

Daftar ISI

<i>Daftar ISI</i>	2
1. <i>Spring Boot Security</i>	16
2. <i>Apa perbedaan utama antara otentikasi JWT dan OAuth?</i>	18
3. <i>JWT (JSON Web Token)</i>	19
4. <i>Cara kerja JWT</i>	21
a. <i>Spring Security JWT in Spring Boot 2</i>	21
5. <i>Implementasi Spring Security Pada Java Spring Boot</i>	24
a. <i>Pom.XML</i>	24
b. <i>Entitas/Model</i>	26
Client	26
Role	30
Role Path	31
User	32
c. <i>Repository</i>	36
ClientRepository	36
RolePathRepository	36
RoleRepository	37
UserRepository	37
d. <i>Service</i>	38
DatabaseSeeder	38
Oauth2ClientDetailsService	41
Oauth2UserDetailsService	42
TemplateCRUD	43
e. <i>Configurasi Security</i>	44
CorsOriginConfiguration	44
Oauth2AccessTokenConverter	45
Oauth2AuthorizationServerConfiguration	46
Oauth2ResourceServerConfiguration	47
WebSecurityConfiguration	49
Config	51
f. <i>Aplication.properties</i>	51
g. <i>Run and Testing with Postman</i>	51
Testing by postman hit login and get token jwt	52
Testing Junit :Login	52
Cara menggunakan token jwt	53
h. <i>Github Project</i>	55
i. <i>Login form security default</i>	55

6.	<i>Mengatur Hak Akses/ROLE pada Akses Endpoint REST API</i>	55
a.	Role dengan : Endpoint Global	55
	Buat endpoint beserta role yang diberikan	55
	Buat rest API sesuai endpoint yang telah di daftarkan	56
	Testing dengan ROLE/Hak Akses yang Sesuai	57
	Testing dengan ROLE/Hak Akses yang Tidak Dimiliki oleh User	59
	Branch	59
b.	Authority Role di method REST API	60
	referensi	60
	Class WebSecurityConfiguration	60
	Application.Properties	60
	Penerapan pada Rest API	61
	Testing : User tidak memiliki role_admin	62
	Testing: User memiliki role_admin	63
c.	HashAuthority Role di method REST API	64
	Testing Postman : Kondisi Success	64
d.	Branch	65
7.	<i>Get User Detail Dari Token/ Read Token / Baca Token</i>	66
a.	Controller	66
b.	Service	67
c.	Service Implementasi	67
d.	Testing Postman	68
e.	Branch	69
8.	<i>Register</i>	70
a.	Register Manual	70
	RegisterModel	70
	Service	70
	Service Impl	70
	Controller	72
	Allow URI:	73
	Testing :Run postman :	73
	Branch :	73
b.	Register By OTP : send OTP to email	74
	Step 1 :	75
	Step 2 : Pom.xml	76
	Step 3 :application.properties	76
	Step 4 : UserService dan UserServiceImpl	77
	Step 5 : Controller Register	78
	Step 6 : utils : Email Templates	79
	Step-7 : SimpleStringUtils	80
	Step-8 : Service : EmailSender	82
	Step-9: Postman	83

Step-10: Lihat inbox email	84
Branch	84
c. Konfirmasi OTP Token User: BY API	84
Controller API Confirm	85
Config	85
Testing Postman	86
d. Register OTP by URI Tymeleaf	86
9. Register OTP by URI Tymeleaf	86
a. Alur	86
b. Branch :	86
c. POM.xml	87
d. Step 1-Controller	87
e. Step 2- Application.properties	88
f. Step 3- Controller Tymeleaf	88
g. Step 4- UI Tymeleaf	90
h. Controller register	90
i. Test Run Postman	91
10. Forget Password : send OTP	93
a. Forget Password OTP Email	93
Controller	93
Model Request	96
Email Template	96
Config	98
b. Test Postman	99
Step 1 send email	99
Step 2 : Chek OTP	99
Step-3: Do changes password	101
11. QUERY JPA	101
a. Eror JPA	101
Pageabel tidak bisa menggunakan IN, solusi native query	101
b. Hati-hati, jika native query, maka deleted_date ikutan ketampil	102
c. Filter And Sorting With JPA	102
d. Criteria QUERY JPA	105
Controller	106
Repository update	107
Criteria query dengan like and lowercase	107
12. File Handling/File Upload /Download File	107

a.	Upload File	107
b.	Service api	108
c.	Tambahkan pada class main	114
d.	Testing	115
	Upload 1 file	115
	Upload many file	116
	Download file	117
e.	Download file tampa download / show file only	117
f.	Jika implementasi web Flux,	119
g.	Branch dan postman lain	121
13.	Deploy Project ke Heroku menggunakan Github	121
a.	Buat akun Heroku	121
b.	Buat Akun Github	121
c.	Masuk Ke akun Heroku, untuk deploy aplikasi	121
d.	Membuat database di heroku	125
e.	Membuat env prod/staging di spring boot	127
	Bagaimana cara melihat env yang sedang aktif ?	128
	Bagaimana mengetahui env yang sedang aktif?	128
f.	Tambahkan pada pom.xml	129
g.	Aotomatic deploy heroku	129
h.	Akses Domain	131
14.	Cron Job Scheduler	132
a.	Step 1- Application Properties	137
b.	Config	137
c.	Main Application	139
d.	Cronjob Example	139
e.	Output	140
f.	Cronjob Generator	140
g.	Branch	140
15.	CronJob Database	140
16.	Download image dari gambar kemudian simpan ke local	141
17.	Docker Spring Boot / Example	141
18.	Docker spring boot	141

a.	Step 1	141
b.	Step 2 : .env	142
c.	Step 3 : docker-compose.yml	142
d.	Step 4: run and noted	144
19.	Docker-compose with heroku-Database : failed	145
a.	Step-1	145
b.	Step-2 : register railway.app untuk database	146
c.	Note:	146
d.	Docker-run	146
e.	Tessting output	147
f.	Branch :	147
20.	Send File To Email	147
a.	Step 1 – Struktur	158
b.	Step 2 :code	158
c.	Step 3 : pom.xml and application.properties	161
d.	Step 4: resouces	162
e.	Lokasi Direktori file	164
f.	REST API	164
g.	Step 5 : Testing By Junit	165
h.	Output	166
i.	Send Multi File	166
21.	Login Costum : With Username And Password Only	168
a.	branch	168
b.	Step 1- pom.xml	168
c.	Step 2- buatlah class userservice	169
d.	Step 3- user impl	169
e.	Application.properties	171
f.	Response Global	171
g.	Model – request	173
h.	Controller	174
i.	Confiq : allow Uri	175
j.	Testing with junit	176

k.	<i>Testing with Postman</i>	178
22.	<i>Cara menggunakan token bearer</i>	178
a.	<i>Get token saat login</i>	178
b.	<i>Input token di header dan output</i>	179
23.	<i>Refresh TOKEN</i>	179
a.	<i>Step 1- lakukan login</i>	179
b.	<i>Step 2- to do refresh token</i>	180
24.	<i>Upload dan simpan barang satu enpoind</i>	180
a.	<i>Output</i>	181
25.	<i>Validation Password saat register</i>	182
a.	<i>Step 1- Pom.xml</i>	182
b.	<i>Step 2-application properties</i>	182
c.	<i>Step 3- struktur kode</i>	184
d.	<i>Step 3 a - setting Character Password dimana?</i>	190
e.	<i>Step 4 – cara menggunakan method validation password ?</i>	190
f.	<i>Tester</i>	191
26.	<i>Menampilkan Eror Validation Hibernate</i>	192
a.	<i>Strukture</i>	192
	<i>apiResponse</i>	193
	<i>Handler</i>	193
b.	<i>POM.xml</i>	194
c.	<i>Controller</i>	195
d.	<i>Entity</i>	195
e.	<i>Testing</i>	196
27.	<i>Validasi</i>	196
a.	<i>Validasi Email manual</i>	196
b.	<i>Validasi String manual</i>	196
28.	<i>Anotasi hibernate</i>	197
a.	<i>@Max @Min :Batasin Request maksimal 5 huruf</i>	197
b.	<i>@Size Batasin Request maksimal 5 huruf</i>	197
29.	<i>Mengenal Lambda Expression untuk Membuat Fungsi Anonymous di Java</i>	198
a.	<i>Apa itu Fungsi Anonymous?</i>	198

b.	Cara Membuat Fungsi Anonymous di Java	199
c.	Mengapa Harus Pakai Fungsi Anonymous?	201
d.	Contoh Program Fungsi Anonymous	202
e.	Akses Variabel untuk Fungsi Anonymous	204
f.	Akhir Kata...	205
30.	MicroService	205
a.	Apa itu MicroService ?	205
b.	Apa Mamfaat ?	205
c.	Perbedaan MicroService VS Monolitik	206
d.	Monolithic Architecture	206
e.	Microservice Architecture	207
f.	MicroService and Container	208
g.	Apakah keuntungan dari Microservice Architecture?	208
h.	Contoh penerapan MicroService	210
	Service 1: Employee	210
	Service 2: Consumer	210
	Service 3: Apps	210
	Output	213
i.	Branch	214
31.	Deploy Heroku Lewat CLI	215
32.	Google Oauth Login – Register	216
a.	Github	216
b.	register untuk dapatkan id google	216
c.	Client id dan token example	223
d.	Pom.xml	224
e.	Resources id google	224
f.	Resources -register timeleaf	225
g.	Daftarkan uri di google	226
h.	Config Security	226
i.	Login controller : Penting	227
	Logic : Login and Register, if user not found, maka do register	227
	If hanya Login saja	232
j.	Register controller	234
k.	Dao request	235

l.	Testing Login with Google	235
m.	Lakukan testing di postman untuk endpoint login atau register	250
n.	referensi	250
o.	Logout oauth2.0 google	250
p.	Melihat informasi expired token oauth gogle	251
	contoh	251
	output	251
33.	Railway.app : deploy project	252
a.	Register uses github	252
b.	create database	252
c.	Push project di github	253
d.	Create project : di railway	253
	Step-1 : pilih github	253
	Step-2: pilih project github	254
	Step-3:Pom.xml	254
	Setting application properties : dengan database railway.app	254
	Step-4: deploy	255
	Redeploy ulang : jika gagal deploy	255
	Jika sudah berhasil : lakukan genereate dns hostname	256
	Jalankan :	256
e.	Railway with Docker-compose	256
34.	FLY.IO-Not Working	258
a.	Instalasi	258
b.	Sign-up	259
c.	Login	260
d.	Check Your App's Status	261
e.	Lounching	261
f.	fly apps open /fred	263
g.	Failed, karena perlu payment cc	265
35.	Logging	265
a.	Apa itu logging?	Error! Bookmark not defined.
b.	Hirarki logging	Error! Bookmark not defined.
c.	Baca log logging	Error! Bookmark not defined.
d.	Cara membuat logging	Error! Bookmark not defined.
e.	Cara membuat output logging : File	Error! Bookmark not defined.
	Struktur project	Error! Bookmark not defined.

Directori file ada disini : _____ Error! Bookmark not defined.
Output: _____ Error! Bookmark not defined.
Branch _____ Error! Bookmark not defined.
https://github.com/rikialdi/binar_batch_5/pull/new/241122-logging _____ Error! Bookmark not defined.

f. memunculkan trace dand debug _____ Error! Bookmark not defined.

36. Download sub Folder Github _____ 266

<https://minhaskamal.github.io/DownGit/> _____ 266

37. Kafka : message bloker _____ 267

a. **Install Download Kafka** _____ 267

URL _____ 267

Link download kafka _____ 267

Step 3: Copy the path of the Kafka folder. Now go to *config* inside kafka folder and open *zookeeper.properties* file. Copy the path against the field *dataDir* and add */zookeeper-data* to the path.

```
13 # See the License for the specific Language governing permissions
14 # limitations under the License.
15 # the directory where the snapshot is stored.
16 dataDir=c:/kafka/zookeeper-data
17 # the port at which the clients will connect
18 clientPort=2181
19 # disable the per-ip limit on the number of connections since
20 maxClientCnxns=0
```

For example if the path is *c:/kafka* _____ 268

Step 4: Now in the same folder *config* open *server.properties* and scroll down to *log.dirs* and paste the path. To the path add */kafka-logs* _____ 268

Step 5: This completes the configuration of zookeeper and kafka server. Now open command prompt and change the directory to the kafka folder. First start zookeeper using the command given below: _____ 268

b. **Integrasi Kafka With Spring Boot : Static Topic/Tabel** _____ 271

Pom.xml _____ 271

Application.properties _____ 272

Struktur project _____ 272

KafkaController _____ 272

KafkaConsumer _____ 273

KafkaProducer _____ 273

Testing with Postman _____ 274

Branch _____ 274

c. **Integrasi Kafka With Spring Boot : Dinamic Topik/Tabel** _____ 274

Struktur project _____ 275

Controller Kafka _____ 275

KafkaTes _____ 275

Branch _____ 276

38. CI-CD	277
a. CI ?	277
b. CD	278
c. Kenapa perlu sebuah CI/CD?	279
d. ISTILAH CI-CD	280
e. Contoh	281
Buat file name dengan nama :	281
Isi file	282
output	284
Output pipeline	284
Ouput jobs	285
39. Jenkin	286
a. Donwload	286
b. RUN JENkIN windows	286
40. Cara Akses Website Diblokir Telkomsel	287
41. Socket Spring Boot/Example	288
42. Web socket react js dan java spring boot	290
a. Clone	290
b. Running BE	291
Update Pom.xml	Error! Bookmark not defined.
Create db	Error! Bookmark not defined.
Application properties	Error! Bookmark not defined.
Output run be	Error! Bookmark not defined.
c. Instal npm	291
d. Running FE	291
Open cmd	291
Ketikkan	291
43. Ngrox	294
a. pengertian	294
b. Download dan register	294
login dulu	294
Kemudian download : tampilan setelah download	294
c. Klik 2 Kali file ngrok.exe	295
d. Connect your account	295
e. run	296
f. output	296

g.	Testing di chrome	296
h.	Kesimpulan	297
44.	Jmeter	298
a.	Pengertian	298
b.	Download	298
	donwload	299
	exkrak file	299
c.	Run Jmeter	300
d.	Menambahkan User Defined Variables	300
	Add user defined variables	301
	Config user defined variables	301
e.	Menambahkan HTTP Header Manager	302
	Add header	303
f.	Menambahkan HTTP Request Default	303
	setting	304
g.	Menambahkan Thread Group	304
	Setting	305
h.	Menambahkan HTTP Request	306
	Setting	306
i.	Menambahkan report untuk menampilkan hasil Performance test	307
j.	Menjalankan Performance test	307
k.	Contoh Hasil Performance Test	308
l.	Kesimpulan	308
m.	Application.properties	309
n.	referensi	310
45.	Transactional spring boot	311
a.	Studi kasus pada tabel barang	311
b.	Service Barang	311
c.	Service Implementasi barang	312
d.	Testing Tampa @Transactional	314
e.	Testing Dengan @Transactional	315
f.	Testing dengan @Transactional ReadOnly True	317
g.	Branch	319
h.	Noted	319
46.	Redis + Spring Security + Docker	319

a.	Instal Docker Compose	320
b.	Docker Compose File	320
c.	Copy file conf Redis	321
d.	Koneksi di application properties	321
e.	Pom.xml	321
f.	Confiq Redis dengan token spring security	322
	Pada class OAuth2AuthorizationServerConfiguration	322
	Pada class WebSecurityConfiguration	323
g.	Running docker: untuk runnig redis yang ada di docker	323
h.	Running aplikasi manual: karena tidak menggunakan docker	324
i.	Jika eror	324
j.	Test postment login	325
47.	Junit and tests postman example JSON OBJEK	325
48.	Swagger	326
a.	Application	326
b.	Pom.xml	327
c.	Cara Access	327
d.	Testing -Try out	328
e.	Detail di Doc Spring JPA	329
49.	Application.properties set value external	329
a.	Di pom.xml	329
b.	Folder berada	330
c.	Cara manggil di applications.properties	330
d.	Ref:	330
50.	issue and solving	331
a.	hide _Links in api	331
b.	Kill port runing	331
51.	Scrum Master	332
52.	Sonarlint	332
a.	Install intelliij	332
b.	Tujuan	333
53.	update maven security and solving	333

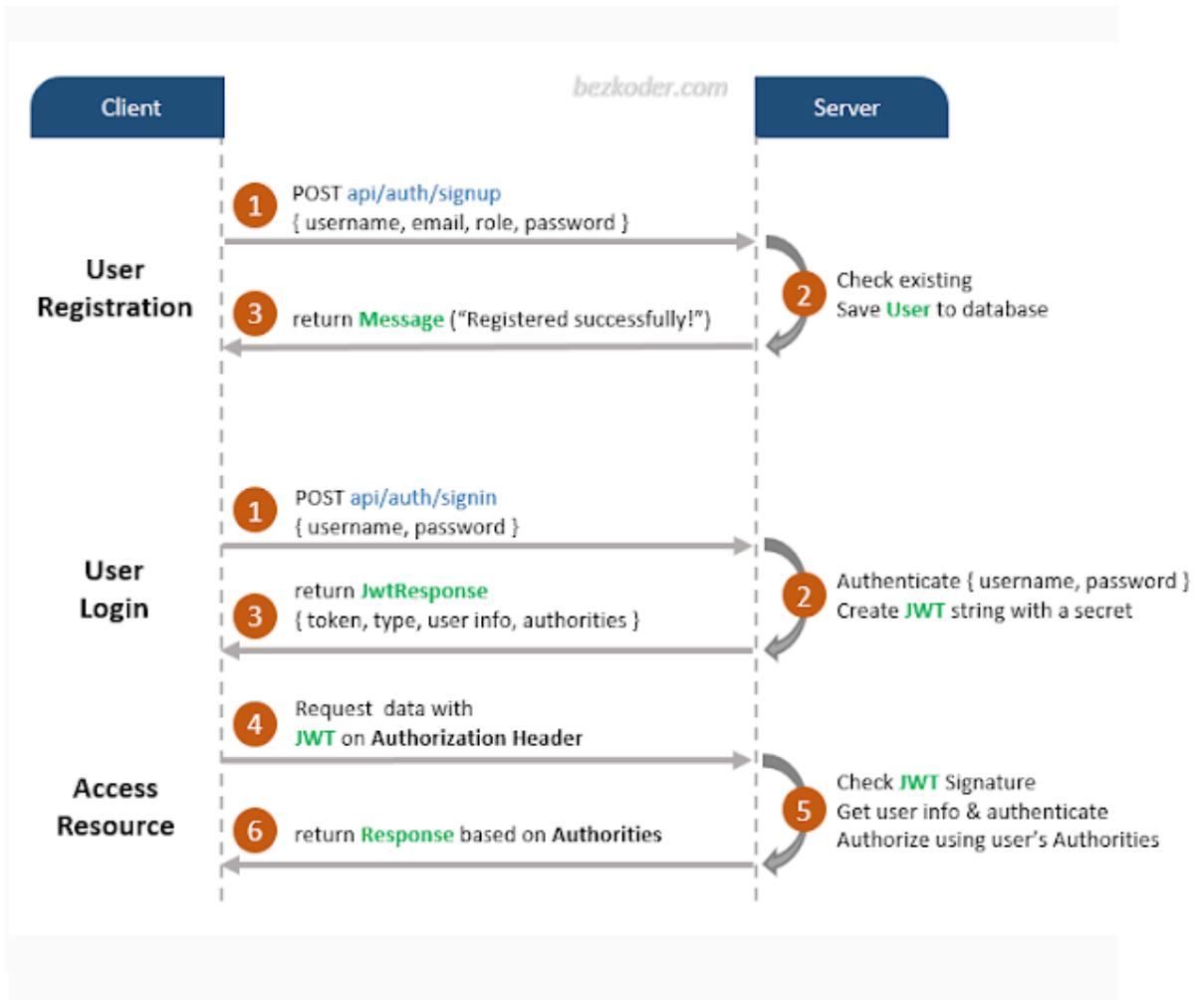
54.	<i>implementasi path api dinamic oauth_user_role</i>	333
a.	db : api url yang di allow	333
b.	output	333
c.	Penerapan	334
55.	<i>Kegunaan JHipster pada api : monitoring</i>	334
a.	Install	334
b.	Create application	335
c.	Chek nvm list untuk melihat version terinstall	337
d.	Cara ganti npm	337
e.	Berhasil dan lakukan jalankan jhipster	337
f.	Run jhispter	338
g.	Pilih microservice	338
56.	<i>Penerapan UUID paa java 1.8 dan 11 keatas</i>	339
57.	<i>Cara Unit Test REST API menggunakan MockMvc</i>	340
a.	Tambahakan @SLF4G di controller	340
b.	Muncul dan create mocktest	340
c.	Dan centang method yang dibutuhkan	341
d.	output	341
e.	Tambahkan manual di class Controller	342
f.	Tambahkan di body	343
g.	Importkan library ini	343
h.	Code ALL	344
i.	Output	350
j.	How to Header in Mock?	350
58.	<i>Spring Cloud Gateway</i>	350
a.	Referensi	350
b.	Analogi	351
c.	Anggapan : kalian sudah punya microservice	351
d.	Step 1 - Create New Project For Config Gateway	351
e.	Step 2- POM.XML	352
f.	Step 3 - SpringCloudConfig	353
g.	Step 4- application.properties	354

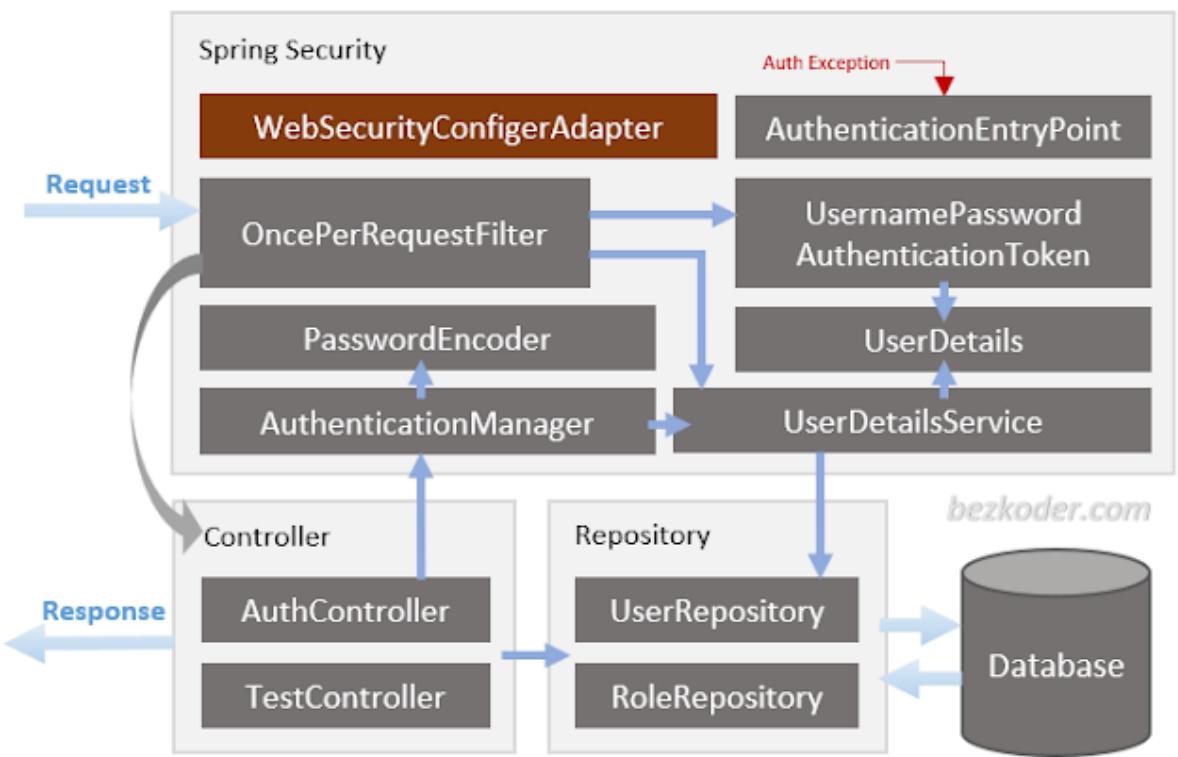
h.	step 5- cloud-gateway-service.iml	354
i.	Output	357
j.	Gitlab	359
59.	Thread	359
a.	Concurrency	360
	Analogi	360
	synchronous VS asynchronous	363
	Synchronous	363
	Asynchronous	363
b.	Parallelism	364
c.	concurrency VS parallelism?	364
d.	implementasi thread di Java?	364
e.	Create Thread	365
f.	Create Thread Dengan Runnable	366
g.	Sleep	367
h.	Join, Notify, Wait	368
i.	Multi Threading with Thread Pool	368
j.	Next lanjut ke slide	368
60.	MultiThread with Thread Pool implementation API	369
a.	Class thread pool	369
b.	Create api thread pool	369
c.	Testing with Jmeter	371
	Download jmeter	371
	documentation jmeter	371
	run jmeter	371
	Tampilan jmeter	372
	Run api- create thread group	372
	Run api – create http request	373
	Run api- add view result tree	373
	Run api	374
	View result	374
d.	How test performance with jmeter 100 user per second	375
	Result view	375
	Daftar Pustaka	377

1. Spring Boot Security

Spring security merupakan fitur dari framework spring. Spring security memungkinkan developer untuk mengintegrasikan fitur keamanan pada aplikasi Java Web dengan cara melakukan *hijacking* pada HTTP request menggunakan filter yang melakukan pengecekan keamanan.

Perhatikan gambar berikut :





Gambar Proses Spring security

Berikut Penjelasan gambar diatas Penjelasan Spring Security

- **WebSecurityConfigurerAdapter** merupakan bagian utama dari implementasi Spring security. Dia menyediakan konfigurasi *HttpSecurity* untuk mengatur *cors*, *csrf*, *session management*, *rules* untuk dapat melindungi berbagai sumberdaya/informasi di aplikasi kita. Kita bisa meng-*extend* dan meng-*customize* konfigurasi bawaannya yang terdiri dari element-element dibawah ini.
- **UserDetailsService** merupakan sebuah interface yang memiliki sebuah *method* untuk mengambil User berdasarkan username dan mengembalikan objek **UserDetails** yang mana Spring Security akan menggunakaninya untuk melakukan *authentication* dan *validation*.
- **UserDetails** berisi informasi yang diperlukan seperti: *username*, *password*, *authorities*, untuk digunakan sebagai *Authentication object*.
- **UsernamePasswordAuthenticationToken** untuk mendapatkan username dan password dari request yang dikirim oleh form login , **AuthenticationManager** akan selanjutnya menggunakan ini untuk meng-*authenticate* akun yang hendak login.

- **AuthenticationManager** memiliki sebuah **DaoAuthenticationProvider** (dibantu oleh **UserDetailsService & PasswordEncoder**) untuk memvalidasi object **UsernamePasswordAuthenticationToken**. Jika berhasil, AuthenticationManager akan mengembalikan data dari object Authentication (termasuk didalamnya granted authorities).
- **OncePerRequestFilter** melakukan sebuah *single execution* untuk setiap request pada API. Ini memberikan sebuah method bernama `doFilterInternal()` method yang akan mengimplementasikan proses parsing & validasi JWT, mengambil User details (menggunakan **UserDetailsService**), mengecek Authorizaion (menggunakan **UsernamePasswordAuthenticationToken**).
- **AuthenticationEntryPoint** untuk menangkap error pada saat authentication.
- **Repository** terdiri dari UserRepository & RoleRepository untuk bekerja dengan Database, akan di import dan digunakan di Controller.
- **Controller** menerima & menangani request setelah request difilter oleh **OncePerRequestFilter**.
- **AuthController** menangani requests di proses signup dan login.
- **TestController** kita gunakan untuk mengecek dan menguji hak akses.

2. Apa perbedaan utama antara otentikasi JWT dan OAuth?

OAuth 2.0 mendefinisikan protokol, yaitu menentukan bagaimana token ditransfer, JWT mendefinisikan format token.

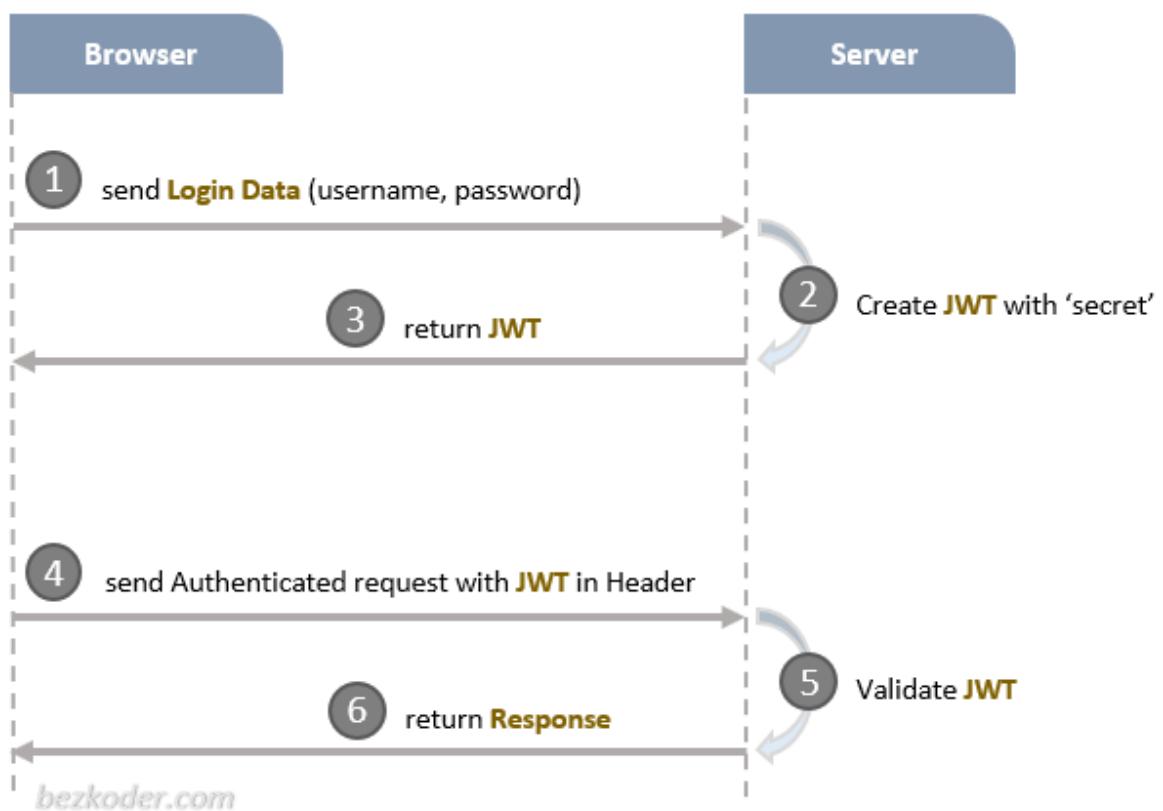
OAuth 2.0 dan "otentikasi JWT" memiliki penampilan yang mirip ketika sampai pada tahap (ke-2) di mana Klien menyajikan token ke Server Sumber Daya: token diteruskan dalam header.

Tetapi "otentikasi JWT" bukan standar dan tidak menentukan bagaimana Klien mendapatkan token di tempat pertama (tahap 1). Di sifullah kompleksitas yang dirasakan dari OAuth berasal: itu juga mendefinisikan berbagai cara di mana Klien dapat memperoleh token akses dari sesuatu yang disebut Server Otorisasi.

Jadi perbedaan sebenarnya adalah bahwa JWT hanyalah format token, OAuth 2.0 adalah protokol (bahwa *boleh* menggunakan JWT sebagai format token).

3. JWT (JSON Web Token)

WT populer untuk Otentikasi dan Pertukaran Informasi. Server mengkodekan data ke dalam JSON Web Token dan mengirimkannya ke Klien. Klien menyimpan JWT, kemudian setiap Permintaan dari Klien ke rute atau sumber daya yang dilindungi harus dilampirkan JWT itu (biasanya di header). Server akan memvalidasi JWT itu dan mengembalikan Response.



The Client typically attach JWT in **Authorization** header with Bearer prefix:

```
Authorization: Bearer [header].[payload].[signature]
```

Pada gambar di atas, ketika pengguna masuk ke situs web, **Server** akan menghasilkan **Session** untuk pengguna itu dan menyimpannya (di Memori atau Basis Data). Server juga mengembalikan a **SessionId** untuk Klien untuk menyimpannya di **Cookie** Browser .

Sesi di Server memiliki waktu kedaluwarsa. Setelah waktu itu, Sesi ini telah kedaluwarsa dan pengguna harus masuk kembali untuk membuat Sesi lain.

Jika pengguna telah masuk dan Sesi belum kedaluwarsa, Cookie (termasuk SessionId) selalu berjalan dengan semua Permintaan HTTP ke Server. Server akan membandingkan ini SessionId dengan yang disimpan Session untuk mengotentikasi dan mengembalikan Respons yang sesuai.

Tidak apa-apa. Tetapi mengapa kita membutuhkan **Otentikasi Berbasis Token**?

Jawabannya adalah kami tidak hanya memiliki situs web, ada banyak platform di sana.

Asumsikan bahwa kita memiliki situs web yang bekerja dengan baik dengan Session. Suatu hari, kami ingin menerapkan sistem untuk Seluler (Aplikasi Asli) dan menggunakan Database yang sama dengan aplikasi Web saat ini. Apa yang harus kita lakukan? Kami tidak dapat mengautentikasi pengguna yang menggunakan Aplikasi Asli menggunakan Otentikasi Berbasis Sesi karena jenis ini tidak memiliki Cookie.

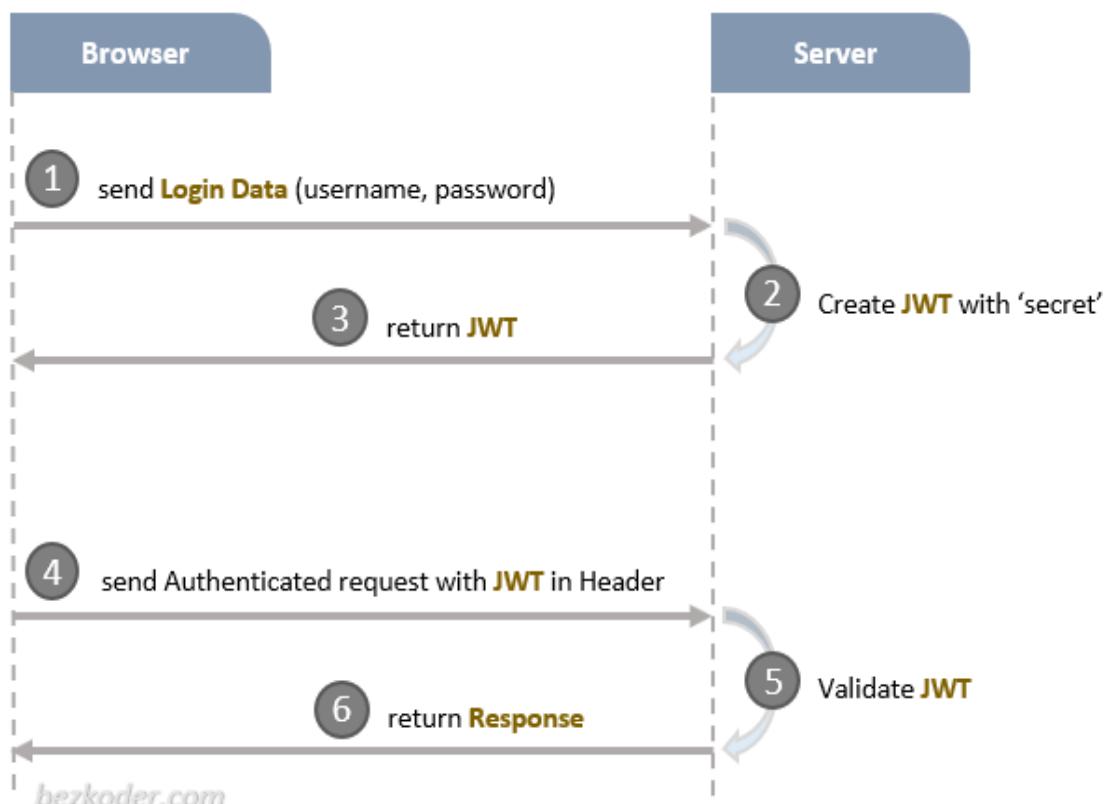
Haruskah kita membangun proyek backend lain yang mendukung Aplikasi Asli?

Atau haruskah kita menulis modul Otentikasi untuk pengguna Aplikasi Asli?

Itu sebabnya **Otentikasi berbasis Token** lahir.

Dengan metode ini, status login pengguna dikodekan ke dalam **JSON Web Token** (JWT) oleh Server dan dikirim ke Klien. Saat ini banyak RESTful API yang menggunakaninya. Mari kita pergi ke bagian berikutnya, kita akan tahu cara kerjanya.

4. Cara kerja JWT



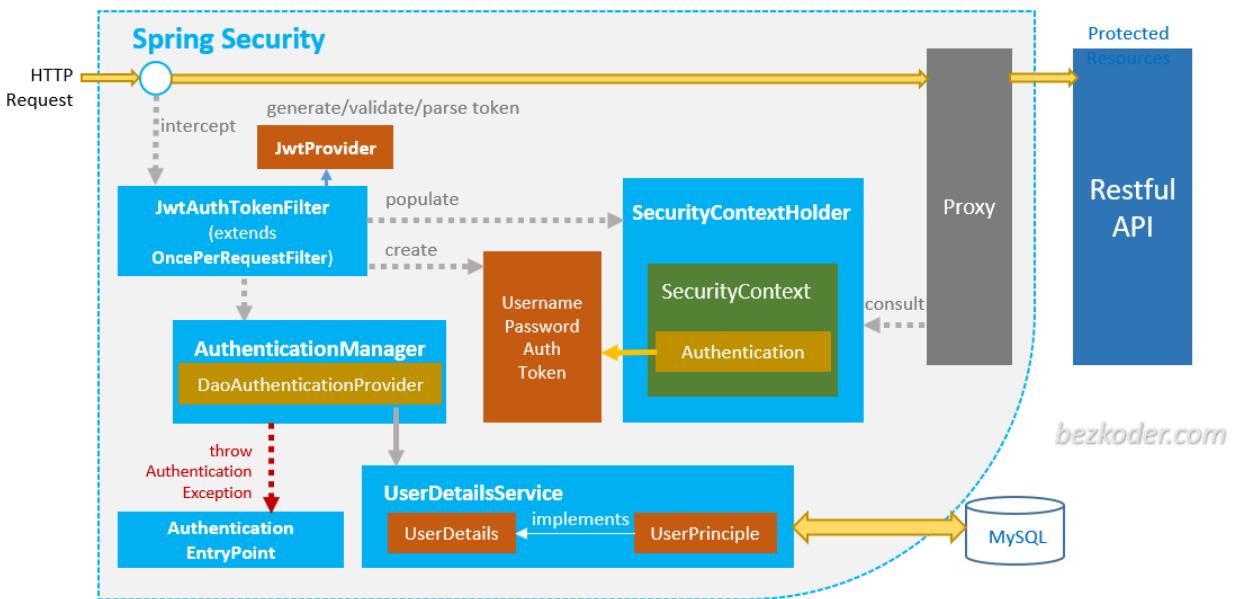
Anda dapat melihat bahwa itu sederhana untuk dipahami. Alih-alih membuat a **Session**, **Server** menghasilkan **JWT** dari data login pengguna dan mengirimkannya ke **Klien**. Klien menyimpan **JWT** dan mulai sekarang, setiap Permintaan dari Klien harus dilampirkan **JWT** (biasanya di **header**). Server akan memvalidasi **JWT** dan mengembalikan Response.

Untuk menyimpan **JWT** di sisi Klien, itu tergantung pada platform yang Anda gunakan:

a. Spring Security JWT in Spring Boot 2

This is diagram for Spring Security/JWT classes that are separated into 3 layers:

- HTTP
- Spring Security
- REST API



Lihat diagram di atas, kita dapat dengan mudah mengaitkan komponen-komponen ini dengan proses Spring Security Authentication: menerima permintaan HTTP, filter, authenticate, store Authentication data, generate token, get User details, authorize, handle exception

At a glance:

- SecurityContextHolder menyediakan akses ke SecurityContext.
- SecurityContext menyimpan Otentikasi dan mungkin meminta informasi keamanan khusus.
- Authentication mewakili prinsipal yang mencakup GrantedAuthority yang mencerminkan izin seluruh aplikasi yang diberikan kepada prinsipal.
- UserDetails berisi informasi yang diperlukan untuk membangun objek Otentikasi dari DAO atau sumber data keamanan lainnya
- JwtAuthTokenFilter (extends OncePerRequestFilter) pra-proses permintaan HTTP, dari Token, buat Otentikasi dan isi ke SecurityContext.
- JwtProvider memvalidasi, mem-parsing token String atau menghasilkan String token dari UserDetails.
- UsernamePasswordAuthenticationToken mendapatkan nama pengguna/kata sandi dari Permintaan masuk dan digabungkan menjadi turunan dari antarmuka Otentikasi.
- AuthenticationManager uses DaoAuthenticationProvider (with help of UserDetailsService & PasswordEncoder) to validate instance

of UsernamePasswordAuthenticationToken, then returns a fully populated Authentication instance on successful authentication.

- SecurityContext is established by calling SecurityContextHolder.getContext().setAuthentication(...) with returned authentication object above.
- AuthenticationEntryPoint handles AuthenticationException.
- Access to Restful API is protected by HTTPSecurity and authorized with Method Security Expressions.

5. Implementasi Spring Security Pada Java Spring Boot

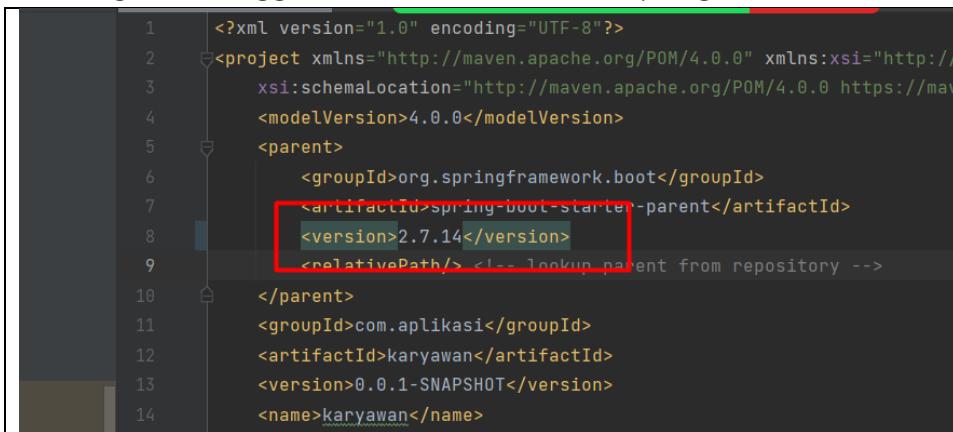
Gitlab : <https://github.com/rikialdi/dibimbing>

Branch : spring-security

Kita ingin menambahkan security dengan jwt token pada project sebelumnya,
Berikut langkah-langkah nya.

a. Pom.XML

Penting , ini menggunakan Version Maven/spring 2.17



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.14</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.aplikasi</groupId>
  <artifactId>karyawan</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>karyawan</name>
```

Silahkan Tambahkan pada pom.xml:

```
<properties>
  <java.version>1.8</java.version>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  <oauth2.version>2.5.2.RELEASE</oauth2.version>
  <jwt.version>1.0.9.RELEASE</jwt.version>
</properties>

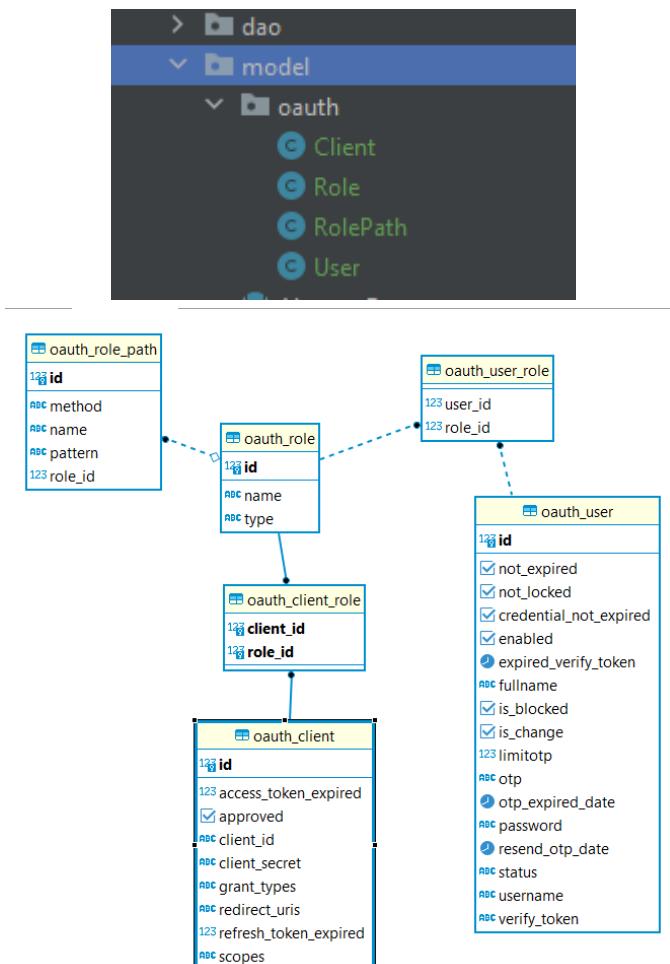
...
<!--      JWT-->
<dependency>

  <groupId>org.springframework.security.oauth</groupId>
  <artifactId>spring-security-oauth2</artifactId>
  <version>${oauth2.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-jwt</artifactId>
  <version>${jwt.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-
```

```
security</artifactId>
      </dependency>
```

b. Entitas/Model

Silahkan tambahkan class dibawah ini :



Gambar Class Entitas

Client

```
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.oauth2.provider.ClientDetails;
import org.springframework.util.StringUtils;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "oauth_client")
public class Client implements ClientDetails, Serializable {

    @Id
```

```

    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String clientId;

    private String clientSecret;

    private String scopes;

    private String grantTypes;

    private String redirectUris;

    private boolean approved;

    @Column(name = "access_token_expired")
    private Integer accessTokenValiditySeconds;

    @Column(name = "refresh_token_expired")
    private Integer refreshTokenValiditySeconds;

    @ManyToMany(targetEntity = Role.class, cascade = CascadeType.ALL,
    fetch = FetchType.EAGER)
    @JoinTable(
        name = "oauth_client_role",
        joinColumns = {
            @JoinColumn(name = "client_id")
        },
        inverseJoinColumns = {
            @JoinColumn(name = "role_id")
        }
    )
    private Set<GrantedAuthority> authorities = new HashSet<>();

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    @Override
    public String getClientId() {
        return clientId;
    }

    public void setClientId(String clientId) {
        this.clientId = clientId;
    }

    @Override
    public Set<String> getResourceIds() {
        Set<String> resources = new HashSet<>();
        resources.add("oauth2-resource");

        return resources;
    }

```

```

@Override
public boolean isSecretRequired() {
    return !StringUtils.isEmpty(clientSecret);
}

public void setClientSecret(String clientSecret) {
    this.clientSecret = clientSecret;
}

@Override
public String getClientSecret() {
    return clientSecret;
}

public String getScopes() {
    return scopes;
}

public void setScopes(String scopes) {
    this.scopes = scopes;
}

@Override
public boolean isScoped() {
    return !StringUtils.isEmpty(scopes);
}

@Override
public Set<String> getScope() {
    Set<String> scope = new HashSet<>();

    if (isScoped()) {
        scope = new
HashSet<>(Arrays.asList(scopes.split("\\" s")));
    }

    return scope;
}

public String getGrantTypes() {
    return grantTypes;
}

public void setGrantTypes(String grantTypes) {
    this.grantTypes = grantTypes;
}

@Override
public Set<String> getAuthorizedGrantTypes() {
    if (null != grantTypes) {
        return new
HashSet<>(Arrays.asList(grantTypes.split("\\" s)));
    }
    return null;
}

@Override
public Set<GrantedAuthority> getAuthorities() {
    return authorities;
}

```

```

    }

    public void setAuthorities(Set<GrantedAuthority> authorities) {
        this.authorities = authorities;
    }

    public String getRedirectUris() {
        return redirectUris;
    }

    public void setRedirectUris(String redirectUris) {
        this.redirectUris = redirectUris;
    }

    @Override
    public Set<String> getRegisteredRedirectUri() {
        if (null != redirectUris) {
            return new
        HashSet<>(Arrays.asList(redirectUris.split("\\s")));
        }
        return null;
    }

    @Override
    public Integer getAccessTokenValiditySeconds() {
        return accessTokenValiditySeconds;
    }

    public void setAccessTokenValiditySeconds(Integer
accessTokenValiditySeconds) {
        this.accessTokenValiditySeconds = accessTokenValiditySeconds;
    }

    @Override
    public Integer getRefreshTokenValiditySeconds() {
        return refreshTokenValiditySeconds;
    }

    public void setRefreshTokenValiditySeconds(Integer
refreshTokenValiditySeconds) {
        this.refreshTokenValiditySeconds =
refreshTokenValiditySeconds;
    }

    public boolean isApproved() {
        return approved;
    }

    public void setApproved(boolean approved) {
        this.approved = approved;
    }

    @Override
    public boolean isAutoApprove(String s) {
        return approved;
    }

    @Override
    public Map<String, Object> getAdditionalInformation() {

```

```
        return new HashMap<>();
    }
}
```

Role

```
import com.fasterxml.jackson.annotation.JsonIgnore;
import org.springframework.security.core.GrantedAuthority;

import javax.persistence.*;
import java.util.List;

@Entity()
@Table(
    name = "oauth_role",
    uniqueConstraints = {
        @UniqueConstraint(
            name = "role_name_and_type",
            columnNames = {"type", "name"}
        )
    }
)
public class Role implements GrantedAuthority {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = 20)
    private String name;

    private String type;

    @OneToMany(mappedBy = "role", fetch = FetchType.LAZY, cascade =
    CascadeType.ALL)
    private List<RolePath> rolePaths;

    @JsonIgnore
    @ManyToMany(targetEntity = User.class, mappedBy = "roles", fetch =
    FetchType.LAZY)
    private List<User> users;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
```

```

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    @Override
    @JsonIgnore
    public String getAuthority() {
        return this.name;
    }

    public List<RolePath> getRolePaths() {
        return rolePaths;
    }

    public void setRolePaths(List<RolePath> rolePaths) {
        this.rolePaths = rolePaths;
    }

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }
}

```

Role Path

```

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.*;
import java.io.Serializable;

@Entity
@Table(name = "oauth_role_path")
public class RolePath implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = 50)
    private String name;

    private String pattern;
}

```

```

private String method;

@ManyToOne(targetEntity = Role.class, cascade = CascadeType.ALL)
@JsonIgnore
private Role role;

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getPattern() {
    return pattern;
}

public void setPattern(String pattern) {
    this.pattern = pattern;
}

public String getMethod() {
    return method;
}

public void setMethod(String method) {
    this.method = method;
}

public Role getRole() {
    return role;
}

public void setRole(Role role) {
    this.role = role;
}

```

User

```

import com.fasterxml.jackson.annotation.JsonIgnore;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

```

```

import javax.persistence.*;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Date;
import java.util.List;

@Entity
@Table(name = "oauth_user")
public class User implements UserDetails, Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = 100, unique=true)
    private String username;

    @Column(length = 100, nullable = true)
    private String fullname;

    @JsonIgnore
    private String password;

    @JsonIgnore
    private String verifyToken;

    @JsonIgnore
    private Date expiredVerifyToken;

    @Column(length = 100, nullable = true)
    private String otp;

    private Date otpExpiredDate;

    @JsonIgnore
    private boolean enabled = true;

    @JsonIgnore
    @Column(name = "not_expired")
    private boolean accountNonExpired = true;

    @JsonIgnore
    @Column(name = "not_locked")
    private boolean accountNonLocked = true;

    @JsonIgnore
    @Column(name = "credential_not_expired")
    private boolean credentialsNonExpired = true;

    @ManyToMany(targetEntity = Role.class, cascade = CascadeType.ALL,
    fetch = FetchType.EAGER)
    @JoinTable(
        name = "oauth_user_role",
        joinColumns = {
            @JoinColumn(name = "user_id")
        },

```

```

        inverseJoinColumns = {
            @JoinColumn(name = "role_id")
        }
    }
private List<Role> roles = new ArrayList<>();

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public List<Role> getRoles() {
    return roles;
}

public void setRoles(List<Role> roles) {
    this.roles = roles;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return this.roles;
}

@Override
public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public void setUsername(String username) {
    this.username = username;
}

@Override
public String getUsername() {
    return username;
}

@Override
public boolean isAccountNonExpired() {
    return accountNonExpired;
}

public void setAccountNonExpired(boolean accountNonExpired) {
    this.accountNonExpired = accountNonExpired;
}

@Override
public boolean isAccountNonLocked() {
    return accountNonLocked;
}

```

```
public void setAccountNonLocked(boolean accountNonLocked) {
    this.accountNonLocked = accountNonLocked;
}

@Override
public boolean isCredentialsNonExpired() {
    return credentialsNonExpired;
}

public void setCredentialsNonExpired(boolean credentialsNonExpired) {
    this.credentialsNonExpired = credentialsNonExpired;
}

public void setEnabled(boolean enabled) {
    this.enabled = enabled;
}

@Override
public boolean isEnabled() {
    return enabled;
}

public String getFullscreen() {
    return fullname;
}

public void setFullscreen(String fullname) {
    this.fullname = fullname;
}

public String getVerifyToken() {
    return verifyToken;
}

public void setVerifyToken(String verifyToken) {
    this.verifyToken = verifyToken;
}

public Date getExpiredVerifyToken() {
    return expiredVerifyToken;
}

public void setExpiredVerifyToken(Date expiredVerifyToken) {
    this.expiredVerifyToken = expiredVerifyToken;
}

public String getOtp() {
    return otp;
}

public void setOtp(String otp) {
    this.otp = otp;
}

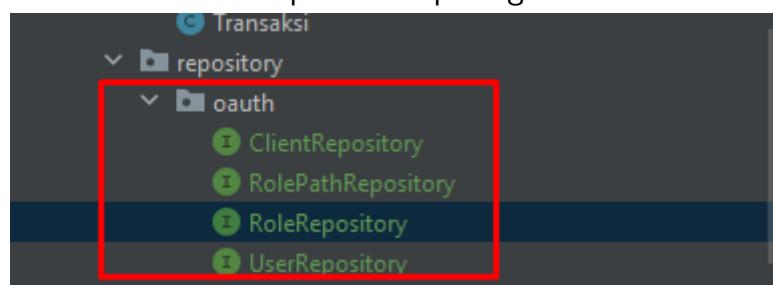
public Date getOtpExpiredDate() {
    return otpExpiredDate;
}
```

```
    }

    public void setOtpExpiredDate(Date otpExpiredDate) {
        this.otpExpiredDate = otpExpiredDate;
    }
}
```

c. Repository

Silahkan tambahkan class repositori seperti gambar dibawah ini :



Gambar Class Repository

ClientRepository

```
import com.dibimbing.dibimbing.model.oauth.Client;
import org.springframework.data.repository.PagingAndSortingRepository;

public interface ClientRepository extends
PagingAndSortingRepository<Client, Long> {

    Client findOneByClientId(String clientId);
}
```

RolePathRepository

```
import com.dibimbing.dibimbing.model.oauth.RolePath;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.List;

public interface RolePathRepository extends
```

```

PagingAndSortingRepository<RolePath, Long> {
    RolePath findOneByName(String rolePathName);

    @Query(value = "SELECT p.* FROM oauth_role_path p " +
        "JOIN oauth_role r ON r.id = p.role_id " +
        "JOIN oauth_user_role ur ON ur.role_id = r.id " +
        "WHERE ur.user_id = ?1", nativeQuery = true)
    <T extends UserDetails> List<RolePath> findByUser(T user);
}

```

RoleRepository

```

import com.dibimbing.dibimbing.model.oauth.Role;
import org.springframework.data.repository.PagingAndSortingRepository;

import java.util.List;

public interface RoleRepository extends
PagingAndSortingRepository<Role, Long> {
    Role findOneByName(String name);

    List<Role> findByNameIn(String[] names);
}

```

UserRepository

```

import com.dibimbing.dibimbing.model.oauth.User;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;

public interface UserRepository extends
PagingAndSortingRepository<User, Long> {
    @Query("FROM User u WHERE LOWER(u.username) = LOWER(?1)")
    User findOneByUsername(String username);

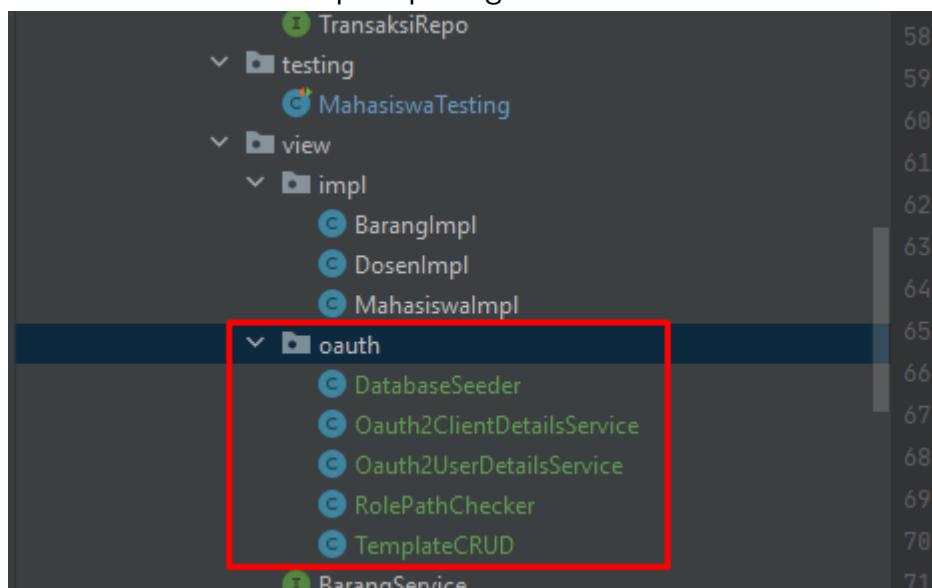
    @Query("FROM User u WHERE u.otp = ?1")
    User findOneByOTP(String otp);

    @Query("FROM User u WHERE LOWER(u.username) = LOWER(:username)")
    User checkExistingEmail(String username);
}

```

d. Service

Tambahkan class service seperti pada gambar dibawah ini :



Gambar Service Oauth

DatabaseSeeder

```
import com.dibimbng.dibimbng.model.oauth.Client;
import com.dibimbng.dibimbng.model.oauth.Role;
import com.dibimbng.dibimbng.model.oauth.RolePath;
import com.dibimbng.dibimbng.model.oauth.User;
import com.dibimbng.dibimbng.repository.oauth.ClientRepository;
import com.dibimbng.dibimbng.repository.oauth.RolePathRepository;
import com.dibimbng.dibimbng.repository.oauth.RoleRepository;
import com.dibimbng.dibimbng.repository.oauth.UserRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.ArrayList;
import java.util.List;

@Component
@Service
public class DatabaseSeeder implements ApplicationRunner {

    private static final String TAG = "DatabaseSeeder {}";
```

```

    private Logger logger =
LoggerFactory.getLogger(DatabaseSeeder.class);

    @Autowired
    private PasswordEncoder encoder;

    @Autowired
    private RoleRepository roleRepository;

    @Autowired
    private ClientRepository clientRepository;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private RolePathRepository rolePathRepository;

    private String defaultPassword = "password";

    private String[] users = new String[]{
        "admin@mail.com:ROLE_SUPERUSER ROLE_USER ROLE_ADMIN",
        "user@mail.com:ROLE_USER"
    };

    private String[] clients = new String[]{
        "my-client-apps:ROLE_READ ROLE_WRITE", // mobile
        "my-client-web:ROLE_READ ROLE_WRITE" // web
    };

    private String[] roles = new String[] {

"ROLE_SUPERUSER:user_role:^/.*:GET|PUT|POST|PATCH|DELETE|OPTIONS",
"ROLE_ADMIN:user_role:^/.*:GET|PUT|POST|PATCH|DELETE|OPTIONS",
"ROLE_USER:user_role:^/.*:GET|PUT|POST|PATCH|DELETE|OPTIONS",
"ROLE_READ:oauth_role:^/.*:GET|PUT|POST|PATCH|DELETE|OPTIONS",
"ROLE_WRITE:oauth_role:^/.*:GET|PUT|POST|PATCH|DELETE|OPTIONS"
};

    @Override
    @Transactional
    public void run(ApplicationArguments applicationArguments) {
        String password = encoder.encode(defaultPassword);

        this.insertRoles();
        this.insertClients(password);
        this.insertUser(password);
    }

    @Transactional
    public void insertRoles() {
        for (String role: roles) {
            String[] str = role.split(":");
            String name = str[0];

```

```

        String type = str[1];
        String pattern = str[2];
        String[] methods = str[3].split("\\\\|");
        Role oldRole = roleRepository.findOneByName(name);
        if (null == oldRole) {
            oldRole = new Role();
            oldRole.setName(name);
            oldRole.setType(type);
            oldRole.setRolePaths(new ArrayList<>());
            for (String m: methods) {
                String rolePathName =
name.toLowerCase()+"_"+m.toLowerCase();
                RolePath rolePath =
rolePathRepository.findOneByName(rolePathName);
                if (null == rolePath) {
                    rolePath = new RolePath();
                    rolePath.setName(rolePathName);
                    rolePath.setMethod(m.toUpperCase());
                    rolePath.setPattern(pattern);
                    rolePath.setRole(oldRole);
                    rolePathRepository.save(rolePath);
                    oldRole.getRolePaths().add(rolePath);
                }
            }
        }
        roleRepository.save(oldRole);
    }
}

@Transactional
public void insertClients(String password) {
    for (String c: clients) {
        String[] s = c.split(":");
        String clientName = s[0];
        String[] clientRoles = s[1].split("\\\\s");
        Client oldClient =
clientRepository.findOneByClientId(clientName);
        if (null == oldClient) {
            oldClient = new Client();
            oldClient.setClientId(clientName);
            oldClient.setAccessTokenValiditySeconds(28800); //1
jam 3600 :token valid : seharan kerja : normal 1 jam
            oldClient.setRefreshTokenValiditySeconds(7257600); //refresh
            oldClient.setGrantTypes("password refresh_token
authorization_code");
            oldClient.setClientSecret(password);
            oldClient.setApproved(true);
            oldClient.setRedirectUris("");
            oldClient.setScopes("read write");
            List<Role> rls =
roleRepository.findByNameIn(clientRoles);

            if (rls.size() > 0) {
                oldClient.getAuthorities().addAll(rls);
            }
        }
        clientRepository.save(oldClient);
    }
}

```

```

        }

    }

    @Transactional
    public void insertUser(String password) {
        for (String userNames: users) {
            String[] str = userNames.split(":");
            String username = str[0];
            String[] roleNames = str[1].split("\\");

            User oldUser =
userRepository.findOneByUsername(username);
            if (null == oldUser) {
                oldUser = new User();
                oldUser.setUsername(username);
                oldUser.setPassword(password);
                List<Role> r =
roleRepository.findByNameIn(roleNames);
                oldUser.setRoles(r);
            }

            userRepository.save(oldUser);
        }
    }
}

```

Oauth2ClientDetailsService

```

import com.dibimbing.dibimbing.repository.oauth.ClientRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.security.oauth2.provider.ClientDetails;
import
org.springframework.security.oauth2.provider.ClientDetailsService;
import
org.springframework.security.oauth2.provider.ClientRegistrationException;
import org.springframework.stereotype.Service;

@Service
public class Oauth2ClientDetailsService implements
ClientDetailsService {

    @Autowired
    private ClientRepository clientRepository;

    @Override
    public ClientDetails loadClientByClientId(String s) throws
ClientRegistrationException {
        ClientDetails client = clientRepository.findOneByClientId(s);
        if (null == client) {
            throw new ClientRegistrationException("Client not
found");
        }
    }
}

```

```

        return client;
    }

    @CacheEvict("oauth_client_id")
    public void clearCache(String s) {
        System.out.println("ini cache oauth_client_id= "+s);
    }
}

```

Oauth2UserDetailsService

```

import com.dibimbing.dibimbing.model.oauth.User;
import com.dibimbing.dibimbing.repository.oauth.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class Oauth2UserDetailsService implements UserDetailsService{

    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String s) throws
UsernameNotFoundException {
        User user = userRepository.findOneByUsername(s);
        if (null == user) {
            throw new
UsernameNotFoundException(String.format("Username %s is not found",
s));
        }
        return user;
    }

    @CacheEvict("oauth_username")
    public void clearCache(String s) {
    }
}

```

TemplateCRUD

```
import com.dibimbng.dibimbng.config.Config;

import java.util.HashMap;
import java.util.Map;

public class TemplateCRUD {
    Config config = new Config();
    public Map template1(Object obj)  {
        Map map = new HashMap();
        try {
            map.put("data",obj );
            map.put(config.getCode(), config.code_sukses);
            map.put(config.getMessage(), config.message_sukses);
            return map;
        } catch (Exception e) {
            System.out.println("eror template1 =" +e);
            map.put(config.getCode(), config.code_server);
            map.put(config.getMessage(), e.getMessage());
            return map;
        }
    }

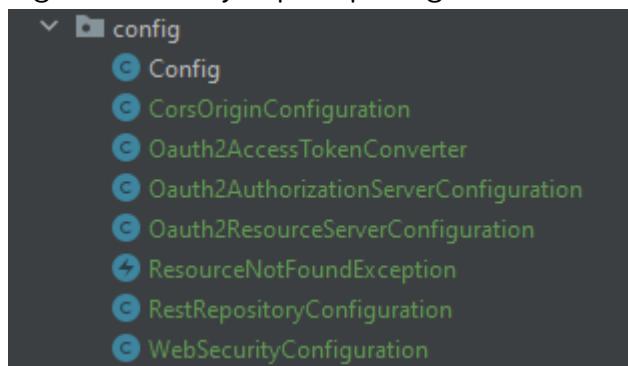
    public Map template1Error(String obj)  {
        Map map = new HashMap();
        map.put(config.getCode(), config.code_server);
        map.put(config.getMessage(), obj);
        return map;
    }

    public Map notFound(String obj)  {
        Map map = new HashMap();
        map.put(config.getCode(), config.code_notFound);
        map.put(config.getMessage(), obj);
        return map;
    }

    public MapisRequired(String obj)  {
        Map map = new HashMap();
        map.put(config.getCode(), config.code_notFound);
        map.put(config.getMessage(), obj);
        return map;
    }
}
```

e. Configurasi Security

Tambahkan konfigurasi security seperti pada gambar dibawah ini :



Gambar Security Konfigurasi Spring Boot

CorsOriginConfiguration

```
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@SuppressWarnings("SpellCheckingInspection")
@Order(Ordered.HIGHEST_PRECEDENCE)
@Component
public class CorsOriginConfiguration implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
    }

    @Override
    public void destroy() {
    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse res,
FilterChain filter) throws IOException, ServletException {
        HttpServletResponse response = (HttpServletResponse) res;
        HttpServletRequest request = (HttpServletRequest) req;
        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Access-Control-Allow-Methods", "POST,
GET, PUT, PATCH, OPTIONS, DELETE");
        response.setHeader("Access-Control-Max-Age", "3600");
    }
}
```

```

        response.setHeader("Access-Control-Allow-Headers", "Origin,
X-Requested-With, Content-Type, Accept, Authorization, Cache-Control,
X-Auth-Token, Data");
//        response.setHeader("Access-Control-Allow-Headers", "*");

        if ("OPTIONS".equalsIgnoreCase(request.getMethod())) {
            response.setStatus(HttpServletResponse.SC_OK);
        } else {
            filter.doFilter(req, res);
        }
    }
}

```

Oauth2AccessTokenConverter

```

import com.dibimbing.dibimbing.repository.oauth.UserRepository;
import com.dibimbing.dibimbing.service.oauth.Oauth2UserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.GrantedAuthority;
import org.springframework.core.userdetails.UserDetails;
import org.springframework.security.oauth2.provider.OAuth2Authentication;
import org.springframework.security.oauth2.provider.token.DefaultAccessTokenConverter;
import org.springframework.stereotype.Component;

import java.util.Collection;
import java.util.Map;

@Component
@SuppressWarnings("unchecked")
public class Oauth2AccessTokenConverter extends DefaultAccessTokenConverter {
    @Autowired
    private UserRepository userRepository;

    @Autowired
    private Oauth2UserDetailsService userDetailsService;

    @Override
    public OAuth2Authentication extractAuthentication(Map<String, ?>
map) {
        final OAuth2Authentication auth =
super.extractAuthentication(map);
        final UserDetails user =
userDetailsService.loadUserByUsername((String) auth.getPrincipal());
        return new OAuth2Authentication(auth.getOAuth2Request(),
auth.getUserAuthentication());
    }

    @Override
    public Collection<GrantedAuthority> getAuthorities() {
        if (user != null) {
            return (Collection<GrantedAuthority>)
}
}

```

```
        user.getAuthorities();
    }

    return auth.getAuthorities();
}

};

}

}
```

Oauth2AuthorizationServerConfiguration

```
import com.dibimbng.dibimbng.service.oauth.Oauth2ClientDetailsService;
import com.dibimbng.dibimbng.service.oauth.Oauth2UserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.oauth2.config.annotation.configurers.ClientDetailsServiceConfigurer;
import org.springframework.security.oauth2.config.annotation.web.configuration.AuthorizationServerConfigurerAdapter;
import org.springframework.security.oauth2.config.annotation.web.configuration.EnableAuthorizationServer;
import org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerEndpointsConfigurer;
import org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerSecurityConfigurer;
import org.springframework.security.oauth2.provider.token.AccessTokenConverter;
import org.springframework.security.oauth2.provider.token.TokenStore;

@Configuration
@EnableAuthorizationServer
public class Oauth2AuthorizationServerConfiguration extends AuthorizationServerConfigurerAdapter {

    @Autowired
    private Oauth2ClientDetailsService clientDetailsService;

    @Autowired
    private Oauth2UserDetailsService userDetailsService;

    @Autowired
    private AuthenticationManager authenticationManager;
```

```

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private AccessTokenConverter accessTokenConverter;

    @Autowired
    private TokenStore tokenStore;

    /**
     * Change server config, password encoder etc.
     */
    @Override
    public void configure(AuthorizationServerSecurityConfigurer
server) throws Exception {
        server.allowFormAuthenticationForClients()
//            .tokenKeyAccess("permitAll()")
//            .checkTokenAccess("isAuthenticated()")
            .passwordEncoder(passwordEncoder)
        ;
    }

    /**
     * Change client details etc.
     */
    @Override
    public void configure(ClientDetailsServiceConfigurer clients)
throws Exception {
        clients.withClientDetails(clientDetailsService);
    }

    /**
     * Change user details etc.
     */
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer
endpoints) throws Exception {
        endpoints.authenticationManager(authenticationManager)
            .tokenStore(tokenStore)
            .accessTokenConverter(accessTokenConverter)
            .userDetailsService(userDetailsService)
        ;
    }
}

```

Oauth2ResourceServerConfiguration

```

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;

```

```

import
org.springframework.security.oauth2.config.annotation.web.configuration.EnableResourceServer;
import
org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.configurers.ResourceServerSecurityConfigurer;

@Configuration
@EnableResourceServer
@EnableGlobalMethodSecurity(securedEnabled = true) //secure
definition
public class OAuth2ResourceServerConfiguration extends
ResourceServerConfigurerAdapter {

    /**
     * Manage resource server.
     */
    @Override
    public void configure(ResourceServerSecurityConfigurer resources)
throws Exception {
        super.configure(resources);
    }
//    private static final String SECURED_PATTERN = "/api/**";
    /**
     * Manage endpoints.
     */
    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.cors()
            .and()
            .csrf()
            .disable()
            .antMatcher("/**")
            .authorizeRequests()

.antMatchers("/", "/showFile/**", "/v1/showFile/**", "/v1/upload",
"/user-register/**", "/swagger-ui/**", "/swagger-ui.html", "/v3/api-
docs/**", "/user-login/**",
                    "/forget-password/**",
"/oauth/authorize**", "/login**", "/error**")
            .permitAll()
            .antMatchers("/v1/role-test-global/list-
barang").hasAnyAuthority("ROLE_READ")
            .antMatchers("/v1/role-test-global/post-
barang").hasAnyAuthority("ROLE_WRITE")
            .antMatchers("/v1/role-test-global/post-barang-
user").hasAnyAuthority("ROLE_USER")
            .antMatchers("/v1/role-test-global/post-barang-
admin").hasAnyAuthority("ROLE_ADMIN")
            .and()
            .authorizeRequests()
                .anyRequest()
                .authenticated()
            .and()
                .formLogin()
                .permitAll()
}

```

```
    }
}
```

WebSecurityConfiguration

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.oauth2.provider.token.AccessTokenConverter;
import
org.springframework.security.oauth2.provider.token.DefaultAccessTokenConverter;
import
org.springframework.security.oauth2.provider.token.DefaultTokenServices;
import org.springframework.security.oauth2.provider.token.TokenStore;
import
org.springframework.security.oauth2.provider.token.store.InMemoryTokenStore;
import
org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;
import
org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

import javax.annotation.Priority;

@SuppressWarnings("SpellCheckingInspection")
@Priority(1)
@Configuration
@EnableWebSecurity
public class WebSecurityConfiguration extends
WebSecurityConfigurerAdapter {

    @Value("${security.bcrypt.cost:13}")
    private int cost;
```

```

    @Value("${security.jwt.enabled:false}")
    private boolean jwtEnabled;

    @Value("${security.jwt.secret_key:s3cr3t}")
    private String jwtSecretKey;

    @Autowired
    private OAuth2AccessTokenConverter accessTokenConverter;

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(cost);
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws
Exception {
        return super.authenticationManagerBean();
    }

    @Bean
    public TokenStore tokenStore() {
        if (jwtEnabled) {
            return new JwtTokenStore((JwtAccessTokenConverter)
accessTokenConverter());
        }
        return new InMemoryTokenStore();
    }

    @Bean
    public AccessTokenConverter accessTokenConverter() {
        if (jwtEnabled) {
            JwtAccessTokenConverter jwtConverter = new
JwtAccessTokenConverter();

            jwtConverter.setAccessTokenConverter(accessTokenConverter);
            jwtConverter.setSigningKey(jwtSecretKey);

            return jwtConverter;
        }
        return new DefaultAccessTokenConverter();
    }

    @Bean
    @Primary
    public DefaultTokenServices tokenServices() {
        DefaultTokenServices services = new DefaultTokenServices();
        services.setTokenStore(tokenStore());
        services.setSupportRefreshToken(true);

        return services;
    }
}

```

Config

```
package com.binar5.apps.config;

import lombok.Data;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

@Data
public class Config {
    String code = "status", message = "message";
    public String codeNotFound ="404";

    public String codeRequired ="403";
    public String isRequired =" is Required";

    public String code_sukses = "200";
    public String code_server = "500";
    public String code_null = "1";
    public String message_sukses = "sukses";

    public String convertDateToString(Date date) {

        DateFormat dateFormat = new
SimpleDateFormat("yyyyMMddHHmmss");
        String strDate = dateFormat.format(date);
        return strDate;
    }
}
```

f. Application.properties

```
security.jwt.enabled=true
```

g. Run and Testing with Postman

Testing by postman hit login and get token jwt

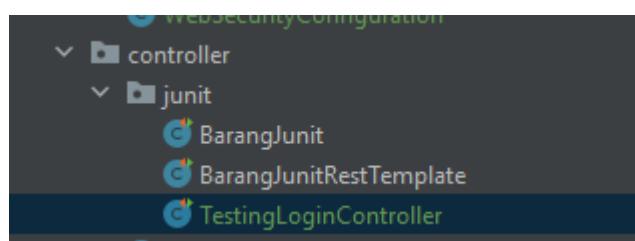
The screenshot shows a Postman request to `localhost:8081/api/oauth/token`. The body is set to `x-www-form-urlencoded` and contains the following parameters:

username	admin@mail.com
password	password
grant_type	password
client_id	my-client-apps
client_secret	password

The response status is 200 OK, and the JSON body includes an access token.

```
1 "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
2 eyJhdWQiOlsib2F1dGgyLXJlc291cmNlIi0sInVzZXJfbmFtZSI6ImFkbWluQG1haWwuY29tIiwigC  
3 I6MTY0MDczNzIxMywiYXV0aG9yaXRpZXMiOlsiUk9MRV9TVVFU1VTRViLCJST0xFX0FETU10IiwidUk9MRV9VU0VS  
4 I0sImp0aSI6IjdiZDU4MzM4LTdjMWEtNDBjNi04ZDY2LWI4ZTU5MzQwZGJjNSIsImNsawVudF9pZCI6Im15LWNsaW  
"token_type": "bearer",  
"refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
```

Testing JUnit :Login



```
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;  
import org.springframework.boot.test.web.client.TestRestTemplate;  
import org.springframework.http.*;  
import org.springframework.test.context.junit4.SpringRunner;  
import org.springframework.util.LinkedMultiValueMap;  
import org.springframework.util.MultiValueMap;  
  
import static org.junit.Assert.assertEquals;  
  
@RunWith(SpringRunner.class)  
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)  
public class TestingLoginController {  
    @Autowired  
    private TestRestTemplate restTemplate;
```

```
@Test
public void restTemplateLogin() throws Exception {
    HttpHeaders headers = new HttpHeaders();

    headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);

    MultiValueMap<String, String> map = new
LinkedMultiValueMap<>();
    map.add("username", "admin@mail.com");
    map.add("password", "password");
    map.add("grant_type", "password");
    map.add("client_id", "my-client-web");
    map.add("client_secret", "password");

    HttpEntity<MultiValueMap<String, String>> request = new
HttpEntity<MultiValueMap<String, String>>(map, headers);

    ResponseEntity<String> response =
restTemplate.postForEntity("http://localhost:9091/api/oauth/token",
request, String.class);

    assertEquals(HttpStatus.OK, response.getStatusCode());
    System.out.println("response =" + response.getBody());
}
}
```

Cara menggunakan token jwt

GET

Params Authorization Headers (11) Body Pre-request Script Tests Settings

<input checked="" type="checkbox"/> Accept	①	/*
<input checked="" type="checkbox"/> Accept-Encoding	①	gzip, deflate, br
<input checked="" type="checkbox"/> Connection	①	keep-alive
<input checked="" type="checkbox"/> Authorization		Bearer eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.e...
<input type="checkbox"/> Content-Type		application/x-www-form-urlencoded

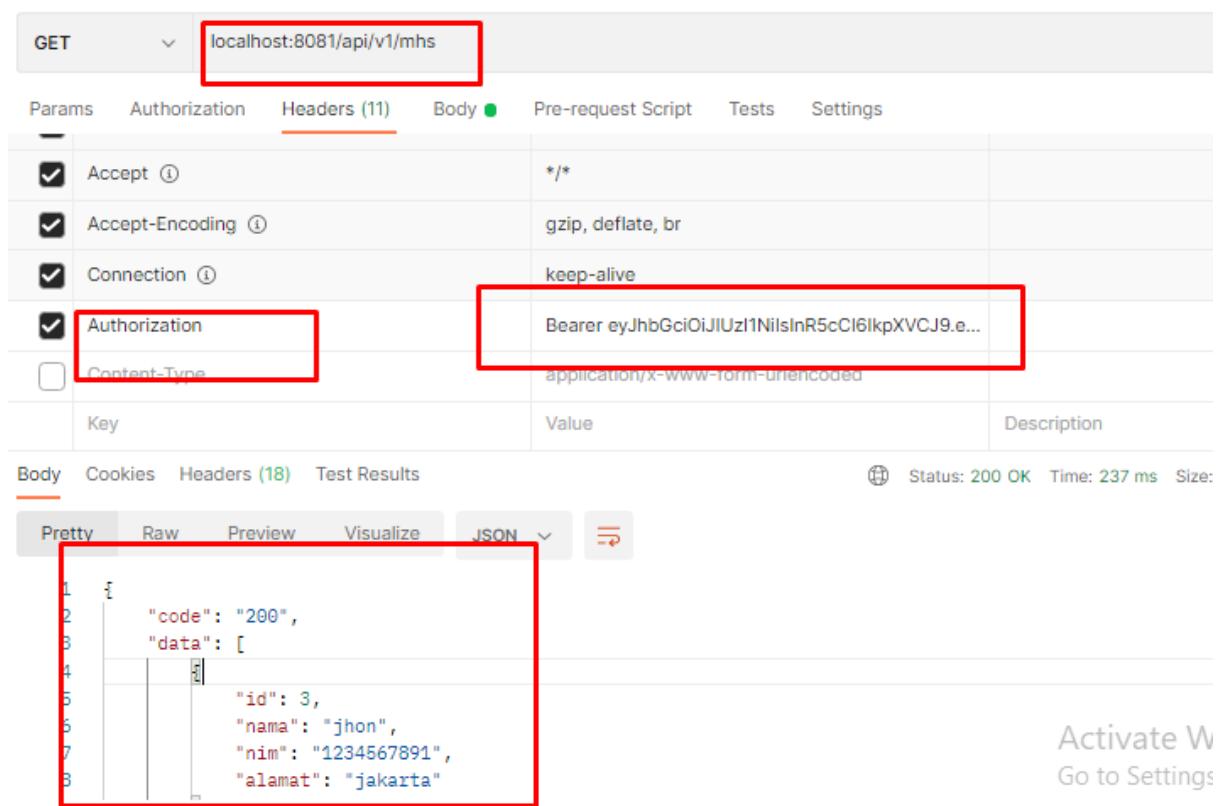
Key Value Description

Body Cookies Headers (18) Test Results Status: 200 OK Time: 237 ms Size:

Pretty Raw Preview Visualize JSON

```
1  {
2    "code": "200",
3    "data": [
4      {
5        "id": 3,
6        "nama": "jhon",
7        "nim": "1234567891",
8        "alamat": "jakarta"
9      }
10   ]
11 }
```

Activate W
Go to Settings



h. Github Project

Silahkan pull dan download project pada url berikut : Gitlab :

<https://github.com/rikialdi/dibimbing>

Branch : spring-security

Binar

211122-spring-security

<https://gitlab.com/rikialdi/binar5-spring.git>

i. Login form security default

Password: diambil dari terminal generate otomatis

User : user

```
2022-11-21 20:05:00.372  WARN 5220 --- [           main] .s.s.UserDetailsS...  
Using generated security password: 095069dd-73ce-44af-a7b0-dc1312fecc11  
  
This generated password is for development use only. Your security configu...
```

6. Mengatur Hak Akses/ROLE pada Akses Endpoint REST API

Ada banyak cara untuk mengurnya: perhatikan cara-cara berikut ini

a. Role dengan : Endpoint Global

Buat endpoint beserta role yang diberikan



```
...  
.antMatchers("/v1/role-test-global/list-barang")  
...
```

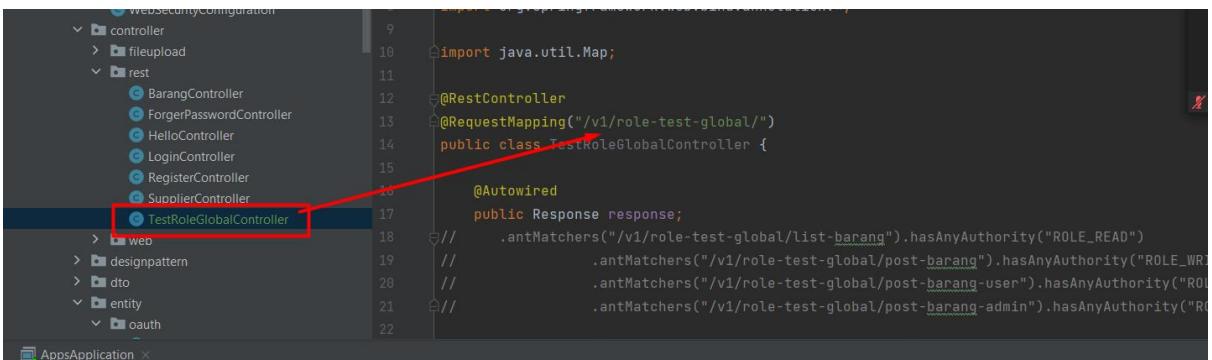
```

    barang").hasAnyAuthority("ROLE_READ")
    .antMatchers("/v1/role-test-global/post-
    barang").hasAnyAuthority("ROLE_WRITE")
    .antMatchers("/v1/role-test-global/post-barang-
    user").hasAnyAuthority("ROLE_USER")
    .antMatchers("/v1/role-test-global/post-barang-
    admin").hasAnyAuthority("ROLE_ADMIN")

```

....

Buat rest API sesuai endpoint yang telah di daftarkan



```

    package com.binar5.apps.controller.rest;

    import com.binar5.apps.entity.Barang;
    import com.binar5.apps.utils.Response;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.http.HttpStatus;
    import org.springframework.http.ResponseEntity;
    import org.springframework.web.bind.annotation.*;

    import java.util.Map;

    @RestController
    @RequestMapping("/v1/role-test-global/")
    public class TestRoleGlobalController {

        @Autowired
        public Response response;

```

```

        //      .antMatchers("/v1/role-test-global/list-
        barang").hasAnyAuthority("ROLE_READ")
        //      .antMatchers("/v1/role-test-global/post-
        barang").hasAnyAuthority("ROLE_WRITE")
        //      .antMatchers("/v1/role-test-global/post-barang-
        user").hasAnyAuthority("ROLE_USER")
        //      .antMatchers("/v1/role-test-global/post-barang-
        admin").hasAnyAuthority("ROLE_ADMIN")

        @GetMapping(value = {"/list-barang", "/list-barang/"})
        public ResponseEntity<Map> list() {
            return new ResponseEntity<Map>(response.sukses("GetMapping -
list-barang - ROLE_READ "), HttpStatus.OK);
        }
    }
}

```

```

    @PostMapping(value = {"/post-barang", "/post-barang/"})
    public ResponseEntity<Map> save() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang - ROLE_WRITE"), HttpStatus.OK);
    }

    @PostMapping(value = {"/post-barang-user", "/post-barang-user/"})
    public ResponseEntity<Map> saveuser() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-user - ROLE_USER"), HttpStatus.OK);
    }

    @PostMapping(value = {"/post-barang-admin", "/post-barang-admin/"})
    public ResponseEntity<Map> saveadmin() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-admin - ROLE_ADMIN"), HttpStatus.OK);
    }
}

```

Testing dengan ROLE/Hak Akses yang Sesuai

Misal username : rikialdipari@gmail.com memiliki role seperti gambar berikut:

23
24 **select ou.id, ou.username , our.role_id , or2.name", or2.type"**
25 **from oauth_user ou, oauth_user_role our , oauth_role or2**
26 **where ou.id =7 and ou.id =our.user_id and or2.id =our.role_id ;**

oauth_user(+) 1 ×

select ou.id, ou.username , our.role_id , or2.name", or2.type" Enter a SQL expression to filter results (use Ctrl+Space)

Grid	1 ou.id	2 ou.username	3 our.role_id	4 or2.name	5 or2.type
1	7	rikialdipari@gmail.com	3	ROLE_USER	user_role
2	7	rikialdipari@gmail.com	4	ROLE_READ	oauth_role
3	7	rikialdipari@gmail.com	5	ROLE_WRITE	oauth_role

select ou.id, ou.username , our.role_id , or2.name", or2.type"
from oauth_user ou, oauth_user_role our , oauth_role or2
where ou.id =7 and ou.id =our.user_id and or2.id =our.role_id ;

Saat hit api dengan “/list-barang” yang hanya bisa diakses oleh ROLE_READ maka seharusnya akan sukses, karena user tersebut memiliki ROLE_READ:

```

16     @Autowired
17     public Response response;
18
19     // .antMatchers("/v1/role-test-global/list-barang").hasAnyAuthority("ROLE_READ")
20     // .antMatchers("/v1/role-test-global/post-barang").hasAnyAuthority("ROLE_WRITE")
21     // .antMatchers("/v1/role-test-global/post-barang-user").hasAnyAuthority("ROLE_USER")
22     // .antMatchers("/v1/role-test-global/post-barang-admin").hasAnyAuthority("ROLE_ADMIN")
23
24     @GetMapping(value = {"list-barang", "list-barang/"})
25     public ResponseEntity<Map> list() {
26         return new ResponseEntity<Map>(response.sukses( obj: "GetMapping - list-barang - ROLE_READ " ), HttpStatus.OK);

```

Mari kita buktikan:

Silahkan lakukan login dengan username : rikialdipari@gmail.com

The screenshot shows a Postman request to `POST {{host}}/user-login/login`. The body contains a JSON object with `"username": "rikialdipari@gmail.com"` and `"password": "password"`. The response status is 200 OK, and the response body is a JWT token.

```

1 {
2   "username": "rikialdipari@gmail.com",
3   "password": "password"
4 }

```

```

1
2   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
3 ey3hdwqiolsib2IdGgyLx1c291cmN1l1osInVzXJfbmFTZSI6InJpa21hbGRpcGFyaUBnWfpbC5jb201LCJzY29wZSI6WyJyZwFkIiwid
3JpdGUlXSw1ZXhwIjoxNjcwDQ4NTEXLc3jdKrb03pdgllcyI6WyJST0xFX1vTRVII1C3ST0xFX1JFQUq1LCJ5T0xFX1dSSVRFile0sImp0eS
I61kxtSwY0ZnhuVTNLUXd5Yw03YjI2b2x4R1hETSi1mNsawVu0d9pZC16Im15LWNsaWVudC13ZWIffQ.
3c9E8nx7kTh932NQXNPu_H_uPjUNEHKenSouQshh",
3 refresh_token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
ey3hdwqiolsib2IdGgyLx1c291cmN1l1osInVzXJfbmFTZSI6InJpa21hbGRpcGFyaUBnWfpbC5jb201LCJzY29wZSI6WyJyZwFkIiwid
3JpdGUlXSw1ZXhwIjoxNjcwDQ4NTEXLc3jdKrb03pdgllcyI6WyJST0xFX1vTRVII1C3ST0xFX1JFQUq1LCJ5T0xFX1dSSVRFile0sImp0eS
I61kxtSwY0ZnhuVTNLUXd5Yw03YjI2b2x4R1hETSi1mNsawVu0d9pZC16Im15LWNsaWVudC13ZWIffQ.
3c9E8nx7kTh932NQXNPu_H_uPjUNEHKenSouQshh",
3 scope: "read write",
3 token_type: "bearer",
3 expires_in: 28799,
3 jti: "LmIf4fxnu3KQwyam7b26olxFXDM"
8

```

Copi pastekan token JWT tersebut: seperti

The screenshot shows a Postman request to `GET {{host}}/v1/role-test-global/list-barang`. The Authorization header is set to `Bearer {{token}}`, where {{token}} is the copied JWT token. The response status is 200 OK, and the response body is a success message.

Headers (7):

- User-Agent
- Accept
- Accept-Encoding
- Connection
- Authorization `Bearer {{token}}`

```

1
2   "code": 200,
3   "data": "GetMapping - list-barang - ROLE_READ ",
4   "status": "sukses"
5

```

Dapat dilihat responeya sukses : berhasil

Testing dengan ROLE/Hak Akses yang Tidak Dimiliki oleh User

Saat hit api dengan “/post-barang-admin” yang hanya bisa diakses oleh ROLE_ADMIN, maka seharusnya akan gagal, karena user tersebut tidak memiliki ROLE_ADMIN:

Ini data role user rikialdipari@gmail.com diambil dari database

```
24 select ou.id, ou.username , our.role_id , or2.name", or2.type"
25 from oauth_user ou, oauth_user_role our, oauth_role or2
26 where ou.id =7 and ou.id =our.user_id and or2.id =our.role_id ;
```

	id	username	role_id	name	type
1	7	rikialdipari@gmail.com	3	ROLE_USER	user_role
2	7	rikialdipari@gmail.com	4	ROLE_READ	oauth_role
3	7	rikialdipari@gmail.com	5	ROLE_WRITE	oauth_role

Mari kita buktikan dengan melakukan hit pada API berikut :

Binar Batch 5 / API Oauth - Test ROLE / ROLE ADMIN

POST `((host))/v1/role-test-global/post-barang-admin`

Headers (8)

- User-Agent
- Accept
- Accept-Encoding
- Connection
- Authorization: {{token}}

Body

Pretty Raw Preview Visualize JSON

1 "error": "access_denied",
2 "error_description": "Access is denied"

Terlihat pada gambar diatas, response eror. Itu dikarenakan endpoint “{{host}}/v1/role-test-global/post-barang-admin” hanya bisa diakses oleh ROLE_ADMIN, padahal username “rikialdipari@gmail.com” tidak memiliki ROLE_ADMIN.

Branch

https://github.com/rikialdi/binar_batch_5/pull/new/071222-authority
071222-authority

b. Authority Role di method REST API

referensi

<https://www.appsdeveloperblog.com/spring-security-preauthorize-annotation-example/>

@PreAuthorize: Dapat memeriksa otorisasi sebelum masuk ke metode. Itu juga memeriksa berdasarkan peran atau argumen yang diteruskan ke metode.

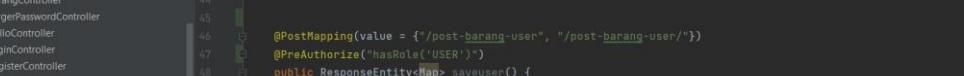
@PostAuthorize: ia memeriksa otorisasi setelah eksekusi metode. Itu juga dapat diotorisasi berdasarkan peran yang masuk, mengembalikan objek dengan metode dan meneruskan argumen ke metode.

Class WebSecurityConfiguration

Application.Properties

```
spring.main.allow-bean-definition-overriding=true
```

Penerapan pada Rest API



```
controller
fileupload
rest
    BarangController
    ForgerPasswordController
    HelloController
    LoginController
    RegisterController
    SupplierController
    TestRoleLocalController
    TestRoleMethodController
web
designpattern
dto
entity
repository
service
testing
utils
validation.web
AppApplication

TestRoleMethodController

    @PostMapping(value = {"/post-barang-user", "/post-barang-user/"})
    @PreAuthorize("hasRole('USER')")
    public ResponseEntity<Map> saveuser() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-user - ROLE_USER"), HttpStatus.OK);
    }

    @PostMapping(value = {"/post-barang-admin", "/post-barang-admin/"})
    // @PreAuthorize("hasAuthority('ADMIN')")
    // https://www.apptechguru.com/spring-security-preauthorize-annotation-example/
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<Map> savedadmin() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-admin - ROLE_ADMIN"), HttpStatus.OK);
    }
}
```

Atau

```
    @PutMapping
    @PreAuthorize("hasRole('USER') or hasRole('ADMIN')")
    public ResponseEntity<Map> updateEmployee(EmployeeModel employeeModel) {
        try {
```

```
package com.binar5.apps.controller.rest;
```

```
import com.binar5.apps.utils.Response;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.EnableAspectJAutoProxy;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PostAuthorize;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import
org.springframework.web.servlet.config.annotation.EnableWebMvc;

import java.util.Map;

@RestController
@RequestMapping("/v1/role-test-method/")
public class TestRoleMethodController {

    @Autowired
    public Response response;
//        .antMatchers("/v1/role-test-global/list-
barang").hasAnyAuthority("ROLE_READ")
//                .antMatchers("/v1/role-test-global/post-
barang").hasAnyAuthority("ROLE_WRITE")
//                .antMatchers("/v1/role-test-global/post-barang-
user").hasAnyAuthority("ROLE_USER")
//                .antMatchers("/v1/role-test-global/post-barang-
admin").hasAnyAuthority("ROLE_ADMIN")

//    @PreAuthorize("hasAuthority('DELETE_AUTHORITY')") // untuk
delete
```

```

    @GetMapping(value = {"/list-barang", "/list-barang/"})
    @PreAuthorize("hasRole('READ')")
    public ResponseEntity<Map> list() {
        return new ResponseEntity<Map>(response.sukses("GetMapping - list-barang - ROLE_READ"), HttpStatus.OK);
    }

    @PostMapping(value = {"/post-barang", "/post-barang/"})
    @PreAuthorize("hasRole('WRITE')")
    public ResponseEntity<Map> save() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang - ROLE_WRITE"), HttpStatus.OK);
    }

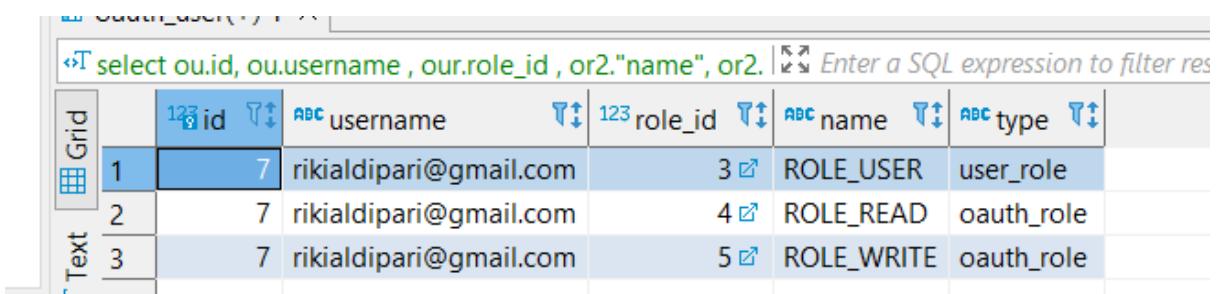
    @PostMapping(value = {"/post-barang-user", "/post-barang-user/"})
    @PreAuthorize("hasRole('USER')")
    public ResponseEntity<Map> saveuser() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-user - ROLE_USER"), HttpStatus.OK);
    }

    @PostMapping(value = {"/post-barang-admin", "/post-barang-admin/"})
    //     @PreAuthorize("hasAuthority('ROLE_ADMIN')")
    //     https://www.appsdeveloperblog.com/spring-security-preauthorize-annotation-example/
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<Map> saveadmin() {
        return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-admin - ROLE_ADMIN"), HttpStatus.OK);
    }

}

```

Testing : User tidak memiliki role_admin



The screenshot shows a MySQL Workbench interface with a query editor at the top containing the SQL query:

```
select ou.id, ou.username , or2.role_id , or2.name", or2.
```

Below the query is a note: "Enter a SQL expression to filter results".

The main area displays a grid of data from three tables: oauth_user (ou), oauth_role (or2), and oauth_userrole (or1). The columns are labeled: id, username, role_id, name, and type. The data is as follows:

	id	username	role_id	name	type
Grid	1	rikialdipari@gmail.com	3	ROLE_USER	user_role
Text	2	rikialdipari@gmail.com	4	ROLE_READ	oauth_role
	3	rikialdipari@gmail.com	5	ROLE_WRITE	oauth_role

Output Postman

The screenshot shows a POST request to `({{host}}/v1/role-test-method/post-barang-admin`. The Headers tab is selected, showing the following configuration:

- User-Agent: `PostmanRuntime/7.29.2`
- Accept: `*`
- Accept-Encoding: `gzip, deflate, br`
- Connection: `keep-alive`
- Authorization: `({{token}})`

The Body tab shows a JSON response with the following content:

```

1 "error": "access_denied",
2 "error_description": "Access is denied"

```

Terlihat akses ditolak, itu disebabkan di database user rikialdipari@gmail.com tidak memiliki role_admin

Testing: User memiliki role_admin

The screenshot shows a SQL query in DBeaver:

```

23 select us FROM oauth_client uc ;
24
25 select ou.id, ou.username , our.role_id , or2."name", or2."type"
26 from oauth_user ou, oauth_user_role our , oauth_role or2
27 where ou.id =7 and ou.id =our.user_id and or2.id =our.role_id ;

```

The results grid shows the following data:

id	username	role_id	name	type
1	rikialdipari@gmail.com	2	ROLE_ADMIN	user_role
2	rikialdipari@gmail.com	4	ROLE_READ	oauth_role
3	rikialdipari@gmail.com	5	ROLE_WRITE	oauth_role

Output postman : berhasil, dikarenakan username rikialdipari@gmail.com memiliki role_admin pada database, sehingga akses akan di allow.

The screenshot shows the Postman interface with a successful API call. The URL is `http://{{host}}/v1/role-test-method/post-barang-admin`. The Headers tab is selected, showing the following configuration:

Header	Value
User-Agent	PostmanRuntime/7.29.2
Accept	/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	<code>{{token}}</code>

The Body tab shows the response:

```
1 "code": 200,
2 "data": "PostMapping post-barang-admin - ROLE_ADMIN",
3 "status": "sukses"
```

c. HashAuthority Role di method REST API

Perbedaan **hasAuthority** dengan **hasRole** adalah dengan ditambahkan role. Contoh

hasAuthority = ROLE_ADMIN

hasRole =ADMIN



The screenshot shows a Java code editor with a sidebar containing project navigation. The main area displays a class with two methods annotated with `@PreAuthorize`. The first method is annotated with `@PreAuthorize("hasRole('ADMIN')")`, and the second method is annotated with `@PreAuthorize("hasAuthority('ROLE_ADMIN')")`.

```
SupplierController  
TestRoleGlobalController  
TestRoleMethodController  
  
> web  
> designpattern  
> dto  
> entity  
  > oauth  
    Client  
    Role  
    RolePath  
    User  
  AbstractDate  
  Barang  
  Barang2  
  Barang3  
  DetailBarang  
  Karyawan  
  
51  
52  
53 @PostMapping(value = {"/post-barang-admin", "/post-barang-admin"})  
54 // @PreAuthorize("hasAuthority('ADMIN')")  
55 // https://www.baeldung.com/spring-security-preauthorize-annotation-example/  
56 @PreAuthorize("hasRole('ADMIN')")  
57 public ResponseEntity<Map> saveadmin() {  
58     return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-admin - ROLE_ADMIN"), HttpStatus.OK);  
59 }  
  
60  
61 @PostMapping(value = {"/post-barang-admin-andauthority", "/post-barang-admin-andauthority"})  
62 @PreAuthorize("hasAuthority('ROLE_ADMIN')")  
63 public ResponseEntity<Map> saveadminandauthority() {  
64     return new ResponseEntity<Map>(response.sukses("PostMapping - post-barang-admin - ROLE_ADMIN"), HttpStatus.OK);  
65 }
```

Testing Postman : Kondisi Success

User rikialdipari@gmail.com memiliki role_admin didatabase

Enter a SQL expression to filter results

Grid	123 id	ABC username	123 role_id	ABC name	ABC type
	1	rikialdipari@gmail.com	2 ↕	ROLE_ADMIN	user_role
	2	rikialdipari@gmail.com	4 ↕	ROLE_READ	oauth_role
	3	rikialdipari@gmail.com	5 ↕	ROLE_WRITE	oauth_role

Sehingga output postman sukses.

The screenshot shows the Postman application interface. At the top, there is a 'POST' method indicator and a URL field containing `(({host}))/v1/role-test-method/post-barang-admin-andauthority`. Below the URL, the 'Headers (8)' tab is selected, showing the following configuration:

Key	Description
User-Agent	PostmanRuntime/7.29.2
Accept	/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	(({token}))

Below the headers, the 'Body' tab is selected, showing a JSON response:

```
1 "code": 200,
2 "data": "PostMapping - post-barang-admin - ROLE_ADMIN",
3 "status": "sukses"
```

d. Branch

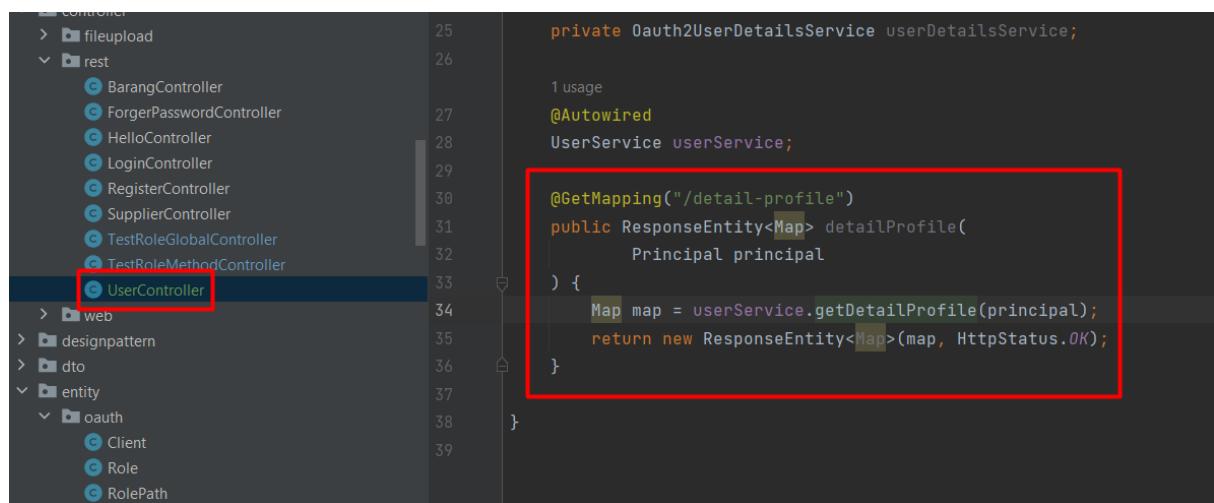
081222-authority-user

https://github.com/rikialdi/binar_batch_5/pull/new/081222-authority-us

7. Get User Detail Dari Token/ Read Token / Baca Token

Kadang kalanya kita ingin melihat detail informasi user dari token yang digunakan saat akses REST API. Maka berikut langkah-langkah:

a. Controller



```
private OAuth2UserDetailsService userDetailsService;
    ...
    @GetMapping("/detail-profile")
    public ResponseEntity<Map> detailProfile(
        Principal principal
    ) {
        Map map = userService.getDetailProfile(principal);
        return new ResponseEntity<Map>(map, HttpStatus.OK);
    }
}
```

```
package com.binar5.apps.controller.rest;

import com.binar5.apps.repository.oauth.UserRepository;
import com.binar5.apps.service.UserService;
import com.binar5.apps.service.oauth.OAuth2UserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.security.Principal;
import java.util.Map;

@RestController
@RequestMapping("/user")
public class UserController {

    @Autowired
    UserRepository userRepository;

    @Autowired
    private OAuth2UserDetailsService userDetailsService;

    @Autowired
    UserService userService;

    @GetMapping("/detail-profile")
```

```

    public ResponseEntity<Map> detailProfile(
        Principal principal
    ) {
        Map map = userService.getDetailProfile(principal);
        return new ResponseEntity<Map>(map, HttpStatus.OK);
    }
}

```

b. Service

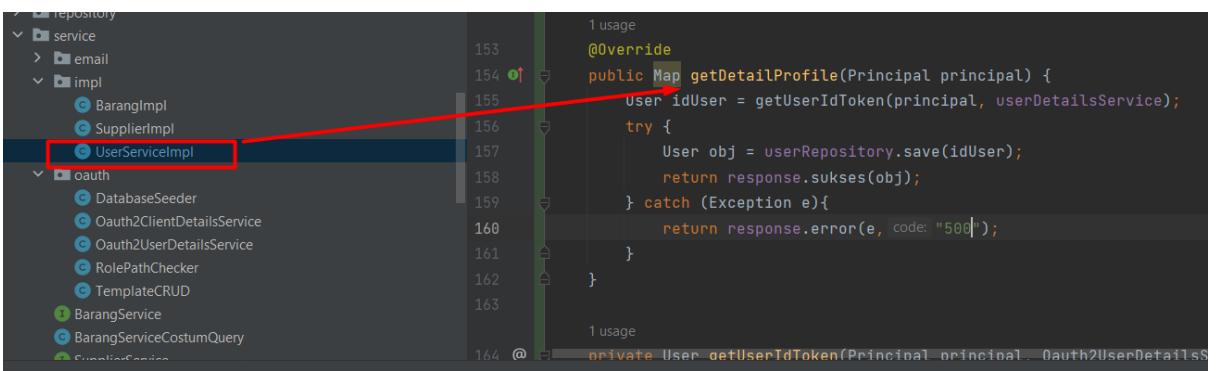


```

public interface UserService {
    Map login(LoginModel objLogin);
    Map registerManual(RegisterModel objModel);
    Map getDetailProfile(Principal principal);
}

```

c. Service Implementasi



```

@Override
public Map getDetailProfile(Principal principal) {
    User idUser = getUserIdToken(principal, userDetailsService);
    try {
        User obj = userRepository.save(idUser);
        return response.sukses(obj);
    } catch (Exception e){
        return response.error(e, code: "500");
    }
}

@Autowired
public Response templateResponse;
@Autowired
private Oauth2UserDetailsService userDetailsService;

@Override
public Map getDetailProfile(Principal principal) {
    User idUser = getUserIdToken(principal, userDetailsService);
    try {
        User obj = userRepository.save(idUser);
        return response.sukses(obj);
    } catch (Exception e){

```

```

        return response.error(e,"500");
    }

}

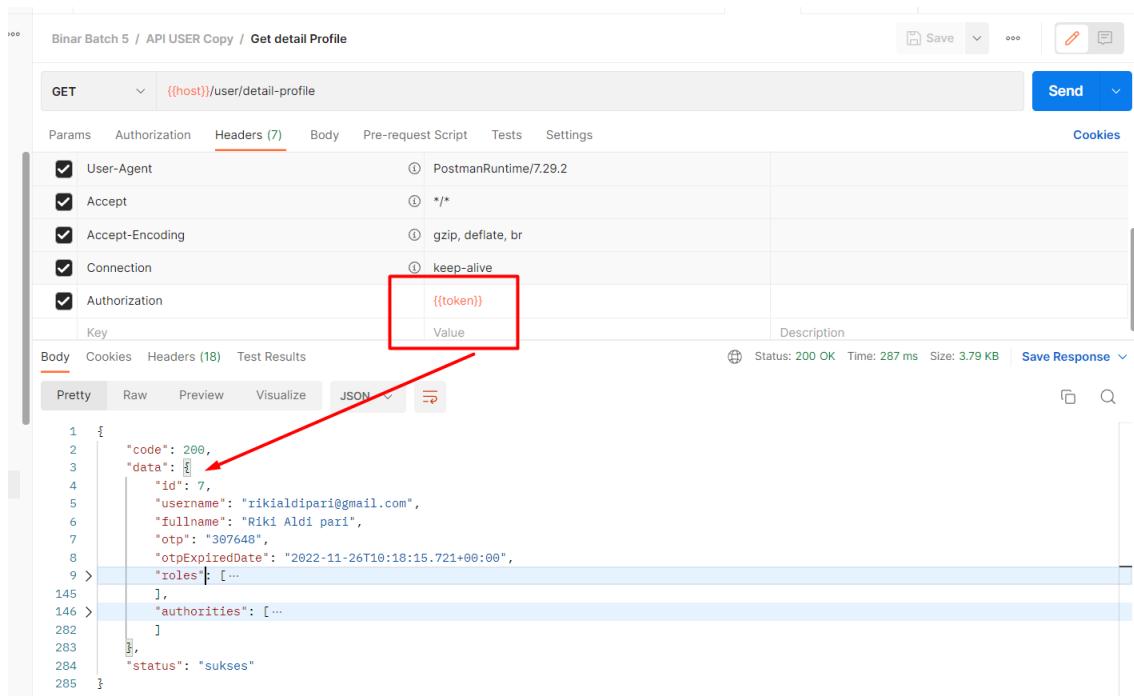
private User getUserIdToken(Principal principal,
Oauth2UserDetailsService userDetailsService) {
    UserDetails user = null;
    String username = principal.getName();
    if (!StringUtils.isEmpty(username)) {
        user = userDetailsService.loadUserByUsername(username);
    }

    if (null == user) {
        throw new UsernameNotFoundException("User not found");
    }
    User idUser =
userRepository.findOneByUsername(user.getUsername());
    if (null == idUser) {
        throw new UsernameNotFoundException("User name not found");
    }
    return idUser;
}

```

d. Testing Postman

Dari token tersebut, akan menampilkan informasi user yang memiliki token tersebut.



The screenshot shows a Postman request configuration for a GET method to the endpoint `/user/detail-profile`. The Authorization header is set to `Bearer {{token}}`, with the value highlighted by a red box. The response body is displayed in Pretty JSON format, showing a successful 200 OK status with a user profile object. An arrow points from the 'Value' field in the Authorization header to the 'data' field in the JSON response, indicating the correlation between the token and the returned user information.

```

{
  "code": 200,
  "data": {
    "id": 7,
    "username": "rikialdiipari@gmail.com",
    "fullname": "Riki Aldi pari",
    "otp": "307648",
    "otpExpiredDate": "2022-11-26T10:18:15.721+00:00",
    "roles": [
      ...
    ],
    "authorities": [
      ...
    ]
  },
  "status": "sukses"
}

```

e. Branch

081222-authority-user

https://github.com/rikialdi/binar_batch_5/pull/new/081222-authority-user

8. Register

a. Register Manual

<https://github.com/rikialdi/dibimbing>

Branch : spring-security-01-registerotp

RegisterModel

Untuk validasi refer ke : 25. Menampilkan Eror Validation Hibernite

```
import lombok.Data;  
  
import javax.validation.constraints.NotEmpty;  
  
@Data  
public class RegisterModel {  
    @NotEmpty(message = "username is required.")  
    private String username;  
    @NotEmpty(message = "password is required.")  
    private String password;  
    @NotEmpty(message = "fullname is required.")  
    private String fullname;  
}
```

Service

```
import java.util.Map;  
  
public interface UserService {  
    Map registerManual(RegisterModel objModel) ;  
}
```

Service Impl

```
package com.binar.grab.service.impl;  
  
import com.binar.grab.config.Config;  
import com.binar.grab.dao.request.RegisterModel;  
import com.binar.grab.model.oauth.Role;  
import com.binar.grab.model.oauth.User;  
import com.binar.grab.repository.oauth.RoleRepository;  
import com.binar.grab.repository.oauth.UserRepository;  
import com.binar.grab.service.UserService;
```

```

import com.binar.grab.util.TemplateResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
@Service
public class UserServiceImpl implements UserService {

    Config config = new Config();
    private static final Logger logger =
LoggerFactory.getLogger(UserServiceImpl.class);
    @Autowired
    RoleRepository repoRole;

    @Autowired
    UserRepository repoUser;

    @Autowired
    private PasswordEncoder encoder;

    @Autowired
    public TemplateResponse templateResponse;
    @Override
    public Map registerManual(RegisterModel objModel) {
        Map map = new HashMap();
        try {
            String[] roleNames = {"ROLE_USER", "ROLE_READ",
"ROLE_WRITE"}; // admin
            User user = new User();
            user.setUsername(objModel.getEmail().toLowerCase());
            user.setFullname(objModel.getFullname());

            //step 1 :
//            user.setEnabled(false); // matikan user

            String password =
encoder.encode(objModel.getPassword()).replaceAll("\s+", "");
            List<Role> r = repoRole.findByNameIn(roleNames);

            user.setRoles(r);
            user.setPassword(password);
            User obj = repoUser.save(user);

            return templateResponse.templateSukses(obj);
        } catch (Exception e) {
            logger.error("Error registerManual=", e);
            return templateResponse.templateError("error:" + e);
        }
    }
}

```

Controller

```
package com.binar.grab.controller;

import com.binar.grab.config.Config;
import com.binar.grab.dao.request.RegisterModel;
import com.binar.grab.model.oauth.User;
import com.binar.grab.repository.oauth.UserRepository;
import com.binar.grab.service.UserService;
import com.binar.grab.util.TemplateResponse;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("/v1/user-register/")
public class RegisterController {
    @Autowired
    private UserRepository userRepository;

    Config config = new Config();

    @Autowired
    public UserService serviceReq;

    @Autowired
    public TemplateResponse templateCRUD;
    @Autowired
    public Response templateCRUD;
    @PostMapping("/register")
    public ResponseEntity<Map> saveRegisterManual(@Valid @RequestBody
RegisterModel objModel) throws RuntimeException {
        Map map = new HashMap();

        User user =
userRepository.checkExistingEmail(objModel.getUsername());
        if (null != user) {
            return new ResponseEntity<Map>(templateCRUD.Error("Username
sudah ada"), HttpStatus.OK);
        }
        map = serviceReq.registerManual(objModel);

        return new ResponseEntity<Map>(map, HttpStatus.OK);
    }
}
```

}

Allow URI:

Testing :Run postman :

The screenshot shows the Postman interface with the following details:

- HTTP Method:** POST
- URL:** {{host}}/v1/user-register/register
- Body (JSON):**

```
1 ...
2 ... "username": "rikialdipari1@gmail.com",
3 ... "password": "password",
4 ... "fullname": "riki aldi pari"
5 ...
```

- Response Status:** 200 OK
- Time:** 759 ms
- Size:** 980 B
- Preview:** Shows the JSON response structure.

Branch:

https://github.com/rikialdi/binar_batch_5/pull/new/231122-spring-security-register

atau

https://gitlab.com/rikialdi/superproof-adam/-/tree/sesi6-register-google?ref_type=heads

b. Register By OTP : send OTP to email

Alur : analogi

1. ketika daftar -> akun disabel
di field name enable user

	id	not_expired	not_locked	credential_not_expired	enabled	expired
1	1	[v]	[v]	[v]	[v]	
2	2	[v]	[v]	[v]	[v]	
3	3	[v]	[v]	[v]	[v]	
4	4	[v]	[v]	[v]	[v]	

Jika enable : false maka:

HTTP Karyawan / API Login / OAUTH LOGIN

POST localhost:9090/api/oauth/token

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Body (x-www-form-urlencoded)

Key	Value
username	rikaldipari@gmail.com
password	password
grant_type	password

Body Cookies Headers (13) Test Results

Pretty Raw Preview Visualize JSON

```
1 "error": "invalid_grant",
2 "error_description": "User is disabled"
```

2. gimana supaya aktif? -> maka dia perlu trigger by email :
3. sukses -> dan akun enable

<https://github.com/rikaldi/dibimbing>

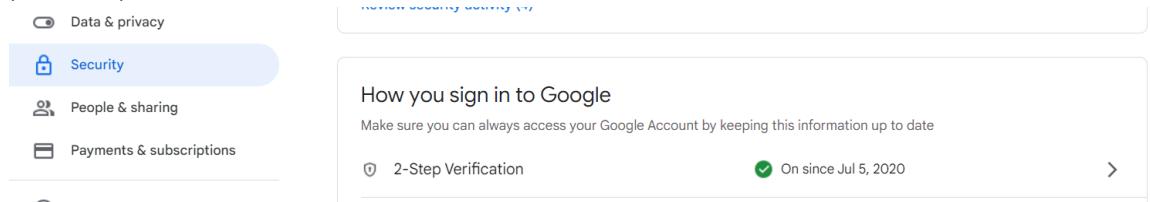
Branch : 04032022-register-otp

Step 1: email get token

https://myaccount.google.com/security?rapt=AEjHL4Mfta4W0SV07XXuiGH9daS0LbSvQKrHNDDxDNq4rKPs07vM830CadXQ1z-dY2EjrXgppqzrGXu0d-R66ZB0-_9ff9_sYw

documentation : <https://www.javacodemonk.com/spring-boot-send-email-with-gmail-smtp-5caeae8f3>

1. pilih step verification:klik



2. pilih app password



Devices that don't need a second step

You can skip the second step on devices you trust, such as your own computer.



Devices you trust

Revoke trusted status from your devices that skip 2-Step Verification.

[REVOKE ALL](#)

App passwords

App Passwords aren't recommended and are unnecessary in most cases. To help keep your account secure, use "Sign in with Google" to connect apps to your Google Account.

App passwords

1 password

3. Create generate

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Name	Created	Last used
Mail on my Windows Computer	Jul 27	Aug 5

Select the app and device you want to generate the app password for.

Select app ▾ Select device ▾

GENERATE

And save your token

Step 2 : Pom.xml

```
<!-- email -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
</dependency>
```

Step 3 :application.properties

```

expired.token.password.minute=1200

#email follow :
https://support.google.com/accounts/answer/185833?p=InvalidSecondFact
or&visit_id=637690832060530868-1439835364&rd=1
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=rikialdipari@gmail.com
spring.mail.password=lblsdoexriohzmxqtj
spring.mail.sender.name=admin
spring.mail.sender.mail=no-reply@test.com
# Other properties
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.connectiontimeout=5000
spring.mail.properties.mail.smtp.timeout=5000
spring.mail.properties.mail.smtp.writetimeout=5000
# TLS , port 587
spring.mail.properties.mail.smtp.starttls.enable=true

```

Step 4: UserService dan UserServicelmpl

Add method baru :

```

Map registerByGoogle(RegisterModel objModel) ;

```

UserServicelmpl

```

@Override
public Map registerByGoogle(RegisterModel objModel) {
    Map map = new HashMap();
    try {
        String[] roleNames = {"ROLE_USER", "ROLE_READ",
"ROLE_WRITE"}; // ROLE DEFUALE
        User user = new User();
        user.setUsername(objModel.getUsername().toLowerCase());
        user.setFullname(objModel.getFullname());
        //step 1 :
        user.setEnabled(false); // matikan user
        String password =
encoder.encode(objModel.getPassword().replaceAll("\s+", ""));
        List<Role> r = repoRole.findByNameIn(roleNames);
        user.setRoles(r);
        user.setPassword(password);
        User obj = repoUser.save(user);
        return response.Sukses(obj);

    } catch (Exception e) {
        logger.error("Eror registerManual=", e);
        return response.Error("eror:"+e);
    }
}

```

Step 5 : Controller Register

```
@Autowired
public EmailSender emailSender;
@Autowired
public EmailTemplate emailTemplate;

@Value("${expired.token.password.minute:}") //FILE_SHOW_RUL
private int expiredToken;

@PostMapping("/register-google")
public ResponseEntity<Map> saveRegisterByGoogle(@Valid @RequestBody
RegisterModel objModel) throws RuntimeException {
    Map map = new HashMap();

    User user =
userRepository.checkExistingEmail(objModel.getUsername());
    if (null != user) {
        return new ResponseEntity<Map>(templateCRUD.Error("Username
sudah ada"), HttpStatus.OK);
    }
    map = serviceReq.registerByGoogle(objModel);
    //gunanya send email
    Map mapRegister = sendEmailegister(objModel);
    return new ResponseEntity<Map>(mapRegister, HttpStatus.OK);
}

// Step 2: sendp OTP berupa URL: guna updeta enable agar bisa login:
@PostMapping("/send-otp") //send OTP
public Map sendEmailegister(
    @RequestBody RegisterModel user) {
    String message = "Thanks, please check your email for activation.";

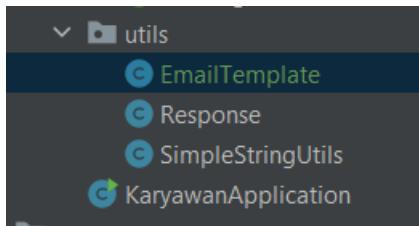
    if (user.getUsername() == null) return templateCRUD.Sukses("No email
provided");
    User found = userRepository.findOneByUsername(user.getUsername());
    if (found == null) return templateCRUD.Error("Email not found");
    //throw new BadRequest("Email not found");

    String template = emailTemplate.getRegisterTemplate();
    if (StringUtils.isEmpty(found.getOtp())) {
        User search;
        String otp;
        do {
            otp = SimpleStringUtils.randomString(6, true);
            search = userRepository.findOneByOTP(otp);
        } while (search != null);
        Date dateNow = new Date();
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(dateNow);
```

```
        calendar.add(Calendar.MINUTE, expiredToken);
        Date expirationDate = calendar.getTime();

        found.setOtp(otp);
        found.setOtpExpiredDate(expirationDate);
        template = template.replaceAll("\\{\\{\\{USERNAME}\\}\\}\\}", 
        (found.getFullscreen() == null ? found.getUsername() : 
        found.getFullscreen()));
        template = template.replaceAll("\\{\\{\\{VERIFY_TOKEN}\\}\\}\\}", otp);
        userRepository.save(found);
    } else {
        template = template.replaceAll("\\{\\{\\{USERNAME}\\}\\}\\}", 
        (found.getFullscreen() == null ? found.getUsername() : 
        found.getFullscreen()));
        template = template.replaceAll("\\{\\{\\{VERIFY_TOKEN}\\}\\}\\}", 
        found.getOtp());
    }
    emailSender.sendAsync(found.getUsername(), "Register", template);
    return templateCRUD.Sukses(message);
}
```

Step 6 : utils : Email Templates



```

"\t\ttext-decoration: none;\n" +
"\t\twidth: 30%;\n" +
"\t\ttext-align: center;\n" +
"\t\tborder: 1px solid #01c853;\n" +
"\t\ttext-transform: uppercase;\n" +
"\t}\n" +
"\ta.btn:hover,\n" +
"\ta.btn:focus {\n" +
"\tcolor: #01c853;\n" +
"\tbackground-color: #fff;\n" +
"\tborder: 1px solid #01c853;\n" +
"\t}\n" +
"\t.user-name {\n" +
"\ttext-transform: uppercase;\n" +
"\t}\n" +
"\t.manual-link,\n" +
"\t.manual-link:hover,\n" +
"\t.manual-link:focus {\n" +
"\tdisplay: block;\n" +
"\tcolor: #396fad;\n" +
"\tfont-weight: bold;\n" +
"\tmargin-top: -15px;\n" +
"\t}\n" +
"\t.mt--15 {\n" +
"\tmargin-top: -15px;\n" +
"\t}\n" +
"</style>\n" +
"</head>\n" +
"<body>\n" +
"\t<div class=\"email-container\"\n" +
"\t<p>Halo <span class=\"user-
name\">{ {USERNAME}}</span> Selamat bergabung</p>\n" +
"\t<p>Harap konfirmasikan email kamu dengan memasukan
kode dibawah ini</p>\n" +
"\t\t\n" +
"\t\tkode: <b>{ {VERIFY_TOKEN}}</b>\n" +
"\t\t\n" +
"\t\t<p class=\"mt--15\">Jika kamu butuh bantuan atau
pertanyaan, hubungi customer care kami di .... atau kirim email ke
....</p>\n" +
"\t\t\n" +
"\t\t<p>Semoga harimu menyenangkan!</p>\n" +
"\t\t\n" +
"\t\t<p>PT ABC,</p>\n" +
"\t\t<p class=\"mt--15\"....</p>\n" +
"\n" +
"\t\t\n" +
"\t</div>\n" +
"</body>\n" +
"</html>";
}

```

Step-7 : SimpleStringUtils

```
package com.binar.grab.util;
```

```

import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;

import java.util.Random;

public class SimpleStringUtils {
    public static String randomString(int size) {
        return randomString(size, false);
    }

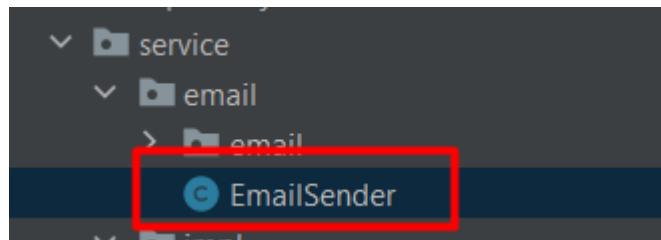
    @SuppressWarnings("SpellCheckingInspection")
    public static String randomString(int size, boolean numberOnly)
    {
        String saltChars = "1234567890";
        if (!numberOnly) {
            saltChars +=
"ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
        }
        StringBuilder salt = new StringBuilder();
        Random rnd = new Random();
        while (salt.length() < size) {
            int index = (int) (rnd.nextFloat() *
saltChars.length());
            salt.append(saltChars.charAt(index));
        }

        return salt.toString();
    }

    public Pageable getShort(String orderby, String ordertype,
Integer page, Integer size) {
        Pageable show_data;
        if (orderby != null) {
            if (ordertype != null) {
                if (ordertype.equals("desc")) {
                    show_data = PageRequest.of(page, size,
Sort.by(orderby).descending());
                } else {
                    show_data = PageRequest.of(page, size,
Sort.by(orderby).ascending());
                }
            } else {
                show_data = PageRequest.of(page, size,
Sort.by(orderby).descending());
            }
        } else {
            show_data = PageRequest.of(page, size,
Sort.by("id").descending());
        }
        return show_data;
    }
}

```

Step-8 : Service : EmailSender



```
package com.binar.grab.service.email;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.task.TaskExecutor;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;

import javax.mail.internet.MimeMessage;

@SuppressWarnings({"WeakerAccess", "ConstantConditions"})
@Component("emailSender")
public class EmailSender {
    private final static Logger logger =
LoggerFactory.getLogger(EmailSender.class);

    @Autowired
    private JavaMailSenderImpl mailSender;

    @Value("${spring.mail.sender.name}")
    private String senderName;

    @Value("${spring.mail.sender.mail}")
    private String senderEmail;

    @Qualifier("taskExecutor")
    @Autowired
    private TaskExecutor taskExecutor;

    public boolean send(String email, String subject, String
message) {
        return send(null, email, subject, message);
    }

    public boolean send(String from, String email, String subject,
String message) {
        MimeMessage mime = mailSender.createMimeMessage();
        if (StringUtils.isEmpty(from)) {
            from = senderEmail;
        }
    }
}
```

```

        if (StringUtils.isEmpty(from)) {
            from = "admin@mail.com";
        }
        boolean success = false;
        try {
            logger.info("Sending email to: "+email);
            logger.info("Sending email from: "+from);
            logger.info("Sending email with subject: "+subject);

            MimeMessageHelper helper = new MimeMessageHelper(mime,
true);
            helper.setFrom(from, senderName);
            helper.setTo(email);
            helper.setSubject(subject);
            helper.setText(message, true);
            mailSender.send(mime);
            success = true;
        } catch (Exception e) {
            logger.error("error: "+e.getMessage());
        }

        return success;
    }

    public void sendAsync(final String to, final String subject,
final String message) {
    taskExecutor.execute(new Runnable() {
        @Override
        public void run() {

            send(to, subject, message);
        }
    });
}

```

Step-9: Postman

POST localhost:8082/api/user-register/send-otp

Body (JSON)

```

1 {
2   "email": "rikialdipari@gmail.com"
3 }

```

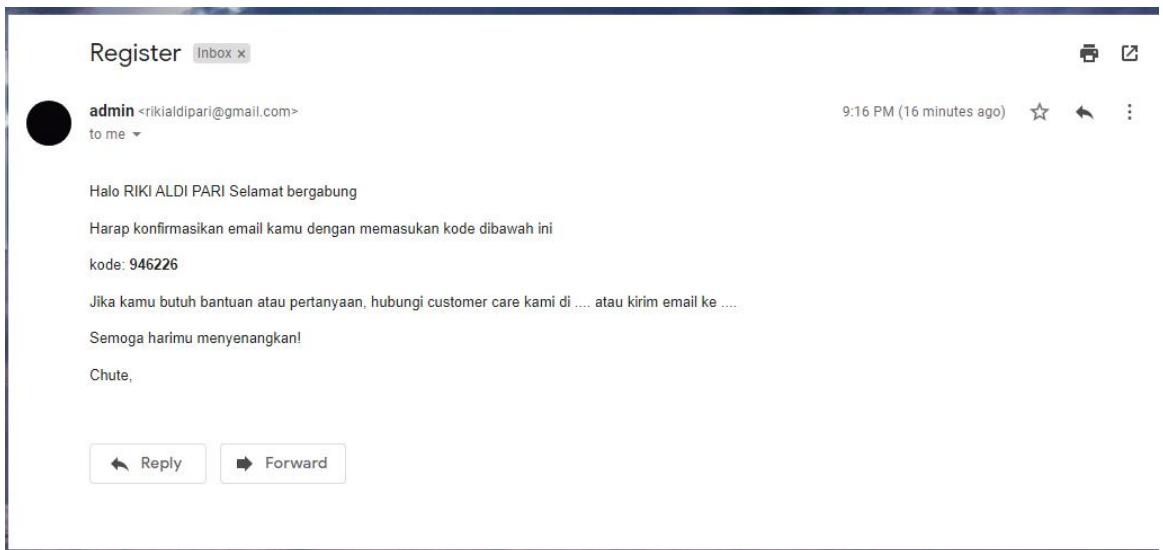
Status: 200 OK Time: 233 ms Size: 787 B Save Response

```

1 {
2   "data": "Thanks, please check your email for activation.",
3   "message": "sukses",
4   "status": "200"
5 }

```

Step-10: Lihat inbox email



Branch

https://github.com/rikialdi/binar_batch_5/pull/new/231122-spring-security-register

https://gitlab.com/rikialdi/superproof-adam/-/tree/sesi6-register-google?ref_type=heads

c. Konfirmasi OTP Token User: BY API

Bertujuan untuk mengupdate user agar dapat login

Controller API Confirm

```
@GetMapping("/register-confirm-otp/{token}")
public ResponseEntity<Map> saveRegisterManual(@PathVariable(value =
"token") String tokenOtp) throws RuntimeException {

    User user = userRepository.findOneByOTP(tokenOtp);
    if (null == user) {
        return new ResponseEntity<Map>(templateCRUD.templateError("OTP tidak
ditemukan"), HttpStatus.OK);
    }
    //validasi jika sebelumnya sudah melakukan aktifasi

    if(user.isEnabled()){
        return new ResponseEntity<Map>(templateCRUD.templateSuccess("Akun
Anda sudah aktif, Silahkan melakukan login"), HttpStatus.OK);
    }
    String today = config.convertDateToString(new Date());

    String dateToken =
config.convertDateToString(user.getOtpExpiredDate());
    if(Long.parseLong(today) > Long.parseLong(dateToken)) {
        return new ResponseEntity<Map>(templateCRUD.templateError("Your
token is expired. Please Get token again."), HttpStatus.OK);
    }
    //update user
    user.setEnabled(true);
    userRepository.save(user);

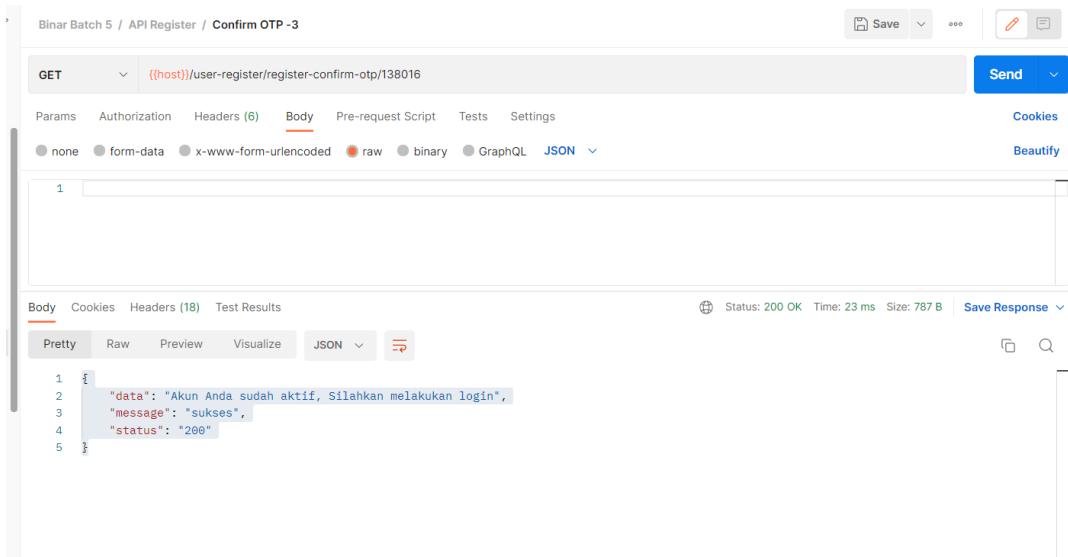
    return new ResponseEntity<Map>(templateCRUD.templateSuccess("Sukses,
Silahkan Melakukan Login"), HttpStatus.OK);
}
```

Config

```
public String convertDateToString(Date date) {

    DateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmss");
    String strDate = dateFormat.format(date);
    return strDate;
}
```

Testing Postman



The screenshot shows the Postman interface with a successful API call. The URL is `({{host}})/user-register/register-confirm-otp/138016`. The response status is 200 OK, and the JSON body contains:

```
1 {
2   "data": "Akun Anda sudah aktif. Silahkan melakukan login",
3   "message": "sukses",
4   "status": "200"
5 }
```

d. Register OTP by URI Tymeleaf

Akan melakukan validasi token menggunakan url : tingal di klik dan otomatis trigger bisa login.

Refer ke section “**Register OTP by URI Tymeleaf**”

9. Register OTP by URI Tymeleaf

a. Alur

- register -> status user disable
- disable -> enable : perlu kirim email : **url** -> klik user , untuk mengaktifkan user disable.

b. Branch :

register-tymeleaf

<https://github.com/rikialdi/dibimbing.git>

branch ke-2 : 241122-logging

https://github.com/rikialdi/binar_batch_5.git

c. POM.xml

```
<!-- //web tymeleaf-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
    <version>2.2.6.RELEASE</version>
</dependency>
```

d. Step 1-Controller

Masih menggunakan class controller register

```
@Value("${BASEURL}")//FILE_SHOW_RUL
private String BASEURL;

@PostMapping("/send-otp-tymeleaf")//send OTP
public Map sendEmailRegisterTymeleaf(@RequestBody RegisterModel user)
{
    String message = "Thanks, please check your email for activation.";

    if (user.getUsername() == null) return templateCRUD.Error("No email provided");
    User found =
userRepository.findOneByUsername(user.getUsername());
    if (found == null) return templateCRUD.Error("Email not found");
//throw new BadRequest("Email not found");

    String template = emailTemplate.getRegisterTemplate();
    if (StringUtils.isEmpty(found.getOtp())) {
        User search;
        String otp;
        do {
            otp = SimpleStringUtils.randomString(6, true);
            search = userRepository.findOneByOTP(otp);
        } while (search != null);
        Date dateNow = new Date();
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(dateNow);
        calendar.add(Calendar.MINUTE, expiredToken);
        Date expirationDate = calendar.getTime();

        found.setOtp(otp);
        found.setOtpExpiredDate(expirationDate);
        template = template.replaceAll("\\{\\{USERNAME}\\}",(
found.getfullname() == null ? found.getUsername() :
found.getfullname()));
        template = template.replaceAll("\\{\\{VERIFY_TOKEN}\\}",(
BASEURL + "/v1/user-register/web/index/" + otp));
        userRepository.save(found);
    } else {
        template = template.replaceAll("\\{\\{USERNAME}\\}",(
found.getfullname() == null ? found.getUsername() :
```

```

        found.getFullscreen());
        template = template.replaceAll("\\{\\{VERIFY_TOKEN}\\}", 
        BASEURL + "/v1/user-register/web/index/" + found.getOtp());
    }
    emailSender.sendAsync(found.getUsername(), "Register", template);
    return templateCRUD.Sukses(message);
}

```

e. Step 2- Application.properties

```
BASEURL=http://localhost:8082/api/
```

f. Step 3- Controller Tymeleaf

The screenshot shows the file structure of a Java project:

- com.binar.grab** package:
 - config** folder:
 - Config
 - CorsOriginConfiguration
 - Oauth2AccessTokenConverter
 - Oauth2AuthorizationServerConfiguration
 - Oauth2ResourceServerConfiguration
 - ResourceConfig.java
 - ResourceNotFoundException
 - RestRepositoryConfiguration
 - ThreadConfiguration
 - WebSecurityConfiguration
 - controller** folder:
 - fileupload
 - tymeleaf** folder:
 - BarangTymeleafController
 - RegisterConfirm**
 - BarangController
 - BarangDummyController
 - ForgerPasswordController
 - MahasiswaController
 - PembeliController
 - PerpustakaanDummyController

The code editor shows the **RegisterConfirm** class:

```

import java.security.Principal;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

@Controller
@RequestMapping("/user-register/register/web/")
public class RegisterConfirm {

    @Autowired
    public UserRepository userRepo;

    Config config = new Config();

    @GetMapping(value = {"/index/{tokenotp}"})
    public String index(Model model, @PathVariable String tokenotp) {
        User user = userRepo.findOneByOTP(tokenotp);
        if (null == user) {
            System.out.println("user null: tidak ditemukan");
            model.addAttribute("erordesc", "User not found for");
            model.addAttribute("title", "");
        }
    }
}

```

```

package com.training.karyawan.controller.tymeleaf;

import com.training.karyawan.config.Config;
import com.training.karyawan.entity.oauth.User;
import com.training.karyawan.repository.oauth.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

import java.security.Principal;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

```

```

@Controller
@RequestMapping("/v1/user-register/web/")
public class RegisterConfim {

    @Autowired
    public UserRepository userRepo;

    Config config = new Config();

    @GetMapping(value = { "/index/{tokenotp}"})
    public String index(Model model,@PathVariable String tokenotp) {
        User user = userRepo.findOneByOTP(tokenotp);
        if (null == user) {
            System.out.println("user null: tidak ditemukan");
            model.addAttribute("erordesc", "User not found for code
"+tokenotp);
            model.addAttribute("title", "");
            return "register";
        }
        if(user.isEnabled()){
            model.addAttribute("erordesc", "Akun Anda sudah aktif, Silahkan
melakukan login ");
            model.addAttribute("title", "");
            return "register";
        }

        String today = convertDateToString(new Date());

        String dateToken =
config.convertDateToString(user.getOtpExpiredDate());
        if(Long.parseLong(today) > Long.parseLong(dateToken)) {
            model.addAttribute("erordesc", "Your token is expired. Please
Get token again.");
            model.addAttribute("title", "");
            return "register";
        }
        user.setEnabled(true);
        userRepo.save(user);
        model.addAttribute("title", "Congratulations,
"+user.getUsername()+" , you have successfully registered.");
        model.addAttribute("erordesc", "");
        return "register";
    }

    @GetMapping(value = "/user1")
    public String index() {
        return "user";
    }

    @RequestMapping(value = "/user")
    public Principal user(Principal principal) {

        return principal;
    }

    public String convertDateToString(Date date) {

        DateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmss");

```

```

        String strDate = dateFormat.format(date);
        System.out.println("Date: " + strDate);
        return strDate;
    }
}

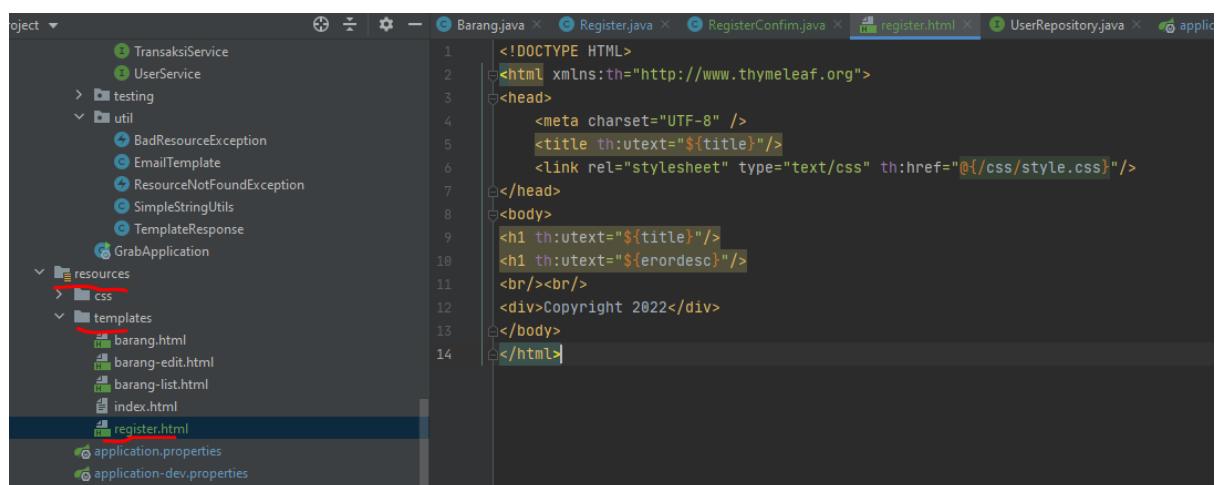
```

g. Step 4- UI Thymeleaf

```

<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8" />
    <title th:utext="${title}">/>
    <link rel="stylesheet" type="text/css"
th:href="@{/css/style.css}"/>
</head>
<body>
<h1 th:utext="${title}">/>
<h1 th:utext="${erordesc}">/>
<br/><br/>
<div>Copyright 2022</div>
</body>
</html>

```



h. Controller register

```

@PostMapping("/register-google-thymeleaf")
public ResponseEntity<Map> saveRegisterByGoogleThymeleaf(@Valid
@RequestBody RegisterModel objModel) throws RuntimeException {
    Map map = new HashMap();
}

```

```

        User user =
userRepository.checkExistingEmail(objModel.getUsername());
        if (null != user) {
            return new ResponseEntity<Map>(templateCRUD.Error("Username
sudah ada"), HttpStatus.OK);
        }
        map = serviceReq.registerByGoogle(objModel);
        //gunanya send email
        Map mapRegister = sendEmailregisterTymeleaf(objModel);
        return new ResponseEntity<Map>(mapRegister, HttpStatus.OK);
    }
}

```

```

40
41
42
43     @PostMapping('/register')
44     public ResponseEntity<Map> saveRegisterManual(@Valid @RequestBody RegisterModel objModel) throws RuntimeException {
45         Map map = new HashMap();
46
47         User user = userRepository.checkExistingEmail(objModel.getUsername());
48         if (null != user) {
49             return new ResponseEntity<Map>(templateCRUD.notFound( objek: "Username sudah ada"), HttpStatus.OK);
50         }
51         map = serviceReq.registerManual(objModel);
52         Map sendOTP = sendEmailregister(objModel);
53         Map sendOTPUri = sendEmailregisterTymeleaf(objModel);
54
55         return new ResponseEntity<Map>(map, HttpStatus.OK);
56     }
57
58     2 usages
      @Autowired

```

i. Test Run Postman

Step 1-Hit api register

Binar Grab / register Copy / register

Save ...

POST localhost:8082/api/user-register/register-tymleaf **Send**

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

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

```

1   {
2     "email": "rikialdipari@gmail.com",
3     "password": "password",
4     "fullname": "riki aldi pari"

```

Body Cookies Headers (18) Test Results Status: 200 OK Time: 6.49 s Size: 3.67 KB Save Response

Pretty Raw Preview Visualize JSON

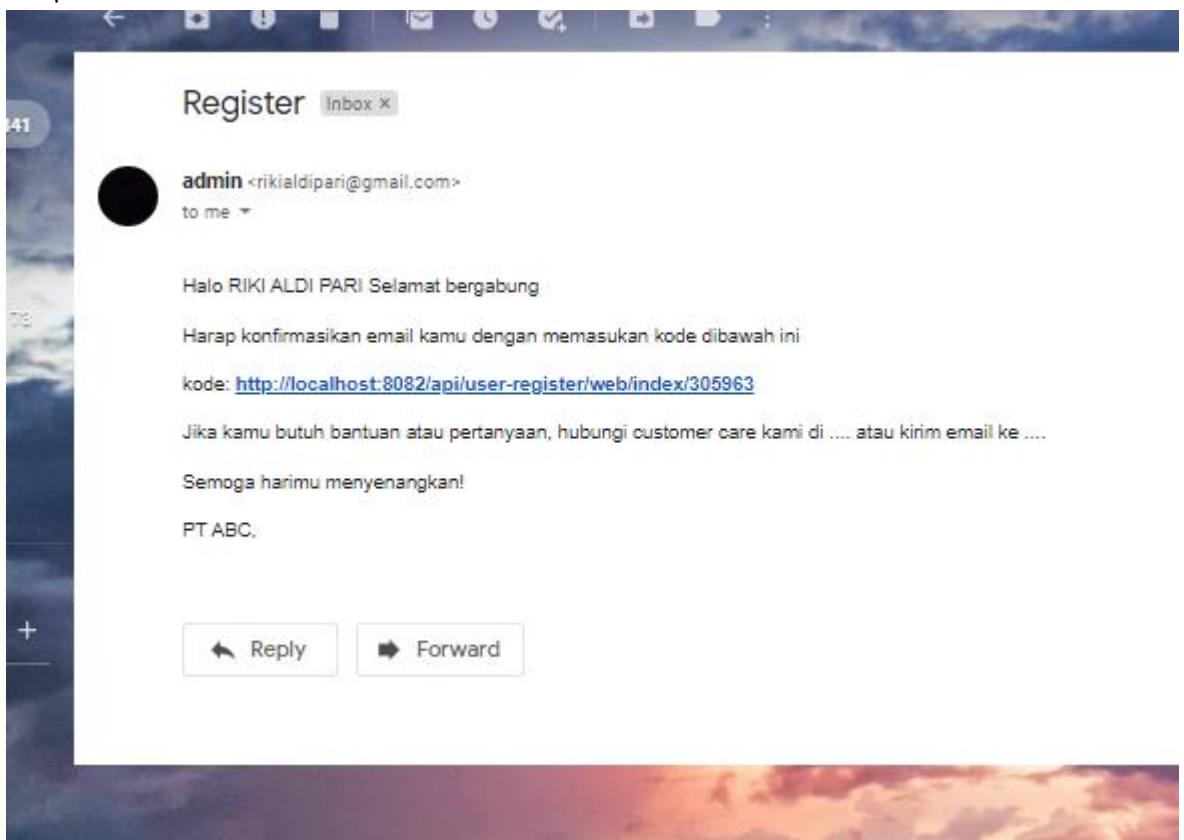
```

1   {
2     "data": {
3       "id": 4,
4       "username": "rikialdipari@gmail.com",
5       "username1": null,
6       "fullname": "riki aldi pari",
7       "otp": "305963",
8       "otpExpiredDate": "2022-03-23T09:20:39.355+00:00",
9       ...

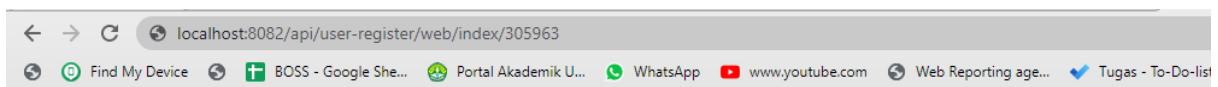
```

Activate Windows
Go to Settings to activate Windows

Step 2-Chek email



Step 3-klik url yang dikirimkan di email, dan akan muncul tampilan berikut :



10. Forget Password : send OTP

Gitlab : <https://github.com/rikialdi/dibimbing>

Branch: 08032022-forget-password

Atau https://gitlab.com/rikialdi/superproof-adam/-/tree/sesi7-forget-password?ref_type=heads

Step 1: Chek email – jika tidak ada maka return email tidak ditemukan

Step 2: Jika email ditemukan – OTP Akan dikirim ke email : Validasi OTP

Step 3: User akan memasukkan OTP untuk dapat melakukan reset Password



a. Forget Password OTP Email

Controller

```
package com.aplikasi.karyawan.controller.oauth;

import com.aplikasi.karyawan.Config;
import com.aplikasi.karyawan.entity.oauth.User;
import com.aplikasi.karyawan.repository.oauth.UserRepository;
import com.aplikasi.karyawan.request.ResetPasswordModel;
import com.aplikasi.karyawan.service.email.EmailSender;
import com.aplikasi.karyawan.service.oauth.UserService;
import com.aplikasi.karyawan.utils.EmailTemplate;
import com.aplikasi.karyawan.utils.Response;
import com.aplikasi.karyawan.utils.SimpleStringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.util.StringUtils;
```

```

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/forget-password/")
public class ForgerPasswordController {

    @Autowired
    private UserRepository userRepository;

    Config config = new Config();

    @Autowired
    public UserService serviceReq;

    @Value("${expired.token.password.minute:}") //FILE_SHOW_RUL
    private int expiredToken;

    @Autowired
    public Response templateCRUD;

    @Autowired
    public EmailTemplate emailTemplate;

    @Autowired
    public EmailSender emailSender;

    @Autowired
    private PasswordEncoder passwordEncoder;

    // Step 1 : Send OTP
    @PostMapping("/send") //send OTP//send OTP
    public Map sendEmailPassword(@RequestBody ResetPasswordModel user) {
        String message = "Thanks, please check your email";

        if (StringUtils.isEmpty(user.getEmail())) return
        templateCRUD.templateError("No email provided");
        User found = userRepository.findOneByUsername(user.getEmail());
        if (found == null) return templateCRUD.notFound("Email not found");
        //throw new BadRequest("Email not found");

        String template = emailTemplate.getResetPassword();
        if (StringUtils.isEmpty(found.getOtp())) {
            User search;
            String otp;
            do {
                otp = SimpleStringUtils.randomString(6, true);
                search = userRepository.findOneByOTP(otp);
            } while (search != null);
            Date dateNow = new Date();
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(dateNow);
        }
    }
}

```

```

        calendar.add(Calendar.MINUTE, expiredToken);
        Date expirationDate = calendar.getTime();

        found.setOtp(otp);
        found.setOtpExpiredDate(expirationDate);
        template = template.replaceAll("\\{\\{\\{PASS_TOKEN}\\}\\}\\}", otp);
        template = template.replaceAll("\\{\\{\\{USERNAME}\\}\\}\\}",

        (found.getUsername() == null ? "" +
            "@UserName"
            :
            "@" + found.getUsername()));

        userRepository.save(found);
    } else {
        template = template.replaceAll("\\{\\{\\{USERNAME}\\}\\}\\}",

        (found.getUsername() == null ? "" +
            "@UserName"
            :
            "@" + found.getUsername()));
        template = template.replaceAll("\\{\\{\\{PASS_TOKEN}\\}\\}\\}",

        found.getOtp());
    }
    emailSender.sendAsync(found.getUsername(), "Chute - Forget
Password", template);

    return templateCRUD.templateSukses("success");
}

//Step 2 : CHek TOKEN OTP EMAIL
@PostMapping("/validate")
public Map checkTokenValid(@RequestBody ResetPasswordModel model) {
    if (model.getOtp() == null) return templateCRUD.notFound("Token is
required");

    User user = userRepository.findOneByOTP(model.getOtp());
    if (user == null) {
        return templateCRUD.templateError("Token not valid");
    }

    return templateCRUD.templateSukses("Success");
}

// Step 3 : lakukan reset password baru
@PostMapping("/change-password")
public Map resetPassword(@RequestBody ResetPasswordModel model) {
    if (model.getOtp() == null) return templateCRUD.notFound("Token is
required");
    if (model.getNewPassword() == null) return
templateCRUD.notFound("New Password is required");
    User user = userRepository.findOneByOTP(model.getOtp());
    String success;
    if (user == null) return templateCRUD.notFound("Token not valid");

    user.setPassword(passwordEncoder.encode(model.getNewPassword().replaceAll(
"\n\s+", "")));
    user.setOtpExpiredDate(null);
}

```

```

        user.setOtp(null);

        try {
            userRepository.save(user);
            success = "success";
        } catch (Exception e) {
            return templateCRUD.templateEror("Gagal simpan user");
        }
        return templateCRUD.templateSukses(success);
    }

}

```

Model Request

```

package com.binar.grab.dao.request;

import lombok.Data;

@Data
public class ResetPasswordModel {
    public String email;

    public String otp;
    public String newPassword;
}

```

Email Template

Class EmailTemplate

```

public String getResetPassword() {

    return "<!doctype html>\n" +
        "<html lang=\"en-US\">\n" +
        "<head>" +
        "<meta content=\"text/html; charset=utf-8\" http-
equiv=\"Content-Type\" />\n" +
        "<title>Reset Password Email Template</title>\n" +
        "<meta name=\"description\" content=\"Reset Password Email
Template.\"> \n" +
        "<style type=\"text/css\"> \n" +
        "a:hover {text-decoration: underline !important;} \n" +
        "</style> \n" +
        "</head>\n" +
        "<body marginheight=\"0\" topmargin=\"0\" marginwidth=\"0\" "
style="margin: 0px; background-color: #f2f3f8;\" leftmargin=\"0\">\n" +

```

```

        "<table cellspacing=\"0\" border=\"0\" cellpadding=\"0\""
width=\"100%\" bgcolor=\"#f2f3f8\"\n" +
        "style=@import
url(fonts.googleapis.com/css?family=Rubik:300,400,500,700|Open+Sans
:300,400,600,700); font-family: 'Open Sans', sans-serif;\">\\n" +
        "<tr>\\n" +
        "<td>\\n" +
        "<table style=\"background-color: #f2f3f8; max-width:670px;
margin:0 auto;\" width=\"100%\" border=\"0\""+
        "align=\"center\" cellpadding=\"0\" cellspacing=\"0\">\\n" +
        "<tr>\\n" +
        "<td style=\"height:80px; \\">&nbsp;</td>\\n" +
        "</tr>\\n" +
        "<tr>\\n" +
        "<td style=\"text-align:center; \\">&gt;\\n" +
        "<a href=\"#\" title=\"logo\" target=\"_blank\">\\n" +
        "<img width=\"60\""

src=\"https://drive.google.com/uc?export=view&id=1NiO5cQZDXA31eFJozu_wrEHUK
4vbzzlo\" title=\"logo\" alt=\"logo\">\\n" +
        "</a>\\n" +
        "</td>\\n" +
        "</tr>\\n" +
        "<tr>\\n" +
        "<td style=\"height:20px; \\">&nbsp;</td>\\n"+
        "</tr>\\n" +
        "<tr>\\n" +
        "<td>\\n" +
        "<table width=\"95%\" border=\"0\" align=\"center\""
cellpadding=\"0\" cellspacing=\"0\""+
        "style=\"max-width:670px;background:#fff; border-radius:3px;
text-align:center;-webkit-box-shadow:0 6px 18px 0 rgba(0,0,0,.06);-moz-box-
shadow:0 6px 18px 0 rgba(0,0,0,.06);box-shadow:0 6px 18px 0
rgba(0,0,0,.06);\\">&gt;\\n" +
        "<tr>\\n" +
        "<td style=\"height:40px; \\">&nbsp;</td>\\n" +
        "</tr>\\n" +
        "<tr>\\n" +
        "<td style=\"padding:0 35px; \\">&gt;\\n" +
        "<h1 style=\"color:#1e1e2d; font-weight:500; margin:0;font-
size:32px;font-family:'Rubik',sans-serif; \\">&gt;Hi {{USERNAME}}, you have
requested verification code for your password.</h1>\\n" +
        "<span "+

        "style=\"display:inline-block; vertical-align:middle;
margin:29px 0 26px; border-bottom:1px solid #cecece;
width:100px;\\"></span>\\n" +
        "<p style=\"color:#455056; font-size:15px;line-height:24px;
margin:0;\\">>\\n" +
        "Please use the verification code below to reset your password
: <br/> </p>" +
        "<strong style=\"font-size:24px; \\">>{{PASS_TOKEN}}</strong>
<br/>\\n" +
        "<p style=\"color:#455056; font-size:15px;line-height:24px;
margin:0;\\">>\\n" +
        "If this action wasn't done by you, please contact us on
<b>rikialdipari@gmail.com</b>. But if it's you, you can ignore this
message. </p>" +
        "</td>\\n"+
        "</tr>\\n"+
        "<tr>\\n" +

```

```

        "<td style=\"height:40px;\"> </td>" +
        "</tr>\n" +
        "</table>\n" +
        "</td>\n" +
        "<tr>\n" +
        "<td style=\"height:20px;\"> </td>\n" +
        "</tr>\n" +
        "<tr>\n" +
        "    <td style=\"text-align:center;\"> </td>\n" +
        "</td>\n" +
        "</tr>\n" +
        "<tr>\n" +
        "    <td style=\"height:80px;\"> </td>\n" +
        "</tr>\n" +
        "</table>\n" +
        "</td>\n" +
        "</tr>\n" +
        "</table>\n" +
        "</body>\n" +
        "</html>\n";
    }
}

```

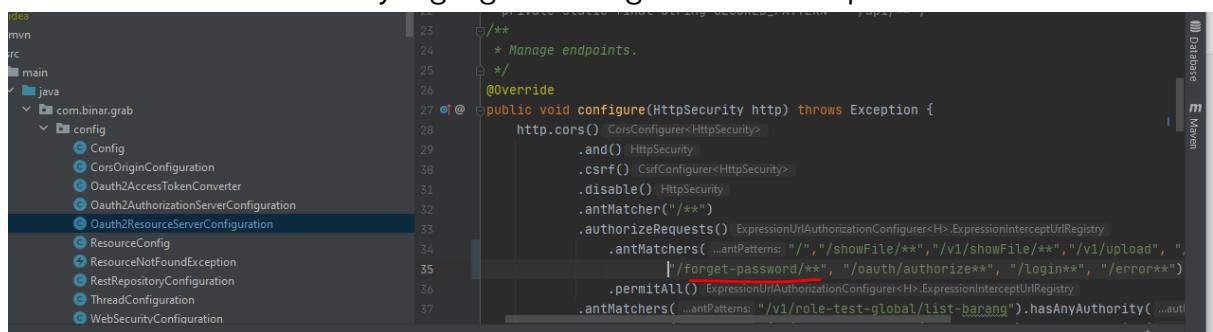
Config

```

public boolean isValidEmail(String email)
{
    String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\.|[a-zA-Z0-9_+&*-]*@[a-zA-Z0-9-]+\\.|[a-zA-Z][a-zA-Z]{2,7})$";
    Pattern pat = Pattern.compile(emailRegex);
    return pat.matcher(email).matches();
}

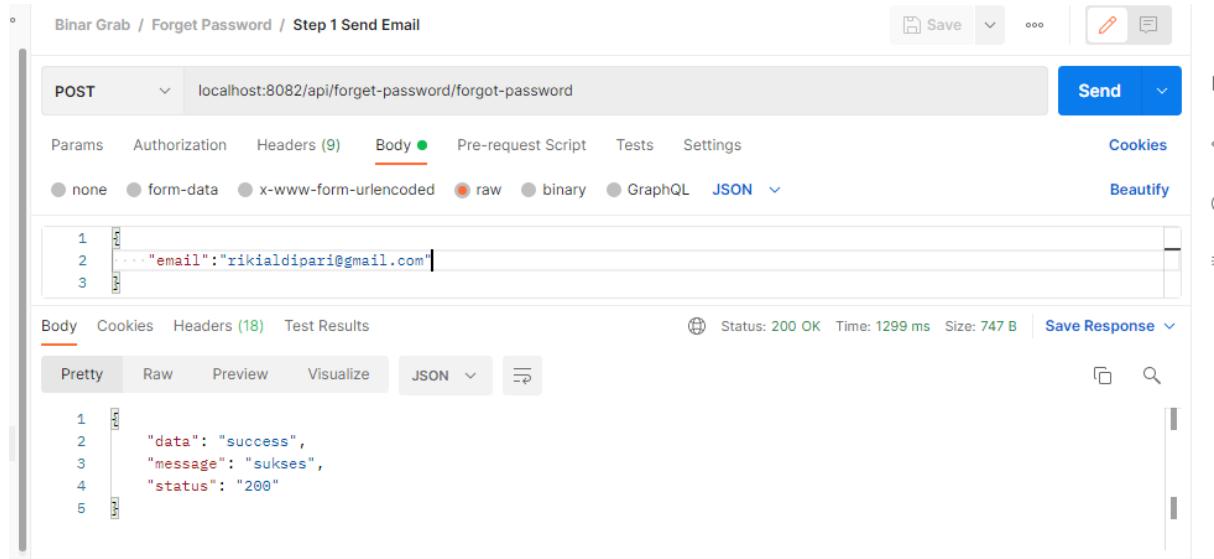
```

Allow URI : tambahkan uri yang digunakan agar diakses tanpa token



b. Test Postman

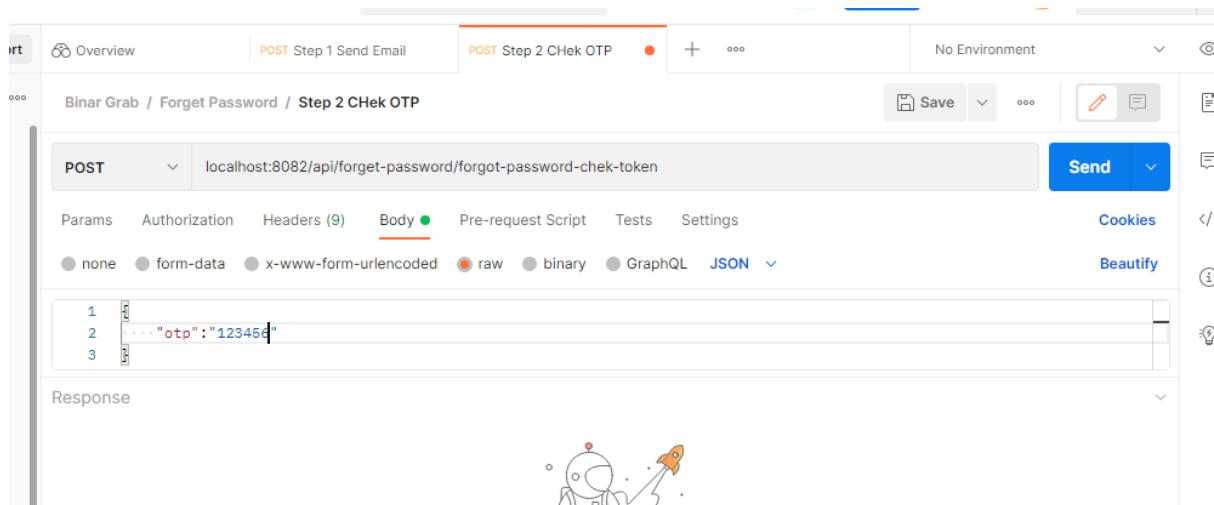
Step 1 send email



Postman screenshot showing the "Step 1 Send Email" request. The request is a POST to `localhost:8082/api/forget-password/forgot-password`. The Body tab is selected, showing a JSON payload with `"email": "rikialdipari@gmail.com"`. The response status is 200 OK with a success message.

```
POST localhost:8082/api/forget-password/forgot-password
{
  "email": "rikialdipari@gmail.com"
}
{
  "data": "success",
  "message": "sukses",
  "status": "200"
}
```

Step 2 : Chek OTP



Postman screenshot showing the "Step 2 CHEK OTP" request. The request is a POST to `localhost:8082/api/forget-password/forgot-password-chek-token`. The Body tab is selected, showing a JSON payload with `"otp": "123456"`. The response area shows a cartoon character with a rocket.

```
POST localhost:8082/api/forget-password/forgot-password-chek-token
{
  "otp": "123456"
}
```

email

Karyawan Aplikasi [Inbox](#) [x](#)

 admin <rikialdipari@gmail.com>
to me ▾



Hi riki aldi pari, you have requested
verification code for your password.

Please use the verification code below to reset your password :

0031

If this action wasn't done by you, please contact us on rikialdipari@gmail.com. But
if it's you, you can ignore this message.

Step-3: Do changes password

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** {{host}}/v1/forget-password/change-password
- Body (JSON):**

```
1 {  
2   "email": "rikialdipari@gmail.com",  
3   "newPassword": "password",  
4   "confirmNewPassword": "password"  
5 }
```
- Status:** 200
- Response Body (Pretty JSON):**

```
1 {  
2   "data": "success",  
3   "message": "Success",  
4   "status": 200  
5 }
```

11. QUERY JPA

a. Error JPA

Pageable tidak bisa menggunakan IN, solusi native query

```
@Query(value = "select uk.* from event_committee_participant uk WHERE uk.id_event = :event and uk.id_event not in (:lisId)", nativeQuery = true)  
public List<EventCommitteeParticipant> getUserDeleted(@Param("event")  
UUID event, @Param("lisId") UUID[] lisId);
```

b. Hati-hati, jika native query, maka deleted_date ikutan ketampil

c. Filter And Sorting With JPA

Controller

```
package com.binar.grab.controller;

import com.binar.grab.config.Config;
import com.binar.grab.model.Barang;
import com.binar.grab.model.Supplier;
import com.binar.grab.model.oauth.User;
import com.binar.grab.repository.BarangRepository;
import com.binar.grab.repository.oauth.UserRepository;
import com.binar.grab.service.SupplierService;
import com.binar.grab.service.oauth.Oauth2UserDetailsService;
import com.binar.grab.util.SimpleStringUtils;
import com.binar.grab.util.TemplateResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;

import java.security.Principal;
import java.util.HashMap;
import java.util.Map;
import com.binar.grab.service.BarangService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/v1/binar/barang")
public class BarangController {

    @Autowired
    public BarangService barangService;

    Config config = new Config();

    @Autowired
    private Oauth2UserDetailsService userDetailsService;

    @Autowired
    public BarangRepository barangRepository;
```

```

    @Autowired
    public TemplateResponse templateResponse;

    @Autowired
    public UserRepository userRepository;

    SimpleStringUtils simpleStringUtils = new SimpleStringUtils();

    @GetMapping("list-barang")// by seller
    public ResponseEntity<Map> listNotif(
        @RequestParam() Integer page,
        @RequestParam(required = true) Integer size,
        @RequestParam(required = false) String nama,// BUY OR SELL : menentukan siapa yang akses
        @RequestParam(required = false) Double priceMin,
        @RequestParam(required = false) Double priceMax,
        @RequestParam(required = false) String orderby,
        @RequestParam(required = false) String ordertype,
        Principal principal) {
        /*
        1.principal : mendapatkan username berdasarkan token user yang akses di client
        2. getShort : shoritng
        */
        Pageable show_data = simpleStringUtils.getShort(orderby, ordertype, page, size);
        Page<Barang> list = null;

        User idUser = getUserIdToken(principal, userDetailsService);
        if (idUser == null) {
            return new
        ResponseEntity<Map>(templateResponse.notFound("User id notfound"), HttpStatus.NOT_FOUND);
        }
        if (nama != null && priceMin !=null && priceMax != null ) {
            list =
        barangRepository.getDataByPriceAndNama(priceMin,priceMax, "%" + nama + "%",show_data);
        } else if ( priceMin !=null && priceMax != null ) {
            list = barangRepository.getDataByPrice(priceMin,priceMax, show_data);
        } else if (nama != null ) {
            list = barangRepository.findByNameLike("%" + nama + "%", show_data);
        } else {
            list = barangRepository.getAllData(show_data);
        }
        return new
        ResponseEntity<Map>(templateResponse.templateSukses(list), new
        HttpHeaders(), HttpStatus.OK);
    }

    public User getUserIdToken(Principal principal,
    OAuth2UserDetailsService userDetailsService) {
        UserDetails user = null;
        String username = principal.getName();
        if (!StringUtils.isEmpty(username)) {

```

```
        user = userDetailsService.loadUserByUsername(username);
    }

    if (null == user) {
        throw new UsernameNotFoundException("User not found");
    }
    User idUser =
userRepository.findOneByUsername(user.getUsername());
    if (null == idUser) {
        throw new UsernameNotFoundException("User name not
found");
    }
    return idUser;

}
```

Tambahkan SimpleStringUtils

```
public Pageable getShort(String orderby, String ordertype, Integer page, Integer size) {
    Pageable show_data;
    if (orderby != null) {
        if (ordertype != null) {
            if (ordertype.equals("desc")) {
                show_data = PageRequest.of(page, size,
Sort.by(orderby).descending());
            } else {
                show_data = PageRequest.of(page, size,
Sort.by(orderby).ascending());
            }
        } else {
            show_data = PageRequest.of(page, size,
Sort.by(orderby).descending());
        }
    } else {
        show_data = PageRequest.of(page, size,
Sort.by("id").descending());
    }
    return show_data;
}
```

Repository

```
public Page<Barang> findByNamaLike(String nama , Pageable pageable);

@Query("select c from Barang c where c.price BETWEEN :priceMin and
:priceMax")// nama class
Page<Barang> getDataByPrice( Double priceMin, Double priceMax,
Pageable pageable);

@Query("select c from Barang c where c.price BETWEEN :priceMin and
:priceMax and c.nama like :nama")// nama class
Page<Barang> getDataByPriceAndNama( Double priceMin, Double
priceMax, String nama, Pageable pageable);
```

d. Criteria QUERY JPA

Perlu ditambahkan di pom.xml

```
<dependency>
<groupId>org.hibernate.validator</groupId>
<artifactId>hibernate-validator</artifactId>
<version>6.0.13.Final</version>
</dependency>
```

Dan jPA diganti

```
public interface EmployeeRepository extends JpaRepository<Employee, Long>, JpaSpecificationExecutor<Employee> {
    import org.springframework.data.repository.PagingAndSortingRepository;
    import org.springframework.data.repository.query.Param;
    import org.springframework.stereotype.Repository;
    import java.util.UUID;
    6 usages
    @Repository
    //public interface EmployeeRepository extends PagingAndSortingRepository<Employee, Long> {
        public interface EmployeeRepository extends JpaRepository<Employee, Long>, JpaSpecificationExecutor<Employee> {
            //JPA Query
            3 usages
        }
    }
}
```

Controller

```
@GetMapping(value = {"list", "/list/"})
public ResponseEntity<Map> listRedeemCoinsHistory(
    @RequestParam() Integer page,
    @RequestParam(required = true) Integer size,
    // @RequestParam(required = false) String orderby,
    // @RequestParam(required = false) String ordertype,
    @RequestParam(required = false) Long userId,
    @RequestParam(required = false) UUID voucherId) {

    try {
        // Pageable show_data =
        simpleStringUtils.getShort(orderby, ordertype, page, size);
        Page<RedeemCoins> redeemCoins = null;
        Specification<RedeemCoins> spec = ((root, query,
        criteriaBuilder) -> {
            List<Predicate> predicates = new ArrayList<>();
            if(userId!=null){
                predicates.add(criteriaBuilder.equal(root.get("user").get("id"),userId));
            }
            if(voucherId!=null){
                predicates.add(criteriaBuilder.equal(root.get("voucher").get("id"), voucherId));
            }
            return criteriaBuilder.and(predicates.toArray(new Predicate[0]));
        });
        PageRequest pageRequest = PageRequest.of(page, size);
        Page<RedeemCoins> list =
        redeemCoinsRepository.findAll(spec,pageRequest);

        return new ResponseEntity<Map>(response.sukses(list), new HttpHeaders(), HttpStatus.OK);
    } catch (Exception e) {
```

```

        return new
ResponseEntity<Map>(response.Error(e.getMessage()), new
HttpHeaders(), HttpStatus.OK);
    }
}

```

Repository update

```

public interface RedeemCoinsRepository extends
JpaRepository<RedeemCoins, UUID>,
JpaSpecificationExecutor<RedeemCoins> {

```

Criteria query dengan like and lowercase

```

predicates.add(criteriaBuilder.like(criteriaBuilder.lower(root.get("event"))
.get("name")), "%" + eventName.toLowerCase() + "%"); //like
gimana?

}

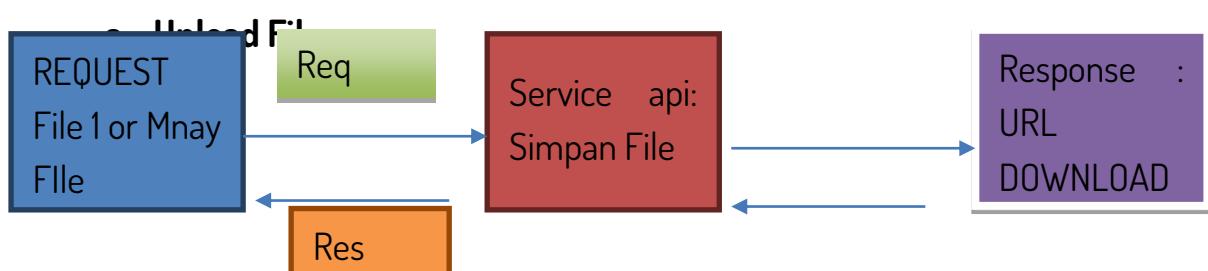
ServiceLookup finalChekTYPE_CERTIFICATE = chekTYPE_CERTIFICATE;
Specification<IssuedCertificate> spec = ((root, query, criteriaBuilder) -> {
List<Predicate> predicates = new ArrayList<>();
predicates.add(criteriaBuilder.equal(root.get("issuedTo"), chekCredentialUser));
predicates.add(criteriaBuilder.equal(root.get("status"), Config.STATUS_REVOKED));
if(eventName!=null && !Strings.isEmpty(eventName)){
    predicates.add(criteriaBuilder.like(criteriaBuilder.lower(root.get("event").get("name")), "%" + eventName.toLowerCase() + "%"));
}
if(finalChekTYPE_CERTIFICATE !=null){
    predicates.add(criteriaBuilder.equal(root.get("certificate").get("typeCertificate"), finalChekTYPE_CERTIFICATE));
}
return criteriaBuilder.and(predicates.toArray(new Predicate[0]));
});

PageRequest pageRequest = PageRequest.of(name.size);

```

12. File Handling/File Upload /Download File

Branch : 11032022-file-handling

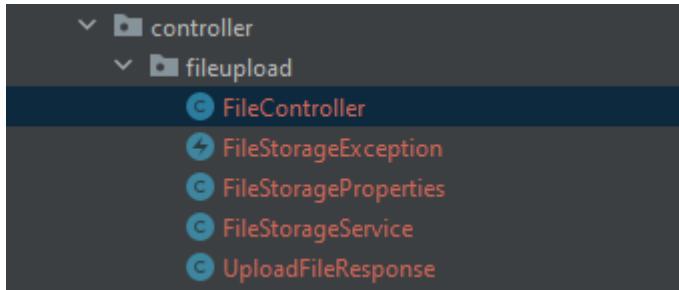


b. Service api

Application.properties

```
spring.servlet.multipart.max-file-size=5MB  
spring.servlet.multipart.max-request-size=5MB  
app.uploadto.cdn=./cdn/ atau app.uploadto.cdnD://folder/subfolder
```

Class



Berikut code program :

```
package com.aplikasi.karyawan.controller.fileupload;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.core.io.Resource;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.MediaType;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.*;  
import org.springframework.web.multipart.MultipartFile;  
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
  
import javax.servlet.http.HttpServletRequest;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.text.SimpleDateFormat;  
import java.util.Arrays;  
import java.util.Date;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@RestController  
@EnableCaching  
public class FileController {  
    private static final Logger logger =  
        LoggerFactory.getLogger(FileController.class);  
}
```

```

    @Value("${app.uploadto.cdn}")//FILE_SHOW_RUL
    private String UPLOADED_FOLDER;

    @Autowired
    private FileStorageService fileStorageService;

    @RequestMapping(value = "/v1/upload", method = RequestMethod.POST, consumes =
{ "multipart/form-data", "application/json"})
    public UploadFileResponse uploadFile(@RequestParam("file") MultipartFile file
throws IOException {

    Date date = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat("ddMyyyyhhmmss");
    // String strDate = formatter.format(date);
    String strDate = String.valueOf(UUID.randomUUID());

    String nameFormat=
file.getOriginalFilename().substring(file.getOriginalFilename().lastIndexOf(".") )
    if(nameFormat.isEmpty()){
        nameFormat = ".png";
    }
    String fileName = UPLOADED_FOLDER + strDate + nameFormat;

    String fileNameforDOWnload = strDate + nameFormat;
    Path TO = Paths.get(fileName);

    //validasi hanya boleh PNG
    if(!file.getContentType().equals("image/png")){
        return null;// eror
    }

    try {
        Files.copy(file.getInputStream(), TO); // pengolahan upload disini :

    } catch (Exception e) {
        e.printStackTrace();
        return new UploadFileResponse(fileNameforDOWnload, null,
                file.getContentType(), file.getSize(), e.getMessage());
    }

    String fileDownloadUri =
ServletUriComponentsBuilder.fromCurrentContextPath()
        .path("/v1/showFile/")
        .path(fileNameforDOWnload)
        .toUriString();

    return new UploadFileResponse(fileNameforDOWnload, fileDownloadUri,
            file.getContentType(), file.getSize(), "false");
}

//    public String getPAth(String type) {
//        try {
//            if (type.toLowerCase().equals("product")) {
//                File theDir = new File(UPLOADED_FOLDER_PRODUCT);
//            }
//        }
//    }

```

```

//          if (!theDir.exists()) {
//              theDir.mkdirs();
//              return UPLOADED_FOLDER_PRODUCT;
//          }
//          return UPLOADED_FOLDER;
//      }

//  } catch (Exception e) {
//      logger.info("Error create folder" + e);
//  }
//  return UPLOADED_FOLDER;
// }

@GetMapping("v1/showFile/{fileName:.+}")
public ResponseEntity<Resource> showFile(@PathVariable String fileName,
HttpServletResponse request) { // Load file as Resource : step 1 load path lokasi
name file
    Resource resource = fileStorageService.loadFileAsResource(fileName);
    // Try to determine file's content type
    String contentType = null;
    try {
        // dapatan URL download
//        System.out.println("resource.getFile().getAbsolutePath" +
resource.getFile().getAbsolutePath());
        contentType =
request.getServletContext().getMimeType(resource.getFile().getAbsolutePath());
    } catch (IOException ex) {
        logger.info("Could not determine file type.");
    }
    // Fallback to the default content type if type could not be determined
    if (contentType == null) {
        contentType = "application/octet-stream"; // type .json
    }
//    System.out.println("filename=2=" + HttpHeaders.CONTENT_DISPOSITION);
//    System.out.println("filename=3=" + resource.getFilename());
//    System.out.println("filename=3=" + resource);
    return ResponseEntity.ok()
        .contentType(MediaType.parseMediaType(contentType))
        .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\"" +
resource.getFilename() + "\"")
        .body(resource);
}

@PostMapping("v1/uploadMultipleFiles")
public List<UploadFileResponse> uploadMultipleFiles(@RequestParam("files")
MultipartFile[] files) throws IOException {
    return Arrays.asList(files)
        .stream()
        .map(file -> {
            try {
                // step 1 : call method upload
                return uploadFile(file);
            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        })
}

```

```

        .collect(Collectors.toList());
    }

private File multipartToFile(MultipartFile upload, String routeName) {
    String base = "";

    logger.info(String.format("Trying upload file: %s",
upload.getOriginalFilename()));

    File file = new File(base + upload.getOriginalFilename());

    try {
        logger.info(String.format("Saving uploaded file to: '%s'",
file.getAbsolutePath()));
        FileOutputStream fos = new FileOutputStream(file);
        fos.write(upload.getBytes());
        fos.close();
    } catch (IOException e) {
        logger.error(String.format("Error: POST|UPLOAD %s", routeName), e);
    }

    return file;
}

private File multipartToFile(MultipartFile upload) {
    return multipartToFile(upload, UPLOADED_FOLDER);
}

}

```

```

package com.binar.grab.controller.fileupload;

public class FileStorageException extends RuntimeException {
    public FileStorageException(String message) {
        super(message);
    }

    public FileStorageException(String message, Throwable cause) {
        super(message, cause);
    }
}

```

```

package com.binar.grab.controller.fileupload;

import org.springframework.beans.factory.annotation.Value;
import
org.springframework.boot.context.properties.ConfigurationProperties;

@ConfigurationProperties(prefix = "file")
public class FileStorageProperties {
    @Value("${app.uploadto.cdn}") //FILE_SHOW_RUL
}

```

```

    private String uploadDir;

    public String getUploadDir() {
        return uploadDir;
    }

    public void setUploadDir(String uploadDir) {
        this.uploadDir = uploadDir;
    }
}

```

```

package com.binar.grab.controller.fileupload;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.net.MalformedURLException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
import java.text.SimpleDateFormat;
import java.util.Date;

@Service
public class FileStorageService {

    private final Path fileStorageLocation;

    Date date = new Date();
    SimpleDateFormat formatter = new
    SimpleDateFormat("ddMyyyyhhmmss");
    String strDate = formatter.format(date);

    @Autowired
    public FileStorageService(FileStorageProperties
fileStorageProperties) {
        this.fileStorageLocation =
Paths.get(fileStorageProperties.getUploadDir())
        .toAbsolutePath().normalize();

        try {
            Files.createDirectories(this.fileStorageLocation);
        } catch (Exception ex) {
            throw new FileStorageException("Could not create the
directory where the uploaded files will be stored.", ex);
        }
    }
}

```

```

public String storeFile(MultipartFile file) {
    // Normalize file name
    String fileName =
StringUtils.cleanPath(file.getOriginalFilename());
    String date_name = strDate+file;
    System.out.println("ini=="+fileName);
    try {
        // Check if the file's name contains d characters
        if(fileName.contains("../")) {
            throw new FileStorageException("Sorry! Filename
contains d path sequence " + fileName);
        }

        // Copy file to the target location (Replacing existing
file with the same name)
        Path targetLocation =
this.fileStorageLocation.resolve(fileName);
        Files.copy(file.getInputStream(), targetLocation,
StandardCopyOption.REPLACE_EXISTING);
        System.out.println("ini==2="+fileName);
        System.out.println("ini==3="+targetLocation);
        return fileName;
    } catch (IOException ex) {
        throw new FileStorageException("Could not store file " +
fileName + ". Please try again!", ex);
    }
}

public Resource loadFileAsResource(String fileName) {
    try {
        Path filePath =
this.fileStorageLocation.resolve(fileName).normalize();

        Resource resource = new UrlResource(filePath.toUri());
        if(resource.exists()) {
            return resource;
        } else {
            System.out.println("ini saya= "+filePath);
            System.out.println("ini saya 2= "+filePath.toUri());
            System.out.println("ini saya 3=
"+filePath.toAbsolutePath());
            throw new FileStorageException("File not found " +
fileName);
        }
    } catch (MalformedURLException ex) {
        throw new FileStorageException("File not found " +
fileName, ex);
    }
}
}

```

```

package com.binar.grab.controller.fileupload;

import lombok.Data;

@Data

```

```
public class UploadFileResponse {  
    private String fileName;  
    private String fileDownloadUri;  
    private String fileType;  
    private long size;  
    private String eror;  
  
    public UploadFileResponse(String fileName, String  
fileDownloadUri, String fileType, long size, String eror) {  
        this.fileName = fileName;  
        this.fileDownloadUri = fileDownloadUri;  
        this.fileType = fileType;  
        this.size = size;  
        this.eror = eror;  
    }  
}
```

c. Tambahkan pada class main

```
3 import ...  
4  
5     import org.springframework.boot.context.properties.EnableConfigurationProperties;  
6  
7     @SpringBootApplication  
8     @EnableConfigurationProperties({  
9         FileStorageProperties.class  
10    })  
11    public class GrabApplication {  
12  
13        public static void main(String[] args) {  
14            SpringApplication.run(GrabApplication.class, args);  
15        }  
16    }  
17 }  
  
@EnableConfigurationProperties({  
    FileStorageProperties.class  
})
```

d. Testing

Upload 1 file

The screenshot shows the Postman application interface. At the top, there is a header bar with the text "POST" and the URL "{{host}}/api/v1/upload". Below the header, there are tabs for "Params", "Authorization", "Headers (10)", "Body", "Pre-request Script", "Tests", and "Settings". The "Body" tab is currently selected, indicated by an orange underline. Under the "Body" tab, there are two radio button options: "none" (gray), "form-data" (orange, selected), and "x-www-form-urlencoded" (gray). Below these are three buttons: "raw" (gray), "binary" (gray), and "GraphQL" (gray). The main body area has a table with columns "KEY", "VALUE", and "DESCRIPTION". There is one row in the table with a checked checkbox next to "file", and the value is "Capture.PNG". Below the table, there are tabs for "Body", "Cookies", "Headers (3)", and "Test Results". The "Body" tab is selected. On the right side of the interface, there is a status bar showing "Status: 200 OK" and "Time: 83 m". At the bottom of the body area, there are four buttons: "Pretty" (selected), "Raw", "Preview", and "Visualize". To the right of these buttons is a "JSON" dropdown menu with a red arrow pointing to it. The JSON response is displayed below, showing a single object with properties: fileName, fileDownloadUri, fileType, size, and error.

```
1  {
2    "fileName": "1292021112131Capture.PNG",
3    "fileDownloadUri": "http://localhost:8080/api/v1/showFile/1292021112131Capture.PNG",
4    "fileType": "image/png",
5    "size": 23474,
6    "error": "false"
7 }
```

Upload many file

The screenshot shows the Postman interface with the following details:

- Request URL:** POST {{host}}/api/v1/uploadMultipleFiles
- Method:** POST
- Body:** form-data (selected)
- Params:** None
- Headers:** (10)
- Body Content:**

KEY	VALUE	DESCRIPTION
files	rm-tools.keystore	
files	VipFlaggingService.java	
Key	Value	Description
- Test Results:** Status: 200 OK Time: 774 ms Size: 116
- Body View:** Pretty, Raw, Preview, Visualize, JSON (selected), Copy
- JSON Response:**

```
1 [  
2 {  
3   "fileName": "129202110916rm-tools.keystore",  
4   "fileDownloadUri": "http://localhost:8080/api/v1/showFile/129202110916rm-tools.keystore",  
5   "fileType": "application/octet-stream",  
6   "size": 2258,  
7   "error": "false"  
8 },
```

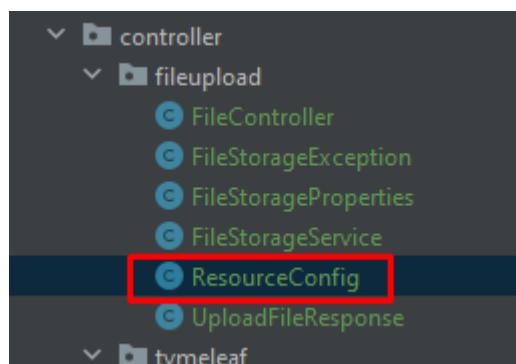
Download file

The screenshot shows the Postman interface with a collection named "BINAR / file upload / file show Copy". A GET request is selected with the URL `({{host}})/api/v1/showFile/1292021104943testlog.log`. The "Params" tab is active, showing a single parameter "Key" with the value "Value". The "Body" tab is also visible. Below the request, the response body is displayed as a log file:

```
2021-09-10 08:25:42.602 INFO 4576 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 411ms. Found 2 repository interfaces.
2021-09-10 08:25:44.630 INFO 4576 --- [main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration' of type [org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration$$EnhancerBySpringCGLIB$$a11b3df] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2021-09-10 08:25:45.895 INFO 4576 --- [main] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 Starting...
2021-09-10 08:25:46.623 INFO 4576 --- [main] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 Start completed.
```

e. Download file tampa download / show file only

Add class



```
package com.binar.grab.controller.fileupload;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegi
```

```

try;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

//step 1
@Configuration
public class ResourceConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**").allowedMethods()
            .allowedMethods("HEAD", "GET", "PUT", "POST",
"DELETE", "PATCH");
    }

    @Override // cara call endpoint :
localhost:9090/api/showFile/namafile.png:
    public void addResourceHandlers(ResourceHandlerRegistry registry)
{
    //
registry.addResourceHandler("/uploads/**").addResourceLocations("file
:uploads/");

registry.addResourceHandler("/showFile/**").addResourceLocations("fil
e:cdn/");
}
}

```

Testing

localhost:9090/api/showFile/namafile.png:

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8082/api/showFile/1132022082313Cbiogdata.PNG
- Params:** (highlighted)
- Headers:** (20)
- Body:** (raw)
- Pre-request Script:** (green dot)
- Tests:**
- Settings:**
- Cookies:**
- Query Params:** (highlighted)

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description	...	

- Response Headers:**
 - Status: 200 OK
 - Time: 1490 ms
 - Size: 304.16 KB
 - Save Response
- Response Body:** (image of a university building entrance)

f. Jika implementasi web Flux,

Pom xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
```

Tidak bisa menggunakan

```
//@EnableWebFlux// : eror : Error creating bean with name
'org.springframework.web.reactive.config.DelegatingWebFluxConfigurati
on': The Java/XML config for Spring MVC and Spring WebFlux cannot
both be enabled, e.g. via @EnableWebMvc and @EnableWebFlux, in the
same application.
```

Namun berikut penerapannya, dan belum di testing

```
@RequestMapping(value = "/v1/upload/flux", method =
RequestMethod.POST, consumes = {"multipart/form-data",
"application/json"})
public Mono<UploadFileResponse>
uploadFileFlux(@RequestParam("file") MultipartFile file) throws
IOException {

    Date date = new Date();
    SimpleDateFormat formatter = new
SimpleDateFormat("ddMyyyyhhmmss");
//        String strDate = formatter.format(date);
    String strDate = String.valueOf(UUID.randomUUID());
    String nameFormat=
file.getOriginalFilename().substring(file.getOriginalFilename().lastI
ndexOf("."));
    if(nameFormat.isEmpty()){
        nameFormat = ".png";
    }
    String fileName = UPLOADED_FOLDER + strDate + nameFormat;

    String fileNameforDOnload = strDate + nameFormat;
    Path TO = Paths.get(fileName);

    //validasi hanya boleh PNG
    if(!file.getContentType().equals("image/png")){
        return null;// eror
    }

    try {
        Files.copy(file.getInputStream(), TO); // pengolahan
    } catch (IOException e) {
        return Mono.error(e);
    }
}
```

```

upload disini :

    } catch (Exception e) {
        e.printStackTrace();
        return Mono.just(new
UploadFileResponse(fileNameforDOWnload, null,
                    file.getContentType(), file.getSize(),
e.getMessage()));
    }

    String fileDownloadUri =
ServletUriComponentsBuilder.fromCurrentContextPath()
    .path("/v1/showFile/")
    .path(fileNameforDOWnload)
    .toUriString();

    return Mono.just(new UploadFileResponse(fileNameforDOWnload,
fileDownloadUri,
                    file.getContentType(), file.getSize(), "false"));
}

@GetMapping("v1/showFile/flux/{fileName:.+}")
public Mono<ResponseEntity<Resource>> showFileFlux(@PathVariable
String fileName, HttpServletRequest request) { // Load file as
Resource : step 1 load path lokasi name file
    Resource resource =
fileStorageService.loadFileAsResource(fileName);
    // Try to determine file's content type
    String contentType = null;
    try {
        // dapatan URL download
//        System.out.println("resource.getFile().getAbsolutePath" +
+ resource.getFile().getAbsolutePath());
        contentType =
request.getServletContext().getMimeType(resource.getFile().getAbsolut
ePath());

    } catch (IOException ex) {
        logger.info("Could not determine file type.");
    }
    // Fallback to the default content type if type could not be
determined
    if (contentType == null) {
        contentType = "application/octet-stream"; // type .json
    }
//    System.out.println("filename=2=" +
HttpHeaders.CONTENT_DISPOSITION);
//    System.out.println("filename=3=" + resource.getFilename());
//    System.out.println("filename=3=" + resource);
    return Mono.just( ResponseEntity.ok()
                    .contentType(MediaType.parseMediaType(contentType))
                    .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;
filename=\" + resource.getFilename() + "\"")
                    .body(resource));
}

```

g. Branch dan postman lain

Postman : <https://www.getpostman.com/collections/c2f47ba653cabcaf30b7>

Github : 141122-file-upload url : <https://gitlab.com/rikialdi/binar5-spring.git>

13. Deploy Project ke Heroku menggunakan Github

a. Buat akun Heroku

<https://id.heroku.com/login>

b. Buat Akun Github

<https://github.com/login>

deploy project

step 1: git reset

step 2 : git init

step 3 : git add .

step 4 : git commit -m "first commit"

step 5 : git branch -M master

Pertama2 : masuk disini untuk generate token

<https://github.com/settings/tokens>

git remote add origin <https://github.com/rikialdi/test.git>

or

git remote set-url origin <git@github.com:rikialdi/test.git>

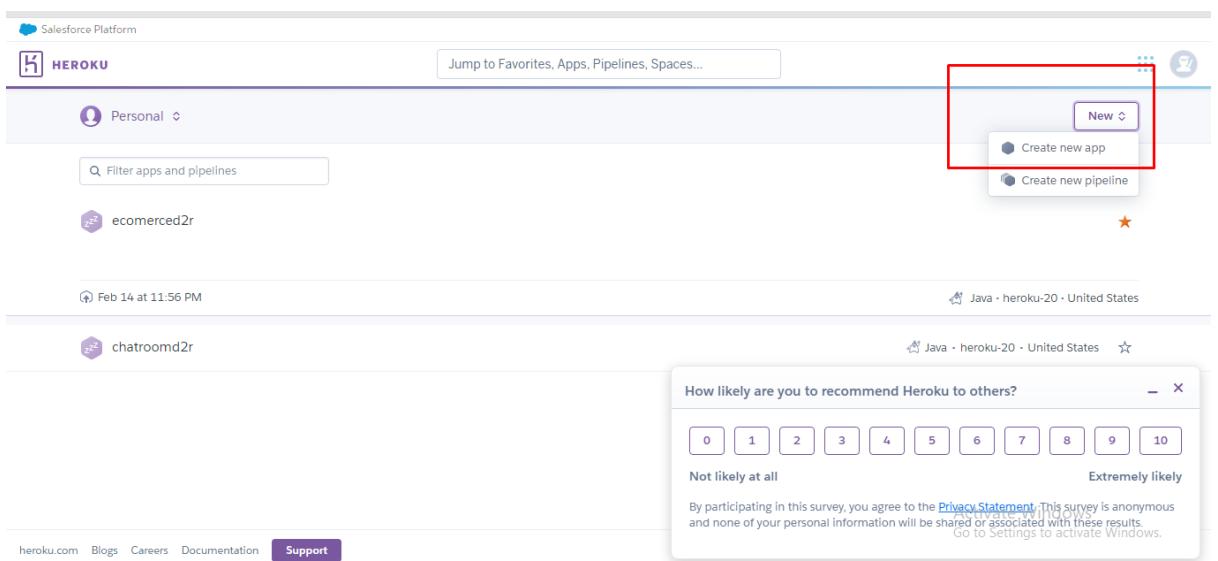
git remote set-url origin [https://rikialdi:\[passwordtoken\]@github.com/rikialdi/test.git](https://rikialdi:[passwordtoken]@github.com/rikialdi/test.git)

step 7: git push -u origin master

1. git remote -v (for checking current repository)

c. Masuk Ke akun Heroku, untuk deploy aplikasi

<https://dashboard.heroku.com/apps>



Buat nama aplikasi kalian

Jump to Favorites, Apps, Pipelines, Spaces...

Create New App

App name

✓

binar-test is available

Choose a region

Add this app to a pipeline

Pipelines form a continuous deployment workflow and enable additional features. [Learn more](#).

[+ Create new pipeline](#)

Name the pipeline Required

✓

Choose a stage to add this app to

staging

This app will be added to **staging** In blnar-test

Create app

Tampilan setelah membuat aplikasi, jika ingin menghubungkan ke github, silahkan pilih github seperti gambar dibawah ini

The screenshot shows the Heroku dashboard for a specific app. At the top, it says "Connected to a pipeline" and "Assigned to staging in blinar-test". Below this, under "Deployment method", there are three options: "Heroku Git" (disabled), "GitHub" (selected and highlighted with a red box), and "Container Registry". On the right side, there's a section for "Deploy using Heroku Git" with instructions to use the CLI or a GUI tool like git. It also shows a terminal window with the command "\$ heroku login". Further down, there's a "Create a new Git repository" section and an "Activate Windows" link.

Pilih tampil repo, Akan tampil semua repo pada akun github.

This screenshot shows the same Heroku dashboard as above, but with a different view. It highlights the "GitHub" option under "Deployment method" and shows the "Connect to GitHub" section. It includes a search bar for "repo-name" with "rklaidl" entered, a "Search" button, and a list of repositories: "rklaidl/rest-api-mysql-spring-boot" and "rklaidl/tutorial_Report_3 7.0". On the right, there's a "Connect to Windows" link.

Tampilan jika sudah terhubung.

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic follows the instructions [here](#).

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. Deployes happen automatically branch is always in a deployable state and any tests have passed before you push. [Learn more](#).

Choose a branch to deploy

Wait for CI to pass before deploy
Only enable this option if you have a Continuous Integration service configured on your repo.

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy



Pada gambar diatas, dapat dilihat pada tombol enable automatic deploy, digunakan untuk deploy otomatis menggunakan ci-cd.

Atau dengan deploy manual dengan menekan tombol deploy branch.

Berikut tampilan gambar berhasil deploy

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

Receive code from GitHub

Build relasitable-oauth2 e59522a4

Release phase

Deploy to Heroku

Your app was successfully deployed.

Activate Window
Go to Settings to activ

Terms of Service Privacy Cookies

d. Membuat database di heroku

Silahkan akses <https://data.heroku.com/>

The screenshot shows the Heroku Datastores interface. At the top, there's a navigation bar with icons for Find My Device, BOSS - Google Sheets, Portal Akademik U... (with a green circular icon), WhatsApp, and Salesforce Platform. Below the navigation bar, the word "DATA" is displayed in a purple box. The main content area has tabs for "Datastores" and "Dataclips". A table lists three database instances:

Name	Plan	Application
postgresql-horizontal-15507	hobby-dev	chatroomd2r
postgresql-encircled-71448	hobby-dev	[Redacted]
postgresql-animated-20821	hobby-dev	ecomerced2r

Silahkan klik salah satu database anda

The screenshot shows the detailed view of the postgresql-encircled-71448 database. The top navigation bar is identical to the previous screenshot. The main page title is "postgresql-encircled-71448". Below the title, it shows the service is heroku-postgresql, plan is hobby-dev, and the billing app is binar-test. There are tabs for Overview, Durability, Settings, and Dataclips. The Overview section includes a "HEALTH" status indicator showing "Available". Below that, it lists PRIMARY (Yes), VERSION (13.6), CREATED (2 minutes ago), MAINTENANCE (Unsupported), ROLLBACK (Unsupported). The utilization section shows "10 of 20 CONNECTIONS" and "47 of 10,000 ROWS IN COMPLIANCE". To the right, it shows a DATA SIZE of 8.7 MB and 14 TABLES. A red box highlights the "CONNECTIONS" and "ROWS IN COMPLIANCE" sections.

Pada gambar diatas menjelaskan limit untuk database posgres gratis di heroku.

Silahkan klik tombol dibawah ini untuk melihat credential database , tujuan agar bisa diakses remote oleh aplikasi local kita.

The screenshot shows the Heroku Datastore settings page for a PostgreSQL database named 'heroku-postgresql'. The 'Settings' tab is active. In the 'Database Credentials' section, there is a 'View Credentials...' button which is highlighted with a red box.

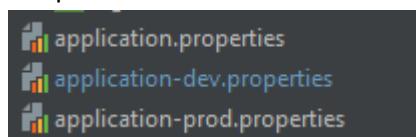
Berikut tampilan credential

This screenshot shows the 'View Credentials...' modal window. It contains the following information:

- Host:** ec2-18-210-191-5.compute-1.amazonaws.com
- Database:** dvch7rlkgrl4i
- User:** ydtyecrssfbazz
- Port:** 5432
- Password:** efe7049a7247bef02e99208410c2f2ad038e2a29e1b94c295d2af0aaca7d8a40
- URI:** postgres://ydtyecrssfbazz:efe7049a7247bef02e99208410c2f2ad038e2a29e1b94c295d2af0aaca7d8a40@ec2-18-210-191-5.compute-1.amazonaws.com:5432/dvch7rlkgrl4i
- Heroku CLI:** heroku pg:psql postgresql-encircled-71448 --app binar-test

e. Membuat env prod/staging di spring boot

Step 1- buatlah 3 buah env seperti gambar dibawah ini



Step 2- isi dari application.properties

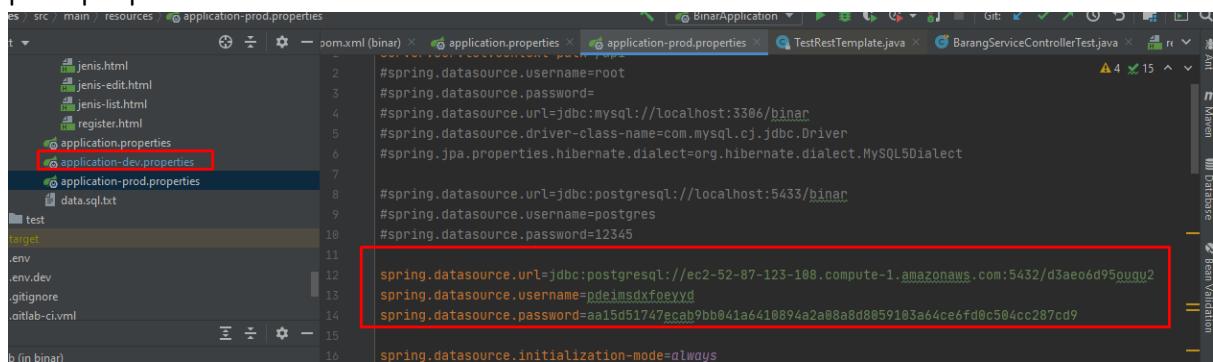
This screenshot shows the 'application.properties' file in an IDE. The line 'spring.profiles.active=prod' is highlighted with a red box. The file also contains other properties like 'jenis.html', 'jenis-edit.html', 'jenis-list.html', and 'register.html'.

Step 3- isi dari file

Application-dev.properties : mengarah ke db DEV

Application-prod.properties: mengarah ke db PROD heroku

Step 4- copy paste credential database di heroku ke application-prod.properties



```
#spring.datasource.username=root
#spring.datasource.password=
#spring.datasource.url=jdbc:mysql://localhost:3306/binar
#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

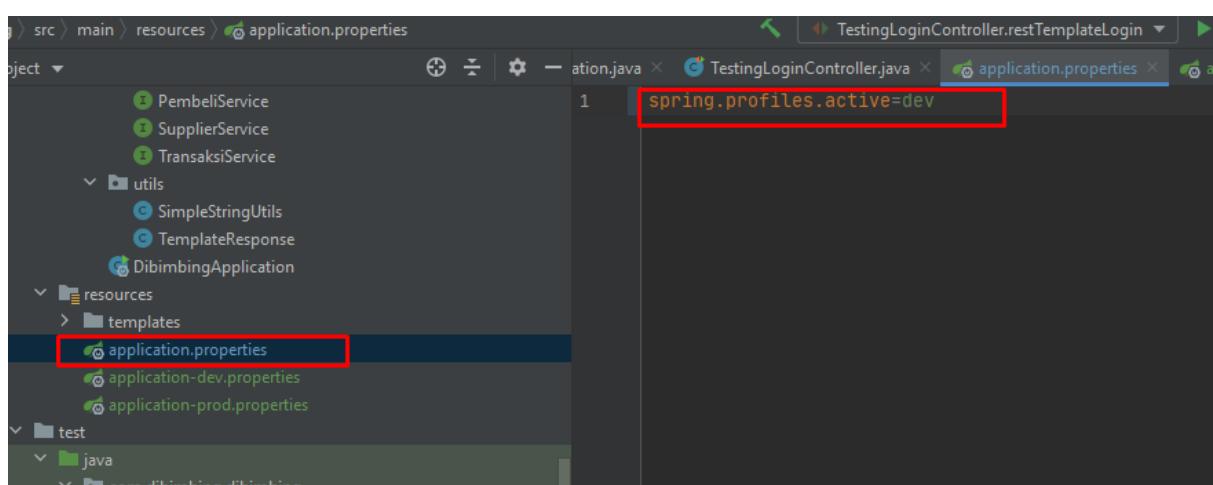
#spring.datasource.url=jdbc:postgresql://ec2-52-87-123-108.compute-1.amazonaws.com:5432/d3aeo6d95ougy2
#spring.datasource.username=pgimsdxfoeyd
#spring.datasource.password=aa15d51747ecab9bb041a6410894a2a08a8d8059103a64ce6fd0c504cc287cd9
#spring.datasource.initialization-mode=always
```

Bagaimana cara melihat env yang sedang aktif ?

Semua di control di file application.properties di setting di variabel spring.profiles.active=dev

Bagaimana mengetahui env yang sedang aktif?

Pada gambar dibawah ini menunjukkan bahwa env yang sedang aktif adalah env dev.



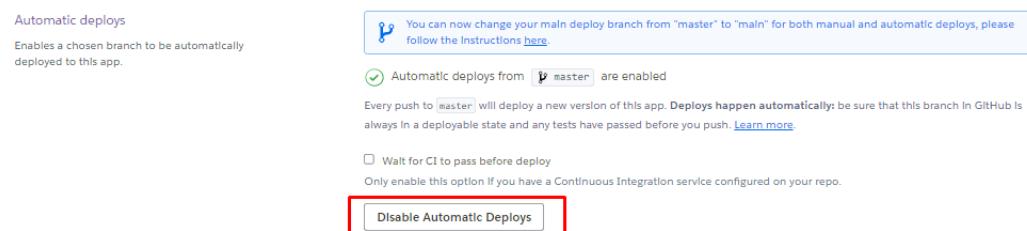
```
spring.profiles.active=dev
```

f. Tambahkan pada pom.xml

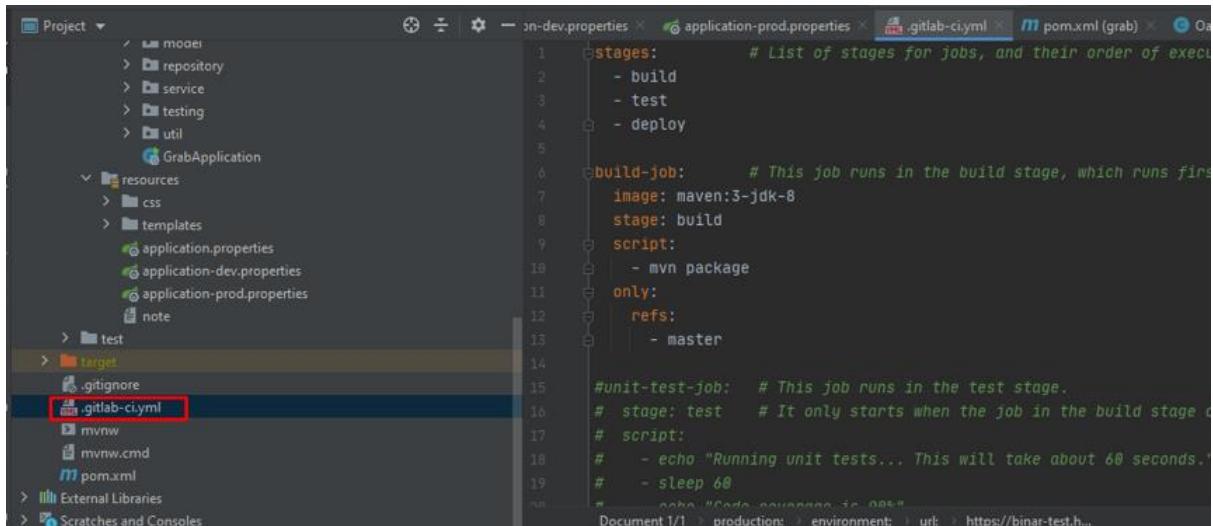
```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

g. Automatic deploy heroku

Step 1- Perlu di allow enable



Step 2- create file pada gambar dibawah ini



Step 3- isi file

```
stages:          # List of stages for jobs, and their order of execution
- build
- test
- deploy

build-job:      # This job runs in the build stage, which runs first.
```

```

image: maven:3-jdk-8
stage: build
script:
  - mvn package
only:
  refs:
    - master

#unit-test-job:  # This job runs in the test stage.
# stage: test  # It only starts when the job in the build stage
completes successfully.
# script:
#   - echo "Running unit tests... This will take about 60 seconds."
#   - sleep 60
#   - echo "Code coverage is 90%"

#lint-test-job:  # This job also runs in the test stage.
# stage: test  # It can run at the same time as unit-test-job (in
parallel).
# script:
#   - echo "Linting code... This will take about 10 seconds."
#   - sleep 10
#   - echo "No lint issues found."

deploy-job:      # This job runs in the deploy stage.
stage: deploy  # It only runs when *both* jobs in the test stage
complete successfully.
script:
  - echo "Deploying application..."
  - echo "Application successfully deployed."

#stages:
#  - build
#  - deploy

#build:
#  image: maven:3-jdk-8
#  stage: "Build"
#  script:
#    - mvn package
#  # when: manual
#  artifacts:
#    name: "$CI_JOB_STAGE-$CI_COMMIT_REF_NAME"
#    paths:
#      - target/*.jar
#  only:
#    refs:
#      - master

production:
  image: ruby:2.4
  stage: deploy
  before_script:
    - gem install dpl
    - wget -qO- https://cli-assets.heroku.com/install-ubuntu.sh | sh
  script:
    - dpl --provider=heroku --app=test --api-key=f4b7b3e5-10
    - export HEROKU_API_KEY=f4b7b3e5-2034-469f-
    - heroku run --app ecomerced2r migrate

```

```
environment:  
  name: production  
  url: https://binar-test.herokuapp.com
```

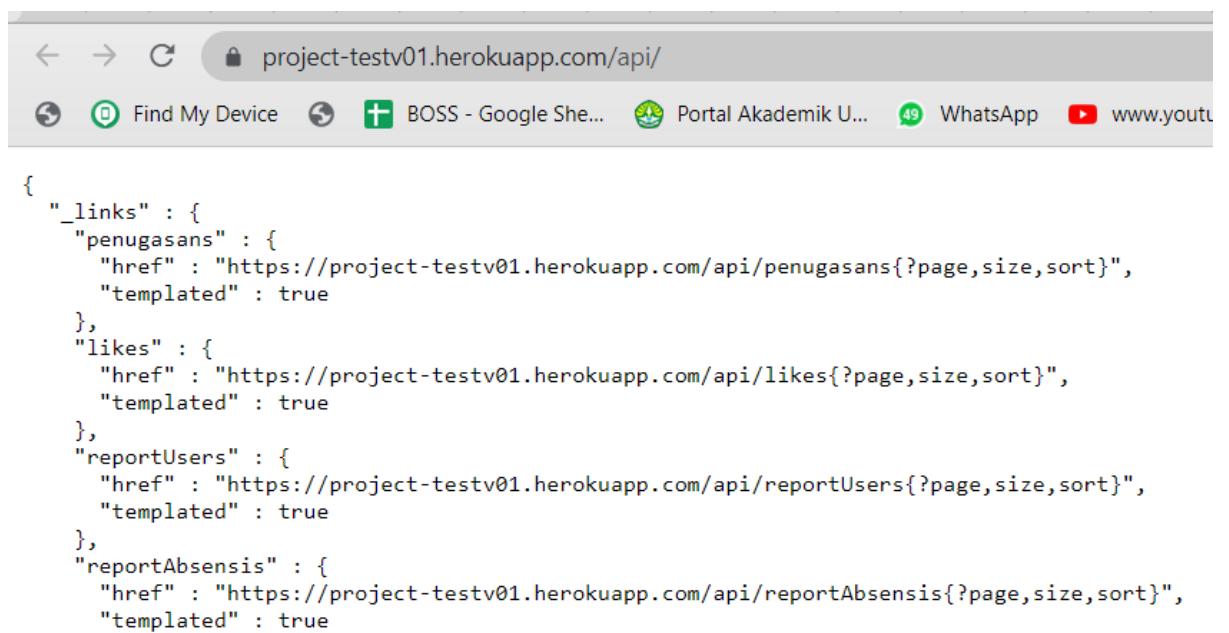
Cara mendapatkan api-key

Pilih accountsetting

The screenshot shows the Heroku dashboard under the 'Manage Account' section. On the right, a sidebar menu is open, with 'Account settings' highlighted and a red box drawn around it. In the main content area, there's a 'Profile' section with fields for 'Email Address' (rikialdipari@gmail.com) and 'Name (Optional)' (rik aldi pari). Below that is an 'SSH Keys' section with a message: 'There are no SSH keys yet. You can register new SSH keys to enable command line access to your apps.' A 'Setup Multi-Factor Authentication' button is visible above this section. At the bottom, there's an 'API Key' section with a red box drawn around the 'Reveal' button next to a password input field. A 'Regenerate API Key...' button is also present. A 'Close Account' link with a warning message is at the very bottom.

h. Akses Domain

<https://project-testv01.herokuapp.com/api>



A screenshot of a web browser window. The address bar shows the URL: project-testv01.herokuapp.com/api/. Below the address bar, there are several browser tabs and icons. The main content area displays a JSON object with various links to other API endpoints.

```
{  
  "_links" : {  
    "penugasans" : {  
      "href" : "https://project-testv01.herokuapp.com/api/penugasans{?page,size,sort}",  
      "templated" : true  
    },  
    "likes" : {  
      "href" : "https://project-testv01.herokuapp.com/api/likes{?page,size,sort}",  
      "templated" : true  
    },  
    "reportUsers" : {  
      "href" : "https://project-testv01.herokuapp.com/api/reportUsers{?page,size,sort}",  
      "templated" : true  
    },  
    "reportAbsensis" : {  
      "href" : "https://project-testv01.herokuapp.com/api/reportAbsensis{?page,size,sort}",  
      "templated" : true  
    }  
  }  
}
```

14. Cron Job Scheduler

a. Apa itu Scheduler

....

Perlu Memahami cron job command.

b. Job command



nil.pro.np

c. Generate Command Cron

<https://crontab.cronhub.io/>

The screenshot shows a table titled "Kalau yang ini adalah beberapa character special yang ada di Cron Expression~". The table has two columns: "Tanda" (Character) and "Arti" (Meaning). The rows are:

Tanda	Arti
*	Mengambil semua nilai
,	Menandakan nilai yang terpisah
-	Sebuah rentang
/	Sebuah increment

The cron expression is made of five fields. Each field can have the following values.

*	*	*	*	*
minute (0-59)	hour (0 - 23)	day of the month (1 - 31)	month (1 - 12)	day of the week (0 - 6)

Here are some examples for you.

Cron expression	Schedule
*****	Every minute
0 *****	Every hour
0 0 ***	Every day at 12:00 AM
0 0 ** FRI	At 12:00 AM, only on Friday
0 0 1 **	At 12:00 AM, on day 1 of the month

Contoh case menjalankan setiap detik ke 5 setiap menitnya

The screenshot shows a Java IDE interface with a code editor displaying a configuration file named `cronobit.properties`. The relevant line of code is:

```
cron.expression=0 * * * *
```

A red box highlights this line. In the background, a browser window displays a digital clock application titled "Jam Online" showing the time as 19.50.14. Another red box highlights the seconds part of the time.

Contoh menjalankan setiap 5 detik sekali dan berulang (/)

The screenshot shows a Java IDE interface with a code editor displaying a configuration file named `cronobit.properties`. The relevant line of code is:

```
cron.expression=5/5 * * * *
```

A red box highlights this line. In the background, a browser window displays a digital clock application titled "Jam Online" showing the time as 19.52.35. Another red box highlights the seconds part of the time.

Contoh menjalankan rentang 20 sd 40 second setiap detiknya

```
20-40 * * * *
```

The screenshot shows a Java application running in IntelliJ IDEA. The code in `ThreadRunnableSleep` contains a loop that prints the current time every two weeks. The output in the terminal shows the time as `Wed Dec 13 19:55:20 WIB 2023` repeated multiple times. A red box highlights this output.

In the browser window, a digital clock displays the time as **19.55.54**. Below the clock, there are two buttons: **Jam Digital** and **Jam Analog**.

Apa itu Jam Online?

Saat Anda ingin tahu, "Jam berapa sekarang?" Anda bisa menggunakan melihat waktu saat ini dalam jam, menit, dan detik.

Untuk melihat waktu di berbagai negara dan zona waktu di seluruh dunia.

Menjalankan di setiap detik ke 20 dan 30 (,

The screenshot shows a developer's environment with multiple windows open:

- IntelliJ IDEA:** The project navigation bar shows "batch6" and "src/main/resources/application-sit.properties". The code editor displays "ThreadConfiguration.java" with annotations like "# Other properties" and "spring.mail.properties.mail.smtp.auth". A red box highlights the cron expression "18,30 * * * *".
- Browser:** A sharing session is active, indicated by a green banner at the top right of the browser window.
- Java Application:** A window titled "19.58.15" displays the current time as 19:58:15. It includes a digital clock and an analog clock.
- Text Editor:** A file named "jamOnline.txt" is open, containing the text "Apa itu Jam Online?". Below it, a paragraph explains the functionality of the digital clock application.
- System Tray:** The Windows taskbar is visible at the bottom, showing icons for File Explorer, Task View, and other system applications.

d. Studi case

Soal:

Aplikasi Training Karyawan

1. **Send Email untuk Reminder Tugas Training:**

Anda bekerja pada aplikasi manajemen karyawan yang menyediakan fungsi untuk mengelola data karyawan dan pelatihan. Implementasikan cronjob yang mengirimkan email sebagai reminder kepada karyawan untuk mengikuti tugas training pada hari Senin setiap minggu. Pastikan email berisi informasi tentang pelatihan yang akan diadakan.

2. **Backup Database:**

Aplikasi manajemen karyawan Anda menggunakan database untuk menyimpan data karyawan. Implementasikan cronjob yang melakukan backup database setiap malam pukul 03:00. File backup harus disimpan di direktori tertentu, dan gunakan nama file yang mencakup tanggal (misalnya, backup_karyawan_2023-01-01.sql).

3. **Update Status Karyawan:**

Karyawan di aplikasi memiliki status yang perlu diperbarui secara otomatis berdasarkan informasi kontrak atau kriteria lainnya. Implementasikan cronjob yang menjalankan pengecekan dan pembaruan status karyawan pada pukul 12:00 setiap bulan pertama. Jika kontrak karyawan akan berakhir dalam 30 hari, statusnya harus diperbarui menjadi "Akan Berakhir".

Latihan

Store Dabatase : nyimpan job expression itu di database PostreSQL

<https://stackabuse.com/guide-to-quartz-with-spring-boot-job-scheduling-and-automation/>

tujuan lebih dinamic, kita bisa ubah, kapanpun ekpresion job yang kita setting.

Branch : cronjob

https://github.com/rikialdi/binar_grab/

a. Step 1- Application Properties

```
#cronjob
http://www.cronmaker.com/;jsessionid=node01eiipoe8dueep1oj9ktr546dq21
36686.node0?0
#https://www.freeformatter.com/cron-expression-generator-quartz.html
#jalankan setiap detik
cron.expression=* * * ? * *

#cron.expression=*/2 * * * * per 2 menit sekali
```

b. Config

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.task.TaskExecutor;
import
org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;

@Configuration
public class ThreadConfiguration {
    //      https://rizkimufrizal.github.io/belajar-spring-async/
    /*
     https://stackoverflow.com/questions/17659510/core-pool-size-vs-
maximum-pool-size-in-threadpoolexecutor
     https://www.baeldung.com/java-threadpooltaskexecutor-core-vs-max-
poolsize
    */
    @Bean
    public TaskExecutor taskExecutor() {
        ThreadPoolTaskExecutor executor = new
ThreadPoolTaskExecutor();
        executor.setCorePoolSize(5); // Jumlah minimum thread yang akan
tetap ada dalam pool
        executor.setMaxPoolSize(10); // Jumlah maksimum thread dalam pool
        executor.setQueueCapacity(250); // Kapasitas antrian untuk tugas
yang menunggu diproses
        executor.setThreadNamePrefix("custom_task_executor_thread"); // Nama
thread
        executor.initialize();
        return executor;
    }
    /*
     Code diatas berfungsi untuk mendefinisikan bean
     asyncTaskExecutor,
     dimana penulis membuat sebuah thread pool dengan prefix
     default_task_executor_thread,
     sehingga nanti nya kita dapat menggunakan prefix tersebut untuk
     memanggil
```

thread pool nya, contoh penggunaan nya yaitu default_task_executor_thread-test atau default_task_executor_thread-sample.

Aturan untuk ukuran ThreadPoolExecutor'skolam umumnya salah dipahami, karena tidak bekerja seperti yang Anda pikirkan seharusnya atau dengan cara yang Anda inginkan.

Ambil contoh ini. Ukuran kumpulan utas awal adalah 1, ukuran kumpulan inti adalah 5, ukuran kumpulan maks adalah 10 dan antrian adalah 100.

Sun's way: saat request masuk thread akan dibuat hingga 5, kemudian tugas akan ditambahkan ke antrian hingga mencapai 100. Ketika antrian penuh akan dibuat thread baru hingga maxPoolSize. Setelah semua utas digunakan dan antrean penuh, tugas akan ditolak. Saat antrian berkurang, begitu juga jumlah utas aktif.

Cara yang diantisipasi pengguna: saat permintaan masuk dalam utas akan dibuat hingga 10, maka tugas akan ditambahkan ke antrean hingga mencapai 100 pada titik mana mereka ditolak. Jumlah utas akan diganti namanya secara maksimal hingga antrian kosong. Ketika antrian kosong, utas akan mati sampai ada yang corePoolSizetersisa.

Perbedaannya adalah pengguna ingin mulai meningkatkan ukuran kolam lebih awal dan ingin antrian menjadi lebih kecil, sedangkan metode Sun ingin menjaga ukuran kolam tetap kecil dan hanya menambahnya setelah beban menjadi banyak.

Berikut adalah aturan Sun untuk pembuatan utas secara sederhana:

Jika jumlah utas kurang dari corePoolSize, buat utas baru untuk menjalankan tugas baru.

Jika jumlah utas sama (atau lebih besar dari) corePoolSize, masukkan tugas ke dalam antrian.

Jika antrian penuh, dan jumlah utas kurang dari maxPoolSize, buat utas baru untuk menjalankan tugas.

Jika antrian penuh, dan jumlah utas lebih besar dari atau sama dengan maxPoolSize, tolak tugas. Panjang dan pendeknya adalah bahwa utas baru hanya dibuat ketika antrian terisi, jadi jika Anda menggunakan antrian tidak terbatas maka jumlah utas tidak akan melebihi corePoolSize.

Untuk penjelasan lebih lengkap, dapatkan dari mulut kuda: ThreadPoolExecutor dokumentasi API.

Ada posting forum yang sangat bagus yang memberi tahu Anda cara ThreadPoolExecutor kerjanya dengan contoh kode:
<http://forums.sun.com/thread.jspa?threadID=5401400&tstart=0>

Info lebih lanjut:

<http://forums.sun.com/thread.jspa?threadID=5224557&tstart=450>
*/
}

c. Main Application

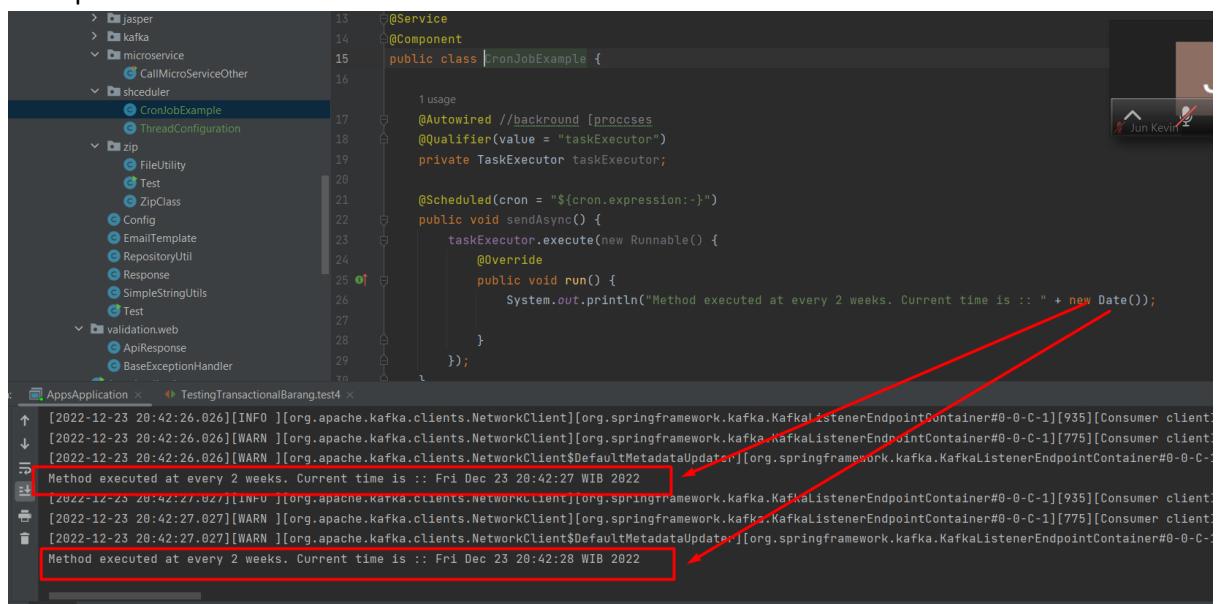
```
import org.springframework.scheduling.annotation.EnableScheduling;  
  
 @SpringBootApplication  
 @EnableConfigurationProperties({  
     FileStorageProperties.class  
 })  
 // @OpenAPIDefinition  
 // @EnableSwagger2  
 @EnableScheduling  
 public class GrabApplication {  
  
     public static void main(String[] args) { SpringApplication.run(GrabApplication.class, args); }  
 }
```

d. Cronjob Example

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.core.task.TaskExecutor;  
import org.springframework.scheduling.annotation.Scheduled;  
import org.springframework.stereotype.Component;  
import org.springframework.stereotype.Service;  
  
import java.util.Date;  
  
@Service  
@Component  
public class CronJobExample {  
  
    @Autowired // background [processes]  
    @Qualifier(value = "taskExecutor")  
    private TaskExecutor taskExecutor;  
  
    @Scheduled(cron = "${cron.expression:-}")  
    public void sendAsync() {  
        taskExecutor.execute(new Runnable() {  
            @Override  
            public void run() {  
                System.out.println("Method executed at every 2 weeks.  
Current time is :: " + new Date());  
  
            }  
        });  
    }  
}
```

e. Output

Akan melakukan akseksi per detik karena di application.properties di set per setiap detik eksekusi task.



```
13  @Service
14  @Component
15  public class CronJobExample {
16
17      @Autowired //background processes
18      @Qualifier(value = "taskExecutor")
19      private TaskExecutor taskExecutor;
20
21      @Scheduled(cron = "${cron.expression:-}")
22      public void sendAsync() {
23          taskExecutor.execute(new Runnable() {
24              @Override
25              public void run() {
26                  System.out.println("Method executed at every 2 weeks. Current time is :: " + new Date());
27              }
28          });
29      }
30  }
```

```
[2022-12-23 20:42:26.026][INFO ][org.apache.kafka.clients.NetworkClient][org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1][935][Consumer client]
[2022-12-23 20:42:26.026][WARN ][org.apache.kafka.clients.NetworkClient][org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1][775][Consumer client]
[2022-12-23 20:42:26.026][WARN ][org.apache.kafka.clients.NetworkClient$DefaultMetadataUpdater][org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]
Method executed at every 2 weeks. Current time is :: Fri Dec 23 20:42:27 WIB 2022
[2022-12-23 20:42:27.027][INFO ][org.apache.kafka.clients.NetworkClient][org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1][935][Consumer client]
[2022-12-23 20:42:27.027][WARN ][org.apache.kafka.clients.NetworkClient][org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1][775][Consumer client]
[2022-12-23 20:42:27.027][WARN ][org.apache.kafka.clients.NetworkClient$DefaultMetadataUpdater][org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]
Method executed at every 2 weeks. Current time is :: Fri Dec 23 20:42:28 WIB 2022
```

f. Cronjob Generator

<https://www.izoom.com/cron-job/>

Situs yang dimaksud kayak gini, nih.

- <http://www.cronmaker.com/>
- <https://www.freeformatter.com/cron-expression-generator-quartz.html>

g. Branch

https://github.com/rikialdi/binar_batch_5/pull/new/231222-scheduler

15. CronJob Database

<https://stackabuse.com/guide-to-quartz-with-spring-boot-job-scheduling-and-automation/>

<https://github.com/StackAbuse/spring-boot-quartz>

h. refer to doc

spring membuat scheduling.docx

E:\binar\Binat Batch 6\MODUL\ spring membuat scheduling.docx

16. Download image dari gambar kemudian simpan ke local

```
URL url = new URL("https://binar-test.herokuapp.com/api/showFile/2432022025921Captureass.PNG"); //("http://google.com/pathtoimage.jpg");
BufferedImage img = ImageIO.read(url);
File file = new File("./cdn/downloaded.jpg");
ImageIO.write(img, "jpg", file);
```

17. Docker Spring Boot / Example

<https://dev.to/tienbku/docker-compose-spring-boot-and-postgres-example-4l82>

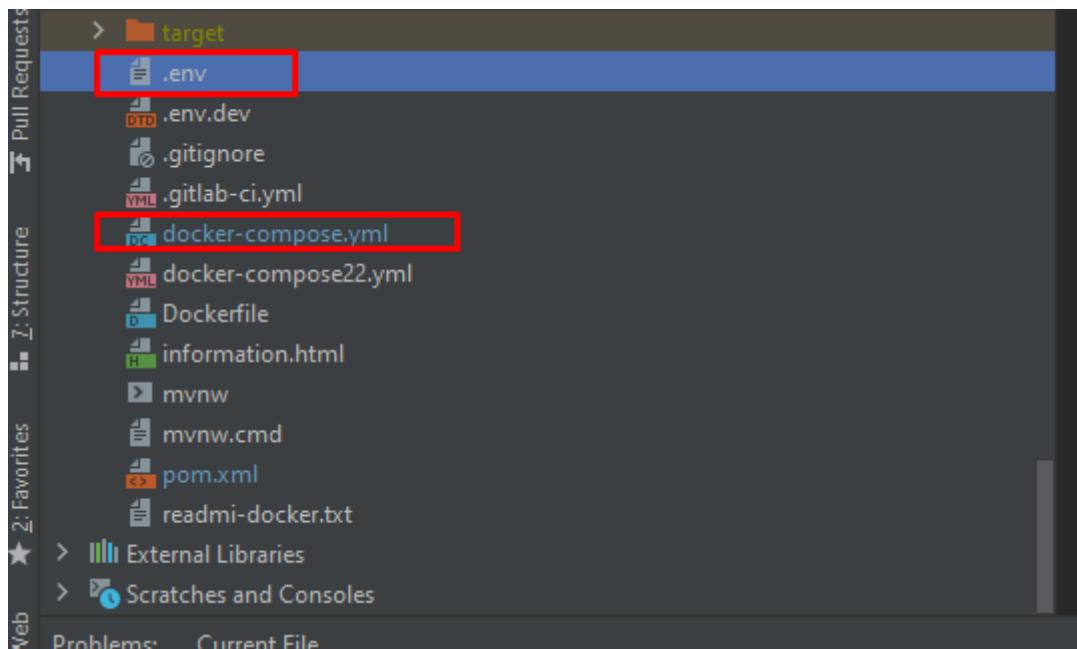
18. Docker spring boot

Branch: docker

Github : https://github.com/rikialdi/binar_grab/

a. Step 1

Tambahkan file berikut



b. Step 2 : .env

```
APP_ENV=dev
DB_USER=postgres
DB_PASS=postgres
DB_NAME=postgres
DB_HOST=postgres
DB_PORT=5432
REDIS_HOST=redis
UID=1000
GROUPS=1000
CDN_PATH=/opt/app/cache/cdn/
CDN_BASE_URL=http://localhost:8082/cdn/
CRON_EXPRESSION=0 0 */14 * * ?
CRON_EXPRESSION_USER=0 1 * * * ?
```

c. Step 3 : docker-compose.yml

```
version: '3.6'
networks:
  grab: {}

services:
#  nginx:
#    image: nginx
#    ports:
#      - 8090:80
#    volumes:
#      - ./nginx/nginx-njs.conf:/etc/nginx/nginx.conf:ro
#      - ./nginx/config/cors.conf:/etc/nginx/config/cors.conf:ro
```

```

#      - ./nginx/route.conf:/etc/nginx/routes/bca.conf:ro
#      - ./nginx/config/nginx-
njs.js:/etc/nginx/javascript/nginx.js:ro
#      - ./nginx/site-with-
oauth2.conf:/etc/nginx/conf.d/default.conf:ro
#      - ./nginx/www:/var/www/html:ro
#      - ./cache/cdn:${CDN_PATH:-/opt/app/cache/cdn}:ro
# networks:
#      - binar-postgres

postgres:
  image: postgres:9.5
  volumes:
    - ./cache/postgres:/var/lib/postgresql/data
#user: ${UID:-1000}:${GROUPS:-1000}
  environment:
    - POSTGRES_USER=${DB_USER:-postgres}
    - POSTGRES_PASSWORD=${DB_PASS:-postgres}
    - POSTGRES_DB=${DB_NAME:-postgres}
    - PGDATA=/var/lib/postgresql/data/data
    - TZ=Asia/Jakarta
  ports:
    - 5436:5432
  networks:
    grab:
      aliases:
        - ${DB_HOST:-postgres}

# redis:
#   image: redis
#   volumes:
#     - ./cache/redis:/data
##   user: ${uid:-1000}:${groups:-1000}
#   networks:
#     grab:
#       aliases:
#         - ${redis_host:-redis}

spring:
  image: maven:3-jdk-8
  ports:
    - 8090:8082
#  restart: unless-stopped
  working_dir: /opt/app
  volumes:
    - .:/opt/app
    - ./cache/maven:/var/maven/.m2
    - ./cache/cdn:${CDN_PATH:-/opt/app/cache/cdn}
#user: ${UID:-1000}:${GROUPS:-1000}
  environment:
    - TZ=Asia/Jakarta
    - SPRING_DATASOURCE_URL=jdbc:postgresql://ec2-18-210-191-
5.compute-1.amazonaws.com:5432/dvch7rlkgrl4i
    - SPRING_DATASOURCE_USERNAME=ydtyecrssfbazz
    -
SPRING_DATASOURCE_PASSWORD=efe7049a7247bef02e99208410c2f2ad038e2a29e1
b94c295d2af0aaca7d8a40
    - SPRING_PROFILES_CDN_PATHACTIVE=${APP_ENV:-dev}
#      - SPRING_CACHE_TYPE=redis

```

```

#      - SPRING_REDIS_HOST=${REDIS_HOST:-redis}
#      - SPRING_REDIS_PORT=${REDIS_PORT:-6379}
#      - SPRING_CACHE_REDIS_TIMETOLIVE=1000
- SECURITY_JWT_ENABLED=true
- SERVER_SERVLET_CONTEXT_PATH=/api
- MAVEN_CONFIG=/var/maven/.m2
- FILE_UPLOADDIR=${CDN_PATH:-/opt/app/cache/cdn/}
- FILE_SHOW_URL=${CDN_BASE_URL:-http://172.20.2.153:8090/cdn/}
- APP_UPLOADTO_CDN=${CDN_PATH:-/opt/app/cache/cdn/}
  - "MAVEN_OPTS=-Xmx2048m -XX:MaxPermSize=500m"
#
#      - CRON_EXPRESSION=${CRON_EXPRESSION:-0 0 1 */14 * ?}
#      - CRON_EXPRESSION_USER=${CRON_EXPRESSION_USER:-0 1 * * * ?}

networks:
  grab:
    aliases:
      - oauth2.api
      - hr.api
  command: [ "mvn", "-Duser.home=/var/maven", "spring-boot:run" ]
#command: [ "mvn", "-Duser.home=/var/maven", "spring-boot:run" ]

```

d. Step 4: run and noted

- docker-compose up -d : menjalankan semua yang semuar servise
- docker-compose up spring : jalankan image spring saja
- docker-compose images / docker-compose ps : show all image yg aktif
- docker-compose down -v : mematikan semua service yang ada
- docker-compose ps : melihat image yang sedang aktif
- remote images :
<https://stackoverflow.com/questions/37971961/docker-error-bind-address-already-in-use>

pass: 12345

- docker-compose logs : melihat logs
- docker-compose logs -f postgres : lebih detail , melihat logs khusu container service spring
- jalankan postgresdi docker secara manual
<https://towardsdatascience.com/local-development-set-up-of-postgresql-with-docker-c022632f13ea>
- docker-compose run -d \--name dev-postgres \e POSTGRES_PASSWORD=Pass2020! \v \${HOME}/postgres-

```
data/:/var/lib/postgresql/data \-p 5432:5432 binar
```

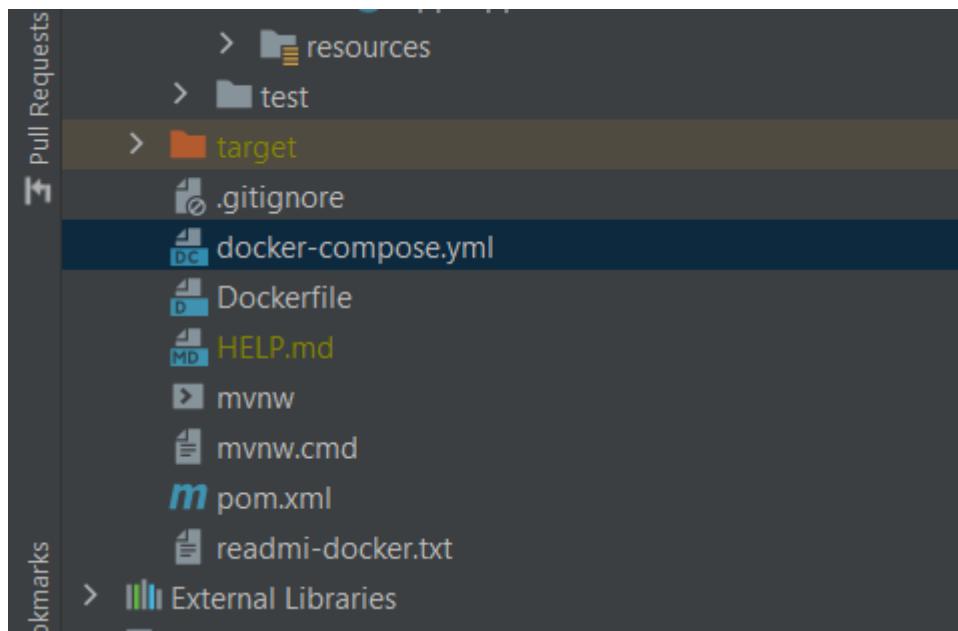
19. Docker-compose with heroku-Database : failed

a. Step-1

Docker-compose.yml

```
version: '3.6'
networks:
  grab: {}

services:
  spring:
    image: maven:3-jdk-8
    ports:
      - 8090:8082
    # restart: unless-stopped
    working_dir: /opt/app
    volumes:
      - ./opt/app
      # - ./cache/maven:/var/maven/.m2
      # - ./cache/cdn:${CDN_PATH}:-/opt/app/cache/cdn}
      #user: ${UID:-1000}:${GROUPS:-1000}
    environment:
      - TZ=Asia/Jakarta
      - SPRING_DATASOURCE_URL=jdbc:postgresql://ec2-18-210-191-5.compute-1.amazonaws.com:5432/dvch7rlkgrl4i
      - SPRING_DATASOURCE_USERNAME=ydtyecrssfbazz
      -
      SPRING_DATASOURCE_PASSWORD=efe7049a7247bef02e99208410c2f2ad038e2a29e1b94c295d2af0aac7d8a40
      - SPRING_PROFILES_CDN_PATHACTIVE=${APP_ENV:-dev}
      - SECURITY_JWT_ENABLED=true
      - SERVER_SERVLET_CONTEXT_PATH=/api
      - MAVEN_CONFIG=/var/maven/.m2
      - FILE_UPLOADDIR=${CDN_PATH}:-/opt/app/cache/cdn/
      - FILE_SHOW_URL=${CDN_BASE_URL}:-http://172.20.2.153:8090/cdn/
      - APP_UPLOADTO_CDN=${CDN_PATH}:-/opt/app/cache/cdn/
    networks:
      grab:
        aliases:
          - oauth2.api
    command: [ "mvn", "-Duser.home=/var/maven", "spring-boot:run" ]
```



b. Step-2 : register railway.app untuk database

<https://railway.app/>

c. Note:

Tidak support jasperreport: sehingga dependency di matikan dulu

d. Docker-run

Docker-compose up -d

e. Tessting output

```
2022-11-22 14:41:23 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.11.1/plexus-compiler-javac-2.11.1.pom  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.11.1/plexus-compiler-javac-2.11.1.pom  
2022-11-22 14:41:24 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.11.1/plexus-compiler-api-2.11.1.pom  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.11.1/plexus-compiler-api-2.11.1.pom  
2022-11-22 14:41:24 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utilss/3.3.4/maven-shared-utilss-3.3.4.jar  
2022-11-22 14:41:24 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-incremental/1.1/maven-shared-incremental-1.1.jar  
2022-11-22 14:41:24 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac-2.11.1.jar  
2022-11-22 14:41:24 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/ow2/asem/asm/9.2/asm-9.2.jar  
2022-11-22 14:41:24 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox-2.0.1.jar  
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-incremental/1.1/maven-shared-utilss-3.3.4/maven-shared-utilss-3.3.4.jar (14 kB at 26 kB/s)  
2022-11-22 14:41:25 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.11.1/plexus-compiler-api-2.11.1.jar  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac-1.1.1.jar (548 kB/s)  
2022-11-22 14:41:25 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac-1.1.1/plexus-compiler-javac-1.1.1.jar  
2022-11-22 14:41:25 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac-1.1.1/plexus-compiler-javac-1.1.1.jar  
2022-11-22 14:41:25 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/ow2/asem/asm/9.2/asm-9.2.jar  
122 kB at 187 kB/s  
2022-11-22 14:41:25 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.11.1/plexus-compiler-manager-2.11.1.jar  
2022-11-22 14:41:25 apps-spring-1 | Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utilss/3.3.4/maven-shared-utilss-3.3.4.jar (153 kB at 230 kB/s)  
2022-11-22 14:41:25 apps-spring-1 | Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac-2.11.1/plexus-compiler-javac-2.11.1.jar  
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0.1/qdox-2.0.1.jar (334 kB at 427 kB/s)  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.11.1/plexus-compiler-api-2.11.1.jar (27 kB at 20 kB/s)  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.11.1/plexus-compiler-javac-2.11.1.jar (20 kB at 20 kB/s)  
2022-11-22 14:41:26 apps-spring-1 | Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.11.1/plexus-compiler-manager-2.11.1.jar (4.7 kB at 4.0 kB/s)  
2022-11-22 14:41:26 apps-spring-1 | Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac-2.11.1/plexus-compiler-javac-2.11.1.jar (264 kB at 203 kB/s)  
2022-11-22 14:41:33 apps-spring-1 | [INFO] Changes detected - recompiling the module!  
2022-11-22 14:41:36 apps-spring-1 | [INFO] Compiling 113 source files to /opt/app/target/classes
```

f. Branch :

https://github.com/rikialdi/binar_batch_5/tree/docker-compose-success

20.Docker – Kafka

a. Project

E:\binar\Binat Batch 6\MODUL\Docker\docker

b. Running docker

Docker-compose up -d

c. Running java project

d. Output

Docker-compose logs -f

```


Dockerfile



```

FROM openjdk:11-jdk-alpine
EXPOSE 8080
COPY --from=nodejs /usr/local/lib/node_modules/.bin/* /usr/local/bin/
COPY --from=python3 /usr/local/lib/python3.8/site-packages/* /usr/local/lib/python3.8/

```



KafkaConfig.java



```

private String bootstrapServer;
@Value(value = "${spring.kafka.bootstrap-servers}")
private String bootstrapServers;

@Bean
public KafkaAdmin kafkaAdmin() {
 Map<String, Object> configs = new HashMap<>();
 configs.put(AdminClientConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServer);
 return new KafkaAdmin(configs);
}

@Bean
public NewTopic topic_demo() { return new NewTopic("topic_demo", 1, (short) 1); }

```


```

e. Postman test kafka with docker

Postman

POST localhost:8080/public/post-message

Body (JSON)

```

1 {
2   "message": "test123"
3 }

```

Response

Status: 200 OK Time: 129 ms Size: 493 B

```

1 {
2   "message": "Message Sent Successfully",
3   "status": "Success"
4 }

```

```

10 import org.springframework.web.bind.annotation.RestController;
11
12 import java.util.HashMap;
13 import java.util.Map;
14
15 @RestController
16 @RequestMapping("/public")
17 public class KafkaController {
18
19     2 usages
20     @Autowired
21     MessageProcess messageProcess;
22
23     @PostMapping("/post-message")
24     public ResponseEntity postMessage(@RequestBody Map<String, Object> body) {
25         Map<String, Object> response = new HashMap<>();
26         try {
27             if(body.containsKey("message")) {
28                 String result = messageProcess.process(body);
29                 response.put("result", result);
30             }
31         } catch (Exception e) {
32             response.put("error", e.getMessage());
33         }
34         return ResponseEntity.ok(response);
35     }
36 }

```

Run: App

```

2023-12-18 13:33:15.760 INFO 23780 --- [nio-8080-exec-3] o.a.k.clients.producer.KafkaProducer : [Producer clientId=producer-1] Instantiated an idempotent producer.
2023-12-18 13:33:15.778 INFO 23780 --- [nio-8080-exec-3] o.a.kafka.common.utils.AppInfoParser : Kafka version: 3.1.2
2023-12-18 13:33:15.778 INFO 23780 --- [nio-8080-exec-3] o.a.kafka.common.utils.AppInfoParser : Kafka commitId: f8c67dc3ae0a3265
2023-12-18 13:33:15.778 INFO 23780 --- [nio-8080-exec-3] o.a.kafka.common.utils.AppInfoParser : Kafka startTimeMs: 1702881195778
2023-12-18 13:33:15.791 INFO 23780 --- [ad | producer-1] org.apache.kafka.clients.Metadata : [Producer clientId=producer-1] Resetting the last seen epoch of partition t
2023-12-18 13:33:15.791 INFO 23780 --- [ad | producer-1] org.apache.kafka.clients.Metadata : [Producer clientId=producer-1] Cluster ID: f0Cu5V4cSvaZ14W7Te2Rbg
test123
2023-12-18 13:33:15.918 INFO 23780 --- [ad | producer-1] o.a.k.c.p.internals.TransactionManager : [Producer clientId=producer-1] ProducerId set to 0 with epoch 0
message yang didapat dari kafka : test123

```

21. Docker- Maven-PostgreSQL : success

a. Studi Case

Kita memiliki 2 service

1. Aplikasi spring boot/maven
2. Postgresql

Existing : base on localhost

Ekpektasi : akan di jalankan di container -> docker

b. Install docker

1. Docker
2. Docker compose

<https://docs.docker.com/desktop/install/windows-install/>

c. command docker compose

1. build
docker file : "docker build ."
docker compose : " docker-compose build ."

- noted : ketika ada perubahan image, disarankan untuk dilakukan build image
2. running : semua service yang ada di dalam docker
docker-compose up -d
running service base on name
docker-compose up **servicename**
 3. **melihat log secara keseluruhan**
docker-compose logs
melihat logs yang aktif- realtime
docker-compose logs -f
melihat logs di salah satu service :
docker-compose logs -f serviceName
 4. **mematikan service secara keseluruhan**
docker-compose down

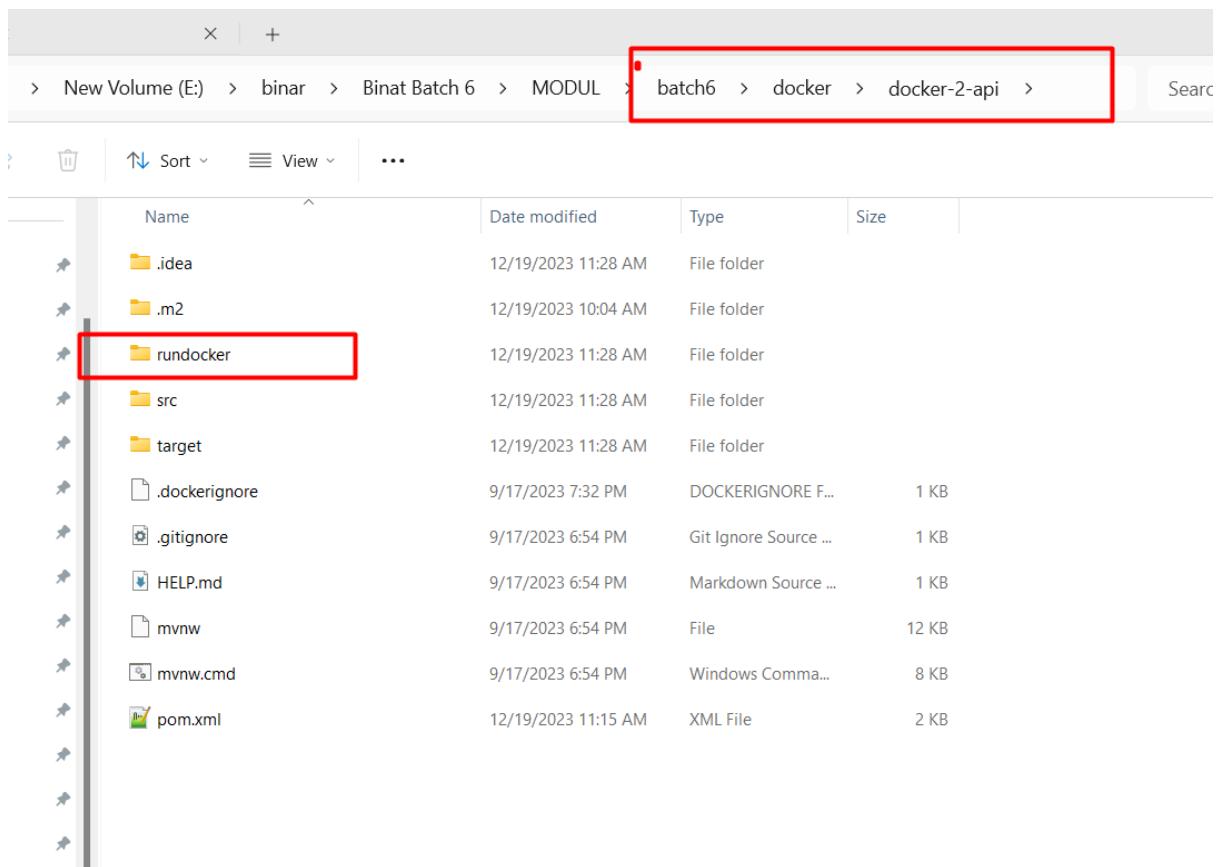
d. url referensi

<https://www.bezkoder.com/docker-compose-spring-boot-mysql/>

e. project

E:\binar\Binat Batch 6\MODUL\Docker\docker-2-api\rundockerBranch :
<https://gitlab.com/rikialdi/batch6> branch docker

Ada di folder



f. code 1 dockerfile

pastikan jdk yang diinstall sama jdk yang dipake di aplikasi jar kalian

```
FROM openjdk:8-jdk-alpine

ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} demo-0.0.1-SNAPSHOT.jar
ENTRYPOINT ["java","-jar","demo-0.0.1-SNAPSHOT.jar"]
```

Noted : nama jar harus sesuaikan dengan nama jar project

g. code 2 docker-compose.yml

noted : terdiri dari 2 service/image

1. postgresdb
2. app

menghubungkan postgresdb ke app dengan "depends_on"

```
version: "3.8"
```

```

services:
  postgresdb:
    image: bitnami/postgresql:latest #mysql:5.7
    restart: unless-stopped
    env_file: ./env
    environment:
      - POSTGRES_PASSWORD=$POSTGRESDB_ROOT_PASSWORD
      - POSTGRES_USER=$POSTGRESDB_USER
      - POSTGRES_DB=$POSTGRESDB_DATABASE
    ports:
      - $POSTGRESQLDB_LOCAL_PORT:$POSTGRESDB_DOCKER_PORT
    volumes:
      - db2:/var/lib/postgresql
  app:
    depends_on:
      - postgresdb
    build: ./ #akan build dockerfile
    container_name: employee1-api
    restart: on-failure
    env_file: ./env
    ports:
      - $SPRING_LOCAL_PORT:$SPRING_DOCKER_PORT
    environment:
      SPRING_APPLICATION_JSON: '{
        "spring.datasource.url"  :
"jdbc:postgresql://postgresdb:$POSTGRESDB_DOCKER_PORT/$POSTGRESDB_DATABASE",
        "spring.datasource.username" : "$POSTGRESDB_USER",
        "spring.datasource.password" : "$POSTGRESDB_ROOT_PASSWORD",
        "spring.jpa.properties.hibernate.dialect" :
"org.hibernate.dialect.PostgreSQLDialect",
        "spring.jpa.hibernate.ddl-auto" : "update"
      }'
    volumes:
      - .m2:/root/.m2
    stdin_open: true
    tty: true

volumes:
  db2:

```

Add container name

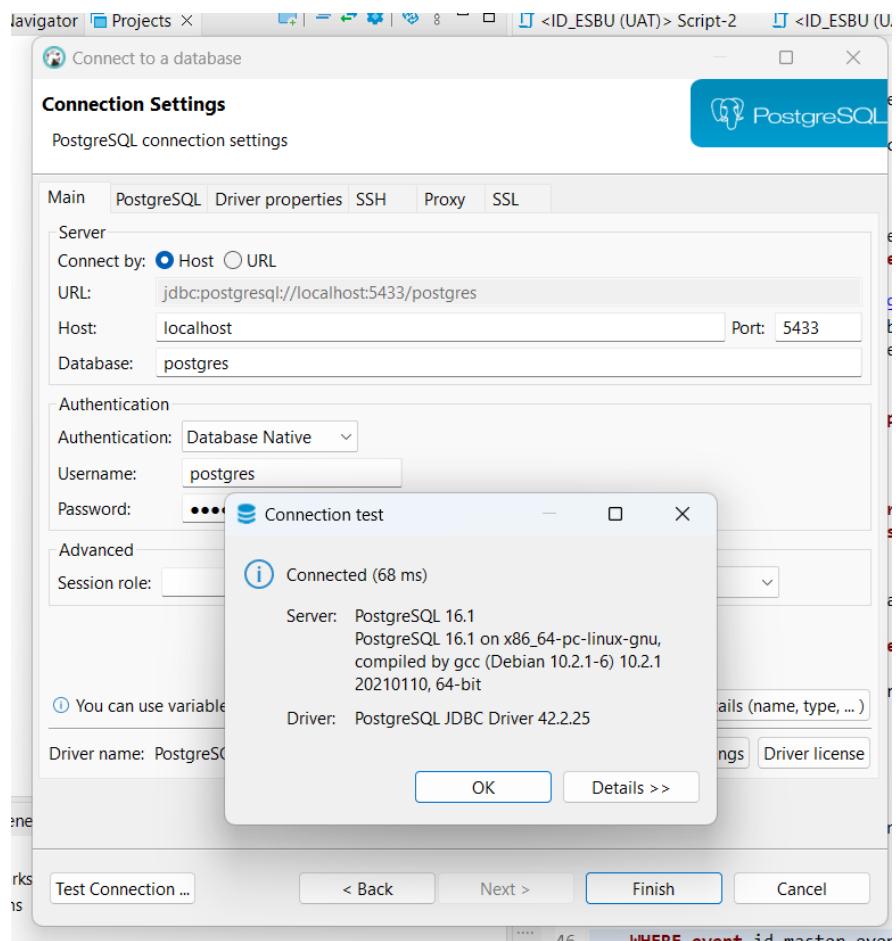
h. code 3.env

```
POSTGRESDB_USER=postgres  
POSTGRESDB_ROOT_PASSWORD=password  
POSTGRESDB_DATABASE=postgres  
POSTGRESQLDB_LOCAL_PORT=5433  
POSTGRESDB_DOCKER_PORT=5432  
  
SPRING_LOCAL_PORT=8081  
SPRING_DOCKER_PORT=8080
```

Noted:

1. POSTGRESDB_DOCKER_PORT : harus port 5432
 2. POSTGRESQLDB_LOCAL_PORT : port yang di assign ke public, misal untuk koneksi ke dbeaver

i. konksi todbeaver

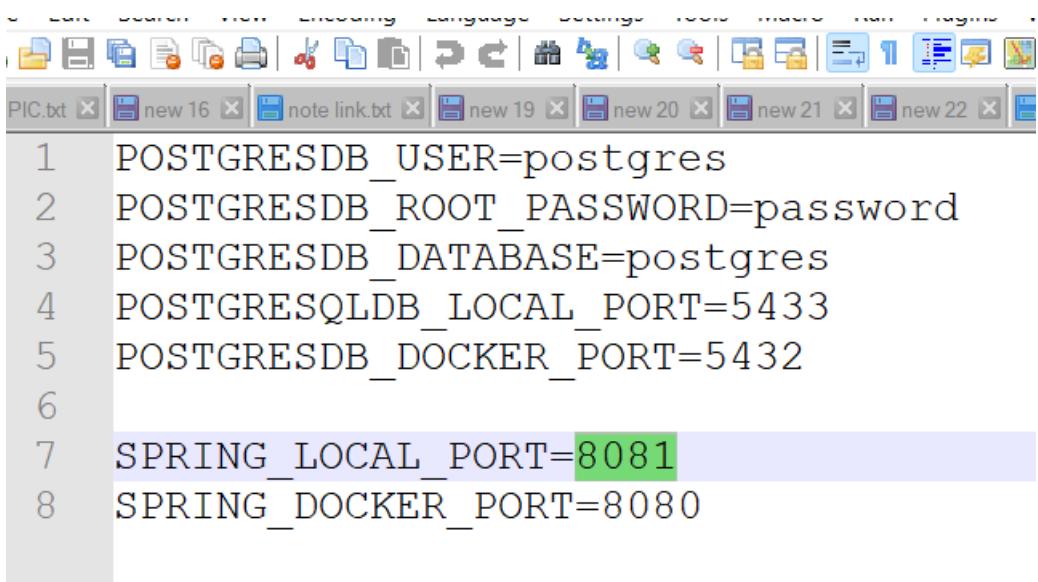


j. noted penting

db harus 5432,
port running
jika ada perubahan image, lakukan
docker-compose build . (pake titik)
atau "docker build ."

```
employee1-api exited with code 1
canceled

E:\binar\Binat Batch 6\MODUL\Docker\a>docker build .
[+] Building 18.8s (3/4)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 188B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:17-jdk-slim
=> [auth] library/openjdk pull token for registry-1.docker.io
0.0s
0.0s
0.0s
0.0s
18.7s
0.0s
```



```

1 POSTGRESDB_USER=postgres
2 POSTGRESDB_ROOT_PASSWORD=password
3 POSTGRESDB_DATABASE=postgres
4 POSTGRESQLDB_LOCAL_PORT=5433
5 POSTGRESDB_DOCKER_PORT=5432
6
7 SPRING_LOCAL_PORT=8081
8 SPRING_DOCKER_PORT=8080

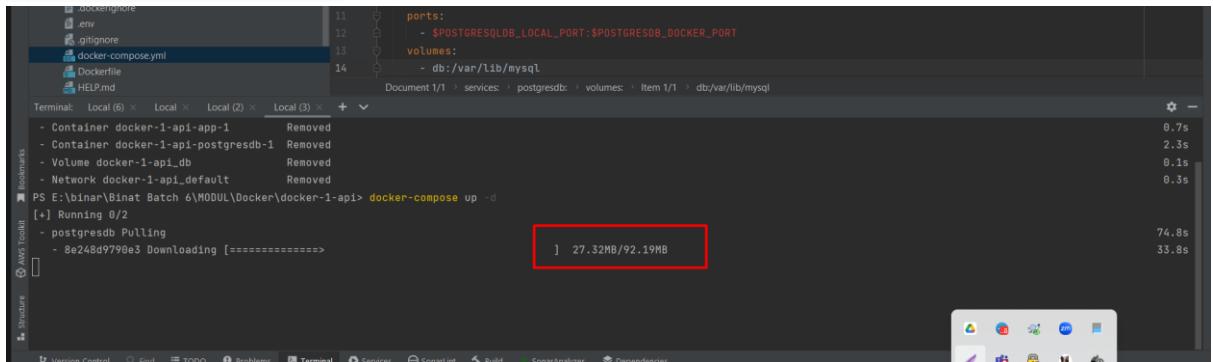
```

k. running

`docker-compose up -build`

or

`docker-compose up -d`

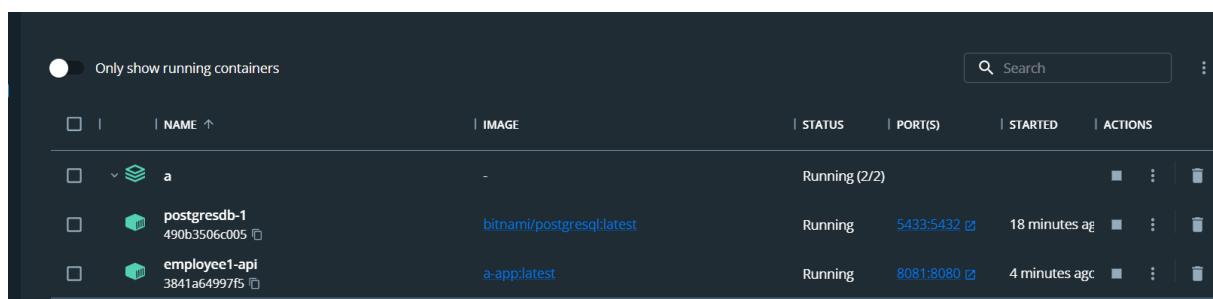


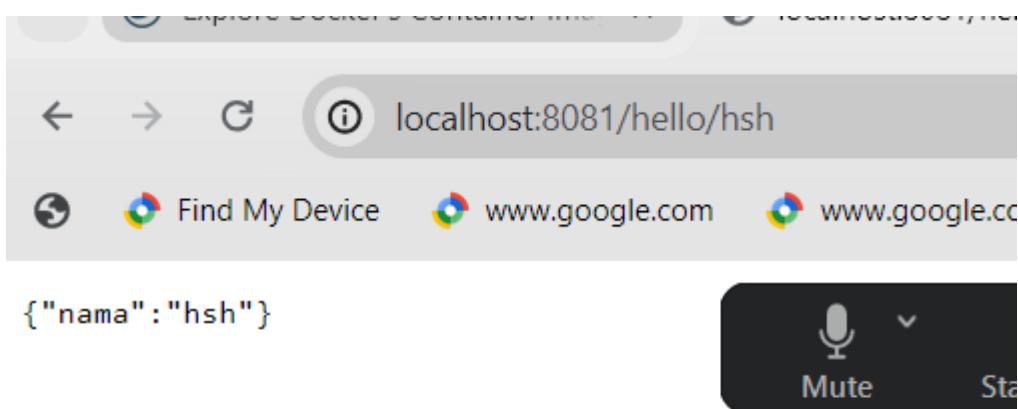
```

11 ports:
12   - $POSTGRESQLDB_LOCAL_PORT:$POSTGRESDB_DOCKER_PORT
13 volumes:
14   - db:/var/lib/mysql
Document 1/1 > services > postgresdb > volumes > Item 1/1 > db/var/lib/mysql

Terminal Local (0) x Local x Local (2) x Local (3) x + v
- Container docker-1-api-app-1      Removed          0.7s
- Container docker-1-api-postgresdb-1 Removed          2.3s
- Volume docker-1-api_db           Removed          0.1s
- Network docker-1-api_default     Removed          0.3s
PS E:\binar\Binat Batch 6\MODUL\ Docker\docker-1-api> docker-compose up -d
[+] Running 0/2
- postgresdb Pulling
- 8e248d9790e3 Downloading [=====] 27.32MB/92.19MB
74.8s
33.8s

```





```
C:\Windows\System32\cmd.exe + -> employee1-api | 2023-12-19 04:18:27.721 INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
employee1-api | 2023-12-19 04:18:27.721 INFO 1 --- [           main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9
.employee1-api | 2023-12-19 04:18:27.781 INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicati
onContext
employee1-api | 2023-12-19 04:18:27.841 INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializatio
n completed in 1284 ms
employee1-api | 2023-12-19 04:18:28.013 INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo
[<name: default>
employee1-api | 2023-12-19 04:18:28.136 INFO 1 --- [           main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.6
.Final
employee1-api | 2023-12-19 04:18:28.151 INFO 1 --- [           main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotation
s {5.1.2.Final}
employee1-api | 2023-12-19 04:18:28.395 INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
employee1-api | 2023-12-19 04:18:28.603 INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
employee1-api | 2023-12-19 04:18:28.626 INFO 1 --- [           main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.d
ialect.PostgreSQLDialect
employee1-api | Hibernate: create table employee (id bigserial not null, address TEXT, name varchar(255), status varchar(255), primary key (id))
employee1-api | 2023-12-19 04:18:29.277 INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementati
on: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
employee1-api | 2023-12-19 04:18:29.285 INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for
persistence unit 'default'
employee1-api | 2023-12-19 04:18:29.562 WARN 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by def
ault. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
employee1-api | 2023-12-19 04:18:29.939 INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) wi
th context path ''
employee1-api | 2023-12-19 04:18:29.948 INFO 1 --- [           main] com.example.demo.DemoApplication : Started DemoApplication in 3.909 seconds
(JVM running for 4.384)
|
```

The screenshot shows the pgAdmin 4 interface. On the left, there's a tree view of databases, schemas, and tables. Under the 'postgres' database, the 'public' schema is expanded, showing the 'employee' table. The 'employee' table has 16K rows. Other options like 'Columns', 'Constraints', and 'Indexes' are also listed.

I. Test postman

The screenshot shows the Postman application interface. A POST request is being made to `localhost:8081/save`. The request body is set to `JSON` and contains the following JSON data:

```

1 {
2   "name": "Novian",
3   "address": "jakarta",
4   "dob": "2023-01-01",
5   "status": "active"
6 }

```

The response status is `200 OK`, time `355 ms`, size `239 B`.

22. Latihan Docker + postgresql + maven + kafka

1. coba buat service di docker : postgresql
koneksikan kedbeaver
2. buatkan service untuk kafka, dari project kalian sebelumnya.
dan di running

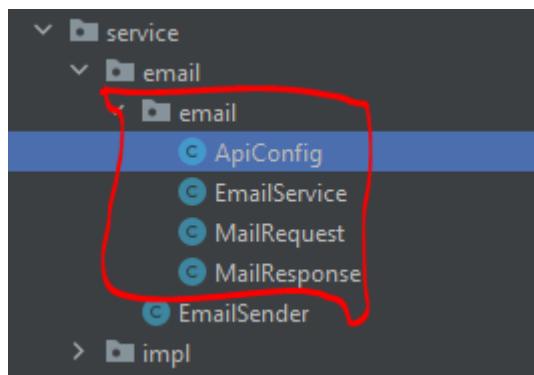
23. Send File To Email

Branch: docker

Github : <https://github.com/rikialdi/dibimbing/>

Branch : spring-security-01-registerotp

a. Step 1 – Struktur



b. Step 2 :code

ApiConfig

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.ui.freemarker.FreeMarkerConfigurationFactoryBean;

@Configuration
public class ApiConfig {

    @Primary
    @Bean
    public FreeMarkerConfigurationFactoryBean factoryBean() {
        FreeMarkerConfigurationFactoryBean bean=new
        FreeMarkerConfigurationFactoryBean();
        bean.setTemplateLoaderPath("classpath:/templates");
        return bean;
    }

}
```

EmailService

```
package com.binar.grab.service.email.email;
```

```

import com.binar.grab.config.Config;
import freemarker.template.Configuration;
import freemarker.template.Template;
import freemarker.template.TemplateException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.ClassPathResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;
import org.springframework.ui.freemarker.FreeMarkerTemplateUtils;

import javax.activation.DataSource;
import javax.imageio.ImageIO;
import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.URL;
import java.nio.channels.Channels;
import java.nio.channels.ReadableByteChannel;
import java.nio.charset.StandardCharsets;
import java.util.Date;
import java.util.Map;

@Service
public class EmailService {

    @Autowired
    private JavaMailSender sender;
    //    @Autowired
    //    private JavaMailSenderImpl sender;

    Config config2 = new Config();

    @Autowired
    private Configuration config;

    @Value("${spring.mail.username}")//FILE_SHOW_RUL
    private String emailPengirim;

    public MailResponse sendEmail(MailRequest request, Map<String,
Object> model) {
        MailResponse response = new MailResponse();
        MimeMessage message = sender.createMimeMessage();
        try {
            // set mediaType
            MimeMessageHelper helper = new MimeMessageHelper(message,
MimeMessageHelper.MULTIPART_MODE_MIXED_RELATED,
                StandardCharsets.UTF_8.name());
            // add attachment :naro logo di di../resources
            helper.addAttachment("logo.png", new
ClassPathResource("logo.png"));

            // 13. Download Gimage dari gambar kemudian simpan ke
local
            // URL url = new URL("https://binar-

```

```

test.herokuapp.com/api/showFile/2432022025921Captureass.PNG"); //("http
p://google.com/pathtoimage.jpg");
//
//      BufferedImage img = ImageIO.read(url);
//      File file = new File("./cdn/downloaded.jpg");
//      ImageIO.write(img, "jpg", file);
//      helper.addAttachment("image1", file);
String fileName= "./cdn/Captureass.PNG";
File file2 = new File(fileName);
String formatFile =
fileName.substring(fileName.lastIndexOf("."));
helper.addAttachment(config2.convertDateToString(new
Date())+"image4"+formatFile, file2);

Template t = config.getTemplate("email-template.ftl");
String html =
FreeMarkerTemplateUtils.processTemplateIntoString(t, model);

helper.setTo(request.getTo());
//      helper.setTo(emailPengirim);
helper.setText(html, true);
helper.setSubject(request.getSubject());
helper.setFrom(request.getFrom());
sender.send(message);

response.setMessage("mail send to : " + request.getTo());
response.setStatus(Boolean.TRUE);

} catch (MessagingException | IOException | TemplateException
e) {
    response.setMessage("Mail Sending failure : " +
e.getMessage());
    response.setStatus(Boolean.FALSE);
}

return response;
}

}

```

MailRequest

```

package com.binar.grab.service.email.email;

import lombok.Data;

@Data
public class MailRequest {

    private String name;
    private String to;
    private String from;
    private String subject;

}

```

MailResponse

```
package com.binar.grab.service.email.email;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class MailResponse {
    private String message;
    private boolean status;
}
```

c. Step 3 : pom.xml and application.properties

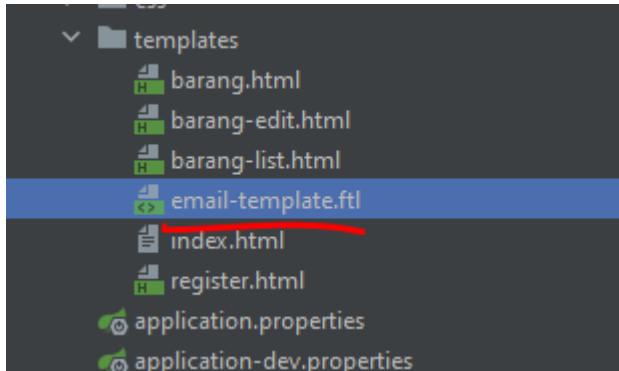
Pom.xml

```
<dependency>
    <groupId>org.freemarker</groupId>
    <artifactId>freemarker</artifactId>
    <version>2.3.31</version>
</dependency>
```

Application.properties

```
BASEURL=http://localhost:8082/api/
SHOWFILEPATH=http://localhost:8082/api/showFile/
```

d. Step 4: resources



Email-template

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Java Techie Mail</title>
</head>

<body style="font-family: 'Space Grotesk'; font-size: 12px">
<table width="100%" border="0" cellspacing="0" cellpadding="0" >
    <tr >
        <td align="center" valign="top" bgcolor="#FFFFFF" style="background-color: #edecea; "><br> <br>
            <table width="50%" border="0" cellspacing="0" cellpadding="0" >
                <tr >
                    <td valign="top" bgcolor="#FFFFFF" align="center" style="background-color: #FFFFFF; font-size: 12px; color: #000000; padding: 20px;">
                        <div style="">
                            
                        </div>
                    </td>
                </tr>
                <tr >
                    <td valign="top" bgcolor="#FFFFFF" style="background-color: #FFFFFF; font-size: 12px; color: #000000; padding: 0px 50px 0px 50px;">
                        <div style="font-size: 12px; color: #000000;">
                            <b>Dear, ${namesaya}</b> <br>
                            <br> Your order confirmation from:
                            <table><tr>
                                <td width="20"></td>
                                <td><b>Jl. Sudirman</b></td>
                            </tr></table><br>
                        </div>
                </tr>
            </table>
        </td>
    </tr>
</table>
<br>
```

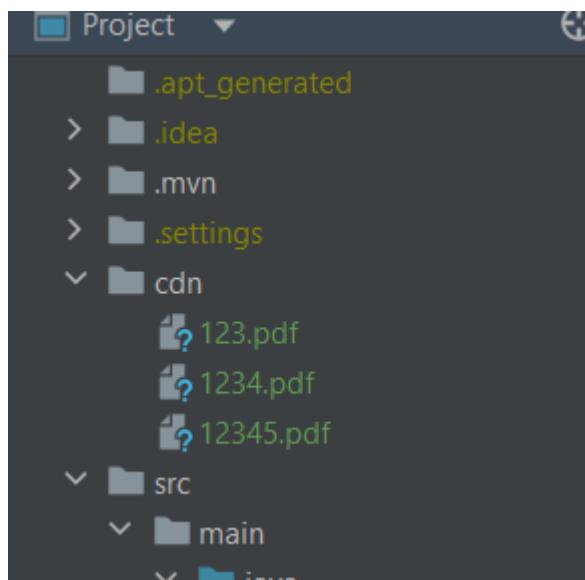
```

        </div>
        <table style="border-bottom:2px dashed " >
            <tr >
                <th style="border-bottom:2px dashed
">Qty</th>
                <th style="border-bottom:2px dashed
">Desc</th>
                <th style="border-bottom:2px dashed
">Amt</th>
            </tr>
            <#list datainvoce as item>
                <tr>
                    <td>${item.nama}</td>
                    <td>${item.satuan}</td>
                    <td>${item.harga}</td>
                </tr>
                <tr>
                    <td colspan="1">1</td>
                    <td colspan="1">Shipping</td>
                    <td colspan="1">10</td>
                </tr>
                <br>
            </#list>
            <tr>
                <td colspan="2" style="text-align:
right;">SUBTOTAL</td>
                <td colspan="1" >10000</td>
            </tr>

        </table>
        <div style="font-size: 12px; color:
#000000;">
            <br> <b>Will be shipped to:</b><br>
            Jl Sudirman <br>
            Jakarta <br>
            20876 <br>
        </div>
        <div style="font-size: 12px; color:
#000000;">
            <br> <li>You ill get notification on our
            app <b>@UserTes</b> when ship
            your items.</li><br>
        </div>
        </td>
    </tr>
</table>
<br> <br></td>
</tr>
</table>
</body>
</html>

```

e. Lokasi Direktori file



f. REST API

```
@Autowired
private EmailService service;
@Autowired
public BarangRepository barangRepository;

SimpleStringUtils simpleStringUtils = new SimpleStringUtils();

@PostMapping("/sendemailfile")
public ResponseEntity<String> sendemailfile(HttpServletRequest
request3) {
    MailRequest request = new MailRequest();
    request.setName("Invoice Test name");
    request.setTo("rikialdipari@gmail.com");
    request.setFrom("rikialdipari@gmail.com");
    request.setSubject("Invoice Test");
    Map model = new HashMap<>();
    model.put("namesaya", "riki aldi pari");

    model.put("chuteicon", "
https://tinypng.com/images/social/website.jpg ");
    BigDecimal total= new BigDecimal(0);

    Pageable show_data = simpleStringUtils.getShort("id", "desc", 0,
1);
    Page<Barang> data = barangRepository.getAllData(show_data);

    model.put("datainvoice", data.getContent());
    model.put("iconuser",
https://tinypng.com/images/social/website.jpg );

    List<String> dataFILE = new ArrayList<>();
```

```

        dataFILE.add(0, "123.pdf");
        dataFILE.add(1, "1234.pdf");
        dataFILE.add(2, "12345.pdf");
        model.put("dataFile", dataFILE);
        MailResponse ma = service.sendEmailWithFile(request, model);
        System.out.println("MailResponse 1="+ma.getMessage());
        return ResponseEntity.ok().body("Successfully");
    }
}

```

g. Step 5 : Testing By Junit

```

@Value("${SHOWFILEPATH}")//FILE_SHOW_RUL
private String SHOWFILEPATH;

@Autowired
public BarangRepository barangRepository;

SimpleStringUtils simpleStringUtils = new SimpleStringUtils();

@Autowired
private EmailService service;

@Test
public void sendEmail() {
    MailRequest request = new MailRequest();
    request.setName("Invoice Test name");
    request.setTo("rikialdipari@gmail.com");
    request.setFrom("rikialdipari@gmail.com");
    request.setSubject("Invoice Test");
    Map<String, Object> model = new HashMap<>();
    model.put("namesaya", "riki aldi pari");

    model.put("chuteicon", "https://binar-
test.herokuapp.com/api/showFile/2432022025921Captureass.PNG");
    BigDecimal total= new BigDecimal(0);

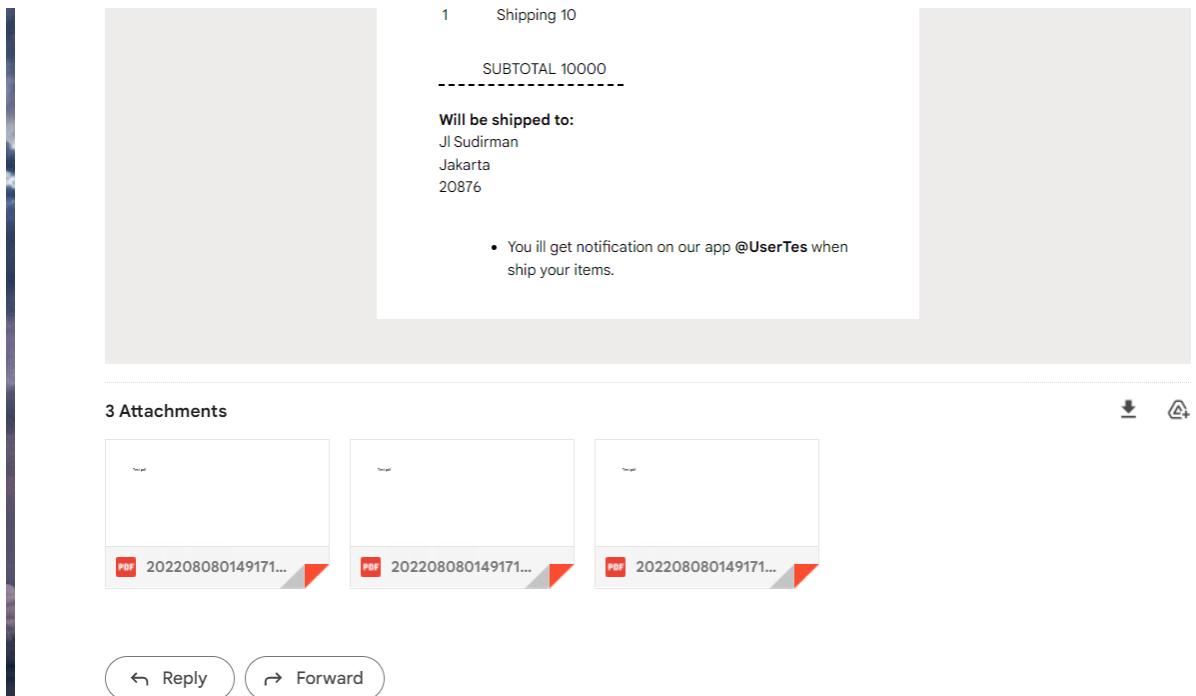
    Pageable show_data = simpleStringUtils.getShort("id", "desc", 0,
1);
    Page<Barang> data = barangRepository.getAllData(show_data);

    model.put("datainvoice", data.getContent());
    model.put("iconuser","https://binar-
test.herokuapp.com/api/showFile/2432022025921Captureass.PNG");
    MailResponse ma = service.sendEmail(request, model);
    System.out.println("MailResponse 1="+ma.getMessage());

}
}

```

h. Output



i. Send Multi File

Email service

```
public MailResponse sendEmailWithFile(MailRequest request, Map<String, Object> model) {
    MailResponse response = new MailResponse();
    MimeMessage message = sender.createMimeMessage();
    try {
        // set mediaType
        MimeMessageHelper helper = new MimeMessageHelper(message,
                MimeMessageHelper.MULTIPART_MODE_MIXED_RELATED,
                StandardCharsets.UTF_8.name());
        // add attachment :naro logo di di.../resources
        helper.addAttachment("logo.png", new ClassPathResource("logo.png"));

        // 13. Download Gimage dari gambar kemudian simpan ke local
        URL url = new URL("https://binar-test.herokuapp.com/api/showFile/2432022025921Captureass.PNG"); //("http://google.com/pathtoimage.jpg");
        BufferedImage img = ImageIO.read(url);
        File file = new File("./cdn/downloaded.jpg");
        ImageIO.write(img, "jpg", file);
        helper.addAttachment("image1", file);

        // String fileName= "./cdn/sss.pdf";
        // File file2 = new File(fileName);
        // String formatFile =
    }
}
```

```

        fileName.substring(fileName.lastIndexOf(".")) ;
        // helper.addAttachment(config2.convertDateToString(new Date())+"image4"+formatFile, file2);

        List<String> dataFILE = (List<String>) model.get("dataFILE");
        for(String ab : dataFILE){
            String fileName= "./cdn/"+ab;
            File file2 = new File(fileName);
            String formatFile =
            fileName.substring(fileName.lastIndexOf(".")) ;
            System.out.println("fileName="+fileName);
            helper.addAttachment(config2.convertDateToString(new Date())+ab+formatFile, file2);
        }

        Template t = config.getTemplate("email-template.ftl");
        String html =
        FreeMarkerTemplateUtils.processTemplateIntoString(t, model);

        helper.setTo(request.getTo());
        // helper.setTo(emailPengirim);
        helper.setText(html, true);
        helper.setSubject(request.getSubject());
        helper.setFrom(request.getFrom());
        sender.send(message);

        response.setMessage("mail send to : " + request.getTo());
        response.setStatus(Boolean.TRUE);

    } catch (MessagingException | IOException | TemplateException e) {
        response.setMessage("Mail Sending failure : " +
e.getMessage());
        response.setStatus(Boolean.FALSE);
    }

    return response;
}

```

Controller

```

@PostMapping("/sendemailfile")
public ResponseEntity<String> sendemailfile(HttpServletRequest request3) {
    MailRequest request = new MailRequest();
    request.setName("Invoice Test name");
    request.setTo("rikialdipari@gmail.com");
    request.setFrom("rikialdipari@gmail.com");
    request.setSubject("Invoice Test");
    Map model = new HashMap<>();
    model.put("namesaya", "riki aldi pari");

    model.put("chuteicon", "https://binar-test.herokuapp.com/api/showFile/2432022025921Captureass.PNG");
    BigDecimal total= new BigDecimal(0);
}

```

```

        Pageable show_data = simpleStringUtils.getShort("id", "desc", 0,
1);
        Page<Barang> data = barangRepository.getAllData(show_data);

        model.put("dataInInvoice", data.getContent());
        model.put("iconUser", "https://binar-
test.herokuapp.com/api/showFile/2432022025921Captureass.PNG") ;

        List<String> dataFile = new ArrayList<>();
        dataFile.add(0, "sss.pdf");
        dataFile.add(1, "s2.pdf");
        dataFile.add(2, "s3.pdf");
        model.put("dataFile", dataFile);
        MailResponse ma = service.sendEmailWithFile(request, model);
        System.out.println("MailResponse 1=" + ma.getMessage());
        return ResponseEntity.ok().body("Access token invalidated
successfully");
    }
}

```

24. Login Costum : With Username And Password Only

a. branch

url : <https://github.com/rikialdi/dibimbing.git>

branch : spring-security-01-registerotp

https://gitlab.com/rikialdi/superproof-adam/-/tree/sesi9-login-custom?ref_type=heads

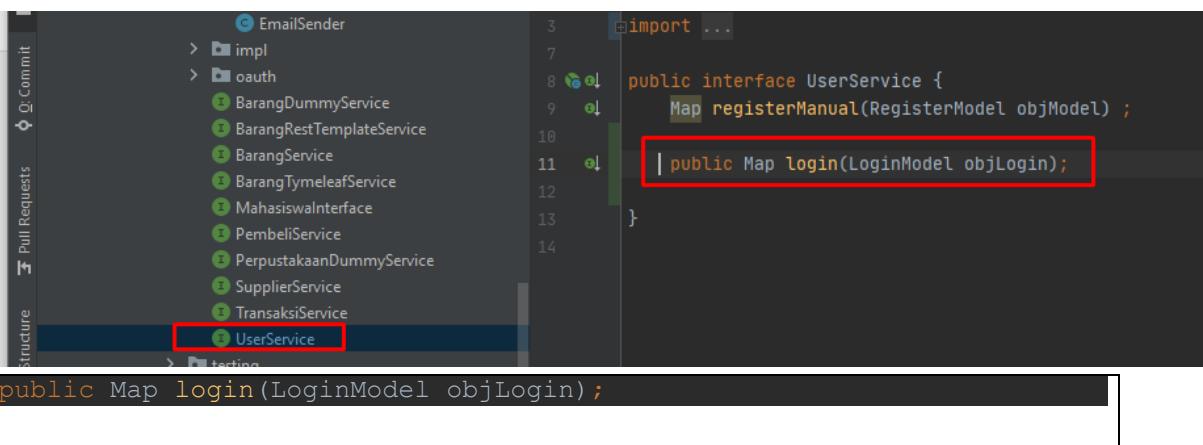
b. Step 1- pom.xml

```

<!--validation-->
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>2.0.1.Final</version>
</dependency>

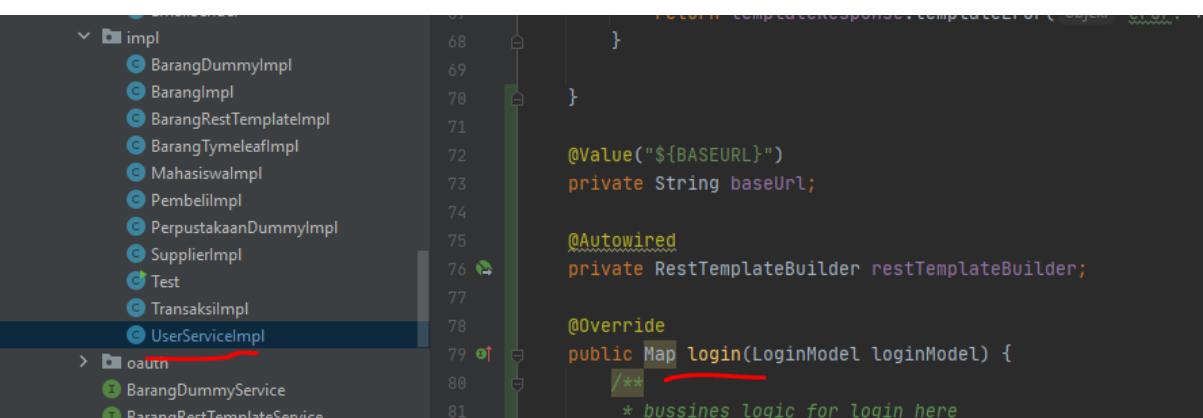
```

c. Step 2- buatlah class userservice



```
import ...  
public interface UserService {  
    Map registerManual(RegisterModel objModel);  
    public Map login(LoginModel objLogin);  
}
```

d. Step 3- user impl



```
package com.binar5.apps.service.impl;  
  
import com.binar5.apps.dto.LoginModel;  
import com.binar5.apps.entity.oauth.Role;  
import com.binar5.apps.entity.oauth.User;  
import com.binar5.apps.repository.oauth.UserRepository;  
import com.binar5.apps.service.UserService;  
import com.binar5.apps.utils.Response;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.boot.web.client.RestTemplateBuilder;  
import org.springframework.core.ParameterizedTypeReference;  
import org.springframework.http.HttpMethod;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
import org.springframework.web.client.HttpStatusCodeException;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;
```

```

import java.util.Map;
@Service
public class UserServiceImpl implements UserService {
    @Value("${BASEURL}")
    private String baseUrl;

    @Autowired
    private RestTemplateBuilder restTemplateBuilder;

    @Autowired
    private UserRepository userRepository;
    @Autowired
    private PasswordEncoder encoder;
    @Autowired
    public Response templateResponse;

    @Override
    public Map login(LoginModel loginModel) {
        /**
         * bussines logic for login here
         */
        try {
            Map<String, Object> map = new HashMap<>();
            User checkUser =
userRepository.findOneByUsername(loginModel.getUsername());

            if ((checkUser != null) &&
(encoder.matches(loginModel.getPassword(), checkUser.getPassword()))) {
                if (!checkUser.isEnabled()) {
                    map.put("is_enabled", checkUser.isEnabled());
                    return templateResponse.templateError(map);
                }
            }
            if (checkUser == null) {
                return templateResponse.notFound("user not found");
            }
            if (!encoder.matches(loginModel.getPassword(),
checkUser.getPassword())) {
                return templateResponse.templateError("wrong password");
            }
            String url = baseUrl + "/oauth/token?username=" +
loginModel.getUsername() +
                "&password=" + loginModel.getPassword() +
                "&grant_type=password" +
                "&client_id=my-client-web" +
                "&client_secret=password";
            ResponseEntity<Map> response =
restTemplateBuilder.build().exchange(url, HttpMethod.POST, null, new
ParameterizedTypeReference<Map>() {
});

            if (response.getStatusCode() == HttpStatus.OK) {
                User user =
userRepository.findOneByUsername(loginModel.getUsername());
                List<String> roles = new ArrayList<>();

                for (Role role : user.getRoles()) {
                    roles.add(role.getName());
                }
            }
        }
    }
}

```

```
//  
    }  
    //save token  
    checkUser.setAccessToken(response.getBody().get("access_token").toString())  
;  
//  
checkUser.setRefreshToken(response.getBody().get("refresh_token").toString()  
));  
//  
    userRepository.save(checkUser);  
  
        map.put("access_token",  
response.getBody().get("access_token"));  
        map.put("token_type",  
response.getBody().get("token_type"));  
        map.put("refresh_token",  
response.getBody().get("refresh_token"));  
        map.put("expires_in",  
response.getBody().get("expires_in"));  
        map.put("scope", response.getBody().get("scope"));  
        map.put("jti", response.getBody().get("jti"));  
  
        return map;  
    } else {  
        return templateResponse.notFound("user not found");  
    }  
} catch (HttpStatusCodeException e) {  
    e.printStackTrace();  
    if (e.getStatusCode() == HttpStatus.BAD_REQUEST) {  
        return templateResponse.templateError("invalid login");  
    }  
    return templateResponse.templateError(e);  
} catch (Exception e) {  
    e.printStackTrace();  
  
    return templateResponse.templateError(e);  
}  
}  
}
```

e. Application.properties

BASEURL=http://localhost:9091/api

f. Response Global

```
package com.binar5.apps.utils;

import com.sun.org.apache.xpath.internal.operations.Bool;
```

```
import org.springframework.stereotype.Component;

import java.util.HashMap;
import java.util.Collections;
import java.util.Map;

@Component
public class Response {

    public Map sukses(Object obj) {
        Map map = new HashMap();
        map.put("data", obj);
        map.put("code", 200);
        map.put("status", "sukses");
        return map;
    }

    public Map error(Object obj, Object code) {
        Map map = new HashMap();
        map.put("code", code);
        map.put("status", obj);
        return map;
    }

    public BooleanisRequired(Object obj) {
        if(obj == null) {
            return true;
        }
        return false;
    }

    public Map templateSukses(Object objek) {
        Map map = new HashMap();
        map.put("data", objek);
        map.put("message", "sukses");
        map.put("status", "200");
        return map;
    }

    public Map templateEror(Object objek) {
        Map map = new HashMap();
        map.put("message", objek);
        map.put("status", "404");
        return map;
    }

    public Map notFound(Object objek) {
        Map map = new HashMap();
        map.put("message", objek);
        map.put("status", "404");
        return map;
    }
}
```

g. Model – request

The screenshot shows a file tree on the left and the code for `LoginModel` on the right. The `LoginModel` class is highlighted with a red box.

```
2 import lombok.Data;
3
4
5 import javax.validation.constraints.NotEmpty;
6
7 @Data
8 public class LoginModel {
9     @NotEmpty(message = "username is required.")
10    private String username;
11    @NotEmpty(message = "password is required.")
12    private String password;
13 }
14
```

```
package com.binar.grab.dao.request;

import lombok.Data;

import javax.validation.constraints.NotEmpty;

@Data
public class LoginModel {
    @NotEmpty(message = "username is required.")
    private String username;
    @NotEmpty(message = "password is required.")
    private String password;
}
```

h. Controller

The screenshot shows the IntelliJ IDEA interface. On the left, the project tree displays various Java files under the 'controller' package, including LoginController.java, which is currently selected and highlighted with a red box. The code editor on the right contains the implementation of the LoginController class.

```
package com.binar.grab.controller;

import ...
import com.binar.grab.config.Config;
import com.binar.grab.dao.request.LoginModel;
import com.binar.grab.dao.request.RegisterModel;
import com.binar.grab.model.oauth.User;
import com.binar.grab.repository.oauth.UserRepository;
import com.binar.grab.service.UserService;
import com.binar.grab.service.email.EmailSender;
import com.binar.grab.util.EmailTemplate;
import com.binar.grab.util.SimpleStringUtils;
import com.binar.grab.util.TemplateResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;

import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/user-login/")
public class LoginController {
    @Autowired
    private UserRepository userRepository;
```

```
package com.binar.grab.controller;

import com.binar.grab.config.Config;
import com.binar.grab.dao.request.LoginModel;
import com.binar.grab.dao.request.RegisterModel;
import com.binar.grab.model.oauth.User;
import com.binar.grab.repository.oauth.UserRepository;
import com.binar.grab.service.UserService;
import com.binar.grab.service.email.EmailSender;
import com.binar.grab.util.EmailTemplate;
import com.binar.grab.util.SimpleStringUtils;
import com.binar.grab.util.TemplateResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;

import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import javax.validation.ConstraintViolationException;
import javax.validation.Valid;
import java.util.Map;

@RestController
@RequestMapping("/user-login/")
public class LoginController {
    @Autowired
```

```

private UserRepository userRepository;

Config config = new Config();

@Autowired
public UserService serviceReq;

@Autowired
public TemplateResponse templateCRUD;

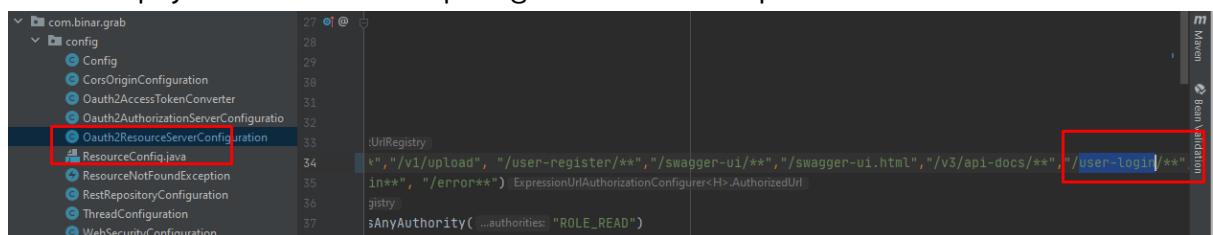
@PostMapping("/login")
@ExceptionHandler(ConstraintViolationException.class)
public ResponseEntity<Map> login(@Valid @RequestBody LoginModel
objModel) {
    Map map = serviceReq.login(objModel);
    return new ResponseEntity<Map>(map, HttpStatus.OK);
}

}

```

i. Config : allow Uri

Guna : supaya bisa diakses tanpa login terlebih :/tampa token



j. Testing with junit

```
java - com - editor - grid - jUnit - Runnings - GitHub - Recent - Help - ResponseEntity<String> response = restTemplate.postForEntity(url, body, String.class);

assertEquals(HttpStatus.OK, response.getStatusCode());
System.out.println("response = " + response.getBody());
```

```
@Test
public void restTemplateLogin2() throws Exception {
    HttpHeaders headers = new HttpHeaders();
    headers.set("Accept", "*/*");
    headers.set("Content-Type", "application/json");
    String bodyTesting = "{\n        \"username\":\"rikialdipari@gmail.com\",\n        \"password\":\"password\"\n    }";
    HttpEntity<String> entity = new HttpEntity<String>(bodyTesting, headers);
    ResponseEntity<String> exchange = restTemplate.exchange(url, HttpMethod.POST, entity, String.class);

    assertEquals(HttpStatus.OK, exchange.getStatusCode());
    System.out.println("response = " + exchange.getBody());}
```

POM.xml

```
<!-- JUNIT -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-test</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
</dependency>
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
</dependency>
```

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment =
SpringBootTest.WebEnvironment.RANDOM_PORT)
public class CallMicroServiceOther {

    @Autowired
    private TestRestTemplate restTemplate;
    @Test
    public void restTemplateLogin2() throws Exception {
        HttpHeaders headers = new HttpHeaders();
        headers.set("Accept", "*/*");
        headers.set("Content-Type", "application/json");
        String bodyTesting = "{\n            \"username\":\"rikialdipari@gmail.com\",\n            \"password\":\"password\"\n        }";
    }
}
```

```

        HttpEntity<String> entity = new HttpEntity<String>(bodyTesting,
headers);

        ResponseEntity<String> exchange =
restTemplate.exchange("http://localhost:8082/api/user-login/login",
HttpMethod.POST, entity, String.class);

        assertEquals(HttpStatus.OK, exchange.getStatusCode());
System.out.println("response =" + exchange.getBody());
}

```

Output-sukses

The screenshot shows the IntelliJ IDEA interface with the following details:

- Code Editor:** Displays a Java test class with the following code snippet:

```

        @Test
        public void restTemplateLogin2() throws Exception {
            HttpHeaders headers = new HttpHeaders();
            headers.set("Accept", "*/*");
            headers.set("Content-Type", "application/json");
            String bodyTesting = "{\n                \"username\":\"rikialdipari@gmail.com\",\n                \"password\":\"password\"\n            }";
            HttpEntity<String> entity = new HttpEntity<String>(bodyTesting, headers);
            ResponseEntity<String> exchange =
                restTemplate.exchange("http://localhost:8082/api/user-login/login",
                    HttpMethod.POST, entity, String.class);
            assertEquals(HttpStatus.OK, exchange.getStatusCode());
            System.out.println("response =" + exchange.getBody());
        }
    
```
- Run Tool Window:** Shows a single test run named 'restTemplateLogin2' with a duration of 2ms. A red box highlights this entry.
- Output Console:** Displays repeated log messages indicating the method was executed at regular intervals (every 2 weeks). The last message shows the response body containing an access token.

k. Testing with Postman

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** `{host}/login-user`
- Body:** JSON (selected)
 - Content Type: application/json
 - Body:

```
1 {
2   ...
3     "email" : "admin@mail.com",
4     "password" : "password"
5 }
```
- Response Status:** 200 OK
- Time:** 4.84 s
- Size:** 4.63 KB
- Save Response:** Available

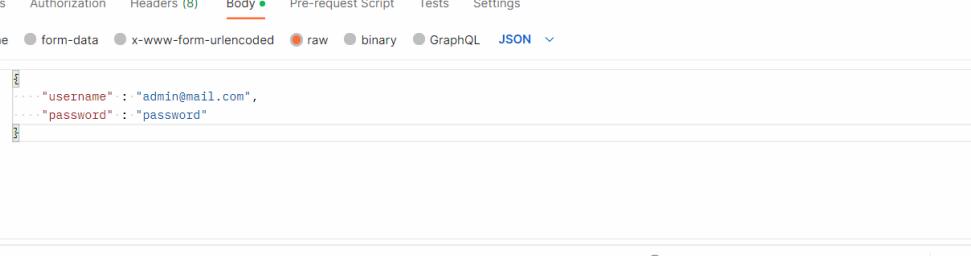
The response body is displayed as follows:

```
1 {
2   ...
3     "data": {
4       "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
5         eyJhdWQiOlsib2F1dGgyLXjlc291cmNlI0sInVzZXJfbmFtZSI6ImFkbWluQG1haWwuY29tIiwic2NvCGuiOlscimVhZCIsIndyaXRlI0sImV4cCI6MTY20TEXNDMw
6           iwiYXV0aG9yaXRpZXMi01siUK9MRV9TVBFU1VTRVI1LCJST0xF0FETU10iwiuk9MRV9VU0VsI0sImp0aSI6IjA2YWIZNDZhLTVmNDMtNDU4ZC1iMWMyLTN1ZDFjZT
7               Uw0GEoxOCIsImNsawVuwdF9pZC16Im15LNwsawUdc13ZWiif0.z9BSqKKwS44jMctfjxPtQ245z1LJv2tD_gqnwt9izW0",
8       "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
9         eyJhdWQiOlsib2F1dGgyLXjlc291cmNlI0sInVzZXJfbmFtZSI6ImFkbWluQG1haWwuY29tIiwic2NvCGuiOlscimVhZCIsIndyaXRlI0sImF0aSI6IjA2YWIZNDZh
10           TVmNDMtNDU4ZC1iMWyLTN1ZDFjZTUw0GEoxOCIsImV4cCI6MTY3NjM0MzEwMiwiYXV0aG9yaXRpZXMi01siUK9MRV9TVBFU1VTRVI1LCJST0xF0FETU10iwiuk9MRV
11               9VU0VsI0sImp0aSI6Im1U2GiyjkzLTkxyV2tmtHgOS04MzE4LWUz0dg3YjUxzDBhZiisImNsawVuwdF9pZC16Im15LNwsawUdc13ZWiifQ.
12                   2FZ06LS5V4q20J2wieJgPjJh0vcJPyfflerM6gjfphfa",
13       "scope": "read write",
14       "token_type": "bearer",
15       "expires_in": 28799,
16       "user": ...
17     },
18     "jti": "06ab346a-5f43-458d-b1c2-3ed1ce508a18"
19   },
20 }
```

25. Cara menggunakan token bearer

a. Get token saat login

Copi paste token access_token



The screenshot shows a Postman request for a login API. The request method is POST to `({host})/user-login/login`. The body is set to JSON and contains the following data:

```
1 {"username": "admin@mail.com",  
2 "password": "password"}  
3  
4
```

The response status is 200 OK, with a time of 12.56 s and a size of 1.59 KB. The response body is as follows:

```
1 {  
2     "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
3         eyJhdWQ10lsib2F1dGyLXJlc29icmlNIl0sInVzXJfbmFtZS16ImFkbWluQG1haWuY29tIiwic2NvcGU10lsicmVhZCIsIndyaXRl1l0sImV4cCI6MTY2OTE1MjAwMyiYXV0aGyaXRpZXMi0lsiuK9MRV9TVBFU1VTRVi1LCJST0xFx0FETU10IiwiUk9MRV9VU0VS10sImp0aSI6IjBlUE1LbmtxOfHwUVlmbza1WwfWfMpJsnZ4NCIsImNsawVudF9pZC16Im15LNWaNVvdC13ZWiifq.Dn81D69nNL0--HjYHanKLkmXS6-WpfdcL-pSus4qE_0",  
4     "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
5         eyJhdWQ10lsib2F1dGyLXJlc29icmlNIl0sInVzXJfbmFtZS16ImFkbWluQG1haWuY29tIiwic2NvcGU10lsicmVhZCIsIndyaXRl1l0sImF0aSI6IjBlUE1LbmtxOfHwUVlmbza1WwfWfMpJsnZ4NCIsImV4cCI6MTY3NjM4DgwMyiYXV0aGyaXRpZXMi0lsiuK9MRV9TVBFU1VTRVi1LCJST0xFx0FETU10IiwiUk9MRV9VU0VS10sImp0aSI6InJXR1ZCvjc3cvZGVhF5Qw82RU1JvzB6M1k5dyIsImNsawVudF9pZC16Im15LNWaNVvdC13ZWiifq.bAs_hCMYltGm7pUwNmxt7X4hCv7_OmnjSHubr4g_KA",  
6     "scope": "read write",  
7     "token_type": "bearer",  
8     "expires_in": 28799,  
9     "jti": "0ePMKnkq8XVQYfo05Yap2ZIJvx4"  
10}
```

b. Input token di header dan output

Tambahkan di header token yang sudah didapatkan

The screenshot shows a Postman workspace with a collection named "Binar Batch 5 / Barang Dengan Token / List Barang Costum". A GET request is made to `((host))/v1/binar/barang/list/costum`. In the Headers tab, the Authorization header is set to `Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJ...`. The response status is 200 OK with a time of 42 ms and a size of 1.51 KB. The response body is a JSON object:

```
18     "alamat": "jl sudirman2 update",
19     "nama": "Zandi",
20     "hp": "09876546254545"
21   },
22   {
23     "hargaRupiah": "Rp. 10.000"
24   },
25   {
26     "created_date": "2022-11-15T14:23:12.151+0700",
     "updated_date": "2022-11-15T14:23:12.151+0700",
   }
```

26.Refresh TOKEN

a. Step 1- lakukan login

Ambil atau copy paste value refresh_token

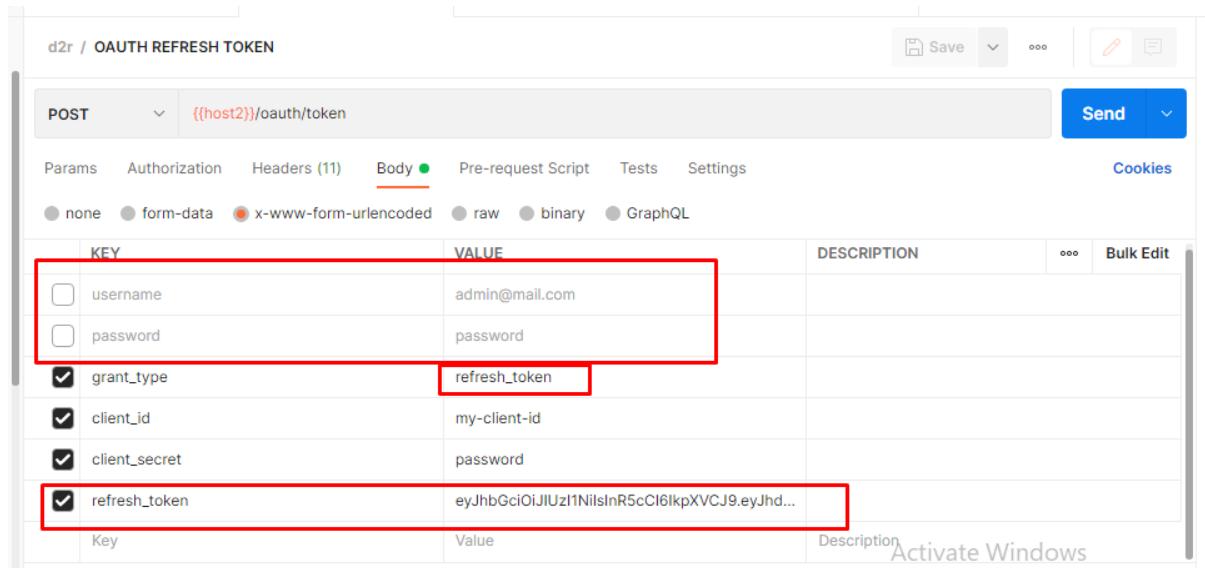
The screenshot shows a Postman workspace with a collection named "BINAR". A POST request is made to `((host2))/oauth/token` under the "POST OAUTH LOGIN" section. The Body tab shows the following parameters:

- client_id: my-client-id
- client_secret: password

The response status is 200 OK with a time of 25.79 s and a size of 1.36 KB. The response body is a JSON object:

```
1   {
2     "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsib2FidGgyLXJlc291cmNlIl0sImV4cCI6MTY3OTc1Nzg2NywidX",
3     "token_type": "bearer",
4     "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsib2FidGgyLXJlc291cmNlIl0sInVzX3fbmFtZSI6ImFkbkluQ",
5     "expires_in": "3110353",
6     "scope": "read write",
```

b. Step 2- to do refresh token



KEY	VALUE	DESCRIPTION	Bulk Edit
<input type="checkbox"/> username	admin@mail.com		
<input type="checkbox"/> password	password		
<input checked="" type="checkbox"/> grant_type	refresh_token		
<input checked="" type="checkbox"/> client_id	my-client-id		
<input checked="" type="checkbox"/> client_secret	password		
<input checked="" type="checkbox"/> refresh_token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhd...		

Param username dan password di hapus.

Value grant_type ganti dengan refresh_token.

Tambahkan param refresh_token.

27. Upload dan simpan barang satu enpoind

Branch : uploadbarang

url : <https://github.com/rikialdi/dibimbing.git>

```
@Value("${app.uploadto.cdn}") //FILE_SHOW_RUL
private String UPLOADED_FOLDER;

@PostMapping(value = "/uploadsimpanbarang/{idsupplier}", consumes =
{"multipart/form-data", "application/json"})
public ResponseEntity<Map> uploadFile(
    @RequestParam(value="file", required = true) MultipartFile
file,
    @PathVariable(value = "idsupplier") Long idsupplier,
    @RequestParam(value="nama", required = true) String nama,
    @RequestParam(value="stok", required = true) int stok,
    @RequestParam(value="satuan", required = true) String
satuan,
    @RequestParam(value="harga", required = true) double harga
) {
    Date date = new Date();
    SimpleDateFormat formatter = new
SimpleDateFormat("ddMyyyyhhmmss");
    String strDate = formatter.format(date);
    String fileName = UPLOADED_FOLDER + strDate +
file.getOriginalFilename();
    String fileNameforDownload = strDate +
```

```

        file.getOriginalFilename();
        Path TO = Paths.get(fileName);
        Map map = new HashMap();
        try {
            Files.copy(file.getInputStream(), TO); // pengolahan upload
disini :
            // insert barang
            Barang b = new Barang();
            b.setNama(nama);
            b.setStok(stok);
            b.setSatuan(satuan);
            b.setHarga(harga);
            b.setFileName(fileNameforDOwnload);
            map = barangService.insert(b, idsupplier);
        } catch (Exception e) {
            e.printStackTrace();
            return new
        ResponseEntity<Map>(templateResponse.templateEror("eror"),
HttpStatus.OK);
        }
        return new
        ResponseEntity<Map>(templateResponse.templateSukses(map),
HttpStatus.OK);
    }
}

```

a. Output

The screenshot shows a POST request in Postman to the endpoint `{{host2}}/oauth/token`. The request body is set to `x-www-form-urlencoded` and contains a single key-value pair: `refresh_token` with the value `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsib2F1dGgyLXJlc291cmNlIl0sImV4cCI6MTY3OTc1ODA3MiwidXi...`.

The response status is 200 OK with a response time of 1694 ms and a size of 1.36 KB. The response body is displayed in JSON format:

```

1   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsib2F1dGgyLXJlc291cmNlIl0sImV4cCI6MTY3OTc1ODA3MiwidXi...",
2   "token_type": "bearer",
3   "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsib2F1dGgyLXJlc291cmNlIl0sInVzZXJfbmFtZSI6ImFkbWluQ...",
4   "expires_in": 31103999,
5   "scope": "read write",
6   "jti": "78943dc0-f85f-4698-b7a2-d4b84df79bd2"

```

28. Validation Password saat register

Branch: validationPassword

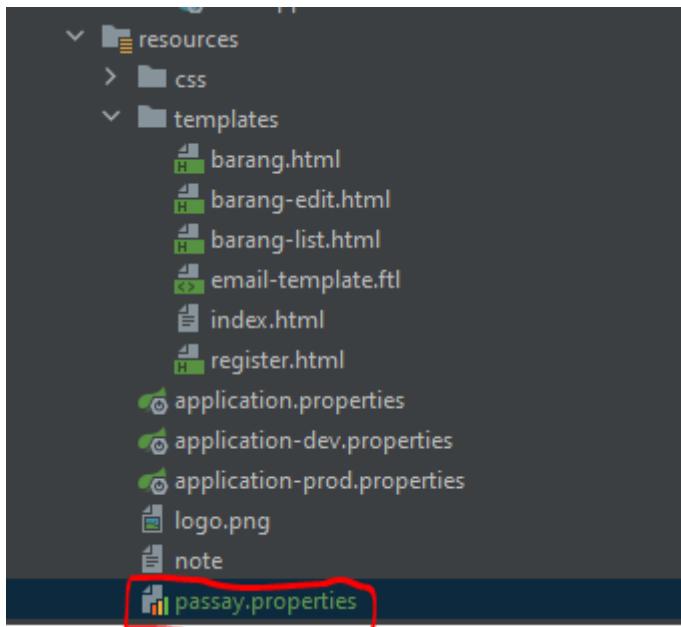
url : <https://github.com/rikialdi/dibimbing.git>

a. Step 1- Pom.xml

```
<!--validation-->
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>2.0.1.Final</version>
</dependency>
    <!--digunakan untuk validasi password -->
<dependency>
    <groupId>org.passay</groupId>
    <artifactId>passay</artifactId>
    <version>1.6.0</version>
</dependency>
```

b. Step 2-application properties

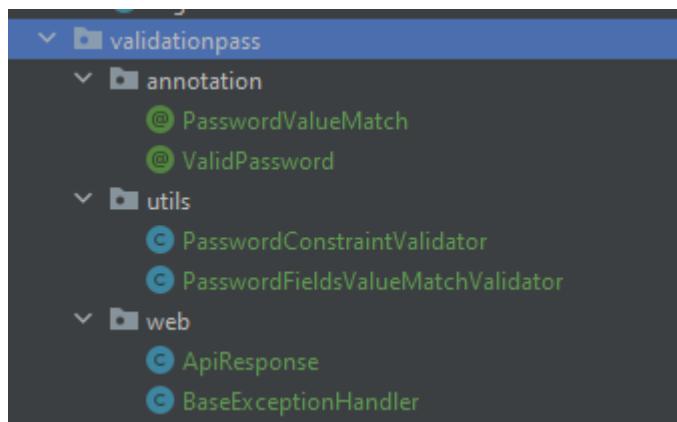
Buatlah file baru di resources:



```
HISTORY_VIOLATION=Password matches one of %1$s previous passwords.  
ILLEGAL_WORD=Password contains the dictionary word '%1$s'.  
ILLEGAL_WORD_REVERSED=Password contains the reversed dictionary word  
'%1$s'.  
ILLEGAL_DIGEST_WORD=Password contains a dictionary word.  
ILLEGAL_DIGEST_WORD_REVERSED=Password contains a reversed dictionary  
word.  
ILLEGAL_MATCH=Password matches the illegal pattern '%1$s'.  
ALLOWED_MATCH=Password must match pattern '%1$s'.  
ILLEGAL_CHAR=Password %2$s the illegal character '%1$s'.  
ALLOWED_CHAR=Password %2$s the illegal character '%1$s'.  
ILLEGAL_QWERTY_SEQUENCE=Password contains the illegal QWERTY sequence  
'%1$s'.  
ILLEGAL_ALPHABETICAL_SEQUENCE=Password contains the illegal  
alphabetical sequence '%1$s'.  
ILLEGAL_NUMERICAL_SEQUENCE=Password contains the illegal numerical  
sequence '%1$s'.  
ILLEGAL_USERNAME=Password %2$s the user id '%1$s'.  
ILLEGAL_USERNAME_REVERSED=Password %2$s the user id '%1$s' in  
reverse.  
ILLEGAL_WHITESPACE=Password %2$s a whitespace character.  
ILLEGAL_NUMBER_RANGE=Password %2$s the number '%1$s'.  
ILLEGAL_REPEAT_CHARS=Password contains %3$s sequences of %1$s or  
more repeated characters, but only %2$s allowed: %4$s.  
INSUFFICIENT_UPPERCASE>Password must contain %1$s or more uppercase  
characters.  
INSUFFICIENT_LOWERCASE>Password must contain %1$s or more lowercase  
characters.  
INSUFFICIENT_ALPHABETICAL=Password must contain %1$s or more  
alphabetical characters.  
INSUFFICIENT_DIGIT=Password must contain %1$s or more digit  
characters.  
INSUFFICIENT_SPECIAL=Password must contain %1$s or more special  
characters.  
INSUFFICIENT_CHARACTERISTICS=Password matches %1$s of %3$s character  
rules, but %2$s are required.  
INSUFFICIENT_COMPLEXITY=Password meets %2$s complexity rules, but
```

```
%3$s are required.
INSUFFICIENT_COMPLEXITY_RULES=No rules have been configured for a
password of length %1$s.
SOURCE_VIOLATION=Password cannot be the same as your %1$s password.
TOO_LONG=Password must be no more than %2$s characters in length.
TOO_SHORT=Password must be %1$s or more characters in length.
TOO_MANY_OCCURRENCES=Password contains %2$s occurrences of the
character '%1$s', but at most %3$s are allowed.
```

c. Step 3- struktur kode



Class PasswordValueMatch

```
package com.binar.grab.controller.validationpass.annotation;

import com.binar.grab.controller.validationpass.utils.PasswordFieldsValueMatchValidator;

import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.*;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

/**
 * <h2>PasswordValueMatch</h2>
 *
 * @author aek
 */
@Target({ TYPE, ANNOTATION_TYPE })
@Retention(RUNTIME)
@Constraint(validatedBy = PasswordFieldsValueMatchValidator.class)
@Documented
public @interface PasswordValueMatch {

    String message() default "Fields values don't match!";
```

```

    Class<?>[] groups() default { };

    Class<? extends Payload>[] payload() default { };

    String field();

    String fieldMatch();

    @Target({ ElementType.TYPE })
    @Retention(RetentionPolicy.RUNTIME)
    @interface List {
        PasswordValueMatch[] value();
    }
}

```

Class ValidPassword

```

package com.binar.grab.controller.validationpass.annotation;

import com.binar.grab.controller.validationpass.utils.PasswordConstraintValidator;

import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

/**
 * <h2>ValidPassword</h2>
 *
 * @author aek
 */
@Documented
@Constraint(validatedBy = PasswordConstraintValidator.class)
@Target({ TYPE, FIELD, ANNOTATION_TYPE })
@Retention(RUNTIME)
public @interface ValidPassword {

    String message() default "Invalid Password";

    Class<?>[] groups() default {};

    Class<? extends Payload>[] payload() default {};
}

```

Class PasswordConstraintValidator

```

package com.binar.grab.controller.validationpass.utils;

import com.binar.grab.controller.validationpass.annotation.ValidPassword;
import lombok.SneakyThrows;
import org.passay.*;
import org.passay.PropertiesMessageResolver;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;
import java.io.InputStream;
import java.util.Arrays;
import java.util.List;
import java.util.Properties;

/**
 * <h2>PasswordConstraintValidator</h2>
 *
 * @author aek
 */
public class PasswordConstraintValidator implements ConstraintValidator<ValidPassword, String> {

    @Override
    public void initialize(final ValidPassword arg0) {

    }

    @SneakyThrows
    @Override
    public boolean isValid(String password,
    ConstraintValidatorContext context) {

        //customizing validation messages
        Properties props = new Properties();
        InputStream inputStream = getClass()

.getClassLoader().getResourceAsStream("passay.properties");
        props.load(inputStream);
        MessageResolver resolver = new
PropertiesMessageResolver(props);

        PasswordValidator validator = new PasswordValidator(resolver,
        Arrays.asList(
/*
example : Aa1@231456
*/
            // length between 8 and 16 characters
            new LengthRule(8, 16),

            // at least one upper-case character
            new CharacterRule(EnglishCharacterData.UpperCase, 1),

            // at least one lower-case character
            new CharacterRule(EnglishCharacterDataLowerCase, 1),

```

```

        // at least one digit character
        new CharacterRule(EnglishCharacterData.Digit, 1),

        // at least one symbol (special character)
        new CharacterRule(EnglishCharacterData.Special, 1),

        // no whitespace= tidak dengan spasi
        new WhitespaceRule(),

        // rejects passwords that contain a sequence of >= 5
        characters alphabetical (e.g. abcdef)
        //tidak boleh karakter berurut contoh abcde ,ini
        tidak boleh, jika abczd : ini boleh
        new
        IllegalSequenceRule(EnglishSequenceData.Alphabetical, 5, false),
        // rejects passwords that contain a sequence of >= 5
        characters numerical (e.g. 12345)
        //tidak boleh numeric berurut contoh 12345 ,ini tidak
        boleh, jika `12365` : ini boleh
        new
        IllegalSequenceRule(EnglishSequenceData.Numerical, 5, false)
    );

    RuleResult result = validator.validate(new
    PasswordData(password));

    if (result.isValid()) {
        return true;
    }

    List<String> messages = validator.getMessages(result);
    String messageTemplate = String.join(", ", messages);
    context.buildConstraintViolationWithTemplate(messageTemplate)
        .addConstraintViolation()
        .disableDefaultConstraintViolation();
    return false;
}
}

```

Class PasswordFieldsValueMatchValidator

```

package com.binar.grab.controller.validationpass.utils;

import
com.binar.grab.controller.validationpass.annotation.PasswordValueMatch;
import org.springframework.beans.BeanWrapperImpl;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

/**
 * <h2>PasswordFieldsValueMatchValidator</h2>
 *
 * @author aek

```

```

/*
public class PasswordFieldsValueMatchValidator implements
ConstraintValidator<PasswordValueMatch, Object> {

    private String field;
    private String fieldMatch;
    private String message;

    public void initialize(PasswordValueMatch constraintAnnotation) {
        this.field = constraintAnnotation.field();
        this.fieldMatch = constraintAnnotation.fieldMatch();
        this.message = constraintAnnotation.message();
    }

    public boolean isValid(Object value,
                          ConstraintValidatorContext context) {

        Object fieldValue = new BeanWrapperImpl(value)
            .getPropertyValue(field);
        Object fieldMatchValue = new BeanWrapperImpl(value)
            .getPropertyValue(fieldMatch);

        boolean isValid = false;
        if (fieldValue != null) {
            isValid = fieldValue.equals(fieldMatchValue);
        }

        if (!isValid) {
            context.disableDefaultConstraintViolation();
            context
                .buildConstraintViolationWithTemplate(message)
                .addPropertyNode(field)
                .addConstraintViolation();
            context
                .buildConstraintViolationWithTemplate(message)
                .addPropertyNode(fieldMatch)
                .addConstraintViolation();
        }
    }

    return isValid;
}
}

```

Class Api Response

```

package com.binar.grab.controller.validationpass.web;

import lombok.*;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString

```

```

public class ApiResponse {

    private Object data;
    private String message;
    private boolean error = true;

    public ApiResponse(Object data, String message) {
        this.data = data;
        this.message = message;
    }
}

```

Class BaseExceptionHandler

```

package com.binar.grab.controller.validationpass.web;

import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;
import java.util.Map;

@Slf4j
@RestControllerAdvice
public class BaseExceptionHandler {

    @ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ApiResponse handleValidationExceptions(MethodArgumentNotValidException ex) {

        Map<String, String> errors = new HashMap<>();

        ex.getBindingResult().getFieldErrors().forEach(error -> {
            if (errors.containsKey(error.getField())) {
                errors.put(error.getField(),
String.format("%s, %s", errors.get(error.getField()),
error.getDefaultMessage()));
            } else {
                errors.put(error.getField(),
error.getDefaultMessage());
            }
        });
        return new ApiResponse(errors, "VALIDATION_FAILED");
    }
}

```

d. Step 3 a – setting Character Password dimana?

Ada di class : PasswordConstraintValidator

```
new CharacterRule(EnglishCharacterData.UpperCase, num: 1),
new CharacterRule(EnglishCharacterData.LowerCase, num: 1),
new CharacterRule(EnglishCharacterData.Digit, num: 1),
new CharacterRule(EnglishCharacterData.Special, num: 1),
new WhitespaceRule(),
// rejects passwords that contain a sequence of >= 5 characters alphabetical
// tidak boleh karakter berurut contoh abcde ,ini tidak boleh, jika abcde : i
new IllegalSequenceRule(EnglishSequenceData.Alphabetical, sl: 5, wrap: false)
// rejects passwords that contain a sequence of >= 5 characters numerical (
// tidak boleh numeric berurut contoh 12345 ,ini tidak boleh, jika '12345' : i
new IllegalSequenceRule(EnglishSequenceData.Numerical, sl: 5, wrap: false)
```

e. Step 4 – cara menggunakan method validation password ?

Misal api yang digunakan adalah : api register, ingin validasi password

Pada controller register, tambahkn text yang di kotak merah dibawah ini :

```
@Autowired
public EmailSender emailSender;

//step 1
@PostMapping("/register")
@ExceptionHandler(ConstraintViolationException.class)
public ResponseEntity<Map> saveRegisterManual(@Valid @RequestBody RegisterModel objModel) throws Exception {
    Map map = new HashMap();
    User user = userRepository.checkExistingEmail(objModel.getEmail());
    if (null != user) {
        return new ResponseEntity<Map>(templateCRUD.notFound("objek: " + user.getUsername() + " sudah ada"), HttpStatus.OK);
    }
    map = serviceReq.registerManual(objModel);
    Map sendOTP = sendEmailRegister(objModel);
    return new ResponseEntity<Map>(map, HttpStatus.OK);
}
```

Pada RegisterModel sebagai Requwat : tambahkan text yang di kotak merah,seperti gmabar dibawah ini “

The screenshot shows a Java code editor with several files listed in the left sidebar. The current file is `RegisterModel.java`. The code contains annotations for password validation:

```

    @PasswordFieldsValueMatchValidator.java
    Register.java
    RegisterModel.java

    ...
    @PasswordFieldMatch.List({
        @PasswordFieldMatch(
            field = "password",
            fieldMatch = "confirmPassword",
            message = "Passwords do not match!"
        )
    })

    public class RegisterModel {
        public Long id;

        public String email;

        public String username;

        @ValidPassword
        @NotEmpty(message = "Password is mandatory")
        public String password;

        @ValidPassword
        @NotEmpty(message = "Confirm Password is mandatory")
        private String confirmPassword;
    }

```

f. Tester

Failed:

The screenshot shows a Postman request for the endpoint `localhost:9090/api/user-register/register`. The request method is POST. The body contains the following JSON payload:

```

1 ... "email": "rikialdipari98@gmail.com",
2 ... "password": "1234567",
3 ... "confirmPassword": "1234567",
4 ... "fullname": "riki aldi pari"

```

The "password" and "confirmPassword" fields are highlighted with a red box. The response status is 400 Bad Request, with the message:

```

1 "data": {
2     "password": "Password must be 8 or more characters in length.,Password must contain 1 or more uppercase characters",
3     "confirmPassword": "Password must be 8 or more characters in length.,Password must contain 1 or more uppercase characters"
4 },
5 "message": "VALIDATION_FAILED",
6 "error": true

```

Success:

Binar Grab / register / register Pass Valida

POST localhost:9090/api/user-register/register

Params Authorization Headers (9) Body **JSON** Pre-request Script Tests Settings Cookies Beautify

Body Cookies Headers (18) Test Results Status: 200 OK Time: 3.44 s Size: 3.67 KB Save Response

```

1 {
2   ...
3   "email": "rikialdipari99@gmail.com",
4   "password": "@Ab12z345",
5   "confirmPassword": "@Ab12z345",
6   "fullname": "riki aldi pari"

```

Pretty Raw Preview Visualize JSON

```

1 > "data": { ...
271 },
272   "message": "sukses",
273   "status": "200"
274

```

Activate Windows
Go to Settings to activate Windows.

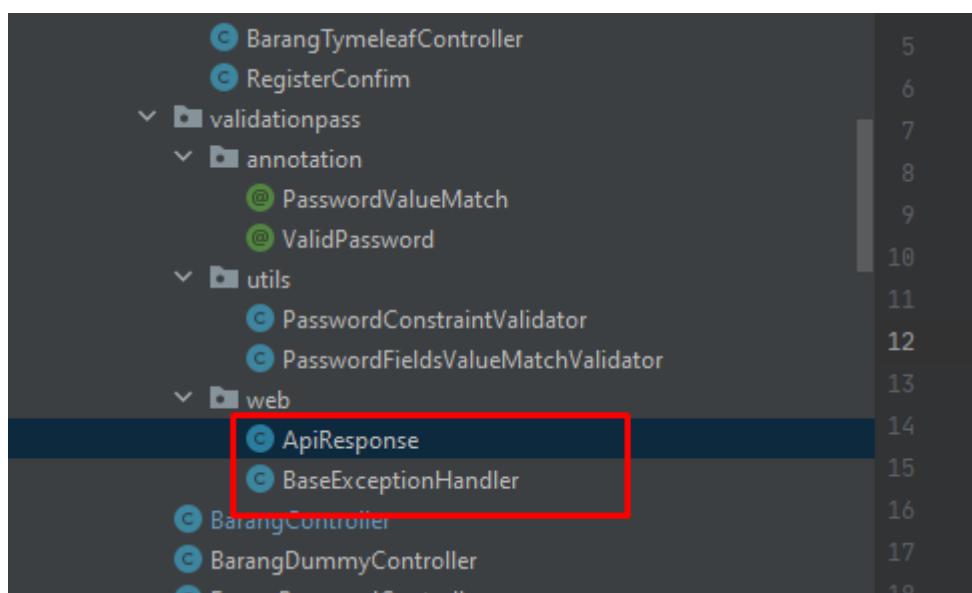
Cookies Bootcamp Desktop Agent Runner Trash

29. Menampilkan Error Validation Hibernate

Branch : showErrorHibernate

url : <https://github.com/rikialdi/dibimbing.git>

a. Strukture



apiResponse

```
package com.binar.grab.controller.validationpass.web;

import lombok.*;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class ApiResponse {

    private Object data;
    private String message;
    private boolean error = true;

    public ApiResponse(Object data, String message) {
        this.data = data;
        this.message = message;
    }
}
```

Handler

```
package com.binar.grab.controller.validationpass.web;

import lombok.extern.slf4j.Slf4j;
import org.apache.tomcat.util.http.fileupload.impl.SizeLimitExceededException;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import org.springframework.web.multipart.MaxUploadSizeExceededException;

import java.util.HashMap;
import java.util.Map;

@Slf4j
@RestControllerAdvice
public class BaseExceptionHandler {

    @ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ApiResponse handleValidationExceptions(MethodArgumentNotValidException ex) {
        Map<String, String> errors = new HashMap<>();
        ...
    }
}
```

```

        ex.getBindingResult().getFieldErrors().forEach(error -> {
            if (errors.containsKey(error.getField())) {
                errors.put(error.getField(), String.format("%s,
%s", errors.get(error.getField()), error.getDefaultMessage())));
            } else {
                errors.put(error.getField(),
error.getDefaultMessage());
            }
        });
        return new ApiResponse(errors, "VALIDATION_FAILED");
    }

    @ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler(SizeLimitExceededException.class)
    public ApiResponse
handleValidationExceptions2(MethodArgumentNotValidException ex) {

    Map<String, String> errors = new HashMap<>();

    ex.getBindingResult().getFieldErrors().forEach(error -> {
        if (errors.containsKey(error.getField())) {
            errors.put(error.getField(), String.format("%s,
%s", errors.get(error.getField()), error.getDefaultMessage())));
        } else {
            errors.put(error.getField(),
error.getDefaultMessage());
        }
    });
    return new ApiResponse(errors, "VALIDATION_FAILED");
}
}

```

b. POM.xml

```

<!--validation-->
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>2.0.1.Final</version>
</dependency>

```

c. Controller

Pada setiap controller tambahkan yang dikotak merah . dan ditambahkan @Valid

```
@PostMapping("/save/{idsupplier}")
@ExceptionHandler(ConstraintViolationException.class)
public ResponseEntity<Map> save(@PathVariable(value = "idsupplier") Long idsupplier,
                                  @Valid @RequestBody Barang objModel) {
    Map obj = barangService.insert(objModel, idsupplier);
    return new ResponseEntity<Map>(obj, HttpStatus.OK);
}

@ExceptionHandler(ConstraintViolationException.class)
```

d. Entity

Setiap req atau entity tambahkan yang dikotak merah dibawah ini

```
public class Barang extends AbstractDate implements Serializable {

    @Id
    @Column(name="id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    // ...
    // ...
    @NotNull(message = "nama is mandatory")
    @NotEmpty(message = "Nama is mandatory")
    @Column(name = "nama", nullable = false, length = 45)
    private String nama;

    @Column(name = "stok", nullable = false, length = 11)
    private int stok;

    @Column(name = "satuan", nullable = false, length = 45)
```

e. Testing

The screenshot shows a Postman interface with a POST request to `localhost:9090/api/v1/binar/barang/save/2`. The request body is set to JSON and contains the following data:

```
1
2   "nama": null,
3   "stok": "200"
```

The response status is 400 Bad Request, and the response body is:

```
1
2   "data": [
3     {
4       "nama": "Nama is mandatory"
5     }
6   ],
7   "message": "VALIDATION_FAILED",
```

30. Validasi

a. Validasi Email manual

```
public boolean isValidEmail(String email)
{
    String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\.|"
        "[a-zA-Z0-9_+&*-]+)*@" + "(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";
    Pattern pat = Pattern.compile(emailRegex);
    return pat.matcher(email).matches();
}
```

b. Validasi String manual

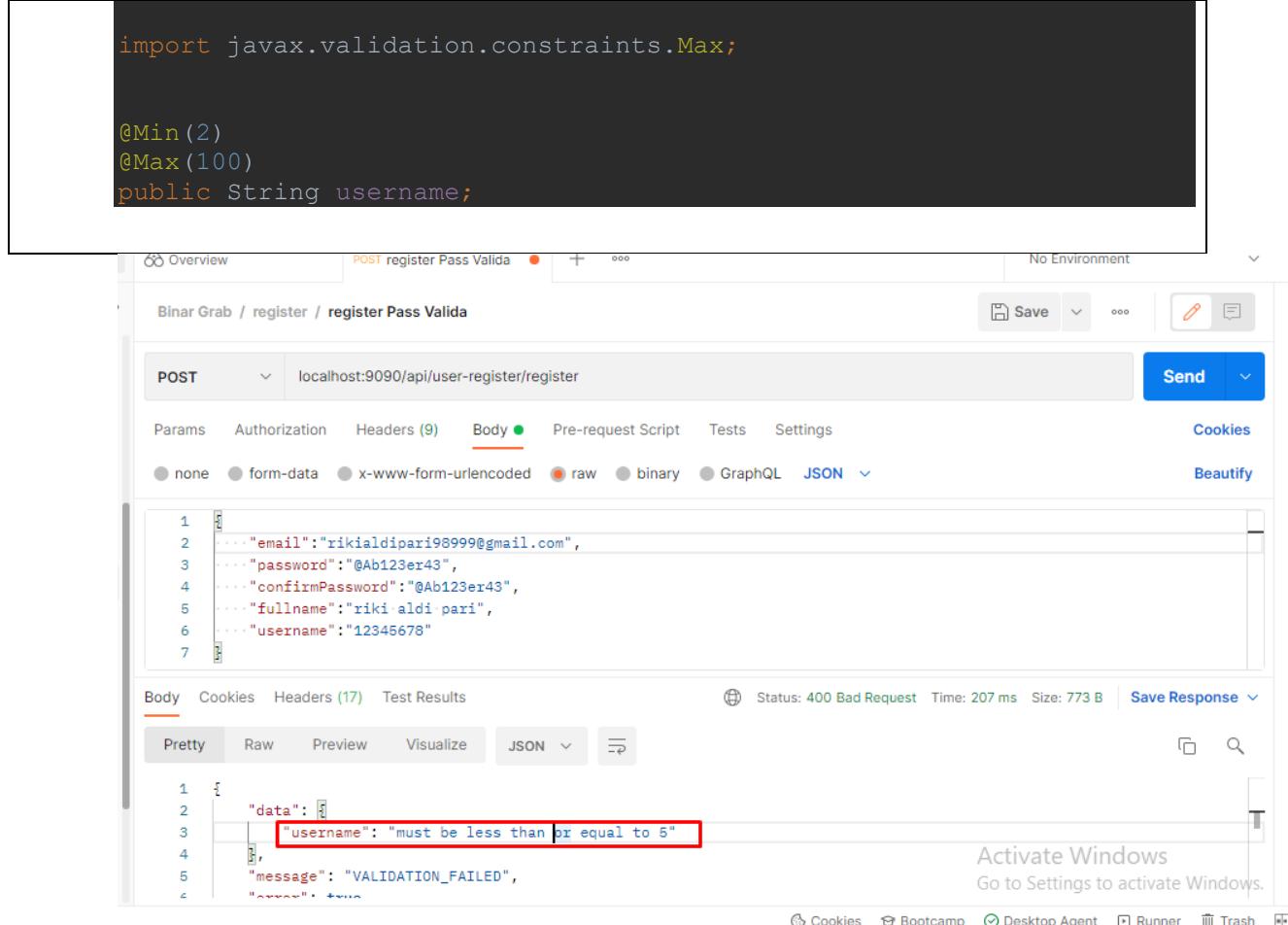
```
//
//      if (StringUtils.isEmpty(request.getKontent())) {
//          return new ResponseEntity<Map>(response.Error("Kontent
Wajib diisi."), HttpStatus.BAD_REQUEST);
//      }
//      if (StringUtils.isEmpty(request.getSubjek())) {
//          return new ResponseEntity<Map>(response.Error("Subjek
Wajib diisi."), HttpStatus.BAD_REQUEST);
//      }
//      if (StringUtils.isEmpty(request.getPengirim())) {
```

```
//             return new ResponseEntity<Map>(response.Error("Pengirim  
Wajib diisi."), HttpStatus.BAD_REQUEST);  
//         }
```

31. Anotasi hibernate

a. @Max @Min :Batasin Request maksimal 5 huruf

```
import javax.validation.constraints.Max;  
  
 @Min(2)  
 @Max(100)  
 public String username;
```



The screenshot shows a POST request to `localhost:9090/api/user-register/register`. The request body is a JSON object with several fields. One of the fields, `username`, has a value of `12345678`. The response status is `400 Bad Request`, and the error message is `"username": "must be less than or equal to 5"`.

b. @Size Batasin Request maksimal 5 huruf

```
import javax.validation.constraints.*;  
  
 @Size(  
     min = 5,  
     max = 14,  
     message = "The author email '${validatedValue}' must be
```

```
between {min} and {max} characters long"
)
public String email;
```

The screenshot shows a Postman collection named "Binar Grab / register / register Pass Valida". A POST request is made to "localhost:9090/api/user-register/register" with the following body:

```
1 ... "email": "rikialdipari98999@gmail.com",
2 ... "password": "@Ab123er43",
3 ... "confirmPassword": "@Ab123er43",
4 ... "fullname": "riki aldi pari",
5 ... "username": "1234"
```

The response status is 400 Bad Request, with the following validation errors:

```
1 "data": [
2   {
3     "email": "The author email 'rikialdipari98999@gmail.com' must be between 5 and 14 characters long",
4     "username": "must be less than or equal to 100"
5   }
6 ]
```

32. Mengenal Lambda Expression untuk Membuat Fungsi Anonymous di Java

Ref: <https://www.petanikode.com/java-lambda/>

Istilah *anonymous* memang untuk menggambarkan sesuatu yang tak memiliki nama. Nah di Java, juga ada fungsi *anonymous* dan [class anonymous](#).

Apa itu fungsi anonymous?

Bagaimana cara membuatnya?

Mengapa harus menggunakan fungsi anonymous?

a. Apa itu Fungsi Anonymous?

Fungsi anonymous adalah fungsi yang tidak memiliki nama. Fungsi anonymous di Java dikenal juga dengan ***lambda expression***.

Fungsi anonymous biasanya dibuat hanya untuk sekali pakai.

Artinya, saat kita membuat fungsi anonymous, kita akan mengeksekusinya saat itu juga. Tidak bisa dipanggil lagi seperti fungsi biasa.

Fungsi ini mulai ditambahkan pada JDK 8.

Jika kamu menggunakan JDK di bawah 8, maka kamu harus upgrade dulu agar bisa menggunakan fungsi anonymous.

Oke, kalau begitu.. bagaimana cara membuat fungsi anonymous?

b. Cara Membuat Fungsi Anonymous di Java

Berikut ini adalah bentuk umum *lambda expression* atau fungsi anonymous di Java:



Ini simbol-simbol yang perlu kamu ingat:

`() -> {}`

Keterangan:

- `()` tempat kita menaruh parameter.
- `->` (operator lambda) simbol yang menandakan fungsi ini adalah lambda/anonymous.
- `{}` body fungsinya.

Contoh:

```
(int x, int y) -> { return x + y }
```

Sebenarnya simbol yang perlu diingat adalah simbol lambda (`->`), karena simbol ini yang membedakan fungsi anonymous dengan fungsi biasa.

Untuk tanda kurung yang ini `()` dengan yang ini `{}`, di fungsi biasa juga ada.

Mari kita lihat contohnya:

```
// ini fungsi biasa
int jumlahkan(int a, int b){
    return a + b;
}

// ini fungsi anonymous
(int a, int b) -> { return a + b }
```

Bahkan tanpa kurung kurawal juga bisa:

```
(int a, int b) -> return a + b;
```

Karena cuma ada satu baris ekspresi, maka tanda kurung `{}` boleh tidak ada.

Oh iya, fungsi anonymous dapat kita buat di berbagai tempat seperti:

- Pada Deklarasi variabel;Contoh:

```
int variabel = () -> { return 0 };
```

- Pada pengisian variabel dan array;Contoh:

```
int variabel;
int arr;
variabel = () -> { return 0 };
arr = () -> { return {0,4,3,2,1} };
```

- Pada saat mengembalikan nilai dengan return;Contoh:

```
int methodName(){
```

```
    return () -> { return 0 };
}
```

- Pada body lambda itu sendiri;Contoh:

```
() -> {
    return () -> 5 + 2;
};
```

- Pada ekspresi kondisional (`? :`)Contoh:

```
String jawab = (int x) -> { x < 10} ? () -> return "yes": () -> retu
"no";
```

c. Mengapa Harus Pakai Fungsi Anonymous?

Lambda expression atau fungsi anonymous sebenarnya hadir untuk menyempurnakan [class anonymous](#).¹

Class anonymous biasanya digunakan untuk mengimplementasikan interface dan [class abstrak](#).

Tapi salahnya:

Saat interface hanya memiliki satu method saja untuk diimplementasikan. Kita harus membuat class (anonymous) baru.

Padahal kan kita cuma butuh method-nya saja.

Di sini lah saat yang tepat untuk menggunakan fungsi anonymous atau *lambda expression*.

Bahkan Netbeans juga akan menyarankan menggunakan *lambda expression* apabila menemukan kasus ini.

d. Contoh Program Fungsi Anonymous

Buatlah proyek baru di Netbeans dengan nama **ContohLambda**.

Setelah itu, buat sebuah interface baru pada pacakage <default package> dengan nama **Clickable** dan isi kodennya seperti ini:

```
interface Clickable {  
    void onClick();  
}
```

Setelah itu, buatlah class **Button** dengan isi seperti ini:

```
public class Button {  
    private Clickable action;  
  
    void setClickAction(Clickable action){  
        this.action = action;  
    }  
  
    void doClick(){  
        action.onClick();  
    }  
}
```

Terakhir, buatlah class **Main** dengan isi seperti ini:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Button btn = new Button();  
  
        // membuat class anonymous untuk implementasi interface  
        btn.setClickAction(new Clickable() {  
            @Override  
            public void onClick() {  
                System.out.println("Tombol sudah diklik!");  
            }  
        });  
    }  
}
```

```
        System.out.println("Yay!");
    }
};

// mencoba klik tombol
btn.setOnClickListener();

}
}
```

Perhatikan baris kode yang saya tandai di atas..

Di sana kita menggunakan class anonymous untuk mengimplementasikan interface `Clickable`.

Nah, kalau mau lebih sederhana.. kita bisa pakai *lambda expression*.

Sekarang coba ubah kodennya menjadi seperti ini:

```
public class Main {

    public static void main(String[] args) {

        Button btn = new Button();

        // membuat class anonymous untuk implementasi interface
        btn.setOnClickListener(() -> {
            System.out.println("Tombol sudah diklik!");
            System.out.println("Yay!");
        });

        // mencoba klik tombol
        btn.setOnClickListener();
    }
}
```

Lebih sederhana yang mana?

Tentu saja yang menggunakan *lambda expression*.

Berikut ini hasil output dari program tersebut:

e. Akses Variabel untuk Fungsi Anonymous

Pada dasarnya, fungsi anonymous adalah bentuk sederhana dari class anonymous.

Jadi dia akan memiliki akses variabel yang sama seperti class anonymous.

Kalau tidak percaya, coba buktikan dengan membuat variabel di luar fungsi anonymous.

Lalu akses variabel tersebut dari dalam fungsi anonymous.

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Button btn = new Button();  
        String name = "Petani Kode";  
  
        // membuat class anonymous untuk implementasi interface  
        btn.setOnClickListener(() -> {  
            System.out.println("Tombol sudah diklik!");  
            System.out.println("Yay!");  
            System.out.println("Hello " + name);  
        });  
  
        // mencoba klik tombol  
        btn.setOnClickListener();  
  
    }  
}
```

Fungsi anonymous akan bisa mengakses variabel yang berada di dalam class (variabel global), dan lokal yang ada di method tempat anonymous class digunakan.

f. Akhir Kata...

Intinya:

Fungsi anonymous hanya bisa digunakan saat ingin mengimplementasikan interface yang memiliki satu method.

Jika interface itu memiliki lebih dari satu method, maka sebaiknya pakai class anonymous.

33. MicroService

a. Apa itu MicroService ?

Microservices adalah suatu framework *Architecture* yang dipakai sebagai model dalam pembuatan aplikasi **cloud** yang modern. Di dalam microservices setiap aplikasi dibangun sebagai sekumpulan service dan setiap layanan berjalan dalam processnya sendiri. Masing-masing dari aplikasi tersebut saling berkomunikasi melalui *API (Application Programming Interface)*.

Microservices merupakan *Architecture* dalam pembangunan aplikasi cloud yang modern. *Architecture* microservices bersifat terdistribusi, sehingga perubahan pada program yang dilakukan oleh satu tim developer tidak mengganggu keseluruhan aplikasi.

b. Apa Manfaat ?

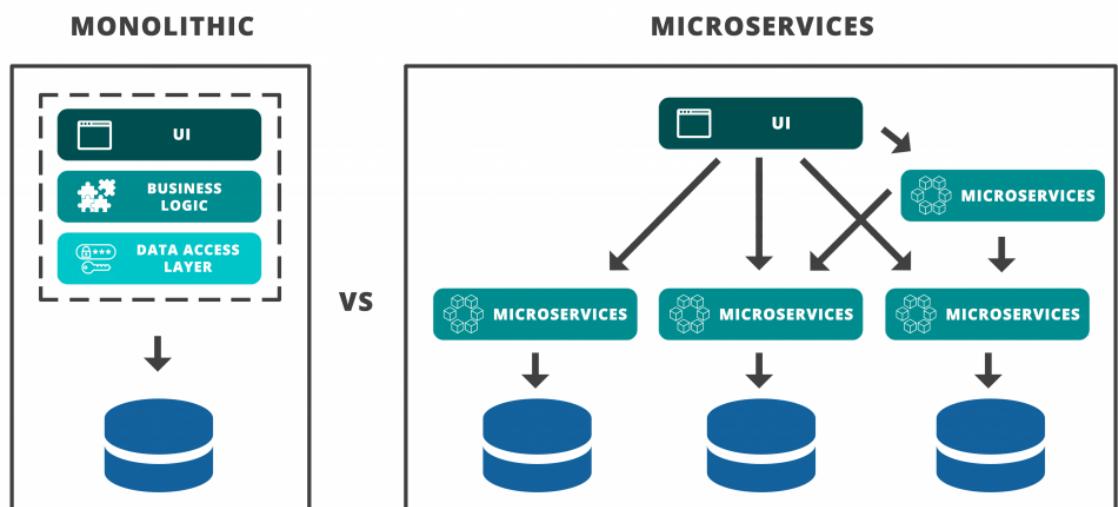
Manfaat utama dalam mempergunakan microservices adalah agar team developer mampu mengembangkan aplikasi secara cepat dengan membuat komponen-komponen dari aplikasi berjalan secara independen sehingga dapat memenuhi kebutuhan bisnis yang terus menerus berubah.

Cara pembangunan aplikasi yang seperti ini dioptimalkan dengan mempergunakan *DevOps* (*Development and Operation*) dan *CI / CD* (*Continuous Integration and Continuous Delivery*)

c. Perbedaan MicroService VS Monolitik

Apa yang membedakan *Architecture Microservices* dengan pendekatan yang lebih tradisional seperti *Monolithic Architecture* adalah bagaimana framework ini memecah aplikasi menjadi fungsi intinya. Setiap fungsi ini disebut sebagai *service*, dapat dibangun dan dijalankan secara independen, yang berarti *service* tersebut dapat berfungsi (dan gagal) tanpa berdampak negatif pada fungsi-fungsi yang lain.

Framework ini membantu dalam sisi teknologi dari DevOps dalam pelaksanaan *continuous integration* dan *continuous delivery* (*CI / CD*) sehingga menjadi lebih mulus dan dapat tercapai.



d. *Monolithic Architecture*

Untuk memahami keunggulan *Microservice Architecture* yang terkini, sangat penting untuk memahami dari mana semuanya dimulai. Awalnya setiap aplikasi yang berada di satu server, terdiri dari tiga lapisan:

- ✓ Presentasi – User Interface
- ✓ Aplikasi / logika bisnis
- ✓ Database – Data Access Layer

Layer-layer ini dibuat dalam satu stack yang saling terkait dan terletak di satu server *Monolithic* di **datacenter**. Pola ini umumnya ada di setiap industry vertikal dan dalam *Architecture* teknologi. Secara umum, aplikasi adalah kumpulan modul programming code yang melayani fungsi tertentu — misalnya, database, berbagai jenis logika bisnis, kode rendering grafik, atau logging.

Dalam *Monolithic Architecture* ini, user berinteraksi dengan *layer* presentasi, yang berbicara dengan *layer* logika bisnis dan kemudian dengan *layer* database, dan informasi atau data yang dicari, kemudian melakukan perjalanan kembali ke *layer* tersebut tersebut untuk mencapai ke pengguna di *layer* teratas.

Meskipun ini adalah cara yang efisien untuk menjalankan aplikasi, hal ini menciptakan banyak titik kegagalan yang dapat mengakibatkan system down yang lama pada saat terjadi kegagalan pada perangkat keras atau bug code.

Sayangnya, “self-healing” tidak ada dalam struktur ini. Jika ada bagian dari sistem yang rusak, maka perlu dilakukan perbaikan dengan campur tangan manusia dalam bentuk perbaikan perangkat keras atau perangkat lunak.

Untuk melakukan scale-up salah satu dari *layer* ini berarti membeli server baru. Anda harus membeli aplikasi *Monolithic* yang berjalan di satu server dan mensegmentasi sebagian pengguna ke sistem baru. Segmentasi ini menghasilkan data silo ke pengguna yang harus direkonsiliasi dengan laporan batch setiap malam. Untungnya, kebutuhan klien menjadi lebih tipis karena laman web dan aplikasi seluler menjadi lebih populer, sehingga metode baru pengembangan aplikasi mulai terbentuk.

e. *Microservice Architecture*

Akhir-akhir ini cloud microservice memecah strategi SOA (Service Oriented Architecture) lebih jauh menjadi kumpulan service fungsional yang lebih

granular. Koleksi microservices ini digabungkan ke dalam microservice besar, memberikan kemampuan yang lebih untuk melakukan fungsi self-healing dengan cepat di layanan keseluruhan atau aplikasi *inducer* di dalam suatu aplikasi yang besar.

Sebuah microservice mencoba untuk menangani satu masalah, seperti pencarian data, fungsi logging, atau fungsi layanan web. Pendekatan ini meningkatkan fleksibilitas — misalnya, memperbarui program satu fungsi tanpa harus melakukan reinstallation atau bahkan men-deploy ulang ke *Architecture* microservice lainnya. Titik-titik kegagalan menjadi independen dan terisolasi satu sama lainnya, sehingga menciptakan *Architecture* aplikasi yang lebih stabil secara keseluruhan.

f. MicroService and Container

Microservices digunakan secara bersamaan dengan Docker Container. **Container** adalah perantara yang sangat tepat untuk mendeploy Microservices. Container dapat diluncurkan dalam hitungan detik, sehingga dapat diterapkan kembali dengan cepat setelah kegagalan atau migrasi, dan dapat di scale-up dengan cepat untuk memenuhi permintaan.

Container bahkan lebih efisien daripada VM, memungkinkan kode (dan *code library* yang diperlukan) untuk diterapkan pada sistem Linux apa pun (atau OS apa pun yang mendukung kontainer Docker).

Karena Container adalah bawaan Linux, perangkat keras komoditas dapat diterapkan ke berbagai microservices di datacenter, private cloud, atau multicloud hybrid.

g. Apakah keuntungan dari *Microservice Architecture*?

Microservices memberikan tim Development keleluasaan untuk melakukan pengembangan program kode secara terdistribusi. Masing-masing dari programmer team dapat melakukan pembuatan komputer program secara bersamaan. Ini berarti lebih banyak developer yang mengerjakan aplikasi yang

sama, pada waktu yang sama. Yang mana dapat membuat lebih sedikit waktu yang dihabiskan dalam pengembangan.

Lebih cepat menanggapi kebutuhan pasar	Karena siklus pengembangan dipersingkat, <i>Architecture microservices</i> mendukung penerapan dan <i>update</i> yang lebih gesit.
Sangat Scalable	Saat permintaan untuk service tertentu tumbuh, Anda dapat menerapkan di beberapa server, dan infrastruktur, untuk memenuhi kebutuhan Anda.
Handal	<i>Service Independent</i> ini jika dibangun dengan benar tidak akan berdampak satu sama lain. Artinya, jika salah satu bagian gagal, seluruh aplikasi tidak akan mati, tidak seperti model aplikasi <i>Monolithic</i> .
Aksesibilitas	Karena aplikasi yang lebih besar dipecah menjadi bagian-bagian yang lebih kecil, developer dapat lebih mudah memahami, memperbarui, dan menyempurnakan bagian tersebut, menghasilkan siklus pengembangan yang lebih cepat, terutama jika dikombinasikan dengan metodologi pengembangan yang gesit.
Lebih terbuka	Karena penggunaan API dan poliglot, pengembang memiliki kebebasan untuk memilih Bahasa pemrograman dan teknologi terbaik untuk fungsi yang diperlukan.

Ref:<https://datacommcloud.co.id/microservices-adalah-perbedaan-monolithic-architecture/>

h. Contoh penerapan MicroService

Service 1: Employee

Branch : <https://github.com/rikialdi/microservice-first.git>



```
14     @RestController
15     @RequestMapping("/employee")
16     public class FirstController {
17
18         1 usage
19         @Bean
20         public TestRestTemplate restTemplate(){
21             return new TestRestTemplate();
22         }
23
24         @GetMapping("/message")
25         public String test() {
26             callServiceEmployee();
27             return "Hello JavaInUse Called in First Service and Called in Second Service";
28         }
29
30         1 usage
31         public void callServiceEmployee() {
32             HttpHeaders headers = new HttpHeaders();
33         }
34     }
```

Service ini memanggil service yang kedua : yaitu service consumer



```
1        1 usage
2        public void callServiceEmployee() {
3            HttpHeaders headers = new HttpHeaders();
4            headers.set("Accept", "*/*");
5            headers.set("Content-Type", "application/json");
6            String url = "http://localhost:8082/consumer/message";
7            ResponseEntity<String> exchange = restTemplate().exchange(url, HttpMethod.GET, requestEntity, null, String.class);
8
9            System.out.println("response dari service 2 = " + exchange.getBody());
10           assertEquals(HttpStatus.OK, exchange.getStatusCode());
11       }
12   }
```

Service 2: Consumer

<https://github.com/rikialdi/microservice-second.git>



```
1     @RestController
2     @RequestMapping("/consumer")
3     public class SecondController {
4
5         1 usage
6         @GetMapping("/message")
7         public String test() { return "Hello JavaInUse Called in Second Service"; }
8     }
```

Service 3: Apps

Branch : 121222-microservice

https://github.com/rikialdi/binar_batch_5/pull/new/121222-microservice

```

package com.binar5.apps.utils.microservice;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.http.*;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import static org.junit.Assert.assertEquals;

@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment =
    SpringBootTest.WebEnvironment.RANDOM_PORT)
public class CallMicroServiceOther {
    @Autowired
    private TestRestTemplate restTemplate;

    @Test
    public void callServiceEmployee() {
        HttpHeaders headers = new HttpHeaders();
        headers.set("Accept", "*/*");
        headers.set("Content-Type", "application/json");
        // String request ="";
        // request Body
        // HttpEntity<String> entity = new HttpEntity<String>(request,
        headers);
        // Long idSupplier = 1L;
        String url = "http://localhost:8083/employee/message";
        ResponseEntity<String> exchange = restTemplate.exchange(url, HttpMethod.GET, null, String.class);

        System.out.println("response =" + exchange.getBody());
        assertEquals(HttpStatus.OK, exchange.getStatusCode());
    }

    // @Test
    // public void eskternalProvinsi() throws IOException {
    //     HttpHeaders headers = new HttpHeaders();
    //     headers.set("Accept", "*/*");
    }
}

```

```

//           headers.set("Content-Type", "application/json");
//           String request ="";
//           // request Body
//           HttpEntity<String> entity = new
HttpEntity<String>(request, headers);
//           String url =
"https://dev.farizdotid.com/api/daerahindonesia/provinsi";
//           ResponseEntity<JsonNode> exchange =
restTemplate.exchange(url, HttpMethod.GET, null, JsonNode.class);
//
//
//           String data = String.valueOf(exchange.getBody());
//           ObjectMapper objectMapper = new ObjectMapper();
//
//           JsonNode jsonNode = objectMapper.readTree(data);
System.out.println("res1=" + jsonNode.get("provinsi"));
//
System.out.println("res2=" + exchange.getBody());
//
//
//           System.out.println("response =" + exchange.getBody());
//           assertEquals(HttpStatus.OK, exchange.getStatusCode());
}
//
//
// @Test
// public void insertSupplier() {
//     HttpHeaders headers = new HttpHeaders();
//     headers.set("Accept", "*/*");
//     headers.set("Content-Type", "application/json");
//     String bodyTesting = "{\n" +
//             "    \"nama\": \"supplier 2\", \n" +
//             "    \"hp\": \"098765462\", \n" +
//             "    \"alamat\": \"jl sudirman2\"\n" +
//             "}";
//     HttpEntity<String> entity = new
HttpEntity<String>(bodyTesting, headers);
//
//     ResponseEntity<String> exchange =
restTemplate.exchange("http://localhost:9091/api/v1/binar/supplier/save", HttpMethod.POST, entity, String.class);
//
//     assertEquals(HttpStatus.OK, exchange.getStatusCode());
//     System.out.println("response =" + exchange.getBody());
}
//
//
// @Test
// public void insertSuppliermap() {
//     HttpHeaders headers = new HttpHeaders();
//     headers.set("Accept", "*/*");
//     headers.set("Content-Type", "application/json");
//     Map bodyTesting = new HashMap();
//     bodyTesting.put("nama", "nama");
//     bodyTesting.put("hp", "445654654");
//     bodyTesting.put("alamat", "alamat53534");
//     HttpEntity<Map> entity = new HttpEntity<Map>(bodyTesting,
headers);
//
//     ResponseEntity<Map> exchange =
restTemplate.exchange("http://localhost:9091/api/v1/binar/supplier/sa

```

```

    ve", HttpMethod.POST, entity, Map.class);
    //
    //      assertEquals(HttpStatus.OK, exchange.getStatusCode());
    //      System.out.println("response  =" + exchange.getBody());
    //
    }
    //
    @Test
    public void listSupplier() {
    HttpHeaders headers = new HttpHeaders();
    //
    headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
    //
    MultiValueMap<Object, Object> map = new
    LinkedMultiValueMap<>();
    //
    map.add("page", "0");
    map.add("size", "10");
    //
    HttpEntity<MultiValueMap<Object, Object>> request = new
    HttpEntity<MultiValueMap<Object, Object>>(map, headers);
    //
    ResponseEntity<String> response =
    restTemplate.exchange("http://localhost:9091/api/v1/binar/supplier/li
    st", HttpMethod.GET, request, String.class);
    //
    //      assertEquals(HttpStatus.OK, response.getStatusCode());
    //      System.out.println("response  =" + response.getBody());
    //
    }
}

```

Output

Service 2 memanggil service 1 dan service 1 memanggil service 2. MicroService saling berkaitan satu sama lain.

Cara memanggil service rest api lain dengan class **RestTemplate**.

Silahkan running service 3 : apps

The screenshot displays two Java IDE interfaces side-by-side, illustrating a microservices architecture example.

Left IDE (IntelliJ IDEA):

- Project Structure:** Shows a project named "CallMicroServiceOther" containing packages like "testing", "utils", "microservice", and "validation.web".
- Code Editor:** The file "CallMicroServiceOther.java" is open, showing Java code that interacts with another service. A red box highlights the following code block:

```
Long idSupplier = 1L;
String url = "http://localhost:8083/employees/1";
ResponseEntity<String> exchange = restTemplate.exchange(url, HttpMethod.GET, null, String.class);
System.out.println("response = " + exchange.getBody());
assertThat(exchange.getStatusCode()).isEqualTo(HttpStatus.OK);
```
- Run Tab:** Shows a successful test run with a duration of 420 ms. The output window shows the printed response: "response =Hello JavaInUse Called in First Service and Called in Second Service".

Right IDE (Eclipse IDE):

- Project Explorer:** Shows a project named "FirstApplication" with packages "main" and "microservice".
- Code Editor:** The file "FirstController.java" is open, showing a REST controller. A red box highlights the following code block:

```
@GetMapping("/message")
public String test() {
    callServiceEmployee();
    return "Hello JavaInUse";
}
```
- Logs:** The Eclipse Log view shows multiple log entries from the application. A red box highlights the last three entries, which are identical to the output in the IntelliJ run tab:

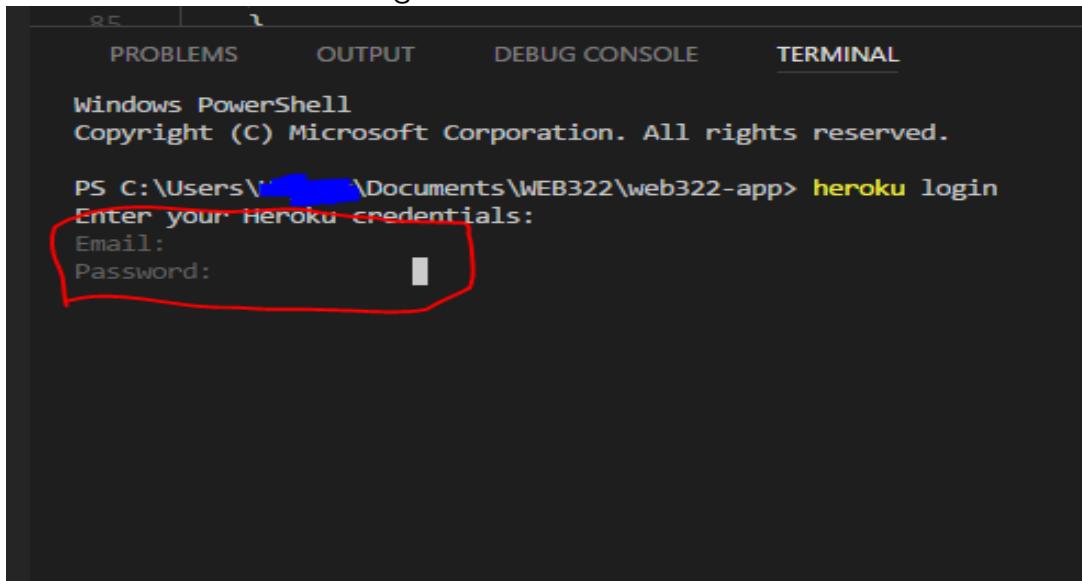
```
2022-12-12 20:19:51.478 INFO 19824 --- [nio-8083-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : response dari service 1 =Hello JavaInUse Called in Second Service
2022-12-12 20:19:51.485 INFO 19824 --- [nio-8083-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : response dari service 1 =Hello JavaInUse Called in Second Service
2022-12-12 20:19:51.485 INFO 19824 --- [nio-8083-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : response dari service 1 =Hello JavaInUse Called in Second Service
```

i. Branch

https://gitlab.com/rikialdi/batch6/-/merge_requests/new?merge_request%5Bsource_branch%5D=microservice-sample

34. Deploy Heroku Lewat CLI

1. DOWNLOAD dan Install Heroku cli
<https://devcenter.heroku.com/articles/heroku-cli#install-the-heroku-cli>
2. buka terminal project / buka heroku cli / terminal intelij juga bisa
3. ketik command \$ heroku login



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

PS C:\Users\[REDACTED]\Documents\WEB322\web322-app> heroku login
Enter your Heroku credentials:
Email: [REDACTED]
Password: [REDACTED]
```

4. Masukan email pw , heroku nya
5. \$ heroku git:clone -a (yourapp-api)
6. Setelah di clone, timpa file clone dengan file project terupdate atau yg ingin di deploy
Atau jika sudah ada project terupdate bisa di push ke heroku master
7. Kemudian ketikan command biasa seperti mau push ke github
8. \$ git add .
9. \$ git commit -am "make it better"
10. \$ git push heroku master

```

remote:      [INFO] Tests are skipped.
remote:      [INFO]
remote:      [INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ commit ---
remote:      [INFO] Building jar: /tmp/build_3619c7a5/target/commit-0.0.1-SNAPSHOT.jar
remote:      [INFO]
remote:      [INFO] --- spring-boot-maven-plugin:2.0.6.RELEASE:repackage (repackage) @ commit ---
remote:      [INFO]
remote:      [INFO] --- spring-boot-maven-plugin:2.0.6.RELEASE:repackage (default) @ commit ---
remote:      [INFO]
remote:      [INFO] --- maven-install-plugin:2.5.2:install (default-install) @ commit ---
remote:      [INFO] Installing /tmp/build_3619c7a5/target/commit-0.0.1-SNAPSHOT.jar to /tmp/codon/tmp/cache/.m2/repository/com/synrgy/commit/0.0.1-SNAPSHOT/commit-0.0.1-SNAPSHOT.jar
remote:      [INFO] Installing /tmp/build_3619c7a5/pom.xml to /tmp/codon/tmp/cache/.m2/repository/com/synrgy/commit/0.0.1-SNAPSHOT/commit-0.0.1-SNAPSHOT.pom
remote:      [INFO] -----
remote:      [INFO] BUILD SUCCESS
remote:      [INFO] -----
remote:      [INFO] Total time: 8.113 s
remote:      [INFO] Finished at: 2022-04-20T14:54:51Z
remote:      [INFO] -----
remote:      -----> Discovering process types
remote:      Procfile declares types     -> (none)
remote:      Default types for buildpack -> web
remote:      -----
remote:      -----> Compressing...
remote:      Done: 108M
remote:      -----> Launching...
remote:      Released v36
remote:      https://commitapps.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/commitapps.git
 bd53979..0f2a5d4  master -> master
Indras-MacBook-Pro:commit indrahasen$
```

11. TUNGGU HINGGA PROSES DEPLOY SELESAI

DAH SEKIAN TERIMAKASIH

35. Google Oauth Login – Register

a. Github

https://github.com/rikialdi/piksi_absensi_be.git

b. register untuk dapatkan id google

<https://console.cloud.google.com/apis/credentials>
pilih web .

Langkah-langkah

The screenshot shows the Google Cloud Platform interface. At the top, there's a navigation bar with 'Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)' and buttons for 'DISMISS' and 'ACTIVATE'. Below the navigation bar, there's a search bar with 'Search Products, resources, docs ()'. The main content area has a blue header 'Google Cloud' with a dropdown 'Select a project' and a 'CREATE PROJECT' button. Underneath, there's a sidebar with 'APIs & Services' and sections for 'Enabled APIs & services', 'Library', 'OAuth consent screen', and 'Page usage agreements'. The 'Credentials' section is highlighted with a red box. The main content area says 'To view this page, select a project.' and has a 'CREATE PROJECT' button in the bottom right corner.

Google Cloud Search Products, resources, docs (/)

New Project

You have 12 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name * My Project 70374

Project ID: triple-carrier-373007. It cannot be changed later. [EDIT](#)

Location * No organization [BROWSE](#)

Parent organization or folder

CREATE CANCEL

Create credential

Google Cloud My Project 70374 Search Products, resources, docs (/)

API APIs & Services Credentials [+ CREATE CREDENTIALS](#) [DELETE](#)

- Enabled APIs & services
- Library
- Credentials**
- OAuth consent screen
- Page usage agreements

Create credentials to access your enabled APIs. [Learn more](#)

API Keys

Name	Creation date	Restrictions	Actions
No API keys to display			

OAuth 2.0 Client IDs

Name	Creation date	Type	Client ID	Actions
No OAuth clients to display				

Service Accounts

Email	Name	Actions
No service accounts to display		

[Manage service accounts](#)

Google Cloud My Project 70374 Search Products, resources, docs (/)

API APIs & Services Credentials [+ CREATE CREDENTIALS](#) [DELETE](#)

- Enabled APIs & services
- Library
- Credentials**
- OAuth consent screen
- Page usage agreements

Create credentials to access your enabled APIs.

API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests user consent so your app can access the user's data

Service account
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

OAuth 2.0 Client IDs

Name	Creation date	Type	Actions
No OAuth clients to display			

The screenshot shows the Google Cloud API & Services dashboard for project 70374. The left sidebar has 'Enabled APIs & services' selected. The main area is titled 'Create OAuth client ID' and contains a note about client IDs and OAuth 2.0. A red box highlights the 'CONFIGURE CONSENT SCREEN' button.

This screenshot shows the 'OAuth consent screen' configuration page. The left sidebar has 'OAuth consent screen' selected. The main area has a heading 'OAuth consent screen'. It includes sections for 'User Type' (with 'Internal' selected), a note about external users, and 'External' as an option. A 'CREATE' button is at the bottom. A feedback link is at the bottom right.

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

User Type

Internal [?](#)

Because you're not a Google Workspace user, you can only make your app available to external (general audience) users.

External [?](#)

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. [Learn more about user type](#)

CREATE

[Let us know what you think](#) about our OAuth experience

≡ Google Cloud • My Project 70374 ▾

Search Products, resources, docs (/)

API APIs & Services	Edit app registration
Enabled APIs & services	App information This shows in the consent screen, and helps end users know who you are and contact you
Library	
Credentials	
OAuth consent screen	
Page usage agreements	

App name * apps_test
The name of the app asking for consent

User support email * rikitokoonline@gmail.com
For users to contact you with questions about their consent

App logo files181589141412022082 **BROWSE**
Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.



≡ Google Cloud • My Project 70374 ▾

Search Products, resources, docs (/)

API APIs & Services	Edit app registration
Enabled APIs & services	Provide users a link to your home page
Library	Application privacy policy link
Credentials	Provide users a link to your public privacy policy
OAuth consent screen	Application terms of service link
Page usage agreements	Provide users a link to your public terms of service

Authorized domains [?](#)
When a domain is used on the consent screen or in an OAuth client's configuration, it must be pre-registered here. If your app needs to go through verification, please go to the [Google Search Console](#) to check if your domains are authorized. [Learn more](#) about the authorized domain limit.

+ ADD DOMAIN

Developer contact information

Email addresses * rikitokoonline@gmail.com [X](#)

These email addresses are for Google to notify you about any changes to your project.

SAVE AND CONTINUE **CANCEL**

Google Cloud My Project 70374 Search Products, resources, docs (/)

API APIs & Services Create OAuth client ID

Enabled APIs & services A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information. [Learn more](#) about OAuth client types.

Library

Credentials Application type * Web application

OAuth consent screen

Page usage agreements

Android

Chrome app

iOS

TVs and Limited Input devices

Desktop app

Universal Windows Platform (UWP)

API APIs & Services Create OAuth client ID

Enabled APIs & services console and will not be shown to end users.

Library

Credentials The domains of the URIs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#).

OAuth consent screen

Page usage agreements

Authorized JavaScript origins ?

For use with requests from a browser

+ ADD URI

Authorized redirect URIs ?

For use with requests from a web server

+ ADD URI

Note: It may take 5 minutes to a few hours for settings to take effect

CREATE CANCEL

Redirect uri wajib

<http://localhost/Callback>

Google Cloud My Project 70374 Search Products, resources, docs (?)

API APIs & Services <ul style="list-style-type: none"> Enabled APIs & services Library Credentials OAuth consent screen Page usage agreements 	Client ID for Web application DOWNLOAD JSON RESET SECRET Authorized JavaScript origins <p>For use with requests from a browser</p> <p>+ ADD URI</p> Authorized redirect URIs <p>For use with requests from a web server</p> <p>URIs 1 * <input type="text" value="http://localhost/Callback"/></p> <p>+ ADD URI</p>
--	--

Note: It may take 5 minutes to a few hours for settings to take effect

SAVE **CANCEL**

OAuth client created

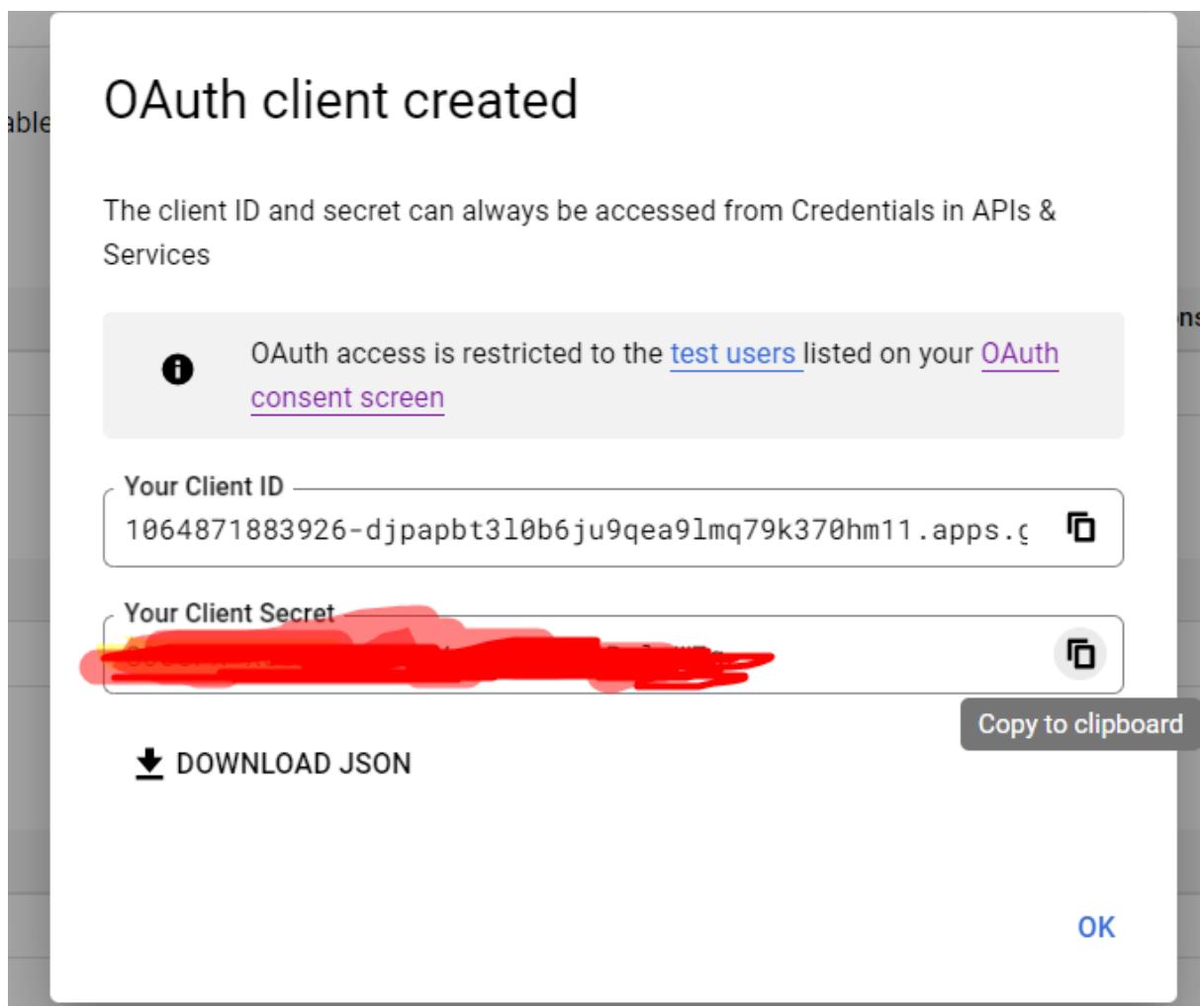
Atau allow redirect URL kamu misal

API APIs & Services <ul style="list-style-type: none"> Enabled APIs & services Library Credentials OAuth consent screen Page usage agreements 	Client ID for Web application DELETE Authorized JavaScript origins <p>For use with requests from a browser</p> <p>URIs 1 * <input type="text" value="http://localhost:3000"/></p> <p>URIs 2 * <input type="text" value="https://tamara.dev.idstar.co.id"/></p> <p>+ ADD URI</p> Authorized redirect URIs <p>For use with requests from a web server</p> <p>URIs 1 * <input type="text" value="http://localhost/Callback"/></p> <p>URIs 2 * <input type="text" value="http://localhost:3000"/></p> <p>URIs 3 * <input type="text" value="https://tamara.dev.idstar.co.id"/></p> <p>+ ADD URI</p>
--	--

Note: It may take 5 minutes to a few hours for settings to take effect

SAVE **CANCEL**

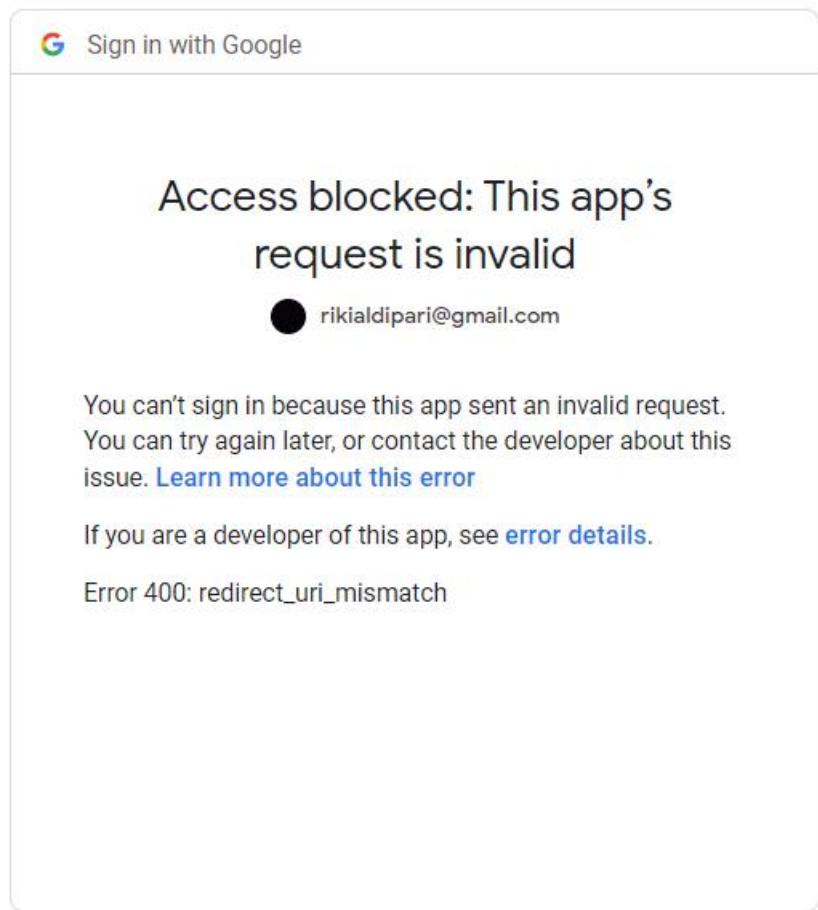
OAuth



This screenshot shows the "Credentials" section of the Google Cloud Platform console. It includes sections for "API Keys" (empty) and "OAuth 2.0 Client IDs". The "OAuth 2.0 Client IDs" table has columns for Name, Creation date, Type, Client ID, and Actions. One entry, "Web client 1", is highlighted with a red border. At the bottom, there is a "Service Accounts" section and a link to "Manage service accounts".

Name	Creation date	Type	Client ID	Actions
Web client 1	Dec 28, 2022	Web application	1064871883926-djp... [copy]	[edit] [trash] [more]

Jika eror seperti dibawah ini: maka tambahin Callback



English (United States) ▾

Help

Privacy

Terms

c. Client id dan token example

```
{  
  "web": {  
    "client_id": "1064871883926-  
djpapbt3l0b6ju9qea9lmq79k370hm11.apps.googleusercontent.com",  
    "project_id": "triple-carrier-373007",  
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",  
    "token_uri": "https://oauth2.googleapis.com/token",  
    "auth_provider_x509_cert_url":  
      "https://www.googleapis.com/oauth2/v1/certs",  
      "client_secret": "GOCS...X8b2tSXmnlsYvAnU02RSjBalzWZq"  
  }  
}
```

d. Pom.xml

```
<!-- google-->
<dependency>
    <groupId>com.google.apis</groupId>
    <artifactId>google-api-services-oauth2</artifactId>
    <version>v2-rev65-1.17.0-rc</version>
</dependency>
<dependency>
    <groupId>com.google.api-client</groupId>
    <artifactId>google-api-client-jackson2</artifactId>
    <version>1.20.0</version>
</dependency>
<dependency>
    <groupId>com.google.http-client</groupId>
    <artifactId>google-http-client-jackson2</artifactId>
    <version>1.42.2</version>
</dependency>
<dependency>
    <groupId>com.google.oauth-client</groupId>
    <artifactId>google-oauth-client-jetty</artifactId>
    <version>1.34.1</version>
</dependency>
<!-- tymleaf-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

e. Resources id google

```
{
  "installed": {
    "client_id": "985707925670-
plhu4csce96ampd314mmvemef3k51225.apps.googleusercontent.com",
    "client_secret": "GOCSPLX-BDzNaTCOanFRt8hizGRmICptcHxo"
  }
}
```

The screenshot shows the project structure on the left and a code editor on the right. The project structure includes a 'resources' folder containing 'pdf-templates' (with files quotation.html, register.html, and reportingts.html) and 'templates' (with various dashboard-related files). Other files like application.properties, application-dev.properties, and application-prod.properties are also listed. A file named 'client_secrets.json' is highlighted with a red box in the list.

```

1  {
2      "installed": {
3          "client_id": "985707925676",
4          "client_secret": "GOCSpx-E
5      }
6  }

```

f. Resources -register timeleaf

The screenshot shows the project structure on the left and a code editor on the right. The project structure includes a 'resources' folder containing 'pdf-templates' (with files quotation.html, register.html, and reportingts.html). A file named 'register.html' is highlighted with a red box in the list.

```

1  <!DOCTYPE HTML>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <meta charset="UTF-8" />
5          <title th:utext="${title}" />
6          <link rel="stylesheet" type="text/css" th:href="@{/css/style.css}"/>
7      </head>
8      <body>
9          <h1 th:utext="${title}" />
10         <h1 th:utext="${erordesc}" />
11         <br/><br/>
12         <div>Copyright 2022</div>
13     </body>
14 </html>

```

```

<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8" />
    <title th:utext="${title}" />
    <link rel="stylesheet" type="text/css"
th:href="@{/css/style.css}"/>
</head>
<body>

```

```

<h1 th:utext="${title}" />
<h1 th:utext="${erordesc}" />
<br/><br/>
<div>Copyright 2022</div>
</body>
</html>

```

g. Daftarkan uri di google

The screenshot shows the Google Cloud Platform API Keys page. On the left, there's a sidebar with navigation links: 'Enabled APIs & services', 'Library', 'Credentials' (which is selected), 'OAuth consent screen', 'Main verification', and 'Usage agreements'. The main content area has a heading 'Create credentials to access your enabled APIs. [Learn more](#)'. Below it is a section titled 'API Keys' with a table header: 'Name' (sorted by creation date), 'Creation date', 'Restrictions', and 'Actions'. A note says 'No API keys to display'. Another section titled 'OAuth 2.0 Client IDs' follows, with a similar table structure. It shows one entry: 'chute-web' (Creation date: Jan 19, 2022, Type: Web application, Client ID: 985707925670-plhu...). The 'Actions' column for this entry has a red box around the edit icon. Below these sections is a 'Service Accounts' section with a table header: 'Email' (sorted by name), 'Name', and 'Actions'. A note says 'No service accounts to display'. At the bottom right of this section is a link 'Manage service accounts'.

Authorized redirect URIs ?

For use with requests from a web server

URIs 1 *

<http://localhost/Callback>

URIs 2 *

<https://fir-project-1d418.firebaseio.com>

URIs 3 *

https://fir-project-1d418.firebaseio.com/_/auth/handler

+ ADD URI

Note: It may take 5 minutes to a few hours for settings to take effect.

h. Config Security

Jika belum ada , abaikan, guna untuk follow akses tampa token

```

src
└── main
    └── java
        └── com.synrgy.commit
            ├── config
            │   ├── Config
            │   ├── OAuth2AccessTokenConverter
            │   └── OAuth2AuthorizationServerConfiguration
            └── controller
                ├── dashboard
                └── fileupload

```

30 .csrf() CsrfConfigurer<HttpSecurity>
 31 .disable() HttpSecurity
 32 .antMatcher(antPattern: "/**")
 33 .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.Expression
 34 .antMatchers(...antPatterns: "/" "/showFile/**", "/v1/showFile/
 35 "/register", "/register/**", "/Login/", "/dashbo
 36 "/loginUser/*") ExpressionUrlAuthorizationConfigurer<...>.Ant
 37 .permitAll() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrl
 38 .antMatchers(...antPatterns: "/v1/role-test-global/list-barang
 39 .antMatchers(...antPatterns: "/v1/role-test-global/post-barang
 40 .antMatchers(...antPatterns: "/v1/role-test-global/post-barang
 41 .antMatchers(...antPatterns: "/v1/role-test-global/post-barang
 42 .and() HttpSecurity
 43 .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.Expression
 44 .anyRequest() ExpressionUrlAuthorizationConfigurer<...>.AuthorizedUrl
 45 .authenticated() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterce

i. Login controller : Pending

Logic : Login and Register, if user not found, maka do register

```

package com.synrgy.commit.controller;

import com.google.api.client.googleapis.auth.oauth2.GoogleCredential;
import com.google.api.client.googleapis.json.GoogleJsonResponseException;
import com.google.api.client.http.javanet.NetHttpTransport;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.services.oauth2.Oauth2;
import com.google.api.services.oauth2.model.Userinfoplus;
import com.synrgy.commit.config.Config;
import com.synrgy.commit.dao.request.LoginModel;
import com.synrgy.commit.dao.request.Register;
import com.synrgy.commit.dao.request.RegisterModel;
import com.synrgy.commit.dao.request.UserUpdateModel;
import com.synrgy.commit.model.oauth.User;
import com.synrgy.commit.repository.oauth.UserRepository;
import com.synrgy.commit.service.EmailSender;
import com.synrgy.commit.service.UserService;
import com.synrgy.commit.util.EmailTemplate;
import com.synrgy.commit.util.Response;
import com.synrgy.commit.util.SimpleStringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.web.client.RestTemplateBuilder;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.HttpMethod;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.util.MultiValueMap;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;

import java.io.IOException;

```

```

import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import javax.validation.ConstraintViolationException;
import javax.validation.Valid;
import java.util.Map;

@RestController
@RequestMapping("/login-user")
public class LoginController {
    @Autowired
    private UserRepository userRepository;

    Config config = new Config();

    @Autowired
    public UserService serviceReq;

    @Value("${expired.token.password.minute:}")//FILE_SHOW_RUL
    private int expiredToken;

    @Autowired
    public Response response;

    @Value("${BASEURL:}")//FILE_SHOW_RUL
    private String BASEURL;

    @Autowired
    private PasswordEncoder passwordEncoder;

    private static final Logger logger =
LoggerFactory.getLogger(LoginController.class);

    @Value("${AUTHURL:}")//FILE_SHOW_RUL
    private String AUTHURL;

    @Autowired
    public RegisterController registerController;

    @Autowired
    private RestTemplateBuilder restTemplateBuilder;

    @Autowired
    public EmailTemplate emailTemplate;

    @Autowired
    public EmailSender emailSender;

    @Value("${APPNAME:}")//FILE_SHOW_RUL
    private String APPNAME;

    @PostMapping("")
    @ExceptionHandler(ConstraintViolationException.class)
    public ResponseEntity<Map> login(@Valid @RequestBody LoginModel
objModel) {
        Map map = serviceReq.login(objModel);
    }
}

```

```

        return new ResponseEntity<Map>(map, HttpStatus.OK);
    }

    @PostMapping("/signin_google")
    @ResponseBody
    public ResponseEntity<Map> repairGoogleSigninAction(@RequestParam
MultiValueMap<String, String> parameters) throws IOException {

        Map<String, Object> map123 = new HashMap<>();
        Map<String, String> map = parameters.toSingleValueMap();
        String accessToken = map.get("accessToken");

        GoogleCredential credential = new
        GoogleCredential().setAccessToken(accessToken);
        System.out.println("access_token user=" + accessToken);
        OAuth2 oauth2 = new OAuth2.Builder(new NetHttpTransport(),
        new JacksonFactory(), credential).setApplicationName(
            "Oauth2").build();
        Userinfoplus profile= null;
        try {
            profile = oauth2 userinfo().get().execute();
        } catch (GoogleJsonResponseException e)
        {
            return new
            ResponseEntity<Map>(response.Error(e.getDetails()),
            HttpStatus.BAD_GATEWAY);
        }
        profile.toPrettyString();
        User user =
        userRepository.findOneByUsername(profile.getEmail());
        if (null != user) {
            if(!user.isEnabled()){
                UserUpdateModel obk = new UserUpdateModel();
                obk.setEmail(user.getUsername());
                sendEmailregister(obk);
                map123.put(config.getCode(), "401");
                map123.put(config.getMessage(), "Your Account is
disabled. Please check your email for activation.");
                map123.put("type", "register");
                System.out.println("masuk 2");
                return new ResponseEntity<Map>(map123,
                HttpStatus.OK);
            }
            for (Map.Entry<String, String> req : map.entrySet()) {
                logger.info(req.getKey());
                logger.info(req.getValue());
            }

            Register register = new Register();
            register.setEmail(profile.getEmail());
            register.setPassword(profile.getId());
            register.setName(profile.getName());

            String oldPassword = user.getPassword();
            Boolean isPasswordMatches = true;
            if (!passwordEncoder.matches(register.getPassword(),
            oldPassword)) {
                userRepository.updatePassword(user.getId(),

```



```

        System.out.println("masuk 1 luar ");
        return new ResponseEntity<Map>(map123, HttpStatus.OK);
    }

    // Step 2: sendp OTP berupa URL: guna updata enable agar bisa
    login:
    @PostMapping("send-otp")//send OTP
    public Map sendEmailegister(@RequestBody UserUpdateModel user) {
        String message = "Thanks, please check your email for
activation.';

        if (user.getEmail() == null) return response.isRequired("No
email provided");
        User found =
userRepository.findOneByUsername(user.getEmail());
        if (found == null) return response.notFound("Email not
found"); //throw new BadRequest("Email not found");

        String template = emailTemplate.getTalentAcc();
        if (StringUtils.isEmpty(found.getOtp())) {
            User search;
            String otp;
            do {
                otp = SimpleStringUtils.randomString(6, true);
                search = userRepository.findOneByOTP(otp);
            } while (search != null);
            Date dateNow = new Date();
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(dateNow);
            calendar.add(Calendar.MINUTE, expiredToken);
            Date expirationDate = calendar.getTime();

            found.setOtp(otp);
            found.setOtpExpiredDate(expirationDate);
            template = template.replaceAll("\\{\\{USERNAME}\\}",

(found.getUsername()));
            template = template.replaceAll("\\{\\{VERIF_LINK}\\}",

BASEURL + "register/web/index/" + otp);
            userRepository.save(found);
        } else {
            template = template.replaceAll("\\{\\{USERNAME}\\}",

(found.getUsername()));
            template = template.replaceAll("\\{\\{VERIF_LINK}\\}",

BASEURL + "register/web/index/" + found.getOtp());
        }
        emailSender.sendAsync(found.getUsername(), APPNAME + "-

Register", template);
        return response.Sukses(message);
    }

}

```

If hanya Login saja

```
@PostMapping("/signin_google")
    @ResponseBody
//    public ResponseEntity<Map>
repairGoogleSigninAction(@RequestParam MultiValueMap<String, String>
parameters) throws IOException {
    public ResponseEntity<Map> repairGoogleSigninAction(@RequestBody
GoogleReq parameters) throws IOException {

        Map<String, Object> map123 = new HashMap<>();
        Map<String, String> map = parameters.toSingleValueMap();
        if(StringUtils.isEmpty(parameters.getAccessToken())){
            return new
ResponseEntity<Map>(response.templateError("Token is required."),
HttpStatus.OK);
        }
        String accessToken = parameters.getAccessToken();

        // step 1 : set toke : google akan memvalidasi token yang
kita kirim
        GoogleCredential credential = new
GoogleCredential().setAccessToken(accessToken);
        System.out.println("access_token user=" + accessToken);
        // step 2 : get informasi akaun dikonversi bentuk objek
        OAuth2 oauth2 = new OAuth2.Builder(new NetHttpTransport(),
new JacksonFactory(), credential).setApplicationName(
                "OAuth2").build();
        // step 3 : oauth2 akan diolah oleh Userinfoplus : goodle
(DTO)
        Userinfoplus profile = null;
        try {
            // get information dari token si Google
            profile = oauth2.userInfo().get().execute();
        } catch (GoogleJsonResponseException e) {
            return new
ResponseEntity<Map>(response.Error(e.getDetails()),
HttpStatus.BAD_GATEWAY);
        }
        profile.toPrettyString();
        // step 4 : kita hanya email, full name dari DTO
Userinfoplus dan kita chek ke db, untuk validasi
        User user =
userRepository.findOneByUsername(profile.getEmail());
        if (null != user) {
            if (!user.isEnabled()) {
                return new ResponseEntity<Map>(response.Error("Your
Account is disable. Please check your email for activation."),
HttpStatus.OK);
            }
            for (Map.Entry<String, String> req : map.entrySet()) {
                logger.info(req.getKey());
                logger.info(req.getValue());
            }

            String oldPassword = user.getPassword();
            System.out.println("password lama :" +
user.getPassword());
        }
    }
}
```

```

//          Boolean isPasswordMatches = true;
String pass = "Password123";
if (!passwordEncoder.matches(pass, oldPassword)) {
    userRepository.updatePassword(user.getId(),
passwordEncoder.encode(register.getPassword()));
    System.out.println("update password berhasil");
    user.setPassword(passwordEncoder.encode(pass));
    userRepository.save(user);
}
//step 5 : login seperti biasa
String url = AUTHURL + "?username=" + profile.getEmail()
+
        "&password=" + pass +
        "&password=" + "password" +
        "&grant_type=password" +
        "&client_id=my-client-web" +
        "&client_secret=password";
ResponseEntity<Map> response123 =
restTemplateBuilder.build().exchange(url, HttpMethod.POST, null, new
ParameterizedTypeReference<Map>() {
});

if (response123.getStatusCode() == HttpStatus.OK) {
    userRepository.save(user);

    map123.put("access_token",
response123.getBody().get("access_token"));
    map123.put("token_type",
response123.getBody().get("token_type"));
    map123.put("refresh_token",
response123.getBody().get("refresh_token"));
    map123.put("expires_in",
response123.getBody().get("expires_in"));
    map123.put("scope",
response123.getBody().get("scope"));
    map123.put("jti", response123.getBody().get("jti"));
    map123.put("status", 200);
    map123.put("message", "Success");
    map123.put("type", "login");

    user.setPassword(oldPassword);
    User datUser = userRepository.save(user);
    map123.put("user", datUser);
    // last step : untuk response
    return new
ResponseEntity<Map>(response.sukses(map123), HttpStatus.OK);

}
} else {
//      register : tidak digunakan
    return new ResponseEntity<Map>(response.Error("Username
is not registered yet. Please contact admin."), HttpStatus.OK);
}
return new ResponseEntity<Map>(response.sukses(map123),
HttpStatus.OK);
}

```

j. Register controller

```
package com.synrgy.commit.controller;

import com.synrgy.commit.config.Config;
import com.synrgy.commit.dao.request.RegisterModel;
import com.synrgy.commit.model.oauth.User;
import com.synrgy.commit.repository.oauth.UserRepository;
import com.synrgy.commit.service.UserService;
import com.synrgy.commit.util.Response;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.validation.ConstraintViolationException;
import javax.validation.Valid;
import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/register")
public class RegisterController {
    @Autowired
    private UserRepository userRepository;

    Config config = new Config();

    @Autowired
    public UserService serviceReq;

    @Value("${BASEURL:}") //FILE_SHOW_RUL
    private String BASEURL;

    @Autowired
    public Response response;
    @PostMapping("")
    @ExceptionHandler(ConstraintViolationException.class)
    public ResponseEntity<Map> saveRegisterManual(@Valid @RequestBody
RegisterModel objModel) throws RuntimeException {
        Map map = new HashMap();

        User user =
userRepository.checkExistingEmail(objModel.getEmail());
        if (null != user) {
            return new ResponseEntity<Map>(response.Error("Email is
Registered, try another email or click Forget Password"),
HttpStatus.OK);
        }
        map = serviceReq.registerManual(objModel);

        return new ResponseEntity<Map>(map, HttpStatus.OK);
    }

}
```

k. Dao request

```
package com.synrgy.commit.dao.request;

import
com.synrgy.commit.controller.validationpass.anotation.PasswordValueMa
tch;
import
com.synrgy.commit.controller.validationpass.anotation.ValidPassword;
import lombok.Data;
import org.aspectj.bridge.Message;

import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.Size;

@Data
public class RegisterModel {
    public Long id;

    public String email;

    @ValidPassword
    @NotEmpty(message = "password is mandatory")
    public String password;

    public String name;

    public String gender;

    public String domicile;

    public String phone_number;

    public String interest;
}
```

I. Testing Login with Google

```
package com.synrgy.commit.google;

import com.google.api.client.auth.oauth2.Credential;
import
com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeI
nstalledApp;
import
com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceive
r;
```

```

import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport;
import com.google.api.client.http.HttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.client.util.store.DataStoreFactory;
import com.google.api.client.util.store.FileDataStoreFactory;
import com.google.api.services.oauth2.Oauth2;
import com.google.api.services.oauth2.model.Tokeninfo;
import com.google.api.services.oauth2.model.Userinfoplus;

import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.List;

/**
 * Command-line sample for the Google OAuth2 API described at <a href="http://code.google.com/apis/accounts/docs/OAuth2Login.html">Using OAuth 2.0 for Login</a>.
 * (Experimental)</a>.
 * https://github.com/google/google-api-java-client-samples/blob/master/oauth2-cmdline-sample/src/main/java/com/google/api/services/samples/oauth2/cmdline/OAuth2Sample.java
 * @author Yaniv Inbar
 */
public class OAuth2Sample {

    /**
     * Be sure to specify the name of your application. If the application name is {@code null} or
     * blank, the application will log a warning. Suggested format is "MyCompany-ProductName/1.0".
     */
    private static final String APPLICATION_NAME = "";

    /** Directory to store user credentials. */
    private static final java.io.File DATA_STORE_DIR =
        new java.io.File(System.getProperty("user.home"),
".store/oauth2_sample");

    /**
     * Global instance of the {@link DataStoreFactory}. The best practice is to make it a single
     * globally shared instance across your application.
     */
    private static FileDataStoreFactory dataStoreFactory;

    /** Global instance of the HTTP transport. */
    private static HttpTransport httpTransport;

    /** Global instance of the JSON factory. */
}

```

```

private static final JsonFactory JSON_FACTORY =
JacksonFactory.getDefaultInstance();

/** OAuth 2.0 scopes. */
private static final List<String> SCOPES = Arrays.asList(
    "https://www.googleapis.com/auth/userinfo.profile",
    "https://www.googleapis.com/auth/userinfo.email");

private static Oauth2 oauth2;
private static GoogleClientSecrets clientSecrets;

/** Authorizes the installed application to access user's
protected data. */
private static Credential authorize() throws Exception {
    // load client secrets
    clientSecrets = GoogleClientSecrets.load(JSON_FACTORY,
        new
InputStreamReader(OAuth2Sample.class.getResourceAsStream("/client_secrets.json")));
    if
(clientSecrets.getDetails().getClientId().startsWith("Enter")
    ||
clientSecrets.getDetails().getClientSecret().startsWith("Enter ")) {
        System.out.println("Enter Client ID and Secret from
https://code.google.com/apis/console/"
            + "into oauth2-cmdline-
sample/src/main/resources/client_secrets.json");
        System.exit(1);
    }
    // set up authorization code flow
    GoogleAuthorizationCodeFlow flow = new
GoogleAuthorizationCodeFlow.Builder(
        httpTransport, JSON_FACTORY, clientSecrets,
SCOPES).setDataStoreFactory(
        dataStoreFactory).build();
    // authorize
    return new AuthorizationCodeInstalledApp(flow, new
LocalServerReceiver()).authorize("user");
}

public static void main(String[] args) {
    try {
        httpTransport =
GoogleNetHttpTransport.newTrustedTransport();
        dataStoreFactory = new
FileDataStoreFactory(DATA_STORE_DIR);
        // authorization
        Credential credential = authorize();
        // set up global Oauth2 instance
        oauth2 = new Oauth2.Builder(httpTransport, JSON_FACTORY,
credential).setApplicationName(
            APPLICATION_NAME).build();
        System.out.println("token saya =
"+credential.getAccessToken());
        // run commands
        tokenInfo(credential.getAccessToken());
        userInfo();
        // success!
        return;
    }
}

```

```

        } catch (IOException e) {
            System.err.println(e.getMessage());
        } catch (Throwable t) {
            t.printStackTrace();
        }
        System.exit(1);
    }

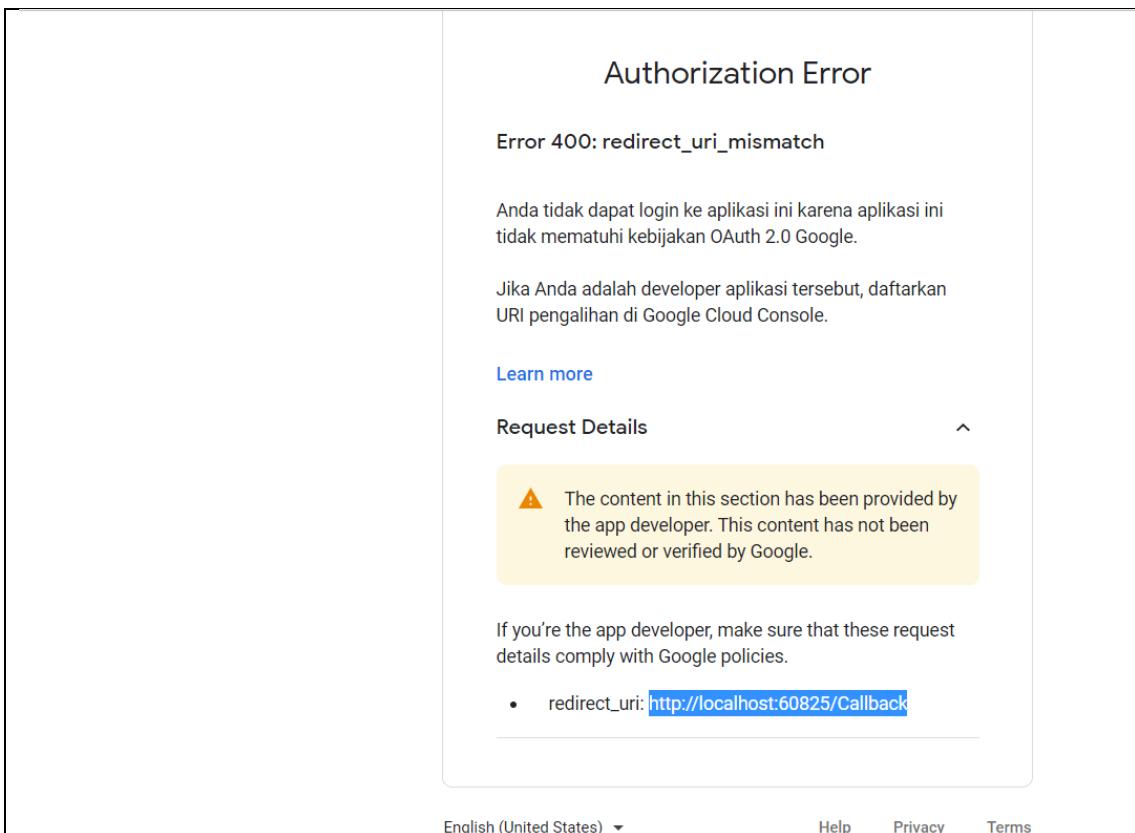
    private static void tokenInfo(String accessToken) throws
IOException {
    header("Validating a token");
    Tokeninfo tokeninfo =
oauth2.tokeninfo().setAccessToken(accessToken).execute();
    System.out.println(tokeninfo.toPrettyString());
    if
(!tokeninfo.getAudience().equals(clientSecrets.getDetails().getClient
Id())) {
        System.err.println("ERROR: audience does not match our
client ID!");
    }
}

private static void userInfo() throws IOException {
    header("Obtaining User Profile Information");
    Userinfoplus userinfo = oauth2 userinfo().get().execute();
    System.out.println(userinfo.toPrettyString());
}

static void header(String name) {
    System.out.println();
    System.out.println("===== " + name + "
=====");
    System.out.println();
}
}

```

lakukan running di postman
maka akan tampil



Akan dapat get token: token di copy paste

```
C:\Users\USER\.jdks\corretto-11.0.15\bin\java.exe ...
Aug 24, 2022 11:45:16 AM com.google.api.client.googleapis.services.AbstractGoogleClient <init>
WARNING: Application name is not set. Call Builder#setApplicationName.
token saya = ya29.a0AVAv9yiuZ-KtnZzFdRpPiINhSFZCpN8d4EMViv7dERss-bRn7DbBpIhfLgPZwW0kaTcbBCIZFuIWjuniCg5ida6CpXgj0d7SeQyAiUhVtRagrzG
=====
===== Validating a token =====
{
    "access_type" : "online",
    "audience" : "985707925670-plhu4csce96ampd3l4mmvemef3k5l225.apps.googleusercontent.com",
    "email" : "rikialdipari@gmail.com",
    "expires_in" : 953,
```

Berikut informasi output dari token tersebut

```
C:\Users\USER\.jdks\openjdk-18.0.2.1\bin\java.exe -javaagent:C:\Program
Files\JetBrains\IntelliJ IDEA Community Edition
2022.1.3\lib\idea_rt.jar=51166:C:\Program Files\JetBrains\IntelliJ IDEA
Community Edition 2022.1.3\bin" -Dfile.encoding=UTF-8 -classpath
"E:\binar\Binat
6\MODUL\karyawan\target\classes;C:\Users\USER\.m2\repository\org\spring
framework\boot\spring-boot-starter-web\2.7.14\spring-boot-starter-web-
2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-
boot-starter\2.7.14\spring-boot-starter-
2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-
```

boot-starter-logging\2.7.14\spring-boot-starter-logging-
2.7.14.jar;C:\Users\USER\.m2\repository\ch\qos\logback\logback-
classic\1.2.12\logback-classic-
1.2.12.jar;C:\Users\USER\.m2\repository\ch\qos\logback\logback-
core\1.2.12\logback-core-
1.2.12.jar;C:\Users\USER\.m2\repository\org\apache\logging\log4j\log4j-to-
slf4j\2.17.2\log4j-to-slf4j-
2.17.2.jar;C:\Users\USER\.m2\repository\org\apache\logging\log4j\log4j-
api\2.17.2\log4j-api-2.17.2.jar;C:\Users\USER\.m2\repository\org\slf4j\jul-to-
slf4j\1.7.36\jul-to-slf4j-
1.7.36.jar;C:\Users\USER\.m2\repository\jakarta\annotation\jakarta.annotation
-api\1.3.5\jakarta.annotation-api-
1.3.5.jar;C:\Users\USER\.m2\repository\org\yaml\snakeyaml\1.30\snakeyaml-
1.30.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-
boot-starter-json\2.7.14\spring-boot-starter-json-
2.7.14.jar;C:\Users\USER\.m2\repository\com\fasterxml\jackson\datatype\jack
son-datatype-jdk8\2.13.5\jackson-datatype-jdk8-
2.13.5.jar;C:\Users\USER\.m2\repository\com\fasterxml\jackson\datatype\jac
kson-datatype-jsr310\2.13.5\jackson-datatype-jsr310-
2.13.5.jar;C:\Users\USER\.m2\repository\com\fasterxml\jackson\module\jacks
on-module-parameter-names\2.13.5\jackson-module-parameter-names-
2.13.5.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-
boot-starter-tomcat\2.7.14\spring-boot-starter-tomcat-
2.7.14.jar;C:\Users\USER\.m2\repository\org\apache\tomcat\embed\tomcat-
embed-core\9.0.78\tomcat-embed-core-
9.0.78.jar;C:\Users\USER\.m2\repository\org\apache\tomcat\embed\tomcat-
embed-el\9.0.78\tomcat-embed-el-
9.0.78.jar;C:\Users\USER\.m2\repository\org\apache\tomcat\embed\tomcat-
embed-websocket\9.0.78\tomcat-embed-websocket-
9.0.78.jar;C:\Users\USER\.m2\repository\org\springframework\spring-
web\5.3.29\spring-web-
5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\spring-
webmvc\5.3.29\spring-webmvc-
5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\spring-
expression\5.3.29\spring-expression-
5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-
boot-starter-thymeleaf\3.0.4\spring-boot-starter-thymeleaf-

3.0.4.jar;C:\Users\USER\.m2\repository\org\thymeleaf\thymeleaf-spring6\3.1.1.RELEASE\thymeleaf-spring6-3.1.1.RELEASE.jar;C:\Users\USER\.m2\repository\org\thymeleaf\thymeleaf\3.0.15.RELEASE\thymeleaf-3.0.15.RELEASE.jar;C:\Users\USER\.m2\repository\org\attoparser\attoparser\2.0.5.RELEASE\attoparser-2.0.5.RELEASE.jar;C:\Users\USER\.m2\repository\org\unbescape\unbescape\1.1.6.RELEASE\unbescape-1.1.6.RELEASE.jar;C:\Users\USER\.m2\repository\org\projectlombok\lombok\1.18.28\lombok-1.18.28.jar;C:\Users\USER\.m2\repository\jakarta\xml\bind\jakarta.xml.bind-api\2.3.3\jakarta.xml.bind-api-2.3.3.jar;C:\Users\USER\.m2\repository\jakarta\activation\jakarta.activation-api\1.2.2\jakarta.activation-api-1.2.2.jar;C:\Users\USER\.m2\repository\org\hamcrest\hamcrest\2.2\hamcrest-2.2.jar;C:\Users\USER\.m2\repository\net\bytebuddy\byte-buddy\1.12.23\byte-buddy-1.12.23.jar;C:\Users\USER\.m2\repository\org\springframework\spring-core\5.3.29\spring-core-5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\spring-jcl\5.3.29\spring-jcl-5.3.29.jar;C:\Users\USER\.m2\repository\junit\junit\4.13.2\junit-4.13.2.jar;C:\Users\USER\.m2\repository\org\hamcrest\hamcrest-core\2.2\hamcrest-core-2.2.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-test\2.7.14\spring-boot-test-2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot\2.7.14\spring-boot-2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\spring-test\5.3.29\spring-test-5.3.29.jar;C:\Users\USER\.m2\repository\org\postgresql\postgresql\42.3.8\postgresql-42.3.8.jar;C:\Users\USER\.m2\repository\org\checkerframework\checker-qual\3.5.0\checker-qual-3.5.0.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\2.7.14\spring-boot-starter-data-jpa-2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-

boot-starter-aop\2.7.14\spring-boot-starter-aop-
2.7.14.jar;C:\Users\USER\.m2\repository\org\aspectj\aspectjweaver\1.9.7\aspe-
ctjweaver-
1.9.7.jar;C:\Users\USER\.m2\repository\jakarta\transaction\jakarta.transaction-
api\1.3.3\jakarta.transaction-api-
1.3.3.jar;C:\Users\USER\.m2\repository\jakarta\persistence\jakarta.persistence-
api\2.2.3\jakarta.persistence-api-
2.2.3.jar;C:\Users\USER\.m2\repository\org\hibernate\hibernate-
core\5.6.15.Final\hibernate-core-
5.6.15.Final.jar;C:\Users\USER\.m2\repository\antlr\antlr\2.7.7\antlr-
2.7.7.jar;C:\Users\USER\.m2\repository\org\jboss\jandex\2.4.2.Final\jandex-
2.4.2.Final.jar;C:\Users\USER\.m2\repository\org\hibernate\common\hibernat-
e-commons-annotations\5.1.2.Final\hibernate-commons-annotations-
5.1.2.Final.jar;C:\Users\USER\.m2\repository\org\glassfish\jaxb\jaxb-
runtime\2.3.8\jaxb-runtime-
2.3.8.jar;C:\Users\USER\.m2\repository\org\glassfish\jaxb\txw2\2.3.8\txw2-
2.3.8.jar;C:\Users\USER\.m2\repository\com\sun\istack\istack-commons-
runtime\3.0.12\istack-commons-runtime-
3.0.12.jar;C:\Users\USER\.m2\repository\org\springframework\data\spring-
data-jpa\2.7.14\spring-data-jpa-
2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\data\spring-
data-commons\2.7.14\spring-data-commons-
2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\spring-
orm\5.3.29\spring-orm-
5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\spring-
aspects\5.3.29\spring-aspects-
5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-
boot-starter-jdbc\2.7.14\spring-boot-starter-jdbc-
2.7.14.jar;C:\Users\USER\.m2\repository\com\zaxxer\HikariCP\4.0.3\HikariCP-
4.0.3.jar;C:\Users\USER\.m2\repository\org\hibernate\validator\hibernate-
validator\6.0.13.Final\hibernate-validator-
6.0.13.Final.jar;C:\Users\USER\.m2\repository\javax\validation\validation-
api\2.0.1.Final\validation-api-
2.0.1.Final.jar;C:\Users\USER\.m2\repository\org\jboss\logging\jboss-
logging\3.4.3.Final\jboss-logging-
3.4.3.Final.jar;C:\Users\USER\.m2\repository\com\fasterxml\classmate\1.5.1\cl-
assmate-

1.5.1.jar;C:\Users\USER\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.15.2\jackson-databind-
2.15.2.jar;C:\Users\USER\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.13.5\jackson-annotations-
2.13.5.jar;C:\Users\USER\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.13.5\jackson-core-
2.13.5.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-starter-webflux\2.7.14\spring-boot-starter-webflux-
2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-starter-reactor-netty\2.7.14\spring-boot-starter-reactor-netty-
2.7.14.jar;C:\Users\USER\.m2\repository\io\projectreactor\netty\reactor-netty-http\1.0.34\reactor-netty-http-
1.0.34.jar;C:\Users\USER\.m2\repository\io\netty\netty-codec-http\4.1.94.Final\netty-codec-http-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-common\4.1.94.Final\netty-common-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-buffer\4.1.94.Final\netty-buffer-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-transport\4.1.94.Final\netty-transport-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-codec\4.1.94.Final\netty-codec-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-handler\4.1.94.Final\netty-handler-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-codec-http2\4.1.94.Final\netty-codec-http2-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-resolver-dns\4.1.94.Final\netty-resolver-dns-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-resolver\4.1.94.Final\netty-resolver-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-codec-dns\4.1.94.Final\netty-codec-dns-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-resolver-dns-native-macos\4.1.94.Final\netty-resolver-dns-native-macos-4.1.94.Final-osx-x86_64.jar;C:\Users\USER\.m2\repository\io\netty\netty-resolver-dns-classes-macos\4.1.94.Final\netty-resolver-dns-classes-macos-
4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-transport-native-

epoll\4.1.94.Final\netty-transport-native-epoll-4.1.94.Final-linux-x86_64.jar;C:\Users\USER\.m2\repository\io\netty\netty-transport-native-unix-common\4.1.94.Final\netty-transport-native-unix-common-4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-transport-classes-epoll\4.1.94.Final\netty-transport-classes-epoll-4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\projectreactor\netty\reactor-netty-core\1.0.34\reactor-netty-core-1.0.34.jar;C:\Users\USER\.m2\repository\io\netty\netty-handler-proxy\4.1.94.Final\netty-handler-proxy-4.1.94.Final.jar;C:\Users\USER\.m2\repository\io\netty\netty-codec-socks\4.1.94.Final\netty-codec-socks-4.1.94.Final.jar;C:\Users\USER\.m2\repository\org\springframework\spring-webflux\5.3.29\spring-webflux-5.3.29.jar;C:\Users\USER\.m2\repository\io\projectreactor\reactor-core\3.4.31\reactor-core-3.4.31.jar;C:\Users\USER\.m2\repository\org\reactivestreams\reactive-streams\1.0.4\reactive-streams-1.0.4.jar;C:\Users\USER\.m2\repository\com\itextpdf\itextpdf\5.5.13.3\itextpdf-5.5.13.3.jar;C:\Users\USER\.m2\repository\com\itextpdf\kernel\7.0.2\kernel-7.0.2.jar;C:\Users\USER\.m2\repository\org\slf4j\slf4j-api\1.7.36\slf4j-api-1.7.36.jar;C:\Users\USER\.m2\repository\com\itextpdf\io\7.0.2\io-7.0.2.jar;C:\Users\USER\.m2\repository\com\itextpdf\layout\7.0.2\layout-7.0.2.jar;C:\Users\USER\.m2\repository\com\itextpdf\forms\7.0.2\forms-7.0.2.jar;C:\Users\USER\.m2\repository\com\itextpdf\pdfa\7.0.2\pdfa-7.0.2.jar;C:\Users\USER\.m2\repository\com\itextpdf\sign\7.0.2\sign-7.0.2.jar;C:\Users\USER\.m2\repository\com\itextpdf\barcodes\7.0.2\barcode-s-7.0.2.jar;C:\Users\USER\.m2\repository\com\itextpdf\font-asian\7.0.2\font-asian-7.0.2.jar;C:\Users\USER\.m2\repository\com\itextpdf\hyph\7.0.2\hyph-7.0.2.jar;C:\Users\USER\.m2\repository\net\sf\jasperreports\jasperreports\6.4.0\jasperreports-6.4.0.jar;C:\Users\USER\.m2\repository\commons-beanutils\commons-beanutils\1.9.0\commons-beanutils-1.9.0.jar;C:\Users\USER\.m2\repository\commons-collections\commons-collections\3.2.2\commons-collections-3.2.2.jar;C:\Users\USER\.m2\repository\commons-digester\commons-digester\2.1\commons-digester-2.1.jar;C:\Users\USER\.m2\repository\commons-logging\commons-logging\1.1.1\commons-logging-

1.1.1.jar;C:\Users\USER\.m2\repository\com\lowagie\itext\2.1.7.js5\itext-
2.1.7.js5.jar;C:\Users\USER\.m2\repository\bouncycastle\bcmail-
jdk14\138\bcmail-jdk14-
138.jar;C:\Users\USER\.m2\repository\bouncycastle\bcprov-jdk14\138\bcprov-
jdk14-138.jar;C:\Users\USER\.m2\repository\org\bouncycastle\bctsp-
jdk14\1.38\bctsp-jdk14-
1.38.jar;C:\Users\USER\.m2\repository\org\bouncycastle\bcprov-
jdk14\1.38\bcprov-jdk14-
1.38.jar;C:\Users\USER\.m2\repository\org\bouncycastle\bcmail-
jdk14\1.38\bcmail-jdk14-
1.38.jar;C:\Users\USER\.m2\repository\org\jfree\jcommon\1.0.23\jcommon-
1.0.23.jar;C:\Users\USER\.m2\repository\org\jfree\jfreechart\1.0.19\jfreechart-
1.0.19.jar;C:\Users\USER\.m2\repository\org\eclipse\jdt\core\compiler\ecj\4.3.
1\ecj-4.3.1.jar;C:\Users\USER\.m2\repository\org\codehaus\castor\castor-
xml\1.3.3\castor-xml-
1.3.3.jar;C:\Users\USER\.m2\repository\org\codehaus\castor\castor-
core\1.3.3\castor-core-1.3.3.jar;C:\Users\USER\.m2\repository\commons-
lang\commons-lang\2.6\commons-lang-
2.6.jar;C:\Users\USER\.m2\repository\javax\inject\javax.inject\1\javax.inject-
1.jar;C:\Users\USER\.m2\repository\stax\stax\1.2.0\stax-
1.2.0.jar;C:\Users\USER\.m2\repository\stax\stax-api\1.0.1\stax-api-
1.0.1.jar;C:\Users\USER\.m2\repository\javax\xml\xml\stream\stax-api\1.0-2\stax-
api-1.0-2.jar;C:\Users\USER\.m2\repository\org\apache\lucene\lucene-
core\4.5.1\lucene-core-
4.5.1.jar;C:\Users\USER\.m2\repository\org\apache\lucene\lucene-analyzers-
common\4.5.1\lucene-analyzers-common-
4.5.1.jar;C:\Users\USER\.m2\repository\org\apache\lucene\lucene-
queryparser\4.5.1\lucene-queryparser-
4.5.1.jar;C:\Users\USER\.m2\repository\org\apache\lucene\lucene-
queries\4.5.1\lucene-queries-
4.5.1.jar;C:\Users\USER\.m2\repository\org\apache\lucene\lucene-
sandbox\4.5.1\lucene-sandbox-
4.5.1.jar;C:\Users\USER\.m2\repository\jakarta-regexp\jakarta-
regexp\1.4\jakarta-regexp-
1.4.jar;C:\Users\USER\.m2\repository\org\olap4j\olap4j\0.9.7.309-JS-
3\olap4j-0.9.7.309-JS-
3.jar;C:\Users\USER\.m2\repository\com\google\zxing\core\3.2.1\core-

3.2.1.jar;C:\Users\USER\.m2\repository\com\ibm\icu\icu4j\57.1\icu4j-57.1.jar;C:\Users\USER\.m2\repository\org\springframework\spring-jdbc\5.3.23\spring-jdbc-5.3.23.jar;C:\Users\USER\.m2\repository\org\springframework\spring-beans\5.3.29\spring-beans-5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\spring-tx\5.3.29\spring-tx-5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\security\oauth\spring-security-oauth2\2.5.2.RELEASE\spring-security-oauth2-2.5.2.RELEASE.jar;C:\Users\USER\.m2\repository\org\springframework\spring-context\5.3.29\spring-context-5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\security\spring-security-core\5.7.10\spring-security-core-5.7.10.jar;C:\Users\USER\.m2\repository\org\springframework\security\spring-security-crypto\5.7.10\spring-security-crypto-5.7.10.jar;C:\Users\USER\.m2\repository\org\springframework\security\spring-security-config\5.7.10\spring-security-config-5.7.10.jar;C:\Users\USER\.m2\repository\org\springframework\security\spring-security-web\5.7.10\spring-security-web-5.7.10.jar;C:\Users\USER\.m2\repository\commons-codec\commons-codec\1.15\commons-codec-1.15.jar;C:\Users\USER\.m2\repository\org\springframework\security\spring-security-jwt\1.0.9.RELEASE\spring-security-jwt-1.0.9.RELEASE.jar;C:\Users\USER\.m2\repository\org\bouncycastle\bcpkix-jdk15on\1.56\bcpkix-jdk15on-1.56.jar;C:\Users\USER\.m2\repository\org\bouncycastle\bcprov-jdk15on\1.56\bcprov-jdk15on-1.56.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-starter-security\2.7.14\spring-boot-starter-security-2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\spring-aop\5.3.29\spring-aop-5.3.29.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-starter-mail\2.7.14\spring-boot-starter-mail-2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\spring-context-support\5.3.29\spring-context-support-5.3.29.jar;C:\Users\USER\.m2\repository\com\sun\mail\jakarta.mail\1.6.7\jakarta.mail-

1.6.7.jar;C:\Users\USER\.m2\repository\com\sun\activation\jakarta.activation\1.2.2\jakarta.activation-
1.2.2.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-devtools\2.7.14\spring-boot-devtools-
2.7.14.jar;C:\Users\USER\.m2\repository\org\springframework\boot\spring-boot-autoconfigure\2.7.14\spring-boot-autoconfigure-
2.7.14.jar;C:\Users\USER\.m2\repository\com\google\apis\google-api-services-oauth2\v2-rev65-1.17.0-rc\google-api-services-oauth2-v2-rev65-1.17.0-rc.jar;C:\Users\USER\.m2\repository\com\google\api-client\google-api-client\1.17.0-rc\google-api-client-1.17.0-rc.jar;C:\Users\USER\.m2\repository\com\google\oauth-client\google-oauth-client\1.17.0-rc\google-oauth-client-1.17.0-rc.jar;C:\Users\USER\.m2\repository\com\google\api-client\google-api-client-jackson2\1.20.0\google-api-client-jackson2-
1.20.0.jar;C:\Users\USER\.m2\repository\com\google\http-client\google-http-client-jackson2\1.42.2\google-http-client-jackson2-
1.42.2.jar;C:\Users\USER\.m2\repository\com\google\http-client\google-http-client\1.42.2\google-http-client-
1.42.2.jar;C:\Users\USER\.m2\repository\org\apache\httpcomponents\httpclient\4.5.14\httpclient-
4.5.14.jar;C:\Users\USER\.m2\repository\org\apache\httpcomponents\httpcore\4.4.16\httpcore-
4.4.16.jar;C:\Users\USER\.m2\repository\com\google\code\findbugs\jsr305\3.0.2\jsr305-
3.0.2.jar;C:\Users\USER\.m2\repository\com\google\guava\guava\30.1.1-android\guava-30.1.1-android.jar;C:\Users\USER\.m2\repository\com\google\guava\failureaccess\1.0.1\failureaccess-
1.0.1.jar;C:\Users\USER\.m2\repository\com\google\guava\listenablefuture\9999.0-empty-to-avoid-conflict-with-guava\listenablefuture-9999.0-empty-to-avoid-conflict-with-
guava.jar;C:\Users\USER\.m2\repository\org\checkerframework\checker-compat-qual\2.5.5\checker-compat-qual-
2.5.5.jar;C:\Users\USER\.m2\repository\com\google\errorprone\error_prone_annotations\2.5.1\error_prone_annotations-
2.5.1.jar;C:\Users\USER\.m2\repository\com\google\j2objc\j2objc-annotations\1.3\j2objc-annotations-

```

1.3.jar;C:\Users\USER\.m2\repository\io\opencensus\opencensus-
api\0.31.1\opencensus-api-
0.31.1.jar;C:\Users\USER\.m2\repository\io\grpc\grpc-context\1.27.2\grpc-
context-1.27.2.jar;C:\Users\USER\.m2\repository\io\opencensus\opencensus-
contrib-http-util\0.31.1\opencensus-contrib-http-util-
0.31.1.jar;C:\Users\USER\.m2\repository\com\google\oauth-client\google-
oauth-client-jetty\1.34.1\google-oauth-client-jetty-
1.34.1.jar;C:\Users\USER\.m2\repository\com\google\oauth-client\google-
oauth-client-java6\1.34.1\google-oauth-client-java6-1.34.1.jar"
com.aplikasi.karyawan.testing OAuth2Sample

Please open the following address in your browser:
https://accounts.google.com/o/oauth2/auth?client\_id=752952635165-rdp8q9jtdge93efnvl6i8vcm2u1l21nm.apps.googleusercontent.com&redirect\_uri=http://localhost:51172/Callback&response\_type=code&scope=https://www.googleapis.com/auth/userinfo.profile%20https://www.googleapis.com/auth/userinfo.email

Attempting to open that address in the default browser now...
Nov 18, 2023 11:08:33 AM
AM
com.google.api.client.googleapis.services.AbstractGoogleClient <init>
WARNING: Application name is not set. Call Builder#setApplicationName.
token          saya = 
ya29.a0AfB_byB0yLCpHJGZolu5ZoTofhpDUXytP0_AL8LT4zcj3i08CdCI-
1Ce9fVjnCHqYelVv80gzzFyRVtBBzdkAPs-
E1KWXQg70SxbI0LA03SXkgq9aTdXdlsMgPsqHF_NQSKwwn62MmY_Py-
ArSPHKaSBJ6-
Uf0uiuVT6aCgYKAb8SARISFQHGx2MiX004ViPaZFkL0vbiQ7mEeg0I71

===== Validating a token =====

{
  "access_type" : "online",
  "audience" : "752952635165-
rdp8q9jtdge93efnvl6i8vcm2u1l21nm.apps.googleusercontent.com",
  "email" : "rikialdipari@gmail.com",
  "expires_in" : 3599,
  "issued_to" : "752952635165-
rdp8q9jtdge93efnvl6i8vcm2u1l21nm.apps.googleusercontent.com",

```

```
"scope" : "https://www.googleapis.com/auth/userinfo.email  
https://www.googleapis.com/auth/userinfo.profile openid",  
"user_id" : "11311117324794844356",  
"verified_email" : true  
}  
  
===== Obtaining User Profile Information =====  
  
{  
"email" : "rikialdipari@gmail.com",  
"family_name" : "pari",  
"given_name" : "riki aldi",  
"id" : "11311117324794844356",  
"locale" : "en",  
"name" : "riki aldi pari",  
"picture" :  
"https://lh3.googleusercontent.com/a/ACg8ocIdWugzrahSEQt5sNgQ4iWF0s  
FR3RpF96zkL10kCEEwcAk=s96-c",  
"verified_email" : true  
}
```

```
Process finished with exit code 0
```

m. Lakukan testing di postman untuk endpoint login atau register

The screenshot shows a POST request to `http://localhost:8081/api/login-user/signin_google`. The 'Body' tab is selected, showing a key-value pair `accessToken` with the value `ya29.a0AV...7d...`. The response body is highlighted with a red box and contains a JSON object with various fields like `access_token`, `refresh_token`, `scope`, `token_type`, `message`, `type`, `expires_in`, and `jti`.

```
1 "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCIkXVCJ9.
2     eyJhdWQiolsib2F1dGgyLXJlc29icmNlIl0sImV4cCI6MTY2MjUyM2MyMiwidXNlc19uYWllIjoicmlraWFsZGlwYXJpQGdtYWlsImvbSiImp0aSI6IjQ4YzIwYjdLTd
3     mItNDMyMC1NjQ2LWF1ZG12ZjU4ZWNlMyIsImNsawVudF9pZC16Im15LNsaWVudC13ZW1lCJzY29wZSI6WyJyZwFkIiwid3JpdGUxx0.
4     WQM5oL_FiauWeuhXIsOEKhjaxX2rHnsQPvWzTngw",
5     "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIkXVCJ9.
6     eyJhdWQiolsib2F1dGgyLXJlc29icmNlIl0sInVzZXifmftZSI6InJpa2lhGRpcGFyaUBnbWFpbC5jb201LCJzY29wZSI6WyJyZwFkIiwid3JpdGUxSwiYXRpIjoiNDh
7     jB1N2MtN2Yyy@0MZiwlWj2NDYtWJyKjYjZmNTh1Y2UziwiZXhwIjoxNjY4NTcxM2IyLCJqdGkiOj1M2RhMmYy1hZTRkLTQwMzk1OTBjZS04MzIwMDU3OTl1NGUiLCJj
8     llbnRfaWQiojteS1jbGllbnQtd2ViIn0.Qvd13vXomaQNqbeEsj3ZXcbb_d-iJN_VUBDEVMa1SbZE",
9     "scope": "read write",
10    "token_type": "bearer",
11    "message": "sukses",
12    "type": "login",
13    "expires_in": 1209599,
14    "jti": "48c20b7c-7f2c-4320-b646-abdb6f58ece3",
15    "access_token": "ya29.a0AV...7d..."}
```

Login berhasil dengan oauth google.

n. referensi

<https://www.webaik.com/2020/08/cara-mendapatkan-oauth-2-api-token-layanan-google.html>

o. Logout oauth2.0 google

```
package com.synrgy.commit.google;

public class LogoutGoogleOauth {
    public static void main(String[] args) {
        System.out.println("klik :
https://www.google.com/accounts/Logout?continue=https://appengine.google.com/_ah/logout?continue=http://localhost:8081");
    }
}
```

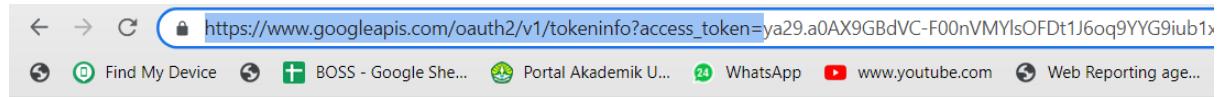
p. Melihat informasi expired token oauth google

https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=ftokenanda

contoh

https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=ya29.a0AX9GBdVC-F00nVMyIs0FDt1J6oq9YYG9iub1xLdkta...

output



```
{  
  "issued_to": "985707925670-plhu4csce96ampd314mmvemef3k5l225.apps.googleusercontent.com",  
  "audience": "985707925670-plhu4csce96ampd314mmvemef3k5l225.apps.googleusercontent.com",  
  "user_id": "113111117324794844356",  
  "scope": "https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/auth/userinfo.email openid",  
  "expires_in": 3223,  
  "email": "rikialdipari@gmail.com",  
  "verified_email": true,  
  "access_type": "online"  
}
```

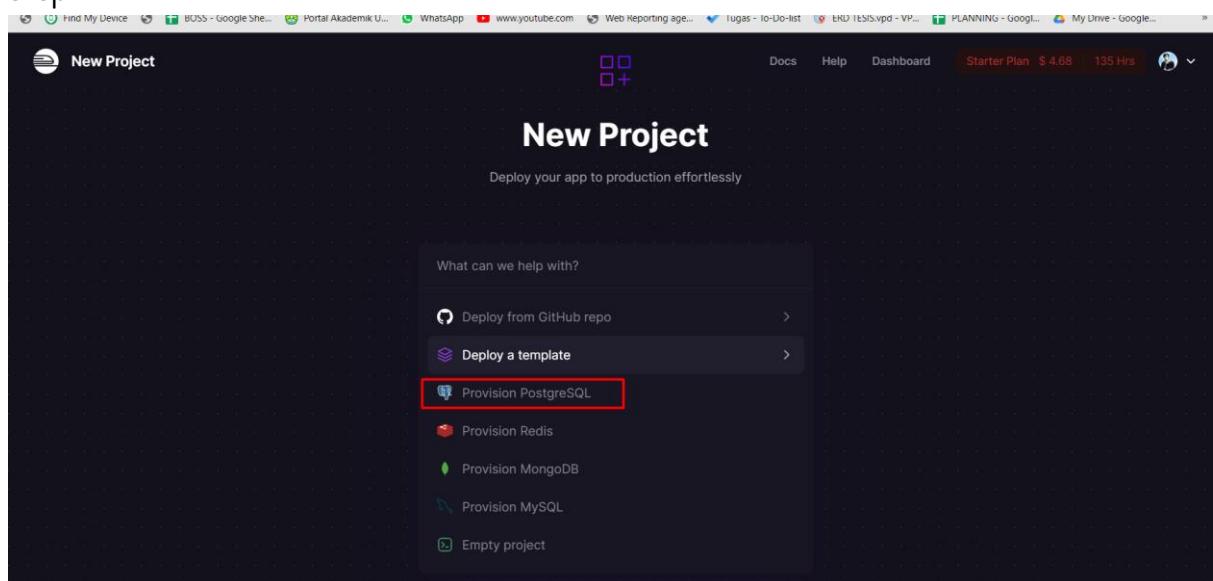
36.Railway.app : deploy project

a. Register uses github

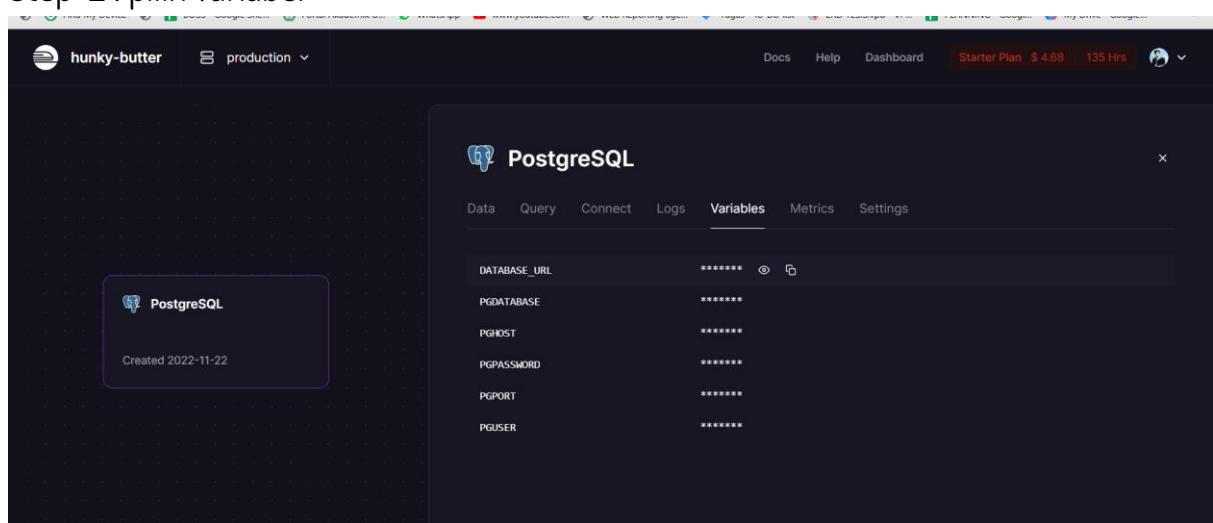
<https://railway.app/dashboard>

b. create database

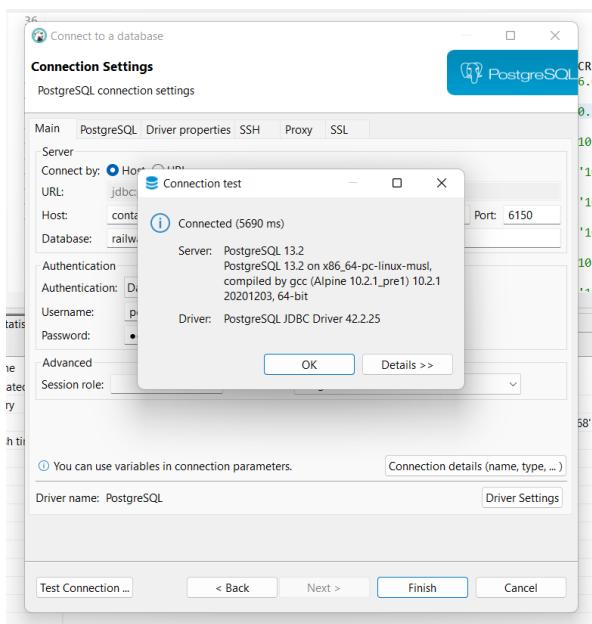
step-1



Step-2 : pilih variabel



Step 3-Test Koneksi DBEaver

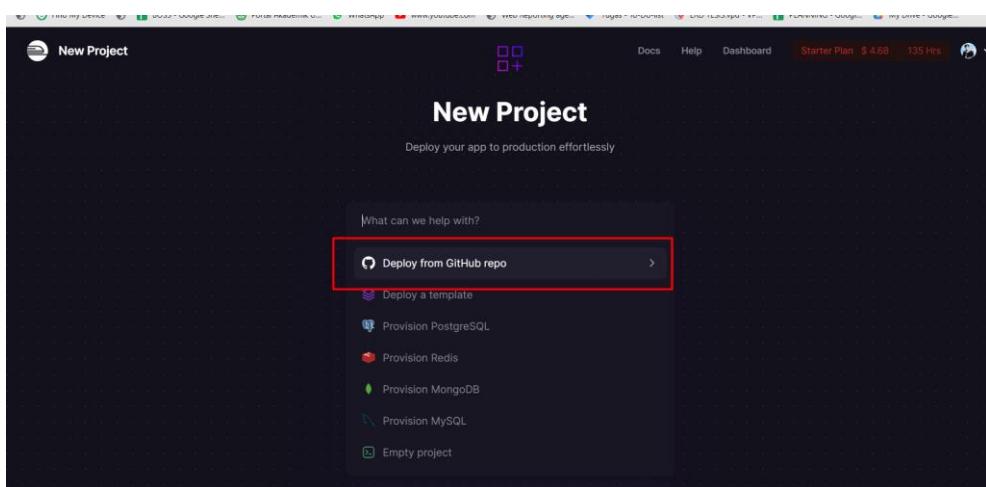


Referensi : <https://ekomenyong.com/posts/how-to-setup-free-postgresql-database-on-railway-app>

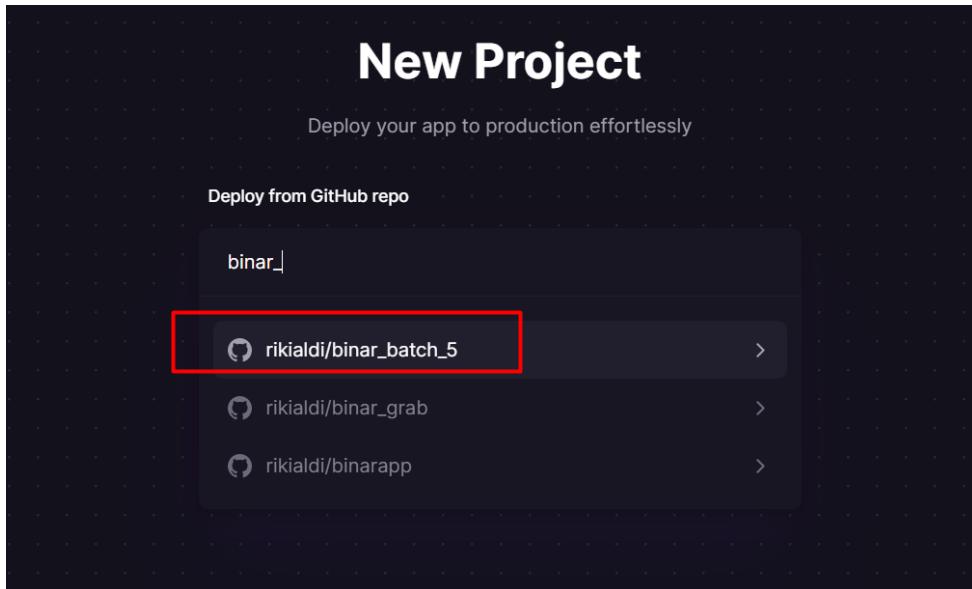
c. Push project di github

d. Create project : di railway

Step-1 : pilih github



Step-2: pilih project github



Step-3:Pom.xml

Ganti dengan jdk 1.8 or 11

The screenshot shows an IDE interface with a file tree on the left and code editor on the right. The file tree shows various files like logback-spring.xml, test, target, .gitignore, .gitlab-ci.yml, 909090docker-9090compose.yml, docker-compose.yml, Dockerfile, mvnw, mvnw.cmd, and pom.xml. The pom.xml file is open in the editor, showing XML code. The line '<java.version>1.8</java.version>' is highlighted with a red box.

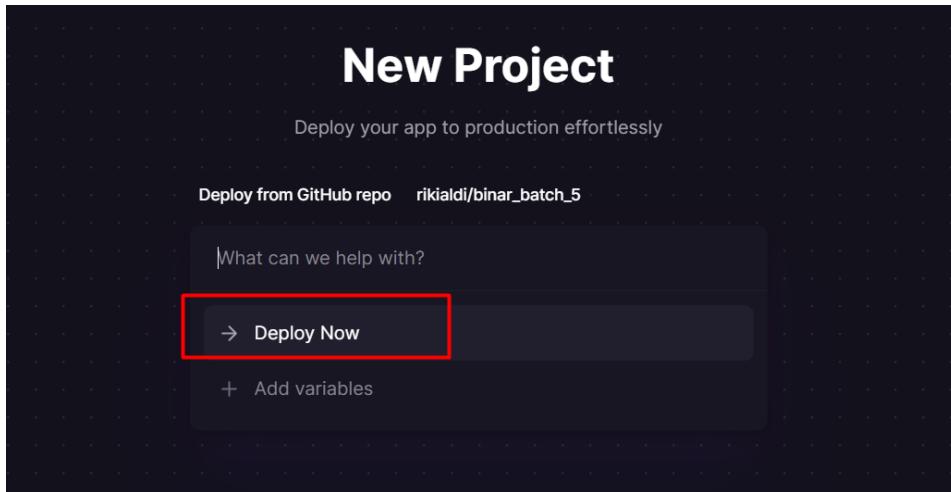
```
<description>Project Binar Batch 5</description>
<properties>
    <java.version>1.8</java.version>
    <oauth2.version>2.0.14.RELEASE</oauth2.version>
    <jwt.version>1.0.9.RELEASE</jwt.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>
    <dependency>
```

Setting application properties : dengan database railway.app

The screenshot shows a terminal or code editor window displaying application properties. Lines 1 through 5 are visible, showing configuration for a Spring Boot application. The database configuration (lines 4-5) is highlighted with a red box.

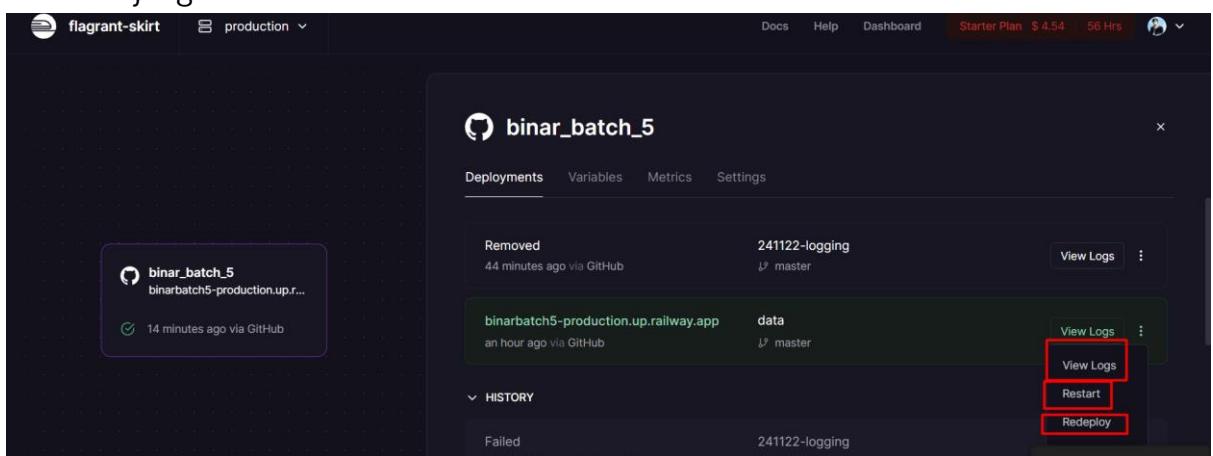
```
1 server.servlet.context-path=/api
2 server.port =9090
3 spring.jpa.hibernate.ddl-auto=create
4 spring.datasource.url=jdbc:postgresql://containers-us-west-53.railway.app:6125/railway
5 spring.datasource.username=postgres
```

Step-4: deploy

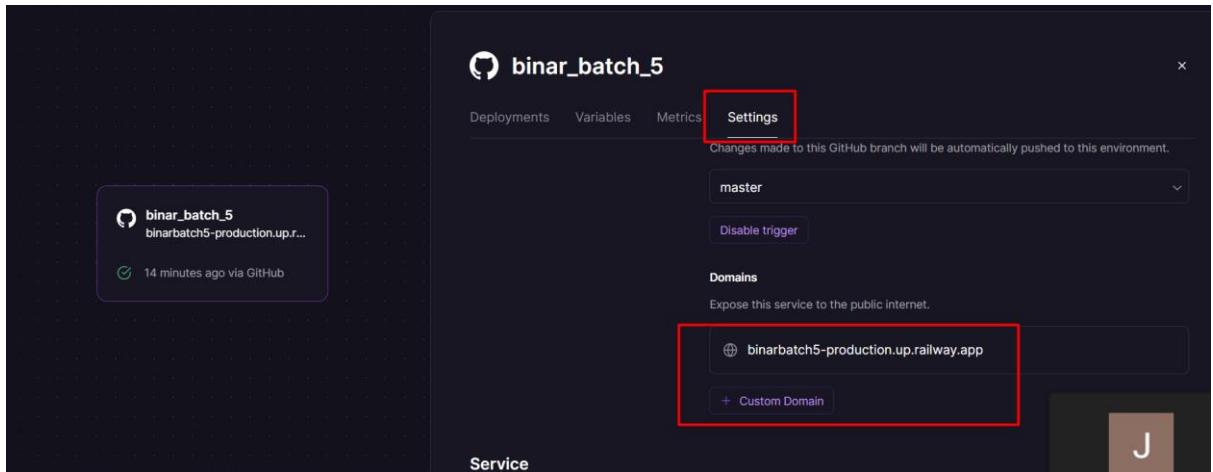


Redeploy ulang : jika gagal deploy

Pastikan jangan ada file dokcer : karena akan otomatis ke sana

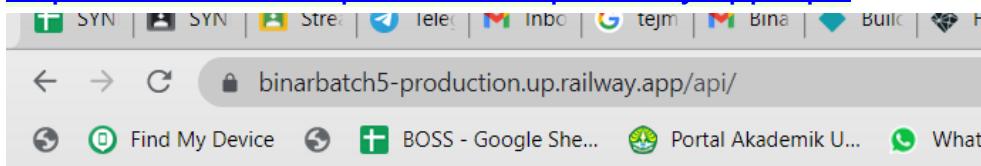


Jika sudah berhasil : lakukan genereate dns hostname



Jalankan :

<https://binarbatch5-production.up.railway.app/api/>



Ini adalah web pertama saya

e. Railway with Docker-compose

01122022-docker-compose-success

https://github.com/rikialdi/binar_batch_5/tree/01122022-docker-compose-success

perhatikan port harus sesuai

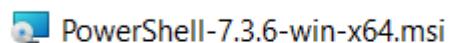
37.FLY.IO-Not Working

- [Hands-on with Fly.io](#)
 - [1. Install flyctl](#)
 - [2. Sign up](#)
 - [3. Sign in](#)
 - [4. Launch app](#)
 - [5. Check app status](#)
 - [6. Visit app](#)
 - [7. Learn more](#)

a. Instalasi

LINK : <https://fly.io/docs/hands-on/install-flyctl/>

- Jika belum install powershel, install dulu <https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-windows?view=powershell-7.3>
- Cara instalasi



Klik 2 kali, next sampai selesai

- Install fly.io
Success instalasi

A screenshot of a Windows PowerShell terminal window titled "PowerShell 7.3.6". The window shows the command "pwsh -Command "iwr https://fly.io/install.ps1 -useb | iex"" being run, which installs flyctl. The output indicates that flyctl was installed successfully to C:\Users\USER\.fly\bin\flyctl.exe and provides instructions to run 'flyctl --help' to get started.

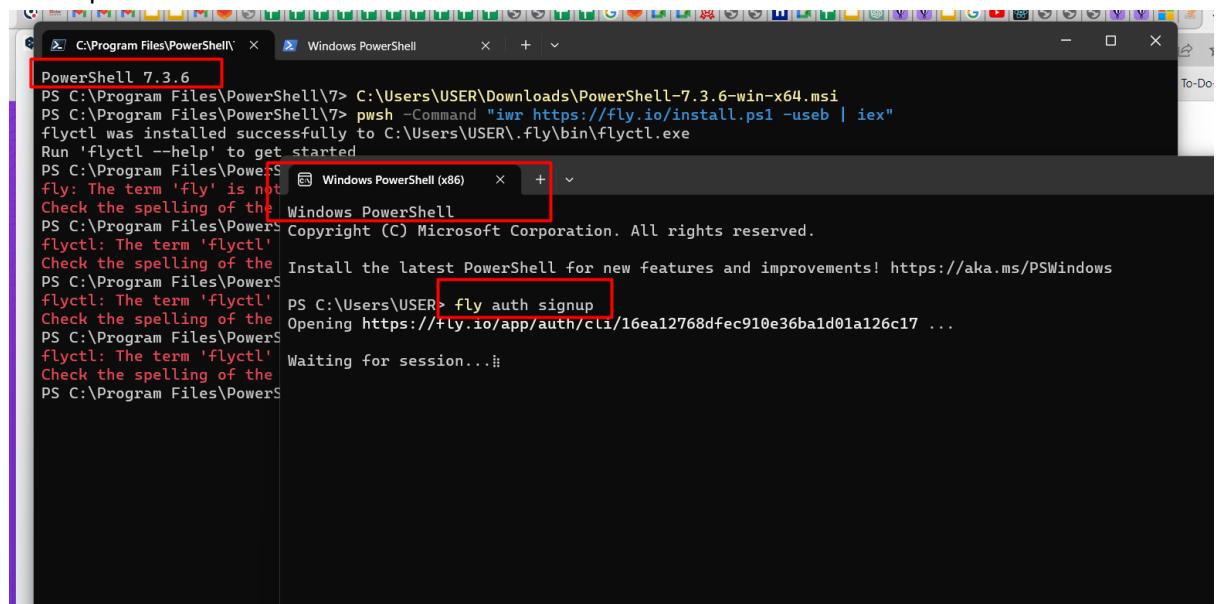
```
PS C:\Program Files\PowerShell\7> C:\Users\USER\Downloads\PowerShell-7.3.6-win-x64.msi
PS C:\Program Files\PowerShell\7> pwsh -Command "iwr https://fly.io/install.ps1 -useb | iex"
flyctl was installed successfully to C:\Users\USER\.fly\bin\flyctl.exe
Run 'flyctl --help' to get started
PS C:\Program Files\PowerShell\7>
```

b. Sign-up

<https://fly.io/docs/hands-on/sign-up/>

```
fly auth signup
```

Buka powersehel baru kemudian,

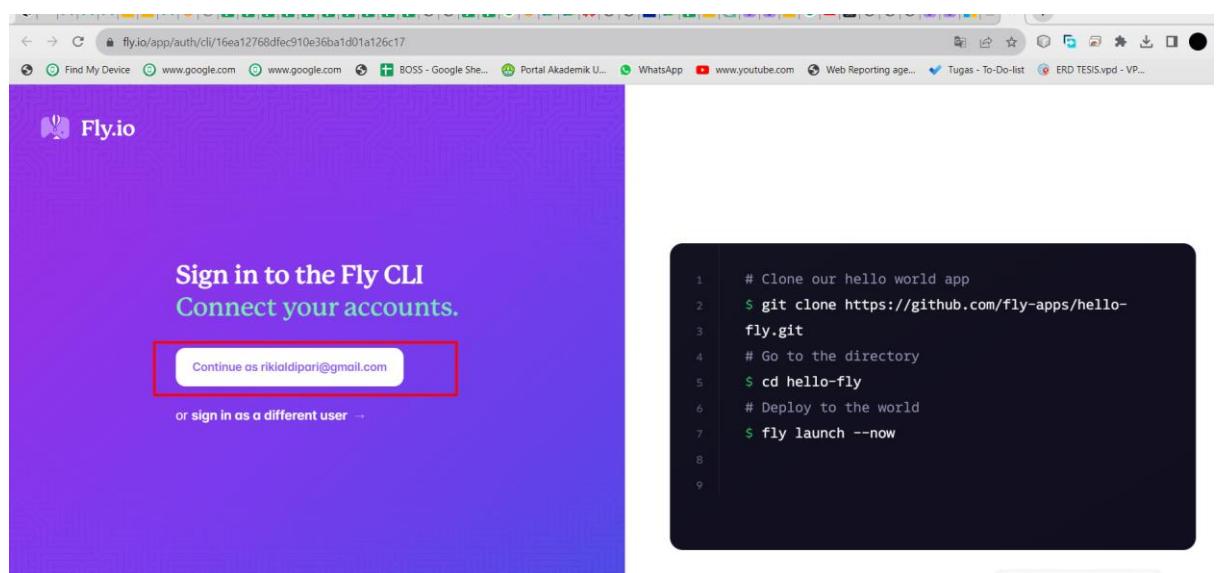


```
PowerShell 7.3.6
PS C:\Program Files\PowerShell\7> C:\Users\USER\Downloads\PowerShell-7.3.6-win-x64.msi
PS C:\Program Files\PowerShell\7> pwsh -Command "iwr https://fly.io/install.ps1 -useb | iex"
flyctl was installed successfully to C:\Users\USER\.fly\bin\flyctl.exe
Run 'flyctl --help' to get started
PS C:\Program Files\PowerShell\7> fly auth signup
fly: The term 'fly' is not
Check the spelling of the
flyctl: The term 'flyctl'
Check the spelling of the
flyctl: The term 'flyctl'
Check the spelling of the
PS C:\Program Files\PowerShell\7> fly auth signup
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\USER> fly auth signup
Opening https://fly.io/app/auth/cli/16ea12768dfec910e36ba1d01a126c17 ...
Waiting for session...#
```

Akan refer ke link

Akan redirect ke link berikut



```

1 # Clone our hello world app
2 $ git clone https://github.com/fly-apps/hello-
3 fly.git
4 # Go to the directory
5 $ cd hello-fly
6 # Deploy to the world
7 $ fly launch --now
8
9

```

Windows PowerShell (x86)

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Users\USER> fly auth signup
Opening https://fly.io/app/auth/cli/16ea12768dfec910e36ba1d01a126c17 ...

Waiting for session... Done
successfully logged in as rikialdipari@gmail.com
PS C:\Users\USER> fly status

```

c. Login

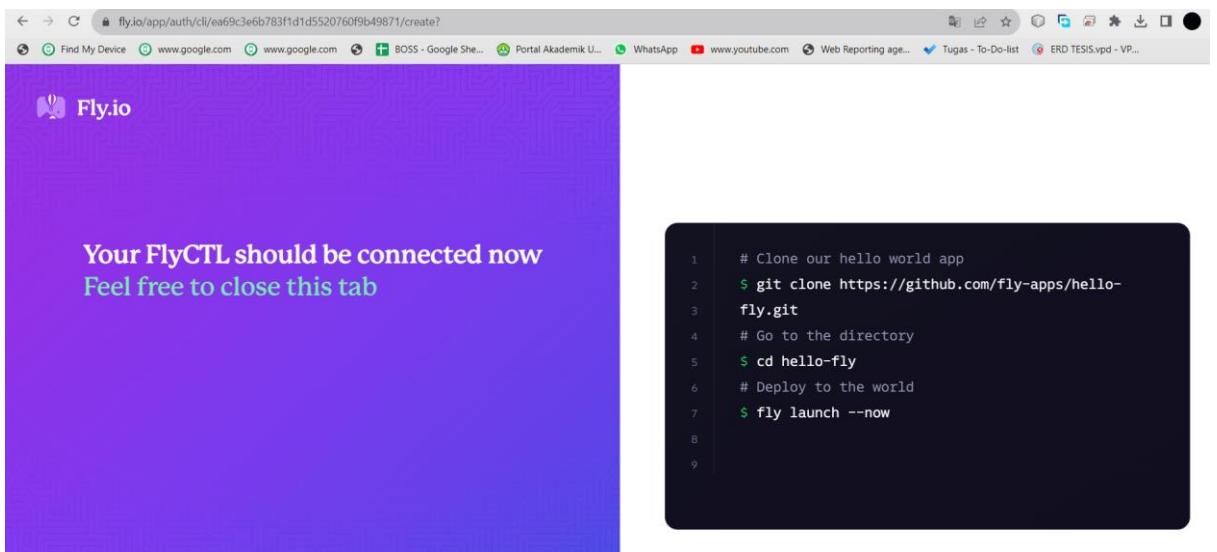
```
fly auth login
```

Cd dulu ke project anda

```
PS C:\Users\USER> cd "E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo"
```

```
PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo> fly auth login
```

Kemudian akan muncul, dan tutup saja



Pengguna Microsoft WSL mungkin perlu menjalankan perintah berikut untuk membuat ini berfungsi:

WAJIB

```
ln -s /usr/bin/wslview /usr/local/bin/xdg-open
```

d. Check Your App's Status

```
PS C:\Users\USER> cd "E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo"
PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo> |
```

e. Launching

<https://fly.io/docs/hands-on/launch-app/>

```
fly launch --image flyio/hellofly:latest
```

Klik Y

```
1 PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo> fly launch --image flyio/hellofly:latest
2 Creating app in E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo
3 ? Would you like to copy its configuration to the new app? Yes
4 Using image flyio/hellofly:latest configuration to the new app? (y/N) y
5 ? Choose an app name (leave blank to generate one):
```

Ketik nama aplikasi anda

```

+ FullyQualifiedErrorId : CommandNotFoundException

PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo> fly launch --image flyio/hellofly:latest
Creating app in E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo
? Would you like to copy its configuration to the new app? Yes
> Using image flyio/hellofly:latestiguration to the new app? (y/N) y
? Choose an app name (leave blank to generate one) demo
Some regions require a paid plan (bom, fra, maa).
See https://fly.io/plans to set up a plan.

? Choose a region for deployment: [Use arrows to move, type to filter]
  Madrid, Spain (mad)
  Miami, Florida (US) (mia)
  Tokyo, Japan (nrt)
  Chicago, Illinois (US) (ord)
  Bucharest, Romania (otp)
  Phoenix, Arizona (US) (phx)
  Querétaro, Mexico (qro)
  Santiago, Chile (scl)
  Seattle, Washington (US) (sea)
> Singapore, Singapore (sin)
  San Jose, California (US) (sjc)
  Sydney, Australia (syd)
  Warsaw, Poland (waw)
  Montreal, Canada (yul)
  Toronto, Canada (yyz)

? App wild-star-2054 already exists, do you want to launch into that app? No
PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo> fly status
App
  Name      = wild-star-2054
  Owner     = personal
  Hostname  = wild-star-2054.fly.dev
  Image     =
  Platform  = machines

PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo> fly launch --image flyio/hellofly:latest
Creating app in E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo
? Would you like to copy its configuration to the new app? Yes
> Using image flyio/hellofly:latestiguration to the new app? (y/N) y
? App wild-star-2054 already exists, do you want to launch into that app? Yes
App will use 'nrt' region as primary do you want to launch into that app? (y/N) y

  Admin URL: https://fly.io/apps/wild-star-2054
  Hostname: wild-star-2054.fly.dev
? Would you like to set up a Postgresql database now? (y/N)

Muhammad Isa Cimb

```

Open URL yang di terminal, pilih token

NAME	USER	EXPIRES AT	ACTIONS
Deploy Token	Riki aldi pari	2123-08-24T12:36:31	

Deploy token

FlyV1

```
fm1r_IJPECAAAAAAAAXdxBBa7RdWHzQwwX+En2Pq15JAwrVodHRwczovL2FwaS5mbHkuaW8vdjGWAJLOAAS01B8Lk7lodHRwczovL2FwaS5mbHkuaW8vYwFhL3YxxDyxD/GMpYBFjMnSxG44skgSDSZ2iE2m00blFdyL1ZKgkCU0g3C0ZS5YP0p22ykAqKWbalWB/MKfRn5ITHTER998qKmnenu5vc5GRGITNtvfP0DTfabVe4avtAMKmomtMEe0K9/MyvAwlEBpIDH0xrF5Hr+eJSHEM7rQjPK8DgzsgX/420EVdzKUA5GBzgAbPAQfBZGCp2J1aWxkZXIfondnHwHEIGmhCgFdZS6p+5bhTJeknkeCERoINiDheYAGGNRYNGOr,fm1a_IJPER998qKmnenu5vc5GRGITNtvfP0DTfabVe4avtAMKmomtMEe0K9/MyvAwlEBpIDH0xrF5Hr+eJSHEM7rQjPK8DgzsgX/420EVxBBrqu3IKAnEoCllt1Ld0c+pw7lodHRwczovL2FwaS5mbHkuaW8vYwFhL3YxlgSSzmUG8rHPAAAAASD/EM8Kkc4ABDr0DMQQVFEWUy3Rlu mmabf0hNWhDcQglrXXy9E9j68ZbZJjSx9gxCdliKE6H3zjrS74y9ZNpmg=
```

f. **fly apps open /fred**

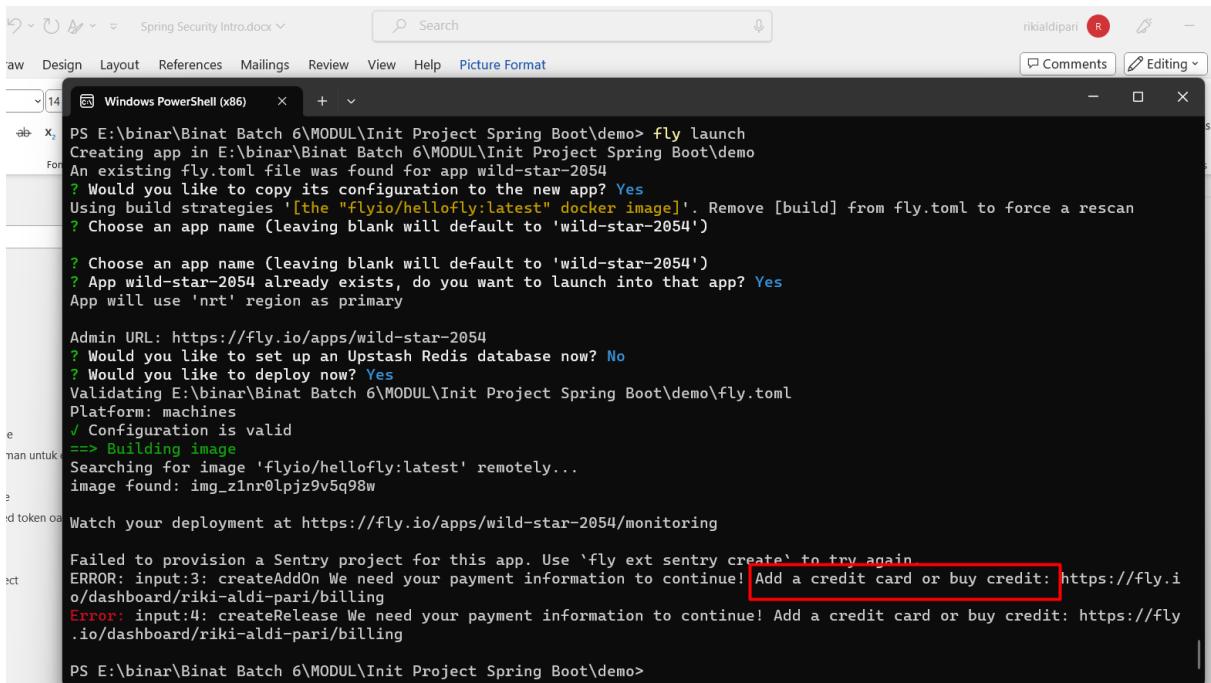
akan otomatis kebuat file

The screenshot shows a Java development environment with the following details:

- Project Structure:** The project is named "demo". It contains a ".mvn" folder, a "src" folder with "main" and "test" subfolders, and a "target" folder. Inside "src/main/java", there is a package "com.example.demo" containing classes "HelloController" and "DemoApplication". Inside "src/main/resources", there is a "static" folder, a "templates" folder, and a "application.properties" file. A file "fly.toml" is also present in the "src" directory.
- fly.toml Content:**

```
1 # fly.toml app configuration file generated for wild-star-2054
2 #
3 # See https://fly.io/docs/reference/configuration/ for information
4 #
5
6 app = "wild-star-2054"
7 primary_region = "nrt"
8
9 [build]
10 image = "flyio/hellofly:latest"
11
12 [http_service]
13 internal_port = 8080
14 force_https = true
15 auto_stop_machines = true
16 auto_start_machines = true
17 min_machines_running = 0
18 processes = ["app"]
19
```
- Run Tab:** Shows log output from the application's startup. The logs indicate the application is running on port 8080 using Tomcat and the embedded Tomcat web server.
- Bottom Navigation:** Includes tabs for Version Control, Run, TODO, Problems, Terminal, SonarLint, SonarAnalyzer, Services, Build, and Dependencies.

g. Failed, karena perlu payment cc



Spring Security Intro.docx

Search

raw Design Layout References Mailings Review View Help Picture Format

Comments Editing

Windows PowerShell (x86)

```
PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo> fly launch
Creating app in E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo
An existing fly.toml file was found for app wild-star-2054
? Would you like to copy its configuration to the new app? Yes
Using build strategies '[the "flyio/hellofly:latest" docker image]'. Remove [build] from fly.toml to force a rescan
? Choose an app name (leaving blank will default to 'wild-star-2054')
? Choose an app name (leaving blank will default to 'wild-star-2054')
? App wild-star-2054 already exists, do you want to launch into that app? Yes
App will use 'nrt' region as primary

Admin URL: https://fly.io/apps/wild-star-2054
? Would you like to set up an Upstash Redis database now? No
? Would you like to deploy now? Yes
Validating E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo\fly.toml
Platform: machines
✓ Configuration is valid
==> Building image
Searching for image 'flyio/hellofly:latest' remotely...
image found: img_z1nr0lpjz9v5q98w

Watch your deployment at https://fly.io/apps/wild-star-2054/monitoring

Failed to provision a Sentry project for this app. Use 'fly ext sentry create' to try again
ERROR: input:3: createAddOn We need your payment information to continue! Add a credit card or buy credit: https://fly.i
o/dashboard/riki-aldi-pari/billing
Error: input:4: createRelease We need your payment information to continue! Add a credit card or buy credit: https://fly
.io/dashboard/riki-aldi-pari/billing

PS E:\binar\Binat Batch 6\MODUL\Init Project Spring Boot\demo>
```

38.Download sub Folder Github

<https://minhaskamal.github.io/DownGit/>

39.Kafka : message bloker

a. Install Download Kafka

URL

<https://www.geeksforgeeks.org/how-to-install-and-run-apache-kafka-on-windows/>

Link download kafka

<https://kafka.apache.org/downloads>

Apache Kafka can be downloaded from its official site kafka.apache.org



HOME
INTRODUCTION
QUICKSTART
USE CASES
DOCUMENTATION
PERFORMANCE
POWERED BY
PROJECT INFO
ECOSYSTEM
CLIENTS
EVENTS
CONTACT US
APACHE

Download

2.5.0 is the latest release. The current stable version is 2.5.0.

You can verify your download by following these [procedures](#) and using these [KEYS](#).

2.5.0

- Released April 15, 2020
- [Release Notes](#)
- Source download: [kafka_2.5.0-src.tgz \(asc, sha512\)](#)
- Binary downloads:
 - Scala 2.12 - [kafka_2.12-2.5.0.tgz \(asc, sha512\)](#)
 - Scala 2.13 - [kafka_2.13-2.5.0.tgz \(asc, sha512\)](#)

We build for multiple versions of Scala. This only matters if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.12 is recommended).

Kafka 2.5.0 includes a number of significant new features. Here is a summary of some notable changes:

For the installation process, follow the steps given below:

Step 1: Go to the Downloads folder and select the downloaded Binary file.

Step 2: Extract the file and move the extracted folder to the directory where you wish to keep the files.

Step 3: Copy the path of the Kafka folder. Now go to *config* inside kafka folder and open *zookeeper.properties* file. Copy the path against the field *dataDir* and add */zookeeper-data* to the path.

```
13 # See the License for the specific language governing permissions
14 # limitations under the License.
15 # the directory where the snapshot is stored.
16 dataDir=c:/kafka/zookeeper-data
17 # the port at which the clients will connect
18 clientPort=2181
19 # disable the per-ip limit on the number of connections since
20 maxClientCnxns=0
```

For example if the path is *c:/kafka*

Step 4: Now in the same folder *config* open *server.properties* and scroll down to *log.dirs* and paste the path. To the path add */kafka-logs*

Step 5: This completes the configuration of zookeeper and kafka server. Now open command prompt and change the directory to the kafka folder. First start zookeeper using the command given below:

```
.\bin\windows\zookeeper-server-start.bat
.\config\zookeeper.properties
```

```
C:\kafka> .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
[2020-06-09 00:17:04,344] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-06-09 00:17:04,352] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-06-09 00:17:04,353] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-06-09 00:17:04,355] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2020-06-09 00:17:04,356] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2020-06-09 00:17:04,381] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2020-06-09 00:17:04,383] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
```

Step 6: Now open another command prompt and change the directory to the kafka folder. Run kafka server using the command:

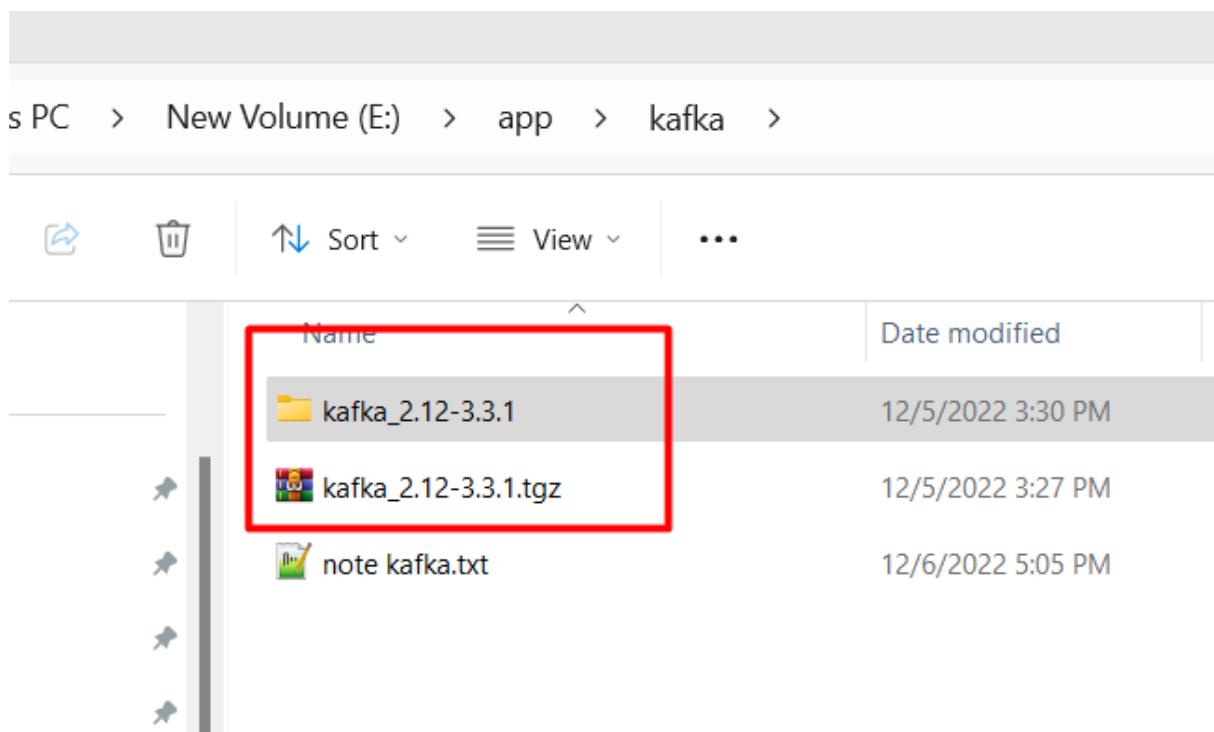
```
.\bin\windows\kafka-server-start.bat
.\config\server.properties
```

```
C:\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties
[2020-06-09 00:30:10,103] INFO Registered Kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jCon
n$)
[2020-06-09 00:30:10,985] INFO starting (kafka.server.KafkaServer)
[2020-06-09 00:30:10,991] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2020-06-09 00:30:11,016] INFO [ZooKeeperClient] Initializing a new session to localhost:2181. (kafka.
rClient)
```

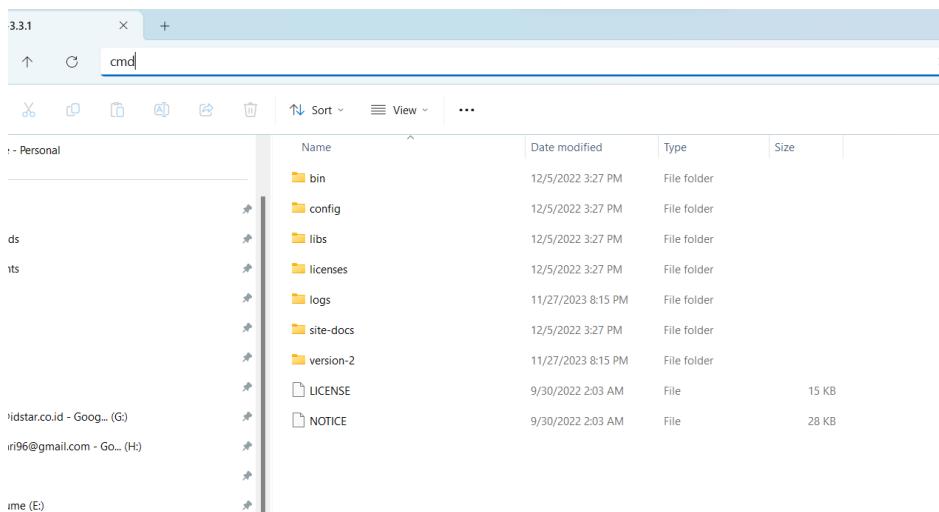
Now kafka is running and ready to stream data.

Praktek

Step 1 : hasil

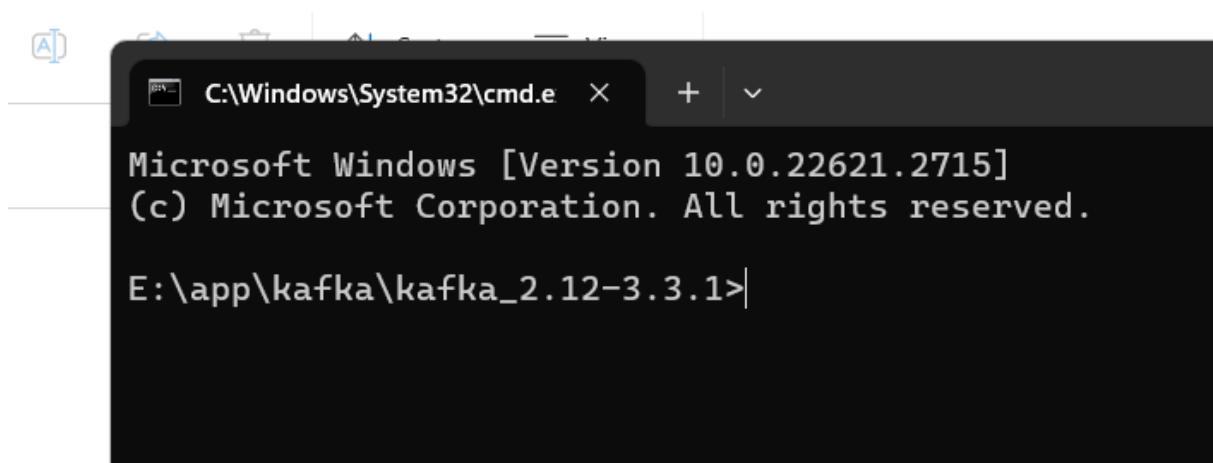


Step 2 : masuk ke folfer , ketikkan CMD



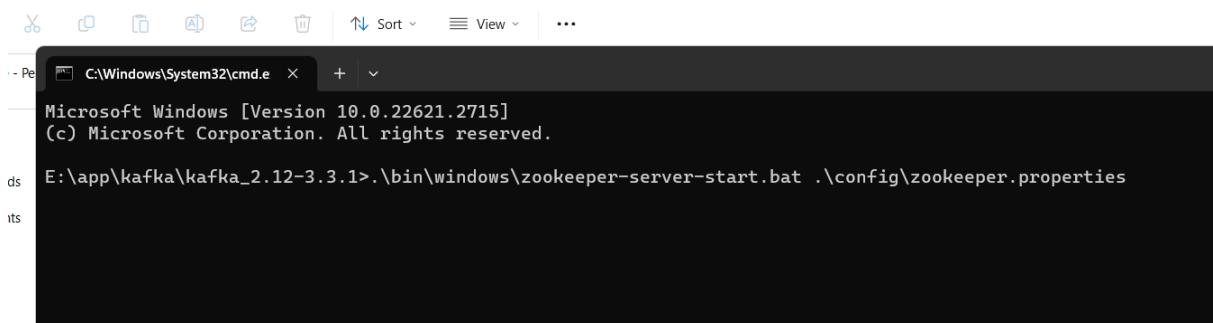
Dan akan muncul

> This PC > New volume (E:) > app > Kafka > Kafka_2.12-3.3.1 >



Step 3 : ketikkan untuk running

```
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```



Berikut outputnya

```

C:\Windows\System32\cmd.e x + v
nFactory)
[2023-11-27 20:18:31,007] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 2 selector threads, 16 worker threads, and 64 kB direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2023-11-27 20:18:31,016] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2023-11-27 20:18:31,035] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
[2023-11-27 20:18:31,036] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
[2023-11-27 20:18:31,037] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)
[2023-11-27 20:18:31,037] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
[2023-11-27 20:18:31,041] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)
[2023-11-27 20:18:31,053] INFO Reading snapshot E:\app\kafka\kafka_2.12-3.3.1\version-2\snapshot.1c0 (org.apache.zookeeper.server.persistence.FileSnap)
[2023-11-27 20:18:31,067] INFO The digest in the snapshot has digest version of 2, , with zxid as 0x1c0, and digest value as 281925555213 (org.apache.zookeeper.server.DataTree)
[2023-11-27 20:18:31,094] INFO Zookeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
[2023-11-27 20:18:31,100] INFO 1 txns loaded in 10 ms (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-11-27 20:18:31,100] INFO Snapshot loaded in 62 ms, highest zxid is 0x1c1, digest is 276081043263 (org.apache.zookeeper.server.ZKDatabase)
[2023-11-27 20:18:31,100] INFO Snapshotting: 0x1c1 to E:\app\kafka\kafka_2.12-3.3.1\version-2\snapshot.1c1 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-11-27 20:18:31,105] INFO Snapshot taken in 5 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2023-11-27 20:18:31,138] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepRequestProcessor)
[2023-11-27 20:18:31,138] INFO zookeeper.request_throttler.shutdownTimeout = 10000 (org.apache.zookeeper.server.RequestThrottler)
[2023-11-27 20:18:31,179] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)

```

Step 4 : buka cmd ke 2/baru dan running script ini

run buka cmd 2

.\bin\windows\kafka-server-start.bat .\config\server.properties

Berikut dan outputnya

```

C:\Windows\System32\cmd.e x + v
You are screen sharing Stop Share
E:\app\kafka\kafka_2.12-3.3.1\bin\windows\kafka-server-start.bat .\config\server.properties
[2023-11-27 20:20:09,512] INFO Registered Kafka:type=kafka.Log4jController MBean [kafka.utils.Log4jControllerRegistration$]
[2023-11-27 20:20:10,014] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2023-11-27 20:20:10,181] INFO starting [kafka.server.KafkaServer]
[2023-11-27 20:20:10,182] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2023-11-27 20:20:10,212] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2023-11-27 20:20:10,221] INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper.ZooKeeper)
[2023-11-27 20:20:10,221] INFO Client environment:host.name=host.docker.internal (org.apache.zookeeper.ZooKeeper)
[2023-11-27 20:20:10,221] INFO Client environment:java.version=17.0.7 (org.apache.zookeeper.ZooKeeper)
[2023-11-27 20:20:10,221] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2023-11-27 20:20:10,222] INFO Client environment:java.home=C:\Program Files\Java\jdk-17 (org.apache.zookeeper.ZooKeeper)
[2023-11-27 20:20:10,223] INFO Client environment:java.class.path=C:\Program Files\IBM\MQ\java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\MQ\java\lib\com.ibm.mq.jar;E:\app\kafka\kafka_2.12-3.3.1\libs\activation-1.1.1.jar;E:\app\kafka\kafka_

```

b. Integrasi Kafka With Spring Boot : Static Topic/Tabel

Pom.xml

```

<!-- step 1 : kafka-->
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka-test</artifactId>
    <scope>test</scope>

```

```

        </dependency>
        <dependency>
            <groupId>org.springframework.kafka</groupId>
            <artifactId>spring-kafka</artifactId>
        </dependency>
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>2.5.0</version>
</dependency>

```

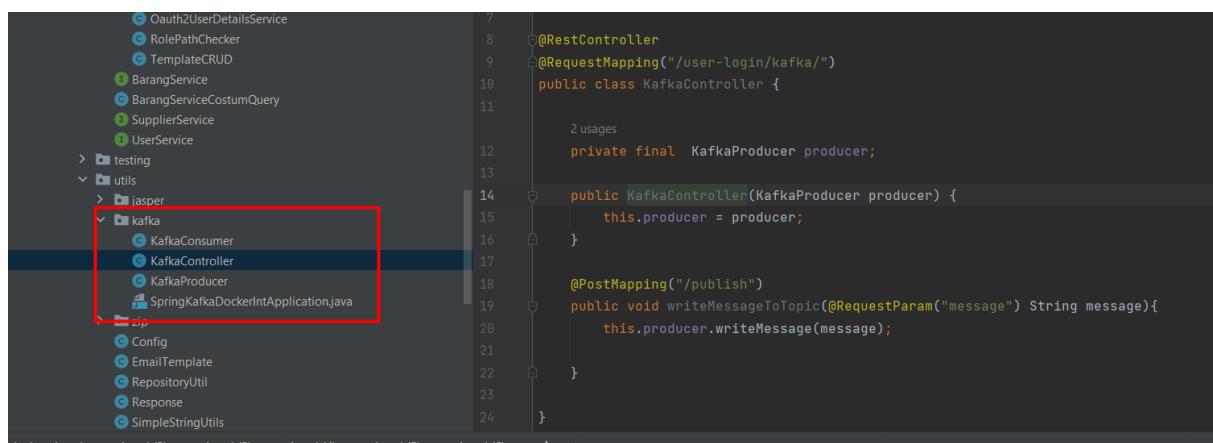
[Application.properties](#)

```

spring.kafka.consumer.bootstrap-servers=localhost:9092
spring.kafka.consumer.group-id="my_group_id"
spring.kafka.consumer.auto-offset-reset=earliest
spring.kafka.consumer.key-
deserializer=org.apache.kafka.common.serialization.StringDeserializer
spring.kafka.consumer.value-deserializer=
org.apache.kafka.common.serialization.StringDeserializer
spring.kafka.producer.bootstrap-servers=localhost:9092
spring.kafka.producer.key-
deserializer=org.apache.kafka.common.serialization.StringDeserializer
spring.kafka.producer.value-
deserializer=org.apache.kafka.common.serialization.StringDeserializer

```

Struktur project



KafkaController

```

package com.binar5.apps.utils.kafka;

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

```

```

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/user-login/kafka/")
public class KafkaController {

    private final KafkaProducer producer;

    public KafkaController(KafkaProducer producer) {
        this.producer = producer;
    }

    @PostMapping("/publish")
    public void writeMessageToTopic(@RequestParam("message") String message) {
        this.producer.writeMessage(message);
    }
}

```

KafkaConsumer

```

package com.binar5.apps.utils.kafka;

import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.kafka.support.KafkaUtils;
import org.springframework.stereotype.Service;

@Service
public class KafkaConsumer {

    @KafkaListener(topics="my_topic", groupId="my_group_id")
    public void getMessage(String message) {
        System.out.println("====");
        System.out.println("Pesan=" + message);
        System.out.println("Group ID =
"+KafkaUtils.getConsumerGroupId());
        // user database kirim
    }
}

```

KafkaProducer

```

package com.binar5.apps.utils.kafka;

import org.springframework.beans.factory.annotation.Autowired;

```

```

import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;

@Service
public class KafkaProducer {

    private static final String TOPIC= "my_topic";

    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;

    public void writeMessage(String msg) {
        this.kafkaTemplate.send(TOPIC, msg);
    }
}

```

Testing with Postman

Pesan yang dikirim sebuah parameter

The screenshot shows the Postman application interface. A POST request is being made to the URL `http://localhost:8080/api/user-login/kafka/publish?message=rumah_makan22ppp`. The 'Headers' tab is selected, displaying the following configuration:

Header	Value
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.2
Accept	/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Key	Value

Branch

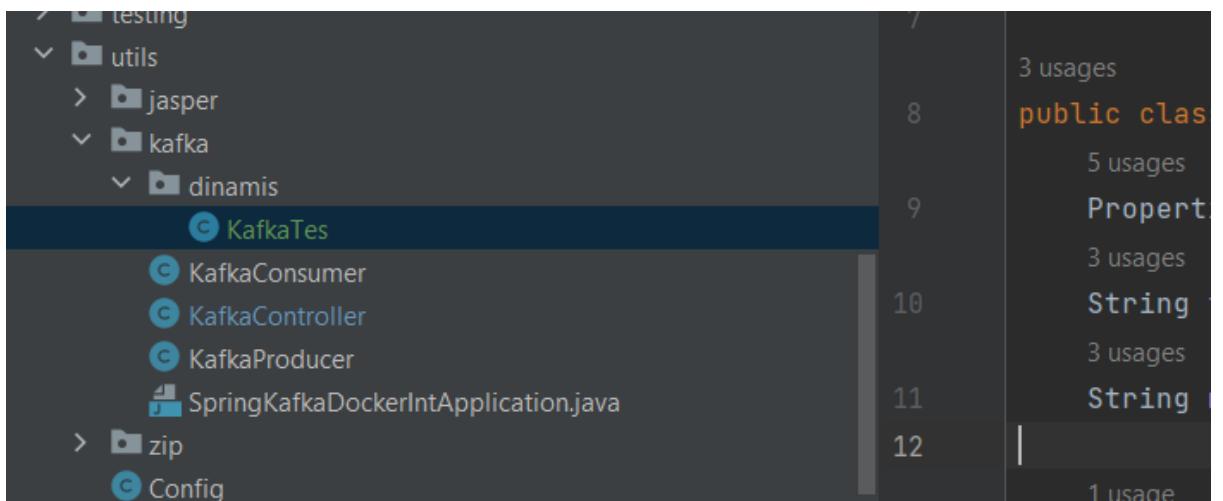
[081222-authority-user](#)

https://github.com/rikialdi/binar_batch_5.git

c. Integrasi Kafka With Spring Boot : Dinamic Topik/Tabel

Lakukan semua config tahap sebelumnya seperti application.properties dan pom.xml

Struktur project



Controller Kafka

```
....  
@PostMapping("/publish22")  
public void writeMessageToTopic2(@RequestParam("topic") String topic,  
@RequestParam("message") String message){  
    KafkaTes prod = new KafkaTes(topic, message);  
    prod.kirim();  
}  
....
```

KafkaTes

```
package com.binar5.apps.utils.kafka.dinamis;  
  
import org.apache.kafka.clients.producer.KafkaProducer;  
import org.apache.kafka.clients.producer.ProducerRecord;  
  
import java.util.Properties;  
  
public class KafkaTes {  
    Properties props = new Properties();  
    String topic = "my-topic";  
    String message = "Hello, Kafka!";  
  
    public KafkaTes(String topic, String message ) {  
  
        this.topic = topic;  
        this.message = message;  
    };
```

```
public void kirim(){
    props = new Properties();
    props.put("bootstrap.servers",
"localhost:9092,localhost:9093,localhost:9094");
    props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
    props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
    KafkaProducer<String, String> producer = new
KafkaProducer<>(props);

    ProducerRecord<String, String> record = new
ProducerRecord<>(topic, message);

    producer.send(record);
    System.out.println("topik saya :" + topic + message);
}

}
```

Branch

[081222-authority-user](#)

https://github.com/rikialdi/binar_batch_5.git

<https://gitlab.com/aryamuhajir8/kafkabinar/-/tree/master>

<https://www.baeldung.com/kafka-topic-creation>

40. Kafka contoh - github

<https://gitlab.com/synrgy5-bej2/frontliner>

atau di E:\binar\Binat Batch 6\MODUL\messagebroker\frontliner

41. Kafka Dynamic Db Consumer

<https://medium.com/bliblidotcom-techblog/dynamic-spring-boot-kafka-consumer-af8740f2c703>

d. kafka finamic with db

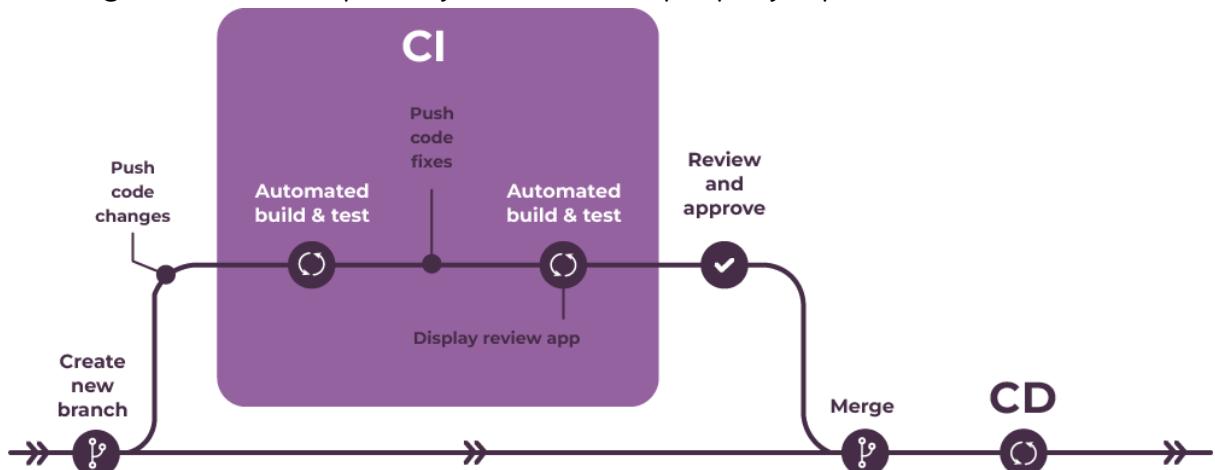
E:\binar\Binat Batch 6\MODUL

MEMBUAT_KAFKA_MENJADI_LEBIH_DINAMIK_DENGAN_MENGGUNAKAN_DATABASE

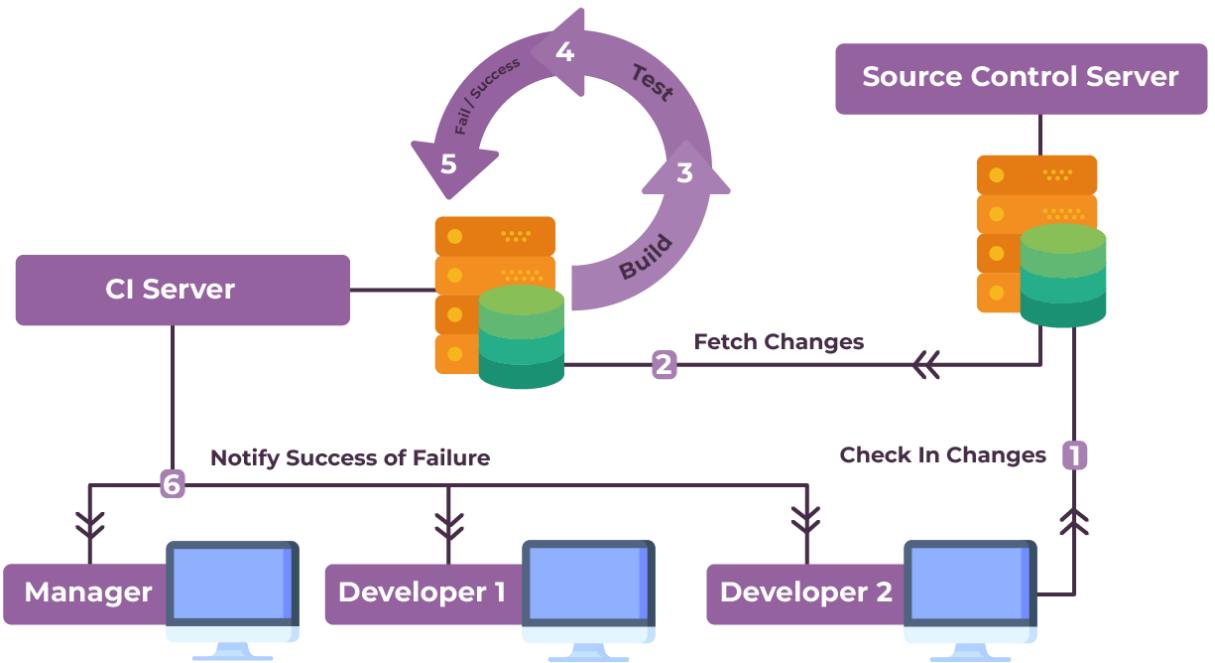
42. CI-CD

a. CI?

Continuous Integration atau CI adalah proses pengujian dan pembuatan software secara otomatis. Dilakukan setelah kode aplikasi yang baru, sudah terintegrasi ke dalam repository bersama (tempat penyimpanan).



Begini cara kerja dari sebuah CI-

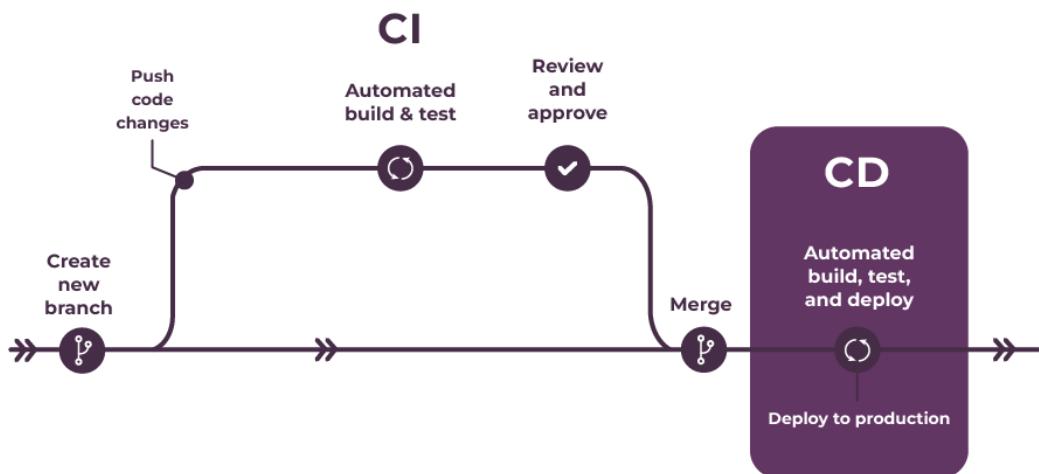


1. Developer melakukan commit ke repository.
2. Setelah itu, remote branch bakal menerima changes tersebut.
3. Terus remote branch menjalankan job build.
4. Setelah itu, job bakal dijalankan buat mengeksekusi seluruh unit test. Kalau masih ada unit test yang failed, maka job bakal gagal dan nggak dilanjutkan ke tahap selanjutnya.
5. Terus di cek apakah prosesnya udah berhasil atau malah gagal.
6. Terakhir, user bakal menerima notifikasi apakah job dari CI berhasil dijalankan atau nggak.

b. CD

CD atau Continuous Deployment adalah **proses delivery aplikasi yang dibuat dalam proses CI ke bagian lingkungan produksi, yang dimasukkan melalui automated test.**

Jadi nih, setelah changes yang di-commit berhasil lolos di semua tahap CI, berarti si aplikasi siap buat di-deploy kapan aja. Inti dari CD adalah **mempermudah dan mempersingkat proses deployment**.



Tanpa CD, proses deployment harus dijalankan secara manual, lho 😞

Iya, misalnya gini di suatu server harus terinstall dulu JDK dan Maven supaya bisa melakukan compile dan menjalankan aplikasi Java.

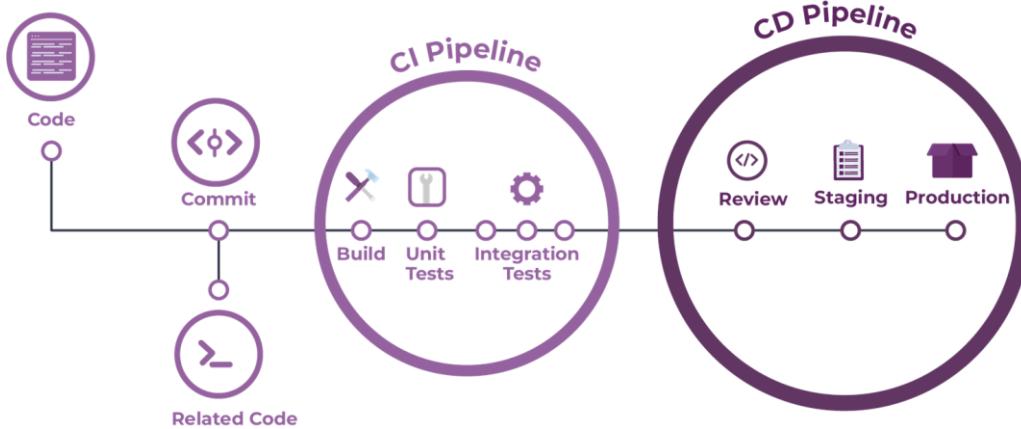
Terus, dari code yang udah dibikin, kita harus membentuk file JAR atau WAR yang nantinya bakal dipindahkan ke server.

Setelah server menerima file JAR atau WAR tersebut, maka bakal dijalankan pakai Java Argument.

c. Kenapa perlu sebuah CI/CD?

Karena CI/CD memungkinkan distribusi kode yang lebih sering. CD secara otomatis mendistribusikan aplikasi dan menjalankan unit test yang udah ditambahkan.

Pipeline CI/CD dirancang untuk tim yang perlu perubahan cukup sering di aplikasi, dengan proses delivery yang handal.



d. ISTILAH CI-CD

- **pipelines**
- **jobs**
- **stages**

Pipeline merupakan komponen tertinggi dari *continuous integration, delivery, dan deployment*. Pipeline terdiri dari:

- **Job** (id: tugas), yang menentukan **apa yang dilakukan**. Contohnya tugas yang mengompilasi code, atau melakukan test code.
- **Stages** (id: tahap), yang menentukan **kapan job dilakukan**. Contohnya, tahap yang menjalankan test setelah tahap kompilasi code dijalankan.

Tugas dieksekusi oleh Runner. Beberapa tugas yang pada tahap yang sama dieksekusi secara paralel, jika tersedia cukup runner untuk menjalankan tugas secara bersamaan (baca: concurrent).

Jika semua tugas pada sebuah tahap:

- **Sukses**, pipeline berlanjut ke tahap berikutnya.
- **Gagal**, tahap berikutnya (biasanya tidak) dieksekusi dan pipeline terhenti.

Pada umumnya, pipeline dieksekusi secara otomatis dan tidak memerlukan intervensi ketika dibuat. Akan tetapi, ada saat di mana Anda dapat secara manual berinteraksi dengan pipeline.

Biasanya, sebuah pipeline berisi empat tahap, dan dieksekusi pada urutan berikut:

- Tahap `build`, dengan tugas bernama `compile`.
- Tahap `test`, dengan tugas bernama `test1` dan `test2`.
- Tahap `staging`, dengan tugas bernama `deploy-ke-staging`
- Tahap `production`, dengan tugas bernama `deploy-ke-production`

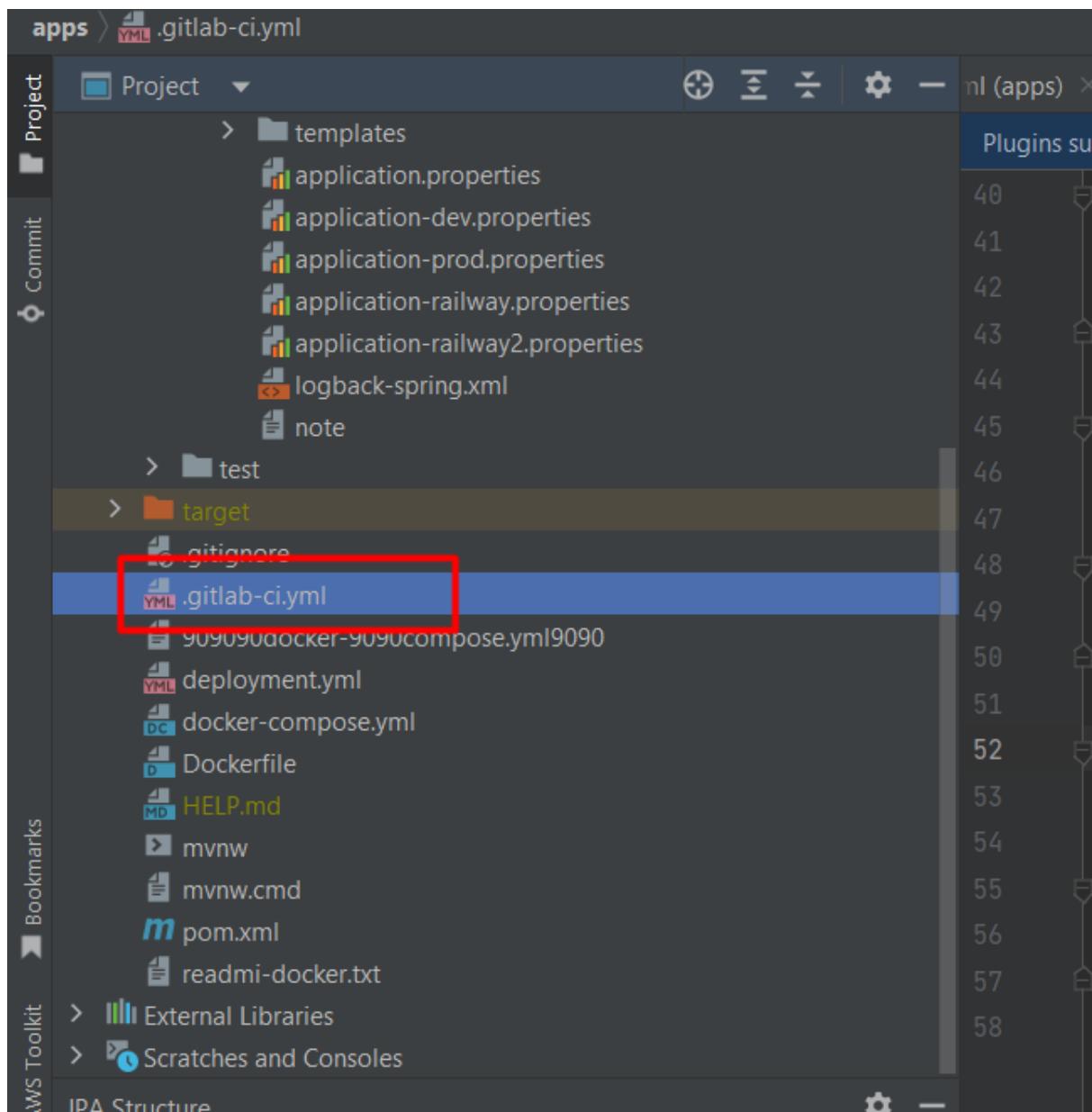


e. Contoh

Harus register dan push project di gitlab

Buat file name dengan nama :

.gitlab-ci.yml



lsl file

```
# This file is a template, and might need editing before it works on
your project.
# This is a sample GitLab CI/CD configuration file that should run
without any modifications.
# It demonstrates a basic 3 stage CI/CD pipeline. Instead of real
tests or scripts,
# it uses echo commands to simulate the pipeline execution.
```

```

# A pipeline is composed of independent jobs that run scripts,
grouped into stages.
# Stages run in sequential order, but jobs within stages run in
parallel.
#
# For more information, see:
https://docs.gitlab.com/ee/ci/yaml/index.html#stages
#
# You can copy and paste this template into a new `*.gitlab-ci.yml` file.
# You should not add this template to an existing `*.gitlab-ci.yml` file by using the `include:` keyword.
#
# To contribute improvements to CI/CD templates, please follow the Development guide at:
# https://docs.gitlab.com/ee/development/cicd/templates.html
# This specific template is located at:
# https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Getting-Started.gitlab-ci.yml

stages:          # List of stages for jobs, and their order of execution
  - build
  - test
  - staging
  - production

build-job:        # This job runs in the build stage, which runs first.
  stage: build
  script:
    - echo "Compiling the code..."
    - echo "Compile complete."

unit-test-job:   # This job runs in the test stage.
  stage: test    # It only starts when the job in the build stage completes successfully.
  script:
    - echo "Running unit tests... This will take about 60 seconds."
    - sleep 60
    - echo "Code coverage is 90%"

lint-test-job:   # This job also runs in the test stage.
  stage: test    # It can run at the same time as unit-test-job (in parallel).
  script:
    - echo "Linting code... This will take about 10 seconds."
    - sleep 10
    - echo "No lint issues found."

my-staging-job:  # This job runs in the deploy stage.
  stage: staging # It only runs when *both* jobs in the test stage complete successfully.
  environment: staging3
  script:
    - echo "Deploying application..."
    - echo "Application successfully mystaging."

```

```

my-production-job:      # This job runs in the production stage.
  stage: production  # It only runs when *both* jobs in the test
  stage complete successfully.
  environment: production
  script:
    - echo "myproduction application..."
    - echo "Application successfully myproduction."

```

output

The screenshot shows a CI/CD pipeline status page for a project named 'binar5-spring'. The left sidebar has sections for Project information, Repository, Issues, Merge requests, CI/CD (with Pipelines selected), Editor, Jobs, Schedules, Security & Compliance, Deployments, and Packages and registries. The main area shows a table with columns: Status, Pipeline, Triggerer, and Stages. There are two rows:

- Passed Pipeline:** Status is 'passed' (highlighted with a red box). Pipeline ID is '#708997198'. Triggered by 'fe8045fd' at 00:01:53 just now. Stages: build, test, staging, production, all marked as green.
- Failed Pipeline:** Status is 'failed' (highlighted with a red box). Pipeline ID is '#708994329'. Triggered by '17878fa9' at 5 minutes ago. Stages: build, test, staging, production, all marked as green, but there is a note 'yaml invalid error'.

Output pipeline

The screenshot shows a detailed view of a pipeline. The left sidebar includes sections for Merge requests (0), CI/CD (with Pipelines selected), Editor, Jobs, Schedules, Security & Compliance, Deployments, Packages and registries, and Infrastructure. The main area displays a pipeline structure with stages: build, test, staging, and production. Each stage contains a list of jobs, all of which are marked as green (passed). A specific job in the test stage is highlighted with a red box and labeled 'unit-test-job - passed'.

Output jobs

The screenshot shows a CI/CD pipeline interface with a sidebar on the left and a main table on the right. The sidebar includes sections for Issues (0), Merge requests (0), CI/CD (selected), Pipelines (Editor, Jobs, Schedules), Security & Compliance, Deployments, Packages and registries, Infrastructure, Monitor, Analytics, Wiki, and Snippets. The main table has tabs for Pipeline, Needs, Jobs (5, highlighted with a red box), and Tests (0). The table columns are Status, Job, Stage, Name, Duration, and Coverage. Each row shows a green 'passed' status icon. The data rows are:

Status	Job	Stage	Name	Duration	Coverage
passed	#3399601622 Y 301122-forget-... ⇨ feb045fd	production	my-production-job	0:00:11 2 minutes ago	
passed	#3399601621 Y 301122-forget-... ⇨ feb045fd	staging	my-staging-job	0:00:11 2 minutes ago	
passed	#3399601618 Y 301122-forget-... ⇨ feb045fd	test	lint-test-job	0:00:23 3 minutes ago	
passed	#3399601614 Y 301122-forget-... ⇨ feb045fd	test	unit-test-job	0:00:15 2 minutes ago	
passed	#3399601608 Y 301122-forget-... ⇨ feb045fd	build	build-job	0:00:14 3 minutes ago	

43.Jenkin

a. Download

<https://www.jenkins.io/download/>

pilih

4. You may also want to verify the package you downloaded. Learn more

⬇ Download Jenkins 2.361.4 LTS for:

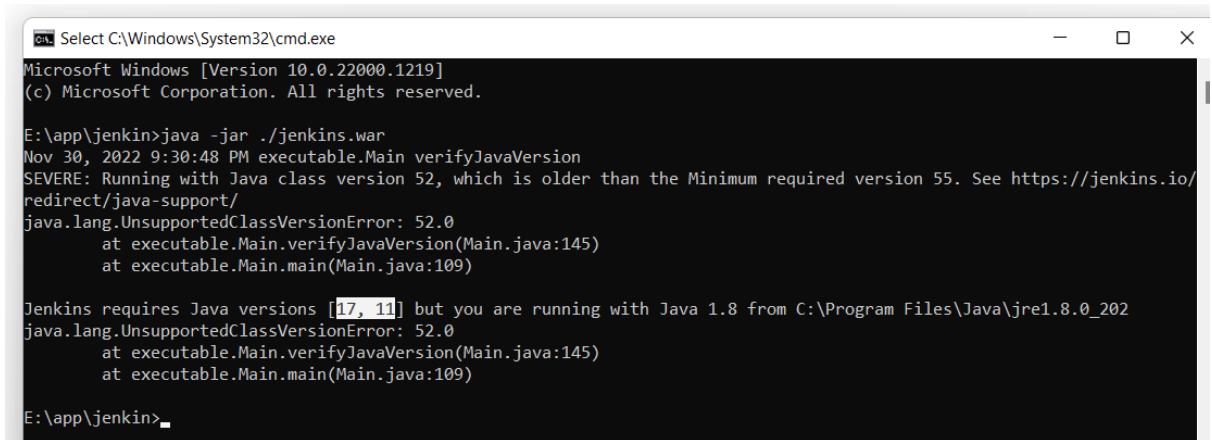
Generic Java package (.war)

SHA-256: b38fe218afb5447b0c9a6fa308d7ab762ac5a58dd89aa68b735067ad6c37c17b

b. RUN JENkIN windows

java -jar ./Jenkins.war

support : JDK : 11 dan 17



```
c:\ Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

E:\app\jenkin>java -jar ./jenkins.war
Nov 30, 2022 9:30:48 PM executable.Main verifyJavaVersion
SEVERE: Running with Java class version 52, which is older than the Minimum required version 55. See https://jenkins.io/redirect/java-support/
java.lang.UnsupportedClassVersionError: 52.0
        at executable.Main.verifyJavaVersion(Main.java:145)
        at executable.Main.main(Main.java:109)

Jenkins requires Java versions [17, 11] but you are running with Java 1.8 from C:\Program Files\Java\jre1.8.0_202
java.lang.UnsupportedClassVersionError: 52.0
        at executable.Main.verifyJavaVersion(Main.java:145)
        at executable.Main.main(Main.java:109)

E:\app\jenkin>
```

44. Cara Akses Website Diblokir Telkomsel

<https://aksesbebas.com/>

45. Socket Spring Boot/Example

a. Reference with Socket Stomp

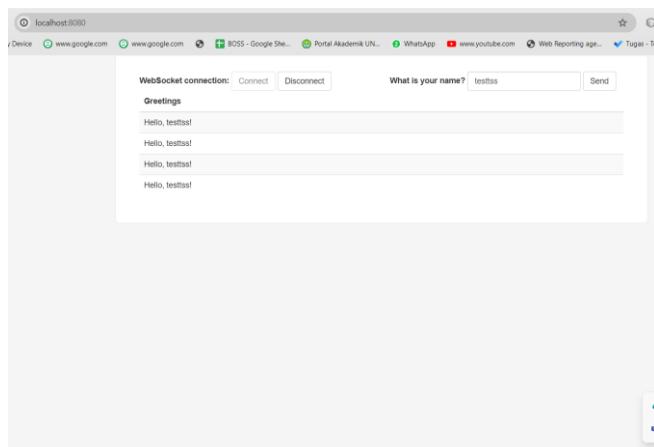
url

<https://spring.io/guides/gs/messaging-stomp-websocket/>

project

E:\binar\Binat Batch 6\MODUL\Socket\gs-messaging-stomp-websocket\complete

output



b. reference Socket io

Socket IO With Springboot Application

<https://medium.com/geekculture/socket-io-with-springboot-application-5a59358f7d48>

git

git clone <https://github.com/smadi997/spring-netty-socketio-seed.git>

project

E:\binar\Binat Batch 6\MODUL\Socket\spring-netty-socketio-seed

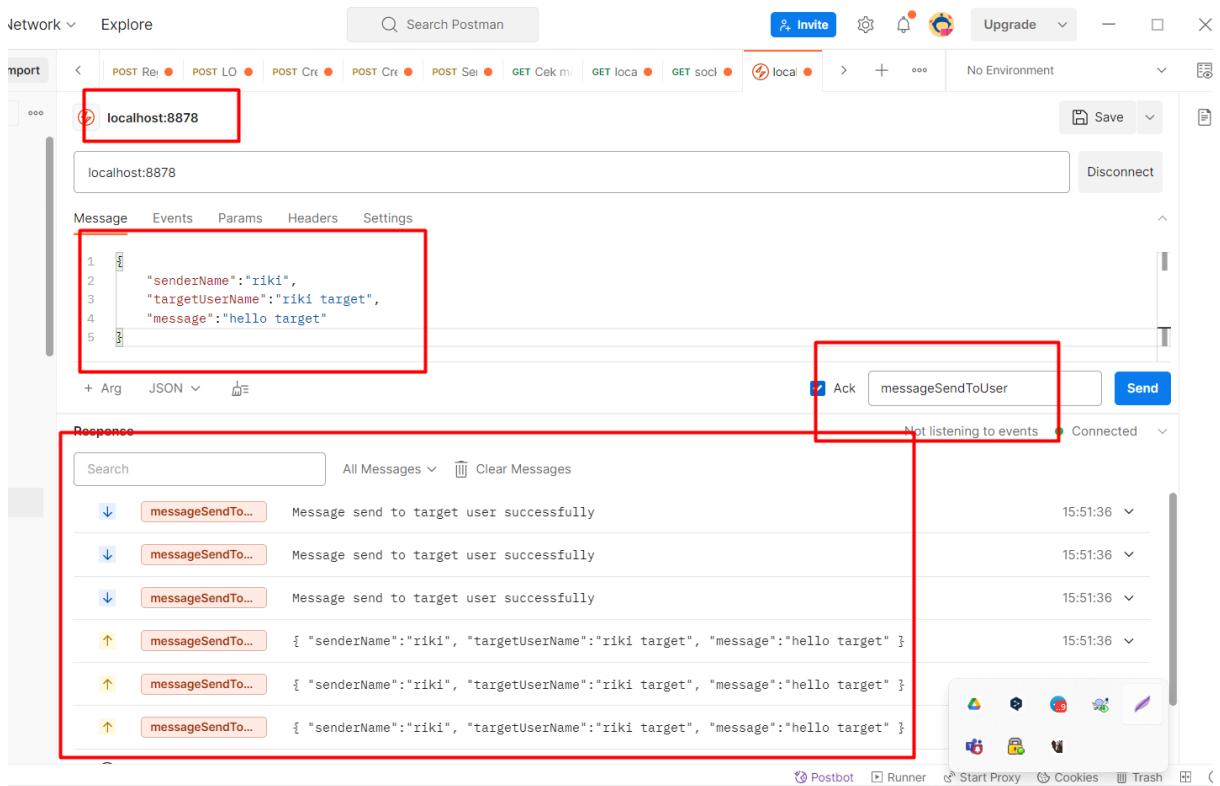
Update Pom XML versi Socket io

```
<dependency>
    <groupId>com.corundumstudio.socketio</groupId>
    <artifactId>netty-socketio</artifactId>
    <version>2.0.6</version>
</dependency>
```

Output

Test in postman – step 1

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Karyawan', 'Karyawan BINAR Batch 6', and 'Socket'. A red box highlights the 'New' button at the top of the sidebar. The main workspace shows a collection named 'heybook / CHAT MESSAGE / Cek member group' with a GET request. A red box highlights the 'Socket.IO' icon in the bottom right corner of the request panel. The interface includes various API icons (HTTP, GraphQL, gRPC, WebSocket, MQTT, Collection, Environment, API, Flows) and a 'Workspace' section.



46. Web socket react js dan java spring boot : Socket stomp

<https://github.com/Thibaut-Mouton/react-spring-messenger-project>

how tu run ?

a. Clone

<https://github.com/Thibaut-Mouton/react-spring-messenger-project>

```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

E:\binar\Binat Batch 6\MODUL\Socket>git clone https://github.com/Thibaut-Mouton/react-spring-messenger-project
Cloning into 'react-spring-messenger-project'...
remote: Enumerating objects: 707, done.
remote: Counting objects: 100% (667/667), done.
remote: Compressing objects: 100% (403/403), done.
remote: Total 707 (delta 290), reused 543 (delta 215), pack-reused 40
Receiving objects: 97% (686/707), 932.00 KiB | 914.00 KiB/s
Receiving objects: 100% (707/707), 1.39 MiB | 1.08 MiB/s, done.
Resolving deltas: 100% (292/292), done.
```

b. Running BE

c. Instal npm

<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

d. Running FE

Open cmd

Name	Date modified	Type	Size
public	12/5/2023 10:43 AM	File folder	
src	12/5/2023 10:43 AM	File folder	
.eslintignore	12/5/2023 10:43 AM	ESLINTIGNORE File	1 KB
.eslintrc.json	12/5/2023 10:43 AM	JSON Source File	1 KB
package.json	12/5/2023 10:43 AM	JSON Source File	2 KB

```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

E:\binar\Binat Batch 6\MODUL\Socket\react-spring-messenger-project\frontend-web>
```

Ketikkan

“Npm install”

Tunggu beberapa saat s+, sampai proses selesai.

```
tend-web>npm install
```

Jika eror, maka ketikkan .

```
!r npm audit fix
npm WARN deprecated babel-eslint@10.1.0: babel-eslint is now @babel/eslint-parser. This package will no longer
pdates.
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random()
circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated @hapi/hoek@8.5.1: This version has been deprecated and is no longer supported or maintained.
npm WARN deprecated @hapi/joi@15.1.1: Switch to 'npm install joi'
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.
npm WARN deprecated core-js@2.6.12: core-js@<3.23.3 is no longer maintained and not recommended for usage due
ber of issues. Because of the V8 engine whims, feature detection in old core-js versions could cause a slowdown
0x even if nothing is polyfilled. Some versions have web compatibility issues. Please, upgrade your dependency
actual version of core-js.

added 2082 packages, and audited 2083 packages in 7m

220 packages are looking for funding
  run 'npm fund' for details

109 vulnerabilities (1 low, 81 moderate, 18 high, 9 critical)

To address issues that do not require attention, run:
  npm audit fix

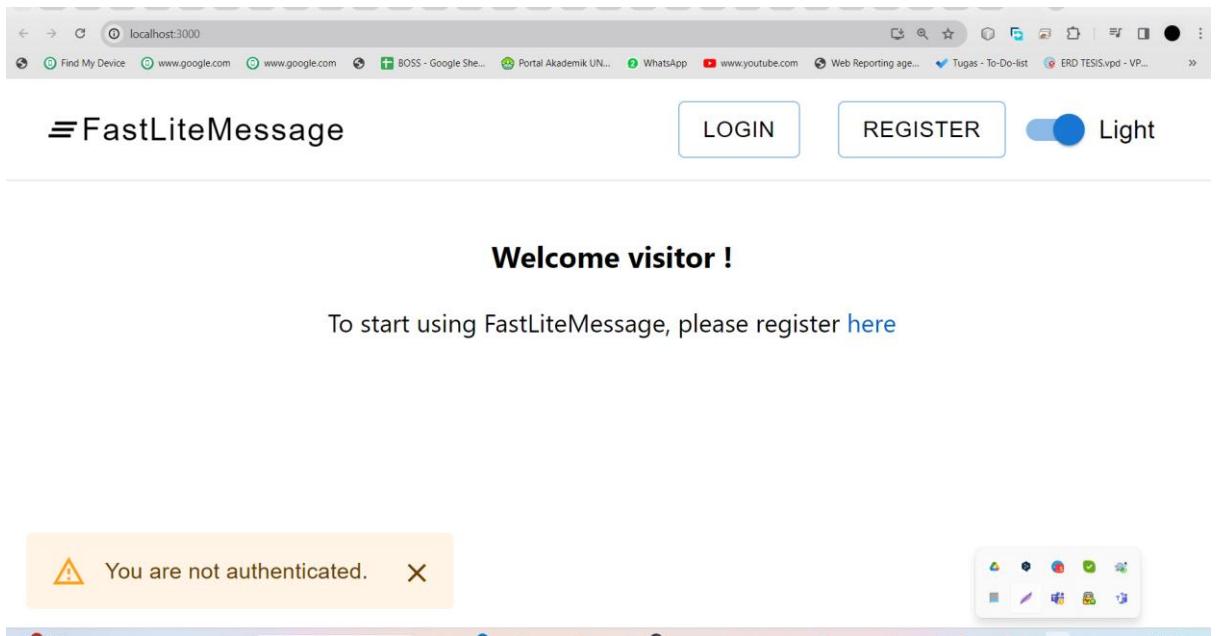
To address all issues (including breaking changes), run:
  npm audit fix --force
Run 'npm audit' for details.

E:\binar\Binat Batch 6\MODUL\Socket\react-spring-messenger-project\frontend-web>npm audit fix --force
npm WARN using --force Recommended protections disabled.
[          ] / audit: timing arborist:ctor Completed in 0ms
```

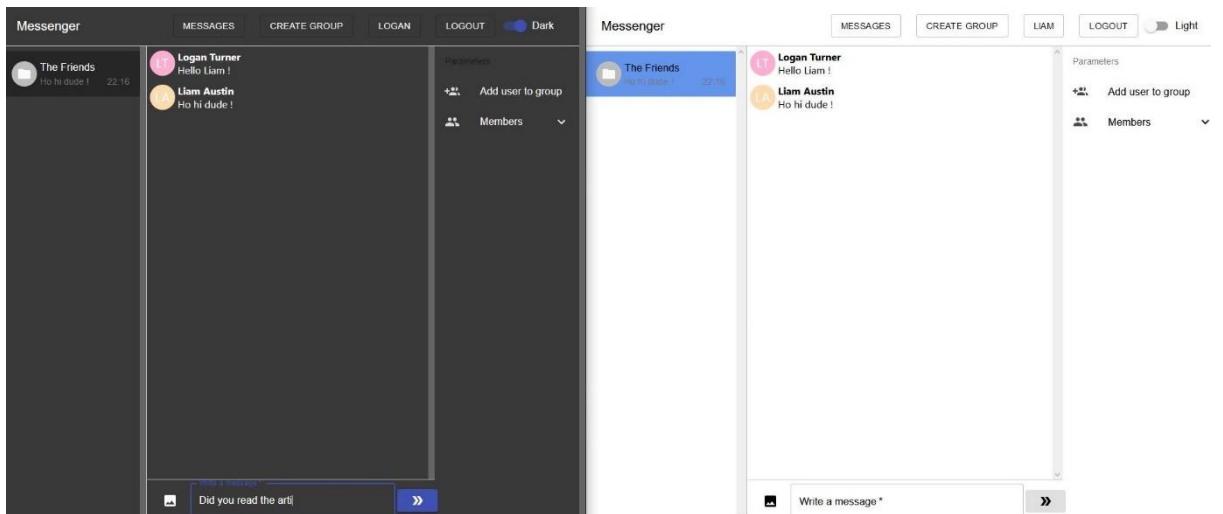
Run fe

Npm start

Tampilan setelah run



Testing



47.Ngrox

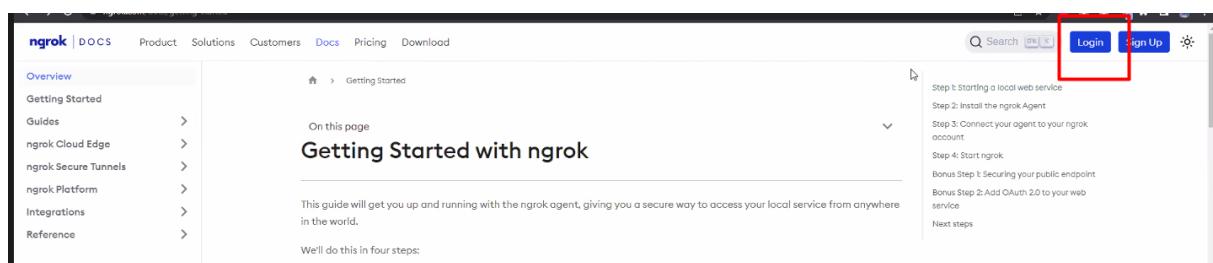
a. pengertian

Ngrok adalah proxy server untuk membuat atau membuka jaringan private melalui NAT atau firewall, lalu Menghubungkan localhost ke internet dengan tunnel yang aman. Atau bahasa gampangnya ngrok adalah layanan gratis yang memberikan kemampuan kepada aplikasi kita agar bisa diakses online.

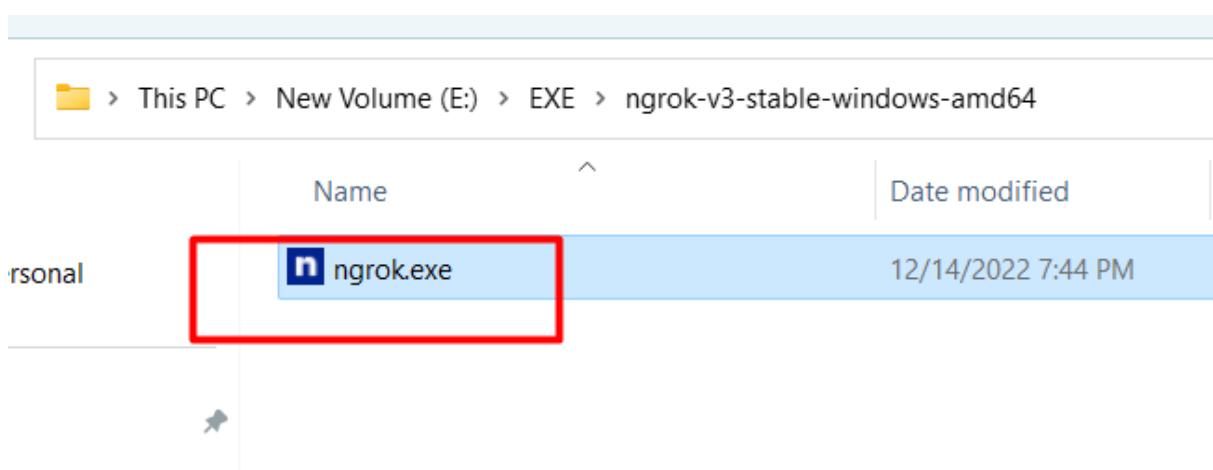
b. Download dan register

<https://dashboard.ngrok.com/get-started/setup>

login dulu



Kemudian download : tampilan setelah download

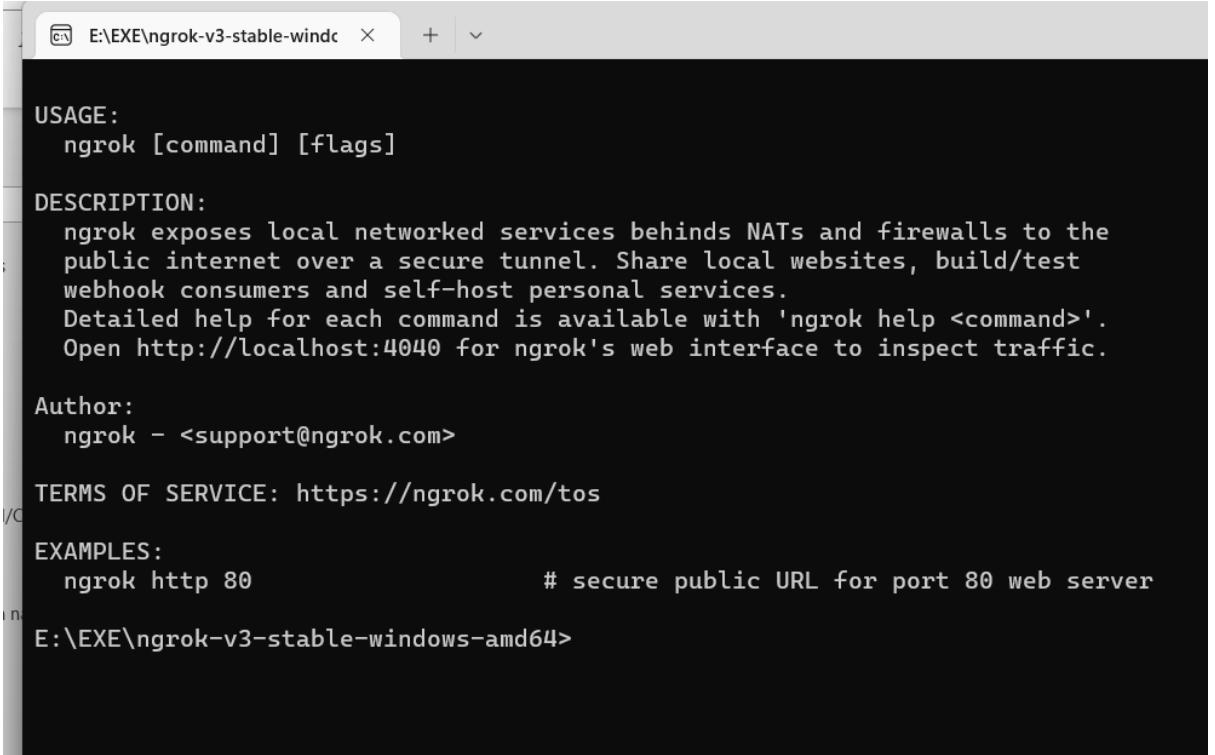


c. Klik 2 Kali file ngrok.exe

<https://ngrok.com/docs/getting-started>

klik 2 kali ngrok.exe

tampilan



```
E:\EXE\ngrok-v3-stable-windows-amd64> ngrok --help
USAGE:
  ngrok [command] [flags]

DESCRIPTION:
  ngrok exposes local networked services behind NATs and firewalls to the
  public internet over a secure tunnel. Share local websites, build/test
  webhook consumers and self-host personal services.
  Detailed help for each command is available with 'ngrok help <command>'.
  Open http://localhost:4040 for ngrok's web interface to inspect traffic.

Author:
  ngrok - <support@ngrok.com>

TERMS OF SERVICE: https://ngrok.com/tos

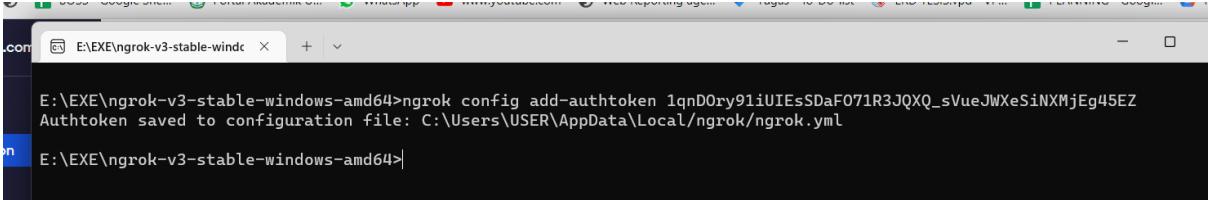
EXAMPLES:
  ngrok http 80                                # secure public URL for port 80 web server
E:\EXE\ngrok-v3-stable-windows-amd64>
```

d. Connect your account

Lakukan login dengan email atau github

```
ngrok config add-authtoken
1qnD0ry91iUIEsSDaF071R3JQXQ_sVueJWXeSiNXMjEg45EZ
```

Output

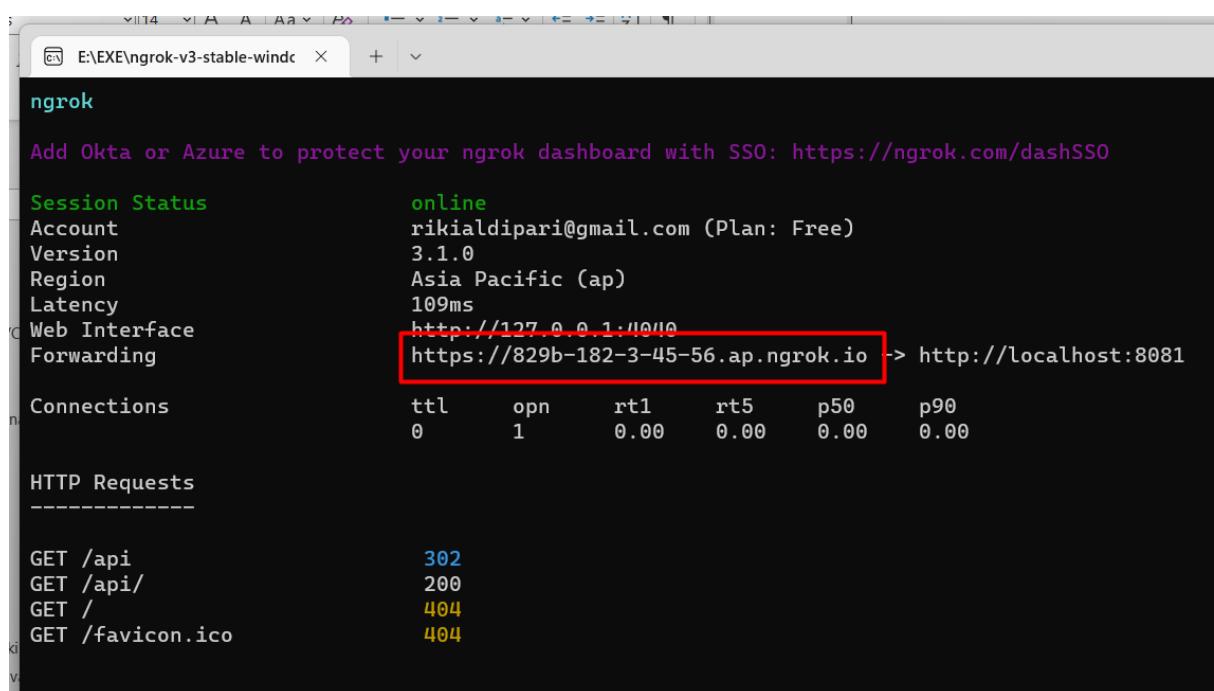


```
E:\EXE\ngrok-v3-stable-windows-amd64>ngrok config add-authtoken 1qnD0ry91iUIEsSDaF071R3JQXQ_sVueJWXeSiNXMjEg45EZ
Authtoken saved to configuration file: C:\Users\USER\AppData\Local/ngrok/ngrok.yml
E:\EXE\ngrok-v3-stable-windows-amd64>
```

e. run

```
ngrok http 8081
```

f. output



```
E:\EXE\ngrok-v3-stable-win32> ngrok

Add Okta or Azure to protect your ngrok dashboard with SSO: https://ngrok.com/dashSSO

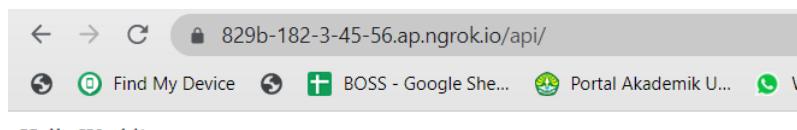
Session Status          online
Account                  rikialdipari@gmail.com (Plan: Free)
Version                 3.1.0
Region                  Asia Pacific (ap)
Latency                109ms
Web Interface           http://127.0.0.1:1010
Forwarding              https://829b-182-3-45-56.ap.ngrok.io -> http://localhost:8081

Connections             ttl     opn      rt1      rt5      p50      p90
                         0       1      0.00    0.00    0.00    0.00

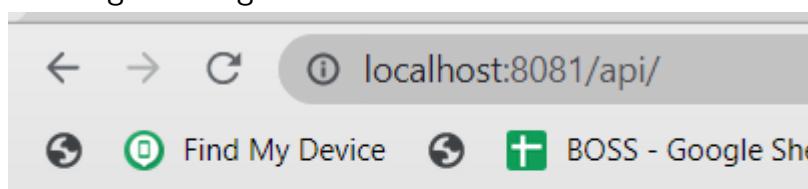
HTTP Requests
-----
GET /api                 302
GET /api/                200
GET /                   404
GET /favicon.ico         404
```

g. Testing di chrome

Copi paste url tersebut



Bandingkan dengan localhost:8081



Hello World!

h. Kesimpulan

Ngrok dapat membuat localhost menjadi bisa akses dari internet.

48.Jmeter

a. Pengertian

Performance Test Menggunakan Jmeter.

Unit testing ini merupakan bagian dari white box testing karena testing ini dieksekusi langsung oleh programmer. Sedangkan untuk black box testing bisa dilakukan menggunakan aplikasi sejenis Fiddler atau menggunakan add-ons yang dimiliki oleh browser-browser seperti Mozilla Firefox dan Google Chrome.

Karena service ini nantinya akan digunakan oleh banyak client dan user, tentu akan ngeri-ngeri sedap, jika testingnya masih dilakukan secara manual. Jadi sebaiknya kita menggunakan Performance Test Tool untuk proses ini.

The Apache JMeter™ application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions

b. Download

https://jmeter.apache.org/download_jmeter.cgi

donwload

Apacne JMeter 5.5 (Requires Java 8+)

Binaries

[apache-jmeter-5.5.tgz sha512 pgp](#)
[apache-jmeter-5.5.zip sha512 pgp](#)

Source

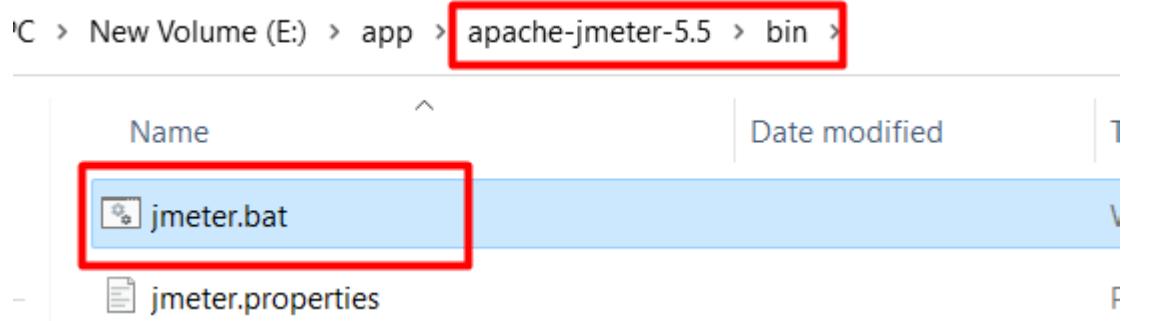
[apache-jmeter-5.5_src.tgz sha512 pgp](#)
[apache-jmeter-5.5_src.zip sha512 pgp](#)

exkak file

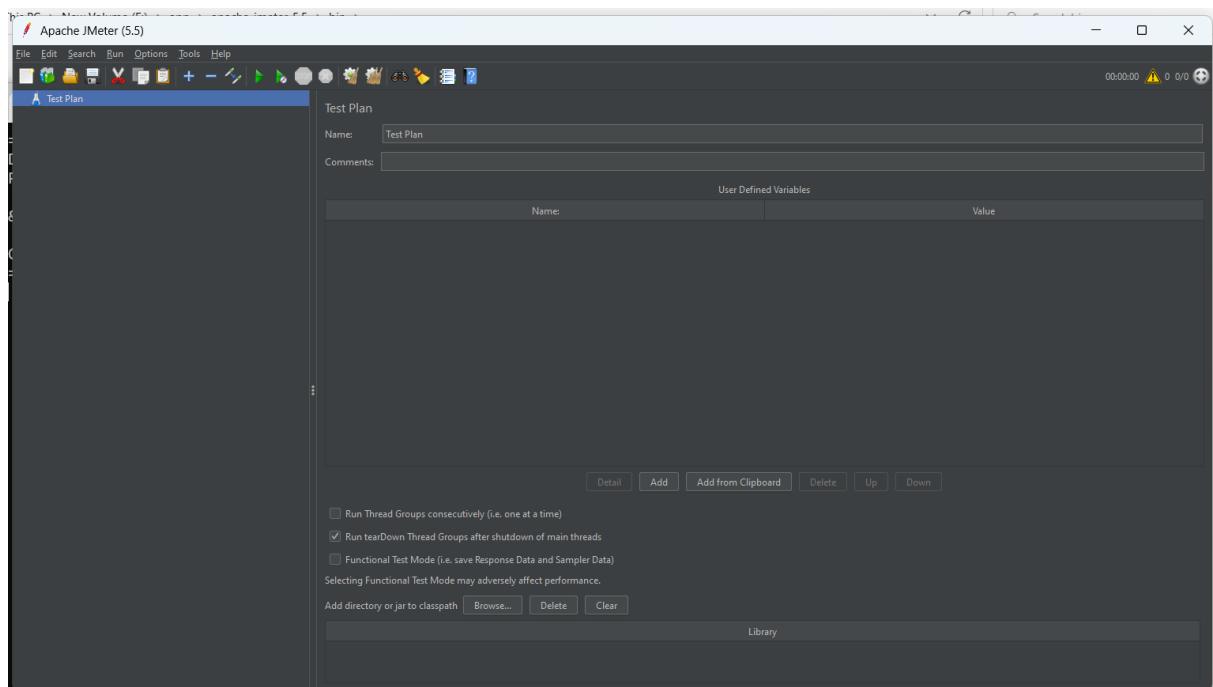
This PC > New Volume (E) > app		
	Name	Date modified
optional	apache-jmeter-5.5	12/15/2022 9:35 AM
	jenkin	11/30/2022 5:30 PM
	kafka	12/5/2022 3:30 PM
	apache-jmeter-5.5.tgz	12/15/2022 9:35 AM

c. Run Jmeter

Untuk menjalankan [JMeter](#), klik ganda file jmeter.bat yang ada di folder bin



Tunggu beberapa saat sampai tampil, aplikasi [JMeter](#)



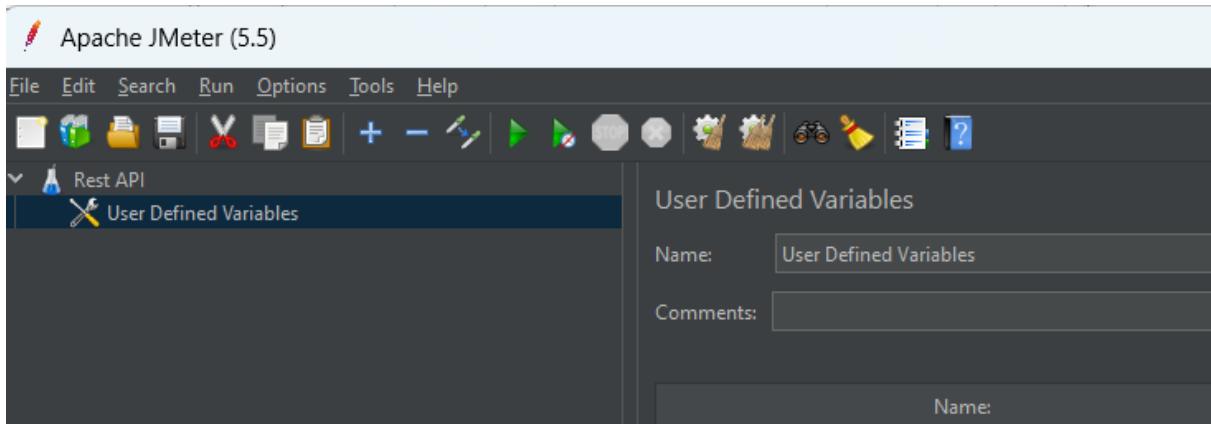
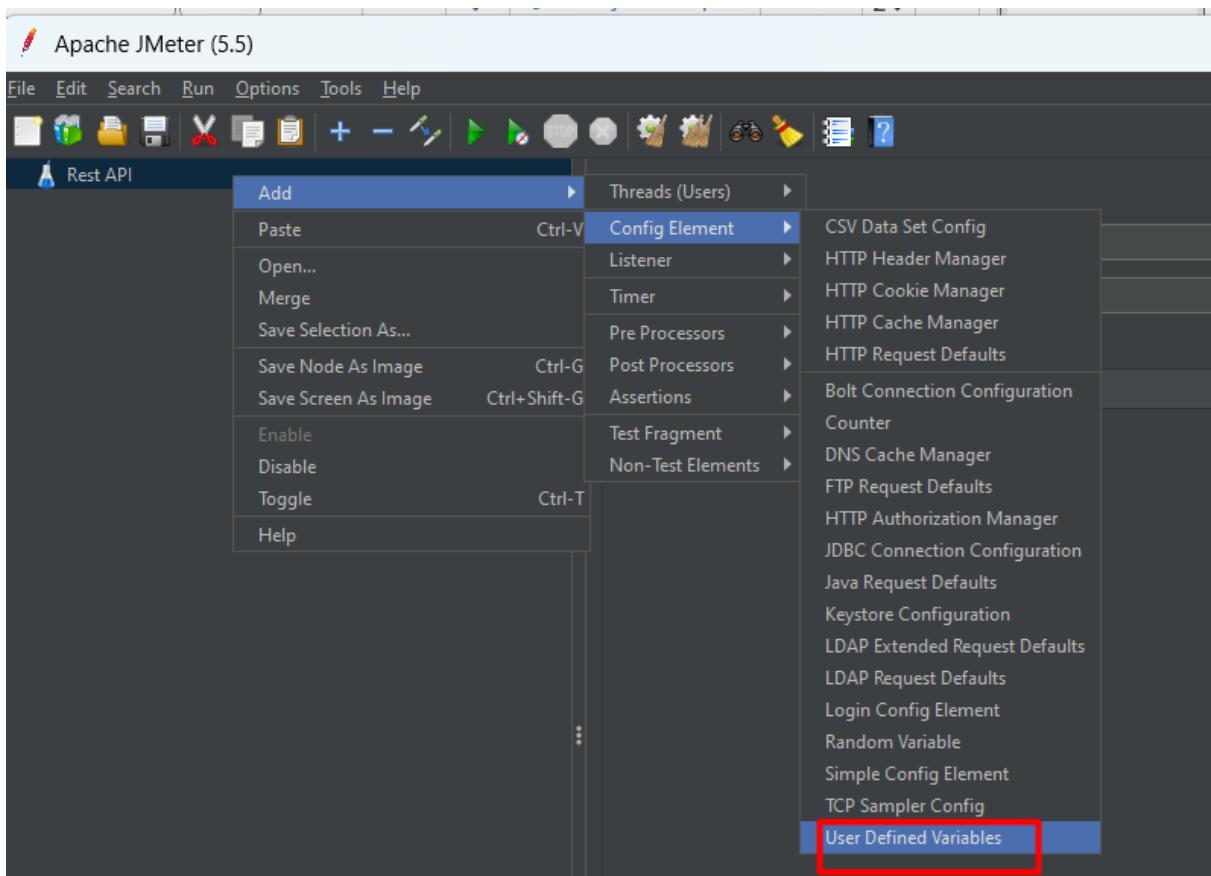
Kemudian ganti informasi Name, sesuai kebutuhan. Misal Performance Test Rest API

d. Menambahkan User Defined Variables

Di node ini kita akan menambahkan informasi global yang sering digunakan pada saat testing seperti informasi host dan port.

Add user defined variables

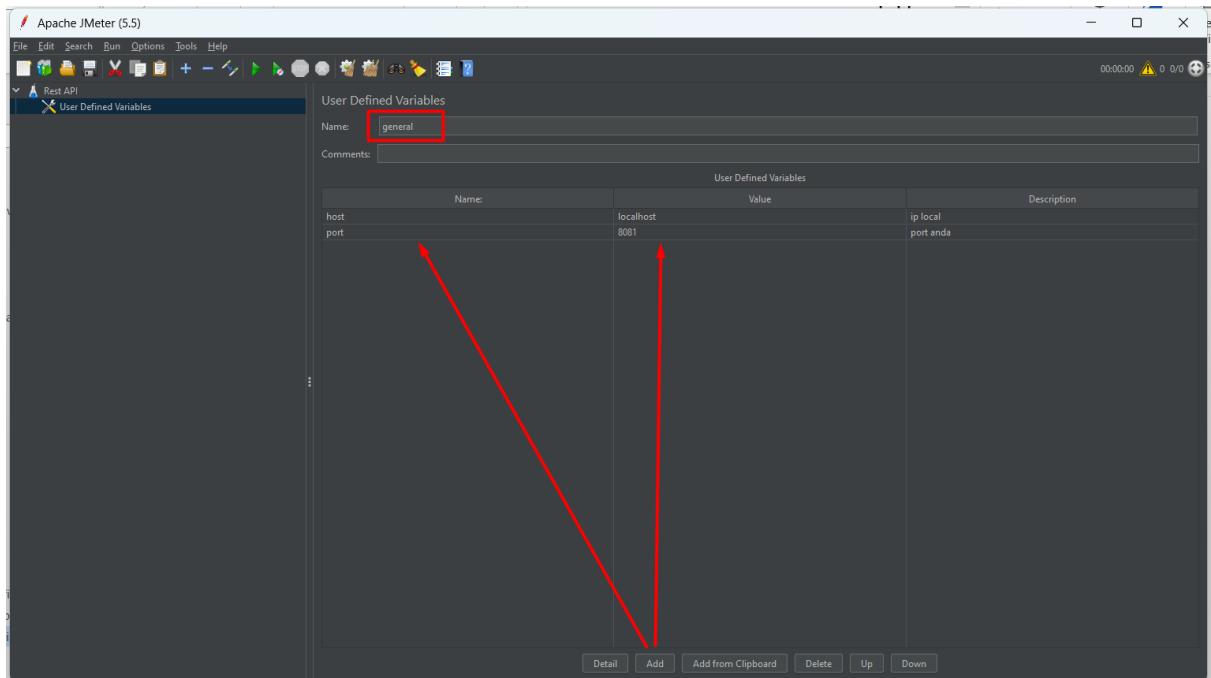
Untuk menambahkan node User Defined Variables klik kanan node Test Plan (Performance Test Rest API) -> Add -> Config Element -> User Defined Variables.



Config user defined variables

Kemudian lakukan pengaturan user defined variables seperti gambar berikut :

Tambahkan ip dan port yang digunakan

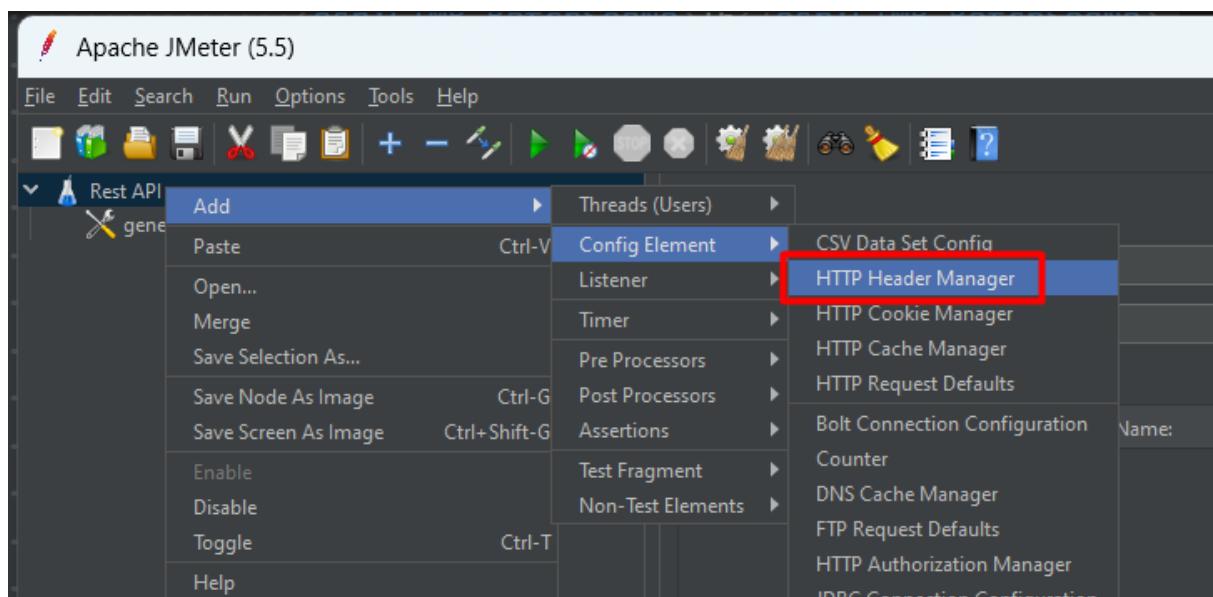


Pada gambar di atas kita menambahkan dua variabel yaitu `host` dan `port`. Nah variabel-variabel ini nantinya akan digunakan pada langkah-langkah berikutnya.

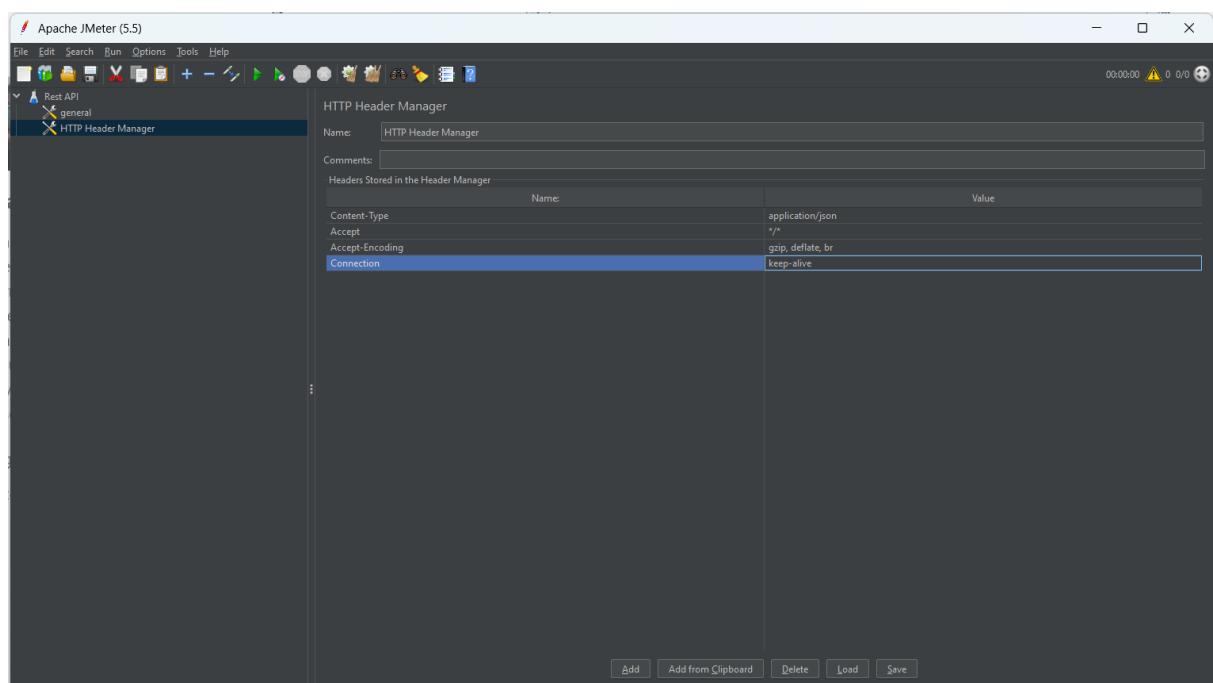
e. Menambahkan HTTP Header Manager

Setelah menambahkan node User Defined Variables, kita lanjutkan dengan menambahkan node HTTP Header Manager. Langkah-langkahnya sama seperti menambahkan node `User Defined Variables`, hanya saja yang dipilih adalah HTTP Header Manager`.

Di node ini kita akan menambahkan informasi apa saja yang dikirimkan JMeter ke HTTP request header.



Add header

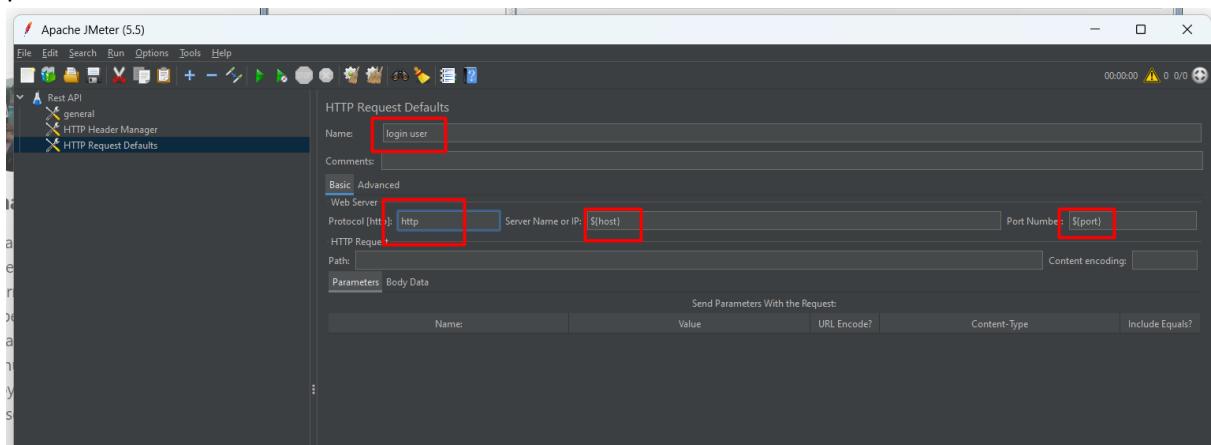


f. Menambahkan HTTP Request Default

Langkah berikutnya adalah menambahkan node HTTP Request Defaults, caranya juga sama seperti sebelumnya hanya saja yang dipilih node HTTP Request Defaults.

setting

Di node ini kita cukup mengeset informasi nama server/ip address, port dan protocol.

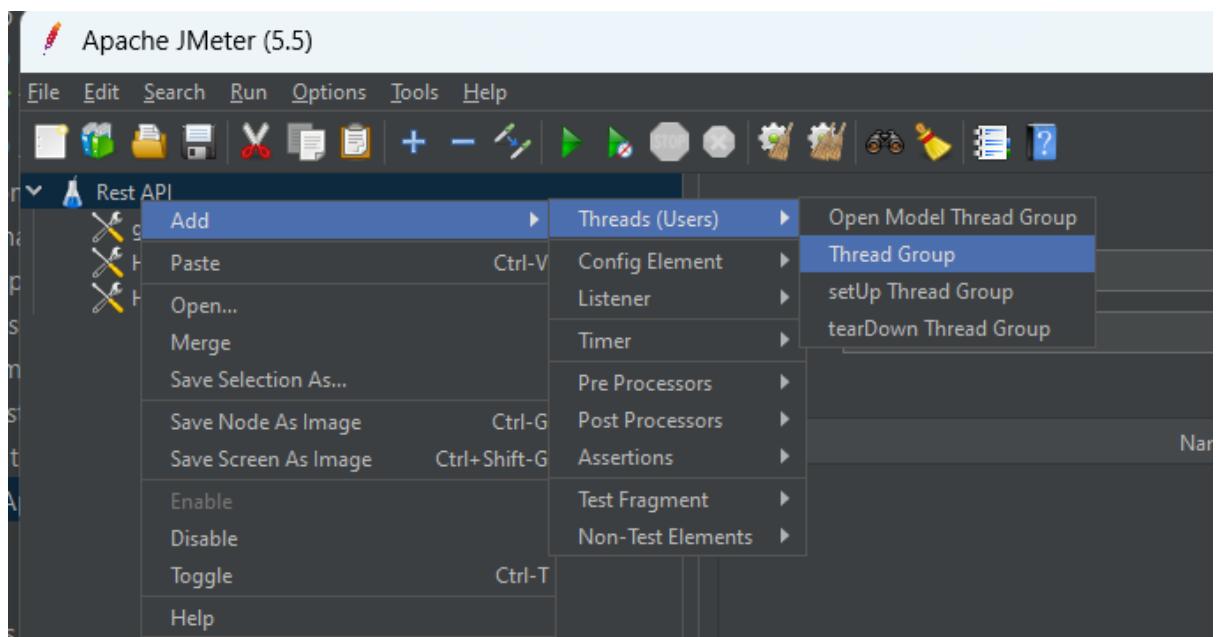


Karena sebelumnya kita sudah mendefinisikan nama host dan port di node User Defined Variables, di node ini kita tinggal panggil variabel tersebut dengan format : \${**NAMA_VARIABEL**}.

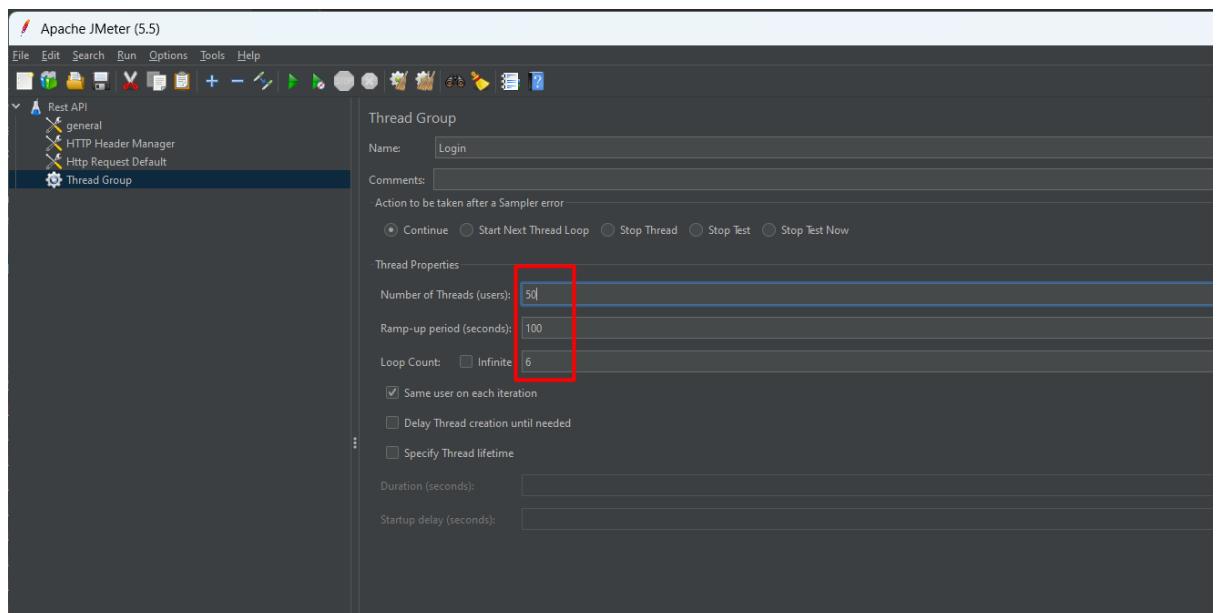
g. Menambahkan Thread Group

Node ini digunakan untuk mengelompokan service yang akan di tes. Misal kita mempunyai service Login, Mahasiswa dan Dosen, ketiga service ini sebaiknya dibuatkan Thread Group masing-masing.

Untuk menambahkan node Thread Group klik kanan node Test Plan (Performance Test Rest API) -> Add -> Threads (Users) -> Thread Group.



Setting



Dari setting di atas, kita akan membuat skenario `performance test` seperti berikut :

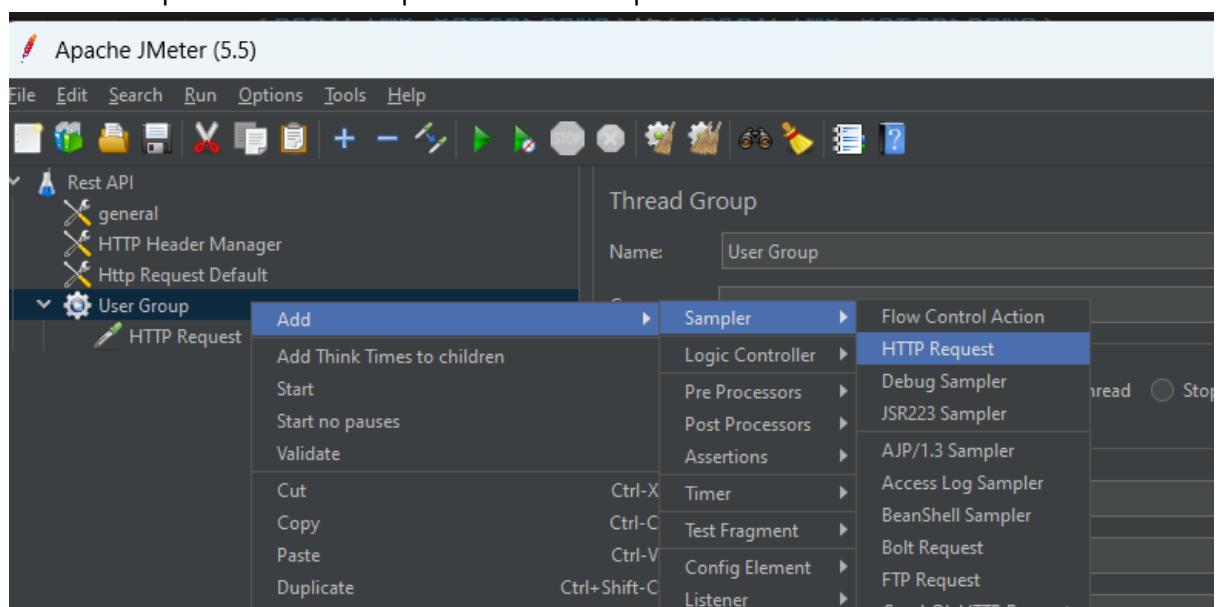
1. Jumlah user sebanyak 50 orang
2. Setiap 2 detik ($100/50$), akan mengirimkan 6 request ke server.
3. Total jumlah sample = 300 (50×6)

Skenario `performance test` ini bisa diganti-ganti nilainya sesuai kebutuhan.

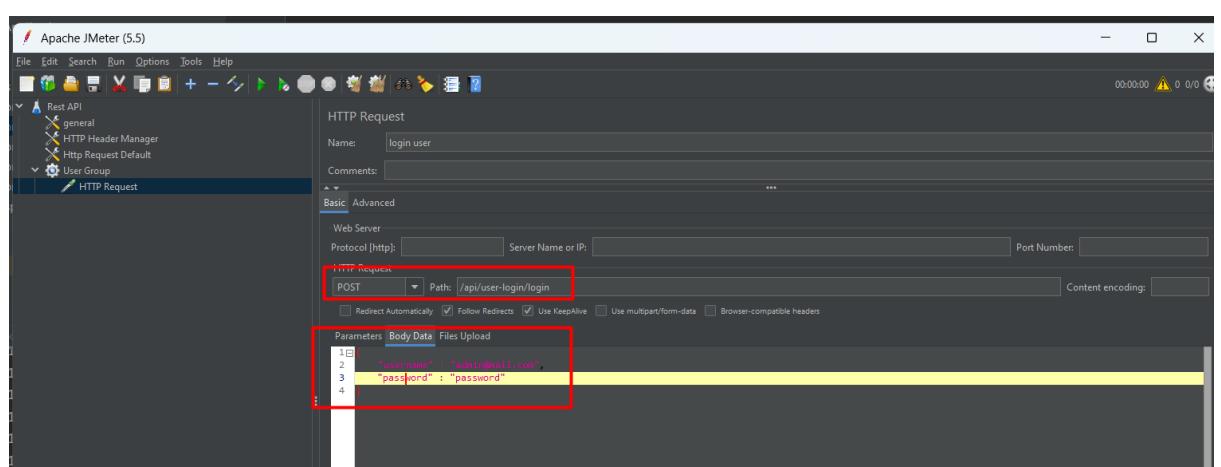
h. Menambahkan HTTP Request

Setelah menentukan skenario performance test, langkah berikutnya adalah menambahkan node HTTP Request. Di node inilah kita akan menentukan Rest API yang akan di tes. Misal rest api login akan kita tambahkan sebagai bagian dari HTTP Request.

Untuk menambahkan node HTTP Request klik kanan node Thread Group (User Group) -> Add -> Sampler -> HTTP Request.



Setting

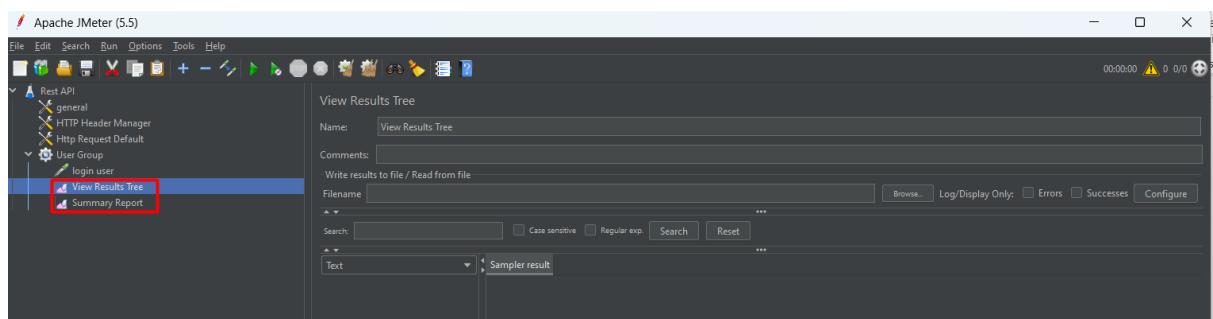


i. Menambahkan report untuk menampilkan hasil Performance test

Ada beberapa format laporan yang digunakan untuk menampilkan hasil performance test yaitu :

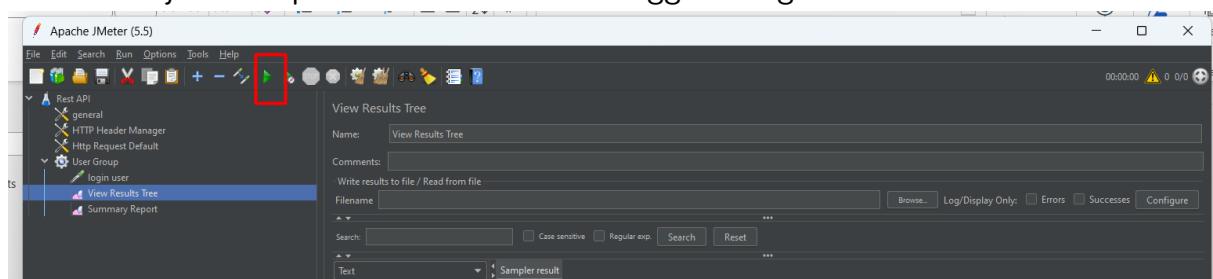
1. View Results in Table
2. View Results Tree
3. Response Times Over Time
4. Transactions per Second
5. Statistical Aggregate Report
6. Summary Report

Untuk menambahkan semua format laporan di atas klik kanan node Thread Group (Web Service Mahasiswa) -> Add -> Listener -> Pilih jenis laporan.

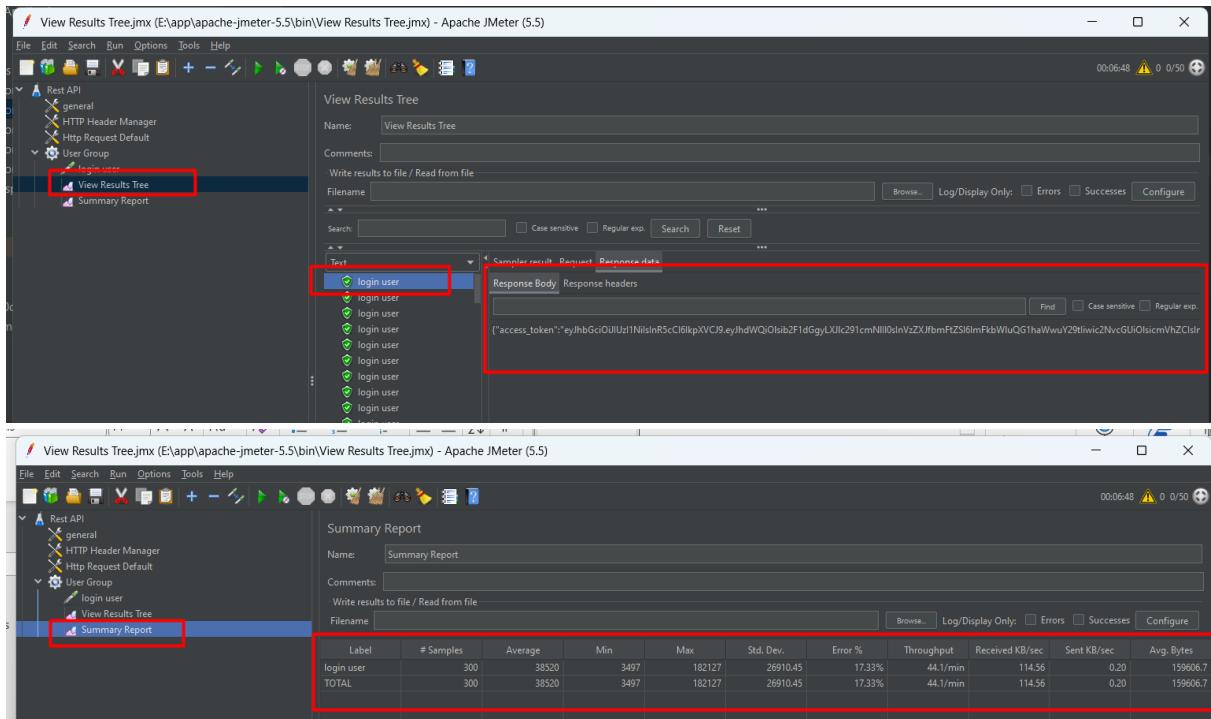


j. Menjalankan Performance test

Untuk menjalankan performance test, kita tinggal mengklik toolbar Start



k. Contoh Hasil Performance Test

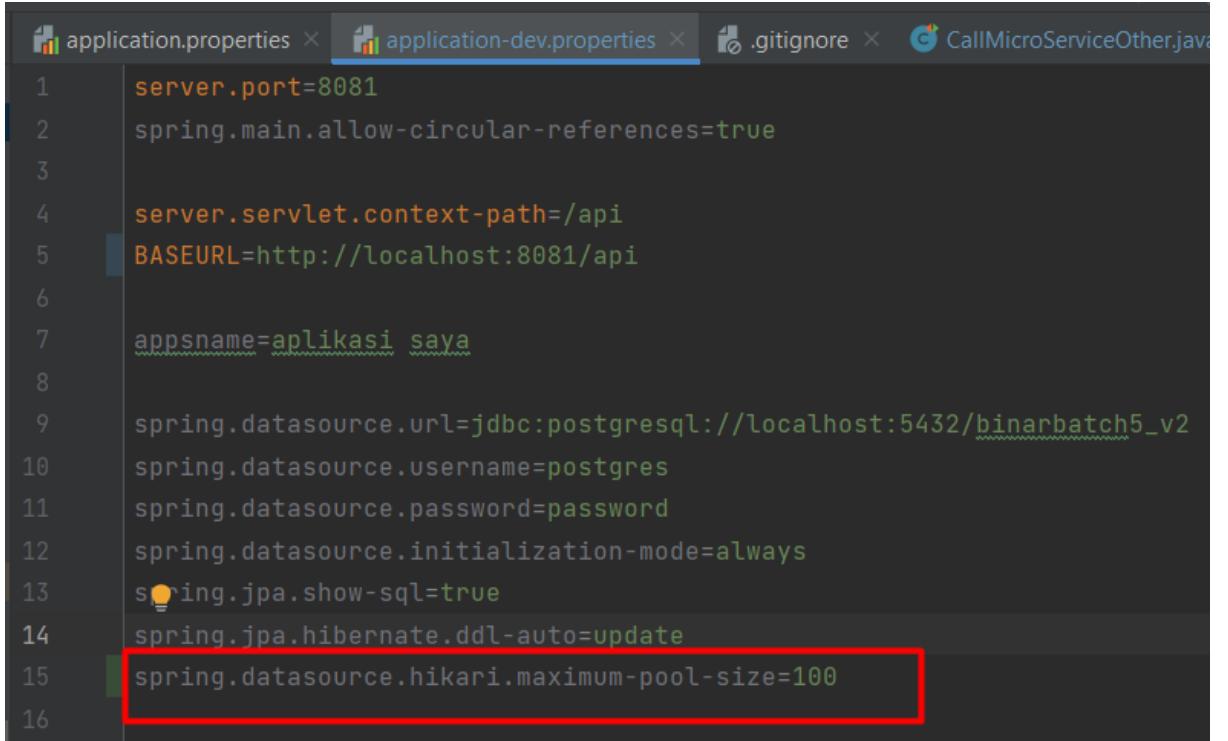


I. Kesimpulan

Sebagai penutup saya belum menemukan referensi yang tepat bagaimana cara menganalisa hasil dari `performance test` di atas, tetapi biasanya yang saya jadikan acuan adalah hasil yang bisa dilihat melalui node `View Results in Table` kolom Status, dari node ini kelihatan apakah ada request service yang berstatus `Warning`. Jika ada mungkin ini bisa menjadi bahan acuan apakah service kita masih perlu di-improve lagi atau skenario `performance test` yang perlu diganti atau karena faktor lainnya seperti kondisi jaringan yang lagi enggak fit pada saat melakukan tes 😊.

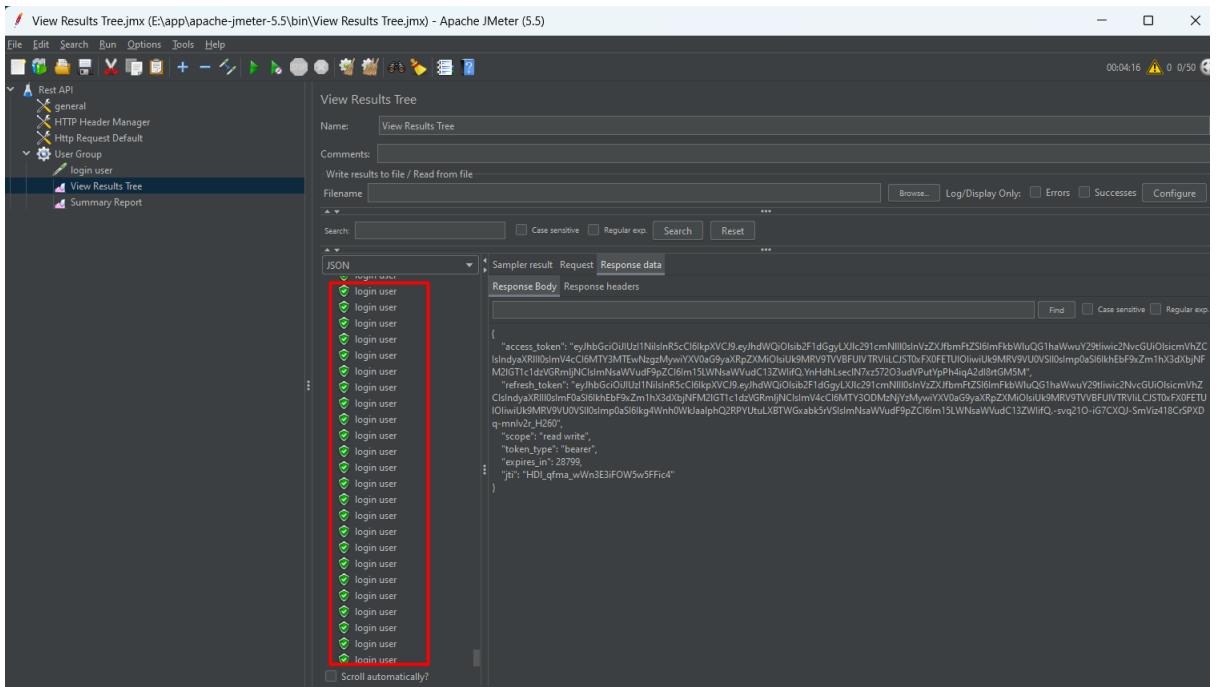
m. Application.properties

Tambahkan di application properties untuk menampung koneksi user lebih banyak.



```
application.properties x application-dev.properties x .gitignore x CallMicroServiceOther.java
1 server.port=8081
2 spring.main.allow-circular-references=true
3
4 server.servlet.context-path=/api
5 BASEURL=http://localhost:8081/api
6
7 appsname=aplikasi saya
8
9 spring.datasource.url=jdbc:postgresql://localhost:5432/binarbatch5_v2
10 spring.datasource.username=postgres
11 spring.datasource.password=password
12 spring.datasource.initialization-mode=always
13 spring.jpa.show-sql=true
14 spring.jpa.hibernate.ddl-auto=update
15 spring.datasource.hikari.maximum-pool-size=100
16
```

Semua sukses



The screenshot shows the Apache JMeter interface with the 'View Results Tree' plugin selected. The left sidebar lists a 'Rest API' category with several sub-items like 'general', 'HTTP Header Manager', 'Http Request Default', 'User Group', and 'login user'. Under 'login user', there are three entries: 'View Results Tree', 'Summary Report', and a long list of 'login user' requests. The right panel displays the 'View Results Tree' configuration and the response data for one of the requests. The response body contains JSON data for an access token, refresh token, scope, and token type. The entire list of 'login user' requests is highlighted with a red box.

```
access_token": "eyJhbGciOiJIUzI1NiIsInR5cCIkVjC9eyJhdWQiOlsb2F1dGgyLXJlc291cmNlIi0sInVzZXJfbmFtZI0lmFkbWluQG1haWwuY29tIiwic2NvcGUoIscmVhZC1nI0yXRII0sVmV4cGJMVT3MTeWNgzbMwyYXV0VG9yaRpZXMi0slkUkMR9V7VFUfIVTRVILCJSTxFOFETU0iwlkUsMR9V9UVJSi0smp0s56krEBf9zCm1h3QxbNFN2lGT1cdvGRmjhIClsImNsWvudf9pZCf6m15LNhsVVudfC13ZWIIQ_YnHdhLscdIN7x57203uduPuyPhliqA2d0rGM5M",
"refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIkVjC9eyJhdWQiOlsb2F1dGgyLXJlc291cmNlIi0sInVzZXJfbmFtZI0lmFkbWluQG1haWwuY29tIiwic2NvcGUoIscmVhZC1nI0yXRII0sVmV4cGJMVT3MTeWNgzbMwyYXV0VG9yaRpZXMi0slkUkMR9V7VFUfIVTRVILCJSTxFOFETU0iwlkUsMR9V9UVJSi0smp0s56krEBf9zCm1h3QxbNFN2lGT1cdvGRmjhIClsImNsWvudf9pZCf6m15LNhsVVudfC13ZWIIQ_YnHdhLscdIN7x57203uduPuyPhliqA2d0rGM5M",
"scope": "read写",
"token_type": "bearer",
"expires_in": 28799,
"jti": "HDQ_qfma_wWn3E3FOW5w5FFic4"
```

n. referensi

<http://coding4ever.net/blog/2015/10/20/performance-test-menggunakan-jmeter/>

49.Transactional spring boot

Terdiri dari commit dan rollback

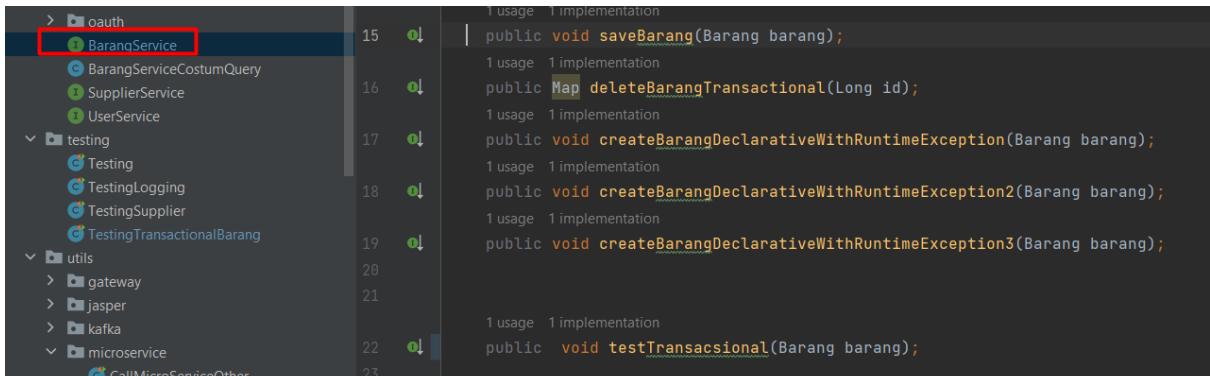
Ditandai dengan annotasi : @Transactional

Tanpa menggunakan @Transactional, jika terjadi eror pada sebuah transaksi, maka tidak akan di rollback.

Dengan menggunakan anotasi @Transactional di spring boot, akan menangulangi masalah diatas, jika terjadi eror di salah satu transaksi, maka transaksi sebelumnya sukses di edit, delete, simpan akan di rollback kembali.

a. Studi kasus pada tabel barang

b. Service Barang



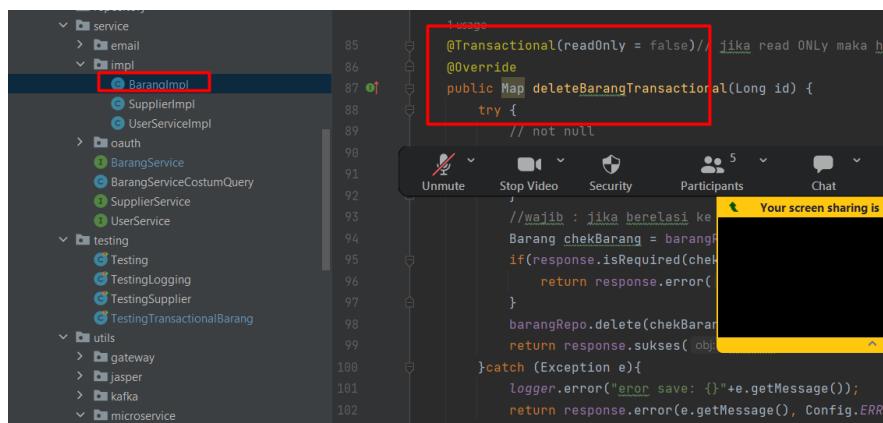
```
public void saveBarang(Barang barang);
public Map deleteBarangTransactional(Long id);
public void createBarangDeclarativeWithRuntimeException(Barang barang);
public void createBarangDeclarativeWithRuntimeException2(Barang barang);
public void createBarangDeclarativeWithRuntimeException3(Barang barang);

public void testTransacsional(Barang barang);
```

c. Service Implementasi barang

Readonly hanya diperlukan jika transaksi tersebut bersifat read only.

// jika read ONLY maka hanya dapat lakukan transaksi read, dan tidak bisa lakukan deleted, insert, update



```
1 usage
@Transactional(readOnly = false)// jika read ONLY maka hanya dapat lakukan transaksi read, dan tidak bisa lakukan deleted, insert, update
@Override
public Map deleteBarangTransactional(Long id) {
    try {
        // not null
        //wajib : jika berelasi ke tabel
        Barang chekBarang = barangRepo.getByID(id);
        if(response.isRequired(chekBarang)){
            return response.error("Barang Id tidak ditemukan.", Config.ERROR_403);
        }
        barangRepo.delete(chekBarang);
        return response.sukses("Sukses Deleted");
    }catch (Exception e){
        logger.error("eror save: {} "+e.getMessage());
        return response.error(e.getMessage(), Config.ERROR_500);
    }
}

@Override
public void saveBarang(Barang barang) {
    barangRepo.save(barang);
}
```

```
// @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
// @Transactional(readOnly = true)// jika read ONLY maka hanya dapat lakukan transaksi read, dan tidak bisa lakukan deleted, insert, update
@Override
public Map deleteBarangTransactional(Long id) {
    try {
        // not null
        if(response.isRequired(id)){
            return response.error("Id wajib diisi.", Config.ERROR_403);
        }
        //wajib : jika berelasi ke tabel
        Barang chekBarang = barangRepo.getByID(id);
        if(response.isRequired(chekBarang)){
            return response.error("Barang Id tidak ditemukan.", Config.ERROR_403);
        }
        barangRepo.delete(chekBarang);
        return response.sukses("Sukses Deleted");
    }catch (Exception e){
        logger.error("eror save: {} "+e.getMessage());
        return response.error(e.getMessage(), Config.ERROR_500);
    }
}

@Override
public void saveBarang(Barang barang) {
    barangRepo.save(barang);
}
@Transactional
@Override
```

```

        public void createBarangDeclarativeWithRuntimeException(Barang
barang) {
    barangRepo.save(barang);
    throw new DataIntegrityViolationException("Throwing exception
for demoing Rollback!!!");
}

@Transactional(rollbackFor = { RuntimeException.class, Error.class
})
@Override
public void createBarangDeclarativeWithRuntimeException2(Barang
barang) {
    barangRepo.save(barang);
    throw new DataIntegrityViolationException("Throwing exception
for demoing Rollback!!!");
}

@Transactional(noRollbackFor = { SQLException.class })
@Override
public void createBarangDeclarativeWithRuntimeException3(Barang
barang) {
    barangRepo.save(barang);
    throw new DataIntegrityViolationException("Throwing exception
for demoing Rollback!!!");
}

@Transactional
@Override
public void testTransacsional(Barang barang) {
    try {
        // transaksi 1 : sukses
        Barang doupdatde = barangRepo.getByID(110L);
        doupdatde.setNama("update =" +new Date());
        barangRepo.save(doupdatde); // jika terjadi eror: update
ini akan rollback

        //transaksi 2 : failed karena null
        barangRepo.save(null);

        /*
        coba testing
        1.@Transactional dihilangkan dan lakukan
        2.lakukan testing
        barangService.createCourseDefaultRatingProgrammatic(null);
        3. lihat yang terjadi di database. transaksi update nama
tetap terjadi, namun transaksi save barang terjadi kegagalan.
        4. kesimpulan : @transaksi wajib ada

        jika tampa @transaksi : tidak melakukan rollback walaupun
terjadi eror di salah satu transaksi

        kapan melakukan commit ? saat transaksi tidak ada yang
eror
    */
} catch (Exception e) {
    System.out.println("eror disini
createCourseDefaultRatingProgrammatic=" +e);
}
TransactionAspectSupport.currentTransactionStatus().setRollbackOnly()

```

```

    ;
}

}

```

d. Testing Tampa @Transactional

Data awal didatabase id :110

	id	harga	nama	satuan	stok	created_date	deleted_date	updated_date	l
7	132	10	test4	pcs	[NULL]	2022-12-20 21:20:23.787	[NULL]	2022-12-20 21:20:23.787	
8	133	10	test1	pcs	[NULL]	2022-12-20 21:21:23.039	[NULL]	2022-12-20 21:21:23.039	
9	137	10	test4	pcs	[NULL]	2022-12-20 21:21:23.174	[NULL]	2022-12-20 21:21:23.174	
10	139	10	test1	pcs	[NULL]	2022-12-20 22:01:59.828	[NULL]	2022-12-20 22:01:59.828	
11	143	10	test4	pcs	[NULL]	2022-12-20 22:01:59.891	[NULL]	2022-12-20 22:01:59.891	
12	144	10	test4	pcs	[NULL]	2022-12-20 22:08:06.943	[NULL]	2022-12-20 22:08:06.943	
13	145	10	test4	pcs	[NULL]	2022-12-20 22:09:04.867	[NULL]	2022-12-20 22:09:04.867	
14	146	10	test1	pcs	[NULL]	2022-12-23 19:05:30.554	[NULL]	2022-12-23 19:05:30.554	
15	150	10	test4	pcs	[NULL]	2022-12-23 19:05:30.693	[NULL]	2022-12-23 19:05:30.693	
16	151	10	test4	pcs	[NULL]	2022-12-23 19:07:17.400	[NULL]	2022-12-23 19:07:17.400	
17	152	10	test4	pcs	[NULL]	2022-12-23 19:08:45.500	[NULL]	2022-12-23 19:08:45.500	
18	153	10	test4	pcs	[NULL]	2022-12-23 19:16:20.175	[NULL]	2022-12-23 19:16:20.175	
19	154	[NULL]	[NULL]	[NULL]	[NULL]	2022-12-23 19:27:47.147	[NULL]	2022-12-23 19:27:47.147	
20	110	40,000	Data	pcs	200	2022-12-20 20:58:32.819	[NULL]	2022-12-23 19:40:20.215	

Perhatikan 2 transaksi berikut: ada dua transaksi update dan insert into tabel barang.

```

@Override
public void testTransacsional(Barang barang) {
    try {
        // transaksi 1 : sukses
        Barang doupdatde = barangRepo.getById( idbebas: 110L);
        doupdatde.setNama("update "+new Date());
        barangRepo.save(doupdatde); // jika terjadi eror: update ini akan rollback

        //transaksi 2 : failed karena null
        barangRepo.save( entity: null);

    /*
    coba testing
    */
    }
}

```

Lakukan testing

```

UserService
└── testing
    ├── Testing
    ├── TestingLogging
    ├── TestingSupplier
    └── TestingTransactionalBarang
        @Test
        public void test4() {
            Barang barang = new Barang();
            barangService.testTransacs();
        }

```

Transaksi 1: akan sukses di update

Transaksi 2 : gagal : karena value insert null

Data di tabase berubah, karena transaksi 1 berhasil update

Record	id	test	value	update	create	update
14	140	10	test1	19:03:50.334	2022-12-23 19:03:50.334	
15	150	10	test4	2022-12-23 19:05:30.693	[NULL]	2022-12-23 19:05:30.693
16	151	10	test4	2022-12-23 19:07:17.400	[NULL]	2022-12-23 19:07:17.400
17	152	10	test4	2022-12-23 19:08:45.500	[NULL]	2022-12-23 19:08:45.500
18	153	10	test4	2022-12-23 19:16:20.175	[NULL]	2022-12-23 19:16:20.175
19	154	[NULL]	[NULL]	2022-12-23 19:27:47.147	[NULL]	2022-12-23 19:27:47.147
20	110	40,000	update =Fri Dec 23 20:08:32 WIB 2022	200	2022-12-20 20:58:32.819	[NULL]

Nah, ini bahaya, karena jika ada salah satu transaksi gagal maka transaksi sebelumnya tidak di rollback.. bagaimana caranya di rollback, yaitu menggunakan anotasi @transactional, perhatikan testing berikutnya.

e. Testing Dengan @Transactional

Data awal didatabase id :110

grid select * from barang b | Enter a SQL expression to filter results (use Ctrl+Space)

	id	harga	nama	satuan	stok	created_date	deleted_date	updated_date	s
7	132	10	test4	pcs	[NULL]	2022-12-20 21:20:23.787	[NULL]	2022-12-20 21:20:23.787	
8	133	10	test1	pcs	[NULL]	2022-12-20 21:21:23.039	[NULL]	2022-12-20 21:21:23.039	
9	137	10	test4	pcs	[NULL]	2022-12-20 21:21:23.174	[NULL]	2022-12-20 21:21:23.174	
10	139	10	test1	pcs	[NULL]	2022-12-20 22:01:59.828	[NULL]	2022-12-20 22:01:59.828	
11	143	10	test4	pcs	[NULL]	2022-12-20 22:01:59.891	[NULL]	2022-12-20 22:01:59.891	
12	144	10	test4	pcs	[NULL]	2022-12-20 22:08:06.943	[NULL]	2022-12-20 22:08:06.943	
13	145	10	test4	pcs	[NULL]	2022-12-20 22:09:04.867	[NULL]	2022-12-20 22:09:04.867	
14	146	10	test1	pcs	[NULL]	2022-12-23 19:05:30.554	[NULL]	2022-12-23 19:05:30.554	
15	150	10	test4	pcs	[NULL]	2022-12-23 19:05:30.693	[NULL]	2022-12-23 19:05:30.693	
16	151	10	test4	pcs	[NULL]	2022-12-23 19:07:17.400	[NULL]	2022-12-23 19:07:17.400	
17	152	10	test4	pcs	[NULL]	2022-12-23 19:08:45.500	[NULL]	2022-12-23 19:08:45.500	
18	153	10	test4	pcs	[NULL]	2022-12-23 19:16:20.175	[NULL]	2022-12-23 19:16:20.175	
19	154	[NULL]	[NULL]	[NULL]	[NULL]	2022-12-23 19:27:47.147	[NULL]	2022-12-23 19:27:47.147	
20	110	40,000	Data	pcs	200	2022-12-20 20:58:32.819	[NULL]	2022-12-23 19:40:20.215	

Perhatikan 2 transaksi berikut: ada dua transaksi update dan insert into tabel barang. Perbedaannya dengan testing sebelumnya adalah diberikan anotasi **@Transactional**

```

    @Transactional
    @Override
    public void testTransaksional(Barang barang) {
        try {
            // transaksi 1 : sukses
            Barang doupdate = barangRepo.getById(idbebas: 110L);
            doupdate.setNama("update "+new Date());
            barangRepo.save(doupdate); // jika terjadi error: update ini akan rollback

            //transaksi 2 : failed karena null
            barangRepo.save(entity: null);
        }
    }

```

Lakukan testing

```

    @Test
    public void test4() {
        Barang barang = new Barang();
        barangService.testTransaksional(barang);
    }

```

Transaksi 1: akan sukses di update

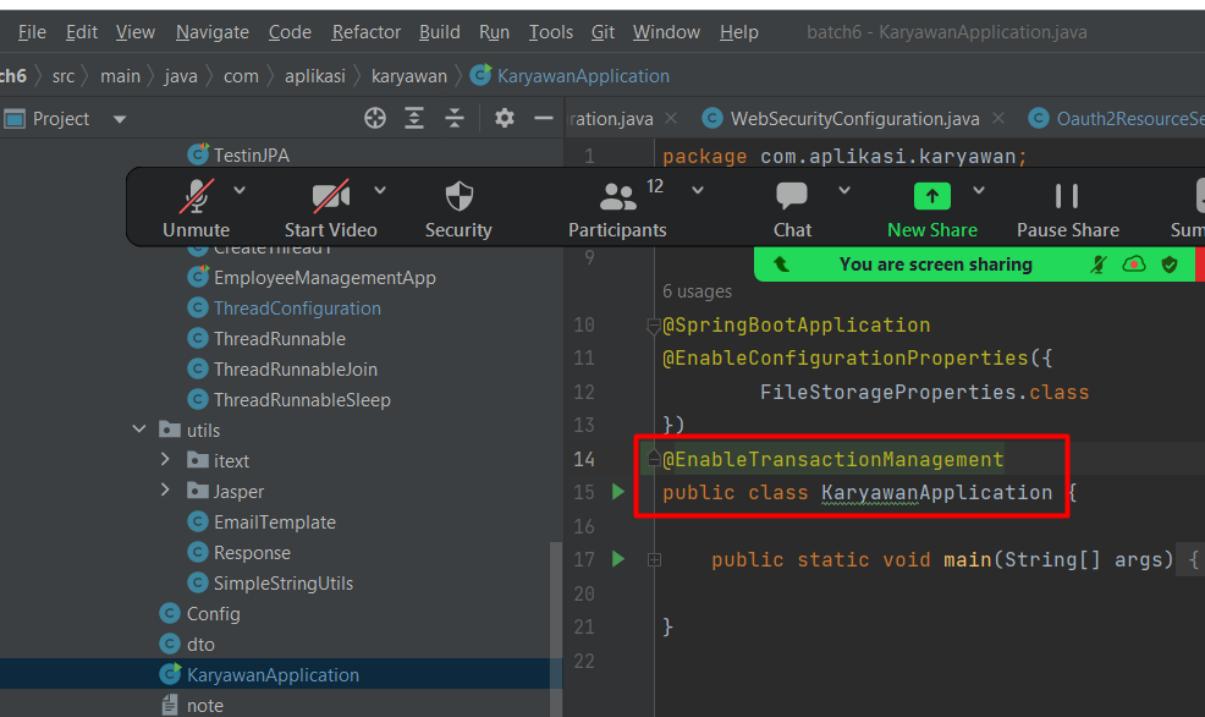
Transaksi 2 : gagal : karena value insert null

Data di tabase tidak berubah, karena diberikan anotasi @Transactional , walaupun transaksi 1 sukses.

14	140	10	test1	pcs	[NULL]	2022-12-23 19:03:50.554	[NULL]	2022-12-23 19:03:50.554
15	150	10	test4	pcs	[NULL]	2022-12-23 19:05:30.693	[NULL]	2022-12-23 19:05:30.693
16	151	10	test4	pcs	[NULL]	2022-12-23 19:07:17.400	[NULL]	2022-12-23 19:07:17.400
17	152	10	test4	pcs	[NULL]	2022-12-23 19:08:45.500	[NULL]	2022-12-23 19:08:45.500
18	153	10	test4	pcs	[NULL]	2022-12-23 19:16:20.175	[NULL]	2022-12-23 19:16:20.175
19	154	[NULL]	[NULL]	pcs	[NULL]	2022-12-23 19:27:47.147	[NULL]	2022-12-23 19:27:47.147
20	110	40,000	update =Fri Dec 23 20:08:32 WIB 2022	pcs	200	2022-12-20 20:58:32.819	[NULL]	2022-12-23 20:08:33.010

Nah, karena jika ada salah satu transaksi gagal maka transaksi sebelumnya akan di rollback. Dan ini adalah solusi nya.

f. Main Application add



```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help batch6 - KaryawanApplication.java
ch6 > src > main > java > com > aplikasi > karyawan > KaryawanApplication
Project + - ration.java < WebSecurityConfiguration.java < Oauth2ResourceSe
Participants Chat New Share Pause Share Sum
Unmute Start Video Security You are screen sharing
EmployeeManagementApp
ThreadConfiguration
ThreadRunnable
ThreadRunnableJoin
ThreadRunnableSleep
utils
itext
Jasper
EmailTemplate
Response
SimpleStringUtils
Config
dto
KaryawanApplication
note
1 package com.aplikasi.karyawan;
12
13
14 @EnableTransactionManagement
15 public class KaryawanApplication {
16
17     public static void main(String[] args) {
18
19
20
21
22 }
```

g. Testing dengan @Transactional ReadOnly True

Artinya transaksi ini hanya bisa di read only, tidak bisa di write.

Contoh pada transaksi delete diberikan anotasi readonly true, walaupun didelete, maka data pada database tidak akan di delete, dikarenakan ada anotasi @Transactional ReadOnly true

```
// @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
1 usage
    @Transactional(readOnly = true)// jika read ONLY maka hanya dapat lakukan transaksi read
    @Override
    public Map deleteBarangTransactional(Long id) {
        try {
            // not null
            if(response.isRequired(id)){
                return response.error( obj: "Id wajib diisi.", Config.ERROR_403);
            }
            //wajib : jika berelasi ke tabel
            Barang chekBarang = barangRepo.getById(id);
            if(response.isRequired(chekBarang)){
                return response.error( obj: "Barang Id tidak ditemukan.", Config.ERROR_403);
            }
            barangRepo.delete(chekBarang);
            return response.sukses( obj: "Sukses Deleted");
        }catch (Exception e){
    
```

Testing

```
testing
  Testing
  TestingLogging
  TestingSupplier
  TestingTransactionalBarang
    @Test
    public void testDeletedTransactional() {
        Map mao = barangService.deleteBarangTransactional( id: 110);
        System.out.println("response 1 = " + mao);
        assertEquals(HttpStatus.OK, exchange.getStatusCode());
    }
  utils
    > gateway
    > jasper
    > kafka
  26
  27
  28
  29
  30
  31
  32
  33
```

Data pada database tetap ada

	123 id	123 harga	abc nama	abc satuan	123 stok	created_date	deleted_date	updated_date
7	132	10	test4	pcs	[NULL]	2022-12-20 21:20:23.787	[NULL]	2022-12-20 21:20:23.787
8	133	10	test1	pcs	[NULL]	2022-12-20 21:21:23.039	[NULL]	2022-12-20 21:21:23.039
9	137	10	test4	pcs	[NULL]	2022-12-20 21:21:23.174	[NULL]	2022-12-20 21:21:23.174
10	139	10	test1	pcs	[NULL]	2022-12-20 22:01:59.828	[NULL]	2022-12-20 22:01:59.828
11	143	10	test4	pcs	[NULL]	2022-12-20 22:01:59.891	[NULL]	2022-12-20 22:01:59.891
12	144	10	test4	pcs	[NULL]	2022-12-20 22:08:06.943	[NULL]	2022-12-20 22:08:06.943
13	145	10	test4	pcs	[NULL]	2022-12-20 22:09:04.867	[NULL]	2022-12-20 22:09:04.867
14	146	10	test1	pcs	[NULL]	2022-12-23 19:05:30.554	[NULL]	2022-12-23 19:05:30.554
15	150	10	test4	pcs	[NULL]	2022-12-23 19:05:30.693	[NULL]	2022-12-23 19:05:30.693
16	151	10	test4	pcs	[NULL]	2022-12-23 19:07:17.400	[NULL]	2022-12-23 19:07:17.400
17	152	10	test4	pcs	[NULL]	2022-12-23 19:08:45.500	[NULL]	2022-12-23 19:08:45.500
18	153	10	test4	pcs	[NULL]	2022-12-23 19:16:20.175	[NULL]	2022-12-23 19:16:20.175
19	154	[NULL]	[NULL]	[NULL]	[NULL]	2022-12-23 19:27:47.147	[NULL]	2022-12-23 19:27:47.147
20	110	40,000	Data	pcs	200	2022-12-20 20:58:32.819	[NULL]	2022-12-23 20:08:33.010

h. Branch

https://github.com/rikialdi/binar_batch_5/pull/new/231222-transactional

https://gitlab.com/rikialdi/superproof-adam/-/tree/sesi8-transactional?ref_type=heads

i. Output Ketika Eror

The screenshot shows a Postman interface with a sidebar navigation menu. The main area displays a PUT request to the URL 'PUT Deleted Update Transactional...'. The request body is set to JSON and contains the following code:

```
1 {  
2   "id": 33  
3 }
```

The response tab shows the following details:

- Status: 500 Internal Server Error
- Time: 43 ms
- Size: (not explicitly shown)
- Body (Pretty):
```json  
1 {  
2 "message": "Transaction silently rolled back because it has been marked as rollback-only",  
3 "status": 404  
4 }  
```

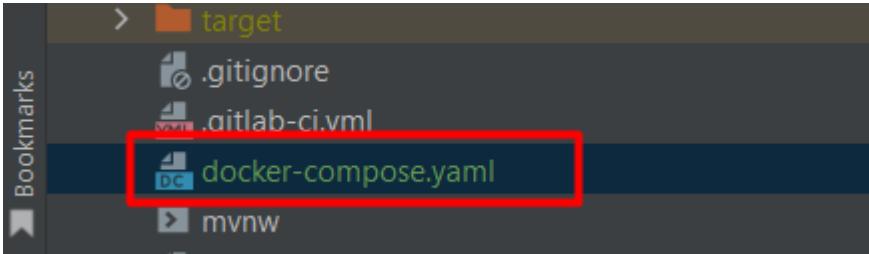
j. Noted

Ketika eror di logika, maka @Transactional tidak berfungsi

50.Redis + Spring Security + Docker

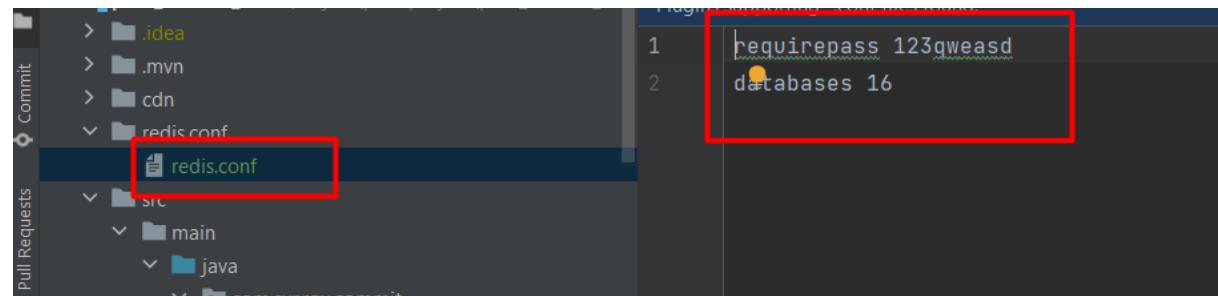
a. Instal Docker Compose

b. Docker Compose File



```
version: '3.7'
services:
# pg-server:
#   image: postgres:15.1
#   container_name: db-lms
#   restart: always
#   environment:
#     POSTGRES_DB: lms
#     POSTGRES_USER: vclude
#     POSTGRES_PASSWORD: 123qweasd
#     PGDATA: /pg-data
#   ports:
#     - "5432:5432"
#   healthcheck:
#     test: pg_isready -U vclude -d lms
#     interval: 10s
#     timeout: 3s
#     retries: 5
  redis-lms:
    container_name: redis-lms
    hostname: redis-lms
    image: redis:latest
    command: [ "redis-server", "/etc/redis/redis.conf" ]
    volumes:
      - ./redis.conf:/etc/redis/redis.conf
    ports:
      - "6379:6379"
# lms-api:
#   container_name: api-lms
#   build: .
#   links:
#     - redis-lms
#   ports:
#     - "80:80"
```

c. Copy file conf Redis

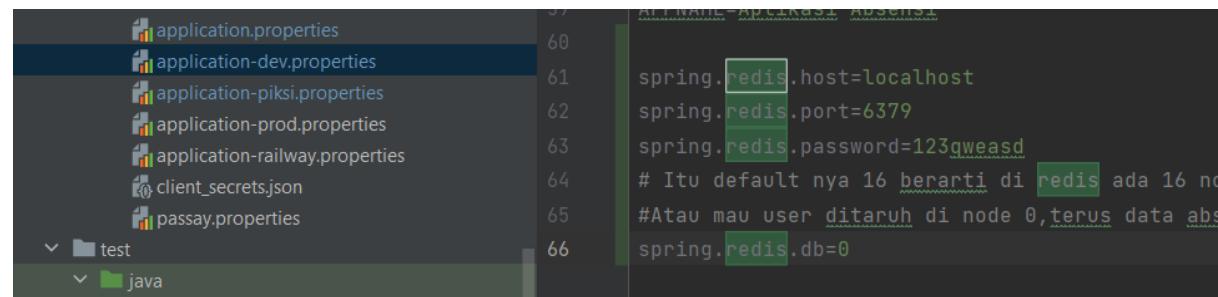


```
requirepass 123qweasd
databases 16
```

penjelasan :

default nya 16 berarti di redis ada 16 node database tuh, misal ada 2 projek mau pakai 1 redis, projek 1 pakai database 0,projek 2 pakai database 1
Atau mau user ditaruh di node 0,terus data absen di node 1 bisa tinggal buat wrappernya.

d. Koneksi di application properties



```
spring.redis.host=localhost
spring.redis.port=6379
spring.redis.password=123qweasd
# Itu default nya 16 berarti di redis ada 16 node database tuh, misal
#ada 2 projek mau pakai 1 redis, projek 1 pakai database 0,projek 2
#pakai database 1 gitu sih
#Atau mau user ditaruh di node 0,terus data absen di node 1 bisa
#tinggal buat wrappernya
spring.redis.db=0
```

```
spring.redis.host=localhost
spring.redis.port=6379
spring.redis.password=123qweasd
# Itu default nya 16 berarti di redis ada 16 node database tuh, misal
#ada 2 projek mau pakai 1 redis, projek 1 pakai database 0,projek 2
#pakai database 1 gitu sih
#Atau mau user ditaruh di node 0,terus data absen di node 1 bisa
#tinggal buat wrappernya
spring.redis.db=0
```

e. Pom.xml

```
<!--      redis-->
<dependency>
    <groupId>org.springframework.boot</groupId>
```

```

        <artifactId>spring-boot-starter-data-redis</artifactId>
        <optional>true</optional>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.springframework.data/spring-
data-redis -->
    <dependency>
        <groupId>org.springframework.data</groupId>
        <artifactId>spring-data-redis</artifactId>
        <version>2.6.1</version>
    </dependency>
    <dependency>
        <groupId>redis.clients</groupId>
        <artifactId>jedis</artifactId>
        <version>2.9.0</version>
    </dependency>

```

f. Config Redis dengan token spring security

Pada class OAuth2AuthorizationServerConfiguration



```

// ...
private TokenStore tokenStore;
// ...
@Autowired
private TokenStore tokenStore() { return new RedisTokenStore(redisConnectionFactory); }
// ...
@Autowired
private RedisConnectionFactory redisConnectionFactory;

```

```

@Autowired
private TokenStore tokenStore() {
    return new RedisTokenStore(redisConnectionFactory);
}
@Autowired
private RedisConnectionFactory redisConnectionFactory;

```



```

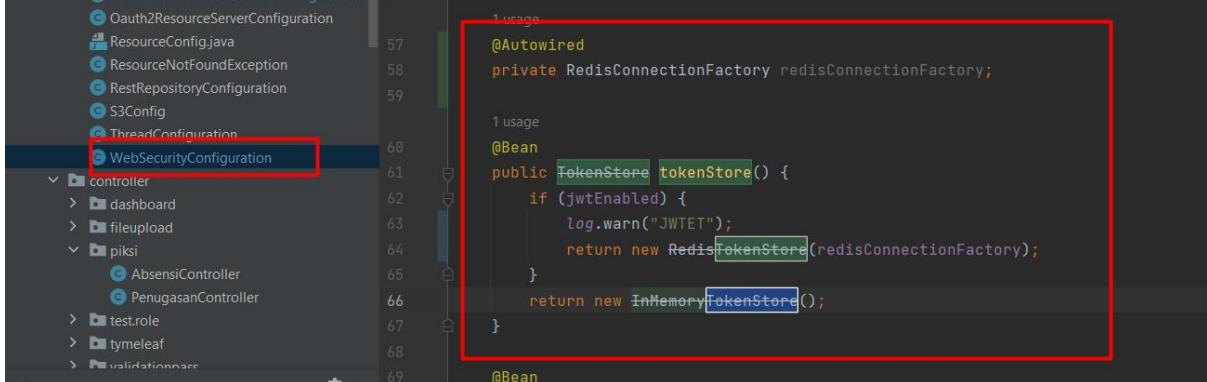
@Override
public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception {
    endpoints.authenticationManager(authenticationManager)
        .tokenStore(tokenStore())
        .accessTokenConverter(accessTokenConverter)
        .userDetailsService(userDetailsService)

    ;
}

.tokenStore(tokenStore())

```

Pada class WebSecurityConfiguration



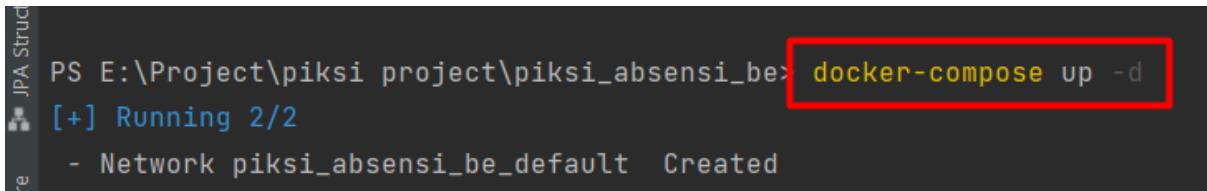
```
    @Autowired
    private RedisConnectionFactory redisConnectionFactory;

    @Bean
    public TokenStore tokenStore() {
        if (jwtEnabled) {
            log.warn("JWTET");
            return new RedisTokenStore(redisConnectionFactory);
        }
        return new InMemoryTokenStore();
    }
```

```
@Autowired
private RedisConnectionFactory redisConnectionFactory;

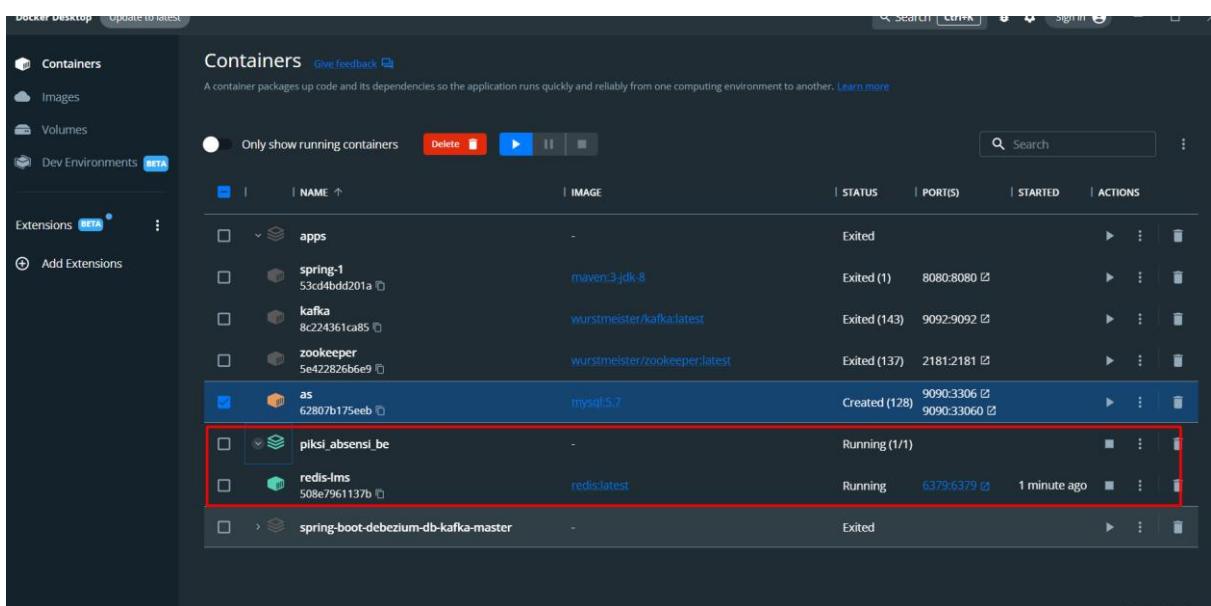
@Bean
public TokenStore tokenStore() {
    if (jwtEnabled) {
        // log.warn("JWTET");
        return new RedisTokenStore(redisConnectionFactory);
    }
    return new InMemoryTokenStore();
}
```

g. Running docker: untuk running redis yang ada di docker

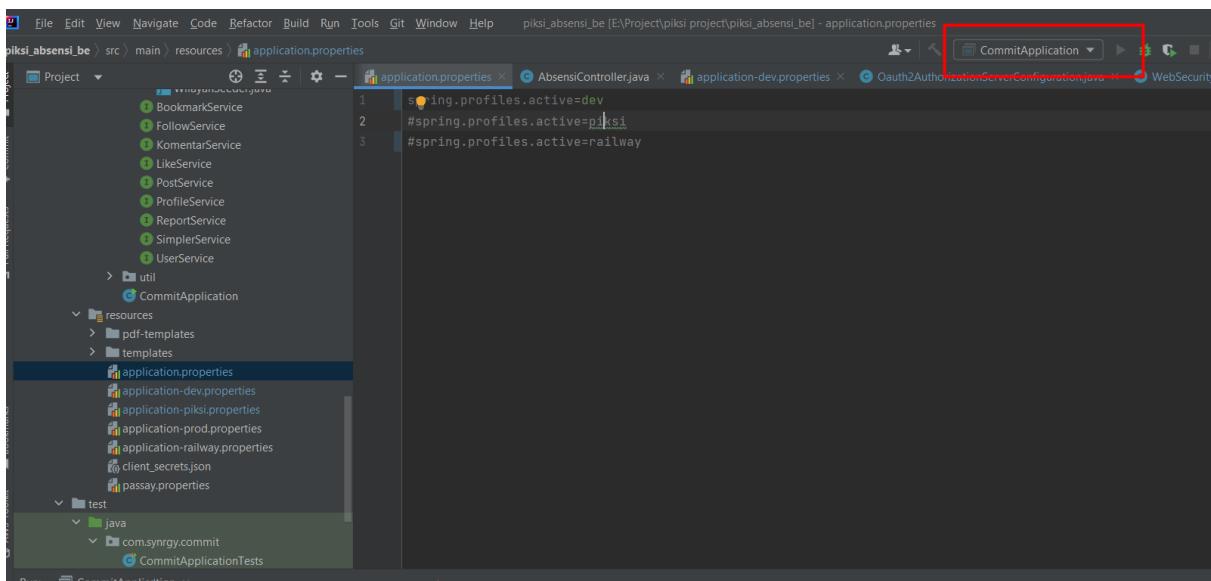


```
PS E:\Project\piksi project\piksi_absensi_be> docker-compose up -d
[+] Running 2/2
  - Network piksi_absensi_be_default  Created
  - Container redis-lms              Started
```

Docker-compose up -d



h. Running aplikasi manual: karena tidak menggunakan docker



i. Jika eror

```
as type=12_1_1_ from auth_client_role authRole0_ inner join auth_role role0_ on authRole0_.role_id=role0_.id where authRole0_.client_id=? client0_.client_secret as client_s5_10_, client0_.grant_types as grant_ty6_10_, client0_.redirect_uris as redirect7_10_, client0_.refresh_token_expir client0_.client_secret as client_s5_10_, client0_.grant_types as grant_ty6_10_, client0_.redirect_uris as redirect7_10_, client0_.refresh_token_expir ; nested exception is java.lang.NoSuchMethodError: 'void org.springframework.data.redis.connection.RedisConnection.set(byte[], byte[])'
```

Ini requirementnya spring redis > 2.0, sama oauthnya pakai latest: ubah di POM.xml

```

<properties>
    <java.version>1.8</java.version>
    <oauth2.version>2.5.2.RELEASE</oauth2.version>
    <jwt.version>1.0.9.RELEASE</jwt.version>
</properties>
<dependencies>

```

j. Test postment login

The screenshot shows a Postman interface with a POST request to `({{host}})/oauth/token`. The request body is set to `x-www-form-urlencoded` and contains the following fields:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> username	admin@mail.com			
<input checked="" type="checkbox"/> password	password			
<input checked="" type="checkbox"/> grant_type	password			
<input checked="" type="checkbox"/> client_id	my-client-web			
<input checked="" type="checkbox"/> client_secret	password			
Key	Value	Description		

The response body is a JSON object containing the access token:

```

1   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
2     eyJhdWQiolsib2F1dGgyLXJlc291cmNlIl0sImV4cCI6MTY3ODI4MDE5MywidXNlc19uYW11Ijo1YWRtaW5AbWFpbC5jb20iLCJqdGkiOiJiZzJ1N1RVWU
3       VoeTlh0OdHRjM4c2tmNVN1RVkicLcjbGllbnRfaQioiJteS1jbGllbnQt2ViIiwic2NvcGUiOlsicmVhZCIsIndyaXRli119.
4         uGNn-P7ackj2PNuckP5cw_zfczyK2z9siT_gDooow8Q",
      "token_type": "bearer",
      "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJhdWQiolsib2F1dGgyLXJlc291cmNlIl0sInVzZXJfbmFtZSI6ImFkbWluQG1haWwuY29tIiwc2NvcGUiOlsicmVhZCIsIndyaXRli10sImF0aSi6Im
          tnMnU3VFVZRwh50WE4N0GMzha2Y1U3VFWsIsImV4cCI6MTY4NDMyODE5MywianRpIjoijluTwx1TmRONDNIMzVVR3RFZmdPX3Vs0dBiiwiY2xpZW50
            X2lkIjoibXktY2xpZW50LXd1YiJ9.8bPxU2pNY_JzqizAHwv5ju0TRX-Wsd0IXfQ_T4XZME0",

```

51. Junit and tests postman example JSON OBJEK

Investree :

```
pm.test("Status code is 200", function () {
```

```
    pm.response.to.have.status(200);
```

```
pm.test("Save Token", function () {
```

```
    var jsonData = pm.response.json();
```

```
    pm.environment.set("access_token", jsonData.access_token);
```

```
    pm.environment.set("refresh_token", jsonData.refresh_token);
```

```
});
```

```
});
```

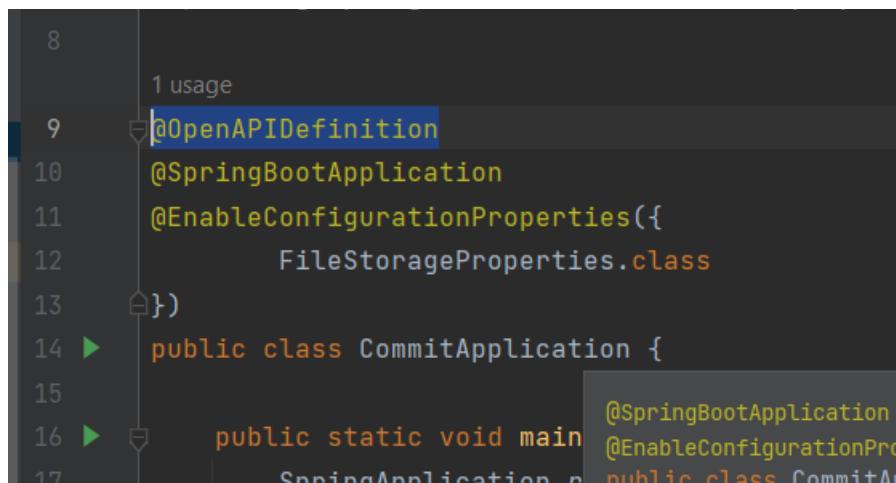
```
@Test
public void restTemplateSaveOBJEK() throws Exception {
    HttpHeaders headers = new HttpHeaders();
    headers.set("Accept", "*/*");
    headers.set("Content-Type", "application/json");
    JSONObject bodyTesting = new JSONObject();
    JSONObject peminjam = new JSONObject();
    peminjam.put("id", 1);
    bodyTesting.put("peminjam", peminjam);

    JSONObject meminjam = new JSONObject();
    meminjam.put("id", 2);
    bodyTesting.put("meminjam", meminjam);

    bodyTesting.put("tenor", 4);
    bodyTesting.put("totalPinjaman", 1000000);
    bodyTesting.put("bungaPersen", 50);
    bodyTesting.put("status", "berjalan");
    System.out.println(bodyTesting.toString());
    HttpEntity<String> entity = new
    HttpEntity<String>(bodyTesting.toString(), headers);
    ResponseEntity<String> exchange =
    restTemplate.exchange("http://localhost:9090/api/v1/transaksi",
    HttpMethod.POST, entity, String.class);
    System.out.println("response = "+exchange.getBody());
    assertEquals(HttpStatus.OK, exchange.getStatusCode());
}
```

52. Swagger

a. Application



```
8
  1 usage
9  @OpenAPIDefinition
10 @SpringBootApplication
11 @EnableConfigurationProperties({
12     FileStorageProperties.class
13 })
14 public class CommitApplication {
15
16     public static void main(String[] args) {
17         SpringApplication.run(CommitApplication.class, args);
18     }
19 }
```

b. Pom.xml

```
<!-- swagger-->
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.6</version>
</dependency>
<!-- <dependency>-->
<!-- <groupId>io.springfox</groupId>
<artifactId>springfox-swagger2</artifactId>-->
<!-- <version>3.0.0</version>-->
<!-- </dependency>-->
<dependency>
    <!-- Generate beautiful documentation from a Swagger-compliant API. -->
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>3.0.0</version>
</dependency>

<!-- swagger-->
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.6</version>
</dependency>
<dependency>
    <!-- Generate beautiful documentation from a Swagger-compliant
API. -->
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>3.0.0</version>
</dependency>
```

c. Cara Access

```
http://localhost:9090/api/swagger-ui/index.html
```

d. Testing -Try out

Pilih try it out

Isi request dan execute

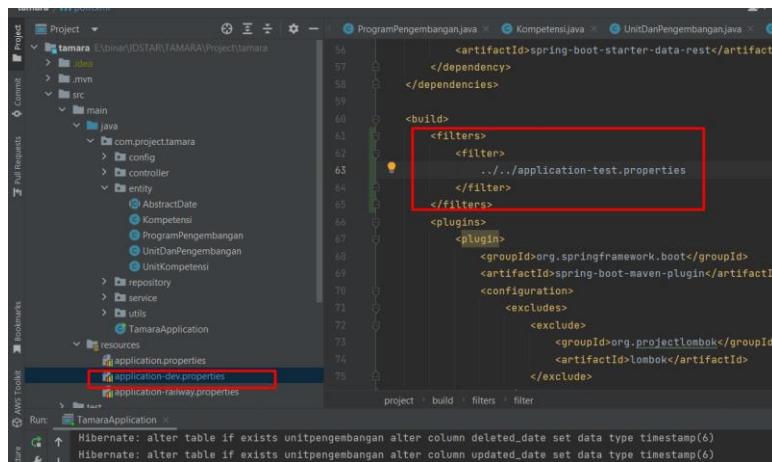
Response

e. Detail di Doc Spring JPA

53. Application.properties set value external

application-eksternal.properties

a. Di pom.xml



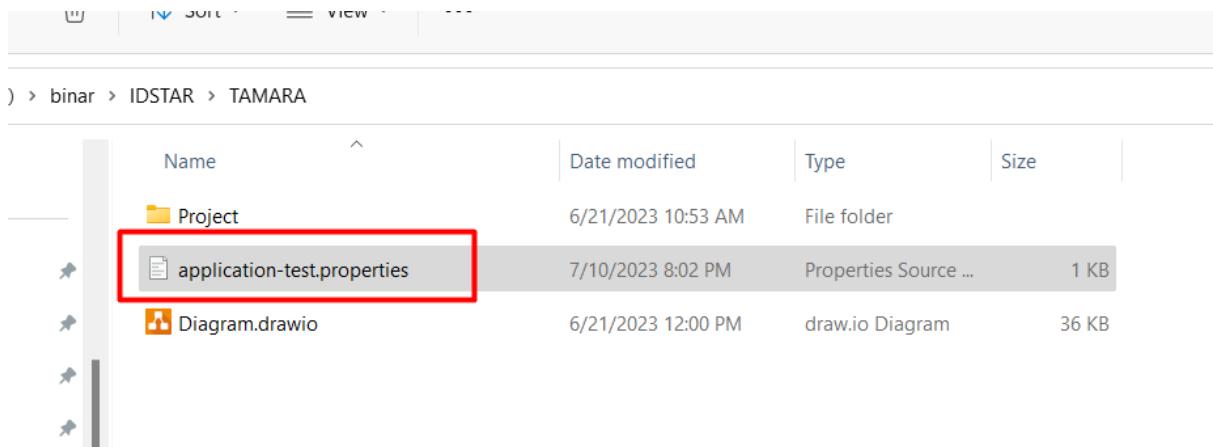
```
<filters>
    <filter>
        .../application-test.properties
    </filter>
</filters>
```

Jika di directory

```
<filters>

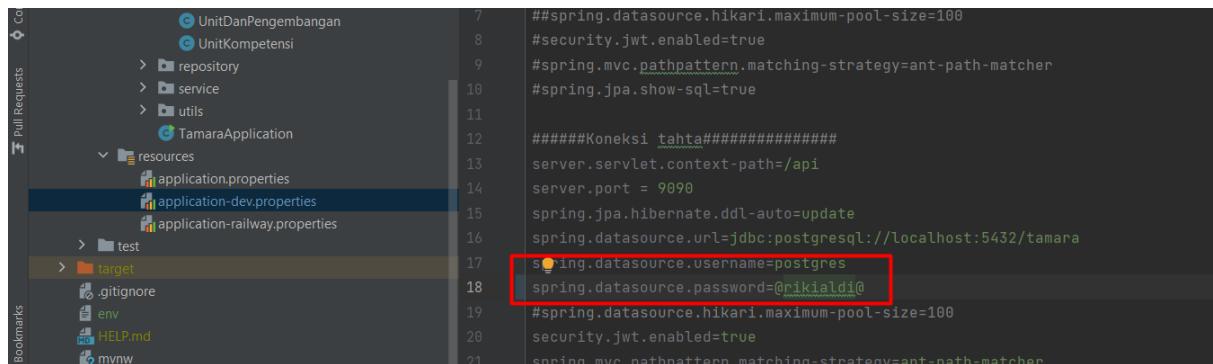
    <filter>
        G:/filter.properties
    </filter>
</filters>
```

b. Folder berada



Name	Date modified	Type	Size
Project	6/21/2023 10:53 AM	File folder	
application-test.properties	7/10/2023 8:02 PM	Properties Source ...	1 KB
Diagram.drawio	6/21/2023 12:00 PM	draw.io Diagram	36 KB

c. Cara manggil di applications.properties



```
UnitDanPengembangan
UnitKompetensi
repository
service
utils
TamaraApplication
resources
application.properties
application-dev.properties
application-railway.properties
test
target
.gitignore
env
HELP.md
mvnw

##spring.datasource.hikari.maximum-pool-size=100
#security.jwt.enabled=true
#spring.mvc.pathpattern.matching-strategy=ant-path-matcher
#spring.jpa.show-sql=true
#####
#Koneksi tahta#####
server.servlet.context-path=/api
server.port = 9090
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:postgresql://localhost:5432/tamara
spring.datasource.username=postgres
spring.datasource.password=@nikialdi0
#spring.datasource.hikari.maximum-pool-size=100
security.jwt.enabled=true
spring.mvc.pathpattern.matching-strategy=ant-path-matcher
```

d. Ref:

<https://fullstackdeveloper.guru/2020/05/24/how-to-populate-application-properties-dynamically-in-spring-boot/>

54.issue and solving

a. hide _Links in api

```
{
  "_embedded" : {
    "questionsTypes" : [ {
      "queTypeID" : 2,
      "queDescription" : "Single choice rating selection",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/question_web/rest/QuestionsType/2"
        },
        "questionsType" : {
          "href" : "http://localhost:8080/question_web/rest/QuestionsType/2{?projectID,queTypeID}",
          "templated" : true
        }
      }
    },
    {
      "queTypeID" : 5,
      "queDescription" : "Subjective questions",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/question_web/rest/QuestionsType/5"
        },
        "questionsType" : {
          "href" : "http://localhost:8080/question_web/rest/QuestionsType/5{?projectID,queTypeID}",
          "templated" : true
        }
      }
    }
  }
}
```

How to hiden, remove, disabel _links ?

Add application run

```
@SpringBootApplication(exclude =
RepositoryRestMvcAutoConfiguration.class) // hiden _ links

7  // @SpringBootApplication
8  // usage
9  @SpringBootApplication(exclude = RepositoryRestMvcAutoConfiguration.class)// hiden _ links
10 public class TamaraApplication {
11     public static void main(String[] args) { SpringApplication.run(TamaraApplication.class, args); }
12 }
13
14
15
16
```

b. Kill port runing

Step 1: netstat -ano | findstr :<PORT>

```

E:\binar\IDSTAR\Project BE Java untuk Global FE\PROJECT_BE\target>netstat -ano | findstr :9090
      TCP    0.0.0.0:9090          0.0.0.0:0          LISTENING      29676
      TCP    [::]:9090           [::]:0          LISTENING      29676

E:\binar\IDSTAR\Project BE Java untuk Global FE\PROJECT_BE\target>

```

Step 2: taskkill /PID <PID> /F

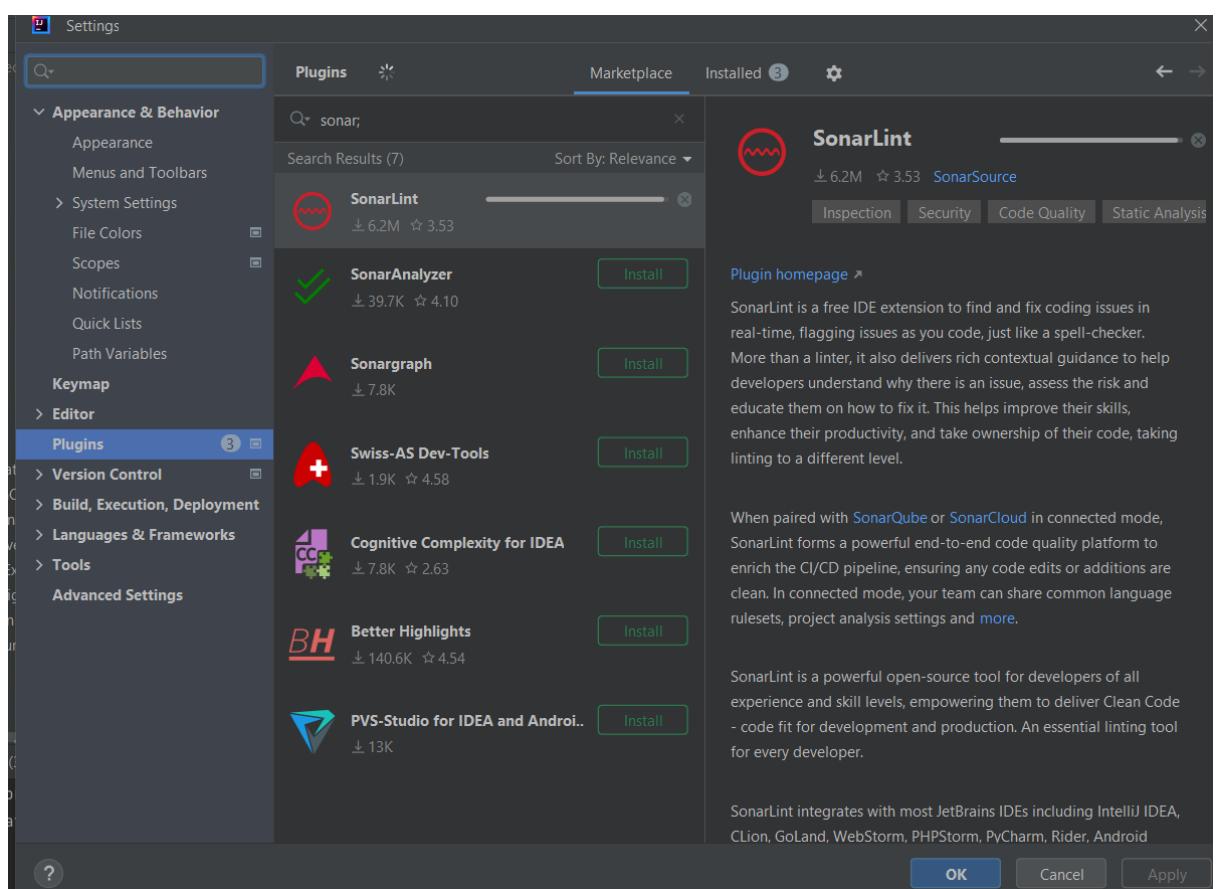
55. Scrum Master

User story point :

<https://www.planitpoker.com/board/#/room/abac32b137af4f729fe3ae44ea6a8e5c>

56. Sonarlint

a. Install intelliij



b. Tujuan

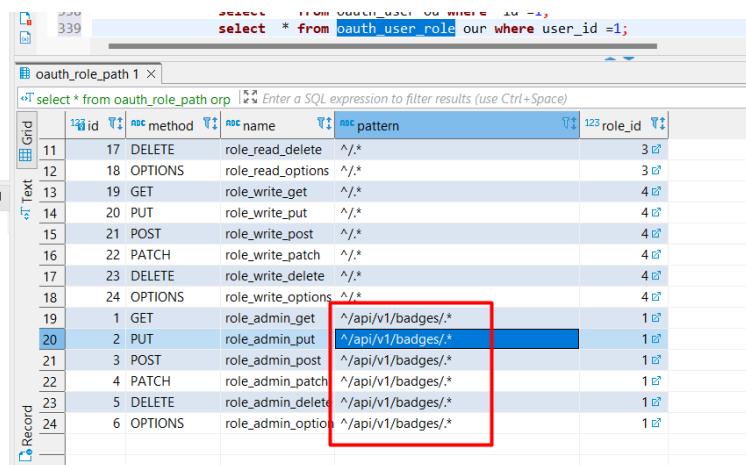
<https://medium.com/javanlabs/meningkatkan-code-quality-dengan-plugin-sonarlint-di-intellij-idea-36705b6cd8fa>

57.update maven security and solving

1. riki2 : mvn 3 not working

58.implementasi path api dinamic oauth_user_role

a. db : api url yang di allow



The screenshot shows a database grid titled 'oauth_role_path 1'. A SQL query is visible at the top: 'select * from oauth_role_path orp'. The grid has columns: id, method, name, pattern, and role_id. A red box highlights the 'pattern' column for row 20, which contains '^/api/v1/badges/*'. The entire row 20 is also highlighted with a blue background.

	id	method	name	pattern	role_id
Grid	11	17	DELETE	role_read_delete	^/*
	12	18	OPTIONS	role_read_options	^/*
	13	19	GET	role_write_get	^/*
	14	20	PUT	role_write_put	^/*
	15	21	POST	role_write_post	^/*
	16	22	PATCH	role_write_patch	^/*
	17	23	DELETE	role_write_delete	^/*
	18	24	OPTIONS	role_write_options	^/*
	19	1	GET	role_admin_get	^/api/v1/badges/*
	20	2	PUT	role_admin_put	^/api/v1/badges/*
	21	3	POST	role_admin_post	^/api/v1/badges/*
	22	4	PATCH	role_admin_patch	^/api/v1/badges/*
	23	5	DELETE	role_admin_delete	^/api/v1/badges/*
	24	6	OPTIONS	role_admin_options	^/api/v1/badges/*

b. output

- Output yang di allow dapat menampilkan data

Import | LIST T | GET APPLIC | GET Hello V | GET Gener | GET LIST | GET TYP | GET CER | GET REA | GET LIST | DEL DELETE | > + *** | No Environment

... TAMARA / Badges / LIST

GET | `({host})/v1/badges/list?page=0&size=10`

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

ordertype	desc
level	9
<input checked="" type="checkbox"/> page	0
<input checked="" type="checkbox"/> size	10
Key	Description

Body Cookies Headers (18) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "data": {
3     "content": [
4       {
5         "created_date": "2023-09-15T01:01:58.665+0700",
6         "updatedDate": "2023-10-20T09:45:12.566+0700",
7         "id": "bb6db3c9-56bf-4252-b32a-e7fa8c4df8c7",
8         "name": "Novice",
9         "level": "1",
10        "x0": 106,
11        "image": "https://tamara-dev-idstar.s3.ap-southeast-1.amazonaws.com/assets/Level1.png",
12        "url": "https://tamara-dev-idstar.s3.ap-southeast-1.amazonaws.com/assets/Level1.png",
13        "updatedBy": {
14          "created_date": "2023-09-15T01:01:58.515+0700",
15          "updatedDate": "2023-09-15T01:01:58.515+0700",
16          "id": 5,
17          "username": "admin@mail.com",
18          "xattr": "Artifical"
}

```

Status: 200 OK Time: 657 ms Size: 7.86 KB

- Output yang not allowed

Import | LIST T | GET APPLIC | GET Hello V | GET Gener | GET LIST | GET TYP | GET CER | GET REA | GET LIST | DEL DELETE | > + *** | No Environment

... TAMARA / Voucher/Reward / LIST

GET | `({host})/v1/voucher/list?page=0&size=10&X-Uri=/api/v1/voucher/list`

Params • Authorization Headers (8) Body Pre-request Script Tests Settings

<input checked="" type="checkbox"/> page	0
<input checked="" type="checkbox"/> size	10
<input type="checkbox"/> name	
<input checked="" type="checkbox"/> X-Uri	/api/v1/voucher/list
Key	Description

Body Cookies Headers (18) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "error": "unauthorized",
3   "error_description": "Not enough access to this endpoint"

```

Status: 401 Unauthorized Time: 666 ms

c. Penerapan

rikii2-jwtfilterpath : tambahkan dinamic filter jwt
menunggu adi :

59.Kegunaan JHipster pada api : monitoring

Noted : project actual dan project jhipster terpisah

a. Install

<https://www.jhipster.tech/installation/>

```
-g : artinya install global  
npm install -g generator-jhipster  
Proses install butuh waktu
```

The screenshot shows a Windows PowerShell window titled 'Windows PowerShell'. The command 'npm install -g generator-jhipster' is being run. A warning message from npm is displayed: 'npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.' The command completes successfully with a duration of 2741ms.

b. Create application

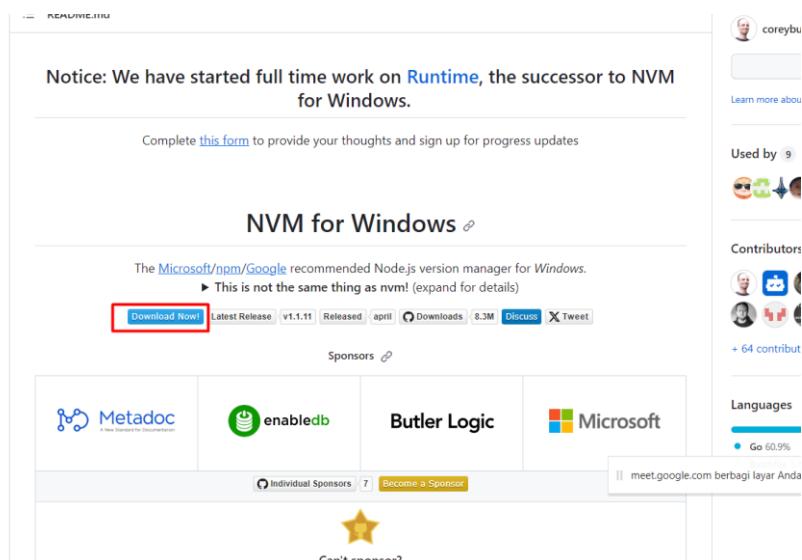
<https://www.jhipster.tech/creating-an-app/>

running jhipster dan jika eror , solusi update npm

```
PS E:\binar>IDSTAR\TAMARA\Project\tamara> jhipster  
FATAL! You are running Node version 16.16.0  
JHipster requires Node version ^18.13.0 || >= 20.2.0  
Please update your version of Node.  
PS E:\binar>IDSTAR\TAMARA\Project\tamara>
```

Isntal dulu NVM

Di <https://github.com/coreybutler/nvm-windows#readme>



Ada disini : <https://github.com/coreybutler/nvm-windows/releases>

A screenshot of the GitHub assets section for the NVM for Windows release. It lists nine assets: nvm-noinstall.zip, nvm-noinstall.zip.checksum.txt, nvm-setup.exe (highlighted with a red box), nvm-setup.zip, nvm-setup.zip.checksum.txt, nvm-update.zip, nvm-update.zip.checksum.txt, Source code (zip), and Source code (tar.gz). The nvm-setup.exe file is highlighted with a red box. Below the assets, there are reaction counts (415, 49, 74, 156, 67, 49) and a total of 631 people reacted. There are also buttons for 'Join discussion' and a comment count of 8.

Kemudian instal node js version yang dibutuhkan : misalnya version 20.
Dan buka terminal baru, kemudian klik

A screenshot of a terminal window titled 'Nvm install 20'. The command 'nvm install 20' is entered, and the output shows 'Downloading node.js version 20.8.1 (64-bit)...'. The terminal window has a dark background with white text.

c. Chek nvm list untuk melihat version terinstall

```
Nvm list
```

```
nvm use 20.8.1  
PS C:\Users\USER> nvm list  
  
 20.8.1  
 * 16.16.0 (Currently using 64-bit executable)  
PS C:\Users\USER>
```

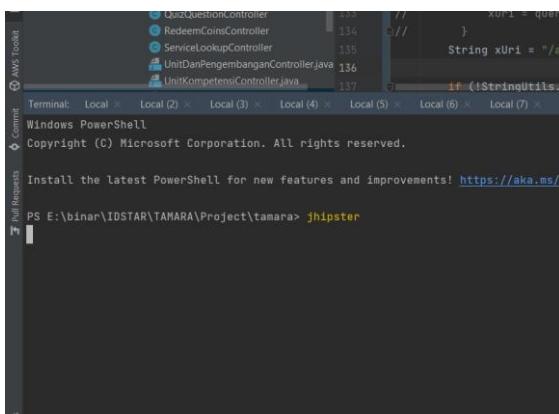
d. Cara ganti npm

```
Nvm use 20
```

```
PS C:\Users\USER> nvm use 20  
Now using node v20.8.1 (64-bit)  
PS C:\Users\USER>
```

e. Berhasil dan lakukan jalankan jhipster

f. Run jhipster

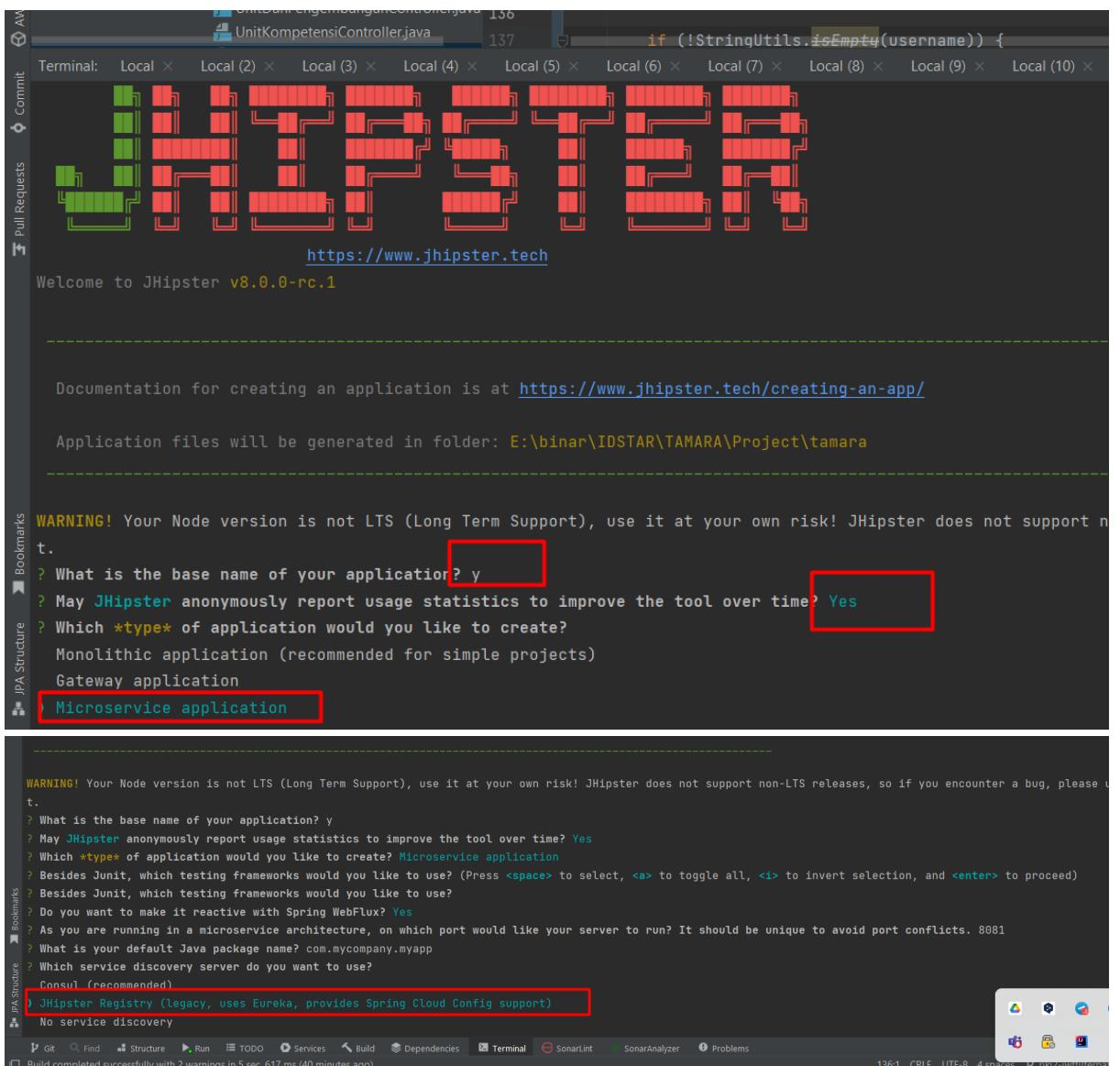


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS E:\binar>IDSTAR\TAMARA\Project\tamara> jhipster
```

g. Pilih microservice



```
https://www.jhipster.tech

Welcome to JHipster v8.0.0-rc.1

Documentation for creating an application is at https://www.jhipster.tech/creating-an-app/

Application files will be generated in folder: E:\binar>IDSTAR\TAMARA\Project\tamara

WARNING! Your Node version is not LTS (Long Term Support), use it at your own risk! JHipster does not support non-LTS releases, so if you encounter a bug, please use LTS.

? What is the base name of your application? y
? May JHipster anonymously report usage statistics to improve the tool over time? Yes
? Which *type* of application would you like to create?
  Monolithic application (recommended for simple projects)
  Gateway application
  Microservice application
```

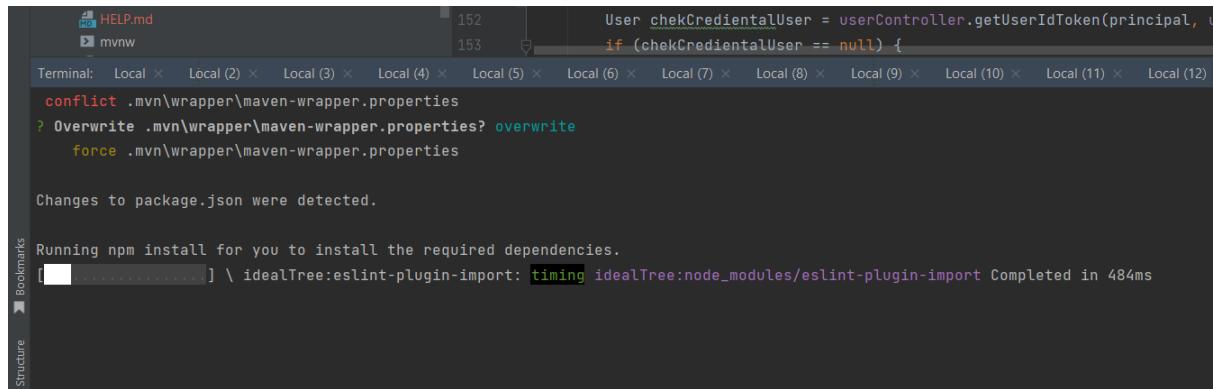
The 'Microservice application' option is highlighted with a red box.

```
WARNING! Your Node version is not LTS (Long Term Support), use it at your own risk! JHipster does not support non-LTS releases, so if you encounter a bug, please use LTS.

? What is the base name of your application? y
? May JHipster anonymously report usage statistics to improve the tool over time? Yes
? Which *type* of application would you like to create? Microservice application
? Besides JUnit, which testing frameworks would you like to use? (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
? Besides Mockito, which mocking frameworks would you like to use?
? Do you want to make it reactive with Spring WebFlux? Yes
? As you are running in a microservice architecture, on which port would you like your server to run? It should be unique to avoid port conflicts. 8081
? What is your default Java package name? com.mycompany.myapp
? Which service discovery server do you want to use?
  consul (recommended)
  JHipster Registry (Legacy, uses Eureka, provides Spring Cloud Config support)
  No service discovery
```

The 'JHipster Registry (Legacy, uses Eureka, provides Spring Cloud Config support)' option is highlighted with a red box.

Dan menunggu proses generate



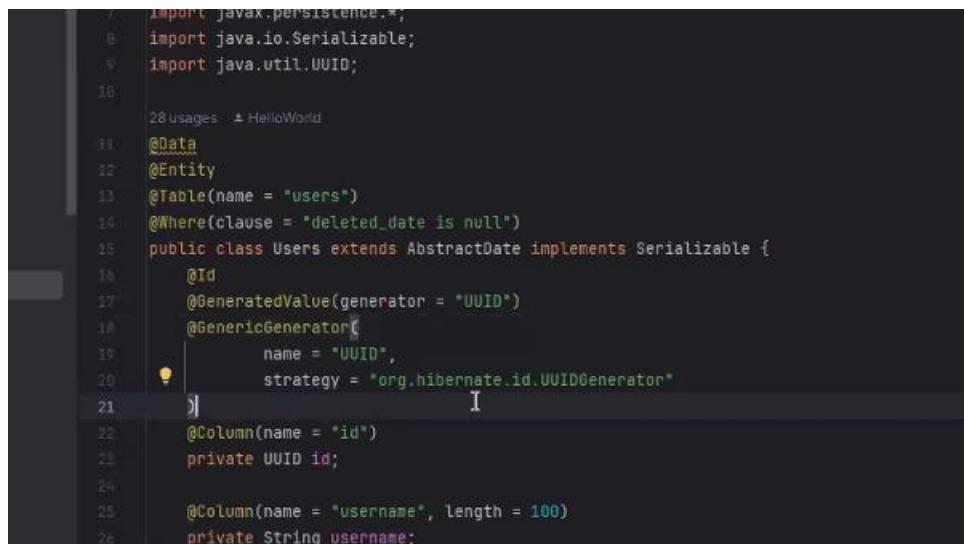
```
HELP.md
mvnw
Terminal: Local 152 Local (2) 153 Local (3) Local (4) Local (5) Local (6) Local (7) Local (8) Local (9) Local (10) Local (11) Local (12)
conflict .mvn\wrapper\maven-wrapper.properties
? Overwrite .mvn\wrapper\maven-wrapper.properties? overwrite
force .mvn\wrapper\maven-wrapper.properties

Changes to package.json were detected.

Running npm install for you to install the required dependencies.
[.....] \ idealTree:eslint-plugin-import: timing idealTree:node_modules/eslint-plugin-import Completed in 484ms
```

60. Penerapan UUID paa java 1.8 dan 11 keatas

Java 1.8 menggunakan UUID sintak seperti ini



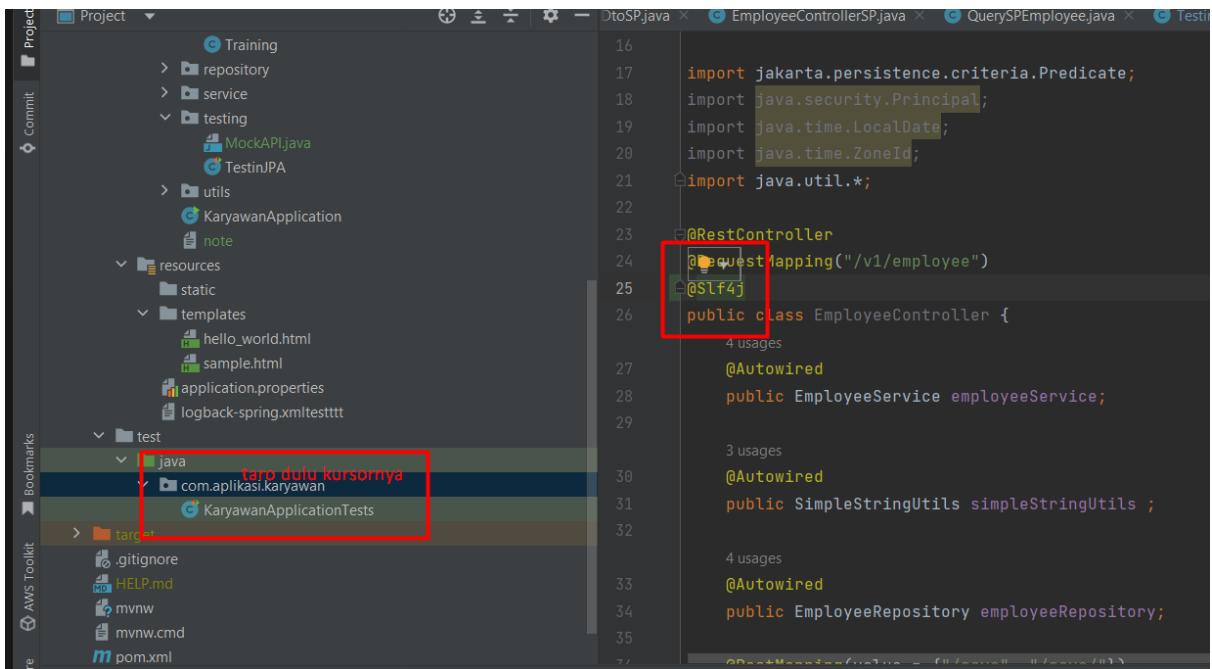
```
import javax.persistence.*;
import java.io.Serializable;
import java.util.UUID;

@Usages + HelloWorld
@Data
@Entity
@Table(name = "users")
@Where(clause = "deleted_date is null")
public class Users extends AbstractDate implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(
        name = "UUID",
        strategy = "org.hibernate.id.UUIDGenerator"
    )
    @Column(name = "id")
    private UUID id;

    @Column(name = "username", length = 100)
    private String username;
```

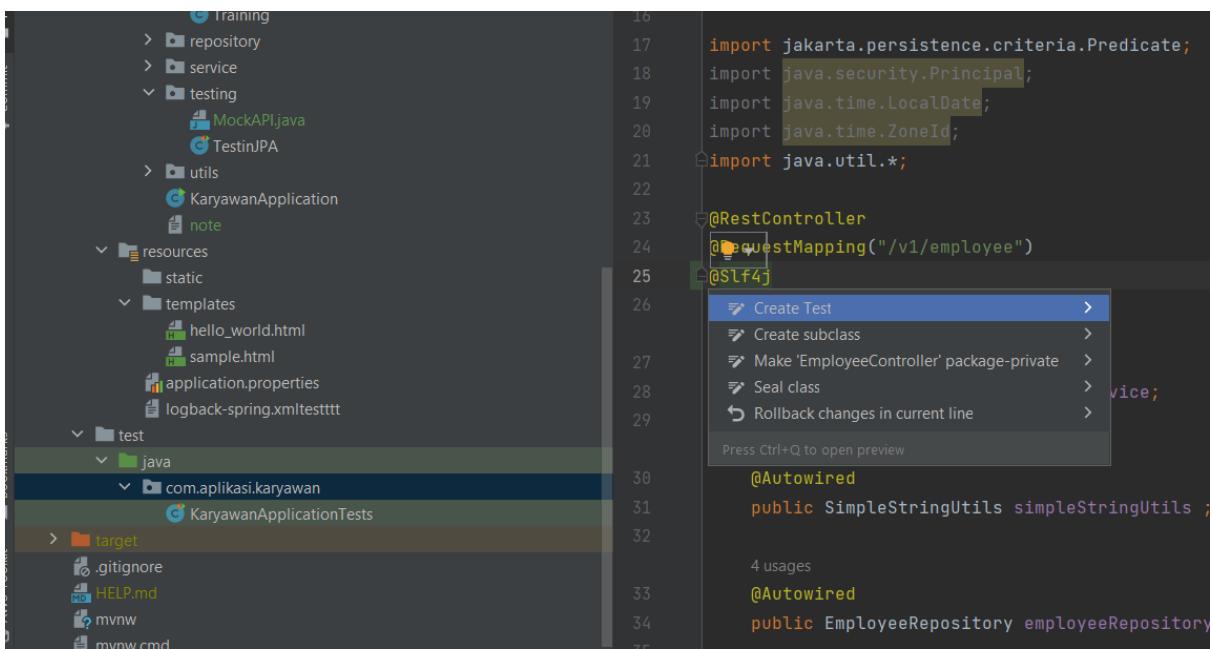
61. Cara Unit Test REST API menggunakan MockMvc

a. Tambahkan @SLF4G di controller



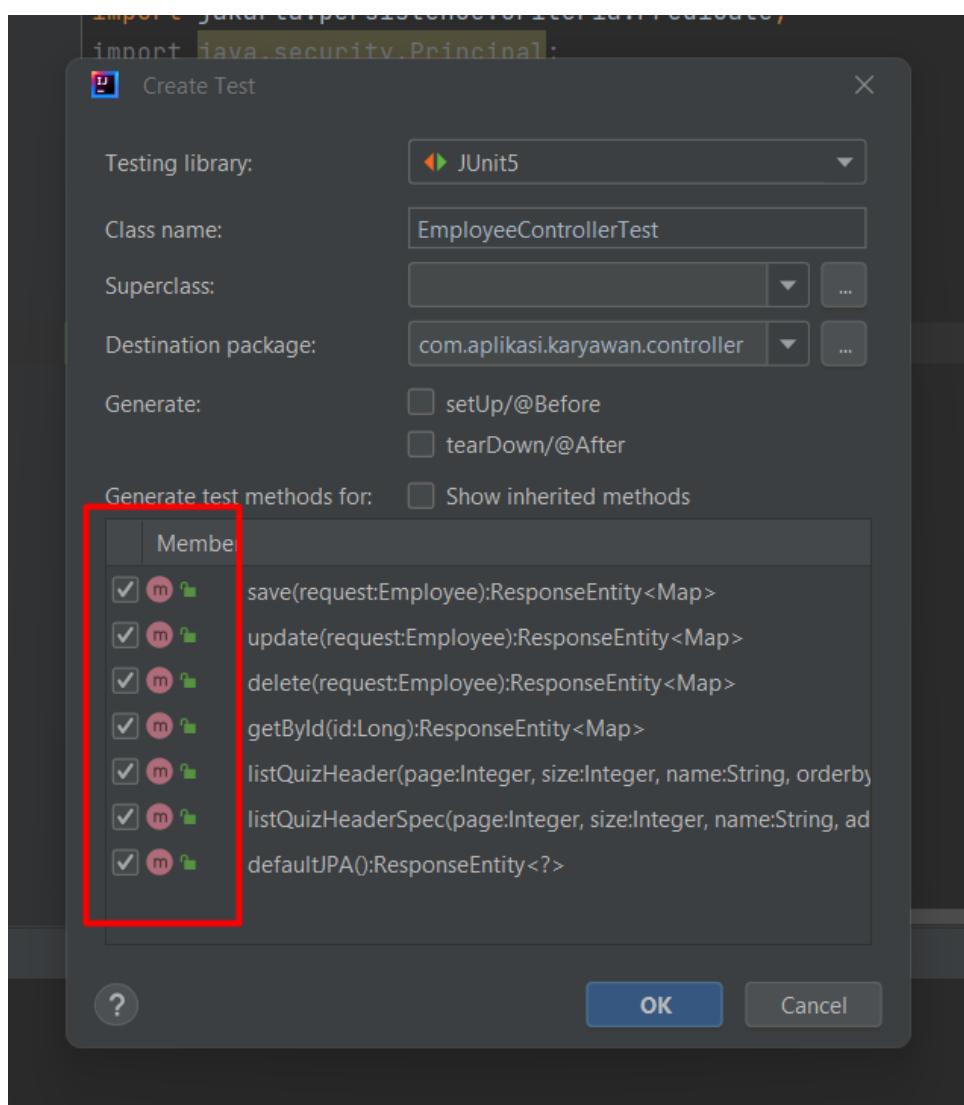
```
16
17 import jakarta.persistence.criteria.Predicate;
18 import java.security.Principal;
19 import java.time.LocalDate;
20 import java.time.ZoneId;
21 import java.util.*;
22
23 @RestController
24 @RequestMapping("/v1/employee")
25 @Slf4j
26 public class EmployeeController {
```

b. Muncul dan create mocktest

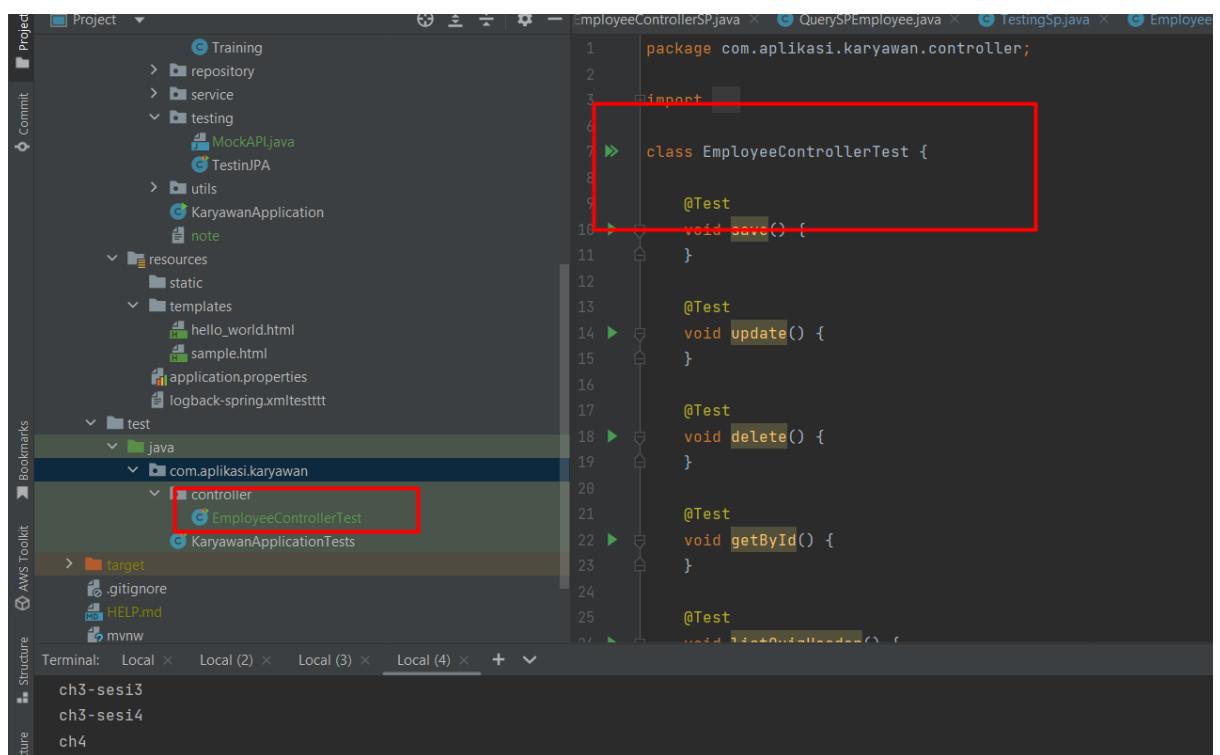


```
16
17 import jakarta.persistence.criteria.Predicate;
18 import java.security.Principal;
19 import java.time.LocalDate;
20 import java.time.ZoneId;
21 import java.util.*;
22
23 @RestController
24 @RequestMapping("/v1/employee")
25 @Slf4j
26
27 Create Test >
28 Create subclass >
29 Make 'EmployeeController' package-private >
30 Seal class >
31 Rollback changes in current line >
32
33 Press Ctrl+Q to open preview
34
35
36 @Autowired
37 public SimpleStringUtils simpleStringUtils;
```

c. Dan centang method yang dibutuhkan



d. output



```
package com.aplikasi.karyawan.controller;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class EmployeeControllerTest {

    @Test
    void save() {
    }

    @Test
    void update() {
    }

    @Test
    void delete() {
    }

    @Test
    void getById() {
    }
}
```

e. Tambahkan manual di class Controller

```
@SpringBootTest
@AutoConfigureMockMvc
@Transactional
```

```
1 package com.aplikasi.karyawan.controller;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.boot.test.autoconfigure.web.servlet.MockMvcTest;
5 import org.springframework.boot.test.context.SpringBootTest;
6 import org.springframework.transaction.annotation.Transactional;
7
8 import static org.junit.jupiter.api.Assertions.*;
9
10 @SpringBootTest
11 @AutoConfigureMockMvc
12 @Transactional
13 class EmployeeControllerTest {
14
15     @Test
16     void save() {
17         }
18
19 }
```

f. Tambahkan di body

```
@Autowired
private MockMvc mockMvc;

//untuk write dan read data json
@Autowired
private ObjectMapper objectMapper;
```

g. Importkan library ini

```
import static org.junit.jupiter.api.Assertions.*;
import static org.springframework.test.web.servlet.MockMvcBuilder.*;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static
```

```
org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
import static
org.springframework.test.web.servlet.result.MockMvcResultHandlers.*;
```

h. Code ALL

```
package com.aplikasi.karyawan.controller;

import com.aplikasi.karyawan.entity.Employee;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.jupiter.api.Test;
import org.objectweb.asm.TypeReference;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigure
MockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.transaction.annotation.Transactional;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static org.junit.jupiter.api.Assertions.*;
import static org.springframework.test.web.servlet.MockMvcBuilder.*;
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
import static
org.springframework.test.web.servlet.result.MockMvcResultHandlers.*;
import static org.junit.jupiter.api.Assertions.*;

@SpringBootTest
@AutoConfigureMockMvc
@Transaction
class EmployeeControllerTest {

    @Autowired
    private MockMvc mockMvc;

    //untuk write dan read data json
    @Autowired
    private ObjectMapper objectMapper;

    @Test
    void save() {
```

```

    }

    @Test
    void update() {
    }

    @Test
    void delete() {
    }

    @Test
    void getById() {
    }

    @Test
    void listQuizHeader() throws Exception {
        mockMvc.perform(
            get("/v1/employee/list?page=0&size=10")
                .accept(MediaType.APPLICATION_JSON)
        ).andExpect(
            status().isOk()
        ).andDo(result -> {
            System.out.println("result+"+result.getResponse().getContentAsString());
        });
        // assertNotNull(merchants);
        // assertTrue(merchants.isEmpty());
    }
}

@Test
void listQuizHeaderSpec() {

}

@Test
void defaultJPA() {

}

@Test
void testCreateEmployeeSuccess() throws Exception {
    CreateMerchantRequest request = new CreateMerchantRequest();
    request.setMerchantName("Toko ABC");
    request.setMerchantLocation("Jogja");
    request.setOpen(true);
    Map map = new HashMap();
    map.put("name", "name");
    map.put("address", "address");
    map.put("dob", "2023-01-01");
    map.put("status", "active");
    mockMvc.perform(
        post("/v1/employee/save")
            .accept(MediaType.APPLICATION_JSON)
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(map))
    ).andExpect(
        status().isOk()
    ).andDo(result -> {
}

```

```

System.out.println("result"+result.getResponse().getContentAsString());
);
}
}

// @Test
void testFlagMerchantSuccess() throws Exception {
    Merchant merchant = new Merchant();
    merchant.setMerchantName("Toko ABC");
    merchant.setMerchantLocation("Jogja");
    merchant.setOpen(true);
    Merchant savedMerchant = merchantRepository.save(merchant);

    UpdateFlagMerchantRequest request = new
UpdateFlagMerchantRequest();
    request.setFlag(true);

    mockMvc.perform(
            patch("/api/merchants/flag/{id}",
savedMerchant.getId())
                    .accept(MediaType.APPLICATION_JSON)
                    .contentType(MediaType.APPLICATION_JSON)
)
.content(objectMapper.writeValueAsString(request))
.andExpect(
        status().isOk()
).andDo(result -> {
    WebResponse<MerchantResponse> response =
objectMapper.readValue(result.getResponse().getContentAsString(), new
TypeReference<>() {});
    MerchantResponse flaggedMerchant = response.getData();

    assertNotNull(flaggedMerchant);
    assertEquals(savedMerchant.getId(),
flaggedMerchant.getId());
    assertTrue(flaggedMerchant.isOpen());
});

}

// @Test
void testGetMerchantsAvailable() throws Exception {
    // Create merchants
    merchantService.create(new CreateMerchantRequest("Toko
ABC", "Jogja", true));
    merchantService.create(new CreateMerchantRequest("Toko
XYZ", "Surabaya", false));

    mockMvc.perform(
            get("/api/merchants")
                    .accept(MediaType.APPLICATION_JSON)
)
.andExpect(
        status().isOk()
).andDo(result -> {
    WebResponse<List<MerchantResponse>> response =
objectMapper.readValue(result.getResponse().getContentAsString(), new
TypeReference<>() {});
    List<MerchantResponse> merchants = response.getData();
});
}

```

```

//           assertNotNull(merchants);
//           assertEquals(2, merchants.size()); // Only one merchant
is open
//       });
//   }
//
//   @Test
//   void testFlagMerchantNotFound() throws Exception {
//       UpdateFlagMerchantRequest request = new
UpdateFlagMerchantRequest();
//       request.setFlag(false);
//
//       // UUID acak yang tidak ada di database
//       UUID nonExistentMerchantId = UUID.randomUUID();
//
//       mockMvc.perform(
//           patch("/api/merchants/flag/{id}",
nonExistentMerchantId)
//               .accept(MediaType.APPLICATION_JSON)
//               .contentType(MediaType.APPLICATION_JSON)
//
//           .content(objectMapper.writeValueAsString(request))
//               ).andExpect(
//                   status().isNotFound()
//               );
//   }
//
//   @Test
//   void testUpdateMerchantSuccess() throws Exception{
//
//       Merchant merchant = new Merchant();
//       merchant.setMerchantName("Toko Indra");
//       merchant.setMerchantLocation("Jogja");
//       merchant.setOpen(true);
//       Merchant savedMerchant = merchantRepository.save(merchant);
//
//       UpdateMerchantRequest request = new
UpdateMerchantRequest();
//       request.setMerchantName("Toko Baru");
//       request.setMerchantLocation("Jakarta");
//       request.setOpen(false);
//
//       mockMvc.perform(
//           patch("/api/merchants/{id}", savedMerchant.getId())
//               .accept(MediaType.APPLICATION_JSON)
//               .contentType(MediaType.APPLICATION_JSON)
//
//           .content(objectMapper.writeValueAsString(request))
//               ).andExpectAll(
//                   status().isOk()
//               ).andDo(result -> {
//                   WebResponse<MerchantResponse> response =
objectMapper.readValue(
//                       result.getResponse().getContentAsString(), new
TypeReference<>() {
//                           });
//                           assertEquals("Toko Baru",
response.getData().getMerchantName());
//                           assertEquals("Jakarta",

```

```

        response.getData().getMerchantLocation());
        //           assertFalse(response.getData().isOpen());
        //       });
        //
        //
        @Test
        void testGetMerchantsEmptyList() throws Exception {
            // Tidak ada merchant yang dibuat
            //
            mockMvc.perform(
                get("/api/merchants")
                    .accept(MediaType.APPLICATION_JSON)
            ).andExpect(
                status().isOk()
            ).andDo(result -> {
                //           WebResponse<List<MerchantResponse>> response =
                objectMapper.readValue(result.getResponse().getContentAsString(), new
                TypeReference<>() {});
                //
                List<MerchantResponse> merchants = response.getData();
                //
                assertNotNull(merchants);
                assertTrue(merchants.isEmpty());
            });
        }
        //
        @Test
        void searchSuccess() throws Exception{
            //
            for (int i = 0; i<50; i++) {
            //
                Merchant merchant = new Merchant();
                merchant.setMerchantName("Toko Indra");
                merchant.setMerchantLocation("Jogjakarta");
                merchant.setOpen(true);
                merchantRepository.save(merchant);
            }
            //
            mockMvc.perform(
                get("/api/merchants")
                    .queryParam("merchantName", "Indra")
                    .accept(MediaType.APPLICATION_JSON)
                    .contentType(MediaType.APPLICATION_JSON)
            ).andExpectAll(
                status().isOk()
            ).andDo(result -> {
                //           PagingResponse<List<MerchantResponse>> response =
                objectMapper.readValue(result.getResponse().getContentAsString(), new
                TypeReference<>() {});
                //
                assertEquals(10, response.getSize());
                assertEquals(5, response.getTotalPage());
                assertEquals(0, response.getCurrentPage());
            });
            //
            mockMvc.perform(
                get("/api/merchants")
                    .queryParam("merchantLocation", "Jogja")
                    .accept(MediaType.APPLICATION_JSON)
                    .contentType(MediaType.APPLICATION_JSON)
            );
        }
    }
}

```

```
//        .andExpectAll(
//            status().isOk()
//        ).andDo(result -> {
//            PagingResponse<List<MerchantResponse>> response =
//        objectMapper.readValue(result.getResponse().getContentAsString(), new
//        TypeReference<>() {});
//
//            assertEquals(10, response.getSize());
//            assertEquals(5, response.getTotalPage());
//            assertEquals(0, response.getCurrentPage());
//        });
//
//        mockMvc.perform(
//            get("/api/merchants")
//                .queryParam("open", "true")
//                .accept(MediaType.APPLICATION_JSON)
//                .contentType(MediaType.APPLICATION_JSON)
//        ).andExpectAll(
//            status().isOk()
//        ).andDo(result -> {
//            PagingResponse<List<MerchantResponse>> response =
//        objectMapper.readValue(result.getResponse().getContentAsString(), new
//        TypeReference<>() {});
//
//            assertEquals(10, response.getSize());
//            assertEquals(5, response.getTotalPage());
//            assertEquals(0, response.getCurrentPage());
//        });
//
//    }
}
```

i. Output

A screenshot of a terminal window titled "EmployeeV2Controller". It shows the output of a test run. The logs include Hibernate insert statements and a success message indicating 8 tests passed. A red box highlights the database insertion log:

```
2023-11-20 20:01:14.540 INFO 19096 --- [main] o.s.t.c.transaction.TransactionContext : Began transaction (1) for test context [DefaultTestContext@37...]
Hibernate: insert into employee (created_date, deleted_date, updated_date, address, dob, name, status) values (?, ?, ?, ?, ?, ?)
result+{"data": {"createdDate": "2023-11-20T20:01:14.846+0700", "updatedDate": "2023-11-20T20:01:14.846+0700", "id": 37, "name": "name", "address": "address", "dob": "2023-01-01", "status": "active"}}
2023-11-20 20:01:14.922 INFO 19096 --- [main] o.s.t.c.transaction.TransactionContext : Rolled back transaction for test: [DefaultTestContext@37...]
```

j. How to Header in Mock?

```
.header("Authorization", token)
```

A screenshot of an IDE showing a Java test class named "EmployeeControllerTest". The code demonstrates how to set an "Authorization" header in a mock HTTP request using the `.header()` method. A red box highlights the header setting line:

```
String token = "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiolsib2F1dGgyLXJlc29icmNlIiwi...";  
mockMvc.perform()  
    .post(urlTemplate: "/v1/employee/save")  
    .accept(MediaType.APPLICATION_JSON)  
    .contentType(MediaType.APPLICATION_JSON)  
    .content(objectMapper.writeValueAsString(map))  
    .header(name: "Authorization", ...values token)
```

The test method is annotated with `@Test`. The test result in the IDE shows it passed with a duration of 713 ms. A red box highlights the test method name:

```
EmployeeControllerTest.testCreateEmployeeSuccess  
EmployeeControllerTest.testCreateEmployee  
testCreateEmployee 713 ms
```

62. Spring Cloud Gateway

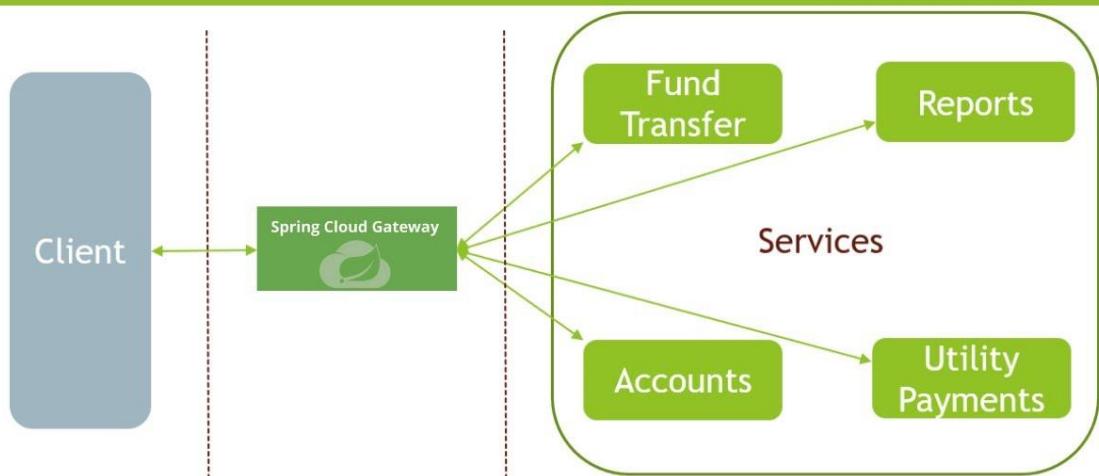
a. Referensi

<https://spring.io/projects/spring-cloud-gateway>

b. Analogi

Springboot and Microservices

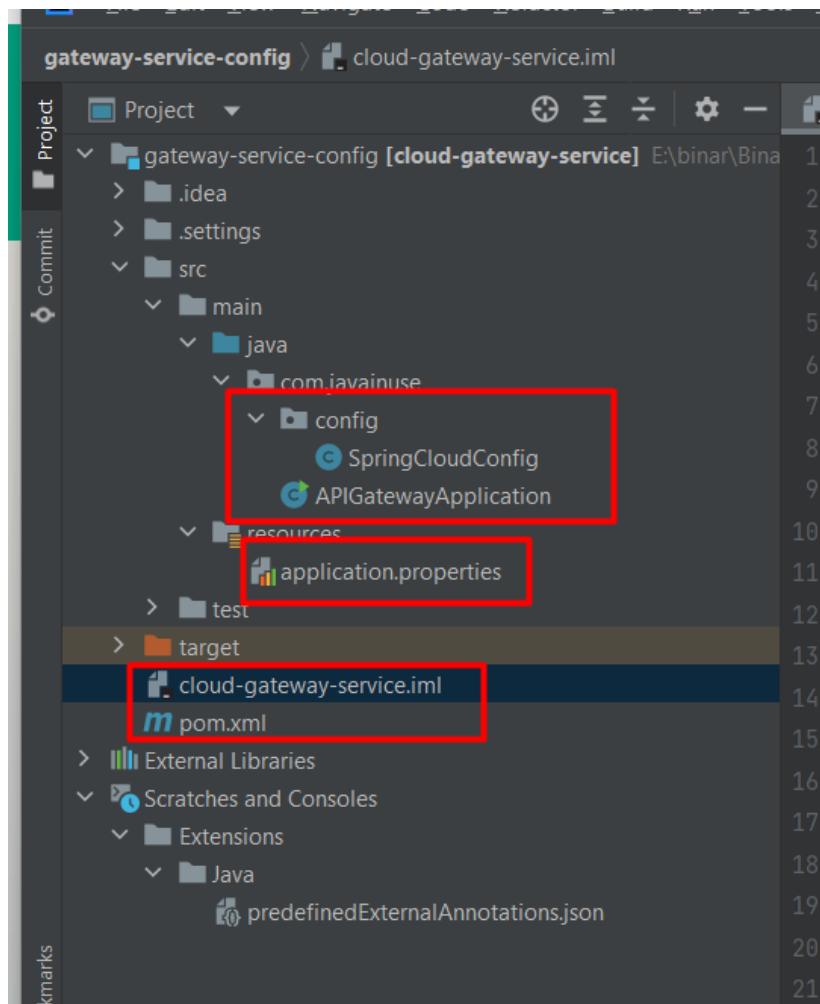
SPRING CLOUD API GATEWAY



c. Anggapan : kalian sudah punya microservice

d. Step 1 – Create New Project For Config Gateway

Struktur project



e. Step 2- POM.XML

Ingin versi maven mempengaruhi, versi cloud. Jika versi maven 2.1 maka cloud juga 2.1

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-gateway</artifactId>
    </dependency>
</dependencies>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Greenwich.SR2</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

f. Step 3 – SpringCloudConfig

```
@Configuration
public class SpringCloudConfig {

    // referensi https://spring.io/projects/spring-cloud-gateway
    @Bean
    public RouteLocator gatewayRoutes(RouteLocatorBuilder builder) {
        return builder.routes()
            .route(r -> r.path("/**"))
                .uri("http://localhost:8081/")
                .id("employeeModule"))

            .route(r -> r.path("/consumer/**"))
                .uri("http://localhost:8082/")
                .id("consumerModule"))
            .build();
    }
}
```

Penjelasan : uri diatas merupakan url microservice yang akan anda daftarkan

```
package com.javainuse.config;

import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SpringCloudConfig {

    // referensi https://spring.io/projects/spring-cloud-gateway
    @Bean
    public RouteLocator gatewayRoutes(RouteLocatorBuilder builder) {
        return builder.routes()
            .route(r -> r.path("/**"))
                .uri("http://localhost:8081/")
                .id("employeeModule"))

            .route(r -> r.path("/consumer/**"))
                .uri("http://localhost:8082/")
                .id("consumerModule"))
            .build();
    }
}
```

g. Step 4- application.properties

```
server.port=8080
```

h. step 5- cloud-gateway-service.iml

Ingat version maven mempengaruhi, version cloud. Jika version maven 2.1 maka cloud juga 2.1

```
<?xml version="1.0" encoding="UTF-8"?>
<module
  org.jetbrains.idea.maven.project.MavenProjectsManager.isMavenModule="true"
  type="JAVA_MODULE" version="4">
  <component name="NewModuleRootManager" LANGUAGE_LEVEL="JDK_1_8">
    <output url="file://$MODULE_DIR$/target/classes" />
    <output-test url="file://$MODULE_DIR$/target/test-classes" />
    <content url="file://$MODULE_DIR$">
      <sourceFolder url="file://$MODULE_DIR$/src/main/java"
        isTestSource="false" />
      <sourceFolder url="file://$MODULE_DIR$/src/main/resources"
        type="java-resource" />
      <sourceFolder url="file://$MODULE_DIR$/src/test/java"
        isTestSource="true" />
      <excludeFolder url="file://$MODULE_DIR$/target" />
    </content>
    <content url="file://$MODULE_DIR$/../src/main/java">
      <sourceFolder url="file://$MODULE_DIR$/../src/main/java"
        isTestSource="false" />
    </content>
    <content url="file://$MODULE_DIR$/../src/main/resources">
      <sourceFolder
        url="file://$MODULE_DIR$/../src/main/resources" type="java-
        resource" />
    </content>
    <content url="file://$MODULE_DIR$/../src/test/java">
      <sourceFolder url="file://$MODULE_DIR$/../src/test/java"
        isTestSource="true" />
    </content>
    <orderEntry type="inheritedJdk" />
    <orderEntry type="sourceFolder" forTests="false" />
    <orderEntry type="library" name="Maven:
      org.springframework.cloud:spring-cloud-starter-gateway:2.1.2.RELEASE"
      level="project" />
    <orderEntry type="library" name="Maven:
      org.springframework.cloud:spring-cloud-starter:2.1.2.RELEASE"
      level="project" />
    <orderEntry type="library" name="Maven:
      org.springframework.boot:spring-boot-starter:2.1.7.RELEASE" />
```

```

    level="project" />
        <orderEntry type="library" name="Maven:
org.springframework.boot:spring-boot:2.1.7.RELEASE" level="project" />
            <orderEntry type="library" name="Maven:
org.springframework:spring-context:5.1.9.RELEASE" level="project" />
                <orderEntry type="library" name="Maven:
org.springframework:spring-aop:5.1.9.RELEASE" level="project" />
                    <orderEntry type="library" name="Maven:
org.springframework:spring-expression:5.1.9.RELEASE" level="project" />
                        <orderEntry type="library" name="Maven:
org.springframework.boot:spring-boot-autoconfigure:2.1.7.RELEASE" level="project" />
                            <orderEntry type="library" name="Maven:
org.springframework.boot:spring-boot-starter-logging:2.1.7.RELEASE" level="project" />
                                <orderEntry type="library" name="Maven: ch.qos.logback:logback-classic:1.2.3" level="project" />
                                    <orderEntry type="library" name="Maven: ch.qos.logback:logback-core:1.2.3" level="project" />
                                        <orderEntry type="library" name="Maven:
org.apache.logging.log4j:log4j-to-slf4j:2.11.2" level="project" />
                                            <orderEntry type="library" name="Maven:
org.apache.logging.log4j:log4j-api:2.11.2" level="project" />
                                                <orderEntry type="library" name="Maven: org.slf4j:jul-to-slf4j:1.7.26" level="project" />
                                                    <orderEntry type="library" name="Maven:
javax.annotation:javax.annotation-api:1.3.2" level="project" />
                                                        <orderEntry type="library" name="Maven:
org.springframework:spring-core:5.1.9.RELEASE" level="project" />
                                                            <orderEntry type="library" name="Maven:
org.springframework:spring-jcl:5.1.9.RELEASE" level="project" />
                                                                <orderEntry type="library" scope="RUNTIME" name="Maven:
org.yaml:snakeyaml:1.23" level="project" />
                                                                    <orderEntry type="library" name="Maven:
org.springframework.cloud:spring-cloud-context:2.1.2.RELEASE" level="project" />
                                                                        <orderEntry type="library" name="Maven:
org.springframework.security:spring-security-crypto:5.1.6.RELEASE" level="project" />
                                                                            <orderEntry type="library" name="Maven:
org.springframework.cloud:spring-cloud-commons:2.1.2.RELEASE" level="project" />
                                                                                <orderEntry type="library" name="Maven:
org.springframework.security:spring-security-rsa:1.0.7.RELEASE" level="project" />
                                                                                    <orderEntry type="library" name="Maven: org.bouncycastle:bcpkix-jdk15on:1.60" level="project" />
                                                                                        <orderEntry type="library" name="Maven: org.bouncycastle:bcprov-jdk15on:1.60" level="project" />
                                                                                            <orderEntry type="library" name="Maven:
org.springframework.cloud:spring-cloud-gateway-core:2.1.2.RELEASE" level="project" />
                                                                                                <orderEntry type="library" name="Maven:
io.projectreactor.addons:reactor-extra:3.2.3.RELEASE" level="project" />
                                                                                                    <orderEntry type="library" name="Maven:
io.projectreactor:reactor-core:3.2.11.RELEASE" level="project" />

```

```

        <orderEntry type="library" name="Maven:
org.reactivestreams:reactive-streams:1.0.2" level="project" />
        <orderEntry type="library" name="Maven:
org.springframework.boot:spring-boot-starter-webflux:2.1.7.RELEASE"
level="project" />
        <orderEntry type="library" name="Maven:
org.springframework.boot:spring-boot-starter-json:2.1.7.RELEASE"
level="project" />
        <orderEntry type="library" name="Maven:
com.fasterxml.jackson.core:jackson-databind:2.9.9" level="project" />
        <orderEntry type="library" name="Maven:
com.fasterxml.jackson.core:jackson-annotations:2.9.0" level="project"
/>
        <orderEntry type="library" name="Maven:
com.fasterxml.jackson.core:jackson-core:2.9.9" level="project" />
        <orderEntry type="library" name="Maven:
com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.9.9"
level="project" />
        <orderEntry type="library" name="Maven:
com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.9.9"
level="project" />
        <orderEntry type="library" name="Maven:
com.fasterxml.jackson.module:jackson-module-parameter-names:2.9.9"
level="project" />
        <orderEntry type="library" name="Maven:
org.springframework.boot:spring-boot-starter-reactor-
netty:2.1.7.RELEASE" level="project" />
        <orderEntry type="library" name="Maven:
io.projectreactor.netty:reactor-netty:0.8.10.RELEASE" level="project"
/>
        <orderEntry type="library" name="Maven: io.netty:netty-codec-
http:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-
common:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-
buffer:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-
transport:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-
resolver:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-
codec:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-codec-
http2:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-
handler:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-handler-
proxy:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-codec-
socks:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-transport-
native-epoll:linux-x86_64:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven: io.netty:netty-transport-
native-unix-common:4.1.38.Final" level="project" />
        <orderEntry type="library" name="Maven:
org.glassfish:javax.el:3.0.0" level="project" />
        <orderEntry type="library" name="Maven:
org.hibernate.validator:hibernate-validator:6.0.17.Final"
level="project" />

```

```
<orderEntry type="library" name="Maven:  
javax.validation:validation-api:2.0.1.Final" level="project" />  
    <orderEntry type="library" name="Maven: org.jboss.logging:jboss-  
logging:3.3.2.Final" level="project" />  
        <orderEntry type="library" name="Maven:  
com.fasterxml:classmate:1.4.0" level="project" />  
            <orderEntry type="library" name="Maven:  
org.springframework:spring-web:5.1.9.RELEASE" level="project" />  
                <orderEntry type="library" name="Maven:  
org.springframework:spring-beans:5.1.9.RELEASE" level="project" />  
                    <orderEntry type="library" name="Maven:  
org.springframework:spring-webflux:5.1.9.RELEASE" level="project" />  
                        <orderEntry type="library" name="Maven:  
org.synchronoss.cloud:nio-multipart-parser:1.1.0" level="project" />  
                            <orderEntry type="library" name="Maven: org.slf4j:slf4j-  
api:1.7.26" level="project" />  
                                <orderEntry type="library" name="Maven:  
org.synchronoss.cloud:nio-stream-storage:1.1.3" level="project" />  
                            </component>  
                            <component name="SonarLintModuleSettings">  
                                <option name="uniqueId" value="cad18939-48e5-4eb0-b892-  
ffbfb296322c" />  
                            </component>  
</module>
```

i. Output

Client hit Router gateway , port 8080

The screenshot shows the Postman application interface. At the top, there's a navigation bar with tabs like Import, POST, Login, POST, GET, GET, and a search bar. Below the navigation is a header bar with the URL `localhost:8080/api/v1/employee/list?size=10` and a Save button. The main area has a red box around the URL field.

The request type is set to `GET`. The Headers section contains the following entries:

Key	Description
Content-Length	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.35.0
Accept	/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	<code>{{token}}</code>

The Body section shows a JSON response with one employee entry:

```
1 {  
2   "data": {  
3     "content": [  
4       {  
5         "id": 35,  
6         "name": "Novian",  
7         "address": "jakarta",  
8         "dob": "2022-12-31",  
9         "status": "active",  
10        "createdDate": "2023-11-20T19:56:14.619+0700",  
11        "updatedDate": "2023-11-20T19:56:14.619+0700"  
12      }  
13    }  
14  }  
15 }
```

At the bottom right of the body panel, there are several icons for sharing and saving the response.

Actual microservice port 8081

HTTP Karyawan / Karyawan BINAR Batch 6 / Karyawan / List

GET {{host6}}/v1/employee/list?page=0&size=10

Params • Authoriz

G host6

	INITIAL	CURRENT	Request Script	Tests	Settings
<input checked="" type="checkbox"/> Postman-To	http://localhost:9090/api	http://localhost:8081/api	evaluated when request is sent		
<input checked="" type="checkbox"/> Host	http://localhost:8081/api	http://localhost:8081/api	evaluated when request is sent		
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.35.0	PostmanRuntime/7.35.0			
<input checked="" type="checkbox"/> Accept	/*				
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br				
<input checked="" type="checkbox"/> Connection	keep-alive				
<input type="checkbox"/> Authorization	{{token}}				
Key	Value				

Body Cookies Headers (18) Test Results

200 OK

Pretty Raw Preview Visualize JSON

```

1
2   "data": {
3     "content": [
4       {
5         "createdDate": "2023-11-20T19:56:14.619+0700",
6         "updatedDate": "2023-11-20T19:56:14.619+0700",
7         "id": 35,
  
```

j. Gitlab

<https://gitlab.com/rikialdi/batch6/-/tree/ch7-sesi2-spring-cloud-gateway/>

63.Thread

Thread merupakan sebuah proses yang bakal dieksekusi di dalam sebuah program tertentu.

Thread ini adalah suatu bagian program yang nggak bergantung pada bagian lain dan bisa dijalankan secara bersama-sama tanpa saling menunggu giliran.

Hal ini berarti suatu thread bisa dimulai, dihentikan, atau diistirahatkan tanpa harus menghentikan yang lainnya.

Pas sebuah program Java dijalankan, sebenarnya ada satu thread yang secara otomatis dibuat.

Thread ini biasa disebut sebagai **thread utama** dan merupakan induk dari dari thread-thread yang lain.

Dengan adanya thread, Java memungkinkan pengguna untuk melakukan program yang menerapkan **concurrency**.

a. **Concurrency**

Emang Concurrency itu apa?

Concurrency adalah teknik pemrograman buat menyelesaikan request yang banyak tapi bisa diselesaikan dalam waktu yang sama.

Ciri utama dari proses concurrent adalah **proses satu dengan proses yang lain nggak bakal bisa dilakukan secara bersamaan di suatu resource tertentu**. Biasanya proses satu bergantian dengan proses lainnya.

Karena prosesnya cepat banget, jadi kadang terlihat kayak dilakukan bersamaan.

Analogi

Seorang pekerja restoran bakal **menumpuk semua permintaan yang datang dan diselesaikan satu persatu**.

Dengan cara demikian, kalau semua proses dilakukan sendiri, maka prosesnya bakal jadi lama karena antrian yang semakin panjang dan pekerja juga bakal kelelahan.

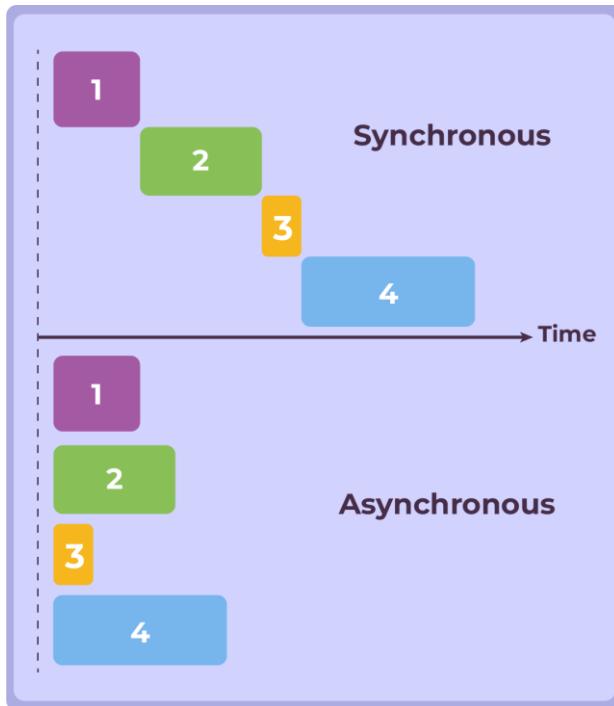


Terus solusinya harus gimana dong?

Buat memproses permintaan yang banyak, maka solusinya adalah melakukan tambahan pekerja supaya proses kerja bisa dilakukan secara bersamaan dan pelayanan terhadap permintaan pengunjung jadi lebih cepat.



synchronous VS asynchronous



Synchronous

Pas kode program kita berjalan secara sequential dan semua tahapan ditunggu sampai prosesnya selesai.

Soalnya kalau udah selesai baru lanjut ke tahap eksekusi berikutnya.

Asynchronous

PAS kode program kita berjalan dan kita nggak perlu menunggu eksekusi kode tersebut selesai.

Dikarenakan nggak perlu menunggu sampai selesai, kita bisa lanjutkan ke tahapan kode program selanjutnya.

b. Parallelism

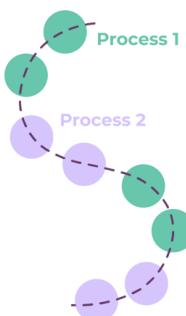
Konsep Parallelism yang mirip dengan Konsep Concurrency~

Parallel process adalah cara menyelesaikan satu permasalahan besar yang kalau dilakukan secara normal butuh waktu super lama banget.

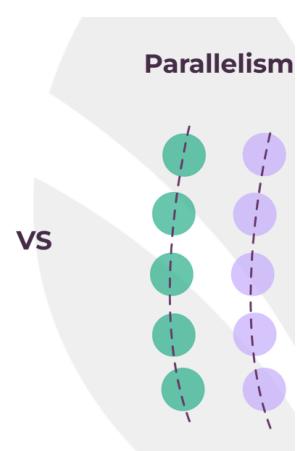
Solusi yang dilakukan, yaitu memecah masalah jadi bagian-bagian yang lebih kecil.

Masalah kecil ini terbebas dan nggak saling mempengaruhi satu sama lain dan diselesaikan secara serempak dan bersamaan.

Concurrency



Parallelism



VS

c. concurrency VS parallelism?

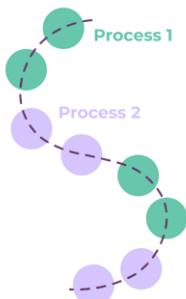
Terus apa bedanya concurrency dengan parallelism?

Concurrency artinya mengerjakan beberapa pekerjaan satu persatu pada satu waktu.

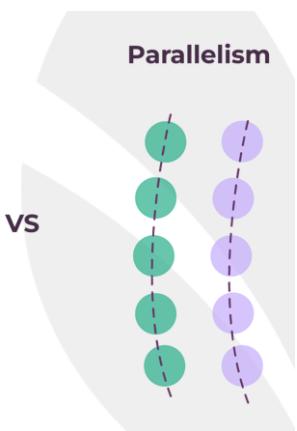
Sedangkan Parallelism yaitu mengerjakan beberapa pekerjaan sekaligus pada satu waktu.

Meskipun keduanya adalah hal yang berbeda, tapi dalam menerapkan concurrency, kita juga melakukan parallelism. Jadi, keduanya punya kaitan yang erat.

Concurrency



Parallelism



VS

d. implementasi thread di Java?

Untuk membuat thread di Java ada dua cara nih, yaitu:

1. Bikin class baru yang meng-implement interface Runnable, atau
2. Bikin class baru yang meng-extend class thread dari java.lang

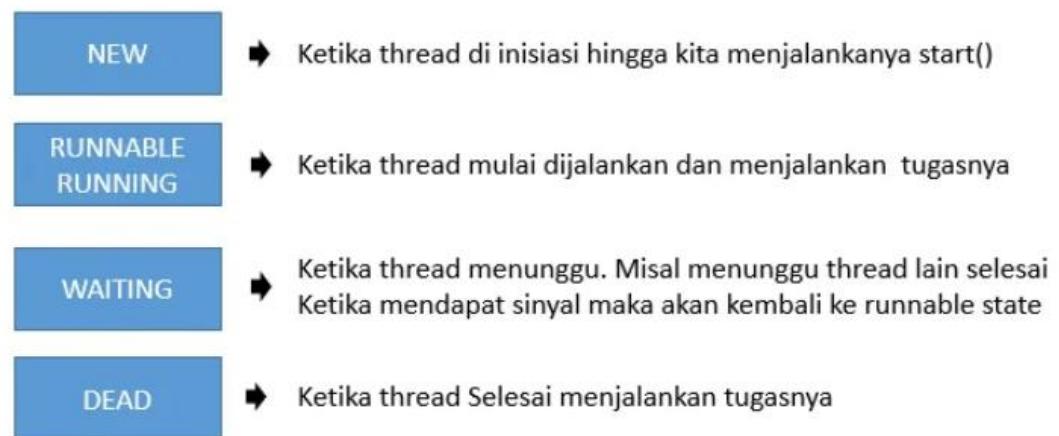
Dengan kedua cara tersebut, kita bisa bikin thread baru yang terpisah dengan thread utama.

Pas Thread berjalan, dia bakal berjalan secara asynchronous. Artinya dia berjalan sendiri dan kode program kita bakal berlanjut ke kode program selanjutnya.

Untuk menjalankan Thread, kita bisa memanggil **function start()** milik Thread.

Gampang banget, kan?!

e. Create Thread



Screenshot of an IDE showing Java code for creating a thread. The code defines a class `CallThread` with a static main method that creates and starts a new thread named `CreateThread1`.

```
package com.aplikasi.karyawan.thread;
public class CallThread {
    public static void main(String[] args) {
        Thread call = new CreateThread1();
        call.start();
    }
}
```

The run configuration in the IDE is set to `CallThread` and the command is `0.\Users\USER\.jdks\openjdk-18.0.2.1\bin\java.exe ...`. The output window shows the message "Inside : Thread0".

```
public class CreateThread1 extends Thread{
    @Override
    public void run() {
        System.out.println("Inside :
"+Thread.currentThread().getName());
    }
}
```

```
public class CallThread {  
  
    public static void main(String[] args) {  
        Thread call = new CreateThread1();  
        call.start();  
    }  
}
```

f. Create Thread Dengan Runnable

```
package com.aplikasi.karyawan.thread;  
  
public class ThreadRunnable implements Runnable{  
    @Override  
    public void run() {  
        System.out.println("Inside Runnable:  
"+Thread.currentThread().getName());  
    }  
}
```

Call

```
package com.aplikasi.karyawan.thread;  
  
public class CallThread {  
  
    public static void main(String[] args) {  
        Thread call = new CreateThread1();  
        call.start();  
  
        Runnable callRunnable = new ThreadRunnable();  
        Thread callRunnableThread = new Thread(callRunnable);  
        callRunnableThread.start();  
    }  
}
```

The screenshot shows an IDE interface with a project structure on the left and a code editor on the right. The project structure includes packages like ch4, chp3, config, controller, dot, entity, repository, request, service, testing, and thread. Under the thread package, there are files CallThread, CreateThread1, and ThreadRunnable. The code editor displays a Java file named CallThread.java with the following content:

```
package com.aplikasi.karyawan.thread;

public class CallThread {
    public static void main(String[] args) {
        Thread call = new CreateThread1();
        call.start();

        Runnable callRunnable = new ThreadRunnable();
        Thread callRunnableThread = new Thread(callRunnable);
        callRunnableThread.start();
    }
}
```

A red box highlights the line of code `Runnable callRunnable = new ThreadRunnable();` in the code editor. The run tab at the bottom shows the output of the program:

```
C:\Users\USER\.jdks\openjdk-18.0.2.1\bin\java.exe ...
Inside : Thread-0
Inside Runnable: Thread-1

Process finished with exit code 0
```

g. Sleep

```
package com.aplikasi.karyawan.thread;

import java.util.logging.Level;
import java.util.logging.Logger;

public class ThreadRunnableSleep implements Runnable{
    @Override
    public void run() {
        System.out.println("Inside Runnable: " + Thread.currentThread().getName());

        try {
            Thread.sleep(1000);
        } catch (InterruptedException ex) {

Logger.getLogger(ThreadRunnableSleep.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

h. Join, Notify, Wait

Join : Kadang kita perlu menunggu sebuah thread menyelesaikan tugasnya, kan?

Untuk melakukan hal tersebut, kita bisa memanggil method join milik thread yang mau kita tunggu.

Refer : <https://medium.com/@ranggaantok/pengenalan-multi-threading-pada-java-66127283b7f1>

i. Multi Threading with Thread Pool

<https://www.sinaungoding.com/multithreading-java-menggunakan-thread-pool/>

<https://www.baeldung.com/thread-pool-java-and-guava>



j. Next lanjut ke slide

E:\binar\Binat Batch 6\Kelas\Chapter 7 /SYNRGY 6 REFINED] BE JAVA - CH7
TOP 4 - Java Thread.pptx

64. MultiThread with Thread Pool implementation API

a. Class thread pool

```
package com.aplikasi.karyawan.thread;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.task.TaskExecutor;
import
org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;

@Configuration
public class ThreadConfiguration {
    //      https://rizkimufrizal.github.io/belajar-spring-async/
    /*
     https://stackoverflow.com/questions/17659510/core-pool-size-vs-
maximum-pool-size-in-threadpoolexecutor
     https://www.baeldung.com/java-threadpooltaskexecutor-core-vs-max-
poolsize
    */

    @Bean
    public TaskExecutor taskExecutor() {
        ThreadPoolTaskExecutor executor = new
ThreadPoolTaskExecutor();
        executor.setCorePoolSize(5); // Jumlah minimum thread yang
akan tetap ada dalam pool
        executor.setMaxPoolSize(10); // Jumlah maksimum thread dalam
pool
        executor.setQueueCapacity(250); // Kapasitas antrian untuk
tugas yang menunggu diproses
        executor.setThreadNamePrefix("custom_task_executor_thread");
// Nama thread
        executor.initialize();
        return executor;
    }

}
```

b. Create api thread pool

```
private final ExecutorService executorService =
Executors.newFixedThreadPool(10);
/*
Penjelasan singkat mengenai contoh di atas:

ApiController: Kelas ini adalah controller Spring Boot yang menangani
permintaan API. Endpoint /api/resource/{requestData} digunakan untuk
mengakses sumber daya, dan metode getResource memulai thread
menggunakan ExecutorService.
```

ExecutorService: Digunakan untuk membuat thread pool. Dalam contoh ini, digunakan `newFixedThreadPool(10)`, yang berarti akan ada maksimal 10 thread yang dapat menjalankan tugas secara bersamaan.

processRequest: Metode ini mensimulasikan pemrosesan yang memakan waktu. Di dunia nyata, di sini Anda akan menempatkan logika pemrosesan data atau tugas yang memerlukan waktu.

Dengan menggunakan `ExecutorService`, Anda dapat mengelola thread secara efisien dan mencegah pembuatan thread yang tidak terbatas, yang dapat menyebabkan masalah kinerja. Pastikan untuk menyesuaikan konfigurasi `ExecutorService` sesuai dengan kebutuhan aplikasi Anda.

```
/*
 * @GetMapping("/list-thread/{requestData}")
 * public String getResource(@PathVariable String requestData) {
 *     // Menggunakan ExecutorService untuk menjalankan thread
 * secara asinkron
 *     executorService.submit(() -> processRequest(requestData));
 *
 *     return "Request is being processed
 * asynchronously="+requestData;
 * }
 *
 * private void processRequest(String requestData) {
 *     // Logika pemrosesan yang memakan waktu
 *     try {
 *         Thread.sleep(2000);
 *         System.out.println("Processed: " + requestData);
 *     } catch (InterruptedException e) {
 *         e.printStackTrace();
 *     }
 * }
 *
 * @Qualifier("taskExecutor") // pemanggilan beans
 * @Autowired
 * private TaskExecutor taskExecutor;
 *
 * @GetMapping("/list-thread-v2/{requestData}")
 * public String getResourcev2(@PathVariable String requestData) {
 *     // Menggunakan ExecutorService untuk menjalankan thread
 * secara asinkron : backgro
 *     taskExecutor.execute(new Runnable() {
 *         @Override
 *         public void run() {
 *             //eksekusi
 *         }
 *     });
 *     return "Request is being processed asynchronously
 * v2="+requestData;
 * }
 }
```

c. Testing with Jmeter

Download jmeter

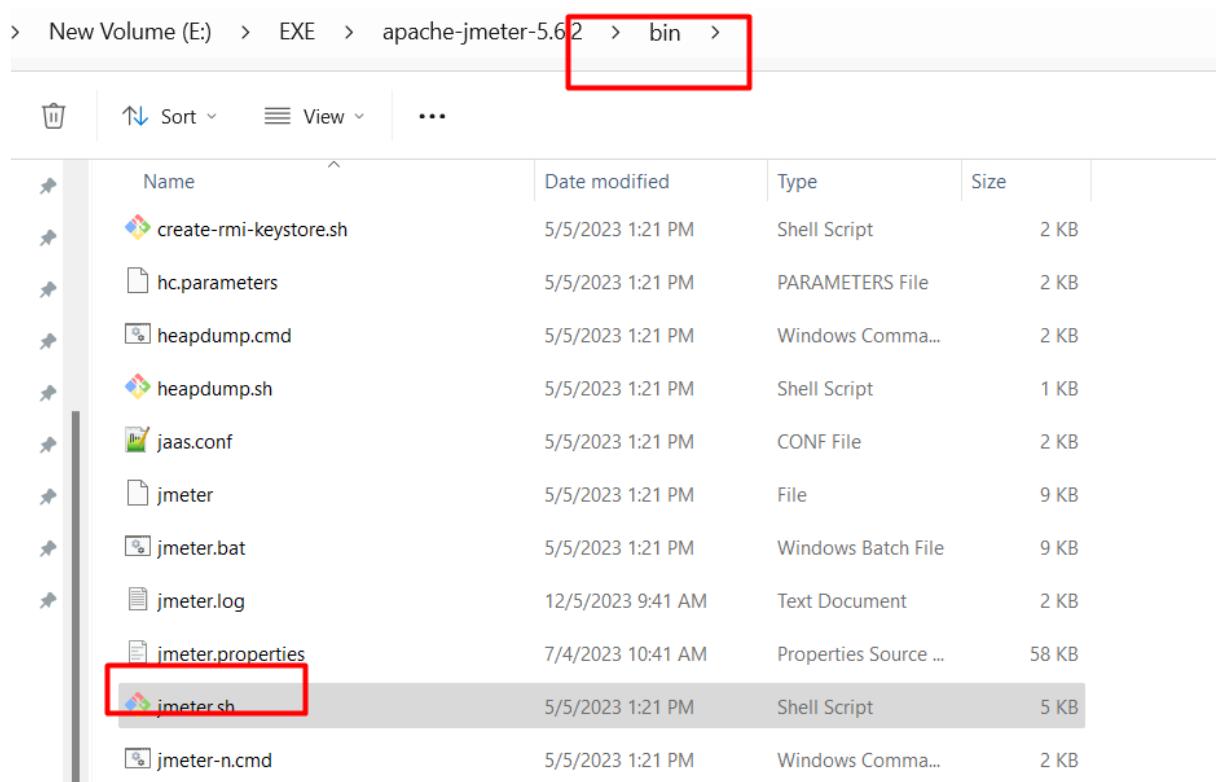
<https://jmeter.apache.org/>

documentation jmeter

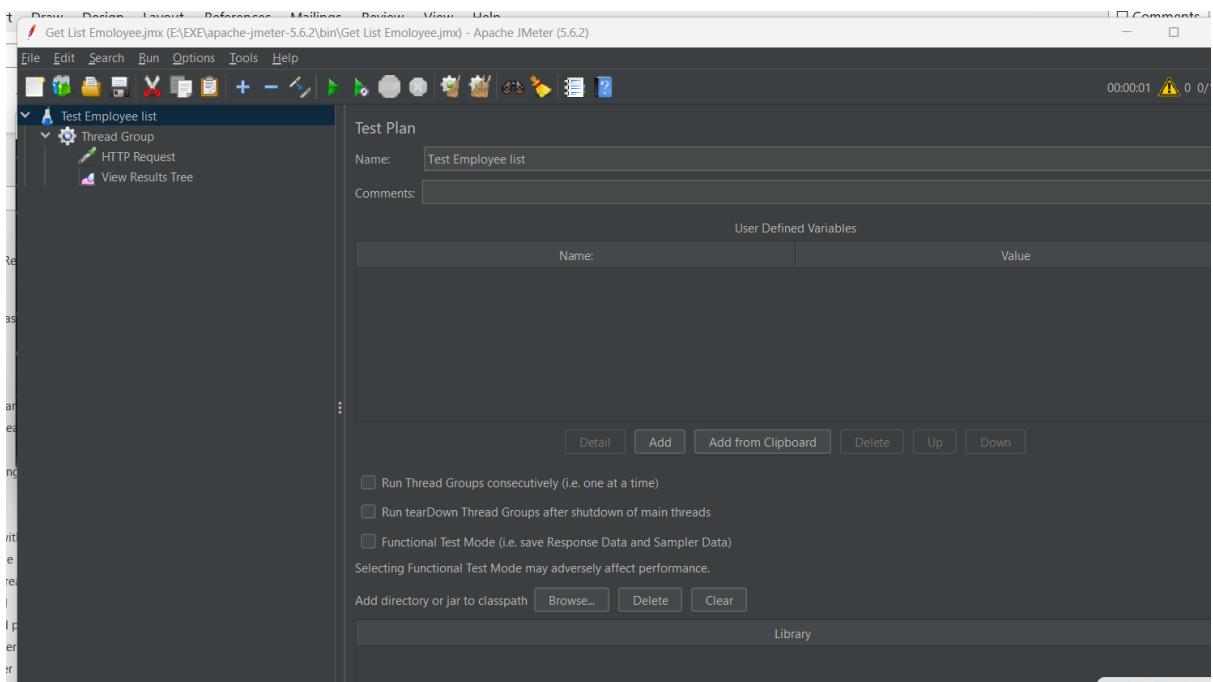
<https://jmeter.apache.org/usermanual/index.html>

<https://adityamns.medium.com/jmeter-testing-api-dan-penggunaan-token-untuk-semua-endpoint-9dc9fdc5dc53>

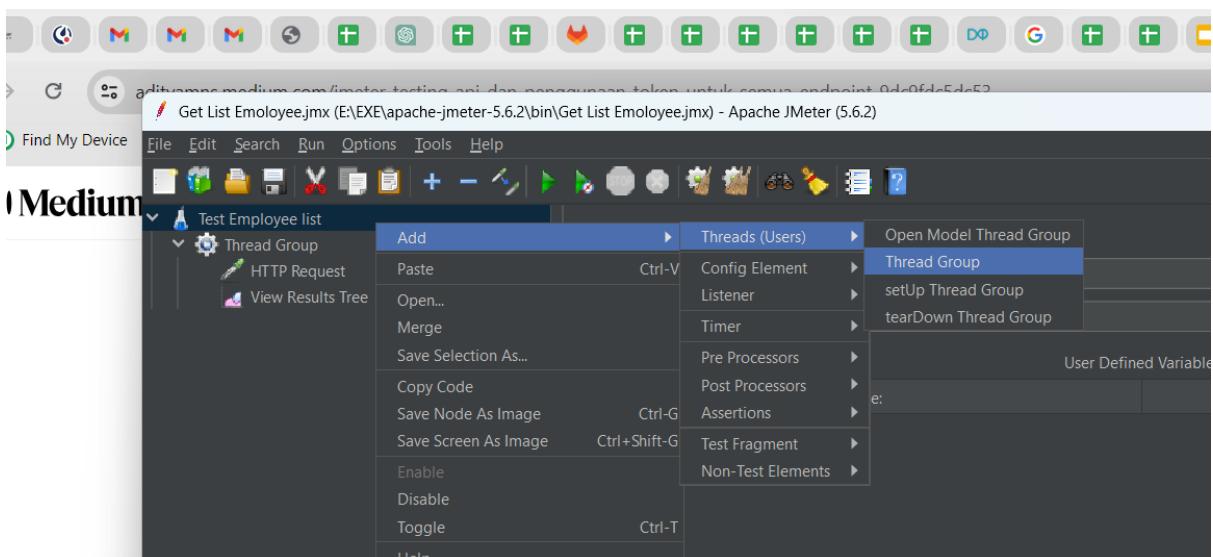
run jmeter



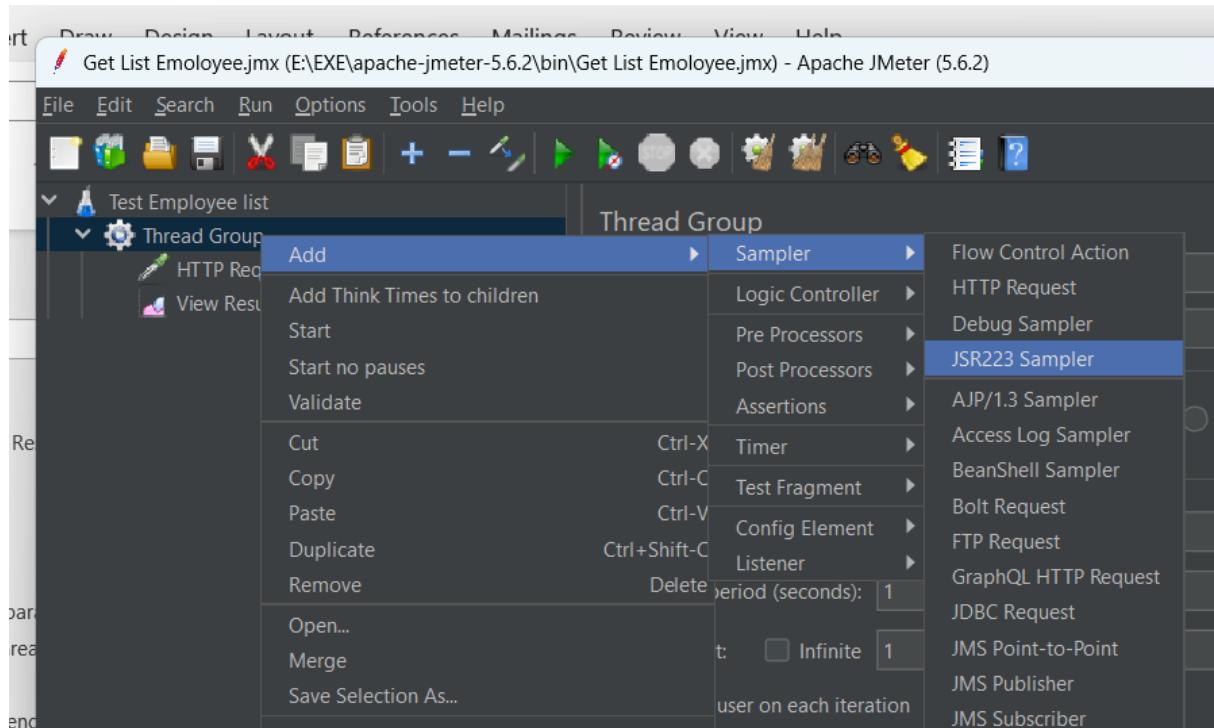
Tampilan jmeter



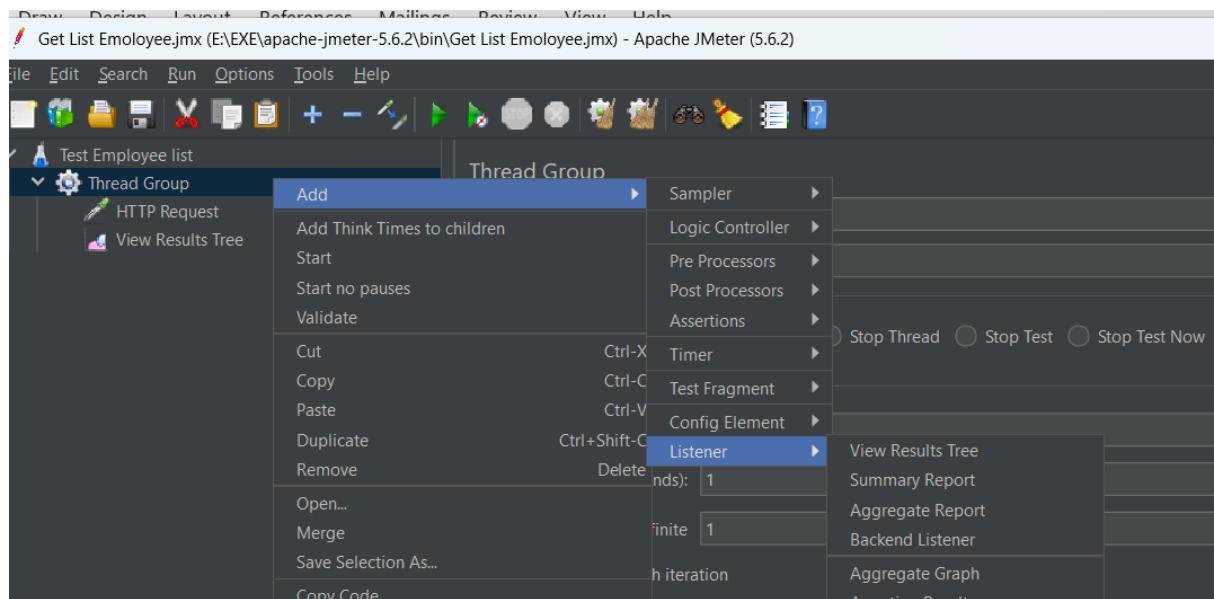
Run api- create thread group



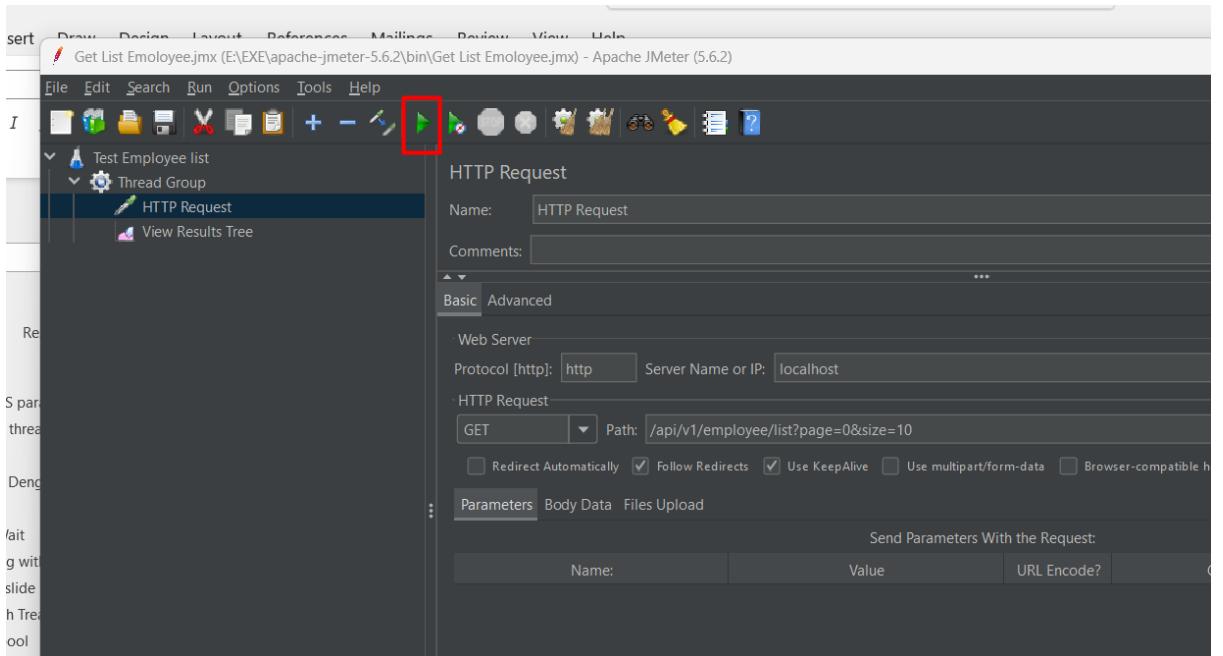
Run api – create http request



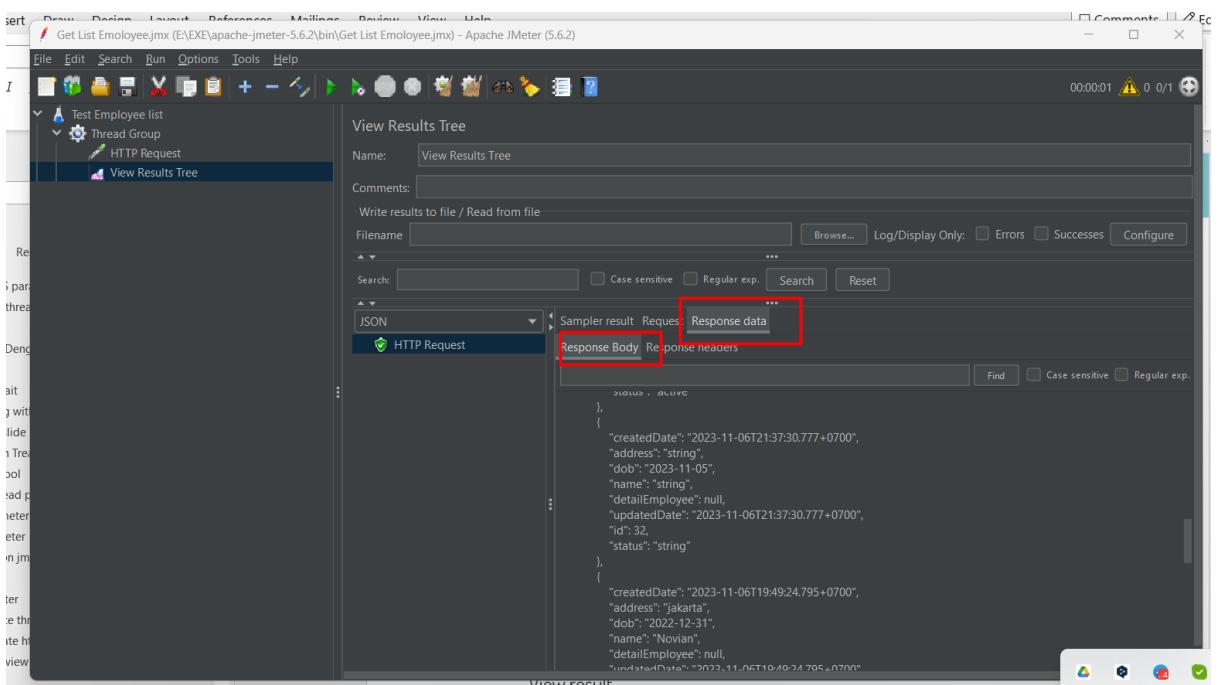
Run api- add view result tree



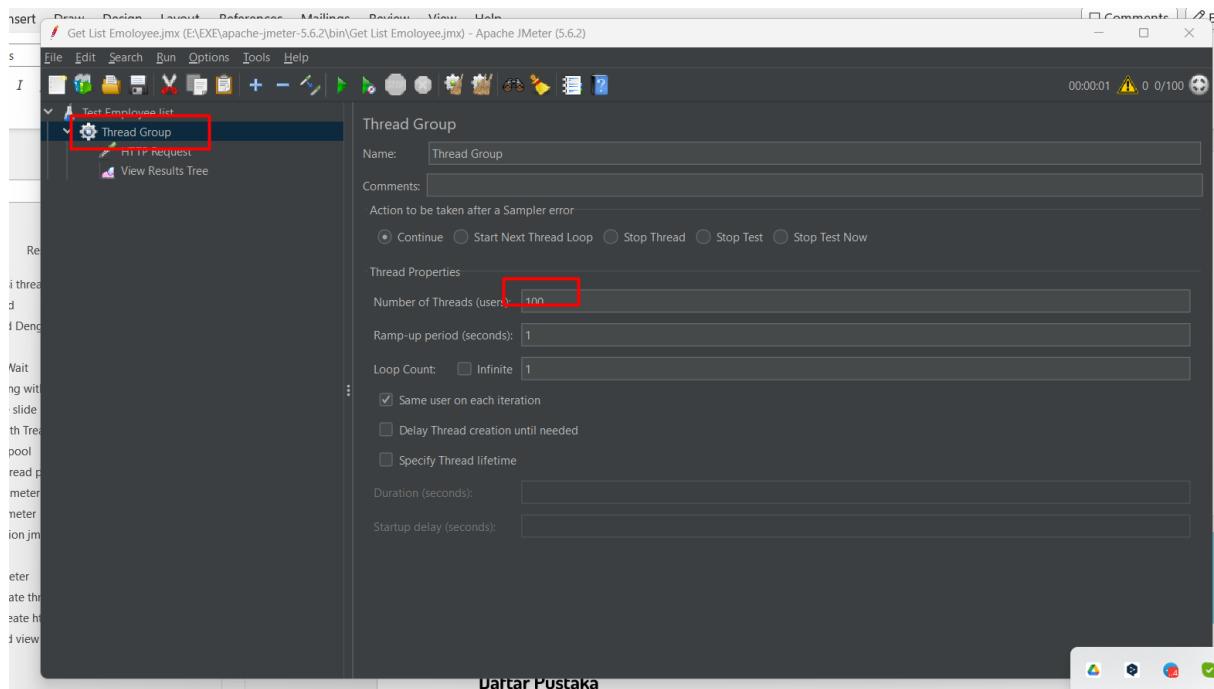
Run api



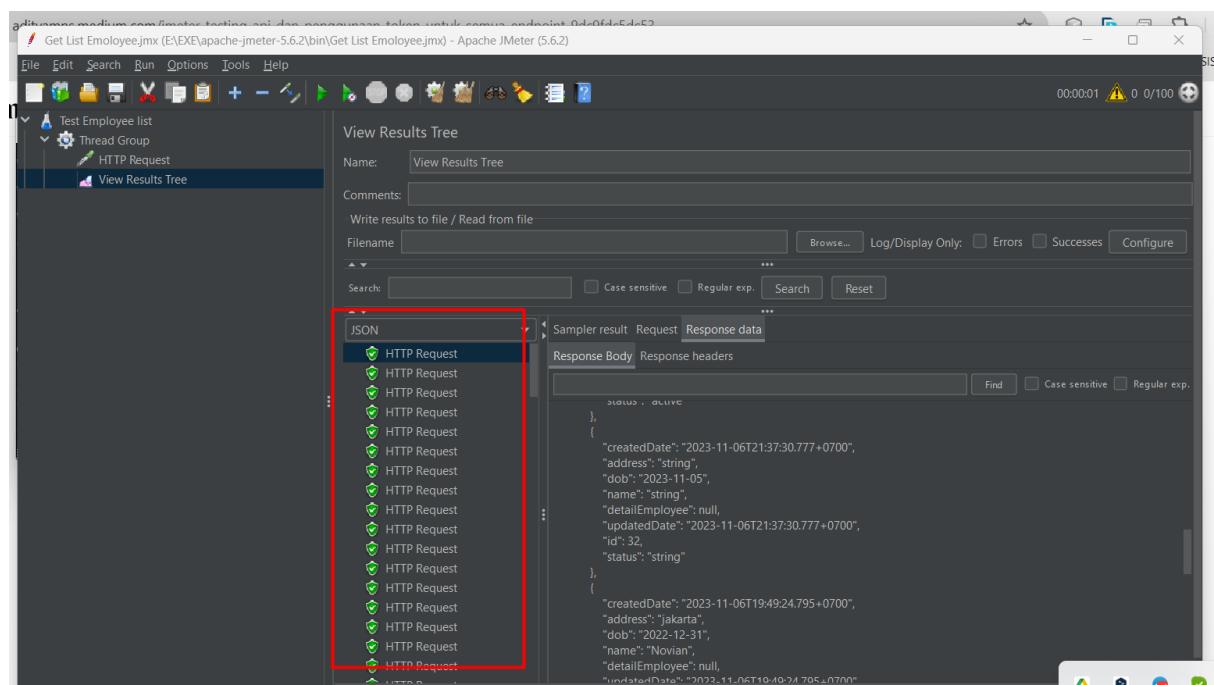
View result



d. How test performance with jmeter 100 user per second



Result view



65. Custom Authentication Provider

Custom Authentication Provider.docx

66. TokenEnchancer

TokenEnchancer.docx

67.Firebase

E:\binar\Binat Batch 6\MODUL.firebaseio\firebasedemo

68.QnA

e. Berapa limit sehari send email

500 per days

A screenshot of a Stack Overflow post. The user 'Dave Todd 9819' (Original Poster) asked the question on May 27, 2023. The question is: "How do I resolve \"550 5.4.5 Daily user sending quota exceeded\" error?". The user explains they received the error message "550 5.4.5 Daily user sending quota exceeded.", apparently indicating they had sent over 500 e-mails today. They mention this has been happening since last week and it is having an extreme adverse affect on their business. Below the question, there is a 'Details' section with 'Composing and Sending Messages, Chrome, Windows'. A 'Locked' button is shown, and a note says 'This question is locked and replying has been disabled.'

I received the "550 5.4.5 Daily user sending quota exceeded.", apparently indicating I had sent over 500 e-mails today. I've only sent SIX!! I need to figure out why this keeps happening (it started last week) and get it resolved once and for all. It is having an extreme adverse affect on my business, as I cannot communicate with my clients.

Details
Composing and Sending Messages, Chrome, Windows

Locked

This question is locked and replying has been disabled.

<https://support.google.com/mail/thread/217820095/how-do-i-resolve-550-5-4-5-daily-user-sending-quota-exceeded-error?hl=en>

Daftar Pustaka

- <https://www.it-swarm-id.com/id/authentication/apa-perbedaan-utama-antara-otentikasi-jwt-dan-oauth/828156483/>
- <https://undebugable.blogspot.com/2020/12/membuat-authentikasi-berbasis-token.html>
- <https://www.bezkoder.com/spring-boot-jwt-mysql-spring-security-architecture/>
- <https://www.bezkoder.com/jwt-json-web-token/>