|  |  |  |  |
| --- | --- | --- | --- |
| logo_udlap    **EDEI** | **Reporte Practica WebSocket Parte 1** | | |
| **DEPARTAMENTO** | Computación, Electrónica y Mecatrónica | **MATERIA** | Sistemas Distribuidos |
| **PROFESOR** | Dr Ofelia Cervantes | **PERIODO** | Primavera 2015 |

Introducción

En el desarrollo de estos documentos, se presentaran los detalles de implementación de la práctica correspondiente a websockets, haciendo uso del sistema de actores.

Los actores son entidades que se ejecutan concurrentemente y se envían mensajes asíncronos en ellos, el modelo de actores está planeado en base a la idea de que los actores no comparten el estado y que de esta manera no hay problema de deadlock para acceder al estado.

Objetivo

El objetivo de esta práctica es implementar un chat anónimo en modo local, utilizando sistema de actores de Akka atravez de Websockets con playFramework.

Desarrollo

Para iniciar con la práctica debemos de descargar el proyecto de la página de Git del curso, colocarlo en una carpeta estratégica en nuestro equipo y atreves de ventana de comandos debemos de ejecutar el comando “activator run”, dentro de la carpeta del proyecto, esto nos permitirá iniciar el servidor donde se va ejecutar el chat, la aplicación por el momento cuenta con errores que deberán ser resueltos durante el trascurso de nuestra práctica.

La segunda parte de la práctica es corregir los errores que se presentaron al iniciar el proyecto, el primer paso para ello es definir un servicio Rest **assets/javascripts/index.js** agregando el siguiente código:

GET /ws controllers.Application.ws

Posteriormente debemos de copiar los siguientes fragmentos de código, el primero será el encargado de recibir los datos, el segunda parte del código deberá enviar el mensaje.

Código de recibir:

var ws;

ws = new WebSocket($("body").data("ws-url")); //Se toma La URL del Tag en Body

ws.onmessage = function(event) {

var message;

message = JSON.parse(event.data);

switch (message.type) {

case "message":

return $("#board tbody").append("<tr><td>"+ message.msg + "</td></tr>");

default:

return console.log(message);

}

};

Código de enviar:

$("#msgform").submit(function(event) {

event.preventDefault();

console.log($("#msgtext").val());

ws.send(JSON.stringify({ //Enviar el mensaje en el formulario como JSON.

msg: $("#msgtext").val()

}));

return $("#msgtext").val("");

});

Con esto debemos de poder ejecutar el servicio web en su totalidad el chat debe funcionar por completo, por lo que ahora dado un vector de string, debemos de enviar cada elemento contenido en el vector utilizando una función que los envié cada 5 segundos.

Vector:

var speech = ["Hello there!","Welcome to the world of Pokémon!",

"My name is Oak!","People call me the Pokémon Prof!",

"This world is inhabited by creatures called Pokémon!",

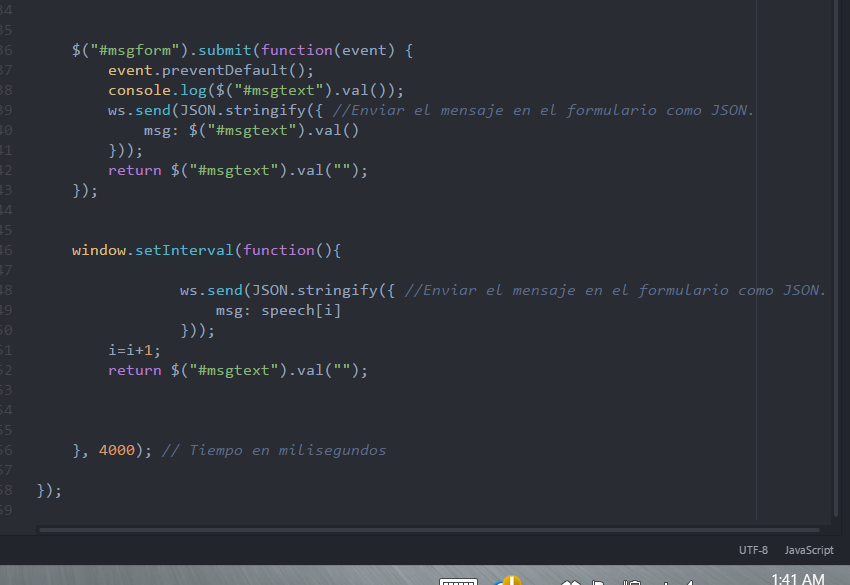
"For some people, Pokémon are pets",

"Other use them for fights",

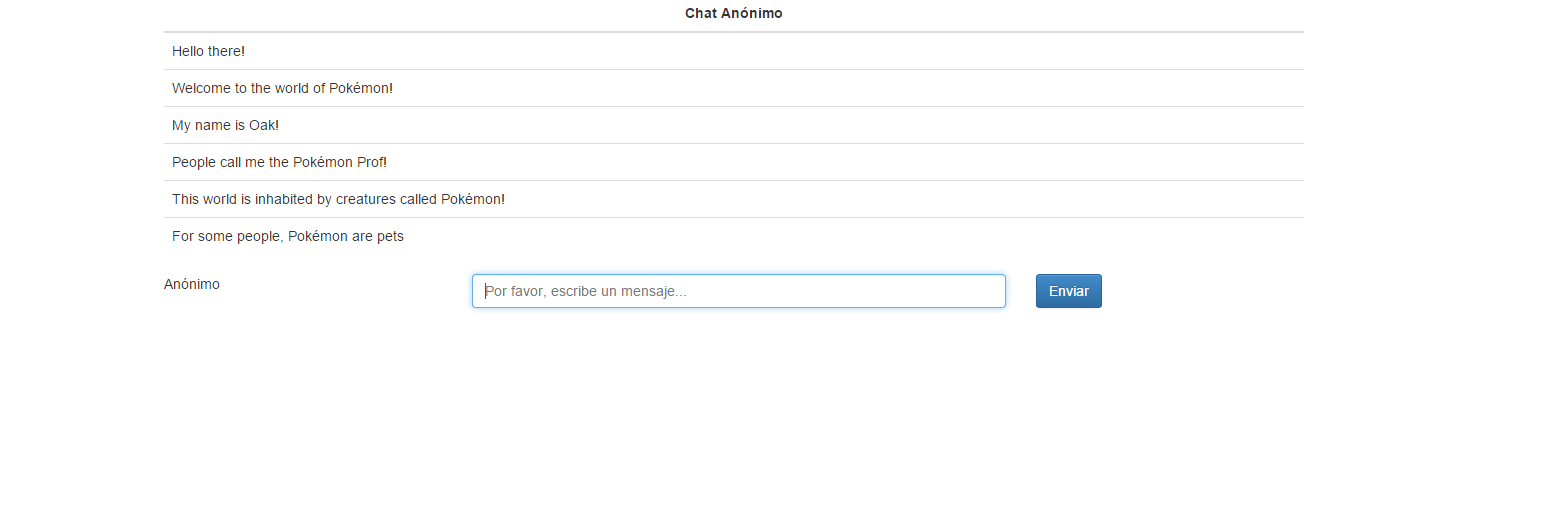
"Myself… I study Pokémon as a profession",

"First, what is your name?"];

Código de la implementación:



Aplicación



Conclusiones

En esta práctica se comprendieron los temas de websockets dentro del ámbito distribuido, los conceptos teóricos aprendidos durante la clase se pusieron en práctica durante el desarrollo de esta práctica, básicamente estos conceptos ayudaron a comprender lo que se estaba llevando a cabo en el código del proyecto que implementaría nuestra aplicación de chat anónimo.

La práctica se completó al 100%, no obstante falta mejorar el desarrollo de aplicaciones utilizando websockets y comprender des los detalles de implementación que se están llevando a cabo en el código.

Preguntas

* ¿**Qué es WebSocket?**
  + Es una tecnología que permite una conexión bidireccional sobre un mismo socket, está diseñada para navegadores y servicios web.
* **¿Cuál es la diferencia entre WebSocket y Servicios REST?**
  + La diferencia es tipo de protocolo en el que están basados, REST está basado en comunicación cliente servidor, mientras que websocket el cliente está conectado al servidor atreves de un conector.
* **¿Cómo se establece una conexión con WebSocket?**
  + el cliente manda una negociación de tipo websocket y el servidor manda una respuesta a la negociaion.
* **Explica la integración de datos en la interfaz utilizando JavaScript.**
* **¿Cómo se implementa el patrón MVC en esta aplicación?**
  + **¿Qué se hace en el controlador?**
    - El controlador en esta aplicación es el encargado de enviar el mensaje, mantener la conexión con el cliente y estar al pendiente de cualquier mensaje que se envié para actualizar la vista o el view,
  + **¿Cómo se implementa la vista?**
    - El view es la página que se muestra en nuestro navegador y esta implementada con html y JavaScrip.