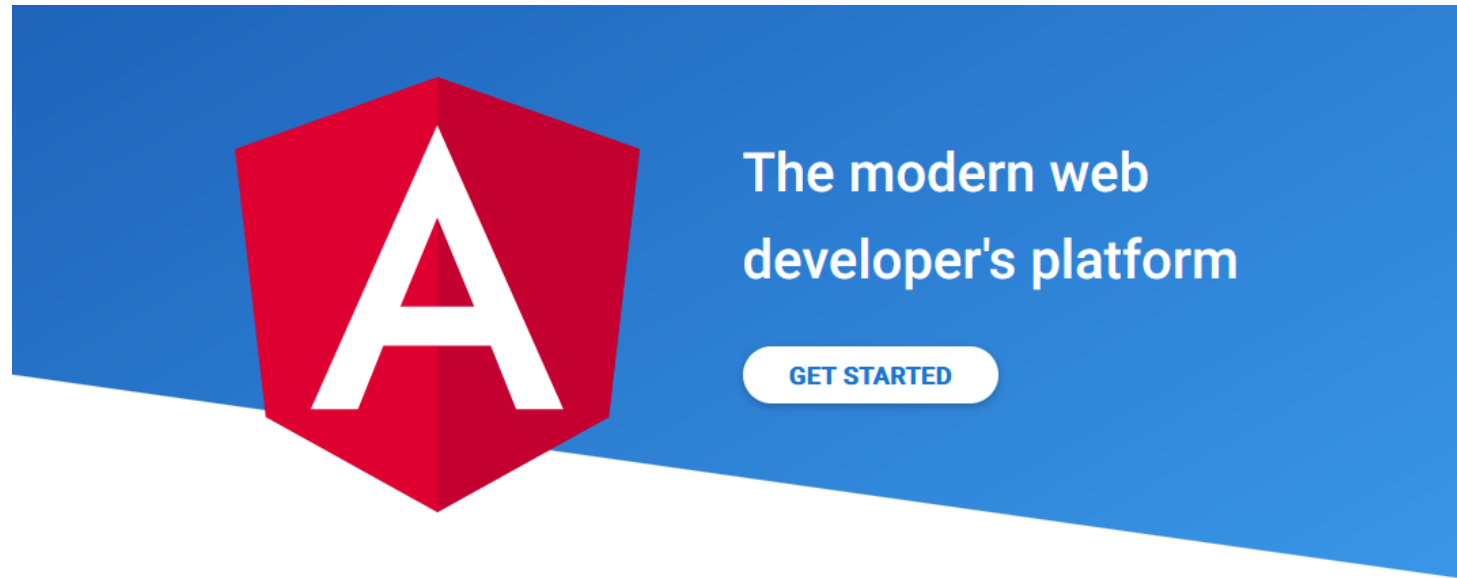


Angular

wersje: 14, 18, 20, 21

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

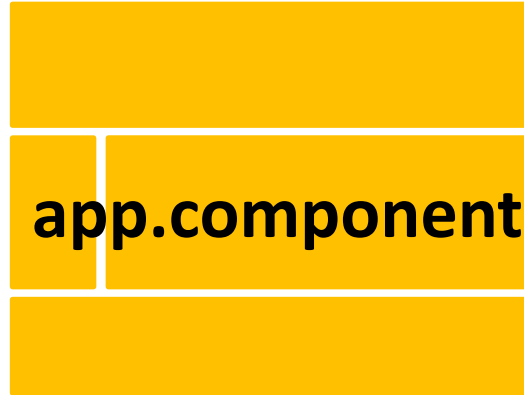


służy do przekazywania treści
użytkownikowi oraz do interakcji z nim

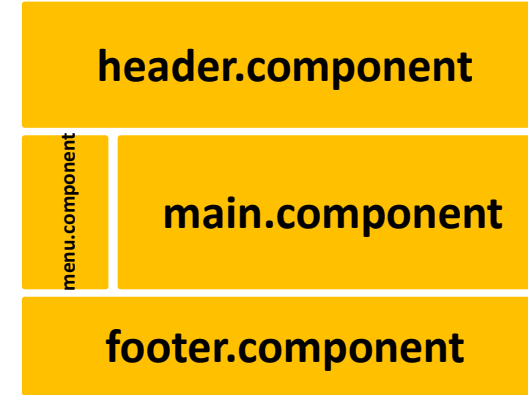
składa się z klasy (logika i dane),
pliku HTML (szablonu), pliku
stylów oraz ma dostęp do
fragmentu strony internetowej,
gdzie wyświetla treści.

Component

jeden komponent



kilka komponentów



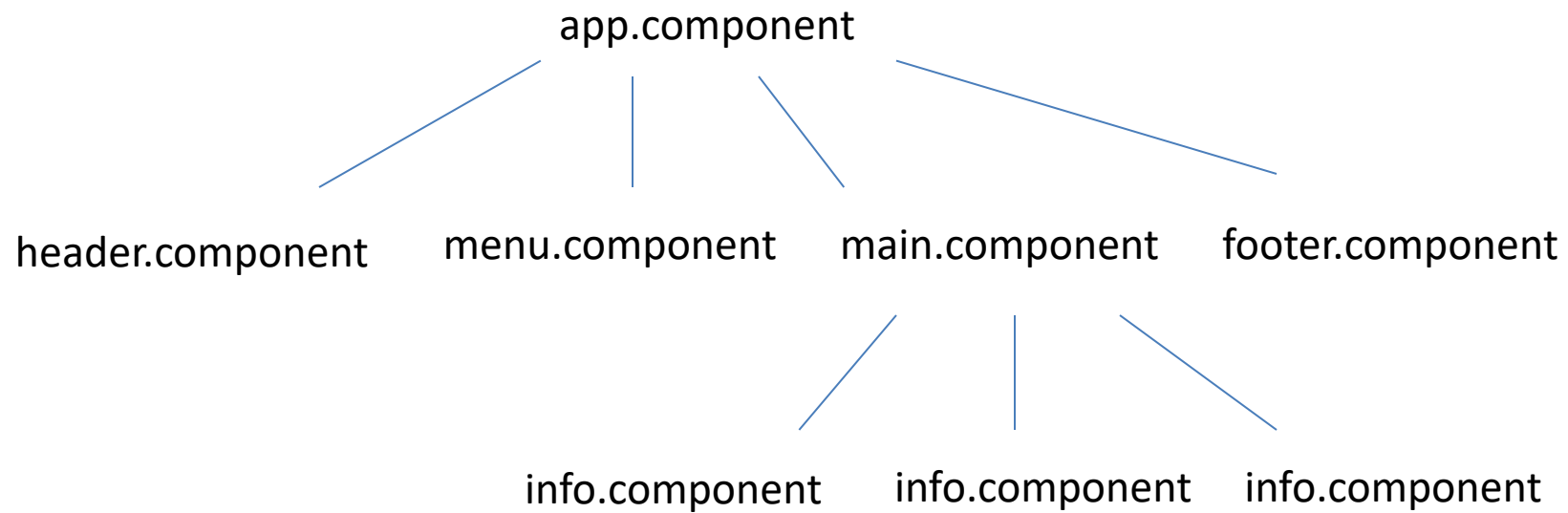
kilka komponentów



aplikacja składa się z komponentów, można różnie ją zaprojektować

Component

aplikacja posiada co najmniej jeden
komponent główny app.component



komponenty aplikacji tworzą drzewo komponentów

Komponenty samodzielne (standalone)

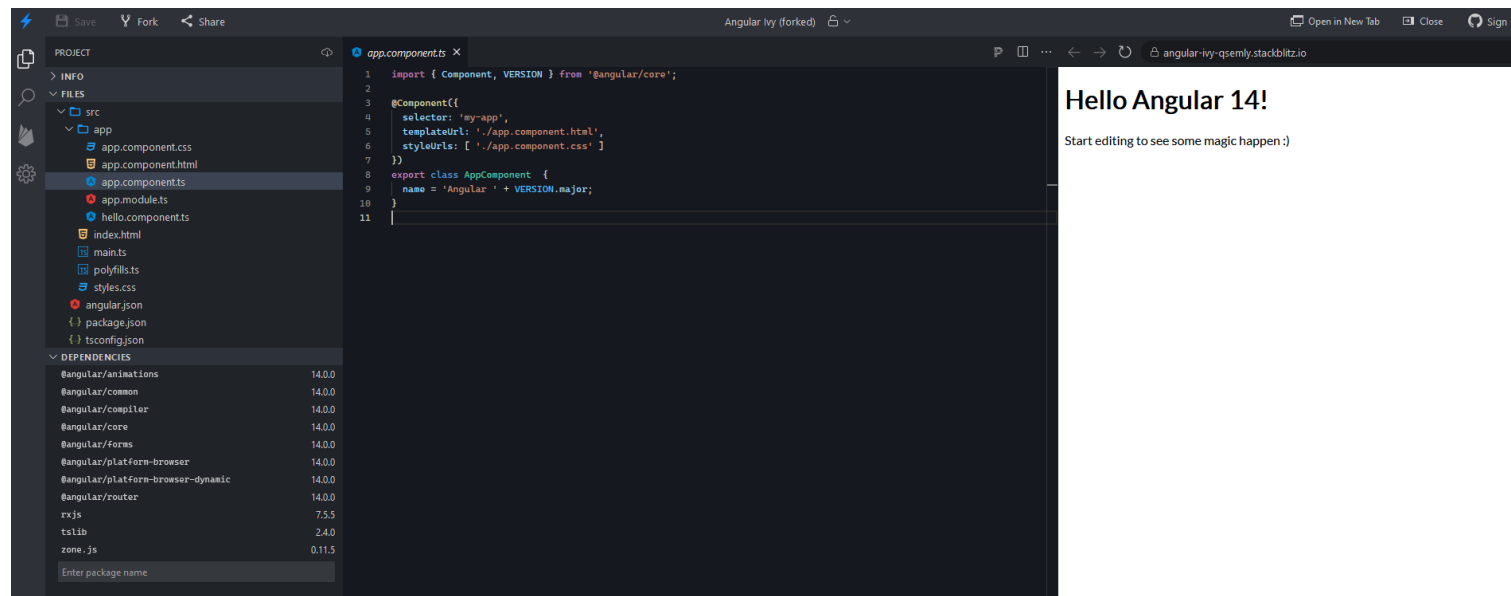
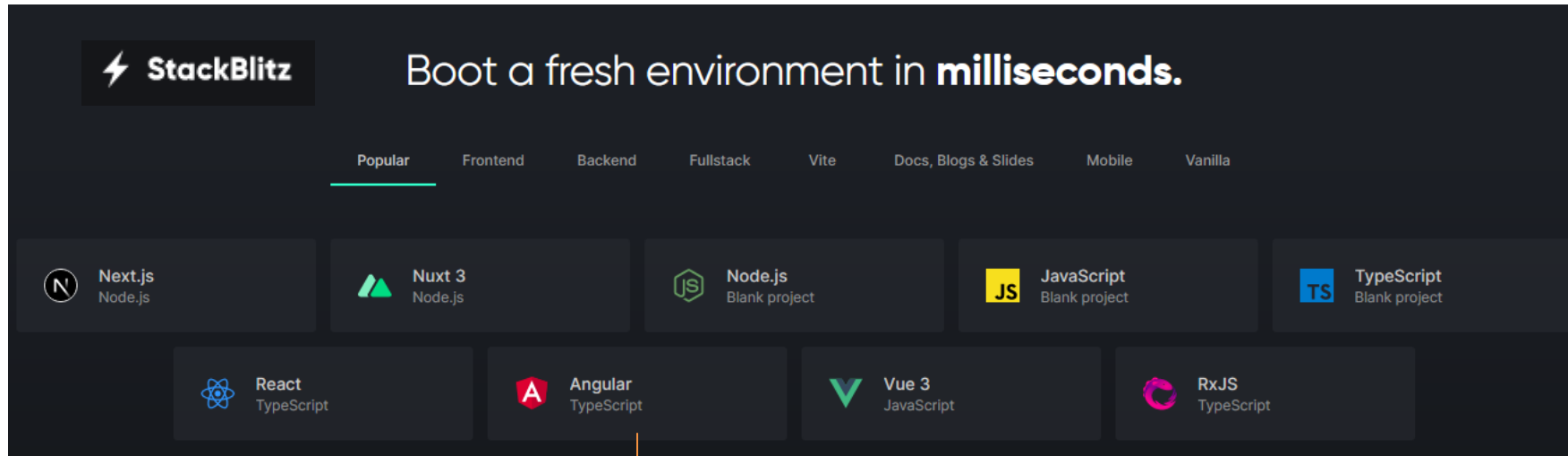
Komponenty, dyrektywy i potoki można oznaczać jako ***samodzielne (standalone: true)***. Klasy Angular oznaczone jako samodzielne nie muszą być deklarowane w NgModule (kompilator Angular zgłosi błąd, jeśli je spróbujemy zadeklarować w ten sposób).

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



v18



Ctrl K



Docs



Tutorials



Playground



Reference

Learn Angular
Introduction



Welcome to the Angular tutorial

This interactive tutorial will teach you the basic building blocks to start building great apps with Angular.

How to use this tutorial

You'll need to have basic familiarity with HTML, CSS and JavaScript to understand Angular.

Each step represents a concept in Angular. You can do one, or all of them.

If you get stuck, click "Reveal answer" at the top.

Alright, let's [get started](#).

app/app.component.ts +

```
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     Welcome to Angular!
7   `,
8   standalone: true,
9 })
10 export class AppComponent {}
11
```

Preview Console Terminal

Welcome to Angular!

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

Instalacja Node.js

Node.js oraz menadżer pakietów npm

<https://nodejs.org/en/>



Long Term Support
długi okres wsparcia technicznego

wersja bieżąca

sprawdzenie zainstalowanej wersji

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

C:\>npm -v
8.19.2
```

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

Introduction

Getting started

- What is Angular?
- Try it
- Setup

Understanding Angular

Developer guides

Best practices

Angular tools

Tutorials

Updates and releases

Reference

Documentation contributors guide

Install the Angular CLI

You can use the Angular CLI to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

To install the Angular CLI, open a terminal window and run the following command:

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

On Windows client computers, the execution of PowerShell scripts is disabled by default. To allow the execution of PowerShell scripts, which is needed for npm global binaries, you must set the following [execution policy](#):

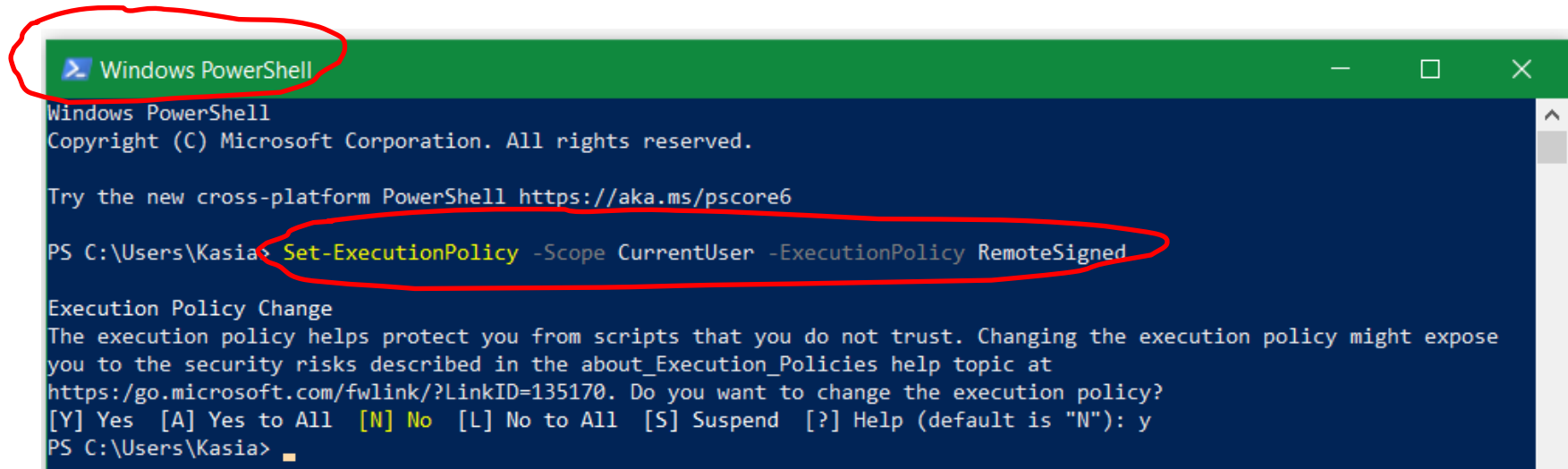
```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Carefully read the message displayed after executing the command and follow the instructions. Make sure you understand the implications of setting an execution policy.

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

w systemie Windows należy włączyć możliwość wykonywania skryptów w PowerShellu

Instalacja Angular CLI



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

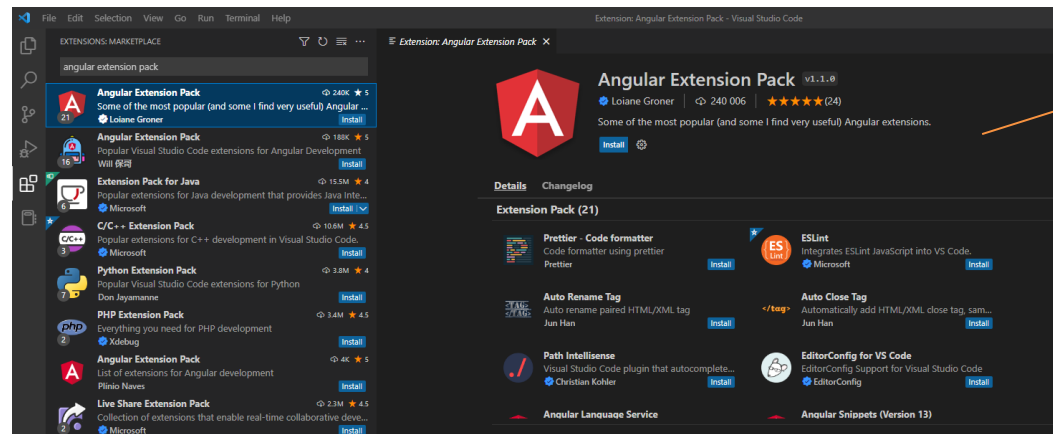
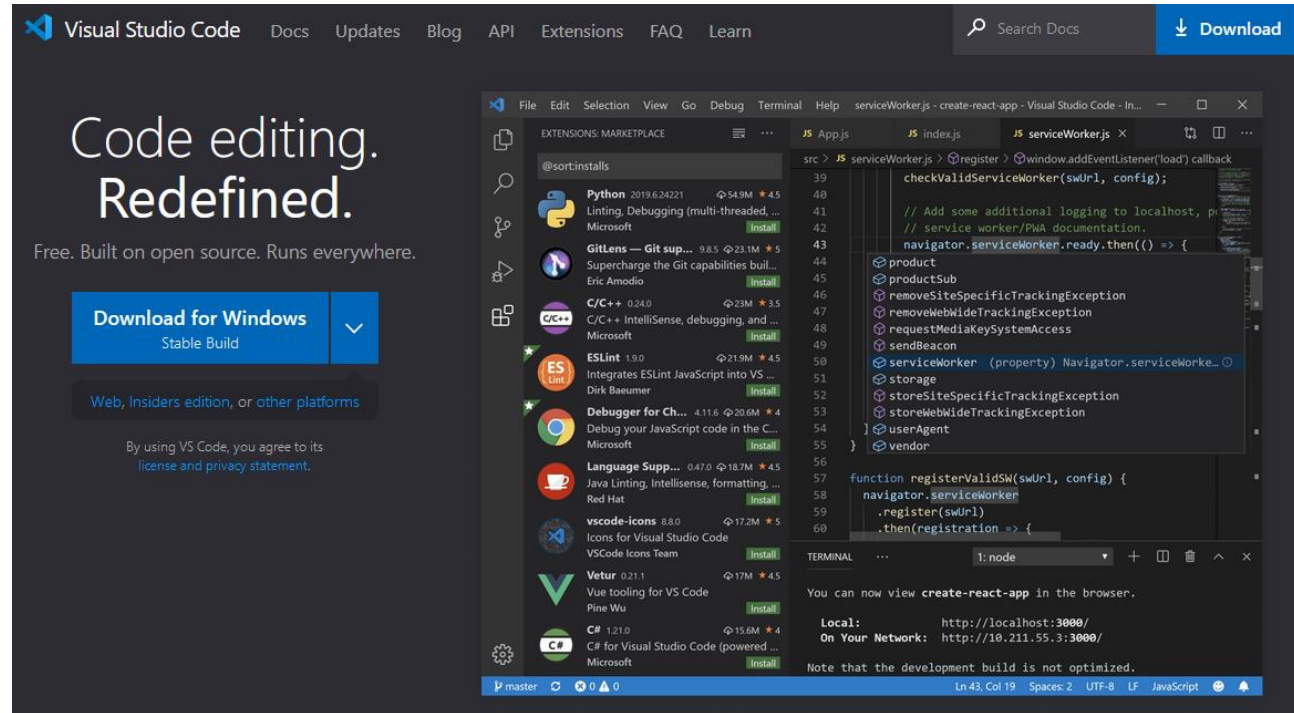
PS C:\Users\Kasia> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): y
PS C:\Users\Kasia>
```

komendę wykonujemy w PowerShellu, a nie w cmd

Instalacja Visual Studio Code

<https://code.visualstudio.com/>



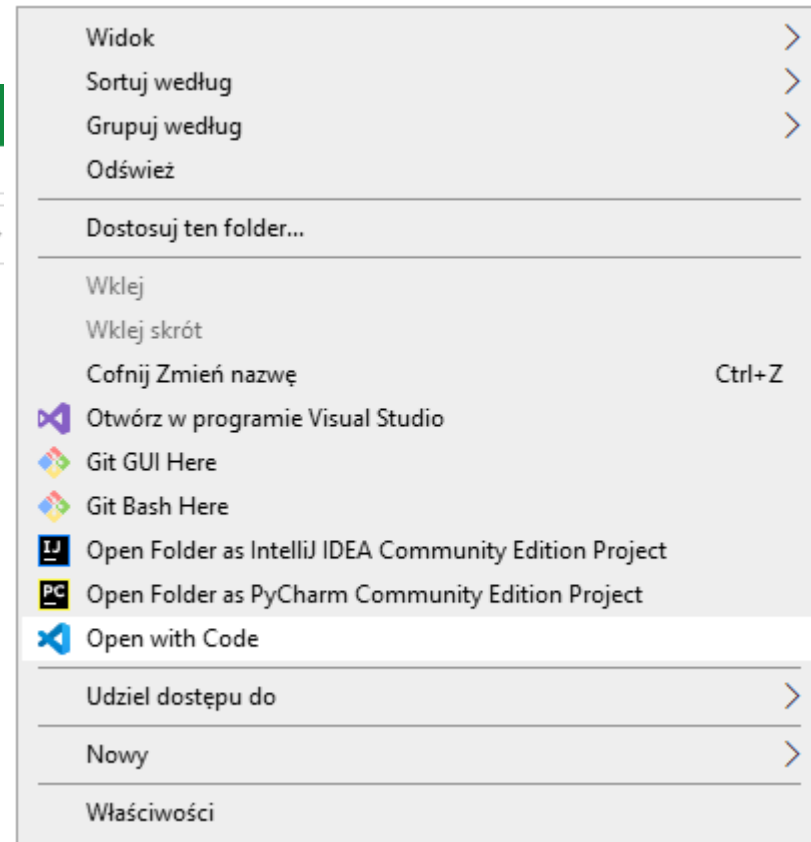
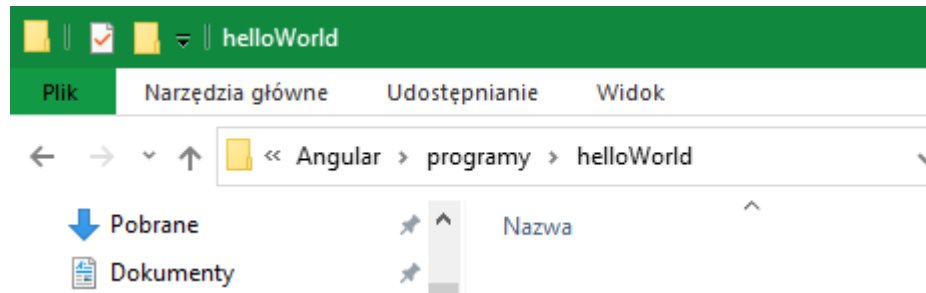
dodatek Angular Extension Pack

Tworzenie nowego projektu

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

helloWorld

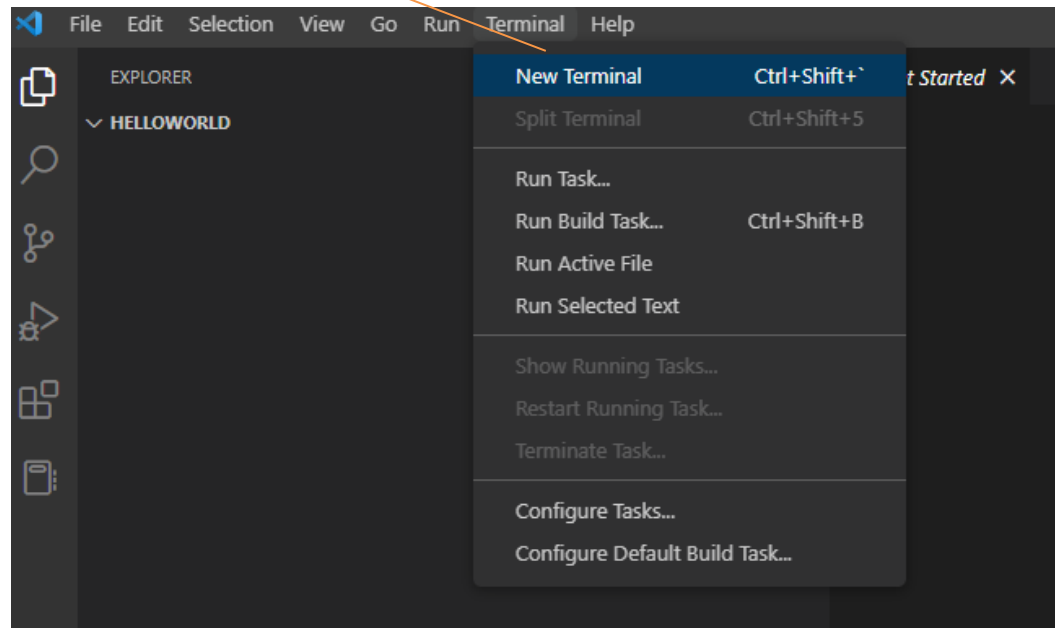
otwieramy folder helloWorld



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:

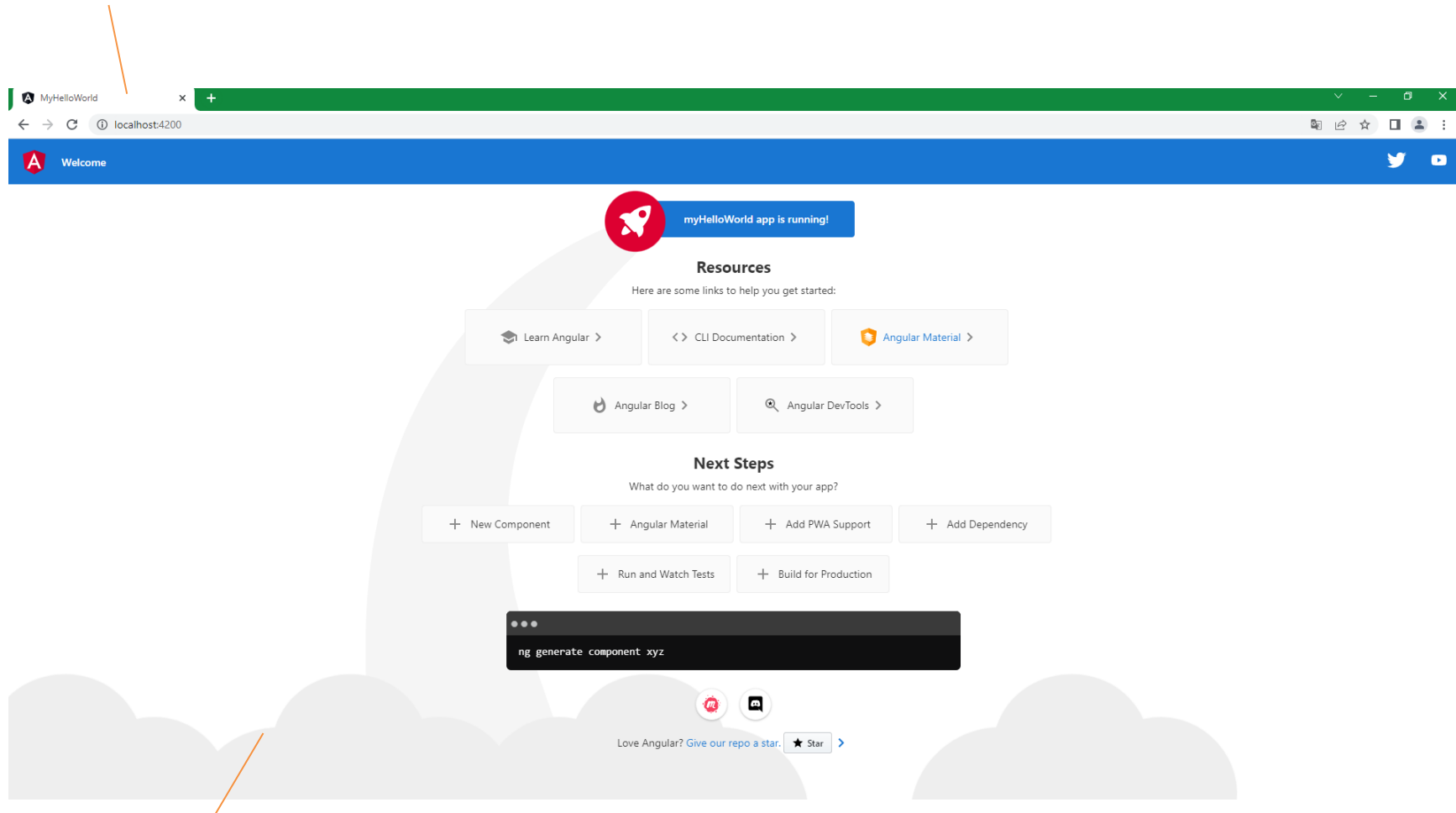
```
cd .\myHelloWorld\           // przejście do katalogu projektu  
ng serve                      // uruchomienie serwera
```

```
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

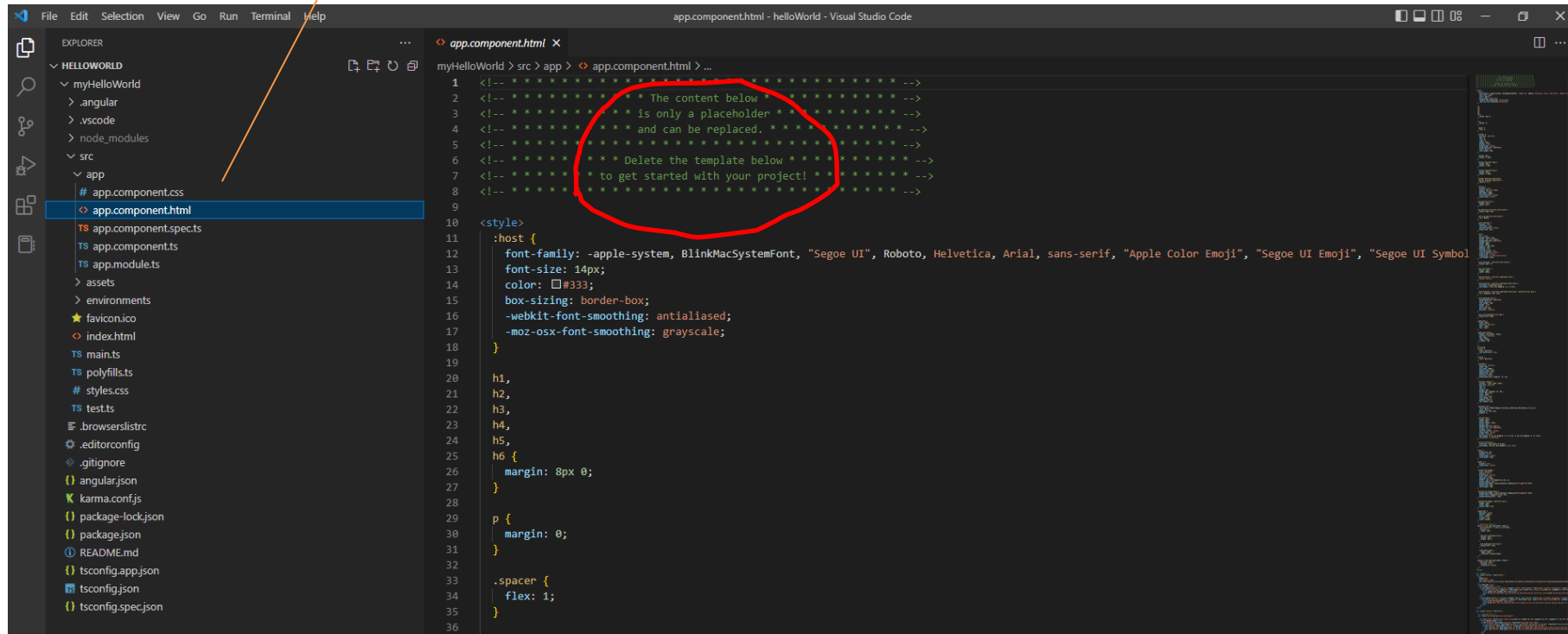
w pasku adresu przeglądarki wpisujemy:
localhost:4200



strona nowego projektu

Tworzenie nowego projektu

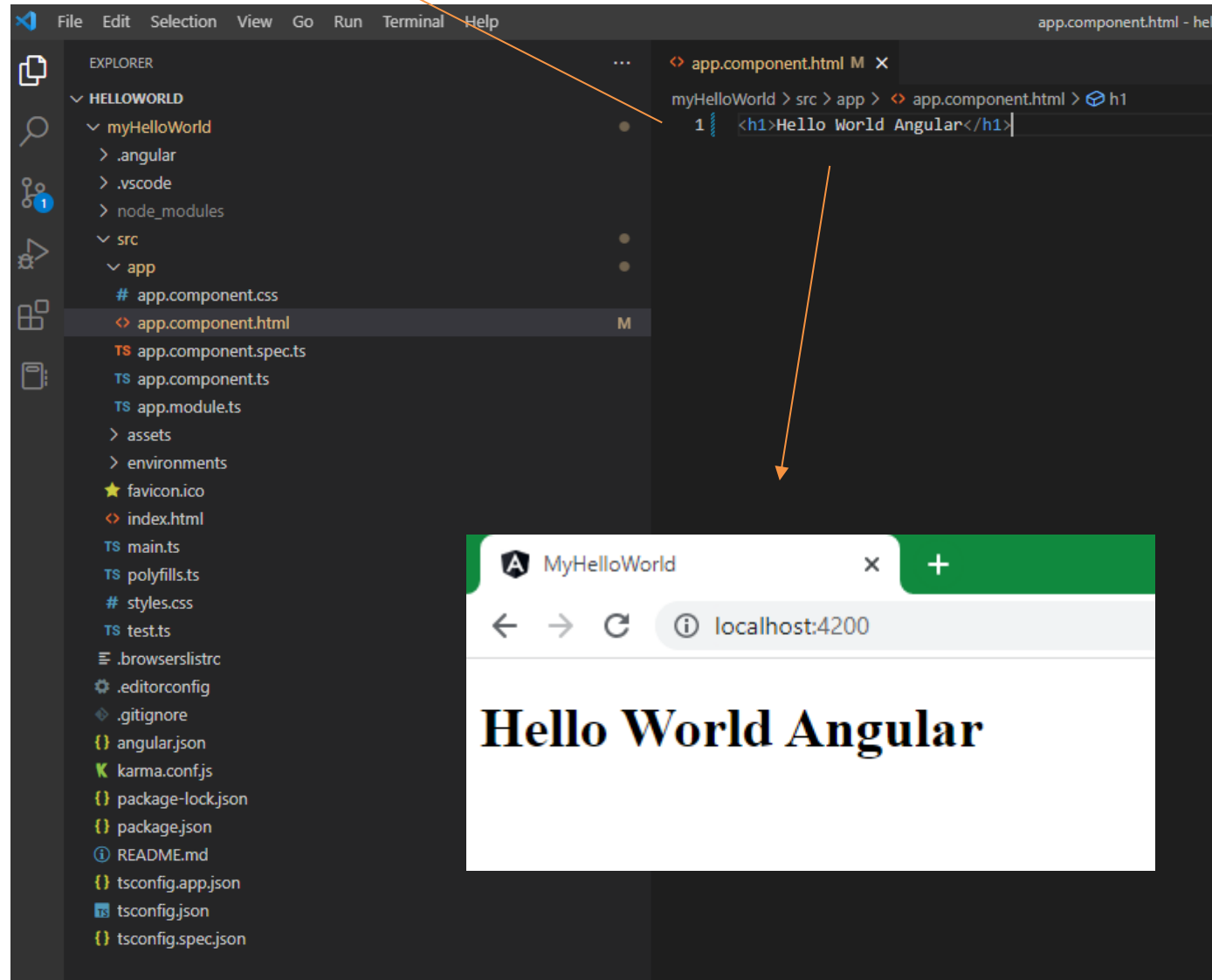
usuwamy zawartość pliku app.component.html (zawartość startowej strony)



```
1 <!-- ***** -->
2 <!-- ***** The content below ***** -->
3 <!-- ***** is only a placeholder ***** -->
4 <!-- ***** and can be replaced. ***** -->
5 <!-- ***** -->
6 <!-- ***** Delete the template below ***** -->
7 <!-- ***** to get started with your project! ***** -->
8 <!-- ***** -->
9
10 <style>
11 :host {
12   font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
13   font-size: 14px;
14   color: #333;
15   box-sizing: border-box;
16   -webkit-font-smoothing: antialiased;
17   -moz-osx-font-smoothing: grayscale;
18 }
19
20 h1,
21 h2,
22 h3,
23 h4,
24 h5,
25 h6 {
26   margin: 8px 0;
27 }
28
29 p {
30   margin: 0;
31 }
32
33 .spacer {
34   flex: 1;
35 }
36
```

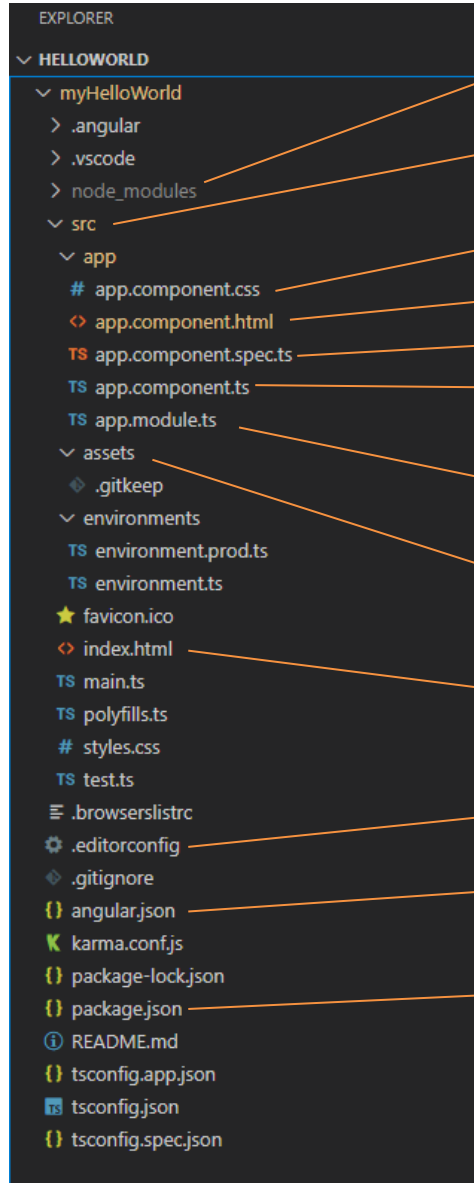
Tworzenie nowego projektu

w pliku `app.component.html` wpisujemy nasz kod



Angular ver.14

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)

dekorator
(konfiguracja
komponentu)

szablon HTML
powiązany
z komponentem

```
TS app.component.ts X
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   styleUrls: ['./app.component.css']
11 })
12 export class AppComponent {
13   title = 'myHelloWorld';
14 }
```

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.

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

```
<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

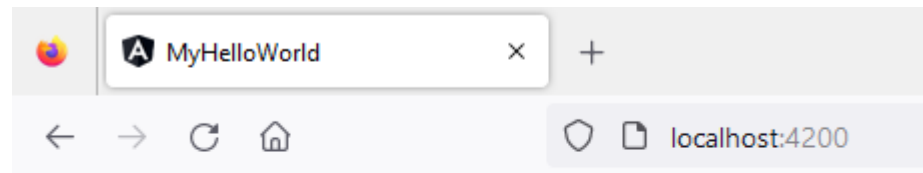
Angular ver.18

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



to jest template

Templates

Szablony to kod HTML, który Angular renderuje dla komponentu.

- **Bindings:** interpolation `{{ name_of_class_field }}` przykład: `<p>{{welcome}}</p>`
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.

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**.

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.

interpolacja

app.component.ts

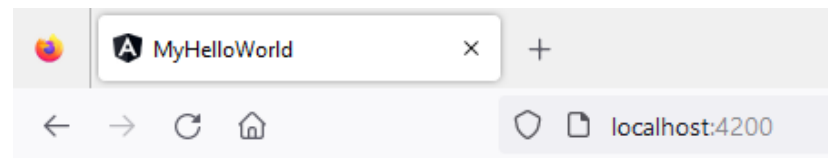
```
import { Component } from '@angular/core';

@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)



Powitać!

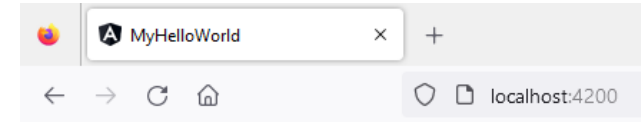
nowe pole o nazwie welcome o wartości typu string „powitać”

interpolacja - wstawianie wyrażenia, wywołanie metody

app.component.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

Powitać!

2*3=6

wywołanie
metody

POWITAĆ!

Template reference variables

(zmienne referencyjne w szablonie)

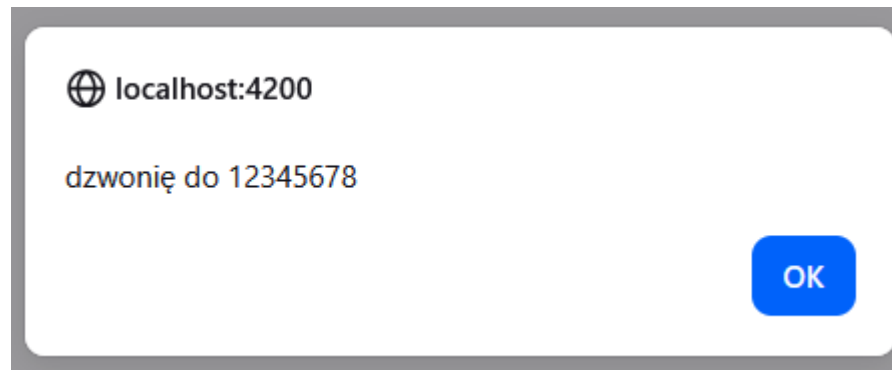
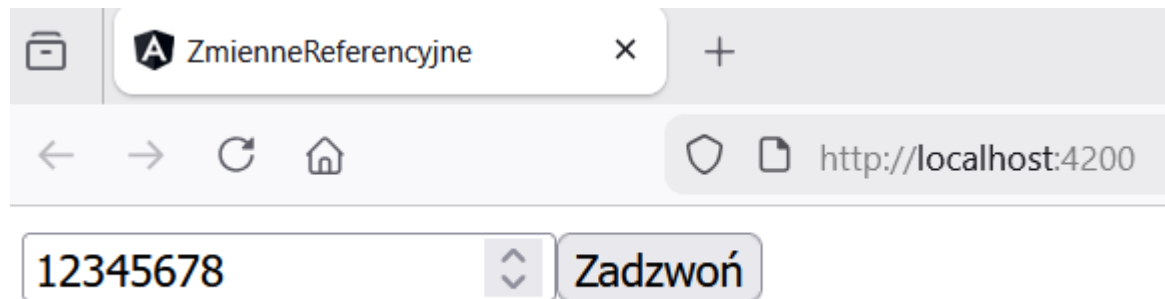
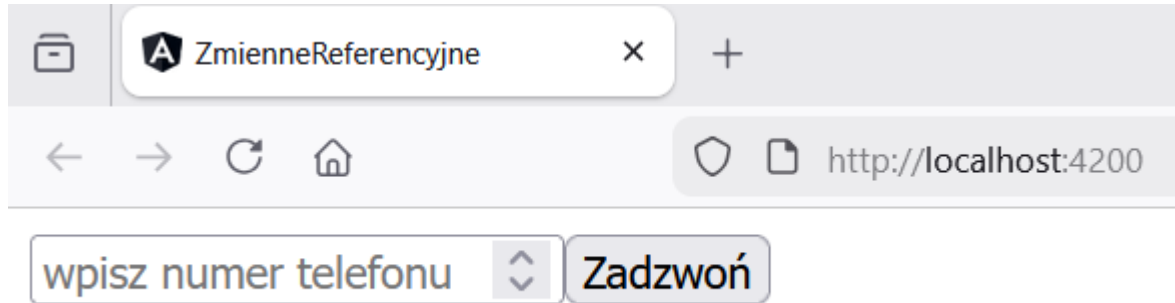
Nadaje nazwę lokalną elementowi (np. `#box`)

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*

```
1  
2 <input type = "number" #nrTelefonu placeholder="wpisz numer telefonu" />  
3 <button type="button" (click)="telefon(nrTelefonu.value)">Zadzwoń</button>  
4  
5 <p>wpisany nr telefonu: {{nrTelefonu.value}}</p>  
6  
7  
8  
9
```

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

zmiennej referencyjnej *nrTelefonu* możemy używać w interpolacji

nrTelefonu:

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

Template reference variables

(zmienne referencyjne w szablonie)

```
EXPLORER
OPEN EDITORS
  X TS app.ts zmienne-referencyjne... M
ZMIENNERREFERENCYJNE
  zmienne-referencyjne
    .angular
    .vscode
    node_modules
    public
    src
      app
        TS app.config.ts
        # app.css
        <> app.html M
        TS app.routes.ts
        TS app.spec.ts
        TS app.ts M

TS app.ts M X
zmienne-referencyjne > src > app > TS app.ts > ...
You, 1 second ago | 1 author (You)
1 import { Component, signal } from '@angular/core';
2
You, 1 second ago | 1 author (You)
3 @Component({
4   selector: 'app-root',
5   imports: [],
6   templateUrl: './app.html',
7   styleUrls: ['./app.css']
8 })
9 export class App {
10   telefon(nrTel: string){
11     alert("dzwonię do " + nrTel)
12   }
13
14 }
15
16
```

metoda *telefon*, której parametrem jest wartość wpisana w input

wiązanie właściwości (property binding)

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

app.html

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

app.ts

wiązanie właściwości
(property) *src* z polem
imageUrl

```
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>
```

app.html

błąd

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

app.html

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

Angular ver. 20

wiązanie właściwości (property binding)

Galeria Pienińska



wiązanie właściwości (property binding)

app.ts

zdjęcia w katalogu **public**

```
8  })
9  export class App {
10     // signal - holds a value (like a state variable),
11     // notifies dependents when the value changes
12     // (powiadamia wartości zależne o zmianie)
13     protected readonly galleryTitle = signal("Galeria Pienińska");
14     zdj2: string = "pictures/zdj2.jpg";
15     zdj3: string = "pictures/zdj3.jpg";
16     zdj4: string = "pictures/zdj4.jpg";
17     aktywuj: boolean = true;
18     wysrodkuj: string = "center";
19     kolor: string = "green";
20 }
21
22
23
```

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

wiązanie właściwości (property 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

The screenshot shows the VS Code editor with the Explorer on the left and the Editor on the right. The Explorer shows the project structure for 'GALERIA_PIENINSKA', including a 'src' folder with an 'app' subfolder containing 'app.html'. The Editor displays the content of 'app.html' with the following HTML code:

```
1 <div id="container">
2   <h1 [style.text-align]="wysrodkuj"
3     [style.color]="kolor">
4     {{galleryTitle()}}
5   </h1>
6   
7   <img src={{zdj2}} [class.image]="aktywuj">
8   <img [src]="zdj3" [class.image]="aktywuj">
9   <img [src]="zdj4" [class.image]="aktywuj">
10 </div>
```

Annotations in the image explain the code:

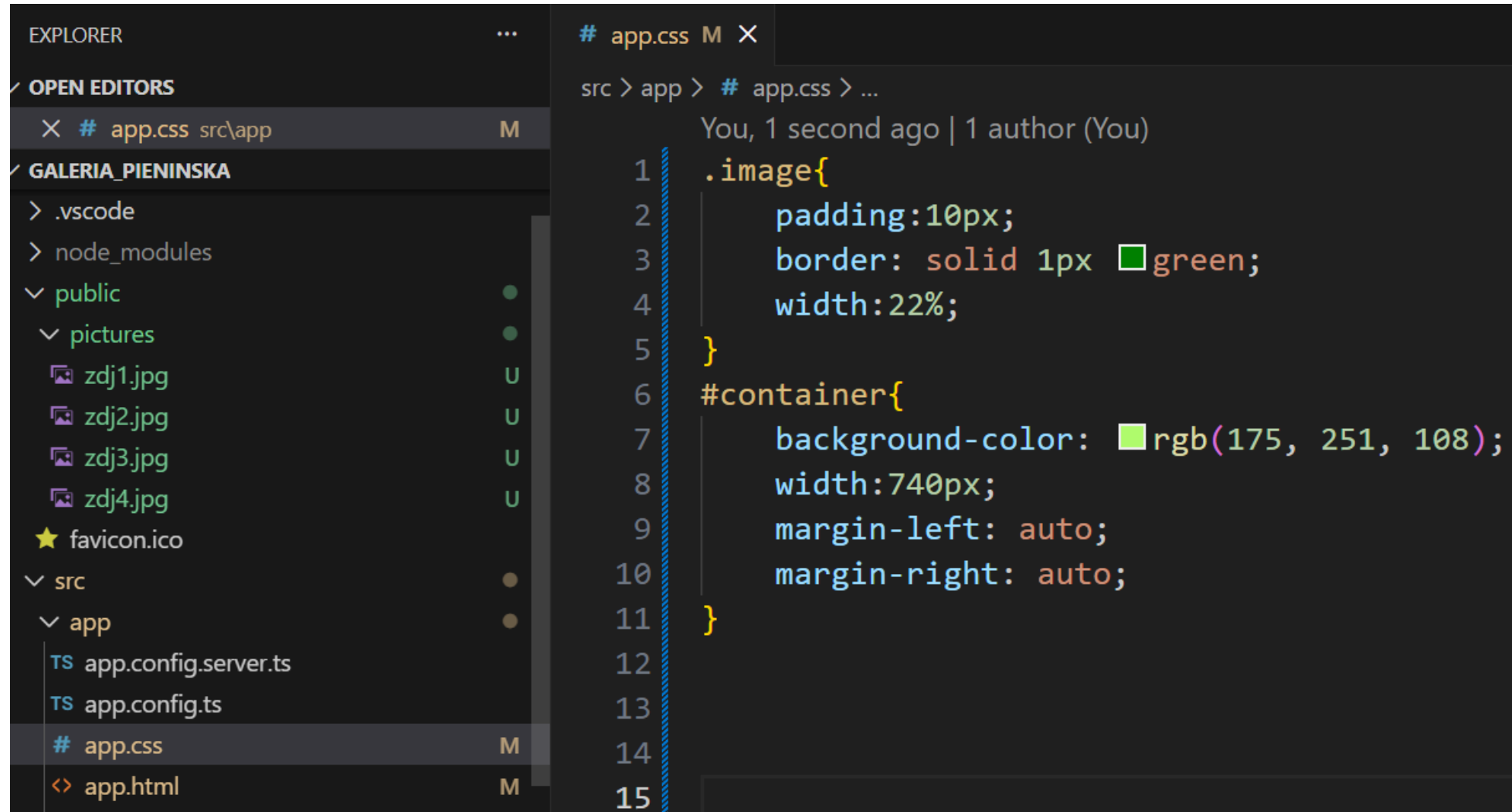
- wiązanie właściwości**: Points to the binding expressions like `[style.text-align]`, `[style.color]`, and `[class.image]`.
- przypisanie stylu**: Points to the `style` attribute on the `<h1>` element.
- poła w klasie app**: Points to the variables `wysrodkuj`, `kolor`, `zdj2`, and `zdj4`.
- klasa *image***: Points to the `class.image` attribute on the `` elements.
- aktywacja klasy *image***: Points to the `aktywuj` variable used in the `[class.image]` binding.

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

Angular ver.20

wiązanie właściwości (property binding)

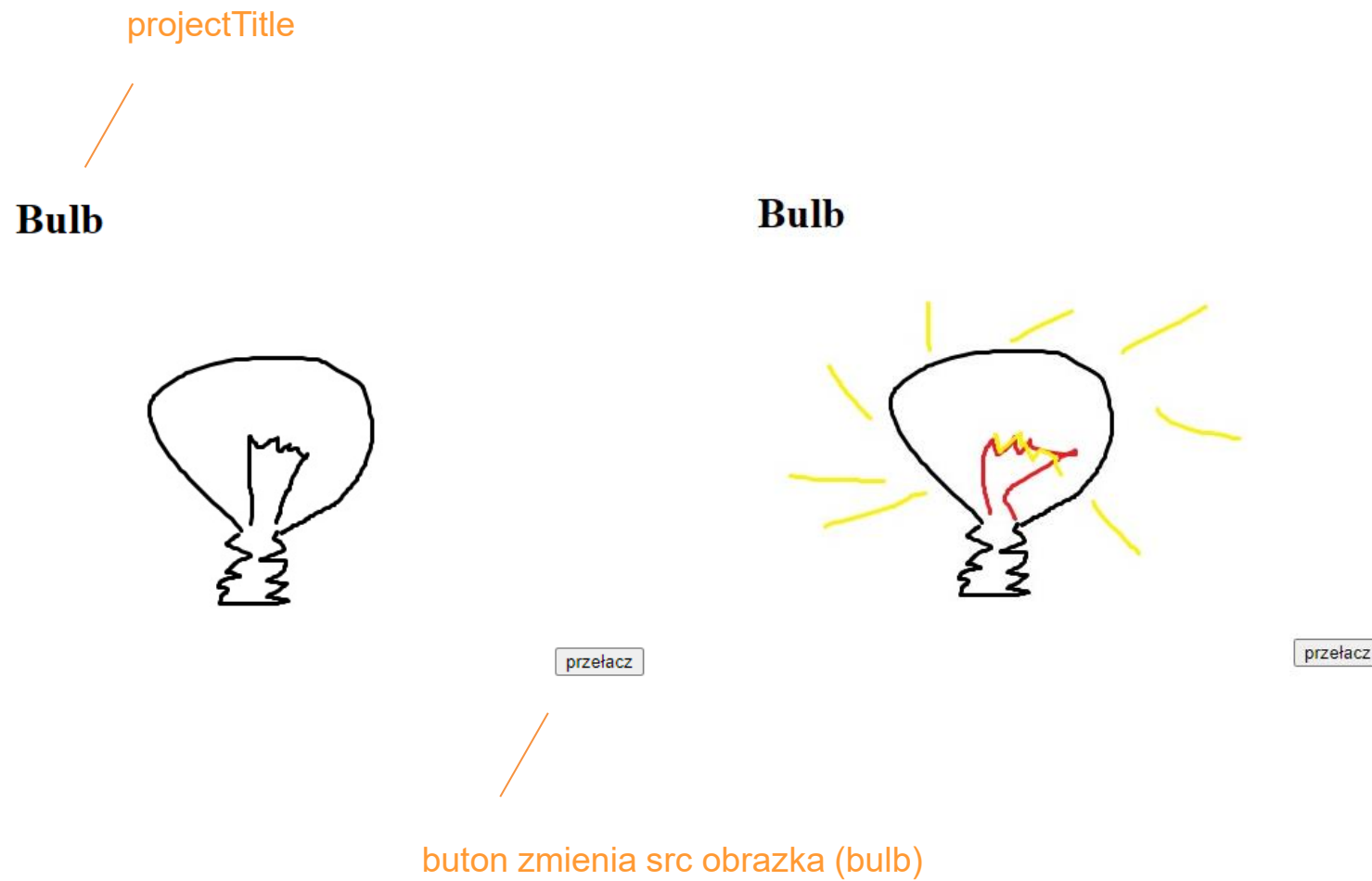
app.css



```
EXPLORER
OPEN EDITORS
  # app.css src\app M
GALERIA_PIENINSKA
  .vscode
  node_modules
  public
    pictures
      zdj1.jpg U
      zdj2.jpg U
      zdj3.jpg U
      zdj4.jpg U
    favicon.ico
  src
    app
      app.config.server.ts
      app.config.ts
      # app.css M
      <> app.html M

# app.css M X
src > app > # app.css > ...
You, 1 second ago | 1 author (You)
1  .image{
2    padding:10px;
3    border: solid 1px green;
4    width:22%;
5  }
6  #container{
7    background-color: rgb(175, 251, 108);
8    width:740px;
9    margin-left: auto;
10   margin-right: auto;
11 }
12
13
14
15
```

wiązanie zdarzeń (event binding)



wiązanie zdarzeń (event binding)

```
<> app.component.html M X
bulbEvent > src > app > <> app.component.html > ...
1 | <h1>{{projectTitle}}</h1>
2 | <img [src]="bulb">
3 | <button (click)="pstryk()">przełącz</button>
```

interpolacja tytułu strony

związanie właściwości src
obrazka do pola bulb

```
TS app.component.ts M X
bulbEvent > 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 |   projectName: 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

wiązanie dwukierunkowe (two-way binding)

zmiany w szablonie
są odzwierciedlone
w kodzie i na odwrót

Zagadka

$2 * 3 = 23$

to co wpisujemy w pole input pokazuje
się jako wynik

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

wiązanie dwukierunkowe (two-way binding)

Zagadka

$$2 * 3 = 23$$

nok

rezultat sprawdzenia
wyniku

przycisk sprawdzający wynik

Zagadka

$$2 * 3 = 6$$

ok

zawartość formularza jest
wpisywana do pola iloczyn

zawartość pola iloczyn
jest wpisywana do wyniku

wiązanie dwukierunkowe (two-way binding)

interpolacja sygnału title

interpolacja pola iloczyn

```
EXPLOLERER
OPEN EDITORS
X app.html zagadka\src\app M
ZAGADKA
  zagadka
    .angular
    .vscode
    node_modules
    public
    src
      app
        app.config.ts
        app.css
        app.html M
        app.routes.ts
        app.spec.ts
        app.ts M

app.html
zagadka > src > app > app.html > ...
Go to component | You, 2 minutes ago | 1 author (You)
1 <h1>{{title()}}</h1>
2 <h2>2 * 3 = {{iloczyn}}</h2>
3 <input type="text" [(ngModel)] = "iloczyn">
4 <button (click)="check()">check</button>
5 <p>{{rezultat}}</p>
6
7 Click to add a breakpoint
```

wiązanie dwukierunkowe pola iloczyn, dzięki niemu gdy zmieni się input (np. użytkownik wpisze liczbę) w kodzie ta zmiana jest odwzorowana w polu iloczyn

pole rezultat w szablonie

wiązanie dwukierunkowe (two-way binding)

```
1 import { Component, signal } from '@angular/core';
2 import { FormsModule } from '@angular/forms';
3
4 @Component({
5   selector: 'app-root',
6   imports: [FormsModule],
7   templateUrl: './app.html',
8   styleUrls: ['./app.css']
9 })
10 export class App {
11   protected readonly title = signal('zagadka');
12   iloczyn: string="";
13   rezultat: string="";
14
15   check(){
16     if(this.iloczyn=="6"){
17       this.rezultat="ok";
18     }else{
19       this.rezultat="nok";
20     }
21   }
22 }
```

import modułu odpowiedzialnego za obsługę formularza

pole iloczyn

pole rezultat

metoda sprawdzająca wartość pola iloczyn i wypisująca rezultat

wiązanie dwukierunkowe (two-way binding)

rozwiązanie zagadki

Zagadka

$7 + 9 = 16$

ok



Zagadka

$6 - 8 = -2$

ok

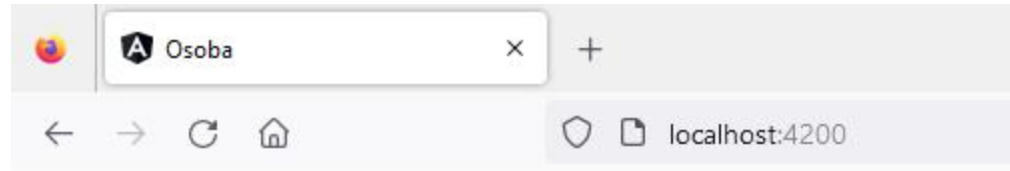


Zagadka

$5 * 4 =$

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

Sluchacz



Kwestionariusz osobowy sluchacza



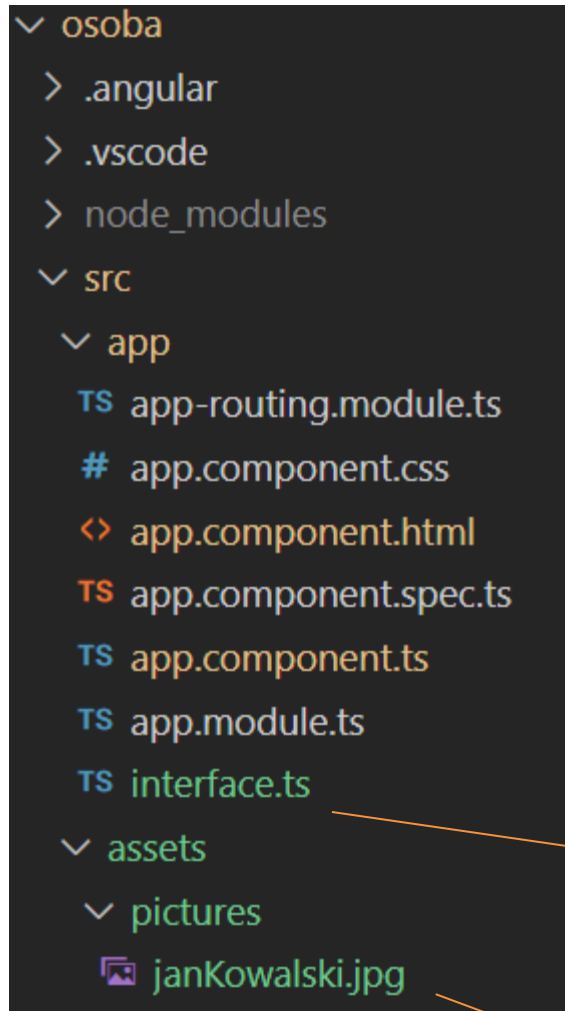
dane sluchacza
pobieramy z obiektu
sluchacz

imię: Jan

nazwisko: Kowalski

kurs kwalifikacyjny: technik programista INF.04

interface.ts



```
TS interface.ts U X
osoba > src > app > TS interface.ts > ...
1  export interface Sluchacz{
2      imie: string;
3      nazwisko: string;
4      kursKwalifikacyjny: string;
5      zdjecie: string;
6  }
```

w katalogu assets
umieszczamy zdjęcie
słuchacza

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

app.component.ts

```
TS app.component.ts M X
osoba > src > app > TS app.component.ts > ...
 1  import { Component } from '@angular/core';
 2  import { Sluchacz } from './interface'
 3
 4  @Component({
 5    selector: 'app-root',
 6    templateUrl: './app.component.html',
 7    styleUrls: ['./app.component.css']
 8  })
 9  export class AppComponent {
10    sluchacz: Sluchacz = {
11      imie: "Jan",
12      nazwisko: "Kowalski",
13      kursKwalifikacyjny: "technik programista INF.04",
14      zdjecie: "assets/pictures/janKowalski.jpg"
15    }
16
17  }
```

import interfejsu
Sluchacz

obiekt sluchacz jest stworzony na podstawie
zaimportowanego interfejsu

app.component.html

```
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

Dyrektywy



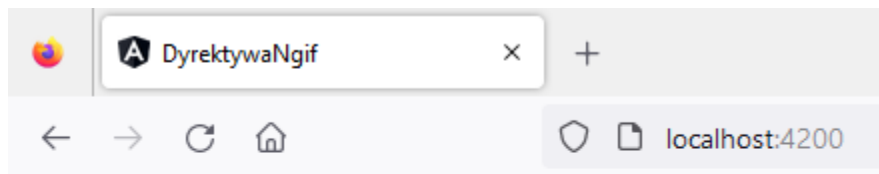
umożliwiają dostęp do struktury dokumentu, mogą zmieniać wygląd i zachowanie danego elementu strony

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

Dyrektywa strukturalna ngIf

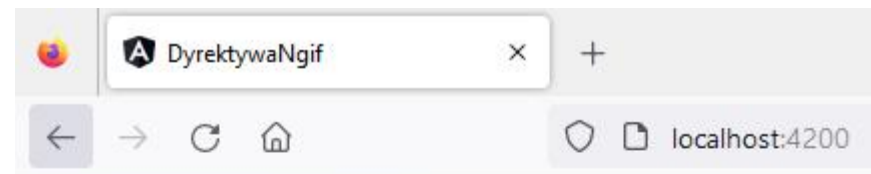
służy do dodawania lub usuwania elementów HTML na stronie (jeśli ma wartość true – element jest uwzględniony w drzewie dokumentów, jeśli false – nie jest uwzględniany).

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 ngIf

```
TS app.component.ts M X
dyrektywaNgif > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    template: `
6      
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 ngIf jest przypisana wartość pola showImage

pole showImage jest zainicjowane wartością false

metoda toggle() (trigowana przez przycisk) przełącza wartość dyrektywy z false na true i na odwrot

Dyrektywa strukturalna ngIf

```
TS app.component.ts M X
dyrektywaNgIf > src > app > TS app.component.ts > ...
1 | import { CommonModule } from '@angular/common';
2 | import { Component } from '@angular/core';
3 |
4 | @Component({
5 |   selector: 'app-root',
6 |   standalone: true,
7 |   imports: [CommonModule],
8 |   template: `
9 |     <img src= "pieniny.jpg" *ngIf="showImage">
10 |     <br>
11 |     <button (click) = "toggle()">toggle</button>
12 |   `,
13 |   styleUrls: ['./app.component.css']
14 | })
15 | export class AppComponent {
16 |   showImage: boolean = false;
17 |   toggle(){
18 |     this.showImage = !this.showImage;
19 |   }
20 | }
```

obrazek
umieszczamy
w katalogu
public

Angular ver.18

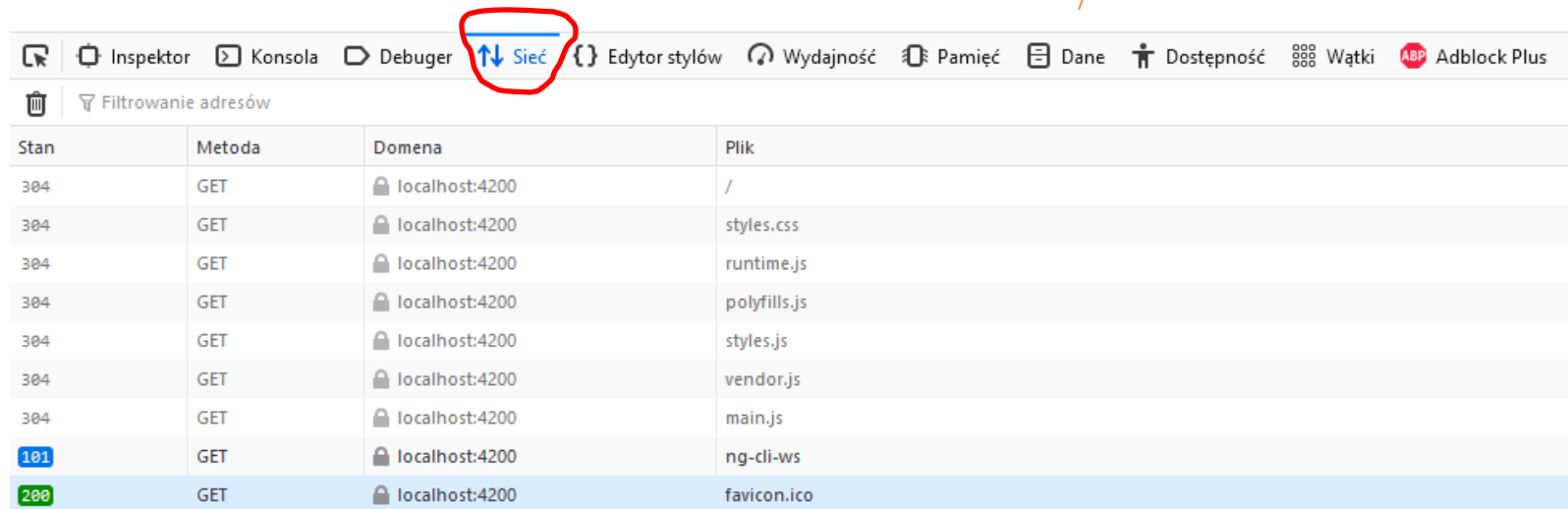
sposób działania nglf



toggle

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

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
304	GET	localhost:4200	pieniny.jpg

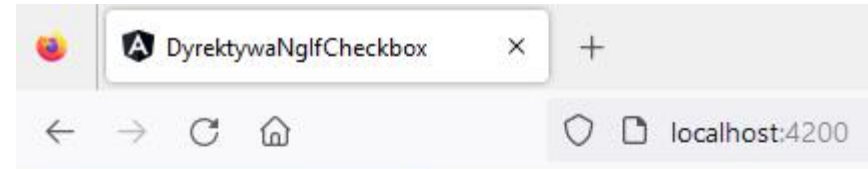
10 żądań | Przesłano: 87,01 kB / 129 B | 1,20 min | DOMContentLoaded

Dyrektywa strukturalna nglf z checkbox



show image

zamiast przycisku checkbox



show image



Dyrektywa strukturalna ngIf z checkbox

```
TS app.component.ts M X
dyrektywaNgIfCheckbox > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    template: `
6      
7      <br>
8      <!--<button (click)="toggle()">toggle</button>-->
9      <label><input type="checkbox" [(ngModel)] = showImage > show image </label>
10     `
11    ,
12    styleUrls: ['./app.component.css']
13  })
14  export class AppComponent {
15    showImage: boolean = false;
16    //toggle(){
17    //  this.showImage = !this.showImage;
18    //}
19  }
```

checkbox powiązany z polem showImage dyrektywą ngModel

Dyrektywa strukturalna nglf z checkbox

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

import modułu odpowiedzialnego za obsługę formularza

Dyrektywa strukturalna ngIf z checkbox

```
<> app.component.html X
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

Dyrektywa strukturalna ngIf z checkbox

TS app.component.ts X

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

```
1 import { CommonModule } from '@angular/common';
2 import { Component } from '@angular/core';
3 import { FormsModule } from '@angular/forms';
4 import { RouterOutlet } from '@angular/router';
5
6 @Component({
7   selector: 'app-root',
8   standalone: true,
9   imports: [RouterOutlet, FormsModule, CommonModule],
10  templateUrl: './app.component.html',
11  styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   title = 'dyrektywaNgIfCheckbox';
15   obrazek: string = "kot.jpg";
16   showImage: boolean = false;
17 }
```

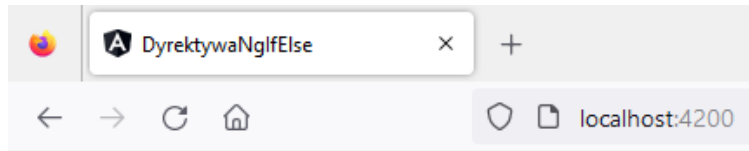
obsługa dyrektywy
*ngif

obsługa formularza
(checkbox)

kot.jpg znajduje się w
katalogu *public*

Angular ver.18

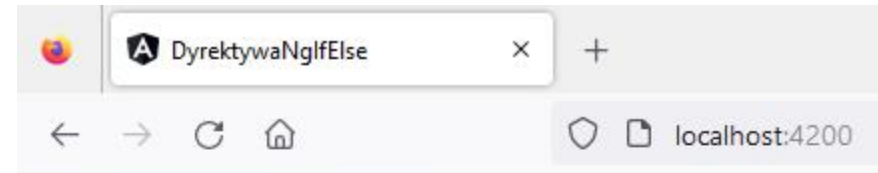
Dyrektywa strukturalna ngIf z else



Nic nie wyświetlam

toggle

efekt działania else przy dyrektywie ngIf



toggle

Dyrektywa strukturalna ngIf z else

```
TS app.component.ts M X
dyrektywaNgIfElse > src > app > TS app.component.ts > AppComponent
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    template: `
6      
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 ngIf jest else, który zamiast zdjęcia wyświetla zawartość znacznika <ng-template #showNothing>

Dyrektywa strukturalna ngIf 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">
```

Dyrektywa strukturalna ngIf z else

```
TS app.component.ts ×
dyrektywaNgIfCheckbox > src > app > TS app.component.ts > ...
1  import { CommonModule } from '@angular/common';
2  import { Component } from '@angular/core';
3  import { FormsModule } from '@angular/forms';
4  import { RouterOutlet } from '@angular/router';
5
6  @Component({
7    selector: 'app-root',
8    standalone: true,
9    imports: [RouterOutlet, FormsModule, CommonModule],
10   templateUrl: './app.component.html',
11   styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   title = 'dyrektywaNgIfCheckbox';
15   obrazek: string = "kot.jpg";
16   showImage: boolean = false;
17   wyswietla_sie_zamiast_obrazka: string = "KOCUREK";
18 }
```

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++

```
TS app.component.ts M X
templateSyntaxIfElse > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2  import { RouterOutlet } from '@angular/router';
3
4  @Component({
5    selector: 'app-root',
6    standalone: true,
7    imports: [RouterOutlet],
8    template: `
9      @if (showImage){
10         
11       }@else {
12         <p>{{information}}</p>
13       }
14     <br>
15     <button (click)="toggle()">toggle</button>
16   `
17   ,
18   styleUrls: ['./app.component.css']
19 })
19 export class AppComponent {
20   title = 'templateSyntaxIfElse';
21   showImage: boolean = false;
22   information: string = "Nic nie wyświetlam";
23   toggle(){
24     this.showImage = !this.showImage;
25   }
26 }
```

pieniny.jpg znajduje się w katalogu *public*

@if template syntax

korzystając z instrukcji warunkowej @if
zaprojektuj formularz do logowania

login

password

zaloguj

login

password

zaloguj

Zły login lub hasło

login

password

zaloguj

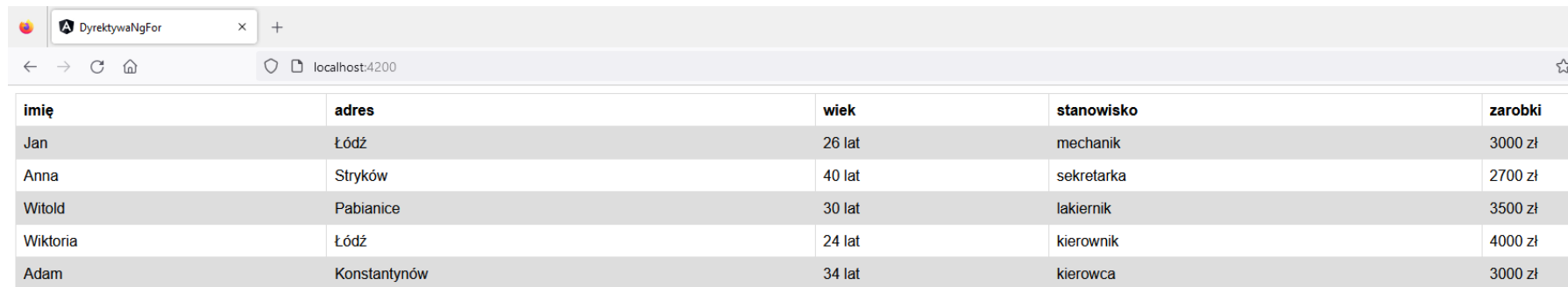
Zalogowało mnie!!!

Dyrektywa strukturalna ngFor

zmienia strukturę drzewa DOM poprzez iterowanie przez tablicę obiektów

*ngFor = "let element of kolekcja"

składnia dyrektywy



The screenshot shows a web browser window with the title "DyrektywaNgFor" and the address bar displaying "localhost:4200". The main content of the browser is a table with five columns: "imię", "adres", "wiek", "stanowisko", and "zarobki". The table contains five rows of data representing employees.

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

```
▼ dyrektywaNgFor
  > .angular
  > .vscode
  > node_modules
  ▼ src
    ▼ app
      # app.component.css
      <> app.component.html
      TS app.component.spec.ts
      TS app.component.ts
      TS app.module.ts
      TS interface.ts
    > assets
    > environments
```

```
TS interface.ts U X
dyrektywaNgFor > src > app > TS interface.ts > ...
1   export interface Pracownik{
2       imie: string;
3       adres:string;
4       wiek: number;
5       stanowisko:string;
6       zarobki:number;
7   }
```

interjejs Pracownik

nowy plik (nazwa jest nasza)

Dyrektywa strukturalna ngFor

Angular ver. 8 – trzeba zaimportować
CommonModule

```
TS app.component.ts M X
dyrektywaNgFor > src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2 import { Pracownik } from './interface';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css']
8 })
9 export class AppComponent {
10   tableHeader: string[] = ['imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
11   pracownicy: Pracownik[] = [
12     {imię: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
13     {imię: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
14     {imię: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
15     {imię: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
16     {imię: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
17   ];
18 }
```

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 X
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 tabeli

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

dyrektywaNgFor_galeria



zdjecia/01.jpg

zdjecia/02.jpg

zdjecia/03.jpg

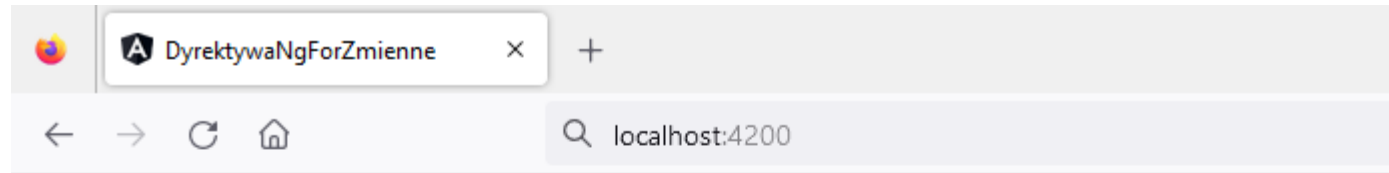
zdjecia/04.jpg

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



nr	imię	adres	wiek	stanowisko	zarobki
1/5	Jan	Łódź	26 lat	mechanik	3000 zł
2/5	Anna	Stryków	40 lat	sekretarka	2700 zł
3/5	Witold	Pabianice	30 lat	lakiernik	3500 zł
4/5	Wiktoria	Łódź	24 lat	kierownik	4000 zł
5/5	Adam	Konstantynów	34 lat	kierowca	3000 zł

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

Dyrektywa strukturalna ngFor, zmienne

```
<> app.component.html M X
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

Dyrektywa strukturalna ngFor, zmienne

```
TS app.component.ts M X
dyrektywaNgForZmienne > src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2 import { Pracownik } from './interface';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css']
8 })
9 export class AppComponent {
10   // nagłówek tabeli
11   tableHeader: string[] = ['nr', 'imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
12   // tabela z pracownikami
13   pracownicy: Pracownik[] = [
14     {imie: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
15     {imie: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
16     {imie: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
17     {imie: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
18     {imie: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
19   ];
20 }
```

doszła nowa kolumna na indeks

Dyrektywa strukturalna ngFor, zmienne

```
# app.component.css M X
dyrektywaNgForZmienne > src > app > # app.component.css >
1  .evenRow{
2    background-color: DodgerBlue;
3  }
4  .oddRow{
5    background-color: Tomato;
6  }
7  .firstRow{
8    color: red;
9  }
10 .lastRow{
11  color: yellow;
12 }
13 .countRow{
14  text-align:right;
15 }
```

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

```
TS interface.ts U X
templateSyntaxFor > src > app > TS interface.ts
1 export interface Pracownik{
2   id: number;
3   imie: string;
4   adres:string;
5   wiek: number;
6   stanowisko:string;
7   zarobki:number;
8 }
```

dodajemy indeks do interfejsu

@for template syntax

```
TS app.component.ts M X
templateSyntaxFor > src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import {Pracownik } from './interface';
4 //import { CommonModule } from '@angular/common';
5
6 @Component({
7   selector: 'app-root',
8   standalone: true,
9   imports: [RouterOutlet],
10  templateUrl: './app.component.html',
11  styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   title = 'templateSyntaxFor';
15   tableHeader: string[] = ['id', 'imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
16   pracownicy: Pracownik[] = [
17     {id: 0, imie: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
18     {id: 1, imie: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
19     {id: 2, imie: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
20     {id: 3, imie: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
21     {id: 4, imie: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
22   ];
23
24   isVisible:boolean = false;
25   toggle(){
26     this.isVisible = !this.isVisible;
27   }
28 }
```

nie jest potrzebny CommonModule

można dodać kolumnę identyfikacyjną

@for template syntax

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

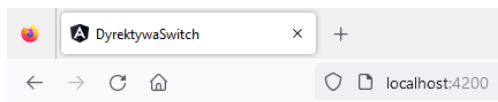
```
<> app.component.html M X
templateSyntaxFor > src > app > <> app.component.html > ...
1 <table>
2   @if(isVisible){
3     <tr>
4       @for(header of tableHeader; track $index){
5         <th> {{ header }}</th>
6       }
7     </tr>
8   }@else{
9     <tr>
10      <th>else</th>
11      <th>else</th>
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>
```

indeks bieżącego elementu w kolekcji

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

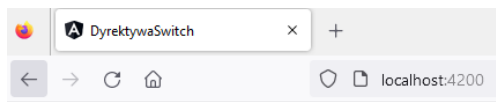
Dyrektywa strukturalna ngSwitch

przełącznik



2

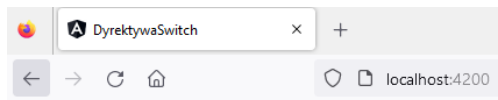
dwa



1

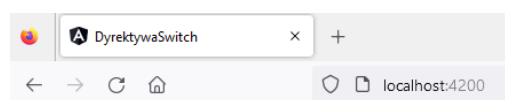
jeden

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 X
dyrektywaSwitch > 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    // liczba losowa z zakresu od 1 do 6
10   random: number = Math.floor(Math.random()*6+1);
11 }
```

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 X
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.  
  }  
}
```

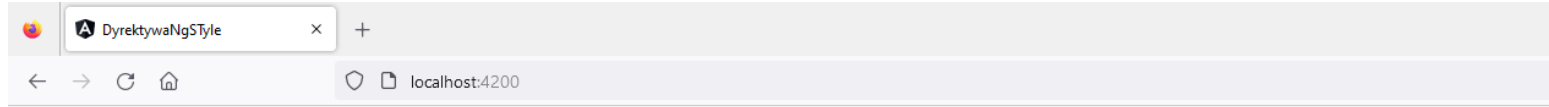
<https://angular.dev/api/core/@switch>

@switch template syntax

```
<> app.component.html M X ↺↻  
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| }
```

Dyrektywa atrybutowa ngStyle

zmiana stylu elementu



napis zmieniający swój styl

zmiana stylu



napis zmieniający swój styl

zmiana stylu

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

Dyrektywa atrybutowa ngStyle

TS app.component.ts M X

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

```
1 | import { NgStyle } from '@angular/common';
2 | import { Component } from '@angular/core';
3 | import { RouterOutlet } from '@angular/router';
4 |
5 | @Component({
6 |   selector: 'app-root',
7 |   standalone: true,
8 |   imports: [RouterOutlet, NgStyle],
9 |   templateUrl: './app.component.html',
10 |  styleUrls: ['./app.component.css']
11 | })
12 | export class AppComponent {
13 |   title = 'dyrektywaNgStyle';
14 |   index: number = 0;
15 |   colors: string[] = ["red", "yellow", "green", "blue"];
16 |   fonts: string[]=["Times New Roman", "Arial","Lucida Handwriting",
17 |   | | | | | | | "Courier New"];
18 |   increment(){
19 |     // 0 - 3
20 |     this.index = Math.floor(Math.random() * 4);
21 |     console.log(this.index);
22 |   }
23 | }
```

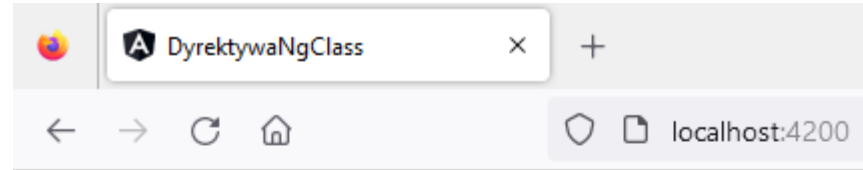
kolory i czcionki wybieramy z tablic

indeksem w tablicy jest liczba losowa
zmieniana przyciskiem

Angular ver.18

Dyrektywa atrybutowa ngClass

ustawia klasę elementu



zakupy:

- chleb
- jabłka
- mąka
- ziemniaki

klasa listy jest ustawiona za pomocą dyrektywy ngStyle

Dyrektywa atrybutowa ngClass

style

```
# app.component.css M X
dyrektywaNgClass > src > app > # app.component.c
1  .red{
2  |   color: red;
3  | }
4  .blue{
5  |   background-color: blue;
6  | }
7  .green{
8  |   border: 10px solid green
9  | }
```

```
TS app.component.ts M X
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 X
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 samochodu (aplikacja z komponentami typu no-standalone)

```
ng g c samochody
```

polecenie utworzenia komponentu samochodu:

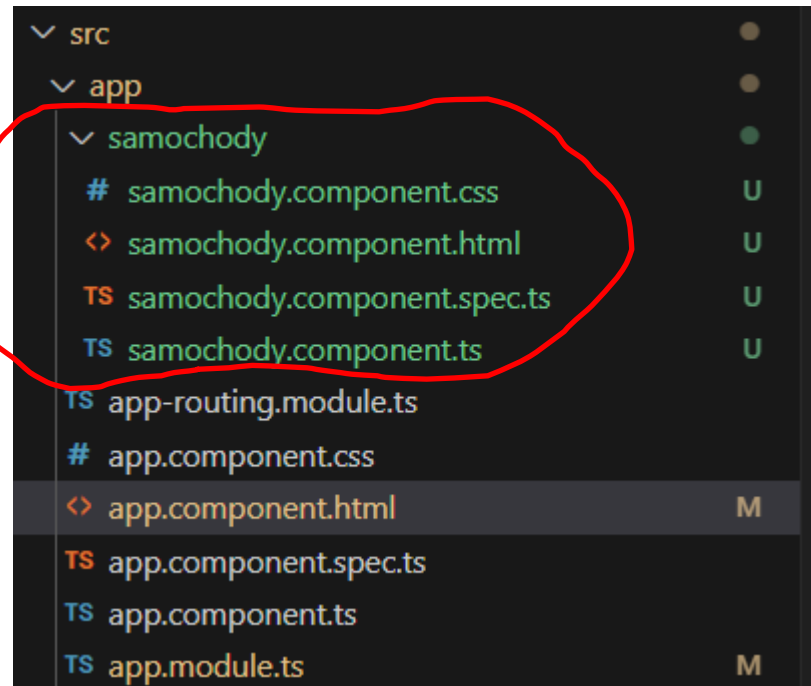
g: generate

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 (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)



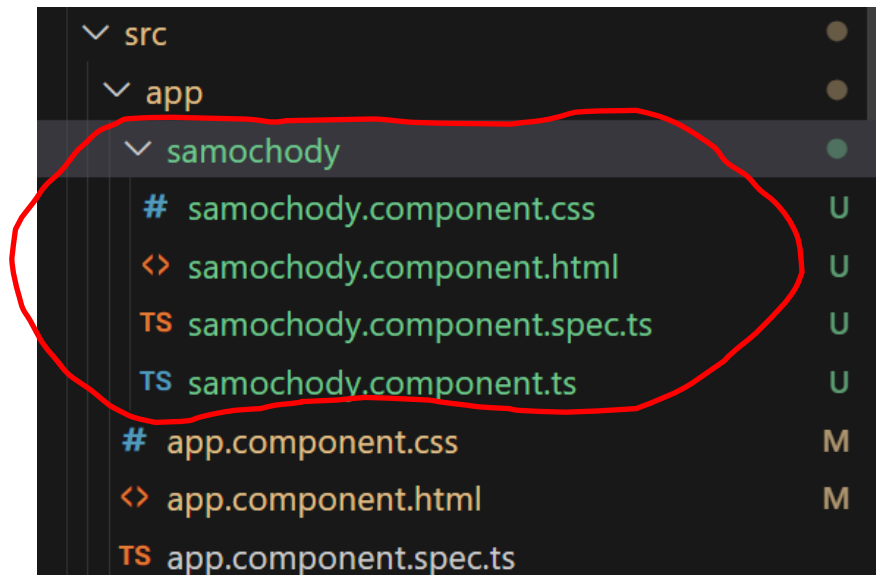
wygenerowane pliki

utworzenie nowego komponentu samochodu (aplikacja z komponentami typu standalone)

```
ng g c samochody
```

polecenie utworzenia komponentu samochodu:
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 typu standalone)

utworzenie nowego komponentu samochodu (aplikacja z komponentami typu standalone)

```
TS app.component.ts M X
_tmp > tmp > 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  import { SamochodyComponent } from './samochody/samochody.component';
5
6  @Component({
7    selector: 'app-root',
8    standalone: true,
9    imports: [CommonModule, RouterOutlet, SamochodyComponent],
10   templateUrl: './app.component.html',
11   styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   title = 'tmp';
15 }
```

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

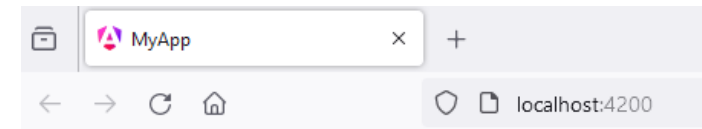
Angular ver.18

utworzenie nowego komponentu samochodu (aplikacja z komponentami typu standalone)

```
TS samochody.component.ts U X
_tmp > tmp > src > app > samochody > TS samochody.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-samochody',
5    standalone: true,
6    imports: [],
7    templateUrl: './samochody.component.html',
8    styleUrls: ['./samochody.component.css']
9  })
10 export class SamochodyComponent {
11   auta: string = "tu pojawią się najnowsze samochody";
12 }
```

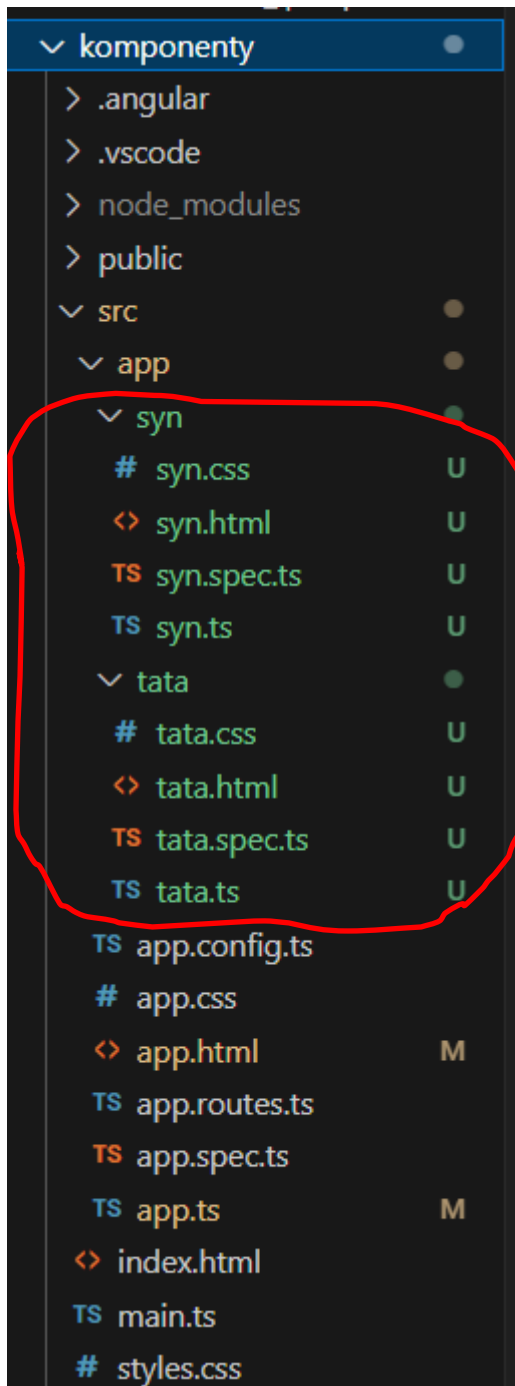
```
<> samochody.component.html U X
_tmp > tmp > src > app > samochody > <> samochody.component.html >
  Go to component
1  <p>{{auta}}</p>
```

```
<> app.component.html M X
_tmp > tmp > src > app > <> app.component.html > ...
  Go to component
1  <app-samochody></app-samochody>
```



tu pojawią się najnowsze 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

```
TS app.ts 1, M X
komponenty > src > app > TS app.ts > ...
...
1 import { Component, signal } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { Tata } from './tata/tata';
4
...
5 @Component({
6   selector: 'app-root',
7   imports: [RouterOutlet, Tata],
8   templateUrl: './app.html',
9   styleUrls: ['./app.css']
10 })
11 export class App {
12   protected readonly title = signal('komponenty');
13 }
```

w głównym komponencie app importujemy component taty

```
TS tata.ts U X
komponenty > src > app > tata > TS 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   styleUrls: ['./tata.css'],
9 })
10 export class Tata {
11   biezaceInfo:string = "To jest info od Taty";
12 }
```

komponencie taty importujemy component syna

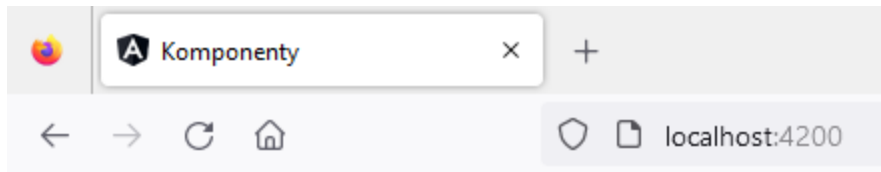
Komponenty – przesyłanie danych

```
<> app.html M X
komponenty > src > app > <> app.html > ...
Go to component | You, 1 second ago | 1 author (You)
1 <app-tata></app-tata>
```

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

```
<> tata.html U X
komponenty > src > app > tata > <> tata.html > ...
Go to component
1 <p>tata works!</p>
2 <app-syn></app-syn>
```

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



tata works!

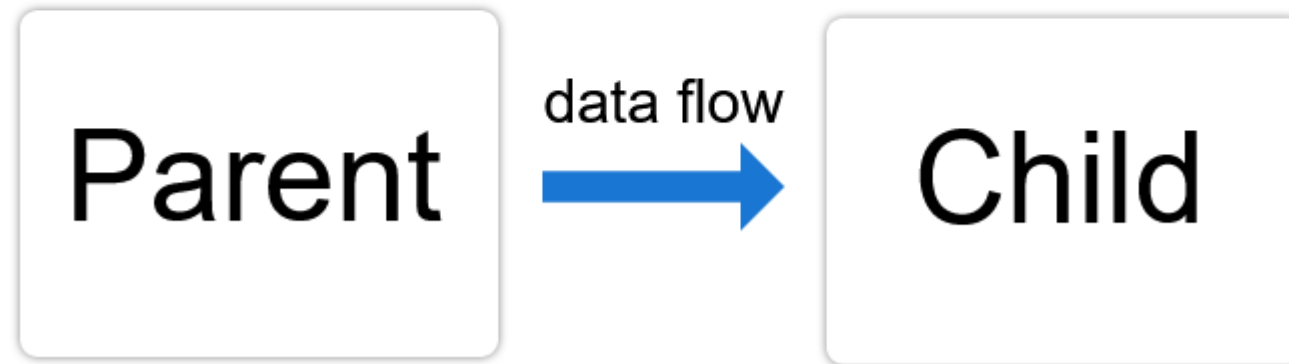
syn works!

komponenty tata i syn (oraz główny) działają

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

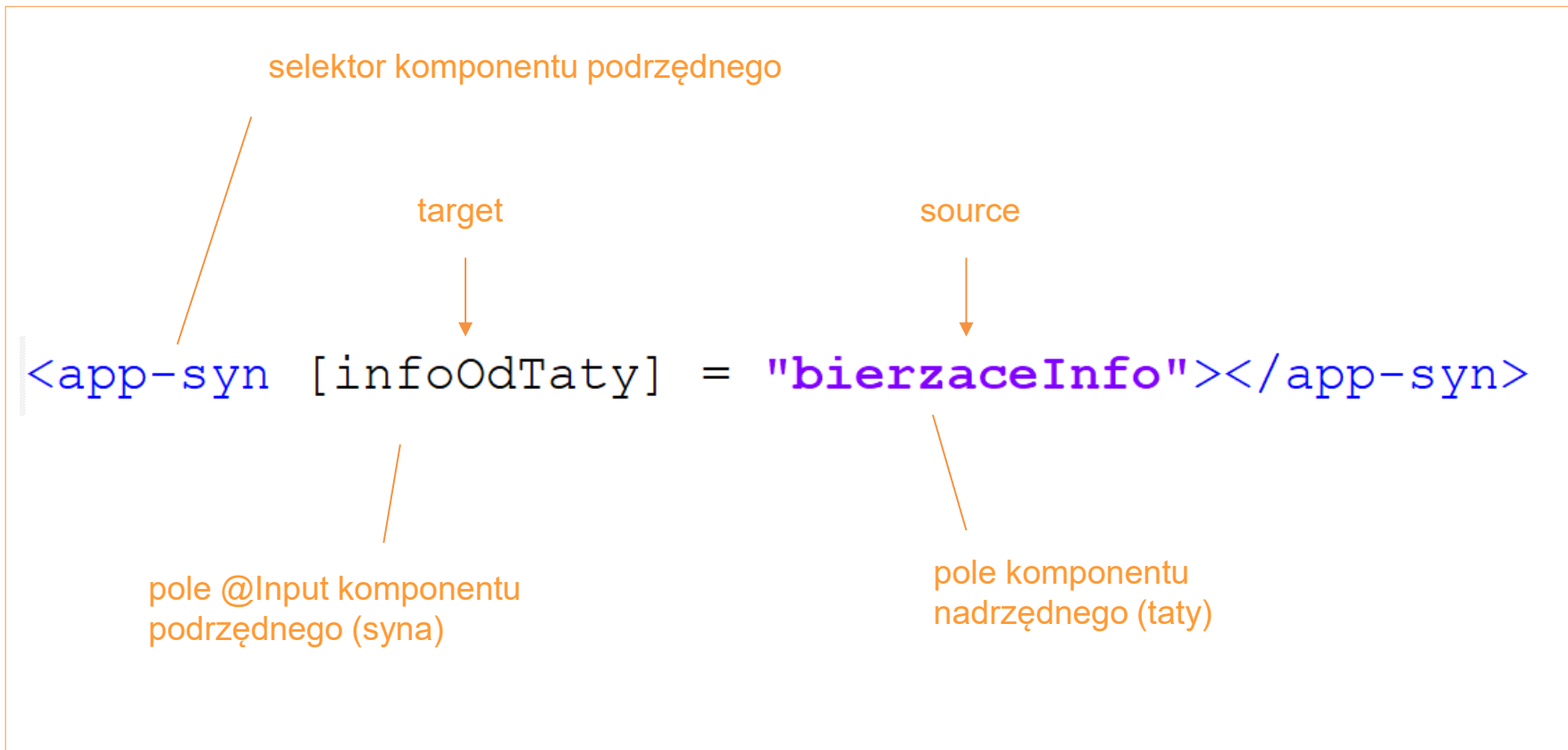
@Input



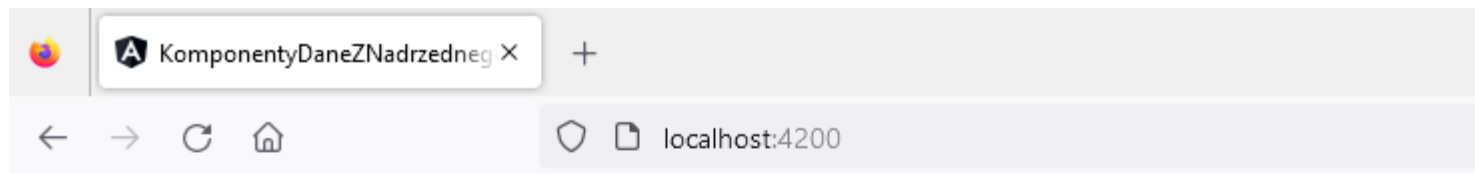
Komponenty – przesyłanie danych z komponentu nadrzędnego (tata) do podrzędnego (syn)

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

tata.component.html



Komponenty – przesyłanie danych z komponentu nadrzędnego (tata) do podrzędnego (syn)



tata works!

syn works!

Info od taty:

Najważniejsze co się czuje, słuchaj zawsze głosu serca, hej

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)

```
TS syn.ts U X
komponenty > src > app > syn > TS syn.ts > ...
1  import { Component, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-syn',
5    imports: [],
6    templateUrl: './syn.html',
7    styleUrls: ['./syn.css'],
8  })
9  export class Syn {
10   @Input() infoOdTaty: string = "";
11 }
```

import komponentu Input

u syna tworzymy pole infoOdTaty z dekoratorem @Input()

```
<> syn.html U X
komponenty > src > app > syn > <> syn.html > ...
Go to component
1  <p>syn works!</p>
2  <p>{{infoOdTaty}}</p>
```

Angular ver.20

w pliku html syna posługujemy się interpolacją

Komponenty – przesyłanie danych z komponentu nadrzędnego (tata) do podrzędnego (syn)

2.konfiguracja komponentu nadrzędnego (tata)

```
TS tata.ts U X
komponenty > src > app > tata > TS 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    styleUrls: ['./tata.css'],
9  })
10 export class Tata {
11   biezaceInfo:string = "Najważniejsze co się czuje, słuchaj zawsze głosu swego serca, hej";
12 }
```

u taty tworzymy pole biezaceInfo

```
<> tata.html U X
komponenty > src > app > tata > <> tata.html > ...
Go to component
1  <p>tata works!</p>
2  <app-syn [infoOdTaty]="biezaceInfo"></app-syn>
3
```

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

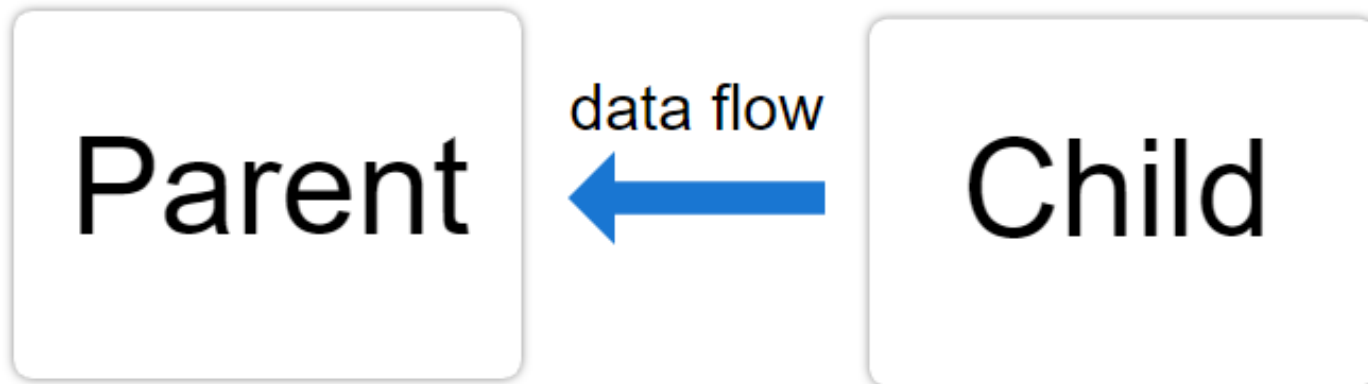
Angular ver.20

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

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

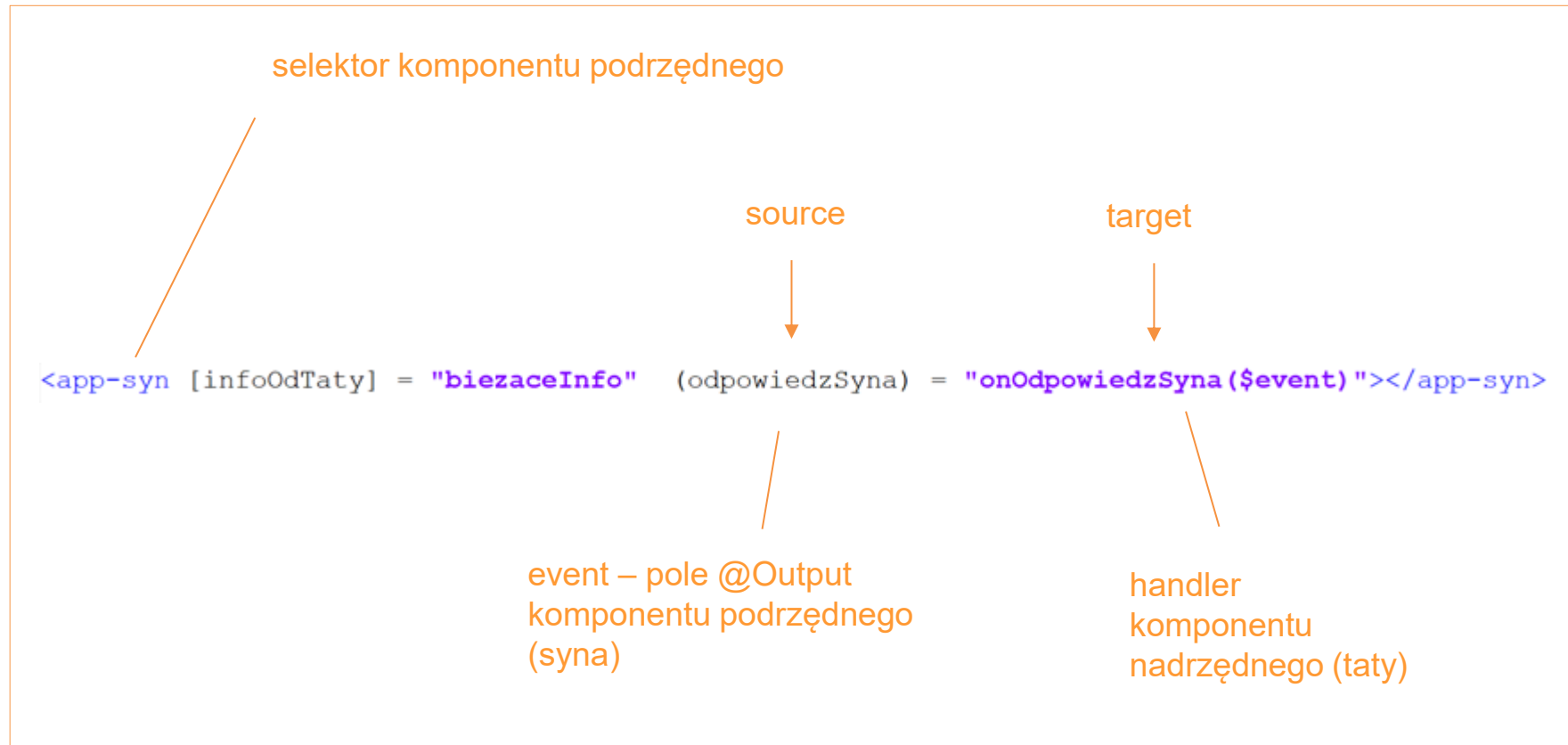
@Output



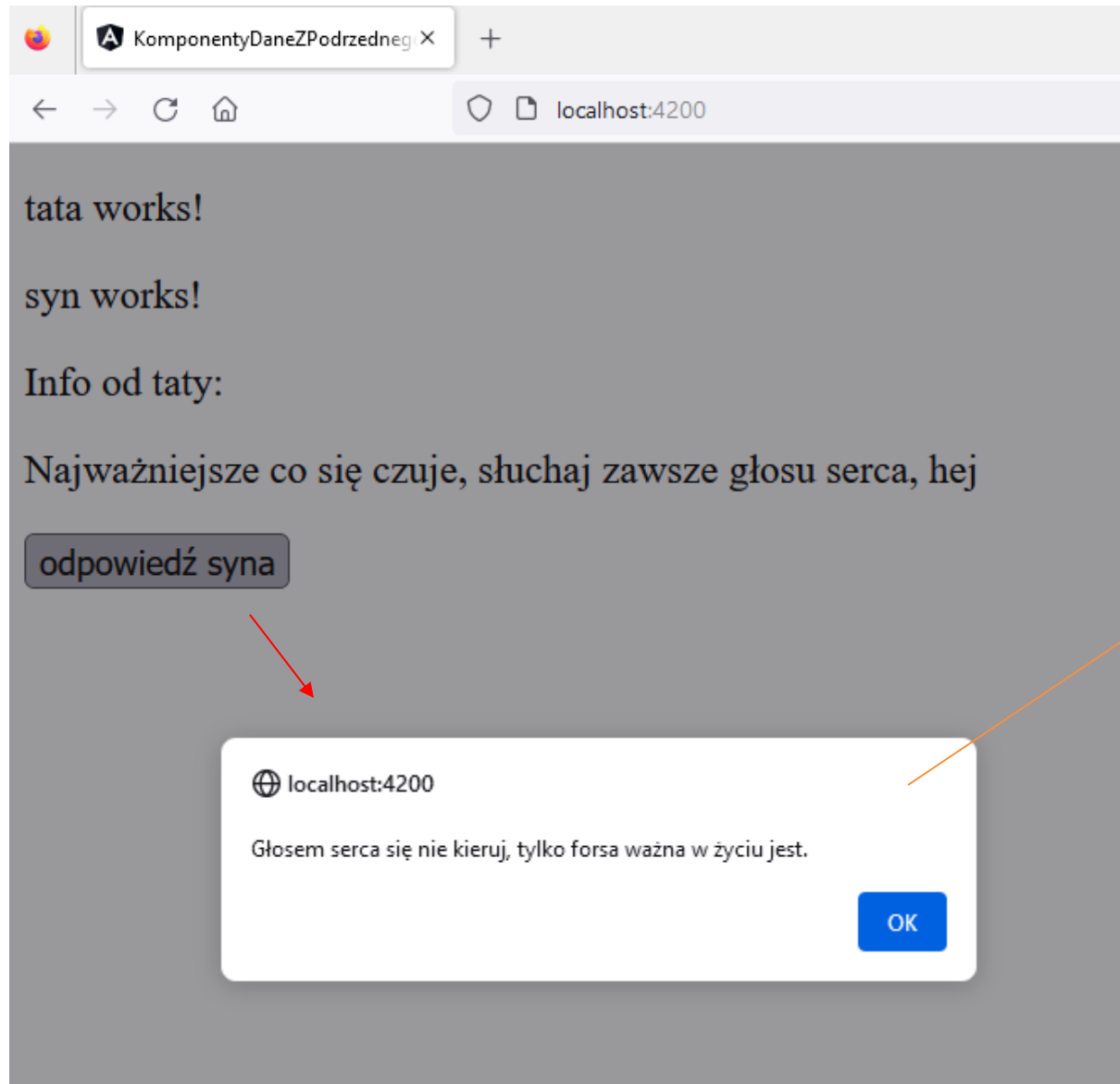
Komponenty – przesyłanie danych z komponentu podrzędnego (syn) do nadrzędnego (tata)

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

tata.component.html



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)

TS syn.ts U X

1.konfiguracja komponentu podrzędnego (syn)

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

```
1 import { Component, Input, Output, EventEmitter } from '@angular/core';
2
3 @Component({
4   selector: 'app-syn',
5   imports: [],
6   templateUrl: './syn.html',
7   styleUrls: ['./syn.css'],
8 })
9 export class Syn {
10  @Input() infoOdTaty: string = "";
11  @Output() odpowiedzSyna = new EventEmitter<string>();
12
13  odpowiedz(){
14    this.odpowiedzSyna.emit('Głosem serca się nie kieruj, tylko forsa ważna w życiu jest!');
15  }
16 }
```

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

będziemy eksportować
dane typu string

u syna tworzymy pole odpowiedzSyna
(zdarzenie, obiekt typu EventEmitter) 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)

```
<> syn.html U X
komponenty > src > app > syn > <> syn.html > button
Go to component
1 <p>syn works!</p>
2 <p>{{infoOdTaty}}</p>
3 <button type="button" (click)="odpowiedz()">odpowieź syna</button>
```

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 X
komponenty > src > app > tata > tata.html > ...
Go to component
1 <p>tata works!</p>
2 <app-syn [infoOdTaty]="biezaceInfo" (odpowiedzSyna)="onOdpowiedzSyna($event)"></app-syn>
```

```
TS tata.ts U X
komponenty > src > app > tata > TS 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   styleUrls: ['./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 }
```

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

definicja metody (handlera) `onOdpowiedzSyna($event)`,

Angular ver.20

Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu z komponentu App do Syn

komponnet Syn

Hello z komponentu App

wyslij z App

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

komponnet App

Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

data-service.ts

z komponentu App do Syn

```
ng g s DataService
```

tworzymy serwis DataService

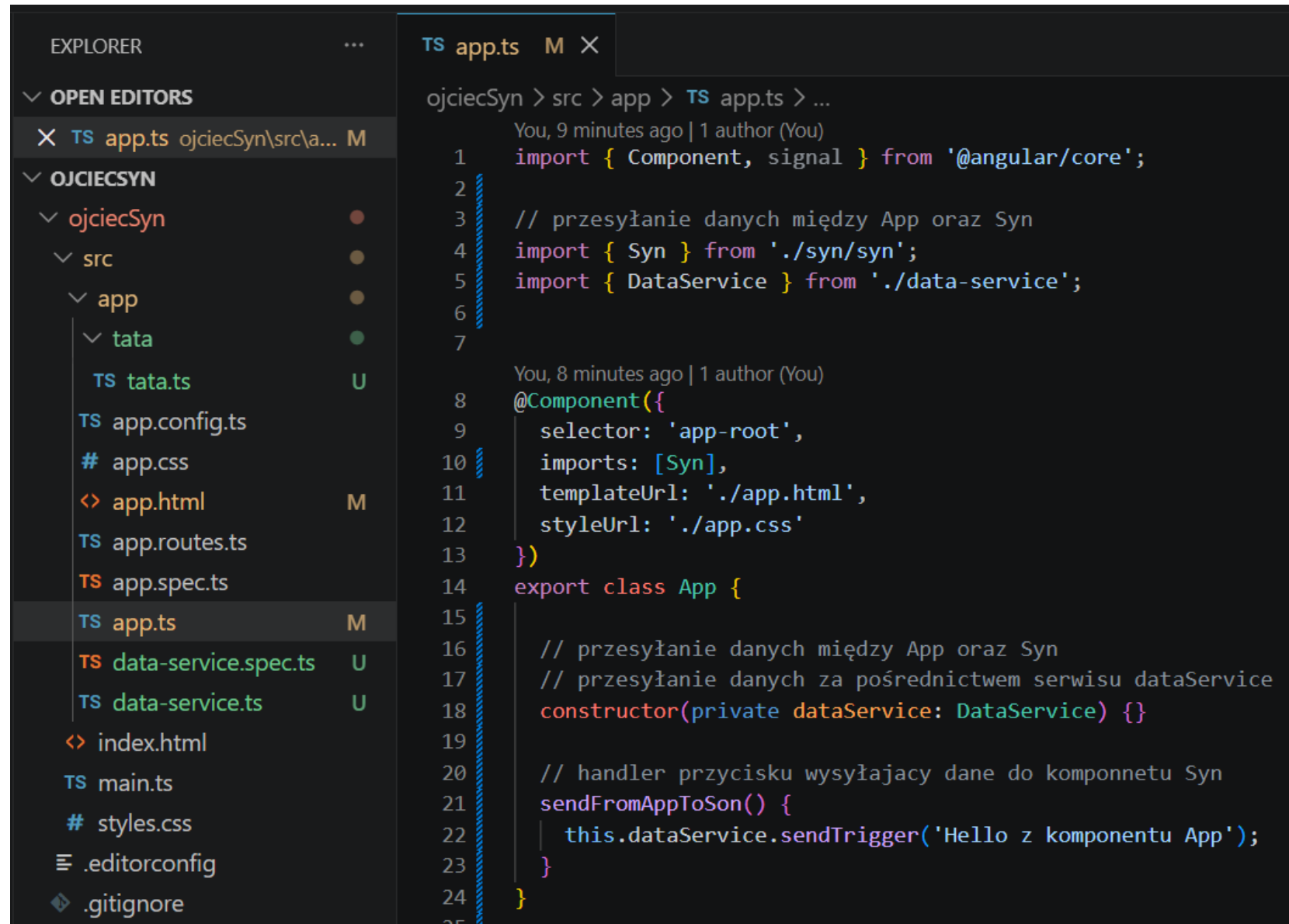
```
EXPLORER
  OPEN EDITORS
    TS data-service.ts ojciec... U
  OJCIECYSYN
    ojciecSyn
      src
        app
          syn
            syn.html U
            TS syn.spec.ts U
            TS syn.ts U
          tata
            # tata.css U
            <> tata.html U
            TS tata.spec.ts U
            TS tata.ts U
          app.config.ts
          # app.css
          <> app.html M
          TS app.routes.ts
          TS app.spec.ts
          TS app.ts 1, M
          TS data-service.spec.ts U
          TS data-service.ts U
        index.html
        TS main.ts

TS data-service.ts U X
ojciecSyn > src > app > TS data-service.ts > ...
1 // przesyłanie danych między App oraz Syn
2
3 import { Injectable } from '@angular/core';
4 import { Subject } from 'rxjs';
5
6
7 // Angular tworzy jedną wspólną instancję serwisu dla całej aplikacji
8 // dzięki temu komponenty App oraz Syn używają tego samego obiektu
9 @Injectable({
10   providedIn: 'root'
11 })
12 export class DataService {
13   // tworzymy obiekt typu Subject, który wysyła i odbiera dane
14   private triggerSource = new Subject<string>();
15
16   // pole do odczytu danych
17   trigger$ = this.triggerSource.asObservable();
18
19   // metoda wysyłająca wartość value do wszystkich subskrybentów
20   sendTrigger(value: string) {
21     this.triggerSource.next(value);
22   }
23 }
24
25
26
27
```

Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

app.ts

z komponentu App do Syn



```
EXPLORER
  OPEN EDITORS
    TS app.ts ojciecSyn\src\a... M
  OJCIECSYN
    ojciecSyn
      src
        app
          tata
            TS tata.ts U
          TS app.config.ts
          # app.css
          <> app.html M
          TS app.routes.ts
          TS app.spec.ts
          TS app.ts M
          TS data-service.spec.ts U
          TS data-service.ts U
        <> index.html
      TS main.ts
      # styles.css
      .editorconfig
      .gitignore

TS app.ts M X
ojciecSyn > src > app > TS app.ts > ...
You, 9 minutes ago | 1 author (You)
1 import { Component, signal } from '@angular/core';
2
3 // przesyłanie danych między App oraz Syn
4 import { Syn } from './syn/syn';
5 import { DataService } from './data-service';
6
7
You, 8 minutes ago | 1 author (You)
8 @Component({
9   selector: 'app-root',
10  imports: [Syn],
11  templateUrl: './app.html',
12  styleUrls: ['./app.css']
13 })
14 export class App {
15
16   // przesyłanie danych między App oraz Syn
17   // przesyłanie danych za pośrednictwem serwisu dataService
18   constructor(private dataService: DataService) {}
19
20   // handler przycisku wysyłający dane do komponentu Syn
21   sendFromAppToSon() {
22     this.dataService.sendTrigger('Hello z komponentu App');
23   }
24 }
25
```

Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

z komponentu App do Syn

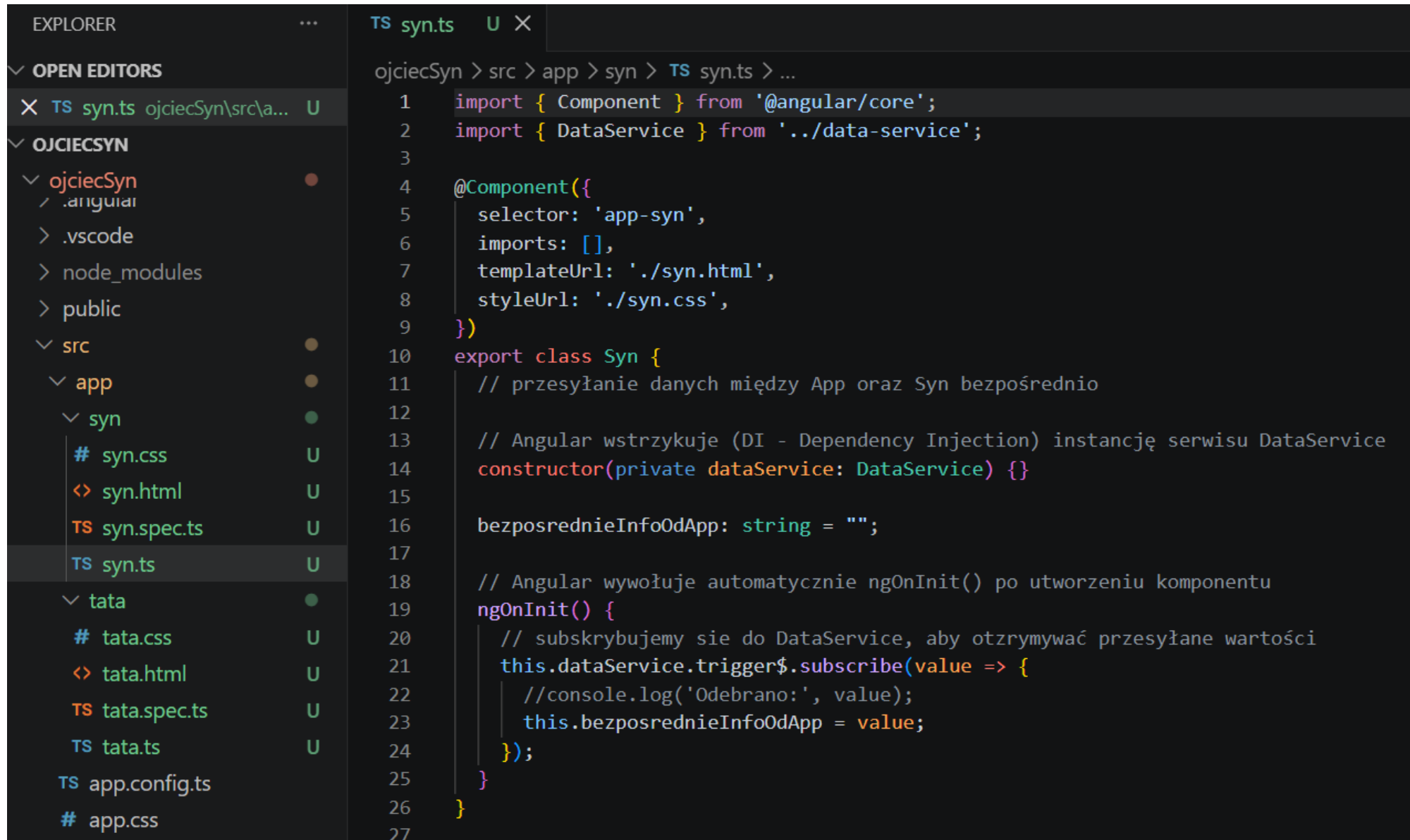
app.html

```
<> app.html M X
ojciecSyn > src > app > <> app.html > ...
Go to component | You, 8 minutes ago | 1 author (You)
1 <!-- przesyłanie danych między App oraz Syn-->
2 <app-syn></app-syn>
3 <button (click)="sendFromAppToSon()">wyslij z App</button>
4
```

Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

syn.ts

z komponentu App do Syn



```
EXPLORER
...
OPEN EDITORS
  TS syn.ts ojciecSyn\src\a... U
OJCIECSYN
  ojciecSyn
    .angular
    .vscode
    node_modules
    public
    src
      app
        syn
          # syn.css U
          <> syn.html U
          TS syn.spec.ts U
          TS syn.ts U
        tata
          # tata.css U
          <> tata.html U
          TS tata.spec.ts U
          TS tata.ts U
      app.config.ts
      # app.css
TS syn.ts U X
ojciecSyn > src > app > syn > TS syn.ts > ...
1  import { Component } from '@angular/core';
2  import { DataService } from '../data-service';
3
4  @Component({
5    selector: 'app-syn',
6    imports: [],
7    templateUrl: './syn.html',
8    styleUrls: ['./syn.css'],
9  })
10 export class Syn {
11   // przesyłanie danych między App oraz Syn bezpośrednio
12
13   // Angular wstrzykuje (DI - Dependency Injection) instancję serwisu DataService
14   constructor(private dataService: DataService) {}
15
16   bezposrednieInfoOdApp: string = "";
17
18   // Angular wywołuje automatycznie ngOnInit() po utworzeniu komponentu
19   ngOnInit() {
20     // subskrybujemy sie do DataService, aby otrzymywać przesyłane wartości
21     this.dataService.trigger$.subscribe(value => {
22       //console.log('Odebrano:', value);
23       this.bezposrednieInfoOdApp = value;
24     });
25   }
26 }
27
```

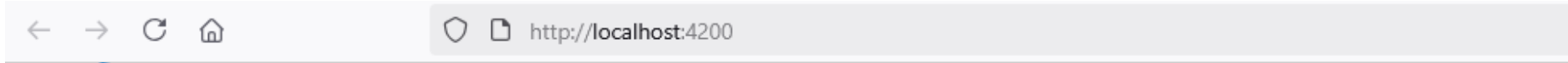
Komponenty – przesyłanie danych między dowolnymi komponentami za pośrednictwem serwisu

z komponentu App do Syn

syn.html

```
<> syn.html U X
ojciecSyn > src > app > syn > <> syn.html > ...
  Go to component
1  <p>{{bezposrednieInfoOdApp}}</p>
2
```

sklep



sklep works!

Sklep

towary

20 ilość

kurier

pomidory

produkt works!


komponent nadrzędny sklep

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

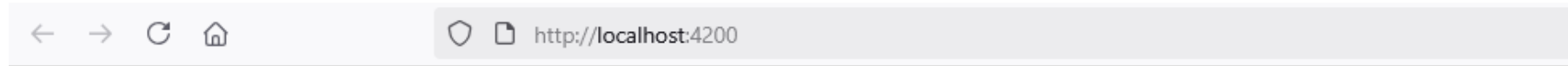
wybór towaru

komponent podrzędny produkt

Wybrany Towar:

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

sklep



sklep works!

Sklep


towary

20 ilość
 kurier
Piotrkowska podaj adres
pomidory

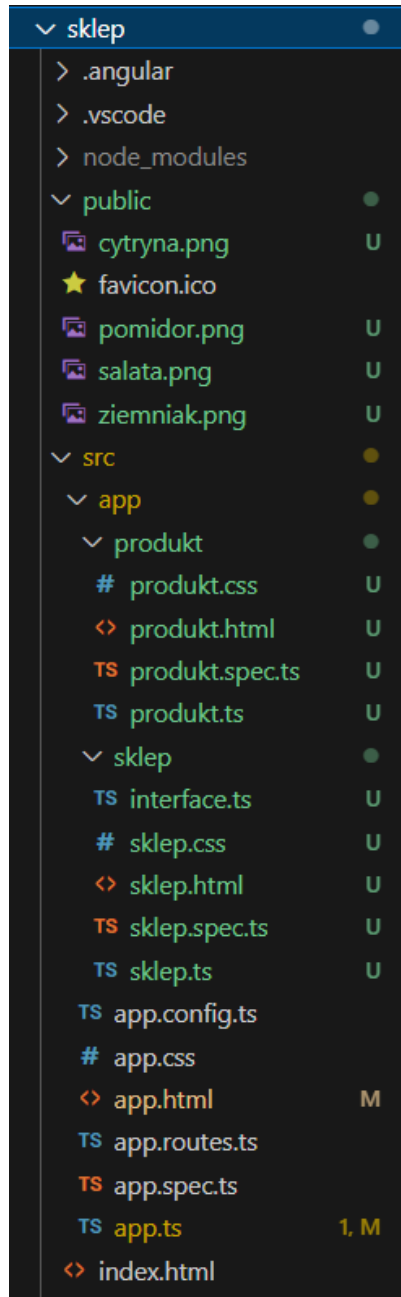
po zaznaczeniu kuriera pojawia się input na wpisanie adresu oraz nowa kolumna z adresem w komponencie produkt

produkt works!

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

Sklep

komponent nadrzędny sklep

```
TS sklep.ts U X
sklep > src > app > sklep > TS sklep.ts > ...
1  import { Component } from '@angular/core';
2  import { FormsModule } from '@angular/forms';
3  import { CommonModule } from '@angular/common';
4  import { Produkt } from '../produkt/produkt';
5  import { Towar } from './interface';
6
7  @Component({
8    selector: 'app-sklep',
9    imports: [FormsModule, CommonModule, Produkt],
10   templateUrl: './sklep.html',
11   styleUrls: ['./sklep.css'],
12 })
13 export class Sklep {
14   towary : Towar[] =[
15     { nazwa: "cytryny", cena: 20, zdjecie: "cytryna.png"},
16     { nazwa: "ziemniaki", cena: 2.2, zdjecie: "ziemniak.png"},
17     { nazwa: "salata", cena: 5.2, zdjecie: "salata.png"},
18     { nazwa: "pomidory", cena: 9, zdjecie: "pomidor.png"}
19   ];
20
21   towar: Towar = this.towary[0];
22   ilosc: number = 0;
23   kurier:boolean = false;
24   adres: string = "";
25 }
```

```
TS interface.ts U X
sklep > src > app > sklep > TS interface.ts > ...
1  export interface Towar {
2    nazwa: string;
3    cena: number;
4    zdjecie:string;
5  }
6
```

Sklep

komponent nadrzędny sklep

```
<> sklep.component.html U X
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 zagnieźdźmy 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>
```

Sklep

komponent nadrzędny sklep

sklep.html U X

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

Go to component

```
1 <p>sklep works!</p>
2 <h1>Sklep</h1>
3 <p>towary</p>
4 <input type="number" [(ngModel)] = "ilosc">ilość<br>
5 <label><input type="checkbox" [(ngModel)] = kurier >kurier</label><br>
6 <label *ngIf = "kurier"><input type="text" [(ngModel)] = "adres">podaj adres</label><br>
7
8 <select [(ngModel)] = "towar">
9   <option *ngFor = "let item of towary" [ngValue]= "item" >{{item.nazwa}}</option>
10
11 </select>
12
13 <app-produkt [wybranyTowar] = "towar" [wybranaIlosc] = "ilosc" [czyKurier] = "kurier" [podanyAdres]="adres"></app-produkt>
14
```

Sklep

komponent podrzędny produkt

```
TS produkt.ts U X
sklep > src > app > produkt > TS produkt.ts > ...
 1  import { Component, Input } from '@angular/core';
 2  import { CommonModule } from '@angular/common';
 3  import { Towar } from '../sklep/interface';
 4
 5  @Component({
 6    selector: 'app-produkt',
 7    imports: [CommonModule],
 8    templateUrl: './produkt.html',
 9    styleUrls: ['./produkt.css'],
10  })
11  export class Produkt {
12
13    // wykrzyknik powoduje, że kompilator nie będzie ostrzegał
14    // o możliwej wartości null lub undefined
15    @Input() wybranyTowar!: Towar;
16
17    @Input() wybranaIlosc!: number;
18
19    @Input() czyKurier!: boolean;
20
21    @Input() podanyAdres!: string;
22  }
```

Sklep

komponent podrzędny produkt

```
<> produkt.html U X
sklep > src > app > produkt > <> produkt.html > ...
  Go to component
1  <p>produkt works!</p>
2  <br><br>
3  <p>Wybrany Towar:</p>
4  <table>
5    <tr>
6      <th>zdjęcie</th>
7      <th>nazwa</th>
8      <th>cena</th>
9      <th>ilość</th>
10     <th>należność</th>
11     <th>kurier</th>
12     <th *ngIf = "czyKurier">adres</th>
13   </tr>
14   <tr>
15     <td><img src={{wybranyTowar.zdjecie}} width="200px"></td>
16     <td>{{wybranyTowar.nazwa}}</td>
17     <td>{{wybranyTowar.cena}} zł</td>
18     <td>{{wybranaIlosc}}</td>
19     <td>{{ wybranyTowar.cena*wybranaIlosc | number}} zł</td>
20     <td *ngIf = "czyKurier else nie">tak</td>
21     <ng-template #nie>
22       <td >nie</td>
23     </ng-template>
24     <td *ngIf = "czyKurier">{{podanyAdres}}</td>
25   </tr>
26 </table>
```

Sklep

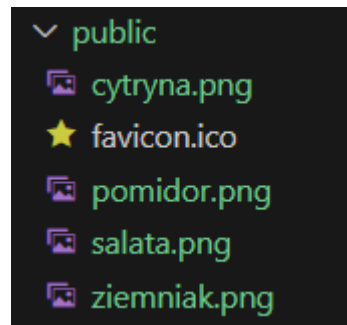
komponent podrzędny produkt

```
# produkt.css U X
sklep > src > app > produkt > # produkt.css > ...
1  table {
2    font-family: arial, sans-serif;
3    border-collapse: collapse;
4    width: 50%;
5  }
6
7  td, th {
8    border: 1px solid #dddddd;
9    text-align: left;
10   padding: 8px;
11  }
12
13  tr:nth-child(even) {
14    background-color: #dddddd;
15  }
16
```

Sklep

komponent główny app

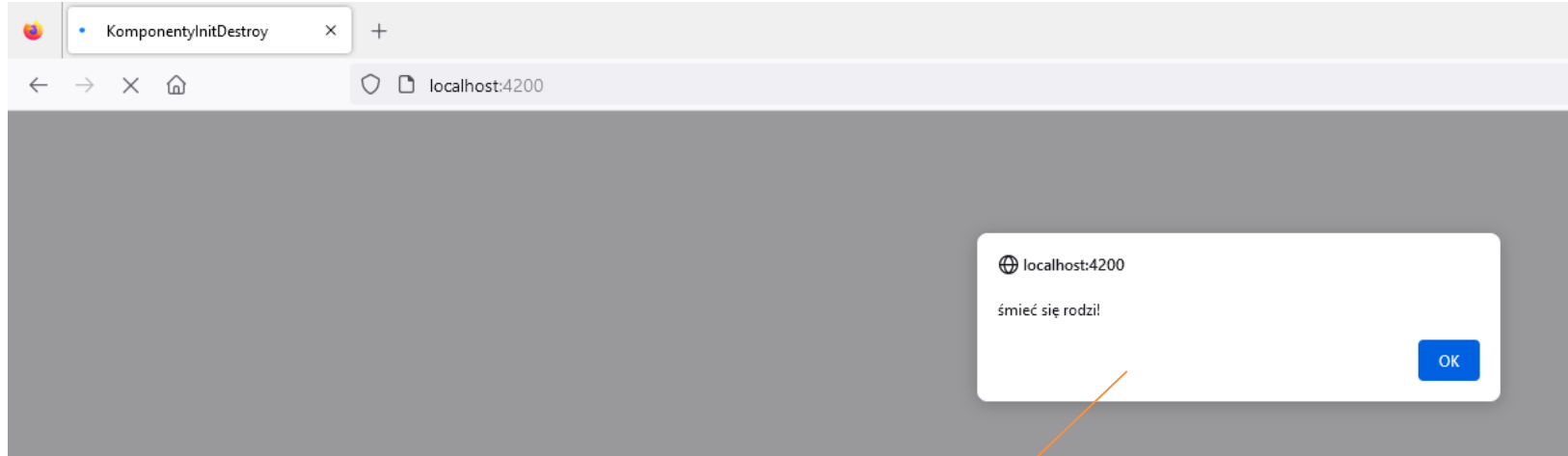
```
<> app.html M X
sklep > src > app > <> app.html > ...
Go to component | You, 1 second ago | 1 author (You)
1 <!-- w komponencie głównym zagnieźdźamy komponent sklep-->
2 <app-sklep></app-sklep>
```



Sklep komponent główny

```
TS app.ts 1, M X
sklep > src > app > TS app.ts > ...
You, 29 minutes ago | 1 author (You)
1 import { Component, signal } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { Sklep } from "../sklep/sklep";
4 You, 29 minutes ago • Uncommitted changes
5
You, 29 minutes ago | 1 author (You)
6 @Component({
7   selector: 'app-root',
8   imports: [RouterOutlet, Sklep],
9   templateUrl: './app.html',
10  styleUrls: ['./app.css']
11 })
12 export class App {
13   protected readonly title = signal('sklep');
14 }
```

Niszczarka śmieci ngOnInit(), ngOnDestroy()



wynik działania metody ngOnInit()
komponentu smiec

Niszczarka śmieci ngOnInit(), ngOnDestroy()



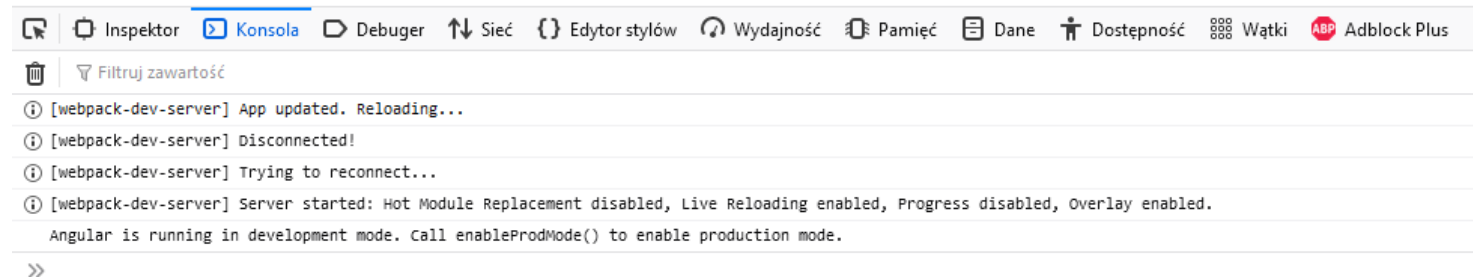
Niszczarka śmieci

odepnij komponent smiec od drzewa dokumentu

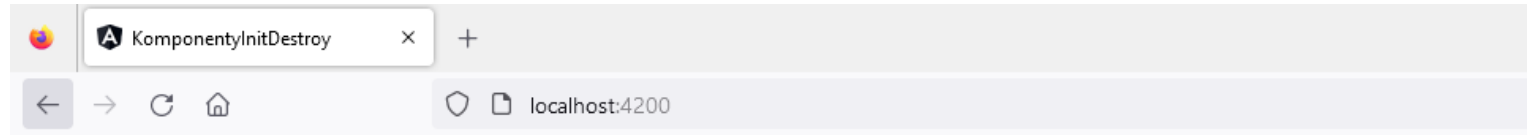
śmieć works!

komponent główny

komponent podrzędny smiec



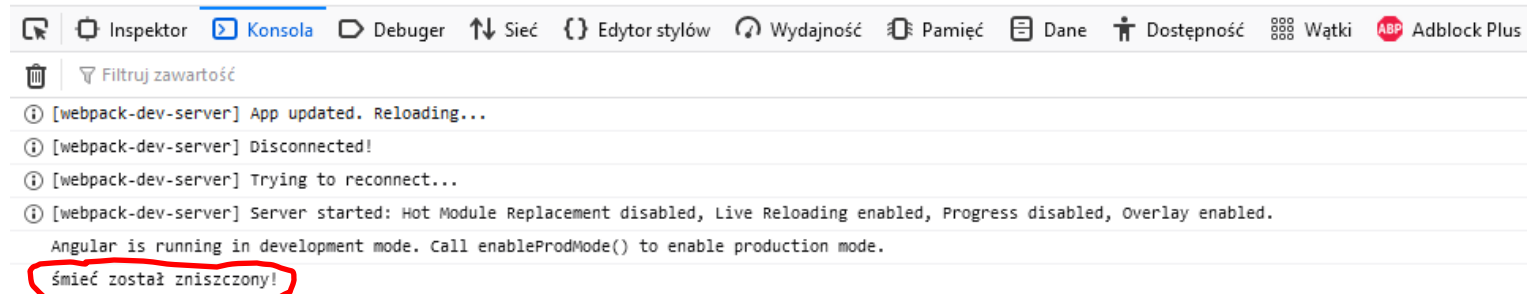
Niszczarka śmieci ngOnInit(), ngOnDestroy()



Niszczarka śmieci

odepnij komponent smiec od drzewa dokumentu

po zaznaczeniu checkboxa
komponent smiec zostaje
odłączony od drzewa
dokumentu



wynik działania metody
ngOnDestroy() komponentu smiec

Niszczarka śmieci komponent smiec

```
<> smiec.component.html U X  
komponentyInicDestroy > src > app > sr  
1 <p>śmieć works!</p>
```

```
TS smiec.component.ts U X  
komponentyInicDestroy > src > app > smiec > TS smiec.component.ts > S  
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()

Niszczarka śmieci komponent główny

```
<> app.component.html M X
komponenty\InitDestroy > 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

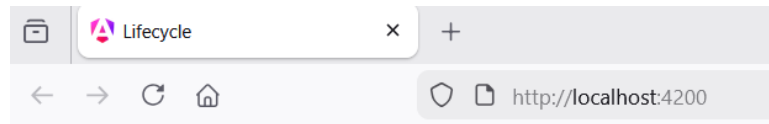
```
TS app.component.ts M X
komponenty\InitDestroy > src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   zniszcz: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

pole zniszcz przyjmuje
wartości true lub false

import FormsModule

```
TS app.module.ts M X
komponenty\InitDestroy > src > app > TS 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 { }
```

Cykl życia komponentu



Lifecycle Demo Component

Wartość inputu: startowa wartość

treść wpisana w komponencie nadrzędnym (app.html)

Element widoku (ViewChild)

Zmień wartość inputu Usun / dodaj komponent

komponent lifecycle-demo

Inspektor Konsola Debugger Sieć Edytor stylów Wyc

Filtruj zawartość

```
GET http://localhost:4200/@ts/v:/arcn/iecnnikprogramista/aplikacjeowe/Angu
GET ws://localhost:4200/?token=DJoPxmGsYjuW
[vite] connected.
GET http://localhost:4200/@ng/component?c=src/app/lifecycle-demo/lifecycle-dem
GET http://localhost:4200/@ng/component?c=src/app/app.ts@App&t=1763743099016
```

Constructor

- ngOnChanges ▶ Object { data: {...} }
- ngOnInit
- ngDoCheck
- ngAfterContentInit
- ngAfterContentChecked
- ngAfterViewInit

Element z ViewChild: ▶ <div style="padding: 10px; background: rgb(70, 70, 253);

- ngAfterViewChecked

Angular is running in development mode.

- ngDoCheck
- ngAfterContentChecked
- ngAfterViewChecked

cykl życia komponentu lifecycle-demo

Cykl życia komponentu

kolejność wywołań *Lifecycle Hooks*:

```
constructor
  ↓
ngOnChanges
  ↓
ngOnInit
  ↓
ngDoCheck —————┐
  ↓                    │
ngAfterContentInit → ngAfterContentChecked (loop)           │
  ↓                    │
ngAfterViewInit →   ngAfterViewChecked (loop)                │
  └──────────────────┘
  ↓
ngOnDestroy
```

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()

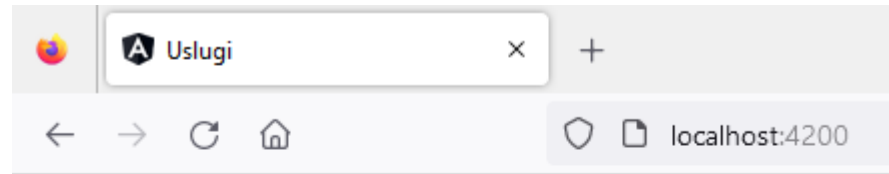
Kiedy wywoływany: tuż przed usunięciem komponentu z DOM.

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)

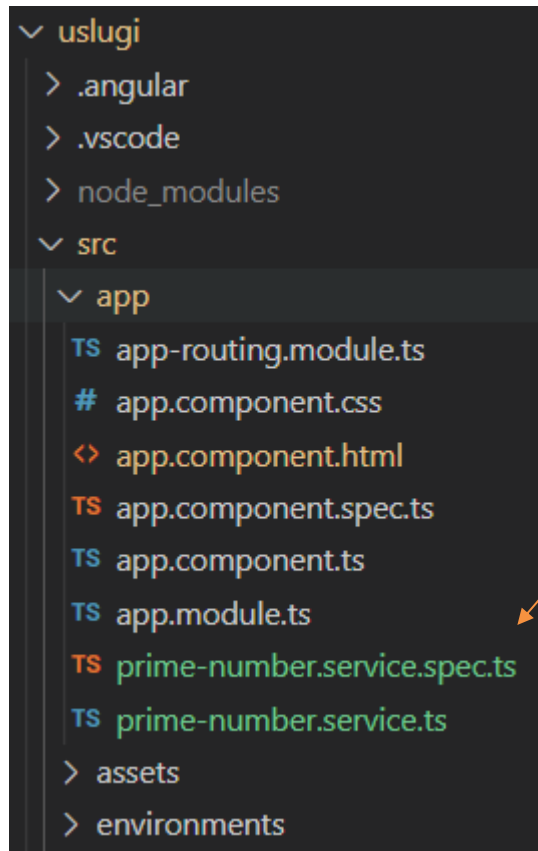


89

po naciśnięciu przycisku usługa (service) generuje liczbę pierwszą

Prime Number Please

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

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

```
TS app.component.ts M X
uslugi > src > app > TS app.component.ts > ...
1  import { Component, Inject } from '@angular/core';
2  import { PrimeNumberService } from './prime-number.service';
3
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css'],
8    providers: [PrimeNumberService]
9  })
10 export class AppComponent {
11   // liczba pierwsza
12   primeNumber!: number;
13   // w konstruktorze przypisujemy liczbę pierwszą zwracaną przez
14   // PrimeNumberService do lokalnego pola primeNumber
15   constructor(private primeNumberService: PrimeNumberService){
16     this.primeNumber = primeNumberService.primeNumberPlease();
17   }
18 }
```

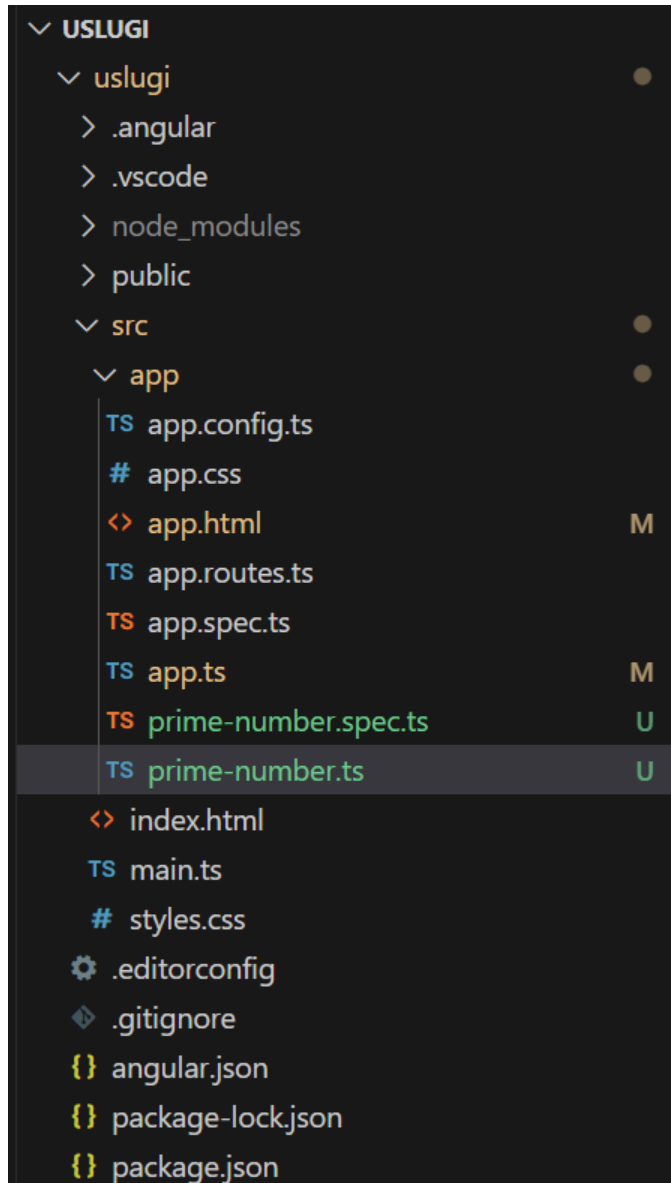
dzięki umieszczeniu serwisu w tabeli providers będzie można odwoływać się do niego w innych komponentach

Usługi PrimeNumberService

konfiguracja widoku

```
<> app.component.html M ✕  
uslugi > src > app > <> app.component.html > ...  
1  
2 <!-- wyświetlamy liczbę pierwszą wygenerowaną przez PrimeNumberService-->  
3 <h1>{{primeNumber}}</h1>  
4 <!-- przeładujemy stronę -->  
5 <button onclick="location.reload();">Prime Number Please</button>
```

Usługi



generate service nazwa usługi

```
ng g s primeNumber
```

generowanie usługi (klasy) PrimeNumber

Usługi PrimeNumber

```
1 import { Component, signal } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { PrimeNumber } from './prime-number';
4
5 @Component({
6   selector: 'app-root',
7   imports: [RouterOutlet],
8   templateUrl: './app.html',
9   styleUrls: ['./app.css']
10 })
11 export class App {
12   protected readonly title = signal('uslugi');
13   // liczba pierwsza
14   primeNumber!: number;
15   // w konstruktorze przypisujemy liczbę pierwszą zwracaną przez
16   // PrimeNumberService do lokalnego pola primeNumber
17   constructor(primeNumber: PrimeNumber){
18     this.primeNumber = primeNumber.primeNumberPlease();
19   }
20 }
```

import usługi

wykorzystanie usługi

Usługi PrimeNumber

```
EXPLOLERER
...
OPEN EDITORS
X <> app.html usługi\src\app M
USLUGI
  > usługi
    > .angular
    > .vscode
    > node_modules
    > public
  > src
    > app
      TS app.config.ts
      # app.css
      <> app.html M
      TS app.routes.ts
      TS app.spec.ts
      TS app.ts 1, M
      TS prime-number.spec.ts U
      TS prime-number.ts U
    <> index.html
    TS main.ts

<> app.html M X
uslugi > src > app > <> app.html > ...
Go to component | You, 1 second ago | 1 author (You)
1 <!-- wyświetlamy liczbę pierwszą wygenerowaną przez serwis PrimeNumber-->
2 <h1>{{primeNumber}}</h1>
3 <!-- przeglądamy stronę -->
4 <button onclick="location.reload();">Prime Number Please</button>
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

po kliknięciu na przycisk
przeładujemy stronę
i wyświetlamy liczbę pierwszą

Formularz

Ankieta

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

pleć

mężczyzna kobieta

zainteresowania

sport film elektronika informatyka

szkoła

Uniwersytet Łódzki ▾

krótko o sobie

jestem Jan Kowalski

submit reset

Ankieta obsługiwana przez Angular

localhost:4200

Twoja ankieta:

imię: Jan
nazwisko: Kowalski
email: jan@kowalski.pl
pleć: kobieta
zainteresowania: sport film elektronika informatyka
szkoła: uniwersytet
o_sobie: jestem Jan Kowalski

OK

po naciśnięciu przycisku wyświetlają się zaznaczone opcje Ankiety w oknie alert

Formularz Ankieta

```
<> app.component.html M X
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="mężczyzna" [(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

```
TS app.component.ts M X
formularz > 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    imie: string = "";
10   nazwisko: string = "";
11   email: string = "";
12   plec: string = "";
13   sport: string = "";
14   film: string = "";
15   elektronika: string = "";
16   informatyka: string = "";
17   szkola: string = "";
18   o_sobie: string = "";

```

pola klasy
AppComponent
odpowiadają
elementom ankiety

metoda onSubmit()
klasy AppComponent

```
onSubmit(){
  // zbieramy dane z ankiety
  if(this.plec!=""){
    if(this.plec=="męczyzna"){
      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 +
    "\npleć: " + this.plec +
    "\nzainteresowania: " + zainteresowania +
    "\nszkola: " + this.szkola +
    "\no_sobie: " + this.o_sobie
  );
}
```

Angular ver.14

Formularz Ankieta

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

import modułu odpowiedzialnego za obsługę formularza

Formularz

Ankieta z walidacją

FormularzWalidacja x +

localhost:4200

Ankieta

imię
 nazwisko
 e-mail

pleć

mężczyzna kobieta

zainteresowania

sport film elektronika informatyka

szkoła

▼

krótko o sobie

submit reset

FormularzWalidacja x +

localhost:4200

Ankieta

imię
Imię jest wymagane
 nazwisko
Nazwisko jest wymagane
 ad e-mail
email jest wymagany
wpisz poprawny email

pleć

mężczyzna kobieta

zainteresowania

sport film elektronika informatyka

szkoła

▼

krótko o sobie

submit reset

poła wymagane są w czerwonej ramce

Formularz Ankieta z walidacją

zmiany związane z
walidacją pola imie
(pole jest wymagane)

zmienna szablonowa
name przypisana do
ngModel

```
<> app.component.html M X
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
zmiennnej szablonowej name

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

Formularz Ankieta z walidacją

```
<> app.component.html M X
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">imie<br>
4   <div *ngIf="name.invalid && name.touched">Imię jest wymagane</div>
5   <input type="text" name="nazwisko" [(ngModel)] = "nazwisko" required #surname="ngModel">nazwisko<br>
6   <div *ngIf="surname.invalid && surname.touched">Nazwisko jest wymagane</div>
7   <input type="text" name="email" [(ngModel)] = "email" required pattern=".+@.+" #myEmail="ngModel">e-mail<br>
8   <div *ngIf="myEmail.invalid && myEmail.touched">email jest wymagany</div>
9   <div *ngIf="myEmail.touched && myEmail.errors?.['pattern']">wpisz poprawny email</div>
10  <h3>pleć</h3>
11  mężczyzna<input type="radio" name="plec" value="męczyzna" [(ngModel)] = "plec">
12  kobieta<input type="radio" name="plec" value="kobieta">
13
14  <h3>zainteresowania</h3>
15
16  sport<input type="checkbox" name="sport" value="1" [(ngModel)] = "sport">
17  film<input type="checkbox" name="film" value="1" [(ngModel)] = "film">
18  elektronika<input type="checkbox" name="elektronika" value="1" [(ngModel)] = "elektronika">
19  informatyka<input type="checkbox" name="informatyka" value="1" [(ngModel)] = "informatyka">
20
21  <h3>szkoła</h3>
22  <select name="szkola" [(ngModel)] = "szkola" required>
23    <option value="podstawowa">Podstawowa</option>
24    <option value="liceum">Liceum Ogólnokształcące</option>
25    <option value="technikum">Technikum</option>
26    <option value="politechnika">Politechnika Łódzka</option>
27    <option value="uniwersytet">Uniwersytet Łódzki</option>
28    <option value="inna">inna</option>
29  </select>
30
31  <h3>krótko o sobie</h3>
32  <textarea name="o_sobie" rows=5 cols=40 [(ngModel)] = "o_sobie"></textarea>
33
34  <hr>
35  <input type="submit" value="submit" [disabled]="!formularz.form.valid" (click)="onSubmit()">
36  <input type="reset" value="reset">
37 </form>
```

szablon ze zmianami
walidacyjnymi

Formularz Ankieta z walidacją

```
TS app.component.ts X
formularzWalidacja > 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   imie: string = "";
10  nazwisko: string = "";
11  email: string = "";
12  plec: string = "";
13  sport: string = "";
14  film: string = "";
15  elektronika: string = "";
16  informatyka: string = "";
17  szkola: string = "";
18  o_sobie: string = "";
19
20  onSubmit(){
21    // zbieramy dane z ankiety
22    if(this.plec!=""){
23      if(this.plec=="meczczyzna"){
24        this.plec="mężczyzna";
25      }else{
26        this.plec="kobieta";
27      }
28    }
29    let zainteresowania = "";
30    if (this.sport == "1"){
31      zainteresowania += " sport";
32    }
33    if (this.film == "1"){
34      zainteresowania += " film";
35    }
36    if (this.elektronika == "1"){
37      zainteresowania += " elektronika";
38    }
39    if (this.informatyka == "1"){
40      zainteresowania += " informatyka";
41    }
42    alert("Twoja ankieta:\n" +
43      "\nimie: " + this.imie +
44      "\nnazwisko: " + this.nazwisko +
45      "\nemail: " + this.email +
46      "\nplec: " + this.plec +
47      "\nzainteresownia: " + zainteresowania +
48      "\nszkola: " + this.szkola +
49      "\no_sobie: " + this.o_sobie
50    );
51  }
52 }
53
```

```
TS app.module.ts M X
formularzWalidacja > src > app > TS 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
7 @NgModule({
8   declarations: [
9     AppComponent
10  ],
11  imports: [
12    BrowserModule,
13    FormsModule
14  ],
15  providers: [],
16  bootstrap: [AppComponent]
17 })
18 export class AppModule { }
```

```
# app.component.css M X
formularzWalidacja > src > app > # app.compo
1 .ng-invalid{
2   border: 1px solid red;
3   width:30%;
4 }
5 #ankietaForm{
6   border:0px;
7 }
```

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

dzięki stylom
wymagane pola mają
czerwoną obramówkę

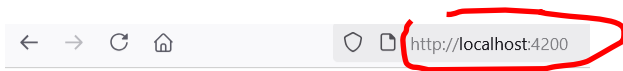
Formularz

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

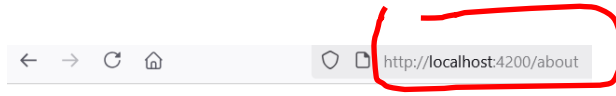
routing - podstrony



[Home](#)
[About](#)
[Contact](#)



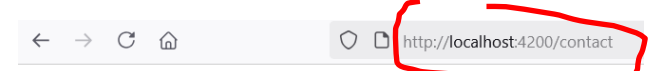
home works!



[Home](#)
[About](#)
[Contact](#)



about works!



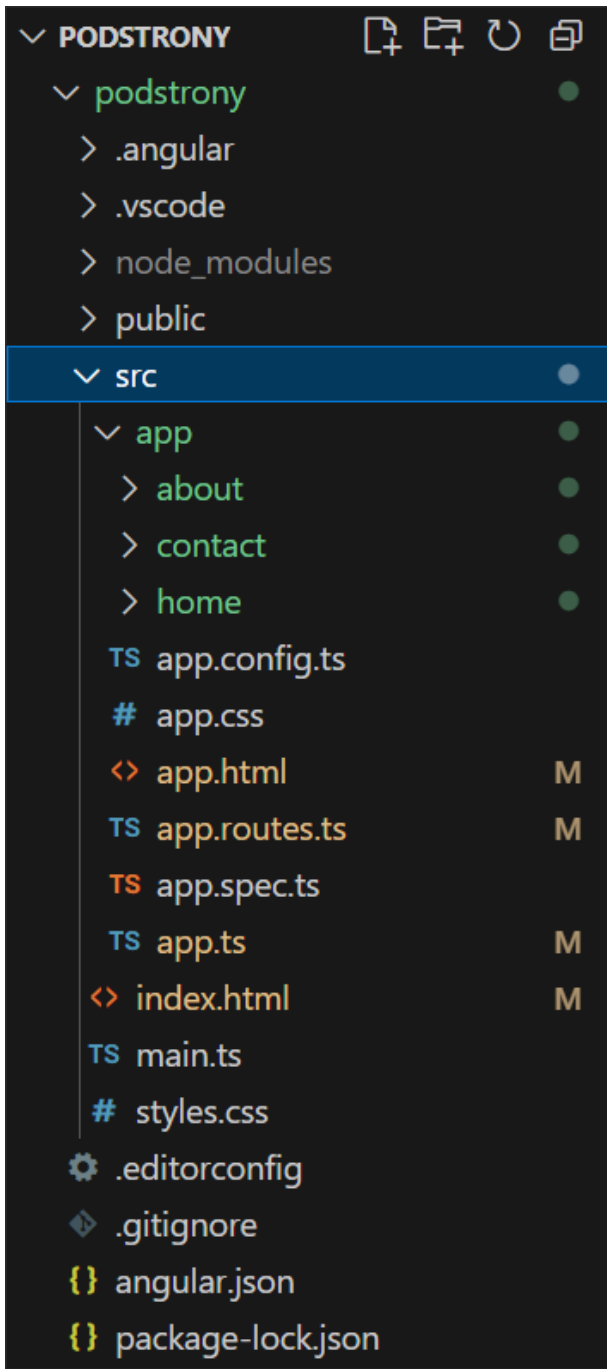
[Home](#)
[About](#)
[Contact](#)



contact works!

podstrony są
komponentami

routing - podstrony



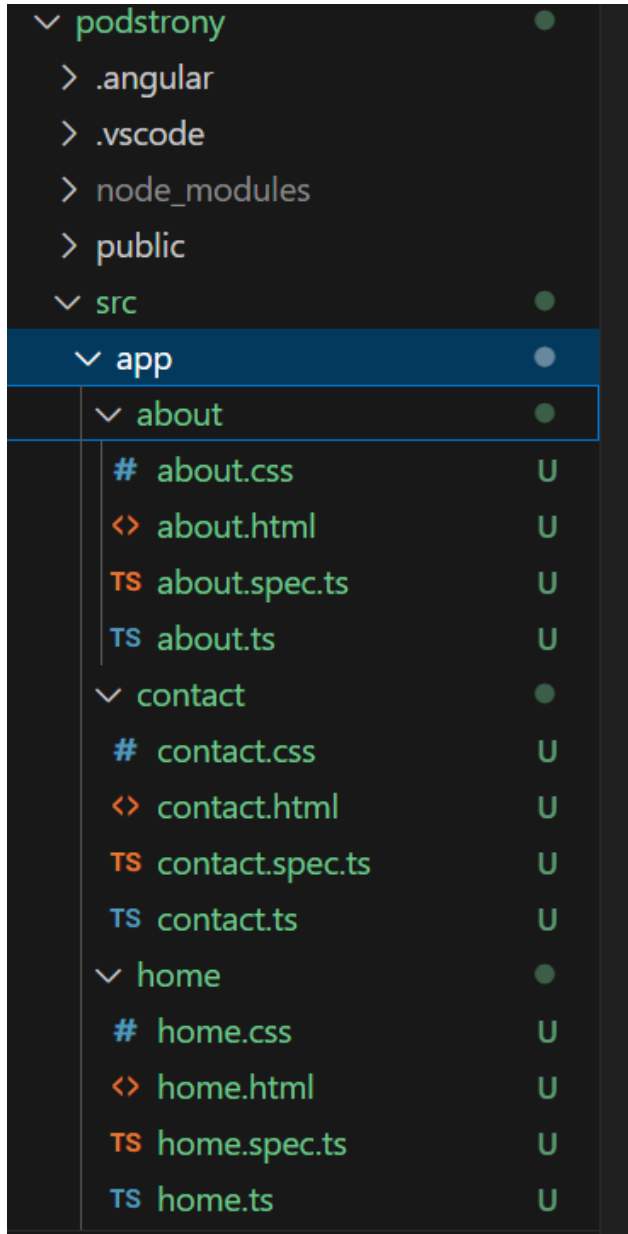
generujemy nowe komponenty,
(nasze podstrony, w terminalu):



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

Angular generate component

routing - podstrony



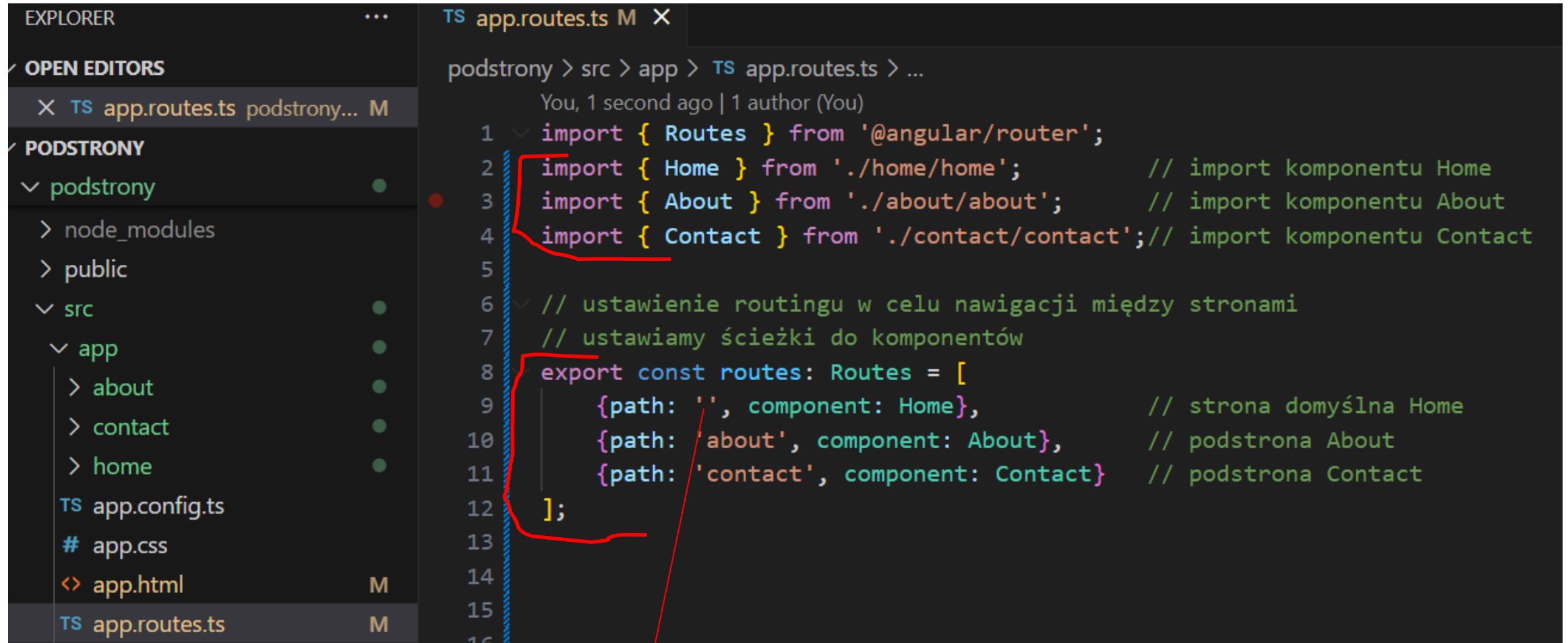
```
<> home.html U X
podstrony > src > app > home > <> home.html > ...
Go to component
1 <p>home works!</p>
2 |
```

```
<> about.html U X
podstrony > src > app > about > <> a
Go to component
1 <p>about works!</p>
2
```

```
<> contact.html U X
podstrony > src > app > contact > <> conta
Go to component
1 <p>contact works!</p>
2
```

routing - podstrony

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



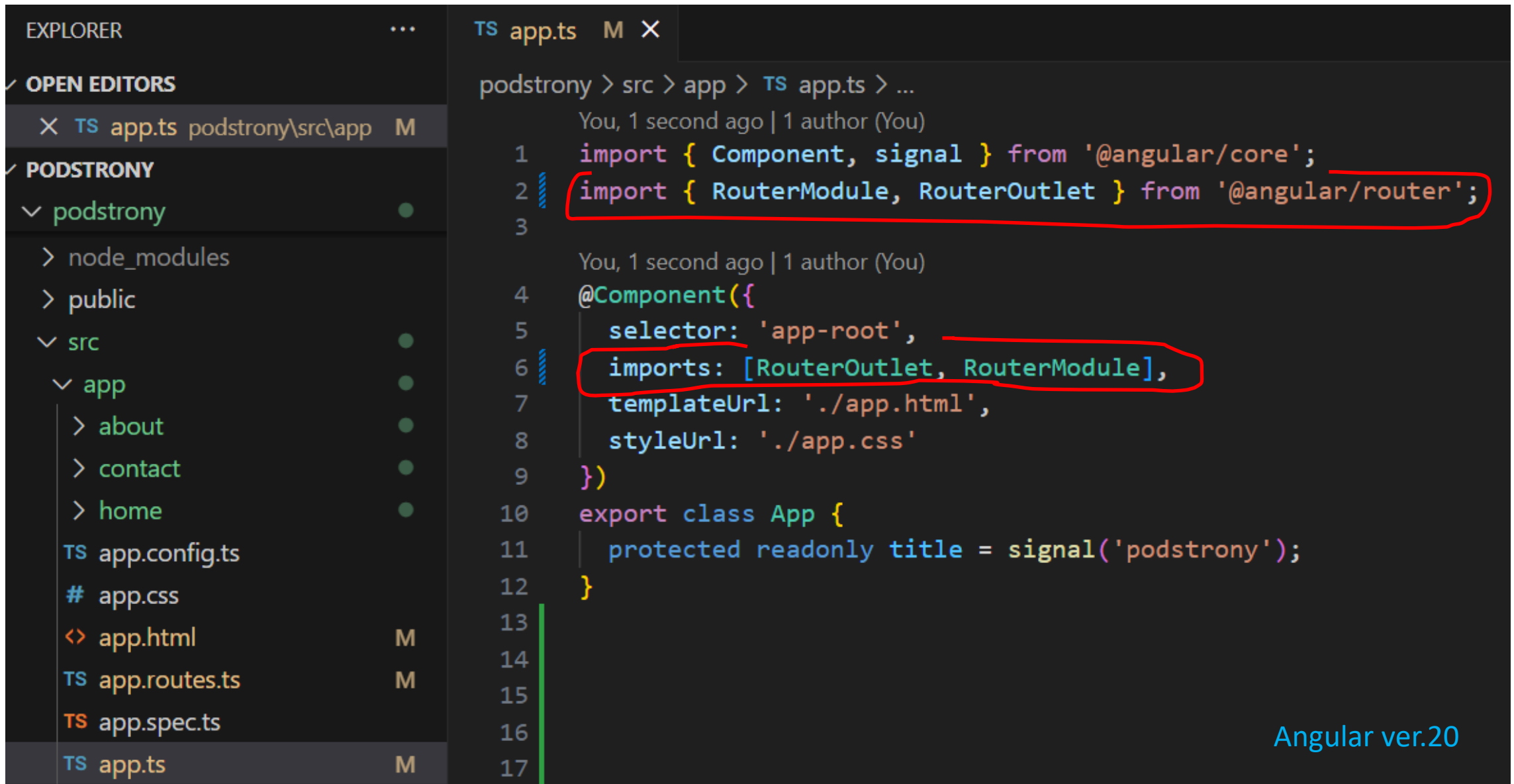
```
EXPLORER
...
OPEN EDITORS
  X TS app.routes.ts podstrony... M
PODSTRONY
  v podstrony
    > node_modules
    > public
    v src
      v app
        > about
        > contact
        > home
        TS app.config.ts
        # app.css
        <> app.html M
        TS app.routes.ts M

TS app.routes.ts M X
podstrony > src > app > TS app.routes.ts > ...
You, 1 second ago | 1 author (You)
1 import { Routes } from '@angular/router';
2 import { Home } from './home/home'; // import komponentu Home
3 import { About } from './about/about'; // import komponentu About
4 import { Contact } from './contact/contact'; // import komponentu Contact
5
6 // ustawienie routingu w celu nawigacji między stronami
7 // ustawiamy ścieżki do komponentów
8 export const routes: Routes = [
9   {path: '', component: Home}, // strona domyślna Home
10  {path: 'about', component: About}, // podstrona About
11  {path: 'contact', component: Contact} // podstrona Contact
12 ];
13
14
15
16
```

między apostrofami nie ma spacji!!!!

routing - podstrony

app.ts – w tym pliku importujemy biblioteki sterujące routingiem



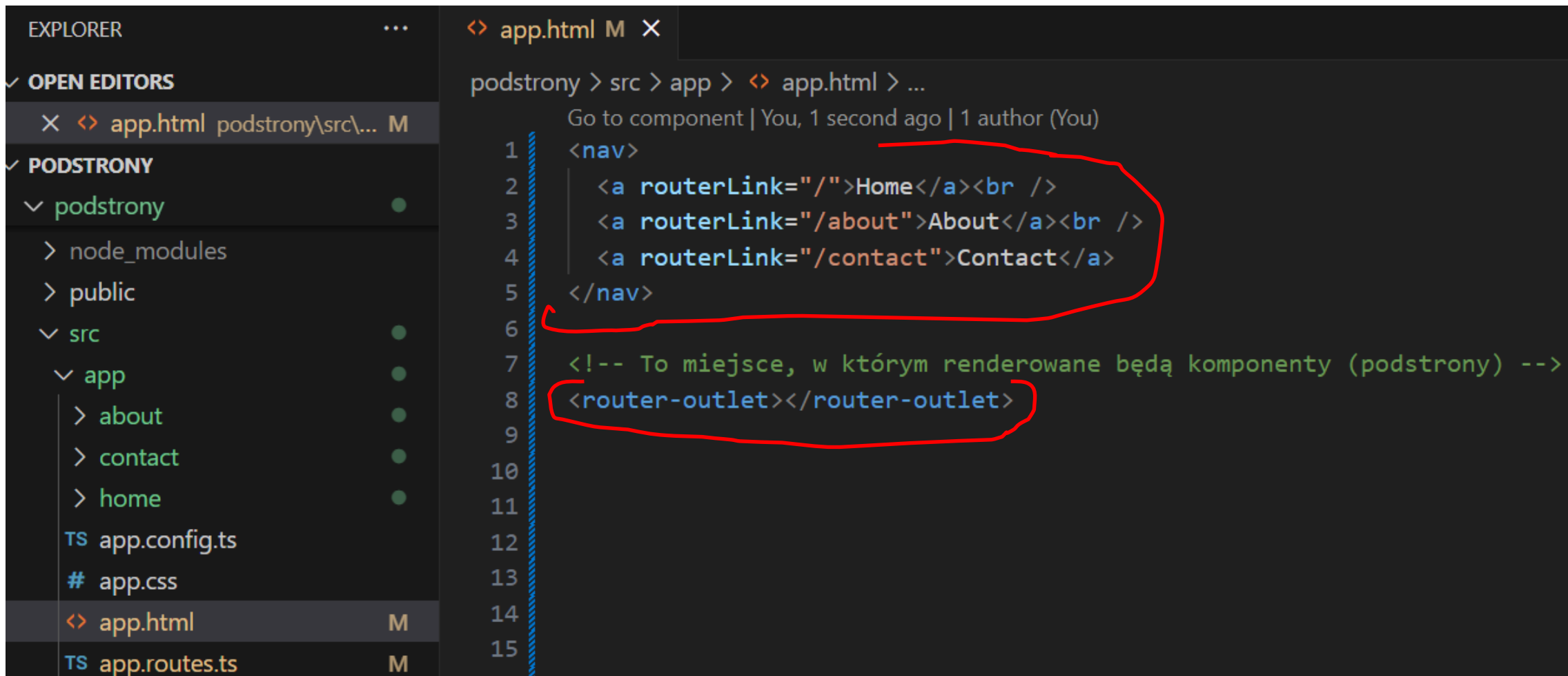
```
EXPLORER
...
OPEN EDITORS
  X TS app.ts podstrony\src\app M
PODSTRONY
  v podstrony
    > node_modules
    > public
    v src
      v app
        > about
        > contact
        > home
        TS app.config.ts
        # app.css
        <> app.html M
        TS app.routes.ts M
        TS app.spec.ts
        TS app.ts M

TS app.ts M X
podstrony > src > app > TS app.ts > ...
You, 1 second ago | 1 author (You)
1 import { Component, signal } from '@angular/core';
2 import { RouterModule, RouterOutlet } from '@angular/router';
3
You, 1 second ago | 1 author (You)
4 @Component({
5   selector: 'app-root',
6   imports: [RouterOutlet, RouterModule],
7   templateUrl: './app.html',
8   styleUrls: ['./app.css']
9 })
10 export class App {
11   protected readonly title = signal('podstrony');
12 }
13
14
15
16
17
```

Angular ver.20

routing - podstrony

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



```
EXPLORER
...
OPEN EDITORS
  X <> app.html podstrony\src\... M
PODSTRONY
  v podstrony
    > node_modules
    > public
    v src
      v app
        > about
        > contact
        > home
        TS app.config.ts
        # app.css
        <> app.html M
        TS app.routes.ts M

podstrony > src > app > <> app.html > ...
Go to component | You, 1 second ago | 1 author (You)
1 <nav>
2   <a routerLink="/">Home</a><br />
3   <a routerLink="/about">About</a><br />
4   <a routerLink="/contact">Contact</a>
5 </nav>
6
7 <!-- To miejsce, w którym renderowane będą komponenty (podstrony) -->
8 <router-outlet></router-outlet>
9
10
11
12
13
14
15
```

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 przez api

Angular (frontend)

```
json
Lista produktów
Pobierz dane
Lista produktów
• jajka - 20 zł
• ziemniaki - 2 zł
• pomidory - 2 zł
• cebula - 2 zł
• chleb - 2 zł
• ser żółty - 2 zł
• mandarynki - 2 zł
```

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

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

response

response

request

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

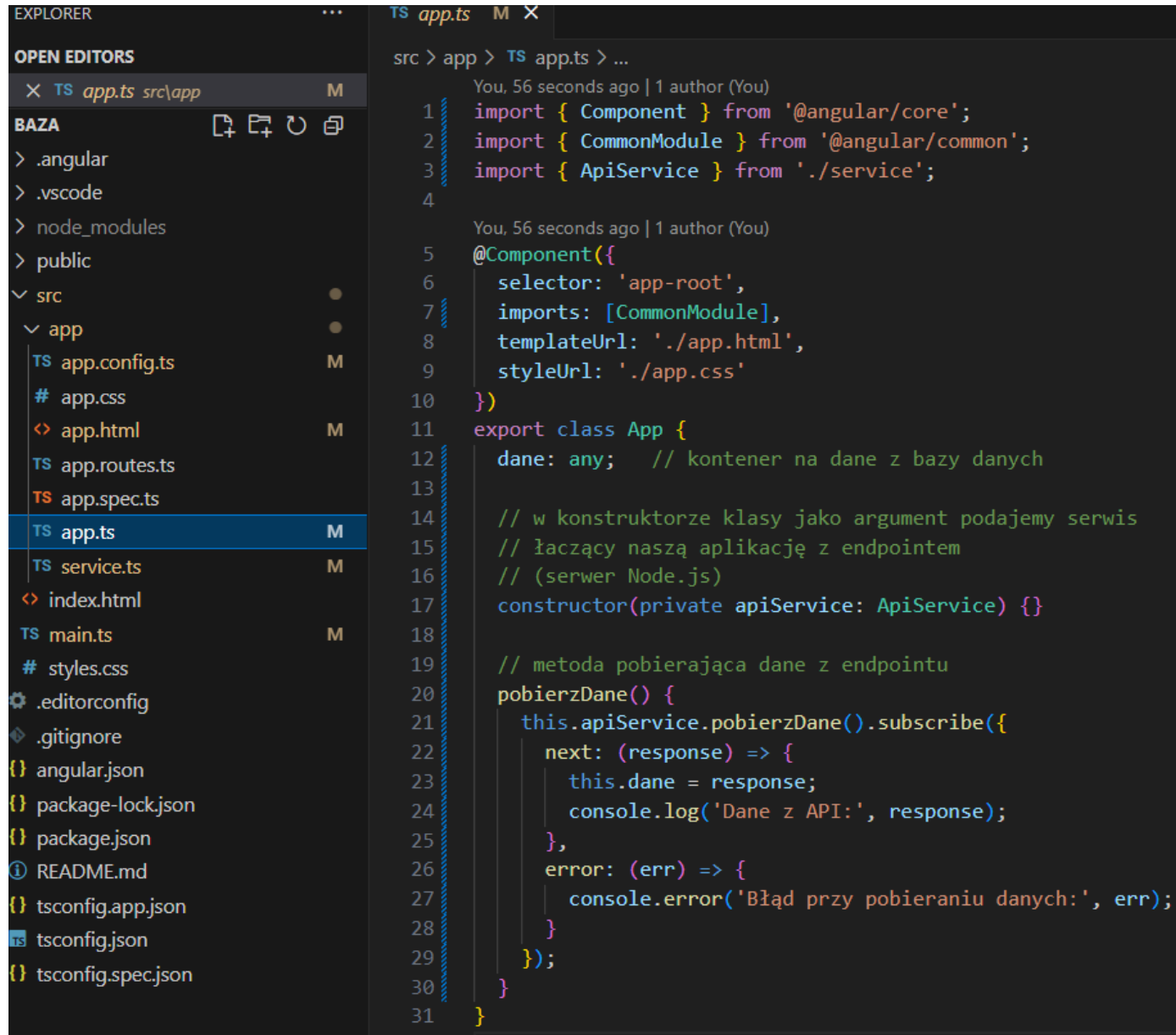
server Node.js Express nasłuchuje na <http://localhost:3002> (endpoint)

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

baza danych sklep MySQL (XAMPP)

Połączenie z bazą danych MySQL Angular (frontend)

app.ts



```
src > app > TS app.ts > ...
You, 56 seconds ago | 1 author (You)
1 import { Component } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { ApiService } from './service';
4
You, 56 seconds ago | 1 author (You)
5 @Component({
6   selector: 'app-root',
7   imports: [CommonModule],
8   templateUrl: './app.html',
9   styleUrls: ['./app.css']
10 })
11 export class App {
12   dane: any; // kontener na dane z bazy danych
13
14   // w konstruktorze klasy jako argument podajemy serwis
15   // łączący naszą aplikację z endpointem
16   // (serwer Node.js)
17   constructor(private apiService: ApiService) {}
18
19   // metoda pobierająca dane z endpointu
20   pobierzDane() {
21     this.apiService.pobierzDane().subscribe({
22       next: (response) => {
23         this.dane = response;
24         console.log('Dane z API:', response);
25       },
26       error: (err) => {
27         console.error('Błąd przy pobieraniu danych:', err);
28       }
29     });
30   }
31 }
```

komenda w terminalu:

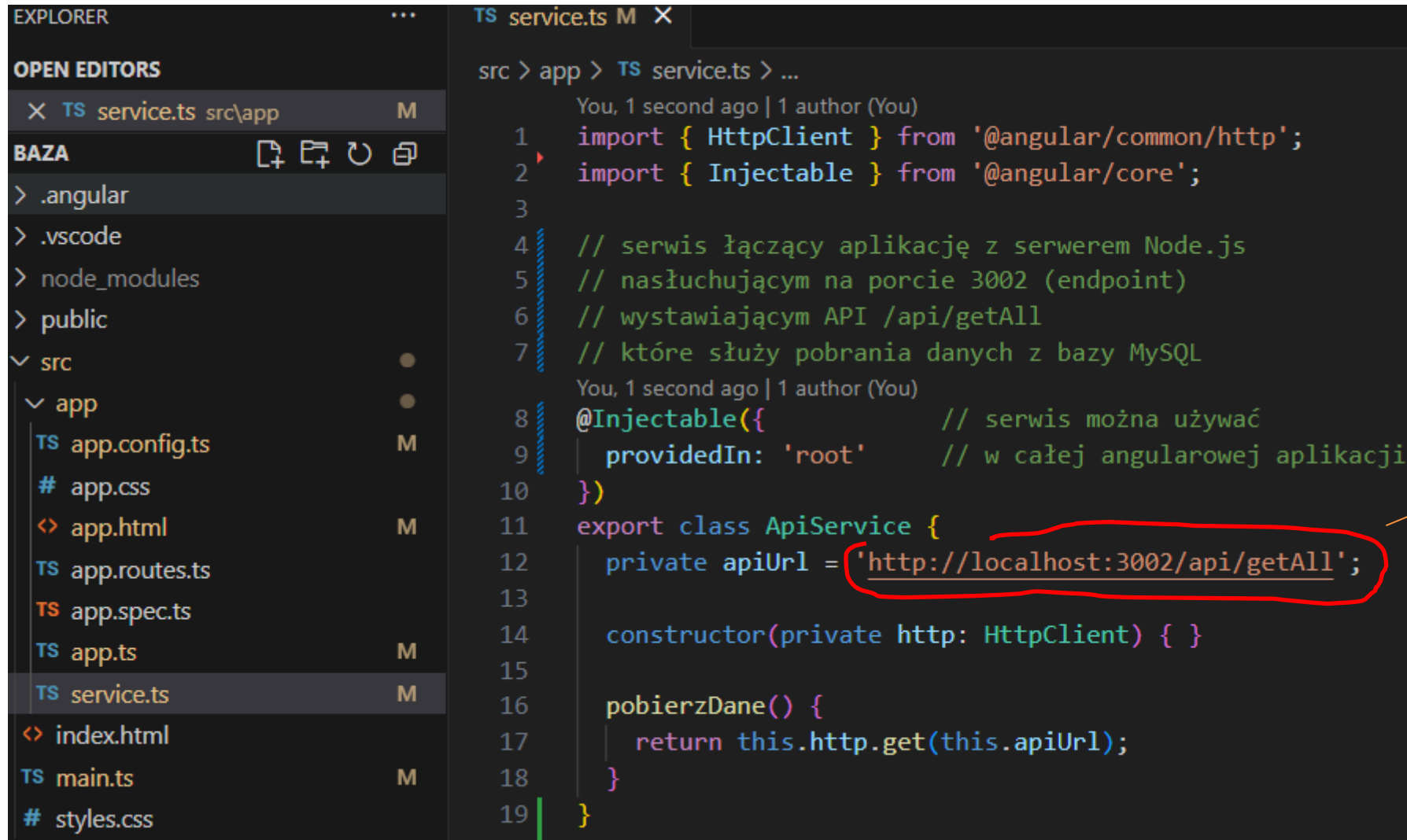
ng new baza

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



```
EXPLORER
OPEN EDITORS
  X TS service.ts src\app M
BAZA
  > .angular
  > .vscode
  > node_modules
  > public
  v src
    v app
      TS app.config.ts M
      # app.css
      <> app.html M
      TS app.routes.ts
      TS app.spec.ts
      TS app.ts M
      TS service.ts M
      <> index.html
      TS main.ts M
      # styles.css

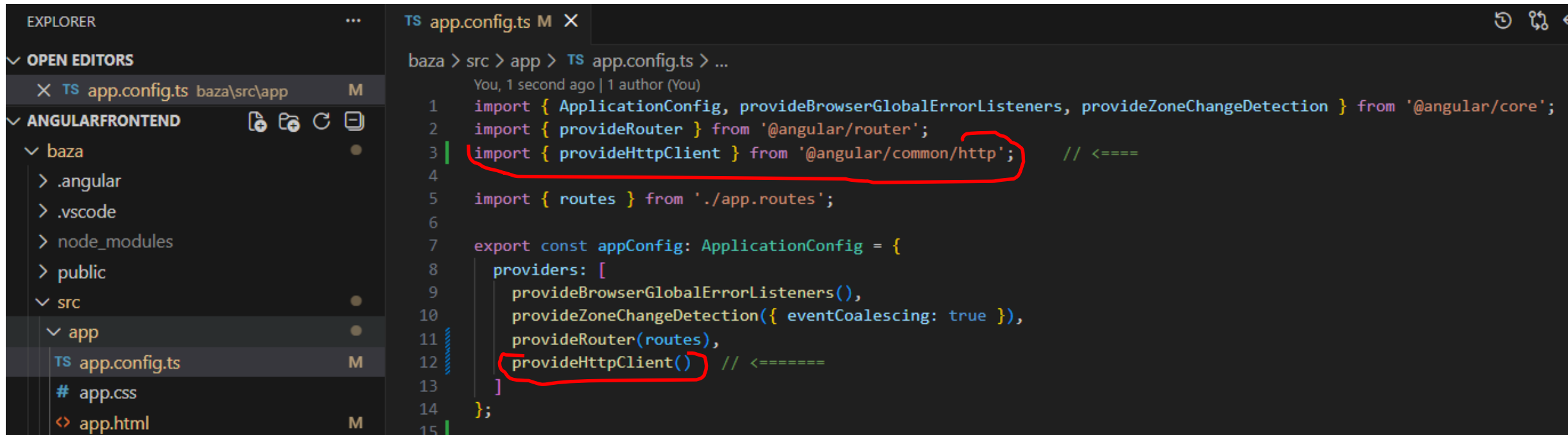
TS service.ts M X
src > app > TS service.ts > ...
You, 1 second ago | 1 author (You)
1  import { HttpClient } from '@angular/common/http';
2  import { Injectable } from '@angular/core';
3
4  // serwis łączący aplikację z serwerem Node.js
5  // nasłuchującym na porcie 3002 (endpoint)
6  // wystawiającym API /api/getAll
7  // które służy pobrania danych z bazy MySQL
You, 1 second ago | 1 author (You)
8  @Injectable({
9    providedIn: 'root' // w całej angularowej aplikacji
10 })
11 export class ApiService {
12   private apiUrl = 'http://localhost:3002/api/getAll';
13
14   constructor(private http: HttpClient) { }
15
16   pobierzDane() {
17     return this.http.get(this.apiUrl);
18   }
19 }
```

adres API

Angular ver.20

Połączenie z bazą danych MySQL Angular (frontend)

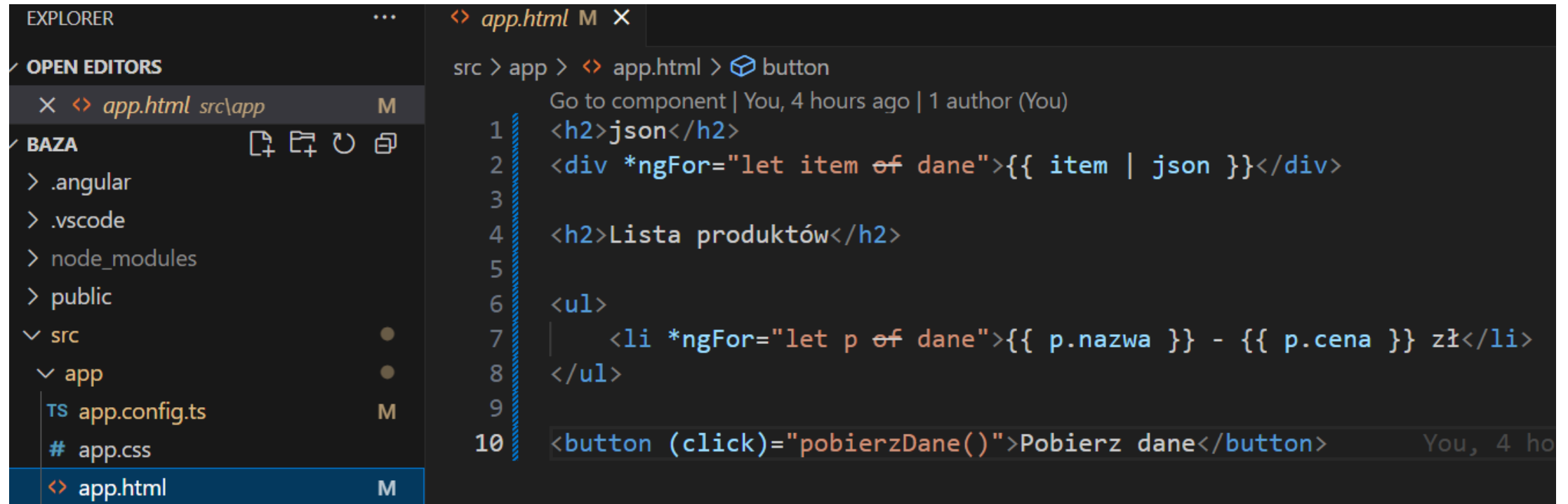
app.config.ts



```
EXPLORER
...
OPEN EDITORS
  X TS app.config.ts baza\src\app M
ANGULARFRONTEND
  baza
    .angular
    .vscode
    node_modules
    public
    src
      app
        TS app.config.ts M
        # app.css
        <> app.html M
TS app.config.ts M X
baza > src > app > TS app.config.ts > ...
You, 1 second ago | 1 author (You)
1 import { ApplicationConfig, provideBrowserGlobalErrorListeners, provideZoneChangeDetection } from '@angular/core';
2 import { provideRouter } from '@angular/router';
3 import { provideHttpClient } from '@angular/common/http'; // <====
4
5 import { routes } from './app.routes';
6
7 export const appConfig: ApplicationConfig = {
8   providers: [
9     provideBrowserGlobalErrorListeners(),
10    provideZoneChangeDetection({ eventCoalescing: true }),
11    provideRouter(routes),
12    provideHttpClient() // <=====
13  ]
14 };
15
```

Połączenie z bazą danych MySQL Angular (frontend)

app.html



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left and the Editor view on the right. The Explorer sidebar shows the project structure with the following folders and files:

- BAZA
 - .angular
 - .vscode
 - node_modules
 - public
- src
 - app
 - app.config.ts
 - app.css
 - app.html

The Editor view shows the content of the `app.html` file:

```
src > app > <> app.html > button
Go to component | You, 4 hours ago | 1 author (You)
1 <h2>json</h2>
2 <div *ngFor="let item of dane">{{ item | json }}</div>
3
4 <h2>Lista produktów</h2>
5
6 <ul>
7   <li *ngFor="let p of dane">{{ p.nazwa }} - {{ p.cena }} zł</li>
8 </ul>
9
10 <button (click)="pobierzDane()">Pobierz dane</button>
```

uruchomienie Angulara w terminalu:

ng serve

Angular ver.20

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

```
EXPLORER
OPEN EDITORS
  X JS index.js
NODESERVER
  > config
  > node_modules
  JS index.js
  info.txt
  {} package-lock.json
  {} package.json

JS index.js
1  const express = require('express');
2  const db = require('./config/db');
3  const cors = require('cors');
4
5  const app = express();
6
7  const PORT = 3002;
8  app.use(cors());
9  app.use(express.json());
10
11
12 // obsługa żądania GET strony głównej
13 app.get('/', (req, res) => {
14   res.send('<h1>Hello World Express!</h1>');
15 });
16
17
18 // Route to get all downloads
19 app.get("/api/getAll", (req,res)=>{
20   db.query("SELECT * FROM towary", (err,result)=>{
21     if(err) {
22       console.error(err);
23       // Wysyłamy odpowiedź z kodem 500 i komunikatem o błędzie
24       return res.status(500).send({ error: 'Błąd serwera' });
25     }
26     res.send(result)
27   });
28 });
29
30 app.listen(PORT, ()=>{
31   console.log(`Server is running on ${PORT}`)
32 })
```

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

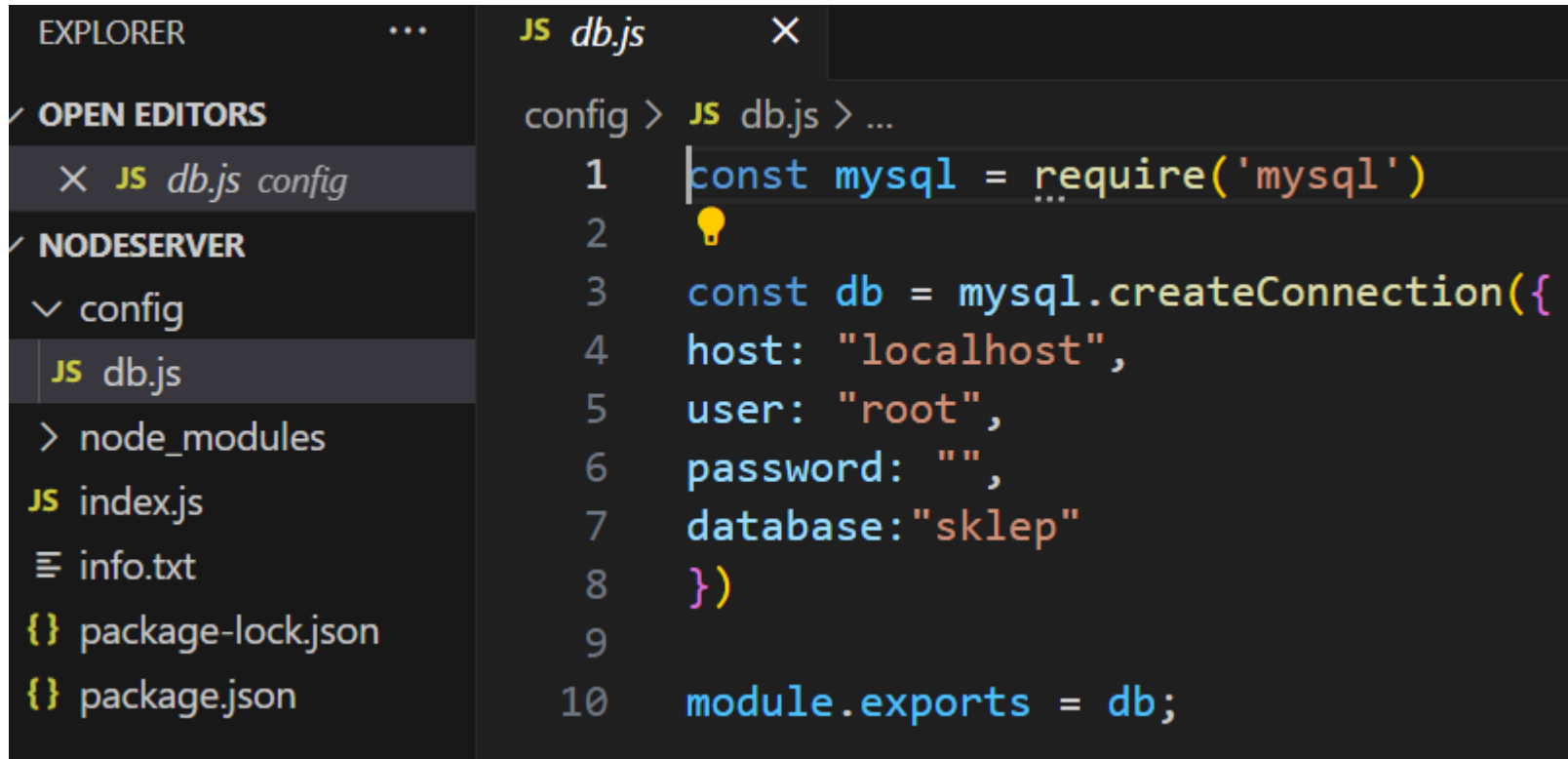
serwer Node.js

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



```
EXPLORER  ...  JS db.js  X
OPEN EDITORS
  X JS db.js config
NODESERVER
  config
  JS db.js
  > node_modules
  JS index.js
  info.txt
  package-lock.json
  package.json

config > JS db.js > ...
1  const mysql = require('mysql')
2  ⚡
3  const db = mysql.createConnection({
4  host: "localhost",
5  user: "root",
6  password: "",
7  database: "sklep"
8  })
9
10 module.exports = db;
```

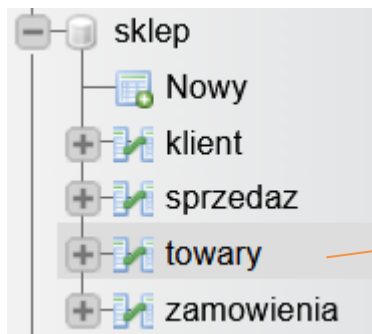
uruchomienie serwera
w terminalu:

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)

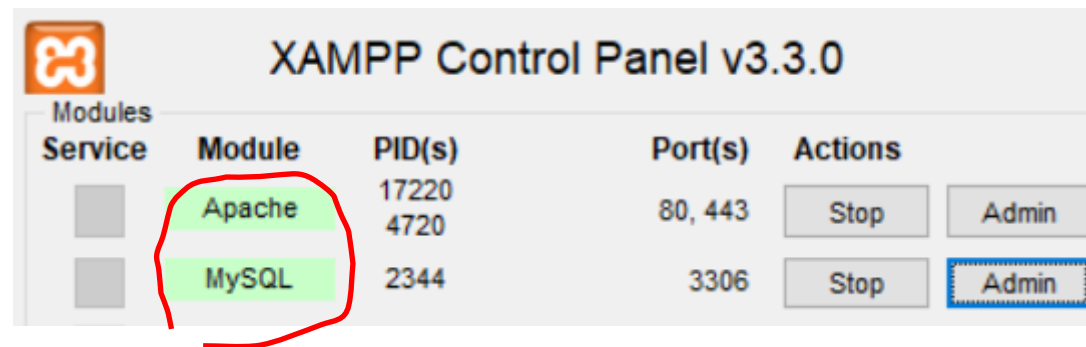


				id	nazwa	cena
<input type="checkbox"/>	Edytuj	Kopiuj	Usuń	1	jajka	20
<input type="checkbox"/>	Edytuj	Kopiuj	Usuń	2	ziemniaki	2
<input type="checkbox"/>	Edytuj	Kopiuj	Usuń	3	pomidory	2
<input type="checkbox"/>	Edytuj	Kopiuj	Usuń	4	cebula	2
<input type="checkbox"/>	Edytuj	Kopiuj	Usuń	5	chleb	2
<input type="checkbox"/>	Edytuj	Kopiuj	Usuń	6	ser żółty	2
<input type="checkbox"/>	Edytuj	Kopiuj	Usuń	7	mandarynki	2

Połączenie z bazą danych MySQL

baza sklep MySQL

w panelu kontrolnym XAMPP
uruchamiamy serwery Apache i
MySQL



The screenshot shows the XAMPP Control Panel v3.3.0 interface. It features a table with columns for Service, Module, PID(s), Port(s), and Actions. The Apache service is running with PID 17220 and 4720 on ports 80 and 443. The MySQL service is running with PID 2344 on port 3306. The Apache and MySQL rows are highlighted in green, and a red circle is drawn around the Apache and MySQL rows. The Admin button for MySQL is highlighted with a blue dashed border.

Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	17220 4720	80, 443	Stop Admin
<input type="checkbox"/>	MySQL	2344	3306	Stop Admin

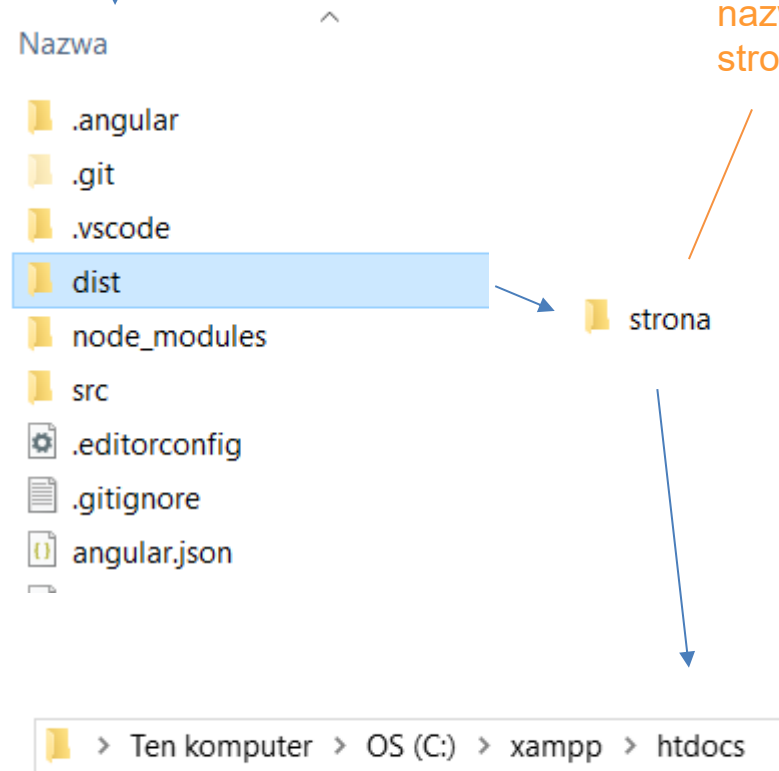
deployment

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 ./**

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


folder `nazwa_projektu` umieszczamy w folderze `htdocs` serwera Apache (XAMPP)


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




deployment

localhost/strona/browser/

 **Angular**



Galeria Pienińska



komponenty:

- tytul
- galeria
- zarowka
- zagadka
- uczen
- slownik
- tabela
- sklep
- balon
- deszcz
- zdjecie

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 przeglądarkę
```

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”;
```