Architecture Project #1

- Overview: Running App
    - 1 System Context Diagram
        - Container Diagram
            - API App Component
            - Mobile App Component
            - Web App Component
    - 2 Current Deployment Diagram
    - 3 One Month Plan
    - 4 Long Term Plan

Overview: Running App

Software Architecture Course

Project Part 1

The goal of the running application is to allow users to track their running habits and get motivated by visualizing their progress.

Unfortunately, customers have been complaining about some problems, and our business is losing customers to the competition.

In this document we have devised a one month plan to tackle the three immediate pain points with an improve in our software architecture.

Primary Goals

- the app is not responsive during a run, especially trying to start and stop it
- some customers mentioned that their runs are disappearing from the web dashboard
- when customers finish a run, it takes more than one day to view on the dashboard

Secondary Goals

- Reduce cost of total cloud account

Scope: Our Running application System and their interactions

Intended audience: CEO, CTO, and CFO are the primary audience for this presentation. Eventually it will be extended to the development team in order to provide clarity on the goals to follow ahead.

Assumptions:

- The dev team is capable to work on the code and provide required refactoring.
- The devops team is capable to work and manage our cloud provider
- The QA team can actually reproduce the issues and test the app is running.

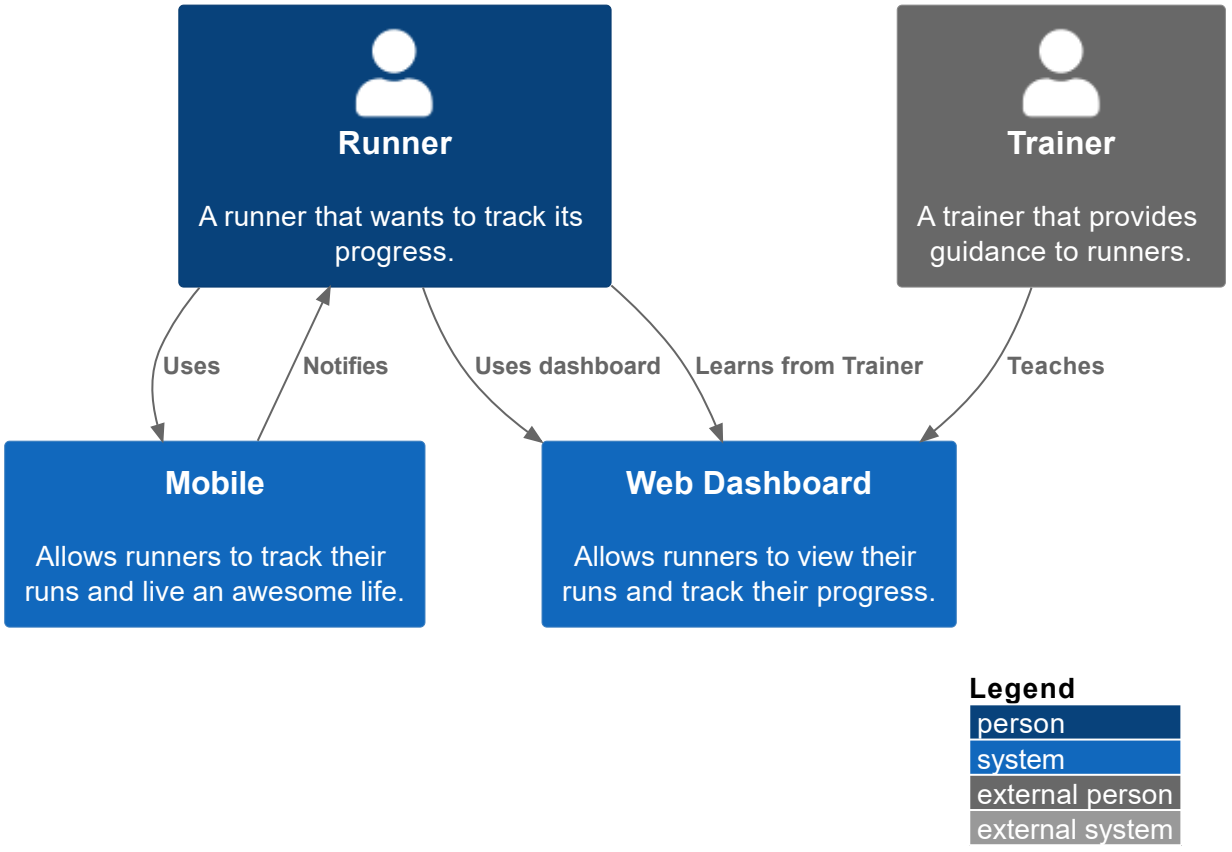
- No staff change is likely to happen on the suggested time frame.
- We will utilize a bit over half the IT team separated at 3 teams of 6 devs/devops/qa, for 2 consecutive sprints.
- The allocation of staff is agreed upon and no changes to the plan will be done on the next month

Limitations: We will not be generating code diagrams for this project. Instead we will focus on the Deployment Diagram, which will map our current platform to the physical cloud architecture, which will help us visualize better the high level architectural changes.

Tools: This document utilizes C4 Diagrams and (Markdown files)[https://www.markdownguide.org/] to generate all the pages using (VSCode)[https://code.visualstudio.com/], the (PlantUML plugin) [https://marketplace.visualstudio.com/items?itemName=jebbs.plantuml] and the (nodejs c4builder project) [https://adrianvlupu.github.io/C4-Builder/#/].

1 System Context Diagram

```
\1 System Context Diagram
```



Level 1: System Context

The system context diagram represents a high-level overview of the interactions of the main components of the current architecture.

Primary Elements

There are currently one main user and two main components on this application. The primary elements are as follow:

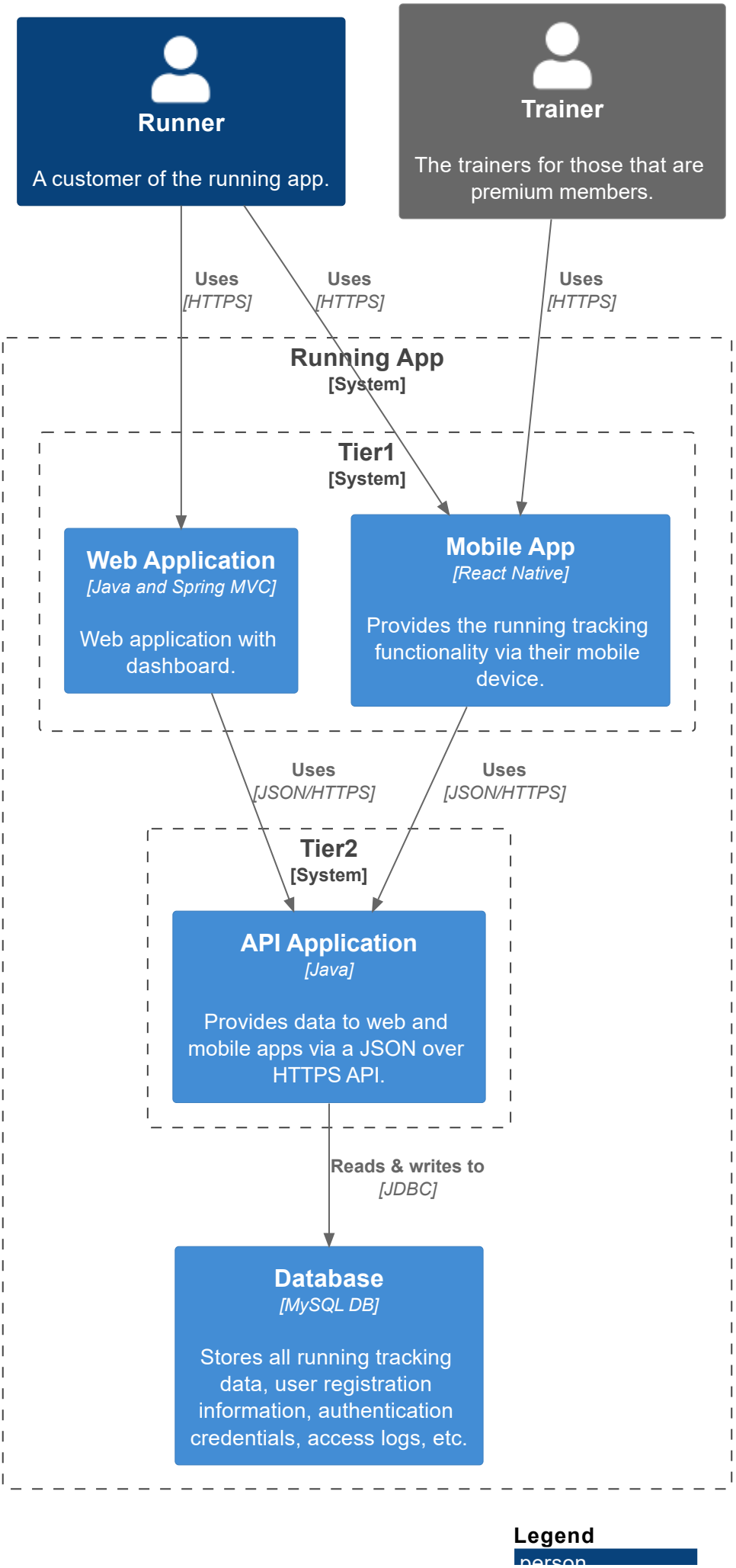
- Our main users are runners that want to track and improve their running habits. These persons range from teens to seniors of all genders.
- The first main component is the mobile application, designed to allow users to select a running program, track their runs with GPS data , count calories of each run , and enhance the running experience with a music playlist.
- The second main component is a web dashboard that is designed to give users an overview of their runs and track their progress over time. They can also manage their accounts and purchase advanced plans with live trainer support.

Secondary Elements

- Trainers are professional runners that provide services guiding our runners into their journey.

Container Diagram

`\1 System Context Diagram\Container Diagram`


Runner
A customer of the running app.
Trainer
The trainers for those that are premium members.
Uses
[HTTPS]
Uses
[HTTPS]
Uses
[HTTPS]
Running App
[System]
Tier1
[System]
Web Application
[Java and Spring MVC]
Web application with dashboard.
Mobile App
[React Native]
Provides the running tracking functionality via their mobile device.
Uses
[JSON/HTTPS]
Uses
[JSON/HTTPS]
Tier2
[System]
API Application
[Java]
Provides data to web and mobile apps via a JSON over HTTPS API.
Reads & writes to
[JDBC]
Database
[MySQL DB]
Stores all running tracking data, user registration information, authentication credentials, access logs, etc.
Legend
person

person
system
container
external person
external system
external container

Level 2: Container diagram

Each part of our application is a two tier app, which means we have a code layer for presentation and one for business logic. The presentation layer on mobile is coded on native code and fetches data from the API via RESTful calls.

The same pattern is observed with our Web Dashboard application, which retrieve data from the API layer and serve the HTML pages to the clients.

API App Component

`\1 System Context Diagram\Container Diagram\API App Component`

Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on https://plantuml.com

(If you use this software, you accept its license)
(Details by typing `license` keyword)

PlantUML 1.2022.3

<b>This version of PlantUML is 684 days old, so you should
<b>consider upgrading from https://plantuml.com/download

[From string (line 2)]

@startuml
!include ../../_C4.puml
cannot include ../../_C4.puml

Level 3: Component diagram

The API layer supports the following RESTful endpoints over HTTPS:

For the mobile app

- /login
- /logout
- /getRunTypes
- /startRun

- /stopRun
- /estimatedCaloryBurned
- /addUserInfo

For the web dashboard:

- /manageAccount
- /getHistoryData
- /viewPlans
- /upgradePlan
- /cancelPlan
- /bookTrainer
- /cancelBooking

Mobile App Component

`\1 System Context Diagram\Container Diagram\Mobile App Component`

Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on https://plantuml.com

(If you use this software, you accept its license)
(Details by typing `license` keyword)

PlantUML 1.2022.3

<b>This version of PlantUML is 684 days old, so you should
<b>consider upgrading from https://plantuml.com/download

[From string (line 2)]

@startuml
!include ../../_C4.puml
cannot include ../../_C4.puml

Level 3: Component diagram

Web App Component

`\1 System Context Diagram\Container Diagram\Web App Component`

Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on https://plantuml.com

(If you use this software, you accept its license)
(Details by typing `license` keyword)

PlantUML 1.2022.3

<b>This version of PlantUML is 684 days old, so you should
<b>consider upgrading from https://plantuml.com/download

[From string (line 2)]

@startuml
!include ../../_C4.puml
cannot include ../../_C4.puml

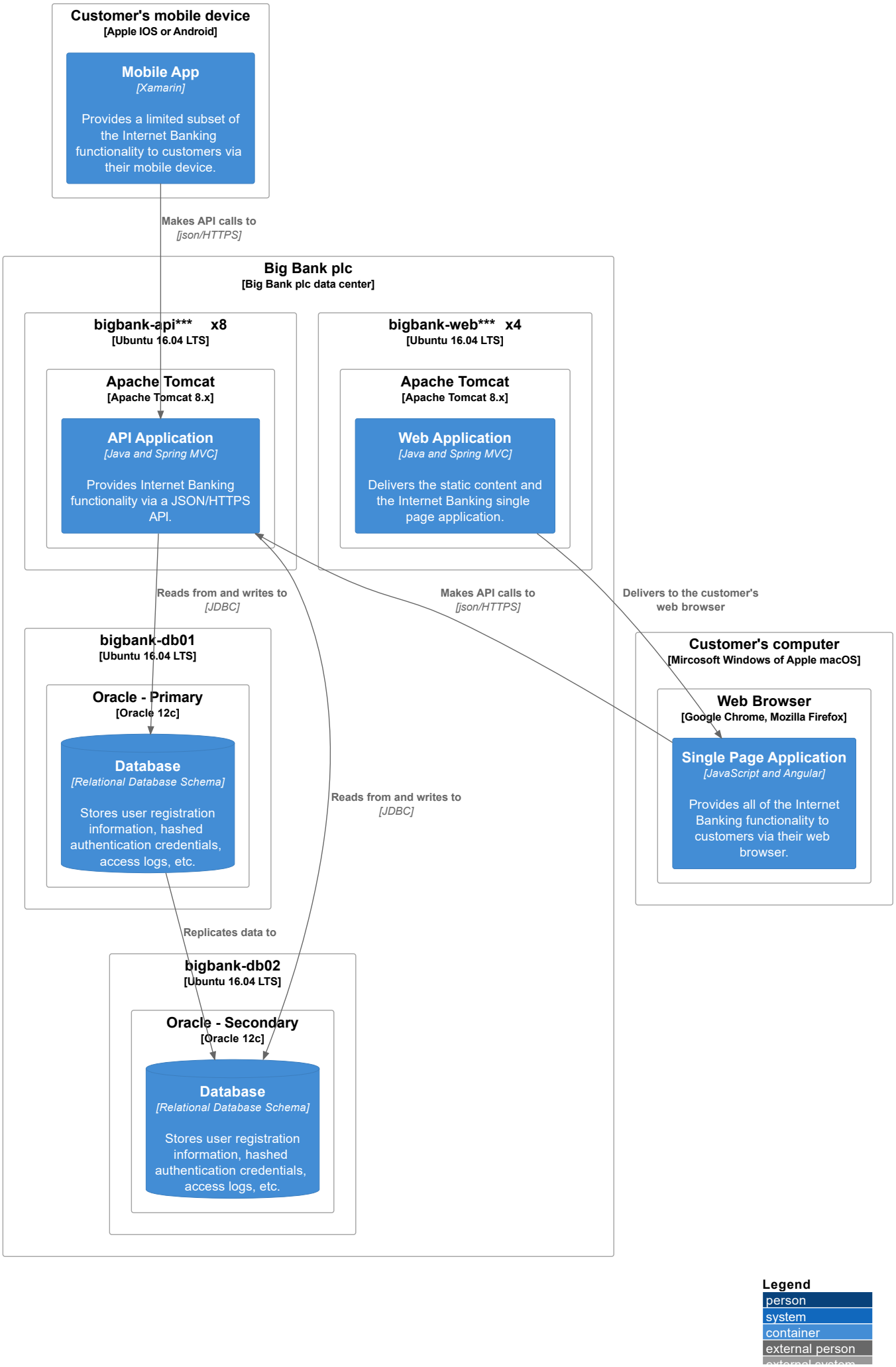
Level 3: Component diagram

The current Web Application is written in Java and queries all data from a single MySQL instance.

This is slowing our application since all reads are competing with the writes.

2 Current Deployment Diagram

```
\2 Current Deployment Diagram
```


Customer's mobile device
[Apple IOS or Android]
Mobile App
[Xamarin]
Provides a limited subset of the Internet Banking functionality to customers via their mobile device.
Makes API calls to
[json/HTTPS]
Big Bank plc
[Big Bank plc data center]
bigbank-api*** x8
[Ubuntu 16.04 LTS]
Apache Tomcat
[Apache Tomcat 8.x]
API Application
[Java and Spring MVC]
Provides Internet Banking functionality via a JSON/HTTPS API.
bigbank-web*** x4
[Ubuntu 16.04 LTS]
Apache Tomcat
[Apache Tomcat 8.x]
Web Application
[Java and Spring MVC]
Delivers the static content and the Internet Banking single page application.
Reads from and writes to
[JDBC]
Makes API calls to
[json/HTTPS]
Delivers to the customer's web browser
bigbank-db01
[Ubuntu 16.04 LTS]
Oracle - Primary
[Oracle 12c]
Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Customer's computer
[Mircosoft Windows of Apple macOS]
Web Browser
[Google Chrome, Mozilla Firefox]
Single Page Application
[JavaScript and Angular]
Provides all of the Internet Banking functionality to customers via their web browser.
Reads from and writes to
[JDBC]
Replicates data to
bigbank-db02
[Ubuntu 16.04 LTS]
Oracle - Secondary
[Oracle 12c]
Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Legend
person
system
container
external person

external system
external container

Deployment diagram

This section we view how containers in the static model are mapped to infrastructure.

Currently we have the mobile app installed on the client directly fetching data from the API server. The installation files are served to the users phone by the respective platform marketplaces.

Our web server is already on Azure cloud, and its deployed on a single large scale VM using a web server. The code also fetches data from the API layer.

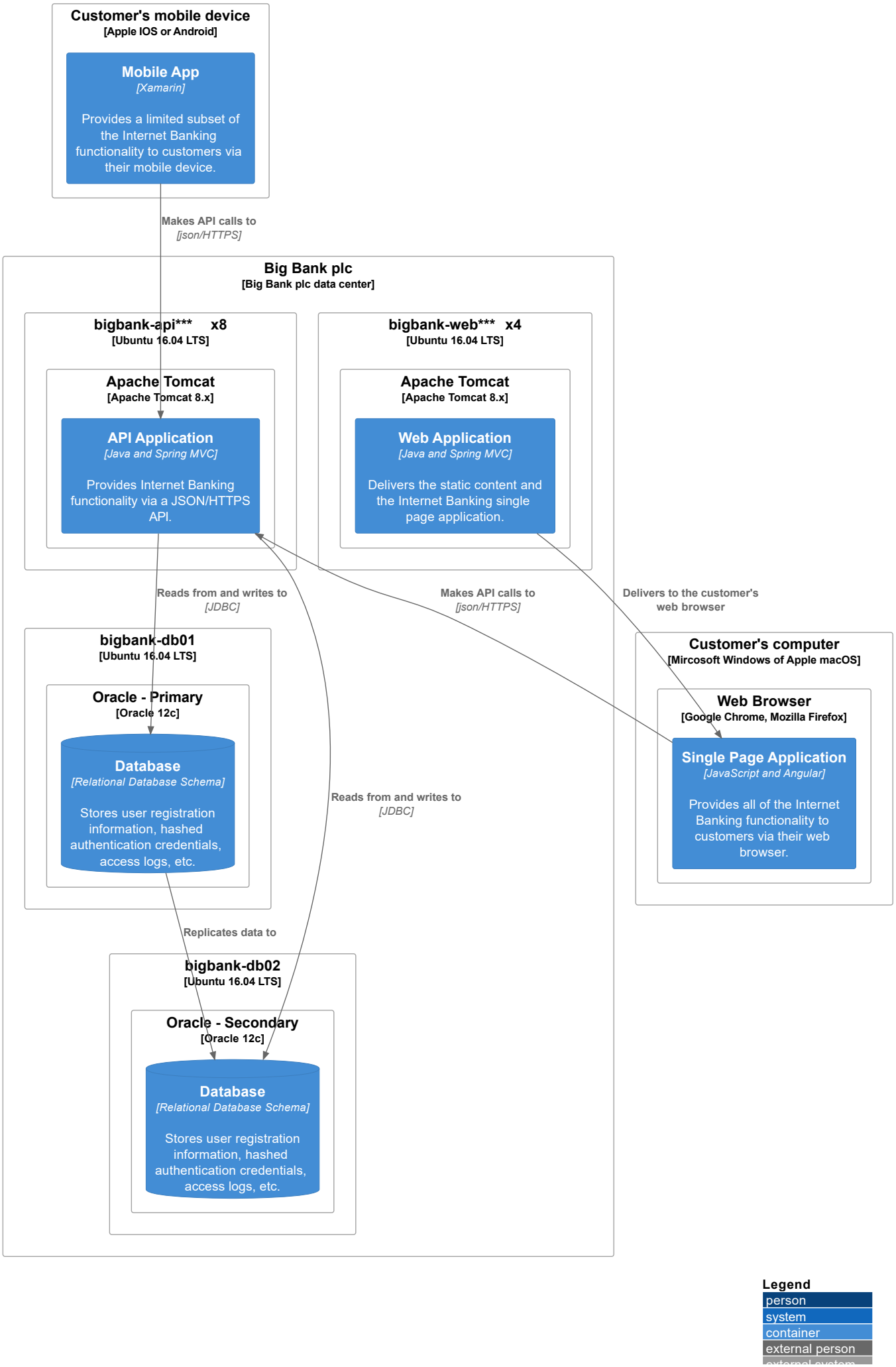
The API layer is on a second large VM instance.

This layer reads and writes all data from a single instance MySQL server.

The Database is a single MySQL database and is managed by our cloud services provider.

3 One Month Plan

```
\3 One Month Plan
```


Customer's mobile device
[Apple IOS or Android]
Mobile App
[Xamarin]
Provides a limited subset of the Internet Banking functionality to customers via their mobile device.
Makes API calls to
[json/HTTPS]
Big Bank plc
[Big Bank plc data center]
bigbank-api*** x8
[Ubuntu 16.04 LTS]
Apache Tomcat
[Apache Tomcat 8.x]
API Application
[Java and Spring MVC]
Provides Internet Banking functionality via a JSON/HTTPS API.
bigbank-web*** x4
[Ubuntu 16.04 LTS]
Apache Tomcat
[Apache Tomcat 8.x]
Web Application
[Java and Spring MVC]
Delivers the static content and the Internet Banking single page application.
Reads from and writes to
[JDBC]
Makes API calls to
[json/HTTPS]
Delivers to the customer's web browser
bigbank-db01
[Ubuntu 16.04 LTS]
Oracle - Primary
[Oracle 12c]
Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Customer's computer
[Mircosoft Windows of Apple macOS]
Web Browser
[Google Chrome, Mozilla Firefox]
Single Page Application
[JavaScript and Angular]
Provides all of the Internet Banking functionality to customers via their web browser.
Reads from and writes to
[JDBC]
Replicates data to
bigbank-db02
[Ubuntu 16.04 LTS]
Oracle - Secondary
[Oracle 12c]
Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Legend
person
system
container
external person

external system
external container

One Month Attack Plan

To improve the user experience for our consumers, we will do the following modifications:

In order to maximize the elasticity of the providers, we will convert all API layer into Functions (lambdas). This will allow us to only use the resources that are actually necessary for each request instead of guessing the size of the VM to host our app.

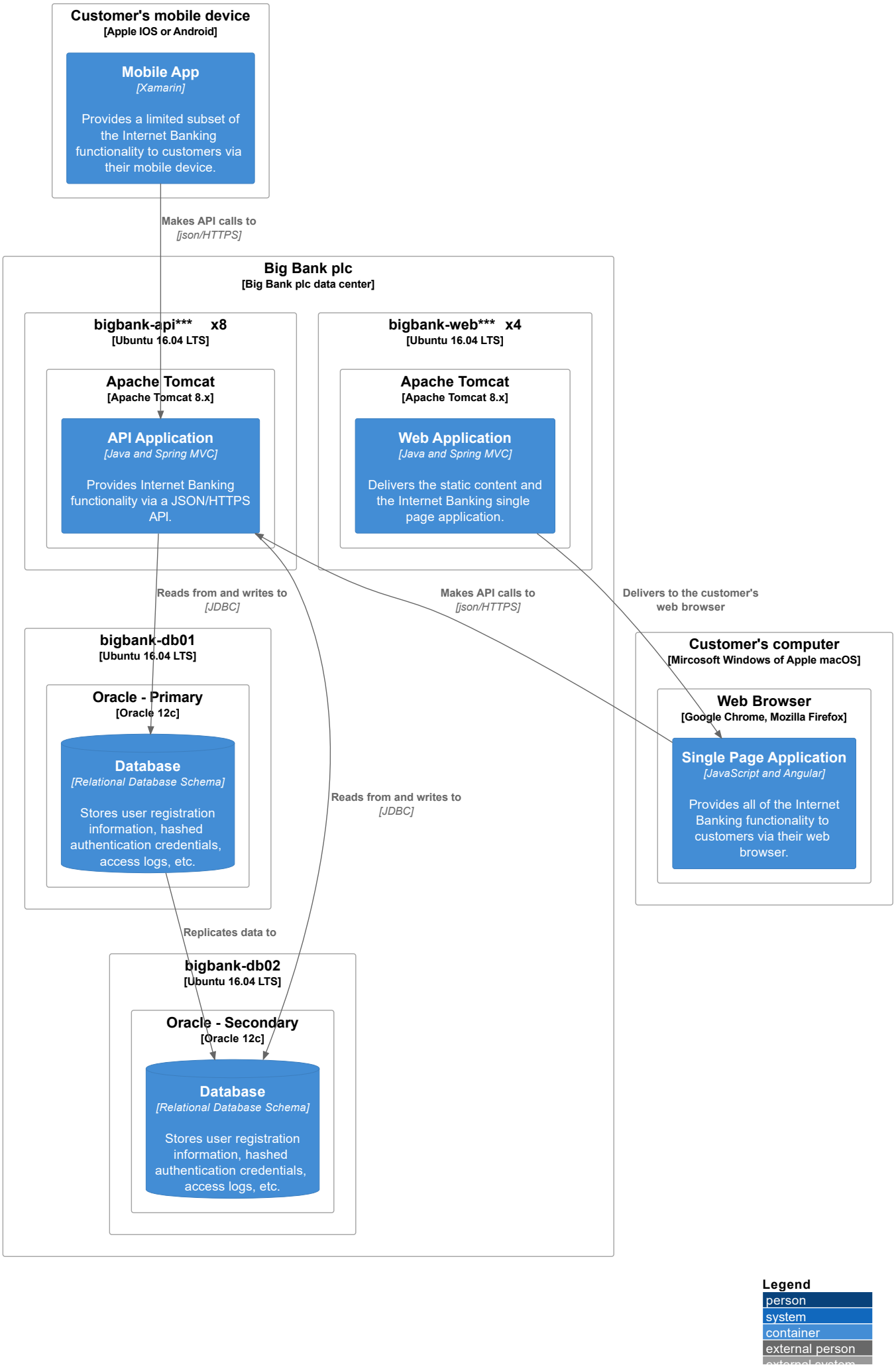
"Azure Functions is a serverless solution that allows you to write less code, maintain less infrastructure, and save on costs. Instead of worrying about deploying and maintaining servers, the cloud infrastructure provides all the up-to-date resources needed to keep your applications running."(https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview?pivots=programming-language-csharp)

Also, for the single MySQL database instance, we will create a second instance that will be focused on write only coming from the mobile apps, such as tracking data that is provided on each run. The original MySQL instance will

Although it seems that we will be utilizing more of the cloud provider, in fact we will be able to better utilize resources by scaling the app up when we need it most, such as 7AM-8AM, 12PM-01PM, 07PM-08PM, and scale down when there are almost no usage, such as during the night, from 11PM-06AM. Remember that for the time being our customer base is only the USA.

4 Long Term Plan

```
\4 Long Term Plan
```


Customer's mobile device
[Apple IOS or Android]
Mobile App
[Xamarin]
Provides a limited subset of the Internet Banking functionality to customers via their mobile device.
Makes API calls to
[json/HTTPS]
Big Bank plc
[Big Bank plc data center]
bigbank-api*** x8
[Ubuntu 16.04 LTS]
Apache Tomcat
[Apache Tomcat 8.x]
API Application
[Java and Spring MVC]
Provides Internet Banking functionality via a JSON/HTTPS API.
bigbank-web*** x4
[Ubuntu 16.04 LTS]
Apache Tomcat
[Apache Tomcat 8.x]
Web Application
[Java and Spring MVC]
Delivers the static content and the Internet Banking single page application.
Reads from and writes to
[JDBC]
Makes API calls to
[json/HTTPS]
Delivers to the customer's web browser
bigbank-db01
[Ubuntu 16.04 LTS]
Oracle - Primary
[Oracle 12c]
Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Customer's computer
[Mircosoft Windows of Apple macOS]
Web Browser
[Google Chrome, Mozilla Firefox]
Single Page Application
[JavaScript and Angular]
Provides all of the Internet Banking functionality to customers via their web browser.
Reads from and writes to
[JDBC]
Replicates data to
bigbank-db02
[Ubuntu 16.04 LTS]
Oracle - Secondary
[Oracle 12c]
Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Legend
person
system
container
external person

external system
external container

Deployment diagram

This section we view how containers in the static model are mapped to infrastructure.

Currently we have the mobile app installed on the client directly fetching data from the API server. The installation files are served to the users phone by the respective platform marketplaces.

Our web server is already on Azure cloud, and its deployed on a single VM using IIS as a web server. The code also fetches data from the API layer.

The API layer reads and writes all data from a single instance MySQL server.

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.