Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ О ВЫПОЛНЕНИИ ДОМАШНЕЙ РАБОТЫ

Студент	Никифорова Ирина Андреевна
Группа	РК6-71б
Тип задания	домашняя работа
Тема лабораторной работы	фреймворк React.js

Студент _____ **Никифорова И. А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Берчун Ю. В.**
подпись, дата *фамилия, и.о.*

Ассистент _____ **Бурков П. В.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2019 г.

Оглавление

Задание на лабораторную работу **2**

Описание решения **2**

Листинг программы **3**

App.js 3

Button.js 5

Header.js 7

Stage.js 8

Task.js 12

index.js 13

Задание на лабораторную работу

Необходимо разработать контейнерные и презентационные компоненты React-приложения "Task Tracker". Приложение должно реализовывать следующие функциональные возможности:

- создание рабочего процесса таск-трекера;

- добавление новых стадий в рабочий процесс таск-трекера (стадии должны добавляться в конец списка стадий);

- добавление задач в рабочий процесс таск-трекера (задачи должны добавляться в конец списка задач самой первой стадии);

- перенос задачи с текущей стадии на следующую (первая задача текущей стадии переходит в конец списка следующей стадии; с последней стадии задача должна удаляться).

Описание решения

Для каждой из сущностей приложения был разработан свой компонент в соответствии со следующей таблицей:

Сущность	**Компонент React**
Приложение	App
Кнопка	Button
Заголовок	Header
Стадия	Stage
Задача	Task

Листинг программы

1. App.js

```
import React, { PureComponent, Fragment } from 'react';
import Stage from '../Stage/Stage.js';
import Button from '../Button/Button.js';
import Header from '../Header/Header.js';


class App extends PureComponent {
   constructor(props) {
     super(props);
     this.state = {
       stages : [],
       newAppendedTask : '',
       newIndexOfAppend : 0,
     };


     this.handleClickOnAddStageButton =
this.handleClickOnAddStageButton.bind(this);
     this.handleMoveTask = this.handleMoveTask.bind(this);
   }


   handleClickOnAddStageButton() {
     this.setState(state => {
       const stagesAdd = state.stages.concat('1');


       return {
         ...state,
         stages : stagesAdd,
       };
     });
```

```
  }

  handleMoveTask(from, taskInnerText) {
    console.log("from:", from);
    console.log("taskInnerText:", taskInnerText);

    if (from + 1 >= this.state.stages.length) {
      alert(":) Задача удалена (:");
      return;
    }

    this.setState(state => ({
      ...state,
      newAppendedTask : taskInnerText,
      newIndexOfAppend : from + 1,
    }));
  }

  render() {
    return (
      <Fragment>
        <div id = "application">
          <Header
            className = "workflow_header"
            placeholder = "Название потока"
          />
          <div className = "workflow">
            {this.state.stages.map((_, index) =>
              <Stage
                key = {index}
                id = {index}
                handleMoveTask = {this.handleMoveTask}
```

```
                newAppendedTask = {this.state.newAppendedTask}
                newIndexOfAppend = {this.state.newIndexOfAppend}
              />
            )}
            <Button
              className = "add_stage_button"
              innerText = "+ добавить стадию"
              handleClick = {this.handleClickOnAddStageButton}
            />
          </div>
        </div>
      </Fragment>
    );
  }
}

export default App;
```

2. Button.js

```
import React, { PureComponent, Fragment } from 'react';

class Button extends PureComponent {
  constructor(props) {
    super(props);
    this.state = {
      className: props.className,
      innerText : props.innerText,
      handleClick : props.handleClick,
    };
```

```
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    if (this.state.handleClick) {
      this.state.handleClick();
    }
  }

  componentDidUpdate(prevProps) {
    if (this.props.innerText !== prevProps.innerText) {
      this.setState({innerText : this.props.innerText});
    }
  }

  render() {
    return (
      <Fragment>
        <div className = {this.state.className} onClick =
{this.handleClick}>
          {this.state.innerText}
        </div>
      </Fragment>
    );
  }
}

export default Button;
```

3. Header.js

```
import React, { PureComponent, Fragment } from 'react';


class Header extends PureComponent {
   constructor(props) {
     super(props);
     this.state = {
       value: '',
       className: props.className,
       placeholder : props.placeholder,
       handleHeaderValueChange : props.handleHeaderValueChange,
     };
     this.handleChange = this.handleChange.bind(this);
   }


   handleChange(event) {
     this.setState({value: event.target.value});
     if (this.state.handleHeaderValueChange) {
       this.state.handleHeaderValueChange(event.target.value);
     }
   }


   render() {
     return (
       <Fragment>
       <input
         type = "text"
         className = {this.state.className}
         value = {this.state.value}
         onChange = {this.handleChange}
```

```
          placeholder = {this.state.placeholder}
          maxLength = "15"
        />
        </Fragment>
      );
    }
}

export default Header;
```

4. Stage.js

```
import React, { PureComponent, Fragment } from 'react';
import Task from '../Task/Task.js';
import Button  from '../Button/Button.js';
import Header from '../Header/Header.js';

class Stage extends PureComponent {
   constructor(props) {
     super(props);
     this.state = {
       id : props.id,
       handleMoveTask : props.handleMoveTask,
       newAppendedTask: props.newAppendedTask,
       newIndexOfAppend: props.newIndexOfAppend,
       headerValue : '',
       tasksInnerText : [],
     };

     this.handleHeaderValueChange =
this.handleHeaderValueChange.bind(this);
```

```
    this.handleClickOnAddTaskButton =
this.handleClickOnAddTaskButton.bind(this);

    this.handleClickOnMoveTaskButton =
this.handleClickOnMoveTaskButton.bind(this);

  }


  componentDidUpdate(prevProps) {

    // если добавляют по моему индексу и (индекс или задача) поменялись с
прошлого раза

    if (

      (this.props.newIndexOfAppend === this.props.id &&

      this.props.newIndexOfAppend !== prevProps.newIndexOfAppend) ||

      (this.props.newIndexOfAppend === this.props.id &&

      this.props.newAppendedTask !== prevProps.newAppendedTask)

    ) {

      this.setState(state => {

        const tasks_add =
state.tasksInnerText.concat(this.props.newAppendedTask);


        return {

          ...state,

          tasksInnerText : tasks_add,

        };

      });

    }

  }


  handleHeaderValueChange(newHeaderValue) {

    this.setState({headerValue: newHeaderValue});

  }


  handleClickOnAddTaskButton() {

    const nextTaskInnerText = prompt("Введите задачу:");
```

```
    if (!nextTaskInnerText) {
      return;
    }

    this.setState(state => {
      const tasks_add = state.tasksInnerText.concat(nextTaskInnerText);

      return {
        ...state,
        headerValue : state.headerValue,
        tasksInnerText : tasks_add,
      };
    });
  }

  handleClickOnMoveTaskButton() {
    if (this.state.tasksInnerText.length === 0) {
      return;
    }

    // запоминаем переносимую задачу
    const taskInnerText =
this.state.tasksInnerText[this.state.tasksInnerText.length - 1];
    const id = this.state.id;

    // удаляем задачу из текущей стадии
    this.setState(state => {
      return {
        ...state,
        headerValue : state.headerValue,
        tasksInnerText : this.state.tasksInnerText.filter((_, i) =>
          i !== (this.state.tasksInnerText.length - 1)
```

```
        ),
      };
    });

    // отправляем данные в компонент Приложение
    this.state.handleMoveTask(id, taskInnerText);
  }



  render() {
    let computedInnerText = "+ добавить задачу";
    if (this.state.headerValue !== '') {
      computedInnerText += " к '" + this.state.headerValue + "'";
    }
    return (
      <Fragment>
        <div className = "stage">
          <Header
            className = "tasks_header"
            placeholder = "Название стадии"
            handleHeaderValueChange = {this.handleHeaderValueChange}
          />
          <div className = "tasks">
            <Button
              className = "add_task_button"
              innerText = {computedInnerText}
              handleClick = {this.handleClickOnAddTaskButton}
            />

            {this.state.tasksInnerText.map((taskInnerText, index) =>
              <Task
                key = {index}
```

```
                innerText = {taskInnerText}
              />
            )}

            <Button
              className = "add_task_button"
              innerText = "→ переместить последнюю"
              handleClick = {this.handleClickOnMoveTaskButton}
            />
          </div>
        </div>
      </Fragment>
    );
  }
}

export default Stage;
```

5. Task.js

```
import React, { PureComponent, Fragment } from 'react';

class Task extends PureComponent {
  constructor(props) {
    super(props);
    this.state = {
      id : props.id,
      innerText: props.innerText,
      handleClick: props.handleClick,
    };
  }

  render() {
```

```
      return (
        <Fragment>
            <div className = "task">
              {this.state.innerText}
            </div>
        </Fragment>
      );
    }
}

export default Task;
```

6. index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App/App.js';
import './index.css';


ReactDOM.render(
    <App />,
    document.getElementById('root')
);
```