

IMPLEMENTASI ARSITEKTUR *MICROSERVICE* PADA APLIKASI *ONLINE TRAVEL TOURINC*

Muhamad Danil Rafiqi¹, Eko Subyantoro², Dewi Kania W.³

¹ mahasiswa, ² pembimbing 1, ³ pembimbing 2

Abstrak

PT Tourinc merupakan perusahaan yang bisnisnya bergerak di bidang penjualan tiket. Tiket yang dijual adalah tiket pesawat, hotel, tours, dan kereta api. PT. Tourinc membutuhkan aplikasi untuk menjalankan proses bisnisnya. Aplikasi yang dibutuhkan memiliki banyak kompleksitas, di antara kompleksitasnya adalah untuk penyediaan produk harus mengakses ke REST API berbagai perusahaan, diantaranya adalah akses ke PT. KAI, penyedia jasa penerbangan (Garuda, Sriwijaya, dll), berbagai hotel. Selain itu, untuk menangani proses pembayaran, harus dikembangkan aplikasi yang dapat mengotomatisasi proses konfirmasi pembayaran. Berdasarkan kebutuhan diatas, solusi yang tepat untuk pengembangan aplikasi adalah menggunakan arsitektur *microservice*. Arsitektur *microservice* dapat mengatasi kompleksitas yang besar karena, tiap fitur aplikasi akan dibagi menjadi bagian kecil. Metode pengembangan aplikasi yang digunakan adalah *Web Service Implementation Methodology* (WSIM), *modeling* dibuat dengan menggunakan *RESTful API Modeling Language* (RAML), dan bahasa pemrograman yang digunakan adalah PHP dengan framework Laravel dan javascript (node.js) dengan framework express.js.

Kata Kunci: *Microservice, REST, RAML, Web Service Implementation Methodology, node.js, laravel*

PENDAHULUAN

PT Tourinc merupakan perusahaan yang bisnisnya bergerak di bidang penjualan tiket. Tiket yang dijual di antara lain adalah tiket pesawat, hotel, dan kereta api. PT Tourinc memiliki beberapa divisi untuk mendukung pelayanan bisnis yang dijalankan, antara lain divisi *Marketing, Accounting, dan Customer Support*. Divisi *Marketing* berfokus pada pemasaran tiket ke konsumen, divisi *Accounting* berfokus pada pelaporan penjualan, dan divisi *Customer Support* sebagai divisi yang bertugas membantu pelayanan kepada konsumen dan juga membantu pelayanan bagian internal perusahaan.

PT. Tourinc membutuhkan aplikasi untuk membantu menjalankan proses bisnisnya, karena inti bisnis yang dijalankan adalah penjualan tiket secara *online*. Fitur fitur yang dibutuhkan di aplikasi Tourinc antara lain berupa service : (1) Data *Order*, (2) Kirim Email, (3) Kirim Notifikasi,

(4) Kirim SMS, (5) Manajemen data user, (6) Manajemen data karyawan, (7) Data *Guest*, (8) Otentikasi Karyawan, (9) Otentikasi User.

Pengembangan aplikasi pada saat ini menurut Nellaiyapen dan Sashidaran (2018) terdapat dua macam arsitektur, yaitu : *Monolithic* dan *Microservice*. Arsitektur *monolithic* merupakan arsitektur pengembangan aplikasi yang mana menjadikan satu antara kode program, *database*, dan tampilan program. Arsitektur *monolithic* memiliki kelemahan yaitu : (1) Performa akan menurun ketika aplikasi semakin besar, (2) Sulit untuk adaptasi teknologi baru, (3) Aplikasi semakin besar dan kompleks sehingga sulit dipahami, (4) Proses *update* akan mempengaruhi keseluruhan aplikasi, (5) Tidak dapat menggunakan bahasa pemrograman yang berbeda. Kelemahan-kelemahan pada arsitektur *monolithic* dapat ditangani dengan menerapkan arsitektur *microservice*. Arsitektur *microservice* merupakan

arsitektur pengembangan aplikasi dengan cara membuat aplikasi berupa *service* terkecil. *Service* tersebut berjalan masing-masing sesuai dengan fungsinya. Kumpulan dari *service* ini akan saling berkomunikasi sehingga menjadi satu kesatuan dan menjadi sistem yang besar. Kelebihan arsitektur *microservice* antara lain adalah : (1) Memiliki kompleksitas yang kecil, (2) Dapat mengembangkan aplikasi multi-platform, (3) Setiap *service* dapat berdiri sendiri, (4) Proses *update* hanya terjadi pada *service* yang ingin dilakukan *update* saja.

PT Tourinc memiliki banyak kompleksitas pada pengembangan aplikasinya. Pengembangan aplikasi untuk penyediaan produk harus terhubung kepada banyak perusahaan. Ketika ingin menyediakan produk Tiket Pesawat maka PT. Tourinc harus mengakses ke REST API berbagai perusahaan penyedia jasa penerbangan. Untuk menyediakan Reservasi Hotel, PT. Tourinc harus mengakses ke banyak REST API hotel. Tidak hanya itu saja, untuk menangani proses pembayaran dan konfirmasi pembayaran, PT Tourinc harus mengembangkan aplikasi yang dapat mengotomatisasi proses konfirmasi agar tidak menghambat proses bisnis.

Berdasarkan kebutuhan diatas, solusi yang tepat untuk pengembangan aplikasi PT. Tourinc adalah dengan menggunakan arsitektur *microservice*. Arsitektur *microservice* dapat mengatasi kompleksitas yang cukup besar, karena tiap tiap kebutuhan pada PT. Tourinc akan dibagi menjadi bagian bagian kecil, selain itu dapat mengembangkan aplikasi lintas platform, dan dapat menggunakan bahasa pemrograman sesuai kebutuhan *service* yang diinginkan.

Tinjauan Pustaka

1. Penelitian Terdahulu

Aliyah dkk (2018), dalam jurnalnya yang berjudul “Implementasi *Web Service* Dalam Monitoring Pendapatan Perusahaan Dari Penjualan Tiket Bus Di Perum Damri Kantor Cabang Bandar Lampung Berbasis Web”. Tujuan dari penelitian ini adalah mempermudah monitoring pendapatan dari penjualan tiket bus secara *realtime* dengan implementasi *web service*.

Putri dkk (2018), dalam jurnalnya yang berjudul “Simulasi Pembayaran Sumbangan Penyelenggaraan Pendidikan (SPP) Pada SDIT Permata Bunda Dengan *Web Service*”. Tujuan dari penelitian ini adalah menghasilkan aplikasi pembayaran spp dengan implementasi *web service*.

Pradyawati dkk (2018), dalam jurnalnya yang berjudul “Implemetasi *Web Service* Untuk Itegrasi Data Akademik Pada Proses Pengolahan Data Penjadwalan Asisten Laboratrium Abcd”. Tujuan dari penelitian tersebut adalah mengintegrasikan *multi database* dengan teknologi *web service* pada proses pengolahan data penjadwalan asisten laboratorium.

Mufrizal & Indarti (2019), dalam jurnalnya yang berjudul “*Refactoring* Arsitektur *Microservice* Pada Aplikasi Absensi PT.Graha Usaha Teknik”. Tujuan dari penelitian ini adalah merubah sistem lama dengan arsitektur monolitik menjadi sistem baru dengan arsitektur *microservice*. Aplikasi dikembangkan dengan menggunakan Docker sebagai teknologi *Container*, dan GitLab untuk membantu proses *Deployment*.

Sendiang dkk (2018), dalam jurnalnya yang berjudul “Implementasi Teknologi *Microservice* Pada Pengembangan *Mobile Learning*”. Tujuan

dari penelitian ini adalah menghasilkan aplikasi dengan implementasi *Microservice*. Pengembangan menggunakan UML sebagai pemodelan sistem, selain itu menggunakan REST *Web Service* untuk membuat tiap tiap servicenya.

Metodologi Pelaksanaan

Metode yang digunakan dalam pengembangan *web service* pada Aplikasi *Online Travel Tourinc* adalah dengan menggunakan metode *Web Service Implementation Methodology* (WSIM). Tahapan pengembangan sistem aplikasi, dimulai dengan *requirements* hingga tahap *deployment*.

1. Requirements

Tahap *requirements* dilakukan wawancara kepada *project manager* PT. Inspire Technology. Wawancara digunakan untuk mengidentifikasi kebutuhan bisnis pada aplikasi *online travel*.

2. Analysis

Pada *analysis* hal hal yang dilakukan adalah menentukan arsitektur dari *web service* yang akan dibangun, memilih platform yang akan digunakan, memilih teknologi untuk hosting *web service*, dan memilih bahasa pemrogramman untuk implementasi.

3. Design

Tahap *design* adalah tahap yang dilakukan dengan cara membuat desain URI, membuat *modelling* REST API, dan membuat ERD.

4. Coding

Tahap *coding* merupakan hasil dari tahap *design* dikembangkan menjadi *web service*. *Web service* dibuat dengan aplikasi *Visual Studio Code*, LAMPP, *framework* laravel dan *framework* express.js.

5. Test

Tahap *test* merupakan tahap dilakukan pengujian *web service* menggunakan *software* Insomnia. Tes dilakukan menggunakan metode *boundary value analysis* dengan menguji batas atas dan batas bawah sehingga dapat mengetahui apakah *web service* dapat bekerja dengan baik atau tidak.

6. Deployment

Tahap *deployment* merupakan tahapan untuk mengkonfigurasi dan melakukan hosting hasil *coding* kedalam *server*.

Hasil dan Pembahasan

1. Requirements

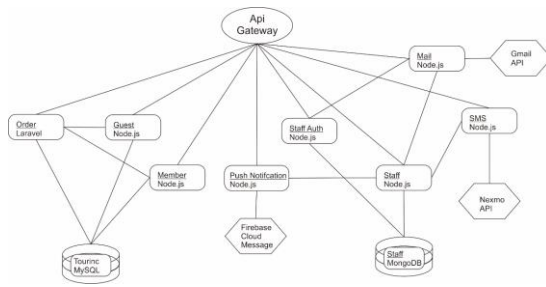
Berdasarkan hasil wawancara yang telah dilakukan dengan *Project Manager* PT. Tourinc, dapat disimpulkan bahwa untuk membangun aplikasi diperlukan beberapa *service*, yaitu : (1) *Data Order*, (2) Kirim Email, (3) Kirim Notifikasi, (4) Kirim SMS, (5) Manajemen data user, (6) Manajemen data karyawan, (7) *Data Guest*, (8) Otentikasi Karyawan.

2. Analysis

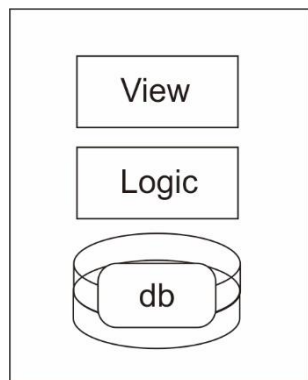
Tahap analisis menentukan mengenai arsitektur dari *microservice* yang akan dibangun, menentukan bahasa pemrogramman, menentukan jenis *database*, menentukan interaksi antar *service*, menentukan kandidat *service*, dan menentukan *web service interface*.

a. Arsitektur *Microservice*

Gambaran umum dari arsitektur *microservice* yang akan dibangun untuk *online travel tourinc* dapat dilihat pada gambar 1 dan perbandingan jika dibangun dengan arsitektur *monolith* pada gambar 2.



Gambar 1. Arsitektur microservice *Tourinc*



Gambar 2. Arsitektur *monolith*

Gambar diatas menjelaskan mengenai arsitektur *microservice* dibangun dan perbandingan jika dibangun dengan arsitektur *monolith*. Gambaran diatas berupa service yang dibuat, service yang saling berhubungan, database yang digunakan, bahasa pemrogramman yang dipakai, dan service pihak ketiganya yang digunakan. Tabel 1 menjelaskan mengenai arti dari simbol pada gambar arsitektur *microservice*.

b. Kandidat Service

Kandidat *service* merupakan objek yang akan dijadikan *service*. Kandidat *service* ditentukan berdasar pada kebutuhan fungsional dari arsitektur *microservice*. Berdasarkan hasil pada tahap *requirements* maka didapat kandidat *service* seperti pada tabel 3 berikut ini.

Tabel 1 Kandidat *Service*

Nama <i>Service</i>	Keterangan
(1)	(2)
SMS	Digunakan untuk mengirim sms.
Notification	Digunakan untuk mengirim push notification
Mail	Digunakan untuk mengirim email.
Order	Digunakan menampilkan data order
Guest	Digunakan untuk mengelola data guest
Member	Digunakan untuk mengelola member
Staff	Digunakan untuk mengelola staff
Staff Auth	Digunakan untuk otentikasi staff

c. Identifikasi Service Interface

Service interface berfungsi sebagai antarmuka dari *service* yang akan berinteraksi dengan aplikasi klien. Teknologi *Service* yang digunakan adalah menggunakan REST API. Tabel 2 akan menjelaskan tentang identifikasi *service interface* pada *microservice* *tourinc*.

Tabel 2 Identifikasi *Service Interface*

Nama <i>Service</i>	<i>Method</i>
(1)	(2)
SMS	POST
Notification	POST
Mail	POST
Order	GET
Guest	GET, POST
Member	GET, POST, PUT, DELETE
Staff	GET, POST, PUT, DELETE
Staff Auth	POST

3. *Design*

Tahapn *design* dilakukan dengan cara mendefinisikan *design uri* (*unified resource indentifier*), rancangan *database*, *Entity Relational Diagram* (ERD), dan *modeling* REST

API menggunakan *Restful Api Modeling Language* (RAML).

a. *Design Uniform Resource Identifier* (URI)

Design URI dimaksudkan agar setiap *service* memiliki *uri* yang unik. Penamaan *uri* yaitu berdasarkan pada nama *service* dan *parameter* agar memudahkan aplikasi klien dalam mengakses *service*. Nama *service* dibuat berdasarkan dari *table* dalam *database*, dan nama *parameter* diambil dari *field* di dalam *database*. Tabel 5 menjelaskan mengenai *design uri microservice*.

Tabel 3 *Design Uniform Resource Identifier* (URI)

<i>Service</i>	URI	<i>Method</i>
(1)	(2)	(3)
SMS	http://tourinc.id/v1/sms	POST
Notifika tion	http://tourinc.id/v1/push-notif	POST
Mail	http://tourinc.id/v1/mail	POST
Order	http://tourinc.id/v1/order	GET
	http://tourinc.id/v1/order/{id}	GET
	http://tourinc.id/v1/order/by-date/{start}/{end}/{type}	GET
Guest	http://tourinc.id/v1/guest	GET,POST
	http://tourinc.id/v1/guest/{id}	GET
	http://tourinc.id/v1/guest/by-date/{start}/{end}	GET
Membe r	http://tourinc.id/v1/member	GET,POST
	http://tourinc.id/v1/user/{id}	GET,PUT,D ELETE GET
	http://tourinc.id/v1/user/by-date/{start}/{end}/{type}	

Staff	http://tourinc.id/v1/staff/	GET,POST
	http://tourinc.id/v1/staff/{id}	GET,PUT,D ELETE
Staff Auth	http://tourinc.id/v1/staff-auth/signin	POST
	http://tourinc.id/v1/staff-auth/signup	POST
	http://tourinc.id/v1/staff-auth/reset	POST
	http://tourinc.id/v1/staff-auth/save-token	POST
	http://tourinc.id/v1/staff-auth/{mail}/token	-

b. Rancangan *Database*

Database pada *microservice* tourinc menggunakan *database* MySQL dan MongoDB. Perancangan desain *database* MySQL dibuat dengan menggunakan PHPMyAdmin, sedangkan *database* MongoDB dibuat dengan menggunakan Visual Studio Code dan hasil desain dapat dilihat pada situs <https://mlab.com>.

c. *RESTful API Modeling Language* (RAML)

Desain REST API dibuat menggunakan RAML yang akan menghasilkan beberapa informasi, yaitu : (1) Informasi dasar, (2) *Security*, (3) *Data Types*, (4) *Resource*, dan (5) *Method*. Hasil *modeling* REST API menggunakan RAML dapat dilihat dibawah ini :

```

#%RAML 1.0
title: Tourinc API
version: v1
protocols: [ HTTPS ]
baseUri: http://tourinc.id/{version}
mediaType: application/json

securitySchemes:
  basicAuth:
    description: Each request need data header
    type: Basic Authentication
    
```

```

describedBy:
  headers:
    Authorization:
      description: Used to send the jwt
"username:password" credentials
      type: string
    responses:
      401:
        description: |Unauthorized, Token Not Valid.
types:
Sms: !include sms.type.json
/sms:
  post:
    description: Send SMS
    body:
      application/json:
        type : Sms
        example : !include sms.example.req.json
    responses:
      200:
        body:
          application/json:
            type : Sms
            example : !include sms.example.res.json
    
```

Gambar 3 Sms.raml

```

types:
PushNotif: !include push-notification.type.json
/push-notif:
  post:
    description: Send Push Notification
    body:
      application/json:
        type : PushNotif
        example : !include push-
notification.example.req.json
    responses:
      201:
        body:
          application/json:
            type : PushNotif
            example : !include push-
notification.example.res.json
    
```

Gambar 4. Push-Notif.raml

```

Mail: !include mail.type.json
/mail:
  post:
    description: Send Mail
    body:
      application/json:
        type : Mail
        example : !include mail.example.req.json
    responses:
      201:
        body:
          application/json:
            type : Mail
    
```

```
example : !include mail.example.res.json
```

Gambar 5. Mail.raml

```

types:
Order: !include order.type.json
/order:
  get:
    description: Get All Order
    responses:
      200:
        body:
          application/json:
            type : Order
            example : !include
order.example.multiple.json
    /{uid}:
      get:
        description: Get Order Detail
        responses:
          200:
            body:
              application/json:
                type : Order
                example : !include
order.example.single.json
    /by-date:
      /{start}/{end}/{type}:
        get:
          description: Get Data Order By Date
          responses:
            200:
              body:
                application/json:
                  example : !include
order.example.multiple.json
    
```

Gambar 6. Order.raml

```

types:
Guest: !include guest.type.json
/guest:
  get:
    description: Get All Guest
    responses:
      200:
        body:
          application/json:
            type : Guest
            example : !include
guest.example.res.multiple.json
    post:
      description: create guest
      body:
        application/json:
          type : Guest
          example : !include guest.example.req.json
    responses:
      201:
        body:
    
```

```

    application/json:
      type : Staff
      example           :           !include
guest.example.res.single.json
/{uid}:
get:
  description: Get Guest Detail
responses:
  200:
    body:
      application/json:
        type : Guest
        example           :           !include
guest.example.res.single.json
/by-date/{start}{end}:
get:
  description: Get Guest Detail
responses:
  200:
    body:
      application/json:
        type : Guest
        example           :           !include
guest.example.res.multiple.json
    
```

Gambar 7. Guest.raml

```

types:
  Member: !include member.type.json
/member:
get:
  description: Get All Member
responses:
  200:
    body:
      application/json:
        type : Member
        example           :           !include
member.multiple.example.json
/{uid}:
get:
  description: Get Member Detail
responses:
  200:
    body:
      application/json:
        type : Member
        example           :           !include
member.single.example.json
put:
  description: Update Member
  body:
    application/json:
      type : Member
      example           :           !include
member.single.example.json
responses:
  200:
    body:
      application/json:
    
```

```

      type: Member
      example:           !include
member.single.example.json
delete:
  description: delete Staff
responses:
  200:
    body:
      application/json:
        example : {"message":"success delete
data"}

put:
  description: Activate Member
responses:
  200:
    body:
      application/json:
        type: Member
        example:           !include
member.single.example.json
/suspend:
put:
  description: Suspend Member
responses:
  200:
    body:
      application/json:
        type: Member
        example:           !include
member.single.example.json
/unsuspend:
put:
  description: Unsuspend Member
responses:
  200:
    body:
      application/json:
        type: Member
        example:           !include
member.single.example.json
/{email}:
/token:
get:
  description: Get Token
responses:
  200:
    body:
      application/json:
        type : Member
        example           :           !include
member.single.example.json
/save-token:
post:
  description: Save Token
  body:
    application/json:
      type : Member
      example           :           {"token-
notification":"skjhfkjhioeinjkiweuewi"}
    
```

```

responses:
  201:
    body:
      application/json:
        type : Sms
        example : { "status": "succes", "code": 201,
"message": "success send sms", "data": { "token-
notification": "skjhfkjhioeinjkiweuwei" } }

```

Gambar 8. Member.raml

```

types:
  Staff: !include staff.type.json
/staff:
  get:
    description: get all staff
    responses:
      200:
        body:
          application/json:
            type: Staff
            example: !include staffs.example.json
  /{id}:
    get:
      description: get detail staff
      responses:
        200:
          body:
            application/json:
              type: Staff
              example: !include staff.example.json
    put:
      description: Update Staff
      body:
        application/json:
          type : Staff
          example : !include staff.example.json
      responses:
        200:
          body:
            application/json:
              type: Staff
              example: !include staff.example.json
    delete:
      description: delete Staff
      responses:
        200:
          body:
            application/json:
              example : {"message": "success delete
data"}

```

Gambar 9. Staff.raml

```

/staff-auth:
  /signin:
    post:
      description: Sign In
      body:
        application/json:
          type : !include staff-auth.type.json
          example : !include staff-auth.example.json
      responses:
        201:
          body:
            application/json:
              type : Staff
              example : !include staff-
auth.result.example.json
  /signup:
    post:
      description: Sign Up
      body:
        application/json:
          type : !include staff-auth.type.json
          example : !include staff-auth.example.json
      responses:
        201:
          body:
            application/json:
              type : Staff
              example : !include staff-
auth.result.example.json

```

Gambar 10. StaffAuth.raml

4. Coding

Coding merupakan tahap implementasi dari tahap sebelumnya ke dalam bahasa yang dikenali oleh komputer. Tahap *coding* dilakukan dengan membuat *service* menggunakan *framework* laravel dan *framework* express. *Coding* dimulai dengan membuat *service* sampai dengan membuat *api gateway*.

5. Test

Tahap *test* dalam menyelesaikan *microservice online travel* tourinc adalah dengan melakukan *black box testing* menggunakan aplikasi Insomnia dan metode *boundary value analysis*. *Test* dilakukan untuk menguji seluruh *service* dengan menjalankan *method http* yang

tersedia di tiap *service* dan menguji batas atas dan batas bawah pada parameter yang dibutuhkan.

Kesimpulan dan Saran

1. Kesimpulan

Kesimpulan yang dapat diambil dari Tugas Akhir yang berjudul “Implementasi Arsitektur *Microservice* pada Aplikasi *Online Travel Tourinc*” adalah telah dihasilkannya *web service* berupa *Represtational State Transfer (REST)* untuk pengembangan aplikasi dengan arsitektur *microservice* pada *Online Travel Tourinc*.


2. Saran

Berdasarkan kesimpulan yang telah diuraikan, maka saran untuk pengembangan *microservice* pada *online travel tourinc* selanjutnya adalah (1) Menambah *service* yang ada dari arsitektur *microservice* yang sudah dibuat sebelumnya (2) Menambah fitur *cache* pada pengembangan selanjutnya agar dapat mempercepat pengaksesan data

REFERENSI

- A.S, R., & Shalahudin, M. (2018). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika Bandung.
- Balachandar, B. M. (2017). *RESTful Java Web Services*. Birmingham: Packt Publishing Ltd.
- Chan, L. P., Ang, C. H., Tan, P. S., Lee, H. B., Cheng, Y., Xu, X., & Yin, Z. (2005). Web Service Implementation Methodology. *OASIS*, 1-35.
- Cholifah, W. N., Yulianingsih, & Sagita, S. M. (2018). Pengujian Black Box Testing Pada Aplikasi Action & Strategy Berbasis Android. *Jurnal String*, 206-210.
- Gilchrist, A. (2015). *REST API Design and Control for DevOps*. Nonthaburi: RG Consulting.
- Hendriyanti, L. (2019). Pengaruh Online Travel Agent Terhadap Pemesanan Kamar di Hotel Mutiara Malioboro Yogyakarta. *Jurnal Media Wisata*, 1-10.
- Mitchell, L. J. (2016). *PHP Web Services*. Sebastopol: O'Reilly Media Inc.
- Sashidaran, D. K., & Nellaiyapen, S. K. (2018). *Fullstack Development With JHipster*. Birmingham: Packt Publishing Ltd.
- Sharma, S., RV, R., & Gonzales, D. (2016). *Microservices: Building Scalable Software*. Birmingham: Packt Publishing Ltd.
- Subramanian, H., & Raj, P. (2019). *Hands-On RESTful API Design Patterns And Best Practices*. Birmingham: Packt Publishing Ltd.
- Yudhanto, Y., & Prasetyo, H. A. (2018). *Panduan Mudah Belajar Framework Laravel*. Jakarta: PT. Elex Media Komputindo.
- Zaman, G. A. (2017). Perancangan dan Implementasi Web Service Sebagai Media Pertukaran Data Pada Aplikasi Permainan. *Jurnal Informatika*, 22-30.


Karya Ilmiah


16753035 Muhamad Danil R: 
8 menit yang lalu


7%

Risiko dari plagiarisme
MEDIUM

Parafrase 1%
Kutipan salah 0%
Concentration 

 Bagikan

 Deep \$ 1.00

 Monetize

 View report \$ 1.88


Tugas Akhir


TA Terbaru - FIX.docx 
2 minggu yang lalu


2%

Risiko dari plagiarisme
MEDIUM

Parafrase 0%
Kutipan salah 0%
Concentration 

 Bagikan

 Deep \$ 1.00

 Monetize

 View report \$ 1.73