Студенческая
ИТ-лаборатория

State management и работа с данными при разработке на Flutter

Владислав Зюзько
Flutter-разработчик в компании Effective

Студенческая
ИТ-лаборатория

effectiveband

Что такое State management? И при чём тут Flutter?

Вводим понятие State

State (состояние) – это набор данных, который определяет внешний вид какой-либо части приложения (UI).

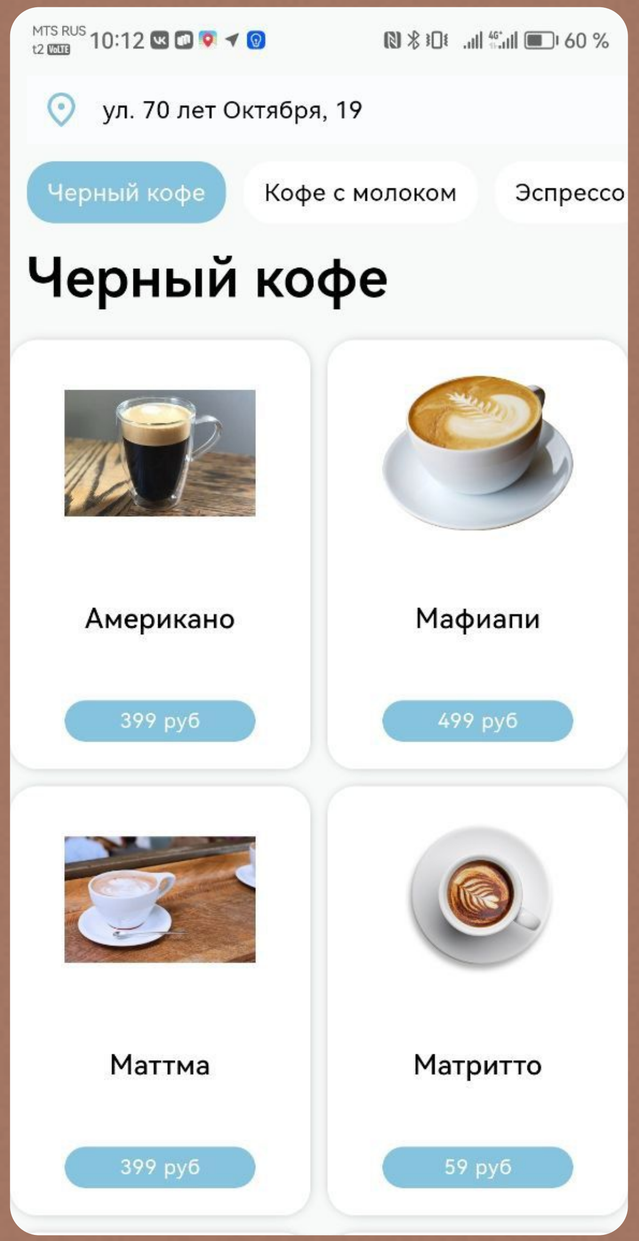
Global State – это набор данных, который определяет внешний вид всего приложения.

UI = f (state)

The layout on the screen

Your build methods

The application state

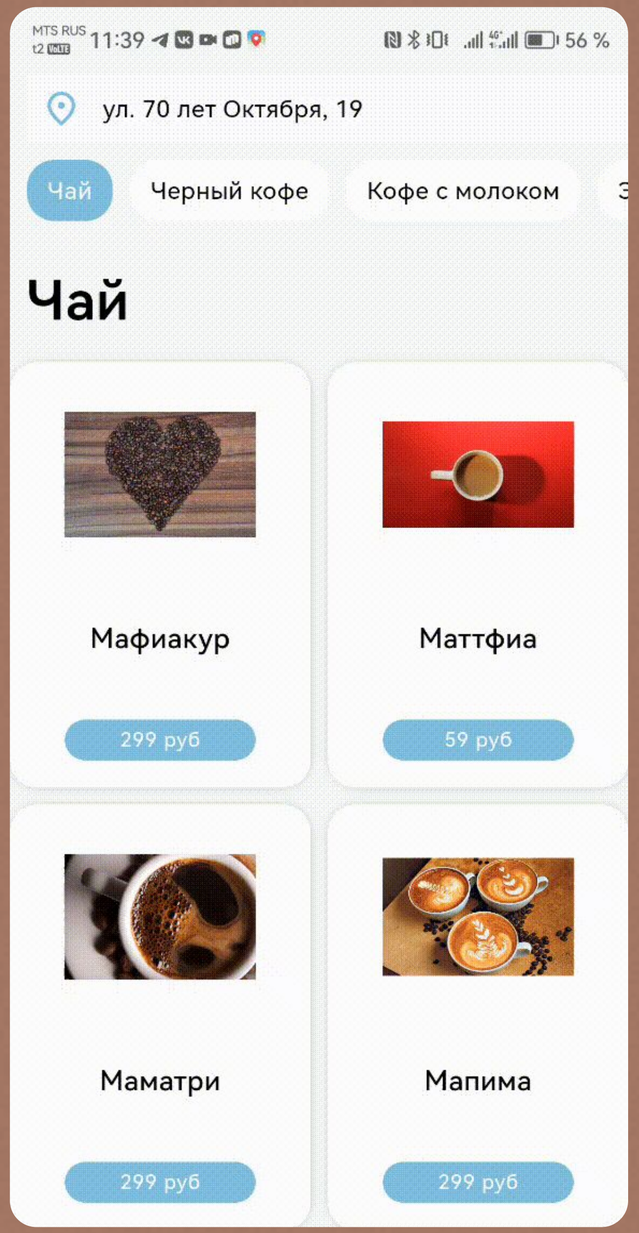


State management

State Management – подход к управлению состоянием в приложении.

Управление над состоянием включает:

- его создание;
- осуществление к нему доступа;
- его обновление;
- его организацию и хранение.



State in Flutter

effectiveband

Всё, что мы видим на экране Flutter-приложения, описывается с помощью виджетов, которые преобразуются в картинку на экране.

Именно поэтому говорят, что Flutter – это фреймворк, в котором UI описывается декларативно.

Виды виджетов в Flutter:

- StatelessWidget – виджет, который не имеет состояния;
- StatefulWidget – виджет, который имеет состояние;
- InheritedWidget – виджет для передачи данных дочерним виджетам.

State in Flutter

effectiveband

```
1   import 'package:flutter/material.dart';
2
3   class YourStatefulWidget extends StatefulWidget {
4     const YourStatefulWidget({super.key});
5
6     @override
7     State<YourStatefulWidget> createState() => _YourStatefulWidgetState();
8   }
9
10  class _YourStatefulWidgetState extends State<YourStatefulWidget> {
11    @override
12    Widget build(BuildContext context) {
13      return const Placeholder();
14    }
15  }
16
```

Updating state in Flutter

effectiveband

```
1   import 'package:flutter/material.dart';
2
3   class ButtonExample extends StatefulWidget {
4     const ButtonExample({super.key});
5
6     final enabledText = 'Enabled';
7     final disabledText = 'Disabled';
8
9     @override
10    State<ButtonExample> createState() => _ElevatedButtonExampleState();
11  }
```

To be continued ▶ ▶

Updating state in Flutter

effectiveband

```
13  class _ElevatedButtonExampleState extends State<ButtonExample> {
14    bool buttonIsEnabled = true;
15
16    @override
17    Widget build(BuildContext context) {
18      return Center(
19        child: ElevatedButton(
20          onPressed: () => setState(() {
21            buttonIsEnabled = !buttonIsEnabled;
22          }),
23          child: Text(
24            buttonIsEnabled ? widget.enabledText : widget.disabledText,
25          ), // Text
26        ), // ElevatedButton
27      ); // Center
28    }
29  }
```

setState method

effectiveband

- Наиболее простой способ обновления состояния
- Обновляет всю структуру виджетов внутри State
- UI-код и бизнес-логика находятся вместе

```
@protected
void setState(VoidCallback fn) {
  final Object? result = fn() as dynamic;
  _element!.markNeedsBuild();
}
```

Element

~

BuildContext

ValueNotifier

effectiveband

```
13  class _ButtonExampleState extends State<ButtonExample> {
14    ValueNotifier<bool> buttonIsEnabled = ValueNotifier(true);
15
16    @override
17    Widget build(BuildContext context) {
18      return Center(
19        child: ElevatedButton(
20          onPressed: () => setState(() {
21            buttonIsEnabled.value = !buttonIsEnabled.value;
22          }),
23          child: ValueListenableBuilder(
24            valueListenable: buttonIsEnabled,
25            builder: (context, value, _) {
26              return Text(
27                value ? widget.enabledText : widget.disabledText,
28              ); // Text
29            },
30          ), // ValueListenableBuilder
31        ), // ElevatedButton
32      ); // Center
33    }
34  }
```

example

ValueNotifier

effectiveband

- Позволяет точечно обновлять UI
- Бизнес-логика всё так же смешивается с UI

```
class ValueNotifier<T> extends ChangeNotifier implements ValueListenable<T> {
  /// Creates a [ChangeNotifier] that wraps this value.
  ValueNotifier(this._value) {
    if (kFlutterMemoryAllocationsEnabled) {
      ChangeNotifier.maybeDispatchObjectCreation(this);
    }
  }
```

ChangeNotifier

effectiveband

```
13  class ButtonController with ChangeNotifier {
14    bool _isEnabled = true;
15    bool get isEnabled => _isEnabled;
16
17    void changeButtonStatus() => _isEnabled = !_isEnabled;
18  }
```

В отличии от ValueNotifier, ChangeNotifier позволяет работать с изменяемыми типами данных.

ChangeNotifier

effectiveband

```
20  class _ButtonExampleState extends State<ButtonExample> {
21    final ButtonController buttonController = ButtonController();
22
23    @override
24    Widget build(BuildContext context) {
25      return Center(
26        child: ElevatedButton(
27          onPressed: () => setState(fn: buttonController.changeButtonStatus),
28          child: ListenableBuilder(
29            listenable: buttonController,
30            builder: (BuildContext context, Widget? _) {
31              return Text(
32                data: buttonController.isEnabled
33                      ? widget.enabledText
34                      : widget.disabledText,
35              ); // Text
36            },
37          ), // ListenableBuilder
38        ), // ElevatedButton
39      ); // Center
40    }
41  }
```

example

ChangeNotifier

effectiveband

- Всё так же позволяет выборочно обновлять UI
- Бизнес-логика отделима от UI-слоя
- Сложно организовывать бизнес-состояния, отсутствие абстракций и инструментов для работы с асинхронными данными

```
Open online interactive samples for ChangeNotifier
mixin class ChangeNotifier implements Listenable {
  int _count = 0;
  static final List<VoidCallback?> _emptyListeners = List<VoidCallback?>.filled(0, null);
  List<VoidCallback?> _listeners = _emptyListeners;
  int _notificationCallStackDepth = 0;
  int _reentrantlyRemovedListeners = 0;
  bool _debugDisposed = false;
  bool _creationDispatched = false;
```

State management in Flutter

effectiveband

State management – подход к управлению состоянием в приложении.

State manager – набор инструментов, с помощью которого достигается выбранный state management.

Наиболее популярные State manager-ы в Flutter:

- Provider – обёртка над InheritedWidget-ами
- Riverpod – предоставляет объекты для реактивного управления состоянием и организации DI
- BLoC – позволяет управлять состоянием реактивно с отделением слоя бизнес-логики от UI

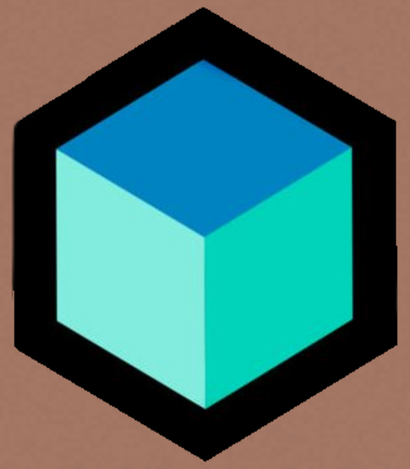
Подробнее про BLoC

Что такое BLoC?

effectiveband

BLoC (Business Logic Component) – архитектурный паттерн, позволяющий управлять состоянием с разделением слоёв бизнес-логики и UI.

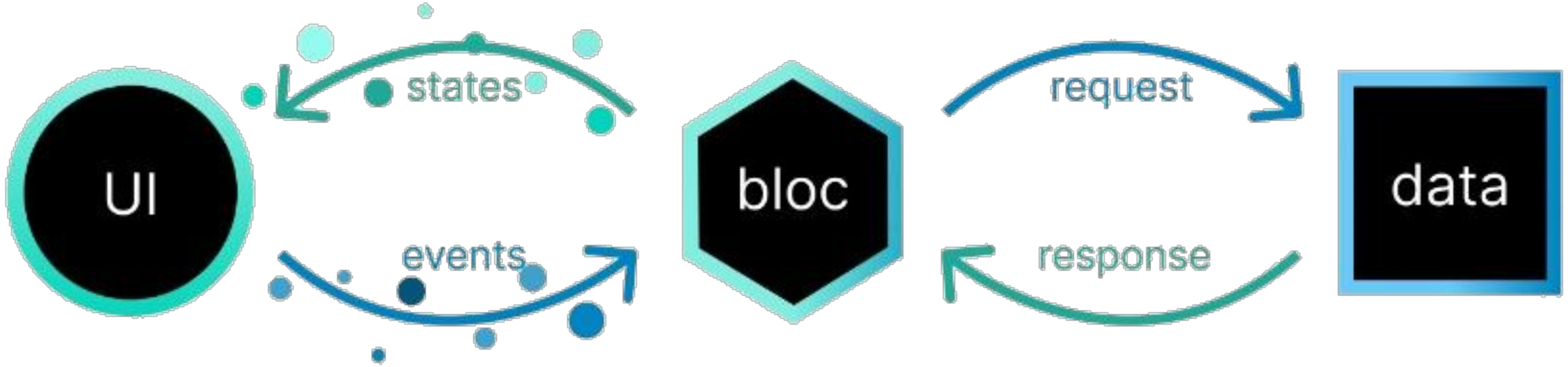
Bloc – класс, который инкапсулирует часть бизнес-логики приложения.



Dart

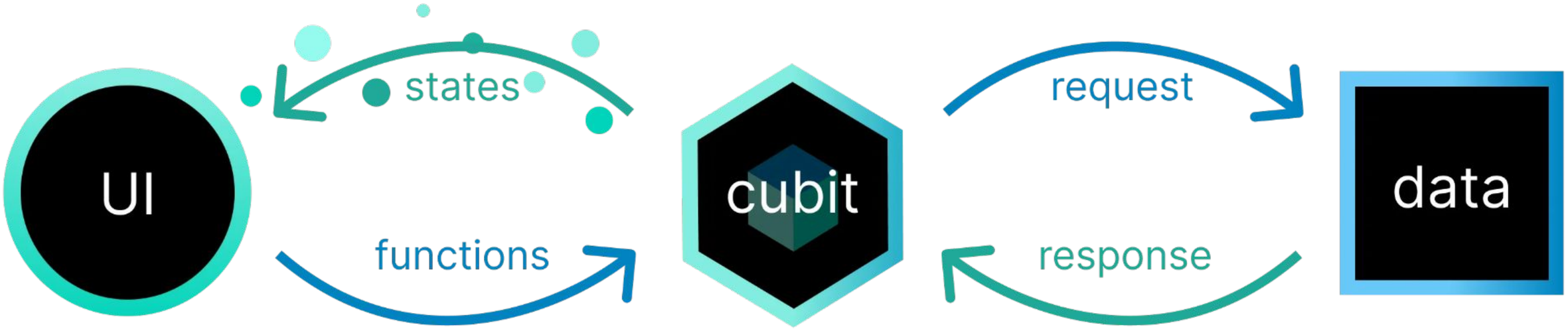
Концепция BLoC

effectiveband



Cubit

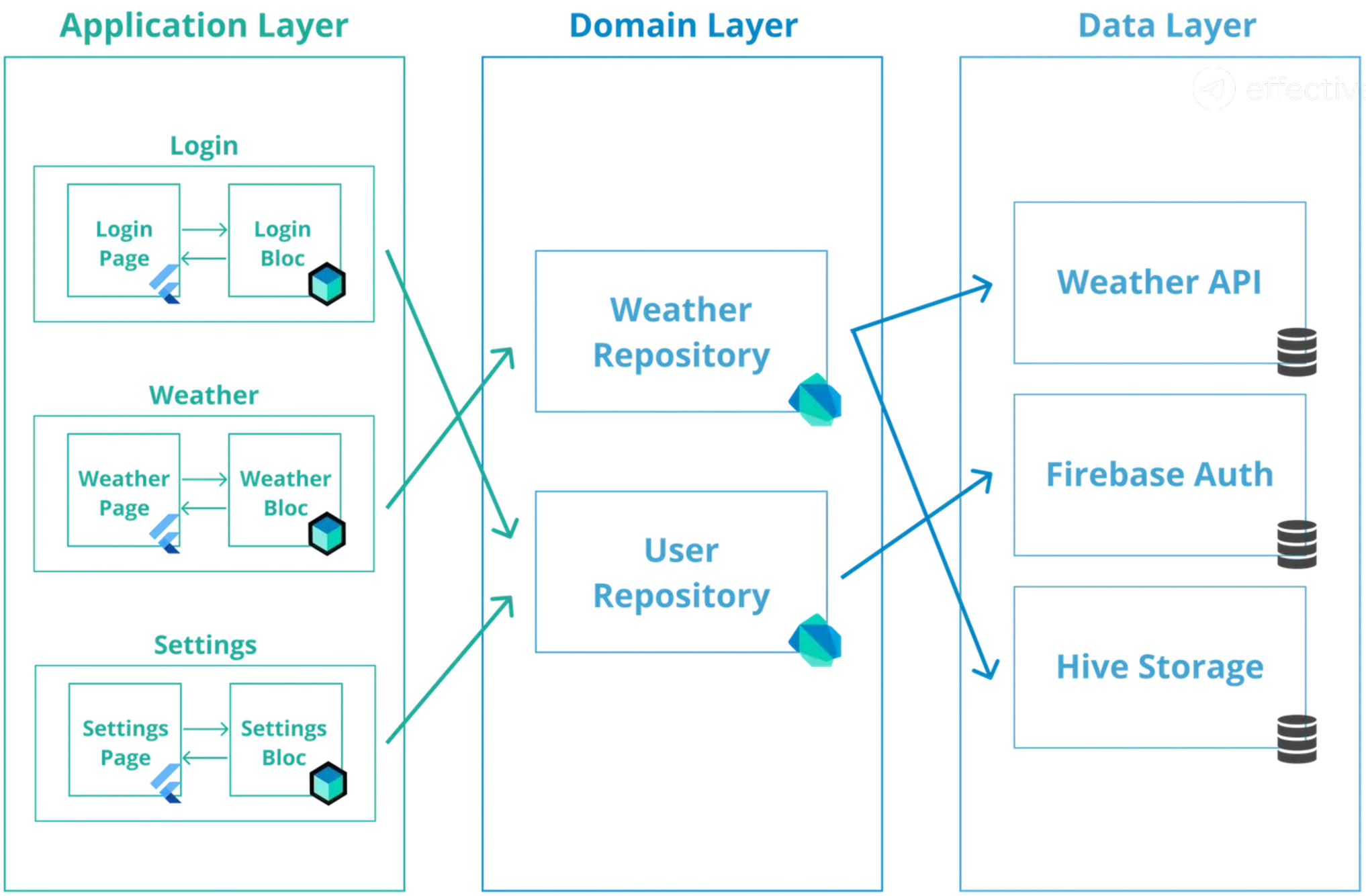
effectiveband



1. Приложение делится на три слоя

effectiveband

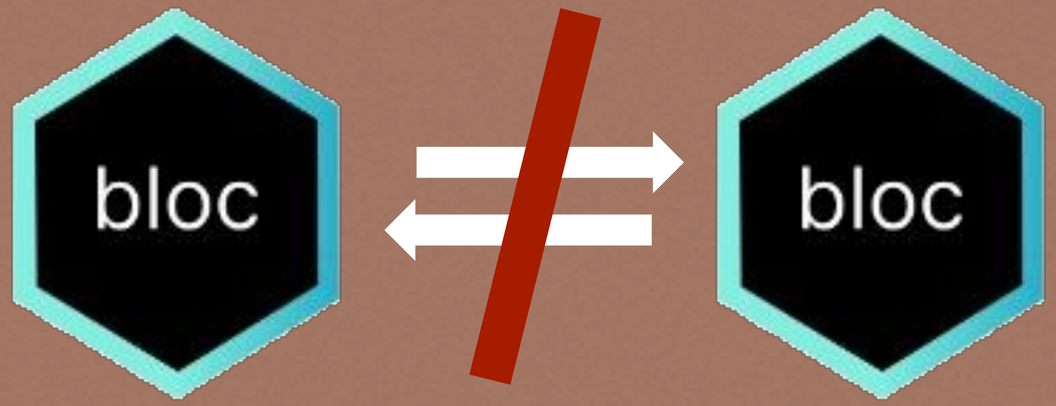
- Presentation (UI) – экраны, виджеты;
- Business logic – представлен Bloc-ами;
- Data – представлен репозиториями, источниками данных.


Application Layer
Domain Layer
Data Layer
effectiveband
Login
Login Page
Login Bloc
Weather
Weather Page
Weather Bloc
Settings
Settings Page
Settings Bloc
Weather Repository
User Repository
Weather API
Firebase Auth
Hive Storage

2. Взаимодействие Bloc-ов

effectiveband

- Bloc-и ничего не должны знать друг о друге, так как это компоненты одного уровня;
- В противном случае будет получена сложная логика, в которой увеличивается вероятность и ценность ошибки, а также повышается сложность тестирования.

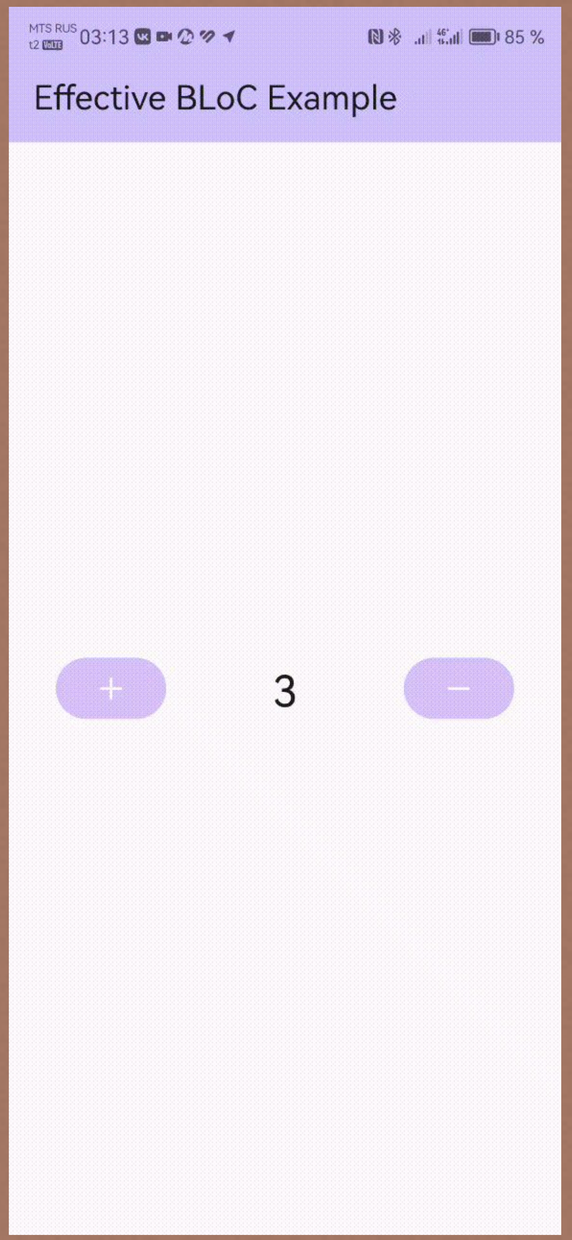


Пакеты для работы с BLoC во Flutter

effectiveband

- bloc (9.0.0 now, 20 days ago)
- flutter_bloc (9.0.0 now, 20 days ago)
- bloc_test (10.0.0 now, 20 days ago)
- bloc_concurrency (0.3.0 now, 20 days ago)

Пример приложения на BLoC



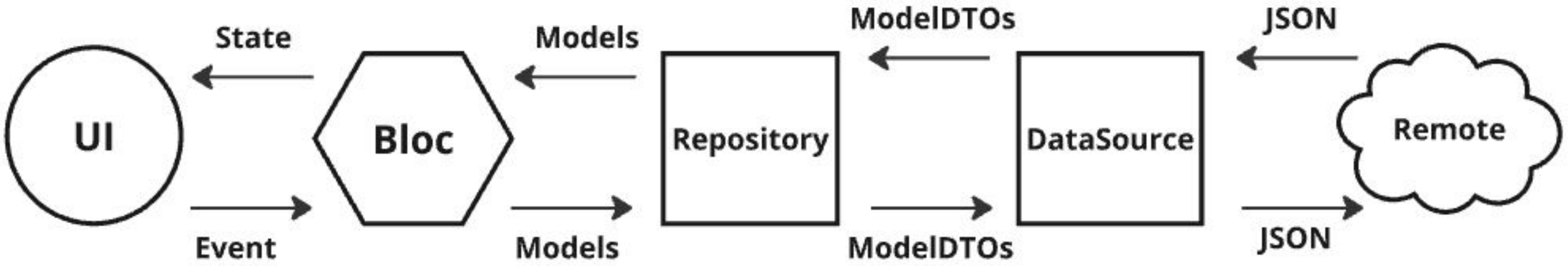
Студенческая
ИТ-лаборатория

effectiveband

Работа с данными в Flutter

Общая архитектура

effectiveband



Инструменты для работы с сетью

effectiveband

- **dio** (5.8.0+1 now, 3 days ago) – из коробки есть интерцепторы, глобальные настройки, таймауты и прочее;
- **http** (1.3.0 now, 12 days ago) – по большей части голый клиент для походов в сеть, по производительности не уступает dio;
- **retrofit** (4.4.2 now, 33 days ago) – модуль для кодогенерации клиента dio и его запросов.

Инструменты для работы с локальным хранилищем

effectiveband

- **shared_preferences** (2.5.1 now, 3 days ago) – хранит данные в виде key-value, подходит для хранения несенсорных простых данных;
- **drift** (2.24.0 now, 11 days ago) – sqlite-хранилище, подходит для хранения реляционных баз данных, есть ORM на Dart;
- **sqflite** (2.4.1 now, 2 months ago) – также предоставляет доступ к sqflite-хранилищу, но взаимодействие с базой возможно только через SQL.

DTO

effectiveband

DTO (Data Transfer Object) – объект для передачи данных, в общем случае не формирующий бизнес-состояние. Соответствует объекту, переданному из удалённого источника, содержит методы для сериализации / десериализации в соответствии с рабочим форматом данных.

```
import 'package:json_annotation/json_annotation.dart';

part 'category_dto.g.dart';

@JsonSerializable()
class ModelDTO {
  final int id;

  const ModelDTO({required this.id});

  factory ModelDTO.fromJson(Map<String, dynamic> json) => _$ModelDTOFromJson(json);

  Map<String, dynamic> toJson() => _$ModelDTOToJson(this);
}
```

DataSource

effectiveband

```
abstract interface class ISomeDataSource {
  Future<List<ModelDTO>> getData();
}

class NetworkSomeDataSource implements ISomeDataSource {
  final Dio _dio;

  const NetworkSomeDataSource({required Dio dio}) : _dio = dio;

  @override
  Future<List<ModelDTO>> getData() async {
    Response response = await _dio.get(
      "/some/url",
    );
    List<ModelDTO> dtos = (
      json.decode(json.encode(response.data))['data'] as List<dynamic>).map(
        (json) => ModelDTO.fromJson(json),
      ).toList();
    return dtos;
  }
}
```

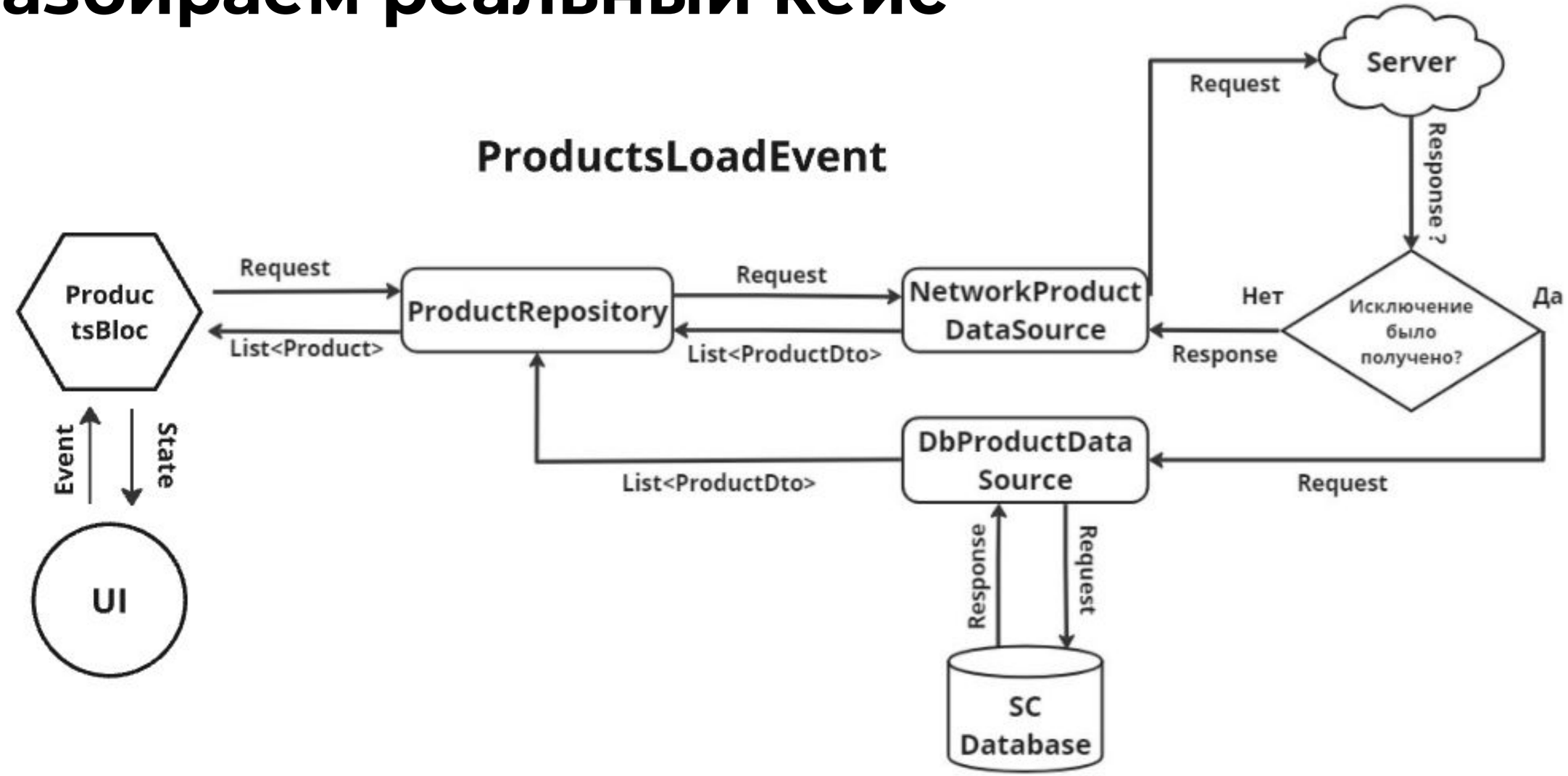
Repository

effectiveband

```
abstract class ISomeRepository {
  Future<List<Model>> getData();
}

class SomeRepository implements ISomeRepository {
  final ISomeDataSource _networkSomeDataSource;

  const SomeRepository({
    required ISomeDataSource networkSomeDataSource,
  }) : _networkSomeDataSource = networkSomeDataSource;

  @override
  Future<List<Model>> getData() async {
    try {
      List<ModelDTO> dtos =  await _networkSomeDataSource.getData();
      return dtos.map((modelDTO) => modelDTO.toModel()).toList();
    } catch (e, st) {
      throw Exception("Models load failed: $st");
    }
  }
}
```

Разбираем реальный кейс

effectiveband



Благодарю за внимание!

@zyuzko2002

Student Labs VK

Владислав Зюзько

Flutter-разработчик в компании Effective