Implementasi metode pengujian equivalence partitioning pada pengembangan RESTful API Sistem Informasi Klinik Pratama UPN "Veteran" Yogyakarta.

Muhamad Iskhak[1], Syafaldi Rizkika[2]

[1,2] Informatika, Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia

[1]123180089@student.upnyk.ac.id, [2]123180127@student.upnyk.ac.id

Keywords: Equivalence Partitioning; Blackbox; Testing; RESTful API; REST; API

Abstract

Purpose:
Implement the Black box testing method with equivalence partitioning technique to measure the quality of RESTful API of Information System Klinik UPN "Veteran" Yogyakarta has been developing.

Design/methodology/approach:
This research was through several stages, starting from the design of the API, implementation, and testing RESTful API that had been created.

Findings/result:
The RESTful API has been runs well, as evidenced by no errors that occur in the system when testing using the equivalence partitioning technique

Originality/value/state of the art:
This research using blackbox testing method with equivalence partitioning technique on RESTful API used on "Klinik Pratama UPN Veteran Yogyakarta" information system

Kata kunci: *Equivalence Partitioning*; *Blackbox*; *Testing*; RESTful API; REST; API

Abstrak

Tujuan:
Menerapkan metode *blackbox testing* dengan teknik *equivalence partitioning* untuk menguji kualitas RESTful API Sistem Informasi Klinik Pratama UPN "Veteran" Yogyakarta yang sedang dikembangkan.

Perancangan/metode/pendekatan:
Penelitian ini dilakukan dengan beberapa tahapan, dimulai dari perancangan desain API, implementasi, dan pengujian RESTful API yang telah dibuat.

Hasil:
RESTful API Klinik Pratama UPN Veteran Yogyakarta berjalan dengan baik dibuktikan dengan tidak ada kesalahan yang terjadi pada sistem ketika dilakukan pengujian dengan menggunakan teknik *equivalence partitioning.*

Keaslian/ state of the art:
Penelitian yang dibuat menggunakan metode *black box* dengan teknik *equivalence partitioning* dengan objek RESTful API dan studi kasus Sistem Informasi Klinik Pratama UPN Veteran Yogyakarta.

1. Pendahuluan

Application Programming Interface (API) merupakan sebuah interface yang menghubungkan suatu aplikasi dengan aplikasi lainya[1,2,3]. Dengan adanya API memungkinkan suatu perangkat lunak melakukan proses operasi logika terhadap suatu basis data tanpa mengetahui detail prosesnya pada perangkat lunak yang sedang di kembangkan[4]. Penerapan API juga mempermudah pengembang dalam melakukan perbaikan atau perubahan sistem dikarenakan proses logika terhadap basis data dengan logika terhadap tampilan antarmuka menjadi suatu program yang terpisah. Serta mempermudah pengembang dalam mengembangkan sistem dengan berbagai jenis platform tanpa perlu membuat ulang proses logika terhadap basis data yang telah dibuat.

Dalam pengembangan API diperlukan sebuah gaya arsitektur untuk pedoman berkomunikasi antara API dengan sistem yang akan terhubung ke API. Terdapat dua arsitektur utama API yaitu, *Simple Object Access Protocol* (SOAP) dan *Representational State Transfer* (REST)[3,5,6]. Dari kedua arsitektur tersebut, arsitektur REST lebih efektif dalam penggunaan *bandwidth*, akses layanan dan pola *request*[6,7]. API yang menggunakan REST sebagai arsitektur disebut sebagai RESTful API[6]. Arsitektur REST merupakan arsitektur yang menerapkan *Hyper Text Type Protocol* (HTTP) untuk melakukan komunikasi. Arsitektur ini bersifat *client - server* dengan melakukan transaksi dimana *client* melakukan *request* pada RESTful API dan RESTful API menerima *request* lalu RESTful API melakukan *request* ke *server* dan mendapat *response* selanjutnya RESTful API mengirim *response* kepada *client*[8]. Dan setiap transaksi yang terjadi merupakan hal yang independen tidak terkait antara satu dengan yang lain[8]. Hal tersebut dapat mempermudah pengembang dalam memperbaiki sistem apabila terdapat kesalahan *response* yang tidak sesuai dengan request yang dikirimkan. Penelitian ini menggunakan gaya arsitektur REST dalam

pengembangan API sebagai *back-end* sistem informasi Klinik UPN Veteran Yogyakarta, agar sistem tetap bisa beradaptasi dan bisa digunakan di berbagai platform.

Dalam pengembangan RESTful API sistem informasi Klinik UPN Veteran Yogyakarta perlu dilakukan pengujian perangkat lunak terhadap RESTful API yang dibuat untuk menemukan kesalahan yang ada agar dapat segera diperbaiki[7]. Pengujian tersebut juga berguna memastikan kualitas dan performa dari RESTful yang dibuat[7, 9, 10, 11]. Terdapat beberapa jenis pengujian perangkat lunak diantaranya *black box testing* yang merupakan pengujian yang berfokus pada fungsional dari perangkat lunak[12]. Pada *black box testing* terdapat beberapa teknik pengujian di antaranya *equivalence partitioning testing* yaitu teknik pengujian *black box* dengan menguji kesesuaian antara masukan data dan keluaran data[13]. Dengan menerapkan kedua metode *black box testing* tersebut dapat membantu pengembang dalam melakukan pengujian dan memastikan perangkat lunak yang dibuat sesuai dengan harapan. Pada penelitian ini, penulis menerapkan metode pengujian *black box testing* dengan teknik *equivalence partitioning* pada RESTful API yang telah dibuat untuk mengetahui kesesuaian antara sistem yang dibuat dengan yang diharapkan oleh pengembang.

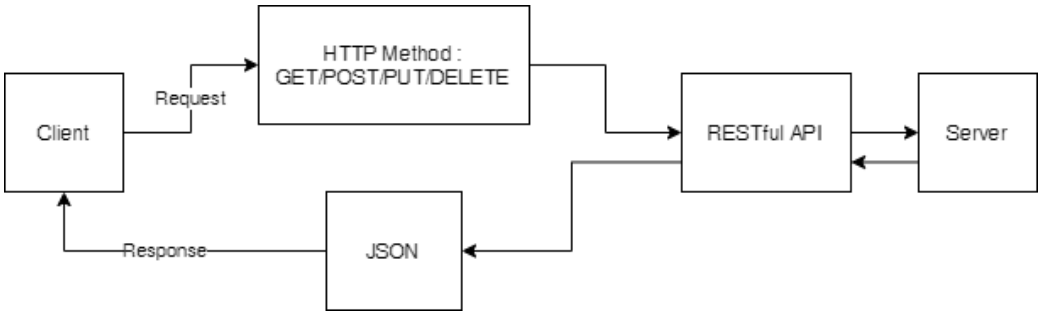
Sebelumnya telah terdapat beberapa penelitian yang berkaitan dengan pengujian perangkat lunak. diantaranya, pengujian menggunakan metode *black box testing* dengan teknik *boundary value analisis* pada perangkat lunak kantor digital politeknik negeri lampung yang dilakukan oleh Tri Snadika Jaya, hasil penelitian menunjukan bahwa perangkat lunak mampu menangani data dengan persentase keberhasilan 91,67%[14]. Penelitian menggunakan metode *black box testing* dengan teknik *boundary value analysis* lain telah dilakukan oleh M. Sidi Mustaqbal, Roeri Fajri Firdaus, dan Hendra Rahmadi dengan studi kasus yang berbeda, menghasilkan beberapa kesimpulan diantaranya fungsionalitas perangkat lunak masih dapat berjalan namun masih memiliki kelemahan kurangnya validasi yang menyebabkan data yang disimpan kurang valid[15]. Penelitian mengenai pengujian perangkat lunak menggunakan metode *blackbox* lain telah dilakukan oleh Mursyidah dan Hari Toha Hidayah pada sistem informasi akuntansi biaya operasional sekolah, hasil penelitian menunjukan tidak ditemukan permasalahan pada sistem[16]. Penelitian lain juga dilakukan oleh Umi Salamah dan Fata Nidaul Khasanah pada Sistem Informasi Penjualan Undangan Pernikahan Online Berbasis Web menggunakan Metode *Black Box Testing* dengan hasil penelitian menunjukan bahwa tidak terdapat kesalahan pada sistem yang dibuat[17]. Pada penelitian ini terdapat perbedaan dengan beberapa penelitian sebelumnya baik dari segi objek, studi kasus ataupun metode yang digunakan.

2. Metode/Perancangan

Pengembangan sistem informasi klinik UPN "Veteran" Yogyakarta terbagi dilakukan dengan membagi 3 bagian secara terpisah yaitu, *back end*, *web front end* dan *mobile front end*. Pada penelitian ini berfokus pada pengembangan *back end* dalam pembuatan RESTful API dengan menerapkan metode pengujian *black box testing* menggunakan teknik *equivalence partitioning*. Adapun tahapan penelitian yang dilakukan yaitu,

2.1. Perancangan REST API

Pada tahap ini peneliti melakukan perancangan desain REST pada API yang akan dibuat. API yang dirancang akan digunakan untuk menghubungkan *client* dengan REST service dengan cara mengambil, mengirim, dan mengubah *resource* yang ada di *database* yang diakses menggunakan *HyperText Transfer Protocol* (HTTP) antara *client* dengan *server* menggunakan konsep *request-response*[8,18] dengan Javascript Object Notation (JSON) sebagai standar format komunikasi[3], sehingga komunikasi antara RESTful API dengan client dapat dilihat pada **Gambar 1.**



Gambar 1. Komunikasi client dengan server melalui RESTful API

Pada desain REST API ini, masing-masing *resource* akan diklasifikasikan berdasarkan *path* dan *query* string dalam *Uniform Resource Identifier* (URI), yaitu sebuah untaian karakter yang digunakan untuk mengidentifikasi nama, atau sumber yang akan digunakan untuk mengidentifikasi dan membedakan *resource* yang akan dipakai, serta *request method* dalam HTTP *request*. Klasifikasi *resource* berdasarkan *path* dibagi menjadi dua jenis seperti berikut.

1. Endpoint /resource akan digunakan untuk request mendapatkan daftar dan menambah data dari *resource* tersebut.
2. Endpoint /resource/id akan digunakan untuk request mendapatkan detail, mengubah dan menghapus *resource* berdasarkan nomor id dari *resource* yang dikirim.

Sementara klasifikasi berdasarkan *HTTP request method* akan dibagi menjadi empat jenis seperti berikut.

1. Request method POST akan digunakan untuk penambahan *resource*
2. Request method GET akan digunakan untuk mendapatkan data dari *resource*
3. Request method PUT akan digunakan untuk mengubah data dari *resource*
4. Request method DELETE akan digunakan untuk menghapus *resource*

2.2. Implementasi

Pada tahap implementasi, penulis membangun RESTful API baru dengan menggunakan bahasa pemrograman *Hypertext Preprocessor* (PHP) dengan framework codeigniter versi ke 4.1.3 sesuai dengan desain rancangan RESTful API yang telah dibuat.

2.3. Pengujian API

Secara umum, pengujian yang dilakukan terhadap RESTful API yaitu[19] :

1. Nilai yang diberikan sesuai dengan kondisi yang diberikan.
2. Memastikan bahwa API tidak memberikan nilai apapun selain kondisi.
3. Memastikan jika API tidak merubah struktur data.

Ada beberapa jenis pengujian perangkat lunak, antara lain :

1. *White box Testing* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan white box testing merupakan petunjuk untuk mendapatkan program yang benar secara 100% [12].
2. *Black Box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program[12]. Proses *Black Box Testing* dengan cara mencoba program yang telah dibuat dengan mencoba memasukkan data pada setiap formnya. Pengujian ini diperlukan untuk mengetahui program tersebut berjalan sesuai dengan yang dibutuhkan oleh perusahaan[13].

Dalam penelitian ini, penulis menggunakan metode *blackbox testing* dengan teknik *equivalence partitioning* yaitu teknik pengujian *black box* dengan menguji kesesuaian antara masukan data dan keluaran data yang diharapkan[13] pada RESTful API Sistem Informasi Klinik UPN Veteran Yogyakarta yang telah dikembangkan, dengan mengujinya menggunakan perangkat lunak postman[20].

3. Hasil dan Pembahasan

3.1. Implementasi rancangan RESTful API

Dari hasil API yang telah dirancang akan dipetakan dan dikategorikan berdasarkan nama proses dan *request method*, sebagian hasil dari perancangan dapat dilihat dari **Tabel 2.** dengan keterangan :

1. Nama proses adalah nama dari proses yang bisa dilakukan dengan *request method* dan URI yang diberikan.
2. Request method adalah metode HTTP yang digunakan pada URI tersebut, ada 4 request method yang akan dipakai. Yaitu,
    1. GET

       Metode GET akan dipakai untuk mengambil *resource* sesuai yang tertera pada URI dari *database*, contoh hasil dari metode GET adalah seperti **Gambar 2.**

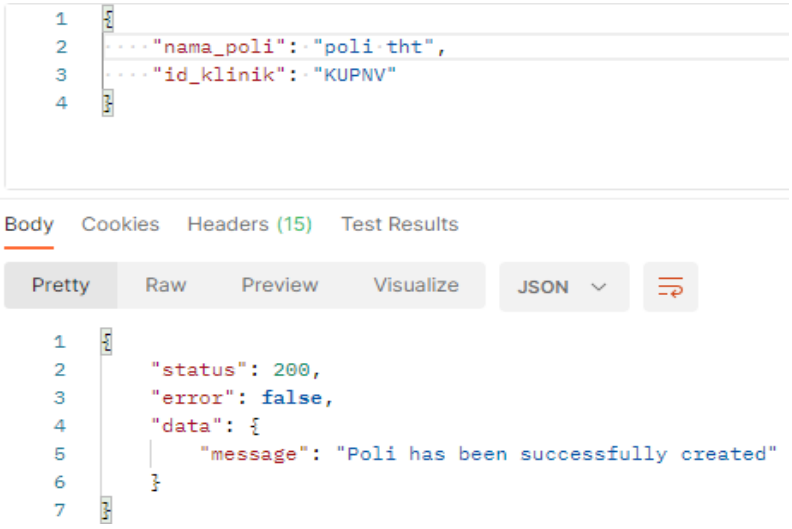
ISSN: 1979-2328


```
{
    "status": 200,
    "error": false,
    "data": [
        {
            "id_poli": "1",
            "nama_poli": "Poli Umum",
            "id_klinik": "KUPNV"
        },
        {
            "id_poli": "2",
            "nama_poli": "poli gigi",
            "id_klinik": "KUPNV"
        },
        {
            "id_poli": "7",
            "nama_poli": "poli jantung",
            "id_klinik": "KUPNV"
        },
```


Gambar 2. contoh hasil dari *request* dengan metode GET

2. POST

Metode POST dipakai untuk mengirim *resource* yang tertera pada URI ke *database*, contoh hasil dari metode POST adalah seperti **Gambar 3.**

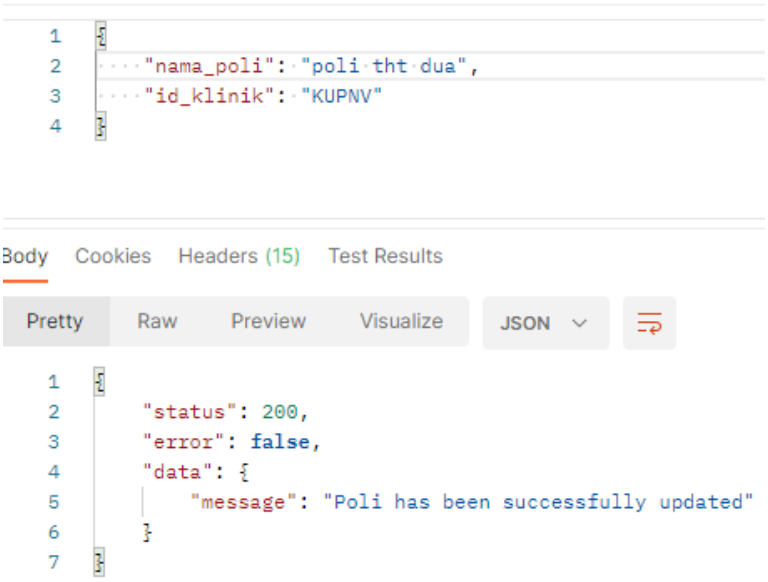


Gambar 3. contoh hasil dari *request* dengan metode POST

3. PUT

Metode PUT dipakai untuk mengirim dan mengganti data salah satu atau sebagian anggota dari *resource*, contoh hasil dari *request* metode PUT adalah seperti **Gambar 4.**.

 ISSN: 1979-2328



Gambar 4. contoh hasil dari *request* dengan metode PUT

4. DELETE

Metode DELETE dipakai untuk menghapus salah satu atau sebagian *resource* dari *database*, contoh hasil dari metode DELETE adalah seperti **Gambar 5.**

Gambar 5. contoh hasil dari *request* dengan metode DELETE

Tabel 1. Proses dan *request method* yang dipakai di dalam API

No	Nama Proses	Request Method	URI
1	menampilkan semua data poli	GET	/poli
2	menampilkan daftar poli sesuai dengan id klinik	GET	/web/datamaster/getpoli/<id_klinik>
3	menampilkan data poli tertentu sesuai dengan id	GET	/poli/<id_poli>

ISSN: 1979-2328

4	mengubah data poli tertentu	PUT	/poli/<id_poli>
5	delete data poli	DELETE	/poli/<id_poli>
6	menambahkan data poli	POST	/poli

3.2. Pengujian

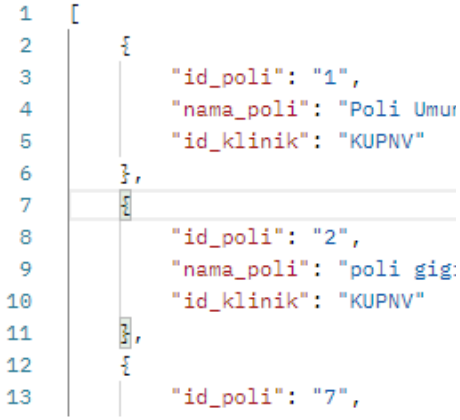
Pengujian API dilakukan dengan menguji masukan sistem menggunakan metode *equivalence partitioning* pada sistem.

3.2.1. Pengujian Equivalence Partitioning

Pengujian teknik *equivalence partitioning* dilakukan dengan menguji path dan request method API menggunakan perangkat lunak postman. Berikut adalah cuplikan hasil pengujian yang sudah dilakukan penulis dapat dilihat pada **Tabel 2**. dengan keterangan sebagai berikut.

1. Daftar Pengujian adalah jenis validasi yang dilakukan untuk kasus uji.
2. Kasus Uji adalah daftar kasus uji yang akan dilakukan untuk testing.
3. Hasil yang Diharapkan adalah respon yang diharapkan untuk terjadi pada kasus uji.
4. Hasil Postman adalah screenshot hasil dari kasus uji yang dilakukan.
5. Hasil adalah kesesuaian dari hasil postman dengan hasil yang diharapkan.

Tabel 2. Kasus yang di uji dan hasilnya

Daftar Pengujian	**Kasus Uji**	**Hasil yang Diharapkan**	**Hasil Postman**	**Hasil**
Validasi Path	Request method GET untuk /poli	Mendapatkan daftar poli		Berhasil

ISSN: 1979-2328

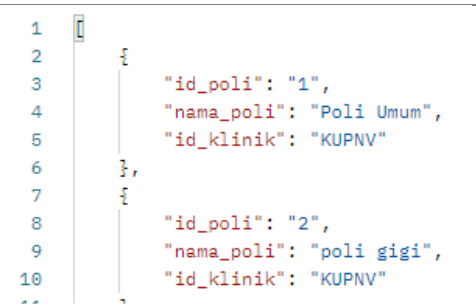
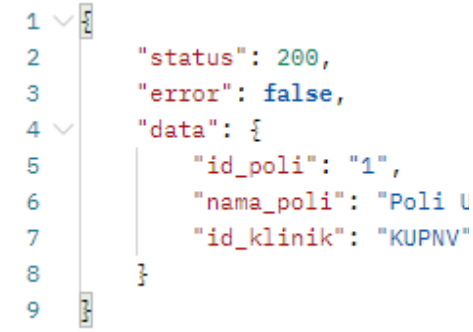
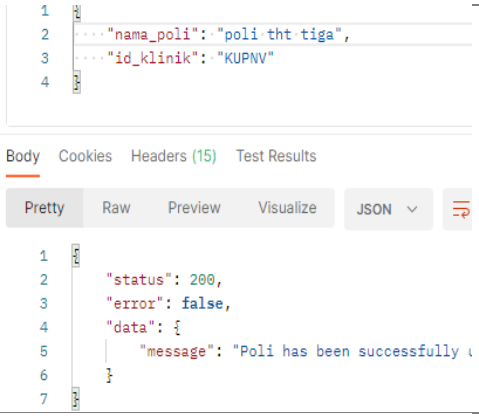
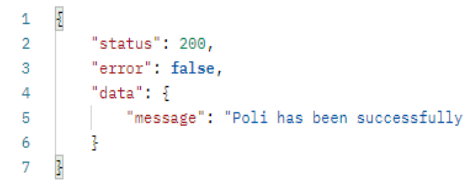
	Request method GET untuk /poli/1	Mendapatkan data poli dengan id 1	{
			"status": 200,
			"error": false,
			"data": {
			"id_poli": "1",
			"nama_poli": "Poli Umum
			"id_klinik": "KUPNV"
			}
			}
				Berhasil

	Request method GET untuk /web/datamaster/getpoli/KUPNV	Mendapatkan daftar poli dengan kode klinik KUPNV	{ "status": 200, "error": false, "data": [{ "id_poli": "1", "nama_poli": "Poli Umum" "id_klinik": "KUPNV" }, { "id_poli": "2", "nama_poli": "poli gigi" "id_klinik": "KUPNV" }, { "id_poli": "7", "nama_poli": "poli jantu "id_klinik": "KUPNV" },	Berhasil

	Request method GET untuk /poli/UMUM	Error dengan kode 404 *not found*	{
			"status": 404,
			"error": true,
			"data": {
			"message": "Not Founc
			}
			}
				Berhasil

Validasi Request Method	Request method POST untuk /poli	Menginput record poli baru	{
			"nama_poli": "poli tht dua",
			"id_klinik": "KUPNV"
			}
			Body Cookies Headers (15) Test Results
			Pretty Raw Preview Visualize JSON
			{
			"status": 200,
			"error": false,
			"data": {
			"message": "Poli has been successfully
			}
			}
				Berhasil

 ISSN: 1979-2328

Request method GET untuk /poli		Mendapatkan daftar poli		Berhasil
Request method GET untuk /poli/1		Mendapatkan data poli dengan id poli 1		Berhasil
Request method PUT untuk /poli/1		Memperbarui atau mengubah data poli dengan id poli 1		Berhasil
Request method DELETE untuk /poli/12		Menghapus data poli dengan id poli 12		Berhasil

Penjelasan pengujian yang dilakukan pada **Tabel 2.** dapat dilihat sebagai berikut.

- Dalam request method GET /poli **Tabel 2.** peneliti melakukan validasi path terhadap endpoint RESTful API yang telah dibuat dimana endpoint dengan path /poli berfungsi untuk mengambil semua data poli dari database, dan di kolom hasil postman telah berhasil mengeluarkan *output* semua data poli yang berada di database dalam bentuk JSON.

- Request method GET /poli/1 pada **Tabel 2.** peneliti melakukan validasi path endpoint API yang berfungsi untuk mengambil data poli yang mempunyai atribut id_poli bernilai 1 dari database, di kolom hasil postman telah berhasil mengeluarkan *output* data poli yang mempunyai atribut id_poli bernilai 1 dari database ke dalam bentuk JSON
- Request method POST /poli pada **Tabel 2.** peneliti melakukan kasus uji dari validasi terhadap method request post yang diharapkan berfungsi untuk memasukan record poli baru ke dalam database. yang pada pengujian ini atribut nama_poli berisi poli tht dua, dan atribut id_klinik berisi KUPNV. Dan setelah berhasil mamasukan data ke dalam database RESTful API mengirimkan status code berisi 200, kondisi errornya false dan data berupa message yang berisi poli has been successfully created. Kasus uji telah berhasil berjalan sesuai dengan yang diharapkan.
- Request method PUT /poli/1 pada **Tabel 2.** peneliti melakukan kasus uji dari validasi terhadap method request put yang diharapkan berfungsi untuk memperbarui record yang ada pada database dengan atribut id_poli bernilai 1 dan data sesuai record yang dikirimkan dan setelah berhasil maka RESTful API mengirimkan status code berisi 200, kondisi errornya false dan data berupa message yang berisi poli has been successfully updated. Kasus uji telah berhasil berjalan sesuai dengan yang diharapkan.
- Request method DELETE /poli/12 pada **Tabel 2.** peneliti melakukan kasus uji dari validasi terhadap method request DELETE untuk menghapus data poli yang mempunyai atribut id_poli bernilai 12 dari database, di kolom hasil postman, pada pengujian yang dilakukan peneliti telah berhasil menghapus data poli yang mempunyai atribut id_poli bernilai 12 dari database.

Dari pengujian yang telah dilakukan didapatkan semua kasus uji pengujian yang dibuat telah mendapatkan hasil sesuai dengan yang diharapkan.

4. Kesimpulan dan Saran

Berdasarkan pengujian yang telah dilakukan pada RESTful API Sistem Informasi Klinik UPN Veteran Yogyakarta menggunakan metode *black box* dengan teknik e*quivalence partitioning*, dapat disimpulkan bahwa RESTful API telah berhasil dikembangkan dengan baik ditunjukan dengan kasus - kasus uji yang telah dilakukan berhasil sesuai dengan yang diharapkan. Adapun untuk penelitian RESTful API selanjutnya penulis memberikan saran agar diterapkan teknik pengujian *black box testing* lainya seperti *behavior testing* untuk menemukan kesalahan yang terjadi pada sistem yang dibuat dan membuat kualitas sistem menjadi lebih baik.

Daftar Pustaka

[1] B. Adi Pranata, A. Hijriani, and A. Junaidi, "Perancangan Application Programming Interface (Api) Berbasis Web Menggunakan Gaya Arsitektur Representational State Transfer (Rest) Untuk Pengembangan Sistem Informasi Administrasi Pasien Klinik Perawatan Kulit," *J. Komputasi*, vol. 6, no. 1, pp. 33–42, 2018, doi: 10.23960/komputasi.v6i1.1554.

[2] S. Lee, J. Jo and Y. Kim, "Method for secure RESTful web service," 2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS), 2015, pp. 77-81, doi: 10.1109/ICIS.2015.7166573.

[3] W. G. Wardhana, I. Arwani, and B. Rahayudi, "Implementasi Teknologi Restful Web Service Dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website (Studi Kasus : Fakultas Teknologi Pertanian Universitas Brawijaya)," *J. Pengemb. Teknol. Inf. dan Ilmu Komputer.*, vol. 4, no. 2, pp. 680–689, 2020, [Online]. Available: http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/7024.

[4] A. Barros, C. Ouyang, and F. Wei, "Static Analysis for Improved Modularity of Procedural Web Application Programming Interfaces," *IEEE Access*, vol. 8, pp. 128182–128199, 2020, doi: 10.1109/ACCESS.2020.3008904.

[5] Y. K. Kurniawan, Y. Oslan, and H. Kristanto, "Implementasi Rest - Api Untuk Portal Akademik Ukdw Berbasis Android," *J. EKSIS*, vol. 6, pp. 29–40, 2013.

[6] S. Malik and D. H. Kim, "A comparison of RESTful vs. SOAP web services in actuator networks," *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, pp. 753–755, 2017, doi: 10.1109/ICUFN.2017.7993893.

[7] H. Pei, B. Yin, M. Xie, and K. Cai, "Dynamic random testing with test case clustering and distance-based parameter adjustment," *Inf. Softw. Technol.*, vol. 131, no. October 2020, p. 106470, 2021, doi: 10.1016/j.infsof.2020.106470.

[8] M. W. R. Fakhrun and S. F. S. Gumilang, "Rancangan Web Service Dengan Metode Rest Api Untuk Integrasi Aplikasi Mobile Dan Website Pada Bank Sampah," *Konf. Nas. Sist. Inf.*, pp. 214–219, 2018.

[9] B. E. Partitioning, "An Effective Equivalence Partitioning Method to Design the Test Case of the WEB Application,". 2012 International Conference on Systems and Informatics (ICSAI 2012), no. Icsai, pp. 2524–2527, 2012.

[10] Z. Zhang, T. Wu and J. Zhang, "Boundary value analysis in automatic white-box test generation," 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE), 2015, pp. 239-249, doi: 10.1109/ISSRE.2015.7381817.

[11] F. Dobslaw, F. G. de Oliveira Neto and R. Feldt, "Boundary Value Exploration for Software Analysis," 2020 IEEE International Conference on Software Testing, Verification

and Validation Workshops (ICSTW), 2020, pp. 346-353, doi: 10.1109/ICSTW50294.2020.00062.

[12] T. Hidayat and M. Muttaqin, "Pengujian Sistem Informasi Pendaftaran dan Pembayaran Wisuda Online menggunakan Black Box Testing dengan Metode Equivalence Partitioning dan Boundary Value Analysis," *J. Tek. Inform. UNIS JUTIS*, vol. 6, no. 1, pp. 2252–5351, 2018, [Online]. Available: www.ccssenet.org/cis.

[13] B. A. Priyaungga, D. B. Aji, M. Syahroni, N. T. S. Aji, and A. Saifudin, "Pengujian Black Box pada Aplikasi Sistem Seleksi Sales Terbaik Menggunakan Teknik Equivalence Partitions," *J. Teknol. Sist. Inf. dan Apl.*, vol. 3, no. 3, p. 150, 2020, doi: 10.32493/jtsi.v3i3.5343.

[14] Jaya, Tri Snadhika, "Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus : Kantor Digital Politeknik Negeri Lampung)", Jurnal Informatika: Jurnal Pengembangan IT (JPIT). Vol.03, pp. 45-48, Jan. 2018.

[15] Mustaqbal, M. Sidi, Firdaus, Roeri Fajri and Rahmadi, Hendra, "PENGUJIAN APLIKASI MENGGUNAKAN BLACK BOX TESTING BOUNDARY VALUE ANALYSIS (Studi Kasus : Aplikasi Prediksi Kelulusan SNMPTN)", Jurnal Ilmiah Teknologi Informasi Terapan (JITTER). Vol. I, pp. 31-36, Agu. 2015

[16] Mursyidah and Hidayat, Hari Toha, "Pengujian Sistem Informasi Akuntansi Biaya Operasional Sekolah Dengan Black Box Testing", Jurnal Infomedia. Vol. 2, pp 7-14, Des. 2017

[17] Salamah, Umi and Khasanah, Fata Nidaul, "Pengujian Sistem Informasi Penjualan Undangan Pernikahan Online Berbasis Web Menggunakan Black Box Testing", Information Management For Educators And Professionals. Vol. 2, pp. 35-46, Des. 2017.

[18] B. Costa, P. F. Pires, F. C. Delicato and P. Merson, "Evaluating a Representational State Transfer (REST) Architecture: What is the Impact of REST in My Architecture?," 2014 IEEE/IFIP Conference on Software Architecture, 2014, pp. 105-114, doi: 10.1109/WICSA.2014.29.

[19] PAHLEVY, BACHTIAR ARIEF (2017) RANCANG BANGUN APPLICATION PROGRAMMING INTERFACE (API) ONGKOS KIRIM DAN TRACKING EKSPEDISI INDONESIA. Other thesis, University of Muhammadiyah Malang.

[20] Postdot Technology. 2017. Postman is the most complete API Development Environment. [Online]. San Francisco. Tersedia: https://www.getpostman.com/postman [13 September 2017].