stucom
Formación
Profesional

Memòria Activitat 10 Web Socket

Hajjoune García, Nabil
2o CFGS Desenvolupament d'Aplicacions Web (DAW2)
10/05/24

Índex

Creació Web Socket i usuaris prèviament connectats.

Quan si intenta fer Log In amb un usuari i aquest usuari està registrat, llavors creem la connexió amb Web Socket, entre el servidor i aquell client.

```
if (!resp.ok) {
    throw new Error('Request failed');
}
resp.json().then(
    function (respJson) {
        if (respJson.exist) {
            socket = new WebSocket("ws://localhost:8090");
            socket.onmessage = function (data) {

                let userConnectedDisplay = document.getElementById("usersConnected").getElementsByTagName("div");
                let dataUser = JSON.parse(data.data);
                console.log(dataUser, "data");
```

Després, una vegada s'ha establert la connexió, enviem l'usuari i la contrasenya per guarda la connexió. També aprofito per demanar al servidor la llista d'usuaris connectats

```
socket.onopen = function (evt) {
    info.innerHTML = "Login successful, with user: " + nick;
    //checking connexion is ready
    if (socket.readyState === WebSocket.OPEN) {
        socket.send(JSON.stringify({ "nick": nick, "pass": pass }));
    }
    socket.send(JSON.stringify({ "request": "getConnectedUsers" }));
};
```

Amb addConn guardem la connexió si no està a l'array.

```
//add conexion
userData.conn = conn;
addConn(connexions, userData.nick, userData.pass, userData.conn);
```

```
function addConn(connexions, nick, pass, conn) {

  const existUser = connexions.some(user => user.nick === nick && user.pass === pass);

  if (!existUser) {
    connexions.push({ nick: nick, pass: pass, conn: conn });
    console.log(`Usuario ${nick} añadido.`);
  }
}
```

Retornem l'array actualitzat.

```
//send users connected first time log in
if (userData.request === "getConnectedUsers") {
  const connectedUsers = connexions.map(conn => ({ nick: conn.nick, pass: conn.pass }));
  conn.send(JSON.stringify({ connectedUsers: connectedUsers }));
}
```

Creem els nous usuarios a la vista del client

```
// printing all users connected when first log in
if (dataUser.connectedUsers) {
    let usersConnectedElement = document.getElementById("usersConnected");
    usersConnectedElement.innerHTML = '';

    if (Array.isArray(dataUser.connectedUsers)) {
        for (let i = 0; i < dataUser.connectedUsers.length; i++) {
            let user = dataUser.connectedUsers[i];
            if (user.nick) {
                let divToAppend = document.createElement("div");
                divToAppend.setAttribute("data-nick", user.nick);
                divToAppend.setAttribute("data-pass", user.pass);
                divToAppend.textContent = user.nick;
                divToAppend.addEventListener('click', function () {
                    let input = document.getElementById("inputMsg");
                    input.value = "[@" + this.getAttribute('data-nick') + "]";

                    let nickClick = this.getAttribute('data-nick');
                    let passClick = this.getAttribute('data-pass');
                    //adding info to a div to have the last user click, necessary to start the game
                    document.getElementById('usersClicked').setAttribute("data-nick", nickClick);
                    document.getElementById('usersClicked').setAttribute("data-pass", passClick);
                })
                usersConnectedElement.appendChild(divToAppend);
            }
        }
    }
}
```

Si el client tanca la connexió amb el servidor, el servidor ha de desvincular l'usuari de la sala i avisar a la resta.

Si l'usuari està registrat, enviem un missatge al servidor amb les credencials de l'usuari.

```
fetch("http://localhost:3000/logOut", {
    method: "POST",
    body: form,
}).then(function (resp) {
    if (!resp.ok) {
        throw new Error('Request failed');
    }
    resp.json().then(
        function (respJson) {
            if (respJson.exist) {
                socket.send(JSON.stringify({ "nick": nick, "pass": pass, "close": true }));
            } else {
                info.innerHTML = respJson.message;
            }
        }
    )
```

En el servidor eliminem la connexió.

```
//to close connexion
let connToDelete;
let addUser = true;

if (userData.close) {
  connToDelete = connexions.find(conexion => conexion.nick === userData.nick && conexion.pass === userData.pass);
  if (connToDelete) {
    addUser = false;
    connToDelete.conn.close();
    connexions.splice(connexions.indexOf(connToDelete), 1);
  }
}
```

També enviem a totes les connexions si l'usuari s'ha de mostrar o no.

```
//send all conexiones new user logged in
connexions.forEach(conn => {
  conn.conn.send(JSON.stringify({ "nick": userData.nick, "pass": userData.pass, "addUser": addUser }));
});
```

En el client quan s'activa l'esdeveniment onclose, eliminem l'usuari de la vista al mateix usuari.

```
socket.onclose = function (evt) {
    info.innerHTML = "Log out successful, with user: " + nick;

    let userConnectedDisplay = document.getElementById("usersConnected").getElementsByTagName("div");
    let userConnectedArray = Array.from(userConnectedDisplay);

    for (let i = 0; i < userConnectedArray.length; i++) {
        const element = userConnectedArray[i];
        let nickUserDisplay = element.getAttribute("data-nick");
        let passUserDisplay = element.getAttribute("data-pass");

        // check if user exist
        if (nick === nickUserDisplay && pass === passUserDisplay) {
            element.remove();
            break;
        }
    }
}
```

Eliminem de la vista a la resta d'usuaris.

```
let userConnectedArray = Array.from(userConnectedDisplay);
//removing any repeated user in the display before uploading
for (let i = 0; i < userConnectedArray.length; i++) {
    const element = userConnectedArray[i];
    let nickUserDisplay = element.getAttribute("data-nick");
    let passUserDisplay = element.getAttribute("data-pass");
    if (dataUser.nick == 'undefined' || nickUserDisplay == 'undefined') {
        element.remove();
    }

    if (dataUser.nick === nickUserDisplay && dataUser.pass === passUserDisplay) {
        if (!dataUser.addUser) {
            element.remove();
        }
        break;
    }
}
```

Quan tanquem la finestra, tanquem la connexió i enviem l'usuari desconnectat a tota la resta.

```
conn.on('close', (evt) => {


  let  connToDelete;
  // connToDelete = connexions.find(conexion => conexion.conn === conn);
  for(let k=0; k< connexions.length;k++){
    if(connexions[k].conn==conn){
      connToDelete=connexions[k];
      break;
    }
  }

  connexions.forEach(connect => {
    connect.conn.send(JSON.stringify({ "nick": connToDelete.nick, "pass": connToDelete.pass, "addUser": false }));
  });
  conn.close();
  connexions.splice(connexions.indexOf(conn), 1);

  console.log("Tancada la connexió");
});
```

L'usuari ha de poder enviar un missatge privat a un altre usuari i a tots els usuaris.

Enviem al servidor el missatge de: qui envia el missatge, a qui i el mateix missatge.

```
document.getElementById("btnSend").addEventListener('click', function () {
    let input = document.getElementById("inputMsg");
    let nick = extractContentNick(input.value);
    let message = extractContentMsg(input.value);
    let nickAuthor = document.getElementById("nick").value;
    let chat = document.getElementById("chat");
    let divToAppend = document.createElement("div");
    divToAppend.innerHTML = "To " + nick + ": " + message;
    chat.appendChild(divToAppend);
    socket.send(JSON.stringify({ "message": message, "nickMsg": nick, "nickAuthor": nickAuthor, "send": true }));
});
```

En el servidor enviem el missatge a la persona indicada si no és el paràmetre 'all'. Si no enviem a totes les connexions el missatge.

```
//send message
if (userData.send) {
  let messageObj = { from: userData.nickAuthor, message: userData.message };
  let userToSendMsg = connexions.find(conn => conn.nick === userData.nickMsg);
  if (userData.nickMsg !== 'all') {
    if (userToSendMsg) {
      userToSendMsg.conn.send(JSON.stringify(messageObj));
    }
  } else {
    connexions.forEach(conn => {
      if (typeof conn.nick !== 'undefined' && conn.nick !== 'undefined') {
        conn.conn.send(JSON.stringify(messageObj));
      }
    });
  }

}
```

Imprimim el missatge.

```
//handling message
if (dataUser.message) {
    let chat = document.getElementById("chat");
    let divToAppend = document.createElement("div");
    divToAppend.innerHTML = "From " + dataUser.from + ": " + dataUser.message;
    chat.appendChild(divToAppend);
}
```

Un client ha de poder enviar una petició per començar un joc de 3 en ratlla amb un altre client i poder acceptar o denegar la petició

Enviem al servidor la petició de joc: qui demana jugar → contra qui es vol jugar.

```
document.getElementById("btnStart").addEventListener('click', function () {
    if (!matchOnGoing) {
        let userClicked = document.getElementById("usersClicked");
        let nickOpponent = userClicked.getAttribute('data-nick');
        let passOpponent = userClicked.getAttribute('data-pass');
        let nickChallenger = document.getElementById("nick").value;
        let passChallenger = document.getElementById("pass").value;
        if (nickOpponent == '' && passOpponent == '') {
            document.getElementById('errorNotify').innerHTML = "Before starting a match, choose an opponent";
        } else {
            document.getElementById('errorNotify').innerHTML = "";
            let infoMatchPreview = document.getElementById("infoMatch");
            infoMatchPreview.classList.remove('d-none');
            infoMatchPreview.classList.add('d-flex');
            infoMatchPreview.innerHTML = "Waiting for <b> " + nickOpponent + " </b> to accept the match";
            matchOnGoing = true;
            socket.send(JSON.stringify({ "match": true, "nickChallenger": nickChallenger, "passChallenger":
            passChallenger, "nickOpponent": nickOpponent, "passOpponent": passOpponent }));
        }
```

Una vegada en el servidor, enviem la petició de joc a l'usuari oponent.

```
//manage match
if (userData.match) {
  connToMatch = connexions.find(conexion => conexion.nick === userData.nickOpponent && conexion.pass === userData.passOpponent);
  if (connToMatch) {
    connToMatch.conn.send(JSON.stringify({ "match": true, "nickChallenger": userData.nickChallenger,
    "passChallenger": userData.passChallenger, "nickOpponent": userData.nickOpponent, "passOpponent": userData.passOpponent }));
  } else {
    console.log("not found");
  }
}
```

Mostrem el missatge per escollir si es vol jugar. Depenen al botó que es premi s'envia al servidor si s'ha acceptat o no jugar.

```
//handling decision
btnAccept.addEventListener('click', function () {
    btnAccept.classList.remove('d-flex');
    btnAccept.classList.add('d-none');
    btnDecline.classList.remove('d-flex');
    btnDecline.classList.add('d-none');
    matchOnGoing = true;
    infoMatch.innerHTML = "Turn: " + dataUser.nickChallenger;
    showPlayersPlaying(dataUser.nickChallenger, dataUser.passChallenger, dataUser.nickOpponent, dataUser.passOpponent);
    socket.send(JSON.stringify({ "acceptedMatch": true, "nickChallenger":
    dataUser.nickChallenger, "passChallenger": dataUser.passChallenger, "nickOpponent":
    dataUser.nickOpponent, "passOpponent": dataUser.passOpponent }));
});

btnDecline.addEventListener('click', function () {
    infoMatch.classList.remove('d-flex');
    infoMatch.classList.add('d-none');
    btnAccept.classList.remove('d-flex');
    btnAccept.classList.add('d-none');
    btnDecline.classList.remove('d-flex');
    btnDecline.classList.add('d-none');
    socket.send(JSON.stringify({ "acceptedMatch": false, "nickChallenger":
    dataUser.nickChallenger, "passChallenger": dataUser.passChallenger, "nickOpponent":
    dataUser.nickOpponent, "passOpponent": dataUser.passOpponent }));
});
```

Si la partida ha estat acceptada s'envia al jugador desafiant que ha estat acceptada i es pot començar el joc. De la mateixa forma si s'ha denegat s'informa igualment.

```
if (userData.acceptedMatch) {
  console.log(userData, "acceptedMatch");
  connToMatchChallenger = connexions.find(conexion => conexion.nick === userData.nickChallenger
    && conexion.pass === userData.passChallenger);
  console.log(connToMatchChallenger, "**");
  connToMatchChallenger.conn.send(JSON.stringify({ "acceptedMatch": true, "nickChallenger": userData.nickChallenger,
  "passChallenger": userData.passChallenger, "nickOpponent": userData.nickOpponent, "passOpponent": userData.passOpponent }));
} else if (userData.acceptedMatch == false) {
  console.log(userData, "Not acceptedMatch");
  connToMatchChallenger = connexions.find(conexion => conexion.nick === userData.nickChallenger
    && conexion.pass === userData.passChallenger);
  console.log(connToMatchChallenger, "?");
  connToMatchChallenger.conn.send(JSON.stringify({ "acceptedMatch": false, "nickChallenger": userData.nickChallenger,
  "passChallenger": userData.passChallenger, "nickOpponent": userData.nickOpponent, "passOpponent": userData.passOpponent }));
}
```

Des del client del desafiant, s'informa de la decisió de l'oponent.

```
if (dataUser.acceptedMatch) {
    let infoMatch = document.getElementById("infoMatch");
    infoMatch.innerHTML = "<b>" + dataUser.nickOpponent + "</b>   has accepted the match<br>Turn: " + dataUser.nickChallenger;
    infoMatch.classList.remove('d-none');
    infoMatch.classList.add('d-flex');
    showPlayersPlaying(dataUser.nickChallenger, dataUser.passChallenger, dataUser.nickOpponent, dataUser.passOpponent);
    turn = dataUser.nickChallenger;
} else if (dataUser.acceptedMatch == false) {
    matchOnGoing = false;
    let infoMatch = document.getElementById("infoMatch");
    infoMatch.classList.remove('d-none');
    infoMatch.classList.add('d-flex');
    infoMatch.innerHTML = "<b>" + dataUser.nickOpponent + "</b>   has decline the match";
}
```

Controlar els torns i l'estat de la partida, informa quan s'acaba

Anem alternant el torn entre els jugadors, enviem el torn del jugador actual i la casella presa.

```
if (matchOnGoing) {
    let indexToPlace;
    if (turn == nickChallenger) {
        cell.innerHTML = "X";
        indexToPlace = cell.getAttribute('data-cell-index');
    } else if (turn == nickOpponent) {
        cell.innerHTML = "O";
        indexToPlace = cell.getAttribute('data-cell-index');
    }
    if (turn.trim() === nickOpponent.trim()) {
        turn = nickChallenger;
    } else if (turn.trim() === nickChallenger.trim()) {
        turn = nickOpponent;
    }
    let infoMatchTurn = document.getElementById("infoMatch");
    infoMatchTurn.classList.remove('d-none');
    infoMatchTurn.classList.add('d-flex');
    infoMatchTurn.innerHTML = "Turn: " + turn;
    socket.send(JSON.stringify({ "turn": turn, "cell": cell.getAttribute('data-cell-index'), "indexToPlace":
    indexToPlace, "nickChallenger": nickChallenger, "passChallenger": passChallenger, "nickOpponent": nickOpponent,
    "passOpponent": passOpponent }));
}
```

Mirem si la partida ha acabat, per si algú ha guanyat o és un empat, enviem el torn actual, els jugadors, la casella presa i si algú ha guanyat.

```
let winner = checkForWinner(board);

console.log(winner);
//check board to find winner
console.log(nextPlayerTurn.nick, "Next player nick");
console.log(currentPlayerTurn.nick, "Current player nick");

connToSendTurnNextPlayer = connexions.find(conexion => conexion.nick === nextPlayerTurn.nick && conexion.pass === nextPlayerTurn.pass);
connToSendTurnCurrentPlayer = connexions.find(conexion => conexion.nick === currentPlayerTurn.nick && conexion.pass === currentPlayerTurn.pass);
console.log(board, "CURRENT BOARD");
connToSendTurnNextPlayer.conn.send(JSON.stringify({ "turn": userData.turn, "board": board, "winner": winner,
"nickChallenger": userData.nickChallenger, "passChallenger": userData.passChallenger, "nickOpponent": userData.nickOpponent,
"passOpponent": userData.passOpponent }));
connToSendTurnCurrentPlayer.conn.send(JSON.stringify({ "turn": userData.turn, "board": board, "winner": winner,
"nickChallenger": userData.nickChallenger, "passChallenger": userData.passChallenger, "nickOpponent": userData.nickOpponent,
"passOpponent": userData.passOpponent }));
```

Funció per comprovar si algú ha guanyat.

```
function checkForWinner(board) {
  const winConditions = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6]
  ];
  let data = null;
  for (const condition of winConditions) {
    const [a, b, c] = condition;
    if (board[a] && board[a] === board[b] && board[a] === board[c]) {
      data = board[a];
    }
  }
  if (!board.includes('')) {
    data = 'Draw';
  }
  return data !== null ? data : null;
}
```

Si algú ha guanyat, informem i imprimim els missatges.

```
//checking board
if(dataUser.winner !== null){
    matchOnGoing = false;
    if(dataUser.winner !== "Draw"){
        let symbol = dataUser.winner == 'X' ? 'X' : 'O';
        if(symbol == 'X'){
            infoMatchTurn.innerHTML = "<b> " + dataUser.nickChallenger + " </b> has won";
        } else {
            infoMatchTurn.innerHTML = "<b> " + dataUser.nickOpponent + " </b> has won";
        }
    } else {
        infoMatchTurn.innerHTML = "<b>Draw</b>";
    }
}
```