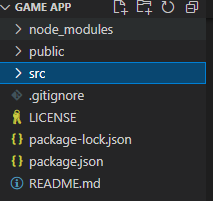
# Dokumentacija

React je JavaScript biblioteka koja olakšava izradu korisničkog interfejsa. Osnovni princip na kome se zasniva su komponente. Cilj je da se složena web stranica prikaže preko većeg broja nezavisnih komponenti što omogućava lakšu preglednost koda, nadogradnju i upravljanje kodom.

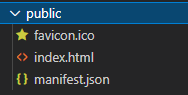
## Struktura aplikacije

U svakoj react aplikaciji postoji najmanje jedna komponenta-root komponenta koja je uglavnom automatski generisana, kao i određeni folder i fajlovi prilikom kreiranja react aplikacije(naredba npx create-react-app game app). Nakon što se pozicioniramo u novokreirani folder game app, sa komandne linije pokrećemo aplikaciju naredbom npm start. Tada nam se automatski generišu naredni folder i fajlovi:



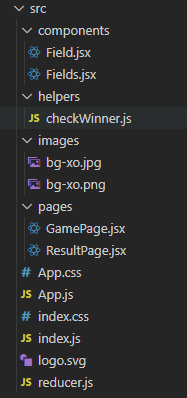
### Public folder

U okviru ovog foldera se nalazi index.html fajl koji je zadužen za prikaz react aplikacije na web stranici. On ima prazan div element sa id=“root” koji će se pozivati u glavnoj komponenti react aplikacije.

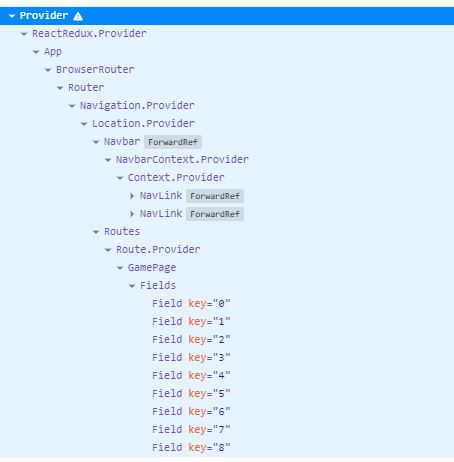


### Src folder

U ovom folderu se nalazi glavna komponenta App.js, ostale komponente u okviru foldera components:Field.jsx,Fileds.jsx foldera pages:GamePage.jsx, ResultPage.jsx. U folderu images se nalaze slike koje se koriste u aplikaciji, u folderu helpers se nalazi komponenta checkWinner koja proverava ko je pobednik igre. App.css i index.css su fajlovi koji su zaduženi za stilizovanje web stranica i njihovih elemenata. Kako je App.js glavna komponenta, stil koji ona ima važi za sve komponente koje je nasleđuju. Index.js je fajl u kome se vrši renderovanje aplikacije, tj. Povezivanje html fajla i react fajlova. Reducer.js je fajl koji inicijalizuje state-memoriju koju naša aplikacija koristi i upravlja promenama u memoriji.



### Hijerarhijska uređenost komponenti



Na slici je prikazana hijerarhija komponenti, iako je na samom vrhu Provider komponenta kojom se omogućuje korišćenje store-a tj memorijskog prostora, kako u okviru ove komponente tako i u okviru svih komponenti koje su njegova deca. Za naše razmatranje najbitnija će nam biti App.js komponenta u kojoj je cenatar react aplikacije.

Komponente mogu da se definišu kao:

1. Klasne-poslednjih godina se ređe koriste
2. Funkcionalne koje imaju ključnu reč function i return
3. Funkcionalne sa strelicom-koje predstavljaju drugačiji zapis funkcionalnih komponenti

Komponente se mogu pisati u js ili jsx format. Eksportuju se kako bi mogle da ih koriste druge komponente. Kada neka komponenta import-uje neku drugu komponentu to znači da se ona nalazi na višem nivou hijerarhije. Tako komponenta Fields,js importuje komponentu Filed.js, pa je na višem nivou u odnosu na nju. Slično GamePage.js importuje Fields.js, dok App.js importuje i GamePage.js i ResultPage.js, a samim tim komponente koje su one ranije importovale.

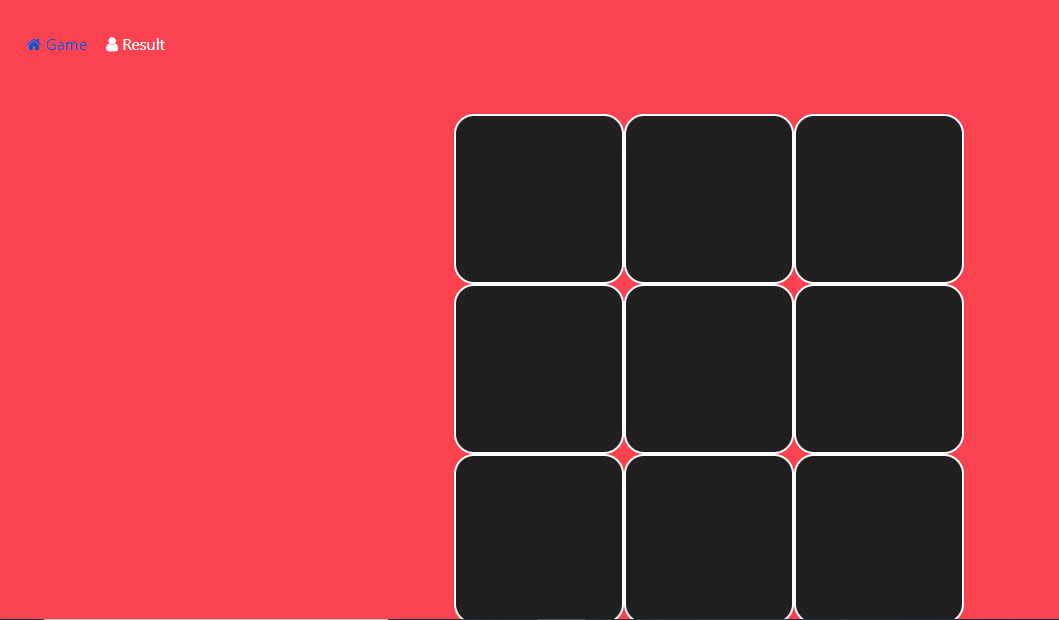
Promenljive, konstante(const,let), funckije unutar komponente definišemo iznad ključne reči return, dok u return delu vraćamo elemente koje želimo da prikažemo na web strani.

Za react strukturu komponenti je karakteristično da se sve informacije čuvaju na vrhu hijerarhije i prosledjuju ka nižim komponentama, a da izvršavanje funkcija započinje u obrnutom smeru, od dna ka vrhu hijerarhije.

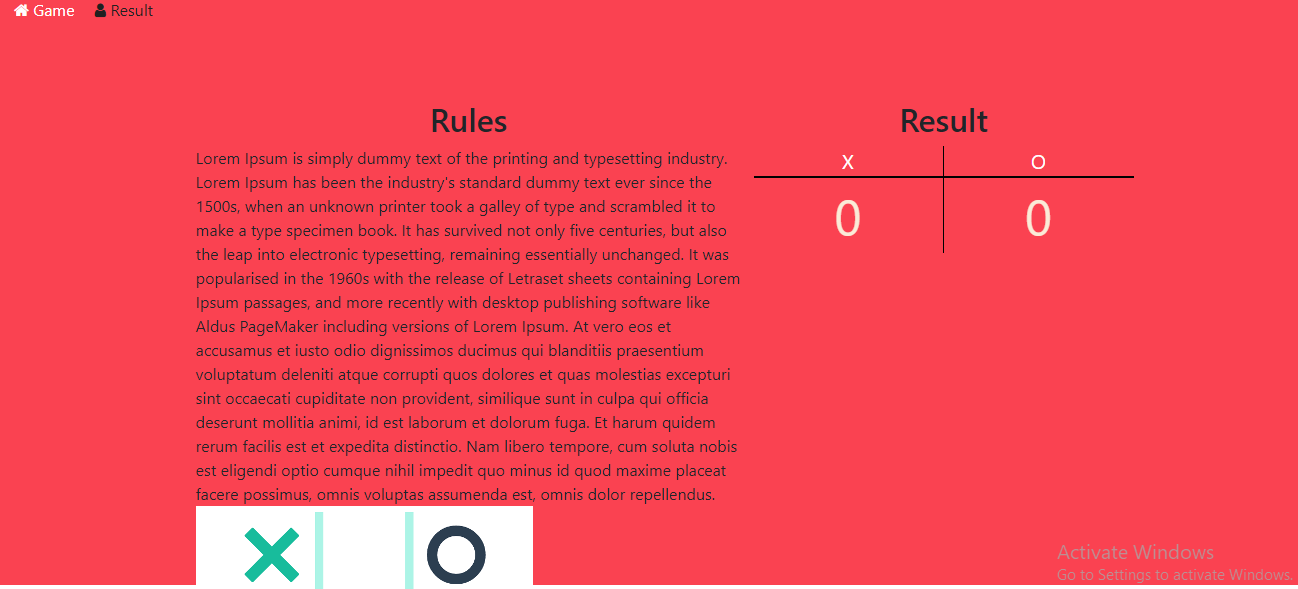
## Izgled aplikacije

Aplikacija se sastoji od dve stranice i simulira rad igrice iks oks (X O):

Stranica za igru:



Stranica sa rezultatima:



## Objašnjenje ključnih fajlova

### Index.js

U ovom fajlu se pokreće react aplikacija. Kreira se memorijski prostor koji služi za čuvanje podataka, na osnovu komponente rootReducer gde se nalaze početne vrednosti igre. Nakon toga se renderuje aplikacija koja pored učitanog store-a u html div element sa id=”root” učitava App.js komponentu.

import React from "react";

import ReactDOM from "react-dom";

import "./index.css";

import App from "./App";

import "bootstrap/dist/css/bootstrap.css";

import {createStore} from 'redux';

import "font-awesome/css/font-awesome.css";

import { Provider } from 'react-redux';

import rootReducer  from './reducer'

const store = createStore(rootReducer);

ReactDOM.render(

<Provider store={store}>

      <App />

    </Provider>

, document.getElementById("root")

);

### Reducer.js

Ova komponenta služi za inicijalizaciju memorijskog prostora i upravljanja istim. Kreiran je niz od 9 string polja koja su prvobitno prazna i koja predstavljaju polja za unos X i O vrednosti. Objekat initialState sadrži property-je:fields,player,numberOfWinX, numberOfWinY i predstavlja početno stanje naše memorije state koju ćemo kasnije menjati. U okviru ove komponente state postavljamo na početno stanje, akciju koju dobijamo od dispečera u App.js komponenti. Akcija može biti ažuriranje polja ili reset stranice. Ukoliko ažuriramo polja-to se vrši nakon svakog poteza gde se učitava prthodno stanje, unosi se na odgovarajuće polje vrednost X ili O, postavlja se naredni igrač, proverava se da li je dobijena dobitna kombinacija preko komponente checkWinner.js, ukoliko jeste, igraču se dodeljuje 1 poen koji se čuva u state-u i polja se resetuju. Učitavamo novi izgled forme. Ukoliko resetujemo polja, samo se njihova vrednost postavlja na prazan string.

import checkWinner from './helpers/checkWinner'

//pocetna vrednost polja

const initialFields = [

  {value:''},{value:''},{value:''},

  {value:''},{value:''},{value:''},

  {value:''},{value:''},{value:''}

 ];

 //pocatna vrednost objekta state

const initialState = {

    fields: initialFields

      player : 'X',

      numberOfWinsX:0,

      numberOfWinsY:0

    };

function rootReducer(state = initialState, action) {

    switch (action.type) {

      case 'UPDATE':

        const newState = {...state};

        newState.fields[action.payload.index].value = action.payload.player;

        newState.player = action.payload.player === 'X' ? 'O':'X';

        if(checkWinner(newState.fields) !== null)

        {

          console.log(newState.fields)

          if(action.payload.player === 'X')

            newState.numberOfWinsX++;

          else newState.numberOfWinsY++;

          newState.fields.forEach(field => {

              field.value = ''

          })

        }

        return {...state, ...newState  };

      case 'RESET':

        state.fields.forEach(field => {

          field.value = ''

        })

        return state;

      default:

        return state

    }

  }

  export default rootReducer;

### CheckWinner.js

Komponenta u kojoj se nalaze polja koja predstavljaju dobitne kombinacije u igri. Kao parameter prima polja igrača. Ukoliko igrač pogodi tri odgovarajuća polja dobija vraća se X ili O vrednost što je signal da je igrač dobio rundu i da mu se upiše poen. Upis poena se vrši u komponenti rootReducer koja importuje ovu komponentu. Ukoliko igrač nije uspeo da napravi neku od dobitnih kombinacija, vraća se null.

export default function check(fields) {

    //polja koja predstavljaju pobedu za nekog igraca

    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 (fields[a].value && fields[a].value === fields[b].value && fields[a].value === fields[c].value) {

            return fields[a].value;

        }

    }

    return null;

}

### Field.jsx

Ovo je komponenta koja služi za prikaz vrednosti unutar svakog polja igre. Na osnovu prosleđenog indeksa komponente Fields.jsx koja je poziva, ona učitava trenutnog igrača iz state-a. Nakon toga prolazi kroz polja i za polje sa odgovarajućim indeksom učitava njegovu vrednost (X ili O). Dugme koje pripada klasi “field” je crno, međutim ukoliko se preko njega pređe mišem on dobija klasu field-active i postaje ljubičasto. Funkcijom onPlay() se proverava da li je polje prazno, ukoliko jeste poziva se dispičer koji šalje objekat sa informacijama o igraču i indeksu polja. Ova komponenta upravlja akcijama koje se dešavaju na poljima igre, pa u return delu ukoliko se pređe preko polja obezbeđuje se promena boje, dok ukoliko se ukolkne na polje poziva se metoda onPlay() i postavlja se vrednost X ili O u odgovarajuće polje.

import React from "react";

import { useDispatch, useSelector } from 'react-redux';

import PropTypes from 'prop-types';

import { useState } from "react";

function  Field ({index}) {

    const dispatch = useDispatch();

    const [player, setPlayer] = useSelector((state) => state.player);

    const field = useSelector(state => state.fields.find((f,i) => i === index ))

    const [buttonClass, setButtonClass ]= useState("field");

    const onPlay = () => {

        if(field.value === '')

            dispatch({type: 'UPDATE', payload: { player: player, index : index } })

    }

    return(

        <button onMouseLeave={() => setButtonClass("field")} onMouseEnter={() => setButtonClass("field field-active")}

        className={buttonClass} onClick= {() => onPlay()}>

            <span className="inner-field"> {field.value} </span>

        </button>

    )

}

export default Field;

Field.propTypes = {

    index : PropTypes.number

}

### Fields.jsx

Ova komponenta je zadužena za prikaz polja preko mapiranja. Iz state-a učitava sva polja. Nakon toga prolazi kroz svako polje i preko komponente Field kojoj prosleđuje indeks pristupa odgovarajućem polju.

import React from 'react';

import Field from '../components/Field';

import { useSelector } from 'react-redux';

function Fields() {

    const fields = useSelector(state => state.fields

    return  (

        <>

            {

                fields.map(function(field, i){

                    return(

                    <Field key={i} index={i}/>

                    )

                })

            }

        </>

    );

}

export default Fields;

### GamePage.jsx

Ova komponenta je zadužena za prikaz svih polja koja se učitavaju komponentom Fields na web stranici Game. Centrira celinu col-md-12 koja predstavlja prostor za reset dugme . Ukoliko se ovo dugme klikne, poziva se metoda resetTable() koju implementira App.js komponenta koja importuje GamePage.jsx.

import React from 'react';

import Fields from '../components/Fields';

import PropTypes from 'prop-types';

function GamePage({resetTable}) {

      return  (

       <>

          <div className="fields">

            <Fields />

            <div className='col-md-12' style={{textAlign:'center'}}>

              <button className="reset" onClick={() => resetTable()}>

                <i className="fa fa-refresh"></i>

              </button>

            </div>

          </div>

      </>

     );

}

export default GamePage;

GamePage.propTypes = {

  resetTable : PropTypes.func

}

### ResultPage.jsx

Komponenta koja je zadužena za prikaz rezultata na web stranici Results. Iz state-a učitava broj poena koje ima igrač X i broj poena koje ima igrač Y. Prikazuje pravila igre, sliku igre, tabelu rezultata,

    import React from 'react';

    import { useSelector } from 'react-redux';

    import bgimage from '../images/bg-xo.png'

    function ResultPage()

    {

        const numberOfWinsX = useSelector((state) => state.numberOfWinsX);

        const numberOfWinsY = useSelector((state) => state.numberOfWinsY);

        return (

        <>

            <section>

                <div className='container content'>

                    <div className='row'>

                        <div className='col-md-6 col-offset-md-6' >

                            <div className='col-md-12' style={{textAlign:'center'}}>

                                <h2>Rules</h2>

                            </div>

                        Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

                        At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus.

                            <img src={bgimage} alt ="XO game"/>

                        </div>

                        <div className='col-md-4 col-md-offset-2' style={{textAlign:'center'}}>

                        <h2>Result</h2>

                            <div className='row' >

                                <div className='col-md-6 x-result-header'>

                                X

                                </div>

                                <div className='col-md-6 y-result-header'>

                                O

                                </div>

                                <div className='col-md-6 x-result'>

                                    <span>{numberOfWinsX}</span>

                                </div>

                                <div className='col-md-6 y-result'>

                                    <span>{numberOfWinsY}</span>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </section>

        </>

        );

    }

    export default ResultPage;

### App.js

Komponenta u kojoj se nalaze sve ostale komponente. Učitava dispičera koji menja stanje state-a, učitava polja igrice, kreira metodu resetTable() koja dispičeru šalje da je tip akcije ‘RESET’, nakon čega se poziva komponenta rootReducer koja upravlja state-om i postavlja vrednosti polja na prazan string. Ova komponenta u html div-u sa idjem root učitava komponente NavBar koji se sastoji od linkova ka rutama <http://localhost:3000/> i <http://localhost:3000/result> i njihovih ikonica, nakon toga dve rute <http://localhost:3000/> i <http://localhost:3000/result> koje kada se pozovu u okviru NavLinka ili ukucaju u browser-u omogućuju učitavanje komponenti GamePage.jsx i ResultPage.jsx. Ova komponenta se importuje u Provider komponenti.

import React, {useState} from 'react';

import { Route, Link, Routes, BrowserRouter, NavLink } from 'react-router-dom';

import GamePage from './pages/GamePage';

import ResultPage from './pages/ResultPage';

import { useDispatch, useSelector } from 'react-redux';

import { Navbar} from 'react-bootstrap';

import './App.css';

function App() {

  const dispatch = useDispatch(); //

  const fields = useSelector(state => state.fields);

  const resetTable = () =>

    {

      dispatch({type: 'RESET'})

      fields.forEach(field => {

        field.value = ''

      })

    }

  return (

    <BrowserRouter >

        <div className="app">

          <Navbar >

          <div className='cont'>

            <NavLink exact to="/" className="link"  activeClassName='is-active'>

                <i className="fa fa-home"></i> <span>Game</span>

            </NavLink>

            <NavLink to="/result" className="link" activeClassName='is-active' >

                <i className="fa fa-user"></i> <span>Result</span>

            </NavLink>

          </div>

          </Navbar>

          <Routes>

              <Route exact path="/" element={<GamePage resetTable = {resetTable} />} />

              <Route exact path="/result" element={<ResultPage />} />

          </Routes>

        </div>

    </ BrowserRouter >

  );

}

export default App;

Link ka privatnom git repozitorijumu: https://github.com/danicazdravkovic/ITEH\_D3