**Raport kwartalny**

**Przygotowała**: Dominika Łoś

**Stanowisko**: Programista

**Dział**: Zespół Oprogramowania Paczkomatu

**Data**: 21.09.2020r.

**Odbiorca**: Orlan Gansiniec

**Stanowisko**: Kierownik Zespołu Testów Aplikacji Operacyjnych

**Dział**: Oprogramowania i Testów / Zespół Testów Aplikacji

**Raportowany** **okres**: 01.07.2020-30.09.2020r.

Niniejszy raport poświęcony jest przekazaniu informacji na temat stanu posiadanej wiedzy z frameworka Angular.

1. **Co to jest Angular, główne wady i zalety.**
2. **Elementy budowy projektu.**
   1. Co to jest komponent.
      1. Do czego służy każdy z 4 plików komponentu.
   2. Serwisy
3. **Elementy natywne frameworka Angular**
   1. @Input
   2. @Output
   3. <router-outlet>
   4. ngFor
   5. ngIf
   6. ngSwitch
   7. Pipes
   8. httpClient

**1. Co to jest Angular, główne wady i zalety.**

Angular to otwarty Framework (szkielet do budowy aplikacji), rozwijany i wspierany przez Google, napisany w języku TypeScript.

**Zalety:**

* Wieloplatformowość – można napisać wiele rodzajów aplikacji np. strony internetowe, aplikacje webowe, aplikacje mobilne, aplikacje desktopowe.
* Bogactwo własnych komponentów
* Umożliwia budowę aplikacji bez korzystania z dodatków
* Obszerna dokumentacja
* Ciągłe wsparcie

**Wady:**

* Do budowy samodzielnej aplikacji trzeba znać przynajmniej podstawy z HTML, CSS i przede wszystkim TypeScript
* Wyższy próg wejścia niż przy innych, podobnych frameworkach
* Duży rozmiar Frameworka
* Wydajność

**2. Elementy budowy projektu.**

Projekt można podzielić na kilka elementów:

**Folder App:**

* Komponenty (defaultowo jest zaimplementowany jeden – app.component)
* Serwisy (defaultowo brak)
* app.module.ts – najważniejszy plik aplikacji. To tutaj są załączane wszystkie komponenty, serwisy, moduły składające się na aplikację.
* app-routing.module.ts – ustawianie routingów dla poszczególnych komponentów

**Folder Assets**

* Tu powinny znajdować się wszelkie multimedia, aby zachować strukturę projektu.

**Folder Environments**

* To tutaj jest możliwa modyfikacji trybu developerskiego (environment.ts) i produkcyjnego aplikacji (environment.prod.ts).

**2.1 Co to jest komponent.**

Element, który ma swój własny widok i „zaplecze” czyli kod odpowiedzialny za wysyłanie informacji na widok i ewentualne przetwarzanie ich.

Każdy komponent po stworzeniu składa się z 4 elementów:

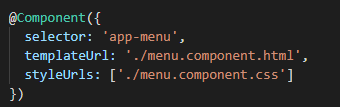
* Name.component.ts
* Name.component.css
* Name.component.html
* Name.component.spec.ts

**2.1.1 Do czego służy każdy z 4 plików komponentu.**

* **Name.component.ts**

Plik odpowiedzialny za połączenie trzech cześci:

Name.component.ts, Name.component.css i Name.component.html za pomocą dekoratora @Component



Selector odpowiada za ustawienie taga, dzięki któremu można wywołać dany komponent z kodu HTML innego komponentu.

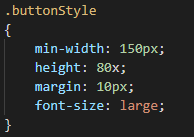
TemplateURL łączy kod HTML z kodem zawartym w pliku Name.component.ts

styleUrls łączy plik ze stylowaniem z plikiem Name.component.ts .

W tym pliku tworzona jest cała logika potrzebna do wysyłania bądź odbierania informacji z widoku (Name.component.html)

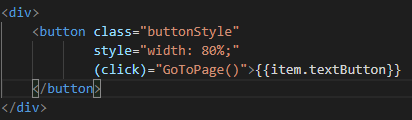
* **Name.component.css**

Plik w którym można zawrzeć stylowanie dla komponentu i wykorzystać na widoku. Np.



* **Name.component.html**

Widok komponentu – czyli to jak widzi go użytkownik końcowy. Np.



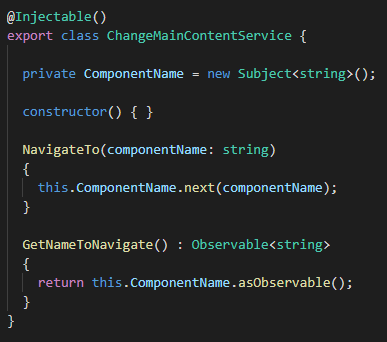
* **Name.component.spec.ts**

Plik w którym pisane są testy dla danego komponentu.

**2.2 Serwisy**

Służą do komunikacji pomiędzy komponentami. Mogą służyć do przesyłania stałych danych, jako pośrednik pomiędzy zewnętrznym API czy do nasłuchiwania zdarzeń emitowanych przez inne serwisy.

Przykład prezentujący serwis, służący do przesyłania, pomiędzy komponentami, informacji na temat tego, na jaki komponent powinna się podmienić główna część aplikacji.



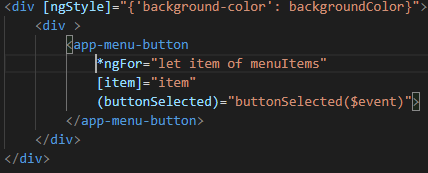
**3. Elementy natywne frameworka Angular**

**3.1 @Input**

Menu-Button.component.ts:

****

Menu.component.html:



Dekorator pozwalający na przesyłanie obiektów pomiędzy komponentami, które są ze sobą połączone. Np. Menu-> Menu-button. Jest to przesyłanie tylko w jednym kierunku – z rodzica do dziecka.

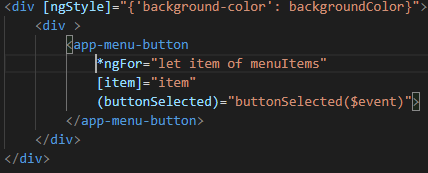
**3.2 @Output**

Menu-Button.component.ts:





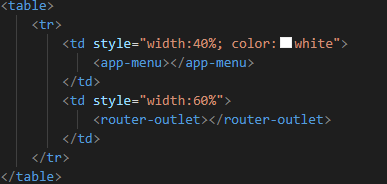
Menu.component.html:



Działa odwrotnie do @Input. Za pomocą EventEmittera pozwala przesyłać dane z dziecka do rodzica.

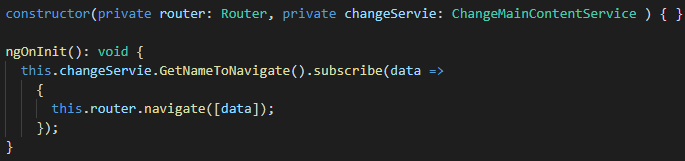
**3.3 <router-outlet>**

Atrybut, który pozwala przełączać się pomiędzy komponentami.



W miejscu, którym został użyty router-outlet będzie zmieniać się kontent.

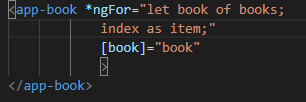
Należy jednak pamiętać o ustawieniu odpowiednich ścieżek w app-routing.module.ts i wykorzystać routing w komponencie do przeskakiwania pomiędzy komponentami. Np.



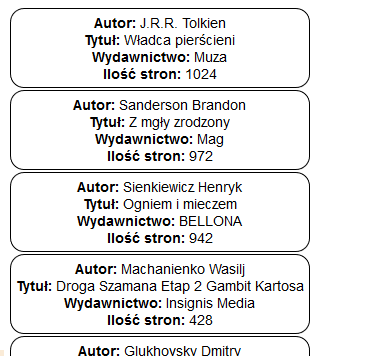
**3.4 ngFor**

Pętla wykorzystywana na widoku. Pozwala działać na pojedynczym elemencie z listy i go modyfikować. Np. dodawać stylowanie w zależności od tego czy element jest pierwszy, ostatni, czy ma indeks parzysty lub nieparzysty.

Kod:



Wynik:



**3.5 ngIf**

Instrukcja warunkowa. Pozwala określić czy coś ma zostać zrobione bądź nie.

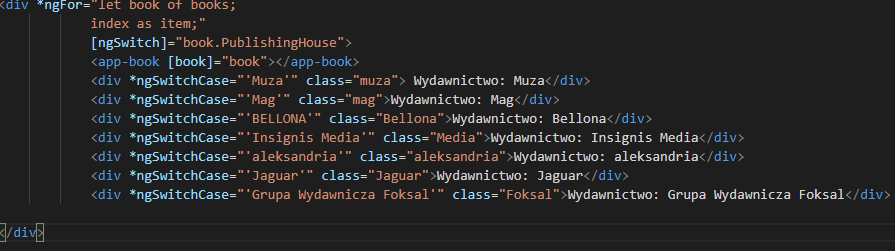
\*ngIf="book"

Kod ten oznacza ze jeśli obiekt book będzie undefined lub null nie zostanie wyświetlony na widoku.

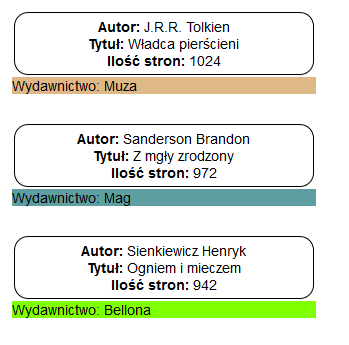
**3.6 ngSwitch**

Pętla wykorzystywana na widoku. Pozwala działać na pojedynczym elemencie z listy i modyfikować np. stylowanie dla danego elementu.

Kod:



Wynik:

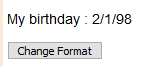


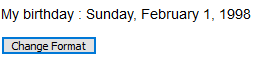
**3.7 Pipes**

Pipes pomagają transformować otrzymywane dane(stringi, daty, liczby itd.).

Podano przykład dla daty:







**3.8 httpClient**

W skrócie: wykorzystanie httpClient pozwala wysyłać zapytania do serwera i otrzymywać z niego odpowiedzi.

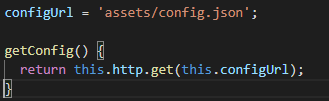
Aby z niego skorzystać potrzebujemy serwisu, w którym będą wywoływane metody np. GET, POST, DELETE itd. Następnie trzeba metody z serwisu subskrybować w komponencie, aby zostały wywołane.

Dodatkowo należy pamiętać, że aby móc korzystać z httpClient, klasa ta powinna zostać wstrzyknięta do konstruktora serwisu.

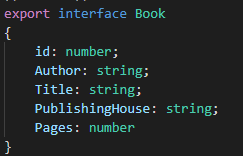


Przykładowe zapytania do serwera:

Get

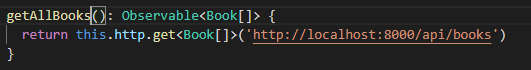


Aby móc odebrać konkretny typ ze strony np. Book, potrzebujemy interfejs po stronie Angulara zbudowany dokładnie tak samo jak typ Book po stronie serwera.



Dzięki temu można wysyłać bardziej sprecyzowane zapytania do serwera:

GetAllBooks:



W zależności od tego jak są zbudowane metody po stronie serwera można wysyłać zapytania o konkretny element:



