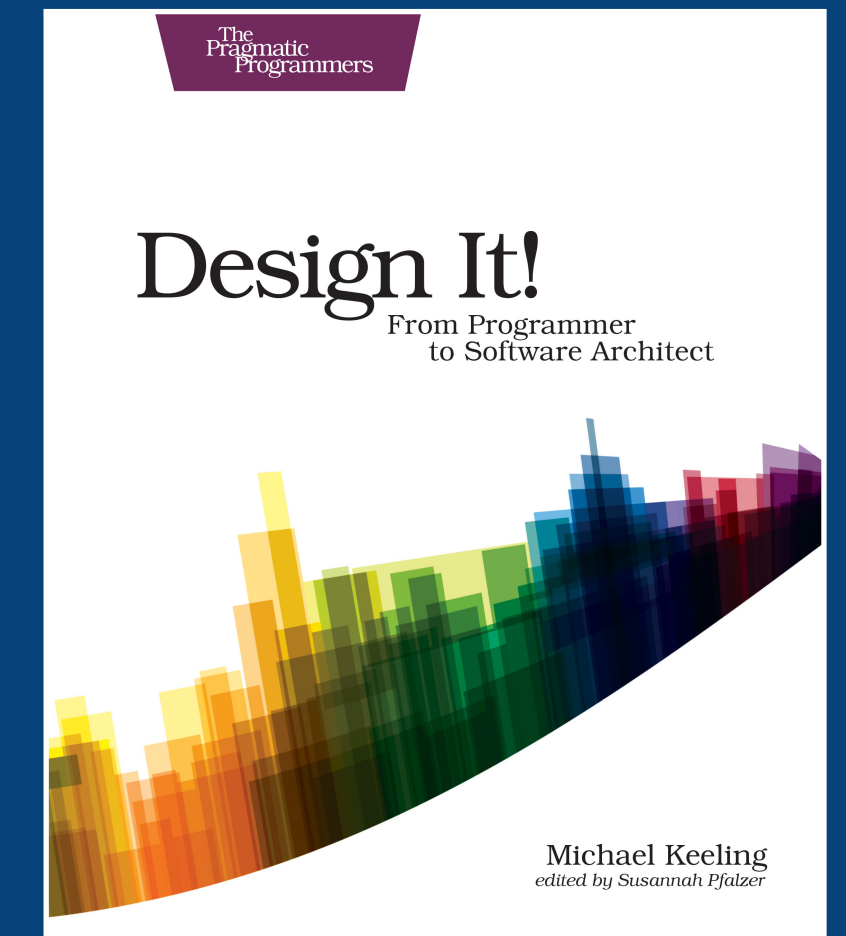
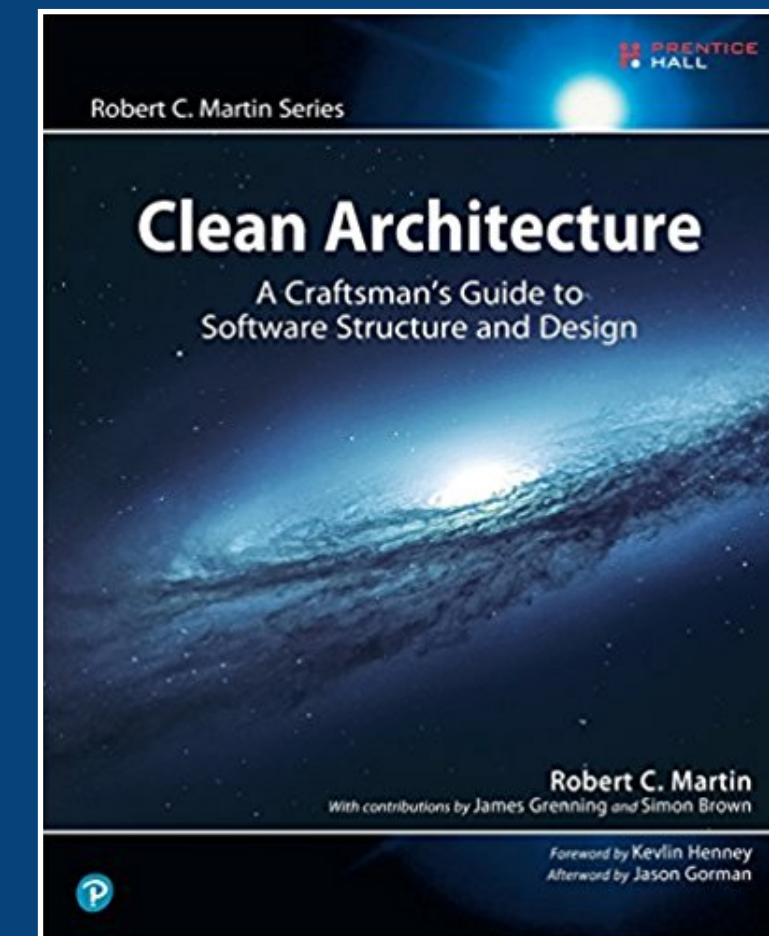
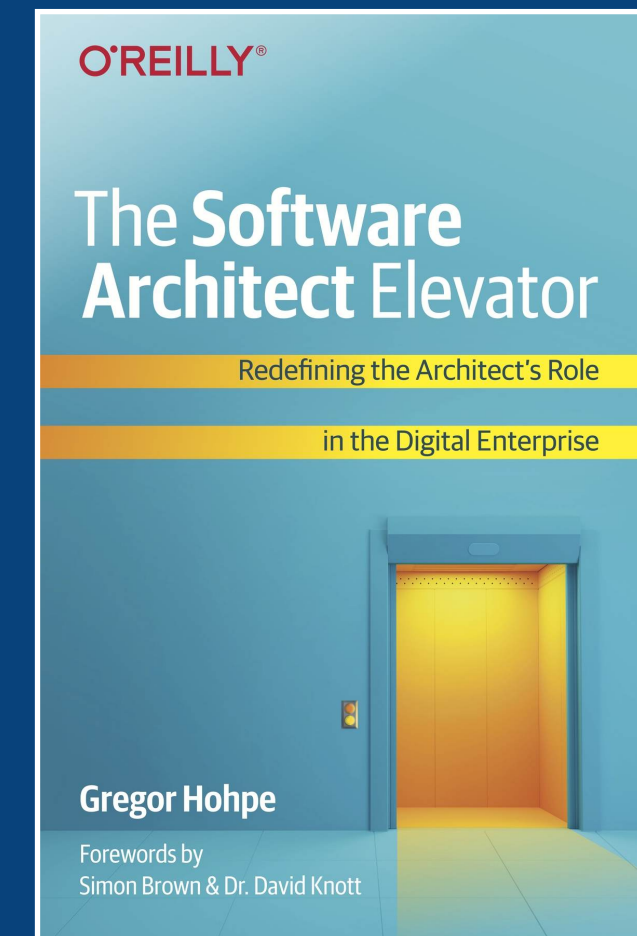
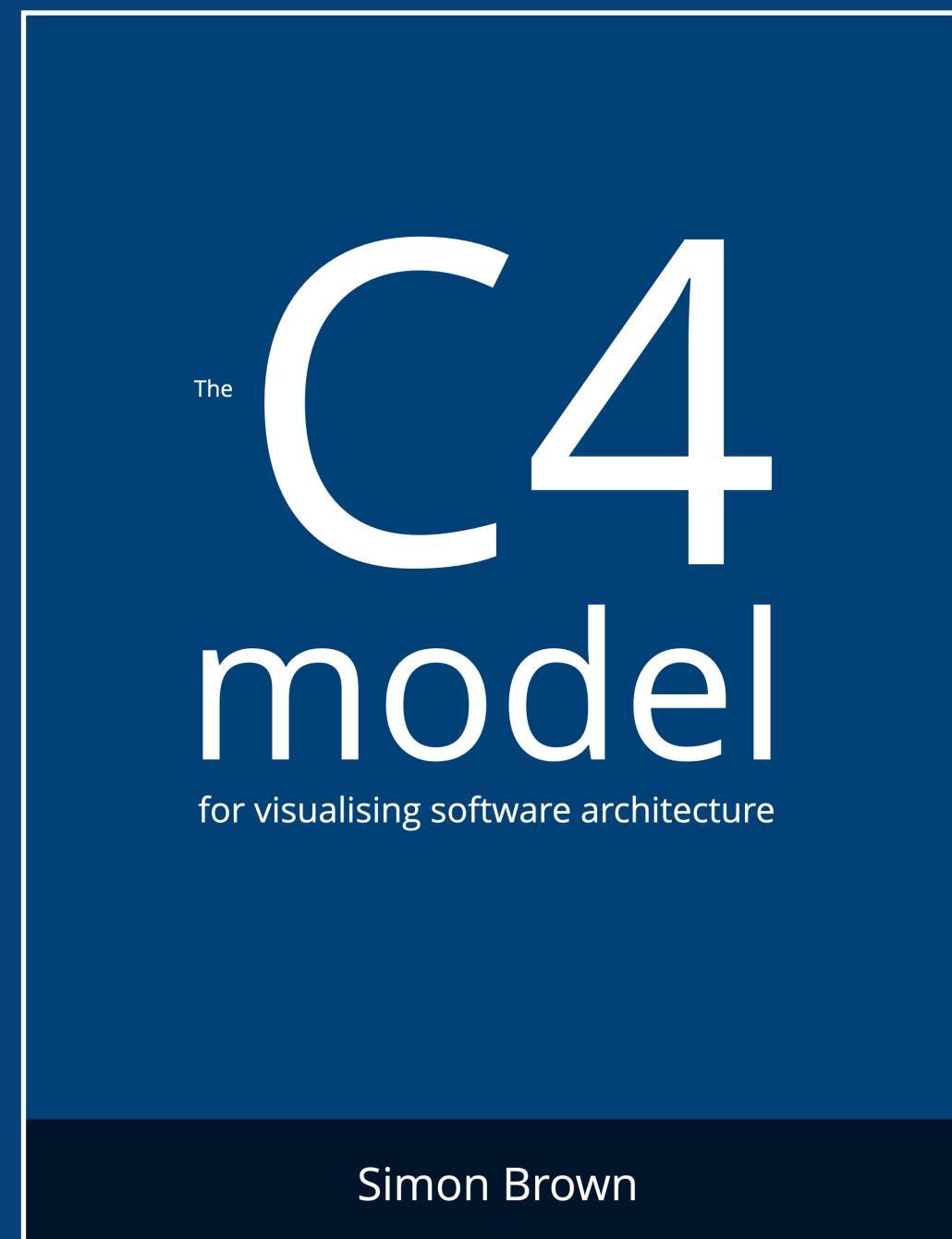


The C4 model for visualising software architecture

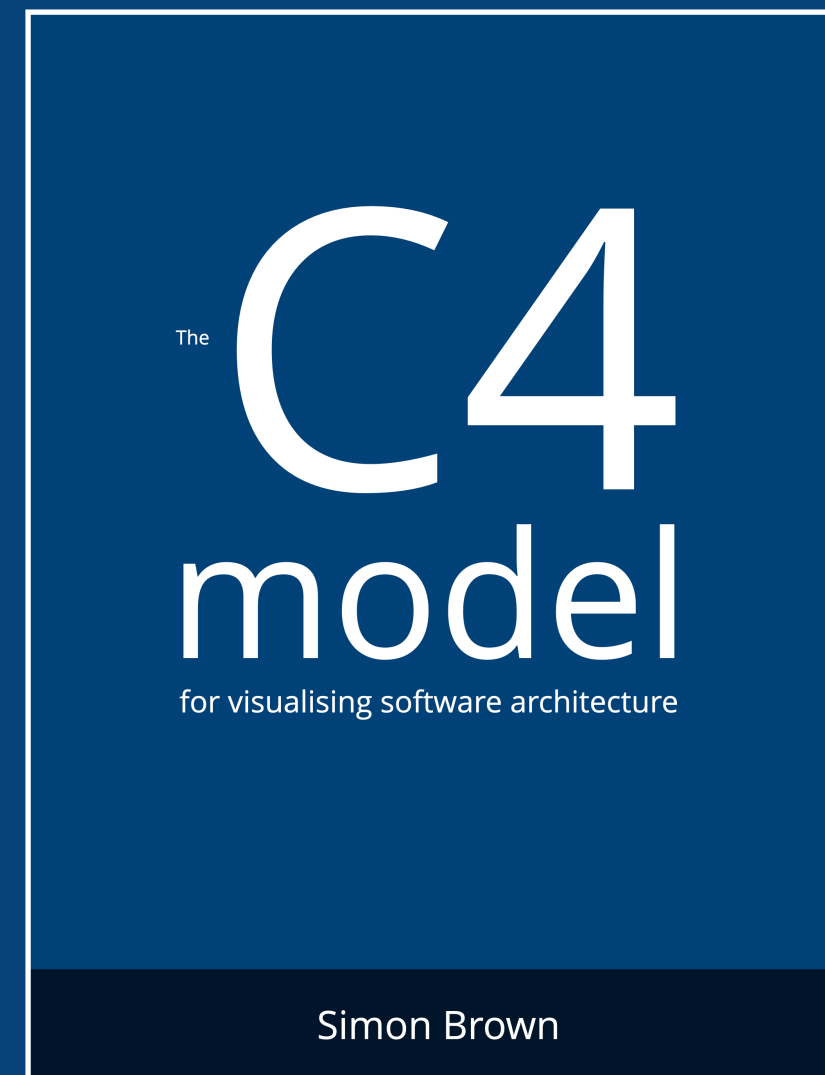
Simon Brown

Simon Brown

Independent consultant specialising in software architecture,
plus the creator of the C4 model and Structurizr



simonbrown.je



Buy both books for \$15
(expires midnight)

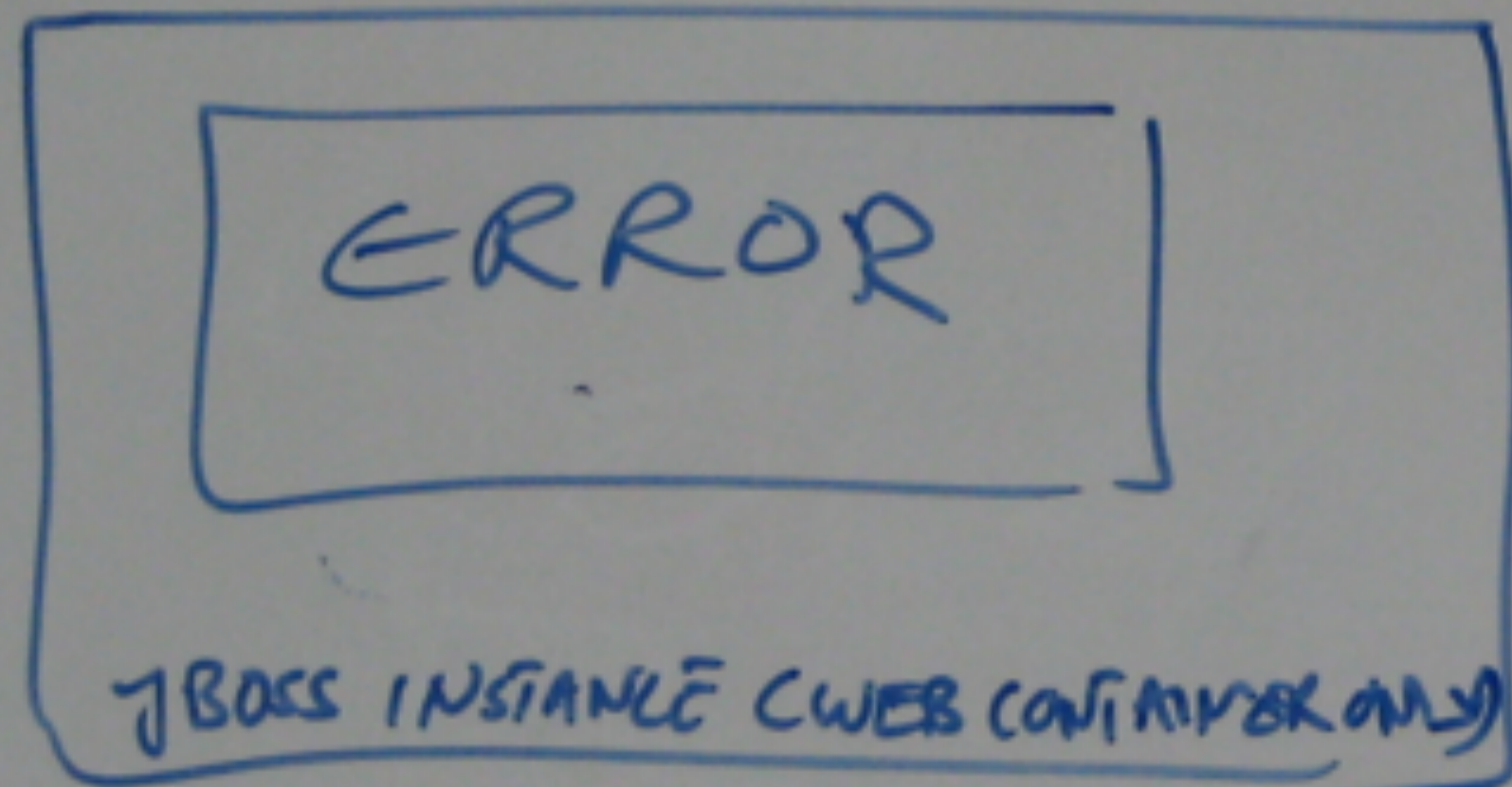
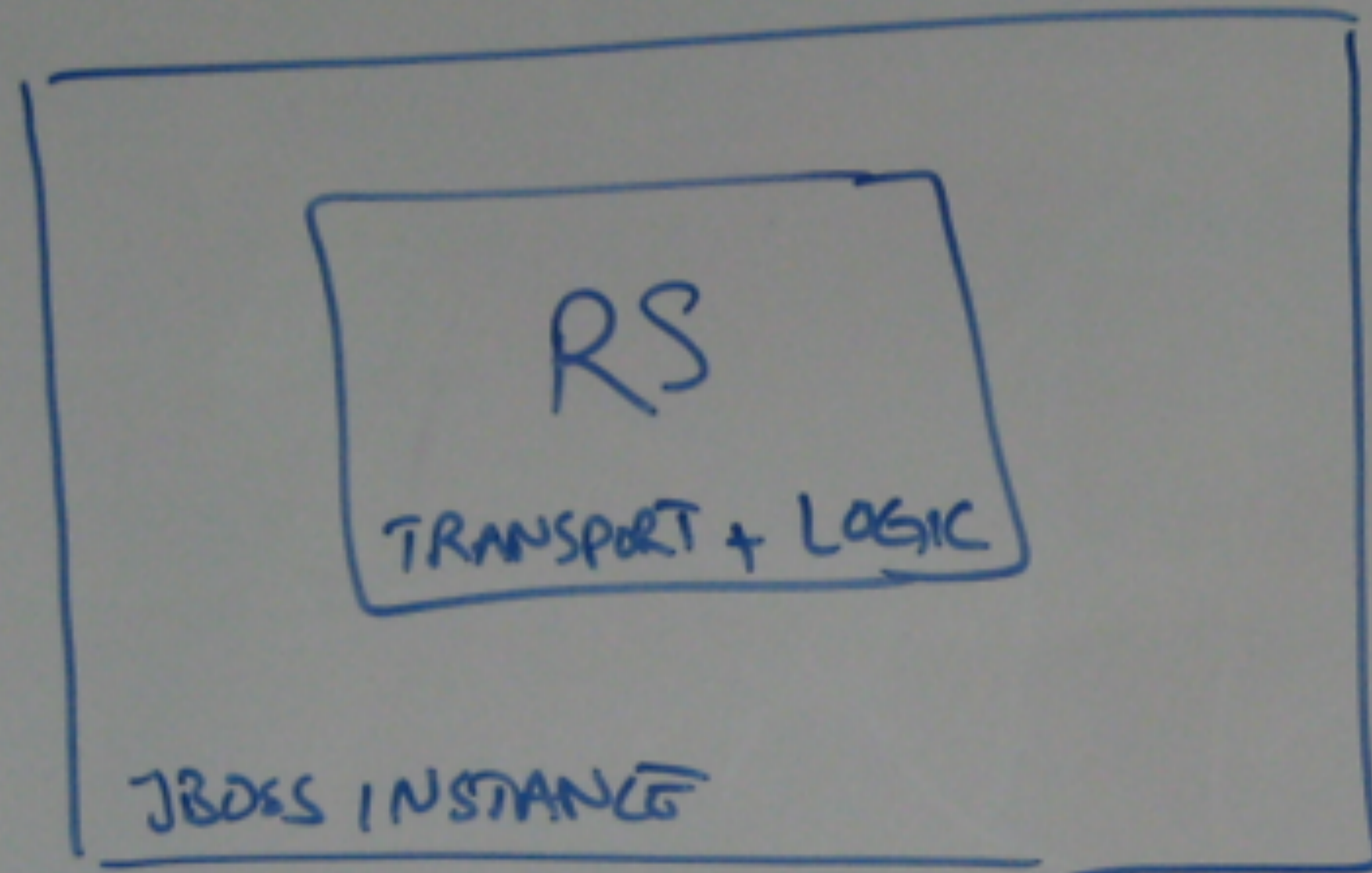


<https://leanpub.com/b/software-architecture/c/portotechhub2024>

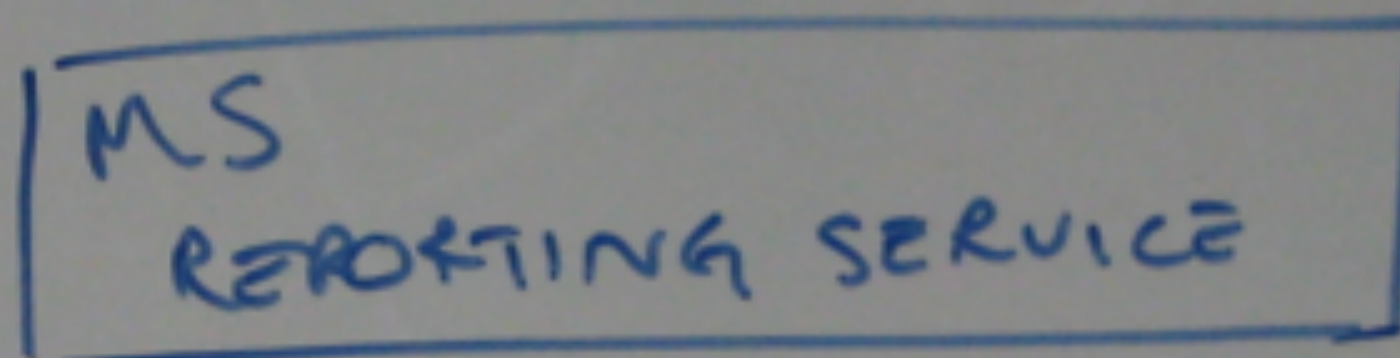
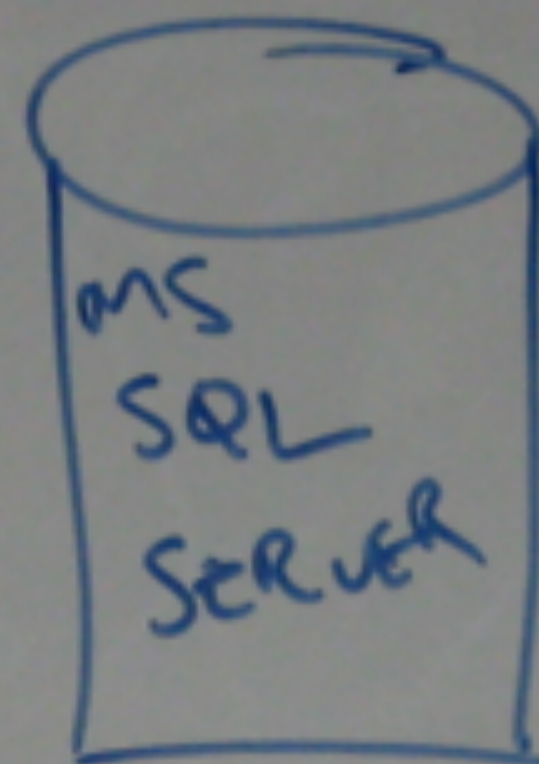
Moving fast in the same direction
as a team requires

good communication

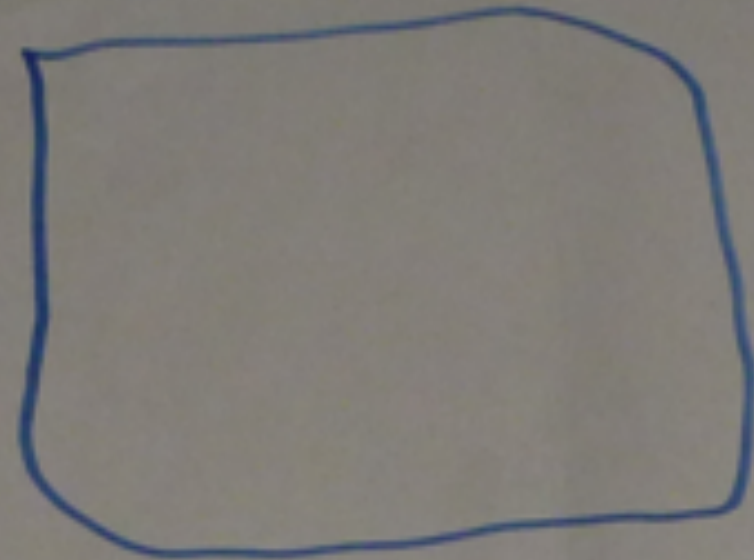
UNIX BOX



WINDOWS BOX



ASP
NET



LOGGING
SERVICE

PARAMETER
MANAGER

RISK
CALCULATION

REPORT
GENERATOR

DATA
IMPORT

AUDITING

VALIDATION

server

TDS

RDS

~~INTER~~
RISK
DATA

PARAMS

SECURITY

AUDIT

FUNCTIONAL VIEW

File Retriever

Scheduler

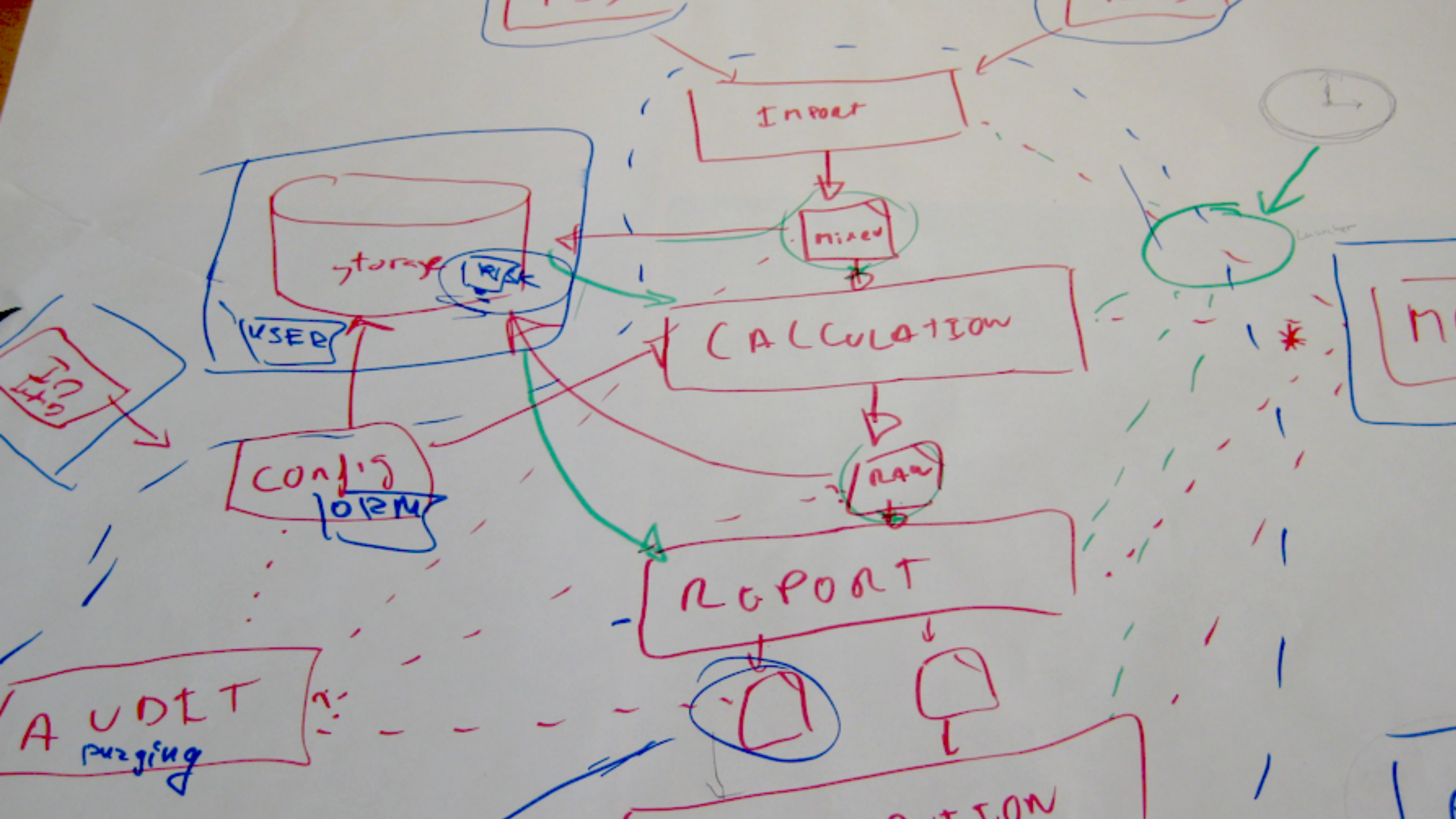
Auditing

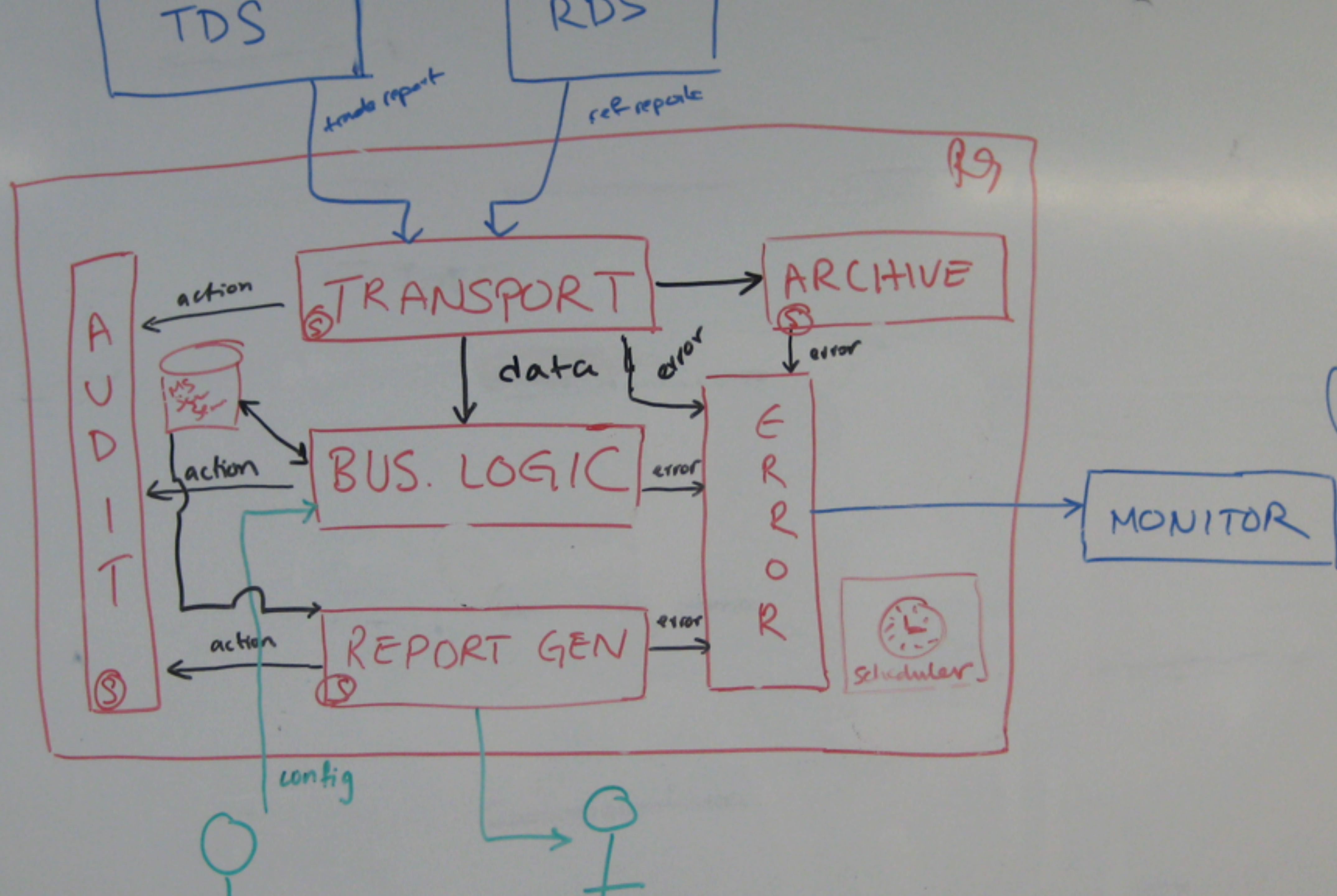
Reference
Archiver

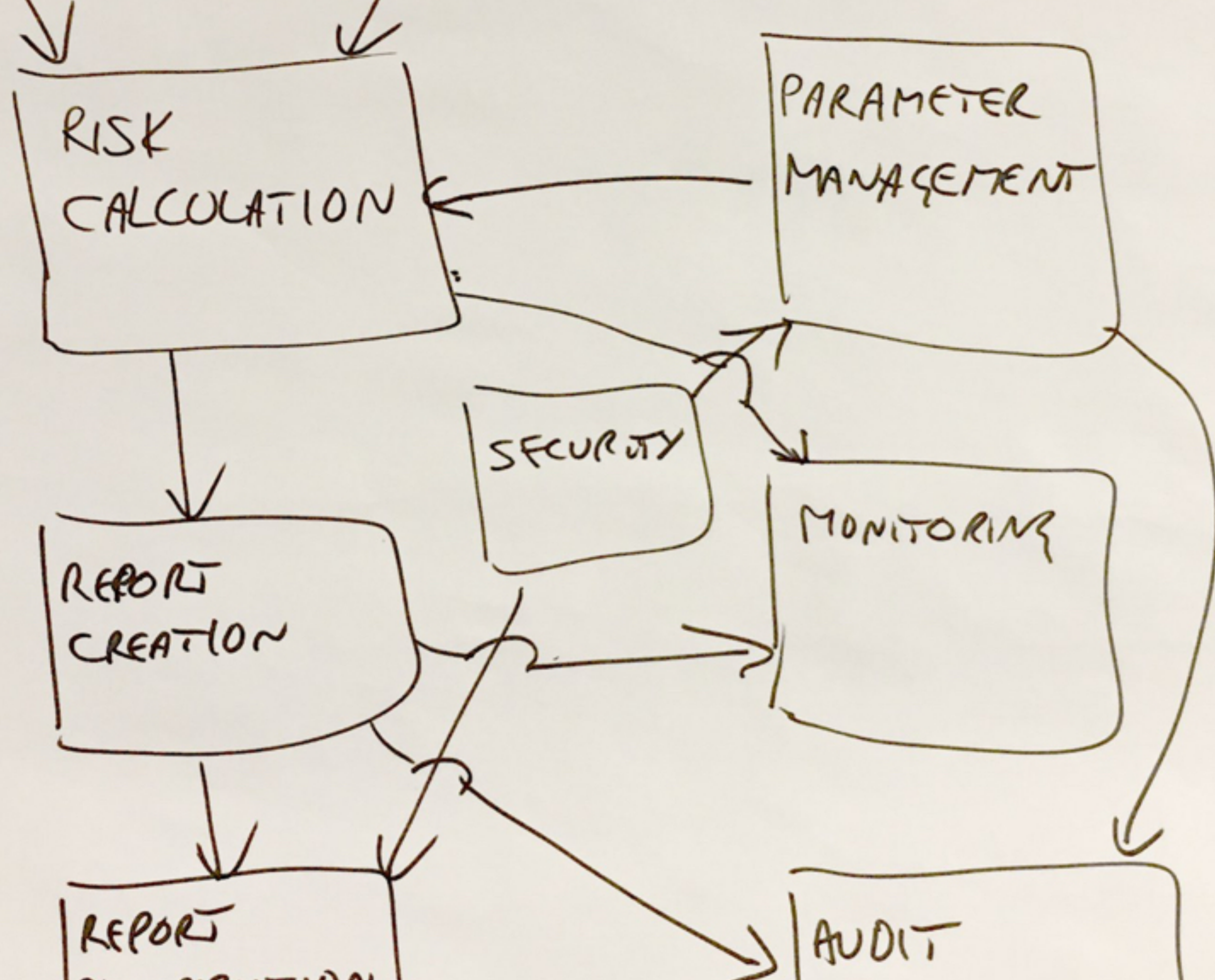
Risk Assessment
Processor

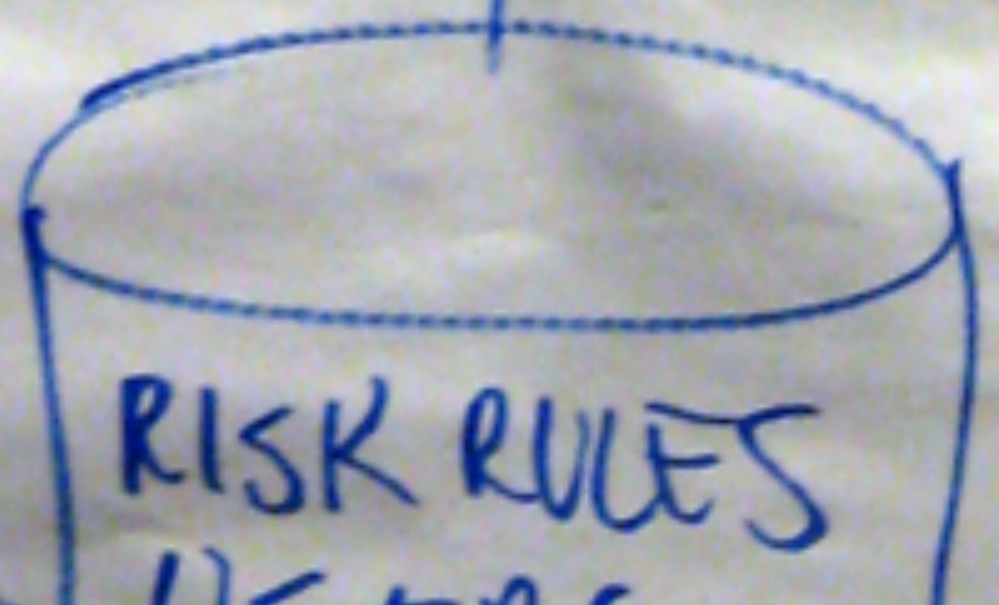
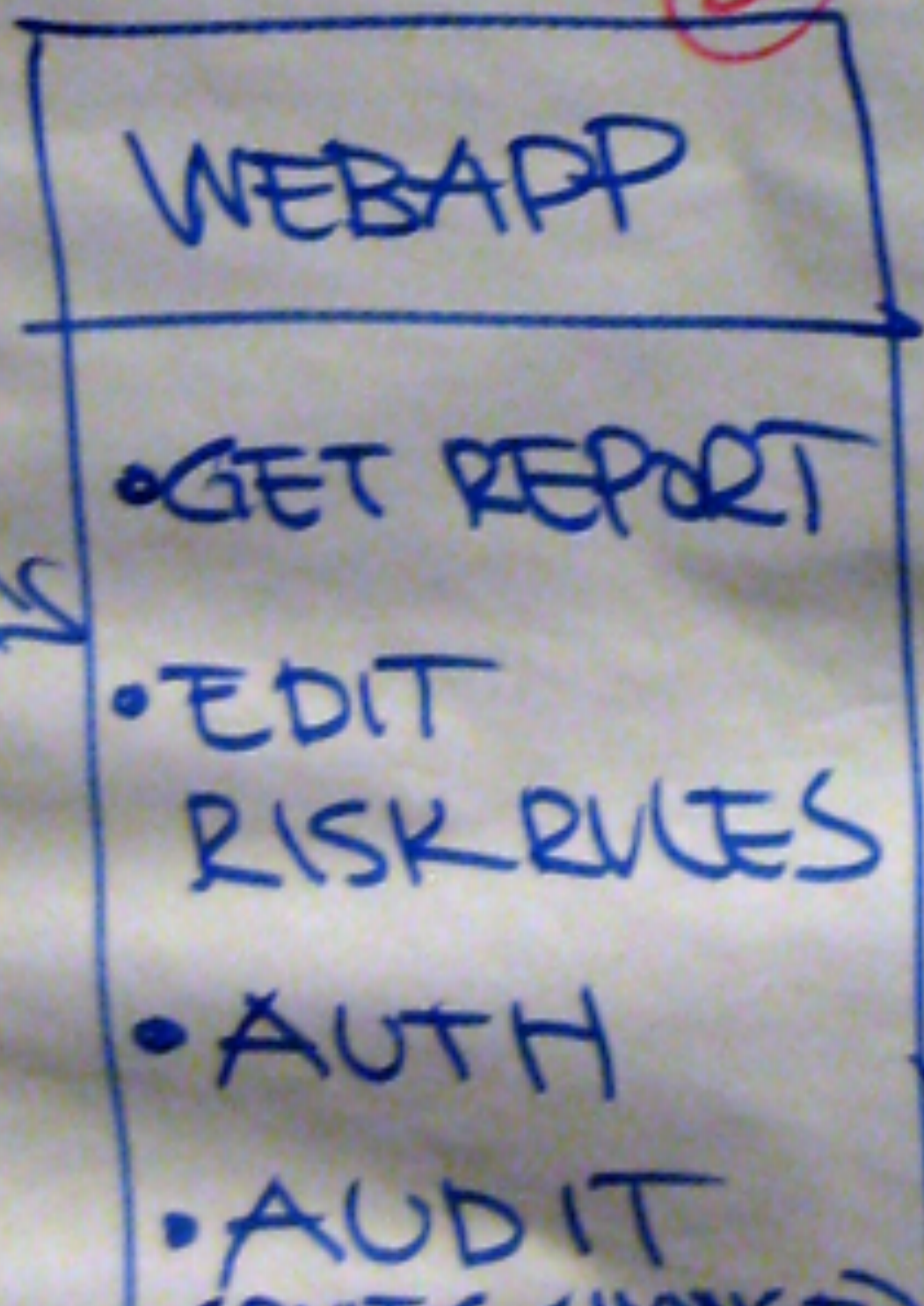
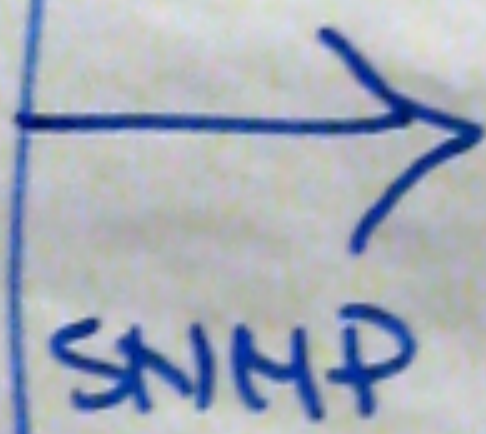
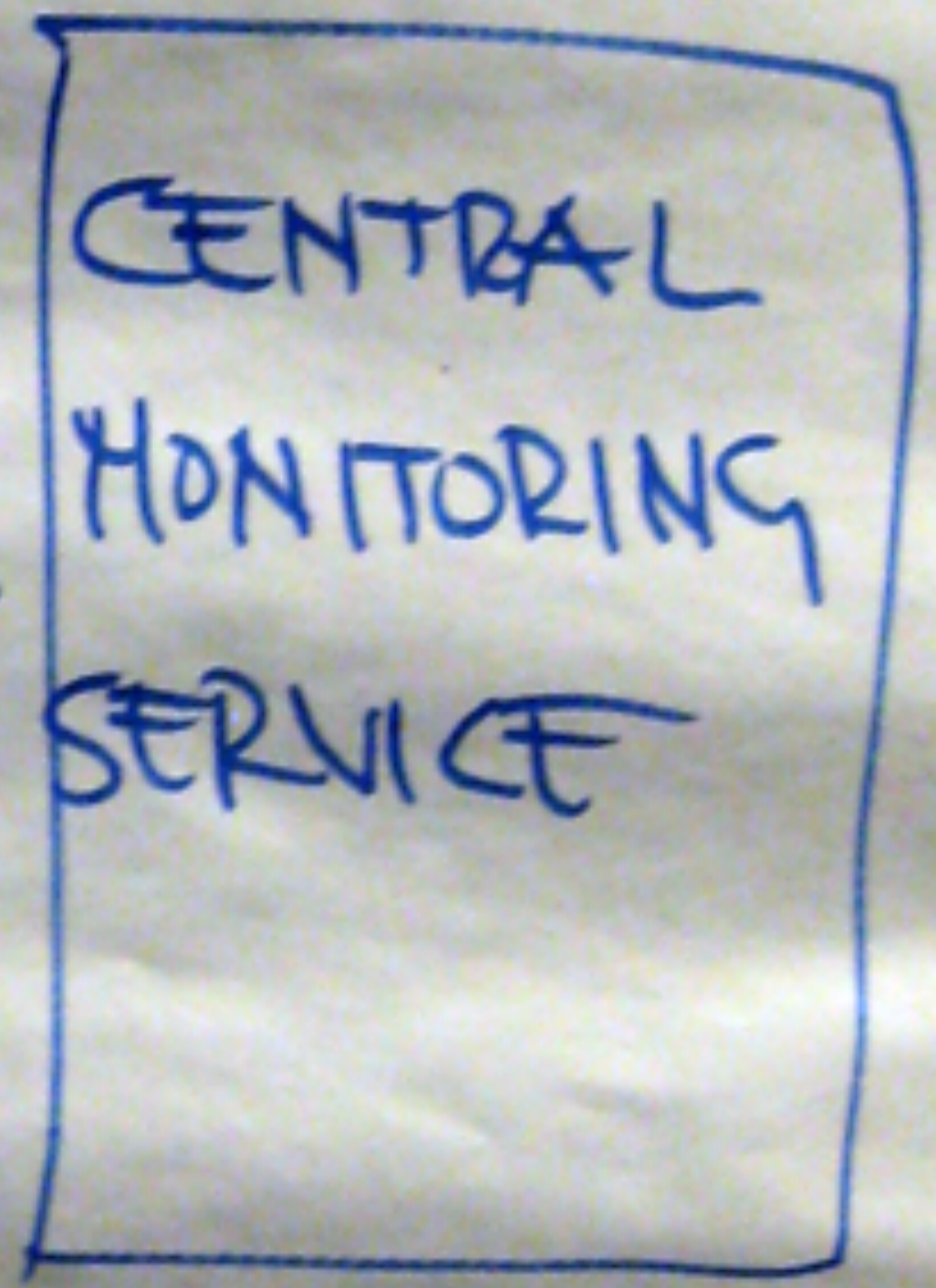
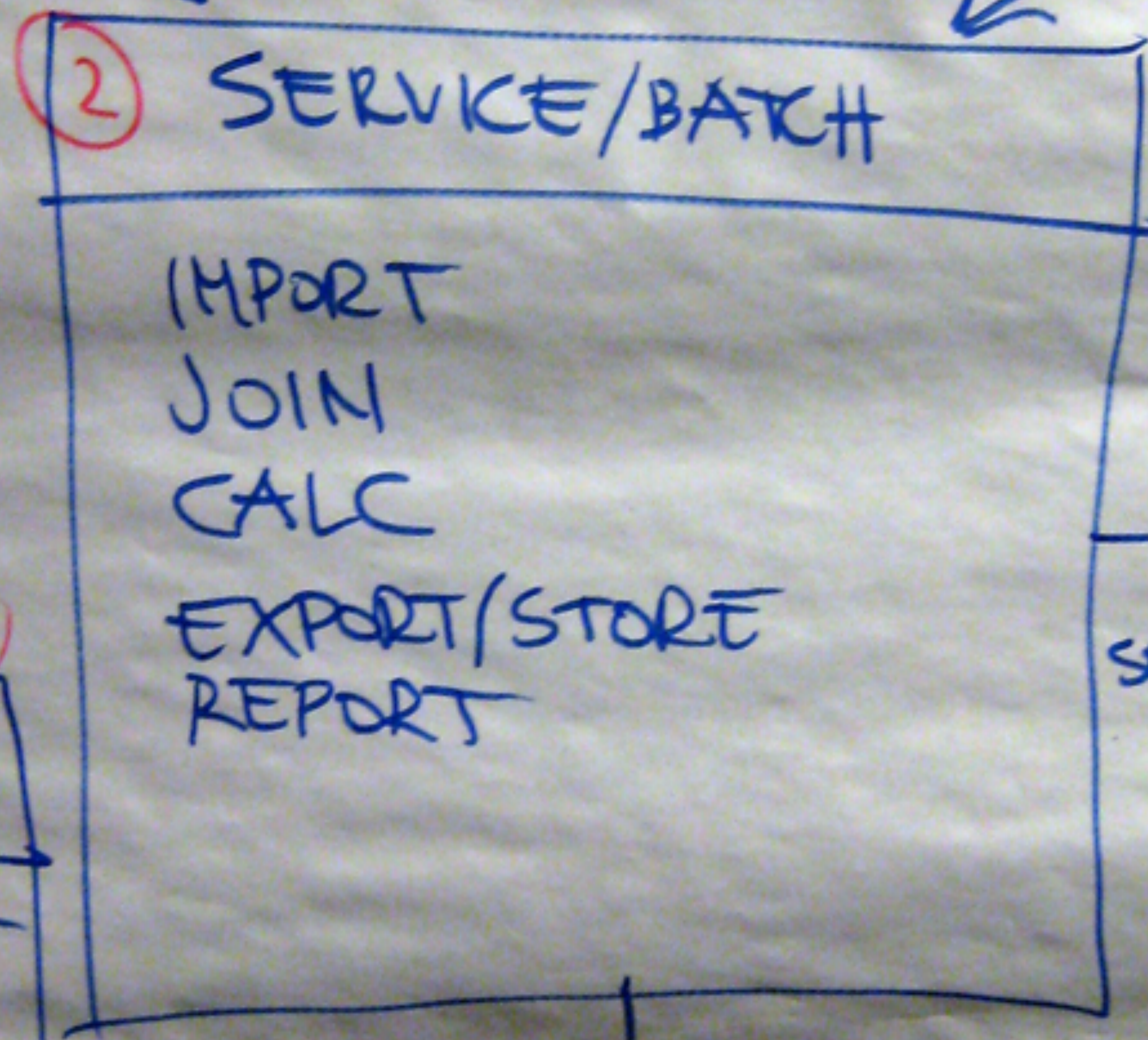
Risk Parameter
Configuration

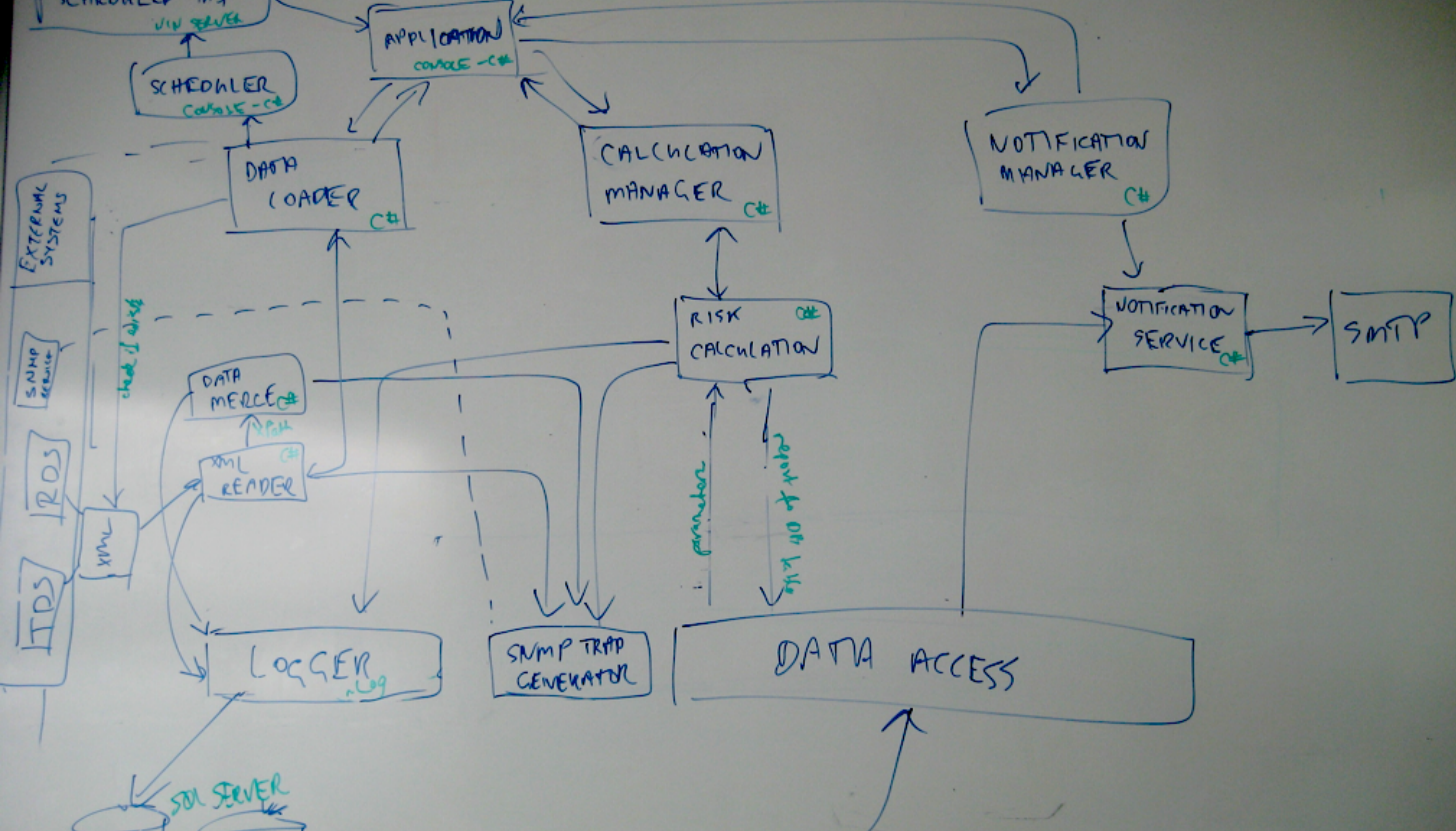
Report

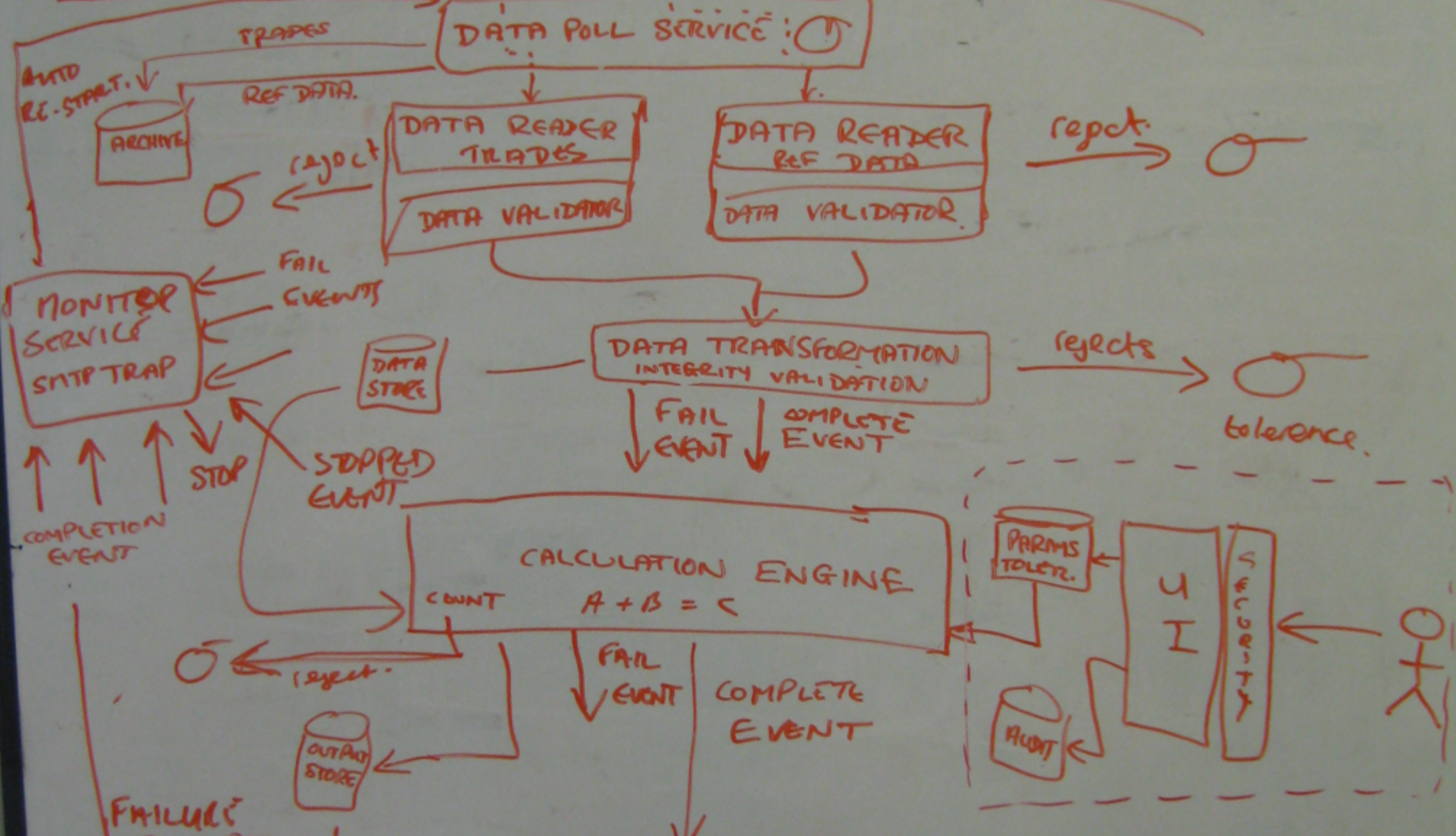


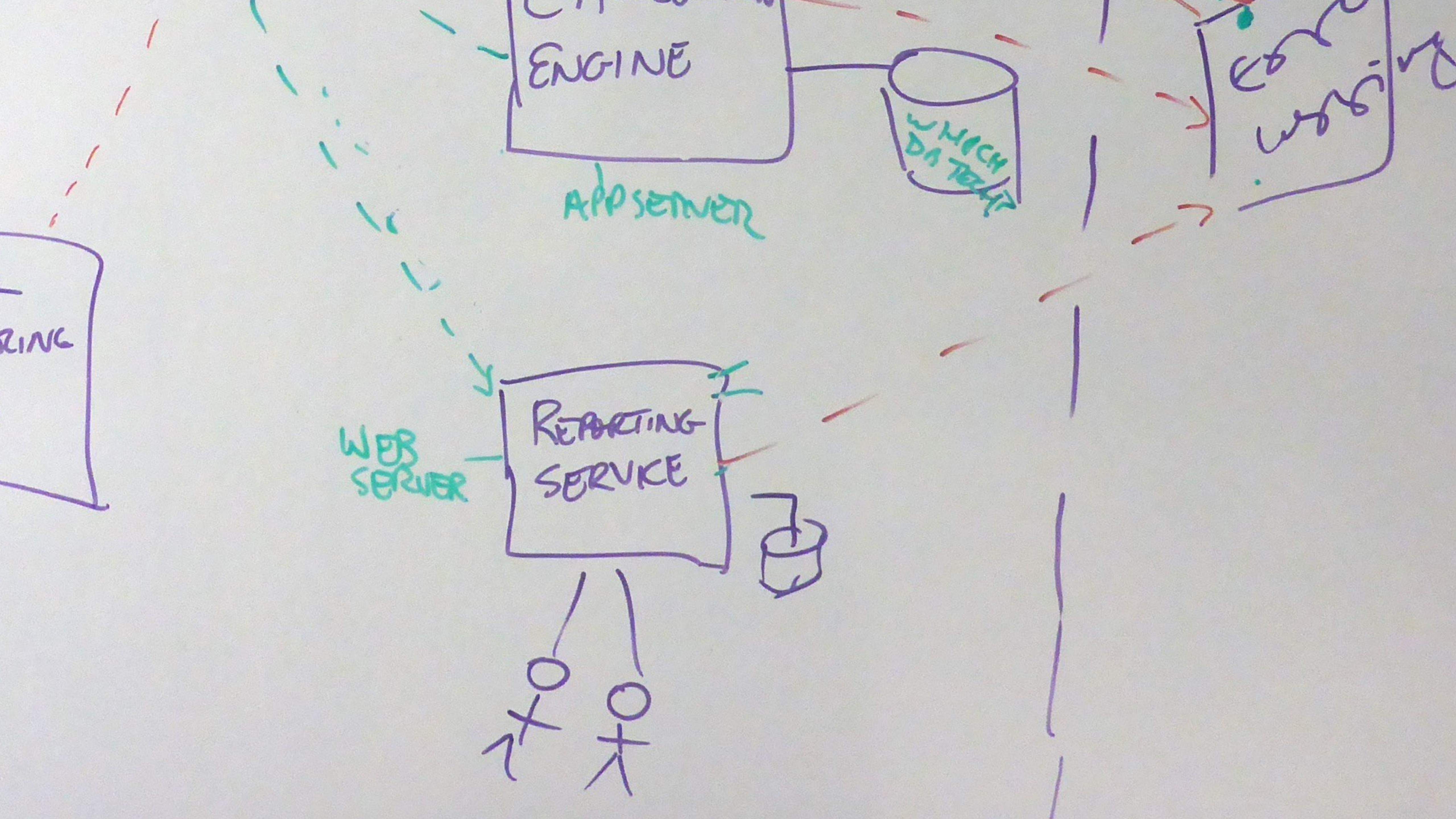














Params

Calcs

Params

ret - client

ret - client

~~Calcs~~ Risk outputs

Flow App Log

EH?

App
Export Log
Date
- Risk Cell
- calcs
- Par

Params - Distinct

Batch

Batch
B-id:
cust id



software architecture diagram



All

Images

Videos

News

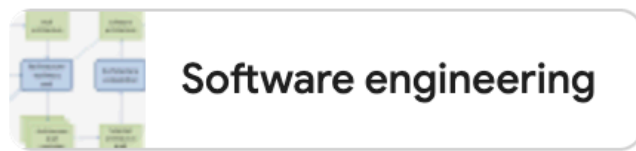
Books

Maps

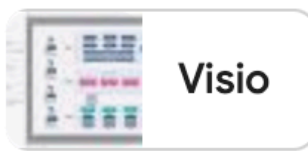
Web

More

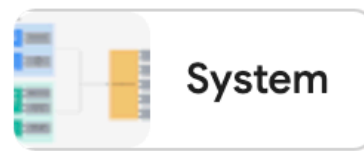
Tools



Software engineering



Visio



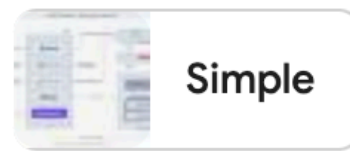
System



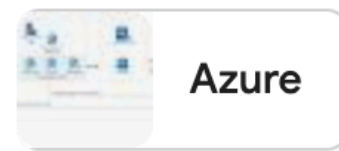
Uml



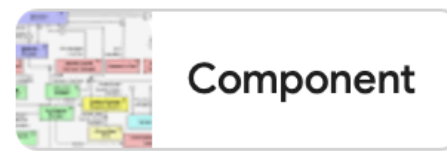
Design



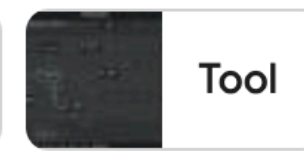
Simple



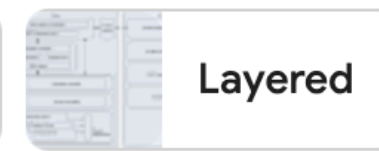
Azure



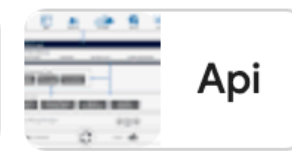
Component



Tool



Layered



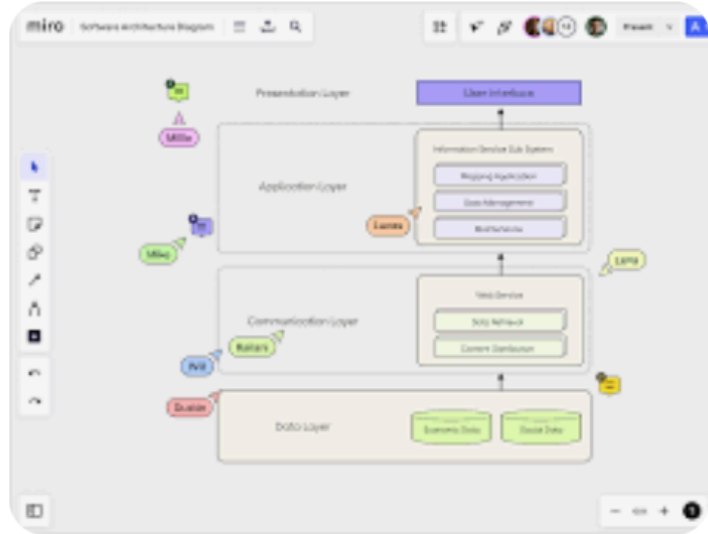
Api



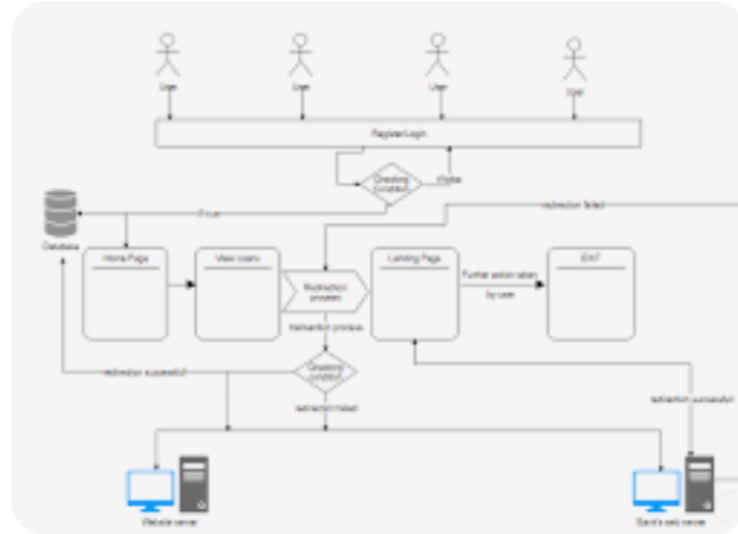
Game



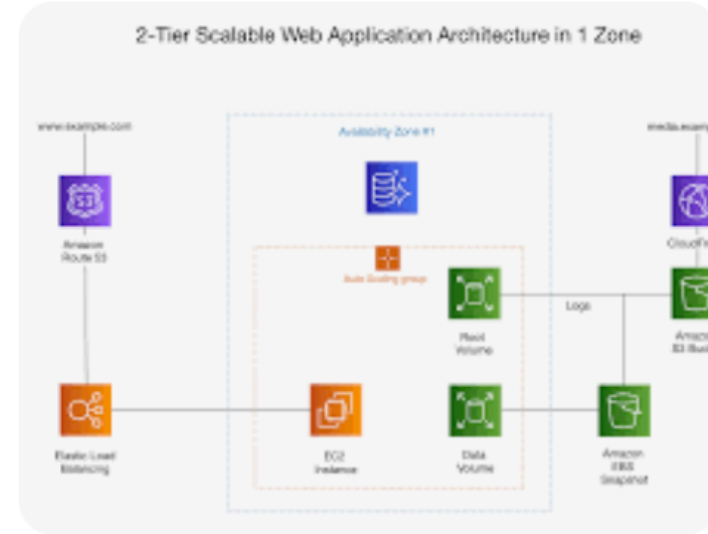
Ec



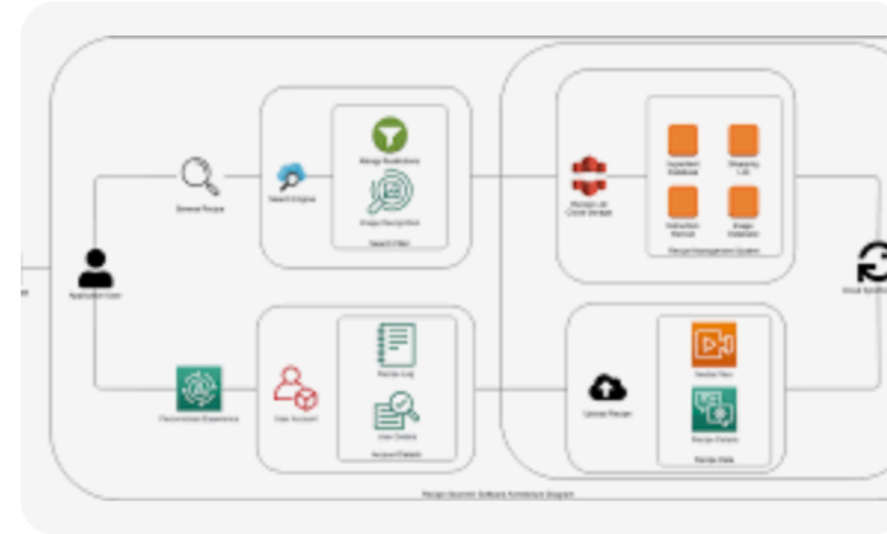
Software Architecture Diagramming



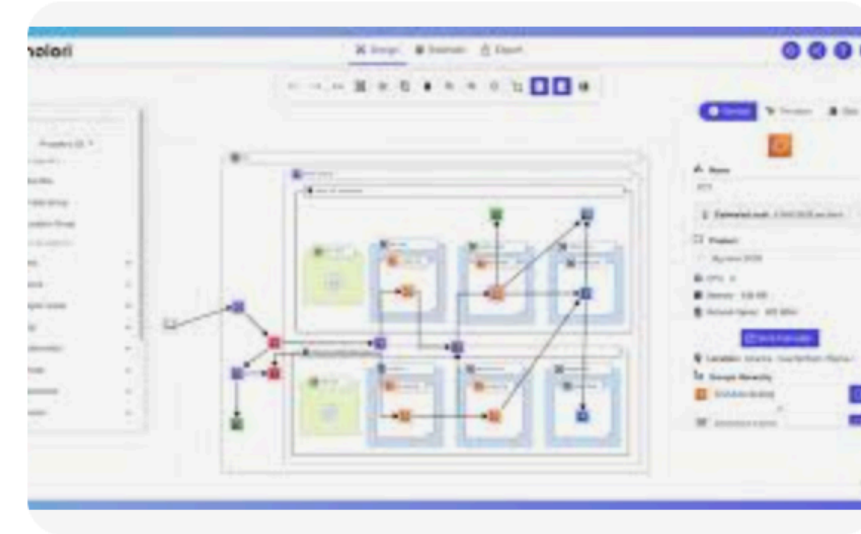
Software Architecture Diagram | Ed...



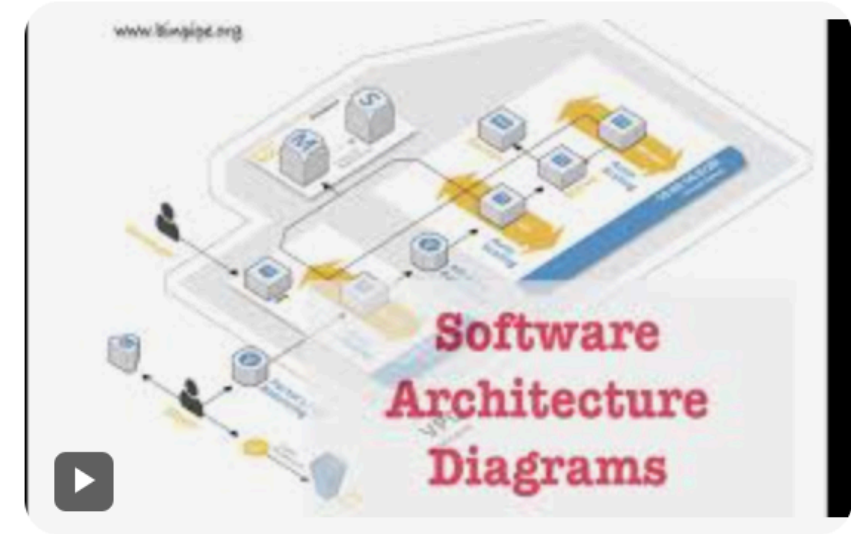
What is an architecture diagram, ...



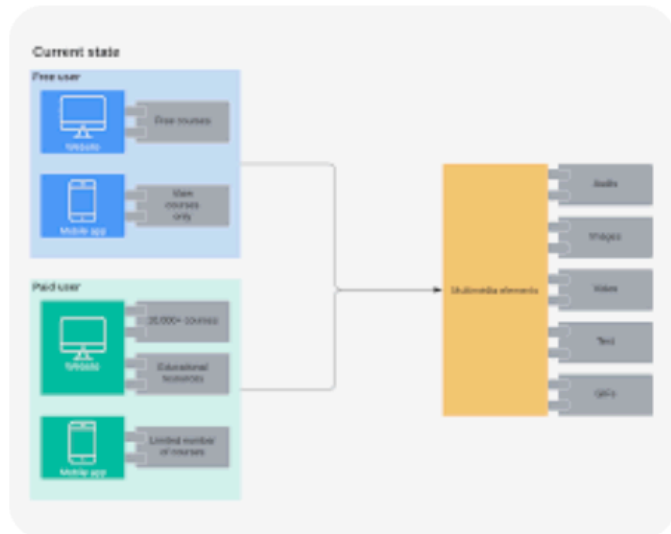
Software Architecture Diagram | Visual ...



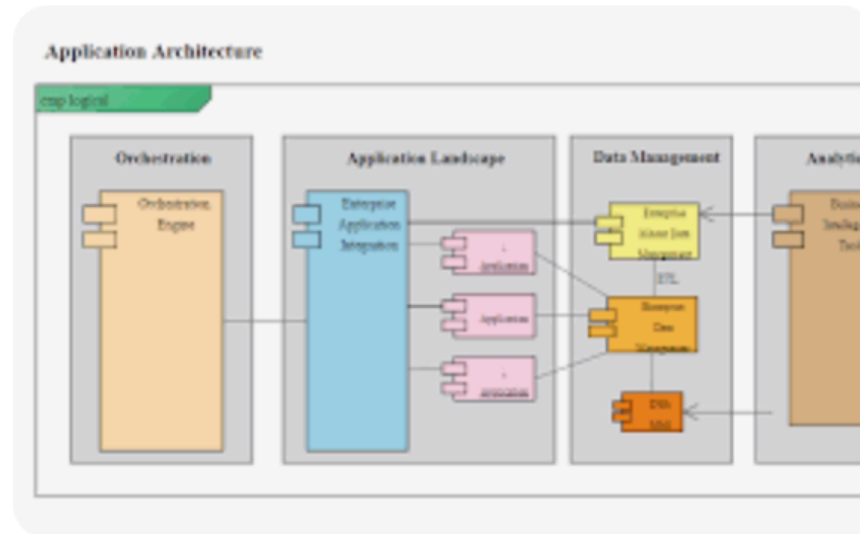
Top 9 Architecture diagram software for ...



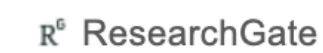
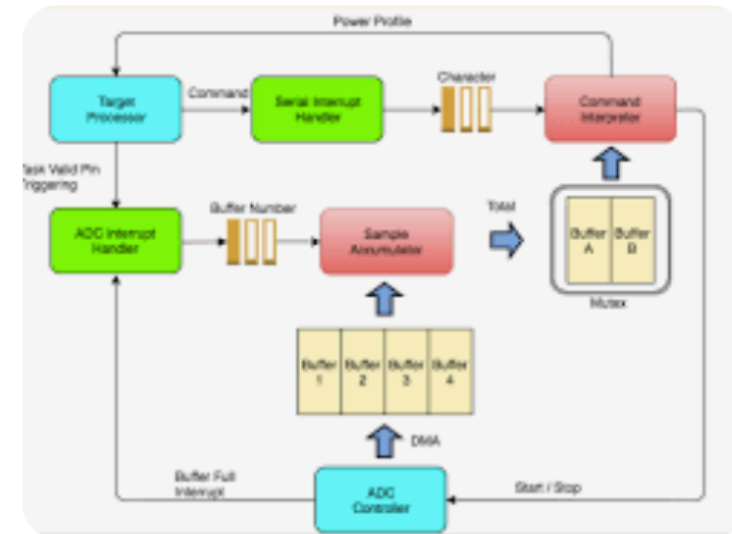
Create Software Architecture Diagrams ...



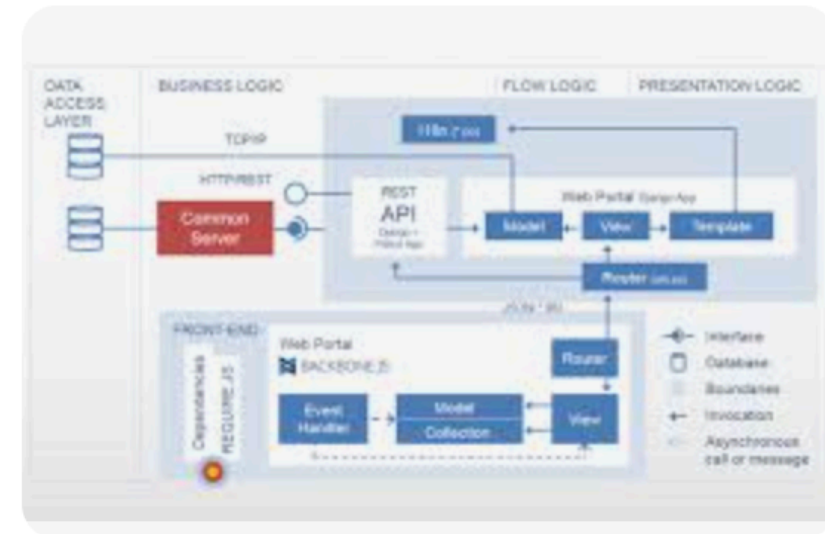
Draw 5 Types of Architectural D...



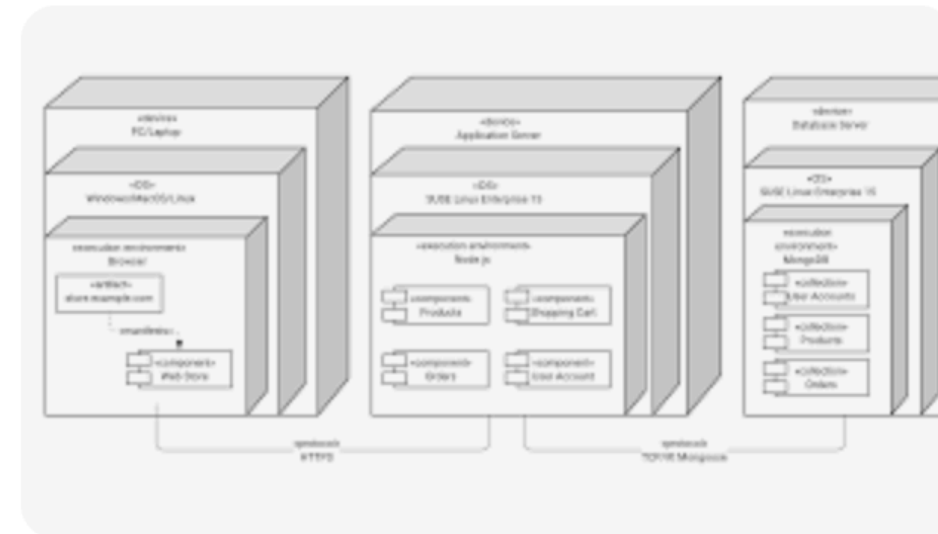
Application Architecture Diagram: A ...



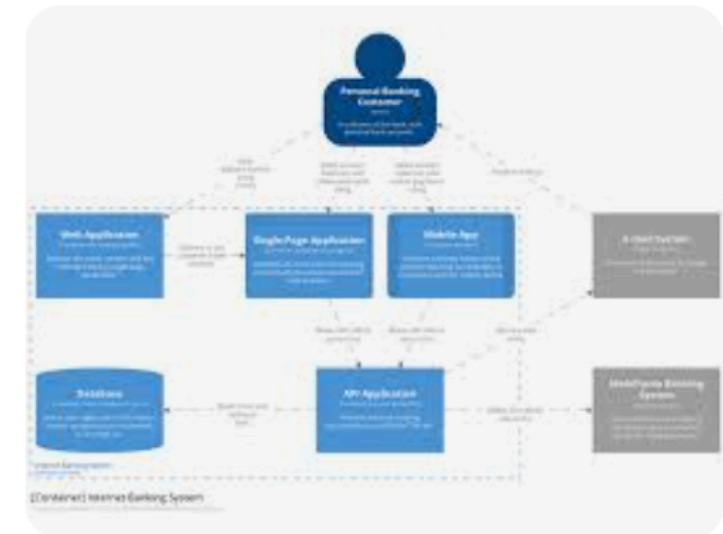
Instrumentation Software Architect...



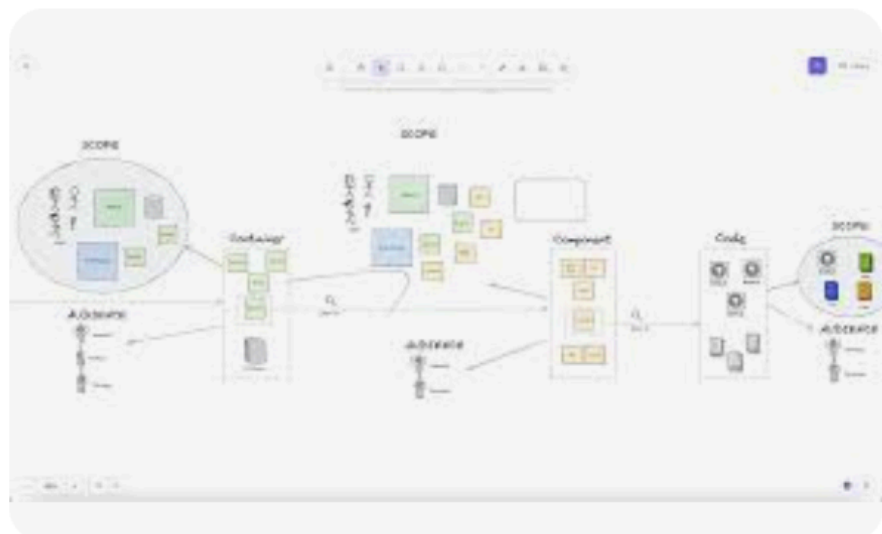
Four Layers Modern Web Application ...



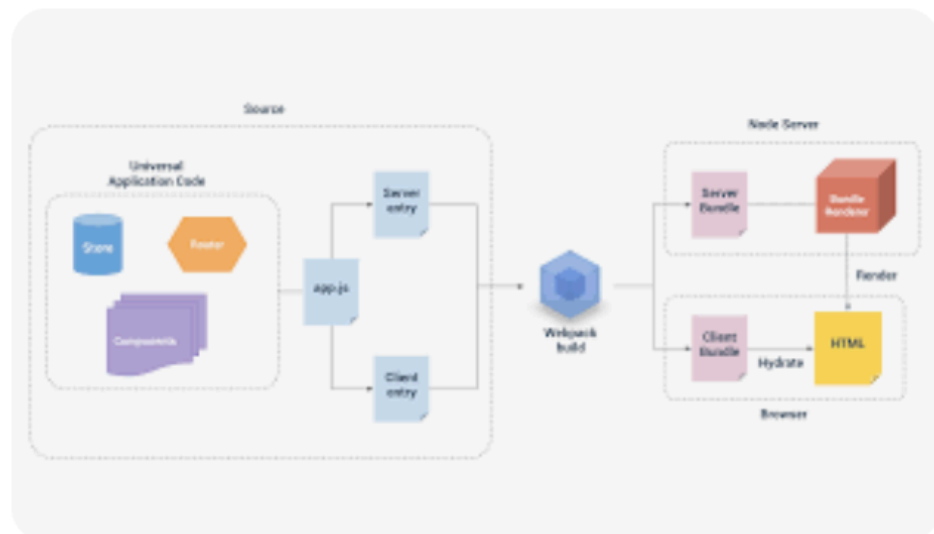
Draw 5 Types of Architectural Diagrams ...



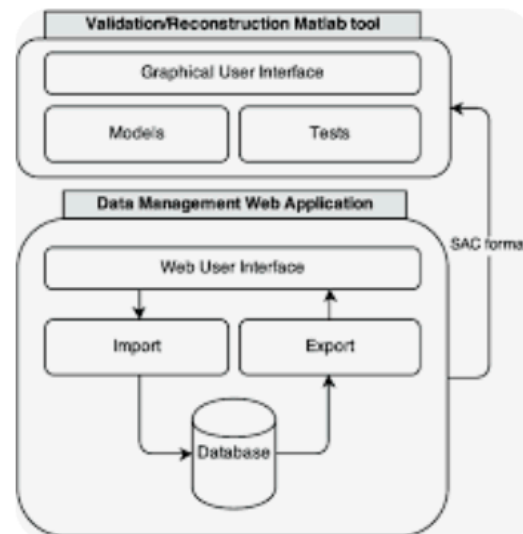
5 great diagramming tools for ...



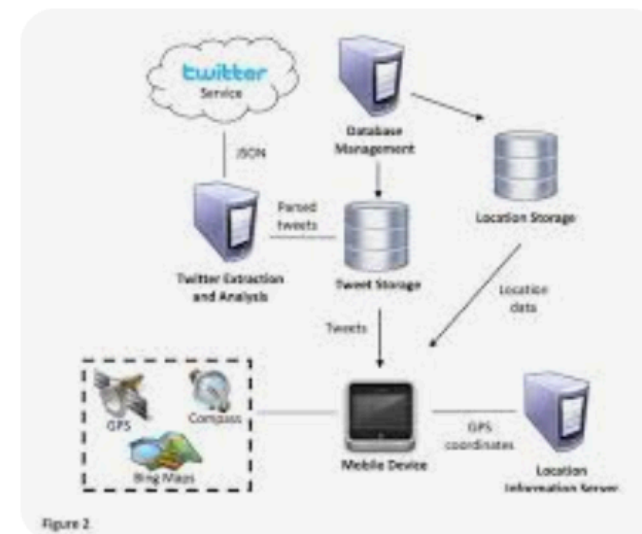
Top 8 diagramming tools for software ...



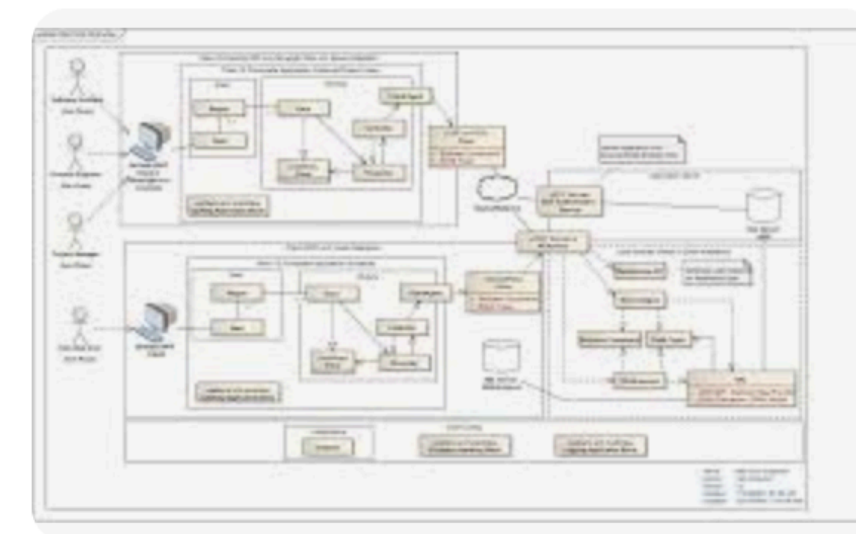
creating software architecture diagram ...



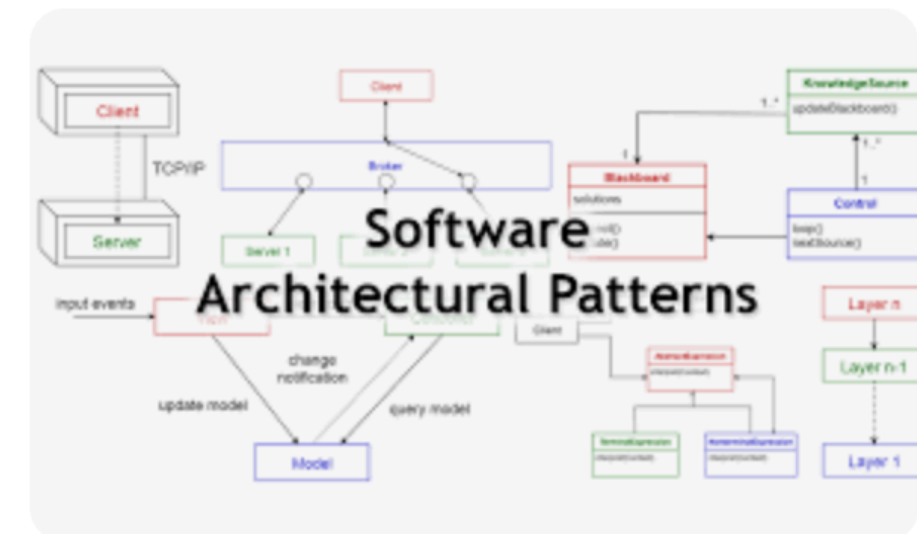
Software architecture di...



tools for architectural diagram ...



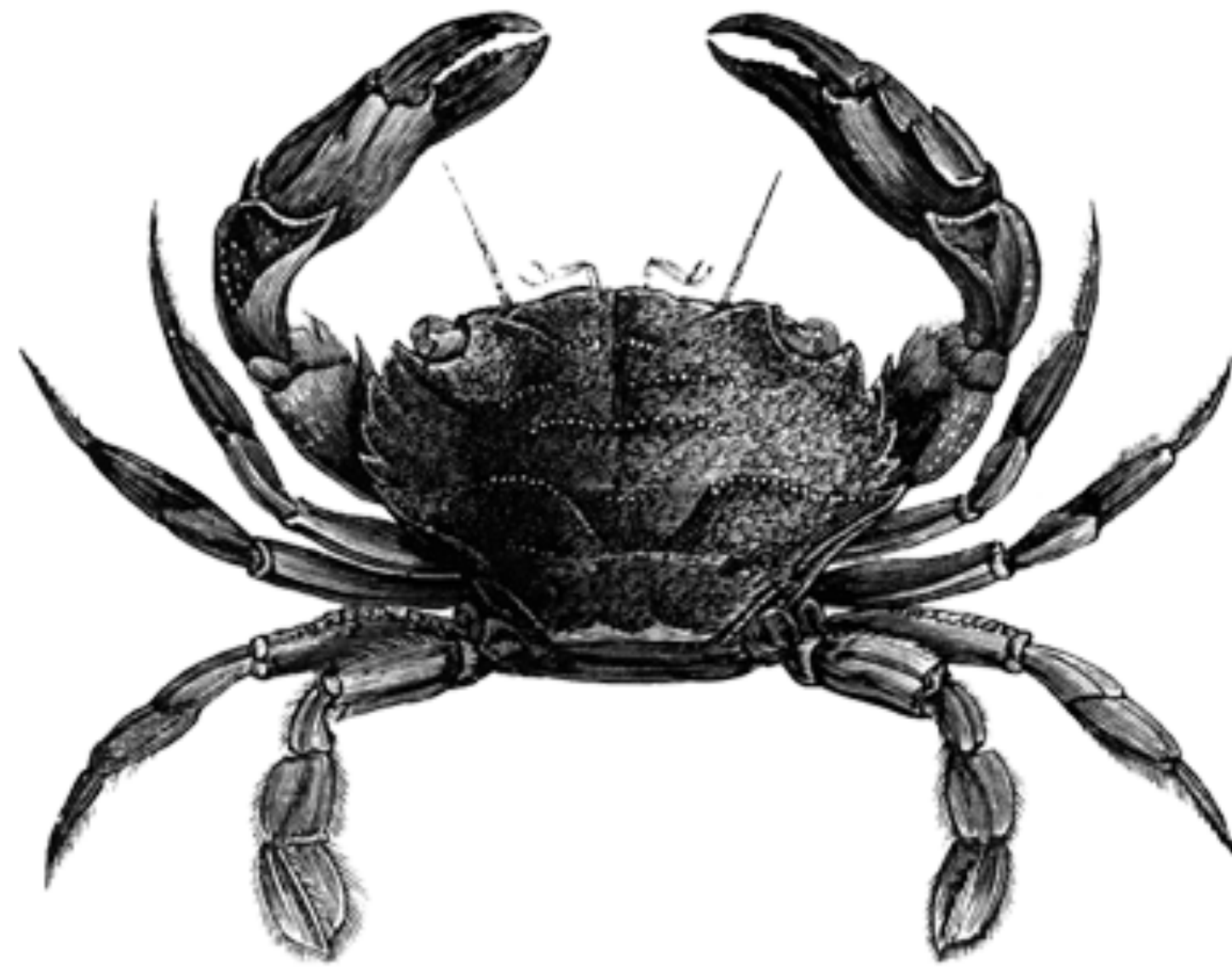
What is Software Architecture



Software architecture diagramming and ...



In my experience,
few people use UML



97 Ways to Sidestep 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."

Our architecture diagrams
don't match the code.

JUST ENOUGH SOFTWARE ARCHITECTURE

A RISK-DRIVEN APPROACH

GEORGE FAIRBANKS

FOREWORD BY DAVID GARLAN



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

Component

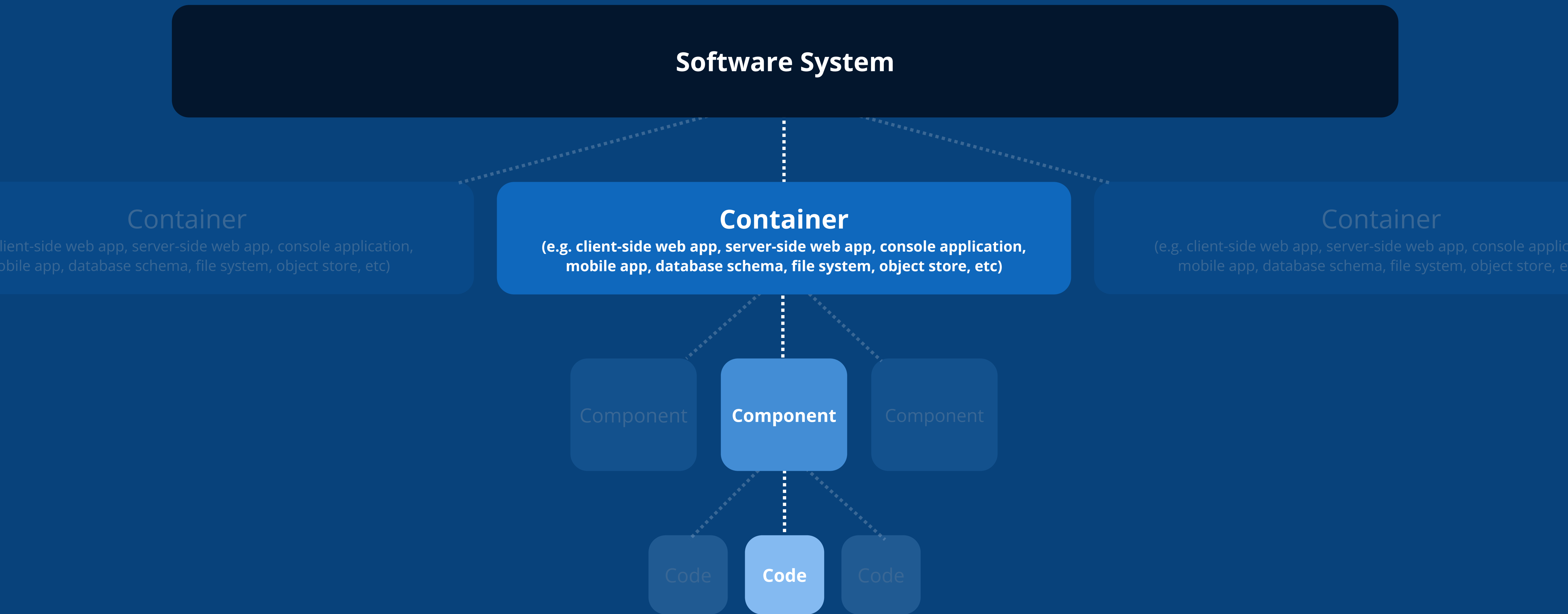
a modular unit with well-defined Interfaces
that is replaceable within its environment

<https://www.omg.org/spec/UML/2.5.1/PDF>

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

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

Abstractions



A **software system** is made up of one or more **containers** (applications and data stores), each of which contains one or more **components**, which in turn are implemented by one or more **code** elements (classes, interfaces, objects, functions, etc).

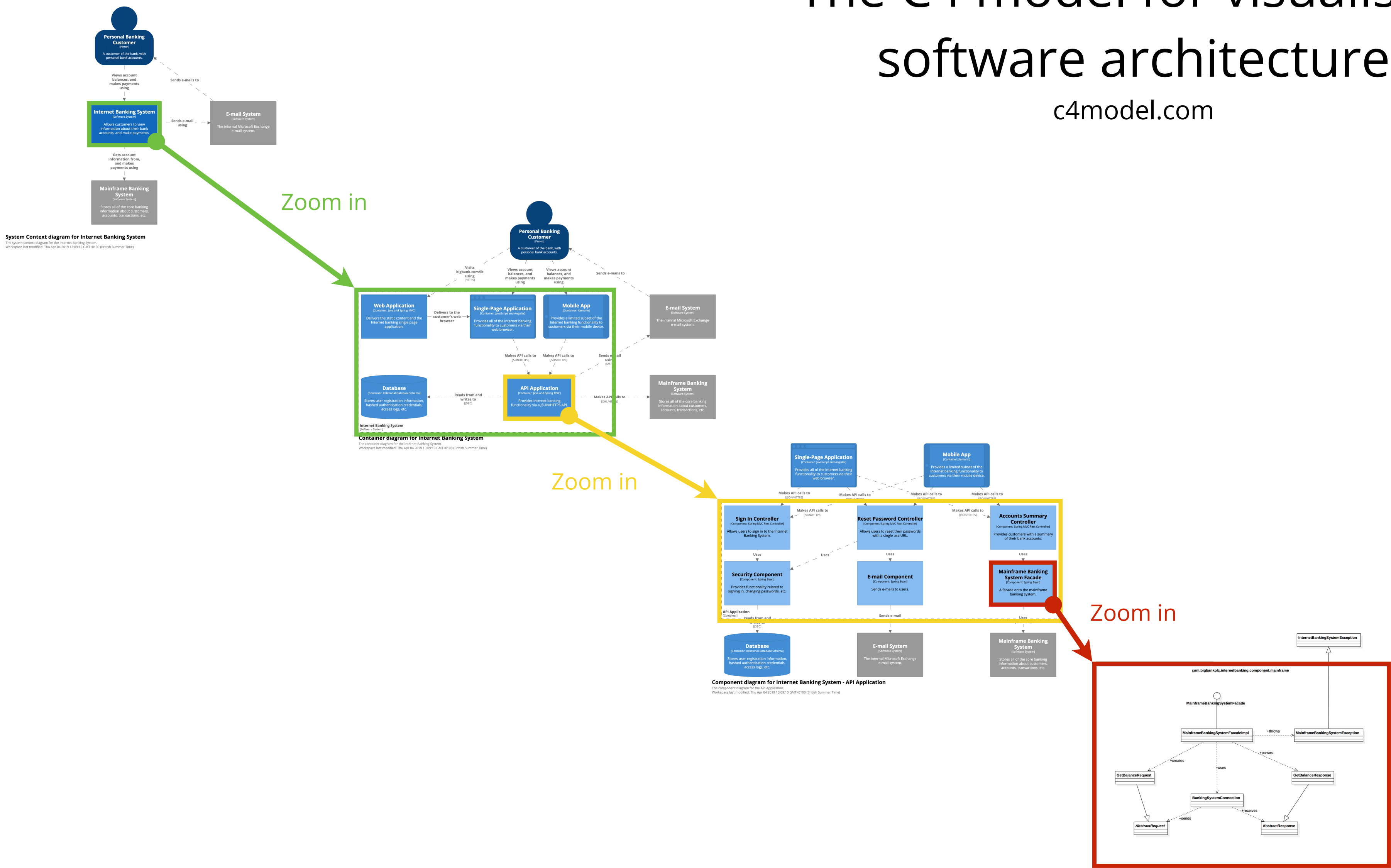
Static structure diagrams

C4

c4model.com

The C4 model for visualising software architecture

c4model.com

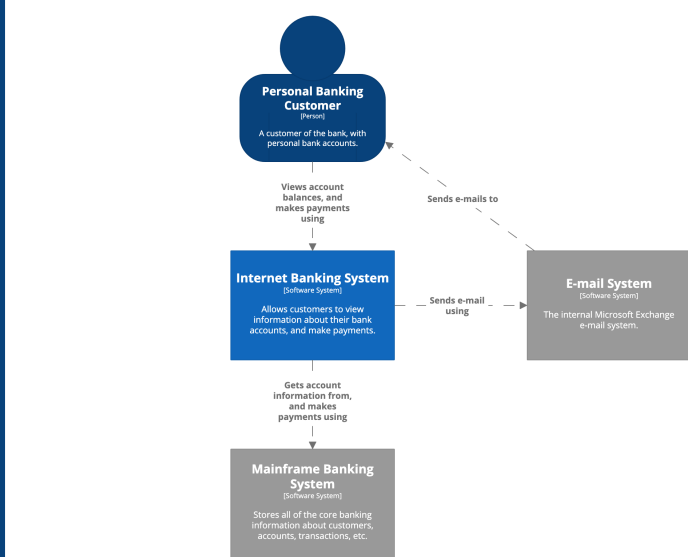
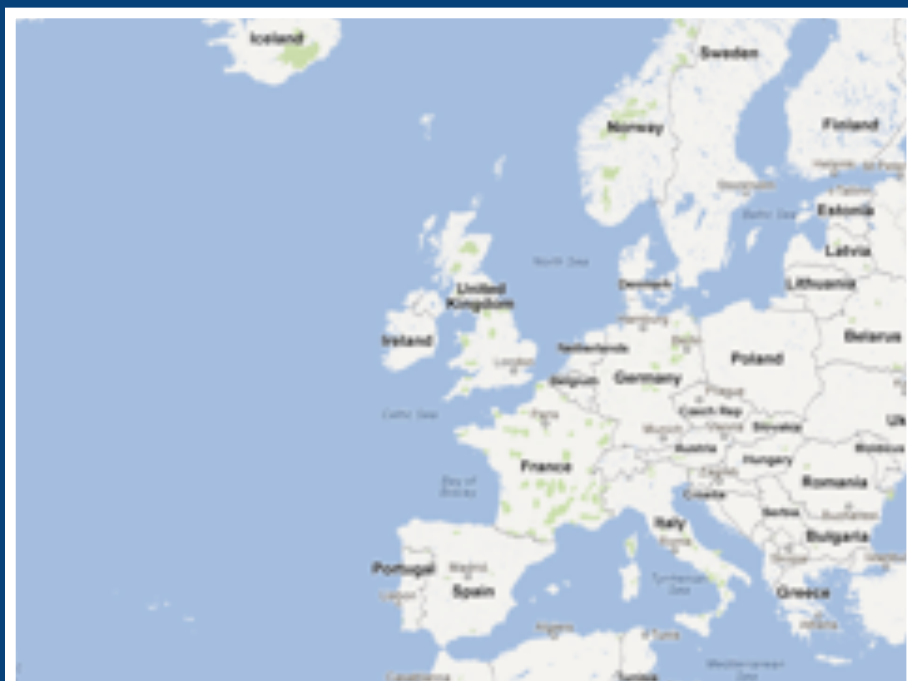


Level 1
Context

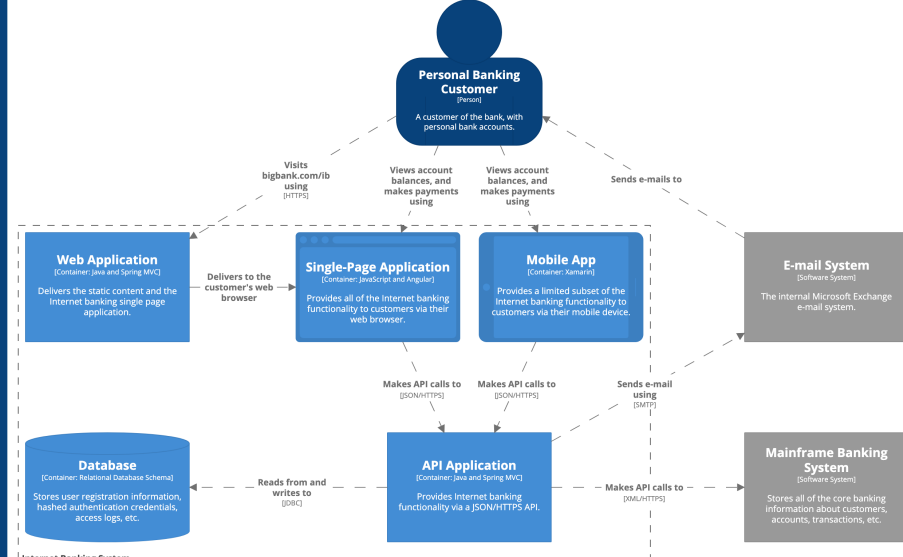
Level 2
Containers

Level 3
Components

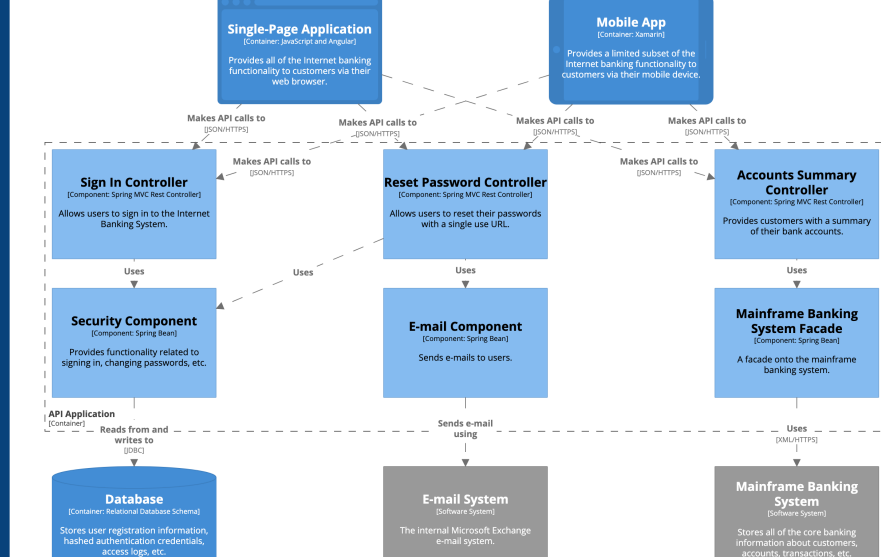
Level 4
Code



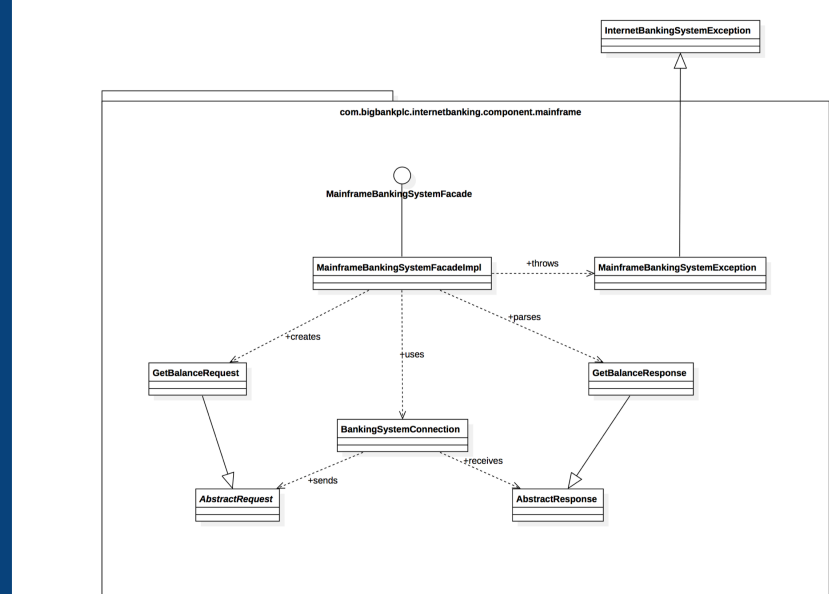
System Context diagram for Internet Banking System
The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:05:10 GMT+0100 (British Summer Time)



Container diagram for Internet Banking System
The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:05:10 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 13:05:10 GMT+0100 (British Summer Time)



Diagrams are maps

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

1. System Context

The system plus users and system dependencies.

2. Containers

The overall shape of the architecture and technology choices.

3. Components

Logical components and their interactions within a container.

4. Code (e.g. classes)

Component implementation details.

Overview first

Zoom & filter

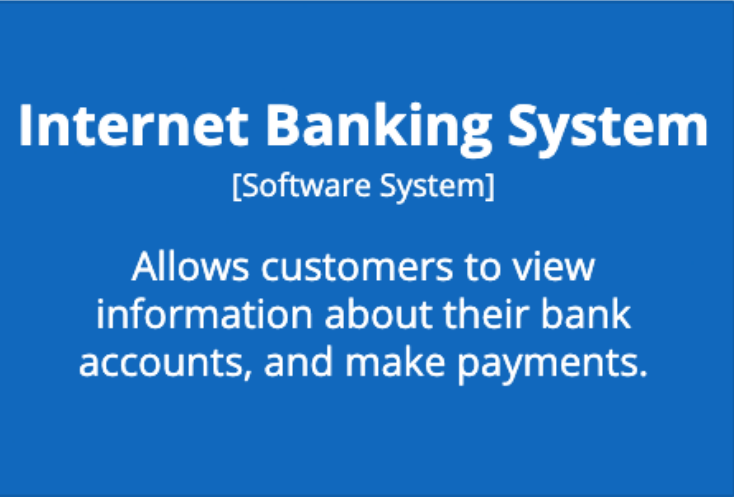
Details on demand

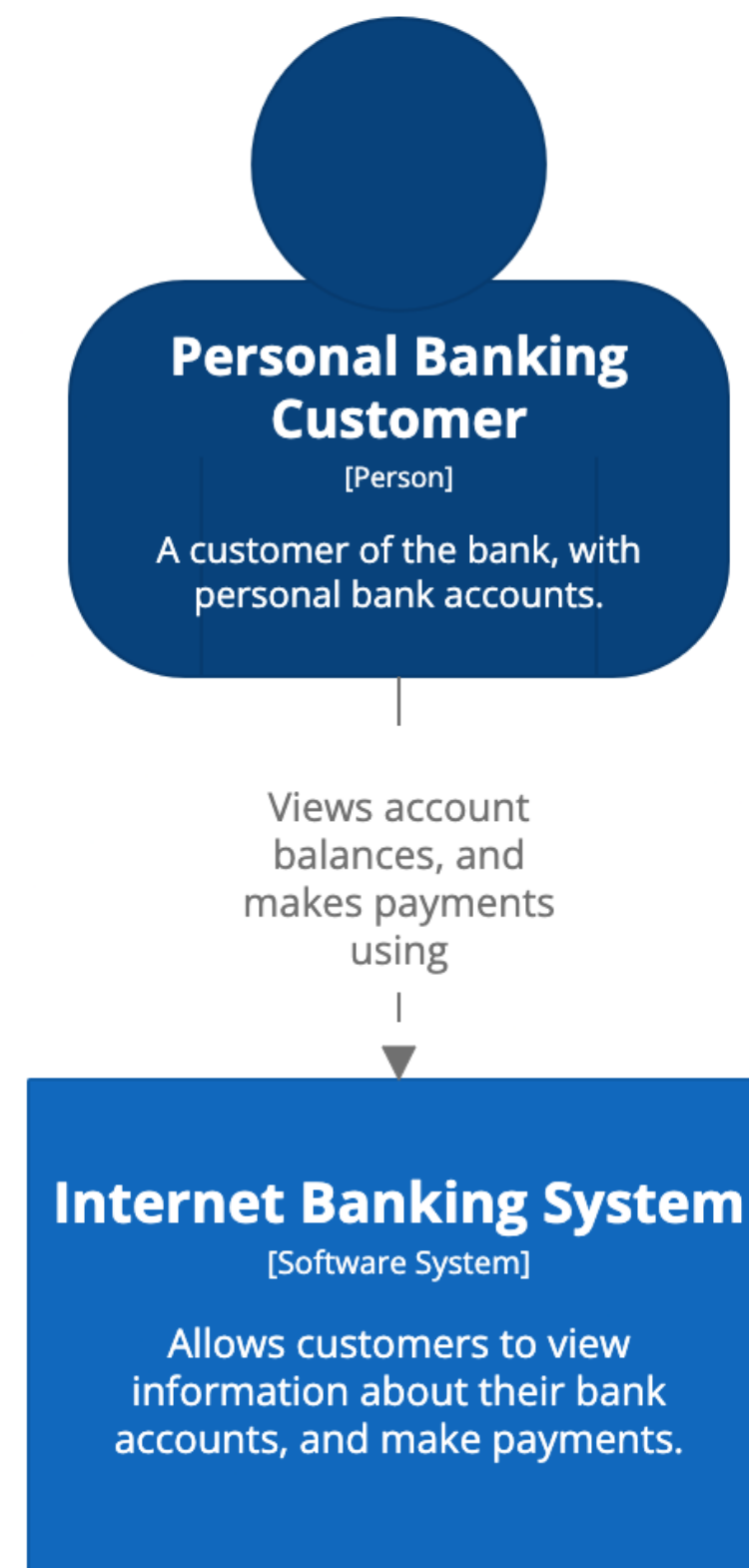
Example

(Internet Banking System)

Level 1

System Context diagram

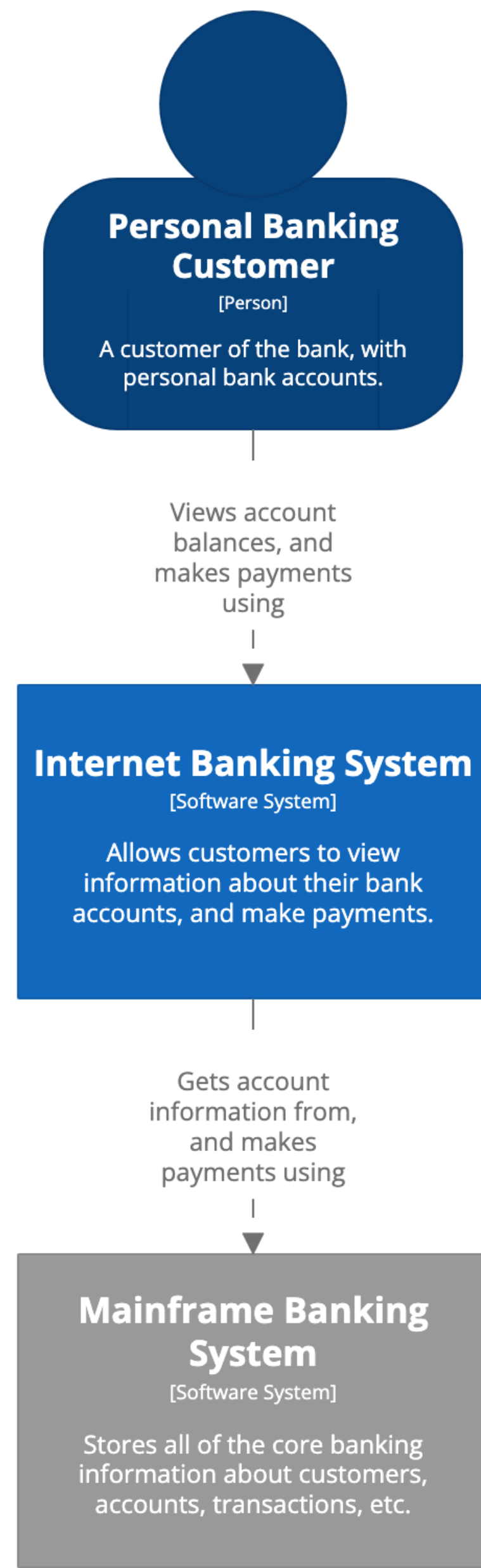




[System Context] Internet Banking System

The system context diagram for the Internet Banking System.

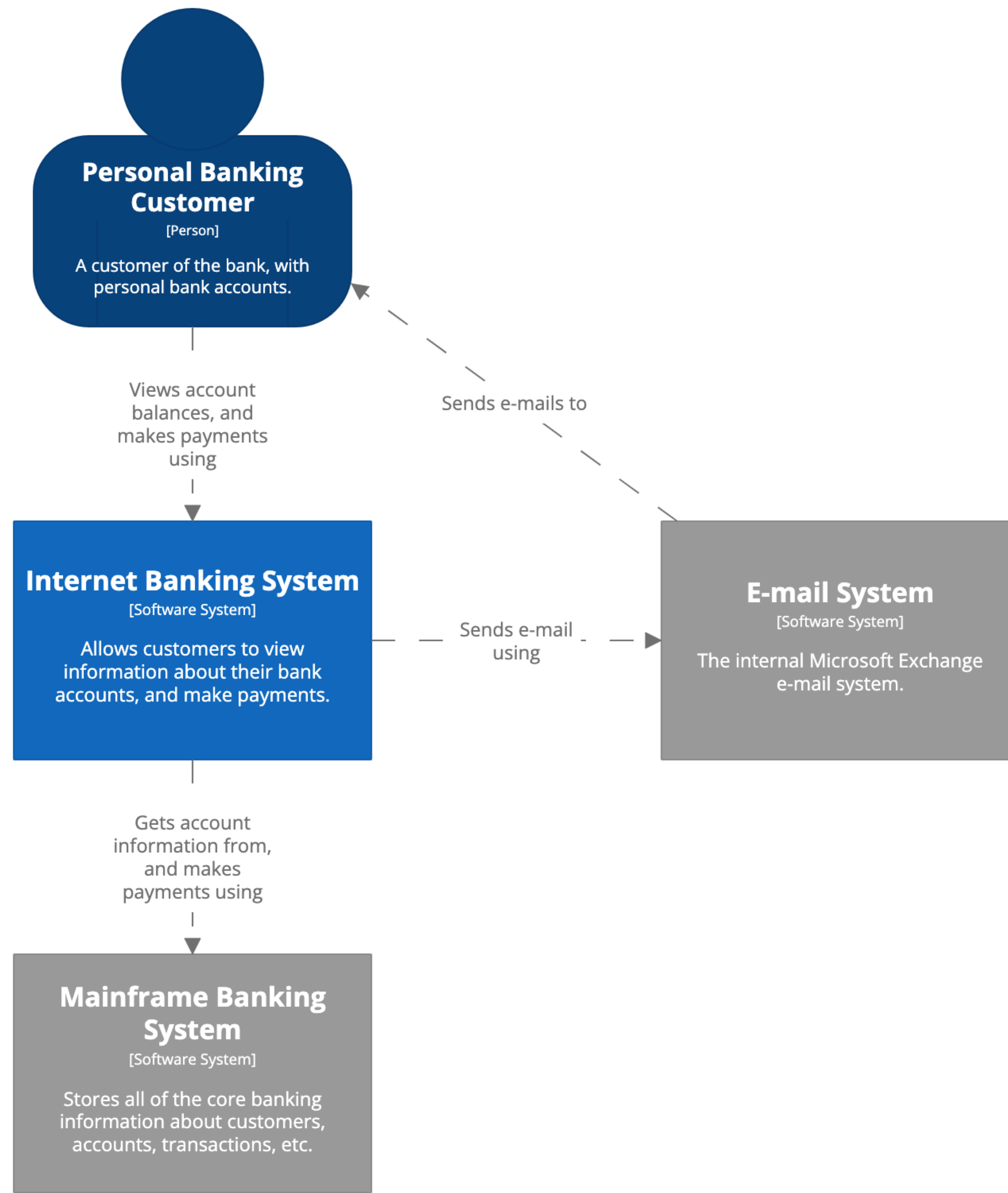
Monday, 27 February 2023 at 15:25 Greenwich Mean Time



[System Context] Internet Banking System

The system context diagram for the Internet Banking System.

Monday, 27 February 2023 at 15:25 Greenwich Mean Time



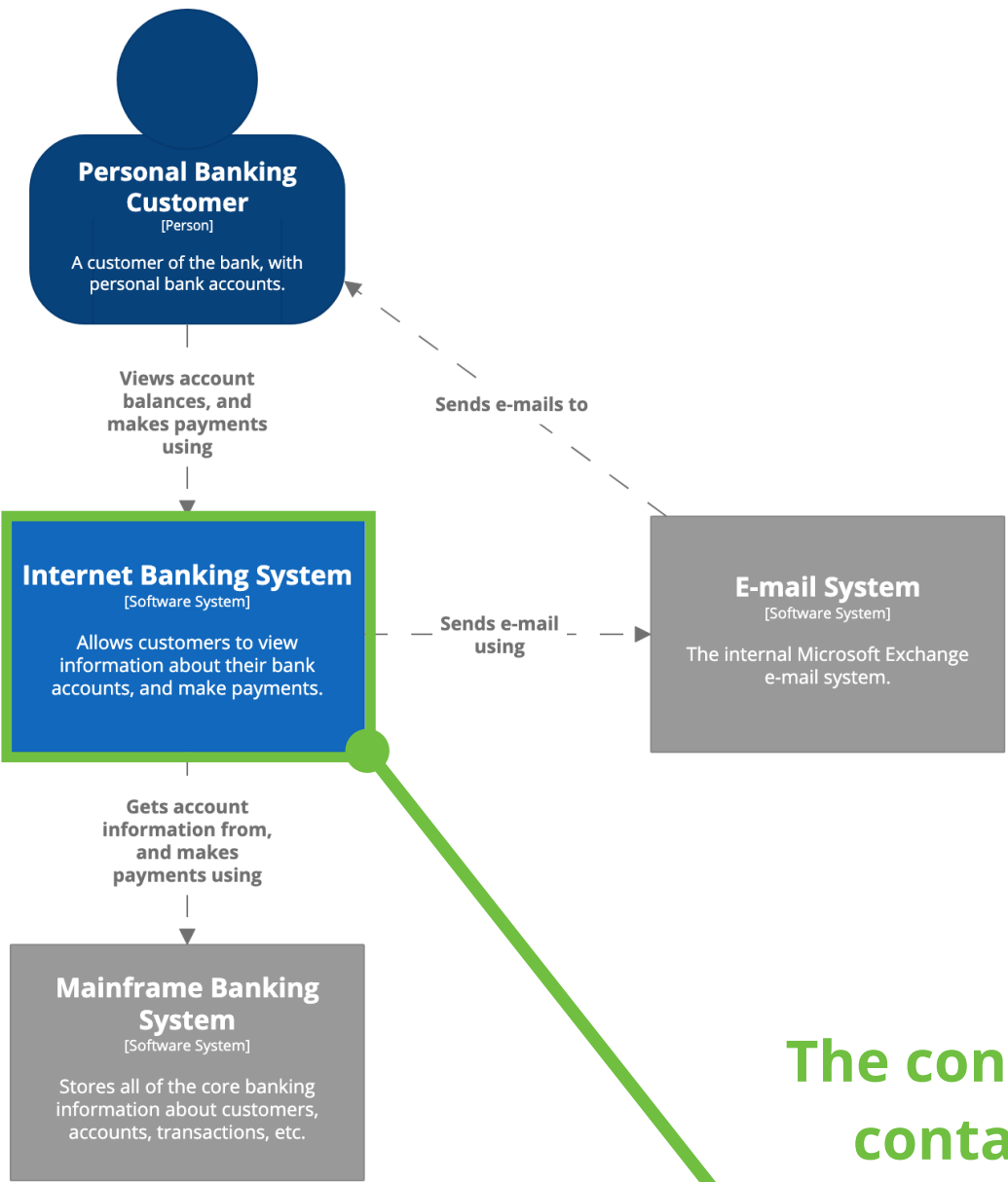
[System Context] Internet Banking System

The system context diagram for the Internet Banking System.

Monday, 27 February 2023 at 15:25 Greenwich Mean Time

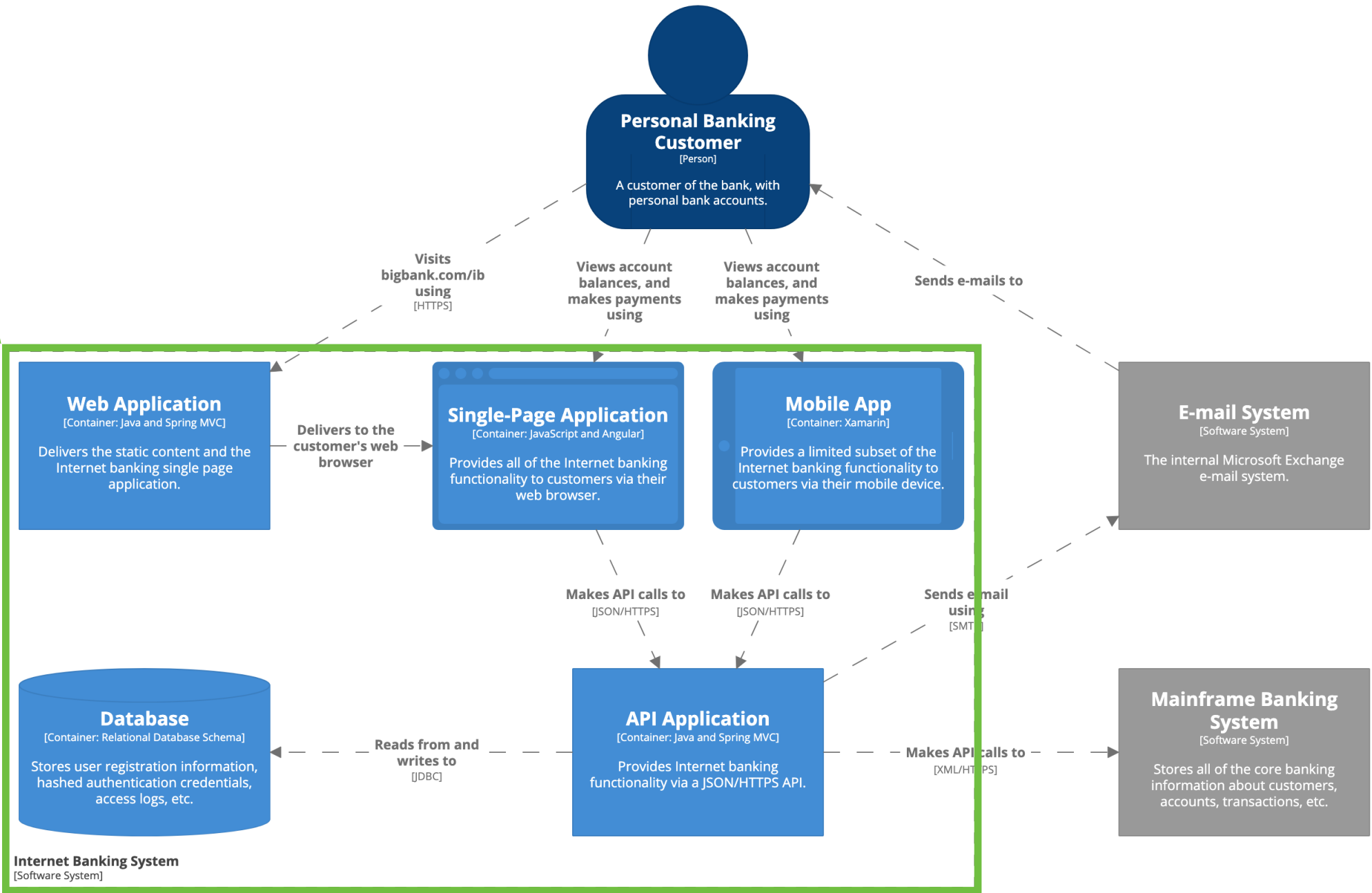
Level 2

Container diagram

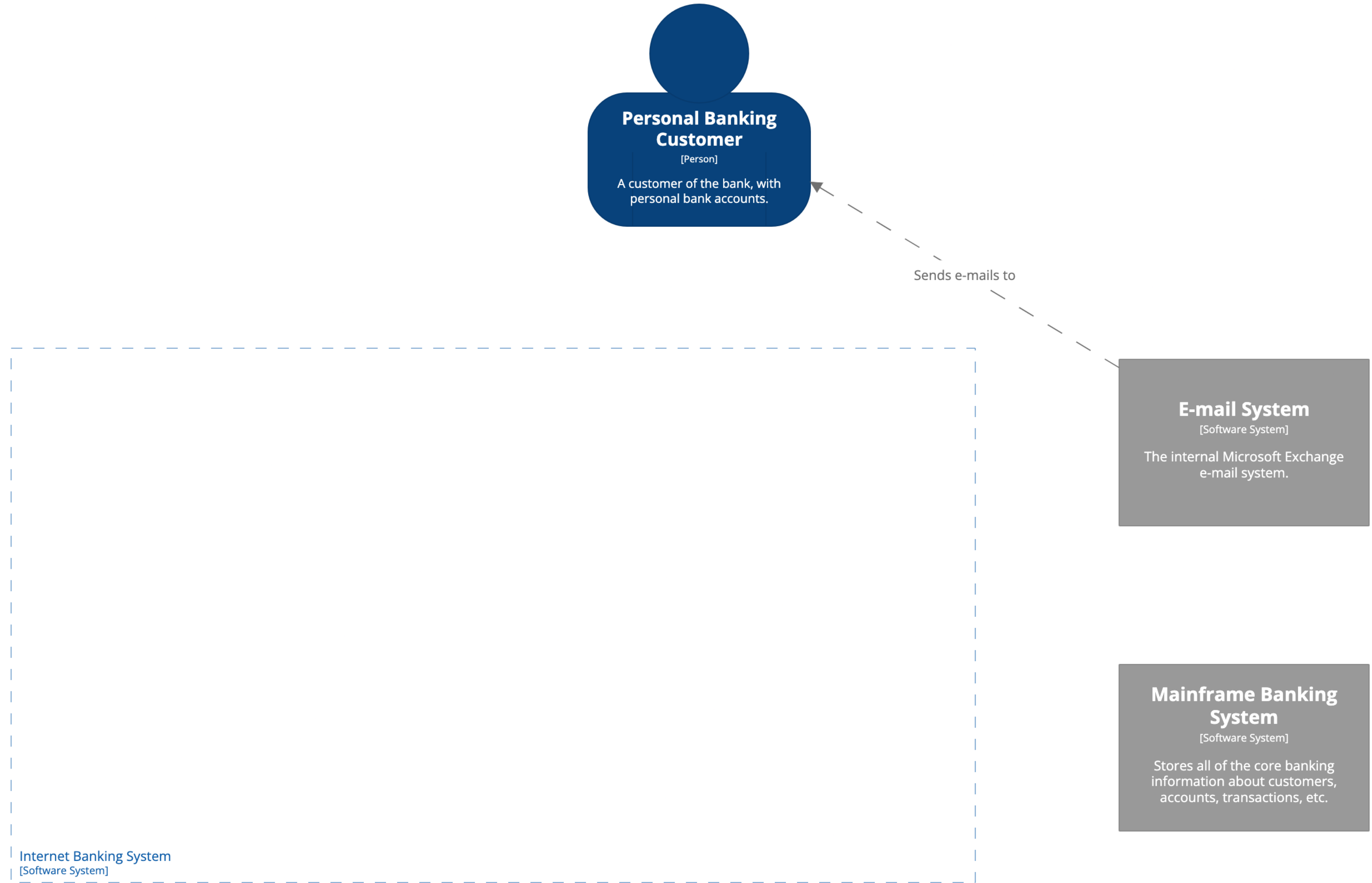


System Context diagram for Internet Banking System
The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

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



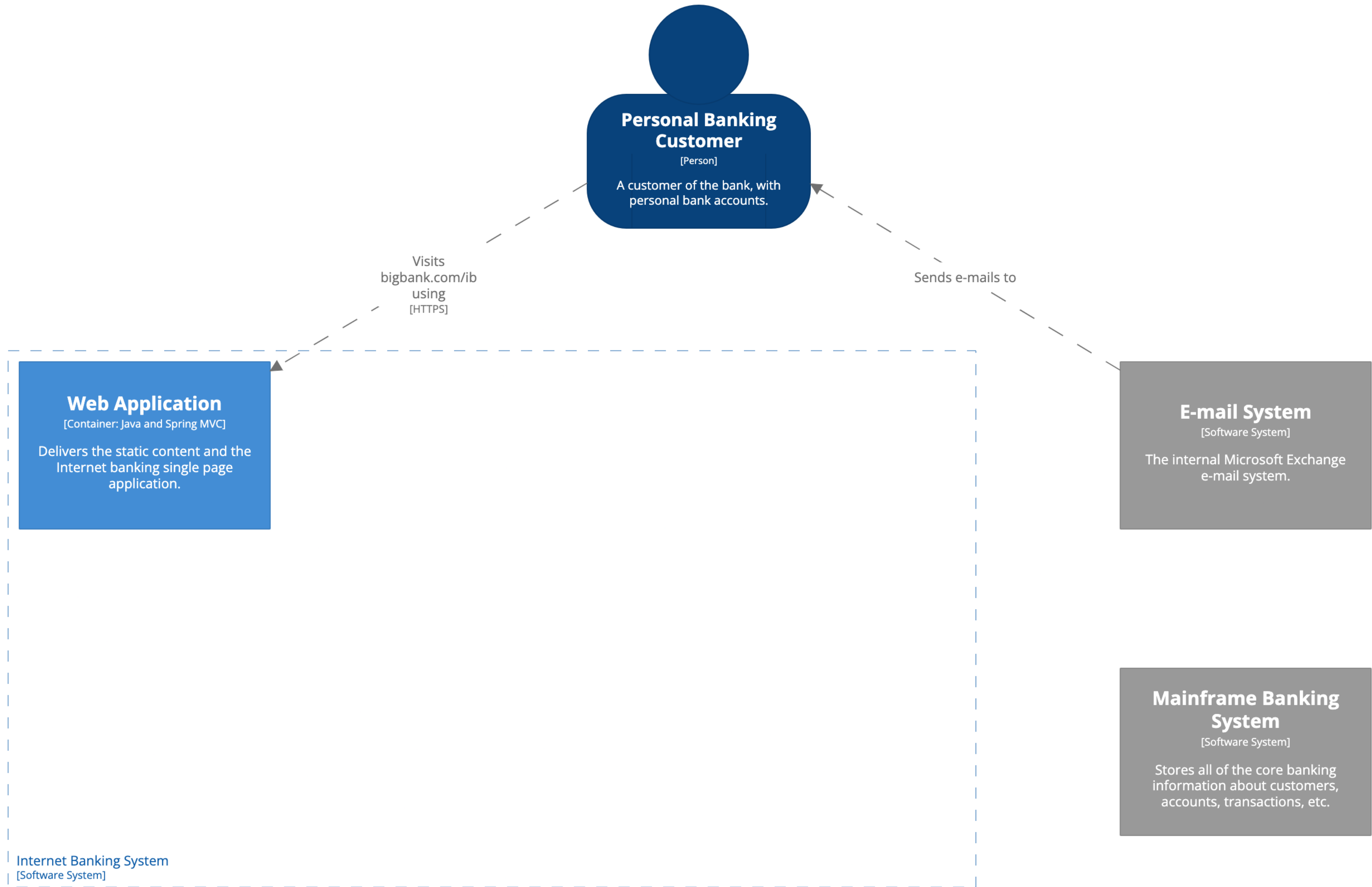
Container diagram for Internet Banking System
The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



[Container] Internet Banking System

The container diagram for the Internet Banking System.

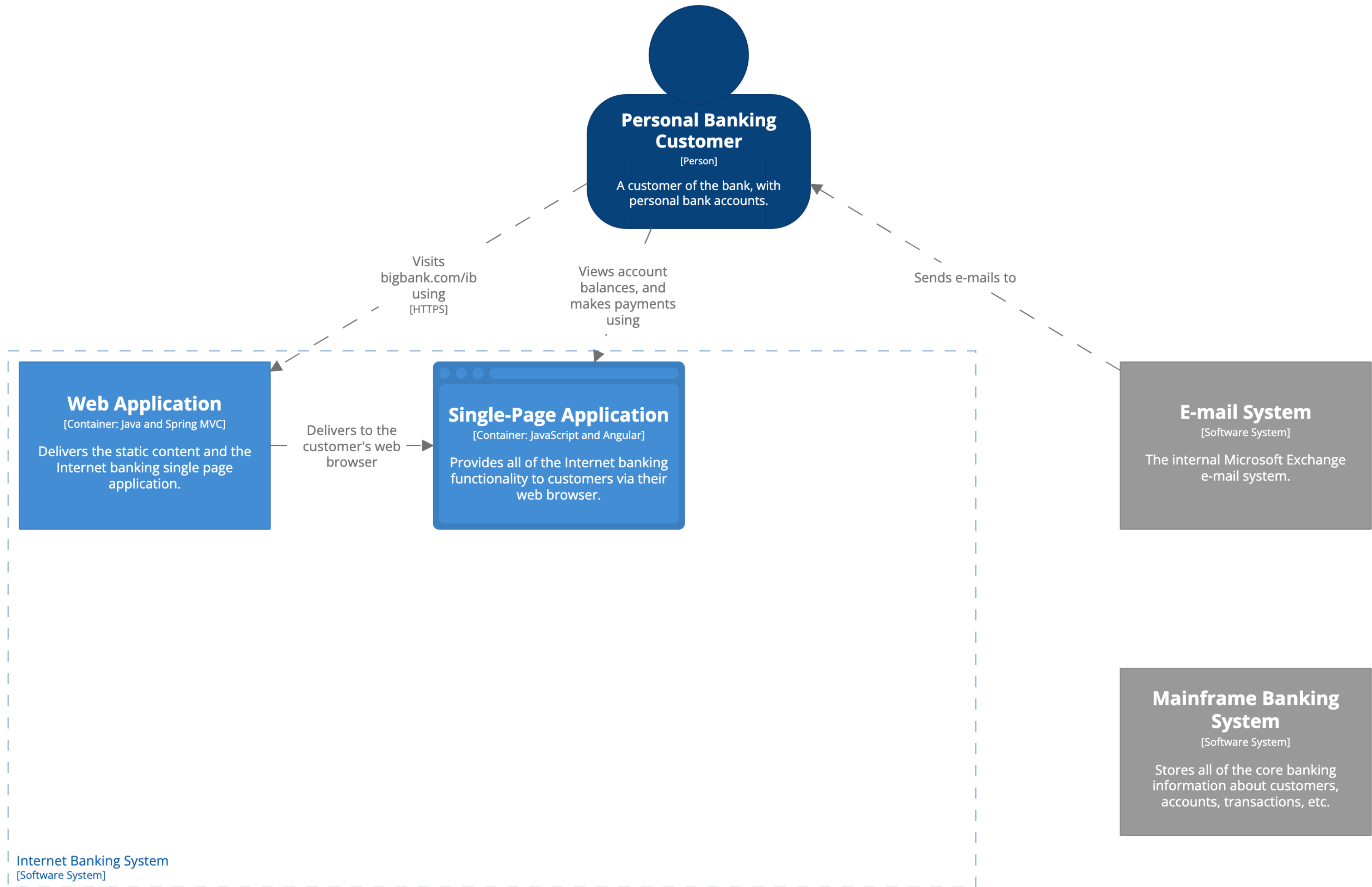
Monday, 27 February 2023 at 15:36 Greenwich Mean Time



[Container] Internet Banking System

The container diagram for the Internet Banking System.

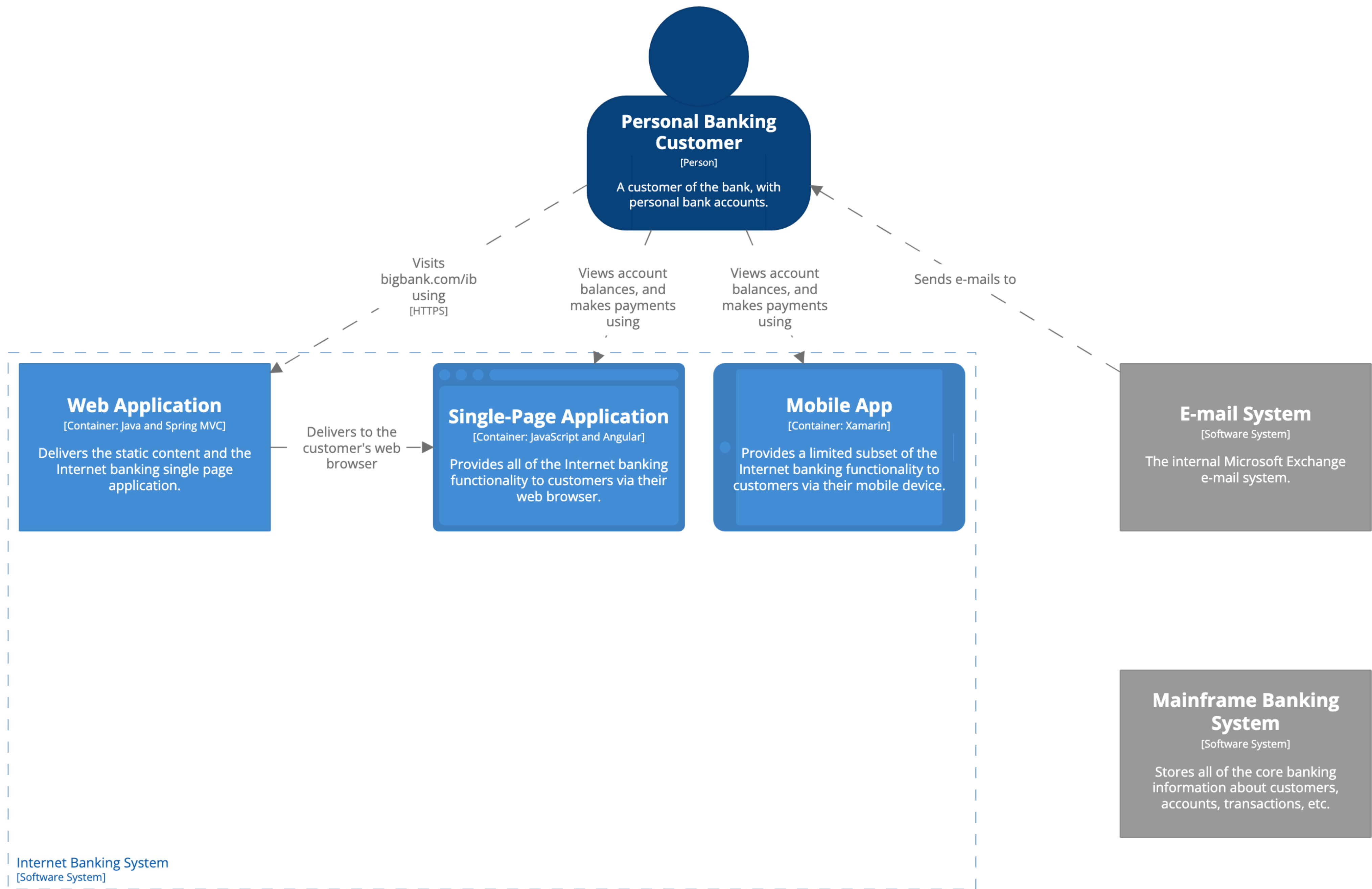
Monday, 27 February 2023 at 15:36 Greenwich Mean Time



[Container] Internet Banking System

The container diagram for the Internet Banking System.

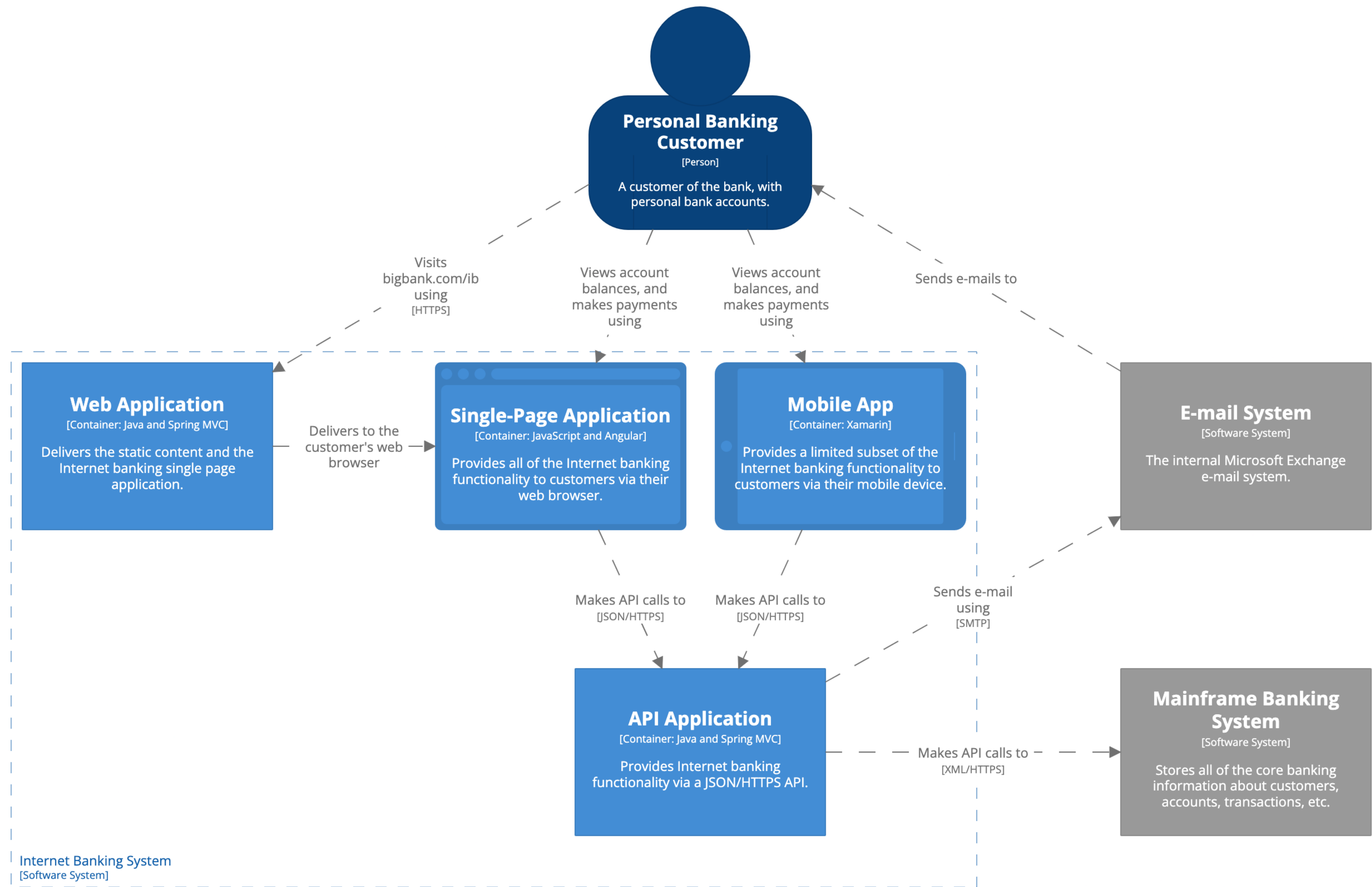
Monday, 27 February 2023 at 15:36 Greenwich Mean Time



[Container] Internet Banking System

The container diagram for the Internet Banking System.

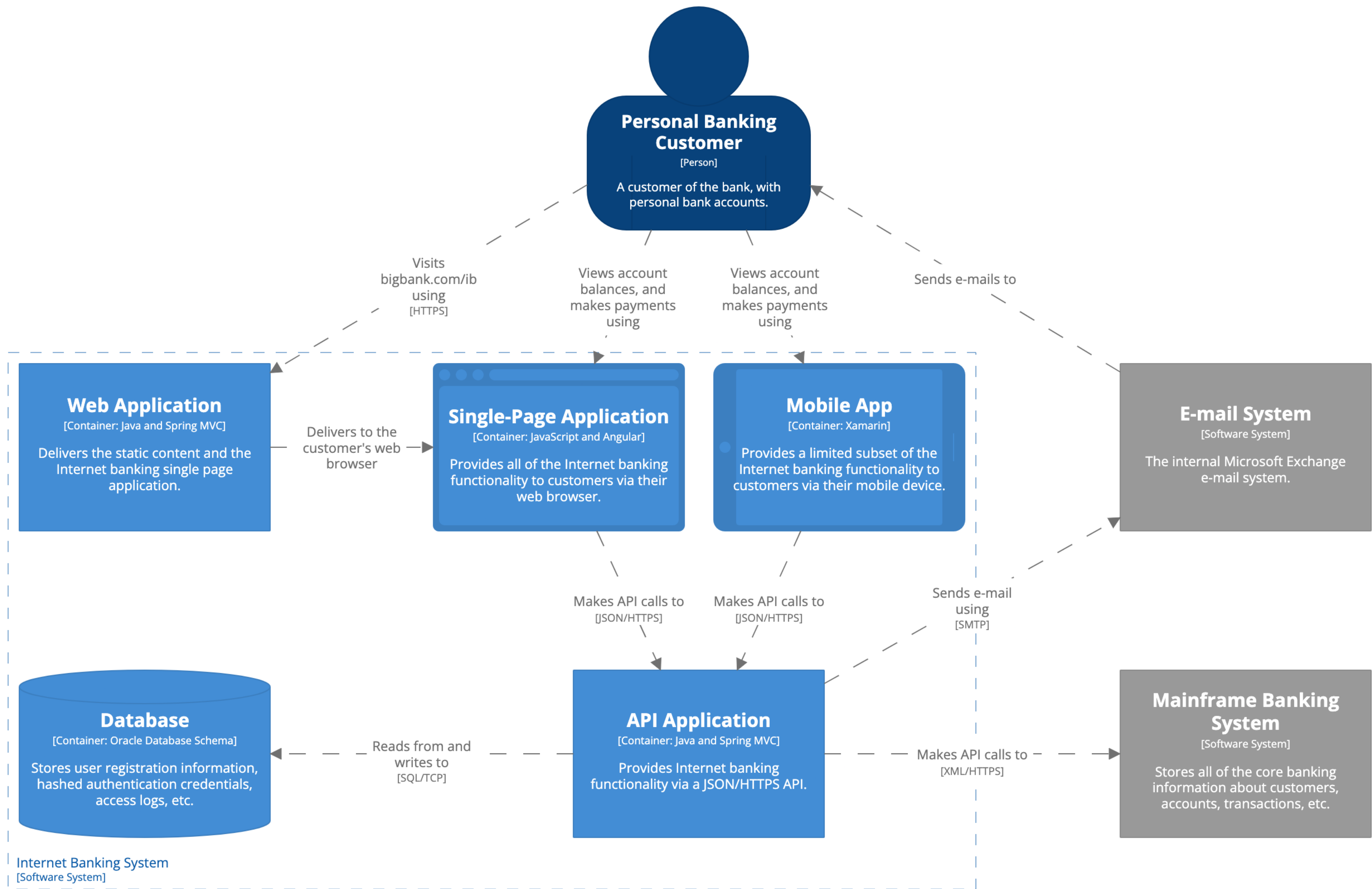
Monday, 27 February 2023 at 15:36 Greenwich Mean Time



[Container] Internet Banking System

The container diagram for the Internet Banking System.

Monday, 27 February 2023 at 15:36 Greenwich Mean Time



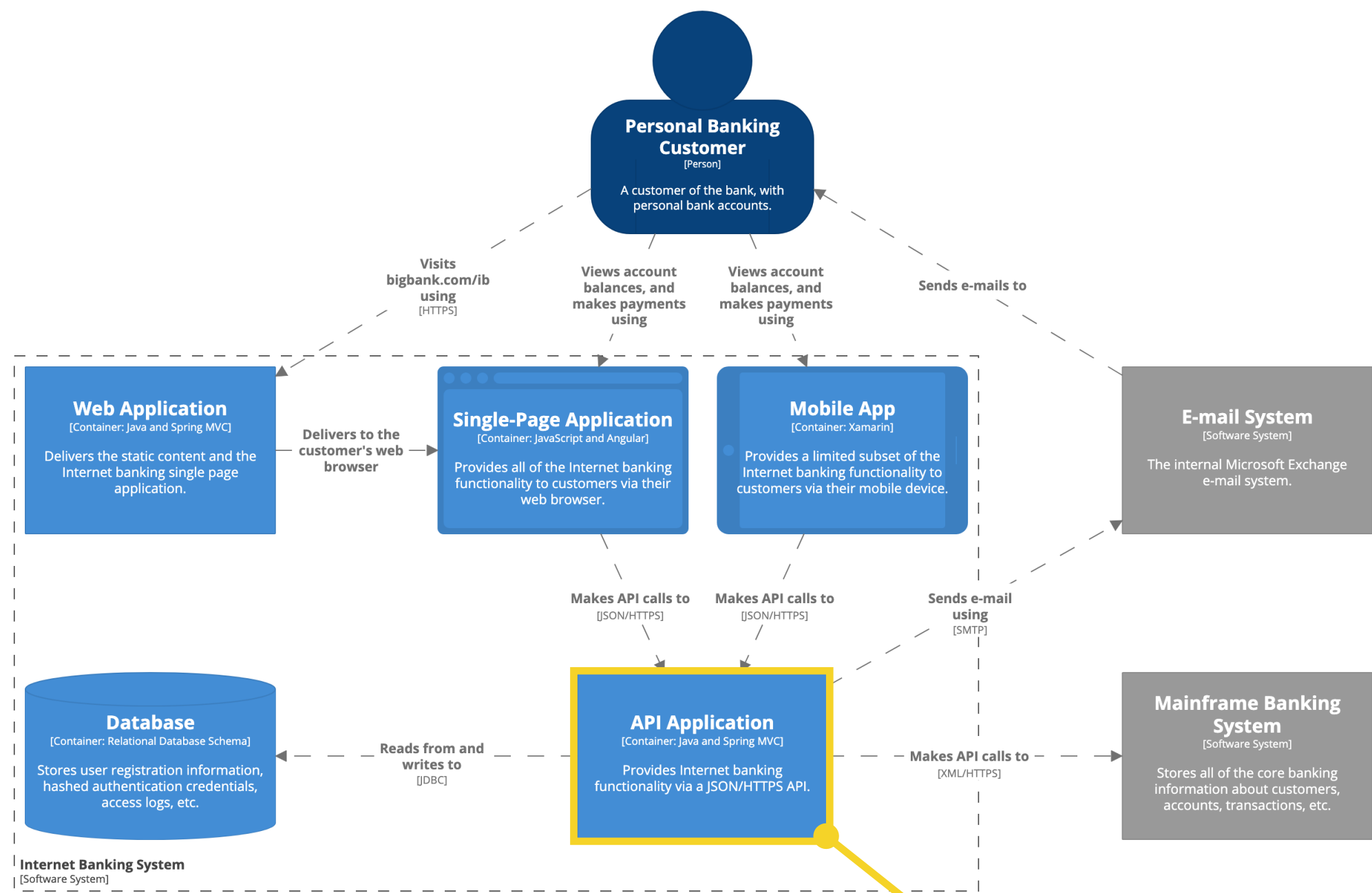
[Container] Internet Banking System

The container diagram for the Internet Banking System.

Monday, 27 February 2023 at 15:36 Greenwich Mean Time

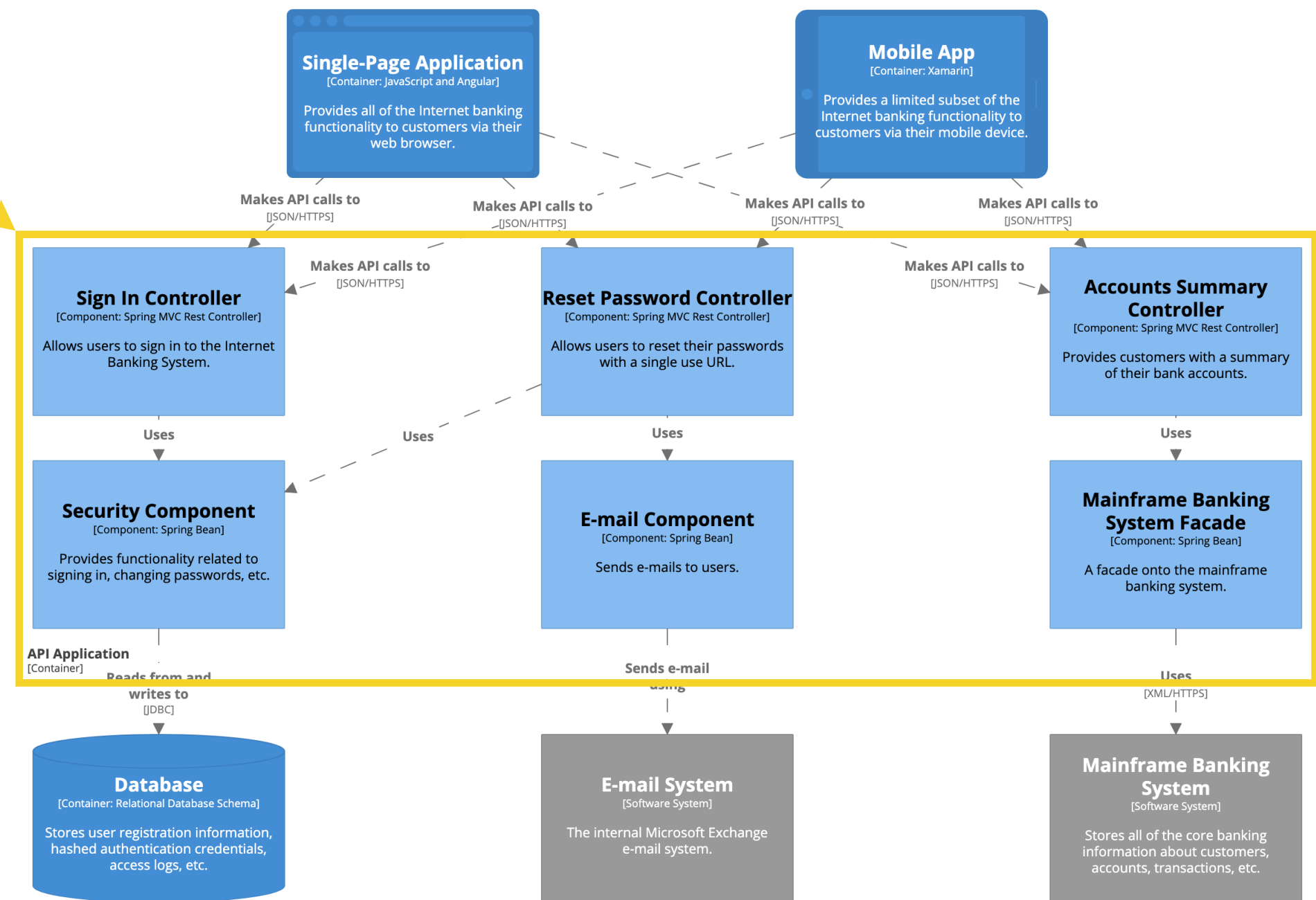
Level 3

Component diagram



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)

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.

API Application
[Container]

Database

[Container: Oracle Database Schema]

Stores user registration information, hashed authentication credentials, access logs, etc.

E-mail System

[Software System]

The internal Microsoft Exchange e-mail system.

Mainframe Banking System

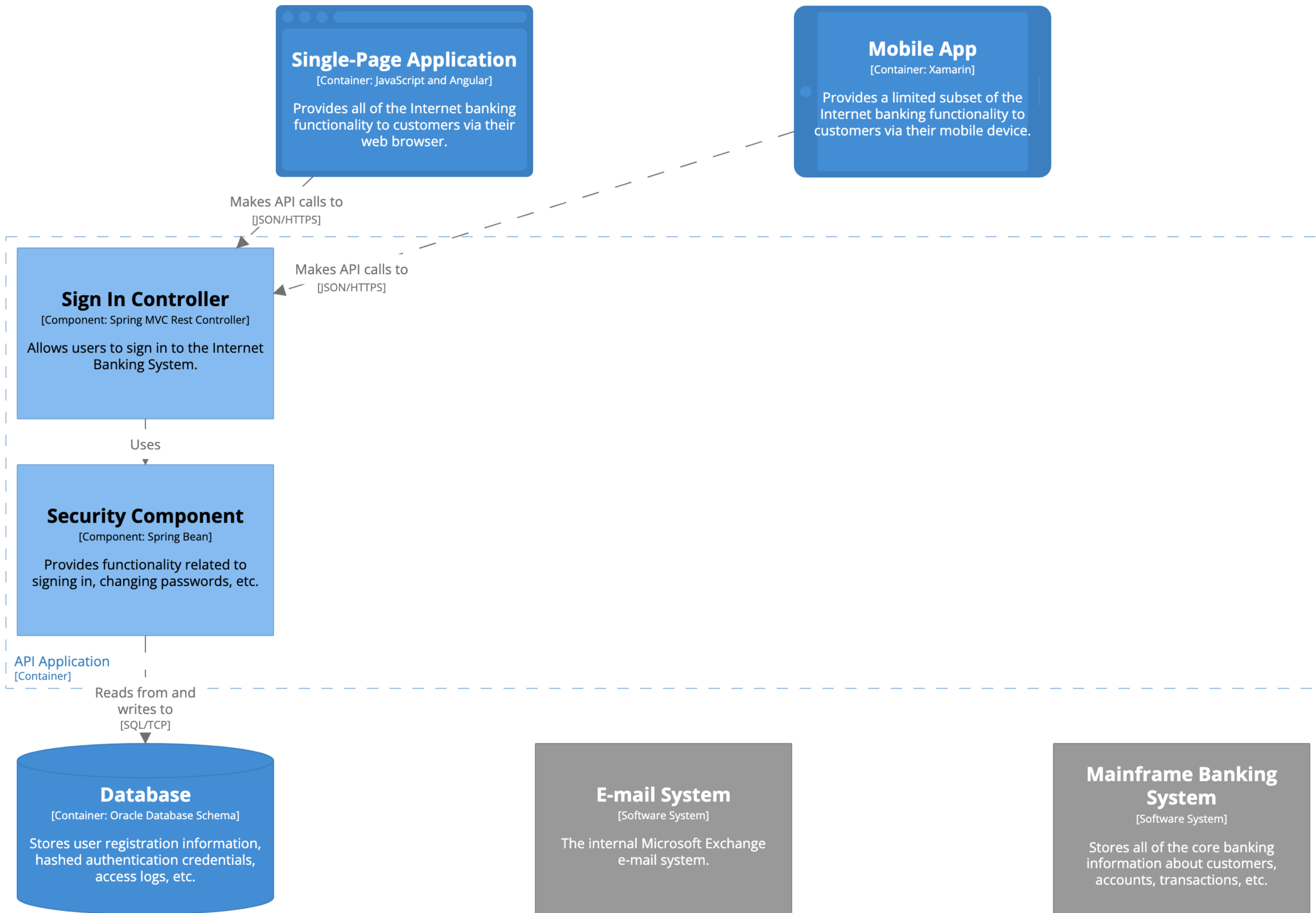
[Software System]

Stores all of the core banking information about customers, accounts, transactions, etc.

[Component] Internet Banking System - API Application

The component diagram for the API Application.

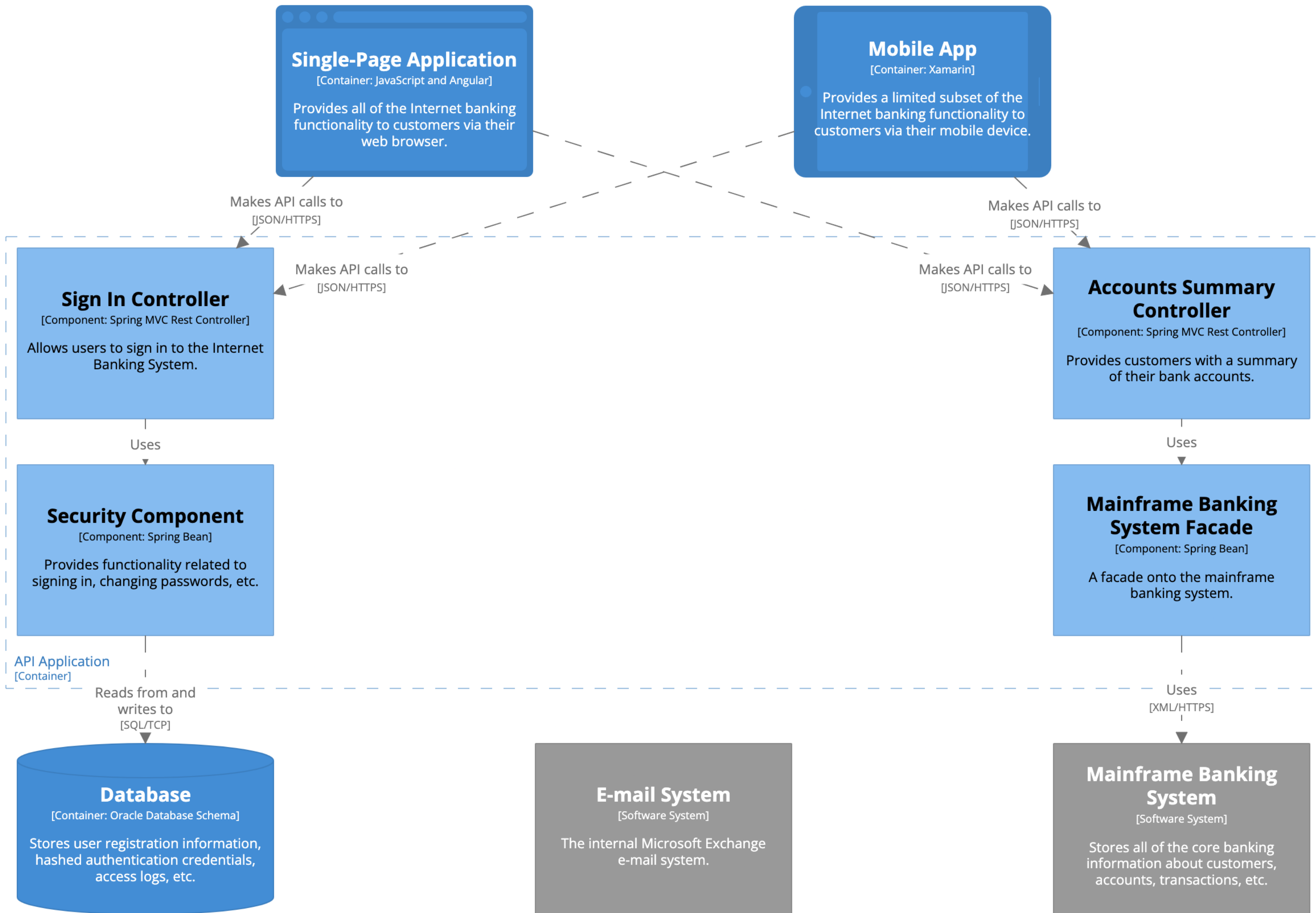
Monday, 27 February 2023 at 15:36 Greenwich Mean Time



[Component] Internet Banking System - API Application

The component diagram for the API Application.

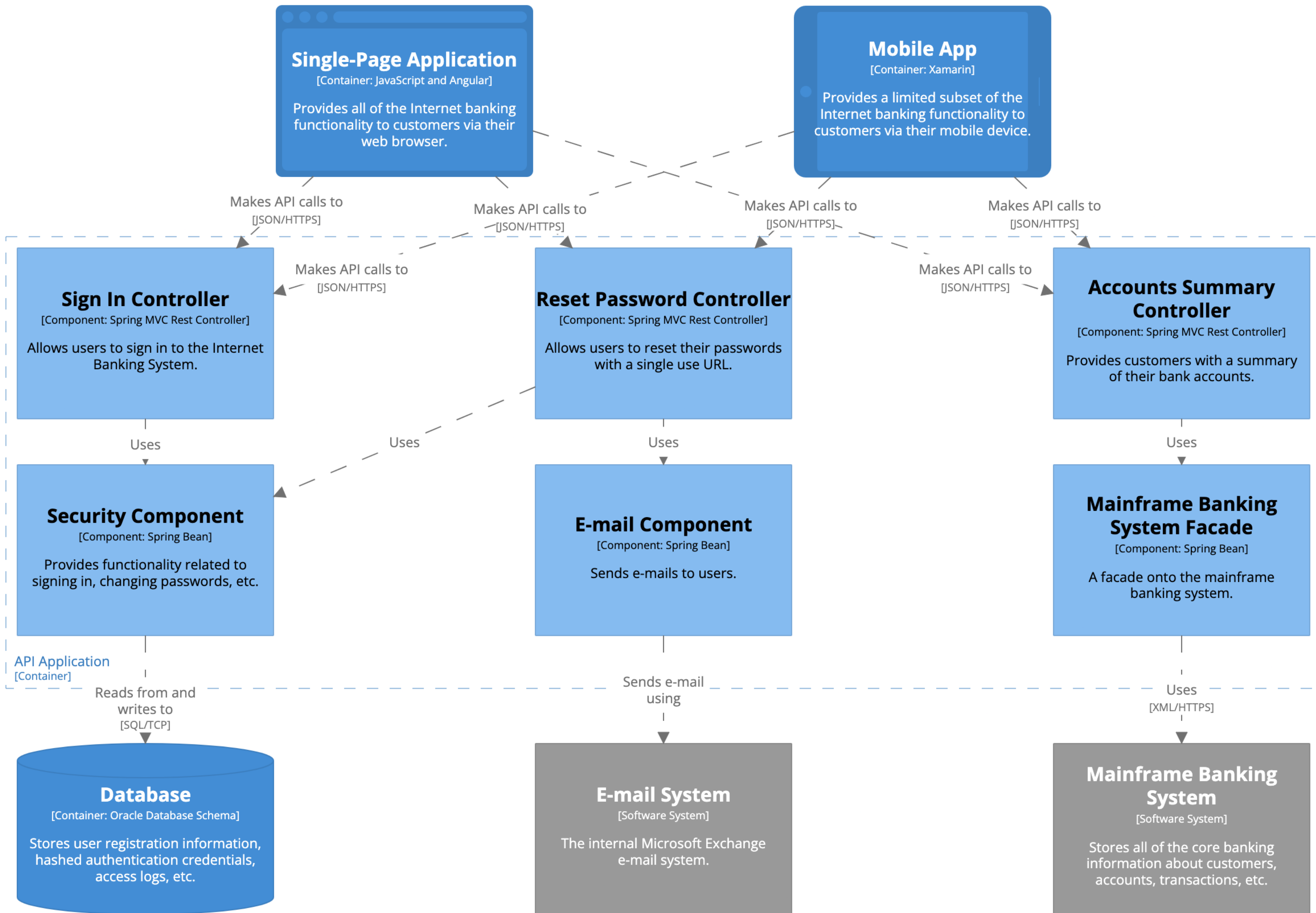
Monday, 27 February 2023 at 15:36 Greenwich Mean Time



[Component] Internet Banking System - API Application

The component diagram for the API Application.

Monday, 27 February 2023 at 15:36 Greenwich Mean Time



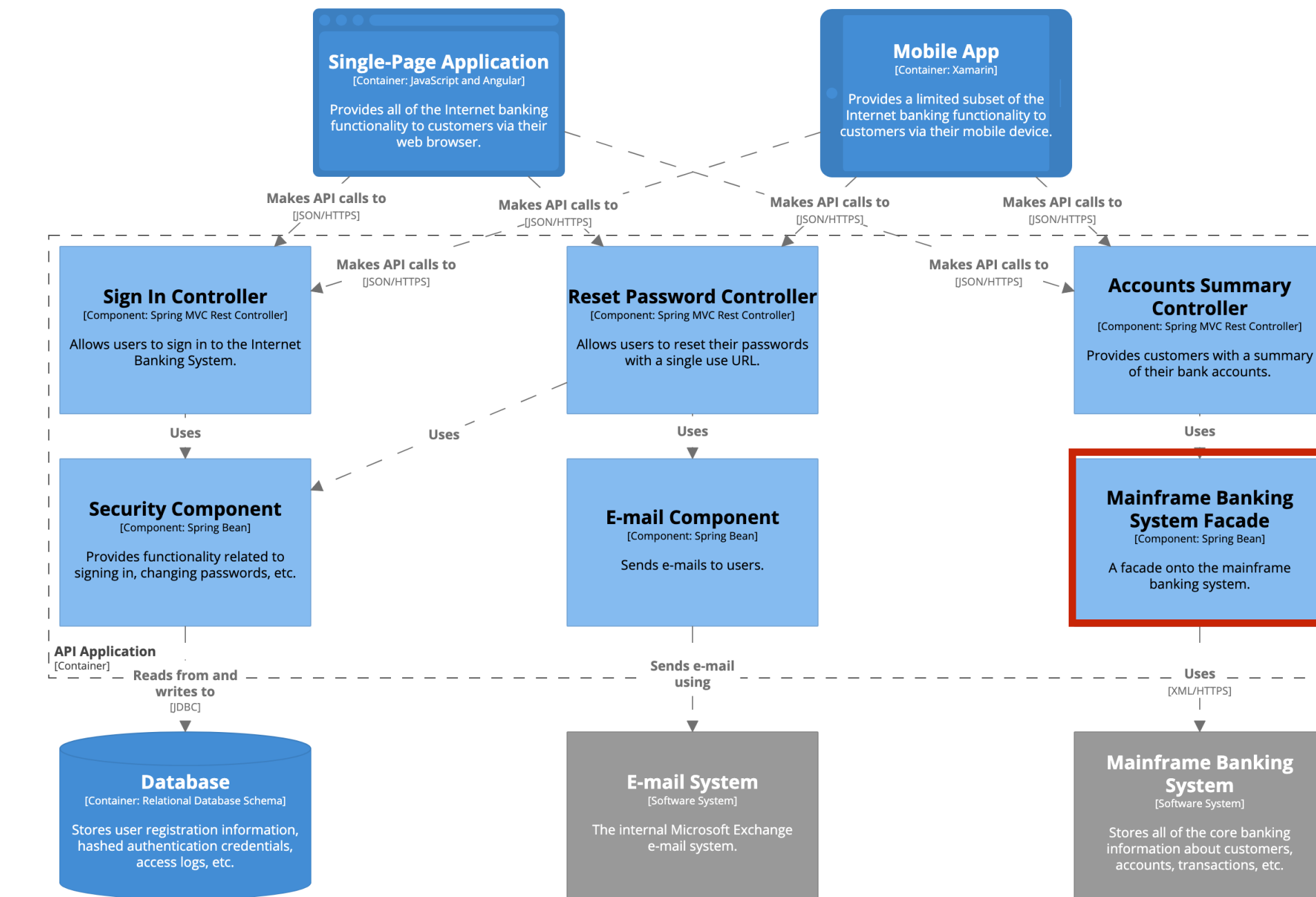
[Component] Internet Banking System - API Application

The component diagram for the API Application.

Monday, 27 February 2023 at 15:36 Greenwich Mean Time

Level 4

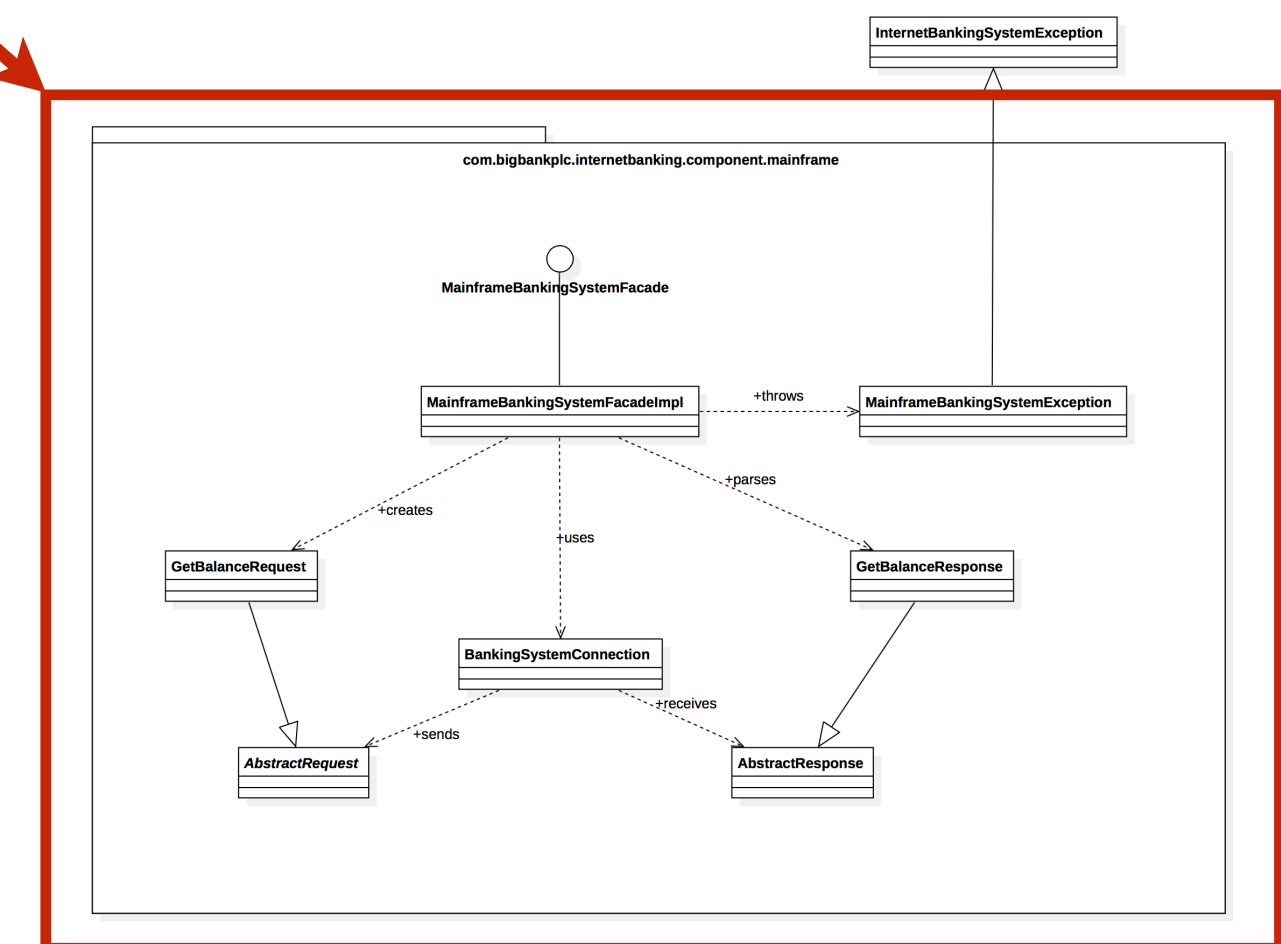
Code diagram

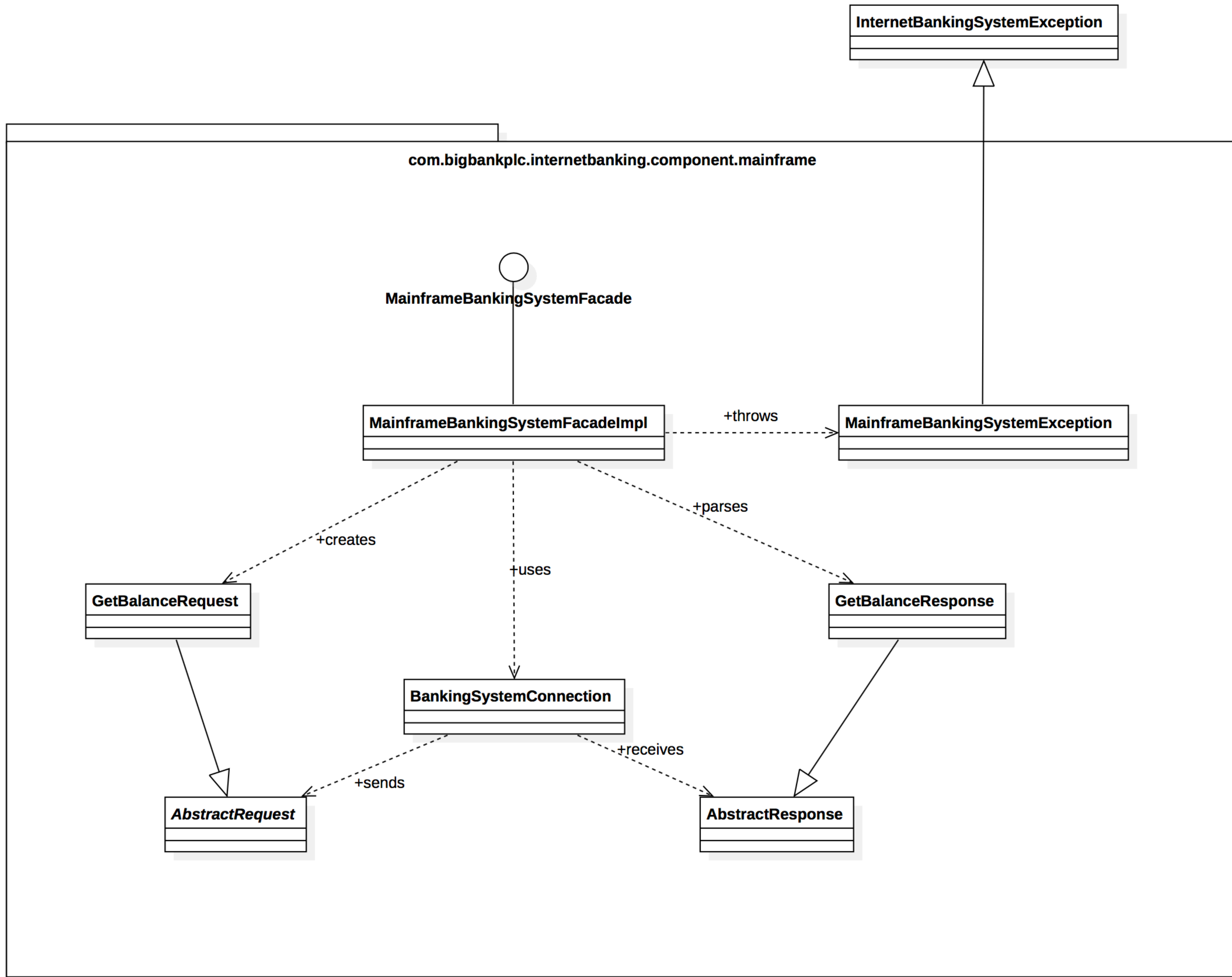


Component diagram for Internet Banking System - API Application

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

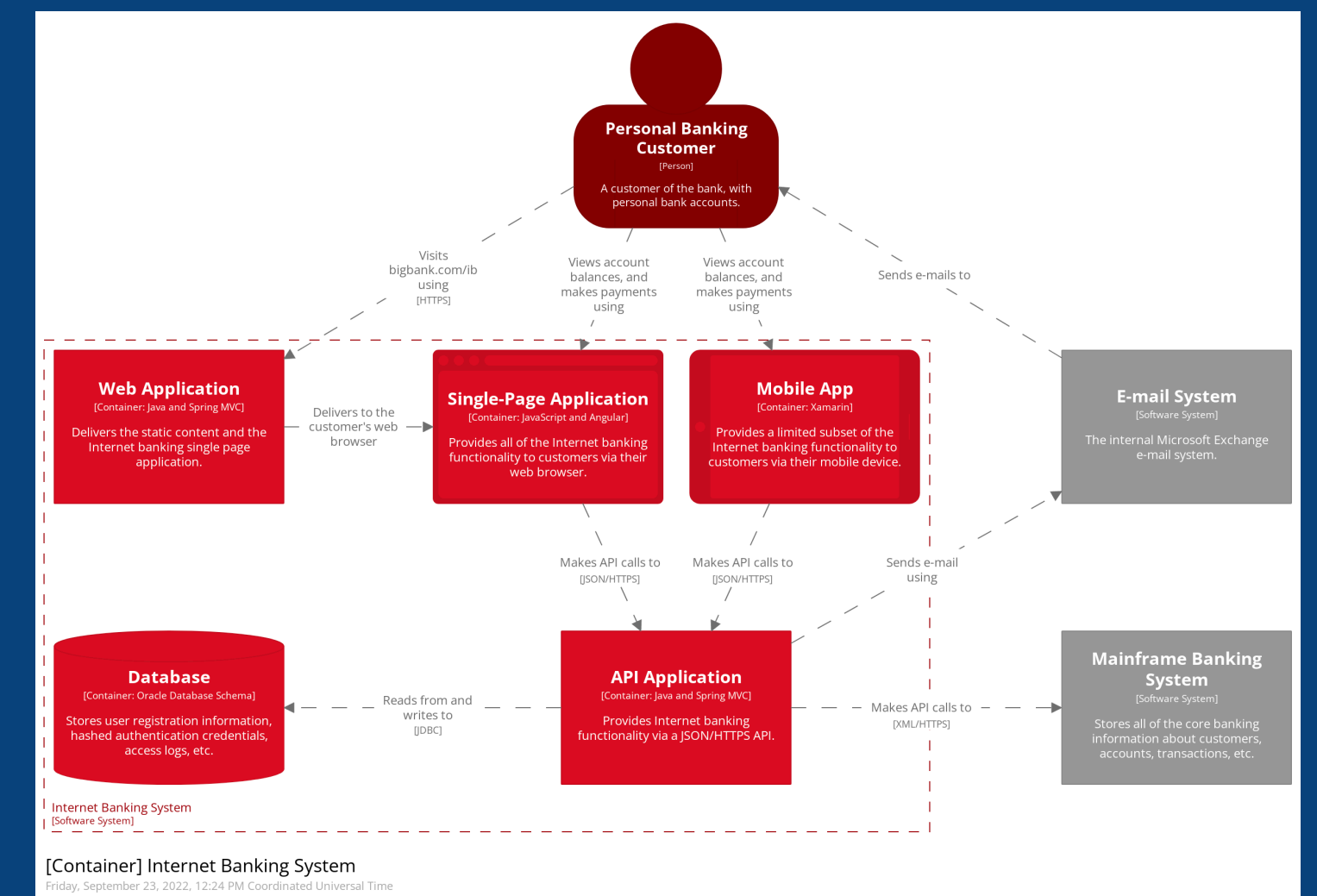
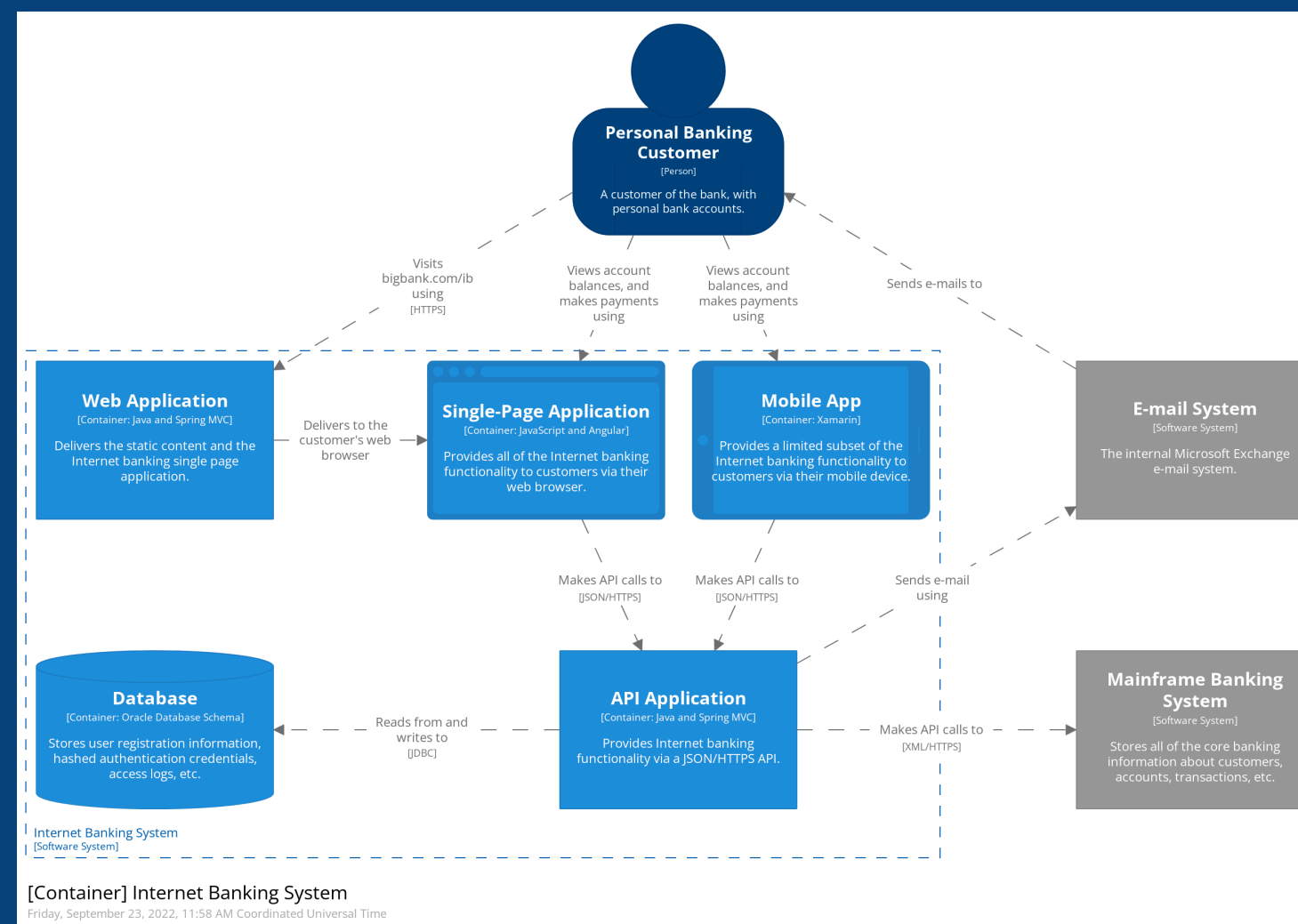
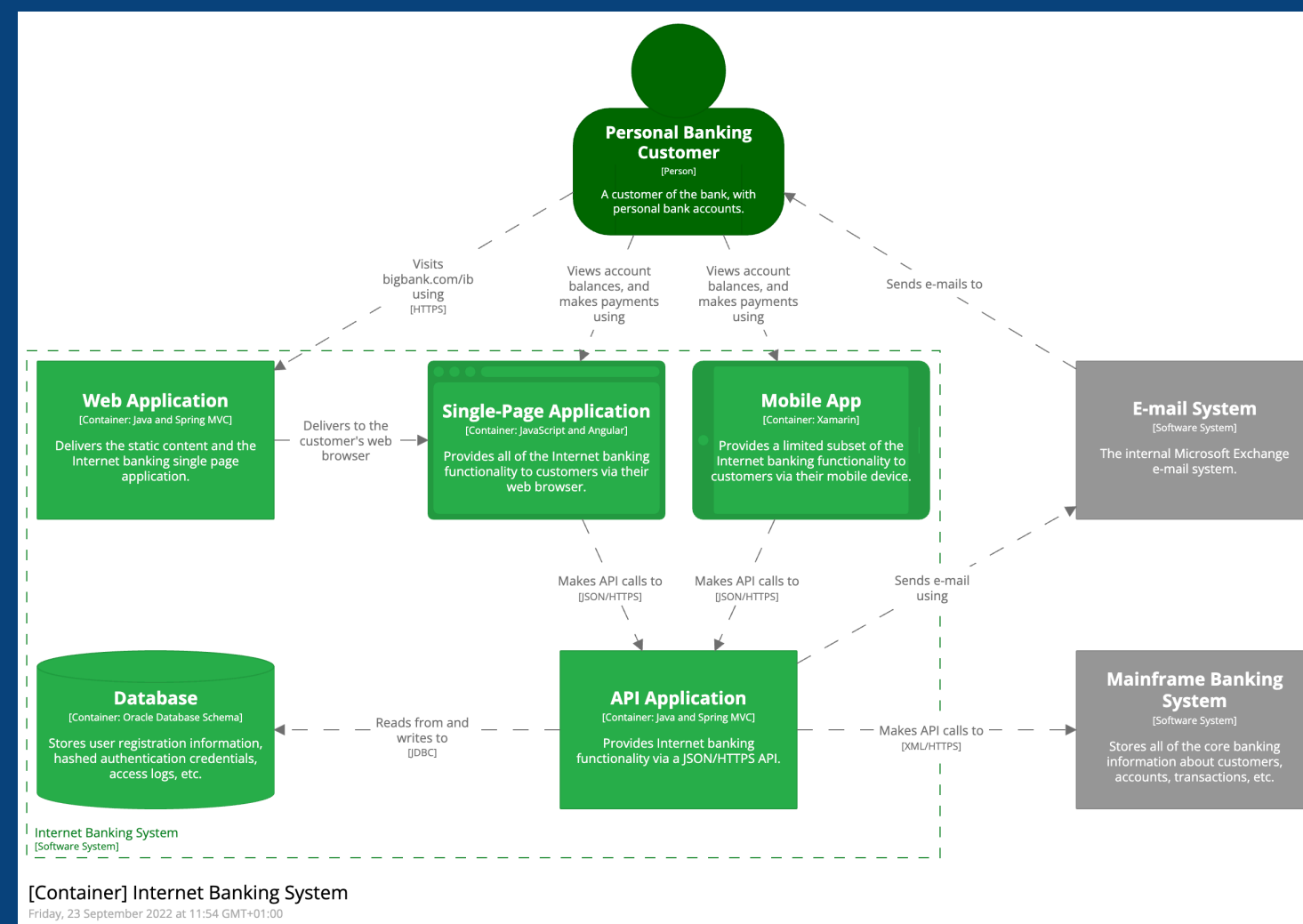
The code level diagram shows the code elements that make up a component



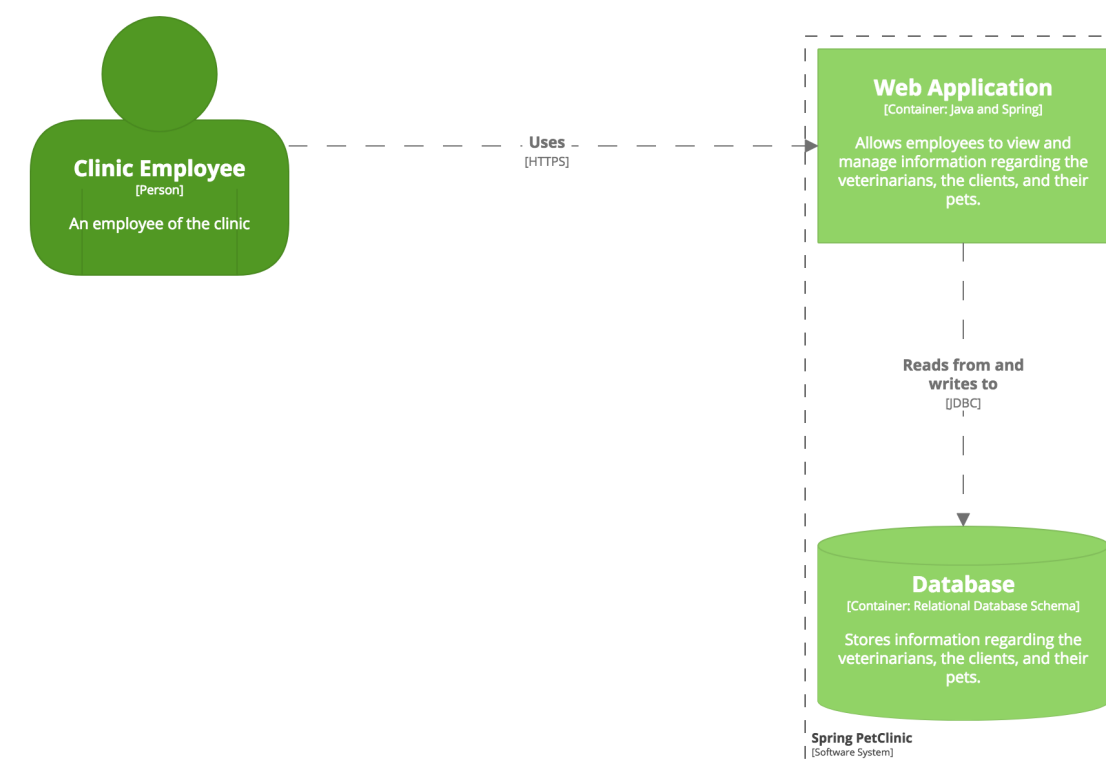


Notation

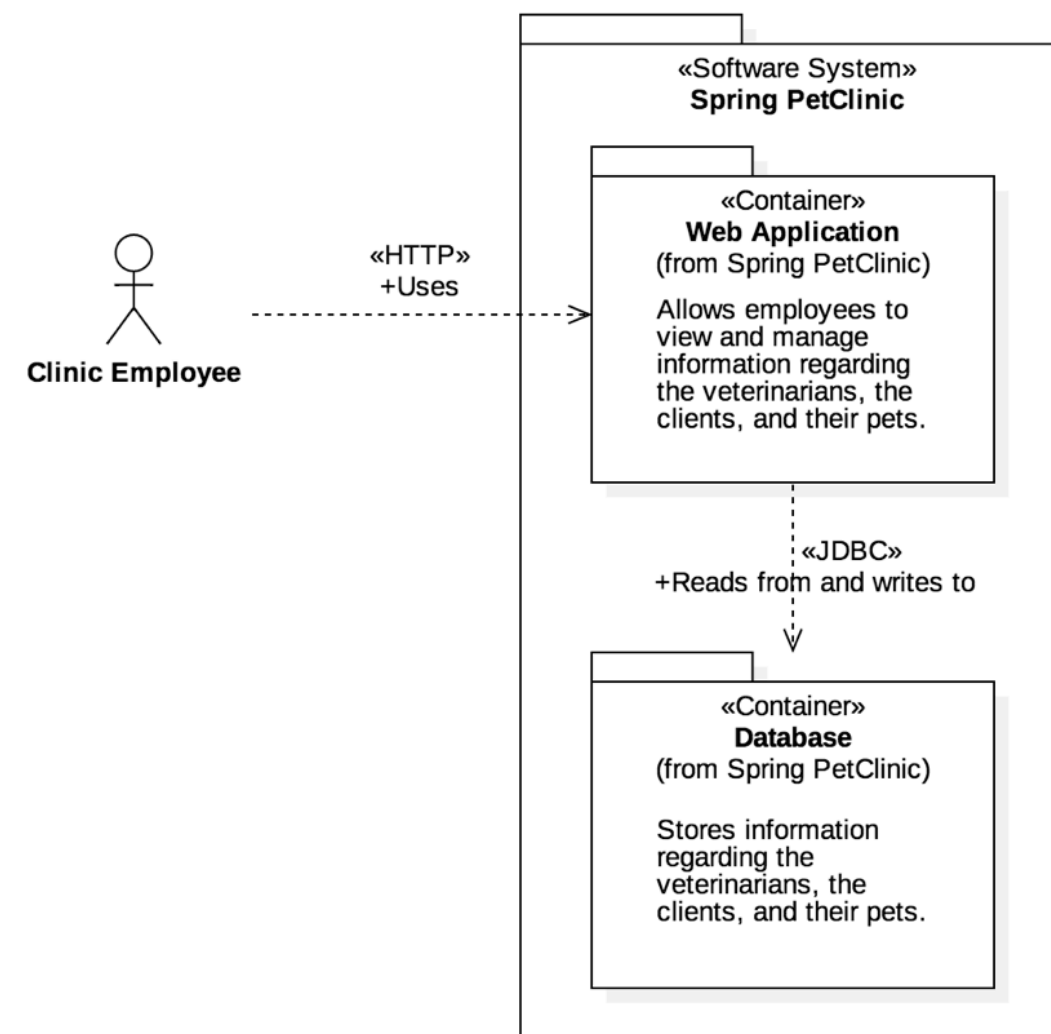
The C4 model is notation independent



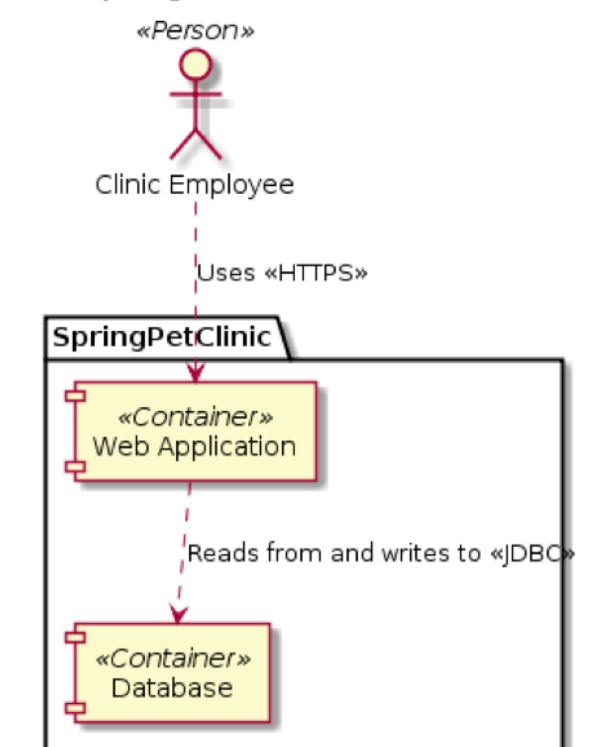
The C4 model is notation independent



Container diagram for Spring PetClinic
The Containers diagram for the Spring PetClinic system.
Last modified: Thursday 17 August 2017 10:15 UTC | Version: 95de1d9f8b6f3560915331664b27a4a75ce1f1f6

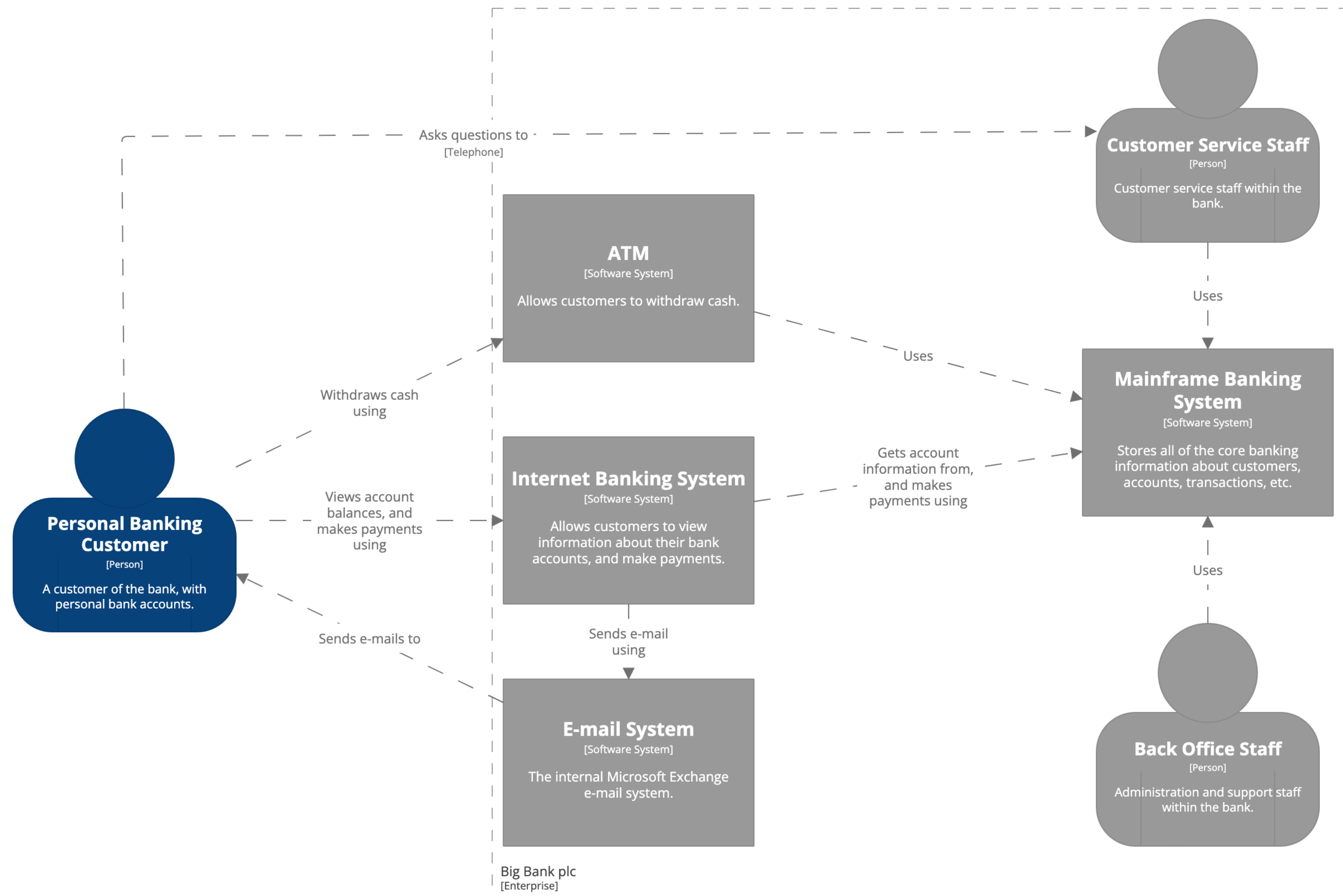


Spring PetClinic - Containers



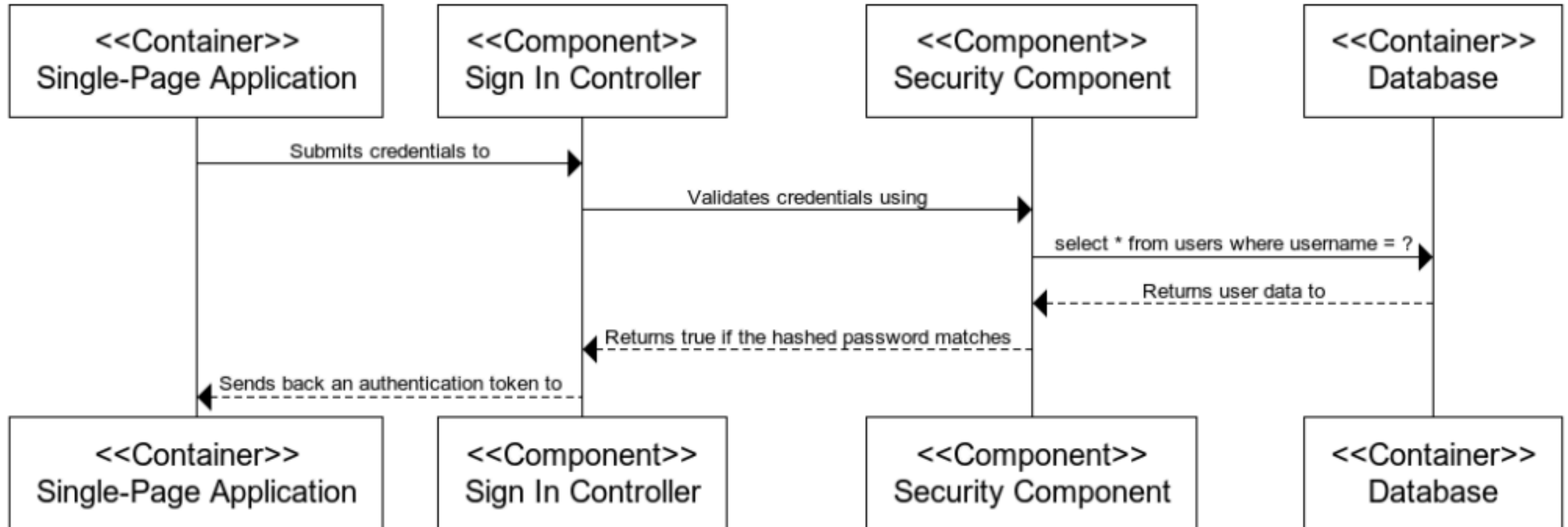
The Container diagram for the Spring PetClinic system.

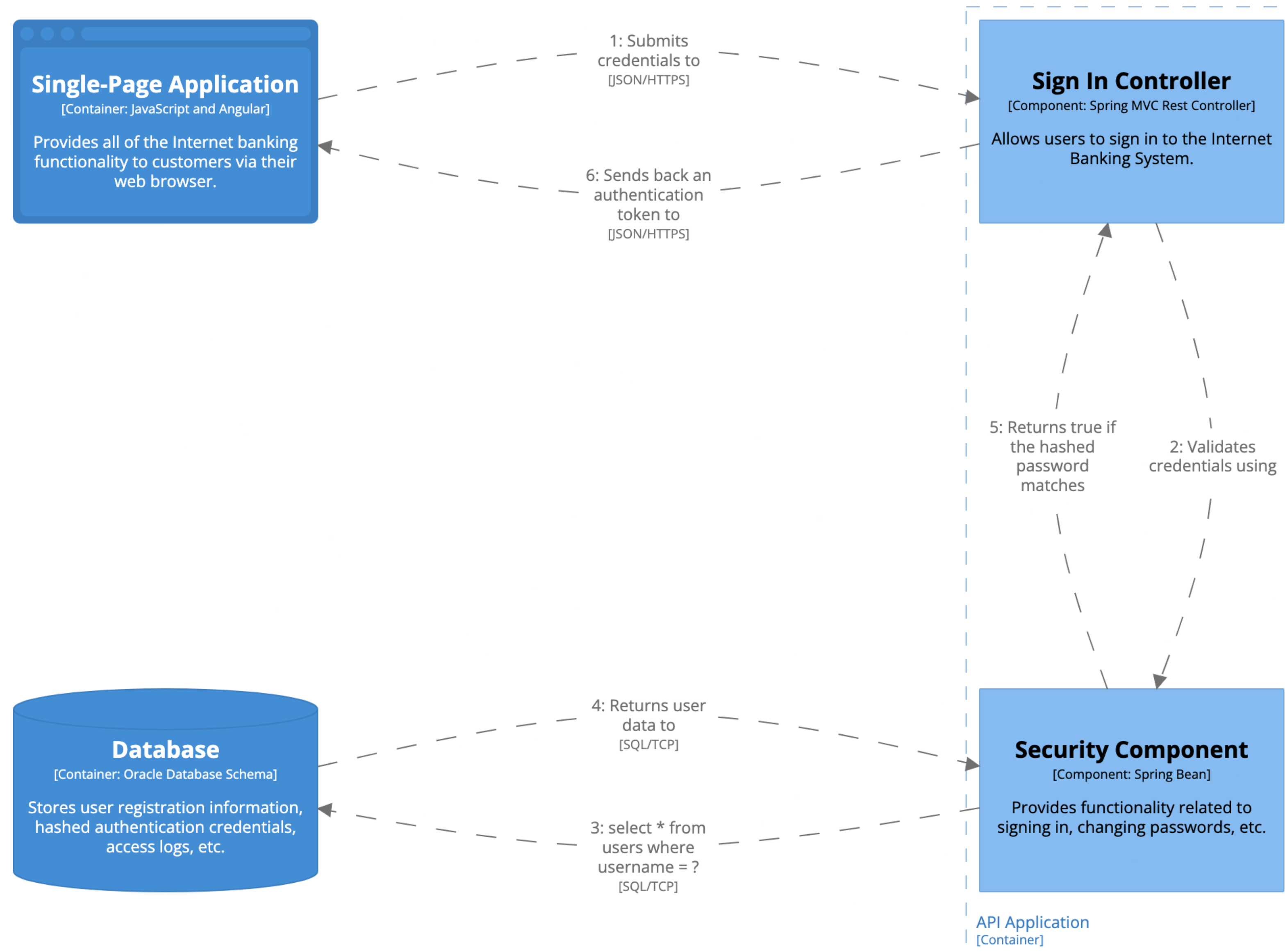
System landscape diagrams



Dynamic diagrams

API Application - Dynamic - SignIn



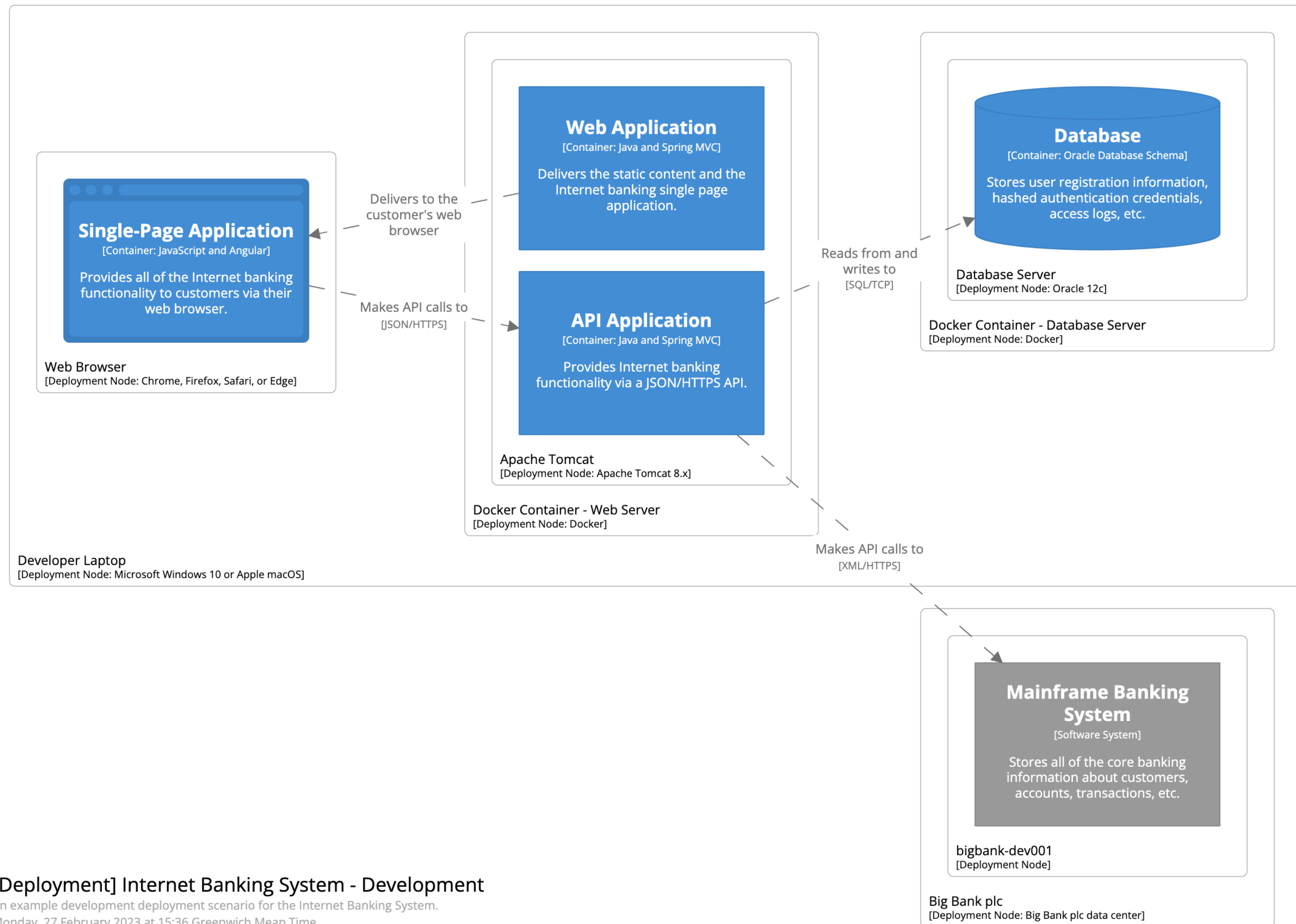


[Dynamic] Internet Banking System - API Application

Summarises how the sign in feature works in the single-page application.

Monday, 27 February 2023 at 15:36 Greenwich Mean Time

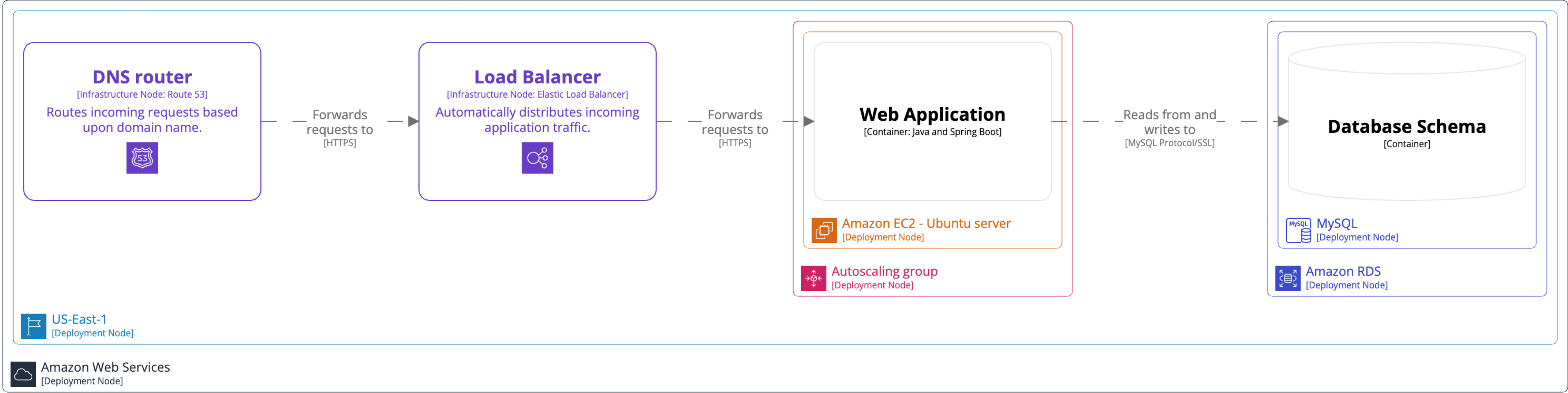
Deployment diagrams



[Deployment] Internet Banking System - Development

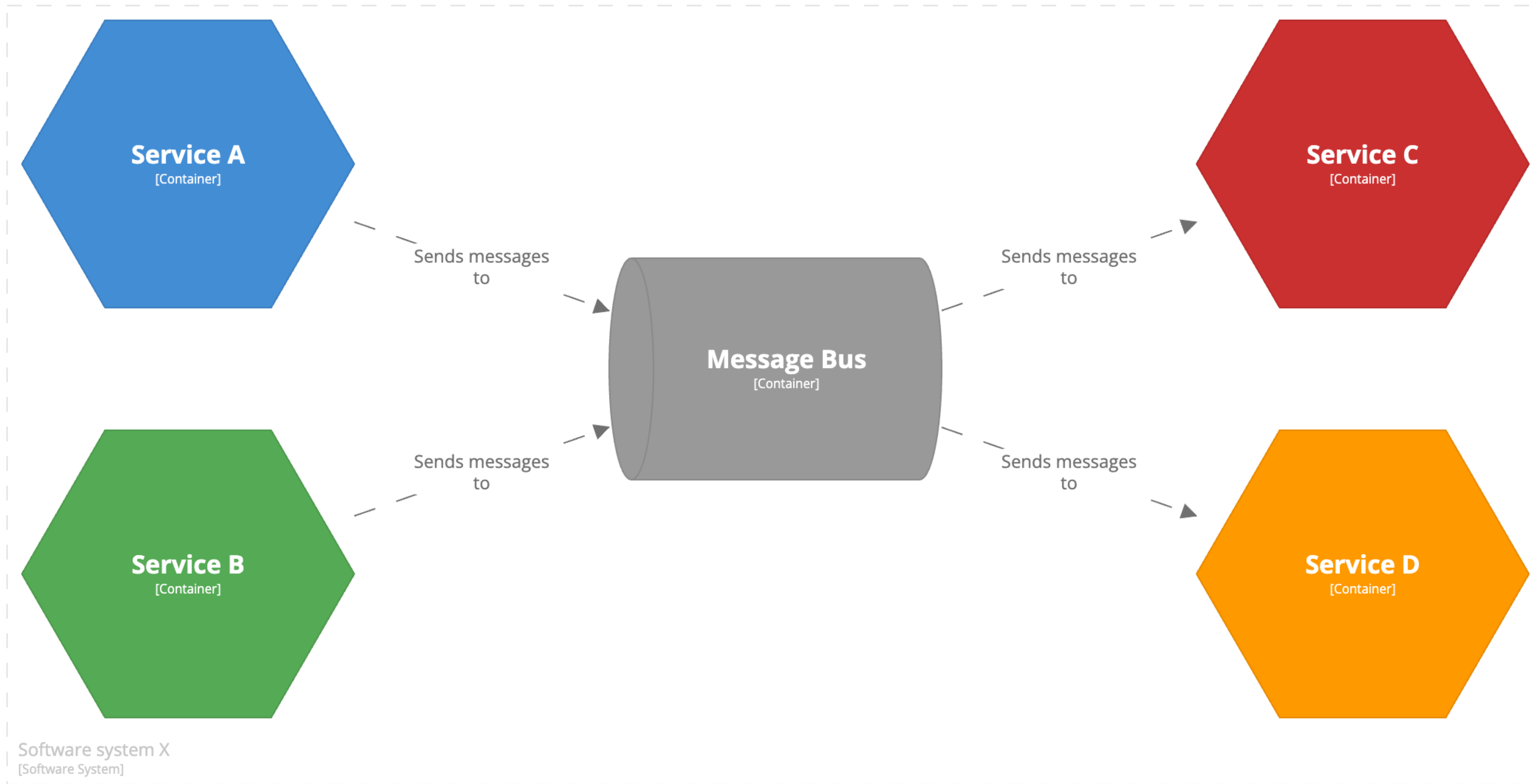
An example development deployment scenario for the Internet Banking System.

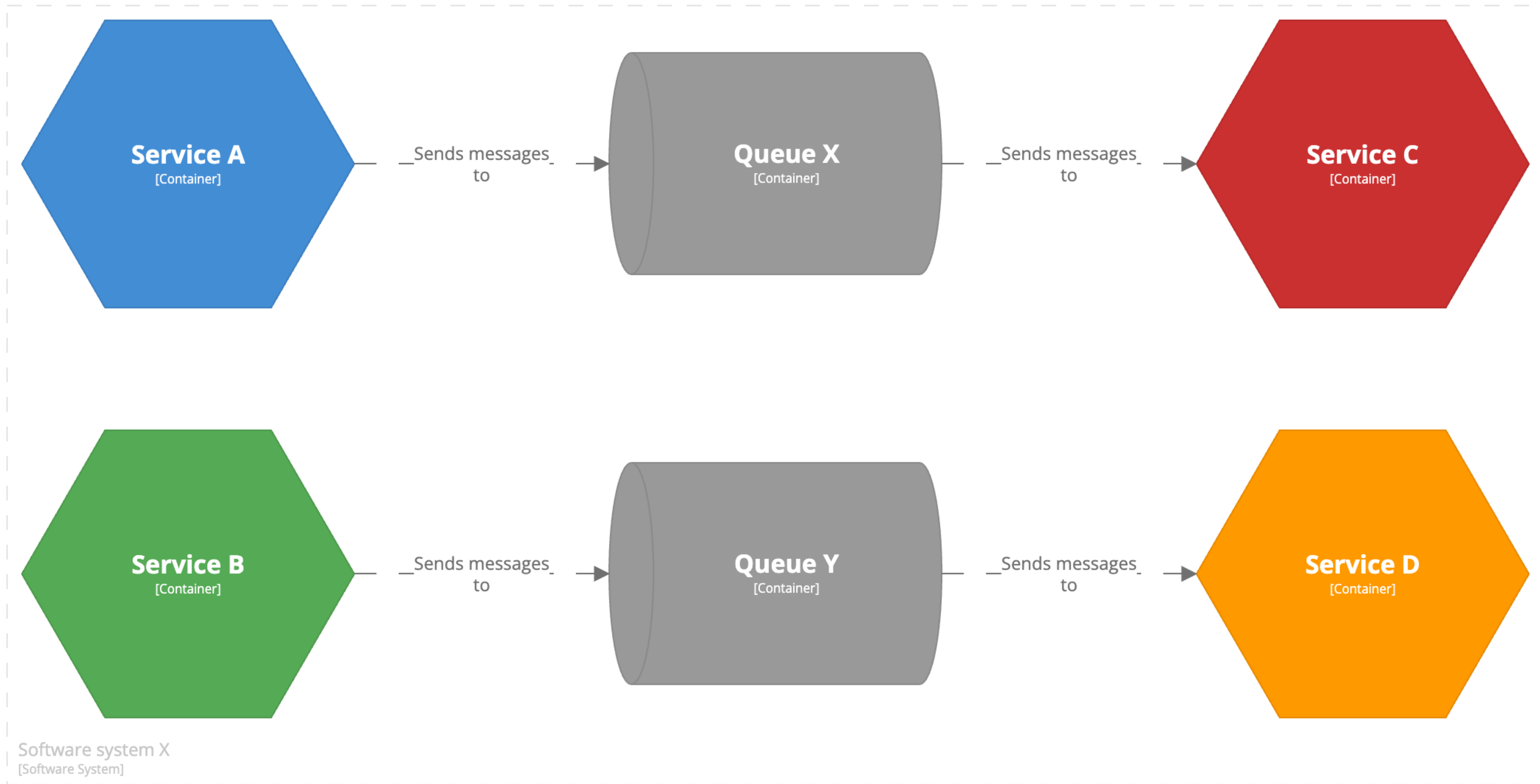
Monday, 27 February 2023 at 15:36 Greenwich Mean Time

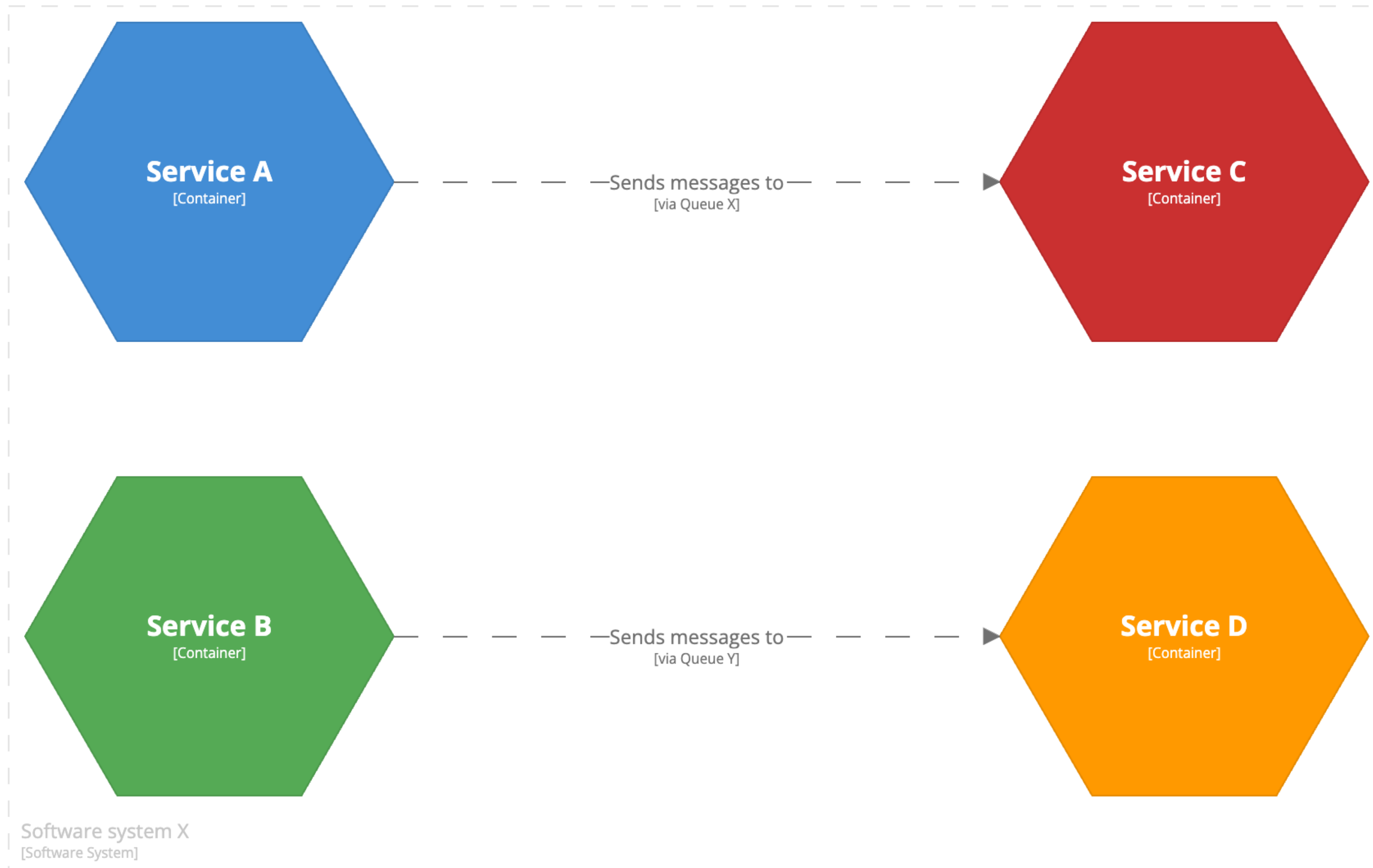


FAQs

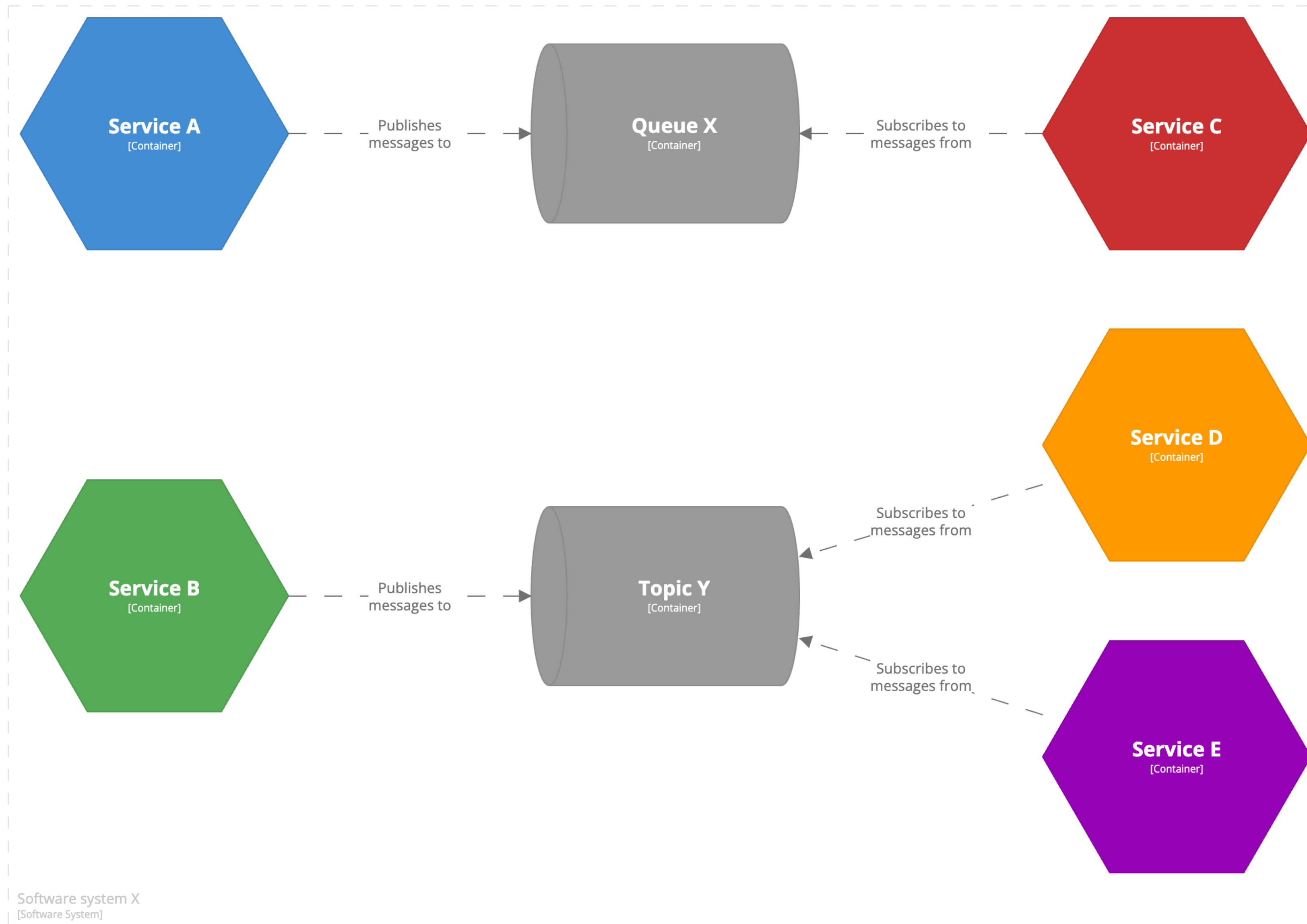
Message-driven architectures







[Container] Software system X



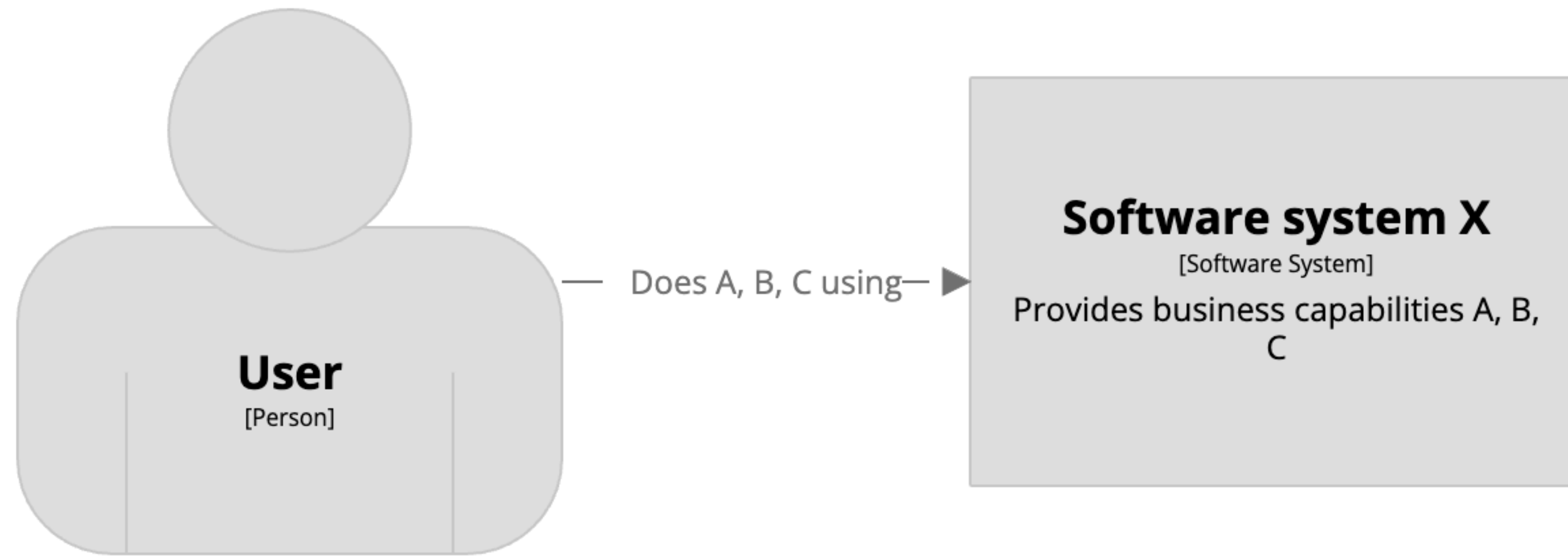
Microservices

A microservice should be modelled
as one of the following:

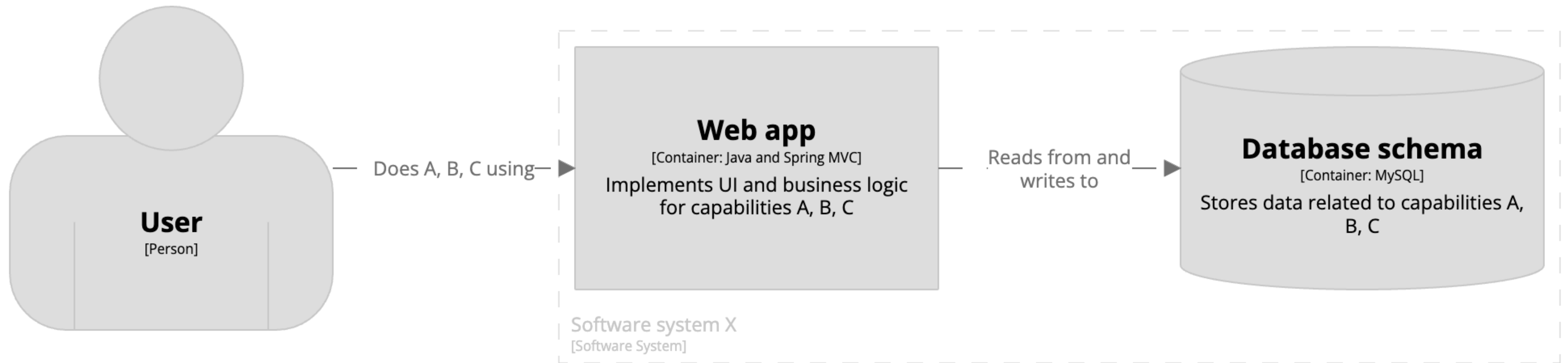
1. A software system
2. A container
3. A group of containers

What is a
“microservice”?

Stage 1: 
(monolithic architecture)



[System Context] Software system X



[Container] Software system X

Stage 2:  
(microservices)

Microservices

a definition of this new architectural term

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014



James Lewis

James Lewis is a Principal Consultant at Thoughtworks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services

CONTENTS

Characteristics of a Microservice Architecture

[Componentization via Services](#)

[Organized around Business Capabilities](#)

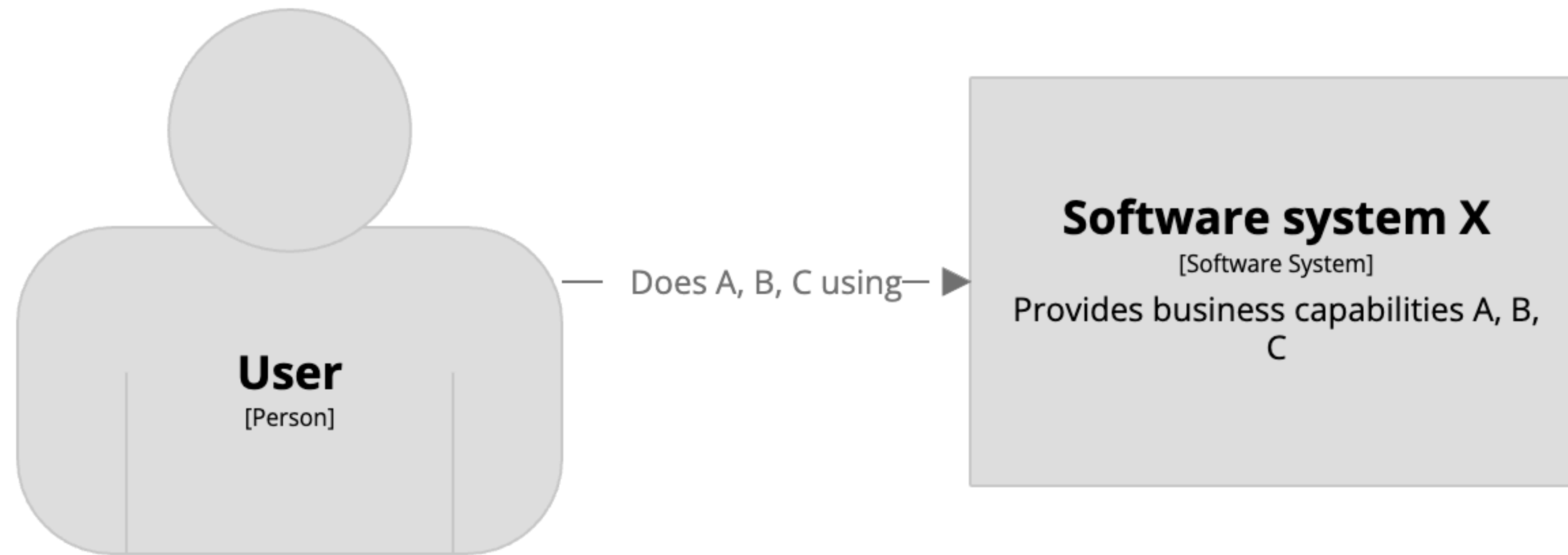
[Products not Projects](#)

[Smart endpoints and dumb pipes](#)

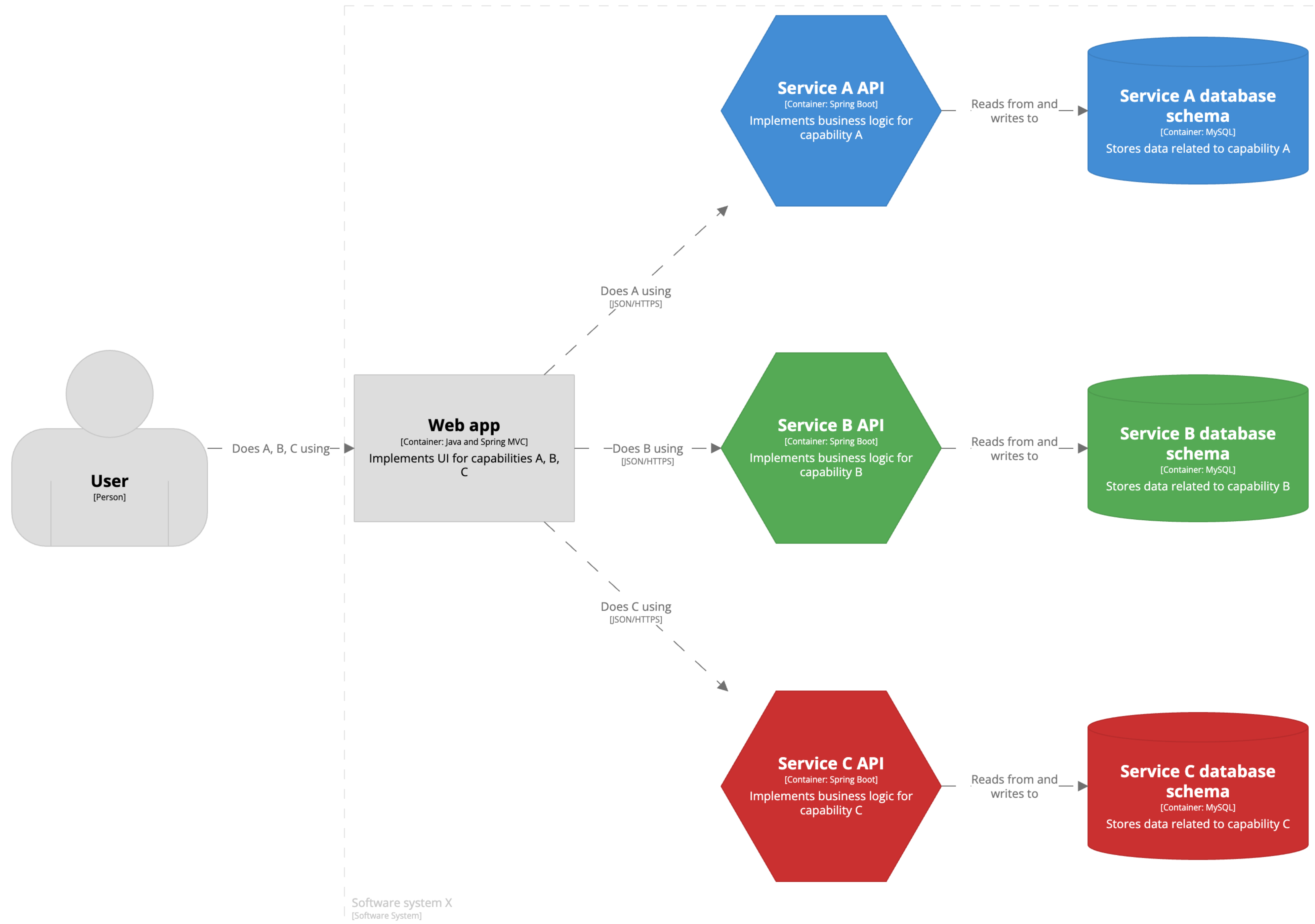
[Decentralized Governance](#)

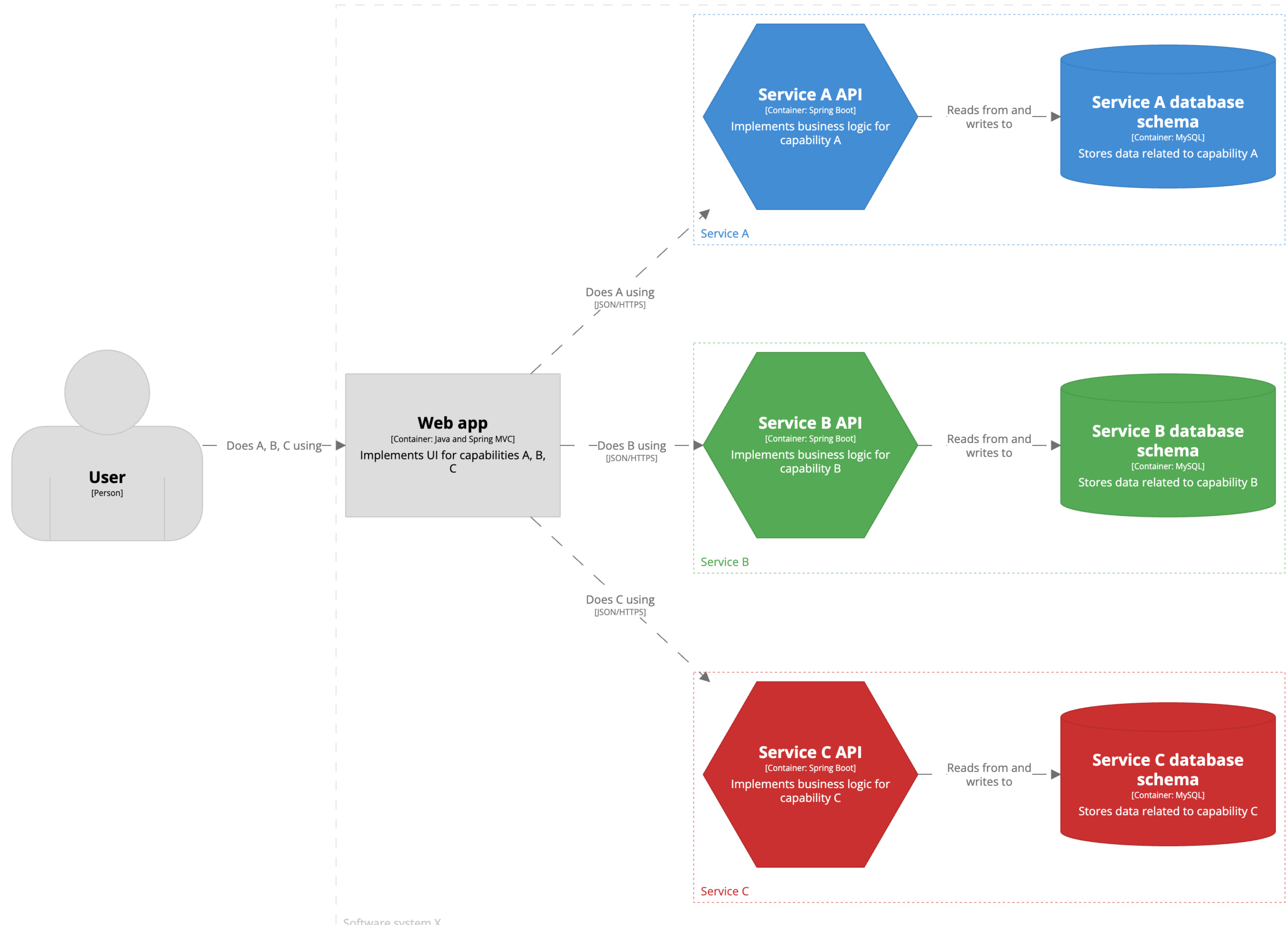
[Decentralized Data Management](#)

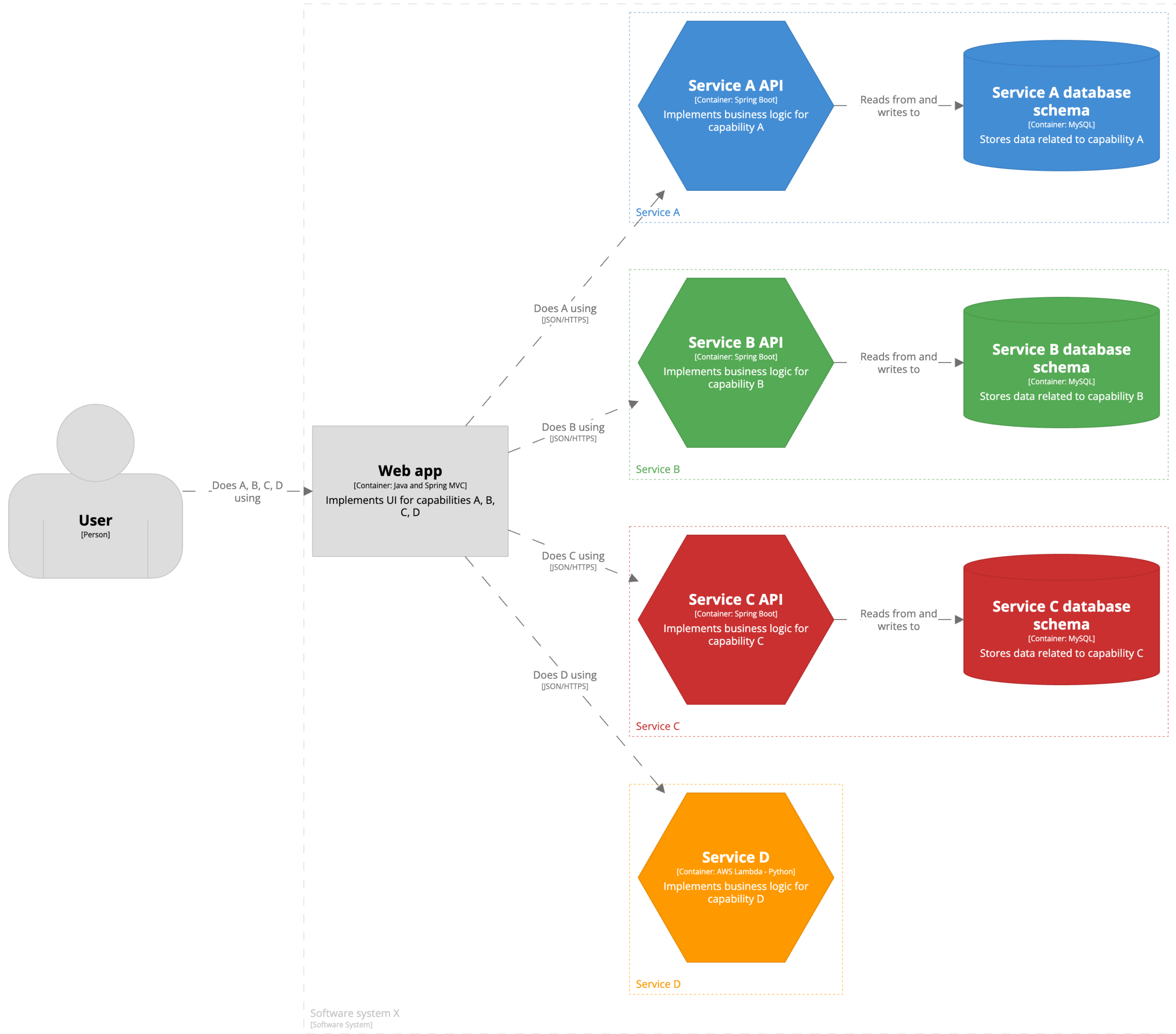
In short, the microservice architectural style [1] is an approach to developing a single software system as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.



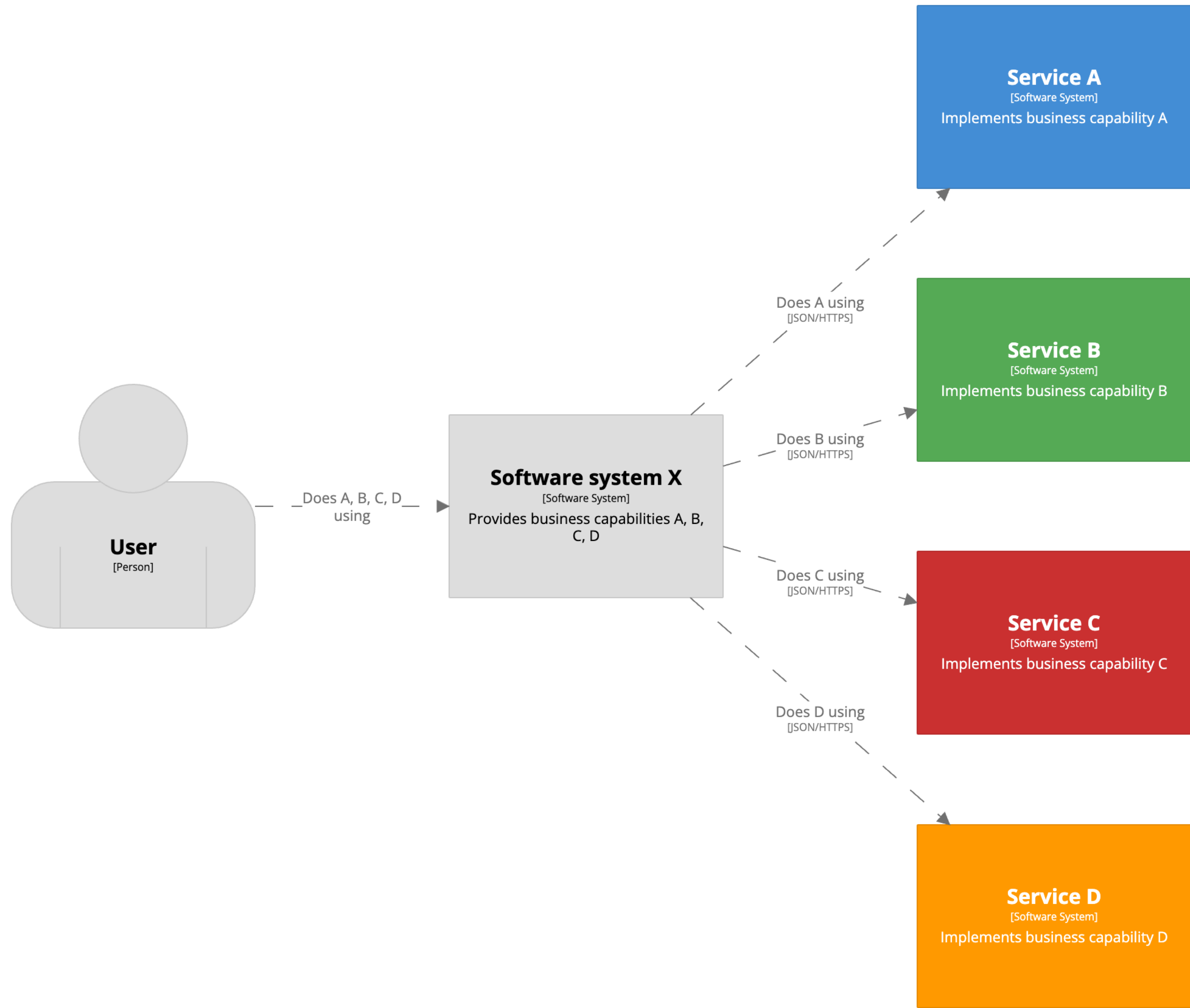
[System Context] Software system X

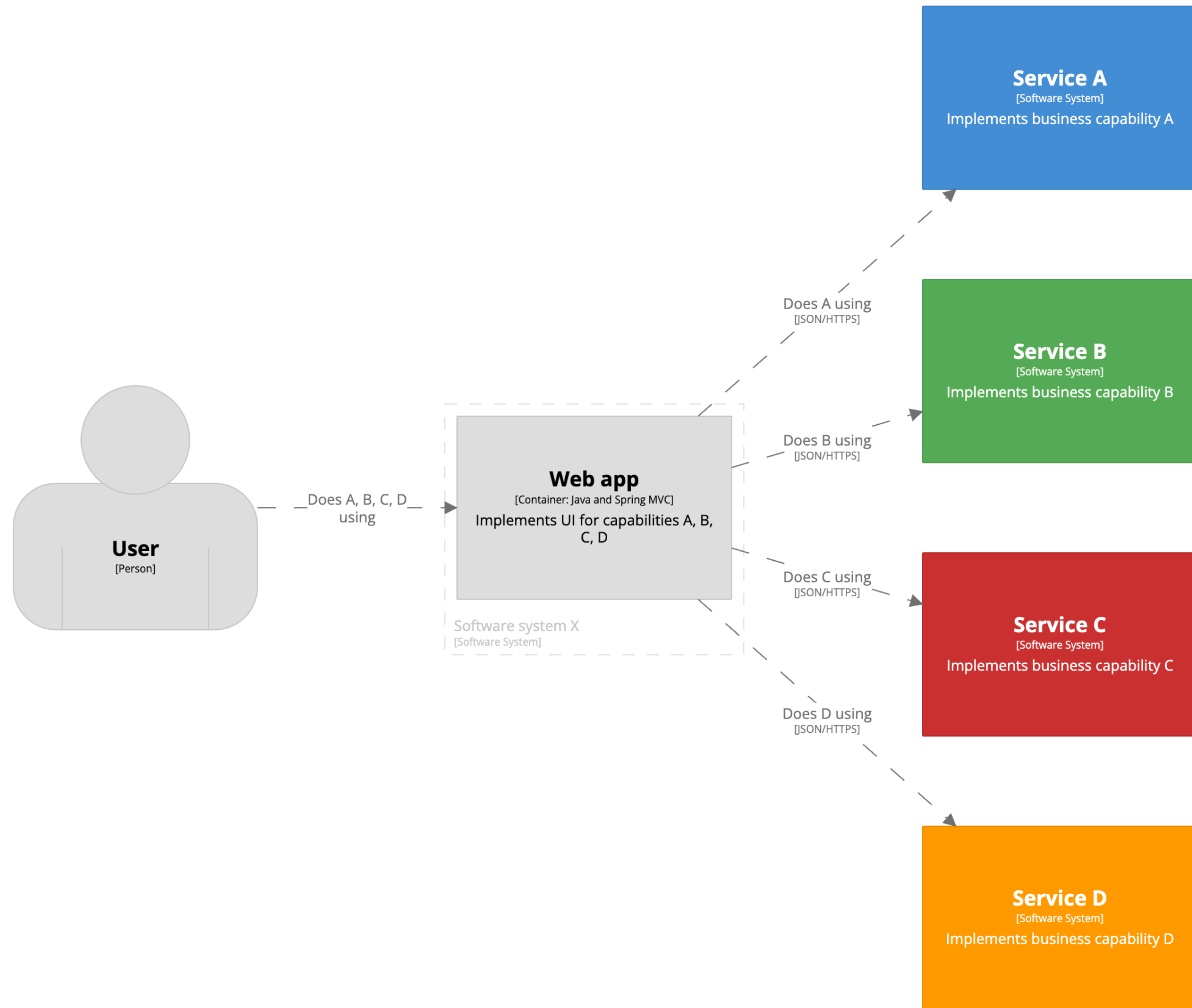






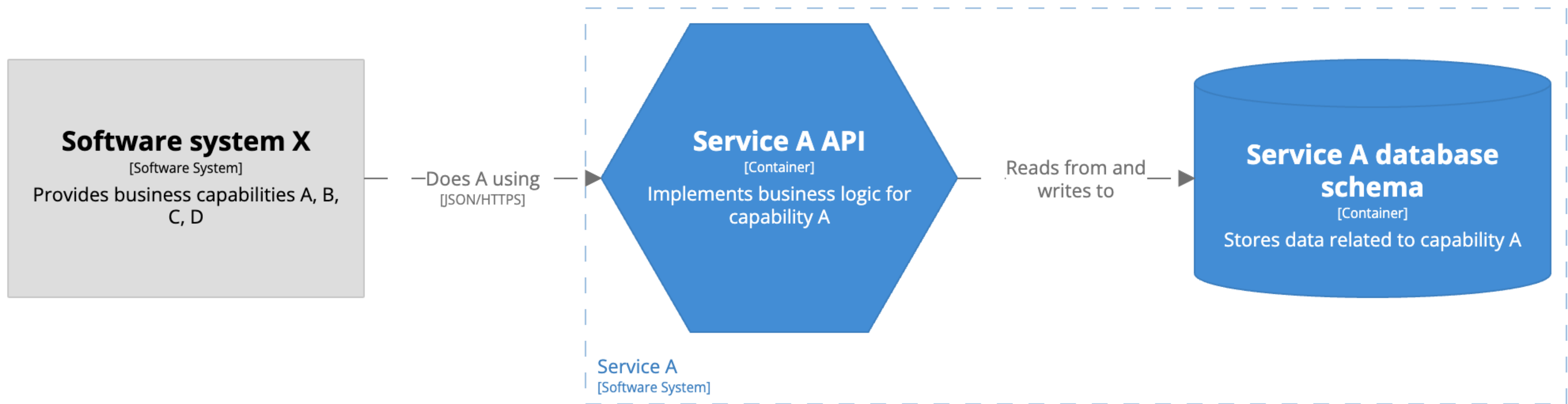
Stage 3: 
(Conway's Law)







[System Context] Service A



[Container] Service A

Summary

The C4 model is...

A set of hierarchical
abstractions

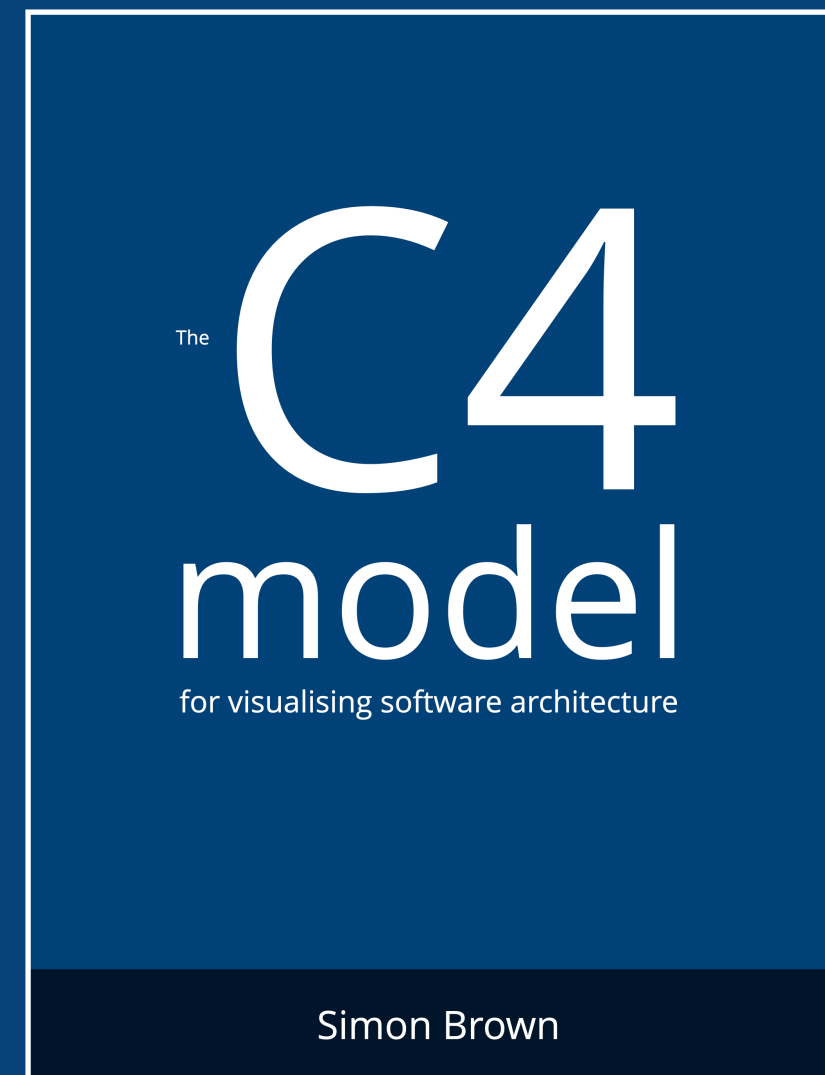
(software systems, containers,
components, and code)

A set of hierarchical
diagrams

(system context, containers, components,
and code)

Notation independent

Tooling independent



Buy both books for \$15
(expires midnight)



<https://leanpub.com/b/software-architecture/c/portotechhub2024>