LABORATORIO II DE SISTEMAS TELEMÁTICOS

PRACTICA DE LABORATORIO

PRESENTADO A:

Fulvio Yesid Vivas Cantero

PRESENTADO POR:

NESTOR JAIME ALEGRIA ALEGRIA

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA DE ELECTRONICA Y TELECOMUNICACIONES

POPAYÁN-CAUCA

2023

Contenido

INTRODUCCION

Durante el desarrollo de la siguiente práctica se descarga las maquias virtuales, se implementaron diferentes conceptos relacionados con el análisis y la simplificación de configuración de cada programa y como lo podemos representar en la vida cotidiana por medio de un uso específico del cual es una solución a una mejora.

MARCO TEORICO

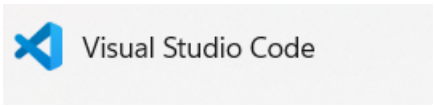
1)Java Script: javaScript es un lenguaje de programación que los desarrolladores utilizan para hacer páginas web interactivas. Desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de JavaScript pueden mejorar la experiencia del usuario de un sitio web. Como lenguaje de scripting del lado del servidor, se trata de una de las principales tecnologías de la World Wide Web. Por ejemplo, al navegar por Internet, en cualquier momento en el que vea un carrusel de imágenes, un menú desplegable “click-to-show” (clic para mostrar), o cambien de manera dinámica los elementos de color en una página web, estará viendo los efectos de JavaScript.

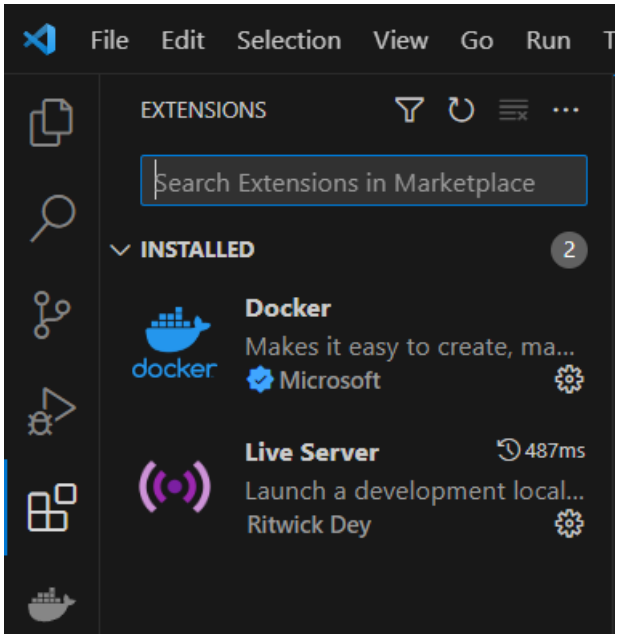
2)Visual Studio Code: Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.
Para tener una idea de la popularidad de Visual Studio Code y la aceptación que ha tenido en el mundo de desarrollo, podemos consultar datos. Según una encuesta realizada por Stack Overflow a más de 80,000 desarrolladores en mayo del 2021, Visual Studio Code es el entorno de desarrollo más usado y con mucha diferencia, un 71.06%. En la siguiente ilustración, puedes ver el top 10.

3) Live server: en Visual Studio Code es una herramienta que nos permite lanzar un servidor de desarrollo local para previsualizar en el navegador lo que estamos escribiendo en nuestro editor de código. Lo interesante de esta herramienta es que la previsualización se actualiza inmediatamente al guardar el proyecto desde el editor. Esto es ideal para revisar los cambios que hacemos en nuestras páginas tanto estáticas como dinámicas.
Para automatizar aún más el proceso, puedes editar la configuración de guardado automático en Visual Studio Code.
Además de live server en Visual Studio Code, existen muchas herramientas que nos permiten visualizar el resultado del código que estamos desarrollando al mismo tiempo que lo desarrollamos. Sin embargo, esta herramienta es la más utilizada cuando escribimos usando VSCode, pues se enfoca en el desarrollo frontend.
Algunas de las características de esta herramienta, además de tener una actualización rápida en el navegador, son el soporte de HTTPS de todos los navegadores, de la extensión de debugger de Chrome y el soporte de archivos para detectar cambios. Además, soporta proxies y un espacio de trabajo multiroot.

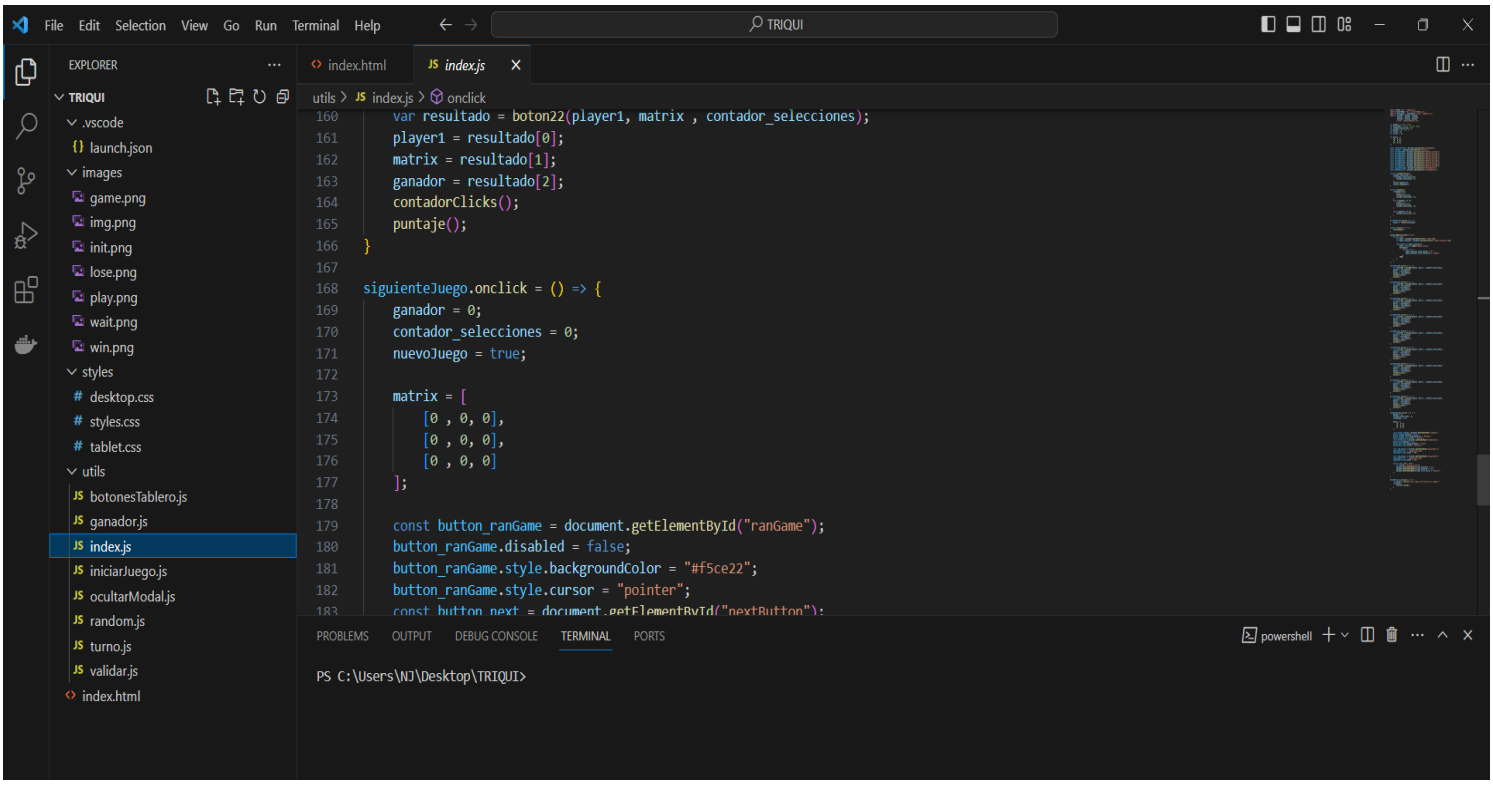
ARCHIVOS

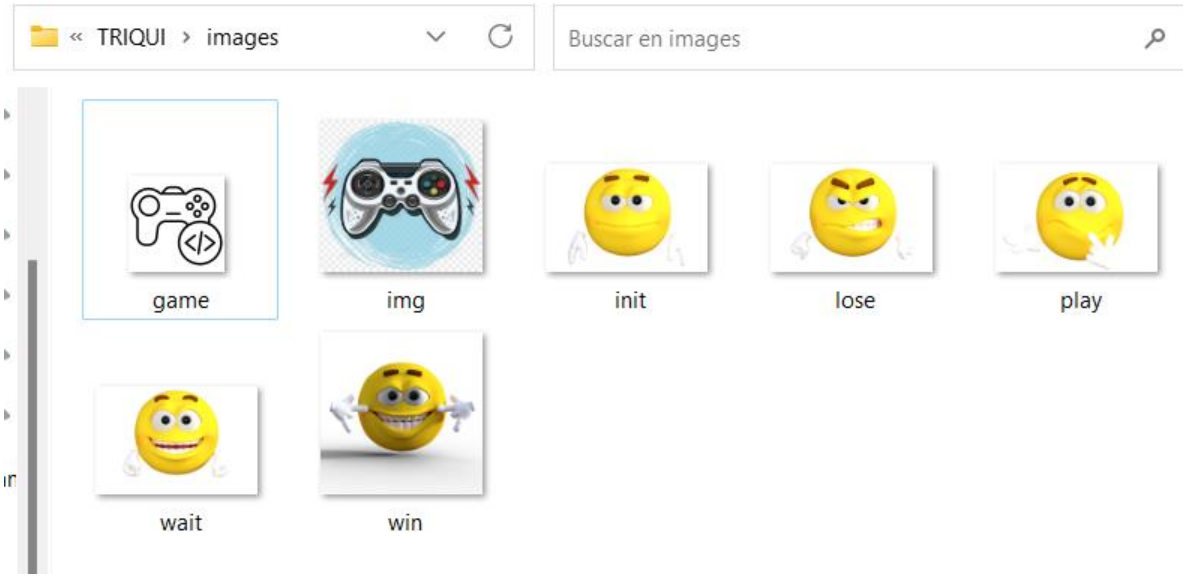
PASO 1: Para hacer el juego de tic tac toe descargamos Visual Studio Code, Docker y Live Server.





PASO 2: Para hacer el juego creamos una carpeta donde va a contener todos los elementos para el funcionamiento correcto de nuestro juego, creamos una carpeta donde va a estar contenida nuestras imágenes y creamos archivos js que son utilizados para hacer páginas web interactivas.





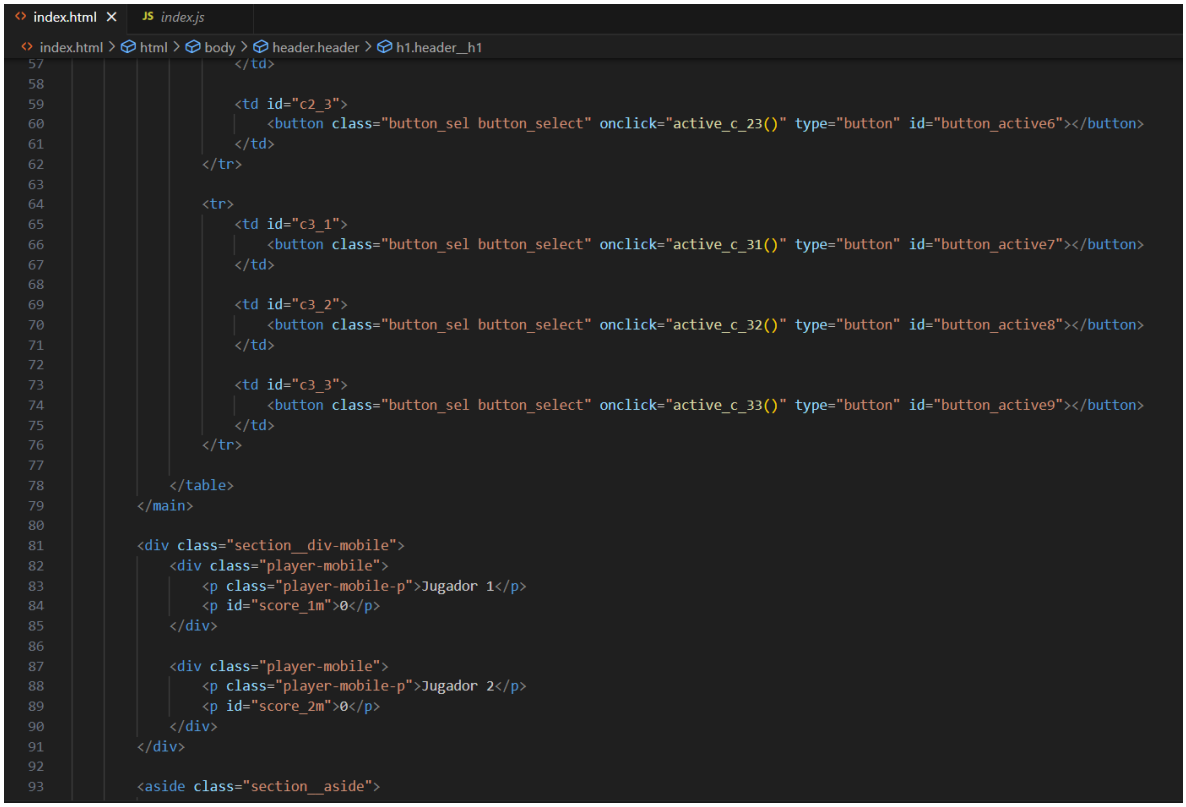
PASO 3: Creamos un archivo HTML para dar forma a aplicaciones web modernas que sean intuitivas, interactivas y fáciles de usar. Al usar la validación simple del lado del cliente, se reduce el tráfico del servidor y mejora la eficiencia general del sitio web.

index.html | index.js

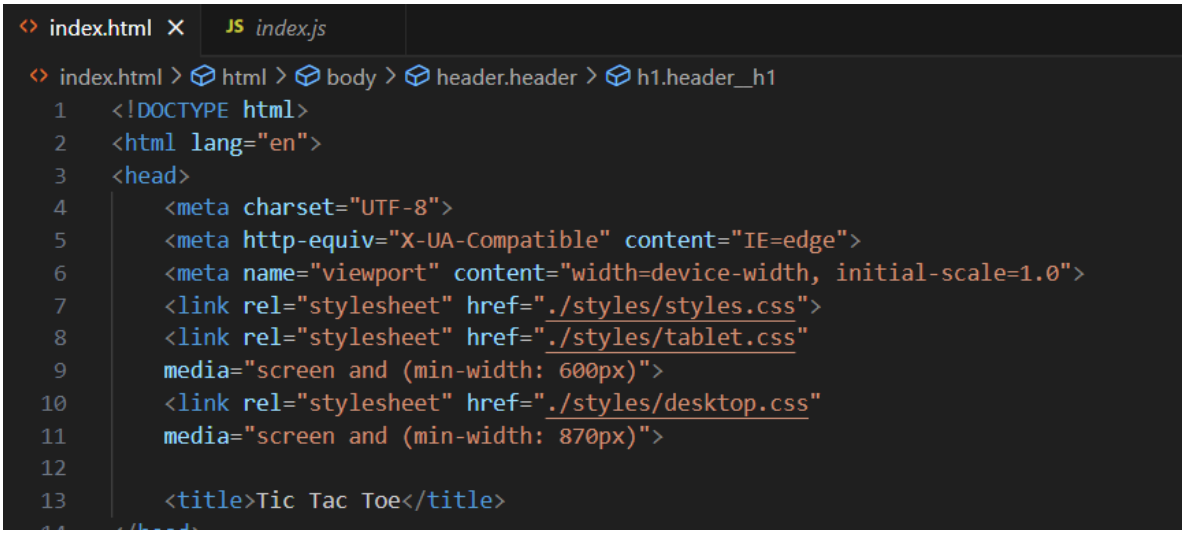
index.html > html > body > header.header > h1.header__h1

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./styles/styles.css">
    <link rel="stylesheet" href="./styles/tablet.css"
    media="screen and (min-width: 600px)">
    <link rel="stylesheet" href="./styles/desktop.css"
    media="screen and (min-width: 870px)">

    <title>Tic Tac Toe</title>
</head>
<body>
    <header class="header">
        <figure class="header_fig">
            <img class="header__fig-icon" src="./images/game.png" alt="Game Icon">
        </figure>
        <h1 class="header__h1">tic tac toe</h1>
    </header>
    <section class="section">
        <aside class="section__aside">
            <h2 class="section__aside-h2" id="player1_title">Jugador 1</h2>
            <picture class="section__aside-picture">
                <img class="emoji" src="./images/init.png" alt="Inicial" id="img_player1">
            </picture>
            <div class="section__div">
                <p class="section__div-p">Puntuación:</p>
                <p class="section__div-number-score" id="score_1">0</p>
            </div>
        </aside>

        <main class="section__main">
            <table class="section__main-table">
                <tr>
                    <td id="c1_1">
```

Se crea la interfaz grafica del juego dando le personalidad con diferentes imágenes para su mejor diseño y la creación de los botones de inicio de juego etc.



PAS0 4: Al inicio de archivo index.html se desarrolla el diseño de la pagina del juego tamaño y se le da una escala que se representa con el siguiente comando: initial-scale=1.0 conseguimos que no se haga zoom sobre el documento. Es bien simple, el contenido de la web no se transformará, ni se agrandará, ni se hará menor.



PASO 5: Se crea el estilo interno de la parte de los jugadores y la tabla como se muestra en los comandos y en que posición de nuestra pagina ban a quedar ubicados.

PASO 6: Se crea el estilo interno de la parte de los jugadores y la tabla como se muestra en los comandos y en que posición de nuestra página ban a quedar ubicados, en este archivo manejamos mas que todo el diseño de los jugadores y los botones con diferentes colores.

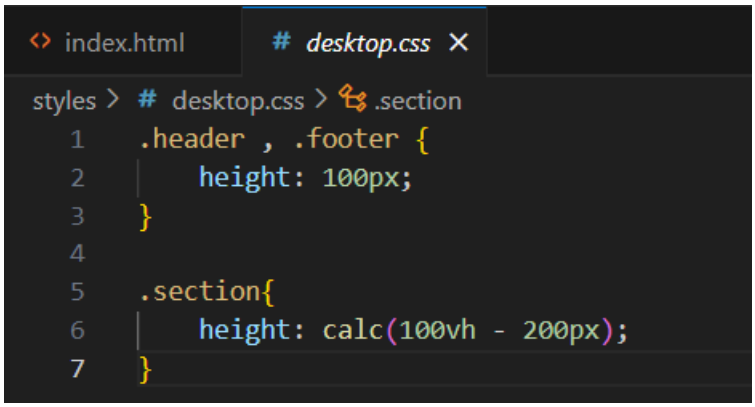
index.html # tablet.css

styles > # tablet.css > .header

```css
.header{
    height: 50px;
}

.header__h1{
    font-size: 4rem;
}

.section{
    flex-direction: row;
    height: calc(100vh - 100px);
}

.section__div-mobile {
    display: none;
}

.section__aside{
    width: 20%;
    height: 100%;
    border: 6px solid white;
    border-radius: 20px;
    background: linear-gradient(#63abf3, black);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-around;
}

.section__aside-h2{
    color: white;
    letter-spacing: 2px;
    font-size: 1.8rem;
    text-align: center;
}
```

Dando como resultado el siguiente diseño.

PASO 6: Esta es la parte del recuadro del triqui, las medidas del tamaño del recuadro.



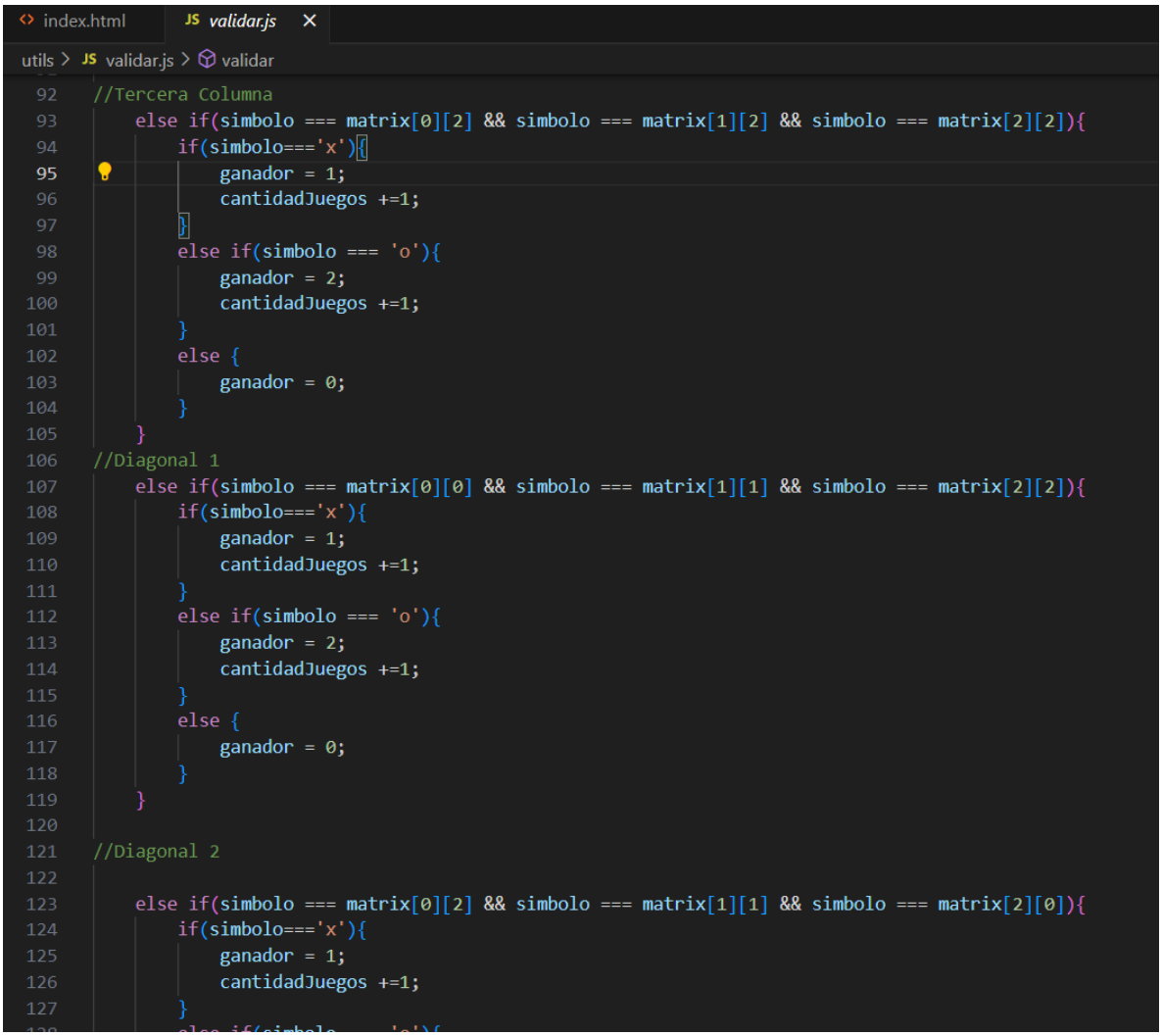
PASO 7: Se crea una clase que es necesaria para cuando se tenga la necesidad de reiniciar el juego.

index.html | iniciarJuego.js

utils > iniciarJuego.js > ...

```
1   import turno from "./turno.js";
2
3   function iniciarJuego(nuevoJuego, player1){
4       var i;
5
6       if(!nuevoJuego){
7           for(i=1 ; i<10 ; i++){
8               let string = "button_active" + i;
9               document.getElementById(string).style.display = 'block';
10          }
11      }
12
13      else {
14          for(i=1 ; i<10 ; i++){
15              let string = "button_active" + i;
16              document.getElementById(string).disabled = false;
17              document.getElementById(string).style.cursor = 'pointer';
18          }
19
20          nuevoJuego = false;
21      }
22
23      turno(player1);
24  }
25
26  export default iniciarJuego;
```

PASO 8: Creamos una clase de validación de la tabla de tic tac toe cuando se señala una de las partes así se horizontal o vertical valide esta acción y pueda colocar la casilla deleccionada.



PASO 9: Se crea una clase que va de la mano con validar al momento de seleccionar la casilla cambia de lugar en este caso a al jugador numero 2.

index.html | turno.js

utils > turno.js > turno

```
function turno(player1){

    if(player1 === true){
        const img_player1 = document.getElementById("img_player1");
        img_player1.src = "./images/play.png";
        img_player1.style.width = '120%';

        const img_player2 = document.getElementById("img_player2");
        img_player2.src = "./images/wait.png";
        img_player2.style.width = '80%';

        document.getElementById("player1_title").style.fontSize = '30px';
        document.getElementById("player2_title").style.fontSize = '18px';
    }
    else {
        const img_player1 = document.getElementById("img_player1");
        img_player1.src = "./images/wait.png";
        img_player1.style.width = '80%';

        const img_player2 = document.getElementById("img_player2");
        img_player2.src = "./images/play.png";
        img_player2.style.width = '120%';

        document.getElementById("player1_title").style.fontSize = '18px';
        document.getElementById("player2_title").style.fontSize = '30px';
    }
}

export default turno;
```

PASO 10: Creamos una clase para contabilizar y validar el ganador de los dos jugadores, cada uno tiene su respectivo código con el tamaño de la x o 0.

index.html | ganador.js

utils > ganador.js > winPlayer1

```
function winPlayer1 (score1){
    const img_player1 =  document.getElementById("img_player1");
    img_player1.src = "./images/win.png";
    img_player1.style.width = '80%';
    const img_player2 =  document.getElementById("img_player2");
    img_player2.src = "./images/lose.png";
    img_player2.style.width = '80%';
    document.getElementById("player1_title").style.fontSize = '18px';
    document.getElementById("player2_title").style.fontSize = '18px';
    const score_1 = document.getElementById("score_1");
    const score_1m = document.getElementById("score_1m");
    score_1.innerText = score1;
    score_1m.innerText = score1;

    const nextButton = document.getElementById("nextButton");
    nextButton.disabled = false;
    nextButton.style.backgroundColor = "#f5ce22";
    nextButton.style.cursor = "pointer";

    for(var i = 1; i < 10 ; i++){
        let string = "button_active"+i;
        document.getElementById(string).disabled = true;
    }

}

function winPlayer2 (score2){
    const img_player1 =  document.getElementById("img_player1");
    img_player1.src = "./images/lose.png";
    img_player1.style.width = '80%';
    const img_player2 =  document.getElementById("img_player2");
    img_player2.src = "./images/win.png";
    img_player2.style.width = '80%';
    document.getElementById("player1_title").style.fontSize = '18px';
    document.getElementById("player2_title").style.fontSize = '18px';
    const score_2 = document.getElementById('score_2');
    const score_2m = document.getElementById('score_2m');
```

PASO 11: Creamos una clase para el color de cada uno de los caracteres y su respectiva casilla en la que se señale.

PASO 12: Se crea una clase para el cambio de jugador cuando señala la casilla y las veces que el jugador va ganando.

index.html | random.js

utils > random.js > ...

```
1  import iniciarJuego from "./iniciarJuego.js";
2
3  function random(nuevoJuego){
4      var player1;
5      const ranMatch = Math.floor(Math.random()*2) + 1; // 1 o 2
6      const playerInit = document.getElementById("playerInit");
7
8      if(ranMatch === 1){
9          playerInit.innerText = "Inicia el Jugador 1";
10         player1 = true;
11     }
12     else {
13         playerInit.innerText ="Inicia Jugador 2";
14         player1 = false;
15     }
16
17     let modal = document.querySelectorAll(".modal")[0];
18     let modal_container = document.querySelectorAll(".modal-container")[0];
19     modal_container.style.opacity = "1";
20     modal_container.style.visibility = "visible";
21     modal.classList.toggle("modal-close");
22
23     iniciarJuego(nuevoJuego , player1); /**********/
24
25     const button_ranGame = document.getElementById("ranGame");
26     button_ranGame.disable = true;
27     button_ranGame.style.backgroundColor = "gray";
28     button_ranGame.style.cursor = "default";
29     const button_reset = document.getElementById("cleanButton");
30     button_reset.disabled = false;
31     button_reset.style.backgroundColor = "#f5ce22";
32     button_reset.style.cursor = "pointer";
33     return player1;
34
35 }
36
37 export default random;
```

EXPERIENCIAS DE LA PRÁCTICA

Durante el desarrollo de la práctica, nos permite afianzar los conocimientos vistos en clase, implementándolas de manera interactiva y practica con el Lubuntu. Al transcurso del desarrollo miramos el proceso de descarga, configuración del programa y poder así llevar a cabo una óptima simulación con los resultados esperados. Durante el proceso de instalación y verificación del estado del programa, se permite desarrollar diferentes nuevas habilidades para solucionar los problemas más frecuentes al momento de querer implementarlo.

Resultados esperados

1. Funcionalidad del juego al momento de iniciar y jugar con un rival.

2. Mediante las herramientas ofrecidas se diseñó y se realizó un juego interactivo.

3.Manejo total del lenguaje gracias al los tutoriales y guías que nos ofrecen.

CONCLUSIONES

1. Durante el desarrollo de la práctica se realiza la creación de un juego muy conocido gracias a visual studio code y live server.
2. Se puede apreciar diferentes diseños al momento de realizar la página para poder distinguir diferentes funcionalidades.

3. Se aprende un lenguaje nuevo de escritura de código.

4. Desarrollo de páginas web muy fáciles de hacer gracias a los tutoriales.

Lecciones aprendidas

1. En la primera parte de la práctica se trata de identificar los comandos y para que se pueden utilizar.
2. A medida que se va solucionando el laboratorio se identifican mala redacción al momento de desarrollar el fichero porque tiene su respectiva nomenclatura.
3. Manejo de directorios, creación de archivos dentro de ellos y manejo de repositorios al momento de compartir con los demás integrantes.

Recomendaciones

1. Se recomienda tener una base de datos o lista de comandos, agilizando así el uso de estos.
2. Se recomienda tener especial atención con el nombre que se le da a usuarios, grupos, directos archivos etc. para así evitar confusiones en el desarrollo del trabajo.
3. Se recomienda manejar un orden a la hora de configurar el juego, para así evitar confusiones o posibles errores dentro del juego.

BIBLIOGRAFIA

- https://aws.amazon.com/es/what-is/javascript/#:~:text=JavaScript%20es%20un%20lenguaje%20de,usuario%20de%20un%20sitio%20web.
- https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/
- https://keepcoding.io/blog/live-server-en-visual-studio-code/#:~:text=3%20%C2%BFQu%C3%A9%20sigue%3F-,%C2%BFQu%C3%A9%20es%20live%20server%20en%20Visual%20Studio%20Code%3F,en%20nuestro%20editor%20d