

# Трудоустройство

## Ошибки

Не слушал опытных разработчиков

Не сразу начал писать код

Не следил за временем

Интересовался теорией, затем изменил его в сторону практики

Раньше начинал вести свой блог и активное общение в чатах

Тянул с трудоустройством, ибо Наниматели отвечают довольно долго

Не прокачивал сразу LinkedIn (Для зарубежных)

## Советы

Упор на практику (самостоятельное написание кода)

Теория без практики выветрится за неделю

Взять 30 вакансий и выпишите повторяющиеся технологии

СЛЕДИТЕ за временем Toggle и Pomodoro

Для понимания англ видео Yandex Browser имеет опцию аудио перевода

Изучение английского языка

Добавьтесь в ТГ-чаты по направлению

Улучшите LinkedIn и общайтесь с рекрутерами

Вести блог для самомотивации (фиксирует достижения и отслеживание успехов + будущее и медийность)

Не кусайте больше, чем можете прожевать и не заикливайтесь на том, чего не знаете, это приведёт к прокрастинации. Фокус на мелких победах будут мотивировать.

## Заключение

Если нет опыта, не заикливайтесь на мелочах, которые не получаются

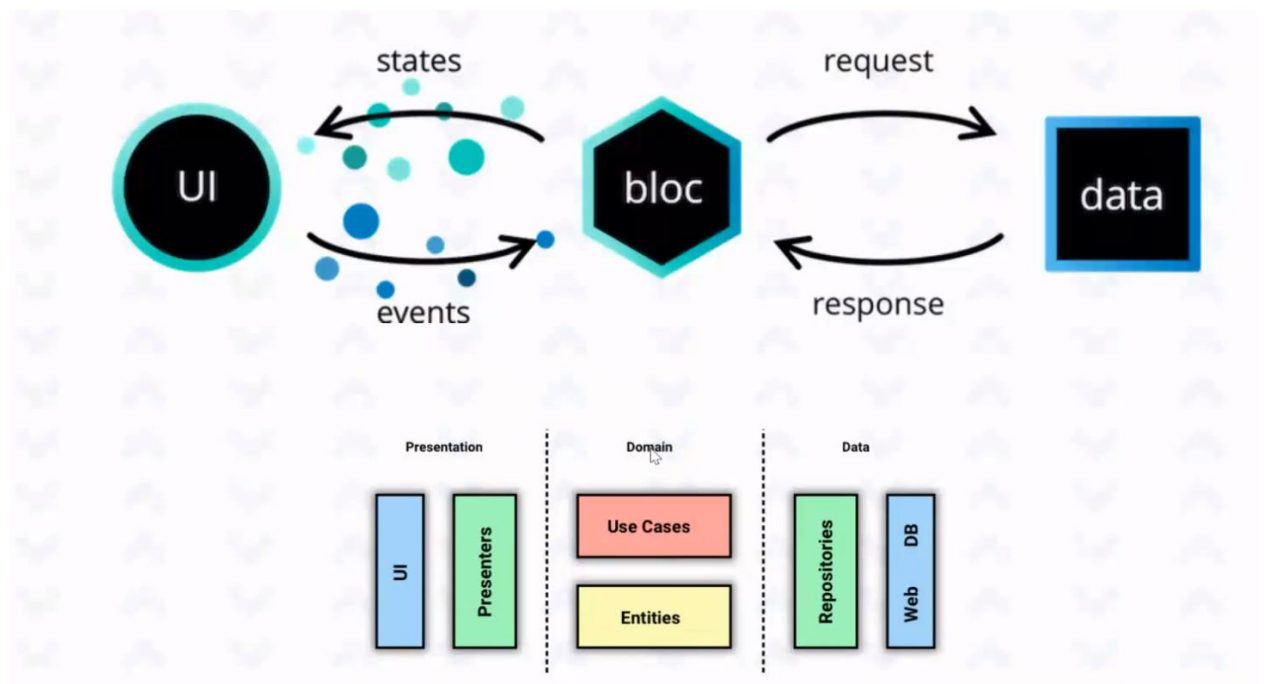
Книги устаревают в момент выхода, инфу лучше черпать с инета  
Книга Flutter apprentice недавно вышла

Flutter позволяет использовать приложение на всех платформах

Flutter – активная поддержка и сам всё рисует с помощью своего движка (в отличие от Xamarin)

## Паттерн BLoC и его реализация во Flutter

Чистая Архитектура + bloc



Реализация Bloc  
flutter\_bloc 8.1.1



Валидация

Основные реализации: BLoC и Cubit

Разница – BLoC – События(events), Cubit – Функции(functions)



- **BlocProvider** — это виджет, который предоставляет блок/кубит своим дочерним элементам через `BlocProvider.of<T>(context)`. В большинстве случаев **BlocProvider** используется для создания новых блоков. В этом случае **BlocProvider** отвечает за создание блока, и он автоматически закрывает блок/кубит.

```
BlocProvider(  
  create: (BuildContext context) => BlocA(),  
  child: ChildA(),  
);
```

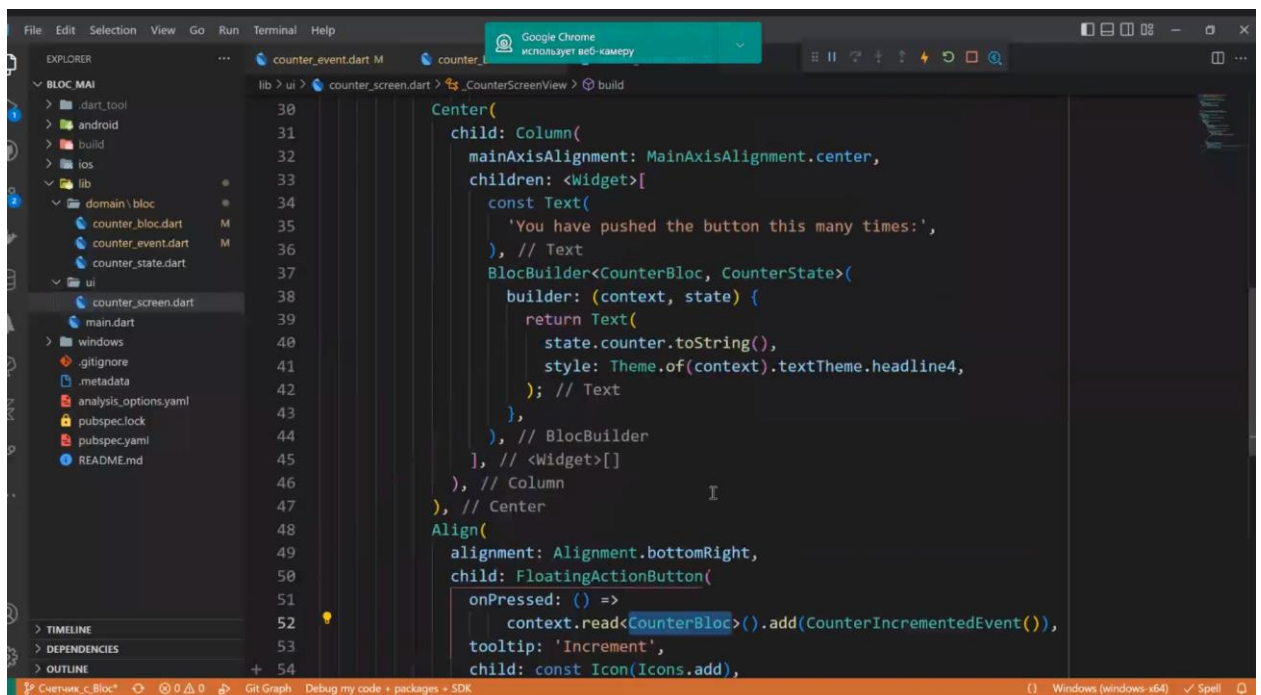
- **ВАЖНО** — По умолчанию **BlocProvider** создает блок/кубит лениво (lazy).

## BlocProvider.value

- В некоторых случаях существующий блок/кубит необходимо сделать доступным для нового маршрута.

```
BlocProvider.value(  
  value: BlocProvider.of<BlocA>(context),  
  child: ScreenA(),  
);
```

- **ВАЖНО** — В данном случае **BlocProvider** не отвечает за создание блока/кубита, и соответственно не закрывает его автоматически.



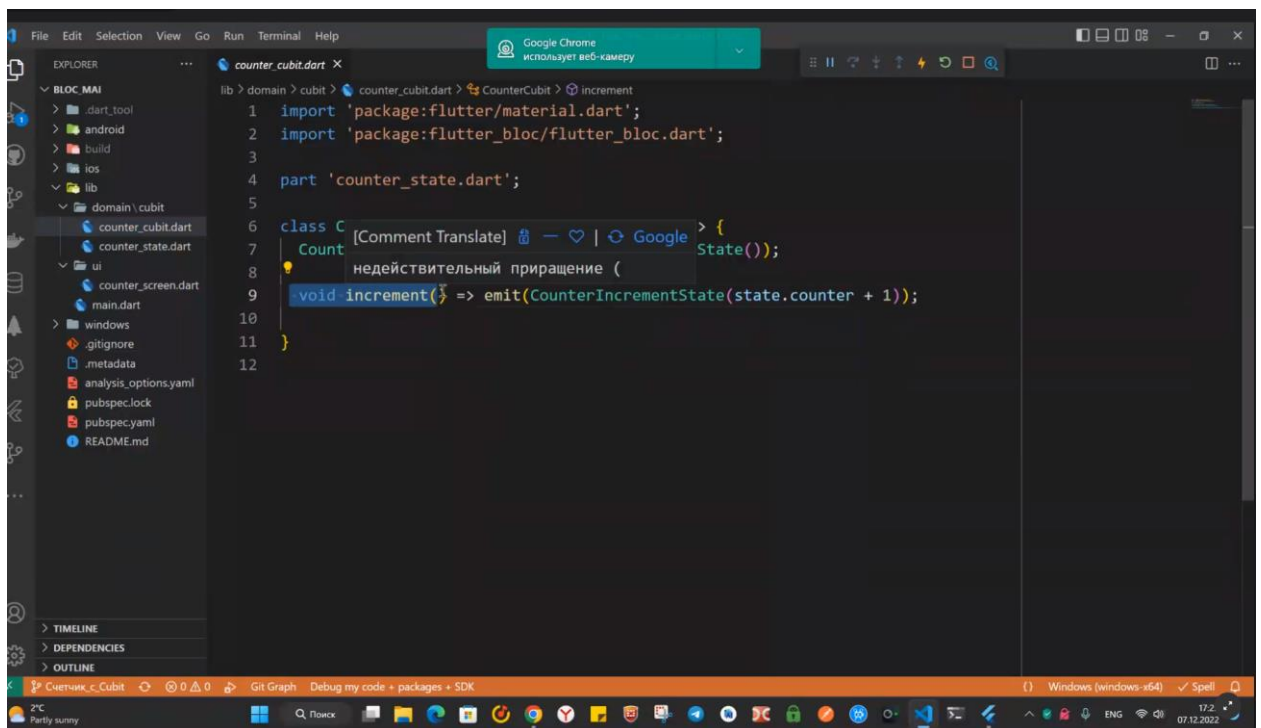
# BlocBuilder

- **BlocBuilder** — это виджет Flutter, который обрабатывает создание виджета в ответ на новые состояния.

```
BlocBuilder<BlocA, BlocAState>(  
  builder: (context, state) {  
    // return widget here based on BlocA's state  
  }  
)
```

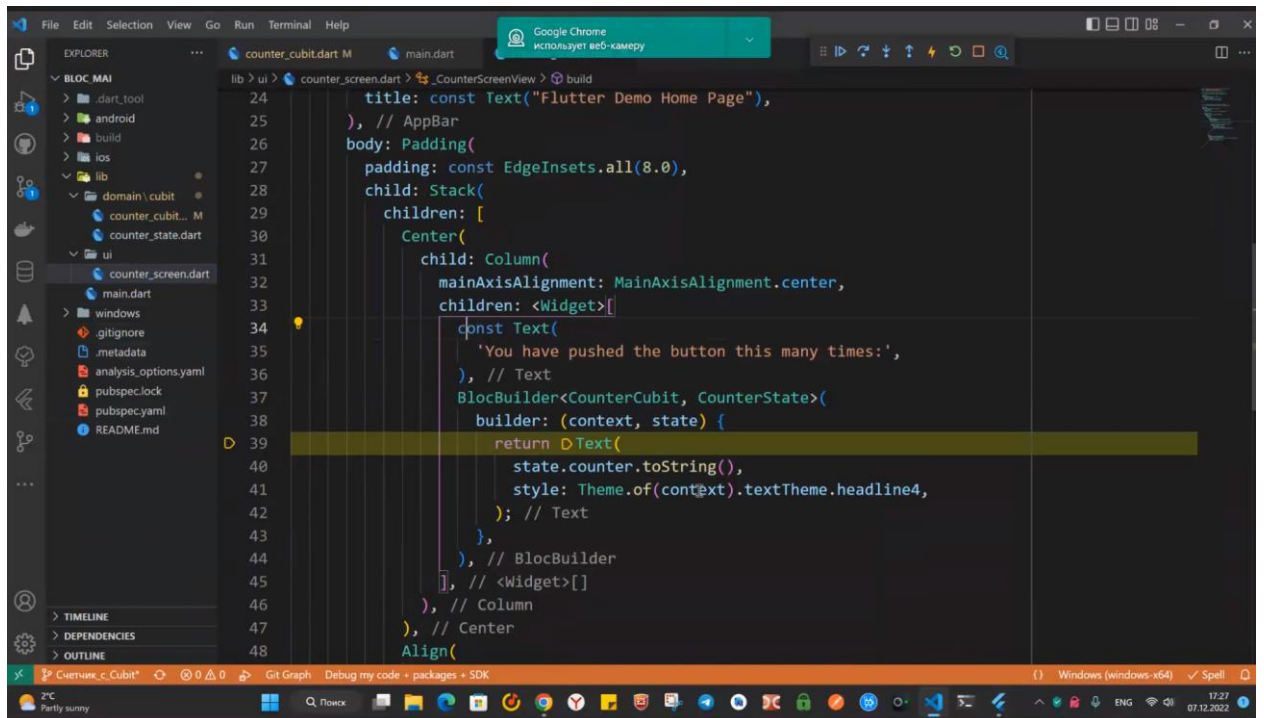
- **ВАЖНО** — Функция build должна быть “Чистой функцией” возвращающей виджет.

Lazy

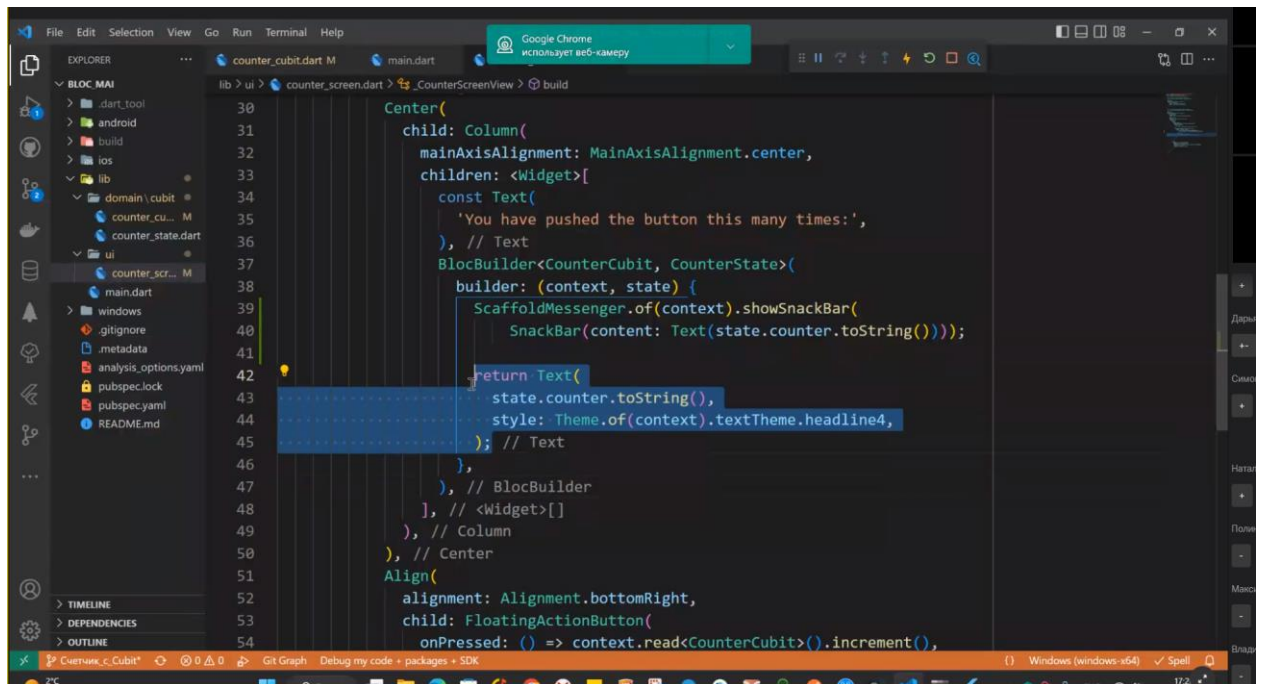


Cubit – 95% всех случаев

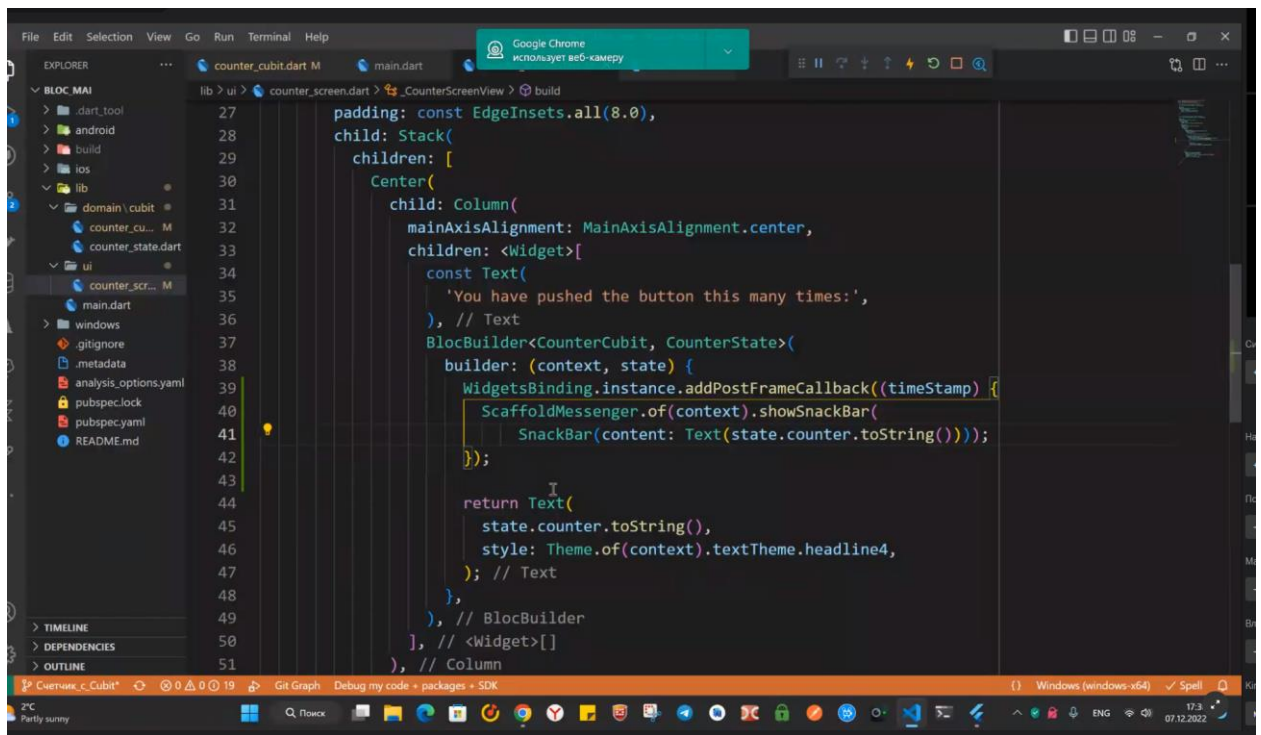




```
lib > ui > counter_screen.dart > _CounterScreenView > build
24 title: const Text("Flutter Demo Home Page"),
25 ), // AppBar
26 body: Padding(
27   padding: const EdgeInsets.all(8.0),
28   child: Stack(
29     children: [
30       Center(
31         child: Column(
32           mainAxisAlignment: MainAxisAlignment.center,
33           children: <Widget>[
34             const Text(
35               'You have pushed the button this many times:',
36             ), // Text
37             BlocBuilder<CounterCubit, CounterState>(
38               builder: (context, state) {
39                 return DText(
40                   state.counter.toString(),
41                   style: Theme.of(context).textTheme.headline4,
42                 ); // Text
43               },
44             ), // BlocBuilder
45           ], // <Widget>[]
46         ), // Column
47       ), // Center
48     ],
49   ),
50 ),
51 ),
52 ),
53 ),
54 ),
55 ),
56 ),
57 ),
58 ),
59 ),
60 ),
61 ),
62 ),
63 ),
64 ),
65 ),
66 ),
67 ),
68 ),
69 ),
70 ),
71 ),
72 ),
73 ),
74 ),
75 ),
76 ),
77 ),
78 ),
79 ),
80 ),
81 ),
82 ),
83 ),
84 ),
85 ),
86 ),
87 ),
88 ),
89 ),
90 ),
91 ),
92 ),
93 ),
94 ),
95 ),
96 ),
97 ),
98 ),
99 ),
100 ),
```

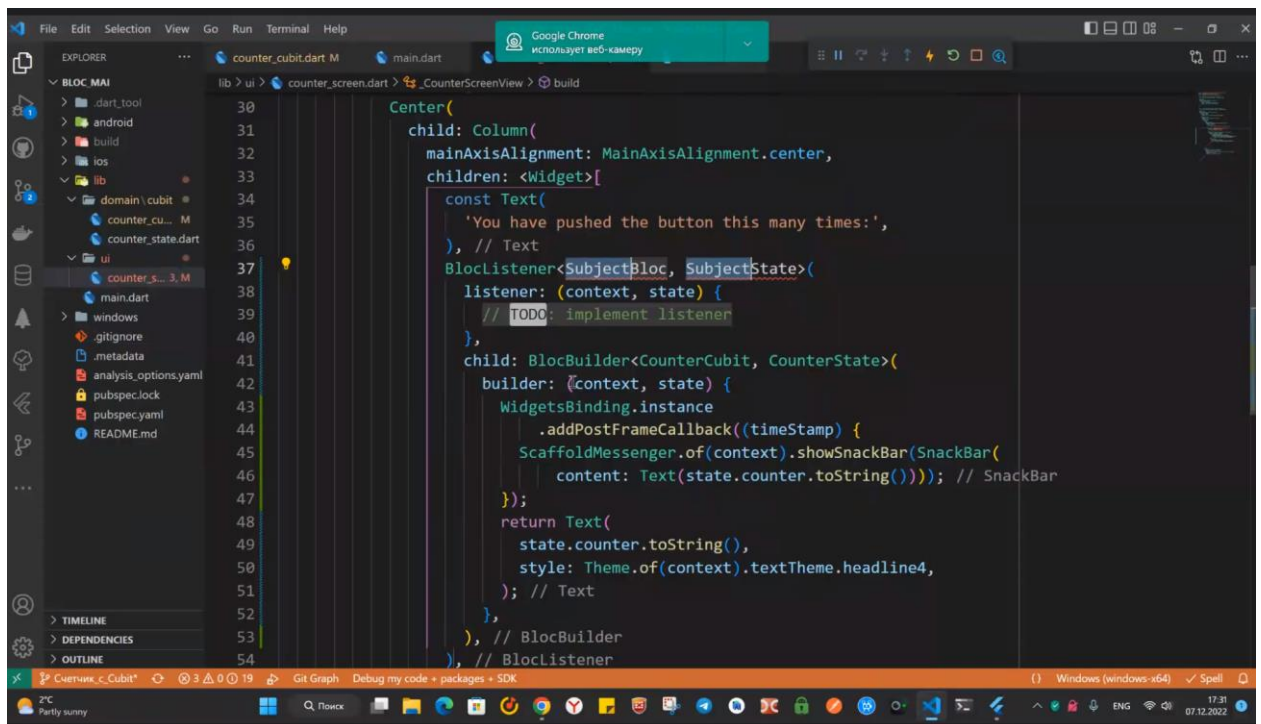


```
lib > ui > counter_screen.dart > _CounterScreenView > build
30 Center(
31   child: Column(
32     mainAxisAlignment: MainAxisAlignment.center,
33     children: <Widget>[
34       const Text(
35         'You have pushed the button this many times:',
36       ), // Text
37       BlocBuilder<CounterCubit, CounterState>(
38         builder: (context, state) {
39           ScaffoldMessenger.of(context).showSnackBar(
40             SnackBar(content: Text(state.counter.toString())));
41         },
42       ), // BlocBuilder
43       return Text(
44         state.counter.toString(),
45         style: Theme.of(context).textTheme.headline4,
46       ); // Text
47     ], // <Widget>[]
48   ), // Column
49 ), // Center
50 ),
51 ),
52 ),
53 ),
54 ),
55 ),
56 ),
57 ),
58 ),
59 ),
60 ),
61 ),
62 ),
63 ),
64 ),
65 ),
66 ),
67 ),
68 ),
69 ),
70 ),
71 ),
72 ),
73 ),
74 ),
75 ),
76 ),
77 ),
78 ),
79 ),
80 ),
81 ),
82 ),
83 ),
84 ),
85 ),
86 ),
87 ),
88 ),
89 ),
90 ),
91 ),
92 ),
93 ),
94 ),
95 ),
96 ),
97 ),
98 ),
99 ),
100 ),
```



Рабочий, но неправильный

Есть альтернатива – BlocListener

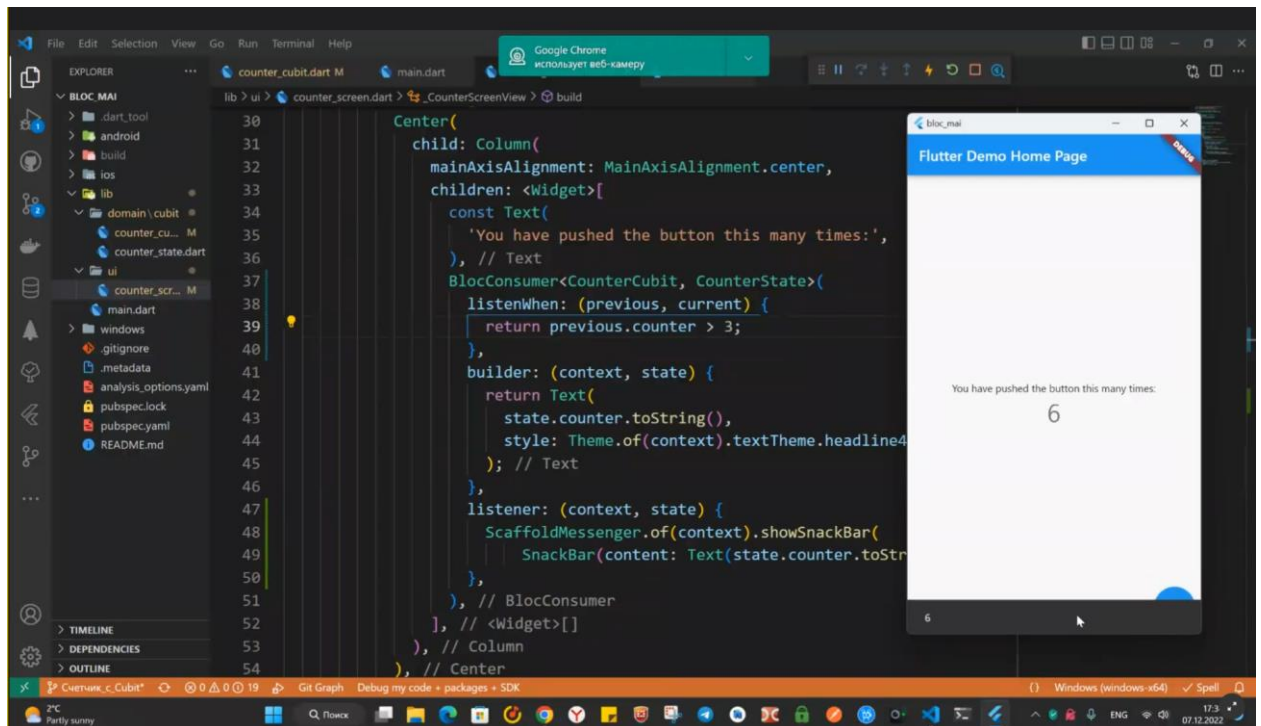


# BlocConsumer

5

- BlocConsumer предоставляет сразу listener и builder, чтобы реагировать на новые состояния. BlocConsumer аналогичен вложенным BlocListener и BlocBuilder, но уменьшает объем шаблонного кода.

```
BlocConsumer<BlocA, BlocAState>(  
  listener: (context, state) {  
    // do stuff here based on BlocA's state  
  },  
  builder: (context, state) {  
    // return widget here based on BlocA's state  
  }  
)
```



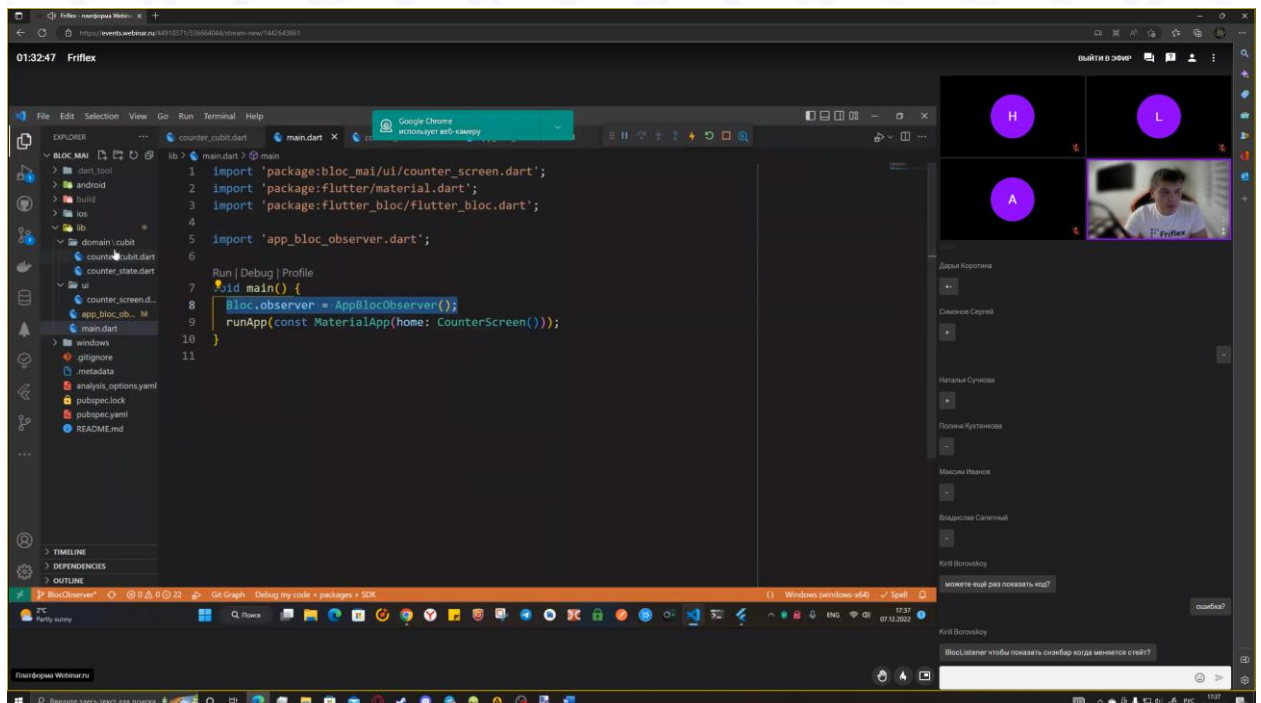


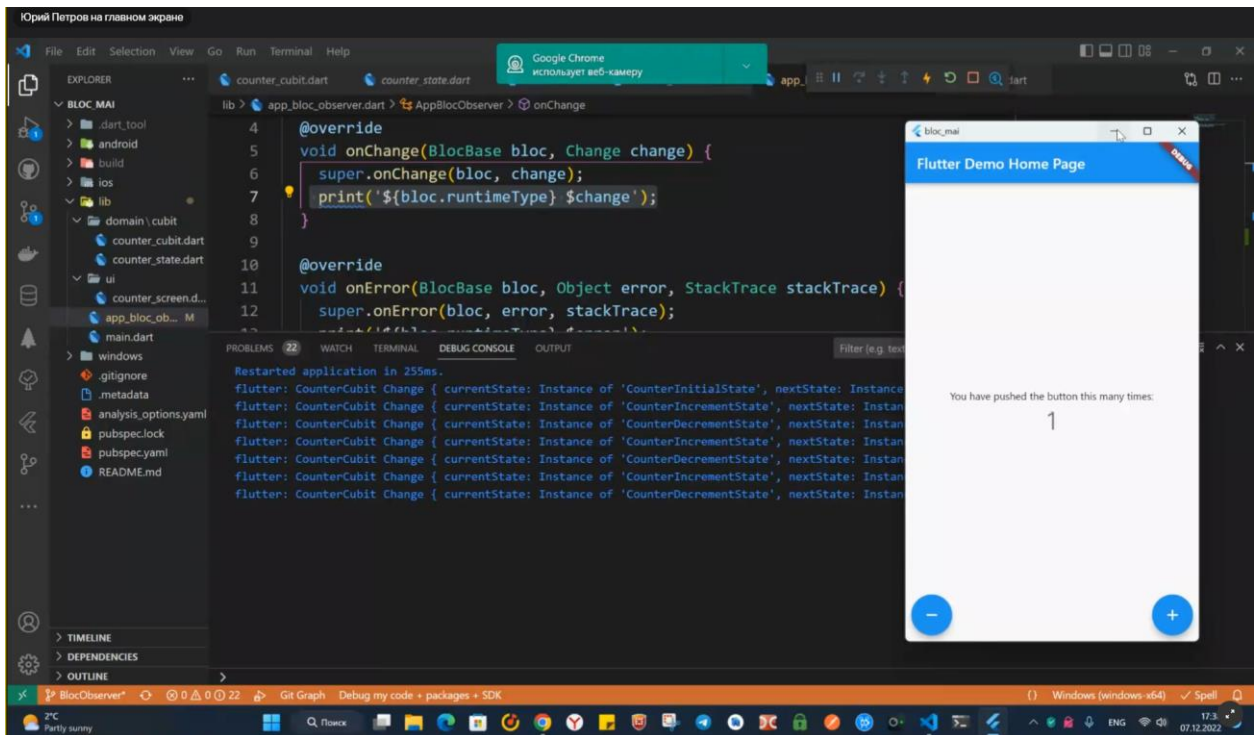
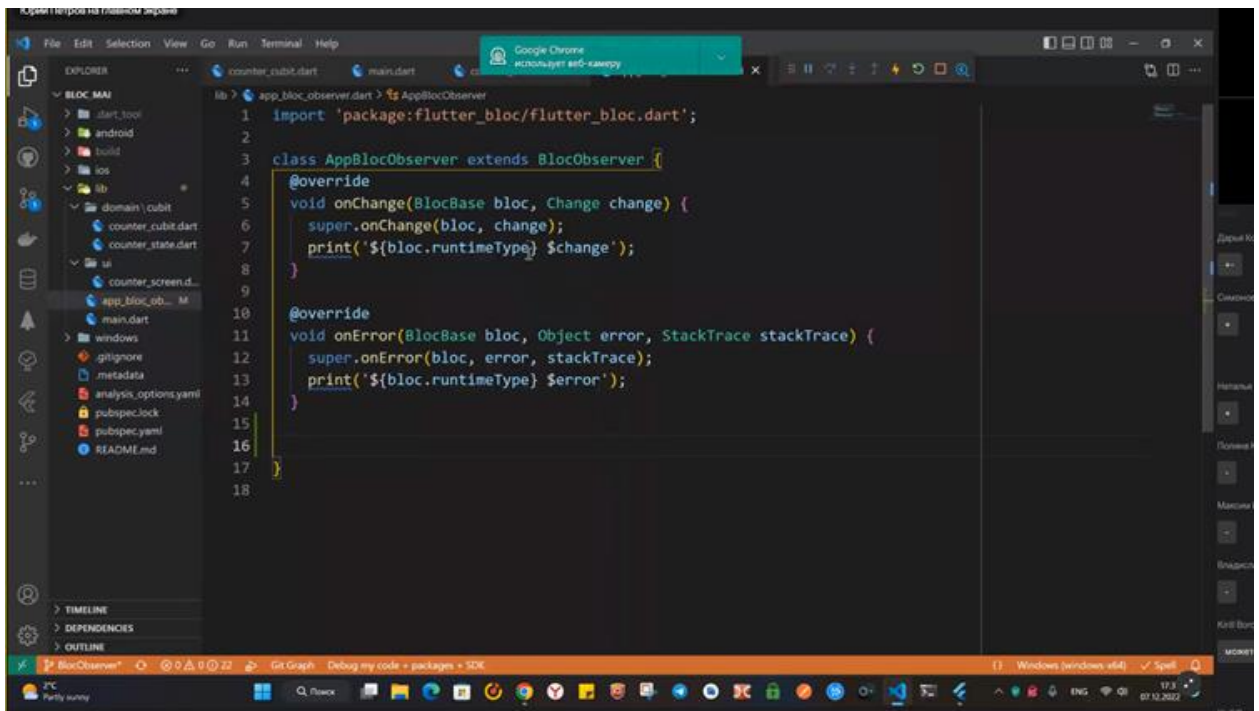
# BlocObserver

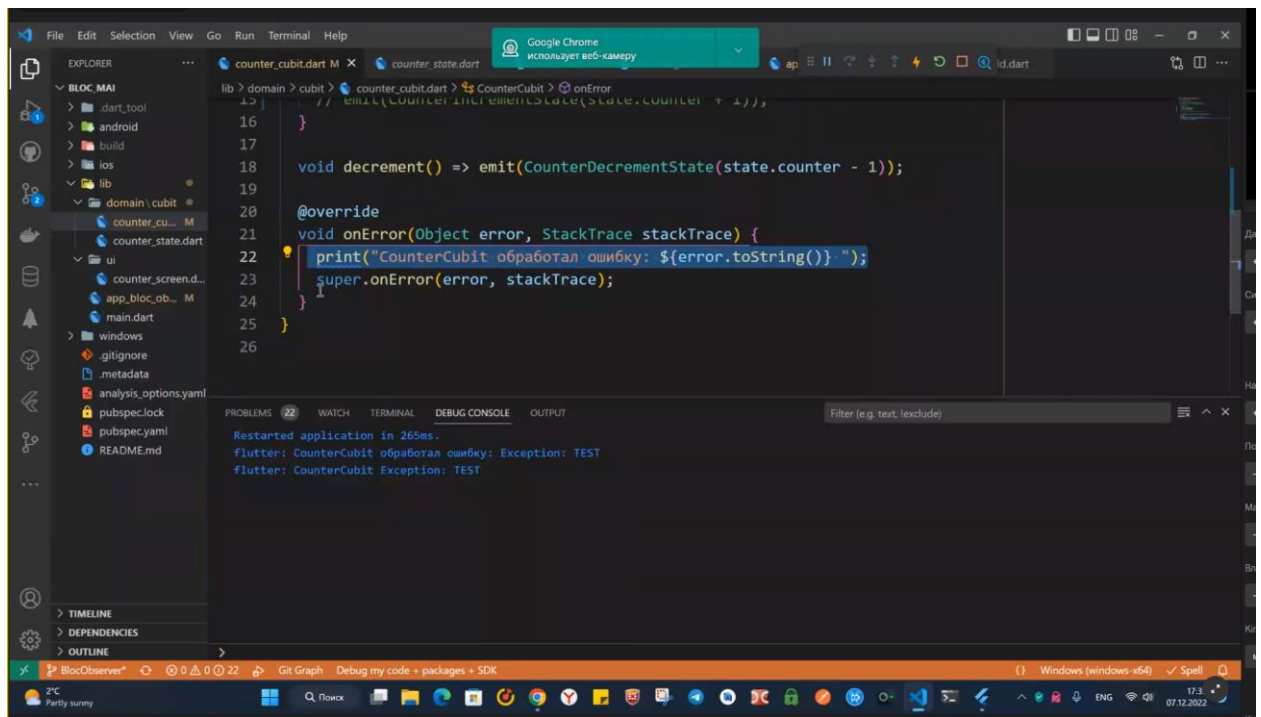
- С помощью BlocObserver мы можем обрабатывать все изменения всех блоков/кубитов в одном месте.

```
class SimpleBlocObserver extends BlocObserver {  
  @override  
  void onChange(BlocBase bloc, Change change) {  
    super.onChange(bloc, change);  
    print('${bloc.runtimeType} $change');  
  }  
}
```

```
void main() {  
  Bloc.observer = SimpleBlocObserver();  
  CounterCubit()  
    ..increment()  
    ..close();  
}
```







Отличие Flutter от MVC

Cubit-роутер смысла пока создавать нет – Auto\_route

ConduitCoreRM

Онлайн-платежи: Обзор, Аспекты безопасности и подводные камни.  
Интеграция на примере Flutter и Stripe.

Платёжная система


## Интеграция со Stripe



































flutter\_stripe 7.0.0

Published 21 days ago • @ flutterstripe.io Not safety

SDK FLUTTER PLATFORM ANDROID IOS WEB 692

Readme Changelog Example Installing Versions Scores

 **Flutter Stripe**

pub v7.0.0                                    

## Вариант интеграции: SDK / SDK + Stripe Elements

```
try {
    final intent = await Stripe.instance.confirmPayment(
        clientSecret,
        PaymentMethodParams.card(
            paymentMethodData: PaymentMethodData(),
        ),
    );

    return intent;
} catch (error) {
    if (error is StripeException) {
        final errorMessage =
            error.error.message ?? CustomErrorObject.defaultErrorCaption;
        throw CustomErrorObject(errorMessage);
    }

    throw CustomErrorObject(CustomErrorObject.defaultErrorCaption);
}
```

```
Stripe.instance.dangerouslyUpdateCardDetails(
    CardDetails(
        number: '4242424242424242',
        cvc: '123',
        expirationMonth: 12,
        expirationYear: 25
    ),
);
```



## Как протестировать интеграцию без бэкенда?

```
@Injectable(as: PaymentsRepository)
class MockPaymentsRepository implements PaymentsRepository {
    @override
    Future<CreatePaymentResult> createPayment({
        required int subscriptionId,
        required int cityId,
    }) async {
        return CreatePaymentResult(
            clientSecret: 'pi_3Luns6IeSIIdLVBri2v3t3CCB_secret_KiFixwJVpLCCWSjnRIsbXJ95t',
            orderId: '73294612'
        );
    }
}
```






# Stripe. Удобства для интеграции.

- Автоматическая обработка 3DSecure.
- Customers. Автоматическая защита от использования платежных средств одного пользователя для оплаты чего-либо, кроме заказов этого пользователя
- Stripe Elements и SDK для Flutter. Автоматическая валидация данных карты в Elements виджете. Автоматическое определение бренда карты.
- Удобные и подробные сообщения об ошибках, которые можно найти в StripeException.
- Полное сокрытие данных карты от клиентского кода при использовании Elements.
- Интеграция с pay plugin для использования систем электронных платежей.



## Системы электронных платежей. Pay plugin.

pay 1.0.11 

Published 2 months ago • @ google.dev [full safety](#)

[SDK](#) | [FLUTTER](#) | [PLATFORM](#) | [ANDROID](#) | [IOS](#) 597

---

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

---

**pub.dev**

A plugin to add payments to your Flutter application.

Platform Support

Android iOS

Google Pay Apple Pay

Getting started

Before you start, create an account with the payment providers you are planning to support and follow the setup instructions:

Apple Pay:

1. Take a look at the [integration requirements](#).
2. Create a [merchant identifier](#) for your business.
3. Create a [payment processing certificate](#) to encrypt payment information.

597 140 97%

LIKES PUBPOINTS POPULARITY

Publisher

@ google.dev

Metadata

A plug-in to add support for payments on Flutter applications.

[Repository \(GitHub\)](#)

Documentation

[API reference](#)

License

@ Apache-2.0 ([LICENSE](#))

Dependencies

flutter,  
flutter\_localizations, meta,  
pay\_android, pay\_ios,  
pay\_platform\_interface



# Google Pay. Настройка.

```
1 {
2   "provider": "google_pay",
3   "data": {
4     "environment": "TEST",
5     "apiVersion": 2,
6     "apiVersionMinor": 0,
7     "allowedPaymentMethods": [
8       {
9         "type": "CARD",
10        "tokenizationSpecification": {
11          "type": "PAYMENT_GATEWAY",
12          "parameters": {
13            "gateway": "stripe",
14            "stripe:version": "2018-10-31",
15            "stripe:publishableKey": "***"
16          }
17        },
18        "parameters": {
19          "allowedCardNetworks": ["VISA", "MASTERCARD"],
20          "allowedAuthMethods": ["PAN_ONLY", "CRYPTOGRAM_3DS"],
21          "billingAddressRequired": true,
22          "billingAddressParameters": {
23            "format": "FULL",
24            "phoneNumberRequired": true
25          }
26        }
27      }
28    ],
29    "merchantInfo": {
30      "merchantId": "***",
31      "merchantName": "MyMerchant"
32    },
33    "transactionInfo": {
34      "countryCode": "US",
35      "currencyCode": "USD"
36    }
37  }
38 }
```

## Google Pay. Настройка.

```
assets:
- assets/payment_profile_google_pay.json
- assets/payment_profile_apple_pay.json
```

```
GetIt.instance.registerSingleton<Pay>(
  Pay.withAssets([
    'payment_profile_google_pay.json',
    'payment_profile_apple_pay.json',
  ]),
);
```

```
googlePayAvailable = await payClient.userCanPay(PayProvider.google_pay);
applePayAvailable = await payClient.userCanPay(PayProvider.apple_pay);

final PaymentsCubit paymentsCubit = getIt();
paymentsCubit.setAvailablePaymentSystems(
  isGooglePayAvailable: googlePayAvailable,
  isApplePayAvailable: applePayAvailable,
);
```

## Google Pay. Использование.

```
result = await pay.showPaymentSelector(
  provider: PayProvider.google_pay,
  paymentItems: paymentItems,
);

final token = result['paymentMethodData']['tokenizationData']['token'];

final tokenJson = Map.castFrom(jsonDecode(token));

final params = PaymentMethodParams.cardFromToken(
  paymentMethodData: PaymentMethodDataCardFromToken(
    token: tokenJson['id'],
  ),
);
```

```
await stripeFacade.confirmPayment(
  result.clientSecret,
  params,
);
```

## Заключение

- Вы должны обращаться с данными карты максимально аккуратно. Признак хорошей интеграции - вы даже не видите данных карты и не имеете к ним доступа.
- Никогда не передавайте стоимость заказа на backend. Передавайте только позиции, которые пользователь хочет видеть в своем заказе.
- По возможности - используйте оплату по ссылке. Таким образом вы максимально делегируете вопрос безопасности платежному шлюзу.
- Обезличенные данные карты (такие как masked pan, brand) - можно и нужно использовать для отображения в UI, чтобы пользователь мог различать карты в списке сохраненных карт.