|  |  |  |
| --- | --- | --- |
| Mata Kuliah | **:** | Pemrograman Web Lanjut (PWL) |
| Program Studi | **:** | D4 – Teknik Informatika **/** D4 – Sistem Informasi Bisnis |
| Semester | **:** | 4 (empat) / 6 (enam) |
| Pertemuan ke- | **:** | 1 (satu) |

# JOBSHEET 04 MODEL dan ELOQUENT ORM

Sebelumnya kita sudah membahas mengenai *Migration, Seeder*, *DB Façade, Query Builder,* dan sedikit tentang *Eloquent ORM* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll.

Dalam pertemuang kali ini kita akan memahami tentang bagainama cara menampilkan data, mengubah data, dan menghapus data menggunakan teknik Eloquent.

Sesuai dengan **Studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL\_POS.**

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

**ORM (Object Relation Mapping)** merupakan teknik yang merubah suatu table menjadi sebuah object yang nantinya mudah untuk digunakan. Object yang dibuat memiliki property yang sama dengan field — field yang ada pada table tersebut. ORM tersebut bertugas sebagai penghubung dan sekaligus mempermudah kita dalam membuat aplikasi yang menggunakan database relasional agar menjadikan tugas kita lebih efisien.

**Kelebihan - Kelebihan Menggunakan ORM**

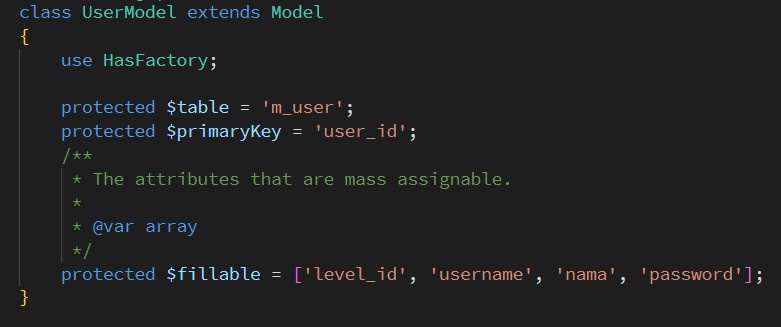
1. Terdapat banyak fitur seperti transactions, connection pooling, migrations, seeds, streams, dan lain sebagainya.
2. perintah query memiliki kinerja yang lebih baik, daripada kita menulisnya secara manual.
3. Kita menulis model data hanya di satu tempat, sehingga lebih mudah untuk update, maintain, dan reuse the code.
4. Memungkinkan kita memanfaatkan OOP (object oriented programming) dengan baik Di Laravel sendiri telah disediakan Eloquent ORM untuk mempermudah kita dalam melakukan berbagai macam query ke database, dan membuat pekerjaan kita menjadi lebih mudah karena tidak perlu menuliskan query sql yang panjang untuk memproses data.
5. **PROPERTI $fillable DAN $guarded**
   1. $fillable

Variable $fillable berguna untuk mendaftarkan atribut (nama kolom) yang bisa kita isi ketika melakukan insert atau update ke database. Sebelumnya kita sudah memahami menambahkan record baru ke database. Untuk langkah menambahkan Variable $fillable bisa dengan menambahkan *script* seperti di bawah ini pada file model

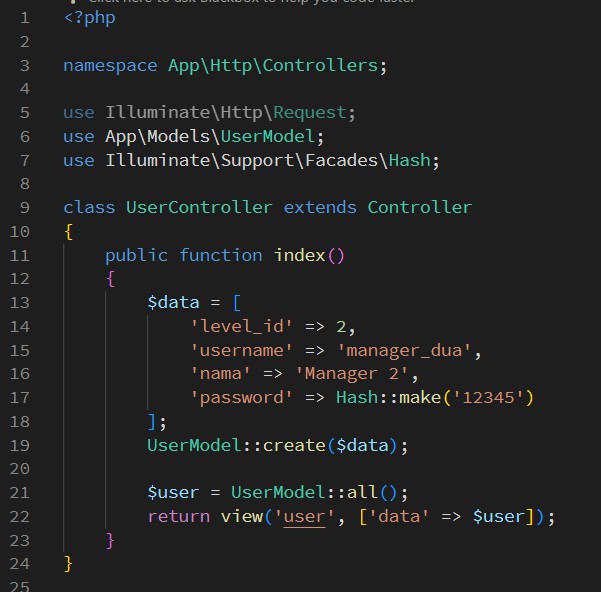


**Praktikum 1** - $fillable:

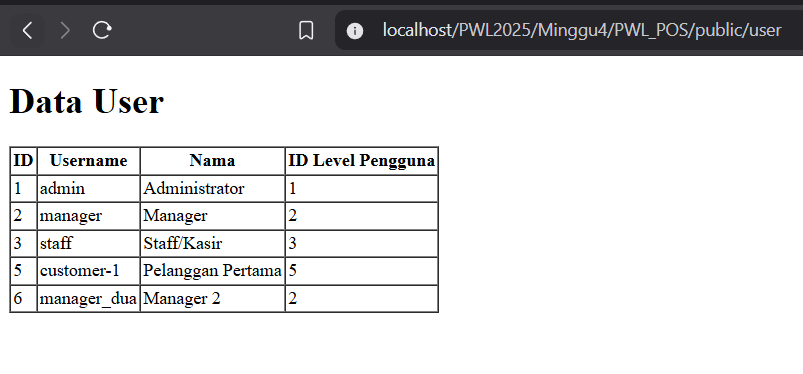
* + 1. Buka file model dengan nama UserModel.php dan tambahkan $fillable seperti gambar di bawah ini



* + 1. Buka file controller dengan nama UserController.php dan ubah *script* untuk menambahkan data baru seperti gambar di bawah ini



* + 1. Simpan kode program Langkah 1 dan 2, dan jalankan perintah web server. Kemudian jalankan link localhostPWL\_POS/public/user pada *browser* dan amati apa yang terjadi

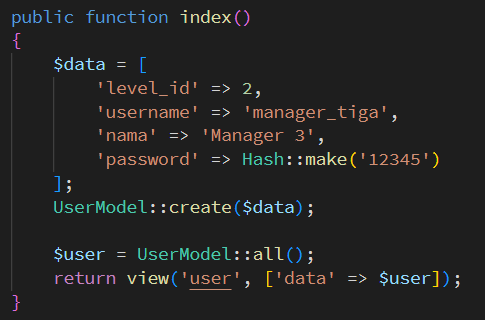


* + 1. Ubah file model UserModel.php seperti pada gambar di bawah ini pada bagian

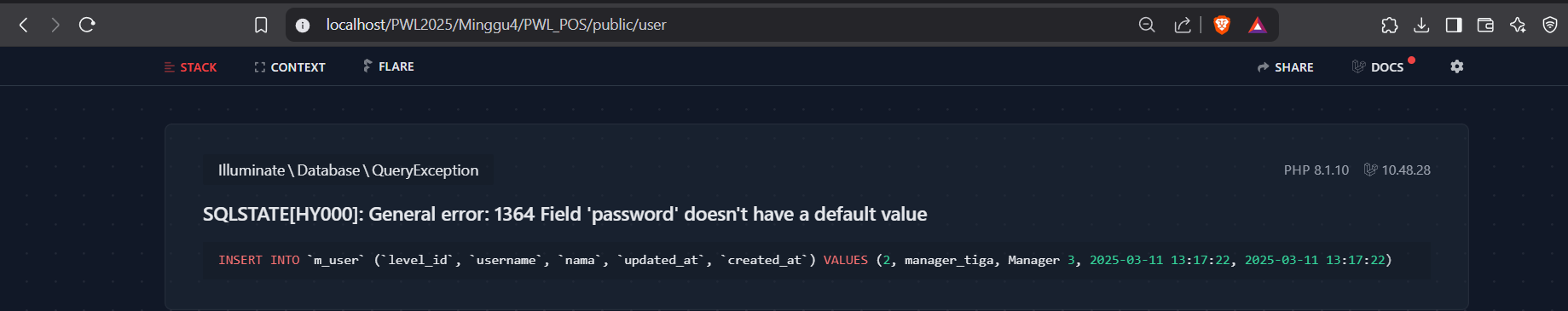
$fillable



* + 1. Ubah kembali file controller UserController.php seperti pada gambar di bawah hanya bagian array pada *$data*



* + 1. Simpan kode program Langkah 4 dan 5. Kemudian jalankan pada *browser* dan amati apa yang terjadi



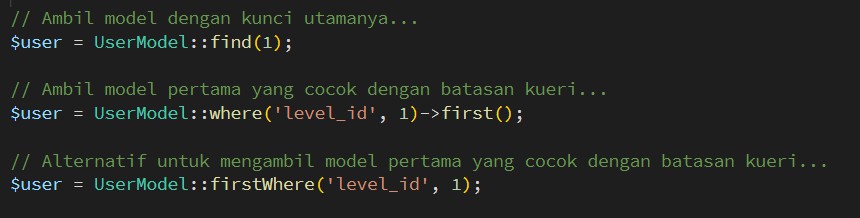
**Hasil Error dikarenakan bagian password tidak diisi nilai pada usermodel.**

* + 1. Laporkan hasil Praktikum-1 ini dan *commit* perubahan pada *git*.
  1. $guarded

Kebalikan dari $fillable adalah $guarded. Semua kolom yang kita tambahkan ke $guarded akan diabaikan oleh Eloquent ketika kita melakukan insert/update. Secara default $guarded isinya array("\*"), yang berarti semua atribut tidak bisa diset melalui ***mass assignment***. ***Mass Assignment*** adalah fitur canggih yang menyederhanakan proses pengaturan beberapa atribut model sekaligus, menghemat waktu dan tenaga. Pada praktikum ini, kita akan mengeksplorasi konsep penugasan massal di Laravel dan bagaimana hal itu dapat dimanfaatkan secara efektif untuk meningkatkan alur kerja pengembangan Anda.

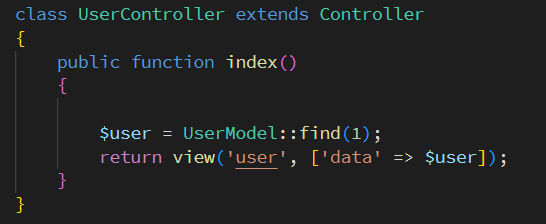
# RETRIEVING SINGLE MODELS

Selain mengambil semua rekaman yang cocok dengan kueri tertentu, Anda juga dapat mengambil rekaman tunggal menggunakan metode find, first, atau firstWhere. Daripada mengembalikan kumpulan model, metode ini mengembalikan satu contoh model dan dilakukan pada controller:

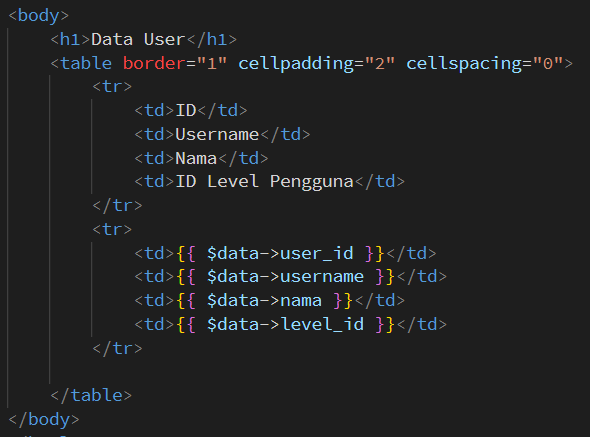


**Praktikum 2.1** – Retrieving Single Models

* 1. Buka file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



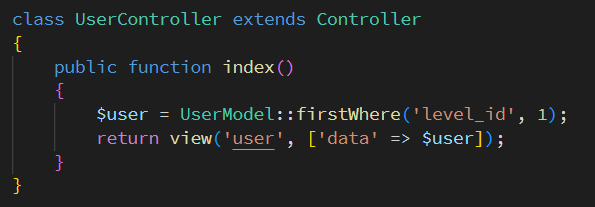
* 1. Buka file *view* dengan nama user.blade.php dan ubah *script* seperti gambar di bawah ini



* 1. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
  2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini

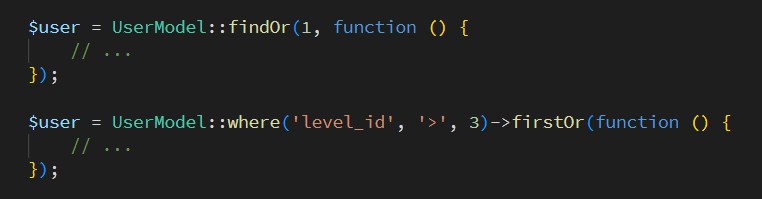


* 1. Simpan kode program Langkah 4. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
  2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini

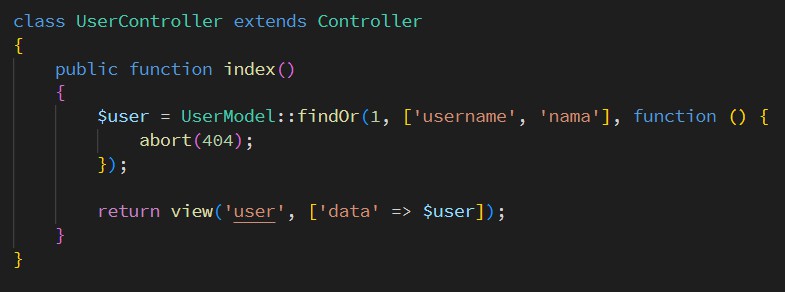


* 1. Simpan kode program Langkah 6. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan

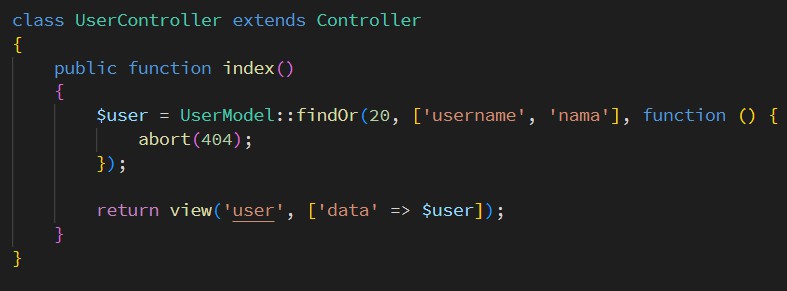
Terkadang Anda mungkin ingin melakukan beberapa tindakan lain jika tidak ada hasil yang ditemukan. Metode findOr and firstOr akan mengembalikan satu contoh model atau, jika tidak ada hasil yang ditemukan maka akan menjalankan didalam fungsi. Nilai yang dikembalikan oleh fungsi akan dianggap sebagai hasil dari metode ini:



* 1. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



* 1. Simpan kode program Langkah 8. Kemudian pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
  2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini

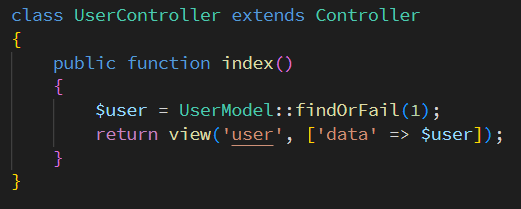


* 1. Simpan kode program Langkah 10. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
  2. Laporkan hasil Praktikum-2.1 ini dan *commit* perubahan pada *git*.

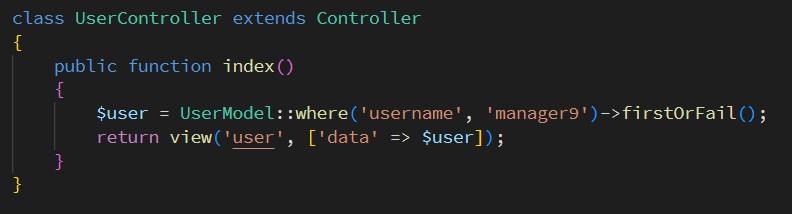
**Praktikum 2.2** – *Not Found Exceptions*

Terkadang Anda mungkin ingin memberikan pengecualian jika model tidak ditemukan. Hal ini sangat berguna dalam *rute* atau pengontrol. Metode findOrFail and firstOrFail akan mengambil hasil pertama dari kueri; namun, jika tidak ada hasil yang ditemukan, sebuah Illuminate\Database\Eloquent\ModelNotFoundException akan dilempar. Berikut ikuti langkah-langkah di bawah ini:

1. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



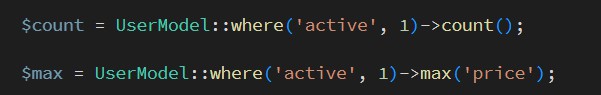
1. Simpan kode program Langkah 1. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



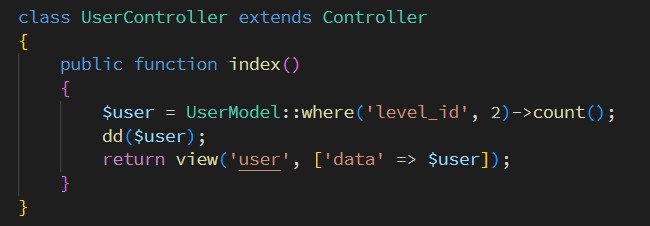
1. Simpan kode program Langkah 3. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
2. Laporkan hasil Praktikum-2.2 ini dan *commit* perubahan pada *git*.

**Praktikum 2.3** – *Retreiving Aggregrates*

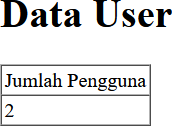
Saat berinteraksi dengan model Eloquent, Anda juga dapat menggunakan metode agregat count, sum, max, dan lainnya yang disediakan oleh pembuat kueri Laravel. Seperti yang Anda duga, metode ini mengembalikan nilai skalar dan contoh model Eloquent:



1. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



1. Simpan kode program Langkah 1. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
2. Buat agar jumlah *script* pada langkah 1 bisa tampil pada halaman *browser*, sebagai contoh bisa lihat gambar di bawah ini dan ubah *script* pada file *view* supaya bisa muncul datanya

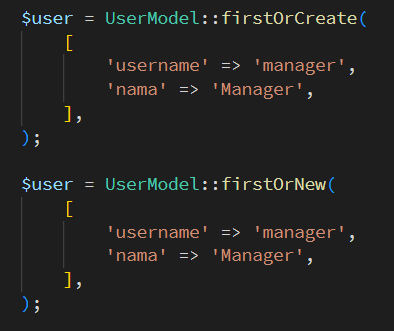


1. Laporkan hasil Praktikum-2.3 ini dan *commit* perubahan pada *git*.

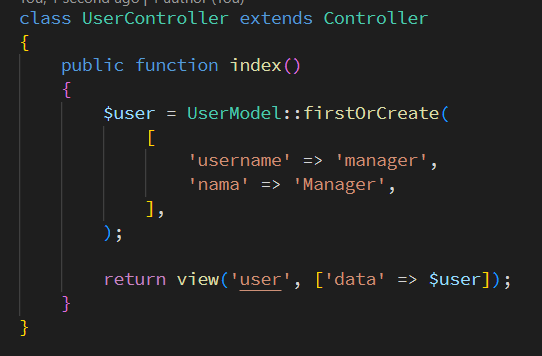
**Praktikum 2.4** – *Retreiving or Creating Models*

Metode firstOrCreate merupakan metode untuk melakukan *retrieving data* (mengambil data) berdasarkan nilai yang ingin dicari, jika data tidak ditemukan maka method ini akan melakukan insert ke table datadase tersebut sesuai dengan nilai yang dimasukkan.

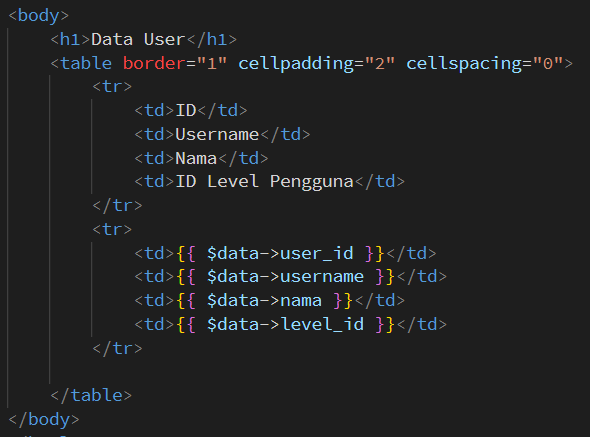
Metode firstOrNew, seperti firstOrCreate, akan mencoba menemukan/mengambil *record/data* dalam database yang cocok dengan atribut yang diberikan. Namun, jika data tidak ditemukan, data akan disiapkan untuk di-*insert*-kan ke database dan model baru akan dikembalikan. Perhatikan bahwa model yang dikembalikan firstOrNew belum disimpan ke database. Anda perlu memanggil metode save() secara manual untuk menyimpannya:



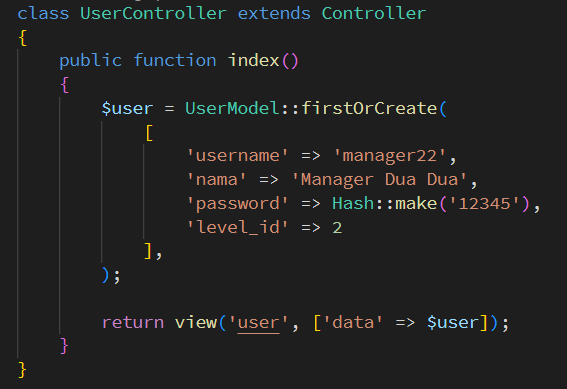
1. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



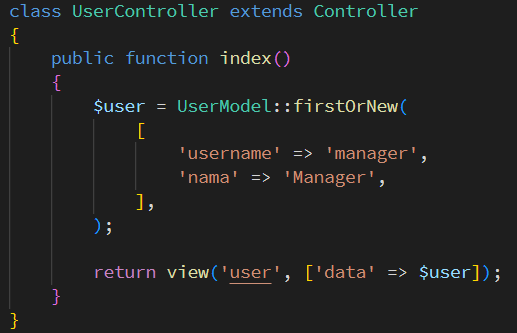
1. Ubah kembali file *view* dengan nama user.blade.php dan ubah *script* seperti gambar di bawah ini



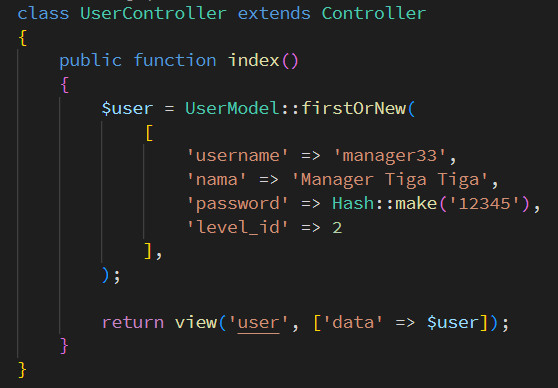
1. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



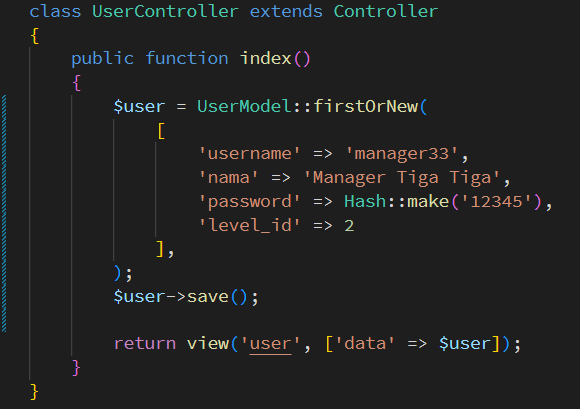
1. Simpan kode program Langkah 4. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel m\_user serta beri penjelasan dalam laporan
2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



1. Simpan kode program Langkah 6. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



1. Simpan kode program Langkah 8. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel *m\_user* serta beri penjelasan dalam laporan
2. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini

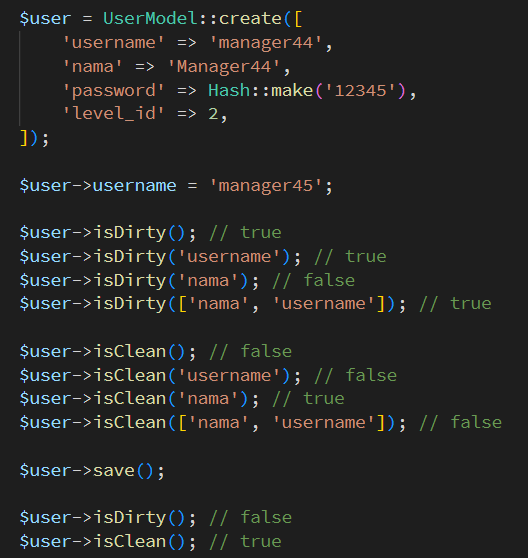


1. Simpan kode program Langkah 9. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel m\_user serta beri penjelasan dalam laporan
2. Laporkan hasil Praktikum-2.4 ini dan *commit* perubahan pada *git*.

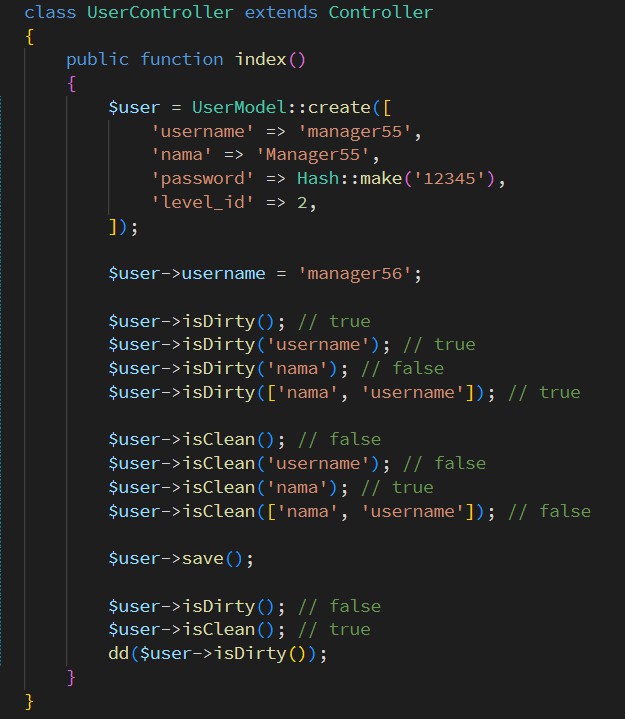
**Praktikum 2.5** – *Attribute Changes*

Eloquent menyediakan metode isDirty, isClean, dan wasChanged untuk memeriksa keadaan internal model Anda dan menentukan bagaimana atributnya berubah sejak model pertama kali diambil.

Metode isDirty menentukan apakah ada atribut model yang telah diubah sejak model diambil. Anda dapat meneruskan nama atribut tertentu atau serangkaian atribut ke metode isDirty untuk menentukan apakah ada atribut yang "kotor". Metode ini isClean akan menentukan apakah suatu atribut tetap tidak berubah sejak model diambil. Metode ini juga menerima argumen atribut opsional:



1. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



1. Simpan kode program Langkah 1. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan

Metode ini wasChanged menentukan apakah ada atribut yang diubah saat model terakhir disimpan dalam siklus permintaan saat ini. Jika diperlukan, Anda dapat memberikan nama atribut untuk melihat apakah atribut tertentu telah diubah:



1. Ubah file controller dengan nama UserController.php dan ubah *script* seperti gambar di bawah ini



1. Simpan kode program Langkah 3. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
2. Laporkan hasil Praktikum-2.5 ini dan *commit* perubahan pada *git*.

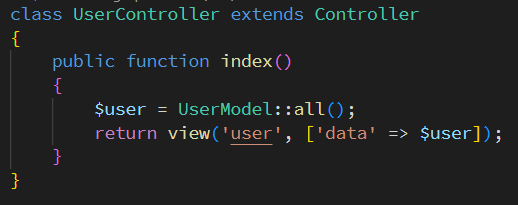
**Praktikum 2.6** – *Create, Read, Update, Delete (CRUD)*

Seperti yang telah kita ketahui, CRUD merupakan singkatan dari *Create, Read, Update* dan *Delete*. CRUD merupakan istilah untuk proses pengolahan data pada database, seperti input data ke database, menampilkan data dari database, mengedit data pada database dan menghapus data dari database. Ikuti langkah-langkah di bawah ini untuk melakukan CRUD dengan Eloquent

1. Buka file *view* pada user.blade.php dan buat scritpnya menjadi seperti di bawah ini



1. Buka file controller pada *UserController.php* dan buat scriptnya untuk *read* menjadi seperti di bawah ini



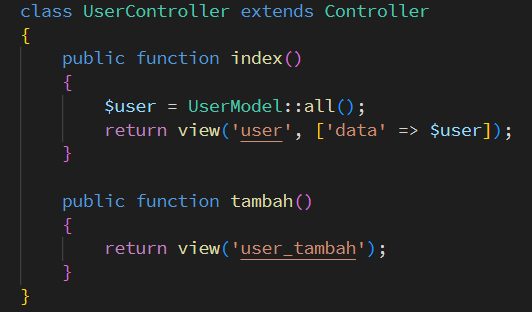
1. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
2. Langkah berikutnya membuat *create* atau tambah data user dengan cara bikin file baru pada *view* dengan nama user\_tambah.blade.php dan buat scriptnya menjadi seperti di bawah ini



1. Tambahkan *script* pada *routes* dengan nama file web.php. Tambahkan seperti gambar di bawah ini



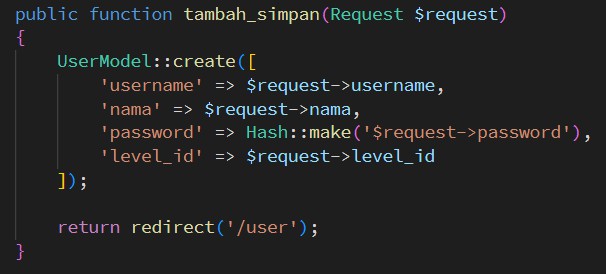
1. Tambahkan *script* pada controller dengan nama file *UserController.php*. Tambahkan *script* dalam class dan buat method baru dengan nama tambah dan diletakan di bawah method index seperti gambar di bawah ini



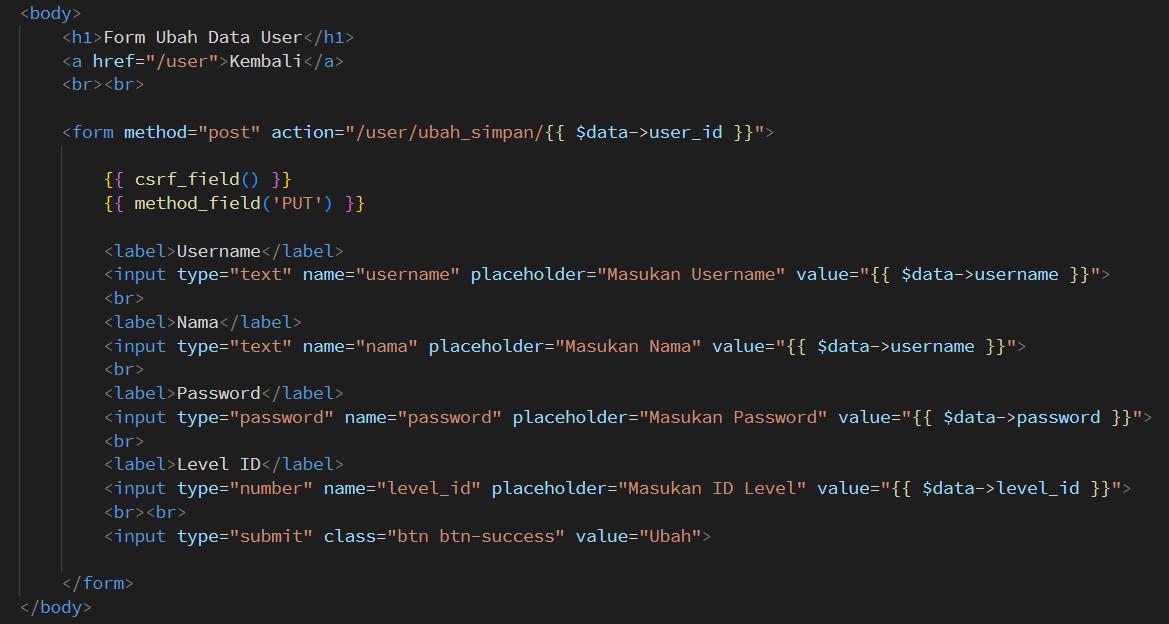
1. Simpan kode program Langkah 4 s/d 6. Kemudian jalankan pada *browser* dan klik link “**+ Tambah User**” amati apa yang terjadi dan beri penjelasan dalam laporan
2. Tambahkan *script* pada *routes* dengan nama file web.php. Tambahkan seperti gambar di bawah ini



1. Tambahkan *script* pada controller dengan nama file *UserController.php*. Tambahkan *script* dalam class dan buat method baru dengan nama tambah\_simpan dan diletakan di bawah method tambah seperti gambar di bawah ini



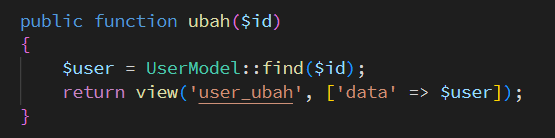
1. Simpan kode program Langkah 8 dan 9. Kemudian jalankan link localhost:8000/user/tambah atau localhost/PWL\_POS/public/user/tambah pada *browser* dan input formnya dan simpan, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan
2. Langkah berikutnya membuat *update* atau ubah data user dengan cara bikin file baru pada *view* dengan nama user\_ubah.blade.php dan buat scriptnya menjadi seperti di bawah ini



1. Tambahkan *script* pada *routes* dengan nama file web.php. Tambahkan seperti gambar di bawah ini



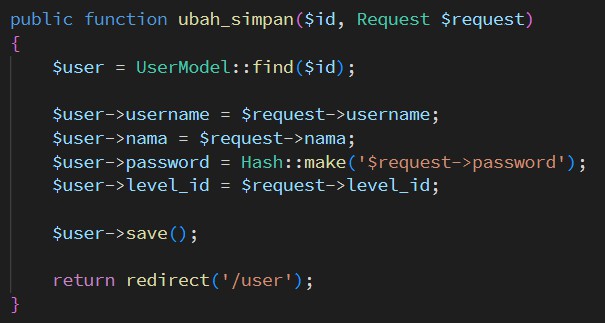
1. Tambahkan *script* pada controller dengan nama file UserController.php. Tambahkan *script* dalam class dan buat method baru dengan nama ubah dan diletakan di bawah method tambah\_simpan seperti gambar di bawah ini



1. Simpan kode program Langkah 11 sd 13. Kemudian jalankan pada *browser* dan klik link “**Ubah**” amati apa yang terjadi dan beri penjelasan dalam laporan
2. Tambahkan *script* pada *routes* dengan nama file web.php. Tambahkan seperti gambar di bawah ini



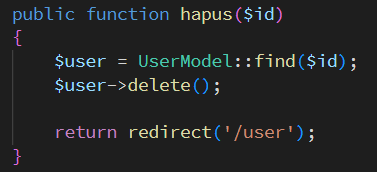
1. Tambahkan *script* pada controller dengan nama file UserController.php. Tambahkan *script* dalam class dan buat method baru dengan nama ubah\_simpan dan diletakan di bawah method ubah seperti gambar di bawah ini



1. Simpan kode program Langkah 15 dan 16. Kemudian jalankan link localhost:8000/user/ubah/1 atau localhost/PWL\_POS/public/user/ubah/1 pada *browser* dan ubah input formnya dan klik tombol ubah, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan
2. Berikut untuk langkah *delete* . Tambahkan *script* pada *routes* dengan nama file web.php. Tambahkan seperti gambar di bawah ini



1. Tambahkan *script* pada controller dengan nama file UserController.php. Tambahkan *script* dalam class dan buat method baru dengan nama hapus dan diletakan di bawah method ubah\_simpan seperti gambar di bawah ini



1. Simpan kode program Langkah 18 dan 19. Kemudian jalankan pada *browser* dan klik tombol hapus, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan
2. Laporkan hasil Praktikum-2.6 ini dan *commit* perubahan pada *git*.

**Praktikum 2.7** – *Relationships*

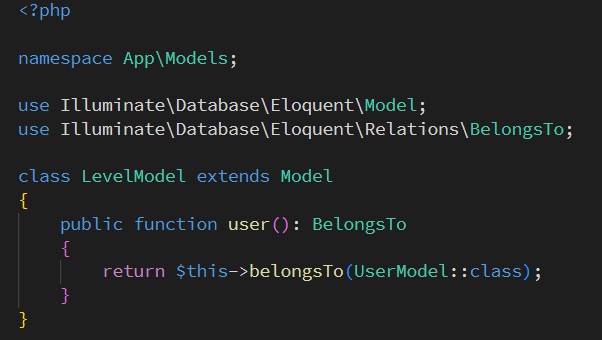
***One to One***

Hubungan satu-ke-satu adalah tipe hubungan database yang sangat mendasar. Misalnya, suatu Usermodel mungkin dikaitkan dengan satu model Levelmodel. Untuk mendefinisikan hubungan ini, kita akan menempatkan Levelmodel metode pada model Usermodel. Metode tersebut Levelmodel harus memanggil hasOne metode tersebut dan mengembalikan hasilnya. Metode ini hasOne tersedia untuk model Anda melalui kelas dasar model Illuminate\Database\Eloquent\Model:



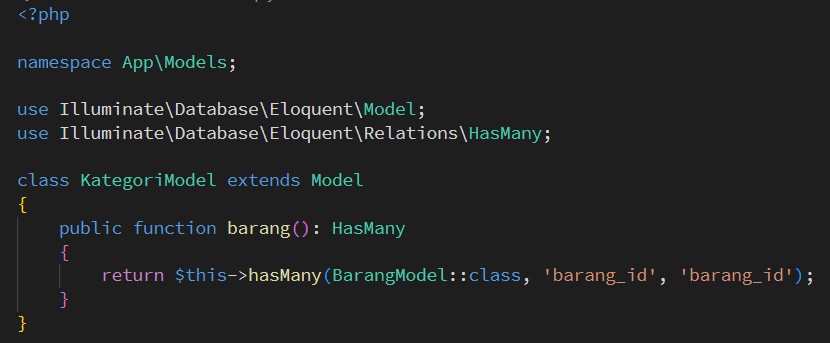
**Mendefinisikan Kebalikan dari Hubungan *One-to-one***

Jadi, kita dapat mengakses model Levelmodel dari model Usermodel kita. Selanjutnya, mari kita tentukan hubungan pada model Levelmodel yang memungkinkan kita mengakses user. Kita dapat mendefinisikan kebalikan dari suatu hasOne hubungan menggunakan belongsTo metode:



## One to Many

Hubungan satu-ke-banyak digunakan untuk mendefinisikan hubungan di mana satu model adalah induk dari satu atau lebih model turunan. Misalnya, 1 kategori mungkin memiliki jumlah barang yang tidak terbatas. Seperti semua hubungan Eloquent lainnya, hubungan satu-ke- banyak ditentukan dengan mendefinisikan metode pada model Eloquent Anda:



## One to Many (Inverse) / Belongs To

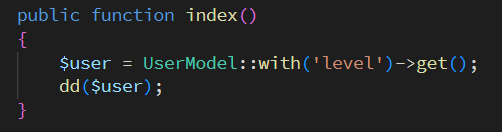
Sekarang kita dapat mengakses semua barang, mari kita tentukan hubungan agar barang dapat mengakses kategori induknya. Untuk menentukan invers suatu hasMany hubungan, tentukan metode hubungan pada model anak yang memanggil belongsTo tersebut:



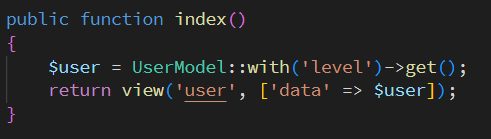
1. Buka file model pada UserModel.php dan tambahkan scritpnya menjadi seperti di bawah ini



1. Buka file controller pada UserController.php dan ubah method *script* menjadi seperti di bawah ini



1. Simpan kode program Langkah 2. Kemudian jalankan link pada *browser*, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan
2. Buka file controller pada UserController.php dan ubah method *script* menjadi seperti di bawah ini



1. Buka file view pada user.blade.php dan ubah *script* menjadi seperti di bawah ini



1. Simpan kode program Langkah 4 dan 5. Kemudian jalankan link pada *browser*, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan
2. Laporkan hasil Praktikum-2.7 ini dan *commit* perubahan pada *git*.

*\*\*\* Sekian, dan selamat belajar \*\*\**