Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki Katedra Informatyki, Elektroniki i Elektrotechniki		
Kierunek: Informatyka	**Laboratorium:** Programowanie obiektowe 2	
Grupa dziekańska: 2ID13B	**Temat ćwiczenia:** CHAT ANONIMOWY Z UŻYCIEM WEB SOKETÓW	**Wykonał zespół w składzie:** Rudnytskyi Dmytro

1. **Ogólny opis projektu wraz informacjami o technologiach, framework'ach, bibliotekach użytych w projekcie**

W moim projekcie wykorzystałem technologie i biblioteki takie jak java, javascript, html, css. Wśród użytych frameworków były:

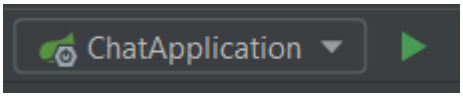
- Spring Boot: framework do tworzenia aplikacji w Javie.
- Spring WebSocket: moduł Spring do obsługi WebSocket.
- Lombok: Biblioteka upraszczająca pisanie kodu za pomocą adnotacji.
- SockJS i STOMP: Protokoły zapewniające komunikację między klientem a serwerem za pośrednictwem gniazd internetowych.
- Base64 (Apache Commons Codec): Biblioteka do kodowania i dekodowania danych w formacie Base64.

2. **Informacje na temat funkcjonalności projektu**

Funkcjonalność projektu obejmuje możliwość wprowadzenia swojego pseudonimu do korespondencji, który zawiera tekst i zdjęcie. Dla znaczników strony internetowej użyto html, dla funkcjonalności javascript, dla projektu css.

3. **Informacje na temat sposobu uruchomienia oraz obsługi projektu**

Aby uruchomić projekt należy uruchomić serwer lokalny.



Połączić się z lokalnym serwerem za pomocą dowolnego browsera. Do korespondencji potrzebne są co najmniej dwie osoby, choć liczba osób jest nieograniczona.

4. **Informacje na temat stworzonych klas, metod, funkcji (bez kodu źródłowego) z opisem ich podstawowej funkcjonalności (przyjmowanymi parametrami, wartościami zwracanymi) oraz ich przeznaczeniem**

- **WebSocketEventListener:**

Klasa odpowiedzialna za obsługę zdarzeń połączenia i rozłączenia WebSocket.

- **Metoda: handleWebSocketConnectListener:**

Obsługuje zdarzenia połączenia WebSocket. Wysyła wiadomość typu JOIN do czatu ogólnego o nowo połączonym użytkowniku.

Akceptowane parametry: SessionConnectEvent connectEvent

Zwracane wartości: Brak.

- **Metoda: handleWebSocketDisconnectListener**

Obsługuje zdarzenia rozłączenia WebSocket. Wysyła wiadomość typu LEAVE do publicznego pokoju rozmów o użytkowniku, który opuścił pokój rozmów.

Akceptowane parametry: SessionDisconnectEvent disconnectEvent.

Zwracane wartości: Brak.

- ## WebSocketConfig:

Klasa konfiguracji do konfigurowania brokera komunikatów WebSocket.

Metoda: configureMessageBroker

Konfiguruje brokera komunikatów do komunikacji przez WebSocket.

Akceptowane parametry: MessageBrokerRegistry config

Zwracane wartości: Brak.

Metoda: registerStompEndpoints

Rejestruje punkty końcowe Stomp dla WebSocket.

Akceptowane parametry: StompEndpointRegistry registry.

Zwracane wartości: Brak.

Metoda: configureWebSocketTransport

Konfiguruje ustawienia transportu WebSocket, w tym limity rozmiaru wiadomości.

Akceptowane parametry: WebSocketTransportRegistration registration

Zwracane wartości: Brak.

- ## ChatController:

Kontroler obsługujący wiadomości czatu przez WebSocket.

Metoda: sendMessage

Obsługuje wysyłanie wiadomości do czatu. Podczas wysyłania obrazu konwertuje go do formatu Base64.

Akceptowane parametry: String userDestination, Message message

Zwracane wartości: Message

Metoda: addUser

Przetwarza dodanie nowego użytkownika do pokoju rozmów. Wysyła wiadomość o dołączeniu nowego użytkownika.

Akceptowane parametry: Message wiadomość, SimpMessageHeaderAccessor headerAccessor

Zwracane wartości: Message

Metoda: leaveChat

Obsługuje opuszczenie pokoju czatu przez użytkownika. Wysyła wiadomość, że użytkownik opuścił czat.

Akceptowane parametry: Wiadomość message

Zwracane wartości: Message

- ## FormController:

Kontroler obsługujący żądania związane z uwierzytelnianiem.

Metoda: showIndexForm

Wyświetla formularz logowania na stronie z adresem /index.

Akceptowane parametry: Brak.

Zwracane wartości: String (nazwa widoku).

- ## Message:

Klasa reprezentująca encję wiadomości czatu.

Metoda: setType

Ustawia typ wiadomości na podstawie przekazanego ciągu znaków.

Akceptowane parametry: String type.

Zwracane wartości: Brak.

Metoda: getContentType

Zwraca typ zawartości wiadomości.

Akceptowane parametry: Brak.

Zwracane wartości: String

Metoda: setContentType

Ustawia typ zawartości wiadomości.

Akceptowane parametry: String contentType.

Zwracane wartości: Brak.

- ## MessageType:

Wyliczenie reprezentujące różne typy wiadomości czatu.

- ## ChatApplication:

Główna klasa aplikacji.

Metoda: main

Uruchamia aplikację Spring Boot.

Opis: Uruchamia aplikację Spring Boot.

Akceptowane parametry: String[] args

Zwracane wartości: Brak.

- **Kod HTML jest przykładem prostego internetowego interfejsu czatu wykorzystującego Spring Boot i WebSocket. Przyjrzyjmy się głównym komponentom i funkcjonalności tego kodu:**

- **Formularz wyboru nazwy użytkownika (`#username-page`):**

  - Żąda od użytkownika wprowadzenia jego nazwy.

  - Zawiera formularz, w którym użytkownik wprowadza nazwę i może dołączyć do pokoju rozmów.

- **Strona czatu (`#chat-page`):**

  - Pojawia się po pomyślnym wprowadzeniu nazwy przez użytkownika.

  - Zawiera tytuł "Spring WebSocket Chat Demo", obszar do wyświetlania wiadomości czatu oraz formularz do wysyłania nowych wiadomości.

  - Zawiera przycisk "Opuść" umożliwiający wyjście z pokoju czatu.

- **JavaScript:**

  - Zmienne i elementy interfejsu:

  - `usernamePage`, `chatPage`, `usernameForm`, `messageForm`: Elementy interfejsu.

  - `stompClient`: Klient Stomp do przesyłania wiadomości STOMP przez WebSocket.

  - `username`: Przechowuje nazwę użytkownika.

  - `colors`: Tablica kolorów do wyświetlania awatarów użytkowników.

  - `connectingElement`: Element wyświetlający status połączenia.

  - Funkcje:

  - `connect(event)`: Obsługuje próbę połączenia z serwerem WebSocket po wprowadzeniu nazwy użytkownika.

  - `onConnected()`: Obsługuje udane połączenie z serwerem.

  - `onError(error)`: Obsługuje błąd połączenia z serwerem.

  - `sendMessage(event)`: Wysyła wiadomość do serwera.

  - `leaveChat()`: Obsługuje wylogowanie użytkownika z pokoju czatu.

  - `onMessageReceived(payload)`: Obsługuje otrzymanie nowej wiadomości z serwera.

  - Dodatkowe funkcje do obsługi różnych typów wiadomości:

  - `handleImageMessage(file)`: Obsługuje wysyłanie wiadomości obrazkowej.

  - `handleTextMessage(messageContent)`: Obsługuje wysyłanie wiadomości tekstowej.

  - `handleImageMessageDisplay(message, messageElement)`: Wyświetla otrzymany obraz na czacie.

  - `handleJoinMessageDisplay(message, messageElement)`: Wyświetla wiadomość o dołączeniu nowego użytkownika.

  - `handleLeaveMessageDisplay(message, messageElement)`: Wyświetla wiadomość o odejściu użytkownika.

- `handleTextMessageDisplay(message, messageElement)`: Wyświetla wiadomość tekstową na czacie.

- Inne funkcje:

- `getAvatarColor(messageSender)`: Generuje kolor awatara użytkownika na podstawie jego nazwy.

Użyte biblioteki:

- Thymeleaf: Do szablonowania kodu HTML.
- Bootstrap CSS: Do stylizacji interfejsu.
- SockJS i Stomp.js: Do przesyłania wiadomości przez WebSocket.

- **CSS**

- Zeruje style dla HTML i body:

- Ustawia wysokość na 100% dla HTML i body.

- Zabrania przewijania, aby uniknąć pasków przewijania.

- Ogólne style treści:

- Ustawia wspólne style dla body, takie jak czcionka, kolor tekstu, kolor tła.

- Klasy do czyszczenia i ukrywania elementów:

- Klasa .clearfix służy do usuwania elementów z przepływu.

- Klasa .hidden ukrywa element.

- Style dla formularzy:

-.form-control stylizuje pola tekstowe formularzy.

-.form-group dodaje wcięcia między grupami formularzy.

-.input stylizuje dane wejściowe.

- Style dla przycisków:

- Stylizuje ogólny wygląd przycisków.

- button.default, button.primary, button.accent zapewniają style dla różnych typów przycisków.

- Style dla strony wyboru nazwy użytkownika:

- Stylizuje elementy na stronie wyboru nazwy użytkownika.

- Style dla strony czatu:

- Stylizuje elementy na stronie czatu, takie jak kontener czatu, obszar wiadomości i dodatkowe style dla różnych elementów.

- Media Queries for Adaptive Design:

- Zawiera style zastosowane dla małych i bardzo małych ekranów.

- **Sekcja nagłówka (`<head>`):**

  - Ustawia ustawienia strony, w tym skalowanie i nagłówek.

  - Pobiera lokalne style i style Bootstrap z CDN.

- **Sekcja Body (`<body>`):**

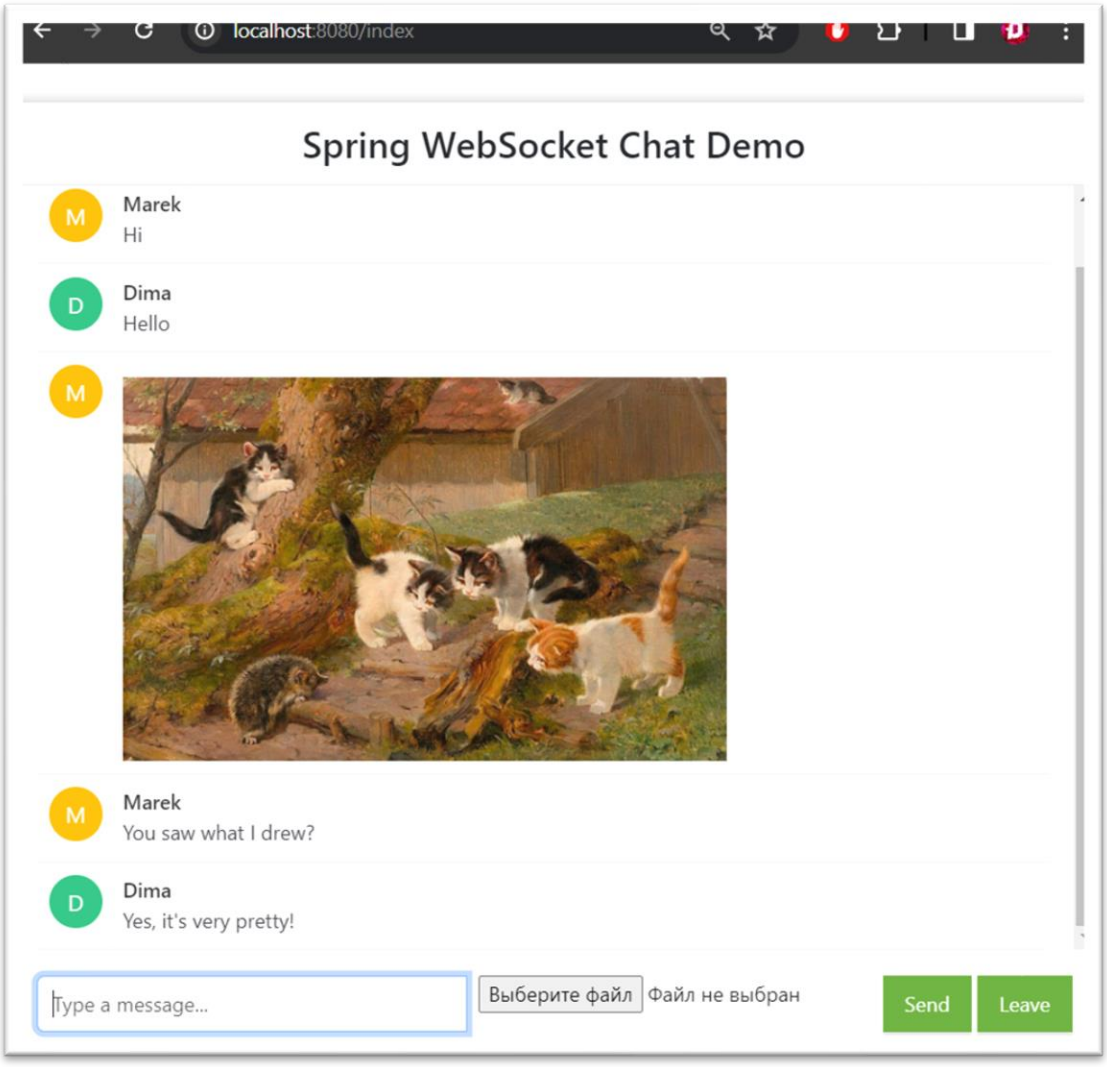
  - Zawiera formularz wyboru nazwy użytkownika i blok czatu.

  - Zawiera skrypty JavaScript do obsługi zdarzeń i interakcji WebSocket.

  - Ładuje biblioteki SockJS i Stomp.js.

5. **Informacje na temat ilości pracy włożonej przez poszczególnych członków zespołu w tworzenie projektu**

   Projekt wykonała jedna osoba, najtrudniejszą częścią projektu było przemyślenie planu, wiele przeróbek i praca z Soketami.



a