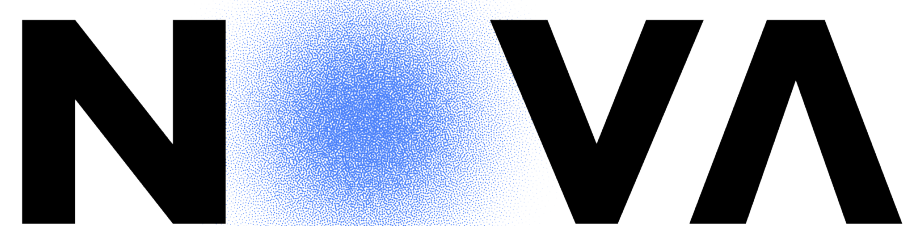


# Internet Applications Design and Implementation

## (Lecture 1 - Introduction and Logistics)

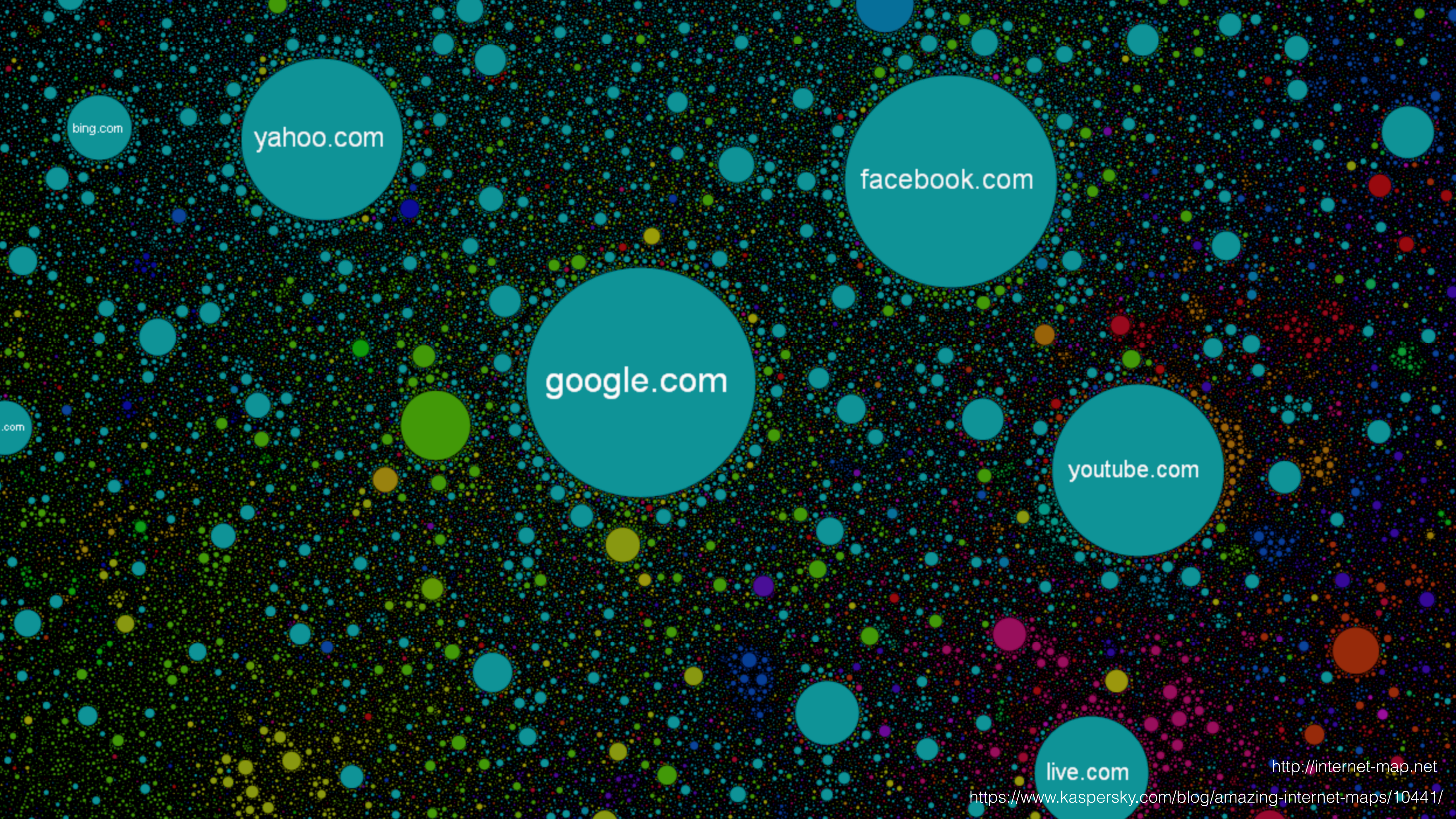
**MIEI - Integrated Master in Computer Science and Informatics**  
**MEI - Master in Computer Science and Engineering**  
**Specialisation block**

**João Costa Seco ([joao.seco@fct.unl.pt](mailto:joao.seco@fct.unl.pt))**



NOVA SCHOOL OF  
SCIENCE & TECHNOLOGY





bing.com

yahoo.com

facebook.com

google.com

youtube.com

live.com

<http://internet-map.net>

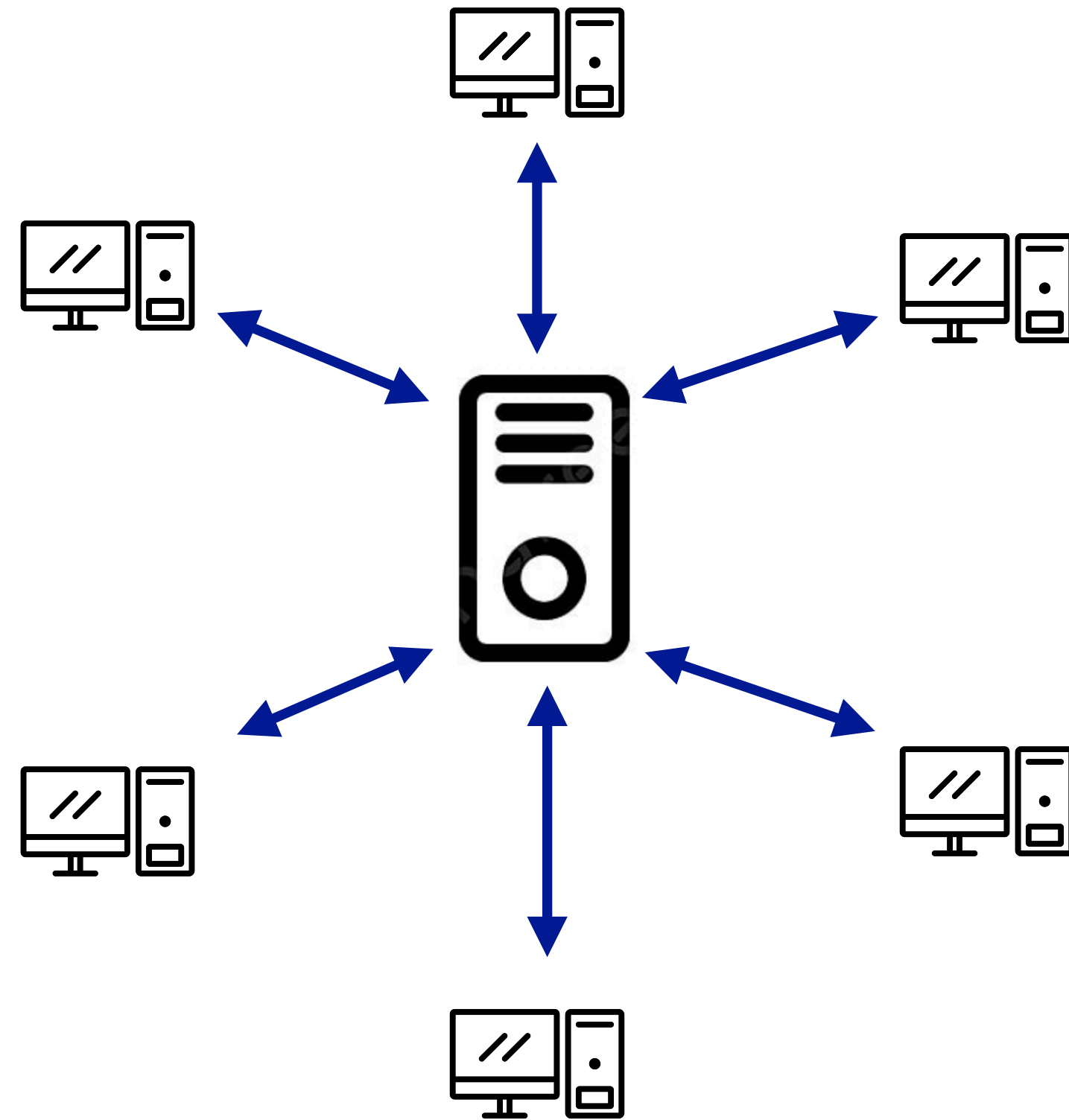
<https://www.kaspersky.com/blog/amazing-internet-maps/10441/>



Standalone

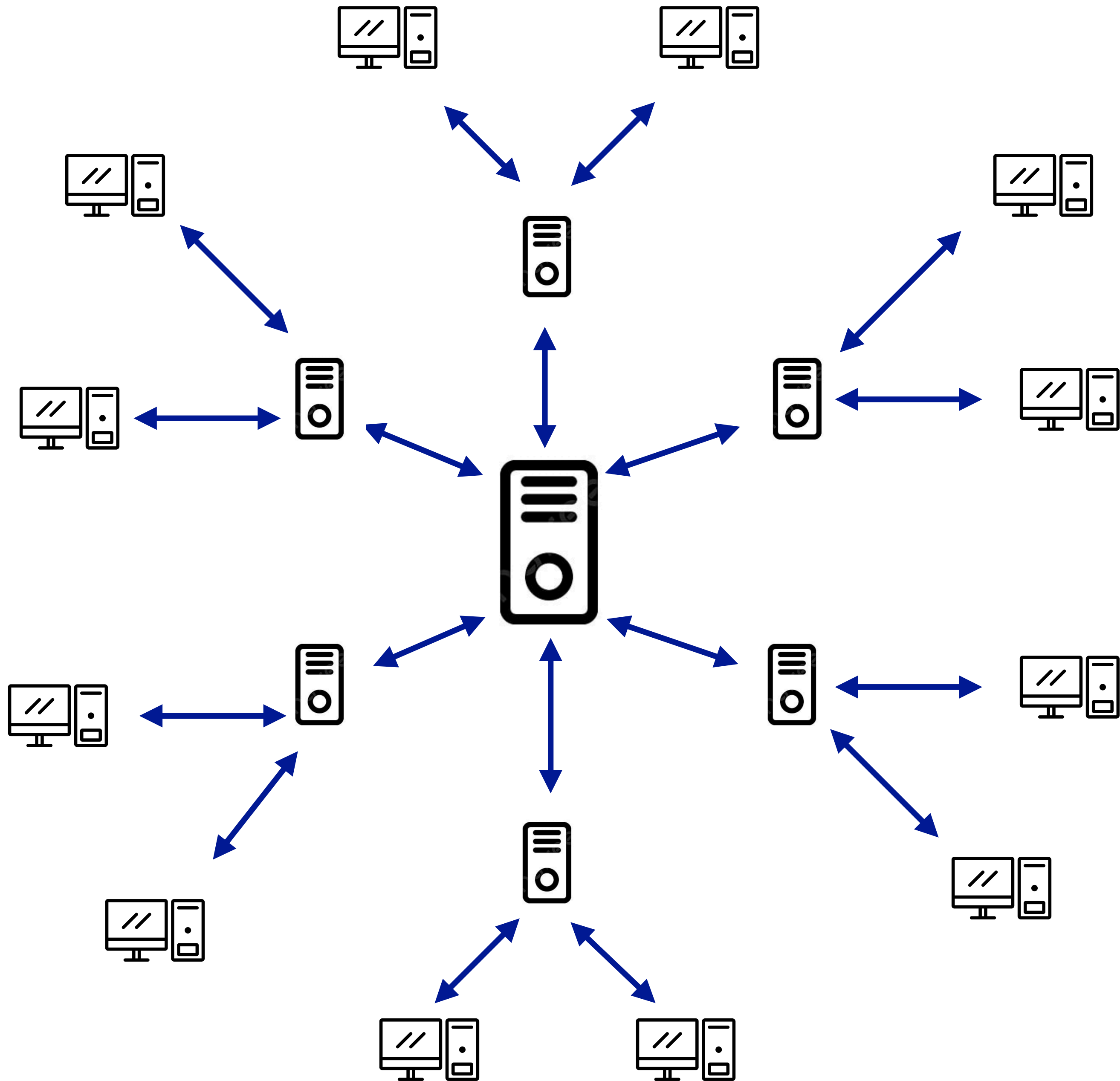


# Client Server



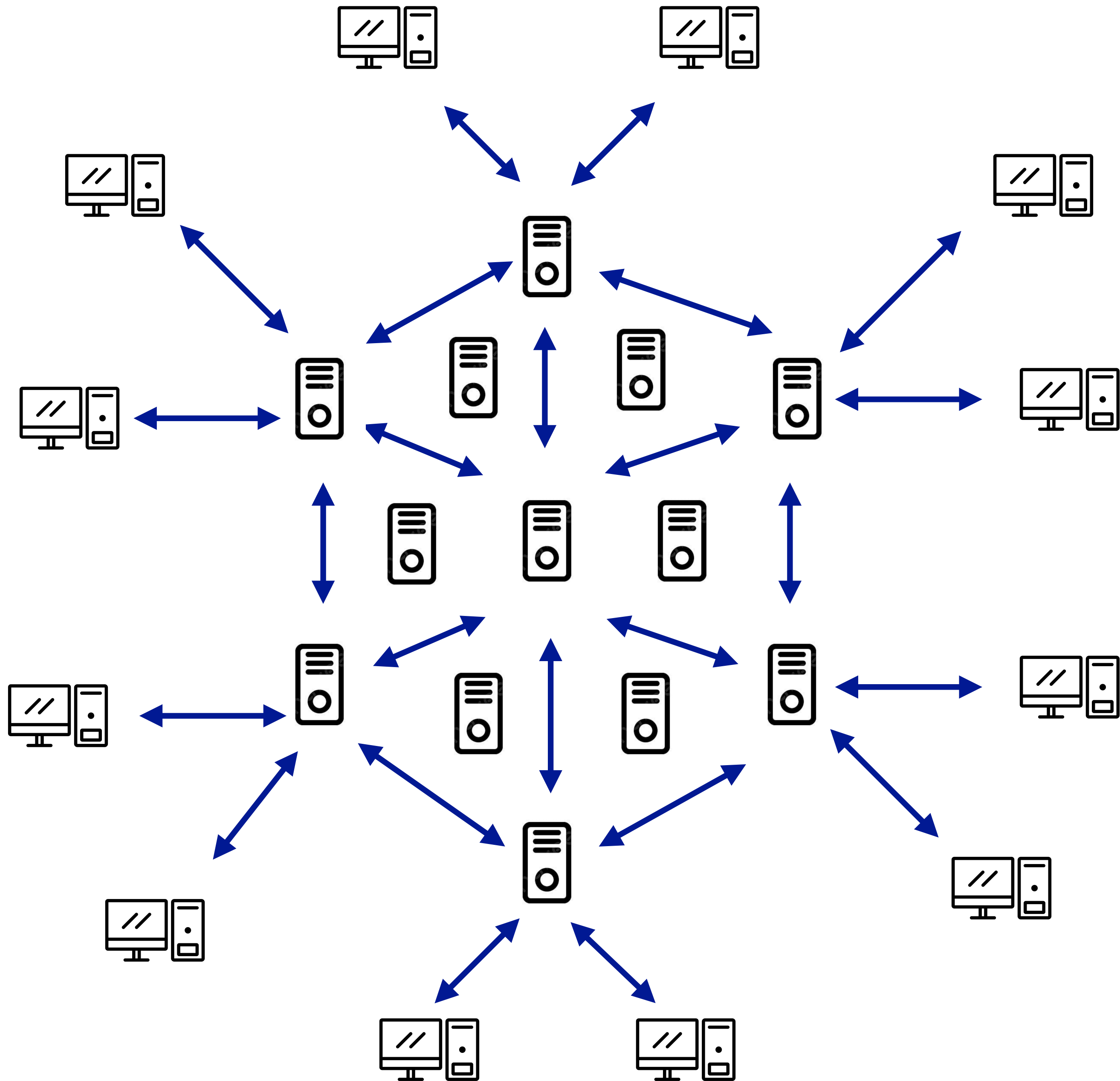


Cloud - Edge



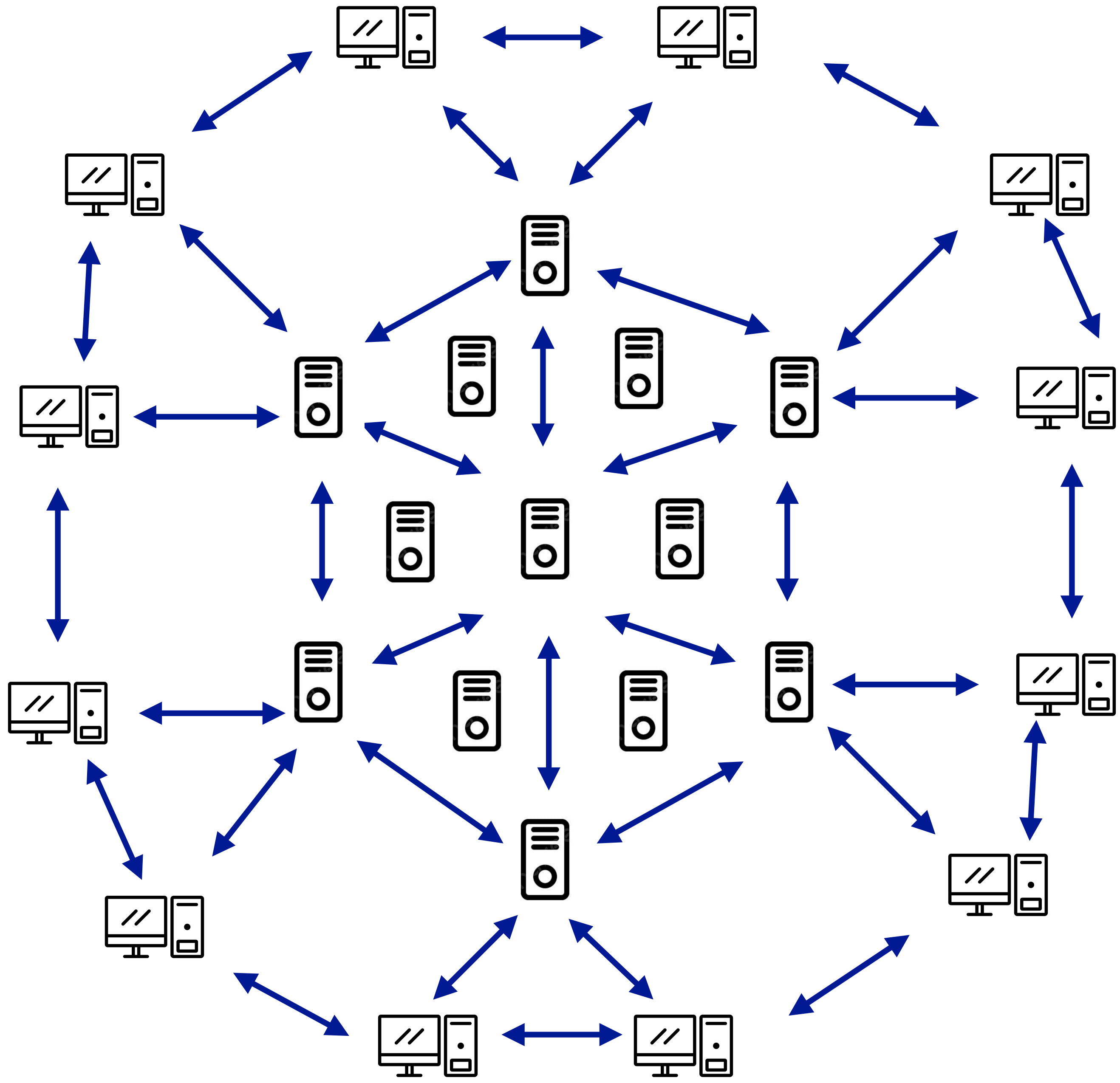


# Micro Services



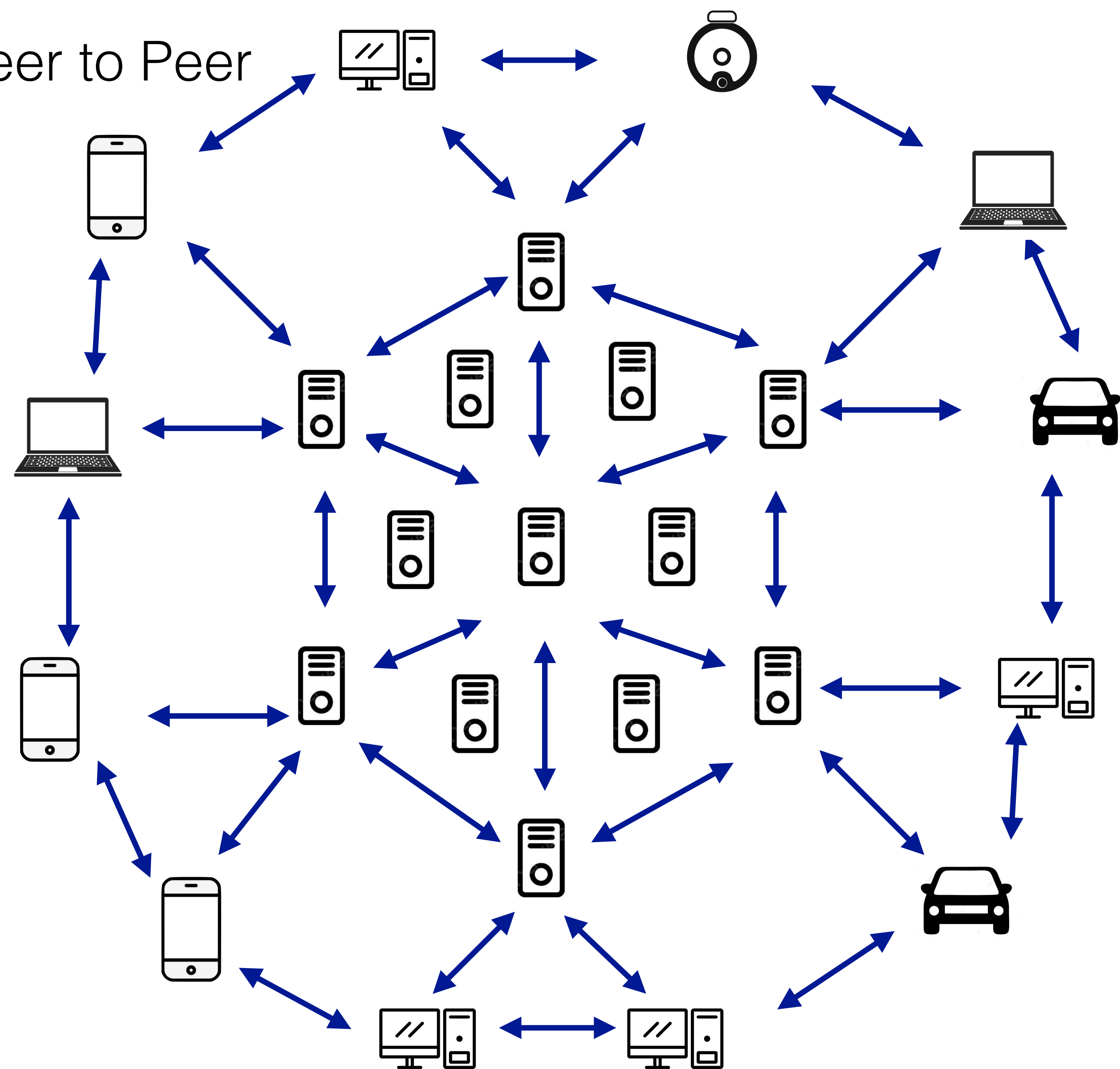


Peer to Peer



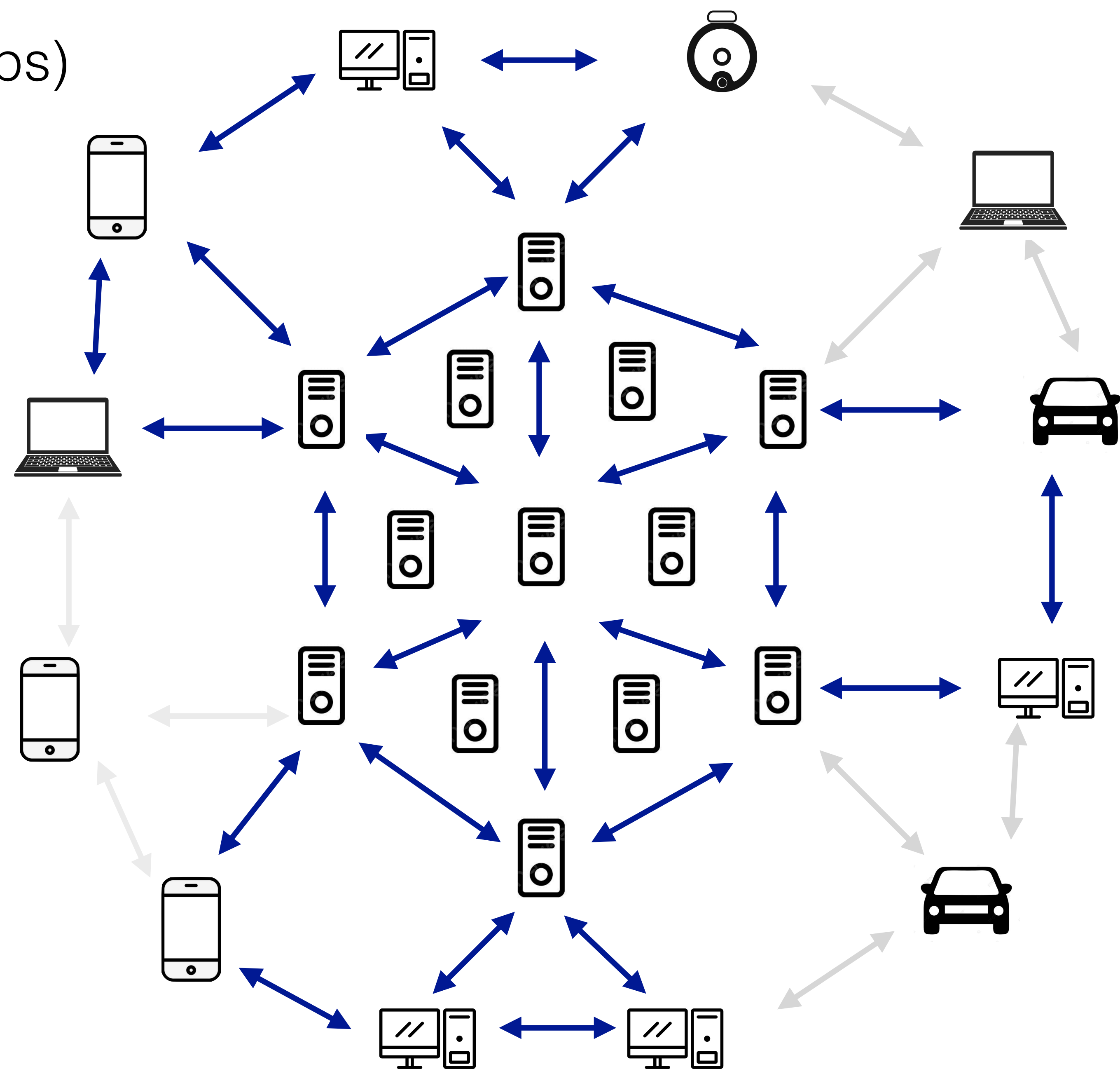


Heterogeneous - Peer to Peer





Local-first (offline ops)





How to program such  
complex systems?



How to produce applications  
that don't crash or get stuck?



How to ensure the  
global behaviour?



How to easily evolve  
such complex systems?

# Internet Applications Design and Implementation

---

An Internet Application (or System) is a **global concept** instantiated into many different **components** running in many **different devices/systems** and **communication over heterogeneous channels**



# Internet Applications Design and Implementation

---

The challenge is to **design**, **specify**, and **implement**

modular,  
loosely-coupled,  
large-scale applications,

so that the software development process is more efficient and the longevity of the applications is longer.

# Internet Applications Design and Implementation

---

**Faster** development cycles,  
functionally **correct** applications,  
applications that can easily be **used by other systems**,  
**heterogeneous** development and execution environments,  
applications are easily **maintainable** (corrections and extensions)  
**secure** systems  
**reliable** and **available** systems  
**performant** applications



**Many of these goals are specific of Internet Applications and should be attained using specific methods, tools and techniques.**

What about Internet Applications?



# Skills for Internet Applications

- Special skills because
  - software evolves fast, we need to develop extensible and maintainable software.
  - data and code have different wills, we need to develop code in sync with data.
  - software is “open” and connected (the world changes), we need to develop loosely coupled, robust software that follows standard API conventions.
  - Internet Apps may grow to have a huge user base (millions of requests)...
  - It also needs to be scalable, secure, replicated, reliable, concurrent, safe...

## Introduction To Continuous Delivery

#1 in the **Continuous Delivery** webinar series

This talk will introduce the principles and practices of Continuous Delivery, an approach pioneered by companies like Facebook, Flickr and ThoughtWorks, that aims to make it possible for an organization to deliver frequently (weekly, daily or even hourly) and confidently. It uses idea -> live (the time from idea being conceived until the feature is live) as a key metric, minimizing that metric across the whole path to production.



By Rolf Russell

DEC 11, 2012 @ 11:48 PM 8,338 VIEWS

## Gmail Outage Embarrasses Internet Giant -- Cause Was a Software Update

### Learning From Netflix. Best Practices to Build Tech Stack—CI/CD Pipeline



Seyhun AKYÜREK · Follow  
4 min read · Apr 26

# “Tools” for Internet Applications

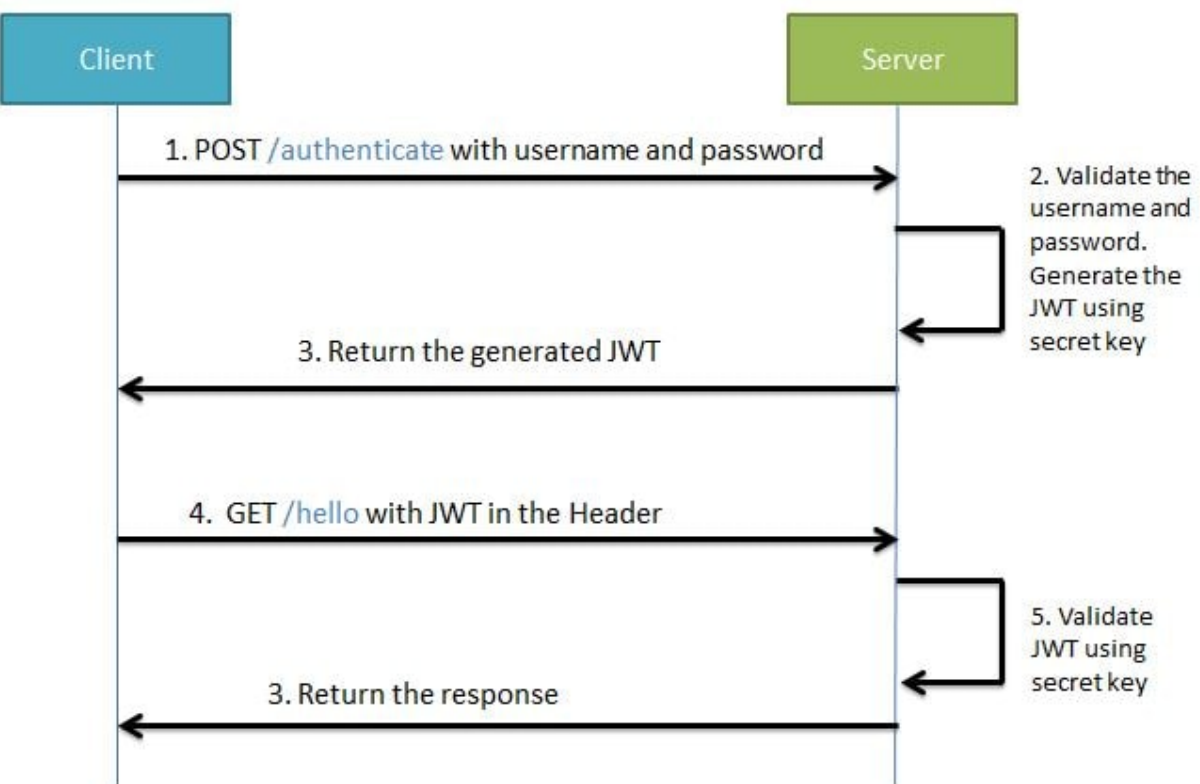
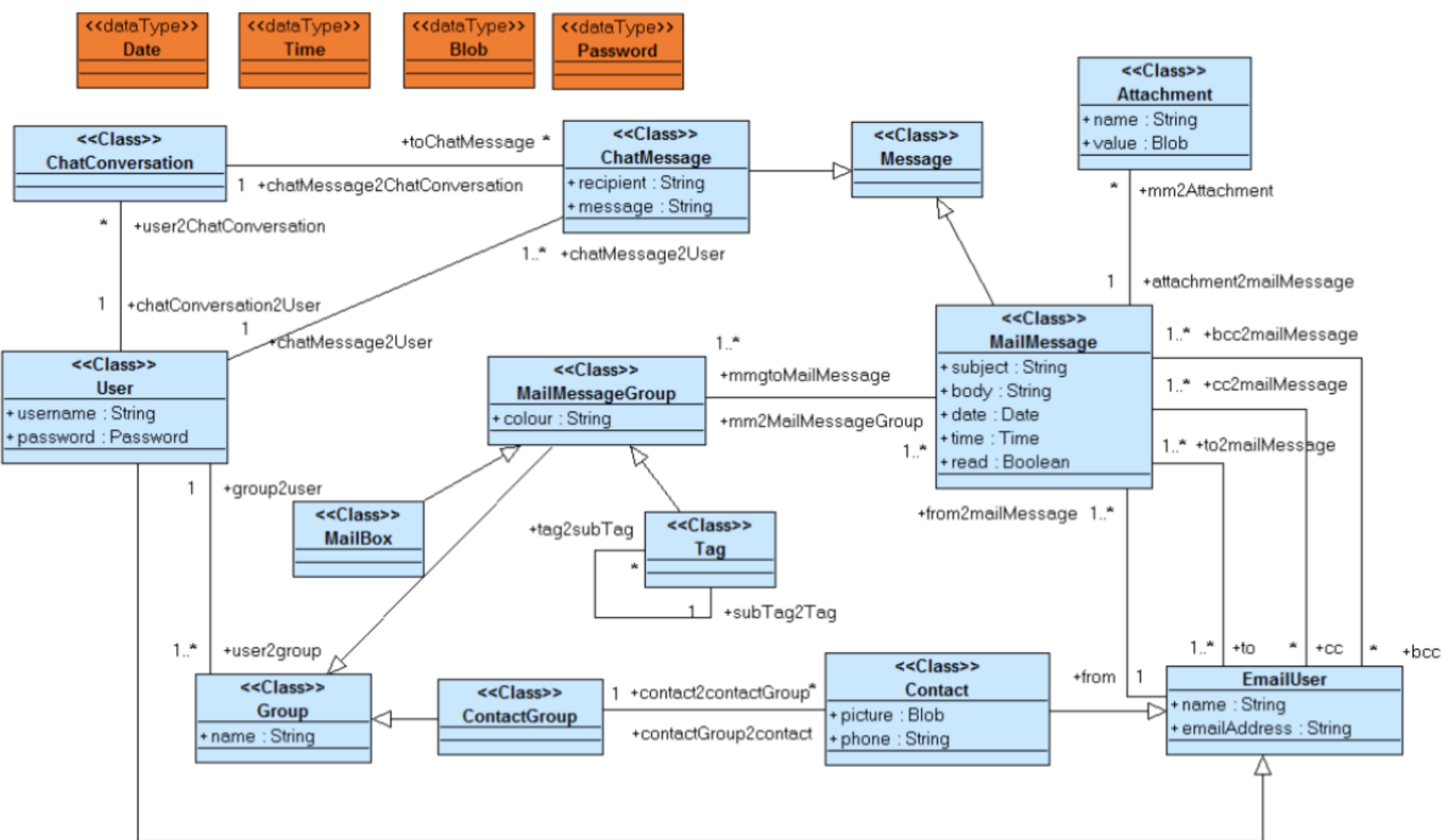
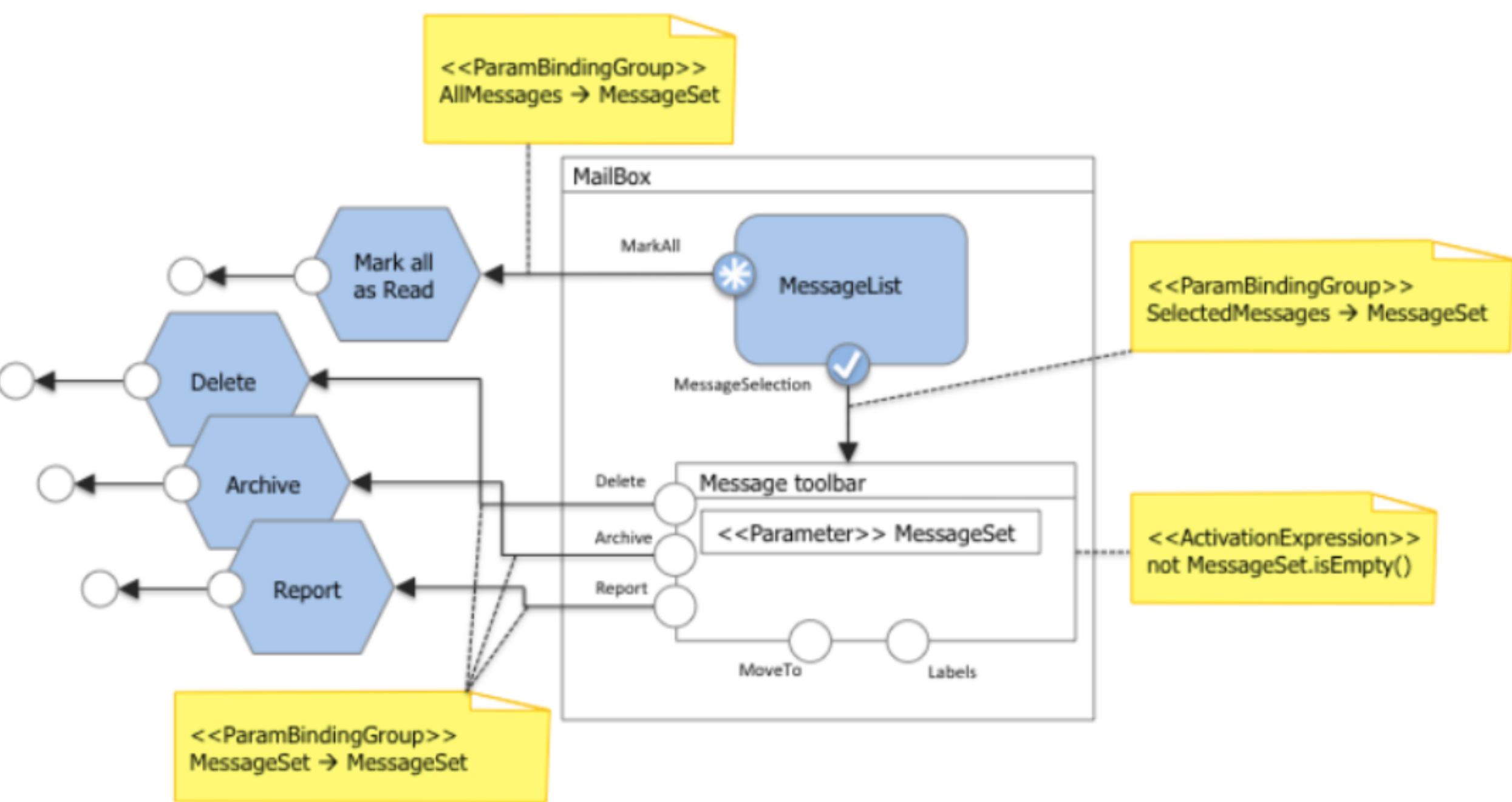
---

- Software Specification (UML, OpenAPI, IFML)
- Software Architecture (Web MVC, SOA, Micro Services, Lambdas, Serveless)
- Software Patterns (Observer, Chain of Responsibility, Facade, Builder, ...)
- Programming Languages (Statically Typed, Dynamic, Concurrent, ...)
- Software Frameworks (Web, Component, Reactive, Data-Abstraction,...)
- Development Methods (Agile, TDD, BDD, Continuous Integration, C. Delivery)
- Deployment Tools (Cloud Managers, Containers, Lambdas, ... )



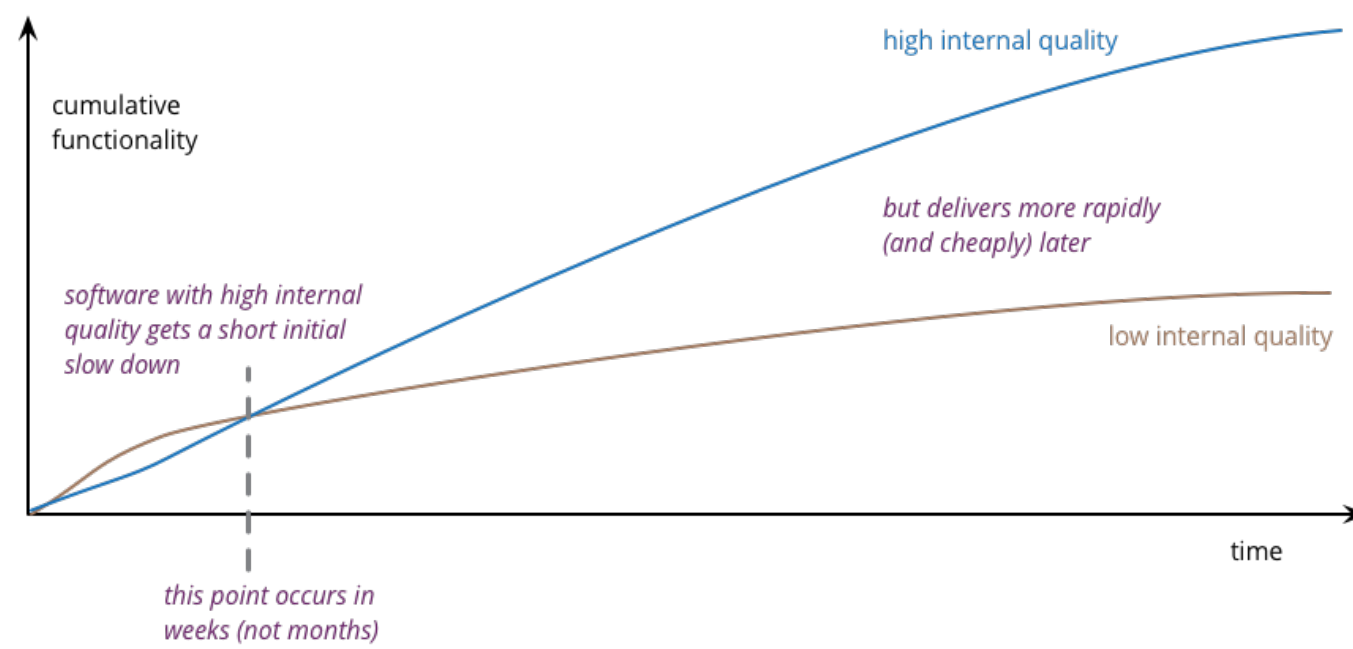
# Software Specification for Internet Applications

- Data schema specification
- Behaviour specification
- Interface-flow specification
- Security specification



# Software Architecture for Internet Applications

- Architecture refers to the internal design of a software system
- It describes the way the highest level components are wired together.
- A good architecture allows a system to be expanded with new capabilities
- A good architecture pays off in quality



in [martinfowler.com/architecture/](http://martinfowler.com/architecture/)

## What is architecture?

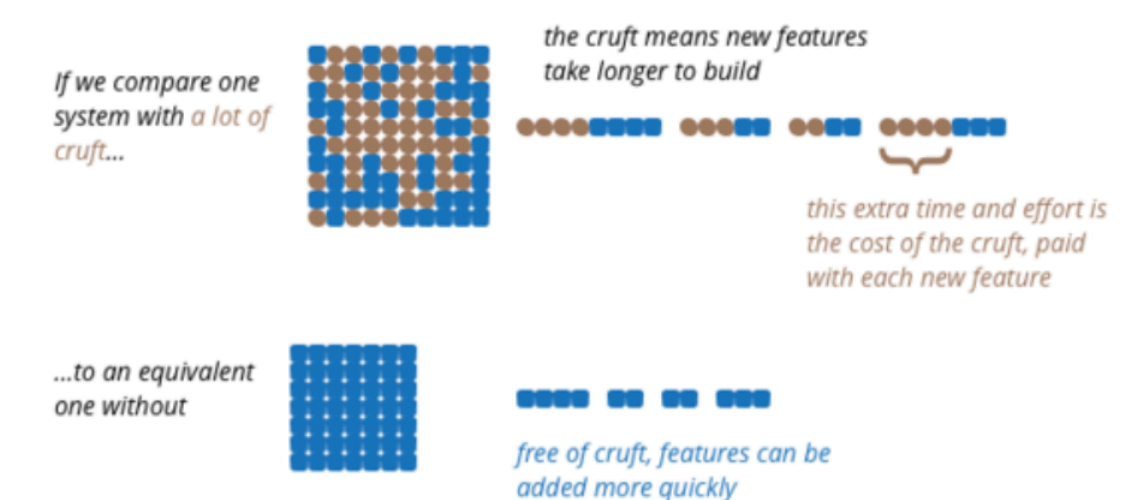
People in the software world have long argued about a definition of architecture. For some it's something like the fundamental organization of a system, or the way the highest level components are wired together. My thinking on this was shaped by an email exchange with Ralph Johnson, who questioned this phrasing, arguing that there was no objective way to define what was fundamental, or high level and that a better view of architecture was **the shared understanding that the expert developers have of the system design.**



Ralph Johnson, speaking at QCon

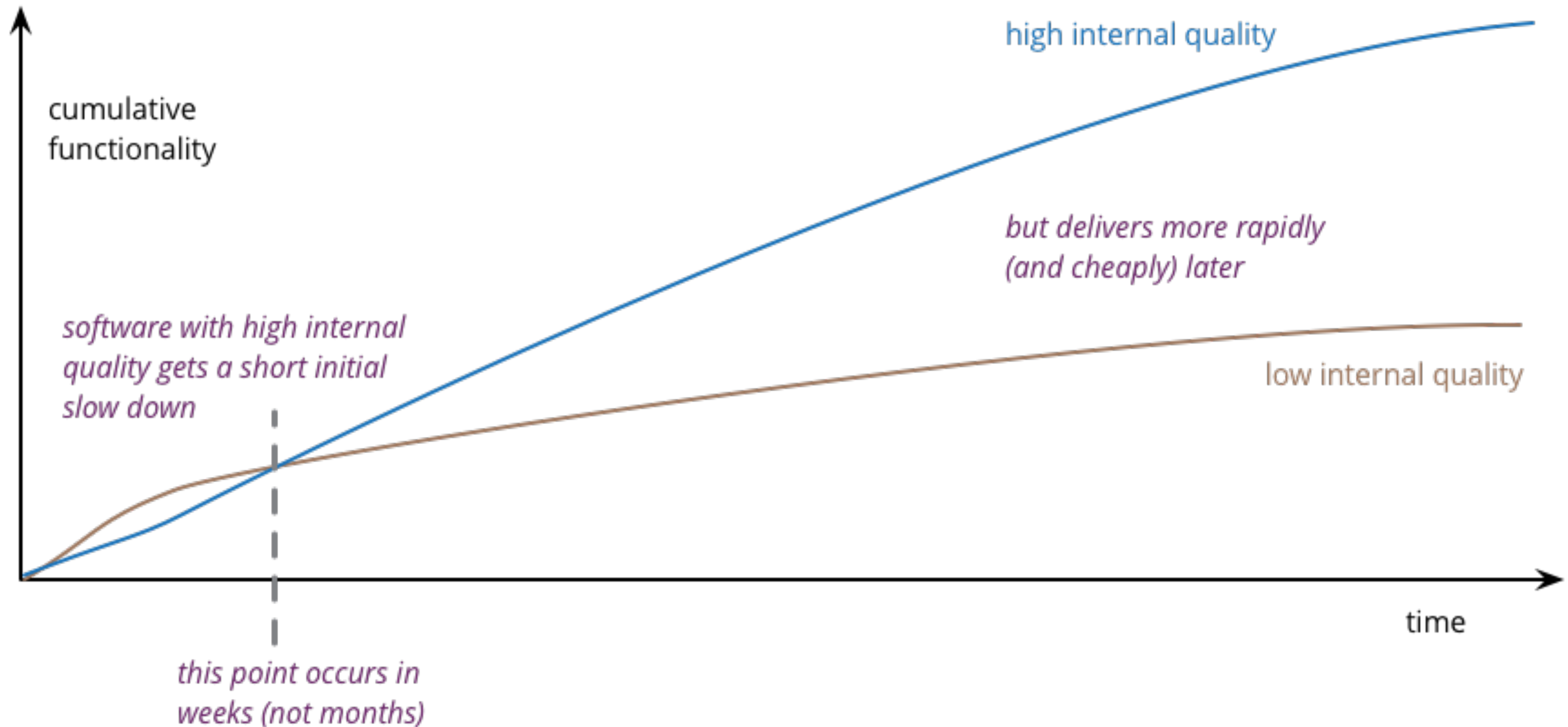
## Why does architecture matter?

Architecture is a tricky subject for the customers and users of software products - as it isn't something they immediately perceive. But a poor architecture is a major contributor to the growth of *cruft* - elements of the software that impede the ability of developers to understand the software. Software that contains a lot of *cruft* is much harder to modify, leading to features that arrive more slowly and with more defects.





# Software Architecture for Internet Applications



in [martinfowler.com/architecture/](http://martinfowler.com/architecture/)



# Software Architecture for Internet Applications

## Application Boundary

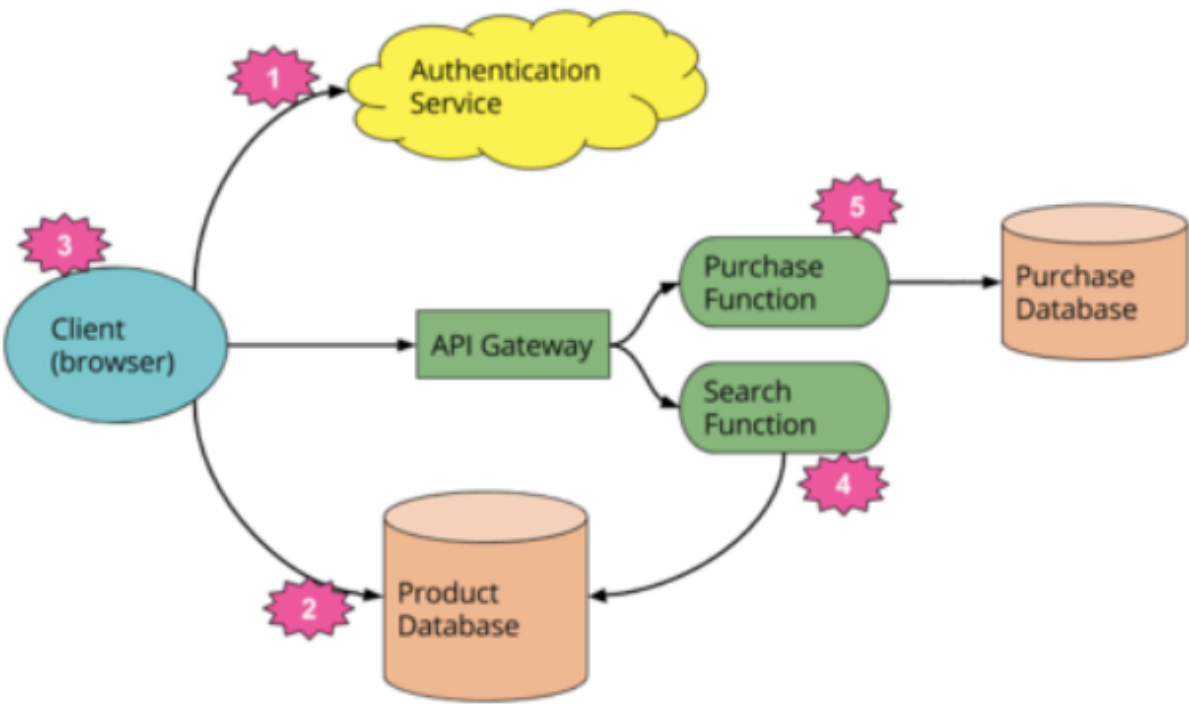
One of the undecided problems of software development is deciding what the boundaries of a piece of software is. (Is a browser part of an operating system or not?) Many proponents of Service Oriented Architecture believe that applications are going away - thus future enterprise software development will be about

## Microservices Guide

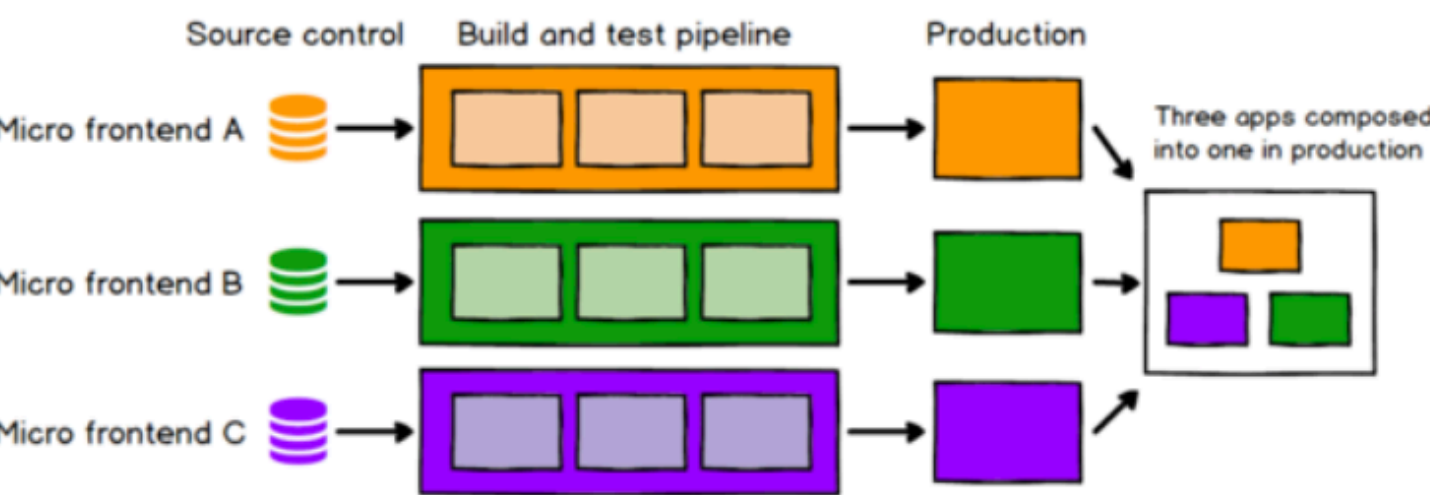


The microservice architectural pattern is an approach to developing a single application as a

## Serverless Architectures



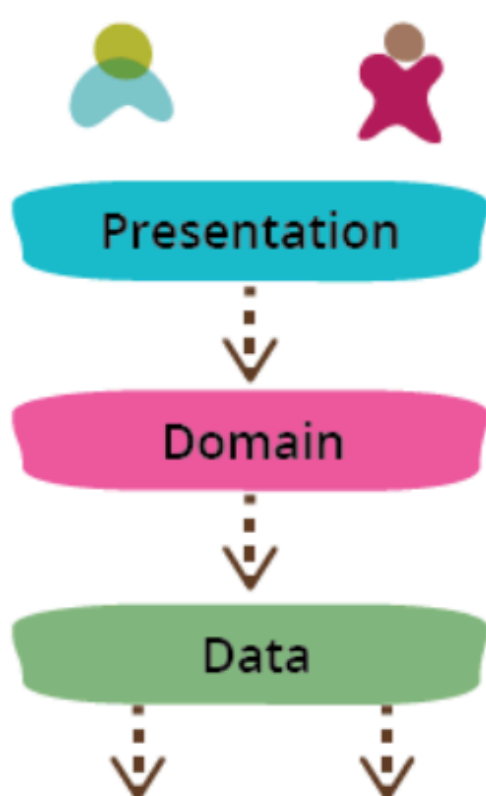
## Micro Frontends



## GUI Architectures

In the mid 2000s I was pursuing a couple writing projects that could have turned into books, but haven't yet made it. One was on the architecture of user interfaces. As part of this work, I drafted a description of how GUI architectures evolved, comparing the default approach of Forms and Controls with the the Model-View-Controller (MVC) pattern. MVC is one of the most ill-understood patterns in the software world

## Presentation Domain Data Layering

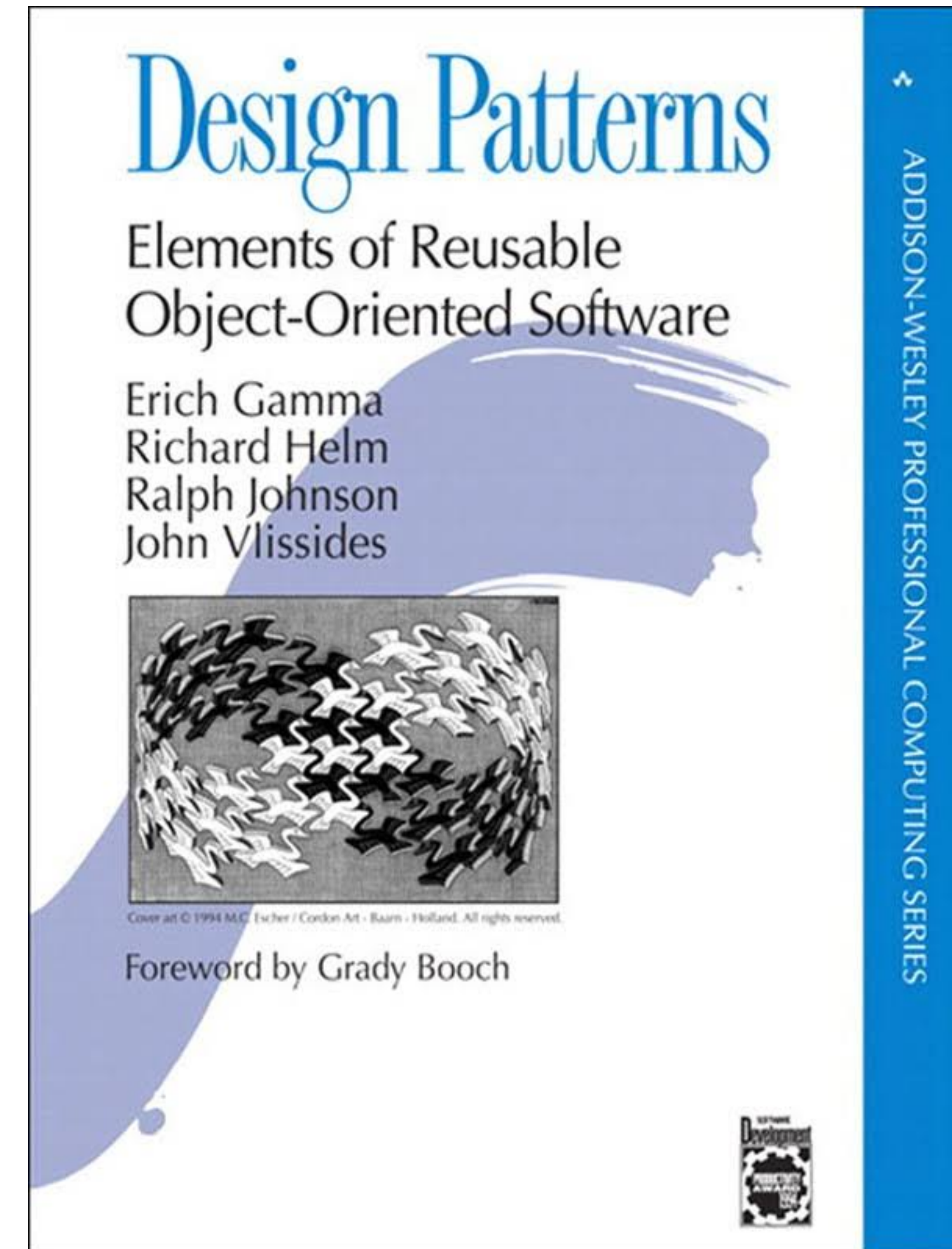


in [martinfowler.com](https://martinfowler.com)



# GOF (Design) Patterns for Internet Applications

- Observer (Web Controllers, Reactive web)
- Chain of Responsibility (Security filters on requests)
- Facade (Service objects, REST interfaces, ORMs)
- Builders (inner to outer representations of data)
- ...
- Frameworks implement design and architectural patterns and styles
  - Layered Architecture
  - Model View Controller
  - Inversion of control
  - REST interfaces



# Software Frameworks for Internet Applications

---

- provide **re-usable code** (as libraries)
- provides **controlled and managed resources**
  - e.g. transactions, log, security
- introduce **abstraction layers**
  - to hide complexity of concrete execution scenarios (hardware/software configuration)
  - Sometimes by scaffolding code (does not evolve well)
- support **test driven development** and **code evolution**
- They work by explicit **configuration** or by **conventions**
- **Improve the overall quality! (correction, maintainability, extensibility)**





Project

Maven Project   Gradle Project

Language

Java   Kotlin   Groovy

Spring Boot

2.2.0 M5   2.2.0 (SNAPSHOT)   2.1.9 (SNAPSHOT)   2.1.8

Dependencies



Start projects with

▼ Developer Tools

**Spring Boot DevTools**  
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

**Lombok**  
Java annotation library which helps to reduce boilerplate code.

**Spring Configuration Processor**  
Generate metadata for developers to offer contextual help and “code completion” when working with custom configuration keys (ex.application.properties/.yml files).

▼ Web

**Spring Web**  
Build web, including RESTful, applications using Spring MVC.  
Uses Apache Tomcat as the default

**Spring Reactive Web**  
Build reactive web applications with Spring WebFlux and Netty.

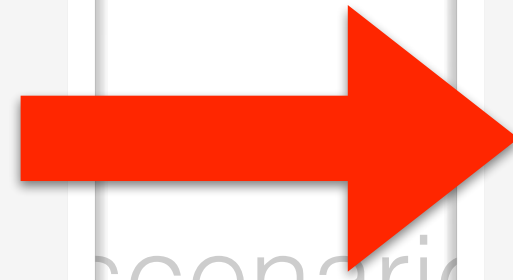
**Rest Repositories**  
Exposing Spring Data repositories over REST via Spring Data REST.

# Software Frameworks for Internet Applications

- provide **re-usable code** (as libraries)
- provides **controlled and managed resources**
  - e.g. transactions, log, security

```
UserTransaction utx = entityManager.getTransaction();

try {
    utx.begin();
    businessLogic();
    utx.commit();
} catch (Exception ex) {
    utx.rollback();
    throw ex;
}
```



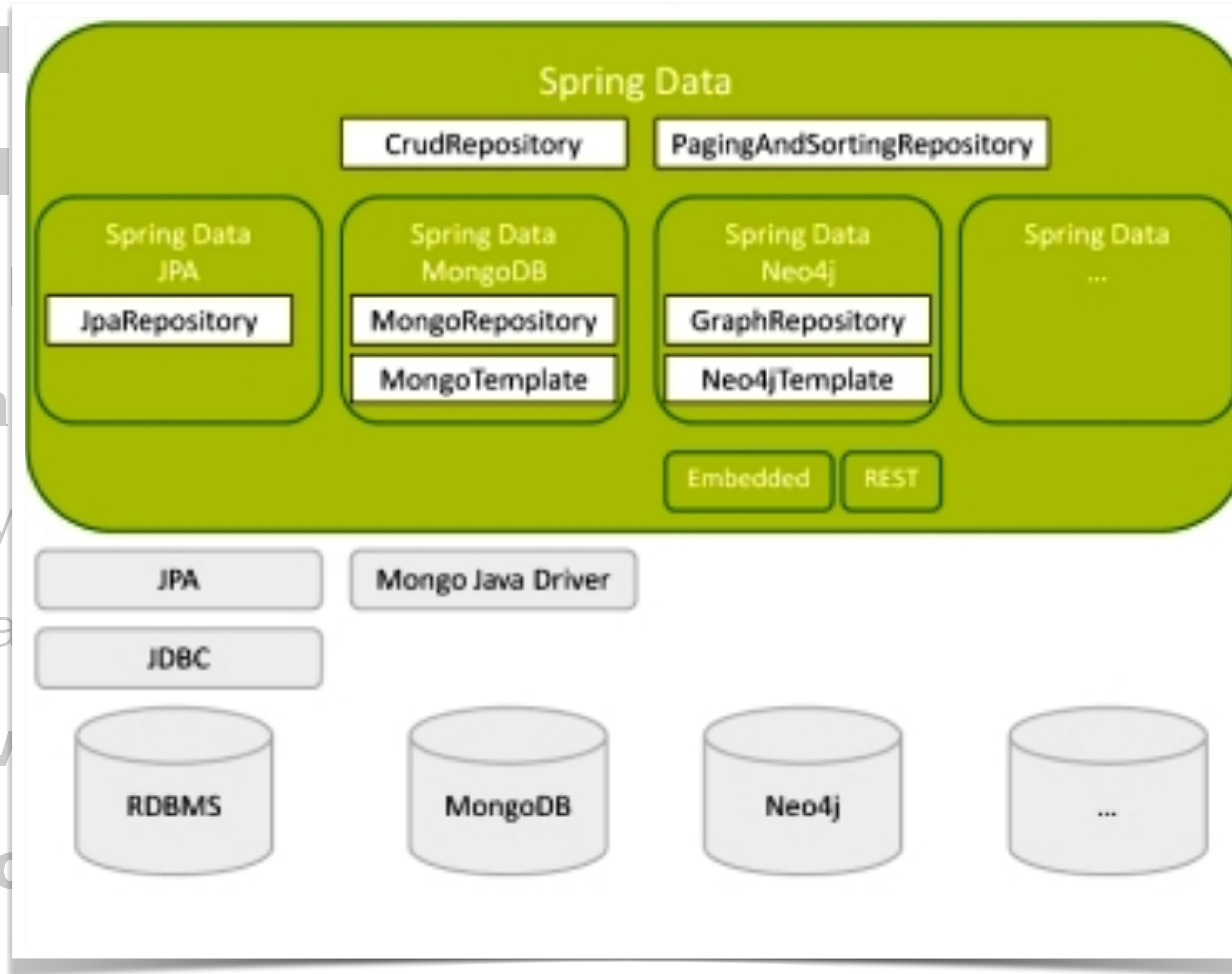
```
@Transactional
public void businessLogic() {
    ... use entity manager inside a transaction ...
}
```

- it's repetitive and error prone
- any error can have a very high impact
- errors are hard to debug and reproduce
- this decreases the readability of the code base
- What if this method calls another transactional method?

<https://dzone.com/articles/how-does-spring-transactional>

# Software Frameworks for Internet Applications

- provide **re-usabl**
- provides **control**
- e.g. transactions,
- introduce **abstra**
- to hide complexity
- Sometimes by sca
- support **test driv**
- work by explicit c



<https://www.infoq.com/articles/spring-data-intro/>



# A spectrum of frameworks for a spectrum of domains

---

- Web Frameworks
- Data manipulation Frameworks
- Resource Oriented Frameworks (REST)
- Process Oriented Frameworks
- Client-side frameworks
- Tierless frameworks
- Low-code platforms

# Software Frameworks for Internet Applications

---

- Examples of Web Frameworks:
  - Ruby on Rails, Django and Python, Spring and Java, Cake and PHP, nodeJS, Meteor, Revel and Go
- Examples of Client Frameworks:
  - AngularJS, React, Vue, Lightweight form, Svelte
- Examples of Data Frameworks:
  - LINQ, Hibernate (JPA), ...
- Examples of Web Service Frameworks/Languages:
  - WS-BPEL...
- Examples of Tierless and Multi-tier Languages:
  - Meteor, Ocaml (w/ Ocsigen), LINKS, UrWeb (research), Hop (research)
- Reactive languages: Loom, Elm, ... (demo)

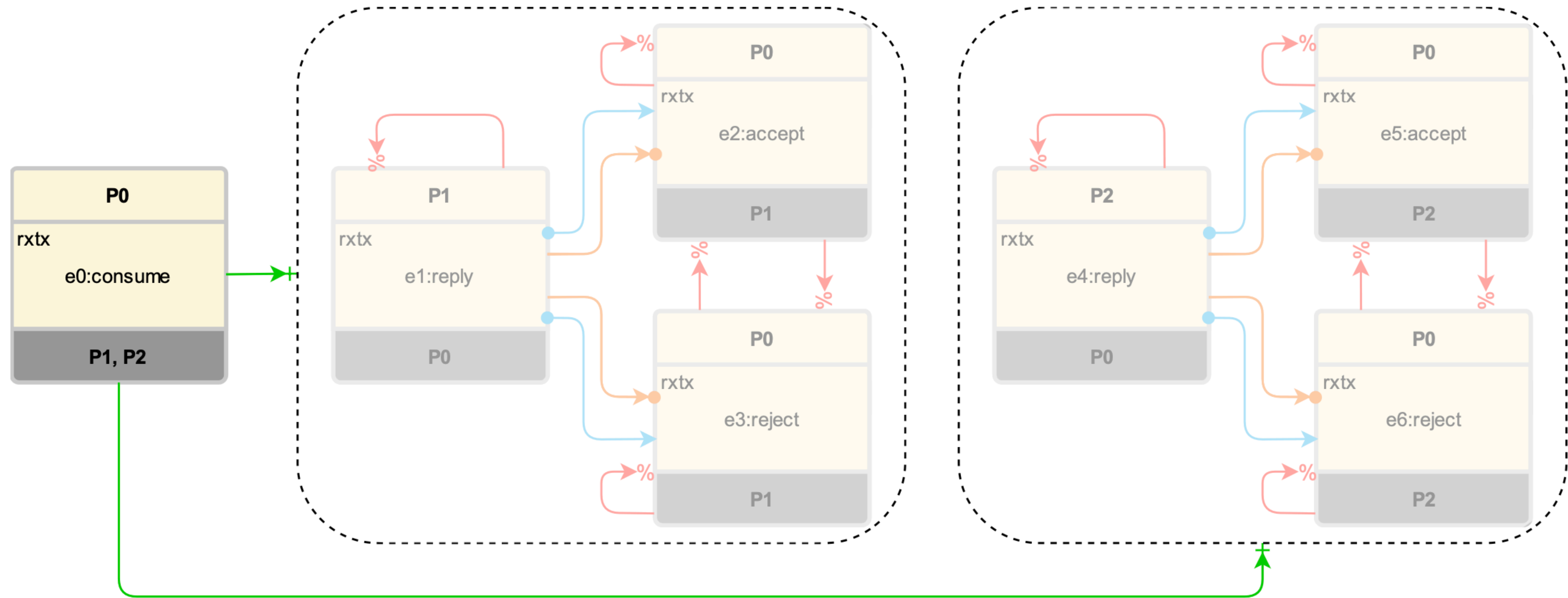
# My Personal Research Perspective

---

- Component-Based, Live and Reactive Programming
- Abstract View of Systems (let compilers and frameworks deal with the details)
- Build tools, frameworks, and programming platforms (e.g. low-code)

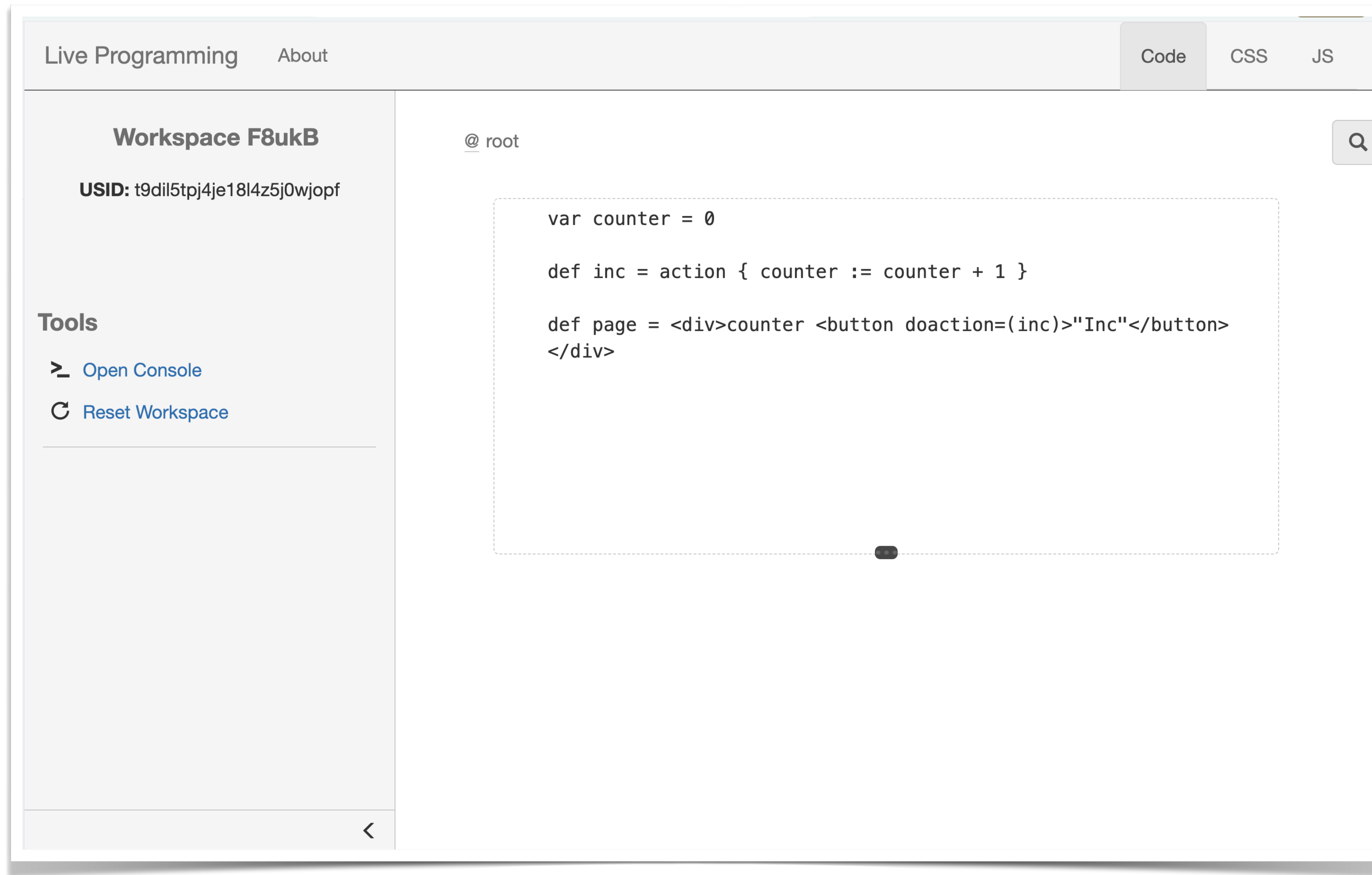


# Example 1: DCR Choreographies



Geraldo, E., Braga, B., Fernandes, N., Ye, D., & Costa Seco, J. (2024). Local Projections of DCR Choreographies with Data and Spawn. 1st Workshop on Principles, Theory, and Practice for Decentralized Applications, Co-Located with ECOOP 2024.

# Example 2: Meerkat language (Live Programming)



Costa Seco J., Aldrich J. The Meerkat Vision: Language Support for Live, Scalable, Reactive Web Apps. International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward)@SPLASH 2024

# Software Frameworks for Web Applications

- Based on Ruby (dynamically typed language)
  - Implements the MVC architectural pattern
  - Pattern components assembled by **conventions** on folders, filenames, and language identifiers
- Very flexible programming language
  - Everything is “re-programmable”
- Rails is a versatile tool
  - Support for scaffolding, migrating code and data, and TDD
- Convention over configuration (even more)
- Big community
  - Big pool of top-of-the-line gems





# Software Frameworks for Web Applications

- Java + J2EE / Spring / Play
- Most used programming language
  - Statically typed and dynamically assembled
  - Well know
  - Easy to start web development
- Huge amount of ready-to-use libraries (Beans)
- Industrial grade efficient implementations
  - Beans framework (MVC is just a module - webmvc)
  - Patterns are assembled **programatically**, or by XML **configuration** files



# Software Frameworks for Web Applications

---

- Elixir + Phoenix is a web development framework written in Elixir which implements the server-side Model View Controller (MVC) pattern.
- Provides high developer productivity and high application performance
- Scaffolding tools like Rails and Django
- Provides LiveView
  - based on web sockets
  - efficiently handle cross border events
  - efficiently refresh web pages based on server side templates

# Software Frameworks for Web Applications

- Spring + Kotlin
- Advantages:
  - It's Completely Interoperable With Java
  - It's (way) More Concise Than Java
  - Safer Code
  - It Comes With a Smarter and Safer Compiler
  - It's Easier to Maintain
  - It's Been Created to Boost Your Productivity
  - It “Spoils” You with Better Support for Functional Programming
  - It Has Null in Its Type System



<https://dzone.com/articles/what-are-the-biggest-advantages-of-kotlin-over-jav>

<https://spring.io/blog/2017/01/04/introducing-kotlin-support-in-spring-framework-5-0>



# Jobs for Internet Applications

A **software architect** is a **software** developer expert who makes high-level design choices and dictates technical standards, including **software** coding standards, tools, and platforms.



[Software architect - Wikipedia](#)

[https://en.wikipedia.org/wiki/Software\\_architect](https://en.wikipedia.org/wiki/Software_architect)

[? About Featured Snippets](#) [Feedback](#)

## The 2019 Roadmap To Fullstack Web Development

### Go Full Stack with Spring Boot and React

Build Your First Full Stack Application with React and Spring Boot. Become a Full Stack Web Developer Now!

★★★★★ 4.5 (378 ratings) 1,918 students enrolled

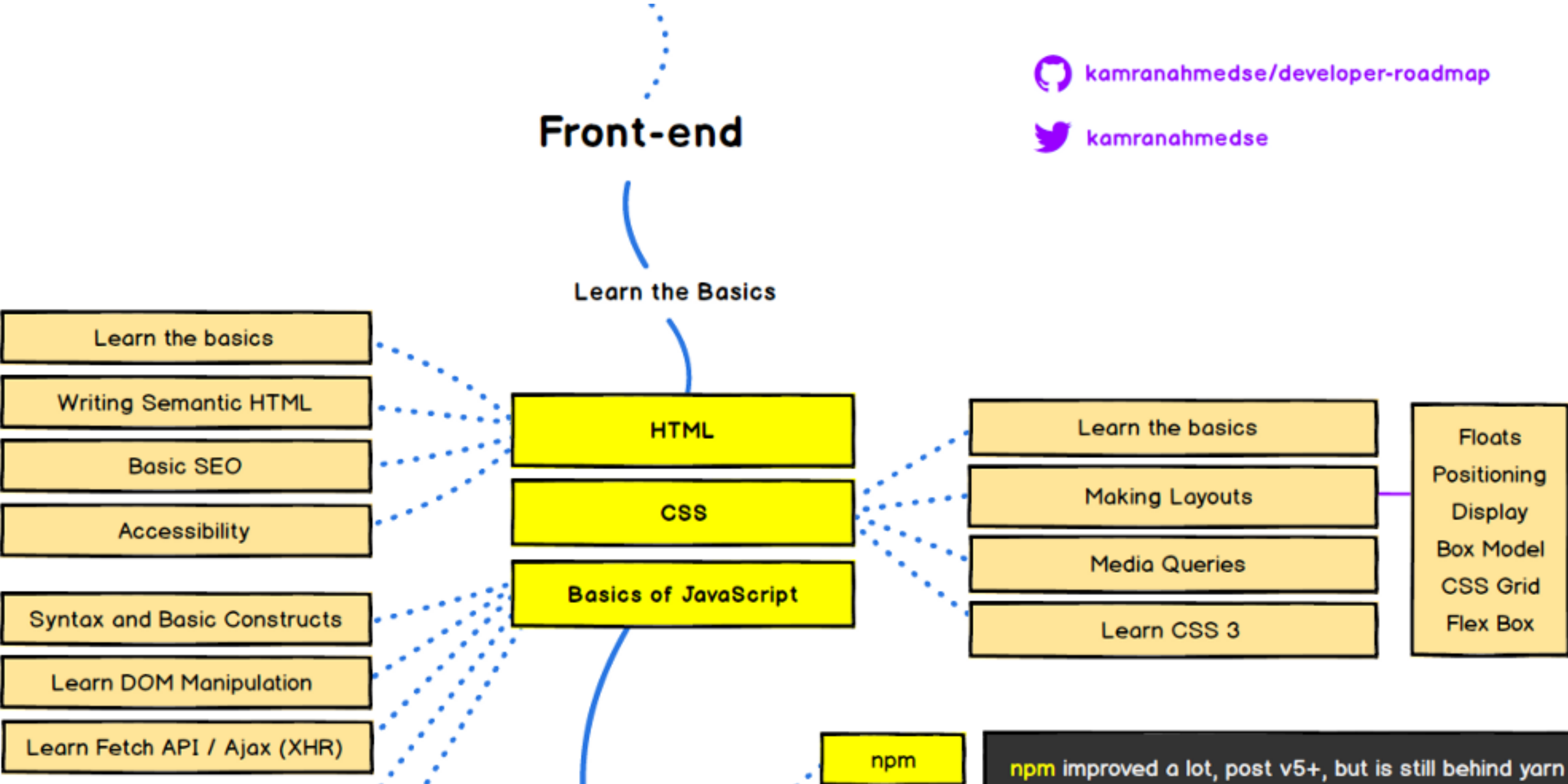
Created by in28Minutes Official Last updated 8/2019 [English](#) [English](#)



[Gift This Course](#) [Wishlist](#)



Preview this course

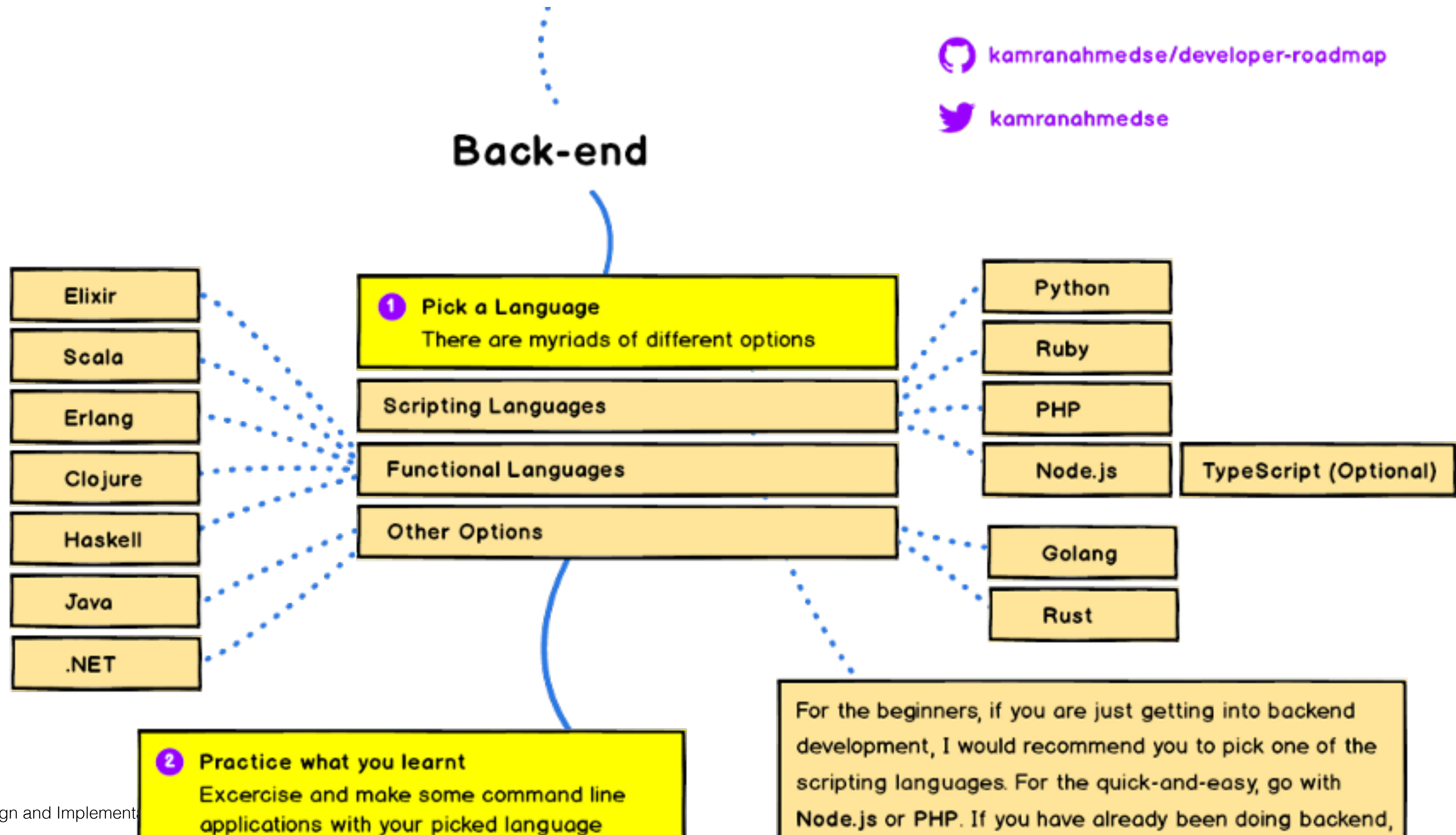
# Programming Languages for Internet Applications



 [kamranahmedse/developer-roadmap](https://github.com/kamranahmedse/developer-roadmap)  
 [kamranahmedse](https://twitter.com/kamranahmedse)



# Programming Languages for Internet Applications





# Development Methods for Internet Applications

- Test Driven Development
  - Founded as part of the “extreme programming (XP)” methodology (1999)
  - Part of the Agile methodology
  - Steps:  
Add a test; Run all tests and see if the new test fails;  
Write the code; Run tests; Refactor code; Repeat
- Behaviour Driven Development
  - Tests are complete examples
- Generic Tests and Test Generation (e.g. QuickCheck)

## Latest Memo: TDD with GitHub Copilot

17 August 2023

by **Paul Sobocinski**

Will the advent of AI coding assistants such as GitHub Copilot mean that we won't need tests? Will TDD become obsolete? To answer this, let's examine two ways TDD helps software development: providing good feedback, and a means to “divide and conquer” when solving problems.

### TDD for good feedback

Good feedback is fast and accurate. In both regards, nothing beats starting with a well-written unit test. Not manual testing, not documentation, not code review, and yes, not even Generative AI. In fact, LLMs provide irrelevant information and even hallucinate. TDD is especially needed when using AI coding assistants. For the same reasons we need fast and accurate feedback on the code we write, we need fast and accurate feedback on the code our AI coding assistant writes.

### TDD to divide-and-conquer problems

Problem-solving via divide-and-conquer means that smaller problems can be solved sooner than larger ones. This enables Continuous Integration, Trunk-Based Development, and ultimately Continuous Delivery. But do we really need all this if AI assistants do the coding for us?

Yes. LLMs rarely provide the exact functionality we need after a single prompt. So iterative development is not going away yet. Also, LLMs appear to “elicit reasoning” (see linked study) when they solve problems incrementally via chain-of-thought prompting. LLM-based AI coding assistants perform best when they divide-and-conquer problems, and TDD is how we do that for software development.

### TDD tips for GitHub Copilot

At Thoughtworks, we have been using GitHub Copilot with TDD since the start of the year. Our goal has been to experiment with, evaluate, and evolve a series of effective practices around use of the tool.

# Deployment Methods for Internet Applications

- Software development is increasingly competitive
- Any mistake can be extremely expensive
- Pressure is on to deliver fast and change even faster
- Companies deploy software at an astonishing pace:
  - Amazon: “every 11.7 seconds”
  - Netflix: “thousands of times per day”
  - Facebook: “bi-weekly app updates”



## Evolving How Netflix Builds, Maintains, and Operates Their Spinnaker Distribution



Adam Jordens · [Follow](#)

Published in The Spinnaker Community Blog · 8 min read · May 5, 2021

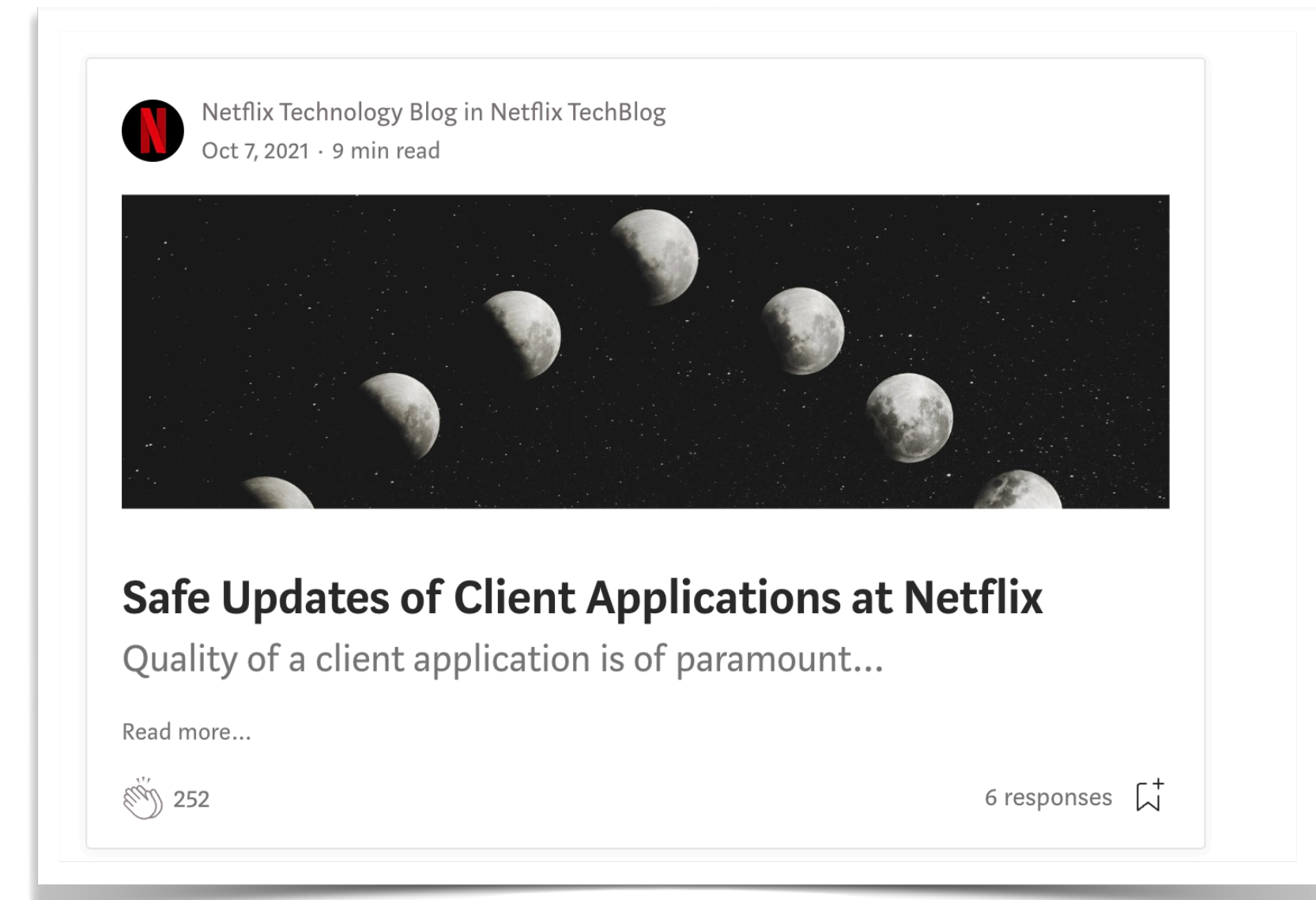
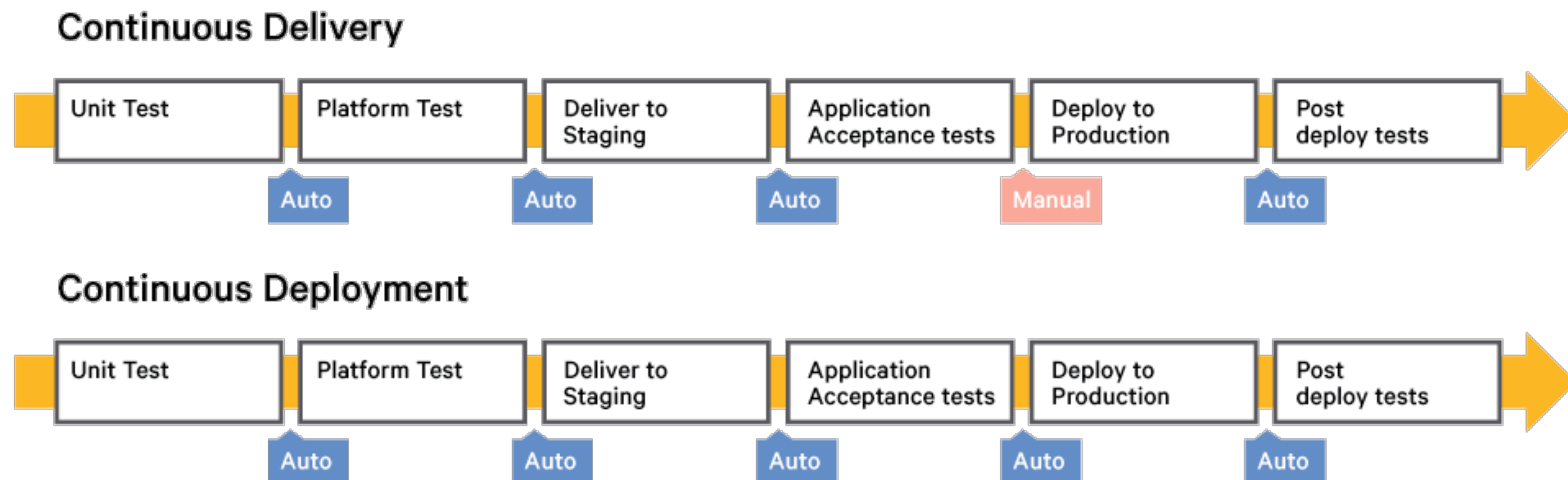
**Much has changed.**

We now orchestrate more than **20k** deployments daily. This number is significantly higher when you consider more discrete changes like dynamic configuration adjustments, security group refinement, and load balancer creation that are handled safely and efficiently by the platform we call Spinnaker.



# Deployment Methods for Internet Applications

- Processes and Methods for software construction and software deployment.
- Specification and development methods
- Testing tools and toolchains
- Validation and Verification techniques





# Course Overview

# Internet Applications Design and Implementation

---

We focus on the **principles and concepts on the development of Internet applications**.

The syllabus follows an approach based on the fundamentals of software development based on **web and service oriented architectural patterns, advanced modularity mechanisms, data persistency abstractions, good development practices, performance concerns**, and **validation techniques**.

Lectures run along **practical assignments** and the **development of a running project** using frameworks, languages, and programming tools for Internet Applications that ensure the safety and compliance of the solution with relation to a specification

# Goals: To Know

---

- Essential aspects of **architectural patterns** for inversion of control and software architectures specific for Internet Applications.
- **Principles of the development** of web applications and single page web applications.
- Mechanisms of **specifying and implementing web services** and web service orchestrations.
- **Internal structure** of an Internet **browser** and its **client applications**.
- Principles of **data-centric** and **user-centric development** in the context of Internet applications.
- Main **data abstraction** mechanisms used in Internet applications.
- Major performance **pitfalls** of Internet applications and their workarounds.
- Main specification and implementation mechanisms for **security policies in Internet Applications**.



# Goals: To Do

---

- **Use development frameworks** that implement architectural styles for Internet applications.
- **Specify and build** web and cloud **applications** to support thin, flat, and native clients.
- **Specify and build** client **applications with reactive and rich behaviour**.
- **Implement authentication mechanisms** and **specify** the core **security rules** of an Internet Application
- Specify and efficiently use abstraction data layers such as **Object Relational Mappings** in Internet applications.
- **Design and deploy** Internet Applications that are efficient and maintainable.

# Resources

---

- Github classroom (assignments/project, starter code)
- [iadi2024.github.io](https://iadi2024.github.io) (slides and other documents)
- Discord (direct contact, conversation, questions)



# Evaluation

---

- Written evaluation component (60%) — two on-campus Tests or one Exam
- Laboratory work component (40%) — there is no “frequência”/not valid next year
  - teams of **3** members
  - 1 project, 2 submissions using GitHub classroom
  - If you don't know how to operate with git, go learn TODAY!**
- A final report (fixed template) presentation and discussion to assert the obtained results.



# 12x, 2h Weekly Lectures. aka “a long and steep 12 week hike”

---

23 Sep - Course Overview

30 Sep - Software Architecture. Services and Micro Services

7 Out - Data Abstraction, JPA & Hibernate

9 Out (Extra) - More Data Abstraction, Associations and Custom Queries

14 Out - Continuous Integration, Unit Testing, Mock Services

21 Out - No Lecture (SPLASH 2024) Autonomous Study

28 Out - Security Mechanisms in Internet Applications

30 Out (Extra) - Model-Based Access Control

4 Nov - Q&A

11 Nov - User-centric development - Client Side Reactive Programming (React)

18 Nov - IFML and UI Design. IFML to React

25 Nov - Client Side State Management

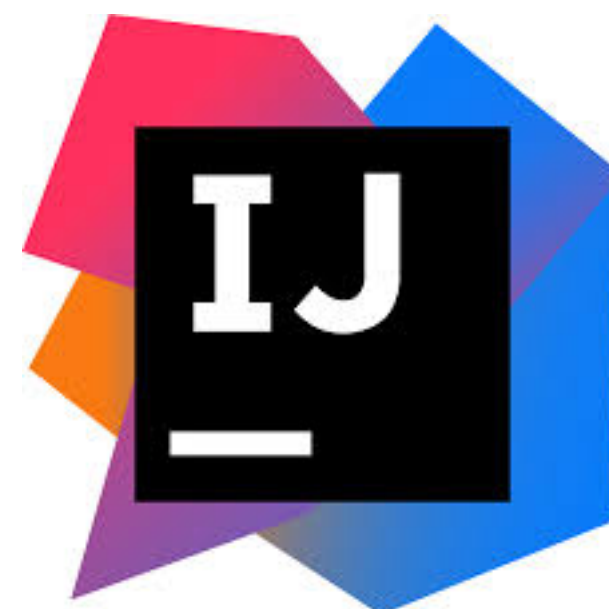
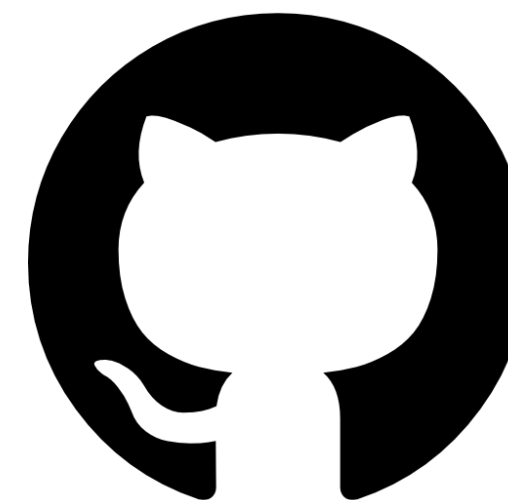
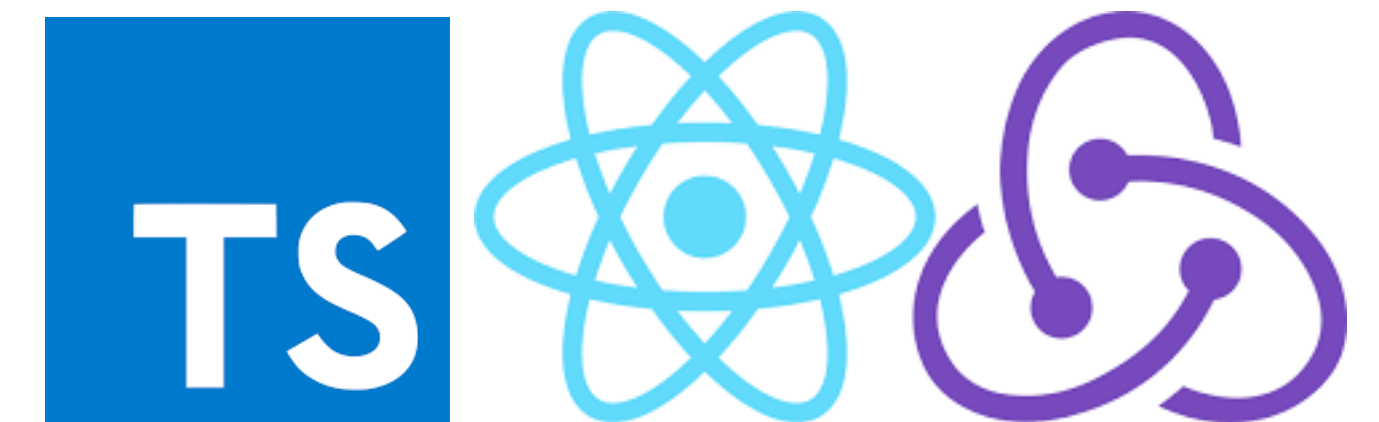
2 Dez - TBD

9 Dez - Q&A



# Lab Exercises

- Tech Stack (mandatory):
  - Server-side: SpringBoot + Kotlin + Swagger + JUnit5 + MySQL
  - Client-side: TypeScript + React + Redux
- Tools stack
  - git (GitHub — mandatory),
  - GitHub classroom
  - IntelliJ
  - httpie/curl/postman



# Project Deliverables

---

- A single project from the beginning:
  - Data-driven development of service based application, specification of data model and API
  - User-driven development of a progressive/single page client application, specification of user interaction
- A single git (mandatory) repository with client and server code, documentation, presentation and report - GitHub classroom
- Report will include UserStories, UML/ER, IFML, OpenAPI spec, security spec
- Unit and Integration Test suite (including security aspects)
- Evaluation will cover documentation, functional correctness, code construction, coverage, security and performance issues.



# Code of conduct

---

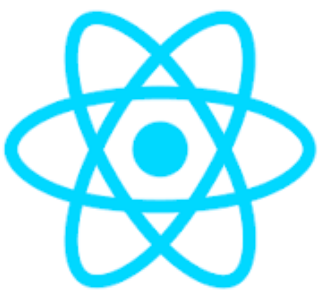
- You are expected to do the practical work in teams independently with the help of the instructor in the lab sessions. You can also use Discord to ask questions anytime. Don't paste code or links to code in Discord.
- We assume fair treatment and a fair balance of work between team members.
  - Look for a team that works. Talk about work style, schedule, commitment level, etc.
- You can use GitHub Co-Pilot and ChatGPT as an aid to understand code or any other topic, not as a primary source of code or text. A reflection on the use of LLMs should be included in the report if used.
- Cheating:
  - If you are looking into someone else's code, you are probably beyond the scope of legal behaviour.
  - Unreported use of Co-Pilot or ChatGPT can be considered cheating.
  - Drastically unbalanced work is cheating and will be penalised (possibly disqualified)



# Bibliography



learn   play   download   interact  
tutorial   handbook   samples   language spec





# Bibliography

---

- Marco Brambilla and Piero Fraternali. Interaction Flow Modeling Language – Model-Driven UI Engineering of Web and Mobile Apps with IFML. Morgan Kaufmann. 2014
- Martin L. Abbott and Michael T. Fischer, The Art of Scalability: Scalable Web Architecture, Processes and Organizations for the Modern Enterprise, Addison-Wesley Professional. 2009
- Martin Fowler. Patterns of Enterprise Application Architecture. Addison-Wesley. 2002
- Software Architecture in Practice (3rd ed). Len Bass, Paul Clements, and Rick Kazman. Addison-Wesley 2015.
- Craig Walls. Spring in Action (4th edition). Manning. 2015

[spring.io](http://spring.io)

[facebook.github.io/react/](https://facebook.github.io/react/)

[typescriptlang.org](http://typescriptlang.org)

[kotlinlang.org](http://kotlinlang.org)



