Jurnal Strategi
Volume 2 Nomor 2 November 2020

Rancang Bangun Web API untuk Akses Informasi Kerja Sama Pendidikan Tinggi

Joses Vito Chandra[1], Robby Tan[2]

[#]*Program Studi Teknik Informatika Universitas Kristen Maranatha*
Jl. Prof. drg. Surya Sumantri No. 65 Bandung
[1]it1672023@student.it.maranatha.edu
[2]robby.tan@it.maranatha.edu

Abstract — **Kerja sama merupakan salah satu bagian yang menjadi tulang punggung sebuah organisasi/ instansi. Direktorat Kerja Sama Universitas Kristen Maranatha adalah sebuah instansi yang bertugas untuk menangani perihal kerja sama universitas dengan pihak eksternal. Proses arsip, pemantauan dan pembaharuan dokumen dalam sebuah kerja sama menjadi poin utama yang perlu diperhatikan. Direktorat Kerja Sama membutuhkan sistem yang diharapkan dapat mempermudah pengarsipan dokumen kerja sama legal, dan mempercepat pencarian data kerja sama. Sistem yang dibangun diharapkan dapat memiliki proses maintenance dan update dengan baik. Solusi yang ditawarkan adalah sebuah dashboard dengan memisahkan antara frontend dan backend. Backend dibuat sebagai sebuah Application Programming Interface (API) yang dikembangkan dengan Node.js dan basis data MySQL. Pengerjaan frontend dan backend dilakukan terpisah untuk memudahkan pemeliharaan sistem. Dalam penelitian ini, fitur berupa pencarian data dan pengarsipan dokumen dirancang sedemikian rupa untuk memenuhi kebutuhan Direktorat akan sistem pengarsipan yang lebih tertata. Berdasarkan fitur dan pengujian yang telah dilaksanakan, rancang bangun API ini dapat membantu proses bisnis yang berlangsung dalam Direktorat Kerja Sama Universitas Kristen Maranatha.**

Keywords— **API, Backend, Kerja Sama, Node.js**

I. PENDAHULUAN

Organisasi yang memiliki banyak data akan sangat membutuhkan ‘satu tempat untuk mengakses dan mengubah data’ untuk kemudahan memperoleh informasi. Sebagai contoh, Direktorat Kerja Sama Universitas Kristen Maranatha (DKS) bertugas untuk menangani kerja sama antara Universitas Kristen Maranatha dan mitranya, maka dari itu Direktorat Kerja Sama harus mencatat data-data berupa informasi kerja sama dengan mitra Maranatha. Saat penelitian ini dibuat, DKS memiliki keterbatasan mengakses data Perjanjian Kerja Sama (PKS) dengan mitra. Keterbatasan tersebut disebabkan karena data masih berupa *file* Excel yang digunakan bersama-sama secara manual yang tidak tersambung dengan dokumen-dokumen kerja sama yang dibuat. Hal tersebut menyebabkan kendala karena tidak dapat mendapatkan informasi tersebut secara *real-time*.

Berdasarkan latar belakang tersebut, penelitian ini dilakukan untuk mendapatkan solusi atas permasalahan yang dimiliki DKS. Salah satu solusi yang dapat diterapkan yaitu penerapan basis data dan API. Basis data disusun secara sistematik dan terpadu sehingga dapat mewakili data pada DKS. Akses terhadap data tersebut dibuat dalam bentuk API untuk mendukung transaksi data sehingga dapat menampilkan data dan mengubah data.

Penelitian ini merupakan sub proyek dari pengembangan web Direktorat Kerja Sama UKM, yang akan dibahas pada penelitian ini adalah pengembangan *backend* dan tidak membahas konversi data dari berkas Excel ke dalam basis data. Bagian *frontend* dikerjakan oleh Raden Kasyfi Aghitya dari Fakultas IT Universitas Kristen Maranatha.

II. KAJIAN TEORI

A. *Express*

Express adalah sebuah *framework* untuk Node.js. *Framework* ini menggunakan pola Model-View-Controller (MVC) yaitu sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (*model*) dari tampilan (*view*) dan logika dalam proses bisnisnya (*controller*). [1]

```
1.  var express = require('express')
2.  var app = express()
3.  var routes = require('./routes')
4.  routes(app);
```

Kode Program **1**. Contoh Kode Program Node.js dengan Framework Express

B. Body-Parser

Sebuah *middleware* yang dirancang untuk Node.js. Body-parser bekerja untuk mengurai/ menerjemahkan *request* sebelum masuk ke *controller* pada API. *Request* yang telah diuraikan akan tersimpan sebagai properti req.body yang kemudian dapat diakses oleh *controller*. [2]

```
var bodyParser = require('body-parser')
app.use(bodyParser.urlencoded({
  extended: true
}));
app.use(bodyParser.json());
```

Kode Program 2. Contoh Kode Program Node.js dengan Express dan body-parser

C. Cross-Origin Resource Sharing (CORS)

CORS adalah sebuah mekanisme yang mengizinkan data pada sebuah domain untuk diakses oleh domain lain. Penggunaan CORS akan mengizinkan setiap rute pada API dapat diakses di luar jaringan lokal. [3]

```
var cors = require('cors'),
app.use(cors())
```

Kode Program 3. Contoh Kode Program Node.js dengan CORS

D. MySQL2

MySQL2, yang merupakan lanjutan dari MySQL-Native, adalah sebuah klien MySQL untuk Node.js dengan fokus pada performa. Penggunaan MySQL2 memungkinkan sebuah API untuk tersambung dengan server MySQL. [4]

```
var mysql = require('mysql2');

var con = mysql.createConnection({
  host: "localhost",
  user: "username",
  password: "password",
  database: "db_dev"
});
```

Kode Program 4. Contoh Kode Program Node.js untuk Koneksi dengan MySQL

E. Method-Override

Penggunaan package "method-override" memungkinkan klien API yang tidak mendukung metode HTTP berupa PUT dan DELETE untuk dapat menggunakannya. Metode HTTP yang ingin diminta dapat dikirimkan melalui req.body._method. [5]

```
var methodOverride = require('method-override');
app.use(methodOverride(req => req.body._method))
```

Kode Program 5. Contoh Kode Program Node.js dengan method-override untuk Routes

F. Multer

Multer adalah *middleware* Node.js untuk menangani multipart/ form-data yang biasa digunakan untuk mengunggah berkas. Dengan menggunakan Multer, API dapat menerima berkas dengan syarat tertentu dan kemudian diproses lebih lanjut. [6]

```
var multer = require('multer')
var upload = multer()
upload.single('file')(req, res, function(err){
  if(err){
    // terjadi kesalahan mengunggah berkas
    return response.fail(err, res)
  }
  // pengunggahan berkas berhasil
})
```

Kode Program 6. Contoh Kode Program Node.js dengan multer

G. JSON Web Token (JWT)

JWT adalah suatu bentuk yang aman dan ringkas untuk mewakili data untuk ditransfer antara dua pihak. Penggunaan JWT memungkinkan *request* yang aman dengan API, mengurangi risiko terjadinya hal yang tidak diinginkan ketika API aktif. [7]

```
var jwt = require('jsonwebtoken')
```

```
2.
3.  exports.jwtSign = function (data) {
4.    var payload = {
5.      payload: data
6.    }
7.    let privateKey = fs.readFileSync('./private.pem', 'utf8');
8.    let token = jwt.sign(payload, privateKey, {
9.      algorithm: '', // diisi dengan algoritma enkripsi jwt
10.     expiresIn: '3h'
11.   });
12.   return token;
13. }
14.
15. exports.jwtVerify = function (token) {
16.   let publicKey = fs.readFileSync('./public.pem', 'utf8');
17.   try {
18.     return jwt.verify(token, publicKey, {
19.       algorithm: 'RS256' // diisi dengan algoritma enkripsi jwt
20.     });
21.   } catch (e) {
22.     return ({
23.       name: 'JsonWebTokenError',
24.       message: 'invalid signature'
25.     })
26.   }
27. }
```

Kode Program 7. Contoh Kode Program JWT untuk Node.js

H. Bcrypt.js

Bcrypt.js menerapkan *salting* untuk melindungi dari *rainbow table attacks*, yaitu tabel yang telah dikomputasi untuk membalikkan fungsi *hash* kriptografi, biasanya untuk memecahkan *hash* kata sandi. Bcrypt.js pada API ini digunakan untuk membandingkan *hash* kata sandi yang dimasukkan pengguna dengan *hash* kata sandi yang ada pada basis data [8].

```
1.  var bcrypt = require('bcryptjs')
2.  bcrypt.compareSync(req.body.password, rows[0].password)
```

Kode Program 8. Contoh Kode Program bcrypt.js untuk Node.js

I. Validator

Validator merupakan sebuah *package* untuk Node.js. *Validator* bekerja sebagai *middleware* yang digunakan saat rute pada API menerima *input*, pengembang tinggal memilih parameter yang perlu diberikan sanitasi dan yang diperlukan untuk memastikan parameter yang diterima dapat digunakan untuk menjalankan *query* ke basis data [9].

```
1.  const { Validator } = require('node-input-validator')
2.  const v = new Validator(req.body, {
3.    token: 'required',
4.    id: 'required',
5.    jenis_dokumen_id: 'required|in:1,2,3',
6.    tglawal: 'required|date',
7.    tglakhir: 'required|after:' + req.body.tgl_awal
8.  })
9.  v.check().then((isMatch) => {
10.   if(!isMatch){
11.     // input tidak valid
12.   }
13.   else {
14.     // input valid
15.   }
16. })
```

Kode Program 9. Contoh Kode Program Node.js dengan Validator pada Routes

J. Backend

Jurnal Christine Dewi dari Fakultas Teknologi Informasi, Program Studi Teknik Informatika Universitas Kristen Satya Wacana Salatiga, yang berjudul "Sistem Pelaporan Infrastruktur Dinas Bina Marga dan PSDA Kota Salatiga Menggunakan Node.js Berbasis Web" [10] memiliki fokus pada sistem web dan basis data sebagai solusi atas masalah dinas Bina Marga dan PSDA kota Salatiga mengenai pengolahan data informasi laporan kerusakan infrastruktur.

Melalui penelitian ini, diketahui pada pembuatan *backend* aplikasi menggunakan teknologi Node.js sebagai web server, Express sebagai kerangka kerja (*framework*) pada Node.js, dan MySQL sebagai basis data. Dalam penelitian ini tidak dibahas tentang pembuatan API. Hasil dari penelitian ini yaitu sebuah aplikasi yang dapat membantu *user* untuk melaporkan dan memantau masalah yang terjadi pada infrastruktur.

K. API

Jurnal Hao Zhong, Peking University, yang berjudul "An Empirical Study on API Usages" merupakan sebuah studi empiris tentang penggunaan API. [11] Dari penelitian tersebut, didapatkan sebuah definisi untuk API yaitu sekumpulan elemen kode yang disediakan oleh *framework* atau *libraries*, dan *framework* atau *libraries* tersebut disebut API *libraries*, sedangkan kode dari API *library* disebut API *code*.

Penelitian ini juga menegaskan bahwa seberapa seringnya *programmer* menggunakan API masih tidak dapat diketahui. Akan tetapi, bersamaan dengan evolusi perangkat lunak API yang tidak populer bisa saja menjadi populer di masa depan.

L. Kerja Sama

Bab Cooperations dalam buku "Changing Cuba-U.S. Relations: Implications for CARICOM States", menguraikan kerja sama antara Kuba dan CARICOM dalam sektor-sektor seperti kesehatan, pendidikan, pengurangan risiko bencana, dan pertukaran budaya. Kuba mengakui kerja sama dengan negara tetangga di bagian Selatan sebagai prinsip utama kebijakan dengan luar negeri dan telah memperkuat inisiatif kerja sama dengan negara-negara CARICOM (*Caribbean Community*) melalui berbagai proyek. Tindakan kerja sama ini telah ditandai dengan berbagai tingkat keberhasilan dan tantangan bagi kedua belah pihak. [12]

III. ANALISIS DAN RANCANGAN SISTEM

Penelitian dilakukan dengan mengadakan wawancara awal terhadap pihak DKS terhadap dokumen-dokumen dan proses yang digunakan selama perjanjian

A. Analisis Proses Bisnis

Calon mitra atau pihak DKS melakukan inisiasi kerja sama melalui pertemuan yang sudah dijanjikan atau melalui kontak digital atau proposal kerja sama. Proposal kerja sama dapat digunakan untuk pembuatan dokumen kerja sama yang memiliki kategori sebagai berikut: SKB (Surat Keputusan Bersama) untuk perjanjian jangka panjang, PKS (Perjanjian Kerja Sama) untuk perjanjian jangka waktu pendek, dan PKSA (Perjanjian Kerja Sama - Aktivitas) untuk aktivitas dari PKS yang sudah diberlakukan. Setiap dokumen SKB/PKS yang sudah legal akan dicatat ke dalam berkas Excel serta memiliki tanggal mulai dan tanggal akhir perjanjian tersebut. Pihak DKS kemudian dapat memberitahu mitra UKM apabila masa berlaku perjanjian tersebut akan segera habis dan menawarkan opsi untuk memperpanjang kerja sama atau tidak.

1) Input Data Mitra: 1) Admin mendapat/melakukan inisiasi dari/ke mitra. 2) Admin memasukkan data mitra. 3) Admin memberi/menerima proposal kerja sama.

2) Pembuatan Dokumen: 1) Admin beserta internal DKS menyusun dokumen sebagai panduan untuk kerja sama dengan mitra. 2) Admin menyampaikan rancangan dokumen kepada mitra untuk konfirmasi. 3) Admin melakukan pengesahan atas dokumen final.

3) Input Data Dokumen: 1) Admin memasukkan data dokumen yang sudah sah (tertandatangani). 2) Admin dapat memperbarui data dokumen yang sudah terdaftar

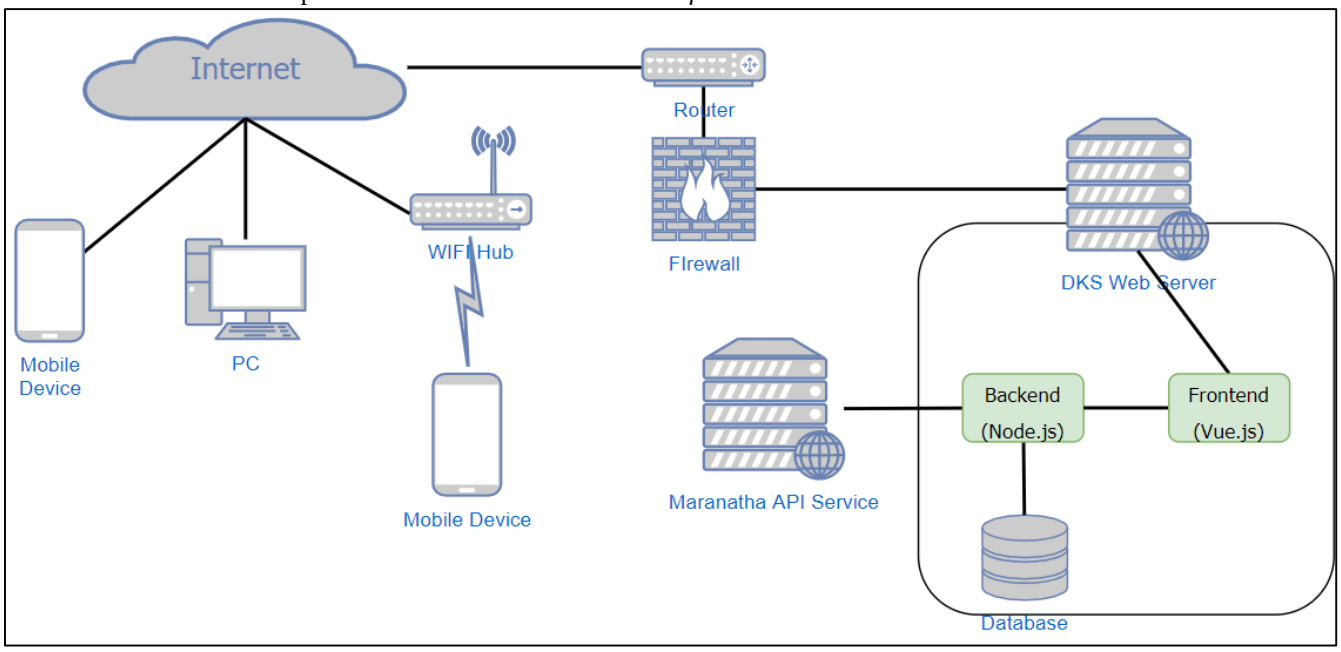
4) Pencarian Data Perjanjian: 1) Admin mencari data perjanjian yang akan segera habis masa berlakunya. 2) Admin memberikan tawaran perpanjangan kerja sama kepada mitra.

5) Akhir Perjanjian: 1) Admin memisahkan dokumen yang masa berlakunya sudah habis (*expired*) dengan yang masih berlaku. 2) Admin mengarsipkan dokumen lama sebagai bukti/berkas perjanjian lalu.

B. Arsitektur dan Lingkup Proyek

Fokus penelitian adalah pengembangan API Service untuk DKS UKM, sehingga pembahasan akan terbatas pada sisi aplikasi backend dan basis data. Pengerjaan aplikasi frontend dapat dilihat pada laporan Raden Kasyfi Aghitya yang berjudul, "Dashboard Direktorat Kerja Sama Maranatha dengan Vue.js". [13]

HTTP *request* berupa GET, POST, PUT, dan DELETE dapat dikirimkan dari *frontend* kepada *backend*. Semua *request* yang dikirimkan pada *backend* kemudian diproses lagi dan dibungkus dalam bentuk JSON untuk diteruskan ke basis data ataupun Maranatha API Service. Pada sebagian besar proses, setelah *request* diteruskan, *backend* kembali mendapat respons dalam bentuk JSON dan respons tersebut akan dikembalikan ke *frontend*.



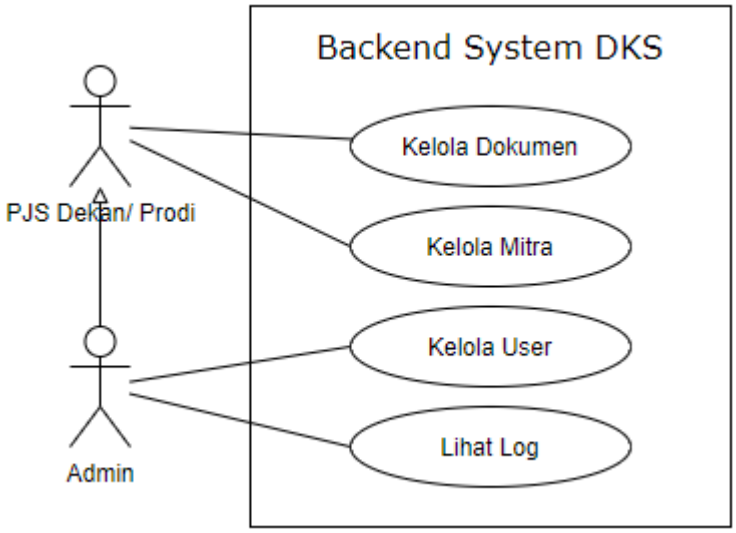
Gambar 1. Posisi Penelitian dalam Proyek

C. Analisis UML

Bagian ini adalah rancangan analisis UML yang akan diterapkan untuk rancang bangun backend Direktorat Kerja Sama. Activity diagram yang ada merupakan perincian atau penjelasan untuk setiap use case.

1) Use Case: Berikut adalah rancangan *use case* yang akan digunakan untuk *RESTful API backend* Direktorat Kerja Sama.

2) Activity Diagram: Dalam rancang bangun API ini, telah disediakan *activity diagram* yang merupakan referensi sehingga API yang dikembangkan memiliki proses, antara lain: Login, Logout, Kelola Dokumen, Kelola Mitra, Kelola Rincian Dokumen Kerja Sama, Perbarui Dokumen SKB, Lihat Log, Lihat User, *Assign User* ke Mitra.

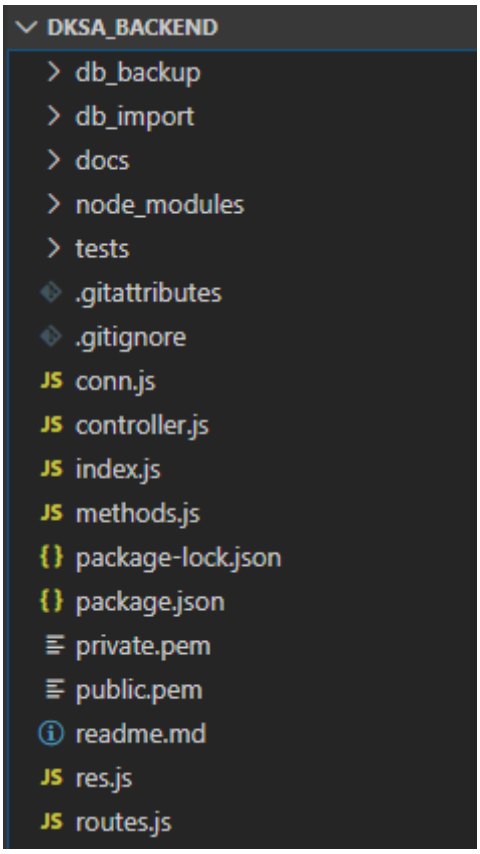


Gambar 2. Use Case DKS Universitas Kristen Maranatha

IV. IMPLEMENTASI

A. Struktur API

Struktur API pada Gambar 3 merupakan direktori utama yang dapat dijalankan menggunakan Node.js, pada direktori ini juga terdapat beberapa berkas dengan ekstensi .js yang merupakan metode-metode yang dibutuhkan agar API dapat bekerja sesuai dengan rancangan.



Gambar 3. Struktur API

1) Berkas index.js: Berkas dengan nama index.js merupakan main dari sebuah program, 'main' ini digunakan untuk mengeksekusi program hingga pengguna atau peladen menutupnya sehingga proses tersebut akan berakhir pada sistem. Index.js ini dieksekusi menggunakan Node.js dari terminal atau cmd.

```
1.  var express = require('express'),
2.    app = express(),
3.    cors = require('cors'),
4.    port = process.env.PORT || 3000,
5.    bodyParser = require('body-parser'),
6.    methodOverride = require('method-override');
7.
8.  app.use(bodyParser.urlencoded({
9.    extended: true
10. }));
11. app.use(bodyParser.json());
12.
13. app.use(cors())
14.
15. app.use(methodOverride(req => req.body._method))
16.
17. var routes = require('./routes');
18. routes(app);
19.
```

```
var cluster = require('cluster');
if (cluster.isMaster) {
  cluster.fork();

  cluster.on('exit', function (worker, code, signal) {
    cluster.fork();
  });
}
if (cluster.isWorker) {
  app.listen(port);
  console.log('RESTful API server started on: ' + port);
}
```

Kode Program 10. Berkas index.js

2) Berkas conn.js: Berkas dengan nama conn.js merupakan sebuah fungsi pemrograman untuk menyambungkan API dengan basis data. Fungsi ini dipanggil melalui index.js yang merupakan main/berkas utama yang dijalankan menggunakan Node.js.

```
var mysql = require('mysql2');

var con = mysql.createConnection({
 host: "localhost",
 user: "root",
 password: "",
 database: "dks_db20192"
});

con.connect(function (err) {
 if (err) throw err;
});

module.exports = con;
```

Kode Program 11. Berkas conn.js

3) Berkas routes.js: Berkas dengan nama routes.js merupakan sebuah fungsi pemrograman untuk dipanggil oleh 'main'. Secara spesifik, routes.js ini menyimpan nama-nama rute yang dapat dipanggil melalui peramban web ataupun program lainnya yang dapat mengirimkan HTTP Request dengan metode GET, POST, PUT, dan DELETE. Setiap rute yang ada akan memanggil fungsi masing-masing yang terdapat pada controller.js.

```
'use strict';

module.exports = function (app) {
  var todoList = require('./controller'),
    path = require('path'),
    multer = require('multer')

  app.route('/')
    .get(todoList.index);

  app.route('/pic')
    .get(todoList.person);

  app.route('/pic')
    .post(todoList.createPerson);
```

Kode Program 12. Berkas routes.js

4) Berkas controller.js: Berkas dengan nama controller.js merupakan sebuah fungsi pemrograman untuk dipanggil oleh routes.js. Controller berlaku sebagai alur yang memastikan setiap rute berfungsi sesuai dengan rancangan yang telah dibuat. Hasil akhir berupa success beserta data kembalian, dan fail yang merupakan kegagalan sistem untuk mengelola data akan dikembalikan melalui controller.

```
var response = require('./res');
```

```
var connection = require('./conn');
var handle = require('./methods')
var axios = require('axios')
var qs = require('querystring')

exports.login = function (req, res) {
// di sini adalah barisan
// kode program untuk fungsi login
}

exports.kategoriMitra = function (req, res) {
  var payload = handle.routeAccess(req.body.token, [4])
  if (payload) {
    connection.query('SELECT * FROM kategori_mitra', function (error, rows, fields) {
      if (error) {
        console.log(error)
      } else {
        response.ok(rows, res)
      }
    });
  } else {
    response.fail("User is not authorized", res)
  }
};
```

Kode Program 13. Berkas controller.js

5) Berkas res.js: Berkas dengan nama res.js merupakan sebuah fungsi pemrograman yang dapat dipanggil dari dalam program itu sendiri yang berbasis Express. Pada res.js terdapat dua metode yaitu success (mengembalikan pesan bahwa proses berhasil beserta data yang diproses atau diminta) dan fail (mengembalikan pesan bahwa proses tidak berhasil dieksekusi oleh sistem karena kesalahan pengguna).

```
'use strict';

exports.ok = function (values, res) {
  var data = {
    'status': 1,
    'message': 'success',
    'values': values
  };
  res.json(data);
  res.end();
};

exports.fail = function (values, res) {
  var data = {
    'status': 0,
    'message': "fail",
    'values': values
  };
  res.json(data);
  res.end();
}
```

Kode Program 14. Berkas res.js

6) Berkas methods.js: Methods.js berlaku sebagai utility class yang menyimpan berbagai fungsi pemrograman yang dapat dijalankan berulang kali atau yang dapat digunakan oleh fungsi lain pada suatu class tertentu. Oleh karena itu, methods.js merupakan contoh reusable code yang dapat membantu pengembang dalam mempersingkat source code — pengembang tidak perlu menyalin barisan perintah yang sama dan bisa saja sangat panjang pada setiap fungsi utama.

```
const jwt = require('jsonwebtoken');
const fs = require('fs');
```

```
3.  var connection = require('./conn');
4.  var axios = require('axios')
5.  var qs = require('querystring')
6.  var response = require('./res')
7.  // di sini adalah barisan kode program untuk verifikasi akses rute
8.  exports.routeAccess = function (token, routeLevel) {
9.    var payload = exports.jwtVerify(token).payload
10.   try {
11.     var found = routeLevel.find(r => r == payload.access_level)
12.     if(found){
13.       return payload
14.     } else {
15.       return undefined
16.     }
17.   } catch (e) {
18.     return undefined
19.   }
20. }
```

Kode Program 15. Berkas methods.js

V. PENGUJIAN

Pengujian hasil panggilan API untuk Direktorat Kerja Sama pertama dilakukan dengan metode black-box testing. Metode ini merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak (tanpa menguji proses yang terjadi dalam aplikasi). Tester dapat mengidentifikasi kesesuaian kondisi input dan output dari pengujian pada fungsi program. Tabel I menunjukkan kesesuaian hasil pengujian dengan yang diharapkan dari cara pengujian yang telah dilakukan terhadap setiap fungsi yang ada pada API.

TABEL I

TABEL HASIL PENGUJIAN BLACK-BOX TESTING

No.	Fungsi yang Diuji	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian			
				Admin	Rektorat	PJS	Guest
1	GET /mitra	Menyimpan Token Pada Request Body dan Memanggil Rute /mitra dengan Metode GET	Data-data Mitra ditampilkan Sesuai dengan Tingkat access_level ID dan Mitra Pada Token	Sesuai			
2	POST /mitra	Memasukkan Data Mitra (Dummy) Bervariasi	Jumlah Mitra Bertambah dan Data yang Baru Sesuai dengan Data pada Request	Sesuai			
3	PUT /mitra	Memasukkan ID Mitra dan Data Barunya (Dummy) Secara Bervariasi	Data Mitra yang ditunjuk Berubah Sesuai dengan Data pada Request	Sesuai			
4	DELETE /mitra	Memasukkan Variasi ID Mitra (Dummy) untuk dihapus	Mitra dengan ID yang ditunjuk Hilang dari Basis Data Apabila Tidak Memiliki Relasi dengan dokumen_detail	Sesuai			
5	GET /users	Menyimpan Token Pada Request Body dan Memanggil Rute /users dengan Metode GET	Informasi Seluruh User ditampilkan	Sesuai			
6	PUT /users	Memasukkan ID Pengguna dan Assign Access Level Beserta Mitranya (Dummy) Secara Bervariasi	Data Pengguna Berubah dan Pengguna Memiliki Hak Akses yang Sesuai Setelah Re-login	Sesuai			
7	GET /dokumen	Menyimpan Token Pada Request Body dan	Seluruh Data Dokumen Perjanjian ditampilkan	Sesuai			

No.	Fungsi yang Diuji	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian			
				Admin	Rektorat	PJS	Guest
		Memanggil Rute /dokumen dengan Metode GET	Sesuai dengan Tingkat access_level ID dan Mitra pada Token				
8	POST /dokumen	Memasukkan Token pada Request Body dan Variasi pada Seluruh Parameter dan Mengunggah Berkas..pdf	Data Dokumen Baru Masuk ke Basis Data dan Berkas yang diunggah Berhasil ditaruh Dalam Folder "docs/"	Sesuai			
9	PUT /dokumen	Memasukkan Token pada Request Body dan Variasi pada Seluruh Parameter dan Mengunggah Berkas .pdf	Data Dokumen Baru Masuk ke Basis Data dan Berkas yang diunggah Berhasil ditaruh Dalam Folder "docs/", dan Berkas yang Lama terhapus Apabila Ada	Sesuai			
10	DELETE /dokumen	Memasukkan Token pada Reqest Body dan Memilih ID Dokumen yang akan ditandai untuk dihapus	Status "is_deleted" Menjadi True pada Dokumen Perjanjian dengan ID yang dituju	Sesuai			
11	POST /renew/ :dokumen_id	Menyimpan Token pada Request Body dan dokumen_id pada Request Header lalu Panggil Rute	Dokumen yang dituju diperpanjang Masa Berlakunya Sesuai dengan Waktu Berlakunya Perjanjian Sebelumnya	Sesuai			
12	GET /docs/ :dokumen_id	Menyimpan Token pada Request Body dan dokumen_id pada Request Header lalu Panggil Rute	Berkas yang diminta Langsung diunduh oleh Peramban Klien Apabila Level Akses dan ID Mitra pada Token Sesuai	Sesuai			
13	GET /detail	Menyimpan Token pada Request Body lalu Panggil Rute	Informasi Detail dari Dokumen ditampilkan	Sesuai			
14	POST /detail	Memasukkan Token dan Variasi Seluruh Parameter pada Request Body	Mitra yang Terlibat dengan Dokumen Perjanjian Bertambah	Sesuai			
15	PUT /detail	Memasukkan Token dan Variasi Seluruh Parameter pada Request Body	Peran, Nomor Dokumen, dan PIC Terpilih Milik Mitra yang Terlibat dengan Dokumen Berubah	Sesuai			
16	DELETE /detail	Memasukkan Token, ID Dokumen dan ID Mitra pada Request Body	Mitra dengan ID yang dimasukkan Menjadi Tidak Terlibat dengan Dokumen Perjanjian	Sesuai			
17	GET /jenisDok	Menyimpan Token pada Request Body dan Memanggil Rute	Master Data Jenis Dokumen Ditampilkan	Sesuai			
18	GET /jenisMitra	Menyimpan Token pada Request Body dan Memanggil Rute	Master Data Jenis Mitra ditampilkan	Sesuai			
19	GET /kategoriMitra	Menyimpan Token pada Request Body dan Memanggil Rute	Master Data Kategori Mitra ditampilkan	Sesuai			
20	GET /negara	Menyimpan Token pada Request Body dan Memanggil Rute	Master Data Negara ditampilkan	Sesuai			
21	POST /login	Menyimpan Username dan Password pada	Pengguna Berhasil Login Menggunakan Akun SAT Apabila Merupakan Dosen	Sesuai			

No.	Fungsi yang Diuji	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian			
				Admin	Rektorat	PJS	Guest
		Request Body dan Memanggil Rute					
22	POST /logout	Menyimpan Token pada Request Body dan Memanggil Rute	Kedua Token Pengguna pada Basis Data Lokal dihapus	Sesuai			

VI. SIMPULAN

Melalui aplikasi backend yang telah dibuat pada penelitian ini, Direktorat Kerja Sama Universitas Kristen Maranatha (DKS) dapat mengakses informasi kerja sama yang pernah atau akan dilaksanakan. Pembuatan API ini mendukung proses *maintenance* yang lebih baik dikarenakan pemisahan antara sisi *frontend* dan *backend*. Penggunaan API untuk setiap *request* terkait memungkinkan informasi kerja sama untuk dapat diakses oleh pihak yang berkepentingan secara real-time serta mendukung kemudahan berbagi data kerja sama dengan pihak internal di universitas/ fakultas/ program studi. Berdasarkan hasil implementasi dengan sisi *frontend*, aplikasi ini dapat menangani proses dan dokumen terkait dengan kerja sama yang dilakukan di Universitas Kristen Maranatha.

DAFTAR PUSTAKA

[1] T. Holowaychuk, “express - npm,” 26 May 2019. [Online]. Available: https://www.npmjs.com/package/express. [Accessed 1 January 2020].

[2] D. Wilson, “body-parser - npm,” 26 April 2019. [Online]. Available: https://www.npmjs.com/package/body-parser. [Accessed 1 January 2020].

[3] D. Wilson and T. Goode, “cors - npm,” 5 November 2018. [Online]. Available: https://www.npmjs.com/package/cors. [Accessed 8 January 2020].

[4] R. Turner, A. Sidorov and S. Dhiman, “mysql2 - npm,” 24 December 2019. [Online]. Available: https://www.npmjs.com/package/mysql2. [Accessed 1 January 2020].

[5] J. Senkpiel, T. Holowaychuk and D. Wilson, “method-override - npm,” 12 July 2018. [Online]. Available: https://www.npmjs.com/package/method-override. [Accessed 11 January 2020].

[6] H. Yaapa, J. Pfluger and L. Unnebäck, “multer - npm,” 16 July 2019. [Online]. Available: https://www.npmjs.com/package/multer. [Accessed 3 February 2020].

[7] D. Schenkelman, S. Iacomuzzi and J. Hanson, “jsonwebtoken - npm,” Auth0, 18 March 2019. [Online]. Available: https://www.npmjs.com/package/jsonwebtoken. [Accessed 1 February 2020].

[8] A. Swain, R. Shtylman and J. Firebaugh, “bcrypt - npm,” 18 November 2019. [Online]. Available: https://www.npmjs.com/package/bcrypt. [Accessed 27 January 2020].

[9] H. Singh, “node-input-validator - npm,” 4 February 2020. [Online]. Available: https://www.npmjs.com/package/node-input-validator. [Accessed 4 April 2020].

[10] C. Dewi and A. W. Sasongko, “Sistem Pelaporan Infrastruktur Dinas Bina Marga dan PSDA Kota Salatiga Menggunakan NodeJs Berbasis Web,” *Indonesian Journal of Computing and Modeling*, vol. 1, no. 1, pp. 10-17, 2018.

[11] H. Zhong and M. Hong, “An Empirical Study on API Usages,” *IEEE Transactions on Software Engineering*, vol. 45, no. 4, pp. 319-334, 1 April 2019.

[12] J. L. Martinez, G. Chami, A. Montoute and D. Mohammed, Changing Cuba-U.S. Relations: Implications for CARICOM States, Switzerland: Palgrave Macmillan, 2019.

[13] R. K. A. Natadilaga, “Dashboard Direktorat Kerja Sama dengan Vue.js,” Universitas Kristen Maranatha, Bandung, 2020.