Fundusze Europejskie
Wiedza Edukacja Rozwój

Politechnika Wrocławska

Unia Europejska
Europejski Fundusz Społeczny

„ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej”

Politechnika Wrocławska

Wydział Informatyki i Zarządzania

Zaawansowane Technologie Webowe

Laboratorium

Temat:	Opracowanie szablonu Single Page z Bootstrap 4 oraz Material Design
Opracował:	mgr inż. Piotr Jóźwiak
Data:	lipiec 2020
Liczba godzin:	2 godziny

Table of Contents

Wstęp

Na kolejny laboratorium postaramy się zgłębić nowoczesne techniki projektowania stron internetowych z wykorzystaniem frameworków wspomagających wytwarzanie szablonów WWW. Omówimy sobie na przykładzie tworzenia strony typu **Single Page,** proces projektowania strony przy współpracy z **`Bootstrap 4`** oraz biblioteką graficzną **`Material Design`**.

Czym jest Single Page Application?

Single Page Application (SPA) jest aplikacją internetową napisaną w postaci pojedynczej strony, czyli taką, która posiada tylko jeden plik **`html`**. Aplikacja z zasady nie przeładowuje strony w trakcie użytkowania. Do odświeżania części strony wykorzystuje się techniki **`AJAX`**. Natomiast do obsługi logiki strony wykorzystuje się **`JavaScript`**. Jest to bardzo popularny schemat wykorzystywany głównie w prostych stronach internetowych, na których wszystkie informacje można zamieścić na jednej choć całkiem długiej stronie. Menu takiej strony zazwyczaj przesuwa widok strony w odpowiednie miejsce.

Framework Bootstrap

`Bootstrap` to potężny front-endowy framework do szybkiego i łatwego tworzenia stron internetowych. Zawiera szablony projektowe oparte na HTML i CSS do tworzenia najpopularniejszych komponentów interfejsu użytkownika, takich jak formularze, przyciski, nawigacje, listy rozwijane, alerty, modały, tabulatory, akordeony, karuzele, tooltipy, etc.

Bootstrap umożliwia tworzenie elastycznych i responsywnych układów internetowych przy niewielkim wysiłku ze strony programisty.

Bootstrap został pierwotnie stworzony przez projektanta i programistę Twittera w połowie 2010 roku. Zanim stał się strukturą open source, Bootstrap był znany jako Blueprint.

Jeśli mamy już jakieś doświadczenie z dowolnym frameworkiem front-endowym, możemy zastanawiać się, co czyni Bootstrap tak wyjątkowym. Oto kilka zalet, dla których warto wybrać framework Bootstrap:

- **Oszczędność czasu** - oszczędzamy dużo czasu i wysiłku, korzystając ze wstępnie zdefiniowanych szablonów i klas Bootstrap, przez co możemy skoncentrować się na innych pracach programistycznych.
- **Responsywność** - za pomocą Bootstrap możemy łatwo tworzyć responsywne strony internetowe, które będą wyświetlane odpowiednio na różnych urządzeniach i rozdzielczościach ekranu bez żadnych zmian w znacznikach.
- **Standaryzacja** - wszystkie komponenty Bootstrap mają te same szablony i style projektowania pochodzące z centralnej biblioteki, dzięki czemu projekt i układ stron internetowych jest spójny.
- **Łatwość użycia** - Bootstrap jest bardzo łatwy w użyciu. Każdy, kto ma podstawową znajomość HTML, CSS i JavaScript, może rozpocząć tworzenie z Bootstrap.
- **Kompatybilność z przeglądarkami** - Bootstrap jest tworzony z myślą o nowoczesnych przeglądarkach internetowych i jest kompatybilny ze wszystkimi nowoczesnymi przeglądarkami, takimi jak Chrome, Firefox, Safari, Internet Explorer itp.
- **Open Source** - najlepsze jest to, że pobieranie i korzystanie z niego jest całkowicie bezpłatne.

Material Design

Material Design jest prostą biblioteką dostarczającą styl graficzny. Sama biblioteka została stworzona przez Google, które chciało dać projektantom stron internetowych łatwy do dostosowania i zestandaryzowany sposób tworzenia styli graficznych. Po raz pierwszy material design zaprezentowano podczas premiery aplikacji Google Now, następnie stopniowo był on wdrażany także w inne aplikacje wydawcy. W 2014 roku Material Design ogłoszono oficjalnym stylem produkowanych przez Google aplikacji mobilnych, między innymi takich jak Gmail, Google Drive, Google Docs. Szybko także inni twórcy zaczęli wykorzystywać jego założenia podczas tworzenia swoich projektów.

Dzisiaj, Material Design to nie tylko aplikacje Google, ale globalny styl, który swoją prostotą, użytecznością i łatwą aplikacyjnością zdobywa coraz szersze rzesze zwolenników i z powodzeniem wykorzystywany jest także na stronach internetowych.

Najwięcej o Material Design powiedzieć nam może jego oficjalna strona internetowa, znajdziemy na niej podstawowe wytyczne oraz założenia stylu, opisane głównie w oparciu o projektowanie aplikacji mobilnych, z łatwością można je jednak przełożyć także na projektowanie stron internetowych.

Jest to bardzo użyteczna biblioteka graficzna dostarczająca żywe i dobrze dobrane schematy kolorów, proste i intuicyjne ikony, różnego rodzaju grafiki, typografię. Na tym laboratorium ze względu na jego długość nie jesteśmy w stanie omówić wszystkich zawiłości tej biblioteki. Zatem po szczegóły odsyłam do oficjalnej strony.

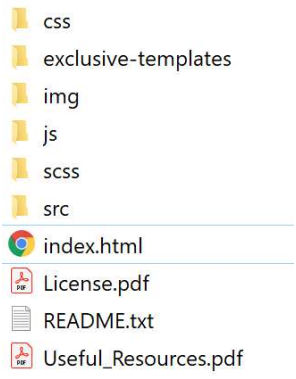
Instalacja oprogramowania

Istnieje wiele sposobów instalacji Bootstrap. Można skorzystać z managera **npm** poleceniem:

```
npm install bootstrap
```

Można też wykorzystać **Content Delivery Network**, które idealnie nadaje się do wykorzystania w produkcyjnej wersji strony poprzez dołączenie poniższego tagu html:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
```



Lub po prostu ściągnąć bibliotekę ze strony producenta: https://getbootstrap.com/.

Jednakże w naszym przypadku chcemy skorzystać oprócz Bootstrapa z biblioteki Material Design, a zatem rozpoczniemy od ściągnięcia pakietu MDB ze strony: https://mdbootstrap.com/docs/jquery/getting-started/download/. W naszym przypadku posłużymy się plikiem zip dla wersji darmowej.

Po jego ściągnięciu należy go rozpakować. Wewnątrz archiwum znajduje się gotowa struktura folderów wraz z bibliotekami Bootstrap, Material Design oraz

jQuery. Przejrzyjmy zawartość folderów **`js`** oraz **`css`**, aby zobaczyć pliki źródłowe bibliotek.

Plik **`index.html`** posłuży nam za bazę do napisania własnego szablonu Single Page Application (SPA).

Opracowanie szablonu strony SPA

Mając już niezbędne biblioteki ściągnięte, możemy przejść do oprogramowania naszego serwisu SPA. Na tej stronie zamieścimy kilka popularnych elementów jakie można spotkać na nowoczesnych stronach. Opracowując po kolei składowe strony przyjrzymy się różnym technikom wykorzystanym do zbudowania szablonu.

Podlinkowanie bibliotek w pliku html

Powiedzieliśmy sobie, że prace będziemy tworzyć w oparciu o już istniejący plik **`index.html`**. Jednakże nie zawsze jest tak, że rozpoczynamy pracę ze skonfigurowanymi bibliotekami. Czasem jesteśmy zmuszeni podłączyć biblioteki do istniejącego projektu. Zatem na wstępie omówmy sobie sposób podłączania/linkowania niezbędnych bibliotek.

Po otwarciu pliku **`index.html`** odnajdujemy w nim w sekcji **`head`** linki do bibliotek:

```
<!-- Bootstrap core CSS -->
<link rel="stylesheet" href="css/bootstrap.min.css">
<!-- Material Design Bootstrap -->
<link rel="stylesheet" href="css/mdb.min.css">
<!-- Your custom styles (optional) -->
<link rel="stylesheet" href="css/style.css">
```

Oraz na końcu sekcji **`body`**:

```
<!-- jQuery -->
<script type="text/javascript" src="js/jquery.min.js"></script>
<!-- Bootstrap tooltips -->
<script type="text/javascript" src="js/popper.min.js"></script>
<!-- Bootstrap core JavaScript -->
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<!-- MDB core JavaScript -->
<script type="text/javascript" src="js/mdb.min.js"></script>
<!-- Your custom scripts (optional) -->
<script type="text/javascript"></script>
```

Pliki bootstrap, mdb oraz jQuery tworzą niezbędne minimum dla strony Material Design. Bardzo istotne jest aby **`css/mdb.min.css`** było podlinkowane przed **`css/bootstrap.min.css`** oraz **`js/jquery.min.js`** przed **`js/bootstrap.min.js`**.

Zwróćmy także uwagę na sekcję wewnątrz tagów **`<body>`** rozpoczynającą się komentarzem:

```
<!-- Start your project here-->
```

To pomiędzy tymi komentarzami będziemy rozwijać nasz szablon.

Struktura strony oraz nawigacja

Korzystając z omówionego wcześniej pliku **`index.html`** przygotujemy najpierw główną hierarchię pliku **html** oraz **navigation bar** na górze strony. Na początek usuńmy starą zawartość strony z pomiędzy komentarza *Start/End your project here*. W to miejsce wkleimy naszą strukturę:

```
<!--Main Navigation-->
<header>

</header>
<!--Main Navigation-->

<!--Main layout-->
<main>

</main>
<!--Main layout-->

<!--Footer-->
<footer>

</footer>
<!--Footer-->
```

Jak widać składa się ona z trzech sekcji: nagłówka, strony zasadniczej oraz stopki.

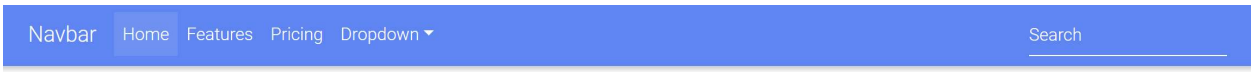
Przejdźmy teraz do opracowania menu oraz paska nawigacji. Skorzystamy tutaj z gotowego przykładu jakich wiele można odnaleźć dla **navbar**: https://mdbootstrap.com/docs/jquery/navigation/navbar/#basic-example. Skopiujmy ten kod i wklejmy go do sekcji **<head>**:

```
<!--Navbar-->
<nav class="navbar navbar-expand-lg navbar-dark primary-color">

  <div class="container">

    <!-- Navbar brand -->
    <a class="navbar-brand" href="#">Navbar</a>

    <!-- Collapse button -->
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#basicExampleNav"
      aria-controls="basicExampleNav" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
```

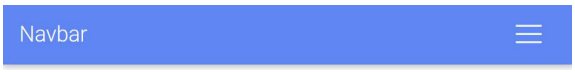
```
    </button>

    <!-- Collapsible content -->
    <div class="collapse navbar-collapse" id="basicExampleNav">

      <!-- Links -->
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#">Home
            <span class="sr-only">(current)</span>
          </a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>

        <!-- Dropdown -->
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" id="navbarDropdownMenuLink" data-
toggle="dropdown" aria-haspopup="true"
            aria-expanded="false">Dropdown</a>
          <div class="dropdown-menu dropdown-primary" aria-
labelledby="navbarDropdownMenuLink">
            <a class="dropdown-item" href="#">Action</a>
            <a class="dropdown-item" href="#">Another action</a>
            <a class="dropdown-item" href="#">Something else here</a>
          </div>
        </li>

      </ul>
      <!-- Links -->

      <form class="form-inline">
        <div class="md-form my-0">
          <input class="form-control mr-sm-
2" type="text" placeholder="Search" aria-label="Search">
        </div>
      </form>
    </div>
    <!-- Collapsible content -->

  </div>
```

```
</nav>
<!--/.Navbar-->
```

Zwróćmy uwagę, że względem podstawowego przykładu został dodany tag ** **. Ma on na celu wycentrować naszą stronę. Zapiszmy plik i zobaczmy rezultat w przeglądarce:



Następnie zmniejszmy okno przeglądarki i zobaczmy, że strona dostosowała się do wąskiego ekranu:



To wszystko otrzymaliśmy za darmo z biblioteki Bootstrap. Oczywiście minie trochę czasu zanim zaczniemy rozumieć znaczenie class styli css zdefiniowanych w bibliotekach. Ale i tak jest szybciej i wygodniej korzystać z opracowanych rozwiązań niż wymyślać je na nowo. Zastanówmy się nad pracą wielkiego zespołu programistów. Gdyby nie powyższa standaryzacja, to próba ogarnięcia różnego podejścia każdego z programistów okaże się bardzo czasochłonna i przysparzająca wiele błędów.

Postarajmy się przeanalizować kod i zmodyfikować go sprawdzając co się zmieni. Spróbujmy także przejrzeć dokumentację do navbar na stronie Material Design.

Następnie dodajmy klasę **.fixed-top** do naszego navbar jak poniżej:

```
<nav class="navbar navbar-expand-lg navbar-dark primary-color fixed-top">
```

Jeśli nie wiemy za co odpowiada ta zmiana, to przekonamy się za kilka chwil. Wrócimy jeszcze do tego, jak zmiana będzie widoczna. Na tę chwilę nie zobaczymy żadnej różnicy.

Landing page full page background

Pora na coś bardziej spektakularnego. Stworzymy tło strony, które będzie zajmować cały ekran na wejście do serwisu.

Dodajmy poniższy kod zaraz za kodem paska nawigacji:

```
<!--Mask-->
<div id="intro" class="view">

  <div class="mask">

  </div>

</div>
<!--/.Mask-->
```

Przyjrzyjmy się znaczeniu poszczególnych elementów:

Klasa **.view** jest wrapperem na obraz tła, który dodatkowo umożliwi nam zdefiniowanie maski. Za pomocą maski możemy przyciemnić lub rozjaśnić obrazek, co zwiększy czytelność tekstu umieszczonego nad obrazkiem.

Klasa **.mask** jest elementem pozycjonowanym absolutnie, co umożliwia nam nałożenie tej warstwy nad obrazek z tła. To na tej warstwie będziemy definiować wcześniej wspomnianą maskę.

Pora na wybranie samego tła. W tym celu poszukajmy obrazu o rozdzielczości przynajmniej 1920px/1280px i wgrajmy go do folderu **img** pod nazwą **bg.jpg**.

Jeszcze wprowadzone zmiany nie powodują, że cokolwiek się zmieniło. Najpierw musimy odpowiednio ostylować nasze klasy. W tym celu w pliku **css/style.css** wklejmy poniższe definicje:

```
html,
body,
header,
#intro {
    height: 100%;
}

#intro {
    background: url("../img/bg.jpg") no-repeat center center fixed;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
}
```

Po tej zmianie powinniśmy zobaczyć już w przeglądarce tło z ustawionym obrazkiem. Ustawiliśmy **height** na 100% dla wszystkich nadrzędnych elementów **#intro** ponieważ jest to jedyny sposób aby pokryć stroną cały ekran. W samej definicji **#intro** wskazaliśmy namiary na plik tła oraz poprzez property **background-size** wstawiając wartość **cover** ustawiliśmy, aby obrazek pokrył cały kontener – całą dostępną przestrzeń.

Kolej na zdefiniowanie maski. Na tę chwilę obrazek nie podlega żadnym modyfikacjom. Jednakże jest on zbyt jasny, aby umieszczenie na nim jakiegoś tekstu było czytelne. Zatem zajmijmy się przykładowym tekstem jak i jego czytelnością.

Dodajmy klasę `.rgba-stylish-strong` do :

```
<div class="mask rgba-stylish-strong">
```

Teraz nasz obrazek tła nie jest już tak jasny. Przeczytajmy dokumentację do definiowania masek tutaj: https://mdbootstrap.com/docs/jquery/css/masks/, aby zobaczyć jak wiele różnych efektów można uzyskać poprzez dodanie zaledwie jednej klasy css. Spróbujmy wstawić inną maskę, np.: `.rgba-red-light`.

Dodajmy tekst do naszego banera. Wewnątrz elementu `.mask .rgba-stylish-strong` wstawmy zawartość:

```
    <!-- Heading -->
    <h2 class="display-4 font-weight-bold white-text pt-5 mb-2">Podróżowanie</h2>

    <!-- Divider -->
    <hr class="hr-light">

    <!-- Description -->
    <h4 class="white-text my-
4">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Deleniti consequuntu
r.</h4>
    <button type="button" class="btn btn-outline-
white">Dowiedz się więcej<i class="fas fa-book ml-2"></i></button>
```

W powyższym przykładzie dodaliśmy Heading, Divider oraz krótki Opis. Przyjrzyjmy się klasom, które zostały wykorzystane.

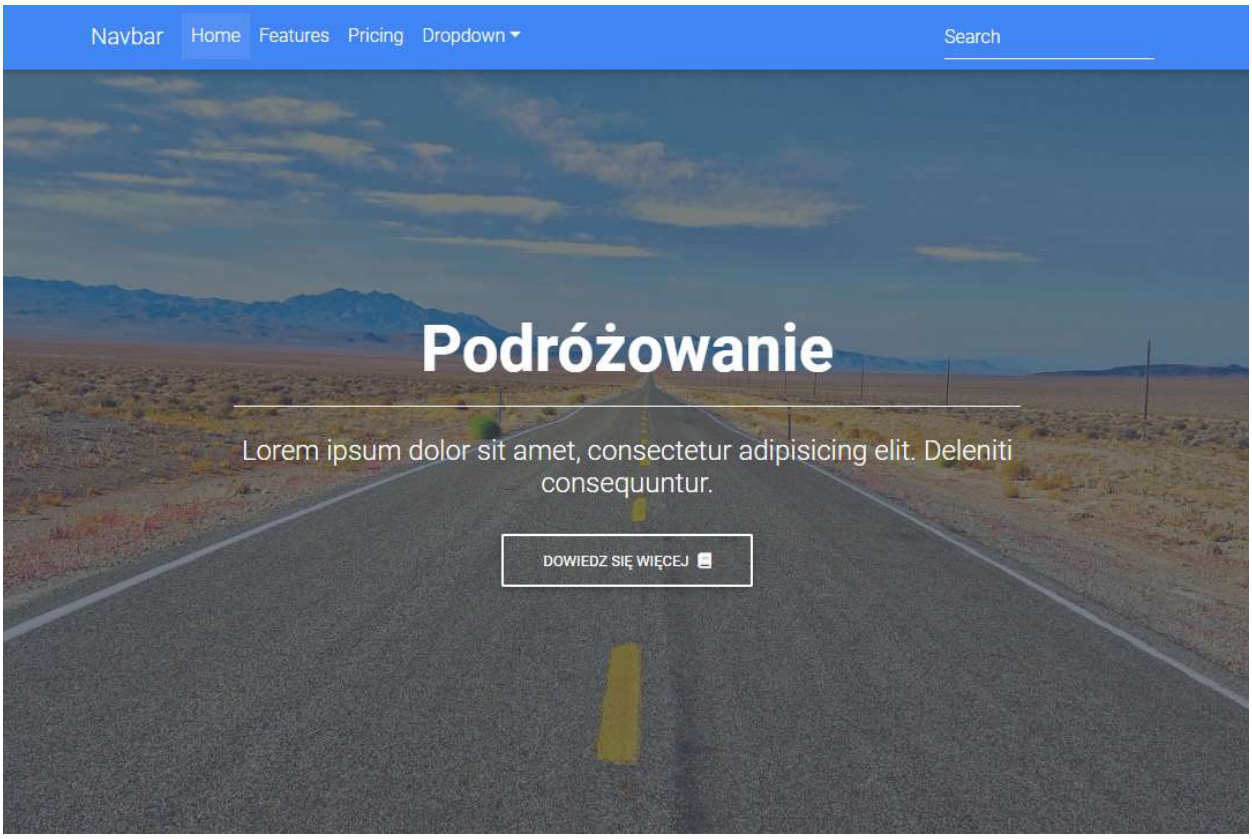
- `.display-4` - tworzy dużej wielkości nagłówek, więcej czytaj tutaj: Typography Docs
- `.font-weight-bold` - pogrubia tekst, czytaj tutaj: Text Utilities Docs
- `.white-text` - ustawia biały kolor, czytaj o kolorach tutaj: Color Docs
- `.btn-outline-white` - tworzy przeźroczysty przycisk, o przyciskach przeczytaj tutaj: Buttons Docs
- `.fa-book` - jest jedną z 700 ikon dostępnych w MDB, o ikonach czytaj tutaj: Icons Docs

Niestety nasz tekst jest wyświetlany z lewej strony. Nie jest to eleganckie. Zatem zajmijmy się teraz wypozycjonowaniem go na środku.

Aby tego dokonać musimy elementy Heading, Divider oraz Description zamknąć wewnątrz poniższego kodu:

```
  <div class="mask rgba-stylish-strong">
    <div class="container-fluid d-flex align-items-center justify-content-
center h-100">
      <div class="row d-flex justify-content-center text-center">
        <div class="col-md-10">

              <!-- Heading -->
              <!-- Divider -->
              <!-- Description -->

        </div>
      </div>
    </div>
  </div>
```

Zdaje sobie sprawę, że konstrukcja wydaje się być skomplikowana, ale zaraz omówimy sobie najważniejsze części. Najpierw sprawdźmy rezultat. Odświeżmy stronę, powinniśmy zobaczyć piękną stronę startową przypominającą tą:



Wykorzystaliśmy klasę `.container-fluid` ponieważ chcemy skorzystać z możliwości jakie nam dostarcza Bootstrap Grid, w którym możemy definiować wiersze oraz kolumny. Jest to jedna z ważniejszych funkcjonalności Bootstrapa, o której można przeczytać więcej tutaj: https://mdbootstrap.com/docs/jquery/layout/grid-usage/ oraz https://mdbootstrap.com/docs/jquery/layout/overview/. Najważniejsze jest zapamiętanie, że wykorzystanie Grid systemu zawsze zakłada użycie konstrukcji: `.container` (lub `.container-fluid`) > `.row` > `column`.

Następnie skorzystaliśmy z klasy `.justify-content-center`, która wypośrodkowuje zawartość horyzontalnie korzystając z Flexbox.

Dodaliśmy `.row` wewnątrz kontenera co jest standardową konstrukcją gridów Bootstrapa. Przy wykorzystaniu Flexboxa (`.d-flex` oraz `.justify-content-center`) ustawiliśmy zawartość `.row` jako wyśrodkowaną.

Wewnątrz `.row` utworzyliśmy kolumnę `.col-md-10`. Jej zadaniem jest ograniczenie szerokości zawartości na dużych ekranach. Dba ona, aby zawartość nie rozszerzała się od krawędzi do krawędzi monitora, co

wyglądałoby brzydko. Ponieważ ta klasa jest responsywna, to wciąż na małych ekranach telefonów zawartość będzie widoczna.

To tyle. Utworzyliśmy stronę startową. Przyjrzyjmy się następnemu elementowi naszej strony.

Sekcja z gridem

Kolejnym elementem naszej strony będzie przygotowanie eleganckiego gridu z 6 kafelkami przedstawiającymi opis ostatnich wypraw.

Zanim rozpoczniemy prace nad tą sekcją to musimy przygotować odpowiednie miejsce dla niej. Odszukajmy w kodzie miejsce:

```
<!--Main layout-->
<main>

</main>
<!--Main layout-->
```

To tutaj będziemy rozwijać kolejne sekcje naszej strony. Zatem zmieńmy ten element na poniższą zawartość:

```
<!--Main layout-->
<main class="mt-5">
  <div class="container">
      <!--Section: LastTravels-->
      <section id="lastTravels">
      </section>
      <!--Section: LastTravels-->

      <hr class="my-5">

      <!--Section: Gallery-->
      <section id="gallery">
      </section>
      <!--Section: Gallery-->

      <hr class="my-5">

      <!--Section: Contact-->
      <section id="contact">
      </section>
      <!--Section: Contact-->

  </div>
</main>
<!--Main layout-->
```

Jak widać w powyższym przykładzie dodaliśmy wiele tagów **<section>** w których rozwijać będziemy kolejne obszary strony. W tym momencie skupimy się na sekcji o id **lastTravels**. Sam tag **<section>** nie ma specjalnego znaczenia, ale został wprowadzony dla czytelności kodu. Zazwyczaj każda sekcja posiada unikalny **id**, aby można było podłączyć ją do linków w menu.

Pomiędzy sekcjami wstawione zostały separatory **<hr>** z ustawioną klasą **my-5**. Klasa ta odpowiada za ustawienie odpowiedniego marginesu pomiędzy sekcjami.

Przygotujmy teraz ramę na nasze kafelki. Będzie ona składała się z dwóch wierszy i trzech kolumn. Wklejmy poniższy kod zamiast **<section id="lastTravels">**:

```
    <!--Section: LastTravels-->
    <section id="lastTravels" class="text-center">
      <!--Grid row-->
      <div class="row">

        <!--Grid column-->
        <div class="col-lg-4 col-md-12 mb-4">

        </div>
        <!--Grid column-->

        <!--Grid column-->
        <div class="col-lg-4 col-md-6 mb-4">

        </div>
        <!--Grid column-->

        <!--Grid column-->
        <div class="col-lg-4 col-md-6 mb-4">

        </div>
        <!--Grid column-->

      </div>
      <!--Grid row-->

      <!--Grid row-->
      <div class="row">

        <!--Grid column-->
        <div class="col-lg-4 col-md-12 mb-4">

        </div>
        <!--Grid column-->
```

```
        <!--Grid column-->
        <div class="col-lg-4 col-md-6 mb-4">

        </div>
        <!--Grid column-->

        <!--Grid column-->
        <div class="col-lg-4 col-md-6 mb-4">

        </div>
        <!--Grid column-->

      </div>
      <!--Grid row-->
    </section>
    <!--Section: LastTravels-->
```

Teraz czas przygotować grafiki. Wgrajmy do folderu **img** sześć obrazków o rozmiarze około 800px/600px. Wykorzystamy je do naszych kafelków. W przykładzie będziemy używać nazw **trip1.jpg** do **trip6.jpg**.

Następnie wstawmy w pierwszym wierszu i pierwszej kolumnie obrazek podróży według schematu:

```
        <!--Grid column-->
        <div class="col-lg-4 col-md-12 mb-4">
          <img src="img/trip1.jpg" class="img-fluid" alt="">
        </div>
        <!--Grid column-->
```

Zwróćmy uwagę, że obrazek ma ustawioną klasę **img-fluid**. Klasa ta powoduje, że obrazek będzie responsywny. Będzie zajmował cały dostępny obszar, który w naszym przypadku został ograniczony przez zewnętrzny kontener ** ** z ustawionymi parametrami Grid systemu bootstrapa. Zapiszmy ustawienia i zobaczmy rezultat w przeglądarce.

Dodajmy teraz efekt, który będzie się pojawiał po kliknięciu na obrazek. Musimy w tym celu zmienić implementację kolumny na:

```
        <!--Grid column-->
        <div class="col-lg-4 col-md-12 mb-4">
          <div class="view overlay z-depth-1-half">
            <img src="img/trip1.jpg" class="img-fluid" alt="">
            <a href="#!">
              <div class="mask"></div>
            </a>
          </div>
```

```
        </div>
        <!--Grid column-->
```

Zwróćmy uwagę, że oprócz dodania linku z maską to nasz obrazek został opakowany w ** ** z klasą `.view`. Material Design domyślnie dla klasy `.view` oraz `.btn` definiuje efekt *Waves*. Kliknijmy teraz na obrazek i zobaczmy rezultat. Więcej o efektach można przeczytać tutaj: Waves Effect Docs.

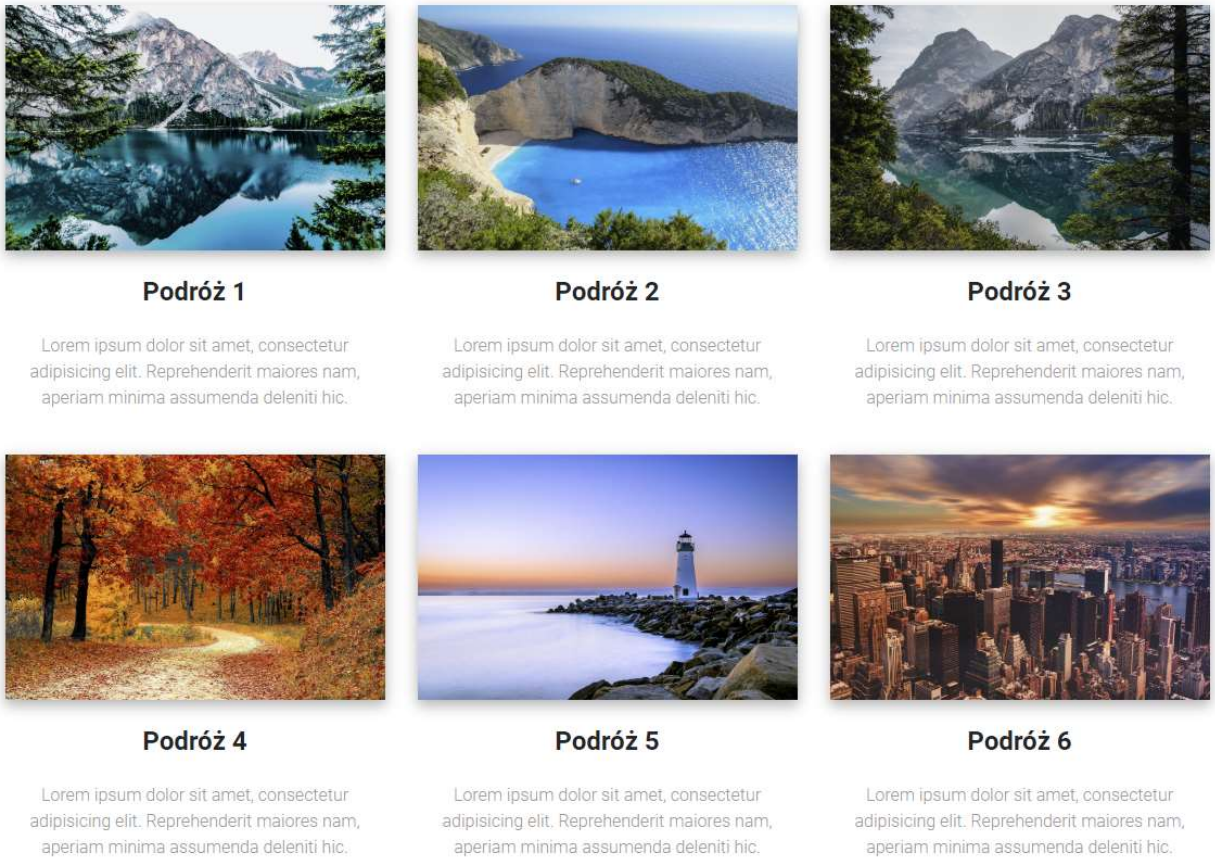
Poprzez dodanie klasy `.z-depth-1-half` włączyliśmy cień pod obrazkiem. Więcej o tej funkcjonalności można znaleźć tutaj: Shadows Docs.

Dodajmy też efekt widoczny po najechaniu kursorem na obrazek. Zapewni to nam link **<a>** zdefiniowany pod obrazkiem, a w nim ** ** z ustawioną klasą `.mask`. Włączenie tej funkcjonalności jest banalnie proste. Wystarczy obok klasy `.mask` dodać kolejną klasę `.rgba-blue-light`. Wykonajmy to i sprawdźmy rezultat. Więcej o tego rodzaju efektach można przeczytać tutaj: Hover Effects Docs.

Pozostało dodać nam krótki opis pod obrazkiem. Zatem końcowy efekt definicji pojedynczego kafelka może przyjąć postać z poniższego przykładu:

```
<!--Grid column-->
<div class="col-lg-4 col-md-12 mb-4">
  <div class="view overlay z-depth-1-half">
    <img src="img/trip1.jpg" class="img-fluid" alt="">
    <a href="#!">
       <div class="mask rgba-blue-light"></div>
    </a>
  </div>
  <h4 class="my-4 font-weight-bold">Podróż 1</h4>
  <p class="grey-text">
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reprehenderit maiores n
am, aperiam minima assumenda deleniti hic.
  </p>
</div>
<!--Grid column-->
```

Wykonajmy podobną definicję dla pozostały kafelków. Efekt końcowy powinien przypominać ten z poniższego obrazka:



Na koniec chciałem jeszcze powrócić do klasy **`.fixed-top`** zdefiniowanej dla naszego menu. Obiecałem, że wrócę do tego. Teraz można przekonać się do czego ona służy. Zwróćmy uwagę, że przesuwając stronę niżej, menu pozostaje przyklejone zawsze na górze. Jest to funkcjonalność nazywana **sticky**. Klasa ta odpowiada za przyklejenie elementu na stałe do góry ekranu.