# Wstęp

W dzisiejszym, coraz bardziej wymagającym świecie wytwarzania oprogramowania duże znaczenie ma zarówno jakość, czas dostarczenia produktu, jak i procesy związane z pracą nad projektem. W celu osiągnięcia jak najwyższej jakości i niezawodności, wdrażane są rozmaite metodologie i procesy optymalizujące pracę.

Jednym z trendów na dzisiejszym rynku jest ciągłe dążenie do usprawnienia procesu wytwarzania oprogramowania, równeiż poprzez wdrażanie elementów automatyzacji samego pisania kodu. Powszechnie używane są na przykład narzędzia ciągłej integracji zwalniające programistów z uciążliwych zadań związanych z administracją i zarządzaniem programem (*Jenkins*), aplikacje do kontereneryzacji aplikacji w celu ustandaryzowania środowiska uruchomieniowego (*Docker*) i inne.

Innowacją i krokiem naprzód są programy pomagające programiście pisać sam kod. W nowoczesnych środowiskach programistycznych (*IDE*) dostępne są na przykład funkcje generujące skrawki kodu przy użyciu skrótów klawiszowych (code snippets). Dostępne są również samodzielne generatory, tworzące pliki źródłowe programu od podstaw, a także pozwalające na ciągłą pracę z projektem poprzez generowanie jego elementów. Przykładem takiego oprogramowania są napisane we frameworku Yeoman generatory w pewnym stopniu wyręczające programistów z pracy nad tworzeniem samego kodu.

W ramach przedstawianej pracy pokazany będzie proces tworzenia takiego generatora, sposób jego działania, zastosowane w nim technologie i rozwiązania, a także szeroko płynące korzyści z napisania własnego generatora na potrzeby własne.

# 1 Podstawy teoretyczne

## 1.1 Historia

Yeoman jest zespołem narzędzi opartym na platformie *NodeJS*, wspomagającym budowanie aplikacji internetowych łącząc wiele różnych rozwiąząń, jak np.:

* EJS (Embedded JavaScript) - praca z szablonami za pomocą języka JavaScript
* Inquirer - zespół interaktywnych interfejsów linii komend do komunikacji z użytkownikiem
* Yo - narzędzie linii komend do uruchamiania generatorów Yeoman

Logo frameworku Yeoman

Ekosystem generatorów Yeoman zawiera szereg innych narzędzi, między innymi do zarządzania zależnościami aplikacji, uruchamiania testów, optymizacji kodu i przygotowania środowiska programistycznego.

Yeoman został po raz pierwszy przedstawiony na konferencji Google I/O w 2012 roku i od tego czasu jest dominującą technologią w zakresie generowania kodu i w ten sposób usprawniania procesu wytwarzania aplikacji. Aktualnie na oficjalnej stronie (<http://yeoman.io/>) dostępnych jest publicznie ponad 6000 generatorów utworzonych przez członków społeczeństwa, pomagających wystartować z projektem w ulubionej technologii. Yo - narzędzie potrzebne do uruchamiania generatorów zostało pobrane z repozytorium *npm* ponad 3 miliony razy, w tym ponad 100 000 razy w ostatnim miesiącu (maj 2017).

System generatorów Yeoman w swoim początku był portem generatorów Rails, tak aby działały na platformie NodeJS i pozwoliły programistom na pisanie własnych narzędzi, identyfikując powtarzające się czynności i elementy dla ich reużywalności.

## 1.2 Korzyści płynące z generatorów

Dobrze przygotowany generator potrafi na podstawie interakcji z użytkownikiem utworzyć kompletny, kompilujący się i gotowy do uruchomienia program. Jednym z głównych założeń, które programista powinien mieć na uwadze pisząc własny generator, jest aby generator nie powodował w programie żadnych usterek i nie psuł jego struktury. Zawsze przed momentem, w którym pliki są zmieniane i tworzone, uzytkownik może dokładnie obejrzeć zmiany jakie zajdą w programie. Używanie generatorów w kompatybilnych projektach jest zatem bezpieczne i nie powinno powodować żadnych negatywnych, nieoczekiwanych efektów.

Generator Yeoman pozwala zadbać o dobrą strukturę katalogów i podział na moduły, w pewien sposób wymuszając na programiście dobre praktyki i przyspieszając pracę z projektem. Generator do odpowiedniego działania musi wiedzieć na przykład, gdzie spodziewać się plików konfiguracyjnych, gdzie znajdują się pliki klas i jak wygląda struktura programu. Takie warunki działania generatora, pomagają zadbać o dobrą strukturę projektu, chociażby dla zachowania samej jego kompatybilności z generatorem.

Generator Yeoman to nie tylko szybki start z projektem i dobre praktyki, ale także dynamiczne implementowanie konkretnych funkcjonalności. Wiele zadań w toku pracy nad oprogramowaneim jest powtarzalnych, dodawane kolejno nowe moduły są często kopią poprzednich dotycząc tylko innych danych. Przykładowo w aplikacjach internetowych zapisujących obiekty w bazie danych, dla każdej nowej encji (obiektu biznesowego) zwykle implementuje się operacje dodawania, usuwania i edycji obiektów. Jest to czynnością wysoce powtarzalną i rzadko różniącą się czymś więcej, niż tylko nazwą klasy.   
Jest to idealne zastosowanie dla generatora, który mógłby przyjmować jako dane wejściowe nazwę encji i jej podstawowe atrybuty, generując:

* klasę encji
* klasę DAO (*Data Access Object)* z podstawowymi metodami
* serwis zawierający logikę biznesową
* kontroler stanowiący interfejs do interakcji z zewnętrznymi systemami

Wygenerowane klasy umieszczone byłyby w odpowiednich katalogach, zapewnione by było ich współdziałanie i kompilowalność, pozwalając programiście od razu implementować logikę biznesową bez obaw o tworzenie w programie odpowiedniej, działającej struktury.

Programista, w alternatywie do kopiowania kodu z poprzednich implementacji podobnych modułów, może uruchomić generator nie narażając się na możliwość pomyłki. Odpowiadając na kilka pytań otrzymuje działający kod napisany w zgodzie z nowoczesnymi praktykami programistycznymi. Takie podejście pozwala na uniknięcie błędów i znacznie przyspiesza pracę.

Yeoman, jako środowisko jest efektywny w każdym używanym dziś języku programowania. Generowany kod może mieć dowolną formę, zależną od programisty i użytkownika generatora, dlatego Yeoman znajduje wiele różnych zastosowań.

Najbardziej popularne przypadki użycia generatora:

* tworzenie nowego projektu
* tworzenie nowych elementów istniejącego projektu
* tworzenie modułów i pakietów
* prototypowanie i przygotowywanie wersji demo aplikacji
* wymuszanie na programistach dobrych praktyk
* generowanie przykładowego kodu w tutorialach
* tworzenie prezentacji multimedialnych

Sam Yeoman, bez odpowiedniej implementacji nie podejmuje żadnych decyzji odnośnie generowanego kodu. Fakt, że wszystko zależne jest od programisty pozwala na elastyczność i tworzenie generatorów nie związanych ściśle z programowaniem. Ciekawym przypadkiem użycia jest stworzony przez Adriana Bateman (Program Manager w zespole Microsoft Edge Web Platform) generator specyfikacji technicznej opartej na bibliotece ReSpec, przyspieszający tworzenie nowych dokumentów technicznych.

<https://adrianba.net/2015/03/14/using-yeoman-to-start-writing-technical-specifications-with-respec/>

<https://github.com/adrianba/generator-respec>

## 1.3 Przykładowe generatory

Na rynku dostępnych jest wiele powszechnie używanych implementacji, dobrze przetestowanych i ciągle rozwijanych

* **JHipster** - bez wątpienia najbardziej popularny, oparty na otwartym oprogramowaniu, bardzo dobrze realizujący wcześniej wspomniane założenia i pokazujący jak bardzo użyteczny może być generator Yeoman.

Logo JHipster

Generuje w pełni funkcjonalną aplikację internetową w języku Java (Spring Boot + Angular), pozwala zarówno na samo stworzenie projektu jak i ciągłą pracę z nim przy użyciu podgeneratorów. Używany jest codziennie w wielu firmach, oferując szereg narzedzi realizujących różne funkcjonalności, jak np. wtyczki do IDE i podgeneratory konfiguracji chmurowej. Oferuje dużą różnorodność w wyborze używanych technologii, posiada duże wsparcie społeczeństwa programistycznego i wielu fanów.

* **Angular** – generator aplikacji jednostronnych AngularJS, pozwalający na utworzenie kodu zarówno w języku JavaScript, CoffeeScript jak i TypeScript. Posiada szereg podgeneratorów dla popularnych elementów aplikacji AngularJS, jak kontrolery, widoki, dyrektywy i filtry. Utworzony przez zespół dostarczający framework Yeoman.
* **generator** – generator generatorów Yeoman. Przygotowuje strukturę plików potrzebną do sprawnego rozpoczęcia implementacji własnego generatora, tworzy pliki zwykle dodawane w przypadku pracy z repozytorium kodu i NodeJS.

# 2 Środowisko programisty

## 2.1 Ekosystem Yeoman