C4 Model

Context | Container | Component | Code

Introduction

- The C4 model helps software development teams easily describe and communicate software architecture. It allows them to create detailed "maps" of their code, similar to how you would use Google Maps to zoom in and out of an area.

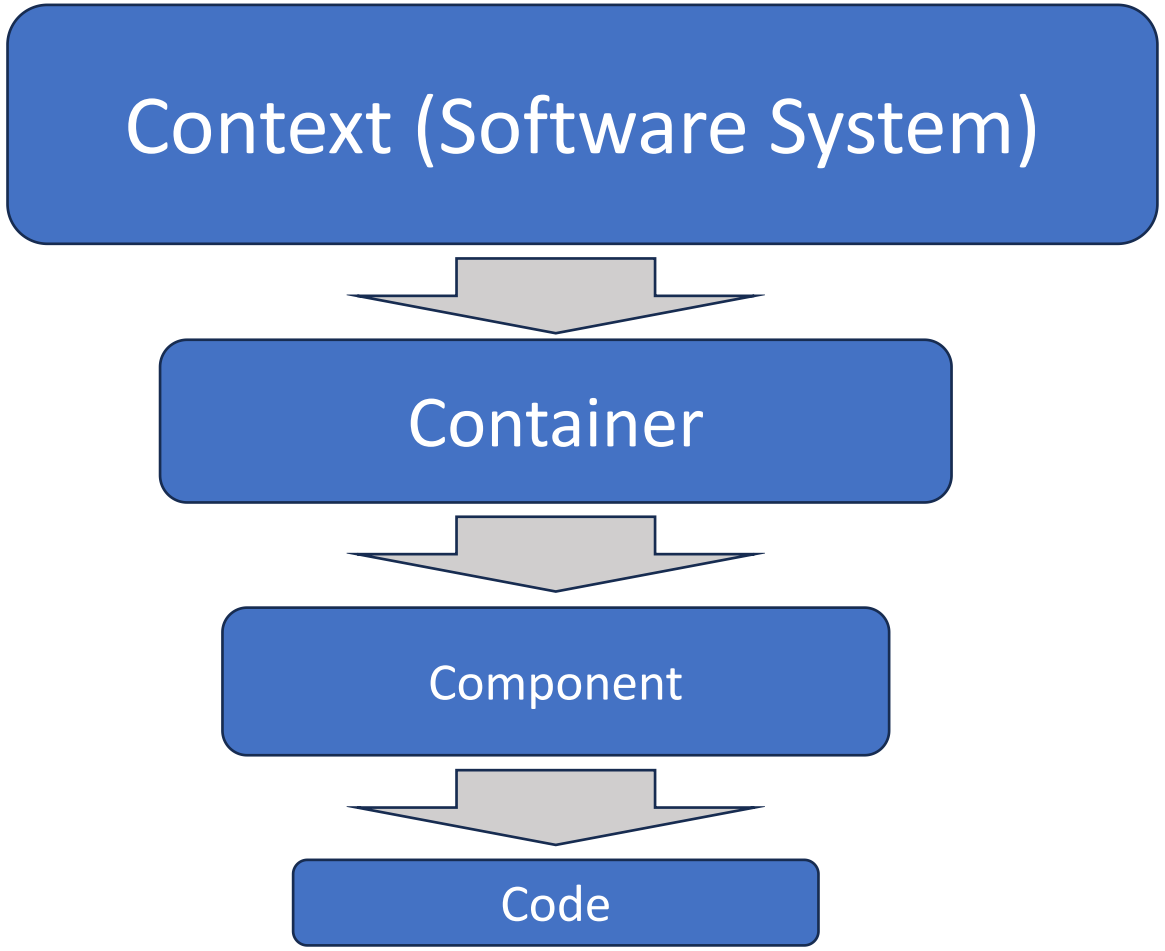
- This approach is useful for both planning new designs and documenting existing code.

Introduction

- The C4 model uses an "abstraction-first" method to diagram software architecture.

- You don't need to use all 4 diagram levels—just the ones that add value. For many teams, the System Context and Container diagrams are enough.

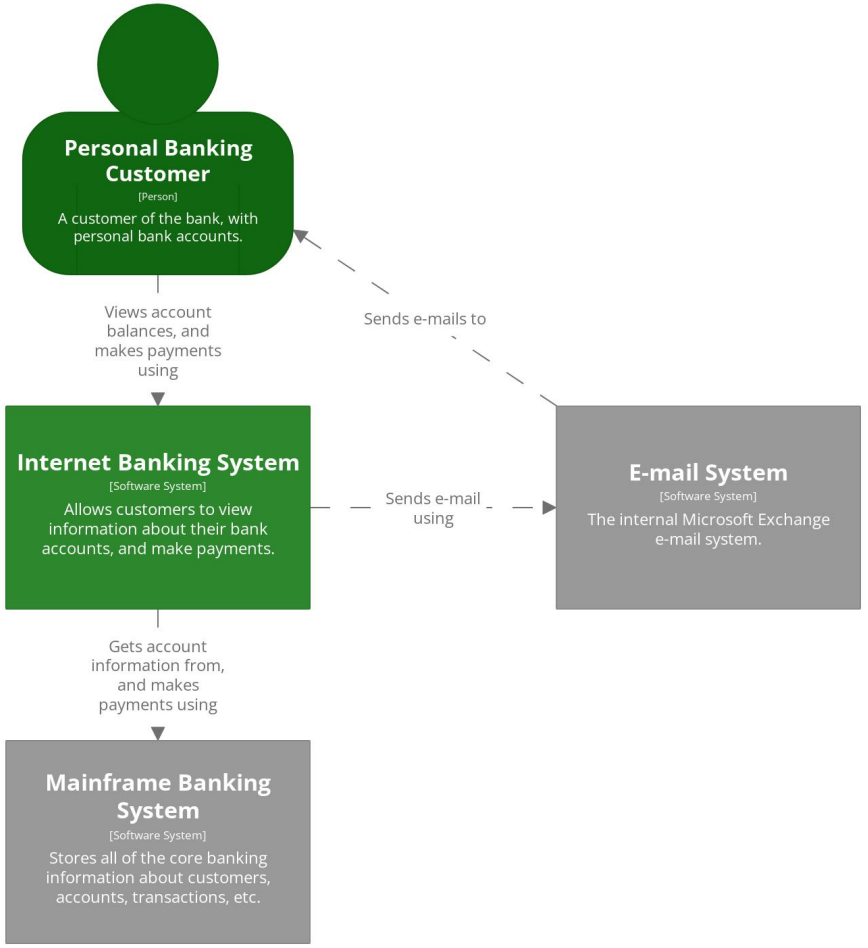
Abstraction

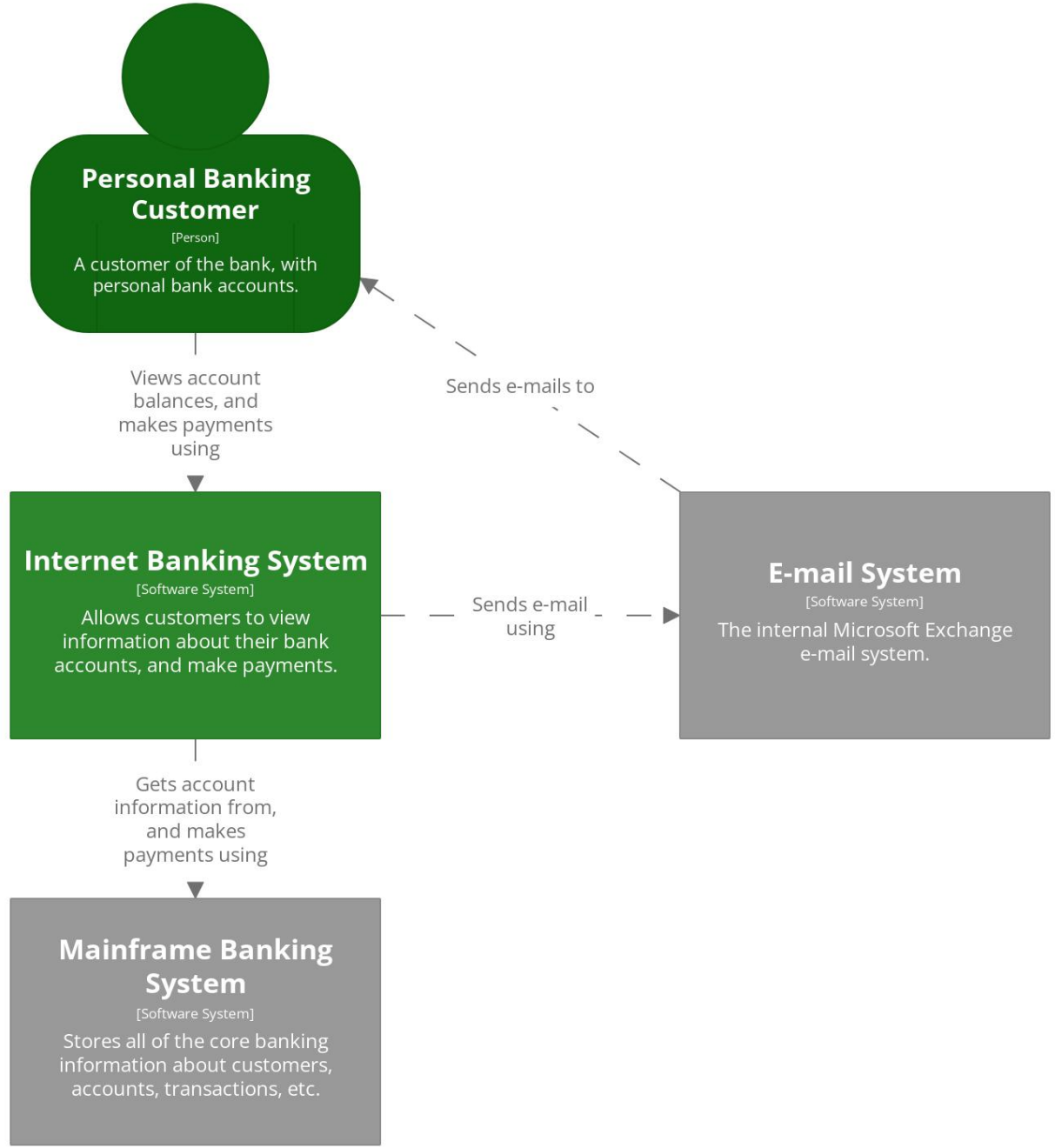


- A **software system** is made up of one or more **containers** (applications and data stores)
- Each containers contains **components**
- Each components are implemented by one or more **code** elements (classes, interfaces, objects, functions, etc)

Software System (Context)

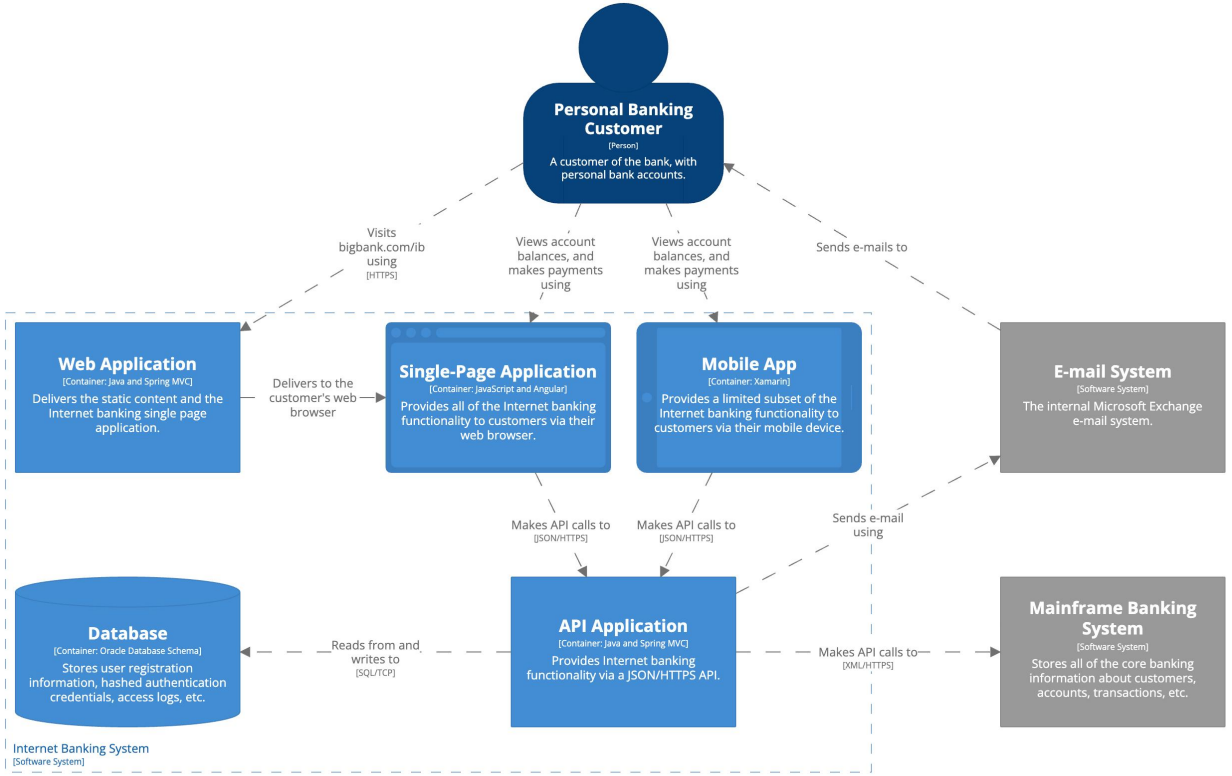
- A **System Context** diagram provides a starting point, outlining your software system's scope and its interactions with users and other systems.
- It grasp the bigger picture before delving into more intricate details.




Personal Banking Customer
[Person]
A customer of the bank, with personal bank accounts.
Views account balances, and makes payments using
Sends e-mails to
Internet Banking System
[Software System]
Allows customers to view information about their bank accounts, and make payments.
Sends e-mail using
E-mail System
[Software System]
The internal Microsoft Exchange e-mail system.
Gets account information from, and makes payments using
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.

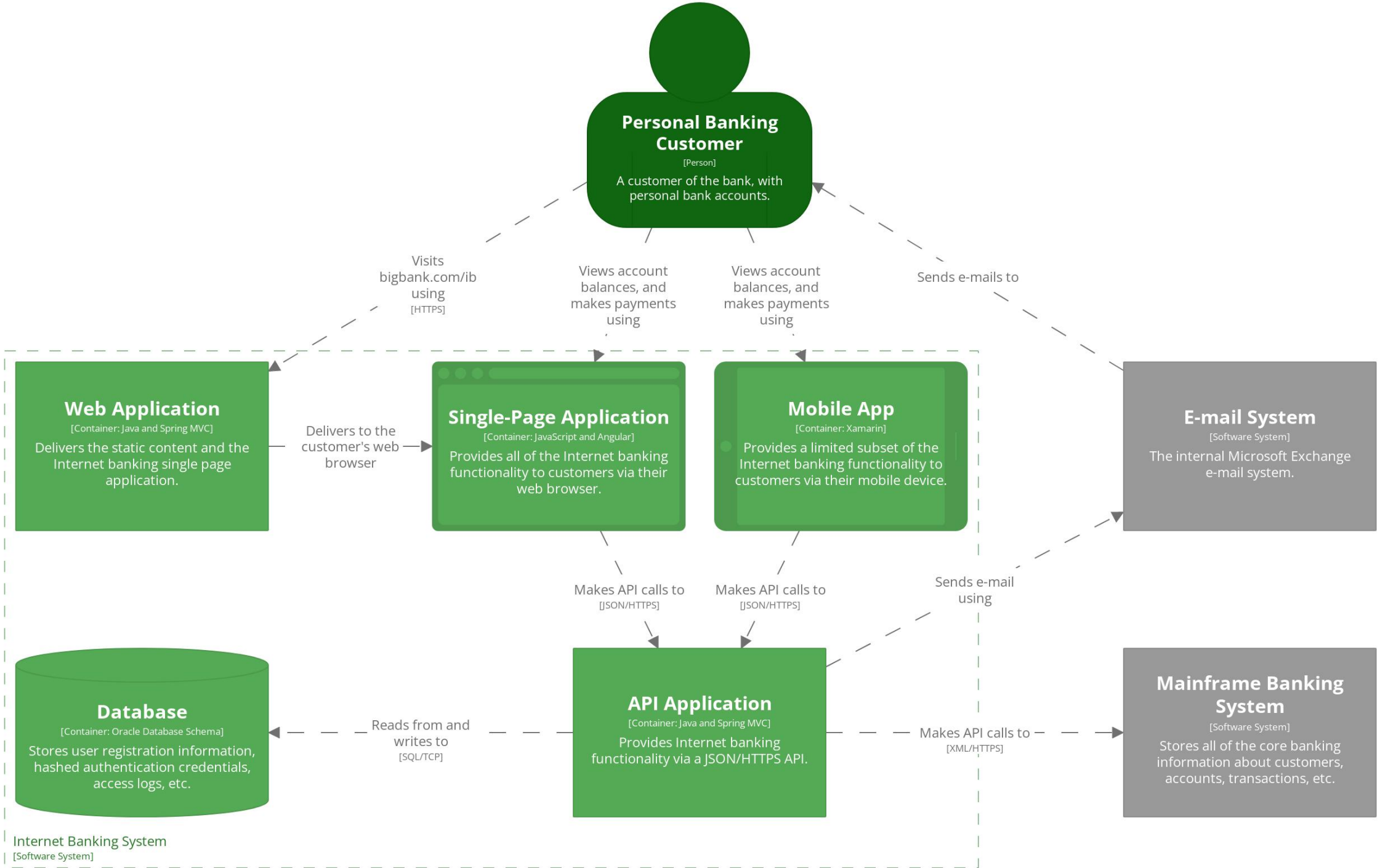
Container

- A **Container diagram** zooms into the software system in scope, showing the high-level technical building blocks.
- A container represents an **application** or a **data store**. A container is something that needs to be running in order for the overall software system to work.



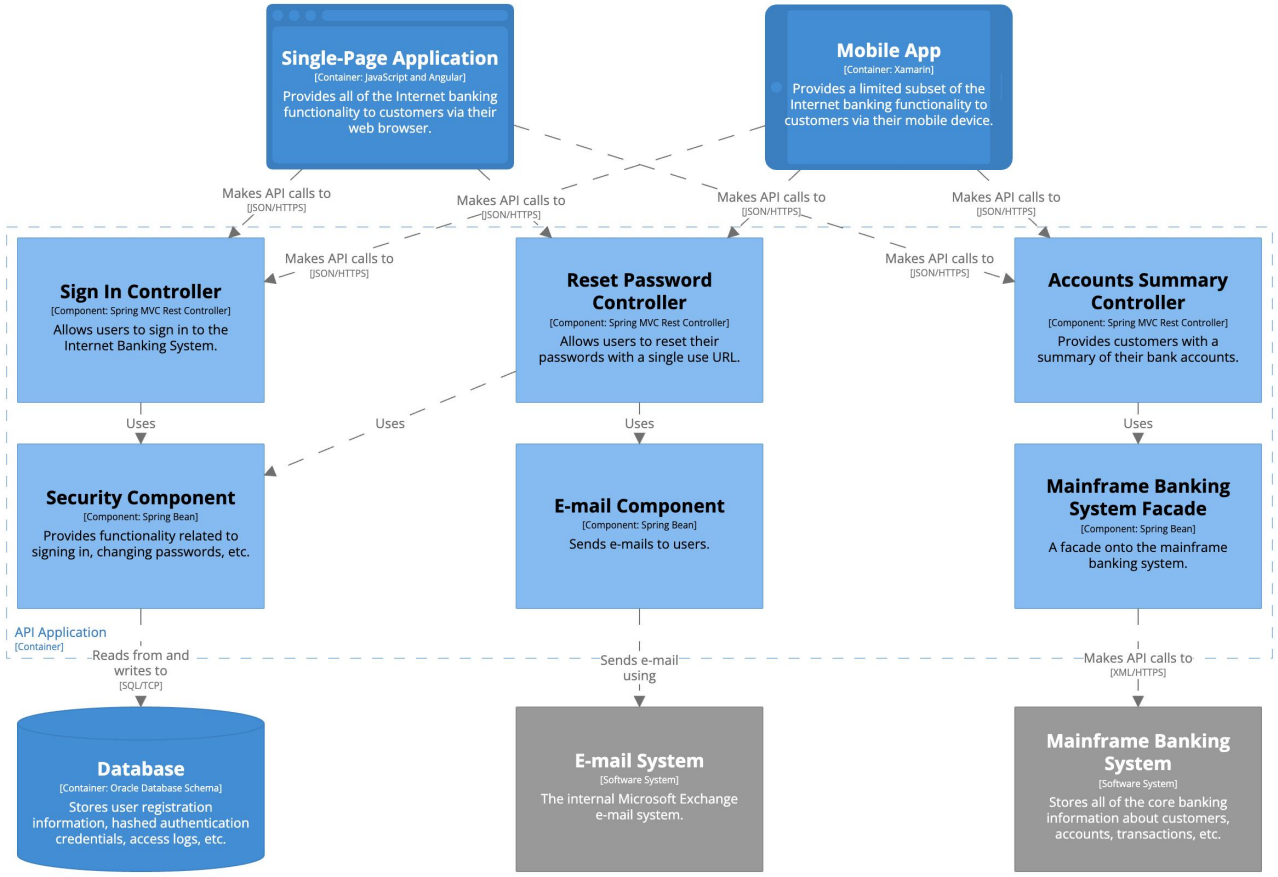
[Container] Internet Banking System

The container diagram for the Internet Banking System - diagram created with Structurizr.
Saturday, 11 November 2023 at 09:04 Greenwich Mean Time


Personal Banking Customer
[Person]
A customer of the bank, with personal bank accounts.
Visits bigbank.com/ib using
[HTTPS]
Views account balances, and makes payments using
Views account balances, and makes payments using
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
Database
[Container: Oracle Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Reads from and writes to
[SQL/TCP]
API Application
[Container: Java and Spring MVC]
Provides Internet banking functionality via a JSON/HTTPS API.
Makes API calls to
[XML/HTTPS]
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.
Internet Banking System
[Software System]

Components

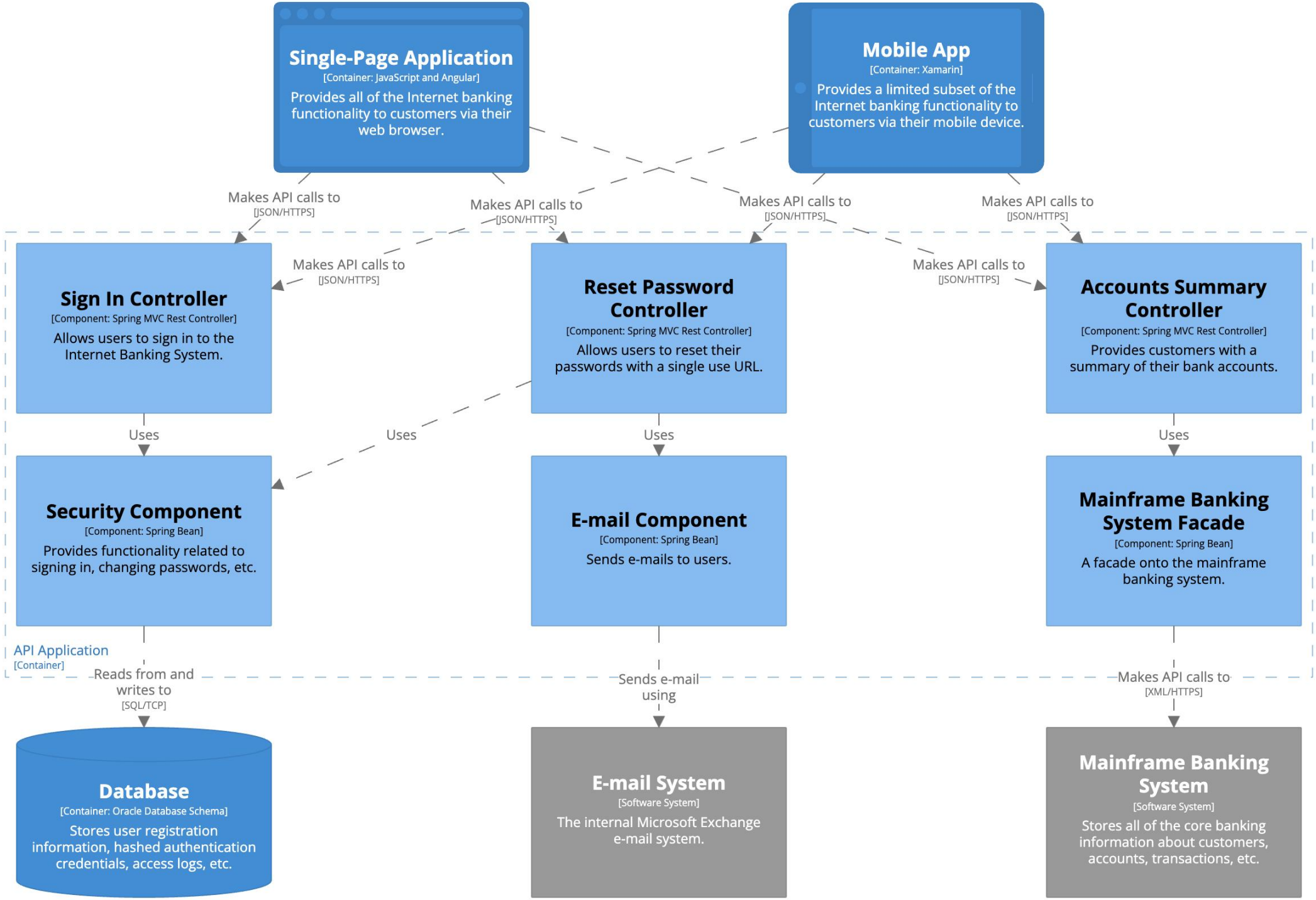
- A **Component** diagram zooms into an individual container, showing the components inside it.
- The Component diagram offers a closer look at the **individual elements within each container**, much like inspecting the items within a room.
- This stage is about understanding how these components interact, a critical factor in ensuring the system's overall strength and responsiveness.



[Component] Internet Banking System - API Application

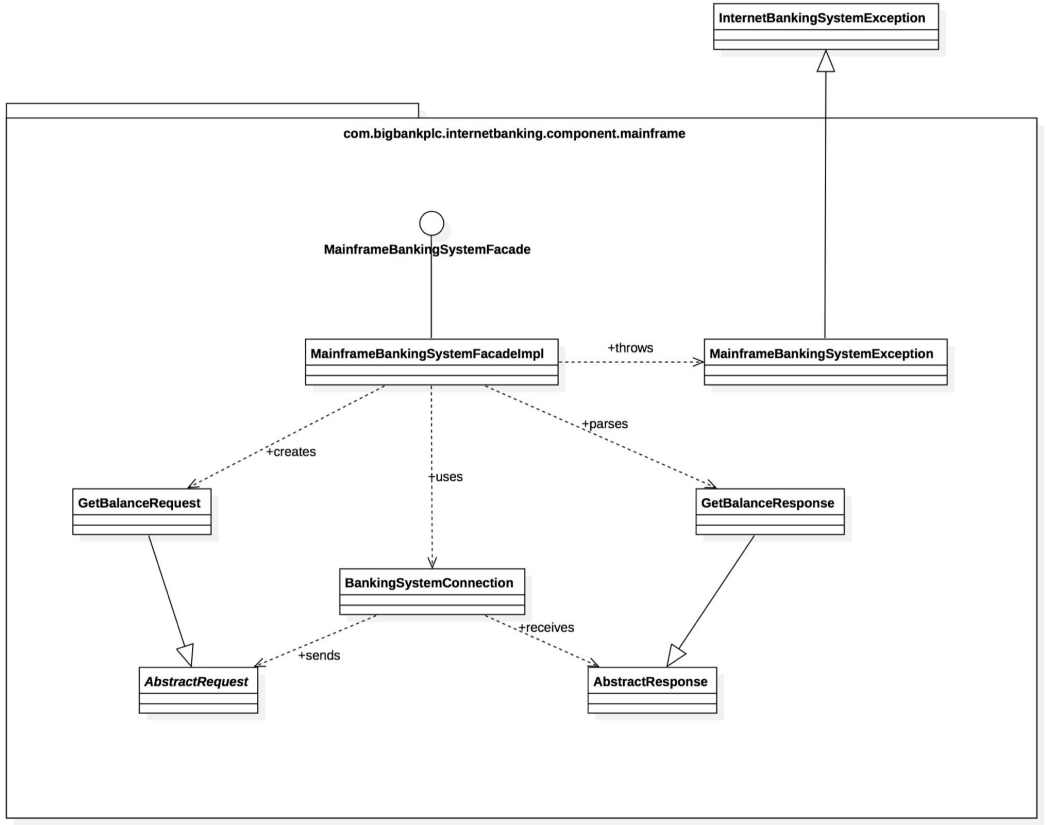
The component diagram for the API Application - diagram created with Structurizr.

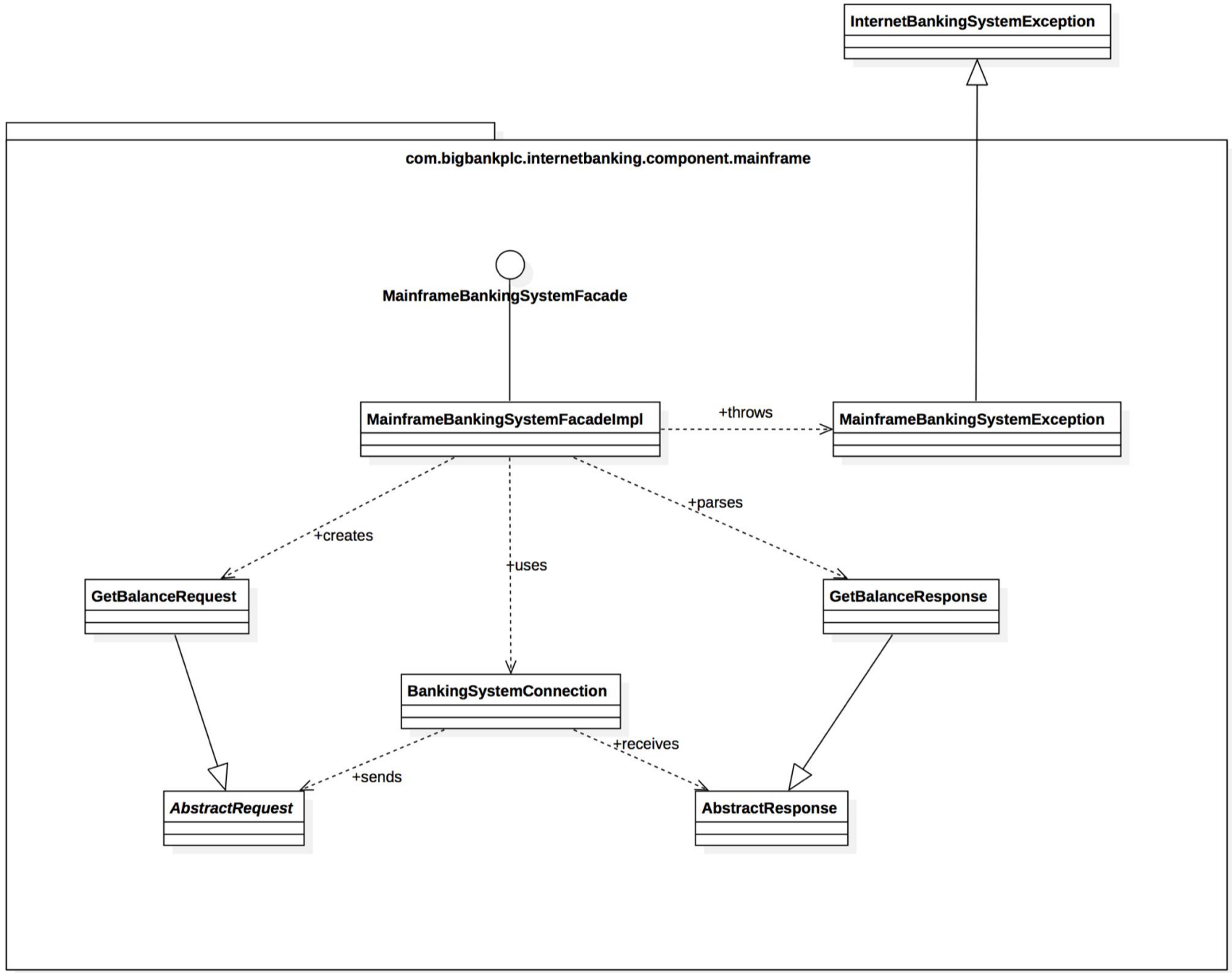
Saturday, 11 November 2023 at 09:04 Greenwich Mean Time

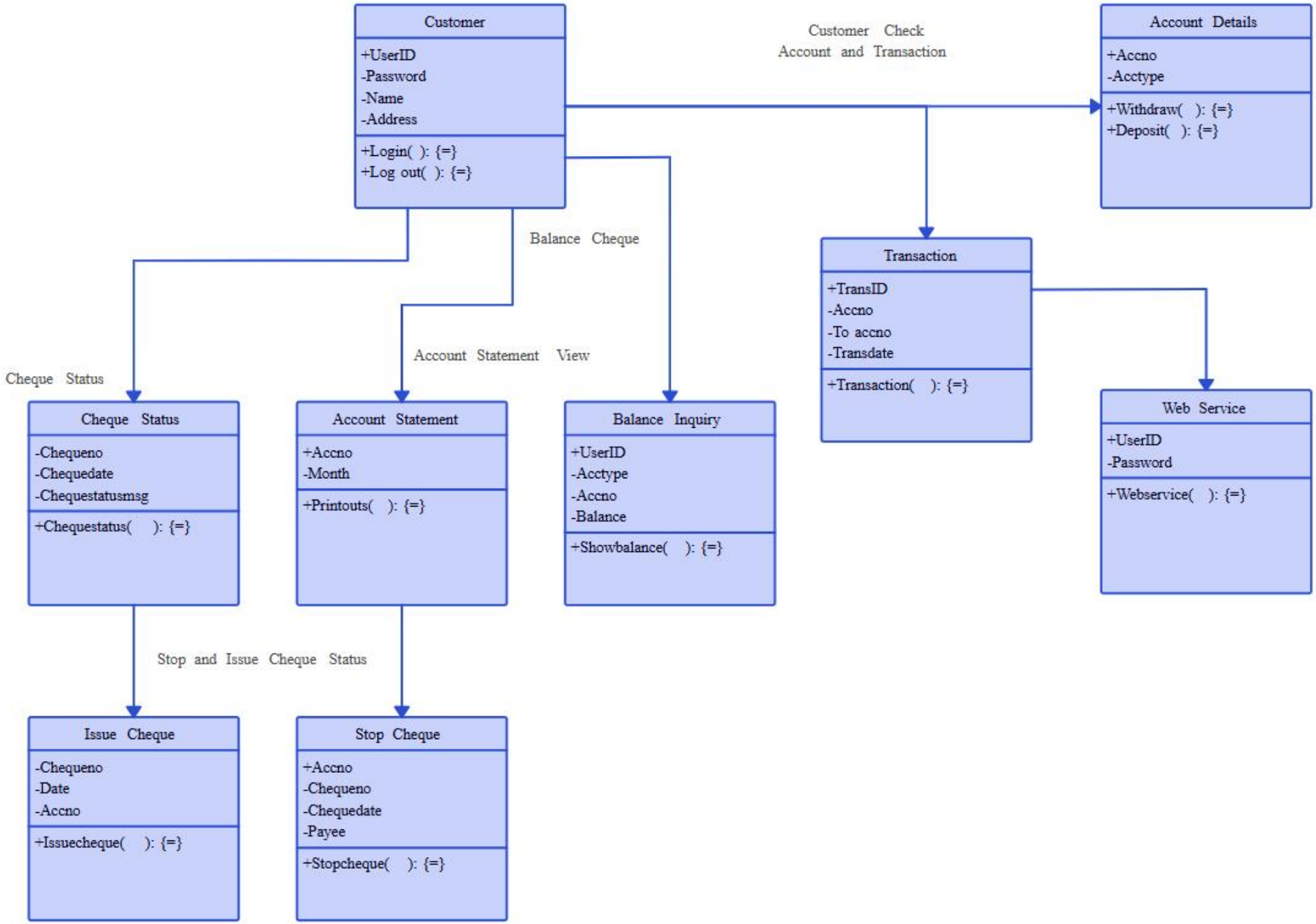

Single-Page Application
[Container: JavaScript and Angular]
Provides all of the Internet banking functionality to customers via their web browser.
Mobile App
[Container: Xamarin]
Provides a limited subset of the Internet banking functionality to customers via their mobile device.
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Makes API calls to
[JSON/HTTPS]
Sign In Controller
[Component: Spring MVC Rest Controller]
Allows users to sign in to the Internet Banking System.
Reset Password Controller
[Component: Spring MVC Rest Controller]
Allows users to reset their passwords with a single use URL.
Accounts Summary Controller
[Component: Spring MVC Rest Controller]
Provides customers with a summary of their bank accounts.
Uses
Uses
Uses
Uses
Security Component
[Component: Spring Bean]
Provides functionality related to signing in, changing passwords, etc.
E-mail Component
[Component: Spring Bean]
Sends e-mails to users.
Mainframe Banking System Facade
[Component: Spring Bean]
A facade onto the mainframe banking system.
API Application
[Container]
Reads from and writes to
[SQL/TCP]
Sends e-mail using
Makes API calls to
[XML/HTTPS]
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.

Codes

- A **code** (e.g. UML class) diagram can be used to zoom into an individual component, showing how that component is implemented.
- Provides a detailed view of the system's building blocks, showcasing the classes and interfaces that underpin the components.
- This is an optional level of detail




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


Customer
+UserID
-Password
-Name
-Address
+Login( ): {=}
+Log out( ): {=}
Customer Check
Account and Transaction
Account Details
+Accno
-Acctype
+Withdraw( ): {=}
+Deposit( ): {=}
Balance Cheque
Transaction
+TransID
-Accno
-To accno
-Transdate
+Transaction( ): {=}
Cheque Status
Account Statement View
Cheque Status
-Chequeno
-Chequedate
-Chequestatusmsg
+Chequestatus( ): {=}
Account Statement
+Accno
-Month
+Printouts( ): {=}
Balance Inquiry
+UserID
-Acctype
-Accno
-Balance
+Showbalance( ): {=}
Web Service
+UserID
-Password
+Webservice( ): {=}
Stop and Issue Cheque Status
Issue Cheque
-Chequeno
-Date
-Accno
+Issuecheque( ): {=}
Stop Cheque
+Accno
-Chequeno
-Chequedate
-Payee
+Stopcheque( ): {=}

References

- https://c4model.com/
- https://www.lucidchart.com/blog/c4-model
- https://miro.com/diagramming/c4-model-for-software-architecture/