2.1 Bloc como está implementado nessa app. Identificar os componentes da arquitetura Bloc dentro do código;

Event: temos o PostFetched que é um tipo de PostEvent, e o PostBloc responderá a esse evento quando precisar de mais Posts pra apresentar, ao receber os eventos ele o converterá em estados. States: no PostState, teremos PostInitial, PostSucess e PostFailure. BlocProvider: widget que é utilizado para fornecer a instância do PostBloc para os widgets descendentes. Ele é usado para injetar o PostBloc no widget da árvore de widgets. PostList: widget que representa a lista de posts. BlocObserver: é utilizado para observar eventos de transição de estado e erros que ocorrem dentro dos blocos em sua aplicação.

2.2 A estrutura de pastas da app para atender questões como modularização, componentes e funcionalidades;

```
├── lib
|   ├── posts
|   |   ├── bloc
|   |   |   └── post_bloc.dart
|   |   |   └── post_event.dart
|   |   |   └── post_state.dart
|   |   └── models
|   |   |   └── models.dart*
|   |   |   └── post.dart
|   |   └── view
|   |   |   ├── posts_page.dart
|   |   |   └── posts_list.dart
|   |   |   └── view.dart*
|   |   └── widgets
|   |   |   └── bottom_loader.dart
|   |   |   └── post_list_item.dart
|   |   |   └── widgets.dart*
|   |   ├── posts.dart*
|   ├── app.dart
|   ├── simple_bloc_observer.dart
|   └── main.dart
├── pubspec.lock
├── pubspec.yaml
```

2.3 Que dados são tratados pelo Gerenciamento de Estado.

Entrada do usuário (scrolling), dados das postagens, mensagens de erro.