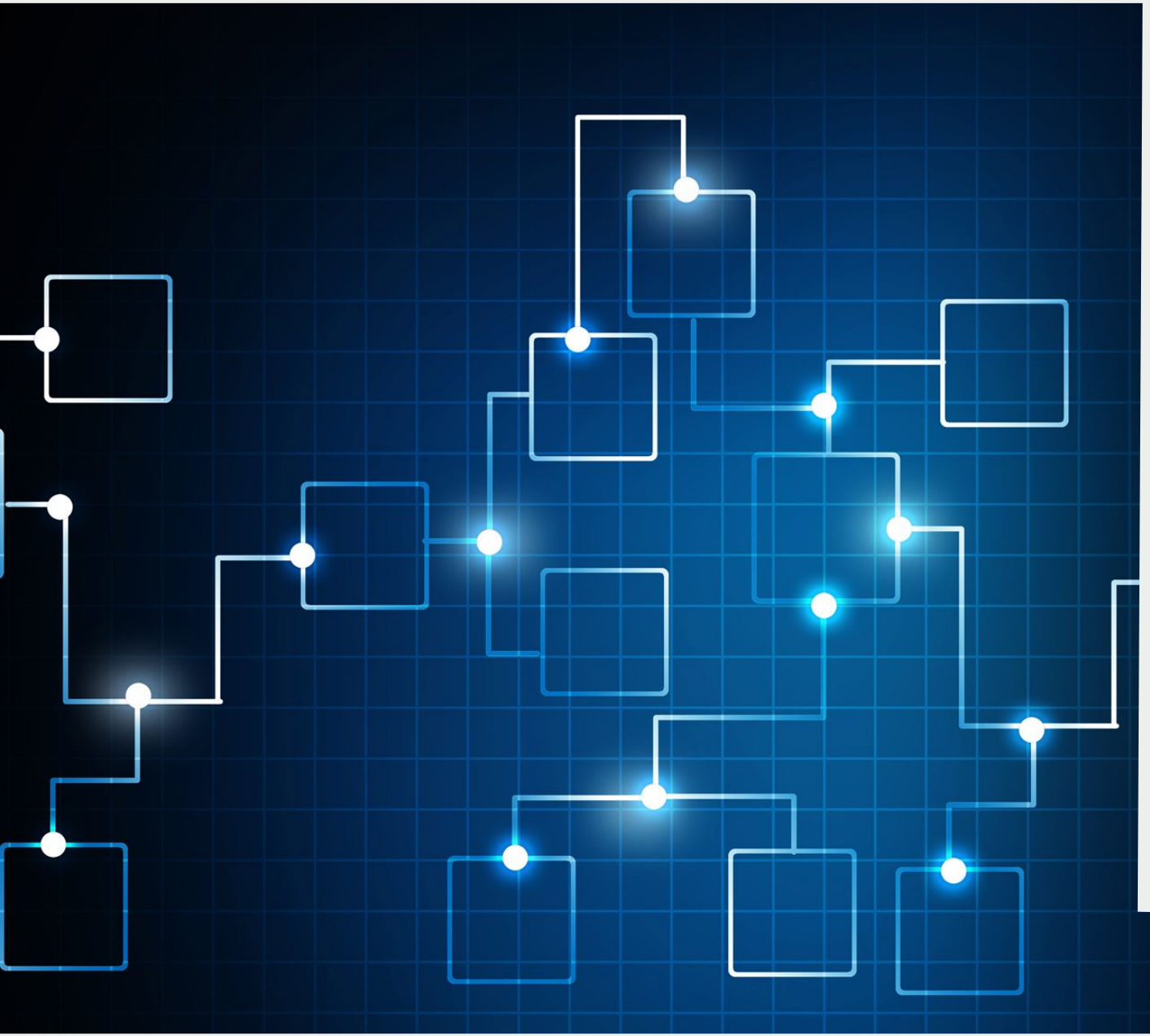
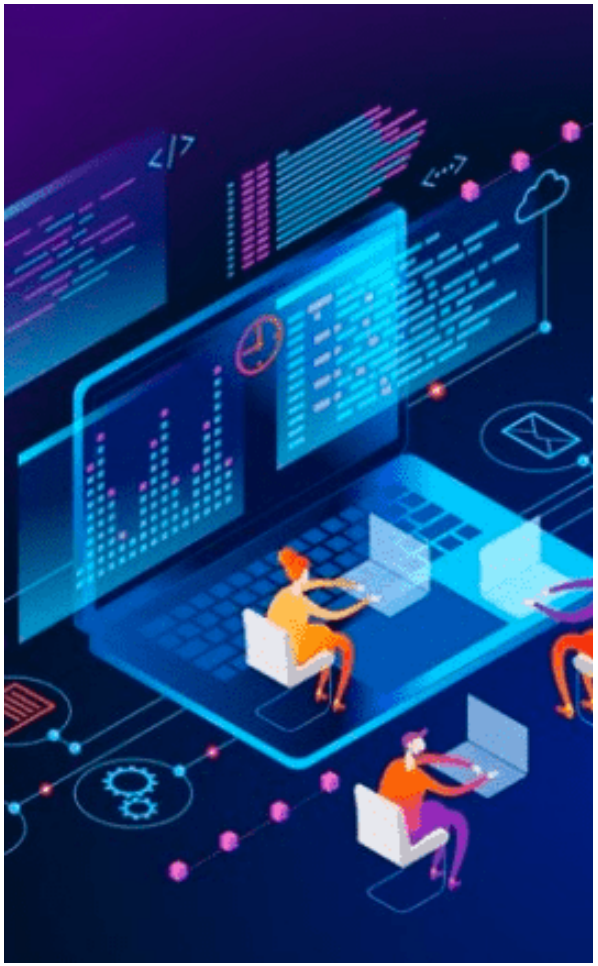


TDP024- Enterprise System

DESIGNING SCALABLE SOLUTIONS FOR COMPLEX BUSINESS NEEDS



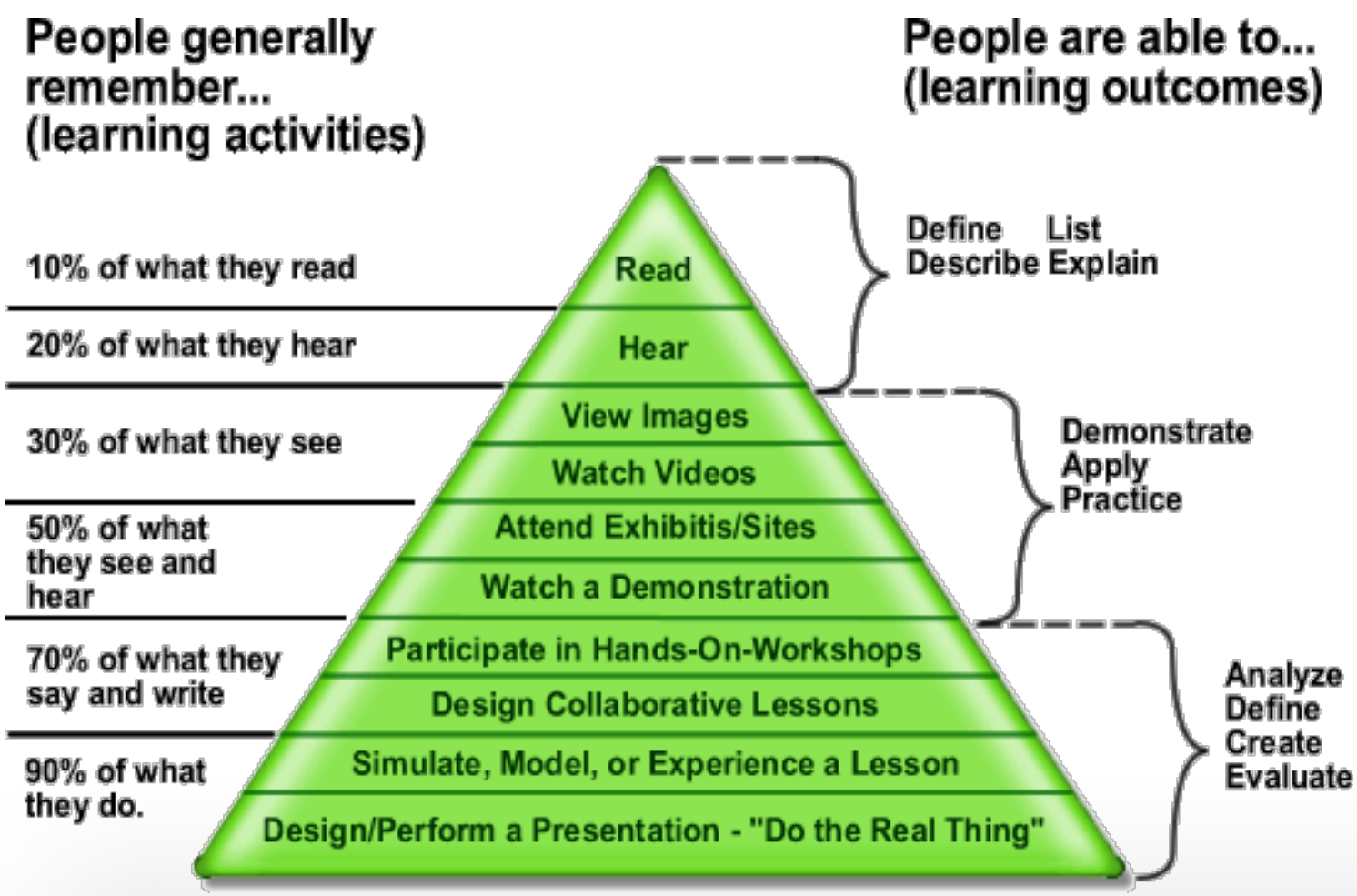
TDP024 -Outline

- Introduction
- Whats is ES
- What is SOA

Anders Fröberg (anders.froberg@liu.se)

Institutionen för Datavetenskap (IDA)

Kursens kunskapsfilosofi



KOMMER NI KOMMA IHÅG ?

- Load up your guns, and bring your friends
- This is the end the only end my Friend
- Äntligen stod prästen i predikstolen
- I'll be back
- Han kom som ett yrväder en aprilafton och hade ett höganäskrus i en svångrem om halsen
- Jag har inget fint att ge, mer än 60 kilo ensamhet
- It was the best of times, it was the worst of times
- Face it tiger, you just hit the jackpot!

TDP024 Kursinformation

Kurshemsidan innehåller information om laborationerna, seminariet och examinationen.

- Föreläsningskoden finns på hemsidan/gitlab, detta är kurslitteraturen för denna kurs. **Att förstå hur denna kod är uppbyggd, och att följa de riktlinjer som ges i denna kod är avgörande för att få godkänt på laborationerna.**

- Kursen ger betyg U/3/4/5

Kursansvarig: Anders Fröberg anders.froberg@liu.se
Kursassistent: Sahand Sadjadee sahand.sadjadee@liu.se
Labassistent: Charlie Simonsson charlie.simonson@liu.se
Examinator: Anders Fröberg

TDP024 – In a nutshell

I examensordningen står det att för alla kandidatexamina skall bland andra följande mål uppnås: (Även andra examina än kandidatexamina har nästan identiska mål)

1. visa förmåga att **söka**, **samla**, **värdera** och **kritiskt tolka** relevant information i en problemställning samt att kritiskt diskutera företeelser, frågeställningar och situationer

2. visa sådan färdighet som fordras för att **självständigt arbeta** inom det område som utbildningen avser

3. visa förmåga att **självständigt identifiera, formulera och lösa problem** samt att genomföra uppgifter **inom givna tidsramar**

4. visa förmåga att **muntligt** och **skriftligt** redogöra för och diskutera information, problem och lösningar i dialog med olika grupper

Utvärdering och Förändringar

- Oklarheter vad som tar tid
  - Projektet tar mycket tid
- Git var inte med så mycket efter labben
  - Använd det även om vi inte har officiella krav

TDP024 – In a nutshell part 2

- Laborationer +Projekt med betyg 5Hp
- Två labbar
  - Git (lab 1)
  - Maven (lab 2,)
- Projekt
  - SOA+Kafka
- Individuell rapport och seminarier 1Hp
  - Sök och välj ut tre bra forskningsartiklarna relaterade till kursens innehåll . Skriv en sammanfattning och analys av alla tre artiklarna på en A4-sida vardera.
  - Värdera varje artikel - Varför är den bra
  - Presentera och Diskutera på seminarium
- Fokus på kod och struktur/arkitektur

TDP024 – Laborationer/Scenarion

- Laborationerna genomförs i par, registrering via webreg.
- Salarna: Thinlinc/RDP till Linux maskiner,SU-sal, eller egen dator.
- Ni använder valfri IDE/editor för utveckling, Maven för att bygga

Course Overview

Enterprise Systems Introduction

The course introduces architecture and development of large-scale enterprise software systems.

Design Principles and Tools

Emphasizes scalable, maintainable, and secure system design using modern methodologies and tools.

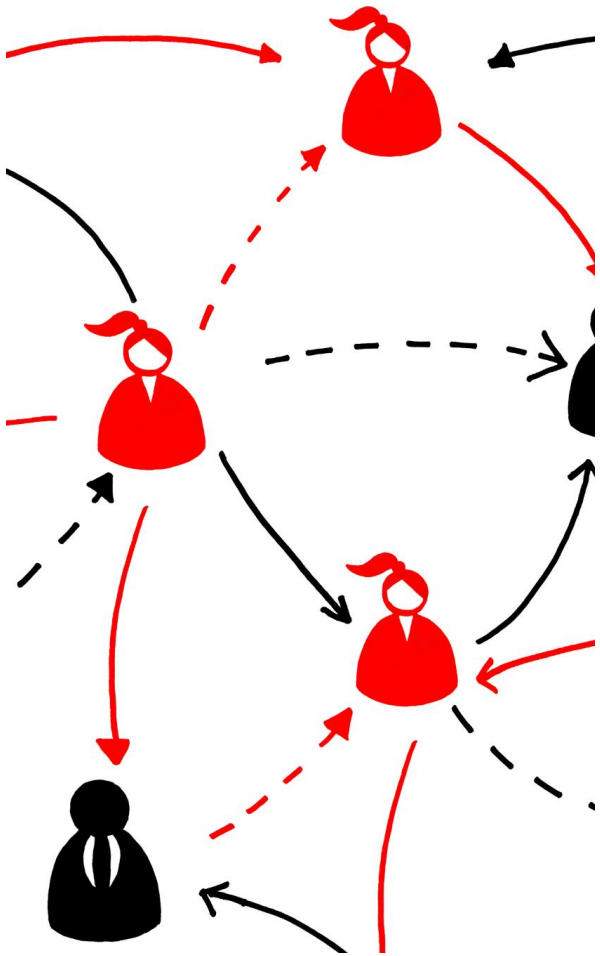
Practical Learning Approach

Hands-on projects and case studies provide real-world experience in implementing enterprise solutions.

Advanced Topics Not Covered

Covers system integration, performance optimization, and security compliance more in advanced course.

INTRODUCTION TO ENTERPRISE SYSTEMS



Definition and Examples of Enterprise Systems

Enterprise Systems Overview

Enterprise Systems support core business processes and information flow in complex organizations, enabling seamless operations.

Enterprise Resource Planning (ERP)

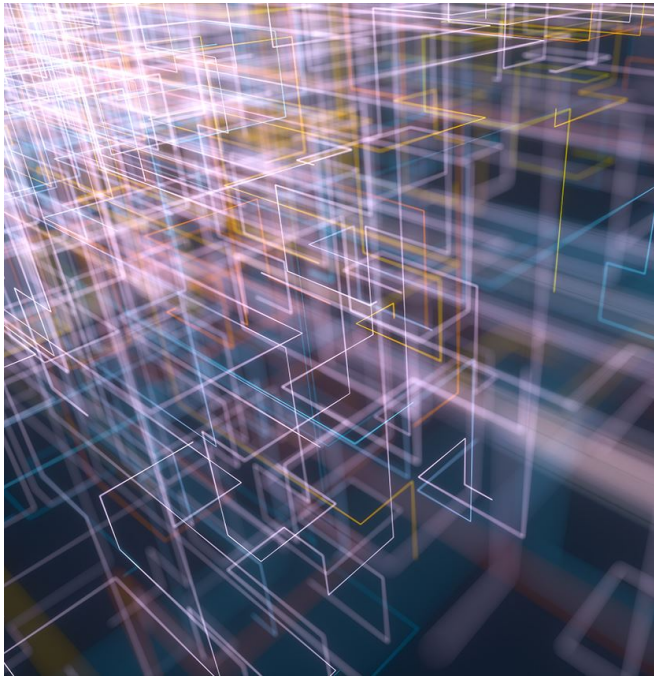
ERP systems integrate finance, HR, and supply chain functions into a unified platform for efficient management.

Customer Relationship Management (CRM)

CRM systems manage customer interactions and data to enhance relationships and improve sales performance.

Supply Chain Management (SCM)

SCM systems oversee logistics and inventory, ensuring smooth supply chain operations and timely deliveries.



Key Characteristics of Enterprise Systems

Scalability

Enterprise systems must scale efficiently to support increasing data volumes and user numbers.

Reliability

Consistent performance and uptime are critical to ensure business continuity in enterprise systems.

Security Features

Robust security protects sensitive data and ensures compliance with regulations.

Integration and Maintainability

Integration allows seamless data exchange; maintainability enables easy updates and management.

Technologies Involved in Enterprise Systems

Backend Development

Backend development uses languages like Java, C#, Python, and Node.js to build robust server-side logic.

Frontend Technologies

Frontend frameworks such as Angular, React, and Vue.js create responsive and interactive user interfaces.

Databases

Databases like SQL Server, Oracle, PostgreSQL, and MongoDB efficiently store and manage enterprise data.

Middleware and APIs

Middleware and APIs including REST, GraphQL, and SOAP enable communication between system components.

Cloud Platforms

Cloud platforms like AWS, Azure, and Google Cloud provide scalable infrastructure and services.

Technologies Involved in Enterprise Systems

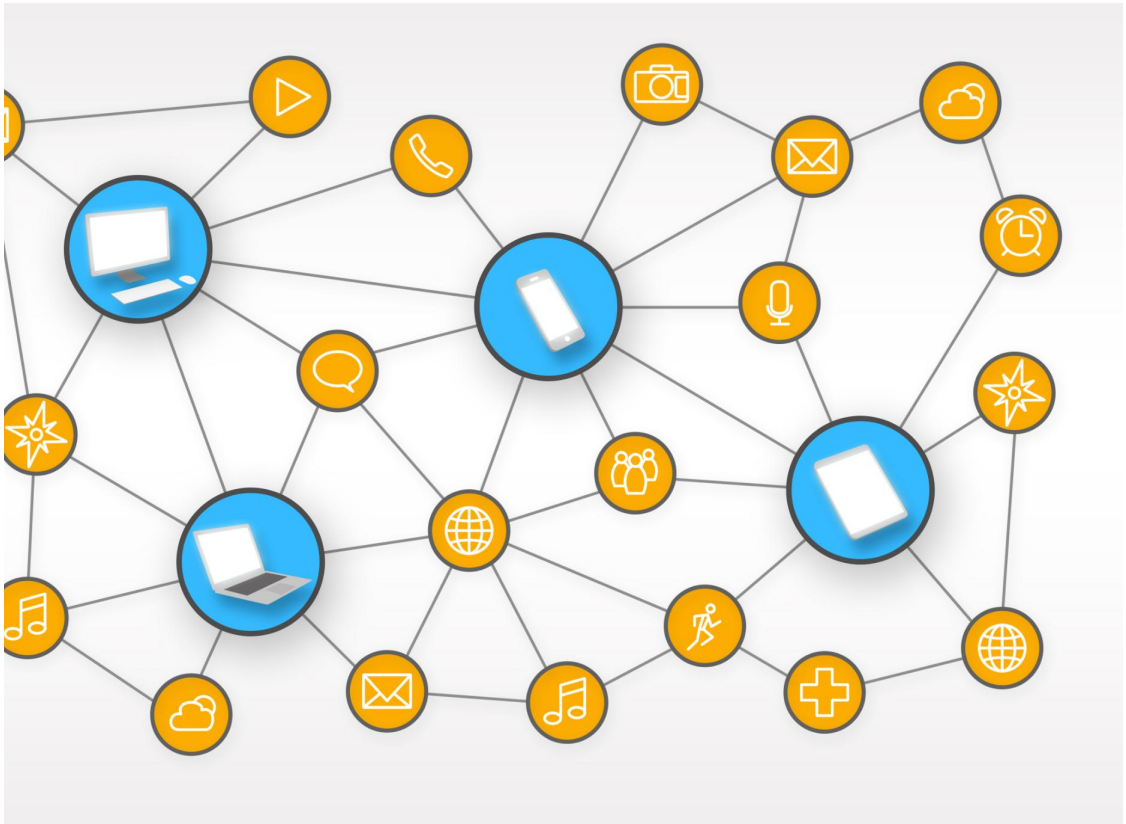
Backend Development

Backend development uses languages like Java, C#, Python, and Node.js to build robust server-side logic.

Middleware and APIs

Middleware and APIs including REST, GraphQL, and SOAP enable communication between system components.

Enterprise Software Architecture



Service-Oriented Architecture (SOA)

Modular Service Design

SOA structures systems as independent services, each handling specific business functions to improve modularity and flexibility.

Interoperable Communication

Services communicate through standardized protocols like SOAP and REST, enabling seamless integration across platforms.

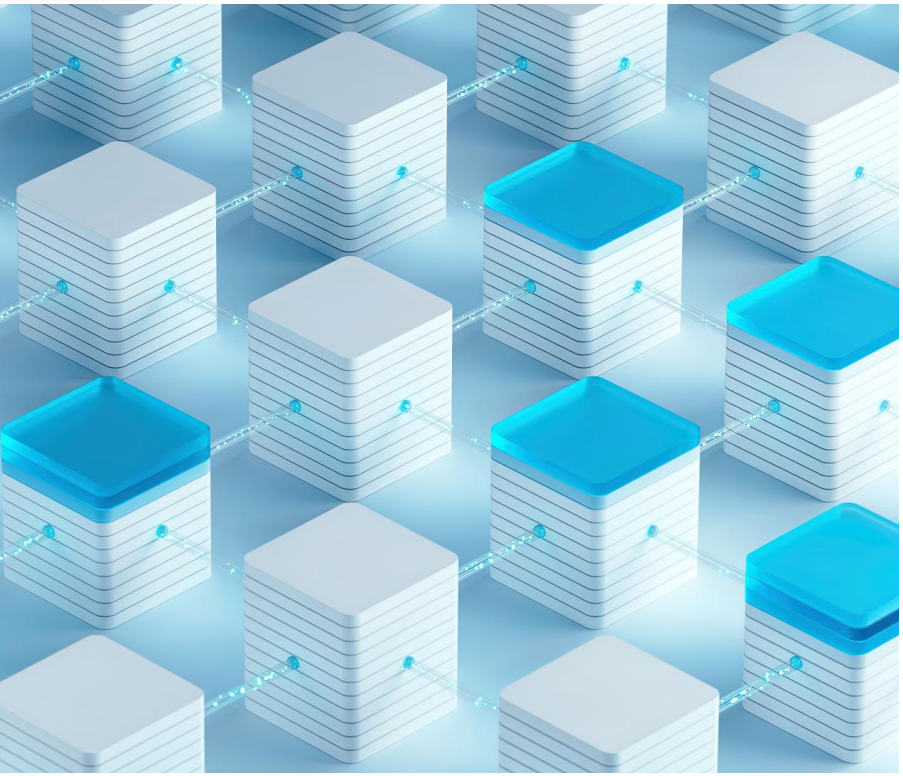
Reuse and Scalability

Reusable services reduce redundancy and development effort, promoting scalability and faster adaptation to business changes.

Business and IT Alignment

SOA encapsulates business logic in services, fostering better alignment between IT solutions and organizational goals.

Layered Architecture



Presentation Layer Role

Handles user interface and interactions, enabling effective communication between users and the system.

Business Logic Layer Function

Encapsulates core application rules and processes user inputs to generate appropriate responses.

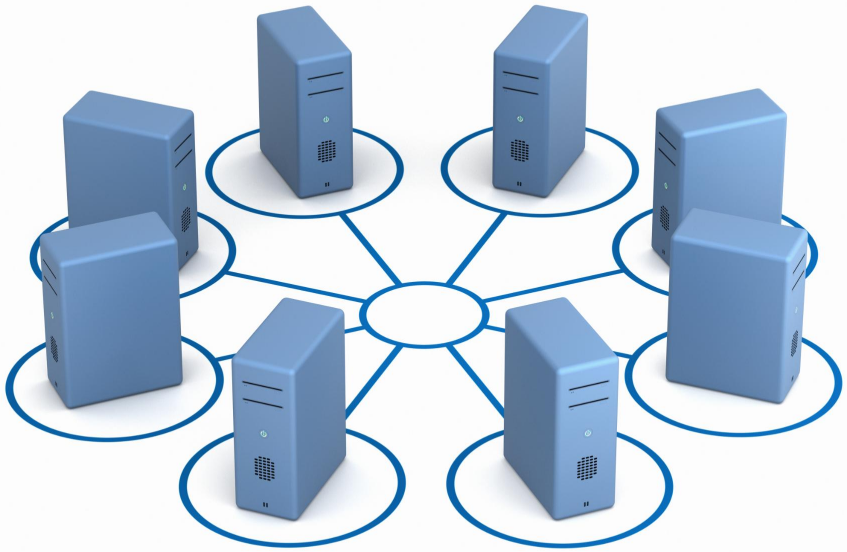
Data Access Layer Purpose

Manages database interactions, abstracting data storage and retrieval complexities for other layers.

Benefits of Layered Architecture

Enhances maintainability, scalability, and collaboration by separating concerns and allowing independent layer updates.

Microservices vs Monoliths



Monolithic Architecture

Monoliths consolidate all functionalities into one codebase, simplifying initial development but complicating scaling and updates.

Microservices Architecture

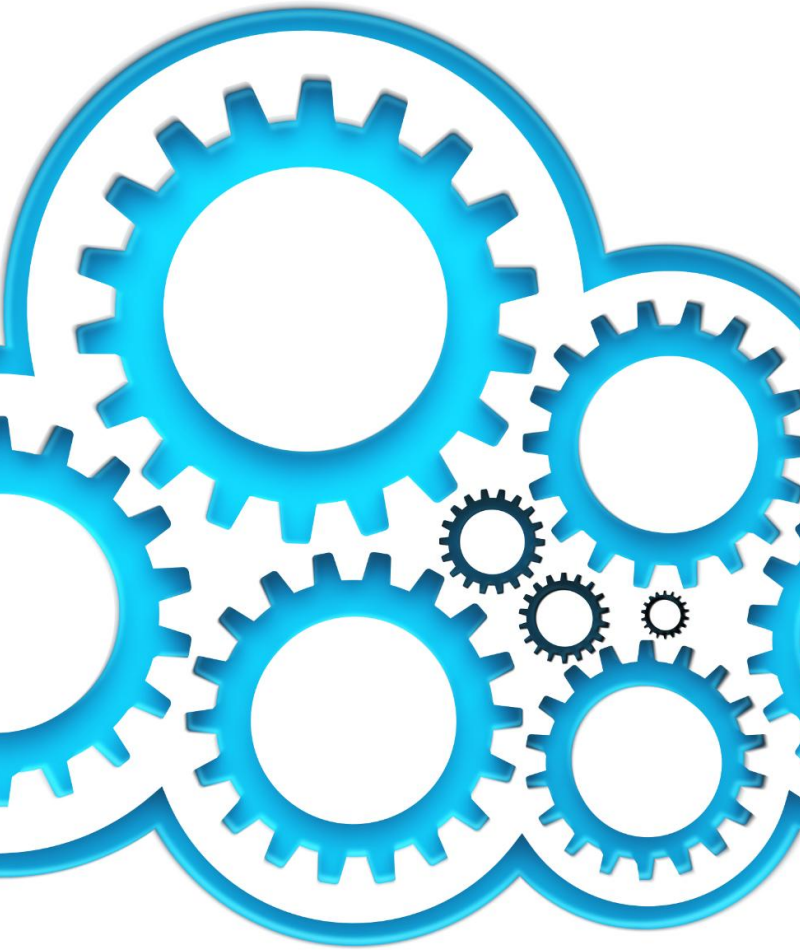
Microservices break the system into independent services, improving scalability, fault isolation, and flexibility.

Trade-offs and Challenges

Microservices introduce complexity in communication and monitoring, requiring robust orchestration compared to monoliths.

Choosing the Right Architecture

Selection depends on team size, system complexity, and scalability needs, balancing simplicity and flexibility.



Event-Driven Architecture

Core Concept of EDA

EDA enables communication through events indicating significant state changes between decoupled components.

Benefits of EDA

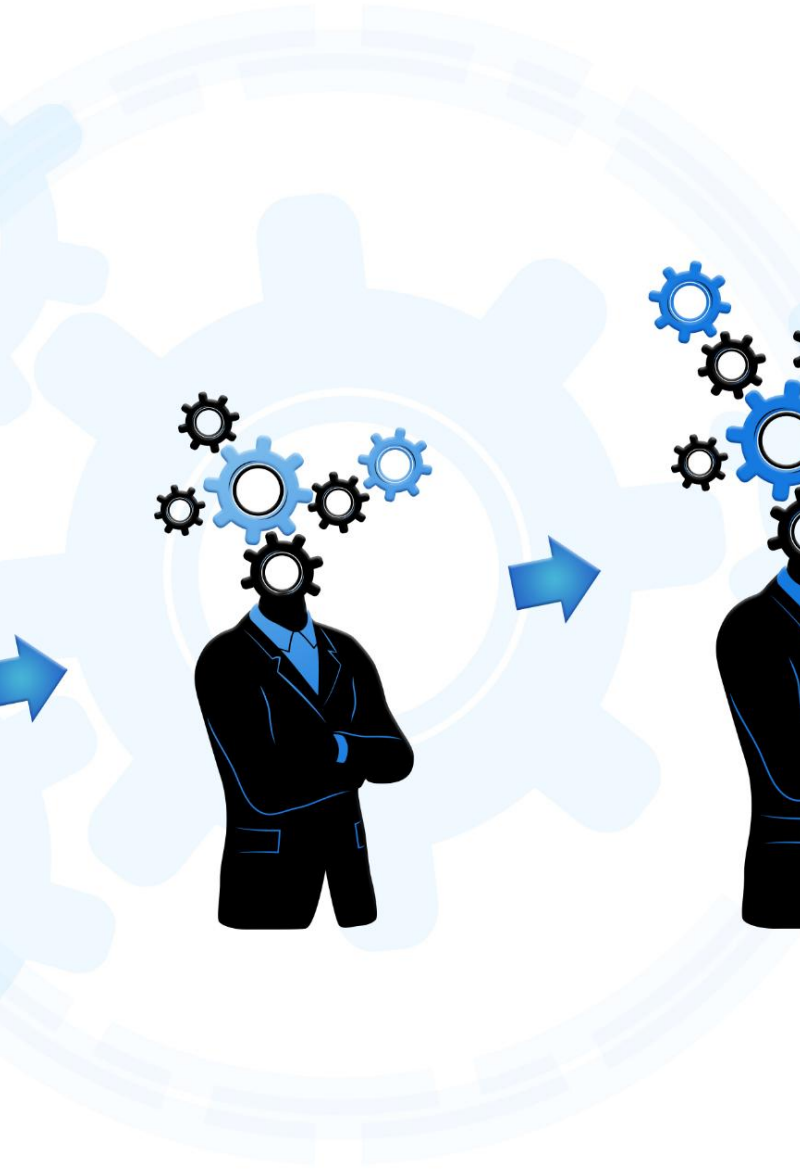
EDA improves scalability, flexibility, responsiveness, and fault tolerance in real-time distributed systems.

Implementation Infrastructure

Message brokers and event stores provide the necessary infrastructure for managing and processing events efficiently.

Challenges of EDA

Handling event complexity and ensuring consistency are key challenges when adopting event-driven architecture.



Domain-Driven Design (DDD)

Collaborative Domain Modeling

DDD promotes close collaboration between developers and domain experts to create accurate domain models.

Ubiquitous Language

A shared vocabulary bridges the gap between technical and business stakeholders for clear communication.

Bounded Contexts

Systems are divided into bounded contexts to manage complexity and maintain clarity in domain logic.

Core Building Blocks

Entities, value objects, aggregates, and repositories encapsulate domain logic and ensure consistency.

Middleware and API Gateways

Role of Middleware

Middleware bridges different applications, enabling message routing, data transformation, and protocol translation.

Common Middleware Types

Enterprise service buses, message brokers, and integration platforms are typical middleware solutions.

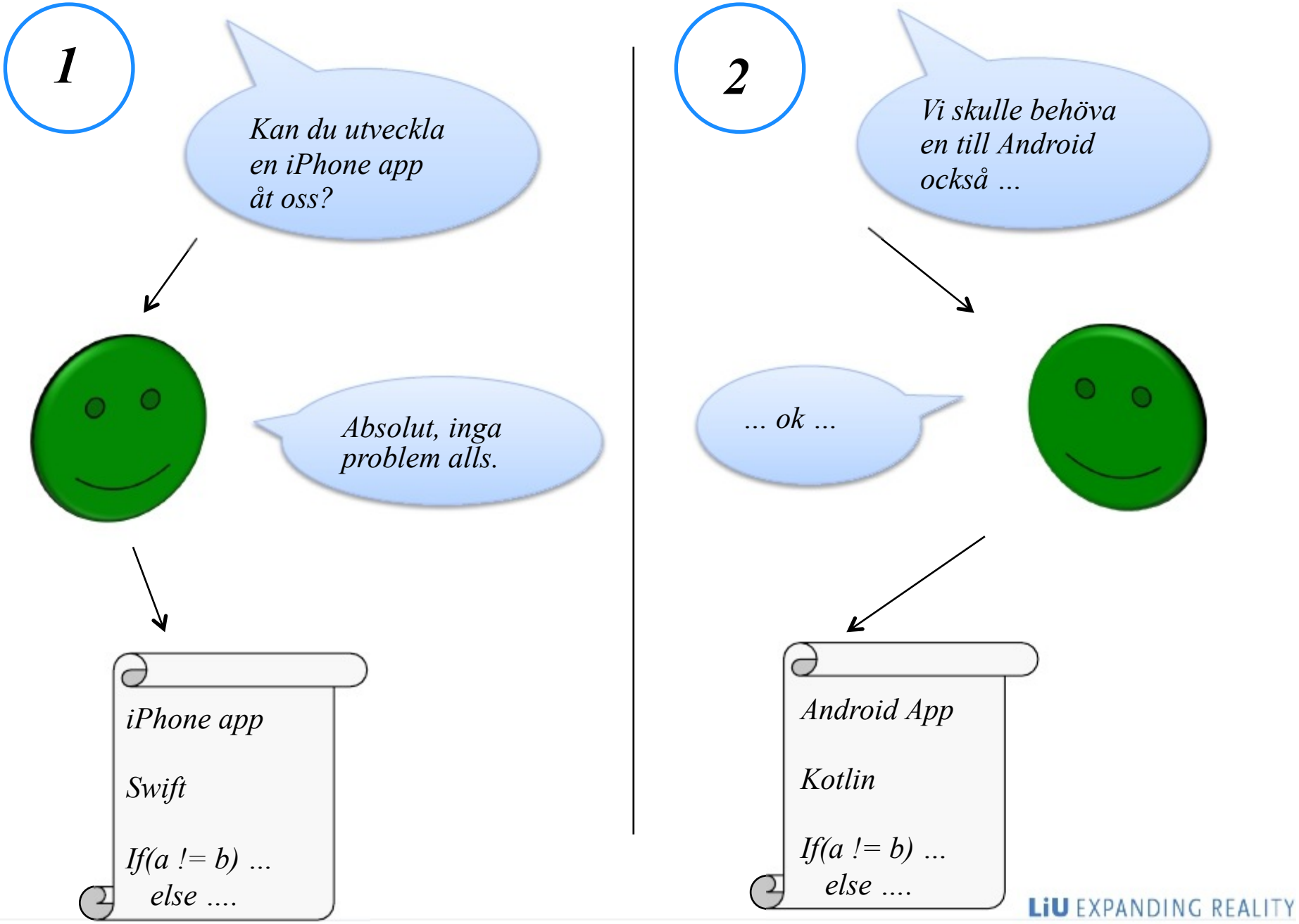
API Gateway Functions

API gateways manage client requests, routing, authentication, rate limiting, and monitoring for backend services.

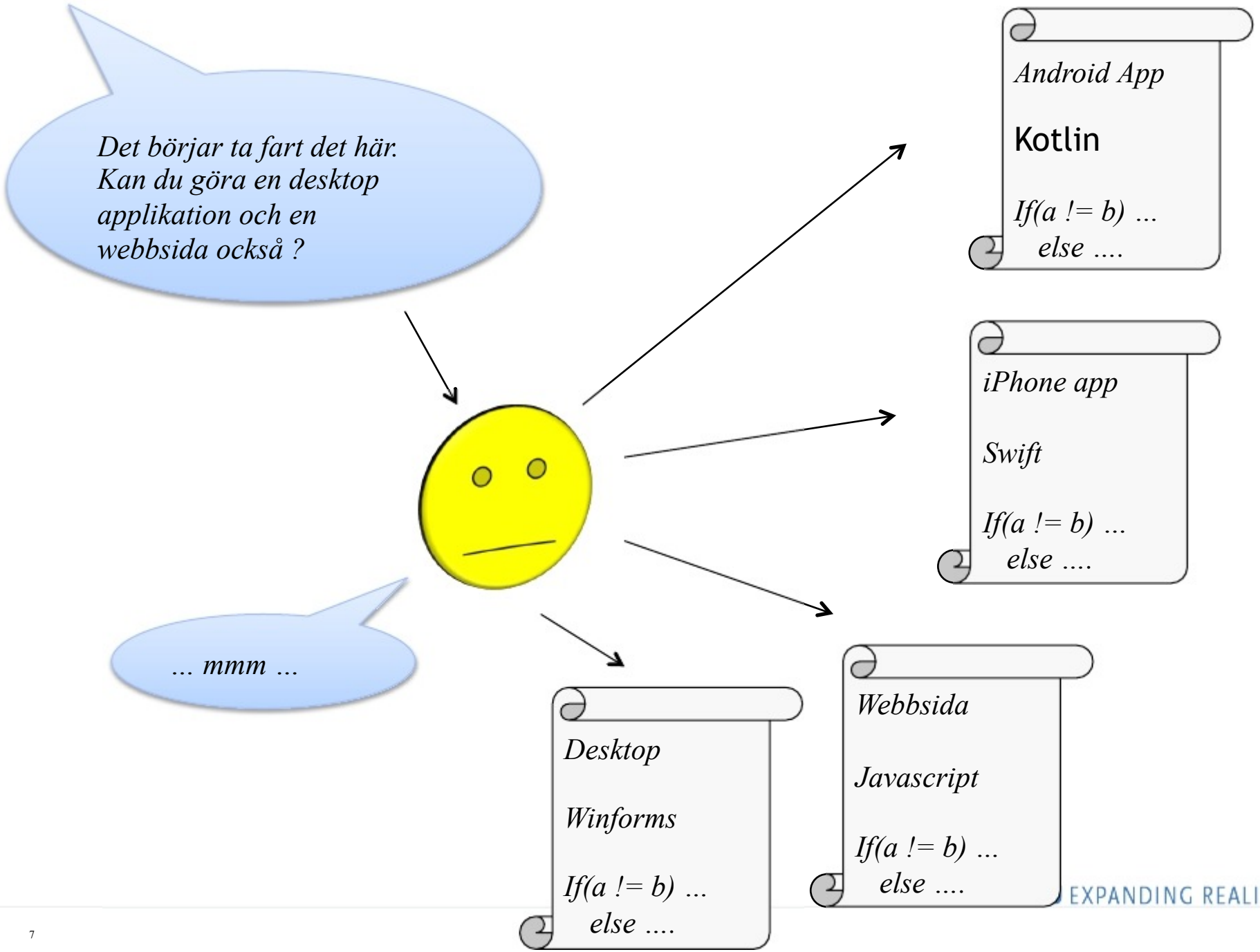
Security and Simplification

API gateways enhance security by enforcing access controls and simplify client interactions by aggregating service calls.

Låt mig berätta en historia



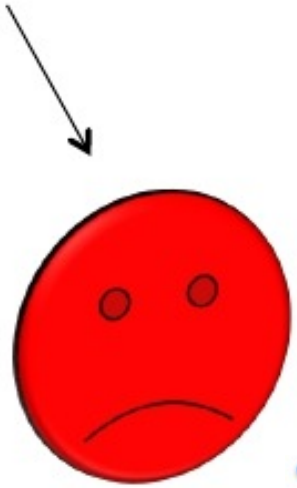
(Eller flutter.io/react-native , men då spricker mitt exempel)


Det börjar ta fart det här.
Kan du göra en desktop
applikation och en
webbsida också ?
... mmm ...
Android App
Kotlin
If(a != b) ...
else ....
iPhone app
Swift
If(a != b) ...
else ....
Webbsida
Javascript
If(a != b) ...
else ....
Desktop
Winforms
If(a != b) ...
else ....

Vi har pratat med våra kunder, de tycker att ”a” skall vara lika med ”b” …

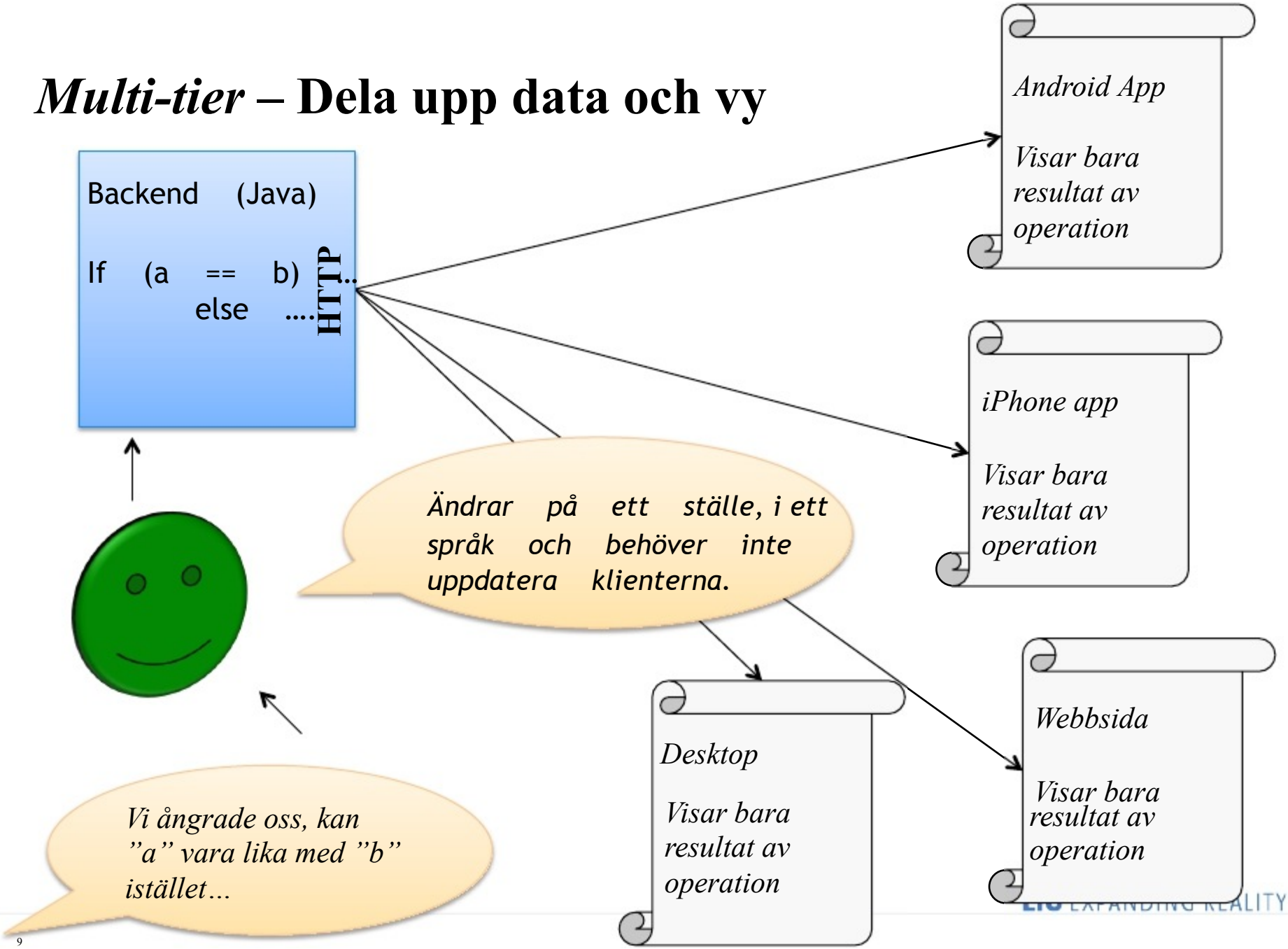
Fyra applikationer, alla skriva i olika språk som nu skall ändra sin logik.

Sedan måste jag uppdatera mina applikationer till ”Play” och ”Appstore”, webbservern skall uppdateras och alla som har installerat programmet på sin desktop måste också uppdatera …



Vad var det nu DRY stod för igen ?

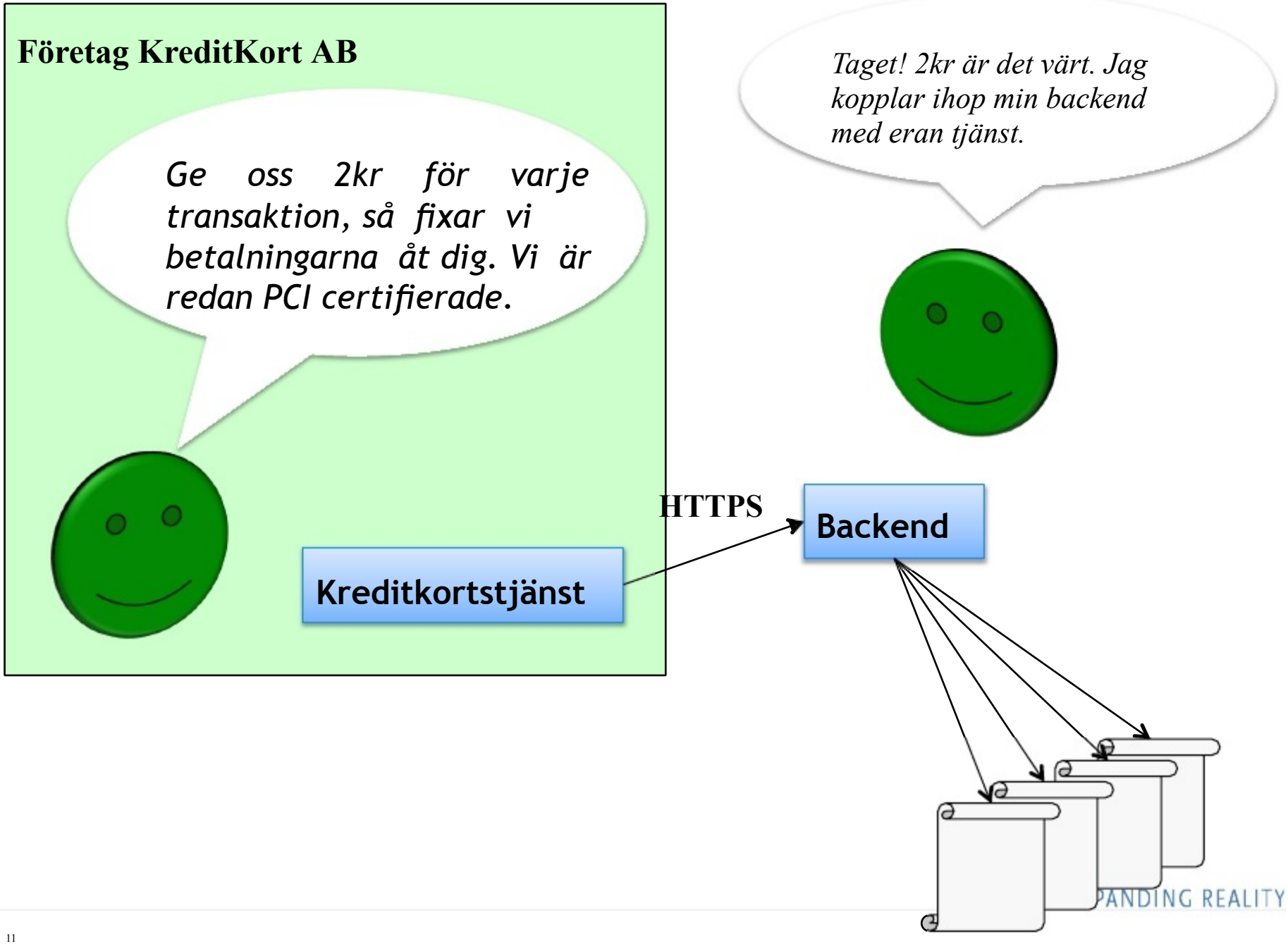
Multi-tier – Dela upp data och vy



Kunderna älskar applikationerna, vi har tänkt att vi ska ha försäljning i applikationerna. Fixar du kreditkortsbetalningar ?

PCI security standards är komplicerade. Det kräver aK de synar mina system i sömmarna och aK jag lär mig allt om deras krav.

Kanske måste jag certifiera mina applikationer varje år ?


Företag KreditKort AB
Ge oss 2kr för varje transaktion, så fixar vi betalningarna åt dig. Vi är redan PCI certifierade.
Kreditkortstjänst
HTTPS
Backend
Taget! 2kr är det värt. Jag kopplar ihop min backend med eran tjänst.

Allt fungerar superbra! Jag råkade köpa ett C++ bibliotek för 50000kr och min chef kommer nog inte bli glad om vi inte använder det. Kan du få in det någonstans i systemet?

Min backend är skriven i Java och jag kan väldigt lite om C++

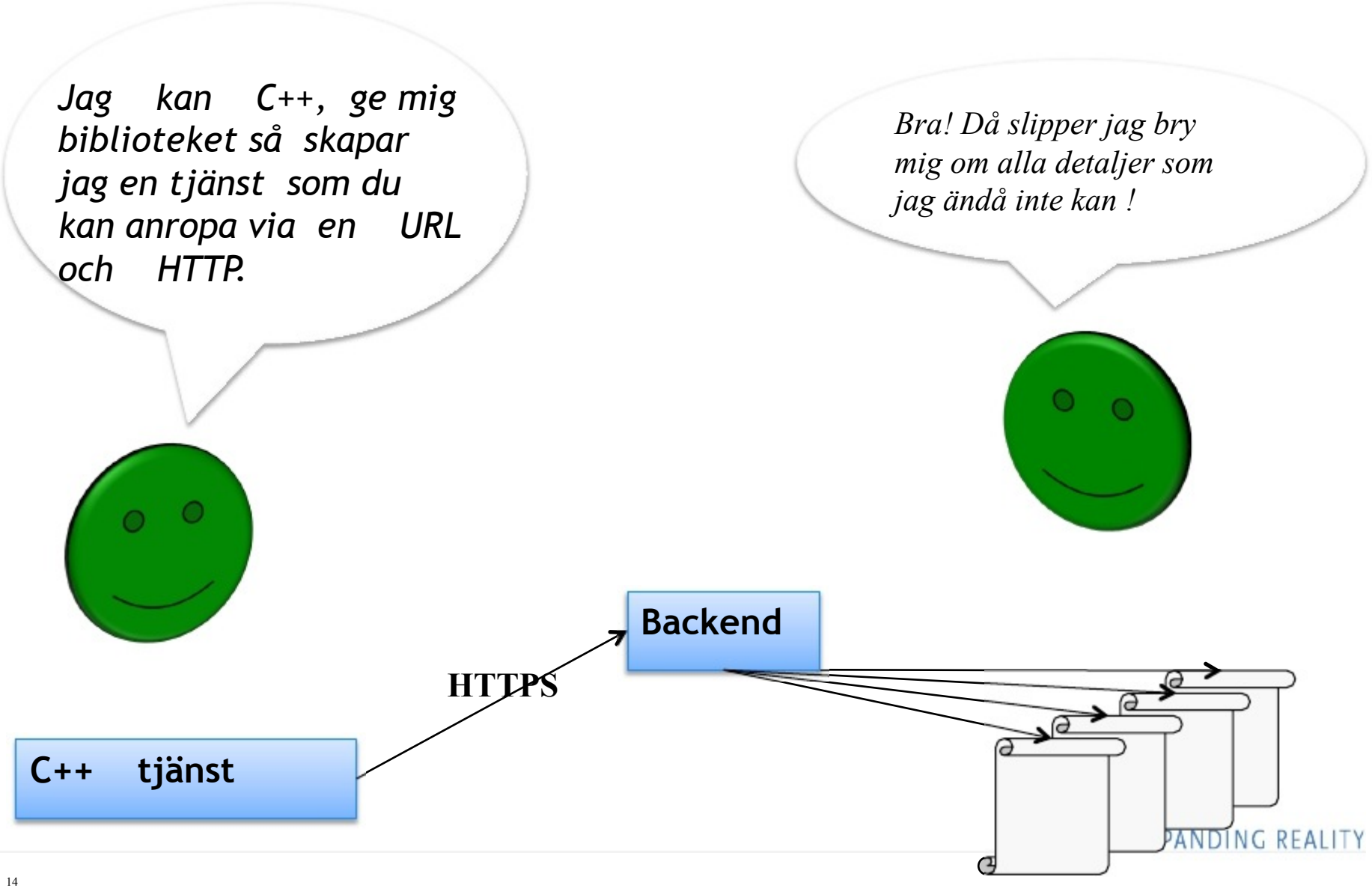
Undrar vilka ”dependencies” deras C++ bibliotek har på andra bibliotek, O/S, etc…

Det finns ju native i Java som kanske kan köra C++ koden, men jag är inte säker hur stabilt det är.

Vad händer om de ändrar i C++ koden ?

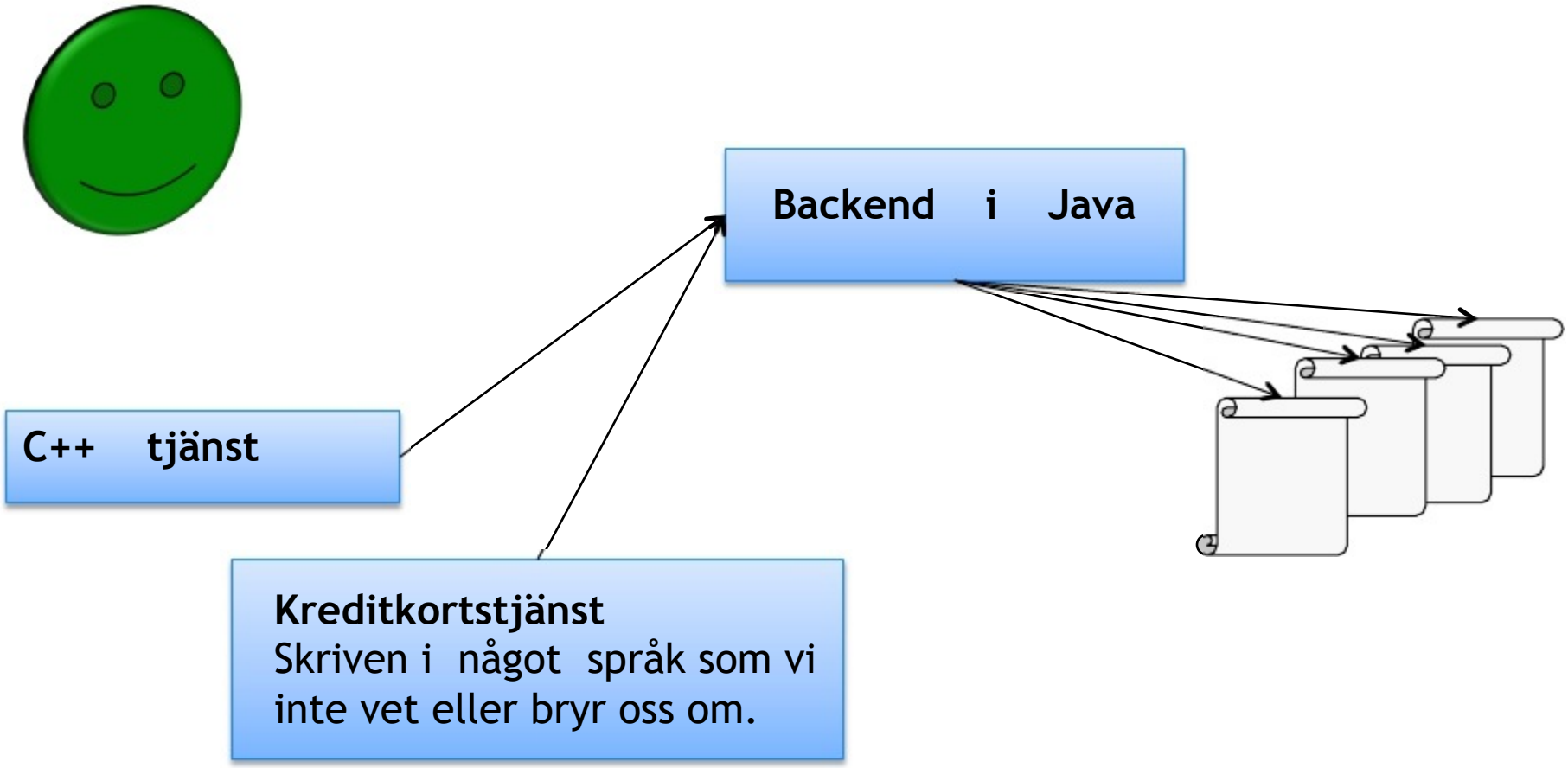
Vad var det KISS stod för nu igen ?

Använd kompetensen inom företaget!



SOA – Service Oriented Architecture

Outsourcing av moduler inom och utom företaget



Intermezzo – SOA och HTTP

- Det är inte nödvändigt att använda HTTP i SOA. Man kan använda vad som helst för system för att kommunicera över nätverk.
- System som CORBA, DCOM har använts förr men visar sig vara onödigt komplicerade. De är inte heller framtidskompatibla, (CORBA i iPhone?)
- Vi använder HTTP i våra SOA tjänster, och det är det vanligaste att man gör idag.
- Dock måste man tänka på att existerande tjänster kan ha utvecklats under en period då CORBA eller DCOM var populärt.

Intermezzo – REST och SOAP

REST

- Vi anropar vår backend genom att konstruera URL:er

  GET example.com/account/124/username

- Returvärde ofta i JSON eller XML

- Om man vill skicka med parametrar gör man det som vanligt:

  GET example.com/account?id=3

Old school such as **SOAP**

- Ett strikt protokoll där information skickas fram och tillbaka med XML

  POST example.com
  …. xml dokument …

- XML dokumentet beskriver vilken metod som skall exekveras och vilka parametrar som skickas.

Intermezzo – REST och SOAP - Hybrid

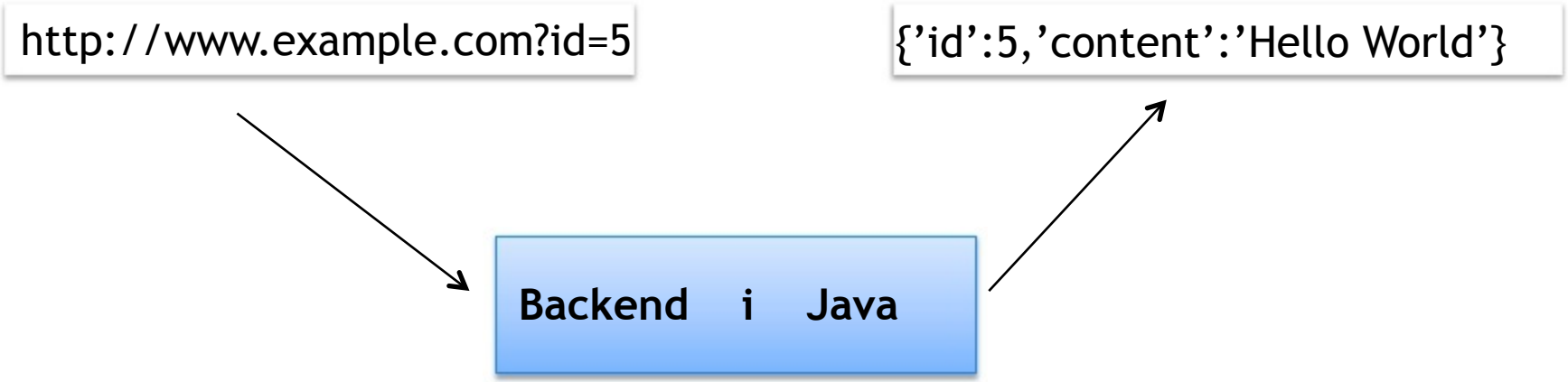
- Det vanliga är att man blandar dessa lite, en slags hybrid.
- Amazon EC2 och Route 53 använder en hybrid där man ibland skickar skickar en begäran som liknar REST och får tillbaka XML men ibland måste författa ett XML dokument (dock inte enligt SOAP standarden).
- Twitter har något de kallar ”GET Search”, där man själv kan knopa ihop en url för att få tillbaka en twitter feed i JSON.

  http://search.twitter.com/search.json?q=rest

Intermezzo – REST och SOAP - Denna kurs

- I denna kurs använder vi REST. SOAP är onödigt komplicerat och håller på att fasas ut i många stora system.

- Vi använder parametrar in i våra tjänster och får ut JSON (som Twitter GET search).

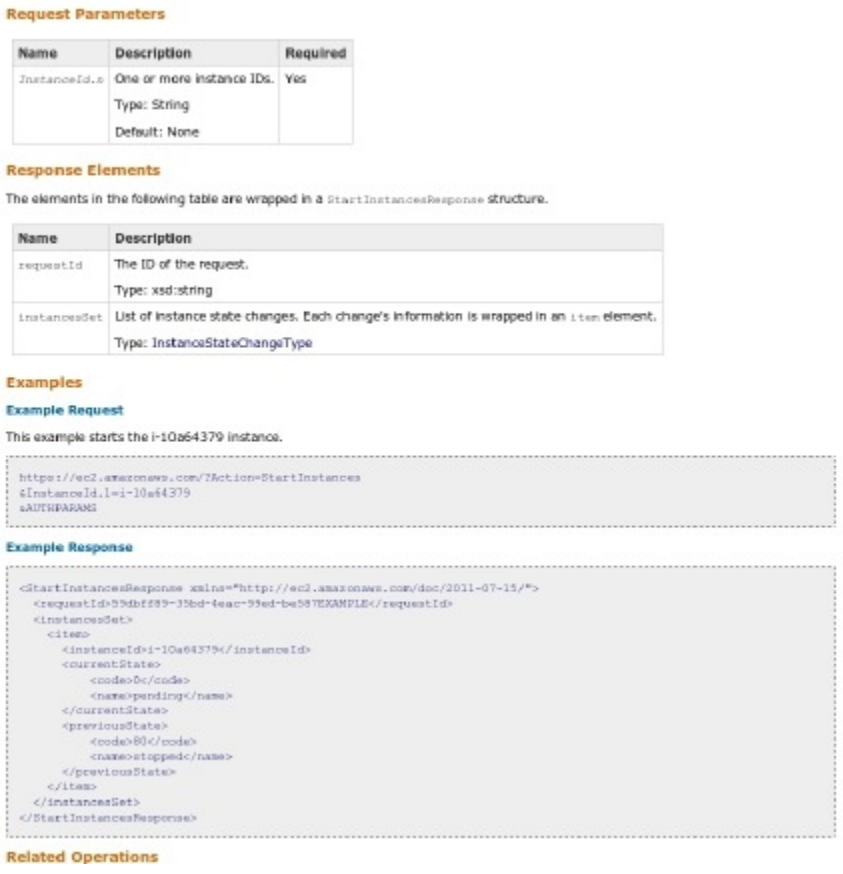


SOA - Principer

- SOA har inget bibliotek, API, kontrakt eller regelverk som man måste följa.
- SOA är ett sätt att dela in ett system i små delar som alla kan kommunicera.
- Hur man gör detta är upp till individen eller företaget.
- Trots detta finns det några riktlinjer, eller principer, som man bör följa för att lyckas med sin SOA.
- Notera dock att SOA tjänster har utvecklats de senaste 10-20 åren, och man kan i verkligheten stöta på äldre SOA tjänster som inte följer dessa principer.

SOA – Principer - Kontrakt

- En tjänst behöver någon typ av dokumentation som talar om hur man anropar tjänsten.



Request Parameters

Name	Description	Required
InstanceId.n	One or more instance IDs. Type: String Default: None	Yes

Response Elements

The elements in the following table are wrapped in a StartInstancesResponse structure.

Name	Description
requestId	The ID of the request. Type: xsd:string
instancesSet	List of instance state changes. Each change's information is wrapped in an item element. Type: InstanceStateChangeType

Examples

Example Request

This example starts the i-10a64379 instance.

```
https://ec2.amazonaws.com/?Action=StartInstances
&InstanceId.1=i-10a64379
&AUTHPARAMS
```

Example Response

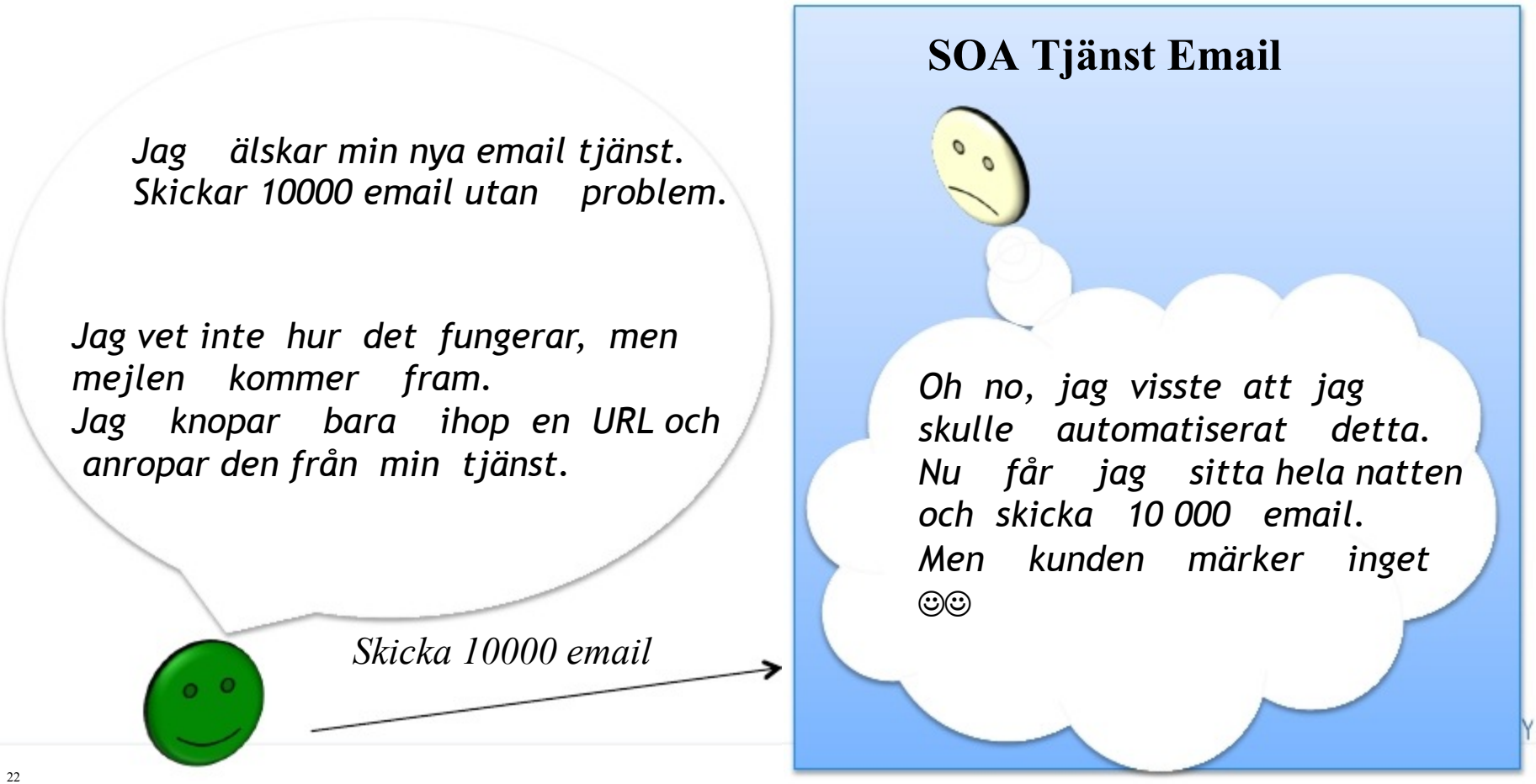
```
<StartInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2011-07-15/">
  <requestId>59dbff89-35bd-4eac-99ed-be587EXAMPLE</requestId>
  <instancesSet>
    <item>
      <instanceId>i-10a64379</instanceId>
      <currentState>
          <code>0</code>
          <name>pending</name>
      </currentState>
      <previousState>
          <code>80</code>
          <name>stopped</name>
      </previousState>
    </item>
  </instancesSet>
</StartInstancesResponse>
```

Related Operations

http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/

SOA – Principer – Abstraction and Autonomy

- Tjänsten skall ha full kontroll över sin egen logik och hålla den gömd från klienten (gömd som i abstraherad).



SOA – Principer - Stateless

- Tjänsten skall vara stateless

Kräver lite fler bilder →

Vi håller på att utöka vårt utbud, kan du skapa en ”kundvagn” ?

Det finns ju Sessions i min server mjukvara, på så vis kan servern automatiskt hålla reda på vem som är vem, och jag kan lagra varor i sessionsdata

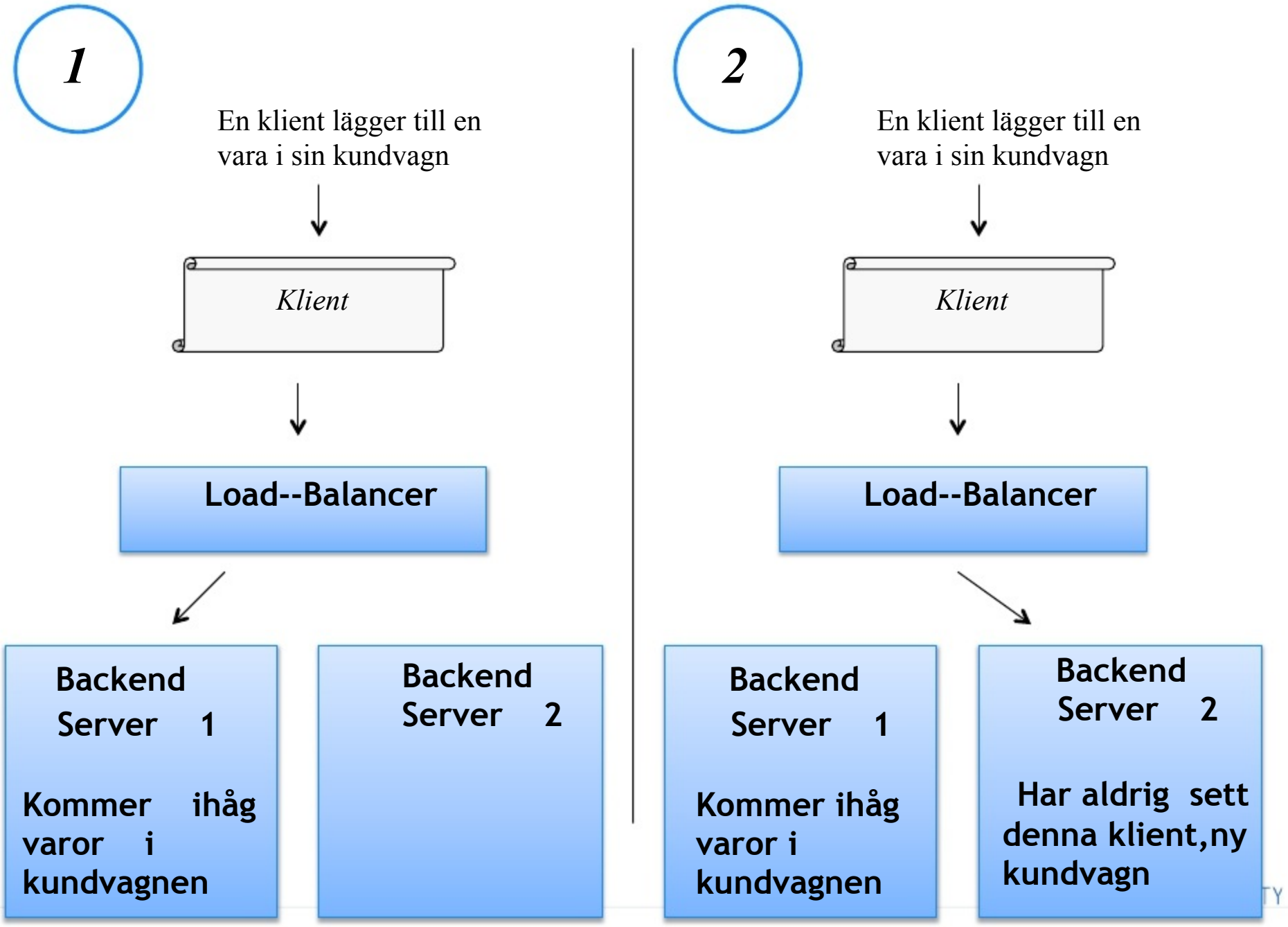
Backend

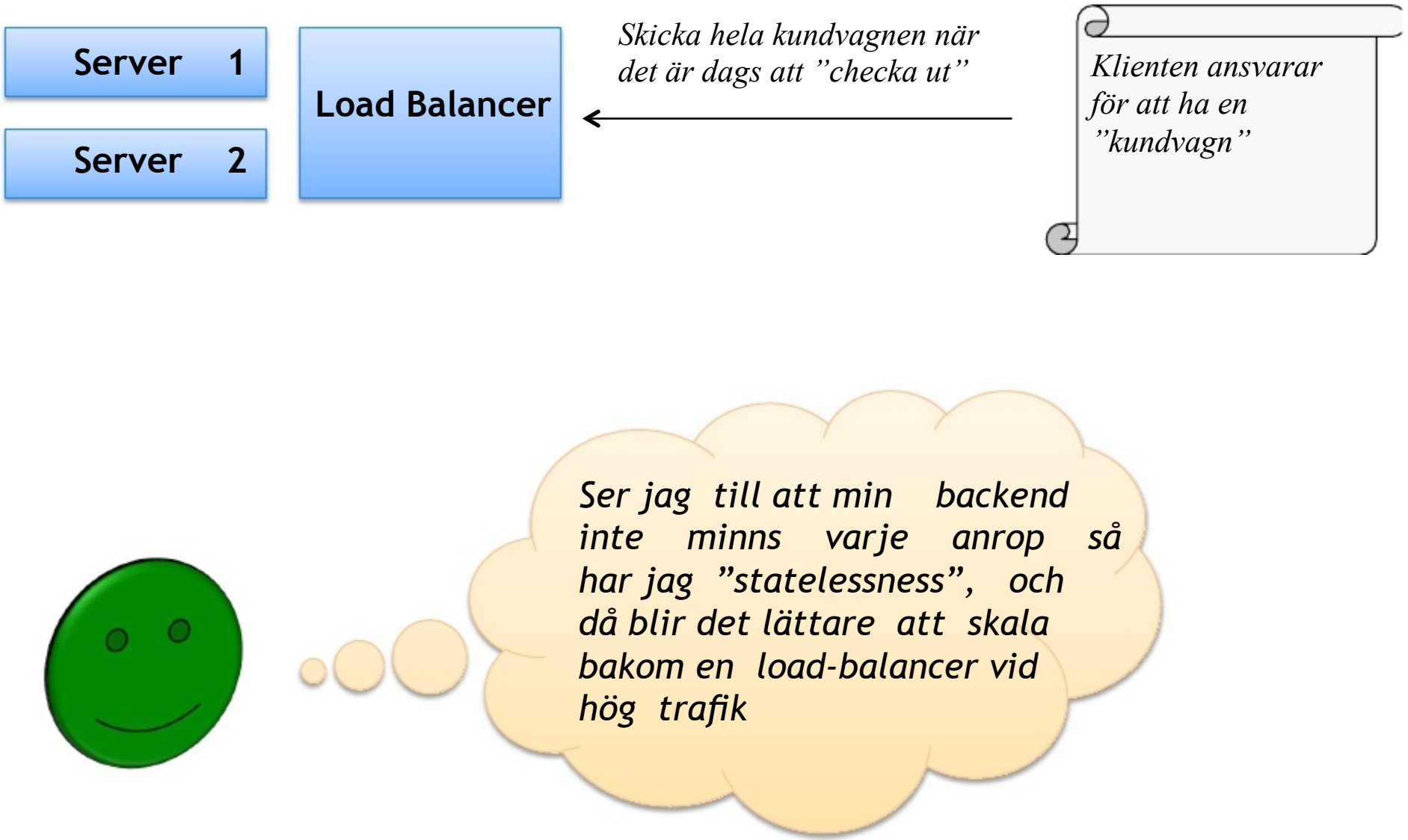
Spara sessionsdata på servern som kommer ihåg användare mellan anrop och sparar deras kundvagn

Jag vill inte spara ”korgen” i databasen, då får jag massor med död data då användare fyller korgar och inte checkar ut …

Vi har precis skaffat pengar till en tv-reklam, räkna med 3-5 gånger så mycket trafik som vanligt!

Det borde inte vara något problem om jag startar några extra servrar, skalar ut, och sätter en load-balancer mellan dessa

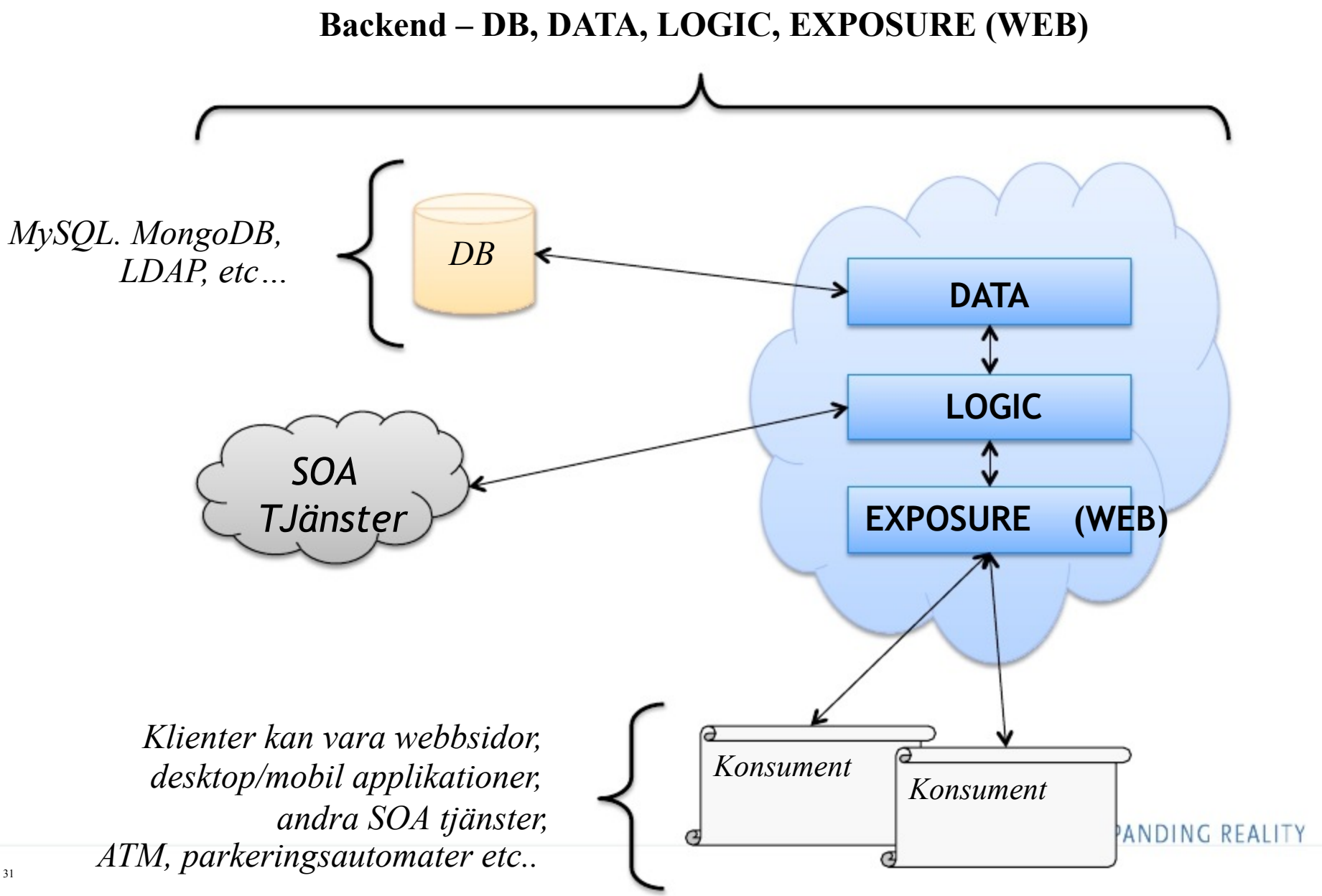

1
En klient lägger till en vara i sin kundvagn
Klient
Load--Balancer
Backend Server 1
Kommer ihåg varor i kundvagnen
Backend Server 2
2
En klient lägger till en vara i sin kundvagn
Klient
Load--Balancer
Backend Server 1
Kommer ihåg varor i kundvagnen
Backend Server 2
Har aldrig sett denna klient,ny kundvagn


Server 1
Server 2
Load Balancer
Skicka hela kundvagnen när det är dags att ”checka ut”
Klienten ansvarar för att ha en ”kundvagn”
Ser jag till att min backend inte minns varje anrop så har jag ”statelessness”, och då blir det lättare att skala bakom en load-balancer vid hög trafik

SOA Principer - Summering

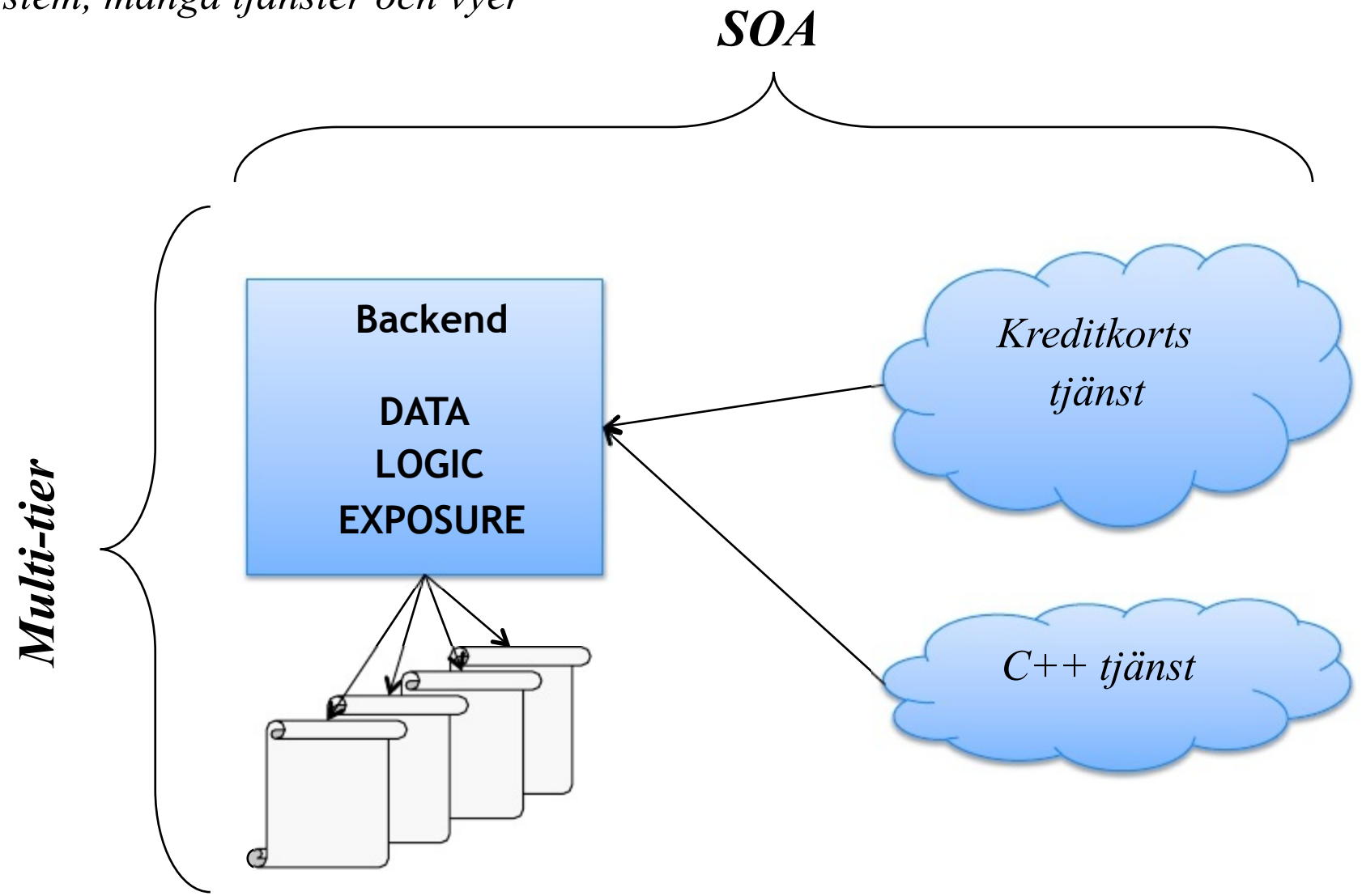
1. Contract
2. Abstraction
3. Autonomy
4. Stateless

Backend – SOA Tjänst – Egentligen flera lager



Enterprise Systems

Ett system, många tjänster och vyer



Inte bara Java

- Vi använder mycket Java i denna kurs.
- Men det går att utveckla för SOA med vilket språk som helst (som har något form av nätverksstöd).
- Det är den frihet man vill uppnå med SOA, att inte tvinga på något språk eller verktyg.
- Tänk genom hela kursen: Hur hade jag gjort detta i språk X med ramverk Y ?
- Poängen med den här kursen är att förstå arkitekturen och de problem som uppstår när man vill använda sig av just SOA.

Varför Java ?

- Utan tvekan det mest använda språket för Enterprise Systems idag.
- Vore oansvarigt att inte ge en introduktion till Java EE som del av en komplett utbildning inom programmering och/eller datavetenskap.
- Även om Oracle äger Java och därmed Java EE så har bland andra följande företag stort inflytande över utvecklingen tack vare stora investeringar av tid, utvecklare och pengar:
  Credit Suisse, Google, VM Ware, Eclipse Foundation, Hewlett-Packard, Ericsson, IBM, RedHat, Fujitsu, Intel, SAP

Java

Java SE

- Själva språket Java, dess syntax och standardbibliotek.
- Innehåller kompilator och andra verktyg.
- Koden kompileras till bytekod som exekveras i en virtuell maskin (JVM)

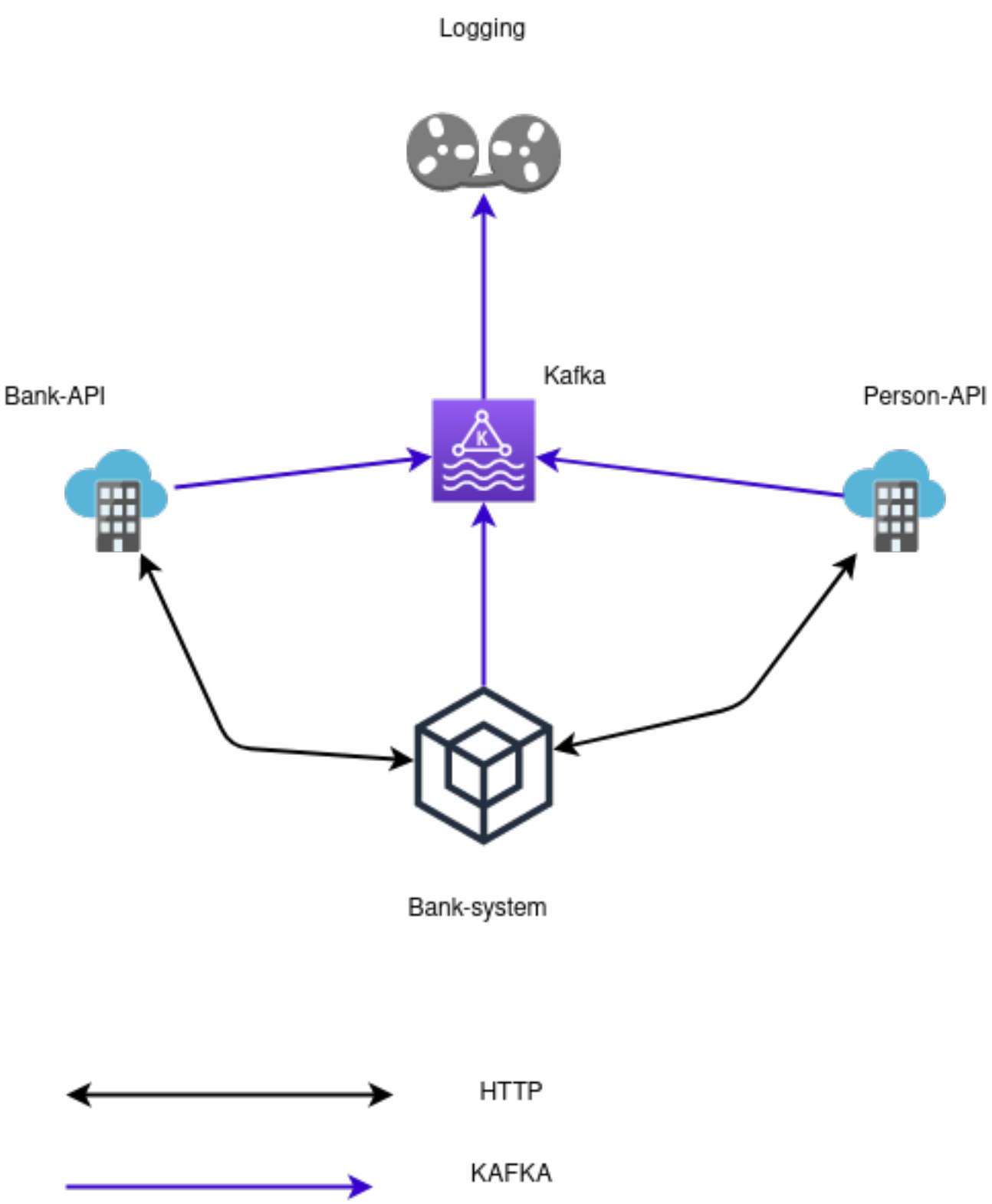
Java EE

- Bibliotek för att bygga enterprise system.
- Server för tjänster att köra på.
- De delar som behövs för att lösa vanliga problem vid ES utveckling.

Projektet

- Exakta instruktioner finns på kurshemsidan. Kom ihåg Webreg.

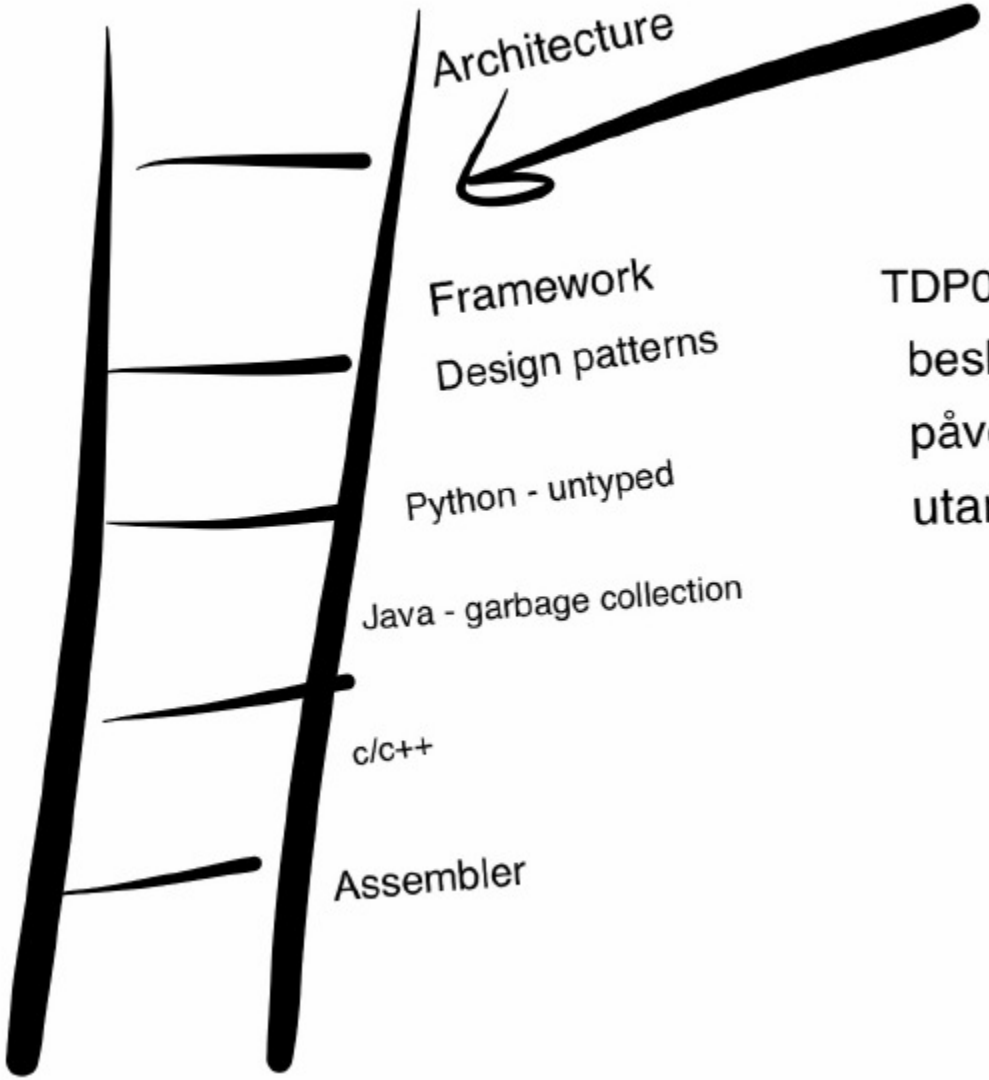
- Ni laddar ner ett ”kodskelett” från gitlab. Detta är ett antal projekt som innehåller de delarna ni behöver för att utveckla ***data***, ***logic*** och ***web***. Vi kommer i följande föreläsningar titta på exakt hur dessa utvecklas.


Logging
Kafka
Bank-API
Person-API
Bank-system
HTTP
KAFKA

Betyg 4 och 5

- Betyg 4 – Utöka ditt projekt felhantering (bara java koden)
- Betyg 5 - Betyg 4 + En avancerad del av Laboration 1 eller 2
- Redovisning sker genom muntlig presentation i laborationssalen till er assistent samt manuell genomgång av er kod av kursledning (kolla hemsidan för mer detaljer).

Abstraktion



TDP024 hamnar högt i abstraktionen,
besluten vi tar i våra scenarion
påverkar inte bara oss själva
utan hela organisationen.