Frontend With React

עבודה על פרוייקט צד לקוח

TomerBu

שליפת מידע מוגן עם JWT

לטובת השלמה אוטומטית - נגדיר טיפוס למוצר:

@types.d.ts

```
export type ProductType = {
  id: number;
  name: string;
  price: number;
  category: string;
  description: string;
  imageUrl: string;
};
```

תזכורת לגבי useEffect:

קוד שרץ רק פעם אחת עם הטעינה הראשונית של הקומפוננטה:

```
import React, { useEffect, useState } from 'react';

function MyComponent() {
  const [data, setData] = useState(null);

  useEffect(() => {
    // לנתונים fetch נשלח בקשת
    fetch('https://jsonplaceholder.typicode.com/posts/1')
      .then((response) => response.json())
      .then((json) => setData(json))
      .catch((error) => console.error('Error fetching data:', error));
  }, []); // אפקט זה יפעל רק פעם אחת

  return (
    <div>
      <h1>Fetched Data:</h1>
      {data ? <pre>{JSON.stringify(data, null, 2)}</pre> : <p>Loading...</p>}
    </div>
  );
}

export default MyComponent;
```

תזכורת לגבי useEffect:

קוד שיופעל אחרי כל שינוי שנגדיר לו:

```
import React, { useEffect, useState } from 'react';

function MyComponent() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    console.log('Count changed:', count);
  }, [count]); // משתנה count אפקט זה יופעל בכל פעם שמשתנה

  return (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Increment Count</button>
    </div>
  );
}

export default MyComponent;
```

הקוד בדוגמא יופעל
בכל פעם שהערך של count
ישתנה - כי שמנו את הcount
במערך של הdependencies

שליפת מידע מוגן עם JWT

```
import { useState, useEffect } from "react";
import { ProductType } from "../@types";
import { getProducts } from "../services/products-service";

const useProducts  = () => {
  const [products, setProducts] = useState<ProductType[]>([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);

  useEffect(() => {
    setLoading(true);
    getProducts()
      .then((response) => {
        setProducts(response.data);
      })
      .catch((error) => {
        setError(error);
      })
      .finally(() => {
        setLoading(false);
      });
  }, []);

  return { products, loading, error };
};

export default useProducts;
```

נגדיר custom hook לשליפת המוצרים כולל loading, error, products

כך אנו מפרידים את הלוגיקה של fetch מהקומפוננטה
- תפקיד הקומפוננטה - להציג ממשק משתמש
תפקיד הhook - לשלוף פרטי מוצרים בhttp

שליפת מידע מוגו עם JWT

```
import { Link } from "react-router-dom";
import useProducts from "../hooks/useProducts";

const Products = () => {
  const { products, loading, error } = useProducts();

  return (
    <div>
      <h1 className="text-center mb-3 text-3xl">Products</h1>

      {error && <p>Error: {error.message ?? "something went wrong"}</p>}
      {loading && <p>Loading...</p>}
      {!loading && !error && (
        <ul>
          {products &&
            products.map((p) => (
              <li className="flex p-4 shadow-lg" key={p.id}>
                <Link to={`${p.id}`}>
                  <h2>{p.name}</h2>
                </Link>
              </li>
            ))}
        </ul>
      )}
    </div>
  );
};

export default Products;
```

הקומפוננטה Products שמשתמשת בhook שהגדרנו useProducts:

אפשר לראות שהשימוש בhook מסיר מהקומפוננטה את האחריות של שליפת מידע והקומפוננטה אכן עוסקת בהצגת מידע למשתמש

הגדרות לaxios עבור כלל האפליקציה:

כל האפליקציה שלנו עובדת מול אותו API
ולכן נכון להגדיר במקום אחד מרכזי את הכתובת של הAPI
וכן הגדרות נוספות לבקשות http שאנחנו רוצים לכלול בכל האפליקציה.

לדוגמא - נרצה לשלוח JWT עם כל בקשה
נרצה להאזין לבקשות היוצאות

ולהפעיל מתודה בזמן שהבקשה מצליחה או נכשלת

```
import axios, { AxiosRequestConfig } from "axios";

const baseUrl = import.meta.env.VITE_BASE_URL;

const client = axios.create({
  baseURL: baseUrl,
  headers: {
    Authorization: `Bearer ${localStorage.getItem("token")}`,
  },
});

const onSuccess = (response) => {
  console.debug("Request Successful!", response);
  return response;
};

const onError = (error) => {
  console.error("Request Failed:", error);
  return error;
};

const request = (options: AxiosRequestConfig) => {
   return client(options).then(onSuccess).catch(onError);
}

//request({ url: "/products", method: "GET" })
export default request;
```

1 יצירת מופע של axios
עם הגדרות שלנו:

2 הגדרת מאזין להצלחה
שישרשר הלאה את הpromise
(בדומה לmiddleware או לציתות-לטלפון)

3 הגדרת מאזין לכשלון הבקשה
שישרשר הלאה את הpromise
(בדומה לmiddleware או לציתות-לטלפון)

4 נגדיר פונקציה שמקבלת אובייקט
מסוג AxiosRequestConfig
ומחזירה את הclient שלנו

שימוש בפונקציה request (הaxios שהגדרנו בשלב הקודם)

src/services/products-service.ts

```
import request from "../utils/axios-interceptors";

const productUrl = "/products";

export const getProducts = () =>
  request({
    url: productUrl,
  });

export const getProduct = (id: number) =>
  request({
    url: `${productUrl}/${id}`,
  });
```

כתובת חלקית
(הקליינט של axios מוסיף את זה לכתובת המלאה של הapi)

שימוש בפונקציה request - אין קוד חוזר - רק הגדרות של http כגון כתובת

בשלב הזה אנחנו שולפים יפה מידע מאובטח עם JWT

Dynamic Routes
ניווט לעמוד מוצר לפי הid של המוצר

src/routes/Products.ts

```
import { Link } from "react-router-dom";
import useProducts from "../hooks/useProducts";

const Products = () => {
  const { products, loading, error } = useProducts();

  return (
    <div>
      <h1 className="text-center mb-3 text-3xl">Products</h1>

      {error && <p>Error: {error.message ?? "something went wrong"}</p>}
      {loading && <p>Loading...</p>}
      {!loading && !error && (
        <ul>
          {products &&
            products.map((p) => (
              <li className="flex p-4 shadow-lg" key={p.id}>
                <Link to={`${p.id}`}>
                  <h2>{p.name}</h2>
                </Link>
              </li>
            ))}
        </ul>
      )}
    </div>
  );
};

export default Products;
```

בלחיצה על מוצר - נשתמש בLink
שיקשר אותנו לעמוד של המוצר לפי Id

Dynamic Routes
ניווט לעמוד מוצר לפי הid של המוצר

נגדיר קומפוננטה לפרטי מוצר בודד:

src/routes/Product.tsx

```
const Product = () => {
  return (
    <div>Product</div>
  )
}

export default Product
```

Dynamic Routes
ניווט לעמוד מוצר לפי הid של המוצר

src / App.tsx

```
<Route
  path="/products/:id"
  element={
    <ProtectedRoute>
      <Product />
    </ProtectedRoute>
  }
/>
```

נגדיר נתיב שינווט אותנו
Route (הגדרות לראוטור)

Dynamic Routes
עבודה על הקומפוננטה של פרטי מוצר:

src/routes/Product.tsx

```
import { useParams } from "react-router-dom"

const Product = () => {
  const params = useParams()
  const id = params.id;

  console.log(id);
  return (
    <div>Product</div>
  )
}

export default Product
```

1

חילוץ הid של המוצר מהurl

```
import { useParams } from "react-router-dom"

const Product = () => {
  const {id} = useParams()
  console.log(id);

  return (
    <div>Product</div>
  )
}

export default Product
```

2

חילוץ הid של המוצר מהurl
(שימוש בobject destructuring)
יותר קצר ומודרני

Dynamic Routes
עבודה על הקומפוננטה של פרטי מוצר:

src/routes/Product.tsx

```
import { useParams } from "react-router-dom";

const Product = () => {
  const {id: idString} = useParams();
  const id = parseInt(idString);

  if (!id || isNaN(id)) {
    return <div>Invalid ID</div>;
  }

  return <div>Product</div>;
};

export default Product;
```

1
2
3

ככלל - הטיפוס של הפרמטרים שמקבלים מהparams
הוא תמיד מחרוזת.

נרצה להמיר את זה למספר:

1) נחלץ את הid ונקרא לו idString
2) נמיר למספר
3) נבדוק אם ההמרה הצליחה או נכשלה. אם נכשלה נציג הודעת שגיאה
אחרת נתקדם הלאה.

Dynamic Routes
עבודה על הקומפוננטה של פרטי מוצר:

```
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";

const Product = () => {
  const [product, setProduct] = useState<ProductType>(null);
  const { id: idString } = useParams();
  const id = parseInt(idString);

  useEffect(() => {
    getProduct(id)
      .then((product) => {
        setProduct(product);
      })
      .catch((error) => {
        console.error(error);
      })
      .finally(() => {});
  }, [id]);

  if (!id || isNaN(id)) {
    return <div>Invalid ID</div>;
  }

  return (
    <div>
      Product Details
      {product && (
        <div>
          <div>{product.name}</div>
          <div>{product.price}</div>
        </div>
      )}
    </div>
  );
};

export default Product;
```

בעזרת useEffect אנחנו מריצים את הפונקציה getProduct(id)
רק פעם אחת כשהקומפוננטה עולה - ולא בכל רינדור של הקומפוננטה.

שינוי מצב כגון setProduct גורר רינדור מחדש של הקומפוננטה

ולכן useEffect מונע פה קריאה רקורסיבית של getProduct=>setProduct

ביקורת:

כל הרעיון של useEffect
ניהול state למצב וכו'
נראה לנו מוכר מאוד מהקומפוננטה של Products
שם יצרנו Hook כדי להפריד את הלוגיקה.

אפשרות: 1) נעתיק את הhook ונשנה אותו לצרכים שלנו
(לא טוב כי זה מפר את עקרון DRY)

אפשרות 2) נכתוב custom Hook גנרי
לשליפת מידע מhttp
(בעמוד הבא)

Custom Hook גנרי
לשליפת מידע מhttp

src/hooks/useFetch.ts

```
import { useEffect, useState } from "react";
import request from "../utils/axios-interceptors";

const useFetch = <T = any> (url: string) => {

    const [loading, setLoading] = useState(true);
    const [data, setData] = useState<T>(null);
    const [error, setError] = useState<string>(null);

    useEffect(() => {
        request({url})
        .then((res) => {
            setData(res.data);
            setLoading(false);
        })
        .catch((err) => {
            setError(err);
            setLoading(false);
        });
    }, [url]);
  return {loading, data, error}
}

export default useFetch
```

שימוש בCustom Hook הגנרי שהגדרנו:

Dynamic Routes
הקומפוננטה של פרטי מוצר:

src/routes/Product.tsx

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import { getProduct } from "../services/products-service";
import { ProductType } from "../@types";
import useFetch from "../hooks/useFetch";

const Product = () => {

  const { id: idString } = useParams();
  const id = parseInt(idString);

  const {data: product, error, loading} = useFetch<ProductType>(`/products/${id}`);

  if (id === null || isNaN(id)) {
    return <div>Invalid product id</div>;
  }

  return (
    <div>
      <h1>Product Details</h1>
      {product && (
        <div>
          <h2>{product.name}</h2>
          <p>{product.description}</p>
          <p>{product.price}</p>
        </div>
      )}
    </div>
  );
};

export default Product;
```

כך נשתמש בCustom Hook הגנרי שהגדרנו:

Dynamic Routes
עבודה על הקומפוננטה של פרטי מוצר:

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import { getProduct } from "../services/products-service";
import { ProductType } from "../@types";
import useFetch from "../hooks/useFetch";

const Product = () => {

  const { id: idString } = useParams();
  const id = parseInt(idString);

  const {data: product, error, loading} = useFetch<ProductType>(`/products/${id}`);

  if (id === null || isNaN(id)) {
    return <div>Invalid product id</div>;
  }

  return (
    <div>
      <h1>Product Details</h1>
      {product && (
        <div>
          <h2>{product.name}</h2>
          <p>{product.description}</p>
          <p>{product.price}</p>
        </div>
      )}
    </div>
  );
};

export default Product;
```

קיצורי מקשים:

alt shift f

פרמוט המסמך

ctrl + d

מסמנים ביטוי או שורה עם העכבר ואז לוחצים ctrl+d

וזה מביא לסימון של הערך הזהה הבא במסמך

alt ⬇

הורדת שורה למטה

ctrl + p

חיפוש של קובץ בפרוייקט

alt shift ⬇

שכפול שורה

ctrl .
quick fix
(alt enterל מקביל)

ctrl + shift + p

חיפוש של הגדרות או תוספים בvscode