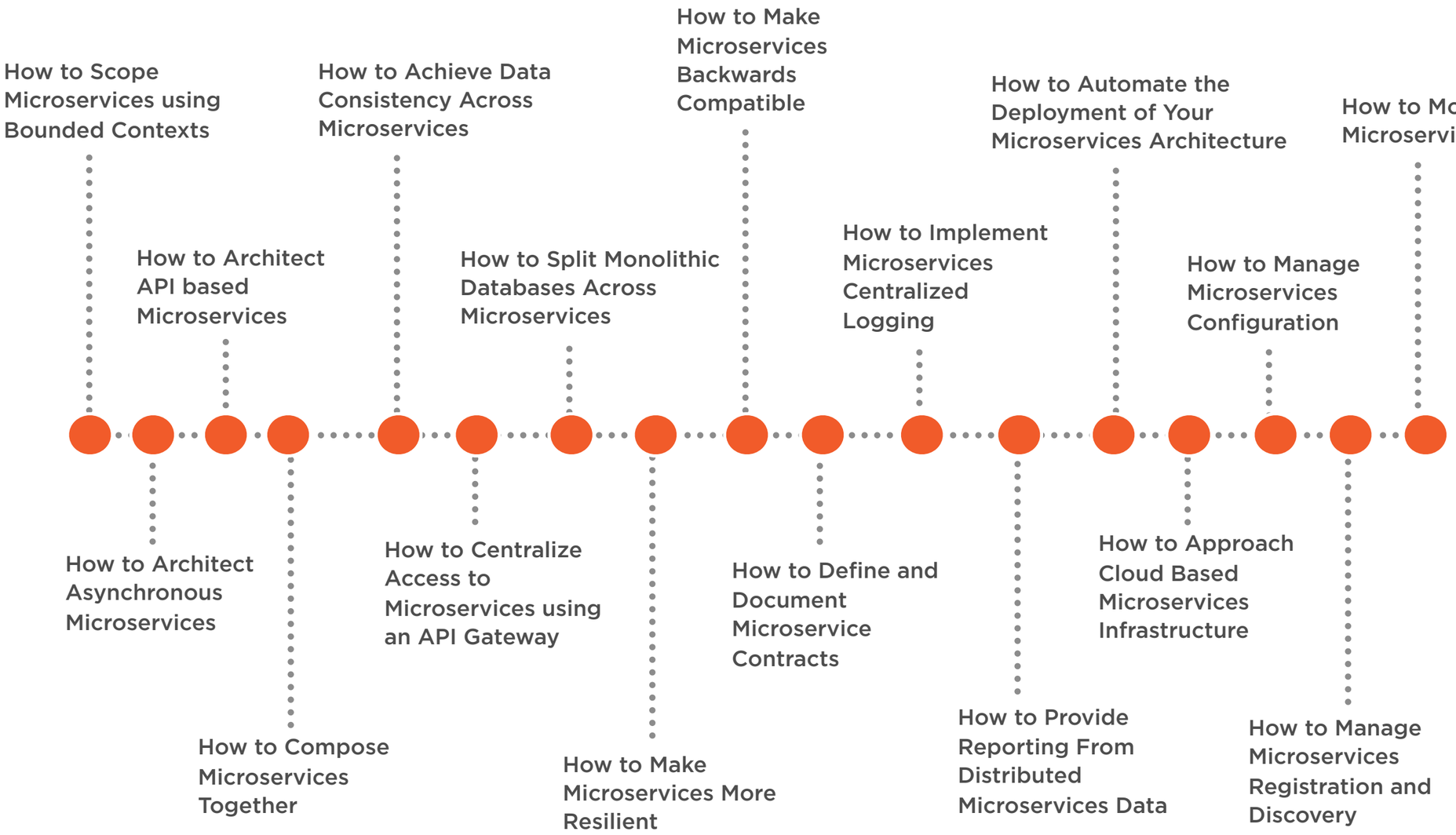
How to Architect API Based Microservices

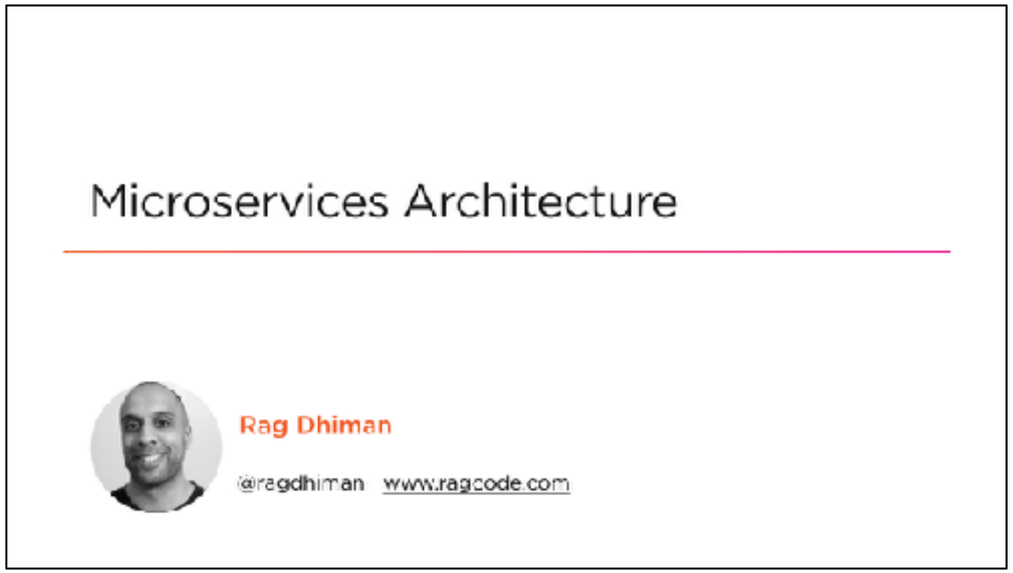
Rag Dhiman

@ragdhiman www.ragcode.com

Microservices Architectural Design Patterns Playbook



Microservices Architectural Design Patterns Playbook



Overview

Introduction

Functional Requirements

Architecture Options

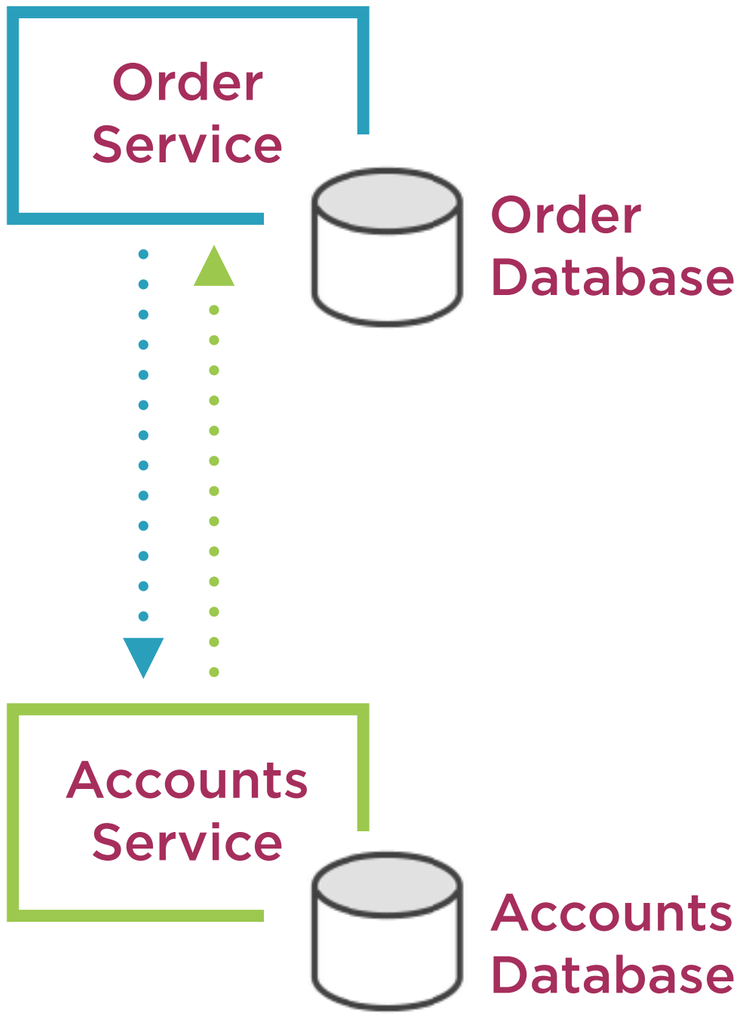
REST Architectural Style

API Architectural Patterns

- Facade Design Pattern
- Proxy Design Pattern
- Stateless Service Pattern

Introduction

Introduction



Microservices

- API vs worker based

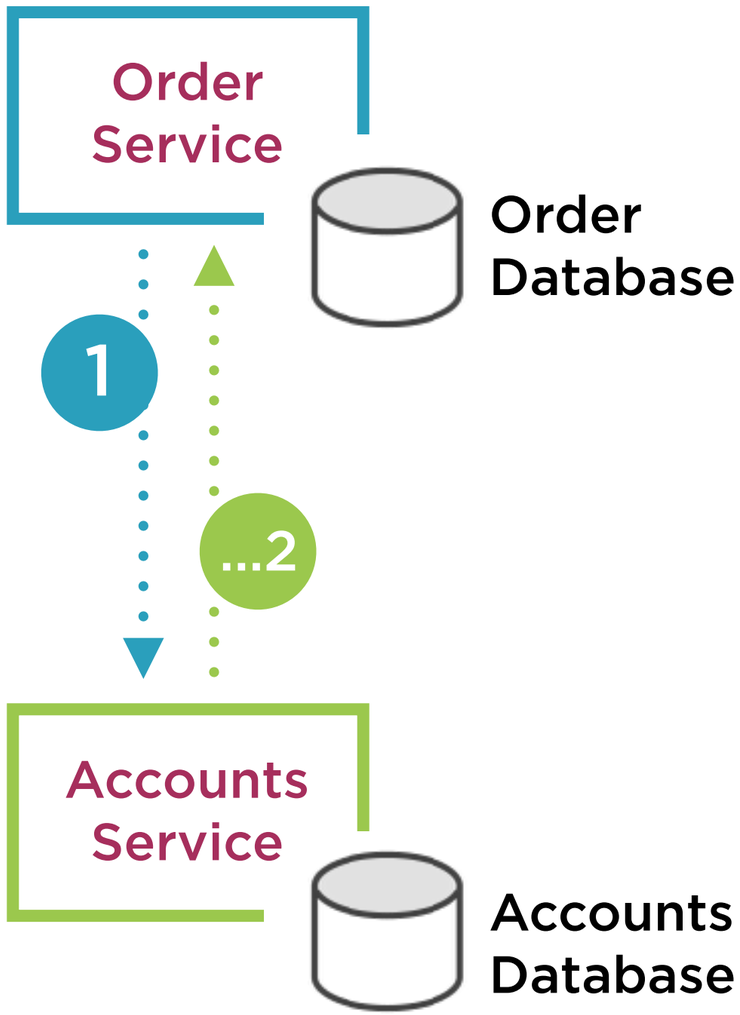
Architecture

- API vs application

How to architecture

- Functional requirements
- Architecture styles
- Architecture patterns

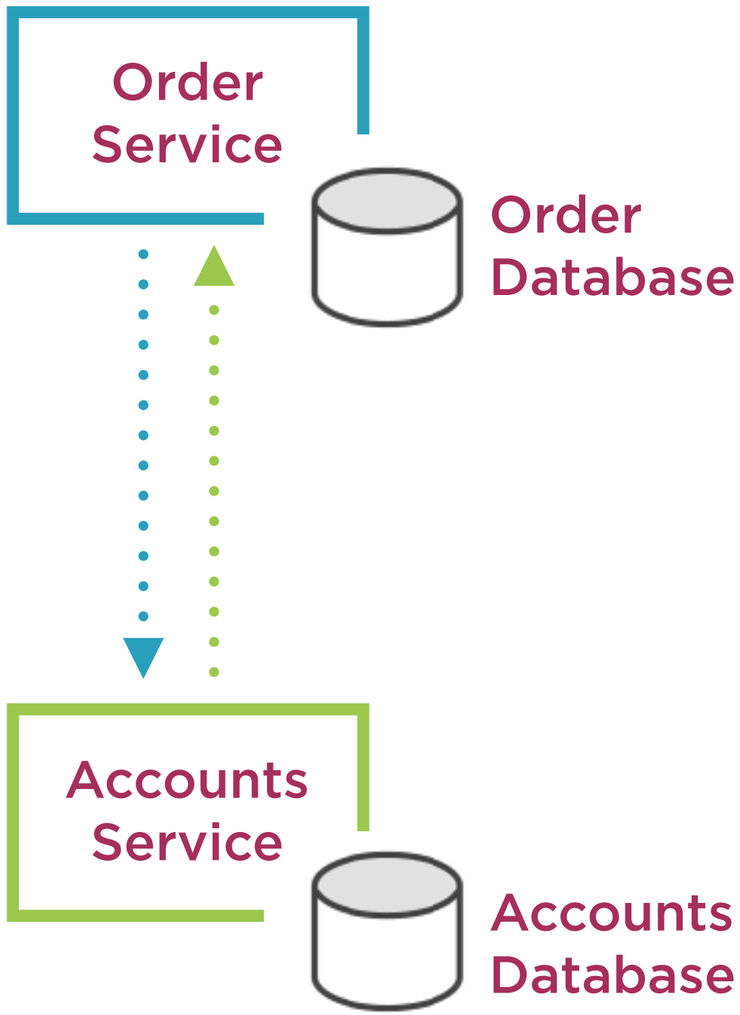
Functional Requirements



Autonomous microservices principle

- Loosely coupled
- Independently changeable
- Independently deployable
- Support and honor contracts
- Technology agnostic API
- Stateless API

Architecture Options



API architectural styles

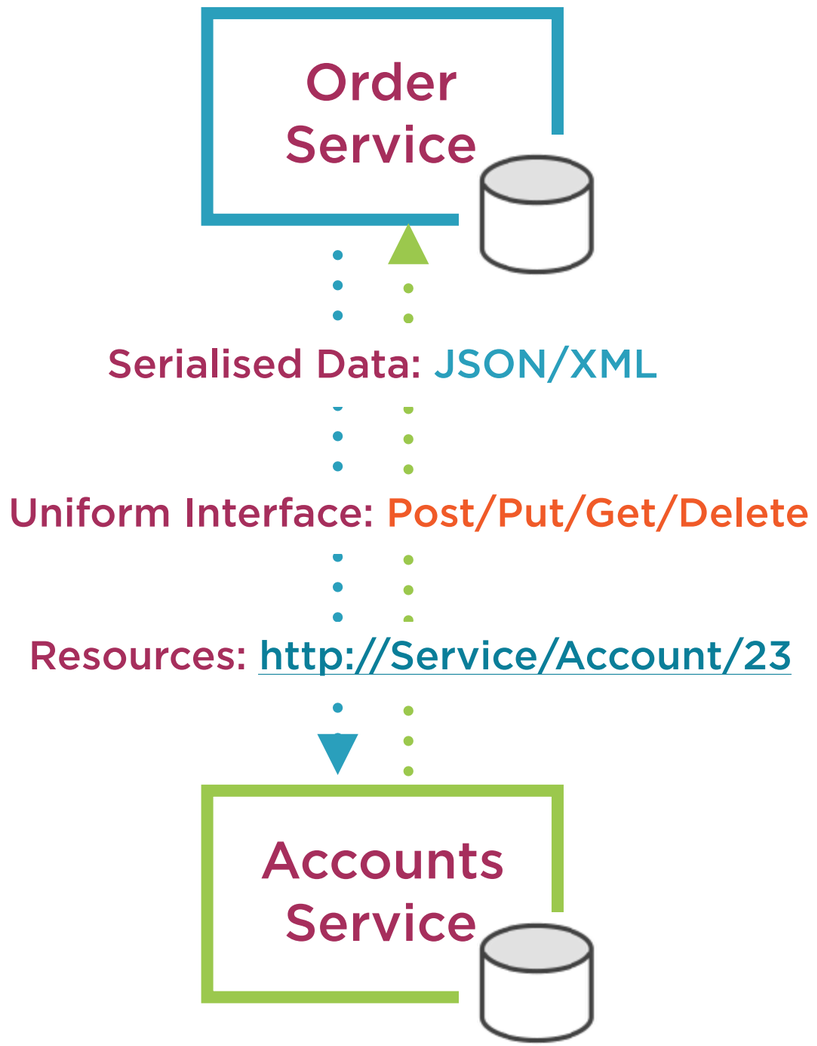
- Pragmatic REST
- HATEOS (True REST)
- RPC
- SOAP

API architectural patterns

- Facade pattern
- Proxy pattern
- Stateless service pattern

REST Architectural Style

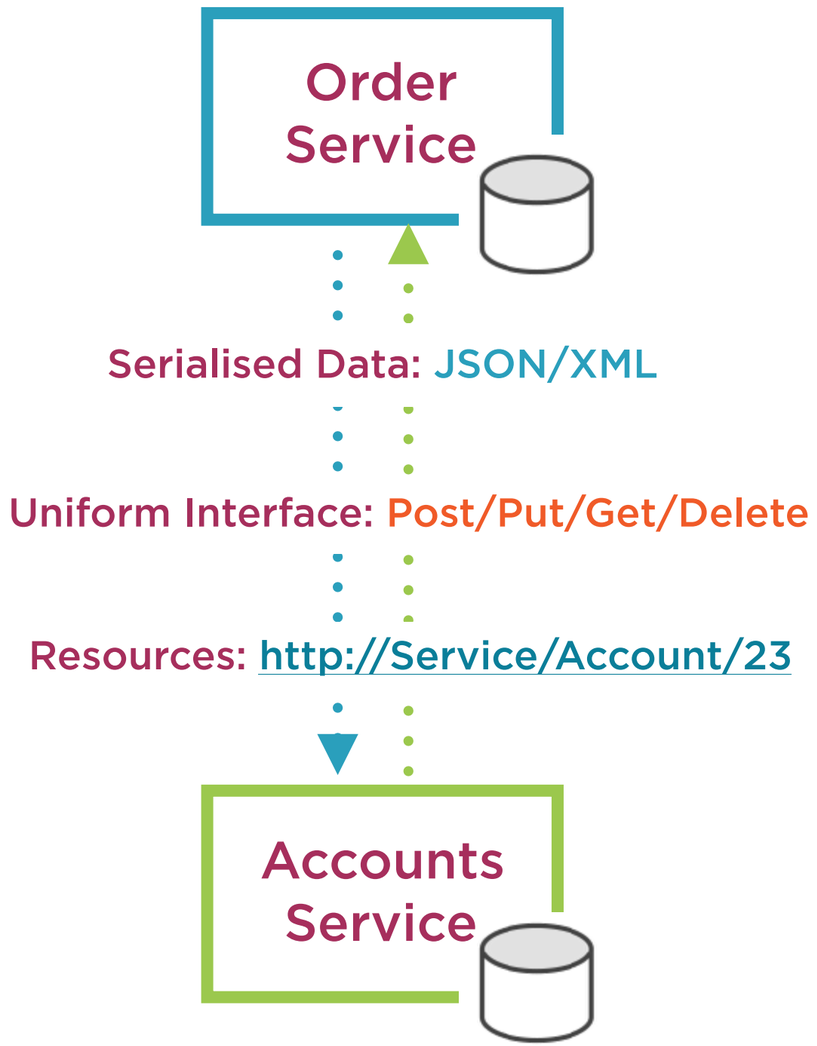
REST Architectural Style



What is REST?

- A style that defines constraints
- Uses HTTP based infrastructure
- Inherits advantages of the web
- Key concepts
  - JSON or XML
  - Resource endpoints
  - Uniform resource interface

Applying REST Constraints to Microservices



Client-server

Stateless

Cacheable

Layered system

Uniform Interface

- Resources
- Manipulation of resources from request
- Self-descriptive message

RESTful API Example

API	Description	Request body	Response body	HTTP Response Code
GET /api/customer	Get all customers	None	Array of customers	200/OK
GET /api/customer/{id}	Get an customer by ID	None	Customer	200/OK
POST /api/customer	Add a new customer	Customer	Customer	201/Created
PUT /api/customer/{id}	Update an existing customer	Customer	None	200/OK
DELETE /api/customer/{id}	Delete a customer	None	None	204/No Content

Pragmatic RESTful Microservices

Actions and Tasks

/api/customer/calculateInvoice/2017

/api/routes/optimizeRoutes/

/api/Offers/expireVouchers/

/api/customer/emailVoucher/

/api/reports/consolidate/

/api/Offers/expireVouchers/453

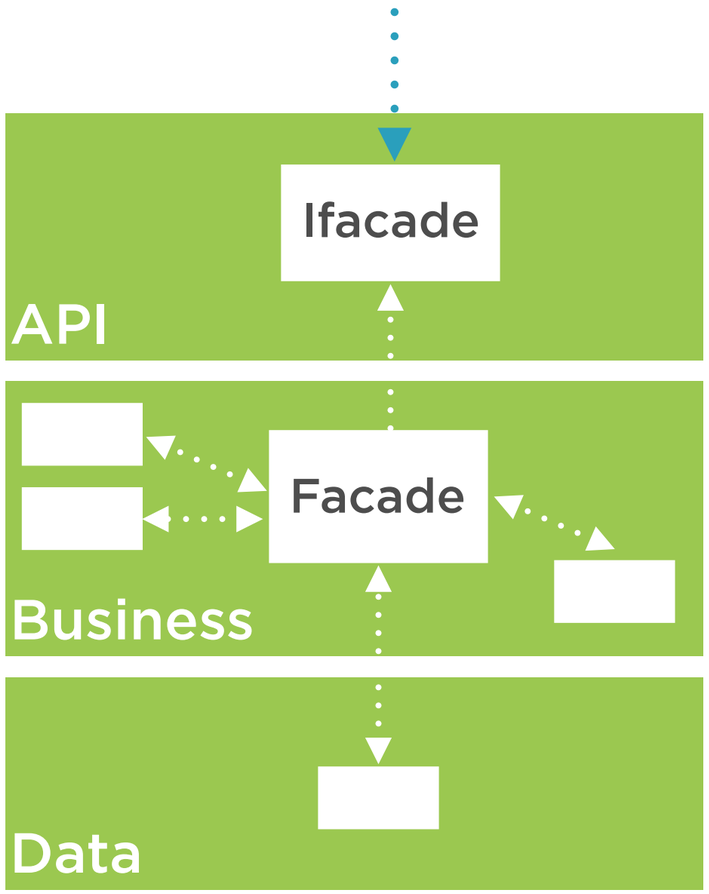
Extended unofficial version of REST

Pragmatic additions for microservices

- Not all about resource CRUD
- Action\Task based endpoints
- Verbs instead of nouns
- Query string and or request body
- Response body for output
- Callback address for output
- HTTP status codes

API Architectural Patterns

Facade Design Pattern



A single interface to subsystems

Each subsystem represented by a class

API requests honored by the facade

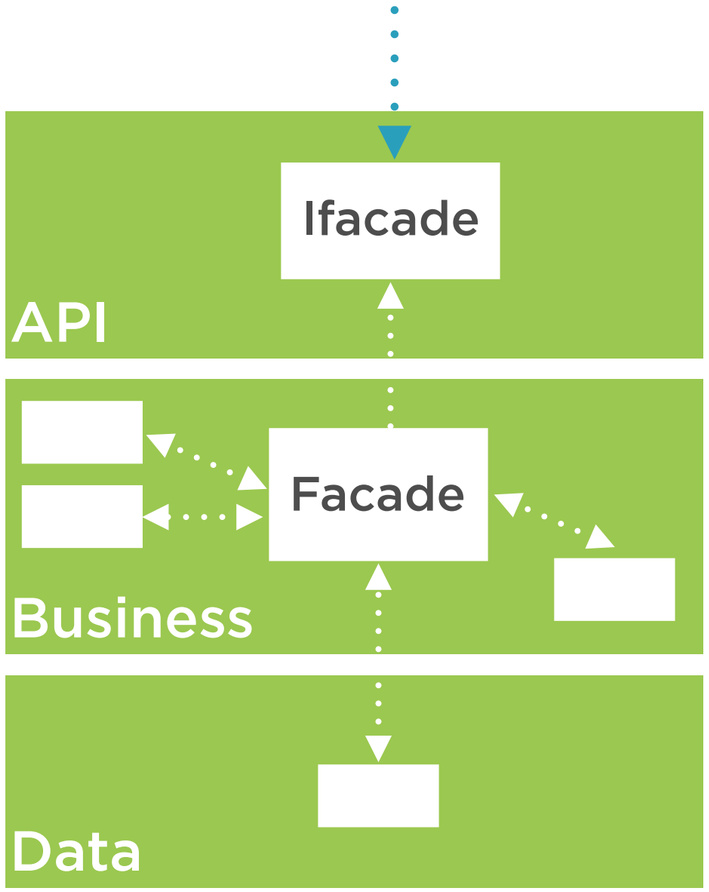
Microservice API is just a facade

- Business logic lives behind the facade
- Helps with information hiding
  - Shared models
  - Decoupling

Facade helps enforce design principles

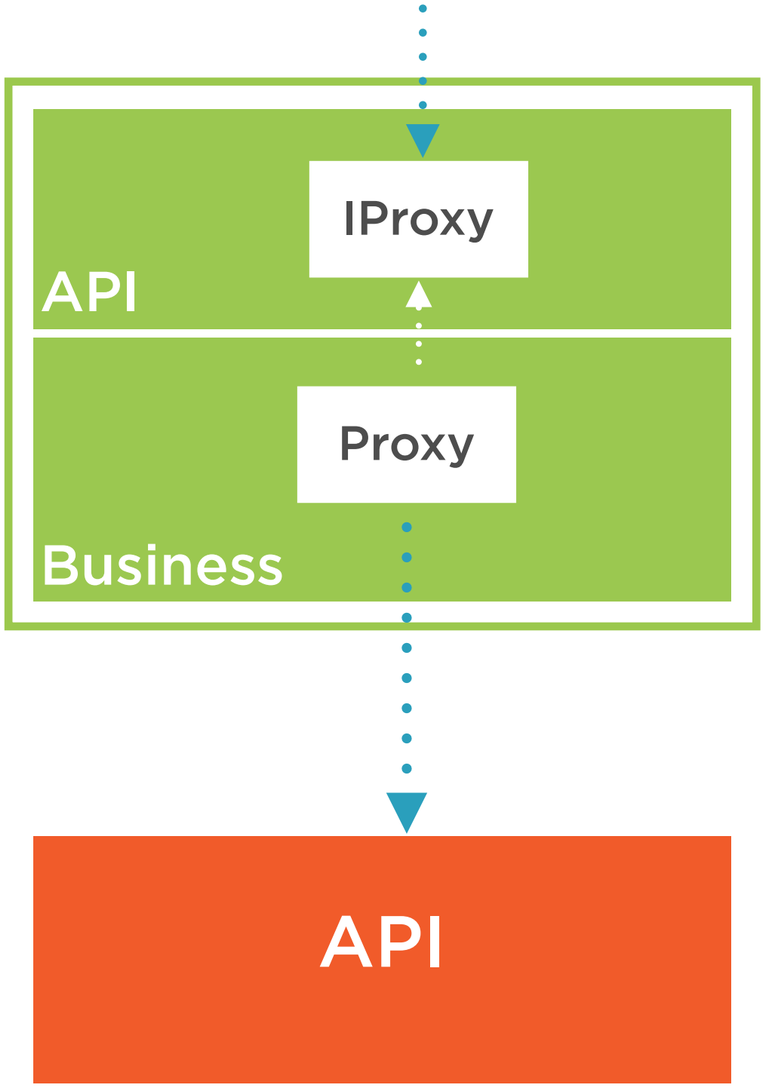
- Single cohesion

Approach to the API Facade Design Pattern



1. **Design the API**
   - Endpoints
   - Parameters
   - Responses
   - Status codes
2. **Mock the API for consumer testing**
   - Facade that returns hardcoded
     - Responses
     - Statuses
3. **Revise and implement the actual facade**
   - Using feedback from 2

Proxy Design Pattern



Placeholder for another object

Used to control access to other object

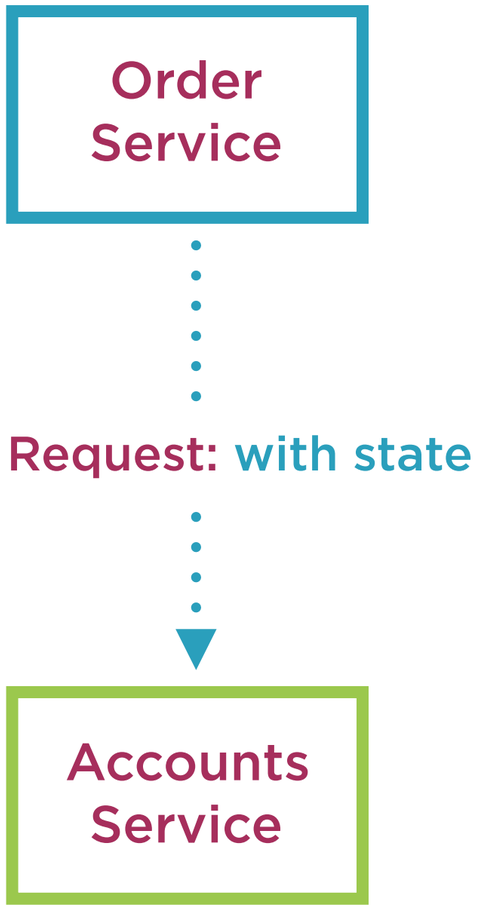
Other object could be an external API

Proxy object doesn't contain business logic

Proxy object is a wrapper which provides

- Simplification
- Transformation
- Security
- Validation

Stateless Service Pattern



What are stateful services

- Avoid for microservices

Stateless Service Pattern

- No state information maintained
- Clients maintain state
- State sent as part of request

Advantages for microservice architecture

- Scalability, performance and availability

Caveat

- Increased network traffic

Demo

APIs to illustrate

- Stateless Restful API
  - Using facade design pattern
  - Using proxy design pattern

Summary

Introduction

Functional Requirements

Architecture Options

REST Architectural Style

API Architectural Patterns

- Facade Design Pattern
- Proxy Design Pattern
- Stateless Service Pattern

Microservices Architectural Design Patterns Playbook