

LAPORAN 9

Integratif Programming

The /authors API Resource



Disusun Oleh :

Nama : Ichsanul Dwi Prayitno

Nim : 4.33.20.0.14

Kelas : TI-4A

PROGRAM STUDI S.TR TEKNOLOGI REKAYASA KOMPUTER

JURUSAN TEKNIK ELEKTRO

POLITEKNIK NEGERI SEMARANG

TAHUN 2024

PRAKTEK

I. Lembar Kerja

No	Praktik	Hasil Program
1	Pembuatan Route /author pada web.php	<p>Buka file web.php dan tambahkan route author seperti berikut</p> <pre> Intergrative_Programming - web.php 1 \$router->get('/authors', 'AuthorsController@index'); 2 \$router->get('/authors/{id:[\d]+}', [3 'as' => 'authors.show', 4 'uses' => 'AuthorsController@show' 5]); 6 \$router->post('/authors', 'AuthorsController@store'); 7 \$router->put('/authors/{id:[\d]+}', 'AuthorsController@update'); 8 \$router->delete('/authors/{id:[\d]+}', 'AuthorsController@destroy');</pre>
2	Pembuatan AuthorController	Berikut adalah kode untuk author controller

		<pre> 1 <?php 2 3 namespace App\Http\Controllers; 4 5 use App\Models\Author; 6 use App\Transformer\AuthorTransformer; 7 use Illuminate\Http\Request; 8 9 class AuthorsController extends Controller 10 { 11 public function index() 12 { 13 return \$this->collection(14 Author::all(), 15 new AuthorTransformer() 16); 17 } 18 19 public function show(\$id) 20 { 21 return \$this->item(22 Author::findOrFail(\$id), 23 new AuthorTransformer() 24); 25 } 26 27 public function store(Request \$request) 28 { 29 \$this->validateAuthor(\$request); 30 31 \$author = Author::create(\$request->all()); 32 \$data = \$this->item(\$author, new AuthorTransformer()); 33 34 return response()->json(\$data, 201, [35 'Location' => route('authors.show', ['id' => \$author->id]) 36]); 37 } 38 39 public function update(Request \$request, \$id) 40 { 41 \$this->validateAuthor(\$request); 42 \$author = Author::findOrFail(\$id); 43 44 \$author->fill(\$request->all()); 45 \$author->save(); 46 47 \$data = \$this->item(\$author, new AuthorTransformer()); 48 49 return response()->json(\$data, 200); 50 } 51 52 public function destroy(\$id) 53 { 54 Author::findOrFail(\$id)->delete(); 55 56 return response()->json(['Pesan' => "Data Berhasil Di Hapus"], 200); 57 } 58 59 /** 60 * Validate author updates from the request. 61 * 62 * @param Request \$request 63 */ 64 private function validateAuthor(Request \$request) 65 { 66 \$this->validate(\$request, [67 'name' => 'required max:255', 68 'gender' => [69 'required', 70 'regex:/^(male female)\$/i', 71], 72 'biography' => 'required' 73], [74 'gender.regex' => "Gender format is invalid: must equal 'male' or 'female'" 75]); 76 } 77 } 78 </pre>
--	--	--

3	Pengeditan Controller	<p>Pada file controller tambahkan <code>\$this->fractal->parseIncludes();</code> pada function <code>__construct</code></p> <pre> Intergrative_Programming - Controller.php 1 public function __construct(FractalResponse \$fractal) 2 { 3 \$this->fractal = \$fractal; 4 \$this->fractal->parseIncludes(); 5 } </pre>
4	Pengeditan Project/app/Http/Response/FractalResponse.php	<p>Tambahkan argumen pada function <code>__construct</code> dan buatlah function baru berupa <code>parseIncludes</code></p> <pre> Intergrative_Programming - FractalResponse.php 1 public function __construct(Manager \$manager, SerializerAbstract \$serializer, Request \$request) 2 { 3 \$this->manager = \$manager; 4 \$this->serializer = \$serializer; 5 \$this->manager->setSerializer(\$serializer); 6 \$this->request = \$request; 7 } 8 9 /** 10 * Get the includes from the request if none are passed. 11 * 12 * @param null \$includes 13 */ 14 15 public function parseIncludes(\$includes = null) 16 { 17 if (empty(\$includes)) { 18 \$includes = \$this->request->query('include', ''); 19 } 20 \$this->manager->parseIncludes(\$includes); 21 } </pre>
5	Pengeditan Project/app/Providers/FractalServiceProvider.php	<p>Tambahkan kode <code>\$app['request']</code> dalam function <code>register</code></p> <pre> Intergrative_Programming - FractalServiceProvider.php 1 class FractalServiceProvider extends ServiceProvider 2 { 3 public function register() 4 { 5 // Bind the DataArraySerializer to an interface contract 6 \$this->app->bind(7 'League\Fractal\Serializer\SerializerAbstract', 8 'League\Fractal\Serializer\DataArraySerializer' 9); 10 11 \$this->app->bind(FractalResponse::class, function (\$app) { 12 \$manager = new Manager(); 13 \$serializer = \$app['League\Fractal\Serializer\SerializerAbstract']; 14 15 return new FractalResponse(\$manager, \$serializer, \$app['request']); 16 }); 17 18 \$this->app->alias(FractalResponse::class, 'fractal'); 19 } 20 } </pre>

6	<p>Pembuatan file transformer author</p> <p>Project/app/Transformer/AuthorTransformer.php</p>	<p>Berikut adalah isi kode dari authorTransformer</p> <pre> 1 <?php 2 3 namespace App\Transformer; 4 5 use App\Models\Author; 6 use League\Fractal\TransformerAbstract; 7 8 class AuthorTransformer extends TransformerAbstract 9 { 10 protected array \$availableIncludes = [11 'books' 12]; 13 14 public function transform(Author \$author) 15 { 16 return [17 'id' => (int) \$author->id, 18 'name' => \$author->name, 19 'gender' => \$author->gender, 20 'biography' => \$author->biography, 21 'created' => \$author->created_at->toIso8601String(), 22 'updated' => \$author->updated_at->toIso8601String(), 23 // add other author attributes here 24]; 25 } 26 27 public function includeBooks(\$author) 28 { 29 \$books = \$author->books; // Assuming \$author has a 'books' relationship 30 return \$this->collection(\$books, new BookTransformer()); 31 } 32 } 33 </pre>
7	<p>Pembuatan Testing</p>	<p>Buatlah test baru untuk AuthorController dengan isi berikut</p>

```

Intergrative_Programming - AuthorsControllerTest.php

1
2 // See Author Data
3 $this->seeJson([
4     'id' => $author->id,
5     'name' => $author->name,
6 ]);
7
8 // Test included book Data (the first record)
9 $actual = $data['books']['data'][0];
10 $this->assertEquals($book->id, $actual['id']);
11 $this->assertEquals($book->title, $actual['title']);
12 $this->assertEquals($book->description, $actual['description']);
13 $this->assertEquals(
14     $book->created_at->toIso8601String(),
15     $actual['created']
16 );
17 $this->assertEquals(
18     $book->updated_at->toIso8601String(),
19     $actual['updated']
20 );
21 }
22 /** @test */
23 public function store_can_create_a_new_author()
24 {
25     $postData = [
26         'name' => 'H. G. Wells',
27         'gender' => 'male',
28         'biography' => 'Prolific Science-Fiction Writer',
29     ];
30
31     $this->post('/authors', $postData, ['Accept' => 'application/json']);
32
33     $this->seeStatusCode(201);
34     $data = $this->response->getData(true);
35     $this->assertArrayHasKey('data', $data);
36     $this->seeJson($postData);
37
38     $this->seeInDatabase('authors', $postData);
39 }
40
41 /** @test */
42 public function validation_validates_required_fields()
43 {
44     // $author = factory(\App\Author::class)->create();
45     $author = Author::factory()->create();
46
47     $tests = [
48         ['method' => 'post', 'url' => '/authors'],
49         ['method' => 'put', 'url' => "/authors/{$author->id}"],
50     ];
51
52     foreach ($tests as $test) {
53         $method = $test['method'];
54         $this->{$method}($test['url'], [], ['Accept' => 'application/json']);
55         $this->seeStatusCode(422);
56         $data = $this->response->getData(true);
57
58         $fields = ['name', 'gender', 'biography'];
59
60         foreach ($fields as $field) {
61             $this->assertArrayHasKey($field, $data);
62             $this->assertEquals(["The {$field} field is required."], $data[$field]);
63         }
64     }
65 }
66
67 /** @test */
68 public function validation_invalidates_incorrect_gender_data()
69 {
70     foreach ($this->getValidationTestData() as $test) {
71         $method = $test['method'];
72         $test['data']['gender'] = 'unknown';
73         $this->{$method}($test['url'], $test['data'], ['Accept' => 'application/json']);
74
75         $this->seeStatusCode(422);
76
77         $data = $this->response->getData(true);
78         $this->assertCount(1, $data);
79         $this->assertArrayHasKey('gender', $data);
80         $this->assertEquals(
81             ["Gender format is invalid: must equal 'male' or 'female'"],
82             $data['gender']
83         );
84     }
85 }
86
87 /** @test */
88 public function validation_invalidates_name_when_name_is_just_too_long()
89 {
90     foreach ($this->getValidationTestData() as $test) {
91         $method = $test['method'];
92         $test['data']['name'] = str_repeat('a', 256);
93
94         $this->{$method}($test['url'], $test['data'], ['Accept' => 'application/json']);
95
96         $this->seeStatusCode(422);
97
98         $data = $this->response->getData(true);
99         $this->assertCount(1, $data);
100         $this->assertArrayHasKey('name', $data);

```

```

Intergrative_Programming - AuthorsControllerTest.php
1      $this->assertEquals(["The name must not be greater than 255 characters."], $data['name']);
2    }
3  }
4
5  /** @test */
6  public function validation_is_valid_when_name_is_just_long_enough()
7  {
8      foreach ($this->getValidationTestData() as $test) {
9          $method = $test['method'];
10         $test['data']['name'] = str_repeat('a', 255);
11
12         $this->{$method}($test['url'], $test['data'], ['Accept' => 'application/json']);
13
14         $this->seeStatusCode($test['status']);
15         $this->seeInDatabase('authors', $test['data']);
16     }
17 }
18
19 /** @test */
20 public function store_returns_a_valid_location_header()
21 {
22     $postData = [
23         'name' => 'H. G. Wells',
24         'gender' => 'male',
25         'biography' => 'Prolific Science-Fiction Writer'
26     ];
27
28     $this
29         ->post(
30             '/authors',
31             $postData,
32             ['Accept' => 'application/json']
33         )
34         ->seeStatusCode(201);
35
36     $data = $this->response->getData(true);
37     $this->assertArrayHasKey('data', $data);
38     $this->assertArrayHasKey('id', $data['data']);
39
40     // Check the Location header
41     $id = $data['data']['id'];
42     $this->seeHeaderWithRegExp('Location', "#/authors/{$id}#");
43 }
44
45 /** @test */
46 public function update_can_update_an_existing_author()
47 {
48     // $author = factory(\App\Author::class)->create();
49     $author = Author::factory()->create();
50
51
52     $requestData = [
53         'name' => 'New Author Name',
54         'gender' => $author->gender === 'male' ? 'female' : 'male',
55         'biography' => 'An updated biography',
56     ];
57
58     $this
59         ->put(
60             "/authors/{$author->id}",
61             $requestData,
62             ['Accept' => 'application/json']
63         )
64         ->seeStatusCode(200)
65         ->seeJson($requestData)
66         ->seeInDatabase('authors', [
67             'name' => 'New Author Name'
68         ])
69         ->notSeeInDatabase('authors', [
70             'name' => $author->name
71         ]);
72
73     $this->assertArrayHasKey('data', $this->response->getData(true));
74 }
75
76 /** @test */
77 public function update_method_validates_required_fields()
78 {
79     // $author = factory(\App\Author::class)->create();
80     $author = Author::factory()->create();
81
82     $this->put("/authors/{$author->id}", [], ['Accept' => 'application/json']);
83     $this->seeStatusCode(422);
84     $data = $this->response->getData(true);
85
86     $fields = ['name', 'gender', 'biography'];
87
88     foreach ($fields as $field) {
89         $this->assertArrayHasKey($field, $data);
90         $this->assertEquals(["The {$field} field is required."], $data[$field]);
91     }
92 }
93
94 /** @test */
95 public function delete_can_remove_an_author_and_his_or_her_books()
96 {
97     // $author = factory(\App\Author::class)->create();
98     $author = Author::factory()->create();
99
100

```

		<div data-bbox="582 212 662 235">● ● ●</div> <div data-bbox="790 212 1149 235">Intergrative_Programming - AuthorsControllerTest.php</div> <pre> 1 2 3 \$this 4 ->delete("/authors/{\\$author->id}") 5 ->seeStatusCode(204) 6 ->notSeeInDatabase('authors', ['id' => \\$author->id]) 7 ->notSeeInDatabase('books', ['author_id' => \\$author->id]); 8 } 9 10 /** @test */ 11 public function deleting_an_invalid_author_should_return_a_404() 12 { 13 \$this 14 ->delete('/authors/99999', [], ['Accept' => 'application/json']) 15 ->seeStatusCode(404); 16 } 17 18 /** 19 * Provides boilerplate test instructions for validation. 20 * @return array 21 */ 22 private function getValidationTestData() 23 { 24 // \$author = factory(\App\Author::class)->create(); 25 \$author = Author::factory()->create(); 26 27 return [28 // Create 29 [30 'method' => 'post', 31 'url' => '/authors', 32 'status' => 201, 33 'data' => [34 'name' => 'John Doe', 35 'gender' => 'male', 36 'biography' => 'An anonymous author' 37] 38], 39 // Update 40 [41 'method' => 'put', 42 'url' => "/authors/{\\$author->id}", 43 'status' => 200, 44 'data' => [45 'name' => \\$author->name, 46 'gender' => \\$author->gender, 47 'biography' => \\$author->biography 48] 49] 50]; 51 } 52 } 53 </pre>
--	--	---

		<pre> 1 ?php 2 3 namespace Tests\App\Http\Controllers; 4 5 use App\Models\Author; 6 use tests\TestCase; 7 use Illuminate\Http\Response; 8 use Laravel\Lumen\Testing\DatabaseMigrations; 9 10 class AuthorsControllerTest extends TestCase 11 { 12 use DatabaseMigrations; 13 14 /** @test */ 15 public function index_responds_with_200_status_code() 16 { 17 \$this->get('/authors')->seeStatusCode(Response::HTTP_OK); 18 } 19 20 /** @test */ 21 public function index_should_return_a_collection_of_records() 22 { 23 // \$authors = factory(\App\Author::class, 2)->create(); 24 \$authors = Author::factory(2)->create(); 25 26 \$this->get('/authors', ['Accept' => 'application/json']); 27 28 \$body = json_decode(\$this->response->getContent(), true); 29 \$this->assertArrayHasKey('data', \$body); 30 \$this->assertCount(2, \$body['data']); 31 32 foreach (\$authors as \$author) { 33 \$this->seeJson([34 'id' => \$author->id, 35 'name' => \$author->name, 36 'gender' => \$author->gender, 37 'biography' => \$author->biography, 38 'created' => \$author->created_at->toIso8601String(), 39 'updated' => \$author->updated_at->toIso8601String(), 40]); 41 } 42 } 43 44 /** @test */ 45 public function show_should_return_a_valid_author() 46 { 47 \$book = \$this->bookFactory(); 48 \$author = \$book->author; 49 50 \$this->get("/authors/{ \$author->id }", ['Accept' => 'application/json']); 51 \$body = json_decode(\$this->response->getContent(), true); 52 \$this->assertArrayHasKey('data', \$body); 53 54 \$this->seeJson([55 'id' => \$author->id, 56 'name' => \$author->name, 57 'gender' => \$author->gender, 58 'biography' => \$author->biography, 59 'created' => \$author->created_at->toIso8601String(), 60 'updated' => \$author->updated_at->toIso8601String(), 61]); 62 } 63 64 /** @test */ 65 public function show_should_fail_on_an_invalid_author() 66 { 67 \$this->get('/authors/1234', ['Accept' => 'application/json']); 68 \$this->seeStatusCode(Response::HTTP_NOT_FOUND); 69 70 \$this->seeJson([71 'message' => 'Not Found', 72 'status' => Response::HTTP_NOT_FOUND 73]); 74 75 \$body = json_decode(\$this->response->getContent(), true); 76 \$this->assertArrayHasKey('error', \$body); 77 \$error = \$body['error']; 78 79 \$this->assertEquals('Not Found', \$error['message']); 80 \$this->assertEquals(Response::HTTP_NOT_FOUND, \$error['status']); 81 } 82 83 /** @test */ 84 public function show_optionally_includes_books() 85 { 86 \$book = \$this->bookFactory(); 87 \$author = \$book->author; 88 89 \$this->get(90 "/authors/{ \$author->id }?include=books", 91 ['Accept' => 'application/json'] 92); 93 94 \$body = json_decode(\$this->response->getContent(), true); 95 96 \$this->assertArrayHasKey('data', \$body); 97 \$data = \$body['data']; 98 \$this->assertArrayHasKey('books', \$data); 99 \$this->assertArrayHasKey('data', \$data['books']); 100 \$this->assertCount(1, \$data['books']['data']); </pre>
--	--	---

Pada file

Project/tests/App/Http/Response/FractalResponseTest.php

Tambahkan `$request = m::mock(Request::class);` di setiap function

Intergrative_Programming - FractalResponseTest.php

```
1 $request = m::mock(Request::class);
```

Lalu tambahkan argumen `$request` di setiap new

`FractalResponse`

Intergrative_Programming - FractalResponseTest.php

```
1 $fractal = new FractalResponse($manager, $serializer, $request);
```

Dan buat lah 2 function baru yaitu

`it_should_parse_passed_includes_when_passed` , dan

`it_should_parse_request_query_includes_with_no_arguments`

Intergrative_Programming - FractalResponseTest.php

```
1 /** @test */
2 public function it_should_parse_passed_includes_when_passed()
3 {
4     $serializer = m::mock(SerializerAbstract::class);
5     $manager = m::mock(Manager::class);
6     $manager->shouldReceive('setSerializer')->with($serializer);
7     $manager
8         ->shouldReceive('parseIncludes')
9         ->with('books');
10
11     $request = m::mock(Request::class);
12     $request->shouldNotReceive('query');
13
14     $subject = new FractalResponse($manager, $serializer, $request);
15     $subject->parseIncludes('books');
16 }
17
18 /** @test */
19 public function it_should_parse_request_query_includes_with_no_arguments()
20 {
21     $serializer = m::mock(SerializerAbstract::class);
22     $manager = m::mock(Manager::class);
23     $manager->shouldReceive('setSerializer')->with($serializer);
24     $manager
25         ->shouldReceive('parseIncludes')
26         ->with('books');
27
28     $request = m::mock(Request::class);
29     $request
30         ->shouldReceive('query')
31         ->with('include', '')
32         ->andReturn('books');
33
34     (new FractalResponse($manager, $serializer, $request))->parseIncludes();
35     $this->assertTrue(true); // Placeholder assertion
36 }
```

		<p>Lalu buatlah file test baru</p> <p>Project/tests/App/Transformer/AuthorTransformerTest.php</p> <p>dengan isi</p> <pre> 1 <?php 2 3 namespace Tests\App\Transformer; 4 5 use App\Models\Author; 6 use tests\TestCase; 7 use App\Transformer\AuthorTransformer; 8 use Laravel\Lumen\Testing\DatabaseMigrations; 9 10 class AuthorTransformerTest extends TestCase 11 { 12 use DatabaseMigrations; 13 14 public function setUp() : void 15 { 16 parent::setUp(); 17 18 \$this->subject = new AuthorTransformer(); 19 } 20 /** @test */ 21 public function it_can_be_initialized() 22 { 23 \$this->assertInstanceOf(AuthorTransformer::class, \$this->subject); 24 } 25 26 /** @test */ 27 public function it_can_transform_an_author() 28 { 29 // \$author = factory(\App\Models\Author::class)->create(); 30 \$author = Author::factory()->create(); 31 32 \$actual = \$this->subject->transform(\$author); 33 34 \$this->assertEquals(\$author->id, \$actual['id']); 35 \$this->assertEquals(\$author->name, \$actual['name']); 36 \$this->assertEquals(\$author->gender, \$actual['gender']); 37 \$this->assertEquals(\$author->biography, \$actual['biography']); 38 \$this->assertEquals(39 \$author->created_at->toIso8601String(), 40 \$actual['created'] 41); 42 \$this->assertEquals(43 \$author->updated_at->toIso8601String(), 44 \$actual['created'] 45); 46 } 47 48 /** @test */ 49 public function it_can_transform_related_books() 50 { 51 \$book = \$this->bookFactory(); 52 \$author = \$book->author; 53 54 \$data = \$this->subject->includeBooks(\$author); 55 \$this->assertInstanceOf(\League\Fractal\Resource\Collection::class, \$data); 56 } 57 } 58 59 </pre>
8	Hasil	Hasil Test

```
PS D:\laragon\www\Intergrative_Programming\Project> .\vendor\bin\phpunit
PHPUnit 10.5.11 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.1.10
Configuration: D:\laragon\www\Intergrative_Programming\Project\phpunit.xml

..... 47 / 47 (100%)

Time: 03:06.538, Memory: 36.00 MB

OK (47 tests, 222 assertions)
PS D:\laragon\www\Intergrative_Programming\Project> |
```

Hasil /author/1?include=books

Authors / Pengambilan Data

GET http://localhost:8000/authors/1?include=books

Status: 200 OK Time: 1007 ms Size: 154 KB

Body

```
{
  "data": {
    "id": 1,
    "name": "Angela Larson",
    "gender": "male",
    "biography": "Officia modi esse exercitationem omnis at quisquam error. Quibusdam occaecati perferendis sapiente nam deleniti est. Est in rerum dicta. Dolor dolor architecto eveniet consectetur. Aut aut architecto eius qui sint.",
    "created": "2024-05-15T13:59:20+00:00",
    "updated": "2024-05-15T13:59:20+00:00",
    "books": [
      {
        "id": 1,
        "title": "Laudantium unde eum sequi est nulla magni sunt et sed omnis",
        "description": "Quasi voluptatem atque accusantium dolore. Praesentium ea voluptatum beatae tenetur ratione. Dolorem ipsa aspernatur non commodi et sapiente.",
        "author": "Angela Larson",
        "created": "2024-05-15T13:59:21+00:00",
        "updated": "2024-05-15T13:59:21+00:00"
      },
      {
        "id": 2,
        "title": "Eius unde error quod eum",

```

Hasil CRUD /author

Hasil query Read / Pembacaan seluruh data

Authors / Pengambilan Data

GET http://localhost:8000/authors

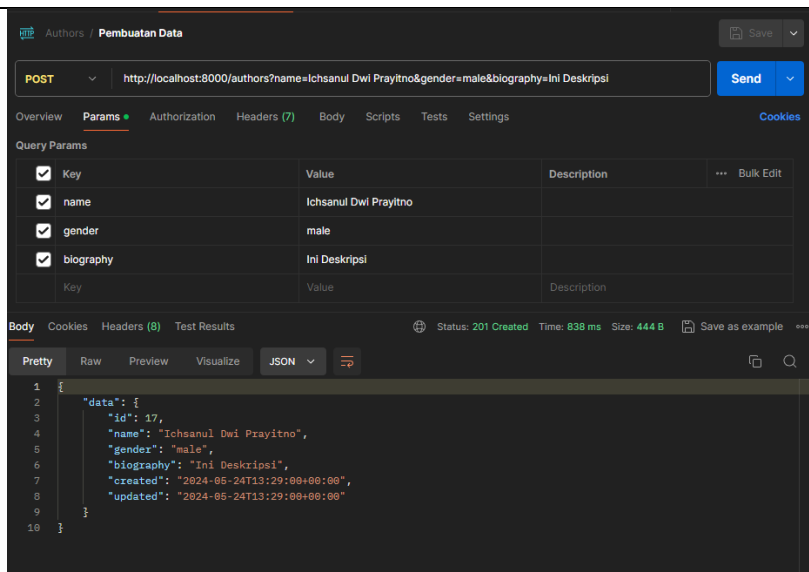
Status: 200 OK Time: 353 ms Size: 354 KB

Body

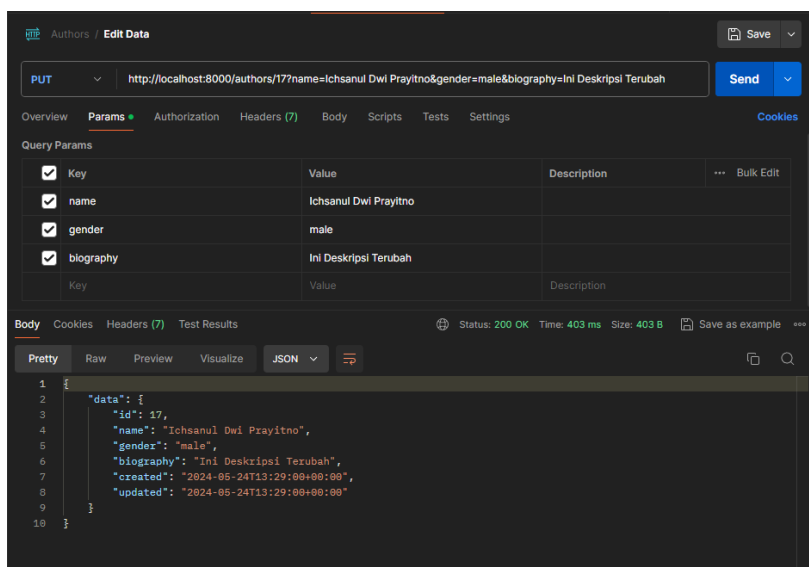
```
{
  "data": [
    {
      "id": 1,
      "name": "Angela Larson",
      "gender": "male",
      "biography": "Officia modi esse exercitationem omnis at quisquam error. Quibusdam occaecati perferendis sapiente nam deleniti est. Est in rerum dicta. Dolor dolor architecto eveniet consectetur. Aut aut architecto eius qui sint.",
      "created": "2024-05-15T13:59:20+00:00",
      "updated": "2024-05-15T13:59:20+00:00"
    },
    {
      "id": 2,
      "name": "Gail Auer",
      "gender": "male",
      "biography": "Molestiae labore inventore maxime aperiam recusandae odio error. Voluptatum rerum fugit atque eveniet. Enim ab repellendus provident et. Deserunt deserunt eveniet suscipit nihil.",
      "created": "2024-05-15T13:59:20+00:00",
      "updated": "2024-05-15T13:59:20+00:00"
    },
    {
      "id": 3,
      "name": "Randal Zulauf",

```

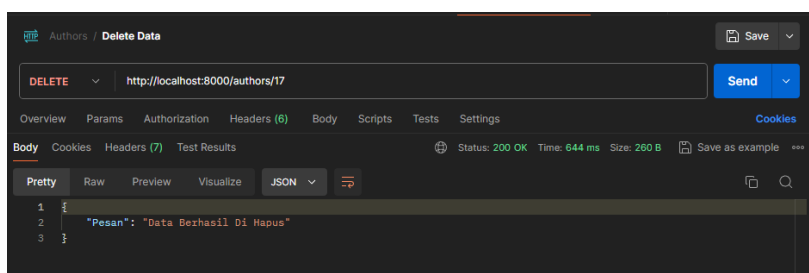
Hasil query Post / Pembuatan data



Hasil query Put / Pengeditan data



Hasil query Del / Penghapusan data



TEORI

I. Lembar Soal

N/A

II. Lembar Kerja

N/A