Back-End Basic

bangkit

Ground Rules

Observe the following rules to ensure a supportive, inclusive, and engaging classes

Give full attention in class

Mute your microphone when you're not talking

Keep your camera on

Turn on the CC Feature on Meet

Use raise hand or chat to ask questions

Make this room a safe place to learn and share

bangkit

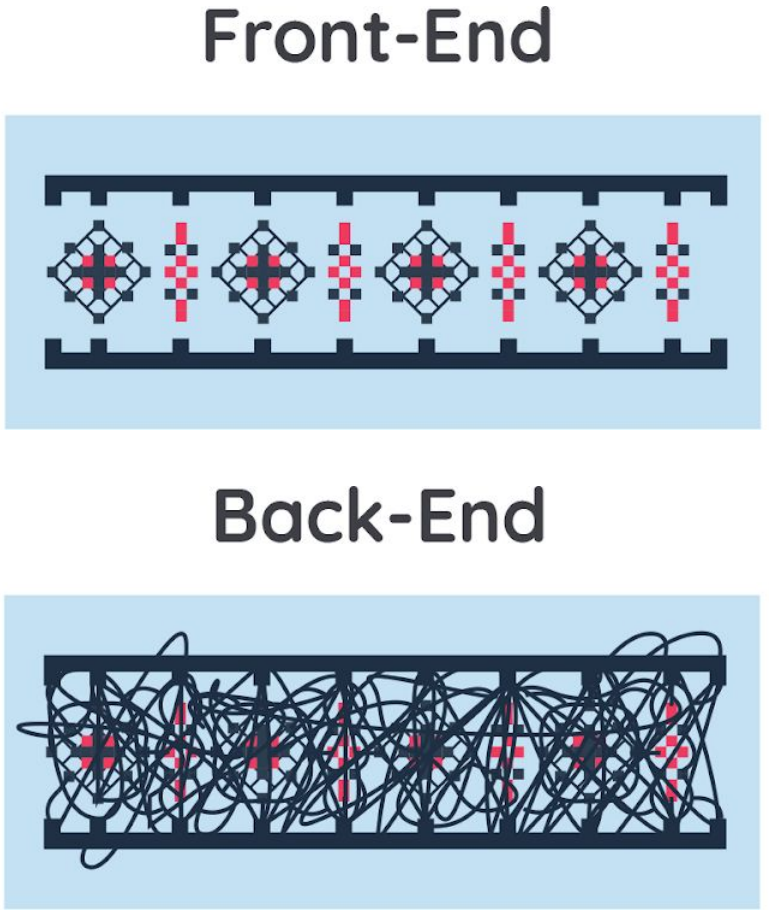
Prerequisite

- Familiar with Basic Front-End Web Development.
- JavaScript Programming:
  - JavaScript Basic
  - ES6 Syntax
  - Promise

Introduction to Back-End

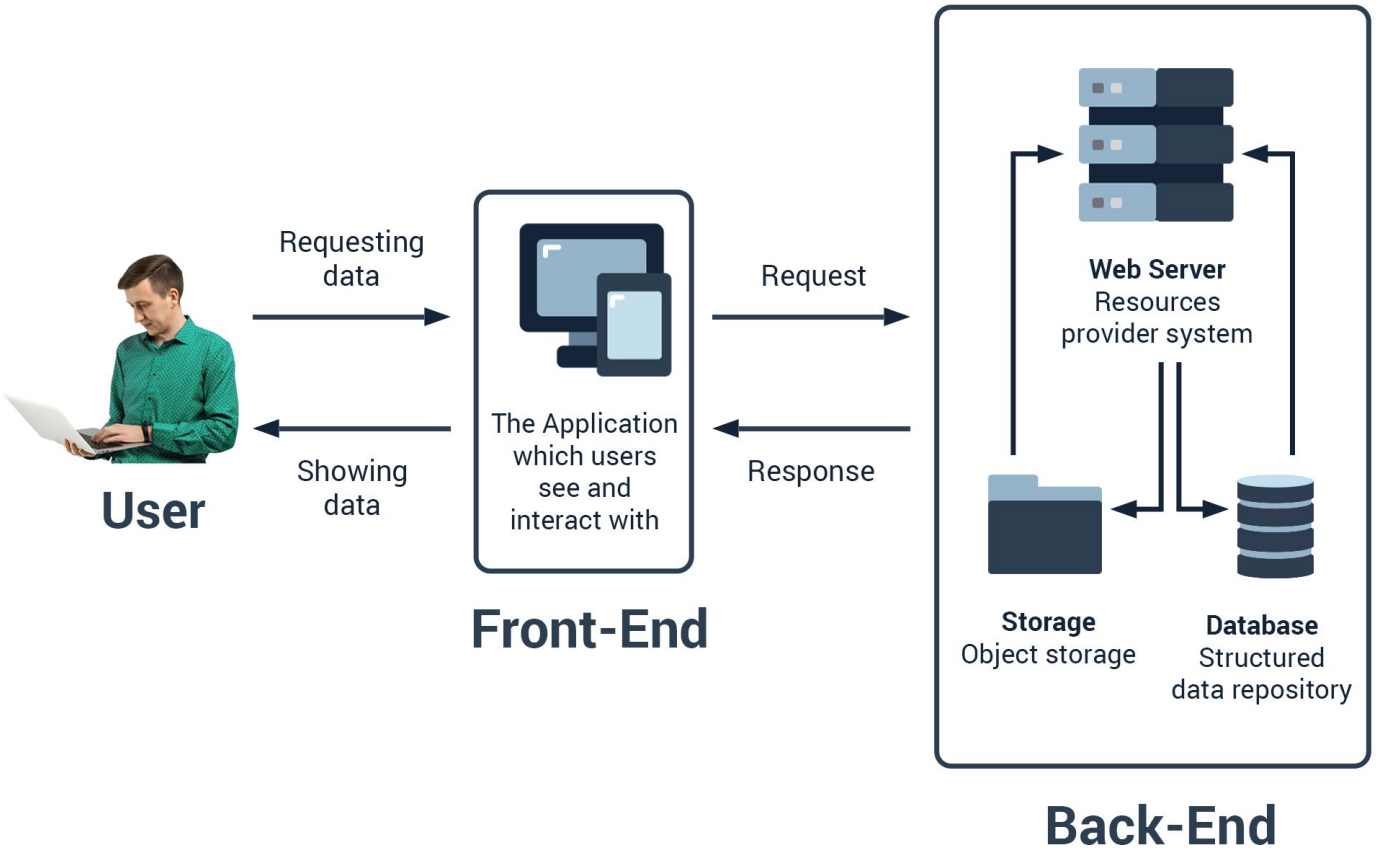
What is Back-End?

- **Front-End:** Part of the application that the user can see and use directly.
- **Back-End:** Part of the application that support to the needs of the front-end application.



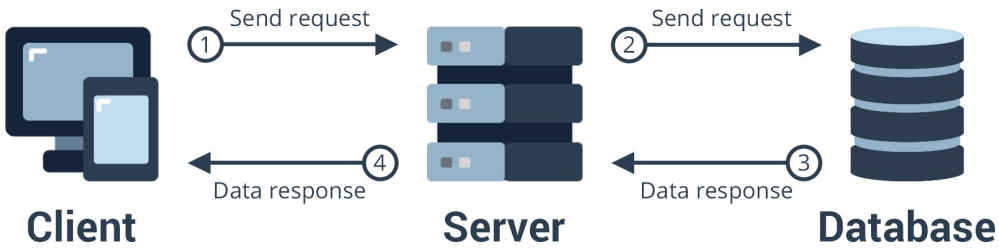
bangkit

Scope of Back-End and Front-End



Client-Server Communication

- One of the protocols for interacting with the web server is **HTTP/HTTPS**
- HTTP/HTTPS using **request-response** pattern
- To get something from server (response), we need **to make a request first**.



Introduction to REST

Common Terms

- REST : **RE**presentational **S**tate **T**ransfer.
- API : **A**pplication **P**rogramming **I**nterface.
- RESTful API : An API that implements a REST architecture.

Request dan Response Format

- REST APIs often use JSON as the request and response format.
- JSON has a structure like JavaScript Object.
- The difference is, Key is written using double quotes (“”).
- Can contain primitive value, object, or array.


```
{
  "message": "Berikut daftar kopi yang tersedia",
  "coffees": [
    {
      "id": 1,
      "name": "Kopi Tubruk",
      "price": 12000
    },
    {
      "id": 2,
      "name": "Kopi Tarik",
      "price": 15000
    },
    {
      "id": 3,
      "name": "Kopi Jawa",
      "price": 18000
    }
  ]
}
```

HTTP Verbs/Methods and Response Code

- **HTTP Verbs/Methods**
  - GET
  - POST
  - PUT
  - DELETE
- **HTTP Response Code**
  - 200 (OK)
  - 201 (Created)
  - 400 (Bad Request)
  - 401 (Unauthorized)
  - 403 (Forbidden)
  - 404 (Not Found)
  - 500 (Internal Server Error)

URL Design

- Use nouns instead of verbs in path endpoints.
  - ~~/getArticles~~ -> GET /articles
  - ~~/postComment~~ -> POST /comments
- Use the plural in endpoints for resource collections.
  - ~~/article~~ -> /articles
  - ~~/article/:id~~ -> /articles/:id
- Use chained endpoints for resources that have hierarchies/relationships.
  - ~~/comments~~ -> /articles/:id/comments

Node.js Basic

bangkit

What is Node.js?

- Created in 2009 by Ryan Dahl.
- JavaScript Runtime to run JS outside the browser environment.
- Allows us to become a Full-Stack Developer just by learning one programming language. JavaScript!

Node.js Basic for the Back-End

- Process Object
- Modularization
- Events
- Filesystem
- Stream
- Package Manager (NPM)
- HTTP

Create HTTP Server in Native Ways

```
// load http module
const http = require('http');

// create a HTTP server
const server = http.createServer(requestListener);

// run the HTTP server
server.listen(3000, 'localhost', () => {
   console.log('Server running on http://localhost:3000');
});
```

Routing Request in Native Ways

```
const requestListener = (request, response) => {
    response.setHeader('Content-Type', 'application/json'); // set the content-type manually
    response.statusCode = 200; // set the status code manually

    const { method, url } = request;

    // Routing response based on url & HTTP verb/method manually using if-else
    if (url === '/' && method === 'GET') {
        response.end(JSON.stringify({ message: 'You are doing GET' }));
    } else if (url === '/' && method === 'POST') {
        response.end(JSON.stringify({ message: 'You are doing POST' }));
    } else {
        response.end(JSON.stringify({ message: 'Unknown action'}))
    }
};
```

Why Framework?

- Easier to use
- Many built-in feature
- More focus on business logic
- Proven code

Create HTTP Server in Hapi Framework

```
// load Hapi module
const Hapi = require('@hapi/hapi');

(async () => {
 // create HTTP server
 const server = Hapi.server({ host: 'localhost', port: 3000 });

 // run the HTTP server
 await server.start();
 console.log('server start at ', server.info.uri);
})()
```

Routing Request in Hapi Framework

```
server.route([
  {
    method: 'GET',
    path: '/',
    handler: () => ({ message: 'You are doing GET' }),
  },
  {
    method: 'POST',
    path: '/',
    handler: () => ({ message: 'You are doing POST' }),
  },
]);
```

Deploy Node.js Apps on Google Compute Engine

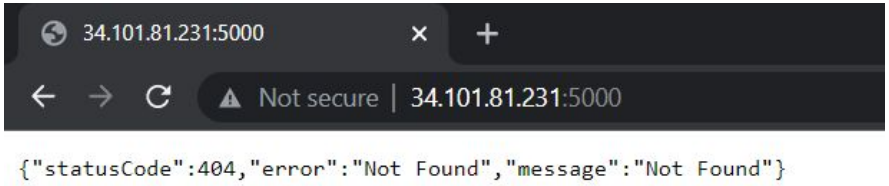
bangkit

Google Compute Engine

- Virtual servers residing in the Google Cloud infrastructure
- More economical than on-premise servers
- New Customer gets $300 Free Credits (verify using Visa/Mastercard online transaction-enabled debit/credit card)

Deploy Node.js App to Google Compute Engine

- Create a GCP account and get $300 free credits
- Create GCE Server and Remote via SSH
- Install Node.js inside GCE Server
- Upload the web service project to GitHub and clone repo in GCE Server
- Change host to 0.0.0.0
- Run the Node.js Application



bangkit

Process Manager

- The web service must run continuously-in-background
- pm2 : Tools for Node.js process manager
- Ensure the process will continue to work.
  If the process stops, pm2 will automatically restart the process.

```
[PM2] Spawning PM2 daemon with pm2_home=/home/fikri/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/fikri/.nvm/versions/node/v14.17.6/bin/npm in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	↺	status	cpu	memory
0	notes-api	fork	0	online	0%	32.6mb

```
fikri@web-server:~$
```

Testing HTTP Server with Postman

Postman

- API Caller with complete feature
- Automation Testing
- Easy Graphical Interface
- Available on Multiplatform
- Free

Why Automation Test with Postman?

- Effective
- Integrate with CI/CD workflow
- Friendly API

```
// check status code
pm.expect(response).to.have.status(200)

// check headers property value
const contentType = response.headers.get('Content-Type')
pm.expect(contentType).to.includes('application/json')

// check body property value as JSON
const responseJson = response.json()
const { status, message } = responseJson
pm.expect(status).to.eql('success')
pm.expect(message).to.eql('request is success')
```

Sharing Session

Demo Link

https://github.com/dicodingacademy/ilt-cloud-2-bangkit-demo

Quiz

Discussion

Thank You

bangkit