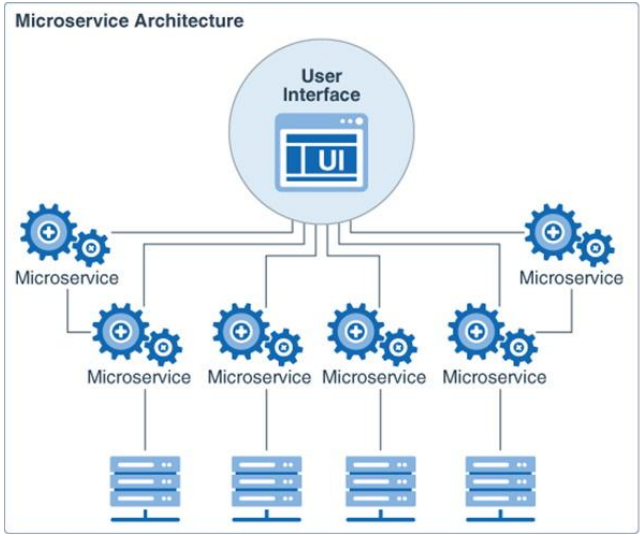
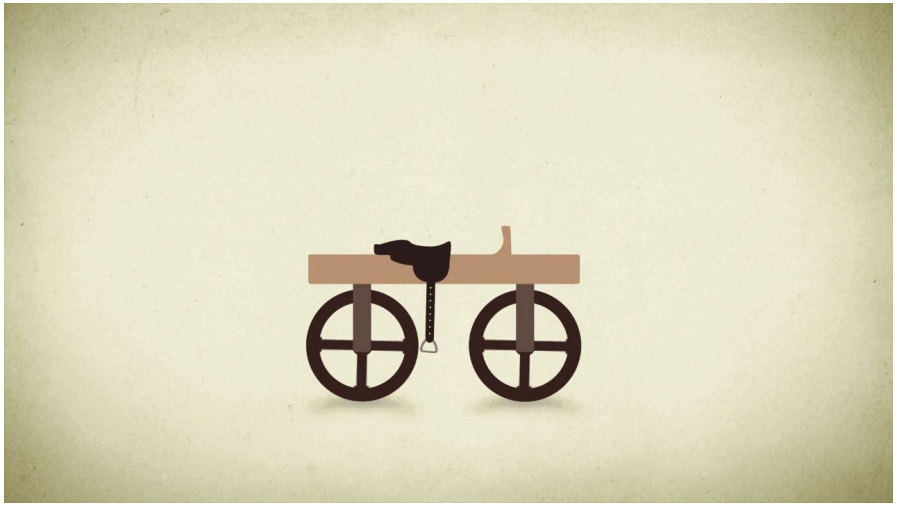
Les micro-services

what and why ?

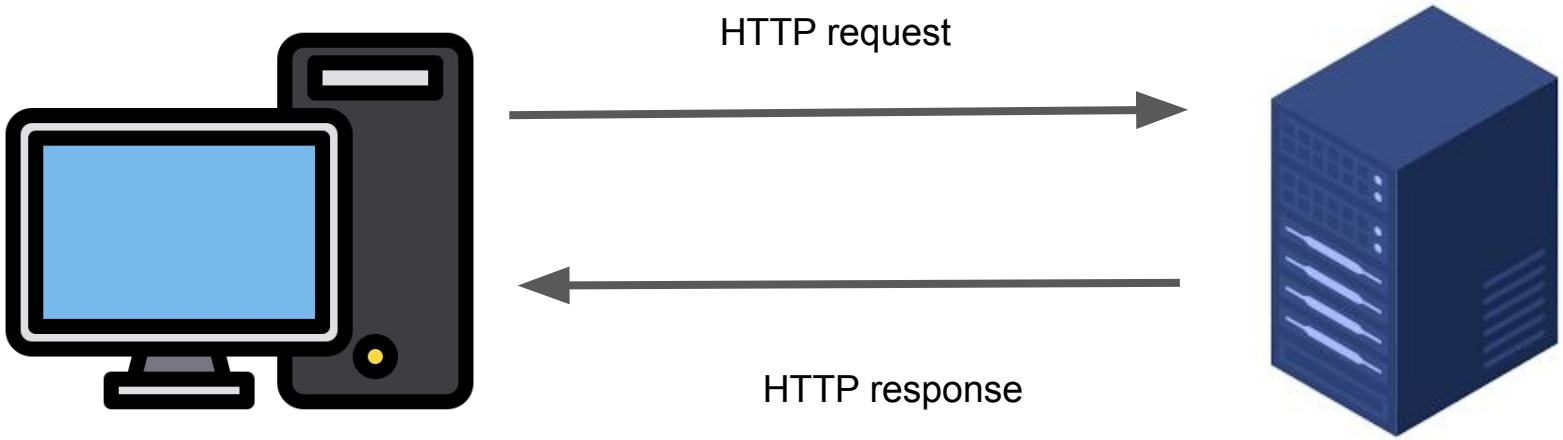
Micro-service is Software Architecture



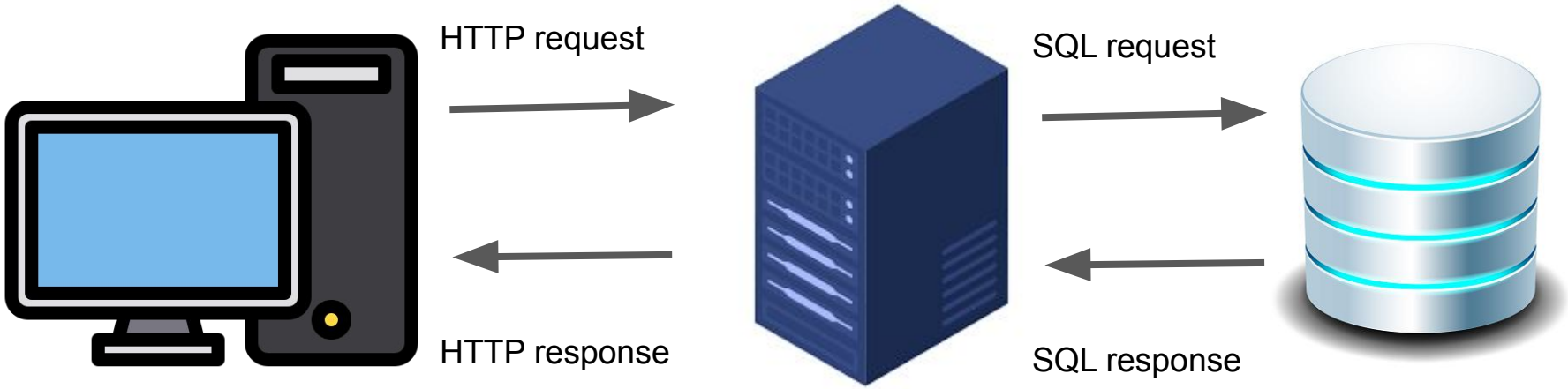
A very brief history Software architecture



first App



first App with database



One App

Simple

All code available in the same place

One app to deploy

BDD Side

Simple

one database to manage

low latency (especially if on the same server)

Why change ?

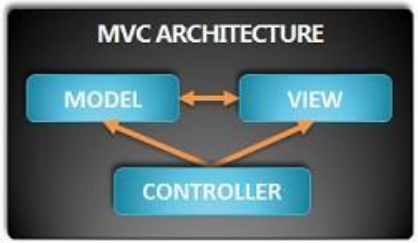
modularity

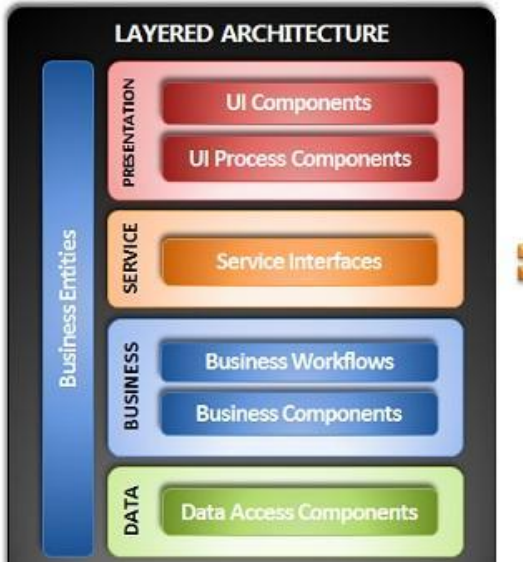
build time

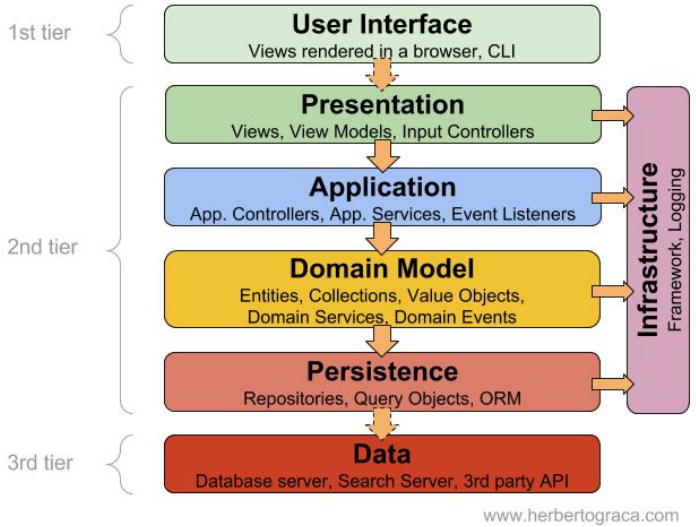
deployment

scaling / tuning

Improved Architecture : layers







some down side

~~modularity~~

build time

deployment

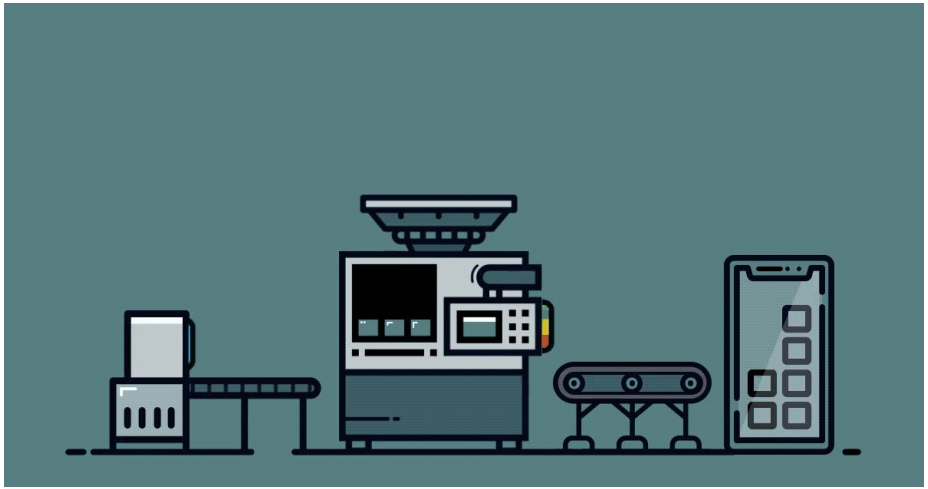
scaling / tuning

IT Evolution

CI/CD

on demande Virtualisation

automation



Micro-service

Micro-service

micro : small

service : one responsibility

autonomous application (replacement & update)

centralized management

centralized data

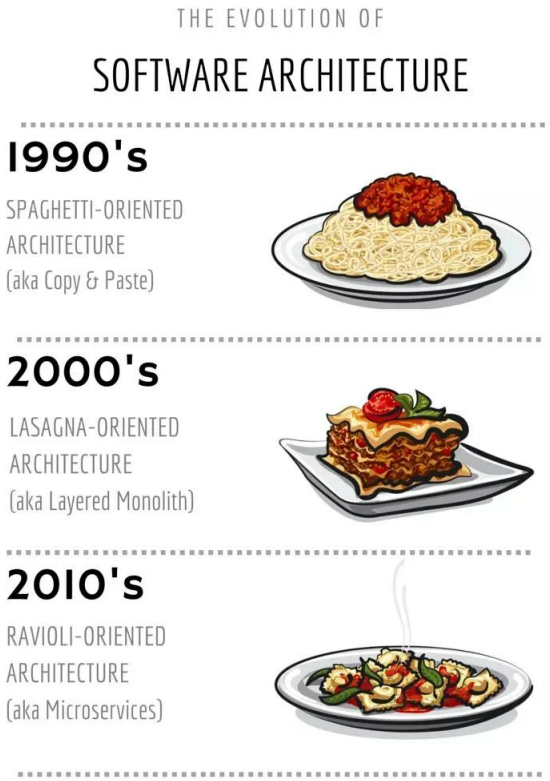
Design for failure

Monolith First

⇒ Split the monolith

https://martinfowler.com/bliki/MonolithFirst.html

The pasta theory of code



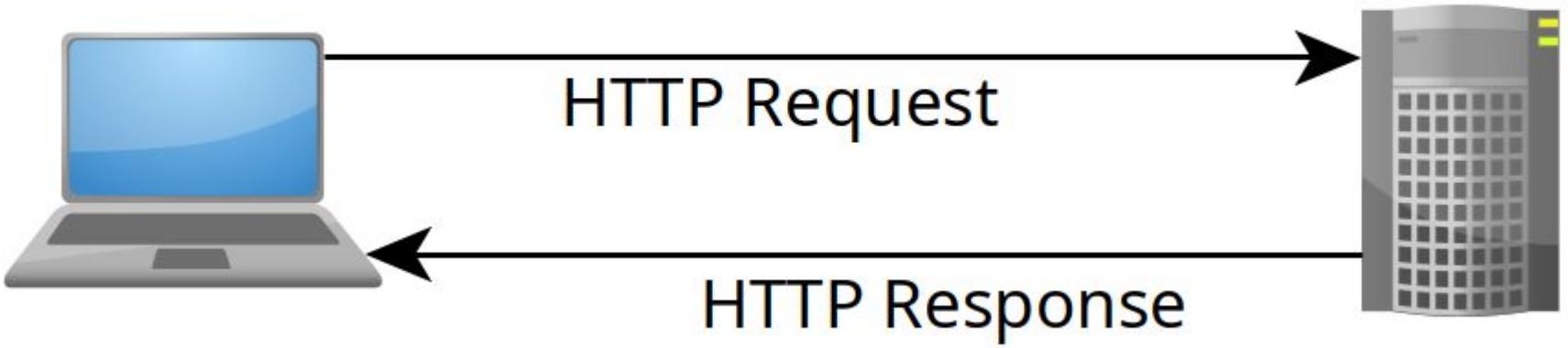
Good practice

API : technologie Agnostique,Low coupled

KISS : pour permettre l’utilisation simple

Communication

API : REST



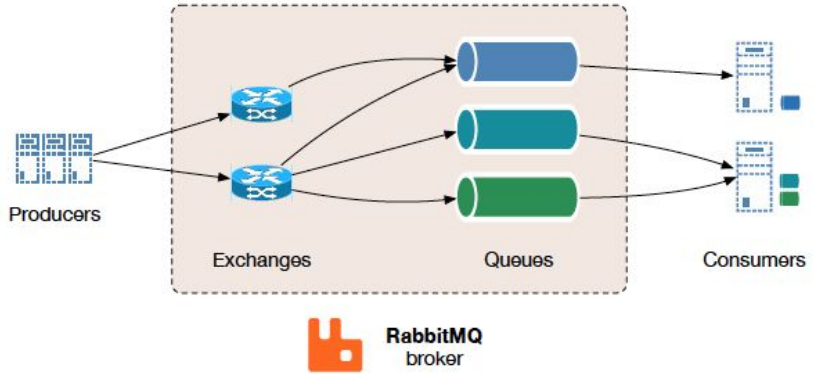
REST : advantage

compatibility

- all platform
- all language
- lots of clients (curl, browsers,...)
- format (xml, json, yaml,...)

Communication

message Broker



RabbitMQ

redis

REmote DIctionary Server

APACHE kafka™

Message broker

event publication

asynchronous

multiple emitter and/or receiver

Why you should do microservices

hard to maintain and update existing app

team composition and team responsibility

different technology

different deployment schedule

different scaling

Why you should NOT do microservices

existing app working

complexity outside the code

network issue

What you should consider

-microservice size and responsibility

- security

    update more servers (patching,...)

    update more codebase

automation

existing monolith

easier to cut than to re-assemble

Documentation is key

- Markdown
- mermaid

Demo :

https://www.markdownguide.org/cheat-sheet/

https://mermaid.live/edit

Documentation

https://simongomezuniv.github.io/td_rtfm

Monolithic system

Single Reponsability Principle

12 principle

SOLID :

- Single Responsability Principle
- Open/Close principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inverstion principle

Le Modèle de Maturité de Richardson

- architecture monolithique
- définition des micro-services
- integration
- découpage d’un monolithe
- déploiement
- monitoring
- scaling