PENGEMBANGAN REST API SIABANG (SISTEM ADMINISTRASI PEMBANGUNAN) MENGGUNAKAN JAVA

(*SiAbang REST API (Development Administrative System) Using Java Development*

Dafa Rozzi Pratama[1], Dr. Eng. Budi Irmawati. S.kom., MT.[1], Rendy Robbani, A.Md[2]
[1]Dept Informatics Engineering, Mataram University
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
[2]Division of Administration and Development of the City of Mataram
Jl. Pejanggik No.16, Mataram, NTB, Indonesia
Email: dafarozzipratama@gmail.com, budi-i@unram.ac.id, mail@rendyrobbani.com

Abstrak

Bagian Administrasi dan Pembangunan Kota Mataram adalah salah satu unit organisasi di Pemerintah Kota Mataram yang bertanggung jawab untuk melaksanakan tugas-tugas terkait dengan pengelolaan administrasi dan pembangunan Kota Mataram. Salah satu tantangan yang dihadapi oleh Bagian Administrasi dan Pembangunan Kota Mataram adalah menyiapkan laporan TPK setiap triwulan sekali. Laporan TPK tersebut masih disusun dengan menggunakan aplikasi Excel karena sistem SiAbang yang pernah dibuat sebelumnya belum menyediakan backend web yang diperlukan. Untuk mengatasi permasalahan tersebut, diperlukan microservice dengan konsep REST API yang dapat men-generate data langsung ke sistem informasi pusat Kota Mataram. Perancangan REST API dilakukan dengan menggunakan metode Extreme Programming. Extreme Programming dikenal sebagai metode teknis atau menggambarkan bagaimana tim teknis mengembangkan perangkat lunak secara efektif menggunakan berbagai prinsip dan teknik pengembangan perangkat lunak praktis. Pada tahap pengujian sistem, dilakukan pemeriksaan berbagai kemungkinan kesalahan yang terjadi selama pengoperasian REST API. Jika REST API berfungsi, REST API berikutnya yang akan diuji adalah REST API spasial yang menggunakan metode GET dan menghasilkan respons sebagai data dalam format JSON. Menggunakan REST API sebagai back end pada Sistem Informasi Administrasi Pembangunan, memungkinkan pegawai kantor di bagian Administrasi Pembangunan Kota Mataram untuk mengakses data. Jika terdapat perubahan data, maka API akan mengirimkan data terbaru.

Keywords: *Company Profile, REST API, Microservice, Spring Boot, Java, Extreme Programming*

1. PENDAHULUAN

Bagian Administrasi dan Pembangunan Kota Mataram adalah salah satu unit organisasi di Pemerintah Kota Mataram yang bertanggung jawab untuk melaksanakan tugas-tugas terkait dengan pengelolaan administrasi dan pembangunan Kota Mataram. Bagian Administrasi dan Pembangunan Kota Mataram memiliki peran penting dalam memastikan pelaksanaan kebijakan pemerintah kota berjalan dengan baik dan memastikan pembangunan Kota Mataram dapat berjalan sesuai dengan rencana yang telah ditetapkan. Salah satu tugas utama dari Bagian Administrasi dan Pembangunan Kota Mataram yaitu membuat laporan keuangan dan kinerja untuk mengevaluasi program dan kegiatan yang telah dilakukan.

Salah satu tantangan yang dihadapi oleh Bagian Administrasi dan Pembangunan Kota Mataram adalah menyiapkan laporan TPK (Tim Pelaksana Kegiatan) setiap triwulan sekali. Laporan TPK tersebut masih disusun dengan menggunakan aplikasi *Excel* karena sistem SiAbang yang pernah dibuat sebelumnya belum menyediakan *backend web* yang diperlukan. Sehingga, pekerjaan tersebut menjadi tidak efisien karena setelah laporan tersebut selesai, laporan tersebut harus di-*upload* ke *web* sistem informasi pusat Kota Mataram. Sehingga, staf kantor tersebut akan mengalami kesulitan untuk mengolah laporan tersebut. Untuk mengatasi permasalahan tersebut, diperlukan *microservice* dengan konsep REST API yang dapat men-*generate* data langsung ke sistem informasi pusat Kota Mataram. Dengan begitu, staf kantor di Bagian Administrasi dan Pembangunan Kota Mataram akan lebih mudah dalam mengolah laporan.

Microservices adalah arsitektur perangkat lunak untuk mengembangkan aplikasi yang terdiri dari beberapa layanan kecil yang berjalan dalam prosesnya sendiri dan berkomunikasi melalui mekanisme sederhana, seringkali menggunakan HTTP sebagai media komunikasi API [1]. API atau *Application Programming Interface* adalah integrasi elemen fungsional, protokol, dan alat lain yang digunakan untuk mengembangkan aplikasi [2]. Menurut konsep ini, API dapat diartikan sebagai perantara yang digunakan oleh banyak aplikasi atau *client* dan *server* baik pada *platform* yang sama maupun pada *platform* yang berbeda untuk berkomunikasi.

REST API (Representational State Transfer Application Programming Interface) adalah jenis arsitektur *web* dari implementasi API (Application Programming Interface) yang digunakan untuk menghubungkan dan berkomunikasi antara satu aplikasi dengan aplikasi lainnya melalui internet [3]. Konsep REST API memungkinkan *user* mengakses informasi di sebuah sistem berbasis *web* dengan lebih aman. *User* tidak dapat mengakses *database* secara langsung, namun dapat membuat permintaan melalui internet menggunakan API. Dengan demikian, *user* tetap mendapat informasi *up-to-date* jika ada data yang di-*update*.

Pada pengabdian masyarakat ini, penulis ingin mempermudah para staf kantoran untuk mengakses semua informasi yang ada di Bagian Administrasi Pembangunan Kota Mataram, maka penulis membuat sebuah REST API yang dapat memenuhi kebutuhan para staf kantoran Bagian Administrasi Pembangunan Kota Mataram.

2. TINJAUAN PUSTAKA

2.1. Profil Singkat Bagian Administrasi dan Pembangunan Kota Mataram

Berdasarkan Peraturan Walikota Mataram Nomor 2 Tahun 2020, Kedudukan, Tugas, Fungsi, dan Tata Kerja Kantor Wilayah Kota Mataram Pasal 23 (1), Bagian Administrasi Pembangunan dipimpin oleh seorang kepala bagian yang berada di bawah dan bertanggung jawab langsung kepada Asisten Perekonomian dan Pembangunan. Bagian Administrasi Pembangunan Kota Mataram bertempat di Kantor Walikota Kota Mataram di Jl. Pejanggik No.16, Mataram Barat, Kecamatan Selaparang, Kota Mataram, Nusa Tenggara Barat.

Berdasarkan Peraturan Walikota Mataram No.2 Tahun 2020 tentang kedudukan, tugas dan fungsi serta Tata Kerja Sekretariat Daerah Kota Mataram pasal 23 ayat 2 dan 3, Bagian Administrasi Pembangunan bertugas melaksanakan penyiapan pengoordinasian perumusan kebijakan daerah, pengoordinasian pelaksanaan tugas Perangkat Daerah, pemantauan dan evaluasi pelaksanaan kebijakan daerah di bidang penyusunan program, pengendalian program dan evaluasi dan pelaporan. Untuk melaksanakan tugasnya, Bagian Administrasi Pembangunan Menyelenggarakan fungsi:

1. Pembuatan program di bidang administrasi pembangunan.
2. Koordinasi persiapan kegiatan di bidang administrasi pembangunan.
3. Penyiapan koordinasi perumusan kebijakan daerah di bidang penyiapan program, pengelolaan program, evaluasi dan pelaporan.
4. Penyiapan koordinasi penyusunan pedoman pelaksanaan tugas organisasi perangkat daerah di bidang penyusunan program, pengelolaan program, evaluasi dan pelaporan.
5. Penyiapan monitoring dan evaluasi pelaksanaan kebijakan daerah dalam hal pencapaian tujuan kebijakan, dampak buruk dan faktor-faktor yang mempengaruhi pencapaian tujuan kebijakan di bidang penyusunan program, pengelolaan program, evaluasi dan pelaporan.
6. Melaksanakan tugas lain yang berkaitan dengan tugas Asisten Pembangunan Ekonomi.

2.2. *Rest API*

REST API (Representational State Transfer Application Programming Interface) adalah jenis arsitektur *web* dari implementasi API (Application Programming Interface) yang digunakan untuk menghubungkan dan berkomunikasi antara satu aplikasi dengan aplikasi lainnya melalui internet [3]. REST API terdiri dari beberapa komponen yaitu [3]:

- URL *Design*, REST API diakses melalui protokol HTTP, sehingga diperlukan penamaan dan struktur URL yang jelas dan mudah dipahami. URL API sering disebut sebagai *endpoint* saat dipanggil. Contoh panggilan URL yang baik adalah seperti berikut: *users, users/9016, users/9016/biodata* dan seterusnya.
- HTTP *verbs*, Adalah metode yang digunakan saat melakukan *request* sehingga *server* dapat mengetahui apa yang diinginkan oleh *client*. Beberapa HTTP *verbs* yang sering digunakan di REST API antara lain: GET, POST, PUT, DELETE.
- HTTP *response code*, Adalah standar dalam memberikan informasi hasil permintaan kepada *client*. terdapat 3 kelompok kode yang umum digunakan di REST API yaitu: 2XX yang menunjukkan *request* yang berhasil dilakukan, 4XX yang menunjukkan bahwa terdapat kesalahan *request* pada sisi *client*, 5XX yang menunjukkan terdapat kesalahan *request* pada sisi *server*.
- *Format response*, Setiap *request* yang dilakukan oleh *client* akan mendapatkan data *response* dari *server* dalam format XML atau JSON. Setelah mendapatkan data *response* tersebut, *client* dapat menggunakan data tersebut dengan cara *parsing* (mengurai) data sesuai dengan kebutuhan.

2.3. *Application Programming Interface* (API)

Application Programming Interface atau yang disingkat API adalah integrasi dari dua bagian dari sistem aplikasi. Terdiri dari elemen fungsional, protokol, dan alat lain yang digunakan pengembang untuk membangun aplikasi. Ada beberapa web API yang mendukung operasi CRUD (Create, Read, Update, Delete) yang dilakukan melalui protokol HTTP menggunakan metode GET, POST, PUT dan DELETE [2].

- Memiliki *response* dengan *header* terima dan kode status HTTP.
- Respon dalam format JSON dan XML.
- Berisi metode MVC seperti perutean, pengontrol, hasil tindakan, *filter*, *template*, wadah IOC, dan lainnya.

Web API bekerja pada semua jenis *server* seperti Apache atau *server* web lainnya. Web API mendukung banyak bahasa pemrograman yang digunakan.

2.4. *Spring Framework*

Menurut Toha, Spring adalah *framework opensource* yang dikembangkan oleh Rod Johnshon, dan pada akhir 1996 Sun Microsystems menerbitkan spesifikasi Java Beans 1.00A [4]. Spesifikasi ini menjelaskan aturan pengkodean Java yang memungkinkan objek menjadi komponen yang dapat digunakan kembali dalam aplikasi Java yang lebih kompleks. Toha juga percaya bahwa Spring adalah kerangka kerja (framework) yang berguna untuk membangun sebuah aplikasi *enterprise*. Spring menyertakan kerangka kerja ringan yang mendukung pengembangan aplikasi bisnis yang dapat digunakan sepenuhnya [4]. Toha menambahkan bahwa Spring Framework memiliki beberapa fitur baru yaitu [4]:

1. *Transaction Management*: Spring Framework menyediakan lapisan abstraksi umum untuk mengelola transaksi, yang memudahkan pengembang untuk mengelola berbagai hal.
2. JDBC *Exception Handling*: Lapisan abstraksi JDBC menyediakan pengecualian hierarkis untuk memfasilitasi penanganan kesalahan.
3. *Integration with Hibernate*, JDO dan iBatis: Spring memberikan layanan integrasi terbaik dengan *Hibernate*, JDO dan lainnya.
4. AOP Framework: Spring *adalah framework* AOP terbaik yang pernah ada.

MVC Spring hadir dengan kerangka kerja aplikasi *web* MVC yang dibangun di atas inti Spring. Spring adalah kerangka kerja yang sangat fleksibel untuk strategi antarmuka dan mencakup beberapa teknologi visual seperti JSP, Velocity, Tiles, iText, dan POI.

2.5. *Model-View-Controller* (MVC)

MVC memisahkan aplikasi menjadi tugas-tugas modul yang saling terkait: *model*, *view*, dan *controller*. Model modul adalah logika bisnis aplikasi dan inti dari aplikasi. Tampilan adalah antarmuka *user* pengontrol. Ini adalah wajah umum saat merespons *user*. Pengontrol mengimplementasikan aliran kontrol antara tampilan dan model [5]. Model-View-Controller (MVC) adalah sebuah konsep yang diperkenalkan oleh pendiri Smalltalk (Trygve Reenskaug) untuk mengenkapsulasi data dengan pemrosesan (model), memisahkan data dari proses manipulasi (pengontrol), dan tampilan disajikan dalam antarmuka *user* (tampilan). Definisi teknis arsitektur MVC dibagi menjadi tiga lapisan [5]:

a. *Model*

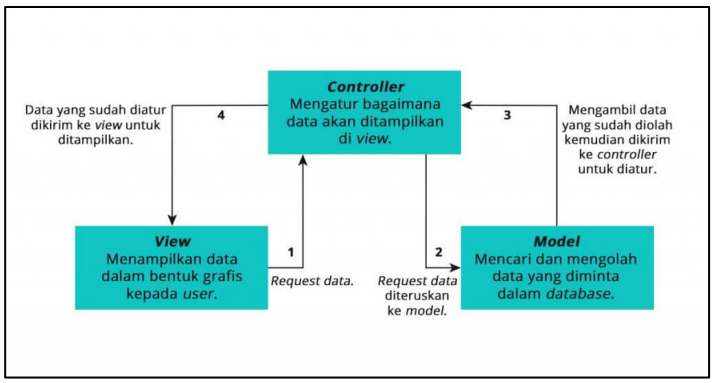
Digunakan untuk mengelola data dan memberi tahu pengamat tentang perubahan data. Hanya *template* yang berisi informasi dan fungsi pemrosesan data. Sebuah model berisi lebih dari sekedar informasi dan fungsi yang beroperasi di dalamnya. Alat pemodelan digunakan untuk memodelkan komputer atau proses dunia nyata abstrak. Ini menggambarkan tidak hanya keadaan proses atau sistem, tetapi juga pengoperasian sistem. Misalnya, pengembang dapat menentukan model yang menggabungkan pemrosesan latar belakang dengan antarmuka *user* grafis.

b. *View*

Bertanggung jawab untuk memasang grafik ke perangkat. Monitor biasanya memiliki rasio aspek 1:1 dan mengetahui cara melakukannya. Tampilan dilampirkan ke model dan kontennya terlihat di tingkat permukaan. Selain itu, saat model berubah, tampilan secara otomatis menggambar ulang bagian layar yang terpengaruh oleh perubahan untuk mencerminkan perubahan tersebut. Model yang sama dapat memiliki beberapa tampilan, dan setiap tampilan tersebut dapat merender konten model dalam area pandang yang berbeda.

c. *Controller*

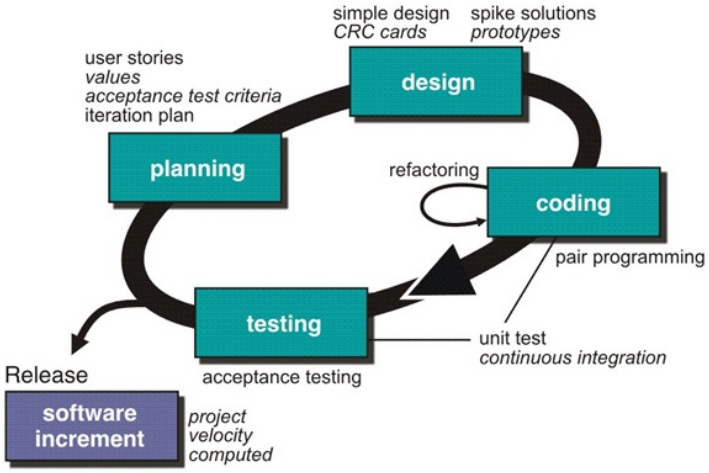
Menerima masukan dari *user* dan menginstruksikan model dan tampilan untuk bertindak atas masukan tersebut. Dengan demikian, pengontrol bertanggung jawab untuk memetakan tindakan *user* akhir ke respons aplikasi. Misalnya, saat *user* mengklik tombol atau memilih *item menu*, pengontrol bertanggung jawab atas respons aplikasi.



Gambar 1. Alur MVC

3. METODE PENGABDIAN MASYARAKAT

Pada pengabdian masyarakat ini, dilakukan proses perancangan REST API dengan menggunakan metode *Extreme Programming* (XP). *Extreme Programming* (XP) merupakan rekayasa perangkat lunak yang terdapat sebuah proses yang cenderung menggunakan pendekatan berorientasi objek dan bertujuan untuk tim yang terdiri dari skala kecil sampai menengah. Metode ini cocok untuk situasi di mana persyaratan proyek tidak jelas atau mengalami perubahan yang cepat [6].



Gambar 2. Tahapan *Extreme Programming*

Berikut adalah tahapan perancangan REST API SiAbang menggunakan metode *Extreme Programming* [6]:

3.1. Tahap *Planing*

Tahapan ini dimulai dengan mendengarkan kumpulan kebutuhan aktifitas dari sistem tertentu, sehingga memungkinkan *user* dapat memahami proses bisnis dalam sistem dan memperoleh pemahaman yang jelas mengenai fitur utama, fungsionalitas dan hasil yang diinginkan. Pada tahap ini dilakukan pendataan terhadap kebutuhan sistem SiAbang.

3.2. Tahap *Design*

Tahap *design* atau perancangan ini melibatkan pembuatan pemodelan sistem berdasarkan hasil analisis kebutuhan yang diperoleh. Selain itu dibuatkan juga pemodelan basis data untuk menggambarkan hubungan antar data. Pemodelan sistem yang digunakan yaitu *Use-Case Diagram* dan *Entity Relational Diagram*.

3.3. Tahap *Coding*

Tahapan ini merupakan implementasi dari perancangan model sistem yang telah dibuat kedalam kode program yang menghasilkan prototipe dari perangkat lunak. Dalam pengembangan REST API SiAbang ini menggunakan *Spring Framework* yang berasal dari bahasa pemrograman Java. Untuk proses *coding* tersebut menggunakan metode MVC (Model, View, Controller) yang berfungsi untuk memisahkan logika aplikasi, data dan *interface user*.

3.4. Tahap *Testing*

Tahapan ini adalah tahapan pengujian aplikasi yang telah selesai dibangun. Dalam sistem ini, *output* aplikasi yang dihasilkan dalam format JSON, sehingga dibutuhkan aplikasi Postman untuk menampilkan *output* dari sistem tersebut. Postman merupakan sebuah aplikasi yang secara khusus dirancang untuk melakukan pengujian dan *debugging* pada API (Application Programming Interface). Sehingga aplikasi Postman tersebut sangat mudah digunakan *user* untuk menguji keluaran sistem tersebut.

4. HASIL DAN PEMBAHASAN

4.1. *Planning*

Pada tahap ini dijelaskan kebutuhan sistem dari sistem SiAbang yang akan dibangun. Penulis merangkum semua kebutuhan sistem informasi, yaitu:

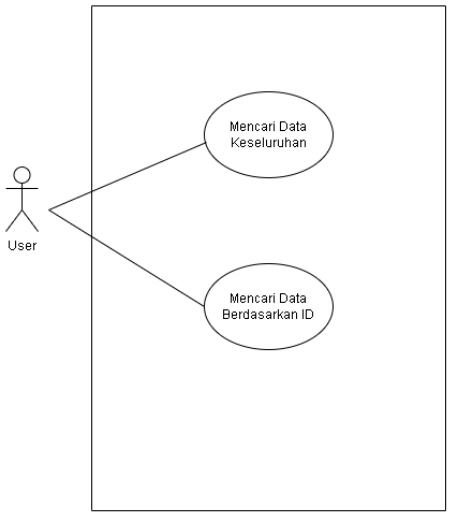
1. *User* dapat melihat seluruh data TPK (Tenaga Penunjang Kegiatan).
2. *User* dapat melihat data TPK (Tenaga Penunjang Kegiatan) berdasarkan ID.

4.2. *Design*

Pada tahap ini penulis membangun sistem sesuai dengan kebutuhan yang diminta dan dijelaskan sebelumnya. Penulis menggunakan alat bantu sistem seperti *use-case diagram* dan *entity relationship diagram*. Pada basis data, dibangun menggunakan MariaDB, di mana kerangka kerja Spring Boot secara otomatis mengisi basis data yang dihasilkan untuk digunakan nanti dalam pengkodean.

4.2.1. *Usecase* diagram

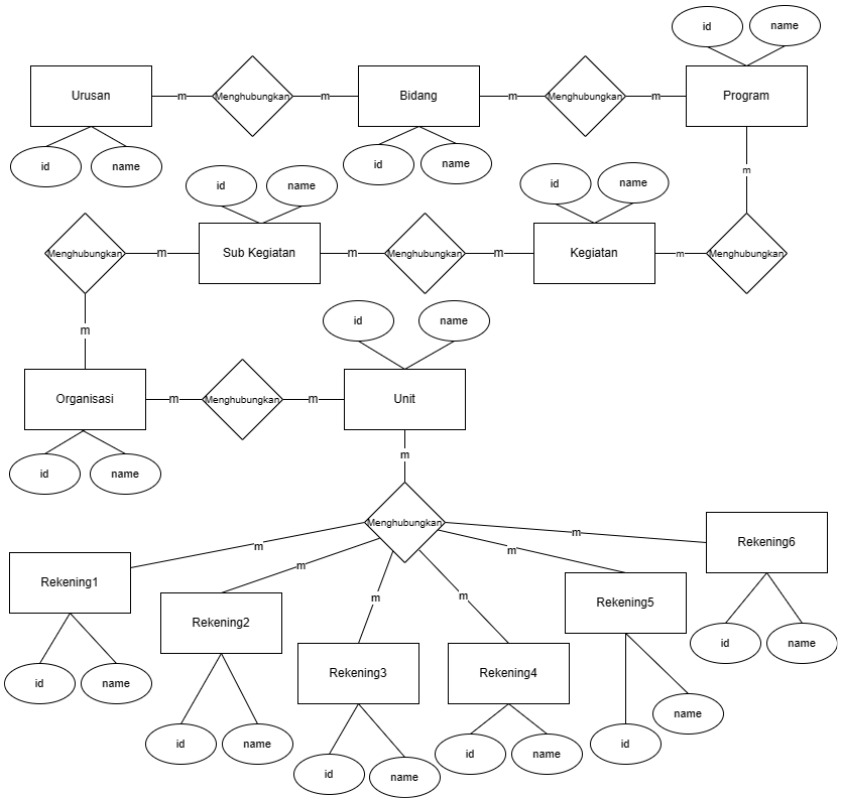
Pada Gambar 2, *use case diagram* ini melibatkan satu aktor saja yaitu *User*. *User* pada sistem ini dapat melakukan pencarian data keseluruhan dan pencarian data berdasarkan nomor ID.



Gambar 3. *Use-Case Diagram* SiAbang

4.2.2. *Entity Relationship Diagram* (ERD)

Pada Gambar 3 ERD ini menghasilkan 13 entitas. Entitas urusan terhubung dengan entitas bidang kemudian entitas bidang terhubung ke entitas program seterusnya hingga terhubung ke entitas rekening 6. Pada entitas ini dilakukan pemisahan entitas yang berfungsi untuk mempercepat proses pembacaan data Ketika mengakses *database*.



Gambar 4. *Entity Relational Diagram* SiAbang

4.3. *Coding*

Pada tahap *coding*, merupakan tahap untuk pengimplementasian berdasarkan tahapan *design* sebelumnya. Pada tahap ini, penulis menggunakan metode MVC (Model View Controller). Dalam pembuatan REST API menggunakan *Springboot* Java, dilakukan beberapa tahapan pembuatan *class* yaitu Model, Repository, DTO (Data Transfer Object), Service, Controller.

4.3.1. Model (Entity)

```
@Entity
@Table(name = "urusan")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

public class Urusan {

    @Id
    @Column(nullable = false, length = 1)
    private String id;

    @Column(nullable = false, length = 255)
    private String name;

}
```

Gambar 5. *Code Model* (Entity) Urusan

Pada Gambar 5 di atas merupakan bagian *model* atau *entity* yang mengelola dan berhubungan langsung dengan *database*. Dalam pembuatan model tersebut, penulis menggunakan *library* Lombok. *Library* ini terdiri dari:

a. *@Setter* yang berfungsi untuk *generate setter method* untuk setiap atribut model *class*.
b. *@Getter* yang berfungsi untuk *generate getter method* untuk setiap atribut model *class*.
c. *@NoArgsConstructor* digunakan untuk *generate constructor* kosong.
d. *@AllArgsConstructor* digunakan untuk *generate constructor* yang mempunyai parameter sama dengan semua atribut.
e. *@Entity* digunakan untuk menandai bahwa sebuah kelas Java mewakili sebuah entitas atau objek yang akan disimpan dalam *database.*
f. *@Table* digunakan untuk membuat tabel baru yang akan di-*generate* ke *database*.

Setelah semua Library dimasukkan, kemudian membuat variable sementara sesuai dengan yang ada di *database*. Pada *code* di atas, terdapat variable id dengan *range* 1 dan variabel *name* dengan *range* 255.

4.3.2. Repository

```
public interface UrusanRepository extends JpaRepository<Urusan, String> {

    Optional<Urusan> findByName(String name);
    Boolean existsByName(String name);

}
```

Gambar 6. *Code UrusanRepository*

Pada Gambar 6 di atas, merupakan *Interface Repository* yang berfungsi untuk meng-*extends JpaRepository*. *JpaRepository* adalah repositori untuk *Java Peristance* API yang memudahkan pengembang untuk melakukan CRUD pada data *string*. Dengan *JpaRepository*, tidak perlu membuat *method insert()* atau *save()* berkali-kali dalam repositori. Pada *code* di atas, terdapat variabel *Optional<Urusan>* yang digunakan jika *name* bernilai *null* karena *name* yang di masukan tidak ada dalam *database*, setelah itu, *code findByName* digunakan untuk mencari data berdasarkan nama yang ingin di cari. Kemudian terdapat *Boolean existsByName* yang digunakan untuk mengecek apakah terdapat nama yang sama.

4.3.3. DTO (Data Transfer Object)

```
@Getter
@Setter
@NoArgsConstructor
public class UrusanRequestDto {
    private String name;

}
```

Gambar 7. *Code UrusanRequestDto*

```
@Getter
@Setter
@NoArgsConstructor
public class UrusanResponseDto {
    private String id;

    private String name;
}
```

Gambar 8. *Code UrusanResponseDto*

DTO atau kepanjangan dari *Data Transfer Object* merupakan sebuah *class* untuk menampung hasil *request* dari sebuah *service*. Misalkan jika pada sebuah *service* terdapat entitas Urusan, di dalam DTO juga terdapat Urusan. Pada DTO tidak perlu membuat anotasi @Entity karena sifatnya hanya penampung. Pada Gambar 7 dan Gambar 8 terdapat dua jenis DTO yaitu *request* dan *response* yang berisi dua variabel penampung sementara yaitu *id* dan *name*. Perbedaannya dimana *UrusanResponseDto* tidak menampilkan *id* entitas hanya menampilkan nama entitas dan hasil *request* tersebut dikirim ke *user*, sedangkan *UrusanResponseDto* menampilkan *id* dan *name* entitas dan hasil *request* tersebut dikirim ke *server*.

4.3.4. Service

```java
@Service
@Transactional
public class UrusanServiceImpl implements UrusanService{

    @Autowired
    private UrusanRepository repository;

    @Override
    public List<UrusanResponseDto> findAll() {
        List<UrusanResponseDto> response = new ArrayList<>();
        List<Urusan> entities = repository.findAll();
        for (Urusan dataurusan: entities) response.add(toDto(dataurusan));
        return response;
    }

    @Override
    public Optional<UrusanResponseDto> findById(String id) {
        Urusan urusan = repository.findById(id).orElse(other: null);
        if (urusan != null) return Optional.of(toDto(urusan));
        return Optional.empty();
    }

    private UrusanResponseDto toDto(Urusan dataurusan) {
        UrusanResponseDto response = new UrusanResponseDto();
        response.setId(dataurusan.getId());
        response.setName(dataurusan.getName());
        return response;
    }
}
```

Gambar 9. *Code UrusanServiceImpl*

Service merupakan sebuah *class* yang berfungsi untuk melakukan akses *service* atau API, *service* ini dipanggil oleh sebuah *controller*. *Service Layer* adalah *layer* dimana kita menyimpan *logic* dari sebuah aplikasi seperti memanipulasi data, mem-*filter* dan lain-lain. Pada Gambar 9 tersebut, terdapat 3 *logic* yang akan digunakan yaitu *method findAll()* yang berfungsi melakukan pencarian dan pengambilan seluruh data yang tersedia dalam tabel Urusan pada sebuah *database*, *method findById()* yang berfungsi melakukan pencarian dan pengambilan data berdasarkan id yang tersedia dalam tabel Urusan pada sebuah *database* dan *method UrusanResponseDto toDto(Urusan dataurusan)* yang berfungsi untuk mentransfer data dari *service* ke DTO atau *Data Transfer Object*. *Method* tersebut berasal dari *library* dari *UrusanRepository* yaitu *JPARepository*. *Method findAll()* tersebut menggunakan *Array List* yang berfungsi untuk memudahkan penyimpanan dan pengambilan data. Setelah itu, data yang disimpan di *Array List* tersebut dikirim ke DTO menggunakan *method UrusanResponseDto()*. *Method findById()* tersebut akan membuat objek baru yang diambil dari *library UrusanRepository*, setelah itu dibuatkan kondisi bila data tersebut tidak sama dengan *null*, maka *id* tersebut akan dikirim ke DTO. Bila data tersebut *null*, maka akan membuat objek kosong untuk menampung hasil pencarian berdasarkan *id* yang ingin di cari. Pada *method UrusanResponseDto()*tersebut, akan membuat objek baru Bernama *response* yang berfungsi untuk mentransfer data ke DTO. Data yang ditransfer yaitu *id* dan *name*.

4.3.5. Controller

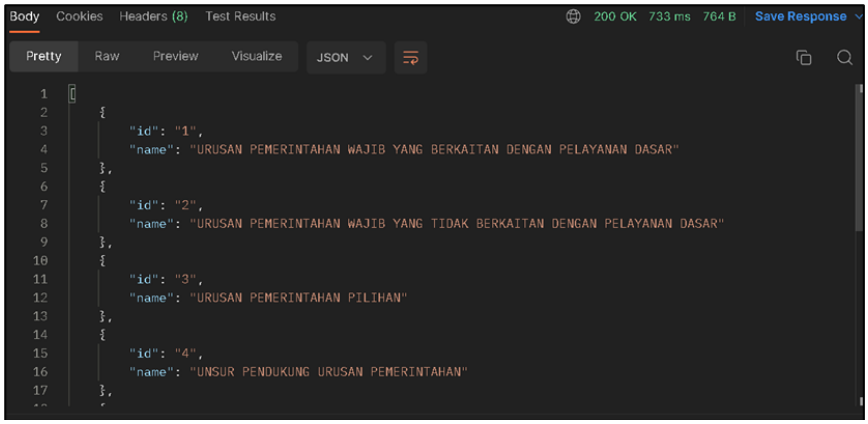
```java
@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("api/ref/urusan")
public class UrusanController {

    @Autowired
    private UrusanService service;

    @GetMapping
    public ResponseEntity<List<UrusanResponseDto>> findAll(){
        return ResponseEntity.status(200).body(service.findAll());
    }

    @GetMapping("{id}")
    public ResponseEntity<UrusanResponseDto> findById(@PathVariable String id) {
        return ResponseEntity.of(service.findById(id));
    }
}
```

Gambar 10. *Code UrusanController*

Controller merupakan pintu masuk dari sebuah *request*, berfungsi untuk memproses *request* ataupun mengembalikan *request*. Ketika memproses *request* untuk meminta sebuah data, controller akan berhubungan pada sebuah *class service*. Pada Gambar 10 tersebut, anotasi *@CrossOrigin* berfungsi untuk mengizinkan permintaan lintas *domain* ke *endpoint localhost:3000*. Anotasi *@RestController* berfungsi untuk membuat kelas yang bertindak sebagai RESTful *web service*. Dengan menggunakan anotasi *@RestController* ini pada kelas, maka dapat mengembalikan data dalam bentuk format tertentu seperti JSON atau XML sebagai *response* HTTP dari *endpoint* yang dibuat. Anotasi *@RequestMapping("api/ref/urusan")* digunakan untuk meminta *request* GET data ke *endpoint api/ref/urusan*. Pada anotasi *@GetMapping* tersebut, berfungsi merespons permintaan GET dari *client* dengan memetakan permintaan ke URL tertentu.Pada anotasi tersebut, akan mengembalikan *response* HTTP yang valid dalam bentuk objek *ResponseEntity*. *ResponseEntity.status(200)* digunakan untuk membuat objek *ResponseEntity* dengan kode status HTTP 200 (OK) yang menandakan apakah permintaan tersebut berhasil atau gagal Kode status HTTP 200 (OK) menunjukkan bahwa permintaan telah berhasil dan *response* yang diberikan oleh *server* adalah valid. Pada anotasi *@GetMapping("{id}")* tersebut, akan memanggil *method findById()* yang berasal dari *class ServiceUrusan*. Dengan begitu, Data akan melakukan GET *mapping* sesuai dengan *id* yang ingin di cari.

4.4. *Testing*

Setelah tahap pengkodean, tahap pengujian sistem dilakukan, di mana berbagai kesalahan yang terjadi selama pengoperasian REST API diselidiki. Jika REST API berfungsi, REST API berikutnya yang akan diuji adalah REST API spasial yang menggunakan metode GET dan menghasilkan respons sebagai data dalam format JSON seperti yang ditunjukkan di bawah ini:



Gambar 11. Hasil *Run* REST API Menggunakan Postman

5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengabdian masyarakat yang dilakukan di Bagian Administrasi Pembangunan Kota Mataram dapat disimpulkan:

1. Sistem Informasi Administrasi Pembangunan (SiAbang) Merupakan sistem informasi Administrasi Pembangunan berbasis *website* yang digunakan untuk melihat dan mengolah informasi terbaru mengenai data administrasi sebagai tugas perangkat daerah.
2. Menggunakan REST API sebagai *back end* pada Sistem Informasi Administrasi Pembangunan, memungkinkan pegawai kantor di bagian Administrasi Pembangunan Kota Mataram untuk mengakses data. Jika terdapat perubahan data, maka API akan mengirimkan data terbaru.

UCAPAN TERIMA KASIH

Terima kasih kepada Bagian Administrasi dan Pembangunan Kota Mataram yang telah memberikan kesempatan untuk melakukan kegiatan pengabdian lapangan. Karyawan dan kolega adalah pihak yang menyediakan data dan informasi sebagai dasar untuk membangun sistem informasi SiAbang tersebut. Serta kepada teman-teman dan keluarga yang telah membantu penulis dalam melaksanakan kegiatan pengabdian masyarakat ini.

DAFTAR PUSTAKA

[1] M. Rezaldy, I. Asror, and I. L. Sardi, "Desain dan Analisis Arsitektur Microservices Pada Sistem Informasi Akademik Perguruan Tinggi Dengan Pendekatan Architecture Tradeoff Analysis Method (ATAM) (Studi Kasus: iGracias Universitas Telkom) Design and Analysis of Microservices Architecture On Academic Information System With Higher Education Approach Architecture Tradeoff Analysis Method (ATAM) (Case Study: iGracias Telkom University)."

[2] S. N. Yanti and E. Rihyanti, "Penerapan Rest API untuk Sistem Informasi Film Secara Daring," *Jurnal Informatika Universitas Pamulang*, vol. 6, no. 1, p. 195, Mar. 2021, doi: 10.32493/informatika.v6i1.10033.

[3] M. Angga and K. Perdana, "Pengembangan REST API Layanan Penyimpanan Menggunakan Metode Rapid Application Development (Studi Kasus: PT. XYZ)."

[4] O. Sukardi, J. Nurma Sari, and D. A. Wibowo, "Rancang Bangun Sistem Informasi Pengembangan Sumber Daya Manusia menggunakan Teknologi Java pada Level Control (Studi Kasus PT. Chevron Pacific Indonesia)," 2015. [Online]. Available: http://jurnal.pcr.ac.id

[5] P. Simanjuntak and A. Kasnady, "Analisis Model View Controller (MVC) pada Bahasa PHP," 2016.

[6] A. Supriyatna, "Metode Extreme Programming pada Pembangunan Web Aplikasi Seleksi Peserta Pelatihan Kerja," *Jurnal Teknik Informatika*, vol. 11, no. 1, pp. 1–18, May 2018, doi: 10.15408/jti.v11i1.6628.