```
import React, { useEffect, useState } from 'react'
import { Navigate } from 'react-router-dom'
import { roles_leader } from '../../constants/roles/LEADER_ROLES'
import PercentUI from '../components/UI/percent'
import LoadAnimate from '../components/UI/loadanimate'
import { api } from '../../Api'

export default function AddLeaderOtchet() {

    const [inputValues, setInputValues] = useState({})
    const [inputValues2, setInputValues2] = useState({})
    const [responseData, setResponseData] = useState([])
    const [dateValue, setDateValue] = useState('');
    const [clients, setClients] = useState([])
    const [clientdata, setClientdata] = useState([])
    const [loading, setLoading] = useState(true);
    const [loading2, setLoading2] = useState(true);
    const [loading3, setLoading3] = useState(true);
    const [loading4, setLoading4] = useState(true);
    const [wait, setWait] = useState(false)

    const formatDateForDatabase = (inputDate) => {
        const dateParts = inputDate.split('-');
        const formattedDate = `${dateParts[2]}.${dateParts[1]}.${dateParts[0]}`;
        return formattedDate;
    };

    const createTable = async () => {
        setWait(false)
        try {
            const response = await api.post('/newotchet/liderotchetbeta');
            setLoading2(false);
            fetchData();
        } catch (error) {
            console.log(error);
        }
    }

    const handleSubmit = async () => {
        setLoading(false)
        try {
            const updatedDataPromises = responseData.flatMap(elem =>
                elem.otchet.map(item => {
                    const smValue = inputValues[item._id]?.sm || item.sm;
                    const summa = (inputValues[item._id]?.comPersent100 * 0.4).toFixed(0);
                    const today = new Date();
                    const dd = String(today.getDate()).padStart(2, '0');
                    const mm = String(today.getMonth() + 1).padStart(2, '0')
                    const yyyy = today.getFullYear();

                    const twoMonthsLater = new Date(today);
                    twoMonthsLater.setMonth(today.getMonth() + 2);

                    const ddd = String(twoMonthsLater.getDate()).padStart(2, '0');
                    const mmm = String(twoMonthsLater.getMonth() + 1).padStart(2, '0');
                    const yyyyy = twoMonthsLater.getFullYear();

                    const formattedTwoMonthsLater = ddd + '.' + mmm + '.' + yyyyy;
                    const formattedDate = dd + '.' + mm + '.' + yyyy;
```

```
                if (smValue === 3) {
                    setClients(prevClients => ({
                        ...prevClients,
                        [item._id]: {
                            clients: inputValues[item._id]?.sity === '' ? '' : inputValues[item._id]?.sity ||
item.sity,
                            buyer_logist: inputValues[item._id]?.buyer === '' ? '' : inputValues[item._id]?.buyer
|| item.buyer,
                            date_to: formattedDate,
                            date_go: formattedTwoMonthsLater,
                            summa: summa,
                            order_count: 1,
                            status: true
                        }
                    }))
                } else if (smValue === 4) {
                    setClients(prevClients => ({
                        ...prevClients,
                        [item._id]: {
                            clients: inputValues[item._id]?.sity === '' ? '' : inputValues[item._id]?.sity ||
item.sity,
                            buyer_logist: inputValues[item._id]?.admin === '' ? '' :
inputValues[item._id]?.admin || item.admin,
                            date_to: formattedDate,
                            date_go: formattedTwoMonthsLater,
                            summa: summa,
                            order_count: 1,
                            status: true
                        }
                    }))
                } else if (smValue === 5) {
                    setClients(prevClients => ({
                        ...prevClients,
                        [item._id]: {
                            clients: inputValues[item._id]?.sity === '' ? '' : inputValues[item._id]?.sity ||
item.sity,
                            buyer_logist: "x% online",
                            date_to: formattedDate,
                            date_go: formattedTwoMonthsLater,
                            summa: summa,
                            order_count: 1,
                            status: true
                        }
                    }))
                } else if (smValue === 2) {
                    setClients(prevClients => {
                        const { [item._id]: deletedClient, ...newClients } = prevClients;
                        return newClients;
                    });
                } else if (smValue === 1) {
                    setClients(prevClients => {
                        const { [item._id]: deletedClient, ...newClients } = prevClients;
                        return newClients;
                    });
                }
                const updatedData = {
                    sm: inputValues[item._id]?.sm || item.sm,
                    sity: inputValues[item._id]?.sity === '' ? '' : inputValues[item._id]?.sity || item.sity,
                    admin: inputValues[item._id]?.admin === '' ? '' : inputValues[item._id]?.admin ||
item.admin,
```

```
                    buyer: inputValues[item._id]?.buyer === '' ? '' : inputValues[item._id]?.buyer ||
item.buyer,
                    comPersent100: inputValues[item._id]?.comPersent100 === '' ? '' :
inputValues[item._id]?.comPersent100 || item.comPersent100,
                    comPersent2: inputValues[item._id]?.comPersent2 === '' ? '' :
inputValues[item._id]?.comPersent2 || item.comPersent2,
                    comPersent3: inputValues[item._id]?.comPersent3 === '' ? '' :
inputValues[item._id]?.comPersent3 || item.comPersent3,
                    comPersent4: inputValues[item._id]?.comPersent4 === '' ? '' :
inputValues[item._id]?.comPersent4 || item.comPersent4,
                    indexPersent100: inputValues[item._id]?.indexPersent100 === '' ? '' :
inputValues[item._id]?.indexPersent100 || item.indexPersent100,
                    indexPersent2: inputValues[item._id]?.indexPersent2 === '' ? '' :
inputValues[item._id]?.indexPersent2 || item.indexPersent2,
                    indexPersent3: inputValues[item._id]?.indexPersent3 === '' ? '' :
inputValues[item._id]?.indexPersent3 || item.indexPersent3,
                    indexPersent4: inputValues[item._id]?.indexPersent4 === '' ? '' :
inputValues[item._id]?.indexPersent4 || item.indexPersent4,
                    uhod: inputValues[item._id]?.uhod === '' ? '' : inputValues[item._id]?.uhod || item.uhod,
                    prihod: inputValues[item._id]?.prihod === '' ? '' : inputValues[item._id]?.prihod ||
item.prihod,
                    itog: inputValues[item._id]?.itog === '' ? '' : inputValues[item._id]?.itog || item.itog,
                    itogIndex: inputValues[item._id]?.itogIndex === '' ? '' : inputValues[item._id]?.itogIndex
|| item.itogIndex,
                };
                return api.patch(`/update/liderotchetbetas/${item._id}`, updatedData);
            })
        );

        await Promise.all(updatedDataPromises);

        const sumAllItog = responseData.map(elem =>
            elem.otchet.reduce((acc, elem) => {
                const inputValue = inputValues[elem._id]?.itog || 0;
                return acc + (inputValue ? parseFloat(inputValue) : 0);
            }, 0)
        );

        const sumAllItogcashback10 = responseData.map(elem =>
            elem.otchet.reduce((acc, elem) => {
                const inputValue = inputValues[elem._id]?.sm == 1 || inputValues[elem._id]?.sm == 2 ?
(inputValues[elem._id]?.comPersent100 * 0.1) : 0
                return acc + (inputValue ? parseFloat(inputValue) : 0);
            }, 0)
        );

        const sumAllItogIndex = responseData.map(elem =>
            elem.otchet.reduce((acc, elem) => {
                const inputValue = inputValues[elem._id]?.itogIndex || 0;
                return acc + (inputValue ? parseFloat(inputValue) : 0);
            }, 0)
        );

        const sumAllUhod = Object.values(inputValues).reduce((acc, item) => {
            const uhod = item?.uhod || 0;
            return acc + (uhod ? parseFloat(uhod) : 0);
        }, 0);

        const sumAllPrihod = Object.values(inputValues).reduce((acc, item) => {
            const prihod = item?.prihod || 0;
```

```
            return acc + (prihod ? parseFloat(prihod) : 0);
        }, 0);

        const itogSum = parseFloat(sumAllItog) + parseFloat(sumAllItogIndex) +
parseFloat(sumAllPrihod);


        const allraznica = sumAllItogcashback10;

        const updatedInputValues = { ...inputValues };
        responseData.map(elem => {
            elem.otchet.forEach(elem => {
                updatedInputValues[elem._id] = {
                    sm: inputValues[elem._id]?.sm || elem.sm,
                    sity: inputValues[elem._id]?.sity || elem.sity,
                    admin: inputValues[elem._id]?.admin || elem.admin,
                    buyer: inputValues[elem._id]?.buyer || elem.buyer,
                    comPersent100: inputValues[elem._id]?.comPersent100 || elem.comPersent100,
                    comPersent2: inputValues[elem._id]?.comPersent2 || elem.comPersent2,
                    comPersent3: inputValues[elem._id]?.comPersent3 || elem.comPersent3,
                    comPersent4: inputValues[elem._id]?.comPersent4 || elem.comPersent4,
                    indexPersent100: inputValues[elem._id]?.indexPersent100 || elem.indexPersent100,
                    indexPersent2: inputValues[elem._id]?.indexPersent2 || elem.indexPersent2,
                    indexPersent3: inputValues[elem._id]?.indexPersent3 || elem.indexPersent3,
                    indexPersent4: inputValues[elem._id]?.indexPersent4 || elem.indexPersent4,
                    uhod: inputValues[elem._id]?.uhod || elem.uhod,
                    prihod: inputValues[elem._id]?.prihod || elem.prihod,
                    itog: inputValues[elem._id]?.itog || elem.itog,
                    itogIndex: inputValues[elem._id]?.itogIndex || elem.itogIndex,
                };
            });
        });

        setInputValues(updatedInputValues);

        const updatedDataPromises2 = responseData.flatMap(elem =>
            elem.itog.flatMap(elem => {
                const sum1 = parseFloat(inputValues2[elem._id]?.sum1) || 0;
                const sum2 = parseFloat(inputValues2[elem._id]?.sum2) || 0;
                const sum3 = parseFloat(inputValues2[elem._id]?.sum3) || 0;
                const sum4 = parseFloat(inputValues2[elem._id]?.sum4) || 0;
                const sum5 = parseFloat(inputValues2[elem._id]?.sum5) || 0;
                const itogs = parseFloat(sum1) - parseFloat(sum2) - parseFloat(sum3) -
parseFloat(sum4) - parseFloat(sum5);

                const updatedData2 = {
                    ros1: inputValues2[elem._id]?.ros1 === '' ? '' : inputValues2[elem._id]?.ros1 ||
elem.ros1,
                    ros2: inputValues2[elem._id]?.ros2 === '' ? '' : inputValues2[elem._id]?.ros2 ||
elem.ros2,
                    ros3: inputValues2[elem._id]?.ros3 === '' ? '' : inputValues2[elem._id]?.ros3 ||
elem.ros3,
                    ros4: inputValues2[elem._id]?.ros4 === '' ? '' : inputValues2[elem._id]?.ros4 ||
elem.ros4,
                    ros5: inputValues2[elem._id]?.ros5 === '' ? '' : inputValues2[elem._id]?.ros5 ||
elem.ros5,
                    sum1: inputValues2[elem._id]?.sum1 === '' ? '' : inputValues2[elem._id]?.sum1 ||
elem.sum1,
                    sum2: inputValues2[elem._id]?.sum2 === '' ? '' : inputValues2[elem._id]?.sum2 ||
elem.sum2,
```

```
                    sum3: inputValues2[elem._id]?.sum3 === '' ? '' : inputValues2[elem._id]?.sum3 ||
elem.sum3,
                    sum4: inputValues2[elem._id]?.sum4 === '' ? '' : inputValues2[elem._id]?.sum4 ||
elem.sum4,
                    sum5: inputValues2[elem._id]?.sum5 === '' ? '' : inputValues2[elem._id]?.sum5 ||
elem.sum5,
                    allItogIndex: sumAllItogIndex.toString(),
                    allItog: sumAllItog.toString(),
                    allItogPrihod: sumAllPrihod.toString(),
                    allItogUhod: sumAllUhod.toString(),
                    raznica: allraznica.toString(),
                    itogs: parseFloat(itogSum) - parseFloat(itogs),
                };

                return api.patch(`/update/liderotchetbetasitog/${elem._id}`, updatedData2);
            })
        );

        await Promise.all(updatedDataPromises2);

        const updatedInputValues2 = { ...inputValues2 };
        responseData.map(elem => {
            elem.itog.forEach(elem => {
                updatedInputValues2[elem._id] = {
                    ros1: inputValues2[elem._id]?.ros1 || elem.ros1,
                    ros2: inputValues2[elem._id]?.ros2 || elem.ros2,
                    ros3: inputValues2[elem._id]?.ros3 || elem.ros3,
                    ros4: inputValues2[elem._id]?.ros4 || elem.ros4,
                    ros5: inputValues2[elem._id]?.ros5 || elem.ros5,
                    sum1: inputValues2[elem._id]?.sum1 || elem.sum1,
                    sum2: inputValues2[elem._id]?.sum2 || elem.sum2,
                    sum3: inputValues2[elem._id]?.sum3 || elem.sum3,
                    sum4: inputValues2[elem._id]?.sum4 || elem.sum4,
                    sum5: inputValues2[elem._id]?.sum5 || elem.sum5,
                    allItogIndex: inputValues2[elem._id]?.allItogIndex || elem.allItogIndex,
                    allItog: inputValues2[elem._id]?.allItog || elem.allItog,
                    allItogPrihod: inputValues2[elem._id]?.allItogPrihod || elem.allItogPrihod,
                    allItogUhod: inputValues2[elem._id]?.allItogUhod || elem.allItogUhod,
                    raznica: inputValues2[elem._id]?.raznica || elem.raznica,
                    itogs: inputValues2[elem._id]?.itogs || elem.itogs,
                };
            });
        });

        setInputValues2(updatedInputValues2);
        setWait(false)
        fetchData();
    } catch (error) {
        console.error('Ошибка при обновлении данных:', error);
    }
};


const fetchData = async () => {
    const response = await api.get('/test/liderotchetbeta')
    setResponseData(response.data)
    setWait(true)
    setLoading(true)
    setLoading4(true)
    setLoading3(true)
```

```
    setLoading2(true)
    const initialInputValues = {};

    response.data.map(elem => {
        elem.otchet.forEach(elem => {
            initialInputValues[elem._id] = {
                list: elem.list,
                sm: elem.sm || 0,
                sity: elem.sity || "",
                admin: elem.admin || "",
                buyer: elem.buyer || "",
                comPersent100: elem.comPersent100 || 0,
                comPersent2: elem.comPersent2 || 0,
                comPersent3: elem.comPersent3 || 0,
                comPersent4: elem.comPersent4 || 0,
                indexPersent100: elem.indexPersent100 || 0,
                indexPersent2: elem.indexPersent2 || 0,
                indexPersent3: elem.indexPersent3 || 0,
                indexPersent4: elem.indexPersent4 || 0,
                uhod: elem.uhod || 0,
                prihod: elem.prihod || 0,
                itog: elem.itog || 0,
                itogIndex: elem.itogIndex || 0,
            }
        });
    })

    setInputValues(initialInputValues);

    const initialInputValues2 = {};
    response.data.map(elem => {
        elem.itog.forEach(elem => {
            initialInputValues2[elem._id] = {
                ros1: elem.ros1 || '',
                ros2: elem.ros2 || '',
                ros3: elem.ros3 || '',
                ros4: elem.ros4 || '',
                ros5: elem.ros5 || '',
                sum1: elem.sum1 || 0,
                sum2: elem.sum2 || 0,
                sum3: elem.sum3 || 0,
                sum4: elem.sum4 || 0,
                sum5: elem.sum5 || 0,
                allItogIndex: elem.allItogIndex || 0,
                allItog: elem.allItog || 0,
                allItogPrihod: elem.allItogPrihod || 0,
                allItogUhod: elem.allItogUhod || 0,
                raznica: elem.raznica || 0,
                itogs: elem.itogs || 0,
            }
        })
    })

    setInputValues2(initialInputValues2)

}

const [data, setData] = useState()
const [selectedTeam, setSelectedTeam] = useState('lider')
```

```
const fetchDataBuyer = async () => {
    try {
        if (selectedTeam === 'lider') {
            const response = await api.get('/test/simCardLiders');
            setData(response.data);
        } else if (selectedTeam === 'liberty') {
            const response = await api.get('/test/simCardLiberty');
            setData(response.data);
        } else if (selectedTeam === 'monaco') {
            const response = await api.get('/test/simCardMonacos');
            setData(response.data);
        } else if (selectedTeam === 'turan') {
            const response = await api.get('/test/simCardTurans');
            setData(response.data);
        } else if (selectedTeam === 'fenix') {
            const response = await api.get('/test/simCardFenixes');
            setData(response.data);
        } else if (selectedTeam === 'fbox') {
            const response = await api.get('/test/simCardNewOtdel');
            setData(response.data);
        }
    } catch (error) {
        console.error(error);
    }
};


useEffect(() => {
    fetchDataBuyer();
}, [selectedTeam]);

const [selectedCurator, setSelectedCurator] = useState('');

const [curatorData, setCuratorData] = useState([]);

const handleCuratorChange = async (event) => {
    const selectedCuratorValue = event.target.value;
    setSelectedCurator(selectedCuratorValue);

    const selectedCuratorData = data.find((item) => item.curator === selectedCuratorValue);
    if (selectedCuratorData && selectedCuratorData.slot.length > 0) {
        setCuratorData(selectedCuratorData.slot);
    } else {
        setCuratorData([]);
    }
};


const handleBuyerChange = (event, id) => {
    const { value } = event.target;
    setInputValues((prevInputValues) => ({
        ...prevInputValues,
        [id]: {
            ...prevInputValues[id],
            buyer: value,
        },
    }));
};
```

```
    useEffect(() => {
        fetchData()
        fetchDataclient()
        fetchLogist()
    }, [])


    const deleteTable = async () => {
        try {
            await api.delete('/test/liderotchetbeta');
        } catch (error) {
            console.error('Error deleting table:', error);
        }
    };

    const addSoloTable = async (id) => {
        setLoading4(false)
        try {
            await api.post('/insert/liderotchetbeta', { id });
            setWait(false)
            fetchData()
        } catch (error) {
            console.error('Ошибка при добавлении', error);
        }
    }

    const fetchDataclient = async () => {
        const response = await api.get('/leaderclient')
        setClientdata(response.data)
    }


    const isClientExist = (clientId, clientdata) => {
        return clientdata.some(client => {
            return (
                client.clients.toLowerCase().replace(/\s/g, '') ===
clientId.clients.toLowerCase().replace(/\s/g, '') &&
                client.buyer_logist === clientId.buyer_logist
            );
        });
    };

    const getCurrentDate = () => {
        const today = new Date();
        const dd = String(today.getDate()).padStart(2, '0');
        const mm = String(today.getMonth() + 1).padStart(2, '0');
        const yyyy = today.getFullYear();
        return dd + '.' + mm + '.' + yyyy;
    };

    const parseDate = (dateString) => {
        const [day, month, year] = dateString.split('.').map(Number);
        return new Date(year, month - 1, day);
    };

    const isClientExisValue = (clientId, clientdata) => {
        return clientdata.some(client => {
            return (
                client.clients.toLowerCase().replace(/\s/g, '') ===
clientId.clients.toLowerCase().replace(/\s/g, '') &&
```

```
            client.buyer_logist === clientId.buyer_logist &&
            parseFloat(client.summa) <= 50000 &&
            parseFloat(client.order_count) <= 10 &&
            parseDate(client.date_go) > parseDate(getCurrentDate())
        );
    });
};

const Clients = async () => {
    try {
        for (const [clientId, clientData] of Object.entries(clients)) {
            try {
                if (isClientExist(clientData, clientdata)) {
                    if (isClientExisValue(clientData, clientdata)) {
                        const existingClient = clientdata.find(client =>
client.clients.toLowerCase().replace(/\s/g, '') === clientData.clients.toLowerCase().replace(/\s/g, ''));
                        await api.patch(`/leaderclient-update/${existingClient._id}`, { summa:
clientData.summa, order_count: clientData.order_count });
                    } else {
                        console.log(`Клиент ${clientData.clients} превысил лимит`);
                    }
                } else {
                    await api.post(`/new-leaderclient/${clientId}`, clientData);
                }
            } catch (error) {
                console.error(`Ошибка при обновлении или создании клиента ${clientData.clients}:`,
error);
            }
        }
    } catch (error) {
        console.error('Ошибка при обновлении или создании клиентов:', error);
    }
};

const handleSend = async () => {
    if (!dateValue) {
        alert('Заполните поле даты');
        return;
    }

    const formattedDate = formatDateForDatabase(dateValue);
    const [datas] = responseData;
    const filteredOtchet = datas.otchet.filter(item => item.sity !== "" && item.comPersent !== 0);
    const data = {
        date: formattedDate,
        otchet: filteredOtchet,
        itog: datas.itog
    };

    try {
        setLoading3(false);
        await api.post('/test/liderdatas', data);
        await Clients();
        setWait(false)
        fetchData()
    } catch (error) {
        console.error('Ошибка:', error);
    }
};
```

```
    const [logist, setLogist] = useState([])

    const fetchLogist = async () => {
        try {
            const response = await api.get('/test/simCardLiderLogs')
            const response2 = await api.get('/test/simCardLibertyLogs')
            const response3 = await api.get('/adminlogist')
            const filtered = response3.data.filter(item => item.team === 'leader')
            setLogist([...response.data, ...response2.data, ...filtered])
        } catch (error) {
            console.log(error);
        }
    }

    let filteredLogist = logist
        .flatMap(item => item.slot)
        .filter(log => log.logist !== '' && log.status === "2")
        .map(item => item.logist)

    const roles = JSON.parse(localStorage.getItem('roles'));
    if ((roles && hasAccess(roles.roles.role, roles_leader)) || roles.roles.role.includes("ВМ ЛДР")) {
        return (
            <div>
                {wait ? (<div className='mx-auto '>
                    {responseData.length ? <PercentUI /> : ''}
                    <div className='flex gap-5'>
                        {
                            responseData.length ? (<div className='flex justify-center items-center mb-5
gap-2'>
                                <label htmlFor="curatorSelect" className="block lg:text-[12px] text-[6px]
font-semibold">
                                    Выберите команду:
                                </label>
                                <select
                                    value={selectedTeam}
                                    onChange={e => setSelectedTeam(e.target.value)}
                                    className="border px-4 py-2 lg:text-[12px] text-[5px] rounded-md bg-gray-100
text-gray-800 shadow-md"
                                >
                                    <option value="lider">лидер</option>
                                    <option value="liberty">liberty</option>
                                    <option value="turan">туран</option>
                                    <option value="monaco">монако</option>
                                    <option value="fenix">ильяс</option>
                                    <option value="fbox">fbox</option>
                                </select>

                            </div>) : ''
                        }
                        {responseData.length ? (<div className='flex justify-center items-center mb-5 gap-2'>
                            <label htmlFor="curatorSelect" className="block lg:text-[12px] text-[6px]
font-semibold">
                                Выберите куратора:
                            </label>
                            {data ? (
                                <select
                                    id="curatorSelect"
                                    value={selectedCurator}
                                    onChange={handleCuratorChange}
```

```
                        className="border px-4 py-2 lg:text-[12px] text-[5px] rounded-md bg-gray-100
text-gray-800 shadow-md"
                    >
                        <option value="">Не выбрано</option>
                        {data.map((item) => (
                            <option
                                key={item._id}
                                value={item.curator}
                                className="bg-white hover:bg-blue-500 hover:text-white"
                            >
                                {item.curator}
                            </option>
                        ))}
                    </select>
                ) : (
                    ""
                )}
            </div>) : ''}
        </div>
        <div className="overflow-x-auto flex flex-col items-center justify-center">
            {responseData.map((elem) => {
                return (
                    <div>
                        <table className="bg-base-100 min-w-full text-center border
border-collapse">
                            {responseData.length ? (<thead className="bg-gradient-to-r lg:text-[12px]
text-[4px] from-blue-500 to-purple-500 text-white">
                                <tr>
                                    <th className="lg:py-1 lg:px-3 border" rowSpan="3">
                                        №
                                    </th>
                                    <th className="lg:py-1 lg:px-3 border" colSpan="3">
                                        Комиссия
                                    </th>
                                    <th className="lg:py-1 lg:px-3 border" rowSpan="3">
                                        см
                                    </th>
                                    <th className="lg:py-1 lg:px-2 border">100%</th>
                                    <th className="lg:py-1 lg:px-2 border">30%</th>
                                    <th className="lg:py-1 lg:px-2 border">10%</th>
                                    <th className="lg:py-1 lg:px-2 border">60%</th>

                                    <th className="lg:py-1 lg:px-2 border" colSpan="4">
                                        итоги
                                    </th>
                                </tr>
                                <tr>
                                    <th className="lg:py-1 lg:px-2 border" colSpan="3">
                                        индекс
                                    </th>
                                    <th className="lg:py-1 lg:px-2 border">100%</th>
                                    <th className="lg:py-1 lg:px-2 border"></th>
                                    <th className="lg:py-1 lg:px-2 border">34%</th>
                                    <th className="lg:py-1 lg:px-2 border">66%</th>

                                    <th className="lg:py-1 lg:px-2 border" rowSpan="2">
                                        уход
                                    </th>
                                    <th className="lg:py-1 lg:px-2 border" rowSpan="2">
                                        приход
```

```
                                    </th>
                                    <th className="lg:py-1 lg:px-2 border" rowSpan="2">
                                        итог (+40 отправка)
                                    </th>
                                    <th className="lg:py-1 lg:px-2 border" rowSpan="2">
                                        итог индекс
                                    </th>
                                </tr>
                                <tr>
                                    <th className="lg:py-1 lg:px-2 border lg:w-[200px]">имя и город</th>
                                    <th className="lg:py-1 lg:px-2 border">админ</th>
                                    <th className="lg:py-1 lg:px-2 border">байер</th>
                                    <th className="lg:py-1 lg:px-2 border" colSpan="5"></th>
                                </tr>
                            </thead>) : ''}
                            {
                                elem.otchet.map((item, index) => {
                                    const isOdd = index % 2 === 1;

                                    return (
                                        <tbody key={item._id} className={isOdd ? " text-[4px] lg:text-[10px]"
: "text-[4px] lg:text-[10px]"}>
                                            <tr>
                                                <td className="border lg:py-1 lg:px-3" rowSpan="2">
                                                    {item.list}

                                                </td>
                                                <td className="border lg:py-1 lg:px-3" rowSpan="2">
                                                    <input
                                                        type="text"
                                                        autoComplete="off"
                                                        className={isOdd ? "bg-inherit p-1 text-center
outline-none " : 'bg-inherit p-1 w-full text-center outline-none'}
                                                        placeholder="Назовите имя и город"
                                                        value={inputValues[item._id]?.sity || ''}
                                                        onChange={(e) => {
                                                            setInputValues((prevState) => ({
                                                                ...prevState,
                                                                [item._id]: {
                                                                    ...(prevState[item._id] || {}),
                                                                    sity: e.target.value !== '' ? e.target.value : '',
                                                                },
                                                            }));
                                                        }}
                                                    />
                                                </td>
                                                <td className="border lg:py-1 lg:px-2" rowSpan="2">
                                                    <select
                                                        className={isOdd ? "bg-inherit p-1 text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                                        value={inputValues[item._id]?.admin || ''}
                                                        onChange={(e) => {
                                                            setInputValues((prevState) => ({
                                                                ...prevState,
                                                                [item._id]: {
                                                                    ...(prevState[item._id] || {}),
                                                                    admin: e.target.value !== '' ? e.target.value : '',
                                                                },
                                                            }));
                                                        }}
```

```
                                        >
                                            <option value="">Выберите админа</option>
                                            {logist.map(item =>
                                                item.curator && <option className='text-blue-900
font-semibold' value={item.curator}>{item.curator}</option>
                                            )}
                                            {filteredLogist.map(item =>
                                                item && <option value={item}>{item}</option>
                                            )}
                                        </select>
                                    </td>
                                    <td className="border lg:py-1 lg:px-2" rowSpan="2">
                                        {data ? (
                                            <div>
                                                {selectedCurator && curatorData.length > 0 ? (
                                                    <select
                                                        id="buyerSelect"
                                                        className={isOdd ? "bg-inherit outline-none
cursor-pointer w-[70px] lg:w-full" : 'lg:w-full w-[70px] outline-none bg-inherit cursor-pointer'}
                                                        value={inputValues[item._id]?.buyer}
                                                        onChange={(event) => handleBuyerChange(event,
item._id)}
                                                    >
                                                        <option value="">{inputValues[item._id]?.buyer !==
" ? inputValues[item._id]?.buyer : "Не выбрано"}</option>
                                                        <option
value={selectedCurator}>{selectedCurator}</option>
                                                        {curatorData
                                                            .filter((slotItem) => slotItem.buyer.trim() !== "" &&
slotItem.status == "2")
                                                            .map((slotItem, index) => (
                                                                <option key={index} value={slotItem.buyer}>
                                                                    {slotItem.buyer}
                                                                </option>
                                                            ))}
                                                    </select>
                                                ) : (
                                                    <div>{inputValues[item._id]?.buyer !== " ?
inputValues[item._id]?.buyer : 'Список пуст'}</div>
                                                )}
                                            </div>
                                        ) : (
                                            <div>Загрузка данных...</div>
                                        )}
                                    </td>
                                    <td className="border lg:py-1 lg:px-2" rowSpan="2">
                                        <select
                                            className={isOdd ? "bg-inherit outline-none
cursor-pointer" : 'outline-none bg-inherit cursor-pointer'}
                                            value={inputValues[item._id]?.sm || 1}
                                            onChange={(e) => {
                                                const selectedValue = parseInt(e.target.value);
                                                const percent100 =
inputValues[item._id]?.comPersent100
                                                let comPersent2, comPersent3, comPersent4;
                                                const uhod = parseFloat(inputValues[item._id]?.uhod) ||
0;
                                                switch (selectedValue) {
                                                    case 1:
```

```
                                                comPersent2 = percent100 ? (percent100 *
0.3).toFixed(0) : '';
                                                comPersent3 = percent100 ? (percent100 *
(percent100 <= 1000 ? 0.15 : 0.1)).toFixed(0) : '';
                                                comPersent4 = percent100 ? (percent100 *
(percent100 <= 1000 ? 0.55 : 0.6) + 40).toFixed(0) : '';
                                                break;
                                            case 2:
                                                comPersent2 = percent100 ? (percent100 *
0.5).toFixed(0) : '';
                                                comPersent3 = percent100 ? (percent100 *
(percent100 <= 1000 ? 0.15 : 0.1)).toFixed(0) : '';
                                                comPersent4 = percent100 ? (percent100 *
(percent100 <= 1000 ? 0.35 : 0.4) + 40).toFixed(0) : '';
                                                break;
                                            case 3:
                                                comPersent2 = percent100 ? (percent100 *
0.4).toFixed(0) : '';
                                                comPersent3 = percent100 ? (percent100 *
0.1).toFixed(0) : '';
                                                comPersent4 = percent100 ? (percent100 * 0.5 +
40).toFixed(0) : '';
                                                break;
                                            case 4:
                                                comPersent2 = percent100 ? (percent100 *
0.4).toFixed(0) : '';
                                                comPersent3 = percent100 ? (percent100 *
0.1).toFixed(0) : '';
                                                comPersent4 = percent100 ? (percent100 * 0.5 +
40).toFixed(0) : '';
                                                break;
                                            case 5:
                                                comPersent2 = percent100 ? (percent100 *
0.4).toFixed(0) : '';
                                                comPersent3 = percent100 ? (percent100 *
0.1).toFixed(0) : '';
                                                comPersent4 = percent100 ? (percent100 * 0.5 +
40).toFixed(0) : '';
                                                break;
                                            default:
                                                comPersent2 = '';
                                                comPersent3 = '';
                                                comPersent4 = '';
                                                break;
                                        }

                                        setInputValues((prevState) => ({
                                            ...prevState,
                                            [item._id]: {
                                                ...(prevState[item._id] || {}),
                                                sm: selectedValue,
                                                comPersent100: percent100,
                                                comPersent2,
                                                comPersent3,
                                                comPersent4,
                                                itog: comPersent4 ? parseFloat(comPersent4) -
parseFloat(uhod) : ''
                                            },
                                        }));
                                    }}
```

```
                                        >
                                            <option value={1}>1</option>
                                            <option value={2}>2</option>
                                            <option value={3}>x% байер</option>
                                            <option value={4}>x% логист</option>
                                            <option value={5}>x% онлайн</option>
                                        </select>
                                    </td>
                                    <td className="border lg:py-1 lg:px-2">
                                        <input
                                            autoComplete="off"
                                            type="text"
                                            className={isOdd ? "bg-inherit w-full text-center
outline-none" : 'w-full bg-inherit text-center outline-none'}
                                            value={inputValues[item._id]?.comPersent100 || ''}
                                            onChange={(e) => {
                                                const comPersent100 = e.target.value;
                                                let comPersent2, comPersent3, comPersent4;
                                                switch (inputValues[item._id]?.sm) {
                                                    case 1:
                                                        comPersent2 = comPersent100 ? (comPersent100
* 0.3).toFixed(0) : '';
                                                        comPersent3 = comPersent100 ? (comPersent100
* (comPersent100 <= 1000 ? 0.15 : 0.1)).toFixed(0) : '';
                                                        comPersent4 = comPersent100 ? (comPersent100
* (comPersent100 <= 1000 ? 0.55 : 0.6) + (comPersent100 !== '' ? 40 : 0)).toFixed(0) : '';
                                                        break;
                                                    case 2:
                                                        comPersent2 = comPersent100 ? (comPersent100
* 0.5).toFixed(0) : '';
                                                        comPersent3 = comPersent100 ? (comPersent100
* (comPersent100 <= 1000 ? 0.15 : 0.1)).toFixed(0) : '';
                                                        comPersent4 = comPersent100 ? (comPersent100
* (comPersent100 <= 1000 ? 0.35 : 0.4) + (comPersent100 !== '' ? 40 : 0)).toFixed(0) : '';
                                                        break;
                                                    case 3:
                                                        comPersent2 = comPersent100 ? (comPersent100
* 0.4).toFixed(0) : '';
                                                        comPersent3 = comPersent100 ? (comPersent100
* 0.1).toFixed(0) : '';
                                                        comPersent4 = comPersent100 ? (comPersent100
* 0.5 + (comPersent100 !== '' ? 40 : 0)).toFixed(0) : '';
                                                        break;
                                                    case 4:
                                                        comPersent2 = comPersent100 ? (comPersent100
* 0.4).toFixed(0) : '';
                                                        comPersent3 = comPersent100 ? (comPersent100
* 0.1).toFixed(0) : '';
                                                        comPersent4 = comPersent100 ? (comPersent100
* 0.5 + (comPersent100 !== '' ? 40 : 0)).toFixed(0) : '';
                                                        break;
                                                    case 5:
                                                        comPersent2 = comPersent100 ? (comPersent100
* 0.4).toFixed(0) : '';
                                                        comPersent3 = comPersent100 ? (comPersent100
* 0.1).toFixed(0) : '';
                                                        comPersent4 = comPersent100 ? (comPersent100
* 0.5 + (comPersent100 !== '' ? 40 : 0)).toFixed(0) : '';
                                                        break;
                                                    default:
```

```
                                                comPersent2 = '';
                                                comPersent3 = '';
                                                comPersent4 = '';
                                                break;
                                            }

                                            setInputValues((prevState) => ({
                                                ...prevState,
                                                [item._id]: {
                                                    ...(prevState[item._id] || {}),
                                                    comPersent100: e.target.value !== '' ?
e.target.value : '',
                                                    comPersent2,
                                                    comPersent3,
                                                    comPersent4,
                                                    itog: e.target.value === '' ? '' : comPersent4,
                                                },
                                            }));
                                        }}
                                    />
                                </td>
                                <td className="border lg:py-1
lg:px-2">{inputValues[item._id]?.comPersent2 || ''}</td>
                                <td className="border lg:py-1
lg:px-2">{inputValues[item._id]?.comPersent3 || ''}</td>
                                <td className="border lg:py-1
lg:px-2">{inputValues[item._id]?.comPersent4 || ''}</td>

                                <td className="border lg:py-1 lg:px-2" rowSpan="2">
                                    <input
                                        autoComplete="off"
                                        type="text"
                                        className={isOdd ? "bg-inherit w-full text-center
outline-none" : 'bg-inherit w-full text-center outline-none'}
                                        value={inputValues[item._id]?.uhod || ''}
                                        onChange={(e) => {
                                            const uhod = e.target.value;
                                            const comPersent4 =
parseFloat(inputValues[item._id]?.comPersent4) || 0;
                                            setInputValues((prevState) => ({
                                                ...prevState,
                                                [item._id]: {
                                                    ...(prevState[item._id] || {}),
                                                    uhod: uhod,
                                                    itog: (!uhod || isNaN(uhod)) ?
comPersent4.toString() : (comPersent4 - parseFloat(uhod)).toString()
                                                },
                                            }));
                                        }}
                                    />
                                </td>
                                <td className="border lg:py-1 lg:px-2" rowSpan="2">
                                    <input
                                        autoComplete="off"
                                        type="text"
                                        className={isOdd ? "bg-inherit w-full text-center
outline-none" : 'bg-inherit w-full text-center outline-none'}
                                        value={inputValues[item._id]?.prihod || ''}
                                        onChange={(e) => {
                                            const prihod = e.target.value;
```

```
                                        setInputValues((prevState) => ({
                                            ...prevState,
                                            [item._id]: {
                                                ...(prevState[item._id] || {}),
                                                prihod: prihod,
                                            },
                                        }));
                                    }}
                                />
                            </td>

                            <td className="border lg:py-1 lg:px-2" rowSpan="2">
                                {inputValues[item._id]?.itog || ''}
                            </td>
                            <td className="border lg:py-1 lg:px-2" rowSpan="2">
                                {inputValues[item._id]?.itogIndex || ''}
                            </td>
                        </tr>
                        <tr>
                            <td className="border lg:py-1 lg:px-2">
                                <input
                                    autoComplete="off"
                                    type="text"
                                    className={isOdd ? "bg-inherit w-full text-center
outline-none" : 'bg-inherit w-full text-center outline-none'}
                                    value={inputValues[item._id]?.indexPersent100 || ''}
                                    onChange={(e) => {

                                        const indexPersent100 = e.target.value;
                                        // const indexPersent2 = (indexPersent100 *
0.34).toFixed(0);
                                        const indexPersent3 = (indexPersent100 *
0.34).toFixed(0);
                                        const indexPersent4 = (indexPersent100 *
0.66).toFixed(0)

                                        setInputValues((prevState) => ({
                                            ...prevState,
                                            [item._id]: {
                                                ...(prevState[item._id] || {}),
                                                indexPersent100: e.target.value !== '' ?
e.target.value : '',
                                                // indexPersent2,
                                                indexPersent3,
                                                indexPersent4,
                                                itogIndex: indexPersent4,

                                            },
                                        }));
                                    }}
                                />
                            </td>
                            <td className="border lg:py-1 lg:px-2">
                                {/* <input
                                    type="text"
                                    className={isOdd ? "bg-inherit w-full text-center
outline-none" : 'bg-inherit w-full text-center outline-none'}
                                    value={inputValues[item._id]?.indexPersent2 || ''}
                                    onChange={(e) => {
                                        setInputValues((prevState) => ({
```

```
                                                ...prevState,
                                                [item._id]: {
                                                    ...(prevState[item._id] || {}),
                                                    indexPersent2: e.target.value !== '' ? e.target.value
: '',
                                                },
                                            }));
                                        }}
                                    /> */}
                                </td>
                                <td className="border lg:py-1 lg:px-2">
                                    <input
                                        type="text"
                                        className={isOdd ? "bg-inherit w-full text-center
outline-none" : 'bg-inherit w-full text-center outline-none'}
                                        value={inputValues[item._id]?.indexPersent3 || ''}
                                        onChange={(e) => {
                                            setInputValues((prevState) => ({
                                                ...prevState,
                                                [item._id]: {
                                                    ...(prevState[item._id] || {}),
                                                    indexPersent3: e.target.value !== '' ? e.target.value
: '',
                                                },
                                            }));
                                        }}
                                    />
                                </td>
                                <td className="border lg:py-1 lg:px-2">
                                    <input
                                        type="text"
                                        className={isOdd ? "bg-inherit w-full text-center
outline-none" : 'bg-inherit w-full text-center outline-none'}
                                        value={inputValues[item._id]?.indexPersent4 || ''}
                                        onChange={(e) => {
                                            setInputValues((prevState) => ({
                                                ...prevState,
                                                [item._id]: {
                                                    ...(prevState[item._id] || {}),
                                                    indexPersent4: e.target.value !== '' ? e.target.value
: '',

                                                    itogIndex: e.target.value !== '' ? e.target.value : ''
                                                },
                                            }));
                                        }}
                                    />
                                </td>
                            </tr>
                        </tbody>
                    );
                },)
            }
        </table>
        {responseData.length ? (loading4 ? (<button
            onClick={(e) => addSoloTable(elem._id)}
            className='lg:ml-3 mt-5 mb-5 ml-7 bg-gradient-to-r from-blue-500
to-purple-500 hover:bg-blue-600 text-white lg:py-2 lg:px-6 p-1 rounded-lg shadow-lg transition-all
duration-300 ease-in-out text-[10px] transform hover:scale-105 focus:outline-none'
        >добавить слот
        </button>) : (<button
```

```
                                    className='lg:ml-3 mt-5 mb-5 bg-gradient-to-r from-blue-500 to-purple-500
hover:bg-blue-600 text-white lg:py-2 lg:px-6 p-1 rounded-lg shadow-lg transition-all duration-300
ease-in-out text-[10px] transform hover:scale-105 focus:outline-none'
                                ><LoadAnimate /></button>)) : ''}
                            </div>
                        )
                    })}
                    {responseData.length ? (<button
                        className="lg:fixed lg:right-3 lg:bottom-1 mt-5 mb-5 bg-gradient-to-r from-blue-500
to-purple-500 hover:bg-blue-600 text-white py-2 px-6 rounded-lg shadow-lg transition-all duration-300
ease-in-out text-[10px] transform hover:scale-105 focus:outline-none"
                        onClick={handleSubmit}
                    >
                        {loading ? 'Сохранить' : <LoadAnimate />}
                    </button>) : ''}
                    <table className="mx-auto lg:min-w-full w-[300px]  lg:text-[12px] text-[5px] text-center
border">
                        {responseData.length ? (<thead className="bg-gradient-to-r from-blue-500
to-purple-500 text-white">
                            <tr>
                                <td className='border' >росходы</td>
                                <td className='border' >сумма</td>
                            </tr>
                        </thead>) : ''}
                        {
                            responseData.map(elem => {
                                return elem.itog.map((item, index) => {
                                    const isOdd = index % 2 === 1;
                                    return (
                                        <tbody key={item._id} className='lg:text-[12px] text-[5px]'>
                                            <tr>
                                                <td className='border'>
                                                    <input
                                                        type="text"
                                                        autoComplete="off"
                                                        className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                                        value={inputValues2[item._id]?.ros1 || ''}
                                                        onChange={(e) => {
                                                            setInputValues2((prevState) => ({
                                                                ...prevState,
                                                                [item._id]: {
                                                                    ...(prevState[item._id] || {}),
                                                                    ros1: e.target.value !== '' ? e.target.value : '',
                                                                },
                                                            }));
                                                        }}
                                                    />
                                                </td>
                                                <td className='border '>
                                                    <input
                                                        type="text"
                                                        autoComplete="off"
                                                        className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                                        value={inputValues2[item._id]?.sum1 || ''}
                                                        onChange={(e) => {
                                                            setInputValues2((prevState) => ({
                                                                ...prevState,
                                                                [item._id]: {
```

```
                                        ...(prevState[item._id] || {}),
                                        sum1: e.target.value !== '' ? e.target.value : '',
                                    },
                                }));
                            }}
                        />
                    </td>
                </tr>
                <tr>
                    <td className='border'>
                        <input
                            type="text"
                            autoComplete="off"
                            className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                            value={inputValues2[item._id]?.ros2 || ''}
                            onChange={(e) => {
                                setInputValues2((prevState) => ({
                                    ...prevState,
                                    [item._id]: {
                                        ...(prevState[item._id] || {}),
                                        ros2: e.target.value !== '' ? e.target.value : '',
                                    },
                                }));
                            }}
                        />
                    </td>
                    <td className='border'>
                        <input
                            type="text"
                            autoComplete="off"
                            className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                            value={inputValues2[item._id]?.sum2 || ''}
                            onChange={(e) => {
                                setInputValues2((prevState) => ({
                                    ...prevState,
                                    [item._id]: {
                                        ...(prevState[item._id] || {}),
                                        sum2: e.target.value !== '' ? e.target.value : '',
                                    },
                                }));
                            }}
                        />
                    </td>
                </tr>
                <tr>
                    <td className='border '>
                        <input
                            type="text"
                            autoComplete="off"
                            className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                            value={inputValues2[item._id]?.ros3 || ''}
                            onChange={(e) => {
                                setInputValues2((prevState) => ({
                                    ...prevState,
                                    [item._id]: {
                                        ...(prevState[item._id] || {}),
                                        ros3: e.target.value !== '' ? e.target.value : '',
```

```
                                        },
                                    }));
                                }}
                            />
                        </td>
                        <td className='border'>
                            <input
                                type="text"
                                autoComplete="off"
                                className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                value={inputValues2[item._id]?.sum3 || ''}
                                onChange={(e) => {
                                    setInputValues2((prevState) => ({
                                        ...prevState,
                                        [item._id]: {
                                            ...(prevState[item._id] || {}),
                                            sum3: e.target.value !== '' ? e.target.value : '',
                                        },
                                    }));
                                }}
                            />
                        </td>
                    </tr>
                    <tr>
                        <td className='border'>
                            <input
                                type="text"
                                autoComplete="off"
                                className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                value={inputValues2[item._id]?.ros4 || ''}
                                onChange={(e) => {
                                    setInputValues2((prevState) => ({
                                        ...prevState,
                                        [item._id]: {
                                            ...(prevState[item._id] || {}),
                                            ros4: e.target.value !== '' ? e.target.value : '',
                                        },
                                    }));
                                }}
                            />
                        </td>
                        <td className='border'>
                            <input
                                type="text"
                                autoComplete="off"
                                className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                value={inputValues2[item._id]?.sum4 || ''}
                                onChange={(e) => {
                                    setInputValues2((prevState) => ({
                                        ...prevState,
                                        [item._id]: {
                                            ...(prevState[item._id] || {}),
                                            sum4: e.target.value !== '' ? e.target.value : '',
                                        },
                                    }));
                                }}
                            />
```

```
                                    </td>
                                </tr>
                                <tr>
                                    <td className='border '>
                                        <input
                                            type="text"
                                            autoComplete="off"
                                            className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                            value={inputValues2[item._id]?.ros5 || ''}
                                            onChange={(e) => {
                                                setInputValues2((prevState) => ({
                                                    ...prevState,
                                                    [item._id]: {
                                                        ...(prevState[item._id] || {}),
                                                        ros5: e.target.value !== '' ? e.target.value : '',
                                                    },
                                                }));
                                            }}
                                        />
                                    </td>
                                    <td className='border '>
                                        <input
                                            type="text"
                                            autoComplete="off"
                                            className={isOdd ? "bg-inherit p-1 w-full text-center
outline-none" : 'p-1 bg-inherit w-full text-center outline-none'}
                                            value={inputValues2[item._id]?.sum5 || ''}
                                            onChange={(e) => {
                                                setInputValues2((prevState) => ({
                                                    ...prevState,
                                                    [item._id]: {
                                                        ...(prevState[item._id] || {}),
                                                        sum5: e.target.value !== '' ? e.target.value : '',
                                                    },
                                                }));
                                            }}
                                        />
                                    </td>
                                </tr>
                                <tr>
                                    <td colSpan='2' className='bg-gradient-to-r from-blue-500
to-purple-500 text-white'>итоги</td>
                                </tr>
                                <tr>
                                    <td className='border '>индекс:</td>
                                    <td className='border '>{item.allItogIndex}</td>
                                </tr>
                                <tr>
                                    <td className='border '>касса:</td>
                                    <td className='border'>{item.allItog}</td>
                                </tr>
                                <tr>
                                    <td className='border '>приход:</td>
                                    <td className='border '>{item.allItogPrihod}</td>
                                </tr>
                                <tr>
                                    <td className='border'>уход:</td>
                                    <td className='border'>{item.allItogUhod}</td>
                                </tr>
```

```
                                    <tr>
                                        <td className='border '>кэшбек 10%:</td>
                                        <td className='border '>{item.raznica}</td>
                                    </tr>
                                    <tr>
                                        <td colSpan='2' className='bg-gradient-to-r from-blue-500
to-purple-500 font-semibold text-white'>общий итог: {item.itogs} </td>
                                    </tr>
                                </tbody>
                            )
                        })
                    })
                }
            </table>

            {responseData.length ? (<input
                type="date"
                value={dateValue}
                onChange={(event) => setDateValue(event.target.value)}
                className='mt-5 text-sm flex items-center bg-base-100 border p-3
border-indigo-600 mb-10 hover:opacity-80 outline-none'
            />) : ''}

            {
                responseData.length ? (loading3 ?
                    <button
                        onClick={handleSend}
                        className='mt-5 mb-5 bg-gradient-to-r from-blue-500 to-purple-500
hover:bg-blue-600 text-white py-2 px-6 rounded-lg shadow-lg transition-all duration-300 ease-in-out
text-[10px] transform hover:scale-105 focus:outline-none'
                    >Отправить отчет</button> : <button
                        className='mt-5 mb-5 bg-gradient-to-r from-blue-500 to-purple-500
hover:bg-blue-600 text-white py-2 px-6 rounded-lg shadow-lg transition-all duration-300 ease-in-out
text-[10px] transform hover:scale-105 focus:outline-none'
                    ><LoadAnimate /></button>) : ''
            }

            {
                responseData.length ? '' : (loading2 ? <button className='mt-5 mb-5
bg-gradient-to-r from-blue-500 to-purple-500 hover:bg-blue-600 text-white py-2 px-6 rounded-lg
shadow-lg transition-all duration-300 ease-in-out text-[10px] transform hover:scale-105
focus:outline-none'
                    onClick={createTable}> создать отчет </button> : <button className='mt-5 mb-5
bg-gradient-to-r from-blue-500 to-purple-500 hover:bg-blue-600 text-white py-2 px-6 rounded-lg
shadow-lg transition-all duration-300 ease-in-out text-[10px] transform hover:scale-105
focus:outline-none'
                    ><LoadAnimate /> </button>)

            }
        </div>
    </div>) : (
        <div className='w-full gap-2 flex-col flex justify-center items-center'><LoadAnimate
/>загружаем данных, просим подождать</div>
    )}
</div >
)
} else {
    return <Navigate to="/access" replace />;
}
}
```