Software Architecture Document (SAD)

Project: Case study Enterprise Architecture & C4 and UML modelling

Project team: BBB-BC

Team: Joeri Berman (2834499)

Luc Urlings (3071081)

Stijn Baltessen (3064565)

Wouter Vandenboorn (3099571)

Floris Feddema (3188256)

Davy de Haas (3099695)

Opdrachtgever: Bartosz Paszkowski

Version: 0.2

Version Date: 17 februari 2020

Status: Concept

# Document Revision

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Changes** | **Author** | **Date** |
| 0.1 | Setup for the semester 6 architecture case study | Wouter Vandenboorn | 17 feb 2020 |
| 0.2 | Added Introduction and c4 models, database and api descriptions. | Wouter Vandenboorn,  Stijn Baltessen | 21 feb 2020 |

# Table of Contents

[**Document Revision**](#_bu2l4jqkimrr) **1**

[**Table of Contents**](#_lbbi792fwpmm) **2**

[**Introduction**](#_8anxnj4hrgfu) **3**

[**Architecture Design**](#_88liwgvquuyy) **4**

[**System Context (C1)**](#_v5og0y2f4s6s) **5**

[**Container Diagram (C2)**](#_ohe2bvfw1sv0) **6**

[**Component Diagrams (C3)**](#_tuzx6llajbs6) **7**

[**Code and Sequence Diagrams (C4)**](#_7neh7frxhps9) **8**

[**Database(s) Structure**](#_72aqxh5tg6xu) **9**

[**Specification of API’s**](#_61kekx28grzl) **10**

[**References**](#_w3ca9631t95g) **11**

# 

# Introduction

This document is meant as an example/template for a software architecture document. We will provide examples for the models for each chapter why we need them and why they are useful. It is used as a guideline for the architecture designs for the project and what choices have been made for the project design.

In the introduction you should describe the application. And the purpose of this document. For example why it is necessary to create a software architecture document before starting to work on the actual code of the project.

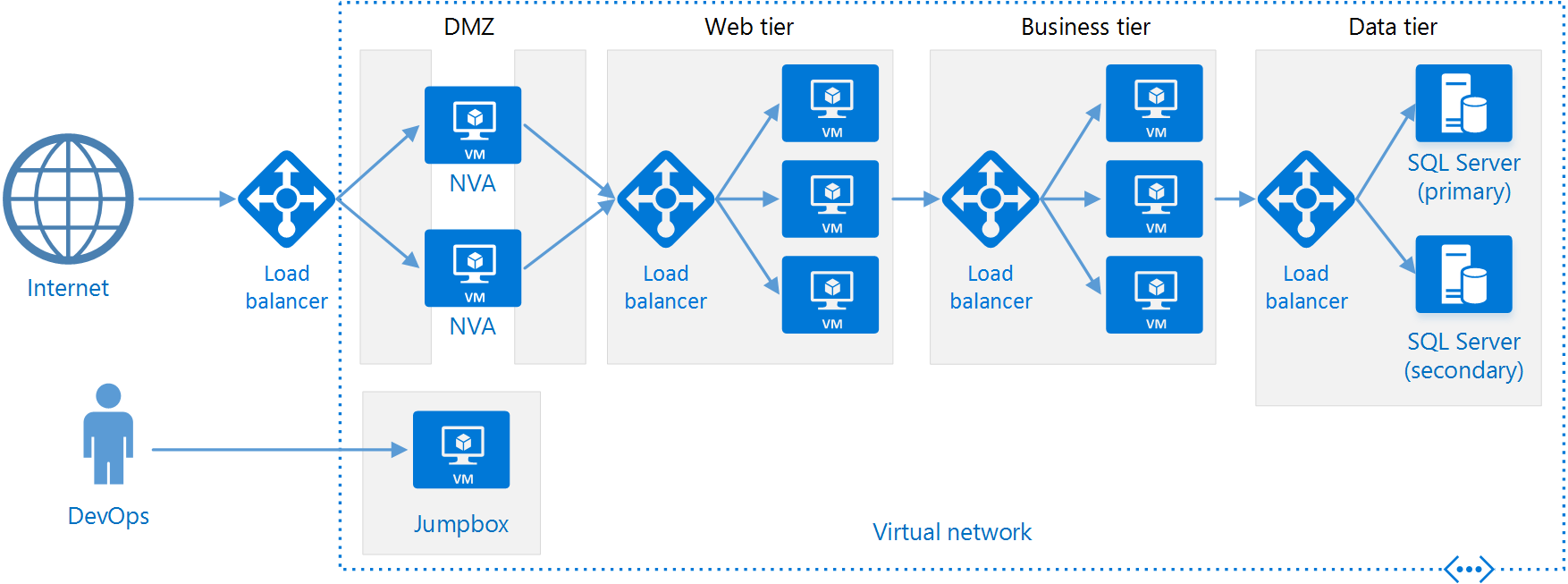
# 

# Architecture Design

It is important to use a structured way of creating applications. To get a wide overall view of the application before you start creating other diagrams you should make a decision on what type of architecture you should use.

An architecture style is a guideline of the structure of the project. If you abide by it’s guidelines it will provide you with a structured approach to the project. Make sure you know the advantages and disadvantages of architecture styles before making a choice so that you can make an educated guess on which type fits best for a specific type of project.

For more information on architecture styles and some examples you can take a look at the following research document : ([Architecture Styles Research](https://drive.google.com/open?id=1Je4vlQqV0N7gu2JzeHvuaON4T20jHRzmweBbOaTjNYU) ).

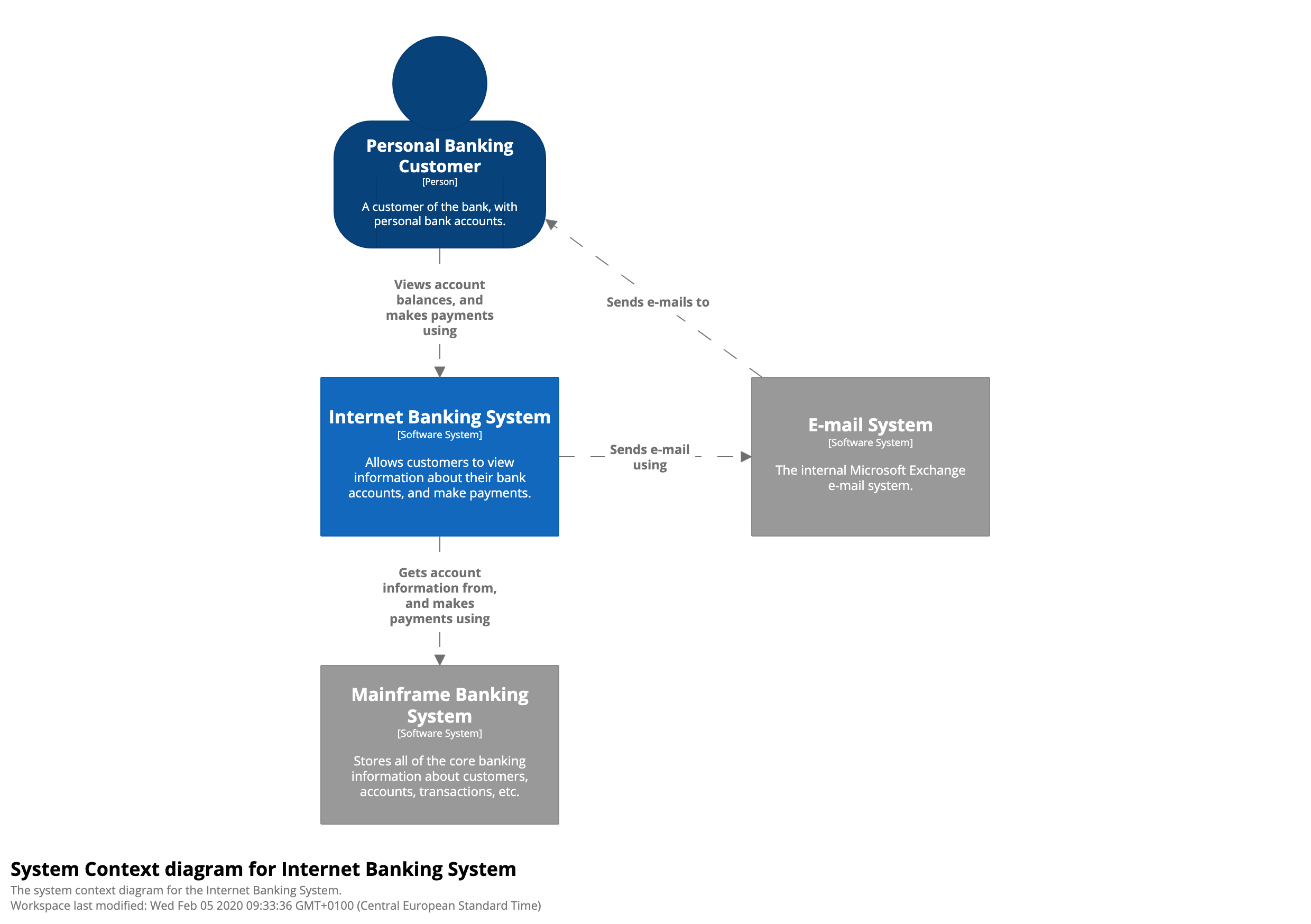


# System Context (C1)

In this chapter the application will be discussed with a system context diagram. This allows you to take a step back and see the big picture.

The way you do this is place your system in the middle in blue, put the systems and users that interact with your system around it. Indicate what the interaction is that is performed. This diagram is meant for technical people as non-technical people.

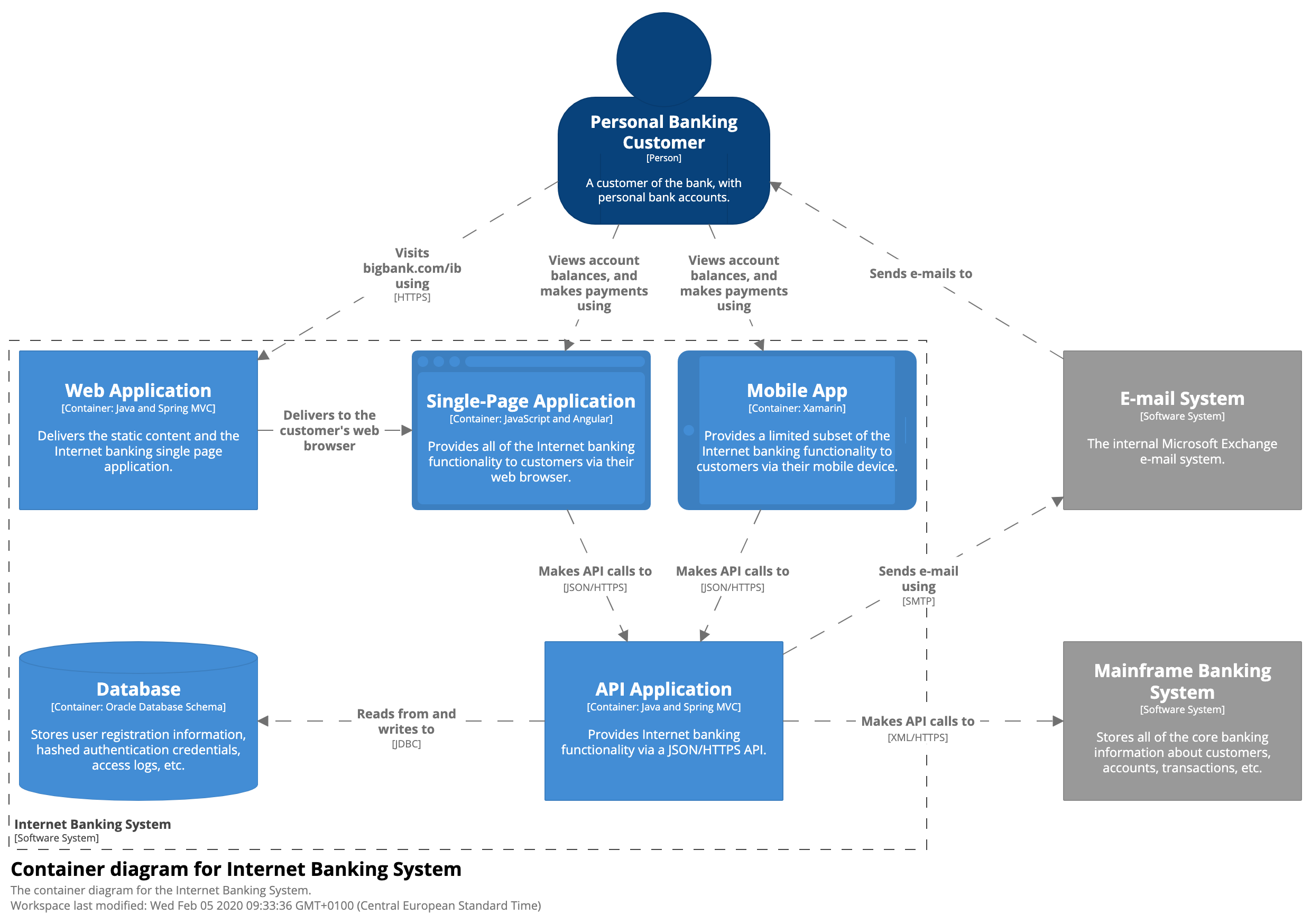
This is a great diagram to show the context of a single software system.

**

# 

# Container Diagram (C2)

A container diagram shows the high level shape of the software architecture, how responsibilities are distributed and what major technology choices are made. It also describes the communication between containers, which container can communicate with who and what kind of data can the send to each other. It doesn’t go into too much detail, but it should provide a clear understanding of what a user can do and how actions will be handled by the entire project.

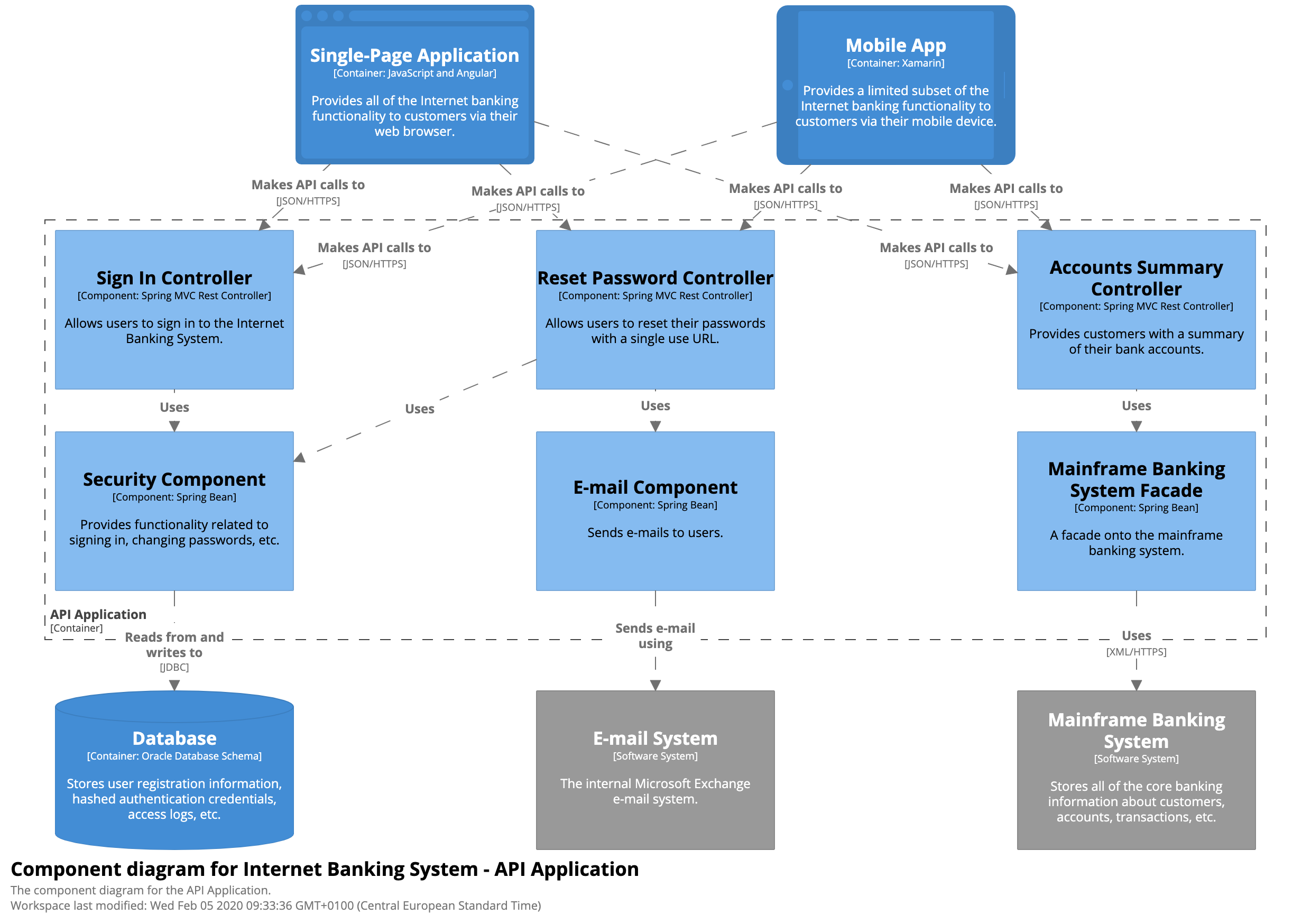


# 

# Component Diagrams (C3)

In the Component Diagram you can zoom in to a single container in this case the API Application of previous diagram.

In this picture you will find the different components that one container has. How they communicate with one another. And for what external purposes these components are used (The grey blocks.).



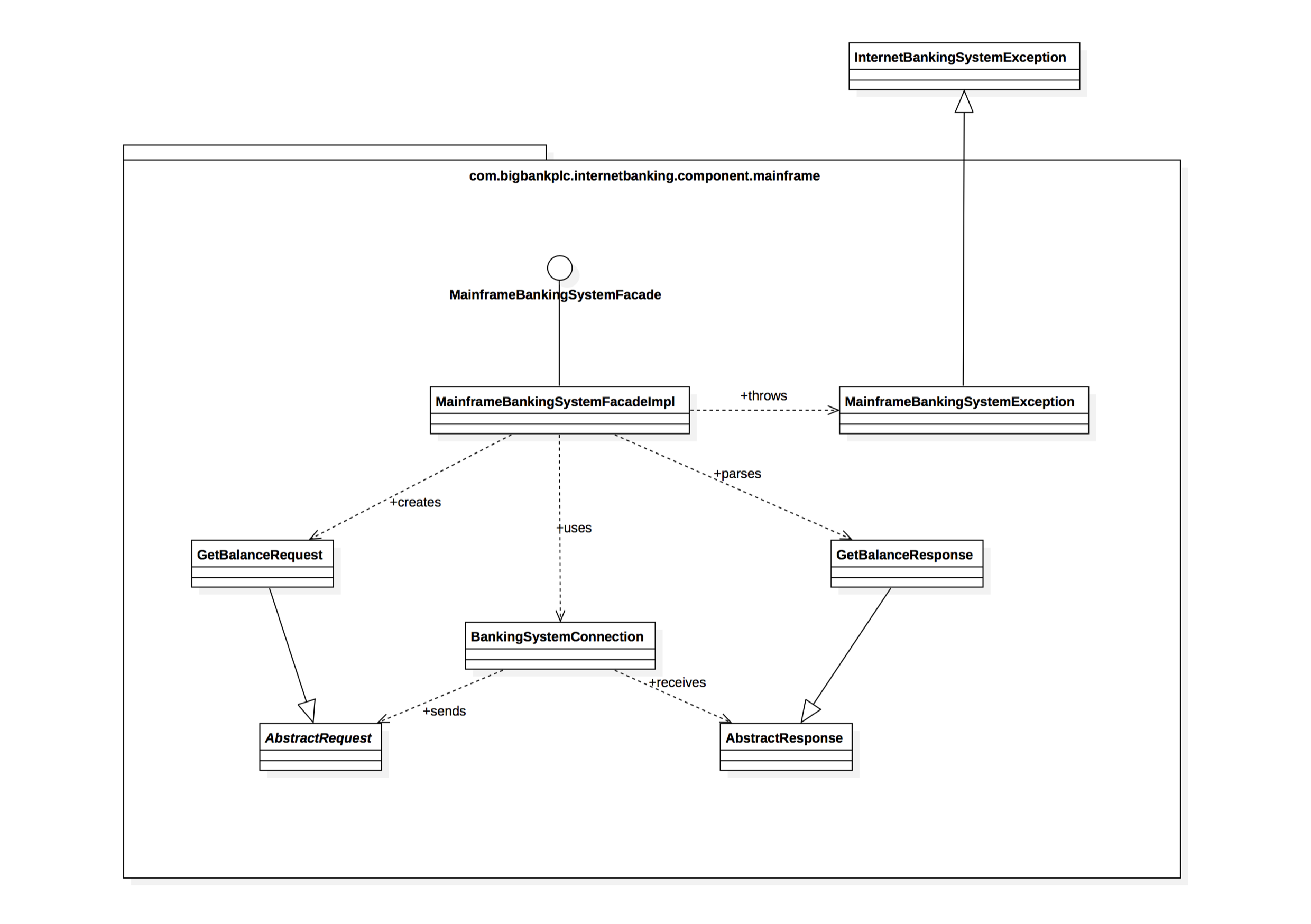
# 

# Code and Sequence Diagrams (C4)

C4 goes into a lot more detail than the previous models that we created. In this chapter we will describe code functionalities in detail. We do this by means of an UML class diagram, Sequence diagrams, entity relationship diagrams and more similar models depending on the complexity and scope of the application. These models are meant to create a deeper and clearer understanding of each component and how it is translated and implemented into code.

Because of the in depth view of each component and the change in requirements for it’s complexity in every application this chapter can look very different based on the needs of a project. It is not set in stone which diagrams are used and choices on which are implemented can be made by the team creating the models and/or the application.

It is recommended to not create a model as in depth as c4 for every feature of the application as it would be too much work, and it’s not always needed for every feature. Instead try and focus on the largest and most important features of the application to model in depth so everyone can get a better understanding of the complexity of these features.



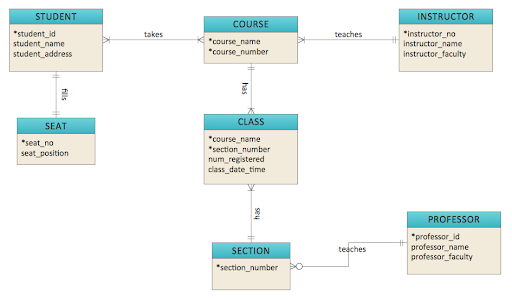
# 

# Database(s) Structure

The structure that your database(s) will use to keep the data of your application persistent.

Use the UML Notation for it. You could do it in ERD or full database diagrams.

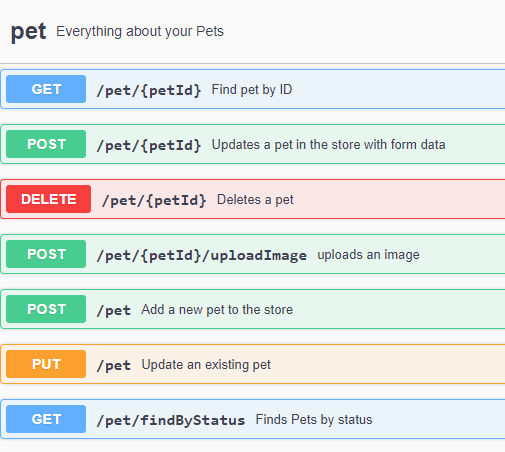
example:



# 

# Specification of API’s

In the case you are using API’s you will want to specify what calls can be made and with what perimeters. And how they will return information towards you and in what kind of format.



example OpenAPI, formerly known as swagger.io

# References

* <https://c4model.com/>
* <https://fhict.instructure.com/>
* <https://swagger.io/>