BAB IV
HASIL DAN PEMBAHASAN

Secara mendalam bab ini akan membahas tahap hasil dan pembahasan yang mengacu pada Gambar 2.1 yaitu, pengkodean, pengujian, dan penyebaran. Hasil akan berisi penjelasan secara detail yang didapat setelah melakukan desain pada bab sebelumnya yang dimulai dari pembahasan dibangunnya sistem pada tahap pengkodean, setelah itu pengujian, dan terakhir penyebaran berupa katalog. Pembahasan akan membahas mengenai perilaku aplikasi secara spesifik dan keseluruhan yang secara detail akan terlampir pada sub bab pembahasan.

4.1 Hasil

Terdapat tiga macam tahapan hasil yang akan dilalui yaitu, pengkodean yang berisi potongan kode dari dibangunnya sebuah REST API, pengujian yang berisi tampilan dari diujinya fungsi menggunakan aplikasi Postman, dan penyebaran yang menampilkan antarmuka dari katalog fungsi REST API UTC. Dengan menggunakan bahasa pemrograman PHP dan *framework* Laravel dalam pembangunan sistem, dihasilkan REST API UTC sesuai dengan kebutuhan pengembang platform lain.

4.1.1 Pengkodean

Pada dasarnya tahap ini sama seperti tahap pengembangan aplikasi pada umumnya. Satu perbedaan sederhana terletak pada penciptaan antarmuka. *Web service* tidak memiliki *Graphical User Interface (GUI)* seperti aplikasi pada umumnya. Pengkodean REST API UTC dimulai dari pembuatan *controller* yang berperan untuk pengolah data yang berfungsi untuk ambil data atau tambah data.

Trainings Controller pada Gambar 4.1 memuat kode untuk menampilkan data secara keseluruhan dan data berdasar *id*. Pada saat melakukan uji pada *method* GET *all* dan GET *id* maka keluaran akan tampak seperti pada Tabel 3.1.

```
<?php

namespace App\Http\Controllers;

use App\Training;
use Illuminate\Http\Request;
use App\Http\Resources\TrainingResource;
```

```
class TrainingController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $trainings = Training::paginate(10);

        return TrainingResource::collection($trainings);
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $training = Training::findOrFail($id);

        return new TrainingResource($training);
    }
}
```

Gambar 4.1 *Training Controller*

Pada Gambar 4.2 memuat tampilan kode dari *Users Controller* yang memiliki fungsi untuk menampilkan daftar peserta berdasar *e-mail* dan berdasar *id*. Dalam menampilkan daftar maka dilakukan pengujian pada *method* GET dengan keluaran seperti pada Tabel 3.2.

Function createUser berfungsi untuk menyimpan data pengguna yang telah berhasil masuk. Ketika melakukan pengujian perlu memasukkan *key* sesuai dengan yang ada pada basis data. Masukan serta keluaran yang terdapat pada *method* POST dapat dilihat pada Tabel 3.3.

Pengguna dapat melakukan perubahan informasi data diri pada fitur akun. Untuk itu diperlukan fungsi seperti yang tertera pada potongan kode *function updatebyid*. Perlu melakukan masukan dan kemudian akan mendapat keluaran seperti pada Tabel 3.4.

```
<?php

namespace App\Http\Controllers;

use App\Users;
use Illuminate\Http\Request;
use App\Http\Resources\UsersResource;
```

```
class UsersController extends Controller
{
    public function index()
    {
        $users = Users::paginate(10);

        return UsersResource::collection($users);
    }

    public function show($id)
    {
        $users = Users::findOrFail($id);

        return new UsersResource($users);
    }

    public function cek(Request $request,$email )
    {
        $user = Users::where('email',$email)->first();
        if ($user) {
            $res['success'] = true;
            $res['message'] = $user;
        } else {
            $res['success'] = false;
            $res['message'] = 'Cant find user!';
        }
        return response($user);

    }

    public function createUser(Request $request){
        $users = new Users();

        $users->email = $request->input('email');
        $users->google_id = $request->input('google_id');
        $users->name = $request->input('name');
        $users->avatar = $request->input('avatar');

        $users->save();
        return response()->json($users);
    }

    public function updatebyid(Request $request,$id) {
        $users = Users::find($id);
        $users->email = $request->input('email');
        $users->google_id = $request->input('google_id');
        $users->name = $request->input('name');
        $users->avatar = $request->input('avatar');
        $users->phone = $request->input('phone');
        $users->address = $request->input('address');
        $users->institusi = $request->input('institusi');
        $users->pekerjaan = $request->input('pekerjaan');

        $users->save();
        return response()->json($users);
    }
}
```

Gambar 4.2 *Users Controller*

Participants Controller dapat dilihat pada Gambar 4.3. Tampilan kode menjelaskan bahwa ketika fungsi melalui tahap uji *method* POST maka harus mengisi masukan berupa *user_id*, *training_id*, dan *is_paid* berdasarkan kolom pada basis data. Isian serta keluaran pada *method* POST dapat dilihat pada Tabel 3.6. Pengujian juga akan dilakukan pada *method* GET, potongan kode *public function getTrainingBy Use ($user_id)* merupakan kode untuk menampilkan keluaran berdasarkan tabel yang berada di basis data. Keluaran akan tampak seperti pada Tabel 3.5.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Participants;
use DB;

class ParticipantsController extends Controller
{
    public function create(Request $request){
        $participants = new Participants();

        $invoice_id = date('ym').rand(1000,9999);

        $participants->user_id = $request->input('user_id');
        $participants->training_id = $request->input('training_id');
        $participants->is_paid = $request->input('is_paid');
        $participants->invoice_id = $invoice_id;

        $participants->save();
        return response()->json($participants);
    }

    public function getTrainingByUser($user_id)
    {
        $participants= DB::table('participants')
        ->join('users','participants.user_id','=','users.id')
        -
>join('trainings','trainings.id','=','participants.training_id')
        -
>RightJoin('training_images','training_images.training_id','=','trainin
gs.id')
        ->where('participants.user_id','=',$user_id)->get();

        //return response($participants);
        return response()->json($participants);
    }
}
```

Gambar 4.3 *Participants Controller*

Potongan kode model pada *Training Model* berfungsi untuk menyambungkan organisasi serta gambar pelatihan dengan basis data. Detail kode dapat dilihat pada Gambar 4.4.

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Training extends Model
{
    public function organization()
    {
        return $this->belongsTo('App\Organization');
    }

    public function training_images()
    {
        return $this->hasOne('App\Training_images');

    }
}
```

Gambar 4.4 *Training Model*

Users Model pada Gambar 4.5 memuat potongan kode untuk menyambungkan dengan basis data pada tabel *users* dan berfungsi untuk dapat menampilkan sesuai dengan apa yang ada pada tabel *users* tersebut.

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Users extends Model
{
    protected $table = 'users';
    protected $fillable = ['email','google_id','name','avatar', 'phone',
'address', 'institusi', 'pekerjaan'];
    public $timestamps = false;

}
```

Gambar 4.5 *Users Model*

Langkah selanjutnya pembuatan *model* untuk menghubungkan dengan basis data. *Participants Model* dapat dilihat pada Gambar 4.6 yang berfungsi untuk mengambil data yang dibutuhkan untuk ditampilkan dari basis data.

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Participants extends Model
{
    protected $table = 'participants';
    protected $fillable =
['user_id','training_id','is_paid','invoice_id'];
    public $timestamps = false;

    public function training()
    {
        return $this->belongsTo('App\Training');
    }

    public function users(){
        return $this->belongsTo('App\Users');
    }

    public function training_images()
    {
        return $this->hasOne('App\Training_images');

    }
}
```

Gambar 4.6 *Participants Model*

Keseluruhan pembuatan REST API UTC menghasilkan delapan fungsi yang terdiri dari lima *method* GET, dua *method* POST, dan satu *method* PUT. Proses pemanggilan fungsi dapat dilihat pada Gambar 4.7.

```
Route::get('/trainings', 'TrainingController@index');
Route::get('/trainings/{id}', 'TrainingController@show');
Route::get('/users', 'UsersController@index');
Route::get('/users/email/{email}', 'UsersController@show');
Route::get('/participants/{user_id}',
'ParticipantsController@getTrainingByUser');
Route::post('/users', 'UsersController@createUser');
```

```
Route::post('/participants','ParticipantsController@create');
Route::put('/usersupdate/{id}', 'UsersController@updatebyid');
```

Gambar 4.7 Pemanggilan Fungsi REST API

4.1.2 Pengujian

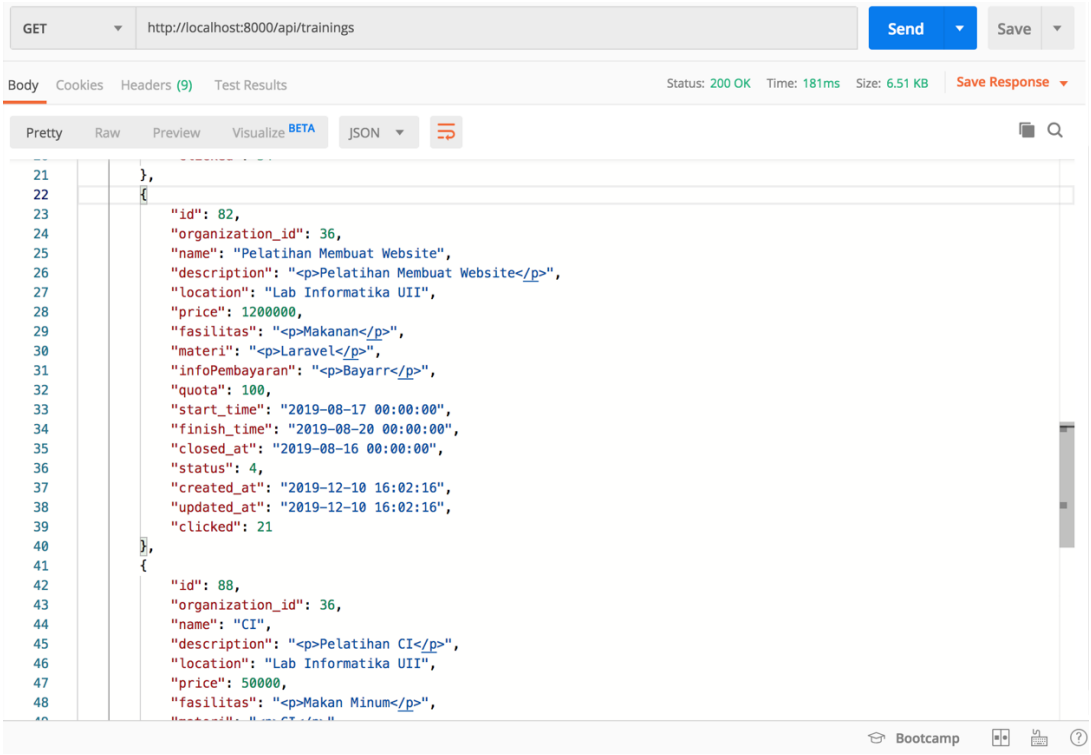
Pengujian melibatkan pelaksanaan rangkaian tugas dan membandingkan hasil dengan *output* yang diharapkan. Menerapkan tes fungsional di tahap awal pengembangan berguna untuk mempercepat pengembangan, meningkatkan kualitas, dan mengurangi risiko pada tahap akhir (AppPerfectCorporation, n.d.). Pengujian menggunakan aplikasi Postman dengan memasukkan tiga *method* yang dibangun yaitu GET, POST, dan PUT. Dalam upaya memastikan bahwa sistem yang dikembangkan telah sesuai dengan kebutuhan, diperlukan sebuah skenario pengujian. Daftar pengujian dapat dilihat pada Tabel 4.1.

Tabel 4.1 Pengujian Fungsi dengan Postman

No	Method	URI	Key	Status
1	GET	/trainings		200 OK
2	GET	/trainings/{id}		200 OK
3	GET	/users		200 OK
4	GET	/users/email/{email}		200 OK
5	GET	/participants/{user_id}		200 OK
6	POST	/users	- email - google_id - name - avatar	200 OK
7	POST	/participants	- user_id - trainings_id - is_paid	200 OK
8	PUT	/usersupdate/{id}	- name - google_id - name - avatar - phone - address - institusi - pekerjaan	200 OK

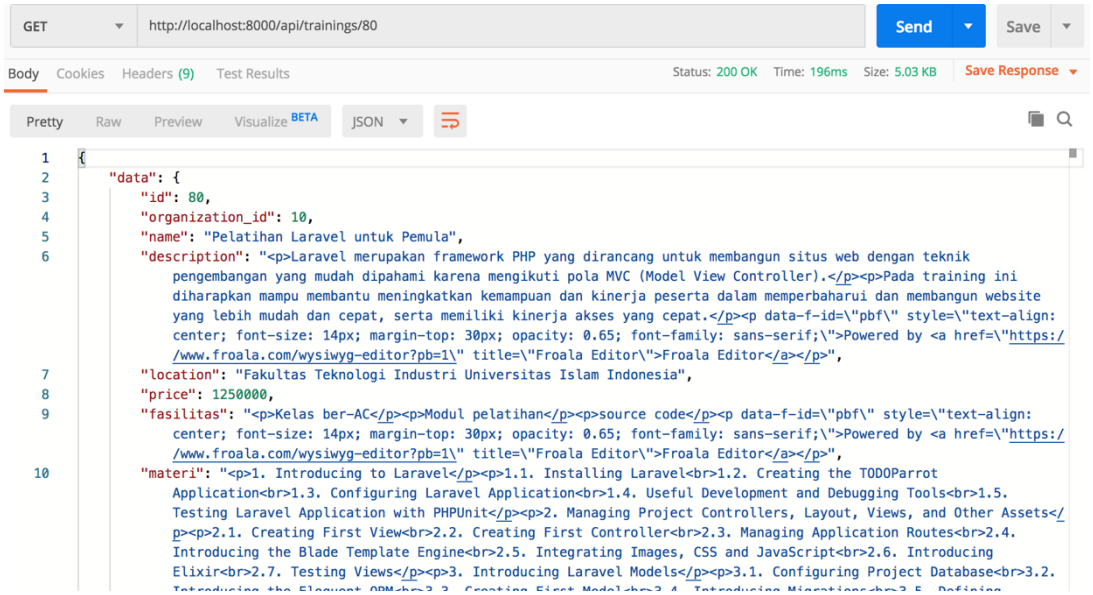
Dapat dilihat pada Gambar 4.8 tampilan dari pengujian GET *trainings* dengan fungsi menampilkan seluruh pelatihan yang tersedia. Pengujian menggunakan Postman berhasil

dilakukan dengan status kode 200 OK yang menunjukkan bahwa fungsi berhasil diuji sesuai dengan daftar pengujian pada Tabel 4.1. Keluaran yang dihasilkan menampilkan data yang sama dengan yang telah dirancang pada Tabel 3.1.



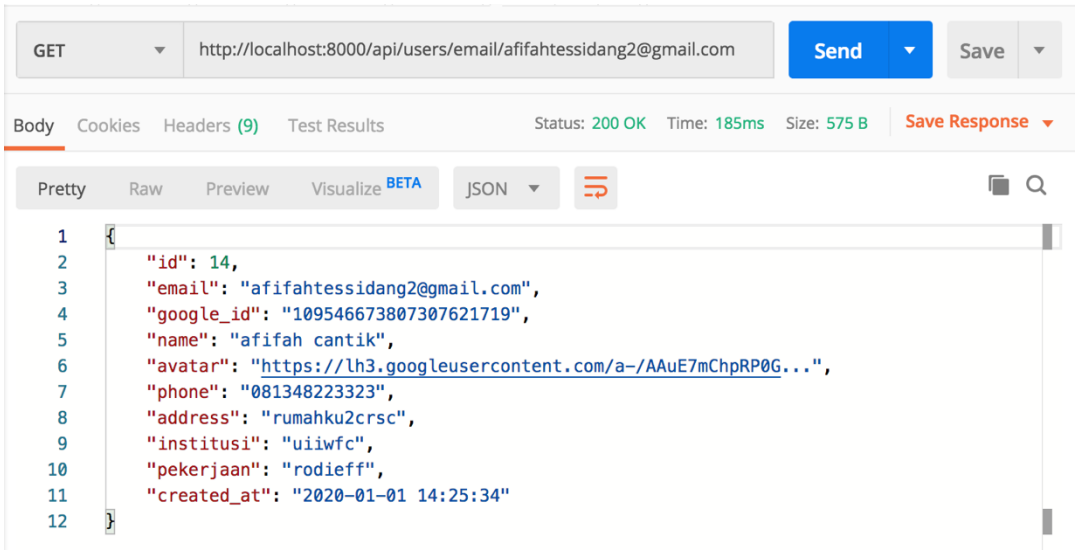
Gambar 4.8 Pengujian GET */trainings* dengan Postman

Gambar 4.9 menampilkan hasil dari pengujian menggunakan Postman dengan status kode 200 OK sesuai dengan daftar pengujian pada Tabel 4.1 yang menunjukkan bahwa fungsi berhasil diuji. GET *trainings id* dengan fungsi menampilkan pelatihan berdasar *id* memiliki keluaran yang sesuai pada Tabel 3.1.



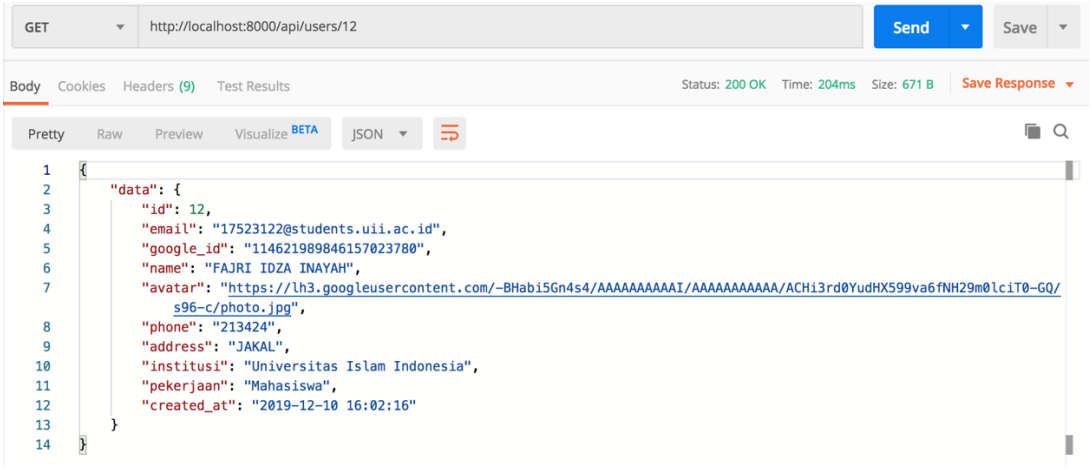
Gambar 4.9 Pengujian GET */trainings/{id}* dengan Postman

Tampilan hasil yang didapatkan pada GET *users* berdasar *e-mail* dapat dilihat pada Gambar 4.10 dengan status pengujian 200 OK sesuai dengan yang telah disajikan pada Tabel 4.1. Dengan fungsi menampilkan data peserta secara keseluruhan, keluaran yang dihasilkan pada pengujian ini sesuai dengan yang telah dirancang pada Tabel 3.2.



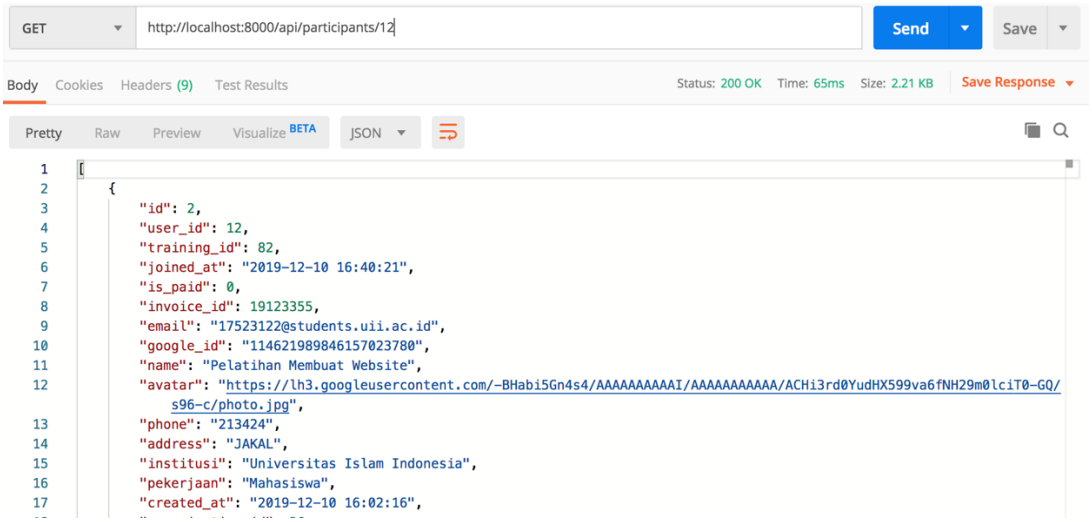
Gambar 4.10 Pengujian GET */users* dengan Postman

GET *users id* berfungsi untuk menampilkan data peserta berdasar *id* dapat dilihat pada Gambar 4.11 dengan status pengujian 200 OK sesuai dengan yang tercatat pada Tabel 4.1. Keluaran yang dihasilkan pada pengujian sama dengan yang telah dirancang pada Tabel 3.2.



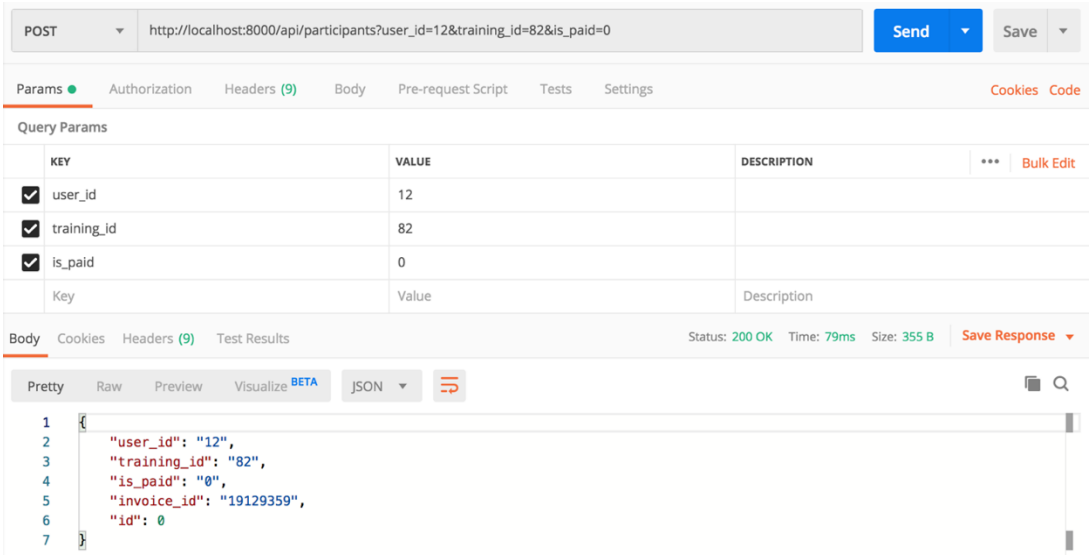
Gambar 4.11 Pengujian GET */users/{id}* dengan Postman

Method GET *participants* pada Gambar 4.12 memiliki hasil uji yang sukses dengan status 200 OK sesuai pada Tabel 4.1. GET *participants* yang berfungsi menampilkan data peserta memiliki keluaran yang dapat dilihat pada tampilan dengan detail seperti pada Tabel 3.5.



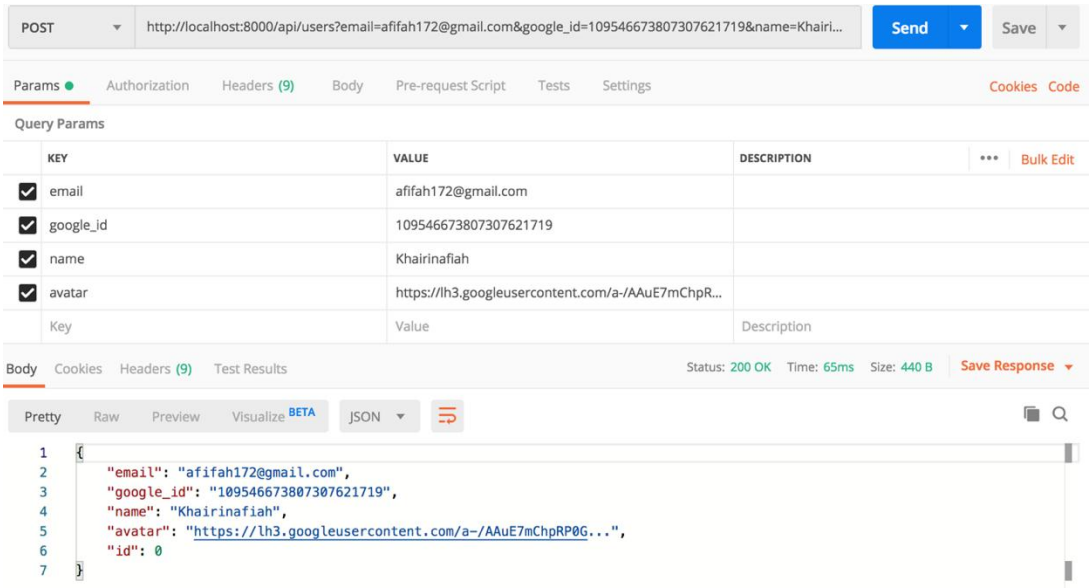
Gambar 4.12 Pengujian GET */participants/{user_id}* dengan Postman

Gambar 4.13 POST *participants* berfungsi menyimpan data peserta ketika berhasil melakukan pendaftaran pelatihan telah berhasil diuji dengan Postman dengan status 200 OK sesuai dengan Tabel 4.1. Pengisian *key* pada pengujian harus sama dengan basis data.



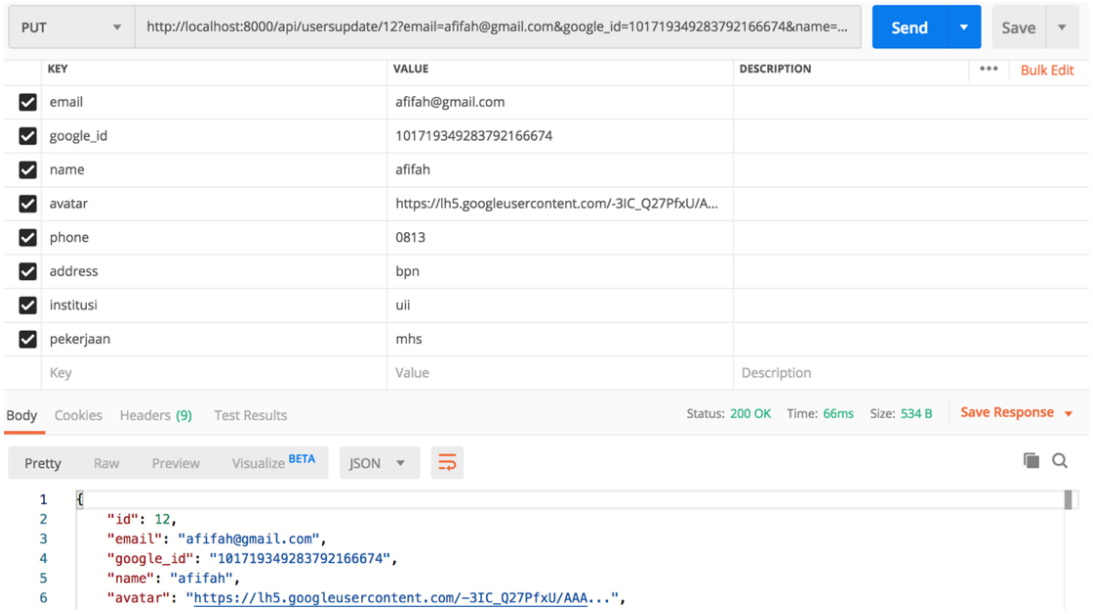
Gambar 4.13 Pengujian POST */participants* dengan Postman

Hasil pengujian yang didapat POST *users* pada Gambar 4.14 adalah status 200 OK. Hasil ini telah sesuai dengan yang dicantumkan pada Tabel 4.1. POST *users* memiliki fungsi menyimpan data pengguna ketika berhasil masuk, parameter masukan serta keluaran pada tampilan telah sesuai dengan yang dirancang pada Tabel 3.3.



Gambar 4.14 Pengujian POST */users* dengan Postman

Gambar 4.15 menampilkan hasil sesuai dengan Tabel 4.1 yang menunjukkan pengujian PUT *users* telah berhasil. Tampilan hasil telah sesuai dengan yang dirancang pada Tabel 3.4.



Gambar 4.15 Pengujian PUT */usersupdate/{id}* dengan Postman

4.1.3 Penyebaran

Tahap penyebaran merupakan tahap implementasi fungsi pada platform lain. Tahap ini dilakukan ketika fungsi telah melalui tahap pengujian menggunakan Postman. Penyebaran dilakukan dengan pendokumentasian fungsi berupa sebuah katalog berbasis *web*. Gambar 4.16 merupakan potongan kode dari dibangunnya sebuah katalog pada tahap penyebaran.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>REST API UTC</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.
min.css">

<style>
  header {
    text-align: center;
    font-family: "Tw Cen MT";
    color: #08387F;
  }
</style>
</head>

<body>
```

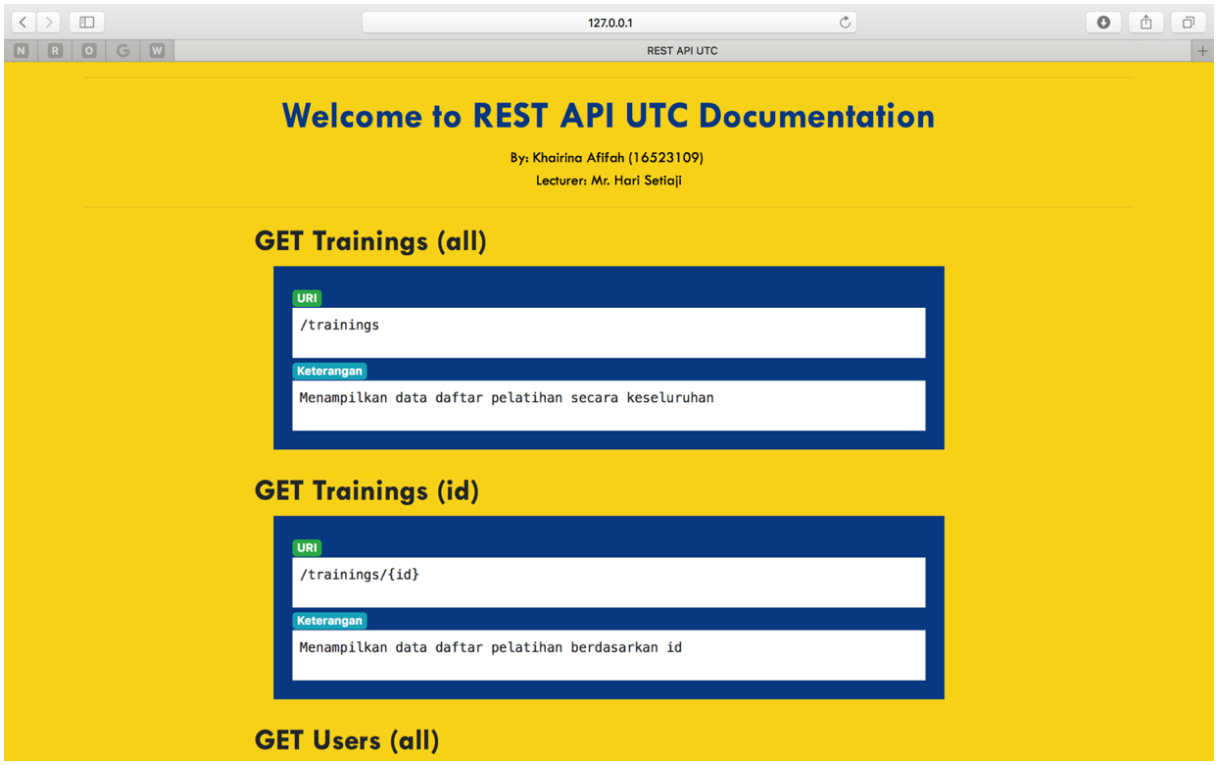
```
        <div class="container">
            <div class="row">
                <div class="col-md-12">

                <header>
                    <hr>
                        <div>
                            <h1><b>Welcome to REST API UTC
Documentation</b></h1>
                            <p>By: Khairina Afifah (16523109)
                            <br>Lecturer: Mr. Hari Setiaji</br></p>
                        </div>
                    <hr>
                </header>

        <h2><b>GET Trainings (all)</b></h2>
        <div class="col-md-8" style="margin: 0 auto">
            <div class="card-body" style="background-color:
#08387F">
                <span class="badge badge-success">URI</span>
                <div class="bg-white p-2">
                    <pre>/trainings</pre>
                </div>
                <span class="badge badge-info">Keterangan</span>
                <div class="bg-white p-2">
                    <pre>Menampilkan data pelatihan yang tersedia
secara keseluruhan</pre>
                </div>
            </div>
        </div>
<br>
.
.
.
.
<footer>
  <p class="title text-center">© 2019</p>
</footer>
    </body>
</html>
```

Gambar 4.16 Katalog REST API

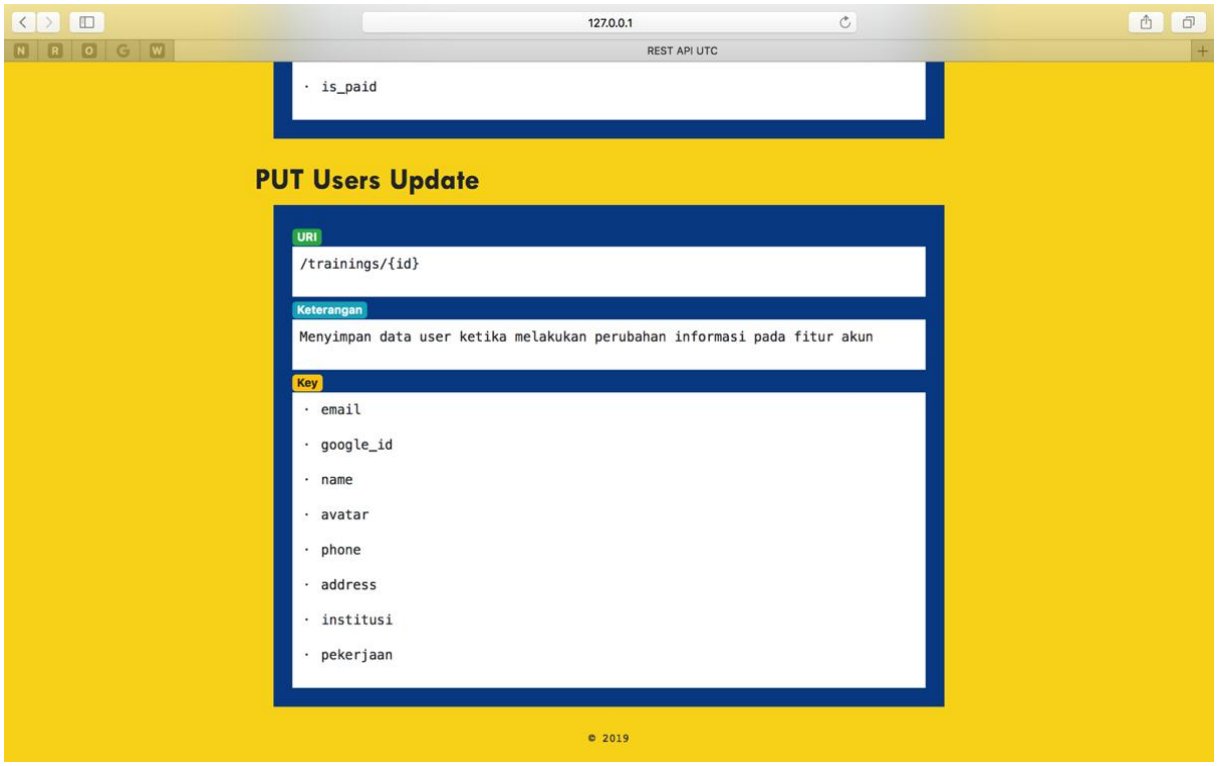
Hasil akhir dari pengembangan fungsi merupakan penyebaran yang dilakukan dengan pembuatan katalog berbasis *web.* Katalog akan memuat kumpulan fungsi yang telah berhasil melewati tahap pengujian. Dengan antarmuka sederhana katalog memiliki salah satu tujuan untuk memudahkan pengembang dalam pengambilan data. Dengan metode *iterative incremental* pengembangan *web service* sukses menghasilkan fungsi API dengan keseluruhan memiliki status uji 200 OK. Dengan pengembangan bertahap, seluruh fungsi selanjutnya

akan disentralkan pada katalog. Keseluruhan katalog memuat total delapan fungsi dengan *method* GET, POST, dan PUT.

Sedikit dari tampilan katalog dapat dilihat pada Gambar 4.17. Tampilan utama dari katalog REST API UTC memuat *method* yang digunakan, URI sesuai yang telah dirancang, dan keterangan mengenai fungsi tersebut. Tampilan selanjutnya dapat dilihat pada Gambar 4.18. Selain *method*, URI, dan keterangan. Katalog juga berisi *key* yang dibutuhkan oleh pengembang platform lain untuk memenuhi kebutuhan pengembang. Antarmuka katalog bersifat sederhana dengan salah satu tujuan memudahkan pengembang lain dalam mengakses informasi.



Gambar 4.17 Katalog REST API UTC



Gambar 4.18 Katalog REST API UTC

4.2 Pembahasan

Dibutuhkan aplikasi pengujian untuk membantu dalam proses pengembangan fungsi API. Keseluruhan hasil yang telah didapat telah berhasil diujikan dengan menggunakan aplikasi Postman. Pengujian dilakukan dengan aplikasi Postman karena fitur Postman yang sederhana membuat pengujian API dapat dilakukan dengan baik dan cepat (Wagner, 2014). Berbagai pertimbangan yang dilakukan dalam pemilihan aplikasi pengujian berpatok pada kemudahan, kecepatan pemahaman fitur, dan keefektifan serta efisiensi yang diberikan. Dengan berbagai penelitian terdahulu yang dijadikan acuan pada penelitian ini, Postman merupakan aplikasi yang banyak digunakan. Untuk itu, pembelajaran akan lebih mudah dan cepat karena banyaknya penelitian lain yang membahas hal serupa.

Seluruh fungsi yang telah berhasil dikembangkan dan diuji selanjutnya digabungkan pada pendokumentasian dalam bentuk katalog. Katalog berbasis *web* dengan menggunakan *framework* Laravel memuat delapan fungsi REST API sesuai dengan yang telah dirancang dan dikembangkan. Katalog yang telah usai dikembangkan selanjutnya akan digunakan oleh pengembang *mobile* untuk memenuhi kebutuhan fitur yang ada pada aplikasi UTC berbasis *mobile*. Pada akhirnya, katalog akan dapat diakses oleh para pengembang lain untuk membantu dalam mengambil informasi data pada *web* UTC.

Katalog yang dikembangkan pada penelitian masih sangat sederhana dan dapat diakses oleh publik. Hal ini dapat terjadi dengan pertimbangan isi dari katalog hanya sebatas konten pelatihan dan tidak adanya konten yang bersifat data pribadi atau *private*. Penelitian tidak bermaksud untuk mengambil jalur alternatif yang mudah dikembangkan namun, dengan aplikasi UTC yang tergolong masih baru penelitian ini merupakan terbukanya pintu bagi penelitian lebih lanjut. Pengembangan REST API pada aplikasi *web* UTC tidak berhenti pada penelitian ini dan akan terus berkembang mengikuti perkembangan dari *web* UTC itu sendiri. Berdasarkan tujuan yang telah dibahas pada bab awal bahwa penelitian dapat dijadikan sebagai bahan rujukan. Oleh karena itu, untuk *web* yang tergolong masih baru dan minim informasi, katalog REST API yang dikembangkan saat ini dirasa telah dapat membantu pengembang lain secara umum dalam memudahkan kebutuhan komunikasi data.