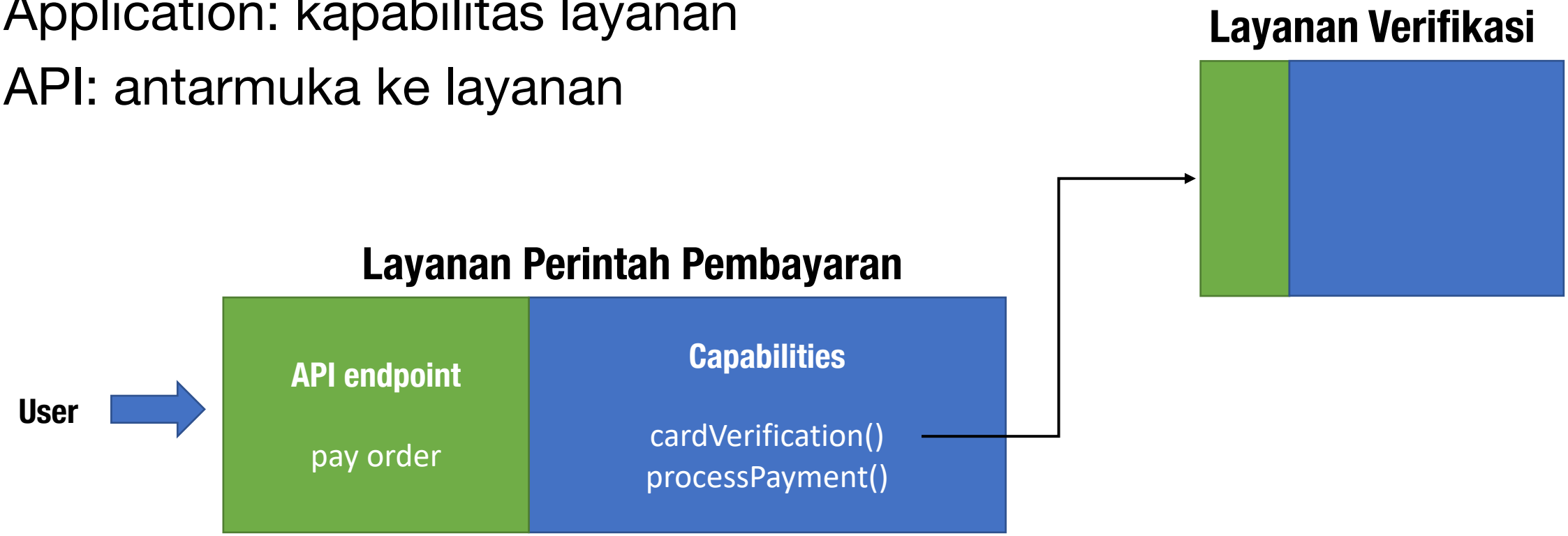

Application Programming Interface
(API)

Microservice Components

- Application: kapabilitas layanan
- API: antarmuka ke layanan



The Amazon API Mandate

- Tahun 2002, Jeff Bezos mengeluarkan sebuah mandat untuk internal perusahaan (Amazon) yang isinya:
    1. All teams will henceforth expose their data and functionality through service interfaces.
    2. Teams must communicate with each other through these interfaces.
    3. There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
    4. It doesn't matter what technology they use. HTTP, Corba, Pubsub, custom protocols — doesn't matter.
    5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
    6. Anyone who doesn't do this will be fired.
    7. Thank you; have a nice day!

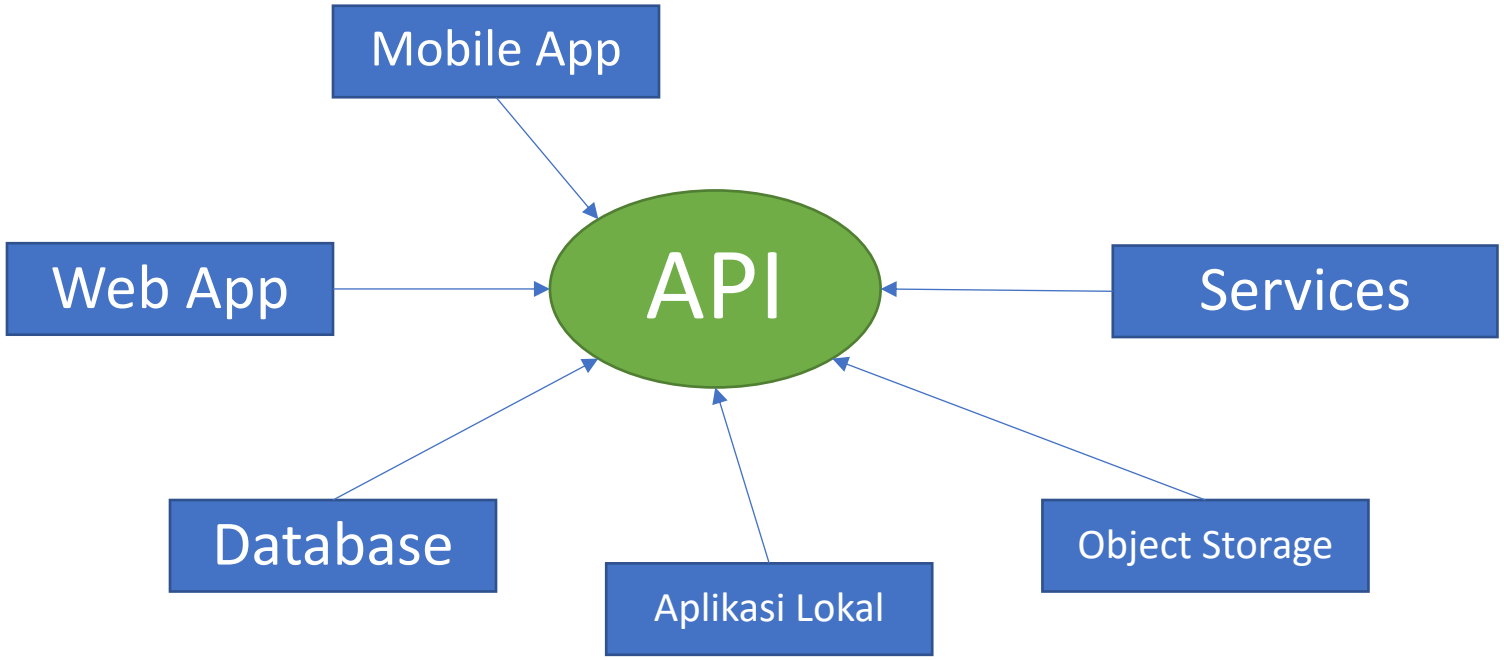
Application Programming Interface (API)

- API merepresentasikan *kontrak* antarlayanan. Layanan yang menggunakan API dari layanan lain memiliki pengetahuan tentang cara mengakses API tersebut dan respon yang akan diperoleh.
- API merupakan antarmuka ke aplikasi yang menjalankan layanan sehingga memegang peranan yang sangat penting dalam integrasi microservices.
- Contoh API: Web API, yang dapat diimplementasikan dengan framework tertentu, seperti SOAP, REST, GraphQL, gRCP, Twirp, dll. (umumnya bekerja di atas HTTP).

Application Programming Interface (API)

- API membuat aplikasi-aplikasi dapat berkomunikasi satu sama lain
- Sebuah aplikasi dapat menggunakan API pihak ketiga untuk menambah fungsi tertentu agar tidak perlu membangun fungsi tersebut dari nol.
- Contoh: aplikasi pemesanan kamar hotel menggunakan API untuk fitur peta dari penyedia layanan peta, menggunakan API dari bank untuk fungsi pembayaran, menggunakan API dari penyedia layanan autentikasi untuk keperluan login, dan sebagainya.

API dan Aplikasi



- API menentukan data dan layanan yang dapat diakses oleh aplikasi atau layanan lain.

Design Styles

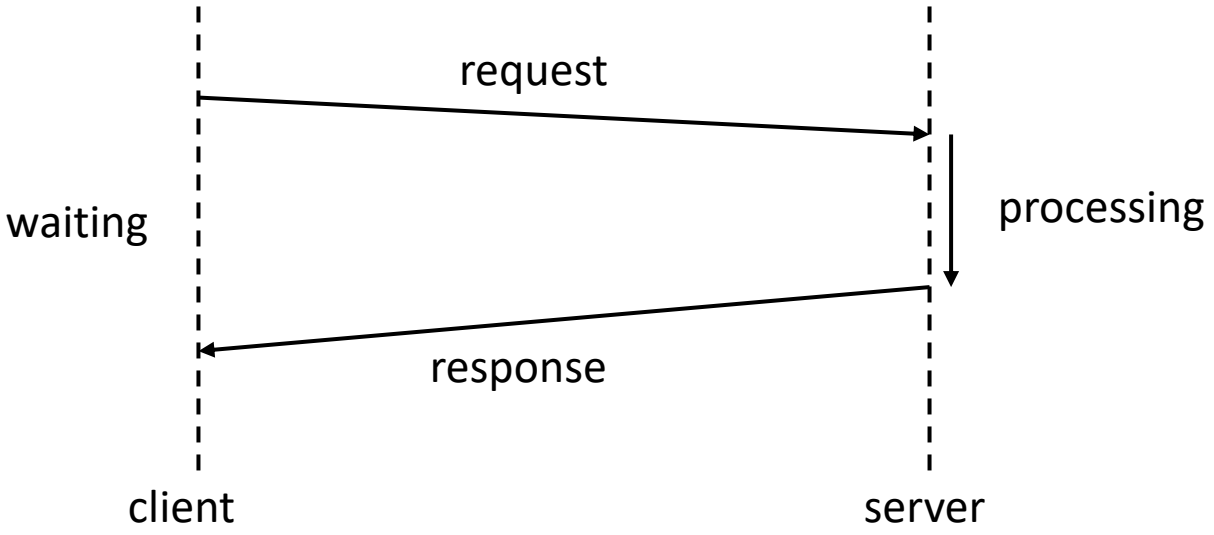
- **Synchronous**: data atau layanan dapat langsung diberikan saat diminta. Aplikasi yang mengirimkan API *request* harus menunggu data atau layanan tersebut sebelum melakukan langkah berikutnya (dapat menjadi *bottleneck*).
- **Asynchronous**: data atau layanan tidak langsung tersedia karena memerlukan waktu. Aplikasi yang mengirimkan *request* dapat melanjutkan proses yang lain tanpa perlu menunggu. Saat data atau layanan yang diminta telah tersedia, akan dikirimkan ke aplikasi tanpa perlu mengirimkan *request* ulang.

API Architectural Style

- Remote Procedure Call (RPC)
- Simple Object Access Protocol (SOAP)
- Representational State Transfer (REST)

RPC

- Client-server request/response model
- Contoh implementasi:
  - XML-RPC
  - JSON-RPC
  - NFS (Network File System)
  - Simple Object Access Protocol (SOAP)



SOAP

- Dapat beroperasi di atas HTTP, SMTP, TCP, UDP, or JMS
- SOAP message menggunakan format XML dengan 4 elemen:
  - Envelope
  - Header
  - Body
  - Fault

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Header/>
        <soap:Body>
                <soap:Fault>
                        <faultcode>soap:Server</faultcode>
                        <faultstring>Query request too large.</faultstring>
                </soap:Fault>
        </soap:Body>
</soap:Envelope>
```

REST

REST

- Disertasi berjudul *Architectural Styles and the Design of Network-based Software Architectures* (2000) oleh Roy Fielding
- an architectural style
- designed for loosely coupled applications
- has six constraints:
  - Client-Server
  - Stateless
  - Cache
  - Uniform Interface
  - Layered System
  - Code-On-Demand

Client-Server

- Client and server separation
  - Client:
    - requests for resource
    - Responsible for requesting, collecting, consuming, presenting resource
    - doesn’t care about process in the server
  - Server:
    - holds the resource
    - responsible for storing and processing
    - doesn’t care about frontend UI

Stateless

- Server does not maintain session
- Client manages state
- The request must include all information required
- No memory of past client activity

Cacheable

- Server can specify how response can be cached by the client and control the duration
- However, client may receive stale data

Uniform Interface

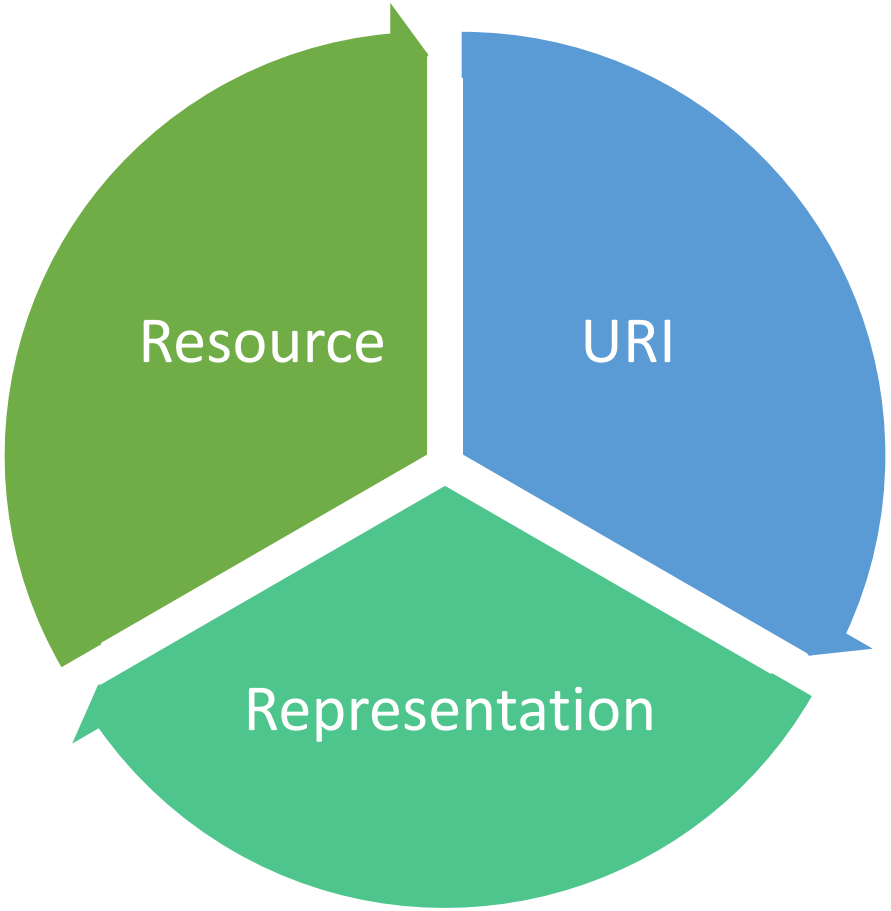
- A resource is identified in request by URI
- Manipulation through representation: modify/delete resource on server
- Self-descriptive: the message contains all information
- Hypermedia as engine of application state (HATEOAS): include links to discover other resources

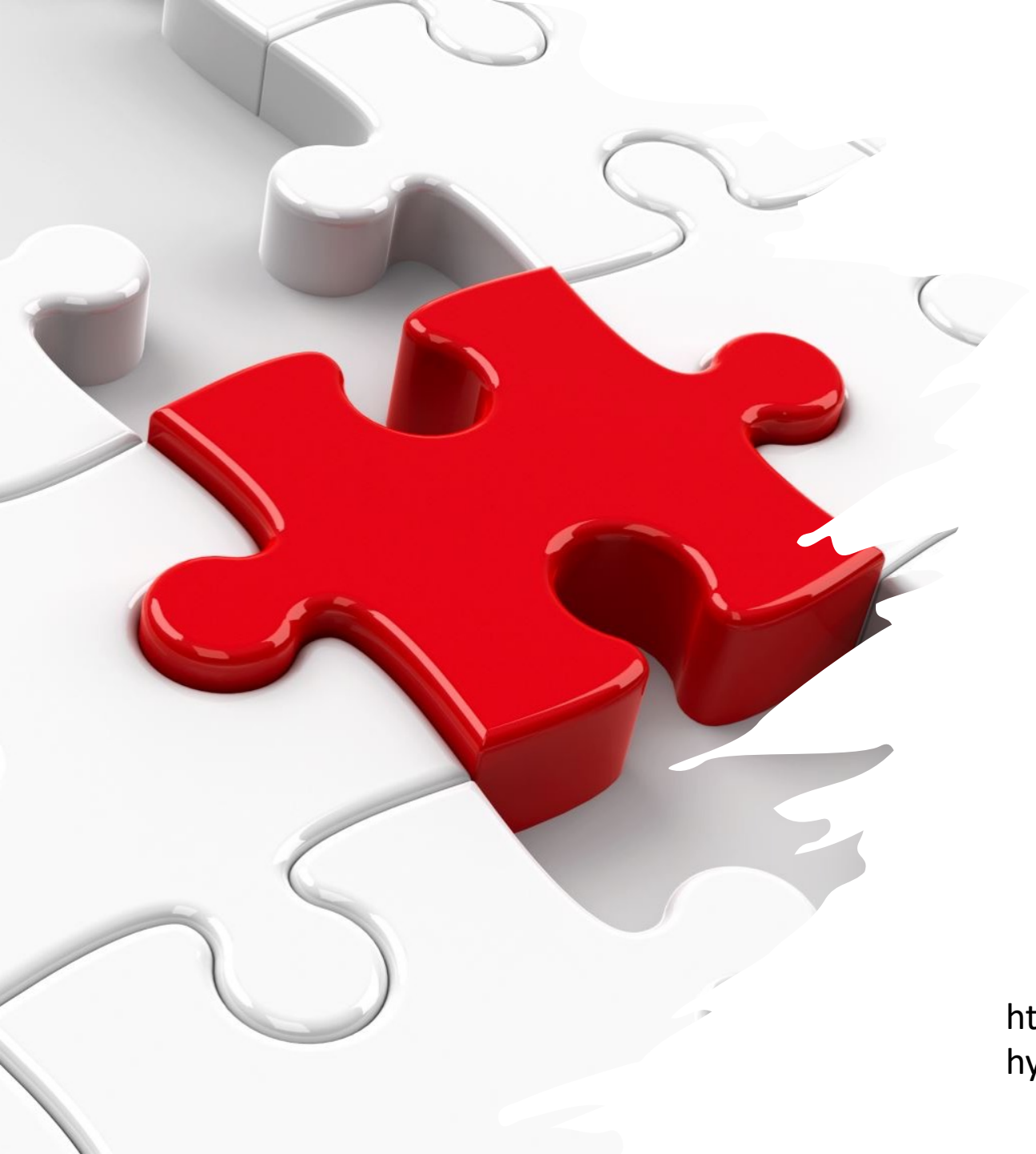
Layered System

- Layers of service between server and client (multi-tier)
- Example: load balancer, proxy, application server, and other intermediary servers

Code on demand

- Optional
- Server sends code to client to execute
- Enable application to be extensible


Resource
URI
Representation



REST is an architectural style, can be applied to a communication protocol.

However, many REST implementations are HTTP-based.

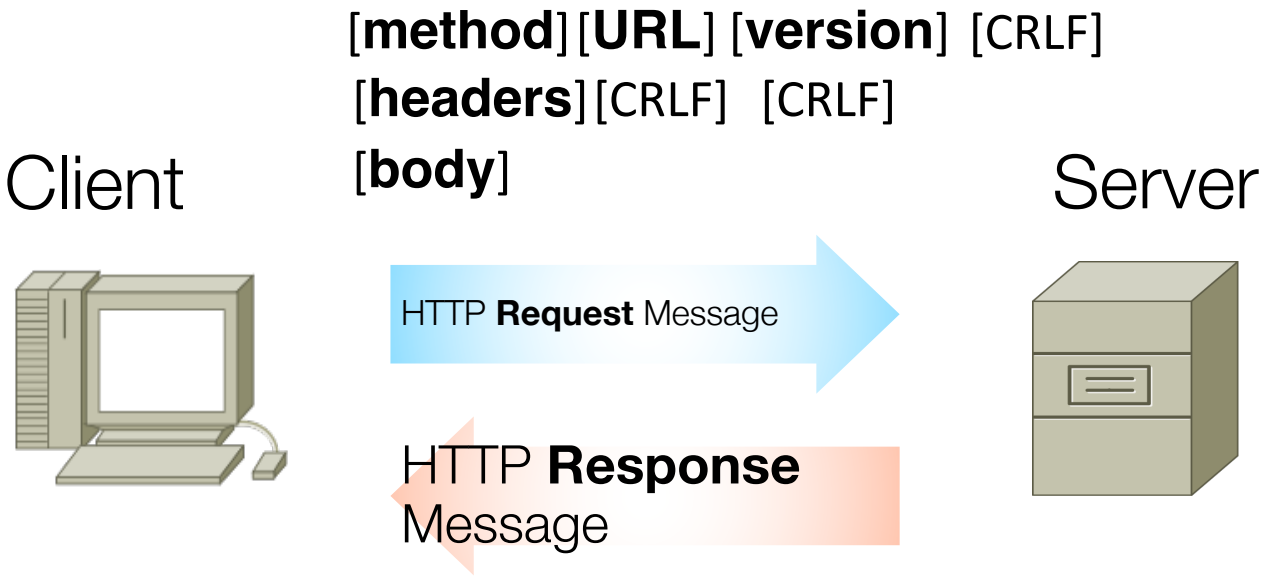
https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

HTTP Revisited

HTTP

- URIs represent resource
- HTTP verbs represent actions

HTTP Message



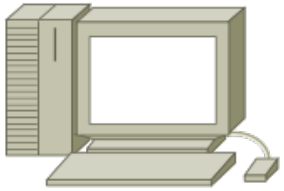
HTTP Example

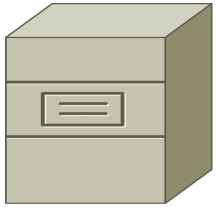
Request

Header

```
GET /index.html HTTP/1.1\r\n
Host: example.itb.ac.id\r\n
User-Agent: Mozilla/5.0\r\n
\r\n
```

Client





Server

Status

Header

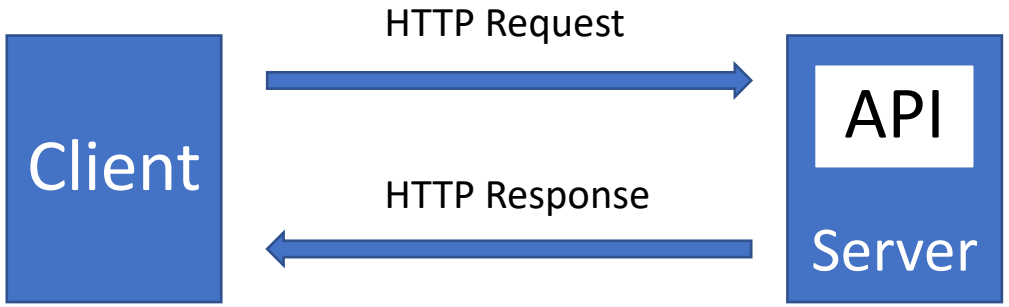
Data

```
HTTP/1.1 200 OK\r\n
Content-type: text/html\r\n
Content-length: 136\r\n
\r\n
<HTML><HEAD>
<TITLE>My Homepage</TITLE>
.....
```

REST API

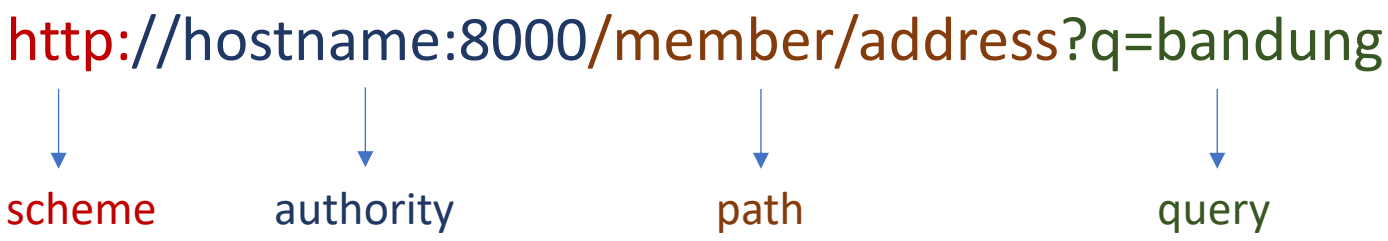
REST API

- Pertukaran pesan REST API menggunakan HTTP message:
  - HTTP requests/responses
  - HTTP verbs
  - HTTP status codes
  - HTTP headers/body



REST API Request

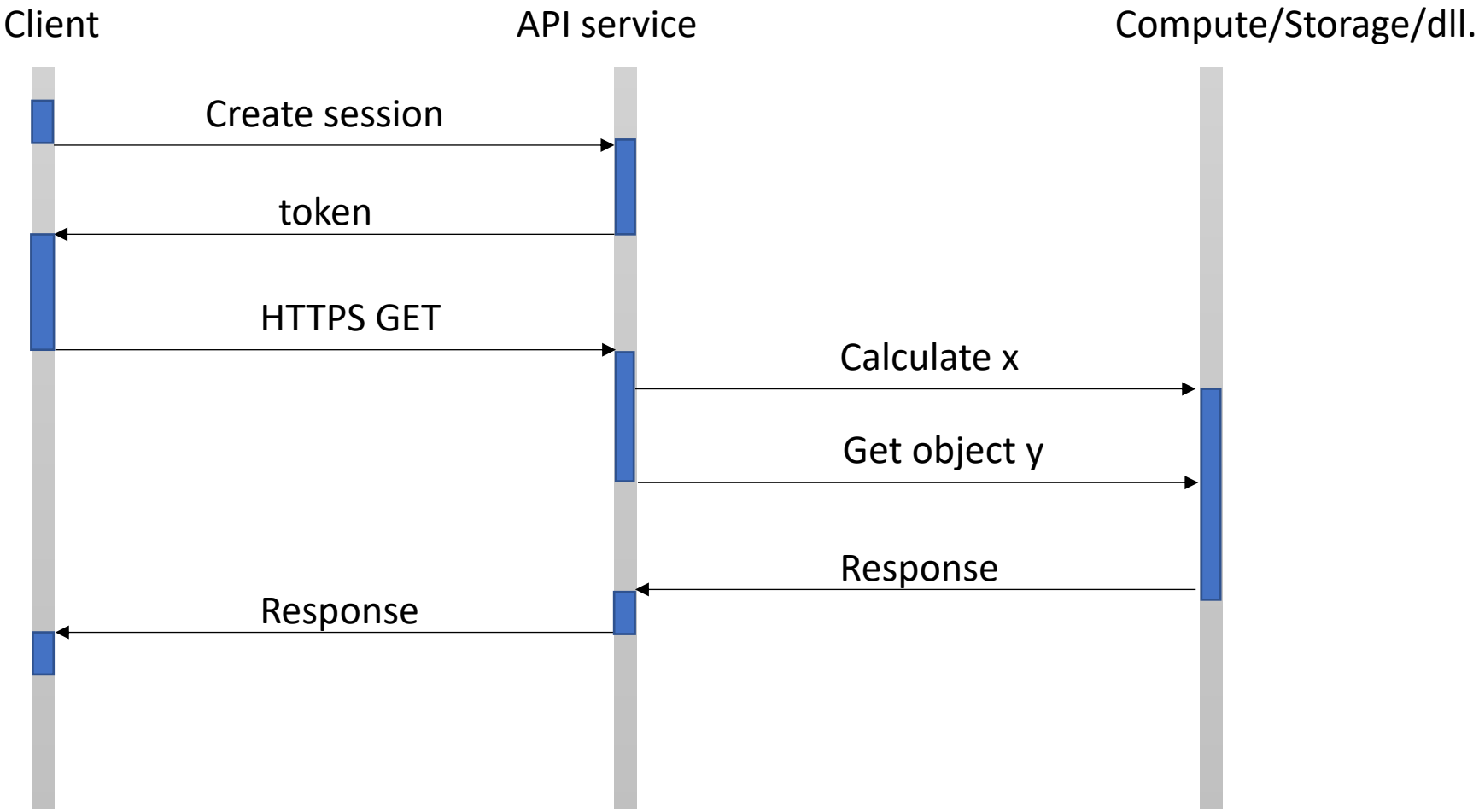
- Komponen REST API request
  - Uniform Resource Identifier (URI)
    - Scheme: http, https
    - Authority: host, port
    - Path: resource location
    - Query: optional
  - HTTP Method
  - Header: request header, entity header
  - Body: HTTP POST, PUT, PATCH



REST API Response

- Komponen
  - HTTP Status
    - 1xx - Informational
    - 2xx - Success
    - 3xx - Redirection
    - 4xx - Client Error
    - 5xx - Server Error
  - Header
    - Response header: Cache-Control
    - Entity header: Content-Type
  - Body

Sequence Diagram



Benefit of REST API

- Platform independent API
- Client written in a programming language can communicate with API written in other programming language

REST vs. SOA

- **Service**:
  - self-contained, independent software performing a task.
  - Service is not an API, but an architectural and deployment artifact used to implement enterprise solution.
  - Collection of methods
- REST related to **RESOURCE**, not service
  - Resource is accessed through an interface
  - Resource represents something

REST API Guidelines

Use nouns instead of verbs to access resource

- **Good**
  /users
  /user/18217999
- **Bad**
  /getAllUsers
  /getUser

Use 2 base URLs per resource

- **Example**
  /users/9999
  /address/bandung
  /dairy/milk
- users, address, dairy: **collection**
  9999, bandung, milk: **element**

Use plural and concrete nouns

- **Plural** noun, e.g., GET */students*, is **more intuitive** than abstract noun, e.g., GET */student* (we would like to read all students data)
- Use **concrete** names, *e.g., music, students, and food*, instead of abstract names, *e.g., items and things*.
- Be **consistent**, avoid mixed types.

Use HTTP verbs for CRUD operations

- **CRUD**: **C**reate-**R**ead-**U**pdate-**D**elete
- HTTP verbs: Post-Get-Put-Delete
- Example of HTTP requests

```
POST /user/1234 HTTP/1.1
GET /users HTTP/1.0
```

Put state and attribute behind the HTTP question mark

- GET /students?location=bandung&sex=female

Use relevant status codes

- Status code between application and API:
  - It works! – 200 OK
  - Application error – 400 Bad Request
  - API error – 500 Internal Server Error
- Other relevant codes:
  - 201 – Created
  - 404 – Not Found
  - dll

Default format: JSON

- Universal and less verbose than XML
- Example:

```
{ “student” :
{ “NIM” : “18217999”,
  “Name” : “John Doe”
}
}
```


Authentication and Authorization

REST API Authentication and Authorization

- Authentication: the process or action of verifying the identity of a user or process.
- Authorization: permission to perform action on specific resources

REST API Authentication

- Basic authentication
  - Authorization: Basic <username>:<password>
  - Credentials are encoded, but not encrypted.
- Bearer/token authentication
  - Authorization: Bearer <bearer token>
  - OAuth, SSO
- API Key/Token
  - All API requests must use this token obtained from the server
  - Penggunaan:
    - dalam header: Authorization: <API Key>
    - dalam body: menggunakan key tertentu dengan format JSON
    - pada query string (hanya untuk public key)

REST API Authorization

- OAuth 2.0 Authorization Framework (RFC 6749)
  enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf

API Rate Limit

- Tujuan:
  - mencegah overload dan abuse
  - keamanan terhadap serangan
  - quality of service
- Informasi rate limit diberikan di dalam response
- Request yang kena batasan dapat diberikan respon HTTP dengan kode 429 atau 403


REST API Implementation using
Python

Frameworks and Server Installation

- Web Framework
  - `pip3 install starlette` # Starlette is an ASGI framework/toolkit
- Server
  - `pip3 install uvicorn` # Uvicorn is an ASGI server
- API development framework
  - `pip3 install fastapi`

Asynchronous Server Gateway Interface (ASGI)

- An interface between web server, framework, and applications
- Send and receive message as Python's dictionary with predefined format.
- Implementations:

  Server: Daphne, Hypercorn, **Uvicorn**

  Application frameworks: Django/Channels, FastAPI, Quart, **Starlette**
- Specification: https://asgi.readthedocs.io/en/latest/specs/main.html

FastAPI

- A framework for creating API using Python 3.6+
- Based on **OpenAPI** specification for API creation
- Automatic documentation using **SwaggerUI**
- Based on **Pydantic**, a library for data validation

Virtual Environment Setup

- `mkdir <directory>`
- `cd <directory>`
- `pipenv install    # install virtual env in this directory`
- `pipenv shell   # activate virtual env`
- `exit           # deactivate virtual env`

A Simple Example of using FastAPI

- Python script `main.py`:

```
from fastapi import FastAPI  # import FastAPI class.
app = FastAPI()              # create FastAPI instance.
@app.get("/")                # define path operation decorator.
async def root():            # Python function. The use of async is optional.
      return {"Hello": "World"} # return the content (dict, list, int, str,  pydantic model, etc.).
```

- Run the server:
  `uvicorn main:app --reload`

Note:
1. Use `--reload` to automatically reload server when code changes.
2. path = endpoint, operation = method/verb, @something = decorator
3. @app.get(“/”) decorator indicates the following function will be called when it receives request to URL “/” using GET method.

Use Case: Restaurant Menu

Soup

G.S.T. Included

Long Soup (Egg Noodle) $5.20
Short Soup (Won Ton) $5.20
Combination Soup $5.20
(mixed vegetables, chicken & pork)
Chicken & Sweet Corn Soup $5.20
Prawn & Sweet Corn Soup $5.20
Crab Meat & Sweet Corn Soup $5.20

Appetizers & Entree

Vegetarian Mini Spring Rolls (4) $4.80
Mini Spring Rolls (4) $4.80
Home Made Dim Sims
(Steamed Or Fried) (4) $4.80
Tasty Deep Fried Chicken Wings $4.80
Sesame Prawn Toast $5.20
King Prawn Cutlets (3) $6.60
Mixes Entree $5.20
(Spring Roll, Fried Dim Sim & King Prawn Cutlet)
Prawn Cocktail $6.90
Garlic King Prawns $10.00
Prawn Chips $2.80

Main Meal

First select a sauce option then select a meat option

Satay Sauce
Szechuan Sauce
Plum Sauce
(onion , celery, carrot, capsicum, baby corn and mushrooms)
Black Bean Sauce
Barbeque Sauce
(broccoli, onion, celery, carrot, capsicum, baby corn and mushrooms)

Main Meal (Continued)

Oyster Sauce
Garlic Sauce
(broccoli, onion, celery ,carrot, baby corn and mushrooms)
Chilli Sauce
(broccoli, onion, celery, carrot, baby corn, mushrooms ,capsicum, and chillies)
Mongolian Sauce
Peking Sauce
(onion, celery, carrot, capsicum and shallots)
Ginger and Shallot Sauce
(ginger, shallots, onion, celery, carrot, zucchini baby corn and mushrooms)
Sweet and Sour Sauce
(onion, celery, carrot, zucchini, capsicum, baby corn and mushrooms)

Meat

G.S.T. Included

Chicken $13.90
Beef $13.90
Pork Fillet $13.90
(not available for sweet & sour sauce | see house special)
Lamb $17.00
Combination $16.00
Squid $16.00
King Prawns $20.00
Mixed Seafood (king prawns & squid) $20.00
Duck (half boneless duck) $20.50
Mini Prawns $14.50

Extras

	Packed Separately	On the Meal
Cashew Nuts OR Almonds	$2.00	$1.50
Crispy Chow Mein Noodles (per pack)	$2.00	$1.50
All Sauces packed separately	$0.50	N/A

House Specially

G.S.T. Included

Honey Sauce with king Prawns (deep fried) $20.00
Honey Sauce with Chicken (deep fried) $13.90
Lemon Sauce with King Prawns (deep fried) $20.00
Lemon Sauce with Chicken (deep fried) $13.90
Lemon Sauce with Duck (deep fried) $20.50
Steamed Duck with Crab Meat Sauce & Vegetables $20.50
Chilli Hot Duck with Vegetable (deep fried) $20.50
Sweet & Sour Pork deep fried with Vegetables $13.90

Omelettes

(egg, onion, celery, peas and bean sprouts)

Mini Prawn Omelette $14.50
Ham Omelette $14.50
Combination Omelette $14.50
King Prawn Omelette $18.00
Vegetable Omelette $14.50

Chow Mein

Broccoli, onion , Celery, Carrot, Baby Corn and Mushrooms served with Crispy Noodles
Price of Meat (Plus) $1.50

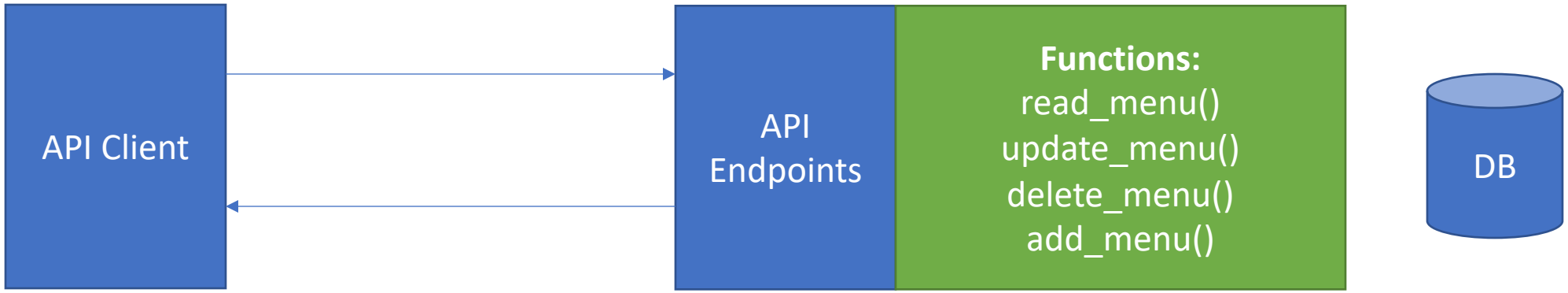
Asian Dishes

Singapore Noodles $15.00
Deep Fried Tofu with Spicy Sauce $14.00
Chinese Vegetables in Oyster Sauce OR Garlic Sauce
- Bok Choy $13.50
- Bok Choy and Broccoli $13.50

Thai Sauce with Deep Fried Chicken $13.90
Thai Sauce with Deep Fried Tofu $14.00
with Thai sauce or spicy salt
Green Curry OR Yellow Curry with Vegetables and
- Chicken $13.90
- Beef $13.90
- King Prawns $20.00

API Endpoint

- Endpoint/Path: `/menu/{item_id}`
- Method/Operations:
  - Retrieve specific menu item (GET)
  - Update menu item (UPDATE)
  - Delete item from the menu (DELETE)
  - Add new menu item (POST)



Example: Read from JSON data

Example: menu.json file

```
{
        "menu": [
        {
                "id":1,
                "name":"nasi”
        },
        {
                "id":2,
                "name":"sayur”
        }
        ]
}
```

Example: Read JSON data

```
import json
from fastapi import FastAPI
with open("menu.json", "r") as read_file:
      data = json.load(read_file)
app = FastAPI()
@app.get('/menu/{item_id}')
async def read_menu(item_id: int):
      for menu_item in data['menu']:
            if menu_item['id'] == item_id:
                  return menu_item
      raise HTTPException(
            status_code=404, detail=f'Item not found'
      )
```

Example: using Pydantic

```
from fastapi import FastAPI
from pydantic import BaseModel

class Item(BaseModel):                  # the data model
       id: int
       name: str

app = FastAPI()

@app.post('/menu')
async def add_menu(menu_item: Item):
       return menu_item
```