Logical Components and Modelling Using C4

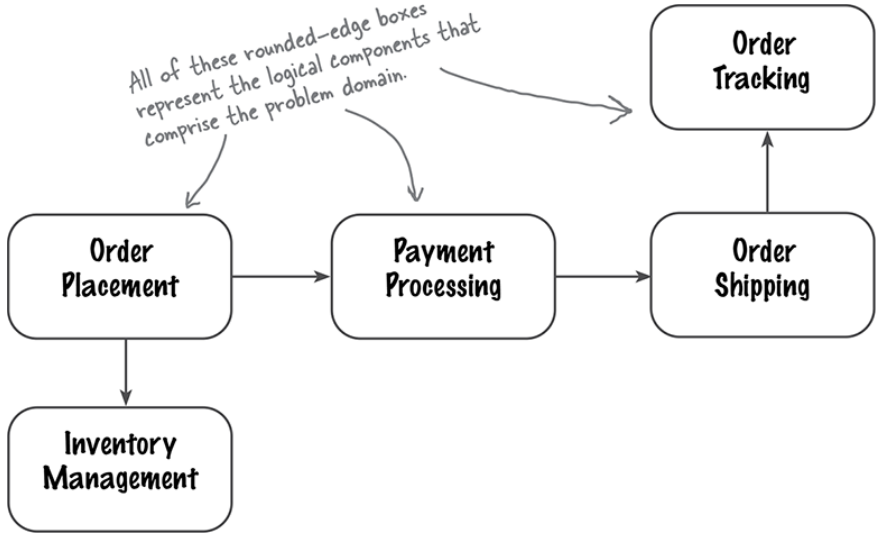
UNSW
SYDNEY

These lecture slides are from the book "*Head First Software Architecture*",

by Raju Gandhi, Mark Richards, Neal Ford, O'Reilly Media, Inc., March 2024

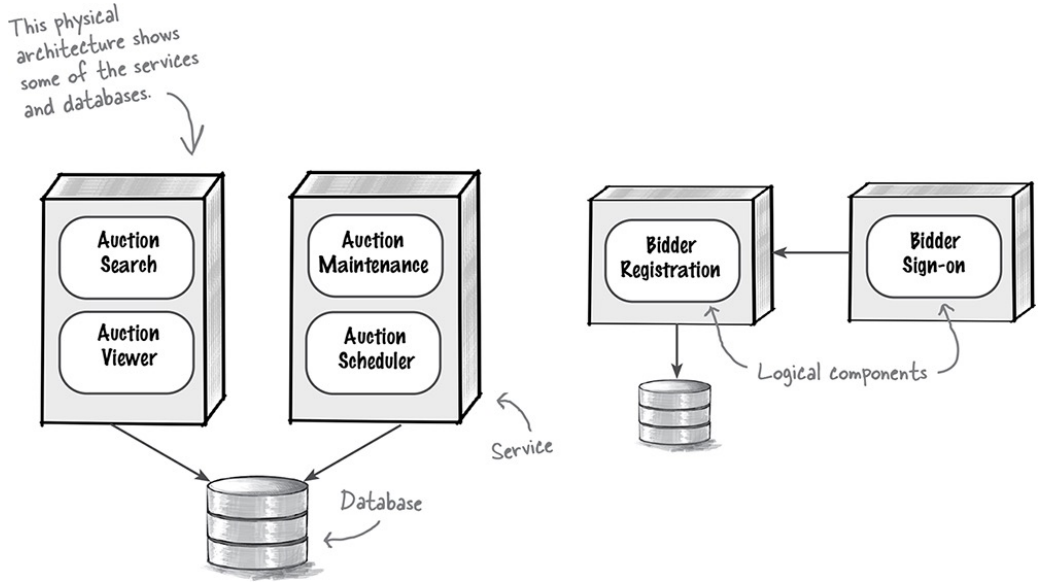
What Are Logical Components?

- Functional building blocks of the system
- Represent major features or responsibilities
- Typically map to folders or modules in the codebase



Logical vs Physical Architecture

- Logical Architecture: Describes what the system does (functional perspective)
- Physical Architecture: Describes how the system is built and deployed (technical perspective)
- Example:
  - Logical: Bidding, Registration, Payment
  - Physical: APIs, databases, gateways, services



Creating a Logical Architecture

Follow a 4-step process:

- ❖ Identify core components
- ❖ Assign requirements
- ❖ Analyse roles & responsibilities
- ❖ Align with architectural characteristics

- ➢ Revisit this cycle whenever system changes are introduced

Align with Architectural Characteristics

- Break down or merge components based on:
  - Scalability
  - Availability
  - Performance
- Example: Move bid logging to separate Bid Tracker to improve speed and availability

Component Coupling

- Afferent (incoming): How many depend on this component
- Efferent (outgoing): How many this component depends on
- Total Coupling = Afferent + Efferent

Goal: Keep coupling low for flexibility and maintainability

The Law of Demeter

- Also known as the Principle of Least Knowledge
- Each component should only interact with its immediate neighbors
- Avoid tight coupling caused by too much knowledge about the system

Coupling Trade-offs

- Tightly Coupled System: Easier to trace workflow, harder to change
- Loosely Coupled System: More maintainable, but harder to understand in one place

Remember: Everything is a trade-off

Logical Components: Summary

- Logical components are your system's functional map
- Use descriptive names based on responsibilities
- Avoid entity trap and generic components
- Reduce coupling using the Law of Demeter
- Regularly reevaluate components as requirements evolve

Introduction to C4 Architectural Modelling

Fethi Rabhi (June 2025)

Challenges in Architecture modelling

Its all about tradeoffs

- Addressing functional requirements
- Balancing non-functional requirements
- Finding a balance between “understandability” (by humans) and “correctness” (the code) is a complex undertaking, especially in cross-functional teams, where you’re explaining to a mixed group of technical and non-technical people

Iterative process

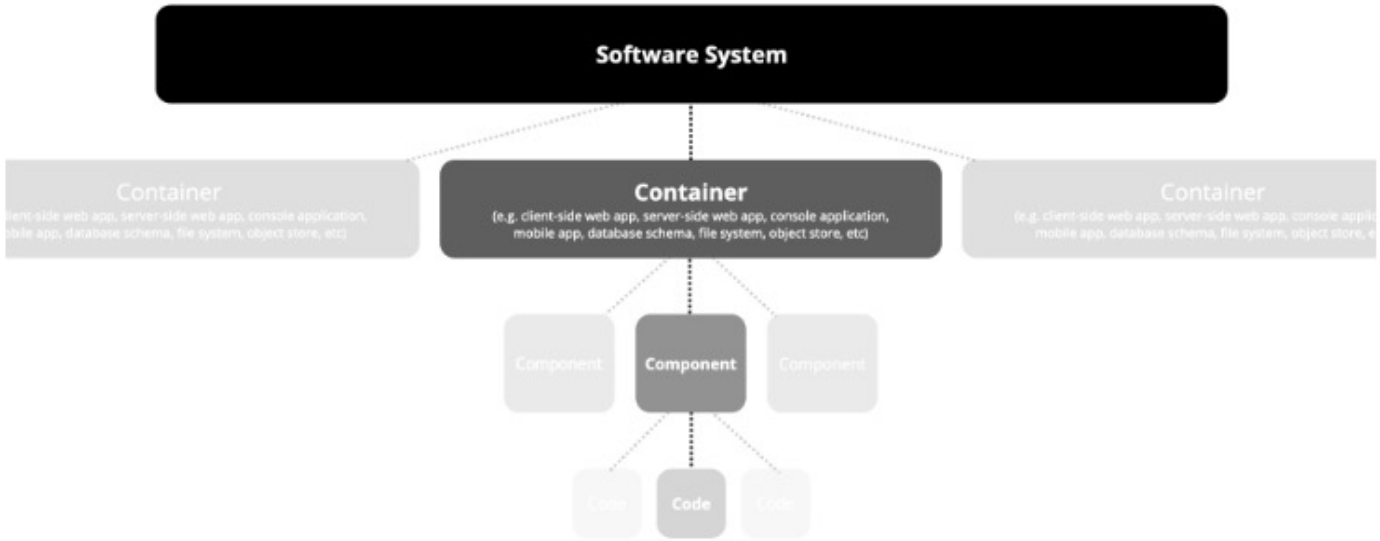
- There isn’t one but multiple software architectures
- High level architectures: closer to requirements
- Low level architectures: closer to implementations

Lack of standardization in modelling architectures

- Simple/informal => Ambiguity in meaning
- Formal (e,g. UML) => Learning curve / understandability

What is C4 ?

- Gives **names** to different design concepts
- Focuses on intuitive visual representations of these concepts
- Defines a set of hierarchical diagram arranged by levels
- Lightweight methodology for visual and verbal communication
- Allows more efficient conversations
- Notation independent
- Tooling independent



C4 Levels

System context level

Showing overall system + users + external systems.

Useful for Business stakeholders, execs and non-tech users

Containers level

Showing major application/components like web apps, APIs, DBs

Useful for developers, tech leads and architects.

Components level

Showing modules/services/classes within a container (e.g. routes, services, repositories)

Mainly for developers.

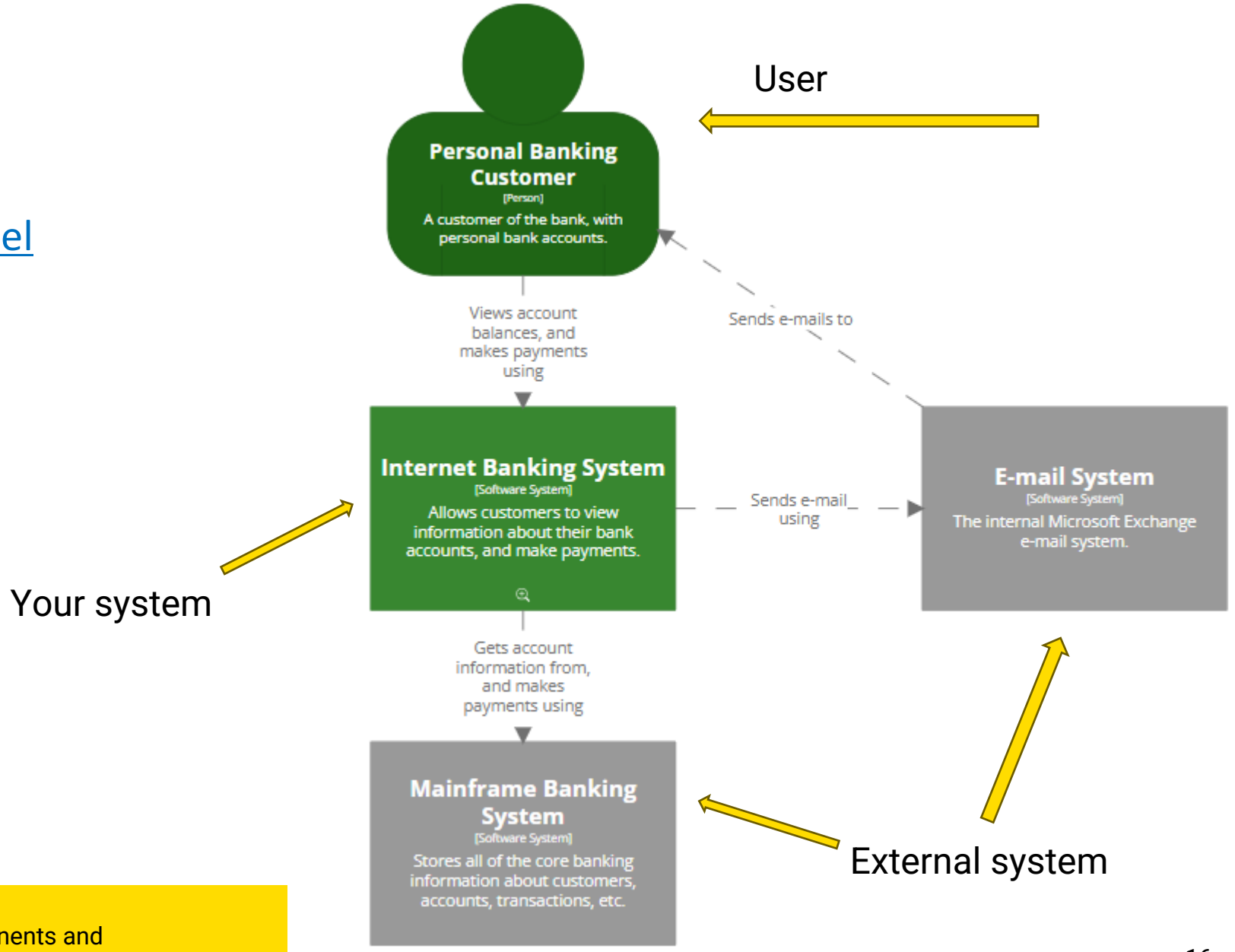
Code level

Level 1

- A context diagram is the most general description of what your system does
- Shows who will use it, and what other systems it will interact with.
- Will help you describe the scope of your project and help you pinpoint who the user is and what problem you're going to solve

Example

From [Example | C4 model]

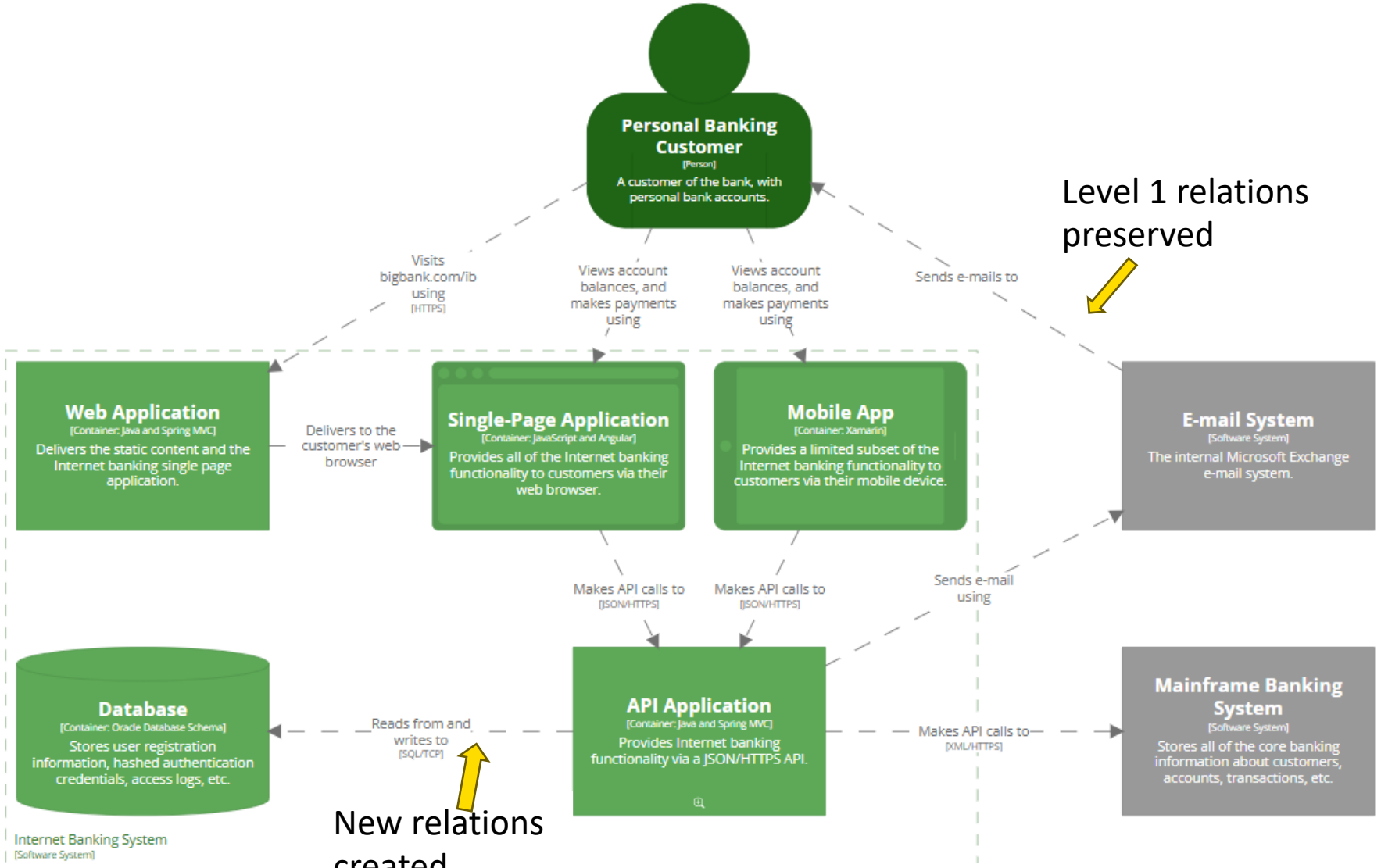


Level 2

- Container diagram takes the first step into describing the software system and shows the APIs, applications, databases, and microservices that the system will use.
- Each of these applications or services is represented with a container and the interactions between them are shown at a high level.

Example

Your system decomposed into several containers



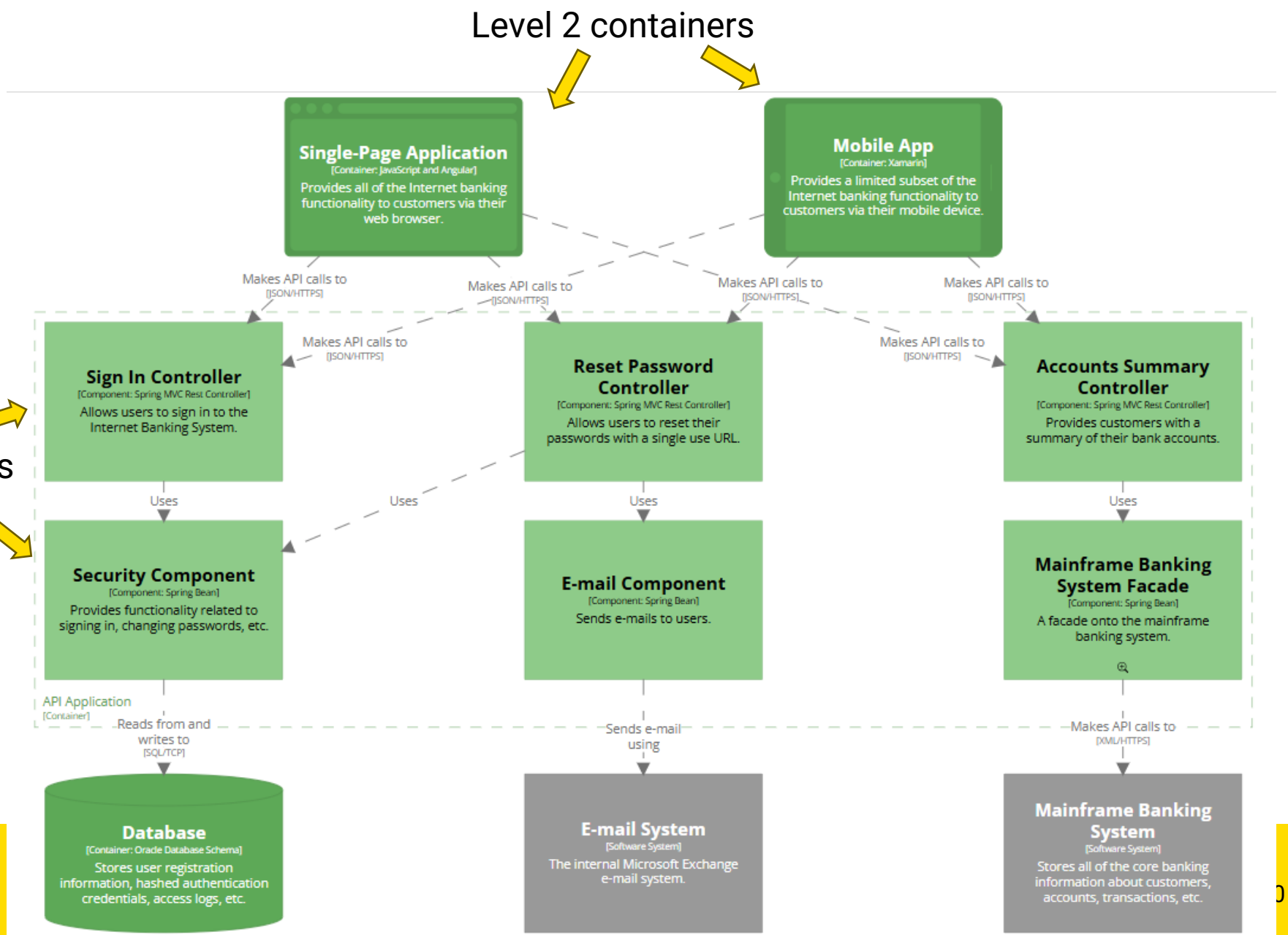
Level 1 relations preserved

New relations created

Level 3

- One step deeper than the container diagram, the component diagram details groups of code within a single container.
- These components represent abstractions of your codebase.
- Comparable to a UML component diagram but follows a less-strict set of “rules” in order to create the software architecture diagram.

Example

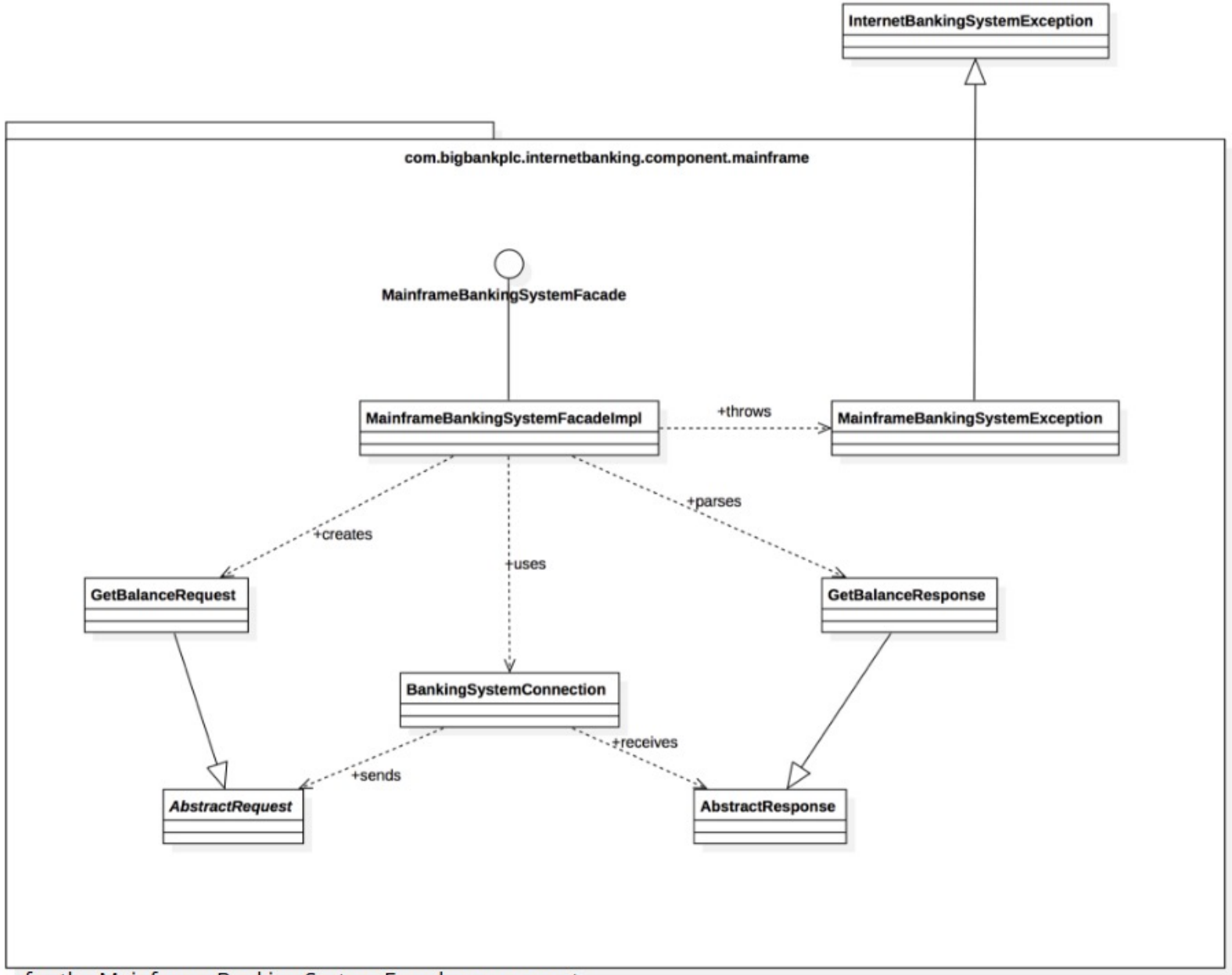


UNSW SYDNEY

Level 4

- Has lots of detail to show how the code of a single component is actually implemented.
- Can use a UML class diagram or entity relationship diagram that describes the component.

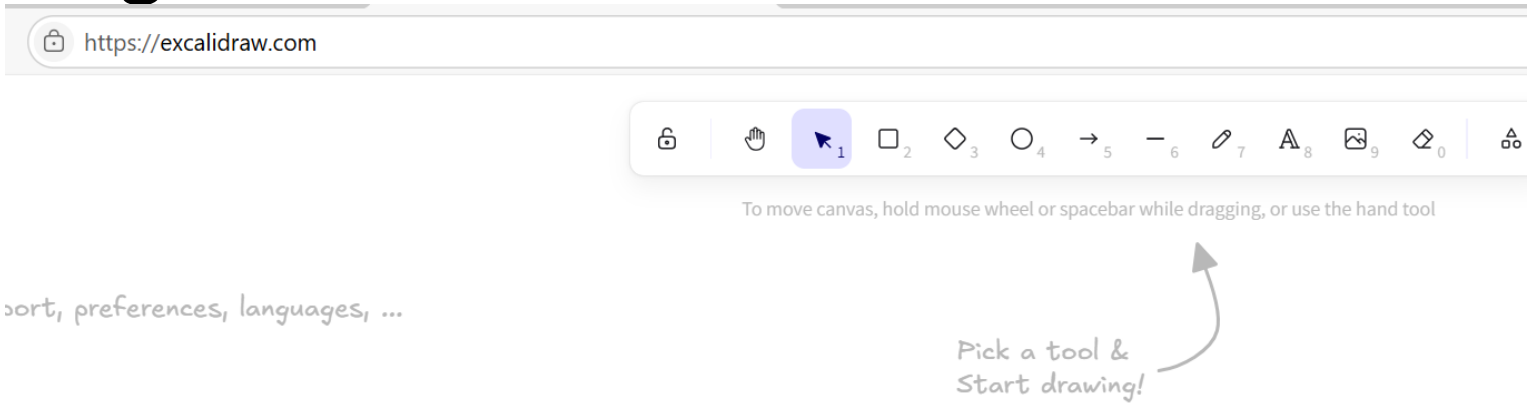
Example

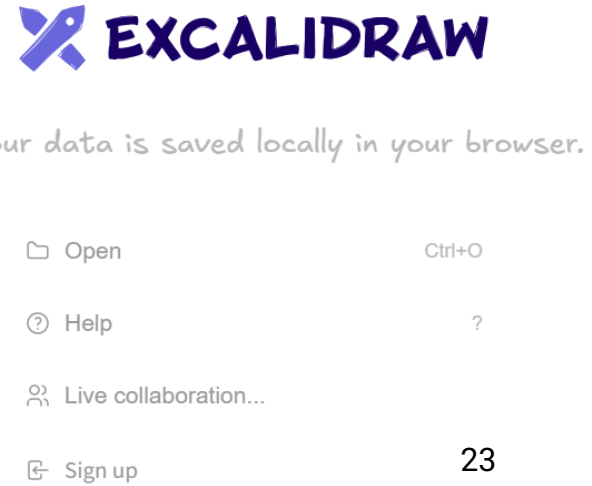


Class diagram for the Mainframe Banking System Facade component

Recommended Modelling tool

Simple one
https://excalidraw.com/





1. Enter C4

Excalidraw Libraries

A directory of public libraries that you can easily add to Excalidraw.

Follow the instructions if you want to **add your own library** into this list.

All the following libraries are under MIT License.

Sort By · New · Updated · Total Downloads · Downloads This Week · Author · Name

C4

(tip: you can type anywhere to start searching)

Hexagonal Architecture

@Armando Cordova Pelaez

7582

Created: 24 Sep 2021

Useful to diagram and learn more about Hexagonal (aka Ports and Adapters) Architecture by Alistair Cockburn and implementation by Jakub Nabrdalik. More information: https://gist.github.com/corlaez/32707a1c41485d056c00251206435c89

Add to Excalidraw | Download

C4 Architecture

@Dmitry Burnyshev

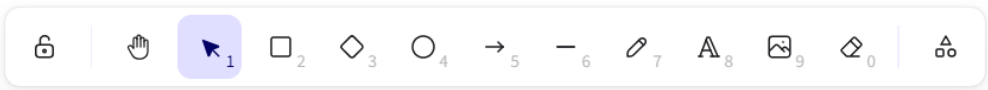
3080

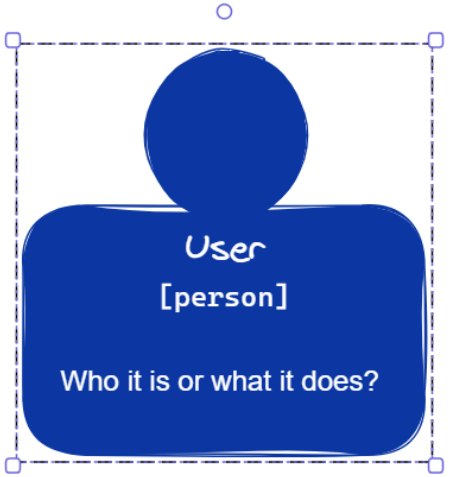
Created: 24 May 2022

C4 Simon's Brown concept elements based on https://c4model.com/

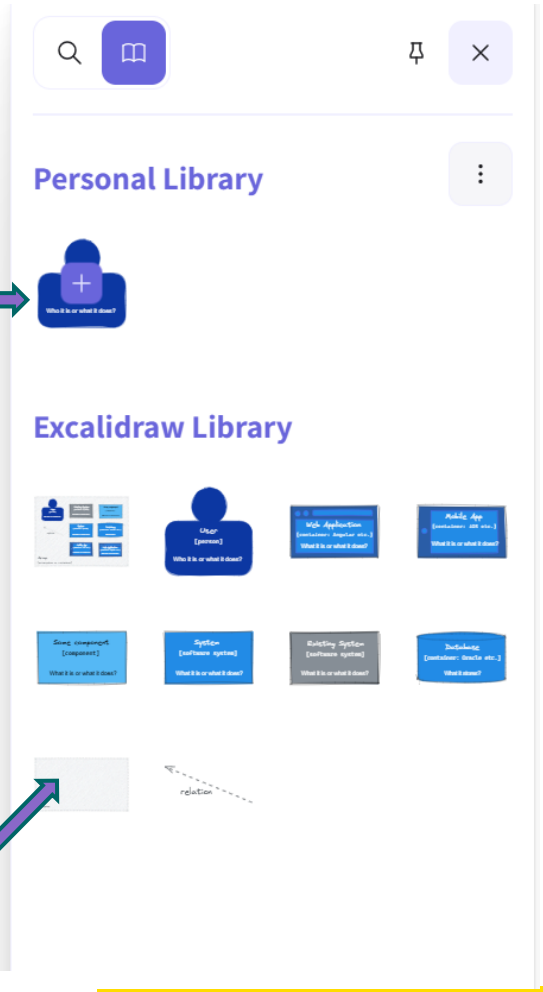
Items: C4 elements, Person, Web App, Mobile App, Component, System, Existing System, Database, Group, Relation

2. Add to Excalidraw



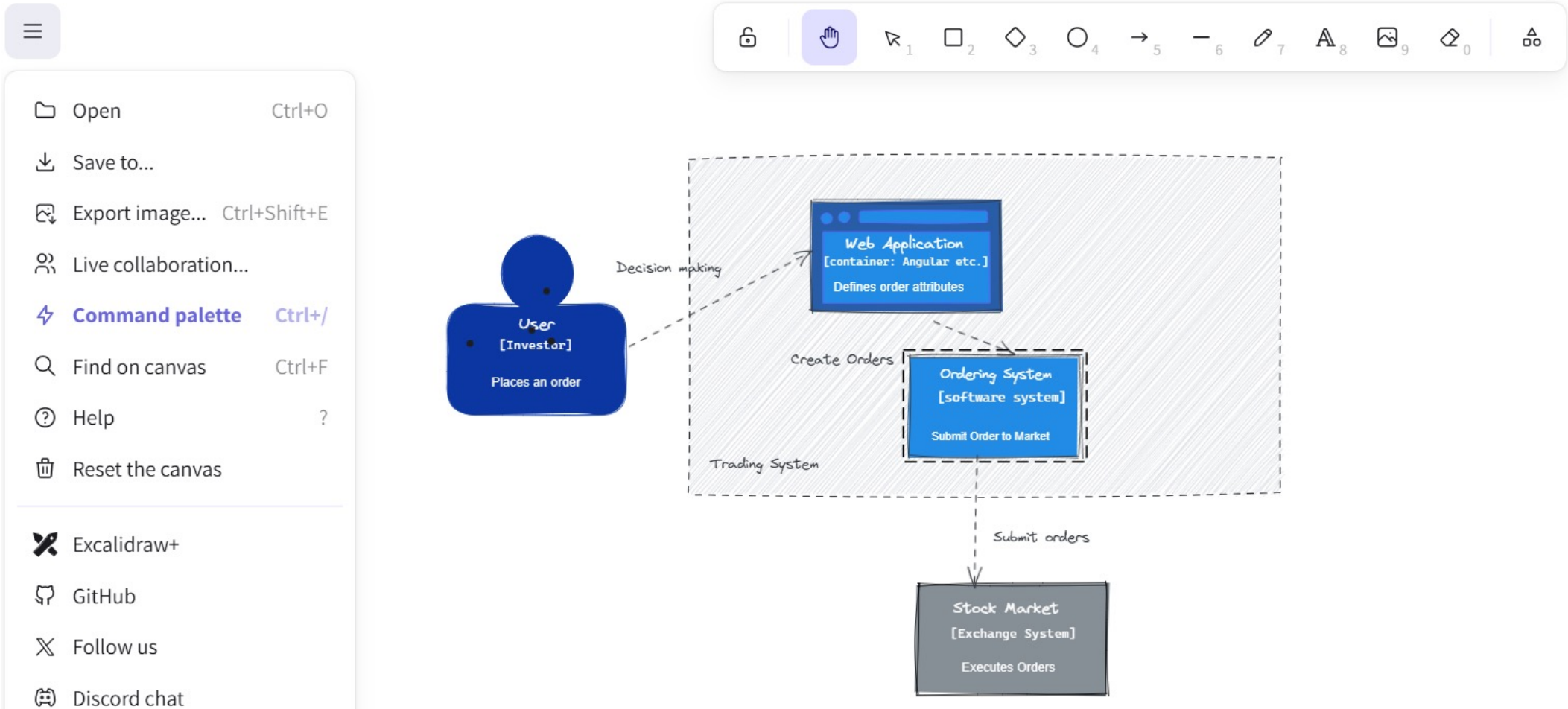


You can create your own shapes



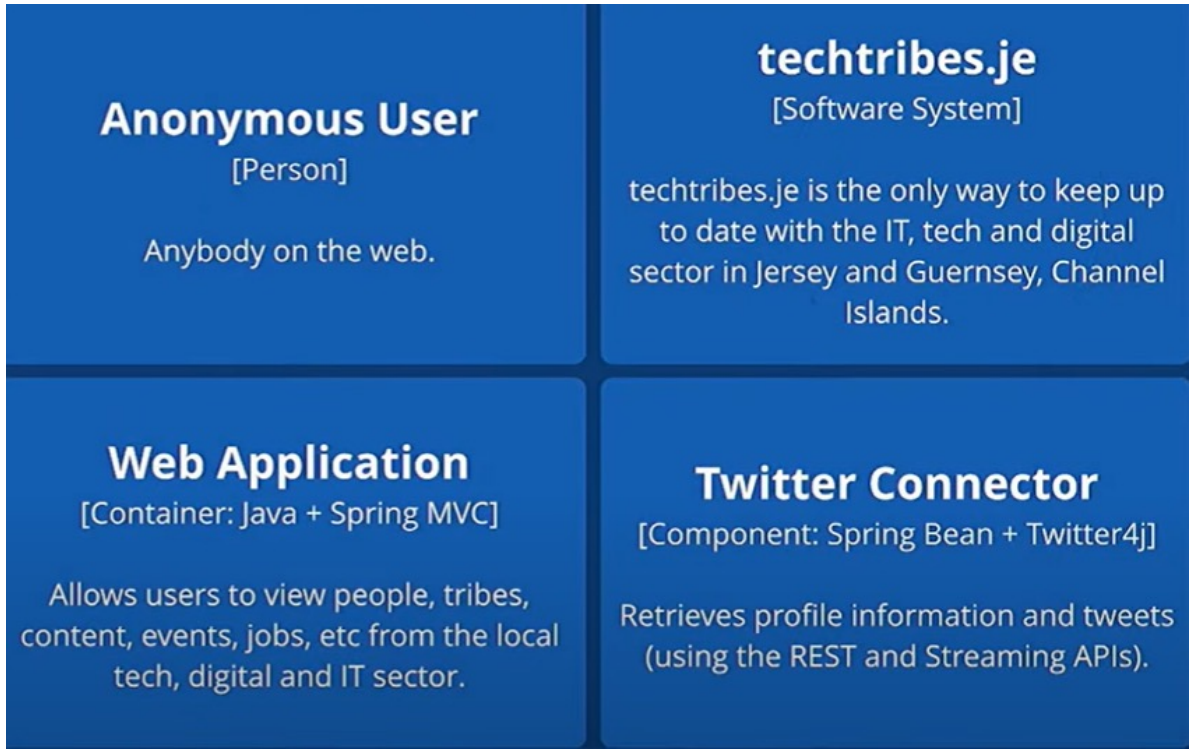
You can drag shapes into the canvas

You can use these shapes

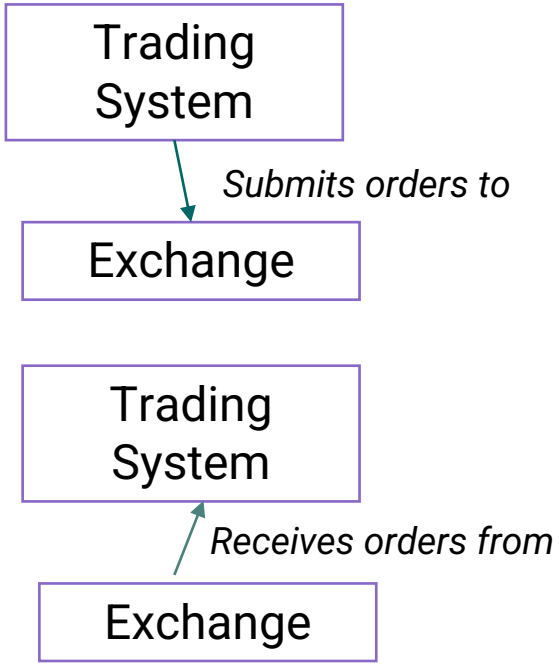

Open Ctrl+O
Save to...
Export image... Ctrl+Shift+E
Live collaboration...
Command palette Ctrl+/
Find on canvas Ctrl+F
Help ?
Reset the canvas
Excalidraw+
GitHub
Follow us
Discord chat
Sign up
Theme
English
Canvas background
User
[Investor]
Places an order
Decision making
Web Application
[container: Angular etc.]
Defines order attributes
Create Orders
Ordering System
[software system]
Submit Order to Market
Trading System
Submit orders
Stock Market
[Exchange System]
Executes Orders

Good modelling practices

- Add a title to your diagram
- Avoid acronyms for business terms
- Consistent naming of components



Good modelling practices (cont.)



- Lines
  - Clearly labelled
  - Undirectional (follows words in boxes)
- Legend
  - Use it for additional shapes/colours you introduce
  - Also additional icons that describe components (AWS-style)
  - Use it to enhance only (if remove them, diagram still makes sense)
- A good diagram should be self-explanatory
- More details in Simon Brown's video at: https://www.youtube.com/watch?v=x2-rSnhpw0g&t=785s

Review checklist | C4 model

C4 Model Checklist

Software architecture diagram review checklist

General

Does the diagram have a title?	Yes	No
Do you understand what the diagram type is?	Yes	No
Do you understand what the diagram scope is?	Yes	No
Does the diagram have a key/legend?	Yes	No

Elements

Does every element have a name?	Yes	No
Do you understand the type of every element? (i.e. the level of abstraction; e.g. software system, container, etc)	Yes	No
Do you understand what every element does?	Yes	No
Where applicable, do you understand the technology choices associated with every element?	Yes	No
Do you understand the meaning of all acronyms and abbreviations used?	Yes	No
Do you understand the meaning of all colours used?	Yes	No
Do you understand the meaning of all shapes used?	Yes	No
Do you understand the meaning of all icons used?	Yes	No

Resources

C4 Model: https://c4model.com/abstractions

Tutorial video: https://www.youtube.com/watch?v=x2-rSnhpw0g&t=785s&ab_channel=AgileontheBeach

Articles

- Should you use the C4 model for system architecture design? https://icepanel.medium.com/c4-model-for-system-architecture-design-225e00ebbd9
- C4 model for system architecture design https://icepanel.medium.com/c4-model-for-system-architecture-design-225e00ebbd9

Other tools

- Flowchart maker https://app.diagrams.net/
- Open source tool https://plantuml.com/
- Lucid Charts https://www.lucidchart.com/blog/c4-model
- Gliffy https://www.gliffy.com/blog/c4-model

END