

Pengembangan Backend Rest API (S3 Upload, Template, Space) Untuk Metaverse BNITopia Menggunakan Java Springboot di PT. Bank Negara Indonesia (Persero), Tbk.

1st Nurakhmad Sustantyo
Prodi S1 Terapan Teknologi Rekayasa
Multimedia
Fakultas Ilmu Terapan,
Telkom University
Bandung, Indonesia
nurakhmadsustantyo@student.telkomu
niversity.ac.id

2nd Dr. Ismail, S.Si., M.T.
Prodi S1 Terapan Teknologi Rekayasa
Multimedia
Fakultas Ilmu Terapan,
Telkom University
Bandung, Indonesia
ismailrusli@telkomuniversity.ac.id

3rd Dr. Duddy Soegiarto, S.T., M.T.
Prodi S1 Terapan Teknologi Rekayasa
Multimedia
Fakultas Ilmu Terapan,
Telkom University
Bandung, Indonesia
duddysu@telkomuniversity.ac.id

Pada laporan proyek akhir ini penulis mengangkat pengembangan REST API pada proyek BNITopia yang mana sebuah proyek yang ada di PT. Bank Negara Indonesia dimana penulis melaksanakan kegiatan magangnya. REST API yang dibuat ini nantinya digunakan pada aplikasi BNITopia. Penulis disini berfokus untuk membuat fitur Template dan Space. Pada pengembangannya penulis menggunakan metodologi Agile. API yang nantinya sudah dibuat akan digunakan oleh tim Metaverse pada aplikasi unity.

Kata Kunci: REST API, Backend, BNITopia

I. PENDAHULUAN

Metaverse akhir – akhir ini sering diperbincangkan. Hal ini disebabkan, CEO dari Facebook yaitu Mark Zuckerberg berbicara tentang teknolgi metaverse ini kedepannya akan menjadi masa depan dari internet [1]. Yang mana, hal ini membuat perusahaan – perusahaan lain belomba – lomba dalam menciptakan terknologi metaverse ini sebagai salah satu bentuk pelayanannya.

Salah satu perusahaan yang ingin menerapkan metaverse pada pelayanannya adalah PT. Bank Negara Indonesia (Persero) Tbk. Yang mana pada metaverse yang akan dibuat BNI ini akan digunakan mengenalkan produk yang ada pada BNI lewat digitalisasi. Dalam pengerjaannya ini sendiri Divisi Pengembangan Digital ditunjuk untuk melakukannya.

Divisi Pengembangan Digital ini biasanya divisi yang menangani aplikasi Mobile Banking yang dimiliki oleh BNI. Dan sekarang divisi tersebut diminta untuk membuat pelayanan bank dengan metaverse. Pada proyek metaverse ini sendiri dibagi lagi menjadi beberapa divisi yaitu 2D/3D Model dan juga Metaverse. Yang mana 2D/3D model ini ditugaskan untuk membuat asset yang nantinya digunakan didalam metaverse. Seperti karakter, gedung, dan komponen lainnya. Sedangkan divisi Metaverse ditugaskan untuk membuat mekanisme yang nantinya digunakan didalam metaverse. Seperti kontrol karakter saat bergerak, sistem

multiplayer yang nantinya digunakan didalam metaverse, dsb. Divisi Metaverse juga memiliki sub-divisi yaitu Backend. Yang mana tugas dari Backend sendiri adalah membuat database untuk menyimpan data yang nantinya dibutuhkan didalam metaverse.

II. KAJIAN TEORI

A. Profil Perusahaan

PT Bank Negara Indonesia (Persero), Tbk (selanjutnya disebut “BNI” atau “Bank”) pada awalnya didirikan di Indonesia sebagai Bank sentral dengan nama “Bank Negara Indonesia” berdasarkan Peraturan Pemerintah Pengganti Undang-Undang No. 2 tahun 1946 tanggal 5 Juli 1946. Selanjutnya, berdasarkan Undang-Undang No. 17 tahun 1968, BNI ditetapkan menjadi “Bank Negara Indonesia 1946”, dan statusnya menjadi Bank Umum Milik Negara. Selanjutnya, peran BNI sebagai Bank yang diberi mandat untuk memperbaiki ekonomi rakyat dan berpartisipasi dalam pembangunan nasional dikukuhkan oleh UU No. 17 tahun 1968 tentang Bank Negara Indonesia 1946 [2].

Berdasarkan Peraturan Pemerintah No. 19 tahun 1992, tanggal 29 April 1992, telah dilakukan penyesuaian bentuk hukum BNI menjadi Perusahaan Perseroan Terbatas (Persero). Penyesuaian bentuk hukum menjadi Persero, dinyatakan dalam Akta No. 131, tanggal 31 Juli 1992, dibuat di hadapan Muhani Salim, S.H., yang telah diumumkan dalam Berita Negara Republik Indonesia No. 73 tanggal 11 September 1992 Tambahan No. 1A [2].

BNI merupakan Bank BUMN (Badan Usaha Milik Negara) pertama yang menjadi perusahaan publik setelah mencatatkan sahamnya di Bursa Efek Jakarta dan Bursa Efek Surabaya pada tahun 1996. Untuk memperkuat struktur keuangan dan daya saingnya di tengah industri perbankan nasional, BNI melakukan sejumlah aksi korporasi, antara lain proses rekapitalisasi oleh Pemerintah di tahun 1999, divestasi

saham Pemerintah di tahun 2007, dan penawaran umum saham terbatas di tahun 2010 [3].

Ada pula visi dan misi dari BNI yaitu : [3]

Visi :

Menjadi Lembaga Keuangan yang terunggul dalam layanan dan kinerja secara berkelanjutan.

Misi :

- 1) Memberikan layanan prima dan solusi digital kepada seluruh Nasabah selaku Mitra Bisnis pilihan utama.
- 2) Memperkuat layanan internasional untuk mendukung kebutuhan Mitra Bisnis Global.
- 3) Meningkatkan nilai investasi yang unggul bagi Investor.
- 4) Menciptakan kondisi terbaik bagi Karyawan sebagai tempat kebanggaan untuk berkarya dan berprestasi.
- 5) Meningkatkan kepedulian dan tanggung jawab kepada lingkungan dan Masyarakat.
- 6) Menjadi acuan pelaksanaan kepatuhan dan tata kelola perusahaan yang baik bagi industri.

Selama pelaksanaan magang divisi yang dipetakan untuk pengerjaan proyek adalah Divisi Pengembangan Digital. Divisi ini berada di bawah Direktur IT dan Operasi yang memiliki tugas khusus dalam proses pengembangan teknologi dan layanan digital yang dimiliki BNI. Proyek magang diberikan pada salah satu unit di Divisi Pengembangan Digital yaitu unit MBC (Mobile Banking Channel). Sebagai contoh produk yang dihasilkan dari divisi ini adalah aplikasi BNI mobile.

B. Landasan Teori

Agile adalah metode yang mana membangun *software* yang didasarkan pada proses pengerjaan berulang yang terdiri dari aturan dan solusi yang sudah disepakati. Metode ini dilakukan dengan sistem kolaborasi antar tim secara terstruktur dan terorganisir. Untuk pengerjaan BNI Topia sendiri menggunakan salah satu metodologi dari agile yaitu kanban. Kanban sendiri adalah suatu metode manajemen proyek. Yang mana membuat papan, kolom, dan kartu untuk mengelola tugas dan alur kerja agar lebih efektif [4].

Metaverse adalah sebuah perangkat virtual yang dimana semua orang dapat melakukan interaksi satu sama lain secara virtual tanpa adanya kontak fisik [5]. Yang dimana *metaverse* ini biasa dijalankan menggunakan suatu perangkat yang bernama *Virtual Reality (VR)*. *Metaverse* ini sendiri sudah cukup banyak digunakan pada beberapa aplikasi seperti *Game* ataupun *NFT Marketplace*.

Springboot adalah suatu *framework* dari spring yang memberikan kemudahan untuk *programmer* memilih *library* pada java yang nantinya akan digunakan baik dari spring ataupun *library*-nya [6].

Pada umumnya untuk *Application Programming Interface (API)* itu sendiri merupakan konsep fungsi antarmuka pemrograman pada aplikasi, yang memungkinkan akses dan pemanfaatan oleh pihak lain tanpa perlu mengubah struktur kode utama maupun basis data sistem. Selain itu, API juga memfasilitasi komunikasi antar sistem, meskipun berjalan di platform yang berbeda. Namun, API sendiri membutuhkan arsitektur pengembang yang disebut REST.

Representational State Transfer (REST) sendiri ialah arsitektur standar *web* yang biasa dikenal menggunakan protokol *Hypertext Transfer Protocol (HTTP)*, yang mana biasanya berisikan *file Javascript Object Notation (JSON)*. *File-file* tersebutlah yang nantinya akan disajikan kepada para pengguna saat mengakses API-nya [7].

III. METODE

Metaverse BNITopia adalah suatu proyek *Research and Development* yang sedang dikembangkan oleh Bank Negara Indonesia (BNI). Yang mana proyek ini bertujuan untuk mengenalkan produk Bank Negara Indonesia secara digital melalui dunia *metaverse*. Proyek inilah yang ditugaskan untuk para peserta magang. Para peserta magang dibagi menjadi dua tim yaitu 2d/3d Modelling dan Metaverse. Tim Metaverse sendiri dibagi lagi menjadi dua sub-tim yang mana Frontend Metaverse dan Backend Metaverse. Yang mana para peserta magang yang terlibat di tim 2d/3d Modelling ditugaskan untuk membuat *asset* untuk dunia *metaverse*-nya. Sedangkan untuk peserta magang yang terlibat dalam tim Frontend Metaverse ditugaskan untuk mengerjakan mekanisme karakter, *multiplayer*, pembuatan dunia *metaverse*-nya dan tim Backend Metaverse membuat *database* yang nantinya diperlukan oleh tim Frontend Metaverse.

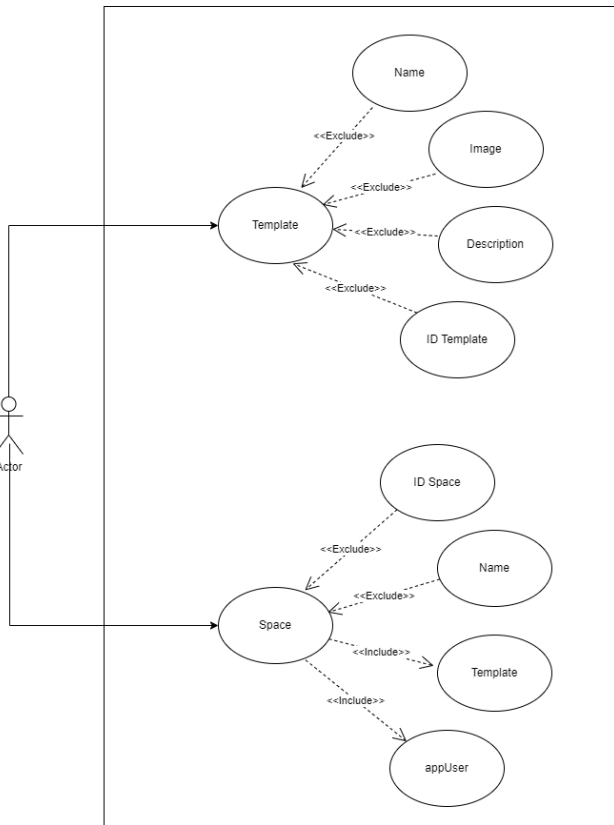
Pada saat pelaksanaan magang, mentor memberi perintah yang mana untuk *backend*-nya dapat menampung kelas template dan space. Yang mana kelas template dapat menampung data sebagai berikut :

- 1) ID
- 2) Name
- 3) Image
- 4) Description

Untuk kelas space sendiri nantinya diminta untuk menampung data sebagai berikut :

- 1) ID
- 2) Name
- 3) Template
- 4) appUser

Terdapat juga bentuk use case untuk ketentuan *requirement*-nya sendiri. Yang terdapat pada gambar 3.1.



Gambar 3. 2 Use Case Requirement

Model Template merupakan suatu kelas yang nantinya akan menjadi sebuah tabel. Tabel tersebut nantinya akan digunakan untuk menampung data template ruang yang disediakan oleh BNI. Model space sendiri nantinya juga digunakan untuk menampung data ruang metaverse milik pengguna. Seperti contoh pada gambar 3.1. nantinya setiap use case akan dijadikan atribut. Yang mana, setiap atribut nantinya akan menjadi *field* kolom pada masing – masing model dan akan menjadi sebuah tabel pada *database*.

A. MRSC Pattern

Pada proses pembuatan Rest API dibagi menjadi empat bagian. Yang mana setiap tahapannya Saling berhubungan satu dengan yang lainnya. Untuk pengimplementasian *source code* melalui beberpa tahapan yaitu model, repository, service dan juga controller. Tahapan – tahapan tersebut memiliki fungsi tersendiri.



Gambar 3. 1 MRSC Pattern

Model merupakan tahapan pembuatan entitas pada data yang menentukan struktur atau perilaku aplikasi. Yang mana nantinya entitas akan dibuat kedalam tabel *database*. *Repository* merupakan tahapan yang mengelola pengambilan data maupun manipulasi data dari sumber data seperti *database*. *Repository* biasanya diimplementasi menggunakan *framework* Spring Data JPA, yang mana sebuah *framework* yang menyediakan operasi CRUD dan metode kueri berdasarkan antarmuka *Repository*. *Service* sendiri bertindak sebagai perantara dari *Repository* dan *Controller* biasanya terdiri dari logika dan melakukan operasi pada data yang diambil melalui *Repository*. *Controller* ditugaskan untuk

menangani HTTP *request* yang ada, berinteraksi dengan *Service* dan memberikan umpan balik respon yang sesuai.

B. Space

Kelas Space ini digunakan untuk menyimpan data dari ruang *metaverse* yang dimiliki ataupun yang sudah pernah dikunjungi oleh pengguna. Yang mana nantinya pengguna tidak perlu lagi mencari – cari lagi ruang yang sudah pernah dikunjungi dan ingin dikunjungi kembali. Untuk kelas space ini sendiri menggunakan *endpoint* `"/api/v1/space"`. Berikut Usecase dari kelas space.

1) Get All Space

Fungsi ini digunakan untuk menampilkan semua *data* space yang ada pada *database*. Untuk *endpoint* `"/api/v1/space"`. Untuk proses dari *get all* ini sendiri cukup sederhana yang mana cukup memasukkan API yang disebutkan tadi maka data yang ada dalam *database* akan langsung muncul.

2) Create Space

Pada fungsi ini pengguna bisa membuat nama dan bebas memilih template yang diinginkan. Lalu data yang sudah dibuat oleh pengguna ini akan disimpan kedalam *database* berupa nama space, id akun, id template. Untuk *endpoint* yang digunakan pada fungsi ini adalah `"/api/v1/space/create"`. Untuk proses *create* datanya sendiri nantinya. Akan ada suatu form yang disediakan. Yang mana pada form tersebut diminta menginputkan nama, id appuser dan juga id template. Apabila id appuser atau id template yang diinputkan tidak ada pada *database* sebelumnya, maka akan terjadi error. Apabila data yang diinputkan semua sudah valid maka data akan masuk kedalam *database*.

3) Update Space

Pada fungsi ini pengguna bisa melakukan *update* pada space yang dimiliki seperti mengubah nama spacenya apabila nantinya sudah bosan dengan nama space yang dipunya. Nantinya data juga akan ter-*update* pada *database*. *Endpoint* yang digunakan pada fungsi ini adalah `"/api/v1/space/update"`. Untuk proses *update* ini kurang lebih hampir sama dengan *create*. Dimana nantinya akan disediakan form yang mana nantinya diminta menginputkan id space yang ingin di *update*, nama, id template dan juga id appuser. Apabila id space, id appuser ataupun id template tidak ditemukan. Maka akan terjadi error ketika ingin mengetesnya.

4) Delete Space

Untuk fungsi ini pengguna bisa menghapus space yang dipunya apabila pengguna merasa sudah bosan atau sudah lama tidak digunakan. *Endpoint* yang digunakan pada fungsi ini

adalah `"/api/v1/space/delete/{id}"`. Untuk fitur *delete* ini sendiri sebenarnya cukup sederhana. Yang mana nantinya hanya mencantumkan API yang disebutkan sebelumnya dan menambahkan id space yang ingin dihapus. Apabila id space yang diminta tidak ditemukan maka akan terjadinya error.

C. Template

Kelas *template* ini digunakan untuk menyediakan beberapa jenis *template* ruang *metaverse* yang nantinya akan disediakan oleh perusahaan nantinya dan pengguna bebas menggunakannya. Untuk *Endpoint*nya sendiri kelas ini menggunakan `"/api/v1/template"`.

1) Get All Template

Fungsi ini digunakan untuk menampilkan semua data yang ada pada *template*. *Endpoint* yang digunakan pada kelas ini kurang lebih sama dengan yang digunakan oleh kelas ini yaitu `"/api/v1/template"`. Fungsi *get all* ini sendiri cukup sederhana. Yang mana hanya menjalankan API yang disebutkan sebelumnya maka data yang ada dalam *database* akan muncul semua.

2) Create Template

Fungsi ini digunakan untuk nantinya menambahkan *template* pada kelas *template* dengan memasukkan nama, deskripsi dan gambar yang nantinya diintegrasikan dengan Amazon S3 untuk menyimpan gambar nantinya. Untuk *Endpoint* kelas ini menggunakan *endpoint* `"/api/v1/template/create"`. Untuk fungsi *create* pada *template* ini sendiri akan disediakan form yang mana nantinya bisa menginputkan nama, gambar dan juga deskripsi. Apabila inputan yang diinputkan tidak valid, maka akan muncul pesan error dan diminta untuk menginputkan data kembali. Apabila data valid akan muncul pesan berhasil dan data yang diinputkan tadi akan masuk kedalam *database*.

3) Update Template

Fungsi ini digunakan untuk melakukan *update* pada *template* seperti merubah nama, deskripsi dan juga gambar pada *template*. Untuk *endpoint* yang digunakan pada fungsi ini adalah `"/api/v1/template/update"`. Fungsi *update* *template* ini dilakukan dengan cara mengisi form id *template* yang ingin di *update*, nama *template*, gambar untuk *template* dan juga deskripsi untuk *templat*nya. Apabila inputan yang dimasukkan tidak valid maka akan terjadi error dan diminta untuk melakukan input ulang. Apabila inputan valid maka akan muncul pesan berhasil dan data yang tadi ingin di *update* maka akan ter-*update* pada *database*.

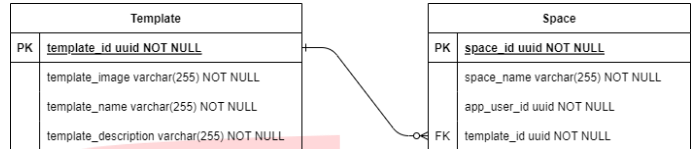
4) Delete Template

Fungsi ini digunakan untuk menghapus *template* apabila *template* sudah usang atau tidak pernah digunakan maka *template* yang tersedia nantinya bisa dihapus. *Endpoint* yang digunakan fungsi ini adalah `"/api/v1/template/delete/{id}"`. Untuk fungsi

delete data *template* ini sendiri cukup sederhana. Yang mana nantinya cukup mencantumkan API yang sudah ada lalu tinggal menambahkan id data yang ingin dihapus pada bagian akhir API.

D. ER Diagram

Setelah melakukan perancangan pada Model *Template* dan *Space* maka akan menghasilkan 2 buah tabel yang mempunyai relasi *one to many*, yang mana satu *Template* bisa digunakan banyak *Space* bisa dilihat pada gambar 3.3.



Gambar 3. 3 ER Diagram

IV. HASIL DAN PEMBAHASAN

A. Implementasi

Implementasi dari pekerjaan ini sendiri adalah melaksanakan perencanaan yang tadi sudah disusun secara terperinci. *Backend* yang dibuat oleh penulis terdiri dari 2 kelas yaitu *Template* dan *Space*. yang mana kelas – kelas tersebut sudah dibicarakan kembali dan sudah disetujui oleh perusahaan dan juga para mentor.

1) Template

Tabel 4. 1 Implementasi Kelas *Template*

Implementasi Pembuatan <i>backend</i> Pada Kelas <i>Template</i>	
<p>Model Class</p> <p>Berikut model yang digunakan pada kelas <i>template</i>. Merupakan data – data yang harus diisi.</p>	<pre>@Id @GeneratedValue(generator = "UUID") @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUID6") private UUID id; 1 usage private String name; no usages private String image; 1 usage private String description;</pre>
<p>Get All Function</p> <p>Dibuatnya fitur <i>get all</i> ini sendiri yaitu agar memudahkan pengguna untuk menemukan semua jenis <i>template</i>.</p>	<pre>@GetMapping("") public List<Template> findAll() { return templateSe</pre>

<p>Create Function</p> <p>Dibuat fitur ini agar admin bisa menambahkan template yang nantinya bisa digunakan oleh pengguna – pengguna lain.</p>	<pre> if (result.hasErrors()) { output.put("Id", null); output.put("Status", "Create data failed"); output.put("Errors", result.getAllErrors()); return ResponseEntity.badRequest().body(output); } else { Template data = new Template(); data.setName(template); data.setDescription(description); data.setImage(url); this.templateService.save(data); output.put("Status", "Create data success"); return ResponseEntity.ok().body(output); } </pre>
<p>Update Function</p> <p>Fitur ini digunakan untuk admin bisa melakukan pembaharuan pada template. Seperti memperbarui nama template, mengganti gambar ataupun mengganti deskripsi templatanya.</p>	<pre> if (result.hasErrors()){ output.put("Status", "Update data failed"); output.put("Errors", result.getAllErrors()); return ResponseEntity.badRequest().body(output); } else { try { Template data = new Template(); data.setName(template); data.setDescription(description); data.setImage(url); data.setId(UUID.fromString(id)); templateService.save(data); output.put("Status", "Update data success"); return ResponseEntity.ok(output); } catch (EmptyResultDataAccessException e){ output.put("Status", "Id not found"); return ResponseEntity.badRequest().body(output); } } </pre>
<p>Delete Function</p> <p>Fungsi ini digunakan untuk memudahkan admin menghapus template yang sudah jarang digunakan atau bahkan tidak pernah digunakan lagi.</p>	<pre> try { templateService.delete(id); output.put("Status", "Data has been deleted"); return ResponseEntity.ok(output); } catch (EmptyResultDataAccessException e){ output.put("Status", "Id not found"); return ResponseEntity.badRequest().body(output); } </pre>

2) Space

Tabel 4. 2 Implementasi Kelas Space

Implementasi Pembuatan backend Pada Kelas Space	
<p>Model Class</p> <p>Berikut model kelas yang digunakan pada kelas space. Model – model tersebut digunakan untuk menampung data.</p>	<pre> @Entity @GeneratedValue(generator = "UUID") @GeneratedValue(generator = "UUID", strategy = "org.hibernate.id.UUIDGenerator") private UUID id; 1 usage private String name; no usages private Integer likes; 1 usage @ManyToOne @JoinColumn(nullable = false, name = "template_id") private Template template; 1 usage @ManyToOne @JoinColumn(nullable = false, name = "app_user_id") private AppUser appUser; </pre>
<p>Get All Function</p> <p>Fitur ini digunakan untuk mempermudah admin untuk mengetahui Space apa saja yang ada pada database dengan menampilkan semua Space</p>	<pre> @GetMapping("") public List<Space> findAll() { return spaceService.findAll(); } </pre>
<p>Create Function</p> <p>Digunakan agar pengguna bisa membuat Space dengan nama yang diinginkan pengguna.</p>	<pre> if (result.hasErrors()) { output.put("Id", null); output.put("Status", "Create data failed"); output.put("Errors", result.getAllErrors()); return ResponseEntity.badRequest().body(output); } else { Space _space = new Space(space.getName(), space.getAppUser(), space.getTemplate(), space.getVisitor()); _space.setVisitor(0); _space.setLikes(0); this.spaceService.save(_space); output.put("status", "Create data success"); return ResponseEntity.ok().body(output); } </pre>

<p>Update Function</p> <p>Digunakan agar pengguna bisa memperbaharui nama Spaceny dengan yang baru apabila pengguna bosan dengan nama yang lama</p>	<pre> if (result.hasErrors()){ output.put("Status", "Update data failed"); output.put("Errors", result.getAllErrors()); return ResponseEntity.badRequest().body(output); }else { try { spaceService.findById(space.getId()); spaceService.save(space); output.put("Status", "Update data success"); return ResponseEntity.ok(output); } catch (EmptyResultDataAccessException e){ output.put("Status", "Id not found"); return ResponseEntity.badRequest().body(output); } } </pre>
<p>Delete Function</p> <p>Fungsi ini dibuat agar mempermudah pengguna jika pengguna ingin menghapus Space yang dipunyai</p>	<pre> try { spaceService.delete(id); output.put("Status", "Data has been deleted"); return ResponseEntity.ok(output); }catch (EmptyResultDataAccessException e){ output.put("Status", "Id not found"); return ResponseEntity.badRequest().body(output); } </pre>

	yang ada dalam database	space muncul - status http code: 200	space muncul - status http code: 200	
2	Menambah data baru pada Space ke dalam database	- Muncul pesan berhasil - Data masuk kedalam database - Status http code: 200	- Pesan berhasil muncul - Data baru berhasil masuk kedalam database - Status http code: 200	Berhasil
3	Melakukan Pembaharuan data pada Space	- Muncul pesan Berhasil - Data pada database berhasil diperbarui - Status http code: 200	- Pesan berhasil muncul - Data pada database berhasil diperbarui - Status http code: 200	Berhasil
4	Menghapus data pada Space	- Data berhasil dihapus dari database - Pesan data berhasil	- Data berhasil dihapus dari database - Pesan data berhasil	Berhasil

B. Hasil

Pada pengujian projek *backend metaverse* ini dilakukan pengujian dan diverifikasi oleh mentor dengan menentukan valid atau tidaknya kode yang sudah dibuat selama kegiatan magang berlangsung. Yang mana pengujian ini menggunakan aplikasi Postman untuk menguji masing – masing API-nya

Tabel 4. 3 Hasil Pengujian Pada Backend

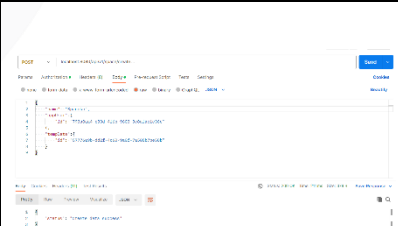
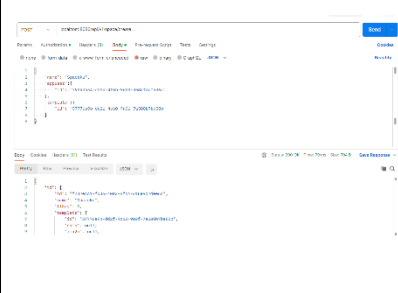
No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil yang Didapatkan	Status
1	Menampilkan semua data Space	- Semua data	- Semua data	Berhasil

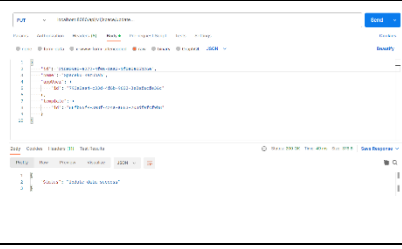
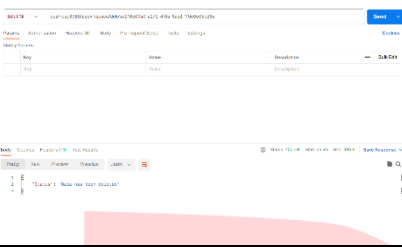
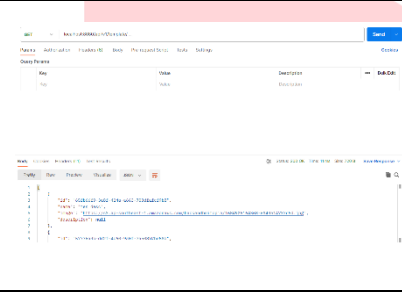
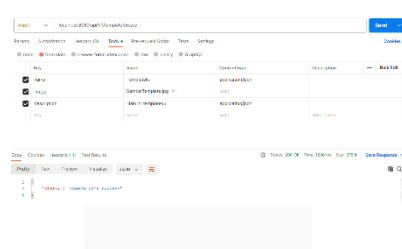
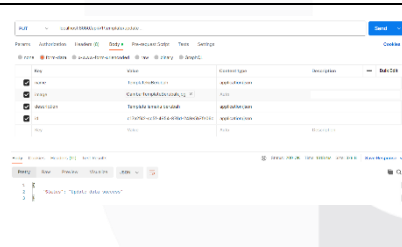
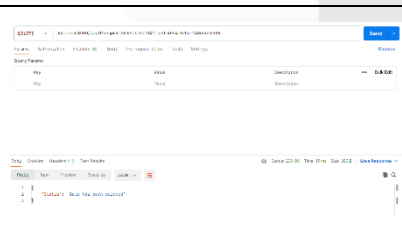
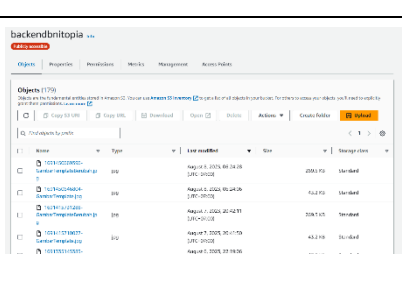
		dihapus muncul - Status http code: 200	dihapus muncul - Status http code: 200	
5	Menampilkan semua data Template yang ada pada <i>database</i>	- Berhasil memunculkan semua data yang ada pada tabel <i>database</i> Template - Status http code: 200	- Berhasil memunculkan semua data yang ada pada tabel <i>database</i> Template - Status http code: 200	Berhasil
6	Menambahkan data Template baru kedalam <i>database</i>	- Data baru berhasil ditambahkan kedalam <i>database</i> - Muncul pesan berhasil - Status http code: 200	- Data baru berhasil ditambahkan kedalam <i>database</i> - Muncul pesan berhasil - Status http code: 200	Berhasil
7	Melakukan pembaharuan data pada Template	- Muncul pesan berhasil - Data dalam <i>database</i> berhasil	- Muncul pesan berhasil - Data dalam <i>database</i> berhasil	Berhasil

		diperbarui - Status http code: 200	diperbarui - Status http code: 200	
8	Menghapus data yang ada di <i>database</i> pada kelas Template	- Muncul pesan data berhasil dihapus dari <i>database</i> - Data berhasil dihapus dari <i>database</i> - Status http code: 200	- Muncul pesan data berhasil dihapus dari <i>database</i> - Data berhasil dihapus dari <i>database</i> - Status http code: 200	Berhasil

Dan juga berikut penulis sertakan tangkapan layar dari pengujian yang dilakukan pada tabel 4.3 tadi sebagai berikut.

Tabel 4. 4 Hasil Pengujian Pada Postman

N o	Nama	Gambar
1	Menampilkan semua data Space yang ada dalam <i>database</i>	
2	Menambahkan data baru pada Space ke dalam <i>database</i>	

3	Melakukan Pembaharuan data pada Space	
4	Menghapus data pada Space	
5	Menampilkan semua data Template yang ada pada database	
6	Menambahkan data Template baru kedalam database	
7	Melakukan pembaharuan data pada Template	
8	Menghapus data yang ada di database pada kelas Template	
9	Contoh hasil data foto ketika masuk ke dalam AWS S3	

Pengerjaan proyek *backend* REST API pada kelas Template dan Space untuk *metaverse* BNITopia dengan menggunakan metode agile telah berhasil diselesaikan. Semua hasil pengujian sudah sesuai dengan keinginan perusahaan melalui verifikasi dari mentor.

REFERENSI

- [1] K. Hays, "Why is Mark Zuckerberg so obsessed with building the metaverse? It has a lot to do with Apple.," 27 Oktober 2022. [Online]. Available: <https://www.businessinsider.com/mark-zuckerberg-metaverse-push-about-wresting-control-from-apple-facebook-2022-10#:~:text=Mark%20Zuckerberg%20is%20willi ng%20to,escape%20the%20clutches%20of%20Apple.> [Accessed 22 Maret 2023].
- [2] "Sejarah," PT. Bank Negara Indonesia (Persero), Tbk., [Online]. Available: <https://www.bni.co.id/id-id/perseroan/tentang-bni/sejarah.> [Accessed 23 Maret 2023].
- [3] "Visi & Misi," PT. Bank Negara Indonesia (Persero), Tbk., [Online]. Available: <https://www.bni.co.id/id-id/perseroan/tentang-bni/visi-misi.> [Accessed 23 Maret 2023].
- [4] J. Martins, "Apa itu papan Kanban? Panduan pemula.," asana, 10 Oktober 2022. [Online]. Available: <https://asana.com/id/resources/what-is-kanban.> [Accessed 23 Maret 2023].
- [5] "MENGENAL APA ITU METAVERSE DAN CARA KERJANYA," Nagitec, [Online]. Available: [https://nagitec.com/mengenal-apa-itu-metaverse-dan-cara-kerjanya/.](https://nagitec.com/mengenal-apa-itu-metaverse-dan-cara-kerjanya/) [Accessed 24 Maret 2023].
- [6] D. H. F. Rizal, "Spring Boot," 27 Desember 2020. [Online]. Available: <https://medium.com/the-legend/spring-boot-31097079b349.> [Accessed 24 Maret 2023].
- [7] H. S. U. R. S. M. Fuadi Aziz Muri, "Search Engine Get Application Programming Interface," vol. 5, no. 2, p. 89, 2019.

C. KESIMPULAN

