ARSITEKTUR BERBASIS LAYANAN

“Monolitics vs Microservice vs SOA “

Dibuat Oleh :

AKBAR HIDAYATULLAH

2311082004

TRPL 3A

D4 TEKNOLOGI REKAYASA PERANGKAT LUNAK

TEKNOLOGI INFORMASI

POLITEKNIK NEGERI PADANG

2025

LAPORAN PRAKTIKUM

Pengenalan Arsitektur Microservice dengan Laravel

---

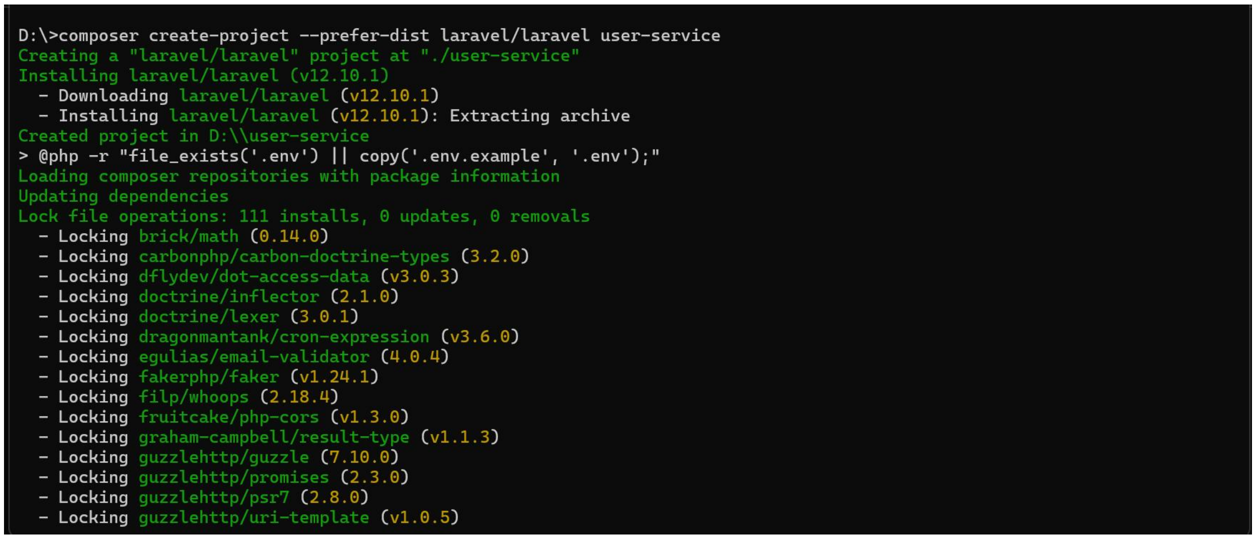
1. Pendahuluan

Arsitektur microservice merupakan pendekatan pengembangan perangkat lunak yang memecah aplikasi besar menjadi beberapa service kecil yang berdiri sendiri. Pada praktikum ini dilakukan pembuatan dua service terpisah berbasis Laravel, yaitu **User Service** sebagai API Provider dan **Order Service** sebagai API Consumer. Kedua service berkomunikasi menggunakan REST API.

2. Langkah Kerja

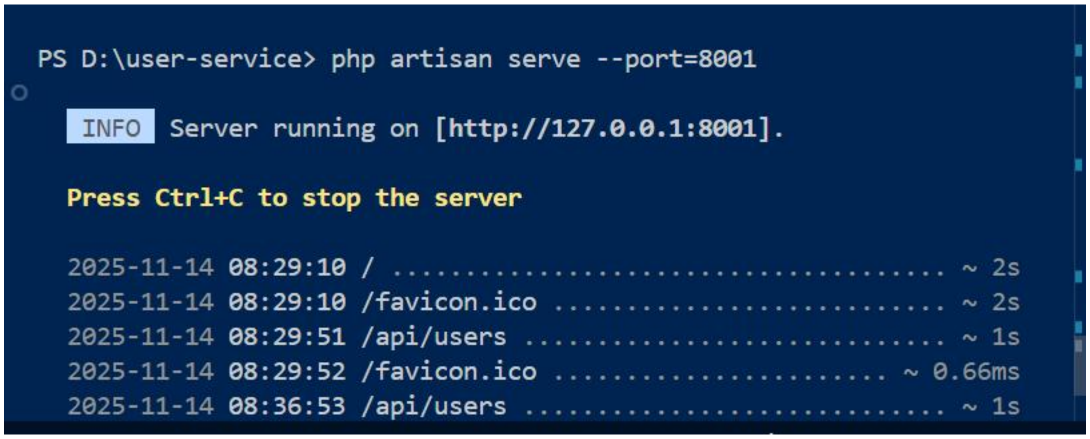
2.1 Membuat User Service (API Provider)

1. Membuat proyek Laravel baru:
2. composer create-project laravel/laravel user-service



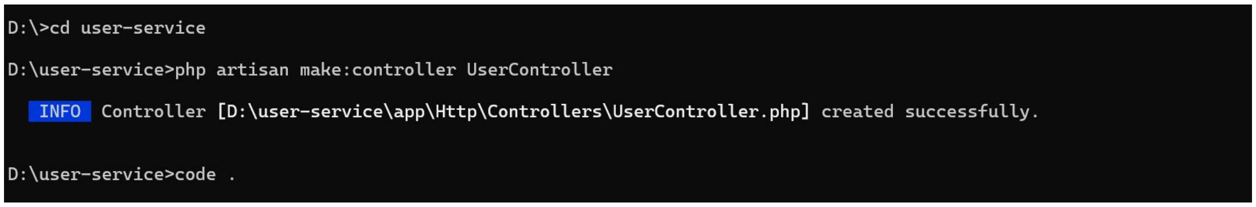
```
D:\>composer create-project --prefer-dist laravel/laravel user-service
Creating a "laravel/laravel" project at "./user-service"
Installing laravel/laravel (v12.10.1)
  - Downloading laravel/laravel (v12.10.1)
  - Installing laravel/laravel (v12.10.1): Extracting archive
Created project in D:\\user-service
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
  - Locking brick/math (0.14.0)
  - Locking carbonphp/carbon-doctrine-types (3.2.0)
  - Locking dflydev/dot-access-data (v3.0.3)
  - Locking doctrine/inflector (2.1.0)
  - Locking doctrine/lexer (3.0.1)
  - Locking dragonmantank/cron-expression (v3.6.0)
  - Locking egulias/email-validator (4.0.4)
  - Locking fakerphp/faker (v1.24.1)
  - Locking filp/whoops (2.18.4)
  - Locking fruitcake/php-cors (v1.3.0)
  - Locking graham-campbell/result-type (v1.1.3)
  - Locking guzzlehttp/guzzle (7.10.0)
  - Locking guzzlehttp/promises (2.3.0)
  - Locking guzzlehttp/psr7 (2.8.0)
  - Locking guzzlehttp/uri-template (v1.0.5)
```

3. Menjalankan server pada port 8001:



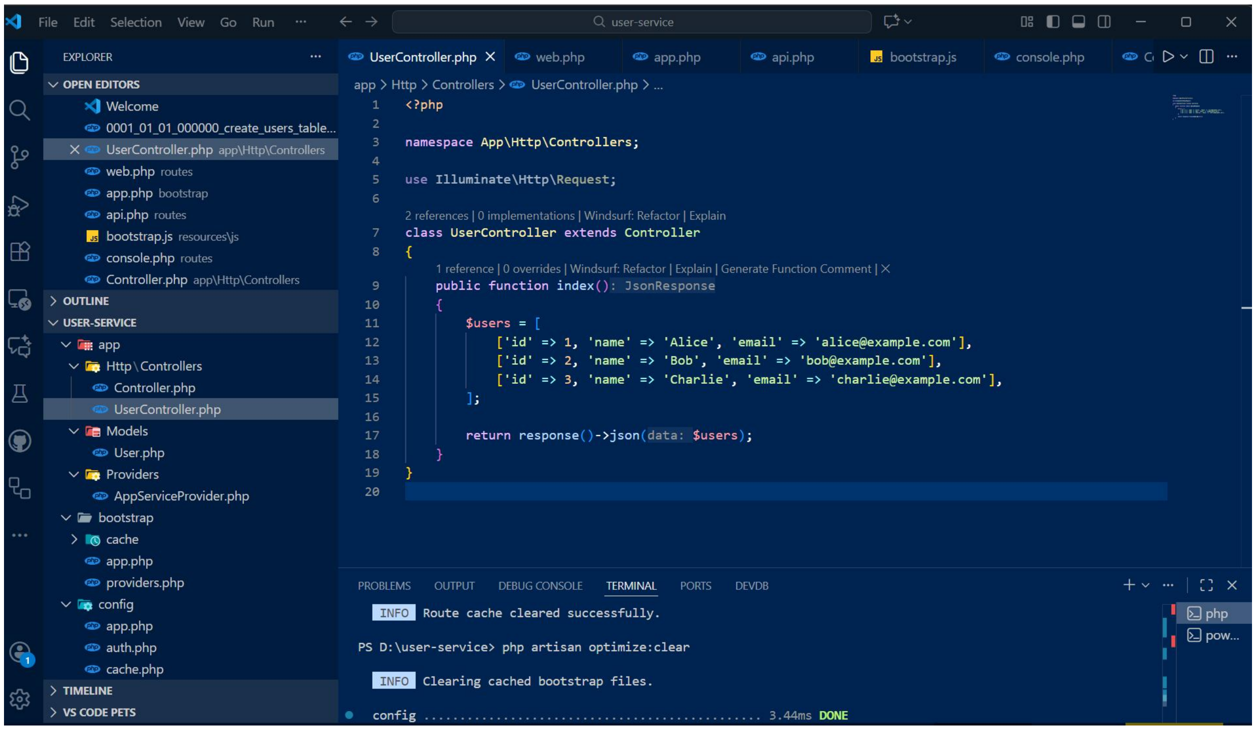
```
PS D:\user-service> php artisan serve --port=8001

  INFO  Server running on [http://127.0.0.1:8001].

  Press Ctrl+C to stop the server

  2025-11-14 08:29:10 / ................................... ~ 2s
  2025-11-14 08:29:10 /favicon.ico .......................... ~ 2s
  2025-11-14 08:29:51 /api/users ............................ ~ 1s
  2025-11-14 08:29:52 /favicon.ico ...................... ~ 0.66ms
  2025-11-14 08:36:53 /api/users ............................ ~ 1s
```

4. php artisan serve --port=8001
5. Membuat controller:
6. php artisan make:controller UserController



```
D:\>cd user-service

D:\user-service>php artisan make:controller UserController

  INFO  Controller [D:\user-service\app\Http\Controllers\UserController.php] created successfully.

D:\user-service>code .
```

7. Menambahkan kode pada UserController untuk menghasilkan data user dalam format JSON.

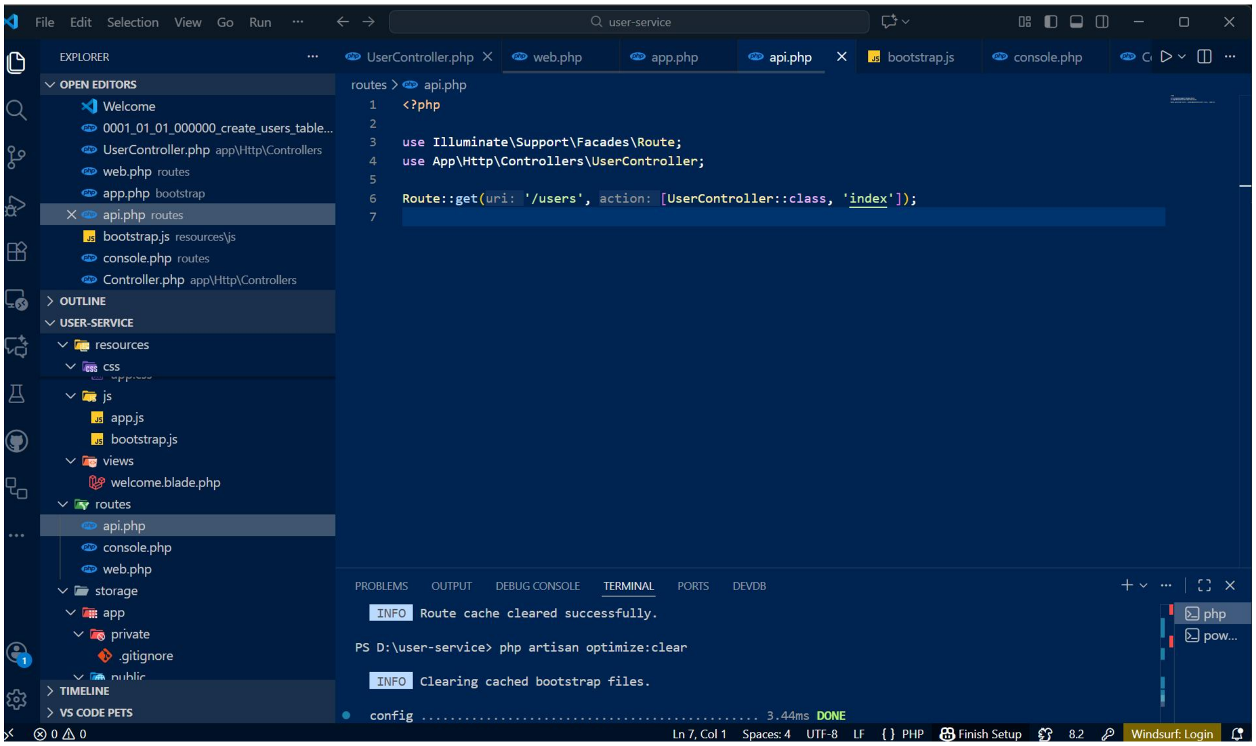

File Edit Selection View Go Run ...
user-service
EXPLORER
OPEN EDITORS
Welcome
0001_01_01_000000_create_users_table...
UserController.php app\Http\Controllers
web.php routes
app.php bootstrap
api.php routes
bootstrap.js resources\js
console.php routes
Controller.php app\Http\Controllers
OUTLINE
USER-SERVICE
app
Http \ Controllers
Controller.php
UserController.php
Models
User.php
Providers
AppServiceProvider.php
bootstrap
cache
app.php
providers.php
config
app.php
auth.php
cache.php
TIMELINE
VS CODE PETS
UserController.php web.php app.php api.php bootstrap.js console.php
app > Http > Controllers > UserController.php > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDB
php
pow...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

2 references | 0 implementations | Windsurf: Refactor | Explain
class UserController extends Controller
{
    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment | X
    public function index(): JsonResponse
    {
        $users = [
            ['id' => 1, 'name' => 'Alice', 'email' => 'alice@example.com'],
            ['id' => 2, 'name' => 'Bob', 'email' => 'bob@example.com'],
            ['id' => 3, 'name' => 'Charlie', 'email' => 'charlie@example.com'],
        ];

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

```
  INFO  Route cache cleared successfully.

PS D:\user-service> php artisan optimize:clear

  INFO  Clearing cached bootstrap files.

  config ............................................... 3.44ms DONE
```

8. Menambahkan route pada routes/api.php:

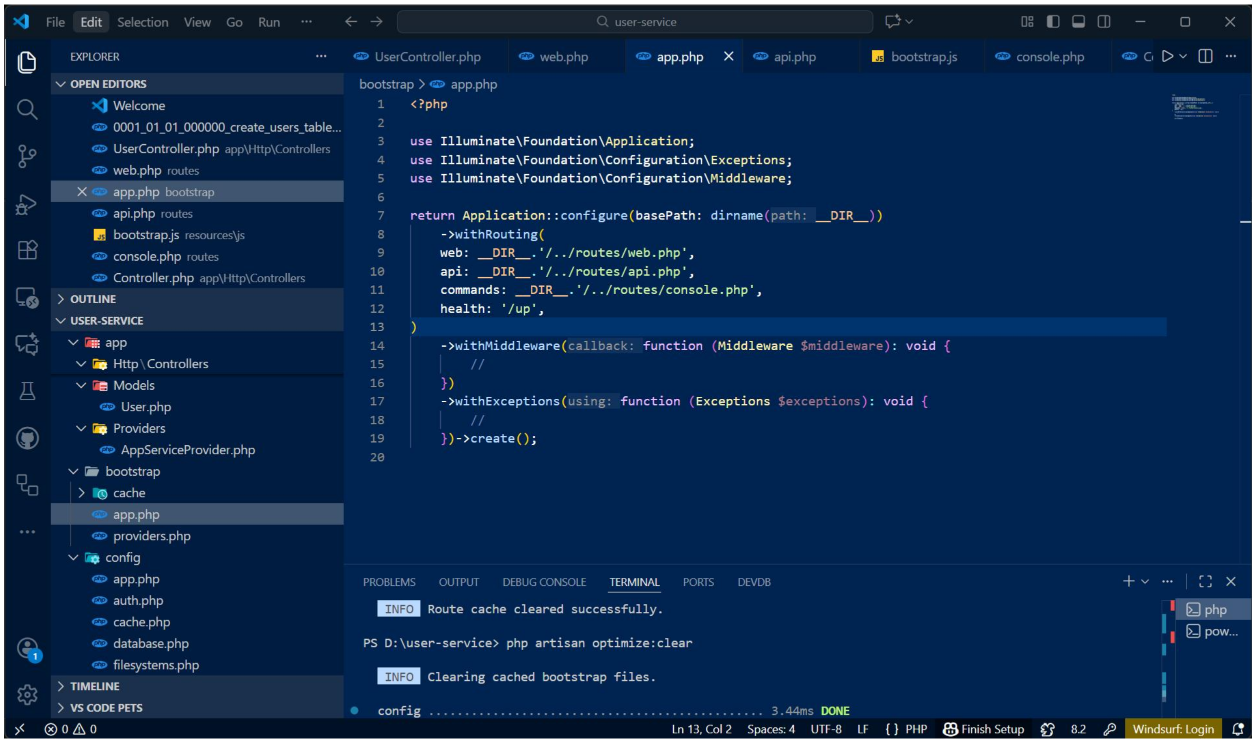
   Route::get('/users', [UserController::class, 'index']);



```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\UserController;

Route::get(uri: '/users', action: [UserController::class, 'index']);
```

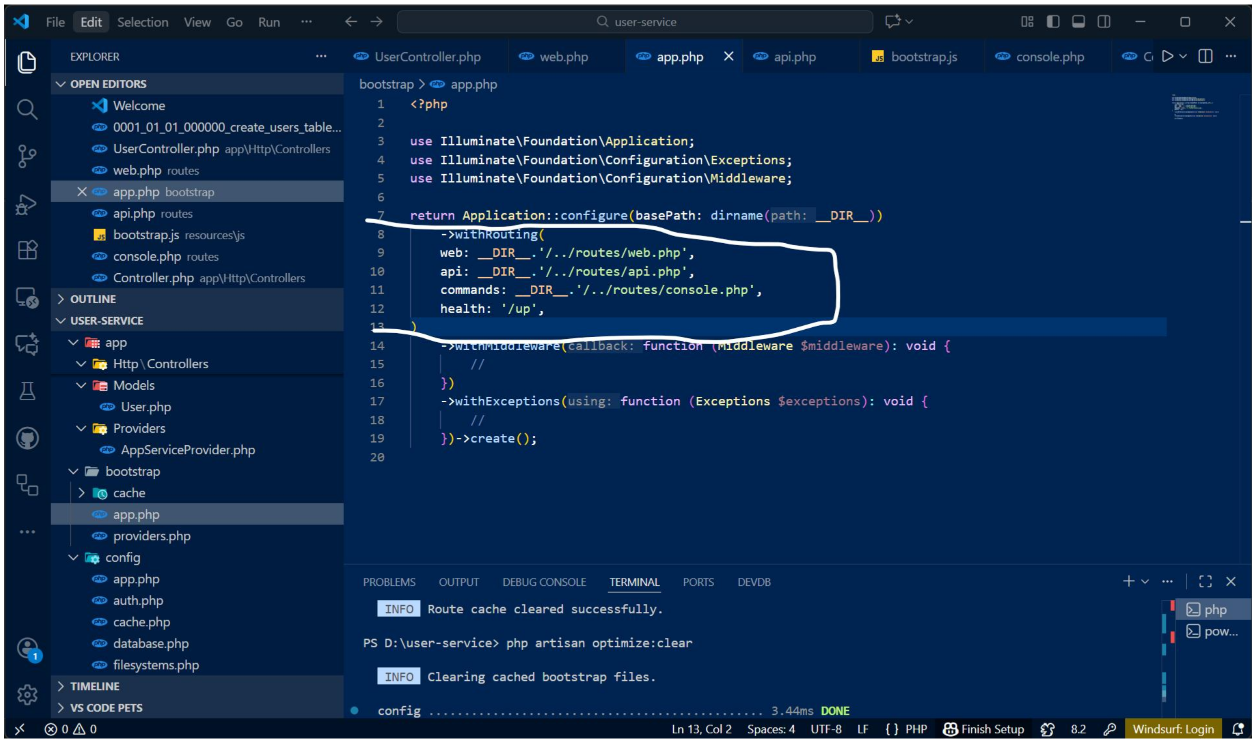
```
 INFO  Route cache cleared successfully.

PS D:\user-service> php artisan optimize:clear

 INFO  Clearing cached bootstrap files.

config ............................................... 3.44ms DONE
```

9. Mengaktifkan routing API di Laravel 11 melalui bootstrap/app.php.



```php
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(path: __DIR__))
    ->withRouting(
    web: __DIR__.'/../routes/web.php',
    api: __DIR__.'/../routes/api.php',
    commands: __DIR__.'/../routes/console.php',
    health: '/up',
)
    ->withMiddleware(callback: function (Middleware $middleware): void {
        //
    })
    ->withExceptions(using: function (Exceptions $exceptions): void {
        //
    })->create();
```

```
 INFO  Route cache cleared successfully.

PS D:\user-service> php artisan optimize:clear

 INFO  Clearing cached bootstrap files.

config ............................................... 3.44ms DONE
```

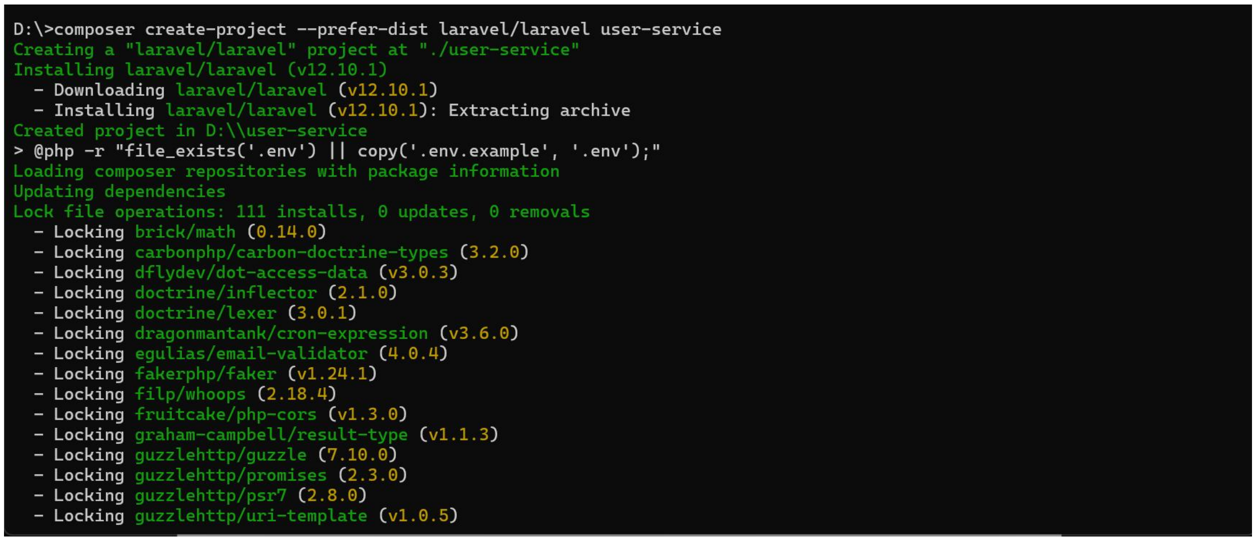


```
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(path: __DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(callback: function (Middleware $middleware): void {
        //
    })
    ->withExceptions(using: function (Exceptions $exceptions): void {
        //
    })->create();
```

```
 INFO  Route cache cleared successfully.

PS D:\user-service> php artisan optimize:clear

 INFO  Clearing cached bootstrap files.

 config ............................................. 3.44ms DONE
```

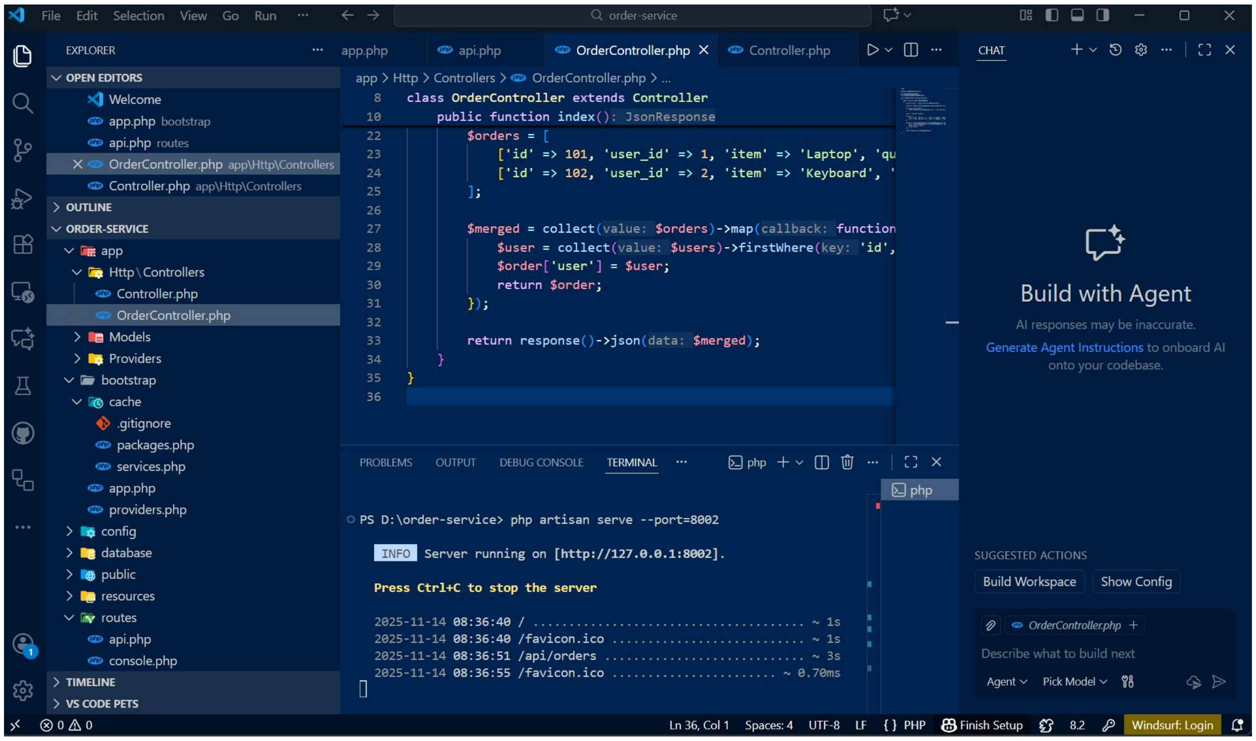
3.2 Membuat Order Service (API Consumer)

1. Membuat proyek Laravel baru:
2. composer create-project laravel/laravel order-service



```
D:\>composer create-project --prefer-dist laravel/laravel user-service
Creating a "laravel/laravel" project at "./user-service"
Installing laravel/laravel (v12.10.1)
  - Downloading laravel/laravel (v12.10.1)
  - Installing laravel/laravel (v12.10.1): Extracting archive
Created project in D:\\user-service
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
  - Locking brick/math (0.14.0)
  - Locking carbonphp/carbon-doctrine-types (3.2.0)
  - Locking dflydev/dot-access-data (v3.0.3)
  - Locking doctrine/inflector (2.1.0)
  - Locking doctrine/lexer (3.0.1)
  - Locking dragonmantank/cron-expression (v3.6.0)
  - Locking egulias/email-validator (4.0.4)
  - Locking fakerphp/faker (v1.24.1)
  - Locking filp/whoops (2.18.4)
  - Locking fruitcake/php-cors (v1.3.0)
  - Locking graham-campbell/result-type (v1.1.3)
  - Locking guzzlehttp/guzzle (7.10.0)
  - Locking guzzlehttp/promises (2.3.0)
  - Locking guzzlehttp/psr7 (2.8.0)
  - Locking guzzlehttp/uri-template (v1.0.5)
```

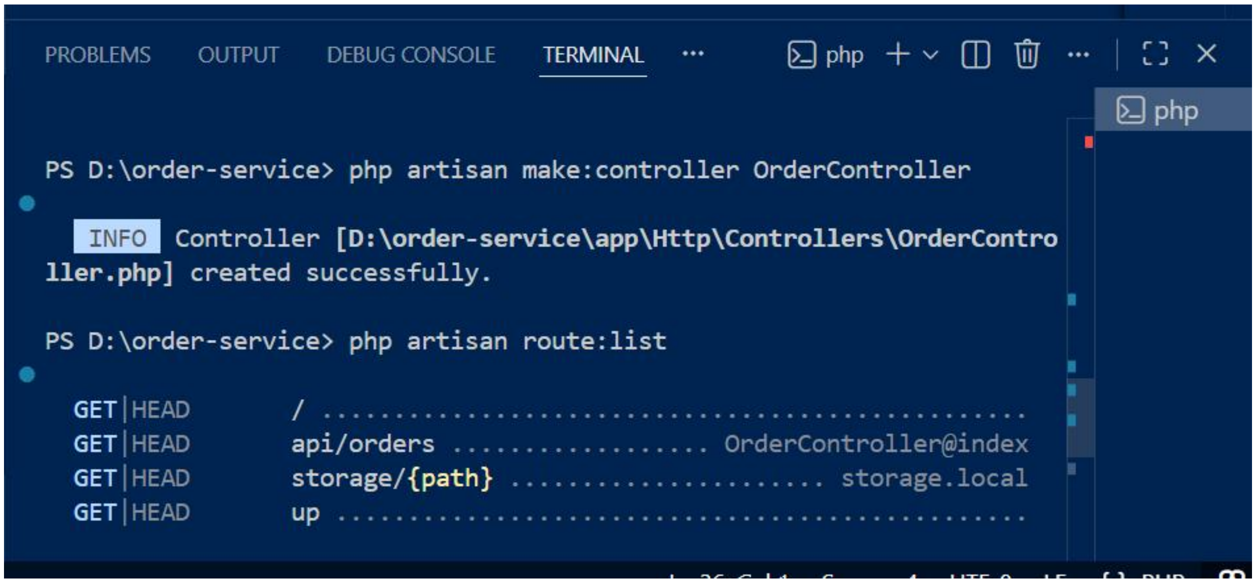
3. Menjalankan server pada port 8002:



```php
class OrderController extends Controller
    public function index(): JsonResponse
        $orders = [
            ['id' => 101, 'user_id' => 1, 'item' => 'Laptop', 'qu
            ['id' => 102, 'user_id' => 2, 'item' => 'Keyboard', '
        ];

        $merged = collect(value: $orders)->map(callback: function
            $user = collect(value: $users)->firstWhere(key: 'id',
            $order['user'] = $user;
            return $order;
        });

        return response()->json(data: $merged);
    }
}
```

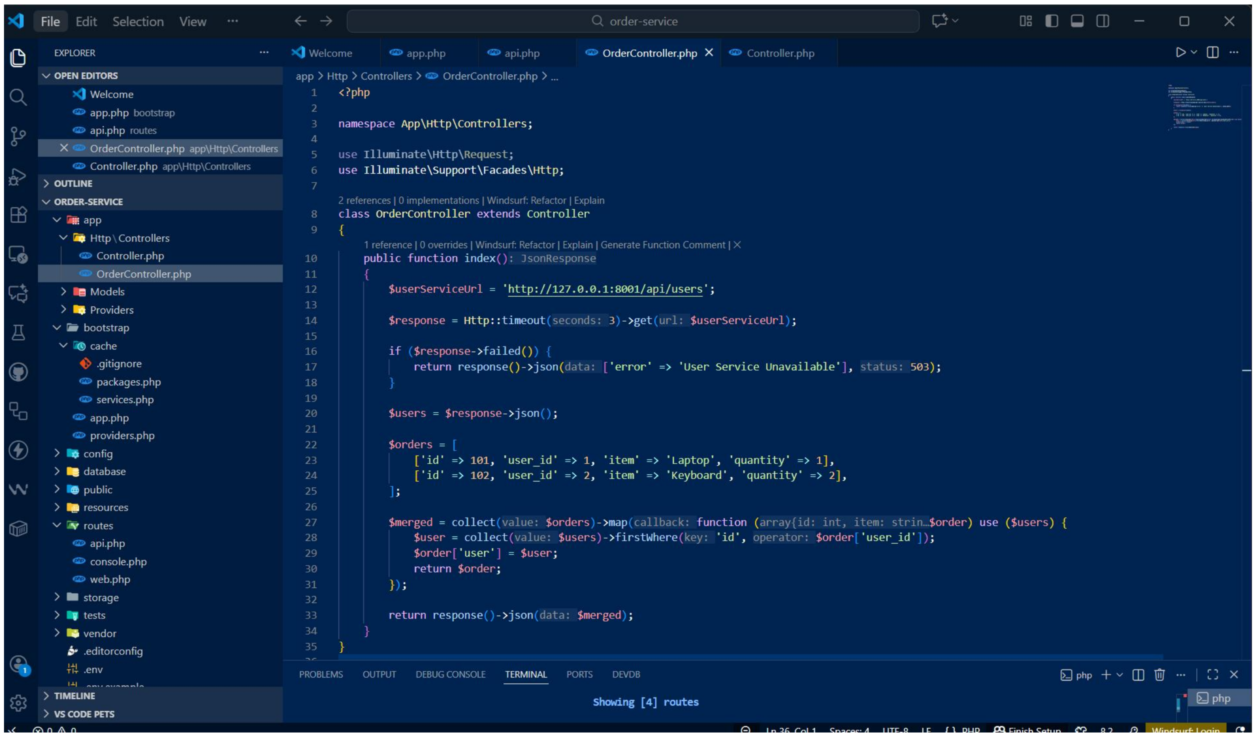
```
PS D:\order-service> php artisan serve --port=8002

  INFO  Server running on [http://127.0.0.1:8002].

  Press Ctrl+C to stop the server

  2025-11-14 08:36:40 / ..................................... ~ 1s
  2025-11-14 08:36:40 /favicon.ico .......................... ~ 1s
  2025-11-14 08:36:51 /api/orders ........................... ~ 3s
  2025-11-14 08:36:55 /favicon.ico ...................... ~ 0.70ms
```

4. php artisan serve --port=8002
5. Membuat controller:
6. php artisan make:controller OrderController



```
PS D:\order-service> php artisan make:controller OrderController

  INFO  Controller [D:\order-service\app\Http\Controllers\OrderContro
ller.php] created successfully.

PS D:\order-service> php artisan route:list

  GET|HEAD       / .............................................
  GET|HEAD       api/orders .................. OrderController@index
  GET|HEAD       storage/{path} ..................... storage.local
  GET|HEAD       up ............................................
```

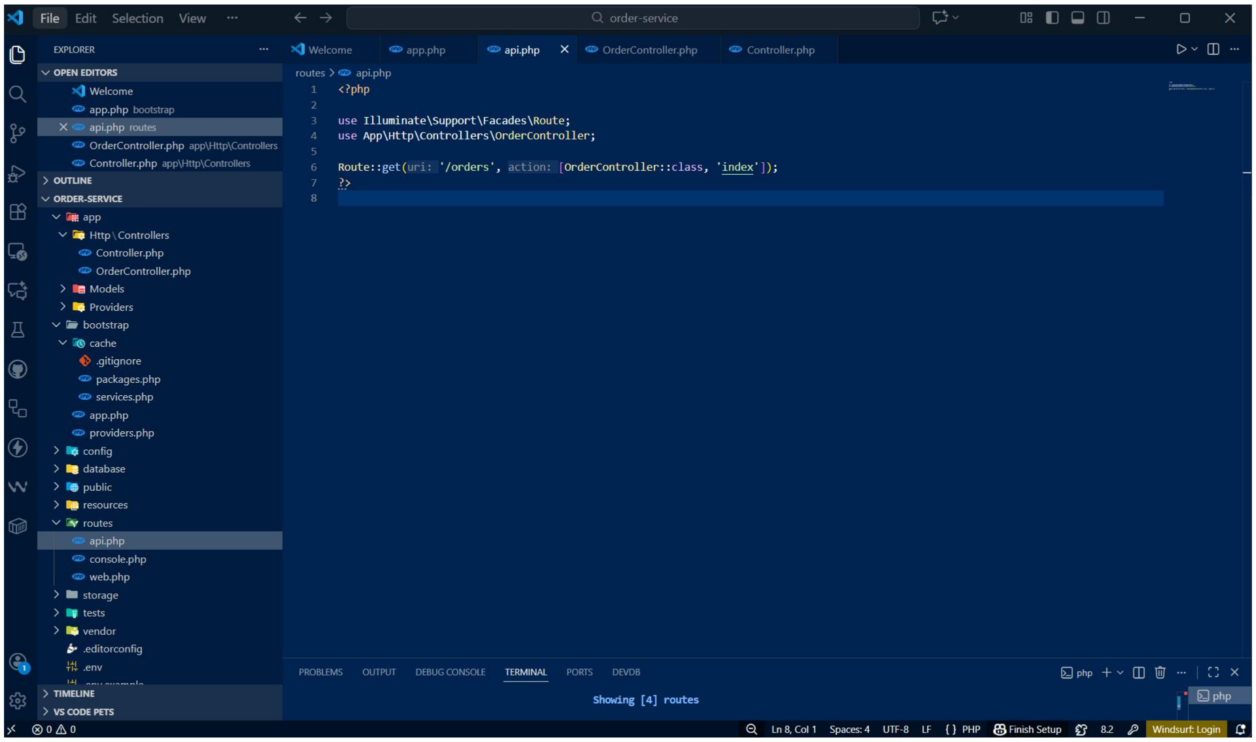
7. Menambahkan kode untuk mengambil data user dari User Service menggunakan HTTP Client Laravel.



```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;

2 references | 0 implementations | Windsurf: Refactor | Explain
class OrderController extends Controller
{
    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment | X
    public function index(): JsonResponse
    {
        $userServiceUrl = 'http://127.0.0.1:8001/api/users';

        $response = Http::timeout(seconds: 3)->get(url: $userServiceUrl);

        if ($response->failed()) {
            return response()->json(data: ['error' => 'User Service Unavailable'], status: 503);
        }

        $users = $response->json();

        $orders = [
            ['id' => 101, 'user_id' => 1, 'item' => 'Laptop', 'quantity' => 1],
            ['id' => 102, 'user_id' => 2, 'item' => 'Keyboard', 'quantity' => 2],
        ];

        $merged = collect(value: $orders)->map(callback: function (array{id: int, item: strin…$order) use ($users) {
            $user = collect(value: $users)->firstWhere(key: 'id', operator: $order['user_id']);
            $order['user'] = $user;
            return $order;
        });

        return response()->json(data: $merged);
    }
}
```

```
Showing [4] routes
```

8. Menambahkan route pada routes/api.php:



```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\OrderController;

Route::get(uri: '/orders', action: [OrderController::class, 'index']);
?>
```

```
Showing [4] routes
```

9. Route::get('/orders', [OrderController::class, 'index']);

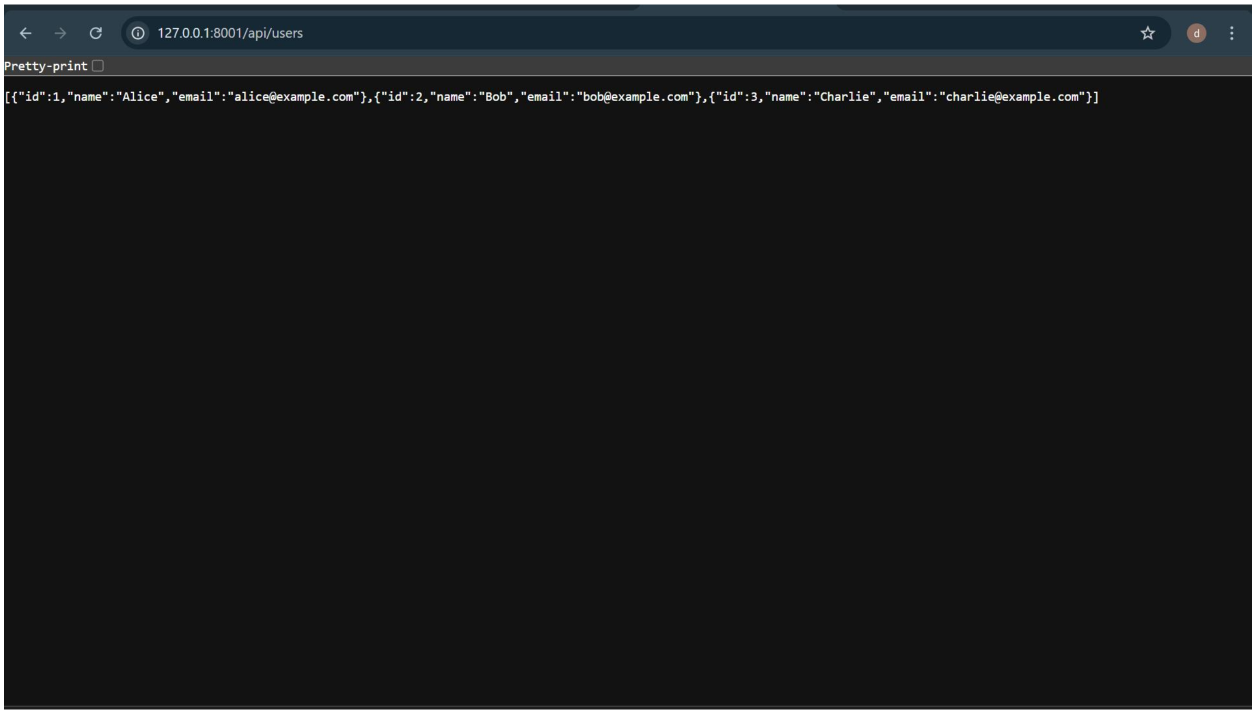
10. Menggabungkan data user dan order sehingga menghasilkan output JSON sesuai praktikum.

4. Hasil Pengujian

Hasil pengujian User Service (API Provider):

Endpoint: http://127.0.0.1:8001/api/users

Output:

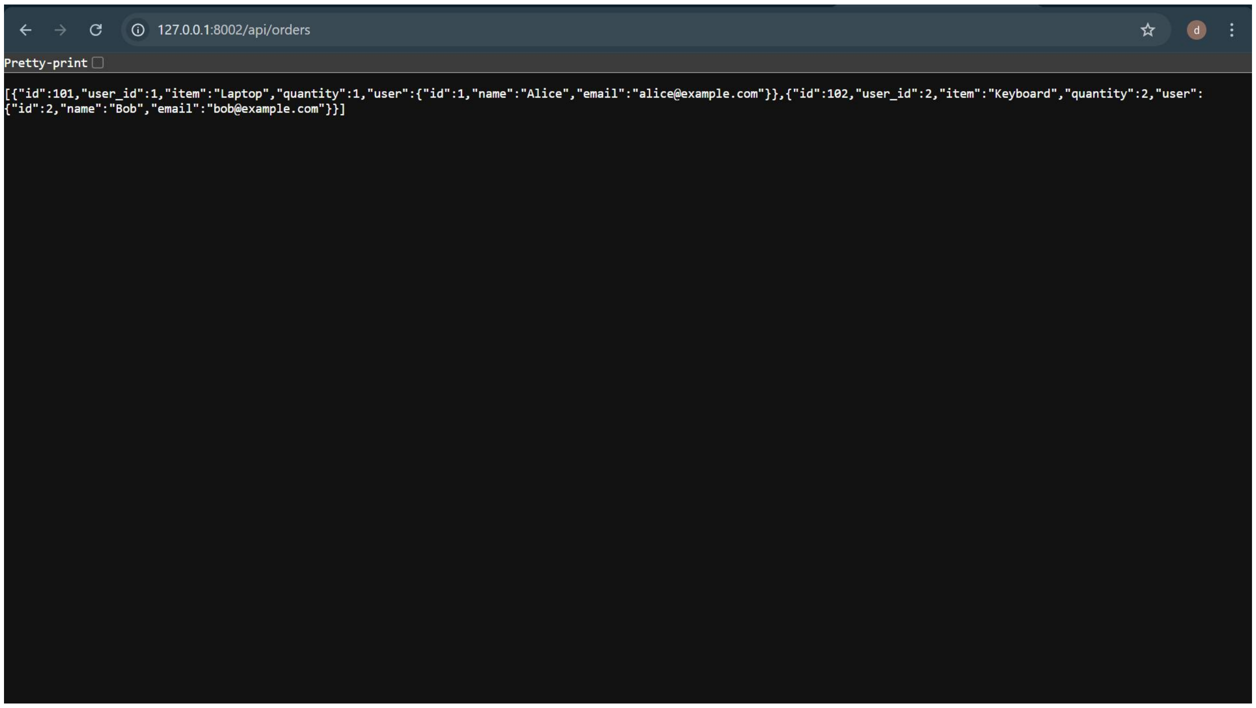


```
[{"id":1,"name":"Alice","email":"alice@example.com"},{"id":2,"name":"Bob","email":"bob@example.com"},{"id":3,"name":"Charlie","email":"charlie@example.com"}]
```

Hasil pengujian Order Service (API Consumer):

Endpoint: http://127.0.0.1:8002/api/orders

Output:



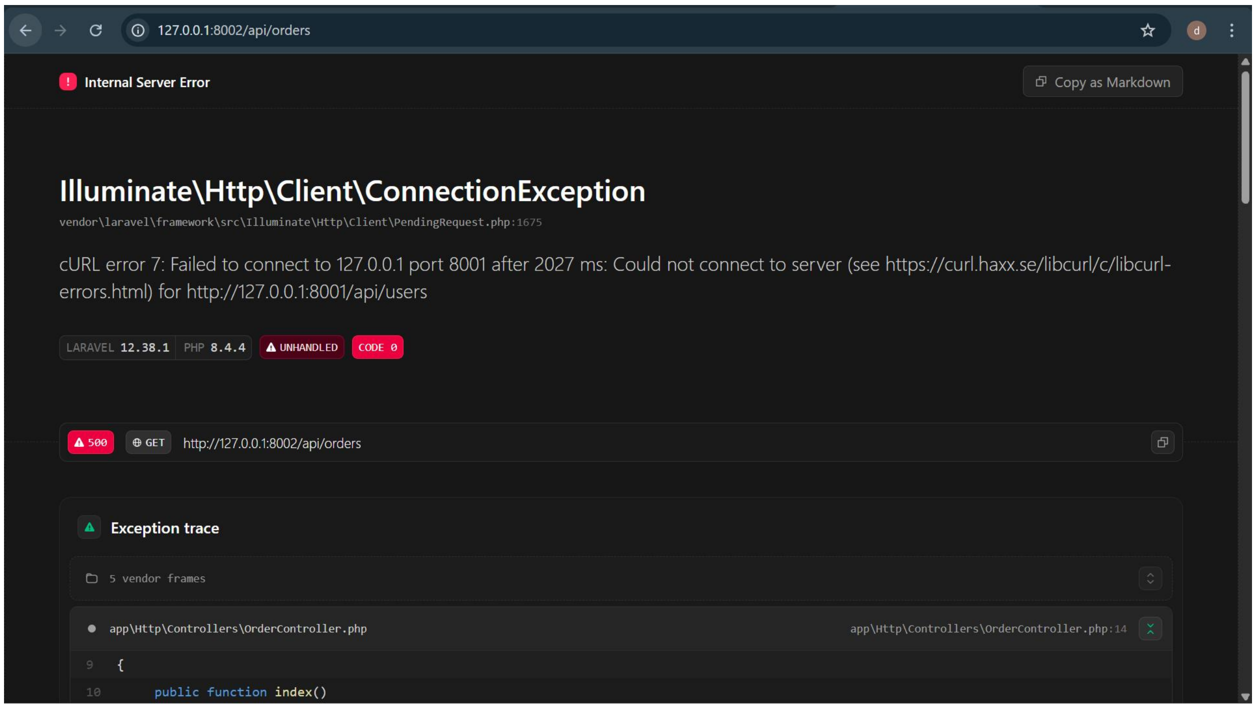
```
[{"id":101,"user_id":1,"item":"Laptop","quantity":1,"user":{"id":1,"name":"Alice","email":"alice@example.com"}},{"id":102,"user_id":2,"item":"Keyboard","quantity":2,"user":
{"id":2,"name":"Bob","email":"bob@example.com"}}]
```

---

5. Jawaban Refleksi dan Diskusi

1. Apa yang terjadi jika User Service tidak aktif?

Jika User Service tidak aktif, Order Service tidak dapat mengambil data user sehingga proses penggabungan data gagal. Order Service akan memberikan pesan error "User Service Unavailable" karena service yang dituju tidak merespons.



Illuminate\Http\Client\ConnectionException

```
vendor\laravel\framework\src\Illuminate\Http\Client\PendingRequest.php:1675
```

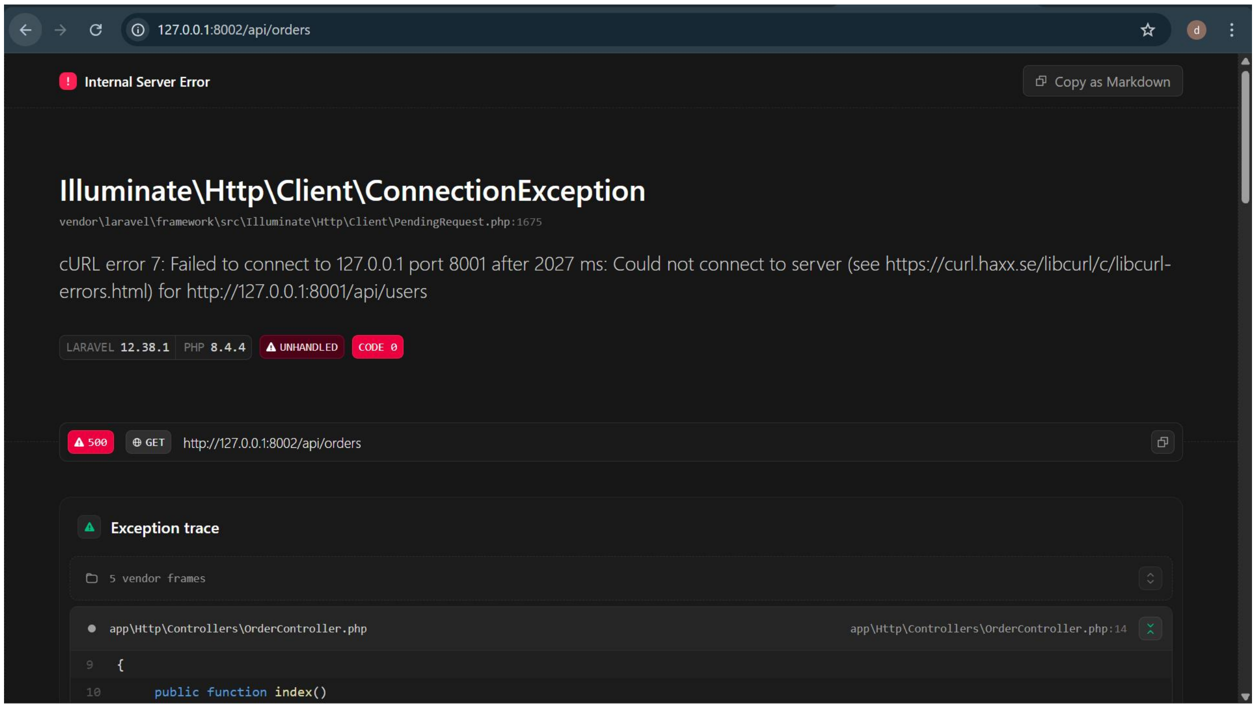
```
cURL error 7: Failed to connect to 127.0.0.1 port 8001 after 2027 ms: Could not connect to server (see https://curl.haxx.se/libcurl/c/libcurl-errors.html) for http://127.0.0.1:8001/api/users
```

```
app\Http\Controllers\OrderController.php                app\Http\Controllers\OrderController.php:14
{
    public function index()
```

2. Bagaimana cara mengatasi kegagalan komunikasi antar service?

Beberapa cara untuk mengatasi kegagalan komunikasi:

- Menggunakan **timeout** dan **error handling** agar service tidak menunggu terlalu lama.
- Menggunakan **retry mechanism** untuk mencoba kembali request yang gagal.
- Menggunakan **circuit breaker** untuk memutus request ke service yang sedang bermasalah.
- Menggunakan **caching** untuk menyediakan data sementara jika service lain tidak responsif.
- Menyediakan **fallback response** agar sistem tetap dapat berjalan.



```
Illuminate\Http\Client\ConnectionException
vendor\laravel\framework\src\Illuminate\Http\Client\PendingRequest.php:1675

cURL error 7: Failed to connect to 127.0.0.1 port 8001 after 2027 ms: Could not connect to server (see https://curl.haxx.se/libcurl/c/libcurl-errors.html) for http://127.0.0.1:8001/api/users

LARAVEL 12.38.1 | PHP 8.4.4 | UNHANDLED | CODE 0

500  GET  http://127.0.0.1:8002/api/orders

Exception trace
5 vendor frames
app\Http\Controllers\OrderController.php          app\Http\Controllers\OrderController.php:14
{
    public function index()
```

3. Apa keuntungan dan tantangan microservice dibandingkan monolitik?

Keuntungan microservice:

- Dapat dikembangkan dan dideploy secara terpisah.
- Skalabilitas lebih mudah dan fleksibel.
- Isolasi kesalahan lebih baik (jika satu service rusak, yang lain tetap berjalan).

Tantangan microservice:

- Arsitektur lebih kompleks dan membutuhkan banyak konfigurasi.
- Komunikasi antar service rentan mengalami error.
- Debugging lebih sulit karena banyak service yang saling terhubung.
- Membutuhkan sistem monitoring dan DevOps yang baik.