

**LAPORAN PRAKTIKUM  
KONSTRUKSI PERANGKAT LUNAK**

**MODUL VIII**

**Runtime Configuration dan Internationalization**



**Disusun Oleh:**

**Dewi Atika Muthi / 2211104042**

**SE-06-02**

**Asisten Praktikum:**

**Muhamad Taufiq Hidayat**

**Dosen Pengampu:**

**Riyan Dwi Yulian Prakoso, S.Kom., M.Kom.**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2025**

# BAB I

## PENDAHULUAN

### A. DASAR TEORI

**Runtime configuration** adalah teknik untuk menetapkan nilai variabel dan pengaturan program selama eksekusi aplikasi. Ini dilakukan tanpa mengubah kode sumber secara langsung, melainkan dengan menggunakan file konfigurasi eksternal atau variabel lingkungan. Konsep ini penting saat pengembang:

1. Tidak memiliki akses ke source code.
2. Menginginkan fleksibilitas dalam mengatur performa aplikasi saat runtime.

Dalam konteks .NET Core (modul referensi), konfigurasi dapat dilakukan melalui:

1. File `runtimeconfig.json`, tempat menyimpan properti konfigurasi dalam format JSON.
2. MSBuild Properties, yaitu konfigurasi pada file proyek seperti `.csproj`.
3. Environment Variables, menggunakan variabel seperti `COMPlus_GCRetainVM`.

Di ekosistem Node.js, konfigurasi runtime sering dilakukan dengan file JSON seperti `config.json` atau melalui `process.env` untuk variabel lingkungan.

Ilustrasi sederhana file konfigurasi JSON:

```
{
  "configProperties": {
    "maxConnections": 100,
    "enableLogging": true
  }
}
```

**Internationalization** (i18n) adalah proses mendesain aplikasi agar dapat disesuaikan dengan berbagai bahasa dan wilayah tanpa memerlukan modifikasi kode. Internationalization menyediakan pondasi teknis yang memungkinkan localization, yaitu proses penerjemahan konten dan adaptasi aplikasi untuk pasar tertentu.

Konsep i18n dalam pengembangan perangkat lunak meliputi:

1. Pemisahan teks dari logika program (misalnya, melalui file `.json` atau `.po`).
2. Penggunaan library seperti `i18next`, `gettext`, atau `intl` dalam JavaScript.
3. Memilih resource file berdasarkan bahasa lokal pengguna saat runtime.

Contoh struktur `locales/en/translation.json` dalam aplikasi Node.js:

```
{
  "welcomeMessage": "Welcome to our site!",
  "logout": "Log out"
}
```

Ilustrasi Proses i18n:

```
App Logic -----> [Language Resource] -----> Display in user's language
```

## **B. MAKSUD DAN TUJUAN**

Tujuan dari praktikum ini adalah:

1. Mahasiswa mampu membuat aplikasi berbasis console.
2. Memahami konsep dasar dan penerapan konfigurasi aplikasi pada saat runtime menggunakan file JSON.
3. Mahasiswa mampu menghubungkan konfigurasi dengan logika program.
4. Mahasiswa terbiasa dengan struktur project yang modular.

## BAB II IMPLEMENTASI

### A. PRAKTIKUM (GUIDED)

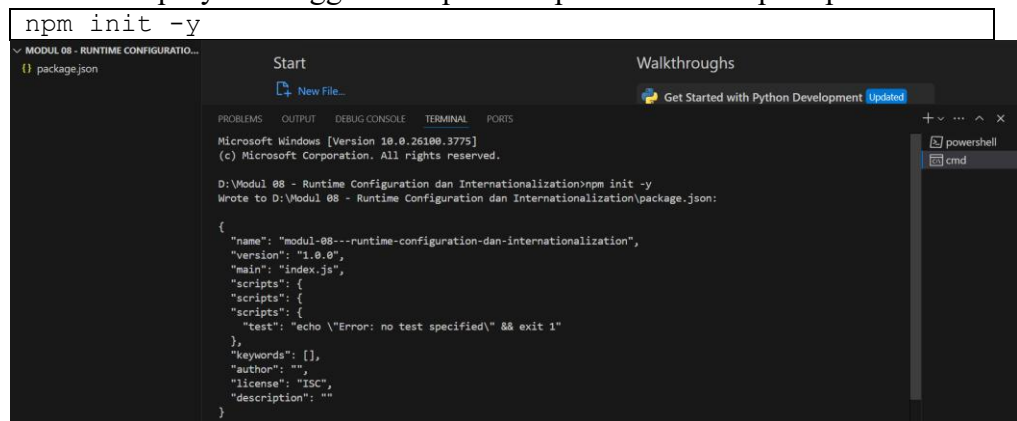
#### Soal Studi Case

Membuat system sederhana untuk transfer uang dengan menggunakan pilihan metode-metode transfer.

#### 1. Persiapan Awal

- Buat folder proyek bernama Modul 08 - Runtime Configuration dan Internationalization.
- Inisialisasi proyek menggunakan perintah pada command prompt:

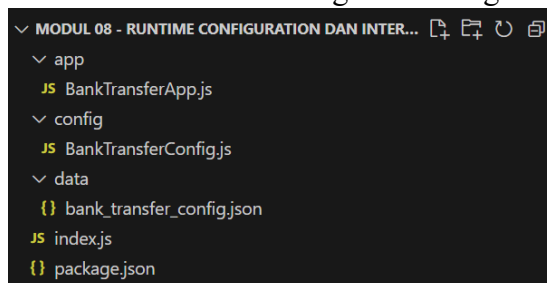
```
npm init -y
```



```
{
  "name": "modul-08---runtime-configuration-dan-internationalization",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Ini akan membuat file package.json.

- Lalu membuat folder dengan file dengan struktur berikut:



```
MODUL 08 - RUNTIME CONFIGURATION DAN INTER...
├── app
│   └── BankTransferApp.js
├── config
│   └── BankTransferConfig.js
├── data
│   └── bank_transfer_config.json
├── index.js
└── package.json
```

#### 2. File Konfigurasi

**File data/bank\_transfer\_config.json:**

```
{
  "lang": "en",
  "transfer": {
    "threshold": 25000000,
    "low_fee": 6500,
    "high_fee": 15000
  },
  "methods": ["RTO (real-time)", "SKN", "RTGS", "BI FAST"],
  "confirmation": {
    "en": "yes",
    "id": "ya"
  }
}
```

```
}  
}
```

### Deskripsi Program

File ini menyimpan seluruh pengaturan yang akan digunakan oleh aplikasi, termasuk bahasa, batas fee, dan daftar metode transfer. Tujuannya agar konfigurasi mudah diubah tanpa harus mengedit kode program.

### 3. Class Konfigurasi

#### File config/BankTransferConfig.js:

```
const fs = require('fs');  
const path = require('path');  
  
class BankTransferConfig {  
  constructor() {  
    this.configPath = path.join(__dirname,  
    '../data/bank_transfer_config.json');  
    this.defaultConfig = {  
      lang: "en",  
      transfer: {  
        threshold: 25000000,  
        low_fee: 6500,  
        high_fee: 15000  
      },  
      methods: ["RTO (real-time)", "SKN", "RTGS", "BI FAST"],  
      confirmation: {  
        en: "yes",  
        id: "ya"  
      }  
    };  
    this.config = this.loadConfig();  
  }  
  
  loadConfig() {  
    if (fs.existsSync(this.configPath)) {  
      const data = fs.readFileSync(this.configPath, 'utf8');  
      return JSON.parse(data);  
    } else {  
      this.saveConfig(this.defaultConfig);  
      return this.defaultConfig;  
    }  
  }  
  
  saveConfig(config) {  
    fs.writeFileSync(this.configPath, JSON.stringify(config,  
    null, 2));  
  }  
}  
  
module.exports = BankTransferConfig;
```

### Deskripsi Program

Class ini bertanggung jawab membaca konfigurasi dari file JSON. Jika file tidak ditemukan, maka akan dibuat otomatis menggunakan default. Ini mendemonstrasikan konsep runtime configuration.

#### 4. Aplikasi Transfer

##### File app/BankTransferApp.js:

```
const readline = require('readline');
const BankTransferConfig =
  require('../config/BankTransferConfig');

class BankTransferApp {
  constructor() {
    this.config = new BankTransferConfig().config;
    this.rl = readline.createInterface({
      input: process.stdin,
      output: process.stdout
    });
  }

  askQuestion(query) {
    return new Promise(resolve => this.rl.question(query,
      resolve));
  }

  async run() {
    const lang = this.config.lang;

    const promptAmount = lang === "en" ?
      "Please insert the amount of money to transfer: " :
      "Masukkan jumlah uang yang akan di-transfer: ";

    const amountStr = await this.askQuestion(promptAmount);
    const amount = parseFloat(amountStr);

    const fee = amount <= this.config.transfer.threshold
      ? this.config.transfer.low_fee
      : this.config.transfer.high_fee;

    const total = amount + fee;

    if (lang === "en") {
      console.log(`Transfer fee = ${fee}`);
      console.log(`Total amount = ${total}`);
    } else {
      console.log(`Biaya transfer = ${fee}`);
      console.log(`Total biaya = ${total}`);
    }

    console.log(lang === "en" ? "Select transfer method:" :
      "Pilih metode transfer:");
    this.config.methods.forEach((method, idx) => {
      console.log(`${idx + 1}. ${method}`);
    });

    await this.askQuestion(lang === "en" ? "Choose a method
      (press Enter after): " : "Pilih metode (tekan Enter setelah):
      ");

    const confirmationPrompt = lang === "en" ?
      `Please type "${this.config.confirmation.en}" to confirm
      the transaction: ` :
      `Ketik "${this.config.confirmation.id}" untuk
      mengkonfirmasi transaksi: `;
  }
}
```

```

        const confirmationInput = await
this.askQuestion(confirmationPrompt);

        if (
            (lang === "en" && confirmationInput.trim().toLowerCase()
=== this.config.confirmation.en) ||
            (lang === "id" && confirmationInput.trim().toLowerCase()
=== this.config.confirmation.id)
        ) {
            console.log(lang === "en" ? "The transfer is completed"
: "Proses transfer berhasil");
        } else {
            console.log(lang === "en" ? "Transfer is cancelled" :
"Transfer dibatalkan");
        }

        this.rl.close();
    }
}

module.exports = BankTransferApp;

```

### Deskripsi Program

Script ini membaca input dari user, menentukan biaya transfer berdasarkan threshold, lalu meminta konfirmasi. Bahasa yang digunakan tergantung konfigurasi, ini mencerminkan konsep internationalization.

#### 5. File Utama

##### File index.js:

```

const BankTransferApp = require('./app/BankTransferApp');

const app = new BankTransferApp();
app.run();

```

#### 6. Run Program

##### Sourcecode

```
node index.js
```

##### Screenshot Output 1:

```

PS D:\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Debugger attached.
Please insert the amount of money to transfer: 5000000
Transfer fee = 6500
Total amount = 5006500
Select transfer method:
1. RTO (real-time)
2. SKN
3. RTGS
4. BI FAST
Choose a method (press Enter after): 1
Please type "yes" to confirm the transaction: yes
The transfer is completed
Waiting for the debugger to disconnect...
PS D:\Modul 08 - Runtime Configuration dan Internationalization>

```

## Screenshot Output 2:

```
PS D:\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Debugger attached.
Please insert the amount of money to transfer: 43000000
Transfer fee = 15000
Total amount = 43015000
Select transfer method:
1. RTO (real-time)
2. SKN
3. RTGS
4. BI FAST
Choose a method (press Enter after): 4
Please type "yes" to confirm the transaction: yes
The transfer is completed
Waiting for the debugger to disconnect...
```

### Deskripsi Program:

Setelah menjalankan program dengan dua input berbeda, berikut hasilnya:

1. **Input: 5000000**

**Output:**

Transfer fee = 6500

Total amount = 5006500

Penjelasan: Nilai transfer 5.000.000 lebih kecil dari threshold 25.000.000, sehingga fee yang digunakan adalah low\_fee = 6.500.

2. **Input: 43000000**

**Output:**

Transfer fee = 15000

Total amount = 43015000

Penjelasan: Nilai transfer 43.000.000 melebihi threshold 25.000.000, sehingga fee yang digunakan adalah high\_fee = 15.000.

Program menggunakan logika perbandingan berikut:

```
const fee = amount <= this.config.transfer.threshold
  ? this.config.transfer.low_fee
  : this.config.transfer.high_fee;
```

Artinya:

1. Jika nominal transfer lebih kecil atau sama dengan threshold, maka biaya transfer menggunakan low\_fee.
2. Jika nominal lebih besar dari threshold, maka menggunakan high\_fee.

Perhitungan ini membuktikan bahwa nilai-nilai pada file konfigurasi (bank\_transfer\_config.json) berhasil digunakan saat runtime tanpa perlu mengubah kode program. Program menjadi dinamis dan mudah disesuaikan hanya dengan mengubah isi JSON.



## B. LATIHAN (CHALLENGE)

Ketentuan:

1. Ubah bahasa default menjadi id lalu jalankan lagi programnya.
2. Tambahkan opsi metode transfer baru di file JSON.
3. Apa yang terjadi kalau input transfer amount bukan angka?

Jawaban:

### 1. Challenge 1: Ubah bahasa default menjadi id lalu jalankan lagi programnya

Perubahan yang dilakukan: File `bank_transfer_config.json` diubah pada bagian `lang` dari `"en"` menjadi `"id"`:

```
data > {} bank_transfer_config.json > lang
1  {
2    "lang": "id",
3    "transfer": {
4      "threshold": 25000000,
5      "low_fee": 6500,
6      "high_fee": 15000
7    },
8    "methods": ["RTO (real-time)", "SKN", "RTGS", "BI FAST"],
9    "confirmation": {
10     "en": "yes",
11     "id": "ya"
12   }
13 }
```

Menjalankan ulang program dan mendapatkan **output** sebagai berikut:

```
PS D:\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Debugger attached.
Masukkan jumlah uang yang akan di-transfer: 5000000
Biaya transfer = 6500
Total biaya = 5006500
Pilih metode transfer:
1. RTO (real-time)
2. SKN
3. RTGS
4. BI FAST
Pilih metode (tekan Enter setelah): 4
Ketik "ya" untuk mengkonfirmasi transaksi: ya
Proses transfer berhasil
Waiting for the debugger to disconnect...
PS D:\Modul 08 - Runtime Configuration dan Internationalization>
```

Bahasa luaran menjadi bahasa indonesia dengan menggunakan fungsi `if-else` pada file `BankTransferApp.js`.

## 2. Challenge 2: Tambahkan opsi metode transfer baru

Perubahan: Tambahkan "QRIS" pada array methods di file:

Hasil perubahan pada **bank\_transfer\_config.json**:

```
data > {} bank_transfer_config.json > ...
1  {
2    "lang": "id",
3    "transfer": {
4      "threshold": 25000000,
5      "low_fee": 6500,
6      "high_fee": 15000
7    },
8    "methods": ["RTO (real-time)", "SKN", "RTGS", "BI FAST", "QRIS"],
9    "confirmation": {
10     "en": "yes",
11     "id": "ya"
12   }
13 }
```

Hasil perubahan pada **BankTransferConfig.js**:

```
config > JS BankTransferConfig.js > BankTransferConfig > constructor > confirmation
4  class BankTransferConfig {
5    constructor() {
6      transfer: {
7        threshold: 25000000,
8        low_fee: 6500,
9        high_fee: 15000
10     },
11     methods: ["RTO (real-time)", "SKN", "RTGS", "BI FAST", "QRIS"],
12     confirmation: {
13       en: "yes",
14       id: "ya"
15     }
16   }
17 }
18
19 ;
```

Mencoba menjalankan ulang program, dan mendapatkan output sebagai berikut:

```
PS D:\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Debugger attached.
Masukkan jumlah uang yang akan di-transfer: 54000000
Biaya transfer = 15000
Total biaya = 54015000
Pilih metode transfer:
1. RTO (real-time)
2. SKN
3. RTGS
4. BI FAST
5. QRIS
Pilih metode (tekan Enter setelah): 5
Ketik "ya" untuk mengkonfirmasi transaksi: ya
Proses transfer berhasil
Waiting for the debugger to disconnect...
PS D:\Modul 08 - Runtime Configuration dan Internationalization>
```

Metode transfer baru QRIS sudah ditambahkan ke sistem dan proses transfer berhasil menggunakan metode QRIS.

### 3. Challenge 3: Apa yang terjadi kalau input transfer amount bukan angka?

Simulasi: User memasukkan huruf, misalnya abc, saat diminta:

Masukkan jumlah uang yang akan di-transfer: abc

#### Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Debugger attached.
Masukkan jumlah uang yang akan di-transfer: lima juta
Biaya transfer = 15000
Total biaya = NaN
Pilih metode transfer:
1. RTO (real-time)
2. SKN
3. RTGS
4. BI FAST
5. QRIS
Pilih metode (tekan Enter setelah): 5
Ketik "ya" untuk mengkonfirmasi transaksi: ya
Proses transfer berhasil
Waiting for the debugger to disconnect...
PS D:\Modul 08 - Runtime Configuration dan Internationalization> |
```

Nilai **amount** akan menjadi **NaN (Not a Number)** jika input bukan angka, karena fungsi `parseFloat()` pada `BankTransfeerApp.js`:

```
app > JS BankTransferApp.js > BankTransferApp > run > fee
4   class BankTransferApp {
17   async run() {
20     const promptAmount = lang === "en" ?
21       "Please insert the amount of money to transfer: " :
22       "Masukkan jumlah uang yang akan di-transfer: ";
23
24     const amountStr = await this.askQuestion(promptAmount);
25     const amount = parseFloat(amountStr);
```

Dengan penjelasan sebagai berikut: (sumber: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/parseFloat](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseFloat) )

- If the argument's first character can't start a legal number literal per the syntax above, `parseFloat` returns `NaN`.

Syntax-wise, `parseFloat()` parses a subset of the syntax that the `Number()` function accepts. Namely, `parseFloat()` does not support non-decimal literals with `0x`, `0b`, or `0o` prefixes but supports everything else. However, `parseFloat()` is more lenient than `Number()` because it ignores trailing invalid characters, which would cause `Number()` to return `NaN`.

Similar to number literals and `Number()`, the number returned from `parseFloat()` may not be exactly equal to the number represented by the string, due to floating point range and inaccuracy. For numbers outside the `-1.7976931348623158e+308` - `1.7976931348623158e+308` range (see `Number.MAX_VALUE`), `-Infinity` or `Infinity` is returned.

Dan perhitungannya menjadi

```
const total = amount + fee; // NaN + fee = NaN
```

### Output:

Biaya transfer: tetap akan dicetak (pakai high\_fee)

Total biaya: Total biaya = NaN

Maka dari itu program tetap berjalan, tapi hasil tidak valid (karena tidak ada validasi input angka).

**Jika saya coba untuk mengimprove persoalan ini**, saya akan menambahkan validasi pada BankTransferApp.js:

```
const amountStr = await this.askQuestion(promptAmount);
const amount = parseFloat(amountStr);

// Tambahkan validasi di sini
if (isNaN(amount)) {
  console.log("Input tidak valid. Harap masukkan angka.");
  this.rl.close();
  return;
}
```

### Output setelah tambahan validasi:

```
PS D:\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Debugger attached.
Masukkan jumlah uang yang akan di-transfer: Hehehe
Input tidak valid. Harap masukkan angka.
Waiting for the debugger to disconnect...
PS D:\Modul 08 - Runtime Configuration dan Internationalization> █
```

### Penjelasan:

Tambahan validasi ini mengecek apakah input yang sudah diparse menjadi angka (amount) adalah NaN (Not a Number). Ini bisa terjadi jika user memasukkan huruf atau simbol yang bukan angka.

Dengan validasi ini, user diberi tahu kesalahan inputnya apa, dan program berhenti secara bersih, tidak terus berjalan dengan hasil yang tidak valid.

## **BAB III**

### **KESIMPULAN DAN SARAN**

#### **A. KESIMPULAN**

Dari praktikum ini, dipelajari dua konsep penting dalam pengembangan perangkat lunak modern, yaitu runtime configuration dan internationalization (i18n). Konfigurasi runtime membantu menjaga fleksibilitas dan portabilitas aplikasi, memungkinkan penyesuaian tanpa harus mengubah kode sumber. Di sisi lain, internationalization memungkinkan aplikasi untuk menjangkau lebih banyak pengguna dari berbagai latar belakang bahasa dan budaya dengan cara yang sistematis dan efisien.

Pemahaman konsep ini akan sangat bermanfaat dalam membangun aplikasi skala besar yang mendukung multi-lingual user interface dan dapat diatur ulang secara dinamis saat dijalankan.

#### **B. REFERENSI**

Modul Praktikum: Konstruksi Perangkat Lunak — Modul 8, Runtime Configuration dan Internationalization, 2023.  
Node.js Documentation - process.env  
Mozilla Developer Network (MDN)  
i18next - Internationalization for JavaScript: <https://www.i18next.com/>  
MDN Web Docs – Internationalization: <https://github.com/dotnet/AspNetCore.Docs>  
GitHub Docs: <https://github.com/dotnet/AspNetCore.Docs>