*JIKA (Jurnal of Informatics) Universitas Muhammadiyah Tangerang* *P ISSN : 2549-0710*
*Vol 8, No 2, April 2024, pp 238-245* *E ISSN : 2722-2713*
*DOI : 10.31000/jika.v8i2.10977*

# RANCANG BANGUN API MANAGEMENT PADA APLIKASI SSC MENGGUNAKAN FRAMEWORK WEBIX DI PT XYZ

**Bayu Andy Daniswara[1)], Yeremia Alfa Susetyo[2)]**
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60 Salatiga
Co Responden Email: 672020258@student.uksw.edu

***Article history***
*Received 24 Feb 2024*
*Revised 14 Mar 2024*
*Accepted 16 Apr 2024*
*Available online 30 Apr 2024*

***Keywords***
*API,*
*Management API,*
*Webix Framework*
*Agile*

***Abstract***

***Modern retail companies are increasingly facing challenges in managing various systems and applications that support their business operations. PT XYZ, a leading retail company in Indonesia, has adopted the use of APIs in several of its applications. The company needs to have the skills to manage and monitor APIs by managing access rights and tracking API usage logs. Therefore, the development of API Management systems becomes crucial to enhance efficiency and agility in API administration. This research aims to design and build API Management using the Webix framework. Agile methodology is an approach to software development that originates from iterative work processes involving the application of pre-agreed rules and solutions. By applying Agile methodology, this research involves a series of stages, ranging from requirements, design, development, testing, implementation, to review. The end result of this research is the design of API Management which includes API monitoring, API documentation, and API testing using widgets from the Webix framework.***

**Riwayat**
Diterima 24 Feb 2024
Revisi 14 Mar 2024
Disetujui 16 Apr 2024
Terbit Online 30 Apr 2024

**Kata Kunci**
*API,*
Manajemen API,
*Framework* Webix,
*Agile*

**Abstrak**

*Perusahaan retail modern kini semakin menghadapi tantangan dalam mengelola berbagai sistem dan aplikasi yang mendukung operasional bisnis mereka. PT XYZ, sebuah perusahaan retail terkemuka di Indonesia, mengadopsi penggunaan API pada beberapa aplikasinya. Perusahan harus memiliki keterampilan untuk mengatur serta memantau API dengan mengelola hak akses dan melacak log penggunaan API. Maka dari itu, pengembangan sistem Management API menjadi sangat penting guna meningkatkan efisiensi dan kelincahan dalam administrasi API. Penelitian ini bertujuan merancang dan membangun API Management dengan menggunakan framework Webix. Metode agile merupakan pendekatan pengembangan perangkat lunak yang berasal dari proses kerja berulang yang melibatkan penerapan aturan dan solusi yang telah disetujui sebelumnya. Dengan menerapkan metodologi Agile, penelitian ini melibatkan serangkaian tahapan, mulai dari persyaratan, desain, pengembangan, pengujian, implementasi, hingga peninjauan. Hasil akhir dari penelitian ini adalah rancangan API Management yang mencakup pemantauan API, dokumentasi API, dan pengujian API dengan menggunakan widget dari framework webix.*

## PENDAHULUAN

PT XYZ merupakan salah satu perusahaan besar di Indonesia yang bergerak di industri retail. Pada Tahun 2023, PT XYZ memiliki lebih dari 17.800 cabang toko yang beroperasi. Dalam menangani skala transaksi yang besar tiap harinya, maka diperlukan berbagai sistem dan aplikasi untuk meningkatkan efisiensi operasional. Solusi yang diambil adalah pemanfaatan API (*Application Programming Interface*) yang memungkinkan perusahaan untuk mengintegrasikan antara berbagai sistem dan aplikasi secara efisien.

*Application Programming Interface* (API) adalah seperangkat peraturan yang telah ditetapkan untuk memungkinkan aplikasi yang berbeda berkomunikasi satu sama lain(Mathijssen et al., 2020). Salah satu aspek penting adalah kegunaan API, yang mengukur sejauh mana API sesuai dengan kebutuhan dan harapan audiens sasaran (Meng, Steinhardt, & Schubert, 2020). Kinerja API dalam aplikasi dituntut memiliki performa yang dapat

diandalkan dalam menangani berbagai permintaan dari beragam aplikasi antar platform (Setiawan, Made, Adnyana, & Budiarta, 2022). Setiap perubahan yang tidak terduga dalam API dapat menyebabkan kegagalan besar dalam operasi layanan, hal ini dapat menyebabkan perusahaan menghadapi kerugian finansial(Ehsan et al., 2022).

Pengujian Pengelolaan API yang dilakukan pada PT XYZ masih menggunakan Postman. Salah satu kendala yang dihadapi dalam penggunaan postman pembatasan workspace dan batasan jumlah request API yang dapat disimpan. Hal ini dapat menjadi kendala dalam mengelola semua request API yang diperlukan untuk berbagai keperluan bisnis.

Untuk mengatasi kendala tersebut diperlukannya API Management untuk mengelola API yang ada di PT XYZ. API *Management* adalah proses yang melibatkan pembuatan, publikasi, dan pengelolaan koneksi API. API *Management* juga mencakup data tentang penggunaan, kinerja, dan tren API menjadi bagian penting dari API *Management*. API *Management* terdiri dari beberapa komponen yang bekerja sama untuk mengelola API dengan baik. API *Gateway* merupakan salah satu komponen yang penting dalam API *Management* untuk menangani semua permintaan dalam memastikan keamanan koneksi API. API *Gateway* juga mengumpulkan dan mempublikasi data analitik terkait dengan API (Siriwardena, 2020).

Salah satu teknologi yang dapat digunakan sebagai solusi perancangan API *Management* adalah *framework* Webix. Framework webix adalah sebuah kerangka kerja javascript yang dikembangkan oleh XBSoftware. Penggunaan webix memungkinkan developer lebih mudah dalam mengembangkan *front-end* yang kompleks (Nugroho & Susetyo, n.d.). Dalam perancangan backend, framework flask memberikan fleksibilitas kepada *developer* dalam proses pengembangan aplikasi. Flask dikategorikan sebagai sebuah *micro-framework* karena penggunaanya tidak memerlukan alat atau pustaka khusus (Ningtyas & Setiyawati, 2021).

Selain itu, Flask juga menyediakan perpustakaan dan sejumlah kode program yang dapat digunakan untuk membuat sebuah situs web tanpa perlu membangun semuanya dari awal.

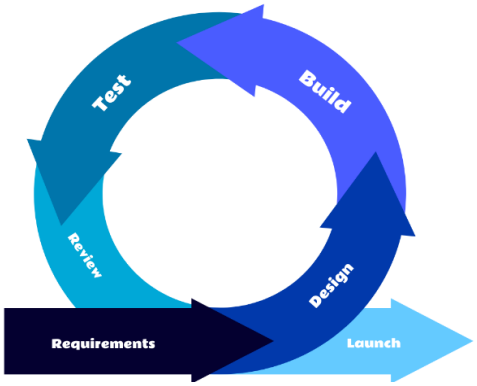
Pada penelitian ini akan dilakukan perancangan API Management menggunakan framework Webix dengan memperhatikan efisiensi dalam mengelola API pada PT XYZ. Tahapan metode yang dilakukan oleh peneliti merupakan metode Agile yang memiliki kerangka kerja proyek dengan memecah proyek menjadi beberapa fase dinamis. Dengan menerapkan metode agile peneliti dapat merancang API Management secara fleksibel. Tahapan metode yang dilakukan meliputi perencanaan, tahap desain, implementasi, *testing*, *review*, dan *deployment*.

Dalam era digital saat ini, banyak perusahaan memerlukan sistem Manajemen API untuk mengelola API dengan efisien. Namun, merancang sebuah sistem Manajemen API memerlukan investasi waktu dan sumber daya yang signifikan. Oleh karena itu, tujuan dari penelitian ini adalah merumuskan pengembangan Manajemen API. Hal ini diharapkan dapat memberikan flesibilitas pada perusahaan dalam mengelola API dengan efisien.

## METODE PENELITIAN

Metode penelitian yang digunakan pada penelitian ini menggunakan metode Agile. Metode Agile adalah metodologi pengembangan *software* yang didasarkan pada proses pengerjaan berulang yang terdiri dari aturan dan solusi yang sudah disepakati. Metode ini dapat disesuaikan dan dilengkapi untuk menangani pengembangan perangkat lunak dalam skala besar, masalah dalam keterlibatan *user*, arsitektur perangkat lunak, dan koordinasi antar tim. Tujuan utamanya adalah untuk mengurangi biaya tambahan dalam proses pengembangan *software* dengan fleksibilitas dalam menerima perubahan tanpa mengorbankan proses atau memerlukan pekerjaan yang berlebihan (Ariesta, Dewi, Sariasih, & Fibriany, 2021). Pada gambar 1 dijelaskan bahwa terdapat beberapa Langkah dalam metode ini, yaitu *requirements*, *design*, *build*, *test*, *review*, dan *launch*.

*JIKA (Jurnal of Informatics) Universitas Muhammadiyah Tangerang* *P ISSN : 2549-0710*
*Vol 8, No 2, April 2024, pp 238-245* *E ISSN : 2722-2713*
*DOI : 10.31000/jika.v8i2.10977*

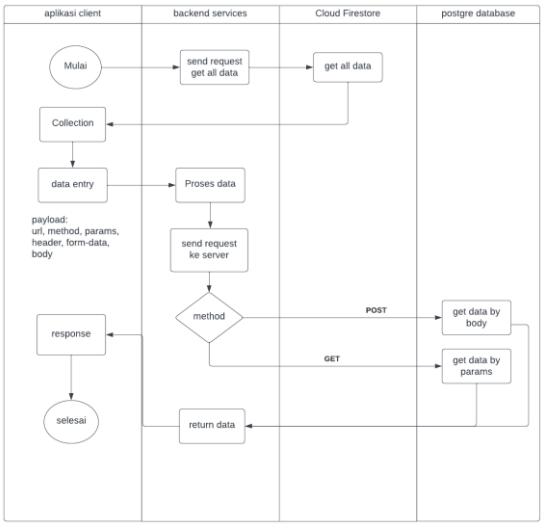



Gambar 1. Tahapan Penelitian Agile

Tahap awal dalam perancangan API Management merupakan *requirements* atau persyaratan yang diperlukan. Komponen yang diperlukan dari suatu API *Management* antara lain API Gateway. API Gateway bertindak sebagai *gatekeeper* untuk semua API dengan menegakkan kebijakan dan permintaan keamanan API yang relevan. Salah satu manfaat dari API Gateway adalah menyembunyikan struktur internal aplikasi. Diperlukannya juga *testing* API Dimana, *user* mengirimkan HTTP *request* dan menerima respond berdasarkan *request* yang dilakukan *user*.

Pada API Management ini juga memerlukan fitur *Environment* yang dapat digunakan untuk membuat *variabel* dengan nilai yang berbeda. *Environment* ini mengambil data *realtime* dari *firebase database*. *Database real-time* dirancang untuk menyesuaikan dengan perubahan data yang terjadi dan menyediakan informasi secara langsung, meningkatkan atau memperbaiki informasi (Wingerath, Ritter, & Gessert, 2019). *Firebase database* merupakan real-time database yang tersimpan di *cloud* dan *support multiplatform* seperti Android, iOS, dan Web. Firebase dapat diintegrasikan dengan berbagai *framework* lain seperti node, java, javascript, dan sebagainya (Ilham Firman Maulana, 2020).

Tahap berikutnya adalah membuat perancangan *design* API Management dengan menggunakan *activity* diagram. *Activity* diagram merupakan spesifikasi semi-formal bersifat intuitif dan fleksibel yang digunakan untuk mendeskripsikan perilaku sistem beserta dengan logika dari operasi yang kompleks.




Gambar 2. Activity Diagram

Activity diagram pada gambar 2 merupakan proses kerja API Management yang menampilkan ajax *request* dalam menangani HTTP *request* yang dilakukan oleh user. Data yang diperoleh dari *cloud* firestore merupakan data dengan format JSON. *Cloud Firestore* merupakan database terbaru dari Firebase untuk pengembangan aplikasi mobile. Dengan menggunakan webix ajax, HTTP *request* yang dilakukan oleh user akan dikemas menjadi suatu payload kemudian dikirimkan ke *services* backend. Proses ini dilakukan dalam menghindari CORS (*Cross-Origin Resource Sharing*) yang merupakan fitur keamanan browser yang membatasi permintaan HTTP.

Tahapan berikutnya adalah *build* atau implementasi *framework* webix pada perancangan API Management. *Framework* webix merupakan struktur terintegrasi yang memanfaatkan Ontology Web Language (OWL) dan Semantic Web Rule Language (SWRL) untuk melakukan penilaian risiko dan manajemen yang dinamis pada berbagai tingkatan (operasional, taktik, dan strategis) (Riesco & Villagrá, 2019). framework webix memiliki arsitektur komponen yang dimodifikasi untuk berinteraksi dengan kontur web, menerapkan desain interface yang dinamis serta manajemen data yang otomatis (Kedrin & Rodyukov, 2021). Dengan memanfaatkan widget UI pada *framework* webix untuk mendapatkan data setiap kali terdapat perubahan data dari *firebase database*. Asynchronous HTTP requests disebut juga dengan *request* Ajax digunakan

untuk berkomunikasi dengan layanan web dan mengambil data (Freeman, 2020). Proses dalam mendapatkan data dari firebase berjalan secara *asynchronous* sehingga user tidak perlu melakukan refresh pada saat terdapat perubahan data.

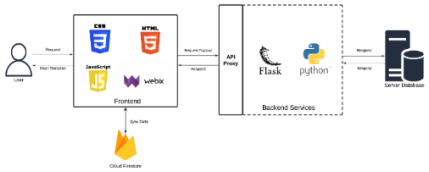
Tahapan berikutnya adalah *testing* atau pengujian API Management apakah sudah berjalan seperti yang diharapkan. Pada penelitian ini menggunakan metode pengujian Black Box Testing. Dengan menggunakan metode Black Box Testing dapat menemukan kesalahan dalam fungsi, *interface*, model data, dan akses ke sumber data eksternal dalam sistemnya tanpa melihat kode program yang digunakan (Hendri, Hasiholan Manurung, Ferian, Hanaatmoko, & Yulianti, 2020).

Tahap berikutnya adalah tahap *review* setelah dilakukannya pengujian API Management. Pada tahap ini peneliti akan melakukan *review* kembali dan membenahi apabila terdapat bug maupun hal yang kurang sesuai. Setelah dilakukannya *review* maka akan dikembalikan pada tahap pertama, apakah terdapat penambahan ketentuan dari *client*.

Tahap terakhir merupakan tahap *launch* atau *deployment* dengan menerapkan rancangan API Management pada aplikasi *Store Support Control* (SSC). Aplikasi tersebut memanfaatkan API dalam memonitor seluruh transaksi, mengidentifikasi permasalahan, perbaikan data pada toko, dan masih banyak fungsi yang lain.

## HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan rancangan API Management yang dapat digunakan untuk mengelola API, baik dalam pengujian API maupun dokumentasi API. Dalam perancang sistem, penggunaan database realtime diperlukan untuk memungkinkan sinkronisasi data secara langsung. Firebase database digunakan untuk menangani kebutuhan dalam sinkronisasi data secara realtime. Dengan mengintegrasikan Firebase SDK untuk JavaScript, framework webix dapat mengambil data dari cloud firestore tanpa menggunakan backend service.




Gambar 3. Arsitektur sistem

Gambar 3 merupakan arsitektur API Management yang telah dibangun dengan menggunakan framework webix, *flask*, dan cloud firestore. Dalam komponen frontend, berbagai teknologi digunakan untuk mendukung tampilan yang menarik dan responsif. Framework webix digunakan untuk merancang tampilan frontend serta melakukan AJAX *request* dan sinkronisasi data secara realtime dari cloud firestore. Backend services bertanggung jawab dalam mengolah payload *request* yang dilakukan user dan juga mengirimkan kembali respond dari database server. Proses dimulai dengan melakukan sinkronisasi data API dari cloud firestore dan menampilkannya pada collection. Pada saat melakukan HTTP *request* user dapat melakukannya dengan memilih API pada collection atau dengan menginputkan endpoint API secara manual. HTTP *request* yang dilakukan oleh user, akan dikrimkan dalam bentuk payload JSON ke backend services dengan menggunakan webix AJAX. Hal ini dilakukan untuk menghindari CORS (Cross-Origin Resource Sharing) yang merupakan fitur kemanan pada browser. Respond kembalian dari backend service akan diolah kembali pada frontend sebelum ditampilkan ke user.

```
1  setUpEnvironmentListener(){
2    var envLogStream = webix.firestore.collection("id_firebase")
3    envLogStream.onSnapshot((snapshot) => {
4      snapshot.docChanges().forEach((change) => {
5        var data = change.doc.data();
6        data.id = change.doc.id;
8        if(change.type === "added"){
         //kodisi pada saat terdapat penambahan data pada firebase database
         }
9        if(change.type === "modified"){
         //kodisi pada saat terdapat perubahan data pada firebase database
         }
10       if(change.type === "removed"){
         //kodisi pada saat terdapat penghapusan data pada firebase database
         }
11     });
12   });
   }
```

Gambar 4. Kode program environment

Gambar 4 merupakan fungsi dalam memperoleh data environment yang disimpan pada firebase database. Data yang diperoleh dari firebase berupa data JSON. Pada baris 2 digunakan untuk mengambil referensi koleksi firestore berdasarkan id firestorenya. Pada baris 3 merupakan metode mendaftarkan *listener* pada objek 'envLogStream' untuk

mendengarkan apabila terdapat perubahan data pada firestore. Setiap kali ada perubahan, fungsi callback ini akan dijalankan. Perulangan yang terdapat pada baris 4 digunakan untuk melakukan tindakan tertentu berdasarkan jenis perubahan yang terjadi.

Penggunaan realtime database pada sistem ini digunakan sehingga user tidak perlu melakukan refresh page pada saat terdapat perubahan data pada database. Hal ini juga dimanfaatkan supaya tidak sembarang user dapat merubah data environment maupun collection Api yang tersimpan di database. Meskipun hal ini menjadikan user tidak fleksibel dalam mengatur environmentnya sendiri, tetapi hal ini diterapkan untuk meningkatkan keamanan pada sistem.

```
1  webix.ajax().headers(
2              {"Content-type": "application/json"}
3  ).post(urlConfig.heldesk_service_url + "//ssc/proxy", payloaditem,{
4    //mengolah respond
5  });
```

Gambar 5. Kode webix ajax

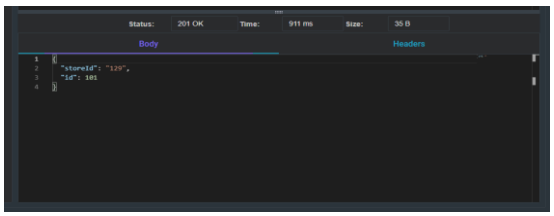
Gambar 5 merupakan fungsi webix ajax untuk melakukan HTTP request yang diarahkan ke backend *services* dengan nama proxy. Pada webix ajax header sudah ditentukan dalam bentuk "application/json" karena setiap ajax request yang dilakukan mengirimkan payload dalam bentuk JSON. Kemudian peneliti mengolah respond yang didapatkan dari backend service.

```
1  def proxy(self, payload):
2      try:
3          url = payload.get("url")
4          method = payload.get("method")
5          if method == HTTPMethods.POST.value:
6              hit_method = HTTPMethods.POST
7          if method == HTTPMethods.GET.value:
8              hit_method = HTTPMethods.GET
9
10         param = payload.get("params")
11         body = payload.get("body")
12         header = payload.get("header")
13         kwargs = {}
14         if header:
15             kwargs['headers'] = header
16
17         print(url, hit_method, kwargs)
18         if (param != ""):
19             response = http_request(url, hit_method, params = param, **kwargs)
20         if (body != ""):
21             response = http_request(url, hit_method, json = body, **kwargs)
22         return response.json(), response.status_code
23     except Exception as e:
24         raise e
```

Gambar 6. Fungsi proxy

Gambar 6 merupakan fungsi proxy pada backend services yang melakukan HTTP *request*. Pada baris 4 mengambil method yang telah dikirimkan dan menentukan hit_method yang digunakan pada HTTP *request*. Pada baris 13 digunakan untuk menyimpan seluruh header untuk melakukan HTTP *request*. pada baris 18 – 21 merupakan HTTP *request* yang dilakukan berdasarkan apakah terdapat parameter atau bodynya. Pada baris 22 merupakan hasil respond dalam bentuk JSON dan juga status codenya.

Potongan gambar kode diatas merupakan bagian penting pada API Management yang bertanggung jawab dalam memproses testing API. Salah satu keunggulan dalam menggunakan backend service untuk melakukan HTTP *request* adalah penanganan error. Backend service dapat menangani kesalahan dari HTTP *request* secara lebih rinci. Hal ini memungkinkan untuk memberikan *respond* yang lebih informatif.

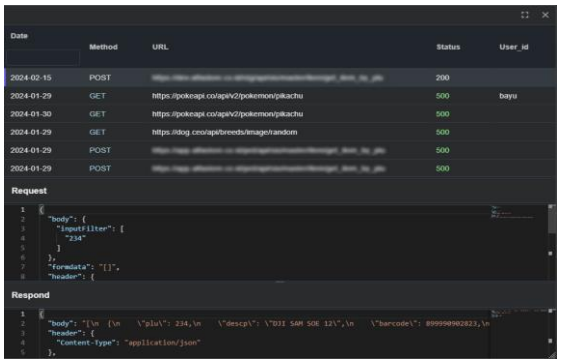



Gambar 7. Hasil *request*

Gambar 7 merupakan tampilan hasil *request* yang telah dilakukan oleh *user*, dengan mengembalikan status dan juga respon. Kemudian segala *request* yang telah dilakukan oleh *user* akan disimpan pada history untuk memudahkan *user* dalam melihat error pada APInya.

Penerapan API Management ini dapat menggantikan beberapa fungsi pada postman. Postman merupakan alat untuk mengelola API suatu Perusahaan, tetapi mungkin bukan solusi terbaik untuk keamanan karena fitur yang terbatas (Lamba, 2019). Meskipun postman menjadi salah satu teknologi yang populer dan berguna dalam pengujian API, terdapat keterbatasan banyak permintaan API dalam melakukan testing API di satu bulan. Pada postman workspace hanya terbatas untuk 3 akun saja, sedangkan dengan menerapkan API Management ini developer dapat mengatur berdasarkan kode jabatan yang ada. Hal ini

dapat menghemat administrasi pada perusahaan.




Gambar 7. History *log*

Gambar 7 merupakan tampilan fitur history log yang dapat memperlihatkan detail HTTP *request* yang dilakukan dan respondnya. Hal ini dapat membantu developer pada saat akan melakukan dokumentasi API. Data log yang ditampilkan merupakan data yang tersimpan pada cloud firestore, setiap kali sistem melakukan pengujian API.

Pengelolaan authentikasi pada database API Management ini dilakukan pada cloud firestore. Developer dapat mengatur aturan akses untuk menentukan siapa yang dapat mengubah atau membaca data yang tersimpan. Cloud Firestore digunakan dalam mengoptimalkan skalabilitas data yang tersimpan. Database ini juga dapat digunakan untuk perancangan aplikasi multi-platform. Integrasi antar layanan atau platform yang mudah menjadi solusi dalam merancang API Management. Framework webix memungkinkan untuk melakukan sinkronisasi data secara realtime tanpa menggunakan backend service. Hal ini dapat meningkatkan kinerja dalam pengelolaan API pada perusahaan.

Dengan memanfaatkan keamanan dari aplikasi SSC, dimana API Management akan diimplementasikan. Dengan menerapkan metode yang mirip dengan *rate limiting* yang berperan dalam pembatasan aksesbilitas API. *Rate Limiting* merupakan tindakan yang dilakukan untuk mengatur dan membatasi jumlah HTTP Request yang diterima oleh API yang mungkin terlalu banyak. Metode *Rate Limiting* ini sebelumnya telah diterapkan pada penelitian yang dilakukan oleh M. Ainurrahman dan Siswanto yang berjudul "Penerapan Fungsi Transforming dan Rate Limiting untuk Management API di Perusahaan". Dengan mengembangkan dan menyesuaikan kebutuhan akses API pada PT XYZ, metode *Rate Limting* diterapkan pada aplikasi SSC. Dalam keamanan yang terdapat pada aplikasi SSC, maka tiap kode jabatan merupakan parameter untuk fleksibilitas user dalam mengakses API.

API Management ini dapat membantu developer pada saat akan melakukan dokumentasi API. Fitur history yang menyimpan seluruh hasil API *testing* pada cloud firestore dapat memberikan detail dari hasil pengujian API. Mengoptimalkan dan merancang dokumentasi API yang terbaru menjadi masalah dalam OpenAPI yang terus berkembang (Wang, Tian, & He, 2023). OpenAPI adalah sebuah standar yang digunakan untuk menentukan, menggambarkan, dan mengelola API secara terbuka. Dengan sistem ini developer dapat mengoptimalkan API yang terus berkembang dengan memanfaatkan fitur history.

Rancangan API Management ini akan memudahkan perusahaan dalam mengelola API pada PT XYZ. API Management dapat membuat layanan lebih efektif dan fleksibel dalam melakukan manajemen dan pengaturan pada API. Pada penelitian yang berjudul "Implementasi Aplikasi Manajemen API Berbasis Protokol REST menggunakan Platform WSO2", disebutkan bahwa fungsi dasar dari API Management adalah keamanan, pemantauan, dan pengendalian versi. API Management dapat melakukan pemantauan hasil uji API yang dapat dilihat dari API history yang tersimpan pada database.

*Table 1 Hasil Pengujian Black Box*

| Pengujian | Yang Diharapkan | Pengamatan | Kesimpulan |
|---|---|---|---|
| Memastikan bahwa *endpoint* API mengembalikan data yang sesuai dengan *request* | *Endpoint* mengembalikan data sesuai dengan *request* | *Endpoint* mengembalikan data sesuai *request* | Valid |
| Memastikan fitur Environment | Mendapatkan data dan sinkronisa | Mendapatkan data dan sinkronisa | Valid |

| | | | |
|---|---|---|---|
| mendapatkan real-time data dari cloud firestore | si data dari cloud firestore | si data dari cloud firestore | |
| Dapat mengembalikan log detail hasil request yang dilakukan oleh user yang tersimpan pada cloud firestore | Menampilkan log detail hasil request dari cloud firestore | Menampilkan log detail hasil request dari cloud firestore | Valid |

Tabel 1 merupakan hasil pengujian *Black box* dari hasil uji menunjukkan bahwa fitur – fitur dan proses pengujian API dapat beroperasi tanpa kendala. Ketika terdapat perubahan data pada cloud firestore, data yang terdapat pada collection maupun environment akan tersinkronisasi secara langsung. Oleh karena itu API Management membuktikan keberhasilan dalam menggantikan peran postman untuk melakukan *testing* dan monitoring API. Sehingga API Management ini dapat diterapkan pada aplikasi SSC yang berfungsi untuk memonitoring setiap kesalahan yang terjadi pada transaksi di PT XYZ.

## KESIMPULAN

Berdasarkan hasil penelitian dapat disimpulkan bahwa API Management mampu mengelola API perusahaan secara fleksibel dan efisien. Pengujian API yang awalnya dilakukan pada Postman dapat dilakukan di API Management dengan mengatur authentikasi berdasarkan kode jabatan. Beberapa fitur pada API Management memudahkan dalam pengujian API, seperti environtmen yang disimpan pada realtime database sehingga apabila terdapat perubahan user tidak perlu melakukan refresh page. Fitur log history juga dapat membantu developer pada saat akan melakukan dokumentasi API dan mempermudah dalam melakukan pemantauan penggunaan API. Hal ini didukung dengan hasil pengujian Blackbox testing yang menunjukan setiap fitur dapat bekerja dengan baik.

Metode agile membantu penelitian dalam merancang API Management yang dikerjakan dengan waktu yang terbatas. Dengan membagi project kedalam beberapa fase dinamis dan melakukan review setiap kali terdapat bug maupun kesalahan pada API Management. Framework Webix memainkan peran penting dalam perancangan API Management. Dengan memanfaatkan library dan widget dari framework webix, hal ini memungkinkan perancangan tampilan yang kompleks lebih mudah.

Saran bagi penelitian selanjutnya adalah agar dapat mengembangkan API Management dengan fitur yang lebih lengkap. Seperti memungkinkan gateway yang dihosting sendiri, ip statis, dukungan virtual network, dsb. Penelitian lebih lanjut tetap diperlukan guna mengembangkan API Management sesuai dengan kebutuhan perusahaan.

## REFERENSI

Ariesta, A., Dewi, Y. N., Sariasih, F. A., & Fibriany, F. W. (2021). PENERAPAN METODE AGILE DALAM PENGEMBANGAN APPLICATION PROGRAMMING INTERFACE SYSTEM PADA PT XYZ. *Jurnal CoreIT: Jurnal Hasil Penelitian Ilmu Komputer Dan Teknologi Informasi*, *7*(1), 38. https://doi.org/10.24014/coreit.v7i1.12635

Ehsan, A., Abuhaliqa, M. A. M. E., Catal, C., & Mishra, D. (2022, May 1). RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. *Applied Sciences (Switzerland)*, Vol. 12. MDPI. https://doi.org/10.3390/app12094369

Freeman, A. (2020). Making HTTP Requests. In *Pro Angular 9* (pp. 599–623). Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-5998-6_24

Hendri, H., Hasiholan Manurung, J. W., Ferian, R. A., Hanaatmoko, W. F., & Yulianti, Y. (2020). Pengujian Black Box pada Aplikasi Sistem Informasi Pengelolaan Masjid Menggunakan Teknik Equivalence Partitions. *Jurnal Teknologi Sistem Informasi Dan*

*Aplikasi*, *3*(2), 107. https://doi.org/10.32493/jtsi.v3i2.4694

Ilham Firman Maulana. (2020). Penerapan Firebase Realtime Database pada Aplikasi E-Tilang Smartphone berbasis Mobile Android. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(5), 854–863. https://doi.org/10.29207/resti.v4i5.2232

Kedrin, V. S., & Rodyukov, A. V. (2021). System technologies for the formation of a data control contour for the personal account of an applicant based on the 1C:Enterprise 8.3 platform. *Informatics and Education*, (2), 12–23. https://doi.org/10.32517/0234-0453-2021-36-2-12-23

Lamba, A. (2019). API Design Principles & Security Best Practices – Accelerate Your Business Without Compromising Security. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3535436

Mathijssen, M., Overeem, M., & Jansen, S. (2020). *Identification of Practices and Capabilities in API Management: A Systematic Literature Review*. Retrieved from http://arxiv.org/abs/2006.10481

Melville, N., & Kohli, R. (2021). *Roadblocks to Implementing Modern Digital Infrastructure: Exploratory Study of API Deployment in Large Organizations*. https://doi.org/10.24251/HICSS.2021.723

Meng, M., Steinhardt, S. M., & Schubert, A. (2020). Optimizing API documentation: Some guidelines and effects. *SIGDOC 2020 - Proceedings of the 38th ACM International Conference on Design of Communication*. Association for Computing Machinery, Inc. https://doi.org/10.1145/3380851.3416759

Ningtyas, D. F., & Setiyawati, N. (2021). Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request. *Jurnal Janitra Informatika Dan Sistem Informasi*, *1*(1), 19–34. https://doi.org/10.25008/janitra.v1i1.120

Nugroho, C., & Susetyo, Y. A. (n.d.). IMPLEMENTATION OF WEBIX DYNAMIC SCRIPTING FOR WEB-BASED FORM APP DEVELOPMENT AS AN ALTERNATIVE TO DATA-COLLECTING APPLICATIONS. *Jurnal Teknik Informatika (JUTIF)*. https://doi.org/10.20884/1.jutif.2022.3.2.212

Riesco, R., & Villagrá, V. A. (2019). Leveraging cyber threat intelligence for a dynamic risk framework. *International Journal of Information Security*, *18*(6), 715–739. https://doi.org/10.1007/s10207-019-00433-2

Setiawan, G. H., Made, I., Adnyana, B., & Budiarta, K. (2022). *Pengujian Performa API (Application Programming Interface) dengan Metode Load Testing*. Bali.

Siriwardena, P. (2020). Edge Security with an API Gateway. In *Advanced API Security* (pp. 103–127). Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-2050-4_5

Wang, S., Tian, Y., & He, D. (2023). *Improving API Documentation Comprehensibility via Continuous Optimization and Multilingual SDK*.

Wingerath, W., Ritter, N., & Gessert, F. (2019). *Real-Time Databases*. https://doi.org/10.1007/978-3-030-10555-6_3