

# Dasar-dasar Pemrograman Web Menggunakan Golang dan ReactJS



**Golang**



**React**

AHMAD FATHONI R  
RONI ANDARSYAH



# **Dasar-dasar Pemrograman Web**

## **Menggunakan Golang dan ReactJS**

Ahmad Fathoni R

Roni Andarsyah



PT. Penerbit Buku Pedia  
2023

# **Dasar-dasar Pemrograman Web Menggunakan Golang dan React JS**

Penulis:

Ahmad Fathoni R

Roni Andarsyah

ISBN:

*Editor:*

Rolly Maulana Awangga

*Penyunting:*

*Desain sampul dan Tata letak:*

Ahmad Fathoni R

*Penerbit:*

Penerbit Buku Pedia

*Redaksi:*

Athena Residence Blok. E No. 1, Desa Ciwaruga,

Kec. Parongpong, Kab. Bandung Barat 40559

Tel. 628-775-2000-300

Email : penerbit@bukupedia.co.id

*Distributor:*

Informatics Research Center

Jl. Sariasih No. 54

Bandung 40151

Email : irc@ulbi.ac.id

Cetakan Pertama, 2023

Hak Cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan

Dengan cara apapun tanpa ijin tertulis dari penerbit

## PRAKATA

**R***epresentational State Transfer* merupakan salah satu metode pendekatan yang popular dalam membangun aplikasi. REST API akan memungkinkan komunikasi antara klien dan server melalui protokol HTTP dan menggunakan HTTP method yang telah ditentukan, aplikasi yang menggunakan REST API dapat berkomunikasi secara terstruktur dan konsisten.

Buku ini hadir sebagai panduan praktis para pengembang perangkat lunak yang tertarik untuk mempelajari cara membangun aplikasi web modern dengan menggunakan teknologi REST API, salah satunya adalah Gin *Framework* yang menggunakan Bahasa pemrograman Golang. Dalam buku tutorial ini akan membahas bagaimana cara penggunaan REST API dalam konteks pembangunan sebuah *E-Library*, selain itu anda juga akan dipandu bagaimana cara untuk mengintegrasikan REST API dengan framework JavaScript popular yaitu ReactJS, serta tahapan *deployment* yang akan membuat aplikasi dapat diakses oleh publik.

Link Github: [https://github.com/bukped/elibrary\\_v3](https://github.com/bukped/elibrary_v3)

# DAFTAR ISI

PRAKATA .....	i
DAFTAR ISI .....	ii
BAB 1 Pengenalan Tools .....	5
A. Visual Studio Code .....	5
B. Web Browser .....	5
C. Postman .....	5
D. PostgreSQL .....	6
E. Golang .....	6
F. <i>Application Programming Interface</i> .....	6
G. ReactJS .....	7
H. Website .....	7
I. <i>Framework</i> .....	7
J. <i>Microservice</i> .....	7
K. <i>Gin Framework</i> .....	8
BAB 2 Instalasi Kebutuhan Perangkat .....	9
A. Node JS dan React .....	9
B. Instalasi Postman .....	14
C. Instalasi PostgreSQL .....	15
D. Instalasi dan Setup Golang .....	21
E. Instalasi Gin .....	25
BAB 3 Tahapan Pembuatan Front-end .....	28
A. Pembuatan Front-End Buku .....	28
1. Menyiapkan Struktur Folder dan File .....	28
2. Menyiapkan <i>dependency</i> .....	29
3. Menyiapkan Koneksi API .....	29
4. Menyiapkan Rute API .....	30
5. Membuat Kode Untuk Tampilan Book List .....	31

6.	Set Up Index dan App JS .....	36
7.	Mencoba Pada Browser.....	38
8.	Pembuatan Halaman UpdateBook .....	38
B.	Pembuatan Front-End Peminjaman Buku .....	42
1.	Import dan Pembuatan State .....	42
2.	Membuat Tampilan dan Assign Function .....	44
<b>BAB 4 Tahapan Pembuatan Back-end.....</b>		<b>50</b>
A.	Perancangan Endpoint Autentifikasi User .....	50
1.	Koneksi Database.....	50
2.	Membuat Initializer Environment .....	51
3.	Membuat File Initializer.....	51
4.	Pembuatan Model .....	53
5.	Pembuatan Controller User.....	55
6.	Pembuatan Middleware .....	60
7.	Pembuatan URL Endpoint Autentifikasi .....	62
8.	Uji Coba Endpoint User.....	63
B.	Perancangan Endpoint Buku.....	64
1.	Membuat Model Buku.....	64
2.	Membuat Controller Buku.....	65
3.	Pembuatan URL Endpoint.....	69
4.	Percobaan Endpoint Buku Pada Postman .....	70
C.	Pembuatan Endpoint Pinjam Buku .....	73
1.	Membuat Model dan Controller Pinjam .....	73
2.	Pengujian Endpoint Pinjaman Buku .....	78
<b>BAB 5 Penggunaan Git dan Deploy Aplikasi .....</b>		<b>80</b>
A.	Penggunaan Git SCM .....	80
B.	Tahapan Deploy Database .....	88
C.	Tahapan CI/CD .....	97

D.	Tahapan Deploy Backend .....	101
E.	Tahapan Deploy Frontend .....	105
<b>BAB 6 Latihan Soal.....</b>		<b>109</b>
A.	Pilihan Ganda .....	109
B.	Isian .....	112
<b>Daftar Pustaka.....</b>		<b>114</b>
<b>TENTANG PENULIS.....</b>		<b>115</b>

# BAB 1

## Pengenalan Tools

---

### A. Visual Studio Code

Visual Studio Code atau biasa disebut dengan VSCode adalah aplikasi editor teks yang digunakan untuk membuat atau mengubah teks yang berguna pada pembuatan sistem. VSCode ini dibuat oleh perusahaan besar yaitu Microsoft untuk platform Windows, Linux, dan juga MacOS. VSCode dapat mengenali berbagai jenis bahasa pemrograman seperti Python, Java, C++, Ruby, PHP dan GO. VSCode juga terdapat fitur agar dapat mempermudah menulis serta menjalankan program dalam bentuk ekstensi yang bisa diunduh dalam aplikasinya sendiri (Microsoft, n.d.).

### B. Web Browser

Web browser merupakan perangkat lunak yang digunakan untuk mengakses informasi yang tersedia dalam sebuah aplikasi web. Informasi yang tersedia biasanya diberikan dalam format teks, gambar, maupun video. Saat ini terdapat banyak Web Browser populer seperti Mozilla Firefox, Google Chrome, Opera, Microsoft Edge, dan Safari. Setiap web browser memiliki kelebihan dan kekurangannya masing masing, web browser bisa digunakan sesuai dengan keinginan preferensi pengguna.

### C. Postman

Postman adalah aplikasi yang berfungsi sebagai REST *Client* untuk menguji coba API yang telah dibuat dengan mengirimkan HTTP *Request* kedalam aplikasi tersebut (Edy, Ferdiansyah, Pramusinto, & Waluyo, 2019). Pada Postman terdapat banyak fitur untuk membantu pengembangan REST API seperti pembuatan dokumentasi, publikasi API, dan penggunaan *request* yang beragam seperti GET, POST, PUT, Delete, dll.

## D. PostgreSQL

PostgreSQL adalah RDBMS yang dapat diakses oleh semua orang karena aplikasinya sendiri merupakan open-source. PostgreSQL menyediakan layanan RDBMS yang stabil yang didukung oleh banyak pengembangan fitur dari komunitas, RDBMS ini sangat diminati oleh banyak pengembang aplikasi web, mobile, dan aplikasi analitik (Makris, Tserpes, Spiliopoulos, & Anagnostopoulos, 2019).

PostgreSQL ini akan digunakan sebagai perangkat lunak untuk menunjang manajemen *database* dalam aplikasi ini agar mempermudah proses pengembangan aplikasi kali ini.

## E. Golang

Golang merupakan salah satu bahasa program diciptakan oleh Karyawan Google yaitu Robert Griesemer, Rob Pike, dan Ken Thompson. Tujuan dari dibuatnya golang adalah menciptakan bahasa pemrograman yang ekspresif, cepat, efisien, dan mudah untuk ditulis. Bahasa pemrograman seperti C atau C++ merupakan bahasa pemrograman yang cepat saat dieksekusi namun dalam cara penulisan terlalu sulit untuk diterapkan, adapun bahasa pemrograman yang dalam penulisan cukup singkat dan simpel tetapi dalam segi pembuatan program dan eksekusi yang tidak begitu efisien seperti Java dan Python (McGrath, 2020). Golang adalah salah satu bahasa pemrograman *open-source* atau dapat digunakan oleh siapa saja, bentuk dari bahasa pemrograman ini mempunyai struktur yang mirip dengan bahasa pemrograman C namun dengan penulisan yang lebih singkat dengan tujuan menghindari kompleksitas dan kesulitan dalam penerapan.

Golang mempunyai kelebihan lainnya seperti dapat menggunakan kinerja CPU dengan mengefisiensikan kinerja dari bagian multi-core agar mempermudah komputer dalam mengeksekusi banyak *task* secara bersamaan namun tidak memberatkan perangkat itu sendiri.

## F. Application Programming Interface

Application Program Interface (API) merupakan sebuah service yang menyediakan ketersediaan data yang nanti akan direquest oleh aplikasi client respon dari request tersebut biasanya memiliki format json (Kurniawan, 2020).

Bentuk respon dari API dibagi menjadi beberapa bagian yaitu ada GET, POST, PUT, dan Delete. Dari respon tersebut masing-masing memiliki fungsinya, salah satunya GET merupakan perintah untuk melakukan request pengambilan data dari sebuah server ke client.

## **G. ReactJS**

ReactJS merupakan library dari JavaScript open-source yang digunakan pada segi front-end untuk membangun User Interface berdasarkan komponen dari UI yang digunakan. ReactJS diciptakan oleh Facebook dan komunitas developer. React dapat digunakan pada pengembangan sebuah sistem berbasis website, mobile, atau kerangka kerja yang berasal dari server-side (Freeman, 2019). Jadi secara singkat ReactJS merupakan kerangka kerja yang dikhususkan untuk membuat sebuah tampilan website berdasarkan komponen yang ada pada library ReactJS itu sendiri ataupun menggunakan library tambahan untuk mempercantik tampilan pada website.

## **H. Website**

Website merupakan kumpulan dari halaman situs, biasanya dirangkum dalam sebuah domain yang bertempat dalam World Wide Web (WWW) di internet. Halaman web biasanya ditulis dalam format HTML, yaitu protokol yang digunakan untuk menampilkan informasi yang diambil dari web server kepada pengguna melalui browser (Pamungkas, 2018).

## **I. Framework**

Framework merupakan komponen pemrograman yang berfungsi untuk menyediakan berbagai skrip dalam bahasa pemrograman tertentu. Penggunaan framework akan mempermudah dalam penulisan program dan membantu dalam menjalankan program agar dapat berjalan dengan semestinya (Sari & Wijanarko, 2019).

## **J. Microservice**

Microservice merupakan gaya arsitektur yang dibangun di atas konsep modularisasi yang dibuat dengan baik, yaitu dengan menggunakan alamat dari sebuah aplikasi yang berbeda. Layanan microservice menawarkan kapabilitas

bisnis yang menawarkan akses internal terhadap sebuah data (Mendonca, Jamshidi, Garlan, & Pahl, 2019). Contoh dari microservice sendiri adalah API, karena dalam sebuah API akan menawarkan akses kedalam internal dengan melakukan request dari client menuju server yang dimana akan direspon dengan data.

#### K. **Gin Framework**

Gin merupakan *Framework* yang diprogram dengan bahasa pemrograman Golang. Gin akan memungkinkan Golang untuk berjalan lebih cepat karena penggunaan memori yang cukup ringan dan performa API yang dapat diprediksi. Gin juga mendukung penggunaan Middleware yang membuat penggunaan autorisasi menjadi lebih mudah (Gin, n.d.). Kelebihan dalam menggunakan Gin sendiri adalah mempunyai *Error Management* yang akan memungkinkan untuk mempermudah mencari *Error* pada sintax ataupun pada kode yang tersembunyi yang sulit ditemukan sekalipun.

# BAB 2

## Instalasi Kebutuhan Perangkat

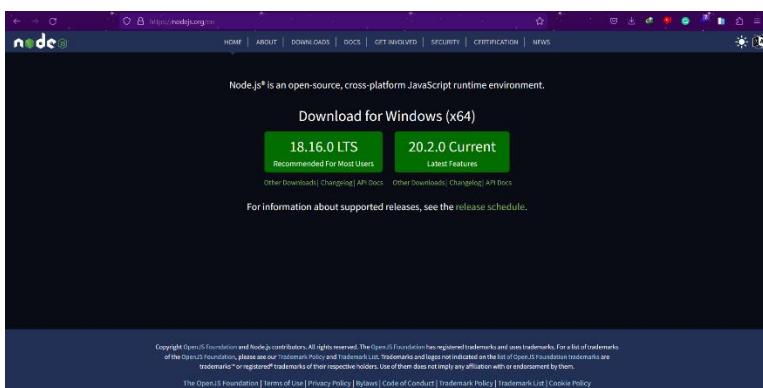
Sebelum memasuki tahap pengembangan aplikasi, sebelumnya ada yang harus anda persiapkan terlebih dahulu. Yaitu instalasi kebutuhan perangkat yang akan digunakan untuk membuat dan menjalankan pada saat mengembangkan aplikasi yang akan dijelaskan pada tahapan berikut.

### A. Node JS dan React

Pada bagian ini akan dilakukan instalasi terlebih dahulu Node JS agar dapat menjalankan *project-project* yang membutuhkan *dependency* dari Node. Tahapan-tahapan akan dijelaskan dalam berikut:

#### 1. Download Node JS

Instalasi Node JS dapat dilakukan dengan mengakses terlebih dahulu laman situs official Node JS dengan url <https://nodejs.org/en>, Lalu pilih sesuai dengan perangkat yang digunakan.

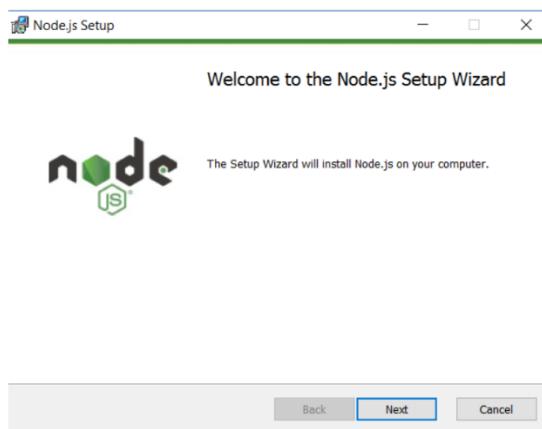


Gambar 1 Halaman Official Node JS

Pada Gambar 1 pilihlah versi 18.16.0 yang merupakan versi yang paling stabil dan dipilih oleh banyak pengguna tunggu hingga proses unduh selesai.

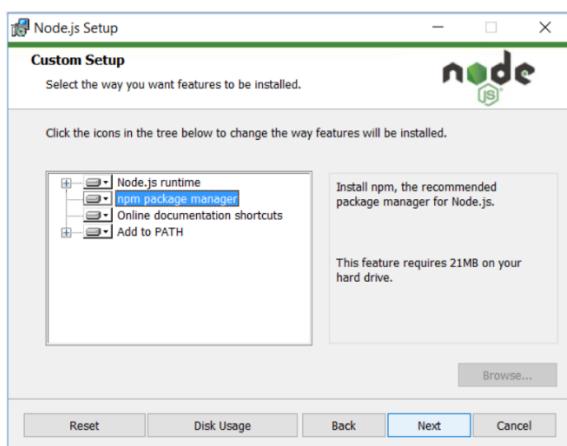
## 2. Installasi Node JS

Setelah proses pengunduhan selesai, selanjutnya adalah membuka file yang sudah diunduh pada laman web official dari node.



Gambar 2 Tampilan awal installasi

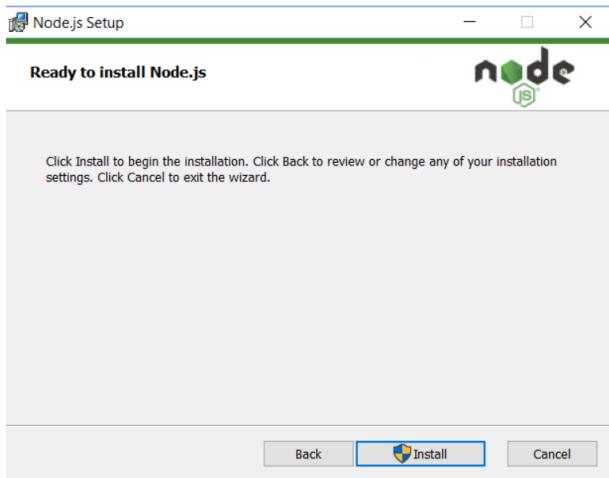
Pada gambar 2 merupakan tampilan awal dari proses installasi Node JS, disini Anda hanya perlu menekan next pada tampilan.



Gambar 3 Pengaturan Installasi Node JS

Pada Gambar 3 akan dilakukan pengaturan untuk Node untuk mengatur

PATH dan runtime dari npm agar dapat digunakan untuk mengatur atau menambah dependency dari suatu aplikasi, disini digunakan saja pengaturan secara default dan tekan pada next.



Gambar 4 Tahap Terakhir Installasi

Pada Gambar 4 adalah tahapan terakhir dari installasi Node, karena semua pengaturan sudah dilakukan. Tekan pada tombol install apabila muncul pop up administrator setting anda hanya perlu menekan pada tombol “yes” dan installasi akan berjalan secara otomatis tunggu hingga proses installasi selesai.

A screenshot of a terminal window titled "C:\WINDOWS\system32\cmd". The window shows the command "node -v" followed by the output "v10.10.0". Below it, the command "npm -v" is shown with the output "8.11.0". The prompt "C:\Users\afato>" is at the bottom.

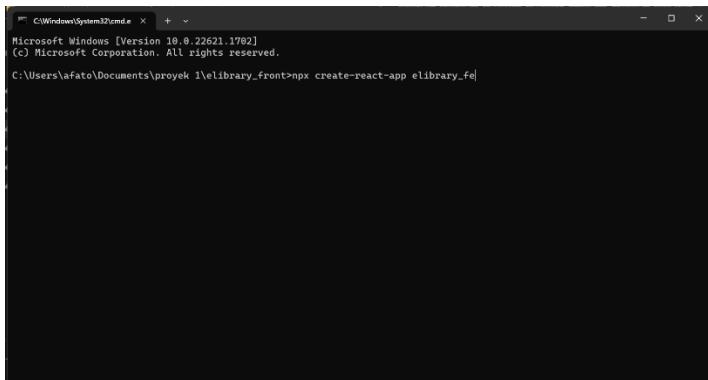
Gambar 5 Pengecekan Installasi Node

Setelah proses installasi selesai, selanjutnya adalah melakukan pengecekan node menggunakan terminal pada perangkat masing-masing dengan cara mengetikkan “node -v” dan “npm -v” apabila output yang dikeluarkan sesuai

dengan Gambar 5, maka installasi node dan NPM sudah berhasil dilakukan.

### 3. Installasi ReactJS

Apabila proses installasi dari Node dan modul NPM sudah terpasang dengan baik, maka proses selanjutnya adalah melakukan persiapan ReactJS agar dapat membuat bagian dari *front-end* aplikasi. Berikut adalah tahapan-tahapannya.



Gambar 6 Command Prompt React

Pertama buka terminal pada platform anda lalu, ketikkan “`npx create-react-app <nama aplikasi>`” lalu tekan enter dan tunggu proses pengunduhan ReactJS hingga selesai.

A screenshot of a Windows Command Prompt window showing the output of the 'npx create-react-app elibrary\_fe' command. The window content includes:

```
Inside that directory, you can run several commands:
  npm start
    Starts the development server.
  npm run build
    Bundles the app into static files for production.
  npm test
    Starts the test runner.
  npm run eject
    Removes this tool and copies build dependencies, configuration files,
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:
  cd elibrary_fe
  npm start

Happy hacking!
  npm notice New major version of npm available! 8.11.0 -> 9.6.6
  npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.6
  npm notice Run npm install -g npm@9.6.6 to update!
  npm notice

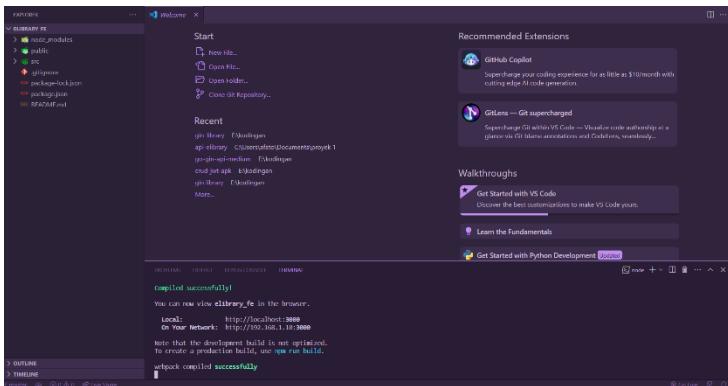
C:\Users\afato\Documents\proyek\1\elibrary_front>cd elibrary_fe
C:\Users\afato\Documents\proyek\1\elibrary_front\elibrary_fe>code .
```

The output shows the creation of a new React application named 'elibrary\_fe' and provides instructions for starting it with 'npm start'.

Gambar 7 Command Prompt setelah proses unduh

Setelah proses pengunduhan React telah selesai, selanjutnya masuk ke direktori folder yang terbuat secara otomatis pada saat pemasangan modul

React dengan menggunakan command “cd <nama\_aplikasi>” lalu tekan enter. Selanjutnya ketikkan “code .” agar dapat membuka text editor Visual Studio Code untuk melakukan perubahan atau penambahan pada kode.



Gambar 8 Tampilan Struktur React JS pada VSCode

Apabila Visual Studio Code telah terbuka, selanjutnya adalah membuka terminal baru pada Visual Studio Code dan mengetikkan perintah berupa “npm start”.

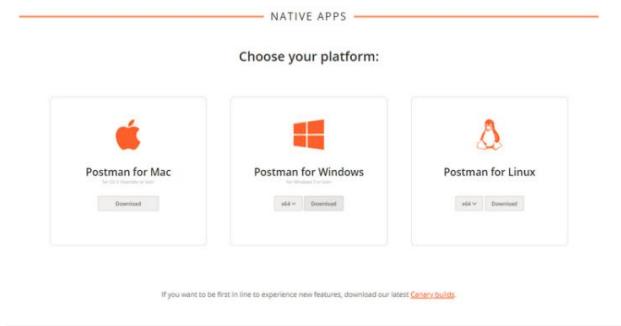


Gambar 9 Tampilan Awal React JS

Setelah mengetikkan “npm start”, browser akan terbuka secara otomatis. Apabila tampilan sudah sesuai dengan Gambar 9 maka proses pemasangan React JS sudah berhasil dan siap untuk digunakan.

## B. Instalasi Postman

Postman adalah perangkat lunak untuk melakukan pengetesan terhadap suatu API atau pembuatan dokumentasinya. Pada bagian ini akan dilakukan pemasangan terlebih dahulu Postman untuk melakukan pengetesan API. Berikut adalah tahapannya.



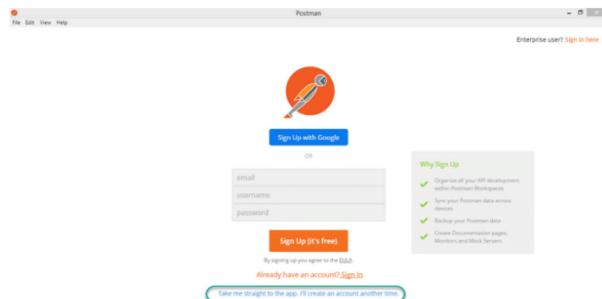
Gambar 10 Halaman Official Postman

Pada Gambar 10 merupakan tampilan *Landing Page* dari web official Postman yang bisa diakses dengan url berikut <https://www.postman.com/downloads/> dan unduh sesuai dengan perangkat yang digunakan.



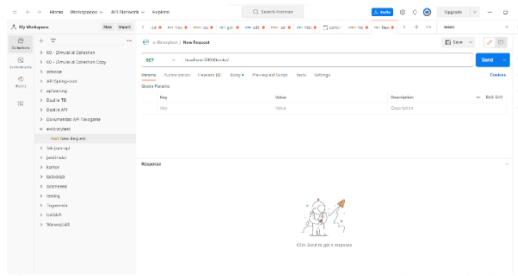
Gambar 11 Proses Instalasi Postman

Pada Gambar 11 adalah saat proses instalasi Postman berjalan, instalasi akan otomatis dilakukan dengan pengaturan *default* pada saat membuka file .exe dari instalasi Postman yang sudah diunduh pada tahap sebelumnya.



Gambar 12 Tahap Awal Postman

Setelah instalasi Postman selesai, selanjutnya akan dihadapkan pada halaman *Landing Page* postman. Pada tahap ini tidak akan langsung memasuki aplikasi secara langsung melainkan *Login* dahulu atau *Sign Up* apabila belum memiliki akun. Anda bisa untuk melewati tahap tersebut dengan menekan tombol “*Take me straight to the app*” agar dapat langsung menggunakan aplikasi tanpa tahap pendaftaran atau login.



Gambar 13 Halaman Utama Postman

Pada Gambar 13 adalah penampilan dari halaman utama Postman. Halaman ini bisa digunakan untuk melakukan tes terhadap API yang sudah dibuat atau mencoba API publik menggunakan Postman.

## C. Instalasi PostgreSQL

Pada tahapan ini akan dilakukan installasi RDBMS yaitu PostgreSQL agar dapat mengatur data-data yang akan diinputkan dan menunjang berjalannya aplikasi agar berjalan dengan semestinya. Berikut adalah tahapan-tahapannya.

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
15.3	<a href="#">postgresql.org</a> ⓘ	<a href="#">postgresql.org</a> ⓘ	☒	☒	Not supported
14.0	<a href="#">postgresql.org</a> ⓘ	<a href="#">postgresql.org</a> ⓘ	☒	☒	Not supported
13.11	<a href="#">postgresql.org</a> ⓘ	<a href="#">postgresql.org</a> ⓘ	☒	☒	Not supported
12.15	<a href="#">postgresql.org</a> ⓘ	<a href="#">postgresql.org</a> ⓘ	☒	☒	Not supported
11.29	<a href="#">postgresql.org</a> ⓘ	<a href="#">postgresql.org</a> ⓘ	☒	☒	Not supported
10.23+	☒	☒	☒	☒	☒

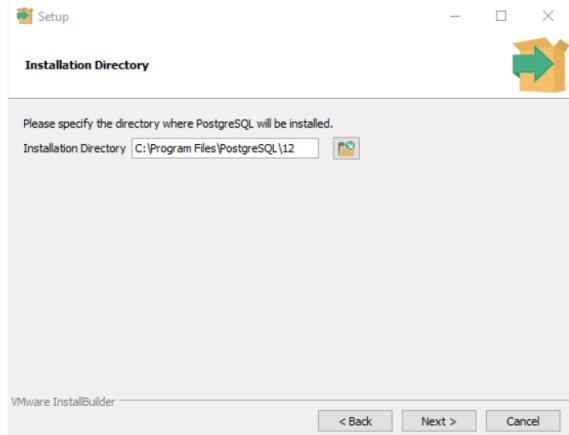
Gambar 14 Kumpulan Link Download Postgres

Tahap pertama yang harus dilakukan saat melakukan instalasi Postgres, adalah mengakses dahulu website official dari Postgres dengan mencarinya dengan keyword PostgreSQL pada search engine apapun. Pilih sesuai dengan perangkat yang digunakan dan arsitektur dari sistem operasi yang digunakan.



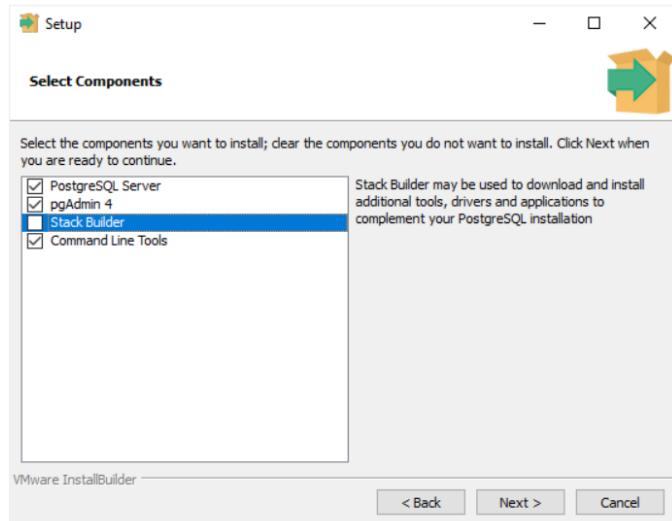
Gambar 15 Tampilan Awal Setup

Setelah proses unduh selesai, buka file installasi yang berformat .exe dan tampilan akan terlihat seperti pada gambar 15. Tekan pada next untuk menuju tahap selanjutnya.



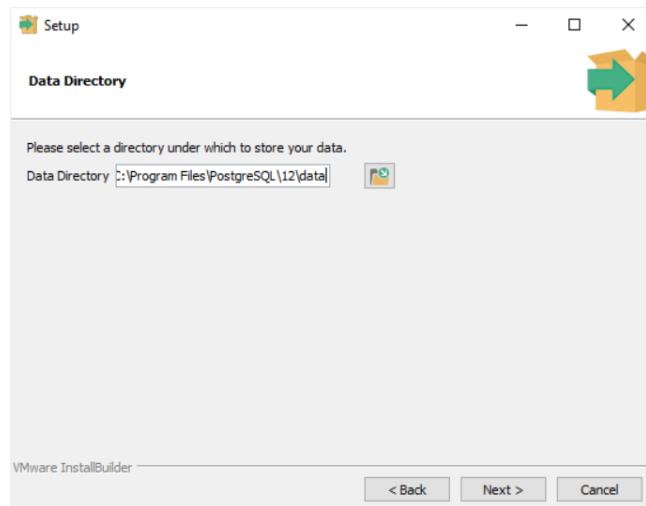
Gambar 16 Pengaturan Lokasi Direktori Postgres

Aturlah direktori dari Postgres sesuai dengan yang diinginkan dengan menekan ikon dari folder. Anda dapat menyimpan direktori dari Postgres dimanapun.



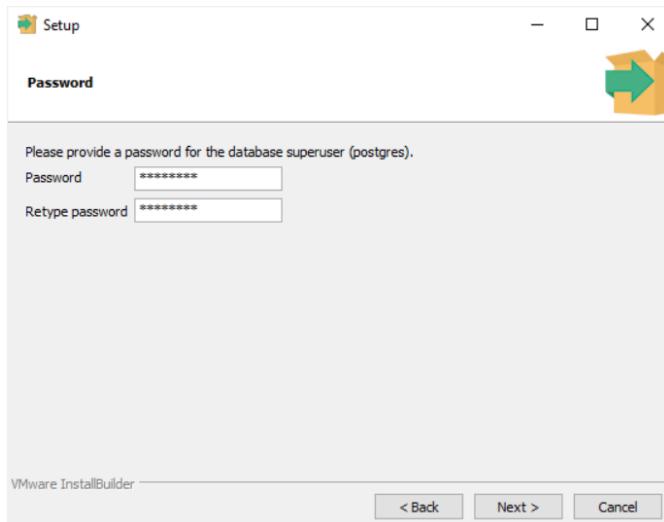
Gambar 17 Pengaturan Komponen Postgres

Pada tahapan ini akan ditampilkan pemilihan komponen seperti pada Gambar 17. Bagian ini dibiarkan saja secara default dan tekan tombol next untuk lanjut.



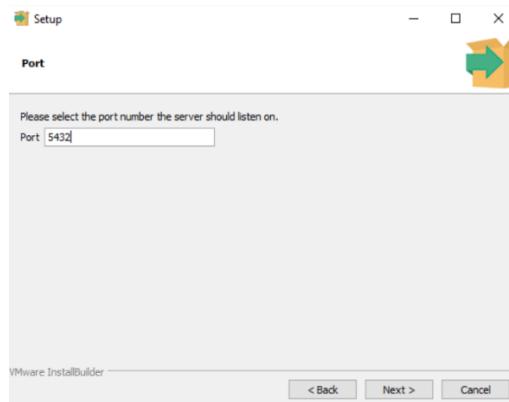
Gambar 18 Pengaturan Direktori Untuk Data

Pada Gambar 18 digunakan untuk mengganti direktori untuk data utama pada Postgres, pengaturan tersebut dapat diganti dengan menekan ikon folder dan menyesuaikannya dengan lokasi yang diinginkan. Apabila sudah menentukan lokasi yang sesuai lanjut dengan menekan tombol next.



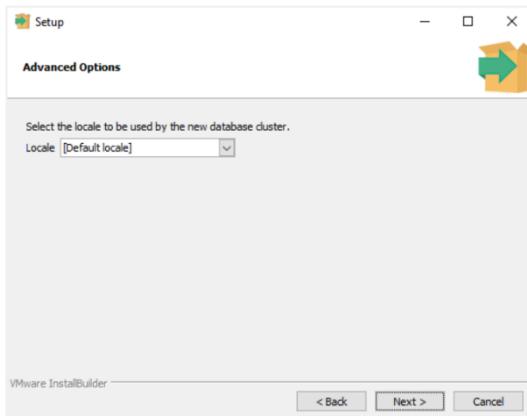
Gambar 19 Pembuatan Password Superuser

Pada Gambar 19 merupakan tahapan pembuatan Password untuk Superuser. Gunakanlah password yang mudah diingat agar tidak mempersulit pada saat membuka aplikasi dari Postgres.



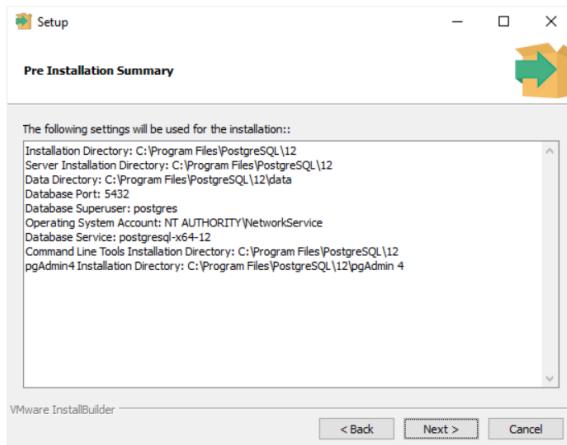
Gambar 20 Pengaturan Port Postgres

Pada tahap ini digunakan untuk mengatur Port pada postgres yang ditampilkan pada Gambar 20. Port diatur sedemikian rupa agar tidak bertabrakan dengan Port lainnya.



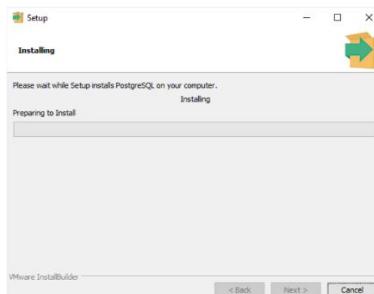
Gambar 21 Pengaturan lokasi cluster postgres

Pada gambar 21 merupakan proses pengaturan cluster database. Pilih pengaturan *default* disini tanpa mengganti apapun.



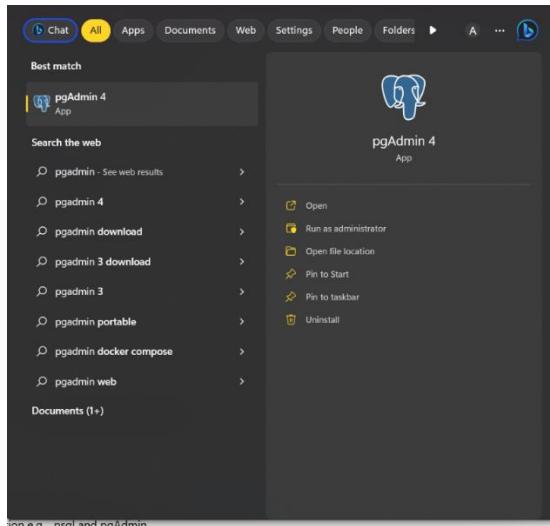
Gambar 22 Peninjauan Komponen Postgres

Pada Gambar 22 menunjukkan halaman instalasi untuk meninjau kembali komponen yang akan dimasukan pada saat proses installasi. Apabila komponen yang dibutuhkan sudah sesuai dengan keinginan maka tekan tombol next sedangkan apabila belum memenuhi keinginan bisa dikembalikan dengan menggunakan tombol back.



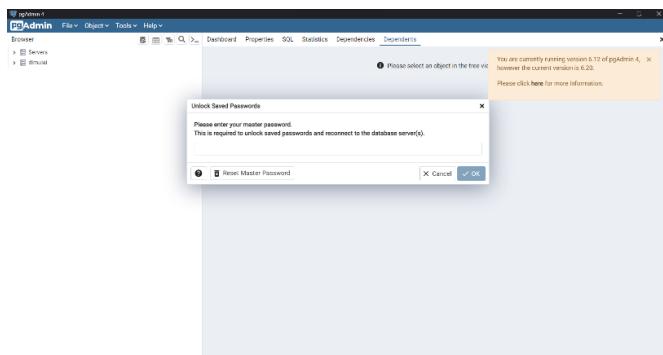
Gambar 23 Proses Instalasi Postgres

Pada Gambar 23 menunjukkan proses instalasi sudah dimulai. Tunggu hingga berakhir sehingga menampilkan tampilan dengan tombol finish untuk tahap akhir.



Gambar 24 pgAdmin pada Start Menu

Selanjutnya untuk memastikan aplikasi terpasang dengan benar, pergi ke Start Menu lalu ketikkan pgAdmin untuk membuka interface dari Postgres itu sendiri.

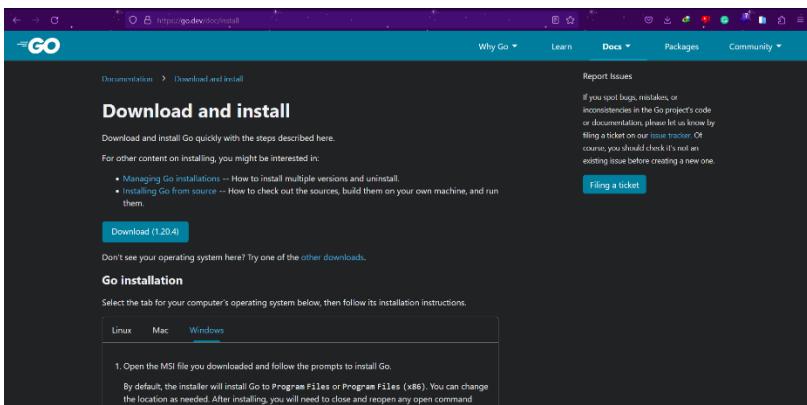


Gambar 25 Tampilan Postgres

Apabila instalasi berjalan tanpa kendala, maka tampilan dari postgres akan terlihat seperti pada Gambar 25. Disini pada saat ingin memasuki aplikasi akan diperintahkan terlebih dahulu untuk menginputkan password yang sudah dibuat pada saat proses instalasi sebelumnya.

#### D. Instalasi dan Setup Golang

Pada tahap ini akan dilakukan instalasi *runtime* untuk Golang agar dapat menjalankan program-program dengan Bahasa Pemrograman Go. Berikut adalah tahapannya.



Gambar 26 Halaman Official Golang

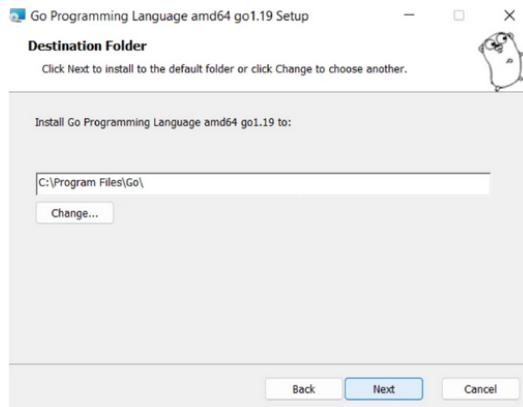
Gambar 26 menunjukkan tampilan pada halaman web resmi Golang yang dapat diakses pada url <https://go.dev/doc/install> pada perangkat masing-masing. Tekan tombol “Download” untuk mengunduh file instalasi golang.



Gambar 27 Set up Golang

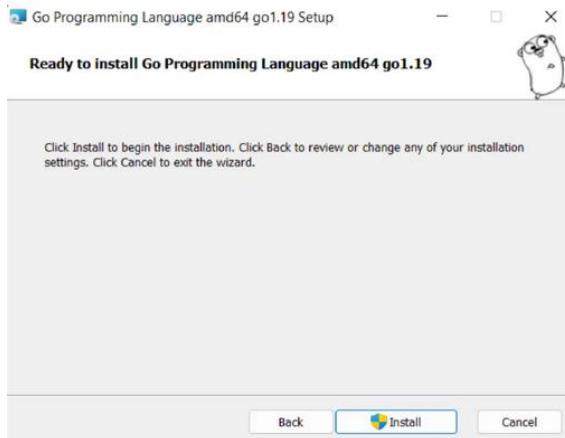
Selanjutnya apabila file instalasi selesai diunduh, tekan pada ikon file dan tampilan akan terlihat seperti pada Gambar 27. Klik next untuk melanjutkan ke

tahap selanjutnya.



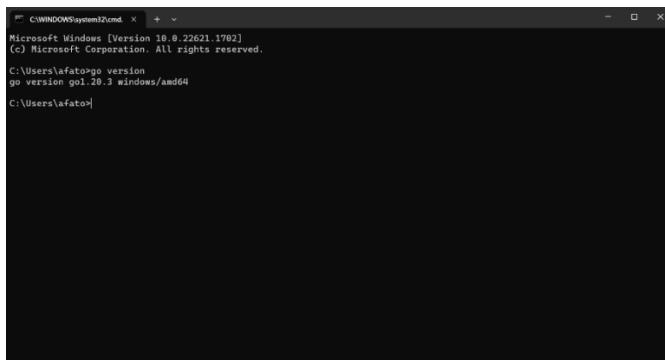
Gambar 28 Pemilihan Direktori Golang

Proses selanjutnya adalah memilih direktori untuk Golang. Pilihlah sesuai dengan direktori yang diinginkan dengan menekan tombol change.



Gambar 29 Tahap Terakhir Instalasi Golang

Pada Gambar 29 merupakan tahap terakhir dari instalasi golang. Klik pada tombol install untuk mulai memasang Golang pada perangkat, apabila muncul Pop up administrator klik yes untuk memulai.



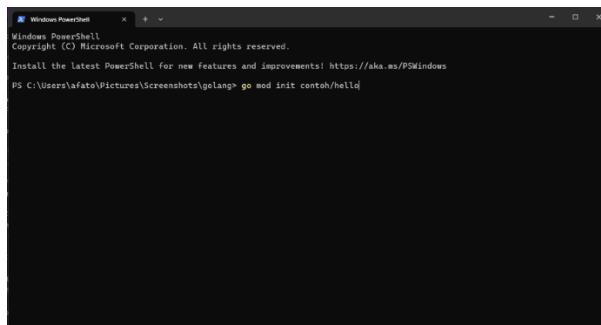
```
C:\WINDOWS\system32\cmd x + v
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\afato>go version
go version go1.20.3 windows/amd64

C:\Users\afato>
```

Gambar 30 Pengecekan Instalasi Golang

Setelah semua tahapan instalasi selesai, buka CMD untuk mengecek Golang sudah terpasang dengan baik dengan memasukan command “go version”. Apabila tampilan sudah sesuai dengan Gambar 30 maka instalasi sudah selesai.



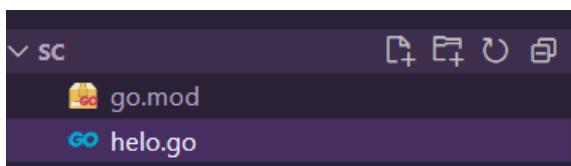
```
Windows PowerShell x + v
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\afato\Pictures\Screenshots\golang> go mod init contoh/hello
```

Gambar 31 Command untuk Membuat go init

Pada tahap selanjutnya adalah mempersiapkan project go untuk membuat kode-kode dari program yang akan dibuat. Pertama yang harus dilakukan adalah membuka cmd lalu mengetikan perintah “go mod init <nama\_package/nama>” sesuai dengan Gambar 31.



Gambar 32 Struktur Awal File Go

Tambahkan file dengan nama “helo.go” lalu copy code seperti pada Kode 1 dibawah yang berfungsi untuk melakukan print tipe *string* pada terminal.

```
package main

import "fmt"

func hello() {
    fmt.Println("Hello, World!")
}
```

Kode 1 Percobaan Hello World

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\afato\Pictures\Screenshots\golang\sc> go run hello.go
Hello, World!
PS C:\Users\afato\Pictures\Screenshots\golang\sc>
```

Gambar 33 Terminal Go

Terakhir adalah melakukan percobaan dengan cara membuka terminal pada Visual Studio Code lalu, mengetikan command “`go run <<nama_file.go>>`” dan apabila output sudah sesuai dengan Gambar 33 maka, Golang sudah siap digunakan.

## E. Instalasi Gin

Gin adalah kerangka kerja yang dibutuhkan untuk pembuatan API pada project ini, maka dibutuhkan instalasi Gin terlebih dahulu. Pada proses ini dibutuhkan terlebih dahulu runtime dari Golang yang sudah dijelaskan pada tahap sebelumnya. Berikut adalah cara untuk melakukan instalasi Gin dalam project Golang yang sudah dibuat.

```
PS C:\Users\afato\Pictures\Screenshots\golang\sc> go get -u github.com/gin-gonic/gin
go: added github.com/bytedance/sonic v1.8.8
go: added github.com/chenhuoyi/base64x v0.0.0-20221115062448-fe3a3bad311
go: added github.com/gin-contrib/sse v0.1.0
go: added github.com/gin-gonic/gin v1.9.0
go: added github.com/go-playground/locales v0.14.1
go: added github.com/go-playground/universal-translator v0.18.1
go: added github.com/goccy/go-json v0.10.2
go: added github.com/julienschmidt/httprouter v1.0.12
go: added github.com/klauspost/constraints v0.2.4
go: added github.com/leedidio/go-ern v1.2.4
```

Gambar 34 Command Instalasi Gin

Tahap pertama adalah membuka terminal pada Visual Studio Code, lalu ketikkan “`go get -u github.com/gin-gonic/gin`” dan tekan enter. Tunggu hingga proses selesai.

```
package main

import "github.com/gin-gonic/gin"

func main() {
    r := gin.Default()
    r.GET("/ping", func(c *gin.Context) {
        c.JSON(200, gin.H{
            "message": "pong",
        })
    })
    r.Run()
}
```

Kode 2 Testing Ping

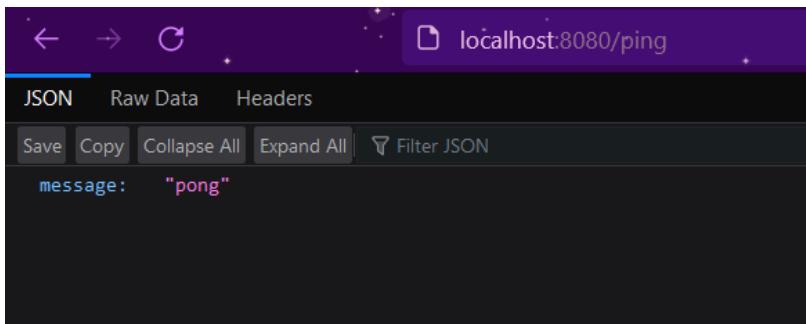
```
PS C:\Users\afato\Pictures\Screenshots\golang\sc> go run hello.go
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
  using env: export GIN_MODE=release
  using code: gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET      /ping           -> main.main.func1 (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Environment variable PORT is undefined. Using port :8080 by default
[GIN-debug] Listening and serving HTTP on :8080
```

Gambar 35 Running Gin

Ubah kode pada “hello.go” seperti pada Kode 2, lalu running pada terminal seperti pada Gambar 35 dengan perintah “`go run hello.go`”. Pada Kode 2 terdapat import “`github.com/gin-gonic/gin`” untuk mengambil komponen dari Gin, lalu dalam function main merupakan rute untuk menambahkan respon

yang dibuat dalam alias yaitu “r” yang berisi pengaturan gin Default, lalu direspon dengan HTTP response 200.



Gambar 36 Test Response Gin pada Browser

Tahap terakhir adalah mencoba membuka URL local yang disediakan pada terminal dalam browser. Apabila output sesuai dengan Gambar 36 yang merupakan output dari endpoint yang telah dibuat maka Gin sudah siap untuk digunakan.

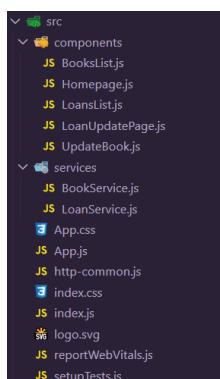
# BAB 3

## Tahapan Pembuatan Front-end

*Front-end* merupakan bagian dari aplikasi yang memuat sektor tampilan dari sebuah aplikasi. *Front-end* bisa juga disebut sebagai *client* atau penerima layanan dari *back-end* atau *server-side*, pada bagian ini kumpulan data-data akan ditampilkan berdasarkan perangkat yang digunakan (Liu & Gupta, 2019). Pada bagian ini akan diarahkan bagaimana cara kita untuk membuat bagian dari penampilan aplikasi agar data yang ditampilkan dapat mudah dimengerti dan mempermudah manajemen data melewati integrasi antara *server-side* dan *client*. Berikut adalah tahapan-tahapan yang dilewati untuk membuat *client* pada buku tutorial ini.

### A. Pembuatan Front-End Buku

#### 1. Menyiapkan Struktur Folder dan File



Gambar 37 Struktur Front-end

Hal yang harus dipersiapkan pertama kali adalah menyiapkan terlebih dahulu Folder dan File dalam project React yang sudah dibuat pada tahap sebelumnya. Hal ini dilakukan untuk mempermudah pada saat akan memasukan kode agar lebih terurut.

## 2. Menyiapkan *dependency*

```
npm install axios
npm install bootstrap
npm install react-router-dom@5.2.0
```

Kode 3 Install Dependency react

Sebelum memasuki pembuatan kode lakukan instalasi *dependency* seperti pada Kode 3 terlebih dahulu, agar dapat memenuhi kebutuhan komponen yang dibutuhkan dalam membuat aplikasi. Kegunaan dari *dependency* tersebut adalah yang pertama Axios digunakan untuk berkomunikasi dengan API berdasarkan URL yang diberikan, lalu bootstrap digunakan untuk mempercantik tampilan dari website, dan react-router-dom digunakan untuk memberikan rute terhadap aplikasi kepada modul-modul yang sudah dibuat agar dapat diakses.

## 3. Menyiapkan Koneksi API

```
import axios from "axios";

export default axios.create({
  baseURL: "sesuaikan_dengan_alamat_api",
  headers: {
    "Content-type": "application/json"
  }
});
```

Kode 4 Alamat API

Proses selanjutnya adalah menyiapkan file “http-common.js” seperti pada Kode 4. *Import* axios dibutuhkan karena akan berkomunikasi dengan API dengan mengambil baseURL dan header untuk menentukan tipe header dari output API.

#### 4. Menyiapkan Rute API

```
import http from "../http-common";

const getBuku = () => {
    return http.get("/books");
};

const getBukuId = (id) => {
    return http.get(`/books/${id}`);
};

const createBuku = (data) => {
    return http.post("/books", data);
};

const updateBuku = (id, data) => {
    return http.put(`/books/${id}`, data);
};

const removeBuku = (id) => {
    return http.delete(`/books/${id}`);
};

const removeAllBuku = () => {
    return http.delete(`/books`);
};

const findBukuByTitle = (title) => {
    return http.get(`/books?title=${title}`);
};
```

Kode 5 Menangkap Rute endpoint

Pada Kode 5 merupakan sekumpulan *function* yang digunakan untuk memberikan rute tambahan pada Base URL yang ada pada “http-common.js”. Beberapa rute digunakan untuk mengambil data dengan GET method, PUT method untuk mengubah data, POST method untuk menambahkan data, dan DELETE method untuk menghapus data.

```
const BookService = {
  getBuku,
  getBukuId,
  createBuku,
  updateBuku,
  removeBuku,
  removeAllBuku,
  findBukuByTitle,
};

export default BookService;
```

Kode 6 Pendefinisian Ulang

Pada Kode 6 *constraint* BookService digunakan untuk menampung semua route yang sudah dibuat dengan mendefinisikannya kembali.

## 5. Membuat Kode Untuk Tampilan Book List

```
import React, { useState, useEffect } from "react";
import BookService from "../services/BookService";
import { Link } from "react-router-dom";

const BooksList = () => {
  const [books, setBooks] = useState([]);
  const [currentBook, setCurrentBook] = useState(null);
  const [judul, setJudul] = useState("");
  const [penulis, setPenulis] = useState("");
  const [penerbit, setPenerbit] = useState("");
  const [tanggalRilis, setTanggalRilis] = useState("");

  useEffect(() => {
    retrieveBooks();
  }, []);
}
```

Kode 7 Pembuatan Kondisi dengan State

Pada Kode 7 digunakan untuk *import* kebutuhan komponen yang dibutuhkan untuk melakukan manajemen *state* pada halaman tertentu. *useEffect* digunakan untuk memberikan efek terhadap komponen tertentu jika mengalami aksi tertentu seperti pada saat mengalami klik atau kondisi apapun, dan *useState* untuk manajemen kondisi dalam halaman tertentu.

```

const retrieveBooks = () => {
  BookService.getAllBuku()
    .then((response) => {
      setBooks(response.data);
    })
    .catch((error) => {
      console.log(error);
    });
};

const saveBook = () => {
  const book = {
    Judul: judul,
    Penulis: penulis,
    Penerbit: penerbit,
    Tanggal_rilis: tanggalRilis,
  };

  BookService.createBuku(book)
    .then((response) => {
      console.log(response.data);
      retrieveBooks();
    })
    .catch((error) => {
      console.log(error);
    });
};

const updateBook = () => {
  const updatedBook = {
    id: currentBook.ID,
    Judul: judul,
    Penulis: penulis,
    Penerbit: penerbit,
    Tanggal_rilis: tanggalRilis,
  };

  BookService.updateBuku(currentBook.ID, updatedBook)
    .then((response) => {
      console.log(response.data);
      retrieveBooks();
      setCurrentBook(null);
      setJudul("");
      setPenulis("");
      setPenerbit("");
      setTanggalRilis("");
    })
    .catch((error) => {
      console.log(error);
    });
};

```

Kode 8 Pembuatan Constraint Buku

Pada Kolom 8 Merupakan sekumpulan *function* yang masing-masing digunakan untuk melakukan pengambilan atau manipulasi terhadap data. *Function* “retrieveBook” digunakan untuk mengambil semua data buku yang diambil oleh API, “saveBook” digunakan untuk menyimpan data buku dengan method POST, dan “updateBook” digunakan untuk mengubah data dari buku dengan mengambil ID dari buku tersebut terlebih dahulu dalam *state management*.

```
const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "judul") {
    setJudul(value);
  } else if (name === "penulis") {
    setPenulis(value);
  } else if (name === "penerbit") {
    setPenerbit(value);
  } else if (name === "tanggal_rilis") {
    setTanggalRilis(value);
  }
};
```

Kode 9 Manajemen Kondisi Form

Pada Kode 9 digunakan untuk manajemen kondisi pada saat form tertentu diubah.

```
return (
  <div className="container">
    <div className="row">
      <div className="col-md-6">
        <div className="mb-3">
          <label className="form-label">Judul:</label>
          <input
            type="text"
            className="form-control"
            name="judul"
            value={judul}
            onChange={handleInputChange}
          />
        </div>
      </div>
    </div>
  </div>
```

```

<div className="mb-3">
    <label className="form-label">Penulis:</label>
    <input
        type="text"
        className="form-control"
        name="penulis"
        value={penulis}
        onChange={handleInputChange}
    />
</div>

<div className="mb-3">
    <label className="form-label">Penerbit:</label>
    <input
        type="text"
        className="form-control"
        name="penerbit"
        value={penerbit}
        onChange={handleInputChange}
    />
</div>

<div className="mb-3">
    <label className="form-label">Tanggal Rilis:</label>
    <input
        type="date"
        className="form-control"
        name="tanggal_rilis"
        value={tanggalRilis}
        onChange={handleInputChange}
    />
</div>

<div className="mb-3">
    <button className="btn btn-primary me-2" onClick={saveBook}>
        Create
    </button>
</div>
</div>

```

Kode 10 Struktur HTML form Buku

Pada Kode 10 digunakan untuk membuat tampilan form dari beberapa *field* berdasarkan model dari API. Manajemen *state* digunakan disini dengan melakukan pemasangan function dengan mendefinisikan properti pada *value* dan kondisi *onChange*

```

<div className="row">
  <div className="col-md-12">
    <table className="table">
      <thead>
        <tr>
          <th>Judul</th>
          <th>Penulis</th>
          <th>Penerbit</th>
          <th>Tanggal Rilis</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {books.map((book) => (
          <tr key={book.ID}>
            <td>{book.Judul}</td>
            <td>{book.Penulis}</td>
            <td>{book.Penerbit}</td>
            <td>{book.Tanggal_rilis}</td>
            <td>
              <button
                className="btn btn-danger btn-sm me-2"
                onClick={() => deleteBook(book.ID)}
              >
                Delete
              </button>
              &nbsp;&nbsp;
              <Link
                to={`/books/update/${book.ID}`}
                className="btn btn-primary btn-sm"
              >
                Update
              </Link>
            </td>
          </tr>
        )))
      </tbody>
    </table>
  </div>
</div>

      </div>
    );
};

export default BooksList;

```

Kode 11 Struktur HTML untuk List Buku

Pada kolom 11 merupakan kode untuk membuat list untuk buku, agar ditampilkan dalam bentuk tabel dan memberikan assign terhadap *function* dan parameter dari buku menggunakan *function* “books” dijadikan alias lalu

dipanggil parameternya pada masing-masing kolom.

## 6. Set Up Index dan App JS

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter } from "react-router-dom";

import App from "./App";

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById("root")
);

reportWebVitals();
```

Kode 12 Pengaturan BrowserRouter pada App.js

Setelah pembuatan modul untuk tampilan dan integrasi sudah selesai, selanjutnya melakukan pengaturan pada Index JS seperti pada Kode 12. Tambahkan *import* untuk “BrowserRouter” dan sisipkan pada render dan disimpan dengan “<App />” didalamnya.

```
import React from "react";
import { Switch, Route, Link } from "react-router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import "@fortawesome/fontawesome-free/css/all.css";
import "@fortawesome/fontawesome-free/js/all.js";
import "./App.css";

import BooksList from "./components/BooksList";
```

Kode 13 Import Komponen React Router

Lakukan import terlebih dahulu seperti pada Kode 13 untuk memanggil modul BookList dan memenuhi kebutuhan komponen agar dapat menjalankan modul tertentu.

```

import UpdateBook from "./components/UpdateBook";
import LoansList from "./components/LoansList";
import LoanUpdatePage from "./components/LoanUpdatePage";
import Homepage from "./components/Homepage";

function App() {
  return (
    <div>
      <nav className="navbar navbar-expand navbar-dark bg-dark">
        <a href="/books" className="navbar-brand">
          Test
        </a>
        <div className="navbar-nav mr-auto">
          <li className="nav-item">
            <Link to={"/homepage"} className="nav-link">
              Home
            </Link>
          </li>
          <li className="nav-item">
            <Link to={"/books"} className="nav-link">
              Books
            </Link>
          </li>
          <li className="nav-item">
            <Link to={"/Loans"} className="nav-link">
              Loans
            </Link>
          </li>
        </div>
      </nav>

      <div className="container mt-3">
        <Switch>
          <Route exact path={["/", "/books"]} component={BooksList} />
          <Route exact path="/loans" component={LoansList} />
          <Route exact path="/books/update/:id" component={UpdateBook} />
          <Route exact path="/loans/update/:id" component={LoanUpdatePage}>
        </Switch>
        <div>
          <div>
            <Route exact path="/add" component={AddBooks} />
            <Route path="/books/:id" component={Book} />
            <Route path="/homepage" component={Homepage} />
          </div>
        </div>
      </div>
    );
}

export default App;

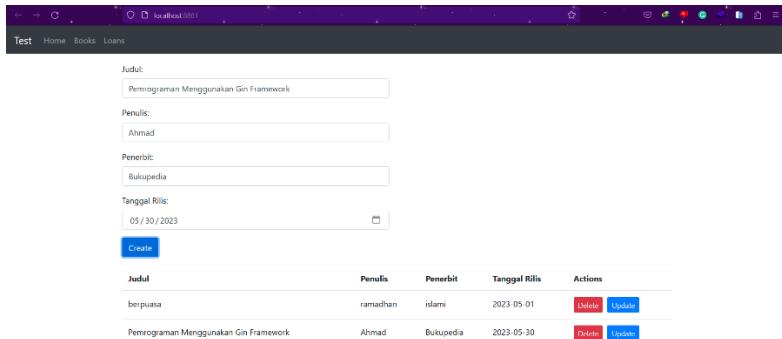
```

Kode 14 Import Modul Komponen dan Pemberian Rute

Lakukan *import* untuk modul komponen dan buatlah rute menggunakan tag “react-router-dom” yaitu “Route path” dan masukkan URL pada setiap route,

setelah pembuatan rute buat navbar agar dapat melakukan navigasi dengan mudah dalam aplikasi dengan menggunakan class navbar dan nav-item untuk kumpulan link dari rute yang sudah disiapkan.

## 7. Mencoba Pada Browser



Gambar 38 Tampilan BooksList

Setelah kode tampilan dibuat buka URL Localhost:8081, maka tampilan akan terlihat seperti pada Gambar 38 yaitu halaman manajemen buku yang dibuat dalam file “BooksList.js”.

## 8. Pembuatan Halaman UpdateBook

```
import React, { useState, useEffect } from "react";
import { useParams } from "react-router-dom";
import BookService from "../services/BookService";

const UpdateBook = () => {
  const { id } = useParams();
  const [judul, setJudul] = useState("");
  const [penulis, setPenulis] = useState("");
  const [penerbit, setPenerbit] = useState("");
  const [tanggalRilis, setTanggalRilis] = useState("");
```

Kode 15 Import Dependency dan Pembuatan State

Pada Kode 15 merupakan *import* untuk *dependency* yang dibutuhkan untuk menjalankan modul “UpdateBook.js”, dan penyimpanan state untuk tiap field.

```

const retrieveBook = () => {
  BookService.get(id)
    .then((response) => {
      const { Judul, Penulis, Penerbit, Tanggal_rilis } = response.data;
      setJudul(Judul);
      setPenulis(Penulis);
      setPenerbit(Penerbit);
      setTanggalRilis(Tanggal_rilis);
    })
    .catch((error) => {
      console.log(error);
    });
};

const updateBook = () => {
  const updatedBook = {
    id: id,
    Judul: judul,
    Penulis: penulis,
    Penerbit: penerbit,
    Tanggal_rilis: tanggalRilis,
  };

  BookService.update(id, updatedBook)
    .then((response) => {
      console.log(response.data);
      // Redirect to the book List page after successful update
      window.location.href = "/books";
    })
    .catch((error) => {
      console.log(error);
    });
};

const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "judul") {
    setJudul(value);
  } else if (name === "penulis") {
    setPenulis(value);
  } else if (name === "penerbit") {
    setPenerbit(value);
  } else if (name === "tanggal_rilis") {
    setTanggalRilis(value);
  }
};

```

Kode 16 Pembuatan Function State

Selanjutnya pada Kode 16 merupakan pembuatan *function* untuk mendeklarasikan state dan menampung buku berdasarkan ID menggunakan *function* “retrieveBook”. Lalu pembuatan *function* untuk menampung *field* pada buku, dan terakhir ada *handler* untuk perubahan pada input yang didefinisikan

dengan “handleInputChange”.

```
return (
  <div className="container">
    <h1>Update Book</h1>
    <div className="mb-3">
      <label className="form-label">Judul:</label>
      <input
        type="text"
        className="form-control"
        name="judul"
        value={judul}
        onChange={handleInputChange}
      />
    </div>

    <div className="mb-3">
      <label className="form-label">Penulis:</label>
      <input
        type="text"
        className="form-control"
        name="penulis"
        value={penulis}
        onChange={handleInputChange}
      />
    </div>

    <div className="mb-3">
      <label className="form-label">Penerbit:</label>
      <input
        type="text"
        className="form-control"
        name="penerbit"
        value={penerbit}
        onChange={handleInputChange}
      />
    </div>
```

```

<div className="mb-3">
  <label className="form-label">Tanggal Rilis:</label>
  <input
    type="text"
    className="form-control"
    name="tanggal_rilis"
    value={tanggalRilis}
    onChange={handleInputChange}
  />
</div>

<div>
  <button className="btn btn-primary" onClick={updateBook}>
    Update
  </button>
</div>
</div>
);

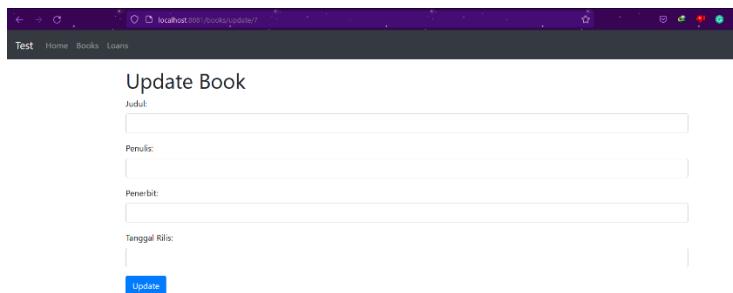
};

export default UpdateBook;

```

Kode 17 Code Interface UpdateBook

Pada Kode 17 merupakan kode untuk tampilan Update Book dengan menggunakan bootstrap dan pendeklarasian HandlerInputChange pada tiap kolom dan pemberian function dalam kondisi “onClick” maka akan melakukan update terhadap buku.



Gambar 39 Tampilan Update Book

Pada Gambar 39 merupakan tampilan dari UpdateBook, yang berfungsi untuk mengubah data buku berdasarkan ID yang diambil dari URL

## B. Pembuatan Front-End Peminjaman Buku

### 1. Import dan Pembuatan State

```
import React, { useState, useEffect } from "react";
import LoanService from "../services/LoanService";
import { Link } from "react-router-dom";

const LoansList = () => {
  const [loans, setLoans] = useState([]);
  const [currentLoan, setCurrentLoan] = useState(null);
  const [userID, setUserID] = useState("");
  const [bookID, setBookID] = useState("");
  const [tanggalPinjam, setTanggalPinjam] = useState("");
  const [dikembalikan, setDikembalikan] = useState(false);

  useEffect(() => {
    retrieveLoans();
  }, []);

  const retrieveLoans = () => {
    LoanService.getAll()
      .then((response) => {
        setLoans(response.data);
      })
      .catch((error) => {
        console.log(error);
      });
};

}
```

Kode 18 Import Untuk LoansList

Import *dependency* terlebih dahulu seperti pada Kode 18 agar dapat menjalankan program. Lalu buat State dengan menggunakan function useState untuk menyimpan kondisi dan useEffect untuk memberikan efek dalam memberikan aksi dan function retrieveLoans digunakan untuk mengambil data-data pinjaman.

```

const saveLoan = () => {
  const loan = {
    UserID: userID,
    BooksID: bookID,
    Tanggal_pinjam: tanggalPinjam,
    Dikembalikan: dikembalikan
  };

  LoanService.create(loan)
    .then((response) => {
      console.log(response.data);
      retrieveLoans();
    })
    .catch((error) => {
      console.log(error);
    });
};

const deleteLoan = (id) => {
  LoanService.remove(id)
    .then((response) => {
      console.log(response.data);
      retrieveLoans();
    })
    .catch((error) => {
      console.log(error);
    });
};

const removeAllLoans = () => {
  LoanService.removeAll()
    .then((response) => {
      console.log(response.data);
      retrieveLoans();
    })
    .catch((error) => {
      console.log(error);
    });
};

const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "userID") {
    setUserID(parseInt(value));
  } else if (name === "bookID") {
    setBookID(parseInt(value));
  } else if (name === "tanggalPinjam") {
    setTanggalPinjam(value);
  } else if (name === "dikembalikan") {
    setDikembalikan(event.target.checked);
  }
};

```

Kode 19 Kumpulan Function

## 2. Membuat Tampilan dan Assign Function

Selanjutnya adalah menambahkan *function* seperti pada Kode 19, *function* tersebut berfungsi untuk menyimpan buku pada saveLoan dengan mengambil input dari form yang dideklarasikan pada tahapan selanjutnya. Lalu Remove All Loans digunakan untuk menghapus semua data pinjaman buku dan handleInputChange berfungsi untuk merekam kondisi apabila ada perubahan dalam sebuah form.

```
return (
  <div className="container">
    <div className="row">
      <div className="col-md-6">
        <div className="mb-3">
          <label className="form-label">User ID:</label>
          <input
            type="number"
            className="form-control"
            name="userID"
            value={userID}
            onChange={handleInputChange}
          />
        </div>

        <div className="mb-3">
          <label className="form-label">Book ID:</label>
          <input
            type="number"
            className="form-control"
            name="bookID"
            value={bookID}
            onChange={handleInputChange}
          />
        </div>
        <div className="mb-3">
          <label className="form-label">Tanggal Pinjam:</label>
          <input
            type="date"
            className="form-control"
            name="tanggalPinjam"
            value={tanggalPinjam}
            onChange={handleInputChange}
          />
        </div>
      </div>
    </div>
  </div>
```

```

<div className="mb-3 form-check">
    <input
        type="checkbox"
        className="form-check-input"
        name="dikembalikan"
        checked={dikembalikan}
        onChange={handleInputChange}>
    />
    <label className="form-check-label">Dikembalikan</label>
</div>

<div className="mb-3">
    <button className="btn btn-primary me-2" onClick={saveLoan}>
        Create
    </button>
</div>
</div>
</div>

<div className="row">
    <div className="col-md-12">
        <table className="table">
            <thead>
                <tr>
                    <th>User ID</th>
                    <th>Book ID</th>
                    <th>Tanggal Pinjam</th>
                    <th>Dikembalikan</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                {loans.map((loan) => (
                    <tr key={loan.ID}>
                        <td>{loan.UserID}</td>
                        <td>{loan.BooksID}</td>
                        <td>{loan.Tanggal_pinjam}</td>
                        <td>{loan.Dikembalikan ? "Yes" : "No"}</td>
                        <td>
                            <button
                                className="btn btn-danger btn-sm me-2"
                                onClick={() => deleteLoan(loan.ID)}>
                                Delete
                            </button>
                            &nbsp;&nbsp;
                            <Link
                                to={`/loans/update/${loan.ID}`}
                                className="btn btn-primary btn-sm">
                                Update
                            </Link>
                        </td>
                    </tr>
                ))}
            </tbody>
        </table>
    </div>
</div>
);

};

export default LoansList;

```

Kode 20 Struktur HTML untuk Tampilan Tambah Pinjaman

Terakhir tambahkan bagian struktur HTML untuk menampilkan data-data dari pinjaman. Data akan ditampilkan dalam bentuk tabel dan kondisi akan disimpan dalam masing-masing label, untuk tampilan dari Update pinjaman akan memiliki halaman tersendiri yang akan dijelaskan dalam tahap selanjutnya.

```
import React, { useState, useEffect } from "react";
import LoanService from "../services/LoanService";

const LoanUpdatePage = (props) => {
  const [loan, setLoan] = useState(null);
  const [userID, setUserID] = useState("");
  const [bookID, setBookID] = useState("");
  const [tanggalPinjam, setTanggalPinjam] = useState("");
  const [dikembalikan, setDikembalikan] = useState(false);

  useEffect(() => {
    retrieveLoan();
  }, []);

  const retrieveLoan = () => {
    const loanId = props.match.params.id;
    LoanService.get(loanId)
      .then((response) => {
        setLoan(response.data);
        setUserID(response.data.UserID);
        setBookID(response.data.BooksID);
        setTanggalPinjam(response.data.Tanggal_pinjam);
        setDikembalikan(response.data.Dikembalikan);
      })
      .catch((error) => {
        console.log(error);
      });
  };

  const updateLoan = () => {
    const updatedLoan = {
      id: loan.id,
      UserID: userID,
      BooksID: bookID,
      Tanggal_pinjam: tanggalPinjam,
      Dikembalikan: dikembalikan,
    };

    LoanService.update(loan.id, updatedLoan)
      .then((response) => {
        console.log(response.data);
        props.history.push("/loans");
      })
      .catch((error) => {
        console.log(error);
      });
  };
}
```

Kode 21 Pendeklarasian Function Tambah Pinjaman

Tahap selanjutnya adalah pembuatan tampilan untuk UpdateLoans yang diawali dengan import dan pembuatan function untuk menyimpan state dari halaman dan pendeklarasian *field* pada tiap *function*.

```
const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "userID") {
    setUserID(value);
  } else if (name === "bookID") {
    setBookID(value);
  } else if (name === "tanggalPinjam") {
    setTanggalPinjam(value);
  } else if (name === "dikembalikan") {
    setDikembalikan(event.target.checked);
  }
};

return (
  <div className="container">
    <h1>Update Loan</h1>
    {loan ? (
      <div>
        <div className="mb-3">
          <label className="form-label">User ID:</label>
          <input
            type="text"
            className="form-control"
            name="userID"
            value={userID}
            onChange={handleInputChange}
          />
        </div>
        <div className="mb-3">
          <label className="form-label">Book ID:</label>
          <input
            type="text"
            className="form-control"
            name="bookID"
            value={bookID}
            onChange={handleInputChange}
          />
        </div>
        <div className="mb-3">
          <label className="form-label">Tanggal Pinjam:</label>
          <input
            type="date"
            className="form-control"
            name="tanggalPinjam"
            value={tanggalPinjam}
            onChange={handleInputChange}
          />
        </div>
    ) : null}
  </div>
)
```

```

<div className="mb-3">
    <label className="form-check-label">Dikembalikan:</label>
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    <input
        type="checkbox"
        className="form-check-input"
        name="dikembalikan"
        checked={dikembalikan}
        onChange={handleInputChange}
    />
</div>

<div>
    <button className="btn btn-primary" onClick={updateLoan}>
        Update
    </button>
</div>
) : (
    <div>Loading loan details...</div>
)
</div>
);
};

export default LoanUpdatePage;

```

Kode 22 Pembuatan Interface UpdateLoan

Tahap terakhir dalam pembuatan *Interface* seperti pada Kode 22 untuk halaman UpdateLoan data akan ditampilkan sama seperti pada halaman Books, namun pada halaman Loans yang ditampilkan adalah data pinjaman berdasarkan *field* yang tersedia pada API. Pembuatan sisa *function* yaitu untuk menampung aksi yang dibuat oleh pengguna dibuat disini dalam *function* handlerInputChange yang berguna untuk merekam perubahan pada form agar data dapat diinputkan kedalam database. Lalu *function* akan di-*assign* kedalam tag HTML yang terdapat pada setiap struktur form masing-masing.

User ID	Book ID	Tanggal Pinjam	Dikembalikan	Actions
1	7	2023-05-01	Yes	<button>Delete</button> <button>Update</button>

Gambar 40 Tampilan Tambah Pinjaman

Pada Gambar 40 merupakan tampilan dari halaman peminjaman buku yang dapat diakses dengan menekan tombol Loans pada Navbar atau pada URL “localhost:8081/loans”. Field UserID dan BookID akan ditampilkan dalam bentuk tabel dan dalam tabel akan ada tombol aksi yang digunakan untuk menghapus data pinjaman buku.

Gambar 41 Update Loan

Gambar 41 adalah tampilan dari UpdateLoan yang digunakan untuk mengubah data pinjaman. Halaman ini dapat diakses dengan menekan tombol “update” dalam halaman list pinjaman dan pinjaman akan ditampilkan berdasarkan id yang dimuat dalam URL.

# BAB 4

## Tahapan Pembuatan Back-end

Pada BAB ini akan membahas tentang perancangan bagian *Back-end* pada aplikasi ini. Hal ini bertujuan agar aplikasi mempunyai alur untuk bagian logikanya dengan cara membuat API. API bisa menghubungkan sistem internal dengan mudah, berkomunikasi antar platform, membuat API akan mempermudah pengembangan aplikasi karena akan melancarkan proses automasi dan pengembangan secara cepat (Iyengar, Khanna, Ramadath, & Stephens, 2017). Berikut adalah tahapan-tahapan dalam membuat API untuk E-Library.

### A. Perancangan Endpoint Autentifikasi User

#### 1. Koneksi Database

```
"gorm.io/driver/postgres"  
"gorm.io/gorm"
```

Gambar 42 Installasi Gorm

Hal yang pertama kali dilakukan adalah melakukan koneksi database dengan memasang *dependency* terlebih dahulu. Pemasangan *dependency* dapat dilakukan dengan cara mengikuti intruksi seperti pada Gambar 42 pada terminal di VSCode yang terpasang pada project, hasil dari pemasangan *dependency* bisa dilihat pada file yang bernama “go.mod”.

```
42 |   gorm.io/driver/postgres v1.5.2 // indirect  
43 |   gorm.io/gorm v1.25.1 // indirect  
44 )  
45
```

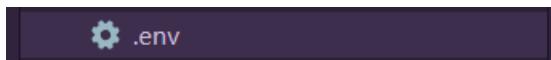
Gambar 43 Hasil Instalasi Gorm

Pada Gambar 43 merupakan keterangan Gorm sudah terpasang pada

*dependency* list dalam file yang bernama “go.mod”. Gorm merupakan *library* yang mengimplementasikan teknik dan menyediakan layer yang berbasis *object-oriented* diantara *database* dan bahasa pemrograman OOP Go (Tomilov, 2020).

## 2. Membuat Initializer Environment

Tahapan selanjutnya adalah membuat initializer yang berfungsi untuk membuat koneksi antara API dan database. Hal tersebut dapat dilakukan dengan cara berikut.



Gambar 44 File .env

Buat file baru yang bernama .env yang digunakan untuk menampung *Environment Variable*. Hal tersebut berfungsi untuk menampung nilai yang berubah-ubah atau dinamis alamat dari database, JWT *Secret Key*, dan port API untuk dijalankan.

```
PORt=3000  
  
SECRET=asd112j3ihshu192  
  
DB="host=localhost user=postgres password=ahmad dbname=elibrary_app  
port=5432 sslmode=disable"
```

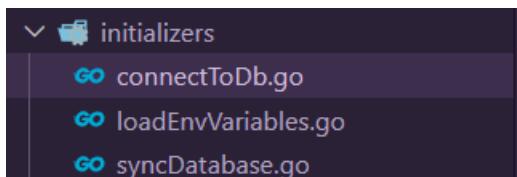
Kode 23 Pengaturan .env

Lalu masukkan *source code* yang ada pada Kode 23 yang berisi PORT, *Secret Key*, dan DB *Address*. Isi dari port dapat disesuaikan dengan pengaturan pada device Anda.

## 3. Membuat File Initializer

Proses selanjutnya adalah membuat file dari *initializer* itu sendiri. Disini initializer terdapat tiga file seperti pada Gambar 40 yaitu “connectToDB.go”, “loadEnvVariables.go”, dan “syncDatabase.go”. File tersebut masing-masing mempunyai kegunaannya yaitu “connectToDB” berfungsi untuk membuat koneksi ke database, “loadEnvVariables” digunakan untuk memuat file .env

supaya bisa terbaca oleh program, dan terakhir “syncDatabase” digunakan untuk migrasi model ke database agar tabel bisa dibuat secara otomatis.



Gambar 45 File Initializers

Selanjutnya adalah memasukkan *source code* dari setiap file dengan mengikuti kode berikut.

```
package initializers

import (
    "os"

    "gorm.io/driver/postgres"
    "gorm.io/gorm"
)

var DB *gorm.DB
func ConnectToDb() {
    var err error
    dsn := os.Getenv("DB")
    DB, err = gorm.Open(postgres.Open(dsn), &gorm.Config{})

    if err != nil {
        panic("failed to connect to database!")
    }
}
```

Kode 24 ConnectToDb Initializer

Pada kode 24 adalah source code untuk melakukan koneksi ke database. Pertama akan dilakukan import dahulu library untuk gorm dan gorm/postgres yang berfungsi untuk melakukan komunikasi antar database lalu ada function untuk menampung method dan variabel yang berfungsi untuk memanggil .env agar alamat dapat terdeteksi oleh gorm.

```
package initializers

import (
    "log"

    "github.com/joho/godotenv"
)

func LoadEnvVariables() {
    err := godotenv.Load()

    if err != nil {
        log.Fatal("error loading .env file")
    }
}
```

Kode 25 Source Code LoadEnvVariables

Pada Kode 25 merupakan code yang digunakan untuk mencari variabel yang terkandung dalam file .env. Namun sebelum melakukan akses terhadap variabel kita harus melakukan instalasi terlebih dahulu untuk mendeteksi file .env yang sudah dibuat dengan cara mengetikkan “go install github.com/joho/godotenv”. Apabila sudah melakukan instalasi maka pembuatan function bisa dilakukan dan library “godotenv” akan dipanggil dalam function tersebut.

#### 4. Pembuatan Model

Apabila struktur dari initializer sudah dibuat, selanjutnya adalah membuat model dari User. Model ini akan digunakan untuk membuat tabel secara otomatis menggunakan library Gorm dari golang. Berikut adalah cara membuat model tersebut.

Gambar 46 File User Model

Buat folder yang bernama Models lalu buat file dengan nama “userModel.go” seperti pada Gambar 46.

```
package models

import "gorm.io/gorm"

type User struct {
    gorm.Model
    Email    string `gorm:"unique"`
    Password string
    Name     string
}
```

Kode 26 Model User

Pada Kode 26 merupakan struktur dari model dari User, pertama akan dilakukan terlebih dahulu *import library* dari gorm yang digunakan membuat tabel. Selanjutnya ada properti dari model itu sendiri yang diberi nama “User” yang berisi “gorm.Model” yang berguna untuk mencetak kolom id, date created, dan deleted secara otomatis oleh *library* gorm. Lalu ada Email, Password, dan Name yang berfungsi untuk mencetak kolom pada database.

```
package initializers

import "elib_v2/models"

func SyncDatabase() {
    DB.AutoMigrate(&models.User{})
}
```

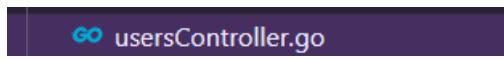
Kode 27 Source SyncDatabase

Kembali ke folder initializers, disini akan dibuat file yang bernama “syncDatabase.go” yang digunakan untuk melakukan *automigrate*. *Automigrate* berfungsi untuk membuat tabel secara otomatis kedalam RDBMS berdasarkan

model yang telah dibuat. Model User akan dipanggil dengan cara melakukan *import* terlebih dahulu *package* models, lalu model user akan dipanggil dalam function yang bernama *SyncDatabase* seperti pada Kode 27.

## 5. Pembuatan Controller User

Setelah tahap pembuatan Initializer dan Model sudah dibuat maka tahapan selanjutnya adalah membuat controller yang berfungsi untuk membuat logika untuk endpoint yang akan dipanggil nantinya.



usersController.go

Gambar 47 File usersController

Buatlah folder yang bernama models, lalu buat file yang bernama “usersController”.

```
package controllers

import (
    "elib_v2/initializers"
    "elib_v2/models"
    "net/http"
    "os"
    "time"

    "github.com/gin-gonic/gin"
    "github.com/golang-jwt/jwt/v4"
    "golang.org/x/crypto/bcrypt"
)
```

Kode 28 Import Library

Setelah membuat file selanjutnya adalah mengimport *library* seperti pada Kode 28 yang akan digunakan untuk membuat function dari controller.

```

func Signup(c *gin.Context) {

    var body struct {
        Email      string
        Password   string
        Name       string
    }

    if c.Bind(&body) != nil {
        c.JSON(http.StatusBadRequest, gin.H{
            "error": "Failed to read body",
        })
    }

    return
}

```

*Kode 29 Function Signup Pembuatan Property*

Lalu membuat variabel **body** yang nantinya akan dipanggil untuk membuat method dari pembuatan user.

```

//hash password disini
hash, err := bcrypt.GenerateFromPassword([]byte(body.Password),
10)

if err != nil {
    c.JSON(http.StatusBadRequest, gin.H{
        "error": "failed to hash password",
    })
}

return
}

// membuat user
user := models.User{Email: body.Email, Password: string(hash),
Name: body.Name}
result := initializers.DB.Create(&user)

```

*Kode 30 Hashing dan penampungan body*

Selanjutnya adalah pembuatan variabel untuk melakukan enkripsi pada password menggunakan bcrypt seperti pada kode 10. Juga pembuatan variabel user untuk mendeklarasikan model dari user dan menampung *value* dari model user.

```
if result.Error != nil {
    c.JSON(http.StatusBadRequest, gin.H{
        "error": "Failed to create user",
    })
}

return
}

c.JSON(http.StatusOK, gin.H{})
```

Kode 31 Method error untuk signup

Pada Kode 31 adalah method untuk menunjukkan error apabila request yang dilakukan adalah salah. Apabila request dilakukan dengan benar maka akan direspon dengan respon StatusOK HTTP 200.

```
func Login(c *gin.Context) {
    var body struct {
        Email    string
        Password string
    }

    // failed read body
    if c.Bind(&body) != nil {
        c.JSON(http.StatusBadRequest, gin.H{
            "error": "Failed to read body",
        })
    }

    return
}
```

Kode 32 Function Login

Selanjutnya setelah membuat function Signup untuk melakukan registrasi, maka tahapan selanjutnya adalah membuat function login. Pertama adalah membuat variabel *constructor* seperti pada Kode 32. Selanjutnya membuat respon apabila ada error pada saat membaca body dengan menggunakan method if dan http StatusBadRequest.

```
var user models.User
initializers.DB.First(&user, "email = ?", body.Email)

if user.ID == 0 {
    c.JSON(http.StatusBadRequest, gin.H{
        "error": "invalid email or password",
    })
}

return
}
```

Kode 33 Comparator Email

Setelah membuat constructor selanjutnya adalah membuat comparator untuk kolom Email yang terdapat pada Kode 33. Variabel user akan memanggil kolom tertentu dengan memanggil initializer, apabila email salah akan direspon dengan output http *Bad Request*.

```
    err :=bcrypt.CompareHashAndPassword([]byte(user.Password),
[]byte(body.Password))

    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{
            "error": "invalid email or password",
        })
    }

    return
}

//generate token
token := jwt.NewWithClaims(jwt.SigningMethodHS256,
jwt.MapClaims{
    "foo": user.ID,
    "exp": time.Now().Add(time.Hour * 24 * 30).Unix(),
})
if err != nil {
    c.JSON(http.StatusBadRequest, gin.H{
        "error": "generate token gagal!",
    })
}

return
}
```

Kode 34 Komparasi password

Berikutnya adalah melakukan komparasi password yang telah melalui tahap enkripsi seperti pada Kode 34. Pada variabel “err” akan dipanggil kolom Password yang sudah dienkripsi dan akan dikomparasikan salah atau benarnya. Apabila password salah akan direspon dengan *Bad Request*, sedangkan apabila password benar token akan ter-*generate* dengan batas waktu tertentu, serta apabila proses *generate* token gagal akan direspon dengan *Bad request* dengan pesan “generate token gagal”.

```
c.SetSameSite(http.SameSiteLaxMode)
c.SetCookie("Authorization", tokenString, 3600*24*30, "", "", 
false, true)

c.JSON(http.StatusOK, gin.H{})}

}
```

Kode 35 Penyimpanan Token

Terakhir adalah tahap penyimpanan token pada Kode 35. Token akan dicetak dalam bentuk string yaitu tokenString dan akan kadaluarsa dalam waktu tertentu. Token akan disimpan dalam cookie pada browser atau pada cookie management aplikasi pengujian API seperti Postman.

## 6. Pembuatan Middleware

Middleware adalah perangkat yang terdapat dalam aplikasi yang mempunyai peran kunci yaitu mengintegrasikan data dengan berbagai perangkat, memungkinkan kita untuk berkomunikasi antar perangkat, serta mengatur keputusan apabila aplikasi tersebut membutuhkan perizinan (Cruz, Rodriguez, Al-Muhtadi, Korotaev, & Alberquerque, 2018). Berikut adalah tahapan yang akan dilewati pada saat membuat middleware.

```
package middleware

import (
    "elib_v2/initializers"
    "elib_v2/models"
    "fmt"
    "net/http"
    "os"
    "time"

    "github.com/gin-gonic/gin"
    "github.com/golang-jwt/jwt/v4"
)
```

Kode 36 Import Komponen Untuk Middleware

Pertama adalah lakukan import terlebih dahulu kebutuhan *library* untuk Middleware. Import ini mencakup *package* dan *library* yang terdiri dari initializers, model, gin, jwt, dll.

```

func RequireAuth(c *gin.Context) {
    //get cookie
    tokenString, err := c.Cookie("Authorization")

    if err != nil {
        c.AbortWithStatus(http.StatusUnauthorized)
    }

    //validasi
    token, err := jwt.Parse(tokenString, func(token *jwt.Token) (interface{}, error) {
        if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
            return nil, fmt.Errorf("Unexpected signing method: %v",
                token.Header["alg"])
        }

        // hmacSampleSecret is a []byte containing your secret, e.g.
        // []byte("my_secret_key")
        return []byte(os.Getenv("SECRET")), nil
    })

    if claims, ok := token.Claims.(jwt.MapClaims); ok && token.Valid {
        //cek waktu expire
        if float64(time.Now().Unix()) > claims["exp"].(float64) {
            c.AbortWithStatus(http.StatusUnauthorized)
        }

        //mencari user dengan token
        var user models.User
        initializers.DB.First(&user, claims["sub"])

        if user.ID == 0 {
            c.AbortWithStatus(http.StatusUnauthorized)
        }

        //memasangkan dengan request
        c.Set("user", user)
        //Lanjut
        c.Next()

    } else {
        c.AbortWithStatus(http.StatusUnauthorized)
    }

    c.Next()
}

```

Kode 37 Function RequireAuth

Function requireAuth digunakan untuk menampung method-method yang berhubungan dengan autentifikasi user. Pertama adalah variabel tokenString yang digunakan untuk mengambil *cookie* yang di *generate* oleh usersController. Apabila tidak ditemukan cookie maka proses tidak akan dilanjutkan dan akan

mengeluarkan output berupa HTTP Response yaitu StatusUnauthorized, sedangkan apabila token ditemukan di *cookie* maka akan dilanjutkan dengan melakukan set user sesuai dengan data dari tabel user tersebut.

## 7. Pembuatan URL Endpoint Autentifikasi

Proses terakhir dalam pembuatan *Endpoint* untuk autentifikasi pengguna adalah membuat URL dari *controller* yang sudah dibuat. Berikut adalah tahapannya.

```
package main

import (
    "elib_v2/controllers"
    "elib_v2/initializers"
    "elib_v2/middleware"

    "github.com/gin-gonic/gin"
)

func init() {
    initializers.LoadEnvVariables()
    initializers.ConnectToDb()
    initializers.SyncDatabase()
}

func main() {
    r := gin.Default()

    api := r.Group("/api")
    {
        //auth
        api.POST("/signup", controllers.Signup)
        api.POST("/login", controllers.Login)
    }
    r.Run()
}
```

Kode 38 main.go

Buka kembali file main.go yang berada dalam direktori utama API, lalu import semua dependency yang dibutuhkan seperti package-package yang sudah dibuat seperti pada tahap sebelumnya. Setelah mengimport *library* yang dibutuhkan selanjutnya adalah membuat function untuk menampung initializer. Terakhir adalah membuat function untuk rute URL dengan cara membuat variabel “r” untuk rute dan “api” untuk menyatukan api dalam satu rute URL

yaitu “/api”.

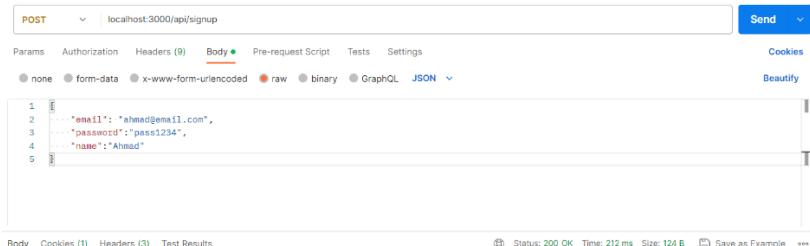
## 8. Uji Coba Endpoint User

Setelah semua tahapan dilalui selanjutnya adalah melakukan ujicoba terhadap Endpoint yang sudah dibuat. Pertama jalankan terlebih dahulu API dengan cara mengetikkan input “go run main.go” pada terminal dalam text editor atau terminal “cmd”.

```
2023/05/25 06:54:26 stdout: Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
2023/05/25 06:54:26 stdout: [GIN-debug] Environment variable PORT="3000"
2023/05/25 06:54:26 stdout: [GIN-debug] Listening and serving HTTP on :3000
2023/05/25 07:57:52 Running build command!
```

Gambar 48 Status Running API

Pada Gambar 43 merupakan keterangan apabila API sudah berjalan dengan menggunakan PORT 3000 dan siap untuk dipakai.



Gambar 49 Register User

Akses URL “localhost:3000/api/signup” untuk melakukan register seperti pada Gambar 44. Apabila pendaftaran berhasil maka respon akan dikembalikan dengan kode 200 Status OK.

The screenshot shows a POST request to 'localhost:3000/api/login'. The 'Body' tab is selected, showing a JSON payload with 'email' and 'password' fields. The response status is 200 OK, and a cookie named 'Authorization' is set.

Name	Value	Domain	Path	Expires	HttpOnly	Secure
Authorization	eyJhbGciOiJI... [REDACTED]	localhost	/	Sat, 24 Jun 2024 [REDACTED]	true	false

Gambar 50 Login

Setelah proses registrasi berhasil, selanjutnya adalah mencoba untuk Login menggunakan API dengan cara memasukan input dengan bentuk JSON pada Postman. Apabila User terdaftar maka API akan merespon dengan kode 200 dan token akan langsung disimpan dalam cookie.

## B. Perancangan Endpoint Buku

### 1. Membuat Model Buku



Gambar 51 File bookModel

Hal yang pertama harus dilakukan pada saat akan merancang endpoint untuk buku adalah membuat file nya terlebih dahulu. Buat file dengan nama bookModel.go seperti pada Gambar 46.

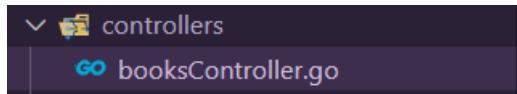
```
package models

type Books struct {
    ID          int `gorm:"primaryKey"`
    Judul       string
    Penulis     string
    Penerbit    string
    Tanggal_rilis string
}
```

Kode 39 Code Untuk Model Buku

Selanjutnya adalah memasukkan kode seperti pada Kode 39 pada file “bookModel.go”. Pada kode tersebut dilakukan terlebih dahulu pendeklarasian *package*, lalu dibuat *struct* untuk membuat struktur tabel sesuai dengan *field* dari buku yang akan diinputkan.

## 2. Membuat Controller Buku



Gambar 52 Controller Buku

Buatlah file seperti pada Gambar 47 yaitu file “booksController.go”, file ini digunakan untuk menampung berbagai *function* untuk *endpoint* yang akan dibuat nanti. Simpan file pada Controllers agar lebih mudah untuk manajemen filenya.

```
package controllers

import (
    "elib_v2/initializers"
    "elib_v2/models"

    "github.com/gin-gonic/gin"
)

func Books(c *gin.Context) {
    var books []models.Books
    initializers.DB.Find(&books)

    c.JSON(200, books)
}
```

Kode 40 Script untuk Controller Buku

Selanjutnya adalah memasukan kode pada file “booksController.go” agar dapat membuat komunikasi dengan *database*. Pertama adalah melakukan import terhadap *library* yang dibutuhkan. Lalu membuat *function* pertama untuk mengambil semua data buku seperti pada Kode 40. Output dari buku akan diambil dengan tipe data *array* agar sejumlah tipe data dapat ditampung dalam satu tempat, apabila pengambilan data berhasil akan direspon dengan HTTP response kode 200 untuk sukses dan output buku akan dikeluarkan.

```
func Viewbookid(c *gin.Context) {
    id := c.Param("id")

    var books []models.Books
    initializers.DB.Find(&books, id)

    c.JSON(200, books)
}
```

Kode 41 Get Buku By ID

Selanjutnya membuat *function* untuk mengambil buku berdasarkan ID seperti pada Kode 41. Kolom ID akan ditampung dengan variabel id dan parameter “id”. Variabel buku akan diambil dengan bentuk array dan hanya akan mengambil *field* “id” nya saja. Memanggil data buku hanya dengan id ini dibutuhkan untuk mengambil item secara spesifik untuk kebutuhan tertentu.

```
func AddBooks(c *gin.Context) {
    // ambil data
    var body struct {
        Judul      string
        Penulis    string
        Penerbit   string
        Tanggal_rilis string
    }

    c.Bind(&body)

    //tambah buku
    post := models.Books{Judul: body.Judul, Penulis: body.Penulis,
Penerbit: body.Penerbit, Tanggal_rilis: body.Tanggal_rilis}
    result := initializers.DB.Create(&post)

    if result.Error != nil {
        c.Status(400)
        return
    }

    c.JSON(200, gin.H{
        "book": post,
    })
}
```

Kode 42 Function AddBooks

Pada Kode 42 merupakan function yang berfungsi untuk menambahkan data buku. *Field* buku akan ditampung terlebih dahulu ditampung dan di *bind* untuk mengubah data menjadi respon JSON, lalu field dari buku akan dideklarasikan ulang dalam variabel post dan variabel result akan memanggil initializer untuk memanggil method POST agar data bisa dimasukan kedalam tabel, terakhir akan direspon dengan output JSON kode 200 apabila input buku berhasil.

```

func UpdateBook(c *gin.Context) {
    // ambil dulu id buku
    id := c.Param("id")

    // ambil data request
    var body struct {
        Judul      string
        Penulis    string
        Penerbit   string
        Tanggal_rilis string
    }

    c.Bind(&body)

    //ambil post update
    var books models.Books
    initializers.DB.First(&books, id)

    //update
    initializers.DB.Model(&books).Updates(models.Books{
        Judul:      body.Judul,
        Penulis:    body.Penulis,
        Penerbit:   body.Penerbit,
        Tanggal_rilis: body.Tanggal_rilis,
    })

    //respon
    c.JSON(200, gin.H{
        "book": books,
    })
}

```

Kode 43 Function Update Buku

Pada Kode 43 adalah *function* yang digunakan untuk melakukan perubahan terhadap data buku. Parameter buku akan dideklarasikan terlebih dahulu pada body, lalu buku akan diambil berdasarkan ID dan dipanggil oleh initialized menggunakan DB.first untuk memanggil *primary key* dari tabel buku tersebut, dan dideklarasikan kembali dalam function model agar dapat melakukan perubahan. Selanjutnya apabila perubahan berhasil akan direspon dengan HTTP response kode 200.

```

func DeleteBook(c *gin.Context) {
    id := c.Param("id")

    initializers.DB.Delete(&models.Books{}, id)

    //respon
    c.JSON(200, gin.H{
        "message": "Buku terhapus!",
    })
}

```

Kode 44 Function Delete Book

Tahap selanjutnya adalah membuat function untuk menghapus buku seperti pada Kode 44. Buku akan dipanggil berdasarkan input ID lalu dipanggil dalam function DB.Delete dan diisi oleh ID dari buku itu sendiri. Apabila proses menghapus buku berhasil maka akan direspon dengan HTTP Response dan message “buku terhapus”.

### 3. Pembuatan URL Endpoint

```

func main() {
    r := gin.Default()
    api := r.Group("/api")
    {
        //crud buku
        api.GET("/books", controllers.Books)
        api.GET("/books/:id", controllers.Viewbookid)
        api.POST("/books", controllers.AddBooks)
        api.PUT("/books/:id", controllers.UpdateBook)
        api.DELETE("/books/:id", controllers.DeleteBook)

    }
    r.Run()
}

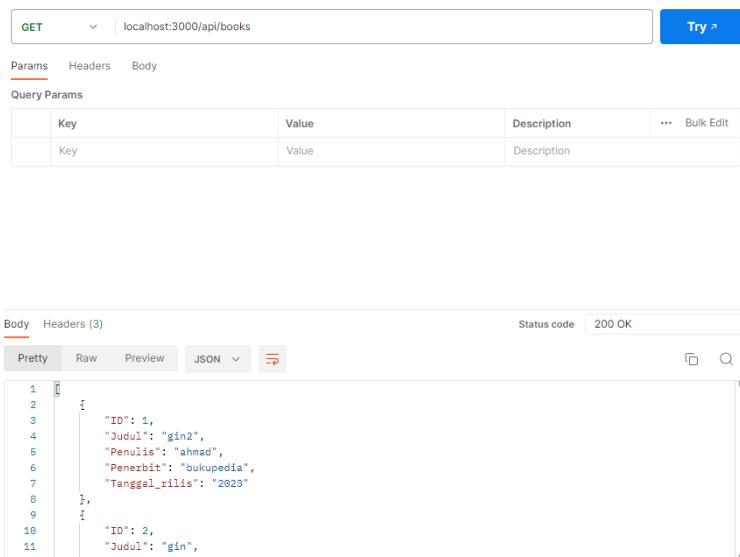
```

Kode 45 Pembuatan Endpoint URL

Pada file “main.go” tambahkan kode untuk membuat rute URL *endpoint* agar dapat memanggil *function* yang telah dibuat. URL akan dibuat secara grup dalam “/api” dan sisanya URL spesifik hanya untuk buku. Terdapat beberapa HTTP *method* yang terdapat dalam URL yang pertama ada GET yang digunakan untuk memanggil data, selanjutnya ada POST untuk menyimpan data, lalu ada PUT untuk mengubah data, dan terakhir ada DELETE untuk menghapus data.

Masing-masing URL mempunyai rute tersendiri, URL yang mempunyai “:id” digunakan untuk memanggil data spesifik menggunakan *primary key* dari tabel buku.

#### 4. Percobaan Endpoint Buku Pada Postman



The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: localhost:3000/api/books
- Params tab is selected, showing a table for "Query Params":

Key	Value	Description
Key	Value	Description

- Headers tab shows "Headers (3)":

Pretty	Raw	Preview	JSON
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29	30	31	32
33	34	35	36
37	38	39	40
41	42	43	44
45	46	47	48
49	50	51	52
53	54	55	56
57	58	59	60
61	62	63	64
65	66	67	68
69	70	71	72
73	74	75	76
77	78	79	80
81	82	83	84
85	86	87	88
89	90	91	92
93	94	95	96
97	98	99	100
101	102	103	104
105	106	107	108
109	110	111	112
113	114	115	116
117	118	119	120
121	122	123	124
125	126	127	128
129	130	131	132
133	134	135	136
137	138	139	140
141	142	143	144
145	146	147	148
149	150	151	152
153	154	155	156
157	158	159	160
161	162	163	164
165	166	167	168
169	170	171	172
173	174	175	176
177	178	179	180
181	182	183	184
185	186	187	188
189	190	191	192
193	194	195	196
197	198	199	200
201	202	203	204
205	206	207	208
209	210	211	212
213	214	215	216
217	218	219	220
221	222	223	224
225	226	227	228
229	230	231	232
233	234	235	236
237	238	239	240
241	242	243	244
245	246	247	248
249	250	251	252
253	254	255	256
257	258	259	260
261	262	263	264
265	266	267	268
269	270	271	272
273	274	275	276
277	278	279	280
281	282	283	284
285	286	287	288
289	290	291	292
293	294	295	296
297	298	299	300
301	302	303	304
305	306	307	308
309	310	311	312
313	314	315	316
317	318	319	320
321	322	323	324
325	326	327	328
329	330	331	332
333	334	335	336
337	338	339	340
341	342	343	344
345	346	347	348
349	350	351	352
353	354	355	356
357	358	359	360
361	362	363	364
365	366	367	368
369	370	371	372
373	374	375	376
377	378	379	380
381	382	383	384
385	386	387	388
389	390	391	392
393	394	395	396
397	398	399	400
401	402	403	404
405	406	407	408
409	410	411	412
413	414	415	416
417	418	419	420
421	422	423	424
425	426	427	428
429	430	431	432
433	434	435	436
437	438	439	440
441	442	443	444
445	446	447	448
449	450	451	452
453	454	455	456
457	458	459	460
461	462	463	464
465	466	467	468
469	470	471	472
473	474	475	476
477	478	479	480
481	482	483	484
485	486	487	488
489	490	491	492
493	494	495	496
497	498	499	500
501	502	503	504
505	506	507	508
509	510	511	512
513	514	515	516
517	518	519	520
521	522	523	524
525	526	527	528
529	530	531	532
533	534	535	536
537	538	539	540
541	542	543	544
545	546	547	548
549	550	551	552
553	554	555	556
557	558	559	560
561	562	563	564
565	566	567	568
569	570	571	572
573	574	575	576
577	578	579	580
581	582	583	584
585	586	587	588
589	590	591	592
593	594	595	596
597	598	599	600
601	602	603	604
605	606	607	608
609	610	611	612
613	614	615	616
617	618	619	620
621	622	623	624
625	626	627	628
629	630	631	632
633	634	635	636
637	638	639	640
641	642	643	644
645	646	647	648
649	650	651	652
653	654	655	656
657	658	659	660
661	662	663	664
665	666	667	668
669	670	671	672
673	674	675	676
677	678	679	680
681	682	683	684
685	686	687	688
689	690	691	692
693	694	695	696
697	698	699	700
701	702	703	704
705	706	707	708
709	710	711	712
713	714	715	716
717	718	719	720
721	722	723	724
725	726	727	728
729	730	731	732
733	734	735	736
737	738	739	740
741	742	743	744
745	746	747	748
749	750	751	752
753	754	755	756
757	758	759	760
761	762	763	764
765	766	767	768
769	770	771	772
773	774	775	776
777	778	779	780
781	782	783	784
785	786	787	788
789	790	791	792
793	794	795	796
797	798	799	800
801	802	803	804
805	806	807	808
809	810	811	812
813	814	815	816
817	818	819	820
821	822	823	824
825	826	827	828
829	830	831	832
833	834	835	836
837	838	839	840
841	842	843	844
845	846	847	848
849	850	851	852
853	854	855	856
857	858	859	860
861	862	863	864
865	866	867	868
869	870	871	872
873	874	875	876
877	878	879	880
881	882	883	884
885	886	887	888
889	890	891	892
893	894	895	896
897	898	899	900
901	902	903	904
905	906	907	908
909	910	911	912
913	914	915	916
917	918	919	920
921	922	923	924
925	926	927	928
929	930	931	932
933	934	935	936
937	938	939	940
941	942	943	944
945	946	947	948
949	950	951	952
953	954	955	956
957	958	959	960
961	962	963	964
965	966	967	968
969	970	971	972
973	974	975	976
977	978	979	980
981	982	983	984
985	986	987	988
989	990	991	992
993	994	995	996
997	998	999	1000

Gambar 53 Get All buku

Pada Gambar 53 merupakan hasil output dari Endpoint URL “localhost:3000/api/books” yang mengeluarkan tipe JSON dengan StatusCode 200 yang berarti data sudah terambil dengan sukses dan data-data buku dikeluarkan dengan array.

POST localhost:3000/api/books

Params Headers Body **•**

none form-data x-www-form-urlencoded raw binary GraphQL JSON **Try** Beautify

```
1 id
2 ... "judul": "gin library",
3 ... "penulis": "ahmad",
4 ... "penerbit": "bukupedia",
5 ... "tanggal_rilis": "2023"
6 
```

Body Headers (3) Status code 200 OK

Pretty Raw Preview JSON **Try**

```
1
2   "book": {
3     "ID": 4,
4     "Judul": "gin library",
5     "Penulis": "ahmad",
6     "Penerbit": "bukupedia",
7     "Tanggal_rilis": "2023"
8   }
9 
```

Gambar 54 Post Books

Gambar 54 merupakan hasil dari output method POST dari URL “localhost:/api/books”. Method tersebut digunakan untuk menginputkan data dari buku dengan form raw dan tipe data JSON, lalu akan direspon dengan output StatusCode 200 dan hasil dari input form tersebut.

PUT localhost:3000/api/books/2

Params Headers Body **•**

none form-data x-www-form-urlencoded raw binary GraphQL JSON **Try** Beautify

```
1
2   "book": {
3     "ID": 2,
4     "Judul": "gin22",
5     "Penulis": "ahmad",
6     "Penerbit": "bukupedia",
7     "Tanggal_rilis": "1985"
8   }
9 
```

Body Headers (3) Status code 200 OK

Pretty Raw Preview JSON **Try**

```
1
2   "book": {
3     "ID": 2,
4     "Judul": "gin22",
5     "Penulis": "ahmad",
6     "Penerbit": "bukupedia",
7     "Tanggal_rilis": "1985"
8   }
9 
```

Gambar 55 Put Books

Pada Gambar 55 adalah Method yang digunakan untuk mengubah data dari buku, pertama ID buku akan diambil dengan mencantumkannya di URL lalu data diinputkan dalam raw body yang berformat JSON. Setelah diinputkan akan direspon dengan output dari inputan itu sendiri dan StatusCode 200 untuk berhasil.

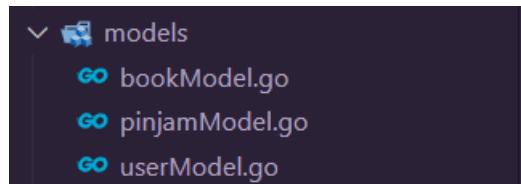
The screenshot shows a REST API testing interface. At the top, there is a header bar with a 'DELETE' button, a URL field containing 'localhost:3000/api/books/4', and a 'Try' button. Below the header, there are tabs for 'Params', 'Headers', and 'Body', with 'Body' being the active tab. Under 'Body', there is a table titled 'Query Params' with columns for 'Key', 'Value', 'Description', and 'Bulk Edit'. The table has one row with 'Key' and 'Value' both empty. Below this table, the 'Body' tab is selected, showing a JSON structure with three lines of code: 1. "message": "Buku terhapus!". The 'Headers' tab shows three entries: 'Content-Type' (application/json), 'Accept' (application/json), and 'Content-Length' (16). The 'Status code' is listed as '200 OK'. At the bottom, there are buttons for 'Pretty', 'Raw', 'Preview', and 'JSON' (with a dropdown menu).

Gambar 56 Delete Buku

Gambar 56 merupakan output dari endpoint Method delete buku. ID buku akan diambil dari URL lalu akan direspon dengan message dari server yaitu “buku terhapus” dan status code 200 apabila respon berhasil.

## C. Pembuatan Endpoint Pinjam Buku

### 1. Membuat Model dan Controller Pinjam



Gambar 57

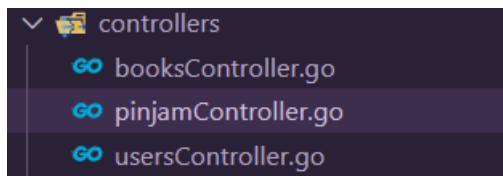
Tahapan yang pertama kali yang harus dilakukan adalah membuat model peminjaman buku seperti pada Gambar 48. Model dari pinjaman buku disimpan dalam folder models bersama dengan model-model lainnya.

```
package models

type Pinjam struct {
    ID          int `gorm:"primaryKey"`
    UserID      int
    User        User
    BooksID     int
    Books       Books
    Tanggal_pinjam string
    Dikembalikan bool
}
```

Kode 46 Model pinjam

Pada kode 46 merupakan struktur dari model pinjam buku. Pada model tersebut terdapat beberapa *field* yang berhubungan dengan beberapa tabel yaitu UserID dan BooksID yang digunakan untuk mengambil *value* primary key dari masing-masing tabel yaitu tabel User dan Books.



Gambar 58 File Pinjam Controller

Pada Gambar 58 merupakan nama dan lokasi dari file “pinjamController.go” yang disimpan bersamaan dengan model-model lainnya.

```
package controllers

import (
    "elib_v2/initializers"
    "elib_v2/models"

    "github.com/gin-gonic/gin"
)

func Pinjam(c *gin.Context) {
    var pinjam []models.Pinjam
    initializers.DB.Find(&pinjam)

    c.JSON(200, pinjam)
}
```

Kode 47 Import dan Function Get All Pinjam

Pada Kolom 47 adalah struktur awal dari controller untuk pinjaman buku. Pertama akan dilakukan *import* terlebih dahulu untuk library yang digunakan, lalu pembuatan *function* untuk pengambilan semua data pinjaman buku yang dideklarasikan dalam bentuk *array*. Lalu function tersebut akan memanggil initializer untuk berkomunikasi dengan database, apabila komunikasi berhasil dilakukan akan direspon dengan kode 200 yaitu StatusOK.

```
func GetPinjamid(c *gin.Context) {
    id := c.Param("id")

    var pinjam models.Pinjam
    initializers.DB.Find(&pinjam, id)

    c.JSON(200, pinjam)
}
```

Kode 48 Get Pinjam By id

Selanjutnya pembuatan *function* untuk mengambil data pinjaman berdasarkan ID seperti pada Kode 48. Pada *function* ini data akan dipanggil

menggunakan initializer dan diambil field id nya saja.

```
func AddPinjam(c *gin.Context) {
    // ambil data
    var body struct {
        UserID      int
        BooksID    int
        Tanggal_pinjam string
        Dikembalikan bool
    }
    c.Bind(&body)

    //tambah buku
    pinjam := models.Pinjam{UserID: body.UserID, BooksID: body.BooksID,
    Tanggal_pinjam: body.Tanggal_pinjam, Dikembalikan: body.Dikembalikan}
    result := initializers.DB.Create(&pinjam)

    if result.Error != nil {
        c.Status(400)
        return
    }

    c.JSON(200, gin.H{
        "pinjaman": pinjam,
    })
}
```

Kode 49 AddPinjam

Pada Kode 49 merupakan kolom untuk menambahkan data pinjaman buku pada tabel pinjam. *Field* terisi dengan beberapa value *foreign key* yang diambil dari *primary key* yang terdapat pada tabel User dan Books. Lalu dideklarasikan pada variabel pinjam dan di-*assign* dalam array. apabila terdapat error pada saat melakukan input, maka akan menampilkan kode 400 Status Bad Request, sedangkan apabila input berhasil akan menampilkan kode 200 StatusOK.

```

func UpdatePinjam(c *gin.Context) {
    id := c.Param("id")

    // ambil data request
    var body struct {
        UserID          int
        BooksID        int
        Tanggal_pinjam string
        Dikembalikan   bool
    }

    c.Bind(&body)

    //ambil post update
    var pinjam models.Pinjam
    initializers.DB.First(&pinjam, id)

    //update
    initializers.DB.Model(&pinjam).Updates(models.Pinjam{
        UserID:      body.UserID,
        BooksID:     body.BooksID,
        Tanggal_pinjam: body.Tanggal_pinjam,
        Dikembalikan: body.Dikembalikan,
    })

    //respon
    c.JSON(200, gin.H{
        "message": pinjam,
    })
}

```

*Kode 50 UpdatePinjam*

Pada Kode 50 merupakan *function* yang digunakan untuk mengubah data peminjaman buku, *field* yang digunakan untuk mengubah data peminjaman buku sama seperti yang digunakan pada penambahan buku yaitu beberapa *foreign key* dll. *Field* id pada tabel pinjam akan dipanggil untuk mengubah data spesifik pada kolom tertentu.

```
func DeletePinjam(c *gin.Context) {
    id := c.Param("id")

    initializers.DB.Delete(&models.Pinjam{}, id)

    //respon
    c.JSON(200, gin.H{
        "message": "Pinjaman terhapus!",
    })
}
```

Kode 51 Function DeletePinjam

Pada Kode 51 merupakan *function* yang digunakan untuk menghapus data pinjaman buku. ID pinjaman akan dipanggil terlebih dahulu untuk memasukan data tertentu kemudian apabila proses penghapusan berhasil maka akan direspon dengan kode 200 dengan statusOK dan message dalam format JSON yaitu “pinjaman terhapus!”.

```
api.GET("/pinjam", controllers.Pinjam)
api.GET("/pinjam/:id", controllers.GetPinjamid)
api.POST("/pinjam", controllers.AddPinjam)
api.DELETE("/pinjam/:id", controllers.DeletePinjam)
```

Kode 52 Tambah Rute pinjam pada main.go

Terakhir adalah menambahkan rute dari peminjaman buku, dengan cara membuka kembali file main.go lalu memanggil *controller* dari pinjamannya tersebut. Terdapat beberapa method yang digunakan pada peminjaman buku yaitu GET untuk mengambil semua data dari pinjaman, POST untuk menambah pinjaman buku, DELETE untuk menghapus pinjaman buku.

## 2. Pengujian Endpoint Pinjaman Buku

The screenshot shows a POST request to the endpoint `/api/pinjam/`. The request body contains the following JSON data:

```
1  {
2   "User": {
3     "ID": 1,
4     "UserID": 1,
5     "User": {
6       "ID": 0,
7       "Email": "",
8       "Password": "",
9       "Name": ""
10    },
11  }
```

The response status code is 200 OK, and the response body is identical to the request body.

Gambar 59 Get All Pinjam

Gambar 59 Merupakan hasil dari output endpoint “/api/pinjam/” yang akan menampilkan semua data pinjaman buku dalam bentuk JSON dan direspon dengan Status Code 200 untuk sukses.

The screenshot shows a POST request to the endpoint `/api/pinjam/`. The request body contains the following JSON data:

```
1  {
2   "User": {
3     "ID": 1,
4     "UserID": 1,
5     "User": {
6       "ID": 0,
7       "Email": "kemarin",
8       "Password": "false"
9     }
10  },
11  "BooksID": 3,
```

The response status code is 200 OK, and the response body is identical to the request body.

Gambar 60 Post Pinjam Buku

Pada Gambar 60 merupakan hasil pengetesan dengan menginputkan data pinjaman yang diambil dari foreign key buku dan user dengan tipe data integer. Respon dari API tersebut mengembalikan hasil dari input dengan array JSON dan Status code 200 untuk sukses.

The screenshot shows a REST API testing interface. At the top, there is a header bar with a 'DELETE' button, a dropdown menu, the URL 'localhost:3000/api/pinjam/2', and a 'Try' button. Below the header, there are tabs for 'Params', 'Headers', and 'Body', with 'Body' being the active tab. Under 'Query Params', there is a table with one row containing 'Key' and 'Value'. In the main body area, there are tabs for 'Body', 'Headers (3)', and 'Status code 200 OK'. The 'Body' tab is active, showing a JSON response with three lines of code: '1', '2 "message": "Pinjaman terhapus!"', and '3'. The 'Headers' tab shows three entries: 'Content-Type: application/json', 'Accept: application/json', and 'User-Agent: Postman/8.0.1'. The 'Status code' tab shows '200 OK'.

Gambar 61 Delete Pinjaman

Pada Gambar 61 merupakan hasil dari pengetesan endpoint “/api/pinjam/{id}” yang digunakan untuk menghapus buku. Buku akan diambil berdasarkan ID yang dari URL lalu dihapus dengan Method Delete dan direspon dengan message “pinjaman terhapus” serta Status Code 200 untuk sukses.

# BAB 5

## Penggunaan Git dan Deploy Aplikasi

Sesudah tahapan pembuatan Front-end dan Back-end selesai tahap selanjutnya adalah penggunaan GIT dan proses *deploy* aplikasi agar dapat digunakan oleh publik, Berikut adalah tahapan-tahapannya.

### A. Penggunaan Git SCM

Git SCM adalah sistem kontrol terdistribusi yang dapat digunakan oleh semua orang karena merupakan salah satu sistem open-source. Git biasanya digunakan untuk distribusi aplikasi antar pengguna maupun perusahaan, berikut adalah tata cara dalam menggunakan Git SCM.

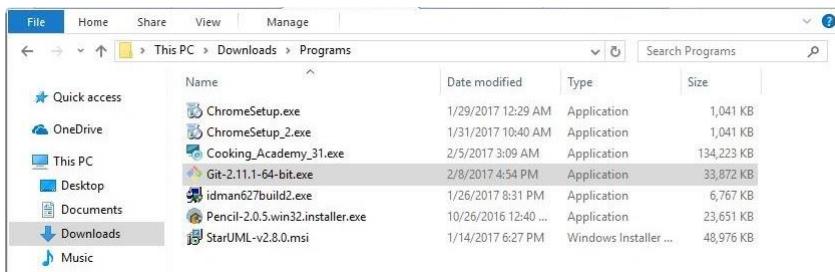
Akses halaman Git SCM official dengan menggunakan url <https://git-scm.com/> lalu tekan tombol “Unduh untuk Windows” seperti pada Gambar 62.



Gambar 62 Halaman Official Git

Apabila proses unduh telah selesai tahapan selanjutnya adalah melakukan pemasangan terhadap Git tersebut dengan cara membuka file yang telah

didownload seperti pada Gambar 62.



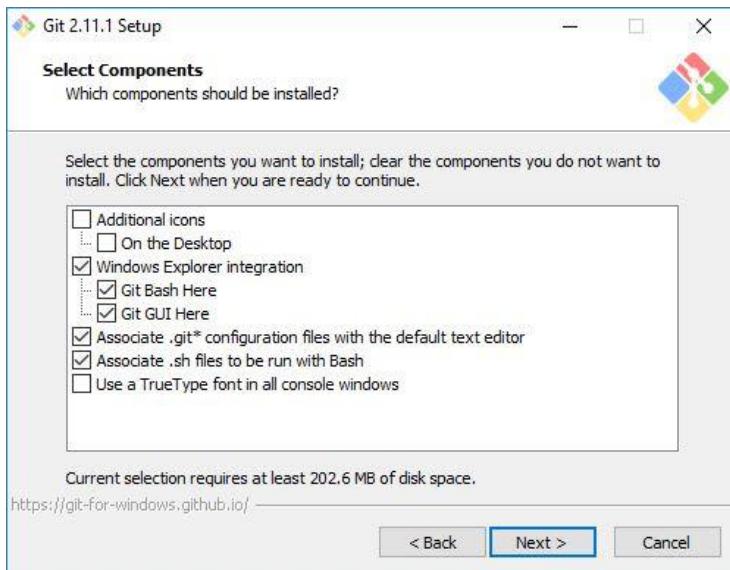
Gambar 63 File Instalasi Git

Setelah File dibuka maka tampilan dari halaman instalasi akan terlihat seperti pada Gambar 63.



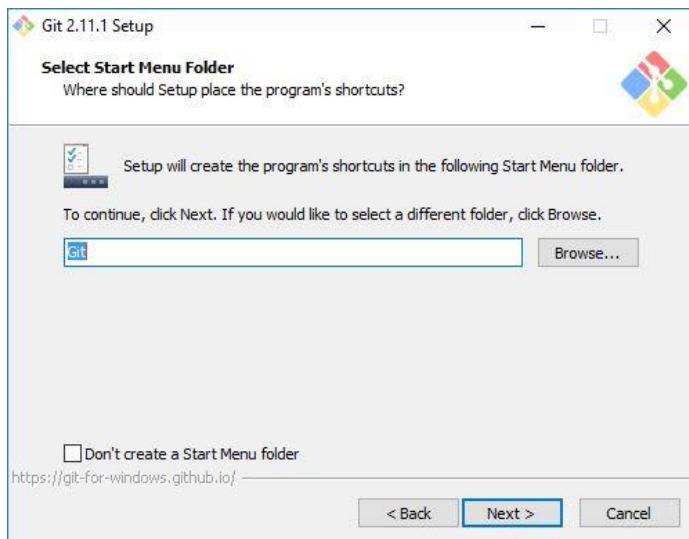
Gambar 64 Halaman Awal Instalasi Git

Tahap selanjutnya adalah memilih kebutuhan apa saja saat menginstall git, disini biarkan saja pengaturan seperti pada Gambar 64.



Gambar 65 Halaman Pemilihan Komponen Git

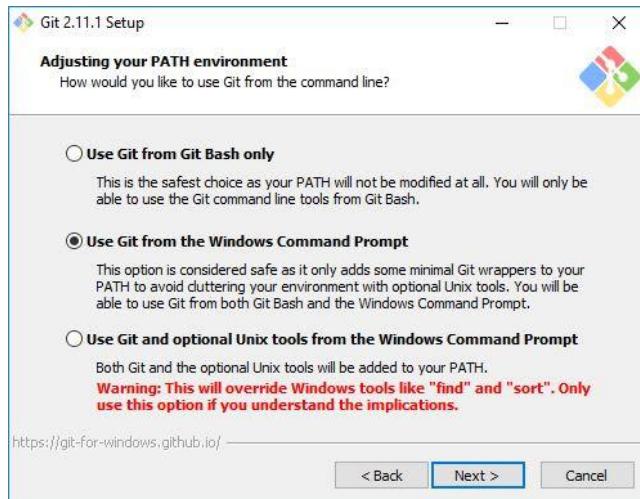
Selanjutnya adalah pemilihan direktori pada git seperti pada Gambar 65, pada tahap ini langsung saja klik next.



Gambar 66 Pemilihan direktori git

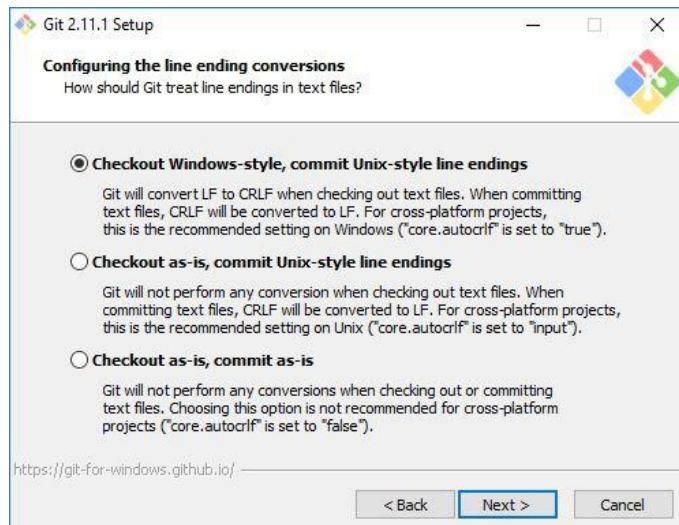
Pada gambar 66 merupakan pemilihan pengaturan untuk PATH dari GIT

tersebut, pilih opsi kedua agar GIT bisa dijalankan dalam command prompt pada Windows.



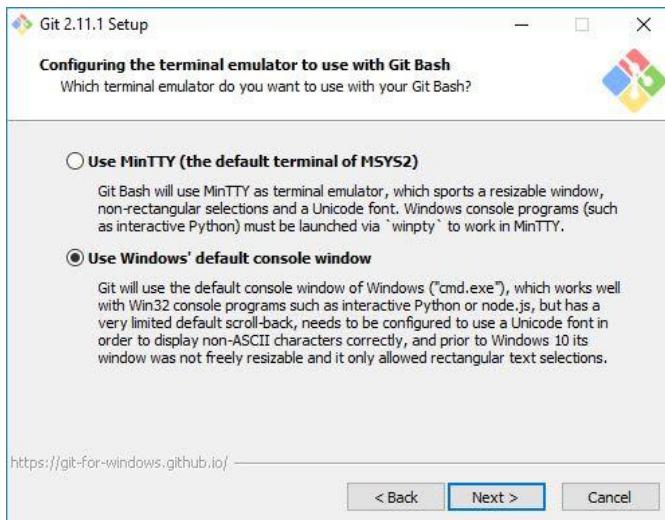
Gambar 67 Pengaturan PATH untuk GIT

Pada Gambar 67 adalah konfigurasi untuk line ending pada git, biarkan saja default lalu klik next.



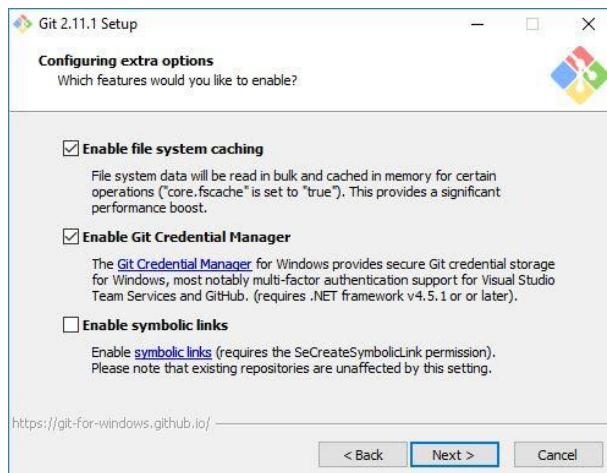
Gambar 68 Konfigurasi Line ending

Pada Gambar 68 adalah pengaturan dari emulator terminal pada git gunakan saja konfigurasi secara *default* lalu klik next.



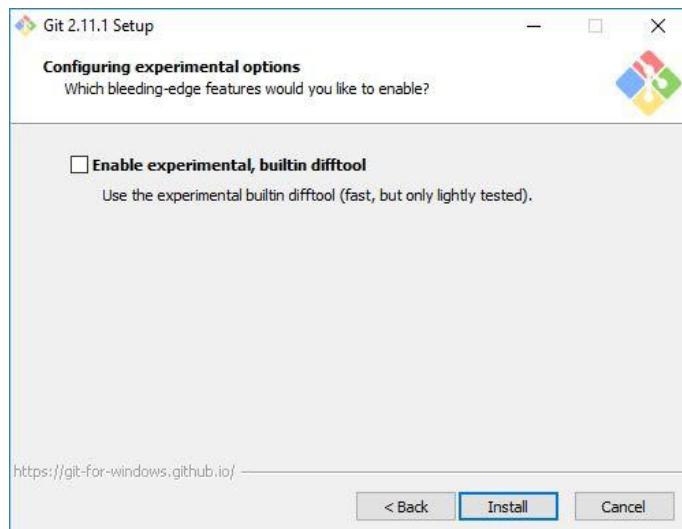
Gambar 69 Pemilihan terminal emulator git

Pada Gambar 69 adalah opsi untuk pengaturan ekstra, disini kita tidak membutuhkan pengaturan ekstra jadi langsung saja untuk menekan tombol next.



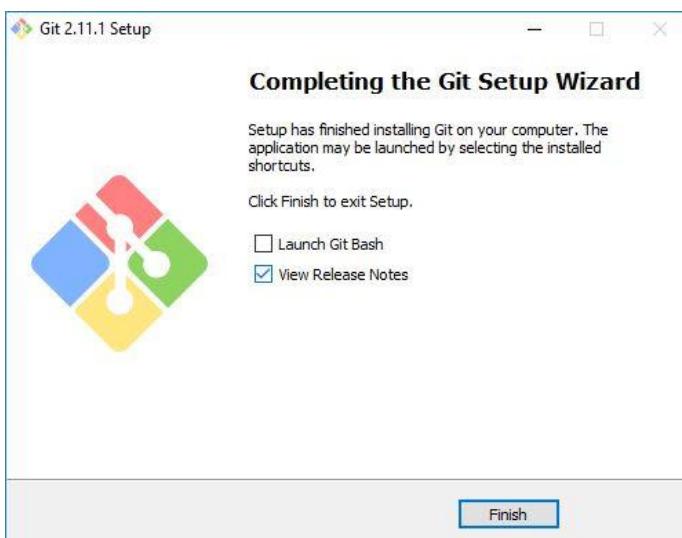
Gambar 70 Pemilihan Opsi Ekstra

Pada Gambar 70 merupakan tahap terakhir dari instalasi Git, disini centang tidak perlu ditekan dan langsung saja tekan tombol install agar installasi segera dilaksanakan.



Gambar 71 Tahap Akhir

Setelah instalasi selesai maka akan ditampilkan halaman seperti pada Gambar 71.



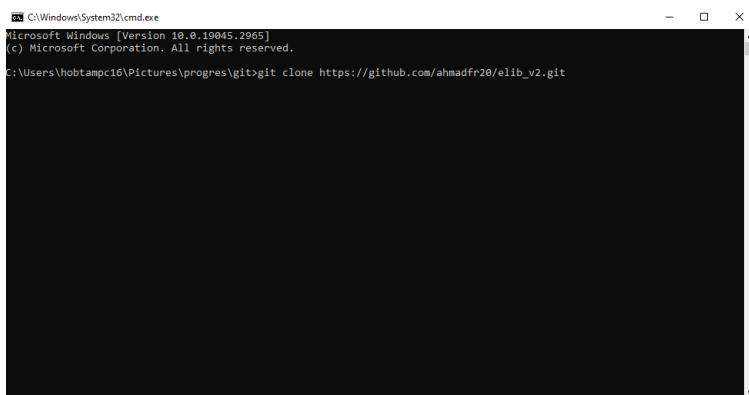
Gambar 72 Tampilan Terakhir Installasi Git

Selanjutnya melakukan konfigurasi Git pada Git Bash/command prompt dengan cara membuka terminal ataupun git bash, lalu mengetikan command seperti pada Kolom dibawah.

```
git config --global user.name  
git config --global user.email
```

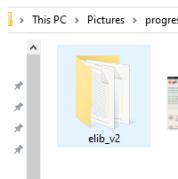
Kode 1 Konfigurasi Git

Setelah Konfigurasi Git selesai, maka Git SCM sudah dapat digunakan untuk melakukan proses kloning, ataupun upload dokumen dari kode yang sudah dibuat.



Gambar 73 Percobaan Kloning Repository

Pada Gambar 73 adalah percobaan kloning untuk repository pada Backend untuk E-library.



Gambar 74 Hasil clone dari repository

Apabila proses klon berhasil, maka file akan langsung terlihat pada direktori yang diinginkan seperti pada gambar 74.

```

49     //crud untuk pinjam
50     api.GET("/pinjam", controllers.Pinjam)
51     api.GET("/pinjam/:id", controllers.GetPinjamid)
52     api.POST("/pinjam", controllers.AddPinjam)
53     api.PUT("/pinjam/:id", controllers.UpdatePinjam)
54     api.DELETE("/pinjam/:id", controllers.DeletePinjam)
55
56     api.GET("/test", controllers.Test)
57

```

Gambar 75 Penambahan Comment

Selanjutnya adalah mencoba untuk melakukan update pada repository dengan menambahkan *comment* pada line 49 pada kode Back-end seperti pada Gambar 75.

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hobtampc16\Pictures\progres\git\elib_v2>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   main.go

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\hobtampc16\Pictures\progres\git\elib_v2>

```

Gambar 76 Pengecekan Status Perubahan

Pada Gambar 76 merupakan cara untuk mengecek status perubahan pada direktori hasil dari clone.

```

C:\Windows\System32\cmd.exe
C:\Users\hobtampc16\Pictures\progres\git\elib_v2>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   main.go

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\hobtampc16\Pictures\progres\git\elib_v2>git add .

C:\Users\hobtampc16\Pictures\progres\git\elib_v2>git commit -m "sedikit edit"
[main 2a13ff9] sedikit edit
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\hobtampc16\Pictures\progres\git\elib_v2>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/ahmadfrz0/elib_v2.git
  0a6b55..2a13ff9  main -> main

C:\Users\hobtampc16\Pictures\progres\git\elib_v2>

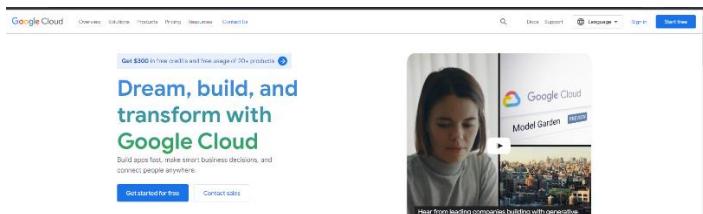
```

Gambar 77 Update pada Repository

Apabila ada perubahan pada saat pengecekan status perubahan, maka repository siap untuk diupdate seperti pada Gambar 77. Pertama masukkan command “git add .” untuk menginput semua file yang terkena perubahan, lalu ketik command untuk memberikan pesan perubahan apa yang telah diinputkan dengan “git commit -m “pesan””, dan terakhir melakukan push terhadap repository dengan command “git push”.

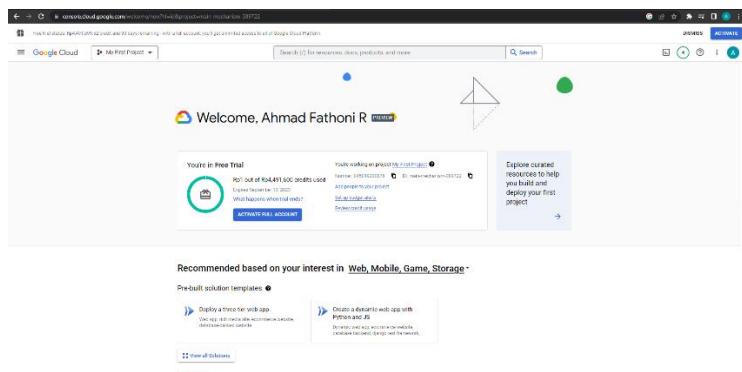
## B. Tahapan Deploy Database

Tahap selanjutnya adalah untuk men-deploy database agar dapat diakses pada API saat melakukan publikasi.



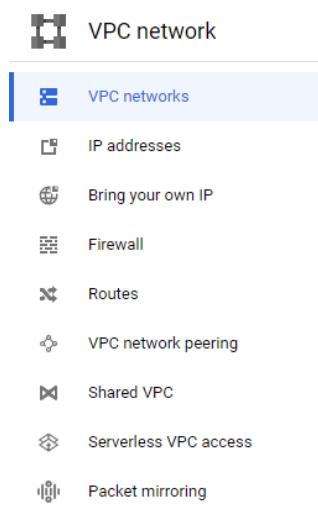
Gambar 78 Halaman Google Cloud

Sebelum memasuki tahap publikasi database yang harus dilakukan adalah membuat akun Google Cloud Platform terlebih dahulu dengan mengakses link “cloud.google.com” seperti pada Gambar 78, hal tersebut bertujuan untuk membuat instansi agar database dapat dipublikasikan klik pada tombol “Get Started For Free”. Google Cloud Platform atau GCP merupakan sebuah platform yang menyediakan jasa untuk membuat *Virtual Machine*, *hosting database*, dll.



Gambar 79 Halaman Utama GCP

Apabila proses register untuk GCP sudah berhasil maka halaman akan terlihat seperti pada Gambar 79.



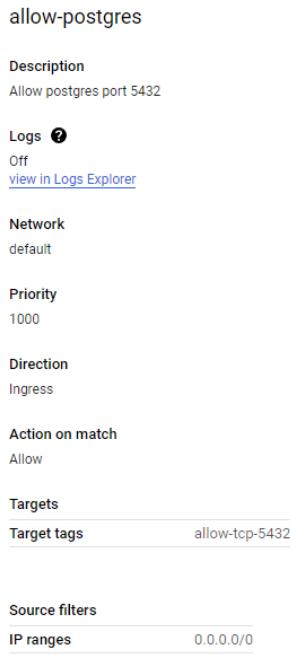
Gambar 80 VPC Network

Selanjutnya adalah melakukan konfigurasi firewall untuk *Virtual Machine* GCP dengan mengetikkan VPC pada searchbar. Hasil akan terlihat seperti pada gambar 80, pada halaman ini pilih Firewall.



Gambar 81 Menu Firewall

Pada menu navigasi *Firewall* pilih “CREATE FIREWALL RULE” seperti pada Gambar 81 untuk membuat pengaturan terhadap firewall.



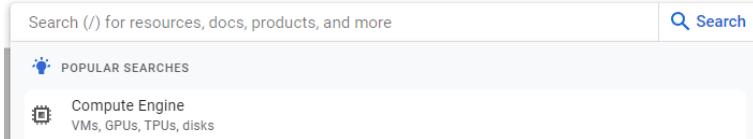
Gambar 82 Pengaturan Firewall

Sesuaikan pengaturan seperti pada gambar 82, dan tag untuk “allow-firewall” akan otomatis terkonfigurasi.



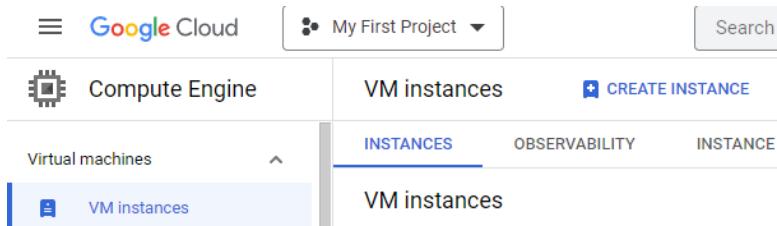
Gambar 83 Hasil Pengaturan Firewall

Pada gambar 83 merupakan hasil dari pengaturan firewall yang dikonfigurasi pada tahap sebelumnya.



Gambar 84 Pembuatan Compute Engine

Setelah konfigurasi VPC dilakukan tahapan selanjutnya adalah membuat Compute Engine untuk membuat *Virtual Machine* agar dapat memulai deploy database. Compute Engine dapat dicari dengan mengetikan Compute Engine pada search bar seperti pada Gambar 84.



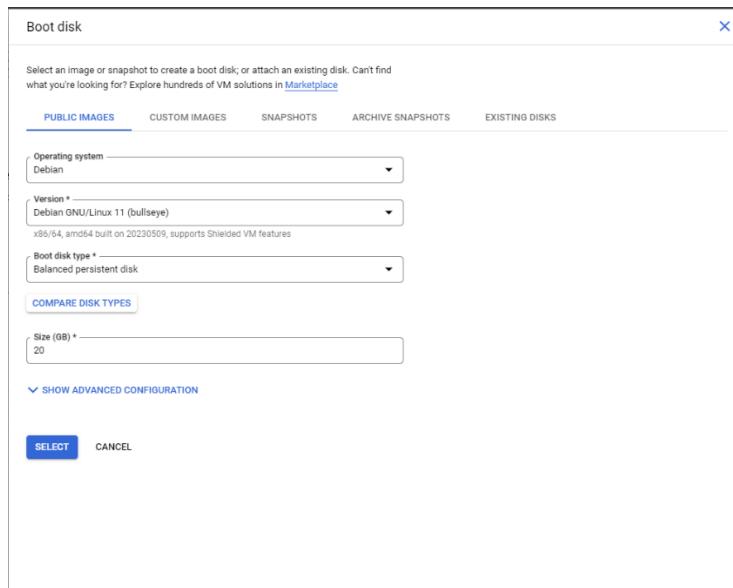
Gambar 85 Pembuatan Instance

Klik pada tombol Create Instance untuk memulai pembuatan *Virtual Machine* seperti pada gambar 85.

A screenshot of the Google Cloud "Create Instance" configuration page. It starts with "MANAGE TAGS AND LABELS" sections for "Region" (set to "us-west1 (Oregon)") and "Zone" (set to "us-west1-b"). Below these are "Machine configuration" sections. Under "Machine types for common workloads, optimized for cost and flexibility", there is a "Series" dropdown set to "E2" and a "Machine type" dropdown set to "e2-micro (2 vCPU, 1 GB memory)". There are also tabs for "General purpose", "Compute optimized", "Memory optimized", and "GPUs".

Gambar 86 Pengaturan Instance

Sesuaikan pengaturan seperti pada gambar 86, untuk machine type kita memakai e2-micro agar tarif dari GCP nya tidak menjadi mahal.



Gambar 87 Pengaturan Boot Disk

Pada pengaturan Boot Disk ukuran diubah menjadi 20 GB seperti pada Gambar 87, agar *Virtual Machine* dapat berjalan dengan mulus.



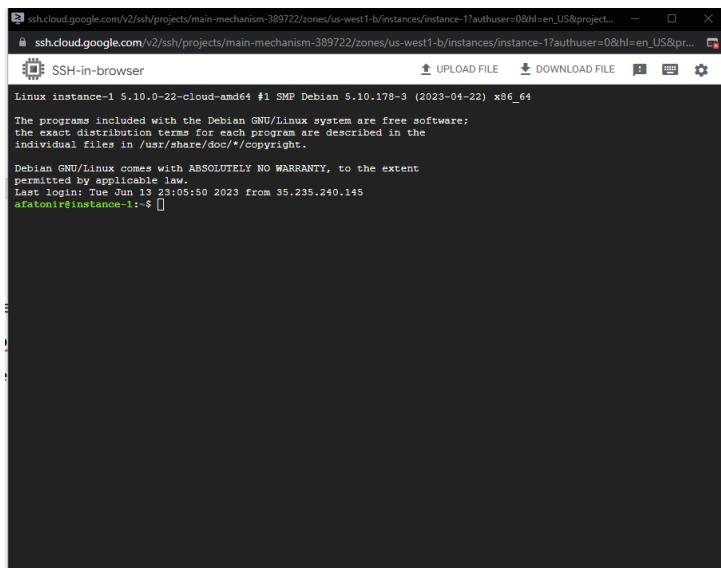
Gambar 88 Pengaturan TCP

Masuk ke Advance Settings lalu pilih networking dan masukkan "allow-tcp-5432" seperti pada Gambar 88.

	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
□	●	instance-1	us-west1-b			10.138.0.2 (node)	35.233.180.60 (node)	SSH

Gambar 89 Hasil Pembuatan VM

Hasil pembuatan VM akan terlihat pada dasbor menu Compute Machine seperti pada Gambar 89. Untuk masuk ke tahap selanjutnya tekan tombol SSH pada kolom instance yang sudah dibuat agar dapat membuka console VM.



Gambar 90 Konsol Virtual Machine

Gambar 90 merupakan hasil dari pembuatan instansi dengan tampilan konsol.

```
-sudo apt-get update
-sudo apt-get install ca-certificates curl gnupg
-sudo install -m 0755 -d /etc/apt/keyrings
-curl -fsSL https://download.docker.com/linux/debian/gpg | sudo
gpg --dearmor -o /etc/apt/keyrings/docker.gpg
-sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

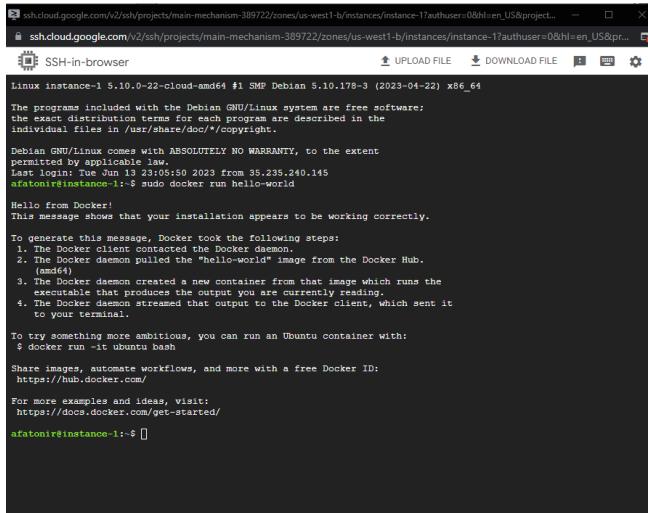
```

-echo \
    "deb [arch="$(dpkg --print-architecture)" signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" |
\
-sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
-sudo apt-get update
-sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin

```

Kode 2 Konfigurasi Docker Pada VM

Tahapan selanjutnya adalah installasi docker dan komponen lainnya pada terminal dengan menggunakan command seperti pada Kode 53. Command dimasukkan secara bertahap tanpa tanda strip.



Gambar 91 Percobaan Docker

Apabila semua tahap installasi selesai, selanjutnya untuk mencoba melakukan *running* pertama pada docker dengan mengetikan “*sudo docker run hello-world*” seperti pada gambar 84. Apabila output sesuai seperti gambar, maka proses installasi sudah selesai.

```
docker run --name some-postgres -e  
POSTGRES_PASSWORD=mysecretpassword -d postgres
```

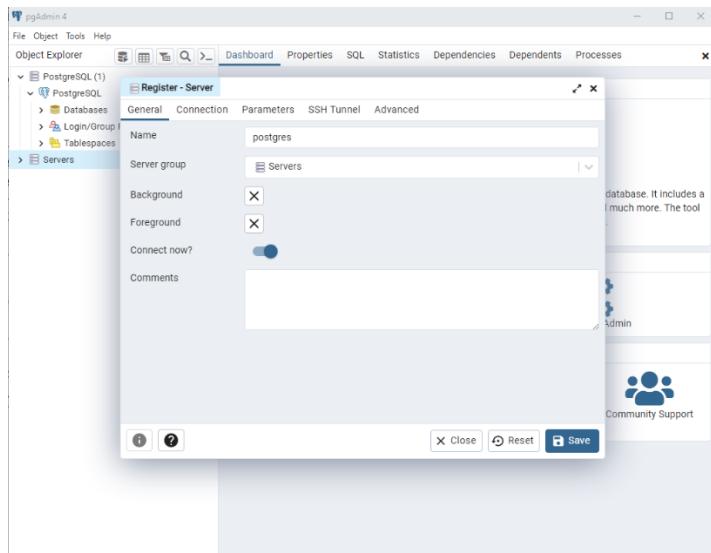
Kode 3 Konfigurasi Postgres Docker

Tahap selanjutnya adalah konfigurasi postgres pada docker seperti pada Kode 54. Masukkan command pada CLI lalu tekan enter.

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
Green	instance-1	us-west1-b			10.138.0.2 (red)	35.233.180.60 (red)	SSH

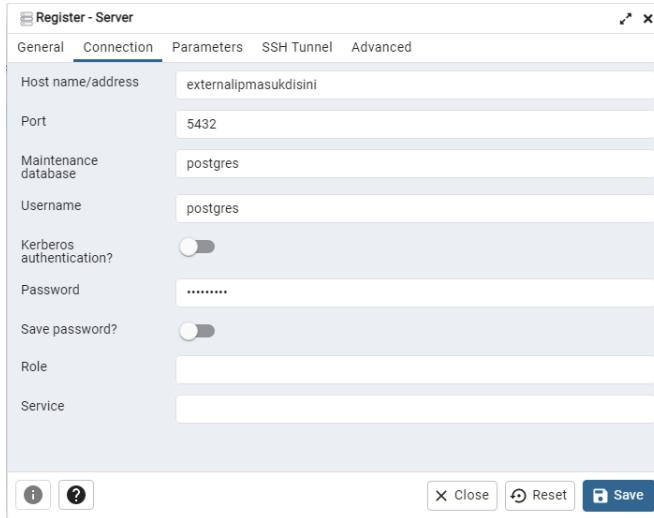
Gambar 92 External IP

Salin External IP yang terdapat pada kolom instance dengan menekan tombol yang ada samping kanan External IP.



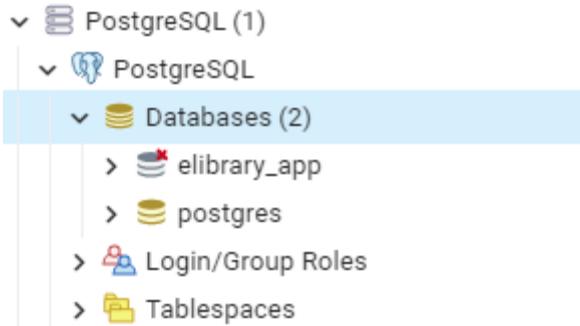
Gambar 93 Konfigurasi pada pgAdmin

Setelah tahap persiapan pada GCP selesai selanjutnya adalah untuk memanggil alamat dari database itu sendiri seperti pada Gambar 93. Nama dari server dapat dibuat sesuai keinginan.



Gambar 94 Konfigurasi Koneksi

Selanjutnya pengaturan pada koneksi seperti pada Gambar 94. Sesuaikan hostname dengan External IP yang sudah disalin pada proses sebelumnya, dan password dengan yang sudah dibuat pada Virtual Machine.



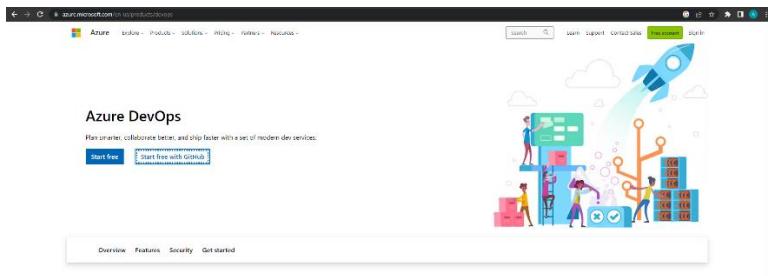
Gambar 95 Hasil Database GCP

Hasil dari konfigurasi dapat dilihat seperti pada Gambar 88 apabila tidak ada kendala. Tambahkan database elibrary\_app agar dapat melakukan automigrate saat deploy API.

## C. Tahapan CI/CD

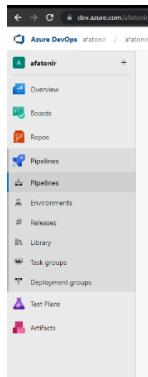
CI/CD merupakan proses integrasi kode kedalam repositori, kemudian dijalankan secara otomatis dengan tujuan untuk menjalankan pengujian secara otomatis, cepat, dan dilakukan secara berulang kali. Proses tersebut akan dijelaskan pada bagian ini pada tahapan-tahapan berikut.

Azure DevOps merupakan perangkat lunak yang dikhususkan untuk bidang DevOps, Azure DevOps ini dibuat oleh Microsoft. Berikut adalah tahapan penggunaannya.



Gambar 96 Landing Page Azure DevOps

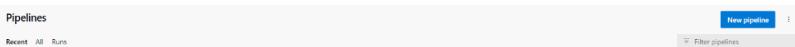
Buka halaman utama Azure dengan mengakses link <https://dev.azure.com/> dan lalu tekan tombol “start free with github” seperti pada Gambar 96.



Gambar 97 Sidebar Azure

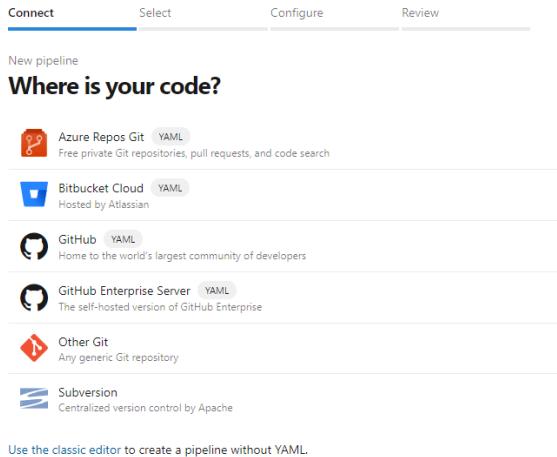
Pada Gambar 97 merupakan sidebar dari Azure, pilih “pipelines” untuk

memulai proses CI/CD.



Gambar 98 Tambahkan Pipeline

Tambahkan pipeline dengan menekan tombol New Pipeline seperti pada Gambar 98.



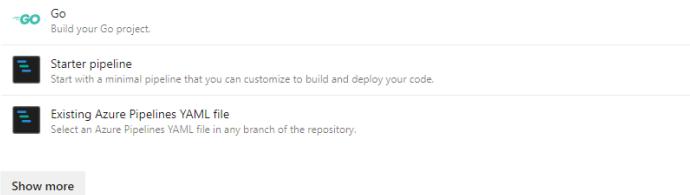
Gambar 99 Pemilihan Repository

Selanjutnya adalah tampilan pemilihan repository seperti pada Gambar 99, pilih Github untuk memilih repository.



Gambar 100 Pilih Repository

Pilih repository 'elib\_v2' seperti pada gambar 100 untuk memulai pengetesan.



Gambar 101 Pemilihan Opsi Build

Pada Gambar 101 merupakan menu pemilihan opsi build. Pilih Go lalu lanjutkan.

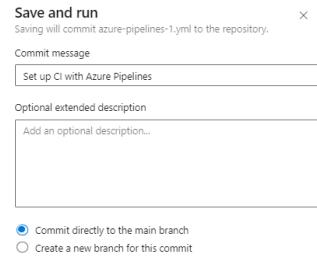
```
YAML
trigger:
- main

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: GoTool@0
  inputs:
    version: '1.13.5'
- task: Go@0
  inputs:
    command: 'get'
    arguments: '-d'
    workingDirectory: '$(System.DefaultWorkingDirectory)'
- task: Go@0
  inputs:
    command: 'build'
    workingDirectory: '$(System.DefaultWorkingDirectory)'
- task: CopyFiles@2
  inputs:
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
- task: PublishBuildArtifacts@1
  inputs:
    artifactName: drop
```

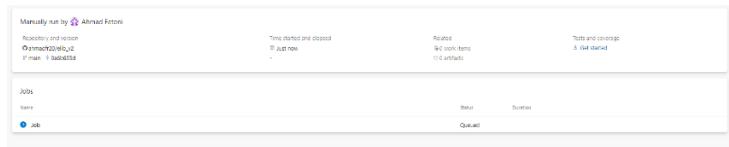
Gambar 102 Script Build

Gambar 102 merupakan kode untuk melakukan build script, ubahlah version pada inputs sesuai dengan versi Go yang digunakan.



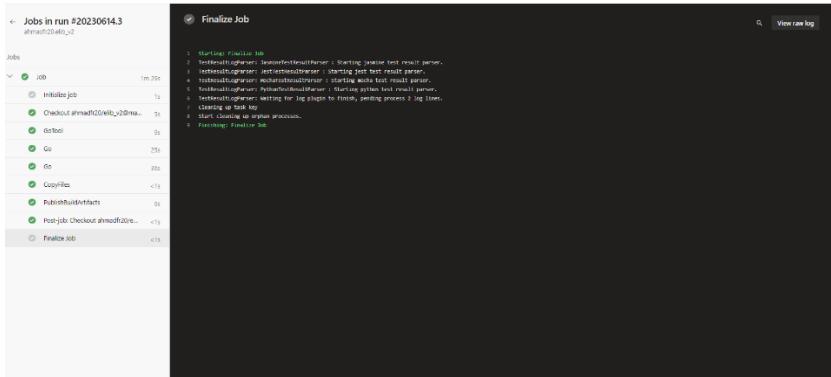
Gambar 103 Opsi Commit

Gambar 96 merupakan tahap terakhir konfigurasi dari CI/CD, yaitu memilih aksi apabila pipeline sudah dijalankan. Pilih commit directly to main branch.



Gambar 104 Proses CI/CD

Gambar 104 adalah proses dari CI/CD tersebut apabila konfigurasi sudah sesuai.



Gambar 105 Hasil CI/CD

Pada Gambar 105 merupakan hasil dari CI/CD, apabila tidak mengalami error atau terdapat warning maka API siap untuk dideploy.

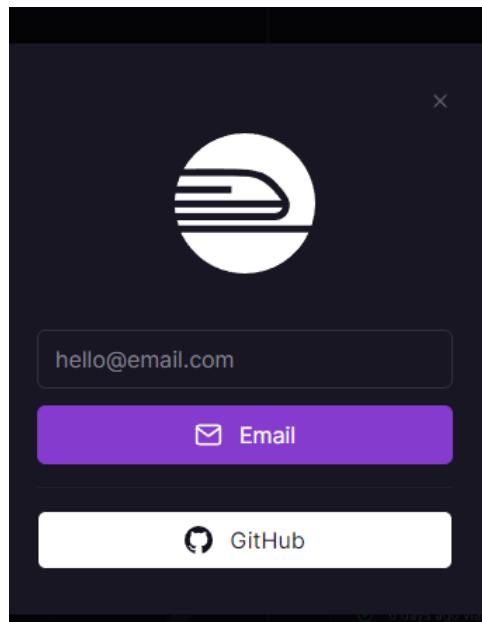
#### D. Tahapan Deploy Backend

Selanjutnya adalah tahapan publikasi terhadap API agar dapat diakses oleh bagian Frontend. Berikut adalah tahapan-tahapannya.



Gambar 106 Railway

Pertama yang harus dilakukan adalah mengakses link “railway.app” seperti pada Gambar 106. Railway merupakan website yang menyediakan jasa untuk Hosting API secara gratis.



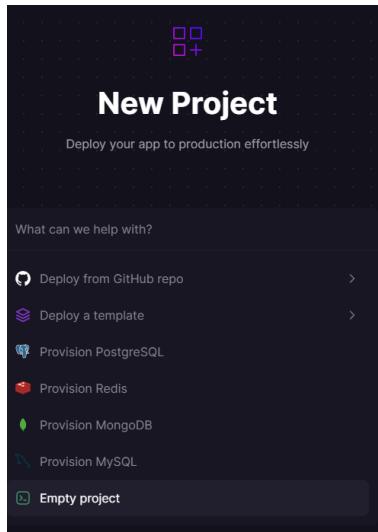
Gambar 107 Login Railway

Gambar 107 merupakan tahap autentifikasi pengguna, Login dengan menggunakan akun github agar dapat mengakses repositori dari API.



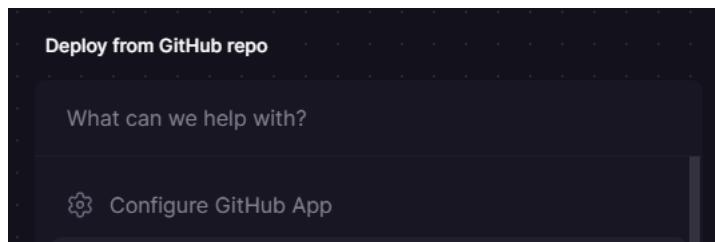
Gambar 108 Pembuatan Project Baru Railway

Pada Gambar 108 Merupakan proses pembuatan project baru pada Railway, klik pada tombol New Project.



Gambar 109 Pemilihan Repository

Gambar 109 merupakan opsi untuk memilih repository yang akan dideploy pada Railway, pilih Deploy from Github repo.



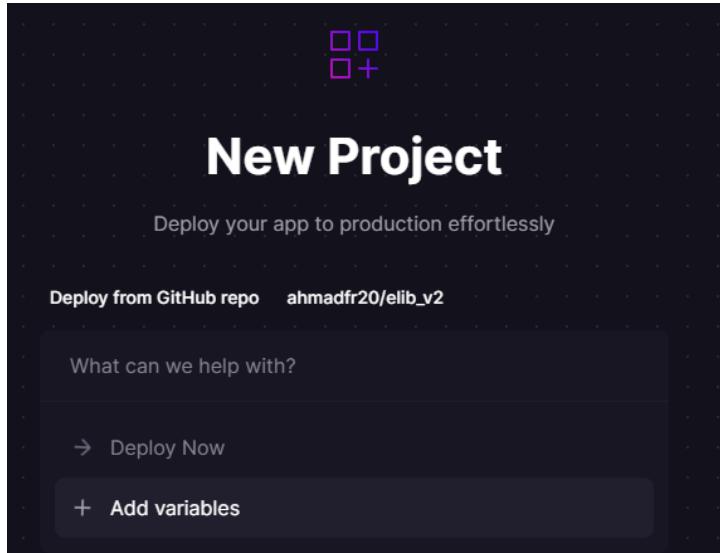
Gambar 110 Konfigurasi Github

Gambar 110 digunakan untuk melakukan konfigurasi Github terlebih dahulu, hal tersebut berguna agar Railway dapat mendeteksi repository apa saja yang terdapat di github.



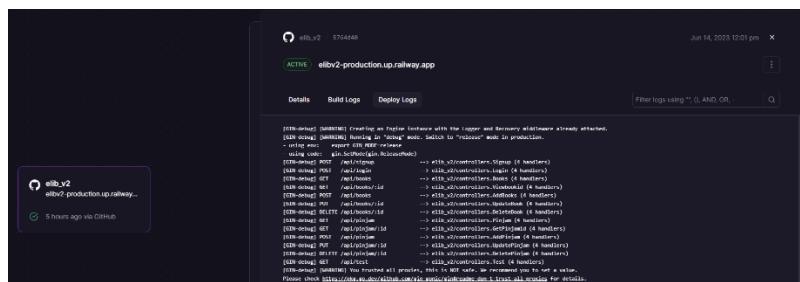
Gambar 111 Pemilihan Repo API

Gambar 111 adalah repository yang akan digunakan untuk dideploy nantinya.



Gambar 112 Proses Akhir

Gambar 112 adalah proses terakhir dalam deploy API, klik pada deploy now agar API segera dipublikasikan.



Gambar 113 Output Hasil Deploy

Gambar 113 merupakan output akhir dari deploy API, apabila tidak menemui masalah apapun maka output akan terlihat seperti pada gambar.

## E. Tahapan Deploy Frontend

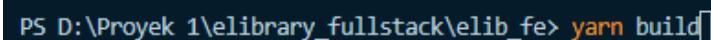
Setelah proses deploy pada bagian Database dan API sudah dilakukan maka tahap terakhir adalah melakukan deploy terhadap bagian Frontend aplikasi, berikut adalah tahapannya.



```
1 import axios from "axios";
2
3 export default axios.create({
4   baseURL: "http://elibv2-production.up.railway.app/api",
5   headers: {
6     "Content-type": "application/json"
7   }
8 });
9 
```

*Gambar 114 Penggantian URL Localhost*

Pertama adalah mengganti URL pada file http-common.js menjadi seperti pada Gambar 114 agar dapat tersambung dengan API.



```
PS D:\Proyek 1\elibrary_fullstack\elib_fe> yarn build
```

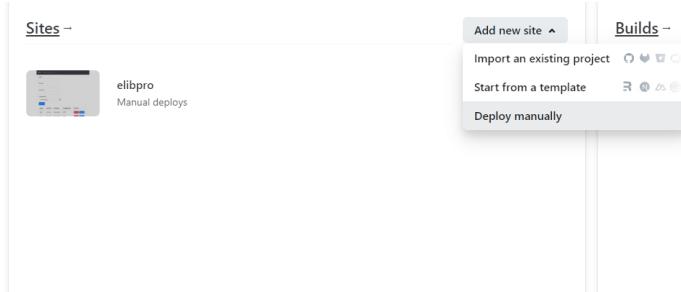
*Gambar 115 yarn build*

Selanjutnya ketikkan command “yarn build” seperti pada gambar 115 pada terminal agar dapat membuat aplikasi dalam tahap produksi.



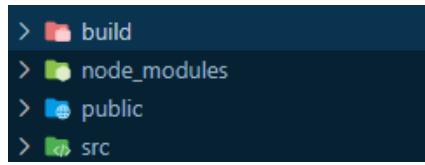
Gambar 116 Landing Page Netlify

Selanjutnya akses website netlify untuk melakukan deployment aplikasi React secara gratis dengan mengakses “netlify.com”. Gambar 116 merupakan tampilan dari website official dari netlify. Klik pada tombol sign up dan pilih github.



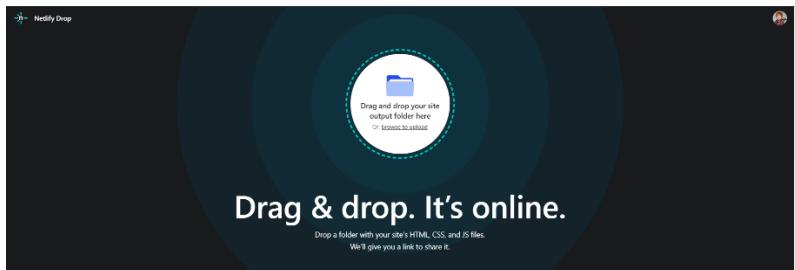
Gambar 117 halaman awal netlify

Gambar 117 merupakan section dari halaman awal Netlify, scroll pada bagian paling bawah dan pilih add new site lalu pilih deploy manually.



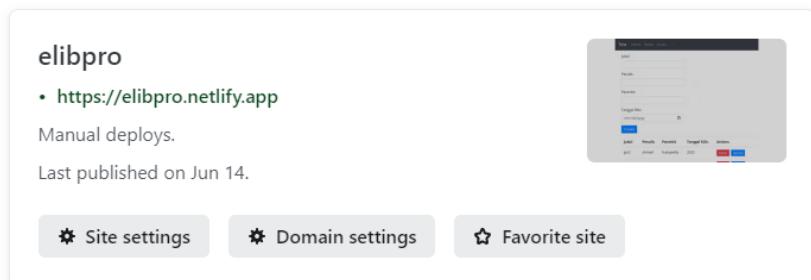
Gambar 118 Hasil yarn build

Gambar 118 merupakan hasil dari proses penginputan command “yarn build”. Kembali ke direktori project react dan lihat folder build.



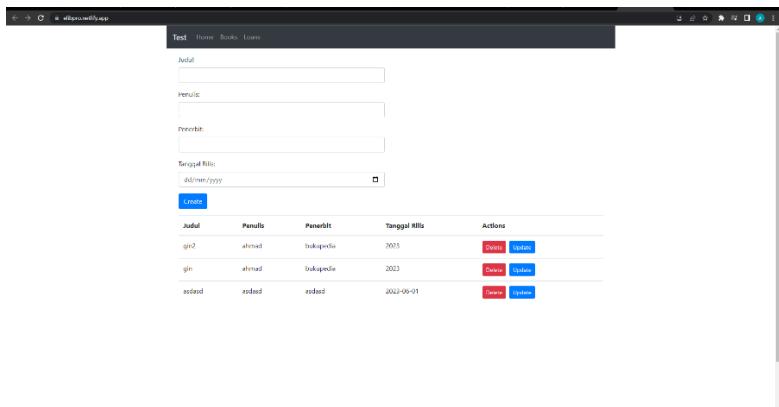
Gambar 119 Drag Drop File

Gambar 119 merupakan form untuk drag drop file yang akan dideploy nantinya. Masukkan file build yang sudah dibuat pada form tersebut.



Gambar 120 URL Hasil Deploy

Gambar 120 merupakan hasil dari URL setelah melakukan deploy, alamat dari website bisa diganti dengan nama yang kita sukai, tetapi untuk “.netlify” belum bisa dihilangkan karena itu merupakan fitur berbayar dari netlify.



Gambar 121 Tampilan pada Browser

Gambar 121 merupakan tampilan saat mengakses URL yang sudah selesai dideploy pada Netlify.

# BAB 6

## Latihan Soal

---

### A. Pilihan Ganda

1. Apa library yang digunakan untuk membuat model dan migrasi database pada bahasa pemrograman GO ....
  - a. gin
  - b. react-router-dom
  - c. gorm
  - d. postgreSQL
  - e. godotenv
  
2. react-router-dom adalah library dari react yang digunakan untuk....
  - a. menyambungkan aplikasi ke database
  - b. berkomunikasi dengan API
  - c. memberikan rute untuk halaman aplikasi
  - d. memberikan sarana untuk auto migrate
  - e. tidak ada jawaban yang benar
  
3. "import elib\_v2/controllers api.GET("/pinjam", \_\_\_\_\_)"  
Agar dapat tersambung dengan controller yang digunakan untuk memanggil semua data pinjaman anda harus menginputkan code pada kolom kosong dengan....
  - a. controllers.Pinjam
  - b. controllers.GetPinjamid
  - c. controllers.Books
  - d. initializers.SyncDatabase
  - e. AllowMethods: []string{GET}
  
4. Library yang digunakan untuk berkomunikasi dengan API pada bagian Front-end adalah ....

- a. gorm
  - b. axios
  - c. Postman
  - d. MySQL
  - e. JSON
5. Respon Status Code dari API yang menunjukan bahwa API berhasil dipanggil adalah
- a. 404
  - b. 204
  - c. 200
  - d. 400
  - e. 502
6. useState adalah salah satu Hook react yang berfungsi untuk...
- a. memperbarui value dari sebuah field
  - b. membuat koneksi dengan database
  - c. mengganti tampilan pada browser
  - d. memberikan rute API
  - e. tidak ada jawaban yang benar
7. Pada saat sebelum akan mendeploy front-end command npm apa yang harus dijalankan terlebih dahulu..
- a. npm start
  - b. npm update
  - c. npm install
  - d. npm run build
  - e. npm init
8. Tipe respon yang dikeluarkan oleh REST API adalah...
- a. integer
  - b. string
  - c. JSON
  - d. XML

- e. JSON array
9. Pada saat proses deploy database platform yang digunakan pada buku tutorial ini adalah...
- a. Google Cloud Platform
  - b. Kubernetes
  - c. Linux
  - d. VMware
  - e. Azure DevOps
10. Dalam Azure DevOps bagian untuk memasukan konfigurasi CI/CD adalah...
- a. pipelines
  - b. docker
  - c. repos
  - d. boards
  - e. artifacts
11. Platform yang digunakan untuk melakukan hosting API adalah...
- a. docker
  - b. kubernetes
  - c. Azure DevOps
  - d. Railway
  - e. Linux
12. File .env merupakan file yang digunakan untuk...
- a. Menampung nilai yang dinamis
  - b. mempercantik tampilan pada browser
  - c. menambahkan data pada tabel
  - d. mengatur alur dari rute aplikasi
  - e. memulai migrasi database secara otomatis
13. Library yang digunakan pada Golang untuk memuat Environment Variable atau .env adalah ...
- a. react-router-dom

- b. godotenv
  - c. gin-gonic
  - d. gorm
  - e. bootstrap
14. Sebelum membuat Instance virtual machine untuk melakukan hosting database pada GCP hal yang harus diatur terlebih dahulu adalah...
- a. langsung saja membuat instance
  - b. melakukan download docker pada Virtual machine
  - c. mengatur firewall untuk allow-5432 pada VPC
  - d. membuat server baru pada postgresql
  - e. mengatur hosting pada API
15. Status Code yang menunjukkan bahwa request API tidak ditemukan adalah
- a. 404
  - b. 502
  - c. 200
  - d. 204
  - e. 400

## B. Isian

```
1. import "_____"
func SyncDatabase() {
    DB._____(&models.User{})
    DB._____(&models.Books{})
    DB._____(&models.Pinjam{})
```

```
}
```

Lengkapi bagian kolom yang kosong agar dapat melakukan migrasi secara otomatis kedalam database!

```
2. func Books(c *gin.Context) {
    var books []models.Books
    initializers.DB.Find(&books)

    c.JSON(____, ____)
}
```

Lengkapi bagian kolom dengan code yang tepat agar dapat merespon dengan status code dan menampilkan data buku!

```
3. func ConnectToDb() {  
    var err error  
    dsn := os.____("DB")  
    DB, err = gorm.Open(postgres.Open(dsn), &gorm.Config{})  
  
    if err != nil {  
        panic("failed to connect to database!")  
  
    }  
}
```

Lengkapi bagian kolom yang kosong dengan kode yang tepat agar dapat mengambil variable DB yang berasal dari env!

4. Sebutkan command dari Go dan React yang digunakan untuk memulai program!

5. Jelaskan yang anda ketahui tentang Google Cloud Platform!

## Daftar Pustaka

- Freeman, A. (2019). *Understanding React*. London, UK: apress.
- Gin. (n.d.). *Gin Web Framework*. (Gin) Retrieved 04 07, 2023 from <https://gin-gonic.com/>
- Iyengar, K., Khanna, S., Ramadath, S., & Stephens, D. (2017). What it really takes to capture the value of APIs. 1-9.
- Kurniawan, N. (2020, 01 02). *Apa Itu API*. (Medium) Retrieved 04 07, 2023 from <https://medium.com/@novancimol12/apa-itu-api-b7ec679a9e24>
- Liu, Z., & Gupta, B. (2019). Study of Secured Full-Stack Web Development. *EPiC Series in Computing*, 317-324.
- McGrath, M. (2020). Go Programming in Easy Steps: Discover Google's Go Language. In M. McGrath, *Go Programming in Easy Steps: Discover Google's Go Language* (pp. 2-4). Warwickshire: In Easy Steps Limited.
- Mendonca, N. C., Jamshidi, P., Garlan, D., & Pahl, C. (2019). Developing Self-Adaptive Microservice Systems: Challenges and Directions. *IEEE Software*, 38(2), 70-79.
- Microsoft. (n.d.). *Why did we build Visual Studio Code?* From Visual Studio Code: <https://code.visualstudio.com/docs/editor/whysvscode>
- Pamungkas, R. (2018). *Teori dan Implementasi Pemrograman Web*. Madiun: Unipma Press.
- ReactJS. (2020). *A JavaScript library for building user interfaces*. From ReactJS: <https://reactjs.org/>
- Sari, D. P., & Wijanarko, R. (2019). Implementasi Framework Laravel pada Sistem Informasi Penyewaan Kamera (Studi Kasus Di Rumah Kamera Semarang). *Informatika dan RPL*, 2(1), 32-36.
- Serrano, D., & Stroulia, E. (2017). Linked Rest APIs: A Middleware for Semantic Rest API Integration. *IEEE 24th International Conference on Web Services*, 138-145.

## TENTANG PENULIS



Ahmad Fathoni R, Lahir di Kota Bandung pada tanggal 20 September 2000. Pendidikan tingkat dasar hingga menengah ditempuh di Kota Sumedang, sedangkan pendidikan tingkat atas di Kota Bandung. Sedang menempuh pendidikan di program D4 TI Program Studi Teknik Infomratika Politeknik Pos Indonesia (Sekarang Universitas Logistik dan Bisnis Internasional).



Roni Andarsyah, S.T., M.Kom., SFPC. Telah menyelesaikan pendidikan Diploma di Politeknik Pos Indonesia di program D3 Teknik Informatika, dan Magister di STMIK LIKMI di Bandung, dan saat ini sedang menjabat sebagai ketua program studi D4 Teknik Informatika di Universitas Logistik Bisnis Internasional.

Buku "Dasar-dasar Pemrograman Web Menggunakan Golang dan ReactJS" adalah panduan praktis yang menawarkan pembaca langkah-demi-langkah dalam membangun aplikasi web modern untuk E-Library. Dalam buku ini, Anda akan belajar tentang penggunaan REST API, Gin Framework Golang, dan ReactJS untuk menciptakan sistem perpustakaan digital dengan integrasi antara Client dan Server.

