Pracownia Specjalistyczna

Aplikacje Internetowe Oparte o Komponenty

Projekt 1

Temat: Kalkulator kalorii - React

Wykonujący projekt:

- Magda Zaborowska
- Patryk Wójtowicz
- Michał Wołosewicz

Studia dzienne I stopnia

Kierunek: Informatyka

Semestr: V Grupa zajęciowa: PS 4

Prowadzący pracownie:

Dr inż. Urszula Kużelewska

.....

OCENA

Data oddania projektu

24.01.2022 r.

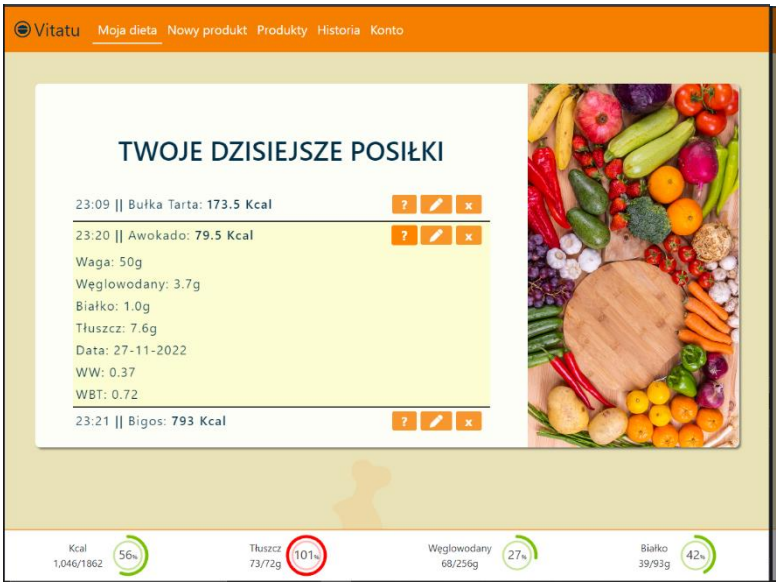
.....

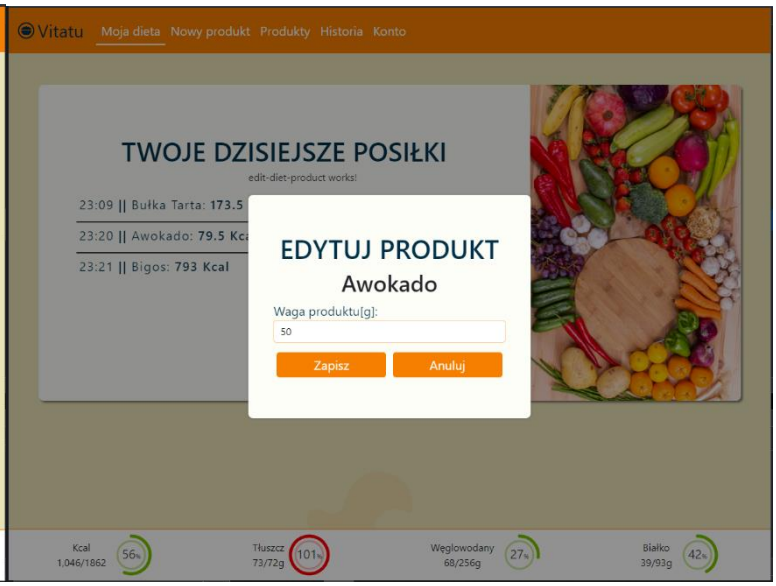
Data i podpis prowadzącego

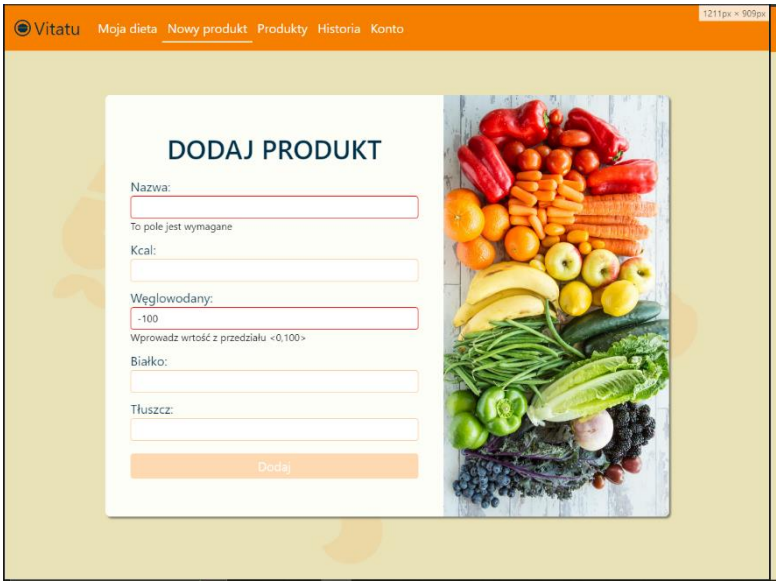
Opis projektu

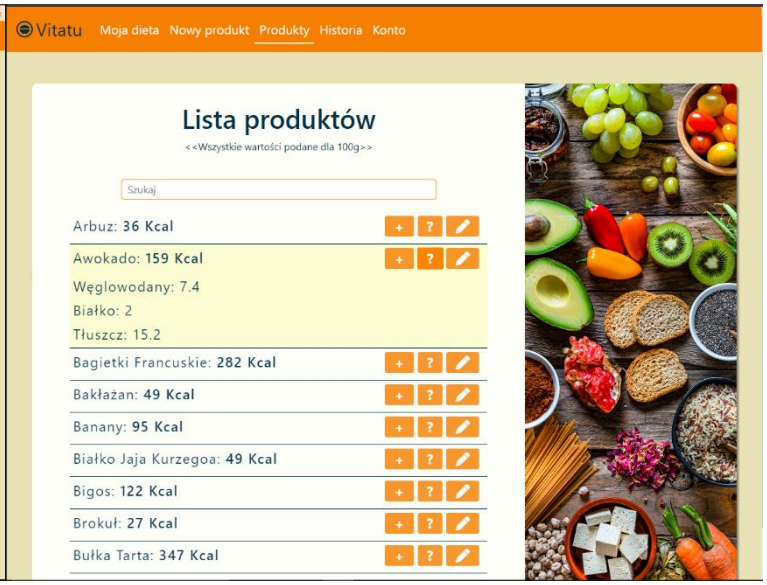
Celem projektu było stworzenie aplikacji internetowej w technologii Angular. Aplikacja ma na celu pomoc monitorowania swojej diety. Na podstawie wzrostu, wieku, wagi itp. wyliczane jest zapotrzebowanie użytkownika na kalorie oraz makroskładniki w ciągu dnia. Użytkownik ma możliwość zapisywania spożytych produktów, których wartości odliczane są od dziennego zapotrzebowania.

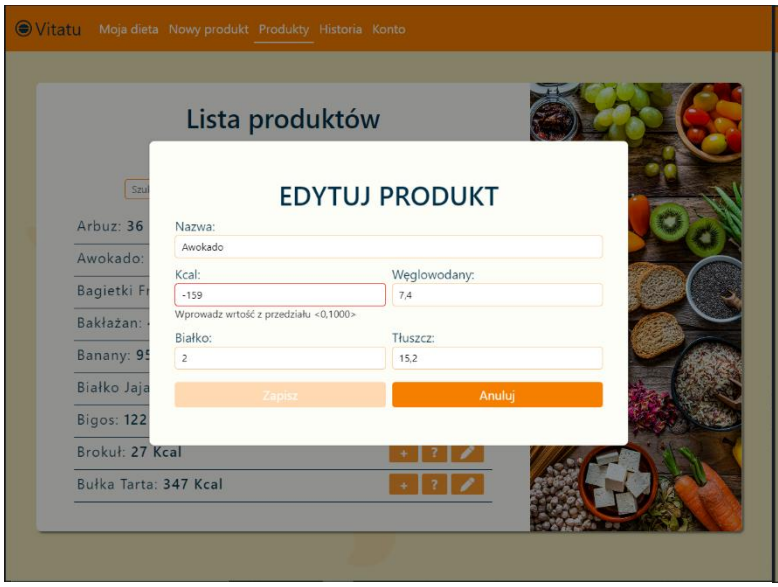
Aplikacja

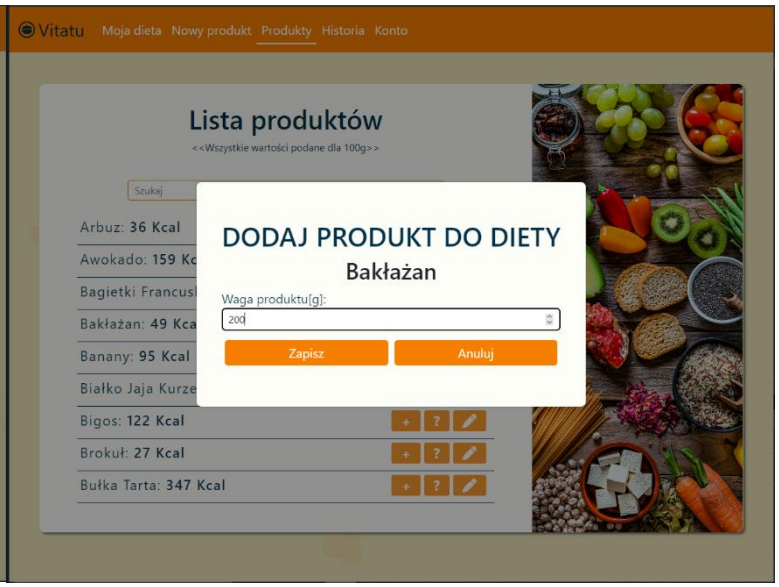


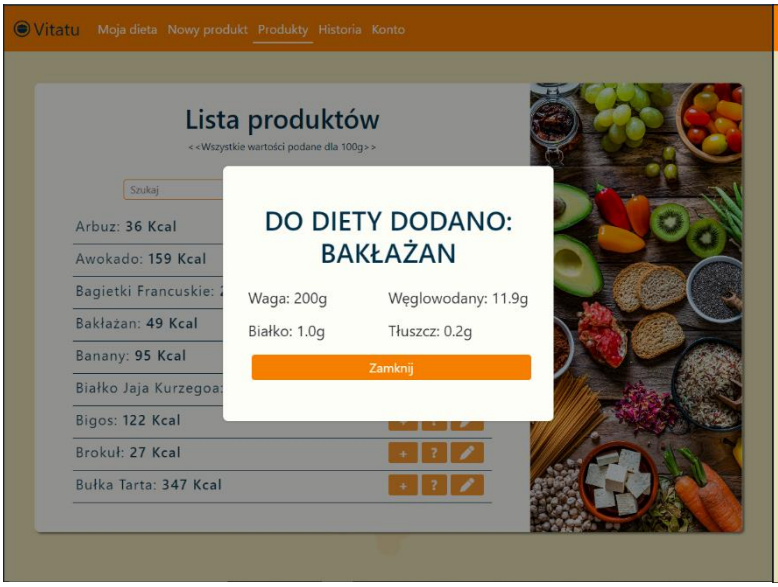


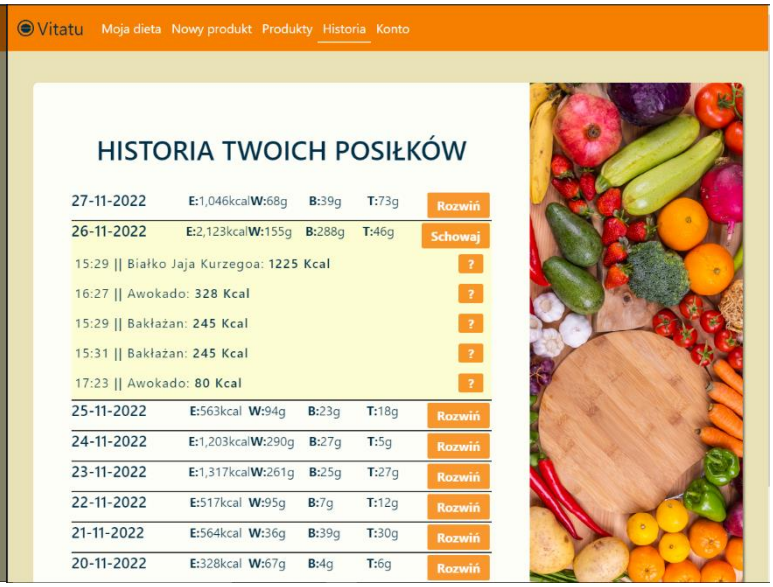


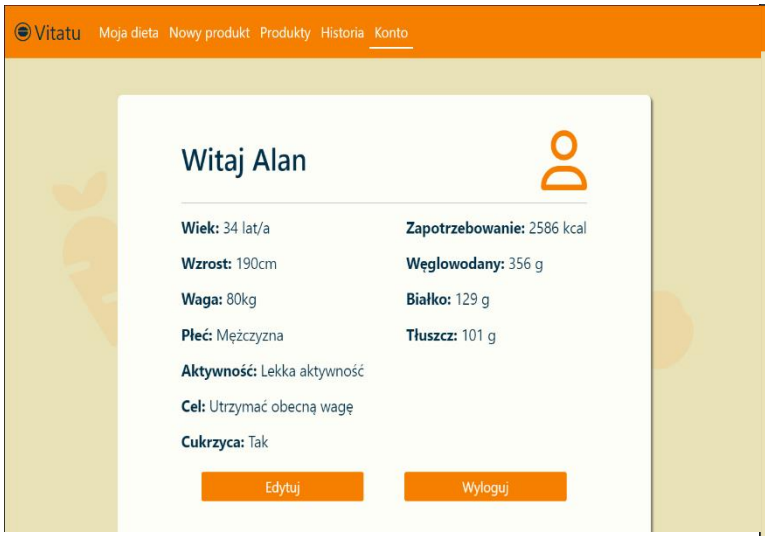


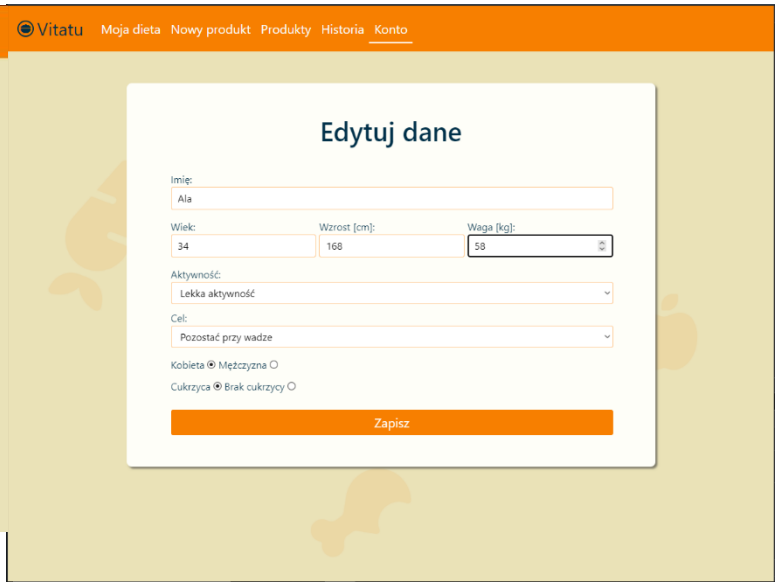

Vitatu
Moja dieta
Nowy produkt
Produkty
Historia
Konto
Lista produktów
EDYTUJ PRODUKT
Nazwa:
Awokado
Kcal:
-159
Węglowodany:
7,4
Wprowadz wrtość z przedziału <0,1000>
Białko:
2
Tłuszcz:
15,2
Zapisz
Anuluj
Arbuz: 36
Awokado:
Bagietki Fr
Bakłażan:
Banany: 95
Białko Jaja
Bigos: 122
Brokuł: 27 Kcal
Bułka Tarta: 347 Kcal


Vitatu
Moja dieta
Nowy produkt
Produkty
Historia
Konto
Lista produktów
<<Wszystkie wartości podane dla 100g>>
Szukaj
DODAJ PRODUKT DO DIETY
Bakłażan
Waga produktu[g]:
200
Zapisz
Anuluj
Arbuz: 36 Kcal
Awokado: 159 Kc
Bagietki Francusk
Bakłażan: 49 Kca
Banany: 95 Kcal
Białko Jaja Kurze
Bigos: 122 Kcal
Brokuł: 27 Kcal
Bułka Tarta: 347 Kcal


Vitatu
Moja dieta
Nowy produkt
Produkty
Historia
Konto
Lista produktów
<<Wszystkie wartości podane dla 100g>>
Szukaj
DO DIETY DODANO:
BAKŁAŻAN
Waga: 200g
Węglowodany: 11.9g
Białko: 1.0g
Tłuszcz: 0.2g
Zamknij
Arbuz: 36 Kcal
Awokado: 159 Kcal
Bagietki Francuskie:
Bakłażan: 49 Kcal
Banany: 95 Kcal
Białko Jaja Kurzegoa:
Bigos: 122 Kcal
Brokuł: 27 Kcal
Bułka Tarta: 347 Kcal


Vitatu
Moja dieta
Nowy produkt
Produkty
Historia
Konto
HISTORIA TWOICH POSIŁKÓW
27-11-2022 E:1,046kcal W:68g B:39g T:73g Rozwiń
26-11-2022 E:2,123kcal W:155g B:288g T:46g Schowaj
15:29 || Białko Jaja Kurzegoa: 1225 Kcal
16:27 || Awokado: 328 Kcal
15:29 || Bakłażan: 245 Kcal
15:31 || Bakłażan: 245 Kcal
17:23 || Awokado: 80 Kcal
25-11-2022 E:563kcal W:94g B:23g T:18g Rozwiń
24-11-2022 E:1,203kcal W:290g B:27g T:5g Rozwiń
23-11-2022 E:1,317kcal W:261g B:25g T:27g Rozwiń
22-11-2022 E:517kcal W:95g B:7g T:12g Rozwiń
21-11-2022 E:564kcal W:36g B:39g T:30g Rozwiń
20-11-2022 E:328kcal W:67g B:4g T:6g Rozwiń


Vitatu
Moja dieta
Nowy produkt
Produkty
Historia
Konto
Witaj Alan
Wiek: 34 lat/a
Wzrost: 190cm
Waga: 80kg
Płeć: Mężczyzna
Aktywność: Lekka aktywność
Cel: Utrzymać obecną wagę
Cukrzyca: Tak
Zapotrzebowanie: 2586 kcal
Węglowodany: 356 g
Białko: 129 g
Tłuszcz: 101 g
Edytuj
Wyloguj

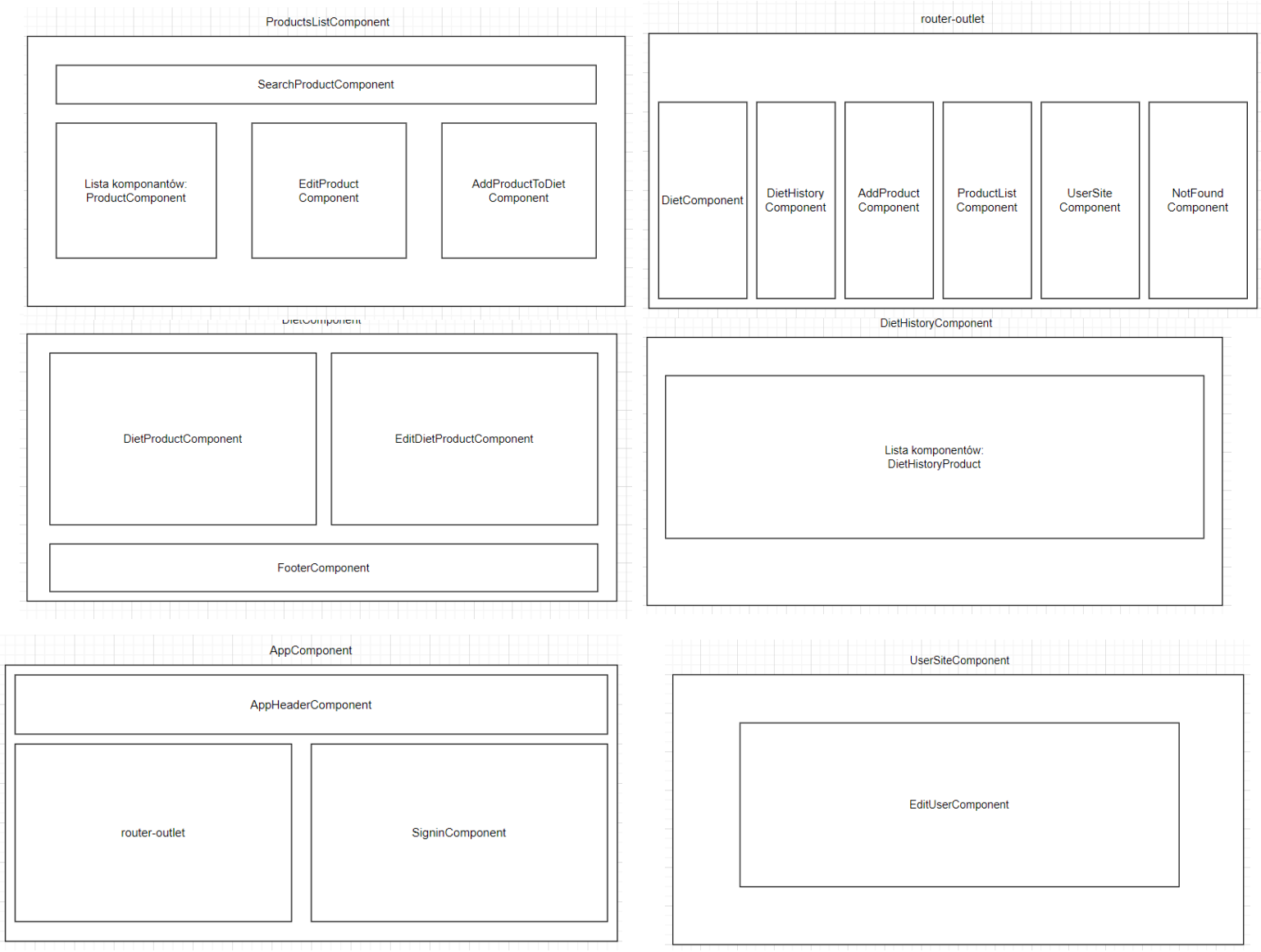

Vitatu
Moja dieta
Nowy produkt
Produkty
Historia
Konto
Edytuj dane
Imię:
Ala
Wiek:
34
Wzrost [cm]:
168
Waga [kg]:
58
Aktywność:
Lekka aktywność
Cel:
Pozostać przy wadze
Kobieta Mężczyzna
Cukrzyca Brak cukrzycy
Zapisz

Funkcjonalności

W projekcie zaimplementowano wszystkie funkcjonalności zadeklarowane przed przystąpieniem do projektu. Są to:

- Logowanie do systemu
- Konto użytkownika – wprowadzanie wagi, wzrostu, celu diety itp.
- Obliczanie zapotrzebowanie kalorycznego oraz na makroskładniki
- Przeliczanie wymiennika węglowodanowego WW oraz białkowo-tłuszczowego WBT dla cukrzyków,
- Baza produktów (nazwa, kcal, węgle, tłuszcze, białka)
- Wyszukiwarka produktów
- Dodawanie nowych produktów do bazy lub edycja istniejących produktów
- Dodawanie zjedzonych produktów do dziennego jadłospisu, edycja i usuwanie
- Wyświetlanie podsumowania posiłku po dodaniu go do jadłospisu aktualnego dnia
- Historia dnia, historia długoterminowa razem ze statystykami
- Statystyki aktualnego dnia

Architektura komponentów



Ścieżki i komponenty związane z routigniem

Główne komponenty to:

- DietComponent,
- UserSiteComponent,
- DietHistoryComponent
- AddProductComponent
- ProductsComponent
- LoginComponent

```
<Routes>
    <Route path="/" element={ProductsComponent}></Route>
    <Route path="/diet" element={DietComponent}></Route>
    <Route path="/diet/:id" element={DietComponent}></Route>
    <Route path="/products" element={ProductsComponent}></Route>
    <Route path="/add-product"  element={AddProductComponent}></Route>
    <Route path="/diet-history" element={DietHistoryComponent}></Route>
    <Route path="/user" element={UserComponent}></Route>
    <Route path="/login" element={LoginComponent}></Route>
</Routes>
```

Przełączamy się pomiędzy nimi za pomocą nawigacji lub modyfikując link. Użyto również ścieżkę z parametrem. Wykorzystywana jest ona do usuwania produktu z diety

```
const deleteDietItemHandler = () => {
    navigate(`/diet/${props.product.IdDiet}`);
};
```

```
useEffect(() => {
    if (params.id) {
        const prodId = props.dietList.findIndex(
            (p) => p.IdDiet === params.id
        );
        if (prodId !== -1) {
            setProductToDelete(props.dietList[prodId]);
            setIsDeleting(true);
        }
    }
}, [params]);
```

API serwera

Do przechowywania danych skorzystano w json-server
Przechowywane są tam:

- Lista wszystkich produktów(kalorie, makroskładniki)
- Historia diety użytkownika(co i kiedy zjadł)

Dostęp do danych zapewniony jest dzięki 4 rodzajów żądań http. Motody dostępowe znajdują się w osobnych plikach: services/dietAPI.ts oraz services/productAPI.ts. Skorzystano z axiosa

```
const productAPI = {
    getProducts: async () => {
        try {
            const response = await api.get("./products");
            const productList: ProductClass[] = [];
            response.data.forEach((prod: ProductType) => {
                productList.push(
                    new ProductClass(
                        prod.carbohydrates,
                        prod.fat,
                        prod.kcal,
                        prod.name,
                        prod.protein,
                        prod.id
                    )
                );
            });
            return productList;
        } catch (err) {
            return false;
        }
    },
    postProduct: async (product: ProductClass) => {
        try {
            const response = await api.post("./products", product);
            return response.data;

        } catch (err) {
            return false
        }
    },
    deleteProduct:async(productId:string)=>{
        try{
            await api.delete(`./products/${productId}`)
        }catch(error){
            return false;
        }
    },
    editProduct:async(product:ProductClass)=>{
        try{
            const response = await api.put(`./products/${product.Id}`,product)
            console.log(response.data);
            return response.data
        }catch(err){
            return false;
        }
    }

};
```

Elementy techniczne

Np.	Nazwa	Pkt	Moje Pkt
1	własna walidacja danych wprowadzanych przez użytkownika (w każdym przypadku wprowadzania danych, co najmniej 4 różne przypadki danych)	2	2
2	obowiązkowa weryfikacja typu danych (PropTypes) przekazywanych do wszystkich komponentów (nie stosujemy typu 'any')	2	2
3	właściwe użycie typów komponentów (czy każdy z komponentów jest właściwie odwzorowany na komponent prezentacyjny lub stanowy)	1	1
4	dwukierunkowa komunikacja pomiędzy komponentami (czy jest w każdym spodziewanym przypadku)	1	1
5	co najmniej 4 komponenty reużywalne (komponenty, które mogą być użyte bez zmian w innym miejscu	2	2
6	modyfikacja danych odbywa się tylko w jednym komponencie	2	2
7	operacje modyfikacji danych za pomocą 4 rodzajów żądań http	1	1
8	żądania do serwera są zapisane w jednym oddzielnym pliku	1	1
9	routing (ścieżki 'routes', w tym jedna z parametrem)	1	1
10	wykorzystanie dwóch zmiennych właściwości routingu (np. navigate, params)	1	1
11	architektura Flux	3	3
12	brak błędów/ostrzeżeń w konsoli przegladarki	1	0
		18	17

1) Własna walidacja danych wprowadzanych przez użytkownika (w każdym przypadku wprowadzania danych, co najmniej 4 różne przypadki danych)

```
const useInput = (validateValue: (arg0: any) => boolean,value:string) => {
    const [inputState, dispatch] = useReducer(
        inputStateReducer,
        {value: value,
        isTouched: false,}
    );

    const valueIsValid = validateValue(inputState.value);
    const hasError = !valueIsValid && inputState.isTouched;

    const valueChangeHandler = (event:React.FormEvent<HTMLInputElement> ) => {
        dispatch({ type: "INPUT", value: event.currentTarget.value });
    };
    const inputBlurHandler = (event: React.FormEvent<HTMLInputElement>) => {
        dispatch({ type: "BLUR",value:'' });
    };
    const reset = () => {
        dispatch({ type: "RESET",value:'' });
    };

    return {
        value: inputState?.value ?? '',
        isValid: valueIsValid,
        hasError,
        valueChangeHandler,
        inputBlurHandler,
        reset,
    };
};
```

Napisano własny Hook który sprawdzał poprawność wprowadzonych danych

```
const {
    value: enteredKcal,
    isValid: enteredKcalIsValid,
    hasError: kcalHasError,
    valueChangeHandler: kcalChangeHandler,
    inputBlurHandler: kcalBlurHandler,
    reset: kcalReset,
} = useInput(
    (value) => +value >= 0 && +value <= 1000 && value.trim() !== "",
    props.product.Kcal + ""
);
```

2) Obowiązkowa weryfikacja typu danych (PropTypes) przekazywanych do wszystkich komponentów (nie stosujemy typu 'any')

```
type MyProps = {
    product: ProductClass;
    editProduct?: (product: ProductClass) => void;
    addProduct?: (product: ProductClass) => void;
};

const EditProduct = (props: MyProps) => {
```

3) Właściwe użycie typów komponentów (czy każdy z komponentów jest właściwie odwzorowany na komponent prezentacyjny lub stanowy)

   Projekt bazuje na komponentach funkcyjnych. W zależności użycia Hooków są one stanowe lub bezstanowe

4) Dwukierunkowa komunikacja pomiędzy komponentami (czy jest w każdym spodziewanym przypadku)
   Do komponentów przesyłane są dane oraz ”Eventy”

```
<DietList
    dietList={dietList}
    productList={productList}
    onDeleteDiet={deleteFromDietHandler}
    onEditDiet={editDietProductHandler}
    historyDiet={historyList}></DietList>
```

5) Co najmniej 4 komponenty reużywalne (komponenty, które mogą być użyte bez zmian w innym miejscu projektu)

```
const Card = (props: CardProps) => {
    return <div className={classes.card} >{props.children}</div>;
};
```

```tsx
const Error = (props: ErrorProps) => {
    return (
        <div className={classes.error}>
            <p >{props.message}</p>
        </div>
    );
};
const Input = (props: inputProps) => {
    return (
        <div className={classes["input-box"]}>
            <label htmlFor={props.id}>{props.label}:</label>
            <input
                type={props.type}
                value={props.value}
                id={props.id}
                onChange={props.onChange}
                onBlur={props.onBlur}
                className={props.isValid ? "" : classes.invalid}></input>
        </div>
    );
};
const DeleteBox = (props: MyProps) => {
    return (
        <div className={classes.box}>
            <div className={classes.content}>
                <Card>
                    <>
                        <h3>Jesteś pewnien?</h3>
                        <p>Zmiany są nieodwracalne</p>
                        <div className={classes.btns}>
                            <button onClick={props.onConfirm}>Usuń</button>
                            <button onClick={props.onCancel}>Anuluj</button>
                        </div>
                    </>
                </Card>
            </div>
        </div>
    );
};
```

6) Modyfikacja danych odbywa się tylko w jednym komponencie

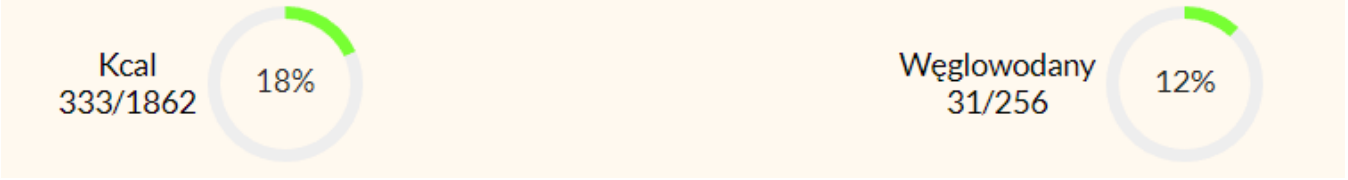
   Komunikacja z bazą wykonuje się jedynie w App.tsx. Jedynie tam wywoływane są funkcje modyfikujące dane.

11) Architektura Flux

Do logowania użyto Reduxa. Dzięki niemu wszystkie komponenty wiedzą czy użytkownik jest zalogowany oraz jaki jest użytkownik(wiek, waga, itp.)

Dodatkowe biblioteki użyte w aplikacji

Skorzystano z gotowego komponentu **React Circular Progressbar.** Są okrągłe progressbary



(https://www.npmjs.com/package/react-circular-progressbar)