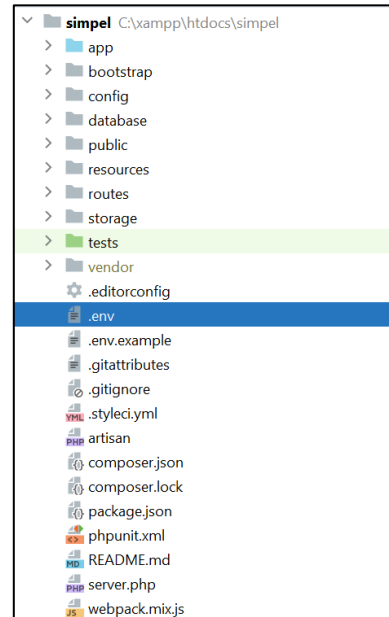


Membuat Model, Migration, dan Seeder

Sebelum membuat model, kita diharuskan untuk mendefinisikan konfigurasi basis data ke dalam proyek Laravel. Silakan buka folder C:/xampp/htdocs/simpel pada text editor misalnya PhpStorm atau Visual Studio Code. Susunan folder Laravel biasanya terdiri dari folder seperti yang tampak pada gambar di samping ini. Buka file .env pada text editor dan atur bagian konfigurasi basis data.

Nama file	.env
Deskripsi	Konfigurasi environment
<pre>... DB_CONNECTION=mysql DB_HOST=127.0.0.1 DB_PORT=3306 DB_DATABASE=simpel DB_USERNAME=root DB_PASSWORD= ...</pre>	

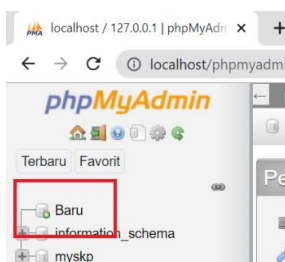


Membuat Basis Data

1. Pada konfigurasi di atas kita mendefinisikan bahwa database engine menggunakan MySQL dengan nama database "simpel". Kemudian buka aplikasi XAMPP Control Panel dan klik start pada MySQL dan Apache seperti di bawah ini.

Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	18520 580	80, 443	Stop
<input type="checkbox"/>	MySQL	20056	3306	Stop

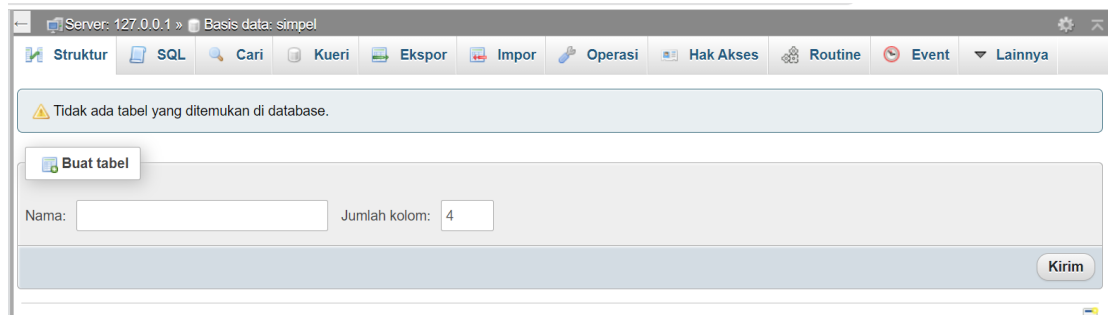
2. Buka browser dan buka halaman <http://localhost/phpmyadmin/> yang akan menuju halaman web phpMyAdmin yang akan kita gunakan untuk membuat basis data pada MySQL. Klik tombol **Baru** pada halaman phpMyAdmin.



3. Ketik "simpel" tanpa petik dan klik tombol Buat pada. Aksi ini akan membuat basis data bernama simpel pada MySQL.



4. Setelah basis data dibuat, maka belum ada tabel yang ditemukan pada basis data tersebut. Untuk membuat tabel, kita akan menggunakan fitur migration dari Laravel.



Bagaimana Cara Membuat Model?

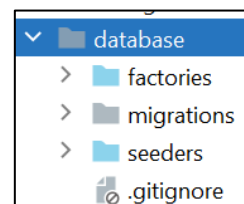
Setelah membuat basis data dan melakukan konfigurasi basis data, mari membuat model Eloquent. Model yang dibuat biasanya disimpan di dalam direktori `app\Models`. Pada desain class diagram sebelumnya telah terdapat 5 model yang harus kita buat. Namun karena Laravel 8 sudah menyertakan model User di dalam paket defaultnya, maka kita hanya membuat 4 model saja. Untuk membuat model cukup mudah apabila menggunakan perintah CLI Artisan `make:model` pada command prompt.

```
php artisan make:model NamaModel
```

Biasanya saat membuat model kita juga perlu membuat migration dan seeder sekaligus. Kita dapat menambahkan opsi `--migration --seed --factory` pada perintah Artisan.

```
php artisan make:model NamaModel --migration --seed --factory
```

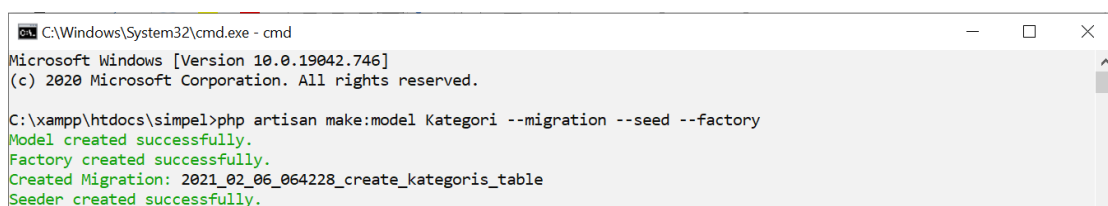
Pada dasarnya **migration** digunakan untuk membuat tabel ke dalam basis data, **seeder** digunakan untuk mengisi data testing ke dalam tabel di basis data, sedangkan **factory** digunakan untuk membuat data-data testing yang dibutuhkan oleh seeder. Ketiga file ini akan disimpan ke dalam folder database.



Membuat Model Kategori

Jangan tutup command prompt sebelumnya dan jalankan perintah artisan di bawah ini untuk membuat model, migration, seeder, dan factory dari entitas Kategori.

```
php artisan make:model Kategori --migration --seed --factory
```



Berdasarkan rancangan Class Diagram sebelumnya terdapat 2 kolom yang harus didefinisikan pada migration yaitu id dan nama. Buka migration kategori yang ada di dalam folder database/migrations. Nama file yang dibuat sama dengan nama yang muncul pada gambar command prompt di atas. Pada method up() edit kode hingga seperti ini.

Kategori	
PK	id
	nama

Nama file	database/migrations/{nama migration}.php
Deskripsi	Mendefinisikan migration untuk model Kategori
<pre>... public function up(){ Schema::create('kategoris', function (Blueprint \$table) { \$table->id(); \$table->string('nama'); \$table->timestamps(); }); } ...</pre>	

1. Kode `$table->id()` ; digunakan untuk membuat kolom id dengan auto-increment dan bertipe data unsigned integer.
2. Kode `$table->string('nama')` ; digunakan untuk membuat kolom nama dengan tipe data string(255).

Pada migration terdapat dua method up dan down. Method up akan dipanggil oleh Artisan secara otomatis saat migration dijalankan. Sedangkan method down dipanggil ketika rollback dijalankan. Rollback merupakan kebalikan dari migration. Jadi rollback digunakan untuk menghapus tabel yang sudah dimigrasikan.

Apabila migration sudah dibuat, edit file factory model Kategori yang terletak yang ada di dalam folder database/factories/KategoriFactory.php. Pada factory terdapat obyek dari class Faker yang dapat digunakan untuk membuat berbagai macam data palsu. File factory pada dasarnya mendefinisikan kolom-kolom pada tabel yang bersesuaian dengan atribut faker. Apabila file sudah terbuka, edit method definition() agar seperti kode di bawah ini.

Nama file	database/factories/KategoriFactory.php
Deskripsi	Mendefinisikan factory untuk model Kategori
<pre>... public function definition(){ return ['nama' => \$this->faker->sentence(2)]; } ...</pre>	

Buka file seeder yang ada di dalam folder database/seeder/KategoriSeeder.php. Pada method run() ubah agar seperti kode di bawah ini.

Nama file	database/seeder/KategoriSeeder.php
Deskripsi	Mendefinisikan seeder untuk model Kategori
<pre>... public function run(){ Kategori::factory() ->count(3) ->create(); } ...</pre>	

Kemudian tambahkan penggunaan kode `use App\Models\Kategori;` pada baris ke-5 sehingga kode pada seeder tampak seperti gambar di samping. Sintaks `use` digunakan untuk mengimpor kelas yang berada di dalam folder lainnya. Sehingga kelas `Kategori` yang berada di folder `app\Models` dapat digunakan di dalam folder seeder.

```
<?php
namespace Database\Seeders;

use App\Models\Kategori;
use Illuminate\Database\Seeder;

class KategoriSeeder extends Seeder
```

Kemudian pada file `database/seeder/DatabaseSeeder.php`, edit method `run()` agar seperti method `run` pada kode di bawah ini. Method `run()` pada dasarnya berguna untuk memanggil kelas `KategoriSeeder` ketika proses seeding berjalan.

Nama file	database/seeder/DatabaseSeeder.php
Deskripsi	Memanggil seeder untuk model <code>Kategori</code>
<pre>... public function run() { \$this->call([KategoriSeeder::class,]); } ...</pre>	

Membuat Model Buku

Buka command prompt yang ada di dalam folder `C:/xampp/htdocs/simpel`. Kemudian jalankan perintah artisan ini untuk membuat model, migration, seeder, dan factory dari entitas `Buku`.

```
php artisan make:model Buku --migration --seed --factory
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\simpel>php artisan make:model Buku --migration --seed --factory
Model created successfully.
Factory created successfully.
Created Migration: 2021_02_06_085056_create_bukus_table
Seeder created successfully.
```

Berdasarkan rancangan Class Diagram sebelumnya terdapat 9 kolom yang harus didefinisikan pada migration yaitu `id`, `kategori_id`, `judul`, `pengarang`, `penerbit`, `tahun`, `isbn`, `stok`, dan `harga`. Buka migration `buku` yang ada di dalam folder `database/migrations`. Nama file yang dibuat sama dengan nama yang muncul pada gambar command prompt di atas. Pada method `up()` edit kode hingga seperti ini.

Nama file	database/migrations/{nama migration}.php
Deskripsi	Mendefinisikan migration untuk model <code>Buku</code>
<pre>... public function up() { Schema::create('bukus', function (Blueprint \$table) { \$table->id(); \$table->foreignId('kategori_id'); \$table->string('judul'); }); }</pre>	

Buku	
FK	<code>kategori_id</code>
PK	<code>id</code>
	<code>judul</code>
	<code>pengarang</code>
	<code>penerbit</code>
	<code>tahun</code>
	<code>isbn</code>
	<code>stok</code>
	<code>harga</code>

```

        $table->string('pengarang');
        $table->string('penerbit');
        $table->year('tahun');
        $table->string('isbn');
        $table->unsignedInteger('stok');
        $table->unsignedInteger('harga');
        $table->timestamps();
    });
}
...

```

Method `foreignId` digunakan untuk membuat kolom foreign key pada tabel. Method `year` digunakan untuk membuat kolom dengan tipe data year. Setelah mengedit migration, buka file factory database/factories/BukuFactory.php dan edit method definition menjadi seperti dibawah ini.

Nama file	database/factories/BukuFactory.php
Deskripsi	Mendefinisikan factory untuk model Buku
<pre> ... public function definition() { return ['judul' => \$this->faker->sentence(4), 'pengarang' => \$this->faker->name, 'penerbit' => \$this->faker->company, 'tahun' => \$this->faker->year, 'isbn' => \$this->faker->isbn13, 'stok' => \$this->faker->randomNumber(2), 'harga' => \$this->faker->randomNumber(2) * 1000,]; } ... </pre>	

Buka file seeder yang ada di dalam folder database/seeder/BukuSeeder.php. Pada method `run()` ubah agar seperti kode di bawah ini.

Nama file	database/seeder/BukuSeeder.php
Deskripsi	Mendefinisikan seeder untuk model Buku
<pre> ... public function run() { Buku::factory() ->count(10) ->for(Kategori::first()) ->create(); } ... </pre>	

Kemudian tambahkan kode `use` seperti di bawah ini ke dalam file seeder seperti contoh sebelumnya pada kelas `Kategori`.

Nama file	database/seeder/BukuSeeder.php
Deskripsi	Mengimpor model Buku dan Kategori pada seeder Buku
<pre> ... use App\Models\Buku; use App\Models\Kategori; ... </pre>	

Tambahkan `BukuSeeder::class` pada method `run` di dalam file `DatabaseSeeder.php` sehingga method `run` tampak seperti di bawah ini. Pada kode dibawah ini `BukuSeeder` akan dipanggil setelah `KategoriSeeder`.

Nama file	database/seeder/DatabaseSeeder.php
Deskripsi	Memanggil seeder untuk model Buku
<pre>... public function run() { \$this->call([KategoriSeeder::class, BukuSeeder::class,]); } ...</pre>	

Seperti pada penjelasan sebelumnya. ORM merupakan Object Relationship Mapper dimana setiap kelas model dapat memetakan relasi dengan model lainnya. Model Kategori memiliki relasi one-to-many dengan model Buku. Buka model Kategori yang terletak pada file app\Models\Kategori.php. Kemudian tambahkan method buku seperti pada kode seperti dibawah ini.

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Kategori extends Model{

    use HasFactory;

    public function buku(){
        return $this->hasMany(related: Buku::class);
    }
}
```

Nama file	app\Models\Kategori.php
Deskripsi	Mendefinisikan relasi kategori dengan buku
<pre>... public function buku() { return \$this->hasMany(Buku::class); } ...</pre>	

Kemudian buka kelas Buku yang terletak pada file app\Models\Buku.php untuk mendefinisikan relasi many-to-one ke model Kategori. Kemudian tambahkan method kategori seperti pada kode di bawah ini.

Nama file	app\Models\Buku.php
Deskripsi	Mendefinisikan relasi buku dengan kategori
<pre>... public function kategori() { return \$this->belongsTo(Kategori::class); } ...</pre>	

Pada kedua class ini tidak perlu menambahkan sintaks use karena model lain yang digunakan berada pada namespace yang sama.

Membuat Model Pembelian

Jangan tutup command prompt sebelumnya dan ketik perintah artisan di bawah ini untuk membuat model dan migration dari entitas Pembelian. Seeder dan factory tidak ditambahkan karena pada model ini tidak diperlukan data palsu.

```
php artisan make:model Pembelian --migration
```

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\simpel>php artisan make:model Pembelian --migration
Model created successfully.
Created Migration: 2021_02_06_122338_create_pembelians_table

```

Berdasarkan rancangan Class Diagram sebelumnya terdapat 4 kolom yang harus didefinisikan pada migration yaitu id, buku_id, stok, dan tanggal. Buka migration pembelian yang ada di dalam folder database/migrations. Nama file yang dibuat sama dengan nama yang muncul pada gambar command prompt di atas. Pada method up() edit kode hingga seperti ini.

Pembelian	
FK	buku_id
PK	id
	stok
	tanggal

Nama file	database/migrations/{nama migration}.php
Deskripsi	Mendefinisikan migration untuk model Pembelian
<pre> ... public function up() { Schema::create('pembelians', function (Blueprint \$table) { \$table->id(); \$table->foreignId('buku_id'); \$table->unsignedInteger('stok'); \$table->timestamps(); }); } ... </pre>	

Pada kode di atas, kolom tanggal di atas tidak dibuat karena sudah terdapat kolom created_at yang ditambahkan melalui kode `$table->timestamps();`. Kemudian kita mendefinisikan relasi model Pembelian dengan model Buku. Model Pembelian memiliki relasi many-to-one dengan model Buku. Buka model Pembelian yang terletak pada file app\Models\Pembelian.php. Kemudian tambahkan method buku seperti pada kode seperti dibawah ini.

Nama file	app\Models\Pembelian.php
Deskripsi	Mendefinisikan relasi pembelian dengan buku
<pre> ... public function buku() { return \$this->belongsTo(Buku::class); } ... </pre>	

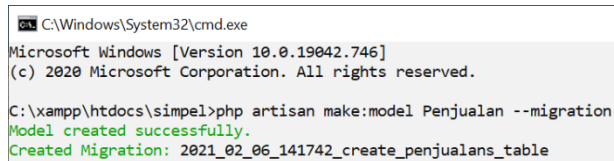
Kemudian buka kelas Buku yang terletak pada file app\Models\Buku.php untuk mendefinisikan relasi one-to-many antara model Buku dengan model Pembelian. Kemudian tambahkan method pembelian seperti pada kode di bawah ini.

Nama file	app\Models\Buku.php
Deskripsi	Mendefinisikan relasi buku dengan pembelian
<pre> ... public function pembelian() { return \$this->hasMany(Pembelian::class); } ... </pre>	

Membuat Model Penjualan

Jangan tutup command prompt sebelumnya dan ketik perintah artisan di bawah ini untuk membuat model dan migration dari entitas Pembelian. Seeder dan factory tidak ditambahkan karena pada model ini tidak diperlukan data palsu.

```
php artisan make:model Penjualan --migration
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\simpel>php artisan make:model Penjualan --migration
Model created successfully.
Created Migration: 2021_02_06_141742_create_penjualans_table
```

Berdasarkan rancangan Class Diagram sebelumnya terdapat 4 kolom yang harus didefinisikan pada migration yaitu id, buku_id, stok, dan tanggal. Buka file migration penjualan yang ada di dalam folder database/migrations. Nama file yang dibuat sama dengan nama yang muncul pada gambar command prompt di atas. Pada method up() edit kode hingga seperti ini.

Penjualan	
FK	buku_id
PK	id
	nama
	stok
	tanggal
	harga

Nama file	database/migrations/{nama migration}.php
Deskripsi	Mendefinisikan migration untuk model Penjualan
<pre>... public function up() { Schema::create('penjualans', function (Blueprint \$table) { \$table->id(); \$table->foreignId('buku_id'); \$table->string('nama'); \$table->unsignedInteger('stok'); \$table->unsignedInteger('harga'); \$table->timestamps(); }); } ...</pre>	

Pada kode di atas, kolom tanggal di atas tidak dibuat karena sudah terdapat kolom created_at yang ditambahkan melalui kode `$table->timestamps();`. Kemudian kita mendefinisikan relasi model Penjualan dengan model Buku. Model Penjualan memiliki relasi many-to-one dengan model Buku. Buka model Penjualan yang terletak pada file app\Models\Penjualan.php. Kemudian tambahkan method buku seperti pada kode seperti dibawah ini.

Nama file	app\Models\Penjualan.php
Deskripsi	Mendefinisikan relasi penjualan dengan buku
<pre>... public function buku() { return \$this->belongsTo(Buku::class); } ...</pre>	

Kemudian buka kelas Buku yang terletak pada file app\Models\Buku.php untuk mendefinisikan relasi one-to-many antara model Buku dengan model Penjualan. Kemudian tambahkan method penjualan seperti pada kode di bawah ini.

Nama file	app\Models\Buku.php
-----------	---------------------

Deskripsi	Mendefinisikan relasi buku dengan penjualan
<pre>... public function penjualan() { return \$this->hasMany(Penjualan::class); } ...</pre>	

Membuat Migration User

Buka DatabaseSeeder.php yang ada di dalam folder database\seeders dan edit method run() menjadi kode di bawah ini. Tambahkan `use App\Models\User;` pada baris ke-5 untuk mengimpor kelas User pada kelas DatabaseSeeder.

Nama file	database/seeder/DatabaseSeeder.php
Deskripsi	Memanggil seeder untuk model User
<pre>... public function run() { User::factory()->create(['email' => 'admin@gmail.com']); \$this->call([KategoriSeeder::class, BukuSeeder::class,]); } ...</pre>	

Menjalankan Migration dan Seeder

Untuk menjalankan migration dan seeder eksekusi perintah Artisan di bawah ini pada command prompt. Fungsi migrate:fresh pada dasarnya menghapus semua tabel yang ada pada basis data dan melakukan migration. Opsi --seed digunakan untuk menjalankan proses seeding setelah proses migration selesai.

```
php artisan migrate:fresh --seed
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\simpel>php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (39.31ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (37.76ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (30.89ms)
Migrating: 2021_02_06_064228_create_kategoris_table
Migrated: 2021_02_06_064228_create_kategoris_table (19.06ms)
Migrating: 2021_02_06_085056_create_bukus_table
Migrated: 2021_02_06_085056_create_bukus_table (20.95ms)
Migrating: 2021_02_06_122338_create_pembelians_table
Migrated: 2021_02_06_122338_create_pembelians_table (19.93ms)
Migrating: 2021_02_06_141742_create_penjualans_table
Migrated: 2021_02_06_141742_create_penjualans_table (21.09ms)
Seeding: Database\Seeders\KategoriSeeder
Seeded: Database\Seeders\KategoriSeeder (6.39ms)
Seeding: Database\Seeders\BukuSeeder
Seeded: Database\Seeders\BukuSeeder (20.09ms)
Database seeding completed successfully.
```