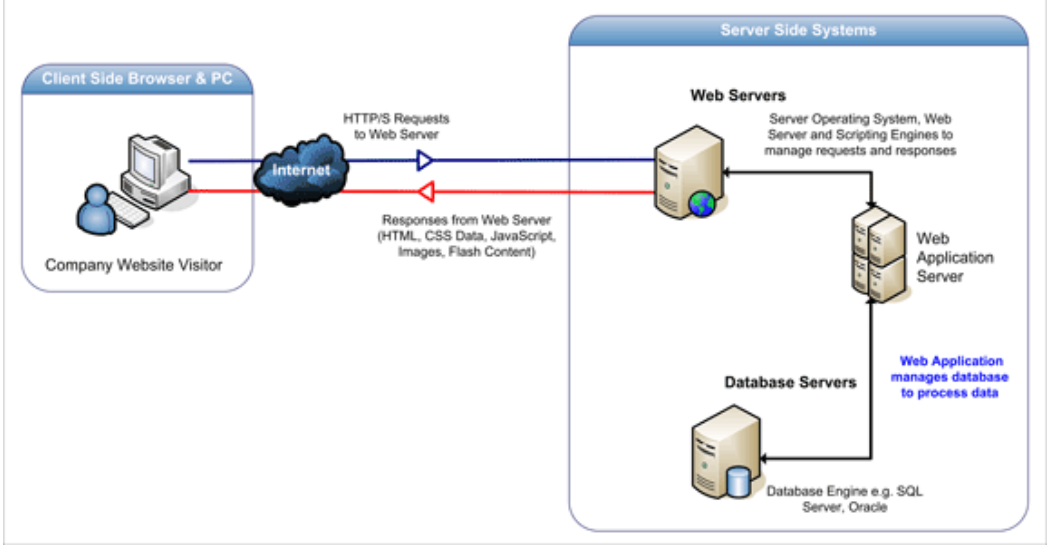
Ringkasan UTS Web Programming

1. Aplikasi web → kumpulan halaman web yang digenerate sebagai respon dari user request.

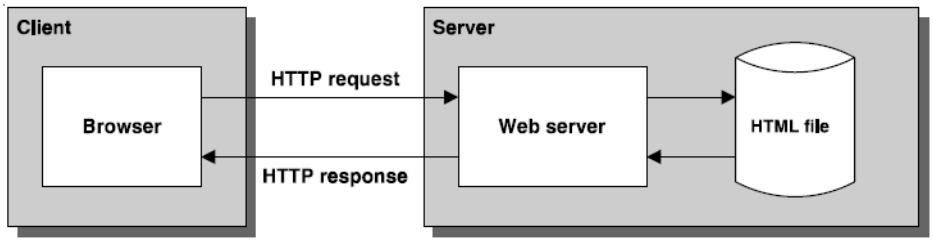
2. Contoh webapp → search engine, online stores, auctions, games, news sites, diskusi grup.

3. Komponen aplikasi web :

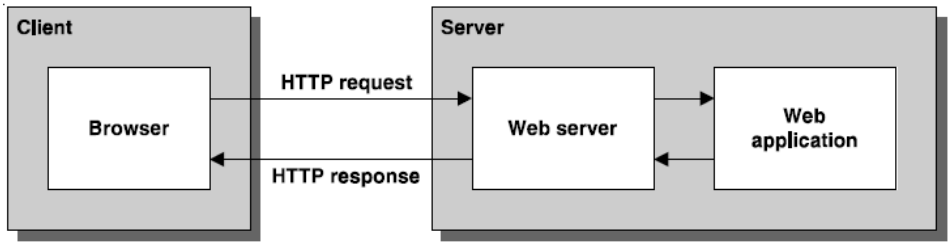


4. Web server bisa terdiri dari web application server dan database server.

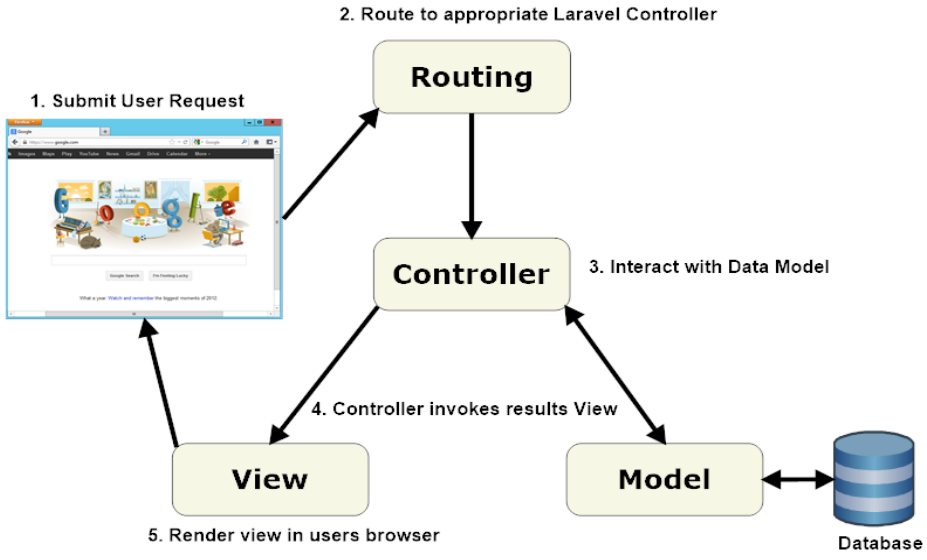
5. Jenis halaman web :
    - Statis → web yang tampilannya sama / tidak ada perbedaan walau ada input. Hanya bisa diubah secara manual. Biasanya menggunakan HTML, CSS, JS.



- Dynamic → web yang konten / tampilannya bisa berubah berdasarkan input / parameter. Biasanya menggunakan : JSP, PHP.



6. Ada 2 arsitektur aplikasi web :
   - Model 1 architecture → PHP bertanggung jawab atas seluruh pekerjaan.
   - MVC (Model View Controller) :



7. Model → domain dimana software kita dibuat. Ini berdasarkan real – world item seperti orang, produk, akun bank. Model

sifatnya permanen dan berada diluar aplikasi, biasanya ada di database. Selain itu business rules yang diapply ke data juga terdapat disini.

8. View → representasi visual dari sebuah model. Biasanya merupakan hasil markup render browser, seperti HTML. Layer ini berfungsi untuk generate user interface berdasar data di model.

9. Controller → penghubung antara view dan model. Controller bertanggungjawab untuk proses input, act berdasarkan model, dan menentukan action apa yang harus dilakukan, misal render view, redirect ke page lain.

10. Laravel adalah sebuah framework aplikasi web berbasis PHP. Laravel bertujuan untuk membuat proses development tanpa mengorbankan fungsionalitas aplikasi.

11. Data models :
    - Mendeskripsikan business rules aplikasi kita.
    - Setiap piece data direpresentasikan dalam database tabel.
    - Laravel menghubungkan data model aplikasi dan tabel database dengan mengubah baris tabel dari database ke object PHP yang mudah dimanipulasi.

12. Beachhead market → strategi dagang yang diambil dari strategi militer berdasarkan lokasi yang dikuasai. Ada 7 kriteria :
    - Apakah target customer memiliki uang? → jika tidak, maka kurang atraktif karena kurang menyediakan cash flow untuk venture baru untuk tumbuh.

- Apakah target customer siap akses sales forces? (apakah mudah ditangani?) → apakah akses ke produk kita bisa dicapai dengan mudah, bisa menggunakan pihak ketiga.
- Apakah target customer punya alasan untuk membeli? → apa yang membuat produk kita lebih dipiluh dibanding solusi yang lainnya yang mirip.
- Bisakah deliver produk, dengan bantuan partner?
- Apakah persaingan yang ketat menghalangi? → melihat seberapa kuat kompetitor kita dari customer viewpoint.
- Jika memenangkan segment ini, bisakah diperluas untuk masuk ke segment tambahan? → jika menang segmen ini, peluang untuk diperluas ke segment tambahan sangat memungkinkan untuk pertumbuhan bisnis.
- Apakah pasar konsisten dengan nilai, passion, dan tujuan founding team? → memastikan founder personal goals tidak berbalik arah dari kriteria – kriteria ini.

13. Customer centric business model design :
    - Apa yang customer butuh diselesaikan dan apa yang bisa dibantu?
    - Apa saja aspirasi dari customer dan bagaimana kita bisa membantu?
    - Bagaimana customer ingin dialamatkan?
    - Hubungan apa yang customer inginkan untuk jalin dengan kita?
    - Apa yang membuat customer mau membayar untuk produk kita?

14. Organization centric business model design :
    - Apa yang bisa kita jual ke customer?

- Bagaimana menjangkau customer dengan efisien?
- Hubungan apa yang harus kita jalin dengan customer?
- Bagaimana menerima keuntungan / uang dari customer?

15. Segmentasi pasar :
- Brainstorming → mencatat ide – ide yang terpikirkan untuk bisnis.
- Persempit ide menjadi beberapa bagian → dengan menggunakan 7 kriteria beachhead.
- Lakukan market research → bicara langsung untuk mendapat insight peluang pasar mana yang terbaik.

16. Tipe customer :
- End user → orang yang menggunakan produk kita (pengguna produk).
- Decision making unit → orang yang menentukan apakah customer akan membeli produk :
  - Champion → orang yang ingin customer membeli produk, seringkali end user.
  - Primary economic buyer → orang dengan otoritas membayar untuk membeli sebuah produk, sesekali adalah end user.
  - Influencer, veto power, purchasing department → orang yang punya control langsung terhadap keputusan primary economic buyer.

17. Service providers :
- Merupakan koneksi antara package dan laravel.
- Bertanggung jawab untuk binding ke laravel service container dan menginformasikan laravel dimana harus load

package resources seperti view, configuration, localization file.

- Punya 2 method : register dan boot dan teletak di Illuminate\Support\ServiceProvider.

18. Konfigurasi :
    - Harus publish konfigurasi package ke config aplikasi agar user bisa override default configuration option.
    - Ketika user eksekusi vendor:publish maka file akan ter-copy ke publish location.
    - Default package configuration → bisa di merge dengan menggunakan mergeConfigFrom di fungsi register di service provider.

19. Routing → untuk mengarahkan suatu aplikasi web menuju ke halaman web tertentu.

20. Views :
    - Untuk register package view dengan laravel, menggunakan loadViewsFrom untuk memberitahu laravel dimana lokasi view.
    - Package view direferensikan dengan package::syntax convention. Sekali view path di register, view bisa di load dari package.
    - Mempublish view → bisa menggunakan service provider method yang mengizinkan array of package melihat path dan location. Caranya dengan eksekusi vendor:publish, package view akan ter-copy ke specified publish location.

21. Commands → untuk register package artisan command, gunakan method commands. Method ini mengembalikan array of command class name.

22. Public assets :
    - Bisa mengandung image, css, javascript.
    - Akses bisa melalui public atau menggunakan service provider publish method.
    - Ketika package user eksekusi vendor:publish, asset akan dicopy ke specified publish location.
    - Ketika kita butuh overwrite asset (setiap kali update), gunakan --force.

23. Basic routing :
    - Route berada di app/Http/routes.php
    - Menerima URL dan closure.

```
Route::get('foo', function () {
    return 'Hello World';
});
```

    - Method routes yang tersedia

```
Route::get($uri, $callback);
Route::post($uri, $callback);
Route::put($uri, $callback);
Route::patch($uri, $callback);
Route::delete($uri, $callback);
Route::options($uri, $callback);
```

    - Bisa menggunakan match atau any ketika kita butuh register route yang respon ke multiple http verbs

```php
Route::match(['get', 'post'], '/', function () {
    //
});

Route::any('foo', function () {
    //
});
```

24. Route parameter :

- Required parameter → ketika butuh capture segmen dari URL dalam route kita.

```php
Route::get('user/{id}', function ($id) {
    return 'User '.$id;
});
```

- Optional parameter → ketika butuh menspesifikasikan parameter, tetapi hal ini optional. Menggunakan tanda ? setelah nama parameter.

```php
Route::get('user/{name?}', function ($name = null) {
    return $name;
});

Route::get('user/{name?}', function ($name = 'John')
    return $name;
});
```

- Regular expression constraints → format route parameter menggunakan where method di route instance. Ini menerima nama parameter dan regular expression yang mendefinisikan bagaimana parameter dibatasi.

```php
Route::get('user/{name}', function ($name) {
    //
})
->where('name', '[A-Za-z]+');

Route::get('user/{id}', function ($id) {
    //
})
->where('id', '[0-9]+');

Route::get('user/{id}/{name}', function ($id, $name) {
    //
})
->where(['id' => '[0-9]+', 'name' => '[a-z]+']);
```

- Regular expression constraints – global constraints → jika ingin parameter route selalu di constraints dengan memberi regular expression, bisa menggunakan pattern method.

```php
/**
 * Define your route model bindings, pattern filters, etc.
 *
 * @param  \Illuminate\Routing\Router  $router
 * @return void
 */
public function boot(Router $router)
{
    $router->pattern('id', '[0-9]+');

    parent::boot($router);
}
```

```php
Route::get('user/{id}', function ($id) {
    // Only called if {id} is numeric.
});
```

25. Named routes :

- Mengizinkan generate URL atau redirect route tertentu.
- Mungkin untuk specify nama route menggunakan array key.

```
Route::get('user/profile', ['as' => 'profile', function () {
    //
}]);
```

- Bisa juga specify nama route untuk controller actions.

```
Route::get('user/profile', [
    'as' => 'profile', 'uses' => 'UserController@showProfile'
]);
```

- Bisa merantai nama method ke akhir definisi route (cara alternative).

```
Route::get('user/profile', 'UserController@showProfile')->name('profile');
```

- Route groups → specify sebagai keyword di route attribute array, mengizinkan set common route name prefix untuk semau route kedalam group.

```
Route::group(['as' => 'admin::'], function () {
    Route::get('dashboard', ['as' => 'dashboard', function () {
        // Route named "admin::dashboard"
    }]);
});
```

- Generate URL ke named route → bisa menggunakan nama route ketika generate URL atau redirect lewat global route functions. Jika ada parameter, maka parameter dilewati sebagai argumen kedua ke route functions.

```
// Generating URLs...
$url = route('profile');


// Generating Redirects...
return redirect()->route('profile');
```

```
Route::get('user/{id}/profile', ['as' => 'profile', function ($id) {
    //
}]);

$url = route('profile', ['id' => 1]);
```

26. Route groups :

- Mengizinkan share atribut route, seperti middleware, namespace melalui banyak route tanpa perlu definisikan attribute di setiap route.
- Menggunakan Route::group method.
- Middleware → middleware dieksekusi ketika kita mendefinisikan ini.

```
Route::group(['middleware' => 'auth'], function () {
    Route::get('/', function ()    {
        // Uses Auth Middleware
    });

    Route::get('user/profile', function () {
        // Uses Auth Middleware
    });
});
```

- Namespace → assign PHP namespace ke group controller. Bisa menggunakan namespace parameter di group atribut array untuk specify namespace semua controller didalam group.

```
Route::group(['namespace' => 'Admin'], function()
{
    // Controllers Within The "App\Http\Controllers\Admin" Namespace

    Route::group(['namespace' => 'User'], function() {
        // Controllers Within The "App\Http\Controllers\Admin\User" Namespace
    });
});
```

- Sub domain routing → bisa di assign untuk route parameter seperti URL, mengizinkan untuk capture porsi subdomain untuk penggunaan di route / controller.

```
Route::group(['domain' => '{account}.myapp.com'], function () {
    Route::get('user/{id}', function ($account, $id) {
        //
    });
});
```

- Route prefix → digunakan untuk prefix setiap route dalam group dengan URL yang diberikan. Contoh ingin prefix route dalam group admin. Selain itu juga bisa menggunakan prefix parameter untuk specify common parameter untuk grouped routes.

```
Route::group(['prefix' => 'admin'], function () {
    Route::get('users', function ()    {
        // Matches The "/admin/users" URL
    });
});
```

```
Route::group(['prefix' => 'accounts/{account_id}'], function () {
    Route::get('detail', function ($accountId)    {
        // Matches The "/accounts/{account_id}/detail" URL
    });
});
```

27. Laravel tidak terikat dengan penggunaan JavaScript atau CSS preprocessor, tetapi menyediakan basic starting point berupa Bootstrap dan Vue. Menggunakan NPM untuk menginstall kedua package tersebut.

28. CSS → Laravel Mix menyediakan expressive dan clean API dimana ekstensi plain CSS yang menambahkan variabel, mixins, dan fitur lainnya yang membuat bekerja dengan CSS lebih mudah. Dicompile dengan SASS atau Less.

29. JavaScript → Laravel tidak mewajibkan framework tertentu untuk JavaScript. Sudah ada basic scaffolding menggunakan library Vue. Kita bisa menggunakan Laravel Mix untuk compile komponen JavaScript ke single, browser – ready JavaScript file.

30. Menulis CSS :

- Ada di package.json yang melibatkan bootstrap.
- Menggunakan npm install dan npm run dev.
- Berada di resources/assets/saas/app.scss.
- App.scss mengimpor file SASS variabel dan load bootstrap. Ini bisa dicustomize.

31. Menulis JavaScript :
- Ada di package.json
- Menggunakan npm install dan npm run dev.
- Berada di resources/assets/js/app.js.
- Di app.js bisa register komponen Vue, atau bisa juga konfigurasi aplikasi JavaScript kita sendiri.
- Hasil compile berada di public/js directory.

32. Menulis komponen Vue :
- Secara default, aplikasi laravel mengandung Example.vue, dimana komponen Vue ada di resources/assets/js/components.
- Example.vue adalah contoh single file vue component yang mendefinisikan JavaScript dan template HTML di file yang sama.
- Single file component mempermudah membuat JavaScript driven application. Caranya :

```
Vue.component('example', require('./components/Example.vue'));
```

- Untuk menggunakan komponen Vue, bisa di drop ke HTML template.
- Contohnya setelah kita run make:auth untuk scaffold autentikasi aplikasi dan registration screen, bisa drop komponen ke home.blade.php.

```
@extends('layouts.app')


@section('content')
    <example></example>
@endsection
```

33. Alasan kita membuat map untuk memperoleh customer :

- Memahami panjang sebuah siklus sales
  - Hal ini berpengaruh terhadap seberapa mahal bagi kita dalam memperoleh customer.
  - Memproyeksikan alur cash secara akurat.
  - Initial contact untuk membayar customer secara cepat untuk membuat bisnis yang baik.
- Membangun dasar untuk perhitungan cost dari customer yang diperoleh
  - Kita akan mencapai titik dimana kita memperoleh uang lebih dari customer yang ada saat ini dibanding customer baru.
  - Biaya selalu lebih mahal ketika kita berpikir untuk memproleh customer baru.
- Identifikasi hambatan tersembunyi yang akan menghalangi kemampuan untuk menjual produk dan dibayar
  - Jika ada sesuatu tentang bisnis kita yang merupakan deal breaker, kita harus mengetahuinya diawal dibanding kita telah secara penuh komit terhadap bisnis kta, menghasilkan uang, dan merekrut karyawan.
- Mampu menunjukkan investor yang potensial yang memahami proses customer membeli

- Prasyarat dalam investing bisnis kita.

34. Faktor – faktor life cycle use case adalah sebagai berikut. Untuk menggambarkannya, biasanya menggunakan diagram, flowchart, atau metode yang menunjukkan suatu sequence.
    - Bagaimana end user menentukan mereka butuh kesempatan melakukan sesuatu yang berbeda.
    - Bagaimana mereka menemukan tentang produk kita.
    - Bagaimana mereka analisis produk kita.
    - Bagaimana mereka memperoleh produk kita.
    - Bagaimana mereka install produk kita.
    - Bagaimana mereka menggunakan produk kita secara detail.
    - Bagaimana mereka menentukan value yang didapat dari produk kita.
    - Bagaimana mereka membayar untuk produk kita.
    - Bagaimana mereka menerima support untuk produk kita.
    - Bagaimana mereka membeli lebih banyak produk atau menyebarkan hal (dengan harapan hal positif) tentang produk kita.

35. Alasan kita harus menentukan 10 customer selanjutnya :
    - Membuat daftar lebih dari 10 customer potensial → customer ini harus yang kira – kira akan membeli produk kita. Dalam list ini diusahakan homogen.
    - Hubungi setiap customer potensial di list → pastikan kita dalam mode inquiry, bukan dalam mode sales. Tentukan apakah kebutuhan customer dan ide produk kita sejalan.
    - Jika customer validasi hipotesis kita pada tahap sebelumnya, tanyakan kepada customer apakah mereka berniat untuk membeli produk kita. Jika mereka antusias, kita bisa

menanyakan untuk *prepay* (semacam memberi uang muka) produk kita dimana ini merupakan level fantastic untuk komitmen.

- Jika customer feedback tidak sesuai asumsi kita, pikirkan bagaimana ini berefek pada analisis kita. Jangan berlebihan setiap percakapan baru walaupun ada ketidakseinambungan jika kita tidak menemukan pattern. Kita akan mengetahui major disconnect ketika beberapa interview.
- Setelah memiliki data, kita bisa modifikasi asumsi awal dan menentukan contact customer tambahan. Tujuannya adalah daftar 10 customer homogen yang tertarik pada produk kita.
- Jika tidak bisa mendaftarkan 10 customer yang tertarik pada produk kita, kita mungkin bisa meninjau ulang pasar.
- Jangan menyebarkan list customer atau informasi yang kita telah dapat ke luar perusahaan.

36. Hal yang harus dilakukan dalam menghadapi negative impact :

- Ketika setiap langkah menghasilkan feedback negative (feedback yang tidak support hipotesis kita), kita telah mendapat informasi berguna yang menunjukkan mungkin terdapat kesalahan pada riset dan data yang digunakan.
- Hasil negative pada satu step bukan berarti akhir dari venture pada kebanyakkan kasus. Akan tetapi, maju terus dengan plan yang salah yang berdasarkan harapan, bukan fakta dapat menyebabkan kegagalan. Untuk itu harus sesuai fakta.
- Melihat peluang yang memungkinkan yang orang lain tidak punya dan menyelesaikan hambatan yang orang lain tidak

bisa. Dalam tahap ini, customer proses berpusat pada permainan.

37. Konfigurasi SQL Server :

- Laravel support SQL Server. Bisa digunakan ketika kita membuat koneksinya. Caranya :

```
'sqlsrv' => [
    'driver' => 'sqlsrv',
    'host' => env('DB_HOST', 'localhost'),
    'database' => env('DB_DATABASE', 'forge'),
    'username' => env('DB_USERNAME', 'forge'),
    'password' => env('DB_PASSWORD', ''),
    'charset' => 'utf8',
    'prefix' => '',
],
```

38. Koneksi read dan write :

- Query database transaction (SELECT, INSERT, UPDATE, DELETE) di laravel bisa menggunakan Query Builder atau Eloquent ORM.
- 2 key yang telah ditambahkan ke array konfigurasi : read dan write. Keduanya mengandung single key yang namanya host. Sisanya akan di merge dari main mysql array.
- Credential, prefix, character set di mysql array akan di share pada connection read dan write.

39. Menggunakan multiple koneksi database :

- Akses lewat connection method di DB facade. Nama yang melalui method connection harus berhubungan dengan connection yang ada di config/database.php

```
$users = DB::connection('foo')->select(...);
```

- Bisa juga menggunakan raw, underlying PDO instance menggunakan getPDO di instance connection

```
$pdo = DB::connection()->getPdo();
```

40. Menjalankan raw SQL query :

- Bisa dilakukan sesudah konfigurasi koneksi database.
- Menjalankan query menggunakan DB facade.
- Method yang tersedia : select, insert, update, delete, statement.
- Argumen pertama melewati select method adalah raw sql queries, argumen kedua adalah parameter binding yang akan di bound ke query. Biasanya jika ada where, dan parameter binding memberikan proteksi dari SQL injection.
- Method select selalu mereturn result berupa array, sehingga jika kita ingin menampilkan, kita menggunakan foreach.

- **Using Named Bindings**

Instead of using **?** to represent your parameter bindings, you may execute a query using named bindings:

```
$results = DB::select('select * from users where id = :id', ['id' => 1]);
```

- **Running An Insert Statement**

To execute an **insert** statement, you may use the **insert** method on the **DB** facade. Like **select**, this method takes the raw SQL query as its first argument and bindings as its second argument:

```
DB::insert('insert into users (id, name) values (?, ?)', [1, 'Dayle']);
```

- **Running An Update Statement**

The update method should be used to update existing records in the database. The number of rows affected by the statement will be returned:

```
$affected = DB::update('update users set votes = 100 where name = ?', ['John']);
```

- **Running A Delete Statement**

The **delete** method should be used to delete records from the database. Like **update**, the number of rows affected will be returned:

```
$deleted = DB::delete('delete from users');
```

- **Running A General Statement**

Some database statements do not return any value. For these types of operations, you may use the **statement** method on the **DB** facade:

```
DB::statement('drop table users');
```

41. Untuk run kumpulan operasi pada database transaction kita bisa gunakan method transactions. Jika ada exception pada closure, maka akan otomatis di rollback. Jika sukses, maka secara otomatis akan commit.

```php
DB::transaction(function () {
    DB::table('users')->update(['votes' => 1]);

    DB::table('posts')->delete();
});
```

42. Menggunakan transactions secara manual :
    - Untuk memulai transaksi sehingga bisa di commit atau rollback manual → beginTransaction

      ```php
      DB::beginTransaction();
      ```
    - Untuk rollback transaction → menggunakan rollback method.

      ```php
      DB::rollBack();
      ```
    - Untuk commit transaction → menggunakan commit method.

      ```php
      DB::commit();
      ```

43. Paginate hasil query builder :
    - Menggunakan paginate method di query builder atau eloquent.
    - Method paginate secara otomatis akan set limit dari page yang akan di view user.
    - Secara default, current page dideteksi melalui value page query string argument pada HTTP request.
    - Value secara otomatis dideteksi oleh laravel dan secara otomatis dimasukkan ke links yang digenerage paginator.

44. Ada 2 tipe pagination :

- Simple pagination → jika link yang ditampilkan hanya next atau previous, menggunakan method simplePaginate. Contoh ini, tampilkan 15 data per halaman. Ini digunakan ketika perform efficient query.

```
$users = DB::table('users')->simplePaginate(15);
```

- Paginate → menampilkan berapa halaman yang ada dari hasil paginate (bisa lompat halaman). Bisa juga di set menggunakan constraint. Contoh dibawah ini :

```
$users = User::where('votes', '>', 100)->paginate(15);
```

  - Paginate dengan query builder

```
$users = App\User::paginate(15);
```

45. Secara manual membuat paginator :

- Menggunakan Illuminate\Pagination\LengthAwarePaginator.
- Memakai class Paginator, dimana class ini tidak harus tau berapa item yang ada. Jadi method ini tidak punya method untuk retrieve index halaman terakhir.
- LengthAwarePaginator menerima hampir semua argumen seperti Paginator, ini membutuhkan banyaknya item yang ada di result set.
- Paginator berhubungan dengan simplePaginate, LengthAwarePaginator berhubungan dengan paginate.
- Method pada paginator instance :

- `$results->count()`
- `$results->currentPage()`
- `$results->firstItem()`
- `$results->hasMorePages()`
- `$results->lastItem()`
- `$results->lastPage()` (Not available when using simplePaginate)
- `$results->nextPageUrl()`
- `$results->perPage()`
- `$results->previousPageUrl()`
- `$results->total()` (Not available when using simplePaginate)
- `$results->url($page)`

46. Value proposition → statement simpel dan jelas mengenai benefit dimana perusahaan memberikan harga untuk benefit yang diterima customer.

47. Menentukan value proposition :
    - What → mengkonversi benefit yang customer dapatkan dari produk ke metrik yang terwujud ataupun tidak sesuai dengan prioritas customer. Contoh : kepuasan konsumen.
    - Goal → menyatakan dengan ringan dan jelas bagaimana produk benefit sejalan dengan apa yang customer inginkan / ingin tingkatkan.
    - Focus → apa yang customer potensial ingin dapatkan dibanding fokus ke detail, teknologi, fitur, dan fungsi. Menjelaskan apa produk yang ditawarkan.

48. Inti dari value proposition :
    - Diperoleh dari top priority customer. Harus mengerti dan map customer.

- Map out state yang memungkinkan dari produk. Indikasikan dengan jelas dimana customer menerima value berdasarkan customer top priority.
- Salah satu cara adalah diagram 1 halaman secara visual, agar customer mudah melihat value proposition dan memperlihatkan ke yang lain validasinya.
- Ketika semua telah dilakukan, ini akan menjadi value bagi kita untuk melaunching bisnis. Effort extra dibutuhkan untuk mengoptimisasi ini.