

Jobsheet API
Mata Kuliah Pemrograman Mobile



Oleh:

Rafif Ramadhani Wibowo 2241760111 SIB 3D

Jurusan Teknologi Informasi
Program Studi D4 Sistem Informasi Bisnis
Politeknik Negeri Malang
Tahun
2024

Step 1: Install Laravel

```
PS C:\laragon\www> composer create-project laravel/laravel api-breeze
Creating a "laravel/laravel" project at "./api-breeze"
Installing laravel/laravel (v11.3.3)
- Installing laravel/laravel (v11.3.3): Extracting archive
Created project in C:\laragon\www\api-breeze
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 107 installs, 0 updates, 0 removals
- Locking brick/math (0.12.1)
- Locking carbonphp/carbon-doctrine-types (3.2.0)
- Locking dflydev/dot-access-data (v3.0.3)
```

Step 2: Install Laravel Breeze

```
PS C:\laragon\www> cd api-breeze
PS C:\laragon\www\api-breeze> composer require laravel/breeze --dev
./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 (v2.2.5)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading laravel/breeze (v2.2.5)
- Installing laravel/breeze (v2.2.5): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
```

```
PS C:\laragon\www\api-breeze> php artisan breeze:install api
./composer.json has been updated
Running composer update laravel/sanctum
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking laravel/sanctum (v4.0.4)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading laravel/sanctum (v4.0.4)
- Installing laravel/sanctum (v4.0.4): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
```

Step 3: Configure the Database and Run Migrations

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel11_api
DB_USERNAME=root
DB_PASSWORD=
```

```
PS C:\laragon\www\api-breeze> php artisan migrate
```

```
WARN The database 'laravel11_api' does not exist on the 'mysql' connection.
```

```
Would you like to create it? (yes/no) [yes]
```

```
> yes
```

```
INFO Preparing database.
```

```
Creating migration table ..... 52.38ms DONE
```

```
INFO Running migrations.
```

```
0001_01_01_000000_create_users_table ..... 194.69ms DONE
```

```
0001_01_01_000001_create_cache_table ..... 44.58ms DONE
```

```
0001_01_01_000002_create_jobs_table ..... 147.89ms DONE
```

```
2024_11_25_085011_create_personal_access_tokens_table ..... 96.79ms DONE
```

Step 4: Create Authentication Endpoints

```
.env  api.php  RegisteredUserController.php  AuthenticatedSessionController.php 1
api-breeze1 > routes > api.php
1  <?php
2
3  use App\Http\Controllers\Auth\AuthenticatedSessionController;
4  use App\Http\Controllers\Auth\RegisteredUserController;
5  use Illuminate\Support\Facades\Route;
6
7  Route::post('/register', [RegisteredUserController::class, 'store']);
8  Route::post('/login', [AuthenticatedSessionController::class, 'store']);
9  Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])->middleware('auth:sanctum');
10
```

Step 5: Update Controllers

```
.env RegisteredUserController.php X AuthenticatedSessionController.php Extension:
api-breeze > app > Http > Controllers > Auth > RegisteredUserController.php > RegisteredUserController

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         $user = User::create([
22             'name' => $request->name,
23             'email' => $request->email,
24             'password' => Hash::make($request->password),
25         ]);
26         event(new Registered($user));
27         $token = $user->createToken('auth_token')->plainTextToken;
28         return response()->json([
29             'access_token' => $token,
30             'token_type' => 'Bearer',
31             'user' => $user
32         ]);
33     }
34 }
```

```
.env RegisteredUserController.php X AuthenticatedSessionController.php X Extension: Thund
api-breeze > app > Http > Controllers > Auth > AuthenticatedSessionController.php > ...

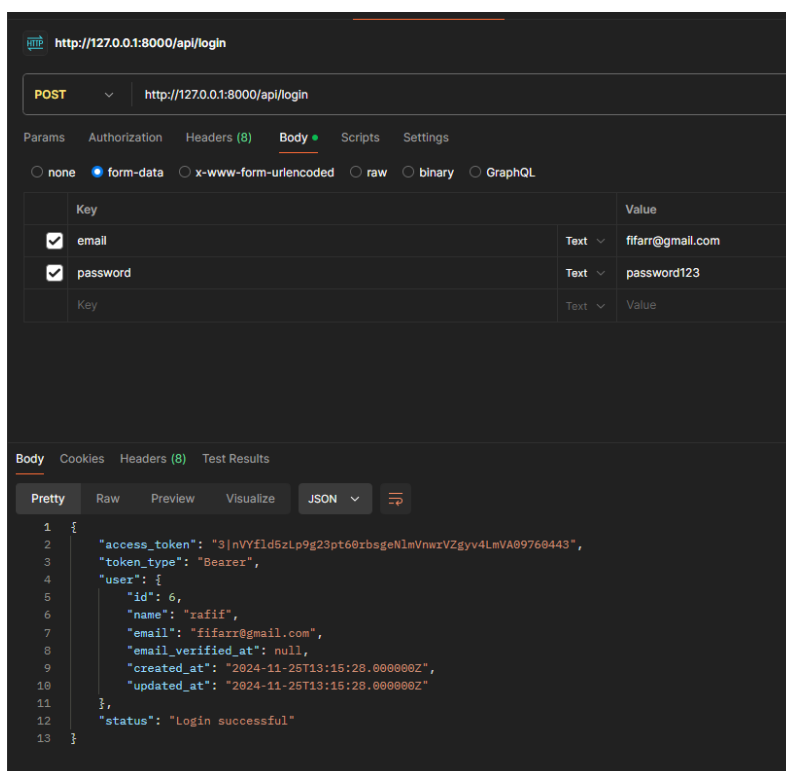
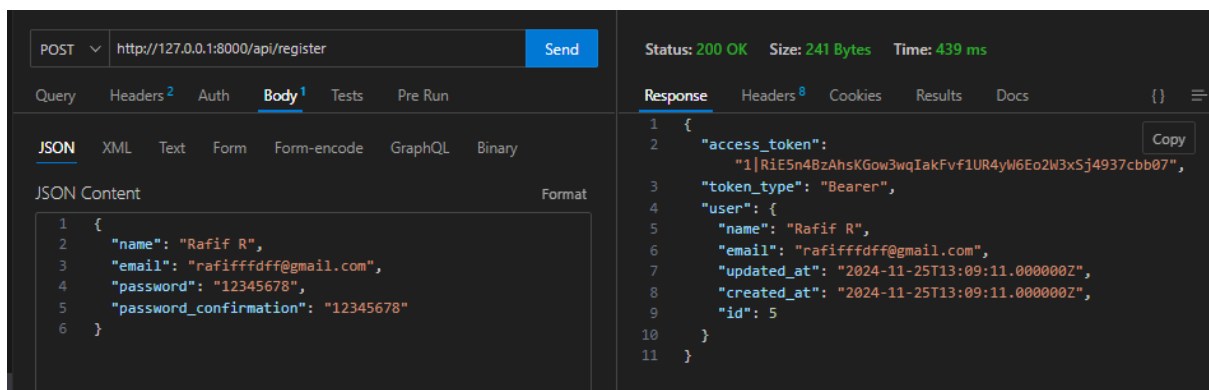
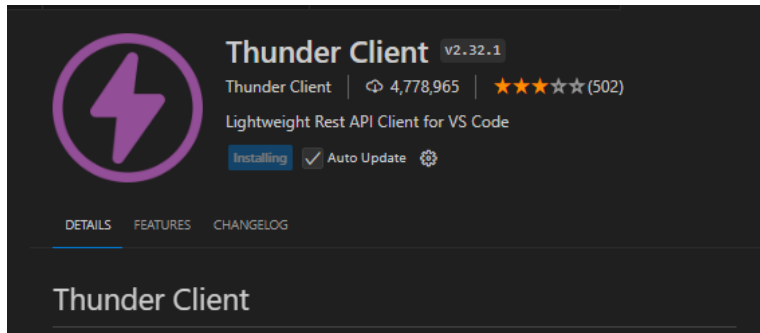
1 <?php
2
3 namespace App\Http\Controllers\Auth;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Auth;
7 use App\Http\Controllers\Controller;
8
9 class AuthenticatedSessionController extends Controller
10 {
11     public function store(Request $request)
12     {
13         $request->validate([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

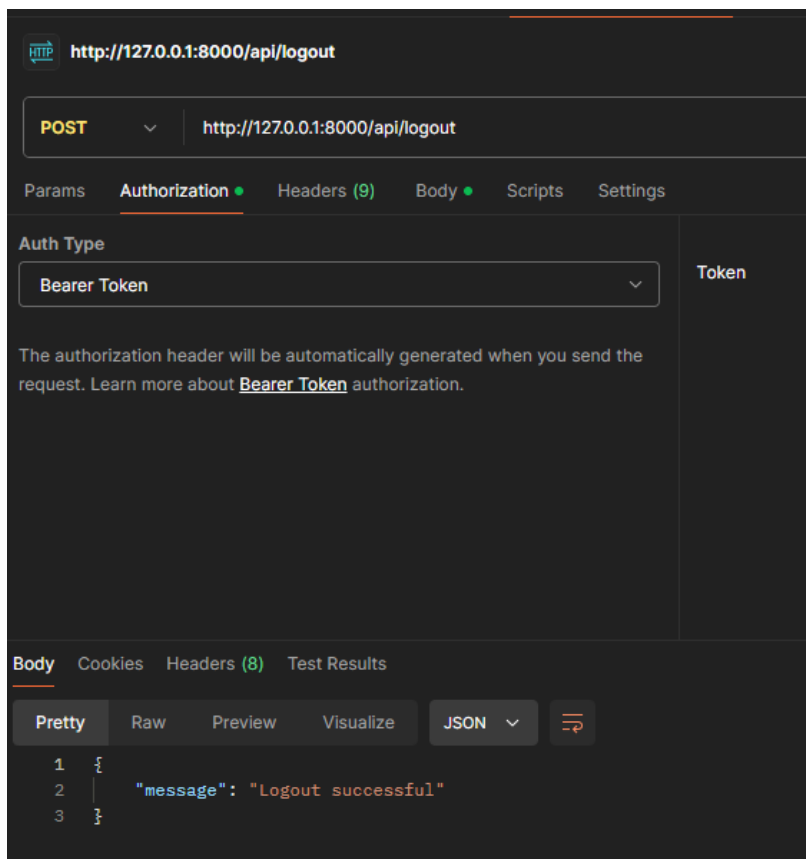
```
PS C:\laragon\www\api-breeze> php artisan serve
forking is not supported on this platform

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server
```

Step 6: Check following API





Pembuatan Aplikasi Mobile Flutter,

Langkah 1: Persiapan Proyek Flutter

```
PS C:\Users\Rafif\Documents\Kuliah\SEMESTER 5\PEM MOBILE> flutter create my_flutter_app
Creating project my_flutter_app...
Resolving dependencies in `my_flutter_app`... (3.2s)
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/
```

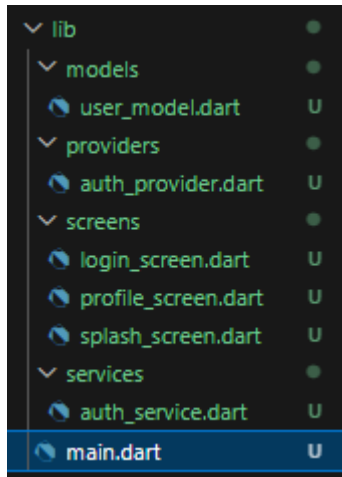
```
10 dependencies:
11   flutter:
12     sdk: flutter
13   http: ^0.13.3
14   shared_preferences: ^2.0.6
15   provider: ^6.0.0
16   flutter_secure_storage: ^5.0.2
17
```

```

PS C:\Users\Rafif\Documents\Kuliah\SEMESTER 5\PEM MOBILE\my_flutter_app> flutter pub get
Resolving dependencies... (1.0s)
Downloading packages...
  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)
  flutter_lints 4.0.0 (5.0.0 available)
  flutter_secure_storage 5.1.2 (9.2.2 available)
  flutter_secure_storage_macos 1.1.2 (3.1.2 available)
  flutter_secure_storage_windows 1.1.3 (3.1.2 available)
  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)

```

Langkah 2: Mengatur Struktur Proyek



Langkah 3: Membuat Model Pengguna

```

lib > models > user_model.dart > ...
1  class User {
2      final int id;
3      final String name;
4      final String email;
5      User({required this.id, required this.name, required this.email});
6      factory User.fromJson(Map<String, dynamic> json) {
7          return User(
8              id: json['id'],
9              name: json['name'],
10             email: json['email'],
11         );
12     }
13 }

```

Langkah 4: Membuat Layanan API

```
lib > services > auth_service.dart > AuthService > getProfile
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    Future<bool> login(String email, String password) async {
10     final response = await http.post(
11       Uri.parse('$apiUrl/login'),
12       headers: {'Content-Type': 'application/json'},
13       body: jsonEncode({'email': email, 'password': password}),
14     );
15     if (response.statusCode == 200) {
16       final data = jsonDecode(response.body);
17       await storage.write(key: 'token', value: data['token']);
18       return true;
19     } else {
20       return false;
21     }
22   }
23
24   Future<User?> getProfile() async {
25     final token = await storage.read(key: 'token');
26     final response = await http.get(
27       Uri.parse('$apiUrl/profile'),
28       headers: {
29         'Content-Type': 'application/json',
30         'Authorization': 'Bearer $token',
31       },
32     );
33     if (response.statusCode == 200) {
34       final data = jsonDecode(response.body);
35       return User.fromJson(data['user']);
36     } else {
37       return null;
38     }
39   }
40
41   Future<void> logout() async {
42     await storage.delete(key: 'token');
43   }
44 }
```

Langkah 5: Menyusun State Management dengan Provider

```
lib > providers > auth_provider.dart > ...
1  import 'package:flutter/material.dart';
2  import '../models/user_model.dart';
3  import '../services/auth_service.dart';
4
5  class AuthProvider with ChangeNotifier {
6    final AuthService _authService = AuthService();
7    User? _user;
8    User? get user => _user;
9    Future<bool> login(String email, String password) async {
10     bool success = await _authService.login(email, password);
11     if (success) {
12       _user = await _authService.getProfile();
13       notifyListeners();
14     }
15     return success;
16   }
17
18   Future<void> logout() async {
19     await _authService.logout();
20     _user = null;
21     notifyListeners();
22   }
23
24   Future<void> loadUser() async {
25     _user = await _authService.getProfile();
26     notifyListeners();
27   }
28 }
29
```


Langkah 6: Membuat Halaman Login

```
lib > screens > login_screen.dart > ...
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   @override
9   Widget build(BuildContext context) {
10     final authProvider = Provider.of<AuthProvider>(context);
11     return Scaffold(
12       appBar: AppBar(title: Text('Login')),
13       body: Padding(
14         padding: EdgeInsets.all(16.0),
15         child: Column(
16           children: [
17             TextField(
18               controller: emailController,
19               decoration: InputDecoration(labelText: 'Email'),
20             ), // TextField
21             TextField(
22               controller: passwordController,
23               decoration: InputDecoration(labelText: 'Password'),
24               obscureText: true,
25             ), // TextField
26             SizedBox(height: 20),
27             ElevatedButton(
28               onPressed: () async {
29                 bool success = await authProvider.login(
30                   emailController.text,
31                   passwordController.text,
32                 );
33                 if (success) {
34                   Navigator.of(context).pushReplacementNamed('/profile');
35                 } else {
36                   ScaffoldMessenger.of(context).showSnackBar(SnackBar(
37                     content: Text('Login failed!'),
38                   )); // SnackBar
39                 }
40               },
41               child: Text('Login'),
42             ), // ElevatedButton
43           ],
44         ), // Column
45       ), // Padding
46     ); // Scaffold
47   }
48 }
```

Langkah 7: Membuat Halaman Profil

```
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     return Scaffold(
11       appBar: AppBar(
12         title: Text('Profile'),
13         actions: [
14           IconButton(
15             icon: Icon(Icons.logout),
16             onPressed: () {
17               authProvider.logout();
18               Navigator.of(context).pushReplacementNamed('/login');
19             },
20           ), // IconButton
21         ],
22       ), // AppBar
23       body: Center(
24         child: user != null
25           ? Column(
26             mainAxisAlignment: MainAxisAlignment.center,
27             children: [
28               Text('Welcome, ${user.name}!'),
29               Text('Email: ${user.email}'),
30             ],
31           ) // Column
32           : CircularProgressIndicator(),
33       ), // Center
34     ); // Scaffold
35   }
36 }
```

Langkah 8: Mengatur Routing dan Provider

```
lib > main.dart > MyApp
1 import 'package:flutter/material.dart';
2 import 'package:provider/provider.dart';
3 import 'screens/splash_screen.dart';
4 import 'screens/login_screen.dart';
5 import 'screens/profile_screen.dart';
6 import 'providers/auth_provider.dart';
7
8 Run | Debug | Profile
9 void main() {
10   runApp(MyApp());
11 }
12
13 class MyApp extends StatelessWidget {
14   @override
15   Widget build(BuildContext context) {
16     return MultiProvider(
17       providers: [
18         ChangeNotifierProvider(create: (_) => AuthProvider()),
19       ],
20       child: MaterialApp(
21         title: 'Flutter App',
22         theme: ThemeData(
23           primarySwatch: Colors.blue,
24         ), // ThemeData
25         initialRoute: '/',
26         routes: {
27           '/': (context) => SplashScreen(),
28           '/login': (context) => LoginScreen(),
29           '/profile': (context) => ProfileScreen(),
30         },
31       ), // MaterialApp
32     ); // MultiProvider
33   }
34 }
```

Langkah 9: Menyiapkan Splash Screen

```
lib > screens > splash_screen.dart > _SplashScreenState > _checkLoginStatus
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() => _SplashScreenState();
8 }
9
10 class _SplashScreenState extends State<SplashScreen> {
11   @override
12   void initState() {
13     super.initState();
14     _checkLoginStatus();
15   }
16
17   void _checkLoginStatus() async {
18     final authProvider = Provider.of<AuthProvider>(context, listen: false);
19     await authProvider.loadUser();
20     if (authProvider.user != null) {
21       Navigator.of(context).pushReplacementNamed('/profile');
22     } else {
23       Navigator.of(context).pushReplacementNamed('/login');
24     }
25   }
26
27   @override
28   Widget build(BuildContext context) {
29     return Scaffold(
30       body: Center(
31         child: Text('My Flutter App', style: TextStyle(fontSize: 24)),
32       ), // Center
33     ); // Scaffold
34   }
35 }
36 }
```