lead-gen-c4

Overview

Level 1: System Context diagram

A System Context diagram is a good starting point for diagramming and documenting a software system, allowing you to step back and see the big picture. Draw a diagram showing your system as a box in the centre, surrounded by its users and the other systems that it interacts with.

Detail isn't important here as this is your zoomed out view showing a big picture of the system landscape. The focus should be on people (actors, roles, personas, etc) and software systems rather than technologies, protocols and other low-level details. It's the sort of diagram that you could show to non-technical people.
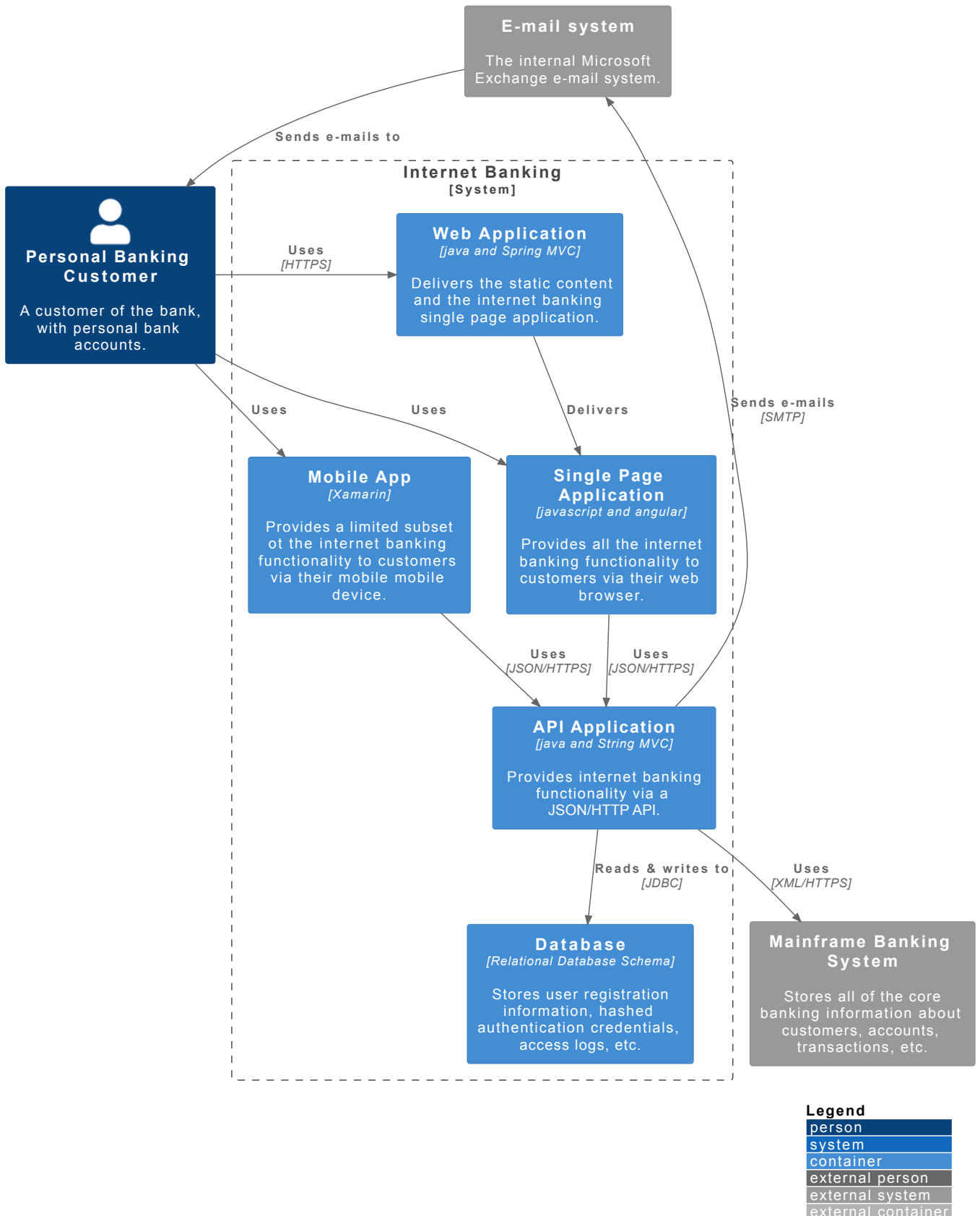
Scope: A single software system.

Primary elements: The software system in scope. Supporting elements: People (e.g. users, actors, roles, or personas) and software systems (external dependencies) that are directly connected to the software system in scope. Typically these other software systems sit outside the scope or boundary of your own software system, and you don't have responsibility or ownership of them.

Intended audience: Everybody, both technical and non-technical people, inside and outside of the software development team.

1 Lead Gen Platform

/1 Lead Gen Platform

E-mail system
The internal Microsoft Exchange e-mail system.

Internet Banking
[System]

Sends e-mails to

Personal Banking Customer
A customer of the bank, with personal bank accounts.

Uses
[HTTPS]

Web Application
[java and Spring MVC]
Delivers the static content and the internet banking single page application.

Uses

Uses

Delivers

Sends e-mails
[SMTP]

Mobile App
[Xamarin]
Provides a limited subset ot the internet banking functionality to customers via their mobile mobile device.

Single Page Application
[javascript and angular]
Provides all the internet banking functionality to customers via their web browser.

Uses
[JSON/HTTPS]

Uses
[JSON/HTTPS]

API Application
[java and String MVC]
Provides internet banking functionality via a JSON/HTTP API.

Reads & writes to
[JDBC]

Uses
[XML/HTTPS]

Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.

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

Legend
person
system
container
external person
external system
external container

Level 2: Container diagram

Once you understand how your system fits in to the overall IT environment, a really useful next step is to zoom-in to the system boundary with a Container diagram. A "container" is something like a server-side web application, single-page application, desktop application, mobile app, database schema, file system, etc. Essentially, a container is a separately runnable/deployable unit (e.g. a separate process space) that executes code or stores data.

The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another. It's a simple, high-level technology focussed diagram that is useful for software developers and support/operations staff alike.
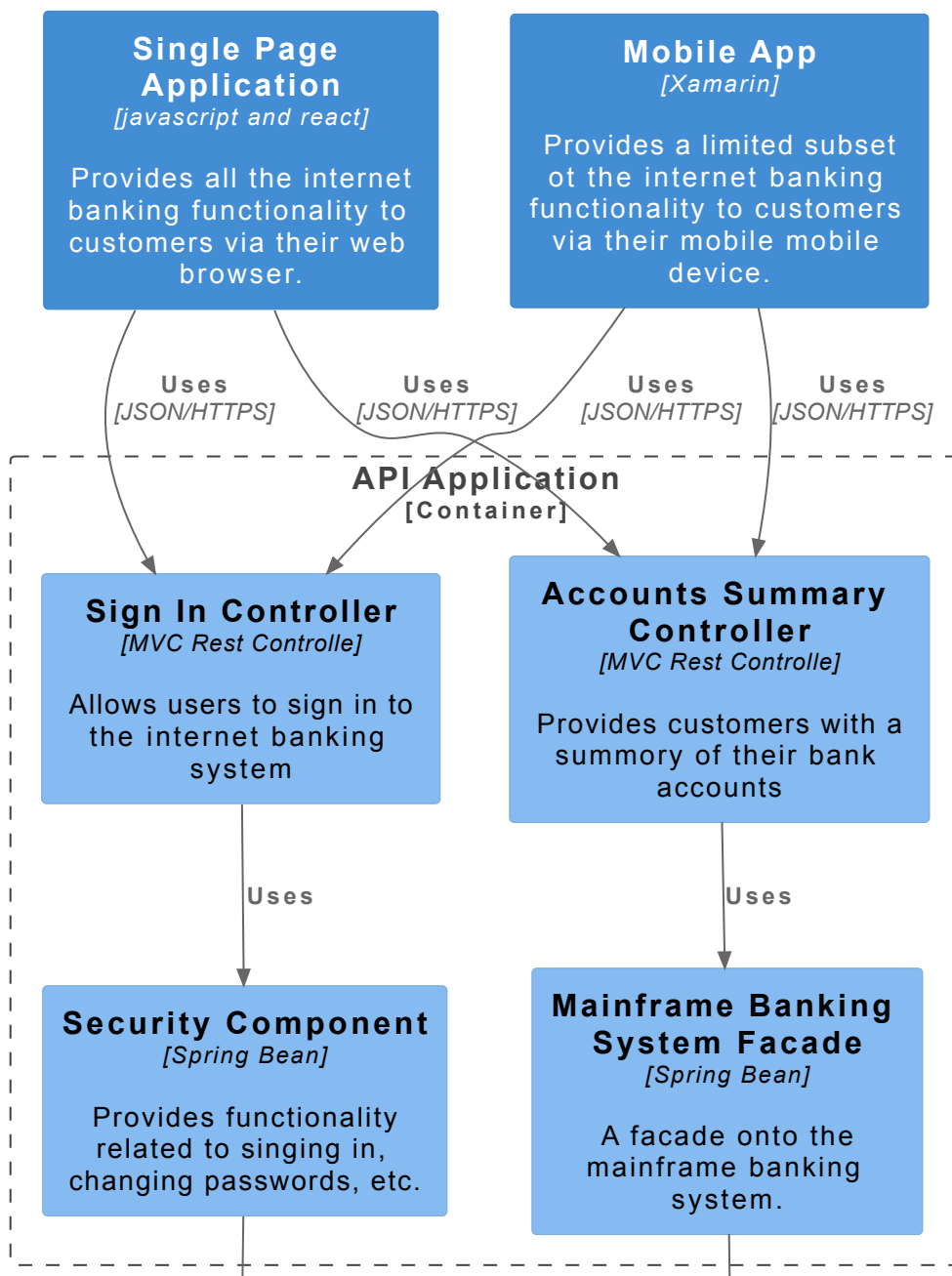
Scope: A single software system.

Primary elements: Containers within the software system in scope. Supporting elements: People and software systems directly connected to the containers.
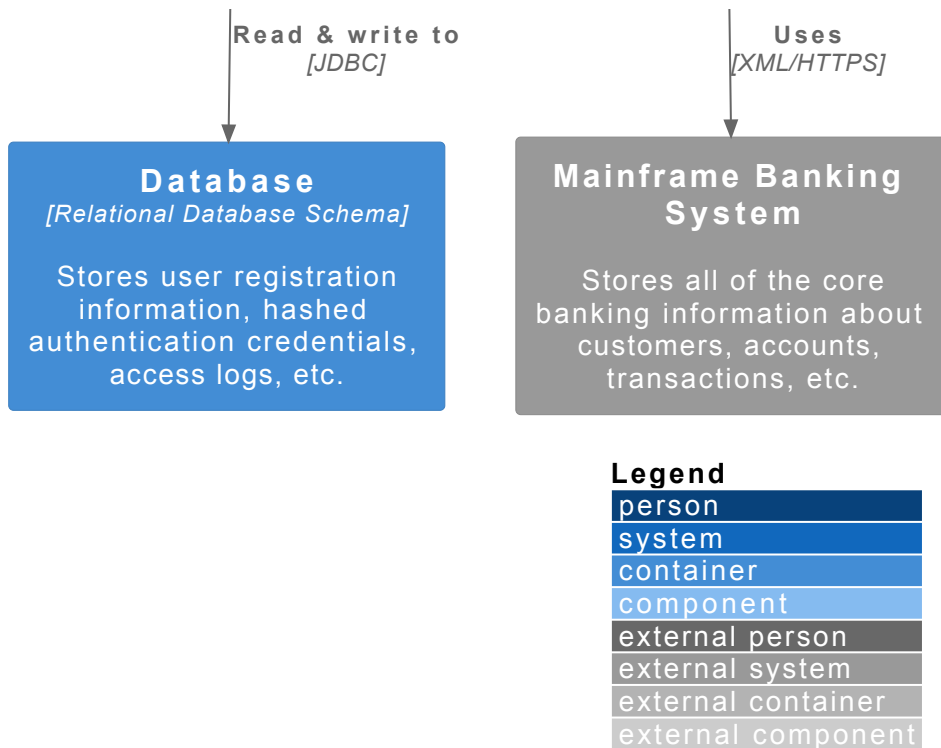
Intended audience: Technical people inside and outside of the software development team; including software architects, developers and operations/support staff.

Notes: This diagram says nothing about deployment scenarios, clustering, replication, failover, etc.

API Application

/1 Lead Gen Platform/API Application

Read & write to
[JDBC]

Uses
[XML/HTTPS]

Database
[Relational Database Schema]

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

Mainframe Banking System

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

Legend

person
system
container
component
external person
external system
external container
external component

Level 3: Component diagram

Next you can zoom in and decompose each container further to identify the major structural building blocks and their interactions.

The Component diagram shows how a container is made up of a number of "components", what each of those components are, their responsibilities and the technology/implementation details.
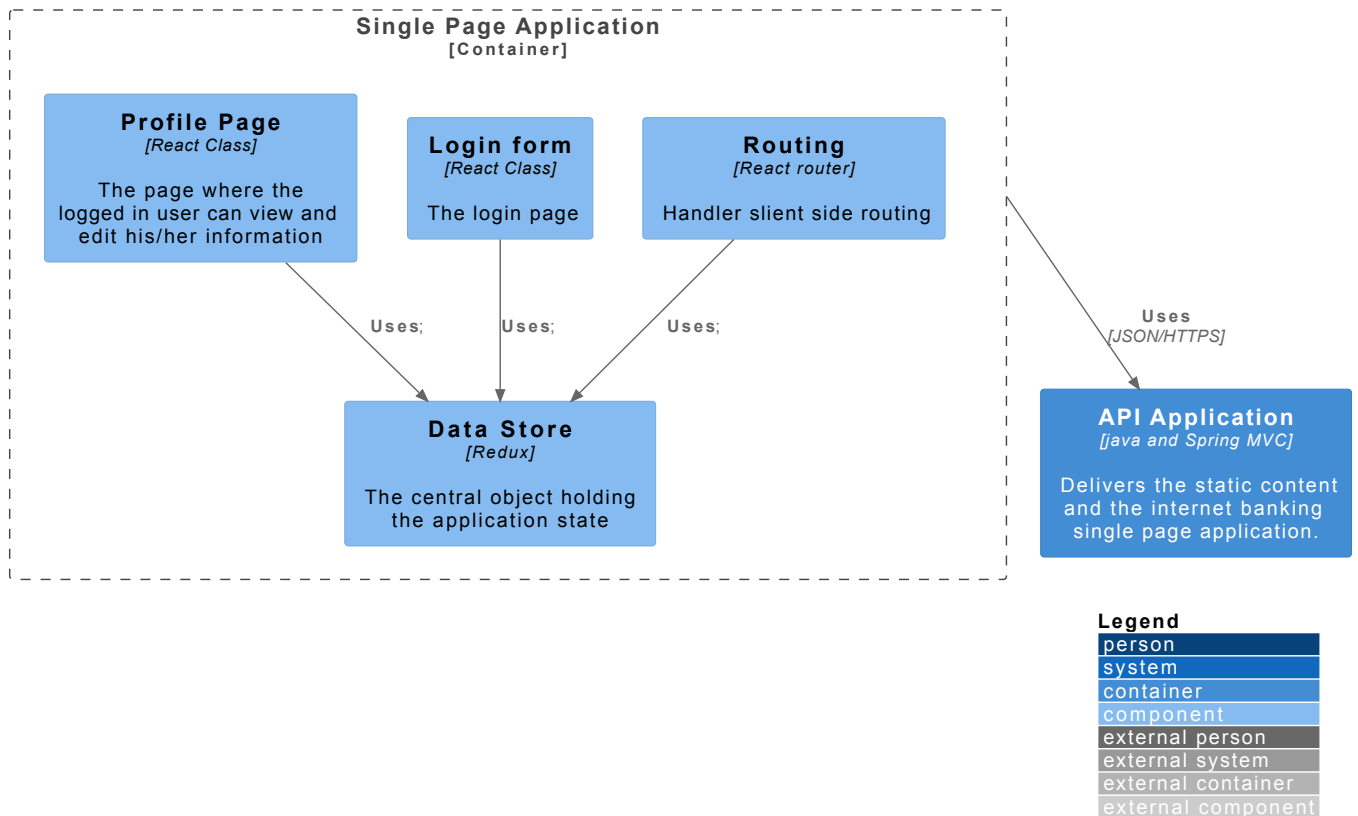
Scope: A single container.

Primary elements: Components within the container in scope. Supporting elements: Containers (within the software system in scope) plus people and software systems directly connected to the components.

Intended audience: Software architects and developers.

Single Page Application

/1 Lead Gen Platform/Single Page Application

Single Page Application
[Container]

Profile Page
[React Class]

The page where the logged in user can view and edit his/her information

Login form
[React Class]

The login page

Routing
[React router]

Handler slient side routing

Uses;   Uses;   Uses;

Uses
[JSON/HTTPS]

Data Store
[Redux]

The central object holding the application state

API Application
[java and Spring MVC]

Delivers the static content and the internet banking single page application.

Legend
person
system
container
component
external person
external system
external container
external component

Level 3: Component diagram

Next you can zoom in and decompose each container further to identify the major structural building blocks and their interactions.

The Component diagram shows how a container is made up of a number of "components", what each of those components are, their responsibilities and the technology/implementation details.

Scope: A single container.

Primary elements: Components within the container in scope. Supporting elements: Containers (within the software system in scope) plus people and software systems directly connected to the components.
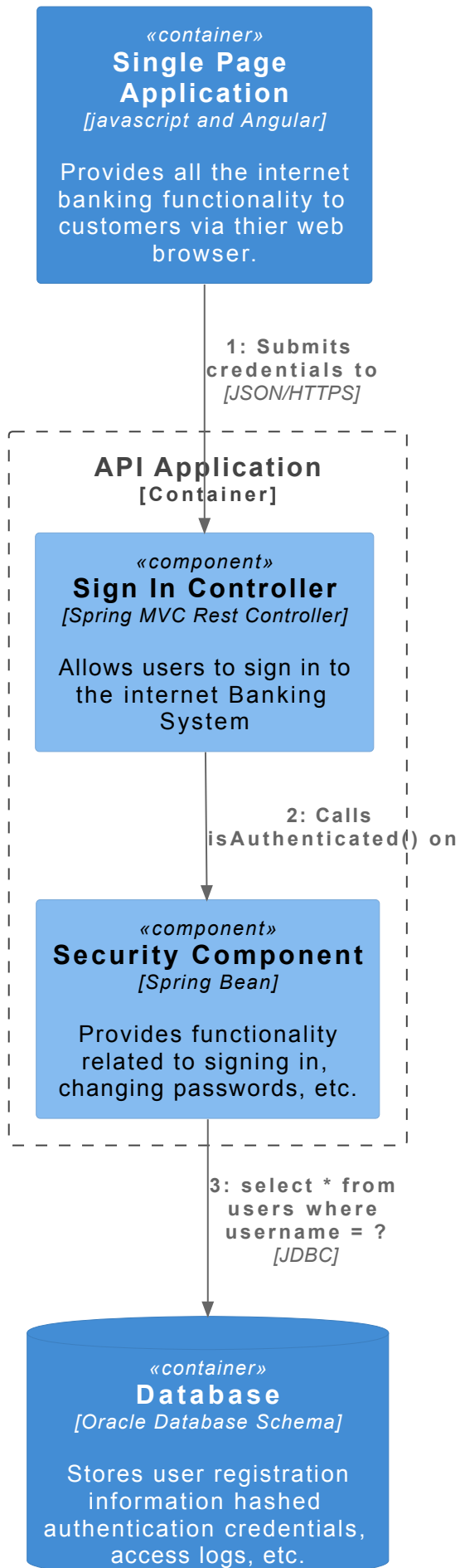
Intended audience: Software architects and developers.

Example of included local image

Dynamic Diagram

/1 Lead Gen Platform/Single Page Application/Dynamic Diagram

«container»
Single Page Application
[javascript and Angular]

Provides all the internet banking functionality to customers via thier web browser.

1: Submits credentials to
[JSON/HTTPS]

API Application
[Container]

«component»
Sign In Controller
[Spring MVC Rest Controller]

Allows users to sign in to the internet Banking System

2: Calls isAuthenticated() on

«component»
Security Component
[Spring Bean]

Provides functionality related to signing in, changing passwords, etc.

3: select * from users where username = ?
[JDBC]

«container»
Database
[Oracle Database Schema]

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

Dynamic diagram

A simple dynamic diagram can be useful when you want to show how elements in a static model collaborate at runtime to implement a user story, use case, feature, etc. This dynamic diagram is based upon a UML communication diagram (previously known as a "UML collaboration diagram"). It is similar to a UML sequence diagram although it allows a free-form arrangement of diagram elements with numbered interactions to indicate ordering.

Scope: An enterprise, software system or container.

Primary and supporting elements: Depends on the diagram scope; enterprise (see System Landscape diagram), software system (see System Context or Container diagrams), container (see Component diagram).

Intended audience: Technical and non-technical people, inside and outside of the software development team.
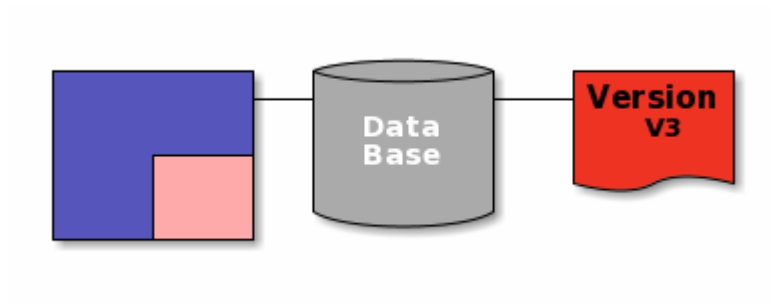
Extended Docs

/1 Lead Gen Platform/Single Page Application/Extended Docs
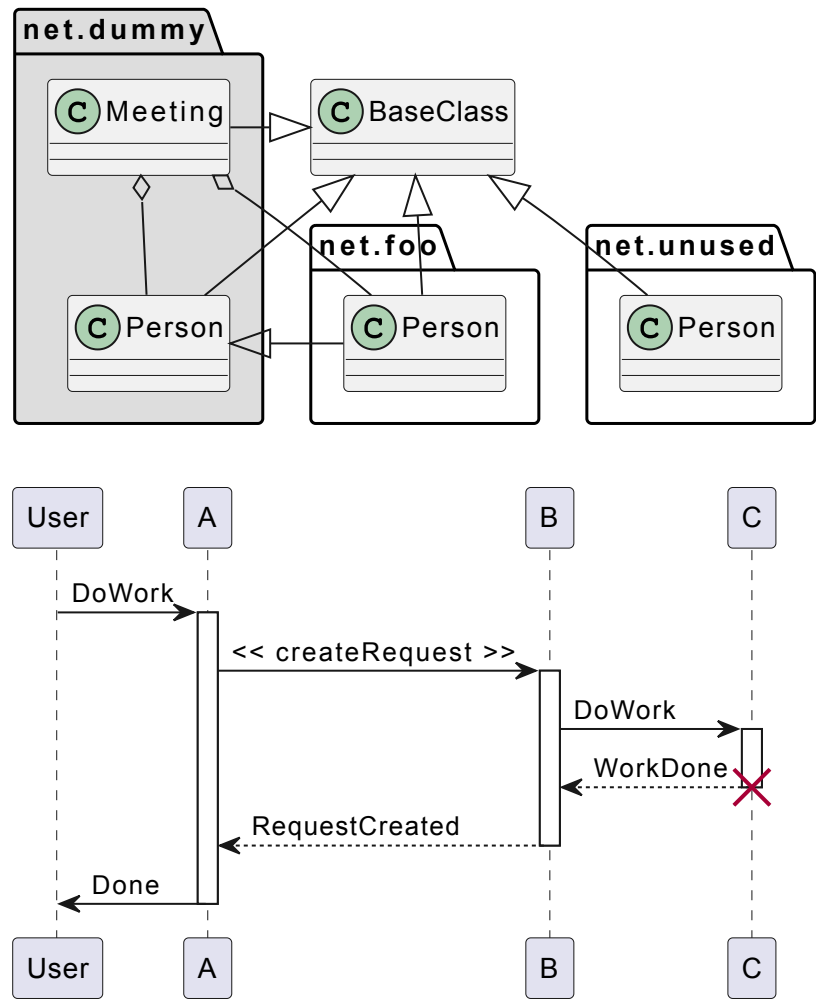
Multiple markdowns can be ordered using `<name>.1.md, <name>.2.md .. <name>.<n>.md`

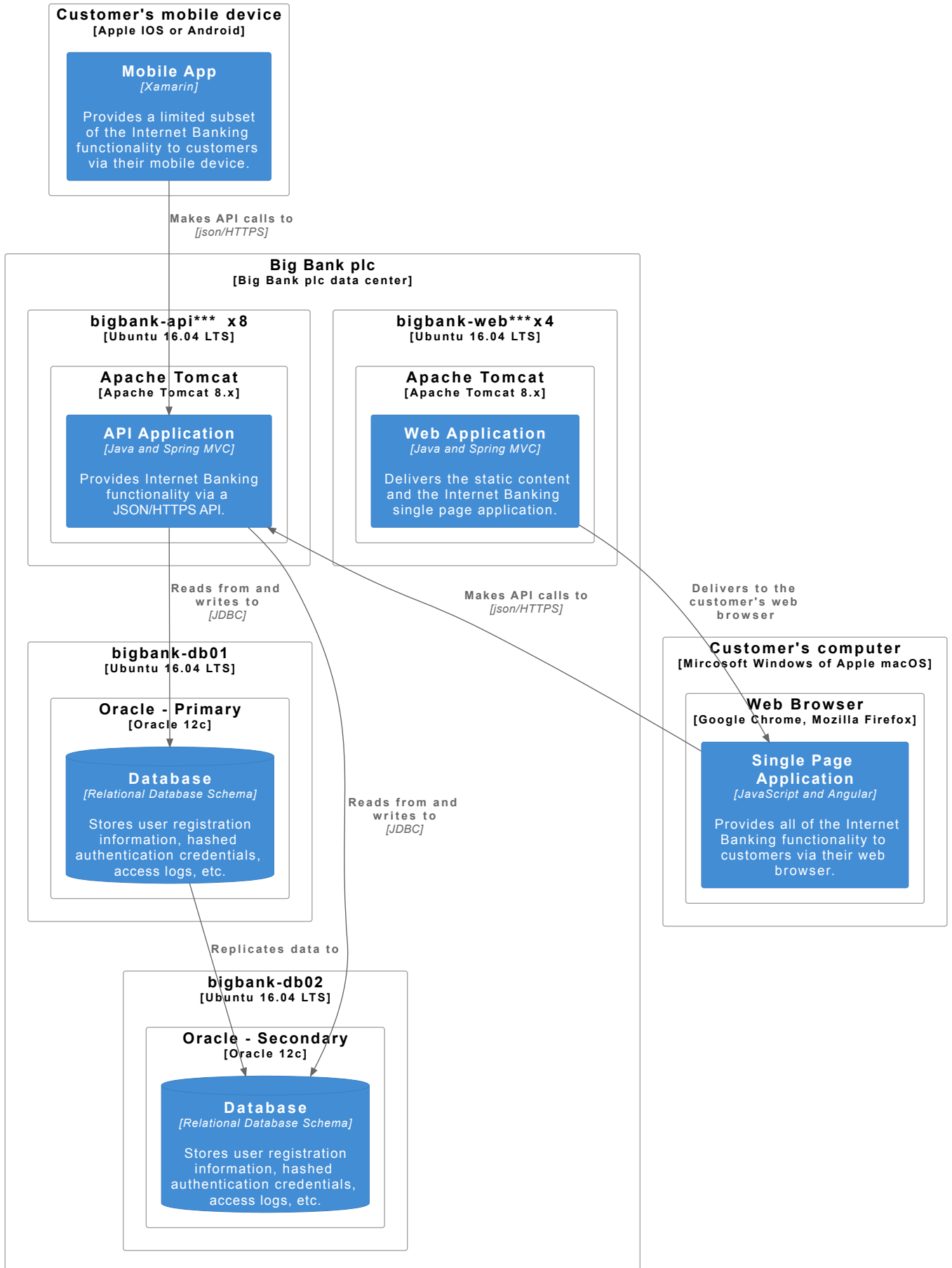You can choose where to place a certain diagram by using `![name](<diagram name>.puml)`

Feel free to add any additional details necesary.

2 Deployment

/2 Deployment

Deployment diagram

A deployment diagram allows you to illustrate how containers in the static model are mapped to infrastructure. This deployment diagram is based upon a UML deployment diagram, although simplified slightly to show the mapping between containers and deployment nodes. A deployment node is something like physical infrastructure (e.g. a physical server or device), virtualised infrastructure (e.g. IaaS, PaaS, a virtual machine), containerised infrastructure (e.g. a Docker container), an execution environment (e.g. a database server, Java EE web/application server, Microsoft IIS), etc. Deployment nodes can be nested.

Scope: A single software system.

Primary elements: Deployment nodes and containers within the software system in scope.

Intended audience: Technical people inside and outside of the software development team; including software architects, developers and operations/support staff.