Internet Applications Design and Implementation

(Lecture 2 - Software Architecture)

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)

(with previous participations of Jácome Cunha (jacome@fct.unl.pt) and João Leitão (jc.leitao@fct.unl.pt))

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Outline

- Software Architecture - Introduction
- Software Architecture for Internet Applications
  - Three-tier architecture
  - Service-based architectures
  - Microservice-based architectures
- Frameworks at the service of Software Architecture
- The architectural style REST to instantiate webservices
- Specifying webservices with OpenAPI and Spring

Internet Applications Design and Implementation

(Lecture 2, Part 1 - Software Architecture - Introduction)

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)

(with previous participations of Jácome Cunha (jacome@fct.unl.pt) and João Leitão (jc.leitao@fct.unl.pt))

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Software Architecture

Introduction

based on the book “Software Architectures in Practice”

Software Architecture

- What is software architecture?
- What are the benefits of using one?

Software Architecture

- What is software architecture?

  “The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.” (in Software Architectures in Practice)

- Structures can be:
  - the module decomposition structure, which divide computational responsibilities and work assignment (among teams). Identify modules or components
  - runtime structures (connectors) that deal with the communication between components (eg services)
  - organisational structures. How components are developed, tested, deployed

Software Architecture…

… is a form of Abstraction
(different views over the same system)

… is in every software system

(from caos to tidy)

… includes Behaviour
(names and connectors have semantics)

Not All Architectures Are Good Architectures

Software Architecture

What is it? Why do we need it?

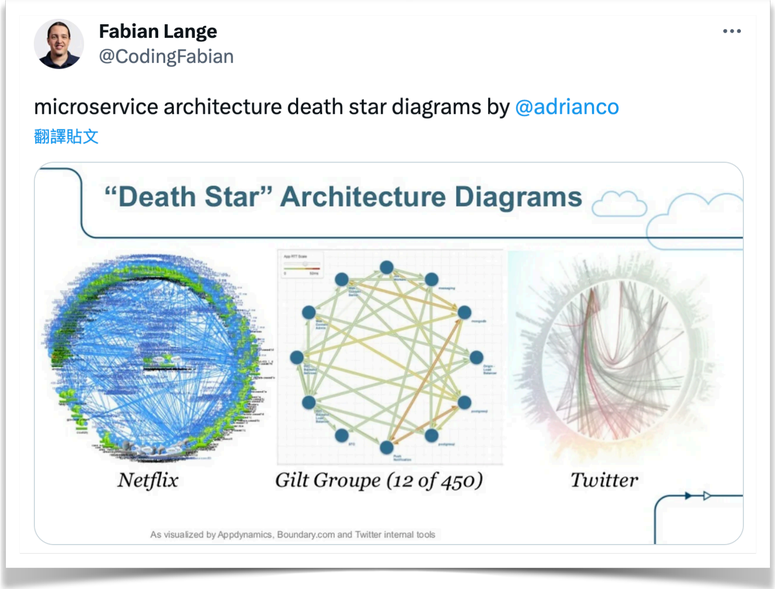
Good organisation of the internal structure of systems leading to:

- Productive development
- Easy evolution
- Easy maintenance
- Trustworthiness (Correctness, Security)
- High Performance Level

Software Architecture (Structures)

- Module structures
  - What is the primary functional responsibility assigned to each module?
  - What are the dependencies to other software elements?
  - What modules are each module related to? by generalisation or specialisation.
- Component and connector structures
  - How do modules interact?
  - What are the shared data stores?
  - What is the data flow in the system?
  - Can the structure change? how?
  - What are the security requirements? performance bottlenecks?

(slides)



- Allocation structures
  - What is the hardware/cloud infrastructure? module ownership? which are the regressions tests? etc.

Break Down

Partition is related to

Abstraction/Reuse

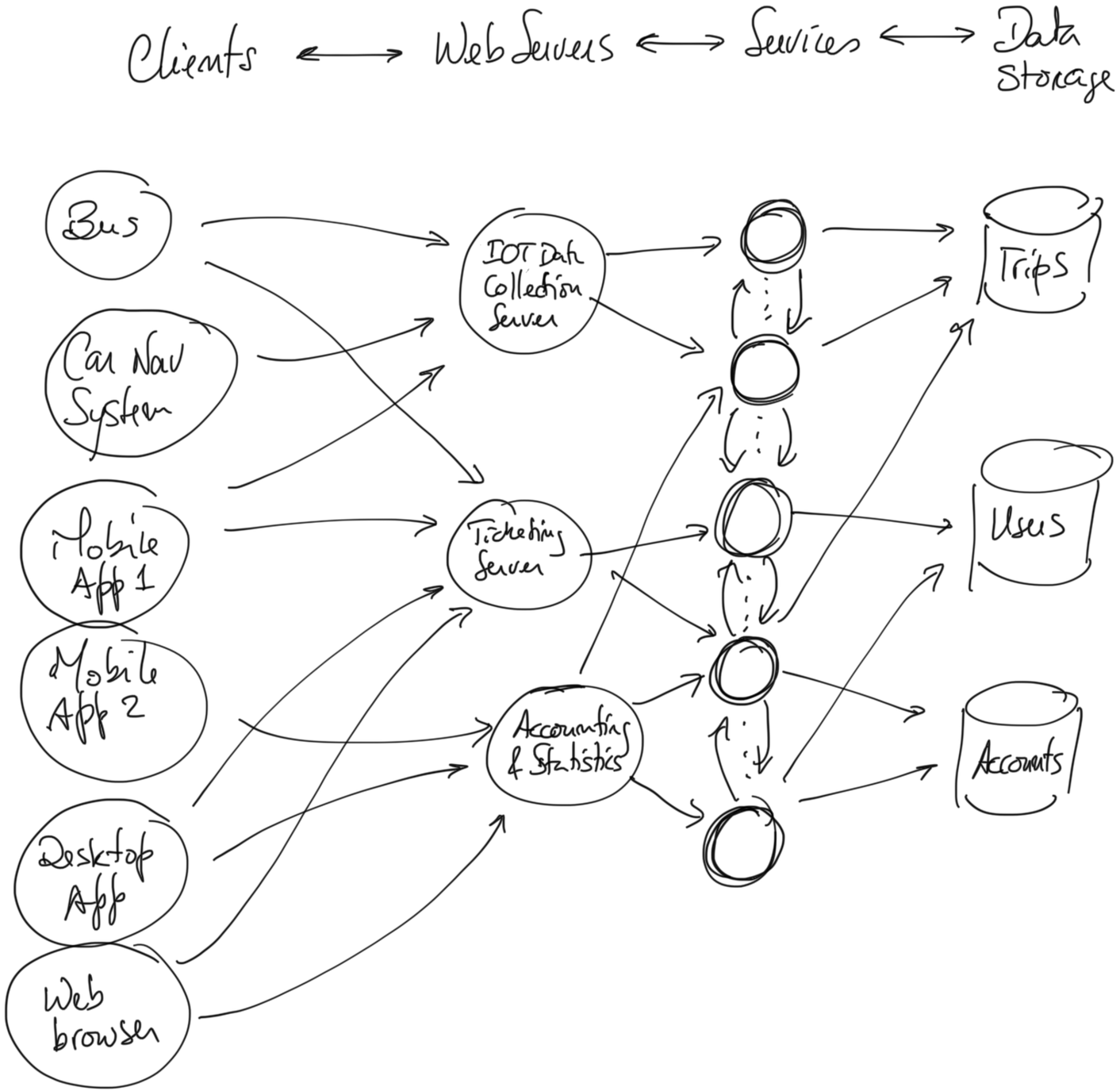
Different Concerns

Complexity

Technology

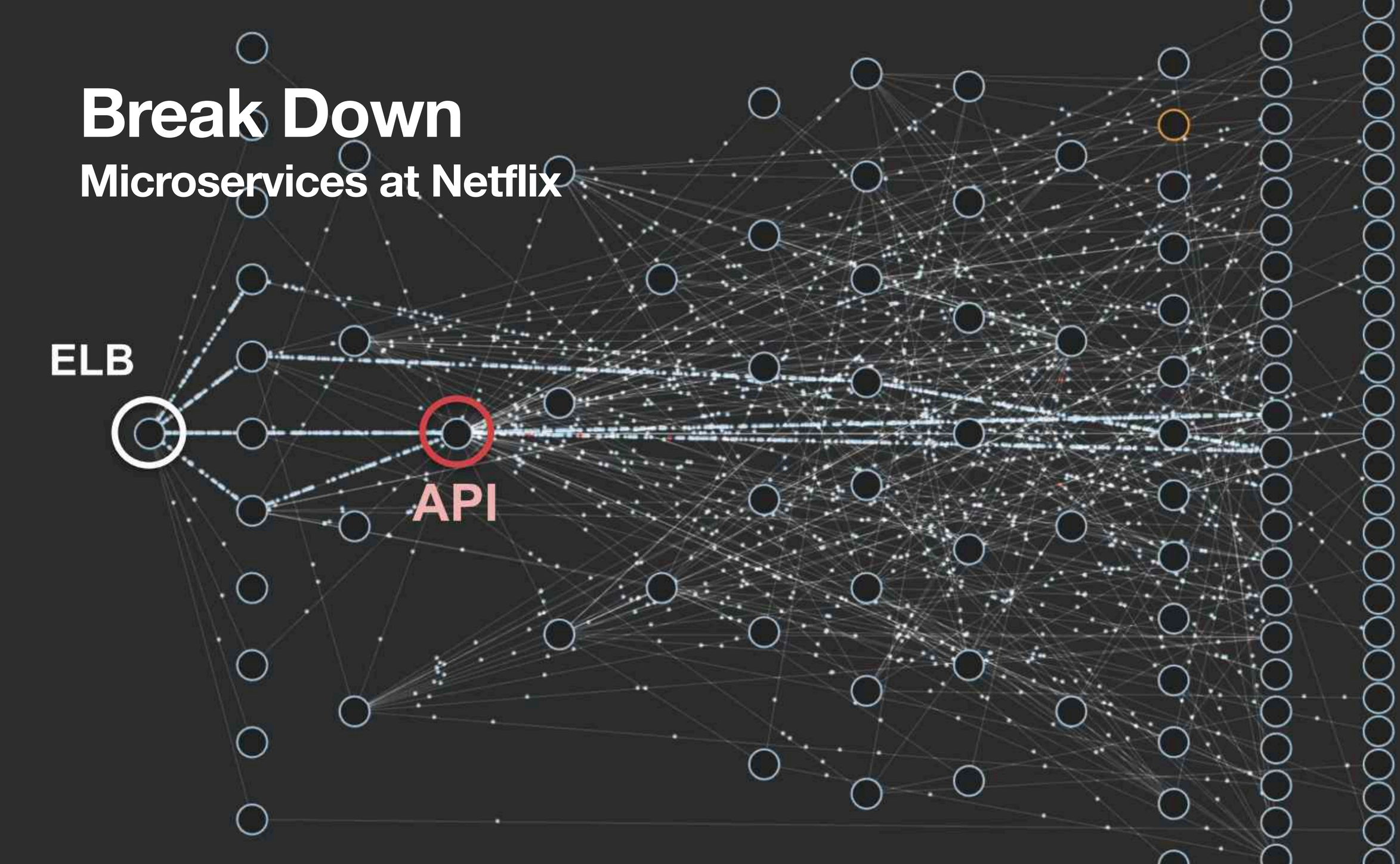
Ownership

Performance



Break Down

Microservices at Netflix



Architectural Patterns

- Pre-determined compositions of architectural elements provide strategies for solving common problems in software systems.
- Examples:
  - Layered pattern — Linear (unidirectional) dependencies between multiple elements
  - Shared-data pattern — Components and connectors to create and manipulate persistent data, connectors are languages like SQL.
  - Client-server pattern — Components: Clients and servers; Connectors: protocols and languages
  - Multi-tier pattern — A generic deployment structure of components in different infrastructures
  - Competence center — Work assignment division by expertise (eg. departments)

Internet Applications Design and Implementation

(Lecture 2, Part 2 - Software Architecture - Internet Applications)

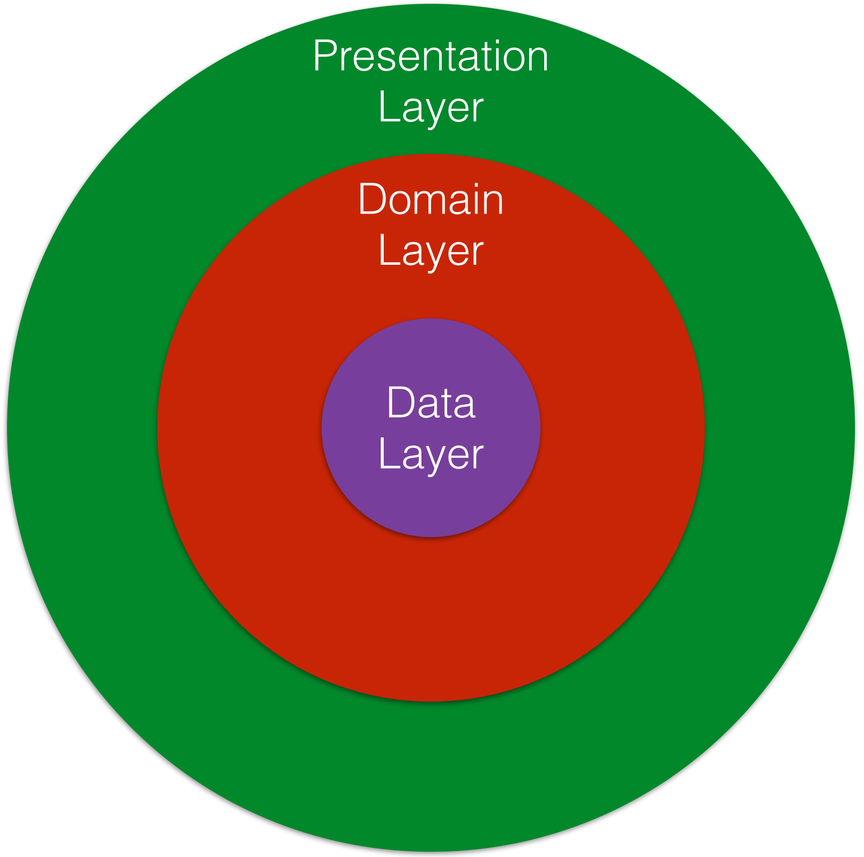
MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)

(with previous participations of Jácome Cunha (jacome@fct.unl.pt) and João Leitão (jc.leitao@fct.unl.pt))

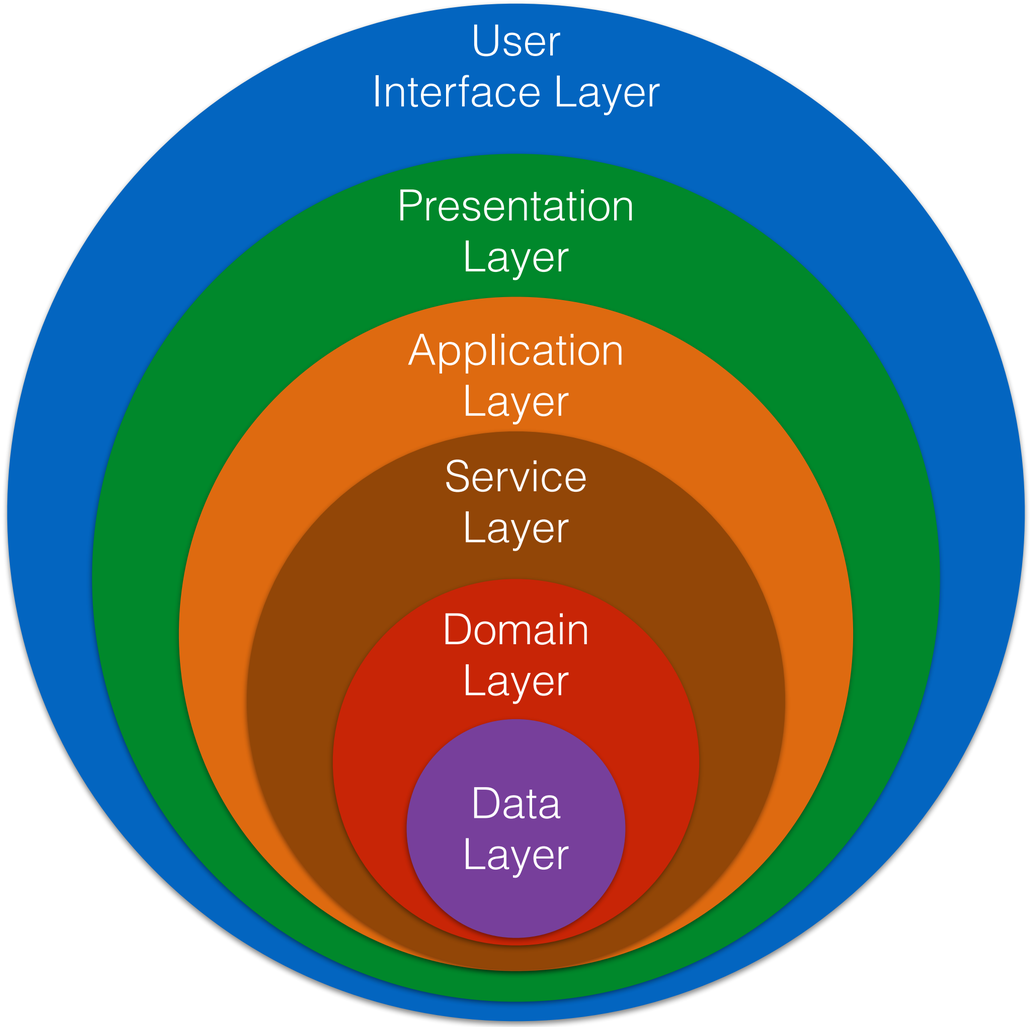
NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Internet Applications are Data-Centric



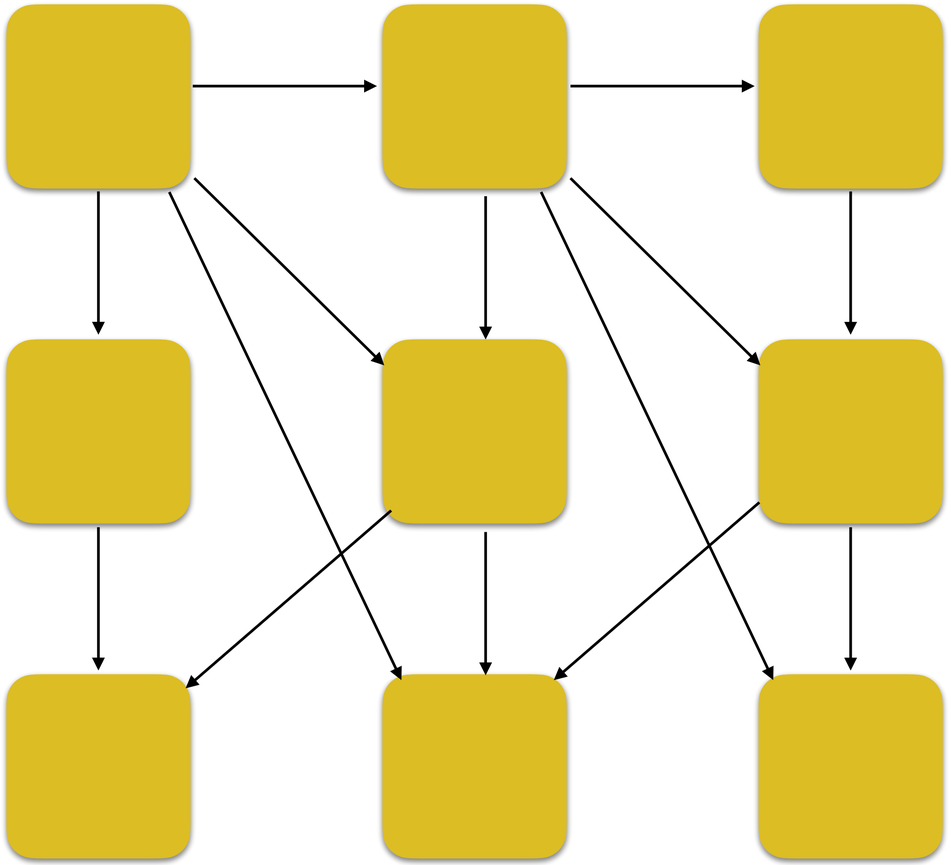
Patterns of Enterprise Application Architecture

Internet Applications are Data-Centric



https://dzone.com/articles/layered-architecture-is-good

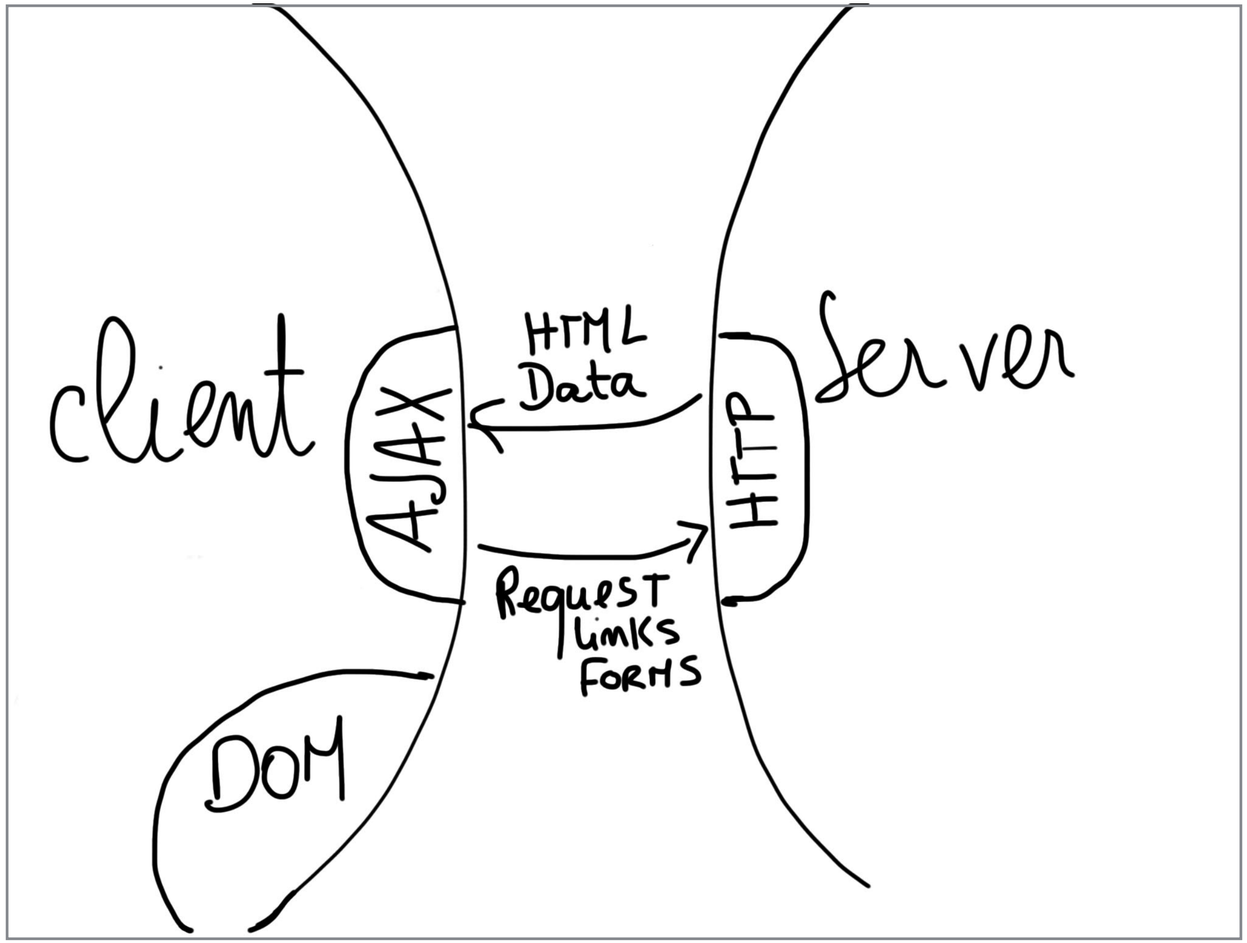
Internet Applications are also Decentralised



https://dzone.com/articles/introduction-to-microservices-part-1

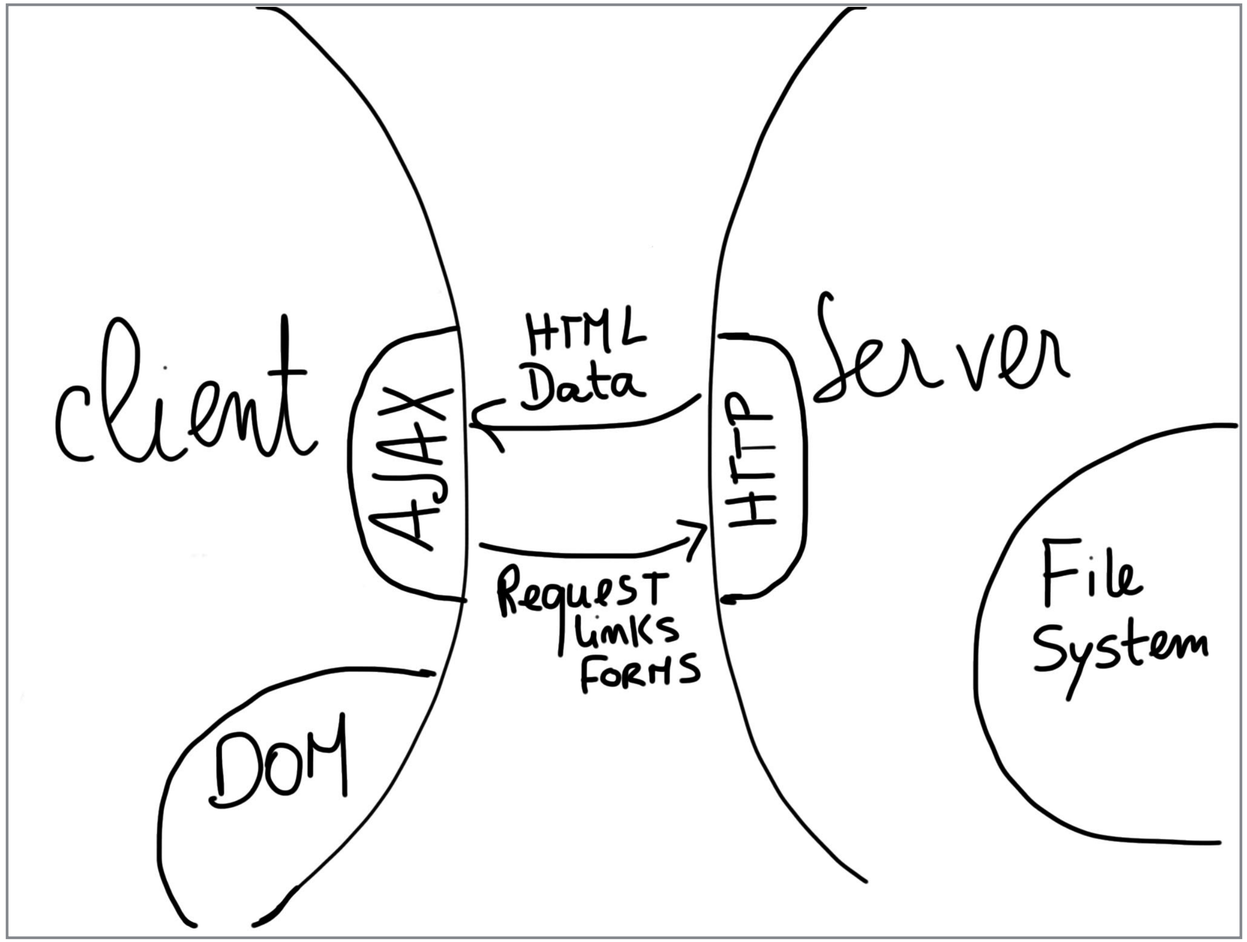
(Logical) Architecture of Web applications

- Tech Details: Interconnection based on the HTTP protocol
  - Method, path, arguments, (a)synchrony, multi-typed response



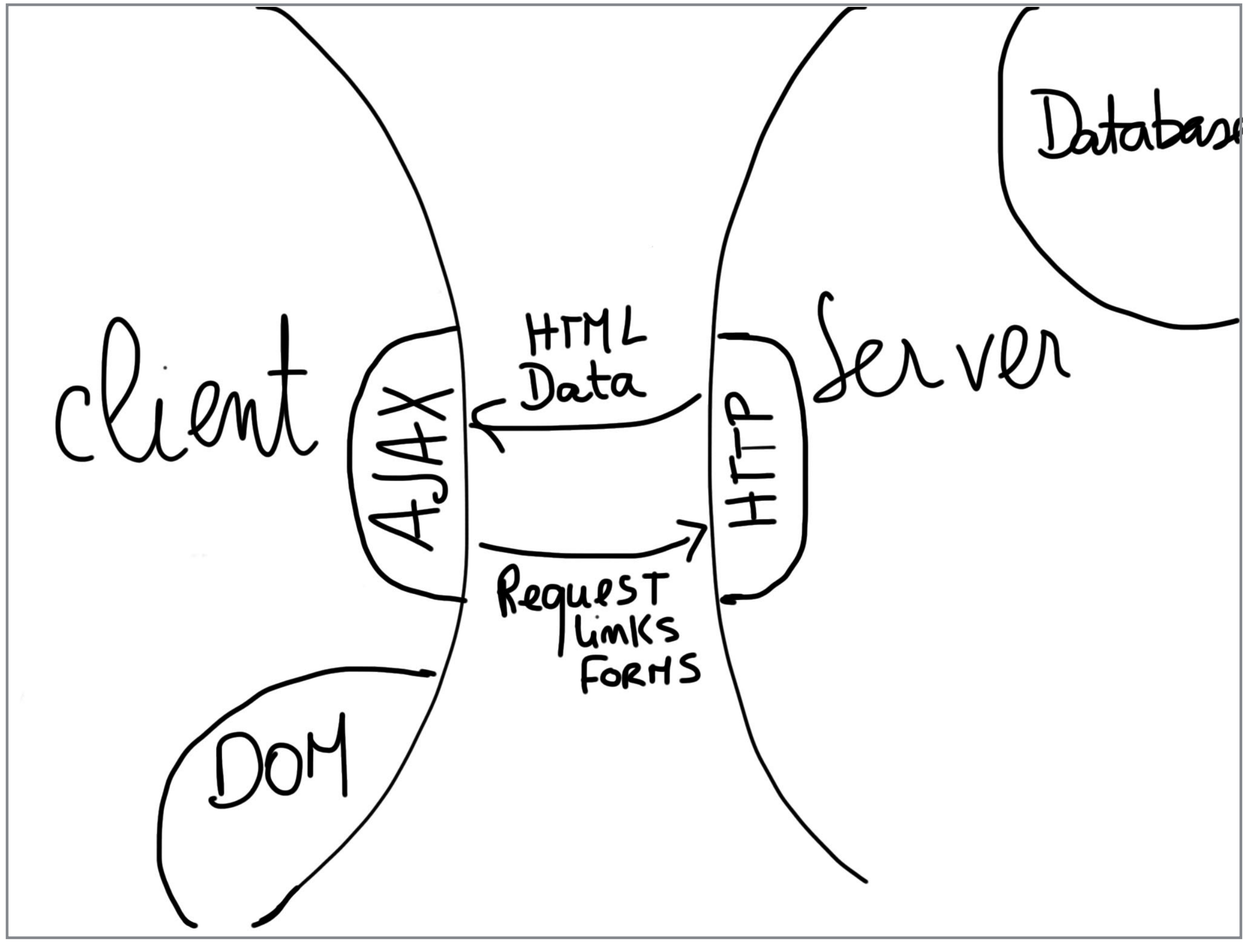
Architecture of Web applications

- Static HTML pages stored in the file system



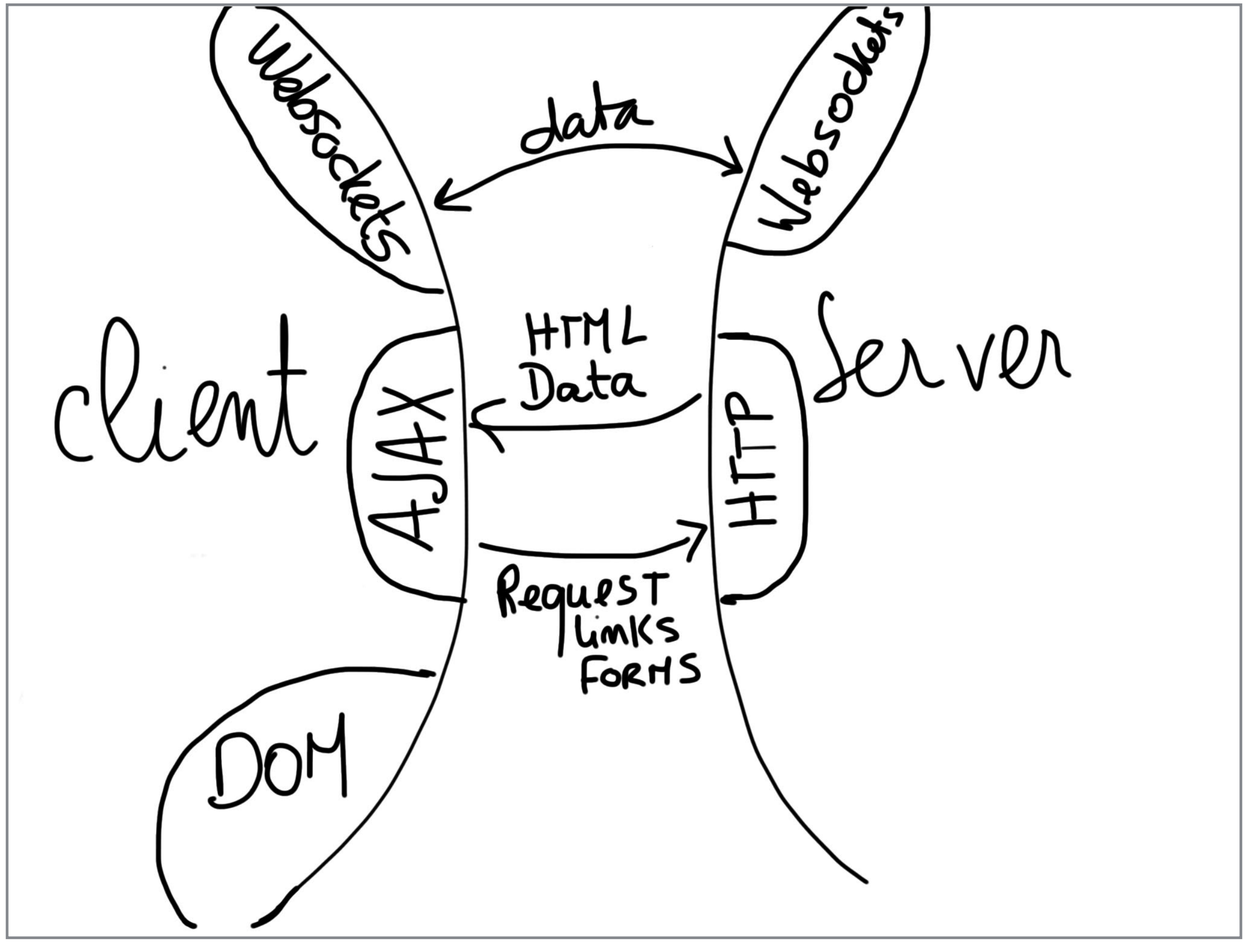
Architecture of Web applications

- Dynamic HTML pages based on data stored in some database.

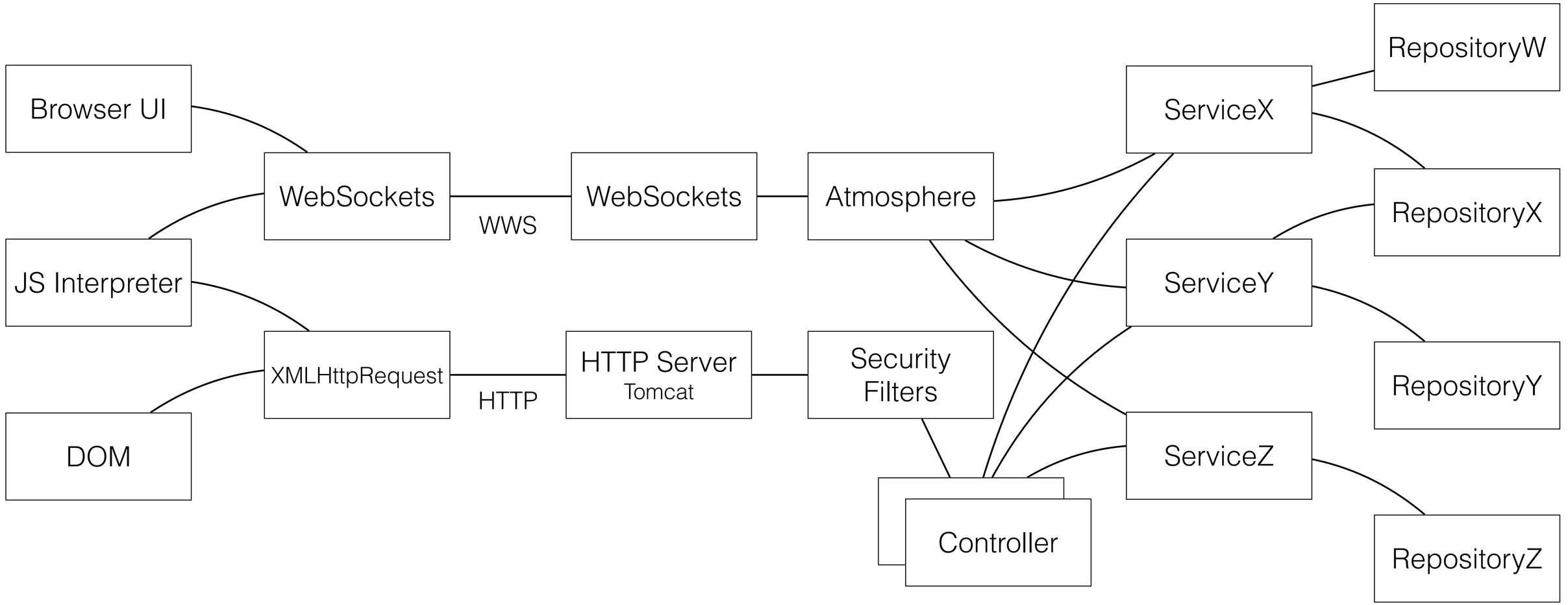


Architecture of Web applications

- Dynamic HTML pages with permanent connection with the server to exchange data.

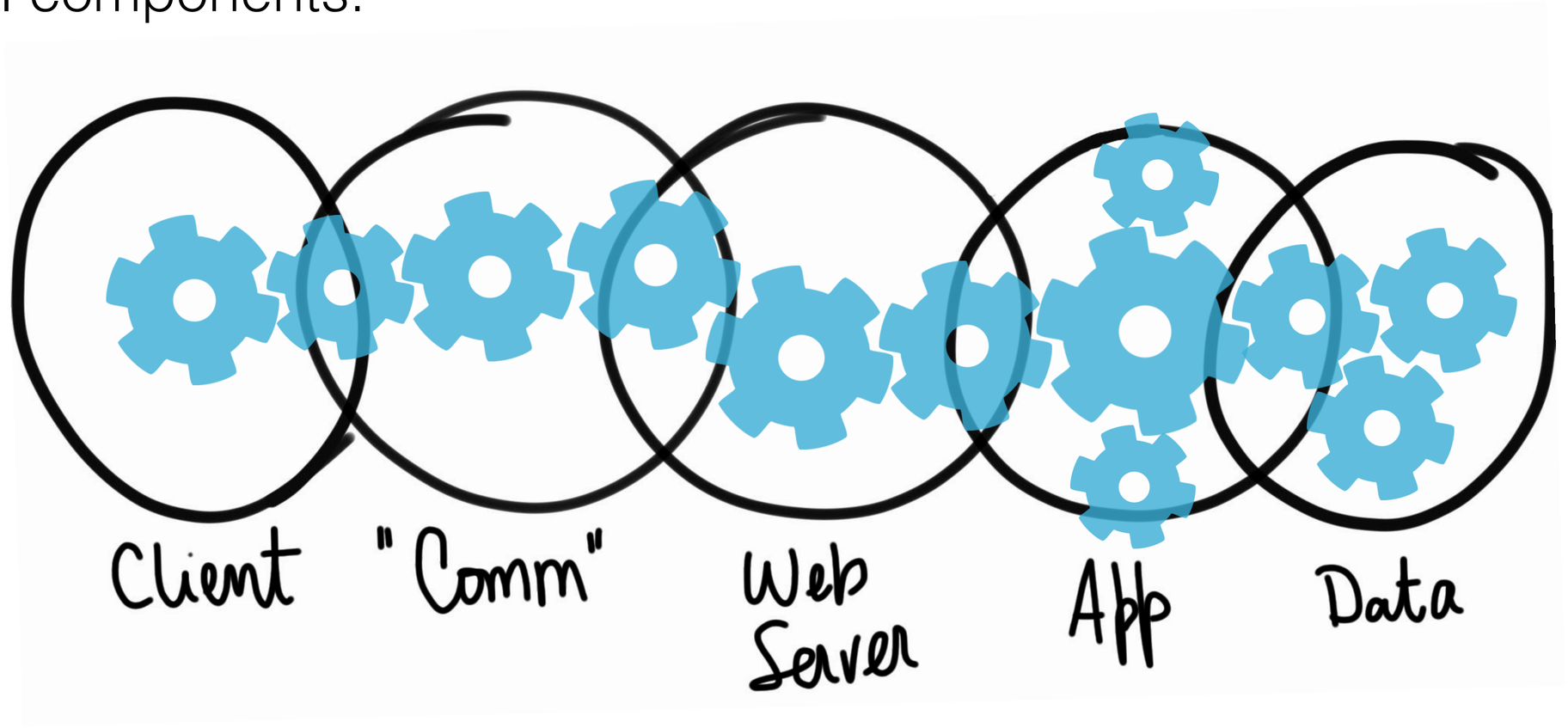


Architecture of Web applications



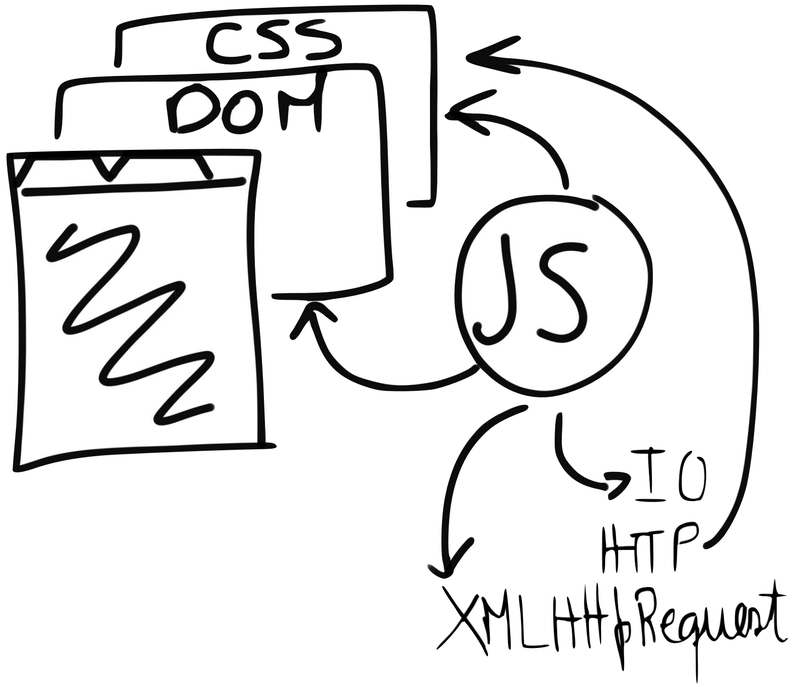
Architecture of Web applications - The Big Picture

- A web application is made of code deployed to the independent and different physical components.



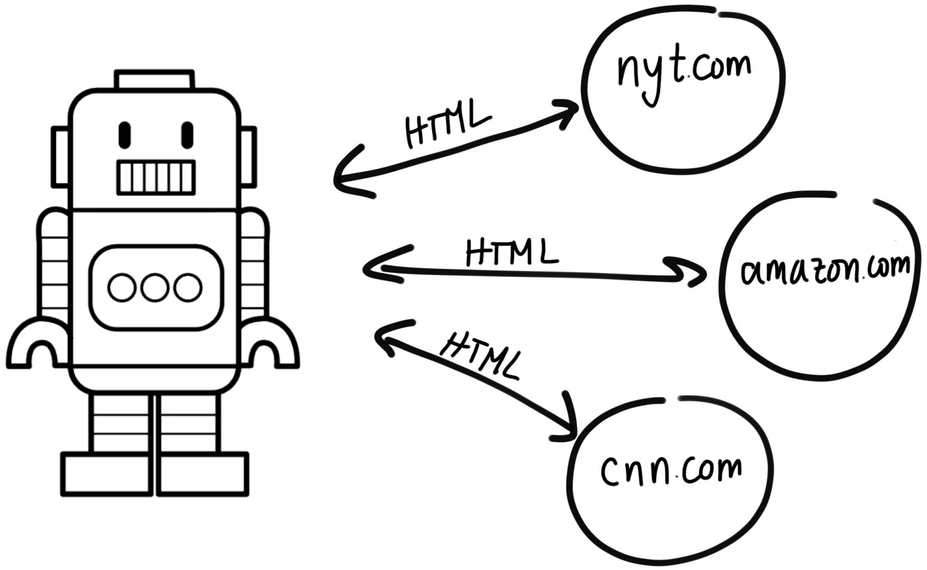
Web client architecture

- Browser (HTML5)
  - HTML (structure and semantics)
  - CSS (style and UI behaviour)
  - JS + AJAX,
    Socket interfaces (behaviour)
  - DOM
    (the supporting data structure)
  - UI Events & callbacks
    (mechanism for dynamic structuring of behaviour)



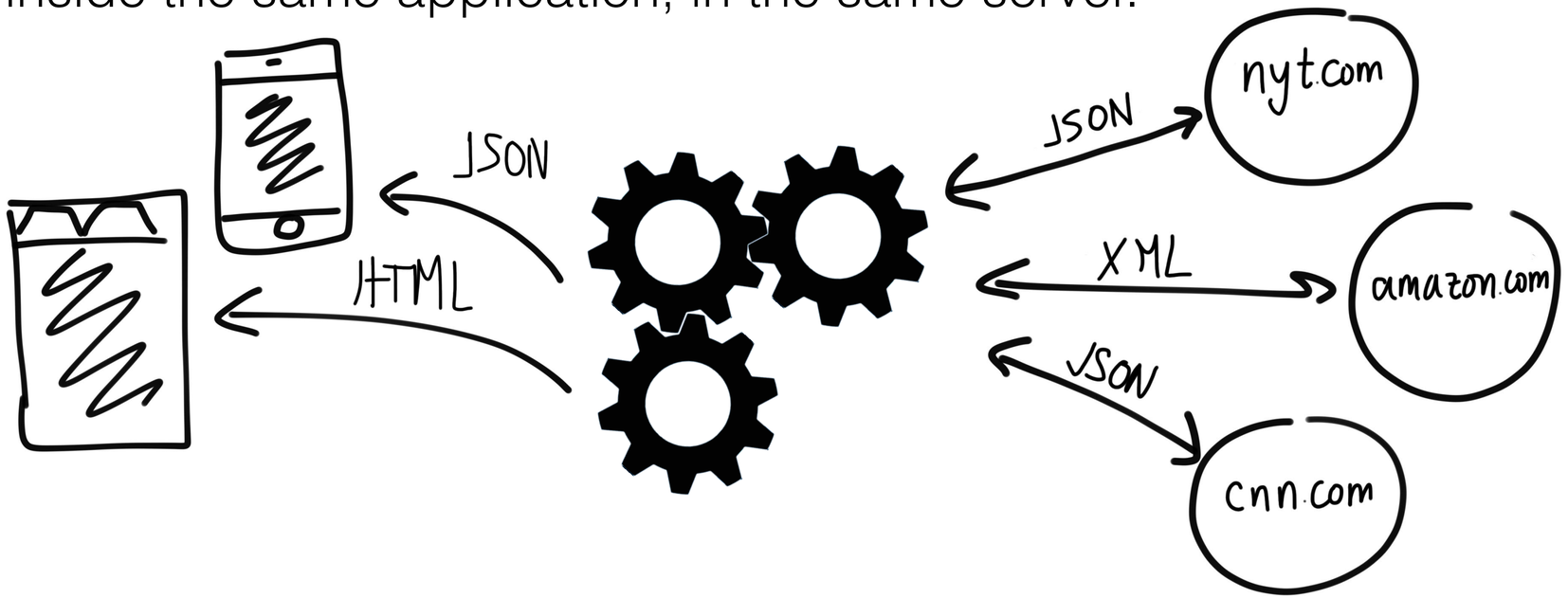
Web client architecture

- Other kind of web clients (asking for HTML)
  - simulate web requests and sessions with web-servers
  - parse/crawl the results to extract information
  - should be built with web-services instead (if possible).



Service based web client architecture

- Provide web content based on data providing services
  - Data can be exchanged in “machine-friendly” formats.
    (e.g. JSON, XML)
  - Combined and refurbished in HTML or data formats.
  - Even reused inside the same application, in the same server.



Interconnection layer (low-level support)

- HTTP/1.1 (*Hypertext Transfer Protocol*) Protocol
  - Method: GET, HEAD, POST, PUT, DELETE (3 more)
  - Arguments (query string and body)
  - Multi-typed message body
  - Cookies
  - Return codes: 1XX, 2XX, 3XX, 4XX, 5XX
- Websockets (standard RFC6455 2011 - ws:// wss://)
  - supported by browsers and web servers to allow two way data communication between client and servers
- HTTP/2 approved May 2015.
  - Faster, compressed, and cyphered transmission of data
- TLS (successor of SSL)
    - Provides cryptographic support for web communications (handshake+symmetric crypto)

1×× Informational
100 Continue
101 Switching Protocols
102 Processing
2×× Success
200 OK
201 Created
202 Accepted
203 Non-authoritative Information
204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status
208 Already Reported
226 IM Used
3×× Redirection
300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
304 Not Modified
305 Use Proxy
307 Temporary Redirect
308 Permanent Redirect
4×× Client Error
400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Payload Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot
421 Misdirected Request
422 Unprocessable Entity
423 Locked
424 Failed Dependency
426 Upgrade Required
428 Precondition Required
429 Too Many Requests
431 Request Header Fields Too Large
444 Connection Closed Without Response
451 Unavailable For Legal Reasons
499 Client Closed Request
5×× Server Error
500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates
507 Insufficient Storage
508 Loop Detected
510 Not Extended
511 Network Authentication Required
599 Network Connect Timeout Error

https://httpstatuses.com/

Web server / App server architecture

- Web servers handle HTTP requests, map URLs to local files, execute local scripts (e.g. CGI, PHP), or locally bound code (e.g. Servlets).
- App Servers modularly manage bound code, and associated resources (e.g. sessions, context, connections).
  - One web server, many applications.
  - Allows the assembly of components of applications (controllers, views, models)

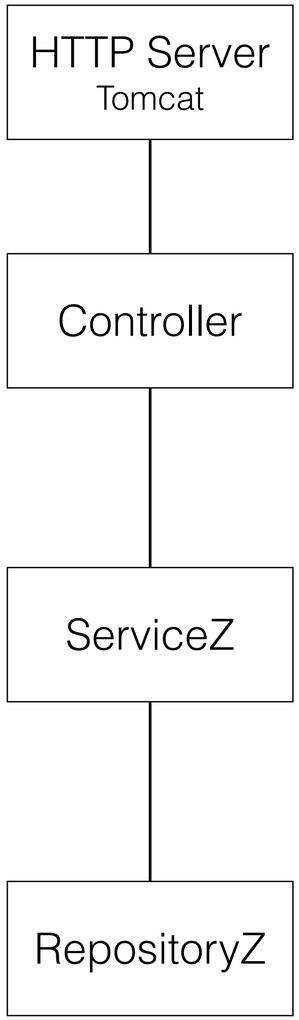
Web architectures, patterns and styles

- Increase the level of abstraction, reusability, and maintainability
  - Software Architectures
  - Architectural and design patterns
  - Architectural styles

- Software frameworks implement some, and complement with libraries and tools.
- User-defined pieces are required to specify the “core” logic, and configure general purpose code.
- All implement the “inversion of control” pattern.

An architecture built with Spring

Example of an Architecture built with Spring



- Spring is a component framework
- Resolves component dependencies by dependency injection
- Uses annotations to configure components

```
@RestController
@RequestMapping("/")
class EmpController(val employees:EmployeeService) {

   // http GET :8080/api/projects/2/team
    @GetMapping( "/api/projects/{id}/team")
    fun teamMembersOfProject(
            @PathVariable id:String
    )
    = employees.teamMembersOfProject(id)


}

@Service
class EmployeeService(val employees:EmployeeRepository) {
    fun teamMembersOfProject(id:String) = employees.findAll()
}

interface EmployeeRepository : CrudRepository<Employee, Long>
```

Service Based Architectures

Service Oriented Architectures

- Are the technological and methodological basis for building open-ended Internet Applications.
- Provide a model of distributed computation based on loosely coupled interactions.
- Define heterogeneous ecosystems of service implementations.
- Use implementation independent data formats (JSON, XML)
- Allows independent development of services by different vendors and technologies
- A service:
  - Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports)
  - Is self-contained
  - May be composed of other services
  - Is a “black box” to consumers of the service

https://web.archive.org/web/20160819141303/http://opengroup.org/soa/source-book/soa/soa.htm

Web architectures, patterns, and styles

- Web services are usually defined over HTTP protocol

- **SOAP** (*Simple Object Access Protocol*)
  - Operation based protocol (on HTTP) to implement web services (XML as format)
- **REST** (*Representational State Transfer*)
  - Resource based architectural style to implement web services over HTTP (or another connection protocol)

Micro Service Architectures

- Are an extreme interpretation of service based architectures.
- Have smaller grained services and interfaces.
- Provide independent and lightweight deployment.
- Allow flexible management (e.g. replicas).
- Based on a clear service ownership model
  (team responsible for all stages of dev&ops).
- Isolated persistent state (sometimes bad).

https://martinfowler.com/microservices/

Amazon's API Mandate (by Jeff Bezos, 2002)

- All teams will henceforth expose their data and functionality through service interfaces.
- Teams must communicate with each other through these interfaces.
- There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
- It doesn't matter what technology you use.
- All service interfaces, without exception, must be designed from the ground up to be externalizeable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
- The mandate closed with: Anyone who doesn't do this will be fired. Thank you; have a nice day!

Internet Applications Design and Implementation

(Lecture 2, Part 3 - Software Architecture - Frameworks)

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

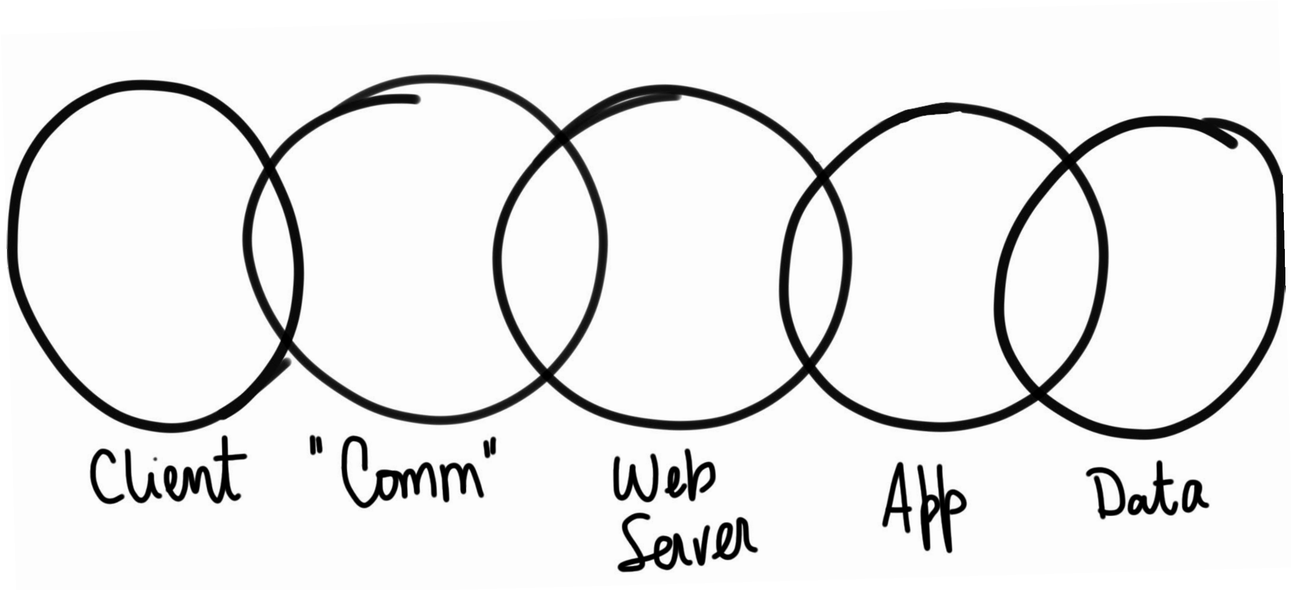
João Costa Seco (joao.seco@fct.unl.pt)

(with previous participations of Jácome Cunha (jacome@fct.unl.pt) and João Leitão (jc.leitao@fct.unl.pt))

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

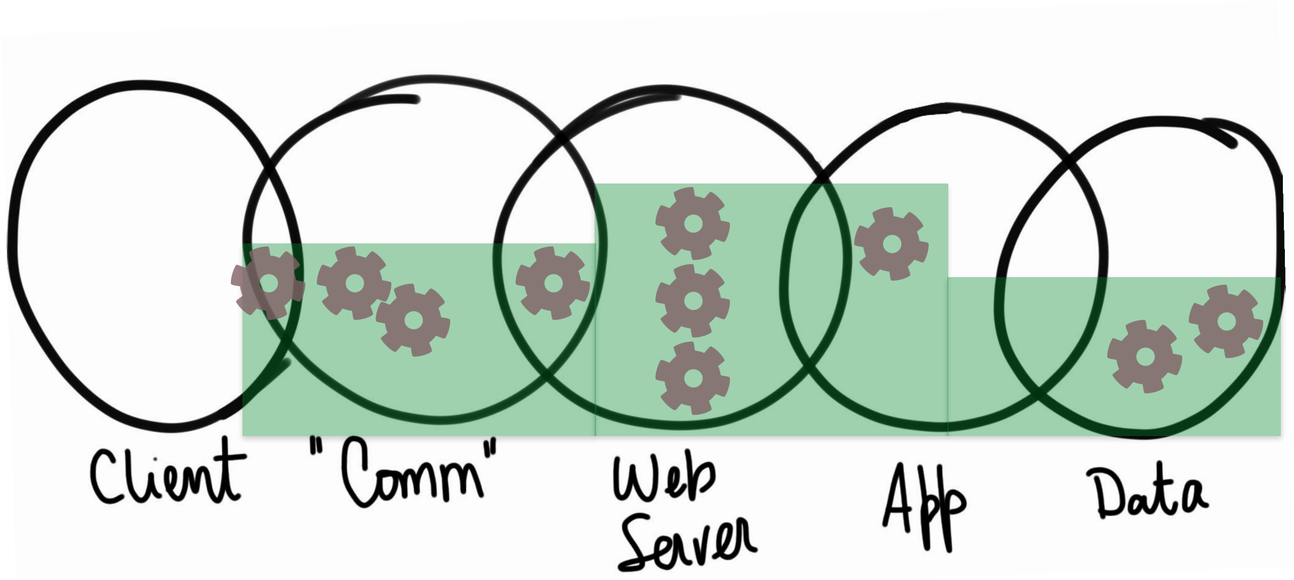
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



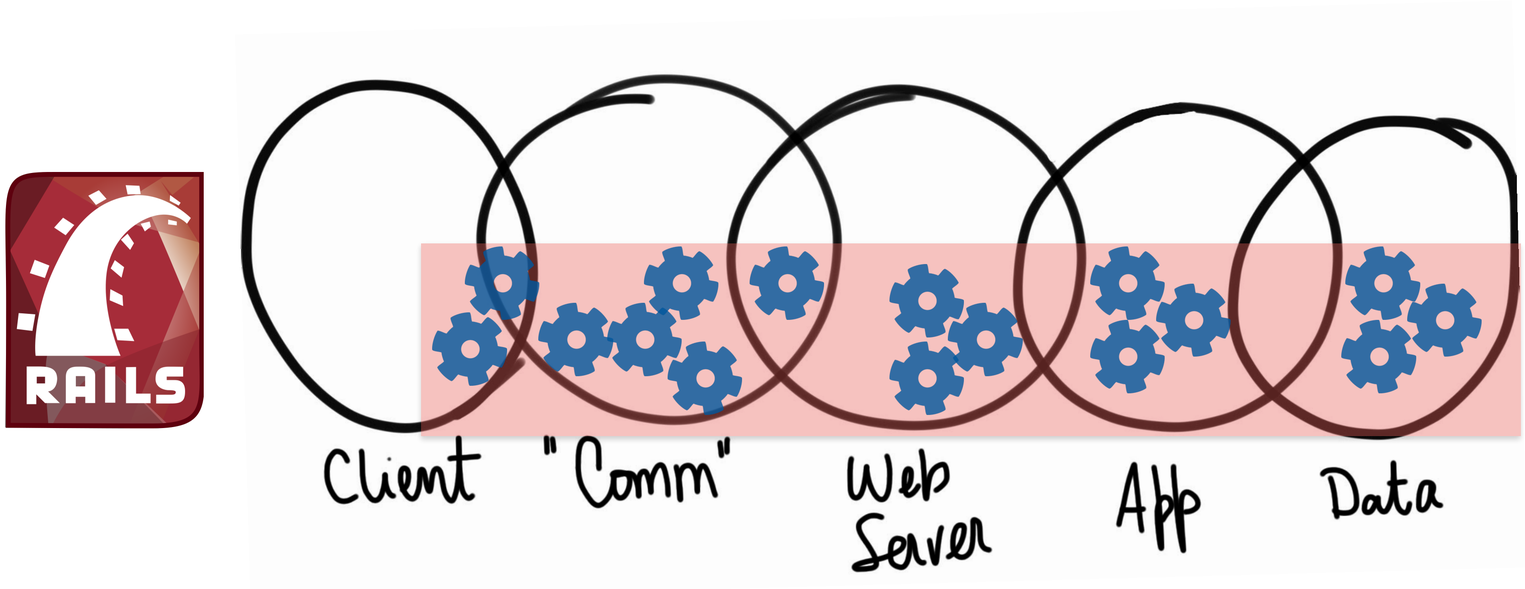
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



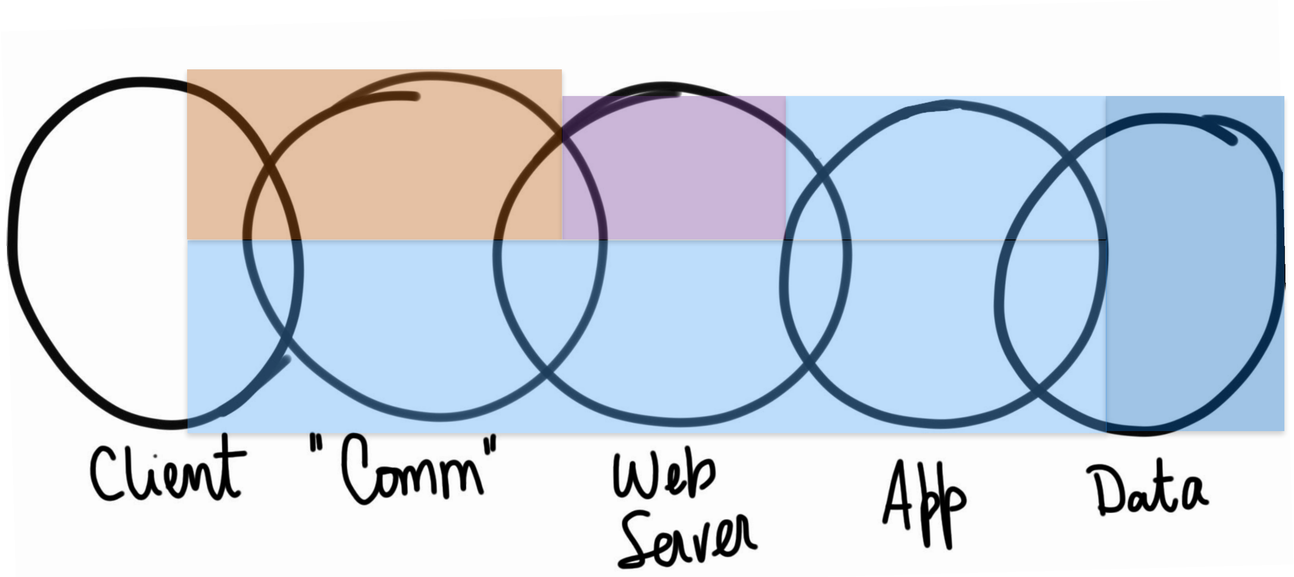
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



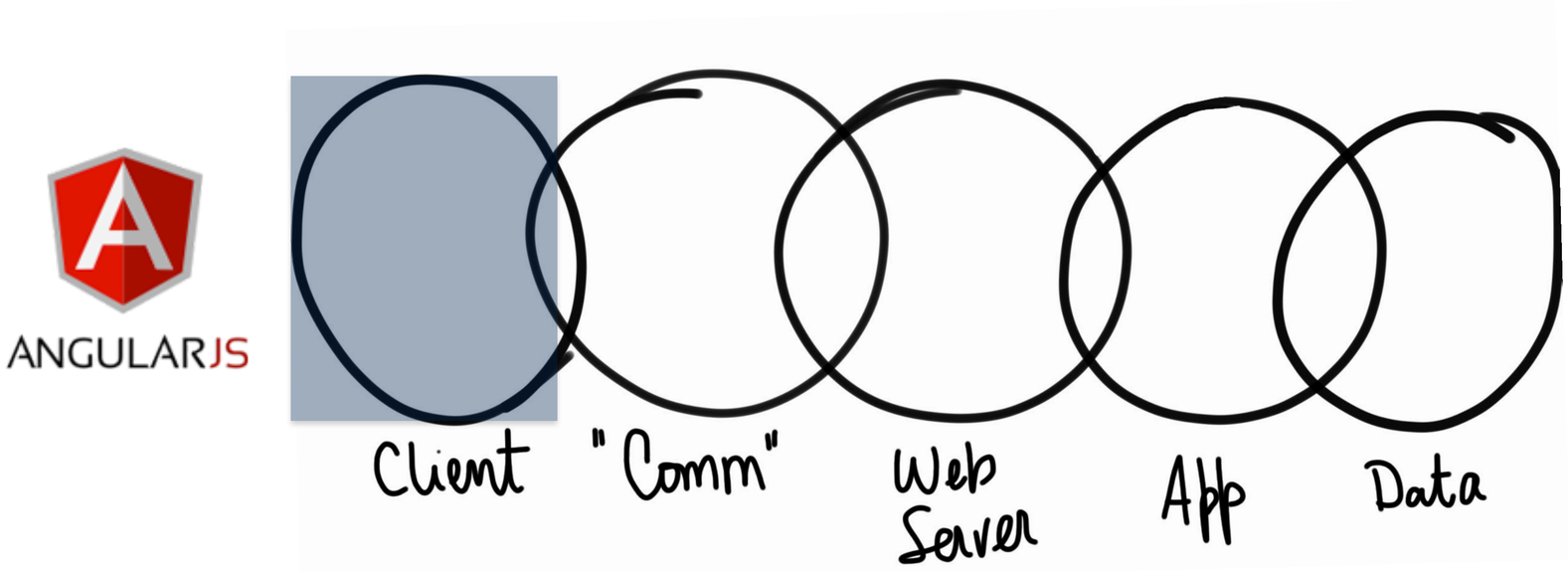
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



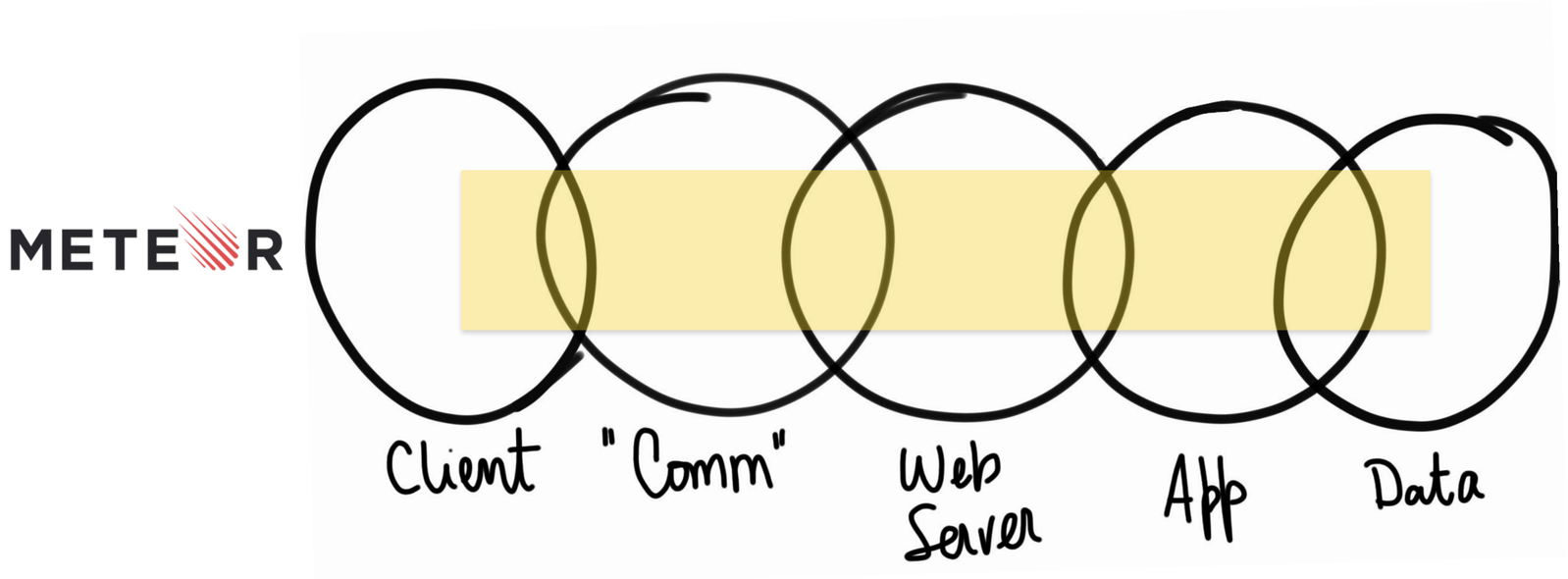
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



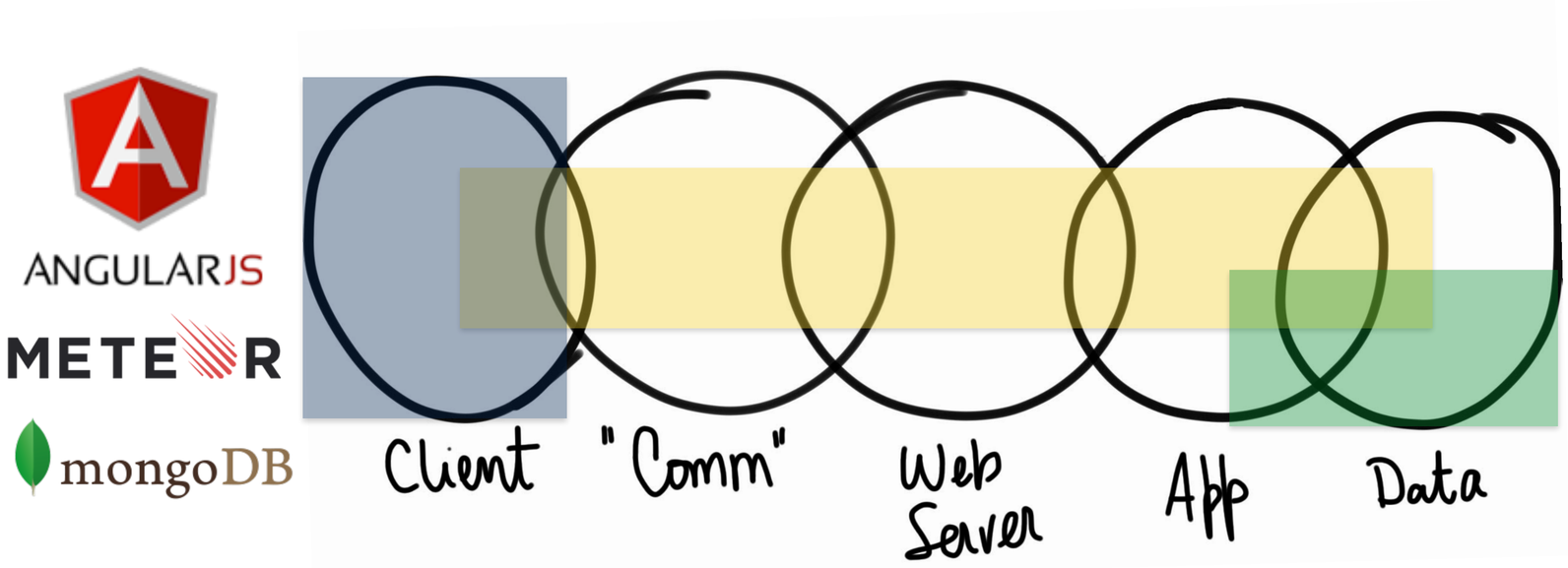
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



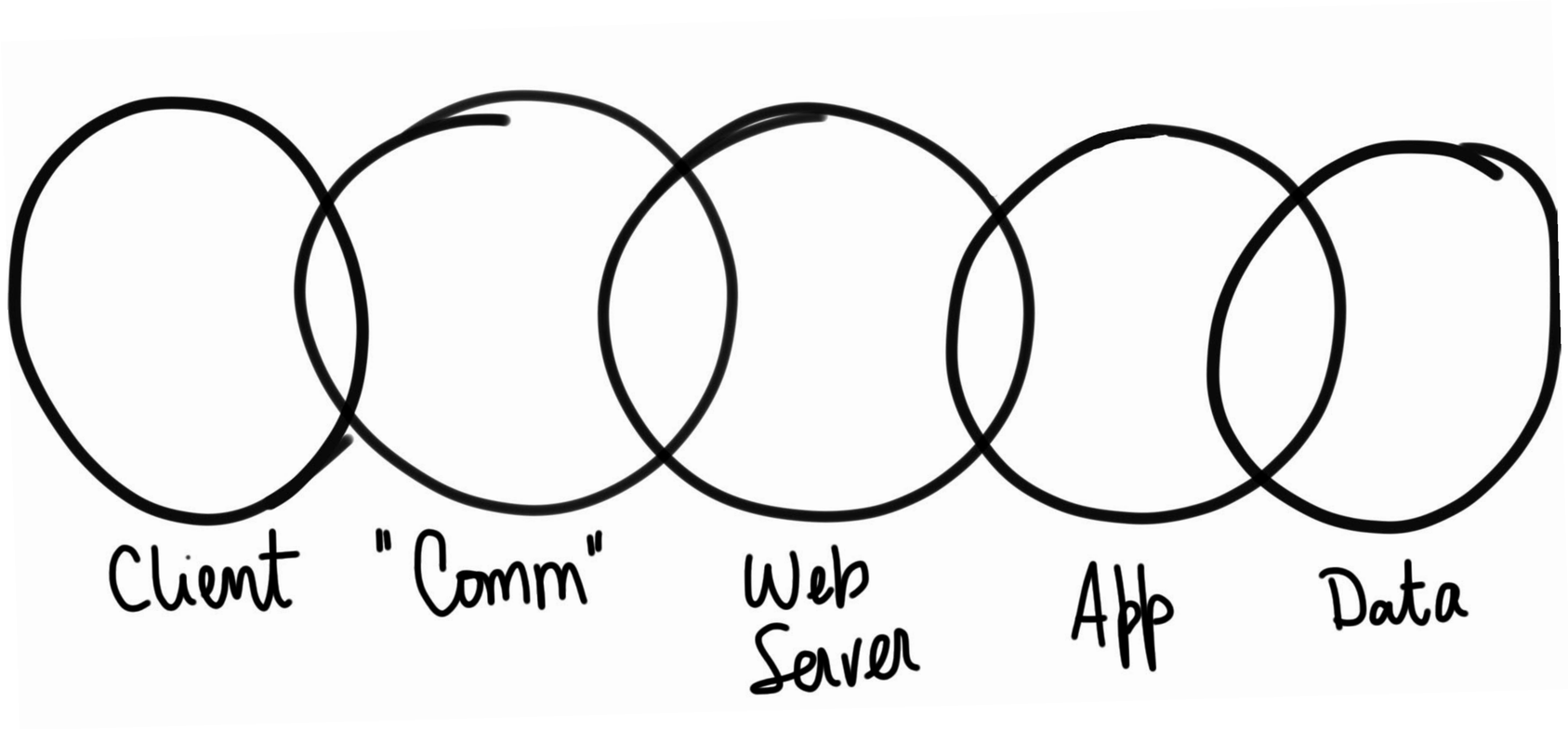
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



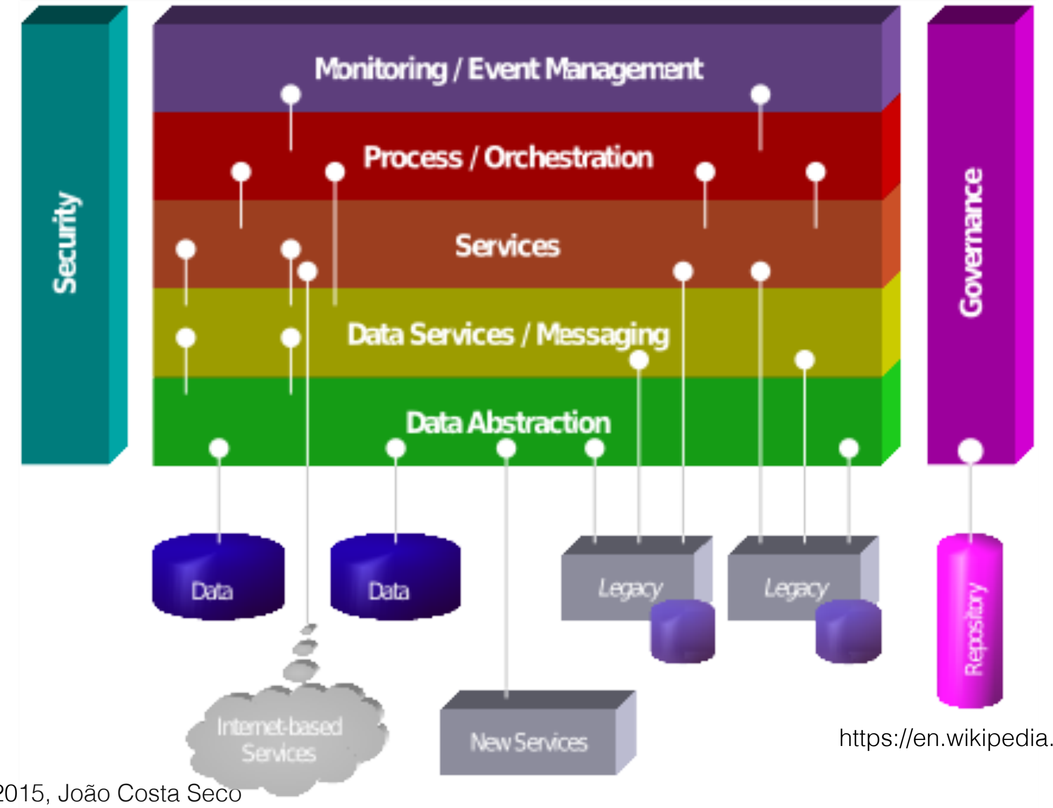
Summary - Web Frameworks

- Web Frameworks are “languages” that carry libraries and abstractions that get compiled to run on the “web virtual machine”.



(web & local) Services

- Service oriented architectures are a way of decoupling implementation from use.
- Services are implemented based on a “contract” or interface and provided by a broker.



https://en.wikipedia.org/wiki/Service-oriented_architecture

Service Orchestration Languages

- Spring Web Flow

```
<?xml version="1.0" encoding="UTF-8"?>
<flow xmlns="http://www.springframework.org/schema/webflow"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/webflow
                          http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd"


        <start-state="welcome">

        <view-state id="welcome" view="/welcome.jsp">
            <transition on="continue" to="finish"/>
            <transition on="cancel" to="cancelled"/>
```

- Jolie

Process Specification Languages

- From processes to code…