

# **PEMROGRAMAN MOBILE**

## **“Flutter Laravel API”**

Dosen Pengampu: Ade Ismail, S.Kom., MTI.



Oleh:

**YUNIKA PUTERI DWI ANTIKA**

**2241760048 / SIB-3E**

**PROGRAM STUDI SISTEM INFORMASI BISNIS**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2024**

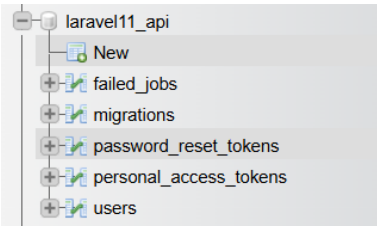
## Membuat API dengan Laravel Breeze di Laravel 11

Langkah	Jawaban/Deskripsi
<b>Langkah 1: Instal Laravel</b>	
1	<p>Pertama, buat proyek Laravel baru menggunakan penginstal Laravel atau Composer.</p> <pre>C:\laragon\www\FlutterAPI_Mobile&gt;composer create-project laravel/laravel api-breeze Creating a "laravel/laravel" project at "./api-breeze" Cannot use laravel/laravel's latest version v11.3.3 as it requires php ^8.2 which is not satisfied by your platform. Installing laravel/laravel (v10.3.3) - Installing laravel/laravel (v10.3.3): Extracting archive Created project in C:\laragon\www\FlutterAPI_Mobile\api-breeze &gt; @php -r "file_exists('.env')    copy('.env.example', '.env');" Loading composer repositories with package information Updating dependencies Lock file operations: 111 installs, 0 updates, 0 removals - Locking brick/math (0.12.1) - Locking carbonphp/carbon-doctrine-types (2.1.0) - Locking dflydev/dot-access-data (v3.0.3) - Locking doctrine/inflector (2.0.10) - Locking doctrine/lexer (3.0.1) - Locking dragonmantank/cron-expression (v3.4.0) - Locking egulias/email-validator (4.0.2) - Locking fakerphp/faker (v1.24.1) - Locking filp/whoops (2.16.0) - Locking fruitcake/php-cors (v1.3.0) - Locking graham-campbell/result-type (v1.1.3) - Locking guzzlehttp/guzzle (7.9.2) - Locking guzzlehttp/promises (2.0.4) - Locking guzzlehttp/psr7 (2.7.0) - Locking guzzlehttp/uri-template (v1.0.3) - Locking hamcrest/hamcrest-php (v2.0.1) - Locking laravel/framework (v10.48.24) - Locking laravel/pint (v1.18.2) - Locking laravel/prompts (v0.1.25) - Locking laravel/sail (v1.38.0) - Locking laravel/sanctum (v3.3.3) - Locking laravel/serializable-closure (v1.3.7) - Locking laravel/tinker (v2.10.0) - Locking league/commonmark (2.5.3)  - Installing spatie/flame-client-php (1.8.0): Extracting archive - Installing spatie/ignition (1.15.0): Extracting archive - Installing spatie/laravel-ignition (2.8.0): Extracting archive 56 package suggestions were added by new dependencies, use 'composer suggest' to see details. Generating optimized autoload files &gt; illuminate\Foundation\ComposerScripts::postAutoloadDump &gt; @php artisan package:discover --ansi  [INFO] Discovering packages.  laravel/sail ..... DONE laravel/sanctum ..... DONE laravel/tinker ..... DONE nesbot/carbon ..... DONE nunomaduro/collision ..... DONE nunomaduro/termwind ..... DONE spatie/laravel-ignition ..... DONE  82 packages you are using are looking for funding. Use the 'composer fund' command to find out more! &gt; @php artisan vendor:publish --tag=laravel-assets --ansi --force  [INFO] No publishable resources for tag [laravel-assets].  No security vulnerability advisories found. &gt; @php artisan key:generate --ansi  [INFO] Application key set successfully.  C:\laragon\www\FlutterAPI_Mobile&gt;cd api-breeze C:\laragon\www\FlutterAPI_Mobile\api-breeze&gt;</pre>
<b>Langkah 2: Instal Laravel Breeze</b>	
1	<p>Selanjutnya, instal Laravel Breeze dan dependensinya.</p>

	<pre> C:\laragon\www\FlutterAPI_Mobile\api-breeze&gt;composer require laravel/breeze --dev Cannot use laravel/breeze's latest version v2.2.5 as it requires php ^8.2.0 which is not satisfied by your platform. ./composer.json has been updated Running composer update laravel/breeze Loading composer repositories with package information Updating dependencies Lock file operations: 1 install, 0 updates, 0 removals - locking laravel/breeze (v1.29.1) Writing lock file Installing dependencies from lock file (including require-dev) Package operations: 1 install, 0 updates, 0 removals - downloading laravel/breeze (v1.29.1) - installing laravel/breeze (v1.29.1): Extracting archive Generating optimized autoload files &gt; illuminate\Foundation\ComposerScripts::postAutoloadDump &gt; @php artisan package:discover --ansi   INFO  Discovering packages.  laravel/breeze ..... DONE laravel/sail ..... DONE laravel/sanctum ..... DONE laravel/tinker ..... DONE nesbot/carbon ..... DONE nunomaduro/collision ..... DONE nunomaduro/termwind ..... DONE spatie/laravel-ignition ..... DONE  82 packages you are using are looking for funding. Use the 'composer fund' command to find out more! &gt; @php artisan vendor:publish --tag=laravel-assets --ansi --force   INFO  No publishable resources for tag [laravel-assets].  No security vulnerability advisories found. Using version ^1.29 for laravel/breeze </pre> <pre> C:\laragon\www\FlutterAPI_Mobile\api-breeze&gt;php artisan breeze:install api   INFO  Breeze scaffolding installed successfully.  C:\laragon\www\FlutterAPI_Mobile\api-breeze&gt; </pre> <p>Perintah ini akan menginstal Breeze dan menyiapkan perancah yang diperlukan untuk autentikasi API.</p>
--	---

### Langkah 3: Konfigurasi Database dan Jalankan Migrasi

1	<p>Perbarui file .env Anda dengan kredensial basis data Anda:</p> <pre> 12 DB_CONNECTION=mysql 13 DB_HOST=127.0.0.1 14 DB_PORT=3306 15 DB_DATABASE=laravel11_api 16 DB_USERNAME=root 17 DB_PASSWORD= </pre>
2	<p>Jalankan migrasi untuk menyiapkan tabel database Anda:</p> <pre> C:\laragon\www\FlutterAPI_Mobile\api-breeze&gt;php artisan migrate   WARN  The database 'laravel11_api' does not exist on the 'mysql' connection.  Would you like to create it? (yes/no) [no] &gt; yes   INFO  Preparing database.  Creating migration table ..... 28ms DONE   INFO  Running migrations.  2014_10_12_000000_create_users_table ..... 46ms DONE 2014_10_12_100000_create_password_reset_tokens_table ..... 24ms DONE 2019_08_19_000000_create_failed_jobs_table ..... 34ms DONE 2019_12_14_000001_create_personal_access_tokens_table ..... 46ms DONE  C:\laragon\www\FlutterAPI_Mobile\api-breeze&gt; </pre>

	
--	--

#### Langkah 4: Buat Titik Akhir Autentikasi

1	<p>Laravel Breeze menyediakan titik akhir yang diperlukan untuk registrasi, login, dan logout. Rute-rute tersebut didefinisikan dalam routes/api.php.</p> <pre> api-breeze &gt; routes &gt; api.php &gt; ... 1  &lt;?php 2 3  use Illuminate\Http\Request; 4  use Illuminate\Support\Facades\Route; 5  use App\Http\Controllers\Auth\AuthenticatedSessionController; 6  use App\Http\Controllers\Auth\RegisteredUserController; 7 8  /* 9   ----- 10   API Routes 11  ----- 12   13   Here is where you can register API routes for your application. These 14   routes are loaded by the RouteServiceProvider within a group which 15   is assigned the "api" middleware group. Enjoy building your API! 16   17  */ 18 19 Route::middleware(['auth:sanctum'])-&gt;get('/user', function (Request \$request) { 20     return \$request-&gt;user(); 21 }); 22 23 Route::post('/register', [RegisteredUserController::class, 'store']); 24 Route::post('/login', [AuthenticatedSessionController::class, 'store']); 25 Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])-&gt;middleware('auth:sanctum'); 26 </pre>
---	--

#### Langkah 5: Perbarui Pengontrol

1	<p>Ubah RegisteredUserController dan AuthenticatedSessionController untuk mengembalikan respons JSON.</p> <ul style="list-style-type: none"> <li>RegisteredUserController.php</li> </ul>
---	--

```

api-breeze > app > Http > Controllers > Auth > RegisteredUserController.php > PHP Intelephense > RegisteredUserController
1  <?php
2
3  namespace App\Http\Controllers\Auth;
4
5  use App\Models\User;
6  use Illuminate\Auth\Events\Registered;
7  use Illuminate\Http\Request;
8  use Illuminate\Support\Facades\Hash;
9  use Illuminate\Validation\Rules;
10 use App\Http\Controllers\Controller;
11
12 class RegisteredUserController extends Controller
13 {
14     public function store(Request $request)
15     {
16         $request->validate([
17             'name' => ['required', 'string', 'max:255'],
18             'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
19             'password' => ['required', 'confirmed', Rules\Password::defaults()],
20         ]);
21
22         $user = User::create([
23             'name' => $request->name,
24             'email' => $request->email,
25             'password' => Hash::make($request->password),
26         ]);
27
28         event(new Registered($user));
29
30         $token = $user->createToken('auth_token')->plainTextToken;
31
32         return response()->json([
33             'access_token' => $token,
34             'token_type' => 'Bearer',
35             'user' => $user
36         ]);
37     }
38 }

```

- AuthenticatedSessionController.php

```

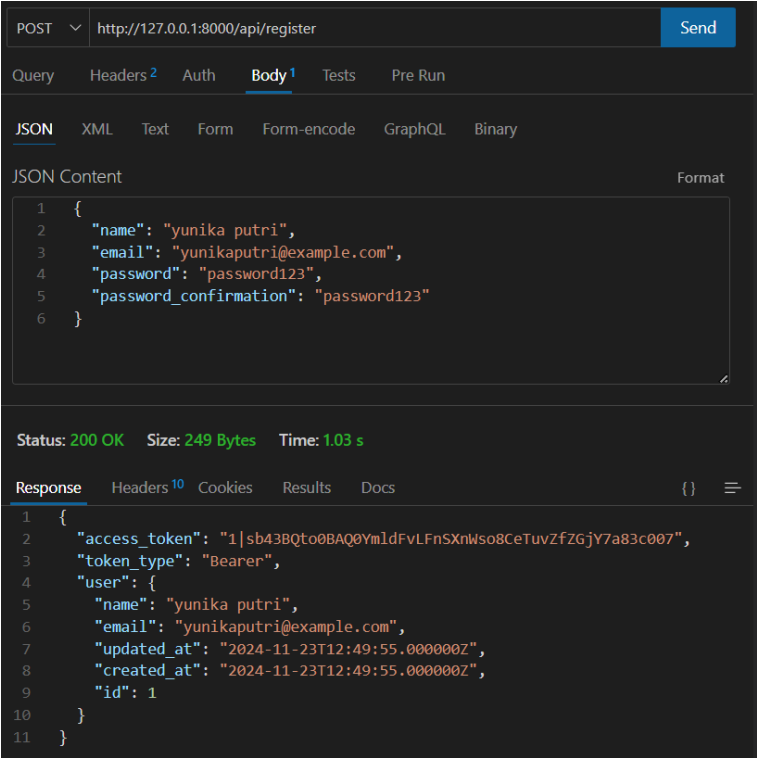
api-breeze > app > Http > Controllers > Auth > AuthenticatedSessionController.php > PHP Intelephense > AuthenticatedSessionController > store
1  <?php
2
3  namespace App\Http\Controllers\Auth;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\Auth;
7  use App\Http\Controllers\Controller;
8
9  class AuthenticatedSessionController extends Controller
10 {
11     public function store(Request $request)
12     {
13         $request->validate([
14             'email' => ['required', 'string', 'email'],
15             'password' => ['required', 'string'],
16         ]);
17
18         if (!Auth::attempt($request->only('email', 'password'))) {
19             return response()->json(['message' => 'Invalid login credentials'], 401);
20         }
21
22         $user = Auth::user();
23         $token = $user->createToken('auth_token')->plainTextToken;
24
25         return response()->json([
26             'access_token' => $token,
27             'token_type' => 'Bearer',
28             'user' => $user,
29             'status' => 'Login successful',
30         ]);
31     }
32
33     public function destroy(Request $request)
34     {
35         $request->user()->currentAccessToken()->delete();
36
37         return response()->json(['message' => 'Logout successful']);
38     }
39 }
40

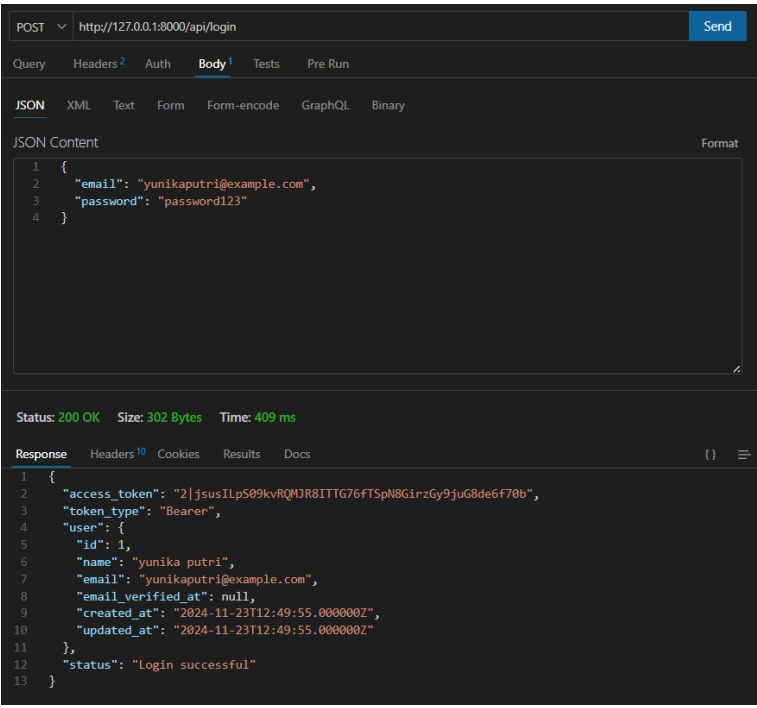
```

## Langkah 5: Jalankan Aplikasi Laravel

1	<pre>C:\laragon\www\Flutter_API\api-breeze&gt;php artisan serve  INFO Server running on [http://127.0.0.1:8000]. INFO Server running on [http://127.0.0.1:8000].  Press Ctrl+C to stop the server  Press Ctrl+C to stop the server Press Ctrl+C to stop the server  ..... ~ 1s █</pre>
---	--

**Langkah 6: Periksa API berikut**

1	<p>Uji API Anda dengan Thunder Client</p>  <pre>POST http://127.0.0.1:8000/api/register Send  Query Headers 2 Auth Body 1 Tests Pre Run  JSON XML Text Form Form-encode GraphQL Binary  JSON Content Format  1 { 2   "name": "yunika putri", 3   "email": "yunikaputri@example.com", 4   "password": "password123", 5   "password_confirmation": "password123" 6 }  Status: 200 OK Size: 249 Bytes Time: 1.03 s  Response Headers 10 Cookies Results Docs {} ≡  1 { 2   "access_token": "1 sb43BQto0BAQ0YmldFvLFnSXnkwso8CeTuvZfGjY7a83c007", 3   "token_type": "Bearer", 4   "user": { 5     "name": "yunika putri", 6     "email": "yunikaputri@example.com", 7     "updated_at": "2024-11-23T12:49:55.000000Z", 8     "created_at": "2024-11-23T12:49:55.000000Z", 9     "id": 1 10  } 11 }</pre>
---	---

	 <pre> POST http://127.0.0.1:8000/api/login Content-Type: application/json  {   "email": "yunikaputri@example.com",   "password": "password123" }  Status: 200 OK Size: 302 Bytes Time: 409 ms  {   "access_token": "2 jsusILp509kvRQMJR8ITTG76fTSpN8GirzGy9juG8de6f70b",   "token_type": "Bearer",   "user": {     "id": 1,     "name": "yunika putri",     "email": "yunikaputri@example.com",     "email_verified_at": null,     "created_at": "2024-11-23T12:49:55.000000Z",     "updated_at": "2024-11-23T12:49:55.000000Z"   },   "status": "Login successful" } </pre>
--	---

## Membuat Aplikasi Mobile Flutter

Langkah	Jawaban/Deskripsi
<b>Langkah 1: Persiapan Proyek Flutter</b>	
1	<p>Buat Proyek Flutter Baru:</p> <pre> C:\laragon\www\Flutter_API\api-breeze&gt;flutter create my_flutter_app Creating project my_flutter_app... Resolving dependencies in `my_flutter_app`... (1.3s) Downloading packages... Got dependencies in `my_flutter_app`. Wrote 129 files.  All done! You can find general documentation for Flutter at: https://docs.flutter.dev/ Detailed API documentation is available at: https://api.flutter.dev/ If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev  In order to run your application, type:  \$ cd my_flutter_app \$ flutter run  Your application code is in my_flutter_app\lib\main.dart.  C:\laragon\www\Flutter_API\api-breeze&gt;cd my_flutter_app C:\laragon\www\Flutter_API\api-breeze\my_flutter_app&gt; </pre>
2	<p><b>Tambahkan Dependencies:</b></p> <p>Buka pubspec.yaml dan tambahkan beberapa dependencies yang diperlukan:</p>

```

30 dependencies:
31   flutter:
32     sdk: flutter
33   http: ^0.13.3
34   shared_preferences: ^2.0.6
35   provider: ^6.0.0
36   flutter_secure_storage: ^5.0.2

```

Jalankan **flutter pub get** untuk mengunduh dependencies.

```

C:\laragon\www\Flutter_API\api-breeze\my_flutter_app>flutter pub get
Resolving dependencies... (1.4s)
Downloading packages... (1.2s)
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  characters 1.3.0 (1.3.1 available)
  clock 1.1.1 (1.1.2 available)
  collection 1.18.0 (1.19.1 available)
  fake_async 1.3.1 (1.3.2 available)
+ ffi 2.1.3
+ file 7.0.1
  flutter_lints 4.0.0 (5.0.0 available)
+ flutter_secure_storage 5.1.2 (9.2.2 available)
+ flutter_secure_storage_linux 1.2.1
+ flutter_secure_storage_macos 1.1.2 (3.1.2 available)
+ flutter_secure_storage_platform_interface 1.1.2
+ flutter_secure_storage_web 1.2.1
+ flutter_secure_storage_windows 1.1.3 (3.1.2 available)
+ flutter_web_plugins 0.0.0 from sdk flutter
+ http 0.13.6 (1.2.2 available)
+ http_parser 4.0.2 (4.1.1 available)
+ js 0.6.7 (0.7.1 available)
  leak_tracker 10.0.5 (10.0.8 available)
  leak_tracker_flutter_testing 3.0.5 (3.0.9 available)
  lints 4.0.0 (5.1.0 available)
  matcher 0.12.16+1 (0.12.17 available)
  material_color_utilities 0.11.1 (0.12.0 available)
  meta 1.15.0 (1.16.0 available)
+ nested 1.0.0
  path 1.9.0 (1.9.1 available)
+ path_provider_linux 2.2.1
+ path_provider_platform_interface 2.1.2
+ path_provider_windows 2.3.0
+ platform 3.1.6
+ plugin_platform_interface 2.1.8
+ provider 6.1.2
+ shared_preferences 2.3.3
+ shared_preferences_android 2.3.3
+ shared_preferences_foundation 2.5.3
+ shared_preferences_linux 2.4.1

```

```

C:\laragon\www\Flutter_API\api-breeze\my_flutter_app>start ms-settings:developers

C:\laragon\www\Flutter_API\api-breeze\my_flutter_app>flutter pub outdated
Showing outdated packages.
[*] indicates versions that are not the latest available.

```

Package Name	Current	Upgradable	Resolvable	Latest
<b>direct dependencies:</b>				
flutter_secure_storage	*5.1.2	*5.1.2	9.2.2	9.2.2
http	*0.13.6	*0.13.6	1.2.2	1.2.2
<b>dev dependencies:</b>				
flutter_lints	*4.0.0	*4.0.0	5.0.0	5.0.0
<b>transitive dependencies:</b>				
async	*2.11.0	*2.11.0	*2.11.0	2.12.0
characters	*1.3.0	*1.3.0	*1.3.0	1.3.1
collection	*1.18.0	*1.18.0	*1.18.0	1.19.1
flutter_secure_storage_macos	*1.1.2	*1.1.2	3.1.2	3.1.2
flutter_secure_storage_windows	*1.1.3	*1.1.3	3.1.2	3.1.2
http_parser	*4.0.2	*4.0.2	*4.0.2	4.1.1
js	*0.6.7	*0.6.7	*0.6.7	0.7.1
material_color_utilities	*0.11.1	*0.11.1	*0.11.1	0.12.0
meta	*1.15.0	*1.15.0	*1.15.0	1.16.0
path	*1.9.0	*1.9.0	*1.9.0	1.9.1
path_provider	-	-	2.1.5	2.1.5
path_provider_android	-	-	2.2.12	2.2.12
path_provider_foundation	-	-	2.4.0	2.4.0
string_scanner	*1.2.0	*1.2.0	*1.2.0	1.4.0
win32	-	-	5.8.0	5.8.0
<b>transitive dev dependencies:</b>				
boolean_selector	*2.1.1	*2.1.1	*2.1.1	2.1.2
clock	*1.1.1	*1.1.1	*1.1.1	1.1.2



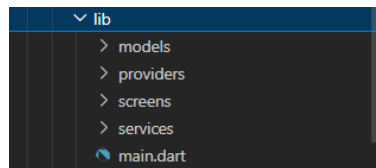
## Buat Splashscreen dengan animasi dari Lottie File

### Langkah 2: Mengatur Struktur Proyek

1

Buat folder berikut untuk mengatur kode Anda dengan lebih baik:

- lib/screens/ untuk menyimpan file layar (UI).
- lib/services/ untuk layanan HTTP dan manajemen API.
- lib/models/ untuk model data.
- lib/providers/ untuk manajemen state menggunakan Provider.



### Langkah 3: Membuat Model Pengguna

1

Buat file user\_model.dart di lib/models/:

```
api-breeze > my_flutter_app > lib > models > user_model.dart > User > User.fromJson
1 class User {
2   final int id;
3   final String name;
4   final String email;
5
6   User({
7     required this.id,
8     required this.name,
9     required this.email,
10  });
11
12   factory User.fromJson(Map<String, dynamic> json) {
13     return User(
14       id: json['id'],
15       name: json['name'],
16       email: json['email'],
17     );
18   }
19 }
20
```

### Langkah 4: Membuat Layanan API

1

Buat file auth\_service.dart di lib/services/:

```

api-breeze > my_flutter_app > lib > providers > auth_provider.dart > AuthService > login
1  import 'dart:convert';
2  import 'package:http/http.dart' as http;
3  import 'package:flutter_secure_storage/flutter_secure_storage.dart';
4  import '../models/user_model.dart';
5
6  class AuthService {
7    final String apiUrl = 'http://your-laravel-api-url.com/api';
8    final storage = FlutterSecureStorage();
9
10   /// Fungsi untuk login
11   Future<bool> login(String email, String password) async {
12     final response = await http.post(
13       Uri.parse('$apiUrl/login'),
14       headers: {'Content-Type': 'application/json'},
15       body: jsonEncode({
16         'email': email,
17         'password': password,
18       })),
19     );
20
21     if (response.statusCode == 200) {
22       final data = jsonDecode(response.body);
23       await storage.write(key: 'token', value: data['token']); // Simpan token ke storage
24       return true;
25     } else {
26       return false; // Gagal login
27     }
28   }
29
30   /// Fungsi untuk mendapatkan profil user
31   Future<User?> getProfile() async {
32     final token = await storage.read(key: 'token'); // Ambil token dari storage
33     final response = await http.get(
34       Uri.parse('$apiUrl/profile'),
35       headers: {
36         'Content-Type': 'application/json',
37         'Authorization': 'Bearer $token',
38       },
39     );
40
41     if (response.statusCode == 200) {
42       final data = jsonDecode(response.body);
43       return User.fromJson(data['user']); // Parse data user dari JSON
44     } else {
45       return null;
46     }
47   }
48
49   /// Fungsi untuk logout
50   Future<void> logout() async {
51     await storage.delete(key: 'token'); // Hapus token dari storage
52   }
53 }
54

```

## Langkah 5: Menyusun State Management dengan Provider

1

Buat file auth\_provider.dart di lib/providers/:

```

api-breeze > my_flutter_app > lib > providers > auth_provider.dart > ...
1  import 'package:flutter/material.dart';
2  import '../models/user_model.dart';
3  import '../services/auth_services.dart';
4
5  class AuthProvider with ChangeNotifier {
6    final AuthService _authService = AuthService();
7    User? _user;
8
9    /// Getter untuk mendapatkan data pengguna
10    User? get user => _user;
11
12    /// Fungsi login
13    Future<bool> login(String email, String password) async {
14      bool success = await _authService.login(email, password);
15      if (success) {
16        _user = await _authService.getProfile(); // Ambil data user setelah login
17        notifyListeners(); // Beritahu listener jika ada perubahan
18      }
19      return success;
20    }
21
22    /// Fungsi logout
23    Future<void> logout() async {
24      await _authService.logout(); // Hapus token
25      _user = null; // Hapus data user
26      notifyListeners(); // Beritahu listener jika ada perubahan
27    }
28
29    /// Fungsi untuk memuat data pengguna saat aplikasi diluncurkan
30    Future<void> loadUser() async {
31      _user = await _authService.getProfile(); // Ambil profil user jika token masih valid
32      notifyListeners(); // Beritahu listener jika ada perubahan
33    }
34  }
35

```

## Langkah 6: Membuat Halaman Login

1 Buat file login\_screen.dart di lib/screens/:

```
api-breeze > my_flutter_app > lib > screens > login_screen.dart > LoginScreen > build
1  import 'package:flutter/material.dart';
2  import 'package:provider/provider.dart';
3  import '../providers/auth_provider.dart';
4
5  class LoginScreen extends StatelessWidget {
6    final TextEditingController emailController = TextEditingController();
7    final TextEditingController passwordController = TextEditingController();
8
9    @override
10   Widget build(BuildContext context) {
11     final authProvider = Provider.of<AuthProvider>(context);
12
13     return Scaffold(
14       appBar: AppBar(title: Text('login')),
15       body: Padding(
16         padding: const EdgeInsets.all(16.0),
17         child: Column(
18           crossAxisAlignment: CrossAxisAlignment.stretch,
19           children: [
20             TextField(
21               controller: emailController,
22               decoration: InputDecoration(
23                 labelText: 'Email',
24                 border: OutlineInputBorder(),
25               ), // InputDecoration
26               keyboardType: TextInputType.emailAddress,
27             ), // TextField
28             SizedBox(height: 16),
29             TextField(
30               controller: passwordController,
31               decoration: InputDecoration(
32                 labelText: 'Password',
33                 border: OutlineInputBorder(),
34               ), // InputDecoration
35               obscureText: true,
36             ), // TextField
37             SizedBox(height: 28),
38             ElevatedButton(
39               onPressed: () async {
40                 // Mengambil input dari TextField
41                 String email = emailController.text.trim();
42                 String password = passwordController.text.trim();
43
44                 // Memanggil fungsi login dari AuthProvider
45                 bool success = await authProvider.login(email, password);
46
47                 if (success) {
48                   Navigator.of(context).pushReplacementNamed('/profile');
49                 } else {
50                   ScaffoldMessenger.of(context).showSnackBar(
51                     SnackBar(
52                       content: Text('Login failed! Please check your credentials.'),
53                     ), // SnackBar
54                   );
55                 }
56               },
57               child: Text('Login'),
58             ), // ElevatedButton
59           ], // Column
60         ), // Padding
61       ); // Scaffold
62     );
63   }
64 }
65
```

## Langkah 7: Membuat Halaman Profil

1 Buat file profile\_screen.dart di lib/screens/:

```

api-breeze > my_flutter_app > lib > screens > profile_screen.dart > ...
1  import 'package:flutter/material.dart';
2  import 'package:provider/provider.dart';
3  import '../providers/auth_provider.dart';
4
5  class ProfileScreen extends StatelessWidget {
6    @override
7    Widget build(BuildContext context) {
8      final authProvider = Provider.of<AuthProvider>(context);
9      final user = authProvider.user;
10
11      return Scaffold(
12        appBar: AppBar(
13          title: Text('Profile'),
14          actions: [
15            IconButton(
16              icon: Icon(Icons.logout),
17              onPressed: () {
18                authProvider.logout();
19                Navigator.of(context).pushReplacementNamed('/login');
20              },
21            ), // IconButton
22          ],
23        ), // AppBar
24        body: Center(
25          child: user != null
26            ? Column(
27              mainAxisAlignment: MainAxisAlignment.center,
28              children: [
29                Text(
30                  'Welcome, ${user.name}!',
31                  style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
32                ), // Text
33                SizedBox(height: 10),
34                Text(
35                  'Email: ${user.email}',
36                  style: TextStyle(fontSize: 16),
37                ), // Text
38              ],
39            ) // Column
40            : CircularProgressIndicator(),
41        ), // Center
42      ); // Scaffold
43    }
44  }
45

```

## Langkah 8: Mengatur Routing dan Provider

1 Buka main.dart dan atur routing serta Provider:


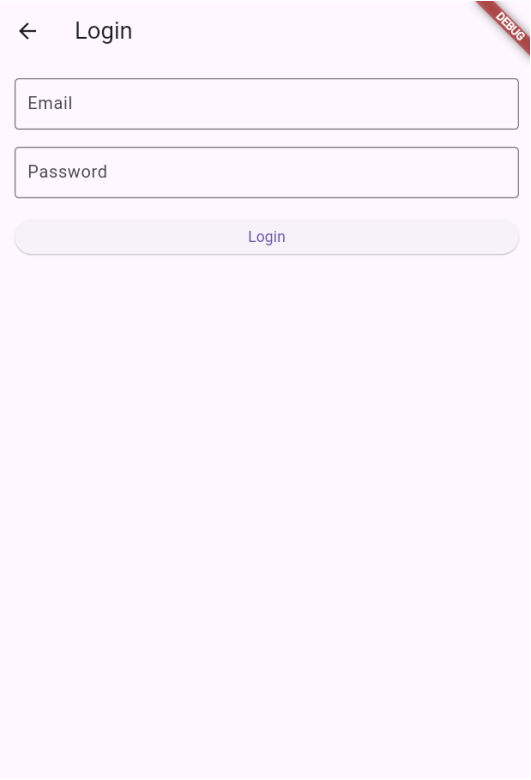
```

api-breeze > my_flutter_app > lib > main.dart > MyApp > build
/
Run | Debug | Profile
8  void main() {
9    runApp(MyApp());
10 }
11
12 class MyApp extends StatelessWidget {
13   @override
14   Widget build(BuildContext context) {
15     return MultiProvider(
16       providers: [
17         ChangeNotifierProvider(create: (_) => AuthProvider()),
18       ],
19       child: MaterialApp(
20         title: 'Flutter App',
21         theme: ThemeData(
22           primarySwatch: Colors.blue,
23         ), // ThemeData
24         initialRoute: '/',
25         routes: {
26           '/': (context) => SplashScreen(),
27           '/login': (context) => LoginScreen(),
28           '/profile': (context) => ProfileScreen(),
29         },
30       ), // MaterialApp
31     ); // MultiProvider
32   }
33 }
34

```

## Langkah 9: Menyiapkan Splash Screen

1	<p>Buat file splash_screen.dart di lib/screens/:</p> <pre> api-breeze &gt; my_flutter_app &gt; lib &gt; screens &gt; splash_screen.dart &gt; _SplashScreenState &gt; build 1  import 'package:flutter/material.dart'; 2  import 'package:provider/provider.dart'; 3  import '../providers/auth_provider.dart'; 4 5  class SplashScreen extends StatefulWidget { 6    @override 7    _SplashScreenState createState() =&gt; _SplashScreenState(); 8  } 9 10 class _SplashScreenState extends State&lt;SplashScreen&gt; { 11   @override 12   void initState() { 13     super.initState(); 14     _checkLoginStatus(); 15   } 16 17   /// Memeriksa status login pengguna 18   void _checkLoginStatus() async { 19     final authProvider = Provider.of&lt;AuthProvider&gt;(context, listen: false); 20     await authProvider.loadUser(); 21 22     // Navigasikan berdasarkan status login pengguna 23     if (authProvider.user != null) { 24       Navigator.of(context).pushReplacementNamed('/profile'); 25     } else { 26       Navigator.of(context).pushReplacementNamed('/login'); 27     } 28   } 29 30   @override 31   Widget build(BuildContext context) { 32     return Scaffold( 33       body: Center( 34         child: Text( 35           'My Flutter App', 36           style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold), 37         ), // Text 38       ), // Center 39     ); // Scaffold 40   } 41 } 42 </pre>
<b>Langkah 10: Menjalankan Aplikasi</b>	
1	<p>Pastikan API Laravel Anda sudah berjalan dan endpoint login serta profil sudah tersedia. Jalankan aplikasi Flutter:</p> <ul style="list-style-type: none"> <li>• Page Splash</li> </ul>

		 <p>A light purple rectangular mockup representing a Flutter app splash screen. It features the text "My Flutter App" centered in a bold, black font. A small red "DESIGN" label is positioned in the top right corner.</p>
	<ul style="list-style-type: none"><li>• Page Login</li></ul>	 <p>A light purple rectangular mockup representing a Flutter app login screen. It includes a back arrow and the text "Login" at the top left. Below this are two input fields labeled "Email" and "Password". A rounded button labeled "Login" is centered below the fields. A small red "DESIGN" label is in the top right corner.</p>
	<ul style="list-style-type: none"><li>• Page Profile</li></ul>	

