



Good API Design

Zdenek “Z” Nemec
z@goodapi.co

GOOD API

Independent API Consulting



Adam Kliment

[Follow @ntmlk](#)

Erik Wilde

[Follow @dret](#)

Zdenek "Z" Nemec

[Follow @zdne](#)



Good API Design

What it takes to design a good API?

API-first

The Three Pillars

Three Pillars of Digital Transformation API-first



Business



Organization



Technology

Business

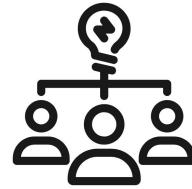


APIs are the products that you create, manage, sell, measure, recombine, and retire.

API First means that if a product in your organization does not have an API, it essentially does not exist.

APIs are the connective fabric.

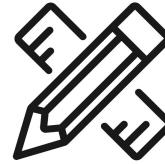
Organization



No matter how good your service or product is, if it is delivered through a bad API, this substantially diminishes the quality of the product.

Teams embrace the fact that APIs are their main deliverables.

Technology



Everything that gets created or consumed in the organization is based on APIs.

The quality of APIs greatly matter for how easy it is to create and consume company products.

APIs change must not create expensive and time-consuming ripple effects through a chain of API dependencies.

APIs are no longer just about the technology

API-first in Practice

Everything produced and consumed in the organization has an API



Sports vs. Tech Company



Where API Comes First



API as first UI



API Before Implementation



API Documentation

The API Product

The API Product

API-as-a-Product

Direct API monetization.

Where possible, the company product is offered directly through a paid API.

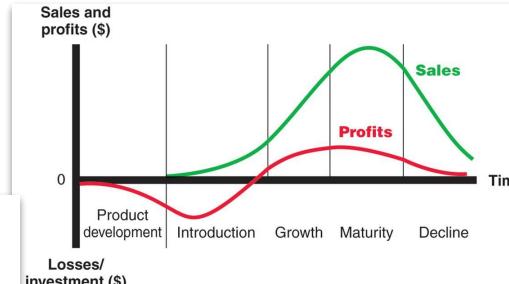
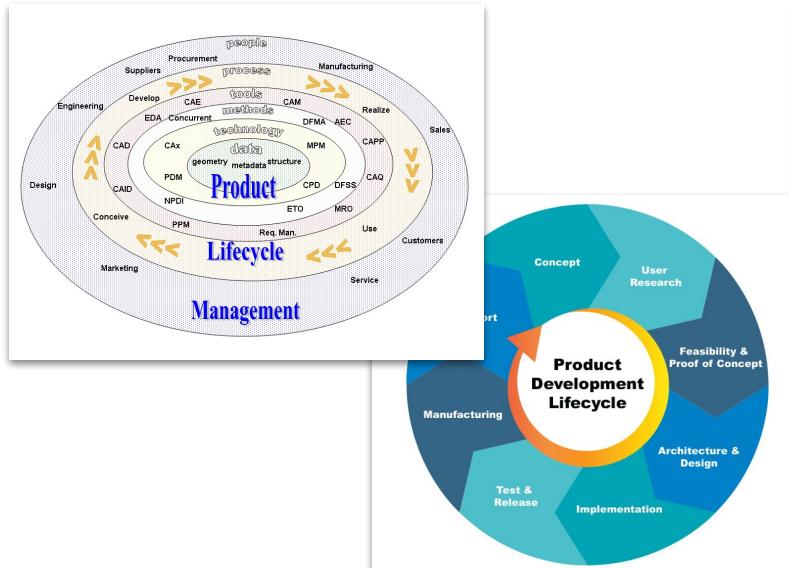
Product as API

Indirect “monetization”.

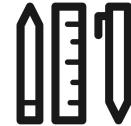
Each product is an API and used through its API. APIs are contributing indirectly to the bottomline through increasing stickiness, driving revenue-generating activities, leveraging partner channels, increasing efficiency or leading innovation.

Design-first

Treat API Development as Product Development



Design-first



API Before Implementation

As with any good product, the work on API product starts with hypothesis, verification and design–prototyping.

Introduction to API Design

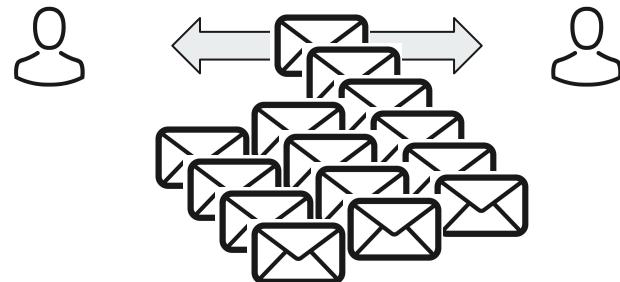
API Design



The Email Story

RE: RE: RE: RE: RE: RE: The API We Want

Have you ever tried to discuss what an API should look like over a plain email?



API Design



Good API Design

API design is API architectural style decision together with a set of additional design constraints applied on a specific domain, business, domain, organizational and technical requirements.

API Design



Good API Design is Verified

- Prototype
- Share
- Verify
 - Stakeholders
 - Users
 - API Developers

API Description

API Description

API Design is formalized using one of the API Description formats

- API Blueprint
- RAML
- Swagger
- OpenAPI Specification 2.0 (fka Swagger)
- OpenAPI Specification 3.0
- AsyncAPI Specification
- GraphQL Schema
- Avro Schema
- Protobuf

API Description



OpenAPI Specification 2.0 (formerly Swagger)

```
swagger: '2.0'
info:
  description: This is a simple API
  version: 1.0.0
  title: Simple Inventory API
  # put the contact info for your development or API team
  contact:
    email: you@your-company.com

  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html

# tags are used for organizing operations
tags:
- name: admins
  description: Secured Admin-only calls
- name: developers
  description: Operations available to regular developers

paths:
  /inventory:
    get:
      tags:
        - developers
        summary: searches inventory
        operationId: searchInventory
        description: |
          By passing in the appropriate options, you can search for
          available inventory in the system
      produces:
        - application/json
```

API Description

API Description must be:

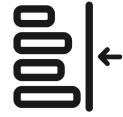
- Human & machine readable
- Accessible
- Understood by Stakeholders, Developers, API Consumers, Tech writers, DevOPS, Support team
- Versioned in VCS
- Treated as Product Requirements

Approved API design becomes the contract

Design Consistency

API Design Maturity





Consistency

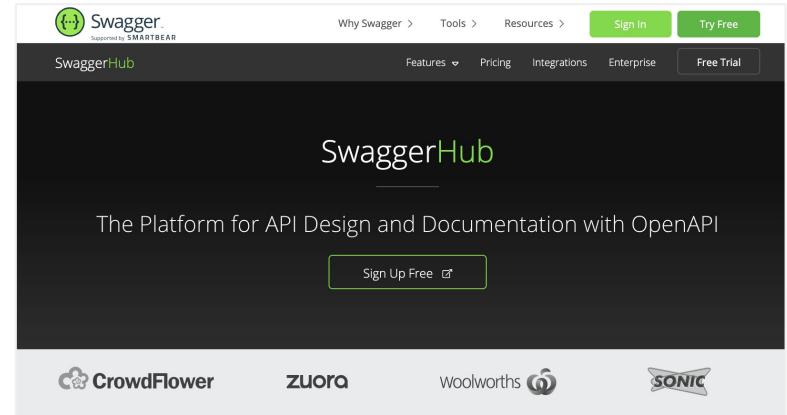
Consistency

Consistency is a win-win for both API consumers and providers:

- Reduces complexity
- Encourages re-use
- Enables upsell
- Reduces time-to-market
- Reduces maintenance
- Enables scaling

How to achieve consistency

Visibility



The screenshot shows the SwaggerHub homepage. At the top, there's a navigation bar with links for "Why Swagger >", "Tools >", "Resources >", "Sign In", and "Try Free". Below the navigation is a secondary menu with "Features", "Pricing", "Integrations", "Enterprise", and a "Free Trial" button. The main content area features the "SwaggerHub" logo and the tagline "The Platform for API Design and Documentation with OpenAPI". A "Sign Up Free" button is prominently displayed. At the bottom, there's a footer section with logos for CrowdFlower, zuora, Woolworths, and SONIC.



API Repository

Organization-wide API repository – register with every API design in the organization.

Standardization

The screenshot shows a Confluence page with a yellow header containing the title 'STANDARDS AND GUIDELINES'. Below the header is a toolbar with buttons for View, Edit, Outline, Delete, and Revisions. A sidebar on the left lists navigation items: Documentation, Standards and Guidelines (which is selected), Core Principles, API Lifecycle, Glossary of Terms, and What is Next. The main content area contains sections for 'INTRODUCTION', 'GOALS', and 'ARCHITECTURE & STANDARDS WORKSTREAM'. The 'INTRODUCTION' section states: 'The goal of the Group API initiative is to make it easier for customers, partners and internal developers to connect to DPDHL APIs. By providing technical API documentation and the ability to obtain access credentials all in one central place, we are speeding up the whole development process of digital applications.' The 'GOALS' section lists four bullet points: 'Offer secure, simplified integration of our services into digital platforms and products', 'Achieve best-in-class API experience for customers, partners and internal developers', 'Reduce time-to-market with more robust, stable and better performing APIs', and 'Provide guidance, expertise and assistance in designing new APIs'. The 'ARCHITECTURE & STANDARDS WORKSTREAM' section states: 'These standards & guidelines are developed and maintained by the Group's API Initiative Architecture & Standards Workstream with input from the nominated BU architects. In the case of any questions or change needs regarding the API Standards & Guidelines, contact api@dpdhl.com.' The 'READING THE STANDARDS & GUIDELINES' section notes: 'The API Standards & Guidelines provide recommendations that are expressed using the RFC 2119 keywords. To avoid misinterpretation especially among non-native speakers, the word MUST is to be used rather than SHALL.' Below this is a link to 'RFC2119'.

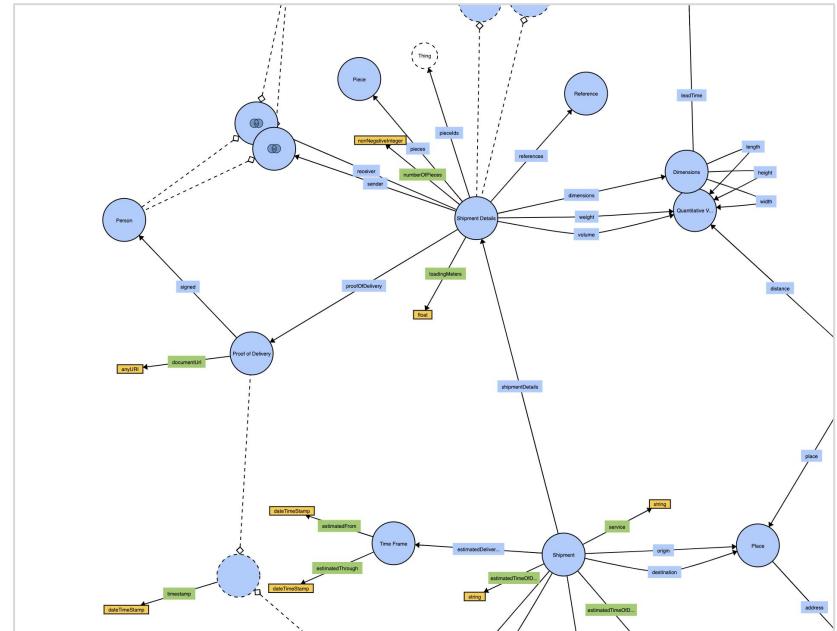
The screenshot shows the GitHub page for the 'adidas-group/api-guidelines' repository. The page title is 'adidas API Guidelines'. It features the 'adidas' logo at the top. Below the logo, there is a section titled 'Motivation' with the following text: 'The goal of this document is to facilitate the work and minimize the effort of all API users while protecting their investment and encouraging API adoption. The guidelines lay down the foundation for collaboration, stability, and extensibility.' There are also links to 'Download PDF' and 'Edit'.

The screenshot shows the 'API Stylebook' landing page. The page has a dark blue background with white text. It features the 'adidas' logo and the heading 'API Stylebook'. Below the heading is the subtext 'Collections Of Resources For API Designers.' and a 'Learn more' button. At the bottom right, there is a Twitter icon and a navigation bar with five dots.

Vocabulary Harmonization & Reconciliation

<https://vocabulary.dhl.com/logistics>

Vocabulary aligned with schema.org and GS1





Automated Consistency Checks

Automated Consistency Checks

Automated API guidelines consistency checks using tools like Apiary Style Guide or Stoplight Spectral.

ORACLE +  apiary

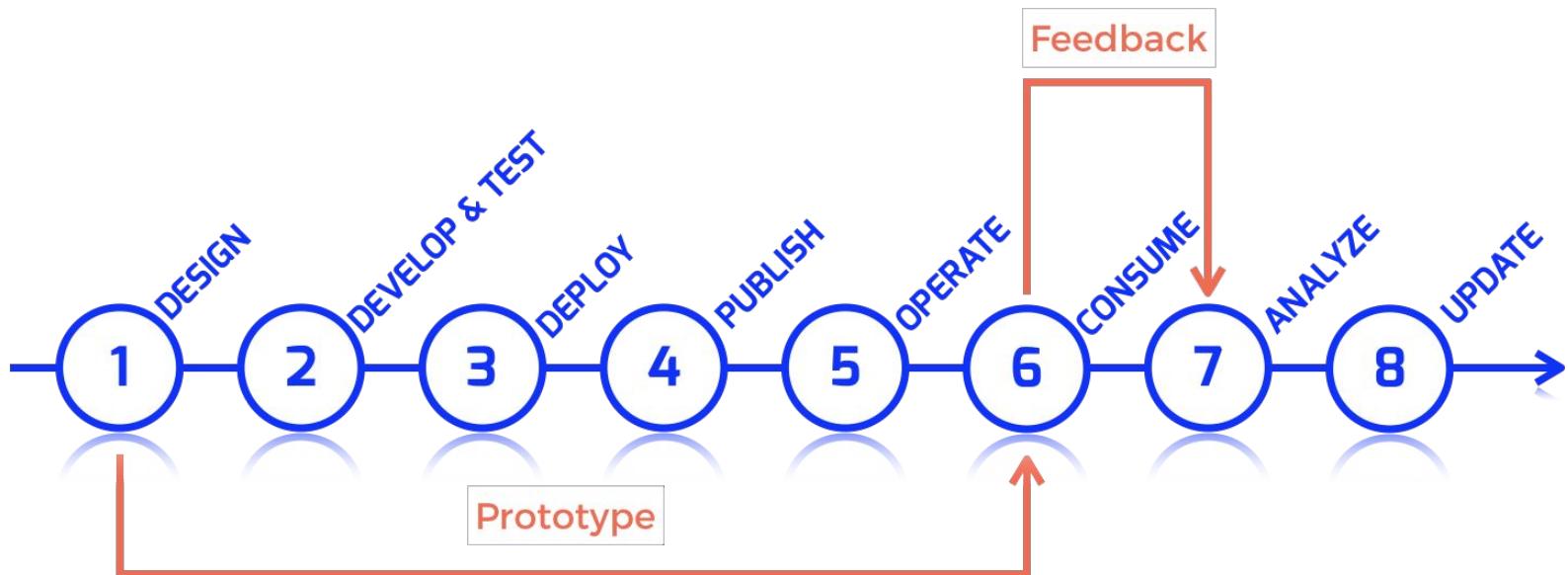


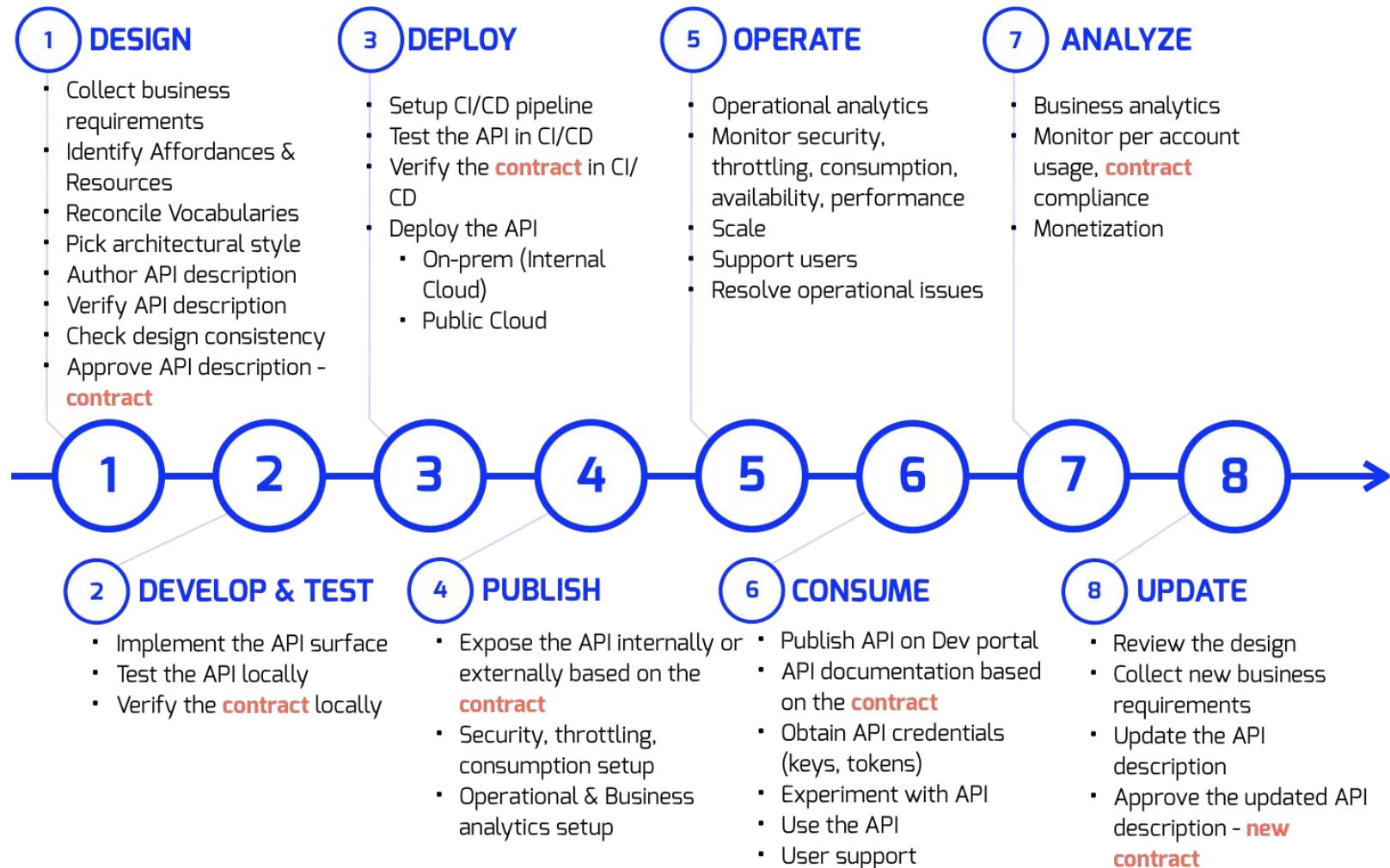
API Lifecycle

Waterfall Lifecycle



Agile API Lifecycle





API Design

API Design Phase

1. Collect business requirements, use cases

Greenfield vs. Brownfield scenario, verification of hypothesis and business alignment

2. Understand, Reconcile and Define Vocabularies

Domain-driven design, explore vocabularies and data models

3. Pick architectural style, decide on the protocol

Role of the API Architect



[API Architectural Styles Workshop](#)

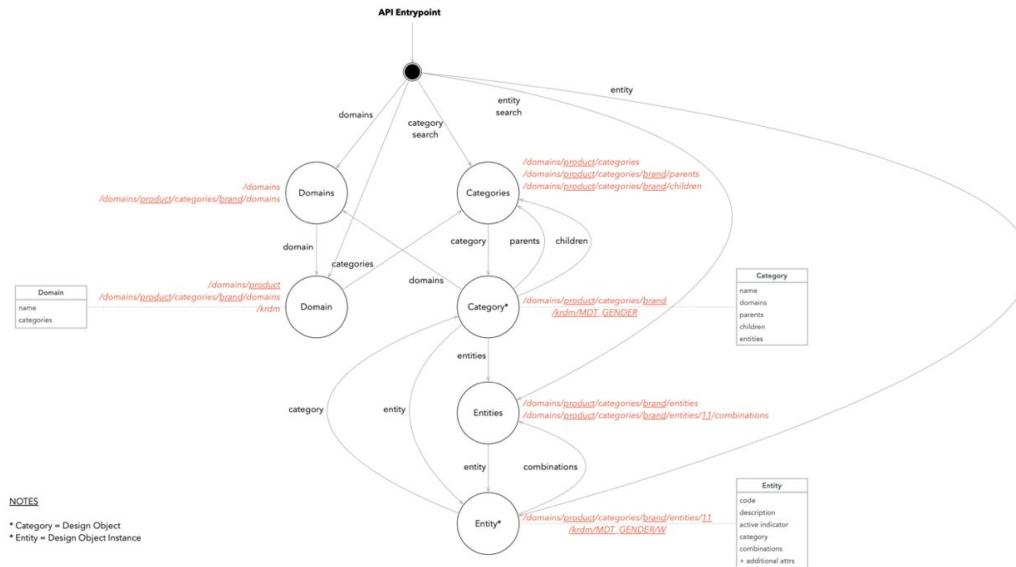
4. Design the API

Based on the style and protocol selection

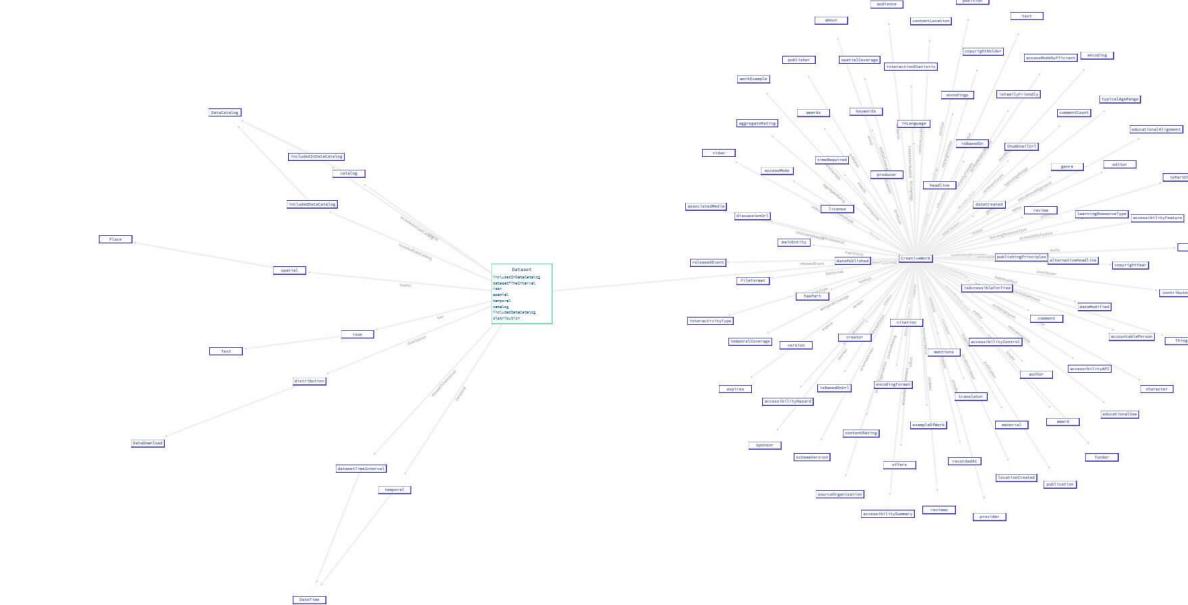
Design API – Web APIs

- 1. Identify Affordances & Resources**
- 2. Draw Finite State Machine Diagram**
- 3. Decide on the Message Formats (Resource Representations)**
- 4. Author API Description**
 - a. Define resources & actions
 - b. Define models (e.g. using supermodel.io)
 - c. Add authentication mechanism

Web API Automaton



Supermodel Data Model



Design API – GraphQL APIs

- 1. Identify core queries**
- 2. Identify mutations**
- 3. Define models**
(e.g. using supermodel.io)
- 4. Author GraphQL Schema**

API Design Phase

5. Check API Design consistency

Style-guides & design patterns, design governance

6. Verify API Description

Developers, clients & stakeholders; Use mock and documentation driven by the API description

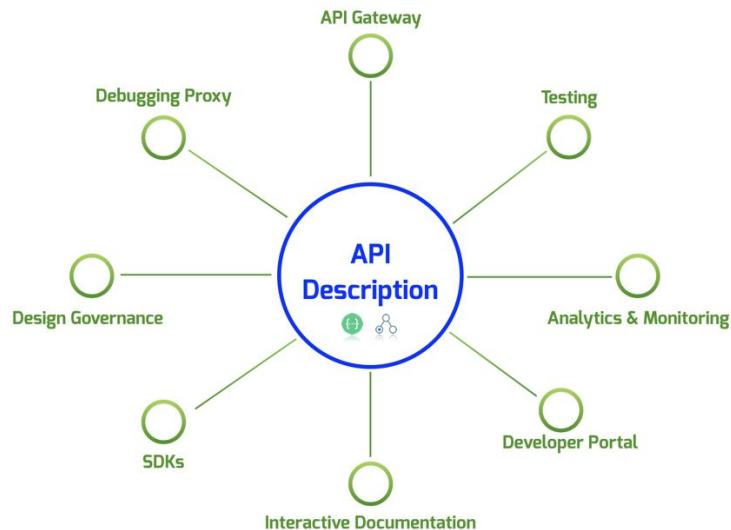
7. Approve the API Description–contract

Binding for every party in API lifecycle

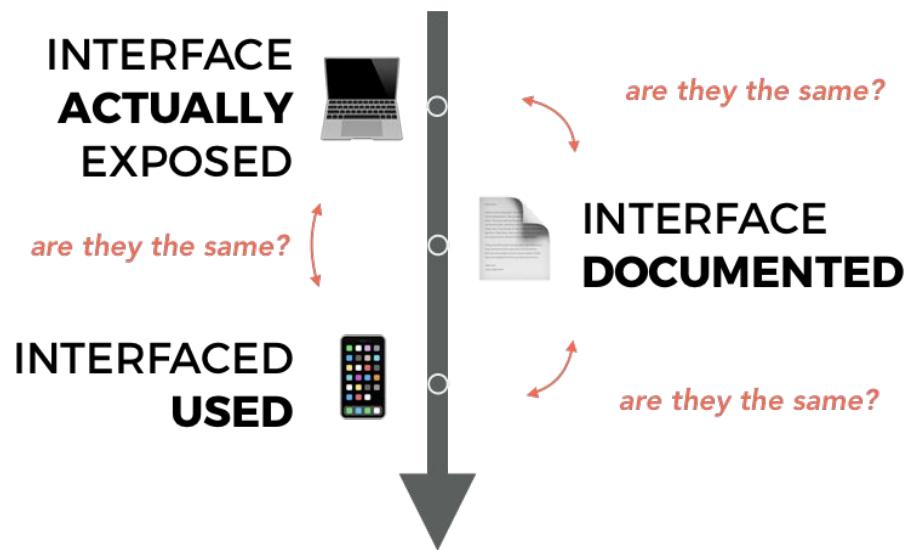
8. Evolve the design–Update Phase

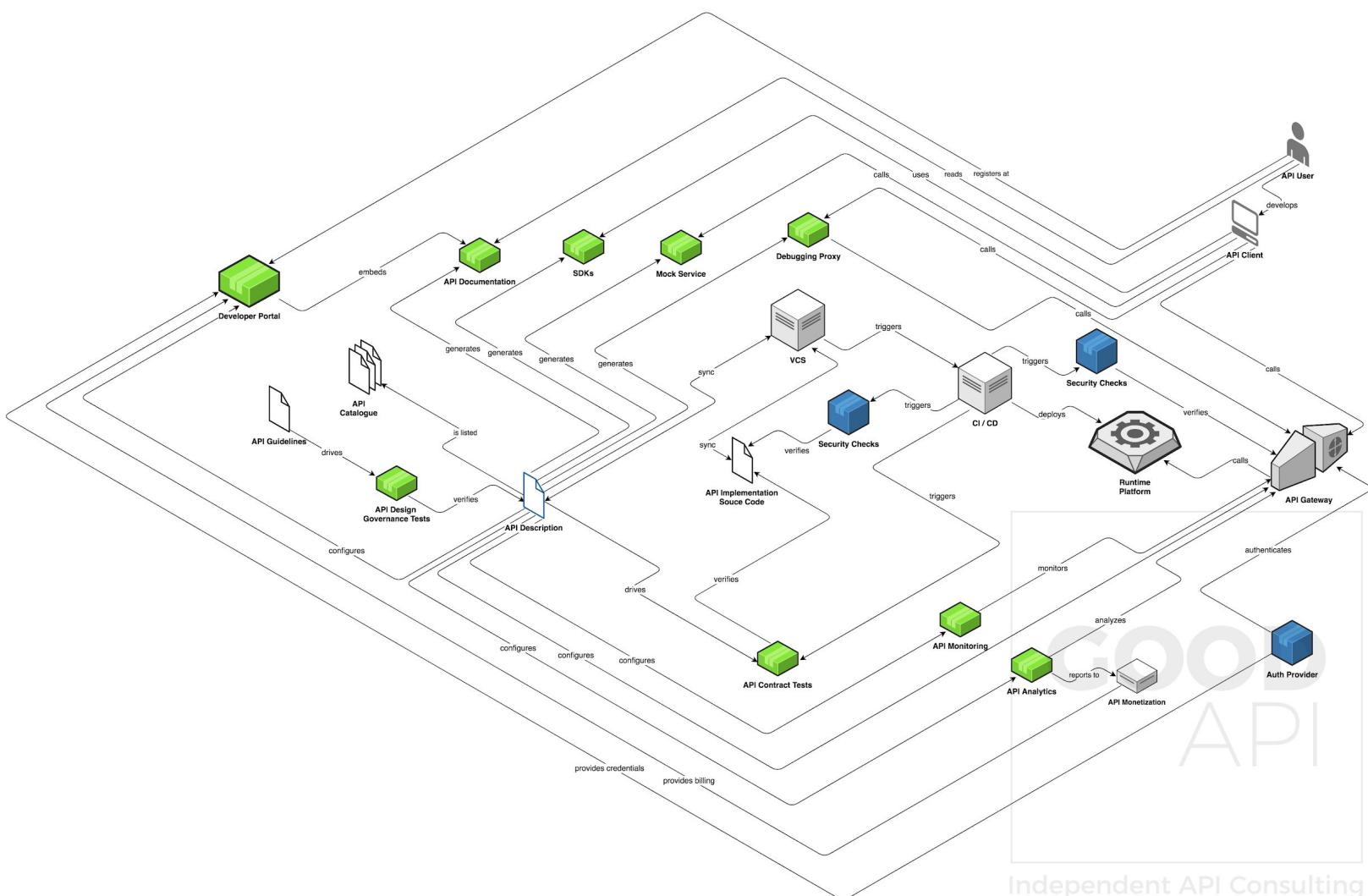
Contract-driven API Lifecycle

API Lifecycle Must be Contract-driven



Keeping Things in Sync is Hard





Evolving the API (design)

The first rule of Fight Club



Never break clients

Any modification to an existing API MUST avoid breaking changes and MUST maintain backward compatibility.

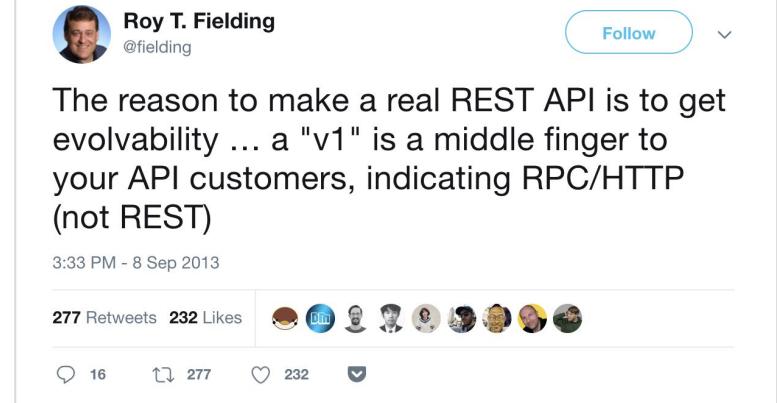
Rules for extending



1. You MUST NOT take anything **away**
2. You MUST NOT change **processing rules**
3. You MUST NOT make optional things **required**
4. Anything you add MUST be **optional**

Identifier Stability

URI is an identifier



Roy T. Fielding
@fielding

Follow

The reason to make a real REST API is to get evolvability ... a "v1" is a middle finger to your API customers, indicating RPC/HTTP (not REST)

3:33 PM - 8 Sep 2013

277 Retweets 232 Likes

16 277 232

Backward incompatible changes

New Resource Variant

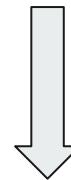
T0

Parameters `first` and `last` optional

T+1

Parameter `first` required

/greeting?first=John&last=Appleseed



New Resource Variant

/named-greeting?first=John&last=Appleseed

Hands-on

1. Collect business requirements, use cases

We are working for a florist company Florist LLC. selling physical products—flowers. As we want to transform our business for the digital era we want to build an order API to provide easy integration for our partners.

For the start we would like our API to serve following use cases:

As a user—customer of the florist company I want to be able to:

1. Retrieve list of my orders
2. Retrieve details about one of my orders, including its items
3. Check status of an order
4. Place a new order
5. Cancel order that has not been yet shipped
6. Return an order

These use cases were already verified with the partners-potential users.

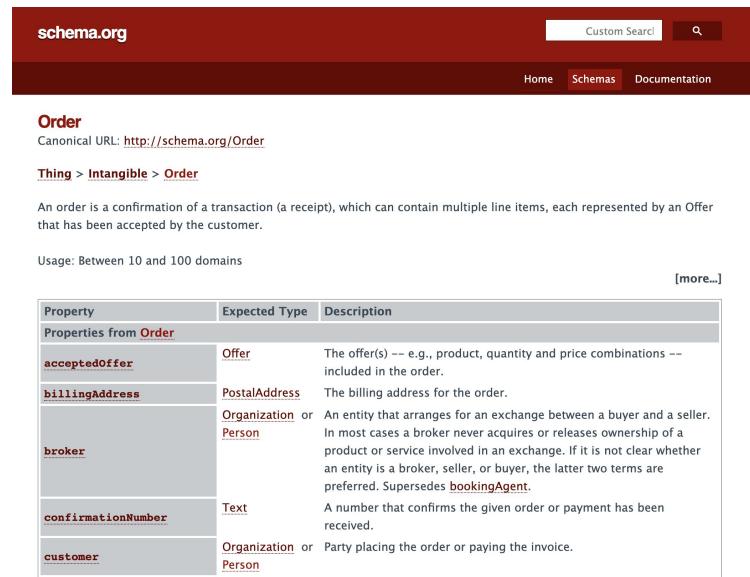
2. Reconcile and Define Vocabularies

Our florist company does not have its own formal vocabulary. But after a quick check of schema.org we are confident we can use the schema.org Order concept without any need for extensions.

Our company order management systems defines the following statuses (schema.org OrderStatus) of the order:

- **Pending (Created) ***
- Confirmed
- **Cancelled ***
- Shipped
- Received
- **Returned ***

Our API will allow its users to create/change only those statuses marked with (*)



The screenshot shows the schema.org website with the URL <http://schema.org/Order>. The page title is "Order". It includes a navigation bar with links for "Home", "Schemas", and "Documentation". Below the title, it says "Canonical URL: <http://schema.org/Order>". The main content area starts with "Thing > Intangible > Order". A note states: "An order is a confirmation of a transaction (a receipt), which can contain multiple line items, each represented by an Offer that has been accepted by the customer." Usage information indicates "Between 10 and 100 domains" and a link "[more...]".

Property	Expected Type	Description
Properties from Order		
acceptedOffer	Offer	The offer(s) -- e.g., product, quantity and price combinations -- included in the order.
billingAddress	PostalAddress	The billing address for the order.
broker	Organization or Person	An entity that arranges for an exchange between a buyer and a seller. In most cases a broker never acquires or releases ownership of a product or service involved in an exchange. If it is not clear whether an entity is a broker, seller, or buyer, the latter two terms are preferred. Supersedes bookingAgent .
confirmationNumber	Text	A number that confirms the given order or payment has been received.
customer	Organization or Person	Party placing the order or paying the invoice.

3. Pick architectural style, decide on the protocol

Based on our business requirements and organizational constraints our API architect has decided to create the API as a proper REST API using the HTTP1.1 protocol.

4. Design API – Web APIs

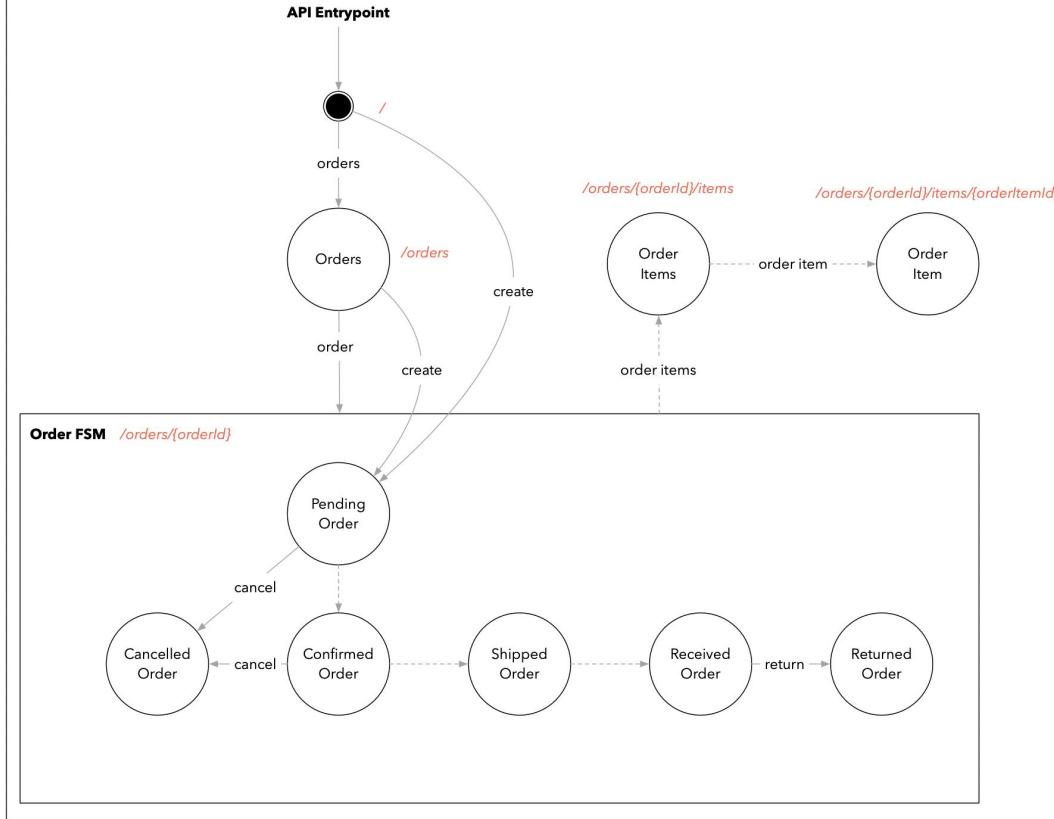
With use cases, domain model and architectural style given, we are going to design the REST API.

Draw FSM

Your task is to identify the resources in the API and relations between them and draw a finite state machine (FSM – automaton) diagram of the API.



Florist LLC. Order API FSM



Florist LLC. Order API Automaton

Author API Description

Search

Swagger: '2.0'
info:
version: 1.0.0
title: Florist LLC Order API
description: >-
Florist LLC. Order API

This is a Good API <http://goodapi.co> demo API.

Message Formats

This API uses the following message formats:

- **RESTful JSON** <<https://restfuljson.org/>>
- Every successful response is following the RESTful JSON conventions to communicate the hypermedia links (HATEOAS). Clients are kindly requested to follow provided links instead of hardcoding the endpoints URLs.
- **Problem Detail** <<https://tools.ietf.org/html/rfc7807>>
- In the case of a error the Problem Detail `application/problem+json` format is used to communicate the details about the problem.

contact:
email: hello@goodapi.co

license:
name: Apache 2.0
url: <http://www.apache.org/licenses/LICENSE-2.0.html>

securityDefinitions:
"API Key":
type: apiKey
in: header
name: api-key

security:
- "API Key": □

tags are used for organizing operations
tags:

Aa 🔍 💬 SAVE ⚙️

This is a Good API <http://goodapi.co> demo API.

Message Formats

This API uses the following message formats:

- RESTful JSON <https://restfuljson.org/>
Every successful response is following the RESTful JSON conventions to communicate the hypermedia links (HATEOAS). Clients are kindly requested to follow provided links instead of hardcoding the endpoints URLs.
- Problem Detail <https://tools.ietf.org/html/rfc7807>
In the case of a error the Problem Detail `application/problem+json` format is used to communicate the details about the problem.

Contact the developer
Apache 2.0

Schemes
HTTPS

Authorize

USERS Operations available to all API users

GET	/	Retrieve API Root	🔒 ↻
GET	/orders	Retrieve List of Orders	🔒 ↻
POST	/orders	Create Order	🔒 ↻
GET	/orders/{orderId}	Retrieve Order	🔒 ↻
PATCH	/orders/{orderId}	Cancel or Return Order	🔒 ↻

Florist LLC. Order API API Description

5. Check API Design consistency

If the organization has API Standards and guidelines this is the time where we should check the compliance of the authored API description.

Ideally, the checks are automated.

6. Verify API Description

Using the tools driven by the authored API description we are going to distribute the API prototype documentation and experiment with the stub implementation (mock server).

For a good design, it is important the everybody involved in the API lifecycle is engaged including the potential customers.

Call the API

Your task is to explore the Florist LLC. Order API documentation and make an actual call to the API! Use your preferred tool to make a call (cURL, Postman, Paw). Try to call different endpoints and observe the response.

The URL of the API documentation is:

<https://app.swaggerhub.com/apis/goodapi/demo-order-api/1.0.0#/>

or:

<https://bit.ly/2I1s0Bc>

GET https://virtserver.swaggerhub.com/goodapi/demo-order-api/1.0.0/orders

Description Headers URL Params Body Auth Options

Request

Request Description

```

1 HTTP/1.1 200 OK
2 Date: Fri, 20 Mar 2018 18:18:49 GMT
3 Content-Type: application/json
4 Content-Length: 529
5 Connection: close
6 Access-Control-Allow-Headers: X-Requested-With,Content-Type,Accept,Origin
7 Access-Control-Allow-Origin: *
8 Access-Control-Allow-Methods: *
9 Access-Control-Credentials: true
10
11 {
12   "url" : "/orders",
13   "createdOrderUrl" : "/orders",
14   "orders" : [ {
15     "url" : "/orders/001",
16     "returnUrl" : "/orders/001",
17     "orderStatus" : "received",
18     "orderDate" : "2020-06-21T14:07:17Z",
19     "orderItems" : [ ]
20   },
21   {
22     "url" : "/orders/002",
23     "cancelUrl" : "/orders/002",
24     "orderStatus" : "pending",
25     "orderDate" : "2020-12-23T22:59:00Z",
26     "orderItems" : [ ]
27   },
28   {
29     "url" : "/orders/003",
30     "orderStatus" : "shipped",
31     "orderDate" : "2020-12-22T10:00:00Z",
32     "orderItems" : [ ]
33   }
34 }

```

z — bash — 87x23

```

yt:~ z$ curl "https://virtserver.swaggerhub.com/goodapi/demo-order-api/1.0.0/orders"
{
  "url" : "/orders",
  "createOrderUrl" : "/orders",
  "orders" : [ {
    "url" : "/orders/001",
    "returnUrl" : "/orders/001",
    "orderStatus" : "received",
    "orderDate" : "2020-06-21T14:07:17Z",
    "orderItems" : [ ]
  },
  {
    "url" : "/orders/002",
    "cancelUrl" : "/orders/002",
    "orderStatus" : "pending",
    "orderDate" : "2020-12-23T22:59:00Z",
    "orderItems" : [ ]
  },
  {
    "url" : "/orders/003",
    "orderStatus" : "shipped",
    "orderDate" : "2020-12-22T10:00:00Z",
    "orderItems" : [ ]
  }
]
yt:~ z$

```

Florist LLC. Order API Mock Interaction

7. Approve the API Description, establish Contract

With verifications done, approved API Descriptions becomes the contract binding all parties in the API lifecycle.

7. Evolve, Update or Retire

As time goes a new requirements - use cases are requested:

As a user–customer of the florist company I want to be able to:

1. Back-order an already received order
2. Retrieve trimmed-down list of my orders without all the order line information

Risk: Potentially backward-incompatible change



GOOD
API

Independent API Consulting

Icons by IconMark from the Noun Project



Independent API Consulting



angelcam

Deutsche Post DHL
Group



GLOBAL
LEGAL
ENTITY
IDENTIFIER
FOUNDATION



WINDING
TREE

ytica