SE 339 - SOFTWARE ARCHITECTURE

MICROSERVICES

Lotfi ben Othmane

Iowa State University

Fall 2017

LAST LECTURE

What is a software architecture?

1. Partition the system into components considering requirements and constraints
2. Assigning responsibility to the components of the system
3. Addressing structural issues of composing elements of a system

GOAL

What is a microservice?

RESOURCES

- Microservices – Flexible software architecture

  by Eberhard Wolff

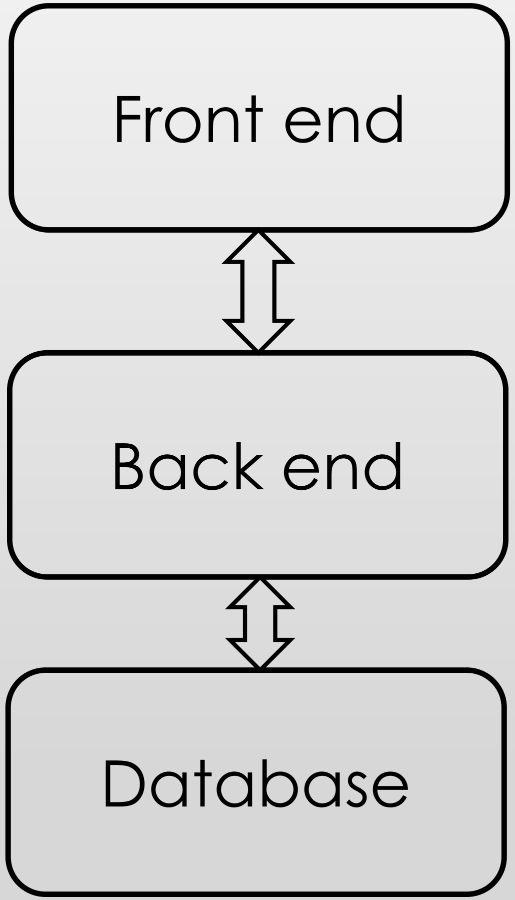
  http://microservices-book.com/content.html

- Microservices -- A definition of this new architectural term

  by James Lewis and Martin Fowler

  https://www.martinfowler.com/articles/microservices.html

MONOLITHIC APPLICATIONS



The code may be organized into modules

1. Dependency problem - Adding new features or even bug fixes require change to many components and redeployment of all the application
2. Interoperability problem - Organization is based on technology
   - Different teams work with different technologies
3. Scalability problem – Should applies to all the application
4. Resilience problem – Fail affects all the application
5. Consistency problem – Shared data needs to be consistent using transactions management protocol.

MONOLITHIC APPLICATIONS

There has been development of architecture styles and techniques to address the problems

- Web services for interoperability
- Transaction management with EJB
- Load balancing
- Etc.

Microservices was first discussed in a workshop of software architects, Venice, 2011

MICROSERVICES

An approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

These services are built around business capabilities and independently deployable by fully automated deployment machinery.

https://www.martinfowler.com/articles/microservices.html

UNIX PHILOSOPHY

- One program should fulfill only one task
- Programs should be able to work together
- A universal interface should be used—e.g., text stream

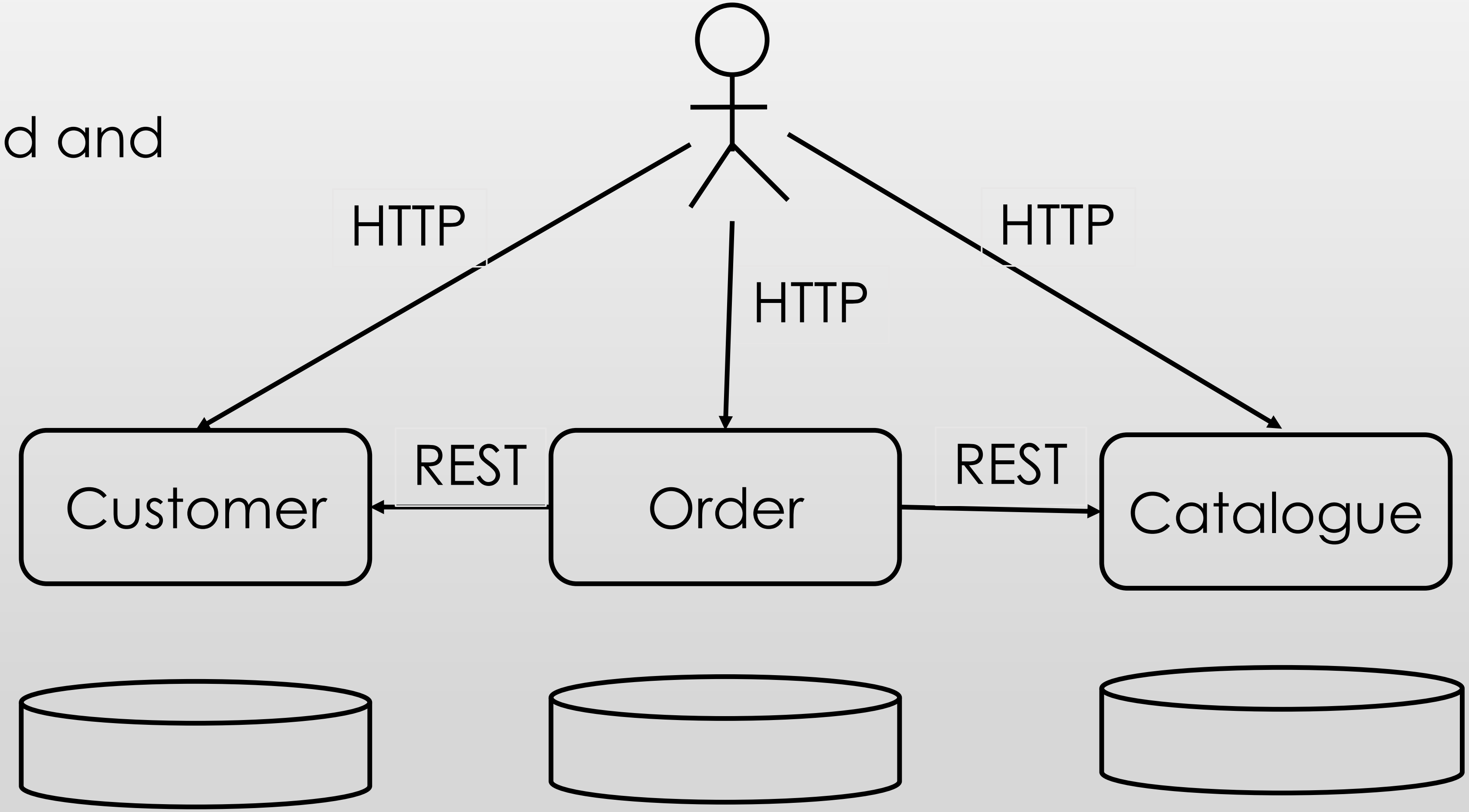
DEPENDENCY PROBLEM

Solution

- The boundaries is based on business context not technology
  - No separation between front end and back end
- Orchestration is implemented in microservices in not in infrastructure or communication
  - Threads and workflows are managed by microservices,
- Each microservice has a clear interface
- Each microservice manages its data
- Microservices run on independent processes
  - Could be deployed independently

DEPENDENCY PROBLEM

No separation between front end and back end



Chapter 13 of book microservices

DEPENDENCY PROBLEM

- One of the main challenges in architecture is to identify cut-points
  - Identify independent components

- Principle 1: Split is based on business capabilities boundaries
- Principle 2: Future changes should require updates to one microservice—minimize propagation

CRITERIA FOR CREATING NEW MICRO SERVICES

- Introduction of different data models
- Mixing of synchronous and asynchronous communication
- Incorporating additional services
- Different load scenarios for different aspects of the service

INTEROPERABILITY PROBLEM

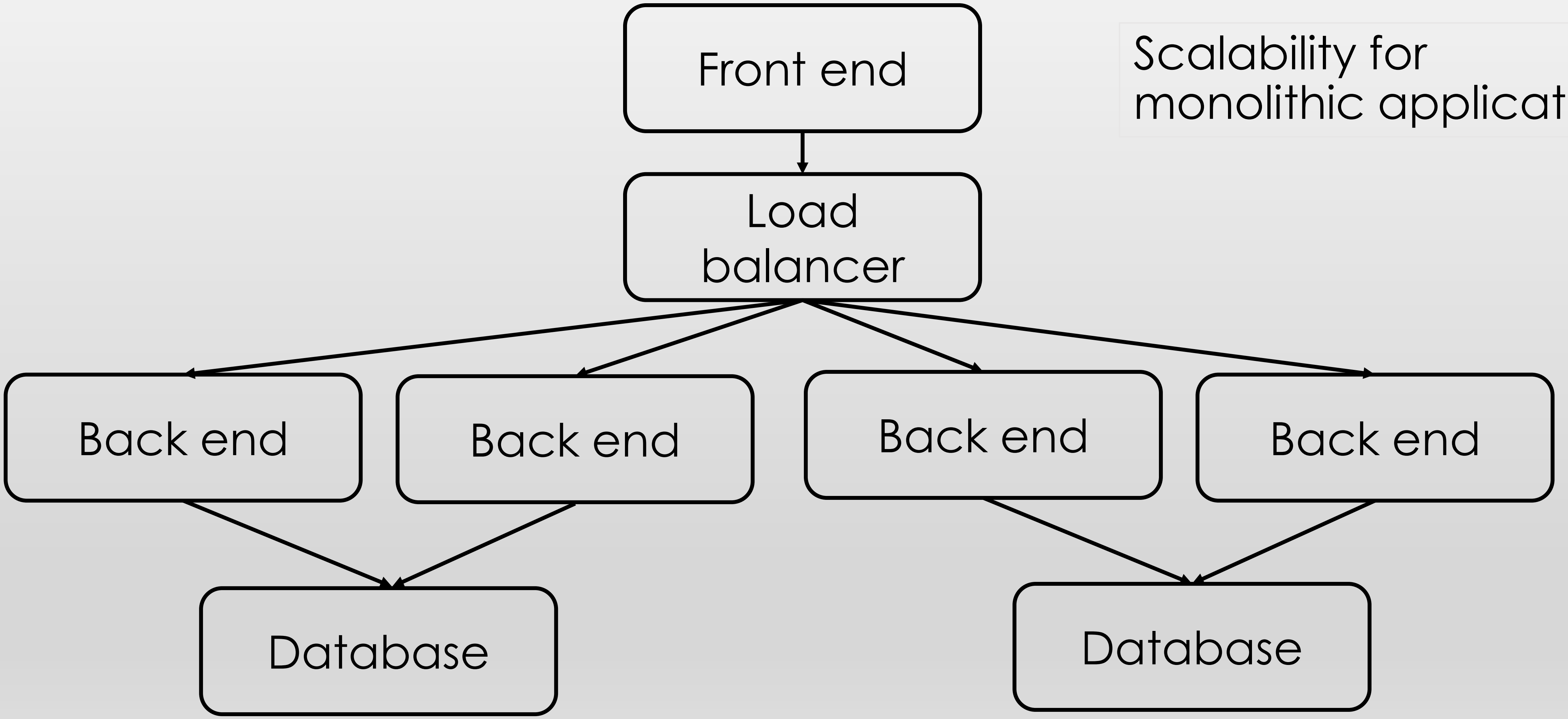
Web service solution

- Web services addresses this by allowing communication between web services using SOAP or Jason
- Web services support transactions for consistency – May be needed in some contexts
- Web services run on one process

Microservice solution

- Use lightweight communication mechanisms such as REST and RPC
- Compensation operation for inconsistency

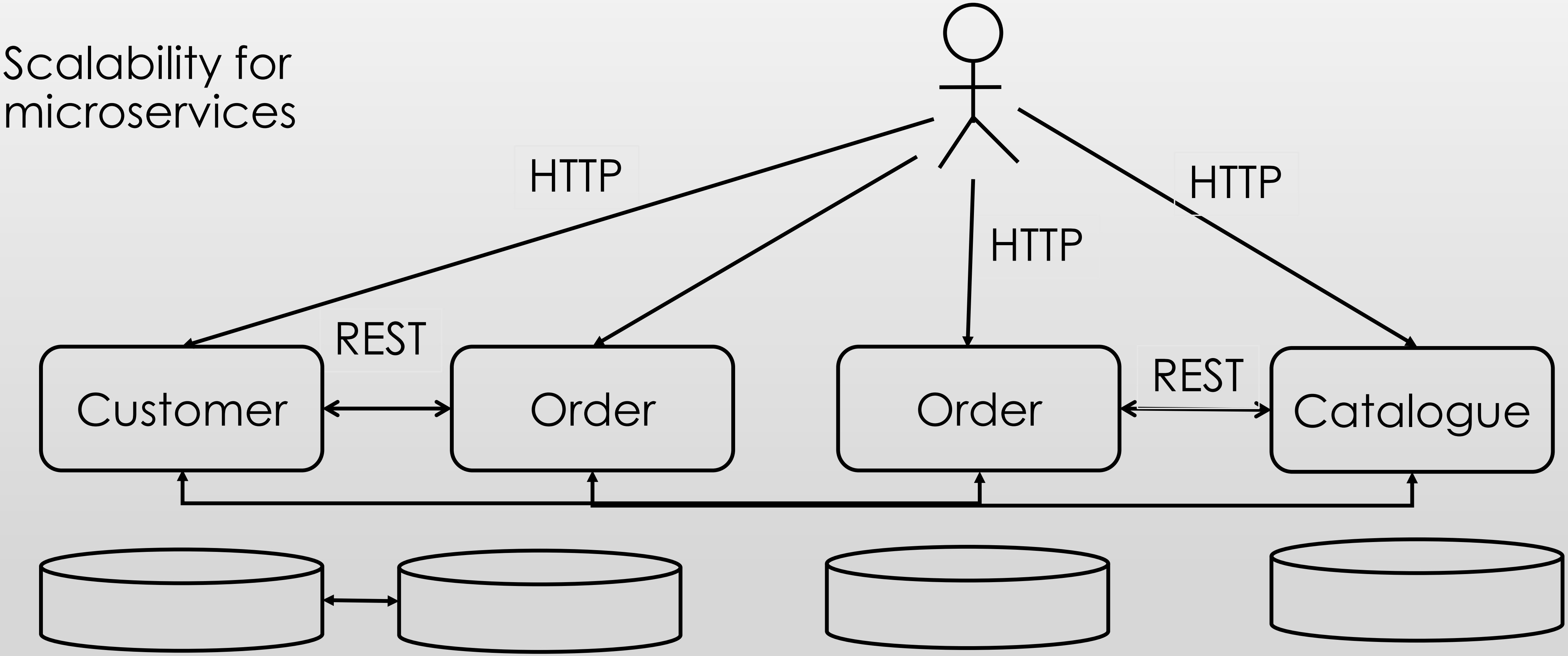
SCALABILITY PROBLEM



Scalability for monolithic applications

SCALABILITY PROBLEM

Scalability for microservices



RESILIENCE PROBLEM

Potential failure

- Service might have bugs – crash
- Service may become unavailable due to hardware or network problem
- Service may become slow to respond

➔Plan for eventual failure

- How should/must the microservice behave in the case of failure of each of the dependencies?
- Use of circuit breaker to handle failure: monitor microservices and trigger correction in case of failure

CONSISTENCY PROBLEM

Microservices related-characteristics

- Asynchronous communication
- No shared data – or minimum shared
- No management of service states for consistency

➔ Implement logic to detect inconsistency and to trigger corrective operations

- Do not use central system for consistency

CHARACTERISTICS

- Organized around business capabilities
- Products not Projects – developers support the product
- Smart endpoints and dumb pipes – request-logic-response
- Decentralized governance – technology choices
- Decentralized data Management
- Infrastructure automation – continuous development
- Design for failure – consider failure
- Evolutionary design – rewriting a component without affecting its collaborators

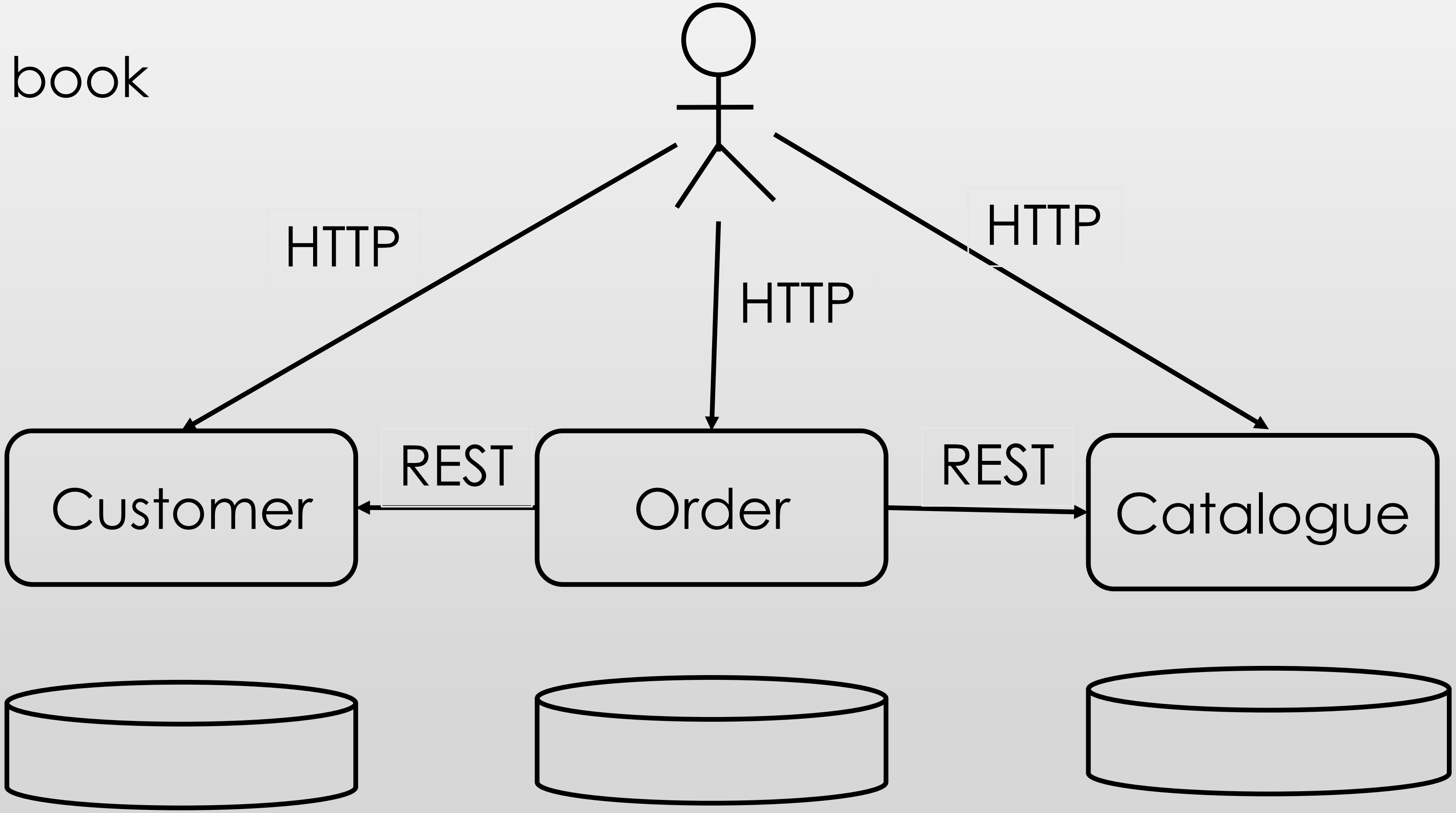
https://www.martinfowler.com/articles/microservices.html#footnote-etymology

BUT

Krutchen says: “The life of a software architect is a long (and sometimes painful) succession of sub-optimal decisions made partly in the dark.”

ASSIGNMENT

Chapter 13 of the book



https://github.com/ewolff/microservice

TECHNOLOGIES

- Use of Spring Framework – Java
- Use of HSQL database
- Spring Cloud Netflix for routing, discovery, resilience, etc.
- Deployment using Docker

MICROSERVICES EXAMPLE

Download the example from:

https://github.com/ewolff/microservice

We are working on an assignment based on the example.

Thank you.

Next class is on UML