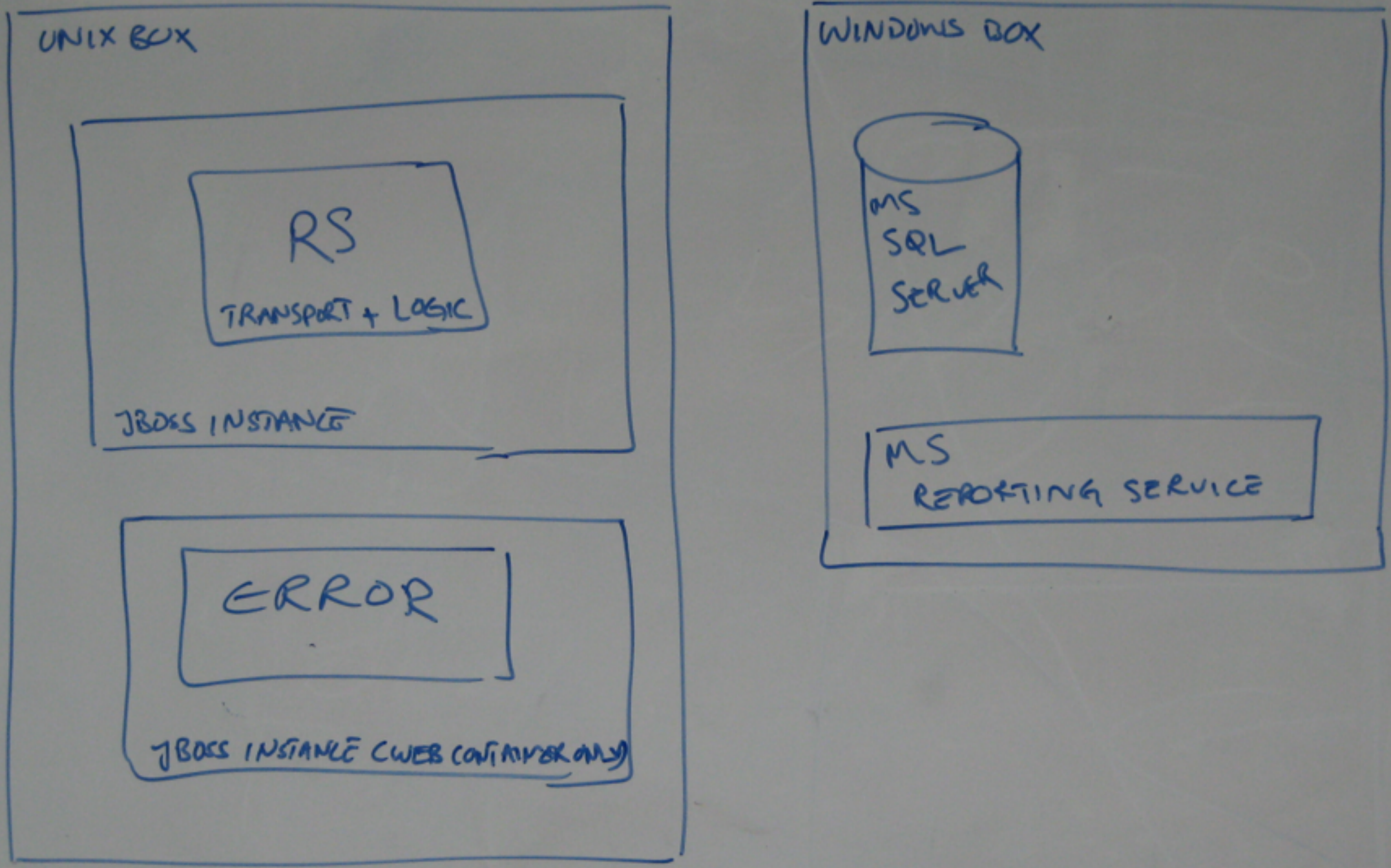
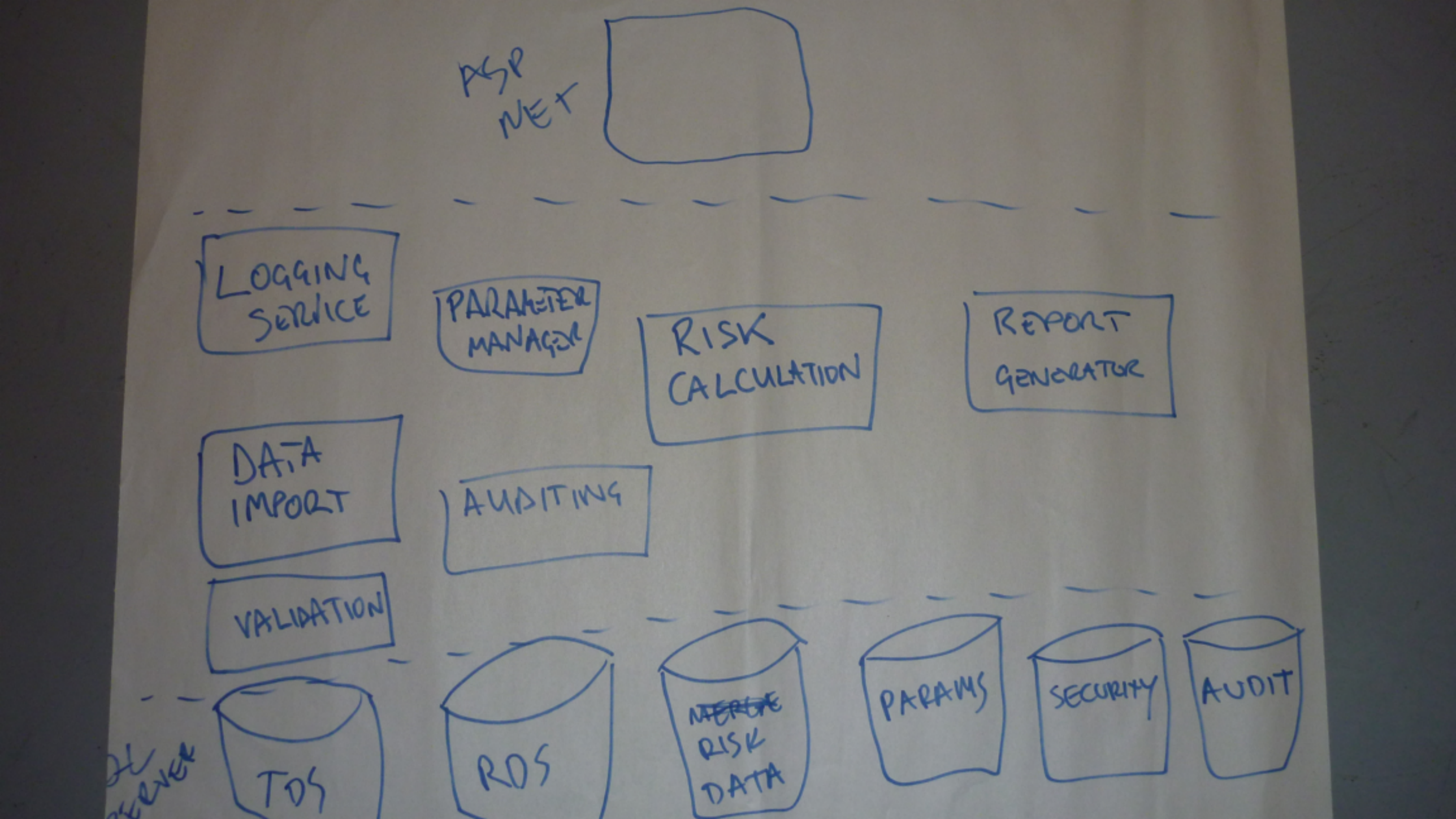
Visualising software architecture with the C4 model

Simon Brown
@simonbrown

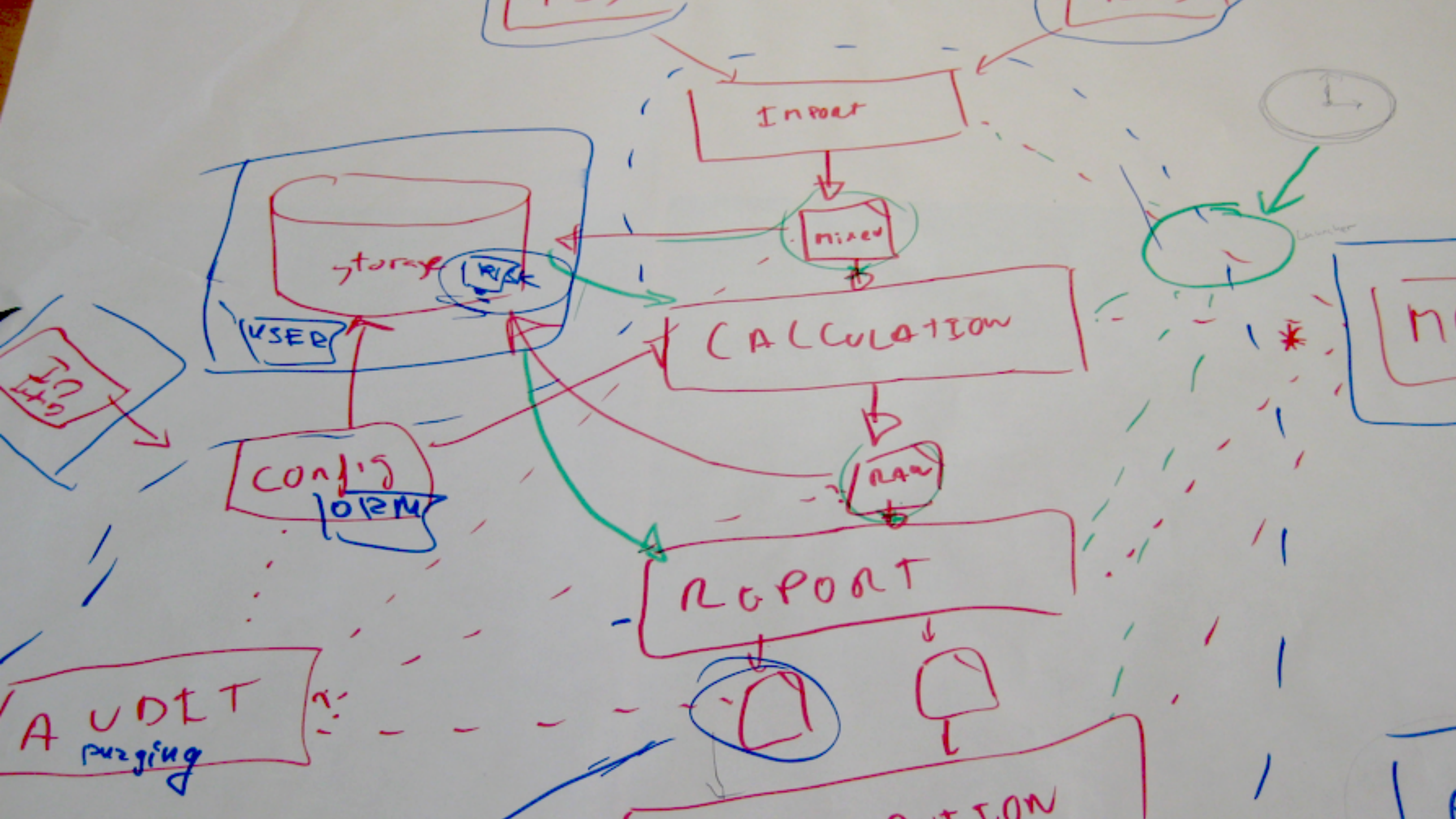
Over the past decade, many teams have thrown away big design up front

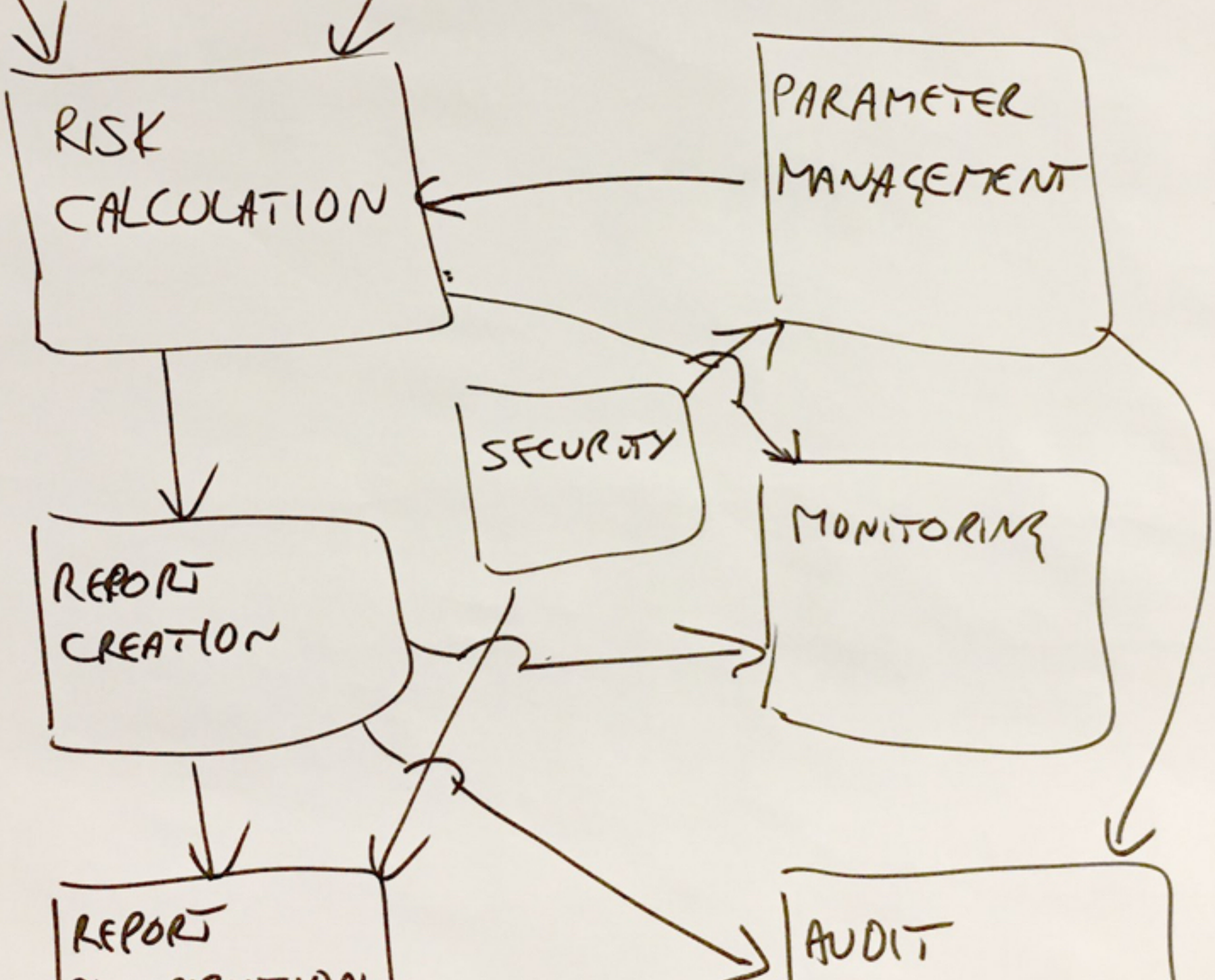
Unfortunately, architectural thinking, documentation, diagramming, and modelling were also often discarded

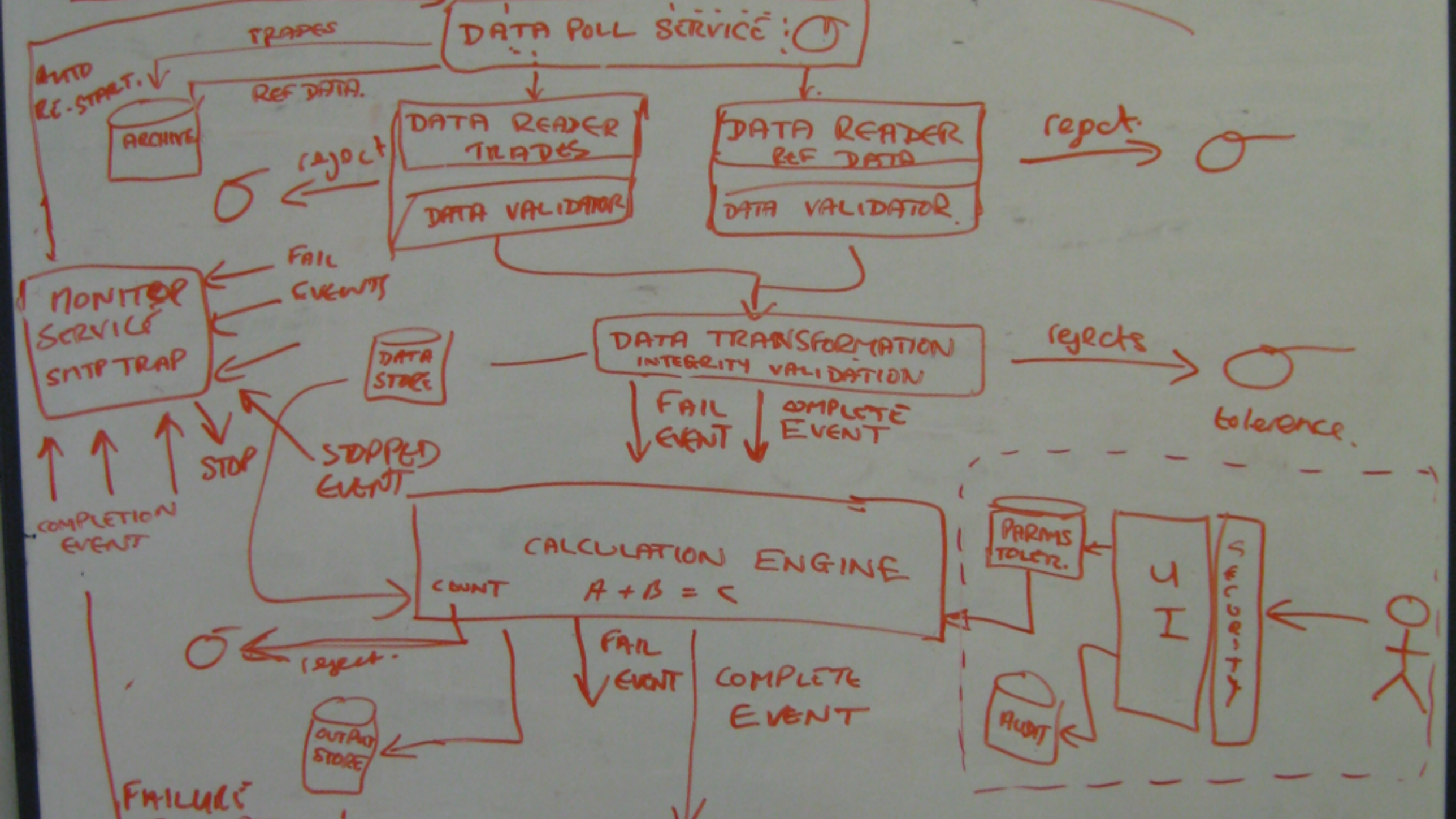

UNIX BOX
RS
TRANSPORT + LOGIC
JBOSS INSTANCE
ERROR
JBOSS INSTANCE (WEB CONTAINER ONLY)
WINDOWS BOX
MS
SQL
SERVER
MS
REPORTING SERVICE

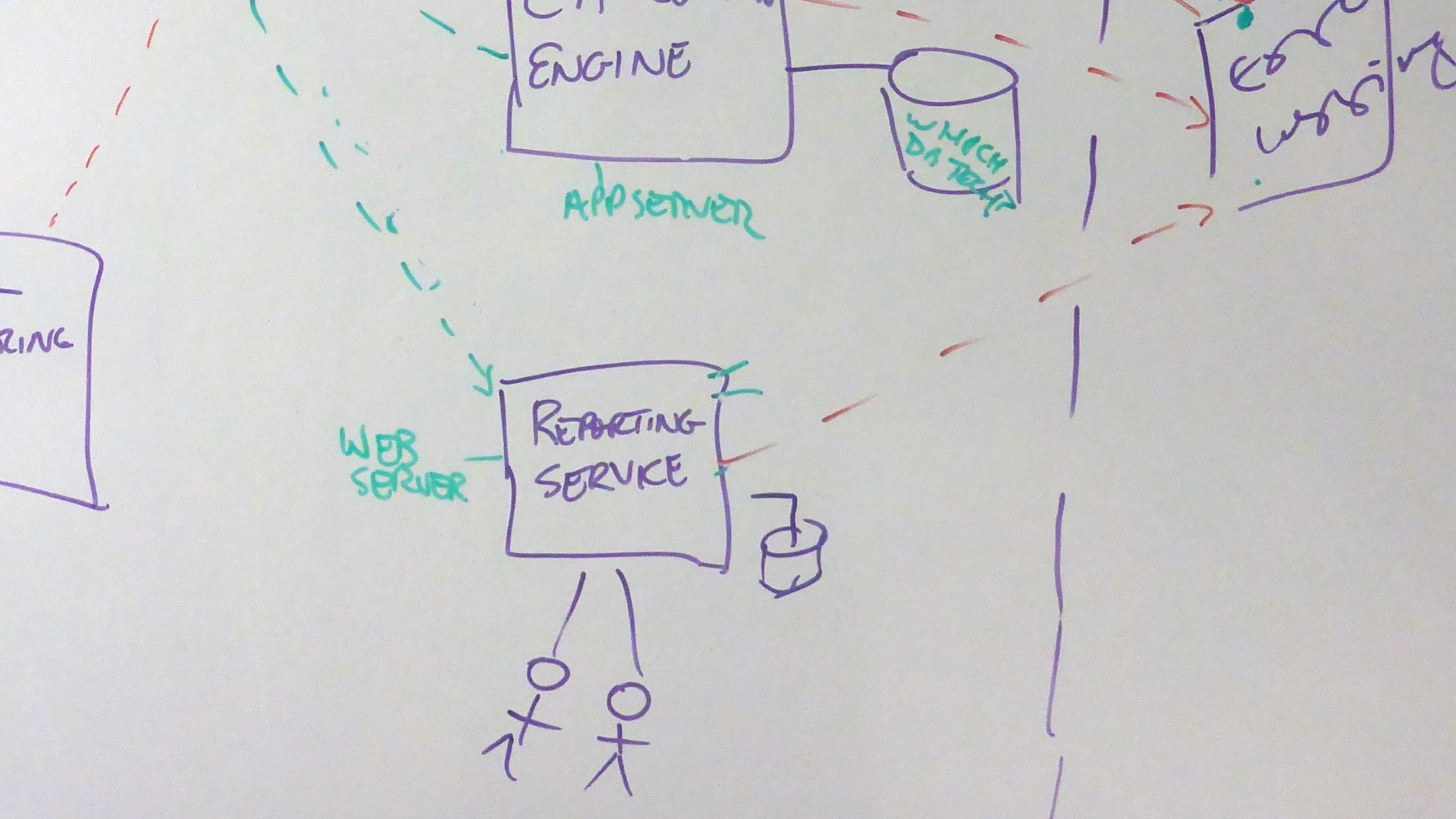

ASP NET
LOGGING SERVICE
PARAMETER MANAGER
RISK CALCULATION
REPORT GENERATOR
DATA IMPORT
AUDITING
VALIDATION
TDS
RDS
RISK DATA
PARAMS
SECURITY
AUDIT

FUNCTIONAL VIEW


Import
storage
risk
USER
config
mixed
CALCULATION
plan
REPORT
AUDIT
purging


RISK
CALCULATION
PARAMETER
MANAGEMENT
SECURITY
MONITORING
REPORT
CREATION
AUDIT

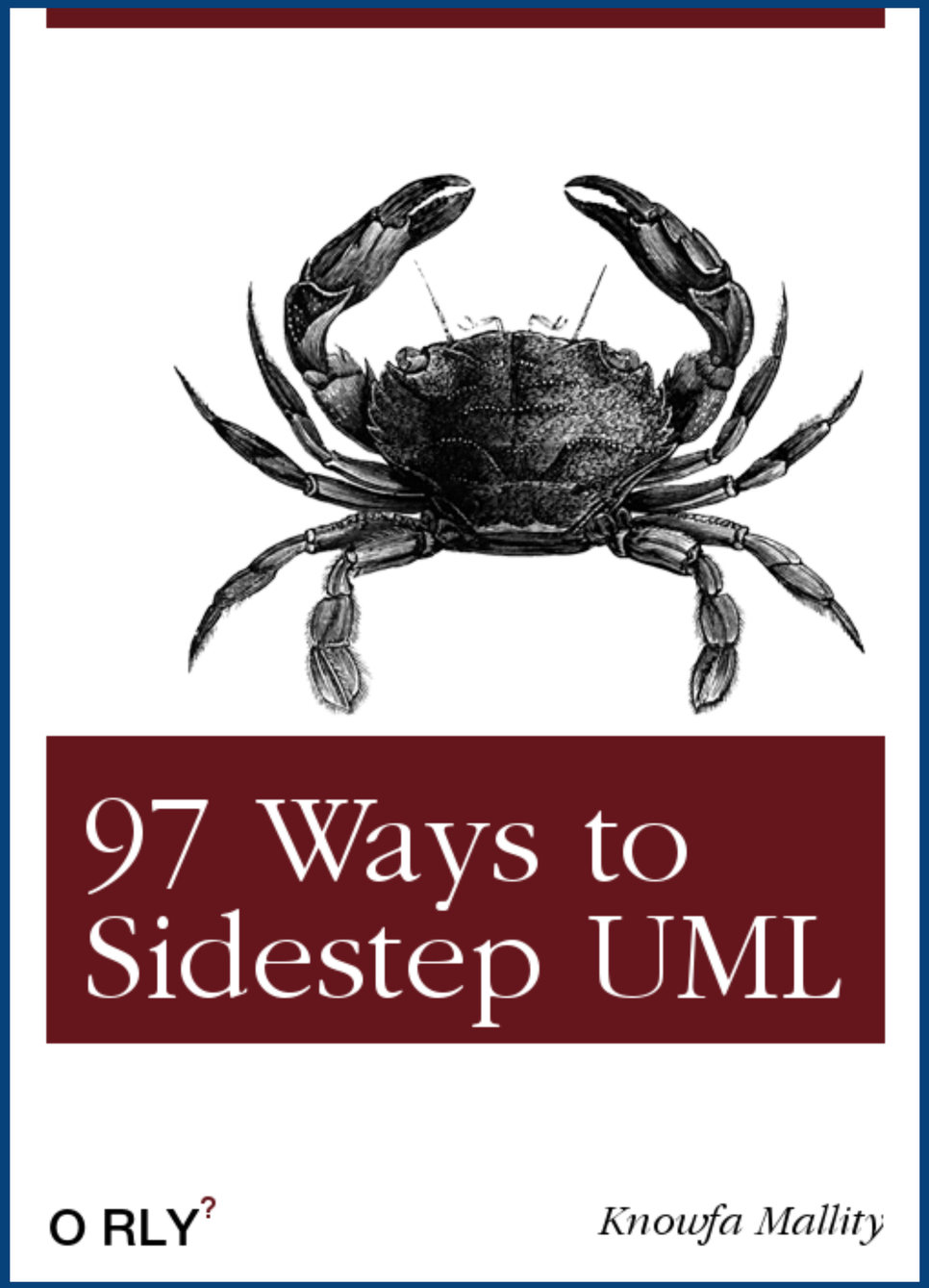

DATA POLL SERVICE
TRADES
REF DATA.
AUTO RE-START.
ARCHIVE
DATA READER TRADES
DATA VALIDATOR
reject
DATA READER REF DATA
DATA VALIDATOR
reject
MONITOR SERVICE SMTP TRAP
FAIL EVENTS
DATA STORE
DATA TRANSFORMATION
INTEGRITY VALIDATION
rejects
tolerance
FAIL EVENT
COMPLETE EVENT
STOP
STOPPED EVENT
COMPLETION EVENT
CALCULATION ENGINE
A + B = C
COUNT
PARMS TOLER.
UI
SECURITY
reject
FAIL EVENT
COMPLETE EVENT
OUTPUT STORE
AUDIT
FAILURE


ENGINE
APP SERVER
RING
WEB SERVER
REPORTING SERVICE


Report Monitor
(other machine)
Data Normalizer
Risk Calculator
Aggregator
CP Risk Calc
Report Generator
Auditing Service
Report Publisher
Report Repository


Params
Calcs
EH?
Batch
Date
- RiskCal
- calcs
- Par

UML?



#2 “Not everybody else on the team knows it.”

#3 “I’m the only person on the team who knows it.”

#36 “You’ll be seen as old.”

#37 “You’ll be seen as old-fashioned.”

#66 “The tooling sucks.”

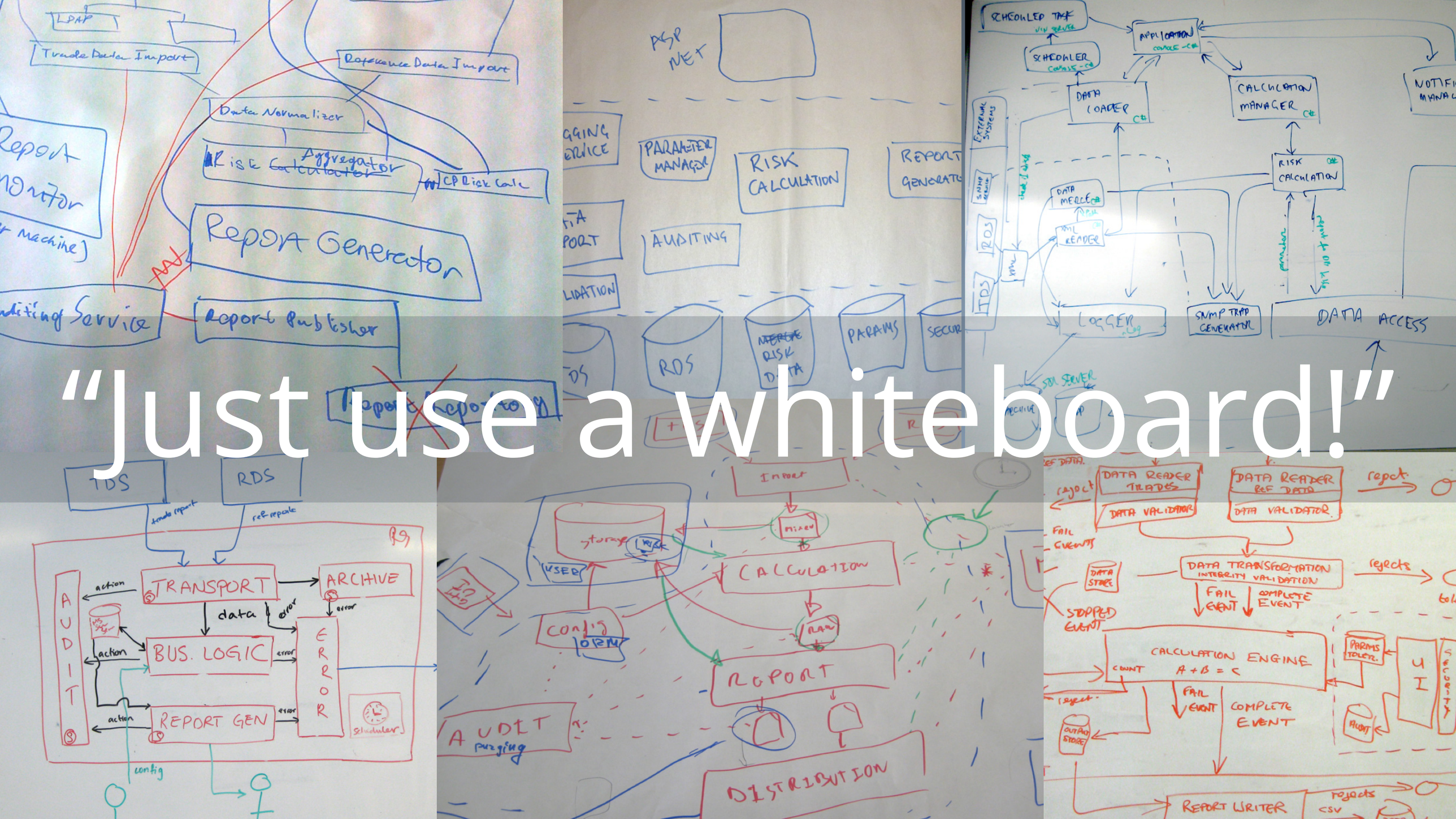
#80 “It’s too detailed.”

#81 “It’s a very elaborate waste of time.”

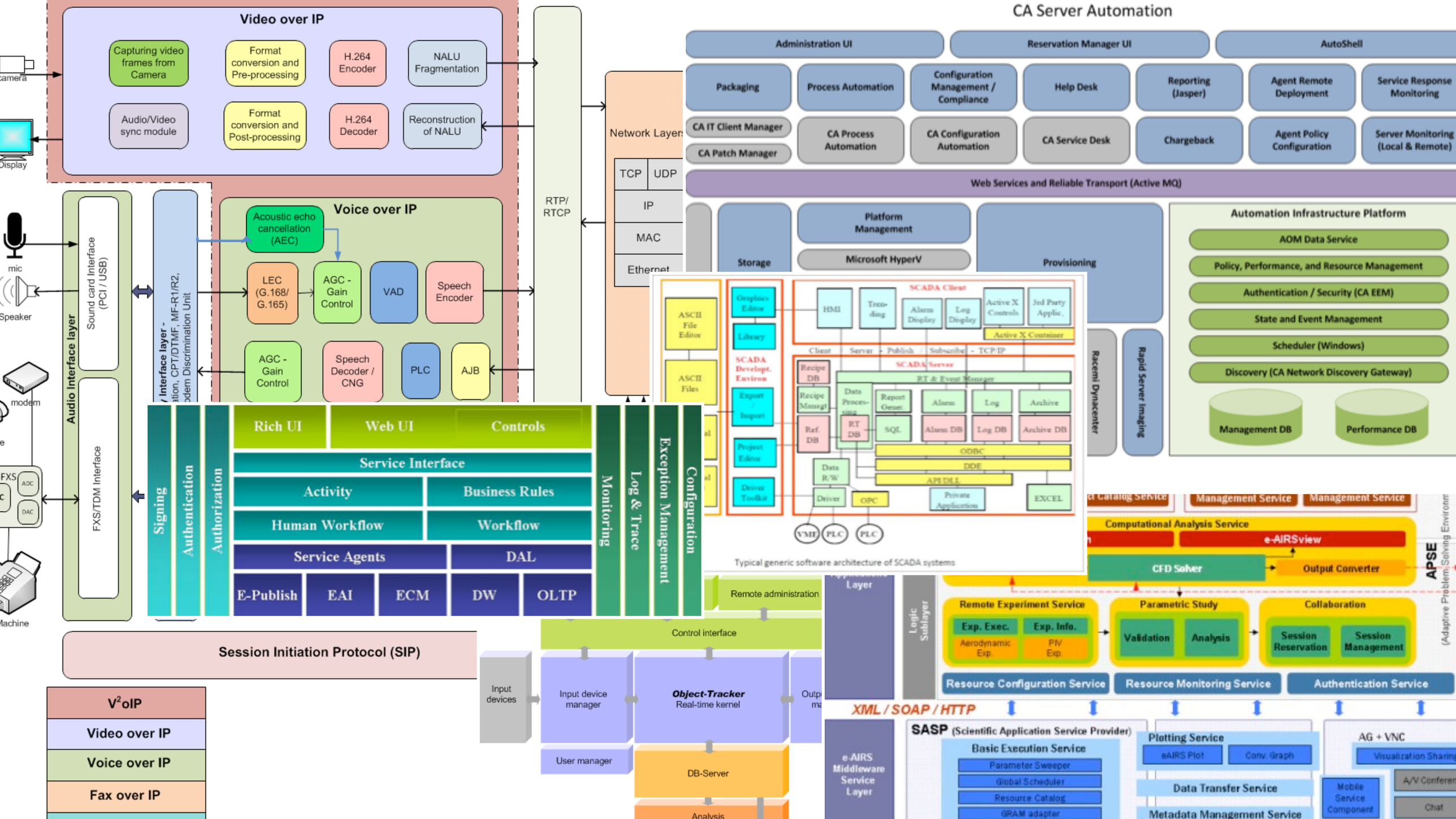
#92 “It’s not expected in agile.”

#97 “The value is in the conversation.”


UML, Cucumber and modeling reality - MPJ's Musings - Fun Fun Function
Just use a whiteboard!
1:42 / 12:48



“Just use a whiteboard!”


Video over IP
Capturing video frames from Camera
Format conversion and Pre-processing
H.264 Encoder
NALU Fragmentation
Audio/Video sync module
Format conversion and Post-processing
H.264 Decoder
Reconstruction of NALU
camera
Display
Voice over IP
Acoustic echo cancellation (AEC)
LEC (G.168/ G.165)
AGC - Gain Control
VAD
Speech Encoder
AGC - Gain Control
Speech Decoder / CNG
PLC
AJB
mic
Speaker
modem
Machine
Audio Interface layer
Sound card Interface (PCI / USB)
FXS/TDM Interface
Interface layer - tion, CPT/DTMF, MF-R1/R2, odem Discrimination Unit
RTP/ RTCP
Network Layers
TCP
UDP
IP
MAC
Ethernet
Signing
Authentication
Authorization
Rich UI
Web UI
Controls
Service Interface
Activity
Business Rules
Human Workflow
Workflow
Service Agents
DAL
E-Publish
EAI
ECM
DW
OLTP
Monitoring
Log & Trace
Exception Management
Configuration
Session Initiation Protocol (SIP)
V²oIP
Video over IP
Voice over IP
Fax over IP
CA Server Automation
Administration UI
Reservation Manager UI
AutoShell
Packaging
Process Automation
Configuration Management / Compliance
Help Desk
Reporting (Jasper)
Agent Remote Deployment
Service Response Monitoring
CA IT Client Manager
CA Patch Manager
CA Process Automation
CA Configuration Automation
CA Service Desk
Chargeback
Agent Policy Configuration
Server Monitoring (Local & Remote)
Web Services and Reliable Transport (Active MQ)
Storage
Platform Management
Microsoft HyperV
Provisioning
Automation Infrastructure Platform
AOM Data Service
Policy, Performance, and Resource Management
Authentication / Security (CA EEM)
State and Event Management
Scheduler (Windows)
Discovery (CA Network Discovery Gateway)
Management DB
Performance DB
Racemi Dynacenter
Rapid Server Imaging
ASCII File Editor
ASCII Files
Graphics Editor
Library
SCADA Develop. Environ
Export
Import
Project Editor
Driver Toolkit
SCADA Client
HMI
Trending
Alarm Display
Log Display
Active X Controls
3rd Party Applic.
Active X Container
Client - Server - Publish / Subscribe - TCP/IP
SCADA Server
Recipe DB
Recipe Managt
Ref. DB
Data Process-ing
RT & Event Manager
Report Gener.
Alarm
Log
Archive
RT DB
SQL
Alarm DB
Log DB
Archive DB
ODBC
DDE
API DLL
Data R/W
Driver
OPC
Private Application
EXCEL
VME
PLC
PLC
Typical generic software architecture of SCADA systems
Management Service
Management Service
Computational Analysis Service
e-AIRSview
CFD Solver
Output Converter
APSE
(Adaptive Problem Solving Environment)
Remote Experiment Service
Exp. Exec.
Exp. Info.
Aerodynamic Exp.
PIV Exp.
Parametric Study
Validation
Analysis
Collaboration
Session Reservation
Session Management
Resource Configuration Service
Resource Monitoring Service
Authentication Service
XML / SOAP / HTTP
Logic Sublayer
e-AIRS Middleware Service Layer
SASP (Scientific Application Service Provider)
Basic Execution Service
Parameter Sweeper
Global Scheduler
Resource Catalog
GRAM adapter
Plotting Service
eAIRS Plot
Conv. Graph
Data Transfer Service
Metadata Management Service
AG + VNC
Visualization Sharing
Mobile Service Component
A/V Conferen
Chat
Remote administration
Control interface
Input devices
Input device manager
Object-Tracker
Real-time kernel
User manager
DB-Server
Analysis

If you’re going to use “boxes & lines”, at least do so in a **structured way**, using a **self-describing notation**

Moving fast in the same direction
as a team requires
good communication

To describe a software architecture, we use a model composed of multiple views or perspectives.

Architectural Blueprints - The “4+1” View Model of Software Architecture

Philippe Kruchten

The description of an architecture—the decisions made—can be organized around these four views, and then illustrated by a few selected *use cases*, or *scenarios* which become a fifth view. The architecture is in fact partially evolved from these scenarios as we will see later.

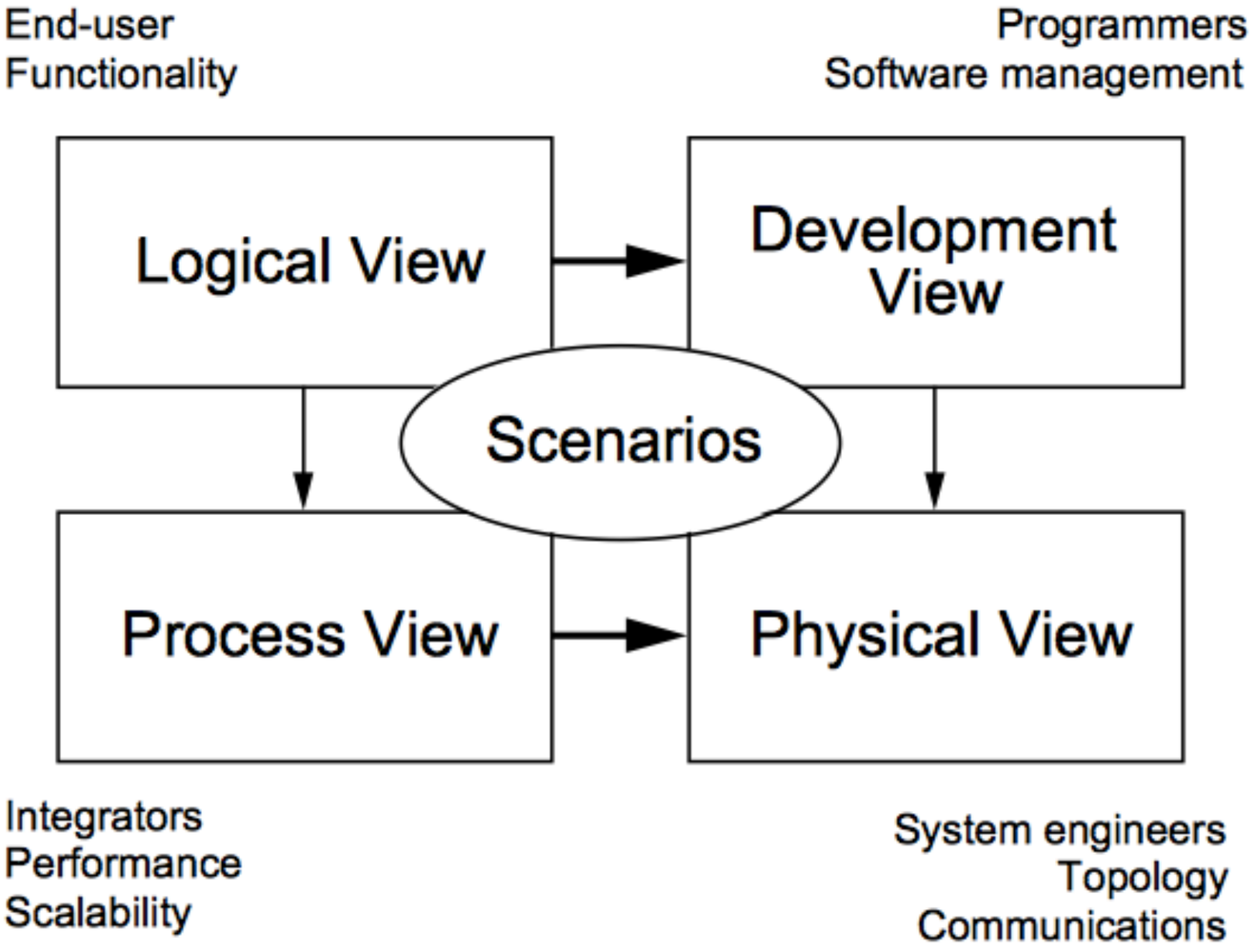
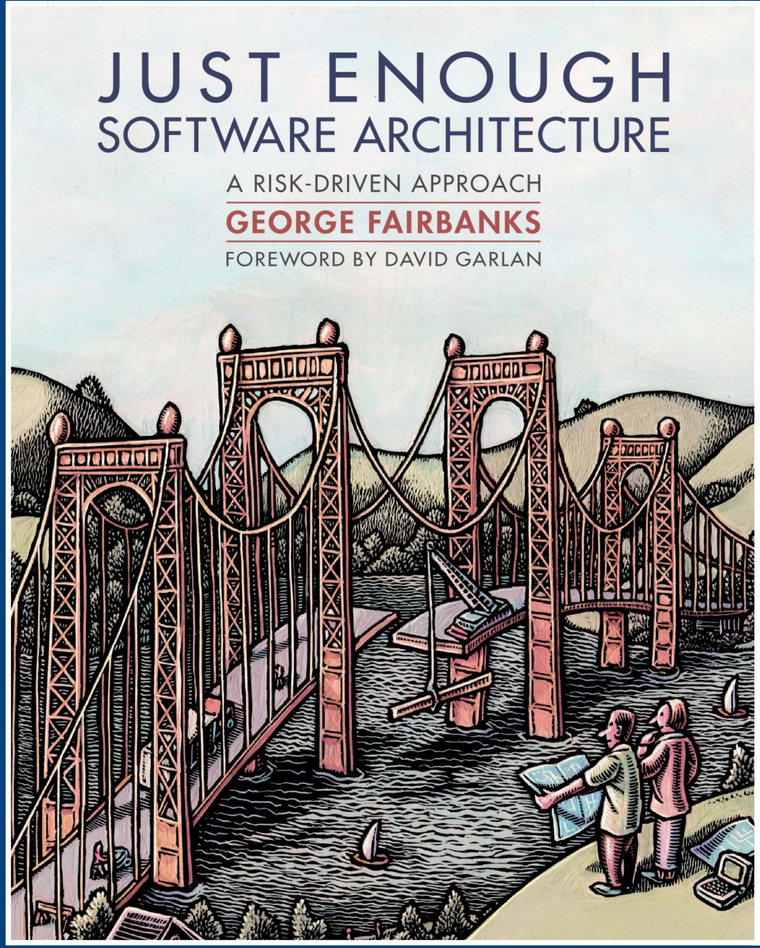


Figure 1 — The "4+1" view model

Why is there a separation between the **logical** and **development** views?

Our architecture diagrams don’t match the code.



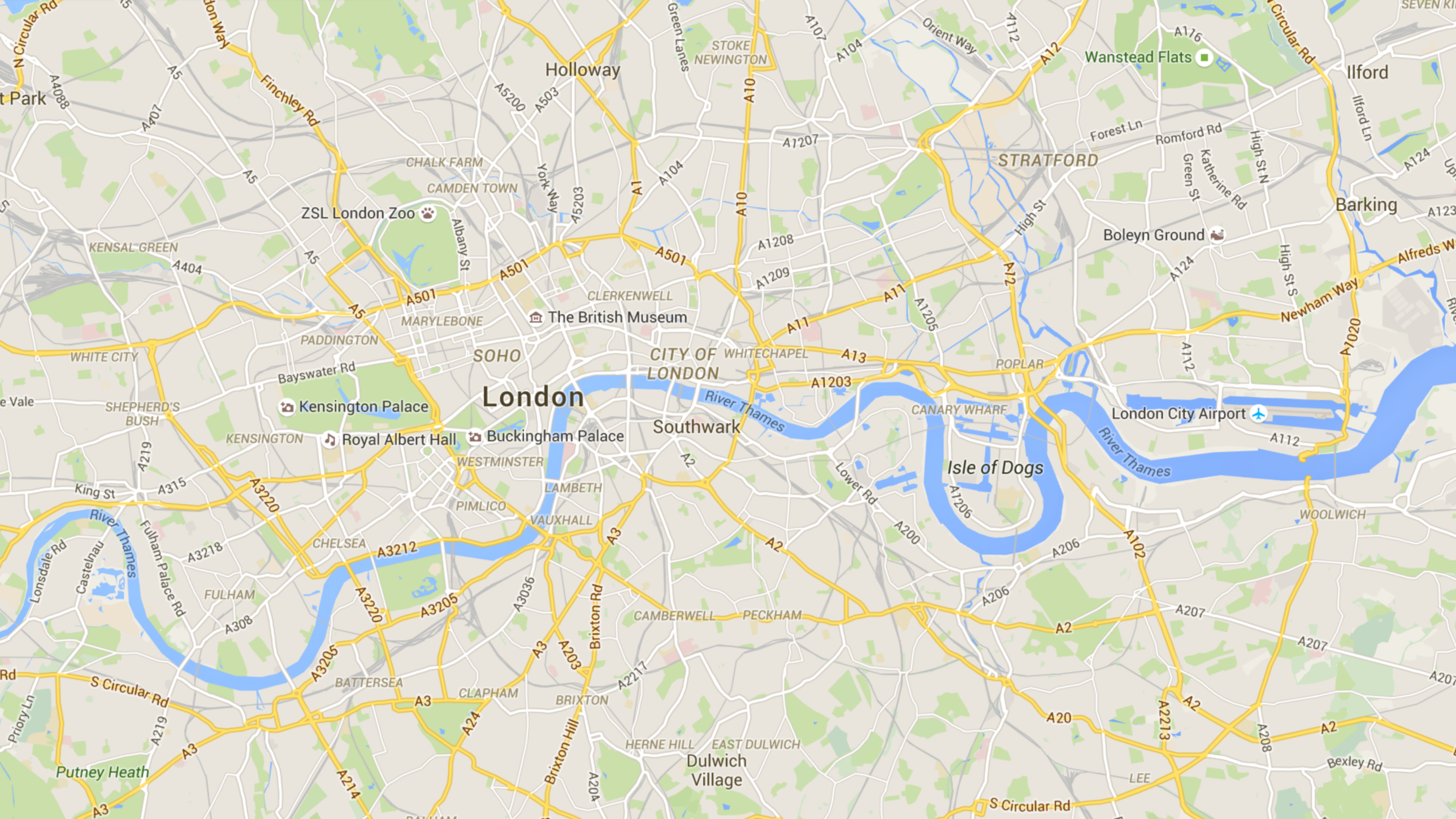
Model-code gap. Your architecture models and your source code will not show the same things. The difference between them is the *model-code gap*. Your architecture models include some abstract concepts, like components, that your programming language does not, but could. Beyond that, architecture models include intensional elements, like design decisions and constraints, that cannot be expressed in procedural source code at all.

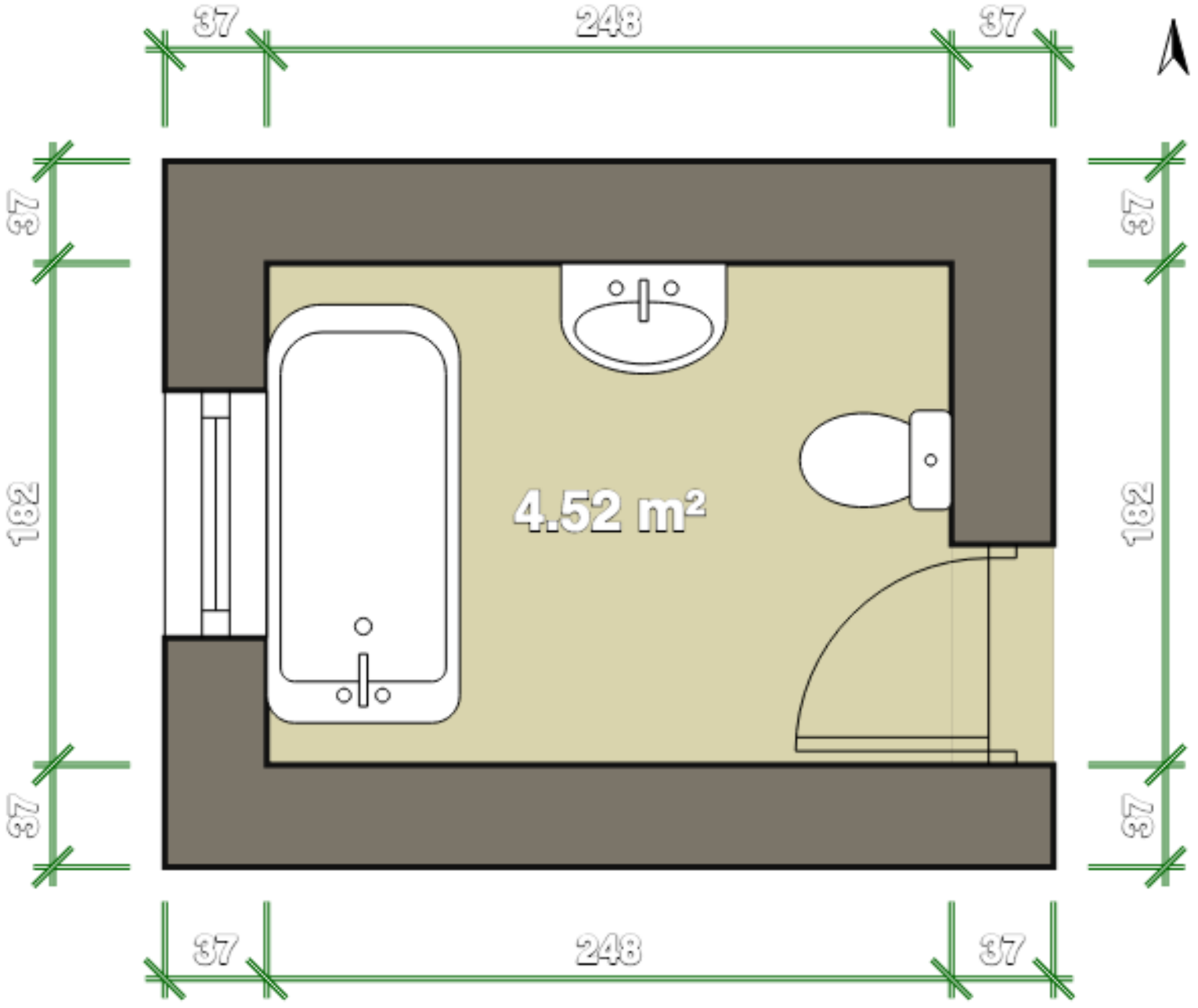
Consequently, the relationship between the architecture model and source code is complicated. It is mostly a refinement relationship, where the extensional elements in the architecture model are refined into extensional elements in source code. This is shown in Figure 10.3. However, intensional elements are not refined into corresponding elements in source code.

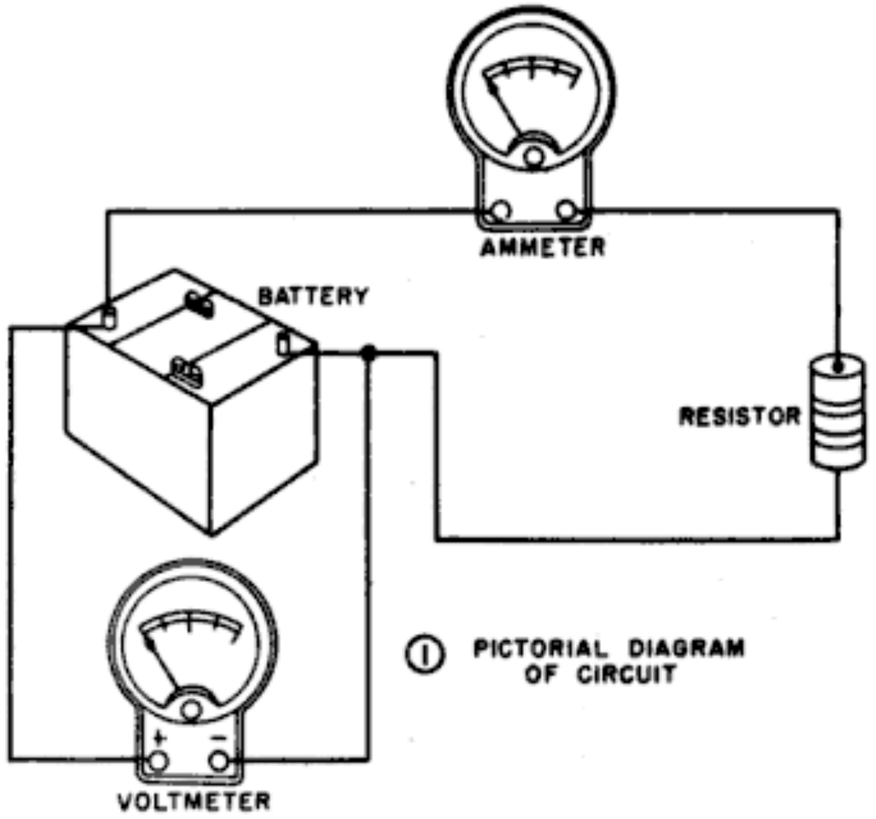
Upon learning about the model-code gap, your first instinct may be to avoid it. But reflecting on the origins of the gap gives little hope of a general solution in the short term: architecture models help you reason about complexity and scale because they are abstract and intensional; source code executes on machines because it is concrete and extensional.

“model-code gap”

We lack a **common vocabulary**
to describe software architecture


London
Holloway
STOKE NEWINGTON
Wanstead Flats
Ilford
Barking
STRATFORD
CHALK FARM
CAMDEN TOWN
ZSL London Zoo
Boleyn Ground
KENSAL GREEN
MARYLEBONE
The British Museum
CLERKENWELL
PADDINGTON
WHITE CITY
SOHO
CITY OF LONDON
WHITECHAPEL
POPLAR
Bayswater Rd
Kensington Palace
SHEPHERD'S BUSH
KENSINGTON
Royal Albert Hall
Buckingham Palace
WESTMINSTER
Southwark
River Thames
CANARY WHARF
London City Airport
Isle of Dogs
LAMBETH
PIMLICO
VAUXHALL
CHELSEA
FULHAM
CAMBERWELL
PECKHAM
BATTERSEA
CLAPHAM
BRIXTON
HERNE HILL
EAST DULWICH
Dulwich Village
Putney Heath
WOOLWICH
LEE
Bexley Rd
Finchley Rd
Albany St
York Way
Green Lanes
Orient Way
Forest Ln
Romford Rd
Green St
Katherine Rd
High St
High St N
High St S
Ilford Ln
Newham Way
Alfreds W
Lower Rd
Brixton Rd
Brixton Hill
Fulham Palace Rd
Lonsdale Rd
Castelnau
Priory Ln
King St
S Circular Rd
N Circular Rd
A1
A2
A3
A5
A10
A11
A12
A13
A20
A24
A40
A102
A104
A107
A112
A116
A123
A124
A200
A201
A202
A203
A204
A206
A207
A208
A214
A219
A308
A315
A404
A407
A501
A503
A1020
A1203
A1205
A1206
A1207
A1208
A1209
A2213
A2217
A3036
A3205
A3212
A3218
A3220
A4088
A5200
A5203


37
248
37
37
182
37
37
182
37
4.52 m²
37
248
37



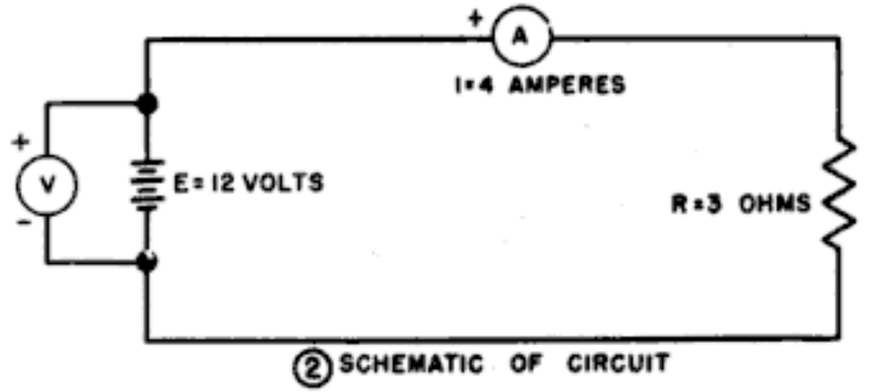
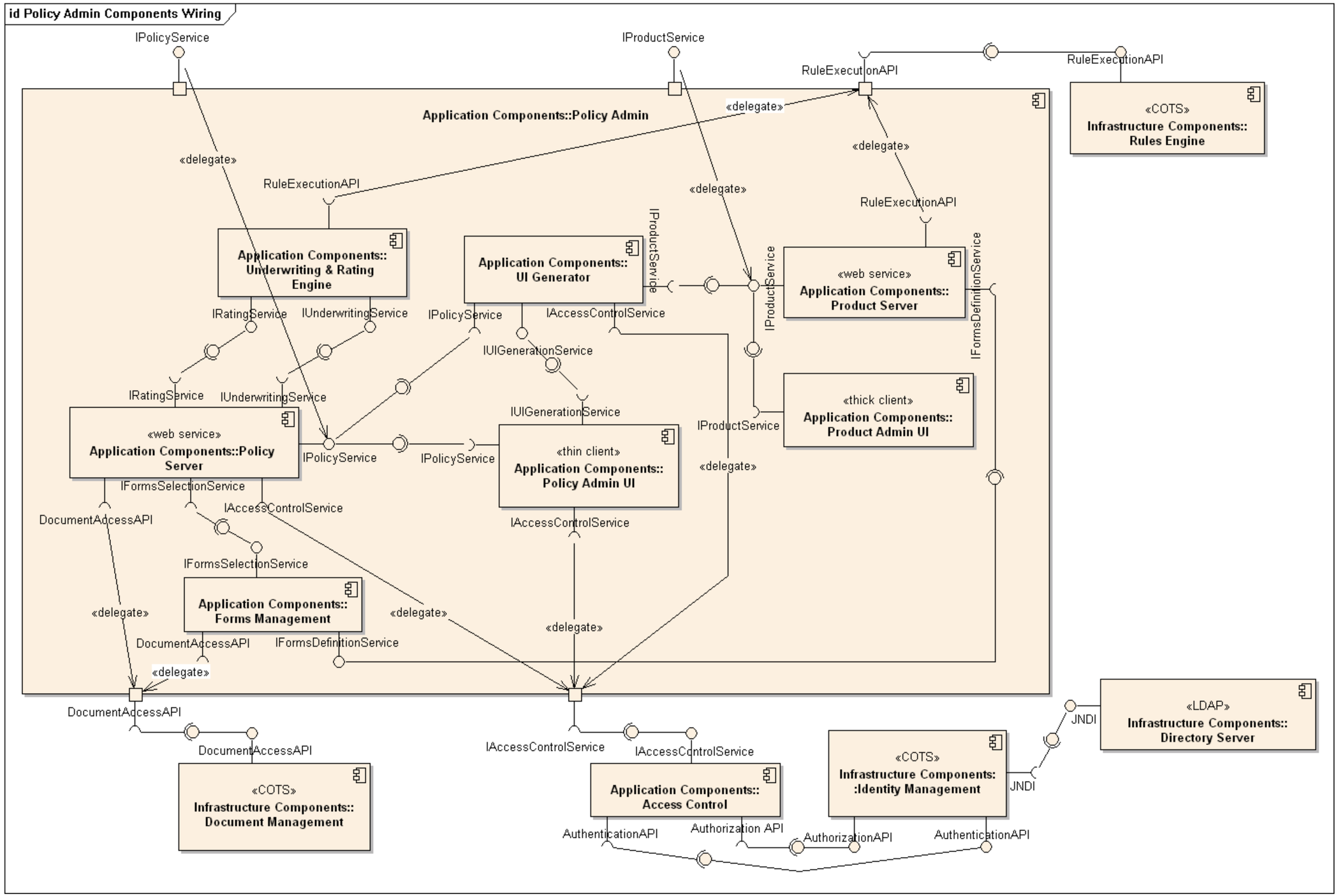
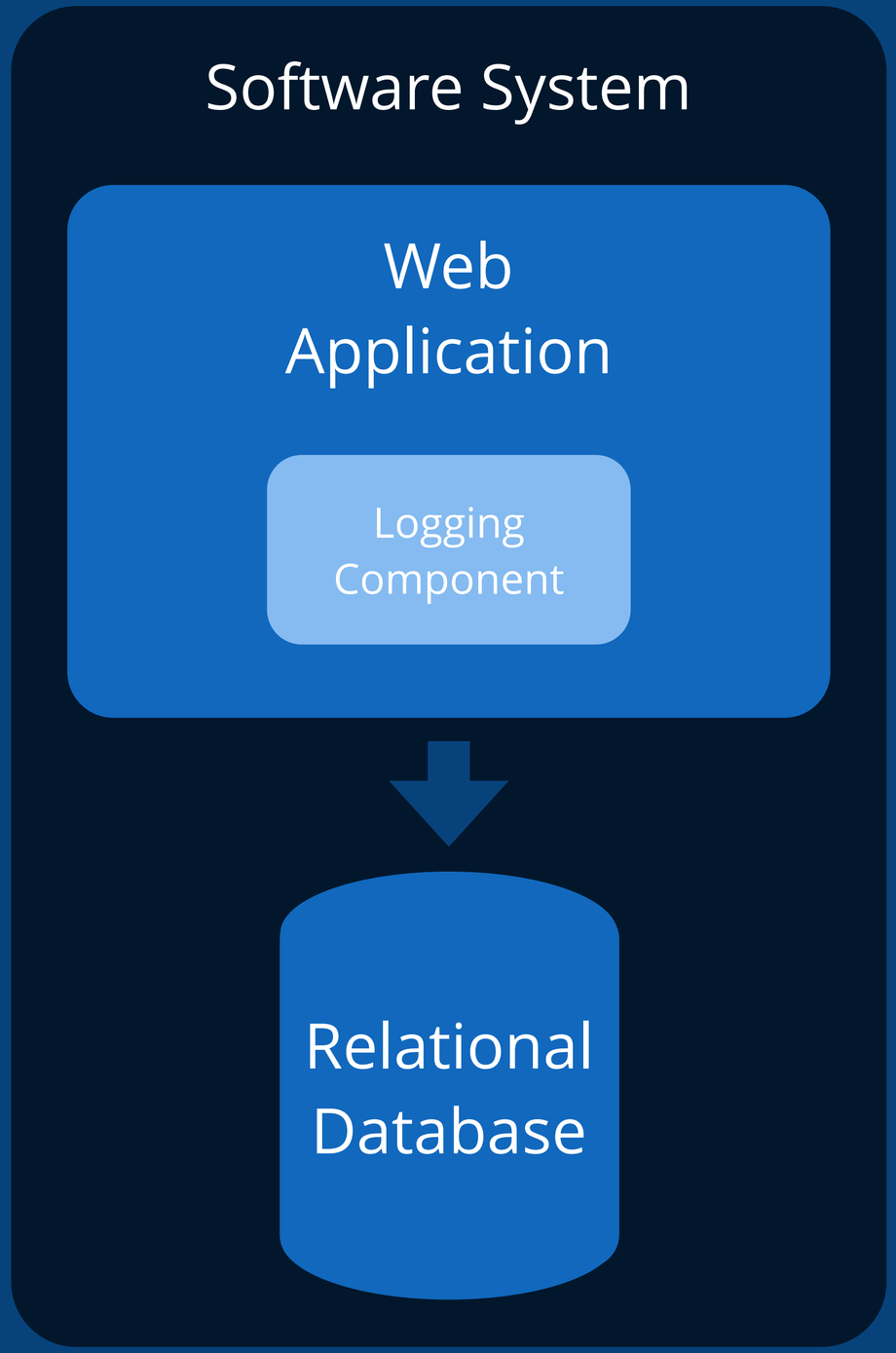


Figure 48. Diagram of a basic circuit.

https://en.wikipedia.org/wiki/Circuit_diagram



https://en.wikipedia.org/wiki/Component_diagram



[1] component

noun | com·po·nent | \kəm-ˈpō-nənt, ˈkäm-ˌ, käm-ˈ\

Simple Definition of COMPONENT

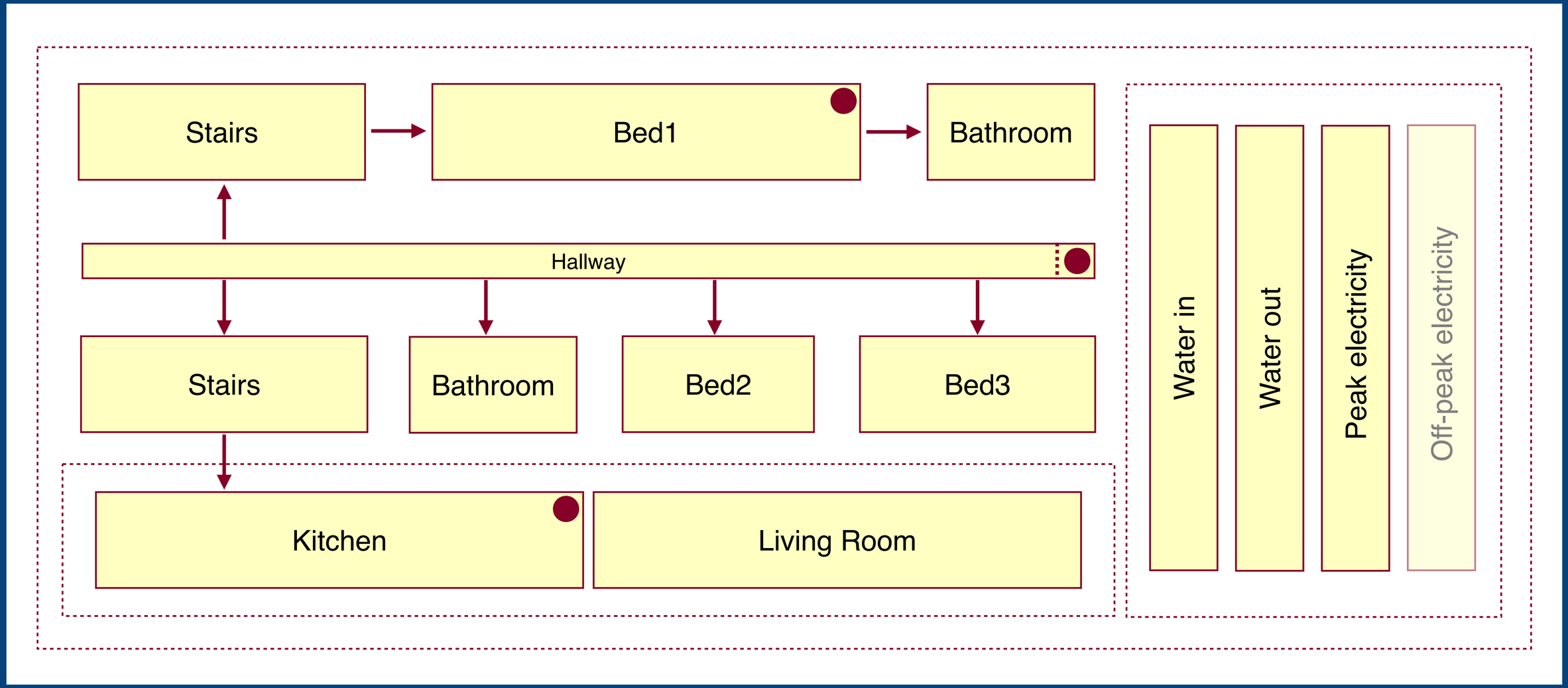
Popularity: Top 30% of words

: one of the parts of something (such as a system or mixture) : an important piece of something

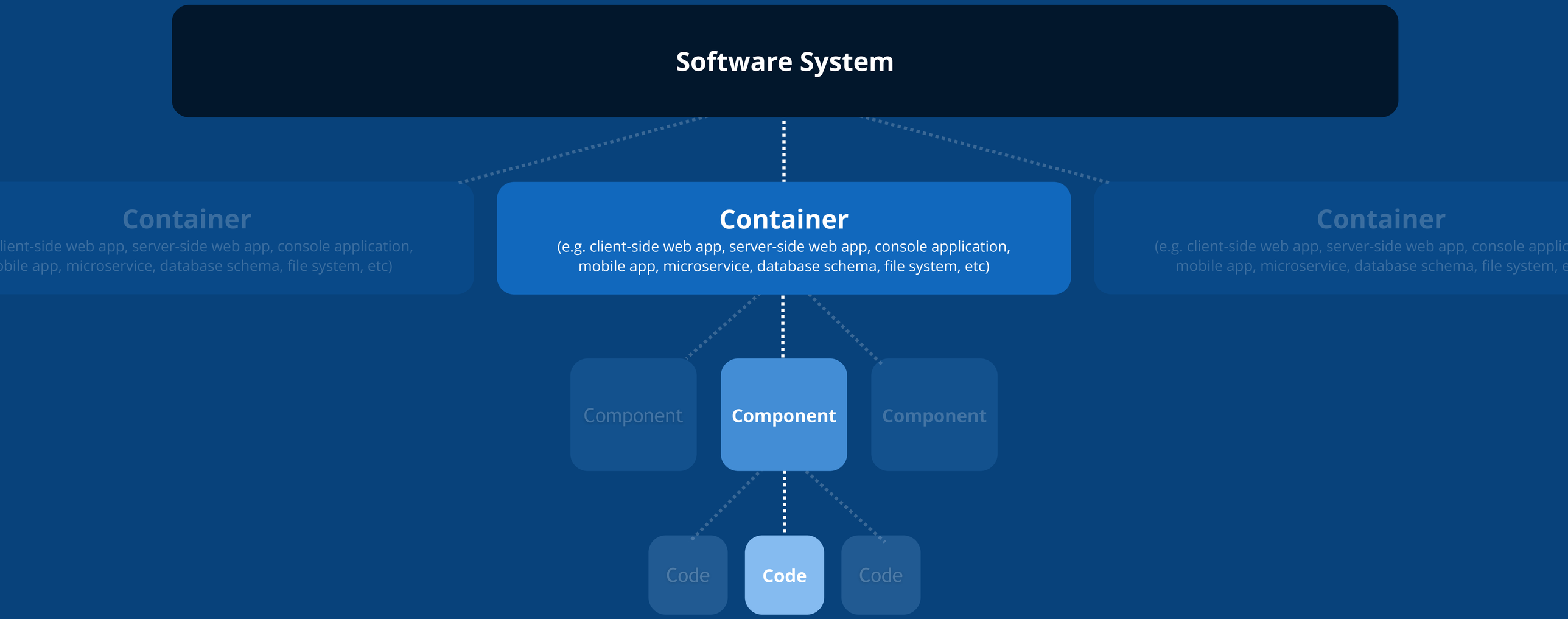
Source: Merriam-Webster's Learner's Dictionary

When drawing software architecture diagrams, think like a software developer

If software developers created building architecture diagrams…



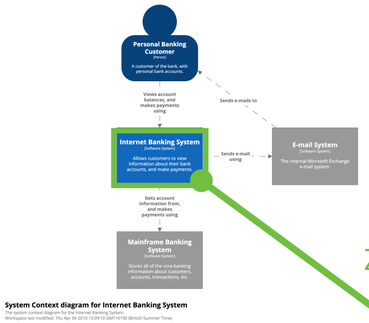
A **common set of abstractions**
is more important
than a common notation



A **software system** is made up of one or more **containers**,
each of which contains one or more **components**,
which in turn are implemented by one or more **code elements**.

The C4 model for visualising software architecture

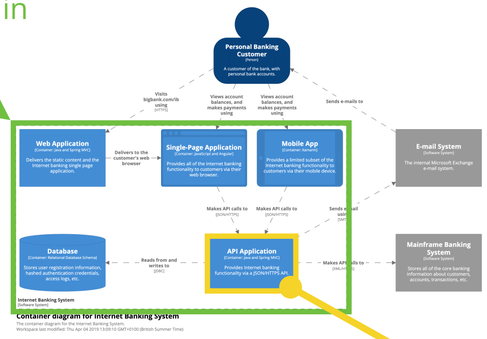
c4model.com



System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

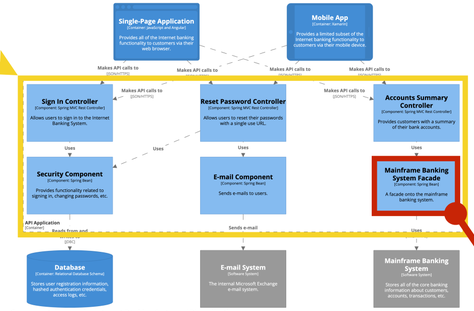
Zoom in



Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

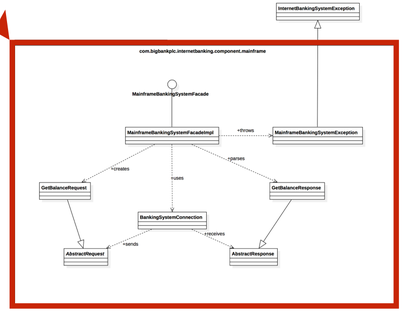
Zoom in



Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

Zoom in



Level 1
System Context

Level 2
Containers

Level 3
Components

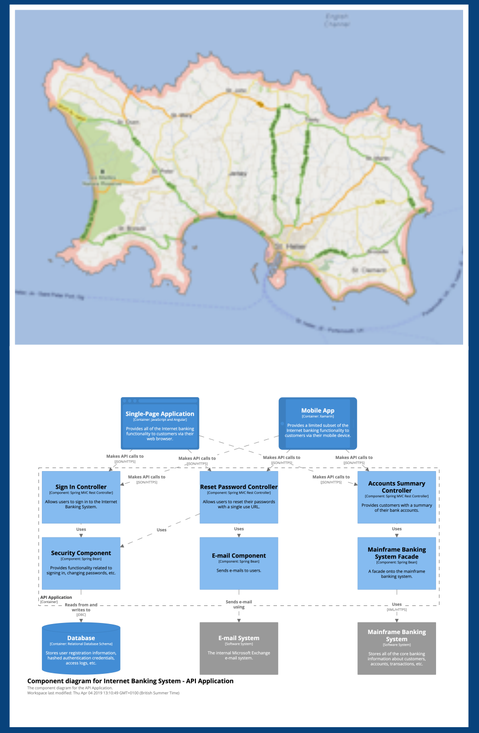
Level 4
Code

System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Workspace last modified: Thu, Apr 04 2019 13:15:48 GMT+0100 (British Summer Time)

Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu, Apr 04 2019 13:15:48 GMT+0100 (British Summer Time)

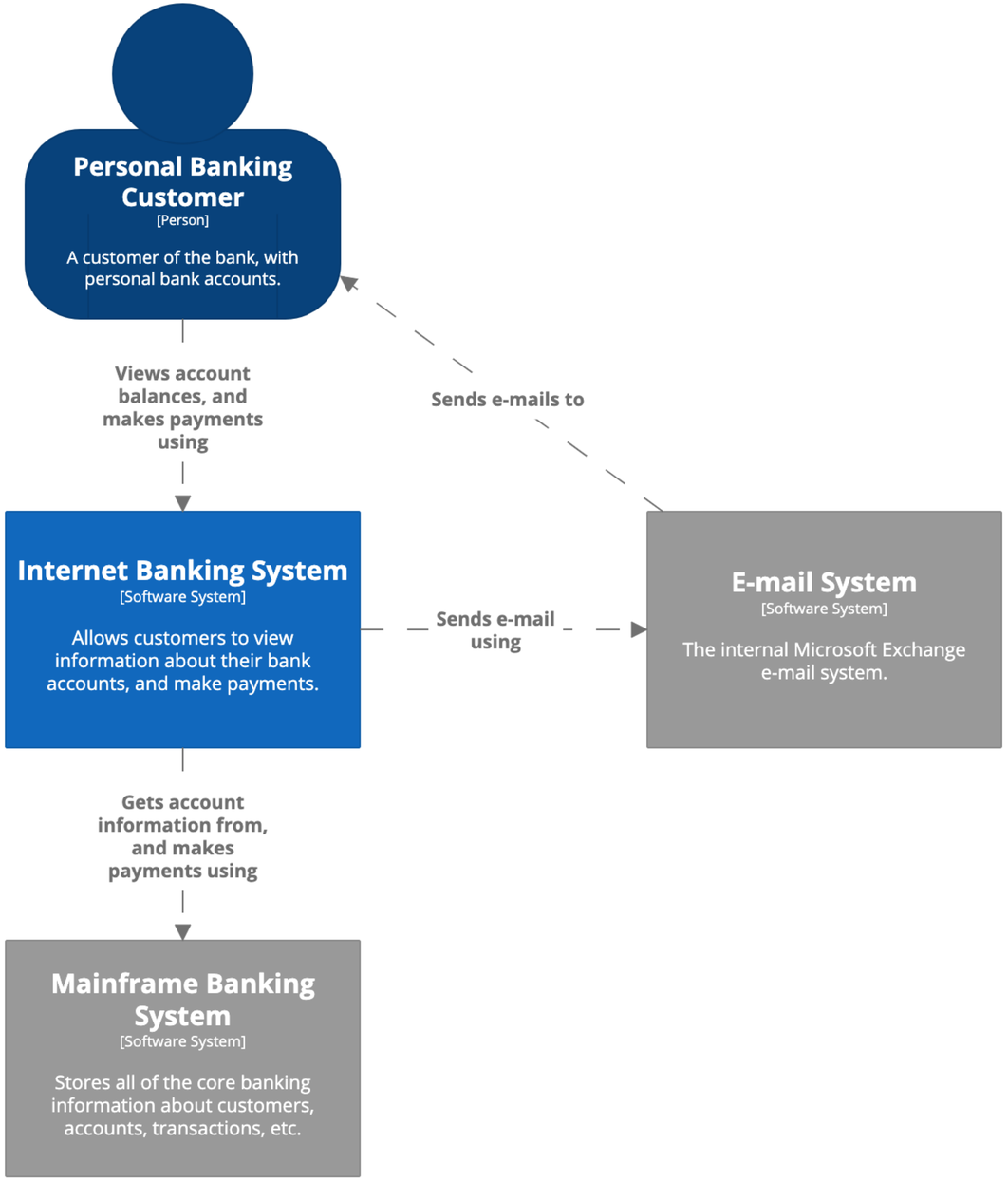


Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu, Apr 04 2019 13:15:48 GMT+0100 (British Summer Time)

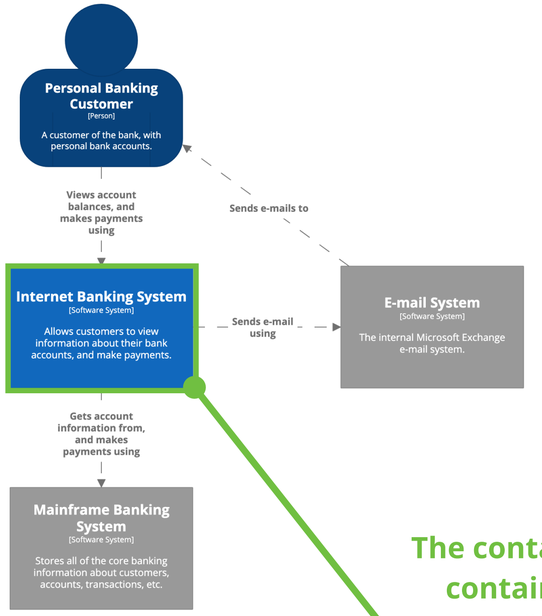
Diagrams are maps

that help software developers navigate a large and/or complex codebase



System Context diagram for Internet Banking System

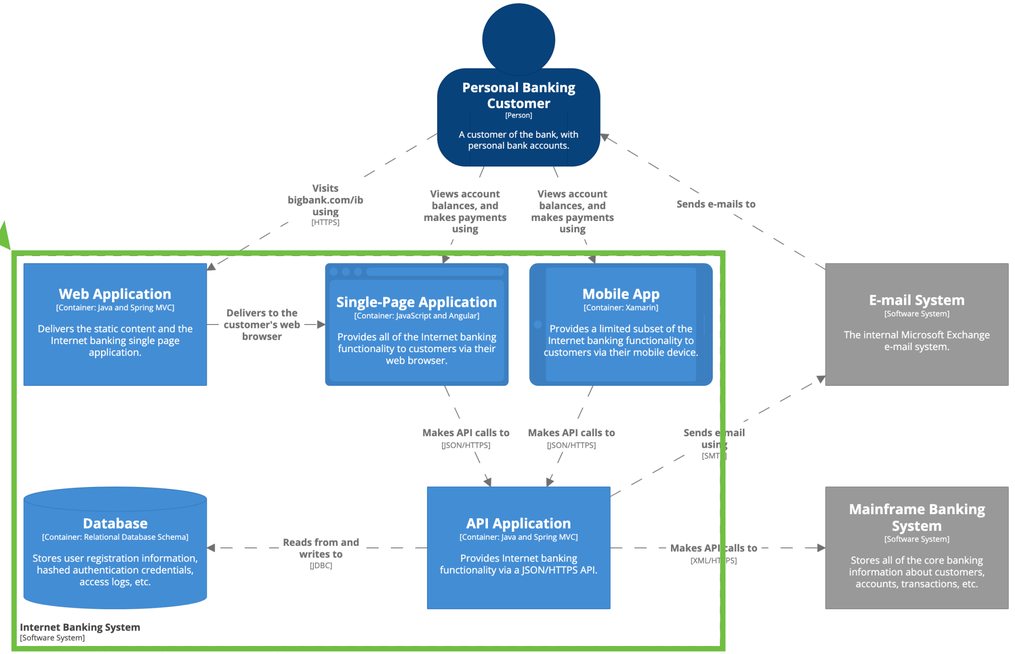
The system context diagram for the Internet Banking System.
Workspace last modified: Wed Feb 05 2020 09:33:36 GMT+0100 (Central European Standard Time)



System Context diagram for Internet Banking System

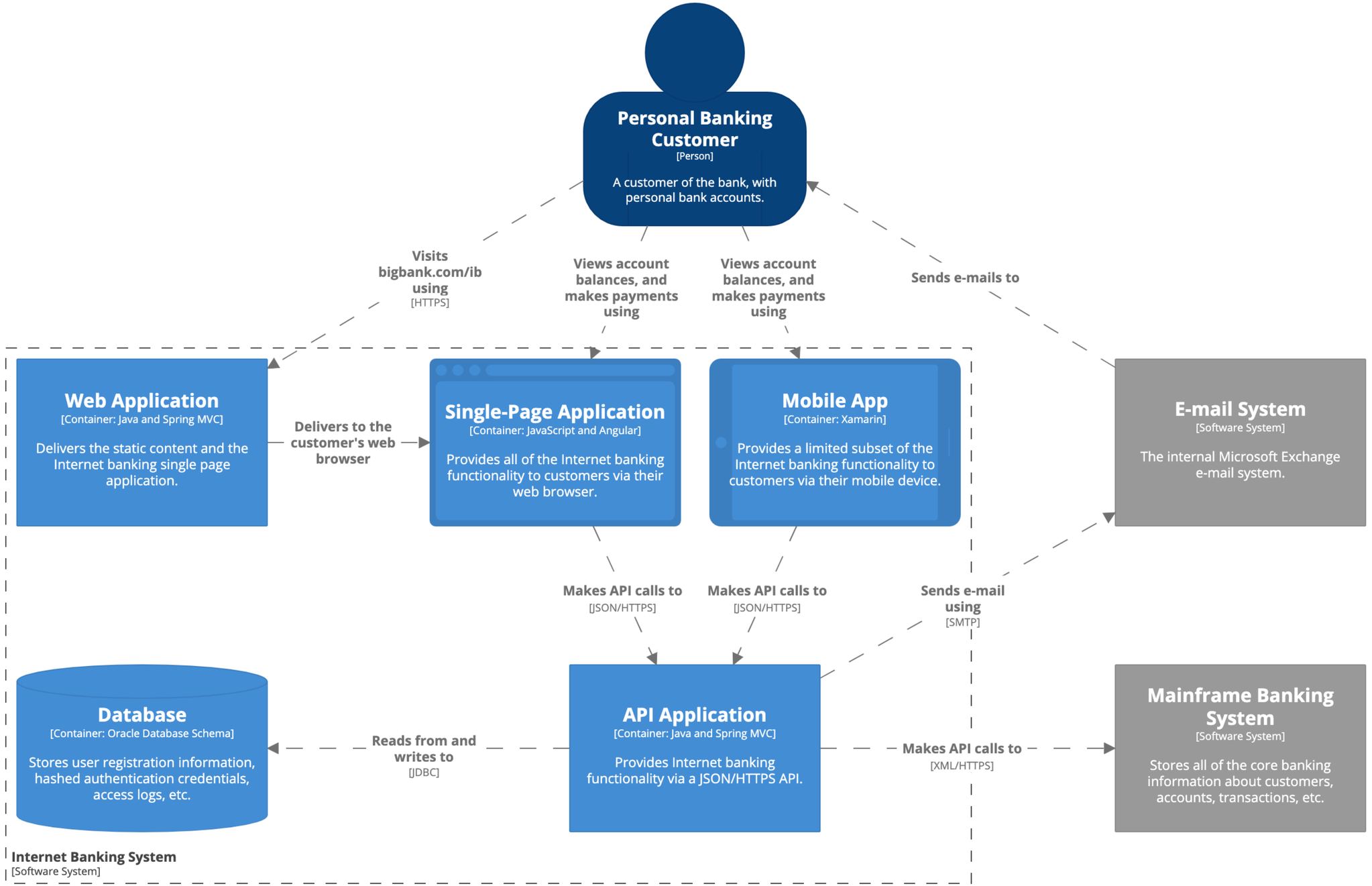
The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

The container diagram shows the containers that reside inside the software system boundary



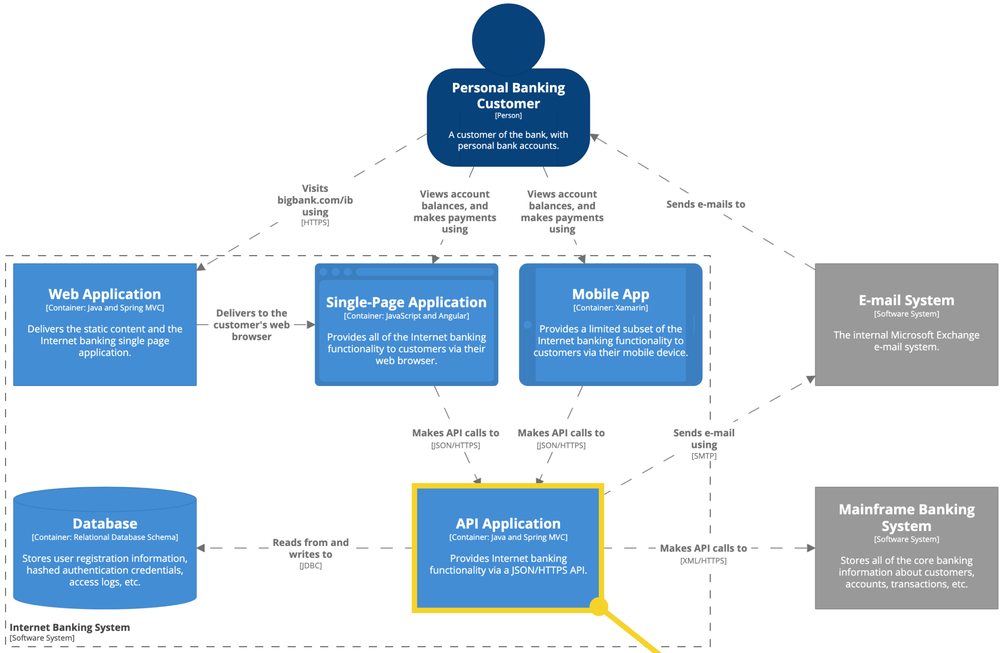
Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



Container diagram for Internet Banking System

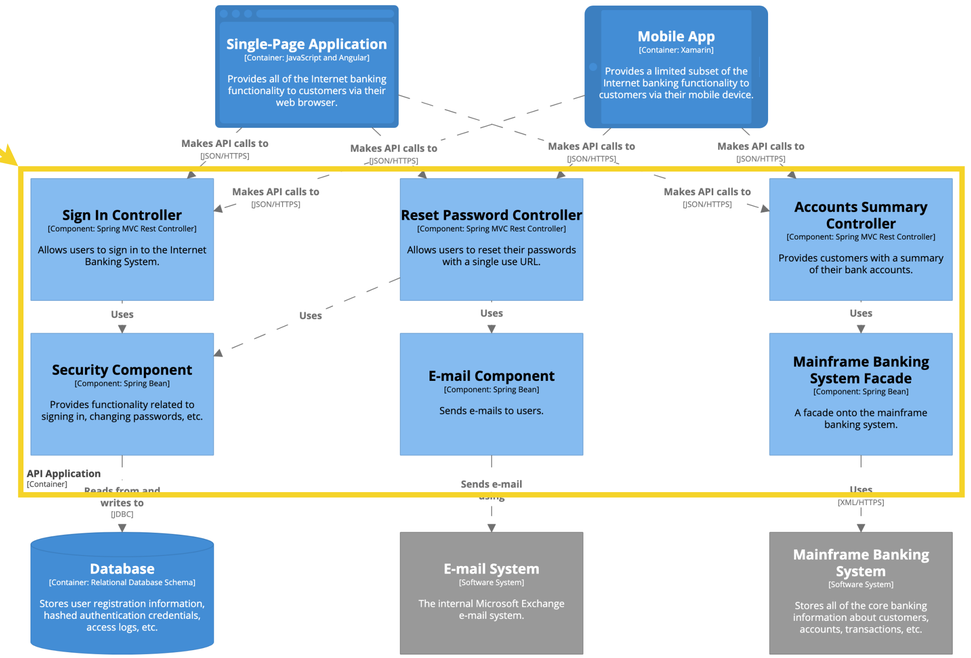
The container diagram for the Internet Banking System.
Workspace last modified: Wed Feb 05 2020 09:33:36 GMT+0100 (Central European Standard Time)



Container diagram for Internet Banking System

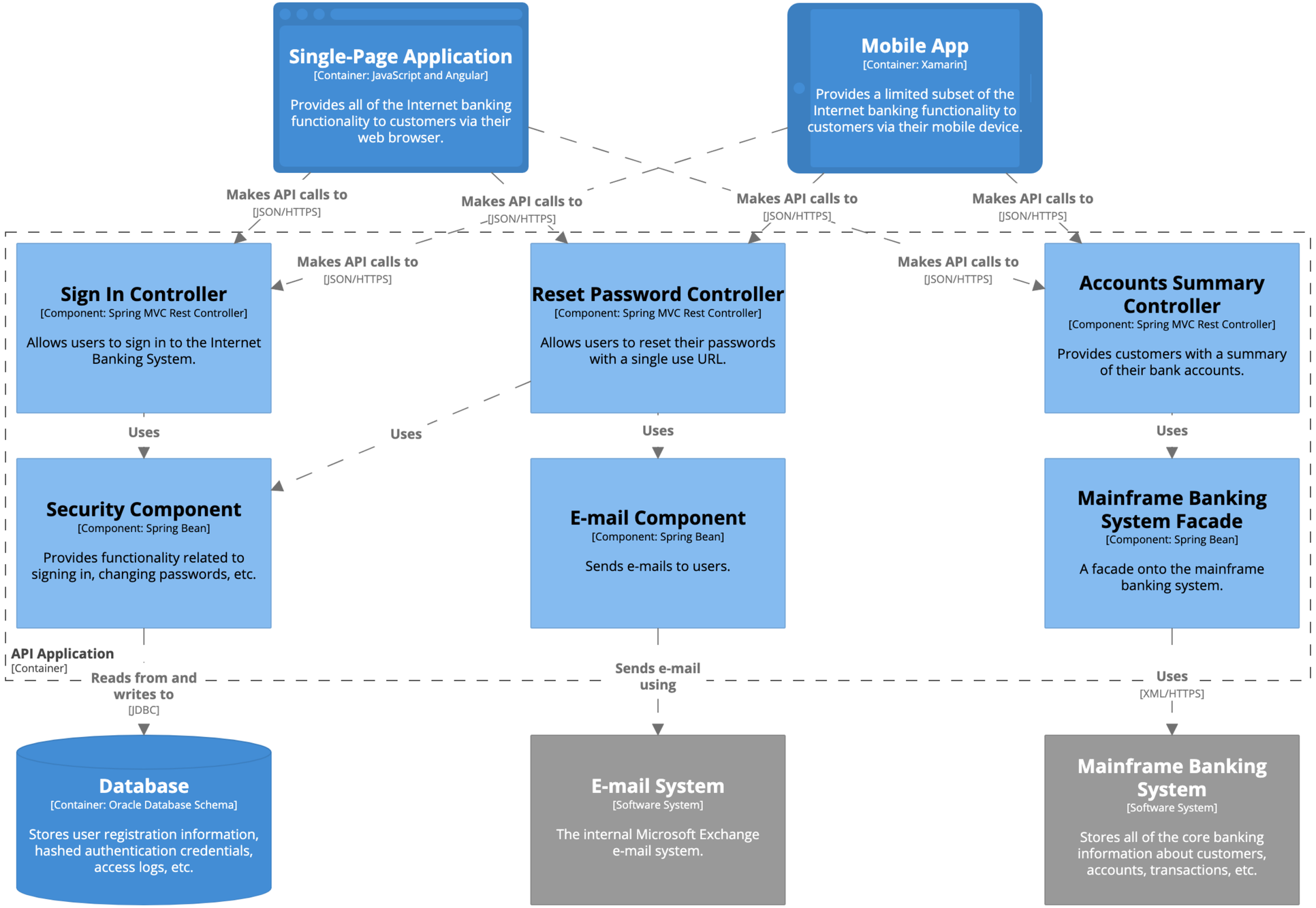
The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

The component diagram shows the components that reside inside an individual container



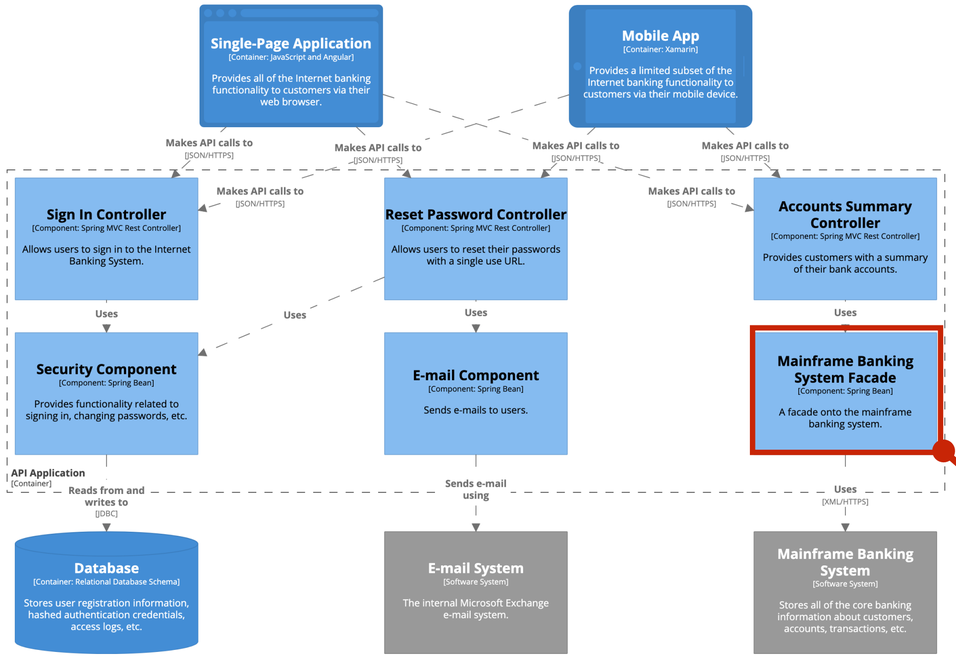
Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



Component diagram for Internet Banking System - API Application

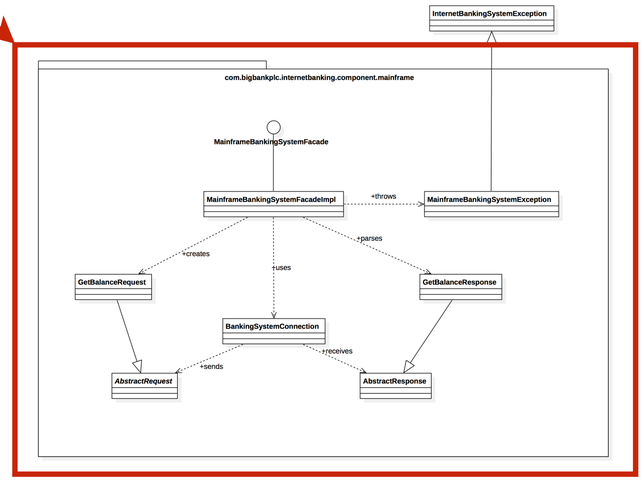
The component diagram for the API Application.
Workspace last modified: Wed Feb 05 2020 09:33:36 GMT+0100 (Central European Standard Time)

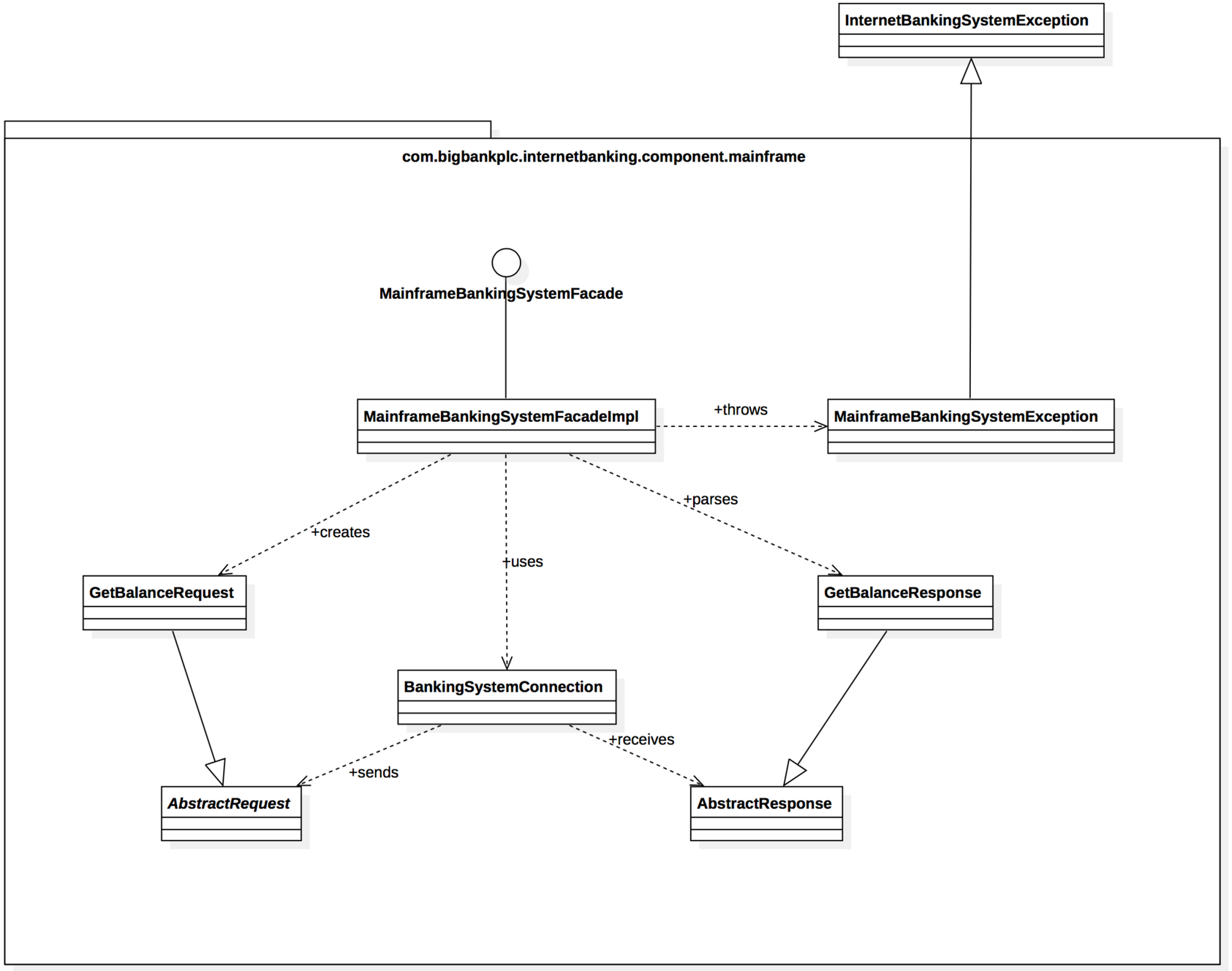


Component diagram for Internet Banking System - API Application

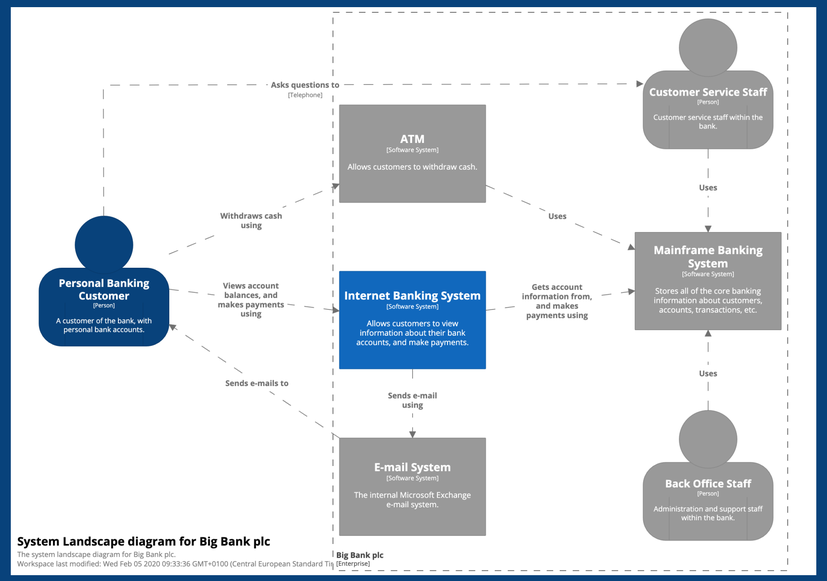
The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

The code level diagram shows the code elements that make up a component

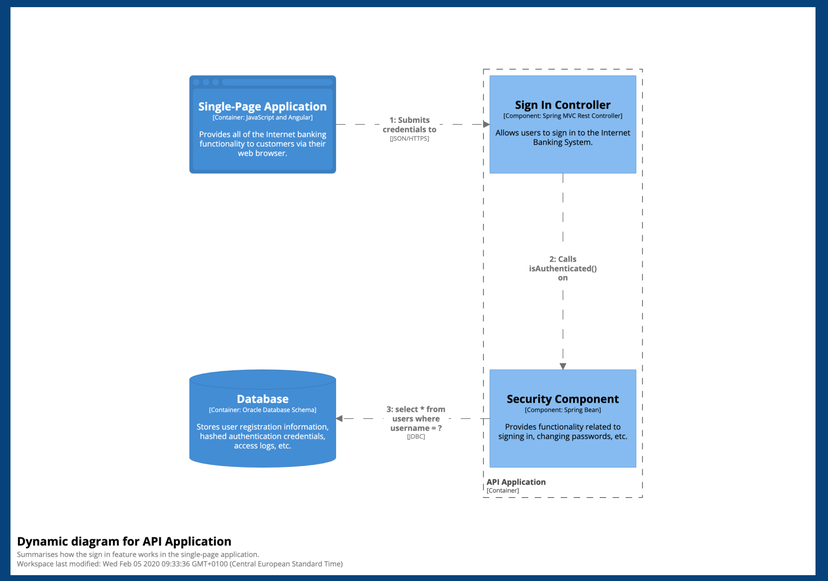



InternetBankingSystemException
com.bigbankplc.internetbanking.component.mainframe
MainframeBankingSystemFacade
MainframeBankingSystemFacadeImpl
+throws
MainframeBankingSystemException
+creates
+uses
+parses
GetBalanceRequest
GetBalanceResponse
BankingSystemConnection
+receives
+sends
AbstractRequest
AbstractResponse

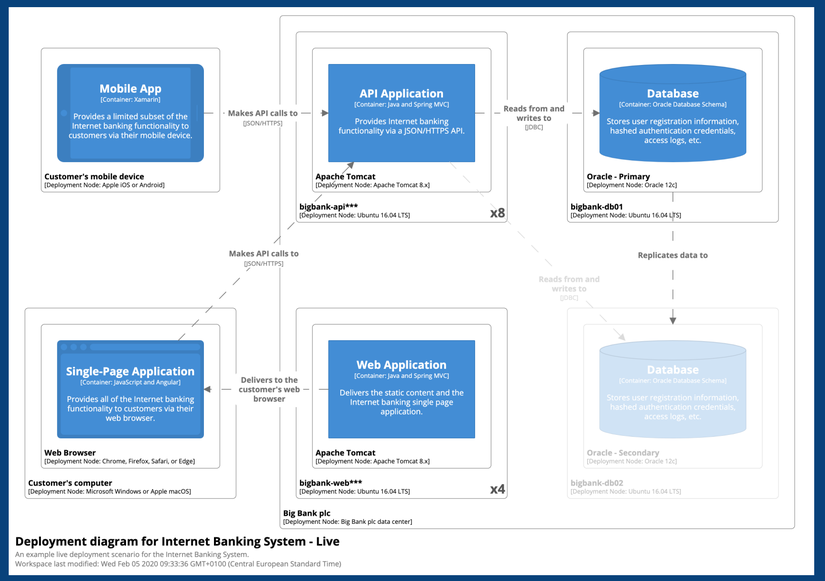
Plus some supplementary diagrams...



System Landscape

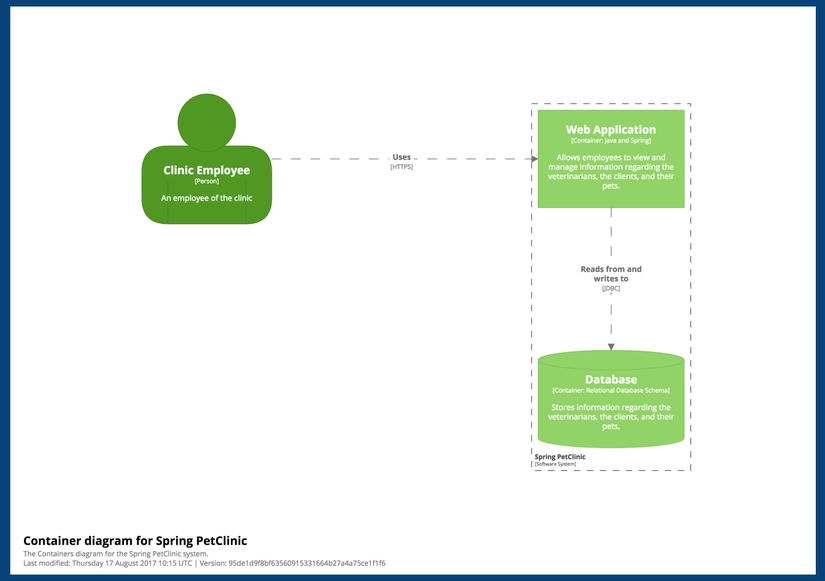


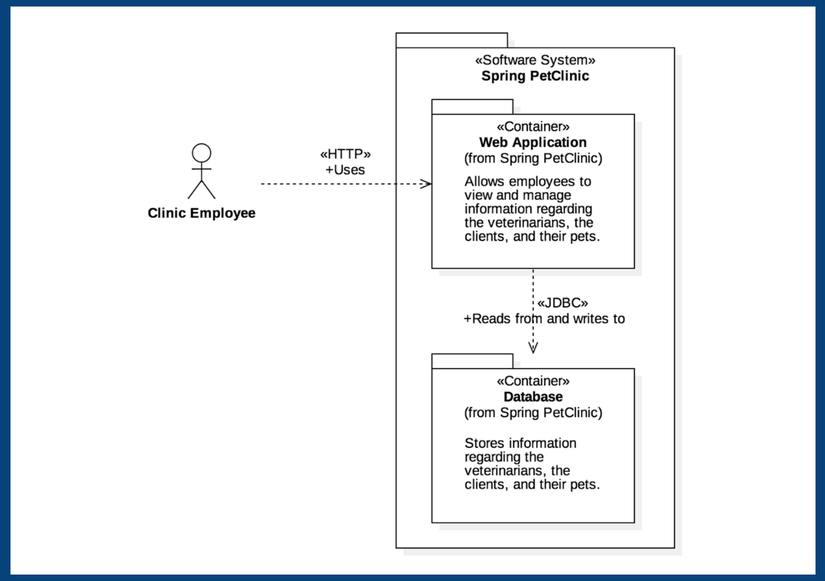
Dynamic

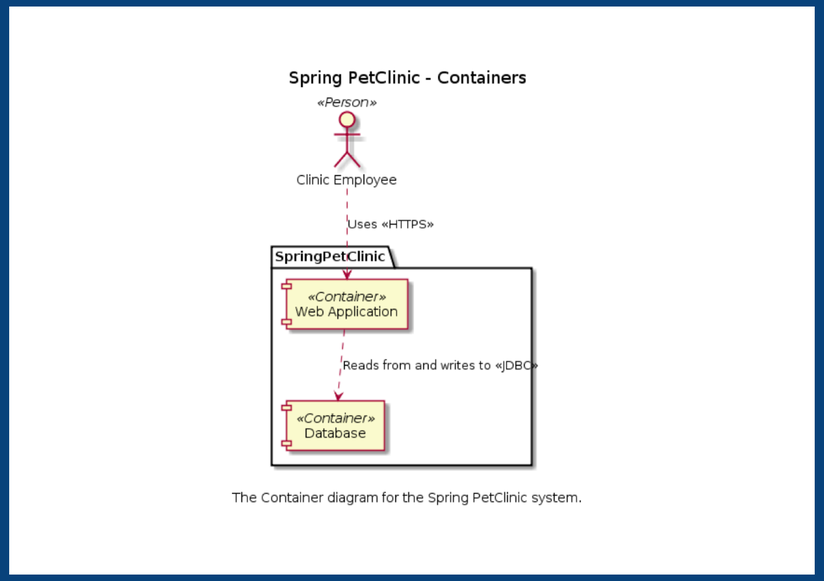


Deployment

The C4 model is **notation independent**







Personal Banking Customer

[Person]

A customer of the bank, with personal bank accounts.

Internet Banking System

[Software System]

Allows customers to view information about their bank accounts, and make payments.

API Application

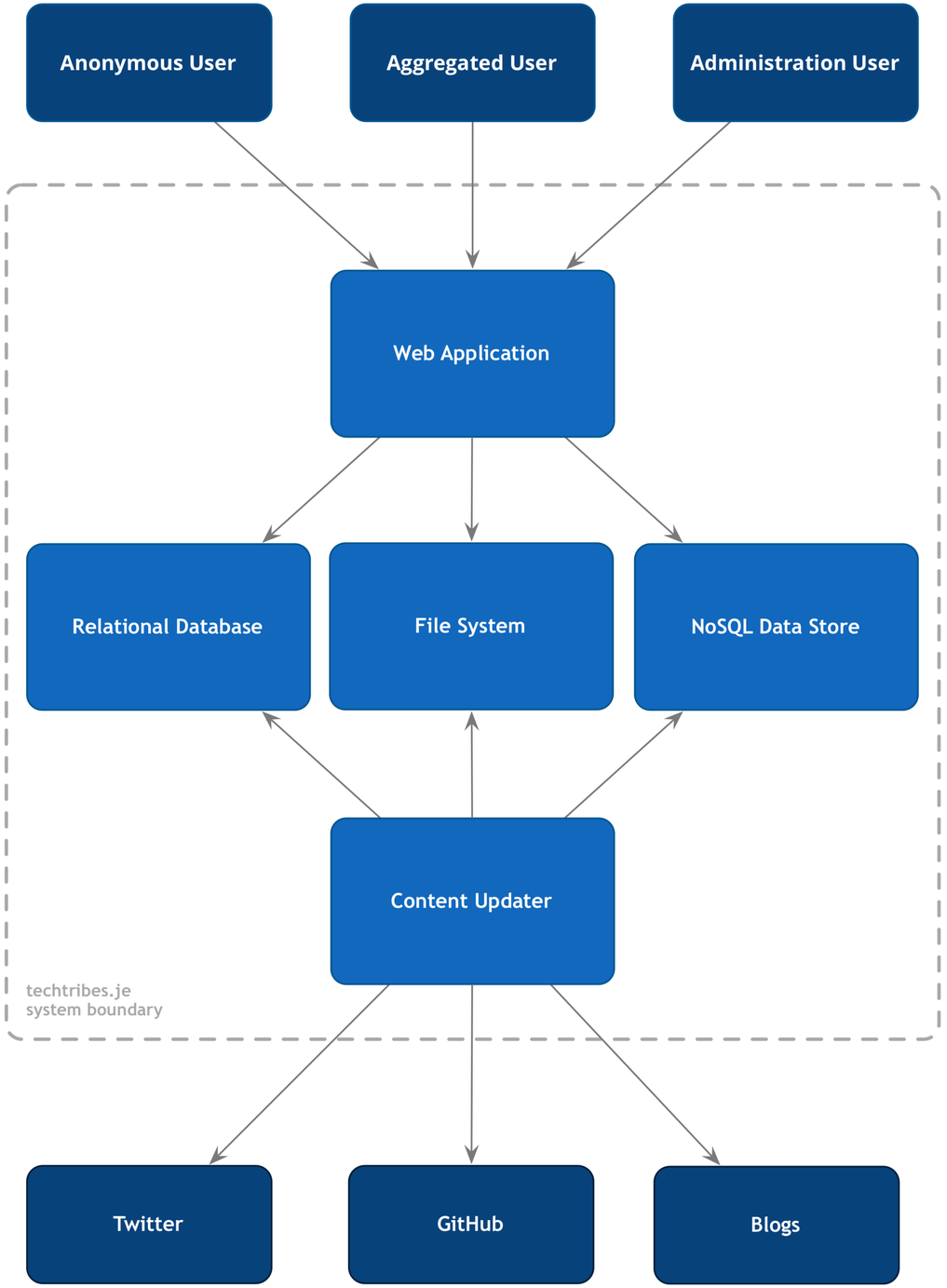
[Container: Java and Spring MVC]

Provides Internet banking functionality via a JSON/HTTPS API.

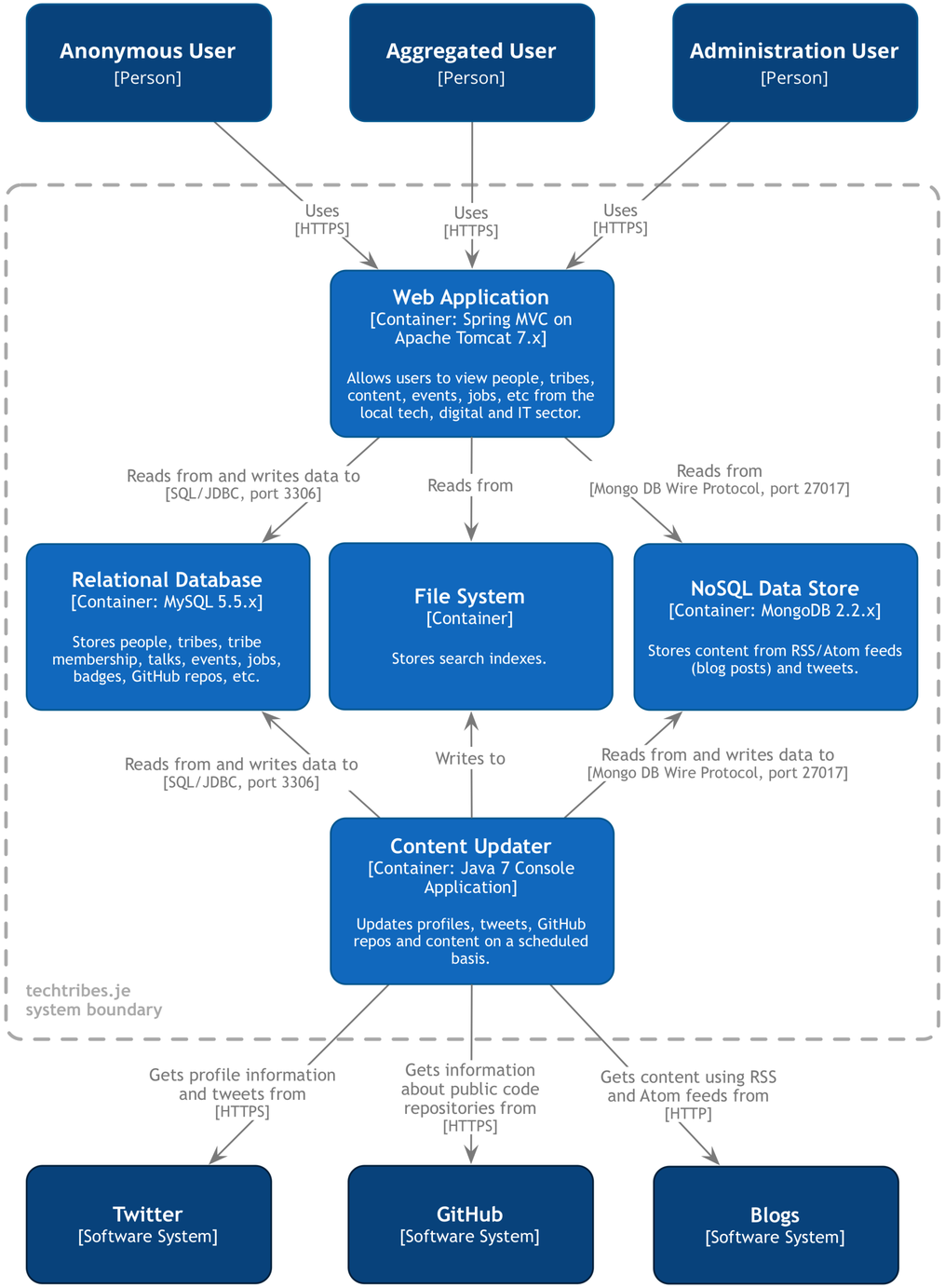
Mainframe Banking System Facade

[Component: Spring Bean]

A facade onto the mainframe banking system.



[Containers] techtribes.je



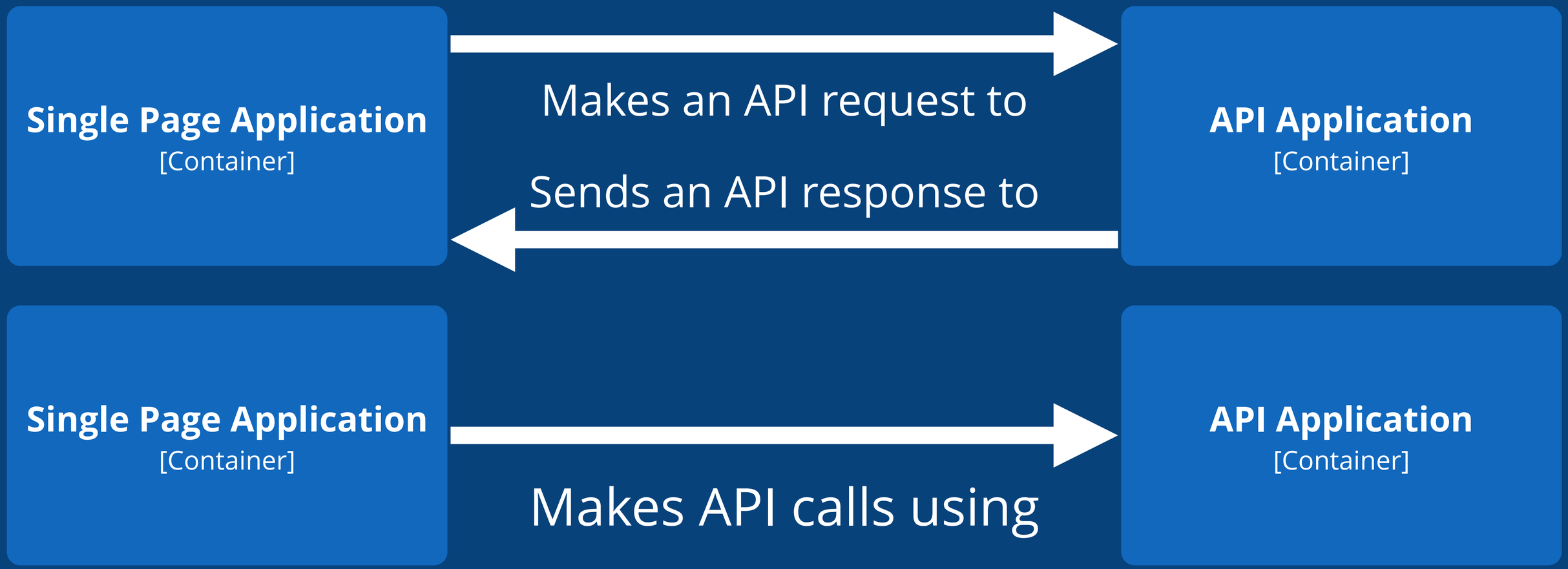
[Containers] techtribes.je

Lines

Favour uni-directional lines showing the most important dependencies or data flow, with an annotation to be explicit about the purpose of the line and direction

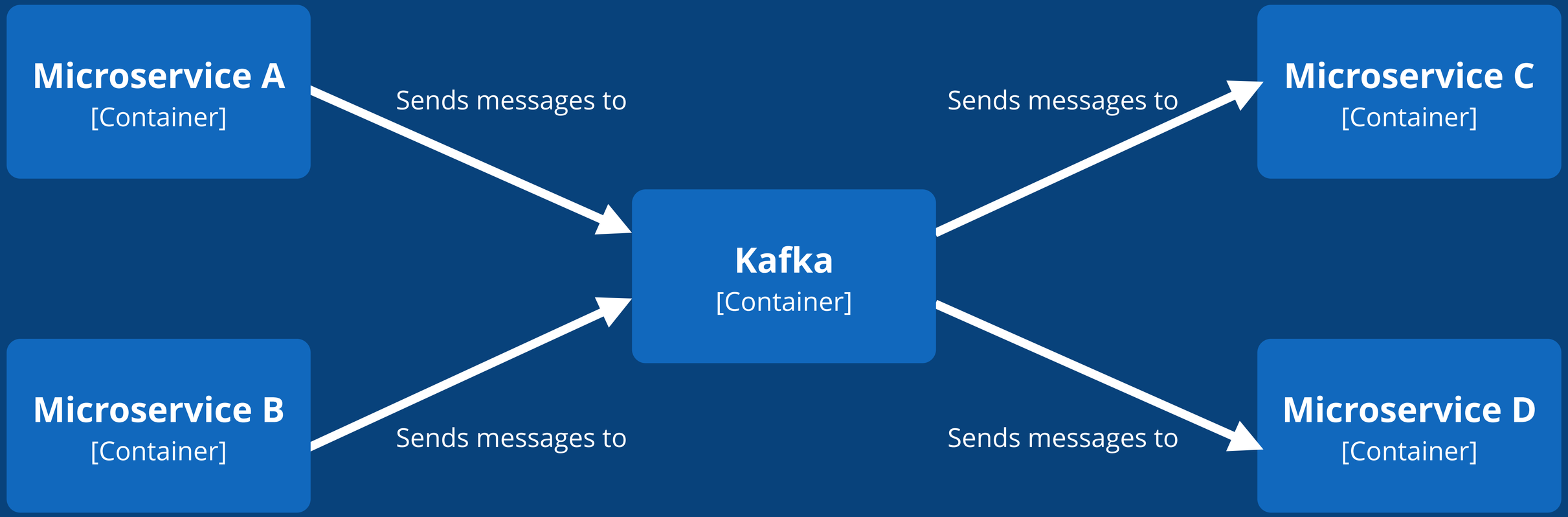
No

Yes

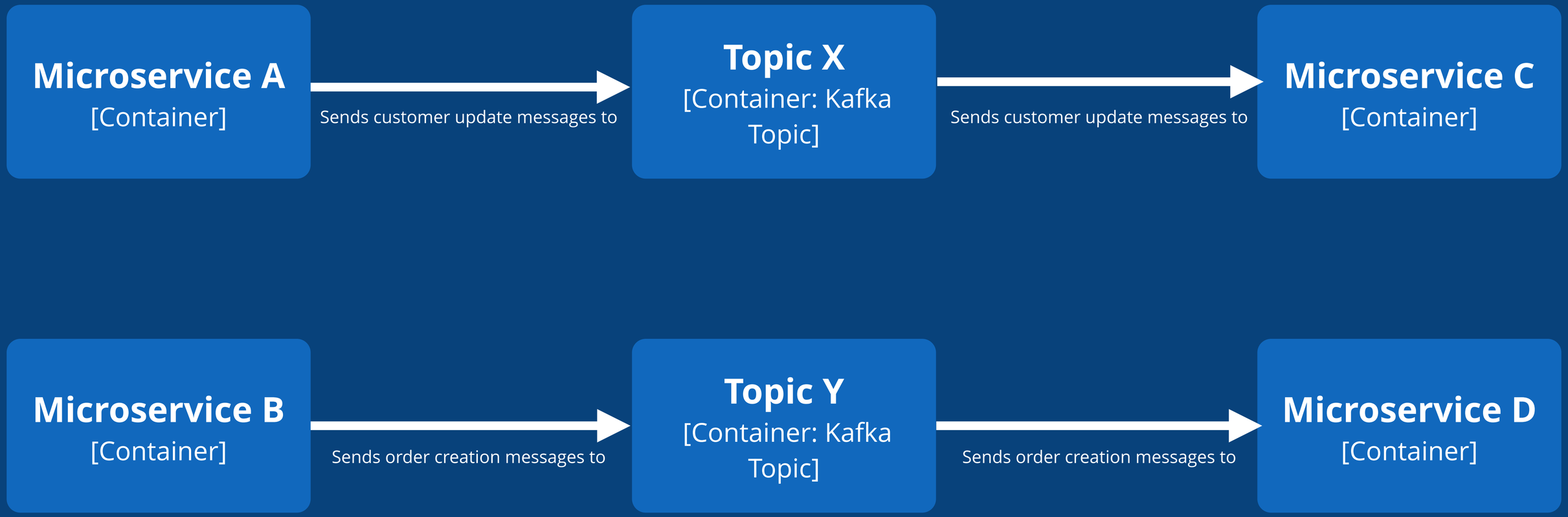


Summarise the intent of the relationship

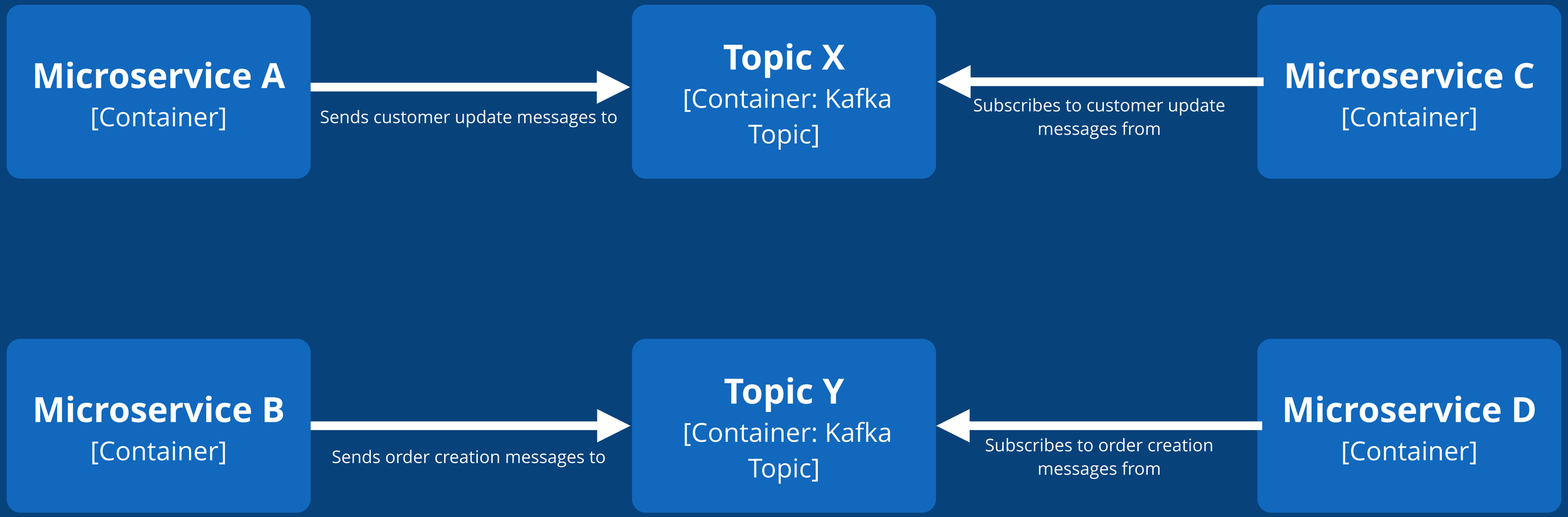
Show both directions when the intents are different



Beware of hiding the true story



Beware of hiding the true story



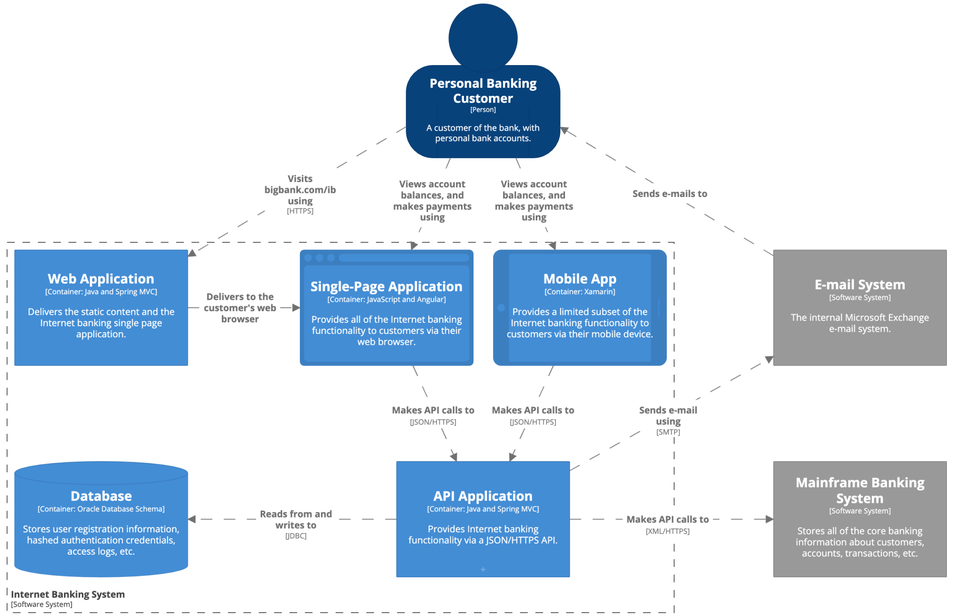
Beware of hiding the true story

Beware of hiding the true story

Key/legend

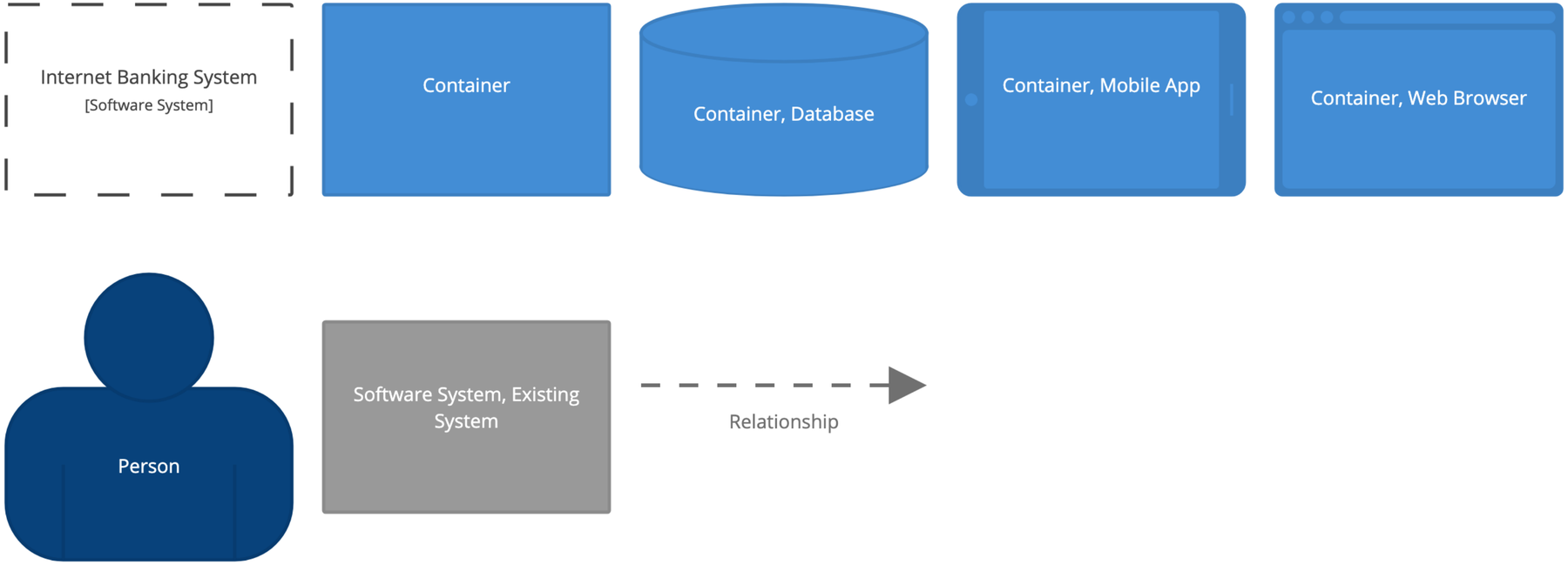
Explain shapes, line styles, colours, borders, acronyms, etc

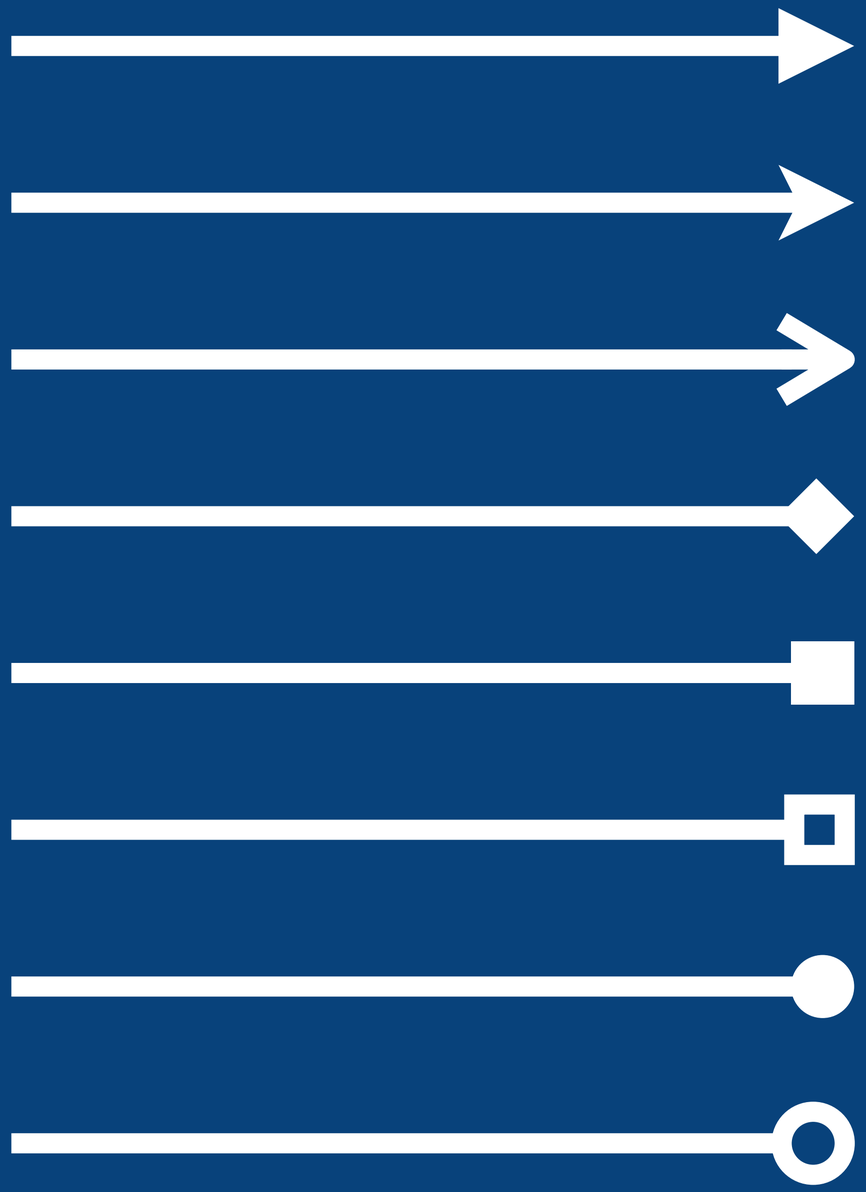
… even if your notation seems obvious!



Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Tue Oct 29 2019 15:41:33 GMT+0000 (Greenwich Mean Time)

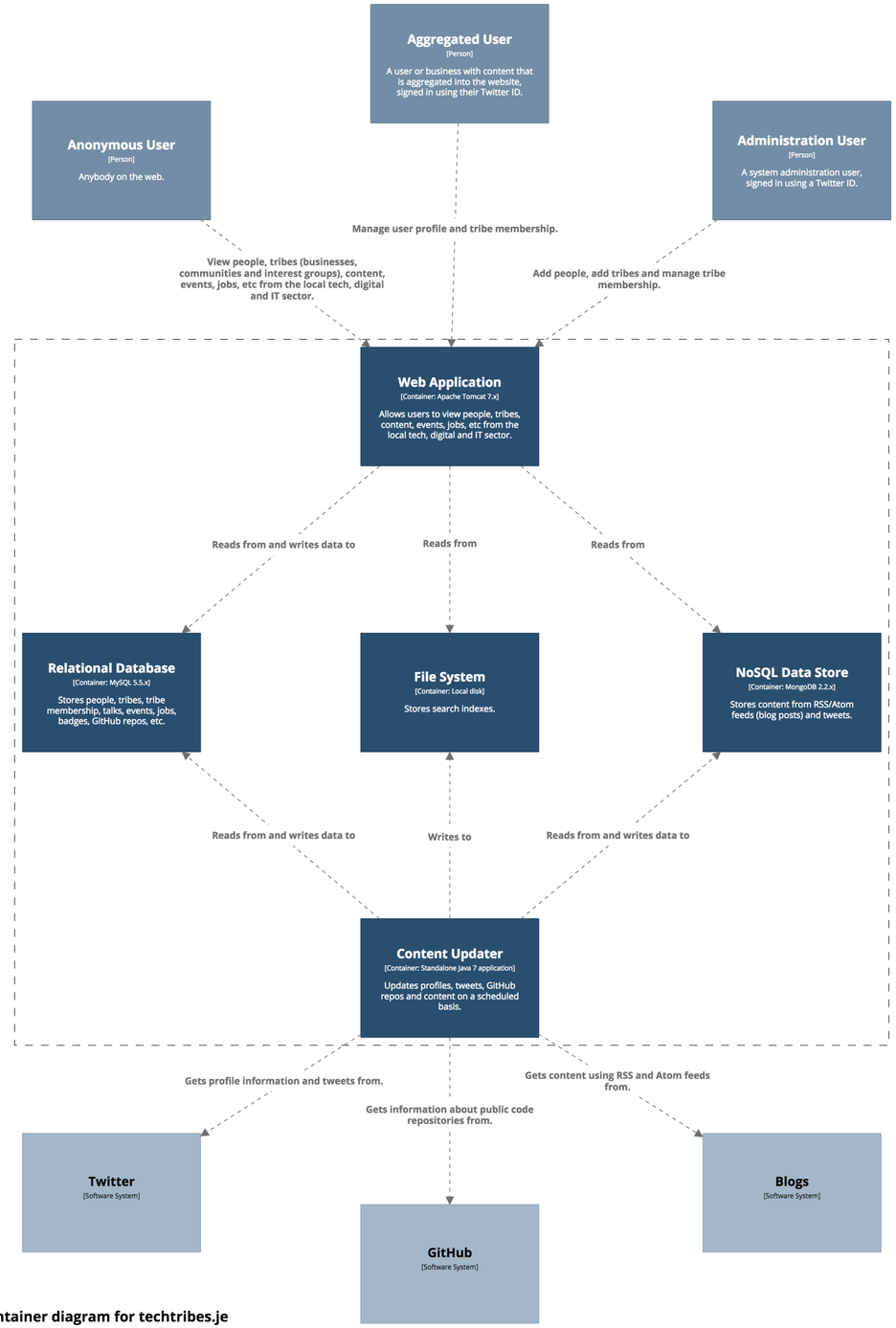




Arrowheads

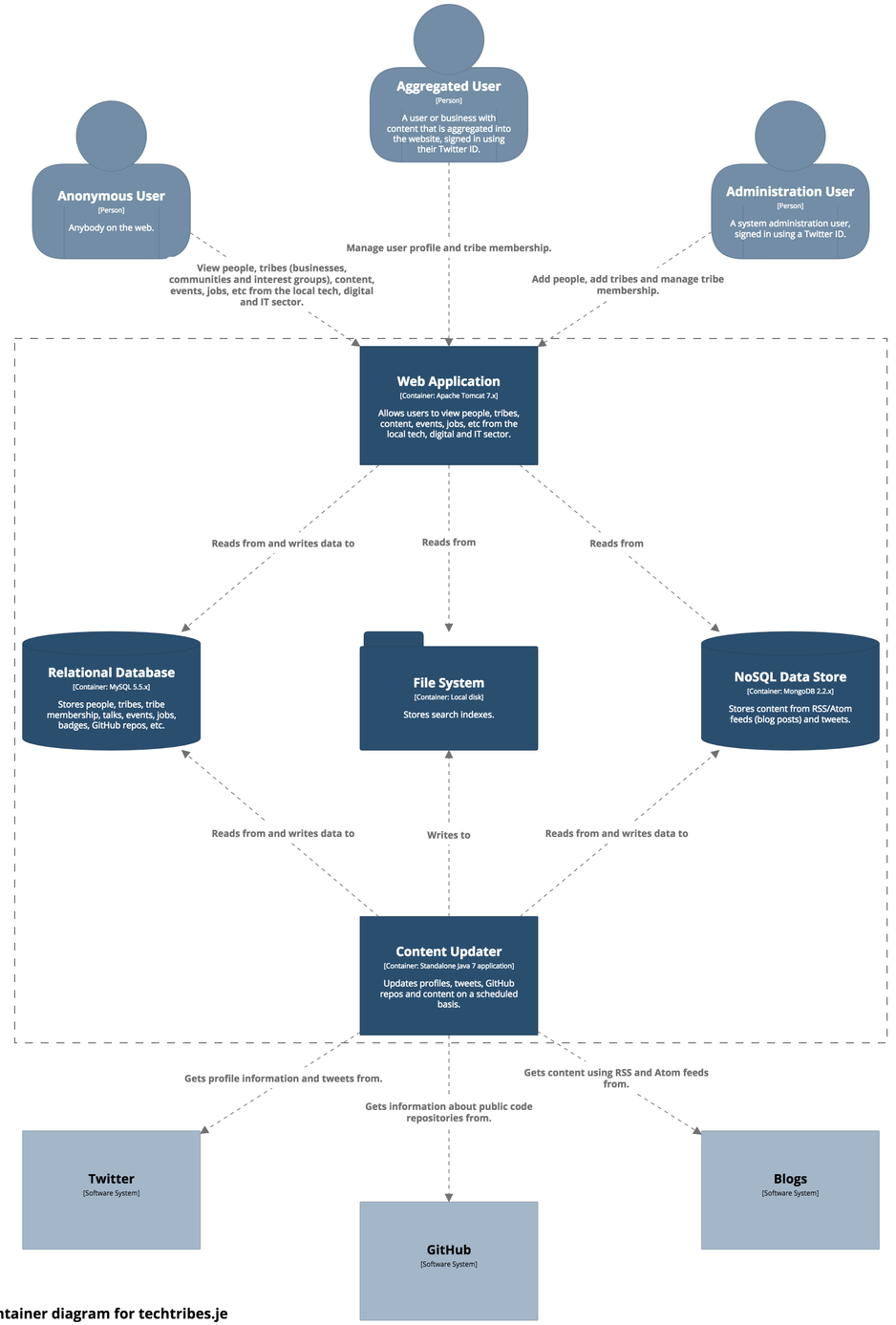
Be careful, using different arrowheads is very subtle; readers may miss them

Use shape, colour and size to **complement** a diagram that already makes sense



Container diagram for techtribes.je

Friday 12 May 2017 10:42 UTC



Container diagram for techtribes.je

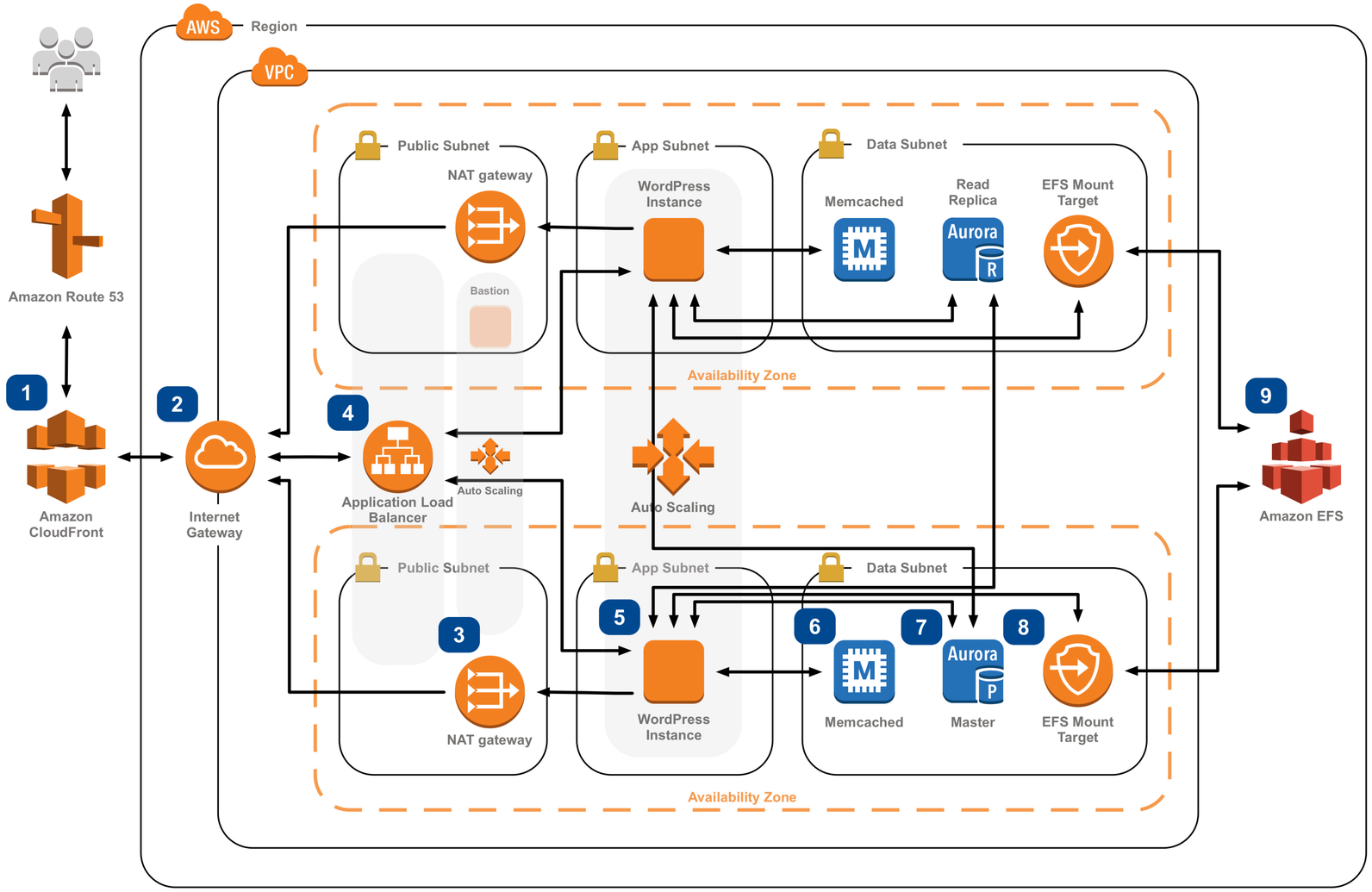
Monday 27 February 2017 09:39 UTC

Be careful with **icons**

WordPress Hosting

How to run WordPress on AWS

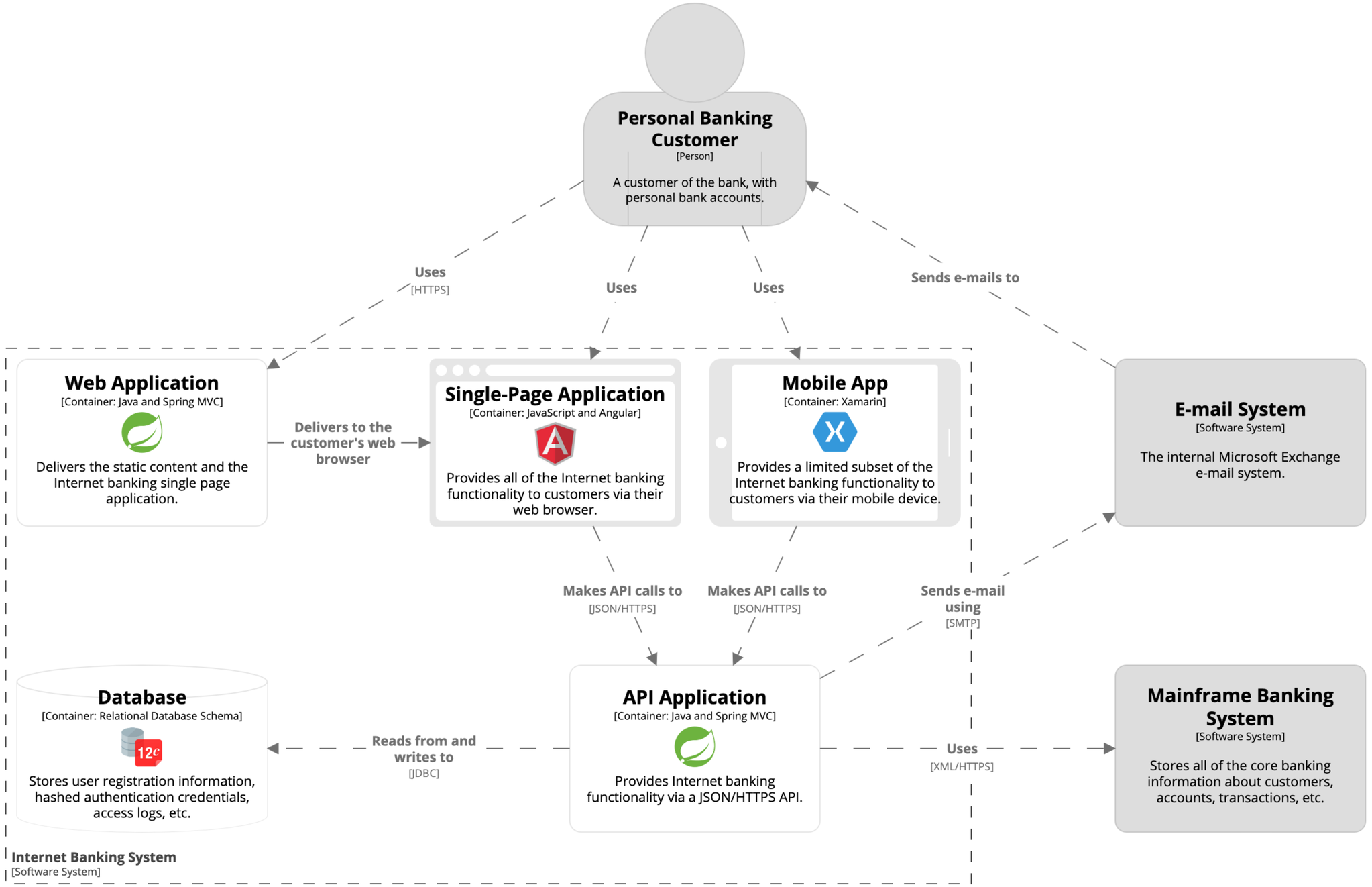
WordPress is one of the world's most popular web publishing platforms, being used to publish 27% of all websites, from personal blogs to some of the biggest news sites. This reference architecture simplifies the complexity of deploying a scalable and highly available WordPress site on AWS.



1. Static and dynamic content is delivered by **Amazon CloudFront**.
2. An **Internet gateway** allows communication between instances in your VPC and the Internet.
3. **NAT gateways** in each public subnet enable Amazon EC2 instances in private subnets (App & Data) to access the Internet.
4. Use an **Application Load Balancer** to distribute web traffic across an Auto Scaling Group of Amazon EC2 instances in multiple AZs.
5. Run your WordPress site using an **Auto Scaling group** of **Amazon EC2 instances**. Install the latest versions of WordPress, Apache web server, PHP 7, and OPcache and build an Amazon Machine Image that will be used by the Auto Scaling group launch configuration to launch new instances in the Auto Scaling group.
6. If database access patterns are read-heavy, consider using a WordPress plugin that takes advantage of a caching layer like **Amazon ElastiCache** (**Memcached**) in front of the database layer to cache frequently accessed data.
7. Simplify your database administration by running your database layer in **Amazon RDS** using either Aurora or MySQL.
8. Amazon EC2 instances access shared WordPress data in an Amazon EFS file system using **Mount Targets** in each AZ in your VPC.
9. Use **Amazon EFS**, a simple, highly available, and scalable network file system so WordPress instances have access to your shared, unstructured WordPress data, like php files, config, themes, plugins, etc.

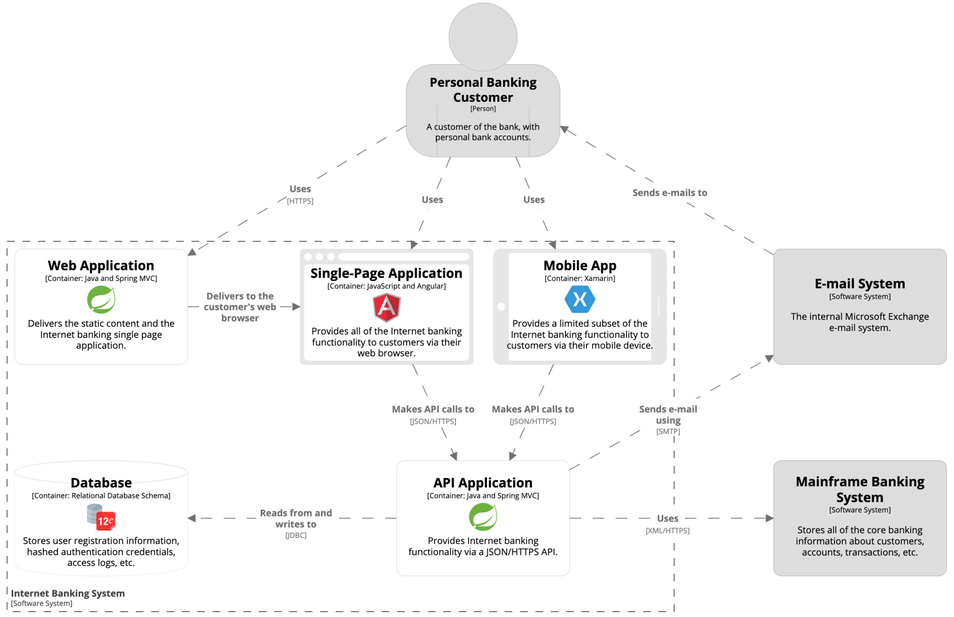
amazon web services

AWS Reference Architectures



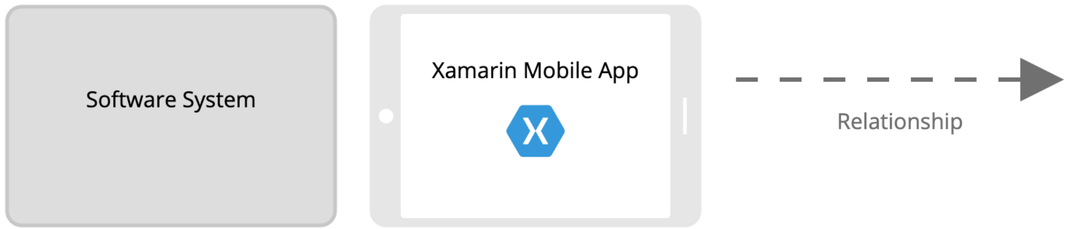
Container diagram for Internet Banking System

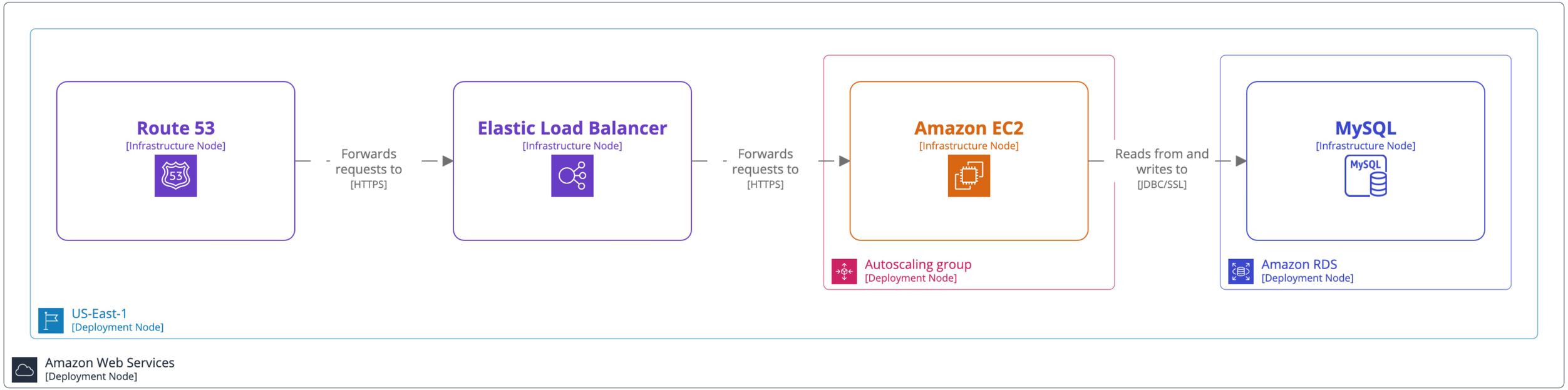
The container diagram for the Internet Banking System.
Workspace last modified: Sat Jan 11 2020 14:47:20 GMT+0000 (Greenwich Mean Time)

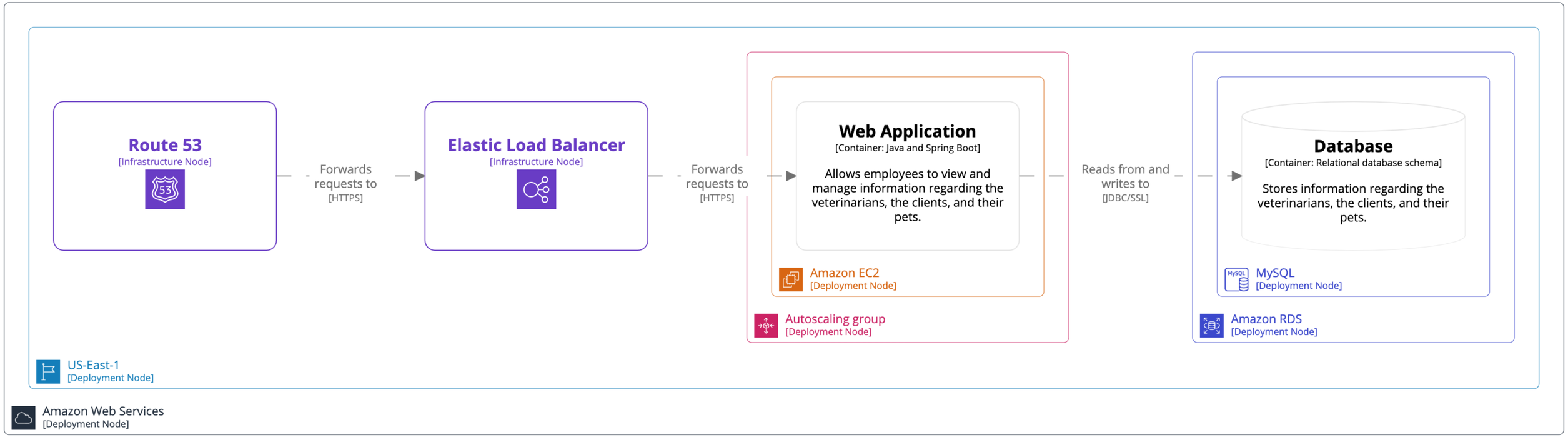

Personal Banking Customer
[Person]
A customer of the bank, with personal bank accounts.
Uses
[HTTPS]
Uses
Uses
Sends e-mails to
Web Application
[Container: Java and Spring MVC]
Delivers the static content and the Internet banking single page application.
Delivers to the customer's web browser
Single-Page Application
[Container: JavaScript and Angular]
Provides all of the Internet banking functionality to customers via their web browser.
Mobile App
[Container: Xamarin]
Provides a limited subset of the Internet banking functionality to customers via their mobile device.
E-mail System
[Software System]
The internal Microsoft Exchange e-mail system.
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Sends e-mail using
[SMTP]
Database
[Container: Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Reads from and writes to
[JDBC]
API Application
[Container: Java and Spring MVC]
Provides Internet banking functionality via a JSON/HTTPS API.
Uses
[XML/HTTPS]
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.
Internet Banking System
[Software System]
Container diagram for Internet Banking System
The container diagram for the Internet Banking System.
Workspace last modified: Sat Jan 11 2020 14:47:20 GMT+0000 (Greenwich Mean Time)

#sa4d-1d


Internet Banking System
[Software System]
Angular Application
Java and Spring Application
Oracle Database
Person


Software System
Xamarin Mobile App
Relationship


Route 53
[Infrastructure Node]
Forwards requests to
[HTTPS]
Elastic Load Balancer
[Infrastructure Node]
Forwards requests to
[HTTPS]
Amazon EC2
[Infrastructure Node]
Autoscaling group
[Deployment Node]
Reads from and writes to
[JDBC/SSL]
MySQL
[Infrastructure Node]
Amazon RDS
[Deployment Node]
US-East-1
[Deployment Node]
Amazon Web Services
[Deployment Node]


Route 53
[Infrastructure Node]
Forwards requests to
[HTTPS]
Elastic Load Balancer
[Infrastructure Node]
Forwards requests to
[HTTPS]
Web Application
[Container: Java and Spring Boot]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
Amazon EC2
[Deployment Node]
Autoscaling group
[Deployment Node]
Reads from and writes to
[JDBC/SSL]
Database
[Container: Relational database schema]
Stores information regarding the veterinarians, the clients, and their pets.
MySQL
[Deployment Node]
Amazon RDS
[Deployment Node]
US-East-1
[Deployment Node]
Amazon Web Services
[Deployment Node]

Use icons to supplement text,
not replace it

Increase the **readability** of software architecture diagrams, so they can **stand alone**

Any narrative should **complement** the diagram rather than explain it

Notation, notation, notation

A software architecture diagram review checklist

Diagram review tool | Printable PDF version

General

Does the diagram have a title?	◉ Yes	○ No	
Do you understand what the diagram type is?	○ Yes	◉ No	
Do you understand what the diagram scope is?	○ Yes	◉ No	
Does the diagram have a key/legend?	◉ Yes	○ No	

Abstractions first, notation second

Ensure that your team has a ubiquitous language to describe software architecture

Thank you!

Simon Brown
@simonbrown