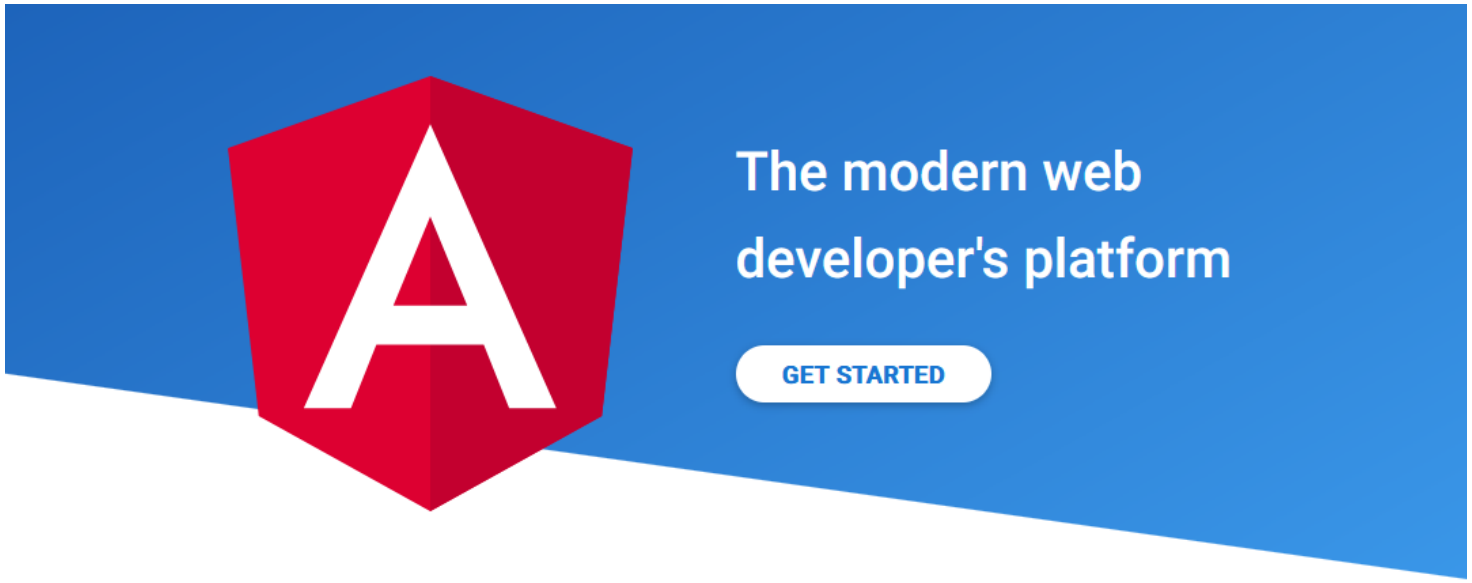
Angular

wersje: 14, 18, 20, 21

tutorial version 2.19

Angular

wspierana przez Google otwarta biblioteka JS napisana w języku TypeScript



DEVELOP ACROSS ALL PLATFORMS

Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

https://angular.io/

Angular

framework SPA (Single Page Aplikation) -
główne okno aplikacji ładowane jest tylko raz,
odświeżane są fragmenty strony bez ponownego
ładowania całej strony

posiada własny kompilator HTML
(można tworzyć własne znaczniki)

kurs Angulara
(YouTube)

korzysta z języka TypeScript

wdraża wzorzec projektowy
MVC (Model-View-Controller)
przy tworzeniu aplikacji webowych

na Angularze opiera się m.in. framework IONIC
do budowy aplikacji mobilnych

platforma programistyczna

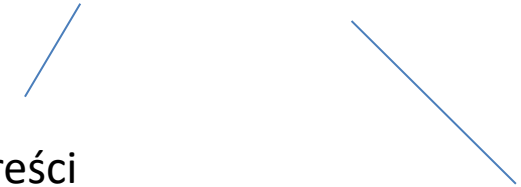
Getting started with Angular

Component

jest częścią UI (User Interface)

element składowy aplikacji w Angularze

komponent



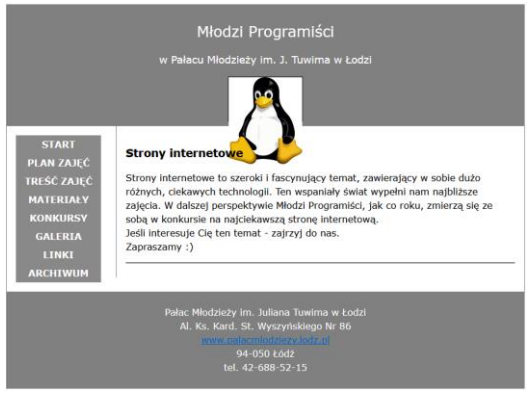
służy do przekazywania treści użytkownikowi oraz do interakcji z nim

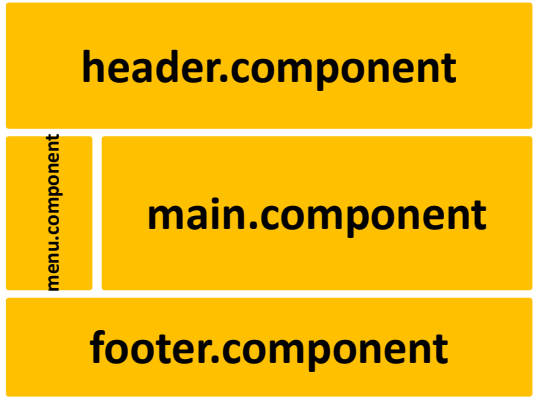
składa się z klasy (logika i dane), pliku HTML (szablonu), pliku stylów oraz ma dostęp do fragmentu strony internetowej, gdzie wyświetla treści.

Component

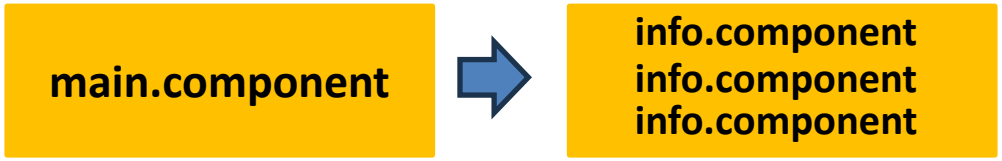
jeden komponent

kilka komponentów





kilka komponentów

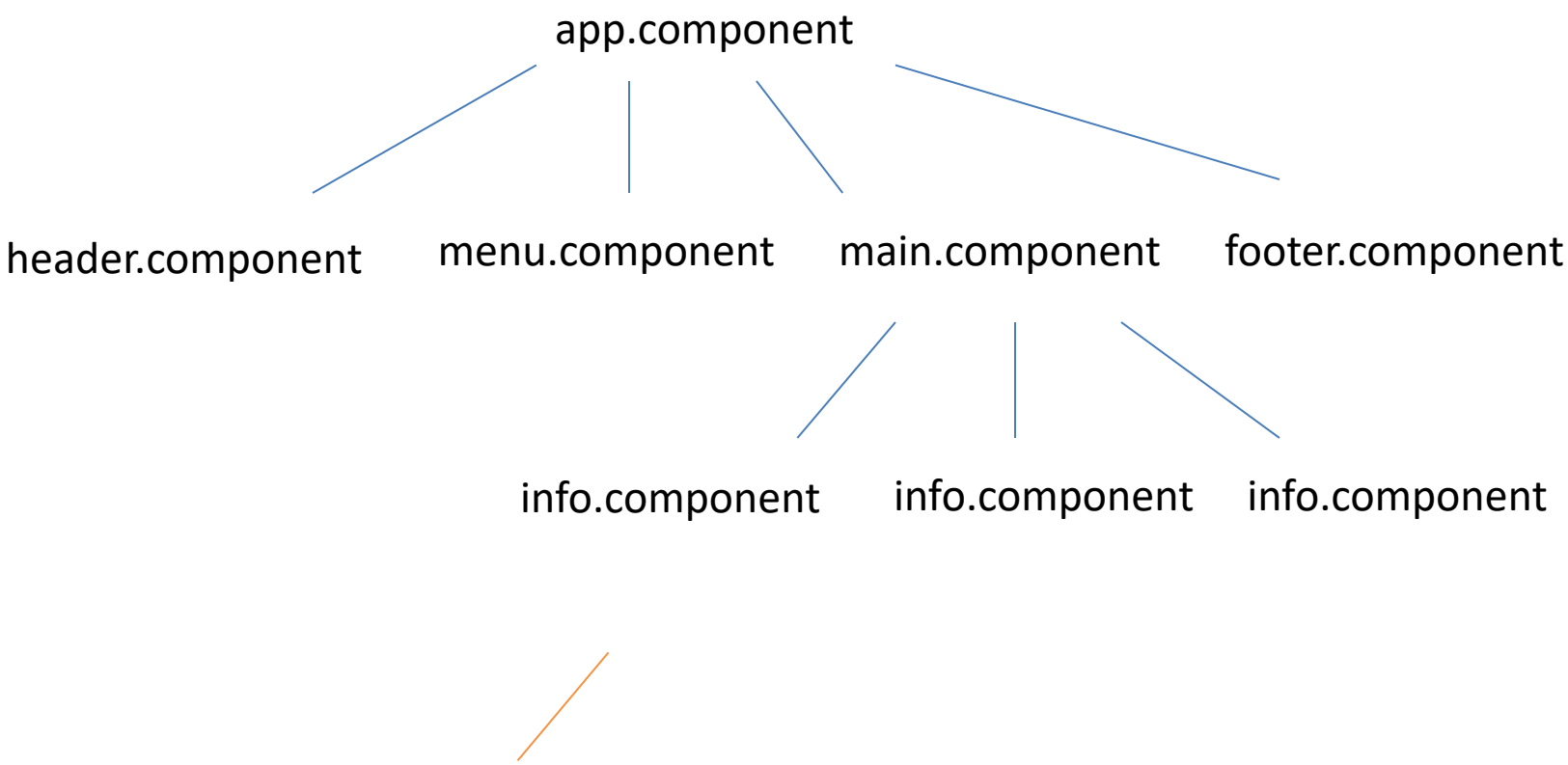


aplikacja składa się z komponentów, można różnie ją zaprojektować

[Understanding Modules with Angular]

Component

aplikacja posiada co najmniej jeden komponent główny app.component



komponenty aplikacji tworzą drzewo komponentów

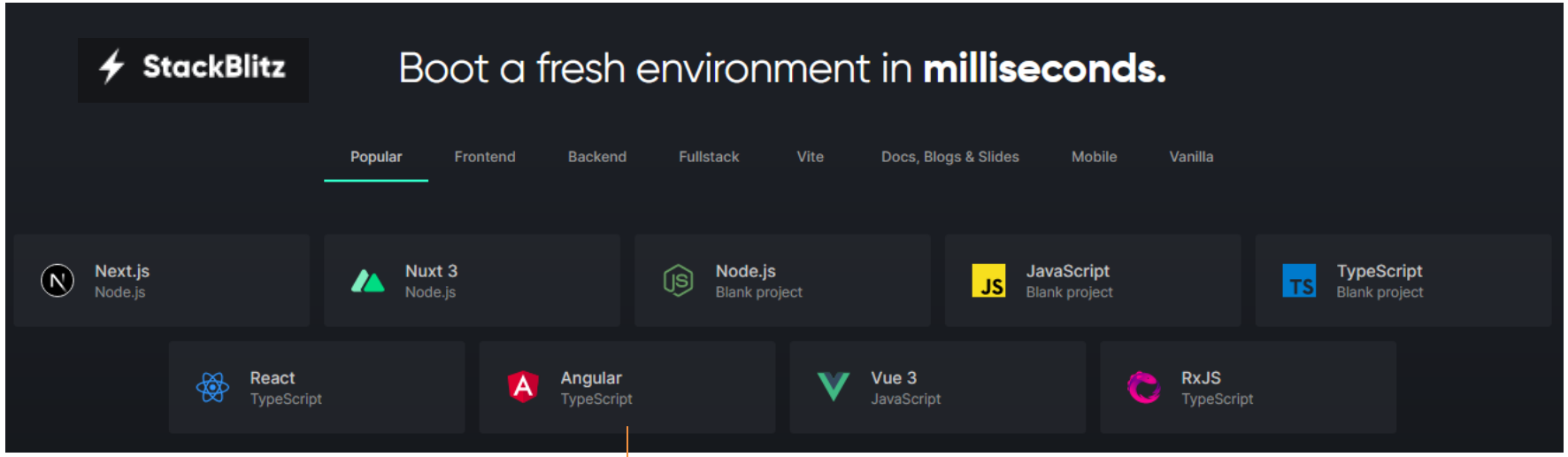
Komponenty samodzielne (standalone)

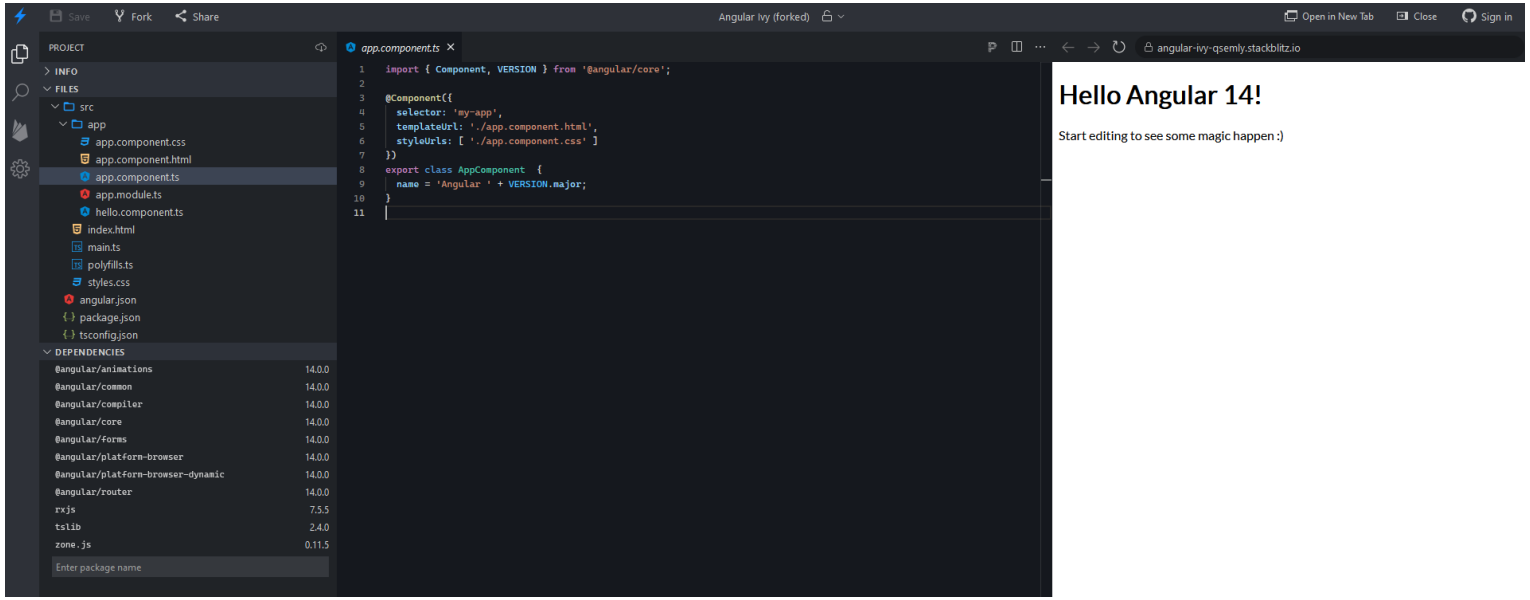
Komponenty, dyrektywy i potoki można oznaczać jako ***samodzielne* (*standalone: true*).**
Klasy Angular oznaczone jako samodzielne nie muszą być deklarowane w NgModule (kompilator Angular zgłosi błąd, jeśli je spróbujemy zadeklarować w ten sposób).

Samodzielne komponenty określają swoje zależności bezpośrednio, zamiast pobierać je za pośrednictwem NgModules.
Na przykład, jeśli X jest samodzielnym komponentem, może bezpośrednio zaimportować inny samodzielny komponent Y

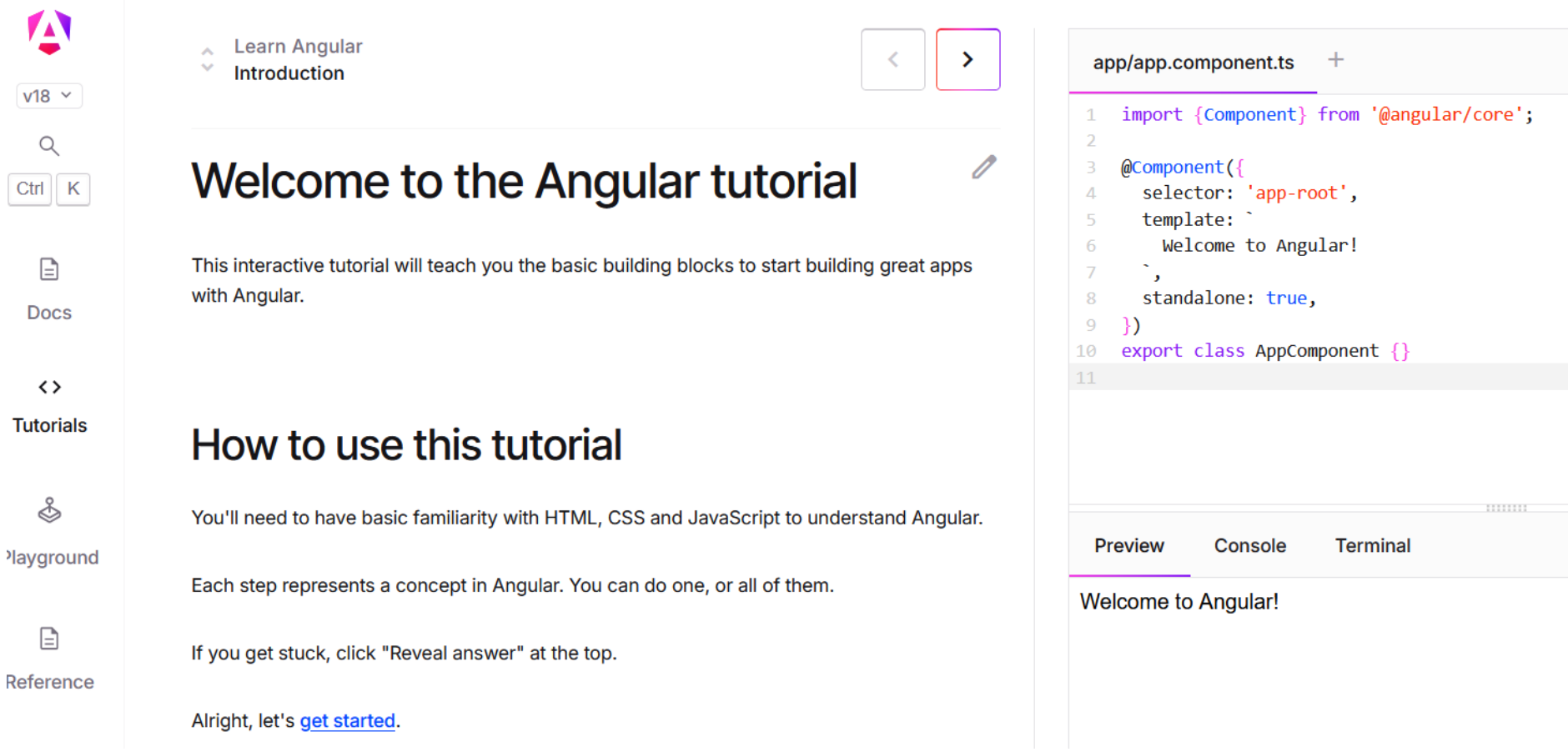
https://angular.io/guide/standalone-components

StackBlitz – Angular on-line





Angular tutorial



https://angular.dev/tutorials/learn-angular

Instalacja Node.js

Node.js oraz menadżer pakietów npm

https://nodejs.org/en/

sprawdzenie zainstalowanej wersji

```
C:\>node -v
v16.18.0

C:\>npm -v
8.19.2
```

wersja bierząca

Long Term Support
długi okres wsparcia technicznego

Instalacja Angular CLI

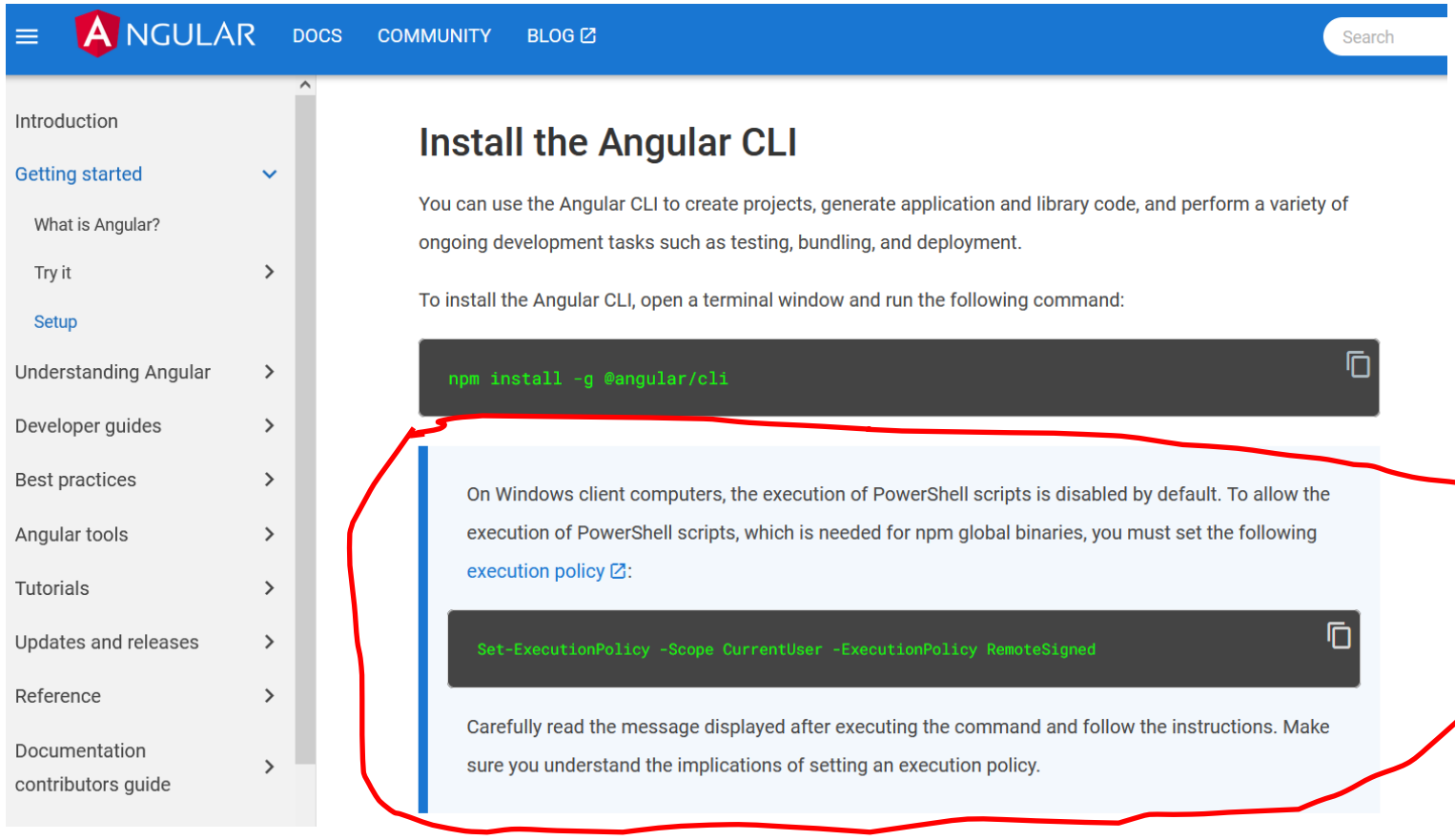
Angular CLI to narzędzie interfejsu wiersza poleceń do inicjalizowania, rozwijania i utrzymywania aplikacji.

CMD

```
C:\>npm install -g @angular/cli
```

Angular instalowany globalnie (dodanie wpisu do PATH)

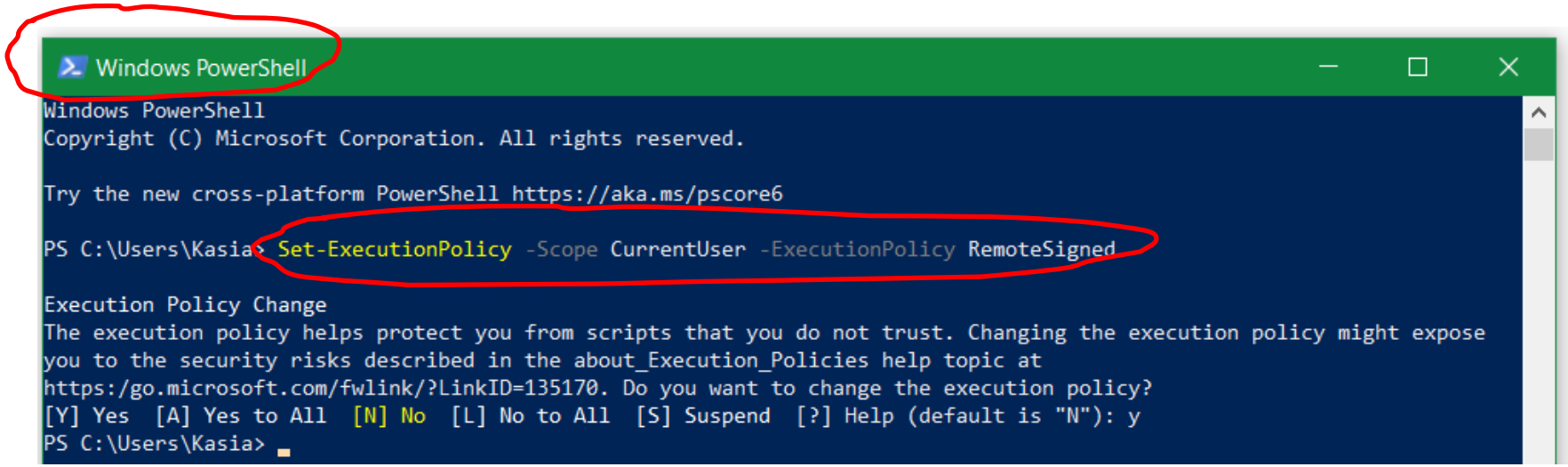
Instalacja Angular CLI



https://angular.io/guide/setup-local

w systemie Windows należy włączyć możliwość wykonywania skryptów w PowerShellu

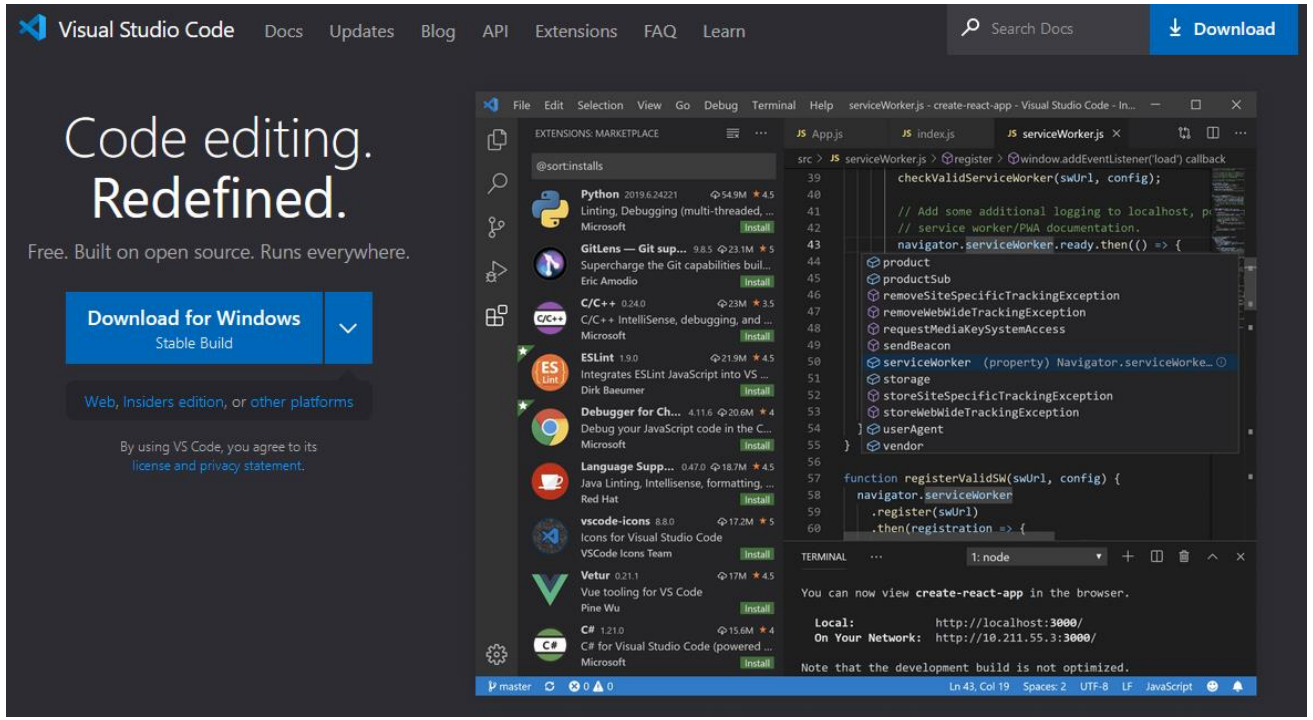
Instalacja Angular CLI

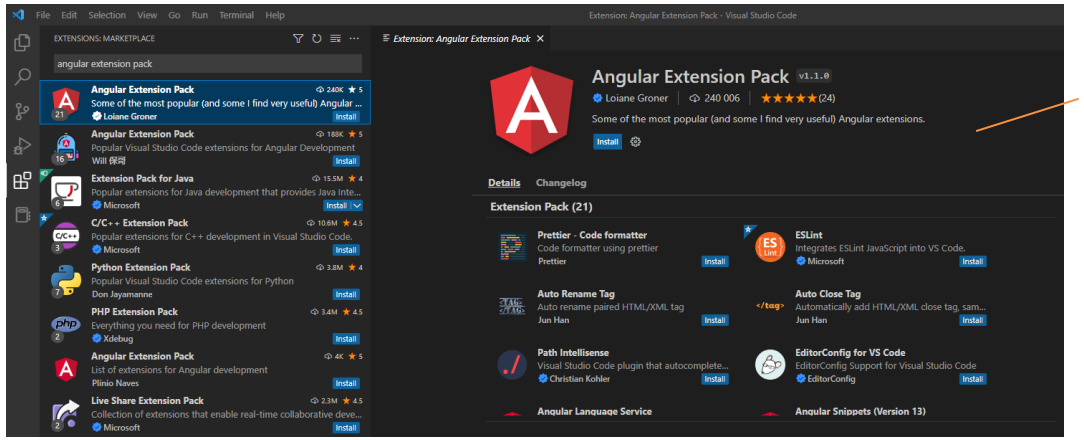


komendę wykonujemy w PowerShellu, a nie w cmd

Instalacja Visual Studio Code

https://code.visualstudio.com/

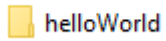




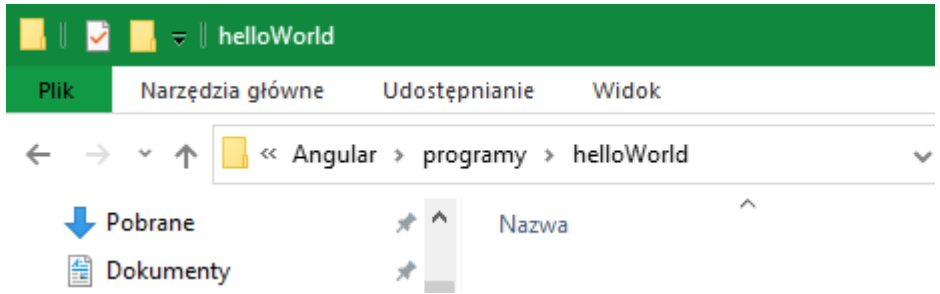
dodatek Angular Extension Pack

Tworzenie nowego projektu

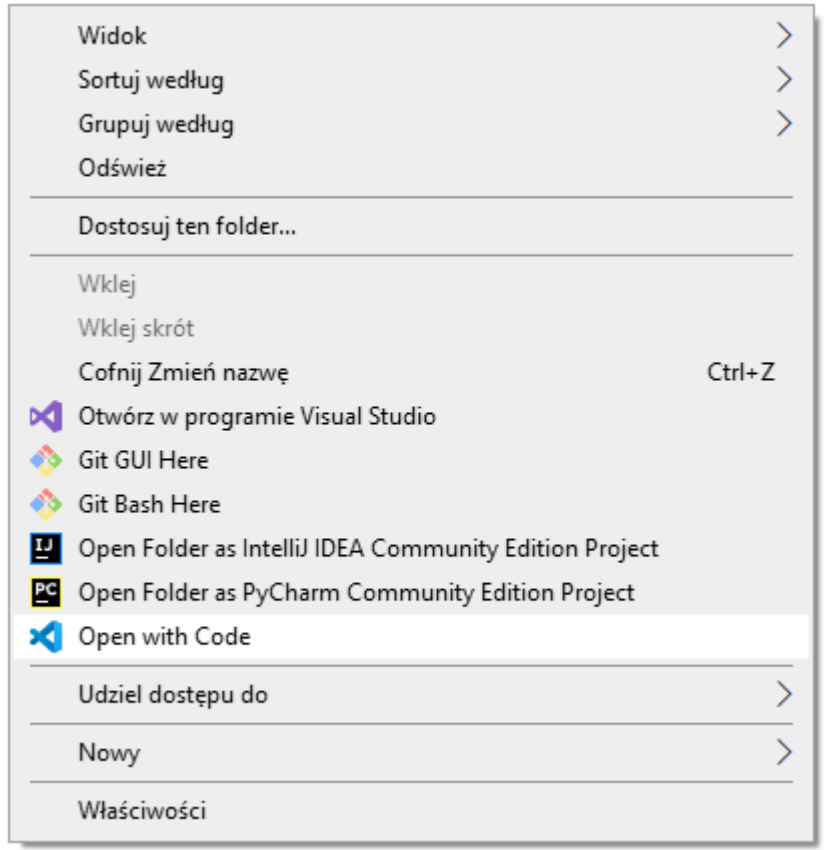
tworzymy folder helloWorld (nie jest to konieczne, nowy projekt utworzy nowy folder)



otwieramy folder helloWorld

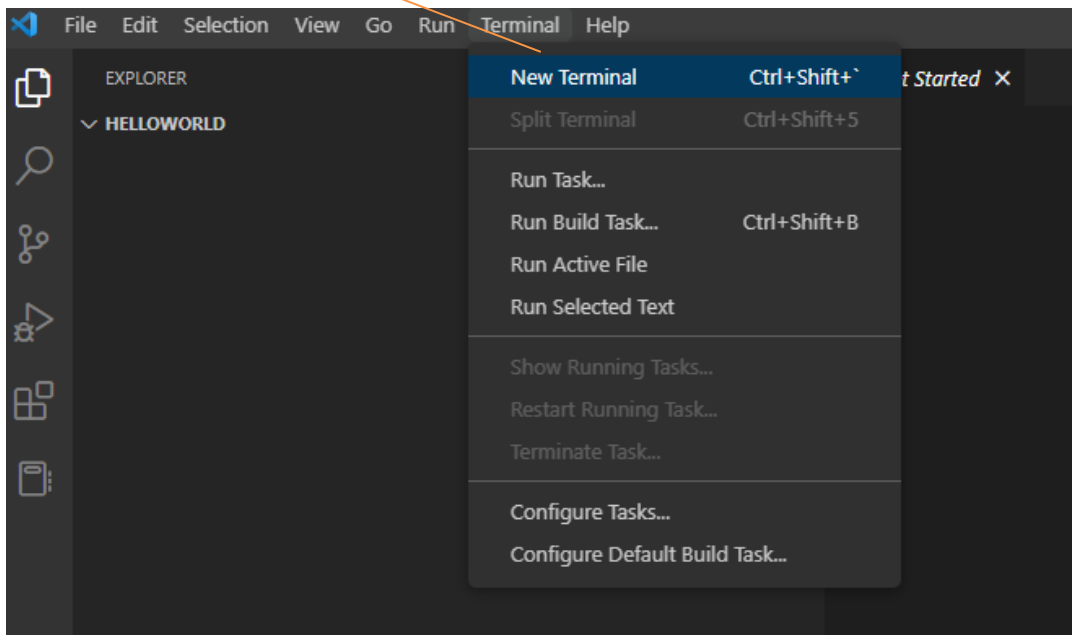


wewnątrz folderu helloWorld
prawy przycisk myszy/Open with Code



Tworzenie nowego projektu

w Visual Studio Code w katalogu projektu otwieramy nowy terminal



Tworzenie nowego projektu - komendy

w terminalu wpisujemy polecenie utworzenia aplikacji o nazwie **myHelloWorld** (z komponentami typu standalone - samodzielne)

```
ng new myHelloWorld
```

wybieramy CSS

```
? Which stylesheet format would you like to use? (Use arrow keys)
❯ CSS
  SCSS   [ https://sass-lang.com/documentation/syntax#scss                ]
  Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less   [ http://lesscss.org                                             ]
```

wybieramy y

```
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation
(SSG/Prerendering)? (y/N)
```

Server-Side Rendering – renderowanie po stronie serwera
Static Site Generation – generowanie stycznych stron w trakcie budowania projektu

obie technologie przyspieszają działanie strony

ng new

Tworzenie nowego projektu

```
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. No
Global setting: disabled
Local setting: No local workspace configuration file.
Effective status: disabled
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE myHelloWorld/angular.json (2954 bytes)
CREATE myHelloWorld/package.json (1045 bytes)
CREATE myHelloWorld/README.md (1066 bytes)
CREATE myHelloWorld/tsconfig.json (863 bytes)
CREATE myHelloWorld/.editorconfig (274 bytes)
CREATE myHelloWorld/.gitignore (548 bytes)
CREATE myHelloWorld/.browserslistrc (600 bytes)
CREATE myHelloWorld/karma.conf.js (1431 bytes)
CREATE myHelloWorld/tsconfig.app.json (287 bytes)
CREATE myHelloWorld/tsconfig.spec.json (333 bytes)
CREATE myHelloWorld/.vscode/extensions.json (130 bytes)
CREATE myHelloWorld/.vscode/launch.json (474 bytes)
CREATE myHelloWorld/.vscode/tasks.json (938 bytes)
CREATE myHelloWorld/src/favicon.ico (948 bytes)
CREATE myHelloWorld/src/index.html (298 bytes)
CREATE myHelloWorld/src/main.ts (372 bytes)
CREATE myHelloWorld/src/polyfills.ts (2338 bytes)
CREATE myHelloWorld/src/styles.css (80 bytes)
CREATE myHelloWorld/src/test.ts (749 bytes)
CREATE myHelloWorld/src/assets/.gitkeep (0 bytes)
⠋ Installing packages (npm)...
✓ Packages installed successfully.
```

instalowanie pakietów

Tworzenie nowego projektu

po pomyślnym zainstalowaniu pakietów przechodzimy do nowo utworzonego katalogu projektu **myHelloWorld** oraz uruchamiamy serwer:

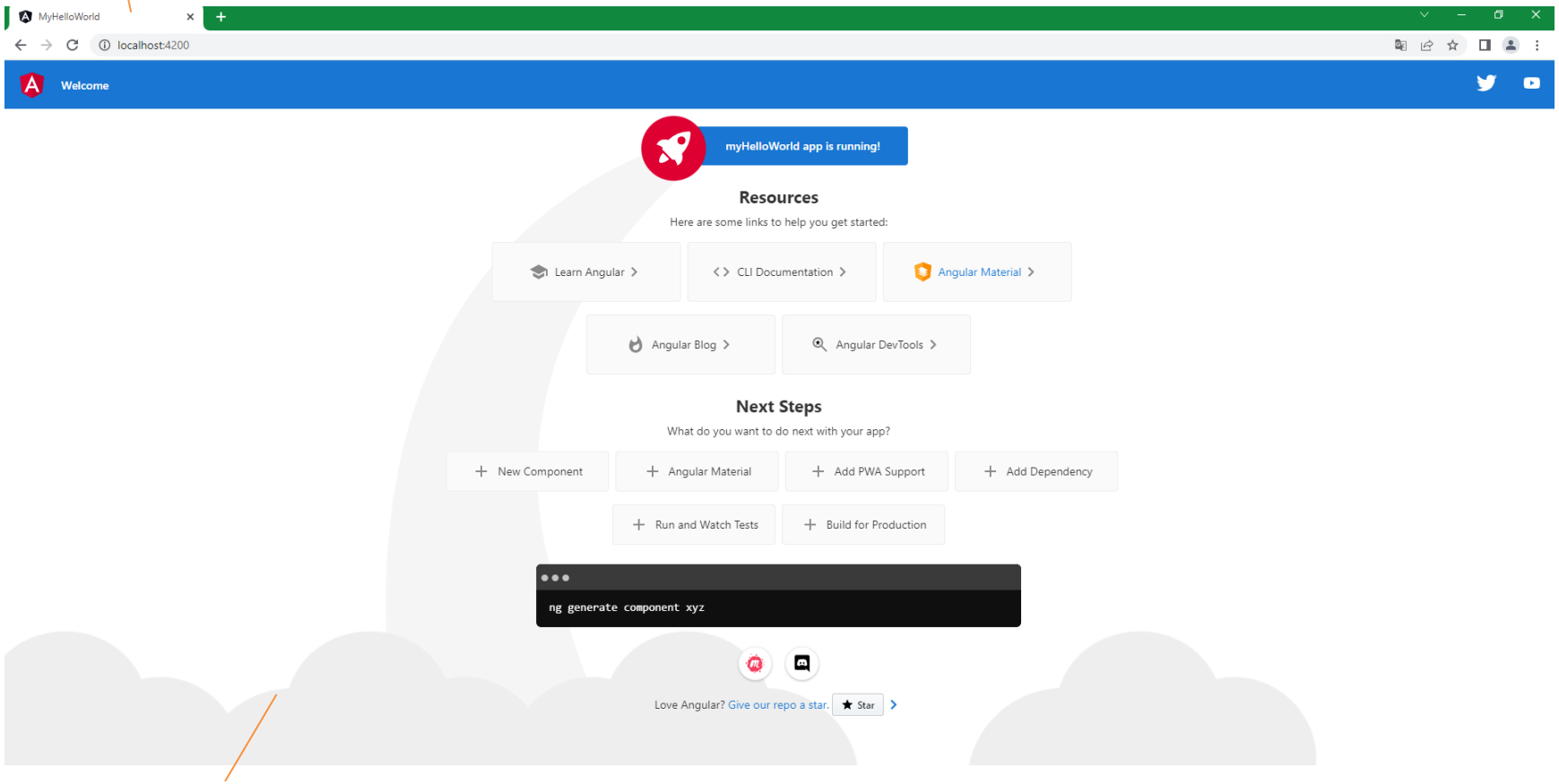
```
cd .\myHelloWorld\          // przejście do katalogu projektu
ng serve                    // uruchomienie serwera
```

```
PS D:\helloWorld> cd .\myHelloWorld\
PS D:\helloWorld\myHelloWorld> ng serve
✓ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   1.77 MB |
polyfills.js          | polyfills     | 318.06 kB |
styles.css, styles.js | styles        | 210.09 kB |
main.js               | main          |  48.05 kB |
runtime.js            | runtime       |   6.52 kB |

                      | Initial Total |   2.34 MB

Build at: 2022-10-24T12:44:09.270Z - Hash: 33e6ac1e37b11bad - Time: 80208ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


✓ Compiled successfully.
```

został uruchomiony Live Development Server nasłuchujący na adresie localhost:4200

Tworzenie nowego projektu

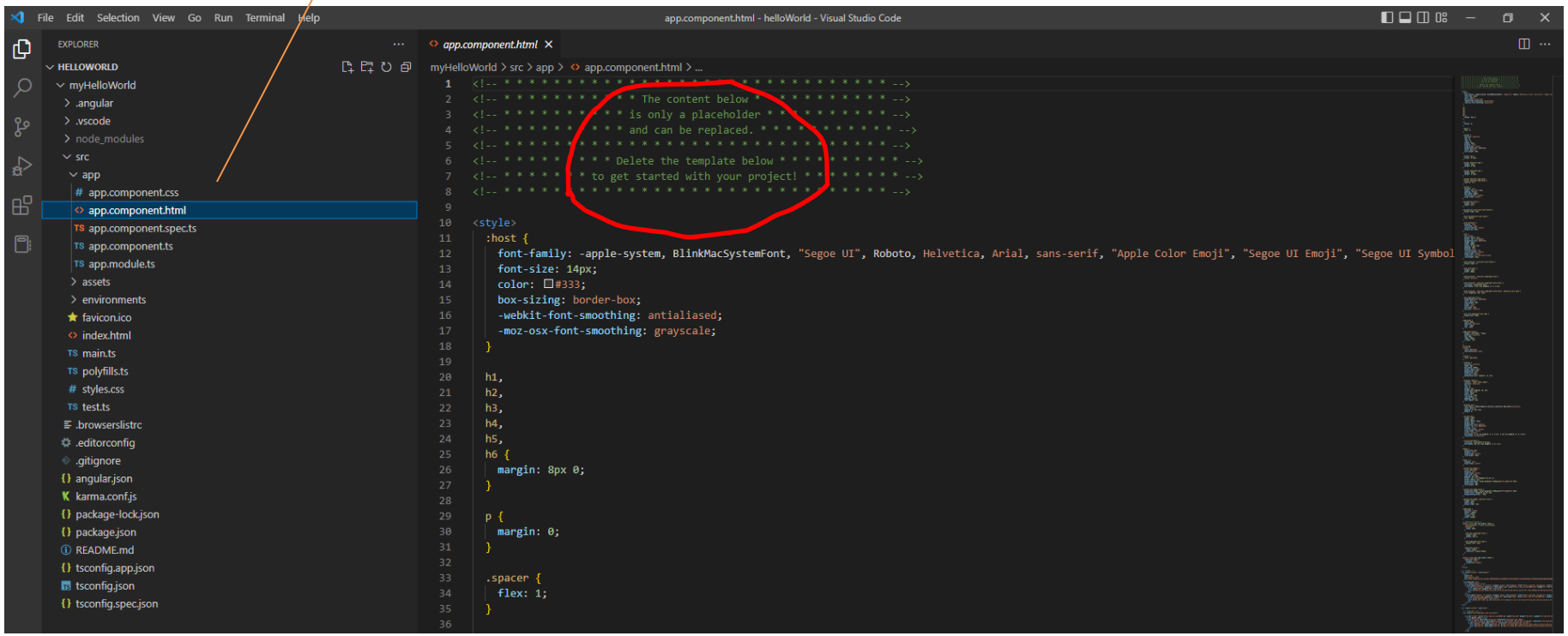
w pasku adresu przeglądarki wpisujemy:
localhost:4200



strona nowego projektu

Tworzenie nowego projektu

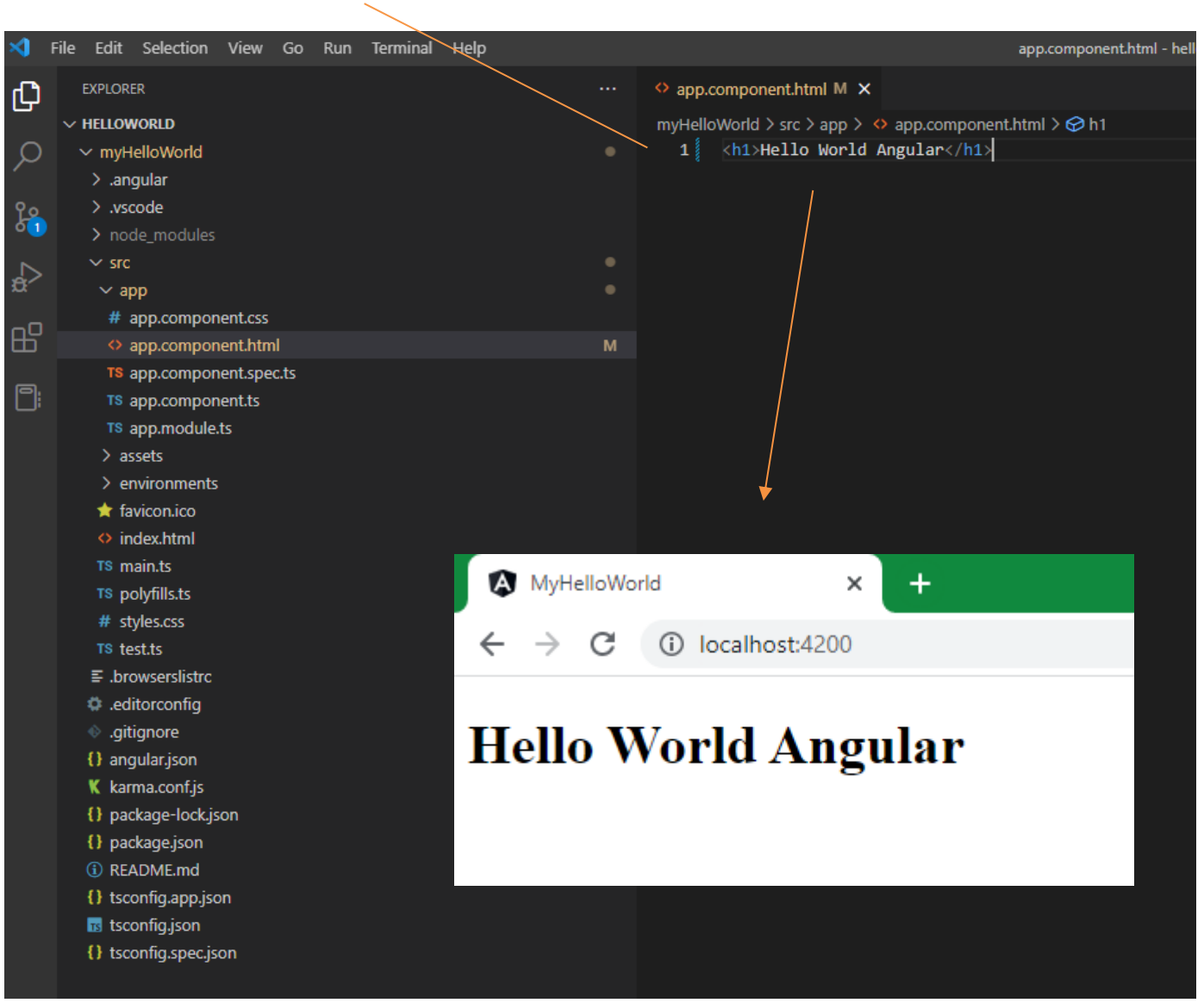
usuwamy zawartość pliku app.component.html (zawartość startowej strony)



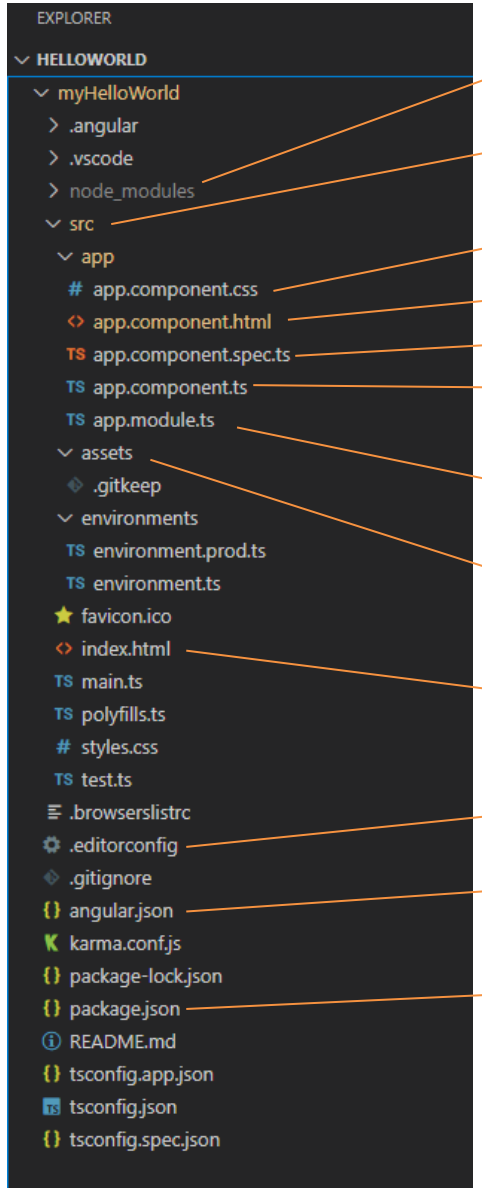
Angular ver.14

Tworzenie nowego projektu

w pliku app.component.html wpisujemy nasz kod



Struktura projektu



katalog z modułami wymaganymi przez aplikację

katalog z elementami użytymi do budowania aplikacji

arkusz stylów css dla komponentu

szablon html komponentu

testy (unit tests)

klasa komponentu napisana w TypeScript

konfiguracja aplikacji (modułów)
(plik pojawia się, gdy tworzymy aplikację z opcją --no-standalone)

katalog plików wykorzystywanych przez aplikację
(zdjęcia, dźwięki itp.)

strona główna

plik konfiguracyjny Visual Studio Code

główny plik konfiguracyjny projektu

zbiór wszystkich pakietów i zależności wykorzystywanych w projekcie

app.component.ts

definicja komponentu AppComponent w postaci klasy z zaimportowanymi modułami (bibliotekami) (główny komponent aplikacji)

wczytanie modułów (dodatkowych bibliotek):
@angular/core - zarządza pracą komponentów
@angular/common - eksportuje wszystkie podstawowe dyrektywy i potoki Angulara
@angular/router - umożliwia nawigację z jednego widoku do drugiego, gdy użytkownicy wykonują zadania aplikacji.

dekorator (konfiguracja komponentu)

szablon HTML powiązany z komponentem

w selektorze <app-root> w pliku index.html Angular umieści zawartość szablonu HTML

```
TS app.component.ts
myHelloWorld > src > app > TS app.component.ts > ...
1   import { Component } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { RouterOutlet } from '@angular/router';
4
5   @Component({
6     selector: 'app-root',
7     standalone: true,
8     imports: [CommonModule, RouterOutlet],
9     templateUrl: './app.component.html',
10    styleUrl: './app.component.css'
11  })
12  export class AppComponent {
13    title = 'myHelloWorld';
14  }
```

```
<body>
  <app-root></app-root>
</body>
```

```
<h1>Hello World Angular</h1>
```

szablon app.component.html zawiera stronę wyświetlaną przez aplikację

pole klasy o nazwie title

arkusz stylów powiązany z szablonem

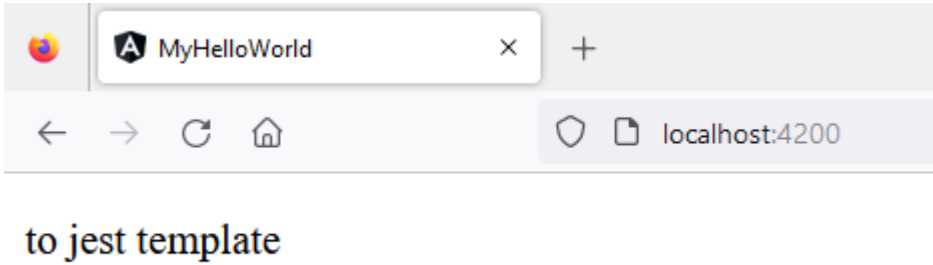
Hello World Angular

Angular ver.18

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: '<p> to jest template </p>',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myHelloWorld';
}
```

template można umieścić bezpośrednio w pliku



Templates

Szablony to kod HTML, który Angular renderuje dla komponentu.

- **Bindings:** interpolation {{ name_of_class_field }} przykład: {{welcome}}
  property [property_of_element] przykład: [src] = „name_of_class_field”
  attributes [attr.name_of_attribute] przykład: [attr.colspan] = „name_of_class_field”
  event (event) przykład: (click) = „pstryk()”
  styles [style.name_of_style] przykład: [style.text-align] = „name_of_class_field”
  class [class.name_of_class] przykład: [class.myClass] = „name_of_class_field”
- **Two-way binding** : [(ngModel)] przykład: [(ngModel)] = „name_of_class_field”
- **Template refs:** local names like #box to reference elements or directives.
- **Structural directives** (*ngIf, *ngFor) change the DOM layout.

https://www.w3schools.com/angular/angular_templates.asp

Interpolation

Displays component values in the DOM with double curly braces.

Read it as: “take this value and print it as text”.

One-way only: **data → view**.

https://www.w3schools.com/angular/angular_templates_interpolation.asp

Interpolation

Interpolation {{ expression }}: Angular evaluates the expression against the component and inserts the result as text (HTML-escaped).

Property values: {{ title }} and {{ name }} read component fields.

Expressions: {{ 2 + 3 }} evaluates arithmetic;
{{ name.toUpperCase() }} calls a method on the string value.

https://www.w3schools.com/angular/angular_templates_interpolation.asp

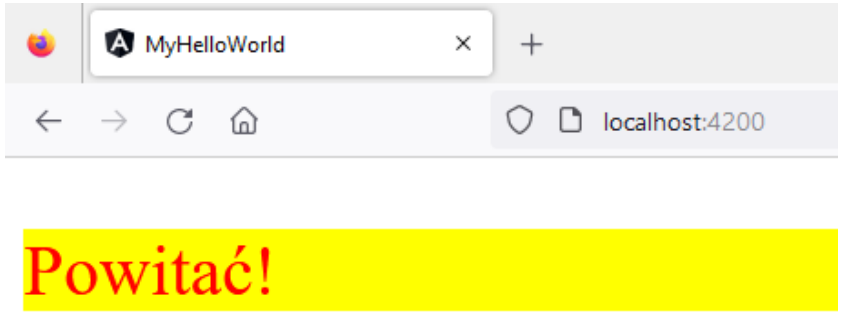
interpolacja

app.component.ts

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template:`
  <style>
  p {
    background-color: yellow;
    font-size: 30px;
    color:red;
  }
  </style>
  <p>{{welcome}}</p>
  `
  //templateUrl: './app.component.html',
  //styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myHelloWorld';
  welcome: string = "Powitać!"
}
```

template wielowierszowy znajduje się pomiędzy znakami backtick `

interpolacja – nazwę pola wstawiamy między dwa nawiasy klamrowe (string interpolation)



nowe pole o nazwie welcome o wartości typu string „powitać”

Angular ver.14

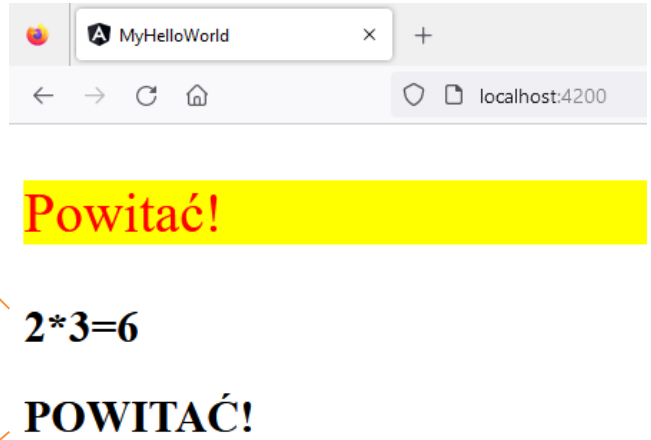
interpolacja - wstawianie wyrażenia, wywoływanie metody

app.component.ts

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template:`
  <style>
  p {
    background-color: yellow;
    font-size: 30px;
    color:red;
  }
  </style>
  <p>{{welcome}}</p>
  <h2> 2*3={{2*3}}</h2>
  <h2>{{welcome.toUpperCase()}}
  `
  //templateUrl: './app.component.html',
  //styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myHelloWorld';
  welcome: string = "Powitać!"
}
```

wyrażenie

wywołanie metody



Angular ver.14

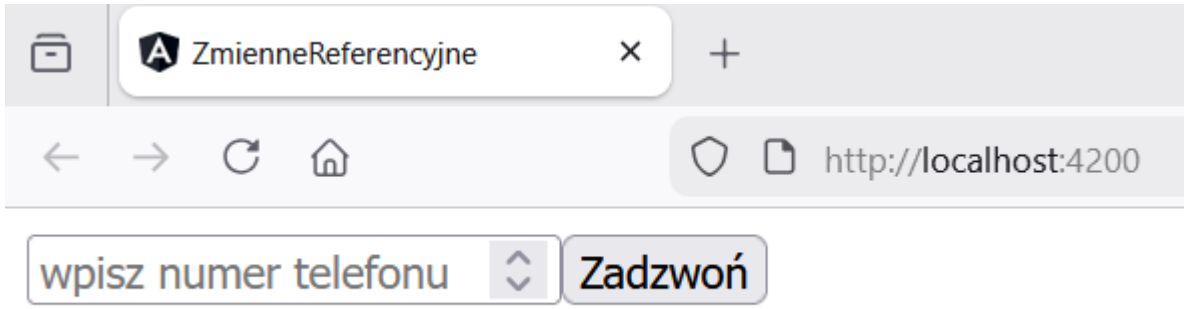
Template reference variables
(zmienne referencyjne w szablonie)

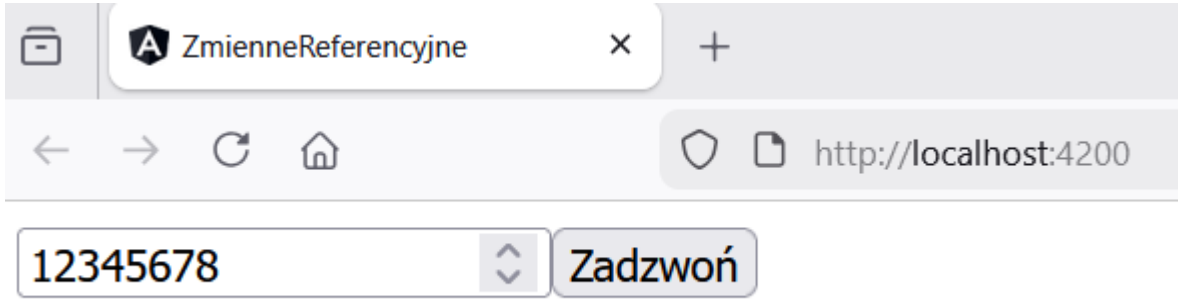
Nadaje nazwę lokalną elementowi (np. #box)

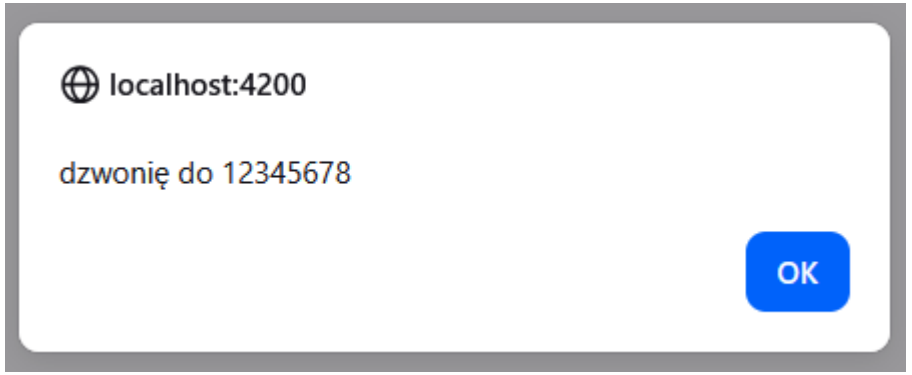
Umożliwia odczyt wartości lub wywoływanie metod bezpośrednio w szablonie.

Zakres ograniczony do szablonu, w którym został zadeklarowany.

Template reference variables
(zmienne referencyjne w szablonie)

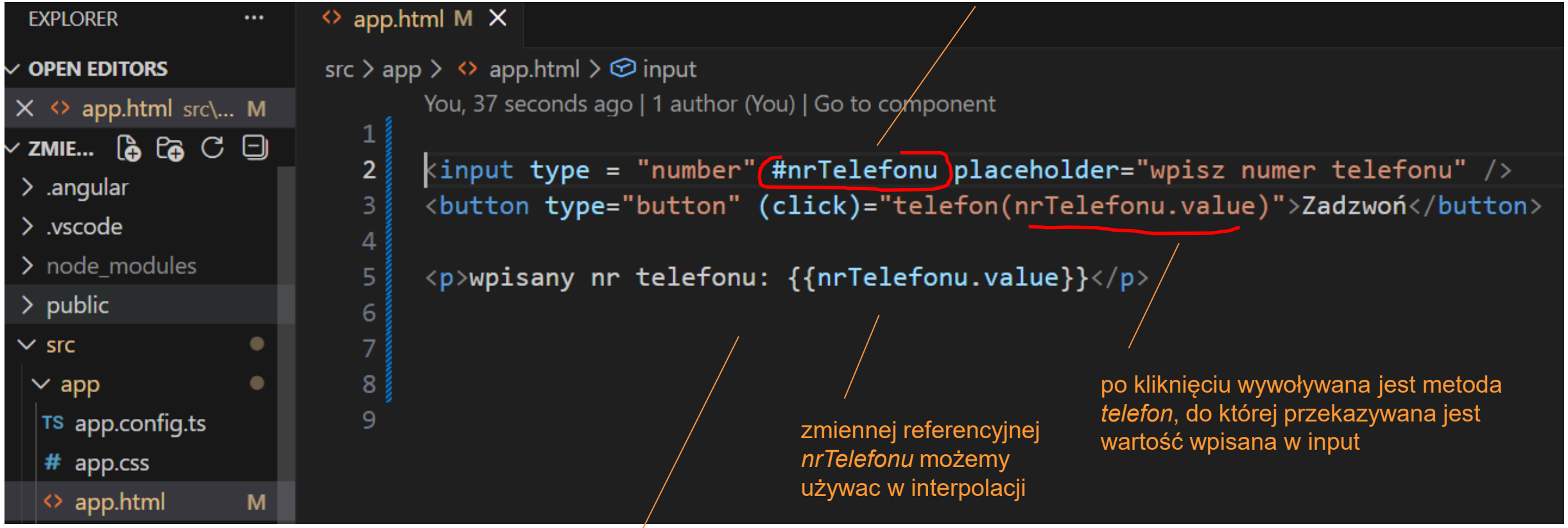






Template reference variables
(zmienne referencyjne w szablonie)

zmienna referencyjna *nrTelefonu*



zmiennej referencyjnej *nrTelefonu* możemy używac w interpolacji

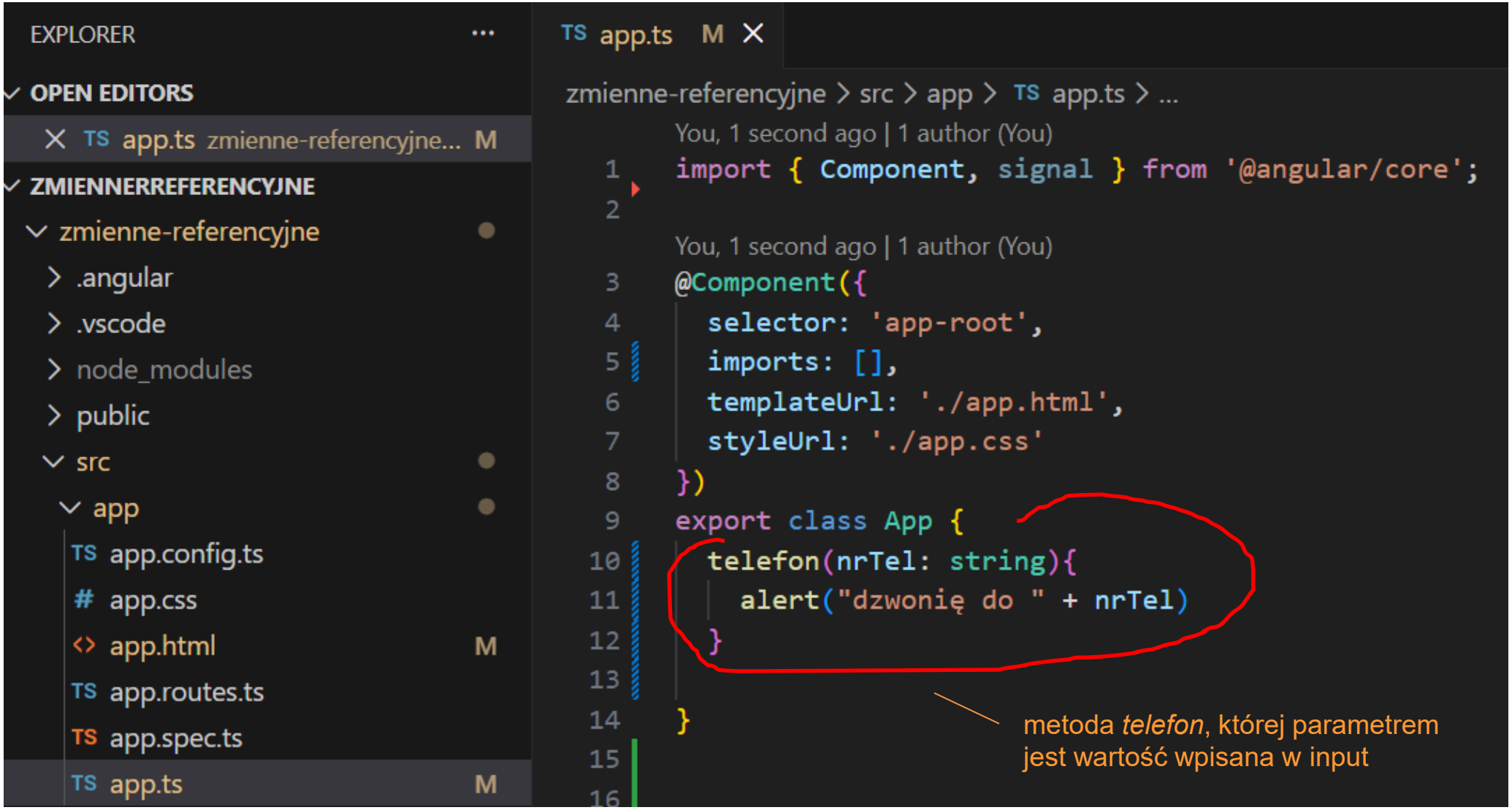
po kliknięciu wywoływana jest metoda *telefon*, do której przekazywana jest wartość wpisana w input

nrTelefonu:

- nie jest zmienną TypeScript
- nie istnieje w pliku app.ts
- działa tylko w HTML (template)
- jest referencją do elementu DOM

Angular ver. 21

Template reference variables
(zmienne referencyjne w szablonie)



wiązanie właściwości (propety binding)

```
<img [src]="imageUrl">
```

app.html

wiązanie właściwości (property) ***src*** z polem ***imageUrl***

```
imageUrl = "https://pictures.com/picture.png"
```

app.ts

błąd

```
Can't bind to 'colspan' since it isn't a known property of 'td'.
app.ts(6, 16): Error occurs in the template of component App.
View Problem (Alt+F8)    No quick fixes available
<td [colspan]="3"></td>        You, 1 second ago • Uncommitted changes
```

app.html

```
<td [attr.colspan]="3"></td>
```

app.html

colspan nie jest właściwością tylko atrybutem, stąd używamy ***[attr.colspan]***

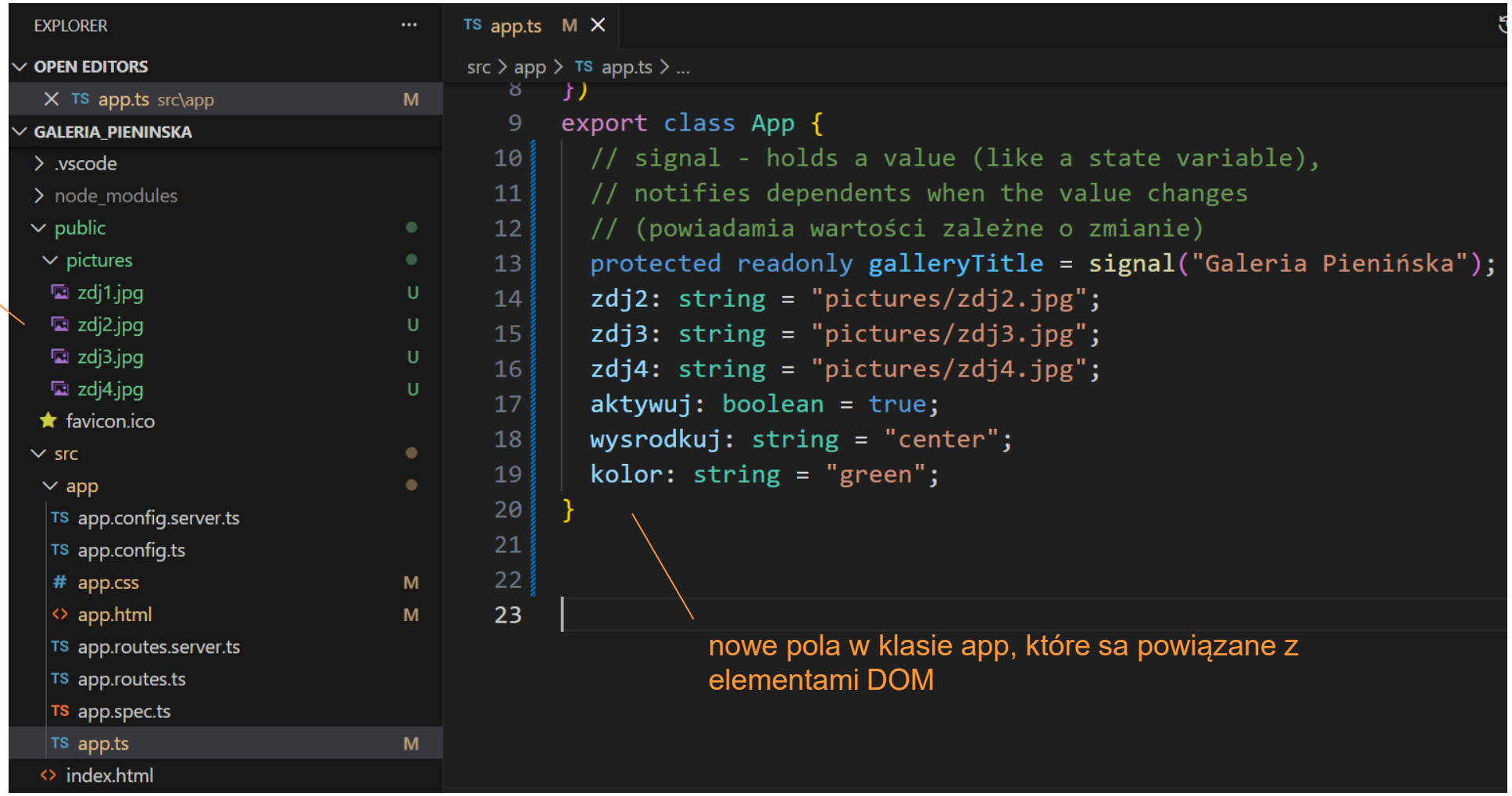
Angular ver. 20

wiązanie właściwości
(propety binding)

wiązanie właściwości (propety binding)

app.ts

zdjęcia w katalogu ***public***



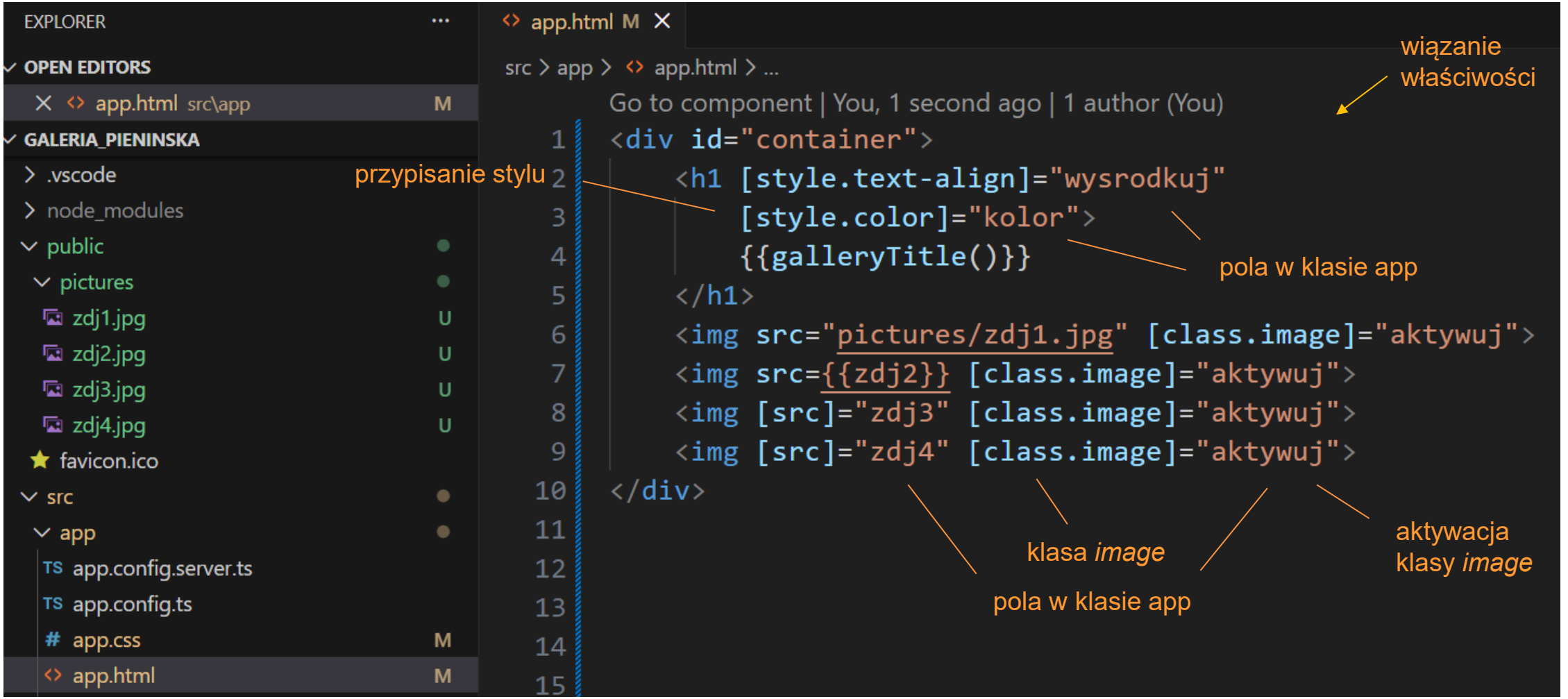
nowe pola w klasie app, które sa powiązane z elementami DOM

Angular ver.20

wiązanie właściwości (propety binding)

powiązanie tworzy aktywne połączenie pomiędzy częścią interfejsu użytkownika utworzoną na podstawie szablonu (elementem DOM, dyrektywą lub komponentem) a modelem (instancją komponentu, do którego należy szablon).

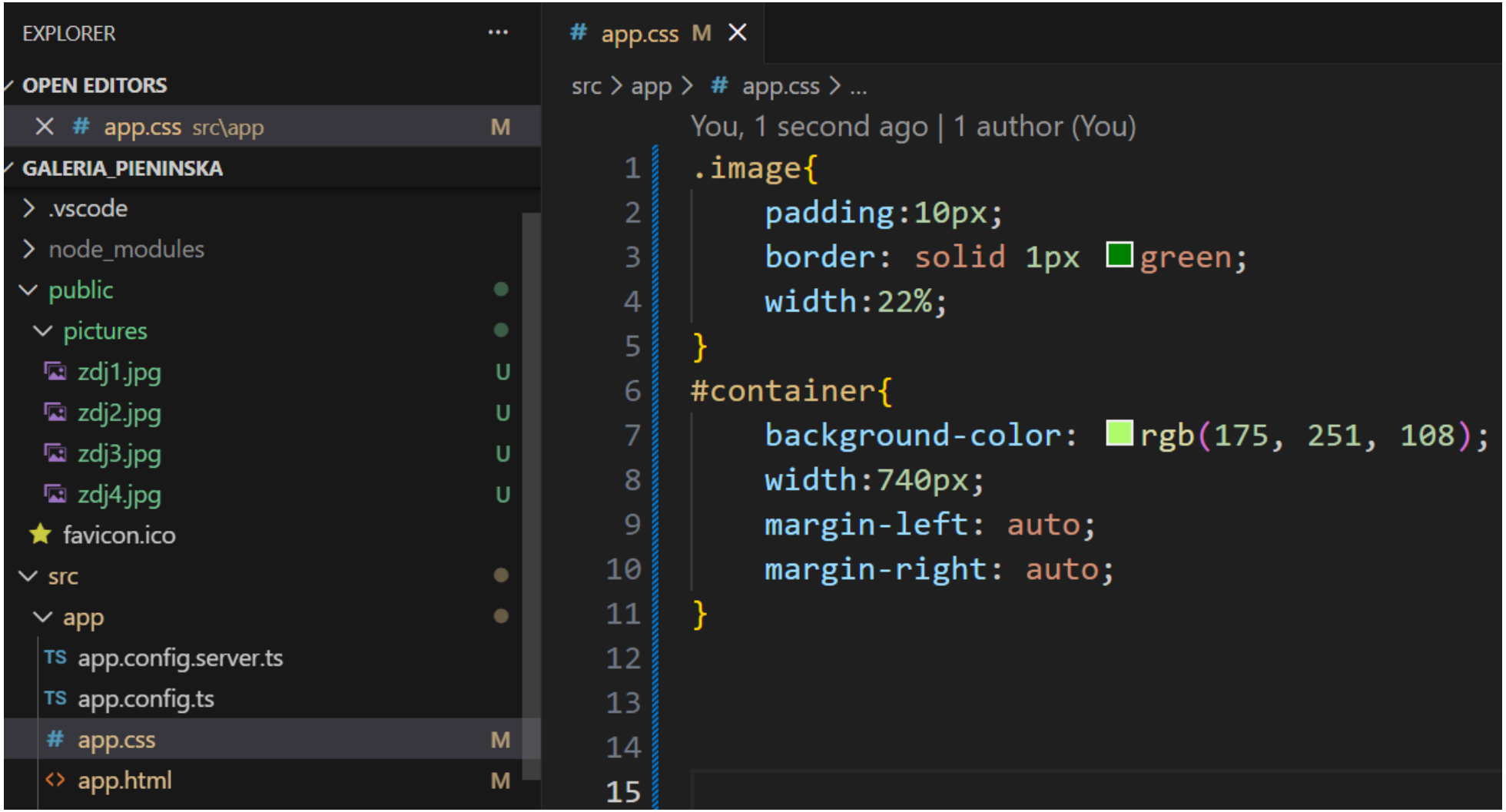
app.html



https://angular.io/guide/binding-overview

wiązanie właściwości (propety binding)

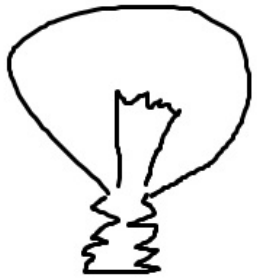
app.css



wiązanie zdarzeń
(event binding)

projectTitle

Bulb



przełacz

Bulb

przełacz

buton zmienia src obrazka (bulb)

wiązanie zdarzeń (event binding)

app.component.html M

bulbEvent > src > app > app.component.html > ...

```
1  <h1>{{projectTitle}}</h1>
2  <img [src]="bulb">
3  <button (click)="pstryk()">przełacz</button>
```

interpolacja tytułu strony

związanie właściwości src obrazka do pola bulb

app.component.ts M

bulbEvent > src > app > app.component.ts > ...

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    projectTitle: string = "Bulb";
10   bulb: string = "assets/pictures/turnedOff.jpg";
11
12   pstryk(){
13     this.bulb = this.bulb == "assets/pictures/turnedOff.jpg" ? "assets/pictures/turnedOn.jpg":"assets/pictures/turnedOff.jpg";
14   }
15 }
```

pole tytułu strony

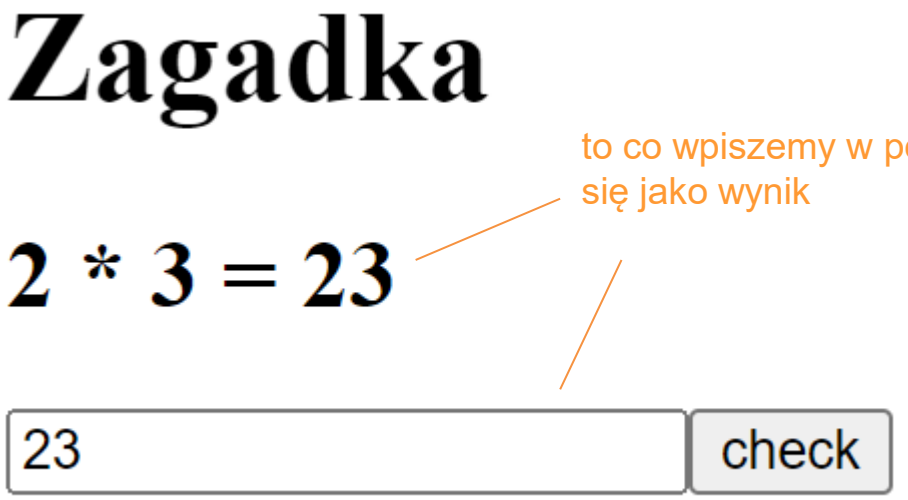
pole bulb (src obrazka)

zdarzenie click przycisku jest związane z metodą pstryk() podmieniającą src obrazka

Angular ver.14

wiązanie dwukierunkowe (two-way binding)

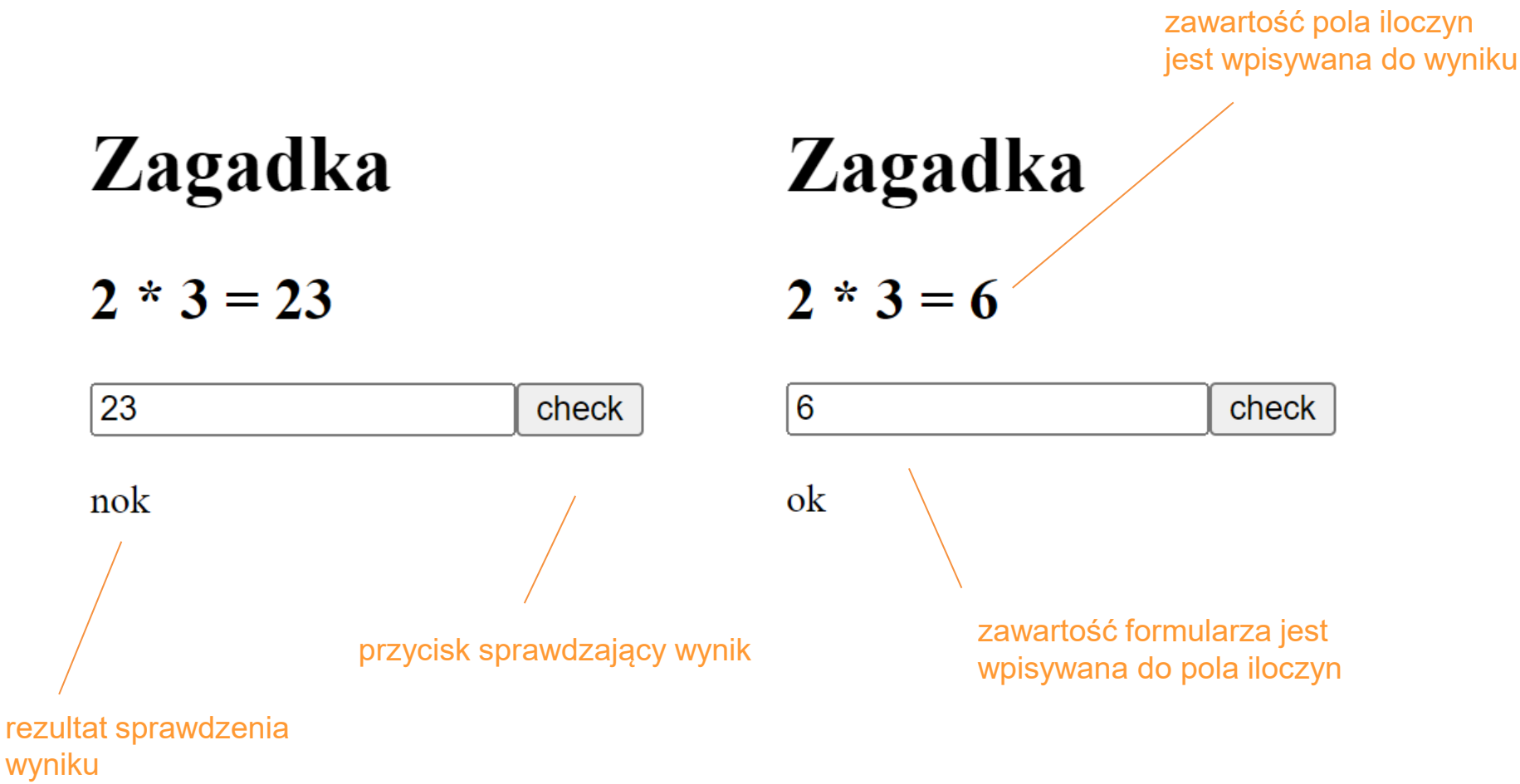
zmiany w szablonie są odzwierciedlone w kodzie i na odwrót



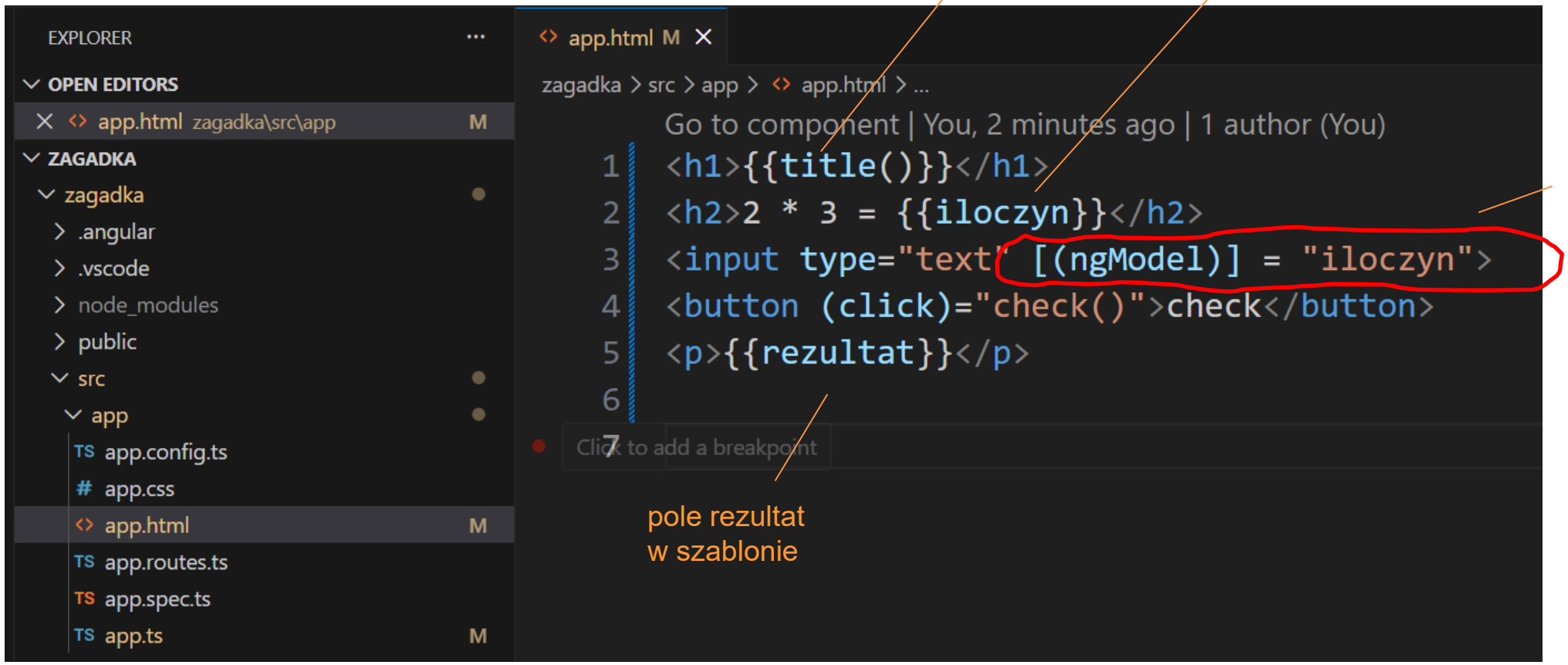
to co wpiszemy w pole input pokazuje się jako wynik

dane z widoku (np. zmiana w szablonie) są przekazywane do komponentu (kodu TipeScript) i na odwrót

wiązanie dwukierunkowe
(two-way binding)

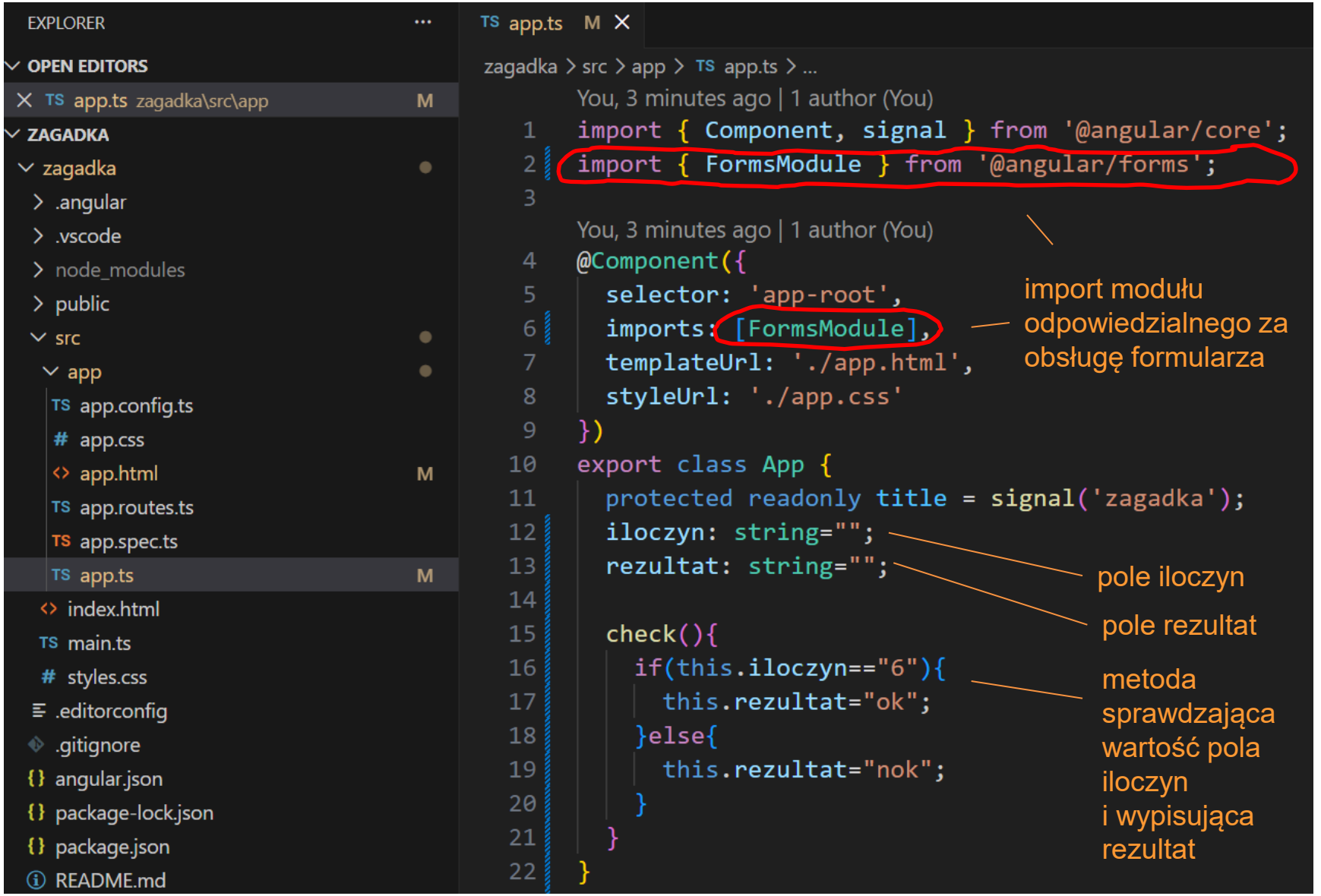


wiązanie dwukierunkowe (two-way binding)



Angular ver.21

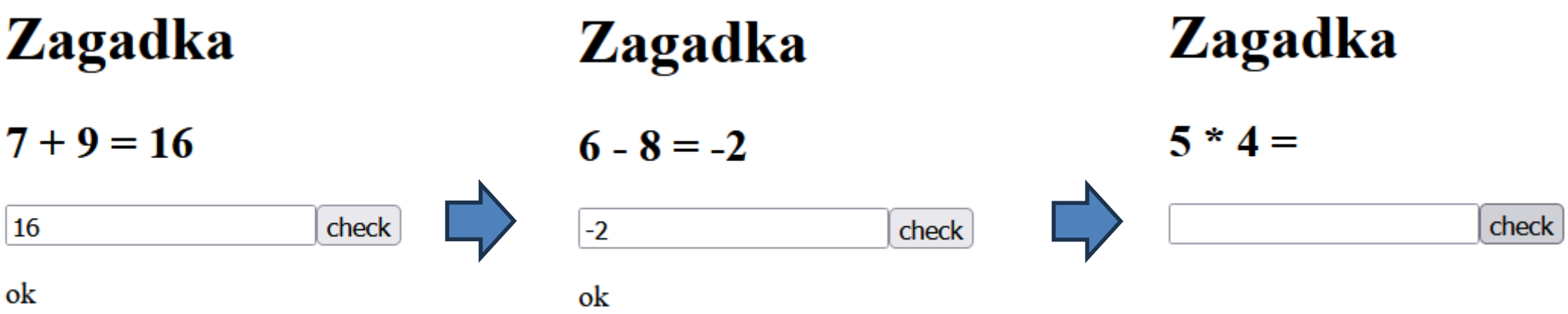
wiązanie dwukierunkowe (two-way binding)



Angular ver.21

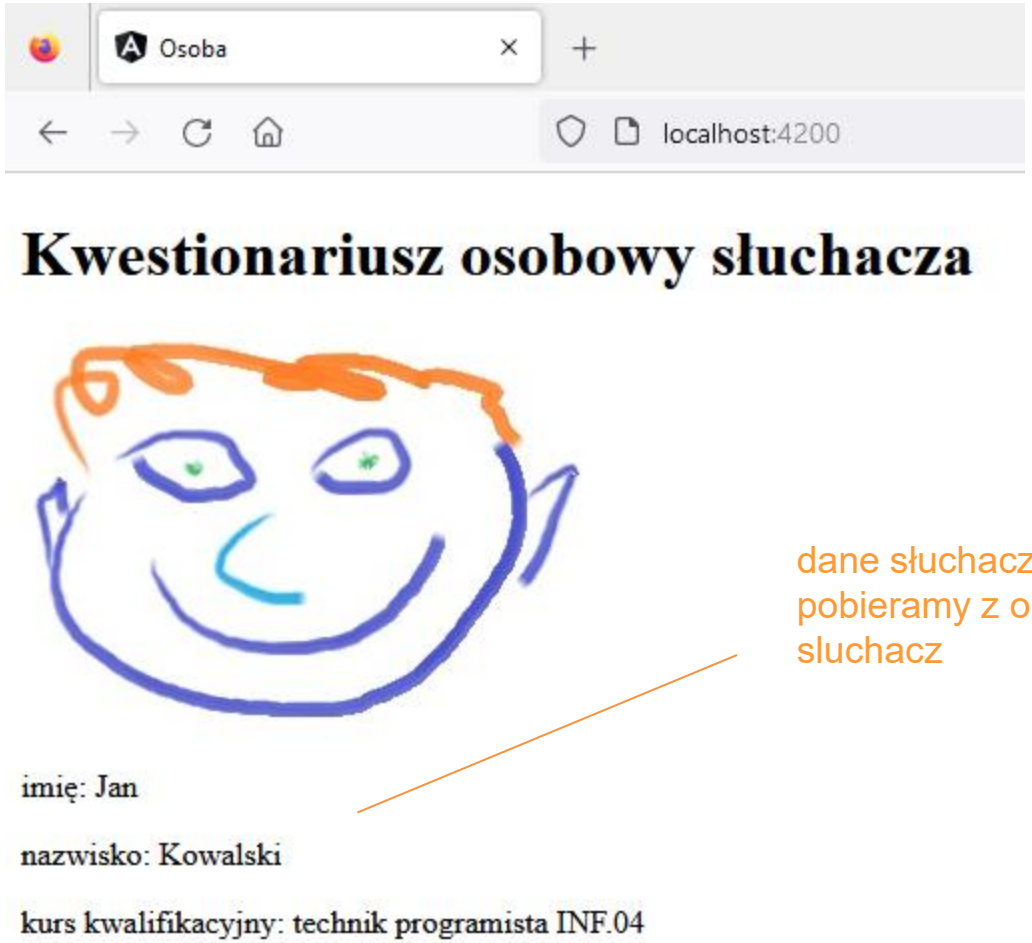
wiązanie dwukierunkowe (two-way binding)

rozwinięcie zagadki



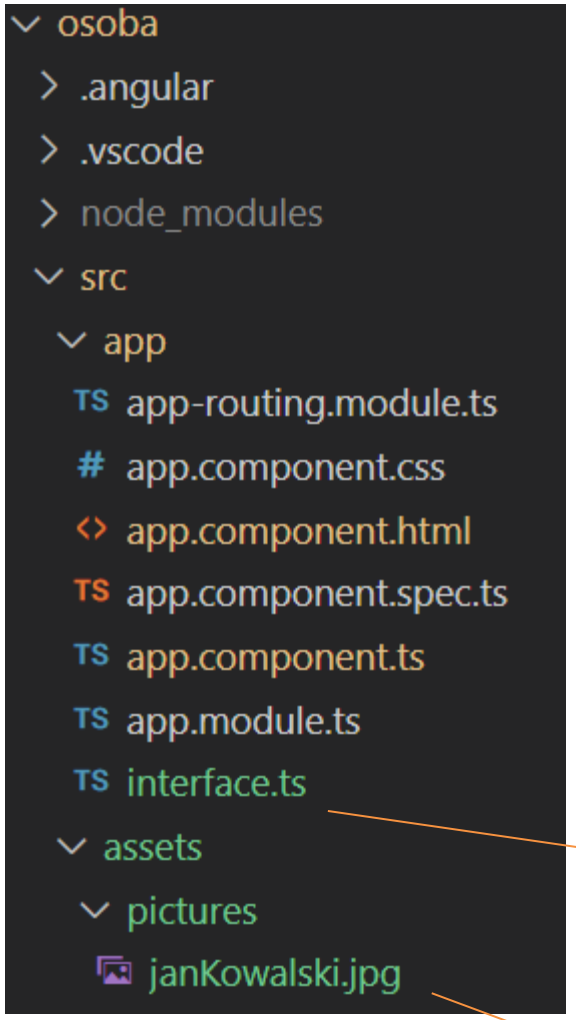
każdorazowo po podaniu
prawidłowego wyniku losowana jest
kolejna zagadka (liczby oraz operator)

Sluchacz



dane słuchacza pobieramy z obiektu sluchacz

interface.ts



TS interface.ts U

osoba > src > app > TS interface.ts > ...

```ts
export interface Sluchacz{
    imie: string;
    nazwisko: string;
    kursKwalifikacyjny: string;
    zdjecie: string;
}
```

w pliku interface,ts
(nazwa jest nasza)
umieszczamy interfejs
Sluchacz oraz go
eksportujemy

w katalogu assets
umieszczamy zdjęcie
słuchacza

Angular ver.14

app.component.ts

TS app.component.ts M

osoba > src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';
import { Sluchacz } from './interface'

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  sluchacz: Sluchacz = {
    imie: "Jan",
    nazwisko: "Kowalski",
    kursKwalifikacyjny: "technik programista INF.04",
    zdjecie: "assets/pictures/janKowalski.jpg"
  }

}
```

import interfejsu Sluchacz

obiekt sluchcz jest stworzony na podstawie zaimportowanego interfejsu

app.component.html

app.component.html M

osoba > src > app > app.component.html > ...

```
1  <div id="container">
2    <h1>Kwestionariusz osobowy słuchacza</h1>
3    <img src={{sluchacz.zdjecie}}>
4    <p>imię: {{sluchacz.imie}}</p>
5    <p>nazwisko: {{sluchacz.nazwisko}}</p>
6    <p>kurs kwalifikacyjny: {{sluchacz.kursKwalifikacyjny}}</p>
7  </div>
```

wykorzystujemy interpolację

Angular ver.14

Dyrektywy

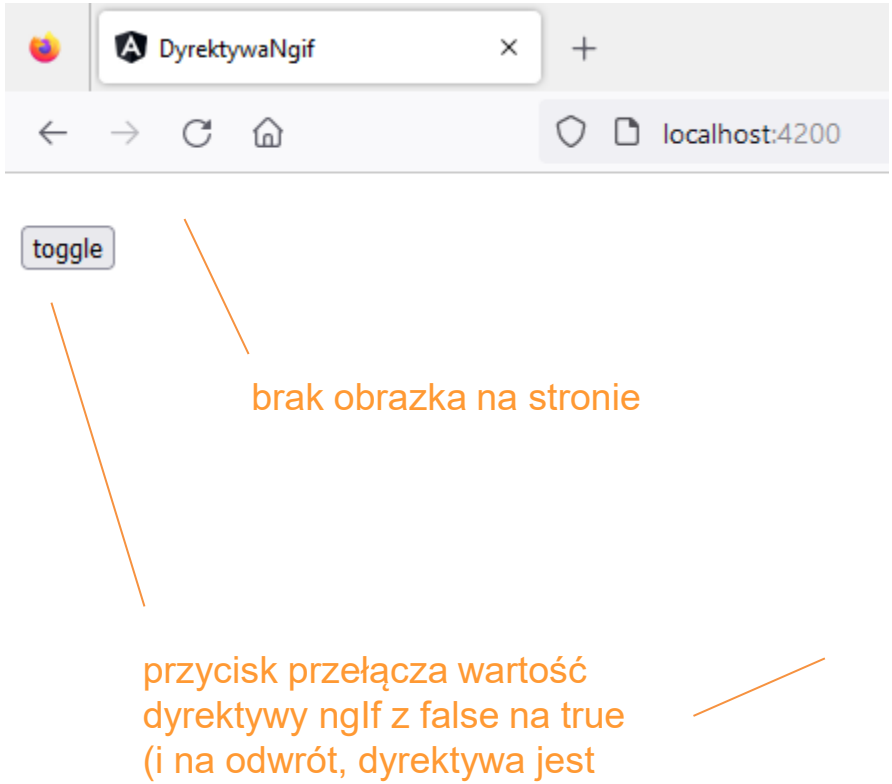
umożliwiają dostęp do struktury dokumentu, mogą zmieniać wygląd i zachowanie danego elementu strony

- komponenty (*Component Directives*)
- dyrektywy strukturalne (*Structural Directives*)
- dyrektywy atrybutowe (*Attribute Directives*)

Dyrektywa strukturalna ngIf

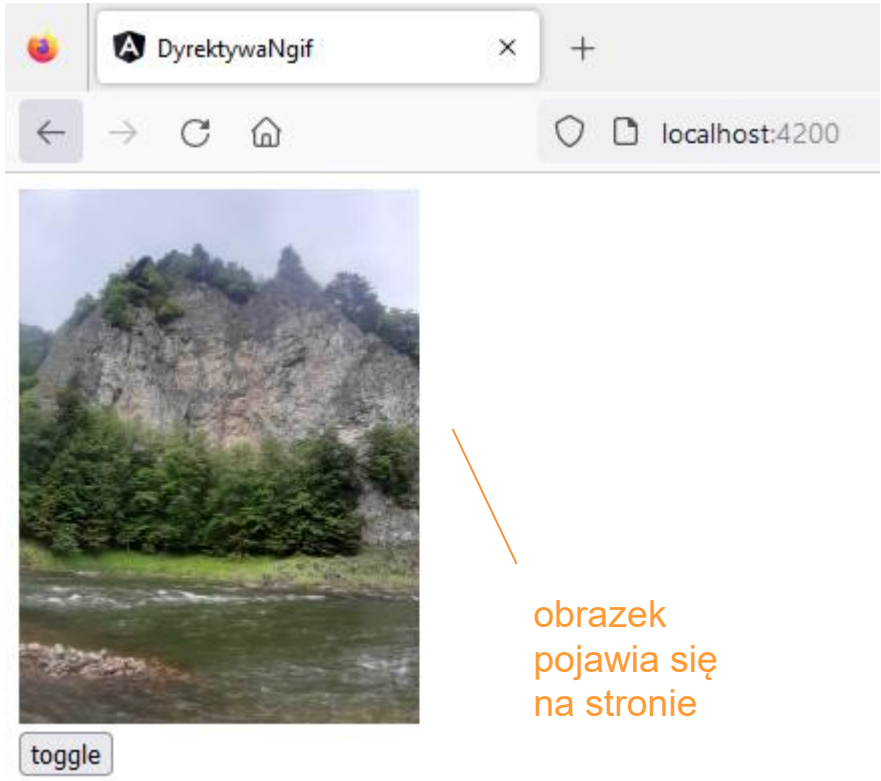
służy do dodawania lub usuwania elementów HTML na stronie (jeśli ma wartość true – element jest uwzględniony w drzewie dokumentów, jeśli false – nie jest uwzględniany).

z pomocą dyrektywy ngIf obrazek pojawia się i znika ze strony



brak obrazka na stronie

przycisk przełącza wartość dyrektywy ngIf z false na true (i na odwrót, dyrektywa jest umieszczona w znaczniku img)



obrazek pojawia się na stronie

Dyrektywa strukturalna
nglf

TS app.component.ts M

dyrektywaNgif > src > app > TS app.component.ts > ...

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     template: `
6       <img src="assets/picture/pieniny.jpg" *ngIf = "showImage">
7       <br>
8       <button (click)="toggle()">toggle</button>
9     `,
10    styleUrls: ['./app.component.css']
11  })
12  export class AppComponent {
13      showImage: boolean = false;
14      toggle(){
15        this.showImage = !this.showImage;
16      }
17  }
```

do dyrektywy nglf jest przypisana wartość pola showImage

pole showImage jest zainicjaowane wartością false

metoda toggle() (trigerowana przez przycisk) przełącza wartość dyrektywy z false na true i na odwrót

Angular ver.14

Dyrektywa strukturalna
ngIf

TS app.component.ts M

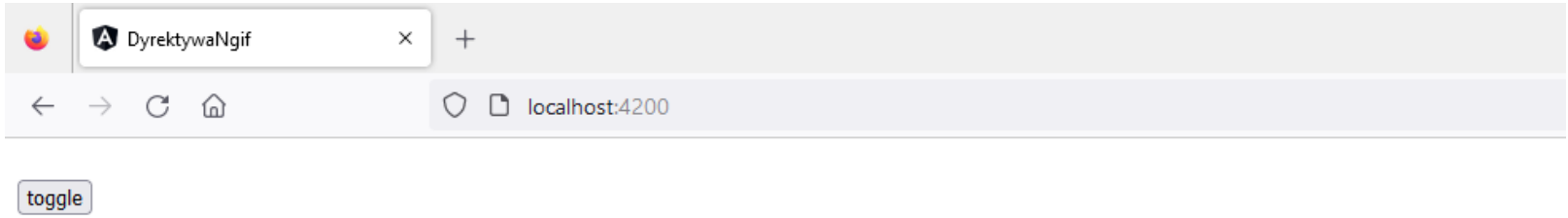
dyrektywaNgIf > src > app > TS app.component.ts > ...

```ts
import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule],
  template: `
      <img  src= "pieniny.jpg" *ngIf="showImage">
      <br>
      <button (click) = "toggle()">toggle</button>
  `,
  styleUrl: './app.component.css'
})
export class AppComponent {
  showImage: boolean = false;
  toggle(){
    this.showImage =  !this.showImage;
  }
}
```

obrazek umieszczamy w katalogu public

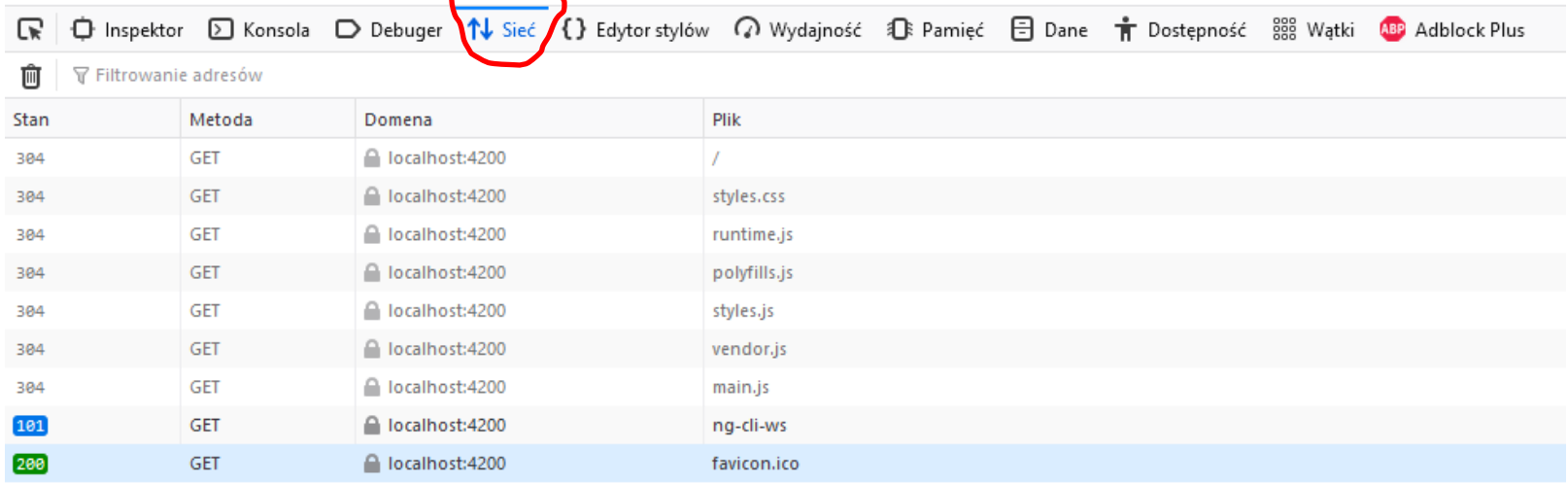
Angular ver.18

sposób działania nglf



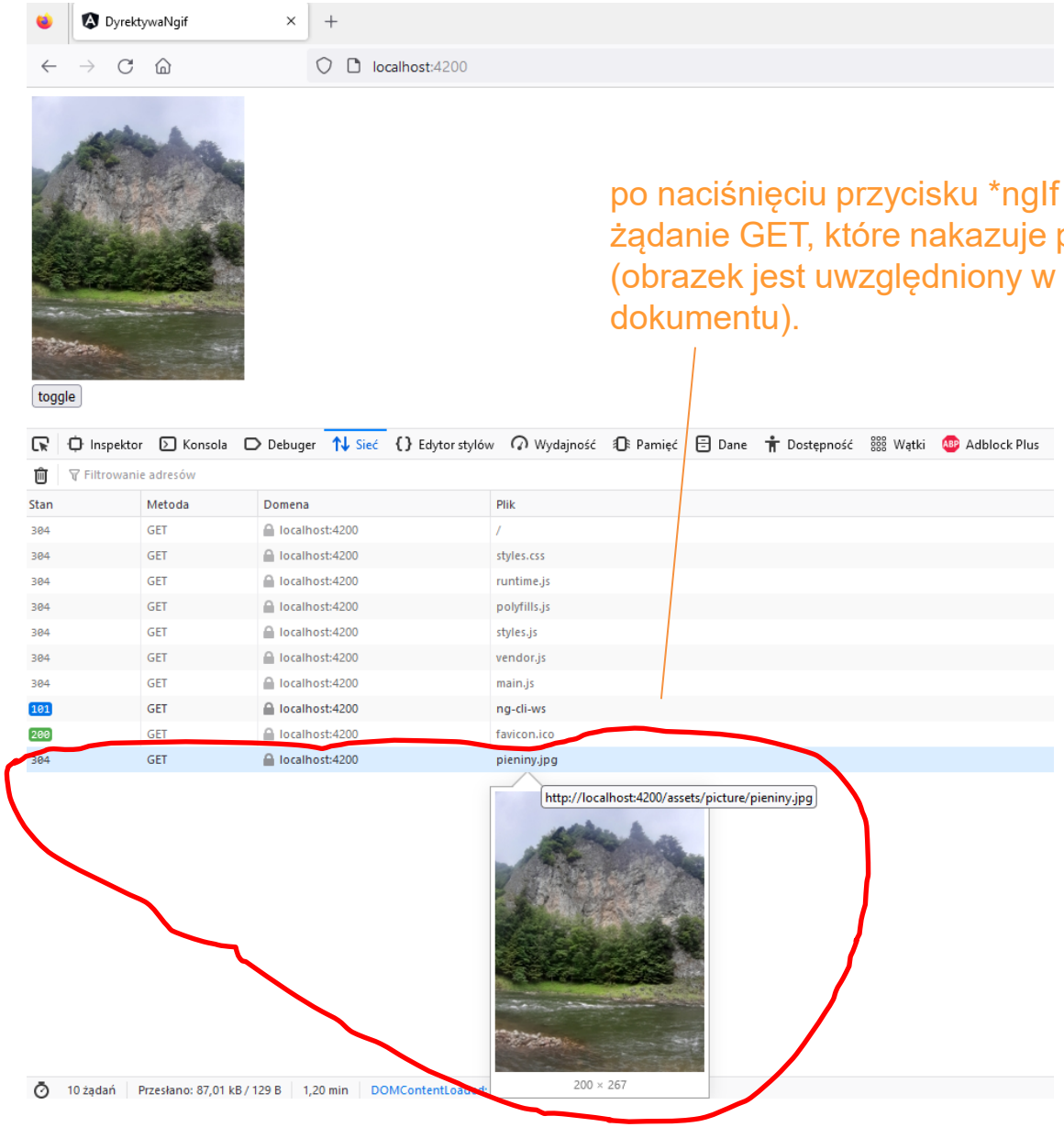
po początkowym załadowaniu strony *nglf = false, stąd brak żądania GET, które nakazuje pobranie obrazka (obrazek nie jest uwzględniony w drzewie dokumentu).

F12 – narzędzia deweloperskie



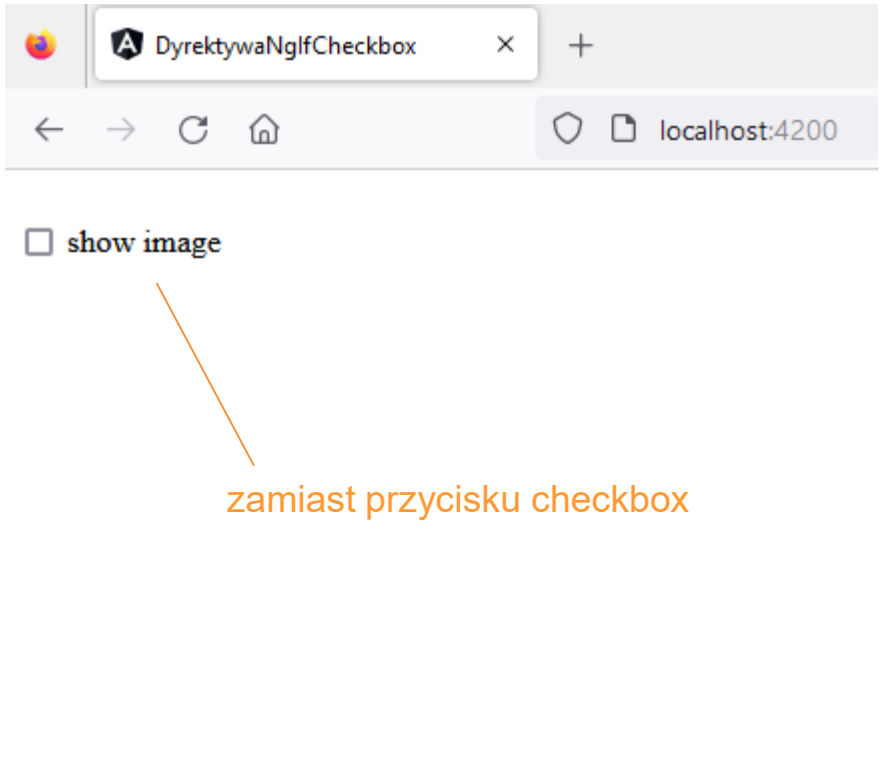
Stan	Metoda	Domena	Plik
304	GET	localhost:4200	/
304	GET	localhost:4200	styles.css
304	GET	localhost:4200	runtime.js
304	GET	localhost:4200	polyfills.js
304	GET	localhost:4200	styles.js
304	GET	localhost:4200	vendor.js
304	GET	localhost:4200	main.js
101	GET	localhost:4200	ng-cli-ws
200	GET	localhost:4200	favicon.ico

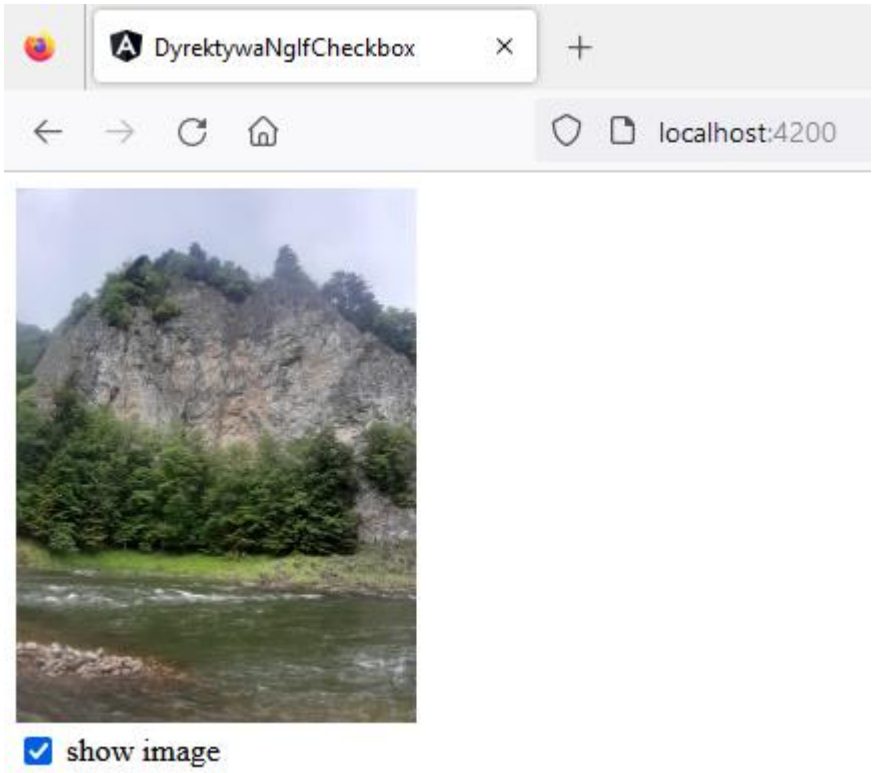
sposób działania
nglf



po naciśnięciu przycisku *ngIf = true, stąd żądanie GET, które nakazuje pobranie obrazka (obrazek jest uwzględniony w drzewie dokumentu).

Dyrektywa strukturalna
nglf z checkbox





Dyrektywa strukturalna
ngIf z checkbox

TS app.component.ts M

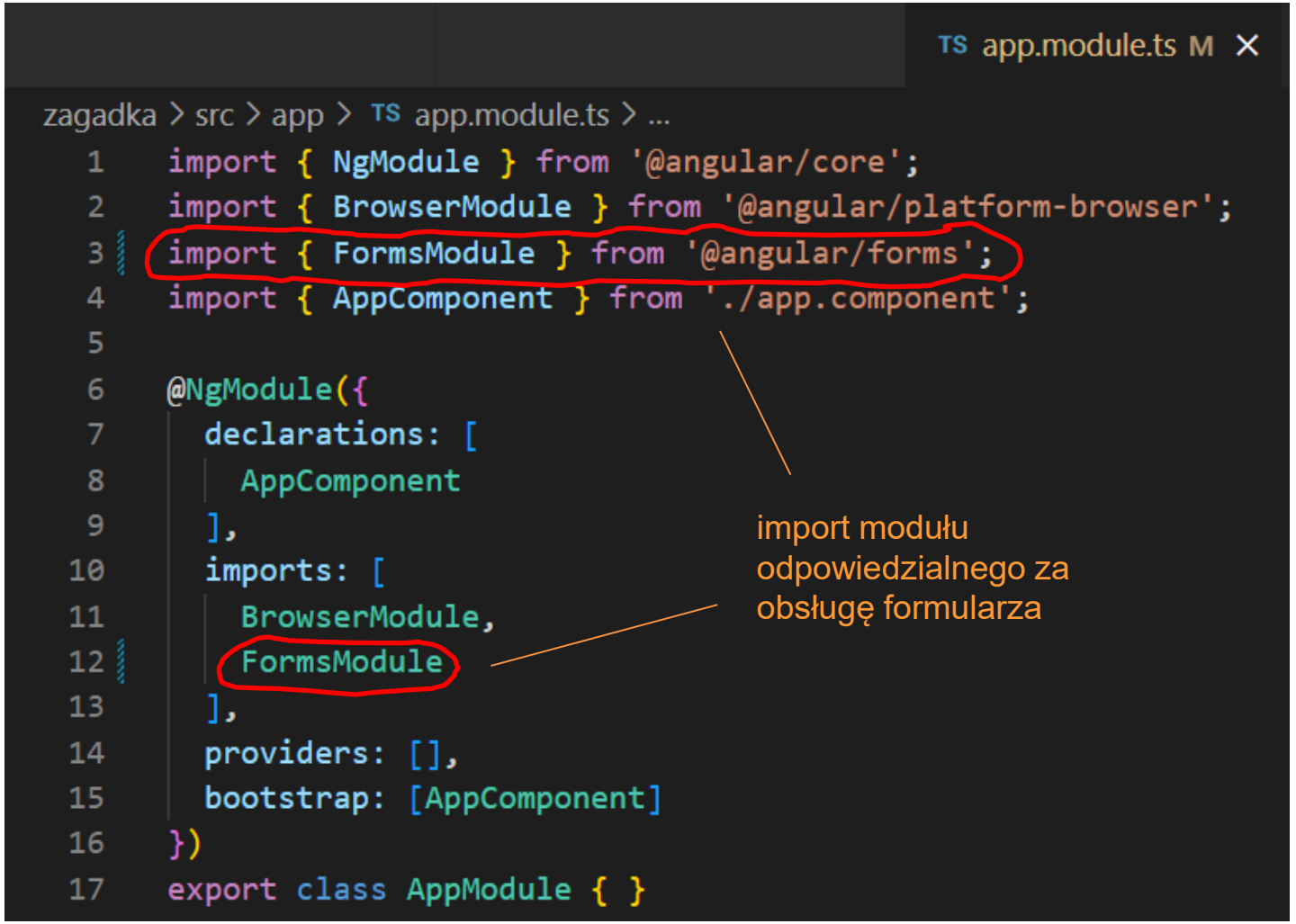
dyrektywaNgIfCheckbox > src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <img src="assets/picture/pieniny.jpg" *ngIf = "showImage">
    <br>
    <!--<button (click)="toggle()">toggle</button>-->
    <label><input type="checkbox" [(ngModel)] = showImage > show image </label>
  `,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
    showImage: boolean = false;
    //toggle(){
    //  this.showImage = !this.showImage;
    //}
}
```

checkbox powiązany z polem showImage dyrektywą ngModel

Angular ver.14

Dyrektywa strukturalna
nglf z checkbox



Angular ver.14

Dyrektywa strukturalna
ngIf z checkbox

app.component.html

dyrektywaNgIfCheckbox > src > app > app.component.html > ...

Go to component

```
1   <img [src] = "obrazek" *ngIf="showImage">
2   <br>
3   <input type="checkbox" [(ngModel)]="showImage">
```

zmiana wartości checkboxa przekłada się poprzez pole showImage na dyrektywę ngif

Angular ver.18

Dyrektywa strukturalna
nglf z checkbox

app.component.ts

dyrektywaNgIfCheckbox > src > app > app.component.ts > ...

```ts
import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, FormsModule, CommonModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'dyrektywaNgIfCheckbox';
  obrazek: string = "kot.jpg";
  showImage: boolean = false;
}
```

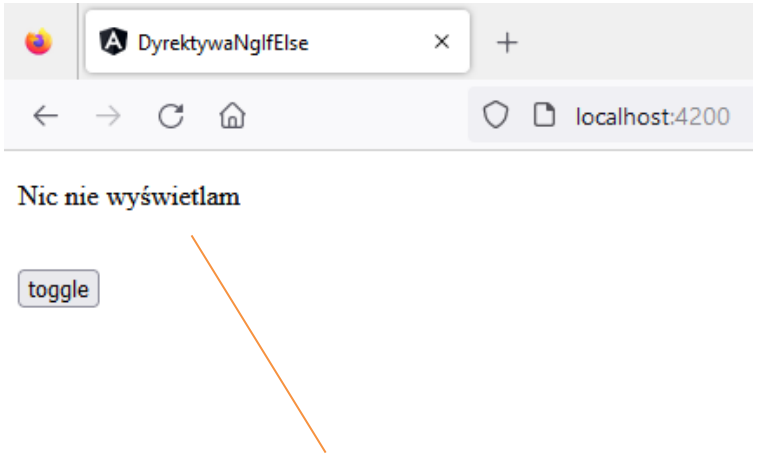
obsługa dyrektywy *ngif

obsługa formularza (checkbox)

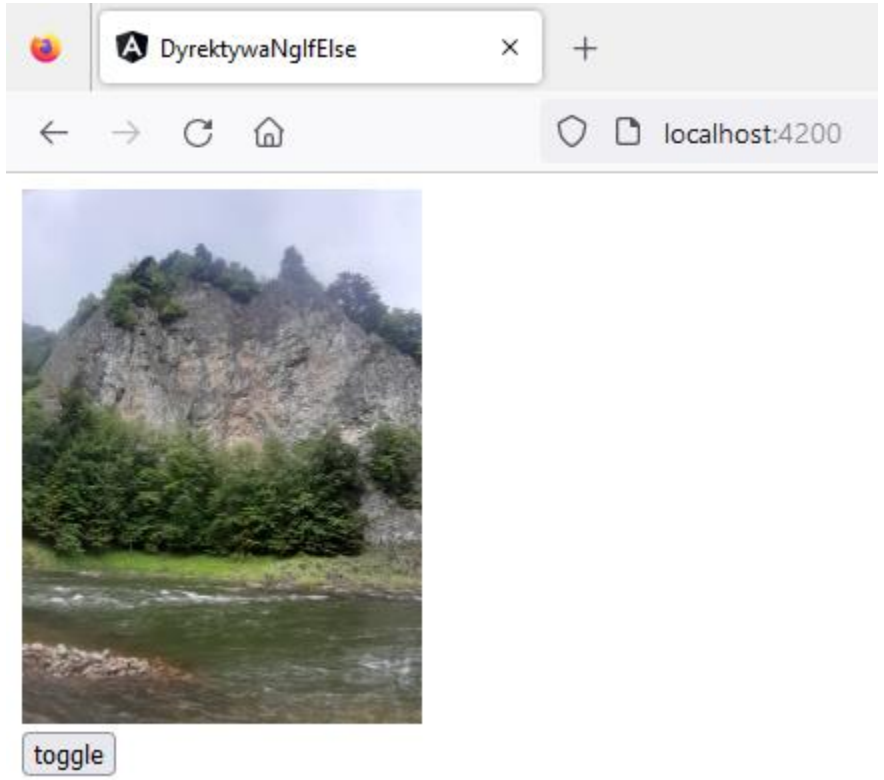
kot.jpg znajduje się w katalogu *public*

Angular ver.18

Dyrektywa strukturalna
nglf z else



efekt działania else przy dyrektywie nglnf



Dyrektywa strukturalna
nglf z else

TS app.component.ts M

dyrektywaNgIfElse > src > app > TS app.component.ts > AppComponent

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     template: `
6       <img src="assets/picture/pieniny.jpg" *ngIf = "showImage else showNothing">
7       <br>
8       <button (click)="toggle()">toggle</button>
9       <ng-template #showNothing>
10        <p>{{information}}</p>
11      </ng-template>
12    `,
13    styleUrls: ['./app.component.css']
14  })
15  export class AppComponent {
16      showImage: boolean = false;
17      information: string = "Nic nie wyświetlam";
18      toggle(){
19        this.showImage = !this.showImage;
20      }
21  }
```

przy dyrektywie ngInf jest else, który zamiast zdjęcia wyświetla zawartość znacznika <ng-template #showNothing>

Angular ver.14

Dyrektywa strukturalna
nglf z else

app.component.html

dyrektywaNgIfCheckbox > src > app > app.component.html > input

```
1   <img [src] = "obrazek" *ngIf="showImage else name">
2   <ng-template #name>
3       <p>{{wyswietla_sie_zamiast_obrazka}}</p>
4   </ng-template>
5   <br>
6   <input type="checkbox" [(ngModel)]="showImage">
```

Angular ver.18

Dyrektywa strukturalna
nglf z else

TS app.component.ts

dyrektywaNgIfCheckbox > src > app > TS app.component.ts > ...

```ts
import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, FormsModule, CommonModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'dyrektywaNgIfCheckbox';
  obrazek: string = "kot.jpg";
  showImage: boolean = false;
  wyswietla_sie_zamiast_obrazka: string = "KOCUREK";
}
```

kot.jpg znajduje się w katalogu *public*

Angular ver.18

@if template syntax

składnia instrukcji warunkowej @if wprowadzono w wersji 17 Angulara

składnia @if przypomina składnię instrukcji warunkowej np. w C++

pieniny.jpg znajduje się w katalogu *public*

app.component.ts M

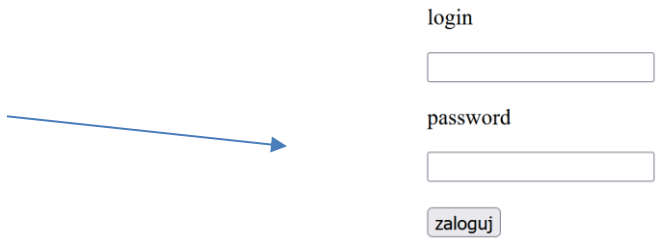
templateSyntaxIfElse > src > app > app.component.ts > ...

```ts
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],
  template: `
  @if (showImage){
    <img src="pieniny.jpg">
  }@else {
    <p>{{information}}</p>
  }
    <br>
    <button (click)="toggle()">toggle</button>
  `,
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'templateSyntaxIfElse';
  showImage: boolean = false;
    information: string = "Nic nie wyświetlam";
    toggle(){
      this.showImage = !this.showImage;
    }
}
```

Angular ver.18

@if template syntax

korzystając z instrukcji warunkowej @if
zaprojektuj formularz do logowania



login

sss

password

ddd

zaloguj

Zły login lub haslo

login

prawidłowy login

password

prawidłowe hasło

zaloguj

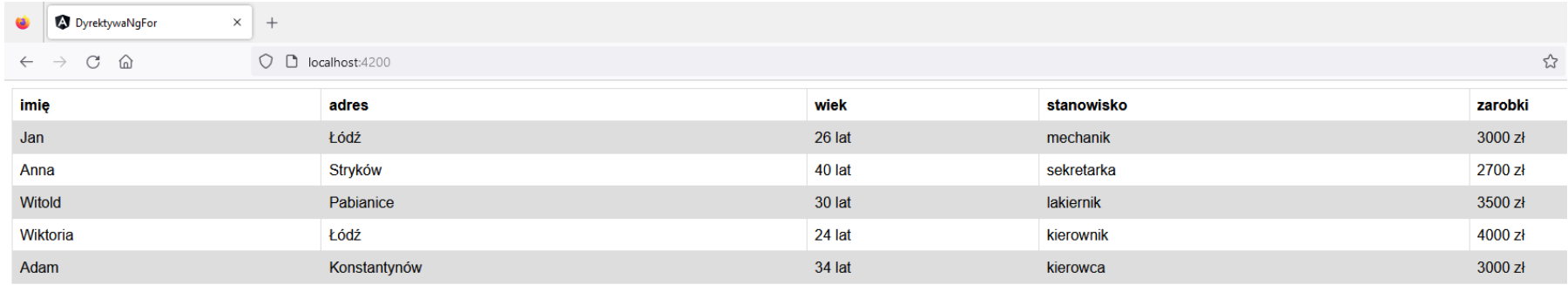
Zalogowało mnie!!!

Dyrektywa strukturalna ngFor

zmienia strukturę drzewa DOM poprzez iterowanie przez tablicę obiektów

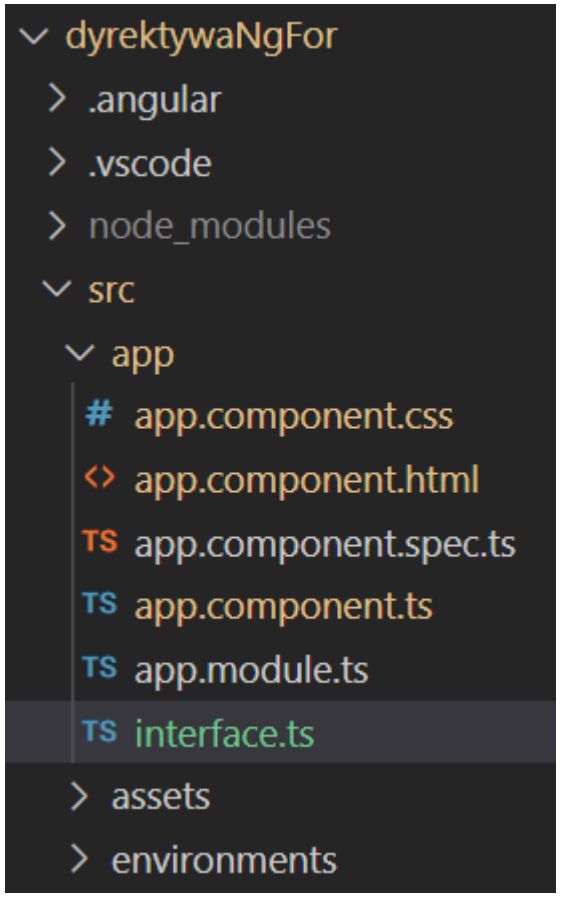
*ngFor = ”let element of kolekcja”

składnia dyrektywy

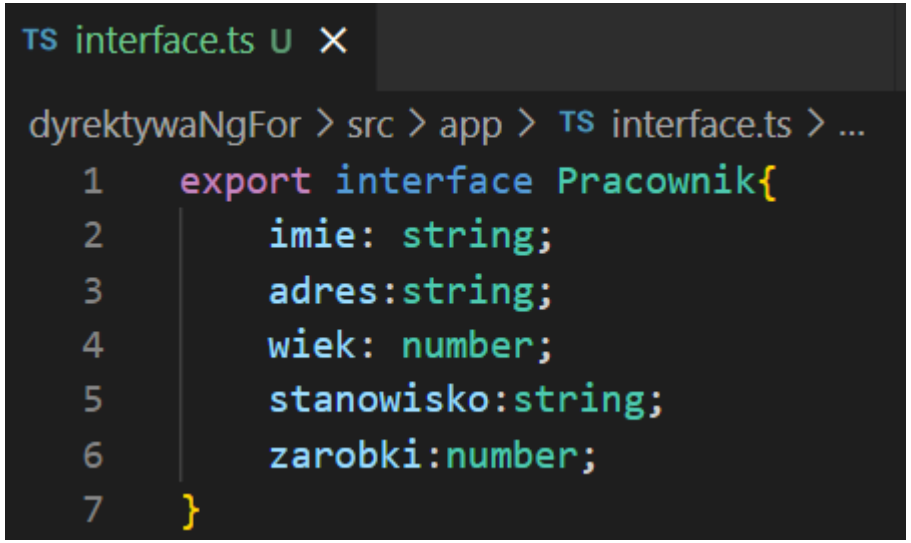


imię	adres	wiek	stanowisko	zarobki
Jan	Łódź	26 lat	mechanik	3000 zł
Anna	Stryków	40 lat	sekretarka	2700 zł
Witold	Pabianice	30 lat	lakiernik	3500 zł
Wiktoria	Łódź	24 lat	kierownik	4000 zł
Adam	Konstantynów	34 lat	kierowca	3000 zł

Dyrektywa strukturalna ngFor



interjejs Pracownik



nowy plik (nazwa jest nasza)

Angular ver.14

Dyrektywa strukturalna ngFor

Angular ver. 8 – trzeba zaimportować CommonModule

TS app.component.ts M

dyrektywaNgFor > src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';
import { Pracownik } from './interface';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  tableHeader: string[] = ['imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
  pracownicy: Pracownik[] = [
    {imie: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
    {imie: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
    {imie: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
    {imie: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
    {imie: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
  ];
}
```

import interfejsu Pracownik

pole tableHeader - nagłówek wyświetlanej tabeli

pole pracownicy (tabela typu Pracownik) - treść wyświetlanej tabeli

Angular ver.14

Dyrektywa strukturalna ngFor

dyrektywa iteruje po tablicy pracownicy przypisując kolejne obiekty typu pracownik do zmiennej osoba, poszczególne pola tej zmiennej (obiektu) wstawiane są do znaczników <td>

app.component.html M

dyrektywaNgFor > src > app > app.component.html > ...

```
1  <table>
2    <tr>
3      <th *ngFor="let header of tableHeader">{{ header }}</th>
4    </tr>
5    <tr *ngFor="let osoba of pracownicy">
6      <td>{{ osoba.imie }}</td>
7      <td>{{ osoba.adres }}</td>
8      <td>{{ osoba.wiek }} lat</td>
9      <td>{{ osoba.stanowisko}}</td>
10     <td>{{ osoba.zarobki}} zł</td>
11   </tr>
12 </table>
```

dyrektywa iteruje po tablicy stringów tableHeader wstawiając w miejsce header poszczególne nagłówki tablicy

DyrektywaNgForZmienne

localhost:4200

imię	adres	wiek stanowisko	zarobki
Jan	Łódź	26 lat mechanik	3000 zł
Anna	Stryków	40 lat sekretarka	2700 zł
Witold	Pabianice	30 lat lakiernik	3500 zł
Wiktoria	Łódź	24 lat kierownik	4000 zł
Adam	Konstantynów	34 lat kierowca	3000 zł

Angular ver.14

Dyrektywa strukturalna
ngFor

dyrektywaNgFor_galeria






zdjecia/01.jpg

zdjecia/02.jpg

zdjecia/03.jpg

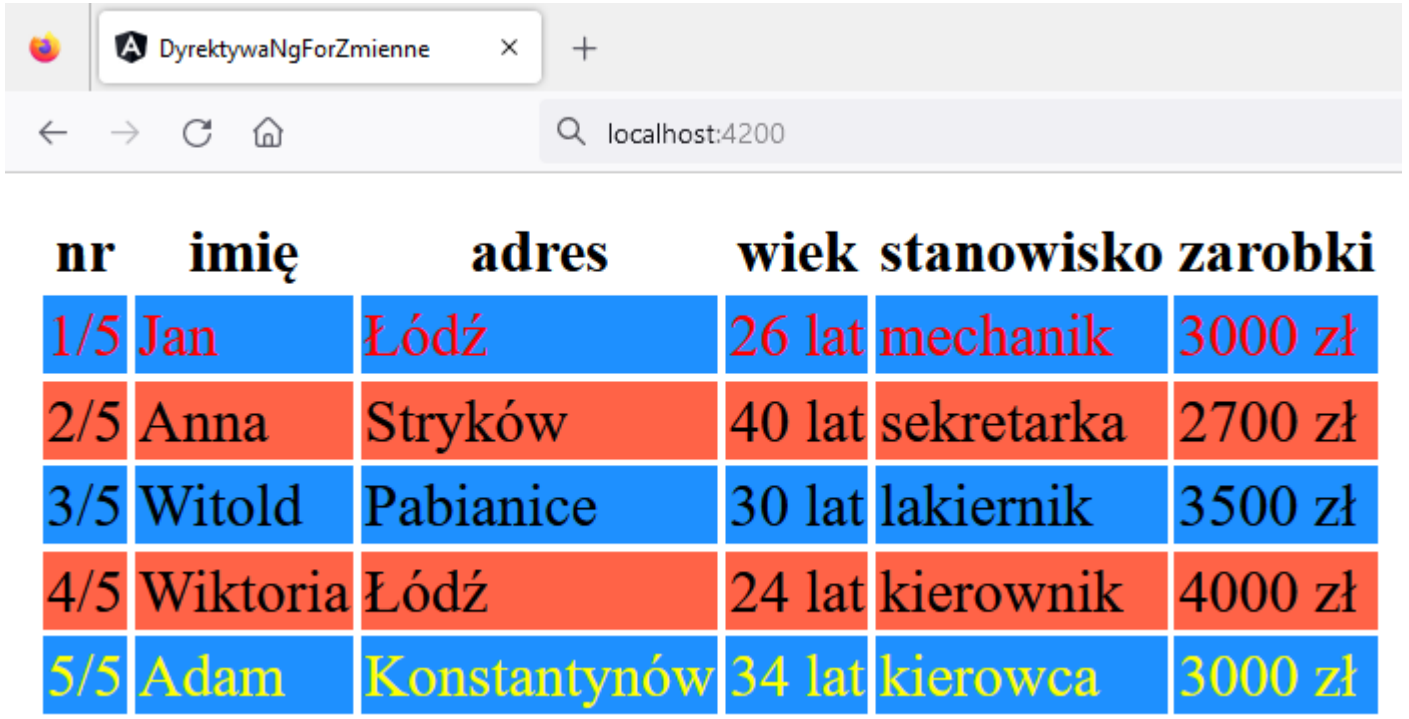
zdjecia/04.jpg

za pomocą dyrektywy ngFor wyświetlić na stronie galerię zdjęć oraz nazwy zdjęć

Dyrektywa strukturalna
ngFor, zmienne

Nazwa zmiennej	opis
index: number	indeks bieżącego elementu w kolekcji
odd: boolean	true – bieżący element przyjmuje nieparzysty indeks w iterowanym obiekcie
even: boolean	true – bieżący element przyjmuje parzysty indeks w iterowanym obiekcie
first: boolean	true – bieżący element jest pierwszym elementem w iterowanym obiekcie
last: boolean	true – bieżący element jest ostatnim elementem w iterowanym obiekcie
count: number	ilość iterowanych elementów

Dyrektywa strukturalna
ngFor, zmienne



za pomocą zmiennych dyrektywy ngFor można zmienić wygląd tabeli z poprzedniego przykładu

Dyrektywa strukturalna
ngFor, zmienne

app.component.html M

dyrektywaNgForZmienne > src > app > app.component.html > ...

```
1  <table>
2    <tr>
3      <th *ngFor="let header of tableHeader">{{ header }}</th>
4    </tr>
5    <tr *ngFor="let osoba of pracownicy;
6                let i = index;
7                let evenRow = even;
8                let oddRow = odd;
9                let firstRow = first;
10               let lastRow = last;
11               let countRow = count"
12               [class.evenRow] = "evenRow"
13               [class.oddRow] = "oddRow"
14               [class.firstRow] = "firstRow"
15               [class.lastRow] = "lastRow"
16               >
17       <td>{{i + 1}}/{{countRow}}</td>
18       <td>{{ osoba.imie }}</td>
19       <td>{{ osoba.adres }}</td>
20       <td>{{ osoba.wiek }} lat</td>
21       <td>{{ osoba.stanowisko}}</td>
22       <td>{{ osoba.zarobki}} zł</td>
23   </tr>
24 </table>
```

zmienne lokalne

zmienne wykorzystane do zmiany klasy

Angular ver.14

Dyrektywa strukturalna
ngFor, zmienne

TS app.component.ts M

dyrektywaNgForZmienne > src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';
import { Pracownik } from './interface';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  // nagłówek tabeli
  tableHeader: string[] = ['nr','imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
  // tabela z pracownikami
  pracownicy: Pracownik[] = [
    {imie: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
    {imie: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
    {imie: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
    {imie: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
    {imie: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
  ];
}
```

doszła nowa kolumna na indeks

Angular ver.14

Dyrektywa strukturalna ngFor, zmienne

app.component.css M

dyrektywaNgForZmienne > src > app > # app.component.css >

```css
.evenRow{
    background-color: DodgerBlue;
}
.oddRow{
    background-color: Tomato;
}
.firstRow{
    color: red;
}
.lastRow{
    color: yellow;
}
.countRow{
    text-align:right;
}
```

style tabeli załączane zmiennymi dyrektywy ngFor

Angular ver.14

@for template syntax

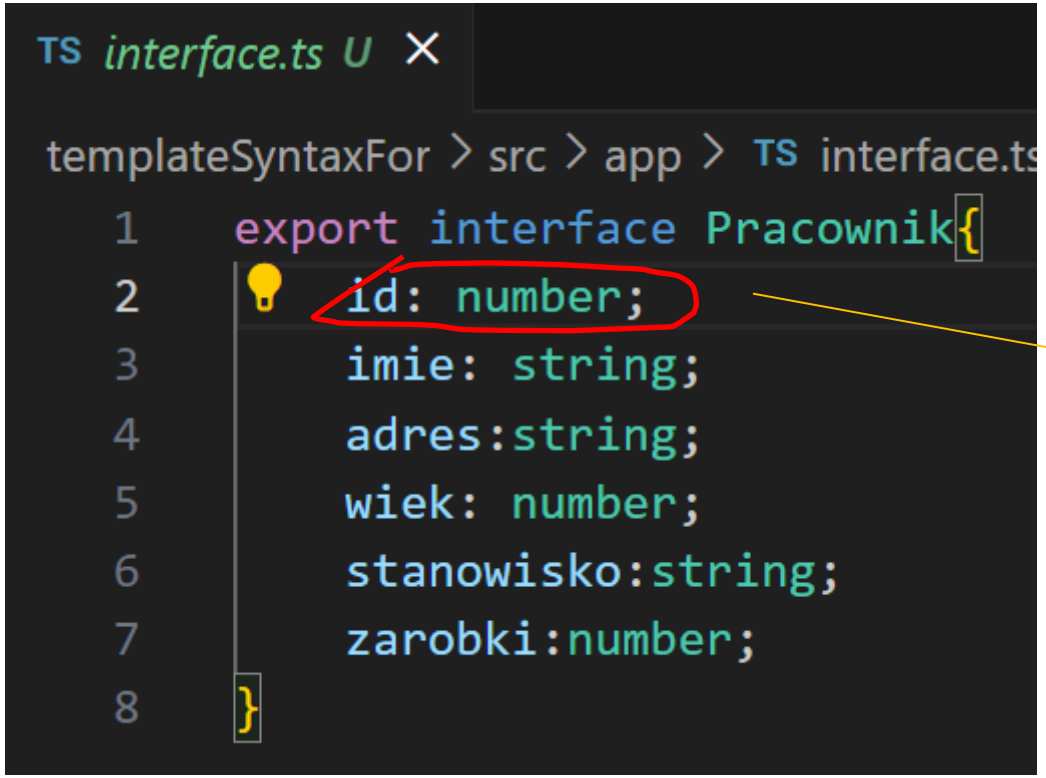
składnia pętli for @for wprowadzona w wersji 17 Angulara

Wartość wyrażenia ***track*** określa klucz używany do kojarzenia elementów tablicy z widokami w DOM.

```
@for(osoba of pracownicy; track osoba.id){
  <tr>
    <td>{{ osoba.id }}</td>
    <td>{{ osoba.imie }}</td>
    <td>{{ osoba.adres }}</td>
    <td>{{ osoba.wiek }} lat</td>
    <td>{{ osoba.stanowisko}}</td>
    <td>{{ osoba.zarobki}} zł</td>
  </tr>
}
```

każdy element kolekcji powinien mieć indeks (klucz)

@for template syntax



dodajemy indeks do interfejsu

@for template syntax

TS app.component.ts M

templateSyntaxFor > src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import {Pracownik } from './interface';
//import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'templateSyntaxFor';
  tableHeader: string[] = ['id', 'imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
  pracownicy: Pracownik[] = [
    {id: 0, imie: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
    {id: 1, imie: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
    {id: 2, imie: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
    {id: 3, imie: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
    {id: 4, imie: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
  ];

  isVisible:boolean = false;
  toggle(){
    this.isVisible = !this.isVisible;
  }
}
```

nie jest potrzebny CommonModule

można dodać kolumnę identyfikacyjną

Angular ver.18

@for template syntax

app.component.html M

templateSyntaxFor > src > app > app.component.html > ...

```
1  <table>
2    @if(isVisible){
3      <tr>
4        @for(header of tableHeader; track $index){
5          <th> {{ header }}</th>
6        }
7      </tr>
8    }@else{
9      <tr>
10       <th>else</th>
11       <th>else</th>
12       <th>else</th>
13       <th>else</th>
14       <th>else</th>
15       <th>else</th>
16     </tr>
17   }
18     @for(osoba of pracownicy; track osoba.id){
19       <tr>
20         <td>{{ osoba.id }}</td>
21         <td>{{ osoba.imie }}</td>
22         <td>{{ osoba.adres }}</td>
23         <td>{{ osoba.wiek }} lat</td>
24         <td>{{ osoba.stanowisko}}</td>
25         <td>{{ osoba.zarobki}} zł</td>
26       </tr>
27     }
28 </table>
29 <button (click) = toggle()>przełącz</button>
```

ta część tabeli pojawia się, gdy zmienna ***isVisible*** przyjmuje wartość ***true***

ta część tabeli pojawia się, gdy zmienna ***isVisible*** przyjmuje wartość ***false***

tabelę z danymi pracowników tworzymy wykorzystując składnię @for

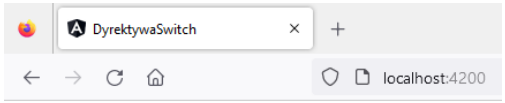
indeks bieżącego elementu w kolekcji

w tablicy tableHeader brak indeksu, dlatego można skorzystać ze zmiennej $index

Angular ver.18

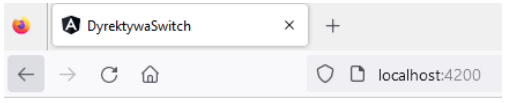
Dyrektywa strukturalna ngSwitch

przełacznik



2

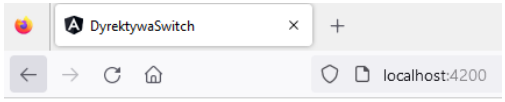
dwa



1

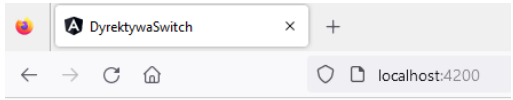
jeden

po przeładowaniu strony wyświetlają się losowe liczby w zakresie od 1 do 6



4

cztery



6

default

dzięki dyrektywie ngSwitch
wyświetlana jest literowa nazwa liczby

Dyrektywa strukturalna ngSwitch

TS app.component.ts M ×

dyrektywaSwitch > src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  // liczba losowa z zakresu od 1 do 6
  random: number =  Math.floor(Math.random()*6+1);
}
```

po przeładowaniu strony do pola random przypisują się losowe liczby w zakresie od 1 do 6
pole jest przypisane do dyrektywy ngSwitch

Dyrektywa strukturalna ngSwitch

app.component.html M

dyrektywaSwitch > src > app > app.component.html > ...

```
1   <h1>{{random}}</h1>
2
3   <ng-container [ngSwitch] = "random">
4       <p *ngSwitchCase="1">jeden</p>
5       <p *ngSwitchCase="2">dwa</p>
6       <p *ngSwitchCase="3">trzy</p>
7       <p *ngSwitchCase="4">cztery</p>
8       <p *ngSwitchCase="5">pięć</p>
9       <p *ngSwitchDefault>default</p> <!-- gdy wypadnie 6-->
10  </ng-container>
```

wypisanie losowej liczby

pole random jest przypisane do dyrektywy ngSwitch

wartość domyślna

kontener wyświetlający jeden z paragrafów

dyrektywa ngSwitch przełącza literową nazwę w zależności od wylosowanej liczby w polu random

@switch template syntax

```
@switch (condition) {
  @case (caseA) {
    Case A.
  }
  @case (caseB) {
    Case B.
  }
  @default {
    Default case.
  }
}
```

https://angular.dev/api/core/@switch

@switch template syntax

app.component.html M

templateSyntaxSwitch > src > app > app.component.html > ...

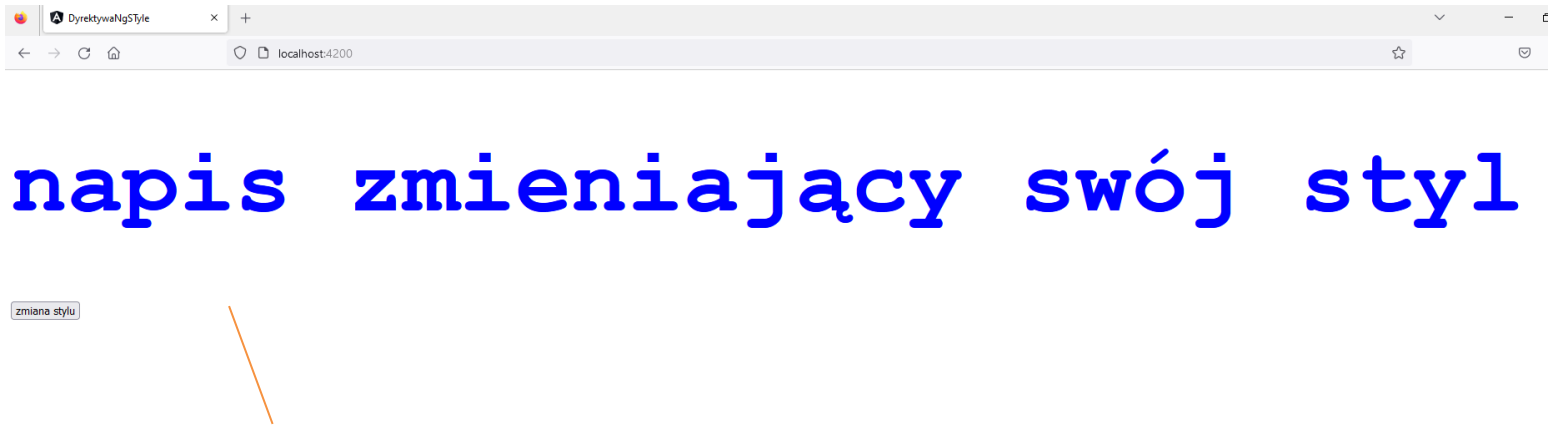
Go to component

```
1  <h1>{{random}}</h1>
2
3  @switch (random){
4    @case (1) {<p>jeden</p>}
5    @case (2) {<p>dwa</p>}
6    @case (3) {<p>trzy</p>}
7    @case (4) {<p>cztery</p>}
8    @case (5) {<p>pięć</p>}
9    @default {<p> default: sześć</p>} <!-- gdy wypadnie 6-->
10 }
```

Angular ver.18

Dyrektywa atrybutowa ngStyle

zmiana stylu elementu



zmiana stylu napisu za pomocą dyrektywy ngStyle

Dyrektywa atrybutowa ngStyle

zmiana stylu napisu za pomocą dyrektywy ngStyle

app.component.html

dyrektywaNgStyle > src > app > app.component.html > button

```
1    <h1 [ngStyle]="{'color' : colors[index],
2                    'font-family' : fonts[index],
3                    'font-size.%' : index*200 + 100
4                     }">
5                     napis zmieniający swój styl</h1>
6    <button type="button" (click)="increment()">zmiana stylu</button>
```

przycisk zmieniający styl napisu

Angular ver.18

Dyrektywa atrybutowa ngStyle

app.component.ts M

dyrektywaNgStyle > src > app > app.component.ts > AppComponent

```ts
import { NgStyle } from '@angular/common';
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, NgStyle],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'dyrektywaNgStyle';
  index: number = 0;
  colors: string[] =["red", "yellow", "green", "blue"];
  fonts: string[]=["Times New Roman", "Arial","Lucida Handwriting",
                   "Courier New"];
  increment(){
    // 0 - 3
    this.index = Math.floor(Math.random() * 4);
    console.log(this.index);
  }
}
```

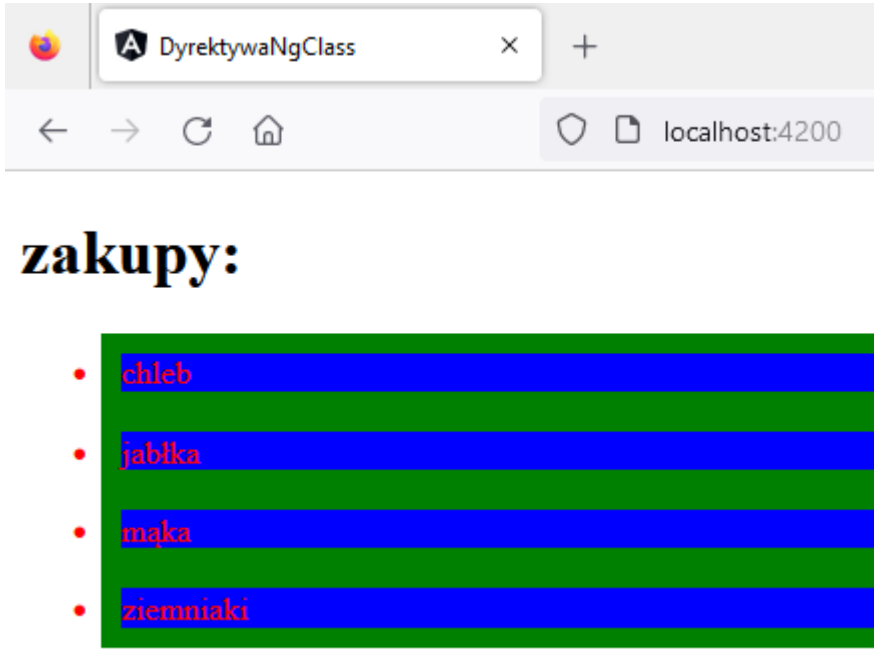
kolory i czcionki wybieramy z tablic

indeksem w tablicy jest liczba losowa zmieniana przyciskiem

Angular ver.18

Dyrektywa atrybutowa ngClass

ustawia klasę elementu



klasa listy jest ustawiona za pomocą dyrektywy ngStyle

Dyrektywa atrybutowa ngClass

style

app.component.css M

dyrektywaNgClass > src > app > # app.component.cs

```
1  .red{
2      color: red;
3  }
4  .blue{
5      background-color: blue;
6  }
7  .green{
8      border: 10px solid green
9  }
```

app.component.ts M

dyrektywaNgClass > src > app > TS app.component.ts > ...

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9      zakupy: string[] = ["chleb", "jabłka", "mąka", "ziemniaki"];
10 }
```

tablica z zakupami

app.component.html M

dyrektywaNgClass > src > app > app.component.html > ...

```
1  <h1>zakupy:</h1>
2  <ul>
3    <li *ngFor="let item of zakupy" [ngClass] = "'red green blue'">{{item}}</li>
4  </ul>
```

iteracja przez tablicę zakupów

ustawienie klasy znacznika

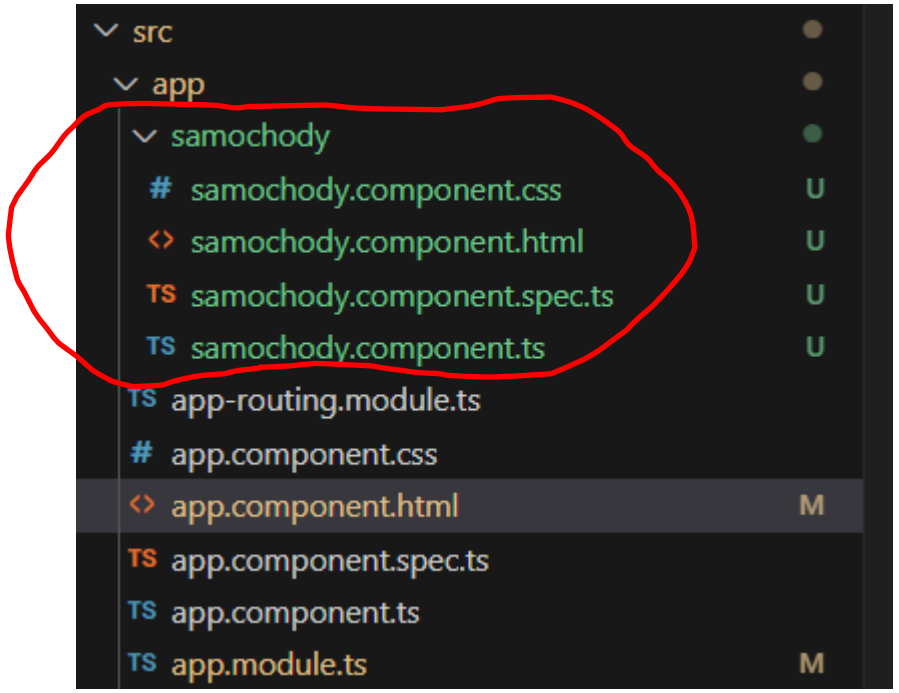
utworzenie nowego komponentu samochody (aplikacja z komponentami typu no-standalone)

polecenie utworzenia komponentu samochody:
g: generate
c: component
(wydajemy je z wnętrza katalogu projektu!)

```
ng g c samochody
```

```
CREATE src/app/samochody/samochody.component.html (25 bytes)
CREATE src/app/samochody/samochody.component.spec.ts (645 bytes)
CREATE src/app/samochody/samochody.component.ts (221 bytes)
CREATE src/app/samochody/samochody.component.css (0 bytes)
UPDATE src/app/app.module.ts (487 bytes)
```

rejestracja nowego komponentu w app.module.ts (w NgModule)



wygenerowane pliki

utworzenie nowego komponentu samochody (aplikacja z komponentami typu no-standalone)

samochody.component.ts U

_hello > my-app > src > app > samochody > TS samochody.component.ts > ...

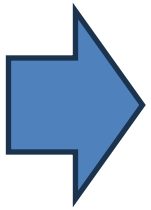
```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-samochody',
  templateUrl: './samochody.component.html',
  styleUrl: './samochody.component.css'
})
export class SamochodyComponent {
  auta: string = "tu pojawią się najnowsze samochody";
}
```

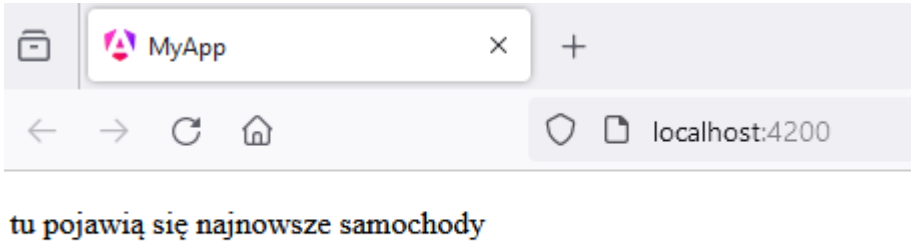
samochody.component.html U

_hello > my-app > src > app > samochody >

Go to component

```html
<p>{{auta}}</p>
```





app.component.html M

_hello > my-app > src > app > app.component.html > app-samochody

Go to component

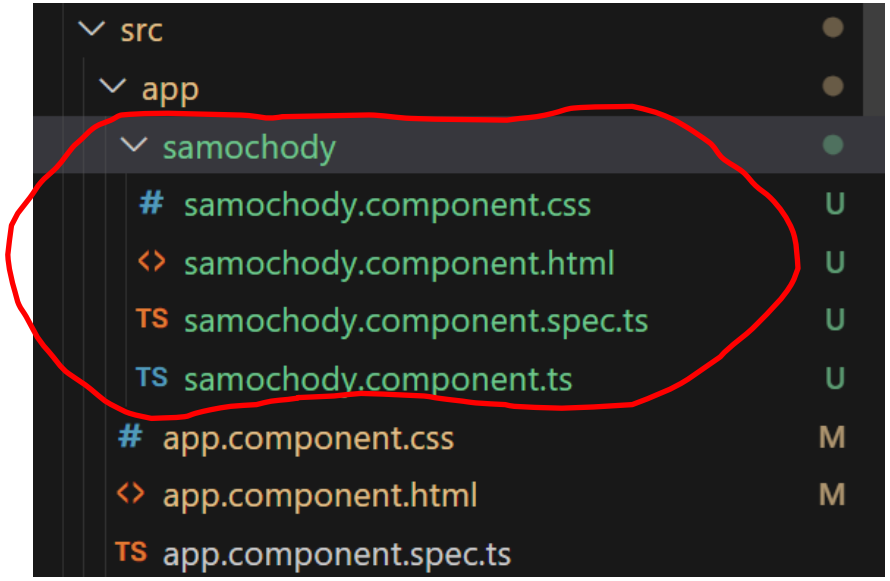
```html
<app-samochody></app-samochody>
```

utworzenie nowego komponentu samochody (aplikacja z komponentami typu standalone)

```
ng g c samochody
```

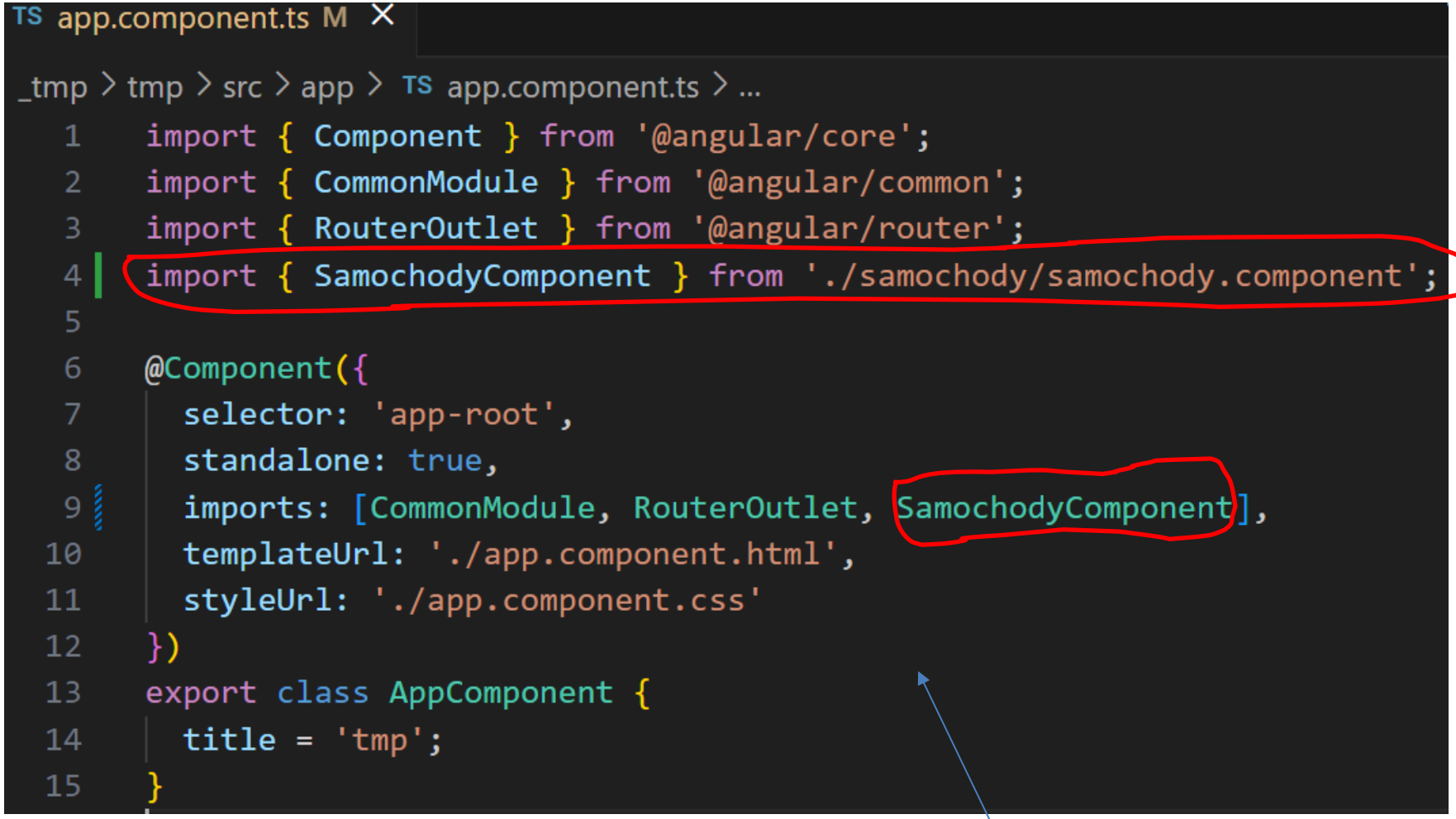
polecenie utworzenia komponentu samochody:
g: generate
c: component
(wydajemy je z wnętrza katalogu projektu!)

```
CREATE src/app/samochody/samochody.component.html (25 bytes)
CREATE src/app/samochody/samochody.component.spec.ts (640 bytes)
CREATE src/app/samochody/samochody.component.ts (258 bytes)
CREATE src/app/samochody/samochody.component.css (0 bytes)
```



wygenerowane składniki komponentu
(brak rejestracji nowego komponentu
– komponent jest typy standallone)

utworzenie nowego komponentu samochody (aplikacja z komponentami typu standalone)



import samodzielnego komponentu SamochodyComponent do komponentu App.Component w pliku app.component.ts

Angular ver.18

utworzenie nowego komponentu samochody (aplikacja z komponentami typu standalone)

samochody.component.ts U

_tmp > tmp > src > app > samochody > TS samochody.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-samochody',
  standalone: true,
  imports: [],
  templateUrl: './samochody.component.html',
  styleUrl: './samochody.component.css'
})
export class SamochodyComponent {
  auta: string = "tu pojawią się najnowsze samochody";
}
```

samochody.component.html U

_tmp > tmp > src > app > samochody > samochody.component.html >

Go to component

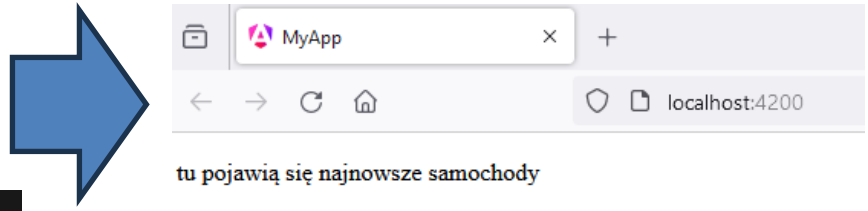
```html
<p>{{auta}}</p>
```

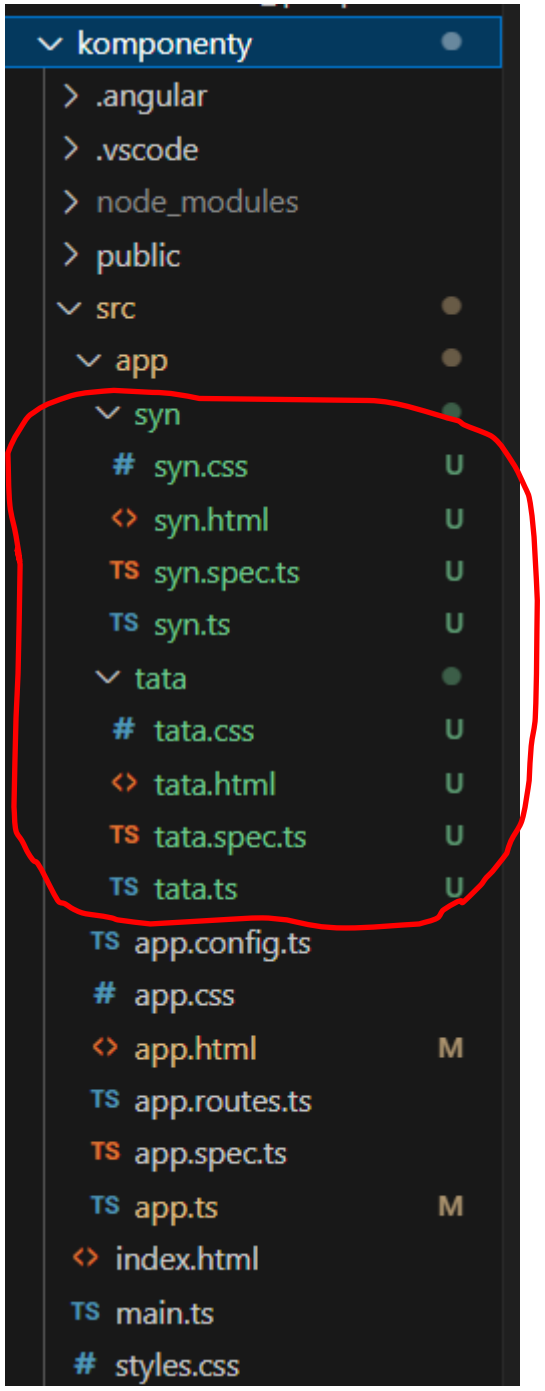
app.component.html M

_tmp > tmp > src > app > app.component.html > ...

Go to component

```html
<app-samochody></app-samochody>
```





Komponenty – przesyłanie danych

każdy nowy komponent posiada pliki: css, html, type script oraz testowy

budują aplikację

generujemy nowe komponenty tata i syn

```
ng g c syn
```

```
ng g c tata
```

generate component nazwa komponentu

pliki komponentu głównego, korzenia drzewa komponentów

Komponenty – przesyłanie danych

app.ts 1, M

komponenty > src > app > app.ts > ...

```ts
...
import { Component, signal } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import {Tata} from './tata/tata';

...
@Component({
  selector: 'app-root',
  imports: [RouterOutlet, Tata],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
export class App {
  protected readonly title = signal('komponenty');
}
```

tata.ts U

komponenty > src > app > tata > tata.ts > ...

```ts
import { Component } from '@angular/core';
import { Syn } from  "../syn/syn"

@Component({
  selector: 'app-tata',
  imports: [Syn],
  templateUrl: './tata.html',
  styleUrl: './tata.css',
})
export class Tata {
    biezaceInfo:string = "To jest info od Taty";
}
```

w głównym komponencie app importujemy component taty

komponencie taty importujemy component syna

Komponenty – przesyłanie danych

app.html M

komponenty > src > app > app.html > ...

Go to component | You, 1 second ago | 1 author (You)

```
1  <app-tata></app-tata>
```

w głównym komponencie app zagnieżdżamy komponent tata (komponent główny jest nadrzędny, tata jest podrzędny)

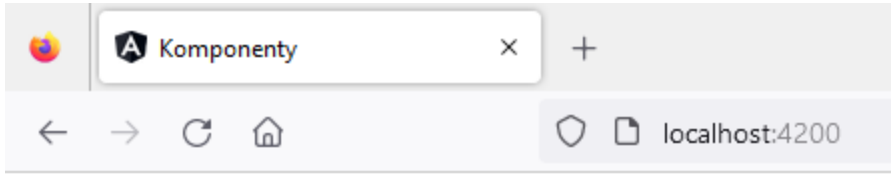
tata.html U

komponenty > src > app > tata > tata.html > ...

Go to component

```
1  <p>tata works!</p>
2  <app-syn></app-syn>
```

w komponencie tata zagnieżdżamy komponent syn (komponent tata jest nadrzędny, syn jest podrzędny)

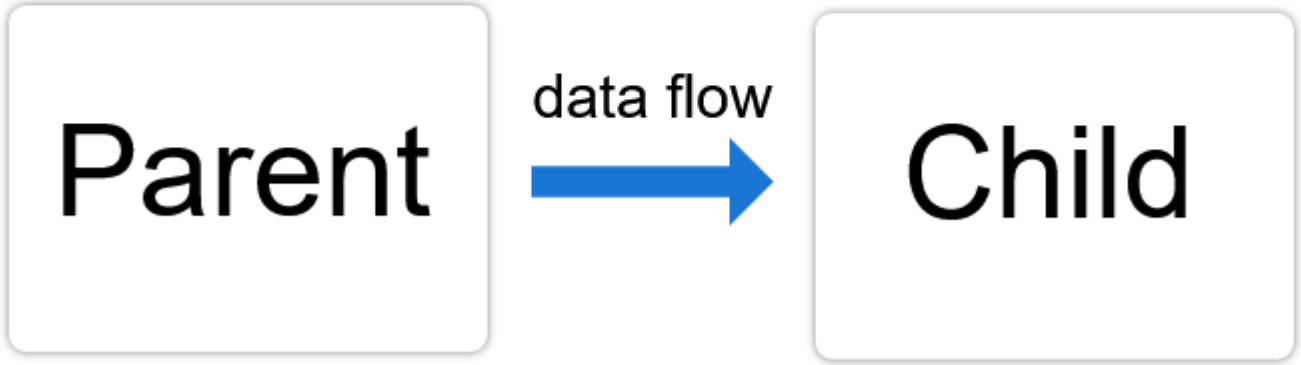


komponenty tata i syn (oraz główny) działają

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

@Input



https://angular.io/guide/inputs-outputs

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

powiazanie pól komponentu nadrzędnego (tata) z podrzędnym (syn)

tata.component.html

selektor komponentu podrzędnego

target

source

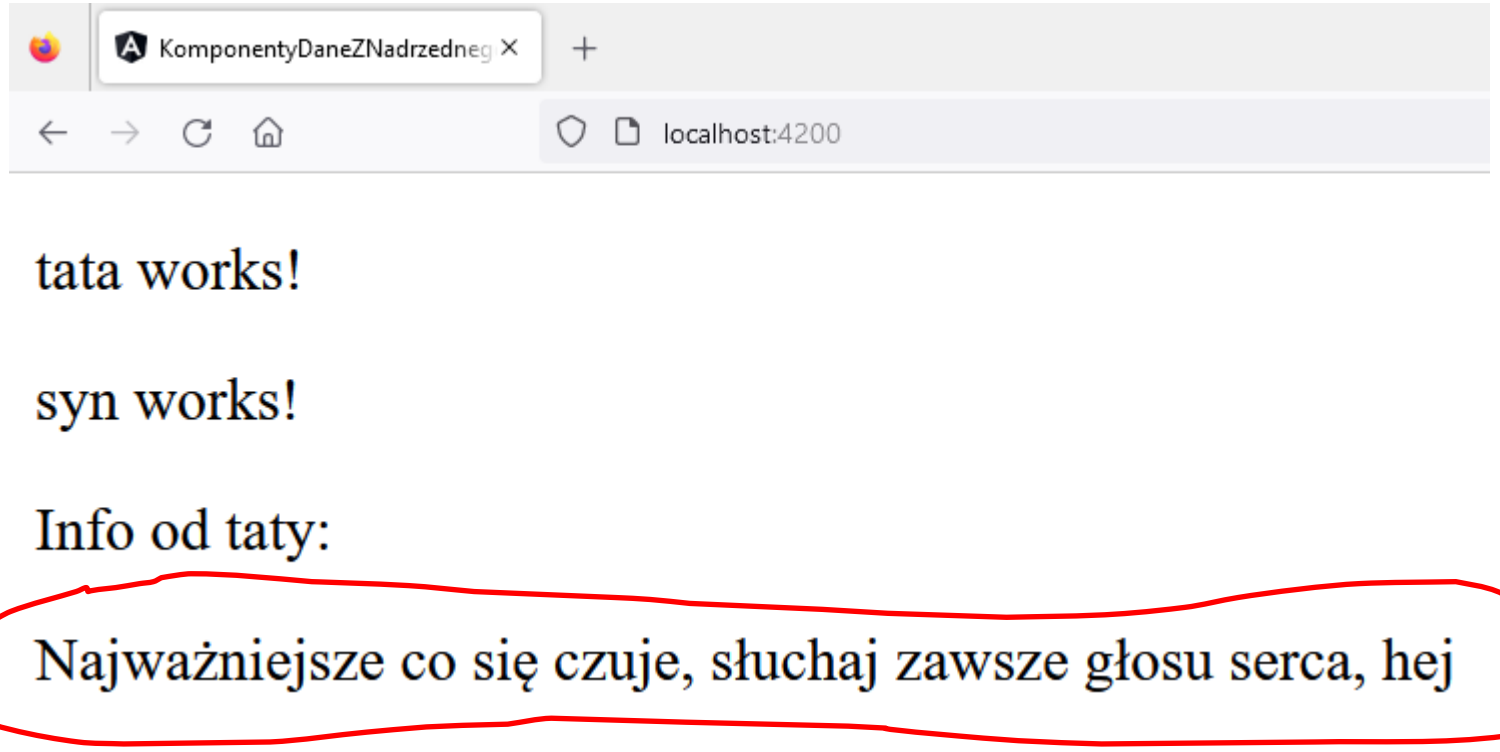
```
<app-syn [infoOdTaty] = "bierzaceInfo"></app-syn>
```

pole @Input komponentu podrzędnego (syna)

pole komponentu nadrzędnego (taty)

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)



informacja przesłana z komponentu nadrzędnego (tata) do komponentu podrzędnego (syn)

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

1.konfiguracja komponentu podrzędnego (syn)

syn.ts U

komponenty > src > app > syn > TS syn.ts > ...

```ts
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-syn',
  imports: [],
  templateUrl: './syn.html',
  styleUrl: './syn.css',
})
export class Syn {
  @Input() infoOdTaty: string = "";
}
```

import komponentu Input

u syna tworzymy pole infoOdTaty z dekoratorem @Input()

syn.html U

komponenty > src > app > syn > syn.html > ...

Go to component

```html
<p>syn works!</p>
<p>{{infoOdTaty}}</p>
```

Angular ver.20

w pliku html syna posługujemy się interpolacją

https://angular.io/guide/inputs-outputs#configuring-the-child-component

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

2.konfiguracja komponentu nadrzędnego (tata)

tata.ts U

komponenty > src > app > tata > tata.ts > ...

```ts
import { Component } from '@angular/core';
import { Syn } from  "../syn/syn"

@Component({
  selector: 'app-tata',
  imports: [Syn],
  templateUrl: './tata.html',
  styleUrl: './tata.css',
})
export class Tata {
  biezaceInfo:string = "Najważniejsze co się czuje, słuchaj zawsze głosu swego serca, hej";
}
```

u taty tworzymy pole biezaceInfo

tata.html U

komponenty > src > app > tata > tata.html > ...

Go to component

```html
<p>tata works!</p>
<app-syn [infoOdTaty]="biezaceInfo"></app-syn>
```

w pliku html taty umieszczamy wiązanie pola syna infoOdTaty z polem taty bierzaceInfo

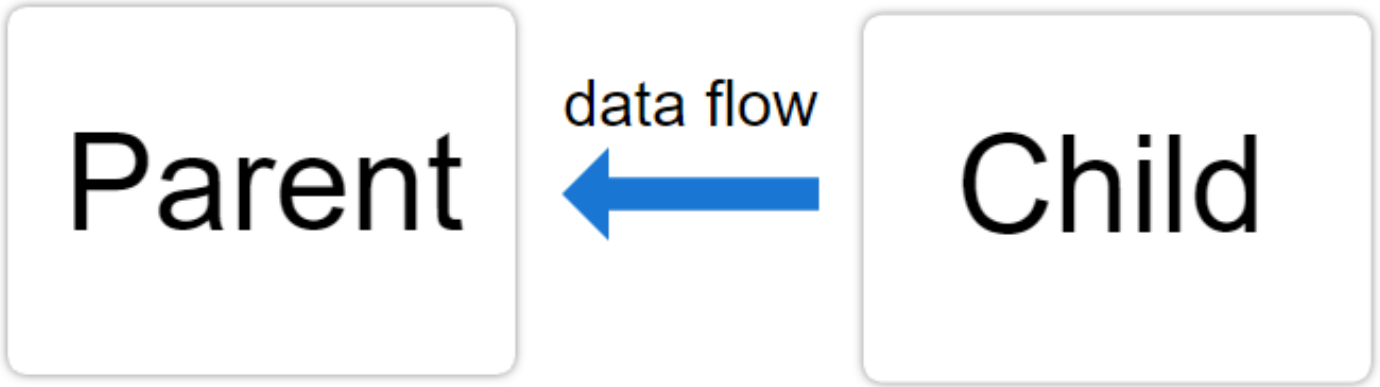
Angular ver.20

https://angular.io/guide/inputs-outputs#configuring-the-parent-component

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

@Output



https://angular.io/guide/inputs-outputs#sending-data-to-a-parent-component

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

powiazanie pól komponentu podrzędnego (syn) z nadrzędnym (tata) (i na odwrót)

tata.component.html

selektor komponentu podrzędnego

source

target

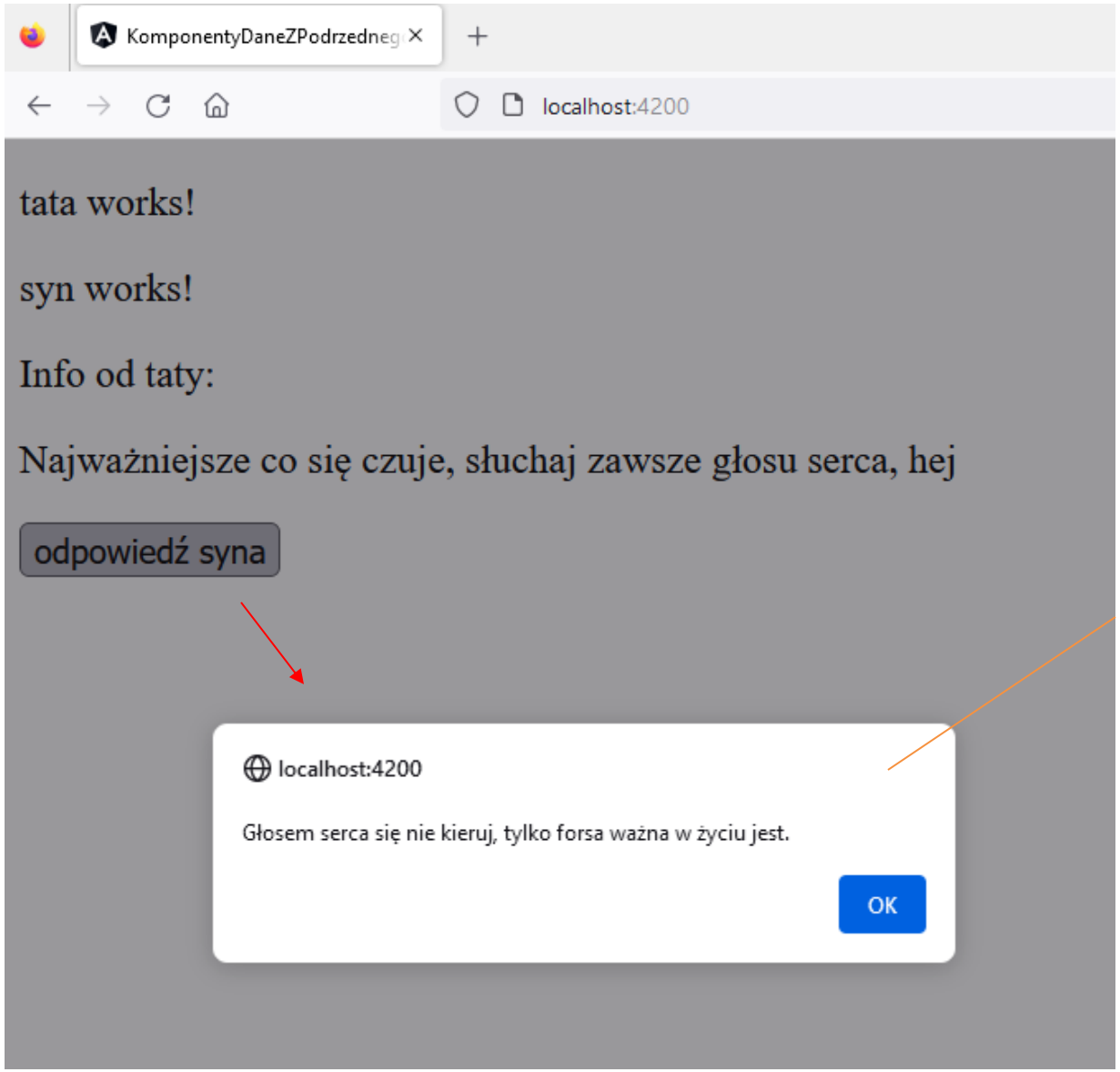
```
<app-syn [infoOdTaty] = "biezaceInfo"  (odpowiedzSyna) = "onOdpowiedzSyna($event)"></app-syn>
```

event – pole @Output komponentu podrzędnego (syna)

handler komponentu nadrzędnego (taty)

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)



informacja przesłana z komponentu podrzędnego (syn) do komponentu nadrzędnego (tata)

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

1.konfiguracja komponentu podrzędnego (syn)

TS syn.ts U

komponenty > src > app > syn > TS syn.ts > ...

```ts
import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-syn',
  imports: [],
  templateUrl: './syn.html',
  styleUrl: './syn.css',
})
export class Syn {
  @Input() infoOdTaty: string = "";
  @Output() odpowiedzSyna = new EventEmitter<string>();

  odpowiedz(){
    this.odpowiedzSyna.emit('Głosem serca się nie kieruj, tylko forsa ważna w życiu jest!');
  }
}
```

import komponentów Output, EventEmiter (ten drugi służy do przekazywania danych)

będziemy eksportować dane typu string

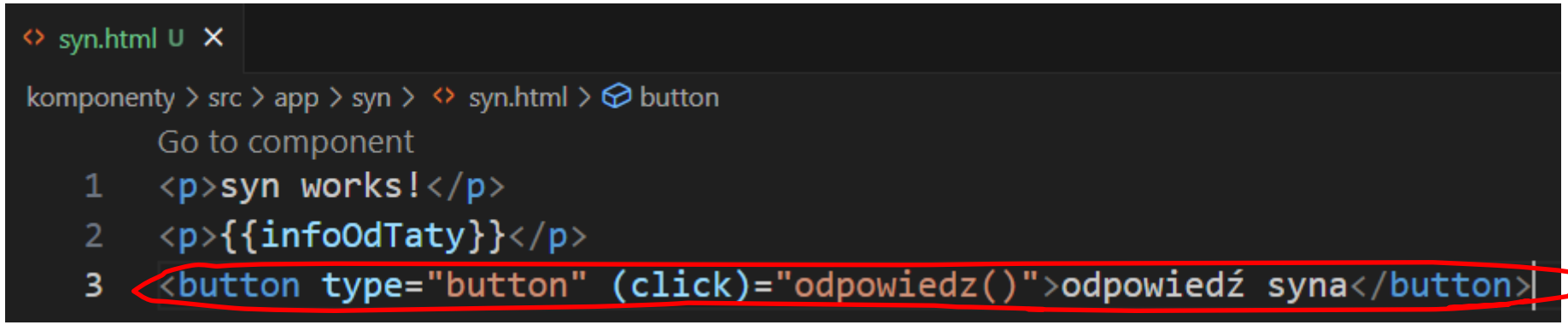
u syna tworzymy pole odpowiedzSyna (zdarzenie, obiekt typu EventEmiter) z dekoratorem @Output

aby przekazać (wyeksportować) dane musimy użyć metody emit(), metoda odpowiedz() jest trigerowana przez naciśnięcie przycisku

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

1.konfiguracja komponentu podrzędnego (syn)



aby wyeksportować dane do komponentu nadrzędnego (tata)
potrzebne jest zdarzenie, w tym przypadku kliknięcie przycisku,
zdarzenie click jest zbindowane z metodą odpowiedz()

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

2.konfiguracja komponentu nadrzędnego (tata)

tata.html U

komponenty > src > app > tata > tata.html > ...

Go to component

```
1  <p>tata works!</p>
2  <app-syn [infoOdTaty]="biezaceInfo" (odpowiedzSyna)="onOdpowiedzSyna($event)"></app-syn>
```

tata.ts U

komponenty > src > app > tata > tata.ts > ...

```
1  import { Component } from '@angular/core';
2  import { Syn } from  "../syn/syn"
3
4  @Component({
5    selector: 'app-tata',
6    imports: [Syn],
7    templateUrl: './tata.html',
8    styleUrl: './tata.css',
9  })
10 export class Tata {
11     biezaceInfo:string = "Najważniejsze co się czuje, słuchaj zawsze głosu swego serca, hej";
12
13     onOdpowiedzSyna(informacja: string){
14     alert(informacja);
15   }
16 }
```

definicja metody (handlera) onOdpowiedzSyna($event),

po kliknięciu na przycisk emitowane jest zdarzenie odpowiedzSyna obsługiwane przez metodę taty (handlera) onOdpowiedzSyna($event), $event zawiera przekazywane dane do taty

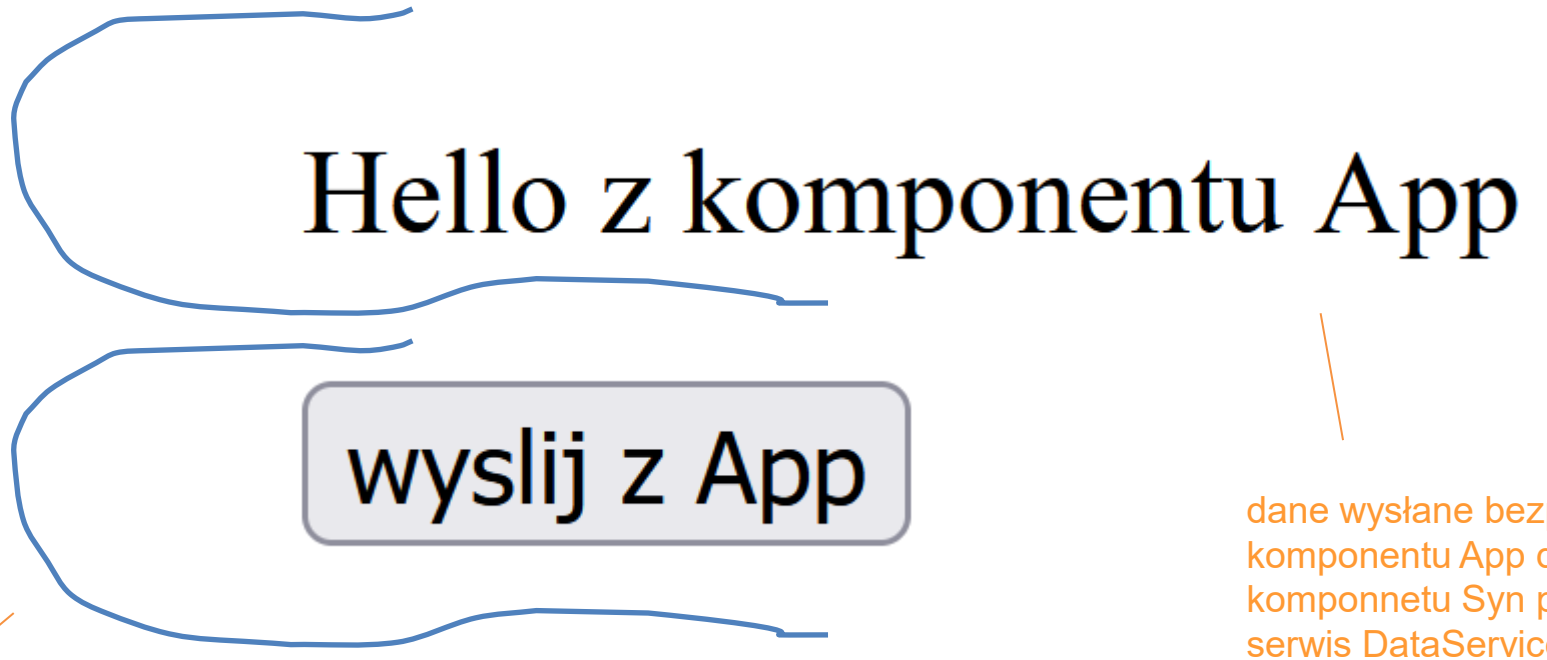
Angular ver.20

https://angular.io/guide/inputs-outputs#configuring-the-parent-component-1

Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

z komponentu App do Syn

komponnet Syn



komponnet App

dane wysłane bezposrednio z komponentu App do komponnetu Syn poprzez serwis DataService

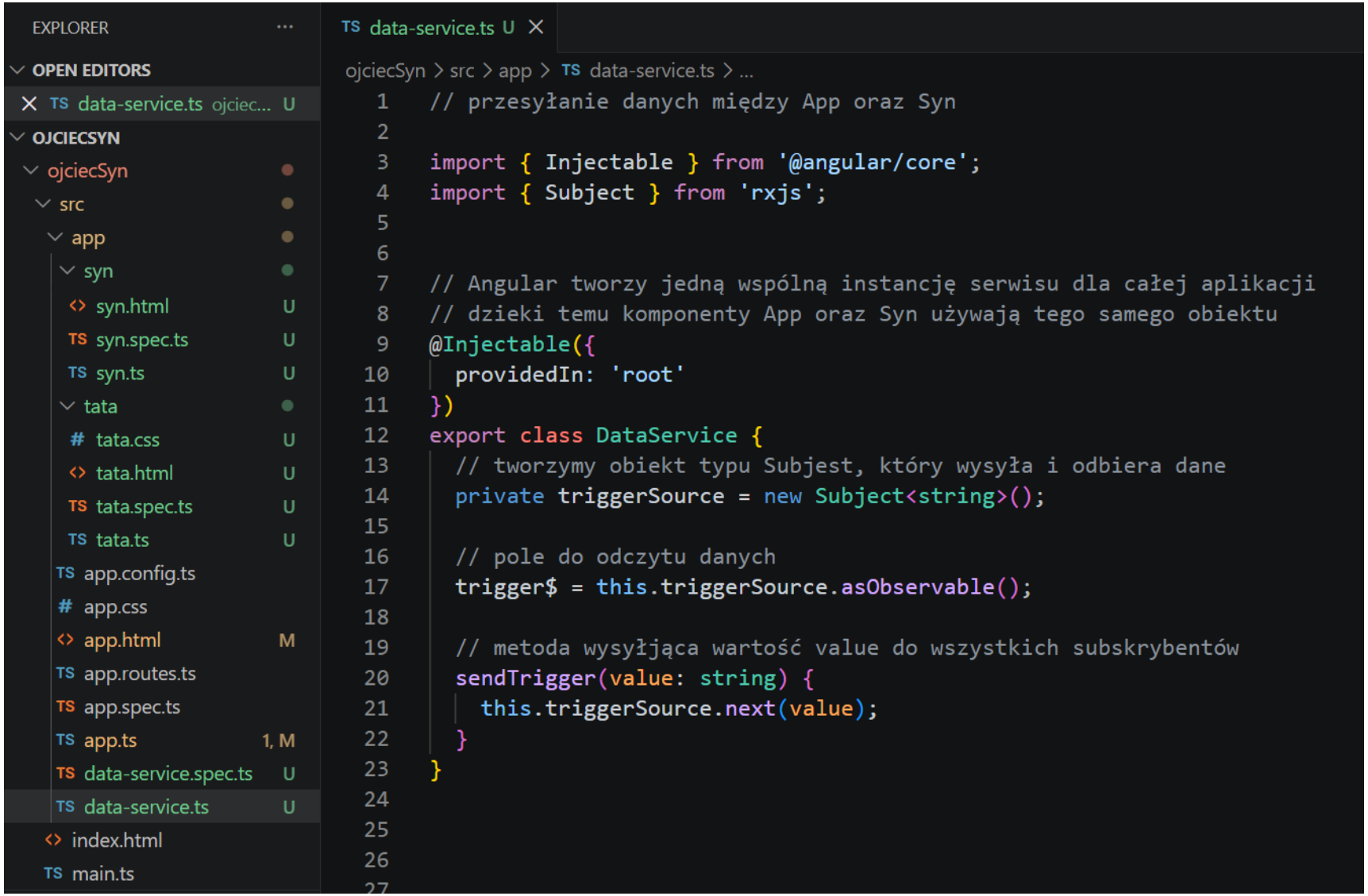
Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

z komponentu App do Syn

data-service.ts

```
ng g s dataService
```

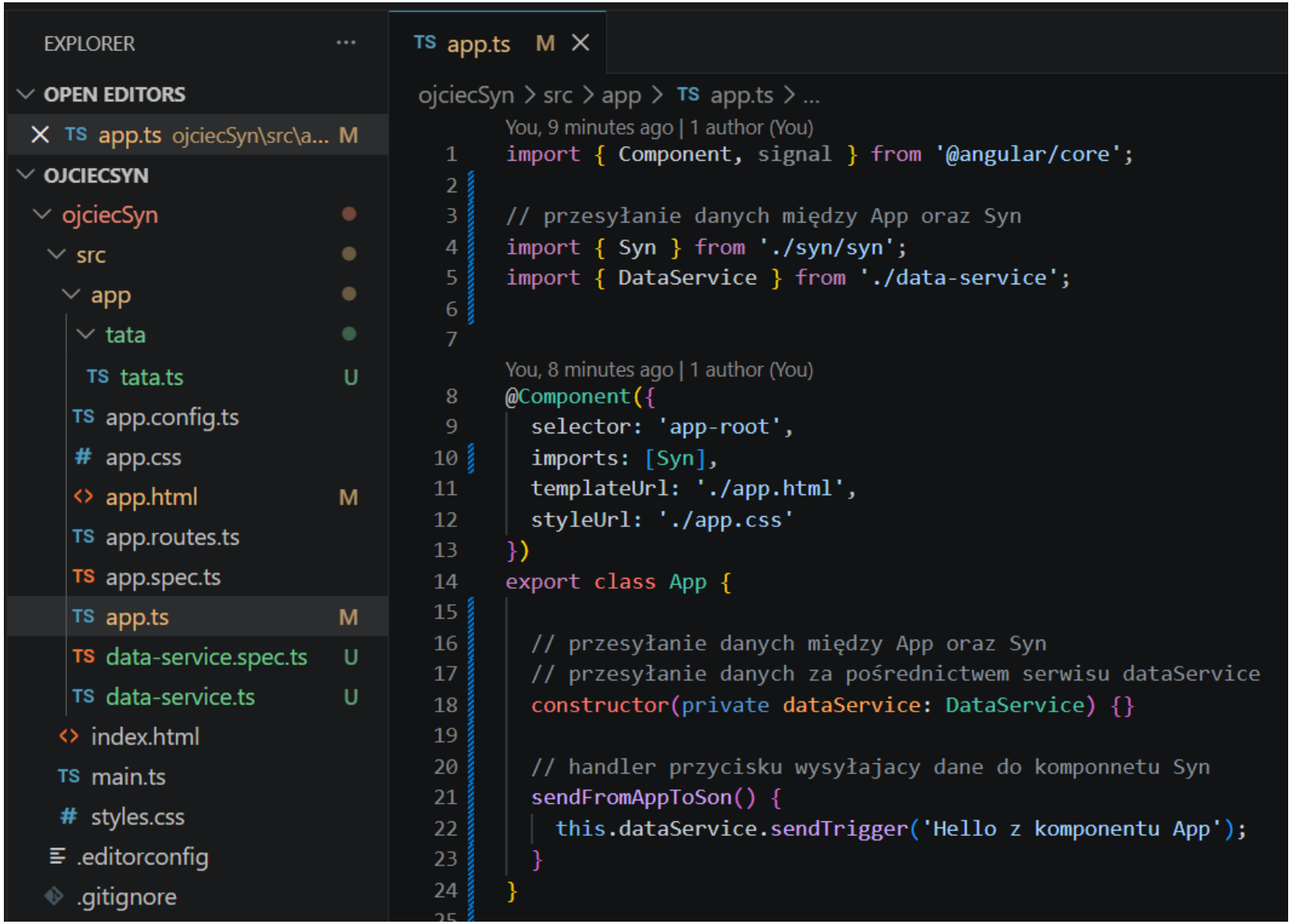
tworzymy serwis DataService



Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

app.ts

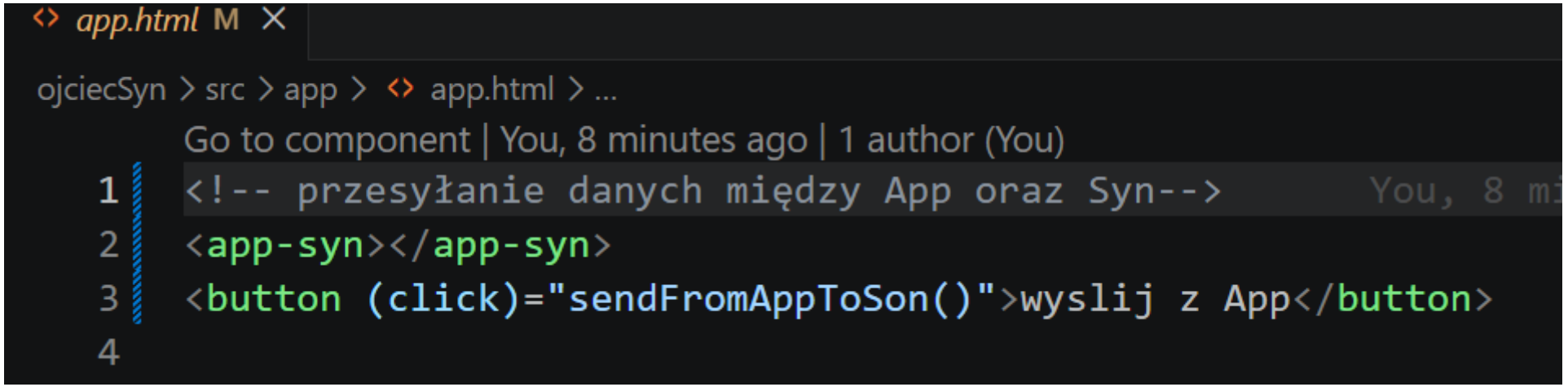
z komponentu App do Syn



Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

z komponentu App do Syn

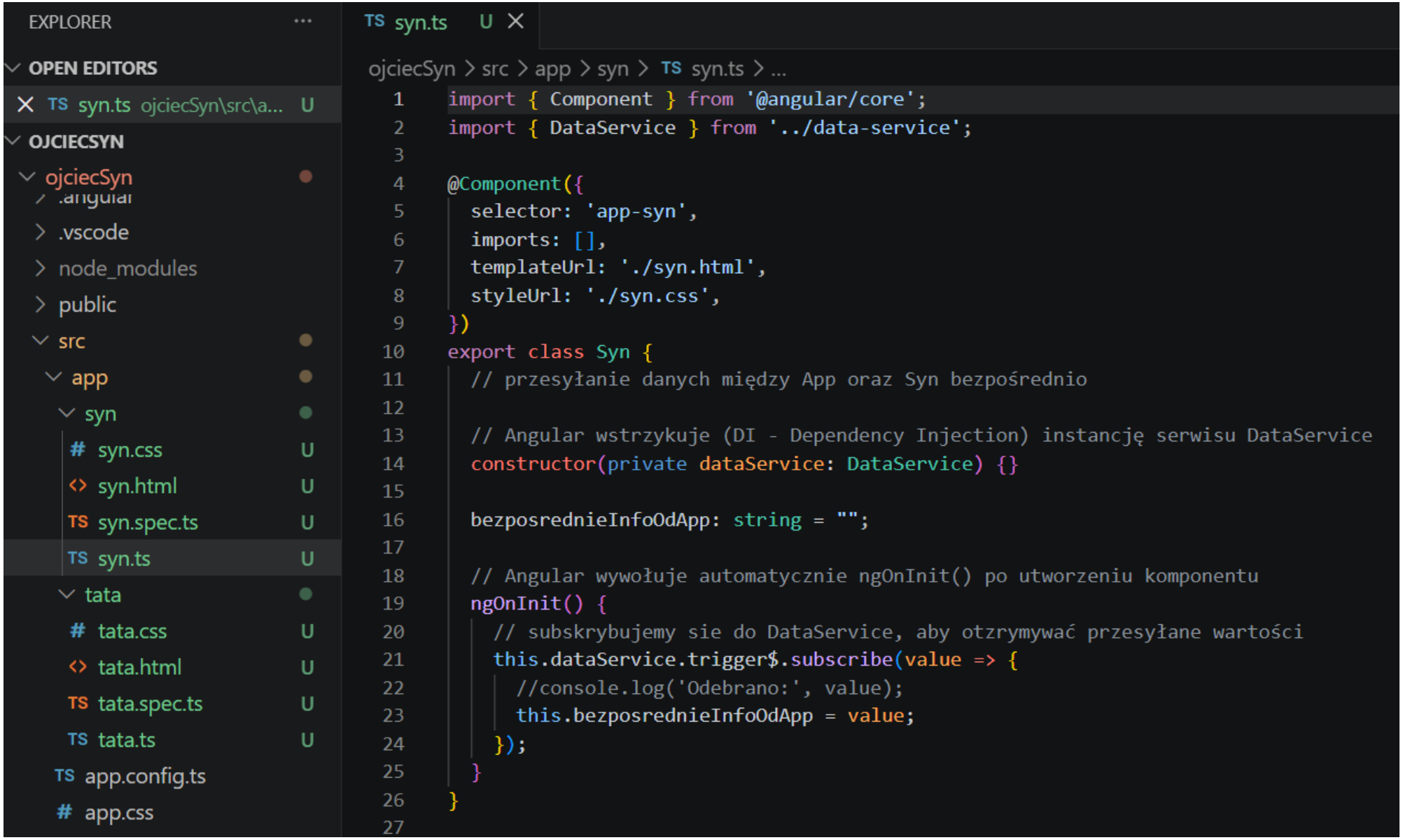
app.html



Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

z komponentu App do Syn

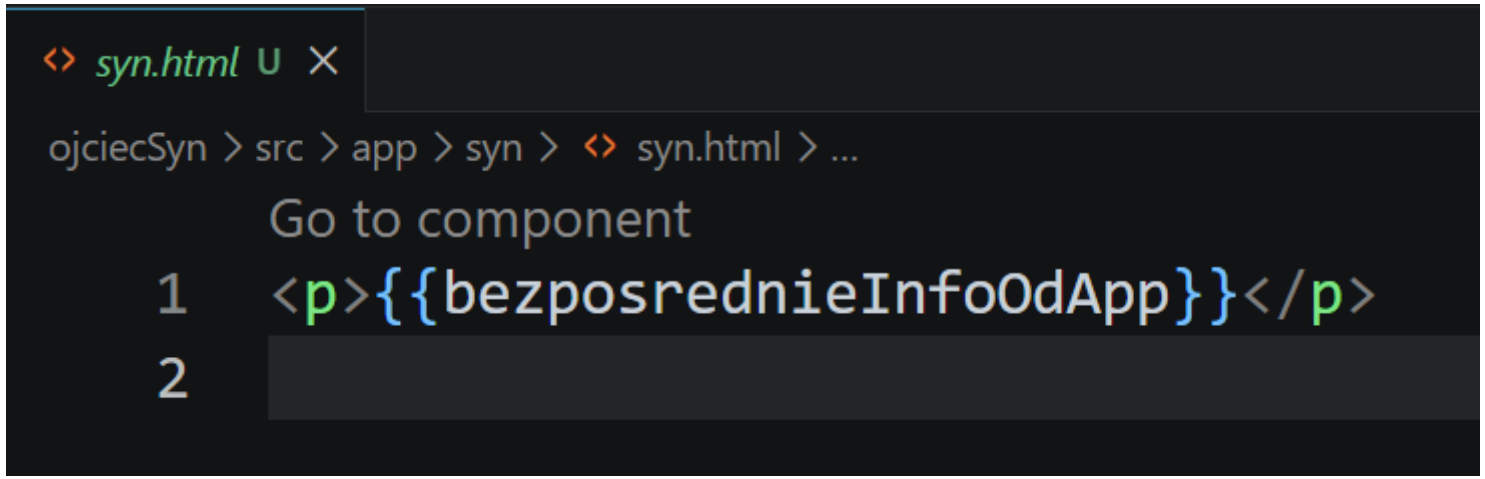
syn.ts



Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

z komponentu App do Syn

syn.html



sklep

http://localhost:4200

sklep works!

Sklep

towary

20 ilość

kurier

pomidory

produkt works!

komponent nadrzędny sklep

pomiędzy komponentami przesyłane są dane – np. wybrany produkt (np. jabłka)

wybór towaru

komponent podrzędny produkt

Wybrany Towar:

zdjęcie	nazwa	cena	ilość	należność	kurier
	pomidory	9 zł	20	180 zł	nie

sklep

http://localhost:4200

sklep works!

Sklep

towary

20 ilość

kurier

Piotrkowska podaj adres

pomidory

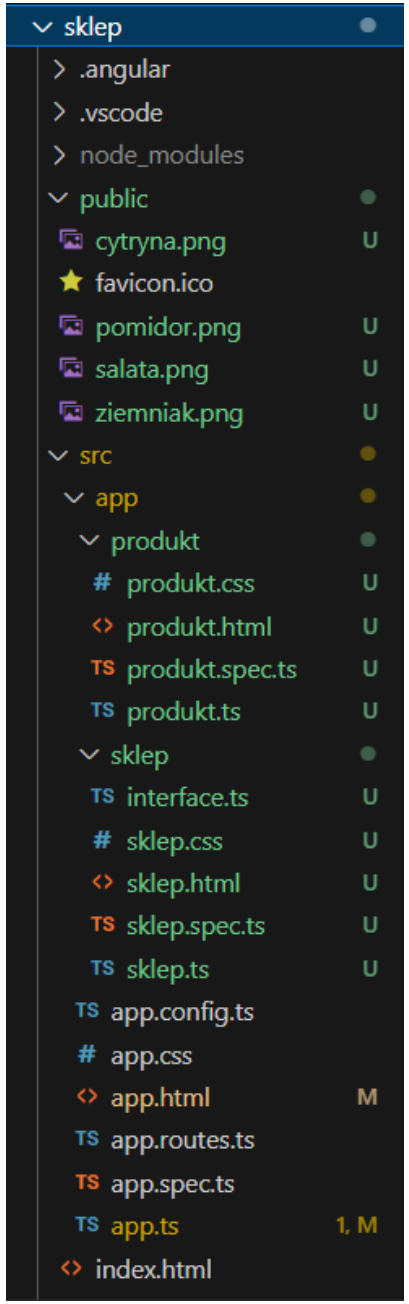
produkt works!

po zaznaczeniu kuriera pojawia się input na wpisanie adresu oraz nowa kolumna z adresem w komponencie produkt

Wybrany Towar:

zdjęcie	nazwa	cena	ilość	należność	kurier	adres
	pomidory	9 zł	20	180 zł	tak	Piotrkowska

sklep



zdjęcia towarów w katalogu public

komponent podrzędny produkt

komponent nadrzędny sklep

Angular ver.20

Sklep
komponent nadrzędny sklep

sklep.ts U

sklep > src > app > sklep > sklep.ts > ...

```ts
import { Component} from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { Produkt } from '../produkt/produkt';
import { Towar } from './interface';

@Component({
  selector: 'app-sklep',
  imports: [FormsModule, CommonModule, Produkt],
  templateUrl: './sklep.html',
  styleUrl: './sklep.css',
})
export class Sklep {
    towary : Towar[] =[
                { nazwa: "cytryny", cena: 20, zdjecie: "cytryna.png"},
                { nazwa: "ziemniaki", cena: 2.2, zdjecie: "ziemniak.png"},
                { nazwa: "salata", cena: 5.2, zdjecie: "salata.png"},
                { nazwa: "pomidory", cena: 9, zdjecie: "pomidor.png"}
                ];

  towar: Towar = this.towary[0];
  ilosc: number = 0;
  kurier:boolean = false;
  adres: string = "";
}
```

interface.ts U

sklep > src > app > sklep > interface.ts > ...

```ts
export interface Towar{
  nazwa: string;
  cena: number;
  zdjecie:string;
}

```

Angular ver.20

Sklep
komponent nadrzędny sklep

sklep.component.html U

komponentySklep > src > app > sklep > sklep.component.html > ...

```
1   <h1>Sklep</h1>
2   <p>towary</p>
3   <!-- select -->
4   <!-- wiązanie dwukierunkowe pola towar - dzięki temu -->
5   <!-- jeśli klient wybierze nową opcję będzie to odwzorowane w polu towar-->
6   <select [(ngModel)] = "towar">
7       <!-- wypisujemy opcje korzystając z dyrektywy *ngFor, iterujemy po tablicy towary -->
8       <!-- ngValue pobiera iterowany obiekt i przekazuje do pola towar -->
9       <option *ngFor = "let item of towary" [ngValue] = "item">{{item.nazwa}}</option>
10  </select>
11
12  <!-- w komponencie sklep zagnieżdżamy komponent podrzędny towar-->
13  <!-- pole towar komponentu sklep jest związane z polem wybranyTowar komponentu towar-->
14  <!-- dzięki temu gdy klient wybierze towar z listy zostanie on przekazany do pola wybranyTowar-->
15  <app-towar [wybranyTowar] = "towar"></app-towar>
```

Angular ver.14

Sklep
komponent nadrzędny sklep

sklep.html U

sklep > src > app > sklep > sklep.html > ...

Go to component

```
<p>sklep works!</p>
<h1>Sklep</h1>
<p>towary</p>
<input type="number" [(ngModel)] = "ilosc">ilość<br>
<label><input type="checkbox" [(ngModel)] = kurier >kurier</label><br>
<label  *ngIf = "kurier"><input type="text" [(ngModel)] = "adres">podaj adres</label><br>

<select [(ngModel)] = "towar">
  <option *ngFor = "let item of towary" [ngValue]= "item" >{{item.nazwa}}</option>

</select>

<app-produkt [wybranyTowar] = "towar" [wybranaIlosc] = "ilosc" [czyKurier] = "kurier" [podanyAdres]="adres"></app-produkt>

```

Angular ver.20

Sklep
komponent podrzędny produkt

TS produkt.ts U

sklep > src > app > produkt > TS produkt.ts > ...

```ts
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';
import { Towar } from '../sklep/interface';

@Component({
  selector: 'app-produkt',
  imports: [CommonModule],
  templateUrl: './produkt.html',
  styleUrl: './produkt.css',
})
export class Produkt {

  // wykrzyknik powoduje, że kompilator nie bedzie ostrzgał
  // o mozliwej wartości null lub undefined
  @Input() wybranyTowar!: Towar;

  @Input() wybranaIlosc!: number;

  @Input() czyKurier!: boolean;

  @Input() podanyAdres!: string;
}
```

Sklep
komponent podrzędny produkt

produkt.html U

sklep > src > app > produkt > produkt.html > ...

Go to component

```
<p>produkt works!</p>
<br><br>
<p>Wybrany Towar:</p>
<table>
  <tr>
    <th>zdjęcie</th>
    <th>nazwa</th>
    <th>cena</th>
    <th>ilość</th>
    <th>należność</th>
    <th>kurier</th>
    <th *ngIf = "czyKurier">adres</th>
  </tr>
  <tr>
    <td><img src={{wybranyTowar.zdjecie}} width="200px"></td>
    <td>{{wybranyTowar.nazwa}}</td>
    <td>{{wybranyTowar.cena}} zł</td>
    <td>{{wybranaIlosc}}</td>
    <td>{{ wybranyTowar.cena*wybranaIlosc | number}} zł</td>
    <td *ngIf = "czyKurier else nie">tak</td>
    <ng-template #nie>
      <td >nie</td>
    </ng-template>
    <td *ngIf = "czyKurier">{{podanyAdres}}</td>
  </tr>
</table>
```

Angular ver.20

Sklep
komponent podrzędny produkt

produkt.css U ×

sklep > src > app > produkt > # produkt.css > ...

```css
table {
  font-family: arial, sans-serif;
  border-collapse: collapse;
  width: 50%;
}

td, th {
  border: 1px solid #dddddd;
  text-align: left;
  padding: 8px;
}

tr:nth-child(even) {
  background-color: #dddddd;
}

```

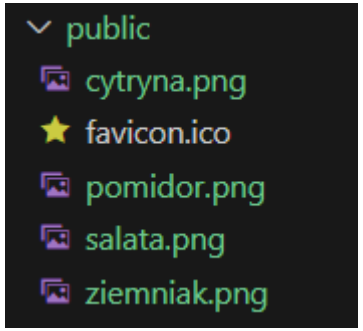
Sklep
komponent główny app

app.html M

sklep > src > app > app.html > ...

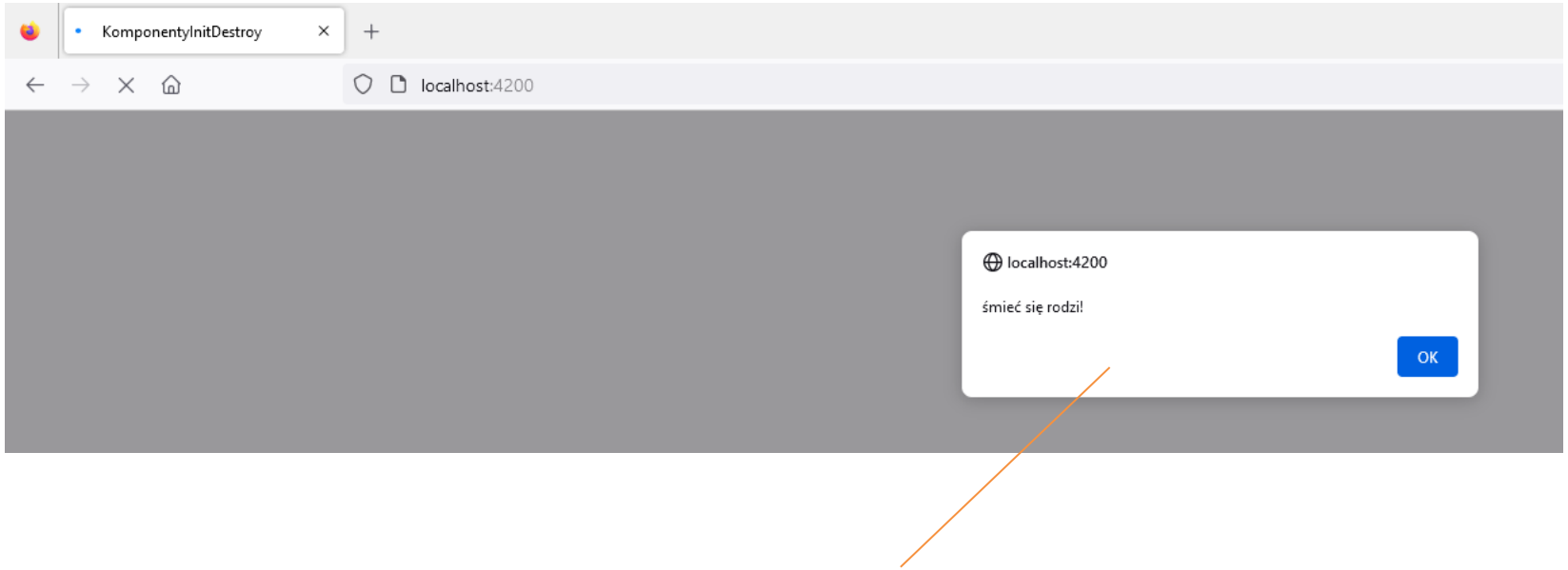
Go to component | You, 1 second ago | 1 author (You)

```
1  <!-- w komponencie głównym zagnieżdżamy komponent sklep-->
2  <app-sklep></app-sklep>
```



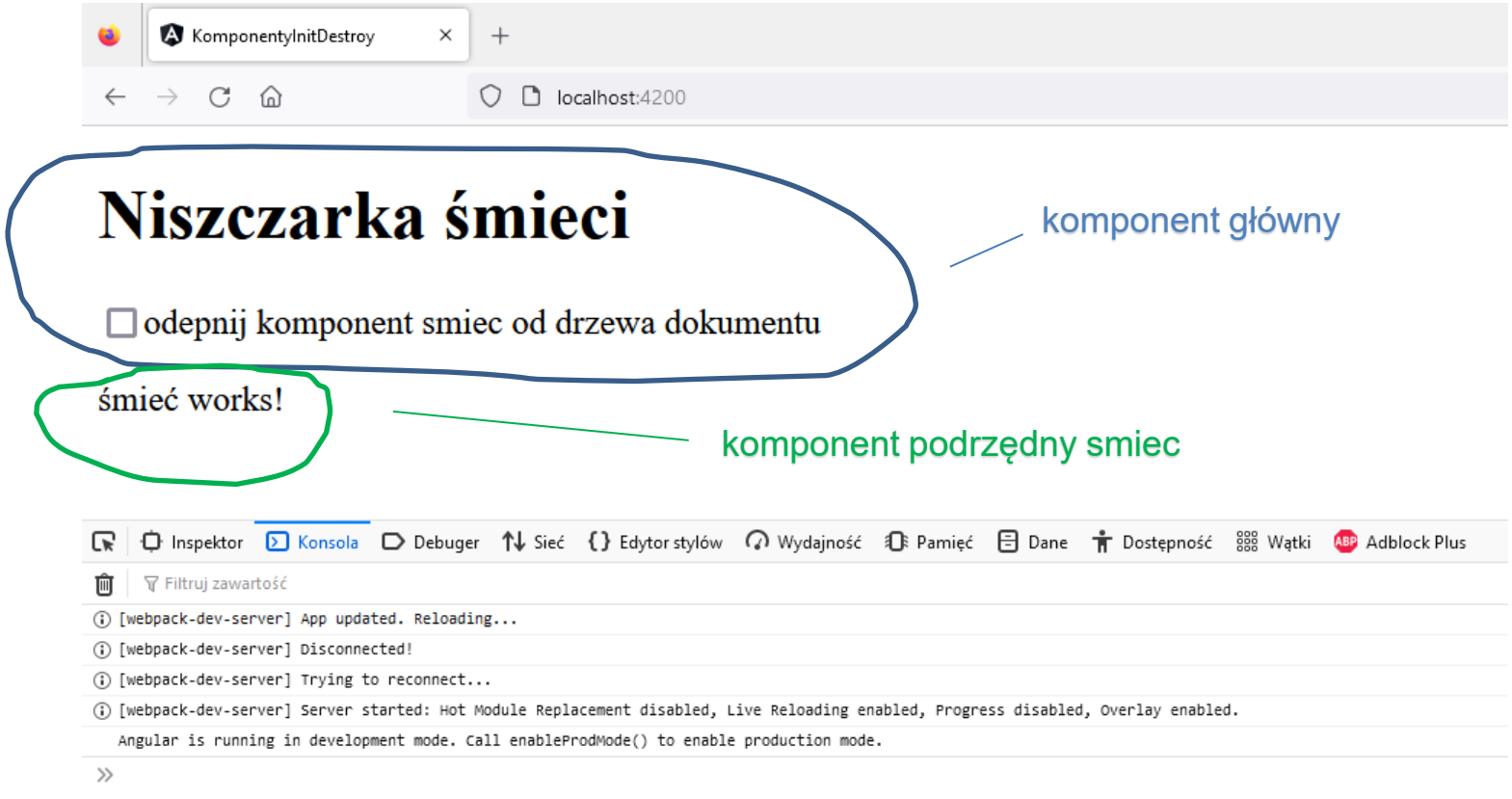
Sklep
komponent główny

Niszczarka śmieci
ngOnInit(), ngOnDestroy()

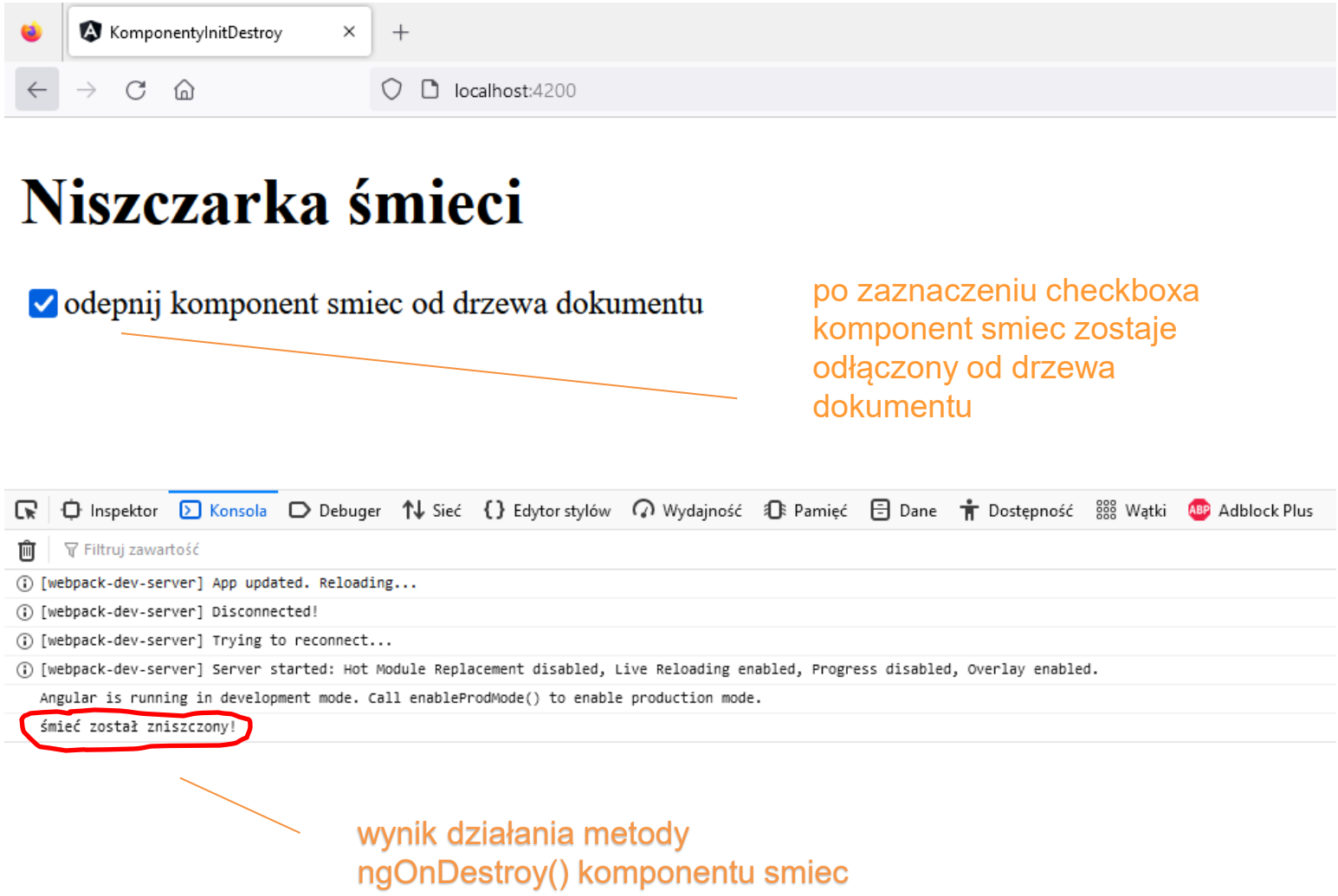


wynik działania metody ngOnInit()
komponentu smiec

Niszczarka śmieci
ngOnInit(), ngOnDestroy()



Niszczarka śmieci
ngOnInit(), ngOnDestroy()



po zaznaczeniu checkboxa komponent smiec zostaje odłączony od drzewa dokumentu

wynik działania metody ngOnDestroy() komponentu smiec

Niszczarka śmieci
komponent smiec

smiec.component.html U

komponentyInitDestroy > src > app > sr

```
1   <p>śmieć works!</p>
```

smiec.component.ts U

komponentyInitDestroy > src > app > smiec > TS smiec.component.ts >

```
1   import { Component, OnInit } from '@angular/core';
2
3   @Component({
4     selector: 'app-smiec',
5     templateUrl: './smiec.component.html',
6     styleUrls: ['./smiec.component.css']
7   })
8   export class SmiecComponent implements OnInit {
9
10    constructor() { }
11
12    ngOnInit(): void {
13      alert("śmieć się rodzi!")
14    }
15
16    ngOnDestroy(): void {
17      console.log("śmieć został zniszczony!")
18    }
19  }
```

definicje metod ngOnInit() oraz ngOnDestroy()

Angular ver.14

Niszczarka śmieci
komponent główny

app.component.html M

komponentyInitDestroy > src > app > app.component.html > ...

```
1   <h1>Niszczarka śmieci</h1>
2   <label><input type="checkbox" [(ngModel)] = zniszcz>odepnij komponent smiec od drzewa dokumentu</label>
3   <app-smiec *ngIf = !zniszcz></app-smiec>
```

usuwamy komponent smiec za pomocą
dyrektywy *ngIf, która ma wartość true lub false
w zależności od ustawienia checkboxa

import FormsModule

app.component.ts M

komponentyInitDestroy > src > app > app.component.ts > ...

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css']
7   })
8   export class AppComponent {
9     zniszcz: boolean = false;
10  }
```

pole zniszcz przyjmuje
wartości true lub false

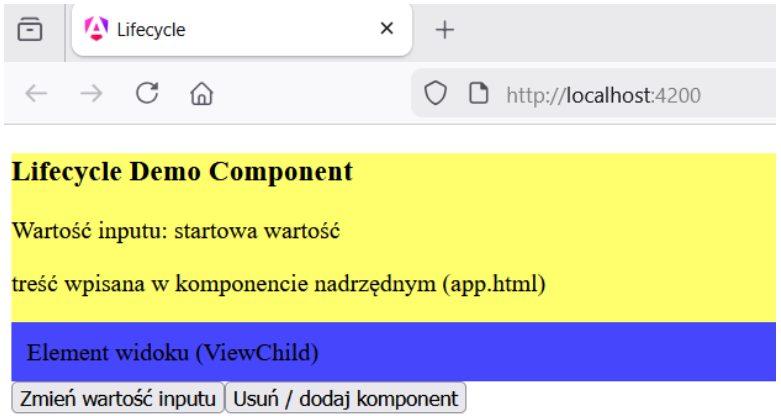
app.module.ts M

komponentyInitDestroy > src > app > app.module.ts > ...

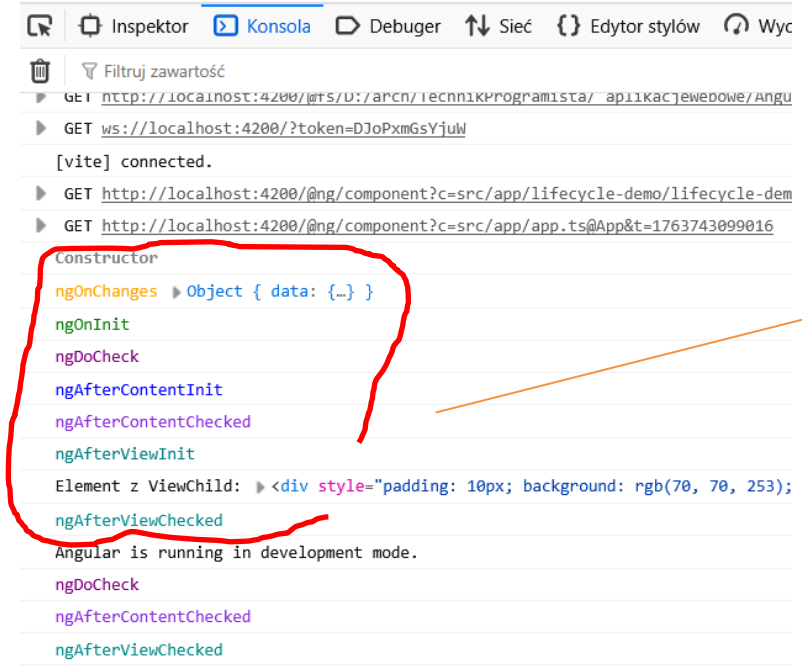
```
1   import { NgModule } from '@angular/core';
2   import { BrowserModule } from '@angular/platform-browser';
3   import { FormsModule } from '@angular/forms'
4
5   import { AppComponent } from './app.component';
6   import { SmiecComponent } from './smiec/smiec.component';
7
8   @NgModule({
9     declarations: [
10      AppComponent,
11      SmiecComponent
12    ],
13    imports: [
14      BrowserModule,
15      FormsModule
16    ],
17    providers: [],
18    bootstrap: [AppComponent]
19  })
20  export class AppModule { }
```

Angular ver.14

Cykl życia komponentu



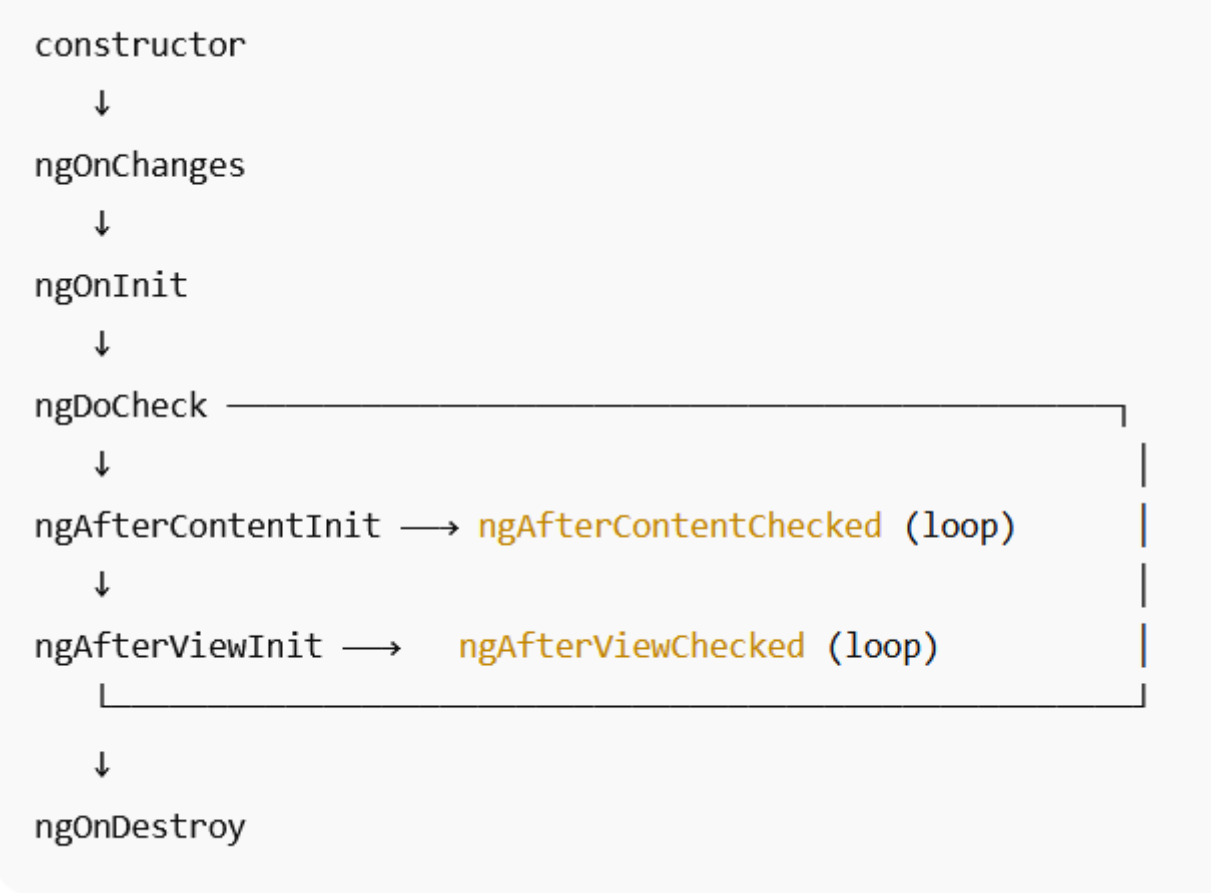
komponent lifecycle-demo



cykl życia komponentu lifecycle-demo

Cykl życia komponentu

kolejność wywołań *Lifecycle Hooks*:



Cykl życia komponentu

1. constructor()

Kiedy wywoływany: w momencie tworzenia instancji klasy komponentu (zanim Angular ustawi @Input).

Do czego służy:
•inicjalizacja pól klasy,
•wstrzykiwanie zależności (DI).

Nie ma jeszcze dostępu do danych wejściowych @Input ani do widoku.

2. ngOnChanges(changes: SimpleChanges)

Kiedy wywoływany:
•**za każdym razem**, gdy @Input zmieni wartość,
•również **przed ngOnInit** przy pierwszym ustawieniu @Input.

Do czego służy:
•reagowanie na zmiany danych przekazywanych z komponentu rodzica,
•porównywanie poprzedniej i nowej wartości.

Cykl życia komponentu

3. ngOnInit()

Kiedy wywoływany: raz, zaraz po pierwszym ngOnChanges.

Do czego służy:
•inicjalizacja komponentu,
•pobieranie danych (np. z API),
•konfiguracje, które wymagają już ustawionych @Input.

4. ngDoCheck()

Kiedy wywoływany:
•za każdym cyklem detekcji zmian Angulara (bardzo często).

Do czego służy:
•tworzenie własnych, niestandardowych mechanizmów wykrywania zmian,
•reagowanie na zmiany, których Angular sam nie wykryje (np. modyfikacja obiektu bez zmiany referencji).

Wywoływany **bardzo często** → nie wkładać tam ciężkich operacji.

Cykl życia komponentu

5. ngAfterContentInit()

Kiedy wywoływany: raz, po osadzeniu zawartości ng-content.

Do czego służy:
•reagowanie na inicjalizację *content projection* (zawartość dostarczona przez rodzica).

6. ngAfterContentChecked()

Kiedy wywoływany: przy **każdym** sprawdzeniu zawartości projected content.

Do czego służy:
•reagowanie na aktualizacje treści przekazanej do ng-content.

Cykl życia komponentu

7. ngAfterViewInit()

Kiedy wywoływany: raz, po inicjalizacji widoku komponentu i jego elementów potomnych.

Do czego służy:
•dostęp do elementów z @ViewChild,
•uruchamianie bibliotek manipulujących DOM (np. chart.js, mapy, animacje).

To tutaj po raz pierwszy this.box.nativeElement jest dostępny.

8. ngAfterViewChecked()

Kiedy wywoływany: przy **każdym** sprawdzaniu widoku.

Do czego służy:
•reagowanie na aktualizacje DOM we własnym widoku.

Również wywoływany bardzo często.

Cykl życia komponentu

9. ngOnDestroy()

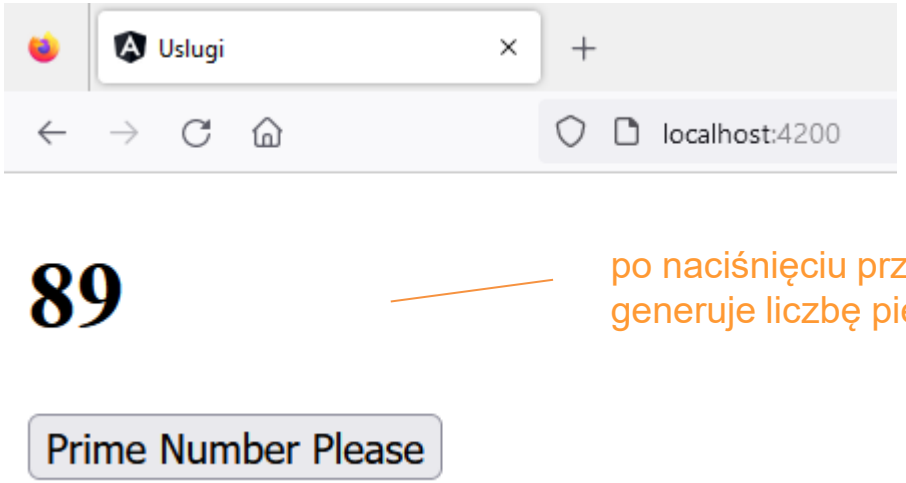
Kiedy wywoływany: tuż przed usunięciem komponentu z DOM.

Do czego służy:

- sprzątanie po komponencie:
  - odpinanie subskrypcji (unsubscribe()),
  - czyszczenie timerów,
  - usuwanie event listenerów,
  - anulowanie połączeń WebSocket.

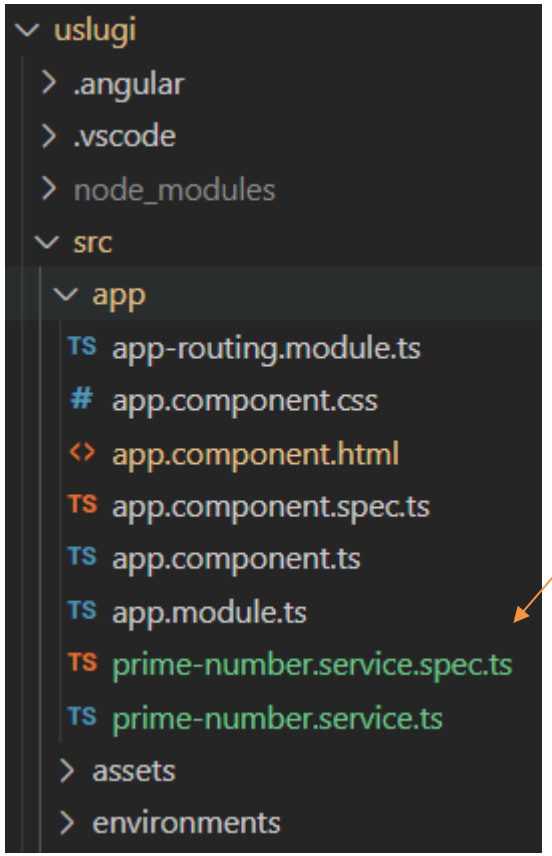
Usługi (Services)

Usługa jest zazwyczaj klasą o wąskim, dobrze zdefiniowanym celu (robi coś konkretnego)



po naciśnięciu przycisku usługa (service) generuje liczbę pierwszą

Usługi



generowanie usługi PrimeNumberService

generate service nazwa usługi

```
ng g s primeNumber
```

```
CREATE src/app/prime-number.service.spec.ts (383 bytes)
CREATE src/app/prime-number.service.ts (140 bytes)
```

Usługi
PrimeNumberService

TS prime-number.service.ts U

uslugi > src > app > TS prime-number.service.ts > ...

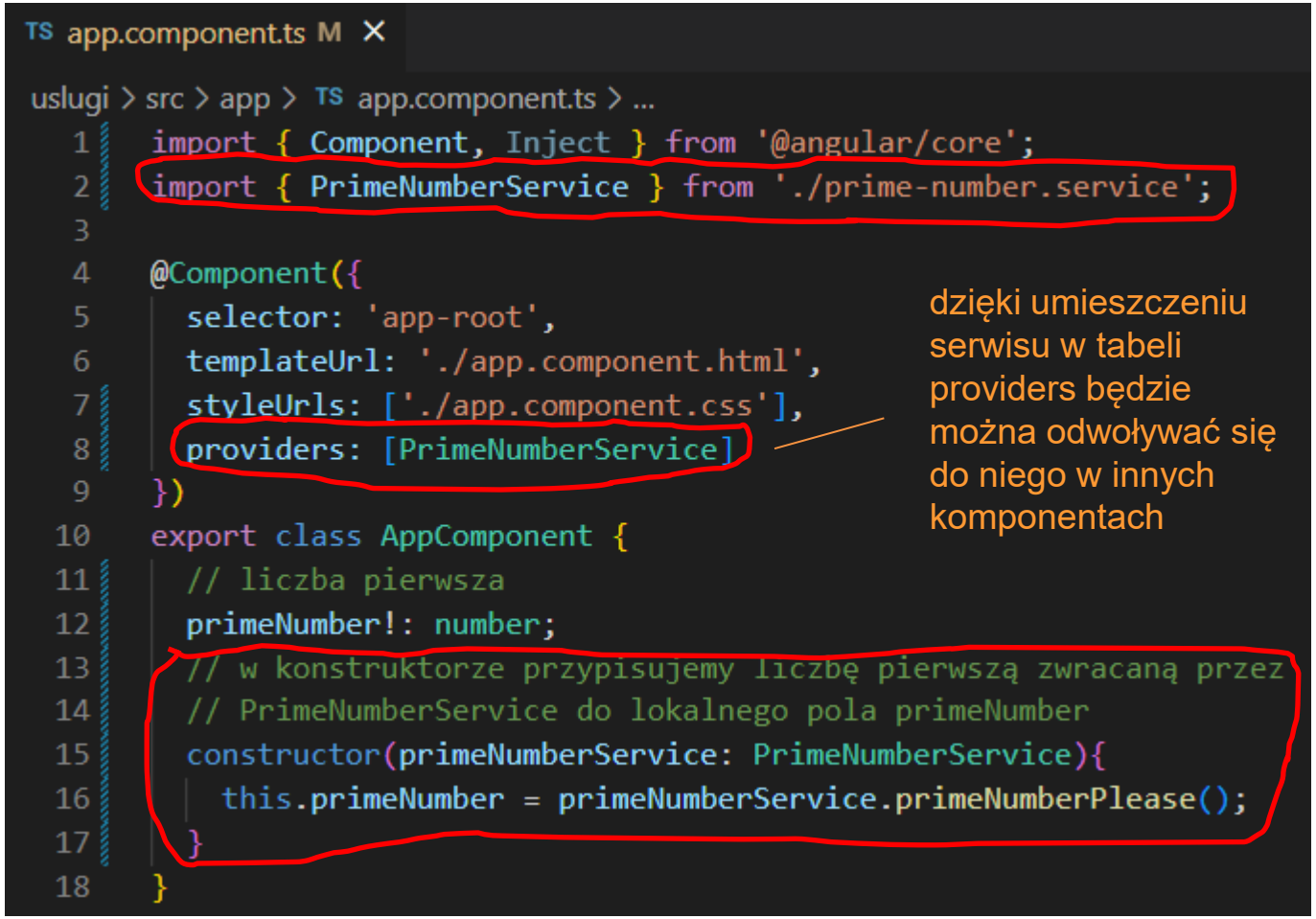
```ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class PrimeNumberService {

  constructor() { }

  // generator liczb pierwszych :)
  primeNumberPlease(): number{
    // tablica liczb pierwszych
    let primeNumbers: number[] = [2,3,5,7,11,13,17,19,23,29,
                                  31,37,41,43,47,53,59,61,67,
                                  71, 73,79,83,89,97,101];
    // losujemy indeks
    let index = Math.floor(Math.random() * 25);
    // zwracamy liczbe pierwszą z tablicy
    return primeNumbers[index];
  }
}
```

metoda usługi PrimeNumberService o nazwie primeNumberPlease generuje liczby pierwsze

Angular ver.14

Usługi
PrimeNumberService

konfiguracja usługi w pliku app.components.ts – dzięki temu będzie można odwoływać się do niej w komponencie głównym



dzięki umieszczeniu serwisu w tabeli providers będzie można odwoływać się do niego w innych komponentach

Angular ver.14

Usługi
PrimeNumberService

konfiguracja widoku

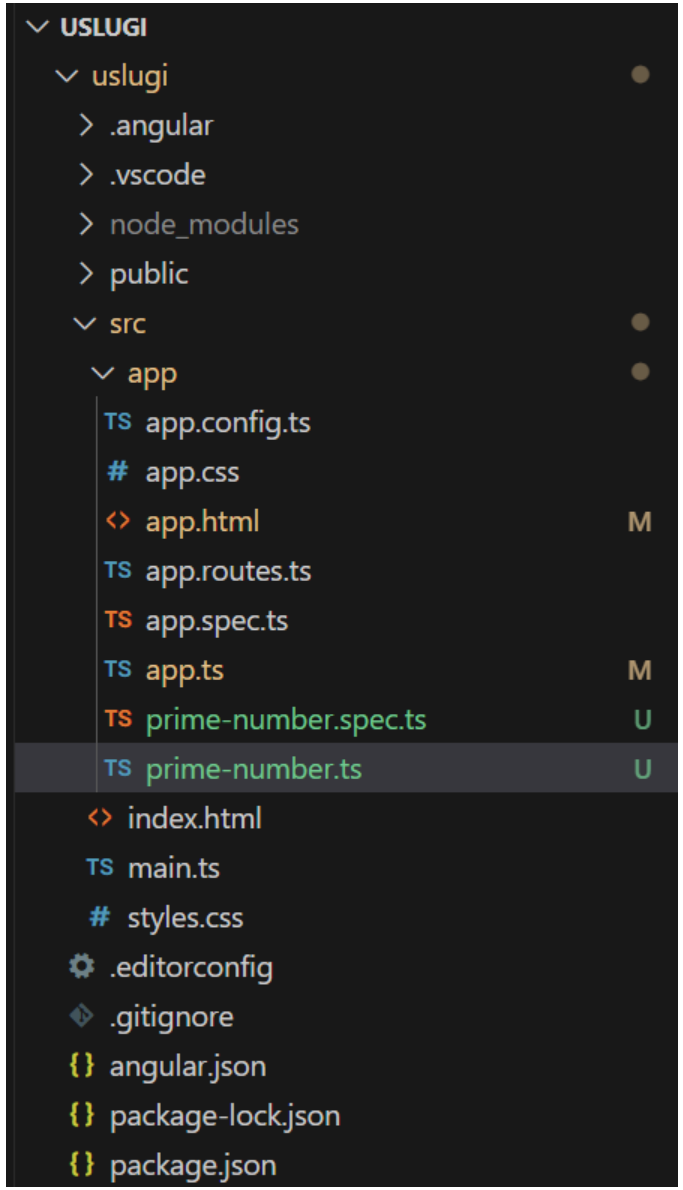
app.component.html M

uslugi > src > app > app.component.html > ...

```
1
2   <!-- wyświetlamy liczbe pierwszą wygenerowaną przez PrimeNumberService-->
3   <h1>{{primeNumber}}</h1>
4   <!-- przeładowujemy stronę -->
5   <button onclick="location.reload();">Prime Number Please</button>
```

Angular ver.14

Usługi



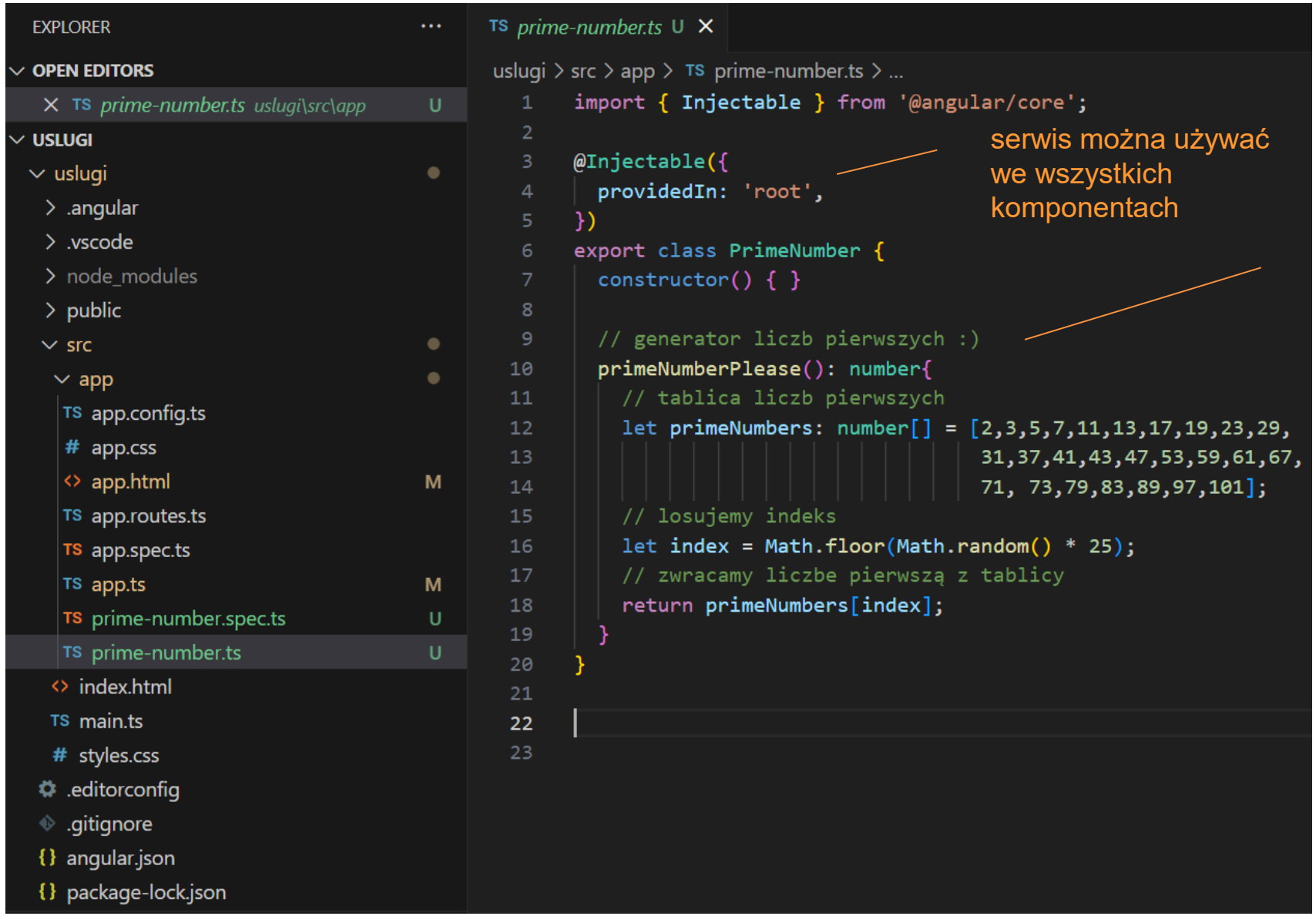
generowanie usługi (klasy) PrimeNumber

generate service nazwa usługi

```
ng g s primeNumber
```

Angular ver.20

Usługi
PrimeNumber

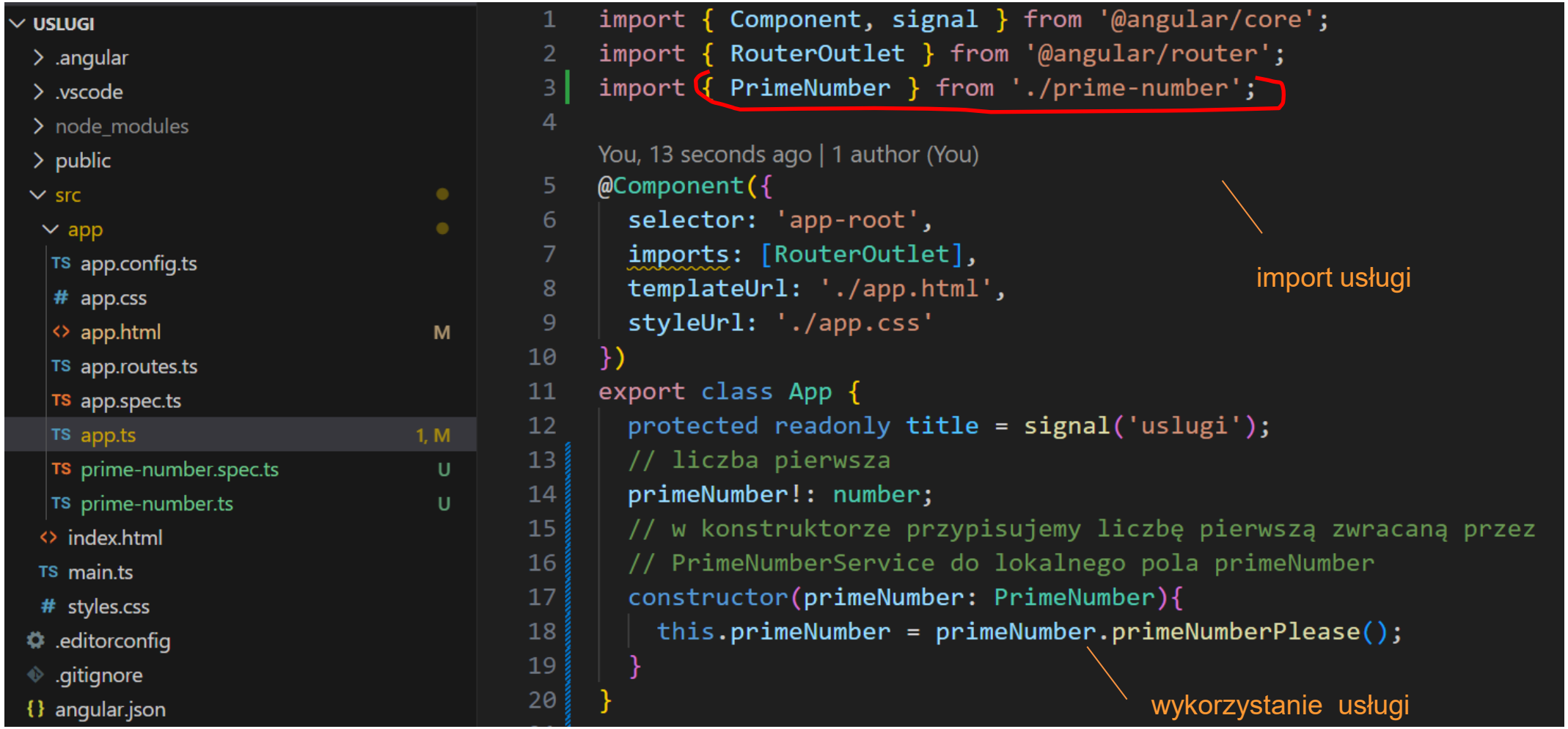


serwis można używać we wszystkich komponentach

metoda usługi PrimeNumber o nazwie primeNumberPlease generuje liczby pierwsze

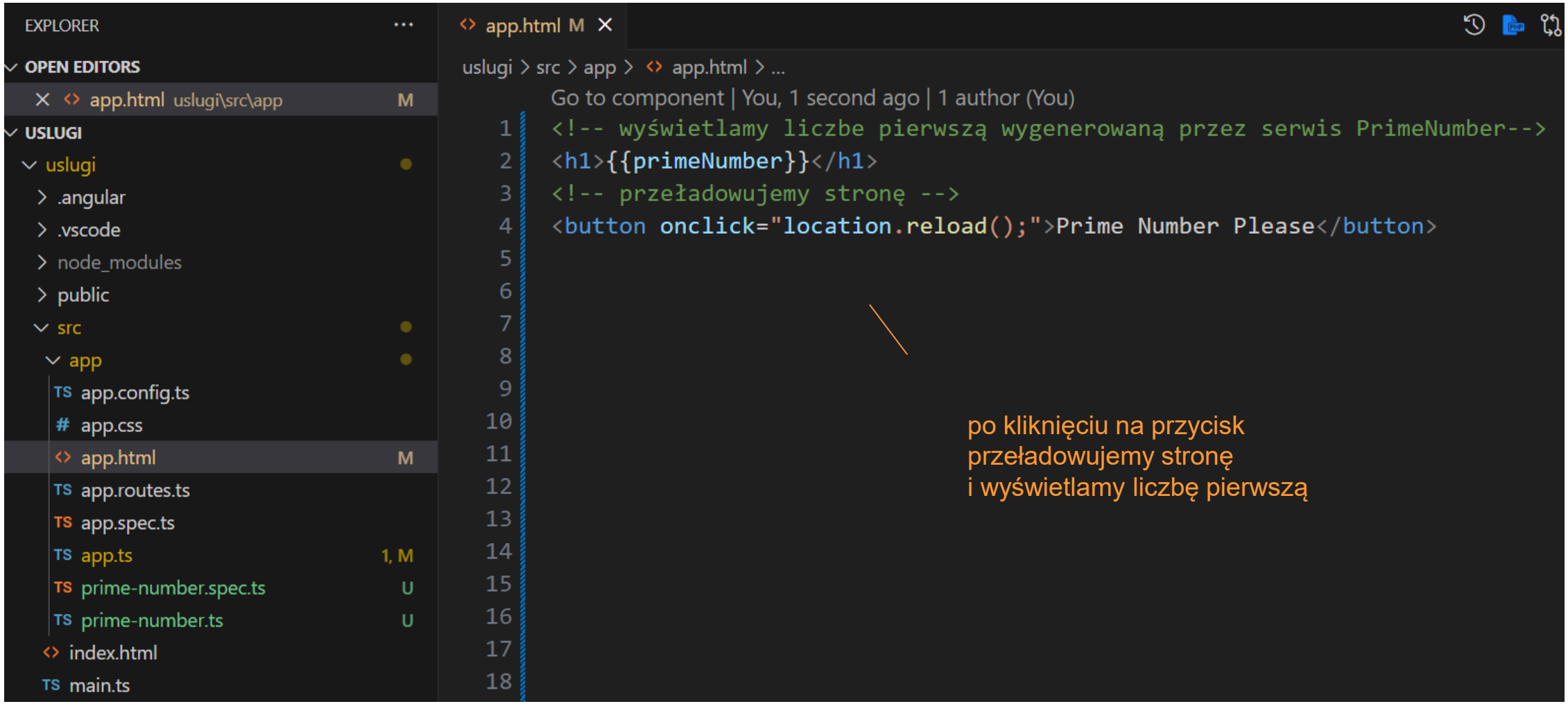
Angular ver.20

Usługi
PrimeNumber



Angular ver.20

Usługi
PrimeNumber



Angular ver.20

Formularz

Ankieta

Jan imię
Kowalski nazwisko
jan@kowalski.pl e-mail

płeć

mężczyzna ○ kobieta ○

zainteresowania

sport ☑ film ☑ elektronika ☑ informatyka ☑

szkoła

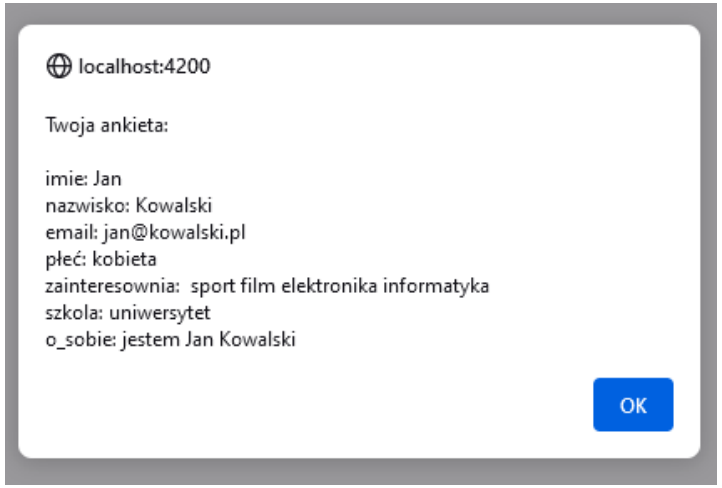
Uniwersytet Łódzki

krótko o sobie

jestem Jan Kowalski

submit reset

Ankieta obsługiwana przez Angular



po naciśnięciu przycisku wyświetlają się zaznaczone opcje Ankiety w oknie alert

Formularz
Ankieta

app.component.html M

formularz > src > app > app.component.html > ...

```
1  <h1>Ankieta</h1>
2  <form name="ankieta" #formularz = "ngForm">
3    <input type="text" name="imie" [(ngModel)] = "imie">imię<br>
4    <input type="text" name="nazwisko" [(ngModel)] = "nazwisko">nazwisko<br>
5    <input type="text" name="email" [(ngModel)] = "email">e-mail<br>
6
7    <h3>płeć</h3>
8    mężczyzna<input type="radio" name="plec" value="mezczyzna" [(ngModel)] = "plec">
9    kobieta<input type="radio" name="plec" value="kobieta">
10
11   <h3>zainteresowania</h3>
12
13   sport<input type="checkbox" name="sport" value="1" [(ngModel)] = "sport">
14   film<input type="checkbox" name="film" value="1" [(ngModel)] = "film">
15   elektronika<input type="checkbox" name="elektronika" value="1" [(ngModel)] = "elektronika">
16   informatyka<input type="checkbox" name="informatyka" value="1" [(ngModel)] = "informatyka">
17
18   <h3>szkoła</h3>
19   <select name="szkola" [(ngModel)] = "szkola">
20   <option value="podstawowa">Podstawowa</option>
21   <option value="liceum">Liceum Ogólnokształcące</option>
22   <option value="technikum">Technikum</option>
23   <option value="politechnika">Politechnika Łódzka</option>
24   <option value="uniwersytet">Uniwersytet Łódzki</option>
25   <option value="inna">inna</option>
26   </select>
27
28   <h3>krótko o sobie</h3>
29   <textarea name="o_sobie" rows=5 cols=40 [(ngModel)] = "o_sobie"></textarea>
30
31   <hr>
32   <input type="submit" value="submit" (click)="onSubmit()">
33   <input  type="reset" value="reset">
34 </form>
```

każdy element formularza jest związany dwukierunkowo z polem o nazwie odzwierciedlającej funkcję elementu

po kliknięciu na przycisk wywoływana jest metoda onSubmit()

Angular ver.14

Formularz
Ankieta

app.component.ts M

formularz > src > app > app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  imie: string = "";
  nazwisko: string = "";
  email: string = "";
  plec: string = "";
  sport: string = "";
  film: string = "";
  elektronika: string = "";
  informatyka: string = "";
  szkola: string = "";
  o_sobie: string = "";
```

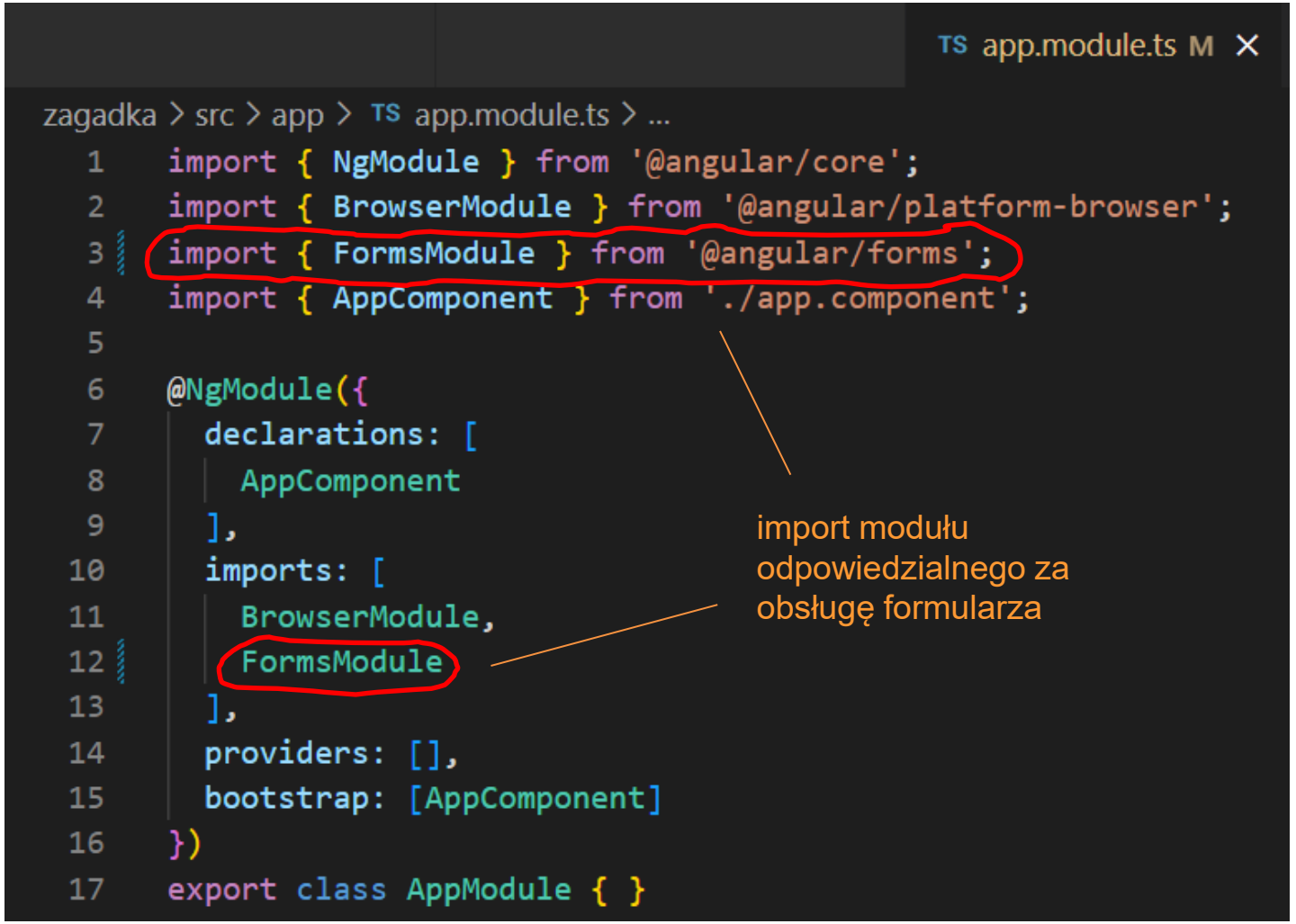
pola klasy AppComponent odpowiadają elementom ankiety

metoda onSubmit() klasy AppComponent

```ts
onSubmit(){
  // zbieramy dane z ankiety
  if(this.plec!=""){
    if(this.plec=="mezczyzna"){
      this.plec="mężczyzna";
    }else{
      this.plec="kobieta";
    }
  }
  let zainteresowania = "";
  if (this.sport == "1"){
    zainteresowania += " sport";
  }
  if (this.film == "1"){
    zainteresowania += " film";
  }
  if (this.elektronika == "1"){
    zainteresowania += " elektronika";
  }
  if (this.informatyka == "1"){
    zainteresowania += " informatyka";
  }
  alert("Twoja ankieta:\n" +
        "\nimie: " + this.imie +
        "\nnazwisko: " + this.nazwisko +
        "\nemail: " + this.email +
        "\npłeć: " + this.plec +
        "\nzainteresownia: " + zainteresowania +
        "\nszkola: " + this.szkola +
        "\no_sobie: " + this.o_sobie
        );
}
}
```

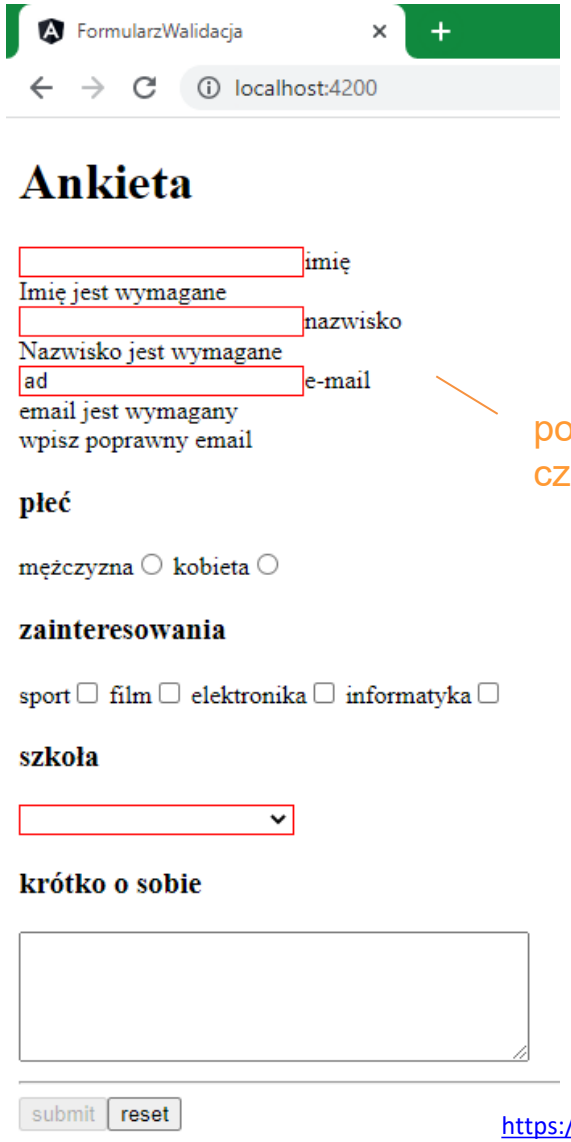
Angular ver.14

Formularz
Ankieta



Angular ver.14

Formularz
Ankieta z walidacją



pola wymagane są w czerwonej ramce

https://angular.io/guide/form-validation

Formularz
Ankieta z walidacją

zmiany związane z walidacją pola imie (pole jest wymagane)

zmienna szablonowa name przypisana do ngModel

app.component.html M

formularzWalidacja > src > app > app.component.html > ...

```
1   <h1>Ankieta</h1>
2   <form id = "ankietaForm" name="ankieta" #formularz = "ngForm">
3     <input type="text" name="imie"  [(ngModel)] = "imie" required #name="ngModel">imię<br>
4     <div *ngIf="name.invalid && name.touched">Imię jest wymagane</div>
```

dyrektywa *ngIf z wykorzystaniem zmiennej szablonowej name

dodatkowa informacja ukazuje się, gdy pole imię nie jest wypełnione i zostało kliknięte albo wybrane klawiszem TAB oraz opuszczone

Angular ver.14

Formularz
Ankieta z walidacją

app.component.html M

formularzWalidacja > src > app > app.component.html > ...

```
<h1>Ankieta</h1>
<form id = "ankietaForm" name="ankieta" #formularz = "ngForm">
  <input type="text" name="imie"  [(ngModel)] = "imie" required #name="ngModel">imię<br>
  <div *ngIf="name.invalid && name.touched">Imię jest wymagane</div>
  <input type="text" name="nazwisko" [(ngModel)] = "nazwisko" required #surname="ngModel">nazwisko<br>
  <div *ngIf="surname.invalid && surname.touched">Nazwisko jest wymagane</div>
  <input type="text" name="email" [(ngModel)] = "email" required pattern=".+@.+" #myEmail="ngModel">e-mail<br>
  <div *ngIf="myEmail.invalid && myEmail.touched">email jest wymagany</div>
  <div *ngIf="myEmail.touched && myEmail.errors?.['pattern']">wpisz poprawny email</div>
  <h3>płeć</h3>
  mężczyzna<input type="radio" name="plec" value="mezczyzna" [(ngModel)] = "plec">
  kobieta<input type="radio" name="plec" value="kobieta">

  <h3>zainteresowania</h3>

  sport<input type="checkbox" name="sport" value="1" [(ngModel)] = "sport">
  film<input type="checkbox" name="film" value="1" [(ngModel)] = "film">
  elektronika<input type="checkbox" name="elektronika" value="1" [(ngModel)] = "elektronika">
  informatyka<input type="checkbox" name="informatyka" value="1" [(ngModel)] = "informatyka">

  <h3>szkoła</h3>
  <select name="szkola" [(ngModel)] = "szkola" required>
  <option value="podstawowa">Podstawowa</option>
  <option value="liceum">Liceum Ogólnokształcące</option>
  <option value="technikum">Technikum</option>
  <option value="politechnika">Politechnika Łódzka</option>
  <option value="uniwersytet">Uniwersytet Łódzki</option>
  <option value="inna">inna</option>
  </select>

  <h3>krótko o sobie</h3>
  <textarea name="o_sobie" rows=5 cols=40 [(ngModel)] = "o_sobie"></textarea>

  <hr>
  <input type="submit" value="submit" [disabled]="!formularz.form.valid" (click)="onSubmit()">
  <input  type="reset" value="reset">
</form>
```

szablon ze zmianami walidacyjnymi

Angular ver.14

Formularz
Ankieta z walidacją

app.component.ts

formularzWalidacja > src > app > app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  imie: string = "";
  nazwisko: string = "";
  email: string = "";
  plec: string = "";
  sport: string = "";
  film: string = "";
  elektronika: string = "";
  informatyka: string = "";
  szkola: string = "";
  o_sobie: string = "";

  onSubmit(){
    // zbieramy dane z ankiety
    if(this.plec!=""){
      if(this.plec=="mezczyzna"){
        this.plec="mężczyzna";
      }else{
        this.plec="kobieta";
      }
    }
    let zainteresowania = "";
    if (this.sport == "1"){
      zainteresowania += " sport";
    }
    if (this.film == "1"){
      zainteresowania += " film";
    }
    if (this.elektronika == "1"){
      zainteresowania += " elektronika";
    }
    if (this.informatyka == "1"){
      zainteresowania += " informatyka";
    }
    alert("Twoja ankieta:\n" +
          "\nimie: " + this.imie +
          "\nnazwisko: " + this.nazwisko +
          "\nemail: " + this.email +
          "\npłeć: " + this.plec +
          "\nzainteresownia: " + zainteresowania +
          "\nszkola: " + this.szkola +
          "\no_sobie: " + this.o_sobie
          );
  }
}
```

app.module.ts M

formularzWalidacja > src > app > app.module.ts > ...

```ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

pliki:
app.component.ts
oraz
app.module.ts
są takie same jak w formularzu bez walidacji

app.component.css M

formularzWalidacja > src > app > # app.compo

```css
.ng-invalid{
    border: 1px solid red;
    width:30%;
}
#ankietaForm{
    border:0px;
}
```

dzięki stylom
wymagane pola mają
czerwoną obramówkę

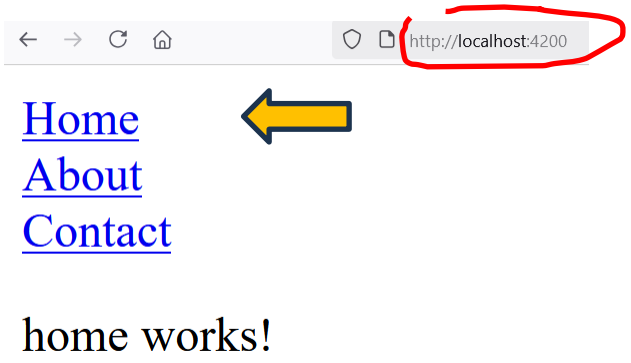
Angular ver.14

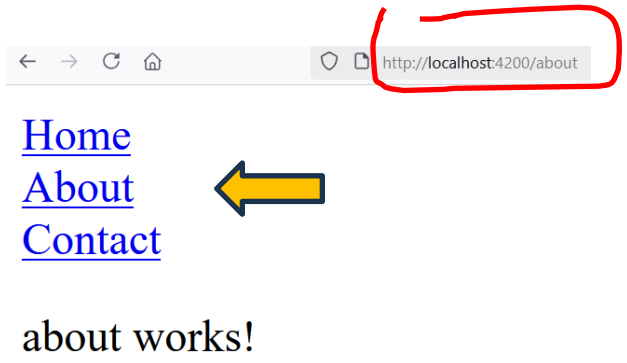
Formularz
Ankieta z walidacją

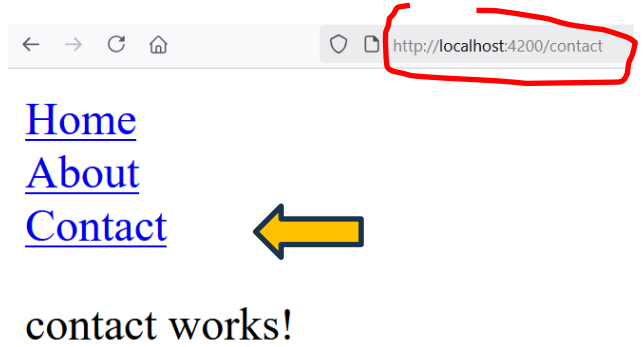
stan weryfikacji
danych opisują klasy
w tabeli

klasa	Opis
valid	dobrze wypełniony
invalid	źle wypełniony
submitted	zatwierdzony
touched	pole zostało wybrane kliknięciem lub tabem
untouched	pole nie wybrane
dirty	użytkownik coś już wpisał (mogło to zostać wykasowane)
pristine	użytkownik nic nie wpisał

routing - podstrony



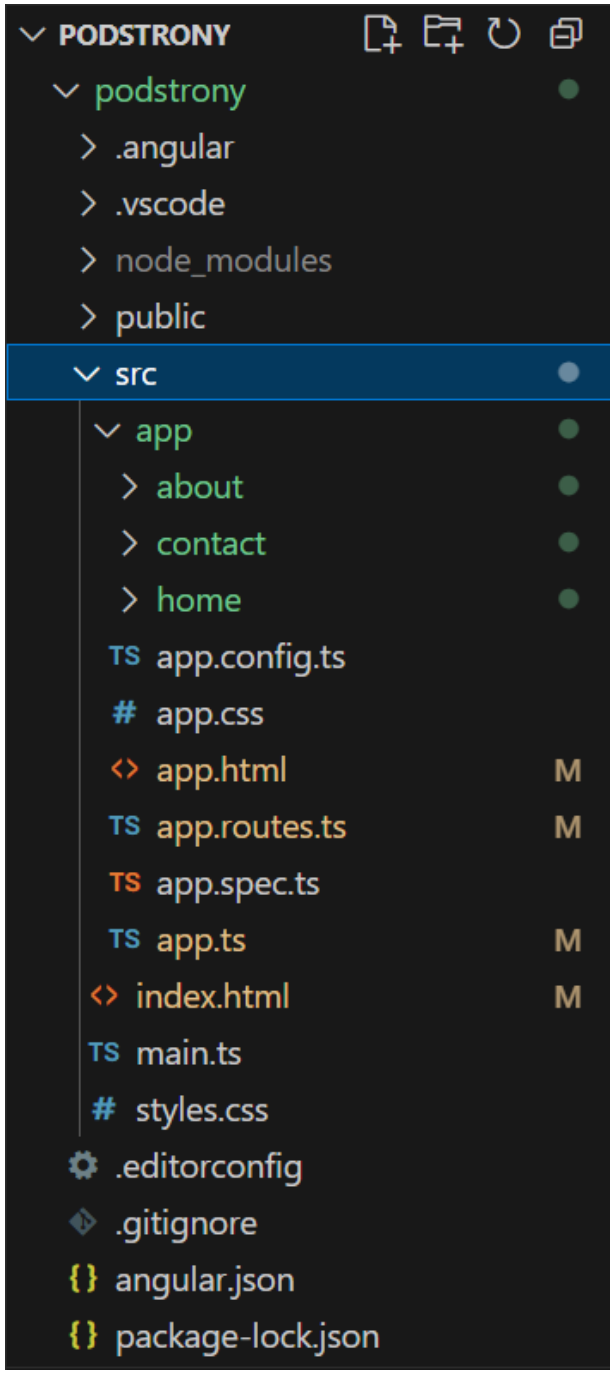




podstrony są komponentami

Angular ver.20

routing - podstrony

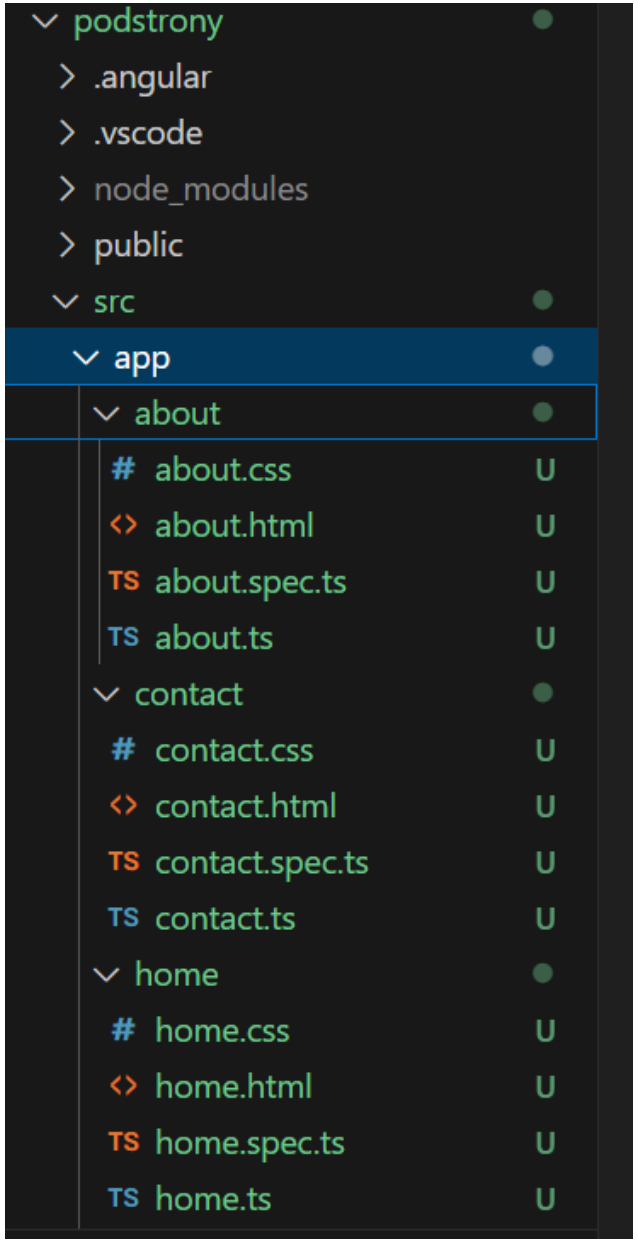


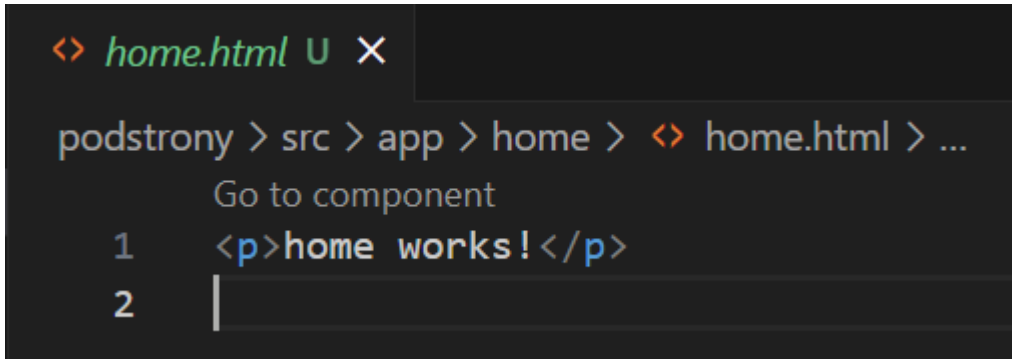
generujemy nowe komponenty,
(nasze podstrony, w terminalu):

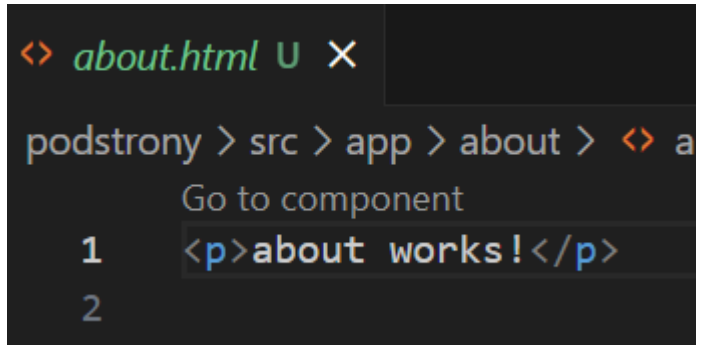
```
ng g c home
ng g c about
ng g c contact
```

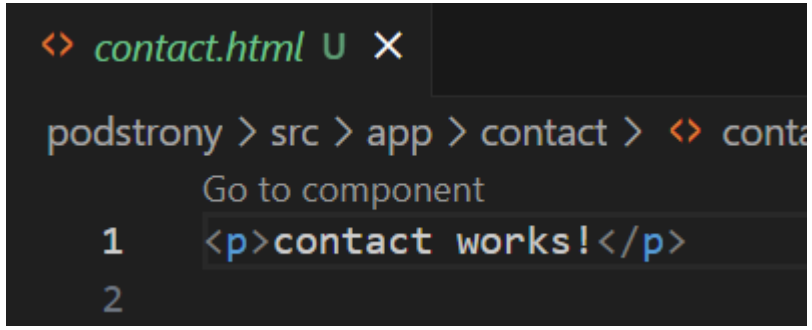
Angular generate component

routing - podstrony



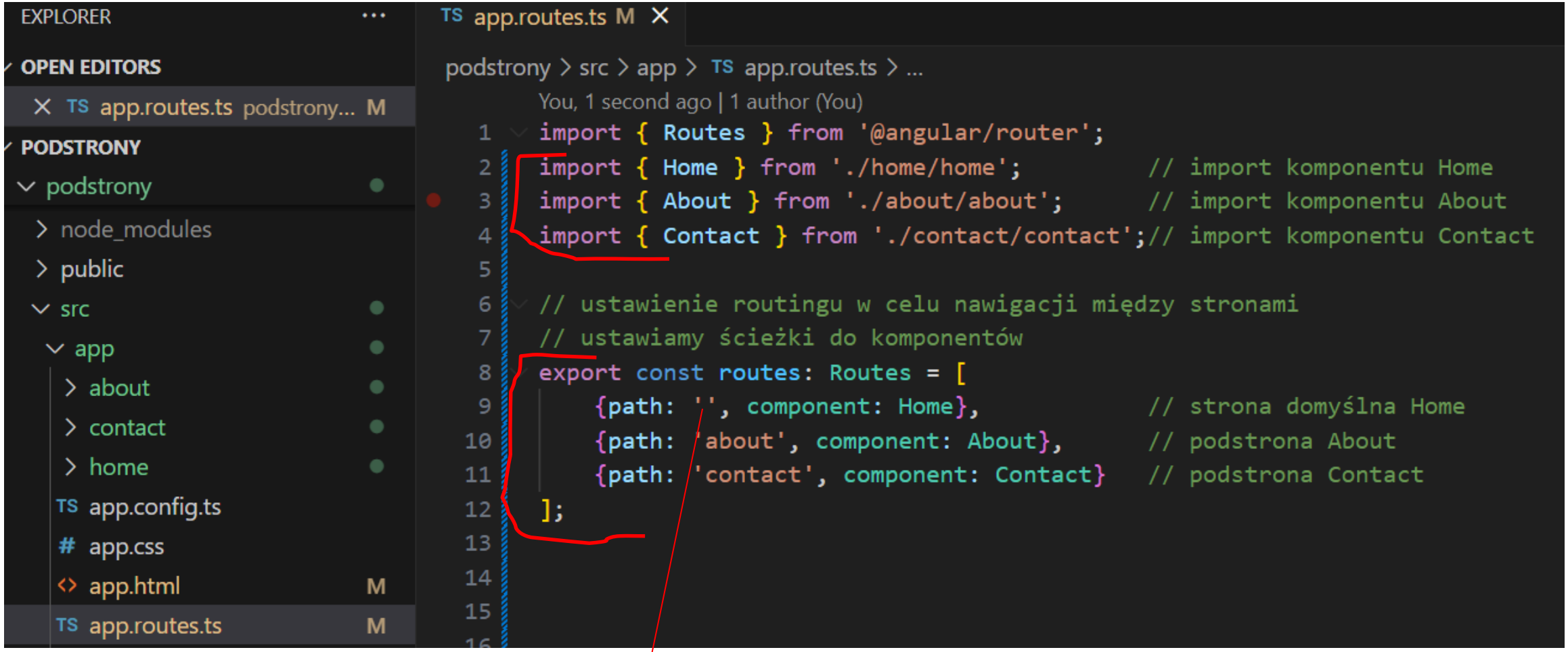






routing - podstrony

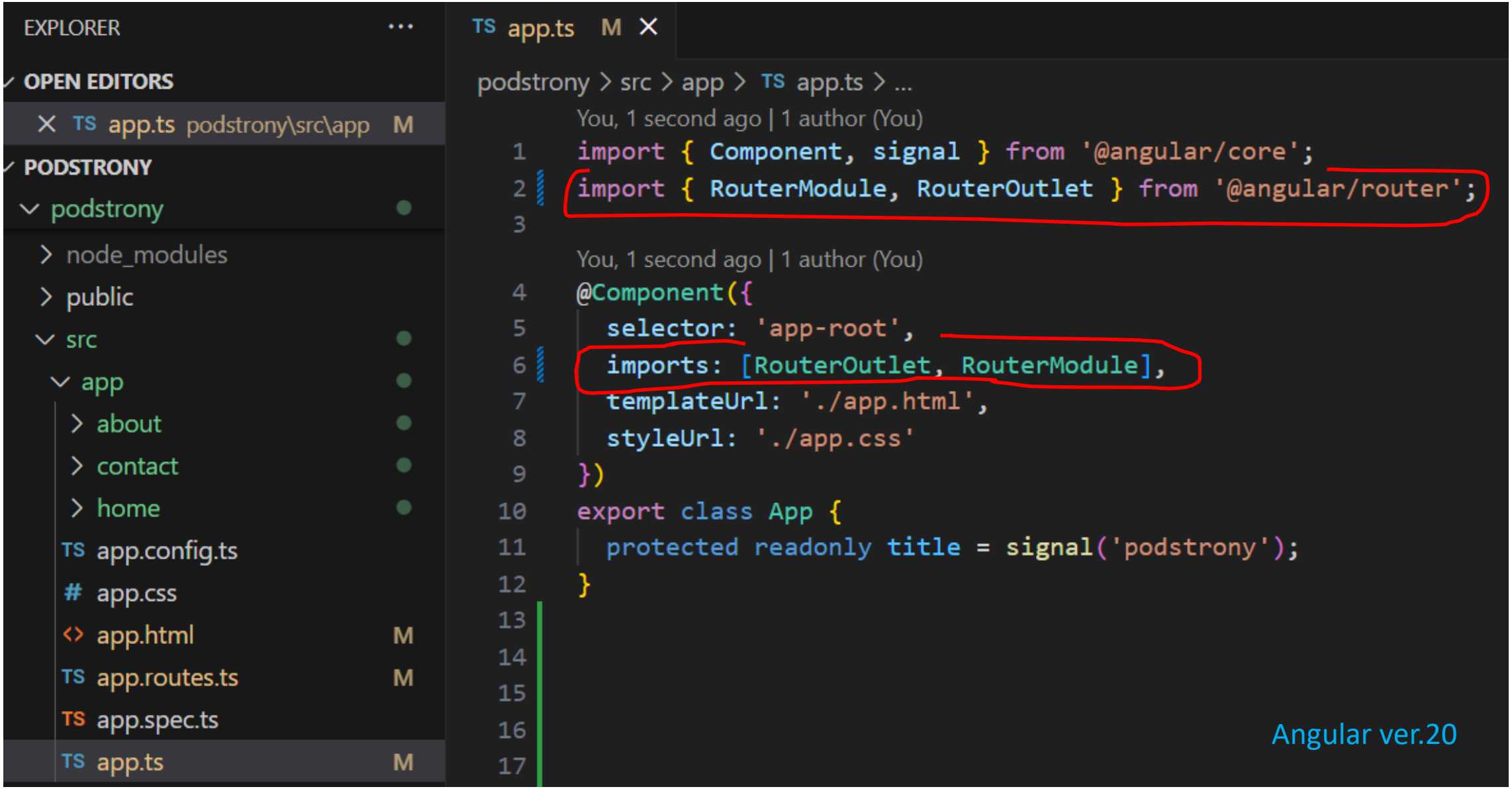
app.routes.ts – w tym pliku ustawiamy ścieżki do podstron (komponentów)



między apostrofami nie ma spacji!!!!

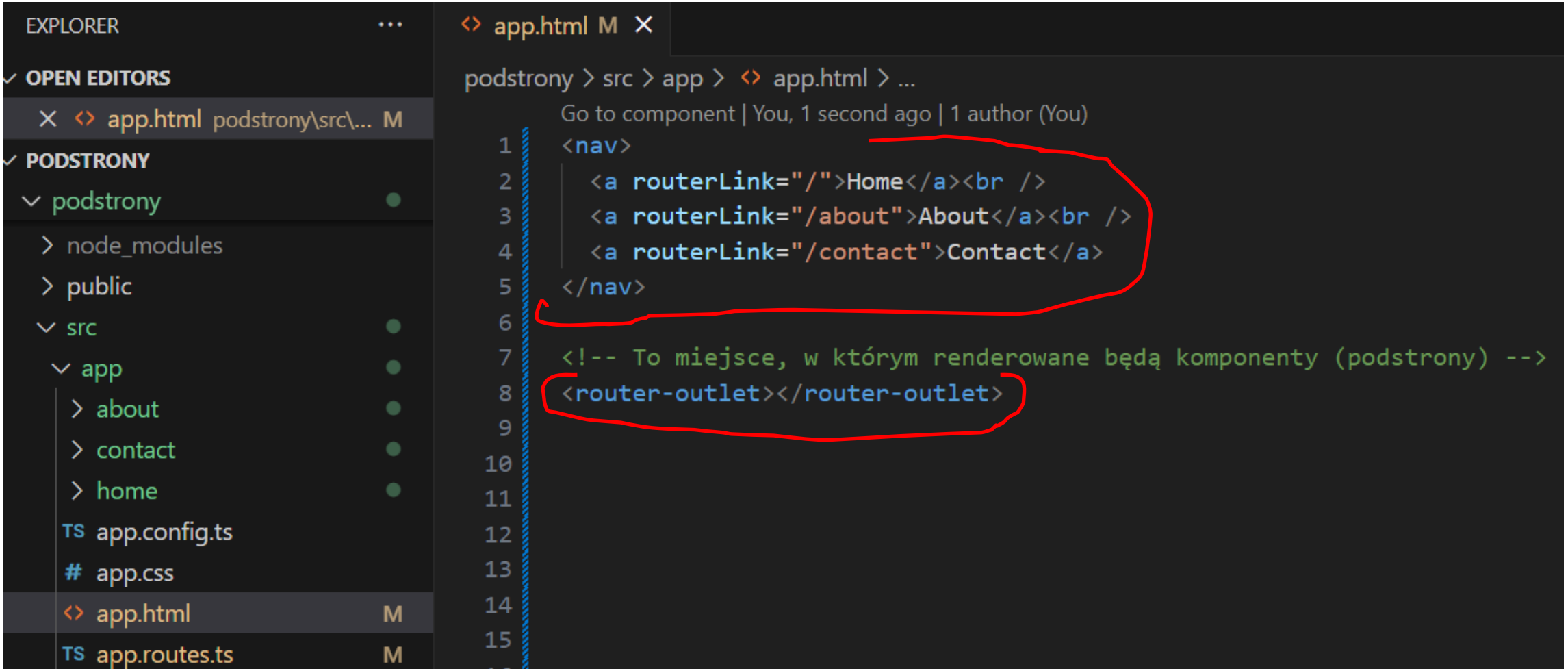
routing - podstrony

app.ts – w tym pliku importujermy biblioteki sterujące routingiem



routing - podstrony

app.html – w tym pliku tworzymy linki do poszczególnych podstron oraz wstawiamy miejsce ich renderowania



Angular ver.20

API

API to sposób, w jaki inne programy mogą **komunikować się z backendem**. Udostępnia zestaw **endpointów** (czyli adresów URL), które przyjmują i zwracają dane — zwykle w formacie **JSON** lub **XML**.

Np. aplikacja webowa może wysłać zapytanie do backendu:

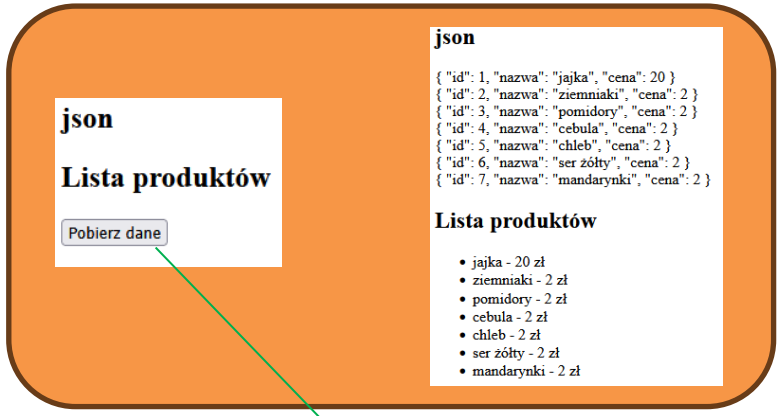
GET http://localhost:3002/api/getAll

a backend odpowie:


```
[
  { "id": 1, "nazwa": "jajka", "cena": 20 }
  { "id": 2, "nazwa": "ziemniaki", "cena": 2 }
  { "id": 3, "nazwa": "pomidory", "cena": 2 }
  { "id": 4, "nazwa": "cebula", "cena": 2 }
  { "id": 5, "nazwa": "chleb", "cena": 2 }
  { "id": 6, "nazwa": "ser żółty", "cena": 2 }
  { "id": 7, "nazwa": "mandarynki", "cena": 2 }
]
```

Połączenie z bazą danych MySQL poprzez api

Angular (frontend)



komunikacja Angulara z bazą danych MySQL poprzez server Node.js (backend)

request
http://localhost:3002/api/getAll

response

response

request

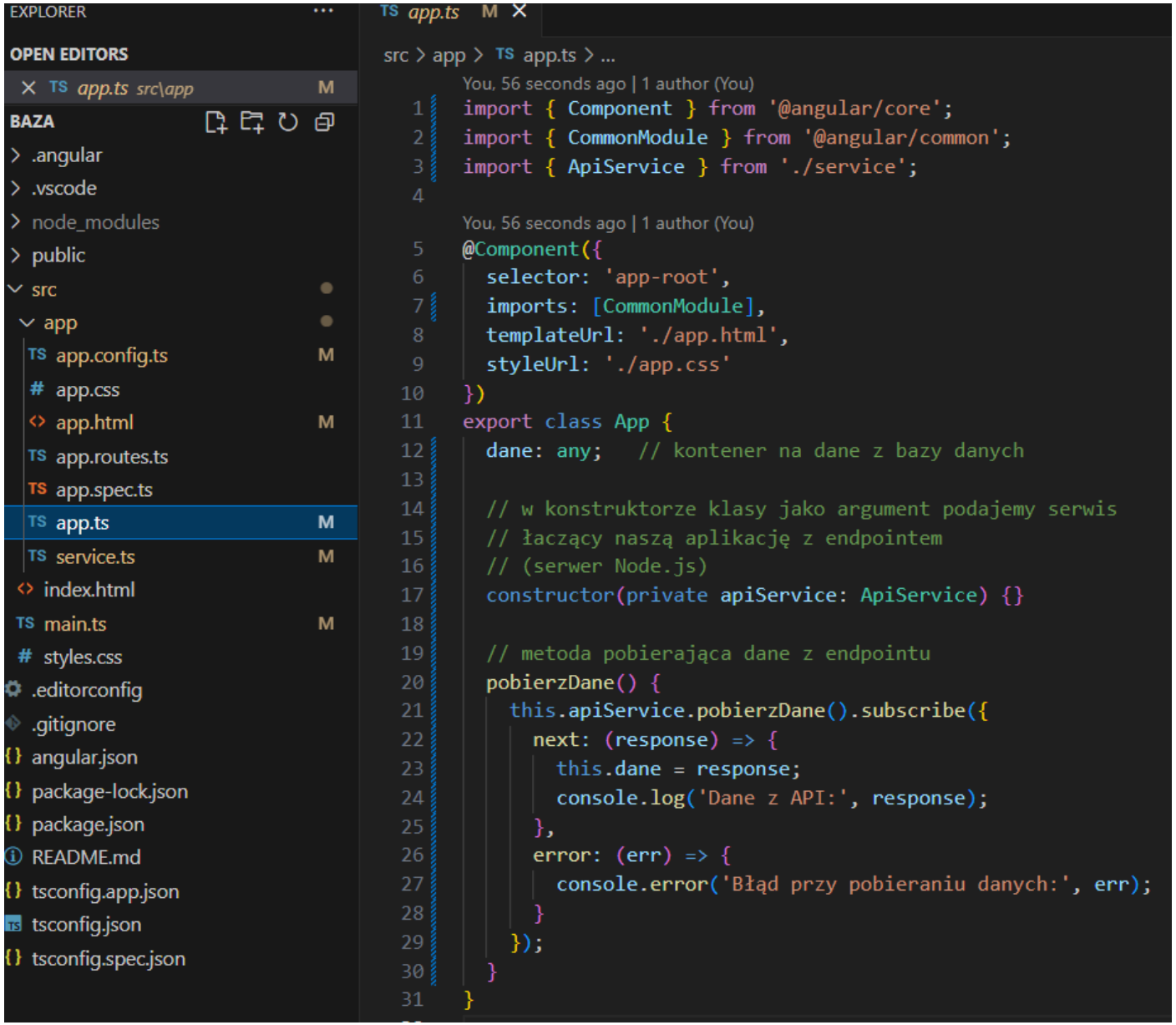
"/api/getAll" => "SELECT * FROM towary"

server Node.js Express nasłuchuje na http://localhost:3002 (endpoint)

baza danych sklep MySQL (XAMPP)

Połączenie z bazą danych MySQL
Angular (frontend)

app.ts



komenda w terminalu:

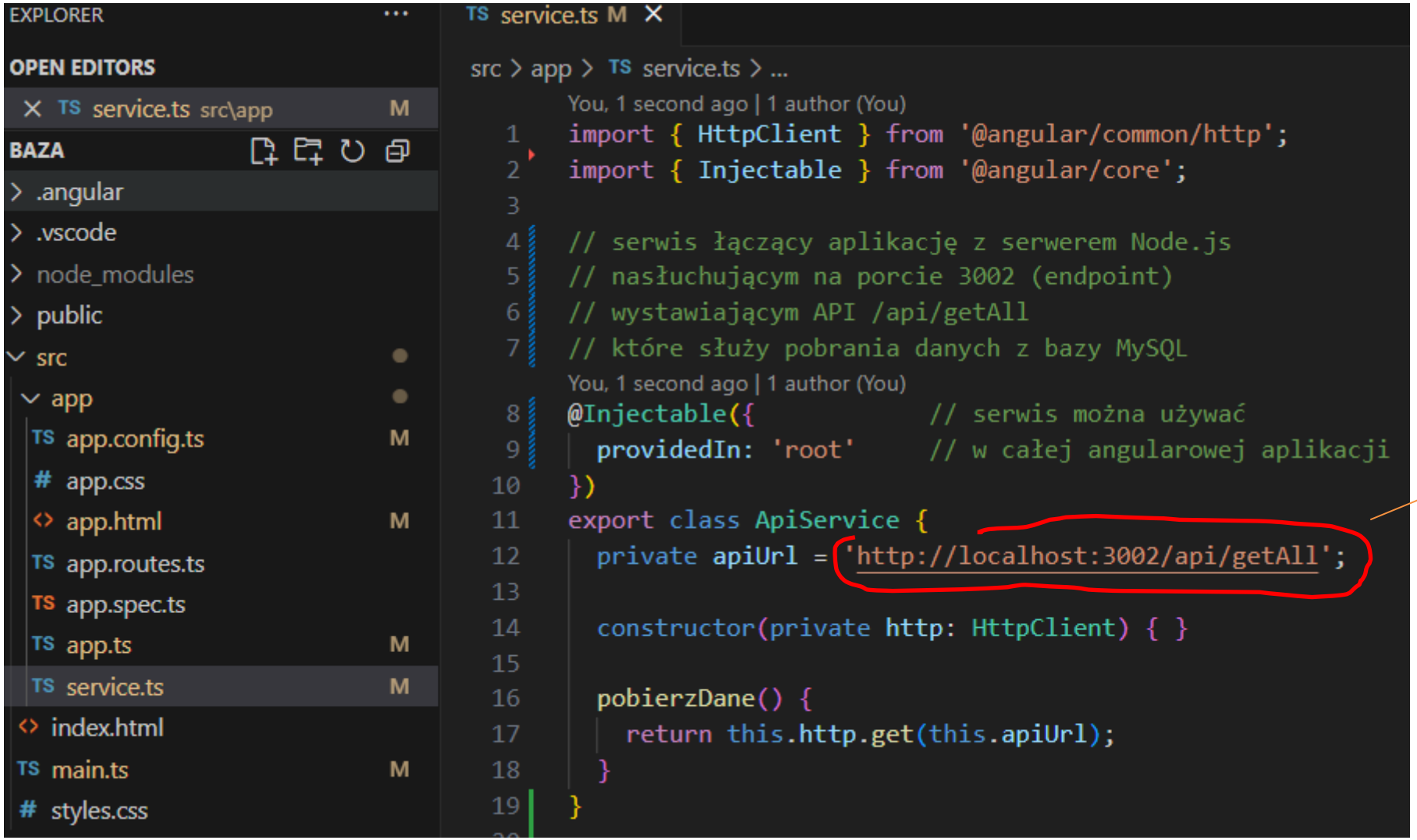
ng new baza

Angular ver.20

Połączenie z bazą danych MySQL
Angular (frontend)

service.ts serwis do połączenia z endpointem

plik *service.ts*
tworzymy ręcznie
(lub poleceniem:
`ng g s nazwa_serwisu`

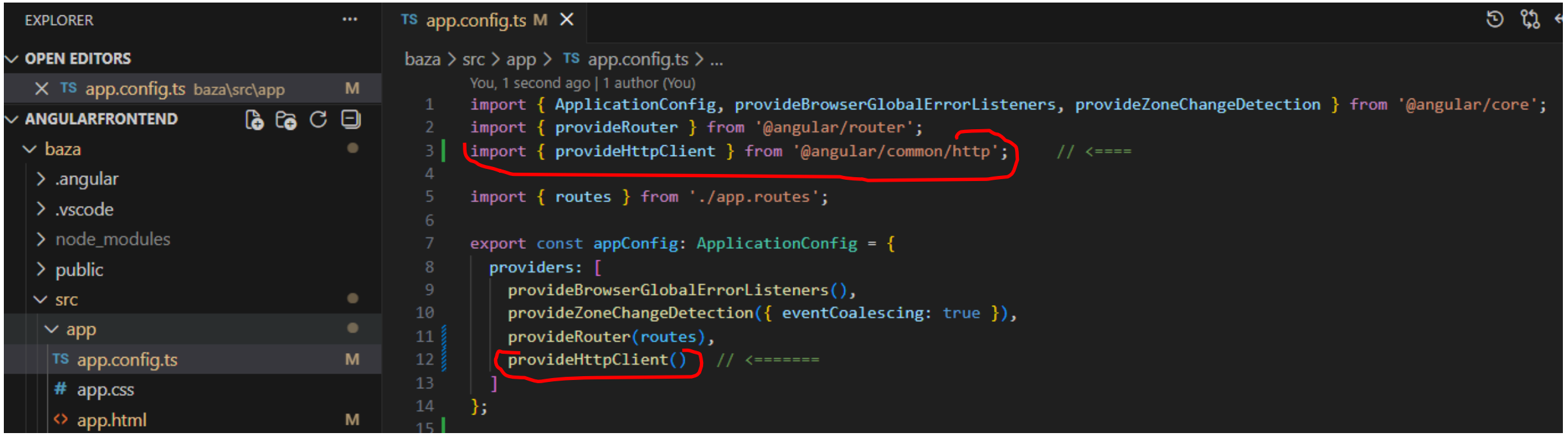


adres API

Angular ver.20

Połączenie z bazą danych MySQL
Angular (frontend)

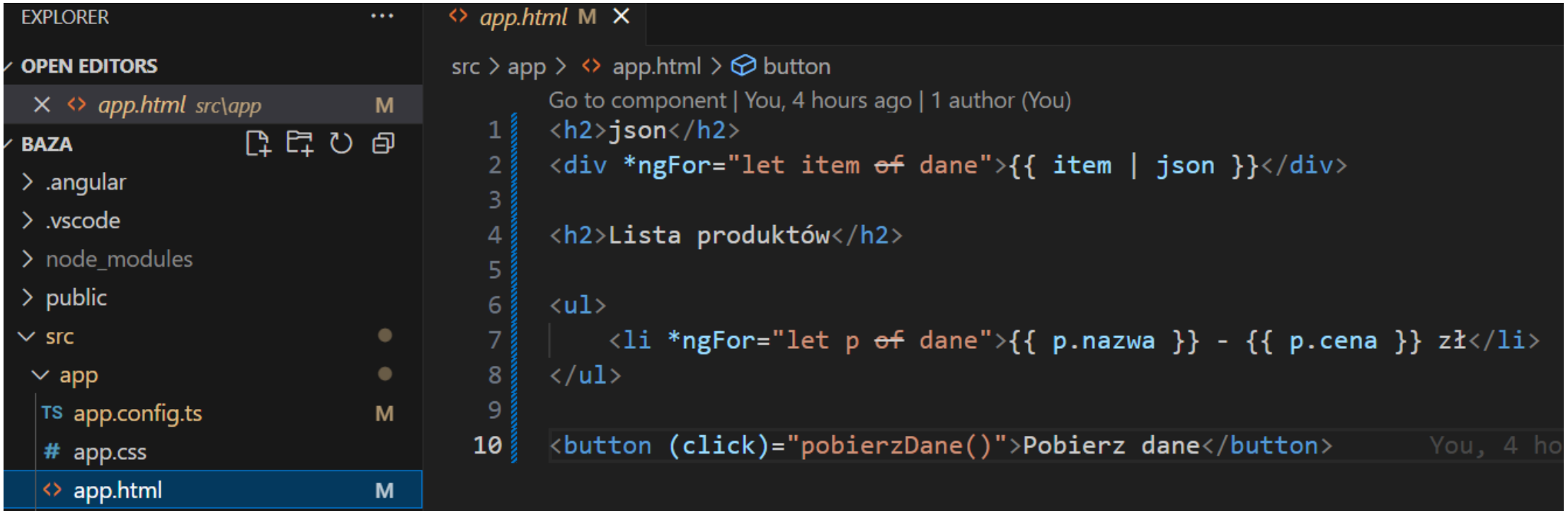
app.config.ts



Angular ver.20

Połączenie z bazą danych MySQL
Angular (frontend)

app.html

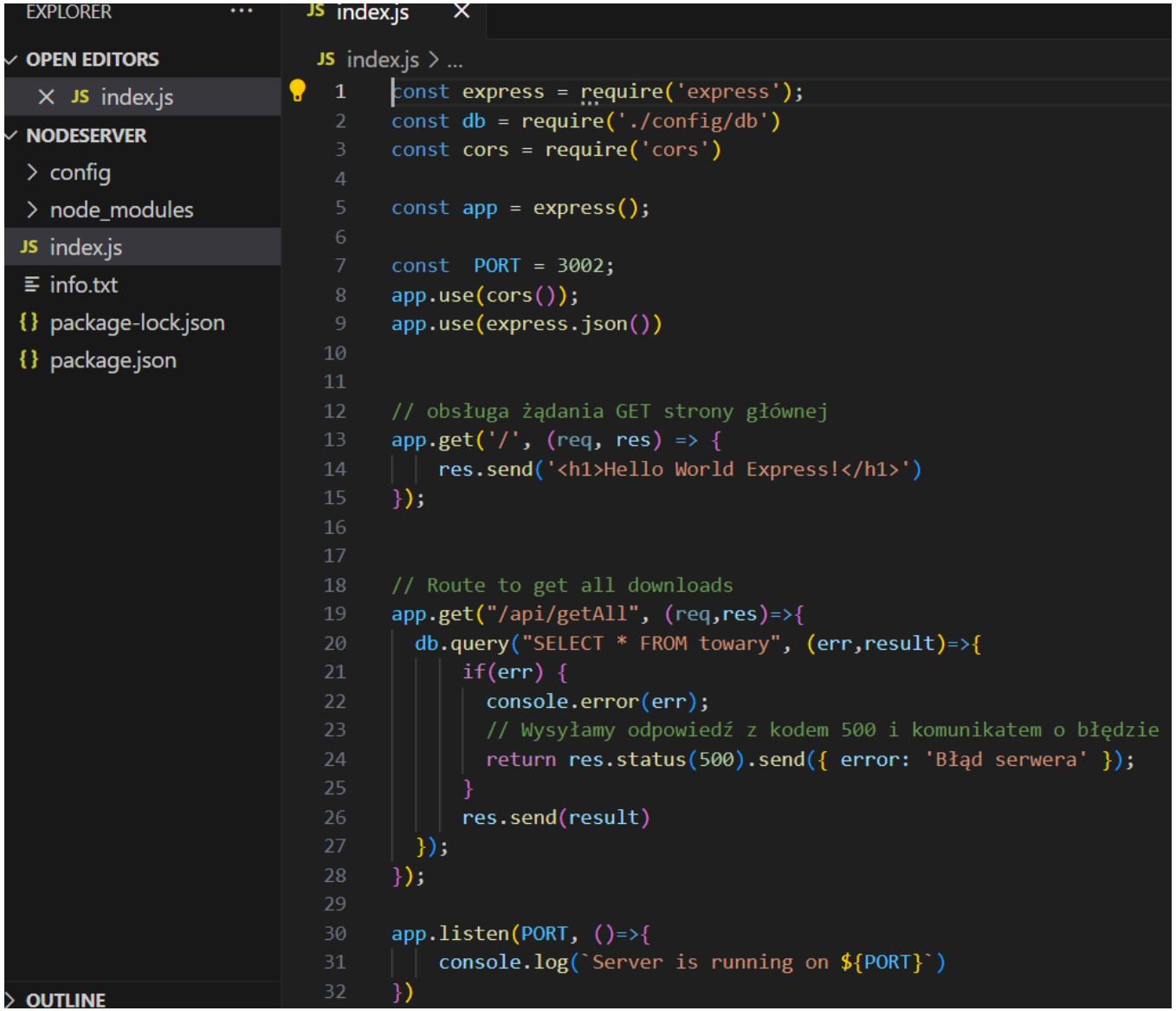


uruchomienie Angulara w terminalu:

ng serve

Angular ver.20

Połączenie z bazą danych MySQL
Node.js Express (backend)




```
/**
 * serwer, który pobiera dane z bazy MySQL sklep
 * tabela: towary    // towary w sklepie
 * pola: id          // id towaru
 *       cena        // cena towaru
 *
 */
```

serwer Node.js

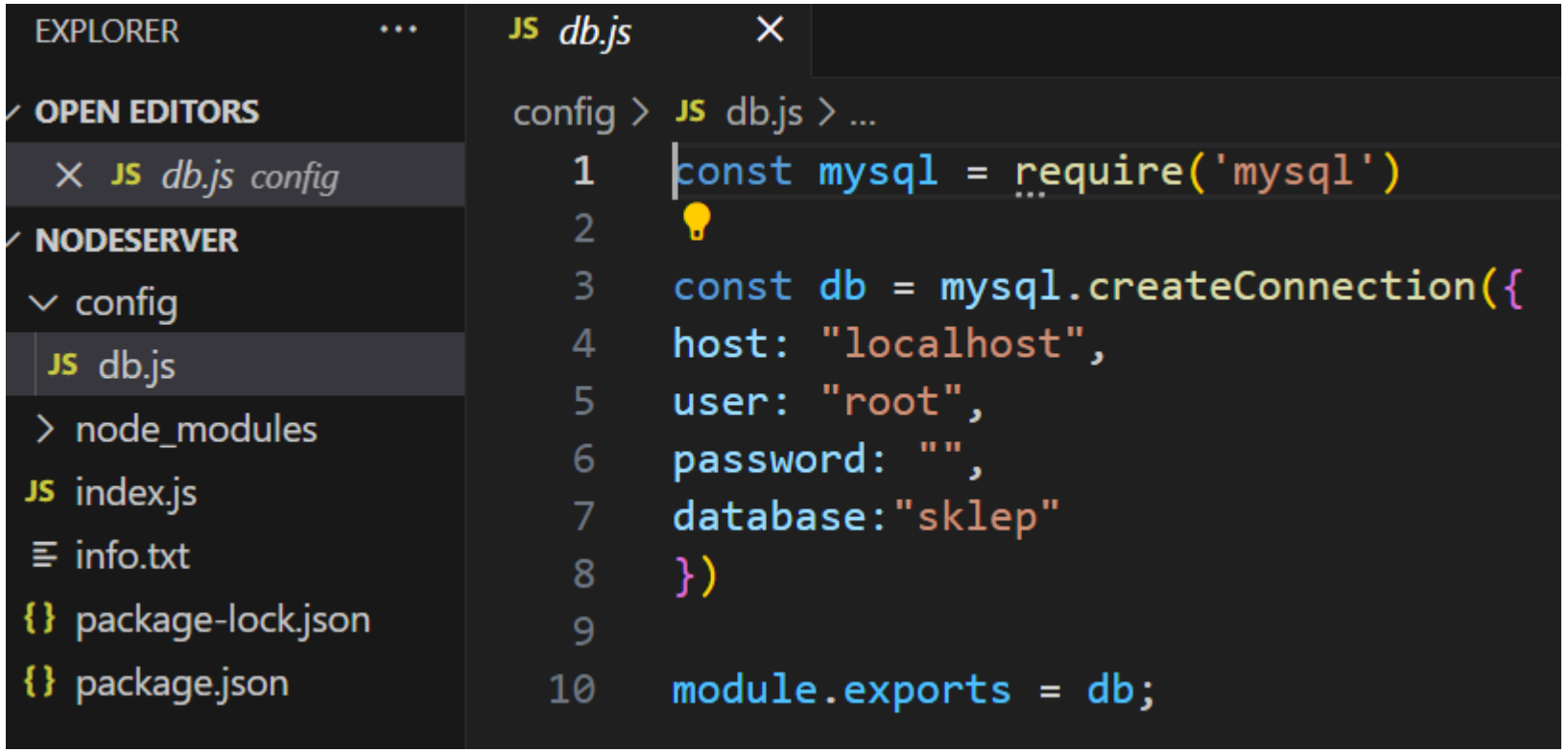
1. plik index.js tworzymy ręcznie

2. komendy w terminalu:

npm init -y
npm install express
npm install mysql
npm install cors

Połączenie z bazą danych MySQL
Node.js Express (backend)

plik konfiguracyjny config/db.js z danymi dostępowymi do bazy danych MySQL

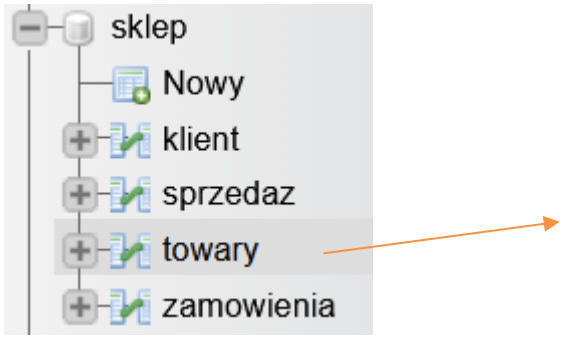


uruchomienie serwera w terminalu:

node index.js

Połączenie z bazą danych MySQL
baza sklep MySQL

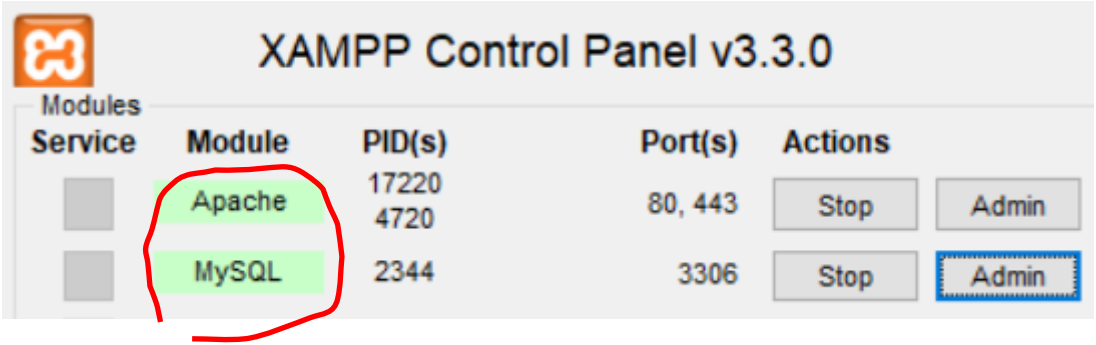
bazę danych sklep wykonujemy w środowisku XAMPP za pomocą phpmyadmin (na początek wystarczy tabela towary)



		id	nazwa	cena
☐	Edytuj Kopiuj Usuń	1	jajka	20
☐	Edytuj Kopiuj Usuń	2	ziemniaki	2
☐	Edytuj Kopiuj Usuń	3	pomidory	2
☐	Edytuj Kopiuj Usuń	4	cebula	2
☐	Edytuj Kopiuj Usuń	5	chleb	2
☐	Edytuj Kopiuj Usuń	6	ser żółty	2
☐	Edytuj Kopiuj Usuń	7	mandarynki	2

Połączenie z bazą danych MySQL
baza sklep MySQL

w panelu kontrolnym XAMPP uruchamiamy serwery Apache i MySQL

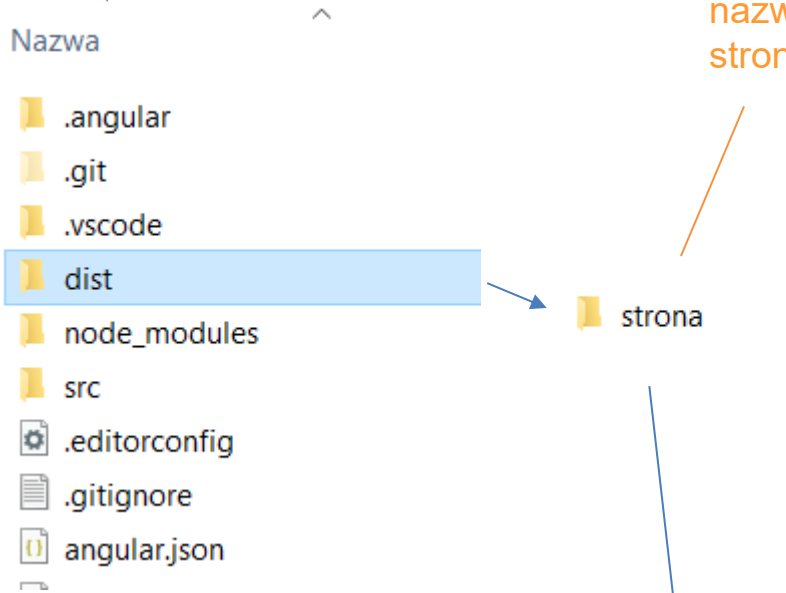


deployment

aby projekt strony wykonany w Angularze przygotować do umieszczenia na zewnętrznym serwerze (np. Apache w XAMPP) należy w folderze projektu wykonać polecenie: **ng build –base-href ./**

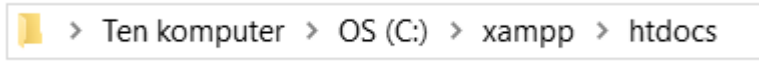
```
ng build --base-href ./
```

w wyniku wykonania polecenia w folderze projektu wygeneruje się folder dist/nazwa_projektu



nazwa_projektu w przykładzie to strona

folder nazwa_projektu umieszczamy w foderze htdocs serwera Apache (XAMPP)



deployment

projekt „strona” hostowany za pomocą programu XAMPP

Angular instalacja

instalacja:

https://nodejs.org/en/ // instalacja Node.js

```
npm install -g @angular/cli
```

w powershelu wpisujemy:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

sprawdzenie zainstalowanej wersji:

```
node -v
npm -v
ng --version
```

Angular nowy projekt

```
// komendy wpisujemy w terminalu Visual Studio Code

ng new nazwa_projektu      // utworzenie nowego projektu
cd nazwa_projektu          // przejście do folderu projektu
ng serve                   // uruchomienie serwera
http://localhost:4200      // należy uruchomić przegladarkę
```

Angular update

instalacja najnowszej wersji Node.js:

https://nodejs.org/en/

```
// komendy wpisujemy w cmd

npm uninstall –g @angular/cli // odinstalowanie starego CLI
npm install –g @angular/cli@latest // instalacja nowego CLI
```

https://angular.dev/update-guide

bootstrap

w katalogu projektu w terminalu instalujemy bibliotekę *bootstrap*:

```
npm install bootstrap --save
```

w pliku *styles.css* dodajemy import:

```
@import „bootstrap/dist/css/bootstrap.css”;
```