Wyższa Szkoła Ekonomii
i Informatyki w Krakowie

Programowanie aplikacji webowych

Angular - wprowadzenie

- Framework do budowy Single Page Apps
- Rozwijany przez Google, oparty o Typescript
- Angular to zintegrowane środowisko wielu gotowych modułów/bibliotek
- Angular jest najczęściej wykorzystywany jest jako client-side framework. Ekosystem jest także wyposażony w moduł Angular Universal obsługujący Server Side Rendering
- Nazwa Angular określa aktualną wersję (aktualnie: wersja 15). Pierwsza wersja nazywała się AngularJS, została kompletnie przepisana
- Przykłady: Google Internals, McDonalds, Autodesk, Sears, Santander, Guardian, Weather, Deutsche Bank, Netflix, PayPal, MS Office Home, Pekao24, Alior

WSEI

Angular - wprowadzenie

- Repozytorium zależności domyślnie oparte jest o narzędzie npm i plik package.json
- Budowa aplikacji oparta jest o narzędzie ng
- ng korzysta z webpack-a do zbudowania aplikacji
- Podczas budowania następuje „tree shaking”, kod TS jest kompilowany do Javascript, Sass/Less kompilowany do CSS, JS jest dzielony na moduły oraz minifikowany.

WSEI

Frameworki - plusy

- gotowa baza kodu przyspiesza proces powstawania aplikacji
- nie wymyślamy koła na nowo
- framework wymusza spójność w kodzie
- framework wymusza stosowanie sprawdzonych praktyk programistycznych (stosowanie określonych wzorców)
- przyspiesza wdrożenia nowych osób do zespołu

WSEI

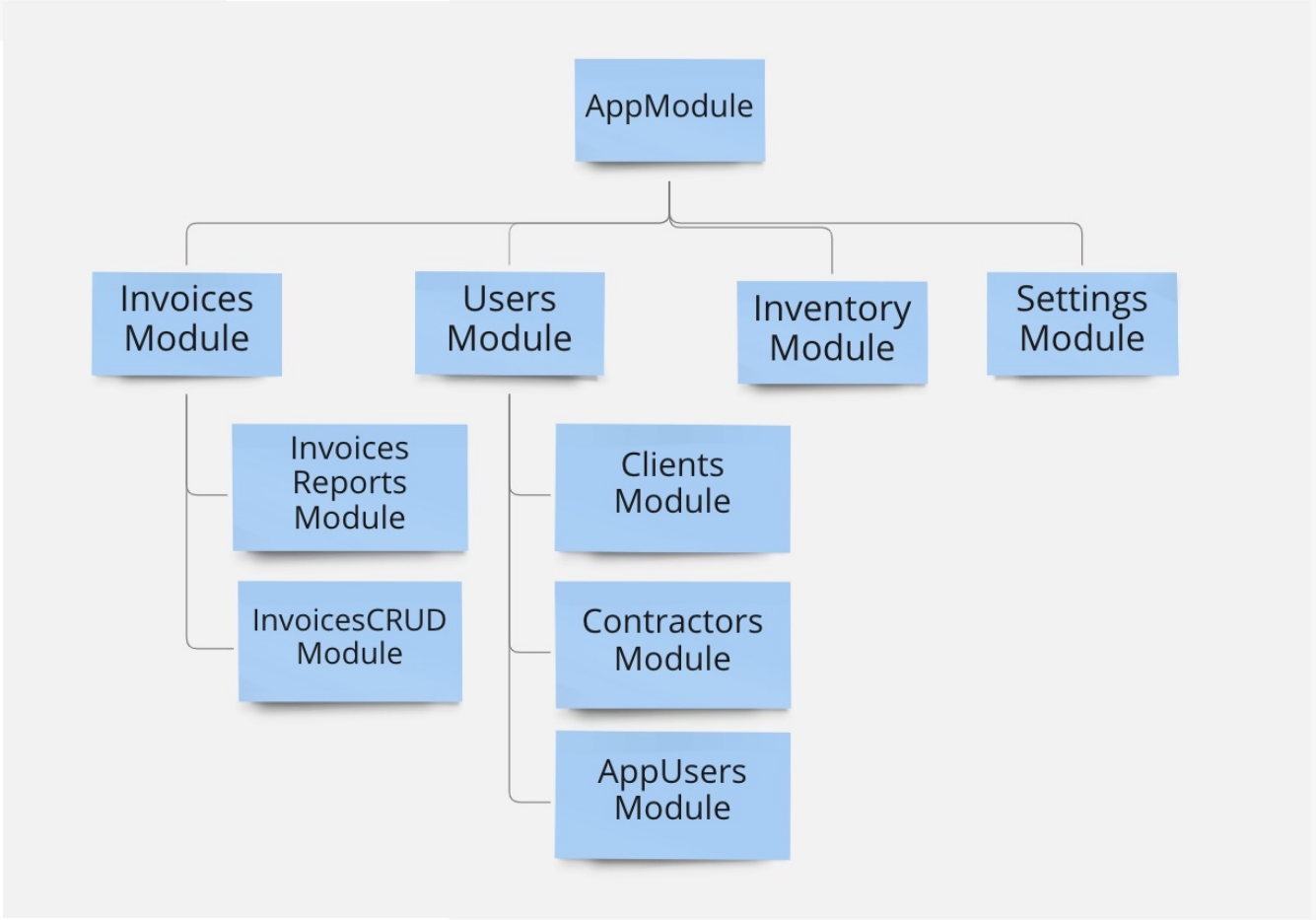
Frameworki - minusy

- trzeba się nauczyć (…i dalej uczyć)
- trzeba go utrzymać
- zazwyczaj zwiększa wielkość repozytorium kodu
- co do zasady spowalnia działanie aplikacji

WSEI

Architektura - moduły

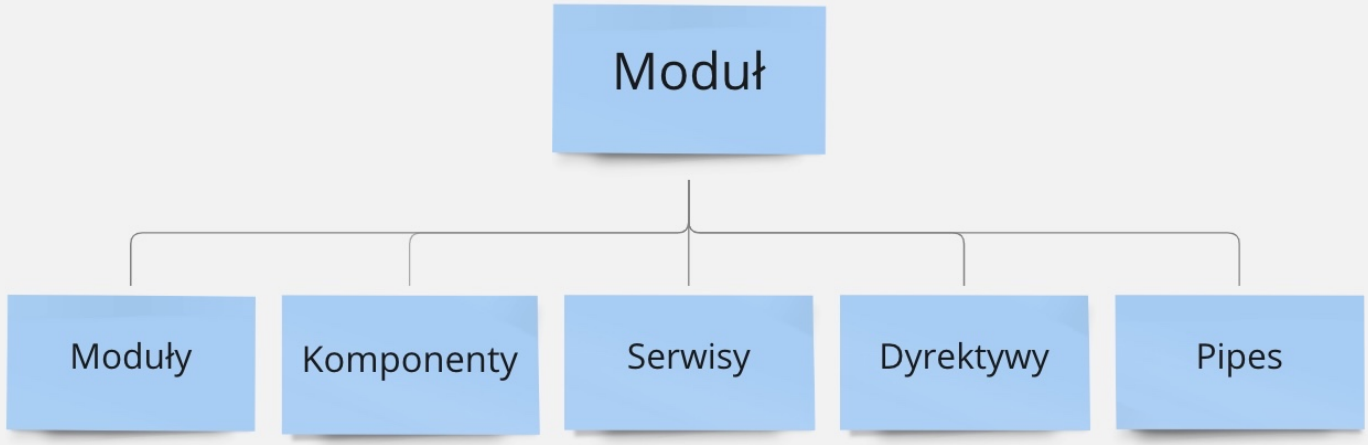
- Aplikacja dzieli się na moduły
- Moduły służą enkapsulacji i logicznemu podziałowi aplikacji
- Moduły budują drzewo routingu
- Angular dostarcza bogaty zestaw modułów rozwiązujących popularne zadania (np. routing, klient http, manipulacja DOM)



WSEI

Architektura - moduły

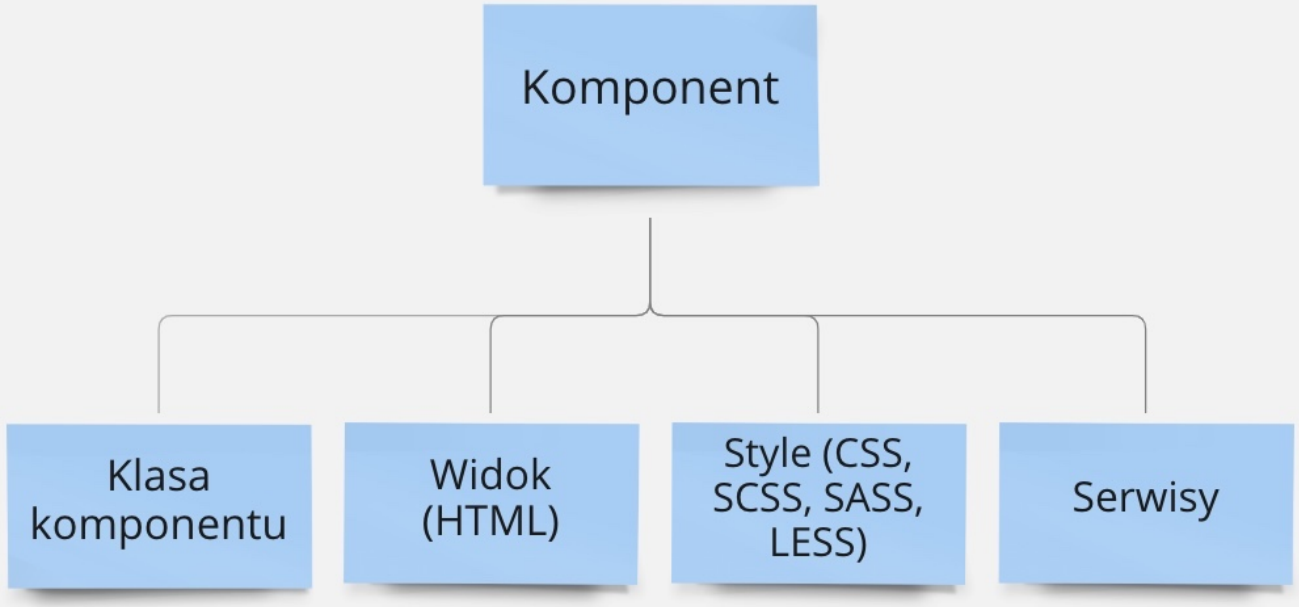
- Moduł może zawierać inne moduły, komponenty, serwisy, dyrektywy i pipes
- W kodzie moduł to klasa z dekoratorem @NgModule
- Od wersji Angular 15 stosowanie modułów jest opcjonalne!



WSEI

Architektura - komponenty

- Komponent odpowiada za pojedynczy element interfejsu użytkownika
- Komponent enkapsuluje logikę i formatowanie potrzebne do wyświetlenie konkretnego elementu UI
- Przykłady: menu, formularz, popup, lista itd
- W kodzie komponent to klasa z dekoratorem @Component
- Każdy komponent definiuje swój unikalny tag html



WSEI

Architektura - komponenty

- Przykład: https://www.pekao24.pl/logowanie

WSEI

Architektura - serwisy

- Serwisy to elementy aplikacji nie powiązane bezpośrednio z UI
- Serwisy w kodzie to (najczęściej) klasy (singletony) oznaczone dekoratorem @Injectable
- Serwisy są dostarczane poprzez moduły, komponenty oraz dyrektywy
- Widoczność serwisu jest zgodna z miejscem dostarczenia (oraz elementach potomnych)

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

@Injectable({
  providedIn: 'root'
})
export class InvoicesService {
  constructor() { }
}
```

```
@NgModule({
  declarations: [InvoiceComponent],
  imports: [CommonModule],
  providers: [InvoicesService]
})
export class InvoicesModule { }
```

WSEI

Architektura - serwisy

- Paradygmat Inversion of Control. „Nie dzwoń do nas, my zadzwonimy do Ciebie”
- IoC stoi u podstaw dostarczania serwisów (a właściwie - całego działania popularnych frameworków)
- Do kompletu (jako implementacja IoC dostarczania serwisu dochodzi wzorzec Dependency Injection)
- Angular posiada wbudowany mechanizm zarządzania zależnościami
- Angular buduje drzewo zależności - zapamiętuje ich tokeny i miejsce „zaczepienia”
- Dostarczane zależności w Angularze mogą dynamicznie podmieniane np. na potrzeby testów jednostkowych

WSEI

Serwisy

- Tworzenie serwisów: **ng g s serviceName**
- Korzystanie z serwisów następuję poprzez wbudowany w Angular mechanizm DI. Podstawowym sposobem jest wstrzykiwanie w konstruktorach komponentów/dyrektyw.

WSEI

Angular - nowy projekt

- Sprawdzanie wersji: **ng version**
- Tworzenie nowego projektu: **ng new TaskManagement —prefix tmg**
- Uruchamianie: **ng serve**
- Budowanie: **ng build**
- Testy: **ng test**

WSEI

Angular - nowy projekt

- ng podczas tworzenia projektu konfiguruje automatycznie nowe repozytorium git
- Repozytorium kodu: katalog /src
- Parametry konfiguracyjne projektu: angular.json
- Konfiguracja TS: tsconfig.*
- Konfiguracja zależności: package.json (poprzez npm/yarn)
- Obsługiwane przeglądarki: Angular 14 i starsze: .browserlistrc, Angular15: tsconfig->target

WSEI

Angular - zmienne środowiskowe

- Domyślne zmienne środowiskowe projektu: /src/environments (do Angular 14)
- Dev na produkcji Zmiana środowisk:
  ng serve —configuration=production
  ng build —configuration=dev
- Angular 15 - konieczna edycja klucza configurations -> config_name -> fileReplacement w angular.json

WSEI

Angular - debugowanie

- SourceMaps
- Chrome Angular DevTools
- Konsola - obiekt ng

WSEI

Angular - komponenty

- Komponent to klasa widoku (z dekoratorem @Component), html oraz style
- Komponent odpowiada za jeden logiczny, kompletny element UI
- Komponenty korzystają z innych komponentów by zbudować bardziej zaawansowane elementy
- Tworzenie nowego komponentu: ng g c nazwaKomponentu

WSEI

Angular - komponenty

- Przekazywanie danych do widoku

WSEI

Komponenty - zdarzenia DOM

- Reagowanie na zdarzenia z widoku
- Zmienne widoku
- Pobieranie wartości z elementów widoku

WSEI

Komponenty - komunikacja

- W komunikacji rodzic-dziecko wartości płyną w dół, w górę płyną zdarzenia
- Przekazywanie danych do komponentu: dekorator @Input() dla właściwości, np. @Input() backgroundColor: string
- Emitowanie zdarzeń: dekorator @Output() dla właściwości, np. @Output() searchEnableChanger = new EventEmitter()

WSEI

Komponenty - cykl życia komponentu

Zdarzenia opisujące cykl życia komponentu:

- OnInit
  Jednorazowy - sygnalizuje że wszystkie @Input() zostały już zainicjalizowane
- OnChanges
  Reaguje na zmiany @Input(). Be careful.
- DoCheck
  Podczas każdego cyklu detekcji zmian. Don’t do that at home.
- AfterContentInit - jednorazowy - po inicjalizacji Projected content
- AfterContentChecked - po sprawdzeniu Projected content
- AfterViewInit - jednorazowy - po inicjalizacji widoku i widoków dzieci
- AfterViewChecked - po sprawdzeniu widoku i widoków dzieci
- OnDestroy - jednorazowy - przy niszczeniu komponentu

WSEI

Komponenty - stylowanie widoku

- Style globalne: plik /src/style.scss
- Style lokalne: style komponentu
- Stylowanie dynamiczne z poziomu TS
- Stylowanie statyczne i dynamiczne hosta

WSEI

Pipes

- Pipe to klasa udekorowana @Pipe
- Pipes pracują najczęściej w widoku komponentu (ale mogą być również zastosowane w TS)
- Pipes służą do transformacji wartości dostarczanej do pipe.
- Angular posiada zestaw wbudowanych pipes dla różnych formatów danych, np. Date, UpperCase, Currency, Decimal, Json, KeyValue, Async
- Pipe mogą przyjmować dodatkowe argumenty w celu modyfikacji działania (np. datetime pozwala zdefiniować format daty)
- Pipe mogą korzystać z globalnych DI Tokens (np. timezone dla Date)
- Składnia: {{ value | pipename:params | anotherpipe:params }},

WSEI

Pipes - własny pipe

- Tworzenie customowego pipe: **ng g p pipename**

WSEI

Dyrektywy

- Dyrektywa to klasa (z dekoratorem @Directive) której zadaniem jest określona modyfikacja drzewa DOM (struktury i/lub atrybutów)
- Dyrektywa zawsze posiada swój selektor (podobnie jak komponent posiada nazwę znacznika).
- Najpopularniejsze typy selektorów: nazwa znacznika, atrybutu, atrybut z określoną wartością
- Hostem dla dyrektywy może być komponent lub inna dyrektywa

WSEI

Dyrektywy

Rodzaje dyrektyw:

- Najpopularniejsza: dyrektywa komponentu:)
  Tak naprawdę klasa komponentu dziedziczy po dyrektywie
- Dyrektywy strukturalne, np: *ngIf, *ngFor, *ngSwitch, *ngTemplateOutlet
  Dyrektywy strukturalne zmieniają strukturę DOM
  Ograniczenie: jedna dyrektywa strukturalna na jeden znacznik
- Dyrektywy niestrukturalne: ngStyle, ngClass, ngModel
  Dyrektywy niestrukturalne modyfikują wygląd/działanie hosta

WSEI

Dyrektywy - własne dyrektywy

- Tworzenie dyrektywy: ng g d directiveName

WSEI

Formularze - Template Driven Forms

- Obsługa formularzy oraz walidacja oparta o dyrektywy w widoku
- Wartości przypięte bezpośrednio do właściwości klasy poprzez two-way binding
- Model formularza może się zmieniać w zależność od dyrektyw zastosowanych w widoku HTML
- Utrudniona kontrola nad zdarzeniami formularza - np. zmiana danych, walidacja danych
- Stosowane zazwyczaj do prostych formularzy (np. newsletter)

WSEI

Formularze - Template Driven Forms

```
<form (ngSubmit)="onSubmit()" #loginForm="ngForm">
  <input type="text" [(ngModel)]="login" />
  <input type="text" [(ngModel)]="password" />
  <button type="submit" [disabled]="!loginForm.form.valid">Login</button>
</form>
```

```
@Component({
  //...
})
export class LoginFormComponent {
  login: string = ''
  password: string = ''
  onSubmit() { }
}
```

WSEI

Formularze - Reactive Forms

- First-choice w pracy z formularzami w Angularze
- Modelowanie formularza następuje w kodzie TS, HTML jedynie odzwierciedla model
- Dowolnie skomplikowany formularz jest tworzony z użyciem czterech obiektów: FormControl, FormArray, FormGroup, FormRecord
- Do budowania modeli formularzy jest przeznaczona dodatkowa klasa FormBuilder
- Pełna, reaktywna kontrola nad wartościami i zdarzeniami formularza
- Może być zastosowany do każdego formularza

WSEI

Routing - podstawy

- Routing definiujemy na poziomie modułów
- Drzewo routingu oparte jest o tablice routingu w poszczególnych modułach
- Pojedynczy route może kierować do komponentu, modułu lub innego route-a
- W ścieżkach rutingu definiujemy „wzorzec” url-a, np: /invoice/:id/edit
- W routingu pierwszy route pasujący do url-a wygrywa (strategia first match win)

WSEI

Routing - podstawy

- Angular obsługuje dwie strategie routingu: HashLocationStrategy i PathLocationStrategy.
  - HashLocationStrategy: https://localhost/#users/list/2022
  - PathLocationStrategy: https://localhost/users/lists/2022 (domyślny)
- Moduł routingu dostarcza dwa pomocnicze serwisy do pracy z routingiem: **Router** i **ActivatedRoute**
- Miejsce „zaczepienia” routingu w kodzie html: **<router-outlet></router-outlet>**
- Link w kodzie html: **<a [routerLink]=„[„user”, 12]”>Profil**

WSEI

Angular Material

Rozbudowana biblioteka komponentów

Zapewnia gotowe szablony/style i unormowany sposób ich zmiany

Pod spodem korzysta z własnego CDK (Component Dev Kit) do budowania komponentów

Bazuje na Materiał Design - zapewnia spójność z formatowaniem całego ekosystemu aplikacji Google

Najpopularniejsza biblioteka gotowych komponentów i formatowania do Angulara

Instalacja w projekcie: **ng add @angular/material**

WSEI

Wyższa Szkoła Ekonomii
i Informatyki w Krakowie