ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Научный руководитель  старший преподаватель  департамента больших данных и информационного поиска  факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.М. Гринкруг  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | УТВЕРЖДЕНО  Академический руководитель  образовательной программы  «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

Сервер для мокирования асинхронных протоколов

Программа и методика испытаний

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.03.07-01 51 01-ЛУ

|  |  |
| --- | --- |
|  | Исполнители  студент группы БПИ152  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /П.И. Данилин/  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2019 г. |

УТВЕРЖДЕН

RU.17701729. 03.07-01 51 01-1-ЛУ

Сервер для мокирования асинхронных протоколов

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

Программа и методика испытаний

RU.17701729.03.07-01 51 01-ЛУ

Листов 22

# Аннотация

В данном программном документе приведена программа и методика испытаний для программного обеспечения «Сервер для мокирования асинхронных протоколов». В разделе «Объект испытаний» указаны наименование программы, а также краткая характеристика области применения. Далее, в разделе «Цель испытаний» находится цель проведения испытаний. Раздел «Требования к программе» содержит требования, подлежащие проверке в ходе испытаний. В разделе «Требования к программной документации» указан состав программной документации. В разделе «Средства и порядок испытаний» указаны технические и программные средства, используемые в ходе испытаний, а также порядок проведения испытаний. Раздел «Методы испытаний» содержит описание используемых методов испытаний, описание проверок с указанием результатов. В последнем разделе «Приложение. Перечень материалов, использованных в ходе испытаний» содержатся публикации и другие материалы, использованные в ходе испытаний.

Оформление программного документа «Программа и методика испытаний» произведено по требованиям ГОСТ 19.301-79 «Программа и методика испытаний. Требования к содержанию и оформлению» [1].

**Содержание**

[Аннотация 3](#_Toc9686531)

[1. Объект испытаний 6](#_Toc9686532)

[1.1. Наименование 6](#_Toc9686533)

[1.2. Краткая характеристика области применения программного продукта 6](#_Toc9686534)

[2. Цель испытаний 7](#_Toc9686535)

[2.1. Цель проведения испытаний 7](#_Toc9686536)

[3. Требования к программе 8](#_Toc9686537)

[3.1. Требования к функциональным характеристикам 8](#_Toc9686538)

[3.2. Требования к надежности программы 8](#_Toc9686539)

[3.3. Требования квалификация и уровня подготовки пользователя 8](#_Toc9686540)

[3.4. Требования к составу и параметрам технических средств 8](#_Toc9686541)

[3.5. Требования к информационной и программной совместимости 9](#_Toc9686542)

[4. Требования к программной документации 10](#_Toc9686543)

[4.1. Состав программной документации 10](#_Toc9686544)

[5. Средства и порядок испытаний 11](#_Toc9686545)

[5.1. Технические средства, используемые во время испытаний 11](#_Toc9686546)

[5.2. Программные средства, используемые во время испытаний 11](#_Toc9686547)

[5.3. Порядок проведения испытаний 11](#_Toc9686548)

[6. Методы испытаний 12](#_Toc9686549)

[6.1. Проверка функциональных требований 12](#_Toc9686550)

[6.1.1. Конфигурация поведения WebSocket сервера списком сообщений 12](#_Toc9686551)

[6.1.2. Конфигурация поведения WebSocket сервера реакцией списком сообщений на сообщение от клиента 13](#_Toc9686552)

[6.1.3. Конфигурация поведения WebSocket сервера комбинацией из п. 1 и 2 14](#_Toc9686553)

[6.1.4. Конфигурация поведения WebSocket с помощью конфигурационного файла при старте сервера 15](#_Toc9686554)

[6.1.5. Конфигурация поведения WebSocket с помощью REST API 16](#_Toc9686555)

[6.1.6. Добавление задержки перед отправкой события в WebSocket 16](#_Toc9686556)

[6.1.7. Конфигурация поведения Server-Sent Events сервера списком сообщений 17](#_Toc9686557)

[6.1.8. Конфигурация поведения Servers-Sent Events с помощью конфигурационного файла при старте сервера 18](#_Toc9686558)

[6.1.9. Конфигурация поведения Servers-Sent Events с помощью REST API 19](#_Toc9686559)

[6.1.10. Добавление задержки перед отправкой события в Server-Sent Events 19](#_Toc9686560)

[6.2. Проверка требований к надежности 20](#_Toc9686561)

[6.2.1. Программа должна осуществлять проверку корректности введенных пользователем данных и выводить сообщение об ошибке при ее нахождении 20](#_Toc9686562)

[6.2.2. Ошибки, возникающие в приложении 20](#_Toc9686563)

[7. Список использованной литературы 22](#_Toc9686564)

# Объект испытаний

## Наименование

Наименование программного продукта – «Сервер для мокирования асинхронных протоколов».

## Краткая характеристика области применения программного продукта

Разрабатываемый программный продукт можно применить при разработке сервера с сетевым интерфейсом, в котором используется протокол WebSocket или Server-Sent Events. А также при разработке приложения, которое использует подобный интерфейс. Пользователь сможет создать имитацию реального сервиса путем описания желаемого поведения, без его непосредственной реализации.

# Цель испытаний

## Цель проведения испытаний

Целью проведения испытаний является проверка корректности работы программы, а также проверка соответствия разработанной программы функциональным требованиям и требованиям к надежности, изложенным в документе «Техническое задание».

# Требования к программе

## Требования к функциональным характеристикам

Создаваемый программный продукт должен иметь следующую функциональность:

1. Конфигурация поведения WebSocket сервера списком сообщений
2. Конфигурация поведения WebSocket сервера реакцией списком сообщений списком сообщений на сообщение от клиента
3. Конфигурация поведения WebSocket сервера комбинацией из п. 1 и 2
4. Конфигурация поведения WebSocket с помощью конфигурационного файла при старте сервера
5. Конфигурация поведения WebSocket с помощью REST API
6. Добавление задержки перед отправкой события в WebSocket
7. Конфигурация поведения Server-Sent Events сервера списком сообщений
8. Конфигурация поведения Servers-Sent Events с помощью конфигурационного файла при старте сервера
9. Конфигурация поведения Servers-Sent Events с помощью REST API
10. Добавление задержки перед отправкой события в Server-Sent Events

## Требования к надежности программы

К программе предъявляются следующие требования надежности:

Программа должна осуществлять проверку корректности введенных пользователем данных и выводить сообщение об ошибке при ее нахождении.

Программа должна корректно завершаться при возникновении ошибок.

## Требования квалификация и уровня подготовки пользователя

Пользователь должен иметь базовые навыки работы в текстовых редакторах, а также уметь читать и редактировать JSON файлы.

Обязательно ознакомление пользователя с руководством оператора.

## Требования к составу и параметрам технических средств

Для эффективной работы с программой предъявляются следующие требования к составу и параметрам технических средств:

1. Процессор Pentium 2 266 MHz или мощнее
2. Оперативная память не менее 1 Гб
3. Свободное дисковое пространство не меньше 1 Гб
4. Исправная сетевая карта
5. Монитор
6. Клавиатура
7. Мышь

## Требования к информационной и программной совместимости

Для корректной работы программного продукта требуется:

1. Установленный JDK версии 1.8 или выше
2. Установленный SBT версии 1.2.8 или совместимый
3. Установленный Docker версии 2.0.0.3 или совместимый

# Требования к программной документации

## Состав программной документации

На испытания должна быть представлена документация к программе в следующем составе:

1. «Сервер для мокирования асинхронных протоколов». Техническое задание. ГОСТ 19.71-208
2. «Сервер для мокирования асинхронных протоколов». Руководство оператора. ГОСТ 19.505-79
3. «Сервер для мокирования асинхронных протоколов». Программа и методика испытаний. ГОСТ 19.301-79
4. «Сервер для мокирования асинхронных протоколов». Текст программы ГОСТ 19.401-78

# Средства и порядок испытаний

## Технические средства, используемые во время испытаний

Для надежной и бесперебойной работы программы требуется следующий состав технических средств:

1. Процессор Intel Core i5 2500
2. Оперативная память 4 Гб
3. Свободное дисковое пространство 128 Гб
4. Исправная сетевая карта
5. Монитор
6. Клавиатура
7. Мышь

## Программные средства, используемые во время испытаний

Для работы программы необходим следующий состав программных средств:

1. JDK версии 1.8
2. SBT версии 1.2.8
3. Docker версии 2.0.0.3

## Порядок проведения испытаний

Испытания должны проводиться в следующем порядке:

1. Проверка требований к программной документации
2. Проверка требований к функциональным характеристикам
3. Проверка требований к надёжности

# Методы испытаний

## Проверка функциональных требований

Чтобы запустить программу требуется ввести в консоль команду:

docker build -t async-mock . **&&** docker run -it --rm -p 2525:2525 -p 5000:5000 async-mock

После этого требуется открыть браузер, и в адресной строке ввести адрес localhost:2525/swagger

### Конфигурация поведения WebSocket сервера списком сообщений

Создадим WebSocket мок со следующей конфигурацией и отправим его на сервер (рис. 1):

{  
 **"port"**: 5000,  
 **"stubs"**: [  
 {  
 **"predicates"**: [],  
 **"responses"**: [  
 {  
 **"events"**: [  
 {  
 **"data"**: **"Hello World!"** }  
 ],  
 **"type"**: **"websocket"** }  
 ]  
 }  
 ]  
}

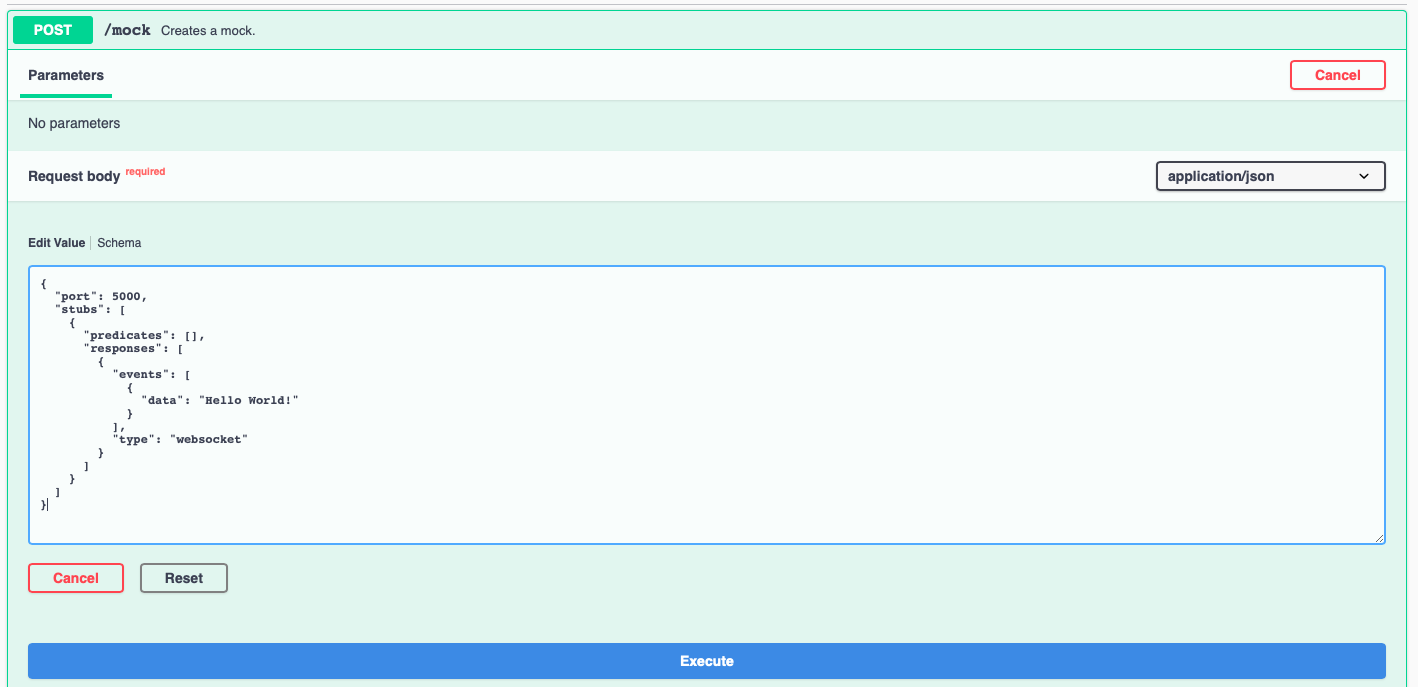


Рисунок 1. Отправка мока с последовательностью событий для WebSocket

Для проверки WebSocket будем использовать утилиту wsc [2]. Следующей командой проверим работоспособность мока:

wsc -er ws://localhost:5000

Вывод программы:

Connected to ws://localhost:5000

< "Hello World!" (5ms)

После каждой проверки будем удалять созданный мок.

### Конфигурация поведения WebSocket сервера реакцией списком сообщений на сообщение от клиента

Создадим WebSocket мок со следующей конфигурацией и отправим его на сервер (рис. 2):

{  
 **"port"**: 5000,  
 **"stubs"**: [  
 {  
 **"predicates"**: [],  
 **"responses"**: [  
 {  
 **"reactions"**: [  
 {  
 **"predicates"**: [],  
 **"reaction"**: [  
 {  
 **"data"**: **"response"** }  
 ]  
 }  
 ],  
 **"events"**: [],  
 **"type"**: **"websocket"** }  
 ]  
 }  
 ]  
}

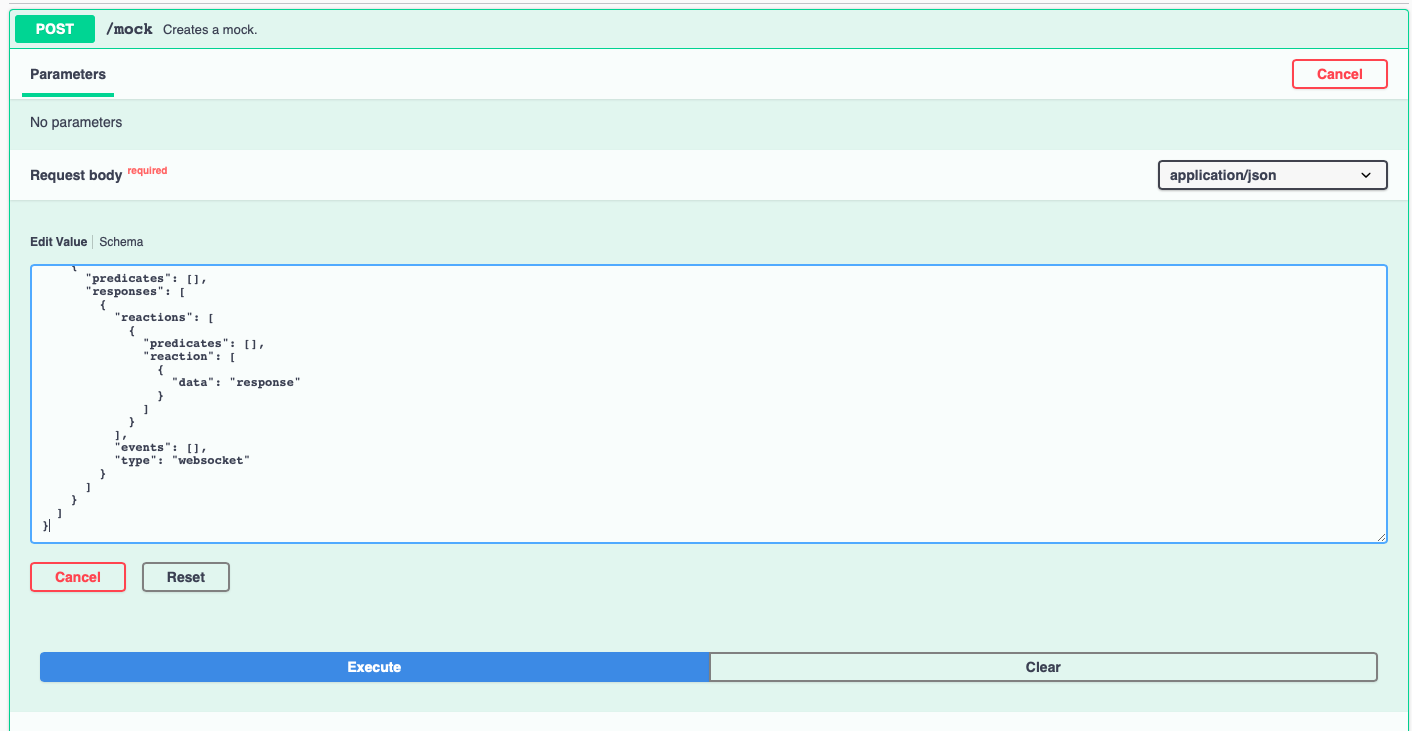


Рисунок 2. Отправка мока с реакцией на события от клиента для WebSocket

Следующей командой проверим работоспособность мока:

wsc -er ws://localhost:5000

Вывод программы:

Connected to ws://localhost:5000

> Test event

< "response" (80ms)

### Конфигурация поведения WebSocket сервера комбинацией из п. 1 и 2

Создадим WebSocket мок со следующей конфигурацией и отправим его на сервер (рис. 3):

{  
 **"port"**: 5000,  
 **"stubs"**: [  
 {  
 **"predicates"**: [],  
 **"responses"**: [  
 {  
 **"reactions"**: [  
 {  
 **"predicates"**: [],  
 **"reaction"**: [  
 {  
 **"data"**: **"response"** }  
 ]  
 }  
 ],  
 **"events"**: [  
 {  
 **"data"**: **"Hello World!"** }  
 ],  
 **"type"**: **"websocket"** }  
 ]  
 }  
 ]  
}

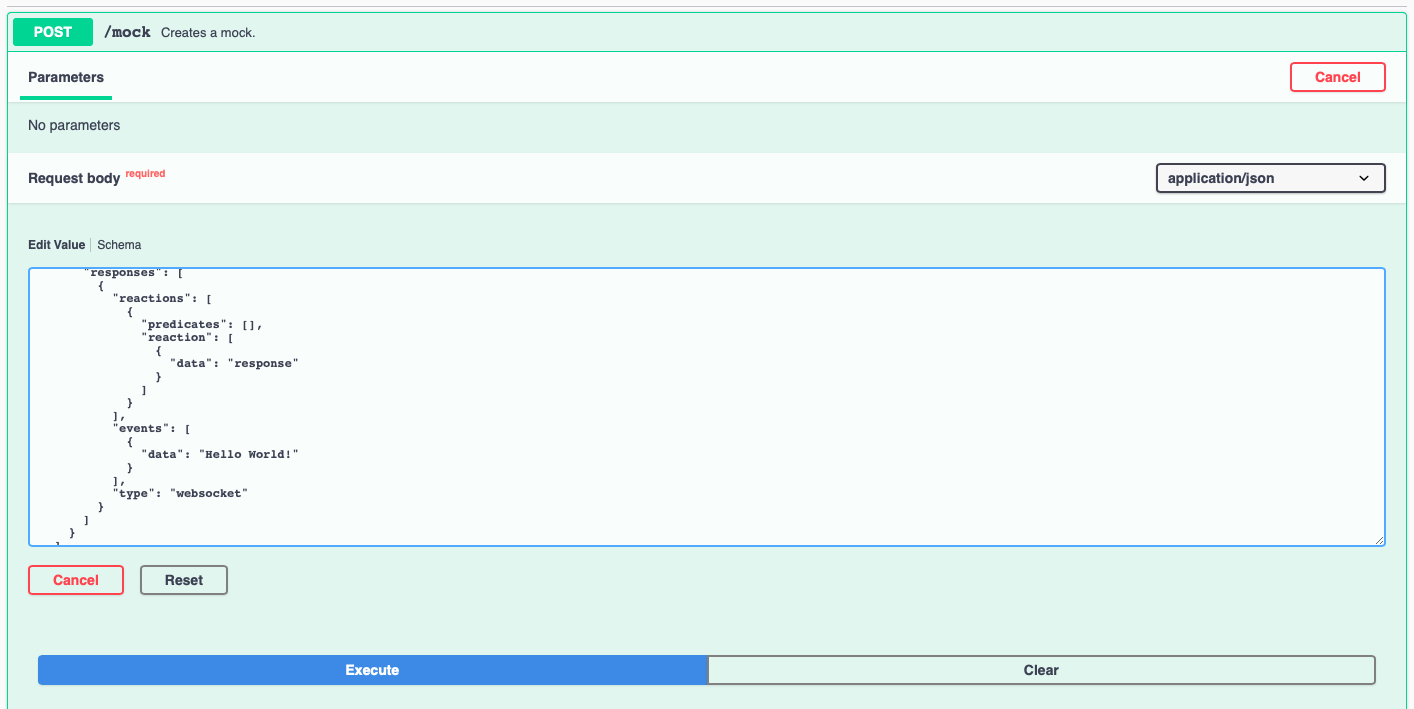


Рисунок 3. Отправка мока с последовательностью событий и реакцией на события от клиента для WebSocket

Следующей командой проверим работоспособность мока:

wsc -er ws://localhost:5000

Вывод программы:

Connected to ws://localhost:5000

< "Hello World!" (5ms)

> Test event

< "response" (28ms)

### Конфигурация поведения WebSocket с помощью конфигурационного файла при старте сервера

Создадим файл configuration.json. В него запишем конфигурацию из п. 6.1.1. Далее запустим сервер, передав аргумент -f configuration.json в параметры запуска. При запуске приложение выводит в лог:

Starting AsyncMock with args [-f, configuration.json]

Проверим работоспособность мока (вывод wsc):

Connected to ws://localhost:5000

< "Hello World!" (5ms)

### Конфигурация поведения WebSocket с помощью REST API

Данное требование проверяется в проверках 6.1.1 – 6.1.3.

### Добавление задержки перед отправкой события в WebSocket

Создадим WebSocket мок со следующей конфигурацией и отправим его на сервер (рис. 4):

{  
 **"port"**: 5000,  
 **"stubs"**: [  
 {  
 **"predicates"**: [],  
 **"responses"**: [  
 {  
 **"reactions"**: [  
 {  
 **"predicates"**: [],  
 **"reaction"**: [  
 {  
 **"data"**: **"response"**,  
 **"delay"**: **"1 second"** }  
 ]  
 }  
 ],  
 **"events"**: [  
 {  
 **"data"**: **"Hello World!"**,  
 **"delay"**: **"2 seconds"** }  
 ],  
 **"type"**: **"websocket"** }  
 ]  
 }  
 ]  
}

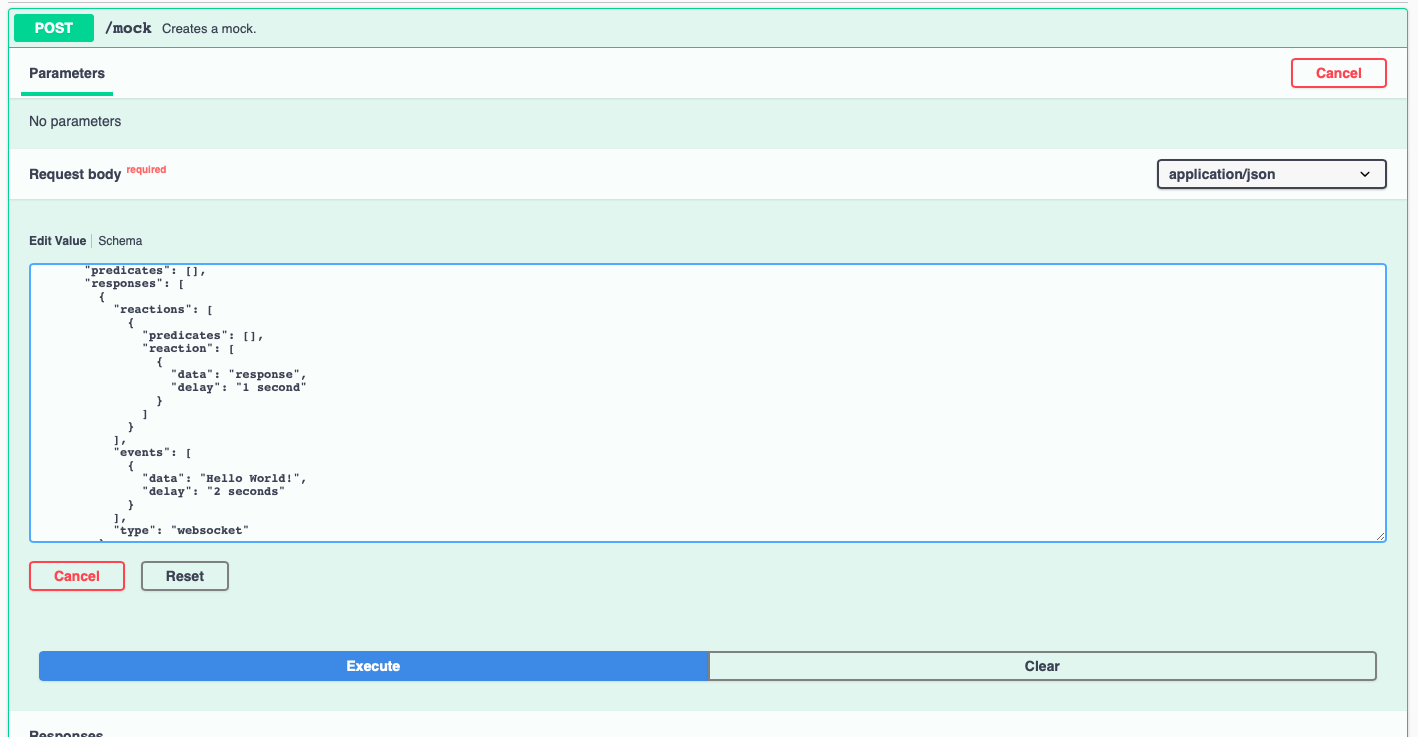


Рисунок 4. Отправка мока с задержкой для WebSocket

Проверим работоспособность мока (вывод wsc):

Connected to ws://localhost:5000

< "Hello World!" (2016ms)

> Test event

< "response" (1029ms)

### Конфигурация поведения Server-Sent Events сервера списком сообщений

Создадим Server-Sent Events мок со следующей конфигурацией и отправим его на сервер (рис. 5):

{  
 **"port"**: 5000,  
 **"stubs"**: [  
 {  
 **"predicates"**: [],  
 **"responses"**: [  
 {  
 **"events"**: [  
 {  
 **"data"**: **"Hello World!"** }  
 ],  
 **"type"**: **"sse"** }  
 ]  
 }  
 ]  
}

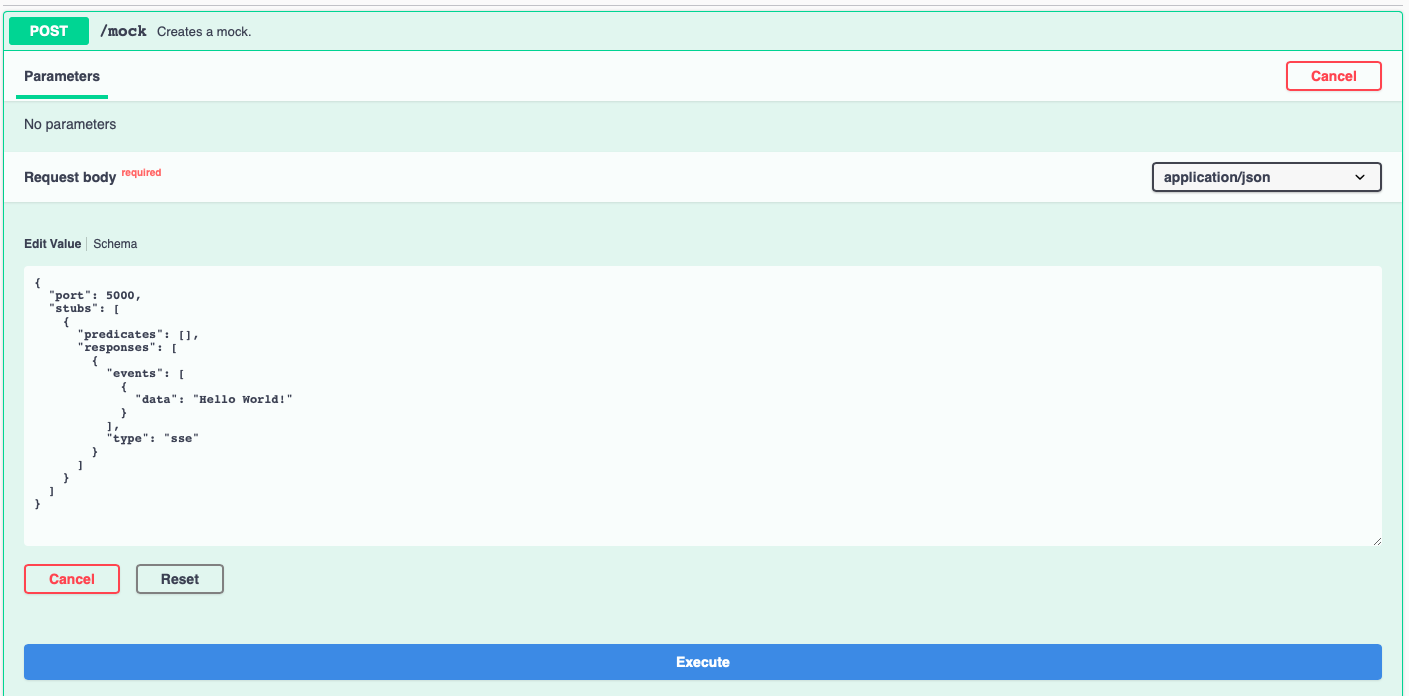


Рисунок 5. Отправка мока с последовательностью событий для Server-Sent Events

Для проверки Server-Sent Events будем заходить в браузере по ссылке localhost:5000. Результат можно увидеть на рис. 6.

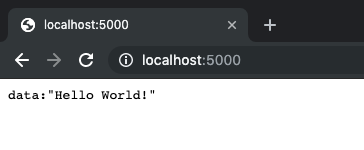


Рисунок 6. Ответ мока с последовательностью событий для Server-Sent Events

### Конфигурация поведения Servers-Sent Events с помощью конфигурационного файла при старте сервера

Создадим файл configuration.json. В него запишем конфигурацию из п. 6.1.7. Далее запустим сервер, передав аргумент -f configuration.json в параметры запуска. При запуске приложение выводит в лог:

Starting AsyncMock with args [-f, configuration.json]

Убедимся, что мок работает. Результат можно увидеть на рис. 7.

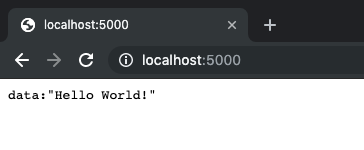


Рисунок 7. Результат проверки конфигурации с помощью файла

### Конфигурация поведения Servers-Sent Events с помощью REST API

Данное требование проверяется в проверках 6.1.7 и 6.1.10.

### Добавление задержки перед отправкой события в Server-Sent Events

Создадим Server-Sent Events мок со следующей конфигурацией и отправим его на сервер (рис. 8):

{  
 **"port"**: 5000,  
 **"stubs"**: [  
 {  
 **"predicates"**: [],  
 **"responses"**: [  
 {  
 **"events"**: [  
 {  
 **"data"**: **"Hello World!"** },  
 {  
 **"data"**: **"Delayed event"**,  
 **"delay"**: **"2 seconds"** }  
 ],  
 **"type"**: **"sse"** }  
 ]  
 }  
 ]  
}

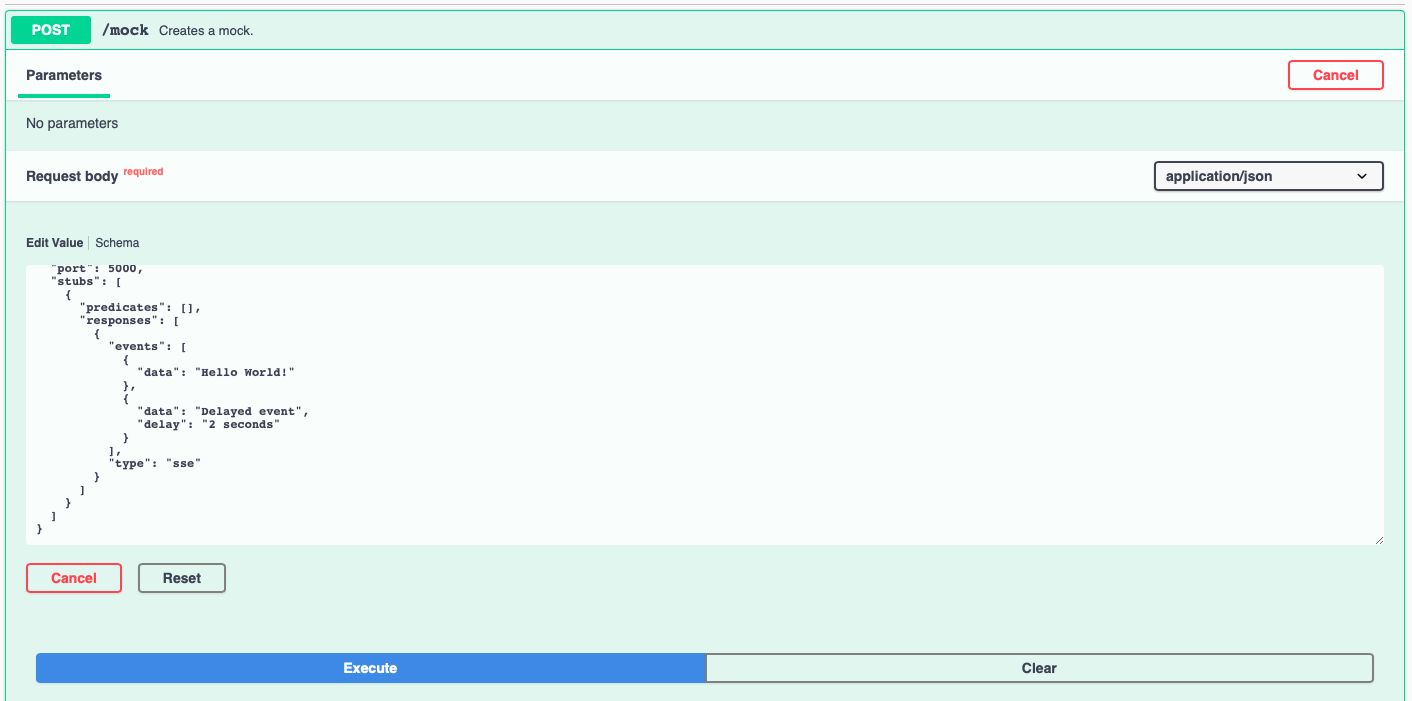


Рисунок 8. Отправка мока с задержкой событий для Server-Sent Events

Результат проверки можно увидеть на рис. 9. Второе событие отображается на странице с задержкой.

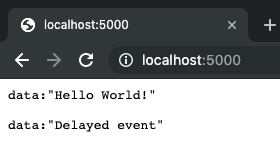


Рисунок 9. Ответ мока с задержкой событий для Server-Sent Events

## Проверка требований к надежности

### Программа должна осуществлять проверку корректности введенных пользователем данных и выводить сообщение об ошибке при ее нахождении

При возникновении ошибок, сервер возвращает наиболее подходящий код ответа с текстом сообщения ошибки.

### Ошибки, возникающие в приложении

**404 Not Found**

Данная ошибка означает, что был неправильно сформирован путь запроса, и сервер не знает, как на данный запрос ответить.

**500 Unexpected Exception**

Данная ошибка означает, что сервер получил необработанную ошибку, и необходимо просмотреть его логи для подробного изучения причины ошибки. Обычно такая ошибка возвращается, если указанный порт занят другим приложением, и сервер не может получить к нему доступ.

# Список использованной литературы

1. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

2. WebSocket client for the terminal [Электронный ресурс]. URL: https://github.com/danielstjules/wsc (дата обращения: 22.04.2019).