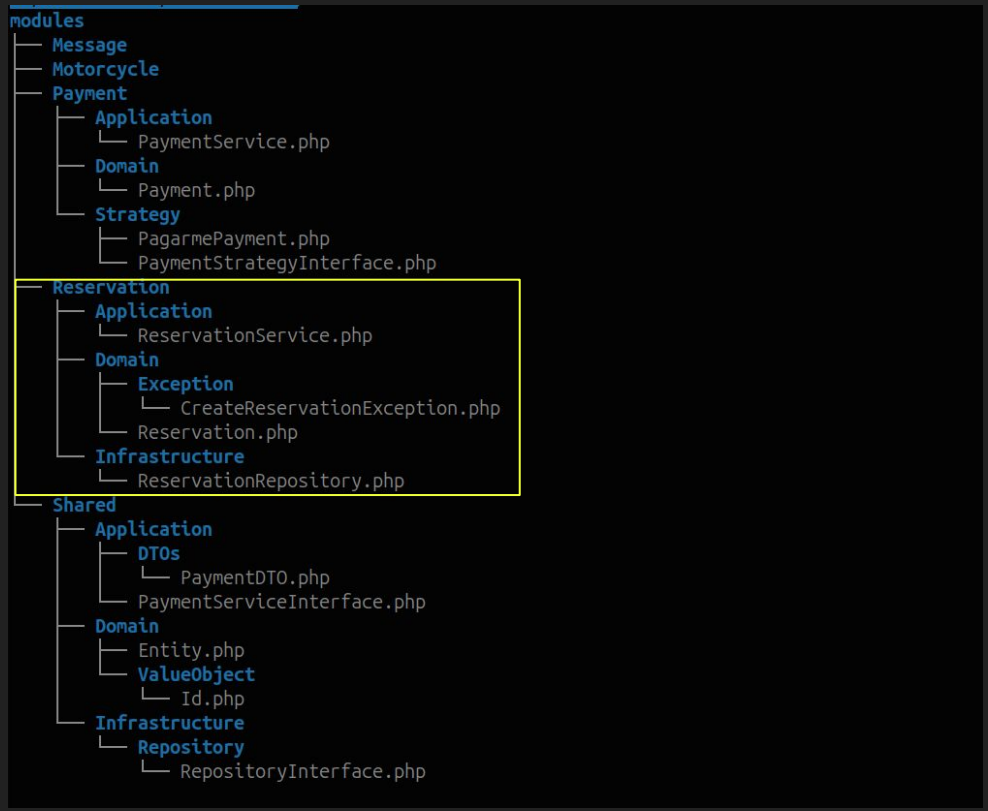
Encontrando pontos de escala



Levando para microsserviços



Como preparar módulo para ser desplugado



Como preparar módulo para ser desplugado

1. Desnormalizar relações
2. Isolar a comunicação (infrastructure)
3. Manter a anti-corrupção em Serviços e Domínios
4. Inverter dependências para comunicação externa

Mão na massa - Aula01

1. Estrutura e regras modulares
2. Criar um service para salvar reservas:

Controller > ReservationService > ReservationModel > ReservationRepository

3. Revisar todo o ponto e os motivos pelo qual devemos trabalhar
4. Pontos extras:
   1. virar o status do stock
   2. Armazenar o pagamento(mocking)

Mão na massa - Aula02

1. Estrangular o módulo de pagamento
2. Criar um projeto aparte (02-microsservicos)
3. Testar a geração de pagamento(como serviço)
4. Adaptar a comunicação entre Monolito(Reservation) para Microsserviço(Payment)

Obrigado

Dúvidas finais?