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	Buka file web.php dan tambahkan route author seperti berikut
		Intergrative_Programming - web.php
		`$router->get('/authors', 'AuthorsController@index');`
		`$router->get('/authors/{id:[\d]+}', [`
		`'as' => 'authors.show',`
		`'uses' => 'AuthorsController@show'`
		`]);`
		`$router->post('/authors', 'AuthorsController@store');`
		`$router->put('/authors/{id:[\d]+}', 'AuthorsController@update');`
		`$router->delete('/authors/{id:[\d]+}', 'AuthorsController@destroy');`

2	Pembuatan AuthorControll er	Berikut adalah kode untuk author controller

Intergrative_Programming - AuthorsController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\Author;
use App\Transformer\AuthorTransformer;
use Illuminate\Http\Request;

class AuthorsController extends Controller
{
    public function index()
    {
        return $this->collection(
            Author::all(),
            new AuthorTransformer()
        );
    }

    public function show($id)
    {
        return $this->item(
            Author::findorFail($id),
            new AuthorTransformer()
        );
    }

    public function store(Request $request)
    {
        $this->validateAuthor($request);

        $author = Author::create($request->all());
        $data = $this->item($author, new AuthorTransformer());

        return response()->json($data, 201, [
            'Location' => route('authors.show', ['id' => $author->id])
        ]);
    }

    public function update(Request $request, $id)
    {
        $this->validateAuthor($request);
        $author = Author::findOrFail($id);

        $author->fill($request->all());
        $author->save();

        $data = $this->item($author, new AuthorTransformer());

        return response()->json($data, 200);
    }

    public function destroy($id)
    {
        Author::findOrFail($id)->delete();

        return response()->json(['Pesan' => "Data Berhasil Di Hapus"], 200);
    }

    /**
     * Validate author updates from the request.
     *
     * @param Request $request
     */
    private function validateAuthor(Request $request)
    {
        $this->validate($request, [
            'name' => 'required|max:255',
            'gender' => [
                'required',
                'regex:/^(male|female)$/i',
            ],
            'biography' => 'required'
        ], [
            'gender.regex' => "Gender format is invalid: must equal 'male' or 'female'"
        ]);
    }
}

```

3	Pengeditan Controller	Pada file controller tambahkan $this->fractal->parseIncludes(); pada function __construct
		Intergrative_Programming - Controller.php
		`public function __construct(FractalResponse $fractal)`
		`{`
		`  $this->fractal = $fractal;`
		`  $this->fractal->parseIncludes();`
		`}`

4	Pengeditan
	Project/app/Http/Response/FractalResponse.php
		Tambahkan argumen pada function __construct dan buatlah function baru berupa parseIncludes
		Intergrative_Programming - FractalResponse.php
		`public function __construct(Manager $manager, SerializerAbstract $serializer, Request $request)`
		`{`
		`  $this->manager = $manager;`
		`  $this->serializer = $serializer;`
		`  $this->manager->setSerializer($serializer);`
		`  $this->request = $request;`
		`}`
		`/**`
		` * Get the includes from the request if none are passed.`
		` *`
		` * @param null $includes`
		` */`
		`public function parseIncludes($includes = null)`
		`{`
		`  if (empty($includes)) {`
		`  $includes = $this->request->query('include', '');`
		`  }`
		`  $this->manager->parseIncludes($includes);`
		`}`

5	Pengeditan
	Project/app/Providers/FractalServiceProvider.php
		Tambahkan kode $app[‘request’] dalam function register
		Intergrative_Programming - FractalServiceProvider.php
		`class FractalServiceProvider extends ServiceProvider`
		`{`
		`  public function register()`
		`  {`
		`  // Bind the DataArraySerializer to an interface contract`
		`  $this->app->bind(`
		`  'League\Fractal\Serializer\SerializerAbstract',`
		`  'League\Fractal\Serializer\DataArraySerializer'`
		`  );`
		`  $this->app->bind(FractalResponse::class, function ($app) {`
		`  $manager = new Manager();`
		`  $serializer = $app['League\Fractal\Serializer\SerializerAbstract'];`
		`  return new FractalResponse($manager, $serializer, $app['request']);`
		`  });`
		`  $this->app->alias(FractalResponse::class, 'fractal');`
		`  }`
		`}`

6	Pembuatan file transformer author Project/app/Transformer/AuthorTransformer.php	Berikut adalah isi kode dari authorTransformer 
7	Pembuatan Testing	Buatlah test baru untuk AuthorController dengan isi berikut

Intergrative_Programming - AuthorsControllerTest.php

```php

    // See Author Data
    $this->seeJson([
        'id' => $author->id,
        'name' => $author->name,
    ]);

    // Test included book Data (the first record)
    $actual = $data['books']['data'][0];
    $this->assertEquals($book->id, $actual['id']);
    $this->assertEquals($book->title, $actual['title']);
    $this->assertEquals($book->description, $actual['description']);
    $this->assertEquals(
        $book->created_at->toIso8601String(),
        $actual['created']
    );
    $this->assertEquals(
        $book->updated_at->toIso8601String(),
        $actual['updated']
    );
}
/** @test **/
public function store_can_create_a_new_author()
{
    $postData = [
        'name' => 'H. G. Wells',
        'gender' => 'male',
        'biography' => 'Prolific Science-Fiction Writer',
    ];

    $this->post('/authors', $postData, ['Accept' => 'application/json']);

    $this->seeStatusCode(201);
    $data = $this->response->getData(true);
    $this->assertArrayHasKey('data', $data);
    $this->seeJson($postData);

    $this->seeInDatabase('authors', $postData);
}

/** @test **/
public function validation_validates_required_fields()
{
    // $author = factory(\App\Author::class)->create();
    $author = Author::factory()->create();

    $tests = [
        ['method' => 'post', 'url' => '/authors'],
        ['method' => 'put', 'url' => "/authors/{$author->id}"],
    ];

    foreach ($tests as $test) {
        $method = $test['method'];
        $this->{$method}($test['url'], [], ['Accept' => 'application/json']);
        $this->seeStatusCode(422);
        $data = $this->response->getData(true);

        $fields = ['name', 'gender', 'biography'];

        foreach ($fields as $field) {
            $this->assertArrayHasKey($field, $data);
            $this->assertEquals(["The {$field} field is required."], $data[$field]);
        }
    }
}

/** @test **/
public function validation_invalidates_incorrect_gender_data()
{
    foreach ($this->getValidationTestData() as $test) {
        $method = $test['method'];
        $test['data']['gender'] = 'unknown';
        $this->{$method}($test['url'], $test['data'], ['Accept' => 'application/json']);

        $this->seeStatusCode(422);

        $data = $this->response->getData(true);
        $this->assertCount(1, $data);
        $this->assertArrayHasKey('gender', $data);
        $this->assertEquals(
            ["Gender format is invalid: must equal 'male' or 'female'"],
            $data['gender']
        );
    }
}

/** @test **/
public function validation_invalidates_name_when_name_is_just_too_long()
{
    foreach ($this->getValidationTestData() as $test) {
        $method = $test['method'];
        $test['data']['name'] = str_repeat('a', 256);

        $this->{$method}($test['url'], $test['data'], ['Accept' => 'application/json']);

        $this->seeStatusCode(422);

        $data = $this->response->getData(true);
        $this->assertCount(1, $data);
        $this->assertArrayHasKey('name', $data);
```

Intergrative_Programming - AuthorsControllerTest.php

```php
        $this->assertEquals(["The name must not be greater than 255 characters."], $data['name']);
    }
}

/** @test **/
public function validation_is_valid_when_name_is_just_long_enough()
{
    foreach ($this->getValidationTestData() as $test) {
        $method = $test['method'];
        $test['data']['name'] = str_repeat('a', 255);

        $this->{$method}($test['url'], $test['data'], ['Accept' => 'application/json']);

        $this->seeStatusCode($test['status']);
        $this->seeInDatabase('authors', $test['data']);
    }
}

/** @test **/
public function store_returns_a_valid_location_header()
{
    $postData = [
        'name' => 'H. G. Wells',
        'gender' => 'male',
        'biography' => 'Prolific Science-Fiction Writer'
    ];

    $this
        ->post(
            '/authors',
            $postData,
            ['Accept' => 'application/json']
        )
        ->seeStatusCode(201);

    $data = $this->response->getData(true);
    $this->assertArrayHasKey('data', $data);
    $this->assertArrayHasKey('id', $data['data']);

    // Check the Location header
    $id = $data['data']['id'];
    $this->seeHeaderWithRegExp('Location', "#/authors/{$id}$#");
}

/** @test **/
public function update_can_update_an_existing_author()
{
    // $author = factory(\App\Author::class)->create();
    $author = Author::factory()->create();


    $requestData = [
        'name' => 'New Author Name',
        'gender' => $author->gender === 'male' ? 'female' : 'male',
        'biography' => 'An updated biography',
    ];

    $this
        ->put(
            "/authors/{$author->id}",
            $requestData,
            ['Accept' => 'application/json']
        )
        ->seeStatusCode(200)
        ->seeJson($requestData)
        ->seeInDatabase('authors', [
            'name' => 'New Author Name'
        ])
        ->notSeeInDatabase('authors', [
            'name' => $author->name
        ]);

    $this->assertArrayHasKey('data', $this->response->getData(true));
}

/** @test **/
public function update_method_validates_required_fields()
{
    // $author = factory(\App\Author::class)->create();
    $author = Author::factory()->create();

    $this->put("/authors/{$author->id}", [], ['Accept' => 'application/json']);
    $this->seeStatusCode(422);
    $data = $this->response->getData(true);

    $fields = ['name', 'gender', 'biography'];

    foreach ($fields as $field) {
        $this->assertArrayHasKey($field, $data);
        $this->assertEquals(["The {$field} field is required."], $data[$field]);
    }
}

/** @test **/
public function delete_can_remove_an_author_and_his_or_her_books()
{
    // $author = factory(\App\Author::class)->create();
    $author = Author::factory()->create();


```

Intergrative_Programming - AuthorsControllerTest.php

```php

        $this
            ->delete("/authors/{$author->id}")
            ->seeStatusCode(204)
            ->notSeeInDatabase('authors', ['id' => $author->id])
            ->notSeeInDatabase('books', ['author_id' => $author->id]);
    }

    /** @test **/
    public function deleting_an_invalid_author_should_return_a_404()
    {
        $this
            ->delete('/authors/99999', [], ['Accept' => 'application/json'])
            ->seeStatusCode(404);
    }

    /**
     * Provides boilerplate test instructions for validation.
     * @return array
     */
    private function getValidationTestData()
    {
        // $author = factory(\App\Author::class)->create();
        $author = Author::factory()->create();

        return [
            // Create
            [
                'method' => 'post',
                'url' => '/authors',
                'status' => 201,
                'data' => [
                    'name' => 'John Doe',
                    'gender' => 'male',
                    'biography' => 'An anonymous author'
                ]
            ],

            // Update
            [
                'method' => 'put',
                'url' => "/authors/{$author->id}",
                'status' => 200,
                'data' => [
                    'name' => $author->name,
                    'gender' => $author->gender,
                    'biography' => $author->biography
                ]
            ]
        ];
    }
}

```

Intergrative_Programming - AuthorsControllerTest.php

```php
<?php

namespace Tests\App\Http\Controllers;

use App\Models\Author;
use tests\TestCase;
use Illuminate\Http\Response;
use Laravel\Lumen\Testing\DatabaseMigrations;

class AuthorsControllerTest extends TestCase
{
    use DatabaseMigrations;

    /** @test **/
    public function index_responds_with_200_status_code()
    {
        $this->get('/authors')->seeStatusCode(Response::HTTP_OK);
    }

    /** @test **/
    public function index_should_return_a_collection_of_records()
    {
        // $authors = factory(\App\Author::class, 2)->create();
        $authors = Author::factory(2)->create();

        $this->get('/authors', ['Accept' => 'application/json']);

        $body = json_decode($this->response->getContent(), true);
        $this->assertArrayHasKey('data', $body);
        $this->assertCount(2, $body['data']);

        foreach ($authors as $author) {
            $this->seeJson([
                'id' => $author->id,
                'name' => $author->name,
                'gender' => $author->gender,
                'biography' => $author->biography,
                'created' => $author->created_at->toIso8601String(),
                'updated' => $author->updated_at->toIso8601String(),
            ]);
        }
    }

    /** @test **/
    public function show_should_return_a_valid_author()
    {
        $book = $this->bookFactory();
        $author = $book->author;

        $this->get("/authors/{$author->id}", ['Accept' => 'application/json']);
        $body = json_decode($this->response->getContent(), true);
        $this->assertArrayHasKey('data', $body);

        $this->seeJson([
            'id' => $author->id,
            'name' => $author->name,
            'gender' => $author->gender,
            'biography' => $author->biography,
            'created' => $author->created_at->toIso8601String(),
            'updated' => $author->updated_at->toIso8601String(),
        ]);
    }

    /** @test **/
    public function show_should_fail_on_an_invalid_author()
    {
        $this->get('/authors/1234', ['Accept' => 'application/json']);
        $this->seeStatusCode(Response::HTTP_NOT_FOUND);

        $this->seeJson([
            'message' => 'Not Found',
            'status' => Response::HTTP_NOT_FOUND
        ]);

        $body = json_decode($this->response->getContent(), true);
        $this->assertArrayHasKey('error', $body);
        $error = $body['error'];

        $this->assertEquals('Not Found', $error['message']);
        $this->assertEquals(Response::HTTP_NOT_FOUND, $error['status']);
    }

    /** @test **/
    public function show_optionally_includes_books()
    {
        $book = $this->bookFactory();
        $author = $book->author;

        $this->get(
            "/authors/{$author->id}?include=books",
            ['Accept' => 'application/json']
        );

        $body = json_decode($this->response->getContent(), true);

        $this->assertArrayHasKey('data', $body);
        $data = $body['data'];
        $this->assertArrayHasKey('books', $data);
        $this->assertArrayHasKey('data', $data['books']);
        $this->assertCount(1, $data['books']['data']);
```

Pada file Project/tests/App/Http/Response/FractalResponseTest.php Tambahkan $request = m::mock(Request::class); di setiap function

Intergrative_Programming - FractalResponseTest.php

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

Lalu tambahkan argumen $request di setiap new FractalResponse

Intergrative_Programming - FractalResponseTest.php

```php
$fractal = new FractalResponse($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

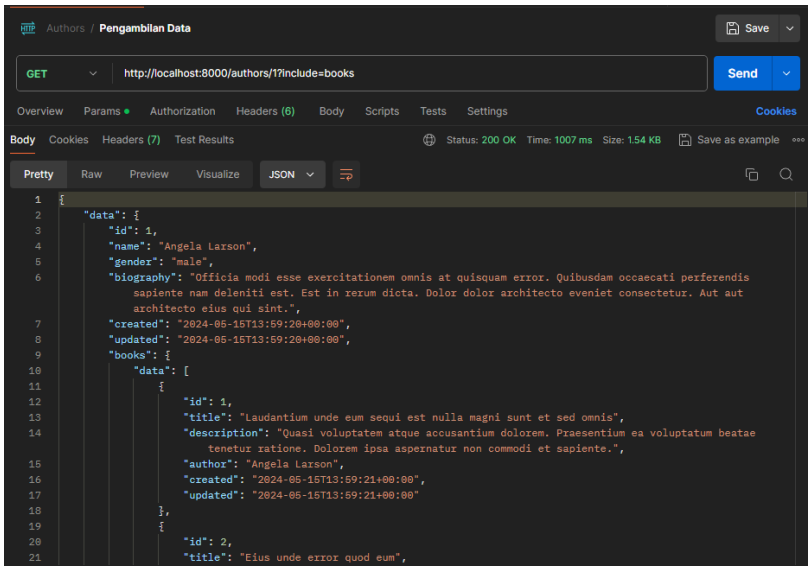
Intergrative_Programming - FractalResponseTest.php

```php
/** @test **/
public function it_should_parse_passed_includes_when_passed()
{
    $serializer = m::mock(SerializerAbstract::class);
    $manager = m::mock(Manager::class);
    $manager->shouldReceive('setSerializer')->with($serializer);
    $manager
        ->shouldReceive('parseIncludes')
        ->with('books');

    $request = m::mock(Request::class);
    $request->shouldNotReceive('query');

    $subject = new FractalResponse($manager, $serializer, $request);
    $subject->parseIncludes('books');
}

/** @test **/
public function it_should_parse_request_query_includes_with_no_arguments()
{
    $serializer = m::mock(SerializerAbstract::class);
    $manager = m::mock(Manager::class);
    $manager->shouldReceive('setSerializer')->with($serializer);
    $manager
        ->shouldReceive('parseIncludes')
        ->with('books');

    $request = m::mock(Request::class);
    $request
        ->shouldReceive('query')
        ->with('include', '')
        ->andReturn('books');

    (new FractalResponse($manager, $serializer, $request))->parseIncludes();
    $this->assertTrue(true); // Placeholder assertion
}
```

		Lalu buatlah file test baru Project/tests/App/Transformer/AuthorTransformerTest.php dengan isi 
8	Hasil	Hasil Test

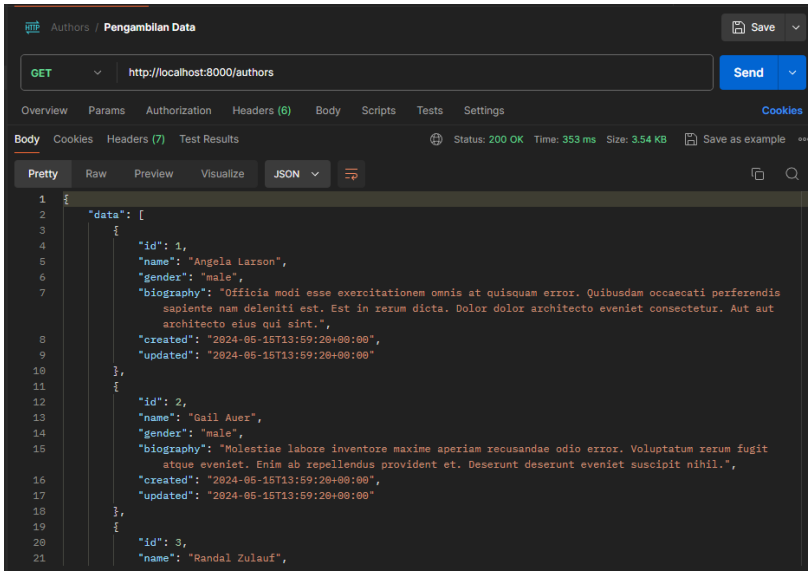
```
PS D:\laragon\www\Intergrative_Programming\Project> .\vendor\bin\phpunit
PHPUnit 10.5.11 by Sebastian Bergmann and contributors.

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

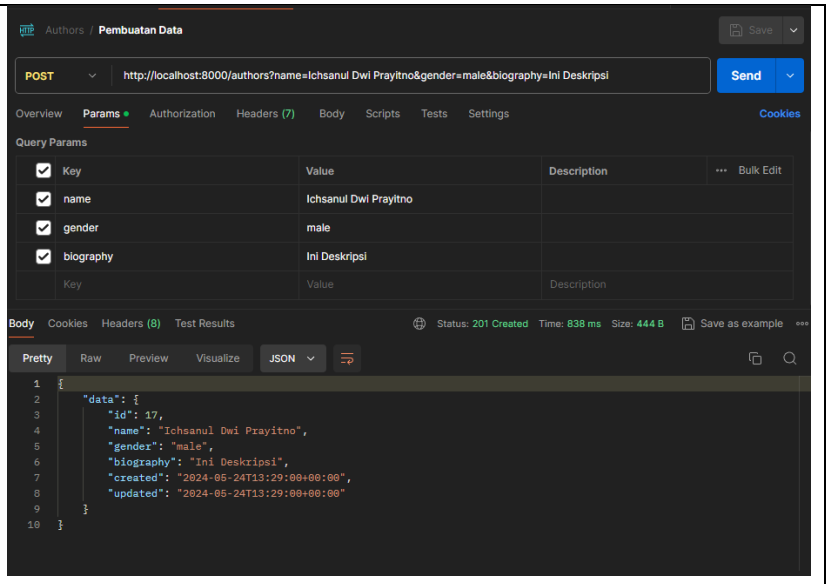


Hasil CRUD /author

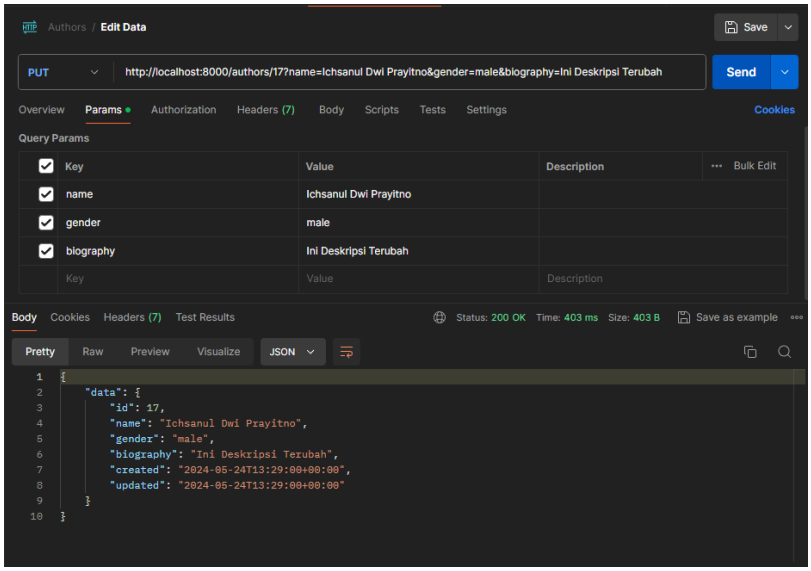
Hasil query Read / Pembacaan seluruh data



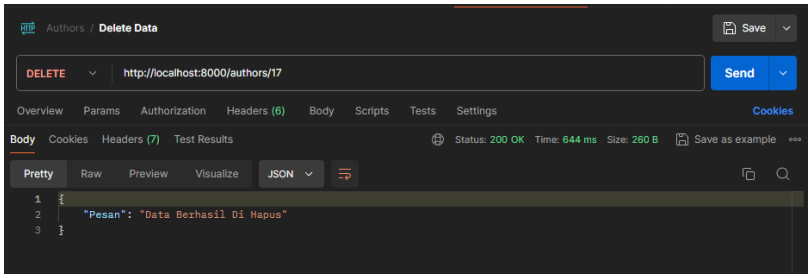
Hasil query Post / Pembuatan data



Hasil query Put / Pengeditan data



Hasil query Del / Penghapusan data



TEORI

I. Lembar Soal

N/A

II. Lembar Kerja

N/A