**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт**

(Национальный Исследовательский Университет)»

Институт №3 «Системы управления, информатика и электроэнергетика».

Кафедра 311

Направление подготовки Прикладная информатика (09.03.03)

Группа М30-217-Бк-21

**Отчет к лабораторной работе №6**

По дисциплине: Проектирование информационных систем

Тема: «Совместная разработка ИС. Элементы документирования разработки»

Работу выполнил студент 2-го курса:

Корсаков Егор Александрович

Москва, 2022 г

Комментирование используется в идеале всегда, это сокращает время на подготовку к поддержке ИС новым разработчиком .Все основные функции кода ниже закомментированы.

Все компоненты представленные ниже отделяются строчным отступом.

import React, { useEffect, useState } from 'react';

import { store } from '../redux/store';

import { data, options } from '../data/data';

import { Circle } from './Circle';

import { FilteredGraph } from './FilteredGraph'

import { redGraph } from '../data/data';

const cleanObj = (obj: Object): Object => { // Создание нового массива, отвязка адреса от исходного

  return JSON.parse(JSON.stringify(obj))

}

const calculateAverage = (arr: any): number[] => { // Вычисляет среднее значениние, для построения точек крастой прямой, возвращает массив 10 элементов

  const av: number = parseInt(arr.reduce((sum: number, elem: string) =>  sum + +elem \* .1 , 0) || 0)

  return Array<number>(10).fill(av)

}

export function BottomPanel() {

  const [ localPoints, setLocalPoints ] = useState<number[]>(store.getState().global.points) // Точки, которые будут показаны

  const [ labels, setLabels ] = useState<string[]>(store.getState().global.labels) // разметка оси абсцисс 'x1', ...

  const [ newData, setNewData ] = useState<any>(cleanObj(data)) // данные, которые будут соответствовать новым точкам по индексам

  const [ newOpions, setNewOptions ] = useState<any>(cleanObj(options)) // массив для оформления данных

  const unsubscribe = store.subscribe(() => {

    let points = store.getState().global.points,

        labs: string[] = []

    const range: any = {} = store.getState().global.range

    const multiple = store.getState().global.multiple

    if (store.getState().global.analys === 'ran' ){ // если в select выбран Range

      points = points.filter( (point, i) => {

        if(+point >= +range.min && +point <= +range.max) {

          labs.push(store.getState().global.labels[i]) // Все значения x, попадающие в диапазон

          return point

        }

      }) // фильтрую точки, входящие в интервал range

    } else {

      if( multiple ){ // Чтобы не прилетало Nan когда ввод пустой

        points = points.filter( (point, i) => {

          if( (i + 1)  % multiple === 0) { // Проверка кратности

            labs.push(store.getState().global.labels[i]) // Все значения x, попадающие в диапазон

            return point

          }

        })

      } else { // если пустой инпут, заполняет исходными точками

        points = store.getState().global.points

        labs = store.getState().global.labels

      }

    }

      setLabels(labs)

      setLocalPoints(points) // После того как стейт обновился, устанавливаю его в локальный

      unsubscribe() // Когда компонент обновляется (а происходит это доаольно часто) вешаются всё новые и новые прослушивальщики store, удаляю их после отработки

  })

  useEffect(() => {

    let datasets = JSON.parse(JSON.stringify(newData.datasets))

        datasets[0].data = labels.map((item, i) => localPoints[i] ) // напрямую (newData.datasets... ) переопределить нельзя, поэтому datasets вытаскивается, обновляется и вноситься обратно

        // после того как в глобальном стейте сгенерировался массив точек, интрефейс обновляется

        datasets[1] = redGraph

        datasets[1].data = calculateAverage(localPoints)

    setNewData({

            ...newData,

            datasets,

            labels

          })

  }, [localPoints])

  return <>

    {

      !newData.datasets[0].data.length ? null  : ( // Отрисовка шкалы

        <div className='filtered-area'>

          <FilteredGraph options={newOpions} data={newData} />

          <Circle options={newOpions} data={newData} />

        </div>

      )

    }

  </>

}

import React from 'react'

import {  PolarArea } from 'react-chartjs-2';

import { backgroundColor, chartOptions } from '../data/data';

type Props = {

  options: Object,

  data: any

}

export function Circle({options, data}: Props) { // Компонент отрисовки Круговой диаграммы, вынесен в одтельный файл, задал цвета

  let dt = JSON.parse(JSON.stringify(data))

  dt.datasets[0].backgroundColor = backgroundColor

  return (

    <PolarArea className='circle-graph' options={chartOptions} data={dt} />

  )

}

import { Line } from 'react-chartjs-2';

export function FilteredGraph( {options, data}: any) { // Отрисовка Нового Графика

  return <>

    {

      data.datasets[0].data.length ? <Line className='main-graph' options={options} data={data} /> : null // Отрисовка шкалы

    }

  </>

}

import { useState, FC } from "react"

import { store } from "../redux/store"

import 'materialize-css'

import {  setMul } from "../redux/reducer"

export const MultipleInp: FC = () => { // Компонент ввода значения для сортировки по нему

  const [ value, setValue ] = useState<string>('') // строка, которая хранит это значение

  const update = (value: string) => { // маска ввода

    if ( /^[1-9]\*$/.test(value) ) {

      setValue(value)

      console.log(value)

      store.dispatch(setMul(parseInt(value)))

    }

  }

  return ( // Отрисовка самого элемента

    <div className="row">

      <div className="input-field col s12">

        <input

          id="text"

          type="text"

          className="validate"

          value={value}

          onChange={(e) => {

            update(e.target.value)

          }}

        />

        <label htmlFor="text">Value</label>

      </div>

    </div>

  )

}

import React, { useEffect, useRef } from 'react'

import noUiSlider from 'nouislider';

import 'nouislider/dist/nouislider.css';

import 'materialize-css/dist/css/materialize.min.css';

import wNumb from 'wnumb';

import { useSelector } from 'react-redux';

import { setRange } from '../redux/reducer';

import { store } from '../redux/store';

interface DivWithSlider extends HTMLDivElement {

  noUiSlider: any

}

interface stateInterface {

  global: {

    points: number[],

    range: {

      min: number,

      max: number

    }

  }

} // 2 object's interfaces for creating by the patterns them

export const Range = () => { // Range-element for updating edges of filtering

  const range = useSelector( ( state: stateInterface ) => state.global.range) // getting authentificative numbers from global state

  const sliderRef = useRef<DivWithSlider>(null) // creating HTML elements in React in the right way

  const changeVal = (sides: any): void => { // updating range bar into global state

    store.dispatch(setRange({min: sides[0], max: sides[1]}))

  }

  useEffect(() => { // re-render Range-element after moving

    const slider = sliderRef.current // extracting slider from ref obj

    if(slider) { // if it there (for ts)

      noUiSlider.create( slider, { // pattern of creating slider in the standart way (lib doc)

        start: [range.min, range.max],

        connect: true,

        step: 1,

        orientation: 'horizontal', // 'horizontal' or 'vertical'

        range: {

          'min': -1000,

          'max': 1000

        },

        tooltips: true,

        format: wNumb({

          decimals: 0

        })

      });

      slider.noUiSlider.on('end', changeVal) // stopped move handler

    }

    return () => { // deletion this after unmount

      if(slider) {

        slider.noUiSlider.destroy();

      }

    };

  }, [range])

  return ( // drawing

    <div ref={sliderRef} className='slider' id='slider' ></div>

  )

}

import React, { useState, useEffect, FC } from "react"

import { Range } from "./Range"

import { store } from "../redux/store"

import 'materialize-css'

import { Select } from "./Select"

import { MultipleInp } from "./Input"

export const Regulate: FC = () => { // Выбор типа фильтрации ( Range, Multiple )

  const [ choice, setChoise ] = useState<string>('ran')

  useEffect(() => { // After uploading app start from global state's value

    setChoise(store.getState().global.analys)

  }, [])

  store.subscribe(() => { // change it after updating into select element throw state

    setChoise(store.getState().global.analys)

  })

  return (

    <div className='regulate-panel'>

      <Select />

      {

        choice === 'ran' ? <Range /> : <MultipleInp />

      }

    </div>

  )

}

import { useEffect, FC } from "react"

import { store } from "../redux/store"

import 'materialize-css'

import { setAnal } from "../redux/reducer"

export const Select: FC = () => {// Selectr element render

  useEffect(() => {

    const elems = document.querySelectorAll('select');

    M.FormSelect.init(elems);

  }, [])

  return (

    <div  className="input-field col s12" >

      <select onChange={ // change value into global state after selecting element

        (e) => {

          store.dispatch(setAnal(e.target.value))

        }

      }>

        <option value="ran">Range</option>

        <option value="mul">Multiple</option>

      </select>

      <label> Select</label>

    </div>

  )

}

// Creation start data of graphs

import { store } from '../redux/store';

import {

  Chart as ChartJS,

  CategoryScale,

  LinearScale,

  PointElement,

  LineElement,

  Title,

  Tooltip,

  Legend,

  PolarAreaController,

  RadialLinearScale,

  ArcElement

} from 'chart.js';

ChartJS.register(

  CategoryScale,

  LinearScale,

  PointElement,

  LineElement,

  Title,

  Tooltip,

  Legend,

  PolarAreaController,

  RadialLinearScale,

  ArcElement

); // Подключение всех функций

export const options = { // Тип графика

  responsive: true, //Автоизменнение шкалы, в зависимости от амплитуды элементов

  interaction: { // Возможно ли пересечение графиков

    mode: 'index' as const,

    intersect: false,

  },

  stacked: false, // Закрашивание поля под графиком

  plugins: { // Основные параметры графика

    title: {

      display: true,

      text: 'Lab2',

    },

    legend: {

      labels:{

        font: {

          size: 27

        }

      }

    }

  },

  scales: { // Шкала

    y1: {

      type: 'linear' as const,

      display: true,

      position: 'left' as const,

      grid: {

        drawOnChartArea: false,

      },

      ticks: {

        font: {

          size: 22

        }

      }

    },

  },

};

export const labels = store.getState().global.labels;

export const data = { //start data for all graphs

  labels,

  datasets: [

    {

      label: 'Dataset',

      data: [0],

      borderColor: 'rgb(53, 162, 235)',

      backgroundColor: ['rgba(53, 162, 235, 0.5)'],

      yAxisID: 'y1',

    },

  ],

};

export const redGraph = {// data for red graph line

  label: 'Dataset',

  data: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

  pointRadius: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

  borderColor: 'rgb(220, 53, 69)',

  backgroundColor: ['rgb(220, 53, 69)'],

  yAxisID: 'y',

}

export const chartOptions = {

  plugins: {

    title: {

      display: true,

      text: "Diagramm"

    },

    legend: {

      labels: {

        font: {

          size: 12

        }

      }

    }

  },

  scales: {

    r: {

      ticks: {

        font: {

          size: 12,

        },

      }

    },

  },

};

export const backgroundColor = [

  'rgb(255, 99, 132)',

  'rgb(75, 192, 192)',

  'rgb(255, 205, 86)',

  'rgb(201, 203, 207)',

  'rgb(54, 162, 235)',

  'rgb(255, 9, 132)',

  'rgb(35, 12, 192)',

  'rgb(25, 205, 86)',

  'rgb(201, 23, 7)',

  'rgb(54, 12, 235)',

]

import { createSlice } from "@reduxjs/toolkit";

import { faker } from '@faker-js/faker';

import type { PayloadAction } from "@reduxjs/toolkit";

export interface initialInterface {

  points: number[],

  analys: string,

  multiple: number,

  labels: string[],

  range: Object

}

const initialState :initialInterface = {

  points: [],

  analys: 'ran', //Какой тип анализа использутеся ran - range, mul - multiple

  multiple: 0,

  labels: ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10'],

  range: { min: -1000, max: 1000 }

}

function generatePoint():number { // start random genearation point

  const value = faker.datatype.number({ min: -1000, max: 1000 })

  return value

}

function generateStartPoints():Array<number> {

  let points: number[] = []

  for(let i = 0; i < 10; i++){

    points.push(generatePoint())

  }

  return points

}

const dataSlice = createSlice( { // Основной редюсер стора, в нём хранятся все точки графика

  name: 'data',

  initialState, // Начальная инициализация значения стора

  reducers: {

    setPoints: { // Редюсер Заполнения

      reducer: (state, acttion: PayloadAction<number[]>) => { // Заполняет массив точек

        state.points = acttion.payload

      },

      prepare: () => { // Срабатывает сразу же как приходят данные, после этого вызывается reducer сверху

        let points = generateStartPoints()

        return { payload: points }

      }

    },

    setRange: {

      reducer: (state, acttion: PayloadAction<Object>) => { // Заполняет range

        state.range = acttion.payload

      },

      prepare: (data: object) => {

        return { payload: data }

      }

    },

    setMul: { // setting multiple element

      reducer: (state, action: PayloadAction<number>) => {

        state.multiple = action.payload

      },

      prepare: (mul: number) => {

        return { payload: mul}

      }

    },

    setAnal: { // setting type of analys

      reducer: (state, action: PayloadAction<string>) => {

        state.analys = action.payload

      },

      prepare: (type: string) => {

        return { payload: type}

      }

    },

  }

} )

export const { setPoints, setRange, setAnal, setMul } = dataSlice.actions

export default dataSlice.reducer

import { configureStore, getDefaultMiddleware, MiddlewareAPI } from '@reduxjs/toolkit'

import data from './reducer'

import type { Middleware } from '@reduxjs/toolkit'

//global state difination as store

const visibleMiddware: Middleware = (api: MiddlewareAPI) => (next) => (action) => { // Пустой middleware

  return next( action )

}

const logger: Middleware = (api: MiddlewareAPI) => next => action => { //info about state action call

  // console.group(action.type)

  // console.info('dispatching', action)

  // console.log('next state', store.getState())

  return next(action)

}

export const store = configureStore({ // Инициализация стора

  reducer: {

    'global': data

  },

  middleware: getDefaultMiddleware =>

  [logger, visibleMiddware, ...getDefaultMiddleware()]

})