INSTITUTO FEDERAL
Paraíba
Campus Cajazeiras

PROGRAMAÇÃO P/ WEB 2

3

9. PADRÕES PARA CONSUMO DE APIS EXTERNAS

PROF. DIEGO PESSOA

DIEGO.PESSOA@IFPB.EDU.BR

@DIEGOEP

CST em Análise e Desenvolvimento de Sistemas

ROTEIRO

- O desafio de projetar uma API que suporta diversos conjuntos de clientes
- Aplicando os padrões API gateway e Backends para front-ends
- Projetando e modelando um API gateway
- Usando programação reativa para simplificar a composição de APIs

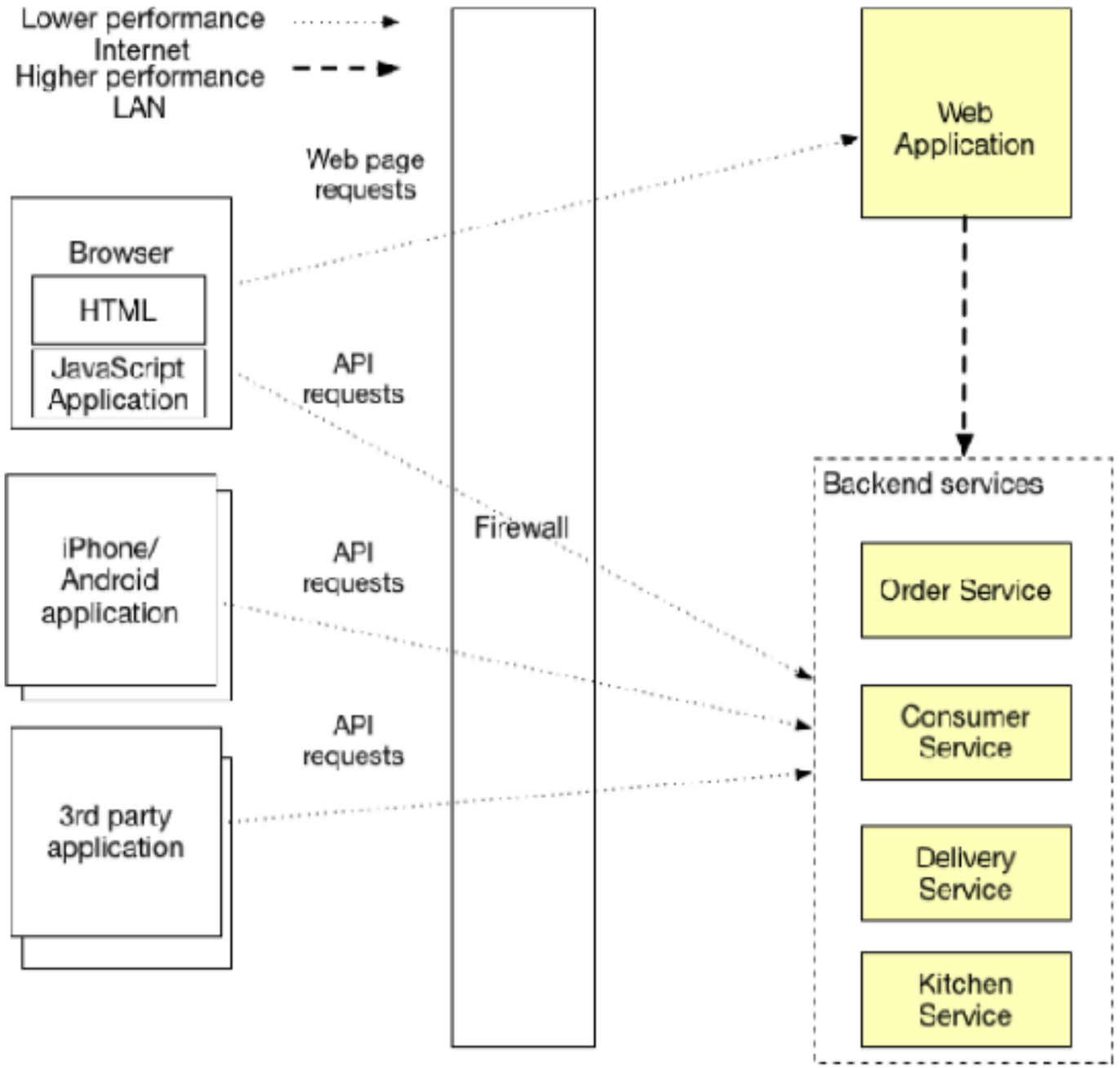
INTRODUÇÃO

- Aplicações com APIs REST podem ser consumidas por vários clientes, tais como aplicações móveis e javascript rodando no browser.
- Numa arquitetura monolítica, a API exposta aos clientes é simplesmente a API da aplicação.
- No caso de microsserviços, o modelo de negócio utiliza várias APIs de serviços distintos.
- Fazer o cliente conhecer todos os serviços não seria eficiente…

1. DESAFIOS PARA CONSUMO POR APIS EXTERNAS

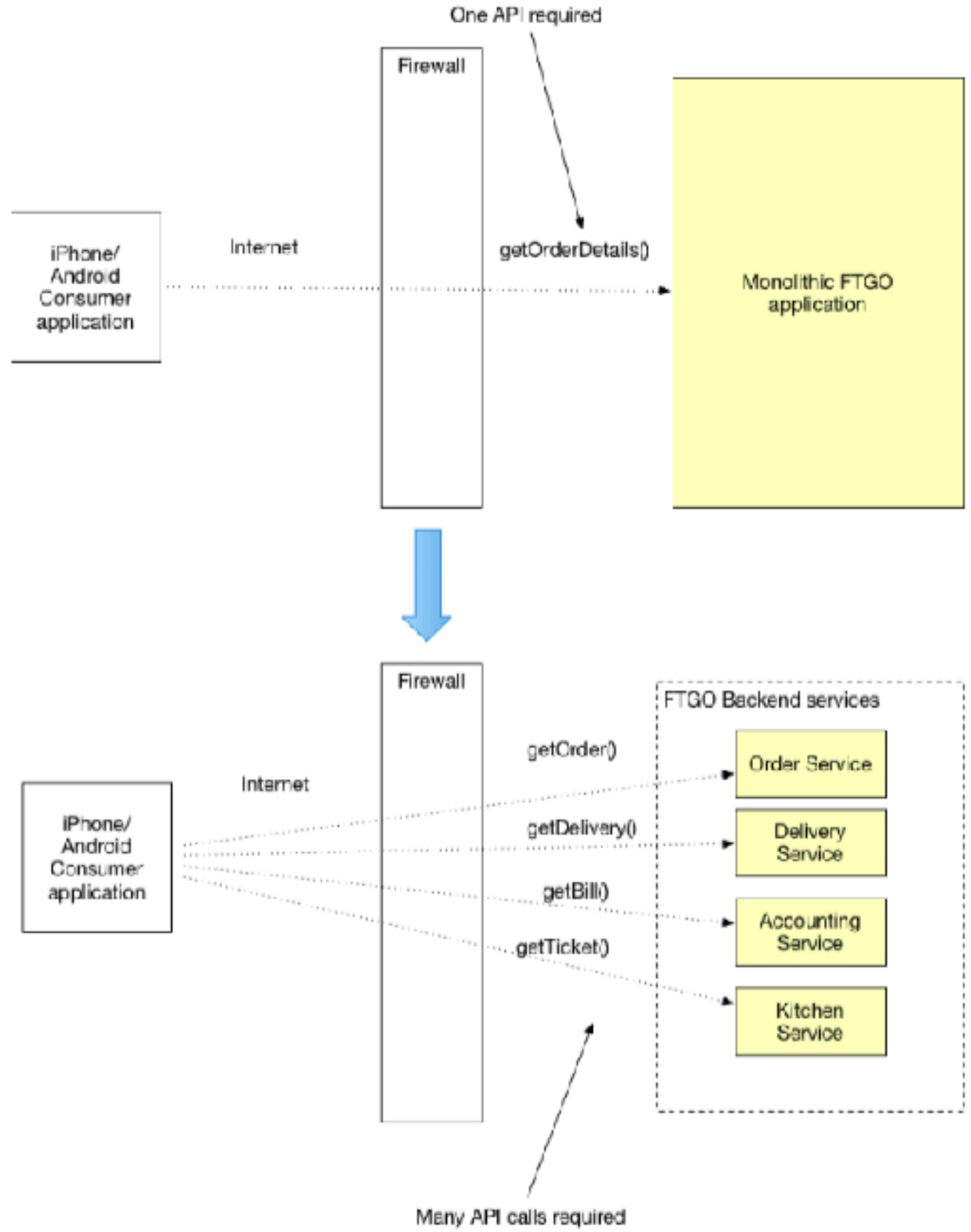
DESAFIOS DE APIS EXTERNAS – APLICAÇÃO EXEMPLO

- 4 tipos de clientes consomem as APIs dos serviços:
- 1. Aplicações Web (backend)
- 2. Aplicações Javascript rodando no browser
- 3. Aplicações móveis
- 4. Aplicações escritas por desenvolvedores terceiros



DESAFIOS DE APIS EXTERNAS – PARA CLIENTES MÓVEIS

- Clientes fazendo múltiplas requisições
- Acoplamento entre frontend e backend
- Uso de protocolos além de HTTP

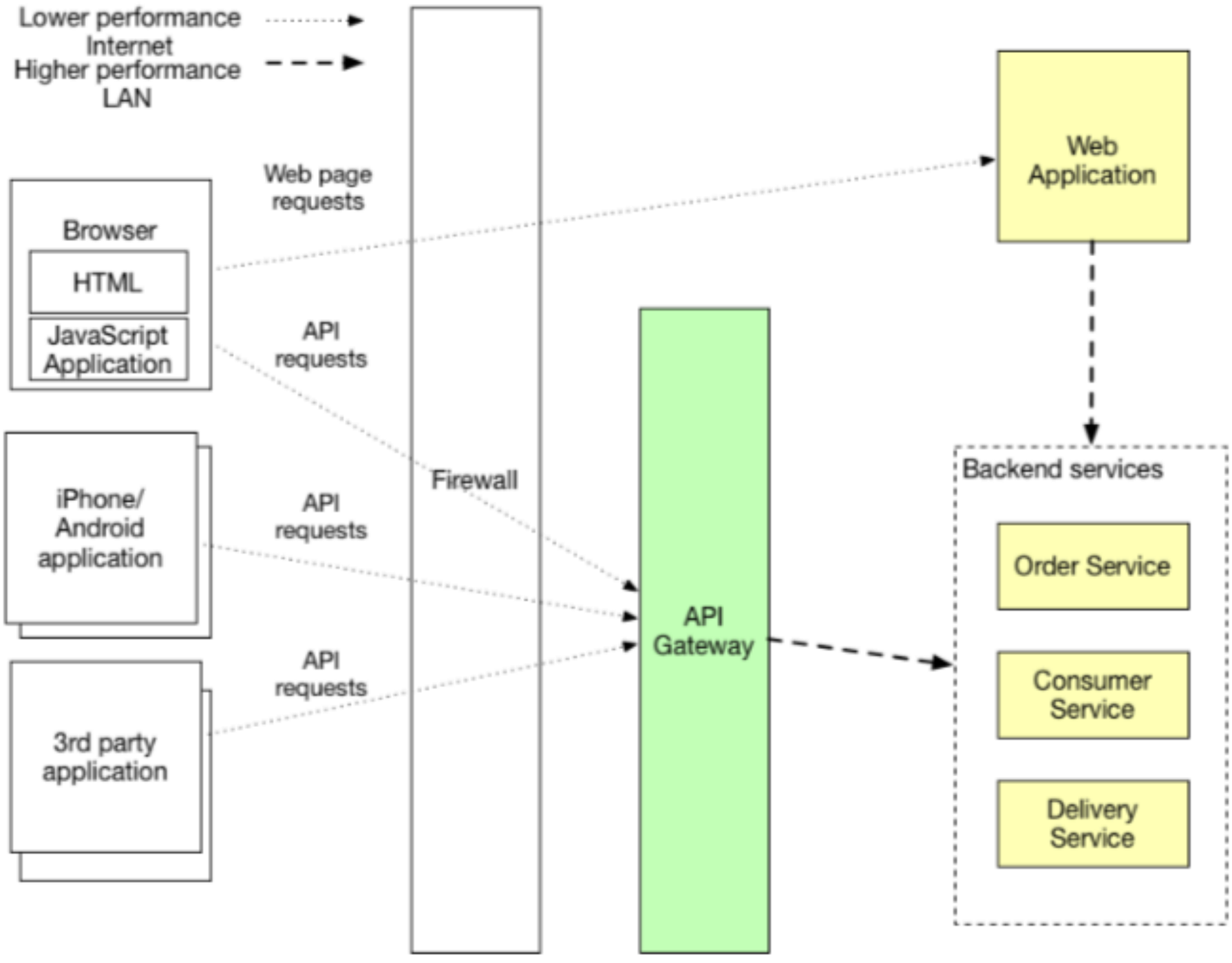


DESAFIOS DE APIS EXTERNAS PARA OUTROS CLIENTES

- Maior latência (uso da rede entre navegador e serviços ao invés de chamadas internas)
- Acoplamento entre aplicação web (javascript) e APIs dos serviços, dificultando a manutenção
- Dificuldade em expor a API da aplicação para terceiros

2. O PADRÃO API GATEWAY

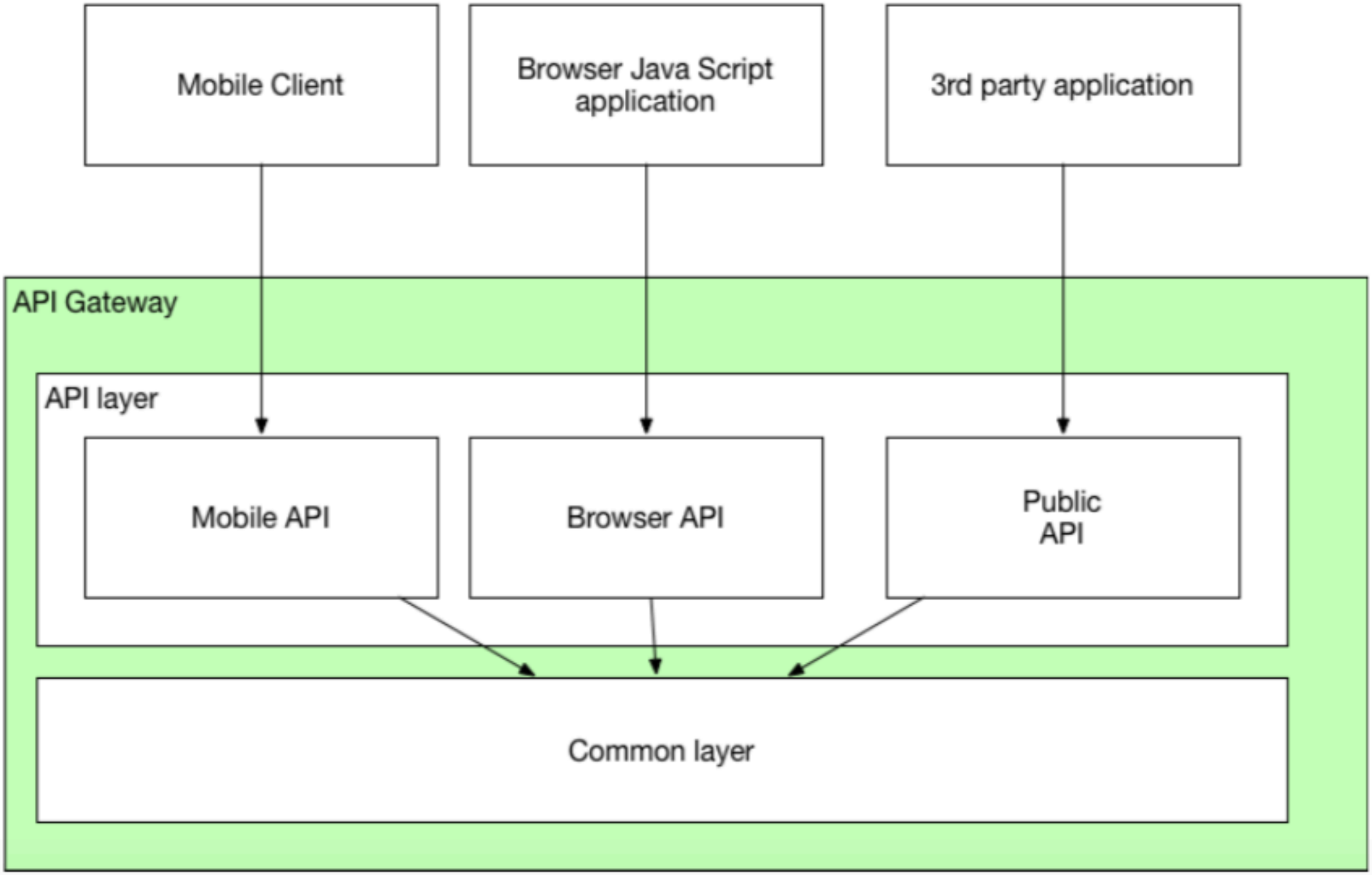
VISÃO GERAL


Lower performance Internet
Higher performance LAN
Web page requests
API requests
API requests
API requests
Browser
HTML
JavaScript Application
iPhone/ Android application
3rd party application
Firewall
API Gateway
Web Application
Backend services
Order Service
Consumer Service
Delivery Service

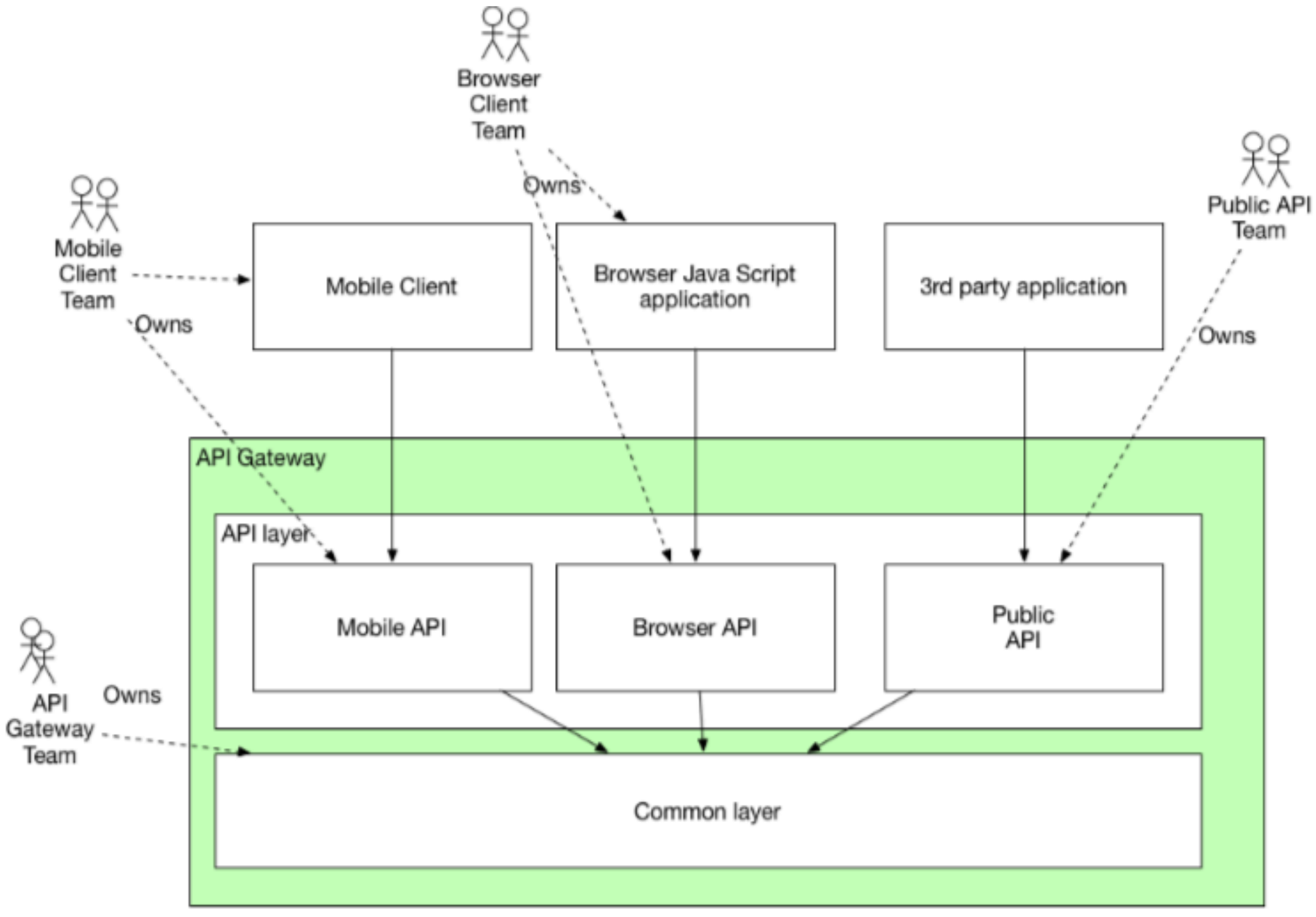
FUNÇÕES DO API GATEWAY

- Roteamento de requisição
- API composition
- Tradução de protocolo (ex.: API REST para gRPC)
- Prover uma API específica para cada cliente
- Implementar funções de borda (autenticação, autorização, limite de requisições, caching, coleta de métricas e logging)

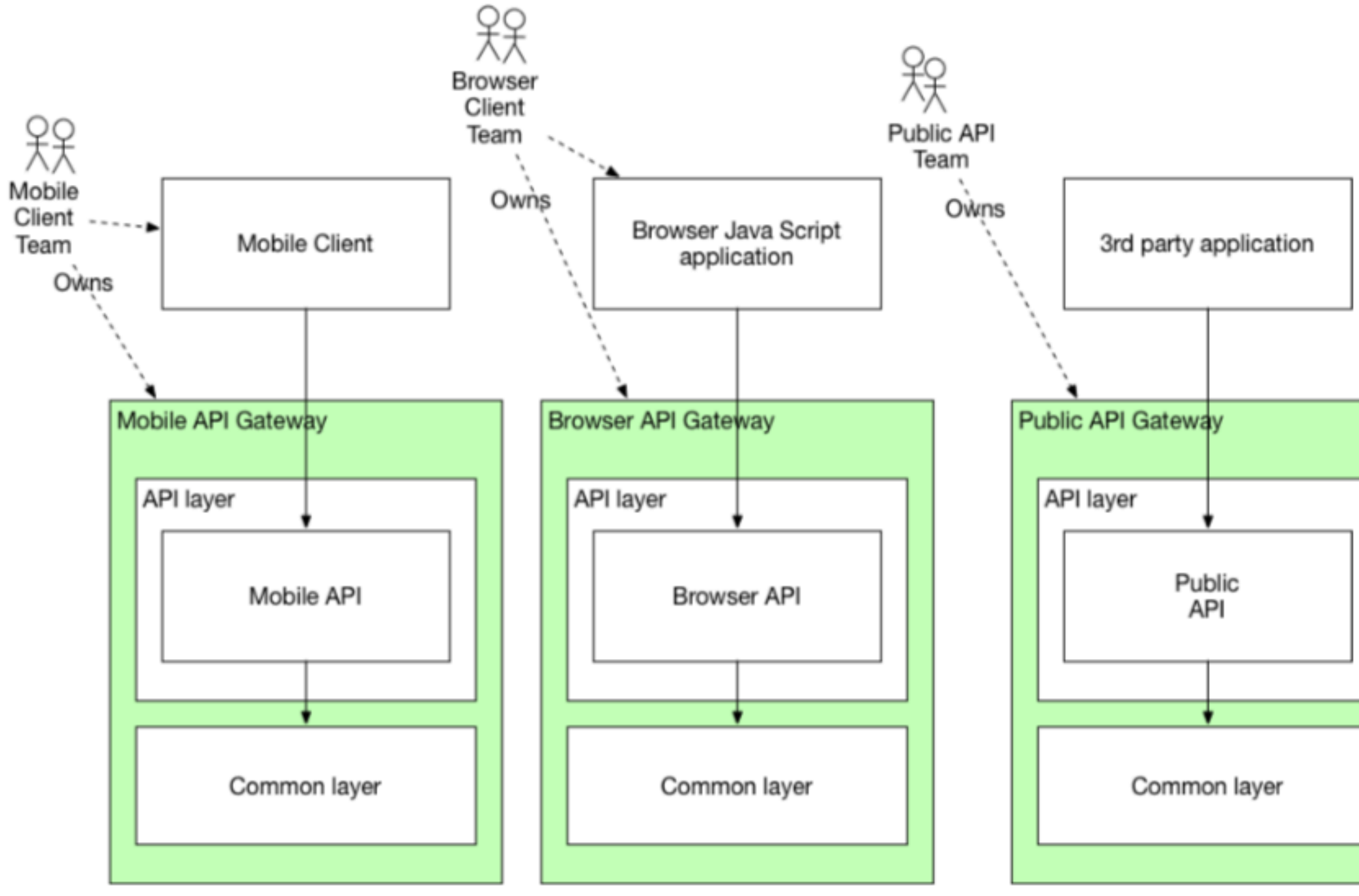
API GATEWAY – ARQUITETURA



API GATEWAY – MODELO DE PROPRIEDADE



USANDO O PADRÃO "BACKENDS PARA FRONT ENDS"



API GATEWAY – BENEFÍCIOS E DESVANTAGENS

- Benefícios:
  - Encapsula estrutura interna da aplicação
  - Permite que a comunicação do cliente com a aplicação seja centralizada no Gateway
  - Provê a cada cliente uma API específica, que ajuda a entender melhor a API e simplificar o código do cliente

NETFLIX COMO EXEMPLO DE API GATEWAY

- Trata bilhões de requisições por dia advindas de vários dispositivos (TVs, bluerays, smartphones)
- 1ª implementação: um API Gateway com APIs separadas para cada dispositivo
- 2ª implementação: aplicação do padrão backends para front ends (um API gateway para cada dispositivo)
  - Utilização do Netflix Falcor para API composition dinâmica.

API GATEWAY – DESAFIOS

- Desempenho e escalabilidade
- Escrever código que seja fácil de manter utilizando abstrações de programação reativa
- Tratamento de falhas
- "Ser um bom cidadão" na arquitetura da aplicação

DESEMPENHO E ESCALABILIDADE

- API gateway é a porta da entrada da aplicação, concentrando todas as requisições externas.
- Ponto chave que afeta a performance e escalabilidade: uso de I/O síncrono ou assíncrono.
- Modelo síncrono (base da Java EE) possui limitações mesmo quando disponibiliza métodos assíncronos (limite de threads)
- Modelo assíncrono (Spring Reactor, NodeJS): mais escalável, sem threads múltiplas. Porém, mais complexo, mais difícil de entender e degubar.

USANDO ABSTRAÇÕES DE PROGRAMAÇÃO REATIVA

- Utilizar chamadas assíncronas paralelas ao invés de usar comunicação síncrona sequencial
- Tradicional: usar I/O baseada em eventos baseada em callbacks.
  - Desvantagem: callback hell
- Moderno: estilo declarativo usando abordagem reativa:
  - Java 8 CompletableFutures
  - Project Reactor Monos
  - RxJava (Reactive Extensions for Java) Observables - criado pela Netflix
  - Scala Futures

TRATAMENTO DE FALHAS

- API gateway deve ser confiável = rodar múltiplas instâncias por trás de um load balancer.
- Usar Circuit Breaker para limitar o tempo de espera de uma determinada chamada e ativar outras instâncias

"SENDO UM BOM CIDADÃO" NA ARQUITETURA

- Usar o padrão Service Discovery para permitir a localização de instâncias de serviços de maneira automática.
- Usar o padrão Observability para permitir o monitoramento do comportamento da aplicação e de possíveis problemas.
- O API gateway (como os outros serviços) devem implementar os padrões que foram selecionados para a arquitetura

3. IMPLEMENTANDO UM API GATEWAY

RESPONSABILIDADES DO API GATEWAY

- Roteamento de requisições - usando path HTTP.
- API composition
- Funções de borda (ex.: autenticação)
- Tradução de protocolo
- Se integrar facilmente à arquitetura da aplicação

OPÇÕES PARA IMPLEMENTAR UM API GATEWAY

- Usar um serviço/produto existente (ex.: AWS API Gateway)
  - Mais simples, porém, menos poderosa (não suporta API composition)
- Desenvolver o próprio API gateway usando algum framework como ponto de partida: abordagem mais flexível, porém, demanda esforço de desenvolvimento

DESENVOLVENDO O PRÓPRIO API GATEWAY

- 1. Implementar um mecanismo para definir regras de roteamento para minimizar a complexidade do código
- 2. Implementar o comportamento de proxy HTTP incluindo como tratar os headers

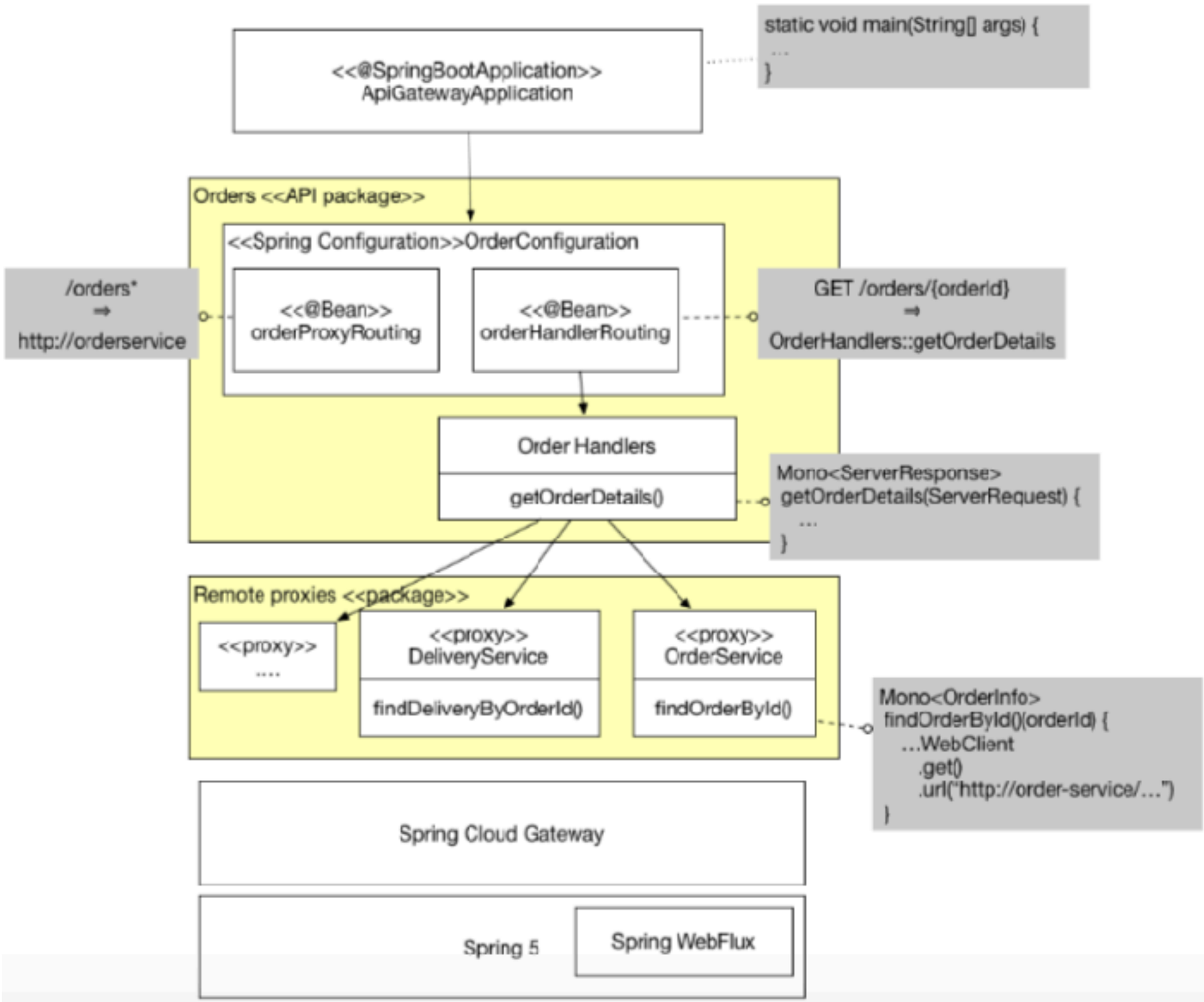
NETFLIX ZUUL

- Implementa funções de borda como roteamento, limitação de requisições e autenticação
- Possui o conceito de filtros para interceptar requisições
- Recomendação: Spring Cloud Zuul
- Limitações: roteamento path-based e arquitetura de consultas para CQRS

SPRING CLOUD GATEWAY

- Framework para API Gateway contendo:
  - Spring Framework 5
  - Spring Boot 2
  - Spring WebFlux (framework reativo do Spring)
  - Project Reactor (NIO-based framework que provê a abstração Mono)
- Disponibiliza:
  - Roteamento para requisições de serviços de backend
  - Implementa tratadores de requisições para API Composition
  - Trata funções de borda como autenticação

ARQUITETURA DE UM API GATEWAY USANDO SPRING CLOUD GATEWAY



API GATEWAY – EXEMPLO

- Exemplo:
- https://github.com/microservice-patterns/ftgo-application/tree/master/ftgo-api-gateway