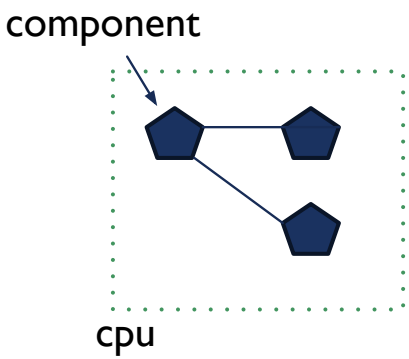
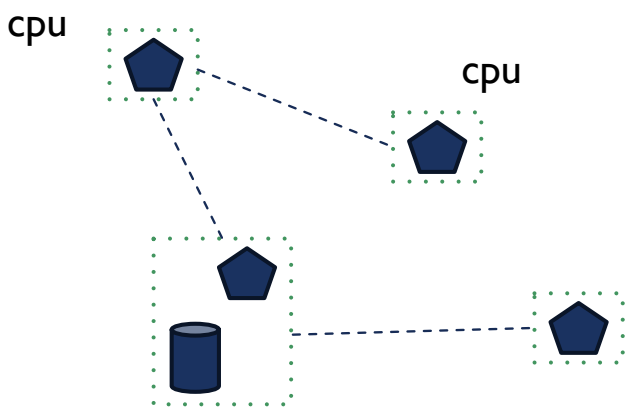
BASIC INTERACTION PATTERNS

WITH (SOME) EXAMPLE

INTRODUCTION

- A software is made of several ‘components’
- The interaction among components can be either synchronous or asynchronous
- Interacting components can run on the same CPU or on different CPUs
- A component can be stateless or stateful





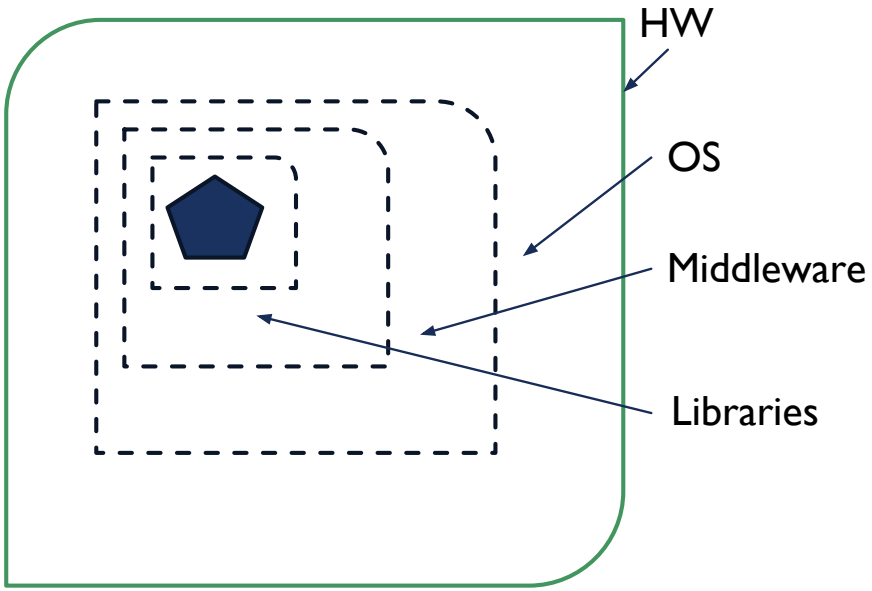
If components run on different CPU they

1. can make progress in parallel (true)
2. can only interact via messages (no shared memory)
3. Do not have a global clock

Then the system is distributed (network can be Partitioned, component can become not Available, data can be not Consistent ☐CAP theorem applies)

MORE ON COMPONENTS

- Any component needs some surrounding support, both hardware or sw, in order to run
- Nevertheless, for now focus on components



BASIC INTERACTION PATTERNS AMONG SW COMPONENTS

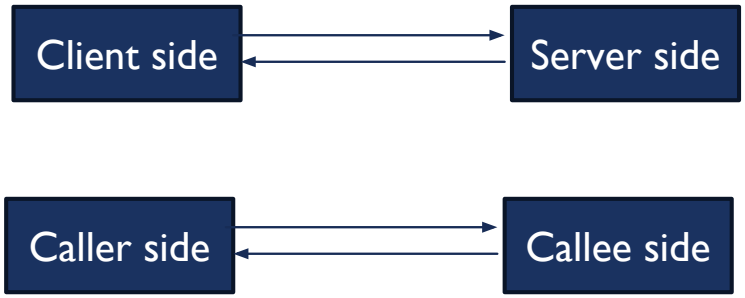
- Short review of the basic interaction patterns (ways) among components in abstract way and providing some technological mapping. We assume no failures

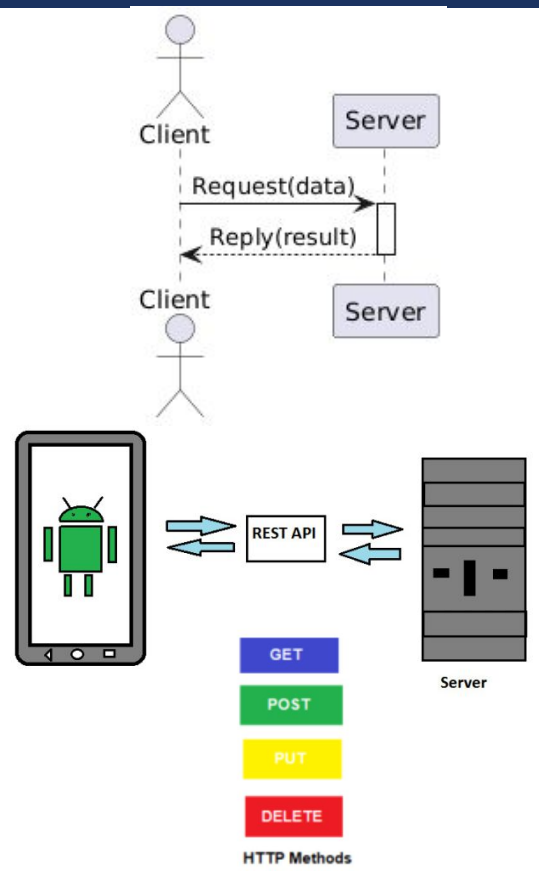
- Interaction patterns:
- Request/Response
- One-Way Message (Fire and Forget)
- Publish/Subscribe
- Event Notification

- Computational Graphs

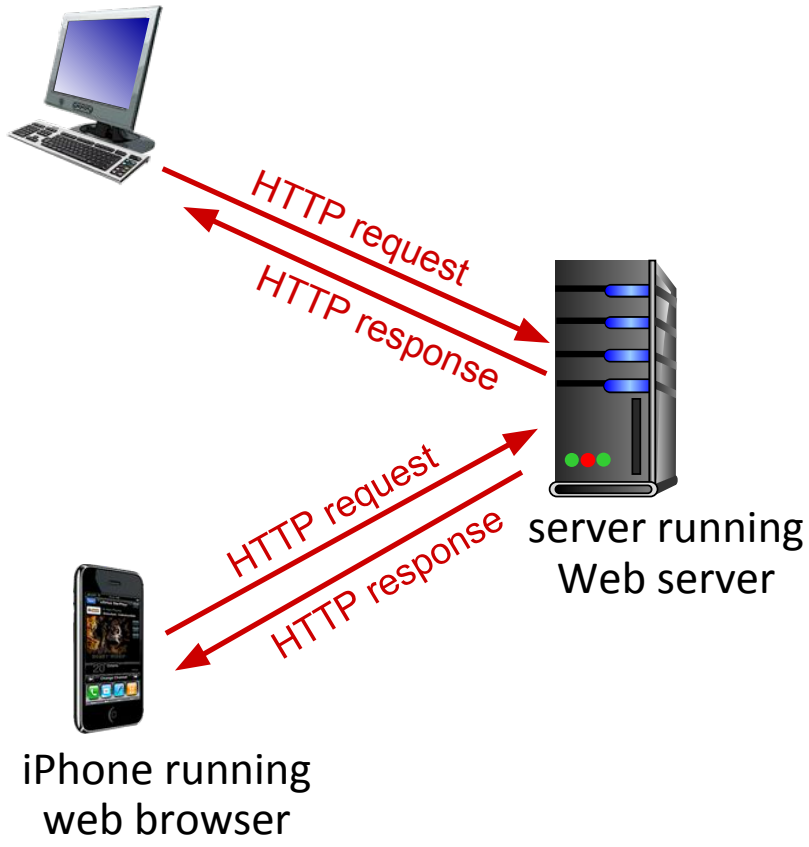
REQUEST/RESPONSE

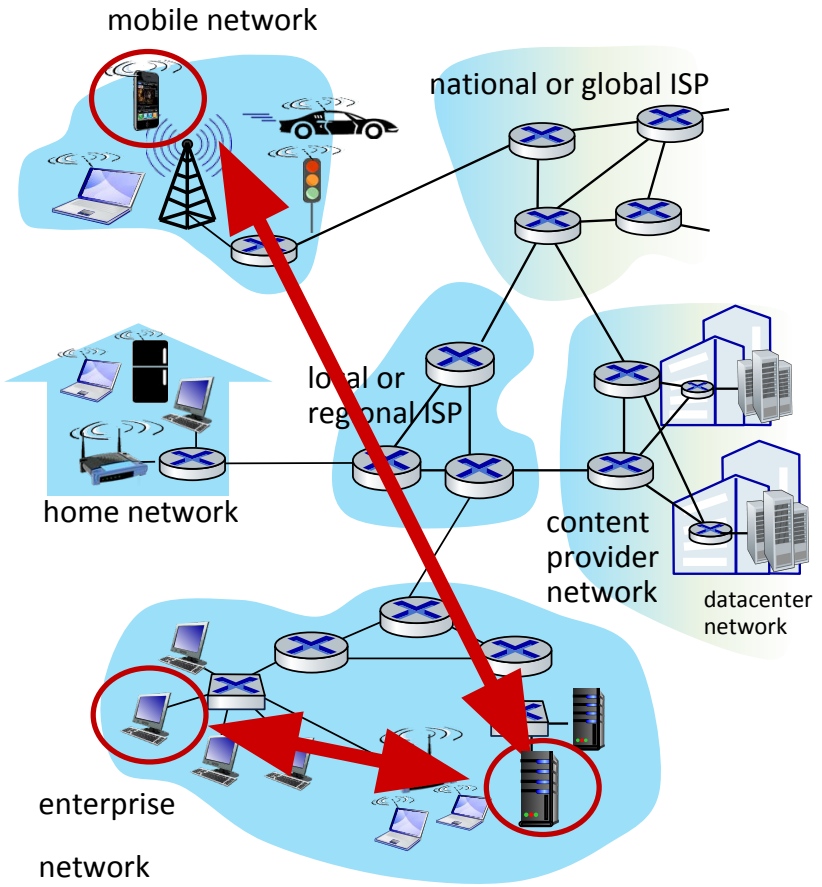
- Perhaps the most common pattern
- Synchronous: caller makes a request and then waits for response
- Examples:
- A function that calls another function inside the same progra,
- An app that sends sensor data to REST API
- A frontend microservice queries backend DB via API



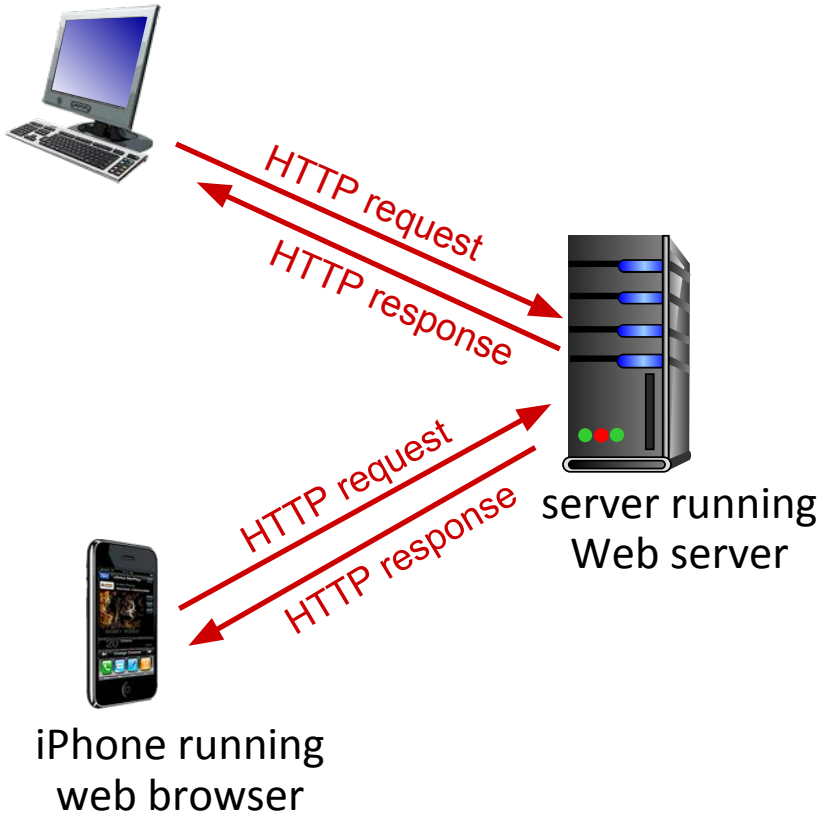


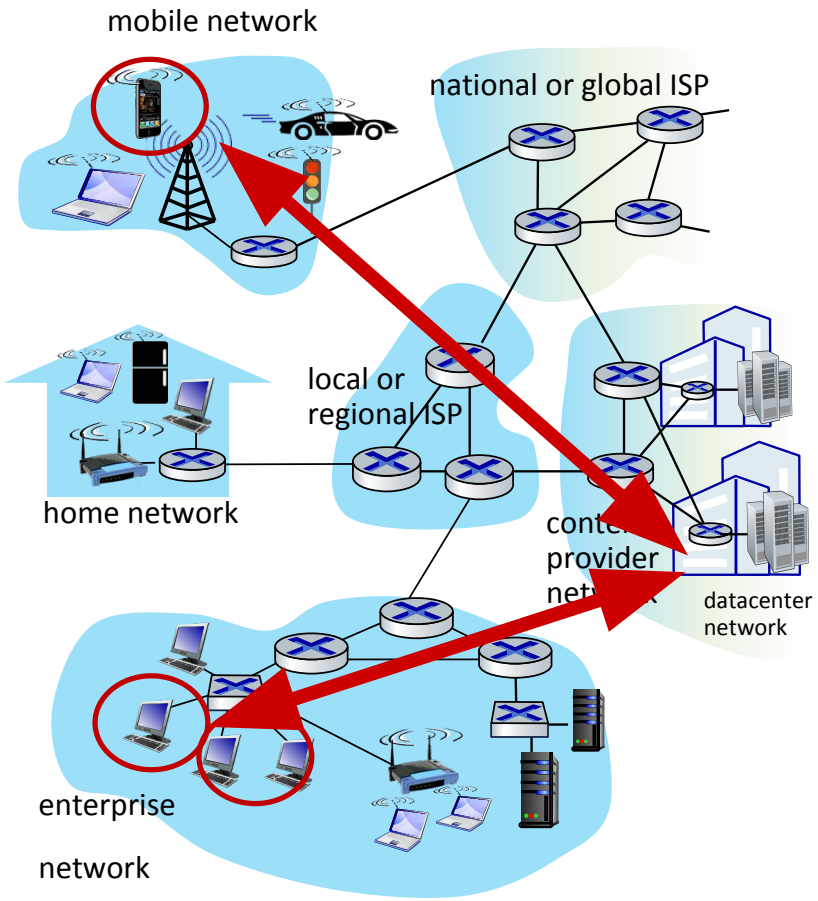
REQUEST/RESPONSE (SOME MORE IMPLEMENTATION DETAILS)





REQUEST/RESPONSE (SOME MORE IMPLEMENTATION DETAILS)

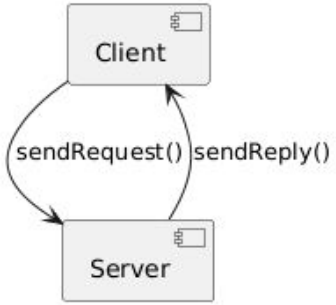


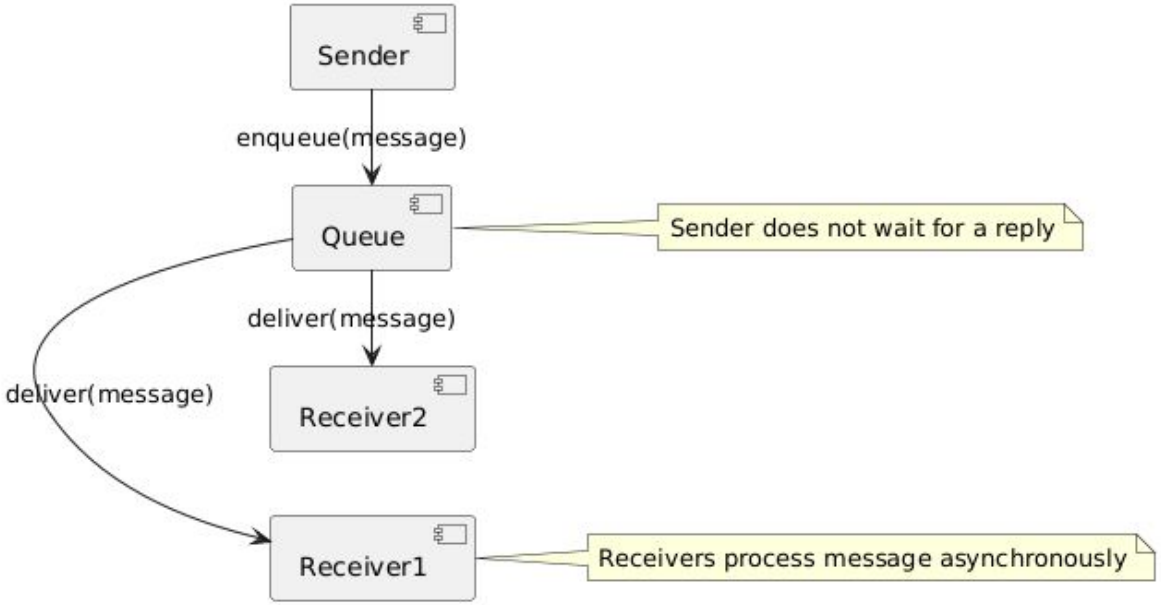


ONE-WAY MESSAGE (FIRE AND FORGET)

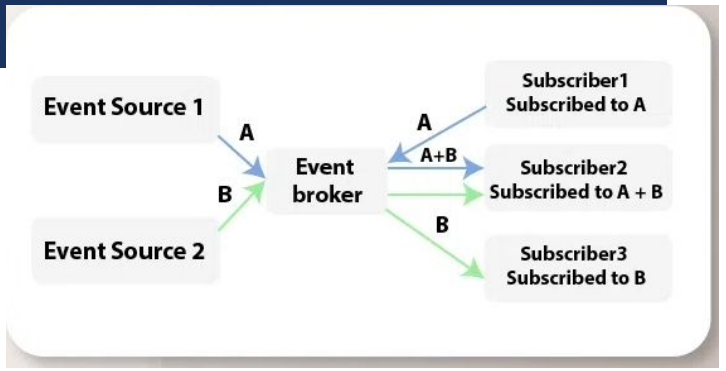
- **Fire-and-forget**
- Component sends an asynchronous message (to a queue or intermediary) without waiting for response
- Android: app sends logs or sensor readings to a remote queue without waiting for the reply, e.g. elaboration

ONE-WAY MESSAGE (FIRE AND FORGET)





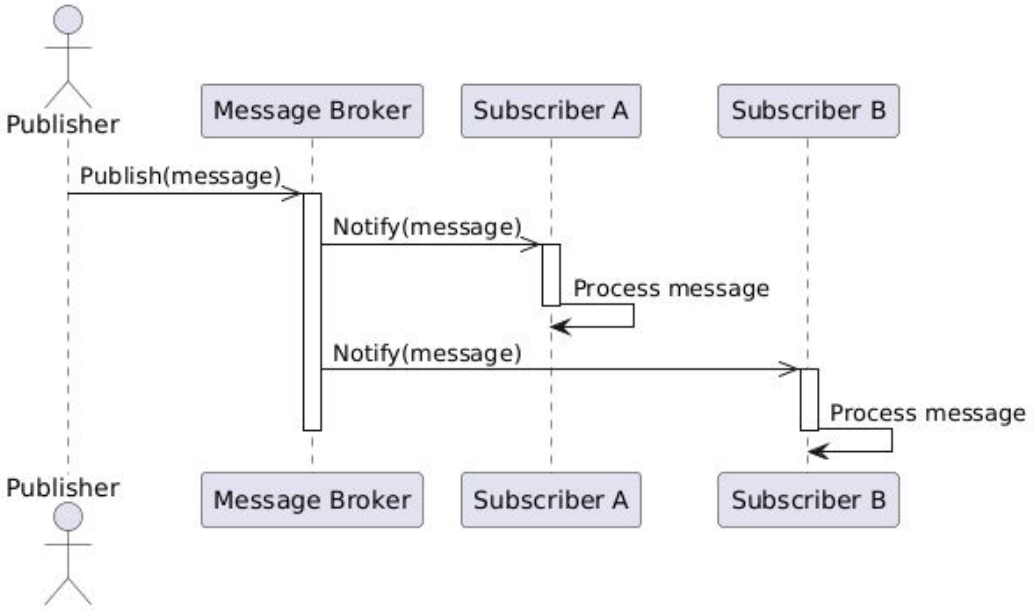
PUBLISH/SUBSCRIBE



- **Publisher** sends events to a **broker** on a specific **topic**
- **Subscribers** receive notifications without direct knowledge of publisher

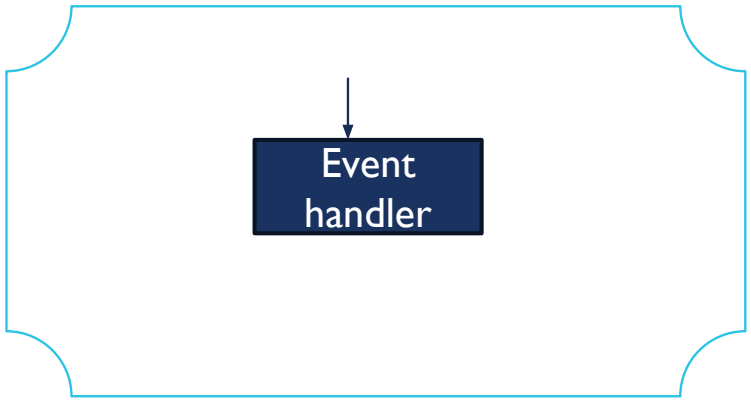
- Example, Android: push information on topics (e.g., weather alerts) using firebase
- Function that subscribes (register) to sensor events (event handler)

PUBLISH/SUBSCRIBE

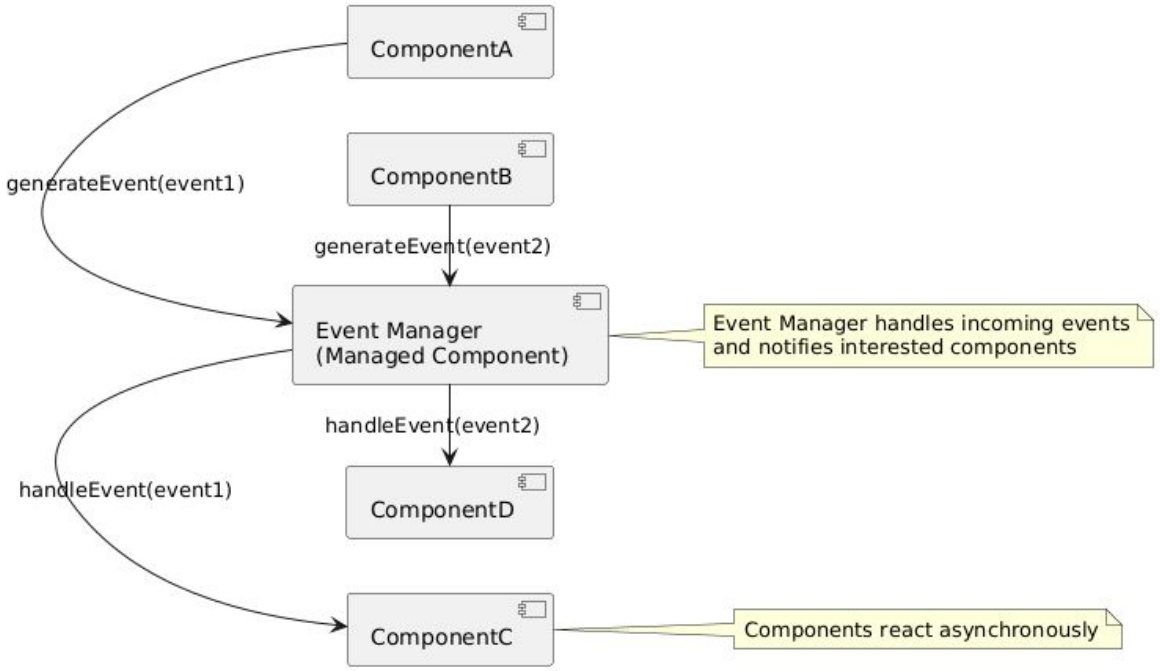


EVENT NOTIFICATION

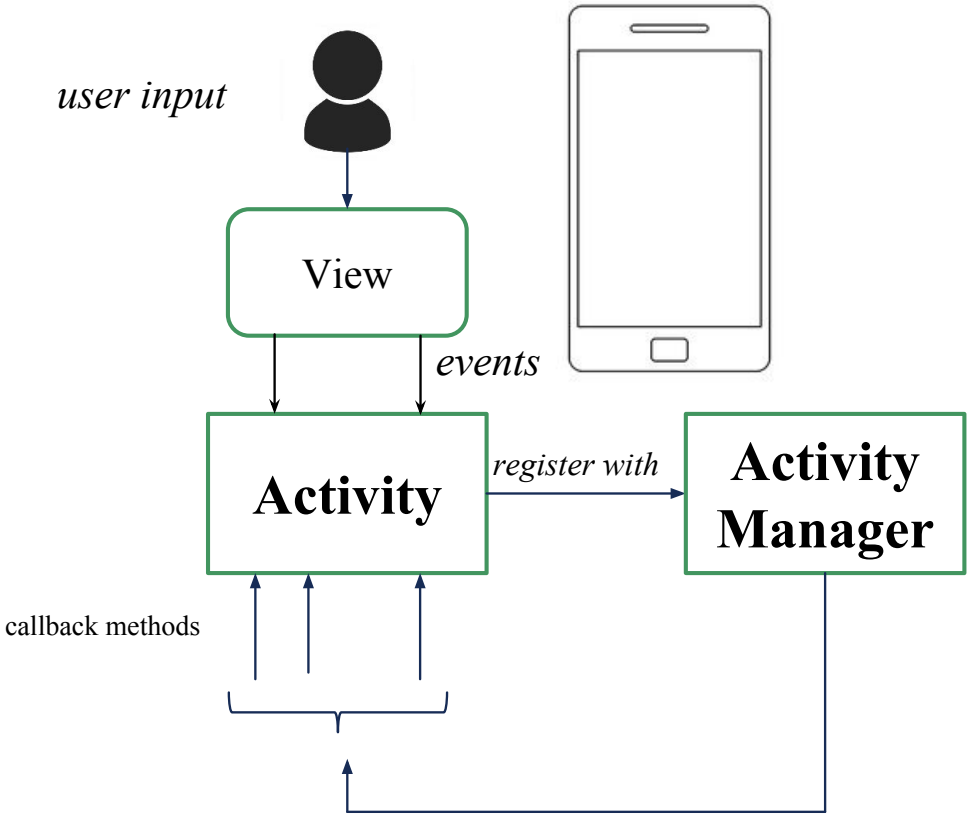
- Passive component that reacts only to events (sometimes called event-handler)
- Foundation of event-driven programming
- The component is dormant until the event occurs (we say it is a managed component)
- Needs for a surrounding software
- Android: All android sw is event-driven
  - UI sw that updates visual components when events (e.g, touch)



EVENT NOTIFICATION



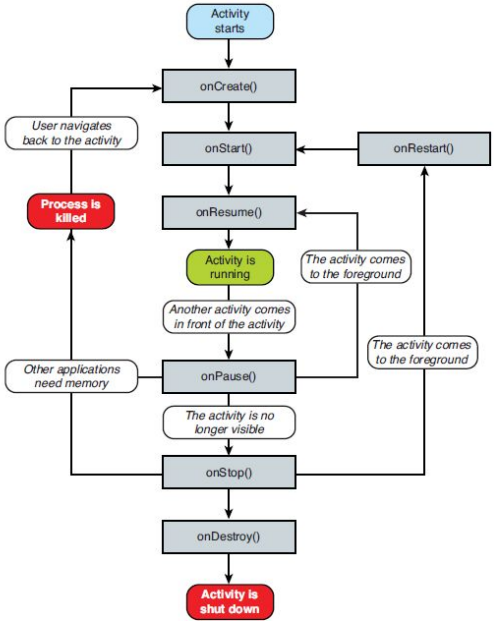
EXAMPLE OF COMPONENT IN ANDROID

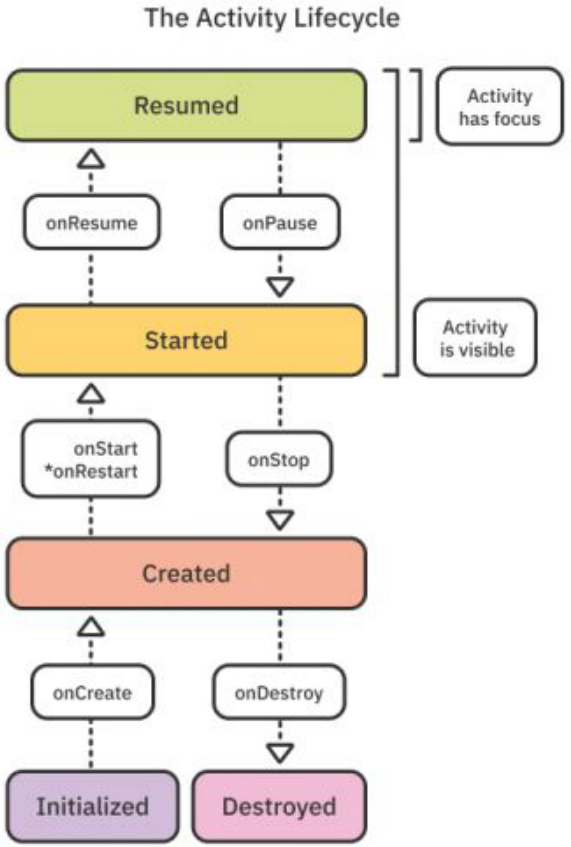


- An Activity is a fundamental software component in any Android application that reacts to events, either generated by the Android framework or by other components.
- Each Activity responds to a set of predefined events (inherited from its base class) by implementing callback methods (*onCreate(),onResume()*…).
- In addition, it registers itself with a system manager and declares which system-wide events it can handle.
- These events are communicated through special messages called Intents.

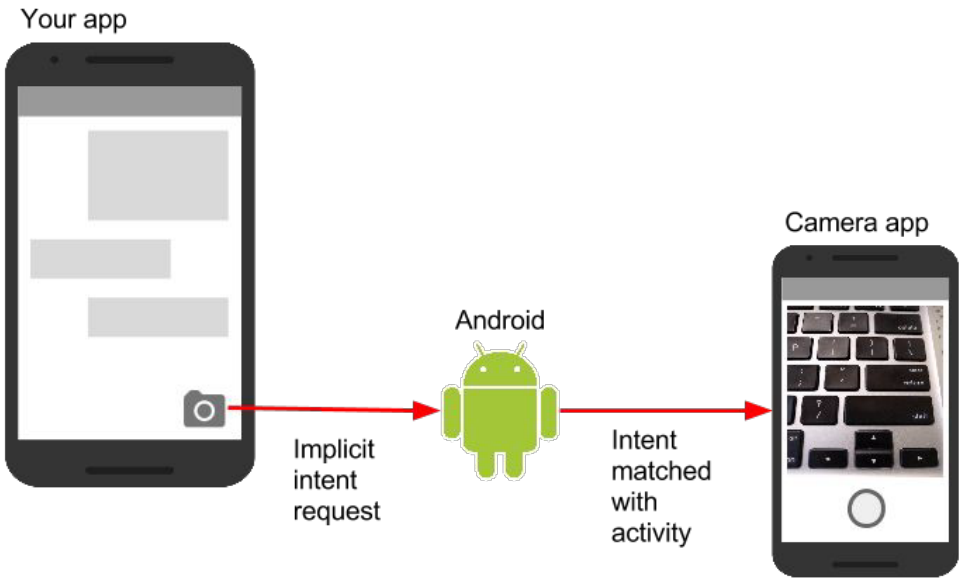
EXAMPLE: ACTIVITY LIFECYCLE

- A managed component in general has a state and reacts differently based on the current state
- The state may change when an event occurs
- The state diagram describes the lifecycle of the component



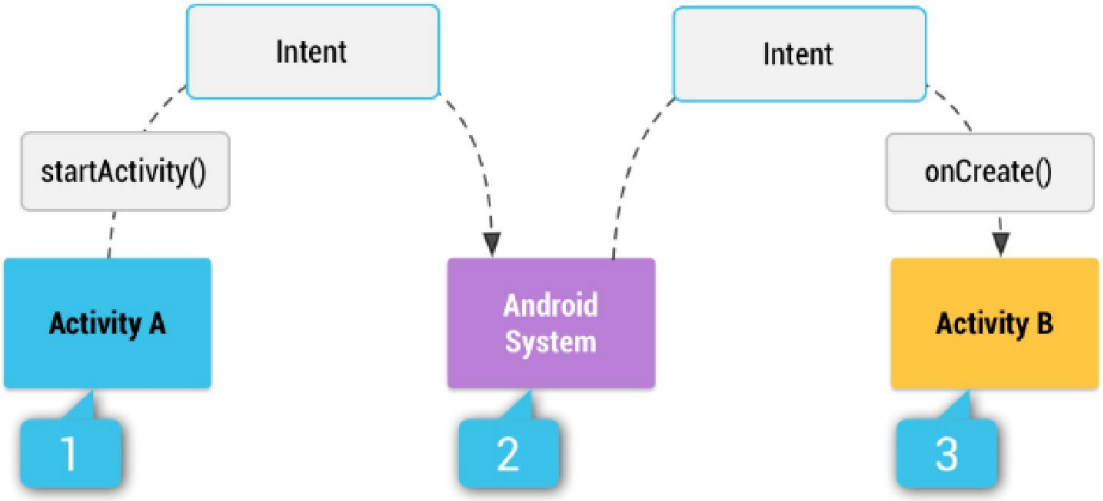


EXAMPLE: INTENETS IN ANDROID



- Implicit Intent are asynchronous messages without a specific destination, carrying a description of the action that the receiving component should perform when process the Intent.
- At registration time, an Activity declares its capabilities to perform specific actions (into the manifest file) and can therefore become eligible for Intents that require those capabilities.
- Intents are not confined to a single application but are system-wide, so other applications can also respond to them.
- They are the glue that integrate another application in the normal navigation flow of the current one

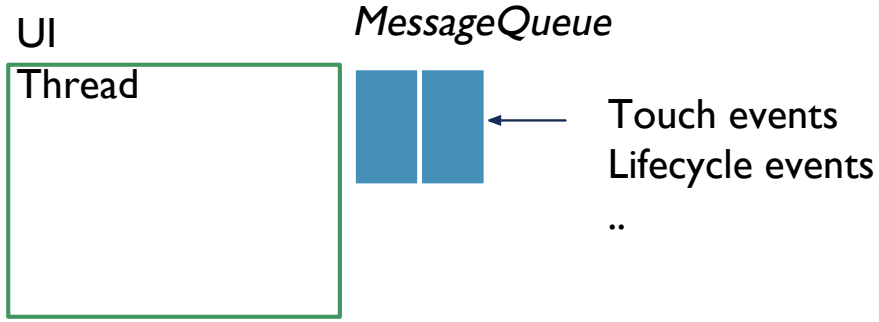
INTENT: BASE CASE



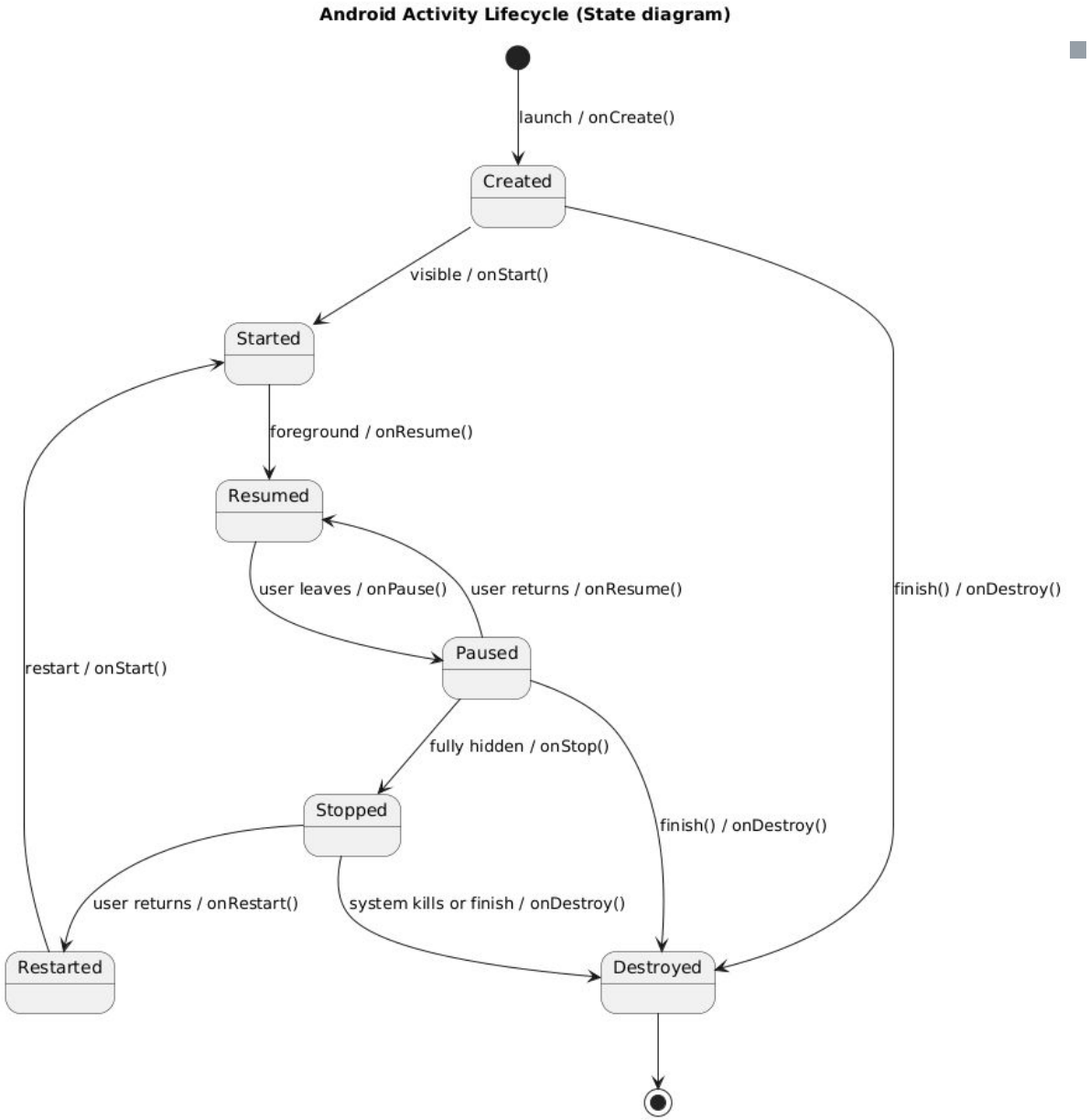
1. A Creates the intent and calls startActivity with a reference to B
2. The ‘android system’ checks if such an Activity is installed
3. The ‘android system’ calls the B’s onCreate method

EXAMPLE: VIEW UPDATES

- The software inside an activity runs on a thread, called **UI thread** or **main thread**
- This software includes handlers of touch events, that are dispatched through a message queue
- View updates is done by an *onDraw* method called every approx. 16.7 ms, which provides a refresh rate of 60 Frame per Second (FPS)



the time distance among two *onDraw* approx 16 ms

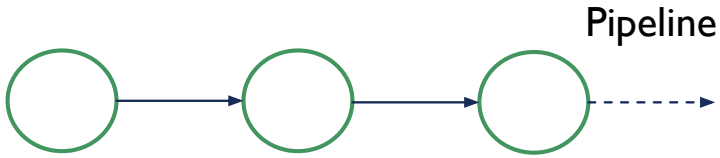

Android Activity Lifecycle (State diagram)
launch / onCreate()
Created
visible / onStart()
Started
foreground / onResume()
Resumed
user leaves / onPause()
user returns / onResume()
Paused
restart / onStart()
fully hidden / onStop()
Stopped
finish() / onDestroy()
user returns / onRestart()
system kills or finish / onDestroy()
Restarted
Destroyed
finish() / onDestroy()

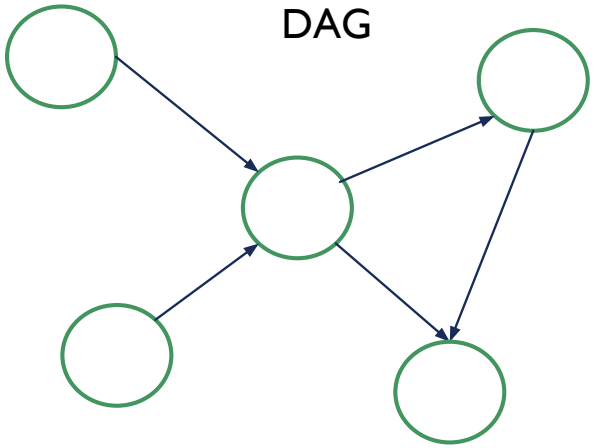
COMPUTATIONAL GRAPH

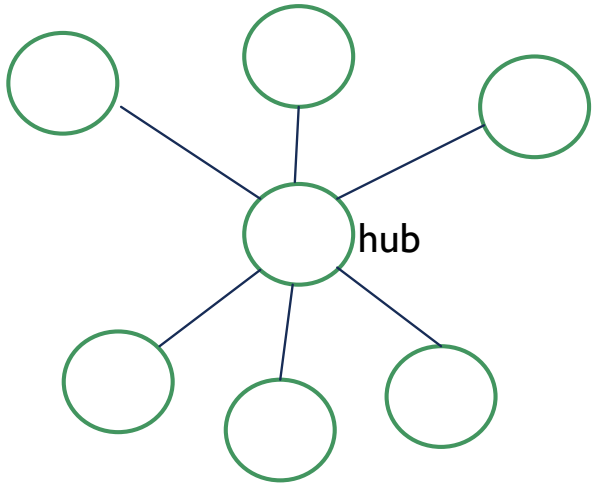
- Basic interaction patterns can be composed into **computational graphs**
- Nodes = components
- Edges = interactions
- Graph types:
  - Linear pipeline
  - Hub
  - DAG (with branches and joins)

COMPUTATIONAL GRAPH

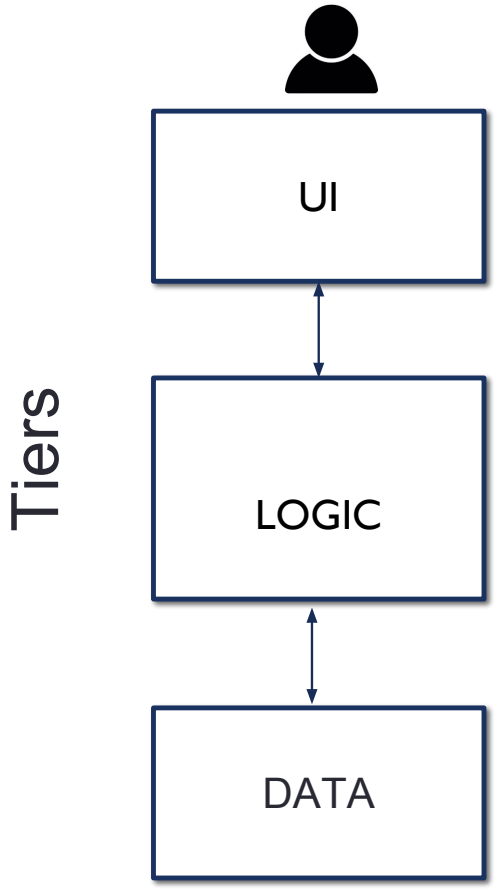
- Graph types:
  - Linear pipeline
  - Hub (star topology)
  - DAG with branches and joins



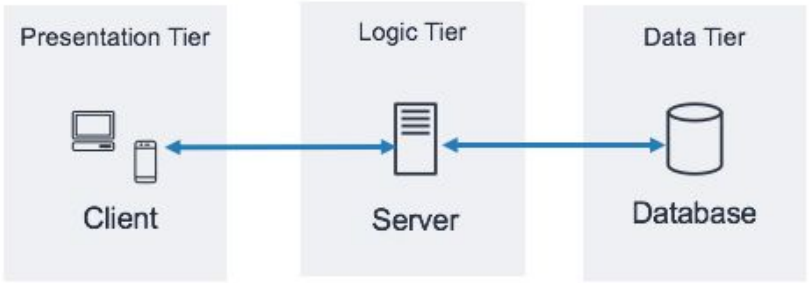


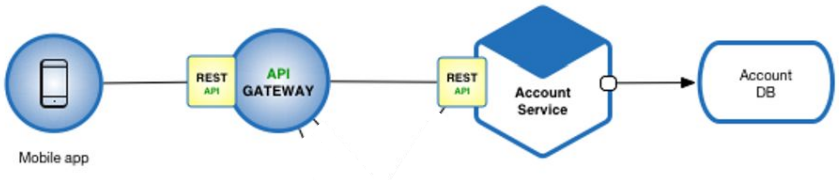


EXAMPLE OF LINEAR COMPUTATION GRAPH



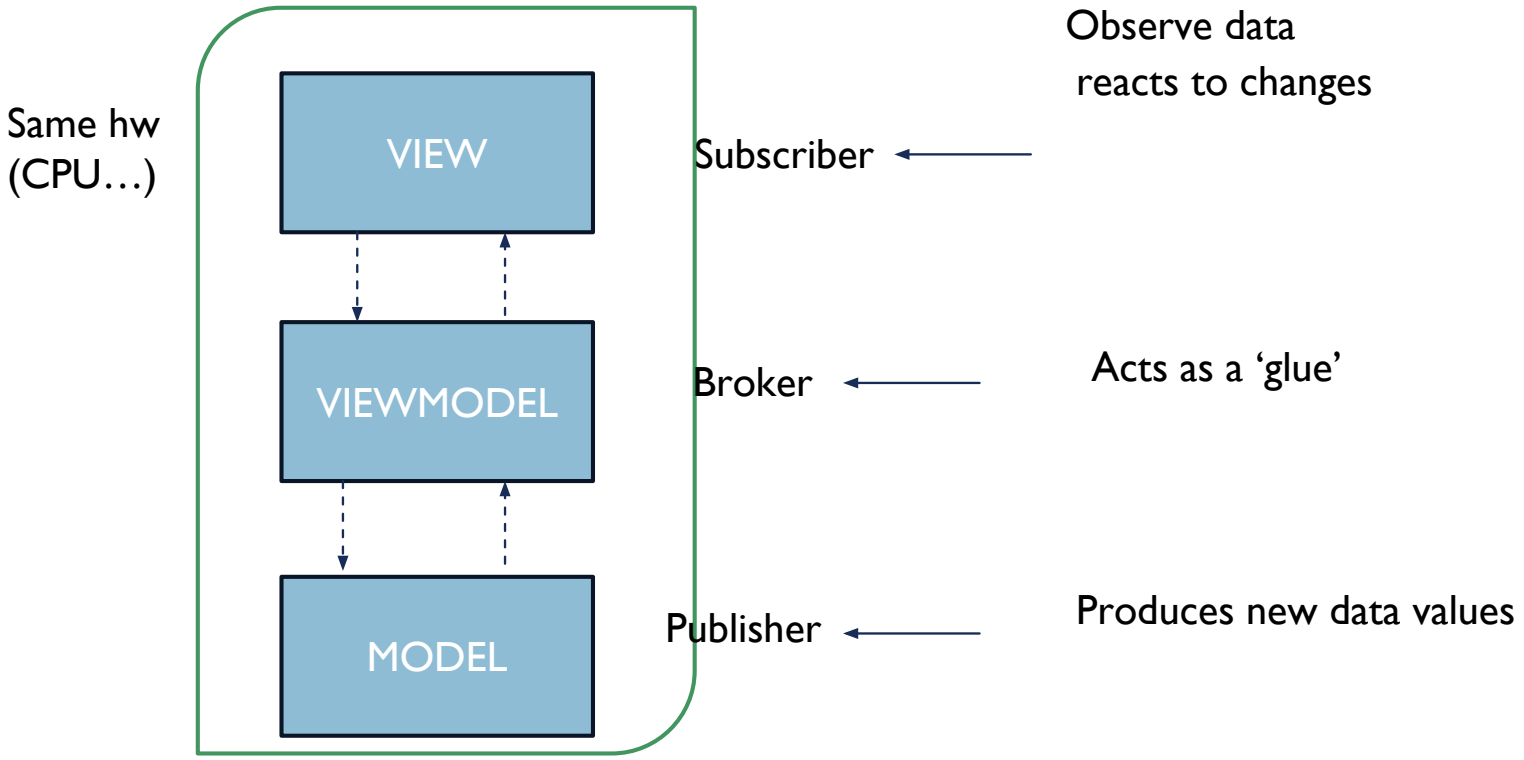
- User Interaction
- What the application does
- How data are stored



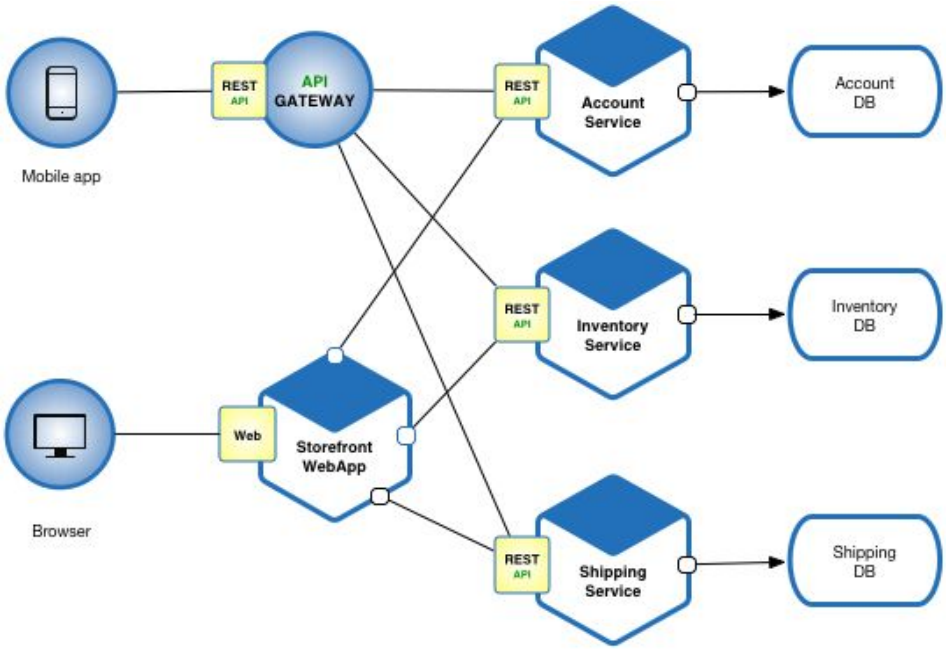


EXAMPLE: MODEL-VIEW-VIEWMODEL (MVVM)

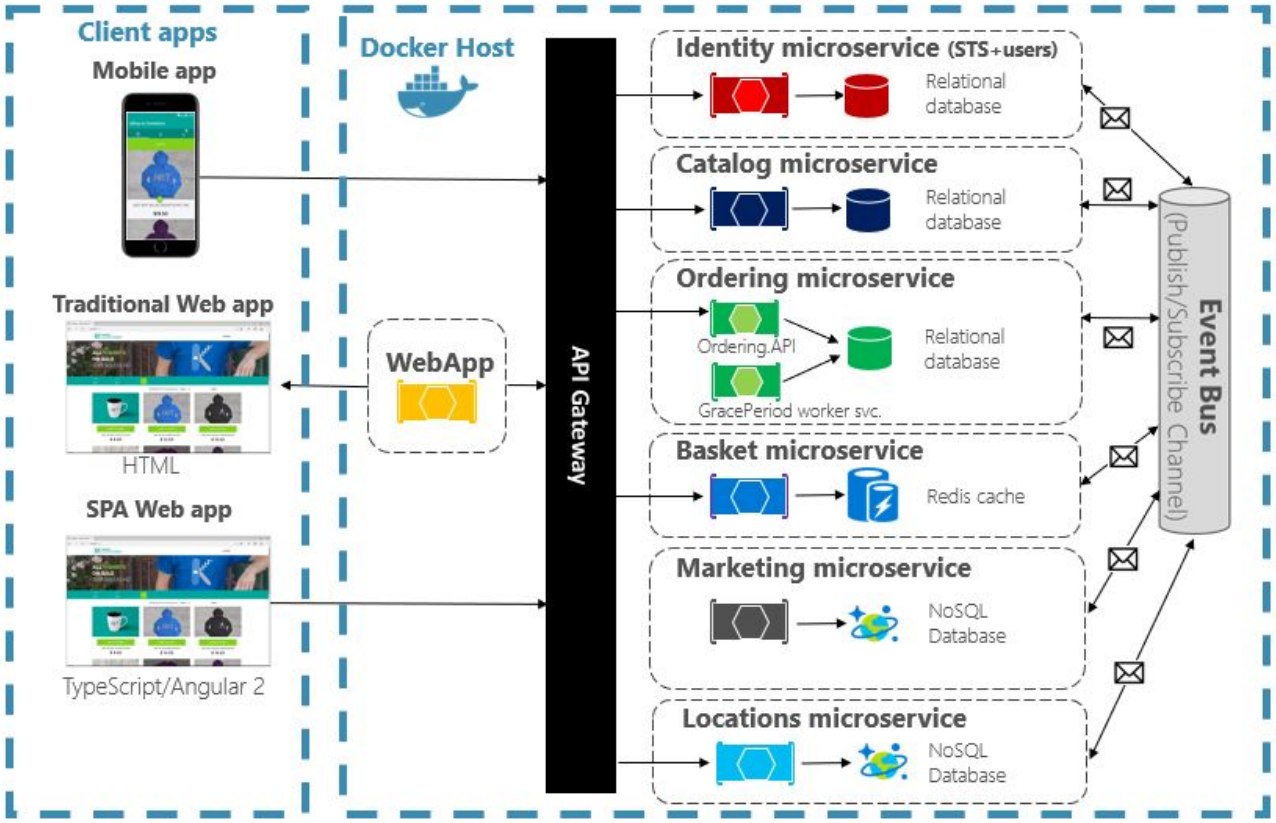
- This is a pattern suggested to organize the software of an android app, which is based on a pub/sub interaction pattern



HUB


Mobile app
REST API
API GATEWAY
REST API
Account Service
Account DB
REST API
Inventory Service
Inventory DB
Browser
Web
Storefront WebApp
REST API
Shipping Service
Shipping DB

GENERAL GRAPH

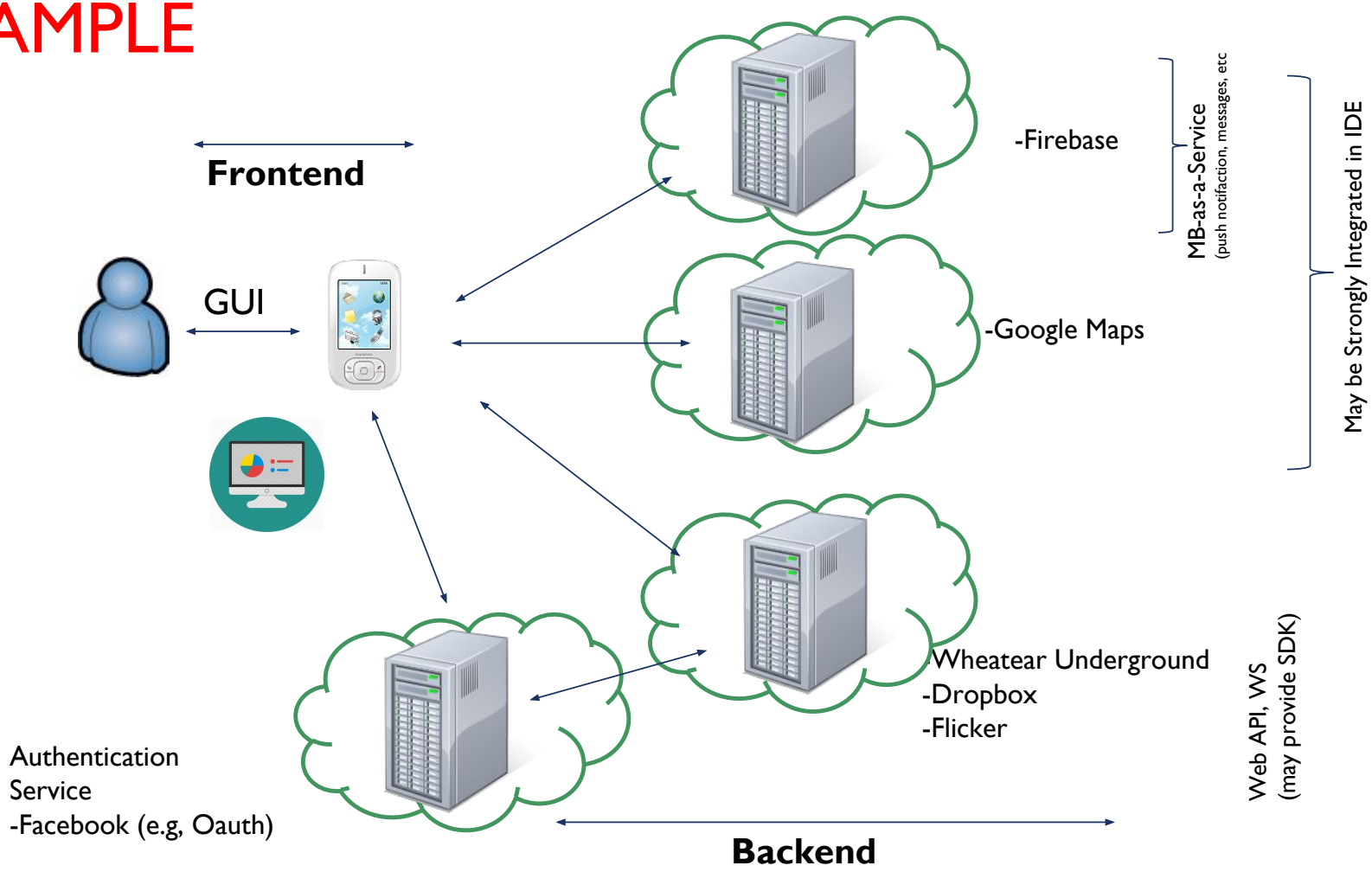
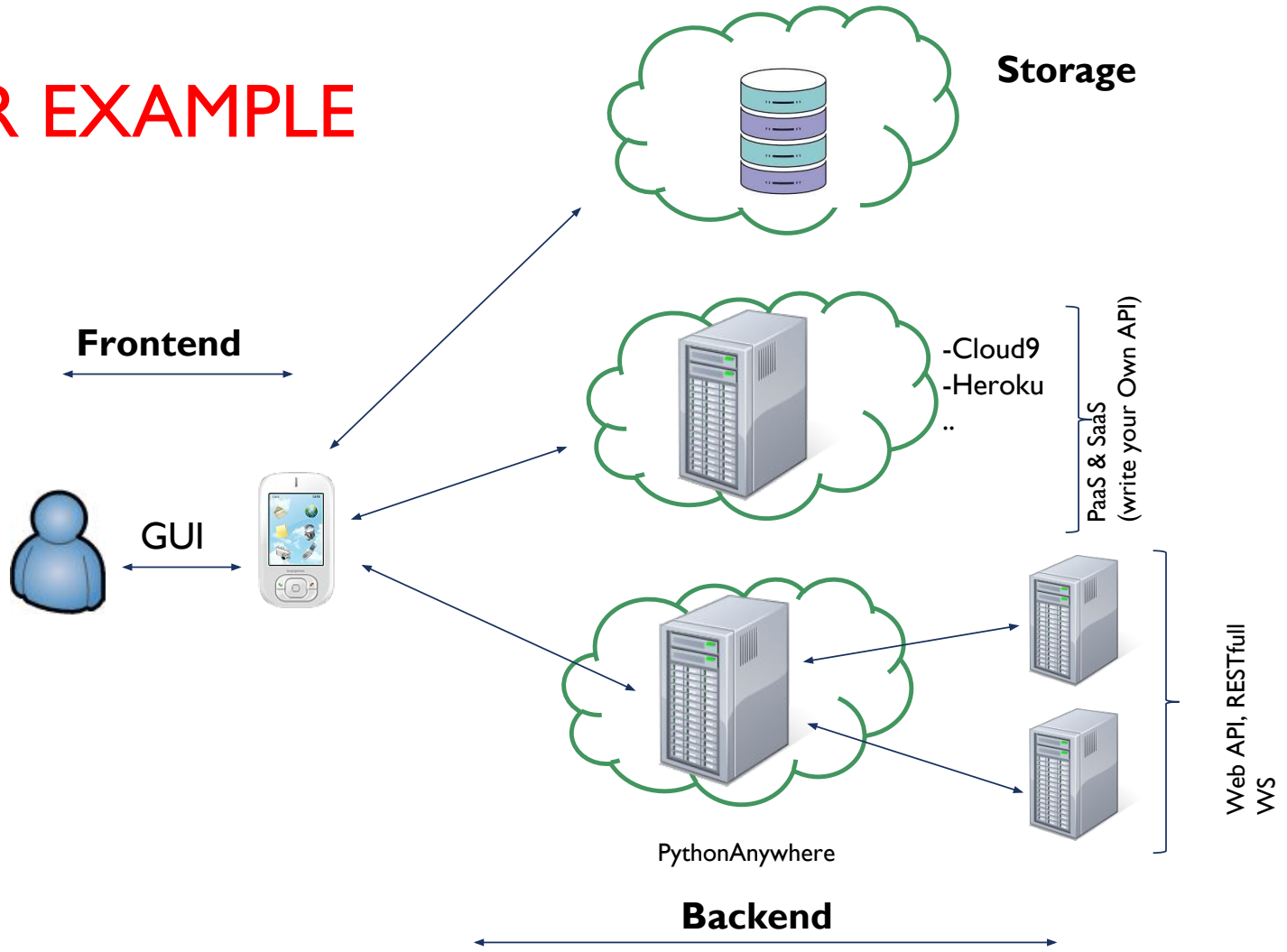


SPA=Sigle Page Application

SOME TERMINOLOGY

- **Cloud-native applications**: applications where the computational graph run enterally on cloud **and** are made of small loosely coupled components that can easily scale
- **Cloud Backend services:** components that run o cloud as support to applications running elsewhere
  - mobile Backend as a Service (mBaaS)
- **Lift & shift**: An application that usually runs on-premises and it is moved almost unchanged to run on a cloud

AN EXAMPLE


Frontend
GUI
-Firebase
MB-as-a-Service
(push notifaction, messages, etc
-Google Maps
May be Strongly Integrated in IDE
Wheatear Underground
-Dropbox
-Flicker
Web API, WS
(may provide SDK)
Authentication
Service
-Facebook (e.g, Oauth)
Backend

ANOTHER EXAMPLE