8.Pizza

1.Запускаем наш json-server.

2.Устанавливаем redux-thunk:

Npm install redux-thunk

3.Создаем санку fetchPizzas в pizzaActionCreators:

export const fetchPizzas=(dispatch)=>{

    axios.get/\*fetch\*/("http://localhost:3001/pizzas")

    /\*.then((response)=>response.json())\*/.then(response=>{

      dispatch(setPizzasActionCreator(response.data));

    console.log(response.data)

    })

}

4.В App заменяем запрос на санку и прокидываем в нее dispatch:

useEffect(() => {

fetchPizzas(dispatch)

console.log(fetchPizzas(dispatch))

}, []

)

5.Теперь переписываем на случай если у нас нет dispatch в App.В pizzaActionCreators:

export const fetchPizzas=()=>(dispatch)=>{

    axios.get/\*fetch\*/("http://localhost:3001/pizzas")

    /\*.then((response)=>response.json())\*/.then(response=>{

      dispatch(setPizzasActionCreator(response.data));

 console.log(response.data)

    })

}

6.В App:

useEffect(() => {

dispatch(fetchPizzas())

}, []

)

7.Наш store:

import { combineReducers, createStore,compose,applyMiddleware } from "redux";

import filters\_Reducer from "./reducers/filters\_Reducer.js"

import pizzas\_Reducer from "./reducers/pizzas\_Reducer.js"

import thunk from "redux-thunk"

let reducers=combineReducers({

filters:filters\_Reducer,

pizzas:pizzas\_Reducer

})

const composeEnchancers=window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose

 let store=createStore(reducers,

    composeEnchancers(applyMiddleware(thunk))

 )

 window.store=store

 export default store

8.Свои наброски в Home.jsx добавил кое-что:

{state.isLoaded?state.pizzas.map((elem,index)=>{

  return <PizzaBlock rating={elem.rating} price={elem.price} sizes={elem.sizes} category={elem.category} key ={index} name={elem.name}

  imageUrl={elem.imageUrl} id={elem.id}  types={elem.types} isLoaded={state.isLoaded}/>

}):Array(10).fill(0).map((\_,index)=>{

  return <Preloader key={index} />

})

}

9.Переде этим добавил isLoaded в useSelector:

const state =useSelector(({pizzas,filters})=>{

    return{

    pizzas:pizzas.pizzas,

    sortBy:filters.sortBy,

    category:filters.category,

    isLoaded:pizzas.isLoaded

    }

    }

    )

10.Также в pizzas\_Reducer добавил:

const pizzas\_Reducer=(state=initialState,action)=>{

      if(action.type==="SET\_PIZZAS"){

    return {

    ...state,pizzas:action.payload,

       isLoaded:true

    }

      }

        return state

    }

isLoaded=true

11.Не делал loader через skeleton вместо этого использовал прелоадер димыча с крутилкой:

import React from 'react'

import loader from "./loader.webp"

const Preloader=(props)=>{

return <div>

<img src={loader} alt={"Loader"}/>

</div>

}

export default Preloader;

12.Создаем АК в pizzaActionCreators:

export  const setLoadedPizzasActionCreator=(payload)=>({

type:"SET\_LOADED",payload:payload

})

13.Меняем санку в pizzaActionCreators

:

export const fetchPizzas=()=>(dispatch)=>{

dispatch(setLoadedPizzasActionCreator(false))

    axios.get/\*fetch\*/("http://localhost:3001/pizzas")

    /\*.then((response)=>response.json())\*/.then(response=>{

      dispatch(setPizzasActionCreator(response.data));

 console.log(response.data)

    })

}

14.Изменяем CategoryWithHooks:

import React from "react";

import { useState } from "react";

const CategoriesWithHooks=(props)=>{

/\*const[activeItem,setActiveItem]=useState(null)\*/

function onSelectItem(index){

  /\*setActiveItem(index) \*/

  props.onClickItem(index)

}

return  <div className="categories">

<ul>

<li  className={/\*activeItem\*/props.category===null?"active":""}

 onClick ={()=>onSelectItem(null)}>Все</li>

{props.items && props.items.map((elem,index)=>{

return <li  className={/\*activeItem\*/props.category===index?"active":""}

 onClick ={()=>onSelectItem(index)} key={`${elem}\_${index}`} > {elem}</li>

})}

</ul>

</div>

}

export default CategoriesWithHooks

15.В редусере ставим category =null

16.Добавляем propTypes в CategoriesWithHooks:

CategoriesWithHooks.propTypes={

items:PropTypes.arrayOf(PropTypes.string).isRequired,

onClickItem:PropTypes.func

}

CategoriesWithHooks.defaultProps={items:[]}

export default CategoriesWithHooks

17.Делаем sortBy через редакс:

<b>Сортировка по:</b>

{/\*5 +\*/} <span onClick={toggleVisiblePopup}>{props.varies[/\*activeItem\*/props.sortBy.id].name/\* был activeItem\*/ }</span>

  {window.visiblePopup=visiblePopup }

</div>

{visiblePopup && <div className="sort\_\_popup">

  <ul>

     {props.varies.map((elem,index)=>{

 /\*3+\*/    return  <li key={`${elem.type}\_${index}`} onClick={()=>chooseParam(index )}

 /\*4+   \*/    className={/\*activeItem\*/props.sortBy.id===index ? "active":""}>{elem.name}</li>

18.Пробуем делать сортировку через json server:

export const fetchPizzas=(category,sortBy)=>(dispatch)=>{

dispatch(setLoadedPizzasActionCreator(false))

    axios.get/\*fetch\*/(`http://localhost:3001/pizzas?category=${category}&sortBy=${sortBy}`)

    /\*.then((response)=>response.json())\*/.then(response=>{

      dispatch(setPizzasActionCreator(response.data));

 console.log(response.data)

    })

}

19.И добавляем изменения в useEffect в App:

useEffect(() => {

dispatch(fetchPizzas(category,sortBy))

},[category,sortBy]

)