

Membuat fitur login

buat table dengan nama **pengguna** seperti ini

The screenshot shows the phpMyAdmin interface on a Windows desktop. The left sidebar lists databases: 'latihan_smk_ds', 'latihan_smk_ds_compro', 'latihan_database', 'mysql', 'performance_schema', 'pertemuan1', 'run_code', 'sistemtiket', 'sys', 'tjm-landingpage', 'tocly_laravel', 'uedemy-nodejs', 'wastewise', and 'wordpress'. The 'latihan_smk_ds_compro' database is selected. In the center, the 'pengguna' table structure is displayed with the following columns:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	int			No	None		AUTO_INCREMENT	Change Drop More
2	<code>name</code>	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
3	<code>email</code>	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
4	<code>password</code>	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	<code>created_at</code>	timestamp			Yes	NULL			Change Drop More
6	<code>updated_at</code>	timestamp			Yes	NULL			Change Drop More

At the bottom, there is a 'Indexes' section and a 'Console' tab.

setelah buat table **pengguna**, buat model untuk pengguna dengan menjalankan
php artisan make:model Pengguna

setelah table Pengguna dibuat ubah model pengguna jadi seperti ini

The screenshot shows the VS Code interface with the Laravel application structure on the left. The current file is `Pengguna.php` located in the `app/Models` directory. The code defines a `Pengguna` model that extends `Authenticatable` and `HasApiTokens`, and uses the `pengguna` table.

```
1  You, 1 second ago | 1 author (You)
2
3  namespace App\Models;
4
5  // use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Illuminate\Notifications\Notifiable;
9  use Laravel\Sanctum\HasApiTokens;
10
11 class Pengguna extends Authenticatable
12 {
13     use HasApiTokens, HasFactory, Notifiable;
14
15     use HasFactory;
16     protected $table='pengguna';
17 }
18
```

A tooltip at the bottom right indicates a "Laravel Extra Intellisense error".

lalu buka auth.php di folder config,ubah key model jadi seperti ini,arahkan ke model pengguna

The screenshot shows the `auth.php` file in the `config` directory. The `'model'` key in the `'users'` provider is set to `App\Models\Pengguna::class`.

```
50 | All authentication drivers have a user provider. This defines how the
51 | users are actually retrieved out of your database or other storage
52 |
53 |
54 | If you have multiple user tables or models you may configure multiple
55 | sources which represent each model / table. These sources may then
56 | be assigned to any extra authentication guards you have defined.
57 |
58 | Supported: "database", "eloquent"
59 |
60 */
61
62 'providers' => [
63     'users' => [
64         'driver' => 'eloquent',
65         'model' => App\Models\Pengguna::class,
66     ],
67
68     // 'users' => [
69     //     'driver' => 'database',
70     //     'table' => 'users',
71     // ],
72
73 /**
74 | -----
75 | Resetting Passwords
76 | -----
77 |
78 | You may specify multiple password reset configurations if
79 | there is more than one user table or model in the application and you
80 | want to use different settings for each. Just add another array item here
81 |
82 |
83 | The expire time is the number of minutes that each reset token will be
84 | available.
85 |
86 | -----
87 | -----
```

A tooltip at the bottom right indicates a "Laravel Extra Intellisense error".

lalu buat fungsi untuk login nya di **LoginController**,buat fungsi dengan nama aksi_login seperti ini

The screenshot shows the VS Code interface with the Laravel application structure on the left. The current file is `LoginController.php`, which contains the following code:

```
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Auth;
7
8 You, 1 second ago | 1 author (You)
9 class LoginController extends Controller
10 {
11     public function index(){
12         return view('backend.login.index');
13     }
14     public function aksi_login(Request $request)
15     {
16         $request->validate([
17             'email' => 'required|email',
18             'password' => 'required'
19         ], [
20             'email.email' => 'Email tidak valid!',
21             'password.required' => 'Password harus di isi!',
22             'email.required' => 'Email harus di isi!',
23         ]);
24     }
25     $credentials = $request->only(['email', 'password']);
26     // mengecek email dan password
27     if (Auth::attempt($credentials)) {
28         $request->session()->regenerate();
29         return redirect()->route('backend.blog');
30     }
31     return redirect()->back();
```

The status bar at the bottom indicates "You, 1 second ago * Uncommitted changes".

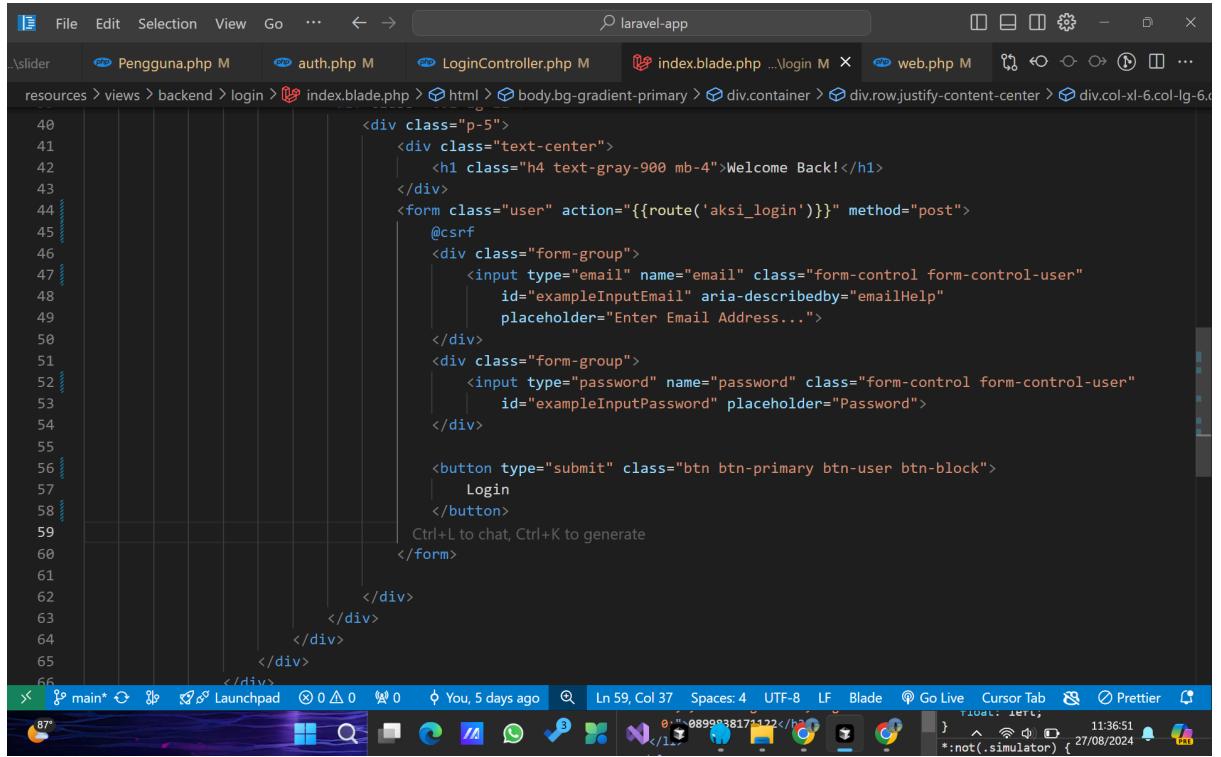
buat route dahulu dengan nama aksi_login

The screenshot shows the VS Code interface with the routes structure on the left. The current file is `routes/web.php`, which contains the following code:

```
19 | be assigned to the "web" middleware group. Make something great!
20 |
21 */
22
23 Route::get('/', [HomeController::class, 'index'])->name('home');
24
25 Route::get('/blog', [BlogController::class, 'index'])->name('blog');
26 Route::get('/blog/detail/{slug}', [BlogController::class, 'detail'])
27 ->name('blog_detail');
28
29
30 // backend
31 Route::get('/login', [LoginController::class, 'index'])->name('login');
32 Route::post('/aksi_login', [LoginController::class, 'aksi_login'])->name('aksi_login');
33
34
35 Route::get('backend/blog', [BackendBlogController::class, 'index'])->name('backend.blog');
36 Route::get('backend/slider', [SliderController::class, 'index'])->name('backend.slider');
37 Route::get('backend/service', [ServiceController::class, 'index'])->name('backend.service');
```

The status bar at the bottom indicates "You, now" and "Ln 33, Col 83".

setelah buat fungsi login,buka folder **backend>login>index.blade.php**
lalu tambahkan action yang mengarahkan ke aksi_login



A screenshot of a code editor window titled "laravel-app". The file being edited is "index.blade.php" located at "resources > views > backend > login". The code is a login form:

```
40 <div class="p-5">
41     <div class="text-center">
42         <h1 class="h4 text-gray-900 mb-4">Welcome Back!</h1>
43     </div>
44     <form class="user" action="{{route('aksi_login')}}" method="post">
45         @csrf
46         <div class="form-group">
47             <input type="email" name="email" class="form-control form-control-user"
48                 id="exampleInputEmail" aria-describedby="emailHelp"
49                 placeholder="Enter Email Address...">
50         </div>
51         <div class="form-group">
52             <input type="password" name="password" class="form-control form-control-user"
53                 id="exampleInputPassword" placeholder="Password">
54         </div>
55
56         <button type="submit" class="btn btn-primary btn-user btn-block">
57             Login
58         </button>
59     </form>
60
61     </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
```

The status bar at the bottom shows "Ln 59, Col 37" and "Spaces: 4".

Membuat middleware

buat middleware dengan menjalankan

php artisan make:middleware AuthLogin

jika sudah di buat buka file nya di folder middleware AuthLogin lalu ubah jadi seperti ini, untuk mengecek apakah sudah login atau belum.

The screenshot shows the VS Code interface with the Laravel application open. The left sidebar displays the project structure under 'LARAVEL-APP'. The main editor window shows the contents of the 'AuthLogin.php' file, which is part of the 'Http\Middleware' directory. The code handles incoming requests and checks if authentication is required.

```

app > Http > Middleware > AuthLogin.php > handle
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Auth;
8  use Symfony\Component\HttpFoundation\Response;
9
10 class AuthLogin
11 {
12     /**
13      * Handle an incoming request.
14      *
15      * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
16     */
17    public function handle(Request $request, Closure $next): Response
18    {
19        if(Auth::check()){
20            return $next($request);
21        }
22        return redirect()->route('login');
23    }
24 }
25

```

setelah itu daftar kan middleware yang telah di buat di file **kernel.php** di folder **Http** dengan nama **AuthWeb**

The screenshot shows the VS Code interface with the Laravel application open. The left sidebar displays the project structure under 'LARAVEL-APP'. The main editor window shows the contents of the 'Kernel.php' file, which is part of the 'Http' directory. The 'protected \$middlewareAliases' array includes the 'AuthWeb' alias, which points to the 'AuthLogin' middleware.

```

app > Http > Kernel.php > ...
41     'api' => [
42         // \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
43         \Illuminate\Routing\Middleware\ThrottleRequests::class.'api',
44         \Illuminate\Routing\Middleware\SubstituteBindings::class,
45     ],
46 },
47
48 /**
49  * The application's middleware aliases.
50  *
51  * Aliases may be used to conveniently assign middleware to routes and groups.
52  *
53  * @var array<string, class-string|string>
54 */
55 protected $middlewareAliases = [
56     'auth' => \App\Http\Middleware\Authenticate::class,
57     'AuthWeb'=>\App\Http\Middleware\AuthLogin::class,
58     'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
59     'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
60     'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
61     'can' => \Illuminate\Auth\Middleware\Authorize::class,
62     'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
63     'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
64     'signed' => \App\Http\Middleware\ValidateSignature::class,
65     'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
66     'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
67 ];
68 }
69

```

A tooltip at the bottom of the editor window says 'Ctrl+L to chat, Ctrl+K to generate'.

A notification bar at the bottom right shows a 'Laravel Extra Intellisense error' with a 'View Error' button and a 'Don't show again' checkbox.

jika sudah di daftarkan pasang middleware di setiap route yang memerlukan authentication/login

The screenshot shows a code editor interface with a dark theme. The left sidebar displays a file tree for a Laravel application named 'LARAVEL-APP'. The 'routes' folder contains 'api.php', 'channels.php', 'console.php', and 'web.php'. The 'web.php' file is currently selected and open in the main editor area. The code in 'web.php' defines several routes:

```
routes > web.php > ...
17 | Here is where you can register web routes for your application. These
18 | routes are loaded by the RouteServiceProvider and all of them will
19 | be assigned to the "web" middleware group. Make something great!
20 |
21 */
22
23 Route::get('/', [HomeController::class, 'index'])->name('home');
24
25 Route::get('/blog', [BlogController::class, 'index'])->name('blog');
Route::get('/blog/detail/{slug}', [BlogController::class, 'detail']);
->name('blog_detail');
26
27
28
29
30
31 // backend
32 Route::get('/login', [LoginController::class, 'index'])->name('login');
Route::post('/aksi_login', [LoginController::class, 'aksi_login'])->name('aksi_login');
33
34
35 Route::middleware(['AuthWeb'])->group(function () {
    Route::get('backend/blog', [BackendBlogController::class, 'index'])->name('backend.blog');
    Route::get('backend/slider', [SliderController::class, 'index'])->name('backend.slider');
    Route::get('backend/service', [ServiceController::class, 'index'])->name('backend.service');
});
36
37
38
39 });
40 | Ctrl+L to chat, Ctrl+K to generate
```

The status bar at the bottom shows various system icons and the date/time: 28/08/2024, 08:27:14.