# Full Stack Enterprise Application Developer

Materi Pembelajaran

Dr. Bambang Purnomosidi D. P.

# Daftar Isi

# 1 Tentang Buku Ini

**Versi**

```
1.0.0-RC--28-Mei-2019--21:58:44
```

Buku ini merupakan buku pegangan program "**Full Stack Enterprise Application Developer**" untuk para mentor, siswa, serta pihak-pihak lain yang berkepentingan dengan proses pendidikan di Praxis Academy. Dengan panduan buku ini, diharapkan proses belajar menjadi lebih efektif dan efisien serta mempunyai pedoman yang jelas. 

```
Praxis Academy
Jl.Garuda no 67, Manukan
Condong Catur
Sleman
Yogyakarta 55283
Indonesia
Email: hello@praxisacademy.id
```

```
8 Web: https//praxisacademy.id
```

# 2 Minggu 01

## Hari 1

### Tujuan

1. Memahami teknologi informasi secara umum dan kaitannya dengan software.
2. Memahami ranah pendidikan yang terkait dengan teknologi informasi: istilah teknik informatika / *informatics*, ilmu komputer / *computer science*.
3. Memahami software paling mendasar: *operating system*
4. Memahami ekosistem sistem operasi saat ini
5. Memahami komponen-komponen sistem operasi - khususnya berbasis Linux
6. Memahami UI di Linux (*GUI* dan *text mode* / *shell*)
7. Memahami dan bisa menggunakan berbagai utilitas dasar dalam Linux
8. Mehamami dasar-dasar dari *shell script*.

### Pembahasan

1. Teknologi informasi dan teknologi software
2. Sistem Operasi: UI, shell, utilities, shell script

### Pembelajaran

```
1 Materi dan Penjelasan
```

1. Halaman Wikipedia - Information Technology, serta komponen-komponen dari TI dengan penekanan lebih pada software.
2. Cara software menempati komputer kita serta bagaimana user berinteraksi dengan *tasks* komputasi. Tekankan pemahaman pada bagaimana sistem operasi bekerja.
3. *Overview* dari daftar sistem operasi yang ada di dunia ini.
4. Komponen-komponen SO Linux.
5. Windowing System yang berfungsi sebagai antarmuka grafis (GUI) di Linux serta impelementasinya di Linux menggunakan X Window System.

6. Keterkaitan X Window System dengan Display Manager, Window Manager, serta Desktop Environment.
7. Linux console dan keterkaitannya dengan Linux shell.
8. Ringkasan dari berbagai command shell terutama yang digunakan di Linux.
9. Beberapa utilitas yang biasanya digunakan pada saat berada di *console* / *shell*. Beberapa utilitas tersebut biasanya berada dalam kelompok GNU Core Utilities dan util-linux. Beberapa petunjuk lainnya adalah Guru99 dan E-Guide - format PDF.
10. Cara membuat shell script.

```
Latihan
```

1. Boot laptop masing-masing, perhatikan proses booting dari awal, jika muncul Grub, usahakan melihat parameter dari Grub tersebut. Cari informasi di Internet tentang parameter Grub tersebut.
2. Cari informasi tentang software yang ada di laptop anda: display manager yang digunakan, window manager yang digunakan, desktop environment yang digunakan, serta shell apa yang digunakan. Dari mana bisa mengetahui informasi tersebut?
3. Cari lokasi dari kernel Linux, sebutkan file-file yang terkait dan kegunaannya.
4. Masuk ke *terminal / comsole* dan kerjakan beberapa perintah berikut melalui *command line / shell*:
    - buat direktori $HOME/praxis/minggu-01/hari-01
    - silahkan coba beberapa perintah di Guru99 dan E-Guide - PDF.
    - tulis hasil dari masing-masing perintah tersebut ke dalam file `cmdline.txt` (gunakan copy paste dari shell)
5. Silahkan coba 30 contoh shell script, masukkan semua file-file yang dihasilkan di direktori $HOME/praxis/minggu-01/hari-01/30shellscript.

```
Kasus
```

Referensi: 1. Wikibooks - Bash Shell Scripting 2. Bash Reference Manual 3. Bash Handbook

Selesaikan kasus-kasus berikut.

0. Semua hasil disimpan pada $HOME/praxis/minggu-01/hari-01/kasus
1. Buat shell script untuk melihat daftar file pada suatu direktori (termasuk direktori anak-anaknya) dan jika terdapat file dengan ekstensi .java - tampilkan tulisan “Ada file Java pada direktori {nama direktori}”. Hasil eksekusi (misalnya):

```
$ cari.sh $HOME/src
Ada file Java pada direktori /home/bpdp/src/hari-01
```

2. Buat shell script untuk menanyakan suatu nama program (misalnya `firefox`), setelah itu mencari PID dari program tersebut dan jika PID program tersebut ada, maka program tersebut akan dimatikan. Saran: gunakan perintah-perintah `ps`, `grep`, `awk`, dan `kill`.

## Hari 2

### Tujuan

1. Siswa memahami peran Git dalam software engineering serta memahami keterkaitan Git dengan istilah-istilah yang biasanya digunakan:
   - software engineering
   - software configuration management
   - version control system
   - distributed version control system
2. Siswa memahami berbagai vendor yang menyediakan fasilitas untuk *remote repository*.
3. Siswa memahami dasar-dasar penggunaan Git serta GitHub dan mampu menggunakan Git serta GitHub untuk *single person development*.
4. Siswa memahami dan mampu membuat file *markdown* untuk dokumentasi di GitHub.
5. Siswa memahami dan mampu menggunakan Git serta GitHub untuk *team* pengembang aplikasi.

### Pembahasan

1. Software engineering, software configuration management, version control. dan distributed version control.
2. Git dan perintah-perintah dasarnya
3. *Markdown* sebagai format untuk dokumentasi
4. Git dan *remote repository* (GitHub, GitLab, Assembla, BitBucket)
5. Git untuk single person development
6. Git untuk tim pengembang aplikasi

### Pembelajaran

```
Materi dan Penjelasan
```

1. Ruang lingkup software engineering.
2. Keterkaitan software engineering dengan SCM - Software Configuration Management.
3. Keterkaitan SCM dengan Version Control dan Distributed Version Control.
4. Git dan keterkaitannya dengan Distributed Version Control
5. Men-*setup* Git - Chapter 1 - Getting Started.
6. Membuat account GitHub serta membuat repo di GitHub.
7. README.md serta mampu menggunakan pemformatan file markdown untuk menuliskan dokumentasi.
8. Penggunaan `git status`, `git add`, `git commit`, `git push` untuk menyimpan ke *remote repo*
9. Penggunaan branching and merging.
10. Pull request untuk repo di GitHub milik sendiri, merging, kemudian sinkronisasi ke lokal repo di komputer.
11. Menggunakan Github untuk kolaborasi tim.

```
Latihan
```

1. Buat repo sesuai keterangan pada panduan umum Praxis Academy untuk hari 1, masukkan semua file-file yang dibuat pada pertemuan hari pertama.
2. Praktikkan Getting Started ini bersama rekan.

```
Kasus
```

1. Praktikkan Team Collaboration with GitHub bersama rekan.

## Hari 3

### Tujuan

1. Siswa bisa memahami keterkaitan antara bahasa pemrograman dengan *compiler/interpreter*.
2. Siswa memahami komponen dari peranti pengembangan (*development tools*) dan bisa mencari komponen-komponen untuk suatu bahasa pemrograman tertentu.
3. Siswa memahami keterkaitan antara Java, JVM, JRE, JDK, JSE, JEE, serta JME.

4. Siswa memahami keterkaitan antara JCP (Java Community Process) dengan spesifikasi Java dan spesifikasi-spesifikasi lainnya di dunia Java.
5. Siswa memahami keterkaitan antara OpenJDK dengan JDK dari berbagai vendor (Oracle, Corretto, dll).
6. Siswa mengetahui riwayat versi dari Java.
7. Siswa mampu menginstall peranti pengembangan Java di komputer masing-masing, setidaknya untuk OpenJDK dan Orache JDK.
8. Siswa mampu menginstall Visual Studio Code serta plugin untuk peranti pengembangan Java. Siswa juga dibebaskan menggunakan editor teks maupun IDE lainnya.
9. Siswa memahami dan mampu membuat *source code* dalam bahasa pemrograman Java, mengkompilasi, serta menjalankan hasilnya - menggunakan *shell command line* maupun dengan membuat file .JAR (dengan file manifest maupun tanpa file manifest).
10. Siswa memahami struktur dasar *source code* dalam bahasa pemrograman Java.
11. Siswa memahmai dan bisa menggunakan variabel, konstanta, operator, ekspresi, statement, dan tipe data dasar di Java.
12. Siswa memahami dan bisa menggunakan perintah-perintah Java untuk mengatur alur kendali program.

### Pembahasan

1. Development tools dan ekosistemnya
2. Dasar-dasar Java:
    - Kompilasi, menjalankan, mempaket dalam bentuk `.jar`.
    - *Tools dan utilities* di JDK.
    - Konstruksi dasar bahasa pemrograman Java: variabel, konstanta, operator, ekspresi, *statement* / pernyataan, tipe data.
    - Pernyataan untuk mengatur kendali program

### Pembelajaran

```
1 Materi dan Penjelasan
```

1. Keterkaitan antara bahasa pemrograman, compiler, dan interpreter.
2. Komponen dari peranti pengembangan (*development tools*) dan bisa mencari komponen-komponen untuk suatu bahasa pemrograman tertentu.
3. Keterkaitan antara Java sebagai bahasa pemrograman, Java sebagai platform software, serta Java Virtual Machine.
4. Perbedaan JRE dengan JDK.

5. Edisi dari JDK: JSE, JavaFX, JEE, Java Embedded, serta JME.
6. Keterkaitan antara JCP (Java Community Process) dengan spesifikasi Java dan spesifikasi-spesifikasi lainnya di dunia Java.
7. Keterkaitan antara OpenJDK dengan JDK dari berbagai vendor - Oracle, Corretto, dll.
8. Riwayat versi dari Java.
9. Cara menginstall JDK 8 (Oracle dan OpenJDK) dari file .tar.gz (bukan dari distro) serta membuat script untuk mengatur instalasi tersebut (berisi perintah `export` env variable yang diperlukan).
10. Instalasi Visual Studio Code serta plugin untuk peranti pengembangan Java.
11. Kompilasi dan menjalankan aplikasi yang diprogram menggunakan Java, lihat juga repo GitHub ini.
12. Struktur dasar *source code* dalam bahasa pemrograman Java.
13. Penggunaan variabel, konstanta, operator - dijelaskan disini dan disini, ekspresi, statement, block, dan tipe data dasar di Java.
14. Penggunaan pernyataan-pernyataan dalam Java untuk mengatur alur kendali program: looping, pengaturan kondisi, switch.
15. Penggunaan array, multidimensional arrays, dan fungsi matematika.
16. Penggunaan String: String: karakter tunggal, split dan join,

```
Latihan
```

1. Install JDK seperti pada point nomor 7 di materi dan penjelasan di atas.
2. Install Visual Studio Code atau editor / IDE apapun yang anda sukai, konfigurasikan supaya bisa digunakan untuk menulis program Java.
3. Kerjakan point nomor 9 di materi dan penjelasan di atas.
4. Kerjakan point nomor 11 - 14 di materi dan penjelasan di atas, buat semuanya masing-masing dalam bentuk *executable* JAR.

```
Kasus
```

Referensi: Bagaimana meminta input dari user di Java?

1. Buat program untuk penjumlahan matriks. Matriks sudah didefinisikan dalam program menggunakan array multidimensi.
2. Perbaiki program penjumlahan matriks tersebut dengan nama lain. Matriks tidak didefinisikan di dalam program, tetepi merupakan hasil input dari pemakai program.

# Hari 4

## Tujuan

1. Siswa memahami pola pikir obyek.
2. Siswa memahami dan mampu membuat *source code* menggunakan dasar-dasar pemrograman berorientasi obyek.
3. Siswa memahami dan mampu menggunakan *package* dalam pembuatan *source code*.
4. Siswa memahami arti penting *build tool* dan mampu menggunakan Gradle untuk proses pengembangan aplikasi menggunakan Java.
5. Siswa memahami perbedaan *dynamic* dengan *static typing* serta arti penting *type system* dalam mencegah *runtime error*.
6. Siswa mampu menggunakan *static typing* di Java.
7. Siswa memahami kemungkinan terjadinya *exception* di Java serta cara menanganinya.

## Pembahasan

1. Pola Pikir Obyek
2. Dasar-dasar Pemrograman Berorientasi Obyek di Java
3. Package di Java
4. *Build tool*: Gradle
5. *Dynamic typing* dan *static typing*
6. Penanganan error / *exception*

## Pembelajaran

```
Materi dan Penjelasan
```

1. Konsep dan contoh dasar OOP - object, class, inheritance, interface
2. Packages di Java, serta tambahan materi ini dan materi ini
3. Build automation tool serta contoh pemanfaatan Gradle sebagai *build automation tool*.
4. Dynamic type - static typing.
5. Exception Handling. Lihat juga materi ini dan materi ini.

```
Latihan
```

1. Salah satu cara untuk menumbuhkan pola pikir obyek adalah penggunaan CRC Card. Pelajari dan buat 1 contoh sistem lain.

2. Kerjakan Java OOP Concepts Tutorial terutama pada bagian *object, class, inheritance, interface*. Buat dalam bentuk *executable .JAR file*, jelaskan dalam README di repo GitHub anda.
3. Kerjakan point 2 di *materi dan penjelasan* di atas.
4. Install Gradle dan kerjakan panduan singkat ini. Jelaskan dalam README.md.
5. Kerjakan point 5 di *materi dan penjelasan* di atas.

```
Kasus
```

Lengkapi / perbaiki *CRC Cards* untuk sistem yang sudah anda buat pada *Latihan point 1*. Implementasikan dalam bentuk source code.

## Hari 5

### Tujuan

1. Siswa memahami arti penting *annotations* dan mampu menggunakannya dalam *source code* dengan baik.
2. Siswa mampu menggunakan *javadoc* untuk membuat dokumentasi dari *source code*.
3. Siswa memahami arti penting dari *generics* dan mampu menggunakannya dalam *source code* dengan baik.

### Pembahasan

1. Annotations
2. Javadoc
3. Generics

### Pembelajaran

```
Materi dan Penjelasan
```

1. Tutorial tentang annotations: dari Oracle JDK, beginnersbook.com, jenkov.com, howtodoinjava.com.
2. Panduan Javadoc dari Oracle dan dari Wikipedia.
3. Tutorial tentang Generics: dari Oracle JDK, Gilad Bracha

```
Latihan
```

1. Kerjakan annotations examples dari Data Flair. Setelah selesai, perbaiki semua *source code* dengan anotasi untuk dokumentasi, setelah itu buat dokumentasi menggunakan Javadoc.
2. Kerjakan tutorial generics dari geeksforgeeks.org.

```
Kasus
```

1. Cari salah satu project kecil yang menggunakan Java di GitHub, buatlah dokumentasinya menggunakan Javadoc.
2. Perhatikan gist untuk generics ini. Kompilasi, buat file .JAR, jalankan. Buat penjelasan tentang generics pada source code tersebut.
3. Kerjakan java2novice.com Java Example Programs tentang Generics.

# 3 Minggu 02

## Hari 1

### Tujuan

1. Siswa memahami UML dan kaitannya dengan software design serta Java
2. Siswa memahami dan mampu menggunakan konsep-konsep lanjut dari OOP di Java.

### Pembahasan

1. UML, *software design / modeling*, OOP, dan Java
2. Object Oriented Programming Lanjut di Java

### Pembelajaran

```
Materi dan Penjelasan
```

1. Halaman Wikipedia untuk UML secara umum, aplikasi dari UML, Software development with UML and modern Java, Software Modeling With UML, Software Design with the UML, Design adn UML Class Diagrams.
2. Tutorial tentang Classes and Objects dari Oracle JDK.
3. Catatan dari corewebprogramming.com
4. Advanced OOP Concepts in Java

```
Latihan
```

1. Kerjakan source code yang ada pada tutorial Classes and Objects di Oracle JDK. Buat menjadi bentuk .JAR menggunakan Gradle.
2. Dengan menggunakan draw.io, gambarkan class diagram dari latihan point 1 di atas.
3. Kerjakan aplikasi ATM dengan menggunakan Java.

```
Kasus
```

Kerjakan kasus *vending machine* yang terdapat di blog ini. Gunakan Gradle untuk mengelola proses *build* dan *run* dari software yang anda buat tersebut. Setelah selesai, jelaskan pada repo GitHub anda.

## Hari 2

### Tujuan

1. Siswa memahami dan mampu membuat *source code* Java yang melibatkan serialisasi dan de-serialisasi.
2. Siswa memahami format XML serta JSON dan mampu menggunakan Java untuk mengolah data dalam serialisasi XML serta JSON
3. Siswa memahami dan mampu menggunakan struktur data dasar di Java menggunakan paket *java utilitiy*.
4. Siswa memahami dan mampu menggunakan struktur data yang terdapat dalam *collections framework*.

### Pembahasan

1. Serialisasi dan de-serialisasi
2. Serialisasi data: XML dan JSON
3. Struktur data dasar di Java menggunakan paket *java utility*.
4. *Java Collections Framework*.

### Pembelajaran

```
Materi dan Penjelasan
```

1. Serialization and Deserialization in Java with Example.
2. Java XML Tutorial.
3. JSON with Java.
4. Java Data Structures
5. 6 Data Structures Every Java Programmer Should Know
6. Java Collections Framework.

```
Latihan
```

1. Kerjakan point 1, 2, 3, 4, dan 6 pada `Materi dan Penjelasan` di atas.

```
Kasus
```

1. Buat 2 buah file JSON dan XML, keduanya mempunyai struktur data dan isi yang sama.
2. Dengan menggunakan Java, baca data dalam masing-masing format tersebut, tampilkan isi data.
3. Bandingkan lama eksekusi untuk membaca file XML dan file JSON tersebut. Mana yang lebih cepat? Jelaskan mengapa demikian.

## Hari 3

### Tujuan

1. Siswa memahami dan mampu menggunakan teknik *Functional Programming* di Java.

### Pembahasan

1. Paradigma *Functional Programming*.
2. *Functional Programming* di Java

### Pembelajaran

```
Materi dan Penjelasan
```

1. Halaman Wikipedia untuk Functional Programming.
2. Java 101: Functional programming for Java developers, Part 1.
3. Java Functional Programming
4. Finally Functional Programming in Java.
5. Java 8 Functional Programming Tutorial
6. FunctionalJava library.

```
Latihan
```

1. Kerjakan tutorial di jenkov.com. Gunakan Gradle dan pastikan bahwa setiap contoh program yang ada disitu harus ada *main*-nya untuk menampilkan hasil-hasil dari teknik *functional programming*.
2. Kerjakan Java 8 Examples - Functional Java.

```
Kasus
```

Kerjakan kasus di repo GitHub

## Hari 4

### Tujuan

1. Siswa memahami *unit testing* dan tujuan penggunaannya di Java
2. Siswa mampu membuat *unit testing* menggunakan JUnit
3. Siswa mampu menggunakan Gradle untuk otomasi *testing*.

### Pembahasan

1. Unit Testing di Java
2. JUnit
3. Gradle dan *unit testing*

### Pembelajaran

```
Materi dan Penjelasan
```

1. Unit Testing - Wikipedia
2. JUnit dan Unit Testing with JUnit - TUtorial.
3. Introduction to Unit Testing with Java.
4. Gradle - Testing in Java and JVM Projects.

```
Latihan
```

1. Kerjakan tutorial - vogella.com

```
Kasus
```

Biasanya programmer akan membuat unit test terlebih dahulu sebelum membuat method / function di Java. Meskipun demikian ada yang berpendapat bahwa hal tersebut hanya akan membuat situasi lebih kompleks. Pada kasus ini kita akan mengimplementasikan *unit testing* di class yang sudah dibuat. Buatlah unit testing pada source code yang sudah anda kerjakan di Tutorial dari Oracle JDK. Test dengan menggunakan Gradle.

## Hari 5

### Tujuan

1. Siswa memahami arti adri Concurrent Programming dan penerapannya di Java

### Pembahasan

1. Memahami *Concurrent Programming*
2. *Concurrent Programming* di Java

### Pembelajaran

```
Materi dan Penjelasan
```

1. Oracle JDK Tutorial - Concurrency
2. Different Approaches to Concurrent Programming in Java
3. Tutorial di vogella.com.
4. Java Concurrency Tutorial
5. Java Concurrency and Multithreading Tutorial

```
Latihan
```

1. Kerjakan 3 dari 5 materi di atas.

```
Kasus
```

Kerjakan Java Concurrency Essentials Tutorial di bagian *Java Concurrency Tutorials – Introduction*, mulai dari Java Concurrency Essentials Tutorial sampai ke Testing Concurrent Applications

# 4 Minggu 03

## Hari 1

### Tujuan

1. Siswa memahami pengertian *software design patterns* dan penggunaannya dalam pembuatan software.
2. Siswa mengetahui dan memahami berbagai tipe *software design patterns* yang ada.
3. Siswa mampu mengaplikasikan *softwarer design patterns* pada peranti pengembangan Java
4. Siswa memahami *pattern* Dependency Injection (DI) dan mampu menggunakannya dalam pengembangan aplikasi.

### Pembahasan

1. Pengertian *software design patterns*
2. Penggunaan *software design patterns* dalam pembuatan software.
3. Tipe-tipe *software design patterns*.
4. *Software design patterns* di Java.
5. *Dependency Injection*.

### Pembelajaran

```
Materi dan Penjelasan
```

1. Ringkasan tentang software design patterns dari Wikipedia.
2. Source Making's Design Patterns
3. Java Design Patterns Example Tutorial
4. Design Patterns Implemented in Java
5. A Quick Intro to DI dan DI di Java.

```
Latihan
```

1. Selesaikan Design Patterns in Java - Tutorial
2. Tutorial di journaldev.com.

```
Kasus
```

1. Ambil salah materi 3 hari di GitHub anda, perbaiki sesuai dengan *software design patterns* yang sudah ada.

## Hari 2

### Tujuan

1. Siswa memahami pengertian *Database Management System* serta fungsinya dalam suatu proyek pengembangan aplikasi.
2. Siswa memahami komponen-komponen dari suatu DBMS.
3. Siswa memahami secara umum SQL: DCL, DDL, DML, dan DQL.
4. Siswa dapat memahami berbagai model data serta relasi antar tabel, E-R diagram, dan normalisasi.
5. Siswa dapat menggunakan perintah-perintah DCL, DDL, DML, dan DQL.
6. Siswa memahami fungsi dari JDBC
7. Siswa dapat menggunakan JDBC untuk mengakses *database*.
8. Siswa memahami fungsi dari ORM.
9. Siswa mengetahui berbagai software ORM.
10. Siswa bisa menggunakan ORM untuk mengakses *database*.

### Pembahasan

1. *Database Management System*
2. Perancangan *database*
3. JDBC
4. ORM

### Pembelajaran

```
Materi dan Penjelasan
```

1. Database dan Database Management System

2. SQL: DCL, DDL, DML, dan DQL.
3. Model data dan normalisasi.
4. JDBC dan JDBC API serta MariaDB Connector/J..
5. ORM.
6. ORM di Java.
7. Ringkasan Hibernate, dokumentasi Hibernate.

```
Latihan
```

1. Install MariaDB.
2. Kerjakan tutorial di tutorialspoint.com dengan menggunakan MariaDB shell.
3. Install MariaDB Connector/J, gunakan Gradle untuk mencoba mengakses MariaDB.
4. Kerjakan tutorial di tutorialspoint.com.

```
Kasus
```

Dari rancangan tabel 3NF, buat berbagai tabel serta relasinya di MariaDB. Buat juga program dengan menggunakan Gradle build tool untuk mengambil data dari tabel-tabel tersebut. Hasil akhir jika run:

```
Janet Jones rents:
1. Pirates of the Caribbean
2. Clash of the Titans
```

## Hari 3

### Tujuan

1. Siswa memahami bahwa ada banyak model data dan implementasi dari model data tersebut dalam NoSQL dan NewSQL.
2. Siswa memahami berbagai tipe DBMS NoSQL
3. Siswa memahami NewSQL serta perbedaannya dengan SQL.
4. Siswa memahami salah satu contoh NoSQL: MongoDB serta model datanya (*document database*).
5. Siswa mampu menggunakan Java untuk mengakses data di MongoDB.
6. Siswa memahami salah satu contoh NewSQL: CockroachDB

7. Siswa mampu menggunakan Java untuk mengakses data di CockroachDB.

### Pembahasan

1. NoSQL dan berbagai kategori NoSQL.
2. NewSQL dan perbedaannya dengan SQL.
3. MongoDB sebagai *document database*.
4. Akses MongoDB menggunakan Java.
5. CockroachDB sebagai *database* NewSQL.
6. Akses CockroachDB menggunakan Java.

### Pembelajaran

```
Materi dan Penjelasan
```

1. NoSQL dan berbagai DBMS NoSQL.
2. NewSQL serta berbagai DBMS NewSQL.
3. MongoDB data model, pengenalan MongoDB, instalasi MongoDB, mongo shell, dan CRUD.
4. MongoDB University.
5. MongoDB Java Driver.
6. FAQ, arsitektur CockroachDB, training, serta dokumentasi CockorachDB.
7. Membangun aplikasi Java dengan CockroachDB.

```
Latihan
```

1. Install MongoDB.
2. Kerjakan mongo shell manual.
3. Kerjakan MongoDB CRUD.
4. Kerjakan MongoDB Reactive Streams Java Driver.
5. Install CockroachDB.
6. Kerjakan getting started.
7. Kerjakan membangun aplikasi Java dengan CockroachDB.

```
Kasus
```

Implementasikan rancangan dirancangan tabel 3NF, buat berbagai tabel serta relasinya di CockroachDB. Buat juga program dengan menggunakan Gradle build tool untuk mengambil data dari tabel-tabel tersebut. Hasil akhir jika run:

```
Janet Jones rents:
1. Pirates of the Caribbean
2. Clash of the Titans
```

## Hari 4

### Tujuan

1. Siswa memahami pengertian RESTful endpoint
2. Siswa memahami cara membangun RESTful endpoint menggunakan Java
3. Siswa memahami cara mengakses RESTful endpoint menggunakan Java
4. Siswa memahami pengertian GraphQL
5. Siswa memahami cara membangun GraphQL server menggunakan Java
6. Siswa memahami cara mengakses GraphQL server menggunakan Java

### Pembahasan

1. RESTful Endpoint
2. GraphQL

### Pembelajaran

```
Materi dan Penjelasan
```

1. Pengertian REST.
2. RESTful API, juga artikel RESTful API ini, serta resources lain.
3. Jersey untuk membuat RESTful API endpoint di Java.
4. Mengakses RESTful API endpoint
5. Pengenalan GraphQL dan The Fullstack Tutorial for GraphQL.
6. GraphQL di Java.

```
Latihan
```

1. Kerjakan Jersey getting started, tidak perlu mengerjakan JavaEE dan Heroku.
2. Akses hasil nomor 1 tersebut dengan menggunakan http client.
3. Kerjakan getting started - GraphQL Java dan Spring Boot.
4. Kerjakan getting started di dokumentasi GraphQL Java.

```
Kasus
```

Buat 2 API, satu menggunakan RESTful API dan satu menggunakan GraphQL. Bandingkan dari sisi:

1. LoC (Line of Code)
2. Source readability
3. Kecepatan akses. Saran: gunakan `time`.

## Hari 5

### Tujuan

1. Siswa memahami arsitektur software *microservices*,
2. Siswa memahami arti dari *full stack* serta komponen-komponennya,
3. Siswa memahami dan mampu menggunakan Micronaut untuk *full stack application development*.

### Pembahasan

1. Arsitektur Software: microservices dan *full stack application*
2. Framework: Dasar-dasar Micronaut

### Pembelajaran

```
Materi dan Penjelasan
```

1. Software architecture.
2. Software architectural patterns.
3. Microservices, resources lain.
4. Front and back ends.

5. Creating your first Micronaut application.
6. Membangun Microservices menggunakan Micronaut.

```
Latihan
```

1. Kerjakan point 5 dan 6 dari materi - penjelasan di atas.

```
Kasus
```

Buatlah aplikasi memnggunakan arsitektur *microservices* dengan minimal ada 2 endpoint. Gunakan Jersey untuk masing-masing RESTful API endpoint. Buat juga 1 API endpoint menggunakan GraphQL. Dari berbagai komponen API tersebut, buat satu program untuk memanfaatkan masing-masing API endpoinit tersebut.

# 5 Minggu 04

## Hari 1

### Tujuan

1. Siswa memahami langkah-langkah yang diperlukan saat berpindah *development tools*.
2. Siswa memahami langkah-langkah untuk memahami JavaScript setelah lebih dahulu memahami Java.
3. Siswa memahami lingkungan eksosistem JavaScript.
4. Siswa memahami dan mampu menggunakan konstruksi dasar dari JavaScript - Node.js maupun JS di browser untuk membuat software.

### Pembahasan

1. Perpindahan *development tools*.
2. Ekosistem JavaScript
3. Konstruksi dasar JavaScript - Node.js dan JS di browser.

### Pembelajaran

```
Materi dan Penjelasan
```

1. How Can We Define a Programming Language menjelaskan keterkaitan antara spesifikasi bahasa pemrograman dengan *reference implementation* dalam bentuk *compiler* maupun *interpreter*. Halaman Wikipedia untuk spesifikasi bahasa pemrograman menjelaskan tentang gambaran umum spesifikasi berbagai bahasa pemrograman.
2. Switching From One Programming Language to Another: The Benefits of Being Flexible menjelaskan tentang perpindahan peranti pengembangan serta bahasa pemrograman.
3. Membandingkan bahasa pemrograman melalui spesifikasi bahasa pemrograman: Java, ECMASCript / JavaScript - lihat di ECMAScript, Kotlin, Go, PHP.
4. Membandingkan implementasi dari masing-masing spesifikasi: Java - OpenJDK, JavaScript - Node.js, Go - Go Compiler, PHP)

5. Ekosistem dari JavaScript / Node.js dan awesome JavaScript memberikan gambaran tentang ekosistem JavaScript.
6. JavaScript di luar browser: Wikipedia dan URL Node.js.
7. Konstruksi dasar JavaScript: Node.js dan JavaScript di browser.

```
Latihan
```

1. Install Node.js.
2. Kerjakan Node.js Quick Guide.
3. Persiapkan peranti pengembangan di sisi client / browser: Google Chrome / Chromium -Firefox.

```
Kasus
```

1. Dengan menggunakan developer tools di Firefox maupun Chrome / Chromium seperti yang telah dikerjakan di atas, kerjakan JavaScript Guide.
2. Dengan menggunakan pengetahuan OOP di Java, ubah artikel di guru99.com dari Java menjadi JavaScript.

## Hari 2

### Tujuan

1. Memahami dan bisa mengintegrasikan HTML, CSS, dan DOM dalam halaman Web
2. Memahami dan mampu menggunakan JavaScript untuk manipulasi tampilan sisi front end di browser.
3. Memahami dasar-dasar cara kerja Vue.js serta penggunaan berbagai tools yang terkait.

### Pembahasan

1. HTNL, CSS, dan DOM
2. JavaScript untuk Front End.
3. Dasar-dasar Vue.js

### Pembelajaran

```
Materi dan Penjelasan
```

1. MDN menyediakan peta pembelajaran untuk sisi Web front end.
2. Dokumentasi untuk memulai belajar Web.
3. Dasar HTML.
4. Dasar CSS.
5. Dasar DOM.
6. JavaScript dan DOM dari guru99.com.
7. JavaScript dan DOM dari tutorialspoint.com.
8. Introduction to Vue.js.

```
Latihan
```

1. Kerjakan point 3 - 7 untuk materi dan penjelasan di atas.
2. Kerjakan instalasi Vue.js seperti di point 8, kerjakan juga beberapa pengenalan Vue.js yang ada pada dokumen itu.

```
Kasus
```

Cobalah membuat contoh materi-materi yang ada di Introduction to Vue.js tanpa menggunakan Vue.js - hanya menggunakan JavaScript biasa untuk memanipulasi DOM.

## Hari 3

### Tujuan

1. Memahami dan mampu menggunakan Vue.js untuk membangun front end pada suatu Web.
2. Memahami dan mampu menggunakan berbagai tools serta ekosistem dari Vue.js

### Pembahasan

1. Vue.js
2. Ekosistem Vue.js

### Pembelajaran

```
Materi dan Penjelasan
```

1. Vue.js Guide: Essentials, Components, Transitions and Animations, Reusability and Compositions, Tooling, Scaling Up.
2. Ekosistem Vue.js: Tooling dan Core Libraries - pilih menu *Ecosystem* dari Web Vue.js.

```
Latihan
```

1. Kerjakan Vue.js examples.

```
Kasus
```

1. Buatlah aplikasi untuk mengambil commit dari GitHub untuk suatu project. Aplikasi tersebut menampilkan input berupa nama repository dari project di GitHub kemudian mengambil data commits dari project tersebut.
2. Perbaiki aplikasi tersebut dengan mengambil branch dari project tersebut. User bisa memilih branch yang akan ditampilkan *commit history*-nya.

## Hari 4

### Tujuan

1. Siswa memahami arsitektur microservices yang scalable untuk aplikasi level enterprise serta kaitannya dengan Micronaut.
2. Siswa memahami dan mampu menggunakan DBMS menggunakan Micronaut.
3. Siswa mampu membuat aplikasi scalable berbasis arsitektur *microservices* menggunakan Micronaut serta Vue.js

### Pembahasan

1. Arsitektur microservices dan Micronaut Framework.
2. Akses DBMS di Micronaut Framework.
3. Micronaut dan service GraphQL.
4. Integrasi *front end* (Vue.js) dan *back end* di Micronaut.

### Pembelajaran

```
Materi dan Penjelasan
```

1. Arsitektur Microservices dan Micronaut
2. Akses DBMS di Micronaut: JPA dan Hibernate - MyBatis.
3. Membuat service di Micronaut menggunakan GraphQL.
4. Micronaut dan Vue.js - contoh project.

```
1 Latihan
```

Kerjakan 2- 4 pada point materi dan penjelasan di atas.

```
1 Kasus
```

Buatlah aplikasi terintegrasi dengan menggunakan arsitektur *microservices*: ada 2 service (1 REST dan 1 GraphQL). Service mengambil data dari DBMS. Layer presentasi (*front end*) menggunakan Vue.js. Skema data serta tanpilan dan fungsionalitas dari aplikasi bebas.