

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №7
по дисциплине «Web-программирование»

Автор: Шарапков Егор Викторович М33081

Преподаватель: Приискалов Роман Андреевич



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург 2022

WebSocket — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени. Веб-сокеты, в отличие от http, для ответа не нужны ваши повторяющиеся запросы. Достаточно выполнить один запрос и ждать отклика. Вы можете просто слушать сервер, который будет отправлять вам сообщения по мере готовности.

Для того чтобы подключить к Nest приложению возможность работы с веб-сокетами, необходимо установить следующие библиотеки:

```
npm i --save @nestjs/websockets @nestjs/platform-socket.io
```

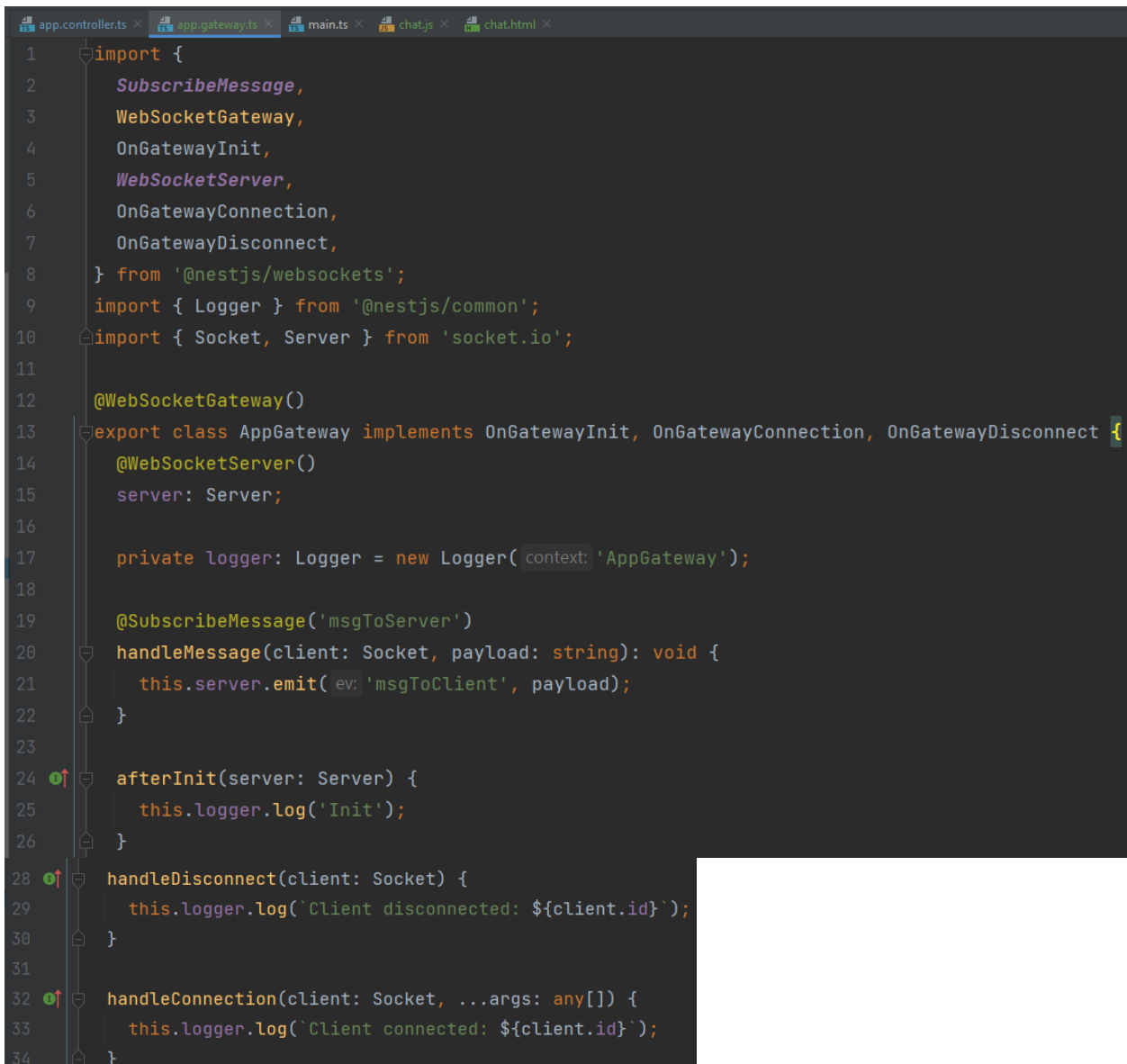
Далее создадим gateway

```
nest g gateway app
```

В Nest шлюз — это класс, помеченный `@WebSocketGateway()` декоратором. Технически шлюзы не зависят от платформы, что делает их совместимыми с любой библиотекой WebSockets после создания адаптера.

Я буду делать простой чат в реальном времени.

Создаю сервер Websockets.



```
1 import {
2   SubscribeMessage,
3   WebSocketGateway,
4   OnGatewayInit,
5   WebSocketServer,
6   OnGatewayConnection,
7   OnGatewayDisconnect,
8 } from '@nestjs/websockets';
9 import { Logger } from '@nestjs/common';
10 import { Socket, Server } from 'socket.io';
11
12 @WebSocketGateway()
13 export class AppGateway implements OnGatewayInit, OnGatewayConnection, OnGatewayDisconnect {
14   @WebSocketServer()
15   server: Server;
16
17   private logger: Logger = new Logger('AppGateway');
18
19   @SubscribeMessage('msgToServer')
20   handleMessage(client: Socket, payload: string): void {
21     this.server.emit('ev: msgToClient', payload);
22   }
23
24   afterInit(server: Server) {
25     this.logger.log('Init');
26   }
27
28   handleDisconnect(client: Socket) {
29     this.logger.log(`Client disconnected: ${client.id}`);
30   }
31
32   handleConnection(client: Socket, ...args: any[]) {
33     this.logger.log(`Client connected: ${client.id}`);
34   }
35 }
```

Теперь перейдем к websockets клиенту.

JS код для работы на стороне клиента.

```
1  const app = new Vue({
2    el: '#app',
3    data: {
4      title: 'Чат с помощью WebSocket',
5      name: '',
6      text: '',
7      messages: [],
8      socket: null
9    },
10   methods: {
11     sendMessage() {
12       if(this.validateInput()) {
13         const message = {
14           name: this.name,
15           text: this.text
16         }
17         this.socket.emit( event: 'msgToServer', message)
18         this.text = ''
19       }
20     },
21     receivedMessage(message) {
22       this.messages.push(message)
23     },
24     validateInput() {
25       return this.name.length > 0 && this.text.length > 0
26     }
27   },
28   created() {
29     this.socket = io('http://localhost:3000')
30     this.socket.on( event: 'msgToClient', listener: (message) => {
31       this.receivedMessage(message)
32     })
33   }
34 })
```

Метод `created()` будет выполняться при создании внешнего интерфейса. В нем создается экземпляр переменной `socket` с помощью библиотеки `socket.io`.

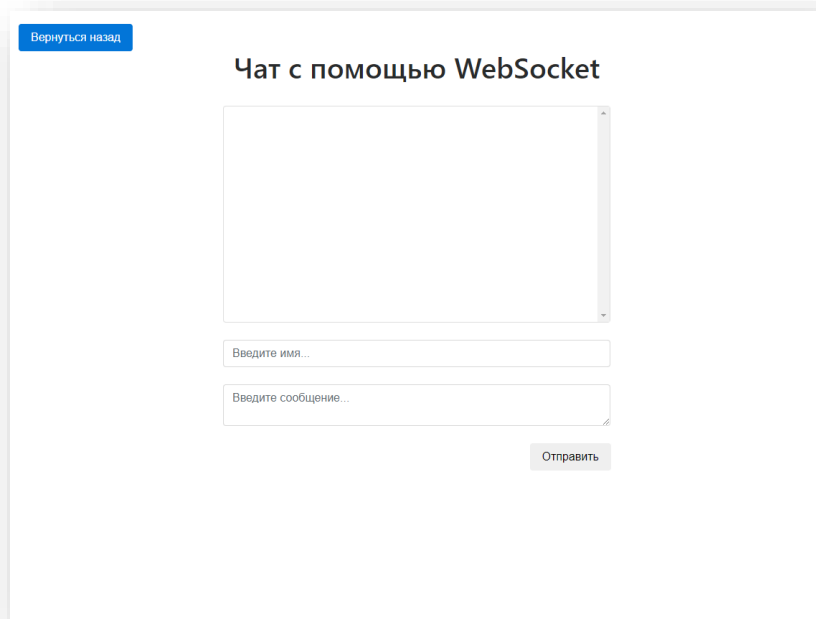
Мы также добавляем слушатель событий в наш сокет, который прослушивает событие `msgToClient`, которое мы создали ранее на нашем сервере.

Затем у нас есть функция `sendMessage()`, которая получает входные данные из нашего макета и отправляет их на наш сервер, используя то же событие, если входные данные верны.

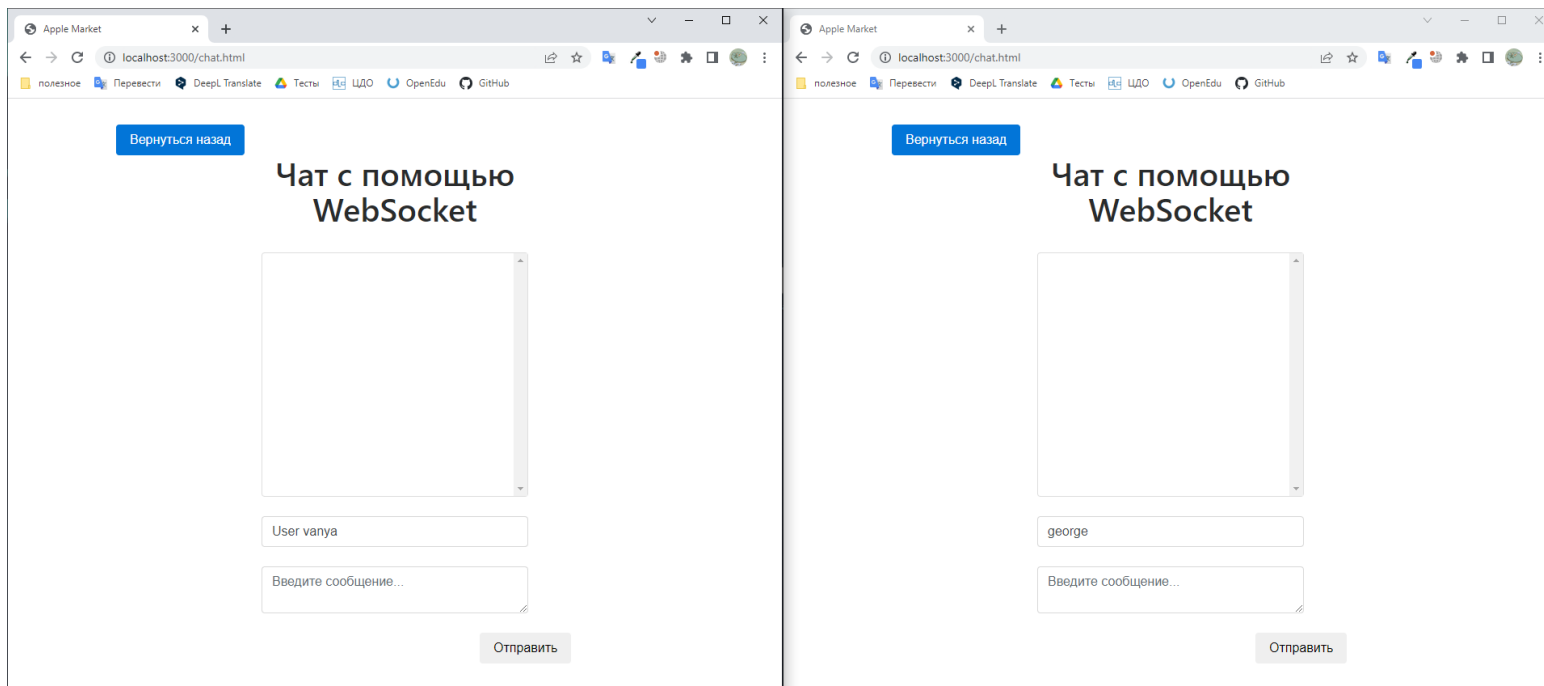
Создаю макет чата.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css" integrity="sha384-rwoI
8   <title>Apple Market</title>
9   <link rel="stylesheet" href="/css/chat.css">
10  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
11  <script src="https://cdn.socket.io/4.3.2/socket.io.min.js" integrity="sha384-KAZ4DtjNhLCh0B/hxXuKqHMLYvx3b5MLT55xPEiNmREKRzeEm+RVF
12 </head>
13 <body>
14 <div id="app" class="container">
15   <div class="row">
16     <div class="col">
17       <a href="http://localhost:3000/"><button class="btn btn-primary">Вернуться назад</button></a>
18     </div>
19   </div>
20   <div class="row">
21     <div class="col-md-6 offset-md-3 col-sm-12">
22       <h1 class="text-center">{{ title }}</h1>
23       <br>
24       <div id="status"></div>
25       <div id="chat">
26         <div class="card">
27           <div id="messages" class="card-block">
28             <ul>
29               <li v-for="message of messages">{{ message.name }}: {{ message.text }}</li>
30             </ul>
31           </div>
32         </div>
33         <br>
34         <input type="text" v-model="name" id="username" class="form-control" placeholder="Введите имя...">
35         <br>
36         <textarea id="textarea" class="form-control" v-model="text" placeholder="Введите сообщение..."></textarea>
37         <br>
38         <div class="row">
39           <div class="col-9"></div>
40           <div class="col-3">
41             <button id="send" class="btn" @click.prevent="sendMessage">Отправить</button>
42           </div>
43         </div>
44       </div>
45     </div>
46   </div>
47 </div>
48
49 <script src="/js/chat.js"></script>
50 </body>
51 </html>
```

Открываю страницу с чатом.



Также открою ее в дополнительном окне браузера, введу имя.



Сообщения после отправки появляются во всех окнах одновременно.

