

PEMROGRAMAN WEB LANJUT

JOBSHEET 11



MUHAMMAD IQBAL MAKMUR AL-MUNIRI
TI-2C / 15 / 2241720099
TEKNIK INFORMATIKA
TEKNOLOGI INFORMASI

Praktikum 1 – Implementasi Eloquent Accessor

1. Sebelum memulai pastikan REST API, terlebih dahulu pastikan sudah ter install aplikasi Postman.
2. Kita akan memodifikasi Table m_user dengan menambahkan column : image, buka terminal lalu ketikkan

Jawab:

```
PS C:\xampp\htdocs\pwl_pos> php artisan make:migration add_image_to_m_user_table

INFO Migration [C:\xampp\htdocs\pwl_pos\database\Migrations\2024_05_10_084416_add_image_to_m_user_table.php] created successfully.

PS C:\xampp\htdocs\pwl_pos>
```

3. Buka file migrasi tersebut lalu modifikasi seperti ini lalu simpan

Jawab:

```
database > migrations > 2024_05_10_084416_add_image_to_m_user_table.php > class > down

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::table('m_user', function (Blueprint $table) {
15             $table->string('image');
16         });
17     }
18
19     /**
20      * Reverse the migrations.
21      */
22     public function down(): void
23     {
24         Schema::table('m_user', function (Blueprint $table) {
25             $table->dropColumn('image');
26         });
27     }
28 };
```

4. Lakukan jalankan update migrasi dengan cara

Jawab:

```
PS C:\xampp\htdocs\pwl_pos> php artisan migrate

INFO Running migrations.

2024_03_29_093417_create_m_user_table ..... 84ms DONE
2024_05_10_084416_add_image_to_m_user_table ..... 3ms DONE
```

5. Lalu lakukan modifikasi models pada App/Models/UserModel.php

Jawab:

```
app > Models > UserModel.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Casts\Attribute;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Tymon\JWTAuth\Contracts\JWTSubject;
9
10 class UserModel extends Authenticatable implements JWTSubject
11 {
12     public function getJWTIdentifier()
13     {
14         return $this->getKey();
15     }
16
17     public function getJWTCustomClaims()
18     {
19         return [];
20     }
21
22     protected $table = 'm_user';
23     protected $primaryKey = 'user_id';
24     protected $fillable = [
25         'username',
26         'nama',
27         'password',
28         'level_id',
29         'image', //tambahan
30     ];
31
32     public function level()
33     {
34         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
35     }
36
37     protected function image(): Attribute
38     {
39         return Attribute::make(
40             get: fn ($image) => url('/storage/posts/' . $image),
41         );
42     }
43 }
44
```

6. Lakukan modifikasi pada Controllers/Api/RegisterController

Jawab:

```
app > Http > Controllers > Api > RegisterController.php > PHP Intelephense > RegisterController > __invoke
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7  use App\Http\Controllers\Controller;
8  use Illuminate\Support\Facades\Validator;
9
10 class RegisterController extends Controller
11 {
12     public function __invoke(Request $request)
13     {
14         $validator = Validator::make($request->all(), [
15             'username' => 'required',
16             'nama' => 'required',
17             'password' => 'required|min:5|confirmed',
18             'level_id' => 'required',
19             'image' => 'required'
20         ]);
21
22         if ($validator->fails()) {
23             return response()->json($validator->errors(), 422);
24         }
25
26         $user = UserModel::create([
27             'username' => $request->username,
28             'nama' => $request->nama,
29             'password' => bcrypt($request->password),
30             'level_id' => $request->level_id,
31             'image' => $request->image
32         ]);
33
34         if ($user) {
35             return response()->json([
36                 'success' => true,
37                 'user' => $user,
38             ], 201);
39         }
40
41         return response()->json([
42             'success' => false,
43         ], 409);
44     }
45 }
```

7. Anda dapat menambahkan detail untuk spesifikasi image pada validator

Jawab:

```
'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
```

8. Ubah atau tambahkan register1 pada routes/api.php

Jawab:

```
Route::post('/register1', App\Http\Controllers\Api\RegisterController::class)->name('register1');
```

9. Simpan dan akses pada aplikasi Postman, atur pada Body isi manual key dan Valuenya pada Key Image tambahkan value file dan upload gambar.

Jawab:

The screenshot shows the Postman interface for a POST request to `http://127.0.0.1:8000/api/register1`. The request is configured with form-data. The body contains the following fields:

Key	Value	Description
username	penggunaempat	
nama	pengguna4	
password	12345	
password_confirmation	12345	
level_id	2	
image	download.jpg	

The response is a JSON object with error messages for each field:

```
1 {
2   "username": [
3     "The username field is required."
4   ],
5   "nama": [
6     "The nama field is required."
7   ],
8   "password": [
9     "The password field is required."
10  ],
11  "level_id": [
12    "The level id field is required."
13  ],
14  "image": [
15    "The image field is required."
16  ]
17 }
```

Overview x POST http://127.0.0.1:8000/api/register1

Save Share

POST http://127.0.0.1:8000/api/register1 Send

Params Auth Headers (9) Body Scripts Tests Settings Cookies

form-data

	Key		Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	Text	penggunaempat			
<input checked="" type="checkbox"/>	nama	Text	pengguna4			
<input checked="" type="checkbox"/>	password	Text	12345			
<input checked="" type="checkbox"/>	password_confirmation	Text	12345			
<input checked="" type="checkbox"/>	level_id	Text	2			
<input checked="" type="checkbox"/>	image	File	download.jpg			
	Key	Text	Value	Description		

Body 201 Created 812 ms 613 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "success": true,
3   "user": {
4     "username": "penggunaempat",
5     "nama": "pengguna4",
6     "password": "$2y$12$vbVv8vETaUeDFdVBXZmFq0r5y4des7GjRkpdR16RQU0otqWIJ1y0",
7     "level_id": "2",
8     "image": "http://127.0.0.1:8000/storage/posts/C:\\xampp\\tmp\\php9B39.tmp",
9     "updated_at": "2024-05-10T09:15:17.000000Z",
10    "created_at": "2024-05-10T09:15:17.000000Z",
11    "user_id": 7
12  }
13 }
```

10. Pada Controllers/Api/RegisterController bagian create user ganti dengan Jawab:

```

26 $user = UserModel::create([
27     'username' => $request->username,
28     'nama' => $request->nama,
29     'password' => bcrypt($request->password),
30     'level_id' => $request->level_id,
31     'image' => $request->file('image')->hashName(),
32 ]);
```

11. Uji coba dan screenshot hasilnya apa perbedaan dari yang sebelumnya.

Jawab:

The screenshot shows a Postman interface for a REST client. The request is a POST to `http://127.0.0.1:8000/api/register1`. The request body is form-data with the following fields:

Key	Type	Value	Description
username	Text	penggunalima	
nama	Text	pengguna5	
password	Text	12345	
password_confirmation	Text	12345	
level_id	Text	2	
image	File	download.jpg	

The response body is JSON, showing a successful registration:

```
{
  "success": true,
  "user": {
    "username": "penggunalima",
    "nama": "pengguna5",
    "password": "$2y$12$UAH4CMAufHX6je4KeJSNq.0SEGvobWeyG3VH3LmJTp5Kxuhz03Rw2",
    "level_id": "2",
    "image": "http://127.0.0.1:8000/storage/posts/9EC0cHXnWzEJfAk94C4U5dajsKeej7snuFZKJLli.jpg",
    "updated_at": "2024-05-10T09:49:58.000000Z",
    "created_at": "2024-05-10T09:49:58.000000Z",
    "user_id": 8
  }
}
```

Penjelasan

```
'image' => $request->file('image')->hashName(),
```

Penggunaan kode diatas bertujuan agar nama file diubah menjadi unique data sehingga file tidak akan bertabrakan dengan file lainnya. Digunakan untuk menghindari konflik nama file yang sama.

TUGAS

Implementasikan API untuk upload file/gambar pada tabel lainnya yaitu tabel m_barang dan gunakan pada transaksi. Uji coba dengan method GET untuk memanggil data yang sudah di inputkan.

Jawab:

Php artisan make:migration add_image_to_m_barang_table

```
PS C:\xampp\htdocs\pwl_pos> php artisan make:migration add_image_to_m_barang_table

INFO Migration [C:\xampp\htdocs\pwl_pos\database\Migrations\2024_05_10_110332_add_image_to_m_barang_table.php] created successfully.
```

Add_image_to_m_barang

```
database > migrations > 2024_05_10_110332_add_image_to_m_barang_table.php > ...

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::table('m_barang', function (Blueprint $table) {
15             $table->string('image');
16         });
17     }
18
19     /**
20      * Reverse the migrations.
21      */
22     public function down(): void
23     {
24         Schema::table('m_barang', function (Blueprint $table) {
25             $table->dropColumn('image');
26         });
27     }
28 };
```


BarangModel.php

```
app > Models > BarangModel.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use App\Models\KategoriModel;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8  use Illuminate\Database\Eloquent\Factories\HasFactory;
9  use Illuminate\Database\Eloquent\Casts\Attribute;
10
11 class BarangModel extends Model
12 {
13     use HasFactory;
14
15     protected $table = 'm_barang';
16     protected $primaryKey = 'barang_id';
17
18     // protected $guarded = ['barang_id'];
19
20     protected $fillable = [
21         'barang_kode',
22         'kategori_id',
23         'barang_nama',
24         'harga_jual',
25         'harga_beli',
26         'image',
27     ];
28
29     public function kategori(): BelongsTo
30     {
31         return $this->belongsTo(KategoriModel::class, 'kategori_id', 'kategori_id');
32     }
33
34     protected function image(): Attribute
35     {
36         return Attribute::make(
37             get: fn ($image) => url('/storage/posts/' . $image),
38         );
39     }
40 }
41
```

Routes

```
38 Route::post('/register1', App\Http\Controllers\Api\RegisterController::class)->name('
39
40 Route::get('barang', [BarangController::class, 'index']);
41 Route::post('barang', [BarangController::class, 'store']);
42 Route::get('barang/{barang}', [BarangController::class, 'show']);
43 Route::put('barang/{barang}', [BarangController::class, 'update']);
44 Route::delete('barang/{barang}', [BarangController::class, 'destroy']);
45
46 Route::get('penjualan1', [PenjualanController::class, 'index']);
47 Route::post('penjualan', [PenjualanController::class, 'store']);
48 Route::get('penjualan/{penjualan}', [PenjualanController::class, 'show']);
49 Route::put('penjualan/{penjualan}', [PenjualanController::class, 'update']);
50 Route::delete('penjualan/{penjualan}', [PenjualanController::class, 'destroy']);
```

PenjualanController.php (API)

app > Http > Controllers > Api > PenjualanController.php > PHP Intelephense > PenjualanController

```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\PenjualanModel;
7 use Illuminate\Http\Request;
8
9 class PenjualanController extends Controller
10 {
11     public function index()
12     {
13         return PenjualanModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $penjualan = PenjualanModel::create($request->all());
19         return response()->json($penjualan, 201);
20     }
21
22     public function show(PenjualanModel $penjualan)
23     {
24         return PenjualanModel::find($penjualan);
25     }
26
27     public function update(Request $request, PenjualanModel $penjualan)
28     {
29         $penjualan->update($request->all());
30         return PenjualanModel::find($penjualan);
31     }
32
33     public function destroy(PenjualanModel $penjualan)
34     {
35         $penjualan->delete();
36
37         return response()->json([
38             'success' => true,
39             'message' => 'Data terhapus'
40         ]);
41     }
42 }
```

