

## Opis

Projekt to sklep internetowy Alledrogo. Na potrzeby projektu przygotowałem:

- projekt wizualny strony wykonany w figmie
- repozytorium i roadmape na githubie
- pełen branding marki “Alledrogo”

## Stack Technologiczny

Do zbudowania sklepu internetowego skorzystałem z:

-  NextJS - framework do biblioteki React używany do budowania aplikacji renderowanych po stronie serwera
-  React - biblioteka Javascript używana do tworzenia UI (interfejsu użytkownika)
-  CSS modules - sposób stylowania używając czystego CSS importowanego w plikach JS
-  Typescript - statycznie typowany język programowania budowany (eksportowany) do Javascriptu
-  NodeJS - środowisko uruchomieniowe Javascriptu po stronie serwera

# Biblioteki

Pomniejsze biblioteki wykorzystane do projektu

ad

Axios - klient HTTP (zarówno dla client-side oraz server-side)

Sharp - biblioteka do optymalizacji zdjęć (zalecana do używania z NextJS)

MUI - biblioteka ułatwiające korzystanie z Material UI (wykorzystałem jedynie ikony)



package.json

```
"dependencies": {  
  "@emotion/react": "^11.10.6",  
  "@emotion/styled": "^11.10.6",  
  "@mui/icons-material": "^5.11.11",  
  "@mui/material": "^5.11.13",  
  "@types/node": "18.15.3",  
  "@types/react": "18.0.28",  
  "@types/react-dom": "18.0.11",  
  "axios": "^1.3.4",  
  "eslint": "8.36.0",  
  "eslint-config-next": "13.2.4",  
  "next": "13.2.4",  
  "react": "18.2.0",  
  "react-dom": "18.2.0",  
  "sharp": "^0.32.0",  
  "typescript": "5.0.2"  
}
```

# Branding

ad



**Główny kolor**  
#FF5A00



**Tło**  
#1A1A1A



**Szary**  
#1E1E1E



**Jasny szary**  
#929292

# Alledrogo

# Alledrogo

**Główny font**  
*Lato Łukasz Dziedzic*

**Poboczny font**  
(użyty w logo)  
*Inter Rasmus Anderson*

# Materiały

ad



Repozytorium na Githubie



Strona



Branding



Projekt wizualny



Roadmap (github project)

## Narzędzia

Narzędzia wykorzystane przy tym projekcie

- ➡ Visual Studio Code - edytor kodu
- ➡ Github & Git - publikacja kodu i kontrola wersji
- ➡ Figma - Projekt wizualny strony, przygotowanie brandingu oraz stworzenie tego PDFu
- ➡ Vercel - Hosting strony
- Showcode - wizualizacja kodu w sprawozdaniu

Oraz pomniejsze narzędzia

# Informacje

ad

Twórca projektu: **Bartosz Sułkowski**

Okres realizacji:

- część praktyczna: 15 marca - 7 kwietnia
- część teoretyczna: 7 kwietnia - 9 kwietnia

Typ renderowania strony: statyczna (Static site generation)

## Wnioski

Projekt nie był trudny w realizacji. Czas wynika z luźnego podejścia do jego przygotowania, co nie jest minusem ponieważ skupiłem się na dobrym kodzie strony i dokładnej dokumentacji.

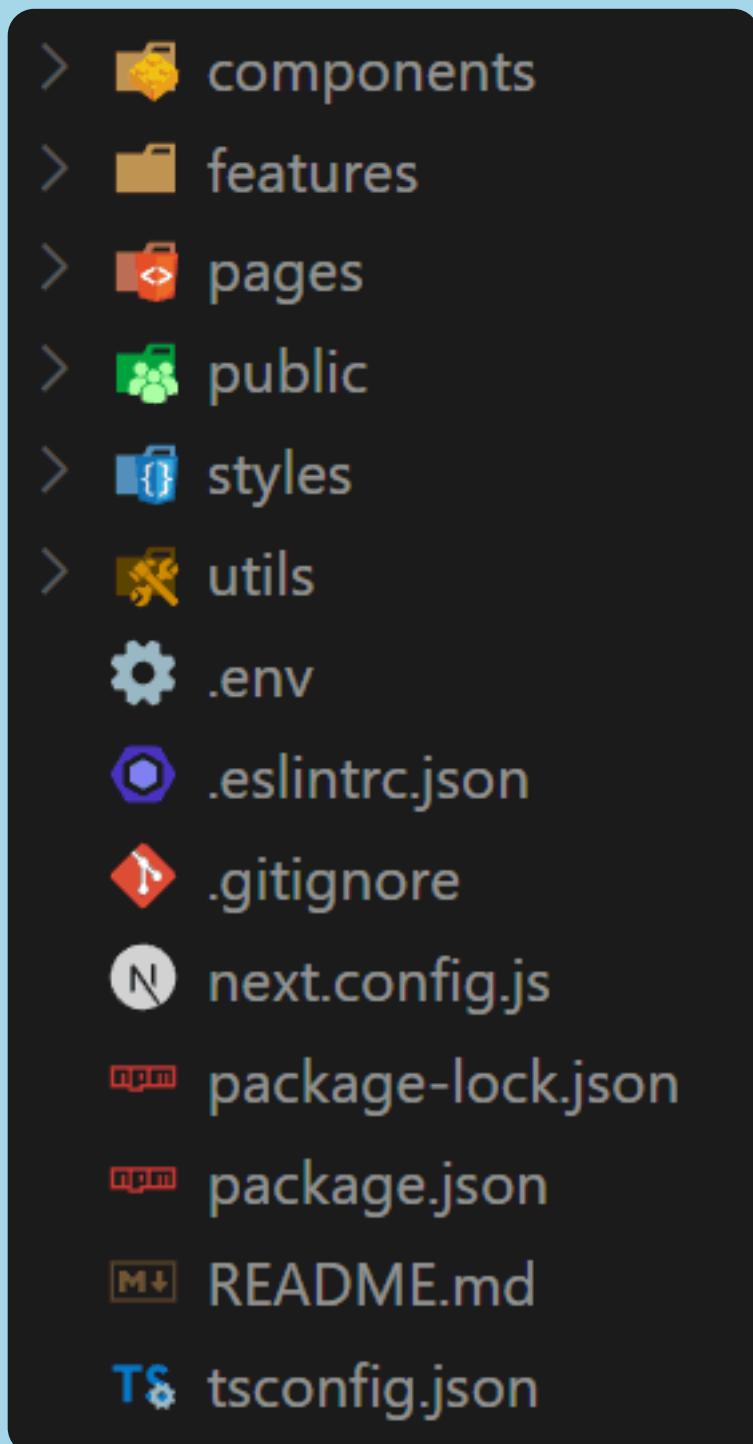
Dobrze przygotowana specyfikacja spowodowała, że nie miałem żadnych dodatkowych pytań podczas tworzenia projektu. Dużym plusem jest swoboda.

# Tłumaczenie

ad

## Project tree

Struktura folderów i projektu



Zmienne środowiskowe

Konfiguracja Gita

Konfiguracja NextJS

Biblioteki NodeJS

Opis projektu

Konfiguracja Typescript

\*Opisy folderów na kolejnych stronach

# Tłumaczenie

ad

## Struktura JSX

JSX to specjalny język rozumiany przez interpreter Reacta. Jest to mix Javascriptu (lub typescriptu - wtedy używamy rozszerzeń `.tsx`) z HTML.

Został stworzony przez Meta (Facebook), czyli autorów Reacta na potrzeby tej biblioteki.

Zawsze wymaga zakończenia tagu, nawet kiedy jest tylko pojedyńczy (np `<br>` musi mieć albo `</br>` lub `<br />`)

Kilka elementów na tym samym poziomie muszą być zagnieżdżone w rodzica.

Sposobem na ominięcie tego, jest zagnieżdżenie elementów w znacznik `<></>` który nie pełni żadnej innej funkcji

Do dodania klasy używamy atrybutu `className` zamiast `class`

W JSX możemy wykonać kod Javascript wewnątrz tagów. Używamy do tego `{ ... }`

# Tłumaczenie

ad

## Komponent

Komponent to funkcja Javascriptowa , którą używamy jako tagu XML.

Nazwy komponentów zaczynamy **zawsze** z dużej litery. W ten sposób interpreter rozróżnia wbudowane (z HTMLu) znaczniki od komponentów.

Komponenty mogą posiadać swoją logikę, będzie ona przed `return (...)`.

Komponenty mogą posiadać **własne atrybuty**. Tworzymy je poprzez argumenty w funkcji.



Text.tsx

```
import styles from './Text.module.css';

interface TextProps { // Typy
    title: string;
    children: React.ReactNode;
}

export const Text = ({ title, children }: TextProps) => {
// Tworzymy komponent <Text title="tytuł">nasza treść</Text>
    return (
        <div className={styles.text}>
            <h1>{title}</h1> /* Tutaj znajdzie się "tytuł" */
            {children} /* Tutaj znajdzie się "nasza treść" */
        </div>
    )
}
```

# Tłumaczenie

ad

## CSS modules

CSS modules w NextJS importujemy poprzez zimportowanie pliku jako *styles*.

Rozszerzenie plików to `.module.css`

## Folder *public*

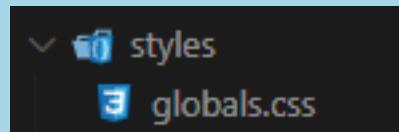
Publiczny folder, jego zawartość jest dostępna.

Struktura dostępu opiera się o foldery, w ten sam sposób jak w folderze “pages”.

## Folder *styles*

Folder do trzymania plików CSS.

Jedyna zawartość w tym projekcie to style globalne odnoszące się do każdej podstrony.



# Tłumaczenie

ad

## Folder *pages*

W folderze *pages* każdy komponent tworzy stronę, bazowaną na nazwie pliku o końcówce `.ts/.tsx` lub `.js/.jsx` lub folderze.

W podfolderze *api* możemy tworzyć **API** dla naszej aplikacji. Tutaj wszystkie strony są generowane po stronie serwera

└─ pages	
└─ api	
TS getItems.ts	/api/getItems
└─ kontakt	
JSX index.tsx	/kontakt
└─ login	
JSX index.tsx	/login
└─ o-nas	
JSX index.tsx	/o-nas
└─ produkty	
JSX index.tsx	/produkty
JSX _app.tsx	Pliki specjalne
JSX _document.tsx	
JSX index.tsx	/

# Tłumaczenie

ad

## Koncept produktów

Aby produkty były dostępne na wielu podstronach (główna w sekcji TopFive oraz produkty) zaimplementowałem API używając NextJS, które podczas zapytania pod `/api/getItems` zwraca JSON z produktami (odczytanymi z pliku).

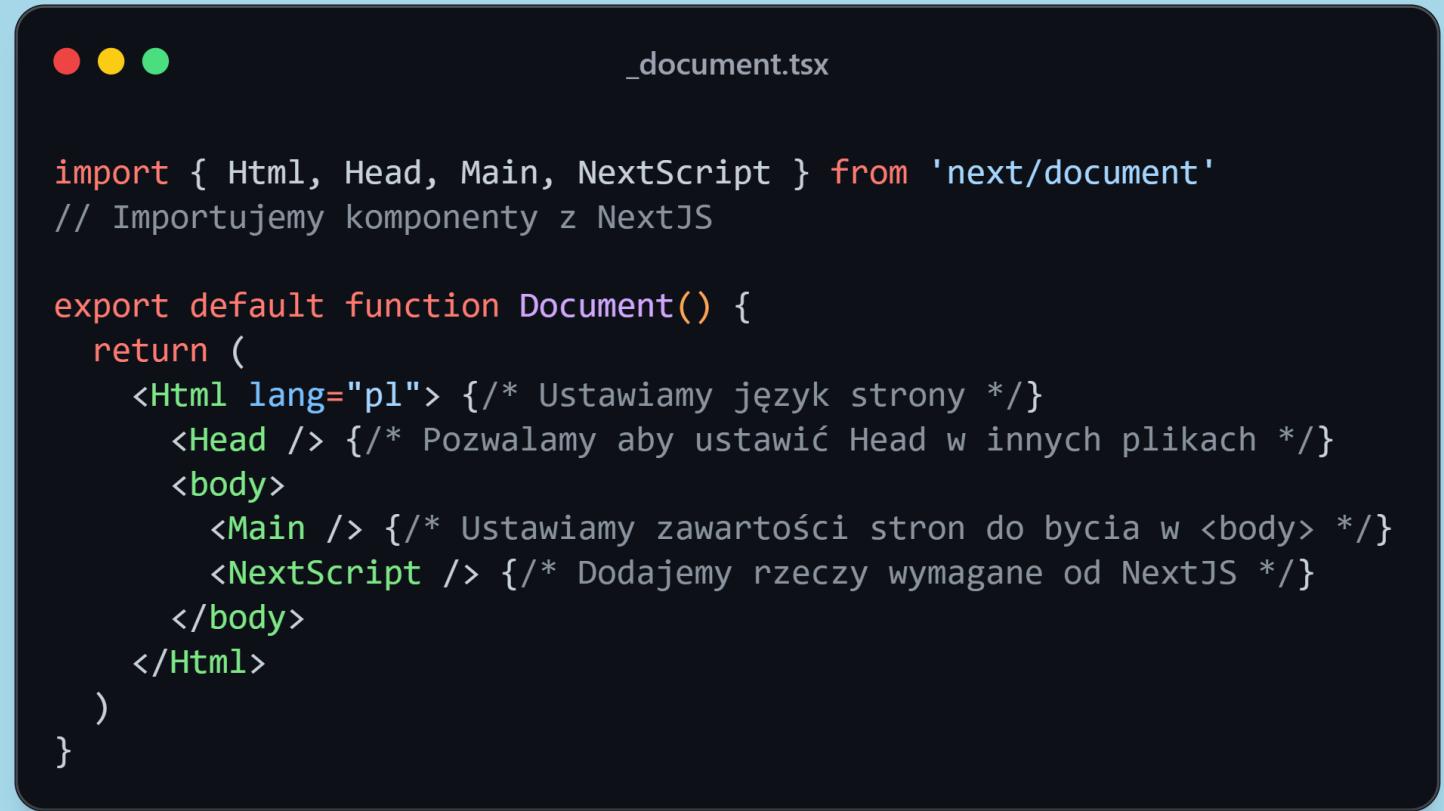
Położenie pliku JSON z produktami ustawiamy w zmiennych środowiskowych (env) w pliku `.env`

# Tłumaczenie

ad

## Plik `_document`

Jest to specjalny plik w folderze `pages` posiadający główną strukturę strony. Ten plik jest renderowany tylko po stronie serwera.



```
_document.tsx

import { Html, Head, Main, NextScript } from 'next/document'
// Importujemy komponenty z NextJS

export default function Document() {
  return (
    <Html lang="pl"> {/* Ustawiamy język strony */}
      <Head /> {/* Pozwalamy aby ustawić Head w innych plikach */}
      <body>
        <Main /> {/* Ustawiamy zawartości stron do bycia w <body> */}
        <NextScript /> {/* Dodajemy rzeczy wymagane od NextJS */}
      </body>
    </Html>
  )
}
```

Struktura tego pliku jest prawie niezmienna w każdym projekcie

# Tłumaczenie

ad

## Plik \_app

Jest to specjalny plik w folderze *pages* ustalający układ elementów na wszystkich stronach. Jest dokładniejszy od pliku *\_document*.

```
● ● ● _app.tsx

import { HamburgerProvider } from '@/components/Navbar/Hamburger/HamburgerContext'
import { Navbar } from '@/components/Navbar/Navbar'
import '@/styles/globals.css' // Globalny plik css
import type { AppProps } from 'next/app'
import { Lato } from 'next/font/google'

// Korzystamy z biblioteki next/font/google, aby łatwo zainportować czcionki z Google Fonts
const lato = Lato({
  subsets: ['latin'],
  weight: ['400', '700'],
})

export default function App({ Component, pageProps }: AppProps) {
  return (
    <HamburgerProvider>
      {/* Pakujemy całą zawartość strony w Provider stanie */}
      {/* dla Hamburger menu, aby wszędzie mieć dostępny status */}
      <main className={`${lato.className} main-container`} >
        {/* Dodajemy klasę z czcionką, oraz */}
        {/* globalną klasę main-container (bez użycia css-modules) */}
        <Navbar />
        <Component {...pageProps} /> {/* Renderujemy komponent strony */}
      </main>
    </HamburgerProvider>
  )
}
```

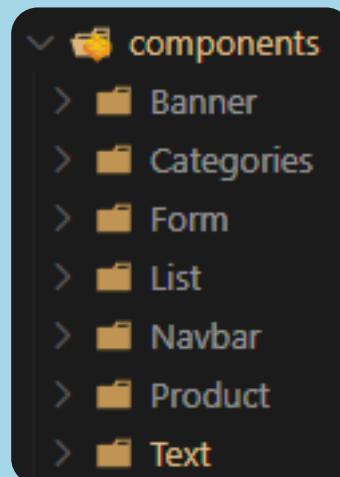
# Tłumaczenie

ad

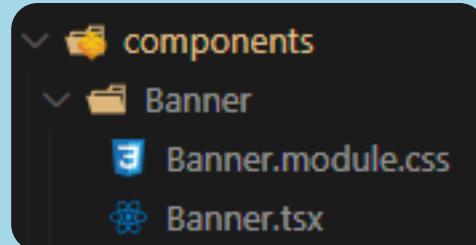
## Folder *components*

Ten folder nie jest tworzony przez NextJS  
(w odróżnieniu od poprzednich)

Folder składa się z podkatalogów nazwanych identycznie jak komponenty.

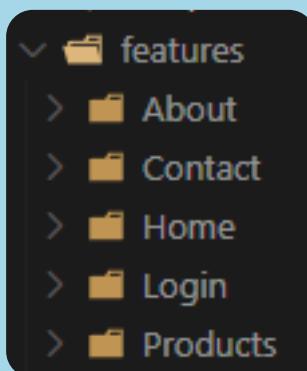
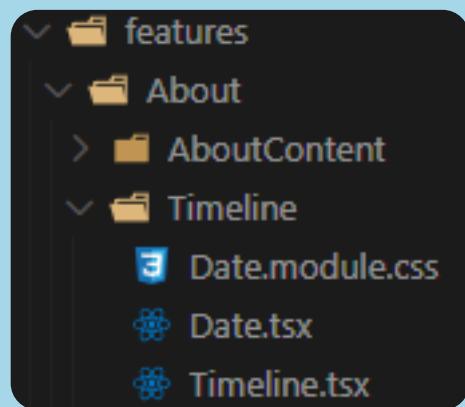


W środku nich znajduje się plik komponentu oraz arkusz stylów.



## Folder *features*

Folder jest bardzo podobny do *components*, jednak każda podstrona posiada osobny folder zawierające komponenty użyte tylko na tej stronie.

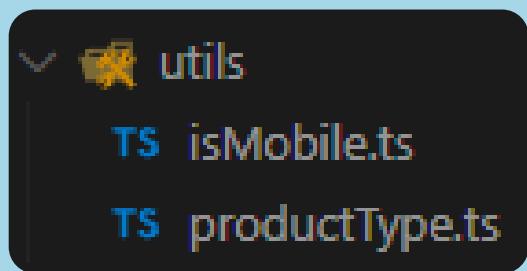


# Tłumaczenie

ad

## Folder *utils*

Jest to folder z innymi rzeczami nie związanymi w żaden sposób z komponentami lub nie pasujący do wcześniejszych folderów.



Zawiera on dwa pliki:

- `isMobile.ts` - reactowy hook do sprawdzenia rozdzielczości ekranu
- `productType.ts` - typy produktów

# Tłumaczenie

ad

## <Image>

To komponent z NextJS, jest rozszerzeniem dla <img> z HTML.

Służy do optymalizacji mediów na stronie oraz polepsza SEO strony oraz jej prędkość.

Główną z funkcji jest *lazy loading* która opóźnia ładowanie obrazów, dopóki użytkownik nie zechce zobaczyć ich (czyli dopóki element nie pojawi się na ekranie).

```
<Image  
  alt="Alternatywny tekst"  
  src="/sciezka/w/public"  
  width={150}  
  height={250}  
 />
```

To są wymagane argumenty. Istnieje wiele argumentów, które nie są wymagane.

`width` i `height` podajemy w nawiasach (ponieważ domyślnie jest liczbą, lecz można też jako tekst)

# Tłumaczenie Material UI Icons

To komponenty z biblioteki MUI  
importujące ikonu z Material UI

ad

```
//      Nazwa z dokumentacji      Stała ścieżka/Nazwa z dokumentacji
import PersonRoundedIcon from '@mui/icons-material/PersonRounded';

<PersonRoundedIcon
  fontSize='large'
  sx={{ style: 'z-css', kolejna: 'właściwość' }}
/>
```

W `sx` wpisujemy style CSS w obiekt  
Javascriptu. Zamiast myślników używamy  
typu pisania camelCase.

Wszystkie opcje są dokładnie opisane w  
dokumentacji biblioteki.

# Style

ad

## Globalne style



styles/global.css

```
:root {  
    /* Ustawiamy zmienne dla kazdego pliku */  
    --background-color: #1a1a1a;  
    --gray: #1E1E1E;  
    --light-gray: #929292;  
    --drop-shadow: drop-shadow(0 20px 13px rgb(0 0 0 / 0.03))  
                  drop-shadow(0 8px 5px rgb(0 0 0 / 0.08));  
    --orange: #FF5A00;  
    --border: 1px solid var(--light-gray);  
}  
  
/*  
Aby objąć każdy nadrzędny element strony  
Odnoszę się do każdego po kolejni  
*/  
html,  
body,  
body > div:first-child,  
div#__next,  
div#__next > div {  
    background-color: var(--background-color);  
    margin: 0;  
    color-scheme: dark;  
    position: relative;  
}  
  
/* Wyłączam style przeglądarki  
dla każdego <a>  
*/  
a {  
    color: white;  
    text-decoration: none;  
}  
  
/* Ustawiamy szerokość  
oraz pozycje dla każdej  
strony */  
.main-container {  
    width: 80%;  
    margin-right: auto;  
    margin-left: auto;  
}
```

# Style

ad

## Globalne style



styles/global.css

```
/* Nakładamy blur podczas
otwartego hamburger menu */
.blur {
    filter: blur(5px);
}

/* Klasa każdego *Content komponentu */
.container {
    margin-top: 50px;
    display: flex;
    justify-content: space-between;
}

.container h1 {
    font-size: 40px;
}

@media (max-width: 700px) {
    /* Zmieniamy ułożenie .container */
    .container {
        flex-wrap: wrap;
        justify-content: center;
    }
}
```

# Strony

# ad

## Strona Główna /



pages/index.tsx

```
import { HamburgerContext } from '@/components/Navbar/Hamburger/HamburgerContext';
import { HomeContent } from '@/features/Home/HomeContent/HomeContent'
import Head from 'next/head' // Zaimportowanie Head - odpowiednika <head> z HTML
import { useContext } from 'react';

export default function Home() {
  const { isOpen } = useContext(HamburgerContext); // Odczytujemy status Hamburger menu
  return (
    <div className={isOpen ? 'blur' : ''}> /* Dodajemy klasę blur, gdy Hamburger menu jest otwarte */
      <Head>
        <title>Alledrogo</title>
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/branding/logo-transparent.svg" />
        <meta name="description" content="Najmniejszy sklep z największym dropshipem!" />
        {/* Dodajemy metadane do strony */}
      </Head>
      <HomeContent /> /* Resztę strony robimy w komponentie HomeContent */
    </div>
  )
}
```



HomeContent.tsx

```
import { Categories } from "@/components/Categories/Categories"
import { TopFive } from "../TopFive/TopFive"
import styles from './HomeContent.module.css'
import { Banner } from "../../../../../components/Banner/Banner"

export const HomeContent = () => {
  return (
    <div className={styles.container}>
      <Categories />
      <div>
        <TopFive />
        <div className={styles.banners}>
          <Banner href="/o-nas" special>O NAS</Banner>
          {/* Dodajemy atrybut special aby zmienić kolor */}
          <Banner href="/produkty">PRODUKTY</Banner>
          <Banner href="/kontakt">KONTAKT</Banner>
        </div>
      </div>
    </div>
  )
}
```

# Strony

## Strona Główna /

ad



components/Banner/Banner.tsx

```
import Link from 'next/link' // Zainportowanie Link - odpowiednika <a> z HTML
import styles from './Banner.module.css'

// Definiujemy typy argumentów, które przyjmie komponent
interface BannerProps {
    // Atrybut special jest opcjonalny
    special?: boolean,
    href: string,
    // children to wszystko, co znajduje się wewnątrz komponentu
    children: React.ReactNode
}

export const Banner = ({ special, href, children }: BannerProps) => {
    return (
        <Link href={href}>
            {/* Sprawdzamy czy jest atrybut special i dodajemy klase .special */}
            <div className={`${styles.banner} ${special && `${styles.special}`}`}>
                <h1>{children}</h1>
            </div>
        </Link>
    )
}
```

## Tłumaczenie

### <Link>

Jest to alternatywa dla tagu <a> w HTML.

Przy przejściu na inną stronę, jedynym co jest pobierane jest plik Javascriptu ładujący nam stronę. Podczas przejścia dlatego nie widać żadnego ładowania, w porównaniu do czystego HTML

Nazwa	Stan	Typ	Inicjator	Rozmiar	Czas	Wodospad
kontaktjs	200	script	route-loader.js:93	16.3 kB	8 ms	

# Style

ad

## Strona Główna /

HomeContent.module.css

```
.banners {  
    display: flex;  
    justify-content: space-between;  
    flex-wrap: wrap;  
}  
  
@media (max-width: 987px) {  
    /* Zmieniamy bannery na kolumnę */  
    .banners {  
        flex-direction: column;  
        align-items: center;  
    }  
}
```

● ● ●

Banner.module.css

```
.banner {  
    width: 250px;  
    padding: 20px;  
    border-radius: 15px;  
    border: 1px solid var(--light-gray);  
    background-color: var(--gray);  
    color: var(--light-gray);  
    text-align: center;  
    filter: var(--drop-shadow);  
    margin-bottom: 30px;  
}  
  
.special {  
    /* Jeżeli banner jest specjalny  
    zmieniamy kolor i border na pomarańczowy*/  
    border: 1px solid var(--orange);  
    color: var(--orange);  
}
```

# Strony

## Strona Główna / TopFive

ad



features/Home/TopFive/TopFive

```
import { Product } from '@/components/Product/Product';
import { useEffect, useState } from 'react';
import styles from './TopFive.module.css';
import axios from 'axios'; // Importujemy biblioteki do komunikacji z API
import { ProductList } from '@/utils/productType';

export const TopFive = () => {
    // Zapisujemy produkty w stanie
    const [products, setProducts] = useState<ProductList>()
    useEffect(() => {
        // Robimy zadanie GET do API
        axios.get(`api/getItems`)
            .then((res) => {
                // Zapisujemy produkty
                setProducts(res.data)
            })
        // Pusta tablica jako drugi argument useEffecta powoduje, że
        // useEffect wykona się tylko raz, gdy komponent się zamontuje
    }, [])
    // Wybieramy 5 produktów
    const topFive = products?.products.slice(0, 5)

    return (
        <div className={styles.container}>
            <h1>Najlepsze produkty</h1>
            <div className={styles.products}>
                {/* Dla każdego produktu dodajemy komponent Product */}
                {topFive?.map((product) => {
                    // Klucz jest wymagany przy mapowaniu przez reacta
                    return <Product title={product.name} price={product.price} image={product.image} key={product.name} />;
                })}
            </div>
        </div>
    )
}
```

# Style

## Strona Główna

/

ad

```
TopFive.module.css

.container {
    padding: 20px;
    background-color: var(--gray);
    border: var(--border);
    filter: var(--drop-shadow);
    border-radius: 30px;
    color: white;
    margin-bottom: 50px;
}

.container h1 {
    margin-top: 0;
    font-size: 40px;
}

.products {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
}

@media screen and (max-width: 987px) {
    /* Na telefonie zmieniamy wiersz
    na kolumnę */
    .container {
        text-align: center;
    }
    .products {
        flex-direction: column;
    }
}
```

# Tłumaczenie

ad

## useEffect

useEffect to funkcja która wykonuje się po wyrenderowaniu komponentu

Domyślnie wykonuje się cały czas w nieskończoność. Jeżeli dodamy pustą tablice, funkcja wykona się tylko raz (po wyrenderowaniu się komponentu).

Jeżeli dodamy jakąś zmienną do tablicy, useEffect wykona się za każdym razem kiedy zmieni się zmienna.

# Strony

# Strona Główna /

## Categories

```
components/Categories/Categories.tsx

import { List } from '@/components/List/List';
import { isMobile } from '@/utils/isMobile';
import { useEffect, useState } from 'react';
import styles from './Categories.module.css';

export const Categories = () => {
    // Zapisujemy czy jesteśmy na urządzeniu mobilnym
    const [mobile, setMobile] = useState(false)
    useEffect(() => {
        // Sprawdzamy czy jesteśmy na urządzeniu mobilnym
        setMobile(isMobile(window))
    })

    // Jeśli jesteśmy na urządzeniu mobilnym
    // nie wyświetlamy kategorie (są w menu)
    if (mobile) return (<></>)
    return (
        <div className={styles.container}>
            <h1>Kategorie</h1>
            <List icon='home' />
            <List icon='electronics' />
            <List icon='clothes' />
        </div>
    )
}
```

Sprawdzamy czy jesteśmy na urządzeniu mobilnym, ponieważ kategorie posiadamy w hamburger menu.



Categories.module.css

```
.container {  
    color: white;  
    background-color: var(--gray);  
    border: var(--border);  
    border-radius: 30px;  
    filter: var(--drop-shadow);  
    padding: 20px;  
    width: fit-content;  
    min-width: 220px;  
    height: fit-content;  
    margin-right: 15px;  
}  
  
.container h1 {  
    margin-top: 0;  
    font-size: 40px;  
}
```

# Tłumaczenie

ad

## isMobile

isMobile to własny *hook*, czyli sposób na wyodrębnienie logiki z komponentu do osobnej funkcji.



utils/isMobile.ts

```
export const isMobile = (window: Window) => {
    // pobieramy window (argument HTML)
    const windowWidth = window.innerWidth
    // pobieramy szerokość okna

    if (windowWidth < 987) {
        // jeśli szerokość okna jest mniejsza niż 987px
        // to przełączamy się na wersję mobilną
        return true
    }
    return false
}
```

Używam sprawdzenia po stronie Javascriptu, a nie @media w kluczowych miejscach z powodu tego że rozdzielcość wyświetlacza na stronie nie zmienia się w trakcie korzystania.

Jeżeli jest taka potrzeba, wystarczy odświeżyć stronę i kluczowe miejsca się zmienią automatycznie.

# Strony

## Produkty /produkty

ad

pages/produkty/index.tsx wygląda tak samo jak w przypadku głównej strony



features/Products/ProductsContent.tsx

```
import { Categories } from '@/components/Categories/Categories'
import { ProductsList } from '../ProductsList/ProductsList'

export const ProductsContent = () => {
  return (
    <div className="container">
      <Categories />
      <ProductsList />
    </div>
  )
}
```



features/Products/ProductsList/ProductsList.tsx

```
import { Product } from '@/components/Product/Product"
import { ProductList } from '@/utils/productType"
import axios from "axios"
import { useEffect, useState } from "react"
import styles from "./ProductsList.module.css"

export const Productslist = () => {
  // Dokładnie to samo co w wypadku TopFive
  const [products, setProducts] = useState<ProductList>()
  useEffect(() => {
    axios.get(`api/getItems`)
      .then((res) => {
        setProducts(res.data)
      })
  }, [])
  // Wybieramy wszystkie produkty
  return (
    <div className={styles.container}>
      <h1>Produkty</h1>
      <div className={styles.products}>
        { products?.products.map((product) => {
          return <Product title={product.name} price={product.price} image={product.image} key={product.name} />;
        })}
      </div>
    </div>
  )
}
```

## Produkty /produkty



Products.module.css

```
.container {  
    padding: 20px;  
    background-color: var(--gray);  
    border: var(--border);  
    filter: var(--drop-shadow);  
    border-radius: 30px;  
    color: white;  
    margin-bottom: 50px;  
}  
  
.container h1 {  
    margin-top: 0;  
    font-size: 40px;  
}  
  
.products {  
    display: flex;  
    flex-wrap: wrap;  
    justify-content: space-around;  
}  
  
@media screen and (max-width: 987px) {  
    /* Na telefonie zmieniamy wiersz  
    na kolumnę */  
    .container {  
        text-align: center;  
    }  
    .products {  
        flex-direction: column;  
    }  
}
```

# Strony

## O nas /o-nas

ad

pages/o-nas/index.tsx wygląda tak samo jak w przypadku głównej strony



features/About/AboutContent/AboutContent.tsx

```
import { Text } from '@/components/Text/Text'
import { Timeline } from '../Timeline/Timeline'

export const AboutContent = () => {
  return (
    <div className="container">
      <div>
        <Text title='O nas'>
          Jesteśmy najmniejszym sklepem bez zarejestrowanej firmy.
          Wszystko co robimy jest nielegalne i nie akceptujemy zwrotów.
          Większość produktów to dropship z aliexpress lub taobao.
        </Text>
      </div>
      <Timeline />
    </div>
  )
}
```

# Komponenty

## Timeline

ad



features/About/Timeline/Timeline.tsx

```
import { Date } from './Date'

export const Timeline = () => {
  return (
    // Tworzymy nieugrupowaną listę z datami
    <ul>
      <Date timestamp="Gdzieś po 1999">Wyśmiewanie Allegro jako Aledrogo</Date>
      <Date timestamp="15 marca 2023">Otrzymanie specyfikacji od Pana Kacpra Ślipko</Date>
      <Date timestamp="Dzisiaj">Sprawdzenie pracy Bartosza Sułkowskiego</Date>
    </ul>
  )
}
```



features/About/Timeline/Date.tsx

```
import { ReactNode } from 'react'
import styles from './Date.module.css'

interface DateProps {
  timestamp: string,
  children: ReactNode
}

export const Date = ({ timestamp, children }: DateProps) => {
  return (
    <li className={styles.li}>
      <p className={styles.timestamp}>{timestamp}</p>
      <p className={styles.description}>{children}</p>
    </li>
  )
}
```

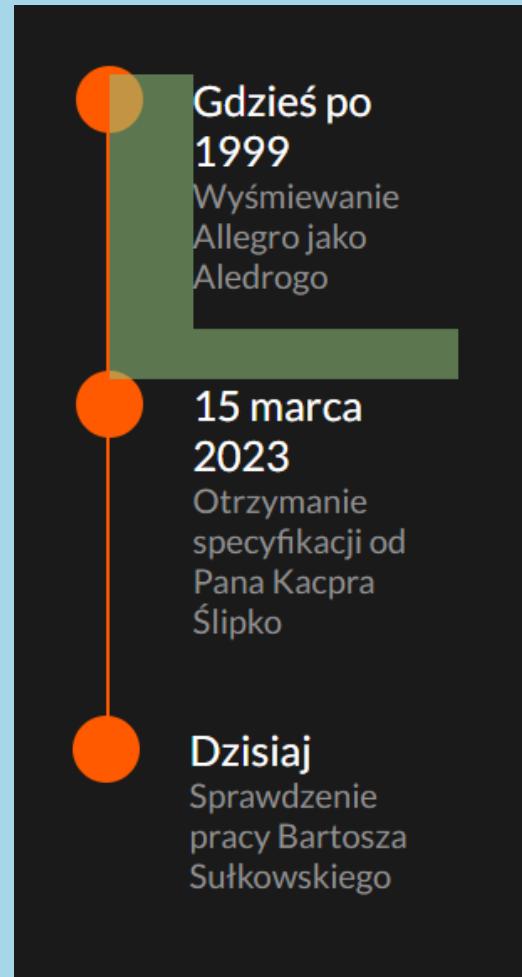
# Style

## O nas /o-nas



Date.module.css

```
.li {  
    /* usuwamy domyślny styl list */  
    list-style-type: none;  
    /* ustawiamy position relative  
    aby użyć potem absolute */  
    position: relative;  
    /* Ustawiamy odstęp, aby zrobić linie */  
    padding: 0 0 30px 50px;  
    /* Ustawiamy linie łączącą elementy */  
    border-left: 2px solid var(--orange);  
}  
  
.li:last-child {  
    /* Usuwamy linię dla  
    ostatniego elementu */  
    border: 0;  
}  
  
.li:before {  
    /* Tworzymy "kropkę" */  
    content: "";  
    background-color: var(--orange);  
    position: absolute;  
    top: -5px;  
    left: -20px;  
    width: 40px;  
    height: 40px;  
    border-radius: 30px;  
}  
  
.timestamp {  
    font-size: 25px;  
    color: white;  
    margin: 0;  
}  
  
.description {  
    margin-top: 0;  
    font-size: 20px;  
    color: var(--light-gray);  
}
```



# Strony

## Kontakt /kontakt

ad

pages/kontakt/index.tsx wygląda tak samo jak w przypadku głównej strony



features/Contact/ContactContent/ContactContent.tsx

```
import { Form } from '../../components/Form/Form';
import { Text } from '@/components/Text/Text';

export const ContactContent = () => {
  return (
    <div className="container">
      <Text title='Kontakt'>
        <p>Nie mam zielonego pojęcia po co do nas pisać,  
bo i tak ci nie odpowiemy :)</p>
        <p>Ale dla tych co myślą że to zrobimy,  
jedyny sposób to formularz, a twoja wiadomość  
poleci w przodzie, tam szukaj odpowiedzi.</p>
      </Text>
      <Form type='contact' />
    </div>
  )
}
```

# Komponenty

ad

## Form

```
components/Form/Form.tsx

import styles from './Form.module.css';
import { FormInput } from './FormInput';

interface FormProps {
    // Type może być tylko 'contact', 'login' lub 'register'
    type: 'contact' | 'login' | 'register';
}

export const Form = ({ type }: FormProps) => {
    return (
        <div className={styles.form}>
            {/* Sprawdzamy czy 'contact' */}
            {type === 'contact' && <>
                <h1>Kontakt</h1>
                <FormInput type='name' />
                <FormInput type='text' />
                <FormInput type='submit' text='WYŚLIJ' />
            </>}
            {/* Sprawdzamy czy 'login' */}
            {type === 'login' && <>
                <h1>Zaloguj się</h1>
                <FormInput type='login' />
                <FormInput type='password' />
                <FormInput type='submit' text='ZALOGUJ' />
            </>}
            {/* Sprawdzamy czy 'register' */}
            {type === 'register' && <>
                <h1>Zarejestruj się</h1>
                <FormInput type='login' />
                <FormInput type='password' />
                <FormInput type='repeatPassword' />
                <FormInput type='submit' text='ZALOGUJ' />
            </>}
        </div>
    )
}
```

Sprawdzamy (za pomocą składni JSX w środku kodu) jaki jest type i tworzymy odpowiedni formularz.

# Style

# ad

## Form



Form.module.css

```
.form {  
  padding: 20px;  
  background-color: var(--gray);  
  border: var(--border);  
  filter: var(--drop-shadow);  
  border-radius: 30px;  
  color: white;  
  width: 30%;  
  font-size: 20px;  
  text-align: center;  
  min-width: 200px;  
  height: fit-content;  
  margin-bottom: 50px;  
}  
  
.form h1 {  
  margin-top: 0;  
  font-size: 40px;  
  text-align: start;  
}
```

# Komponenty

ad

## FormInput



components/Form/FormInput.tsx

```
import styles from './FormInput.module.css';

interface FormInputProps {
  type: 'name' | 'text' | 'submit' | 'login' | 'password' | 'repeatPassword',
  // Ustawiamy na opcjonalne
  text?: string
}

export const FormInput = ({ type, text }: FormInputProps) => {
  return (
    <>
      /* robimy sprawdzenie dla każdej opcji type */
      { type === 'name' && <input type="text" placeholder="Imię" className={styles.text} /> }
      { type === 'login' && <input type="text" placeholder="Login" className={styles.text} /> }
      { type === 'password' && <input type="password" placeholder="Hasło" className={styles.text} /> }
      { type === 'repeatPassword' && <input type="password" placeholder="Powtórz hasło" className={styles.text} /> }
      { type === 'text' && <textarea placeholder="Treść" className={styles.textBox} /> }
      { type === 'submit' && <input type="submit" className={styles.submit} value={text} /> }
    </>
  )
}
```



FormInput.module.css

```
.text {
  width: 80%;
  background-color: transparent;
  border: 1px solid var(--light-gray);
  border-radius: 15px;
  padding: 10px 20px;
  font-size: 25px;
  margin-bottom: 20px;
}

.textBox {
  width: 80%;
  background-color: transparent;
  border: 1px solid var(--light-gray);
  border-radius: 15px;
  padding: 10px 20px;
  font-size: 25px;
  margin-bottom: 20px;
  /* blokujemy rozszerzanie dla
   * textarea */
  resize: none;
}

.submit {
  width: 50%;
  background-color: transparent;
  border: 1px solid var(--orange);
  border-radius: 15px;
  padding: 10px 20px;
  font-size: 25px;
  color: var(--orange);
  font-weight: bold;
  cursor: pointer;
  min-width: 150px;
  margin-bottom: 20px;
}
```

# Strony

## Login /login

ad

pages/login/index.tsx wygląda tak samo jak w przypadku głównej strony



features/Login/LoginContent/LoginContent.tsx

```
import { Form } from "@/components/Form/Form"

export const LoginContent = () => {
  return (
    <div className="container">
      {/* Tworzymy login */}
      <Form type="login" />
      {/* Tworzymy rejestracje */}
      <Form type="register" />
    </div>
  )
}
```

# Strony

## ad

### API /api/getItems

Do zbudowania API używamy języka JS/TS,  
a nie tak jak wcześniej JSX/TSX.

```
pages/api/getItems.ts

import type { NextApiRequest, NextApiResponse } from 'next'
import { promises as fs } from 'fs';
import path from 'path';

type Data = {
  name: string
}

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse<Data>
) {
  // Zapisywanie ścieżki do folderu public
  const publicDirectory = path.join(process.cwd(), 'public');
  // Odczyt pliku JSON (z publicDirectory + ścieżka do pliku JSON z ENV)
  const fileContent = await fs.readFile(publicDirectory + process.env.NEXT_PUBLIC_JSON_DIRECTORY, 'utf8');
  // Zwrócenie JSONa
  res.status(200).json(JSON.parse(fileContent))
}
```

Biblioteka fs i path są wbudowane w Nodejs. Pozwalają na odnoszenie się do plików.

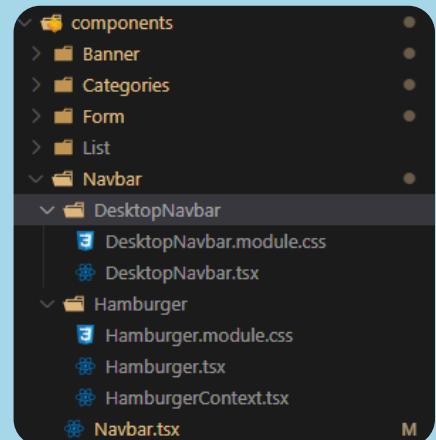
Zwracamy JSONa (produkty) z statusem 200 OK

# Navbar

ad

## Navbar.tsx

To będzie sekcja poświęcona Nabarowi (i Hamburegr menu) z powodu złożoności



components/Navbar/Navbar.tsx

```
import { isMobile } from "@/utils/isMobile"
import { useEffect, useState } from "react"
import { DesktopNavbar } from "./DesktopNavbar/DesktopNavbar"
import { Hamburger } from "./Hamburger/Hamburger"

export const Navbar = () => {
    // Sprawdzamy, czy jesteśmy na urządzeniu mobilnym
    const [mobile, setMobile] = useState(false)
    useEffect(() => {
        setMobile(isMobile(window))
    })

    return (
        <>
        {/* Jeżeli jesteśmy na urządzeniu mobilnym dajemy Hamburger */}
        {/* Inaczej dajemy Navbar */}
        {
            mobile ? <Hamburger /> : <DesktopNavbar />
        }
    </>
)
}
```

# Navbar

ad

## Desktop Navbar



components/Navbar/DesktopNavbar/DesktopNavbar.tsx

```
import styles from './DesktopNavbar.module.css'
import Image from 'next/image'
import Link from 'next/link'
import LoginRoundedIcon from '@mui/icons-material/LoginRounded';
// Importujemy ikony z Material UI

export const DesktopNavbar = () => {
    return (
        <nav className={styles.navbar}>
            {/* Banner odnosi do home page (/) */}
            <Link href="/">
                <Image src={'branding/banner-transparent.svg'} alt="Alledrogo" width={256} height={64} />
            </Link>
            <ul className={styles.list}>
                <li><Link href={'/o-nas'}>O nas</Link></li>
                <li><Link href={'/produkty'} className={styles.margins}>Produkty</Link></li>
                <li><Link href={'/kontakt'}>Kontakt</Link></li>
            </ul>
            {/* Przycisk zaloguj odnosi do login page (/login) */}
            <Link href="/login">
                <div className={styles.login}>
                    <p>Zaloguj się</p>
                    <LoginRoundedIcon fontSize='large' sx={{ color: '#FF5A00', marginLeft: '10px' }}/>
                </div>
            </Link>
        </nav>
    )
}
```



DesktopNavbar.module.css

```
.navbar {
    background-color: var(--gray);
    height: 100px;
    border: 1px solid var(--light-gray);
    border-radius: 30px;
    /* centerujemy navbar */
    margin-right: auto;
    margin-left: auto;
    margin-top: 20px;
    filter: var(--drop-shadow);
    /* centerujemy horyzontalnie */
    display: flex;
    align-items: center;
    padding: 5px 30px;
    justify-content: space-between;
    font-size: 25px;
}

.list {
    display: flex;
    list-style-type: none;
    padding: 0;
}
```



DesktopNavbar.module.css

```
.login {
    color: var(--orange);
    display: flex;
    align-items: center;
}

.margins {
    /* robimy jeden element
    z marginesami */
    --margins: 20px;
    margin-right: var(--margins);
    margin-left: var(--margins);
}
```

# Navbar

## Hamburger

ad



Hamburger.tsx

```
import styles from './Hamburger.module.css';
import MenuIcon from '@mui/icons-material/Menu';
import CloseRoundedIcon from '@mui/icons-material/CloseRounded';
import LoginRoundedIcon from '@mui/icons-material/LoginRounded';
import { useContext } from 'react';
import { List } from '@/components/List/List';
import { HamburgerContext } from './HamburgerContext';
import Link from 'next/link';
import Image from 'next/image';

export const Hamburger = () => {
    // Odczytujemy i zapisujemy stan Hamburger menu
    const { isOpen, toggle } = useContext(HamburgerContext);

    return (
        <>
        {
            isOpen
            ?
            // Jeśli Hamburger menu jest otwarte, renderujemy jego zawartość
            <nav className={styles.menu}>
                <div className={styles.icons}>
                    <Link href="/">
                        <Image alt="home logo" src="/branding/logo-transparent.svg" width={64} height={64} />
                    </Link>
                    <Link href="/login">
                        <LoginRoundedIcon sx={{ color: '#FF5A00', fontSize: 64 }} />
                    </Link>
                    {/* Po kliknięciu zamkniemy Hamburger menu */}
                    <CloseRoundedIcon sx={{ color: '#FF5A00', fontSize: 64 }} onClick={toggle} />
                </div>
                <h1>Kategorie</h1>
                <List icon='home' />
                <List icon='electronics' />
                <List icon='clothes' />
                <List icon='about' />
                <List icon='products' />
                <List icon='contact' />
                <List icon='login' />
            </nav>
            :
            // Jeśli Hamburger menu jest zamknięte, renderujemy tylko ikonę
            // po kliknięciu na którą otwieramy Hamburger menu
            <nav className={styles.icon} onClick={toggle}>
                <MenuIcon fontSize='large' sx={{ color: '#FF5A00' }}/>
            </nav>
        }
    )
}
```

# Navbar

## Style Hamburger

ad



Hamburger.module.css

```
.icon {  
    padding: 10px;  
    background-color: var(--gray);  
    border: var(--border);  
    border-radius: 15px;  
    width: fit-content;  
    /* ustawiamy stałą pozycję na ekranie */  
    position: fixed;  
    top: 15px;  
    left: 15px;  
    filter: var(--drop-shadow);  
    /* ustawiamy ikonę na sam wierzch */  
    z-index: 999;  
}  
  
.menu {  
    height: 100vh;  
    width: 55vw;  
    background-color: var(--gray);  
    border-right: var(--border);  
    color: white;  
    padding: 20px;  
    /* ustawiamy stałą pozycję na ekranie */  
    position: fixed;  
    top: 0;  
    left: 0;  
    /* ustawiamy menu na sam wierzch strony */  
    z-index: 999;  
    /* ✨ wywołujemy animacje otwierania ✨ */  
    animation-duration: 0.5s;  
    animation-name: slidein;  
}
```



Hamburger.module.css

```
.menu h1 {  
    font-size: 35px;  
}  
  
.icons {  
    display: flex;  
    justify-content: space-between;  
}  
  
/* ✨ animacja otwierania ✨ */  
@keyframes slidein {  
    /* zaczynamy z lewej strony */  
    from {  
        left: -250px;  
    }  
    /* konczymy na domyślnej pozycji */  
    to {  
        left: 0;  
    }  
}
```

# Navbar

ad

## Hamburger Context

To komponent użyty w `_app.tsx` do przekazania innym plikom stanu (i opcji edycji) otwarcia Hamburger menu.



HamburgerContext.tsx

```
import { PropsWithChildren, createContext, useState } from "react";

// Tworzymy interfejs dla kontekstu
export interface HamburgerContextData {
  isOpen: boolean;
  toggle: () => void;
}

// Tworzymy kontekst dla komponentu Hamburger
export const HamburgerContext = createContext<HamburgerContextData>({
  isOpen: false,
  toggle: () => {},
});

export const HamburgerProvider = ({ children }: PropsWithChildren) => {
  // Tworzymy stan dla Hamburger menu
  const [isOpen, setIsOpen] = useState(false);

  // Tworzymy funkcję, która zmienia stan Hamburger menu
  const toggle = () => {
    setIsOpen(!isOpen);
  };

  return (
    // Tworzymy provider dla kontekstu
    <HamburgerContext.Provider value={{ isOpen, toggle }}>
      {children}
    </HamburgerContext.Provider>
  );
};
```

# Komponenty

## List

ad

```
components/List/List.tsx

import HomeRoundedIcon from '@mui/icons-material/HomeRounded';
import PhoneAndroidRoundedIcon from '@mui/icons-material/PhoneAndroidRounded';
import CheckroomRoundedIcon from '@mui/icons-material/CheckroomRounded';
import LocalMallRoundedIcon from '@mui/icons-material/LocalMallRounded';
import CallRoundedIcon from '@mui/icons-material/CallRounded';
import styles from './List.module.css'
import PersonRoundedIcon from '@mui/icons-material/PersonRounded';
import LoginRoundedIcon from '@mui/icons-material/LoginRounded';
import Link from 'next/link';

interface ListProps {
    icon: 'home' | 'electronics' | 'clothes' | 'about' | 'products' | 'contact' | 'login'
}

export const List = ({ icon }: ListProps) => {
    return (
        // sprawdzamy czy icon jest równy 'about' lub 'login' i jeśli tak to dodajemy klasę margin
        // sprawdzamy czy icon jest równy 'login' i jeśli tak to dodajemy klasę login
        <div className={`${styles.list}`}
        ${icon === 'about' || icon === 'login' ? styles.margin : ''} ${icon === 'login' ? styles.login : ''}>
            /* wykonujemy sprawdzanie dla każdego icon */
            { icon === "home"
                && <>
                    <HomeRoundedIcon fontSize='large'/>
                    <Link href="/produkty">Dla Domu</Link>
            </>
            }
            { icon === "electronics"
                && <>
                    <PhoneAndroidRoundedIcon fontSize='large'/>
                    <Link href="/produkty">Elektronika</Link>
            </>
            }
            { icon === "clothes"
                && <>
                    <CheckroomRoundedIcon fontSize='large'/>
                    <Link href="/produkty">Moda</Link>
            </>
            }
            { icon === "about"
                && <>
                    <PersonRoundedIcon fontSize='large'/>
                    <Link href="/o-nas">O nas</Link>
            </>
            }
            { icon === "products"
                && <>
                    <LocalMallRoundedIcon fontSize='large'/>
                    <Link href="/produkty">Produkty</Link>
            </>
            }
            { icon === "contact"
                && <>
                    <CallRoundedIcon fontSize='large'/>
                    <Link href="/kontakt">Kontakt</Link>
            </>
            }
            { icon === "login"
                && <>
                    <LoginRoundedIcon fontSize='large'/>
                    <Link href="/login" className={styles.login}>Zaloguj się</Link>
            </>
            }
        </div>
    );
};
```

# Style

## List

ad



List.module.css

```
.list {  
    display: flex;  
    color: white;  
    font-size: 25px;  
    margin-left: 10px;  
    margin-top: 8px;  
}  
  
.list a {  
    margin-left: 10px;  
    margin-top: auto;  
    margin-bottom: auto;  
}  
  
.margin {  
    margin-top: 20px;  
}  
  
.login {  
    color: var(--orange);  
}
```

# Komponenty

## Product

ad



components/Product/Product.tsx

```
import styles from './Product.module.css';
import Image from 'next/image';
import Link from 'next/link';

interface ProductProps {
    title: string;
    price: string;
    image: string;
}

export const Product = ({ title, price, image }: ProductProps) => {
    // dzielimy cenę na złotówki i grosze
    const bigPrice = price.split('.')[0];
    const smallPrice = price.split('.')[1];

    return (
        <Link href="#">
            <div className={styles.product}>
                <Image alt={title} src={`/products/${image}`} width={150} height={250} className={styles.image} />
                {/* Grosze wyświetlamy jako mniejsze od złotówek */}
                <h2>{bigPrice},<span className={styles.small_price}>{smallPrice}</span> zł</h2>
                <p>{title}</p>
            </div>
        </Link>
    )
}
```

● ● ● Product.module.css

```
.product {
    text-align: center;
    padding: 0 10px;
}

.product h2 {
    color: var(--orange);
    font-size: 30px;
}

.image {
    /* zmniejszamy zdjęcie
    aby zawsze miało takie
    same wymiary, ale nie
    było ucięte */
    object-fit: contain;
}

.small_price {
    font-size: 20px;
}
```