

ChatGPT

Dies ist die Kopie eines Gesprächs zwischen ChatGPT und einem anonymen Benutzer.



Why is the table data not showing the updates after the fetching??

```
import React, { useState, useEffect } from "react";
import { useParams } from 'react-router-dom';
import axiosInstance from '../api/AxiosInstance';

const Detailspace = () => {

  const {shipId, containerId} = useParams();

  const [container, setContainer] = useState(null);
  const [combinedSerialNumber, setCombinedSerialNumber] = useState("Loading...");
  const [notes, setNotes] = useState("");

  const [temperature, setTemperature] = useState("");
  const [pressure, setPressure] = useState("");
  const [humidity, setHumidity] = useState("");
  const [vibration, setVibration] = useState("");
  const [altitude, setAltitude] = useState("");
  const [latitude, setLatitude] = useState("");
  const [longitude, setLongitude] = useState("");

  const [saved, setSaved] = useState(false);
  const [notSaved, setNotSaved] = useState(false); // If the save fails this is used

  const tableData = [
    { environment: "Temperature", value: "°C", level: "", problem: "" },
    { environment: "Pressure", value: "Pa", level: "Critical", problem: "High Pressure" },
    { environment: "Humidity", value: "%", level: "", problem: "" },
    { environment: "Vibration", value: "m/s2", level: "", problem: "" },
    { environment: "Altitude", value: "m", level: "", problem: "" },
    { environment: "Latitude", value: "DD", level: "", problem: "" },
    { environment: "Longitude", value: "DD", level: "", problem: "" },
  ];

  const updateTableData = (newValue, sensor) => {
    return tableData.map(item => {
      switch(sensor) {
        case "temperature":
          if (item.environment === "Temperature") {
            return { ...item, value: newValue };
          }
          break;
      }
    });
  }
}
```

```

case "air_pressure":
  if (item.environment === "Pressure") {
    return { ...item, value: newValue };
  }
  break;
case "humidity":
  if (item.environment === "Humidity") {
    return { ...item, value: newValue };
  }
  break;
case "vibration":
  if (item.environment === "Vibration") {
    return { ...item, value: newValue };
  }
  break;
case "altitude":
  if (item.environment === "Altitude") {
    return { ...item, value: newValue };
  }
  break;
case "latitude":
  if (item.environment === "Latitude") {
    return { ...item, value: newValue };
  }
  break;
case "longitude":
  if (item.environment === "Longitude") {
    return { ...item, value: newValue };
  }
  break;
default:
  return item;
}
return item; // Return the item unchanged if no conditions match
});
};

const handleNotesChange = (event) => {
  setNotes(event.target.value);
};

const handleBlur = async () => {
  if (!container) return;

  try {
    const accessToken = localStorage.getItem("accessToken");
    const updatedContainer = { ...container, notes };

    const updateResponse = await axiosInstance.put(`/rest/container/${containerId}`,
updatedContainer, {
  headers: {

```

```

    authorization: `Bearer ${accessToken}`,
  },
});

if (updateResponse.status === 204) {
  setSaved(true);
  setTimeout(() => { setSaved(false); }, 2000);
} else {
  setNotSaved(true);
  setTimeout(() => { setNotSaved(false); }, 2000);
}

} catch (error) {
  console.error("Failed to save notes:", error.message);
}
};

// Fetch Data
useEffect(() => {
  const fetchContainerData = async () => {
    try {
      const accessToken = localStorage.getItem("accessToken");
      let sensor = "";
      // Temperature
      const temperatureResponse = await
axiosInstance.get(`/rest/sensor/temperature/${shipId}/${containerId}/latest`, {
  headers: {
    'authorization': `Bearer ${accessToken}`,
  },
});
      const temperatureValue = temperatureResponse.data.temperature[0]?.value;
      sensor = temperatureResponse.data.temperature[0]?.sensor;
      updateTableData(temperatureValue, sensor);

      // Air Pressure
      const airPressureResponse = await
axiosInstance.get(`/rest/sensor/air_pressure/${shipId}/${containerId}/latest`, {
  headers: {
    'authorization': `Bearer ${accessToken}`,
  },
});
      const airPressureValue = airPressureResponse.data.air_pressure[0]?.value;
      setPressure(airPressureValue);

      // Humidity
      const humidityResponse = await
axiosInstance.get(`/rest/sensor/humidity/${shipId}/${containerId}/latest`, {
  headers: {
    'authorization': `Bearer ${accessToken}`,
  },
});
    }
  };
  fetchContainerData();
}, []);

```

```

const humidityValue = humidityResponse.data.humidity[0]?.value;
setHumidity(humidityValue);

// Air Pressure
const vibrationResponse = await
axiosInstance.get(`/rest/sensor/vibration/${shipId}/${containerId}/latest`, {
  headers: {
    'authorization': `Bearer ${accessToken}`,
  },
});
const vibrationValue = vibrationResponse.data.vibration[0]?.value;
setVibration(vibrationValue);

// Altitude
const altitudeResponse = await
axiosInstance.get(`/rest/sensor/altitude/${shipId}/${containerId}/latest`, {
  headers: {
    'authorization': `Bearer ${accessToken}`,
  },
});
const altitudeValue = altitudeResponse.data.altitude[0]?.value;
setAltitude(altitudeValue);

// Latitude
const latitudeResponse = await
axiosInstance.get(`/rest/sensor/latitude/${shipId}/${containerId}/latest`, {
  headers: {
    'authorization': `Bearer ${accessToken}`,
  },
});
const latitudeValue = latitudeResponse.data.latitude[0]?.value;
setLatitude(latitudeValue);

// Longitude
const longitudeResponse = await
axiosInstance.get(`/rest/sensor/longitude/${shipId}/${containerId}/latest`, {
  headers: {
    'authorization': `Bearer ${accessToken}`,
  },
});
const longitudeValue = longitudeResponse.data.longitude[0]?.value;
setLongitude(longitudeValue);

updateTableDate();

} catch (error) {
  console.error('Error fetching data:', error);
}
};
fetchContainerData();
}, [containerId, shipId])

```

```
// Fetch combined Container Serial Number when id changes
useEffect(() => {
  const fetchSerialNumberOfContainer = async () => {
    try {
      const accessToken = localStorage.getItem("accessToken");
      const containerSerialNumberResponse = await
axiosInstance.get(`/rest/container/${containerId}/serial-number`, {
        headers: {
          authorization: `Bearer ${accessToken}`,
        },
      });

      const fetchedSerialNumber = containerSerialNumberResponse.data.serial_number;
      setCombinedSerialNumber(fetchedSerialNumber);
    } catch (error) {
      console.error("Failed to fetch Serial Numbers of containers:", error.message);
    }
  };
  fetchSerialNumberOfContainer();
}, [containerId])

// Fetch entire Container by ID when the id changes
useEffect(() => {
  const fetchContainerData = async () => {
    try {
      const accessToken = localStorage.getItem('accessToken');

      const containerResponse = await axiosInstance.get(`/rest/container/${containerId}`, {
        headers: {
          authorization: `Bearer ${accessToken}`,
        },
      });
      const fetchedContainer = containerResponse.data.container[0];
      setContainer(fetchedContainer);
      setNotes(fetchedContainer.notes);

    } catch (error) {
      console.error('Failed to fetch container data:', error.message);
    }
  };

  fetchContainerData();
}, [containerId]);

return (
  <div className="flex-grow flex justify-center items-center pt-5 pb-5 pl-24 pr-24 gap-10">
    <div className="h-5/6 w-3/4 p-10">
      <h1 className="text-2xl font-bold border-2 border-black p-1 mb-2">Container
{combinedSerialNumber}</h1>
      <table className="w-full border-collapse border-2 border-black">
```

```

<thead>
  <tr>
    <th className="border border-black px-4 py-2">Environment Data</th>
    <th className="border border-black px-4 py-2">Value</th>
    <th className="border border-black px-4 py-2">Level</th>
    <th className="border border-black px-4 py-2">Problem</th>
  </tr>
</thead>
<tbody>
  {tableData.map((row, index) => (
    <tr key={index}>
      <td className="border border-black px-4 py-2">{row.environment}</td>
      <td className="border border-black px-4 py-2">{row.value}</td>
      <td className="border border-black px-4 py-2">{row.level}</td>
      <td className="border border-black px-4 py-2">{row.problem}</td>
    </tr>
  ))}
</tbody>
</table>
</div>
<div className="h-5/6 w-1/4 pt-10 pr-10">
  <textarea className="resize-none overflow-auto text-base w-full border-2 border-dashed
border-gray-400 bg-gray-100 text-gray-700 h-[385px] p-2"
    name="notes"
    id="notes"
    placeholder="Notes..."
    value={notes | | ""}
    onChange={handleNotesChange}
    onBlur={handleBlur}
  ></textarea>
  {saved && (
    <div className="text-green-400 mt-1">Notes saved!</div>
  )}
  {notSaved && (
    <div className="text-red-400 mt-1">Failed to save notes!</div>
  )}
</div>
</div>
);
};

/*VALUES ('Air Pressure', 'Pa'),
('Humidity', '%'),
('Temperature', '°C'),
('Vibration', 'm/s²'),
('Altitude', 'm'),
('Latitude', 'DD'),
('Longitude', 'DD');*/

export default Detailspace;

```



The issue you're facing with table updates likely stems from how the `tableData` is managed. The `tableData` array is not being updated correctly, as you're directly mutating it inside the `updateTableData` function without triggering a re-render.

In React, you should avoid directly mutating state, because it doesn't trigger a re-render. Instead, you should create a new copy of the array with the updated values and then use `setState` to trigger a re-render.

Here's a suggestion on how to fix it:

1. **Store the table data in state:** Instead of using a regular array for `tableData`, store it in a state variable so you can trigger a re-render when it updates.
2. **Update the table data properly:** When you want to update the table data, use the state setter to update the table data.

Here's how you can modify your code:

js

```
const [tableData, setTableData] = useState([
  { environment: "Temperature", value: "°C", level: "", problem: "" },
  { environment: "Pressure", value: "Pa", level: "Critical", problem: "High Pressure" },
  { environment: "Humidity", value: "%", level: "", problem: "" },
  { environment: "Vibration", value: "m/s2", level: "", problem: "" },
  { environment: "Altitude", value: "m", level: "", problem: "" },
  { environment: "Latitude", value: "DD", level: "", problem: "" },
  { environment: "Longitude", value: "DD", level: "", problem: "" },
]);

const updateTableData = (newValue, sensor) => {
  setTableData((prevData) => {
    prevData.map(item => {
      switch(sensor) {
        case "temperature":
          if (item.environment === "Temperature") {
            return { ...item, value: newValue };
          }
          break;
        case "air_pressure":
          if (item.environment === "Pressure") {
            return { ...item, value: newValue };
          }
          break;
        case "humidity":
          if (item.environment === "Humidity") {
            return { ...item, value: newValue };
          }
          break;
        case "vibration":
          if (item.environment === "Vibration") {
            return { ...item, value: newValue };
          }
          break;
        case "altitude":
          if (item.environment === "Altitude") {
            return { ...item, value: newValue };
          }
          break;
        case "latitude":
          if (item.environment === "Latitude") {
            return { ...item, value: newValue };
          }
          break;
      }
    })
  });
}
```

```

        case "longitude":
            if (item.environment === "Longitude") {
                return { ...item, value: newValue };
            }
            break;
        default:
            return item;
    }
    return item; // Return the item unchanged if no conditions match
})
);
};

```

With this change:

- The `tableData` is now stored in a state variable, so updates will trigger re-renders automatically.
- `updateTableData` now uses `setTableData` to properly update the state.

Additionally, ensure that `updateTableData` is called properly in your `useEffect` after fetching data:

```

js

useEffect(() => {
    const fetchContainerData = async () => {
        try {
            const accessToken = localStorage.getItem("accessToken");

            // Temperature
            const temperatureResponse = await axiosInstance.get(`/rest/sensor/temperature/${shipId}/${containerId}/latest`, {
                headers: {
                    'authorization': `Bearer ${accessToken}`,
                },
            });
            const temperatureValue = temperatureResponse.data.temperature[0]?.value;
            const temperatureSensor = temperatureResponse.data.temperature[0]?.sensor;
            updateTableData(temperatureValue, temperatureSensor);

            // Repeat for other sensors...

        } catch (error) {
            console.error('Error fetching data:', error);
        }
    };
    fetchContainerData();
}, [containerId, shipId]);

```

This should ensure that the data updates properly in the table after fetching.

 Ab hier sind Nachrichten nur für dich sichtbar