

Простоту охота навести: как легко
тестировать клиент-серверные
взаимодействия на примере
WebSocket

Попов Максим
SberUser: АQA внутренних
продуктов Сбера



Занимаюсь созданием
и поддержкой
фреймворка
автотестирования + НТ



Люблю дайвинг



На фото планирую
погружение в этой
прекрасной речке



Чем мы занимаемся?

SCPL



Простыми словами

Что можно сделать на платформе?



Входящие взаимодействия



Исходящие взаимодействия

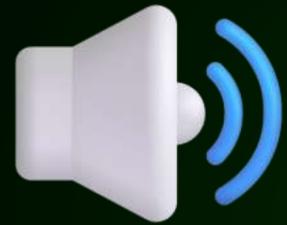


Трансферы звонков



Чаты и задачи оператора

Что тестирую я?



**Инициация
звонка**

**Окончание
звонка**



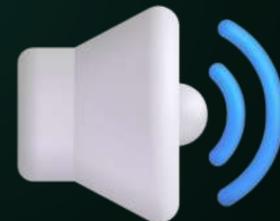
**Статусы
оператора**

**Ответ
на звонок**



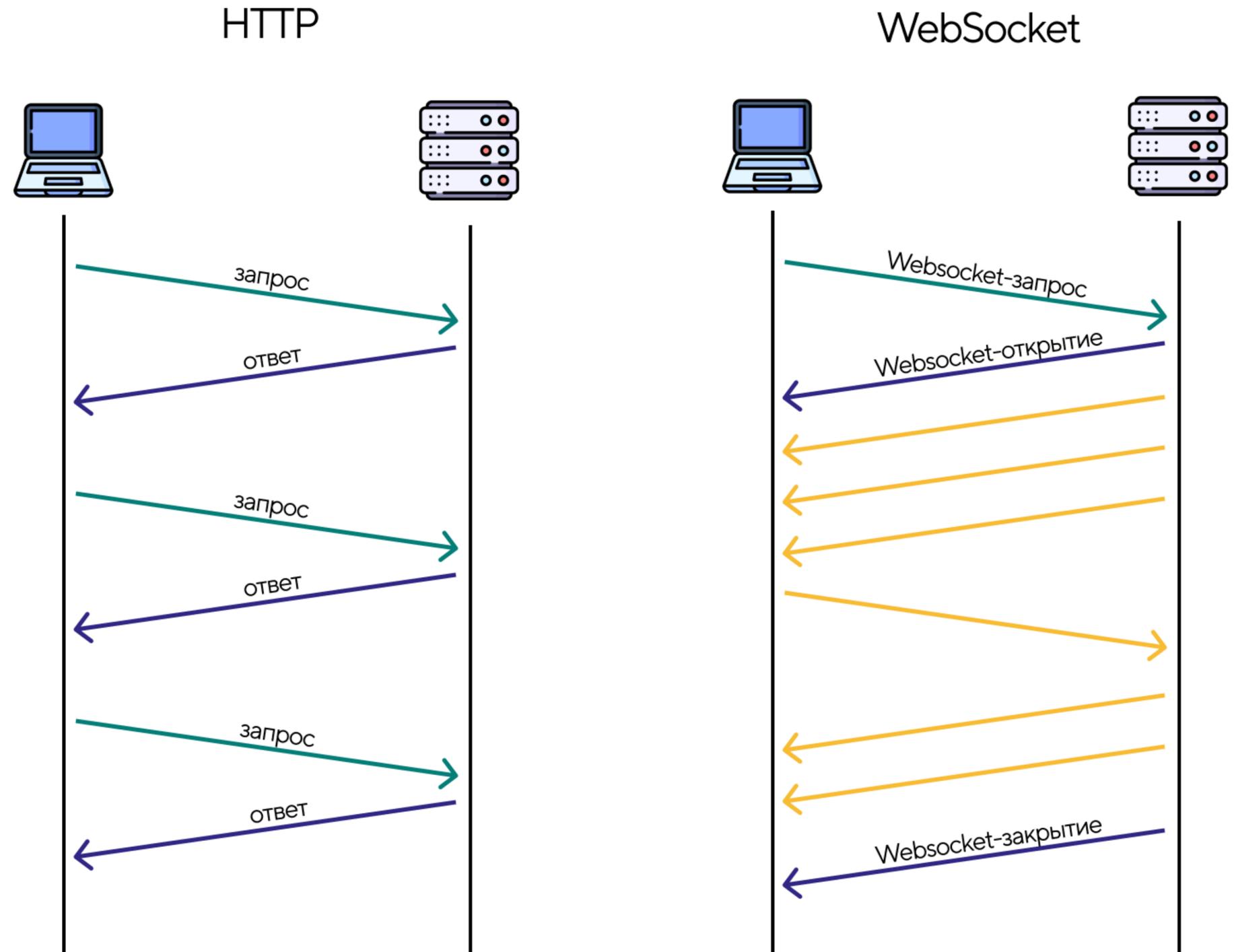
**Функции
во время
звонка**

**Перевод
звонков**



Что такое WebSocket?

- Двухсторонняя отправка
- Мгновенная передача данных
- Открытое соединение
- Real-time использование



Как это внутри?

newOutInteraction
startNewOutInteraction
acceptRtc



Сервер

Клиент

resultNewOutInteraction
resultAcceptRtc
resultStartNewOutInteraction



1 "action": "newOutInteraction",
"destination": "890000002"

2 "action": "resultNewOutInteraction",
"interactionID": "241a07df-5527c23fd6ea",
"workitemID": "6e5ae693-bdc3671a26e7",
"result": "success",
"destination": "890000002"

3 "action": "startNewOutInteraction",
"workitemID": "6e5ae693-bdc3671a26e7"

4 "action": "acceptRtc",
"data": {"Много медиа данных"}



5 "action": "resultAcceptRtc",
"interactionID": "241a07df-5527c23fd6ea",
"result": "success"

6 "action": "resultStartNewOutInteraction",
"workitemID": "6e5ae693-bdc3671a26e7",
"result": "success"

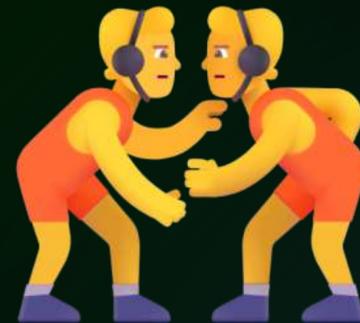
Какие нужно решать задачи?

Отправка и
получение
цепочек
сообщений



Сохранение
полученных
сообщений

Конкурентная
работа



Несколько
подключений

Не только
WebSocket

Достижение
любого состояния

Парсинг
ответов

Моментальная
реакция на ответ

Объединение
серии
запросов



Что может предложить интернет?



1) Крутая статья с хабра

2) Решение от мыслителя с ответов мейл ру

3) Как всегда поможет Stack Overflow

Что может предложить интернет?



404

Page not found

The Page you are looking for doesn't exist or another error occurred.

[Go back](#), or head over to [Scass Tech](#) to choose a new direction.

Точнее почти ничего (самый минимум)



```
def on_open(client):  
    print("Connection opened")  
    client.send("Hello, WebSocket!")  
  
def on_message(client, message):  
    print(f"Received message: {message}")  
  
def on_close(client):  
    print("Connection closed")  
  
ws = websocket.WebSocketApp(  
    "ws://example.com/ws",  
    on_open=on_open,  
    on_message=on_message,  
    on_close=on_close  
)
```

Почему мне (и скорее всего вам) этого не хватило



Непонятно что
отправить



Не управляет
явными
ожиданиям



Нет обработки
сообщений

Посоветовался с экспертами

GigaChat — русскоязычная нейросеть от Сбера

Помочь с текстом Обычный... SB S... B I U S <> ↻ 🌑 | ☰ | 🔗 😊 ⬇️ 🖨️

Заголовок

Для работы с WebSocket в Python можно использовать библиотеку websockets. Она позволяет легко создавать клиенты и серверы для взаимодействия через WebSocket протокол. Ниже приведён пример простого клиента, который подключается к серверу, слушает сообщения в одном канале и отправляет сообщения в другом.

```
import asyncio
import websockets

async def listen_for_messages(websocket):
    async for message in websocket:
        print(f"Получено сообщение: {message}")

async def send_message(websocket, channel, message):
    await websocket.send(f"{channel}:{message}")

async def main():
    uri = "ws://localhost:8765" # Укажите адрес вашего сервера

    async with websockets.connect(uri) as websocket:
        # Запускаем задачу прослушивания сообщений
        asyncio.create_task(listen_for_messages(websocket))

        while True:
            # Ожидаем ввода от пользователя
            input_channel = input("Введите канал для отправки сообщения: ")
            input_message = input("Введите сообщение: ")

            # Отправляем сообщение в указанный канал
            await send_message(websocket, input_channel, input_message)

if __name__ == "__main__":
    asyncio.run(main())
```

Реализуем
задуманное

Думаем.

Первые проблемы

Проблема 1.



Огромная вариативность
тестовых сценариев

Проблема 2.



Сложный вход
для новых членов
команды



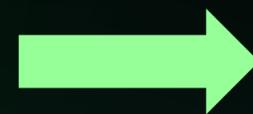
Решение



newOutInteraction

startNewOutInteraction

acceptRtc



outbound_call

Сценарий происходящего

```
def START_NEW_OUT_INTERACTION():  
    return {  
        'action': 'startNewOutInteraction',  
        'data': {  
            'destination': "",  
            'workitemID': "",  
        }  
    }  
  
def outbound_call(destination):  
    return {  
        "action": "outbound_call",  
        "data": {  
            "destination": destination,  
        }  
    }
```

The diagram consists of three red arrows and three red numbers. Arrow 1 points from the number '1' to the 'destination' field in the 'data' object of the 'START_NEW_OUT_INTERACTION()' function. Arrow 2 points from the number '2' to the 'destination' parameter of the 'outbound_call()' function. Arrow 3 points from the number '3' to the 'destination' field in the 'data' object of the 'outbound_call()' function.

Сценарий происходящего



```
def __outbound_call(self, event):  
    self.events_list.append(NEW_OUT_INTERACTION)  
  
    START_NEW_OUT_INTERACTION["destination"] = event["destination"]  
    self.events_list.append(start_new_out_interaction)  
  
    self.events_list.append(ACCEPT_RTC_EVENT)  
  
    self.events_list.append({"action": "wait_outbound_connect"})  
  
    logger.success(f"Действие добавлено в список сообщений")
```

1

2

3

4

5

```
workflow = [  
    WebSocketActions.set_ready_status(),  
    WebSocketActions.outbound_call(self.telephone),  
    WebSocketActions.agent_hangup(),  
    WebSocketActions.finish_wrapup(disposition_id),  
]
```



```
with allure.step("Собрать список технических событий"):  
    list_events = EventCollector().collect_events(workflow)  
  
with allure.step("Совершить исходящий звонок"):  
    with WsClient(user=CORE_USER_INTERACTIONS_1) as client:  
        client.launch_ws(events=list_events)
```

Замена сценарного списка workflow на технический list_events

С этим разобрались



**Сложный вход для
новых членов
команды**



**Огромная вариативность
тестовых сценариев**



Третья проблема



Проблема 3.



Регулярное появление
нового функционала на
платформе

Решение



Что-то отправляем?

В фабрику!

Что-то получили?

Тоже за станок!

Почему фабрика?

Инкапсуляция

ЛОГИКИ

Гибкость

Фабрика прикольно

звучит



Фабрики сообщений

Реакция фабрики на сообщение



с самого детства на завод

молодцы

Вызов внешнего сервиса

Вставка данных

Сохранение данных

Простановка флагов

Фабрики сообщений

```
1  
async def handle_event(self, event):
```

```
    """Метод для получения обработчика сообщения"""
```

```
2  
    events_handlers = {
```

```
        "startNewOutInteraction": self.__workitem_state_id_insert,
```

```
        "acceptInteraction": self.__call_data_insert_response,
```

```
3  
4  
5  
6  
    async def __workitem_state_id_insert(self):
```

```
        """Обработчик для вставки workitem_state_id в сообщение"""
```

```
        self.event['data']['workitemID'] = self.ws_client.workitem_state_id
```

```
        await self.ws_client.send(json.dumps(self.event))
```

```
        logger.success('Завершена обработка сообщения')
```



Фабрики сообщений

```
async def __new_workitem_state(self):  
    """Обработчик для получения interactionID, workitemID"""  
    self.ws_client.interaction_id = self.event['data']['interactionID']  
    self.ws_client.workitem_state_id = self.event['data']['workitemID']  
    self.ws_client.wait_flag = True  
    logger.success('Завершена обработка сообщения')
```



Регулярное появление нового функционала на платформе

Последняя проблема



Зажмурьтесь...

Проблема 4.
Отправка
сообщений строго
в определенный
момент



Решение

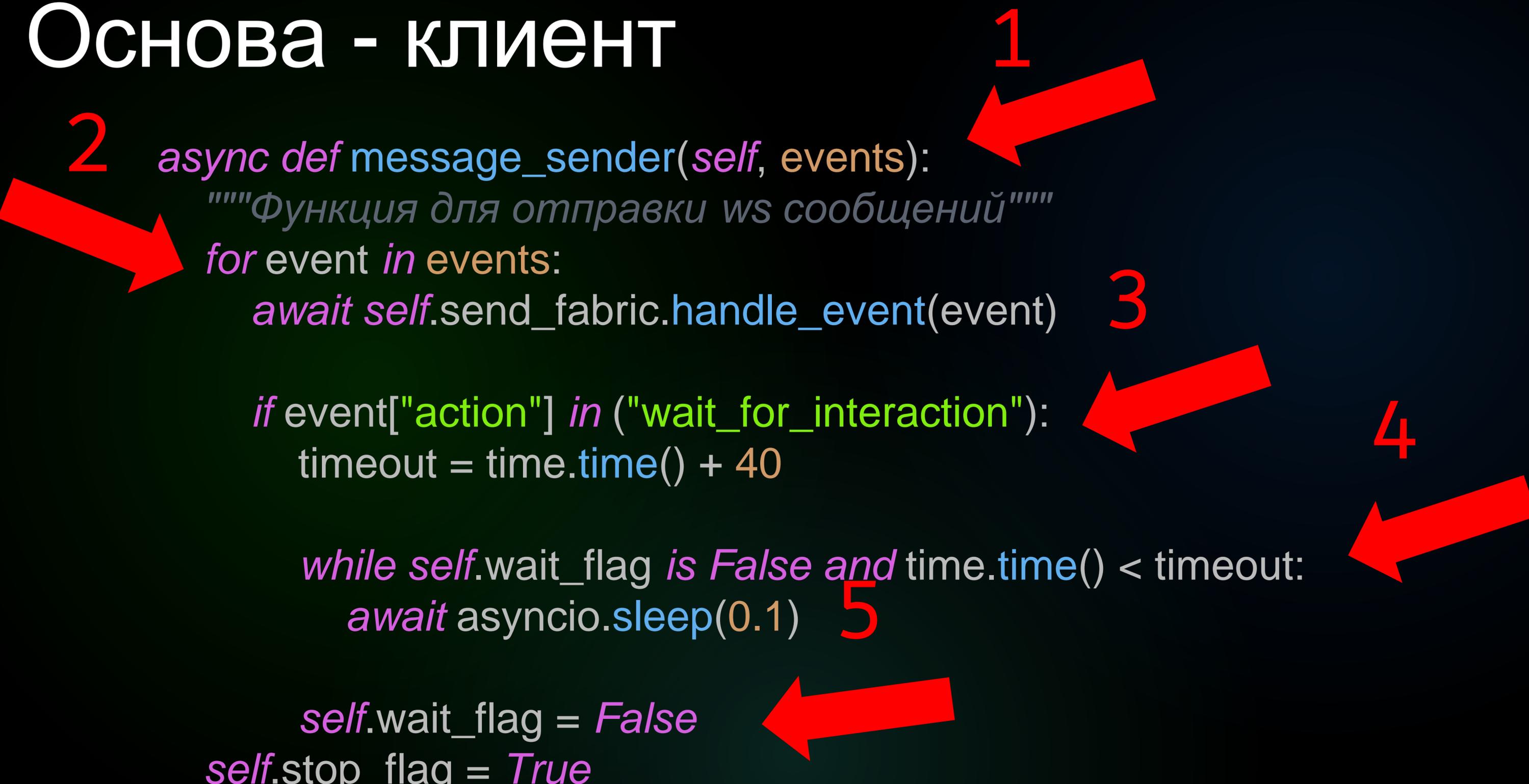


Фабрики
ставят флаги в
нужный
момент

Клиент
останавливает
отправку до
получения флагов

Основа - клиент

```
1  
2 async def message_sender(self, events):  
    """Функция для отправки ws сообщений"""  
    for event in events:  
        await self.send_fabric.handle_event(event) 3  
  
        if event["action"] in ("wait_for_interaction"):  
            timeout = time.time() + 40  
  
            while self.wait_flag is False and time.time() < timeout:  
                await asyncio.sleep(0.1) 5  
  
            self.wait_flag = False  
            self.stop_flag = True
```



The diagram consists of five red arrows pointing to specific lines of code in the Python snippet. Arrow 1 points to the function definition line. Arrow 2 points to the docstring line. Arrow 3 points to the `await self.send_fabric.handle_event(event)` line. Arrow 4 points to the `while self.wait_flag is False and time.time() < timeout:` line. Arrow 5 points to the `await asyncio.sleep(0.1)` line.

Основа - клиент



```
def __outbound_call(self, event):  
    self.events_list.append(NEW_OUT_INTERACTION)
```

```
START_NEW_OUT_INTERACTION["destination"] = event["destination"]  
self.events_list.append(start_new_out_interaction)
```

```
self.events_list.append(ACCEPT_RTC_EVENT)
```

```
self.events_list.append({"action": "wait_outbound_connect"})
```



Основа - клиент



1

```
async def message_reader(self):
```

```
    """Функция для чтения ws сообщений"""
```

```
    while not self.stop_flag:
```

```
        response = json.loads(await self.socket.recv())
```

```
        await self.recv_fabric.handle_event(response.copy())
```

2

3



Отправка сообщений в строго
определенный момент

А теперь сложный кейс (я все объясню)



```
workflow = [  
    WebSocketActions.set_ready_status(),  
    WebSocketActions.inbound_call(telephone=telephone),  
    WebSocketActions.call_function(func, param=data),  
    WebSocketActions.wait(time),  
    WebSocketActions.accept_interaction(),  
    WebSocketActions.set_mute(True),  
    WebSocketActions.set_mute(False),  
    WebSocketActions.set_hold(),  
    WebSocketActions.set_resume(),  
    WebSocketActions.call_function(func, param=data),  
    WebSocketActions.agent_hangup(),  
]  
  
list_events = EventCollector().collect_events(workflow)  
  
with WsClient(user=AUTO_USER_REPORTING_15) as client:  
    client.launch_ws(events=list_events)  
    interaction_id = client.interaction_id  
    return_values = client.return_values
```

4

6

8

10

1

2

3

5

7

9

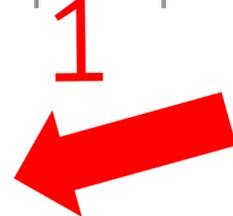
11

12

Еще не
все

Pipeline Call tool

This build requires parameters:

STAND 

ST

NUMBER_OF_CALLS
Количество звонков 

100

TELEPHONE
Номер телефона для звонка 

869547101

 Build

Cancel

Автономный запуск

Генерация данных

Создание метрик

Использование

не только

в тестировании

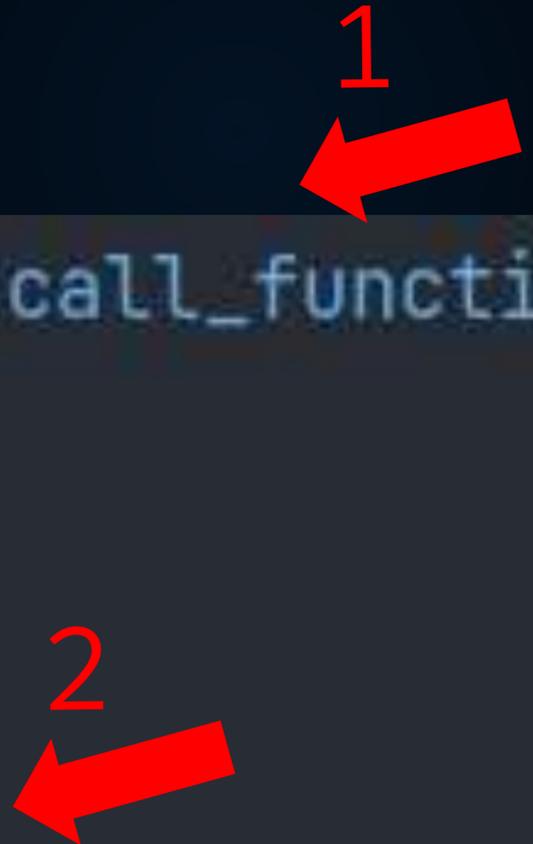
Вызов функции

в любой момент работы механизма

Универсальность

Не надо подгадывать
время

```
WebSocketActions.call_function(  
    "text1",  
    "123",  
    func=func,  
    param1=data,  
    param2=data2  
)
```

A diagram illustrating function call parameters. A red arrow labeled '1' points to the first parameter 'text1'. Another red arrow labeled '2' points to the parameter 'param1=data'.

Простой дебаг тестов



Все полученные сообщения
сохранены

Уникальные логи в обработчиках

- agentCapacity0.json
- agentCapacity1.json
- connectionStatus0.json
- newAgentState0.json
- newAgentState1.json
- newAgentState2.json
- newRtc0.json
- newWorkitemState0.json
- newWorkitemState1.json
- newWorkitemState2.json
- newWorkitemState3.json
- ping0.json
- ping1.json
- ping2.json
- ping3.json
- resultAcceptRtc0.json
- resultChangeAgentState0.json
- resultFinishWorkitem0.json
- resultGetAgentData0.json
- resultGetDictionaryData0.json
- resultHangUp0.json
- resultNewOutInteraction0.json
- resultStartNewOutInteraction0.json
- workitemPartyList0.json

ВЫВОДЫ



**Эффективный расширяемый
подход к клиент-серверным
взаимодействиям**

1

2

**Возможность использования
отдельно от тестового
фреймворка**

3

Простой дебаг тестов

Где попробовать самим (на ГитХабе)





Спасибо

ПОЖАЛУЙСТА