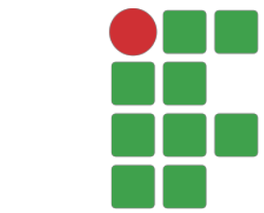


INSTITUTO FEDERAL
Paraíba

Campus Cajazeiras

PROGRAMAÇÃO P/ WEB 2

2. DECOMPONDO APLICAÇÕES EM MICROSSERVIÇOS

PROF. DIEGO PESSOA

DIEGO.PESSOA@IFPB.EDU.BR

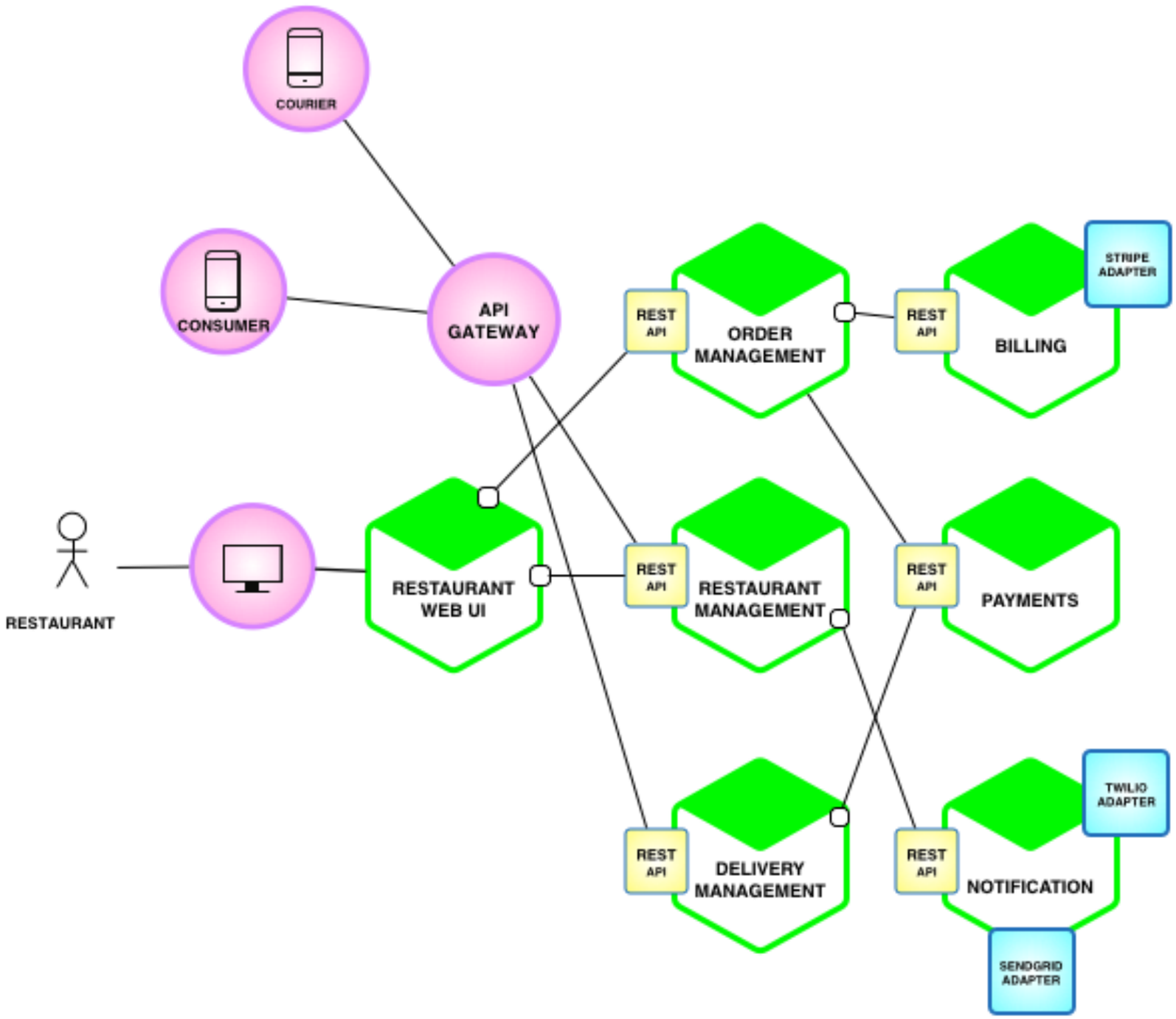
@DIEGOEP

CST em Análise e Desenvolvimento de Sistemas

RESUMO

- Como decompor uma aplicação em microsserviços
- Como usar o conceito de "bounded context" do DDD (Domain-Driven Design) para compreender os dados e facilitar a decomposição

EXEMPLO DE ARQUITETURA DE MICROSSERVIÇOS

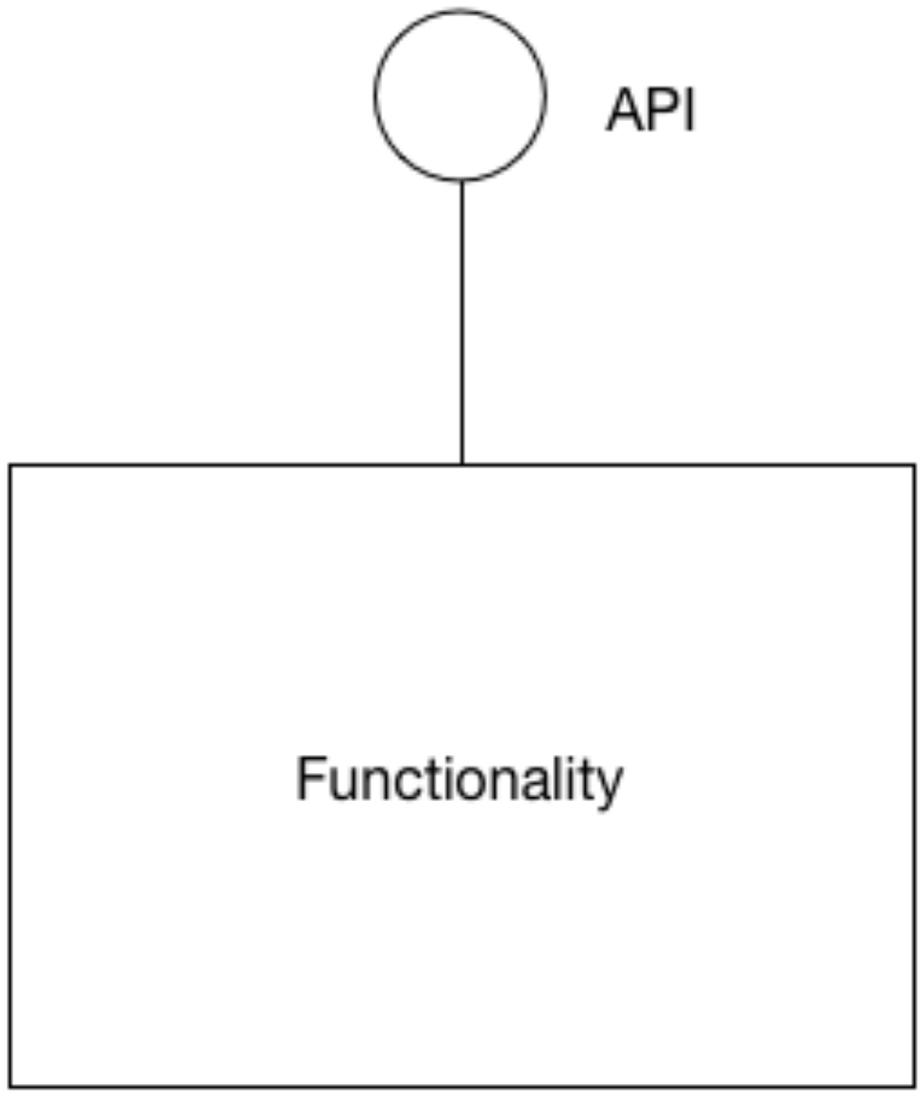


MAS O QUE É UM SERVIÇO?

> ""A MECHANISM TO ACCESS AN UNDERLYING CAPABILITY ""

https://en.wikipedia.org/wiki/Service_(systems_architecture)

VISÃO ABSTRATA DE UM SERVIÇO



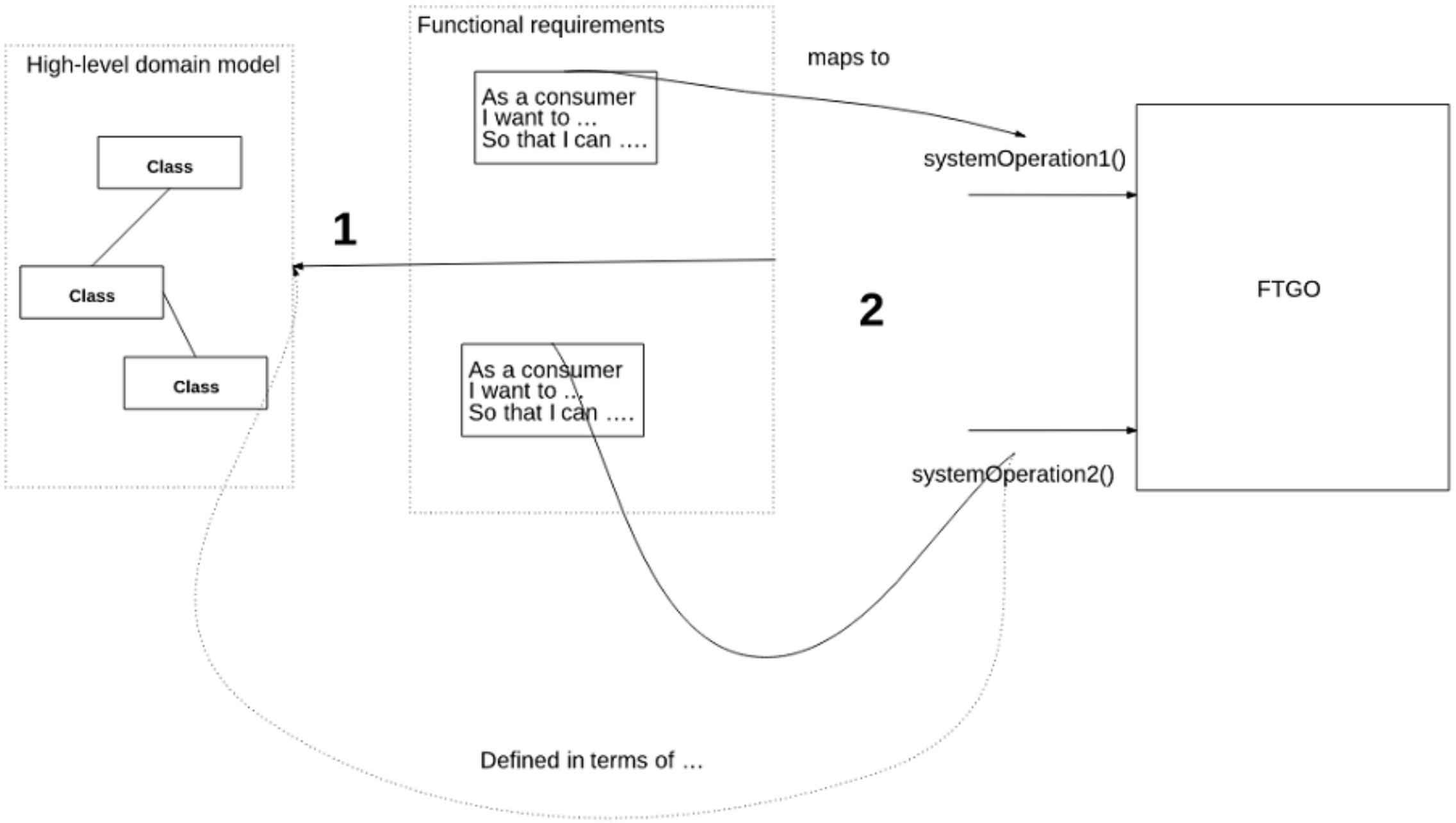
SERVIÇOS FRACAMENTE ACOPLADOS

- Numa arquitetura de microsserviços, o serviços são fracamente acoplados
- Toda interação com um serviço é feita via API, que encapsula os detalhes de implementação
- Isto permite a execução de mudanças no serviço sem impactar seus clientes
- Serviços fracamente acoplados são a chave para melhorar atributos que impactam no tempo de desenvolvimento, como manutenibilidade e testabilidade.
- Serviços fracamente acoplados e pequenos são mais fáceis de entender, mudar e testar

IDENTIFICANDO OPERAÇÕES DO SISTEMA

- O primeiro passo da definição da arquitetura da aplicação é definir as operações do sistema, através da interpretação dos requisitos da aplicação
- As operações do sistema são identificadas e definidas usando um processo em duas etapas, inspirado pelo processo de design orientado a objetos descrito por Craig Larman's no livro "Applying UML and Patterns"
- O primeiro passo é criar um modelo do domínio consistindo das classes principais, o que irá prover um vocabulário com o que será possível descrever as operações do sistema.
- O segundo passo é identificar as operações do sistema e descrever o comportamento de cada um em termos de modelo de domínio.

IDENTIFICANDO OPERAÇÕES DO SISTEMA



CRIANDO UM MODELO DE DOMÍNIO DE ALTO NÍVEL

Pedido Solicitado

Dado um cliente

E um restaurante

Quando o cliente solicita um pedido para o restaurante
com um endereço de entrega que poderá ser servidor por tal restaurante

Então o cartão de crédito do cliente é autorizado

E um pedido é criado no estado “ACEITAÇÃO PENDENTE”

E o pedido é associado ao cliente

E o pedido é associado ao restaurante

Pedido Aceito

Dado um pedido ou um restaurante que está esperando ser aceito

E um correio que está disponível para entregar o pedido

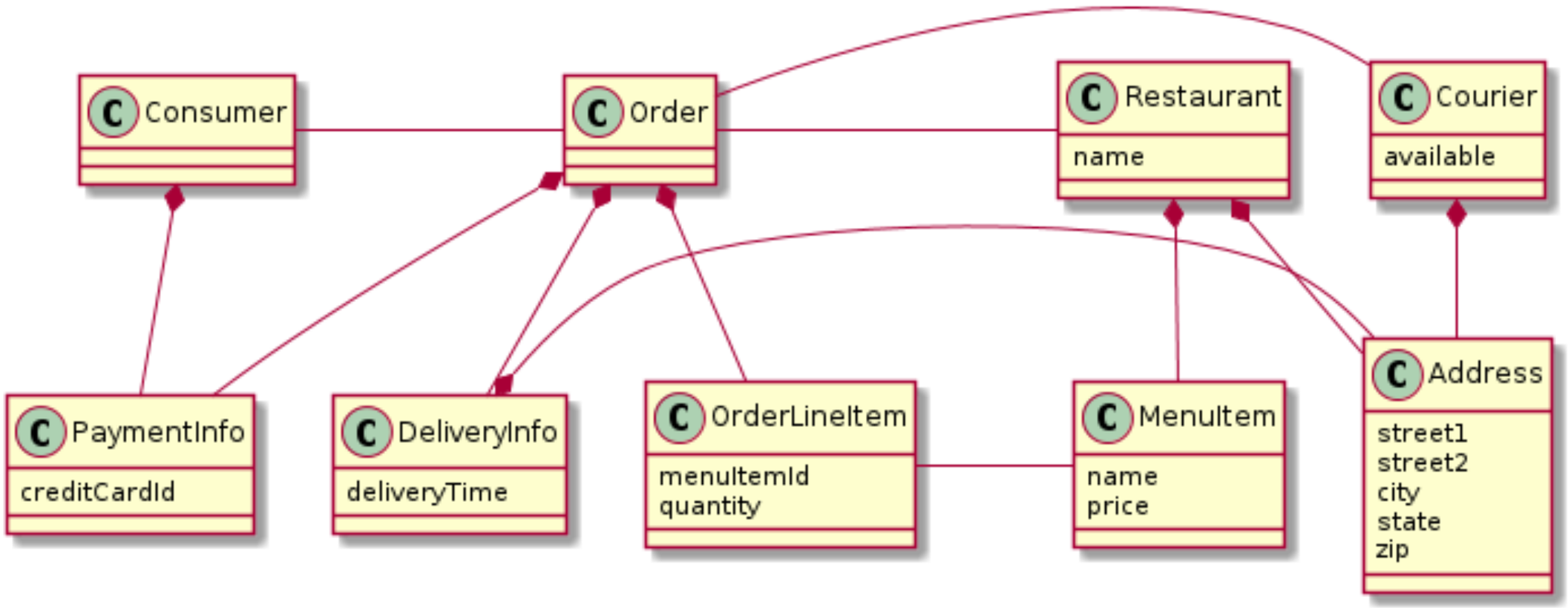
Quando um restaurante aceita um pedido com a promessa de prepará-lo com um tempo estimado de entrega

Então o estado do **pedido é mudado** para “ACEITO"

E o **tempo estimado de entrega** do **pedido** é atualizado

E o correio é designado para entregar o pedido

CRIANDO UM MODELO DE DOMÍNIO DE ALTO NÍVEL

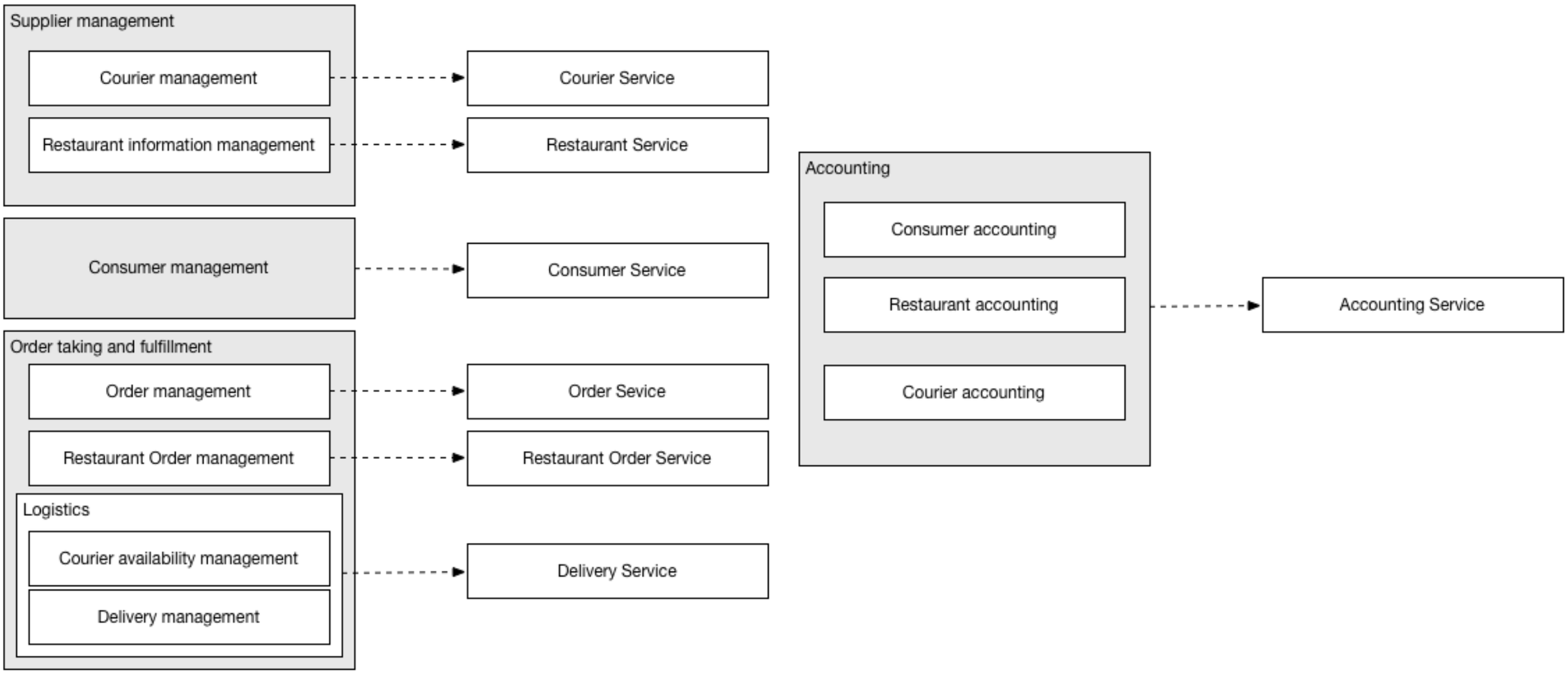


DEFININDO AS OPERAÇÕES DO SISTEMA

Actor	Story	Command	Description
Consumer	Create Order	createOrder()	creates an order
Restaurant	Accept Order	acceptOrder()	indicates that the restaurant has accepted the order and is committed to preparing it by the indicated time
Restaurant	Order Ready for Pickup	noteOrderReadyForPickup()	indicates that the order is ready for pickup
Courier	Update Location	noteUpdatedLocation()	updates the current location of the courier
Courier	Order picked up	noteOrderPickedUp()	indicates that the courier has picked up the order
Courier	Order delivered	noteOrderDelivered()	indicates that the courier has delivered the order

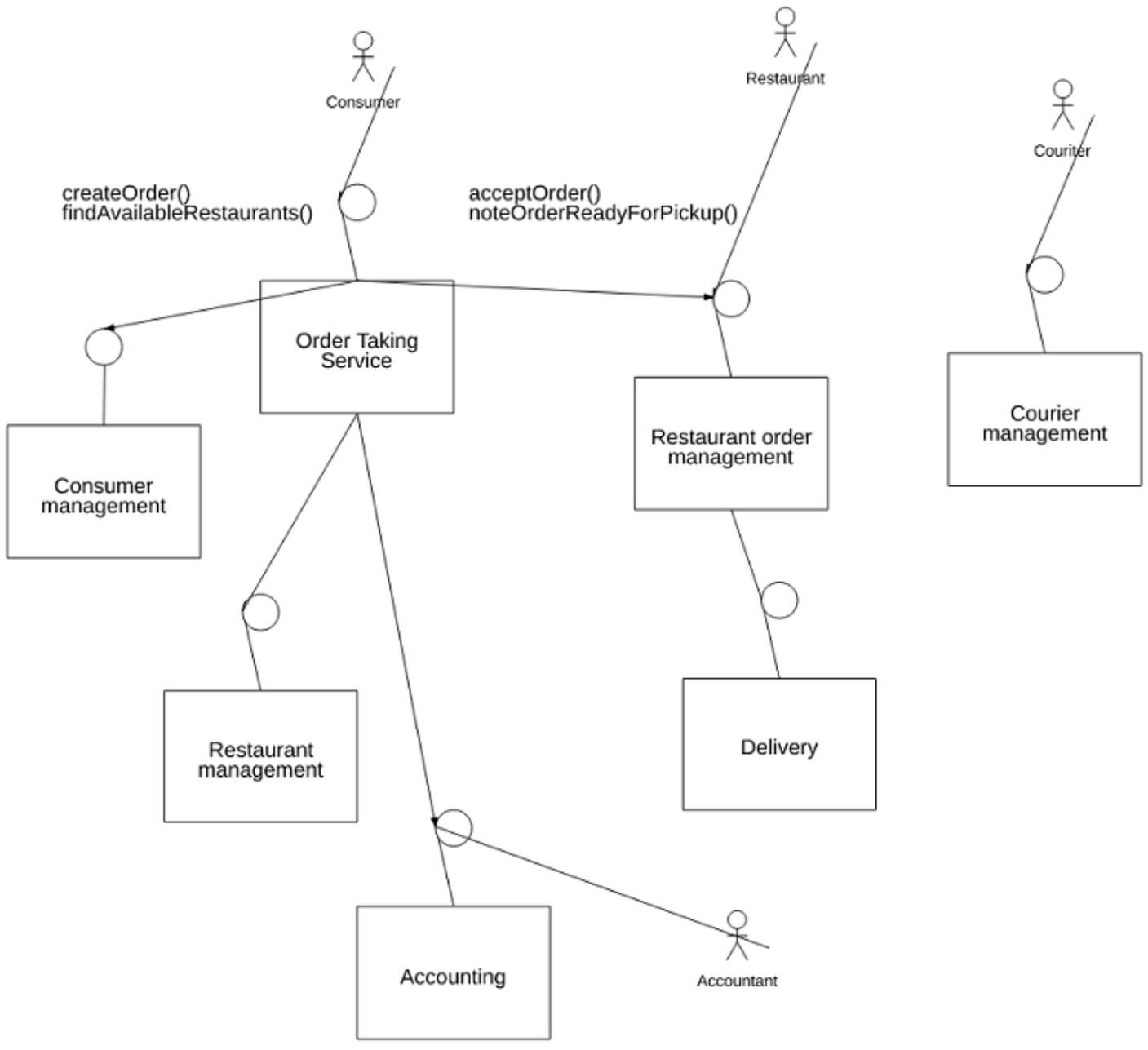
ESTRATÉGIAS PARA DECOMPOR UMA APLICAÇÃO EM SERVIÇOS

- 1. Decompor por funcionalidades de negócio



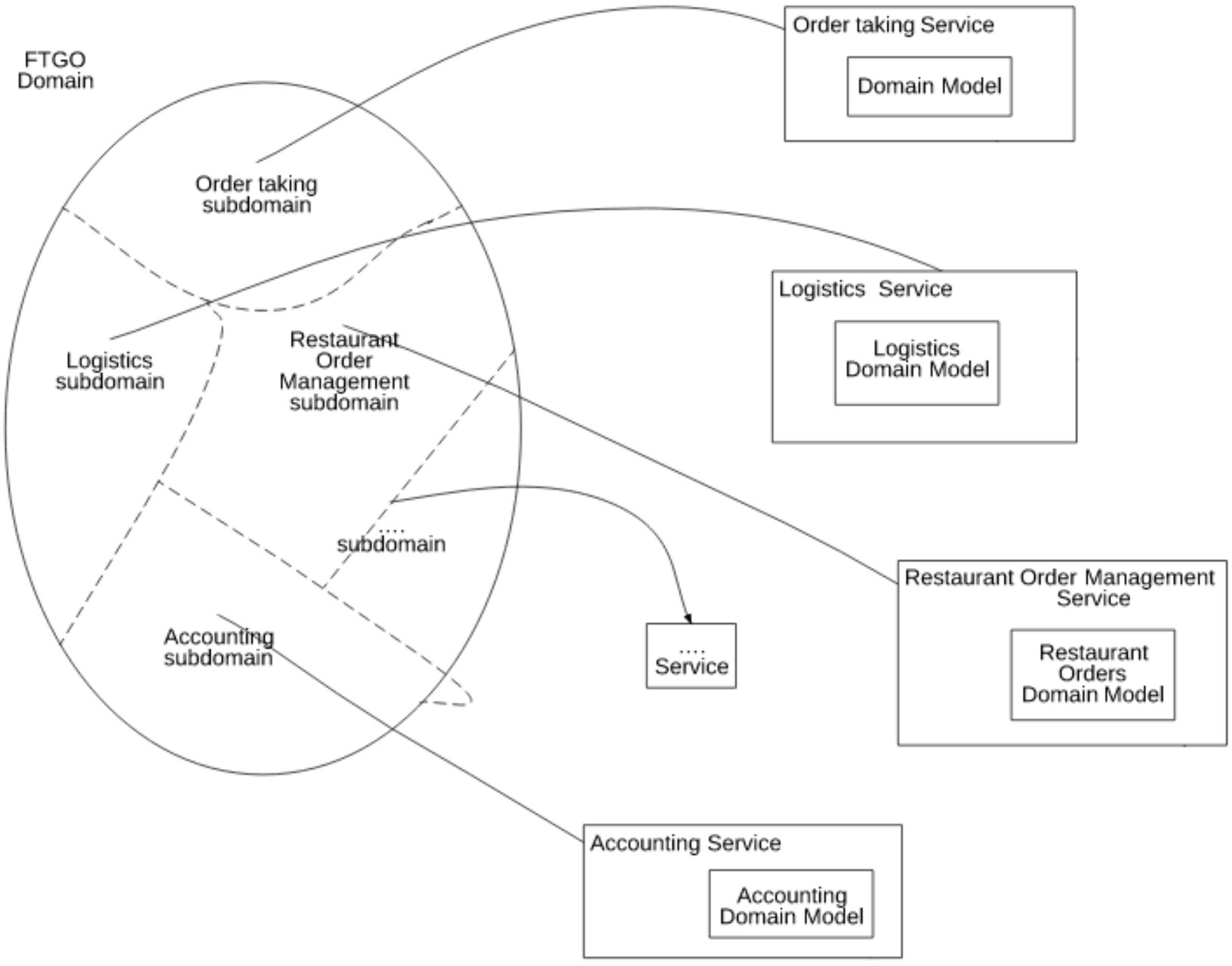
ESTRATÉGIAS PARA DECOMPOR UMA APLICAÇÃO EM SERVIÇOS

- 2. Usando cenários para definir como os serviços se comunicam

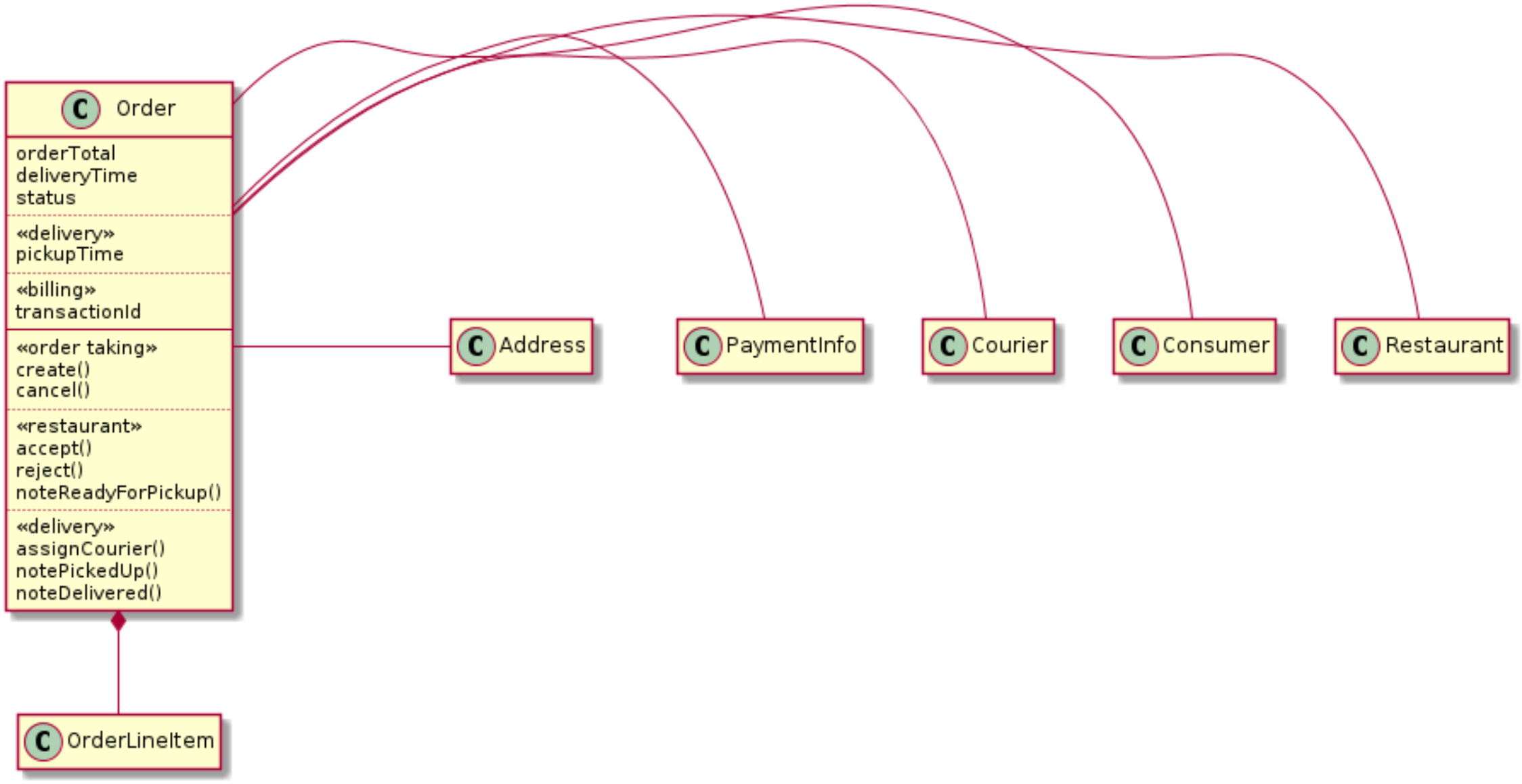


ESTRATÉGIAS PARA DECOMPOR UMA APLICAÇÃO EM SERVIÇOS

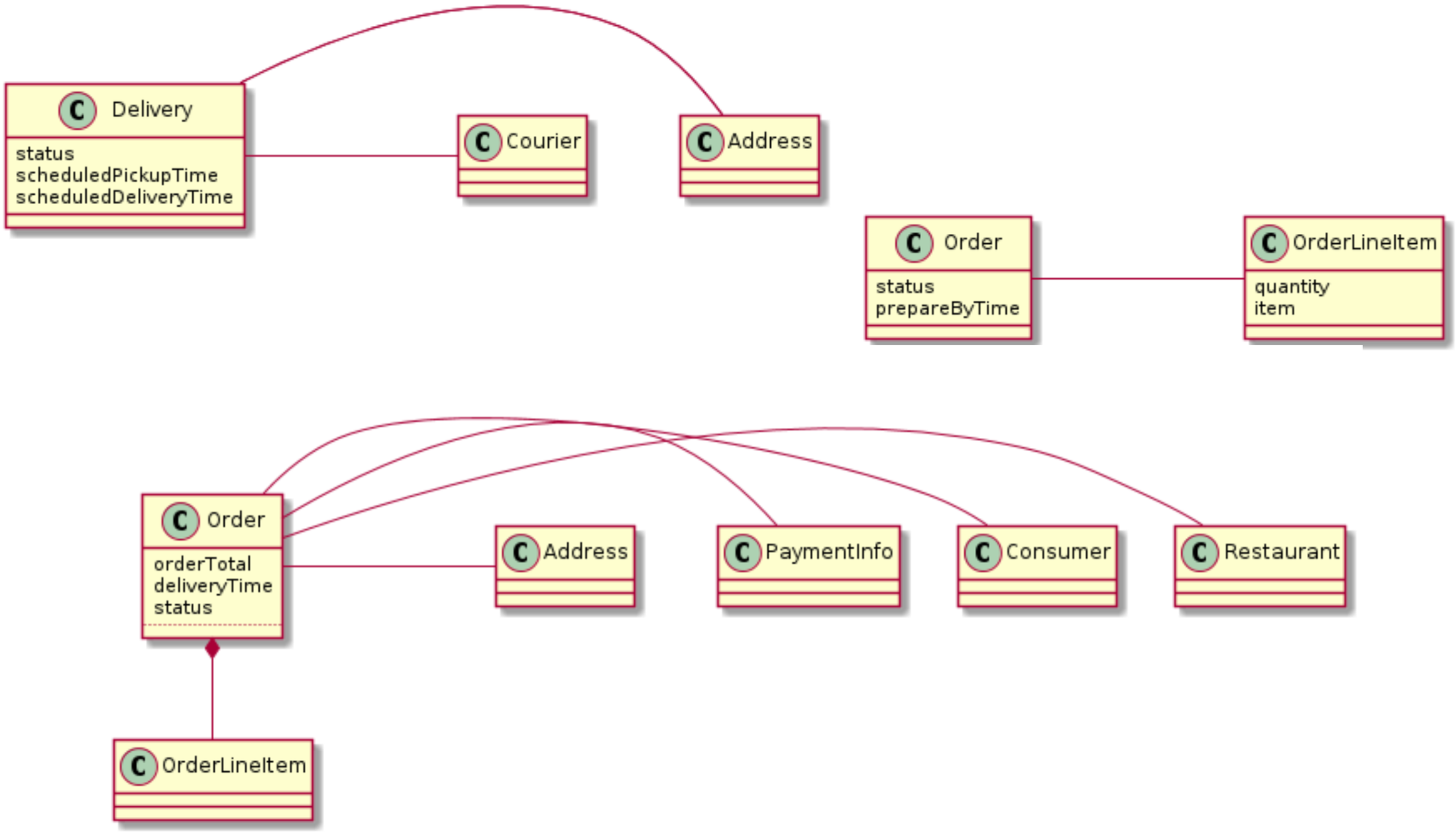
- 3. Decompor por subdomínio/bounded context



ELIMINANDO “GOD" CLASSES



MODELOS DE SUBDOMÍNIO



GUIDELINES PARA DECOMPOSIÇÃO

- Single Responsibility Principle
- Common Closure Principle

SINGLE RESPONSIBILITY PRINCIPLE

"A CLASS SHOULD HAVE ONLY ONE REASON TO CHANGE."

ROBERT C. MARTIN

COMMON CLOSURE PRINCIPLE

THE CLASSES IN A PACKAGE SHOULD BE CLOSED TOGETHER AGAINST THE SAME KINDS OF CHANGES. A CHANGE THAT AFFECTS A PACKAGE AFFECTS ALL THE CLASSES IN THAT PACKAGE.

ROBERT C. MARTIN

HOMEWORK - DECOMPOSIÇÃO EM MICROSSERVIÇOS