PEMROGRAMAN MOBILE

“Flutter Laravel API”

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

Oleh:

YUNIKA PUTERI DWI ANTIKA

2241760048 / SIB-3E

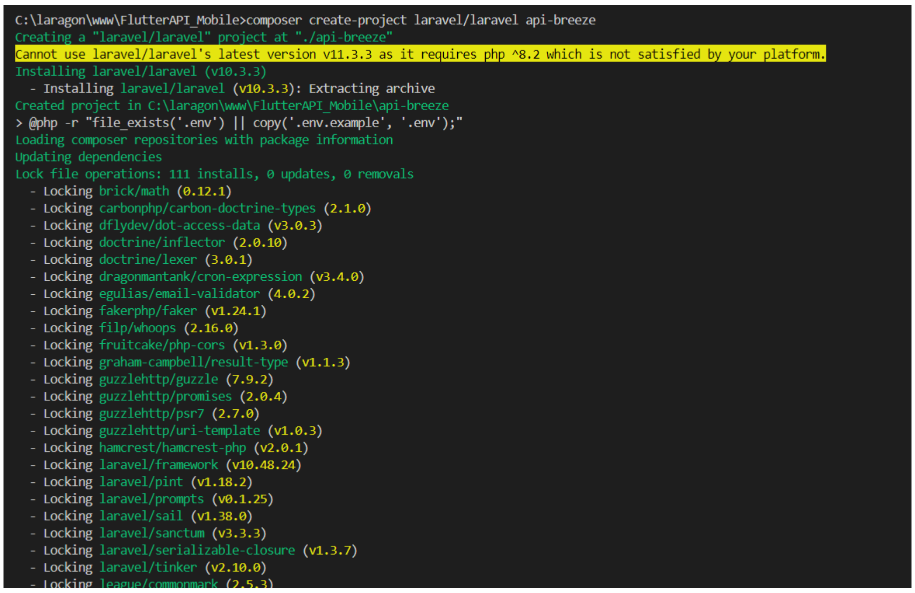
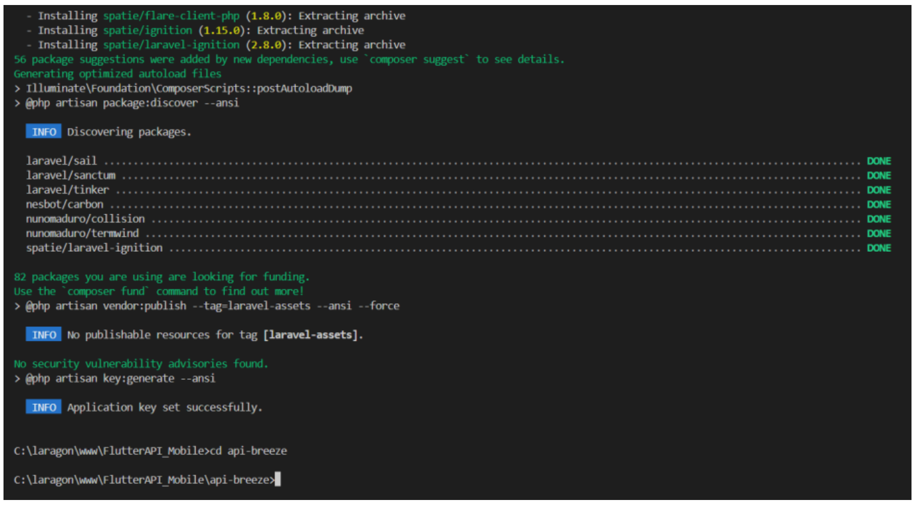
PROGRAM STUDI SISTEM INFORMASI BISNIS

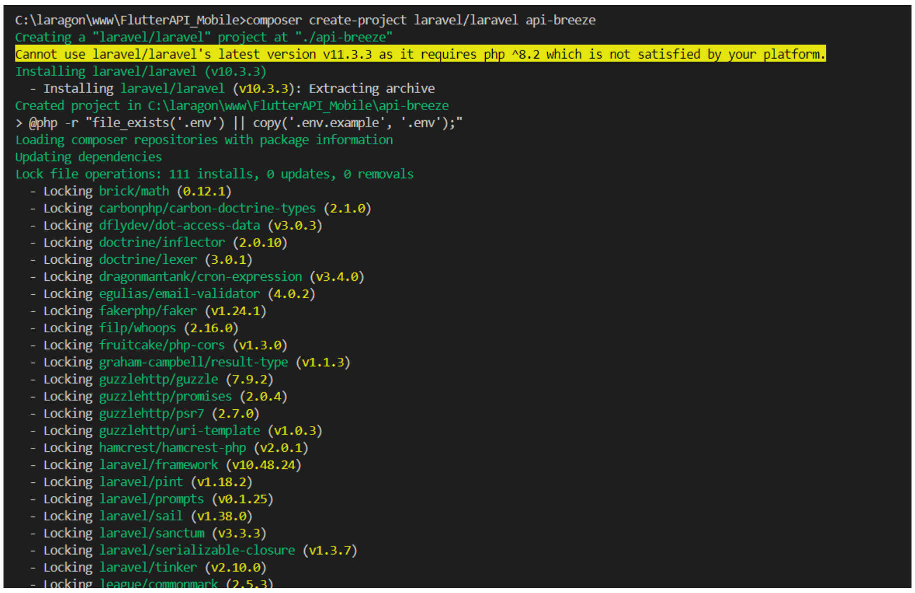
JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

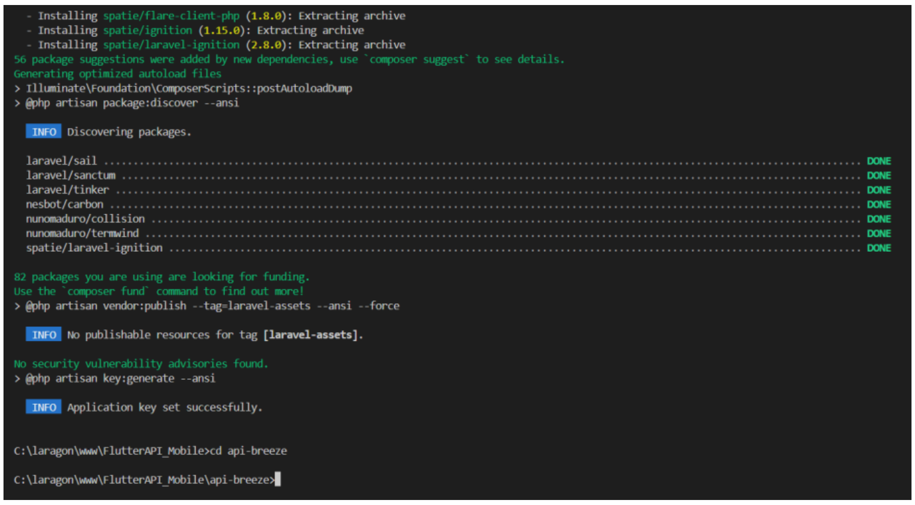
2024

Membuat API dengan Laravel Breeze di Laravel 11

Link Github	
https://github.com/yunikaputri/Pemrograman-Mobile/tree/main/Pertemuan%2013	
Langkah	**Jawaban/Deskripsi**
Langkah 1: Instal Laravel	
1	Pertama, buat proyek Laravel baru menggunakan penginstal Laravel atau Composer.  

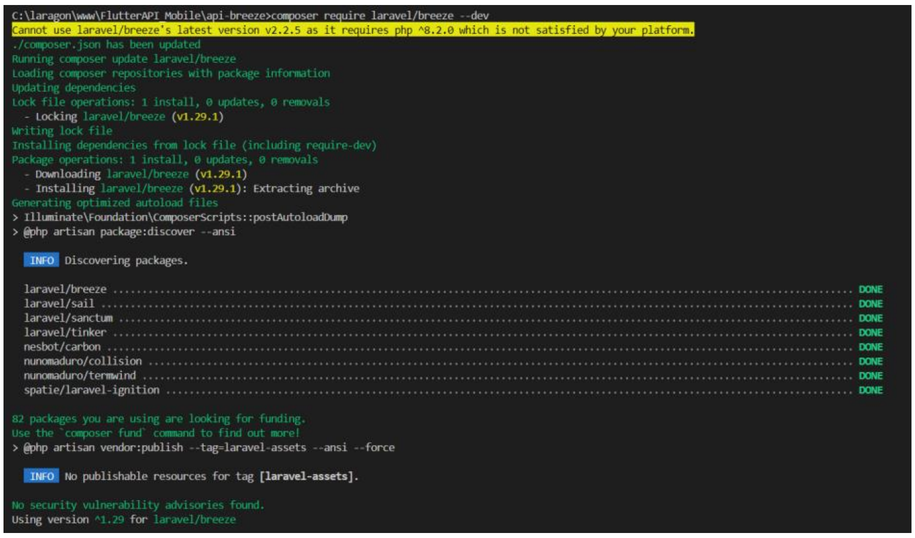


```
C:\laragon\www\FlutterAPI_Mobile>composer create-project laravel/laravel api-breeze
Creating a "laravel/laravel" project at "./api-breeze"
Cannot use laravel/laravel's latest version v11.3.3 as it requires php ^8.2 which is not satisfied by your platform.
Installing laravel/laravel (v10.3.3)
  - Installing laravel/laravel (v10.3.3): Extracting archive
Created project in C:\laragon\www\FlutterAPI_Mobile\api-breeze
> @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)
```

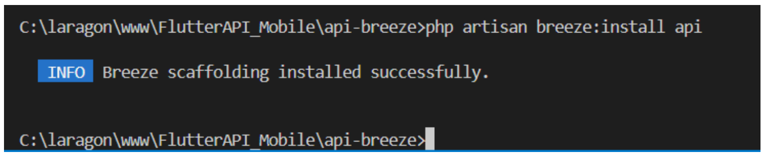


```
  - Installing spatie/flare-client-php (1.8.0): Extracting archive
  - Installing spatie/ignition (1.15.0): Extracting archive
  - Installing spatie/laravel-ignition (2.8.0): Extracting archive
56 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @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!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

  INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
> @php artisan key:generate --ansi

  INFO  Application key set successfully.


C:\laragon\www\FlutterAPI_Mobile>cd api-breeze

C:\laragon\www\FlutterAPI_Mobile\api-breeze>
```

Langkah 2: Instal Laravel Breeze	
1	Selanjutnya, instal Laravel Breeze dan dependensinya.



```
C:\laragon\www\FlutterAPI_Mobile\api-breeze>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
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @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!
> @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
```

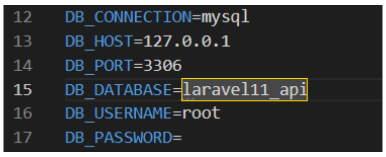


```
C:\laragon\www\FlutterAPI_Mobile\api-breeze>php artisan breeze:install api

  INFO  Breeze scaffolding installed successfully.

C:\laragon\www\FlutterAPI_Mobile\api-breeze>
```

Perintah ini akan menginstal Breeze dan menyiapkan perancah yang diperlukan untuk autentikasi API.

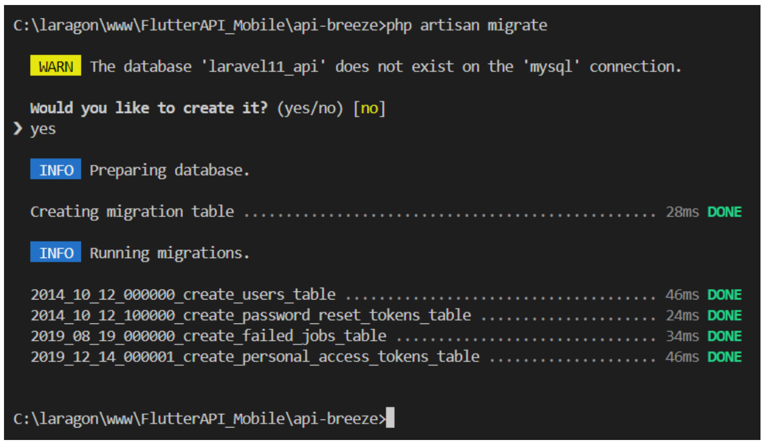
Langkah 3: Konfigurasikan Database dan Jalankan Migrasi

1 Perbarui file .env Anda dengan kredensial basis data Anda:

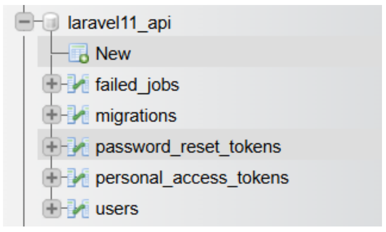


```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel11_api
DB_USERNAME=root
DB_PASSWORD=
```

2 Jalankan migrasi untuk menyiapkan tabel database Anda:



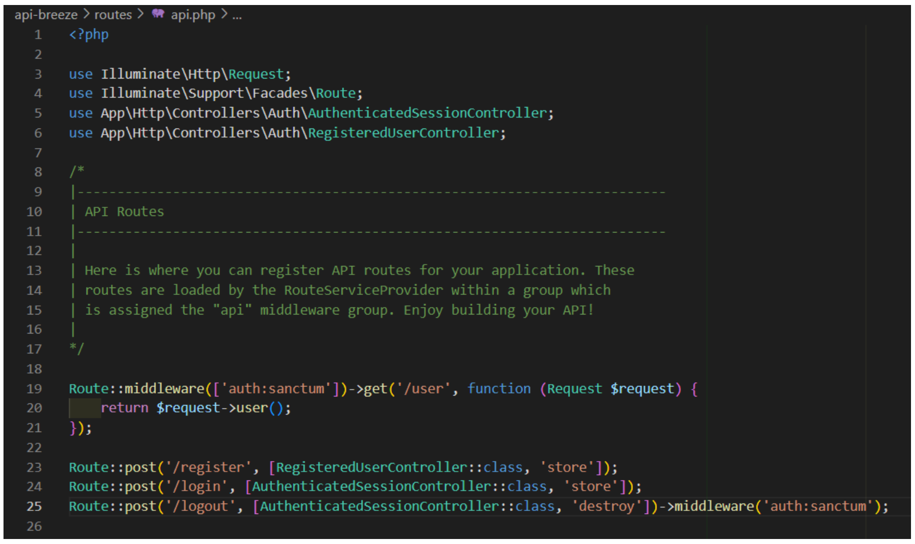
```
C:\laragon\www\FlutterAPI_Mobile\api-breeze>php artisan migrate

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

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



```
laravel11_api
  New
  failed_jobs
  migrations
  password_reset_tokens
  personal_access_tokens
  users
```

Langkah 4: Buat Titik Akhir Autentikasi

1 Laravel Breeze menyediakan titik akhir yang diperlukan untuk registrasi, login, dan logout. Rute-rute tersebut didefinisikan dalam routes/api.php.



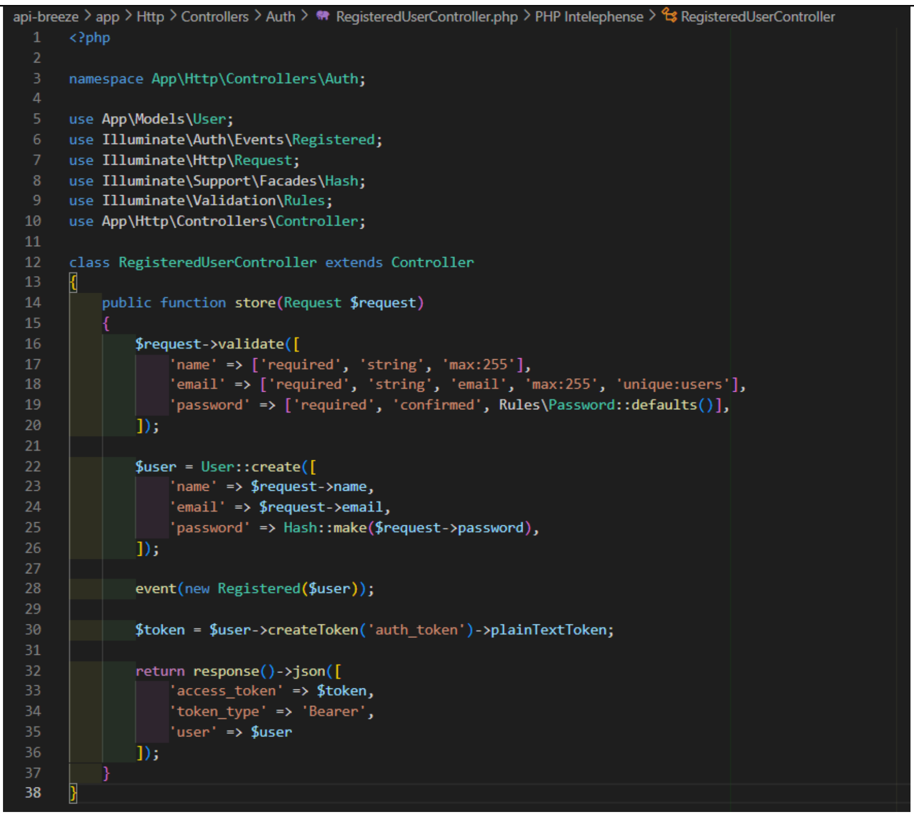
api-breeze > routes > api.php > ...

```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\RegisteredUserController;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::middleware(['auth:sanctum'])->get('/user', function (Request $request) {
    return $request->user();
});

Route::post('/register', [RegisteredUserController::class, 'store']);
Route::post('/login', [AuthenticatedSessionController::class, 'store']);
Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])->middleware('auth:sanctum');

```

Langkah 5: Perbarui Pengontrol

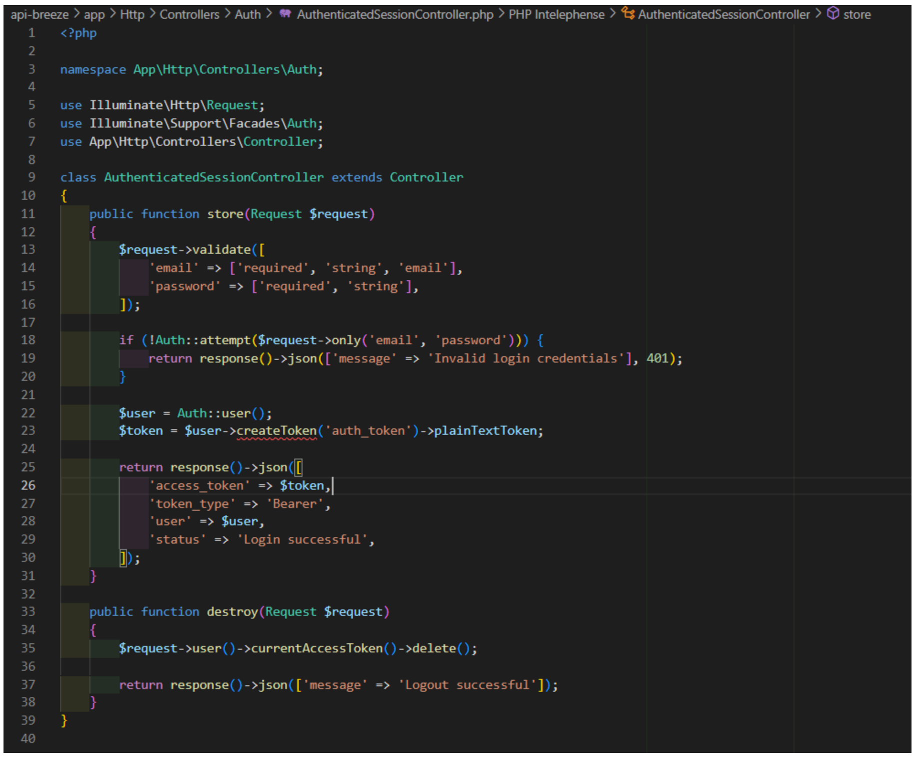
1 Ubah RegisteredUserControllerdan AuthenticatedSessionControlleruntuk mengembalikan respons JSON.

- RegisteredUserController.php


api-breeze > app > Http > Controllers > Auth > RegisteredUserController.php > PHP Intelephense > RegisteredUserController

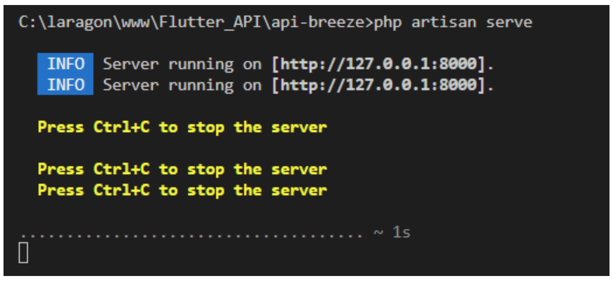
```php
<?php

namespace App\Http\Controllers\Auth;

use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;
use App\Http\Controllers\Controller;

class RegisteredUserController extends Controller
{
    public function store(Request $request)
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'confirmed', Rules\Password::defaults()],
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        event(new Registered($user));

        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json([
            'access_token' => $token,
            'token_type' => 'Bearer',
            'user' => $user
        ]);
    }
}
```

- AuthenticatedSessionController.php


api-breeze > app > Http > Controllers > Auth > AuthenticatedSessionController.php > PHP Intelephense > AuthenticatedSessionController > store

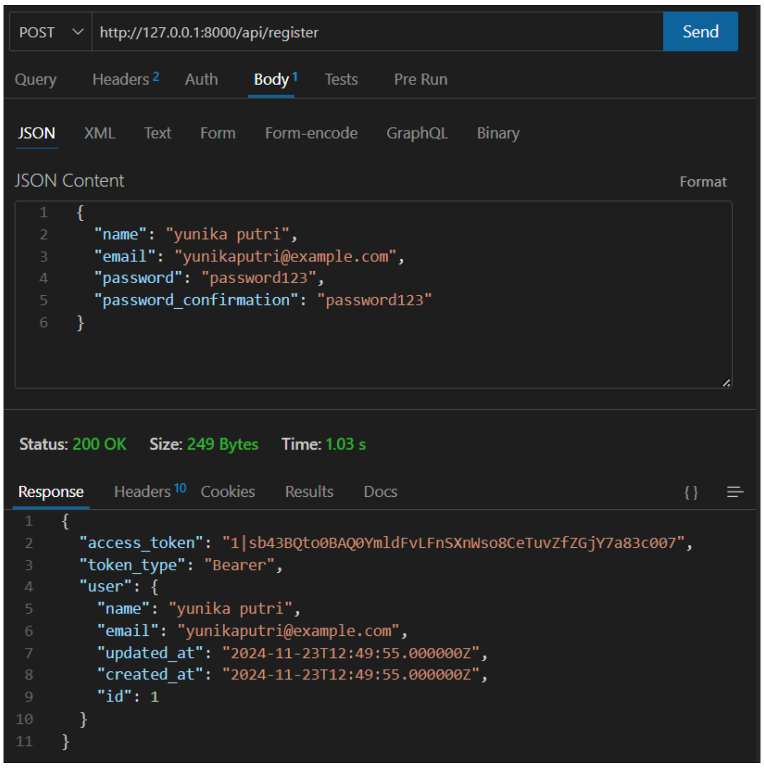
```php
<?php

namespace App\Http\Controllers\Auth;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Http\Controllers\Controller;

class AuthenticatedSessionController extends Controller
{
    public function store(Request $request)
    {
        $request->validate([
            'email' => ['required', 'string', 'email'],
            'password' => ['required', 'string'],
        ]);

        if (!Auth::attempt($request->only('email', 'password'))) {
            return response()->json(['message' => 'Invalid login credentials'], 401);
        }

        $user = Auth::user();
        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json([
            'access_token' => $token,
            'token_type' => 'Bearer',
            'user' => $user,
            'status' => 'Login successful',
        ]);
    }

    public function destroy(Request $request)
    {
        $request->user()->currentAccessToken()->delete();

        return response()->json(['message' => 'Logout successful']);
    }
}

```

Langkah 5: Jalankan Aplikasi Laravel

1	

```
C:\laragon\www\Flutter_API\api-breeze>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
```

Langkah 6: Periksa API berikut

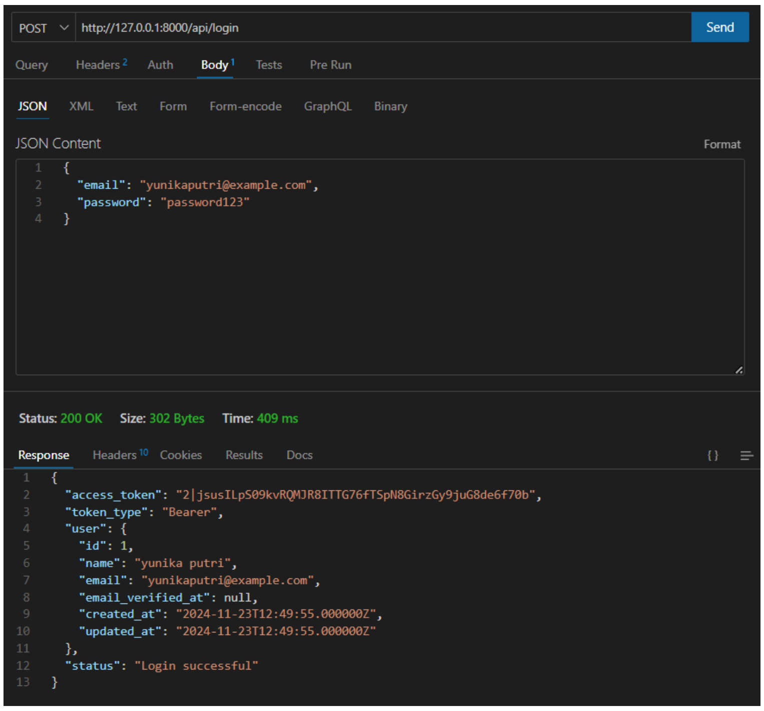
1	Uji API Anda dengan Thunder Client 

```
http://127.0.0.1:8000/api/register
```

```
{
  "name": "yunika putri",
  "email": "yunikaputri@example.com",
  "password": "password123",
  "password_confirmation": "password123"
}
```

Status: 200 OK   Size: 249 Bytes   Time: 1.03 s

```
{
  "access_token": "1|sb43BQto0BAQ0YmldFvLFnSXnWso8CeTuvZfZGjY7a83c007",
  "token_type": "Bearer",
  "user": {
    "name": "yunika putri",
    "email": "yunikaputri@example.com",
    "updated_at": "2024-11-23T12:49:55.000000Z",
    "created_at": "2024-11-23T12:49:55.000000Z",
    "id": 1
  }
}
```

```
POST  http://127.0.0.1:8000/api/login
```

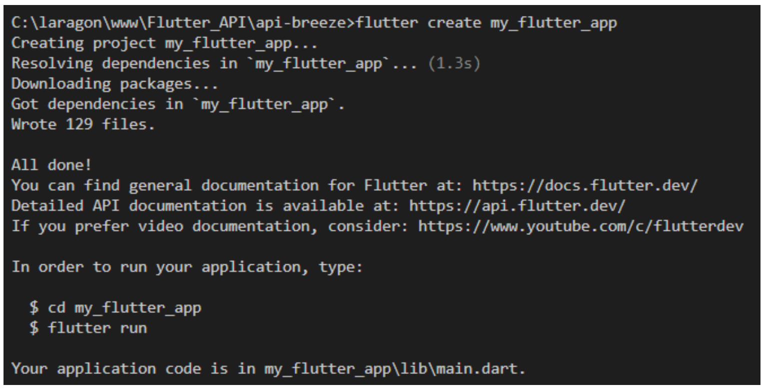
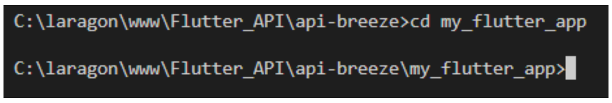
JSON Content

```
{
  "email": "yunikaputri@example.com",
  "password": "password123"
}
```

Status: 200 OK   Size: 302 Bytes   Time: 409 ms

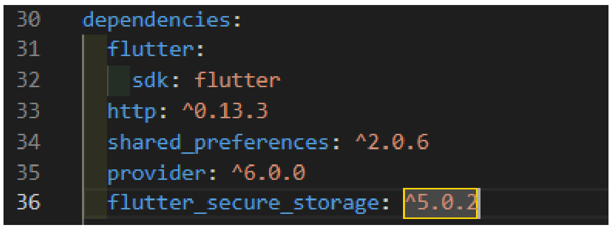
```
{
  "access_token": "2|jsusILpS09kvRQMJR8ITTG76fTSpN8GirzGy9juG8de6f70b",
  "token_type": "Bearer",
  "user": {
    "id": 1,
    "name": "yunika putri",
    "email": "yunikaputri@example.com",
    "email_verified_at": null,
    "created_at": "2024-11-23T12:49:55.000000Z",
    "updated_at": "2024-11-23T12:49:55.000000Z"
  },
  "status": "Login successful"
}
```

Membuat Aplikasi Mobile Flutter

Langkah	Jawaban/Deskripsi
Langkah 1: Persiapan Proyek Flutter	
1	Buat Proyek Flutter Baru:  
2	**Tambahkan Dependencies:** Buka pubspec.yaml dan tambahkan beberapa dependencies yang diperlukan:

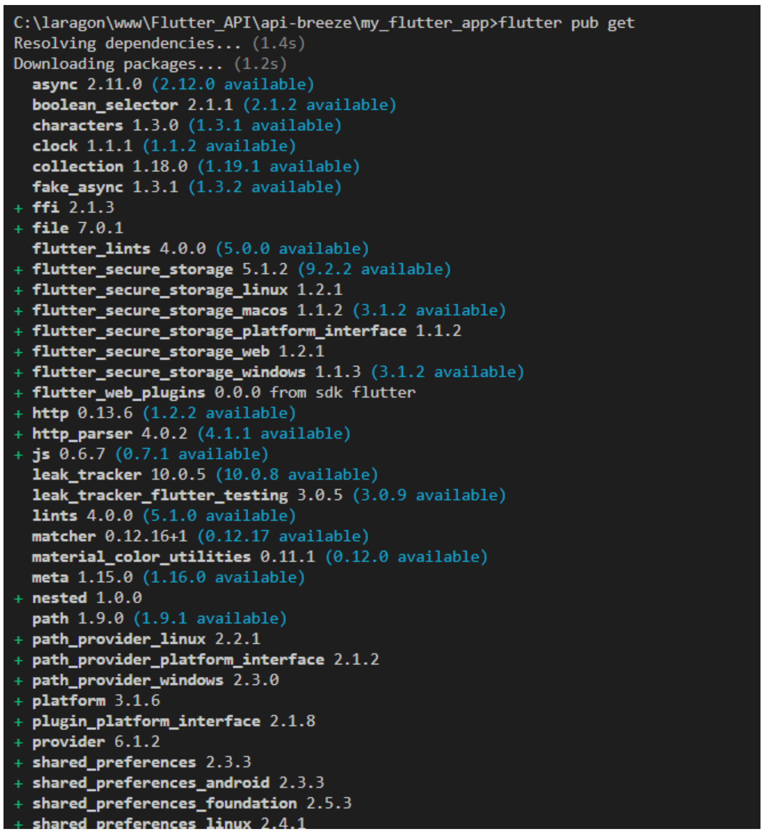
```
C:\laragon\www\Flutter_API\api-breeze>flutter create my_flutter_app
Creating project my_flutter_app...
Resolving dependencies in `my_flutter_app`... (1.3s)
Downloading packages...
Got dependencies in `my_flutter_app`.
Wrote 129 files.

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>cd my_flutter_app

C:\laragon\www\Flutter_API\api-breeze\my_flutter_app>
```

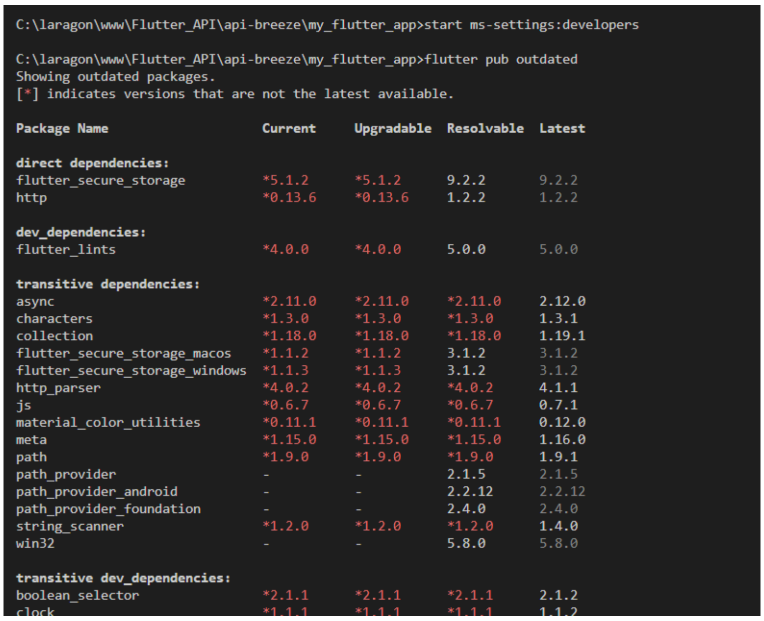


```
dependencies:
  flutter:
    sdk: flutter
  http: ^0.13.3
  shared_preferences: ^2.0.6
  provider: ^6.0.0
  flutter_secure_storage: ^5.0.2
```

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
```



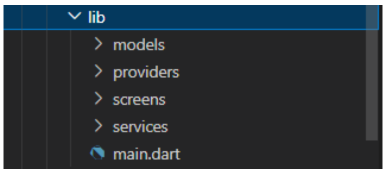
```
C:\laragon\www\Flutter_API\api-breeze\my_flutter_app>start ms-settings:developers

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

direct dependencies:
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

dev_dependencies:
flutter_lints                   *4.0.0     *4.0.0      5.0.0       5.0.0

transitive dependencies:
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

transitive dev_dependencies:
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

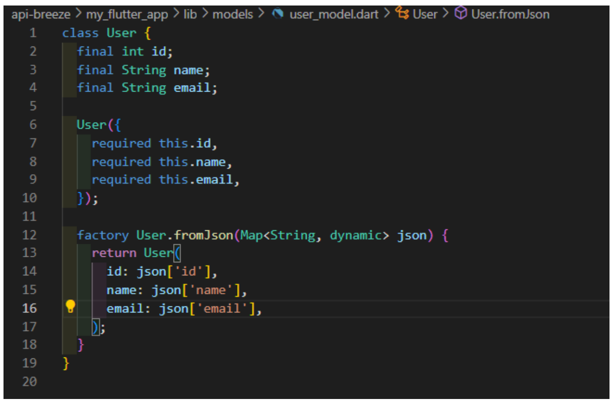
1 Buat folder berikut untuk mengatur kode Anda dengan lebih baik:

- 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

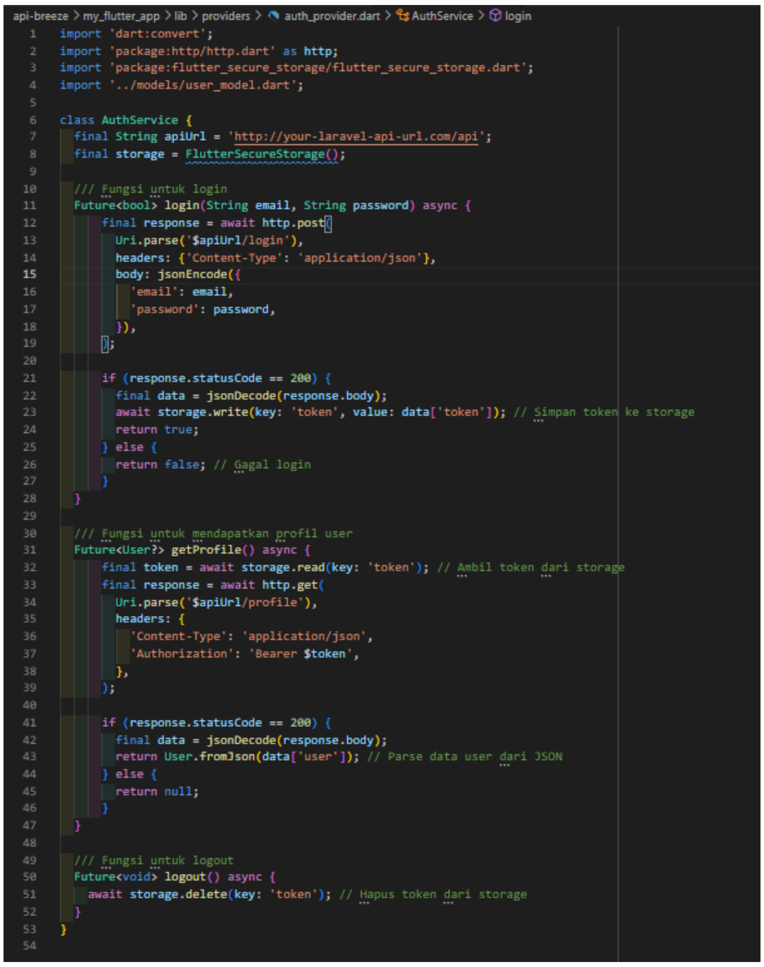
1 Buat file user_model.dart di lib/models/:


api-breeze > my_flutter_app > lib > models > user_model.dart > User > User.fromJson

```dart
class User {
  final int id;
  final String name;
  final String email;

  User({
    required this.id,
    required this.name,
    required this.email,
  });

  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      id: json['id'],
      name: json['name'],
      email: json['email'],
    );
  }
}
```

Langkah 4: Membuat Layanan API

1 Buat file auth_service.dart di lib/services/:

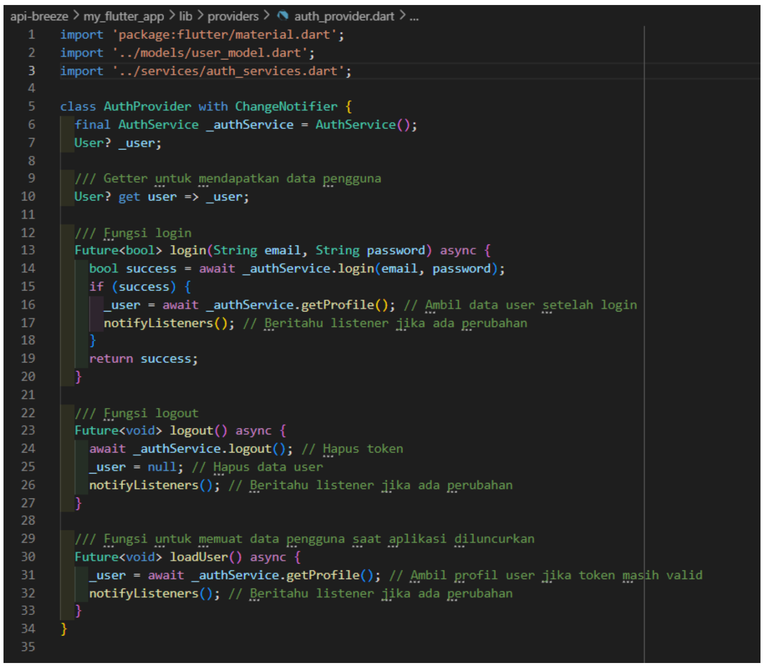


api-breeze > my_flutter_app > lib > providers > auth_provider.dart > AuthService > login

```dart
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:flutter_secure_storage/flutter_secure_storage.dart';
import '../models/user_model.dart';

class AuthService {
  final String apiUrl = 'http://your-laravel-api-url.com/api';
  final storage = FlutterSecureStorage();

  /// Fungsi untuk login
  Future<bool> login(String email, String password) async {
      final response = await http.post(
        Uri.parse('$apiUrl/login'),
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({
          'email': email,
          'password': password,
        }),
      );

      if (response.statusCode == 200) {
        final data = jsonDecode(response.body);
        await storage.write(key: 'token', value: data['token']); // Simpan token ke storage
        return true;
      } else {
        return false; // Gagal login
      }
  }

  /// Fungsi untuk mendapatkan profil user
  Future<User?> getProfile() async {
      final token = await storage.read(key: 'token'); // Ambil token dari storage
      final response = await http.get(
        Uri.parse('$apiUrl/profile'),
        headers: {
          'Content-Type': 'application/json',
          'Authorization': 'Bearer $token',
        },
      );

      if (response.statusCode == 200) {
        final data = jsonDecode(response.body);
        return User.fromJson(data['user']); // Parse data user dari JSON
      } else {
        return null;
      }
  }

  /// Fungsi untuk logout
  Future<void> logout() async {
    await storage.delete(key: 'token'); // Hapus token dari storage
  }
}
```

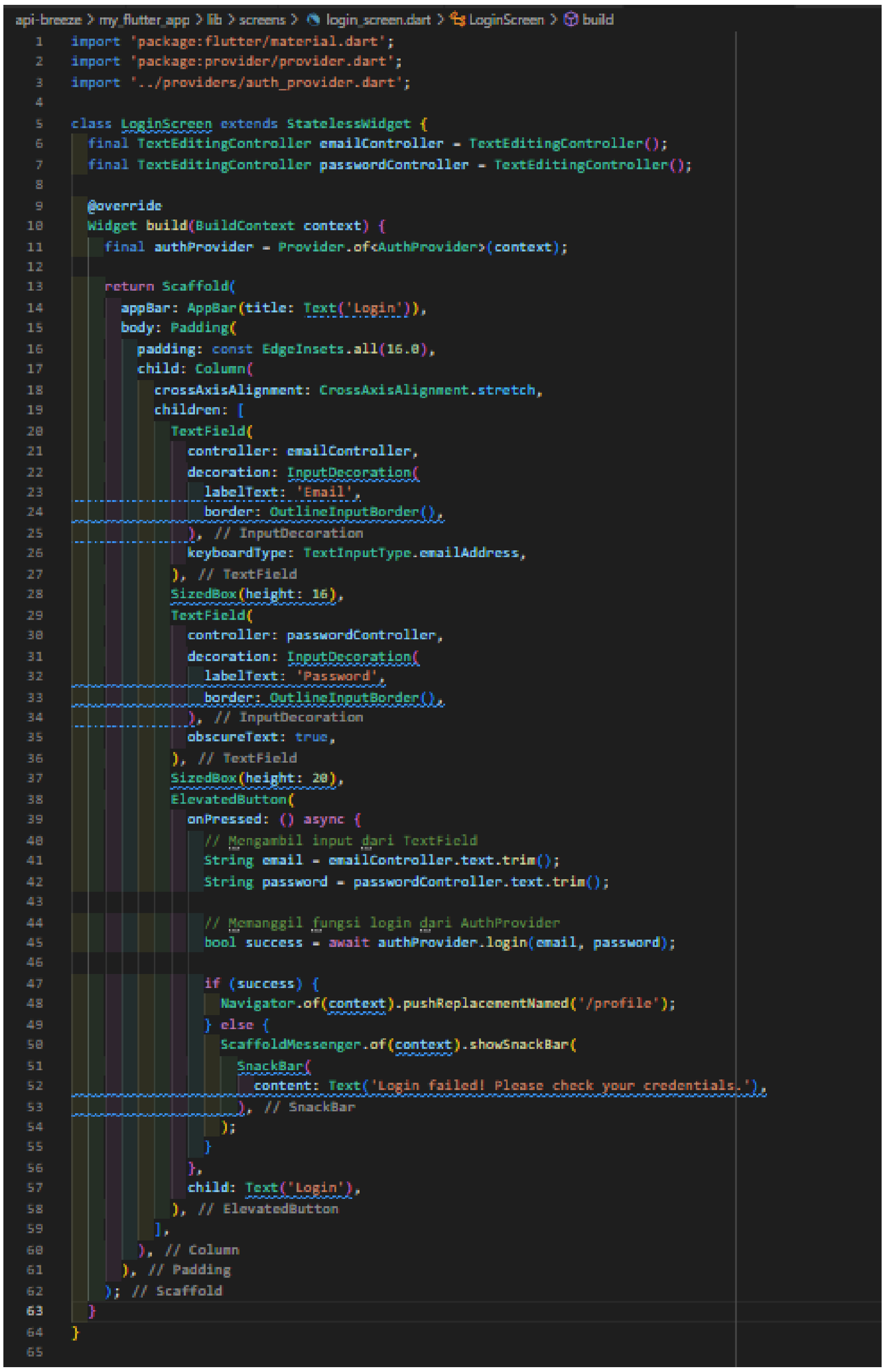
Langkah 5: Menyusun State Management dengan Provider

1 Buat file auth_provider.dart di lib/providers/:



api-breeze > my_flutter_app > lib > providers > auth_provider.dart > ...

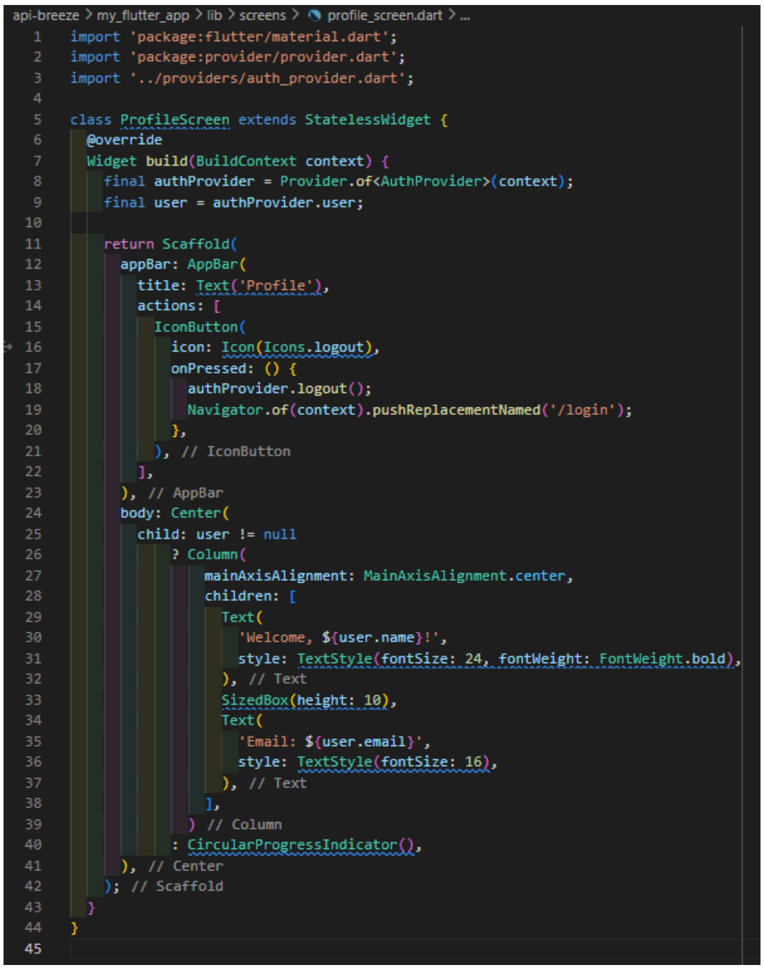
```dart
import 'package:flutter/material.dart';
import '../models/user_model.dart';
import '../services/auth_services.dart';

class AuthProvider with ChangeNotifier {
  final AuthService _authService = AuthService();
  User? _user;

  /// Getter untuk mendapatkan data pengguna
  User? get user => _user;

  /// Fungsi login
  Future<bool> login(String email, String password) async {
    bool success = await _authService.login(email, password);
    if (success) {
      _user = await _authService.getProfile(); // Ambil data user setelah login
      notifyListeners(); // Beritahu listener jika ada perubahan
    }
    return success;
  }

  /// Fungsi logout
  Future<void> logout() async {
    await _authService.logout(); // Hapus token
    _user = null; // Hapus data user
    notifyListeners(); // Beritahu listener jika ada perubahan
  }

  /// Fungsi untuk memuat data pengguna saat aplikasi diluncurkan
  Future<void> loadUser() async {
    _user = await _authService.getProfile(); // Ambil profil user jika token masih valid
    notifyListeners(); // Beritahu listener jika ada perubahan
  }
}
```

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

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/auth_provider.dart';

class LoginScreen extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    final authProvider = Provider.of<AuthProvider>(context);

    return Scaffold(
      appBar: AppBar(title: Text('Login')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextField(
              controller: emailController,
              decoration: InputDecoration(
                labelText: 'Email',
                border: OutlineInputBorder(),
              ), // InputDecoration
              keyboardType: TextInputType.emailAddress,
            ), // TextField
            SizedBox(height: 16),
            TextField(
              controller: passwordController,
              decoration: InputDecoration(
                labelText: 'Password',
                border: OutlineInputBorder(),
              ), // InputDecoration
              obscureText: true,
            ), // TextField
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () async {
                // Mengambil input dari TextField
                String email = emailController.text.trim();
                String password = passwordController.text.trim();

                // Memanggil fungsi login dari AuthProvider
                bool success = await authProvider.login(email, password);

                if (success) {
                  Navigator.of(context).pushReplacementNamed('/profile');
                } else {
                  ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(
                      content: Text('Login failed! Please check your credentials.'),
                    ), // SnackBar
                  );
                }
              },
              child: Text('Login'),
            ), // ElevatedButton
          ],
        ), // Column
      ), // Padding
    ); // Scaffold
  }
}
```

Langkah 7: Membuat Halaman Profil	
1	Buat file profile_screen.dart di lib/screens/:

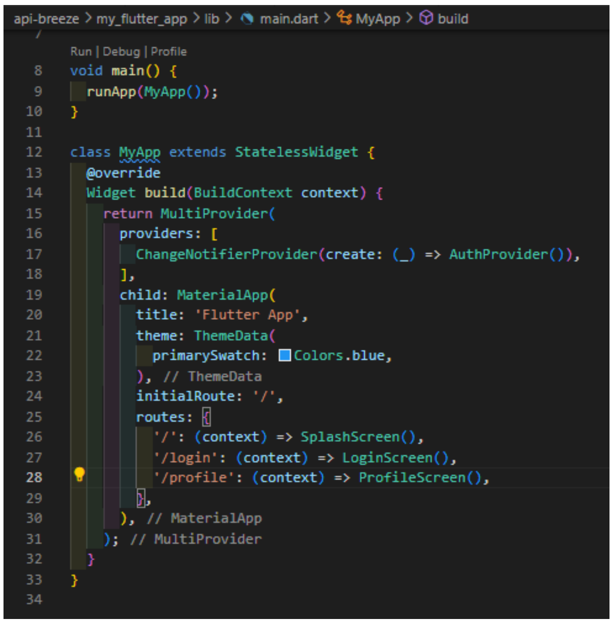


api-breeze > my_flutter_app > lib > screens > profile_screen.dart > ...

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/auth_provider.dart';

class ProfileScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final authProvider = Provider.of<AuthProvider>(context);
    final user = authProvider.user;

    return Scaffold(
      appBar: AppBar(
        title: Text('Profile'),
        actions: [
          IconButton(
            icon: Icon(Icons.logout),
            onPressed: () {
              authProvider.logout();
              Navigator.of(context).pushReplacementNamed('/login');
            },
          ), // IconButton
        ],
      ), // AppBar
      body: Center(
        child: user != null
            ? Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text(
                    'Welcome, ${user.name}!',
                    style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
                  ), // Text
                  SizedBox(height: 10),
                  Text(
                    'Email: ${user.email}',
                    style: TextStyle(fontSize: 16),
                  ), // Text
                ],
              ) // Column
            : CircularProgressIndicator(),
      ), // Center
    ); // Scaffold
  }
}

```

Langkah 8: Mengatur Routing dan Provider

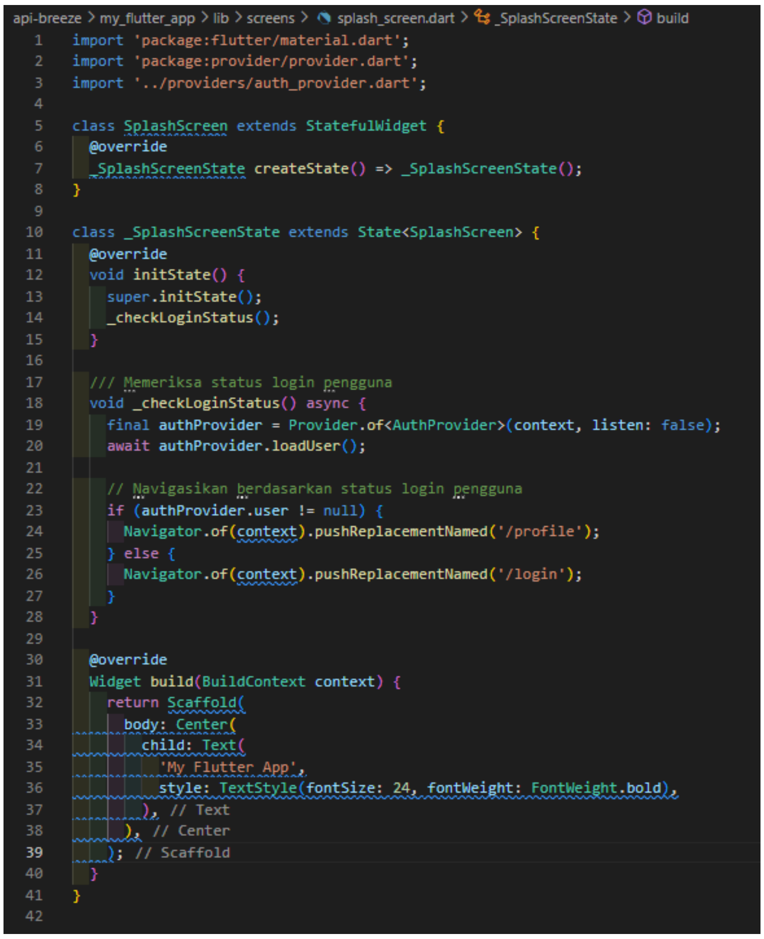
1 Buka main.dart dan atur routing serta Provider:



api-breeze > my_flutter_app > lib > main.dart > MyApp > build

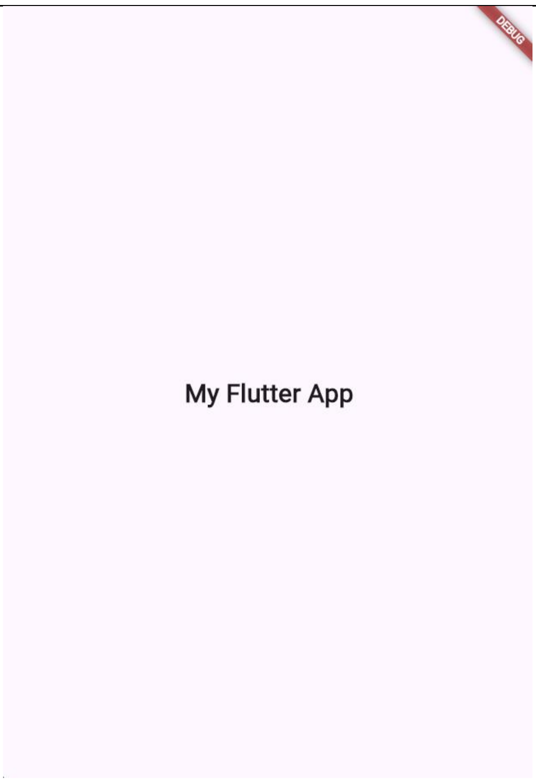
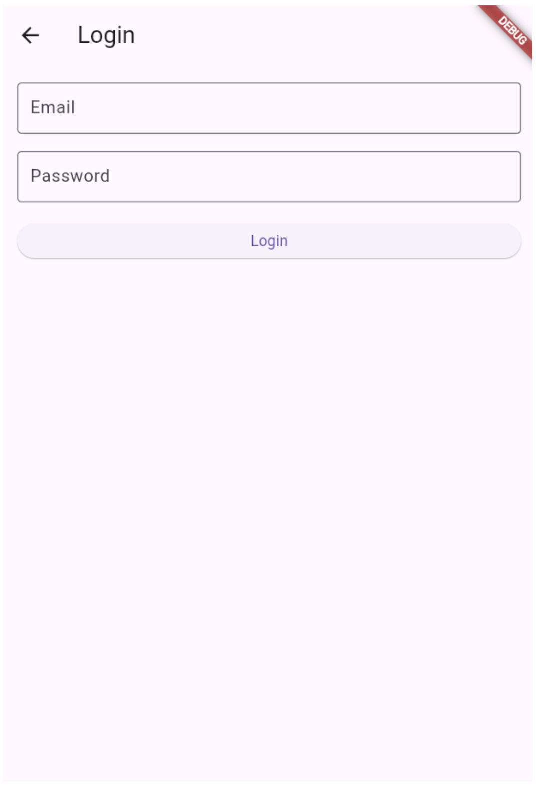
```dart
Run | Debug | Profile
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => AuthProvider()),
      ],
      child: MaterialApp(
        title: 'Flutter App',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ), // ThemeData
        initialRoute: '/',
        routes: {
          '/': (context) => SplashScreen(),
          '/login': (context) => LoginScreen(),
          '/profile': (context) => ProfileScreen(),
        },
      ), // MaterialApp
    ); // MultiProvider
  }
}

```

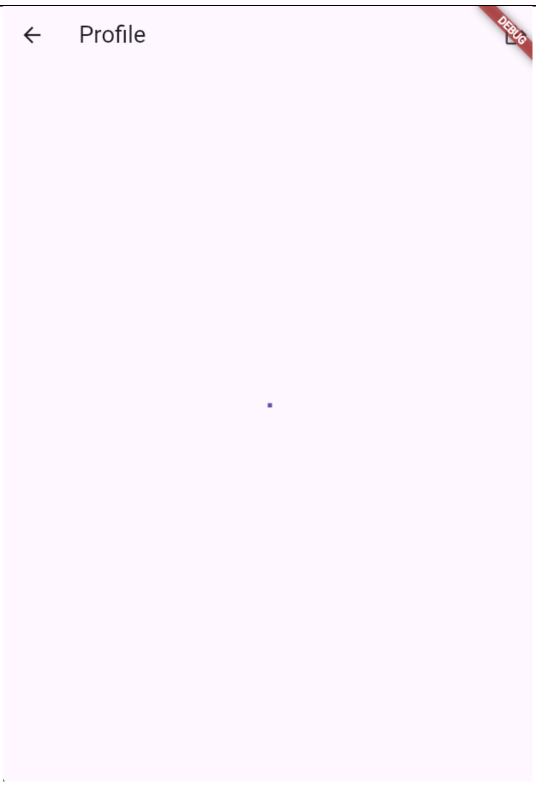
Langkah 9: Menyiapkan Splash Screen

1	Buat file splash_screen.dart di lib/screens/: 

```
api-breeze > my_flutter_app > lib > screens > splash_screen.dart > _SplashScreenState > build
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/auth_provider.dart';

class SplashScreen extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    _checkLoginStatus();
  }

  /// Memeriksa status login pengguna
  void _checkLoginStatus() async {
    final authProvider = Provider.of<AuthProvider>(context, listen: false);
    await authProvider.loadUser();

    // Navigasikan berdasarkan status login pengguna
    if (authProvider.user != null) {
      Navigator.of(context).pushReplacementNamed('/profile');
    } else {
      Navigator.of(context).pushReplacementNamed('/login');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Text(
          'My Flutter App',
          style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
        ), // Text
      ), // Center
    ); // Scaffold
  }
}
```

Langkah 10: Menjalankan Aplikasi	
1	Pastikan API Laravel Anda sudah berjalan dan endpoint login serta profil sudah tersedia. Jalankan aplikasi Flutter: • Page Splash

	 • Page Login  • Page Profile

		ProfileDEBUG