Fundusze Europejskie Wiedza Edukacja Rozwój
Politechnika Wrocławska
Unia Europejska Europejski Fundusz Społeczny

"ZPR PWr - Integrated Development ProgramWrocław University of Technology"

Wrocław University of Science and Technology

Advanced Web Technologies

Lab

Subject: Single Page template with Bootstrap 4 and Material Design

Prepared by: mgr Piotr Jozwiak

Date: July 2020

Number of hours: 2 hours

Table of Contents

Entry

For the next lab, we will try to explore modern web design techniques using frameworks that support the creation of templatesWWW.We will discuss the example of creating a Single Page page, the process of designing a page in cooperation with Bootstrap 4 and the Material Design graphic library.

What is Single Page Application?

Single Page Application(SPA) is a web application written in the form of a single page, i.e. one that has only one html file. The application, as a rule, does not reload the page during use. AJAX techniques are used to refresh parts of the page. JavaScript is used to handle the logic of the page. This is a very popular scheme used mainly in simple websites where all the information can be placed on one, though quite long, page. The menu of such a page usually moves the view of the page to the right place.

Bootstrap framework

bootstrapis a powerful front-end frameworkfor quick and easy web development. It includes HTML and CSS based design templates for creating the most popular UI components such as forms, buttons, navigations, dropdowns, alerts, modals, tabs, accordions, carousels, tooltips, etc.

Bootstrap allows you to create flexible and responsive web layouts with little effort on your part.

Bootstrap was originally created by a Twitter designer and developer in the mid-2010s. Before becoming an open source framework, Bootstrap was known as Blueprint.

If we already have some experience with any front-end framework, we may wonder what makes Bootstrap so special. Here are some advantages to choose the Bootstrap framework:

- **Time saving**- we save a lot of time and effort by using Bootstrap's predefined templates and classes, so we can concentrate on other development work.
- **Responsiveness**- with Bootstrap we can easily create responsive websites that will be displayed appropriately on different devices and screen resolutions without any changes to the markup.
- **Standardization**- all Bootstrap components share the same templates and design styles from a central library, making the design and layout of web pages consistent.
- **Ease of use**- Bootstrap is very easy to use. Anyone with basic knowledge of HTML, CSS and JavaScript can start developing with Bootstrap.
- **Browser Compatibility**- Bootstrap is made with modern web browsers in mind and is compatible with all modern browsers such as Chrome, Firefox, Safari, Internet Explorer, etc.
- **Open Source**- the best part is that it is completely free to download and use.

Material Design

Material Designis a simple supply librarygraphic style. The library itself was created by Google, which wanted to give web designers an easy-to-customize and standardized way to create graphic styles. For the first time, material design was presented during the premiere of the Google Now application, then it was gradually implemented in other applications of the publisher. In 2014, Material Design was announced as the official style of mobile applications produced by Google, including Gmail, Google Drive, Google Docs. Soon, other artists began to use its assumptions when creating their projects.

Today, Material Design is not only thatGoogle applications, but a global style that is gaining more and more supporters with its simplicity, usability and ease of application and is also successfully used on websites.

He can tell us the most about Material Designofficial website, we will find basic guidelines and style assumptions, described mainly on the basis of designing mobile applications, but they can also be easily translated into website design.

It is a very useful graphics library providing vivid and well-chosen color schemes, simple and intuitive icons, various types of graphics, typography. In this lab, due to its length, we are unable to cover all the intricacies of this library. So, for details, I refer you to the official website.

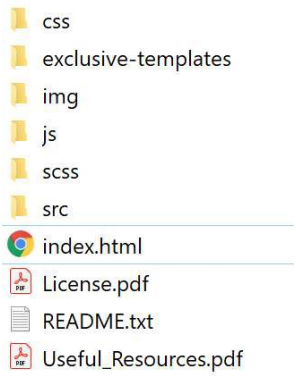
Software installation

There are many waysBootstrap installation. You can use the npm manager with the command:

```
npm installbootstrap
```

You can also use the Content Delivery Network, which is ideal for use in the production version of the site by including the following html tag:

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



Or just download the library from the manufacturer's website:https://getbootstrap.com/.

However, in our case, we want to use the Material Design library in addition to Bootstrap, so we will start by downloading the MDB package from the website:https://mdbootstrap.com/docs/jquery/getting-started/download/. In our case, we will use the zip file for the free version.

After downloading it, unpack it. Inside the archive there is a ready-made folder structure with Bootstrap, Material Design and libraries

jQuery. Let's go through the contentjs and css folders to see library source files.

The index.html file will serve usas a base for writing your own Single Page Application (SPA) template.

Developing a SPA website template

Having already downloaded the necessary libraries, we can go to the software of our SPA service. On this page, we will list some popular elements that can be found on modern websites. By developing the components of the page one by one, we will look at the various techniques used to build the template.

Linking libraries in the html file

We told ourselves that we would create works based on the already existing index.html file. However, it is not always the case that we start working with configured libraries. Sometimes we are forced to connect libraries to an existing project. So first, let's discuss how to connect/link the necessary libraries.

Afteropening the index.html file, we find links to libraries in the head section:

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

And at the end of the body section:

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

Bootstrap, mdb and jQuery files are the bare minimum for a Material Design website. It is very important that css/mdb.min.css is linked before css/bootstrap.min.css and js/jquery.min.js before js/bootstrap.min.js.

Let's also pay attentionto a section inside <body> tags starting with a comment:

```
<!-- Start your projecthere-->
```

It is between these comments that we will develop our template.

Site structure and navigation

Using the discussedindex.html file, we will first prepare the main hierarchy of the html file and the navigation bar at the top of the page. To begin with, let's remove the old page content from between the comment Start/End your project here. Paste our structure here:

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

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

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

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

<!--Footer-->
<footer>

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

As you can see, it consists of three sections: header, main page and footer.

Now let's move on to developing the menu and navigation bar. We will use a ready-made example hereof which many can be found for

navbar:https://mdbootstrap.com/docs/jquery/navigation/navbar/#basic-example.

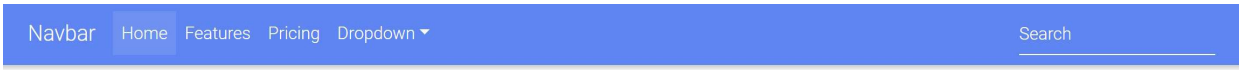
Let's copy this code and paste it into the <head> section:

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

  <divclass="container">

    <!-- Navbar brand -->
    <andclass="navbar-brand"href="#">navbar</and>

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

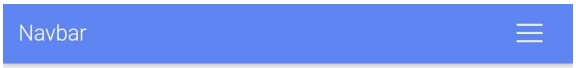
```
    </button>

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

      <!-- Links -->
      <stclass="navbar-nav mr-auto">
        <liclass="nav item active">
          <andclass="nav link"href="#">home
            <spanclass="sr-only">(current)</span>
          </and>
        </li>
        <liclass="nav item">
          <andclass="nav link"href="#">Features</and>
        </li>
        <liclass="nav item">
          <andclass="nav link"href="#">Pricing</and>
        </li>

        <!-- Dropdown -->
        <liclass="nav item dropdown">
          <andclass="nav-link dropdown-toggle"id="navbarDropdownMenuLink"data-
toggle="drop down"aria-haspopup="true"
            aria-expanded="false">Dropdown</and>
          <divclass="dropdown-menu dropdown-primary"aria-
labelledby="navbarDropdownMenuLink">
            <andclass="dropdown item"href="#">action</and>
            <andclass="dropdown item"href="#">Another action</and>
            <andclass="dropdown item"href="#">Something else here</and>
          </div>
        </li>

      </st>
      <!-- Links -->

      <formsclass="form-inline">
        <divclass="md-form my-0">
          <inputclass="form-control mr-sm-
2"type="text"placeholders="search"aria-label="search">
        </div>
      </forms>
    </div>
    <!-- Collapsible content -->

  </div>
```

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

Notice that the tag has been added to the basic example. It's meant to center our page. Let's save the file and see the result in the browser:



Next, let's shrink the browser window and see that the page has adapted to the narrow screen:



We got all this for free from the librarybootstrap. Of course, it will take some time before we start to understand the meaning of css classes defined in libraries. But it is still faster and more convenient to use developed solutions than to invent them anew. Let's consider the work of a large team of programmers. If it wasn't for the above standardization, an attempt to embrace the different approaches of each of the programmers will turn out to be very time-consuming and cause many errors.

Let's try to analyze the code and modify it to see what changes. Let's also try looking at the documentation for navbar on the Material Design website.

Thenlet's add a .fixed-top class to our navbar like this:

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

If we don't know what this change is responsible for, we'll find out in a few moments. We will come back to how the change will be visible. At the moment we will not see any difference.

Landing page full page background

It's time for something more spectacular. We will create a page background that will occupy the entire screen for entering the site.

Let's add the following code right after the navigation bar code:

```
<!--Mask-->
<divid="intro"class="view">

  <divclass="mask">

  </div>

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

Let's take a look at the meaning of the individual elements:

The .view class is a wrapper for the background image, which will additionally allow us to define the mask. Using the mask, we can darken or lighten the image, which will increase the readability of the text placed above the image.

The .mask class is a positioned elementabsolutely, which allows us to apply this layer over the background image. It is on this layer that we will define the previously mentioned mask.

It's time to choose the background itself. To do this, look for an image with a resolution of at least 1920px/1280px and upload it to the img folder under the name bg.jpg.

Changes that have already been made do not change anything. First, we need to style our classes accordingly. To do this, paste the following definitions into the css/style.css file:

```
html,bod
ysuit,he
ader,#in
tro{
    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;
```

After this changewe should see the background with the set image in the browser. We set the height at 100% for all parent#intro elements because it's the only way to cover the entire screen with a page. In the #intro definition itself, we indicated the background file and through the background-size property by inserting the cover value, we set the image to cover the entire container - all available space.

It's time to define the mask. At the moment, the image is not subject to any modifications. However, it is too bright to read any text on it. So let's take a look at the sample text as well as its readability.

Let's add a class`.rgba-stylish-strong`to :

```
<divclass="mask rgba-stylish-strong">
```

Now our background image is not so bright anymore. Let's read the documentation for defining masks here:https://mdbootstrap.com/docs/jquery/css/masks/to see how many different effects can be achieved by adding just one css class. Let's try to insert another mask, e.g.:`.rgba-red-light`.

Let's add text to our banner. Inside the item `.mask .rgba-stylish-strong` let's insert the content:

```
    <!-- Heading -->
    <h2class="display-4 font-weight-bold white-text pt-5 mb-2">Traveling</h2>

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

    <!-- Description -->
    <h4class="white-text my-
4">Lorem ipsum dolor sit amet, consectetur adipisicing elites. Deleniti
consequuntu r.</h4>
    <buttontype="button"class="btn btn-outline-
white">Find out more<andclass="fas fa-book ml-2"></and></button>
```

In the example above, we've added Heading, Divider, and a short Description. Let's take a look at the classes that have been used.

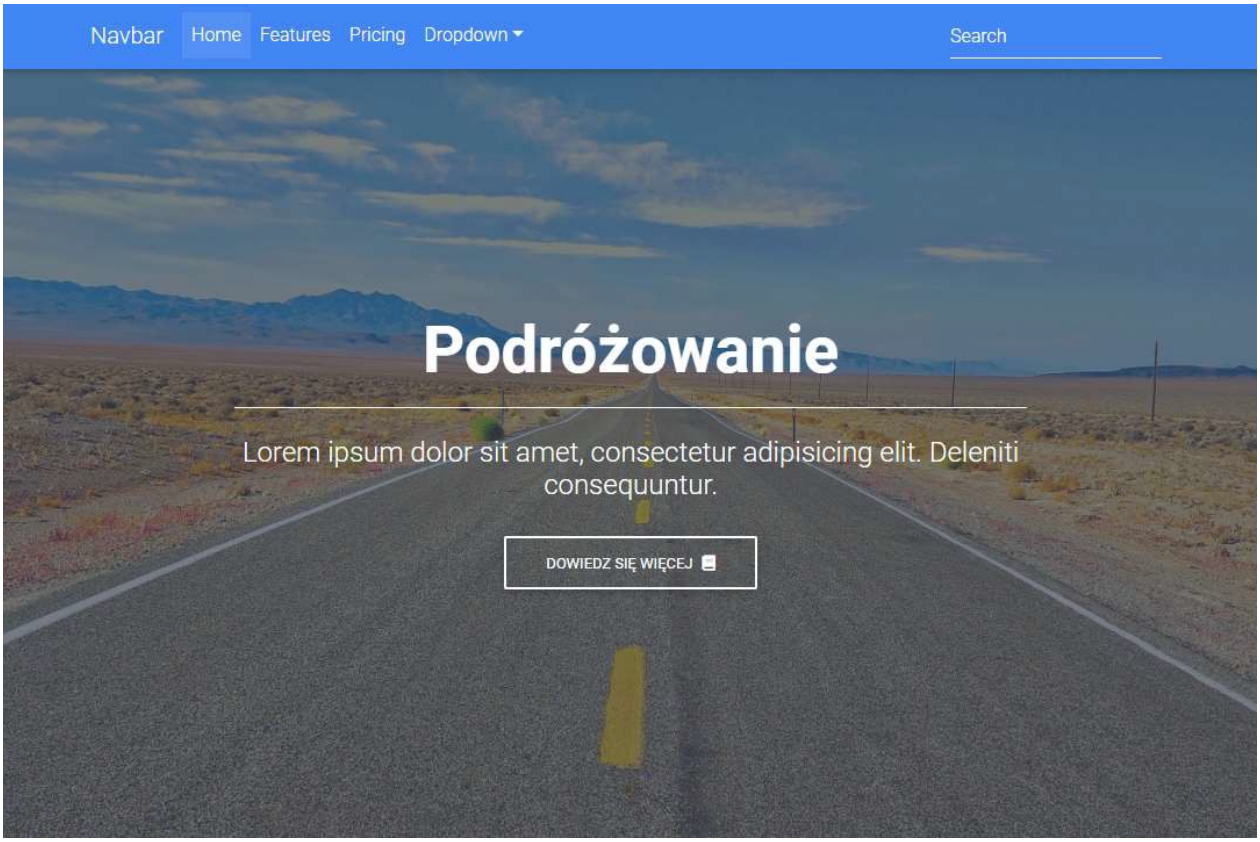
- `.display-4`-creates a large header, read more here: Typography Docs
- `.font-weight-bold`- bold text, read here: Text Utilities Docs
- `.white-text`-sets the white color, read about the colors here: Color Docs
- `.btn-outline-white`- creates a transparent button, about buttonsread here: Buttonsdoc
- `.fa book`-is one of the 700 icons available in the MDB, read about the icons here: Icons Docs

Unfortunately, our text is displayed on the left. It's not elegant. So let's now focus on positioning it in the middle.

To do this, we must enclose the Heading, Divider and Description elements inside the following code:

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

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

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

He realizes that the design seemscomplicated, but we'll cover the most important parts in a moment. Let's check the result first. Let's refresh the page, we should see a beautiful start page similar to this:



We used the class `.container-fluid` because we want to take advantage of the possibilities provided by Bootstrap Grid, in which we can define rows and columns. This is one of the most important features of Bootstrap, which you can read more about here:https://mdbootstrap.com/docs/jquery/layout/grid-usage/ and https://mdbootstrap.com/docs/jquery/layout/overview/. The most important thing to remember is that using the Grid system always implies using the construct: `.container` (or `.container-fluid`) > `.trench`>`column`.

Then we took the class `.justify-content-center`, which centers content horizontally using flexbox.

We added `.trench` inside a container which is a standard Bootstrap grid construct. Using Flexbox (`.d-flex` and `.justify-content-center`) we set the content `.trench` as centered.

Inside `.trench` we have created a column `.col-md-10`. Its purpose is to limit the width of content on large screens. It makes sure that the content does not extend from edge to edge of the monitor, which

it would look ugly. Since this class is responsive, content will still be visible on small phone screens.

That's it. We have created a home page. Let's take a look at the next element of our page.

Grid section

The next element of our website will be the preparation of an elegant grid with 6 tiles presenting the description of recent expeditions.

Before we start working on this section, we need to prepare a suitable place for it. Let's find the place in the code:

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

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

This is where we will develop the next sections of our website. So let's change this element to the following content:

```
<!--Main layout-->
<mainclass="mt-5">
  <divclass="container">
      <!--Section: LastTravels-->
      <sectionid="lastTravels">
      </section>
      <!--Section: LastTravels-->

      <hrclass="us-5">

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

      <hrclass="us-5">

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

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

As seen in the example abovewe have added many <section> tags in which we will develop further areas of the website. At this point, we will focus on the lastTravels id section. The <section> tag itself has no special meaning, but it was introduced for code readability. Typically, each section has a unique id so that it can be linked to links in the menu.

Inserted between sectionsseparators <hr> with my-5 class set. This class is responsible for setting the appropriate margin between sections.

Now let's prepare the frame for our tiles. It will consist of two rows and three columns. Let's paste the following code instead of <section id="lastTravels">:

```
<!--Section: LastTravels-->
<sectionid="lastTravels"class="text center">
  <!--Grid row-->
  <divclass="trench">

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

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

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

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

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

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

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

  <!--Grid row-->
  <divclass="trench">

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

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

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

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

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

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

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

Now it's time to prepare the graphics.Let's upload six images of about 800px/600px to the img folder. We will use them for our tiles. In the example we will use the names trip1.jpg through trip6.jpg.

Then, let's insert in the first row and first column a picture of the trip according to the scheme:

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

Notice that the image has the img-fluid class set. This class makes the image responsive. It will occupy the entire available space, which in our case has been limited by the external container with the bootstrap system's Grid parameters set. Let's save the settings and see the result in the browser.

Now let's add an effect that will appear when you click on the image. We need to change the column implementation to:

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

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

Note that in addition to addinglink with a mask, our image has been wrapped in a with a class `.view`. Material Design by defaultfor the class `.view`and `.btn`defines the Waves effect. Now let's click on the picture and see the result. You can read more about the effects here:Waves Effect Docs.

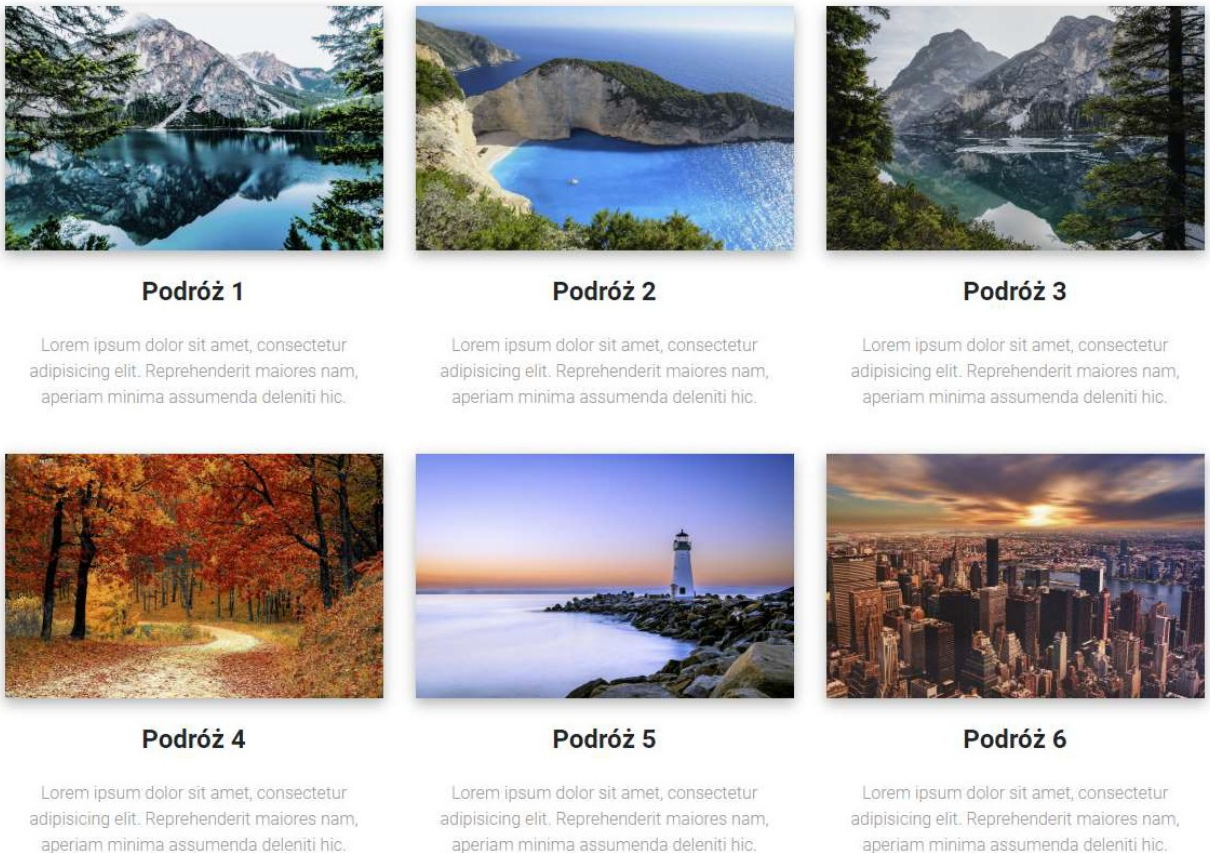
By adding a class `.z-depth-1-half`we turned onshadow under the picture. More about this functionality can be found here:Shadows Docs.

Let's also add an effect visible on hovercursor on the picture. This will provide us with the <a> link defined under the image, and in it a with the set class `.mask`. Enabling this functionality is very simple. Just next to the classroom `.mask`add anotherclass `.rgba-blue-light`. Let's do it and check the result. You can read more about these effects here:Hover Effects Docs.

It remains for us to add a short description under the picture. Thus, the final effect of defining a single tile may take the form of the following example:

```
<!--Grid column-->
<divclass="col-lg-4 col-md-12 mb-4">
  <divclass="view overlay z-depth-1-half">
    <imgsrc="img/trip1.jpg"class="img-fluid"alt="">
    <andhref="#!">
       <divclass="mask rgba blue light"></div>
  </and>
  </div>
  <h4class="my-4 font-weight-bold">Journey 1</h4>
  <pclass="grey text">
Lorem ipsum dolor sit amet, consectetur adipisicing elites. Reprehenderit maiores
n am, aperiam minima assumenda deleniti hic.
  </p>
</div>
<!--Grid column-->
```

Let's do a similar definition for the remaining tiles. The end result should resemble the one in the picture below:



Finally, I wanted to come backto the .fixed-top class defined for our menu. I promised I'd come back to it. Now you can see what it is for. Notice that when you move the page down, the menu stays stuck at the top. This is a functionality called sticky. This class is responsible for gluing the element permanently to the top of the screen.