|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №6**

по дисциплине «Интерфейсы прикладного программирования»

**Тема практической работы:**

**Разработать клиент-серверное приложение чата с использованием WebSocket API**

**Студент группы** ИКБО-20-19 Анваржонов Ж Т

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** Белолипцев М. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «28» ноября 2021 г.

Работа принята «\_\_\_» \_\_\_\_\_\_ 2021 г.

Москва 2021

**Теоретическое введение**

**Цель работы:** Разработка клиент-серверного приложения с использованием технологии Websocket.

**Постановка задачи**

Необходимо разработать клиент-серверное приложение чата с использованием WebSocket API применив любые актуальные технологии для фронтенд и бэкенд разработки.

**Ход работы**

Для реализации технологии WebSocket API был выбран Java + Spring Framework.

Ниже приведены фрагменты программного кода, обеспечивающую реализацию websocket.

**Wsconfig.java:**

*package* ru.anvarzhonov.springbootwebsocket.config;  
  
  
*import* org.springframework.context.annotation.*Configuration*;  
*import* org.springframework.messaging.simp.config.MessageBrokerRegistry;  
*import* org.springframework.web.socket.WebSocketMessage;  
*import* org.springframework.web.socket.config.annotation.*EnableWebSocketMessageBroker*;  
*import* org.springframework.web.socket.config.annotation.StompEndpointRegistry;  
*import* org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;  
  
*@Configuration  
@EnableWebSocketMessageBroker  
public class* WsConfig *implements* WebSocketMessageBrokerConfigurer {  
 *@Override  
 public void* registerStompEndpoints(StompEndpointRegistry registry) {  
 registry.addEndpoint("/javatechie").withSockJS();  
 }  
  
 *@Override  
 public void* configureMessageBroker(MessageBrokerRegistry registry) {  
 registry.enableSimpleBroker("/topic");  
 registry.setApplicationDestinationPrefixes("/app");  
 }  
}

В этой конфигурации у нас WebSocketConfig аннотирован @Configuration, означая, что это конфигурационный Spring класс. Он также аннотирован @EnableWebSocketMessageBroker. Как и предполагает его название @EnableWebSocketMessageBroker включает обработке сообщений по WebSocket, возвращаемый брокером сообщений.

Метод configureMessageBroker() переопределяет поведение по умолчанию в WebSocketMessageBrokerConfigurer для настройки брокера сообщений. Он вызывает enableSimpleBroker() для включения простого брокера сообщений в памятичтобы возвращать обратно сообщения клиенту по направлениям с префиксом /topic. Он также объявляет префикс /app для сообщений, привязанных к методам, аннотированными @MessageMapping.

Метод registerStompEndpoints() регистрирует /javatechie, включая дополнительно SockJS как альтернативный вариант обмена сообщениями, когда WebSocket не доступен.

**chatMessage:**

*package* ru.anvarzhonov.springbootwebsocket.models;  
  
*public class* ChatMessage {  
 *private* String content;  
 *private* String sender;  
 *private* MessageType messageType;  
  
 *public* String getContent() {  
 *return* content;  
 }  
  
 *public void* setContent(String content) {  
 *this*.content = content;  
 }  
  
 *public* String getSender() {  
 *return* sender;  
 }  
  
 *public void* setSender(String sender) {  
 *this*.sender = sender;  
 }  
  
 *public* MessageType getMessageType() {  
 *return* messageType;  
 }  
  
 *public void* setMessageType(MessageType messageType) {  
 *this*.messageType = messageType;  
 }  
  
 *public enum* MessageType{  
 CHAT, LEAVE, JOIN  
 }  
  
  
}

**ChatController:**

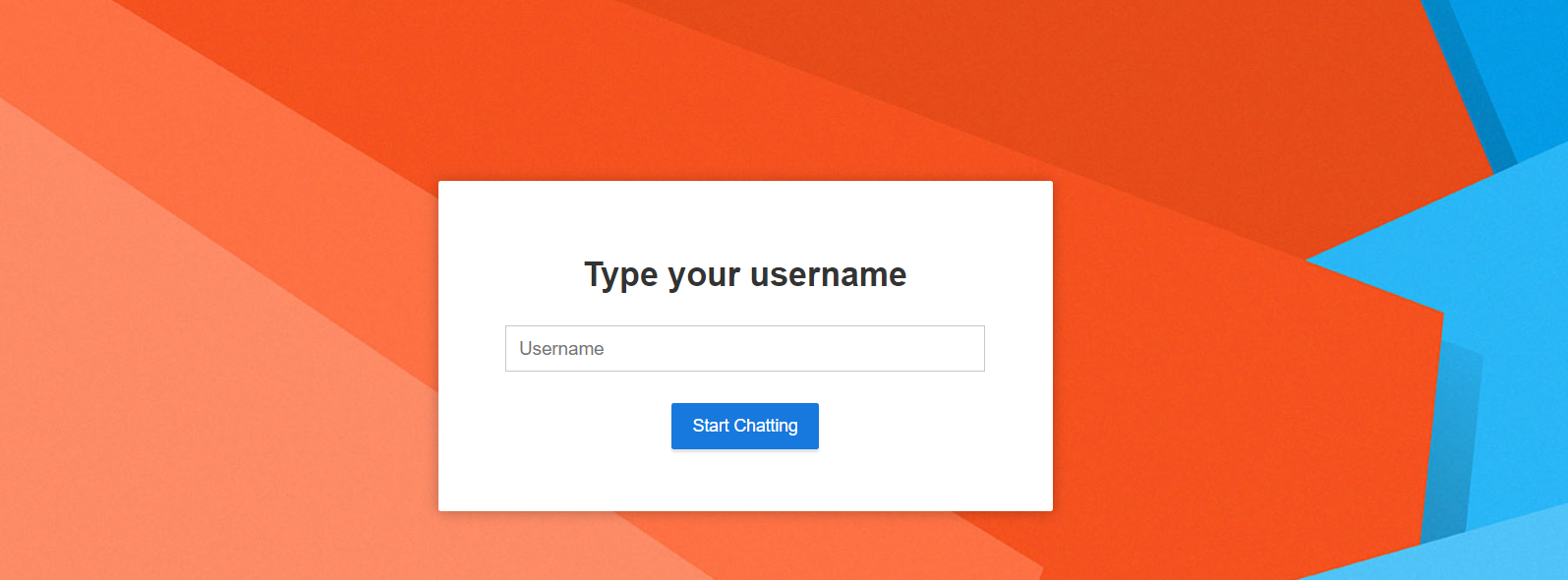
*package* ru.anvarzhonov.springbootwebsocket.controller;  
  
  
*import* org.springframework.messaging.handler.annotation.*MessageMapping*;  
*import* org.springframework.messaging.handler.annotation.*Payload*;  
*import* org.springframework.messaging.handler.annotation.*SendTo*;  
*import* org.springframework.messaging.simp.SimpMessageHeaderAccessor;  
*import* org.springframework.stereotype.*Controller*;  
*import* ru.anvarzhonov.springbootwebsocket.models.ChatMessage;  
  
*@Controller  
public class* ChatController {  
  
 *@MessageMapping*("/chat.register")  
 *@SendTo*("/topic/public")  
 *public* ChatMessage register(ChatMessage chatMessage, SimpMessageHeaderAccessor headerAccessor) {  
 headerAccessor.getSessionAttributes().put("username", chatMessage.getSender());  
 *return* chatMessage;  
 }  
  
 *@MessageMapping*("/chat.send")  
 *@SendTo*("/topic/public")  
 *public* ChatMessage sendMessage (ChatMessage chatMessage) {  
  
 *return* chatMessage;  
 }  
}

**View (index.html):**

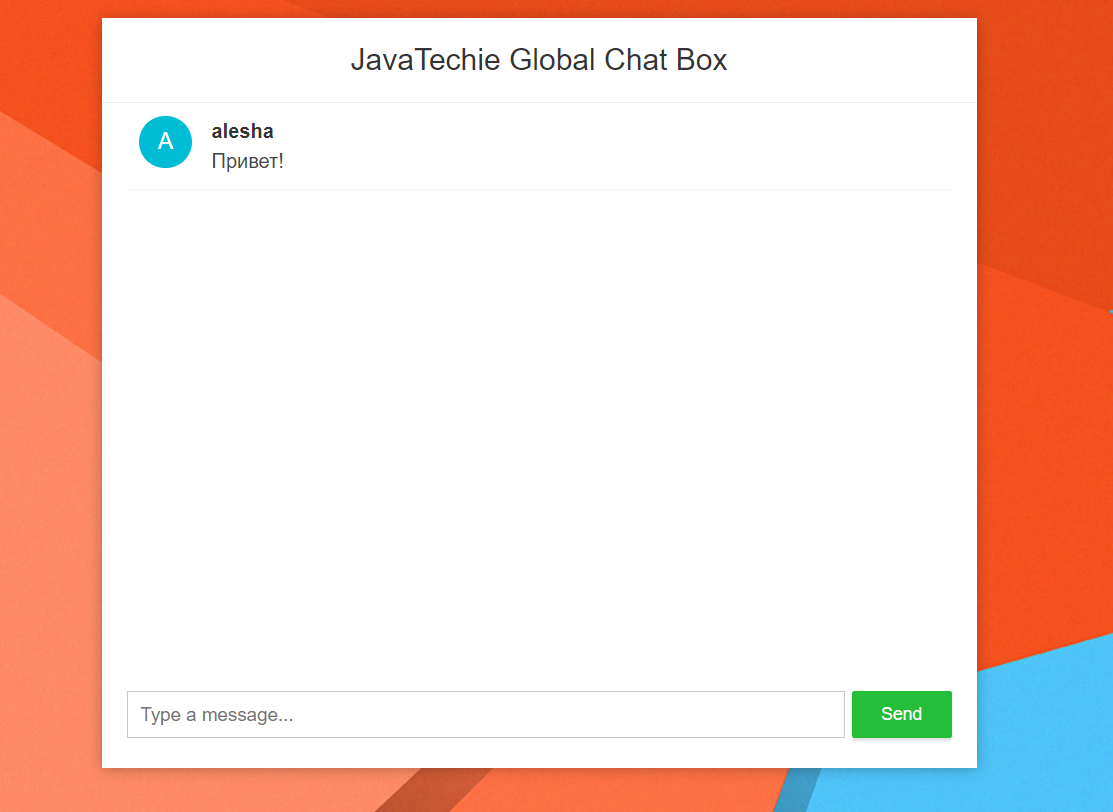
<!DOCTYPE *html*>  
<html>  
<head>  
 <meta *name*="viewport"  
 *content*="width=device-width, initial-scale=1.0, minimum-scale=1.0">  
 <title>Spring Boot WebSocket Chat Application | CalliCoder</title>  
 <link *rel*="stylesheet" *href*="main.css" />  
</head>  
<body *background*="18.jpg"  
 *style*="background-position: center; background-repeat: no-repeat; background-size: cover;">  
<noscript>  
 <h2>Sorry! Your browser doesn't support Javascript</h2>  
</noscript>  
  
<div *id*="username-page">  
 <div *class*="username-page-container">  
 <h1 *class*="title">Type your username</h1>  
 <form *id*="usernameForm" *name*="usernameForm">  
 <div *class*="form-group">  
 <input *type*="text" *id*="name" *placeholder*="Username"  
 *autocomplete*="off" *class*="form-control" />  
 </div>  
 <div *class*="form-group">  
 <button *type*="submit" *class*="accent username-submit">Start  
 Chatting</button>  
 </div>  
 </form>  
 </div>  
</div>  
  
<div *id*="chat-page" *class*="hidden">  
 <div *class*="chat-container">  
 <div *class*="chat-header">  
 <h2>Anvarzhonov Chat</h2>  
 </div>  
 <div *class*="connecting">Connecting...</div>  
 <ul *id*="messageArea">  
  
 </ul>  
 <form *id*="messageForm" *name*="messageForm" *nameForm*="messageForm">  
 <div *class*="form-group">  
 <div *class*="input-group clearfix">  
 <input *type*="text" *id*="message" *placeholder*="Type a message..."  
 *autocomplete*="off" *class*="form-control" />  
 <button *type*="submit" *class*="primary">Send</button>  
 </div>  
 </div>  
 </form>  
 </div>  
</div>  
  
<script  
 *src*="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.min.js"></script>  
<script  
 *src*="https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/stomp.min.js"></script>  
<script *src*="main.js"></script>  
</body>  
</html>

Покажем результат работы:

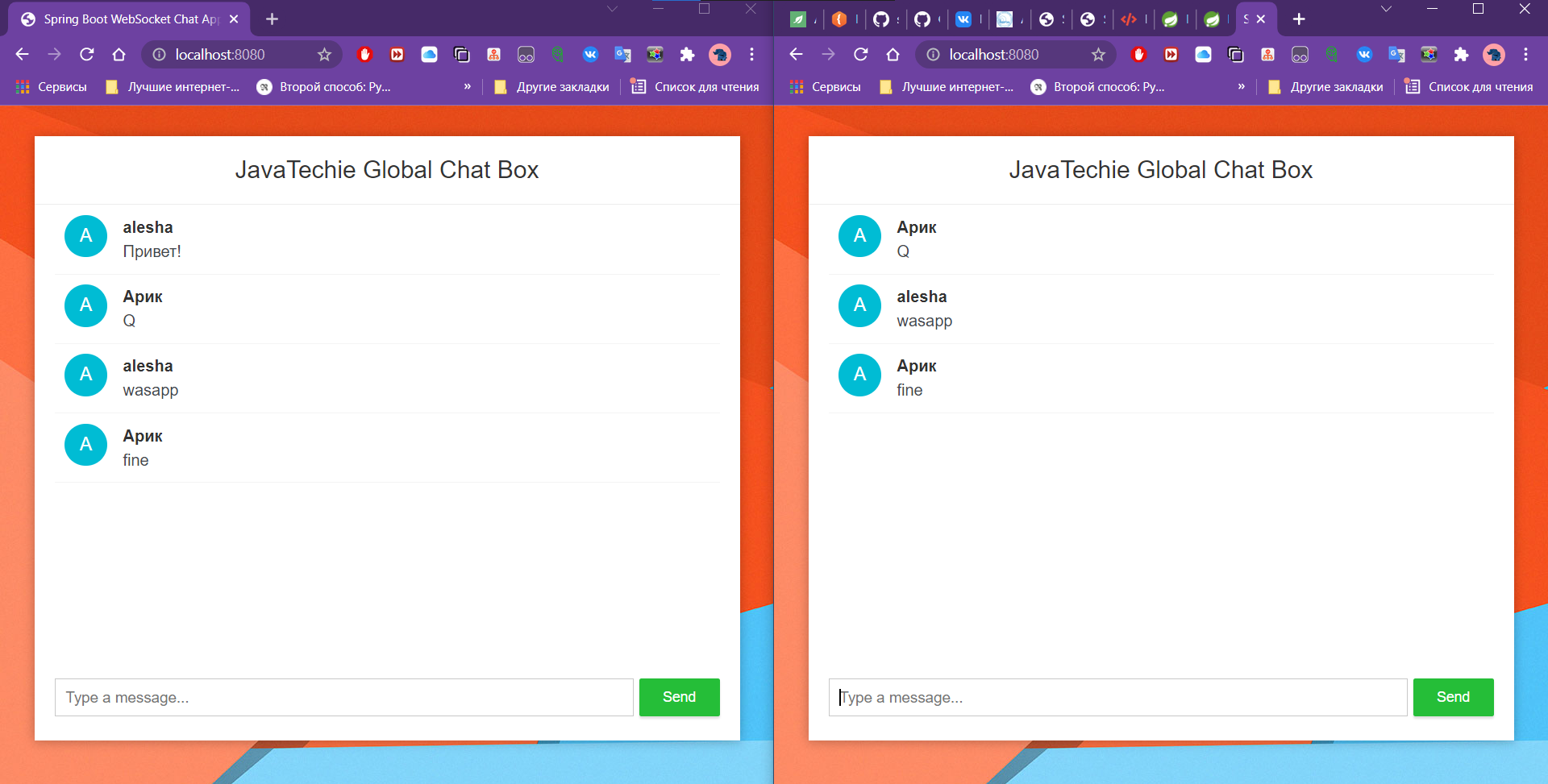
В начале работы приложения требуется ввести имя:



Попробуем зарегистрировать пользователя и написать сообщение:



Добавим еще одного клиента и покажем результат обмена сообщениями:



**Вывод**

В результате выполнения данной практической работы были закреплены навыки работы с веб-сокетами.

# Список используемых информационных источников

1. Журавлев, А. Е. Инфокоммуникационные системы. Программное обеспечение : учебник для вузов / А. Е. Журавлев, А. В. Макшанов, А. В. Иванищев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 376 с. — ISBN 978-5-8114-8515-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/176658 (дата обращения: 15.10.2021). — Режим доступа: для авториз. пользователей.

2. Григорьев, В.К. React.js. Путеводитель по технологии : учебник / К. К. Сухов. — Москва : ДМК Пресс, 2015. — 416 с. — ISBN 978-5-97060-164-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/69954 (дата обращения: 15.10.2021). — Режим доступа: для авториз. пользователей.

3. Заяц, А. М. Проектирование и разработка WEB-приложений. Введение в frontend и backend разработку на JavaScript и react.js : учебное пособие для вузов / А. М. Заяц, Н. П. Васильев. — 3-е изд., стер. — Санкт-Петербург : Лань, 2021. — 120 с. — ISBN 978-5-8114-7042-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/154380 (дата обращения: 20.10.2021). — Режим доступа: для авториз. пользователей.