**Instrukcja wdrożeniowa**

**Michał Dunat 248862**

**Repozytorium aplikacji serwerowej:** [**https://github.com/michu-dunat/document-management-server**](https://github.com/michu-dunat/document-management-server)

**Repozytorium aplikacji klienckiej:** [**https://github.com/michu-dunat/document-management-app**](https://github.com/michu-dunat/document-management-app)

Projekt wdrożono na platformie hostingowej **Heroku** - <https://www.heroku.com>.

Potrzebne aplikacje: **Git** - <https://git-scm.com/download/win>.

**Opis wdrożenia przy wykorzystaniu Heroku:**

Przed przystąpieniem do wdrożenia należy sklonować oba repozytoria lokalnie z platformy **Github** - <https://github.com/>.

W przypadku obu projektów należy im założyć aplikacje na Heroku – przy ich tworzeniu określa się ich nazwę oraz lokalizację serwerów (Europa bądź Stany Zjednoczone).

Należy zainstalować **Heroku CLI** - <https://devcenter.heroku.com/articles/heroku-cli>.

Następnie wykorzystując poniższą komendę logujemy się do Heroku na lokalnej maszynie:

|  |
| --- |
| heroku login |

W repozytoriach należy utworzyć połączenie z repozytorium zdalnym utworzonym dla aplikacji Heroku za pomocą komendy:

|  |
| --- |
| heroku git:remote -a NAZWA\_APLIKACJI\_NA\_HEROKU |

**Konfiguracja aplikacji serwerowej:**

Należy stworzyć plik „system.properties” w folderze głównym projektu oraz dopisać do niego linijkę:

|  |
| --- |
| java.runtime.version=11 |

W kontrolerach przy adnotacji „@CrossOrigin” umieścić URL aplikacji klienckiej Heroku.   
W przypadku pierwotnego wdrożenia wygląda to następująco:

|  |
| --- |
| @CrossOrigin(origins = {"http://localhost:4200", "https://document-management-client.herokuapp.com"}) |

W aplikacji Heroku należy dodać wtyczkę **ClearDB** – . Wymagane jest podpięcie karty kredytowej do konta Heroku. Po weryfikacji konta w polu „Add-ons” znajdującym się w zakładce „Resources” należy wyszukać wtyczkę ClearDB oraz dodać ją do aplikacji.

**Konfiguracja aplikacji klienckiej:**

W folderze „src/environments” w plikach „environment.ts” i „environment.prod.ts” stworzyć   
i ustawić wartości zmiennej „serverUrl”, która będzie przechowywać link URL do aplikacji serwerowych uruchomionych odpowiednio lokalnie i wdrożeniowo. Następnie w serwisach zaimportować tą zmienną i jej używać do wyznaczenia adresów odpowiednich endpointów.

W pliku „package.json” dodać następującą linijkę w obiekcie „scripts”:

|  |
| --- |
| "heroku-postbuild": "ng build --prod && npm install -g http-server-spa", |

Oraz zmodyfikować pole „start”:

|  |
| --- |
| "start": "http-server-spa dist/document-management-app index.html $PORT", |

Gdy chcemy wdrożyć aplikację po raz pierwszy bądź ją zaktualizować na zdalnym serwerze należy użyć kombinacji poniższych komend:

|  |
| --- |
| git add .  git commit -am "make it better"  git push heroku master |

**Opis wdrożenia na własnym serwerze:**

Projekt wdrożono tylko i wyłącznie na Heroku. Nie próbowano dokonać tego na własnym serwerze lecz powinno to być możliwe przy wykorzystaniu **Dockera** - <https://www.docker.com/>. W dalszym ciągu będzie potrzebne odpowiednie dopasowane URL w kontrolerach aplikacji serwerowej oraz serwisach aplikacji klienckiej jak opisano to wcześniej. Poradnik jak utworzyć obraz bazy danych jest przedstawiony na następującej stronie - <https://hub.docker.com/_/mysql>. Opis jak można utworzyć obraz dockerowy aplikacji Springowej opisany jest w tym poradniku - <https://spring.io/guides/gs/spring-boot-docker/>. Środowisko IntelliJ IDEA Ultimate (<https://www.jetbrains.com/idea/>) umożliwia konfigurację uruchomieniową Dockera - <https://www.jetbrains.com/help/idea/docker.html>. Należy pamiętać by w pliku „application.properties” znajdującym się w folderze „src/main/resources” zmienna przechowująca URL do bazy danych wskazywała na kontener Dockerowy. Proces budowania oraz hostowania strony klienckiej projektu opisany jest w tym poradniku - <https://www.indellient.com/blog/how-to-dockerize-an-angular-application-with-nginx/>. W obu przypadkach całość sprowadza się do następujących kroków:

* Odpowiednia modyfikacja kodu aplikacji.
* Utworzeniu pliku „Dockerfile”.
* Stworzenie obrazu Dockerowego aplikacji.
* Uruchomienie stworzonego obrazu na własnym serwerze.