RESTFUL API SISTEM PRESENSI

RESTful API / REST API merupakan implementasi dari API (Application Programming Interface). REST (Representational State Transfer) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi. Dimana tujuannya adalah untuk menjadikan sistem yang memiliki performa yang baik, cepat dan mudah untuk di kembangkan (scale) terutama dalam pertukaran.

REST itu memanfaatkan empat metode di HTTP, yaitu:

1. GET
   Metode GET meminta representasi sumber daya yang ditentukan. Permintaan yang menggunakan GET hanya boleh mengambil data. Metode HEAD meminta respons yang identik dengan permintaan GET, tetapi tanpa badan respons.
2. POST
   Metode POST digunakan untuk mengirimkan entitas ke sumber daya yang ditentukan, sering menyebabkan perubahan keadaan atau efek samping pada server.
3. PUT
   Metode PUT menggantikan semua representasi saat ini dari sumber daya target dengan payload permintaan.
4. DELETE
   Metode DELETE menghapus sumber daya yang ditentukan.

DOKUMENTASI RESTful API

Source (Postman): https://documenter.getpostman.com/view/19273875/UVXqFD5i

1. STUDENT
   a. GET

GET student get index

http://192.168.56.28:8000/api/student

HEADERS

Accept	application/json

Contoh implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/student",
  "method": "GET",
  "timeout": 0,
  "headers": {
    "Accept": "application/json"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

GET student get show

http://192.168.56.28:8000/api/student/10

Contoh Implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/student/10",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

b. POST

POST **student post store**

http://192.168.56.28:8000/api/student

BODY formdata

name	Hafidz Ubaidillah
number	
birth	21-02-2001

Contoh Implementasi

```
var form = new FormData();
form.append("name", "Hafidz Ubaidillah");
form.append("number", "");
form.append("birth", "21-02-2001");

var settings = {
  "url": "http://192.168.56.28:8000/api/student",
  "method": "POST",
  "timeout": 0,
  "processData": false,
  "mimeType": "multipart/form-data",
  "contentType": false,
  "data": form
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

c. PUT

PUT **classroom put update**

http://192.168.56.28:8000/api/classroom/1?name=Lecturer 1&classroom=Physics

PARAMS

name	Lecturer 1
classroom	Physics

Contoh Implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/classroom/1?name=Lecturer 1&classroom=Physics",
  "method": "PUT",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

d. DELETE

DEL student delete destroy

http://192.168.56.28:8000/api/student/53

Contoh implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/classroom/1",
  "method": "DELETE",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

2. CLASSROOM

a. GET

GET classroom get index

http://192.168.56.28:8000/api/classroom

Contoh implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/classroom",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

GET classroom get show

http://192.168.56.28:8000/api/classroom/2

Contoh implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/classroom/2",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

b. POST

POST classroom post store

http://192.168.56.28:8000/api/classroom

BODY formdata

name	Teacher 1
classroom	Chemical

Contoh implementasi

```
var form = new FormData();
form.append("name", "Teacher 1");
form.append("classroom", "Chemical");

var settings = {
  "url": "http://192.168.56.28:8000/api/classroom"
  "method": "POST",
  "timeout": 0,
  "processData": false,
  "mimeType": "multipart/form-data",
  "contentType": false,
  "data": form
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

c. PUT

PUT classroom put update

http://192.168.56.28:8000/api/classroom/1?name=Lecturer 1&classroom=Physics

PARAMS

name	Lecturer 1
classroom	Physics

Contoh impelemtasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/classroom/1?name=Lecturer 1&classroom=Physics",
  "method": "PUT",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

d. Delete

DEL classroom delete destroy

http://192.168.56.28:8000/api/classroom/1

Contoh implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/classroom/1",
  "method": "DELETE",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

3. PRESENCE

a. GET

GET presence get index

http://192.168.56.28:8000/api/presence

Contoh implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/presence",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

GET presence get show

http://192.168.56.28:8000/api/presence/20

Contoh implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/presence/20",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

b. POST

POST presence post store

http://192.168.56.28:8000/api/presence

BODY formdata

id_classroom	2
number	24032001001
status	sick

Contoh implementasi

```
var form = new FormData();
form.append("id_classroom", "2");
form.append("number", "24032001001");
form.append("status", "sick");

var settings = {
  "url": "http://192.168.56.28:8000/api/presence",
  "method": "POST",
  "timeout": 0,
  "processData": false,
  "mimeType": "multipart/form-data",
  "contentType": false,
  "data": form
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

c. PUT

PUT presence put update

http://192.168.56.28:8000/api/presence/2?id_classroom=2&number=24032001001&status=not present

PARAMS

id_classroom	2
number	24032001001
status	not present

Contoh Implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/presence/2?id_classroom=2&number=24032001001&status
  "method": "PUT",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

4. DELETE

DEL **presence delete destroy**

http://192.168.56.28:8000/api/presence/1

Contoh Implementasi

```
var settings = {
  "url": "http://192.168.56.28:8000/api/presence/1",
  "method": "DELETE",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

DOKUMENTASI SOURCE CODE

Source code mentah kami letakkan di repository github dengan prosedur instalasi cloning proyek pada file README.md pada direktori root dan kami juga sediakan source code siap pakai tanpa melakukan prosedur instalasi.

Github Repository : https://github.com/Dzyfhuba/restful-api-prj.git

Ready-To-Use : https://github.com/Dzyfhuba/restful-api-prj/releases

1. Daftar route

Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	/	generated::u01vrY5PZQU7gXh7	Closure	web
	POST	api/classroom	classroom.store	App\Http\Controllers\ClassroomController@store	api
	GET|HEAD	api/classroom	classroom.index	App\Http\Controllers\ClassroomController@index	api
	DELETE	api/classroom/{classroom}	classroom.destroy	App\Http\Controllers\ClassroomController@destroy	api
	PUT|PATCH	api/classroom/{classroom}	classroom.update	App\Http\Controllers\ClassroomController@update	api
	GET|HEAD	api/classroom/{classroom}	classroom.show	App\Http\Controllers\ClassroomController@show	api
	GET|HEAD	api/classroom/{classroom}/edit	classroom.edit	App\Http\Controllers\ClassroomController@edit	api
	GET|HEAD	api/presence	presence.index	App\Http\Controllers\PresenceController@index	api
	POST	api/presence	presence.store	App\Http\Controllers\PresenceController@store	api
	GET|HEAD	api/presence/{presence}	presence.show	App\Http\Controllers\PresenceController@show	api
	PUT|PATCH	api/presence/{presence}	presence.update	App\Http\Controllers\PresenceController@update	api
	DELETE	api/presence/{presence}	presence.destroy	App\Http\Controllers\PresenceController@destroy	api
	GET|HEAD	api/presence/{presence}/edit	presence.edit	App\Http\Controllers\PresenceController@edit	api
	GET|HEAD	api/student	student.index	App\Http\Controllers\StudentController@index	api
	POST	api/student	student.store	App\Http\Controllers\StudentController@store	api
	GET|HEAD	api/student/{student}	student.show	App\Http\Controllers\StudentController@show	api
	DELETE	api/student/{student}	student.destroy	App\Http\Controllers\StudentController@destroy	api
	PUT|PATCH	api/student/{student}	student.update	App\Http\Controllers\StudentController@update	api
	GET|HEAD	api/student/{student}/edit	student.edit	App\Http\Controllers\StudentController@edit	api
	GET|HEAD	api/user	generated::ytFykdGfxx8Nssot	Closure	api App\Http\Middleware\Authenticate:sanctum
	GET|HEAD	sanctum/csrf-cookie	generated::QO8SPyWjNuZ8lZnf	Laravel\Sanctum\Http\Controllers\CsrfCookieController@show	web

2. Daftar Resource

```php
Route::resource('presence', PresenceController::class, ['except' => ['create']]);
Route::resource('student', StudentController::class, ['except' => ['create']]);
Route::resource('classroom', ClassroomController::class, ['except' => ['create']]);
```

3. Model student

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Student extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'number',
        'birth'
    ];
}
```

4. Model Classroom

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Classroom extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'classroom',
        'latitude',
        'longitude'
    ];
}
```

5. Model Presence

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Presence extends Model
{
    use HasFactory;

    protected $fillable = [
        'id_classroom',
        'number',
        'status',
        'latitude',
        'longitude'
    ];
}
```

6. Migration Student

```php
Schema::create('students', function (
Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('number')->unique();
    $table->string('birth');
    $table->timestamps();
});
```

7. Migration Classroom

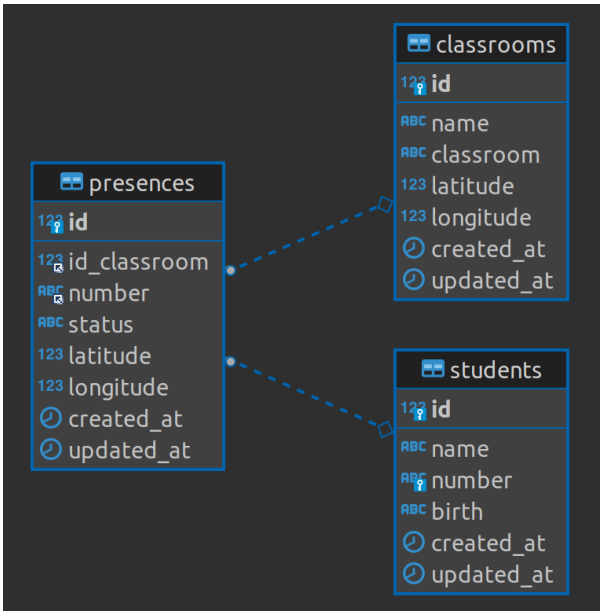
```php
Schema::create('classrooms', function (
Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('classroom');
    $table->double('latitude');
    $table->double('longitude');
    $table->timestamps();
});
```

8. Migration Presence

```php
Schema::create('presences', function (Blueprint $table) {
    $table->id();
    $table->foreignId('id_classroom')->references('id')->on('classrooms');
    $table->string('number');
    $table->enum('status', ['present', 'not present', 'permit', 'sick', 'forget'])->default('forget');
    $table->double('latitude');
    $table->double('longitude');
    $table->timestamps();

    $table->foreign('number')->references('number')->on('students');
```

9. Database



10. Student GET index

```php
public function index()
{
    $student = Student::all();
    return response()->json($student);
}
```

11. Student GET show

```php
public function show($id)
{
    $student = Student::where('id', $id)->get();
    return response()->json($student);
}
```

12. Student Meet Hold

```php
public function generateUniqueNumber($date)
{
    $i = 1;
    do {
        $number = str_replace('-', '', $date).sprintf('%03d', $i++);
    } while (Student::where("number", $number)->first());

    return $number;
}
```

13. Student Post Store

```php
public function store(Request $request)
{
    if (empty($request->number)) {
        $number = $this->generateUniqueNumber(
$request->birth);
        $request->merge(['number' => $number]);
    }

    Student::create($request->all());
    return [
        'status' => 'success',
        'student' => [
            $request->all()
        ]
    ];
}
```

14. Student PUT update

```php
public function update(Request $request, $id)
{
    if (empty($request->number)) {
        $number = $this->generateUniqueNumber($request->birth);
        $request->merge(['number' => $number]);
    }

    Student::where('id', $id)->update($request->all());
    return [
        'status' => 'success',
        'student' => [
            $request->all()
        ]
    ];
}
```

15. Student DELETE Destroy

```php
public function destroy($id)
{
    $student = Student::where('id', $id)->get();
    Student::where('id', $id)->delete();
    return [
        'status' => 'success',
        'student' => $student
    ];
}
```

16. Classroom GET indeks

```php
public function index()
{
    $classroom = Classroom::all();
    return response()->json($classroom);
}
```

17. Classroom GET show

```php
public function show($id)
{
    $classroom = Classroom::where('id', $id)->get();
    return response()->json($classroom);
}
```

18. Classroom POST store

```php
public function store(Request $request)
{
    Classroom::create($request->all());
    return [
        'status' => 'success',
        'classroom' => $request->all()
    ];
}
```

19. Classroom PUT update

```php
public function update(Request $request, $id)
{
    Classroom::where('id', $id)->update($request->all());
    return [
        'status' => 'success',
        'classroom' => [
            $request->all()
        ]
    ];
}
```

20. Classroom DELETE destroy

```php
public function destroy($id)
{
    $classroom = Classroom::where('id', $id)->get();
    Classroom::where('id', $id)->delete();
    return [
        'status' => 'success',
        'classroom' => $classroom
    ];
}
```

21. Presence GET indeks

```php
public function index()
{
    $presence = Presence::all();
    return response()->json($presence);
}
```

22. Presence GET show

```php
public function show($id)
{
    $presence = Presence::where('id', $id)->get();
    return response()->json($presence);
}
```

23. Presence POST store

```php
public function store(Request $request)
{
    Presence::create($request->all());
    return [
        'status' => 'success',
        'presence' => [
            $request->all()
        ]
    ];
}
```

24. Presence PUT update

```php
public function update(Request $request, $id)
{
    Presence::where('id', $id)->update($request->all());
    return [
        'status' => 'success',
        'presence' => [
            $request->all()
        ]
    ];
}
```

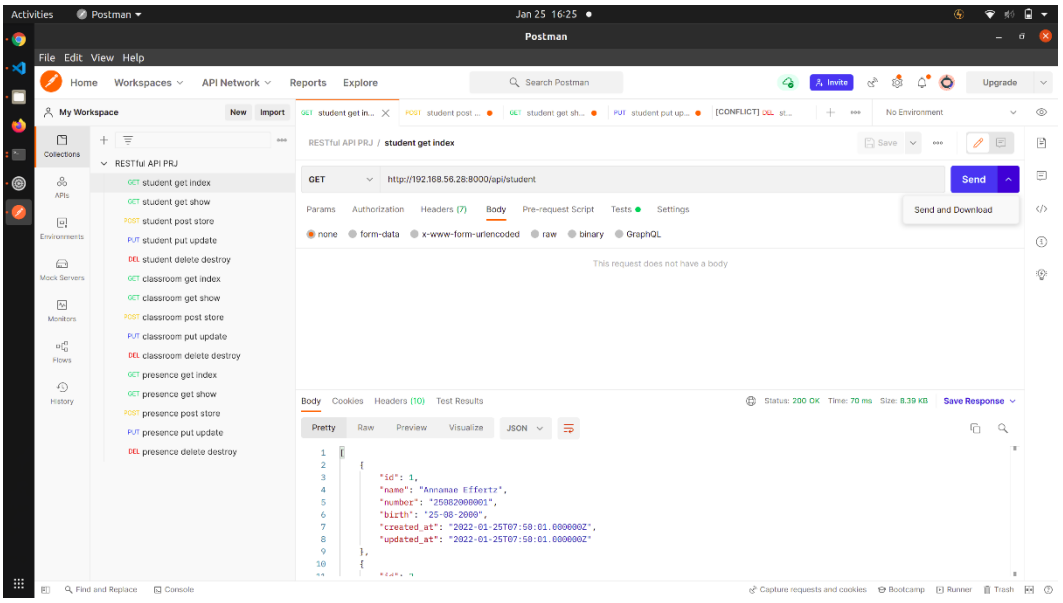
25. Presence DELETE destroy

```php
public function destroy($id)
{
    $presence = Presence::where('id', $id)->get();
    Presence::where('id', $id)->delete();
    return [
        'status' => 'success',
        'presence' => $presence
    ];
}
```

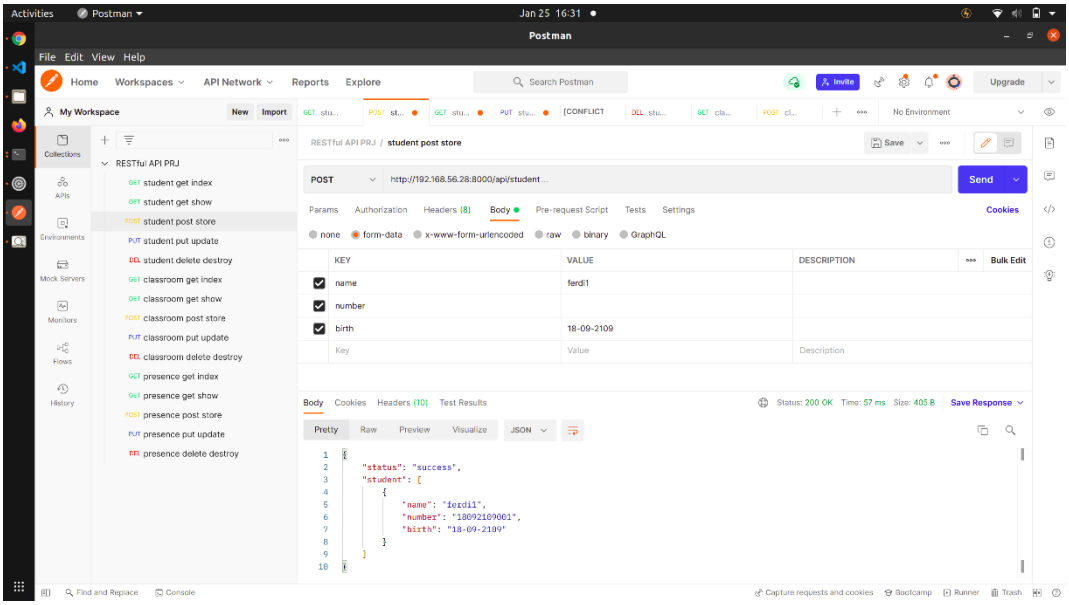
Testing API Request

Kami menggunakan software alat menggunakan Postman untuk menguji fitur API.

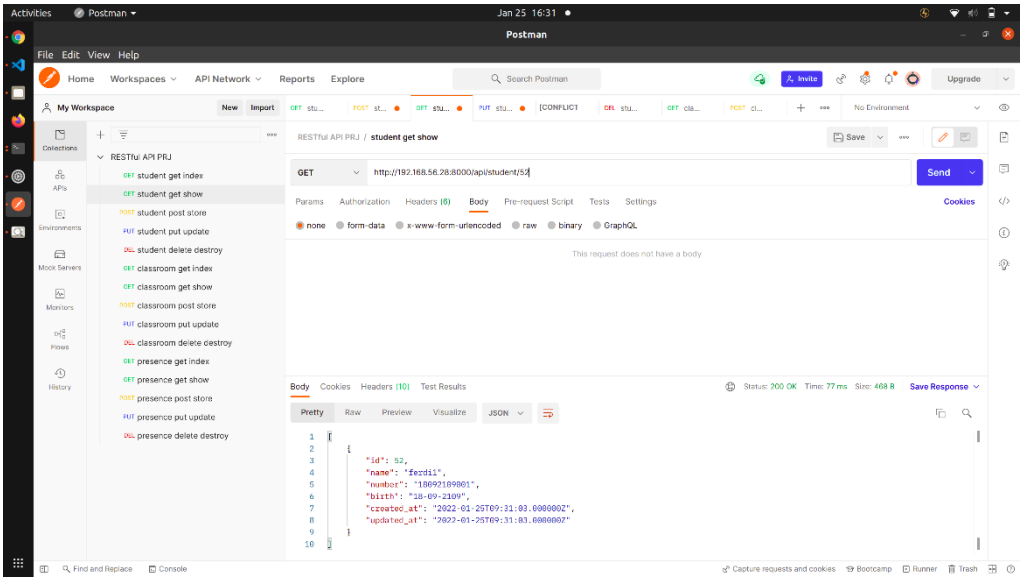
- Testing Student get index



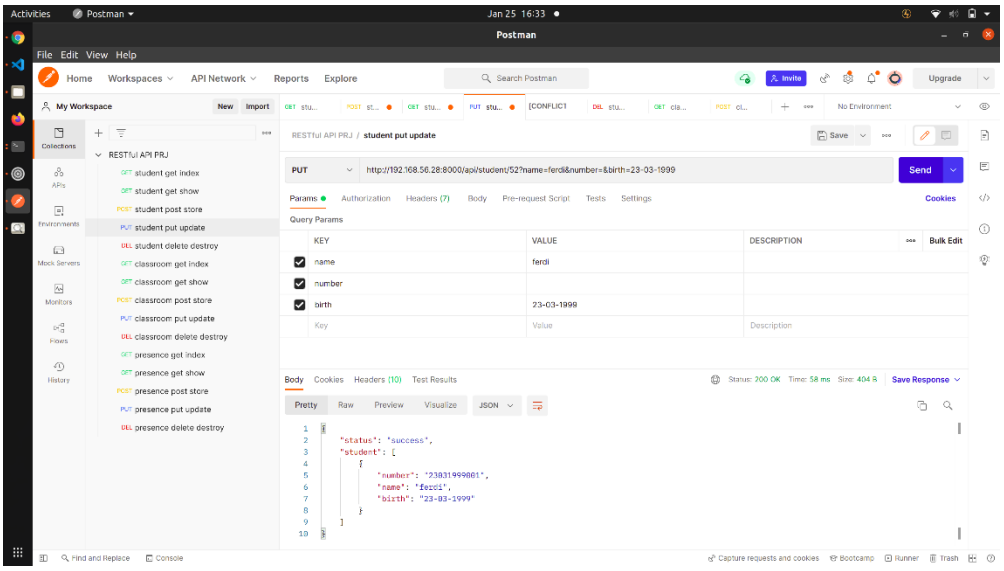
- Testing Student Get Store



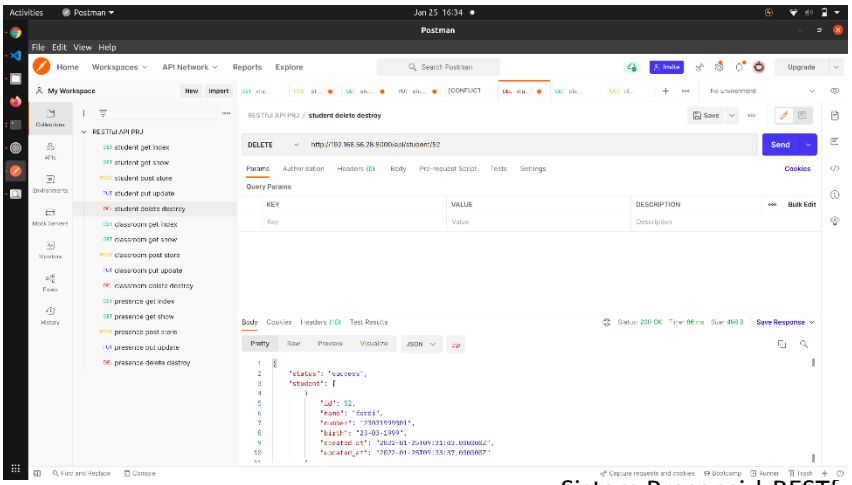
- Testing Get show



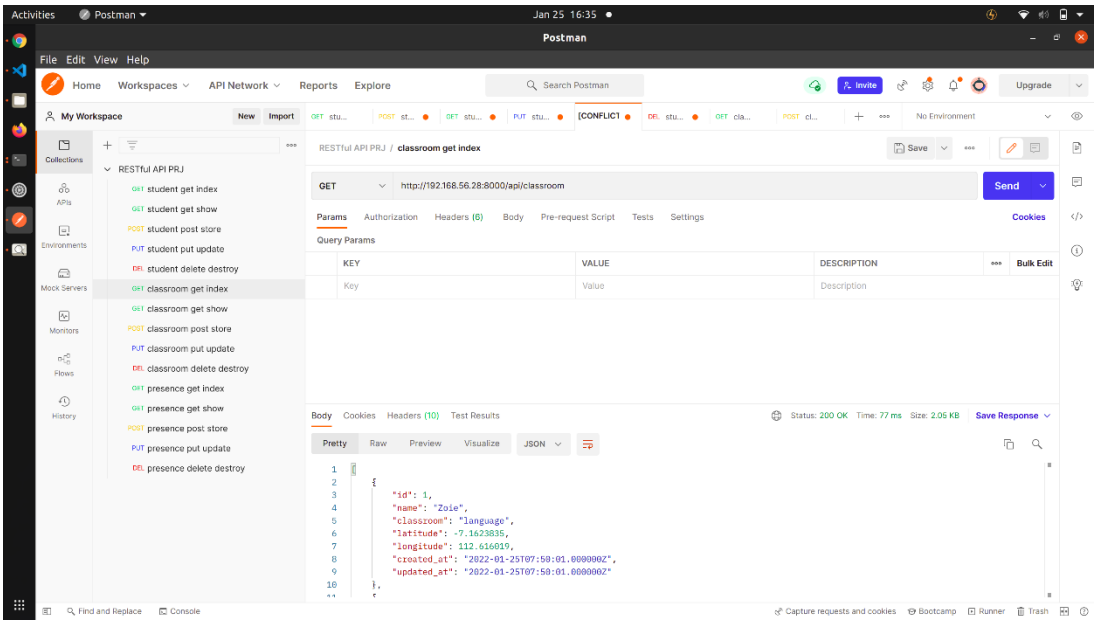
- Testing Student Put Update



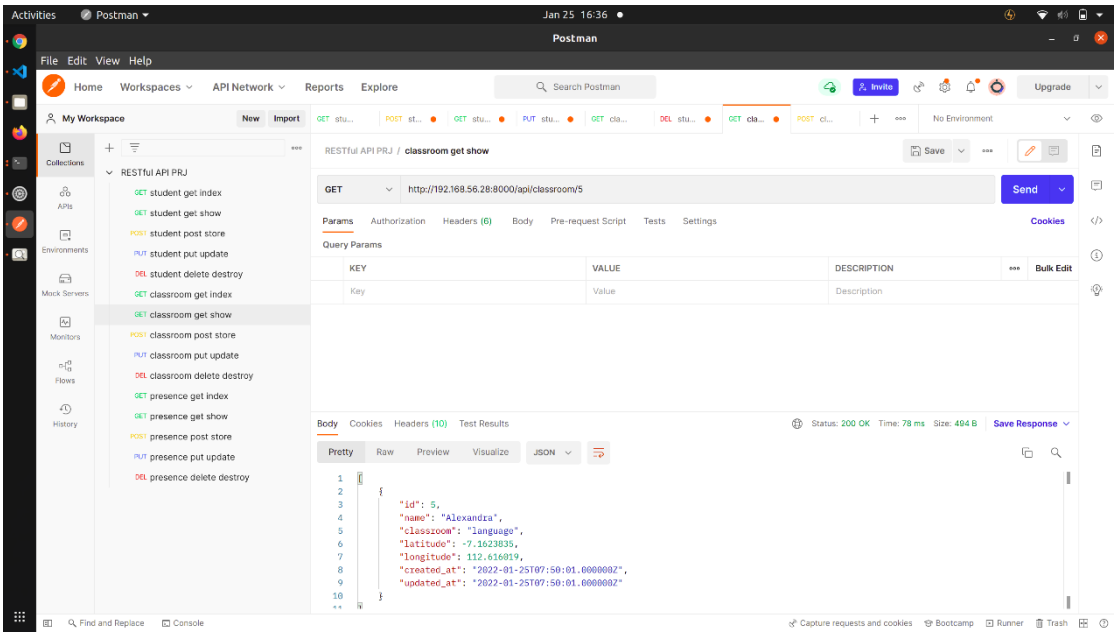
- Testing Student Delete Destroy



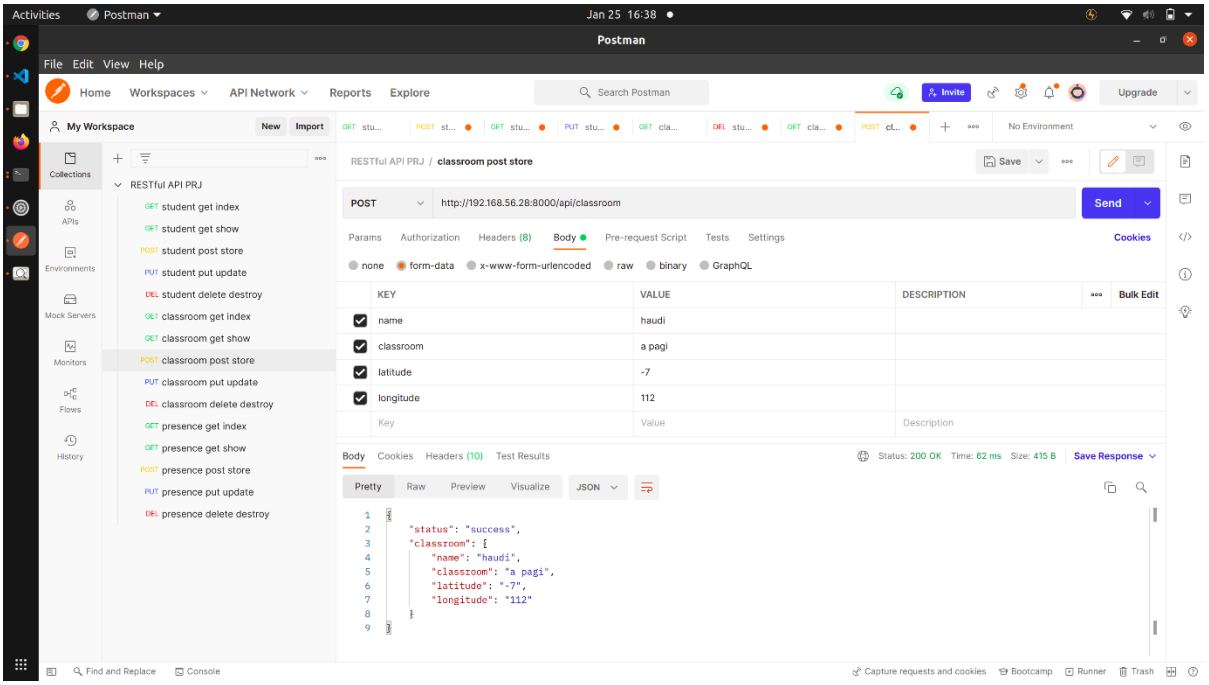
- Testing Classroom get index



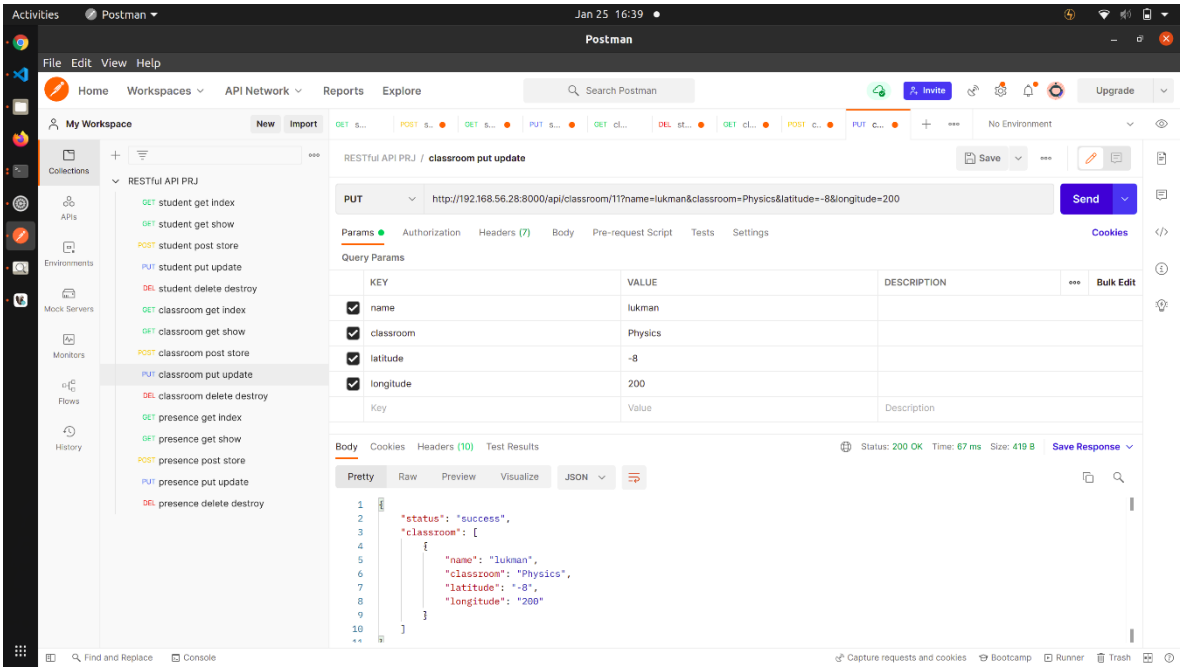
- Testing get show



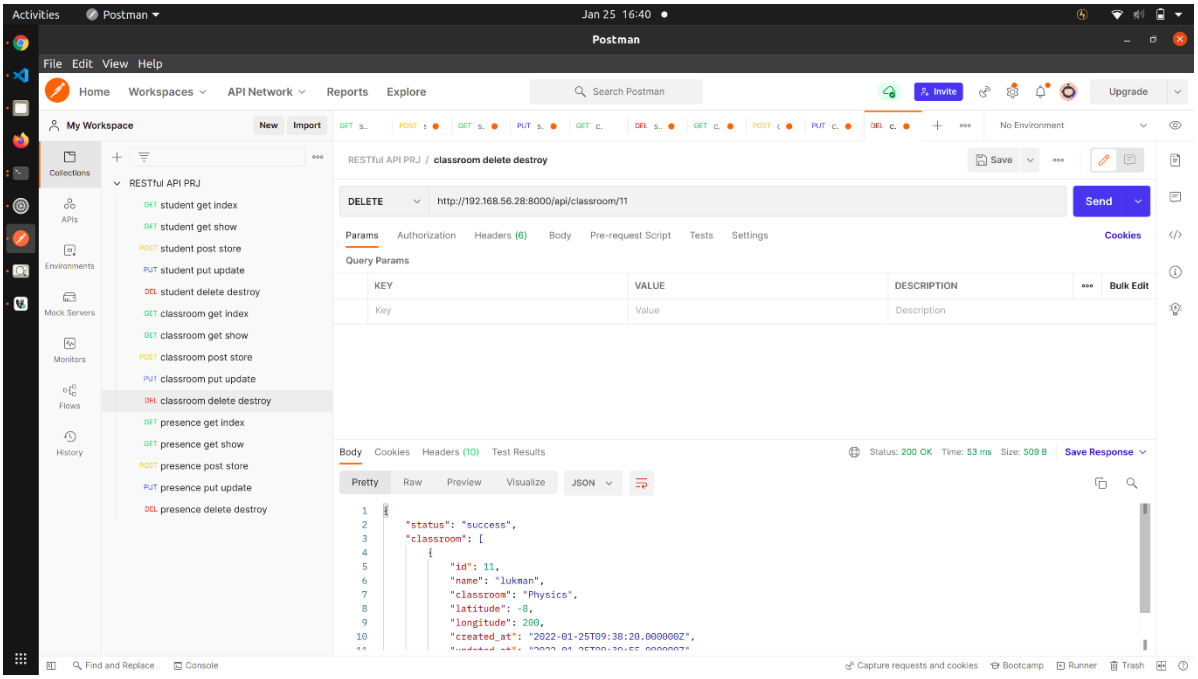
- Testing get store



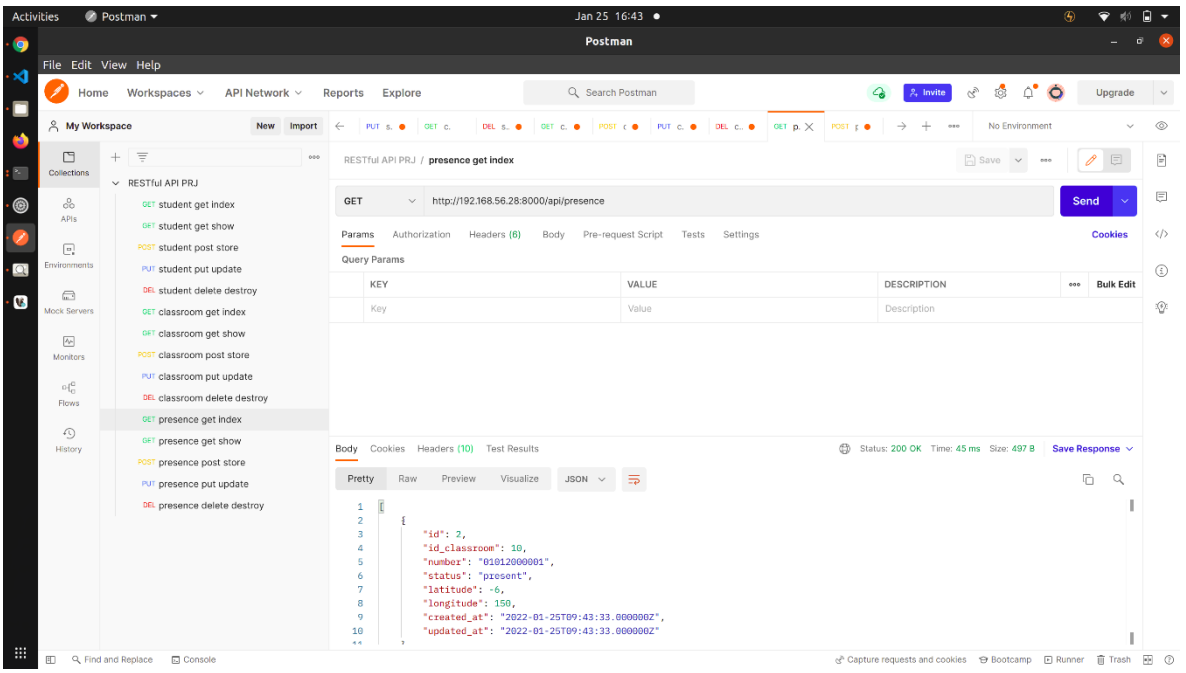
- Testing put update



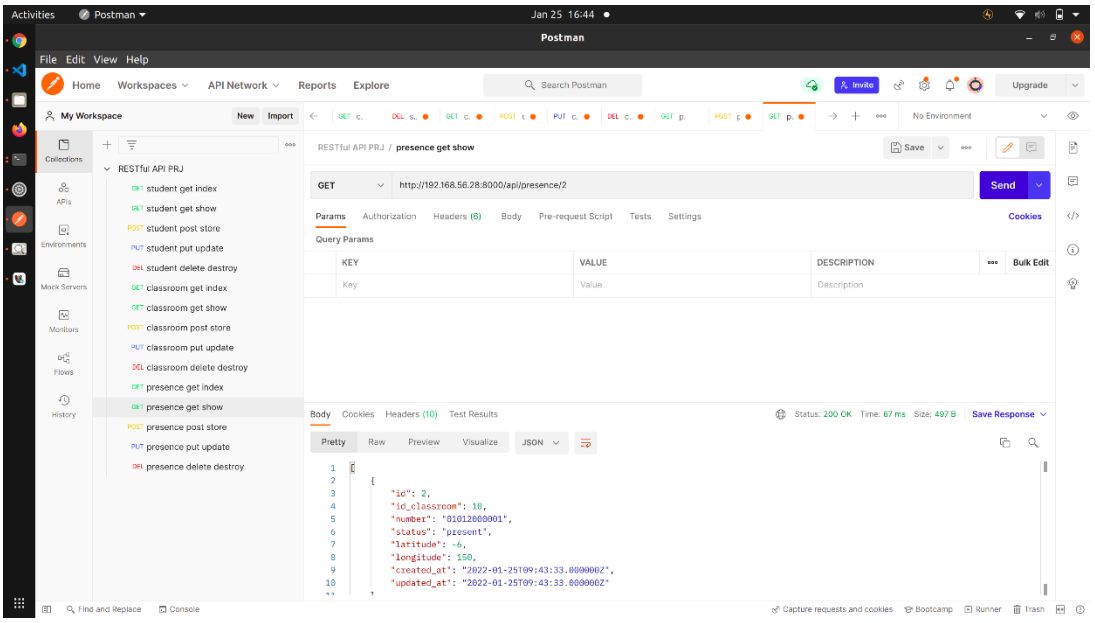
- Testing delete



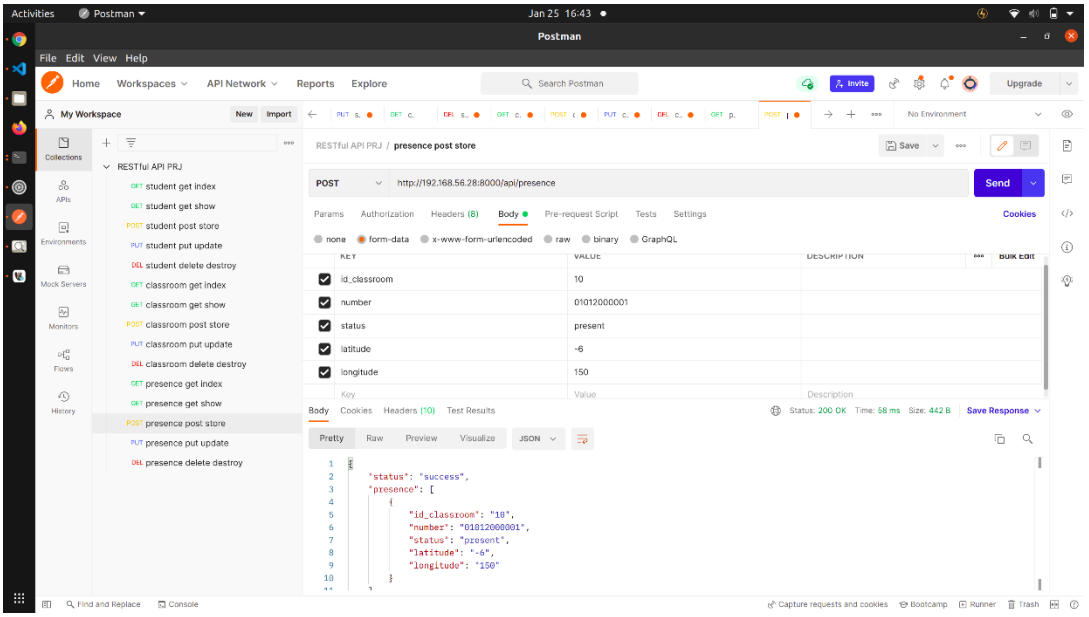
- Testing get presence



- Testing get show



- Testing store



- Testing update