Генерация стабов для тестирования микросервисов, связанных по gRPC

Кирилл Шувалов

Старший инженер-программист

- Перешёл на GO из C/C++
- Занимаюсь написанием сервисов

Зачем

Что хотели сделать

Схема запроса и стаба

Использование дженериков

Использование интерфейсов

Генерация кода

Результаты

Нужно тестировать сервис в собранном виде

```
type ElementConfigGetter interface {
   Get(ctx context.Context, ID int64) (Config, error)
}
```

```
getterMock.EXPECT().Get(contextAny, int64(0)).Return(config, nil)
```

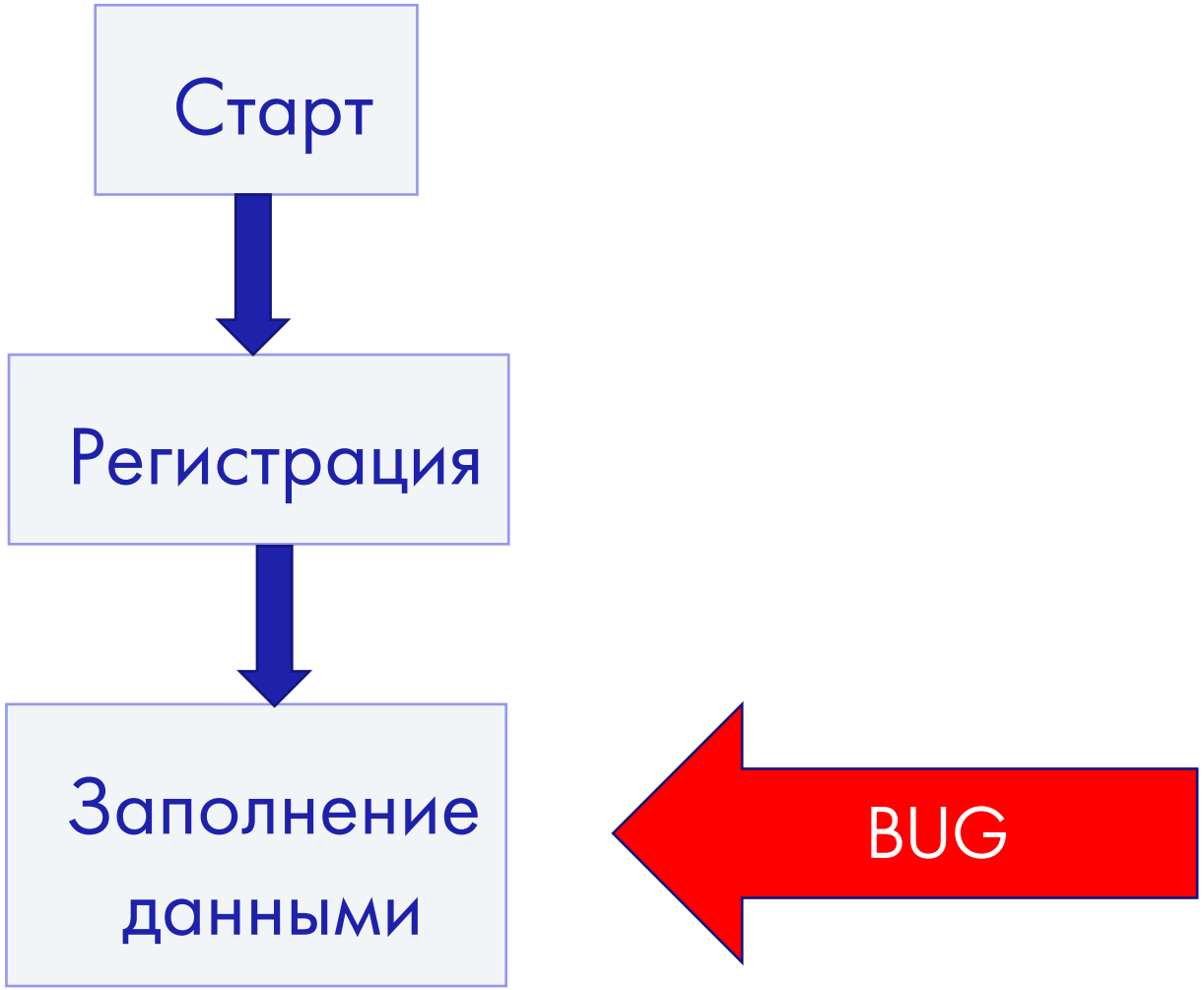
Проверка заполнения общих данных

```
map[string]string{"name": "addr"}

ctx = WithValue(ctx, key, value)
```

Библиотеки работающие с внешним миром

Зафиксировать поведение сервиса

Иметь возможность имитировать сложное поведение



Зачем

Что хотели сделать

Схема запроса и стаба

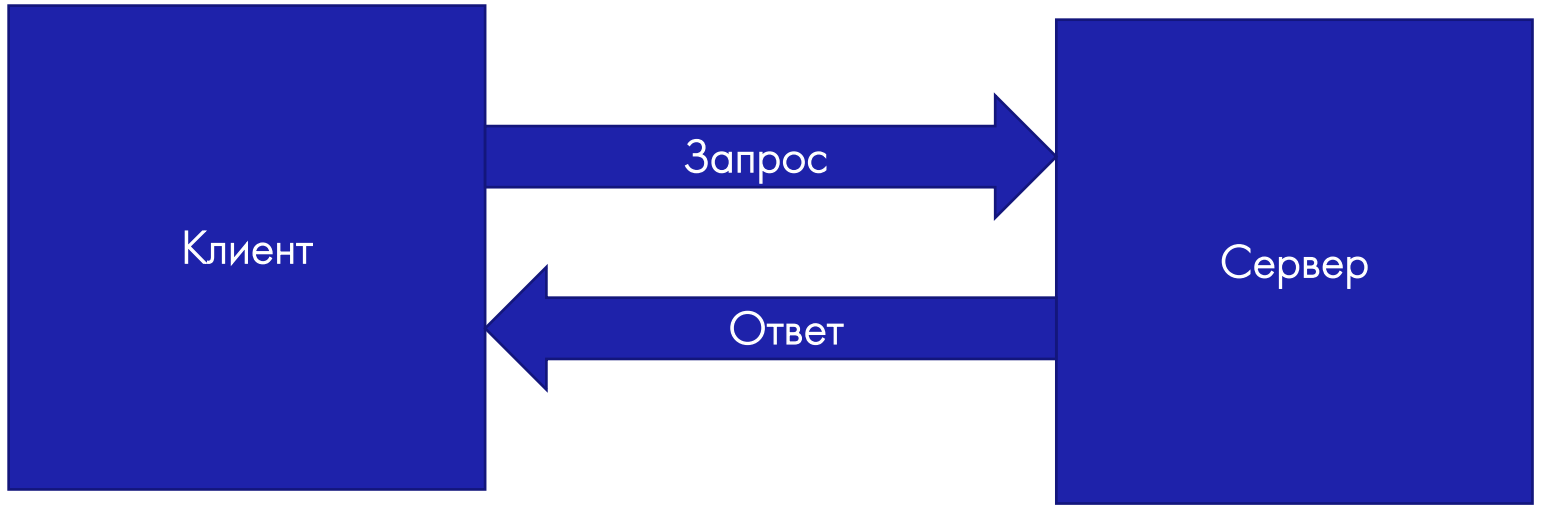
Использование дженериков

Использование интерфейсов

Генерация кода

Результаты

Немного про grpc



Немного про grpc

```
service DealingCards {
  rpc DealingTask (Dealing) returns (stream PlayingCard) {}
  rpc Swap (PlayingCard) returns (PlayingCard) {}
}
```

Немного про grpc

```
protoc
  --go_out=./
  --go-grpc_out=./
  --go_opt=module=${GO_MODULE_PREFIX}
  --go-grpc_opt=module=${GO_MODULE_PREFIX}
  ${file}
```

Как gRPC обменивается данными

Клиент отправляет	Сервер отвечает
Сообщение	Сообщение
Сообщение	Стрим
Стрим	Сообщение
Стрим	Стрим

Требования простота написания и чтения

```
card, err := client.DealingTask(context.TODO(), &poker.Dealing{Deck: 1})
Expect(err).Should(Succeed())
Expect(card).Should(Equal(cardExpect))
```

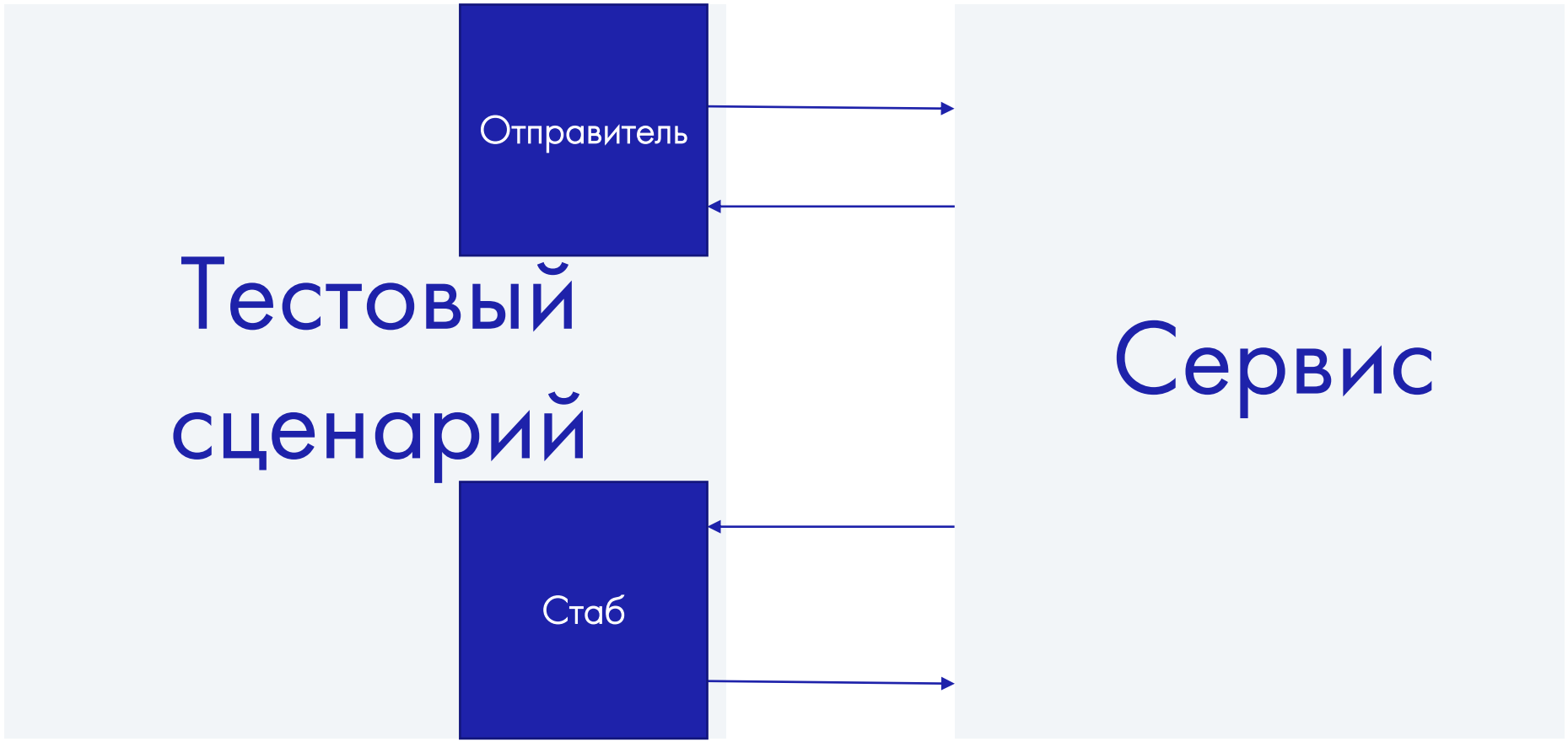
Требования минимум кода

```
type DealingCardsServer interface {
   DealingTask(*Dealing, DealingCards_DealingTaskServer) error
   Swap(context.Context, *PlayingCard) (*PlayingCard, error)
   mustEmbedUnimplementedDealingCardsServer()
}
```

Требования жёсткая типизация

```
 [FAILED] Unexpected call to ... *mocks…because:
  expected call at /go/src/src/..._test.go:224 doesn't match the
argument at index 2.
  Got: 1 (domain.DataType)
  Want: is equal to 0x16a9e3c (*domain.DataType)
```

Должен делать



Зачем

Что хотели сделать

Схема запроса и стаба

Использование дженериков

Использование интерфейсов

Генерация кода

Результаты

Схема отправки запроса

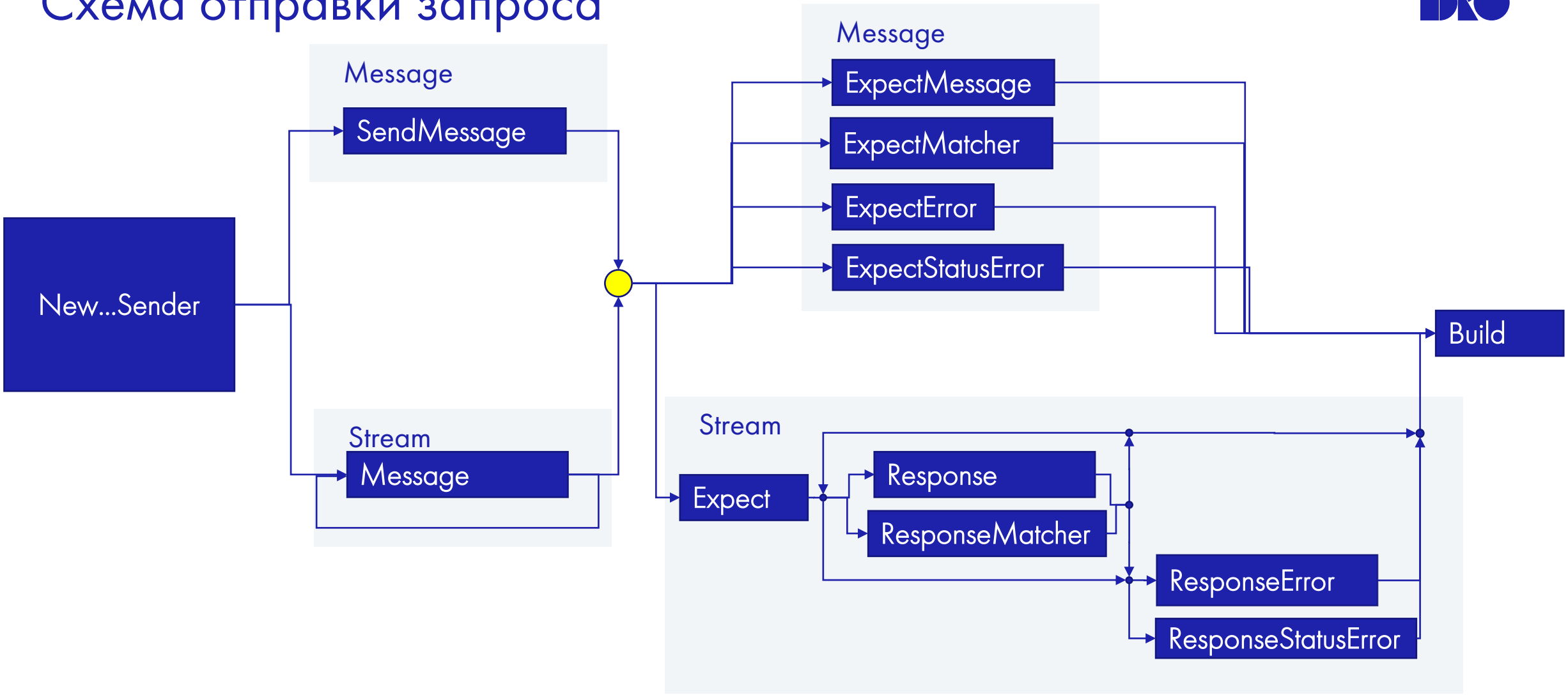
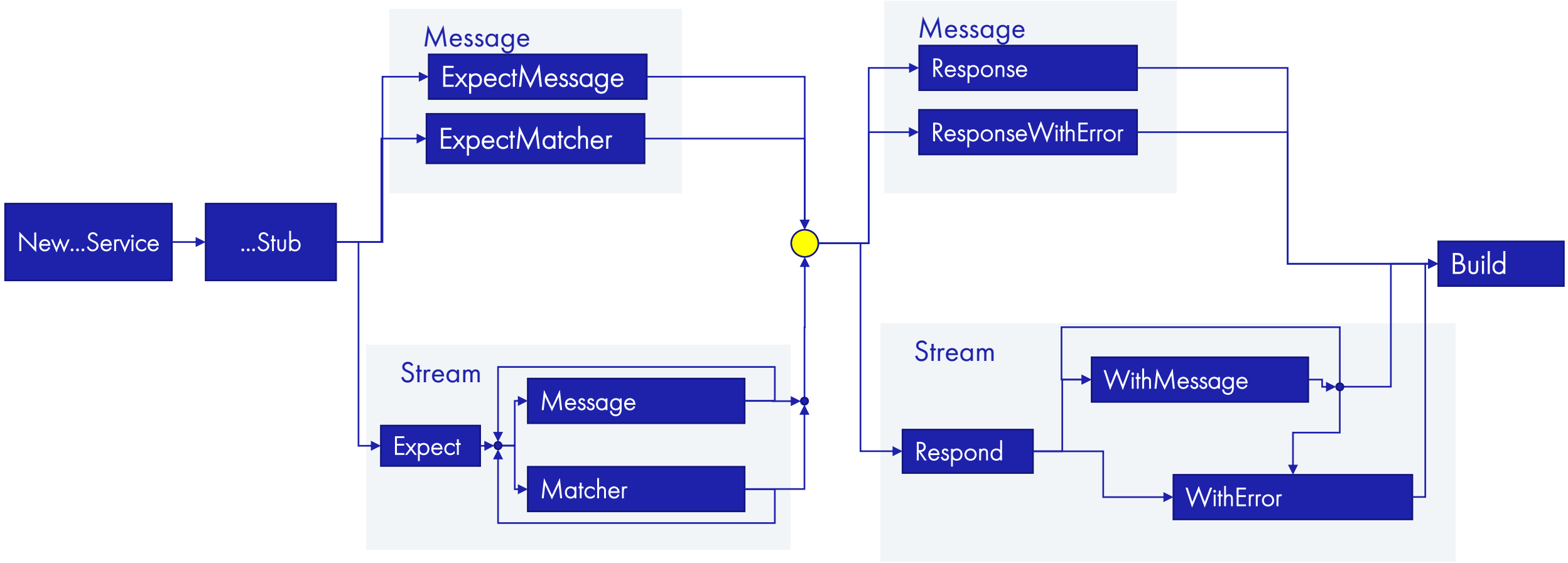


Схема stub


Message
ExpectMessage
ExpectMatcher
Message
Response
ResponseWithError
New...Service
...Stub
Build
Stream
Message
Expect
Matcher
Stream
WithMessage
Respond
WithError

Зачем

Что хотели сделать

Схема запроса и стаба

Использование дженериков

Использование интерфейсов

Генерация кода

Результаты

Generic в Go

```
type senderStreamExpectMessageInterface[TI, TO proto.Message] interface {
   Response(resp TI) senderStreamExpectMessageInterface[TI, TO]
   ResponseError() buildInterface
   ResponseStatusError(status *status.Status) buildInterface
   ResponseMatcher(matcher MatcherInterface[TI]) senderStreamExpectMessageInterface[TI, TO]
   buildInterface
}
```

Зачем

Что хотели сделать

Схема запроса и стаба

Использование дженериков

Использование интерфейсов

Генерация кода

Результаты

Interface

```
type senderStreamExpectMessageInterface[TI, TO proto.Message] interface {
   Response(resp TI) senderStreamExpectMessageInterface[TI, TO]
   ResponseError() buildInterface
   ResponseStatusError(status *status.Status) buildInterface
   ResponseMatcher(matcher MatcherInterface[TI]) senderStreamExpectMessageInterface[TI, TO]
   buildInterface
}
```

Зачем

Что хотели сделать

Схема запроса и стаба

Использование дженериков

Использование интерфейсов

Генерация кода

Результаты

Сокращаем код отправителя со стримом

```
func (c *dealingCardsClient) DealingTask(ctx context.Context, in *Dealing, opts
...grpc.CallOption) (DealingCards_DealingTaskClient, error) {
   stream, err := c.cc.NewStream(ctx, &DealingCards_ServiceDesc.Streams[0],
DealingCards_DealingTask_FullMethodName, opts...)
   if err != nil {
      return nil, err
   }
   x := &dealingCardsDealingTaskClient{stream}
   if err := x.ClientStream.SendMsg(in); err != nil {
      return nil, err
   }
   if err := x.ClientStream.CloseSend(); err != nil {
      return nil, err
   }
   return x, nil
}
```

Сокращаем код отправителя без стримов

```
func (c *dealingCardsClient) Swap(ctx context.Context, in *PlayingCard, opts
...grpc.CallOption) (*PlayingCard, error) {
   out := new(PlayingCard)
   err := c.cc.Invoke(ctx, DealingCards_Swap_FullMethodName, in, out, opts...)
   if err != nil {
      return nil, err
   }
   return out, nil
}
```

Глубины Invoke

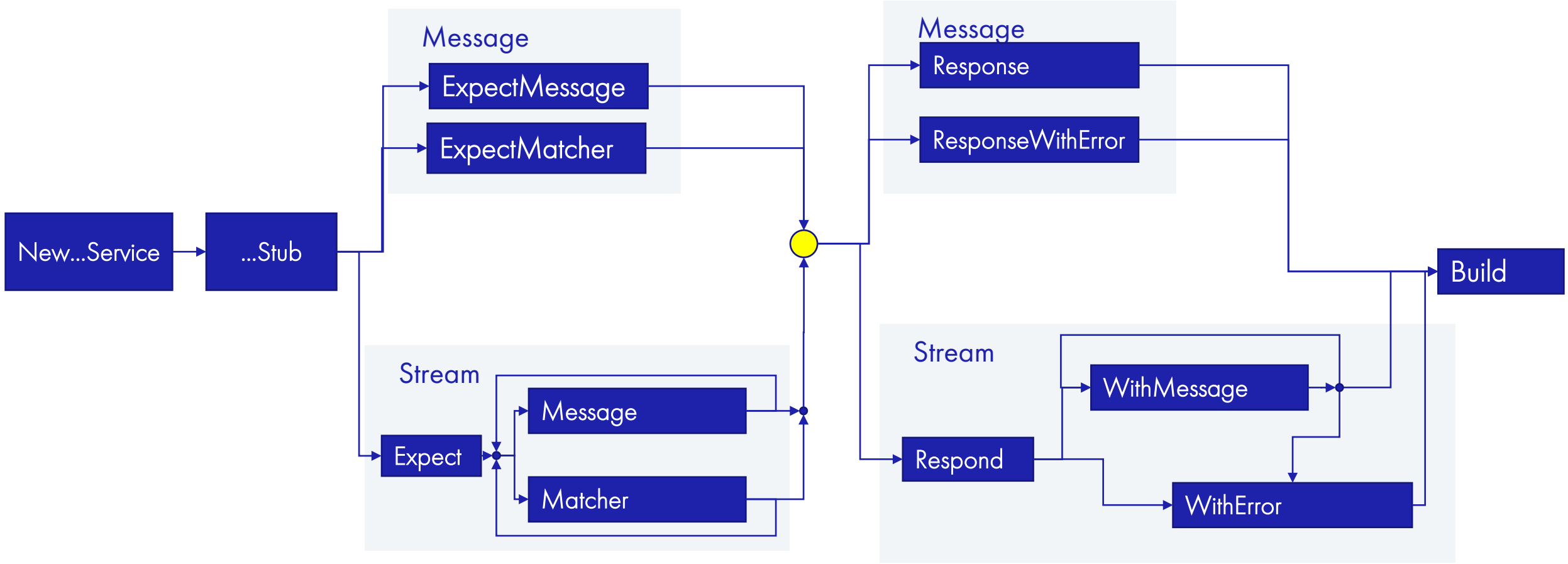
```
func invoke(ctx context.Context, method string, req, reply any, cc *ClientConn,
opts ...CallOption) error {
        cs, err := newClientStream(ctx, unaryStreamDesc, cc, method, opts...)
        if err != nil {
                return err
        }
        if err := cs.SendMsg(req); err != nil {
                return err
        }
        return cs.RecvMsg(reply)
}
```

Работа stub

```
func (s *Stub[TI, TO, E]) Match(req TI) (TO, error) {
…
}

func (s *Stub[TI, TO, E]) MatchResponseStream(req TI, stream grpc.ServerStream) error {
…
}

func (s *Stub[TI, TO, E]) MatchStreamToStream(stream StreamStream[TI, TO]) error {
…
}

func (s *Stub[TI, TO, E]) MatchStreamToMessage(stream StreamMessage[TI, TO]) error {
…
}
```

Альтернатива для каждого сервиса собирать свой grpc.ServiceDesc

Как сделать переход



Реализация переход

```
// SenderInterface interface for sending messages
type SenderInterface[TI, TO proto.Message, E any] interface {
   SendMessage(request TO) E
}

// SenderFirtsStreamInterface the first interface in the cycle of adding
requests to the stream
type SenderFirtsStreamInterface[TI, TO proto.Message, E any] interface {
   Message(TO) E
}
```

Что нужно, чтоб создать отправитель.

```
func NewDealingTaskSender(conn grpc.ClientConnInterface) grpc_checker.SenderInterface[
   *message.PlayingCard,
   *message.Dealing,
   grpc_checker.SenderStreamExpectInterface[
      *message.PlayingCard,
      *message.Dealing,
   ],
] {
   return &grpc_checker.Sender[
      *message.PlayingCard,
      *message.Dealing,
      grpc_checker.SenderStreamExpectInterface[
         *message.PlayingCard,
         *message.Dealing,
      ],
   ]{
      Method: message.DealingCards_DealingTask_FullMethodName,
      Conn: conn,
      StreamDesc: &grpc.StreamDesc{ServerStreams: true, ClientStreams: false},
   }
}
```

Что нужно, чтоб создать stub.

```
type DealingCards struct {
   StubStreamDealingTask grpc_checker.StubMessageToStreamInterface[
      *message.Dealing,
      *message.PlayingCard,
      grpc_checker.StubStreamRespondInterface[
         *message.Dealing,
         *message.PlayingCard,
      ],
   ]
   message.UnimplementedDealingCardsServer
}

func (service *DealingCards) DealingTask(
   msg *message.Dealing,
   stream message.DealingCards_DealingTaskServer,
) (
   error,
) {
   return service.StubStreamDealingTask.MatchResponseStream(msg, stream)
}
```

Инструмент проверки стаба.

```
type Interface interface {
        After(preStub ...Interface)
        CheckCall() error
        Wait(WithTimeout time.Duration, WithPolling time.Duration)
        Times(uint32)
        MaxTimes(uint32)
        MinTimes(uint32)
}
```

protoc

```
protoc /
   --plugin=protoc-gen-custom-plugin=tools/plugins/mt_test_grpc_stub_generation/plugin.py /
   --custom-plugin_out=./ /
    …
```

Зачем

Что хотели сделать

Схема запроса и стаба

Использование дженериков

Использование интерфейсов

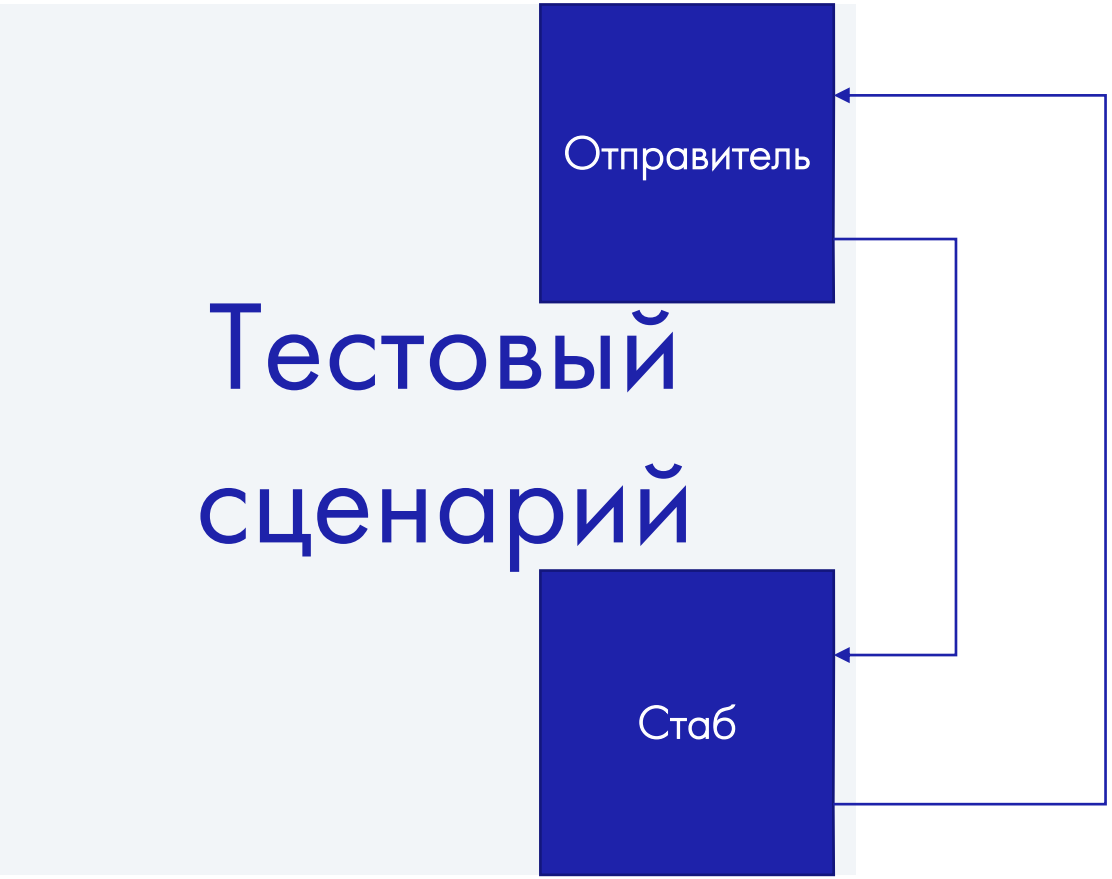
Генерация кода

Результаты

Сервис gRPC

```
service Game {
  rpc Move (stream PlayingCard) returns (stream TopCard) {}
}
```

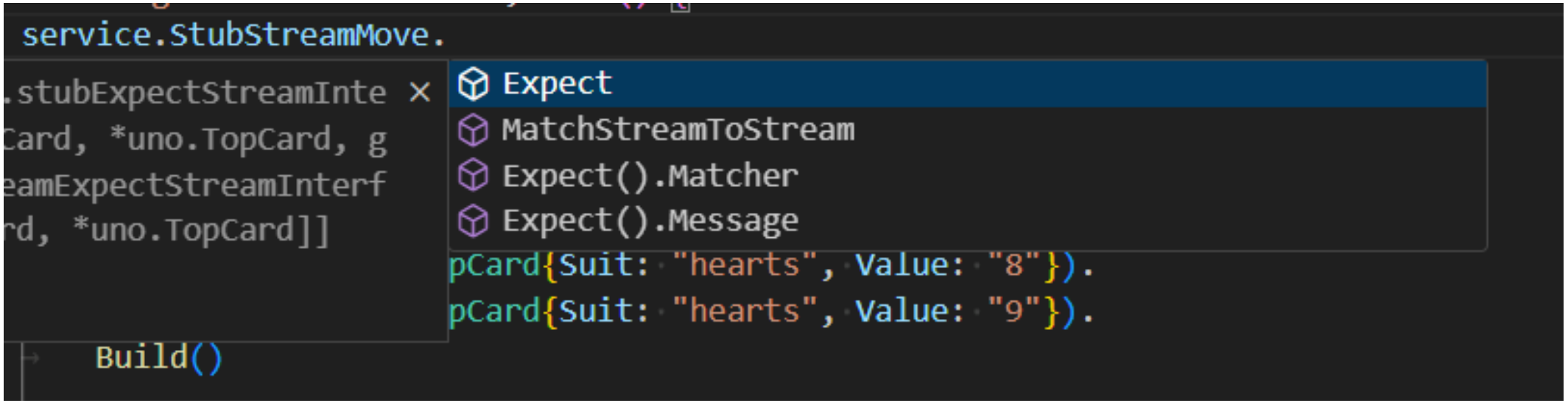
Тестовый сценарий



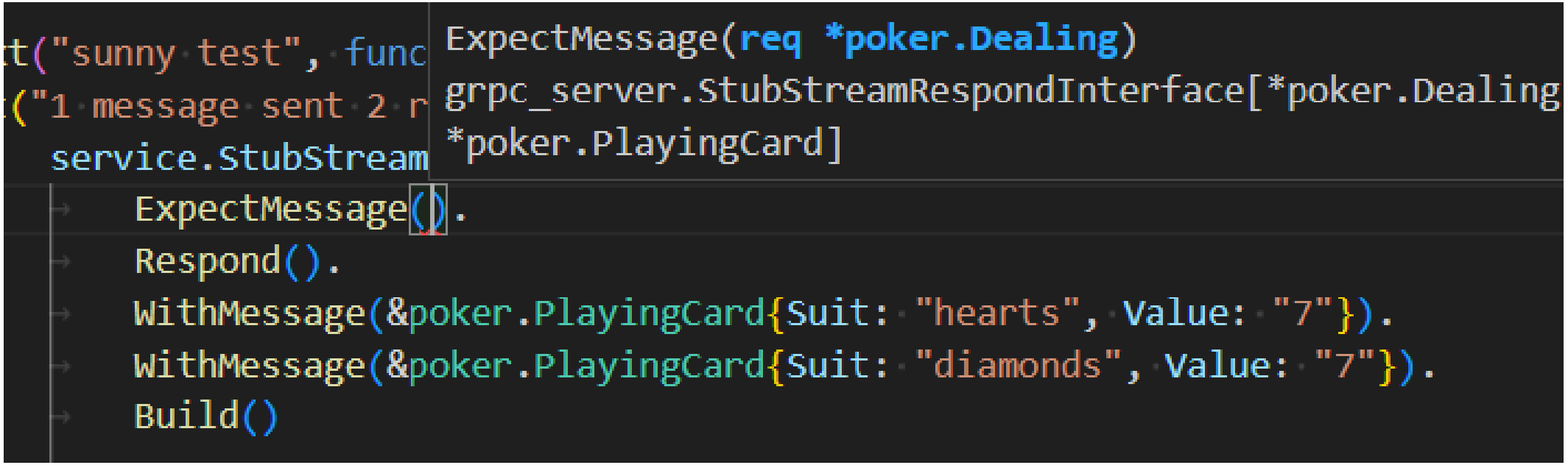
Тестовый сценарий

```
service.StubStreamMove.
   Expect().
   Message(&uno.PlayingCard{Suit: "hearts", Value: "6"}).
   Message(&uno.PlayingCard{Suit: "hearts", Value: "7"}).
   Respond().
   WithMessage(&uno.TopCard{Suit: "hearts", Value: "8"}).
   WithMessage(&uno.TopCard{Suit: "hearts", Value: "9"}).
   Build()

uno_stub.NewMoveSender(ClientConn).
   Message(&uno.PlayingCard{Suit: "hearts", Value: "6"}).
   Message(&uno.PlayingCard{Suit: "hearts", Value: "7"}).
   Expect().
   Response(&uno.TopCard{Suit: "hearts", Value: "8"}).
   Response(&uno.TopCard{Suit: "hearts", Value: "9"}).
   Build()
```

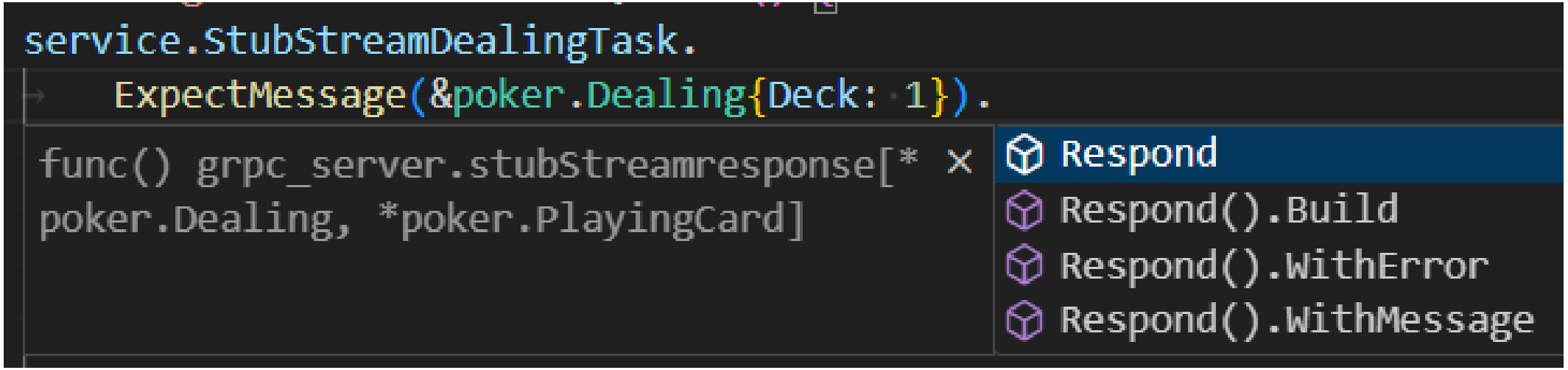
Примеры работы



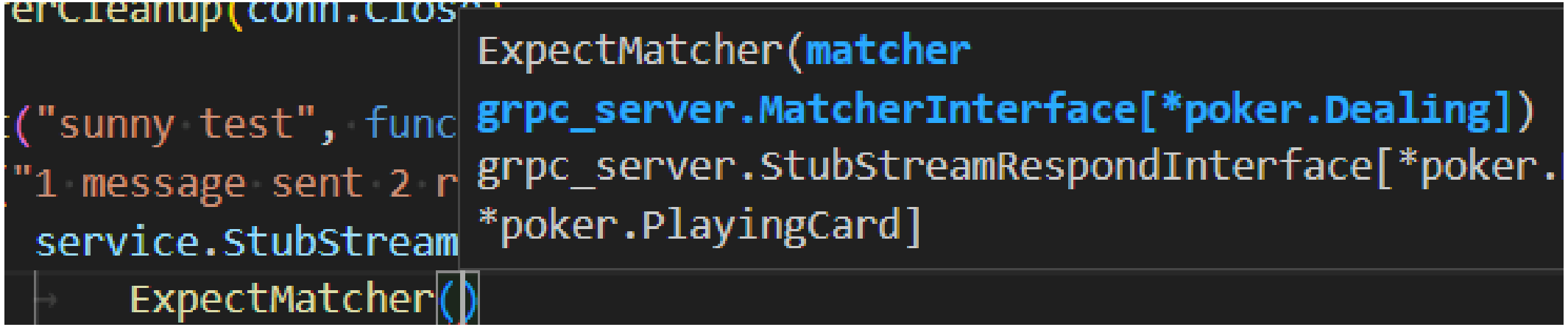
Примеры работы



Примеры работы



Что не нравится



Что не нравится

Никак не могу сесть и написать развёрнутые комментарии к методам интерфейсов, чтоб они всплывали в подсказке.

Что не нравится

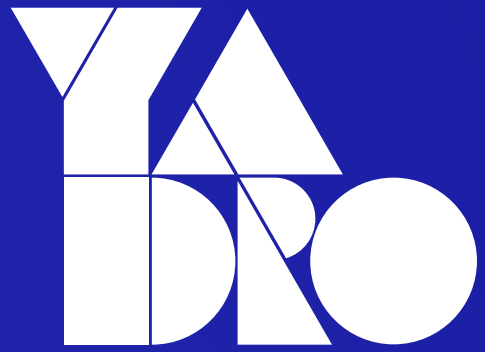
```
func NewMoveSender(conn grpc.ClientConnInterface) grpc_checker.SenderFirtsStreamInterface[
   *message.TopCard,
   *message.PlayingCard,
   grpc_checker.SenderStreamExpectStreamLoopInterface[
      *message.TopCard,
      *message.PlayingCard,
   ],
] {
   return &grpc_checker.Sender[
      *message.TopCard,
      *message.PlayingCard,
      grpc_checker.SenderStreamExpectStreamLoopInterface[
         *message.TopCard,
         *message.PlayingCard,
      ],
   ]{
      Method: message.Game_Move_FullMethodName,
      Conn: conn,
      StreamDesc: &grpc.StreamDesc{ServerStreams: true, ClientStreams: true},
   }
}
```

Плагин уже успешно используется в проекте

Стало больше тестов сервисов и покрытия и проще симулировать баги, что помогает их решению.

Выводы

Не все задачи в проекте стоит решать с помощью Go. Иногда стоит отойти от Go и реализовать это с помощью других инструментов. В текущей работе использовался плагин для protoc.



Москва,
ул. Рочдельская, 15, стр. 13
+7 800 777-06-11
yadro.com