Dokumentacja
Paweł Babiuch

Wykorzystane technologie:

NodeJS – Javascript Server-Side Runtime. Pozwala wykonywać kod JavaScript po stronie serwera, co pozwala np. serwować strony wygenerowane podczas wysyłania, a nie po stronie użytkownika, jak w przypadku technologii PHP, ASP czy JSF.

npm – menedżer paczek dla JavaScript, współpracujący ściślę z NodeJS. Jest dostępny samodzielnie, lub jako część pakietu NodeJS. Uruchamiany z linii komend, robi to, co każdy inny package manager - pobiera paczki z repozytoriów, i pozwala na zarządzanie nimi. npm może instalować paczki globalnie, co jest rekomendowane np. w przypadku pre-procesora SASS lub Express-Generator, albo lokalnie, jako część aplikacji webowej.

Express – framework bazujący na NodeJS. Jego działanie jest podobne do PHPowego Laravela czy Symphony. Zajmuje się generowaniem i serwowaniem plików, routingiem, komunikacją z bazą danych, i ułatwia budowanie dużych aplikacji webowych.

Pug – templating engine. Dawniej Jade, jest to silnik do budowania szablonów stron. Pozwala tworzyć strony wygodniej niż w czystym HTML, plus pozwala zagnieżdżać w sobie szablony.

Bulma – CSS framework. Nowoczesny i popularny framework w stylu Bootstrapa. Sama Bulma jest napisana w SASS - nie zawiera żadnego JavaScriptu lub innego zewnętrznego kodu.

Działanie aplikacji:

Aplikacja jest prostą, statyczną wizytówką dla firmy. Nie posiada podpięcia pod żadną bazę danych.

Do uruchomienia strony wymagane jest użycie npm. Po pobraniu paczki z projektem należy wejść do jego lokalizacji i wpisać:

```
npm install
npm start
```

Po wpisaniu `npm install` zainstalowane zostaną lokalnie wszystkie paczki, z których korzysta projekt - Bulma, Express, Pug, i kilka innych, pomniejszych zależności. uruchomi lokalny serwer http serwowany przez NodeJS/Express, nasłuchujący na porcie 3000.
Jeśli wszystko wykonało się poprawnie, strona będzie dostępna pod adresem http://localhost:3000/

Strona główna jest typowym one-pagem - wyświetla podstawowe informacje o firmie w formie przewijanego pokazu treści. Większość komponentów zbudowano z wykorzystaniem Bulma - spersonalizowany CSS został wykorzystany by nałożyć obrazy tła dla poszególnych segmentów.

Strona galeria wyświetla zdjęcia, które można powiekszyć klikając na nie - w tym celu wykorzystuję krótki fragment JavaScriptu, który dopisuje odpowiednie klasy aktywacyjne dla modali z dużymi zdjęciami.

Strona kontaktu wyświetla prosty tekst i formularz, oraz embed mapy z Google Maps.

Struktura aplikacji:

W folderze root znajdują się:
package.json - plik npm, za pomocą którego wykonywania jest instalacja - jest on również wykorzystywany jako metadane po uploadzie projektu na repozytorium npm.

app.js - punkt wejściowy aplikacji, w którym znajdują się jej najważniejsze skrypty. W app.js wywoływany i konfigurowany jest Express, korzystając z domyślnej konfiguracji w dokumentacji Express. Tworzone są połączenia z podstawowymi kontrolerami routingu, aktywowany zostaje Pug, i wyświetlany jest widok.

routes - w tym folderze znajdują się wszystkie kontrolery routingu. Wykorzystuję tylko index.js - users.js został wygenerowany jako przykład przez generator aplikacji Express, i nie jest wykorzystywany w samej aplikacji.
index.js zawiera trzy definicje ścieżek - index, gallery i contact - odpowiadają one trzem podstronom, na które można wejść z poziomu aplikacji. Wszystkie przyjmują do ścieżki parametr `brand`, podstawiany pod nazwę firmy. Dodatkowo gallery i contact przyjmują parametr `title`, podstawiamy jako dodatek do tytułu strony, po "Projekt - ".

public - ten folder zawiera wszystkie pliki, które serwowane są bezpośrednio do klienta bez przetwarzania przez Express - obrazki w folderze images, pliki js w folderze javascripts, oraz arkusze stylów w stylesheets.
W folderze javascripts znajduje się jeden plik - script.js, odpowiedzialny za uruchamianie modali w galerii, po kliknięciu na zdjęcie.
Stylesheets zawiera kilka arkuszów - w style.css są reguły nakładające obrazki tła na poszczególne sekcje strony głównej. Normalize.css odpowiada za "znormalizowanie" strony - usunięcie domyślnych stylów, jak marginesy dla body. Jest on tworzony przez społeczność i dostępny przez npm, jednak wolałem mieć "twardą" kopię dla ułatwienia podczepienia do reszty kodu. Bulma.scss zawiera podstawowe, "surowe" reguły frameworku Bulma - sam plik składa się z kilku spersonalizowanych reguł, i importów głównej części Bulmy z node_modules. Plik Bulma.css jest skompilowaną wersją .scss, i to on jest podpinany do strony jako arkusz stylów.
W celu wygenerowania nowej wersji Bulma.css, należy zainstalować Dart SASS, i uruchomić w folderze stylesheets komendę `sass bulma.scss bulma.css`. Jest to wymagane tylko wtedy, gdy zmieniła się wersja Bulmy, brakuje pliku Bulma.css, lub zmodyfikowany został plik Bulma.scss.

views - tutaj znajduje się główna część aplikacji - widoki serwowane użytkownikowi. Folder views został rozbity na kilka podfolderów - w jego roocie znajduje się strona 404, domyślna, wygenerowana przez Express.
Folder layouts zawiera główne layouty aplikacji - zawiera tagi DOCTYPE, html, head i body, i wkleja w nie szablony head, navbara i stopki, zostawiając blok content na treści z poszczególnych stron. Istnieją dwa layouty - layout.pug jest standardowym layoutem, front.pug jest używany na stronie głównej - jedyna różnica to zawarcie navbara w elemencie div.hero w przypadku front.pug, by poprawnie był wyświetlany w połączeniu z tym elementem.
Folder components zawiera najmniejsze bloki aplikacji - w tym przypadku, head, w którym deklarowane są tytuł strony, CSS i JS, czyli elementy które nigdy się nie zmieniają, navbar, w dwóch wersjach - navbar-front jest wykorzystywany na stronie głównej i ma troche inną kolorystykę przycisków, oraz przeźroczyste tło, i w końcu footer - stopka strony, która jest wszędzie taka sama.
Widoki są tworzone przez Pug, wywoływany automatycznie przez Express podczas serwowania strony do klienta - syntax Pug jest bardzo podobny do HTML, ale nie używa się ostrych nawiasów i tagów zamykających, zamiast tego wykorzystując do organizacji kodu wcięcia - tagi na tym samym poziomie istnieją obok siebie, te wcięte są potomkami najbliższego wyższego poziomu.
Tagi block oznaczają bloki treści, które mogą być wymienione na inne w szablonach dziedziczących. Tagi include wklejają wymienione szablony, a tag extends deklaruje rodzica - najpierw ładowany jest rodzic, a następnego podmieniane są bloki na te z potomka.

Wracając do root aplikacji, folder bin zawiera skrypt niezbędny do działania aplikacji - został on wygenerowany automatycznie i zawiera podłączenia do wszystkich zależności systemu.
Folder node_modules będzie wygenerowany po zainstalowaniu aplikacji - tam znajdują się wszystkie paczki zależności, jak np. Express, Pug i Bulma.