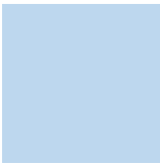


MODUL

PEMROGRAMAN BERBASIS PLATFORM



PERTEMUAN – 5

BloC

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA



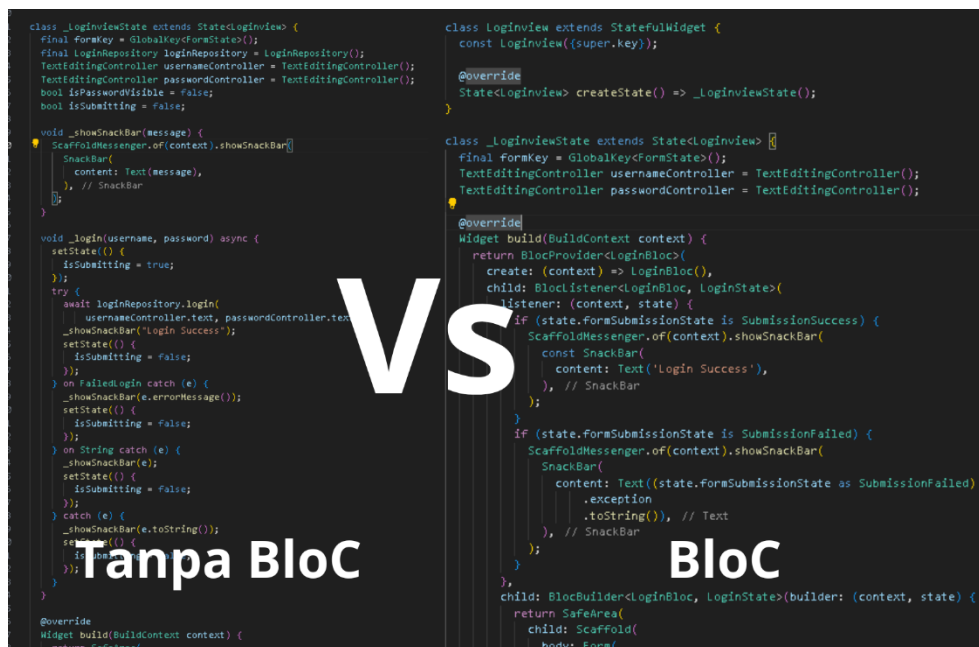
TUJUAN

Setelah menyelesaikan modul ini, praktikan diharapkan mampu :

1. Memahami arsitektur pattern Business Logic Component (BloC)
2. Mampu menerapkan arsitektur pattern BloC dalam proyek Flutter.

DASAR TEORI

A. BloC



Gambar 1. Code tanpa menerapkan BloC di sisi kiri, code menerapkan BloC di sisi kanan

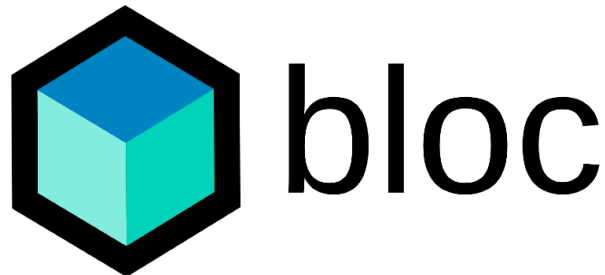
Pada mata kuliah Pemrograman Berbasis Objek, teman-teman telah mempelajari salah satu arsitektur pattern yaitu MVC (Model-View-Controller). **Dengan menerapkan arsitektur pattern, kita mendapatkan source code yang lebih rapi, mudah dikembangkan, mudah di testing, dan efisien.** Salah satu arsitektur pattern yang populer di Flutter adalah BloC (Business Logic Component).

Dapat dilihat pada Gambar 1, code yang tidak menerapkan BloC akan memiliki berbagai state dan business logic di atas function Build-nya. Sedangkan code yang menerapkan BloC hanya mengatur mengenai UI pada widgetnya.



22 BloC membagi software menjadi 3 lapisan, yaitu

- 23 1. Layer Presentation
- 24 2. Layer Business Logic
- 25 3. Layer Data



26

27 Gambar 2. Logo BloC

28 **B. Layer Presentation**

29 Layer ini merupakan lapisan tampilan yang berinteraksi dengan
30 pengguna. Layer ini berisi widget-widget User Interface yang
31 menerima inputan dan meneruskannya ke layer Business Logic dan
32 menampilkan widget-widget serta propertinya berdasarkan layer
33 business logic.

34 **C. Layer Business Logic**

35 Layer ini merupakan lapisan dimana BloC bekerja. Lapisan ini
36 menghubungkan antara layer presentation dengan layer data. Ada
37 tiga komponen penting di layer ini, yaitu

38 **1. State**

39 Di sini kita mendefinisikan state / kondisi dari widget-widget di
40 layer presentation atau kondisi dari aplikasi kita. Widget-widget
41 atau property dari widget yang ditampilkan ke user diatur
42 berdasarkan dari state ini.

43 **2. Event**

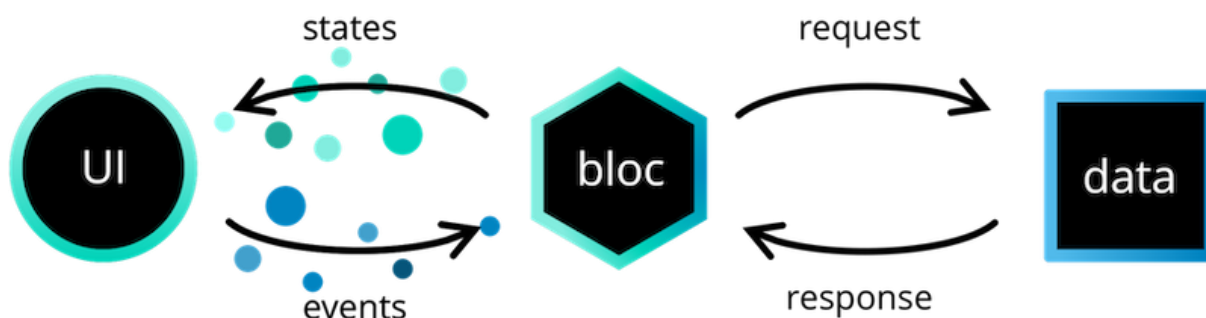
44 Aksi yang dipicu oleh layer presentation karena tindakan user
45 atau aplikasi.

46 **3. BloC**

47 Mapping/ penghubung relasi antara event dengan state. Di sini
48 kita memasang state dengan event.

D. Layer Data

Lapisan ini merupakan penghubung aplikasi dengan database atau sumber data untuk aplikasi. Lapisan ini meminta dan menerima data mentah dari dalam atau luar aplikasi, kemudian mengelola sebelumnya diberikan ke layer Business Logic.



Gambar 3. Arsitektur BloC

E. Koneksi Layer Presentation dan Business Logic Layer

Ada tiga keyword penting yang akan sering digunakan untuk menghubungkan presentation layer dengan business logic layer:

1. BlocProvider

BlocProvider digunakan untuk menyediakan sebuah objek Bloc untuk dapat diakses oleh seluruh childnya. Disini kita mengelola lifecycle dari Bloc. Pastikan telah memanggil BlocProvider sebagai parent dari segala widget disebuah Page untuk dapat mengakses Bloc.

2. BlocListener

BlocListener berfungsi untuk mendengarkan perubahan state dari sebuah Bloc. BlocListener berguna untuk menampilkan suatu efek terhadap perubahan state tertentu tanpa mempengaruhi UI secara keseluruhan, seperti menampilkan ToastBar.

3. BlocBuilder

BlocBuilder berfungsi untuk membangun ulang berdasarkan perubahan state dari sebuah Bloc. Cara kerjanya mirip seperti StatefulWidget yang berubah sesuai dengan perubahan statenya.



75 Harus diperhatikan bahwa BlocListener dan BlocBuilder harus
76 digunakan setelah menggunakan BlocProvider di parent-nya.
77 Terdapat keyword lainnya yang dapat membantu optimasi aplikasi
78 lebih efisien, tetapi tidak dijelaskan di modul ini seperti BlocSelector,
79 BlocConsumer, dan RepositoryProvider. Jika tertarik untuk
80 mempelajari BloC lebih dalam, silahkan membaca lebih lanjut di
81 dokumentasi yang disediakan oleh komunitas BloC [disini](#).

82

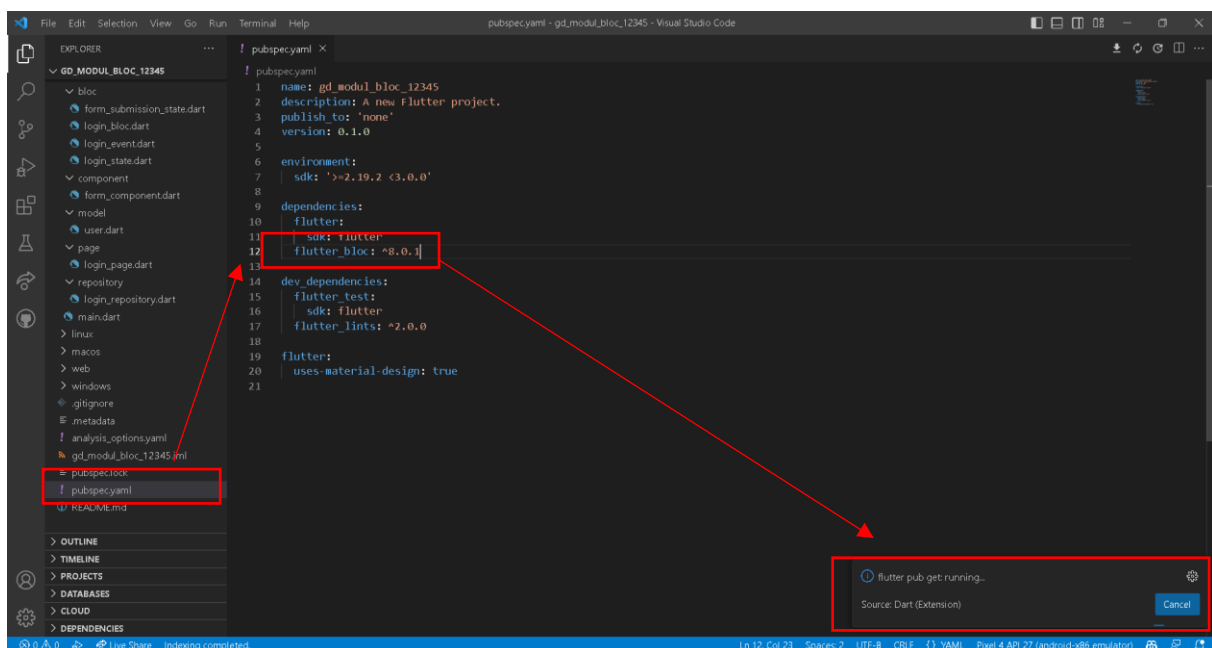
GUIDED 1 – BLOC PATTERN

Poin yang dipelajari dalam Guided 1 ini, yaitu :

1. Memahami struktur folder dan file dalam arsitektur pattern BloC
2. Memahami penerapan BloC dalam logika bisnis login aplikasi

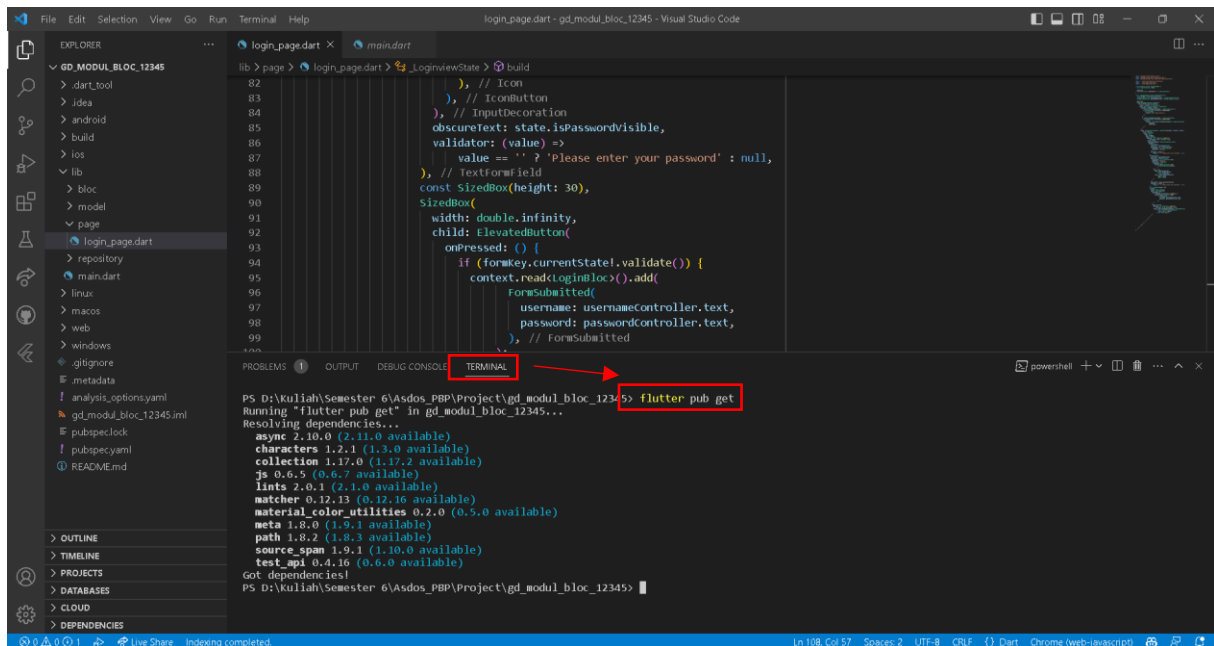
Pada guided 1 ini, kita akan mencoba membuat halaman login dengan menerapkan BloC. Silahkan ikuti langkah – langkah berikut ini :

1. Buat project Flutter baru dan beri nama 'gd_modul_bloc_XXXX' (XXXX: empat digit npm terakhir).
2. Untuk dapat menerapkan BloC pattern, kita perlu menambahkan **dependencies flutter_bloc** seperti di bawah. Perhatikan spasi dan tab di **pubspec.yaml**, perbedaan spasi dapat menyebabkan error. Setelah di save seharusnya flutter pub get otomatis dijalankan.



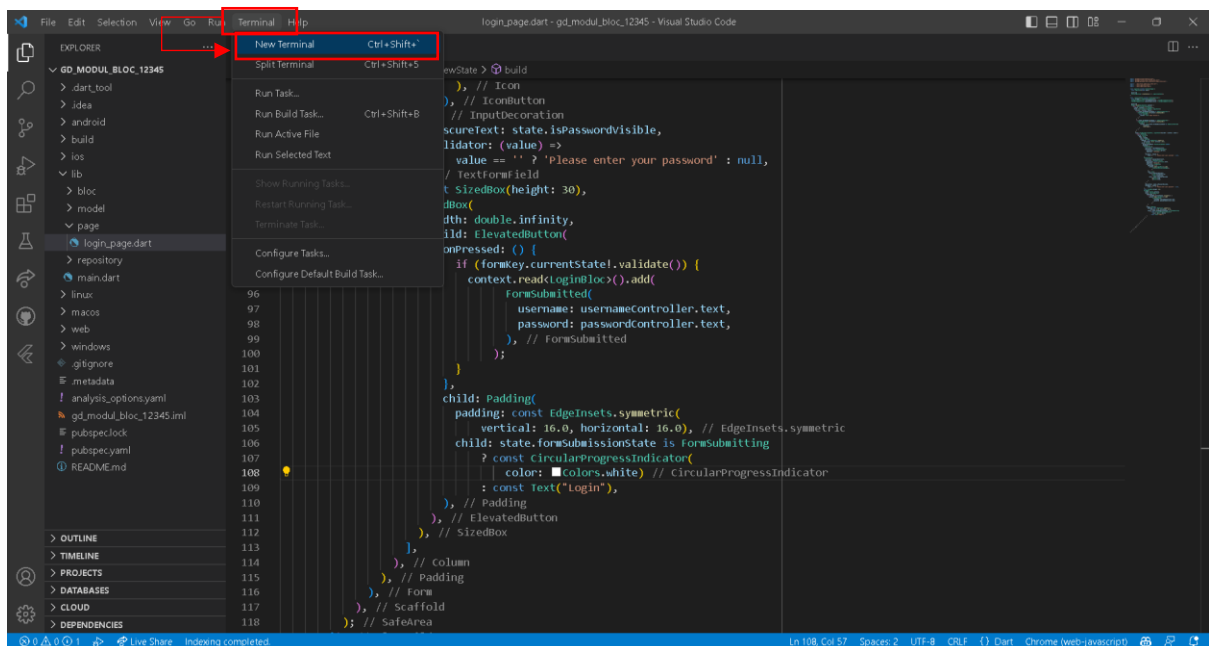
Gambar 4. Menambahkan dependencies flutter_bloc

3. Jika Flutter pub get tidak berjalan secara otomatis, silahkan mengetik command **flutter pub get** di console.



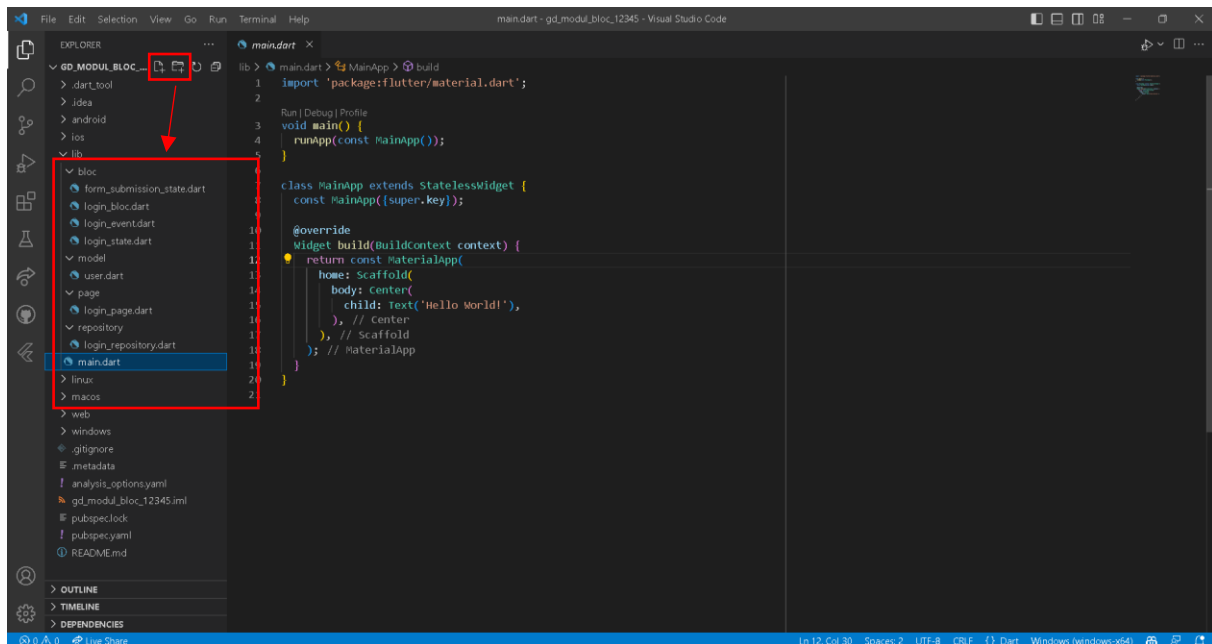
Gambar 5. Menjalankan command flutter pub get di Terminal

4. Bagi yang tidak melihat terminalnya silahkan pergi ke tab **Terminal** dan pilih **New Terminal**. Kemudian kembali ke Langkah 3.



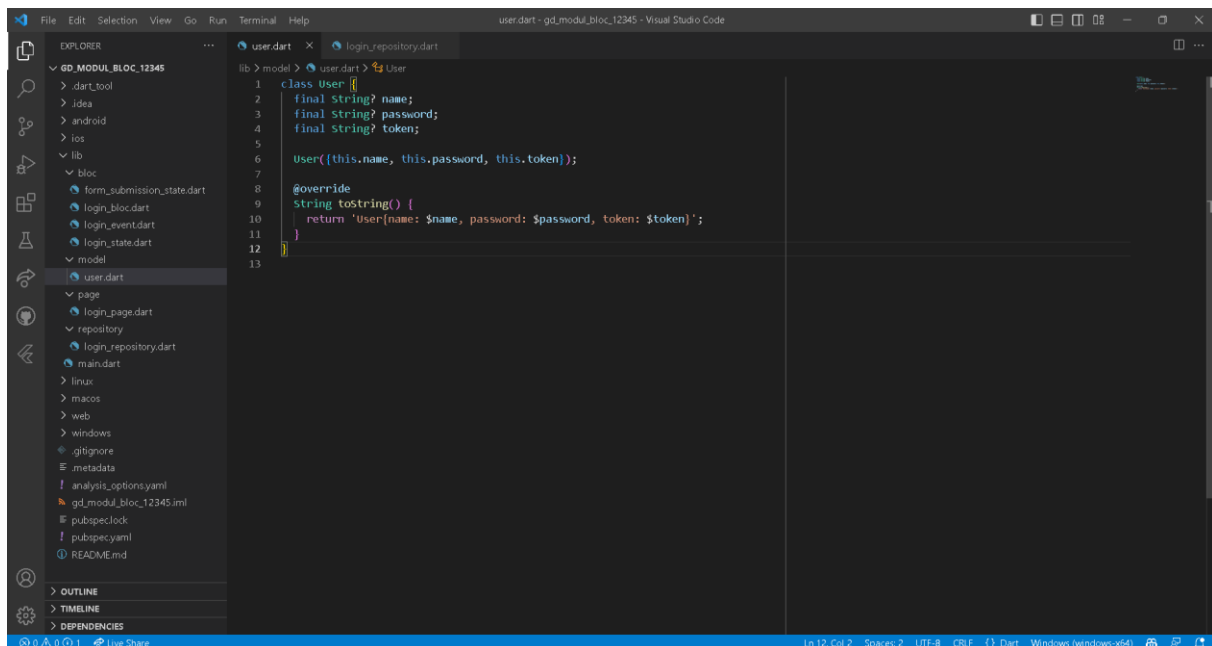
Gambar 6. Cara membuka terminal

5. Buatlah folder dan file kosong seperti di gambar. Perlu diingat format penamaan file harus berupa **Snake Case**.

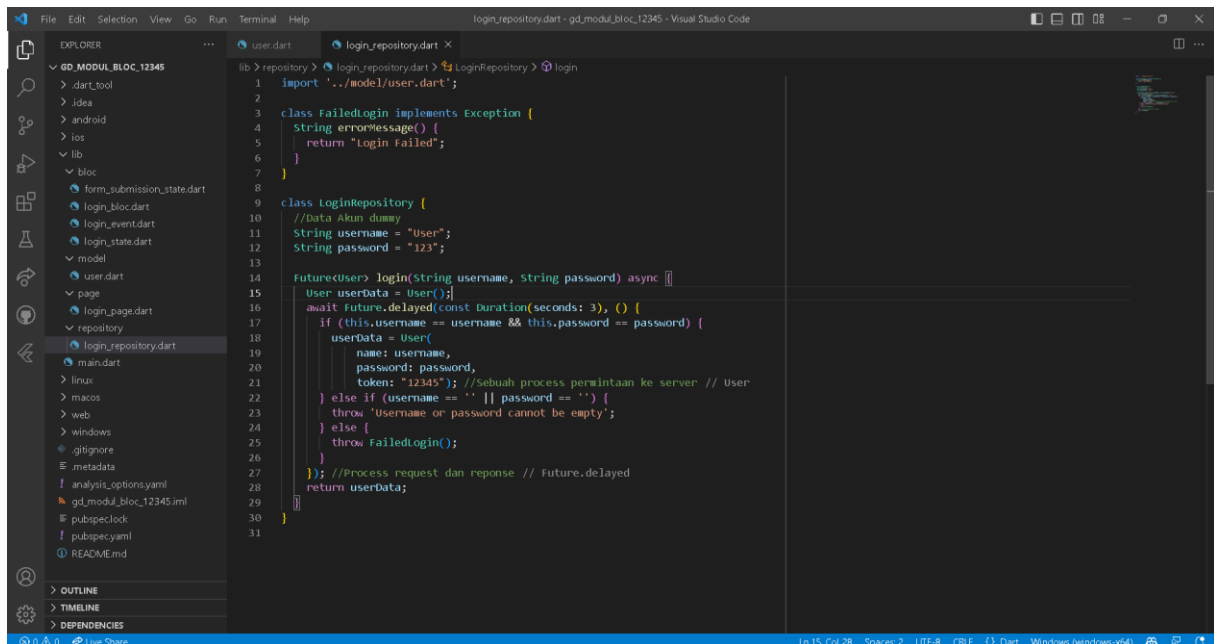


Gambar 7. Struktur file dan folder Guided 1

6. Kita akan mendaur ulang code dari modul sebelumnya, yaitu class **User**, **FailedLogin**, dan **LoginRepository**. Silahkan meng-copy code dari guided sebelumnya sesuai dengan nama filenya. Jika mengalami error di file login_repository.dart, jangan lupa untuk mencoba menggunakan **ctrl + .** untuk melihat saran solusi untuk mengatasi error tersebut. Dalam kasus ini, jangan lupa untuk meng-import **class User** dari file **user.dart**.

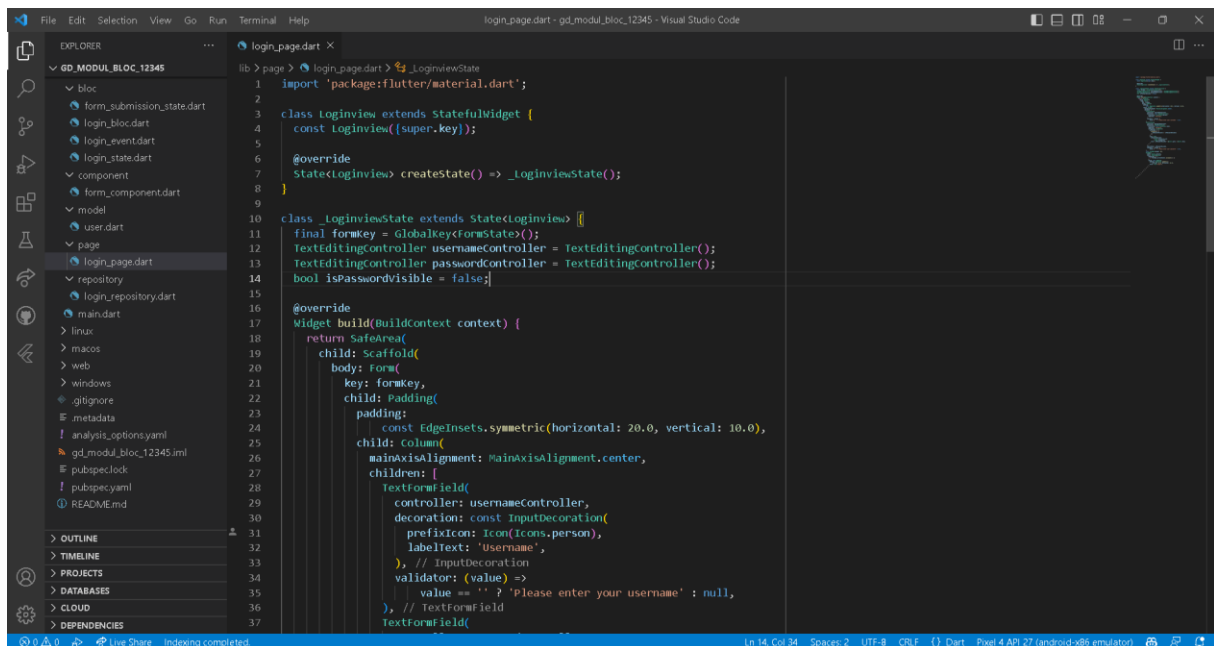


Gambar 8. Class User

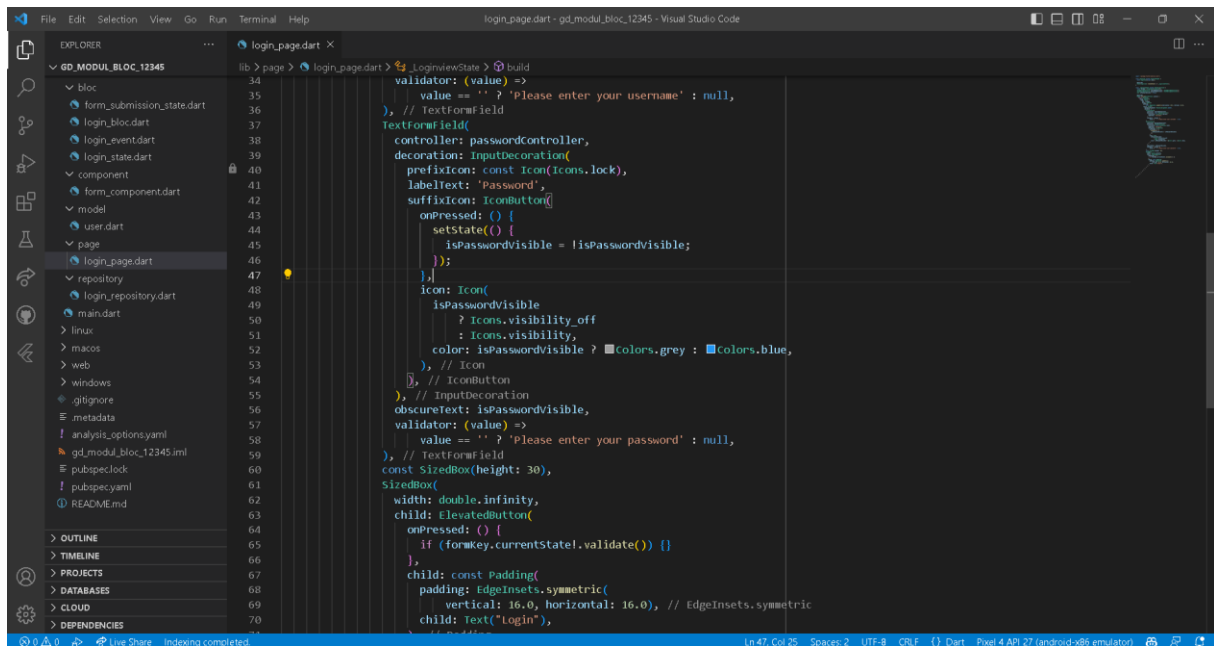


Gambar 9. Class LoginRepository

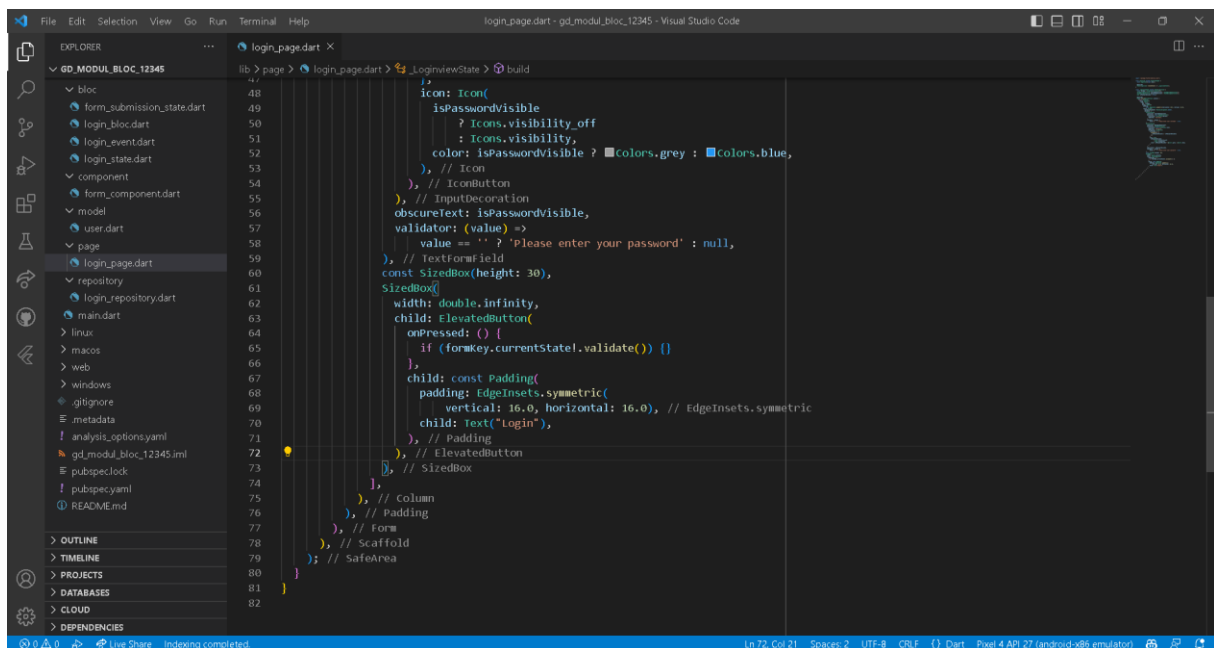
7. Sekarang mari membuat tampilan Login Page-nya terlebih dahulu. Disini kita akan membuat LoginPage biasa tanpa mengaplikasikan BloC pattern terlebih dahulu.



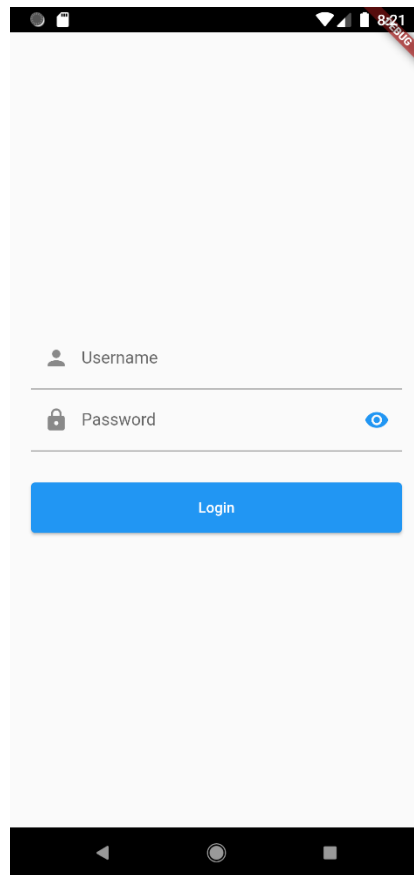
Gambar 10. LoginPage bagian 1



Gambar 11. LoginPage bagian 2

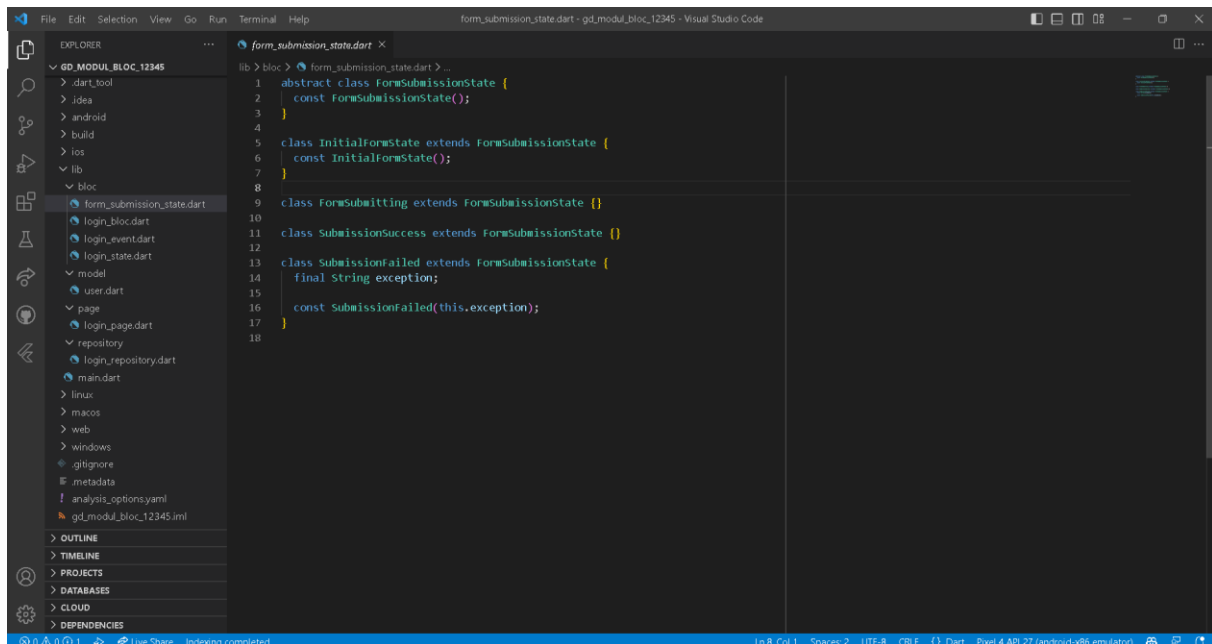


Gambar 12. LoginPage bagian 3



Gambar 13. Tampilan LoginPage di android

8. Buatlah abstract class **FormSubmissionState** dan childnya untuk digunakan dalam state nanti.
 - a. InitialFormState
Kondisi awal form saat sedang dalam proses pengisian.
 - b. FormSubmitting
Kondisi saat aplikasi sedang mengirimkan form ke backend dan menunggu respon selesai. Saat kondisi ini, kita dapat menampilkan tampilan loading untuk memberitahukan kepada user bahwa ada proses yang sedang berjalan.
 - c. SubmissionSuccess
Kondisi saat pengiriman ke backend sukses.
 - d. SubmissionFailed
Kondisi saat pengiriman ke backend gagal. Kita dapat menerima exception handling yang di-throw oleh Repository.

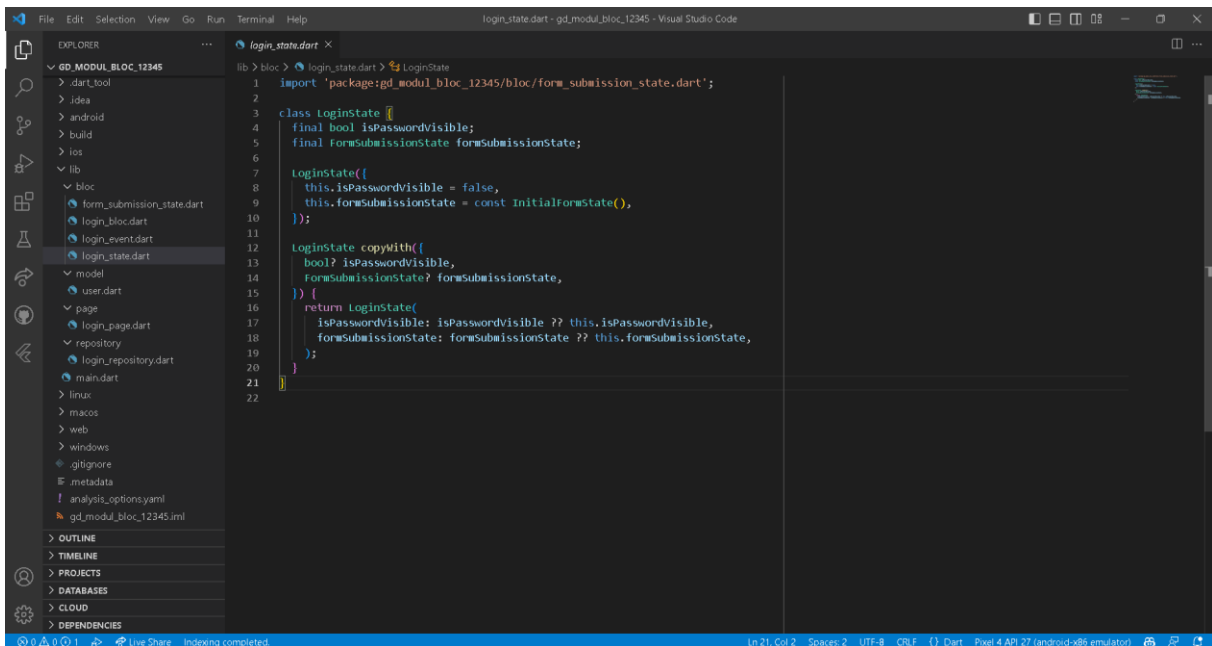


Gambar 14. Abstract class FormSubmissionState

9. Sekarang kita akan membuat class untuk BloC-nya. Pertama kita mulai dengan membuat class LoginState. LoginState berisi dengan variable atau atribut dari widget di LoginPage. Dalam hal ini kita akan memindahkan variable isVisible dari LoginPage ke LoginState. Kita juga membuat variable formSubmissionState dari class FormSubmissionState yang telah kita buat sebelumnya. Variable ini berfungsi untuk menyatakan kondisi progress saat kita menekan tombol Login.

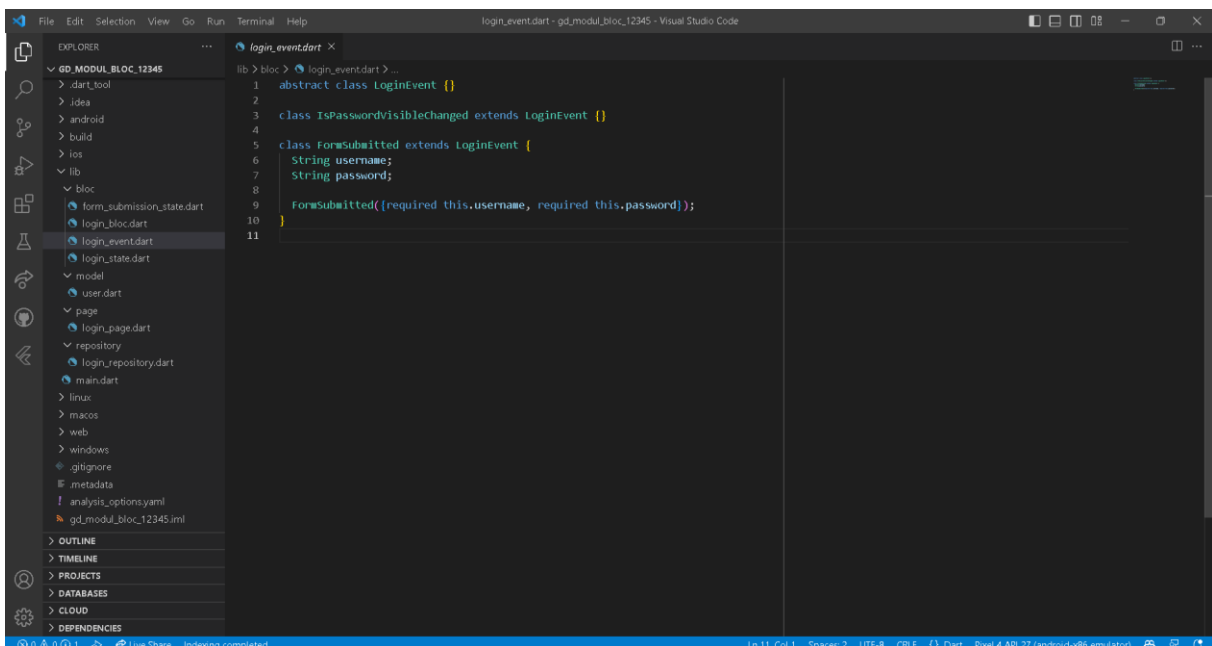
a. copyWith

Method copyWith berfungsi sebagai method set value atribut dari class. Perhatikan penerapan **nullable** pada parameter dan **null aware operator**.



Gambar 15. Class LoginState

10. Selanjutnya kita membuat class dari LoginEvent. Tidak seperti state, event terdiri dari beberapa class dari aksi yang mungkin terjadi di LoginPage, yaitu IsPasswordVisibleChanged (Ketika kita mengganti visibilitas dari field Password) dan FormSubmitted (Ketika kita menekan tombol login). Perlu diperhatikan bahwa format penamaan dari event adalah **past tense**, hal ini untuk mempermudah pemahaman bahwa event telah terjadi.



Gambar 16. Class LoginEvent

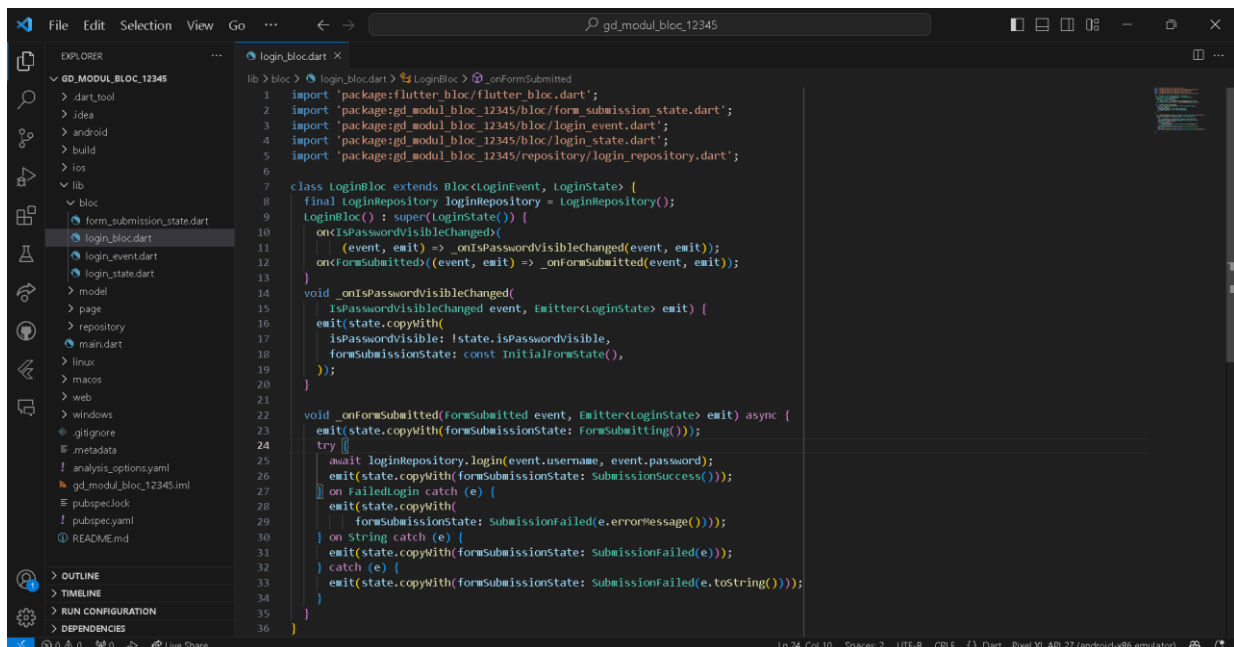
11. Yang terakhir, kita membuat class LoginBloc yang memetakan antara event dengan state. Perhatikan bahwa pemetaan dilakukan di konstruktor dari LoginBloc dengan method private di LoginBloc. Di setiap method kita mengelola state atau logic bisnis yang terjadi di aplikasi. Keyword yang perlu diperhatikan adalah

a. Emit

Emit merupakan objek dari class **Emitter**. Emitter berfungsi untuk memicu state baru dari aplikasi dan memberitahu segala widget yang memantau BloC. Salah penggunaan dari emit adalah pada baris 16 Gambar 17.

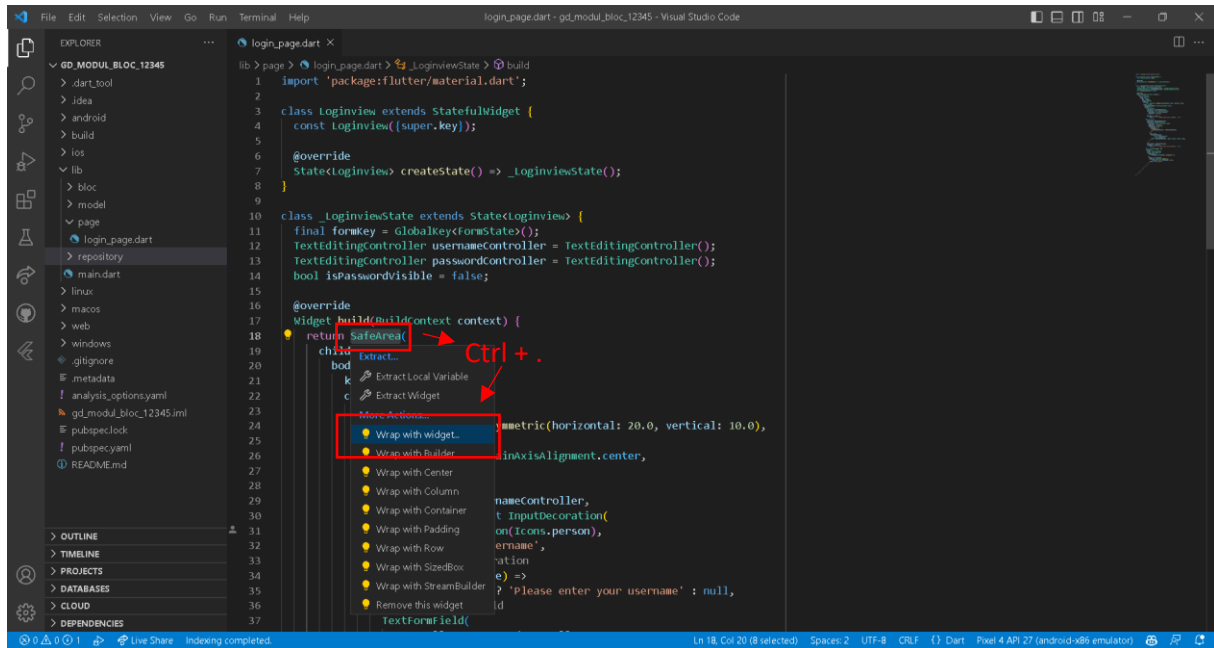
b. Event

Perhatikan Gambar 17, pada baris 25, kita memanggil atribut dari event yaitu username dan password. Darimanakah atribut ini? Perlu diperhatikan di method ini event merupakan object dari class **FormSubmitted** (lihat baris 22). Sehingga sesuai dengan definisi class FormSubmitted yang telah kita buat di atas, tentu event disini memiliki atribut username dan password.



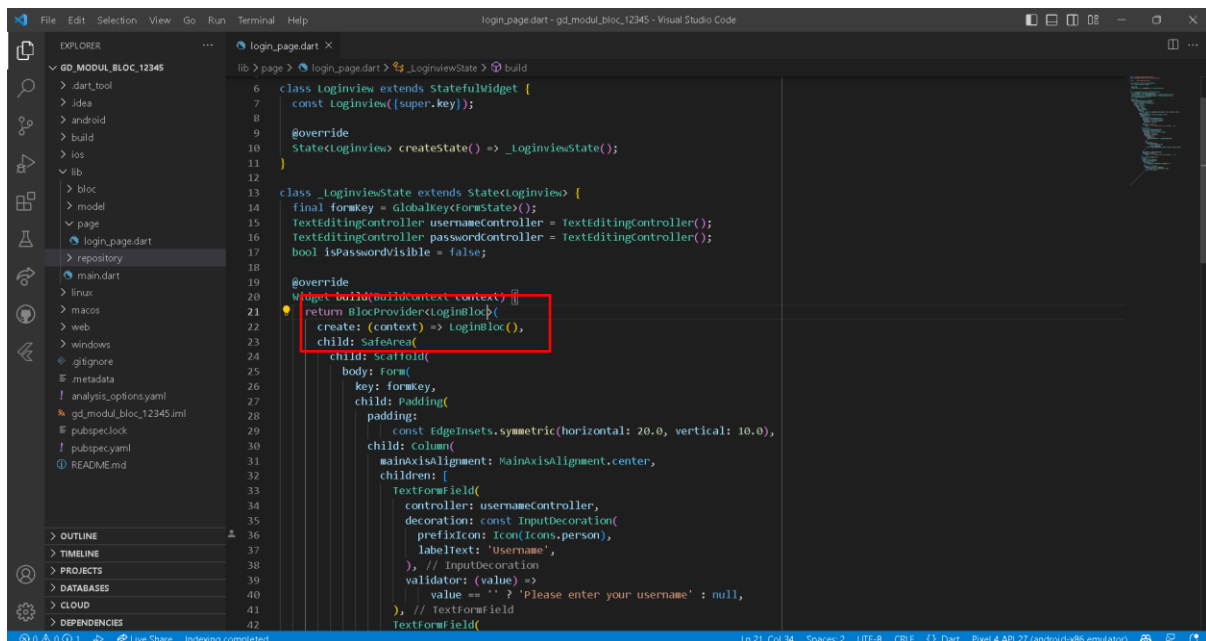
Gambar 17. Class LoginBloc

12. Selanjutnya kita akan menghubungkan BloC dengan LoginPage. Pertama kita perlu membungkus seluruh widget dengan **BlocProvider**. Caranya dengan mengklik widget teratas (Dalam hal ini SafeArea) dan klik shortcut **ctrl + titik (.)**, maka akan muncul menu. Pilihlah **Wrap with Widget**.



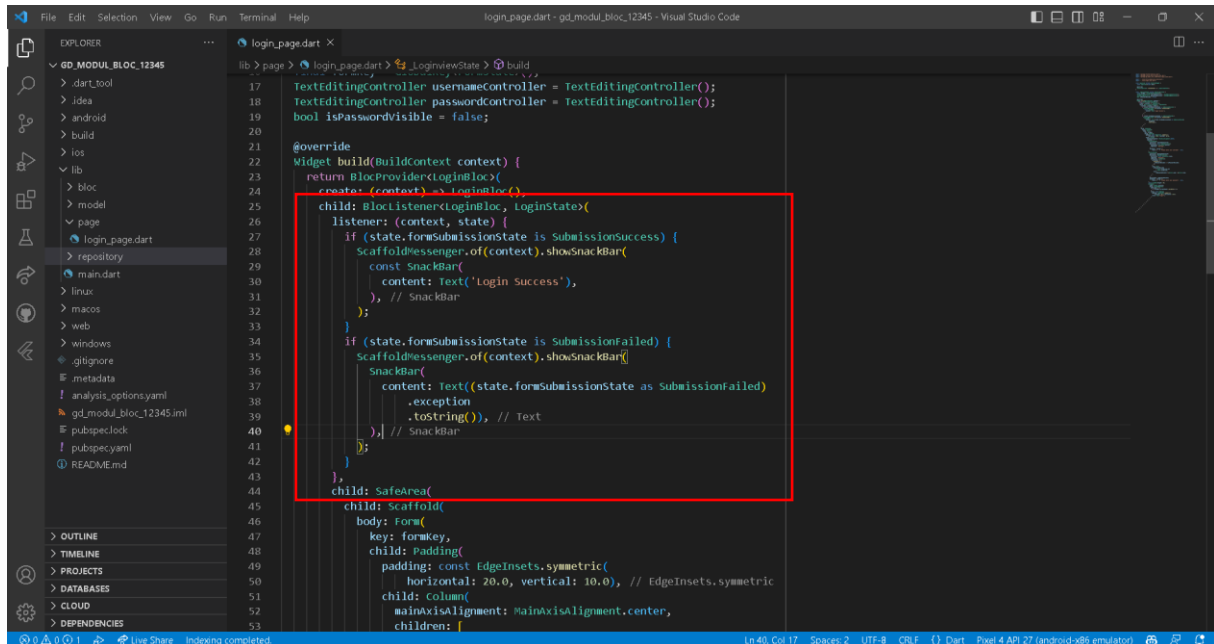
Gambar 18. Cara men-wrap widget baru

13. Silahkan ganti tulisan widget dengan **BlocProvider** seperti di bawah.



Gambar 19. BlocProvider

14. Dengan cara yang sama pada Langkah 12, tambahkan widget baru di atasnya dan ganti dengan **BlocListener** seperti di bawah. Di dalam BlocListener, kita me-listen state dari Bloc LoginBloc. Disini kita menggunakannya untuk menampilkan snackbar jika pengiriman form sukses atau gagal

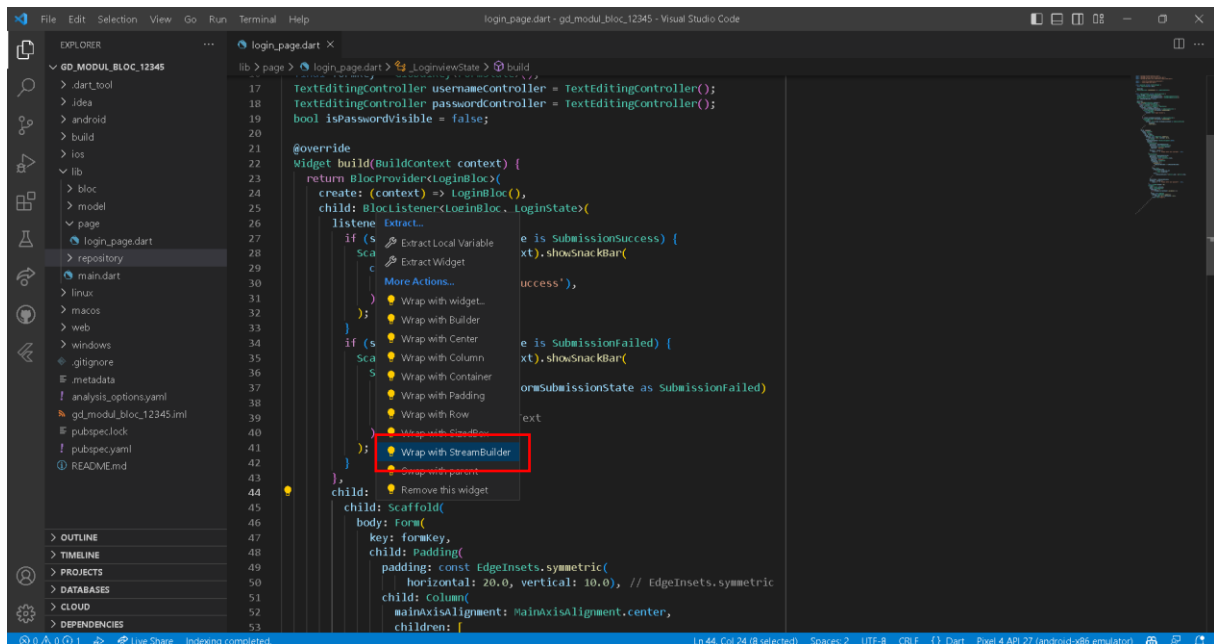


```
lib > page > login_page.dart > _loginViewState > build
login_page.dart
17 TextEditingController usernameController = TextEditingController();
18 TextEditingController passwordController = TextEditingController();
19 bool isPasswordVisible = false;
20
21 @override
22 Widget build(BuildContext context) {
23   return BlocProvider<LoginBloc>(
24     create: (context) => LoginBloc(),
25     child: BlocListener<LoginBloc, LoginState>(
26       listener: (context, state) {
27         if (state.formSubmissionState is SubmissionSuccess) {
28           ScaffoldMessenger.of(context).showSnackBar(
29             const SnackBar(
30               content: Text('Login Success'),
31             ), // SnackBar
32           );
33         }
34         if (state.formSubmissionState is SubmissionFailed) {
35           ScaffoldMessenger.of(context).showSnackBar(
36             SnackBar(
37               content: Text((state.formSubmissionState as SubmissionFailed)
38                 .exception
39                 .toString()), // Text
40             ), // SnackBar
41           );
42         }
43       },
44       child: SafeArea(
45         child: Scaffold(
46           body: Form(
47             key: formKey,
48             child: Padding(
49               padding: const EdgeInsets.symmetric(
50                 horizontal: 20.0, vertical: 10.0), // EdgeInsets.symmetric
51               child: column(
52                 mainAxisAlignment: MainAxisAlignment.center,
53                 children: [

```

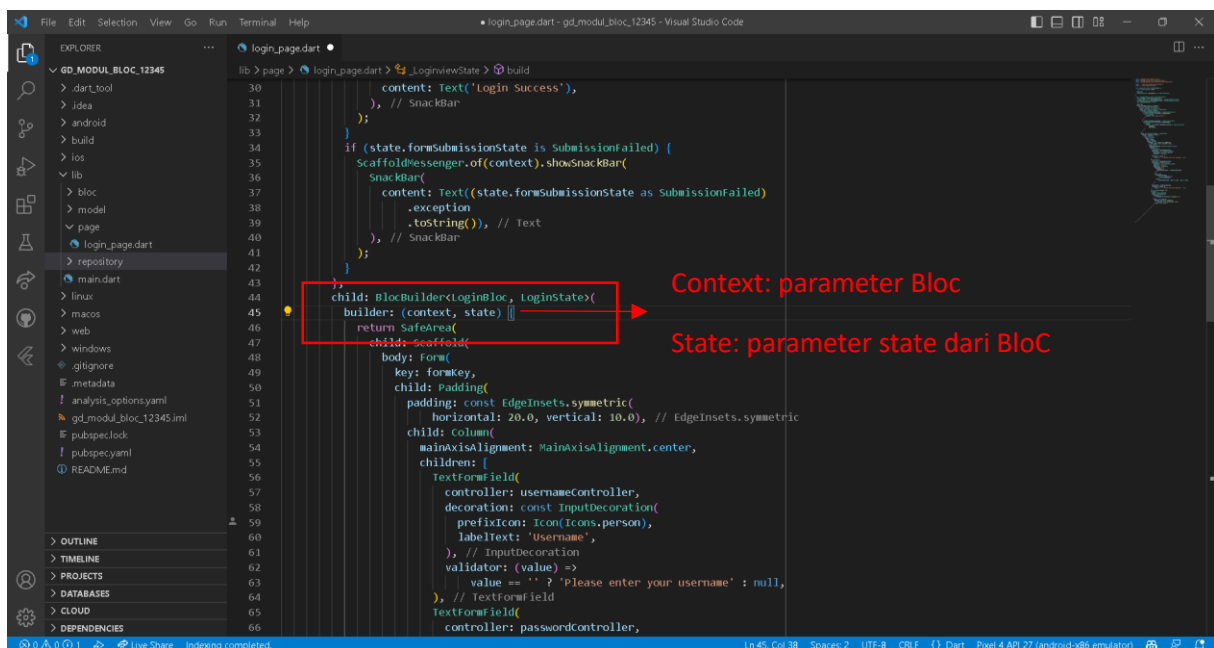
Gambar 20. BlocListener

15. Langkah terakhir, kita perlu menambahkan **BlocBuilder** untuk dapat membangun ulang tampilan jika ada perubahan state. Sama dengan Langkah 12, tapi kita akan memilih **Wrap with StreamBuilder**.



Gambar 21. Cara men-wrap widget dengan StreamBuilder

16. Gantilah isi **StreamBuilder** dengan **BlocBuilder** seperti di bawah.
Melalui BlocBuilder kita dapat mengakses BloC dan statenya melalui keyword **context** dan **state**.



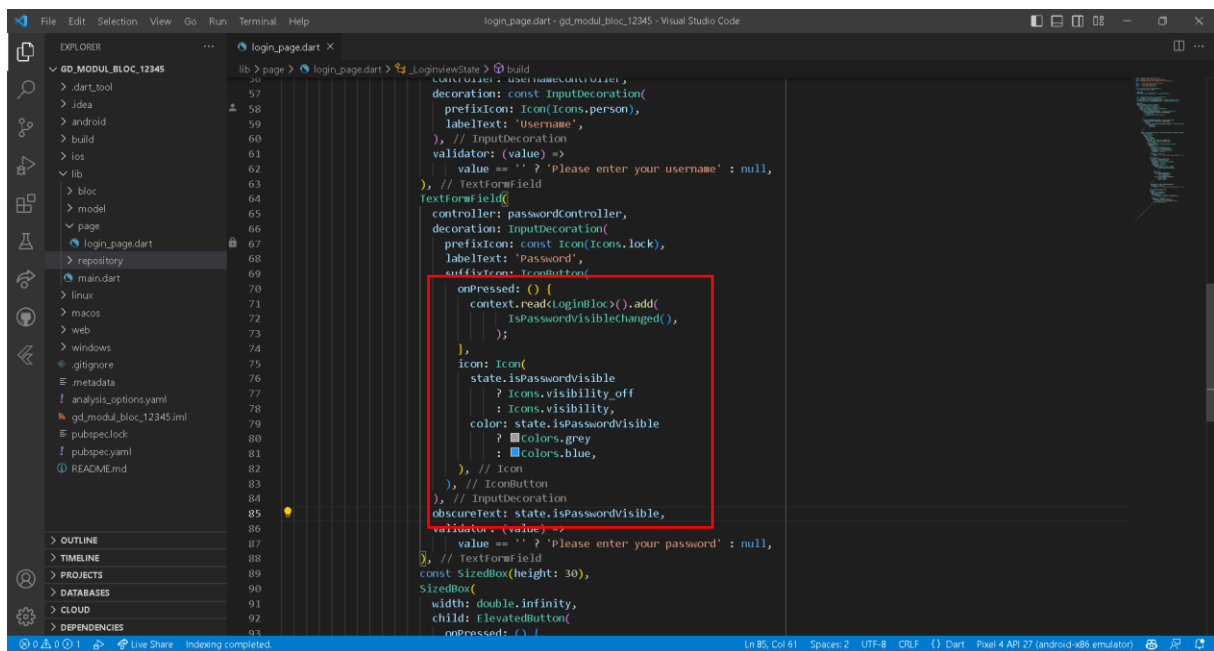
Gambar 22. BlocBuilder

17. Kita akan menggantikan fungsi Stateful dengan BloC pada field password. Perlu diperhatikan pada baris 71 kita menambahkan event IsPasswordVisibleChanged saat IconButton ditekan. Kita dapat

menambahkan event seperti di bawah atau bisa juga dengan menggunakan BlocProvider sebagai berikut (Keduanya memiliki fungsi yang sama). Sekarang kita dapat menghapus variable isPasswordVisible pada baris 14 Gambar 18 karena sudah digantikan dengan state dari LoginBloc.

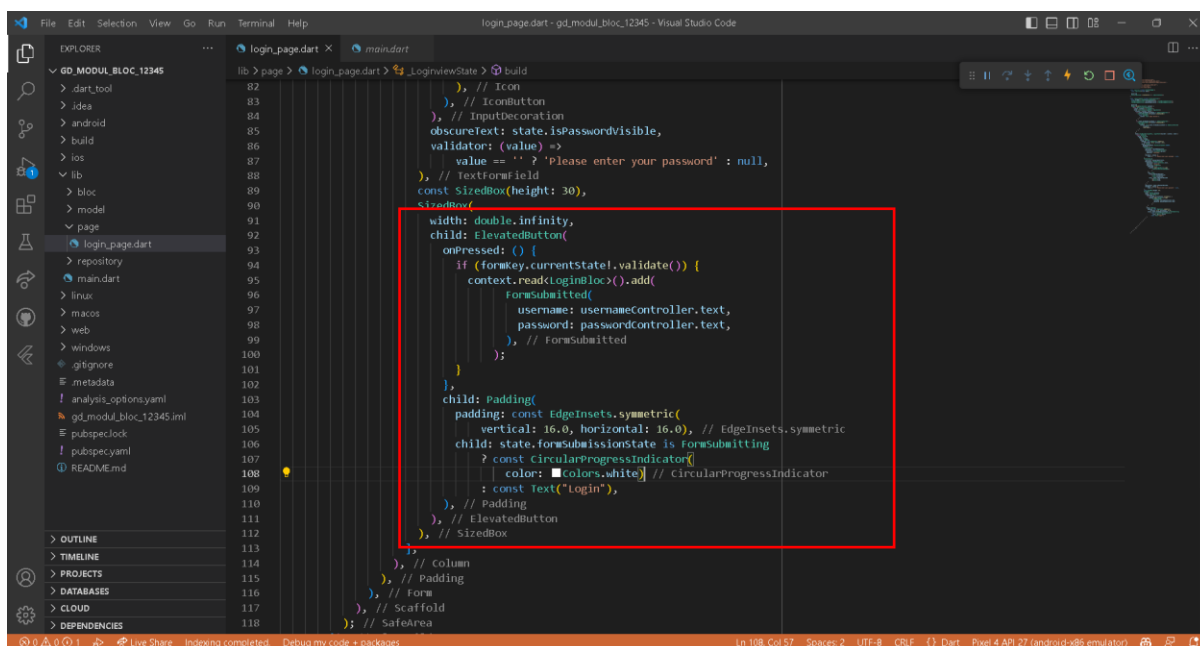
```
BlocProvider.of<LoginBloc>(context).add(IsPasswordVisibleChanged());
```

Gambar 23. Sintaks untuk akses BlocProvider



Gambar 24. Modifikasi suffixIcon dari TextField Password

18. Yang terakhir, kita akan menambahkan event FormSubmitted saat tombol Login ditekan. Untuk meningkatkan UI dari aplikasi, kita menambahkan animasi loading ketika login diproses.



Gambar 25. Tombol Login

NOTE :

Jika tertarik untuk menerapkan BloC pattern di tugas besar, sangat direkomendasikan untuk membaca section [ini](#) dari dokumentasi BloC, terkhusus mengenai BlocBuilder, BlocListener, dan BlocProvider.

ATURAN Pengerjaan Guided :

- Guided dikerjakan selama waktu perkuliahan berlangsung.
- Penamaan projek guided harus sesuai dengan yang sudah dicontohkan.
- Guided dikumpulkan melalui github dengan penamaan setiap file pada github adalah : **NAMAGUIDED_XXXX** (contoh : **Guided1_BloC_9999**)
 - o **NAMAGUIDED → SESUAI DENGAN CONTOH DALAM MODUL INI**
 - o **XXXX → 4 DIGIT TERAKHIR NPM**
- Setelah diupload melalui github, jangan lupa untuk mengumpulkan keseluruhan link file github melalui situs kuliah.
- Cara upload ke github, silahkan melihat pada modul **"UPLOAD GITHUB"**.