|  |  |  |
| --- | --- | --- |
| Subject | **:** | ADVANCE WEB PROGRAMMING |
| Program Studi | **:** | D4 – Informatics Engineering **/** D4 – Business Information Systems |
| Semester | **:** | 4 (four) / 6 (six) |
| Meeting to- | **:** | 11 (eleven) |

# JOBSHEET 11

**RESTFUL API 2**

Before we enter the material, we first create a new project that we will use to build a simple application with the topic *of Point of Sales (PoS),* according to

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik *Point of Sales (PoS),* sesuai dengan **Studi Kasus PWL.pdf**.

Jadi kita bikin project Laravel 10 dengan nama **PWL\_POS.**

*Project* **PWL\_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

# A. ELOQUENT ACCESSOR

Laravel has features called mutator, accessor and casting, these features are used to manipulate data in database attributes very easily. For example, when inserting data with encryption into the database and doing descriptions when displaying from the database automatically.

The accessor can change the value when the attribute or eloquent field is accessed. To define the Accessor can create a method inside the model to specify the attributes to be accessed. The name of the method created must be the same as the name of the attribute to be formatted: example:

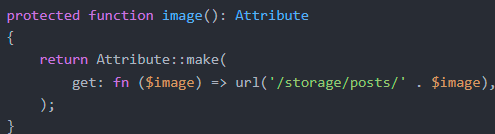
Attribute/field in table first\_name then method firstName() protected function firstName(): Attribute

{

//...

}

If we create an existing image attribute/field in the m\_user we will provide the full path value of the directory where the image file is stored.



That way you can import Eloquent Attributes with

Then a new method with the name image() returns the path of the image file name it is located in



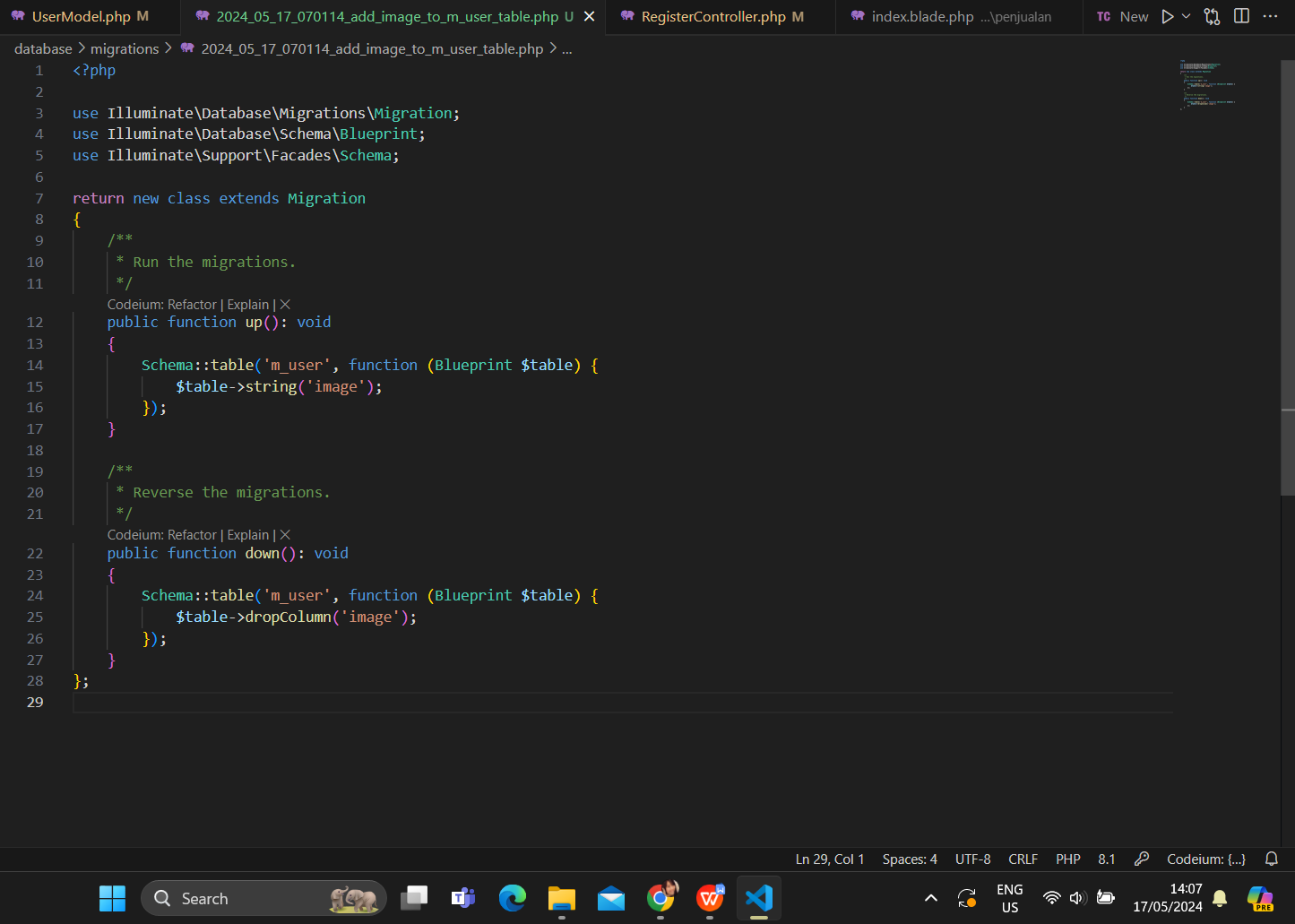
The final result calls the image attribute

**Practicum 1** – Eloquent Accessor Implementation

1. Before starting make sure the REST API, first make sure the Postman application is installed.
2. We will modify the Table m\_user by adding column: image, open terminal and type

**php artisan make:migration add\_image\_to\_m\_user\_table**





<?php

use Illuminate\Database\Migrations\Migration; use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

};

/\*\*

* Run the migrations.

\*/

public function up(): void

{

Schema::table('m\_user', function (Blueprint $table) {

$table->string('image');

});

}

/\*\*

* Reverse the migrations.

\*/

public function down(): void

{

Schema::table('m\_user', function (Blueprint $table) {

$table->dropColumn('image');

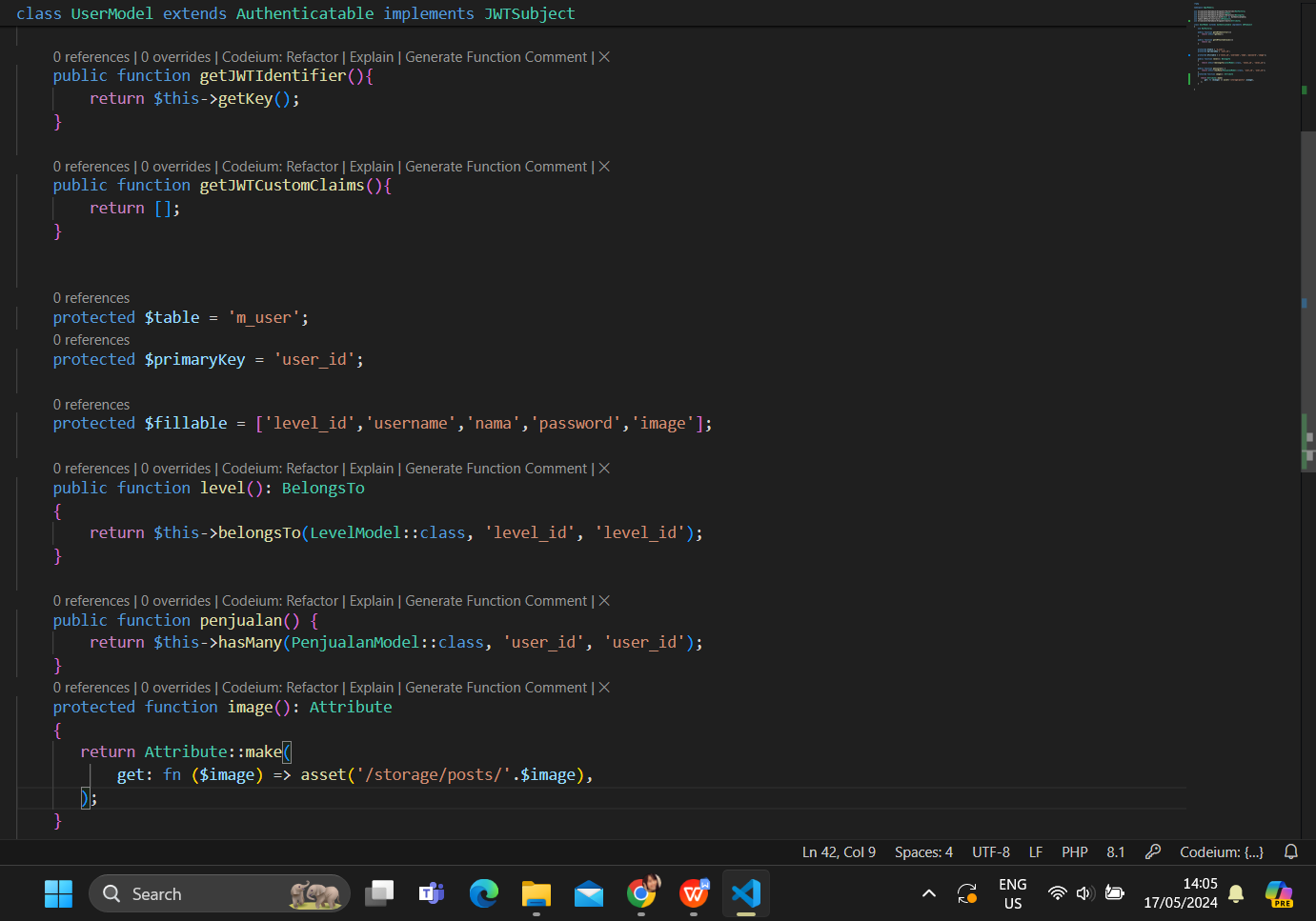
});

}

{

1. Run the migration update by:

php artisan migrate



<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model; use Tymon\JWTAuth\Contracts\JWTSubject;

use Illuminate\Database\Eloquent\Casts\Attribute;

use Illuminate\Foundation\Auth\User as Authenticatable;

class UserModel extends Authenticatable implements JWTSubject

{

public function getJWTIdentifier(){ return $this->getKey();

}

<?php

namespace App\Http\Controllers\Api; use App\Models\UserModel;

use App\Http\Controllers\Controller; use Illuminate\Http\Request;

use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller

{

public function invoke(Request $request)

{

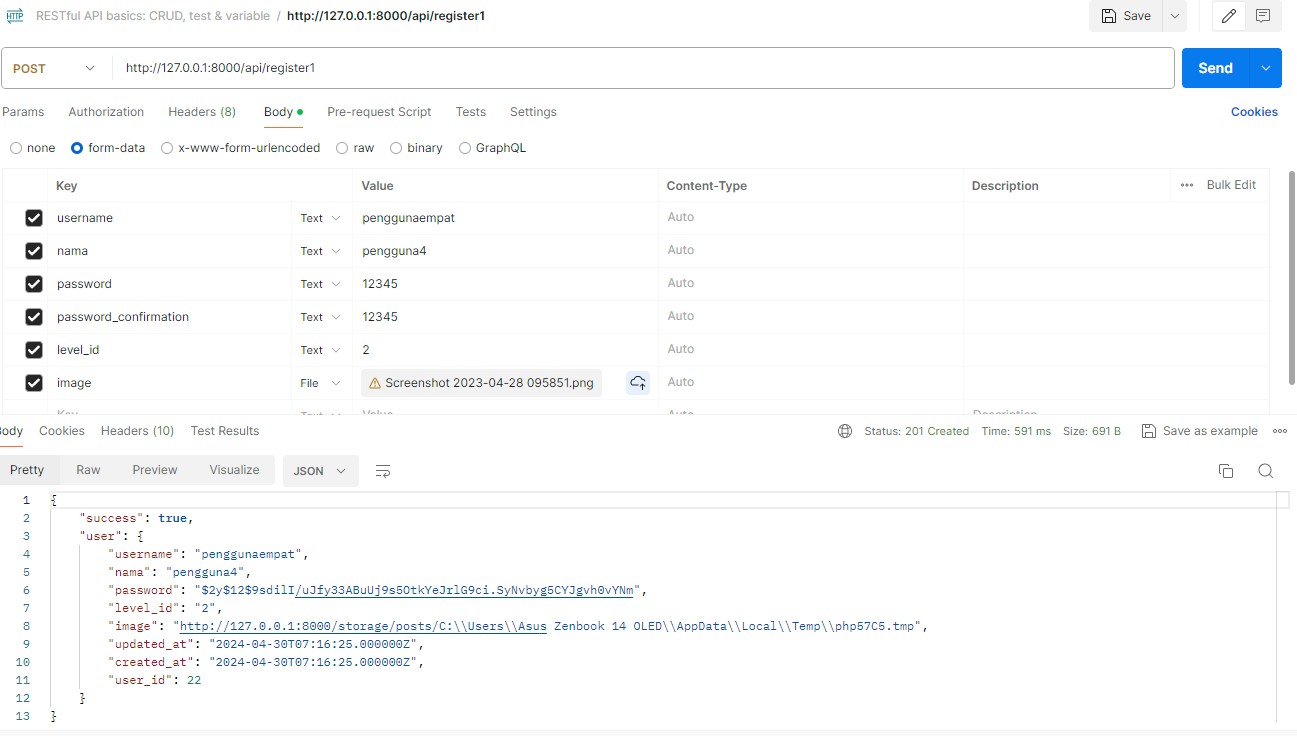
//set validation

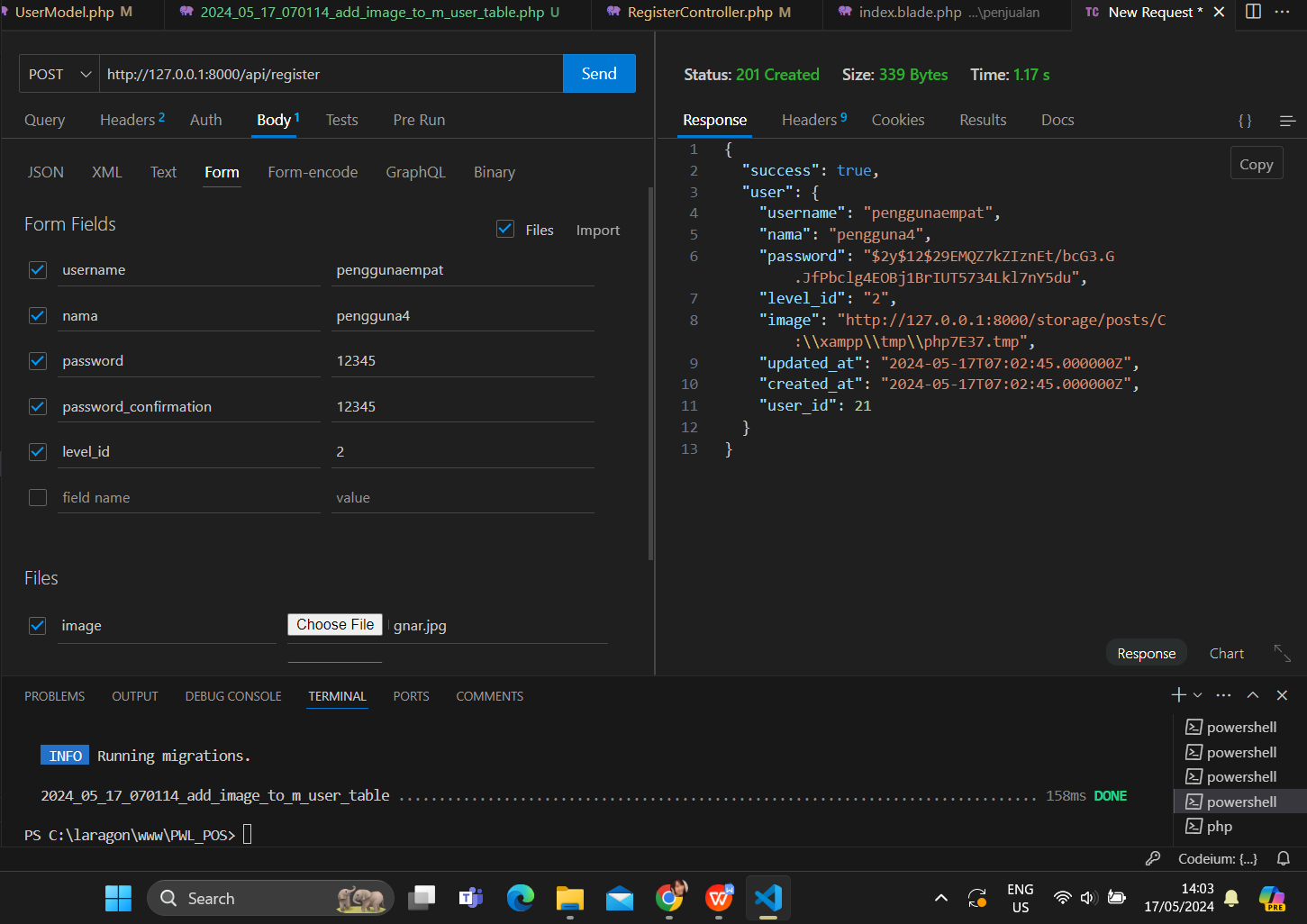
1. You can add details for the image spec to the validator
2. Change or add register1 to routes/api.php



1. Save and access it in the Postman application, set it in the Body fill in the manual Key

and the Value in the Key image add the File value and upload

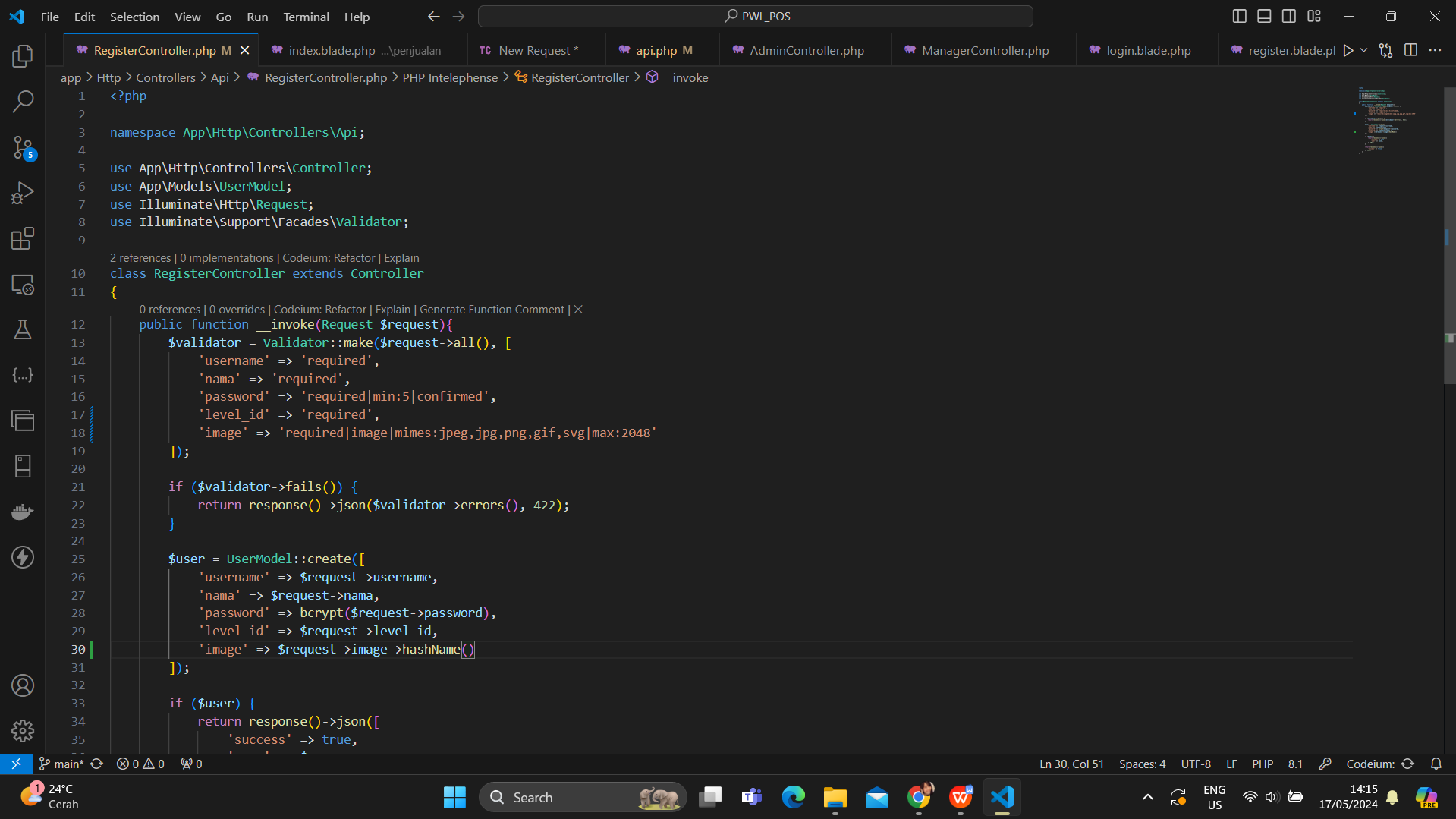
the http://127.0.0.1:8000/api/register1 image with the POST method and click send

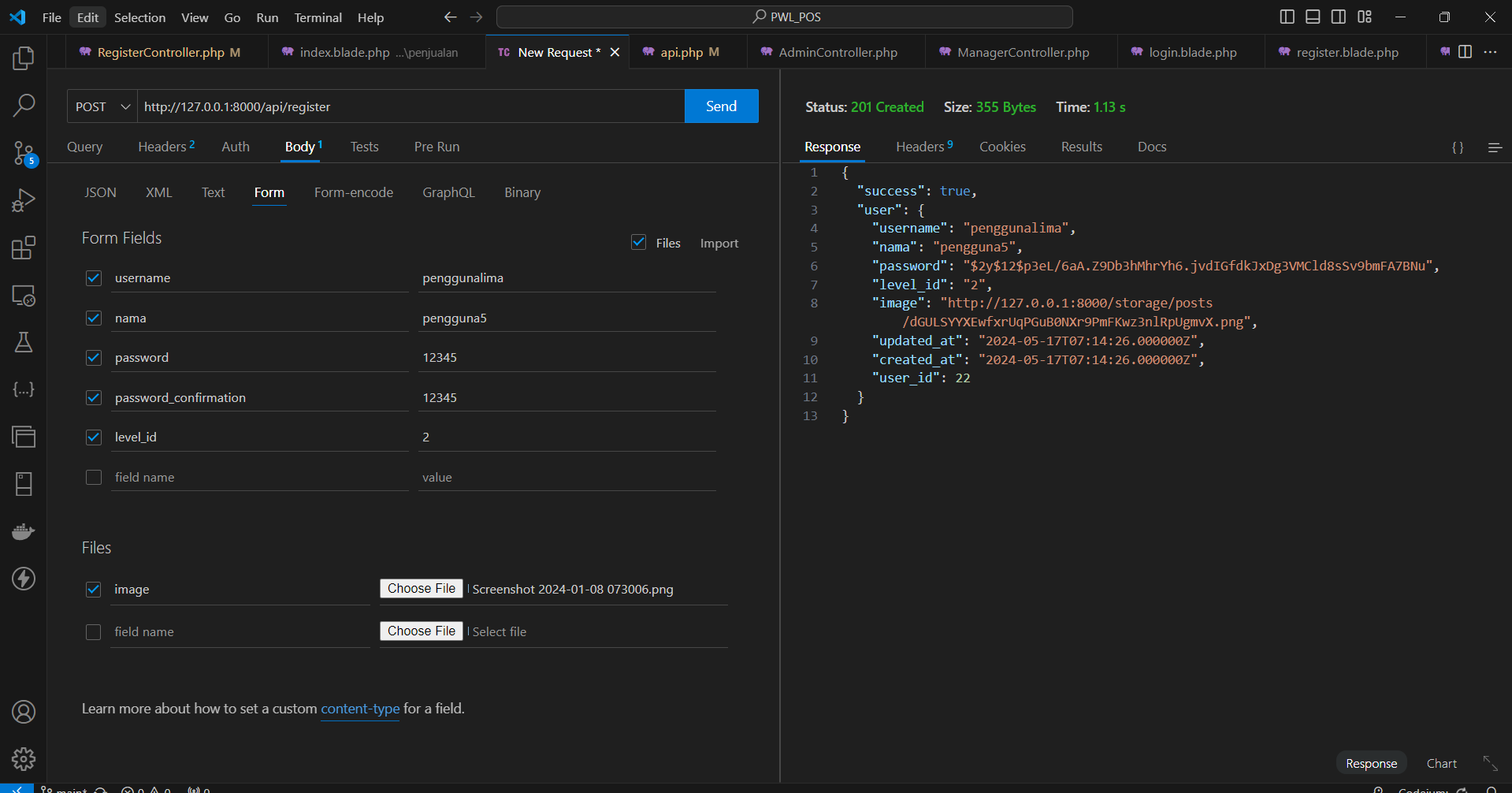


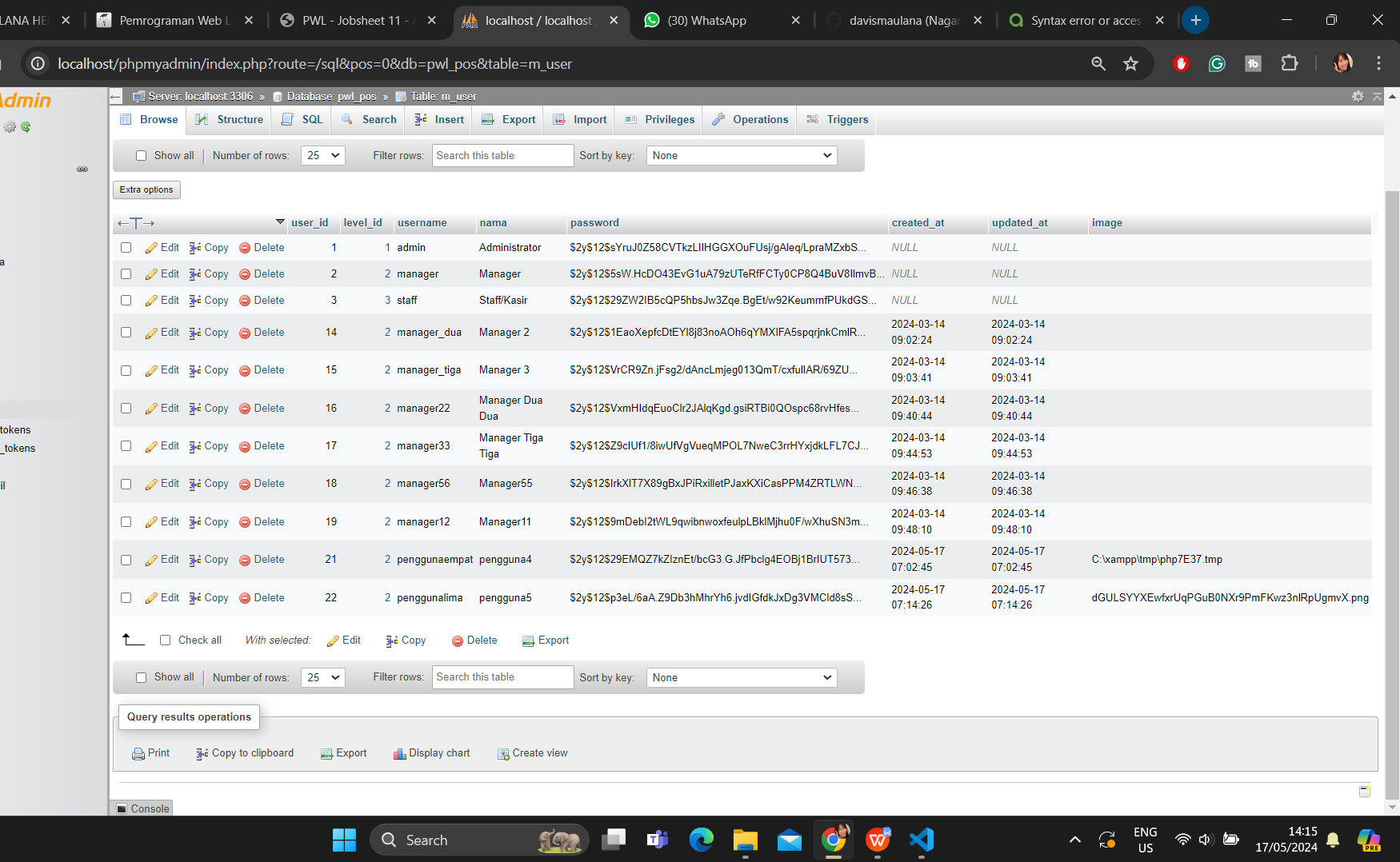
1. In the Controllers/Api/RegisterController section create user replace with



1. Test and screenshot the result what is the difference from the previous one







# ASSIGNMENT

Implement an API to upload files/images to other tables, namely m\_barang tables, and use them in transactions. Test with the GET method to call the data that has been inputted.

*That's it, and happy learning \*\*\**