Pracownia Specjalistyczna

Aplikacje Internetowe Oparte o Komponenty

Projekt 1

Temat: Kalkulator kalorii - Angular

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

29.11.2022 r.

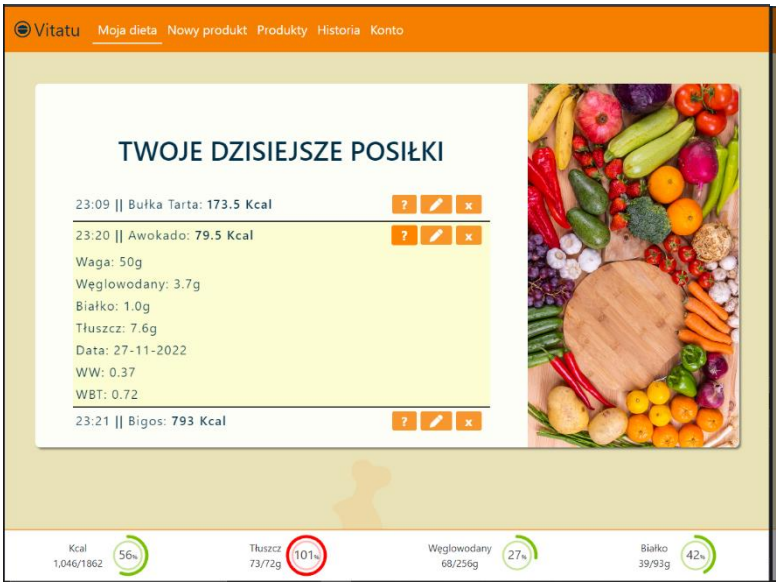
.....

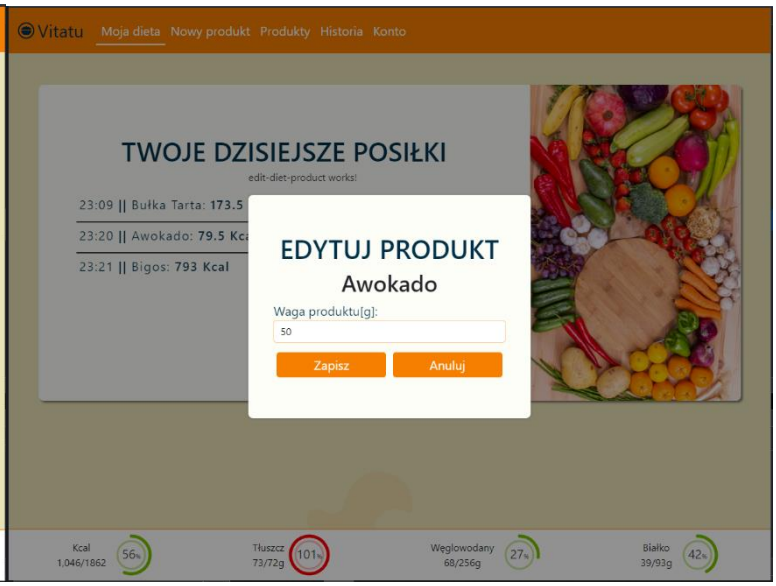
Data i podpis prowadzącego

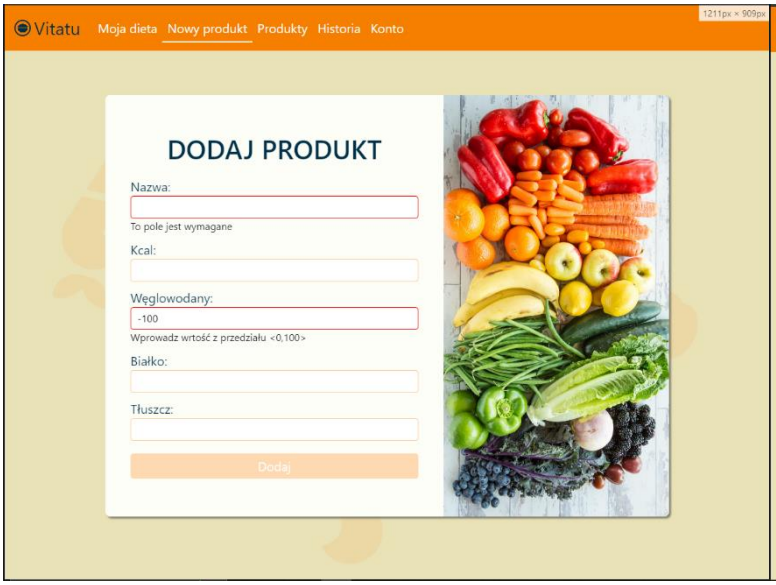
Opis projektu

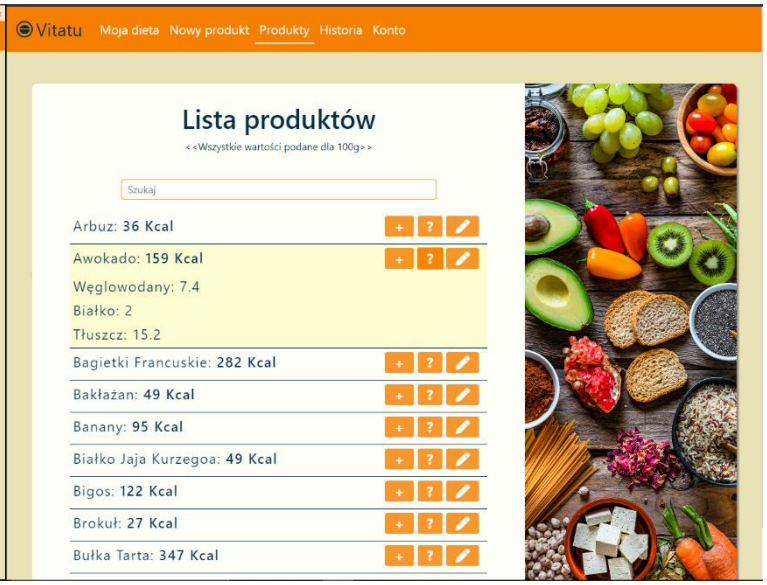
Celem projektu było stworzenie aplikacji internetowej w technologii Angular. Aplikacja ma na celu pomoc monitorowania swojej diety. Na podstawie wzrostu, wieku, wagi itp. wyliczane jest zapotrzebowanie użytkownika na kalorie oraz makroskładniki w ciągu dnia. Użytkownik ma możliwość zapisywania spożytych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

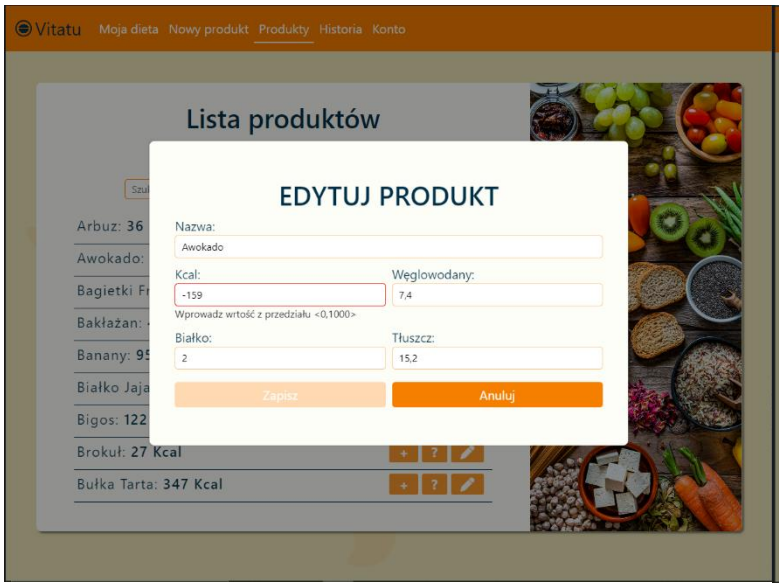
Aplikacja

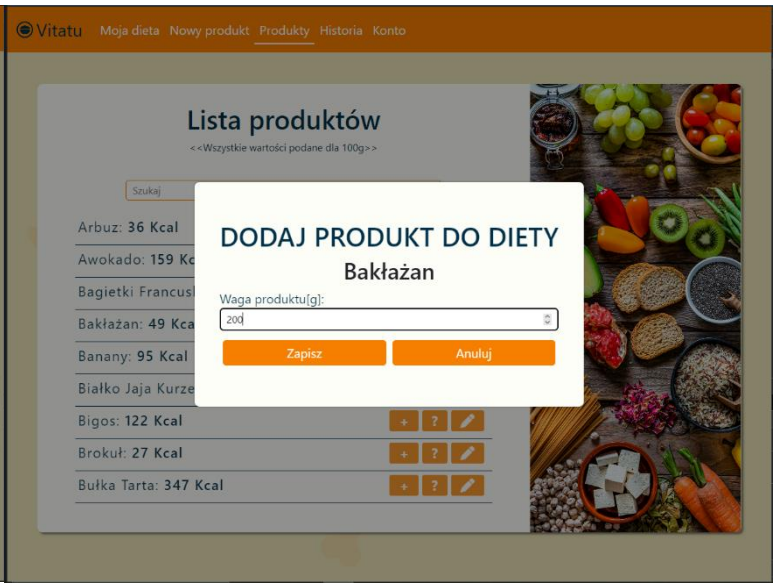


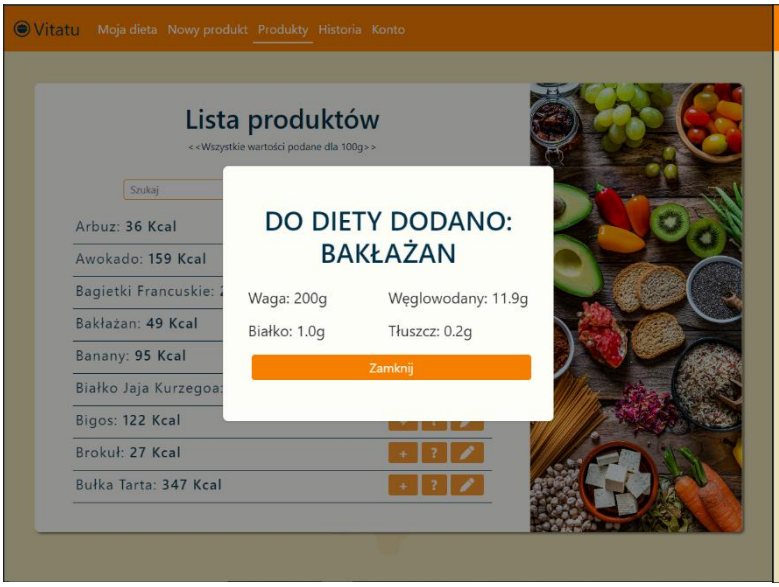


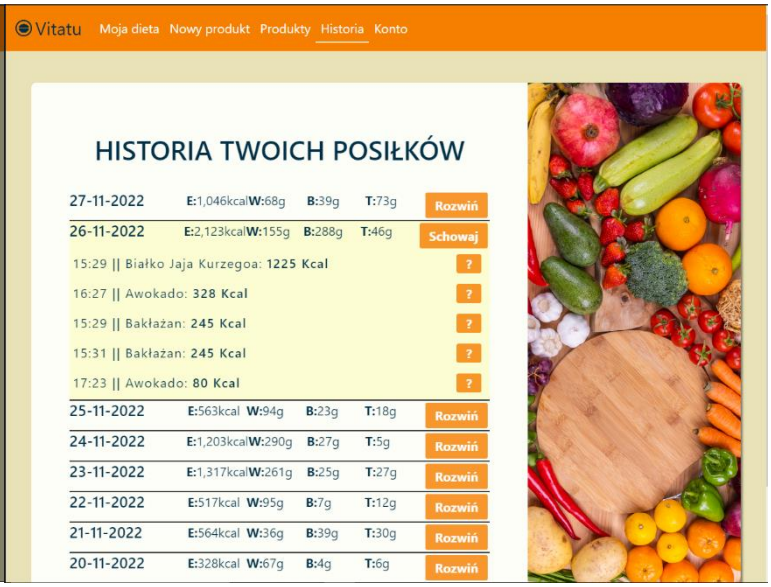


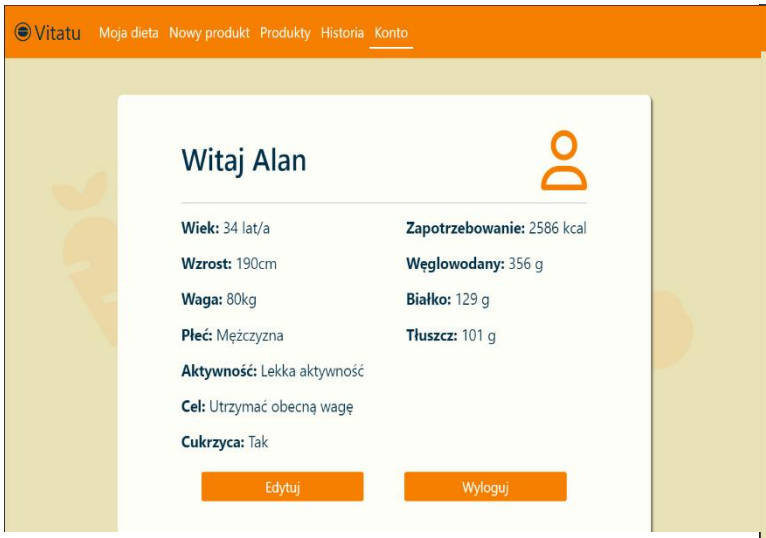


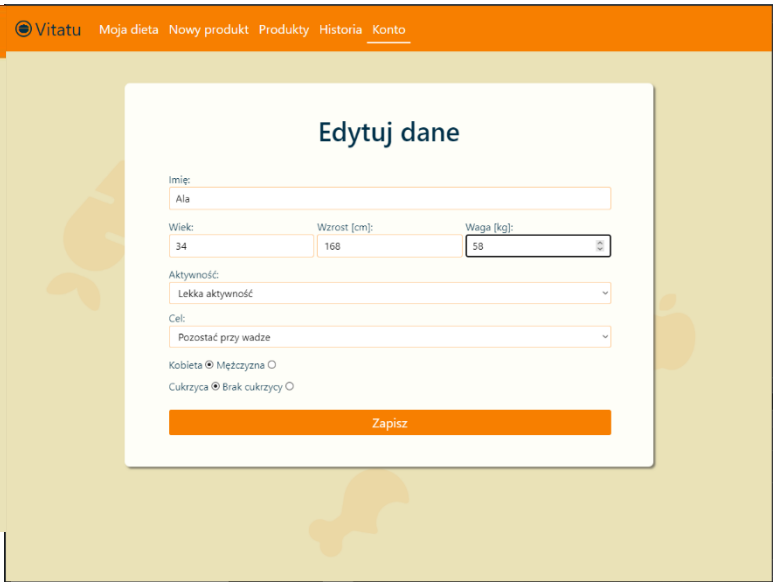










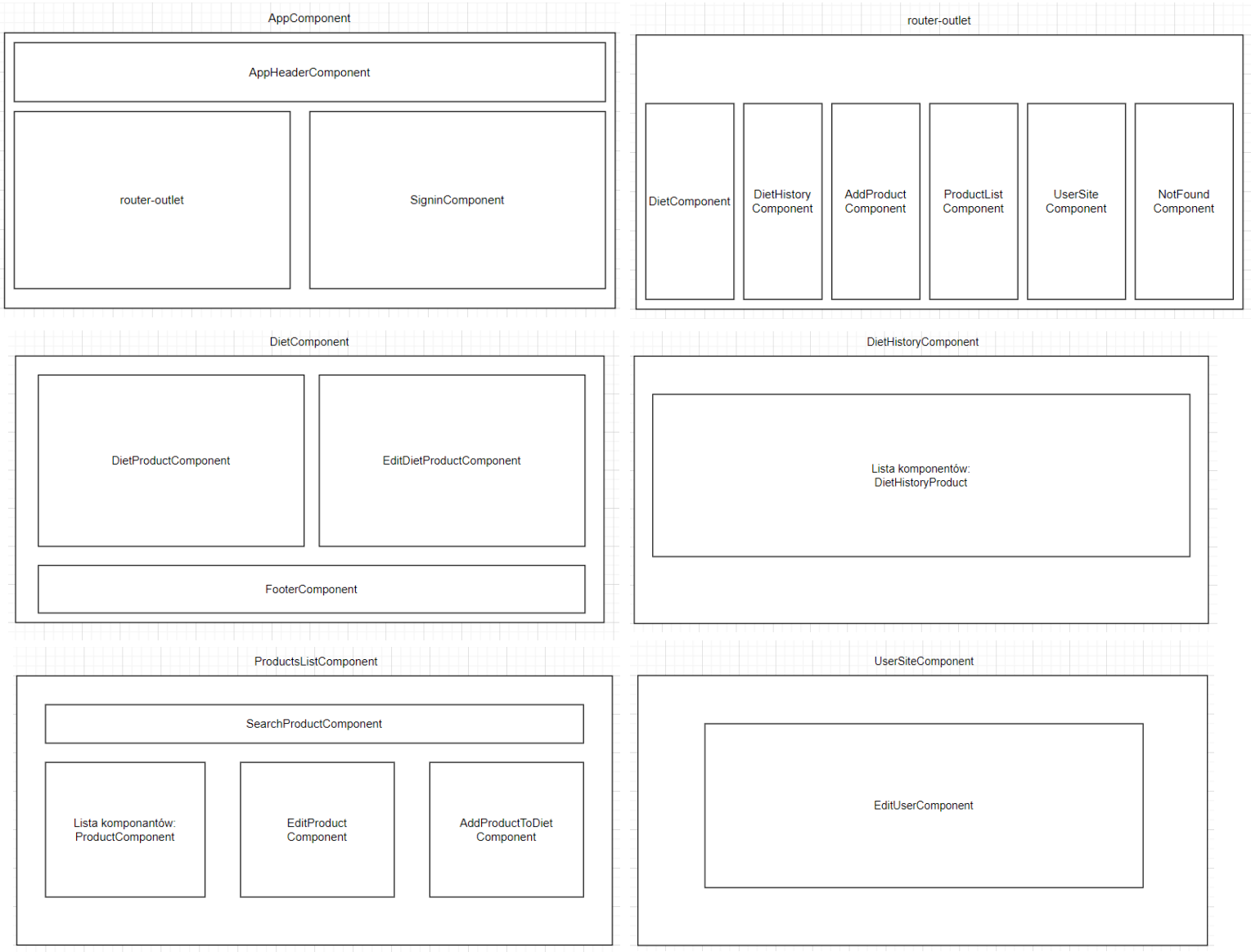


Funkcjonalności

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
- ProductsListComponent

```
const appRoutes: Routes = [
  { path: 'diet', component: DietComponent },
  { path: 'user-site', component: UserSiteComponent },
  { path: 'diet-history', component: DietHistoryComponent},
  { path: 'new-product', component: AddProductComponent },
  { path: 'products-list', component: ProductsListComponent },
  { path: 'products-list/:id', component: ProductsListComponent },
  { path: 'not-found', component: NotFoundComponent },
  { path: '', redirectTo: '/diet', pathMatch: 'full' },
  { path: '**', redirectTo: '/not-found' },
];
```

Przełączamy się pomiędzy nimi za pomocą nawigacji lub modyfikując link. W przypadku podania nieprawidłowego linku, wyświetla się komponent NotFoundComponent.
Użyto również ścieżkę z parametrem. Wykorzystywana jest ona do edycji produktu.

```
onEdit(): void {
  this.router.navigate([{ id: this.product.Id }], { relativeTo: this.route });
  this.onEditEvent.emit(this.product);
}
```

0-1 product.component.ts

```
onShowEdit(): void {
  this.paramsSubscription = this.route.params.subscribe((params) => {
    const id = params['id'];
    this.productToEdit = this.productList.find((p) => p.Id === id);
  });
  this.showEdit = true;
}
```

0-2products-list.component.ts

API serwera

Do przechowywania danych utworzono bazę na stronie https://firebase.google.com
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

```
createPosts(productData: Product) {
  this.http
    .post<{ name: string }>(
      'https://angular-projekt-7b1f8-default-rtdb.firebaseio.com/post.json',
      productData
    )
    .subscribe((responseData) => {
      // console.log(responseData);
    });
}
```

```
deletePost(id: string) {
  this.http
    .delete(
      'https://angular-projekt-7b1f8-default-rtdb.firebaseio.com/post/' +
        id +
        '.json'
    )
    .subscribe();
}
```

```
updateProduct(id: string, value: Product) {
  this.http
    .put(
      'https://angular-projekt-7b1f8-default-rtdb.firebaseio.com/post/' +
        id +
        '.json',
      value
    )
    .subscribe();
}
```

```
featchPosts() {
  return this.http
    .get<{ [key: string]: PostProduct }>(
      'https://angular-projekt-7b1f8-default-rtdb.firebaseio.com/post.json'
    )
    .pipe(
      map((responseData) => {
        const postsArray: Product[] = [];
        for (const key in responseData) {
          if (responseData.hasOwnProperty(key)) {
            const tempProd = responseData[key];
            postsArray.push(
              new Product(
                tempProd.carbohydrates,
                tempProd.fat,
                tempProd.kcal,
                tempProd.name,
                tempProd.protein,
                key
              )
            );
          }
        }
        return postsArray;
      })
    );
}
```

Elementy techniczne

Np.	Nazwa	Pkt
1	klasa TypeScript (czy zdefiniowano i zastosowano klasę do organizacji danych, czy pola w klasie są prywatne)	1
2	typy TypeScript (czy każda zmienna ma przyporządkowany typ)	1
3	zaawansowane elementy TypeScript (jeden z wymienionych): klasy pochodne TypeScript (czy wykorzystano również klasy pochodne) , getter+setter (czy wykorzystano i czy właściwie zostały dobrane)+parametry opcjonalne metod (czy są i czy właściwie dobrane)+modyfikatory dostępu w konstruktorze.	2
4	wykorzystanie formularzy, min. 5 elementów (czy właściwie wybrano dane do wprowadzania i dobrano rodzaj elementu formularza, czy nie ma dwustronnego wiązania danych w szablonie)	1
5	walidacja danych wprowadzanych przez użytkownika (w każdym przypadku wprowadzania danych, czy odpowiednio dobrano walidatory)	2
6	dwukierunkowa komunikacja pomiędzy komponentami (czy jest w każdym spodziewanym przypadku)	2
7	modyfikacja danych odbywa się tylko w jednym komponencie	1
8	operacje modyfikacji danych za pomocą 4 rodzajów żądań http	1
9	dane pochodzące z jednej klasy usługi	1
10	dodatkowy serwis (a)synchroniczny	1
11	własna dyrektywa	1
12	wykorzystanie dowolnego filtru standardowego w szablonie	1
13	implementacja własnego filtru	2
14	routing (ścieżki 'routes', w tym jedna z parametrem, operacje na obiekcie ActivateRoute i Route)	1

1) Zdefiniowano 4 klasy (folder types): Diet, DietHistory, Product, User – wrzystkie ich pola są prywatne

3) Klasa Diet dziedziczy z klasy Product, w każdej klasie wykorzystywane są gettery i settery, W konstruktorze klasy Product i Diet są opcjonalne parametry, odpowiednio: id, idDiet

4, 5) Wykorzystano formularze reaktywne(EditProductComponent, AddProductToDietComponent, AddProductComponent, EditDietProductComponent, EditUserComponent, SigninComponent), zadbano o odpowiednie komunikaty w przypadku wpisania błędnych wartości. W żadnym z nich nie ma dwustronnego wiązania.

```
this.addProductForm = new FormGroup({
  name: new FormControl(this.product.Name, [
    Validators.required,
    this.forbidenName,
  ]),
  kcal: new FormControl(this.product.Kcal, [
    Validators.required,
    Validators.min(0),
    Validators.max(1000),
  ]),
  carbohydrates: new FormControl(this.product.Carbohydrates, [
    Validators.required,
    Validators.min(0),
    Validators.max(100),
  ]),
  protein: new FormControl(this.product.Protein, [
    Validators.required,
    Validators.min(0),
    Validators.max(100),
  ]),
  fat: new FormControl(this.product.Fat, [
    Validators.required,
    Validators.min(0),
    Validators.max(100),
  ]),
});
```

9) Metody związane z http requests umieszczone są w oddzielnych serwisach.

10) Zaimplementowano dodatkowy serwis UserService(user-site), który udostępnia metody logowania i wylogowania, oraz zmiany wartości obiektu User

```
@Injectable({
  providedIn: 'root',
})
export class UserService {
  user = new User('Ala', 34, 168, 58, 1.375, 'Kobieta', true, 1);
  signedin=false;

  changeUser(newUser:User){
    this.user=newUser
  }
  login(){
    this.signedin=true;
  }
  logout(){      You, 31 minut temu • Uncommitted changes
    this.signedin=false;
  }
}
```

11) Utworzono dyrektywę HighlightDirective. Jest ona używana między innymi w komponencie ProductsListComponent do wyróżniania elementu listy, na który najechaliśmy myszką

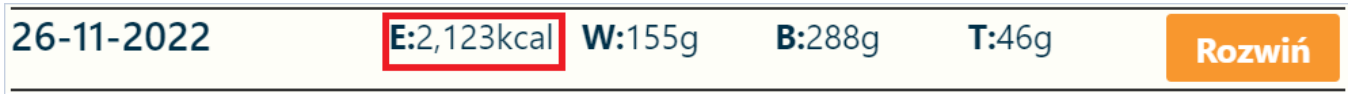
```typescript
@Directive({
  selector: '[appHighlight]',
})
export class HighlightDirective implements OnInit {
  @Input() defaultColor: string = 'rgb(254, 254, 248)';
  @Input() highlightColor: string = 'rgb(254, 254, 248)';
  @HostBinding('style.backgroundColor') backgroundColor: string;
  @HostBinding('style.transition') transition: string;

  constructor() {}
  ngOnInit() {
    this.backgroundColor = this.defaultColor;
  }
  @HostListener('mouseenter') onMouseEnter() {
    this.backgroundColor = 'rgb(253, 253, 210)';
    this.transition = 'background-color .1s';
  }
  @HostListener('mouseleave') onMouseLeave() {
    this.backgroundColor = 'transparent';
    this.transition = 'background-color .1s';
  }
}
```

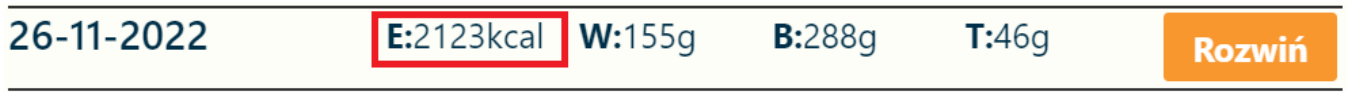
12) Wykorzystano filtr numeryczny między innymi w komponencie DietProductComponent

```
<p>Węglowodany: {{product.Carbohydrates | number: '1.1-2'}}g </p>
<p>Białko: {{product.Protein | number: '1.1-2'}}g </p>
<p>Tłuszcz: {{product.Fat | number: '1.1-2'}}g </p>
<p>Data: {{product.Date | date: 'dd-MM-yyyy' }} </p>
```

13) Zaimplementowano filtr noCommaPipe. Służy on do usunięcia przecinka z zapisu liczby większej od 1000. Używany między innymi w DietHistoryComponent



0-1 Efakt po użyciu | number



0-2 Efekt po użyciu naszego filtra | noComma

Dodatkowe biblioteki użyte w aplikacji

W aplikacji korzystano z ikonek ze strony https://fontawesome.com. Aby ich używać należy w głównym pliku html w sekcji head umieścić swój kit.

```
<script src="https://kit.fontawesome.com/6bf517e9f3.js" crossorigin="anonymous"></script>
```

Użyto również gotowego komponentu **ng-circle-progress (**https://www.npmjs.com/package/ng-circle-progress). Jest to prosty komponent oparty o SVG.
Należało go zinstalować: **npm install ng-circle-progress –save,** następnie umieścić w importach w app.module.ts. Jest po okrągły progres. Użyty jedynie w celach wizualnych w komponencie FooterComponent:

Podział pracy w zespole

- Magda Zaborowska 33%
- Patryk Wójtowicz 33%
- Michał Wołosewicz 33%

Wszyscy w równym stopniu uczestniczyli w zaimplementowaniu poszczególnych elementów kodu.