

1. Zainstalowałam express:

```
/d/Programy/nodejs/aplikacjeinternetowe2
$ npm install -g express
+ express@4.16.3
updated 1 package in 5.906s
```

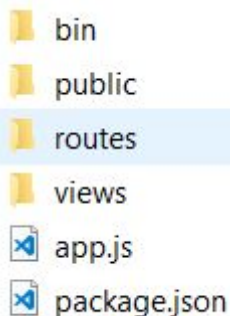
2. Oraz express-generator, który ułatwia budowanie struktury projektu:

```
/d/Programy/nodejs/aplikacjeinternetowe2
$ npm install -g express-generator
C:\Users\klebu\AppData\Roaming\npm\express -> C:\Users\klebu\AppData\Roaming\npm\node_modules\express-generator\bin\express-cli.js
+ express-generator@4.16.0
updated 1 package in 0.973s
```

3. Następnie, za pomocą generatora zbudowałam projekt:

```
/d/Programy/nodejs/aplikacjeinternetowe2
$ express aplikacje_internetowe2
```

4. I otrzymałam następujący układ plików:



- bin
- public
- routes
- views
- app.js
- package.json

5. Uzupełniam plik package.json o moduł nodemailer którego będę chciała użyć do obsługi formularza kontaktowego i instaluję moduły

```
/d/Programy/nodejs/aplikacjeinternetowe2/aplikacje_internetowe2
$ npm install
```


6. Objęłam projekt systemem kontroli wersji (git init) oraz podłączyłam zdalne źródło repozytorium GitHub (git add remote). Dodatkowo utworzyłam plik .gitignore, który ma wykluczyć z wersjonowania niepotrzebne elementy (te, które można z łatwością pobrać do projektu poprzez npm - dzięki czemu z 10 tysięcy plików wersjonowania, pozostało tylko 10 - tych, które faktycznie będą przeze mnie nie zmieniane). Poniżej treść pliku .gitignore:

```
1  /bin
2  /node_modules
3  package_lock.json
```

7. Dodałam pierwszy commit ze strukturą plików do repozytorium:

Branch: master ▾ New pull request

Create new file Upload files Find file Clone or download ▾

 DominikaHolota first commit Latest commit b7dfcd4 31 seconds ago

public/stylesheets	first commit	30 seconds ago
routes	first commit	30 seconds ago
views	first commit	30 seconds ago
.gitignore	first commit	30 seconds ago
app.js	first commit	30 seconds ago
package-lock.json	first commit	30 seconds ago
package.json	first commit	30 seconds ago

Add a README with an overview of your project. Add a README

8. Uruchomiłam projekt:

# Express

Welcome to Express

System wyświetlił stronę powitalną Expressa.

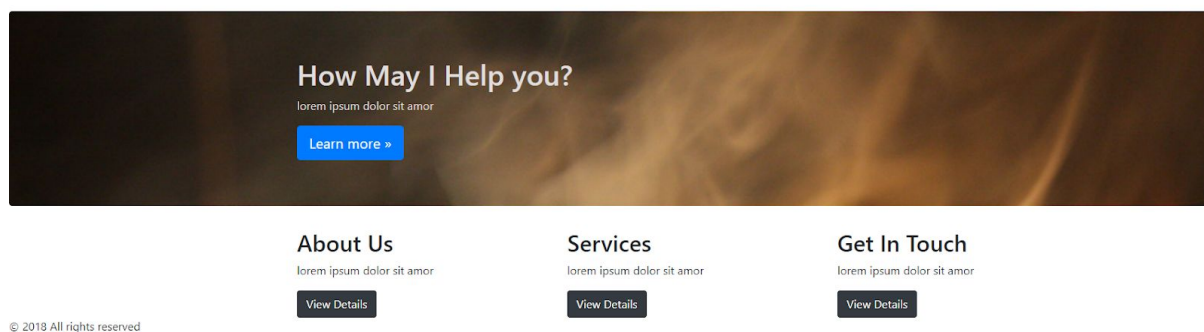
9. Zmodyfikowałam plik layout.jade - podpięłam do niego pliki bootstrapa.

```

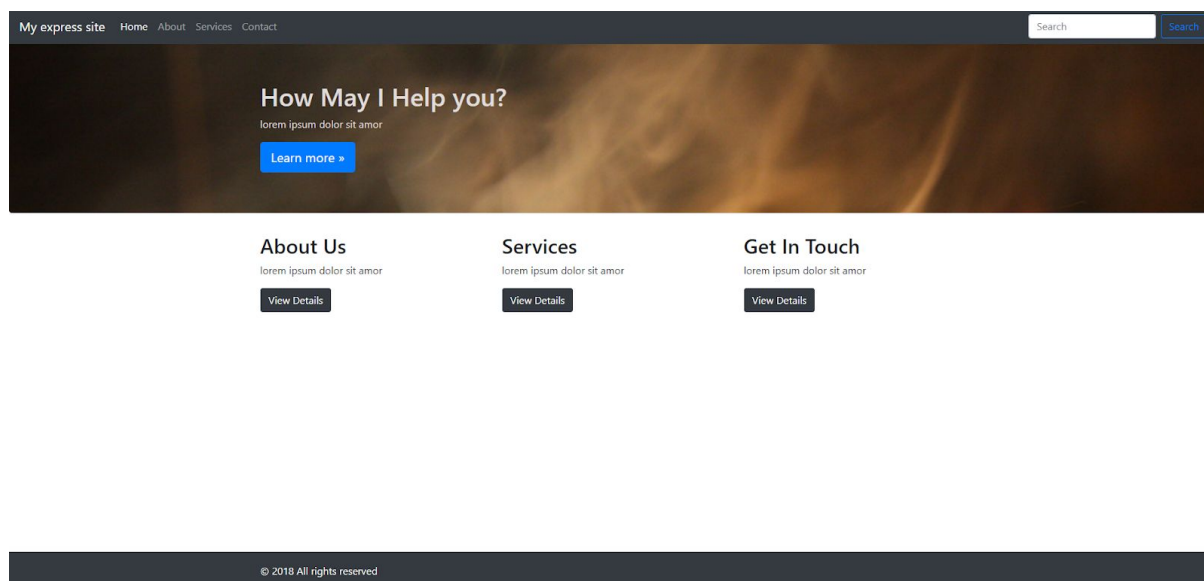
1  doctype html
2  head
3    // Required meta tags
4    meta(charset='utf-8')
5    meta(name='viewport', content='width=device-width, initial-scale=1, shrink-to-fit=no')
6    // Bootstrap CSS
7    link(rel='stylesheet', href='https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css', integrity
8    title Hello, world!
9
10 body
11   block content
12     // Optional JavaScript
13     // jQuery first, then Popper.js, then Bootstrap JS
14     script(src='https://code.jquery.com/jquery-3.3.1.slim.min.js', integrity='sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRV
15
16     script(src='https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js', integrity='sha384-cs/chFZ
17
18     script(src='https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js', integrity='sha384-uefMccjFJA
19

```

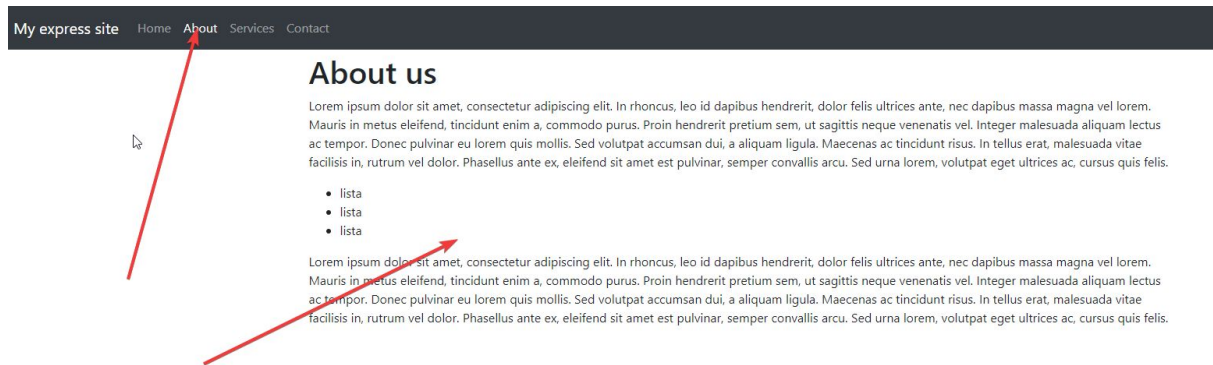
10. Do pliku index.jade dodałam jumbotron i 3 kolumny oraz przykładowe treści, dodałam background image dla klasy .jumbotron:



11. Oraz menu nawigacji i stopkę:



12. A także przykładową podstronę:



Ciekawym zabiegiem jest warunek, pozwalający na dodawanie klasy aktywnej po sprawdzeniu tytułu strony (w tym przypadku jako aktywny oznacza się “About” pozostałe są wyszarzone).

```
ul.navbar-nav.mr-auto
  li(class=(title === 'Home'? 'active': '')).nav-item
    a.nav-link(href='/') Home
  li(class=(title === 'About'? 'active': '')).nav-item
    a.nav-link(href='about') About
  li(class=(title === 'Services'? 'active': '')).nav-item
    a.nav-link(href='#') Services
  li(class=(title === 'Contact'? 'active': '')).nav-item
    a.nav-link(href='#') Contact
```

13. Dodałam również stronę z formularzem kontaktowym (na razie bez obsługi wysyłania):

14. Dodałam reakcję na wysłanie (na razie logowanie treści formularza do konsoli):

```
app.post('/contact/send', (req,res) => {  
  console.log(req.body);  
})
```

Co po wypełnieniu formularza na stronie daje (przykładowo) wynik:

```
GET /contact 304 52.747 ms - -  
GET /stylesheets/style.css 304 0.438 ms - -  
{ name: 'test', email: 'test@test.pl', message: 'test' }
```

15. Następnie dodałam szkielet html wiadomości:

```
app.post('/contact/send', (req,res) => {  
  //add mail body  
  const output = `  
    <p>You have a new message!</p>  
    <h3>Contact details</h3>  
    <ul>  
      <li>Name: ${req.body.name}</li>  
      <li>E-mail: ${req.body.email}</li>  
    </ul>  
    <h3>Message</h3>  
    <p>${req.body.message}</p>  
  `;  
  console.log(output);  
})
```

16. I obsługę wysyłki wiadomości z modułu nodemailer (zgodnie z dokumentacją: <https://nodemailer.com/about/> ). Podpięłam moduł pod istniejącą skrzynkę e-mail na serwerze, dzięki czemu mogłam przetestować działanie funkcjonalności:

☐ **A message from contact form on your website**

**Od:** Domi <express@liniuszka.pl>  
**Do:** express@liniuszka.pl; test@test.pl  
**Data:** 18:42

You have a new message!

**Contact details**

- Name: test
- E-mail: test@test.pl

**Message**

test



17. Dodałam też komunikat z informacją o wysłaniu wiadomości na stronie:

**Message has been sent.**

**Contact**

Name

Enter Name

Email

Enter e-mail adress



Stosowny komunikat jest również wyświetlany w konsoli serwera.

18.