МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет информационных систем и технологий

Кафедра «Информационные системы»

Дисциплина «Интернет программирование»

**Отчёт по лабораторной работе №4**

**REST-контроллеры**

Выполнил студент

гр. ПИбд-21:

Сергеев Н.И.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:

Филиппов А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ульяновск

2022

**Задание**

1. Для каждого класса с бизнес-логикой из предыдущей работы разработать свой REST-контроллер.
2. Работу с базой данных необходимо осуществлять через DAO (JpaRepository).
3. Получение сущностей необходимо реализовать через Get метод контроллера. Идентификатор сущности следует передавать через PathVariable.
4. Создание сущностей необходимо реализовать через Post метод контроллера. Данные следует передавать через тело запроса (RequestBody) с помощью DTO.
5. Обновление сущностей необходимо реализовать через Put метод контроллера. Идентификатор сущности следует передавать через PathVariable, а содержимое сущности через тело запроса с помощью DTO.
6. Удаление сущностей необходимо реализовать через Delete метод контроллера. Идентификатор сущности следует передавать через PathVariable.
7. Реализовать клиентскую часть приложения с помощью Vue.js или React. Рекомендуется использовать приложение из предыдущего семестра.

**Сервер**

1. **Rest-контроллеры**

package ip.labwork.shop.controller;

import ip.labwork.shop.service.ComponentService;

import jakarta.validation.Valid;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/component")

public class ComponentController {

private final ComponentService componentService;

public ComponentController(ComponentService componentService) {

this.componentService = componentService;

}

@PostMapping

public ComponentDTO createComponent(@RequestBody @Valid ComponentDTO componentDTO) {

return componentService.create(componentDTO);

}

@PutMapping("/{id}")

public ComponentDTO updateComponent(@PathVariable Long id,@RequestBody @Valid ComponentDTO componentDTO) {

return componentService.updateComponent(id,componentDTO);

}

@DeleteMapping("/{id}")

public ComponentDTO removeComponent(@PathVariable Long id) {

return componentService.deleteComponent(id);

}

@DeleteMapping

public void removeAllComponent() {

componentService.deleteAllComponent();

}

@GetMapping("/{id}")

public ComponentDTO findComponent(@PathVariable Long id) {

return new ComponentDTO(componentService.findComponent(id));

}

@GetMapping

public List<ComponentDTO> findAllComponent() {

return componentService.findAllComponent();

}

}

package ip.labwork.shop.controller;

import ip.labwork.shop.service.ProductService;

import jakarta.validation.Valid;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/product")

public class ProductController {

private final ProductService productService;

public ProductController(ProductService productService) {

this.productService = productService;

}

@PostMapping

public ProductDTO createProduct(@RequestBody @Valid ProductDTO productDTO){

return productService.create(productDTO);

}

@PutMapping("/{id}")

public ProductDTO updateProduct(@PathVariable Long id,@RequestBody @Valid ProductDTO productDTO){

return productService.updateProduct(id, productDTO);

}

@DeleteMapping("/{id}")

public ProductDTO removeProduct(@PathVariable Long id){

return productService.deleteProduct(id);

}

@DeleteMapping

public void removeAllProduct(){

productService.deleteAllProduct();

}

@GetMapping("/{id}")

public ProductDTO findProduct(@PathVariable Long id){

return new ProductDTO(productService.findProduct(id));

}

@GetMapping

public List<ProductDTO> findAllProduct(){

return productService.findAllProduct();

}

}

package ip.labwork.shop.controller;

import ip.labwork.shop.service.OrderService;

import jakarta.validation.Valid;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/order")

public class OrderController {

private final OrderService orderService;

public OrderController(OrderService orderService) {

this.orderService = orderService;

}

@PostMapping

public OrderDTO createOrder(@RequestBody @Valid OrderDTO orderDTO){

return orderService.create(orderDTO);

}

@PutMapping("/{id}")

public OrderDTO updateOrder(@PathVariable Long id, @RequestBody @Valid OrderDTO orderDTO){

return orderService.update(id, orderDTO);

}

@DeleteMapping("/{id}")

public OrderDTO removeOrder(@PathVariable Long id){

return orderService.deleteOrder(id);

}

@DeleteMapping

public void removeAllOrder(){

orderService.deleteAllOrder();

}

@GetMapping("/{id}")

public OrderDTO findOrder(@PathVariable Long id){

return new OrderDTO(orderService.findOrder(id));

}

@GetMapping

public List<OrderDTO> findAllOrder(){

return orderService.findAllOrder();

}

}

1. **DAO**

package ip.labwork.shop.repository;

import ip.labwork.shop.model.Component;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ComponentRepository extends JpaRepository<Component, Long> {

}

package ip.labwork.shop.repository;

import ip.labwork.shop.model.Product;

import ip.labwork.shop.model.ProductComponents;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface ProductRepository extends JpaRepository<Product, Long> {

@Query("Select os from ProductComponents os where os.product.id = :productId")

List<ProductComponents> getProductComponent(@Param("productId") Long orderId);

}

package ip.labwork.shop.repository;

import ip.labwork.shop.model.Order;

import ip.labwork.shop.model.OrderProducts;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface OrderRepository extends JpaRepository<Order, Long> {

@Query("Select os from OrderProducts os where os.order.id = :orderId")

List<OrderProducts> getOrderProduct(@Param("orderId") Long orderId);

}

1. **DTO**

package ip.labwork.shop.controller;

import ip.labwork.shop.model.Component;

public class ComponentDTO {

private long id;

private String componentName;

private int price;

private int count = 0;

public ComponentDTO(Component component) {

this.id = component.getId();

this.componentName = component.getComponentName();

this.price = component.getPrice();

}

public ComponentDTO(Component component, int count) {

this.id = component.getId();

this.componentName = component.getComponentName();

this.price = component.getPrice();

this.count = count;

}

public ComponentDTO() {

}

public long getId() {

return id;

}

public String getComponentName() {

return componentName;

}

public int getCount() {

return count;

}

public int getPrice() {

return price;

}

}

package ip.labwork.shop.controller;

import ip.labwork.shop.model.Product;

import java.util.List;

import java.util.Objects;

public class ProductDTO {

private long id;

private String name;

private int price;

private List<ComponentDTO> componentDTOList;

private List<OrderDTO> orderDTOList;

private String image;

private int count;

public ProductDTO(Product product) {

this.id = product.getId();

this.name = product.getProductName();

this.price = product.getPrice();

this.image = product.getImage() == null? "" : new String(product.getImage());

this.componentDTOList = product.getComponents().stream()

.filter(x -> Objects.equals(x.getId().getProductId(), product.getId()))

.map(y -> new ComponentDTO(y.getComponent(), y.getCount()))

.toList();

this.orderDTOList = product.getOrders() == null ? null : product.getOrders().stream().filter(x -> Objects.equals(x.getId().getProductId(), product.getId())).map(x -> new OrderDTO(x.getOrder())).toList();

}

public ProductDTO(Product product, int count) {

this.id = product.getId();

this.name = product.getProductName();

this.price = product.getPrice();

this.image = product.getImage() == null? "" : new String(product.getImage());

this.componentDTOList = product.getComponents().stream()

.filter(x -> Objects.equals(x.getId().getProductId(), product.getId()))

.map(y -> new ComponentDTO(y.getComponent(), y.getCount()))

.toList();

this.count = count;

}

public ProductDTO() {

}

public long getId() {

return id;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public String getName() {

return name;

}

public int getPrice() {

return price;

}

public List<ComponentDTO> getComponentDTOList() {

return componentDTOList;

}

public String getImage() {

return image;

}

public void setImage(String image) {

this.image = image;

}

public List<OrderDTO> getOrderDTOList() {

return orderDTOList;

}

}

package ip.labwork.shop.controller;

import ip.labwork.shop.model.Order;

import ip.labwork.shop.model.OrderStatus;

import java.util.Date;

import java.util.List;

import java.util.Objects;

public class OrderDTO {

private long id;

private Date date = new Date();

private int price;

private OrderStatus status = OrderStatus.Неизвестен;

private List<ProductDTO> productDTOList;

public OrderDTO(Order order) {

this.id = order.getId();

this.date = order.getDate();

this.price = order.getPrice();

this.productDTOList = order.getProducts().stream()

.filter(x -> Objects.equals(x.getId().getOrderId(), order.getId()))

.map(y -> new ProductDTO(y.getProduct(), y.getCount()))

.toList();

this.status = Objects.equals(order.getStatus().toString(), "") ? OrderStatus.Неизвестен : order.getStatus();

}

public OrderDTO() {

}

public long getId() {

return id;

}

public Date getDate() {

return date;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public void setDate(Date date) {

this.date = date;

}

public OrderStatus getStatus() {

return status;

}

public void setStatus(OrderStatus status) {

this.status = status;

}

public List<ProductDTO> getProductDTOList() {

return productDTOList;

}

}

1. **Клиентское приложение**

App.jsx

import './App.css';

import { useRoutes, Outlet, BrowserRouter } from 'react-router-dom';

import Header from './components/common/Header';

import Footer from "./components/common/Footer";

import CatalogStudents from './components/catalogs/CatalogStudents';

import Menu from './components/catalogs/Menu';

import Basket from './components/catalogs/Basket';

import History from './components/catalogs/History';

import { useState } from 'react';

function Router(props) {

return useRoutes(props.rootRoute);

}

export default function App() {

const [product,setProduct] = useState([]);

const routes = [

{ index: true, element: <CatalogStudents /> },

{ path: "catalogs/menu", element: <Menu product={product} setProduct={setProduct}/>, label: "Меню" },

{ path: "catalogs/component", element: <CatalogStudents />, label: "Компоненты" },

{ path: "catalogs/basket", element: <Basket product={product} setProduct={setProduct}/>, label: "Корзина" },

{ path: "catalogs/history", element: <History />, label: "История" }

];

const links = routes.filter(route => route.hasOwnProperty('label'));

const rootRoute = [

{ path: '/', element: render(links), children: routes }

];

function render(links) {

return (

<>

<Header links={links} />

<div className="container-fluid p-0">

<Outlet />

</div>

<Footer></Footer>

</>

);

}

return (

<BrowserRouter>

<Router rootRoute={ rootRoute } />

</BrowserRouter>

);

}

DataService.js

import axios from 'axios';

export default class DataService {

static dataUrlPrefix = 'http://localhost:8080';

static async readAll(url, transformer) {

const response = await fetch(this.dataUrlPrefix + url);

const data = await response.json();

return data.map(item => transformer(item));

}

static async read(url, transformer) {

const response = await axios.get(this.dataUrlPrefix + url);

return transformer(response.data);

}

static async create(url, data) {

const requestParams = {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify(data),

};

const response = await fetch(this.dataUrlPrefix + url, requestParams);

}

static async update(url, data) {

const requestParams = {

method: "PUT",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify(data),

};

const response = await fetch(this.dataUrlPrefix + url, requestParams);

return true;

}

static async delete(url) {

const response = await axios.delete(this.dataUrlPrefix + url);

return response.data.id;

}

}

Component.js

export default class Component {

constructor(data) {

this.id = data?.id;

this.price = data?.price || "";

this.componentName = data?.componentName || "";

}

}

Product.js

export default class Product {

constructor(data) {

this.id = data?.id;

this.price = data?.price || "";

this.name = data?.name || "";

this.count = data?.count || 0;

this.image = data?.image || "";

this.componentDTOList = data?.componentDTOList || [];

}

}

Order.js

export default class Order {

constructor(data) {

this.id = data?.id;

this.date = data?.date || "";

this.price = data?.price || 0;

this.productDTOList = data?.productDTOList || [];

this.status = data?.status || "0";

}

}

Card.jsx

import DataService from "../../services/DataService";

export default function Card(props) {

function edit(id) {

props.onEdit(id);

}

function remove(id) {

props.onRemove(id);

}

async function mess(id){

let currentProduct = props.product.filter(x => x.id == id.id);

if (currentProduct.length != 0){

let temp = props.product.filter(x => x.id != id.id);

currentProduct[0].count++;

temp.push(currentProduct[0]);

await props.setProduct(temp);

return;

}else{

id.count++;

props.product.push(id);

props.setProduct(props.product);

}

}

return (

<div className="temp row row-cols-1 row-cols-md-3 g-4" id="tbl-items">

{props.items.map((item) => (

<div className="col" key={item.id}>

<div className="card">

<div

className="container"

style={{ width: "100%", height: "350px" }}

>

<img

src={item["image"]}

className="img-fluid rounded mx-auto d-block"

style={{ width: "100%", height: "350px", objectFit: "contain" }}

alt="Бугер"

/>

</div>

<div className="card-body">

<h5 className="card-title text-center fs-1">{item["price"]}</h5>

<a

href="#"

className="btn btn-outline-dark text-center d-flex justify-content-md-center mx-5"

onClick={(e) => remove(item.id, e)}

>

Удалить

</a>

<a

href="#"

type="button"

className="btn btn-outline-dark text-center d-flex justify-content-md-center mx-5"

data-bs-toggle="modal"

data-bs-target="#staticBackdrop"

onClick={(e) => edit(item.id, e)}

>

Изменить

</a>

<a

type="button"

className="btn btn-outline-dark text-center d-flex justify-content-md-center mx-5"

onClick={()=>mess(item)}

>

в корзину

</a>

</div>

</div>

</div>

))}

</div>

);

}

Footer.jsx

export default function Footer(props) {

return (

<footer className="footer mt-auto d-flex justify-content-center align-items-center">

ООО "Вкусно" © 2022

</footer>

);

}

Header.jsx

import { NavLink } from "react-router-dom";

export default function Header(props) {

return (

<nav className="navbar navbar-expand-lg">

<div className="container-fluid">

<NavLink className="navbar-brand" to={"/"}>

<h1>Очень вкусно и запятая</h1>

</NavLink>

<button

className="navbar-toggler"

type="button"

data-bs-toggle="collapse"

data-bs-target="#navbarSupportedContent"

aria-controls="navbarSupportedContent"

aria-expanded="false"

aria-label="Toggle navigation"

>

<span className="navbar-toggler-icon"></span>

</button>

<div className="collapse navbar-collapse" id="navbarSupportedContent">

<ul className="navbar-nav me-auto mb-2 mb-lg-0">

{props.links.map((route) => (

<li key={route.path} className="nav-item">

<NavLink className="nav-link fs-4" to={route.path}>

{route.label}

</NavLink>

</li>

))}

</ul>

</div>

</div>

</nav>

);

}

Modal.jsx

import React from "react";

export default function Modal(props) {

const formRef = React.createRef();

function hide() {

props.onHide();

}

function done(e) {

e.preventDefault();

if (formRef.current.checkValidity()) {

props.onDone();

hide();

} else {

formRef.current.reportValidity();

}

}

return (

<div className="modal fade show" tabIndex="-1" aria-hidden="true"

style={{ display: props.visible ? 'block' : 'none' }}>

<div className="modal-dialog">

<div className="modal-content">

<div className="modal-header">

<h1 className="modal-title fs-5" id="exampleModalLabel">{props.header}</h1>

<button className="btn-close" type="button" aria-label="Close"

onClick={hide}></button>

</div>

<div className="modal-body">

<form ref={formRef} onSubmit={done}>

{props.children}

</form>

</div>

<div className="modal-footer">

<button className="btn btn-secondary" type="button" onClick={hide}>Закрыть</button>

<button className="btn btn-primary" type="button" onClick={done}>

{props.confirm}

</button>

</div>

</div>

</div>

</div>

);

}

ModalProduct.jsx

import React from "react";

export default function ModalProduct(props) {

const formRef = React.createRef();

function hide() {

props.onHide();

}

function done(e) {

e.preventDefault();

if (formRef.current.checkValidity()) {

props.onDone();

hide();

} else {

formRef.current.reportValidity();

}

}

return (

<div className="modal fade show" tabIndex="-1" aria-hidden="true"

style={{ display: props.visible ? 'block' : 'none' }}>

<div className="modal-dialog">

<div className="modal-content">

<div className="modal-header">

<h1 className="modal-title fs-5" id="exampleModalLabel">{props.header}</h1>

<button className="btn-close" type="button" aria-label="Close"

onClick={hide}></button>

</div>

<div className="modal-body">

<form ref={formRef} onSubmit={done}>

{props.children}

</form>

</div>

<div className="modal-footer">

<button className="btn btn-secondary" type="button" onClick={hide}>Закрыть</button>

<button className="btn btn-primary" type="button" onClick={done}>

{props.confirm}

</button>

</div>

</div>

</div>

</div>

);

}

Table.jsx

import { useState } from 'react';

import styles from './Table.module.css';

export default function Table(props) {

const [tableUpdate, setTableUpdate] = useState(false);

const [selectedItems, setSelectedItems] = useState([]);

function isSelected(id) {

if (!props.selectable) {

return false;

}

return selectedItems.includes(id);

}

function click(id) {

if (!props.selectable) {

return;

}

if (isSelected(id)) {

var index = selectedItems.indexOf(id);

if (index !== -1) {

selectedItems.splice(index, 1);

setSelectedItems(selectedItems);

setTableUpdate(!tableUpdate);

}

} else {

selectedItems.push(id);

setSelectedItems(selectedItems);

setTableUpdate(!tableUpdate);

}

props.onClick(selectedItems);

}

function dblClick(id) {

if (!props.selectable) {

return;

}

props.onDblClick(id);

}

return (

<table className={`table table-hover ${styles.table} ${props.selectable ? styles.selectable : ''}`}>

<thead>

<tr>

<th scope="col">#</th>

{

props.headers.map(header =>

<th key={header.name} scope="col">

{header.label}

</th>

)

}

</tr>

</thead>

<tbody>

{

props.items.map((item, index) =>

<tr key={item.id}

className={isSelected(item.id) ? styles.selected : ''}

onClick={(e) => click(item.id, e)} onDoubleClick={(e) => dblClick(item.id, e)}>

<th scope="row">{index + 1}</th>

{

props.headers.map(header =>

<td key={item.id + header.name}>{item[header.name]}</td>

)

}

</tr>

)

}

</tbody >

</table >

);

}

TableOrder.jsx

import { useEffect } from "react";

import { useState } from "react";

import DataService from "../../services/DataService";

import Order from "../../models/Order";

export default function TableOrder(props) {

const [order, setOrder] = useState(new Order())

const [cost, setCost] = useState(0);

const [del, setDel] = useState(false);

useEffect(() => {

loadItems();

}, []);

useEffect(() => {

loadItems();

setDel(false);

}, [del]);

async function loadItems() {

await summ(props.product.sort((a, b) => a.name > b.name ? 1 : -1));

setOrder({...order,["productDTOList"]: props.product, ["price"]:cost});

}

async function summ(data) {

let tem = 0;

for (let i = 0; i < data.length; i++) {

tem += data[i].count \* Number(data[i].price);

}

setCost(tem);

setOrder({...order,["price"]: tem});

}

async function deleteItem(item) {

let currentProduct = props.product.filter((x) => x.id == item.id)[0];

let temp = props.product.filter((x) => x.id != item.id);

if (currentProduct.count - 1 == 0) {

props.setProduct(temp);

setDel(true);

loadItems();

return;

} else {

currentProduct.count--;

temp.push(currentProduct);

props.setProduct(temp);

setDel(true);

loadItems();

}

}

async function acceptOrder(){

await DataService.create("/order",{...order, ["price"]:cost, ["status"]: "1"} ).then(data => {

props.setProduct([]);

setCost(0);

});

}

return (

<div>

<div style={{ maxWidth: "35%" }}>

<table className="table">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Позиция</th>

<th scope="col"> </th>

<th scope="col">Стоимость</th>

<th scope="col"></th>

</tr>

</thead>

<tbody className="table-group-divider">

{props.product.map((item, index) => (

<tr key={item.id}>

<th scope="row">{index + 1}</th>

<td colSpan="2">{item.name}</td>

<td>

{item.count}x{item.price} руб

</td>

<td>

<button onClick={() => deleteItem(item)}>Удалить</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

<h2 className="ms-5 my-5">Итого: {cost} руб</h2>

<button

className="btn btn-success ms-5 w-25"

type="button"

style={{ color: "black" }}

onClick={acceptOrder}

>

Купить

</button>

</div>

);

}

Toolbar.jsx

import styles from './Toolbar.module.css';

export default function Toolbar(props) {

function add() {

props.onAdd();

}

function edit() {

props.onEdit();

}

function remove() {

props.onRemove();

}

return (

<div className="btn-group mt-2" role="group">

<button type="button" className={`btn btn-outline-dark text-center d-flex justify-content-md-center mx-2 mb-3`} onClick={add}>

Добавить

</button>

<button type="button" className={`btn btn-outline-dark text-center d-flex justify-content-md-center mx-2 mb-3`} onClick={edit} >

Изменить

</button >

<button type="button" className={`btn btn-outline-dark text-center d-flex justify-content-md-center mx-2 mb-3`} onClick={remove}>

Удалить

</button >

</div >

);

}

ToolbarProduct.jsx

export default function ToolbarProduct(props) {

function add() {

props.onAdd();

}

return (

<div className="btn-group mt-2" role="group">

<button

type="button"

className={`btn btn-outline-dark text-center d-flex justify-content-md-center mx-5 mb-3`}

onClick={add}

>

Добавить

</button>

</div>

);

}

Basket.jsx

import Table from "../common/TableOrder";

export default function Basket(props) {

return (

<main className="flex-shrink-0" style={{ backgroundColor: "white" }}>

<h1 className="my-5 ms-5 fs-1">

<b>Корзина</b>

</h1>

<h2 className="my-5 ms-5 fs-3"></h2>

<div className="ms-5 my-5">Список товаров</div>

<Table product={props.product} setProduct={props.setProduct}></Table>

<p> </p>

<p> </p>

</main>

);

}

Catalog.jsx

import { useState, useEffect } from "react";

import Toolbar from "../common/Toolbar";

import Table from "../common/Table";

import Modal from "../common/Modal";

import DataService from '../../services/DataService';

export default function Catalog(props) {

const [items, setItems] = useState([]);

const [modalHeader, setModalHeader] = useState('');

const [modalConfirm, setModalConfirm] = useState('');

const [modalVisible, setModalVisible] = useState(false);

const [isEdit, setEdit] = useState(false);

let selectedItems = [];

useEffect(() => {

loadItems();

}, []);

function loadItems() {

DataService.readAll(props.getAllUrl, props.transformer)

.then(data => setItems(data));

}

function saveItem() {

if (!isEdit) {

DataService.create(props.getAllUrl, props.data).then(() => loadItems());

} else {

DataService.update(props.url + props.data.id, props.data).then(() => loadItems());

}

}

function handleAdd() {

setEdit(false);

setModalHeader('Добавление элемента');

setModalConfirm('Добавить');

setModalVisible(true);

props.onAdd();

}

function handleEdit() {

if (selectedItems.length === 0) {

return;

}

edit(selectedItems[0]);

}

function edit(editedId) {

DataService.read(props.url + editedId, props.transformer)

.then(data => {

setEdit(true);

setModalHeader('Редактирование элемента');

setModalConfirm('Сохранить');

setModalVisible(true);

props.onEdit(data);

});

}

function handleRemove() {

if (selectedItems.length === 0) {

return;

}

if (confirm('Удалить выбранные элементы?')) {

const promises = [];

selectedItems.forEach(item => {

promises.push(DataService.delete(props.url + item));

});

Promise.all(promises).then((results) => {

selectedItems.length = 0;

loadItems();

});

}

}

function handleTableClick(tableSelectedItems) {

selectedItems = tableSelectedItems;

}

function handleTableDblClick(tableSelectedItem) {

edit(tableSelectedItem);

}

function handleModalHide() {

setModalVisible(false);

}

function handleModalDone() {

saveItem();

}

return (

<>

<Toolbar

onAdd={handleAdd}

onEdit={handleEdit}

onRemove={handleRemove}/>

<Table

headers={props.headers}

items={items}

selectable={true}

onClick={handleTableClick}

onDblClick={handleTableDblClick}/>

<Modal

header={modalHeader}

confirm={modalConfirm}

visible={modalVisible}

onHide={handleModalHide}

onDone={handleModalDone}>

{props.children}

</Modal>

</>

);

}

CatalogHistory.jsx

import { useState, useEffect } from "react";

import Table from "../common/Table";

import Modal from "../common/Modal";

import DataService from '../../services/DataService';

export default function CatalogHistory(props) {

const [items, setItems] = useState([]);

const [modalHeader, setModalHeader] = useState('');

const [modalConfirm, setModalConfirm] = useState('');

const [modalVisible, setModalVisible] = useState(false);

const [isEdit, setEdit] = useState(false);

useEffect(() => {

loadItems();

}, []);

function loadItems() {

DataService.readAll(props.getAllUrl, props.transformer)

.then(data => setItems(data));

}

function saveItem() {

if (!isEdit) {

DataService.create(props.getAllUrl, props.data).then(() => loadItems());

} else {

DataService.update(props.url + props.data.id, props.data).then(() => loadItems());

}

}

function edit(editedId) {

DataService.read(props.url + editedId, props.transformer)

.then(data => {

setEdit(true);

setModalHeader('Редактирование элемента');

setModalConfirm('Сохранить');

setModalVisible(true);

props.onEdit(data);

});

}

function handleModalHide() {

setModalVisible(false);

}

function handleModalDone() {

saveItem();

}

return (

<>

<Table

headers={props.headers}

items={items}

/>

<Modal

header={modalHeader}

confirm={modalConfirm}

visible={modalVisible}

onHide={handleModalHide}

onDone={handleModalDone}>

{props.children}

</Modal>

</>

);

}

CatalogProduct.jsx

import { useState, useEffect, Component } from "react";

import Toolbar from "../common/ToolbarProduct";

import Card from "../common/Card";

import ModalProduct from "../common/ModalProduct";

import DataService from "../../services/DataService";

import Table from "../common/Table";

import ToolbarProduct from "../common/Toolbar";

export default function CatalogProduct(props) {

const [items, setItems] = useState([]);

const [modalHeader, setModalHeader] = useState("");

const [modalConfirm, setModalConfirm] = useState("");

const [modalVisible, setModalVisible] = useState(false);

const [isEdit, setEdit] = useState(false);

const [componentProduct, setComponentProduct] = useState({});

useEffect(() => {

loadItems();

}, []);

function loadItems() {

DataService.readAll(props.url, props.transformer).then((data) =>{

setItems(data);

}

);

DataService.readAll(props.url, props.transformer).then((data) =>

setItems(data)

);

}

let selectedItems = [];

async function saveItem() {

if (!isEdit) {

await props.set();

DataService.create(props.url, props.data).then(() => loadItems());

} else {

await props.set();

DataService.update(props.url + "/" + props.data.id, props.data).then(() =>

loadItems()

);

}

}

function handleAdd() {

setEdit(false);

setModalHeader("Добавление элемента");

setModalConfirm("Добавить");

setModalVisible(true);

props.onAdd();

}

function handleEdit(id) {

if (selectedItems.length === 0) {

return;

}

setComponentProduct(

props.componentProduct.filter((x) => x.id == selectedItems[0])[0]

);

}

function edit(editedId) {

DataService.read(props.url+ "/" + editedId, props.transformer).then((data) => {

for(let i = 0; i < data.componentDTOList.length; i++){

props.componentProduct.push(data.componentDTOList[i]);

props.setcomponentProduct(props.componentProduct)

}

setEdit(true);

setModalHeader("Редактирование элемента");

setModalConfirm("Сохранить");

setModalVisible(true);

props.onEdit(data);

});

}

function handleRemove(id, e) {

if (selectedItems.length === 0) {

return;

}

if (confirm("Удалить выбранные элементы?")) {

const promises = [];

selectedItems.forEach((item) => {

props.deleteComponents(item);

});

selectedItems.length = 0;

}

}

function handleModalHide() {

setModalVisible(false);

props.setcomponentProduct([]);

}

function handleModalDone() {

saveItem();

}

async function handleAddComponent() {

if (

props.componentProduct.filter((x) => x.id == componentProduct.id)

.length != 0

) {

await props.updateComponents(componentProduct, isEdit);

setComponentProduct({});

let count = 0;

for (let i = 0; i < props.componentProduct.length; i++) {

count +=

props.componentProduct[i].price \* props.componentProduct[i].count;

}

props.data.price = count;

if(isEdit){

await props.setprice(count);

return;

}

return;

}

props.componentProduct.push(componentProduct);

await props.setcomponentProduct(props.componentProduct);

setComponentProduct({});

let count = 0;

for (let i = 0; i < props.componentProduct.length; i++) {

count +=

props.componentProduct[i].price \* props.componentProduct[i].count;

}

props.data.price = count;

}

function handleRemoveProduct(id) {

if (confirm("Удалить выбранные элементы?")) {

DataService.delete(props.url + "/" + id).then(() => {

loadItems();

});

}

}

function handleFormChangeComponent(event) {

if (event.target.id === "componentName") {

setComponentProduct({

...componentProduct,

["id"]: event.target.value,

["componentName"]: props.component

.filter((x) => x.id == event.target.value)

.map((x) => x.componentName)[0],

["price"]: props.component

.filter((x) => x.id == event.target.value)

.map((x) => x.price)[0],

});

return;

}

setComponentProduct({

...componentProduct,

[event.target.id]: Number(event.target.value),

});

return;

}

const [imageURL, setImageURL] = useState();

const fileReader = new FileReader();

fileReader.onloadend = () => {

const tempval = fileReader.result;

setImageURL(tempval);

props.setData(tempval);

};

function handleOnChange(event) {

event.preventDefault();

const file = event.target.files[0];

fileReader.readAsDataURL(file);

}

function handleTableClick(tableSelectedItems) {

selectedItems = tableSelectedItems;

}

function handleTableDblClick(tableSelectedItem) {

setComponentProduct(

props.componentProduct.filter((x) => x.id == tableSelectedItem)[0]

);

}

return (

<>

<Toolbar onAdd={handleAdd} />

<Card items={items} onEdit={edit} onRemove={handleRemoveProduct} product={props.product} setProduct={props.setProduct}/>

<ModalProduct

header={modalHeader}

confirm={modalConfirm}

visible={modalVisible}

onHide={handleModalHide}

onDone={handleModalDone}

>

<div className="mb-3">

<label htmlFor="name" className="form-label">

Название продукта

</label>

<input

type="text"

id="name"

className="form-control"

required

value={props.data.name}

onChange={props.handleFormChange}

/>

</div>

<div className="mb-3">

<label htmlFor="price" className="form-label">

Цена

</label>

<input

type="text"

id="price"

className="form-control"

required

value={props.data.price}

onChange={props.handleFormChange}

/>

</div>

<div className="col-mb-3">

<label className="form-label" htmlFor="picture">

Изображение

</label>

<input

className="form-control"

id="picture"

type="file"

accept="image/jpeg, image/png, image/jpg"

value=""

onChange={handleOnChange}

/>

</div>

<div className="mb-3">

<label htmlFor="groupId" className="form-label">

Компонент

</label>

<select

id="componentName"

className="form-select"

required

value={componentProduct.id}

onChange={handleFormChangeComponent}

>

<option disabled value="">

Укажите группу

</option>

{props.component.map((group) => (

<option key={group.id} value={group.id}>

{group.componentName}

</option>

))}

</select>

<label htmlFor="count" className="form-label">

Количество

</label>

<input

type="text"

id="count"

className="form-control"

required

value={componentProduct.count ?? 0}

onChange={handleFormChangeComponent}

/>

</div>

<ToolbarProduct

onAdd={handleAddComponent}

onEdit={handleEdit}

onRemove={handleRemove}

/>

<Table

headers={props.catalogStudHeaders}

items={props.componentProduct}

allItems={props.component}

selectable={true}

onClick={handleTableClick}

onDblClick={handleTableDblClick}

/>

</ModalProduct>

</>

);

}

CatalogStudents.jsx

import { useState, useEffect } from 'react';

import Catalog from './Catalog';

import Component from '../../models/Component';

import DataService from '../../services/DataService';

export default function CatalogStudents(props) {

const getAllUrl = '/component';

const url = '/component/';

const transformer = (data) => new Component(data);

const catalogStudHeaders = [

{ name: 'componentName', label: 'Название компонента' },

{ name: 'price', label: 'Цена' }

];

const [data, setData] = useState(new Component());

function handleOnAdd() {

setData(new Component());

}

function handleOnEdit(data) {

setData(new Component(data));

}

function handleFormChange(event) {

setData({ ...data, [event.target.id]: event.target.value })

}

return (

<main className="flex-shrink-0" style={{ backgroundColor: "white" }}>

<Catalog

headers={catalogStudHeaders}

getAllUrl={getAllUrl}

url={url}

transformer={transformer}

data={data}

onAdd={handleOnAdd}

onEdit={handleOnEdit}>

<div className="mb-3">

<label htmlFor="componentName" className="form-label">Название компонента</label>

<input type="text" id="componentName" className="form-control" required

value={data.componentName} onChange={handleFormChange}/>

</div>

<div className="mb-3">

<label htmlFor="price" className="form-label">Цена</label>

<input type="text" id="price" className="form-control" required

value={data.price} onChange={handleFormChange}/>

</div>

</Catalog>

</main>

);

}

History.jsx

import { useState, useEffect } from 'react';

import Catalog from './CatalogHistory';

import Component from '../../models/Component';

import DataService from '../../services/DataService';

import Order from '../../models/Order';

export default function CatalogStudents(props) {

const getAllUrl = '/order';

const url = '/order/';

const transformer = (data) => new Order(data);

const catalogStudHeaders = [

{ name: 'date', label: 'Дата оформления' },

{ name: 'price', label: 'Общая стоимость' },

{ name: 'status', label: 'Статус' }

];

const [data, setData] = useState(new Order());

function handleOnAdd() {

setData(new Order());

}

function handleOnEdit(data) {

setData(new Order(data));

}

function handleFormChange(event) {

setData({ ...data, [event.target.id]: event.target.value })

}

return (

<main className="flex-shrink-0" style={{ backgroundColor: "white" }}>

<Catalog

headers={catalogStudHeaders}

getAllUrl={getAllUrl}

url={url}

transformer={transformer}

data={data}

onAdd={handleOnAdd}

onEdit={handleOnEdit}>

<div className="mb-3">

<label htmlFor="componentName" className="form-label">Название компонента</label>

<input type="text" id="componentName" className="form-control" required

value={data.componentName} onChange={handleFormChange}/>

</div>

<div className="mb-3">

<label htmlFor="price" className="form-label">Цена</label>

<input type="text" id="price" className="form-control" required

value={data.price} onChange={handleFormChange}/>

</div>

</Catalog>

</main>

);

}

Menu.jsx

import { useState, useEffect, Component } from "react";

import CatalogProduct from "./CatalogProduct";

import Product from "../../models/Product";

import Components from "../../models/Component";

import DataService from "../../services/DataService";

export default function Menu(props) {

const url = "/product";

const categoryUrl = "/component";

const transformer = (data) => new Product(data);

const catalogStudHeaders = [

{ name: "componentName", label: "Название компонента" },

{ name: "count", label: "Количество" },

];

const [data, setData] = useState(new Product());

const [component, setComponent] = useState([]);

const [componentProduct, setComponentProduct] = useState([]);

useEffect(() => {

DataService.readAll(categoryUrl, (data) => new Components(data)).then(

(data) => setComponent(data)

);

}, []);

function handleOnAdd() {

setData(new Product());

setComponentProduct([]);

}

function handleOnEdit(data) {

setData(new Product(data));

}

function handleFormChange(event) {

setData({ ...data, [event.target.id]: event.target.value });

}

const [imageURL, setImageURL] = useState();

const fileReader = new FileReader();

fileReader.onloadend = () => {

const tempval = fileReader.result;

setImageURL(tempval);

setData({ ...data, ["picture"]: tempval });

};

function handleOnChange(event) {

event.preventDefault();

const file = event.target.files[0];

fileReader.readAsDataURL(file);

}

function handleUpdateComponents(value, bool) {

if (bool) {

let temp = data.componentDTOList.filter((x) => x.id != value.id);

data.componentDTOList = [];

for (let i = 0; i < temp.length; i++) {

data.componentDTOList.push(temp[i]);

}

data.componentDTOList.push(value);

setData(data);

}

setComponentProduct(

componentProduct.map((obj) => {

if (obj.id == value.id) {

return { ...obj, count: value.count };

} else {

return obj;

}

})

);

}

function setDataa(value) {

setData({ ...data, ["image"]: value });

}

function setComponents() {

data.componentDTOList = [];

for (let i = 0; i < componentProduct.length; i++) {

data.componentDTOList.push(componentProduct[i]);

}

setData(data);

}

function handleDeleteComponents(value) {

setComponentProduct(componentProduct.filter((x) => x.id !== value));

}

function setprice(value){

let count = 0;

for (let i = 0; i < data.componentDTOList.length; i++) {

count +=

data.componentDTOList[i].price \* data.componentDTOList[i].count;

}

data.price = count;

}

return (

<main className="flex-shrink-0">

<CatalogProduct

url={url}

transformer={transformer}

data={data}

onAdd={handleOnAdd}

onEdit={handleOnEdit}

handleFormChange={handleFormChange}

handleOnChange={handleOnChange}

component={component}

catalogStudHeaders={catalogStudHeaders}

componentProduct={componentProduct}

setcomponentProduct={setComponentProduct}

updateComponents={handleUpdateComponents}

deleteComponents={handleDeleteComponents}

setData={setDataa}

set={setComponents}

setprice={setprice}

product={props.product}

setProduct={props.setProduct}

/>

</main>

);

}

App.css

nav {

background-color: #c8afaf;

}

main {

background-color: #8c7b7b;

min-height: 90vh;

}

footer {

background-color: #c8afaf;

color: black;

}

.size {

overflow-x: hidden;

}

.card {

background-color: #8c7b7b;

border-color: #8c7b7b;

}

.card-body {

background-color: #8c7b7b;

}

h1 {

word-wrap: break-word;

}

#banner {

margin: 0px 15px 15px 15px;

padding-top: 15px;

display: flex;

align-items: center;

flex-direction: column;

}

#banner img {

border-radius: 5px;

}

#banner img.show {

width: 100%;

opacity: 1;

transition: opacity 1s, visibility 0s;

}

#banner img.hide {

height: 0;

width: 0;

opacity: 0;

visibility: hidden;

transition: opacity 1s, visibility 0s 1s;

}

@media (max-width: 767px) {

.btn {

padding: 1px 5px;

font-size: 12px;

line-height: 1.5;

border-radius: 3px;

}

}

@media (min-width: 768px) and (max-width: 991px) {

.btn {

padding: 5px 10px;

font-size: 12px;

line-height: 1.5;

border-radius: 3px;

}

}

@media (min-width: 992px) and (max-width: 1199px) {

.btn {

padding: 6px 12px;

font-size: 14px;

line-height: 1.42857143;

border-radius: 4px;

}

}

@media (min-width: 1200px) {

.btn {

padding: 10px 16px;

font-size: 18px;

line-height: 1.3333333;

border-radius: 6px;

}

}

@media (max-width: 600px) {

.tem {

padding-right: 50px;

}

}