Wprowadzenie do aplikacji Internetowych

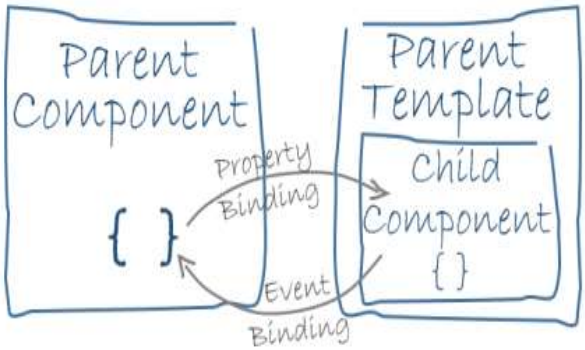
laboratorium 4

Cel zajęć:

Zapoznanie z frameworkiem Angular oraz pojęciem komponentu. Tematem przewodnim dzisiejszego laboratorium jest komponent zarówno pojedynczy jak i komunikacja pomiędzy komponentami.

Wyróżniamy 3 różne przypadki:

- komunikacje rodzic –dziecko
- komunikacje dziecko – rodzic
- komunikacje pomiędzy niepowiązanymi komponentami (realizowane za pomocą serwisów).



Cześć zadań dotyczyć będzie jednego tematu przewodniego rozwijanego przez najbliższe 3 lab. Tematem rozwijanej aplikacji będzie biuro turystyczne. Użytkownicy będą mogli rezerwować udział w wycieczkach dostępnych na stronie biura. Celem aplikacji jest możliwość przeglądnięcia oferty biura dań (pod kątem różnego rodzaju kryteriów) z możliwością oceny wybranej wycieczki oraz zastawienia komentarza na temat wycieczki. Dodatkowo będzie możliwość przeglądania zawartości oraz zakupu wycieczki jako osoba zalogowana – realizacja rejestracji i logowania. Tylko admin będzie miał możliwość dodawania, usunięcia i edycji wycieczki. Backend i autentykacja oparte na Firebase.

Będą istniały warianty realizacji aplikacji:

W wersji łatwiejszej - możliwość oceniania wycieczki przez wszystkich użytkowników

W wersji trudniejszej tylko przez osoby, które kupiły wycieczkę i ja zrealizowały + możliwość zastawiania komentarzy.

Jeśli chodzi o architekturę rozwiązania to przewiduje realizacje projektu w dwóch wersjach:

W wersji trudniejszej - możliwość samodzielnego napisania serwera REST API (ExpressJS) w oparciu o materiały z wykładu i ewentualne „lekkie" podpowiedzi.

W wersja łatwiejsza – backend w całości oparty na Firebase.

Aplikacja będzie realizowana etapami:

Lab 4 – przygotowanie funkcjonalności Frontendu dostępnej dla wszystkich

Lab 5 - obsługa backendu – serwer aplikacyjny + persystencja danych w bazie danych

Lab 6 - wprowadzenie autentykacji (logowanie) oraz obostrzenie dostępu do wybranych funkcji systemu tylko dla osób zalogowanych o odpowiednich preferencjach (np. admin).

Rozwój aplikacji będzie sterowany kolejnymi zadaniami, których celem jest jej ewolucyjny rozwój.

Zadania na rozgrzewkę (bez oceniania).

Zadanie 1. Weryfikacja środowiska pracy

Sprawdź czy masz na swoim komputerze zainstalowane następujące oprogramowanie:

Npm: npm –v

Zapoznaj się z krótkim tutorialem dotyczącym instalacji i używania Angulara

https://angular.io/guide/quickstart. Na jego podstawie sprawdź czy masz zainstalowanego Angulara.

ng version lub ng -v

W przypadku gdyby nie było zainstalowanego angulara proszę o jego instalacje. Zapoznaj się z możliwościami środowiska CLI - https://cli.angular.io/

Zadanie 2. Podstawowym językiem tworzenia oprogramowania w Angularze jest

TypeScript. Przeglądnij tutorial dotyczący TypeScript

https://www.typescriptlang.org/docs/tutorial.html

Zapoznając się z najważniejszymi zasadami i konstrukcjami języka.

Zadanie 3. Stwórz swój pierwszy projekt w Angularze. Zapoznaj się ze strukturą projektu. Stwórz swój pierwszy komponent w Angularze wypisujący informacje o ulubionym Aktorze/Aktorce. Dane aktora: imię, Nazwisko oraz Tytuł ulubionego filmu podaj za pomocą pola/pól typu input.

Wykorzystując interpolacje zweryfikuj wynik działania następujących operacji:

{{2 +2}}

{{ imie.length}}

{{ imie.toUpperCase()}}

{{ a = 2+3}}

{{ windows.location.href}}

Wyciągnij wnioski z otrzymanych wyników i ewentualnie tak zmodyfikuj kod aby zadziałał.

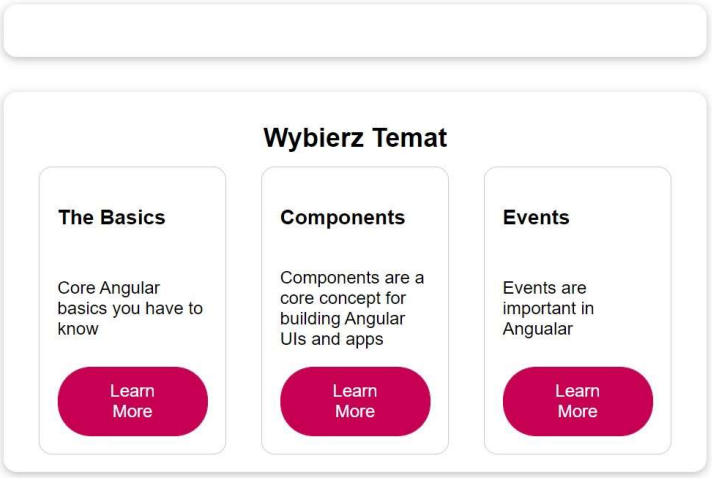
Zadania punktowane. (20 pkt)

Zadanie 5. (1 pkt)

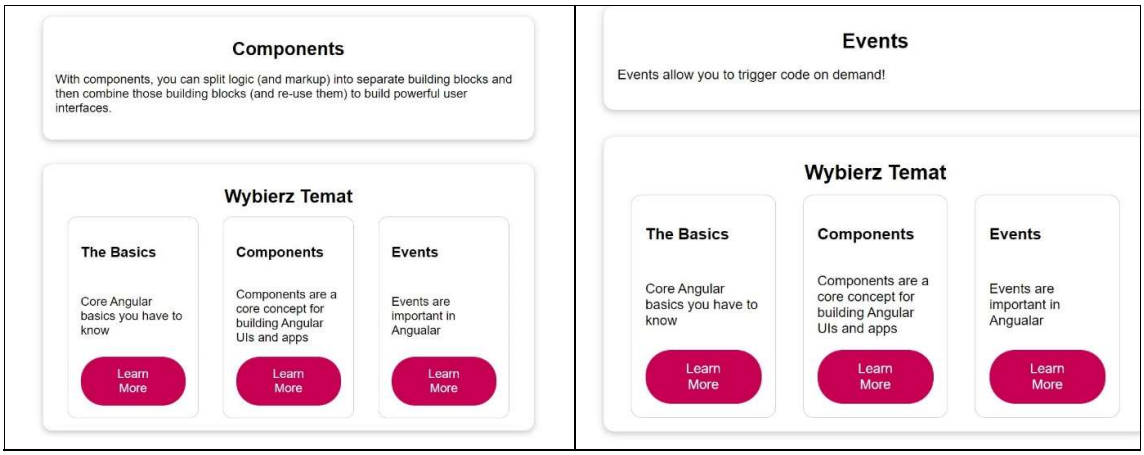
Wybór samochodu. Model danych składa się z kolekcji samochodów. Każdy samochód to marka, model oraz lista dostępnych kolorów. W kolekcji może być kilka modeli tej samej marki. Stwórz aplikacje pozwalająca na wybór pojazdu. Najpierw wybierasz z listy markę, która Cię interesuje. Na tej podstawie druga lista rozwijana wyświetla tylko listę modeli, które są dostępne dla wybranej marki. Wybierz teraz z tej listy model. Dla wybranego pojazdu niech wyświetli się lista dostępnych kolorów – w postaci np. kwadratów o odpowiednim kolorze. Wybór któregokolwiek powinien wyświetlić pełną informacje o wybranym samochodzie tzn. Model, marka, kolor oraz szczegóły wyposażenia. Niech dane będą znajdowały się w pliku JSON, a nie zaszyte w kodzie.

Zadanie 6. (2pkt)

Stwórz aplikację, która pozwala na wybór jednego z tematów dotyczących Angulara i wyświetlenie go w przygotowanym komponencie wynikowym. Przykład inicjalny poniżej:



Po naciśnięciu learn more wybrany temat jest ładowany do komponentu wynikowego -> patrz niżej



Zadanie 7. Restauracja - Lista dań **(6 pkt)**

Stwórz nowy projekt a w nim nowy komponent/komponenty reprezentujące Wycieczki. Zmodyfikuj tak kod aby nowy komponent był komponentem wyświetlanym na starcie naszej aplikacji. Komponent Wycieczki powinien wyświetlać listę wycieczek. Lista wycieczek powinna być zdefiniowana w pliku projektu zwierającym tylko fake data lub w zewnętrznym pliku json. **(1pkt)**

Pojedynczy obiekt Wycieczki powinien zawierać następujące pola: Nazwa, docelowy kraj wycieczki, data rozpoczęcia i zakończenia wycieczki, cena jednostkowa, max ilość miejsc, krótki opis wycieczki oraz link do poglądowego zdjęcia.

Stwórz przynajmniej 10 obiektów i użyj ich w komponencie.

Wyświetl zawartość tablicy obiektów w szablonie komponentu głównego - dyrektywa *ngFor (każda element odpowiednio wystyluj). Wyświetlane zdjęcia wycieczki proszę wyświetlać jako okrągłe. Przy każdym produkcie powinny znajdować się 2 przyciski + i - pozwalające na rezerwacje miejsca na wycieczkę lub rezygnacje z wycieczki (przycisk -).

Jeśli ilość wolnych miejsc wycieczki znajdującej się w tablicy będzie wynosiła 0 to należy wyświetlić inny komunikat niż gdy ilość dostępnych miejsc jest większa od 0.

Podobnie przy rezygnacji – jeśli ilość dostępnych wycieczek jest równa max ilości to nie powinno być możliwości zwrócenia wycieczki. **(1pkt)**

W przypadku gdy ilość miejsc spadnie do zera przycisk + powinien zostać ukryty. Nie chcemy przecież rezerwować wycieczki na której już nie ma miejsca. – dyrektywy ngStyle lub ngClass. Gdy ilość wolnych miejsc będzie zbliżała się do 0 (np. od 3 w dół) należy zaznaczyć to w sposób graficzny np. inne tło, kolor czcionki, wielkość fontów lub inny wizualny sposób. **(1pkt)**

Podobnie należy rozróżnić wycieczki o najniższej cenie jednostkowej oraz najwyższej – za pomocą dodatkowego obramowania obejmującego dany produkt - zielone – najdroższy, czerwone najtańszy. **(1pkt)**

Wypisz nazwę wycieczki oraz kraj dużymi literami -> skorzystaj z odpowiedniego typu pipe. Wyświetl cenę wycieczki wraz z nazwa (lub znakiem płatniczym) skojarzonym z walutą np. USD - $, euro lub złotówki.

Wyświetl również sumaryczną ilość aktualnie zarezerwowanych wycieczek - jeśli wynosi on więcej niż 10 ma być wyświetlana na zielonym tle, jeśli poniżej 10 na czerwonym tle. **(1pkt)**

Uwaga!! Oceniać będę również zaproponowaną architekturę rozwiązania. Powinna być zwinna i elastyczna – pamiętajmy o zasadzie SOLID. **(1pkt)**

Zadanie 8. Usuwanie wycieczki **(1pkt)**

Rozszerz funkcjonalność aplikacji o możliwość usuwania wycieczki. Zrealizuj tą funkcjonalność poprzez dołożenie przycisku Usun obok wycieczki. Naciśniecie tego przycisku powinno skutkować usunięciem wycieczki z listy wycieczek.

Zadanie 10. Dodawanie wycieczki **(2pkt)**

Skoro jest możliwość usuwanie wycieczki z listy, niech będzie także dostępna możliwość dodawania nowej wycieczki. Dodawania odbywa się za pomocą formularza – sugeruje zastosowanie formularza typu Model Driven Forms. Na razie podobnie jak z usuwaniem wycieczki dostęp do tej funkcjonalności będą mieli wszyscy użytkownicy – potem tylko z odpowiednimi uprawnieniami. Zastosuj mechanizm walidacji.

Zadanie 11. Ocena wycieczki **(1+1pkt)**

Rozszerzmy funkcjonalność pojedynczej wycieczki o możliwość oceniania atrakcyjności wycieczki przez klienta (np. wybór ilość gwizdek lub jakaś inna interesująca forma oceniania). Na razie oceniać wycieczkę będzie mógł każdy klient. Później po wprowadzeniu autoryzacji tylko osoba która kupiła i zrealizowała wycieczkę. Docelowo funkcjonalność ta będzie dostępna tylko z poziomu karty poszczególnej wycieczki. Preferowane samodzielna realizacja oceny bez korzystania z gotowych bibliotek**. (extra 1pkt)**

Zastanów się w jaki sposób zrealizujesz ocenę (oddzielny komponent? a może tylko atrybut komponentu Wycieczki?)

Zadanie 12. – filtrowanie listy wycieczek

Tworzymy dodatkowy komponent służący do filtrowania wyświetlanych wycieczek. Kryteriami po których możemy filtrować są: lokalizacja wycieczki, cena (zakres), data (zakres), ocena. Proponuje do realizacji tej funkcjonalności zastosowanie samodzielnie zdefiniowanych potoków. Sposób realizacji opisany w sekcji poniżej **(1 pkt)**

Wersja rozszerzona

Możliwość wyboru kilku wartości dla danego kryterium np. wybór kilku lokalizacji z listy dostępnych, lub oceny 4 i 5 gwiazdek.

Kryteria filtrowania można łączyć tzw. przykładowe kryteria filtrowania: interesują mnie wycieczki o ocenie 3 i 4 gwiazdki odbywające się w takim a taki terminie. Wyniki filtrowania powinny być dostępne online już podczas wyboru wartości w filtrze**. (1 pkt)**

Filtry zwierają tylko wartości dostępne w liście wycieczek - dotyczy np. zakresu cen. Nie od 0 do 10000 tylko od dostępna cena minimalna do cena maksymalna – wartości powiązane z aktualnymi wynikami filtrowania**. (1 pkt)**

Zadanie 13 Koszyk **(1 pkt)**

Stwórz nowy komponent, niepowiązany z pozostałymi zawierający informacje o wybranych wycieczkach, ich ilości oraz sumie całego zamówienia. Jej zawartość będzie powiązana z lista wycieczek. Jeśli dodaje wycieczkę pozycje w koszyku rosną, jeśli usuwam maleją.

Extra **(2pkt)** za ocenę estetyczna rozwiązania.

---------- Potoki własne – implementacja przykładowa ----------

Angular pozwala tworzyć własne potoki . Wymagane jest aby:

- użyć dekorator @Pipe z metadanymi potoku, wśród których jest własność name. Ta wartość zostanie wykorzystana do wywołania potoku
- Implementować metodę transformacji interfejsu PipeTransform. Ta metoda pobiera z potoku wartość oraz zmienną liczbę argumentów dowolnego typu i zwracają wartość przekształconą (piped).

```
// Przykładowa implementacja potoku typu wyszukaj po podanym tekście

@Pipe({ name: 'searchPipe' })

export class SearchPipe implements PipeTransform {

    transform(courses: Course[], searchText: string): Course[] {

        if (!courses)

            return [];

        if (!searchText)

            return courses;

        searchText = searchText.toLowerCase();

          return courses.filter(course => {

              return course.name.toLowerCase().includes(searchText);

          });

    }
```

Użycie zdefiniowanego potoku w szablonie komponentu. Każdy parametr rozdzielany

dwukropkami w szablonie odwzorowuje jeden argument metody w tej samej kolejności

```
<div>
<div *ngFor="let p of (getCourses() | searchPipe : search )">
        ……….
</div>
</div>
```