1. Что такое BLoC? Основные компоненты.

BLoC (Business Logic Component) — это архитектурный паттерн, используемый в приложениях Flutter для отделения бизнес-логики от пользовательского интерфейса. Цель BLoC — обеспечить управление состоянием и отделить логику приложения от его представления, что делает код более читаемым, повторно используемым и тестируемым.

Основные компоненты BLoC:

Stream: Поток данных, который передает изменения состояния. Используется для получения текущих данных и их обновлений.

StreamController: Контроллер потоков, который управляет отправкой данных в потоки и их подписчикам.

Sink: Канал для передачи данных в поток. Он используется для добавления новых данных, которые затем будут переданы всем подписчикам потока.

BLoC-класс: Класс, в котором реализуется вся бизнес-логика и управление состоянием. В нем используются потоки и методы для управления данными, чтобы обеспечивать реактивное обновление интерфейса.

BLoC Provider (InheritedWidget): Виджет, используемый для передачи экземпляра BLoC дочерним виджетам. Это облегчает доступ к BLoC в дереве виджетов, не передавая его напрямую каждому виджету.

2. Как производится оповещение widget’а о изменении состояния?

Оповещение виджета об изменении состояния в BLoC паттерне происходит через использование StreamBuilder. StreamBuilder — это специальный виджет, который позволяет подписаться на поток данных и автоматически обновляться при поступлении новых данных. При добавлении данных в поток (через Sink), StreamBuilder получает уведомление и вызывает метод build, что приводит к перерисовке пользовательского интерфейса с учетом нового состояния.

Пример:

StreamBuilder<List<Product>>(

stream: bloc.productsStream,

builder: (context, snapshot) {

if (snapshot.hasData) {

return ListView.builder(

itemCount: snapshot.data!.length,

itemBuilder: (context, index) {

final product = snapshot.data![index];

return ListTile(title: Text(product.name));

},

);

} else {

return CircularProgressIndicator();

}

},

);

В этом примере StreamBuilder отслеживает изменения потока productsStream, и при получении новых данных автоматически обновляет интерфейс.

3. Где инициализируется BLoC?

BLoC обычно инициализируется на уровне виджета, который отвечает за управлением его жизненным циклом, чаще всего — на уровне приложения или экрана. Для передачи экземпляра BLoC используется BLoC Provider (часто это наследник InheritedWidget), который предоставляет доступ к экземпляру BLoC всем дочерним виджетам.

Пример инициализации BLoC:

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return BlocProvider(

bloc: ProductBloc(), // Инициализация BLoC

child: MaterialApp(

home: HomeScreen(),

),

);

}

}

В этом примере экземпляр ProductBloc инициализируется в BlocProvider и передается всему приложению.

4. Как получить конкретный экземпляр BLoC?

Чтобы получить доступ к конкретному экземпляру BLoC, используется BLoC Provider, реализованный как InheritedWidget. Благодаря этому подходу, любой дочерний виджет может получить экземпляр BLoC, не передавая его через параметры вручную.

Пример получения экземпляра BLoC в дочернем виджете:

final bloc = BlocProvider.of(context);

Здесь BlocProvider.of(context) находит ближайший BlocProvider в дереве виджетов и возвращает экземпляр ProductBloc, который был инициализирован ранее. Это удобно для управления зависимостями и упрощает повторное использование BLoC в различных частях приложения.

Итак, BLoC — это паттерн, используемый для отделения логики от интерфейса и управления состоянием с помощью потоков (Stream). Оповещение виджетов о состоянии производится с помощью StreamBuilder, BLoC инициализируется в BlocProvider, а экземпляр BLoC можно получить через BlocProvider.of(context).