~~Нужно разработать проект одностраничного сайта, на котором будет 3 текстовых поля и кнопка.~~

~~1 поле - цена~~

~~2 поле - количество~~

~~3 поле - сумма~~

Вносить значения можно в любое поле, причем то поле, что менялось раньше всех, должно пересчитаться согласно значения изменившегося поля.

~~При нажатии на кнопку, все 3 поля через ajax отсылаются на сервер и сохраняются во временном файле. При рефреше страницы сохраненные значения подставляются в поля.~~

Проект должен быть реализован с использованием в браузерной части: React + Next.js + Redux. Серверная часть: ~~Node.js~~ + ~~Express.js~~ + Next.js + ~~Yup.~~

Код браузерной части должен быть написан с использованием TypeScript, ~~а серверной - Javascript~~.

~~При обработке входящих данных ajax должен выдавать ошибку, если любое из полей не является десятичным числом. Ошибку необходимо отражать на странице~~.

Для проекта будет выделен git репозиторий, в который нужно загрузить проект. +

В репозитории должно быть 2 ветки, в одной из которых фон страницы будет серым, а поля белые,

а в другом - фон желтым, а поля - серые.

Для получения задания нужно предоставить rsa-ключ и IP-адрес, с которого будет выполняться подключение к git-репозиторию.

Код проекта должен быть документирован и иметь инструкцию по установке.

<https://medium.com/@ypushpraj/starting-up-with-a-react-redux-node-application-part-i-cc078d3a880>

yarn create react-app my-app --template typescript

yarn create next-app

yarn add typescript @types/node @types/react @types/react-dom @types/jest @types/axios @types/react-redux @types/redux

npm install --save redux-thunk

npm i @types/react-redux redux react-redux axios

rsc – Создаёт пустой App

<https://github.com/mluberry/nextjs-express>

<https://github.com/mars/heroku-nextjs-custom-server-express>

const SignupSchema = Yup.object().shape({

    price: Yup.number().test(

      'is-decimal',

      'invalid decimal',

      value => (value + "").match(/^\d\*\.{1}\d\*$/),

    ),

  });

const validationsSchema = yup.object**()**.shape**({**

name: yup.string**()**.typeError**(**'Должно быть строкой'**)**.required**(**'Обязательно'**)**,

secondName: yup.string**()**.typeError**(**'Должно быть строкой'**)**.required**(**'Обязательно'**)**,

password: yup.string**()**.typeError**(**'Должно быть строкой'**)**.required**(**'Обязательно'**)**,

confirmPassword: yup.string**()**.oneOf**([**yup.ref**(**'password'**)]**, 'Пароли не совпадают'**)**.required**(**'Обязательно'**)**,

email: yup.string**()**.email**(**'Введите верный email'**)**.required**(**'Обязательно'**)**,

confirmEmail: yup.string**()**.email**(**'Введите верный email'**)**.oneOf**([**yup.ref**(**'email'**)]**, 'Email не совпадают'**)**.required**(**'Обязательно'**)**



changeValue(value) {

        let new\_values = {...this.props.boxParamForm.values};

        new\_values[value.name] = value.result;

        this.props.changeBoxParamFormValue(new\_values);

    }

<AddBoxParamForm

                                                //isEdit = {this.props.modalBoxOpen}

                                                boxSelectChange = {this.props.boxSelectChange}

                                                boxEdit = {this.props.boxEdit}

                                                boxEditId = {this.props.boxEditId}

                                                materials={this.props.order.material ? this.props.order.material : []}

                                                close={this.props.closeModalBoxParam}

                                                addBox = {this.props.addBox}

                                                changeBoxParamFormValue = {this.props.changeBoxParamFormValue}                  //////////

                                                changeCountBoxParamForm = {this.props.changeCountBoxParamForm}                  //////////

                                                boxParam = {this.props.boxParam}                  //////////

                                                addNewBox = {this.props.addNewBox}                  //////////

                                                deleteBox={this.props.deleteBox}                    //////////

                                                boxParamForm = {this.props.boxParamForm}

                                                boxData = {this.props.boxData}

                                                boxSelect = {this.props.boxSelect}

                                                orderData={this.props.orderData}

                                                />

let mapDispatchToProps = (dispatch) => {

    return {

        changeBoxParamFormValue: (status) => dispatch(changeBoxParamFormValueAC(status)),                                  //////////

}

export const changeBoxParamFormValueAC = (values) => {

    return {

        type: CHANGE\_BOX\_PARAM\_FORM\_VALUE,

        values: values

    }

}

const BoxReducer = (state = initialState, action) => {

    switch(action.type) {

        case CHANGE\_BOX\_PARAM\_FORM\_VALUE:

            console.log(state.boxParamForm);

            debugger;

            return {

                ...state,

                boxParamForm: {

                    ...state.boxParamForm,

                    values: action.values

                }

            }

        default:

            return state

    }

}

node server.js

<http://localhost:5000/api/hello>

***BoxReduser.js***

const CHANGE\_COUNT\_BOX\_PARAM\_FORM = 'CHANGE\_COUNT\_BOX\_PARAM\_FORM';

let initialState = {

    queue: [],

    inputs: {

        price: '',

        count:  '',

        summ: '',

    },

  boxParamForm: {

        article\_id: null,

        boxParam: null,

        values: {},

        count: null,

        isEdit: null,

        editBooxParam: null

    },

};

const BoxReducer = (state = initialState, action) => {

    switch(action.type) {

        case CHANGE\_COUNT\_BOX\_PARAM\_FORM:

            console.log(state.boxParamForm);

            debugger;

            return {

                ...state,

                boxParamForm: {

                    ...state.boxParamForm,

                    count: action.values.result

                }

            }

        default:

            return state

    }

}

export const changeCountBoxParamFormAC = (values) => {

    return {

        type: CHANGE\_COUNT\_BOX\_PARAM\_FORM,

        values: values

    }

}

export default BoxReducer;

***Store.js***

import {applyMiddleware, combineReducers, createStore} from 'redux';

import BoxReducer from "./reducers/BoxReducer";

import thunk from 'redux-thunk';

let reducers = combineReducers({

    box: BoxReducer

})

let store = createStore(reducers, applyMiddleware(thunk));

window.store = store;

export default store;

http://api.ifurn.pro/api/orders/

client\_name":"FishmenClient"

14

***ServiceContainer.jsx***

import React from 'react';

import Service from './Service';

import {connect} from "react-redux";

import {

    changeCountBoxParamFormAC,

} from '../../store/reducers/BoxReducer';

class ServiceContainer extends React.Component {

    constructor(props) {

        super(props);

        this.state = {

            boxData: null,

        }

        this.boxSelectChange = this.boxSelectChange.bind(this);

    }

    componentDidMount() {

    }

    componentWillUnmount() {

        clearInterval(this.intervalCheck);

    }

render() {

        return  <Service

                        changeCountBoxParamForm={this.props.changeCountBoxParamForm}

                        />

    }

let mapStateToProps = (state) => {

    return {

        box: state.box,

        boxParamForm: state.boxParamForm,

    }

}

let mapDispatchToProps = (dispatch) => {

    return {

        changeCountBoxParamForm: (status) => dispatch(changeCountBoxParamFormAC(status)),                                  //////////

    }

}

export default connect(mapStateToProps, mapDispatchToProps)(ServiceContainer);

Очередь

1

2

3