|  |  |  |
| --- | --- | --- |
| Mata Kuliah | **:** | Pemrograman Web Lanjut (PWL) |
| Program Studi | **:** | D4 – Sistem Informasi Bisnis |
| Semester | **:** | 5 |
| Kelas | **:** | SIB |
| NIM | **:** | 2241760007 |
| Nama | **:** | Anaradi Octa Lavechia |
| Jobsheet Ke- | **:** | 3 (tiga) |

**Laporan Jobsheet 03 – MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM**

**Praktikum Ke-1**

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Buka aplikasi phpMyAdmin, dan buat database baru dengan nama PWL\_POS |
| 2 | Buka aplikasi VSCode dan buka folder project PWL\_POS yang sudah kita buat |
| 3 | Copy file .env.example menjadi .env |
| 4 | Buka file .env, dan pastikan konfigurasi APP\_KEY bernilai. Jika belum bernilai silahkan kalian generate menggunakan php artisan. |
| 5 | Edit file .env dan sesuaikan dengan database yang telah dibuat |
| 6 | Laporkan hasil Praktikum-1 ini dan commit perubahan pada git. |

**Praktikum Ke-2.1**

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Buka terminal VSCode kalian, untuk yang di kotak merah adalah default dari laravel |
| 2 | Kita abaikan dulu yang di kotak merah (jangan di hapus) |
| 3 | Kita buat file migrasi untuk table m\_level dengan perintah  php artisan make:migration create\_m\_level\_table --create=m\_level |
| 4 | Kita perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada |
| 5 | Simpan kode pada tahapan 4 tersebut, kemudian jalankan perintah ini pada terminal VSCode untuk melakukan migrasi  php artisan migrate |
| 6 | Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum |
| 7 | Ok, table sudah dibuat di database |
| 8 | Buat table database dengan migration untuk table m\_kategori dan m\_supplier yang sama-sama tidak memiliki foreign key |
| 9 | Laporkan hasil Praktikum-2.1 ini dan commit perubahan pada git. |

**Praktikum Ke-2.2**

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Buka terminal VSCode kalian, dan buat file migrasi untuk table m\_user  php artisan make:migration create\_m\_user\_table --table=m\_user |
| 2 | Buka file migrasi untuk table m\_user, dan modifikasi seperti berikut |
| 3 | Simpan kode program Langkah 2, dan jalankan perintah php artisan migrate. Amati apa yang terjadi pada database. |
| 4 | Buat table database dengan migration untuk table-tabel yang memiliki foreign key     * Table m\_kategori          * Table m\_barang          * Table t\_stok          * Table t\_penjualan        * Table t\_penjualan\_detail          * Table m\_supplier |
| 5 | Jika semua file migrasi sudah di buat dan dijalankan maka bisa kita lihat tampilan designer pada phpMyAdmin seperti berikut |
| 6 | Laporkan hasil Praktikum-2.2 ini dan commit perubahan pada git. |

**Praktikum Ke-3**

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Kita akan membuat file seeder untuk table m\_level dengan mengetikkan perintah php artisan make:seeder LevelSeeder |
| 2 | Selanjutnya, untuk memasukkan data awal, kita modifikasi file tersebut di dalam function run() |
| 3 | Selanjutnya, kita jalankan file *seeder* untuk table m\_level pada terminal dengan perintah php artisan db:seed --class=LevelSeeder |
| 4 | Ketika seeder berhasil dijalankan maka akan tampil data pada table m\_level |
| 5 | Sekarang kita buat file *seeder* untuk table m\_user yang me-refer ke table m\_level dengan perintah php artisan make:seeder UserSeeder |
| 6 | Modifikasi file class UserSeeder seperti berikut |
| 7 | Jalankan perintah untuk mengeksekusi class UserSeeder |
| 8 | Perhatikan hasil seeder pada table m\_user |
| 9 | Ok, data *seeder* berhasil di masukkan ke database. |
| 10 | Sekarang coba kalian masukkan data *seeder* untuk table yang lain, dengan ketentuan seperti berikut     * Isi tabel Kategori          * Isi tabel Supplier            * Isi tabel Barang          * Isi tabel Stok          * Isi tabel Penjualan          * Isi tabel Penjualan\_detail |
| 11 | Laporkan hasil Praktikum-3 ini dan commit perubahan pada git. |

**Praktikum Ke-4**

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Kita buat controller dahulu untuk mengelola data pada table m\_level dengan perintah php artisan make:controller LevelController |
| 2 | Kita modifikasi dulu untuk routing-nya, ada di PWL\_POS/routes/web.php |
| 3 | Selanjutnya, kita modifikasi file LevelController untuk menambahkan 1 data ke table m\_level |
| 4 | Kita coba jalankan di browser dengan url localhost/PWL\_POS/public/level dan amati apa yang terjadi pada table m\_level di database*, screenshot* perubahan yang ada pada table m\_level |
| 5 | Selanjutnya, kita modifikasi lagi file LevelController untuk meng-*update* data di table m\_level seperti berikut |
| 6 | Kita coba jalankan di browser dengan url localhost/PWL\_POS/public/level lagi dan amati apa yang terjadi pada table m\_level di database, screenshot perubahan yang ada pada table m\_level |
| 7 | Kita coba modifikasi lagi file LevelController untuk melakukan proses hapus data |
| 8 | Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m\_level. Kita modifikasi file LevelController seperti berikut |
| 9 | Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil view(‘level’), maka kita buat file view pada VSCode di PWL\_POS/resources/view/level.blade.php |
| 10 | Silahkan dicoba pada browser dan amati apa yang terjadi |
| 11 | Laporkan hasil Praktikum-4 ini dan commit perubahan pada git. |

**Praktikum Ke-5**

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Kita buat controller dahuku untuk mengelola data pada table m\_kategori |
| 2 | Kita modifikasi dulu untuk routing-nya, ada di PWL\_POS/routes/web.php |
| 3 | Selanjutnya, kita modifikasi file KategoriController untuk menambahkan 1 data ke table m\_kategori |
| 4 | Kita coba jalankan di browser dengan url localhost/PWL\_POS/public/kategori dan amati apa yang terjadi pada table m\_kategori di database, screenshot perubahan yang ada pada table m\_kategori |
| 5 | Selanjutnya, kita modifikasi lagi file KategoriController untuk meng-update data di table m\_kategori seperti berikut |
| 6 | Kita coba jalankan di browser dengan url localhost/PWL\_POS/public/kategori lagi dan amati apa yang terjadi pada table m\_kategori di database, screenshot perubahan yang ada pada table m\_kategori |
| 7 | Kita coba modifikasi lagi file KategoriController untuk melakukan proses hapus data |
| 8 | Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m\_kategori. Kita modifikasi file KategoriController seperti berikut |
| 9 | Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil view(‘kategori’), maka kita buat file view pada VSCode di PWL\_POS/resources/view/kategori.blade.php |
| 10 | Silahkan dicoba pada browser dan amati apa yang terjadi. |
| 11 | Laporkan hasil Praktikum-5 ini dan *commit* perubahan pada *git* |

**Praktikum Ke-6**

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Kita buat file model untuk tabel m\_user dengan mengetikkan perintah php artisan make:model UserModel |
| 2 | Setelah berhasil generate model, terdapat 2 file pada folder model yaitu file User.php bawaan dari laravel dan file UserModel.php yang telah kita buat. Kali ini kita akan menggunakan file UserModel.php |
| 3 | Kita buka file UserModel.php dan modifikasi seperti berikut |
| 4 | Kita modifikasi route web.php untuk mencoba routing ke controller UserController |
| 5 | Sekarang, kita buat file controller UserController dan memodifikasinya seperti berikut |
| 6 | Kemudian kita buat view user.blade.php |
| 7 | Jalankan di browser, catat dan laporkan apa yang terjadi |
| 8 | Setelah itu, kita modifikasi lagi file UserController |
| 9 | Jalankan di browser, amati dan laporkan apa yang terjadi |
| 10 | Kita modifikasi lagi file UserController menjadi seperti berikut |
| 11 | Jalankan di browser, amati dan laporkan apa yang terjadi |
| 12 | Jika sudah, laporkan hasil Praktikum-6 ini dan *commit* perubahan pada *git* |

**Soal Praktikum**

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

|  |  |
| --- | --- |
| **Langkah** | **Jawaban/Deskripsi** |
| 1 | Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari APP\_KEY pada file setting .env Laravel?  Berdasarkan pemahaman saya, APP\_KEY pada file setting.env di Laravel adalah kunci rahasia yang digunakan untuk mengamankan data dengan cara mengenkripsi dan mendekripsi informasi penting, seperti sesi pengguna dan cookie. Ini memastikan data aman dan hanya dapat diakses oleh aplikasi yang benar-benar berwenang. |
| 2 | Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk APP\_KEY?  Berdasarkan pemahaman saya, dari penjelasan bapak Dimas kemarin saat jam kuliah, perintah yang digunakan untuk men-*generate* nilai untuk APP\_KEY adalah php artisan key:generate |
| 3 | Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?  Berdasarkan ingatan saya, secara default Laravel memiliki beberapa file migrasi diantaranya file-file berikut yang berfungsi untuk membantu dalam mengatur struktur database dasar yang diperlukan untuk aplikasi Laravel:   * create\_users\_table.php: untuk membuat tabel `users` dan menyimpan data pengguna. * create\_password\_resets\_table.php: untuk membuat tabel `password\_resets` menyimpan data reset password. * create\_failed\_jobs\_table.php: untuk membuat tabel `failed\_jobs` untuk menyimpan informasi pekerjaan yang gagal. |
| 4 | Secara *default*, file migrasi terdapat kode $table->timestamps();, apa tujuan/*output* dari fungsi tersebut?  Berdasarkan pemahaman saya dari hasil praktikum, fungsi dari kode $table->timestamps();, dalam migrasi Laravel menambahkan dua kolom pada tabel database yaitu:   * created\_at: untuk mencatat waktu saat baris data dibuat. * updated\_at: Mencatat waktu saat baris data terakhir diperbarui.   Jadi, kedua kolom tersebut membantu melacak kapan data ditambahkan atau diubah. |
| 5 | Pada File Migrasi, terdapat fungsi $table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?  Berdasarkan pemahaman saya, fungsi dari kode $table->id(); pada file migrasi Laravel menghasilkan kolom dengan tipe data BIGINT yang auto-increment dan menjadi primary key pada tabel. Ini secara otomatis mengatur kolom tersebut sebagai identifikasi unik untuk setiap baris data. |
| 6 | Apa bedanya hasil migrasi pada table m\_level, antara menggunakan $table->id(); dengan menggunakan $table->id('level\_id'); ?  Berdasarkan pemahaman saya dari hasil praktikum, fungsi dari kode $table->id(); pada file migrasi Laravel menghasilkan kolom dengan tipe data BIGINT yang auto-increment dan menjadi primary key pada tabel. Ini secara otomatis mengatur kolom tersebut sebagai identifikasi unik untuk setiap baris data. Sedangkan, perbedaannya adalah table->id();untuk membuat kolom bernama `id ` dengan tipe data BIGINT yang auto-increment dan menjadi primary key tabel secara otomatis. Sementara, $table->id('level\_id'); untuk membuat kolom bernama 'level\_id'dengan tipe data ` BIGINT ` yang auto-increment dan menjadi primary key tabel, tetapi dengan nama kolom yang spesifik yaitu level\_id. Jadi, perbedaannya terletak pada nama kolom yang dihasilkan `id ` vs. `'level\_id' |
| 7 | Pada migration, Fungsi ->unique() digunakan untuk apa?  Berdasarkan pemahaman saya, fungsi dari kode unique() digunakan untuk menetapkan kolom pada tabel agar hanya menerima nilai yang unik. Artinya, tidak boleh ada dua baris atau lebih yang memiliki nilai yang sama di kolom tersebut. |
| 8 | Pada **Praktikum 2.2 - Tahap 2**, kenapa kolom level\_id pada tabel m\_user menggunakan $tabel->unsignedBigInteger('level\_id'), sedangkan kolom level\_id pada tabel m\_level menggunakan $tabel->id('level\_id') ?  Berdasarkan pemahaman saya dari hasil praktikum, kolom level\_id di m\_level didefinisikan sebagai BIGINT dengan auto-increment untuk primary key. Sementara, kolom level\_id di m\_user didefinisikan sebagai unsignedBigInteger untuk menyimpan referensi ke primary key level\_id di m\_level dan diindeks untuk mempercepat pencarian. Jadi, perbedaan ini memastikan bahwa level\_id di m\_user cocok dengan tipe data dan atribut dari level\_id di m\_level, mendukung integritas referensial dalam database |
| 9 | Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class Hash? dan apa maksud dari kode program Hash::make('1234');?  Berdasarkan pemahaman saya dari hasil praktikum, tujuan dari Class `Hash` di Laravel digunakan untuk mengenkripsi password. `Hash::make('12345')` menghasilkan versi terenkripsi dari password `'12345'`, yang disimpan di database untuk keamanan. |
| 10 | Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (*?*), apa kegunaan dari tanda tanya (*?*) tersebut?  Berdasarkan pemahaman saya, dalam *query builder* Laravel tanda tanya (?) adalah placeholder untuk parameter yang akan diisi saat eksekusi query. Fungsinya adalah untuk menggantikan nilai-nilai yang akan diinsert, update, atau digunakan dalam kondisi query. |
| 11 | Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode protected $table = ‘m\_user’; dan protected $primaryKey = ‘user\_id’; ?  Berdasarkan pemahaman saya, tujuan penulisan kode protected $table = ‘m\_user’;untuk menetapkan nama tabel di database yang digunakan oleh model ini. Sementara, tujuan penulisan kode protected $primaryKey = ‘user\_id’; untuk menetapkan nama kolom yang menjadi primary key di tabel tersebut. |
| 12 | Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan  Berdasarkan pemahaman saya, setiap operasi pastinya memiliki kelebihan dan kekurangan masing-masing. Tetapi Menurut saya yang lebih mudah yaitu Eloquent ORM karena lebih mudah dan cepat untuk pengembangan, sementara DB Façade dan Query Builder menawarkan lebih banyak kontrol dan fleksibilitas. |