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

```php
<?php

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

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

```

Step 5: Update Controllers

.env | RegisteredUserController.php | AuthenticatedSessionController.php | Extension:

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

```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
        ]);
    }
}
```

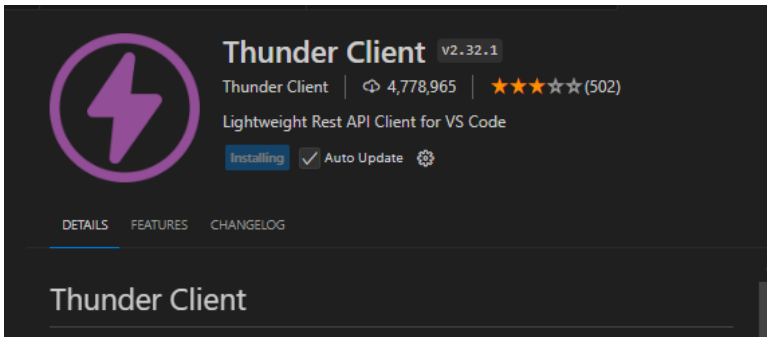
.env | RegisteredUserController.php | AuthenticatedSessionController.php | Extension: Thund

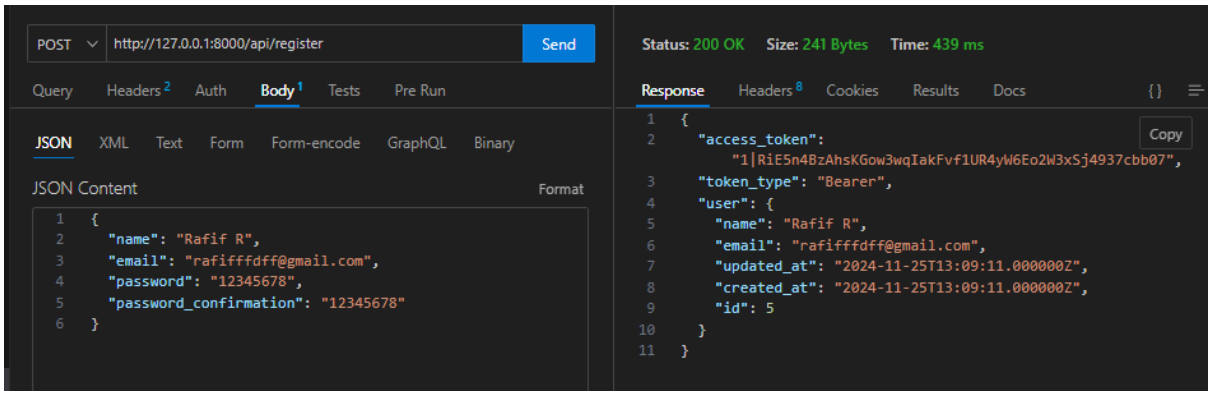
api-breeze > app > Http > Controllers > Auth > AuthenticatedSessionController.php > ...

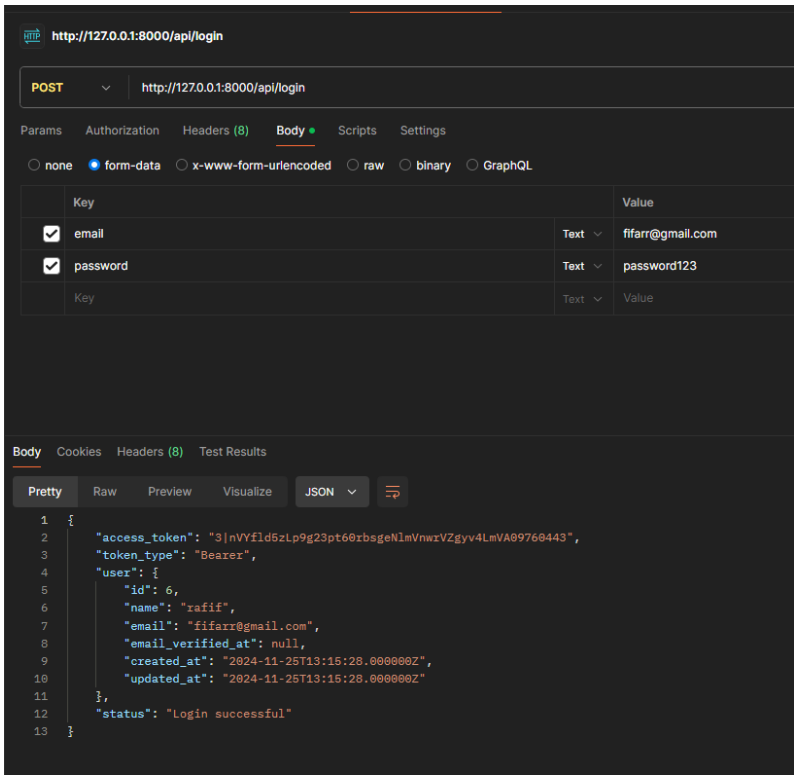
```php
<?php

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']);
    }
}
```

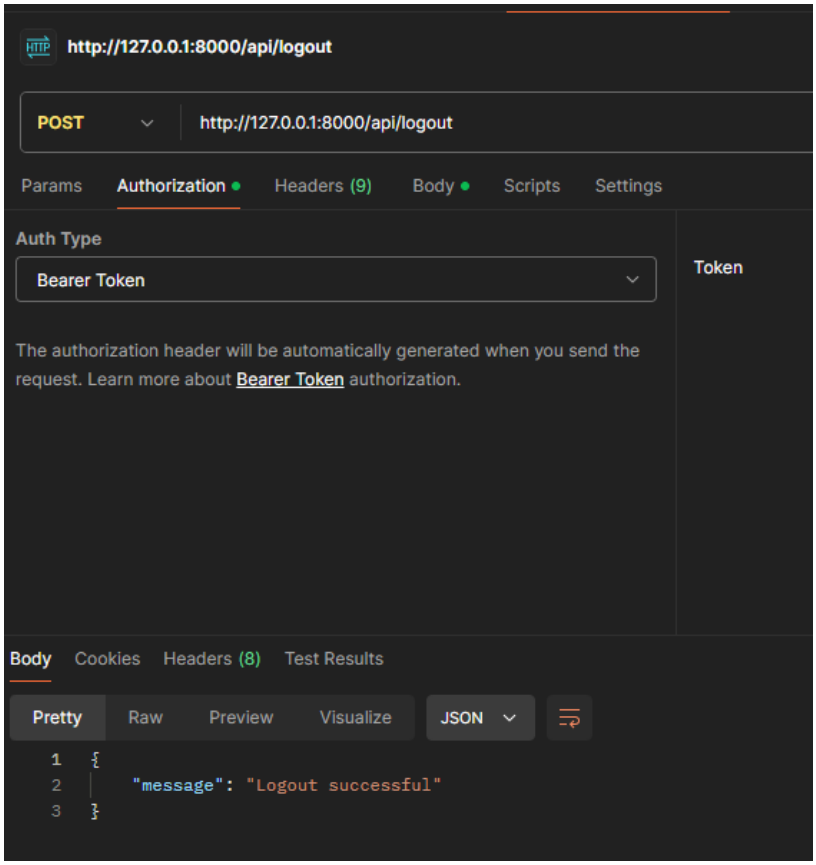
```
PS C:\laragon\www\api-breeze> php artisan serve
  forking is not supported on this platform

  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,

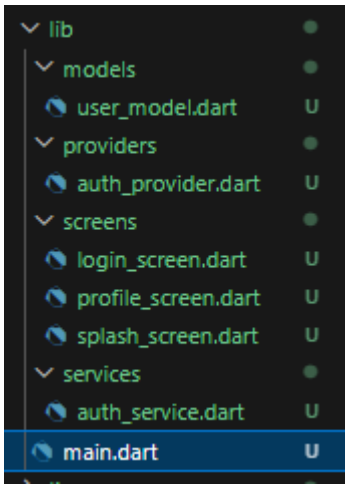
Langkah 1: Persiapan Proyek Flutter

```
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/
```

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

```
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 > ...

```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

Langkah 5: Menyusun State Management dengan Provider

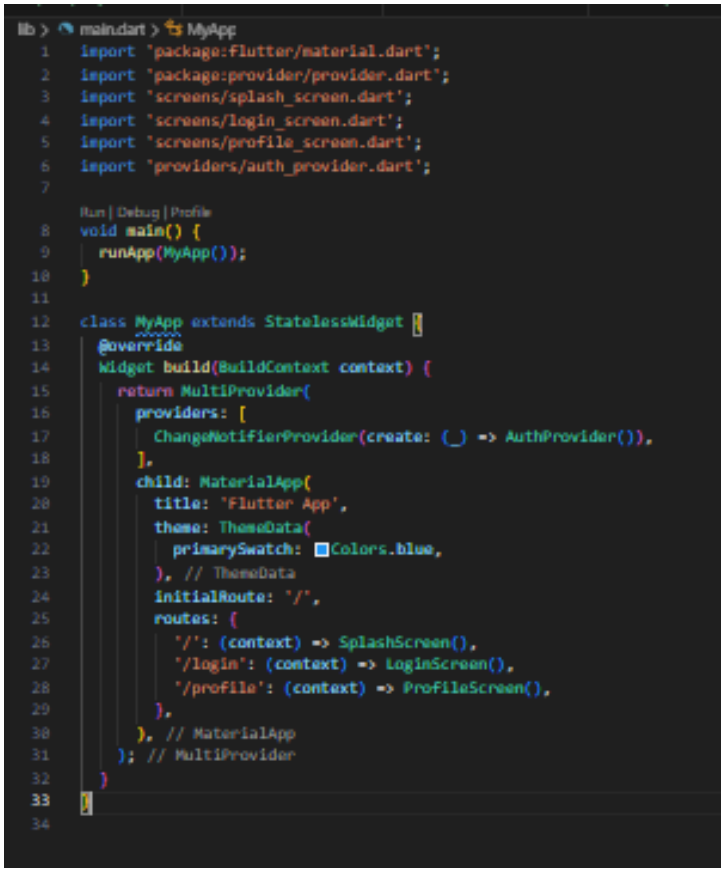
Langkah 6: Membuat Halaman Login

```
lib > screens > login_screen.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: EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: emailController,
              decoration: InputDecoration(labelText: 'Email'),
            ), // TextField
            TextField(
              controller: passwordController,
              decoration: InputDecoration(labelText: 'Password'),
              obscureText: true,
            ), // TextField
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () async {
                bool success = await authProvider.login(
                  emailController.text,
                  passwordController.text,
                );
                if (success) {
                  Navigator.of(context).pushReplacementNamed('/profile');
                } else {
                  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                    content: Text('Login failed!'),
                  )); // SnackBar
                }
              },
              child: Text('Login'),
            ), // ElevatedButton
          ],
        ), // Column
      ), // Padding
    ); // Scaffold
  }
}
```

Langkah 7: Membuat Halaman Profil

```
lib > screens > profile_screen.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}!'),
                  Text('Email: ${user.email}'),
                ],
              ) // Column
            : CircularProgressIndicator(),
      ), // Center
    ); // Scaffold
  }
}
```

Langkah 8: Mengatur Routing dan Provider



```dart
lib > main.dart > MyApp
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'screens/splash_screen.dart';
import 'screens/login_screen.dart';
import 'screens/profile_screen.dart';
import 'providers/auth_provider.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

```dart
lib > screens > splash_screen.dart > _SplashScreenState > _checkLoginStatus
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();
  }

  void _checkLoginStatus() async {
    final authProvider = Provider.of<AuthProvider>(context, listen: false);
    await authProvider.loadUser();
    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)),
      ), // Center
    ); // Scaffold
  }
}
```