Pembuatan CMS Admin

Notes:

code :

```
if($request->file('picturePath'))
{
    $data['picturePath'] = $request->file('picturePath')
                        ->store('assets\food', 'public');
}
```

block code ini berfungsi untuk memindahkan data request bertipe file ke dalam project kita, karena sebenernya data kita itu berada di project xampp . bukan di dalam project kita .

Menambahkan Middleware Admin

di sini , pembuatan CMS admin digunakan agar yang b isa masuk ke dalam dashboard dan management data hanya ADMIN, caranya menggunakan **Middleware**,

1. MyApplication>$ php artisan make:middleware IsAdmin
2. Daftarkan middleware tersebut ke dalam **Kernel.php**

   ```
   protected $routeMiddleware =[
       'admin' => IsAdmin;
   ];
   ```

3. Pada File **app\Http\Middleware\IsAdmin.php** yang sudah dibuat di nomor 1 .

   buat condisi, return Next nya hanya bisa di jalankan oleh User yang sedang login dan User yang sedang login tsb roles nya admin.

   ```
   public function handle(Request $request, Closure $next)
   {
       if(Auth::user() && Auth::user()->roles =="ADMIN")
       {
           return $next($request);
       }
       else{
           return redirect('/');
       }
   }
   ```

Membuat Halaman Dashboard

kita ingin menampilkan halaman dashboard, tapi kita custom tidak menggunakan bawaan laravel, tapi pakai controller **DashboardController**

1. ```
   MyApplication>$ php artisan make:controller DashboardController
   ```

2. Buat function index untuk return view halaman dashboard
3. Edit routenya , tambahkan route tersendiri,
   - dengan prefix dashboard
   - menggunakan middleware auth sunctum 'admin'
   - group endpointnya
   - masukkan link dan controller dan function serta beri nama 'dashboard'

   ```
   Route::prefix('dashboard')->middleware('auth:sanctum')->group(function(){
       Route::get('/', [DashboardController::class, 'index'])-
   >name('dashboard');
   });
   ```

4. Buat route get '/' dengan fungsi mengarah ke dalam route yang telah di buat di nomor 3

   ```
   Route::get('/', function(){
       return redirect()->route('dashboard');
   });
   ```

5. Untuk mendapatkan templatenya dan mengeditnya, jalankan :

   ```
   MyApplication>$ php artisan vendor:publish --tag=jetsream-views
   ```

   kenapa harus di jalankan ini, karena dashboard view itu adalah bagian dari welcome.blade.php,

   tetapi kita tidak punya file welcome.blade.php , agar bisa dapat dan diconfigurasi, kita harus menjalankan command di atas.

6. Edit Dashboardnya , sesuai keinginan,

BE 4.3.1 CRUD USER (CONTROLLER DAN INDEX)

Basic VIEW

bagian ini , kita akan memasuki hubungan antara **Controller** dan **View** , basic dari view yaitu :

1. **Setiap Link** itu pasti menggunakan **ROUTE** atau **FORM**
2. Button **Delete** selalu menggunakan **FORM**
3. **Tag form** selalu memiliki **ACTION , METHOD DAN CSRF**
4. **Table** selalu menggunakan **foreach**
5. Tombol **edit/view** selalu menggunakan parameter **id**

Alur pembuatannya yaitu :
   ```

1. Buat Controllernya dengan resource
2. daftarkan controllernya di route
3. Edit function dari controller dan view nya

1. ```
   MyApps > php artisan make:controller UserController --resource
   ```

2. ```
   Route::resource('users', UserController::class);
   ```

3. resource\view\users\index.blade.php:

   yang penting penting yaitu :

   ```
   //link selalu menggunakan route

   <a href="{{ route('users.create') }}">+ Create User</a>

   <a href="{{ route('users.edit', $item->id) }}">Edit</a>

   //delete selalu menggunakan tag form
   <form action="{{ route('users.destroy', $item->id) }}" method="POST">
       {!! method_field('delete') . csrf_field() !!}
       <button type="submit"> Delete</button>
   </form>

   //return list table selalu menggunakan forelse as something
   @forelse($user as $item)
       <td class="border px-6 py-4 ">{{ $item->name }}</td>
   @empty
        Data Tidak Ditemukan
   @endforelse
   ```

   cara menampilkan paginate : **{{ $user->links() }}**

4. untuk public function index nya :

   ```
   public function index()
   {
       $user = User::paginate(10);

       return view('users.index', ['user'=>$user]);
   }
   ```

LENGKAPNYA

resource\view\users\index.blade.php:

```
https://github.com/belajarkoding/foodmarket-
backend/blob/master/resources/views/users/index.blade.php
```

BE 4.3.2 FORM REQUEST CREATE AND SAVE

kenapa harus buat request ?

karena , kita harus memasukan requirements apa aja yang wajib diisi oleh si user, contoh, jika user masukin email tanpa icon **@** itu akan return error.

Alur Video :

1. Buat Request namanya UserRequest untuk setting requirements form dan setting UserRequest.php nya
2. Setting store controller di userscontroller dan setting data store image
3. buat users.create view

1. ```
   MyApps >$ php artisan make:request UserRequest
   ```

   app\Http\Requests\UserRequest.php

   ```
   use PasswordValidationRules();

   public function authorize()
   {
       return true; //ini wajib diganti jadi true
   }

   public function rules()
   {
       //setting rules nya di sini
       return [
           'name'  => ['required', 'string', 'max:255'],
           'email' => ['required', 'string', 'email', 'max:255',
   'unique:users'],
           'password' => $this -> passwordRules(), // ini harus import
   librarnya dulu
           'address' => ['required', 'string'],
           'roles' => ['required', 'string', 'max:255', 'in:USER, ADMIN'],
           'houseNumber'=> ['required', 'string', 'max:255'],
           'phoneNumber'=> ['required', 'string', 'max:255'],
           'cityNumber'=> ['required', 'string', 'max:255'],
       ]
   }
   ```

2. app\Http\Controllers\UserController.php
   ```

```
public function create(){
    return view('users.create');
}

public function store(UserRequest $request){
    $data = $request -> all();
    $data['photo_profile_path'] = $requst->file('photo_profile_path')
                                    ->store('assets/user', 'public');

    User::create($data);
    return redirect()->route('users.index');
}
```

3. create file : resources\views\users\create.blade.php

   yang perlu di perhatikan yaitu :

```
 @if ($errors->any())
    <div class="mb-5" role="alert">
        <div class="bg-red-500 text-white font-bold rounded-t px-4 py-2">
            There's something wrong!
        </div>
        <div class="border border-t-0 border-red-400 rounded-b bg-red-100
px-4 py-3             text-red-700">
            <p>
                <ul>
                    @foreach ($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </ul>
            </p>
        </div>
    </div>
@endif
```

   simplenya :

```
 @if ($errors->any())
    @foreach ($errors->all() as $error)
        <li>{{ $error }}</li>
    @endforeach
 @endif
```

   yang penting penting dari create.blade.php itu ini (**VALUE TYPE DAN NAME**) :

```
 <form action="{{ route('users.store') }}" method="post"
enctype="multipart/form-data">
@csrf
</form>
```

```
<input value="{{ old('name') }}" name="name" type="text" >
<input value="{{ old('email') }}" name="email" type="email" >
<input name="picturePath"type="file" placeholder="User Image">
```

```
<input value="{{ old('password') }}" name="password" type="password">
<input value="{{ old('password_confirmation') }}"
name="password_confirmation" type="password">
<input value="{{ old('address') }}" name="address" type="text">
<select name="roles">
    <option value="USER">User</option>
    <option value="ADMIN">Admin</option>
</select>
<input value="{{ old('houseNumber') }}" name="houseNumber" type="text">
<input value="{{ old('phoneNumber') }}" name="phoneNumber" type="text" >
<input value="{{ old('city') }}" name="city" type="text">
<button type="submit">
    Save User
</button>
```

LENGKAPNYA :

```
https://github.com/belajarkoding/foodmarket-
backend/blob/master/resources/views/users/create.blade.php
```

BE 4.3.4 CRUD USER DELETE

```
public function destroy(User $user)
{
    $user->delete();
    return redirect()->route('users.index');
}
```

BE 4.3.4 CRUD USER EDIT / UPDATE

```
public function edit(User $user)
{
    return view('users.edit', ['item'=>$user]);
}
```

```
public function update(UserRequest $request, User $user)
{
    $data = $request ->all();
    if($request->file('photo_profile_path'))
    {
        $data['photo_profile_path'] = $request->file('photo_profile_path');
        $data['photo_profile_path'] -> store('assets/user', 'public');
    }
    $user->update($data);
    return redirect() ->route('users.index');
}
```

BE 4.4.1 FOOD CONTROLLER - INDEX

1. buat controllernya dengan resource, dan setting functionnya
2. routing controller yang sudah dibuat
3. Buat halaman view nya .

Langkah-langkah :

1. ```
   MyApps> php artisan make:controller FoodController
   ```

   ```
   public function index()
   {
       $food = Food::paginate(10;
       return view(food.index, ['food'=>$food]);
   }
   ```

2. ```
   Route::resource('food', [FoodController::class]);
   ```

3. **resource\View\Create New Folder (foods)** : **index, edit, create**

   -> index.blade.php

   ```
   Buat table dan list food nya,
   https://github.com/belajarkoding/foodmarket-
   backend/blob/master/resources/views/food/index.blade.php
   ```

   yang perlu diinget dalam pembuatan index, **forelse** jangan lupa, abis itu link selalu menuju root.

BE 4.4.1 FOOD CONTROLLER - CREATE

1. Buat Request nya untuk validasi named **FoodRequest**, setting menjadi true dan isikan requirementnya
2. Setting Function create dan store
3. Buat halaman Create nya dan edit

1. ```
   MyApps > php artisan make:request FoodRequest
   ```

   app\Http\Requests\FoodRequest.php
   ```

```
public function authorize()
{
   return true;
}

public function rules()
{
    return [
       'name' => 'required|max:255',
       'picturePath' = > 'required|image'
       'description' => 'required'
       'ingredients' => 'required'
       'price' => 'required|integer'
       'rate' => 'required|numeric'
    ];
}
```

2. ```
   public function create()
   {
       return view('food.create');
   }

   public function store(Food $food)
   {
       $data = $request->all();

       if($request->file('picturePath'))
       {
           $data[picturePath] = $request->file('picturePath');
           $data[picturePath] -> store('assets\food', 'public');
       }
       Food::create($data);
       return redirect() -> route('food.index');
   }
   ```

3. edit.blade.php

   ```
   https://github.com/belajarkoding/foodmarket-
   backend/blob/master/resources/views/food/edit.blade.php
   ```

TRANSACTION CONTROLLER - INDEX

1. Buat Controller namanya TransactionController
2. Edit function indexnya
3. Buat folder **app\resource\View\Create New Folder (transactions)** : **index,detail** dan edit filenya .

1. ```
   MyApps> php artisan make:controller TransactionController--resorce
   ```

2.
```
public function index()
{
    $transaction = Transaction::with(['food', 'user'])->paginate(10);
    return view('transaction.index', ['transaction'=>$transaction]);
}
```

3. **index.blade.php**

```
https://github.com/belajarkoding/foodmarket-
backend/blob/master/resources/views/transactions/index.blade.php
```

Notes untuk index :

1. Menampilkan username dan food seperti ini :

```
@forelse($transaction as $item)
    {{ $item->food->name }} //INI TOLONG DIPAHAMI
    {{ $item->user->name }} //INI TOLONG DIPAHAMI

    //button
    <a href="{{ route('transactions.show', $item->id) }}">
        View
    </a>
    <form action="{{ route('transactions.destroy', $item->id) }}"
method="POST">
        {!! method_field('delete') . csrf_field() !!}
        <button type="submit" >
        Delete
        </button>
    </form>
@empty
    <tr>
        <td colspan="6" class="border text-center p-5">
            Data Tidak Ditemukan
        </td>
    </tr>
@endforelse
```

2. PAGINATE

```
<div class="text-center mt-5">
    {{ $transaction->links() }}
</div>
```

TRANSACTION CONTROLLER - SHOW DAN CHANGESTATUS

Alur **SHOW TRANSACTION**:

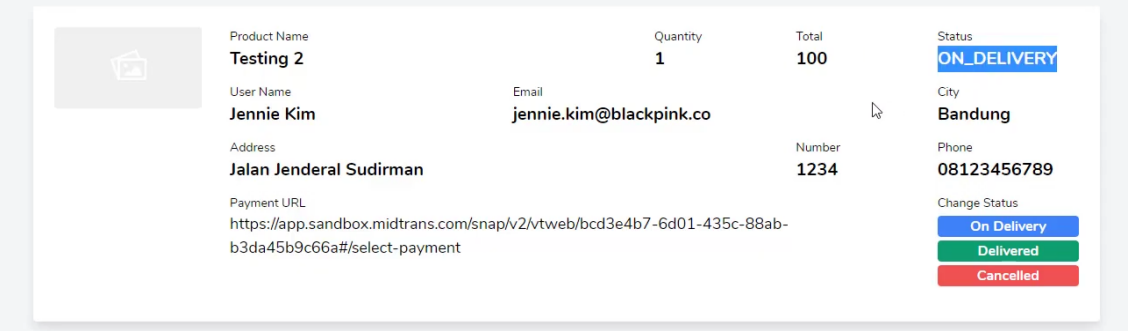
untuk show : kita hanya perlu **membuat file show.blade.php** (pada case ini namanya detail.blade.php - sama aja), abis itu **ubah function show nya**.

1. Setting parameternya menjadi **(Transaction $transaction)** shg kita dapat transaction detailnya

```
public function show(Transaction $transaction)
{
    return view('transactions.detail', ['item'=>$transaction]);
}
```

2. edit **detail.blade.php** nya .

```
https://github.com/belajarkoding/foodmarket-
backend/blob/master/resources/views/transactions/detail.blade.php
```



di atas ini adalah hasil dari **detail.blade.php**

3. kodingan gambar di atas untuk change status seperti ini :

```
<div class="w-1/6">
    <div class="text-sm mb-1">Change Status</div>

    <a href="{{ route('transactions.changeStatus', ['id' => $item->id,
'status' =>      'ON_DELIVERY']) }}">On Delivery
    </a>

    <a href="{{ route('transactions.changeStatus', ['id' => $item->id,
'status' =>      'DELIVERED']) }}"> Delivered
    </a>

    <a href="{{ route('transactions.changeStatus', ['id' => $item->id,
'status' => 'CANCELLED']) }}">
    Cancelled
    </a>

</div>
```

4. Kita harus membuat route dengan function changeStatus tersebut, gimana caranya ? dibawah ini ya

CARA MEMBUAT ROUTE CHANGESTATUS

1. buat function changeStatus dengan parameter request dan dari tag html di atas (id dan status),

```
public function changeStatus(Request $request, $id, $status)
{
    $transaction = Transaction::findOrFail($id);
    $transaction->status = $status; //status transaksi di database = status
dari html

    $transaction->save();
    return redirect()->route('transactions.show', $id);

}
```

2. DAFTARKAN ROUTE nya di web.php

```
Route::get('transaction/{id}/status/{status}',
[TransactionController::class, 'changeStatus'])-
>name('transaction.changeStatus');
```