

**UNIVERSIDAD TÉCNICA DE MACHALA**

Maestría en Software

**Asignatura:**  
Desarrollo de Aplicaciones Web  
  
  
**Tema:**

**Ejecutar Tutorial REACT**

**Docente:**

Ing. Nelson Piedra

**Estudiante:**

Ing. Jimmy Fernando Castillo Crespín

2021-2022

**Informe técnico de la experiencia.**

**Construir un juego de tic-tac-toe interactivo con React**

### Prerrequisitos

* Familiaridad con HTML y JavaScript ES6.
* Familiaridad con conceptos de programación como funciones, objetos, arrays, y en menor medida, clases.
* Instalar nodeJS

**Crear proyecto en react**

npx create-react-app my-app

### Herramientas de desarrollo

### La extensión de React Devtools para [Chrome](https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en) y [Firefox](https://addons.mozilla.org/en-US/firefox/addon/react-devtools/) te permite inspeccionar el árbol de componentes de React con tus herramientas de desarrollo del navegador.

### 

Código fuente:

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

function Square(props) {

  return (

    <button className="square" onClick={props.onClick}>

      {props.value}

    </button>

  );

}

class Board extends React.Component {

  renderSquare(i) {

    return (

      <Square

        value={this.props.squares[i]}

        onClick={() => this.props.onClick(i)}

      />

    );

  }

  render() {

    return (

      <div>

        <div className="board-row">

          {this.renderSquare(0)}

          {this.renderSquare(1)}

          {this.renderSquare(2)}

        </div>

        <div className="board-row">

          {this.renderSquare(3)}

          {this.renderSquare(4)}

          {this.renderSquare(5)}

        </div>

        <div className="board-row">

          {this.renderSquare(6)}

          {this.renderSquare(7)}

          {this.renderSquare(8)}

        </div>

      </div>

    );

  }

}

class Game extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      history: [

        {

          squares: Array(9).fill(null)

        }

      ],

      stepNumber: 0,

      xIsNext: true

    };

  }

  handleClick(i) {

    const history = this.state.history.slice(0, this.state.stepNumber + 1);

    const current = history[history.length - 1];

    const squares = current.squares.slice();

    if (calculateWinner(squares) || squares[i]) {

      return;

    }

    squares[i] = this.state.xIsNext ? "X" : "O";

    this.setState({

      history: history.concat([

        {

          squares: squares

        }

      ]),

      stepNumber: history.length,

      xIsNext: !this.state.xIsNext

    });

  }

  jumpTo(step) {

    this.setState({

      stepNumber: step,

      xIsNext: (step % 2) === 0

    });

  }

  render() {

    const history = this.state.history;

    const current = history[this.state.stepNumber];

    const winner = calculateWinner(current.squares);

    const moves = history.map((step, move) => {

      const desc = move ?

        'Go to move #' + move :

        'Go to game start';

      return (

        <li key={move}>

          <button onClick={() => this.jumpTo(move)}>{desc}</button>

        </li>

      );

    });

    let status;

    if (winner) {

      status = "Winner: " + winner;

    } else {

      status = "Next player: " + (this.state.xIsNext ? "X" : "O");

    }

    return (

      <div className="game">

        <div className="game-board">

          <Board

            squares={current.squares}

            onClick={i => this.handleClick(i)}

          />

        </div>

        <div className="game-info">

          <div>{status}</div>

          <ol>{moves}</ol>

        </div>

      </div>

    );

  }

}

// ========================================

ReactDOM.render(<Game />, document.getElementById("root"));

function calculateWinner(squares) {

  const lines = [

    [0, 1, 2],

    [3, 4, 5],

    [6, 7, 8],

    [0, 3, 6],

    [1, 4, 7],

    [2, 5, 8],

    [0, 4, 8],

    [2, 4, 6]

  ];

  for (let i = 0; i < lines.length; i++) {

    const [a, b, c] = lines[i];

    if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {

      return squares[a];

    }

  }

  return null;

}

**Resultado final**

