

Nino Miguel Naje

BS IT – 3C |

Web Dev

This is a Controller that loads and populates the dashboard of the user side.

The user is asked to input on how many months does he/ she want to save

```
blade.php M  dashboard.blade.php  populatingpagesController X  admin-auth.php M
http > Controllers >  populatingpagesController > ...
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

0 references [0 implementations]
class populatingpagesController extends Controller
{
    0 references [0 overrides]
    public function index(Request $request): View
    {
        $startMonth = $request->input(key: 'start_month');
        $endMonth = $request->input(key: 'end_month');
        $posts = [];

        $monthNames = [
            1 => 'January', 2 => 'February', 3 => 'March', 4 => 'April',
            5 => 'May', 6 => 'June', 7 => 'July', 8 => 'August',
            9 => 'September', 10 => 'October', 11 => 'November', 12 => 'December'
        ];

        if ($startMonth && $endMonth) {


            if (!is_numeric(value: $startMonth) || $startMonth < 1 || $startMonth > 12) {
                $startMonth = 1;
            }

            if (!is_numeric(value: $endMonth) || $endMonth < 1 || $endMonth > 12) {
                $endMonth = 12;
            }

            if ($startMonth > $endMonth) {
                list($startMonth, $endMonth) = [$endMonth, $startMonth];
            }

            for ($month = $startMonth; $month <= $endMonth; $month++) {
                $posts[] = [
                    'Username' => $monthNames[$month],
                    'month' => $month,
                    'content' => "This is the month: " . $monthNames[$month],
                ];
            }
        }

        return view(view: 'dashboard', data: compact(var_name: 'posts', var_names: 'startMonth', 'endMonth'));
    }
}
```


Dashboard
chad

Dashboard

MONTH TRACKER

Start Month:
3
End Month:
5
Submit

March
Months: March
This is the month: March
Continue

April
Months: April
This is the month: April
Continue

May
Months: May
This is the month: May
Continue

This is a controller that is responsible for the users profile update. It lets the user edits his/her profile and is registered in the routes

```

4 references | 0 implementations
class ProfileController extends Controller
{
  /**
   * Display the user's profile form.
   */
  1 reference | 0 overrides
  public function edit(Request $request): View
  {
    return view('profile.edit', data: [
      'user' => $request->user(),
    ]);
  }

  /**
   * Update the user's profile information.
   */
  1 reference | 0 overrides
  public function update(ProfileUpdateRequest $request): RedirectResponse
  {
    $request->user()->fill($request->validated());

    if ($request->user()->isDirty('email')) {
      $request->user()->email_verified_at = null;
    }

    $request->user()->save();

    return Redirect::route('profile.edit')->with(key: 'status', value: 'profile-updated');
  }

  /**
   * Delete the user's account.
   */
  1 reference | 0 overrides
  public function destroy(Request $request): RedirectResponse
  {
    $request->validateWithBag('userDeletion', rules: [
      'password' => ['required', 'current_password'],
    ]);

    $user = $request->user();

    Auth::logout();

    $user->delete();

    $request->session()->invalidate();
    $request->session()->regenerateToken();

    return Redirect::to(path: '/');
  }
}

```

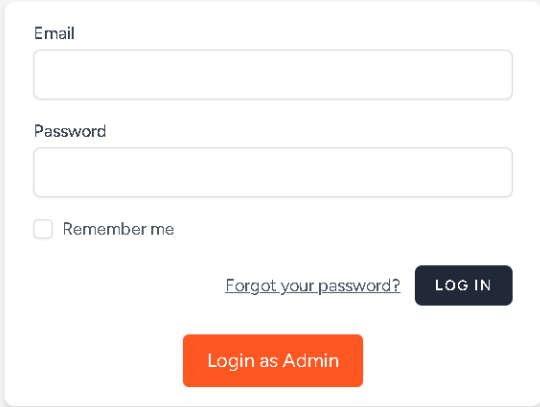
This is the profile controller that is registered in the routes.

```
login.blade.php M dashboard.blade.php web.php X admin-auth.php M
routes > web.php
1 <?php
2
3 use App\Http\Controllers\ProfileController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get(uri: '/', action: function (): View {
7     return view(view: 'welcome');
8 });
9
10 Route::get(uri: '/dashboard', action: function (): View {
11     return view(view: 'dashboard');
12 })->middleware(middleware: ['auth', 'verified'])->name(name: 'dashboard');
13
14 Route::middleware(middleware: 'auth')->group(callback: function (): void {
15     Route::get(uri: '/profile', action: [ProfileController::class, 'edit'])->name(name: 'profile.edit');
16     Route::patch(uri: '/profile', action: [ProfileController::class, 'update'])->name(name: 'profile.update');
17     Route::delete(uri: '/profile', action: [ProfileController::class, 'destroy'])->name(name: 'profile.destroy');
18 });
19
20 require __DIR__.'/auth.php';
21 require __DIR__.'/admin-auth.php';
```

This controller is the one that creates a user account and is in the login page

```
login.blade.php M dashboard.blade.php RegisteredUserController.php X admin-auth.php M
app > Http > Controllers > Auth > RegisteredUserController.php > RegisteredUserController > store()
1 namespace App\Http\Controllers\Auth;
2
3
4
5 use App\Http\Controllers\Controller;
6 use App\Models\User;
7 use Illuminate\Auth\Events\Registered;
8 use Illuminate\Http\RedirectResponse;
9 use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\Support\Facades\Hash;
12 use Illuminate\Validation\Rules;
13 use Illuminate\View\View;
14
15 3 references | 0 implementations
16 class RegisteredUserController extends Controller
17 {
18     /**
19      * Display the registration view.
20      */
21     1 reference | 0 overrides
22     public function create(): View
23     {
24         return view(view: 'auth.register');
25     }
26
27     /**
28      * Handle an incoming registration request.
29      *
30      * @throws \Illuminate\Validation\ValidationException
31      */
32     1 reference | 0 overrides
33     public function store(Request $request): RedirectResponse
34     {
35         $request->validate(rules: [
36             'name' => ['required', 'string', 'max:255'],
37             'email' => ['required', 'string', 'lowercase', 'email', 'max:255', 'unique:'.User::class],
38             'password' => ['required', 'confirmed', Rules\Password::defaults()],
39         ]);
40
41         $user = User::create(attributes: [
42             'name' => $request->name,
43             'email' => $request->email,
44             'password' => Hash::make(value: $request->password),
45         ]);
46
47         event(args: new Registered(user: $user));
48
49         Auth::login(user: $user);
50
51         return redirect(to: route(name: 'dashboard', absolute: false));
52     }
53 }
```

Here it is seen in the log in page.



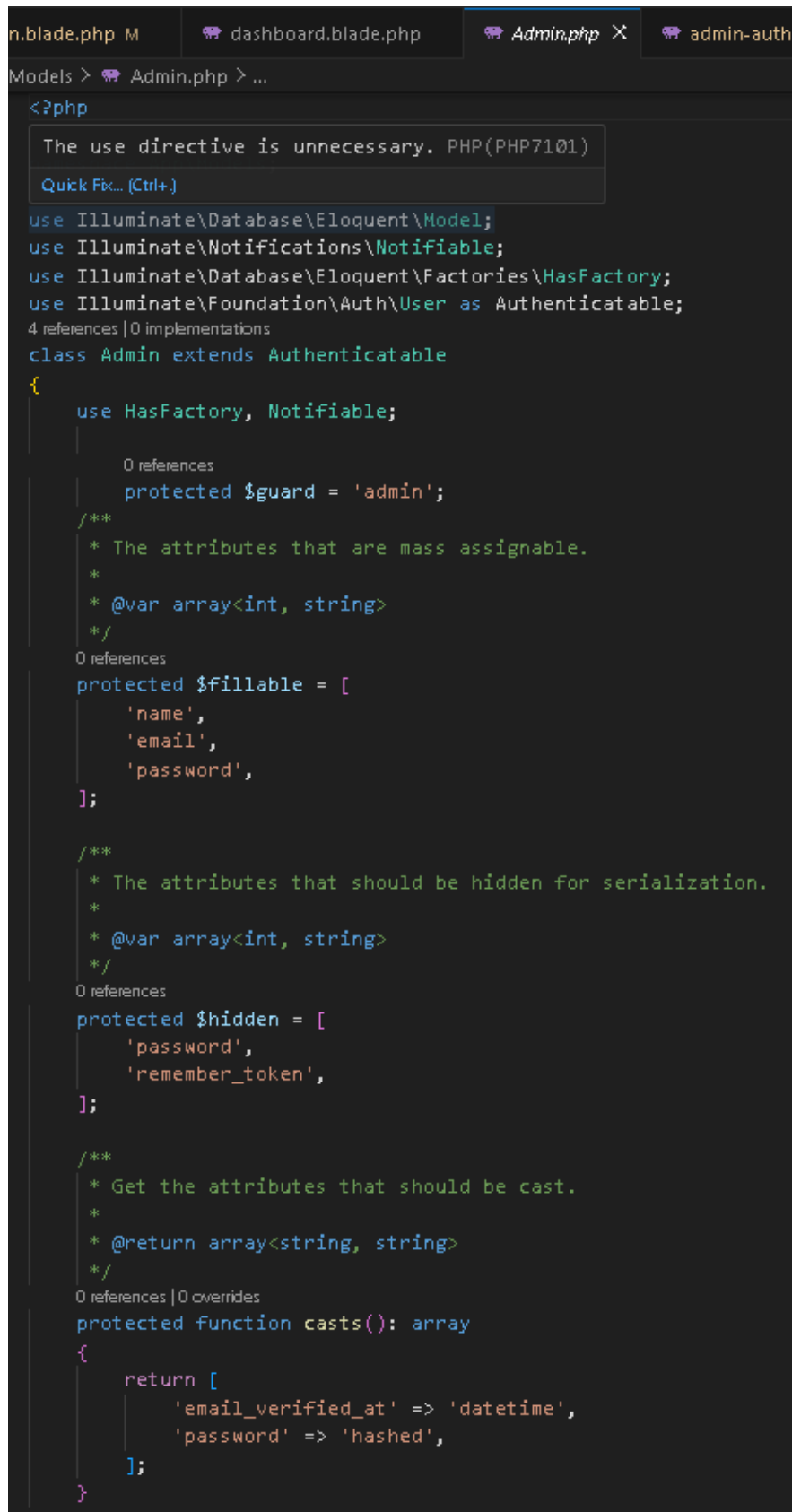
The image shows a login form on a light gray background. At the top center is a logo consisting of three blue cubes arranged in a 2x2 grid with the bottom-right cube missing. Below the logo is a white rounded rectangle containing the form fields. The form has two input fields: 'Email' and 'Password'. Below the 'Password' field is a checkbox labeled 'Remember me'. To the right of the checkbox is a link that says 'Forgot your password?'. To the right of the link is a dark blue button with the text 'LOG IN' in white. Below the 'LOG IN' button is an orange button with the text 'Login as Admin' in white.

Here is the seeding/ creation of dummy user that can be used in the login page.

```
login.blade.php M | dashboard.blade.php | DatabaseSeeder.php X | admin-aut
Database > seeders > DatabaseSeeder.php > ...

1  <?php
2
3  namespace Database\Seeders;
4
5  use App\Models\User;
6  // use Illuminate\Database\Console\Seeds\WithoutModelEvents;
7  use Illuminate\Database\Seeder;
8
9  0 references | 0 implementations
10 class DatabaseSeeder extends Seeder
11 {
12     /**
13      * Seed the application's database.
14      */
15     0 references | 0 overrides
16     public function run(): void
17     {
18         // User::factory(10)->create();
19
20         User::factory()->create(attributes: [
21             'name' => 'Test User',
22             'email' => 'test@example.com',
23         ]);
24     }
25 }
```

An admin model that has the function to delete and update a user.



```
<?php
The use directive is unnecessary. PHP(PHP7101)
Quick Fix... (Ctrl+)

use Illuminate\Database\Eloquent\Model;
use Illuminate\Notifications\Notifiable;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
4 references | 0 implementations
class Admin extends Authenticatable
{
    use HasFactory, Notifiable;

    0 references
    protected $guard = 'admin';

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    0 references
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    0 references
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    0 references | 0 overrides
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
```