如何画软件架构
C4 Model

11/20/2022

C4 Model - a way to create maps of your code, at various levels of detail

Like source code, Google Street View provides a very low-level and accurate view of a location.

Navigating an unfamiliar environment becomes easier if you zoom out though.

Zooming out further will provide additional context you might not have been aware of.

Different levels of zoom allow you to tell different stories to different audiences.

Level 1 - Context

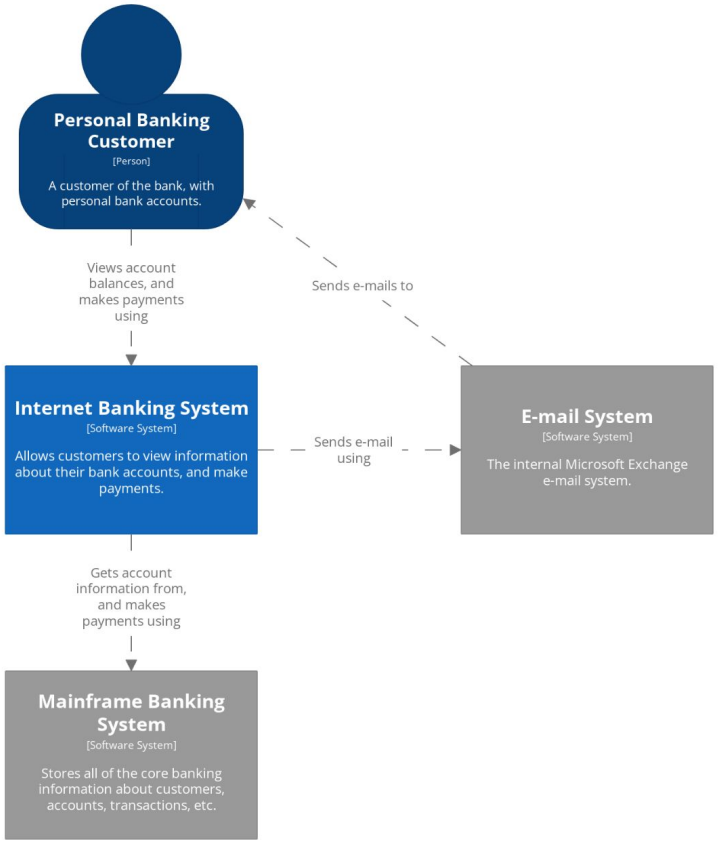
Context即系统上下文，这是最高Level的架构图，不暴露系统的内部细节，所展现的是：

- 本系统的用户是谁
- 与哪些外部系统有交互

系统上下文图可以说是一个系统的外观，不是站在系统内部，而是抽离出来，将本系统看作一个黑盒，站在系统外部来看，来展现与用户、外部系统间的关系。

细节在这里并不重要，因为这是显示系统全景图的缩小视图。重点应该放在用户和软件系统上，而不是技术和其他实现细节。这是可以向所有人展示的架构图， 包括技术人员与非技术人员。

Level 1 - context



[System Context] Internet Banking System

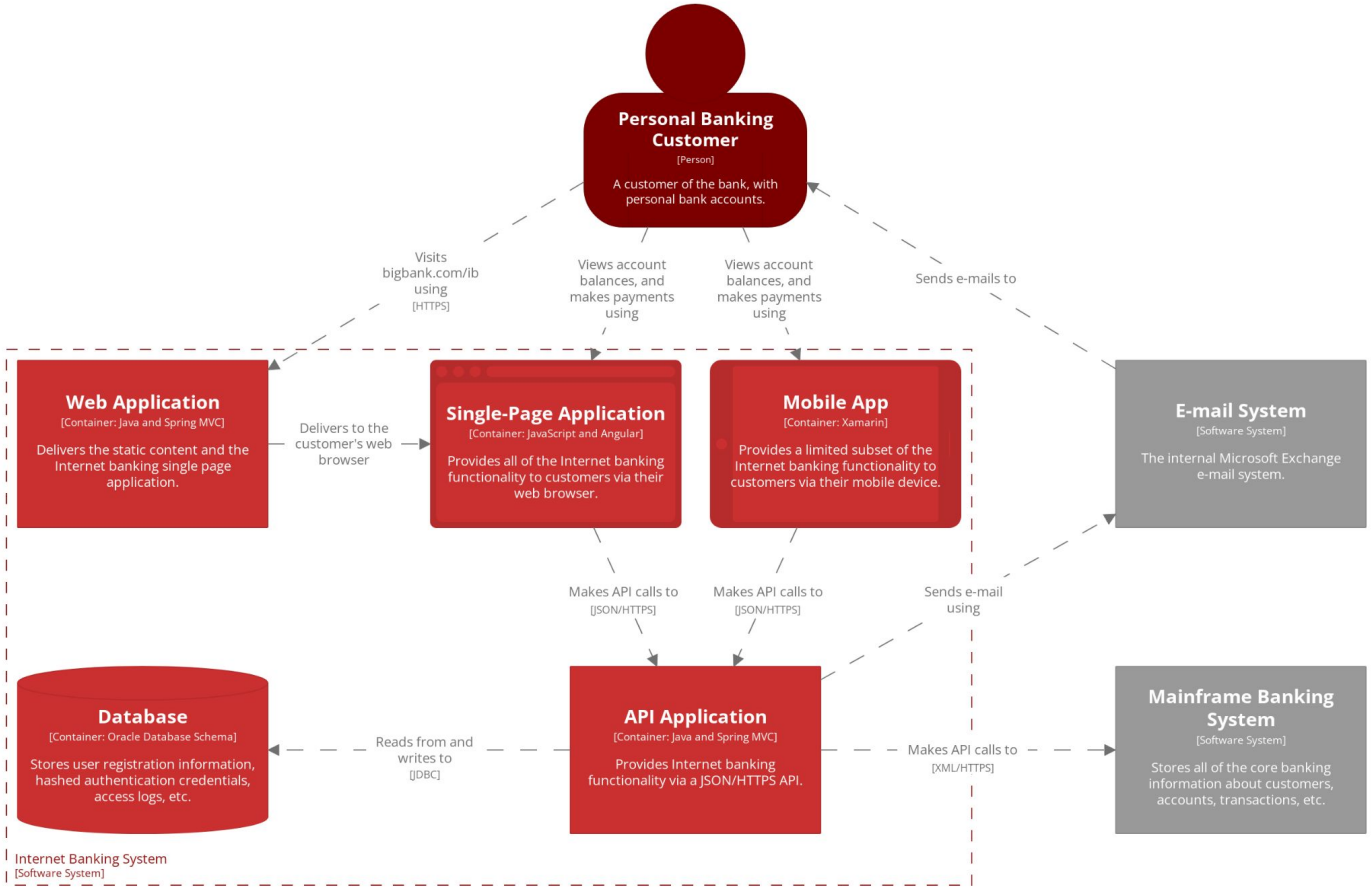
Tuesday, November 8, 2022, 6:48 PM Coordinated Universal Time

Level 2 - Container

这里容器的概念，有别于Docker容器，指的是在一个软件系统中可**独立部署/独立运行**的单元。可以是服务器端 Web 应用、单页应用、桌面应用、移动应用、数据库等。

如果说系统上下文图将本系统看作一个黑盒，那么容器图则是把这个黑盒打开，这个时候看到的是系统内部的所有容器，及各个容器盘根错节的交互关系，可以展现容器的主要技术栈，容器间如何通信。这是一个以技术为重点的简单图表，对软件开发人员、支持和运维人员都很有用。

Level 2 - Container



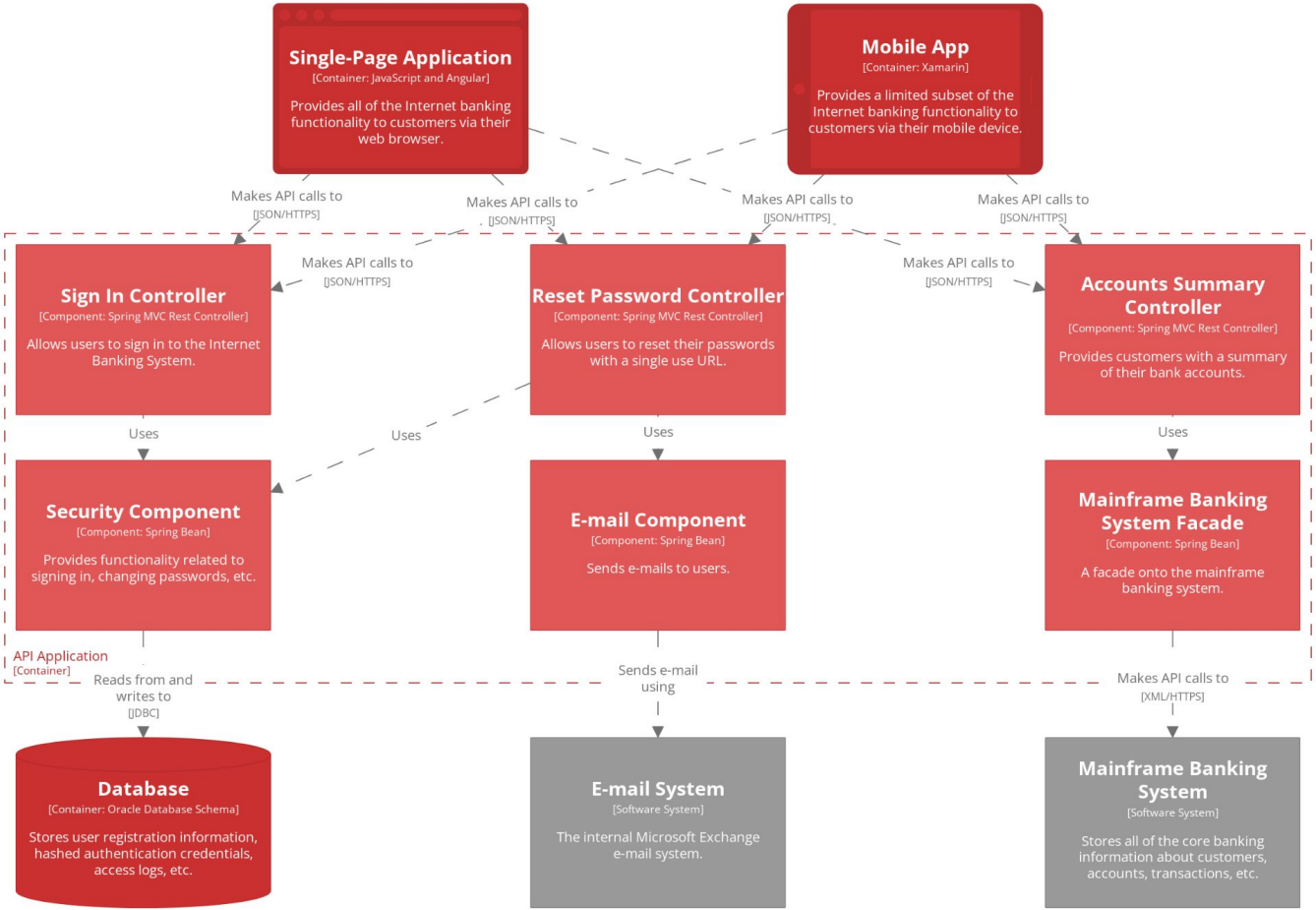
[Container] Internet Banking System

Tuesday, September 27, 2022, 7:30 PM Coordinated Universal Time

Level 3 - Component

组件图则是放大和分解每个容器，显示了容器是如何由多个“组件”组成的，每个组件是什么，它们的职责以及技术/实现细节。

Level 3 - Component



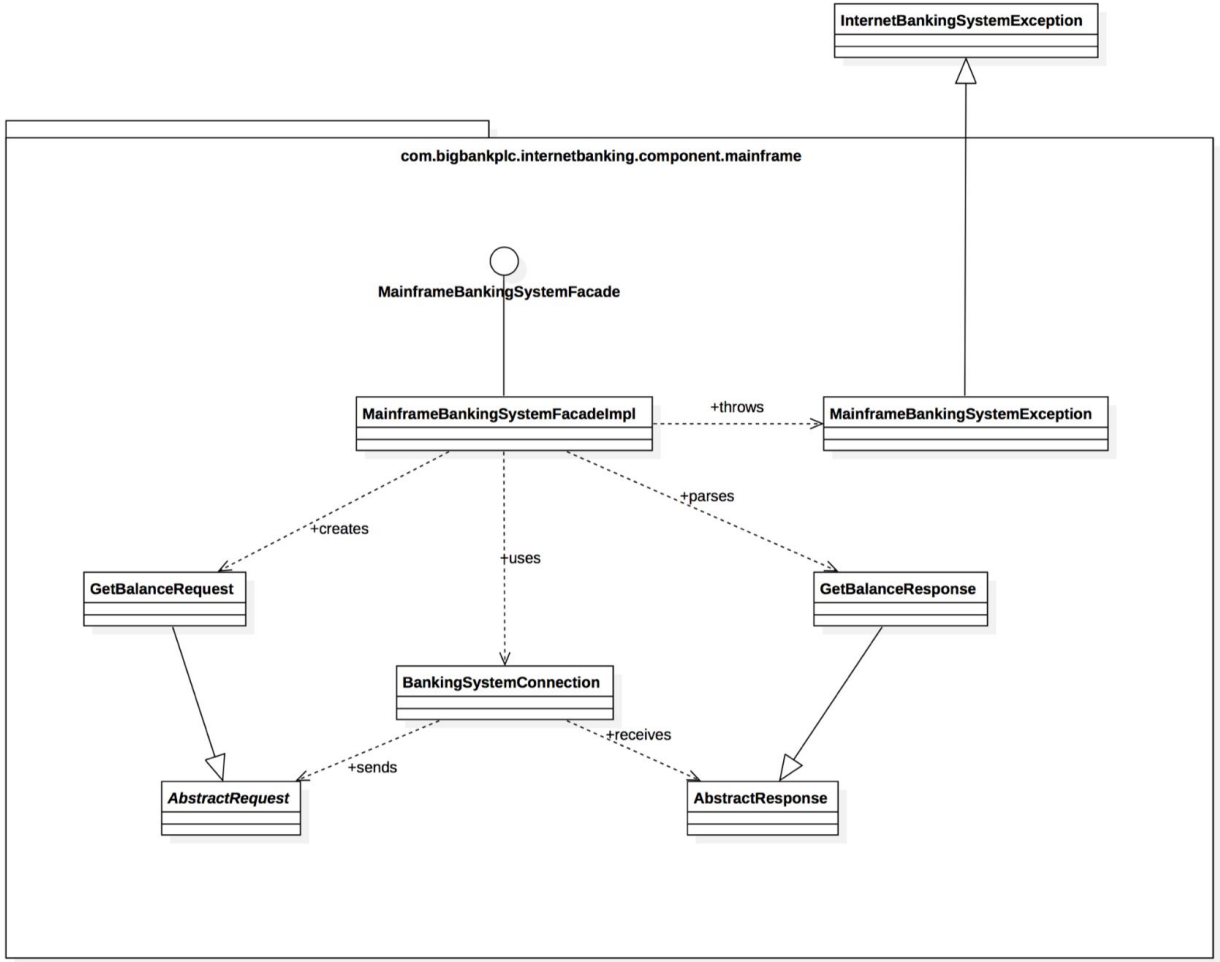
[Component] Internet Banking System - API Application

Tuesday, September 27, 2022, 7:30 PM Coordinated Universal Time

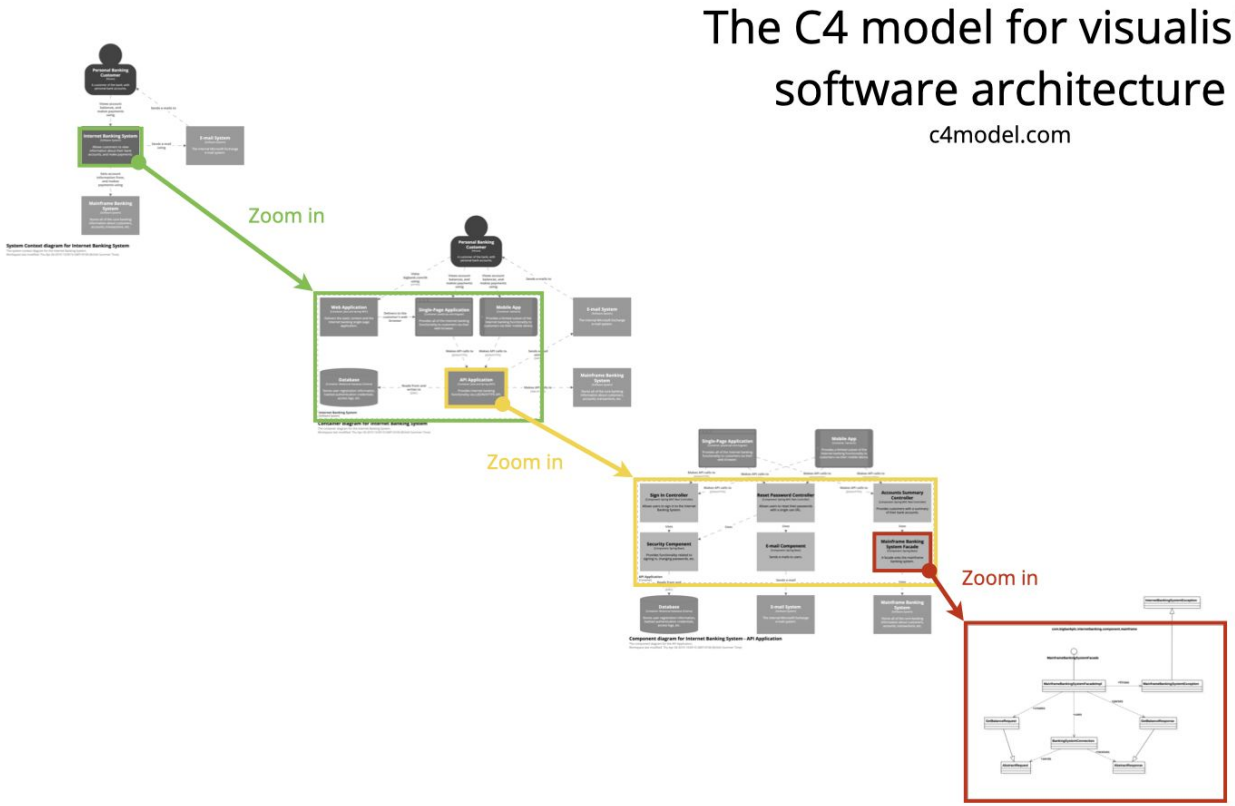
Level 4 - Code

代码层则是放大每个组件以显示它是如何作为代码实现的；使用 UML 类图、实体关系图等。 理想情况下，该图将使用工具（例如 IDE 或 UML 建模工具）自动生成，您应该考虑仅显示那些想要展示的属性和方法。

Level 4 - Code

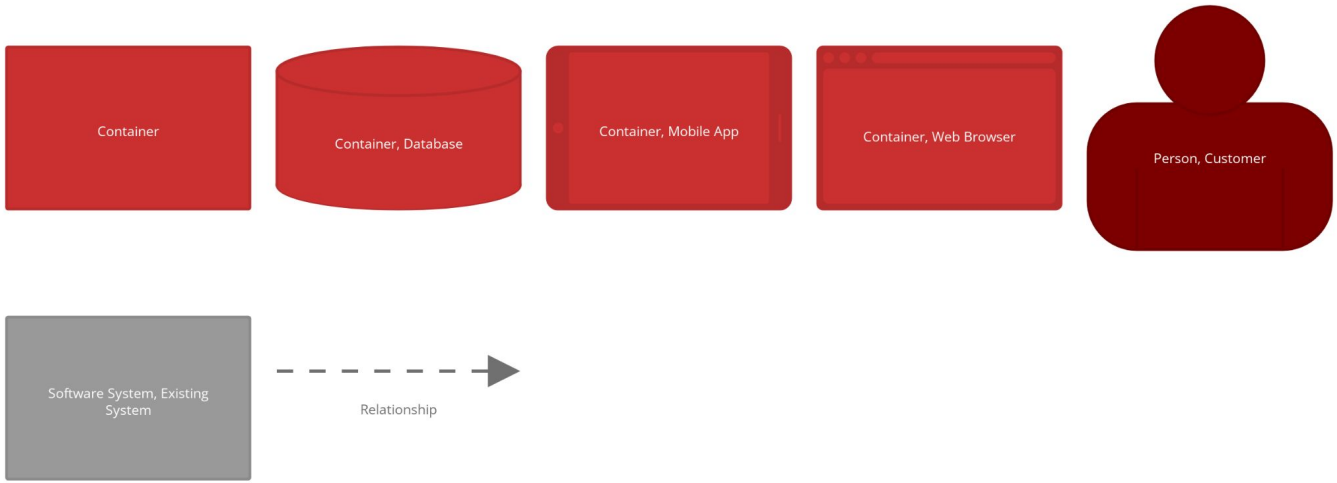

InternetBankingSystemException
com.bigbankplc.internetbanking.component.mainframe
MainframeBankingSystemFacade
MainframeBankingSystemFacadeImpl
+throws
MainframeBankingSystemException
+creates
+uses
+parses
GetBalanceRequest
GetBalanceResponse
BankingSystemConnection
+receives
+sends
AbstractRequest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

Combination 4 steps



C4 模型没有规定任何特定的符号。使用的任何符号都应尽可能具有自描述性，但所有图表都应有一图例，以使符号明确。这也适用于使用 UML、ArchiMate 和 SysML 等符号创建的图表，因为不是每个人都知道所使用的符号。适用于白板、纸张、便签和各种绘图工具的简单符号如下：

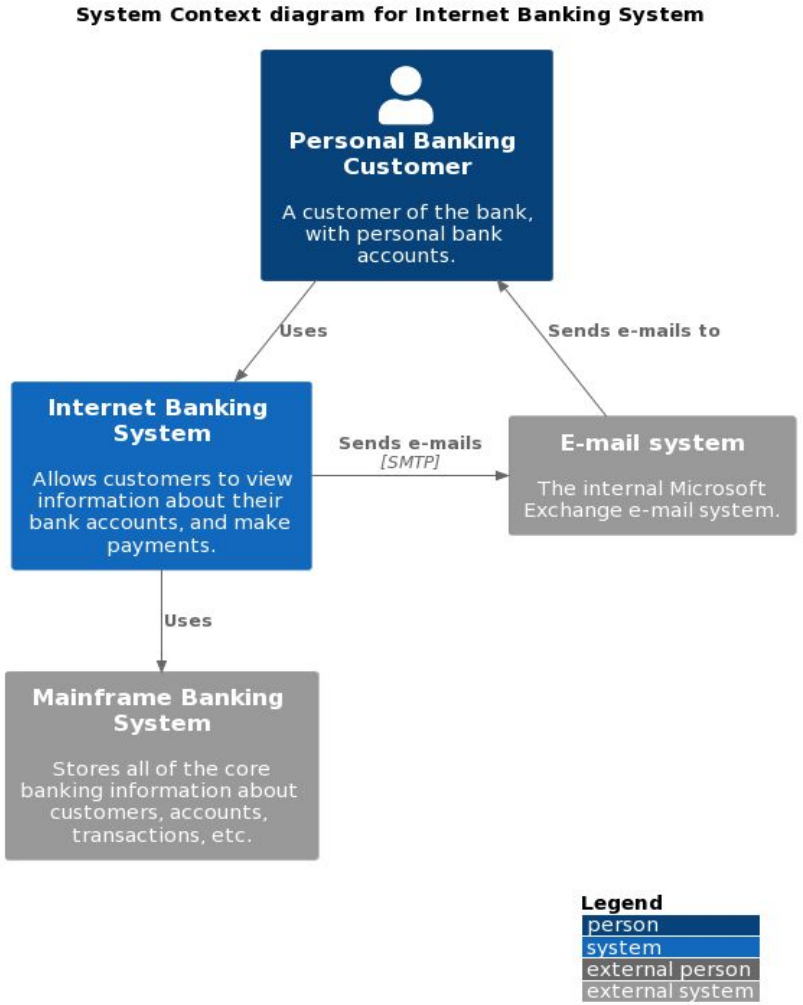
然后可以使用颜色和形状来补充图表，以添加其他信息或只是使图表更美观。



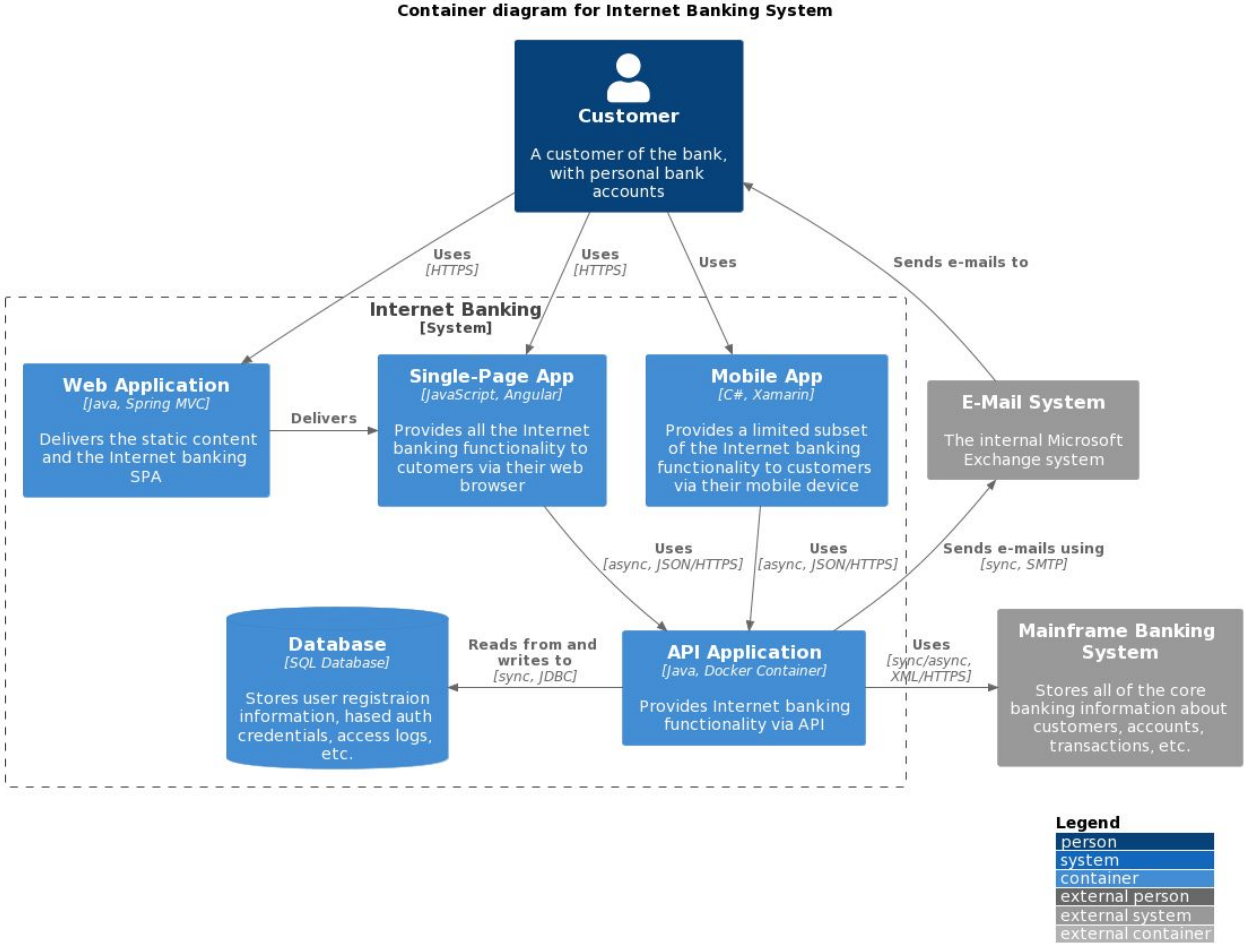
思考

- 对于一个系统，并不一定必须有这几个Level的架构图，而是当想要去画一个架构图时，需要回答几个问题：
  - 受众是谁？
  - 需要哪个Level的图？
  - 重点关注在哪一块？
  - 与其他系统/组件之间的交互有哪些？是否在同一Level展现？
- 对于一个系统的各个Level的架构图，个人认为对于类似tech meeting这样的场景来说，Context图和Container图的意义要高于Component图和Code结构图，而对于同组同org的话后二者更详实，更关注实现细节，受众是开发人员，有时候稳定性更差，参考价值低于代码本身。
- C4 Model的价值更多在于如何认识系统，从哪个角度去剖析系统，而不拘泥于用什么线条、形状画出来。

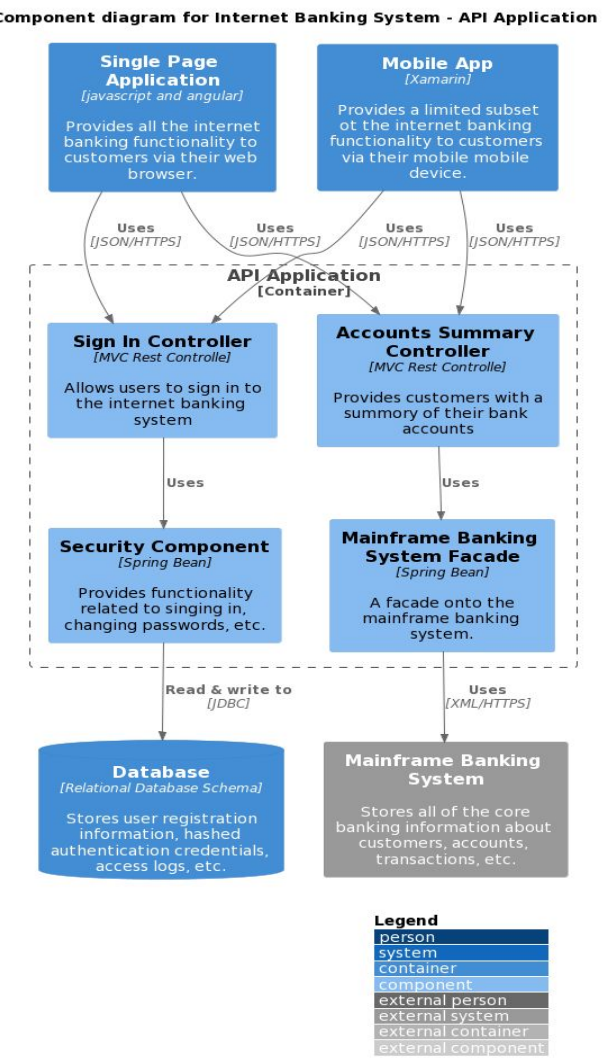
Demo - Design an Online Banking - Context Diagram



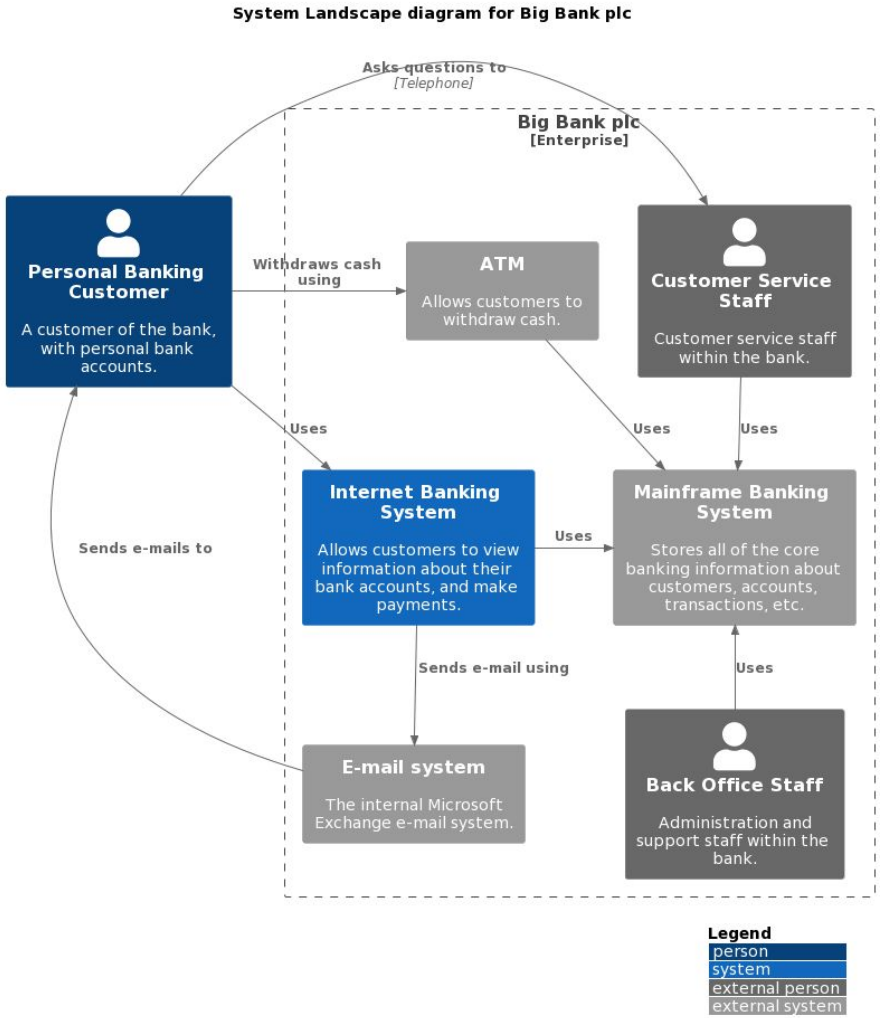
Demo - Design an Online Banking - Container Diagram



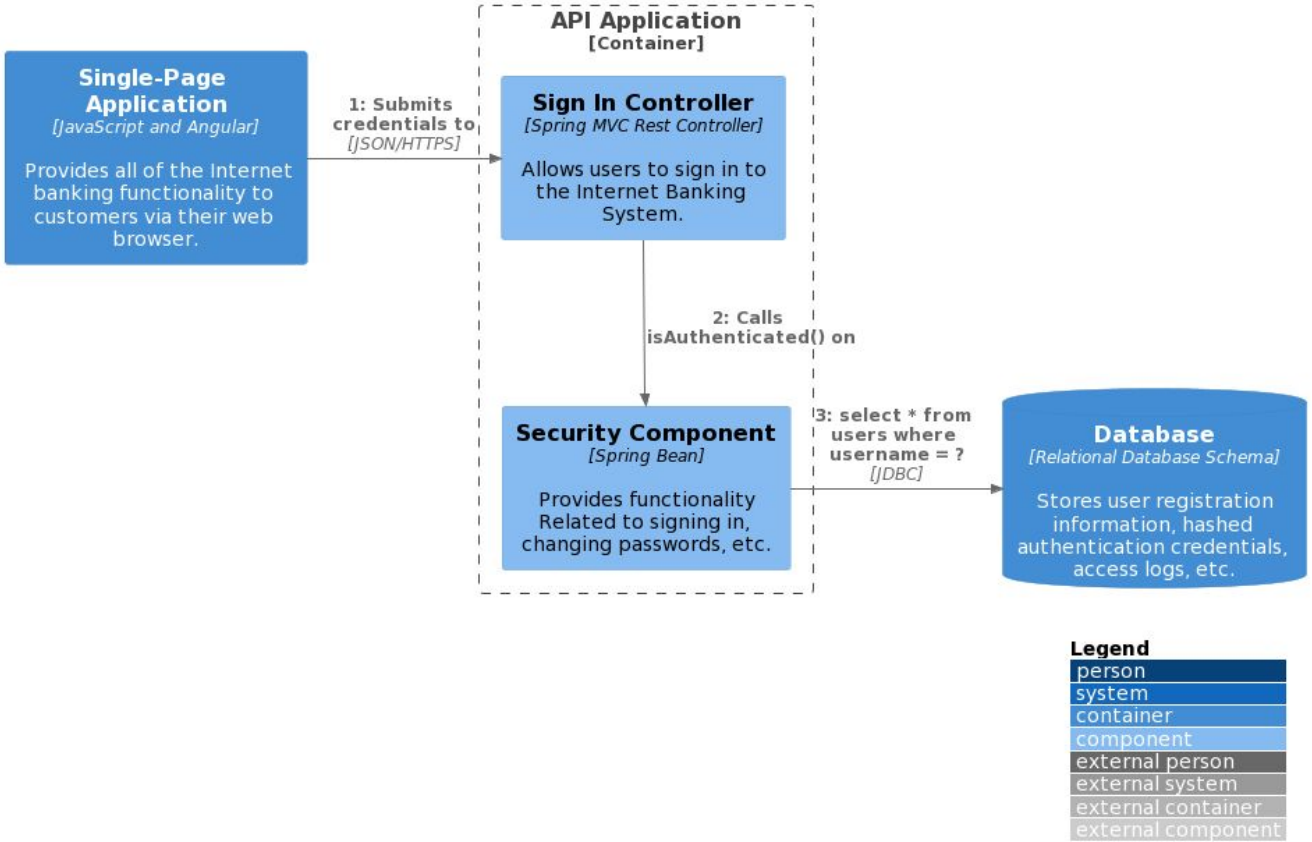
Demo - Design an Online Banking - Component Diagram



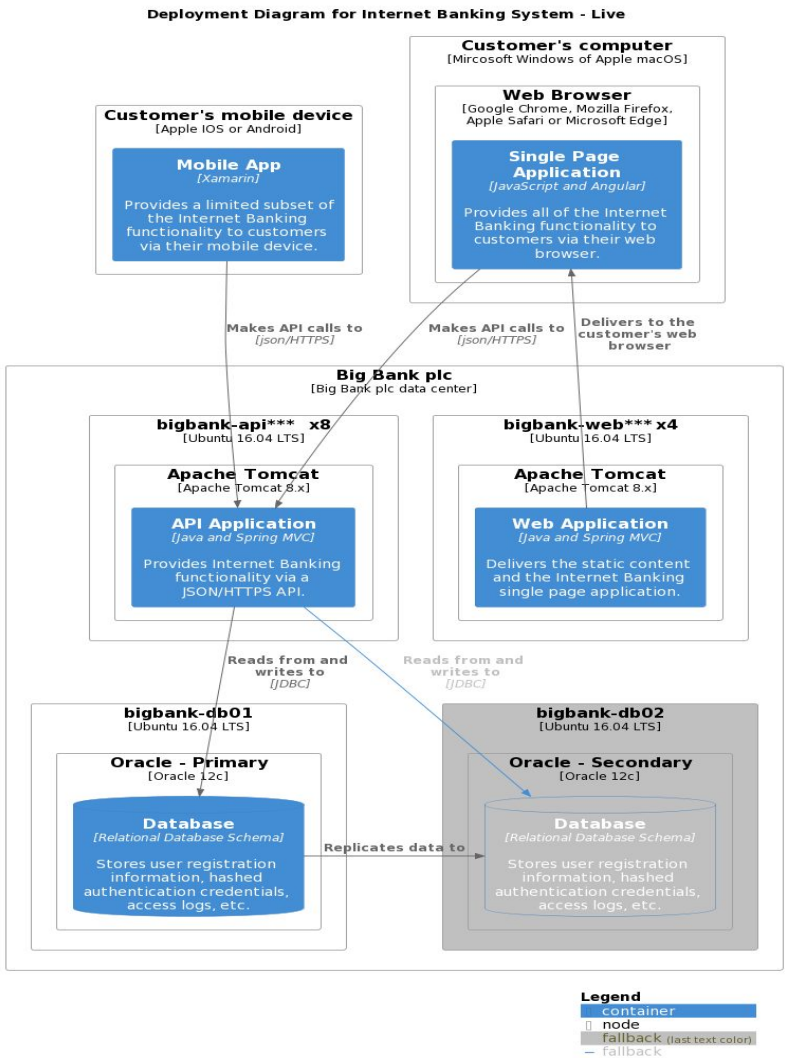
Demo - Design an Online Banking - Supplemental- Landscape Diagram



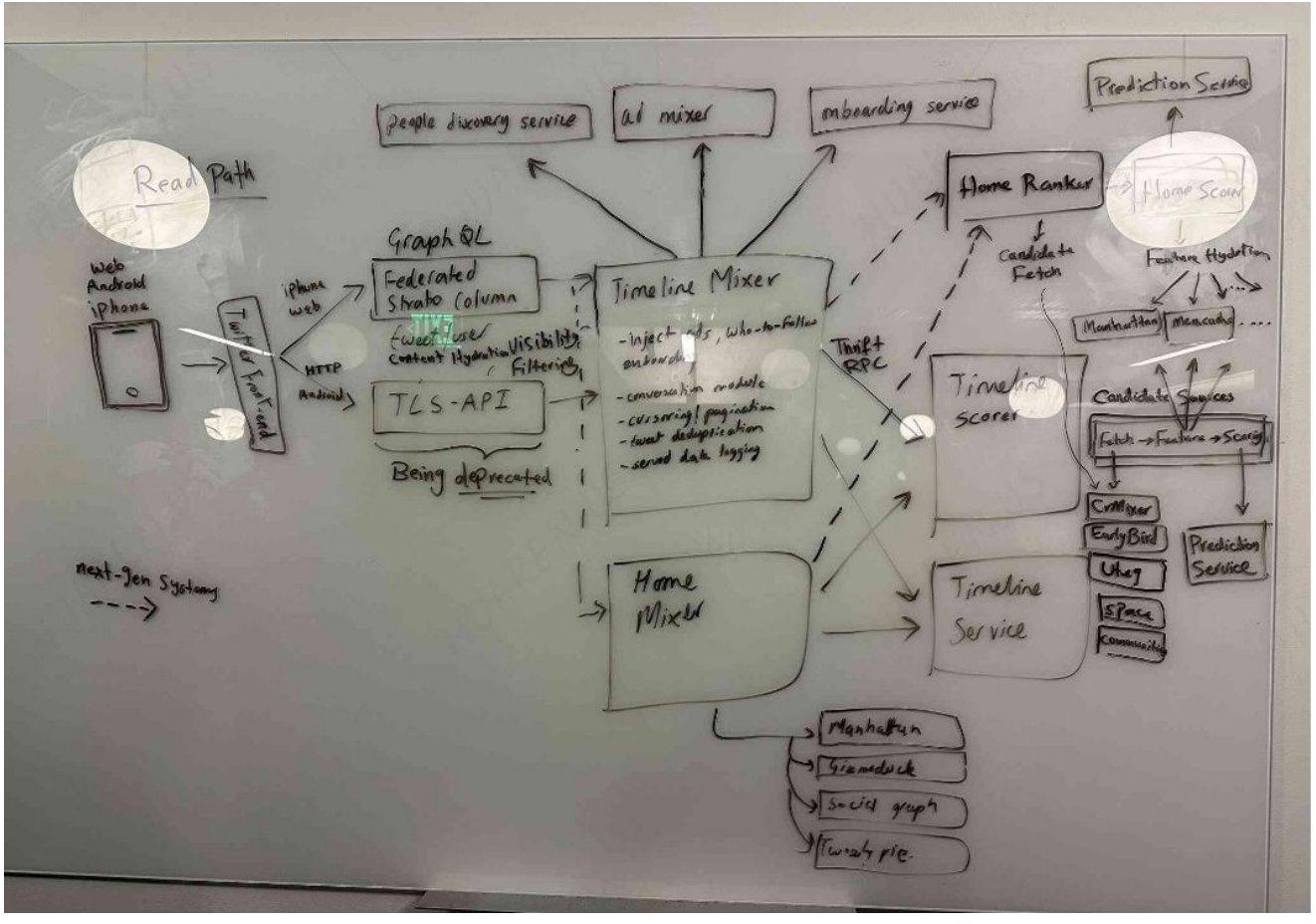
Demo - Design an Online Banking - Supplemental - Dynamic Diagram



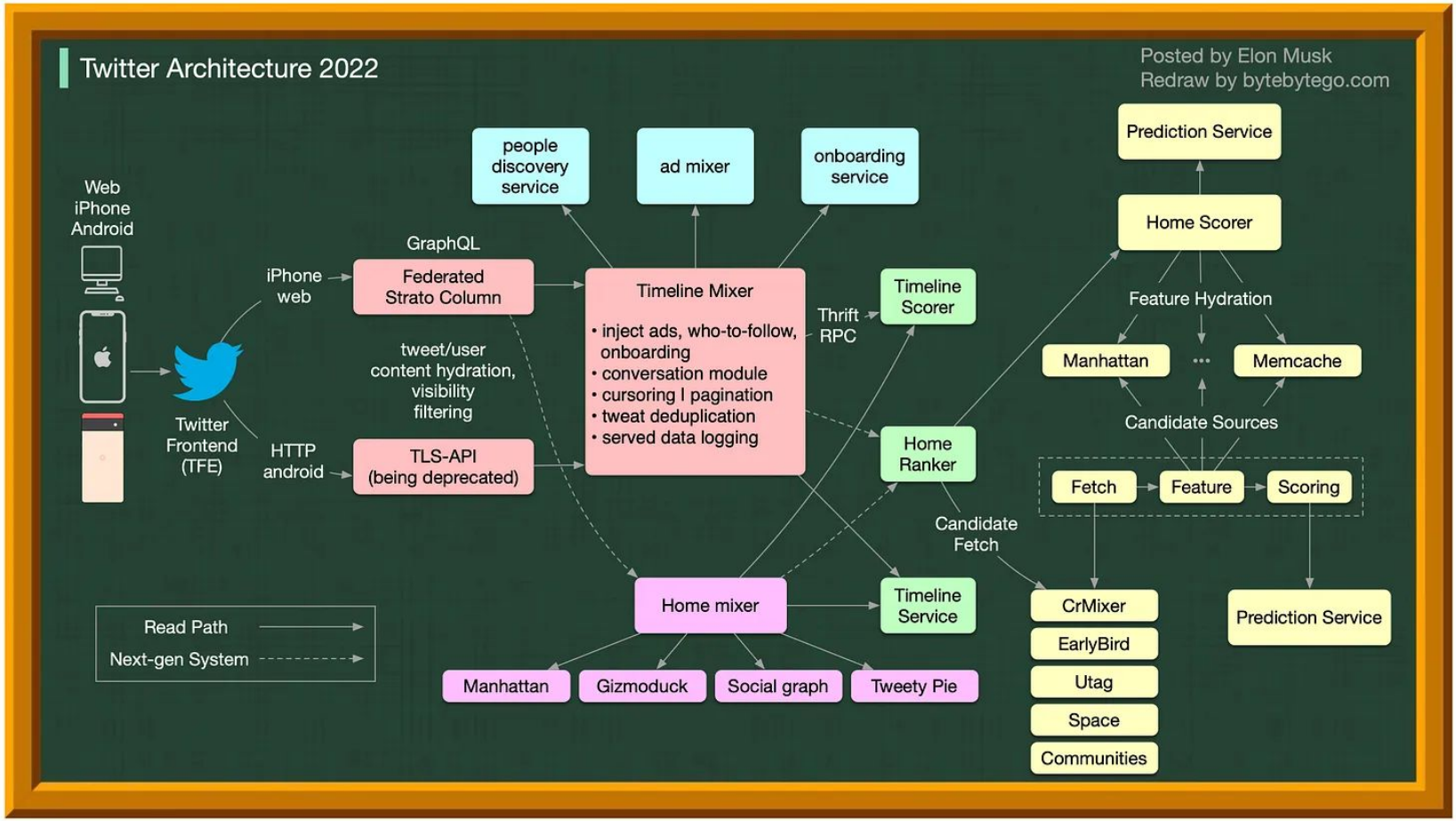
Demo - Design an Online Banking - Supplemental - Deployment Diagram



Twitter Architecture 2022



Twitter Architecture 2022



Reference

https://c4model.com/

https://github.com/plantuml-stdlib/C4-PlantUML/blob/master/samples/C4CoreDiagrams.md

https://blog.bytebytego.com/p/twitter-architecture-2022-vs-2012