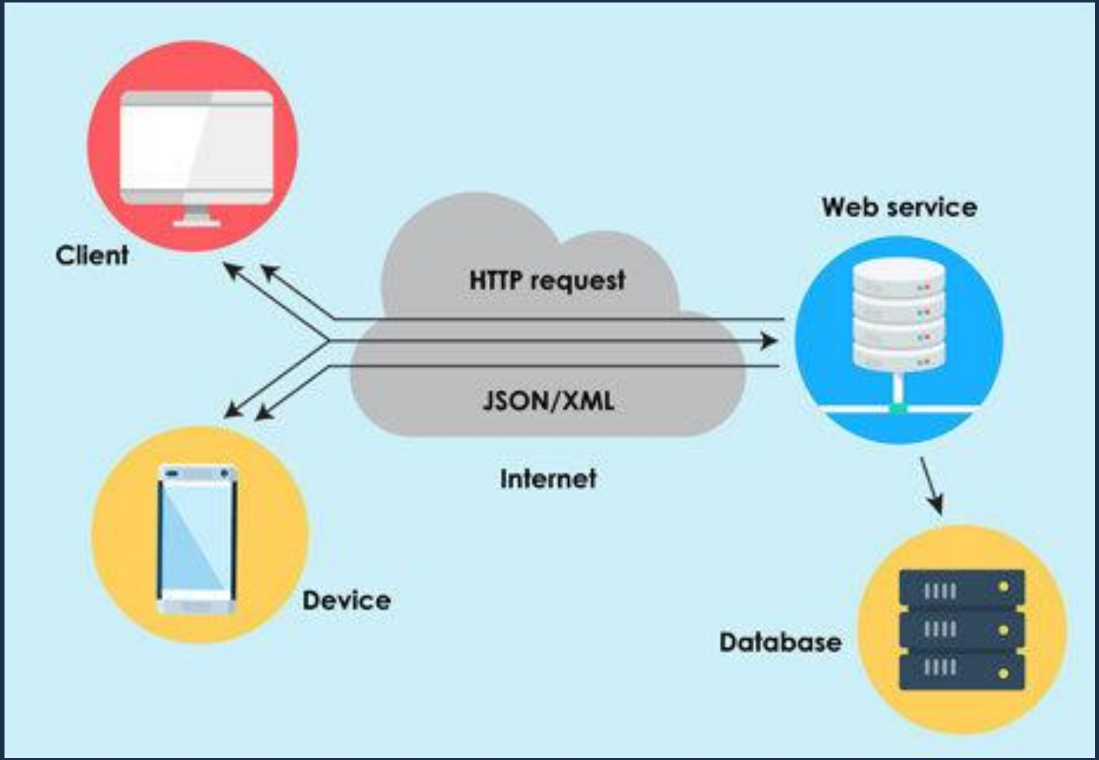
API design and implementation principles

APIs are a big topic –
These are just some core ideas, specifically for CAMD government leads

18F

Application Programming Interface



18F

Via https://www.logigear.com/magazine/API-testing/codeless-api-testing/

The usual software “-ilities” apply:

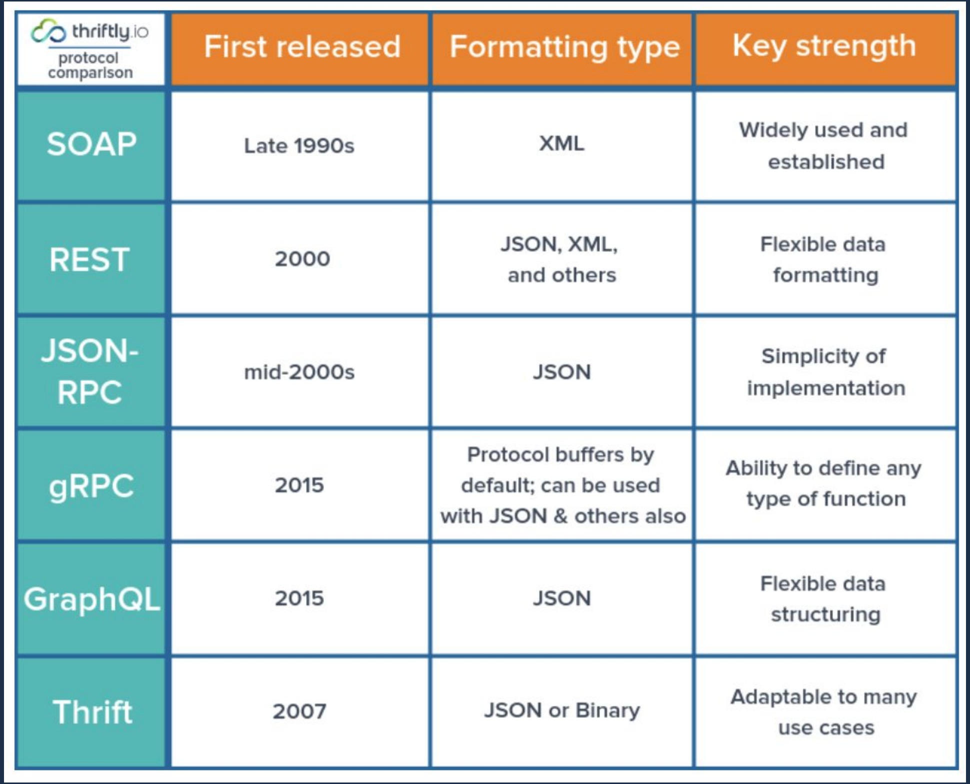
- Consistency
- Flexibility
- Usability
- Agility
- Performance
- Security

1/ API First Design

1. Your API is the first user interface of your application
2. Your API comes first, then the implementation
3. Your API is described (and maybe even self-descriptive)

Via https://medium.com/adobetech/three-principles-of-api-first-design-fa6666d9f694

2/ REST vs everything else



thriftly.io protocol comparison	First released	Formatting type	Key strength
SOAP	Late 1990s	XML	Widely used and established
REST	2000	JSON, XML, and others	Flexible data formatting
JSON-RPC	mid-2000s	JSON	Simplicity of implementation
gRPC	2015	Protocol buffers by default; can be used with JSON & others also	Ability to define any type of function
GraphQL	2015	JSON	Flexible data structuring
Thrift	2007	JSON or Binary	Adaptable to many use cases

Via https://blog.thriftly.io/know-your-api-protocols

3/ Tools that can help: OpenAPI and Postman

4/ Monolithic vs Microservice

Monolithic services

1. Designed, developed and deployed as a single unit.
2. Has to be scaled as a single application and difficult to scale with conflicting resource requirements
3. One unstable service can bring the whole application down.

18F

Via https://medium.com/microservices-in-practice/microservices-in-practice-7a3e85b6624c

Microservices

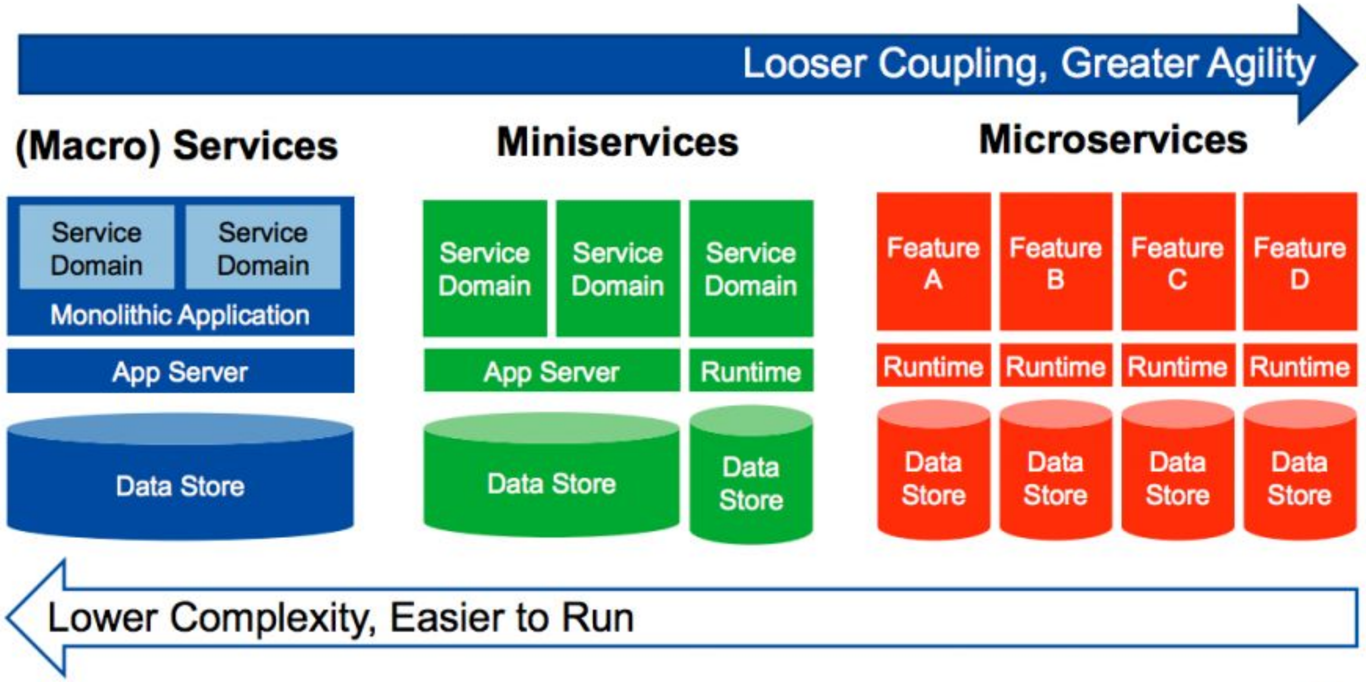
Suite of small and independent services that are running in its own process, developed and deployed independently.

Via https://medium.com/microservices-in-practice/microservices-in-practice-7a3e85b6624c

Microservices

1. Independently deployable and scalable executable
2. Have a single business responsibility
3. Be loosely coupled

Think Multigrained, Not Just "Micro"



Via https://thenewstack.io/miniservices-a-realistic-alternative-to-microservices/

5/ API governance & Conway's law

Primary goal of API governance:
Consistency

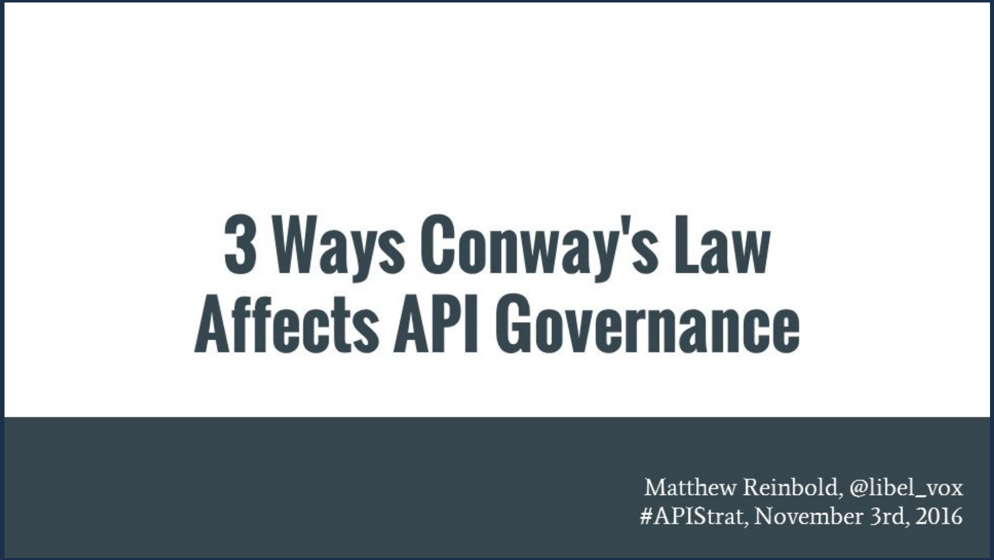
Via https://swagger.io/resources/articles/best-practices-in-api-governance/

M. Conway, How Do Committees Invent?, 1967

> “*organizations which design systems … are constrained to produce designs which are copies of the communication structures of these organizations.*”

A “must read” presentation:

https://matthewreinbold.com/2017/12/07/ConwayAndAPIDesign/


3 Ways Conway's Law Affects API Governance

Matthew Reinbold, @libel_vox
#APIStrat, November 3rd, 2016

Other governance goals:

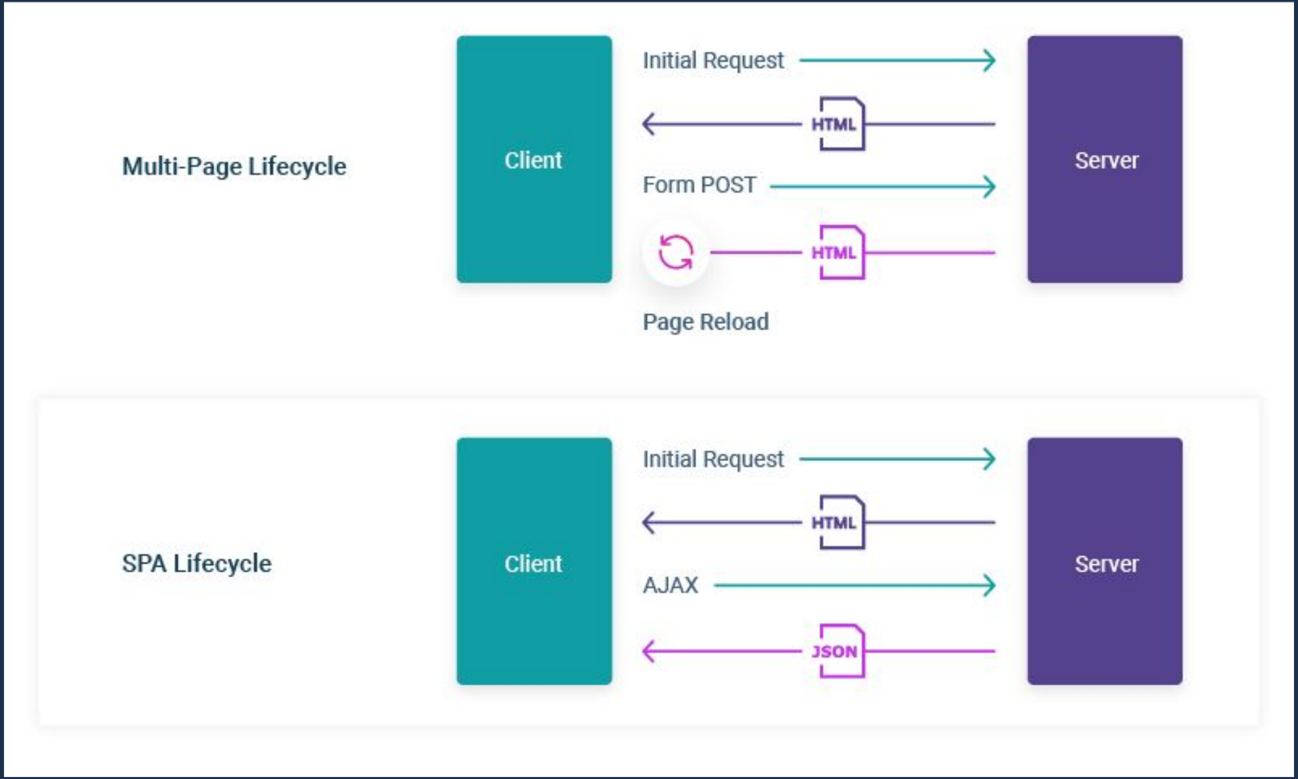
1. Centralization for security policies and enforcement
2. API discovery
3. Deprecation policy

18F

Via https://swagger.io/resources/articles/best-practices-in-api-governance/

6/ Ligtening round

Single Page Application vs traditional web app



Via https://dotcms.com/blog/post/what-is-a-single-page-application-and-should-you-use-one-

Use OpenID Connect for authentication

Logic in API service (middle tier), not database

State in database,
not API service (middle tier)

Comments, questions?

Resources:

https://blog.thriftly.io/know-your-api-protocols
https://www.guru99.com/comparison-between-web-services.html
http://www.melconway.com/Home/Conways_Law.html
https://learning.getpostman.com/docs/postman/mock_servers/intro_to_mock_servers/
https://12factor.net/
https://thenewstack.io/miniservices-a-realistic-alternative-to-microservices/
https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-protocols-openid-connect-code
https://github.com/katopz/best-practices/blob/master/best-practices-for-building-a-microservice-architecture.md
https://medium.com/microservices-in-practice/microservices-in-practice-7a3e85b6624c
https://dzone.com/articles/10-best-practices-for-microservice-architectures
https://3factor.app/
https://12factor.net/
https://cloud.google.com/blog/products/api-management/api-design-best-practices-common-pitfalls