



UMCS

**UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ
W LUBLINIE**

Wydział Matematyki, Fizyki i Informatyki

Kierunek: **Informatyka**

Konrad Pakuła

nr albumu: 275177

Tworzenie aplikacji webowej przy użyciu Spring Boot i Angular framework

**Web application development using Spring Boot
and Angular framework**

Praca licencjacka
napisana w Zakładzie Informatyki Stosowanej
pod kierunkiem dra Rajmunda Kuduka

Lublin 2019

Spis treści

Wstęp	4
1 Wprowadzenie literaturowe	6
1.1 Aplikacje webowe	6
1.2 Technologie w aplikacjach webowych	8
1.2.1 Technologie wykonywane po stronie klienta	9
1.2.2 Technologie wykonywane po stronie serwera	11
1.3 Protokół HTTP	12
1.4 RESTful web services	13
2 Wykorzystywane oprogramowanie	14
2.1 Spring Framework	14
2.2 Spring Boot	15
2.3 Spring Intializr	16
2.4 IntelliJ IDEA	16
2.5 Apache Maven	17
2.6 Angular	19
2.6.1 Historia	19
2.6.2 Instalacja Node.js	20
2.6.3 Node Package Manager	21
2.6.4 Angular CLI	21
2.6.5 Instalacja Angular CLI	21
2.7 JHipster	22
2.7.1 Instalacja	23

3 Tworzenie aplikacji	24
3.1 Założenia tworzonej aplikacji	24
3.2 Dobór bazy danych	24
3.3 Generowanie podstawowego projektu	26
3.4 Uruchomienie aplikacji	27
3.5 Połączenie z bazą danych	27
3.6 Model	28
3.7 Plik startowy serwera	30
3.8 Repozytorium	31
3.9 Serwis	31
3.10 Kontroler	32
3.11 Model w kliencie	34
3.12 Moduł Angular	35
3.13 Serwis klienta	36
3.14 Nawigacja	37
3.15 Generowanie komponentu	37
3.16 Edycja komponentu	39
3.17 Prezentacja komponentu	40
3.18 Prezentacja aplikacji	41
3.18.1 Strona startowa	42
3.18.2 Rejestracja i logowanie	43
3.18.3 Pasek nawigacji	45
3.18.4 Widok dodawania wydarzenia	45
3.18.5 Widok wszystkich wydarzeń	46
3.18.6 Usuwanie wydarzenia	47
3.18.7 Informacje szczegółowe dla wydarzenia	48
Podsumowanie	50
Bibliografia	52

Wstęp

Rozwój aplikacji internetowych w ostatnich latach znacznie przyspieszył. Coraz częściej programiści postanawiają tworzyć aplikacje internetowe niż natywne oprogramowanie. Na przestrzeni ostatnich kilku lat trend pojawiających się stron o charakterze interakcyjnym znacznie wzrósł. Popularnymi w dzisiejszych czasach przykładami wyżej wymienionego typu aplikacji jest m.in. *Netflix*, dający możliwość oglądania interesujących nas seriali lub filmów dzięki przesyłaniu strumieniowemu. Kolejnym przykładem mogą być popularne *Google Docs*, dające możliwość pracy na dokumentach. Jednymi z największych zalet aplikacji webowych jest brak problemów ze zgodnością jak również możliwość udoskonalania danego oprogramowania bez potrzeby aktualizowania oprogramowania po stronie odbiorcy.

Celem pracy jest przedstawienie procesu tworzenia prostej aplikacji webowej oraz ukazanie możliwości zbudowania jej przy użyciu wybranych narzędzi umożliwiających zautomatyzowanie pracy i ominięcie zbędnego kodu o charakterze powtarzalnym.

Pierwszy rozdział zawiera wprowadzenie teoretyczne dające podstawy do zagadnień związanych z tematem pracy. Między innymi zostaną przedstawione pojęcia takie jak, opis protokołu HTTP, czym są aplikacje webowe oraz jakie są ich rodzaje. Ponadto zostanie określona różnica między, elementami wykonywanymi po stronie klienta, jak również po stronie serwera.

Drugi rozdział pracy przedstawia narzędzia użyte w procesie tworzenia aplikacji, jak również ich opis oraz proces ich instalacji, jeśli jest wymagany, czy też konfiguracji pod kątem konkretnego rozwiązania. Dodatkowo zostanie zaprezentowany przykład ich użycia.

W ostatnim rozdziale zostanie opisany proces tworzenia aplikacji, jak również będą przedstawione ogólne założenia realizowanego projektu. Oprócz tego wystąpi omówienie przykładowego komponentu, jak też wszystkich kroków niezbędnych do działania aplikacji. Zaczynając od serwera a kończąc na warstwie wizualnej. Na samym końcu zostanie przedstawione krótkie podsumowanie całej pracy.

Rozdział 1

Wprowadzenie literaturowe

1.1. Aplikacje webowe

Aplikacje internetowe, to inaczej programy które składają się z interaktywnych komponentów, których zadaniem jest wykonanie określonych funkcjonalności oraz umożliwienie komunikacji między użytkownikiem końcowym a samą aplikacją za pomocą klienta przeglądarki [1]. Proces tworzenia takowego oprogramowania w większości przypadków należy do bardzo rozbudowanego. Wynika to z faktu, że osoby współtworzące projekt, muszą wykazać się dużą wiedzą, jeśli planują rozwijać oprogramowanie. W większości przypadków, w komercyjnych projektach występuje podział na zespół pracujące nad serwerem, czyli nad logiką biznesową, oraz grupę programistów, zajmujących się częścią wizualną. Aplikacje na przestrzeni lat coraz częściej zachwycają użytkowników w kontekście nie tylko wizualnym jak również funkcjonalnym [2]. W obecnych czasach aplikacje internetowe zyskują na sile.

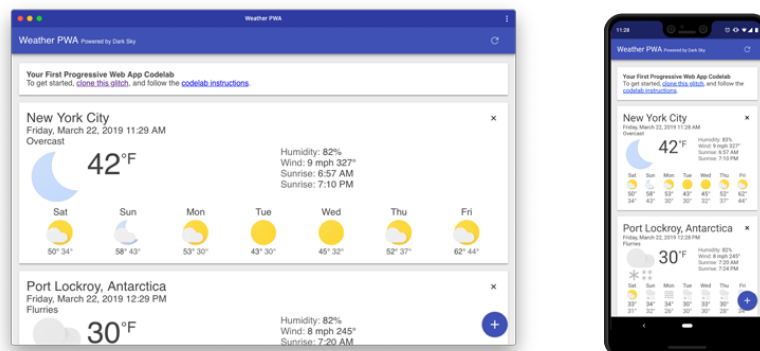
Jednymi z największych zalet tego typu oprogramowania są:

- aplikacje internetowe, nie wymagają instalacji na komputerze lokalnym przez użytkownika końcowego,
- brak problemu ze zgodnością podczas uruchamiania,
- w przypadku awarii, możliwość szybkiej naprawy, dla oprogramowania które działa w sposób niepoprawny,

- korzystanie z przeglądarki jako klienta, przez co występuje brak wymogu tworzenia pod określoną specyfikację i środowisko,
- dodanie nowej funkcjonalności w czasie rzeczywistym.

Aplikację internetową można dodatkowo określić mianem strony internetowej o charakterze interakcyjnym, która ma na celu kooperację z użytkownikiem. W czasie początkowych lat istnienia ogólnodostępnego Internetu, strony miały całkowicie inny wygląd oraz charakter. Pierwotnie były one tworzone w sposób statyczny, w celu dostarczenia informacji. Z czasem, gdy rynek rozwijał się w kierunku aplikacji internetowych, zaczęły powstawać dedykowane języki, które w coraz większy sposób kreowały rozwijające się zapotrzebowanie [3]. Aktualnie, wzrasta liczba projektów komercyjnych tworzących oprogramowanie tego rodzaju, a firmy rezygnują z tworzenia aplikacji natywnych skierowanych na określony system. Obecnie można wymienić kilka typów aplikacji internetowych:

- **Dynamiczne** - jest to strona, która jest modyfikowana po wywołaniu żądania, gdy dostaje informacje z serwera, w sposób dynamiczny.
- **PWA** (ang. *Progressive Web Application*) - są to aplikacje, które mają łączyć ze sobą zalety aplikacji webowych, jak i natywnych. Muszą być dostępne dla każdego, niezależnie od klienta przeglądarki, automatycznie dostosowywane pod sprzęt z którego są wykorzystane. Dodatkowo informacje na nich muszą być aktualne, mają za zadanie imitowanie natywnych aplikacji oraz umożliwienie korzystania w trybie offline, nawet z części funkcjonalności [4].

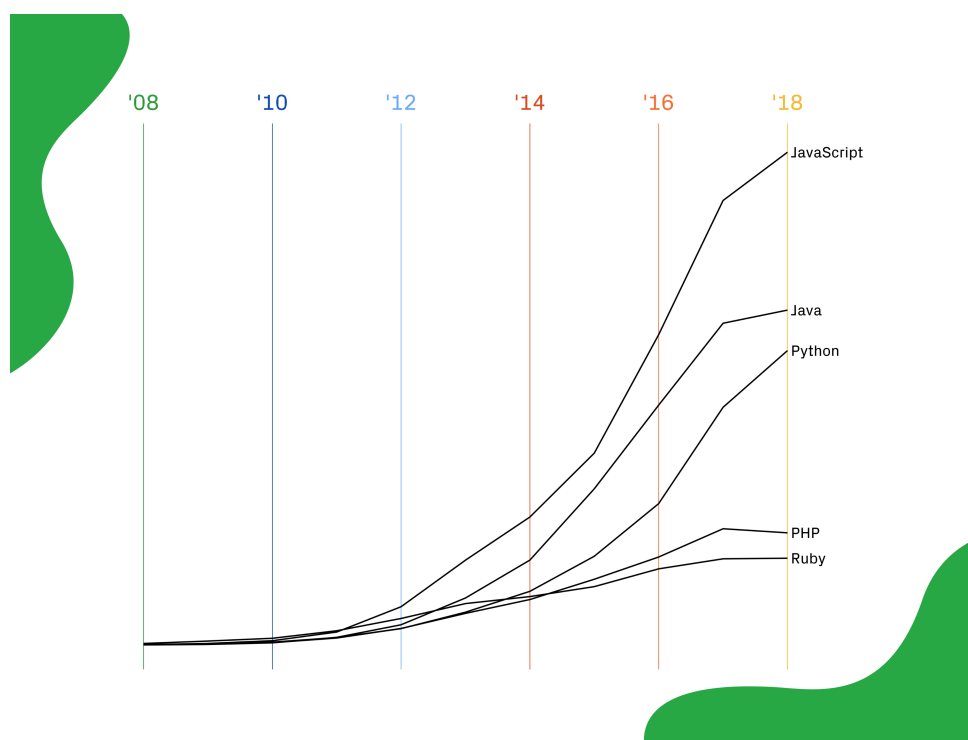


Rys. 1.1. Przykład wyglądu Progressive Web Application [5].

- **SPA** (ang. *Single Page Applications*) - są to aplikacje, ładujące wyłącznie pojedynczą stronę, która w momencie interakcji z użytkownikiem, jest dynamicznie aktualizowana i większość operacji, wykonywanych jest po stronie klienta [6].

1.2. Technologie w aplikacjach webowych

Proces tworzenia aplikacji webowych jest bardzo rozwinięty. Aktualny rynek daje dużą dowolność wyboru technologii tworzenia aplikacji internetowych.



Rys. 1.2. Najpopularniejsze języki programowania w repozytoriach GitHub, lata 2008–2018 [7].

W obecnej chwili aplikacje internetowe rozpowszechniają się w szybkim tempie. Technologie umożliwiające tworzenie takich aplikacji udostępniają coraz więcej modułów rozwijanych przez społeczność, bądź też twórców. Proces tworzenia aplikacji możemy podzielić na dwa typy rozwiązań. Wykonywanych po stronie klienta i po stronie serwera.

1.2.1. Technologie wykonywane po stronie klienta

Aktualnie językami umożliwiającymi tworzenie aplikacji internetowej wykonywanej po stronie klienta są takie technologie jak:

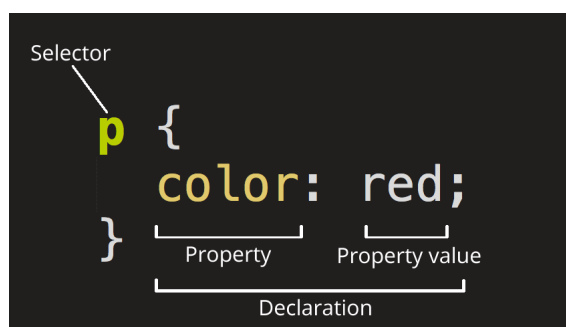
- HTML,
- CSS,
- JavaScript,
- TypeScript.

HTML (ang. *HyperText Markup Language*) jest to język znaczników, który opisuje strukturę stron internetowych. Składa się on z elementów umożliwiających dodawanie komponentów do strony. Jest on podstawowym elementem każdej witryny. Przeglądarka, która jest używana, zaczyna od odczytania danego pliku html a następnie go interpretuje [8].

Przykładowy znacznik będący częścią HTML:

```
<div class="przyklad">  
  <p> To jest przykład </p>  
</div>
```

CSS (ang. *Cascading Style Sheets*) jest to język opisowy służący do możliwości stylizacji wyglądu strony internetowej. Aktualnie jest on podstawowym elementem każdej witryny. Nie jest to język programowania, lecz służy do kreowania wyglądu strony [8].



Rys. 1.3. Przykład składni CSS [9].

Elementami składowymi arkuszy do stylizacji jest selektor oraz deklaracja. Deklaracja obejmuje w swoim zakresie własność i wartość do niej przypisaną. Każda z kolejnych deklaracji jest zakończona średnikiem, a między elementami ustanawiającymi daną własność znajduje się dwukropek, który ma za zadanie oddzielić własność od jej wartości [10].

JS (ang. *JavaScript*) jest to język programowania o charakterze skryptowym, powstały w 1995 roku w korporacji *Netscape* [11]. Pierwsze wydanie powstało w 10 dni i nazywało się *Mocha* lecz z biegiem czasu nazwa finalnie została zmieniona na *JavaScript*. Celem tego języka jest wspomaganie tworzenia stron i aplikacji internetowych. *JS* umożliwia interakcje z użytkownikiem korzystającym z określonej strony [12]. Powstanie *JavaScript* miało na celu ingerencje w język znaczników - *HTML*, by ulepszyć odczucia użytkownika końcowego. Jest językiem programowania, który wykonuje się po stronie klienta. Na przestrzeni lat pozycja *JavaScript* umocniła się, co spowodowało, że jest jednym z najpopularniejszych języków występujących w repozytoriach serwisu *GitHub*.



Rys. 1.4. Logo JavaScript [13].

TS (ang. *TypeScript*) to język programowania, który jest rozszerzoną wersją *JavaScript*, czyli jej nadzbiorem. Stworzony został przez Microsoft w 2012 roku [14]. Jednym z najważniejszych udoskonaleń względem swojego poprzednika, jest dodanie funkcjonalności programowania obiektowego takiego jak klasy, czy też interfejsy.



Rys. 1.5. Logo TypeScript [15].

Jednym z elementów, który został dodany dzięki temu rozszerzeniu jest możliwość *dziedziczenia*. Ponadto bardzo oczekiwanym ulepszeniem jest *typowanie statyczne*, pozwalające z góry określić typ danego elementu, wręcz przeciwnie jak w przypadku *typowania słabego*, mogącego zmieniać się w zależności od potrzeb w danym momencie. Jako, że *TypeScript* jest rozszerzeniem *JavaScript*, każdy poprawny kod w *JS* jest również poprawny w *TS*.

1.2.2. Technologie wykonywane po stronie serwera

Funkcjonalności działające po stronie serwera nie są widoczne dla klienta danej aplikacji. Klient dostaje informacje zwrotną w momencie żądania o składową systemu. Jednymi z najpopularniejszych technologii umożliwiającymi wykonanie po stronie serwera są:

- PHP,
- Node.js,
- Python,
- Java.

PHP jest to język skryptowy działający po stronie serwera. Jest on jednym z najpopularniejszych technologii używanych do tworzenia stron internetowych, gdyż łatwo można go umieścić w kodzie strony. W sytuacji, gdy klient odzywa się z żądaniem o wywołanie funkcji, skrypt działający po stronie serwera zostaje uruchomiony i odpowiada konkretną funkcjonalnością.

Node.js jest to wieloplatformowe środowisko wykonawcze JavaScript. Udogatępnia taką funkcjonalność jak zestaw asynchronicznych operacji. Ma charakter serwerowy. Jest rozwiązaniem open source, chętnie wykorzystywanym przez rzesze programistów.

Python jest to język programowania, który daje możliwość wyboru wzorca programowania. Znaczącą zaletą przemawiającą na jego korzyść jest paradygmat obiektowy. Umożliwia tworzenie serwera przy użyciu frameworka *Flask* lub *Django* [16].

Java jest to uniwersalny język programowania. Powstała w 1991 roku w Sun Microsystems w Mountain View pod początkową nazwą *Oak*. Wywodzi się z dwóch języków programowania: *Smalltalk* i *C++* [17]. Stworzony został przez grupę osób pod kierownictwem Jamesa Goslinga. Opiera się na paradygmacie programowania obiektowego. Został zaprojektowany jako oddzielny byt, którego kod źródłowy nie jest zgodny z żadnym innym językiem programowania. Jednym z głównych zamysłów twórców Javy było stworzenie takiego języka, który dałby możliwość wytworzenia oprogramowania działającego na każdym urządzeniu. Aplikacje stworzone w Javie muszą być wykonywane przez wirtualną maszynę, aby mogły zostać zrozumiane przez system.

1.3. Protokół HTTP

HTTP (ang. *HyperText Transfer Protocol*) jest podstawowym protokołem WWW (ang. *World Wide Web*). W momencie interakcji internetowej jest on wywoływany. Służy do rozpowszechniania informacji przez Internet. Protokół HTTP jest używany przy każdym żądaniu grafiki na stronie internetowej, odnośniku do stron internetowych, czy też formularzu dostępnym na stronie. Określa on w jaki sposób wygląda połączenie na poziomie klient - serwer, w jaki sposób klient oczekuje na dane i w jaki sposób serwer odpowiada na otrzymane informacje.

Podstawowymi transakcjami HTTP są:

- GET
- POST
- PUT
- DELETE

Metody te odpowiadają kolejno operacjom odczytywania, tworzenia, aktualizacji i usuwania. Każda z metod po wykonaniu żądania otrzymuje kod statusu informujący o stanie zrealizowania żądania. Można je podzielić na pięć grup kodów.

- Informacyjne 1xx

- Sukcesu 2xx
- Przekierowania 3xx
- Błędu klienta 4xx
- Błędu serwera 5xx

Kody z pierwszej grupy wskazują na odpowiedź tymczasową, po której klient czeka na kolejne komunikaty. Kody należące do grupy drugiej wskazują, że żądanie od klienta zostało pomyślnie przetworzone. Trzecia grupa kodów odpowiada za przekierowania do nowych zasobów które zostały zastąpione. Czwarta grupa odpowiada za błędy po stronie klienta które wskazują problemy z danym zapytaniem. Kody z ostatniej grupy informują o błędach po stronie serwera, który uniemożliwił spełnienie przez niespodziewany błąd [18].

1.4. RESTful web services

REST (ang. *Representational State Transfer*) jest to struktura wytwarzania oprogramowania umożliwiająca rozwijanie usług internetowych [19]. Daje możliwość tworzenia serwisów które, są lekkie, skalowalne oraz są proste w procesie wspierania jak i konserwacji. Usługi zbudowane na tej architekturze są nazywane usługami RESTful. Jednym z głównych elementów należących do takowych serwisów są zasoby, które są danymi jak i funkcjami udostępnianymi za pomocą ujednoliconego identyfikatora zasobów (URI, ang. *Uniform Resource Identifier*).

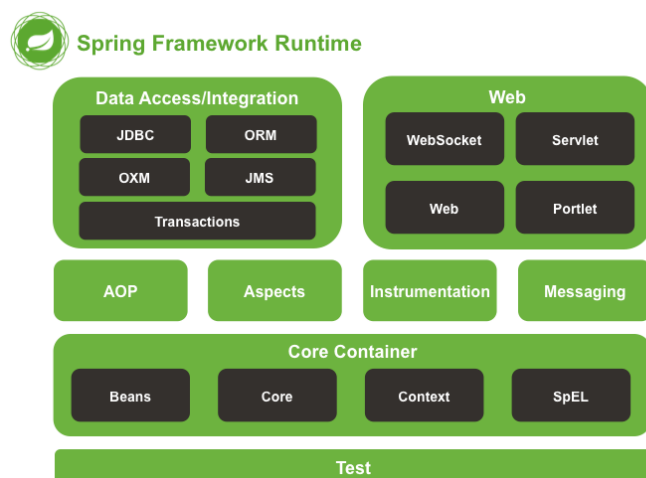
Rozdział 2

Wykorzystywane oprogramowanie

2.1. Spring Framework

Spring jest to najczęściej wykorzystywany szkielet aplikacji do tworzenia oprogramowania w języku Java. Przy jego użyciu w prosty sposób można tworzyć kod, który jest w przystępny sposób testowany, niejednokrotnie wykorzystywany oraz bardzo efektywny pod względem wydajnościowym. Został on stworzony przy założeniach takich jak wstrzykiwanie zależności (DI, ang. *Dependency Injection*) [20]. Spring powstał w 2003 roku. Dostarcza on dużą liczbę modułów podzielonych na grupy pod względem zastosowania.

Możemy wyróżnić kilka grup. Każda z nich posiada zestaw modułów które mogą zostać dostarczone podczas tworzenia aplikacji:



Rys. 2.1. Podział na poszczególne moduły [21].

Przykładowe moduły wchodzące w skład grup:

- spring-core,
- spring-beans,
- spring-aop,
- spring-messaging,
- spring-web,
- spring-jdbc,
- spring-orm.

Moduły te dostarczają wiele funkcjonalności takich jak połączenie z bazą danych, mapowanie obiektów, infrastrukturę transakcyjną, pakiety wsparcia dla *WEB*, *REST Web Service*, oraz cały szereg pożytecznych operacji niwelujących nadmiarowy kod.

2.2. Spring Boot

Konfiguracja projektu na podstawie modułów wchodzących w skład Spring Framework należy do procesu długiego i bardzo rozbudowanego. Wynika to ze względu na dużą liczbę zależności będących skutkiem z połączenia między sobą odpowiednich funkcjonalności modułów. Aby ułatwić proces tworzenia aplikacji na bazie Spring Framework, stworzono nowy byt o nazwie Spring Boot. Umożliwia on utworzenie nowego projektu oraz dostarczenie niezbędnej startowej konfiguracji, umożliwiającej uruchomienie aplikacji z podstawową funkcjonalnością [22].

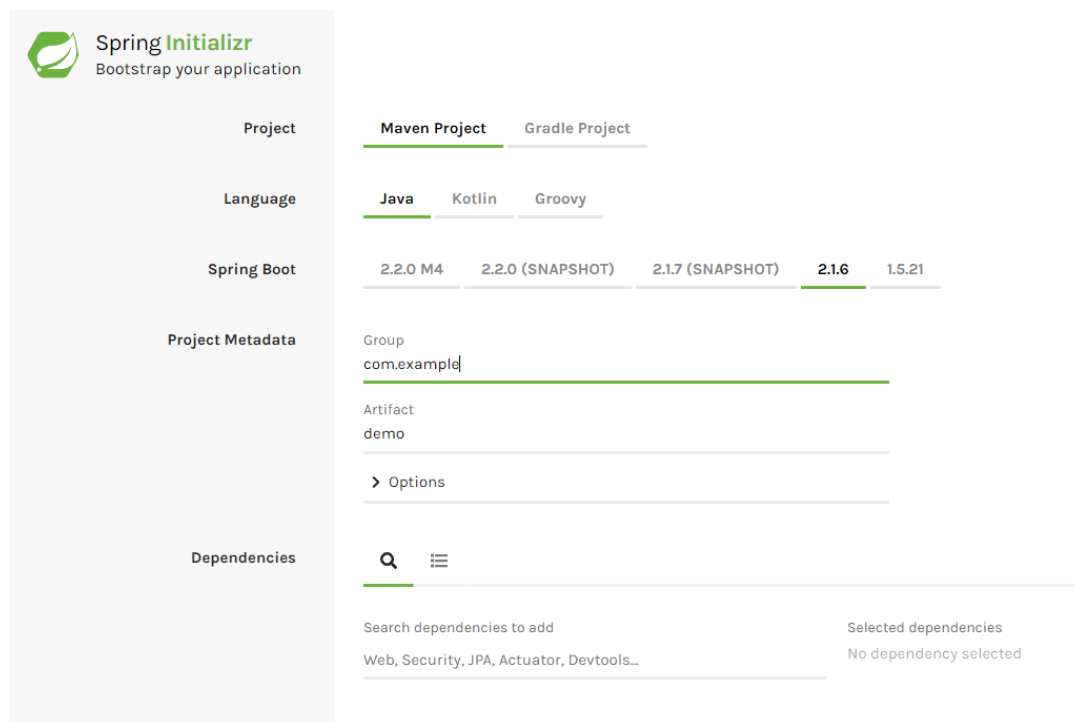
Podstawowymi założeniami przy tworzeniu aplikacji za pomocą Spring Boot jest:

- Przyspieszenie tworzenia startowego projektu przy użyciu modułów niezbędnych dla aplikacji.
- Automatyczna konfiguracja zależności między modułami.
- Dostarczenie początkowej konfiguracji.

- Zaopatrzenie w dodatki występujące zawsze w projektach (zabezpieczenia, metryki, wbudowane serwery).
- Uniknięcie konfigurowania XML w Spring.

2.3. Spring Initializr

Spring Initializr jest to witryna, która umożliwia w prosty sposób wygenerowanie projektu Spring Boot. Podczas tworzenia istnieje możliwość wyboru kilku konfiguracji. Przy wyborze dostępne są następujące opcje: rodzaj projektu, wspierany język, wydanie i wersja Spring Boot, metadane oraz wybór modułów [23].



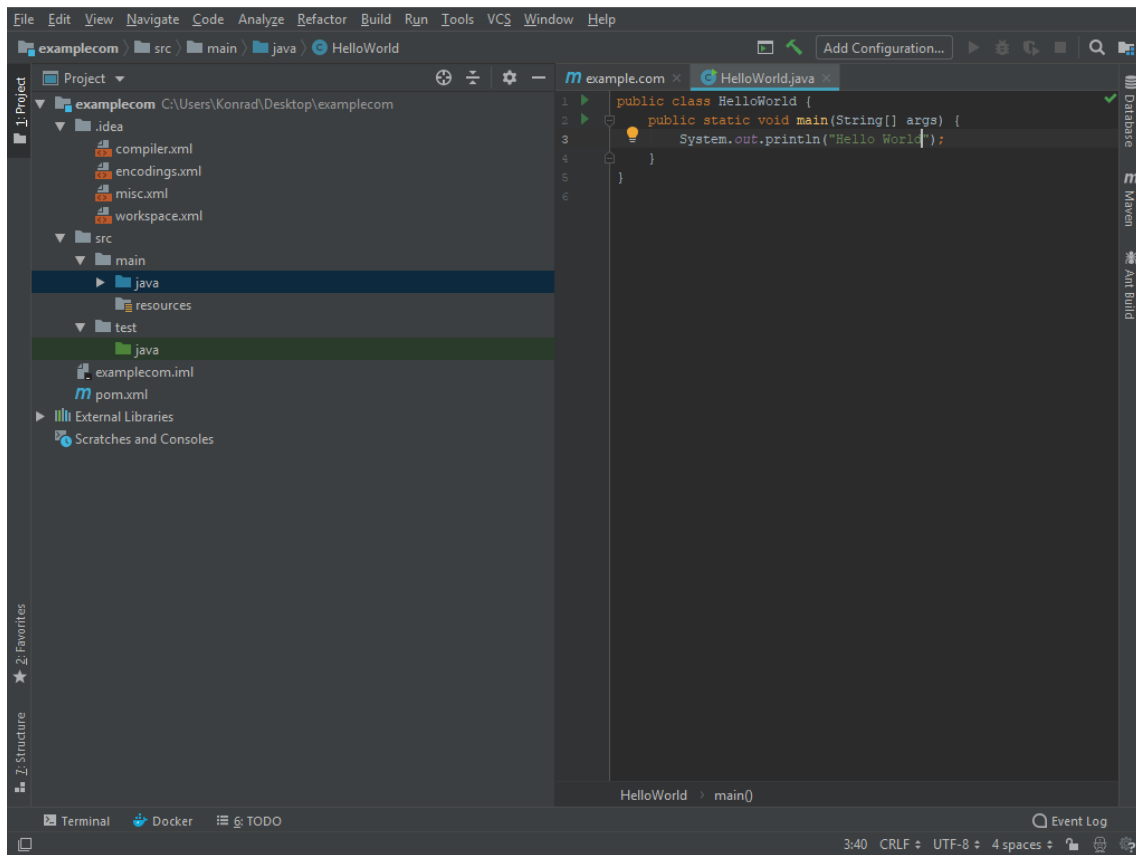
The screenshot shows the Spring Initializr web application. On the left is a sidebar with navigation links: Project, Language, Spring Boot, Project Metadata, and Dependencies. The main content area is titled 'Spring Initializr Bootstrap your application'. It features tabs for 'Maven Project' (selected) and 'Gradle Project'. Under 'Maven Project', there are tabs for 'Java' (selected), 'Kotlin', and 'Groovy'. Below these are version selection buttons for Spring Boot: '2.2.0 M4', '2.2.0 (SNAPSHOT)', '2.1.7 (SNAPSHOT)', '2.1.6' (selected), and '1.5.21'. The 'Project Metadata' section contains input fields for 'Group' (com.example) and 'Artifact' (demo), followed by an 'Options' section. The 'Dependencies' section at the bottom has a search bar with a magnifying glass icon and a list of dependencies to add, including 'Web, Security, JPA, Actuator, Devtools...'. To the right of the search bar, it says 'Selected dependencies' and 'No dependency selected'.

Rys. 2.2. Strona główna Spring Initializr [24].

2.4. IntelliJ IDEA

IntelliJ IDEA jest to zintegrowane środowisko programisty do tworzenia oprogramowania w języku programowania Java [25]. Zostało stworzone przez JetBrains. Występuje ono w dwóch wariantach. Pierwsze wydanie to darmowa i ogólnodostępna wersja, natomiast druga jest to wersja płatna, zawierająca wsparcie

dla wielu frameworków dostępnych dla języka Java oraz gwarantująca pełną integrację z nimi.



Rys. 2.3. Widok ze środowiska IntelliJ IDEA.

2.5. Apache Maven

Apache Maven jest to narzędzie umożliwiające w prosty sposób zarządzanie projektami [26]. Daje możliwość wytworzenia projektu oraz zarządzania niezbędnymi zależnościami i dokumentacją. Jego struktura opiera się na POM (ang. *Project Object Module*) który jest plikiem XML. Składowymi elementami są: informacje związane z konfiguracją zależności, informacjami związanymi z samym projektem, a także rozplanowaniem położenia katalogu głównego.



Rys. 2.4. Logo Apache Maven [27].

Aktualnie na rynku rozwiązań tego typu, możemy nadmienić inny produkt o niemal identycznym działaniu pod względem zarządzania projektami. Oprogramowanie to nazywa się Gradle [28]. Jest ono oparte na języku programowania Groovy. Od 2013 roku, Google ustanowiło go standardem dla tworzenia aplikacji na platformie Android.

Każdy plik POM składa się z kilku sekcji. Poprawność jego działania zapewnia podstawowa konfiguracja przedstawiona poniżej.

Listing 2.1: Minimalna konfiguracja pliku POM.

```
<?xml version="1.0" encoding="utf-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>example.com</groupId>
  <artifactId>example.com</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

Głównym elementem jest znacznik `<project>`, który w swoim obrębie zawiera pozostałe składowe:

- `<modelVersion>` – definiuje wersję narzędzia, zgodną z POM,
- `<groupId>` – reprezentuje unikalny identyfikator dla projektu,
- `<artifactId>` – określa nazwę produkowanego artefaktu przez projekt,
- `<version>` – informuje o aktualnej wersji artefaktu [26].

Dodatkowymi elementami najczęściej występującymi w pliku POM są zależności, które umożliwiają dostarczenie zewnętrznych modułów do projektu. Proces ich dodawania jest prosty. Polega na odnalezieniu modułu w witrynie udostępnionej przez Apache Maven.

Listing 2.2: Przykładowa zależność.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
```

```
<version>2.1.6.RELEASE</version>  
<scope>test </scope>  
</dependency>
```

2.6. Angular

2.6.1. Historia

Angular jest to framework umożliwiający tworzenie rozwiązań o charakterze mobilnym, desktopowym czy również dający możliwość tworzenia aplikacji internetowych. Jest on obecnie jednym z najbardziej rozpoznawalnych narzędzi do tworzenia aplikacji SPA [29].

Historia powstania tego rozwiązania ma już blisko dekadę. Początkowa wersja tego frameworka miała nazwę AngularJs i na przestrzeni ostatnich lat ewoluowała w Angular. Twórcą prototypu był Misko Heaverty, który doprowadził do wytworzenia go w 2009r. Po wydaniu pierwszej wersji, twórcy postanowili rozpocząć pracę nad kolejną, która została stworzona od podstaw.

Angular 2 został stworzony przy użyciu języka TypeScript. Główną zaletą kolejnej edycji okazało się poprawienie wydajności podczas tworzenia witryn. Dodatkowym atutem było stworzenie architektury opartej na komponentach oraz umożliwienie tworzenia aplikacji w oparciu o TypeScript.



Rys. 2.5. Logo Angular [30].

Kolejną wersją był Angular 4. Zmiany, które zostały w niej wprowadzone usprawiły działanie aplikacji, bazując na zmniejszeniu liczby dołączonych domyślnie ustawień. Wersja ta umożliwiała kompatybilność wsteczną.

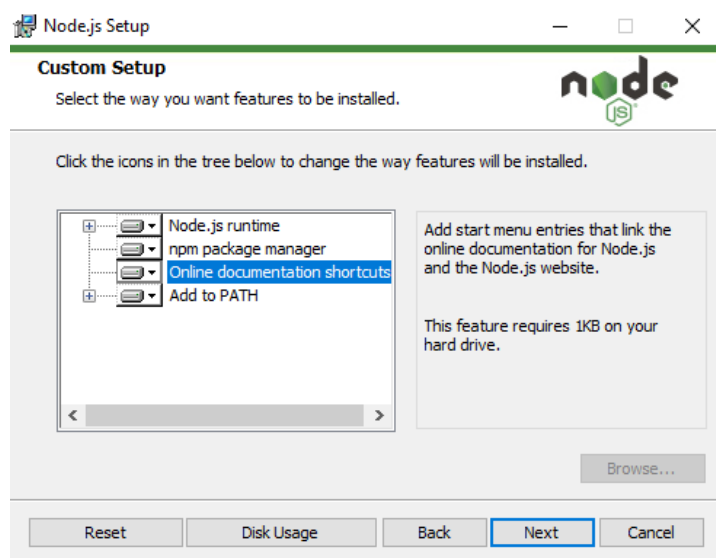
Kolejną stworzoną wersją był Angular 5, który w dużej mierze poprawił użyteczność kompilatora. Jedną z istotniejszych zmian było zoptymalizowanie działania klienta HTTP.

W przypadku Angulara w wersji 6 zmniejszono jego rozmiary i przyspieszono działanie, a ponadto ułatwiono tworzenie oprogramowania z perspektywy programisty.

Najnowsza wersja 7 została wydana w październiku 2018r. W tej odsłonie Angular, został usprawniony o niewielkie zmiany. Między innymi oprócz poprawienia wydajności względem poprzedniej wersji, wprowadzono podpowiedzi działań dla wiersza wbudowanego w projekt. Ponadto została udostępniona możliwość integracji z rozwiązaniem Material UI, udostępniającym gotowe komponenty do tworzenia wizualnej strony oprogramowania.

2.6.2. Instalacja Node.js

W celu korzystania z Angular, niezbędna jest instalacja Node.js, dzięki któremu będzie można wykorzystać *Node Package Manager* [31]. Pierwszym krokiem jest pobranie najnowszej wersji ze strony producenta: <https://nodejs.org/en/download/>. Po wybraniu wersji odpowiadającej dla danego systemu, konieczne jest uruchomienie instalatora. W kolejnym kroku procesu instalacji jesteśmy proszeni o wybranie lokalizacji.

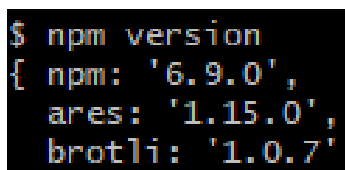


Rys. 2.6. Proces instalacji Node.js.

Następnie o zaznaczenie elementów, które mają zostać zainstalowane. Rekomendowany jest wybór wszystkich dostępnych narzędzi. Po tym kroku następuje proces instalowania wybranych elementów. Instalator w ostatnim etapie informuje o statusie instalacji i stanie finalnym.

2.6.3. Node Package Manager

Node Package Manager jest narzędziem, które ma na celu zarządzanie pakietami w środowisku Node.js. Przy jego pomocy można korzystać z zewnętrznych modułów udostępnionych przez twórców [32]. W momencie zainstalowania Node.js, narzędzie to jest gotowe do użycia. W celu sprawdzenia poprawności działania niezbędne jest wywołanie komendy zwracającej aktualną wersję *NPM*.



```
$ npm version
{ npm: '6.9.0',
  ares: '1.15.0',
  brotli: '1.0.7',
  ... }
```

Rys. 2.7. Przedstawienie wersji NPM.

Po wykonaniu, wyświetlana jest aktualna wersja oraz wszystkie powiązane zależności, które są niezbędne do prawidłowego działania.

2.6.4. Angular CLI

Interfejs wiersza poleceń czyli **CLI** (ang. *Command Line Interface*) jest to narzędzie do tworzenia komponentów, serwisów, nawigacji oraz do automatycznego zarządzania zależnościami [33].

2.6.5. Instalacja Angular CLI

Aby posłużyć się tym szkieletem do budowy aplikacji, niezbędna jest instalacja *Angular CLI* przy użyciu *Node Package Manager*. Do poprawnego zainstalowania należy wywołać polecenie, które ma za zadanie pobranie tego narzędzia z repozytorium.

```
npm install -g @angular/ cli
```

2.7. JHipster

Aktualnie programiści dążą do tworzenia oprogramowania umożliwiającego uniknięcie powtarzalnego kodu, przy projektach, w zakresie ich stosu technologicznego [34]. Tworzenie kodu o takim charakterze i konfigurowanie go jest procesem długotrwałym. Integracja wszystkich składowych i elementów podrzędnych oraz doprowadzenie ich do stanu umożliwiającego uruchomienie z zewnętrznymi modułami, jest często bardzo skomplikowane. Aby uprościć sposób stworzenia i skonfigurowania startowego projektu, z pomocą przychodzi JHipster.

JHipster jest to narzędzie umożliwiające wygenerowanie podstawowego projektu Spring Boot, z dowolnym z trzech rozwiązań: Angular, Vue, React. Dodatkowo daje on możliwość dodania obsługi dodatkowych narzędzi, przydatnych przy tworzeniu rozbudowanych aplikacji i ich podstawową konfigurację. Umożliwia on wybranie takich elementów, jak rodzaj bazy danych, wybór narzędzia do zarządzania projektem, obsługę wyszukiwarki, która usprawnia prędkość zapytań przy dużej ilości danych, obsługę asynchronicznych wiadomości, czy też wsparcie dla narzędzi do testowania.



Rys. 2.8. Oficjalne logo JHipster [35].

Narzędzie to nieustannie zyskuje na popularności. Od momentu powstania, liczba osób, która wykorzystuje to rozwiązanie wzrasta, przekraczając miesięcz-

nie średnio ponad sto tysięcy wygenerowań projektów. Aktualnie liczba zaangażowanych w projekt sięga ponad pięciuset programistów.

2.7.1. Instalacja

W celu stworzenia startowego projektu przy użyciu generatora, potrzebna jest jego instalacja za pomocą *Node Package Module*, czyli menadżera pakietów dla środowiska *Node.js*. Proces instalacji przy poprawnie zainstalowanych narzędziach wymienionych powyżej jest prosty. Ogranicza się do wywołania jednej komendy na rzecz menadżera pakietów.

Listing 2.4: Komenda do instalacji JHipster.

```
npm install -g generator-jhipster
```

Rozdział 3

Tworzenie aplikacji

Głównym założeniem przedstawionej pracy jest zilustrowanie procesu tworzenia aplikacji przy użyciu Spring Boot i Angular framework. W ramach ukazania rozwoju oprogramowania, powstanie aplikacja umożliwiająca zobrazowanie w prosty sposób tego zjawiska.

3.1. Założenia tworzonej aplikacji

Przedstawiony projekt jest aplikacją udostępniającą listę wydarzeń kulturalnych tworzonych przez użytkowników. Głównymi elementami są: możliwość dodawania, edytowania i usuwania wydarzeń w zależności od uprawnień osoby korzystającej. Elementem nieodłącznym, który jest niezbędny do sprawdzenia wszystkich dostępnych wydarzeń, jest rejestracja konta, a następnie zalogowanie do witryny.

3.2. Dobór bazy danych

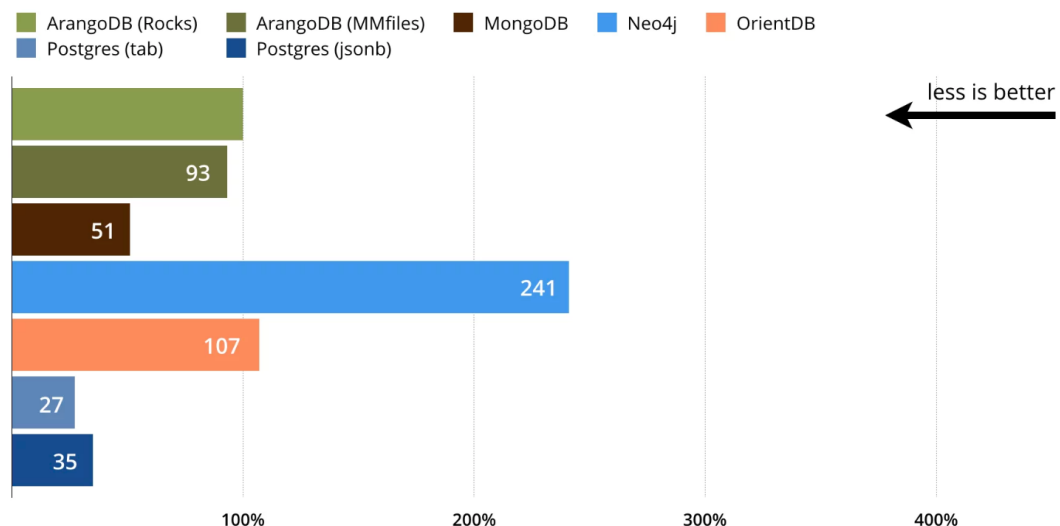
Aktualnie wszyscy programiści dążą do korzystania z baz danych, które działają stabilnie. Dodatkowym atutem, uwzględnianym przy doborze bazy, jest jej wydajność. Obecnie występuje duża ilość tego typu rozwiązań, które przyciągają programistów, ze względu na prędkość wykonywania zapytań, stabilność czy też zużycie zasobów urządzenia, na którym znajduje się baza.

Najpopularniejszymi systemami do zarządzania bazami są:

- Oracle Database,
- Microsoft SQL Server,
- PostgreSQL,
- MongoDB,
- MySql.

Obecnie wybór bazy danych to ciężka decyzja, ze względu na wiele czynników, zależnych od aktualnego stanu i potrzeb programisty lub całego zespołu tworzącego oprogramowanie. W pracy wykorzystano PostgreSQL, czyli jeden z najbardziej rozpoznawalnych systemów, który umożliwia zarządzanie relacyjną bazą danych [36]. Atutem tego rozwiązania jest bardzo dobra wydajność przy małych i średnich projektach. Dodatkowo, korzystanie z niego nie jest obciążone dużym wykorzystaniem zasobów.

Memory Consumption



Rys. 3.1. Zużycie pamięci [37].

Oprócz tego, twórcy udostępniają bardzo dobrze opisaną dokumentację, gwarantującą zaznajomienie się z działaniem oraz wykorzystaniem tego oprogramowania. Opcjonalnie, w przypadku problemów, istnieje możliwość otrzymania

wsparcia, poprzez sekcję skierowaną dla społeczności, w której można uzyskać pomoc.

3.3. Generowanie podstawowego projektu

W celu uniknięcia tworzenia nadmiarowego kodu, za pomocą narzędzia JHipster zostanie wygenerowany startowy projekt, który udostępni początkową konfigurację. Dzięki temu rozwiązaniu, programista może skupić się na wytworzeniu aplikacji pod względem wizualnym oraz logiki działania [34].

Listing 3.1: Komenda do wygenerowania startowego projektu.

```
jhipster
```

W kolejnym kroku, niezbędnym elementem jest wybór nazwy dla projektu, a następnie wsparcia dla monitorowania zasobów, z których korzysta aplikacja. Ponadto, należy dobrać odpowiednią bazę wykorzystywaną podczas procesu tworzenia, a także etapu gdy jest upubliczniona. Końcowym punktem procesu jest dobór narzędzia do zarządzania projektem, czy też rozwiązania po stronie wizualnej.

```
Documentation for creating an application is at https://www.jhipster.tech/creating-an-app/
If you find JHipster useful, consider sponsoring the project at https://opencollective.com/generator-jhipster

WARNING! Your Node version is not LTS (Long Term Support), use it at your own risk! JHipster does not support non-LTS releases, so if you encounter a bug, please use a LTS version first.
? Which *type* of application would you like to create? Monolithic application (recommended for simple projects)
? What is the base name of your application? jevents
? What is your default Java package name? com.jevents
? Do you want to use the JHipster Registry to configure, monitor and scale your application? Yes
? Which *type* of database would you like to use? SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL)
? Which *production* database would you like to use? PostgreSQL
? Which *development* database would you like to use? PostgreSQL
? Do you want to use the Spring cache abstraction? Yes, with the Ehcache implementation (local cache, for a single node)

? Do you want to use Hibernate 2nd level cache? Yes
? Would you like to use Maven or Gradle for building the backend? Maven
? Which other technologies would you like to use? (Press <space> to select, <a> to toggle all, <i> to invert selection)
? Which *Framework* would you like to use for the client? Angular
? Would you like to use a Bootswatch theme (https://bootswatch.com/)? Default JHipster
? Would you like to enable internationalization support? No
? Besides JUnit and Jest, which testing frameworks would you like to use? (Press <space> to select, <a> to toggle all, <i> to invert selection)
? Would you like to install other generators from the JHipster Marketplace? No
```

Rys. 3.2. Widok z generatora.

3.4. Uruchomienie aplikacji

Aby uruchomić serwer niezbędne jest wykonanie dwóch komend, na rzecz *Mavena*, który zarządza projektem [26]. Pierwsza z nich jest odpowiedzialna za wyczyszczenie plików oraz katalogów, generowanych podczas tworzenia projektu. Natomiast kolejna buduje projekt oraz wszystkie artefakty, przy użyciu pliku POM.

Listing 3.2: Komendy do zbudowania serwera.

```
mvn clean
mvn install
```

W przypadku wizualnej części oprogramowania, zbudowanie jej ogranicza się do jednej komendy, wywołanej na rzecz wiersza poleceń.

Listing 3.3: Komenda do wystartowania warstwy wizualnej.

```
npm start
```

Atutem tego rozwiązania jest fakt, że w momencie jakiegokolwiek zmiany w kodzie, widok zostaje automatycznie odświeżony i zaktualizowany.

3.5. Połączenie z bazą danych

Niezbędnym elementem, który jest ważnym filarem do działania aplikacji jest ustanowienie połączenia z bazą danych. W celu skonfigurowania tego aspektu, kluczowe jest zmodyfikowanie pliku odpowiedzialnego za zarządzanie [38].

Listing 3.4: Zmodyfikowany plik application.yml.

```
datasource :
  url : jdbc:postgresql://localhost:5432/jevents
  username: postgres
  password: root
```

jpa :

database : POSTGRESQL

Głównymi składowymi, które muszą zostać zdefiniowane do prawidłowego działania połączenia, jest url, w którym określa się system bazy danych, adres IP urządzenia, gdzie znajduje się baza, udostępniony port oraz nazwa stworzonej bazy. Kolejnymi elementami jest login i hasło, zapewniające dostęp do bazy w aplikacji.

3.6. Model

Programiści, w procesie planowania i wytwarzania oprogramowania dążą do jak najlepszego odzwierciedlenia świata rzeczywistego. Stworzenie odpowiedniego modelu klas, które spełniają postawione przed nimi zadania bywa czasochłonne. Na rzecz pracy zostanie przedstawiony model wydarzenia kulturowego. Składa się z kilku atrybutów, które pomagają określić podstawowe informacje. Elementami składowymi są: nazwa, opis, miejsce, czas wydarzenia, zdjęcie związane z tą okolicznością i nazwa użytkownika udostępniającego informacje. Klasy przedstawiające zdjęcia i użytkownika, zawierają szczegółowe informacje, które je definiują. Przedstawiony poniżej model obrazuje urywek klasy reprezentującej wydarzenie.

Listing 3.5: Plik Events.java.

```
@Entity
@Table(name = "events")
public class Events {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
        generator = "sequenceGenerator")
    @SequenceGenerator(name = "sequenceGenerator")
    private Long id;
```

```
@Column(name = "event_name")
private String eventName;
@OneToOne
private Photo photo;
@ManyToOne
private User user;
...
}
```

W celu umożliwienia mapowania obiektowo-relacyjnego niezbędne jest dodanie adnotacji, które umożliwią rozpoznanie konkretnych atrybutów i automatyczne odwzorowanie ich w bazie [22]. W celu poprawności przebiegu mapowania, należy określić kilka adnotacji, by w przypadku interakcji z bazą danych, narzędzie rozpoznało do jakiej tabeli odnosi się dany obiekt.

Wykorzystane adnotacje:

- *@Entity* – oznaczenie encji, które jest elementem wymaganym do rozpoznania przy mapowaniu,
- *@Table* – oznaczenie tabeli, określa szczegóły dotyczące tabeli, która zostanie zapisana w bazie,
- *@Id* – oznaczenie id, określa klucz główny dla tabeli, który zostanie odzwierciedlony w bazie,
- *@SequenceGenerator* – oznaczenie generatora, używane jest do stworzenie sekwencji w bazie danych
- *@GeneratedValue* – oznaczenie, które ma na celu włączenie automatycznego generowania id w sposób inkrementalny, przyjmuje dodatkowo parametr *generator*, który korzysta z sekwencji o podanej nazwie,
- *@Column* – oznaczenie kolumny, służy do określenia szczegółów kolumny, która zostanie odwzorowana w bazie,
- *@OneToOne* – oznaczenie przedstawiające powiązanie z inną tabelą w relacji jeden do jednego,

- *@ManyToOne* – oznaczenie przedstawiające powiązanie z inną tabelą w relacji wiele do jednego.

3.7. Plik startowy serwera

W procesie tworzenia oprogramowania logika aplikacji jest najważniejszym elementem. Procesy, które dzieją się po stronie serwera, umożliwiają użytkownikom otrzymywanie informacji dla nich udostępnionych, a następnie są wyświetlane przy użyciu komponentów wytworzonych po stronie wizualnej. Podczas konstrukcji serwera, który udostępni logikę związaną z operacjami na danych, wymagane jest stworzenie pliku startowego dla aplikacji przy użyciu Spring Boot.

Listing 3.6: Minimalna konfiguracja pliku startowego - *JeventsApp.java*.

```
@SpringBootApplication
public class JeventsApp{
    public static void main(String [] args) {
        SpringApplication app =
            new SpringApplication (JeventsApp.class);
        app.run();
    }
}
```

Głównymi elementami umożliwiającymi poprawne działanie aplikacji, jest adnotacja *@SpringBootApplication* oraz stworzenie obiektu klasy *SpringApplication* i wywołanie na nim metody, która umożliwia włączenie serwera *app.run()*. Adnotacja ta służy do konfigurowania projektu Spring Boot. Pod *@SpringBootApplication* kryją się dodatkowe adnotacje, które odpowiadają za konfigurację ścieżki, kontekstu aplikacji, a także wyszukiwanie komponentów, oraz usług [20].

Elementy wchodzące w skład wymienionej adnotacji:

- *@Configuration*,
- *@EnableAutoConfiguration*,
- *@ComponentScan*.

3.8. Repozytorium

Jednym z rozwiązań dostarczanych przez *Spring* jest możliwość tworzenia interfejsu, który jest rozszerzany przez *JpaRepository* [39]. Rozwiązanie to umożliwia stworzenie automatycznej implementacji podstawowych operacji, takich jak dodawanie, usuwanie, modyfikowanie czy pobieranie obiektu. W sytuacji, gdy potrzebne jest inne zapytanie do bazy, umożliwiające jest stworzenie natywnego zapytania lub funkcji, która rozpoznaje po nazwie rodzaj żądania.

Listing 3.7: Interfejs udostępniający operacje na bazie.

```
@Repository
public interface EventsRepository extends
    JpaRepository<Events, Long> {
    List<Events> findByEventName( String eventName );
}
```

Poprzez rozszerzanie interfejsu przez *JpaRepository* z parametrami *Events* i *Long*, Spring rozpoznaje, że operacje będą wykonywane na obiektach klasy *Events*. W automatyczny sposób zostanie zweryfikowane, poprzez adnotacje znajdujące się w klasie *Events*, do jakiej tabeli odnosi się zapytanie.

3.9. Serwis

Elementem nieodłącznym przy części serwerowej jest stworzenie serwisu. Serwisy to klasy używane do zapisania logiki biznesowej w oddzielnej warstwie oraz wspomagające czytelność kodu. Każdy stworzony serwis, ma za zadanie odwoływać się do repozytorium zajmującego się określoną encją [40].

Listing 3.8: Interfejs dla serwisu.

```
public interface EventsService {
    Events save(Events events);
    Page<Events> findAll( Pageable pageable );
}
```

```
Optional<Events> findOne(Long id);  
void delete(Long id);  
}
```

W celu zbudowania serwisu, rekomendowane jest stworzenie interfejsu, który udostępni wymagane metody. Powinien on zostać zaimplementowany w serwisie zajmującym się operacjami na rzecz repozytorium.

Listing 3.9: Implementacja serwisu.

@Service

```
public class EventsServiceImpl implements EventsService {  
    private final EventsRepository eventsRepository;  
    public EventsServiceImpl(EventsRepository eventsRepository) {  
        this.eventsRepository = eventsRepository;  
    }  
    @Override  
    public Events save(Events events) {  
        return eventsRepository.save(events);  
    }  
    ...  
}
```

Adnotacja *@Service* służy do poinformowania *Spring*, że klasa jest wykorzystywana jako serwis. Niezbędnym elementem, jest dodanie repozytorium i zainicjowanie go przez konstruktor. Implementowanie *EventsService* wymaga stworzenia funkcjonalności i posłużenie się nią na rzecz repozytorium.

3.10. Kontroler

Zadaniem kontrolera jest udostępnienie informacji w momencie, gdy zostanie wysłane żądanie HTTP, na konkretny adres. W przypadku, gdy serwer je dostaje, rozpoczyna proces wszystkich niezbędnych operacji, które mają na celu zwrócenie informacji do podmiotu żądającego zasobu [40].

Listing 3.10: Kontroler.

```
@RestController
@RequestMapping("/api")
public class EventsResource {
    private static final String ENTITY_NAME = "events";
    private static final String APPLICATION_NAME = "jevents";
    private final EventsService eventsService;
    public EventsResource(EventsService eventsService) {
        this.eventsService = eventsService;
    }
    @PostMapping("/events")
    public ResponseEntity<Events> createEvents(
        @RequestBody Events events) throws URISyntaxException {

        Events result = eventsService.save(events);
        return ResponseEntity.created(new URI("/api/events/"
            + result.getId()))
            .headers(HeaderUtil.createEntityCreationAlert(
                applicationName, false, ENTITY_NAME,
                result.getId().toString()))
            .body(result);
    }
    ...
}
```

Wymagany elementem jest dodanie instancji *EventsService* oraz zainicjowanie jej w konstruktorze.

Kontroler można podzielić na kilka składowych:

- *@RestController* – adnotacja, która jest specjalistyczną odmianą kontrolera. Zawiera w sobie dodatkowo *@ResponseBody*, która odpowiada za poinformowanie, że zwracana informacja ma być w formacie JSON.

- *@RequestMapping* – adnotacja, ma na celu mapowanie żądań, ze wskazanego adresu na metody w kontrolerze.
- *@PostMapping* – adnotacja mająca na celu obsługę żądania HTTP typu POST na wskazany adres. Operacje GET, PUT, DELETE, również posiadają analogiczną adnotację. Każda z nich służy jako skrót do odpowiadającej im adnotacji *RequestMapping(method = RequestMethod.POST)*.
- *@RequestBody* – adnotacja, która umożliwia automatyczne mapowanie przychodzącego żądania na obiekt.
- *ResponseEntity* – metoda, która daje możliwość zaprezentowania odpowiedzi HTTP, poprzez możliwość konfiguracji nagłówków, zwracanego kodu oraz treści.

3.11. Model w kliencie

Podstawowym wymogiem jest odwzorowanie modelu, w taki sposób, aby zgadzał się z przedstawionym w bazie danych. Proces ten, jest bardzo zbliżony do przedstawionego mu modelowania podczas tworzenia części serwerowej. Głównym zadaniem jest dopasowanie odpowiedniego typu zmiennej, aby zgadzał się z przedstawionym w bazie danych. Praktyką stosowaną i zalecaną przez autorów tego rozwiązania, jest stworzenie interfejsu dla modelu, który następnie będzie implementowany przez klasę ją reprezentującą.

Listing 3.11: Reprezentacja modelu w Type Script.

```
export interface IEvents {
    id?: number;
    eventName?: string;
    photo?: IPhoto;
    user?: IUser;
    ...
}

export class Events implements IEvents {
```

```
constructor(  
    public id?: number,  
    public eventName?: string,  
    public photo?: IPhoto,  
    public user?: IUser  
    ...  
)~{}
```

3.12. Moduł Angular

Moduł w Angular jest to mechanizm do grupowania poszczególnych elementów takich jak usługi, dyrektywy, komponenty, które są ze sobą znaczeniowo powiązane [41]. Proces wygenerowania modułu, jest bardzo szybki. Sprowadza się do wywołania *ng generate module events*.

Listing 3.12: Przedstawienie modułu.

```
@NgModule({  
    imports: [  
        JeventsSharedModule,  
        MatCardModule,  
        ...  
    ],  
    declarations: [  
        EventsComponent,  
        EventsItemComponent,  
        ...  
    ],  
})
```

NgModule jest to oznaczenie klasy, odpowiedzialne za pobieranie informacji. Moduły umożliwiają wykorzystanie zewnętrznych bibliotek, poprzez zadeklarowanie ich w *imports*. Dodatkowo określają, które komponenty są częścią modułu

oraz zapewniają ich obsługę. Element *declarations* odpowiada za tworzenie dyrektyw, które znajdują się w zakresie modułu.

3.13. Serwis klienta

Serwisy mają za zadanie pobieranie i zapisywanie danych oraz udostępnianie informacji do komponentów, po wcześniejszym żądaniu HTTP skierowanym w stronę serwera, który odpowiada za wykonanie wymaganych operacji [42]. Używane komponenty powinny jedynie prezentować informację. Ponadto, aby stworzyć serwis, potrzebne jest wywołanie *ng generate service events* na rzecz Angular CLI.

Listing 3.13: Przedstawienie serwisu.

```
type EntityResponseType = HttpResponse<IEvents>;
@Injectable({ providedIn: 'root' })
export class EventsService {
    public resourceUrl = SERVER_API_URL + 'api/events';

    constructor(protected http: HttpClient) {}

    create(events: IEvents): Observable<EntityResponseType> {
        const copy = this.convertDateFromClient(events);

        return this.http
            .post<IEvents>(this.resourceUrl, copy,
                {observe: 'response' })
            .pipe(map((res: EntityResponseType) =>
                this.convertDateFromServer(res)));
    }
}
```

Dekorator *@Injectable* ma za zadanie zapewnienie usługi, poprzez wstrzykiwanie zależności. Jest odpowiedzialny za tworzenie instancji i udostępnianie serwisu innym komponentom. W klasie przedstawiono metodę *create*, która ma za

zadanie, wykonanie żądania *HTTP*, na udostępniony przez serwer adres. Operacja ta zwraca wynik obserwowany przez asynchroniczną funkcję *pipe*, gdzie w sytuacji nadejścia odpowiedzi, zostaje ona mapowana za pomocą funkcji *map* na model przedstawiający obiekt.

3.14. Nawigacja

Organizacja nawigacji jest kluczowym elementem przy przechodzeniu między komponentami [43]. Angular narzuca tworzenie klasy, której celem jest przypisanie każdemu z dostępnych adresów komponentu do jego obsługi.

Listing 3.14: Przedstawienie nawigacji.

```
export const eventsRoute: Routes = [
  {
    path: '',
    component: EventsComponent,
    resolve: {},
    data: {},
    ...
  }
]
```

Element *path* reprezentuje *URL*, do którego można się odwołać. Gdy jest pusty, automatycznie zostaje przypisany do niego adres domyślny. Na tym przykładzie jest to */events*. Znacznik *component* odpowiada za to, który komponent ma zostać przypisany pod adres z pola *path*. Właściwość *resolve* pozwala na pobranie informacji dla aplikacji, przed przejściem do nowego widoku. Element *data* pozwala na dostarczenie danych, które są przewidziane dla tego adresu.

3.15. Generowanie komponentu

Komponent jest podstawowym elementem aplikacji obsługującym warstwę wizualną, przedstawioną użytkownikowi końcowemu [33]. Każdy komponent składa się z trzech plików:

- *html* – umożliwia tworzenie strony interpretowanej przez klienta przeglądarki internetowej,
- *scss* – plik, który zajmuje się stylizacją strony,
- *ts* – umożliwia wykreowanie logiki dla komponentu.

Aby stworzyć komponent, można skorzystać z CLI i wywołać komendę *ng generate component event-item*.

Listing 3.15: Plik events-item.component.ts.

```

@Component({
  selector: 'jhi-events-item',
  templateUrl: './events-item.component.html',
  styleUrls: ['./events-item.component.scss']
})
export class EventsItemComponent implements OnInit {
  currentAccount: any;

  constructor(protected accountService: AccountService) {}

  ngOnInit() {
    this.accountService.identity().then(account => {
      this.currentAccount = account;
    });
  }
}

```

@Component jest to dekorator, który oznacza klasę jako komponent oraz na podstawie informacji zawartych w jej obrębie, ustanawia *selector* odpowiadającemu celu jego znacznikowi, jaki może zostać dodany w pliku html, w celu jego wyświetlenia. Własność *templateUrl* określa plik *html*, który ma być przypisany do komponentu, a *styleUrls* przedstawia plik *scss* do wystylizowania tego elementu [43]. Przedstawiona klasa jest implementowana przez interfejs *OnInit* dający możliwość na zainicjowanie zmiennych, które zostały wstrzyknięte przez konstruktor.

3.16. Edycja komponentu

Celem edycji jest stworzenie widoku w postaci karty, która udostępni wszystkie niezbędne informacje związane z wydarzeniem. Pierwszym krokiem jest dodanie do pliku *html* znacznika pozwalającego na wyświetlenie komponentu.

Listing 3.16: Plik *events-item.component.ts*.

```
<div class="cards">
  <jhi-events-item *ngFor="let event of events"
    [event]="event"></jhi-events-item>
</div>
```

Za pomocą dyrektywy **ngFor* renderowany jest każdy element znajdujący się w zbiorze wszystkich dostępnych wydarzeń. Działa ona jak zwykła pętla *for*. Poprzez *[event]* do komponentu *events-item* przekazywana jest poszczególna instancja wydarzenia.

W kolejnym kroku, w komponencie potrzebne jest utworzenie zmiennej z dekoratorem *@Input*, umożliwiającym odebranie przekazanych danych z przykładu wcześniejszego.

Listing 3.17: Zmodyfikowany plik *events-item.component.ts*.

```
export class EventsItemComponent implements OnInit {
  currentAccount: any;
  @Input() event: Events;
  constructor(protected accountService: AccountService) {}

  ngOnInit() {
    this.accountService.identity().then(account => {
      this.currentAccount = account;
    });
  }
}
```

W taki sposób, można wykorzystać dane dotyczące wydarzenia, poprzez wyświetlenie ich na widoku, udostępniającym listę tego typu zjawisk.

Listing 3.18: Zmodyfikowany plik events-item.component.html.

```
<div mat-card-avatar class="example-header-image"
  [ngStyle]=
  "{ 'background-image': 'url(' + getAvatarURL() + ')' }">
</div>
<mat-card-title>{{ event.eventName }}</mat-card-title>
<mat-card-subtitle>{{ event.street }}, {{ event.city }}
</mat-card-subtitle>
```

W celu wyświetlenia wartości przypisanych do zmiennych, wymagane jest użycie składni, w której niezbędne są podwójne klamry. Przykładem użycia może być `{{event.eventName}}`.

W celu poprawnego przedstawienia zdjęcia, potrzebne jest zmodyfikowanie jego źródła.

Listing 3.19: Zmodyfikowany plik events-item.component.html.

```
<div class="offer-photo">
<img mat-card-image
  alt="Photo of event"
  [src]=" 'data:'
  + event.photo.imageContentType
  + ';base64,' + event.photo.image">
</div>
```

Poprzez zmianę własności `[src]` dla znacznika `img` można wyświetlić zdjęcie powiązane z wydarzeniem.

3.17. Prezentacja komponentu

W procesie wygenerowania i edycji komponentu przedstawionym powyżej, powstał widok, który przedstawia informacje o wydarzeniu, takie jak jego nazwa, opis, zdjęcie, login autora oraz czas i miejsce.



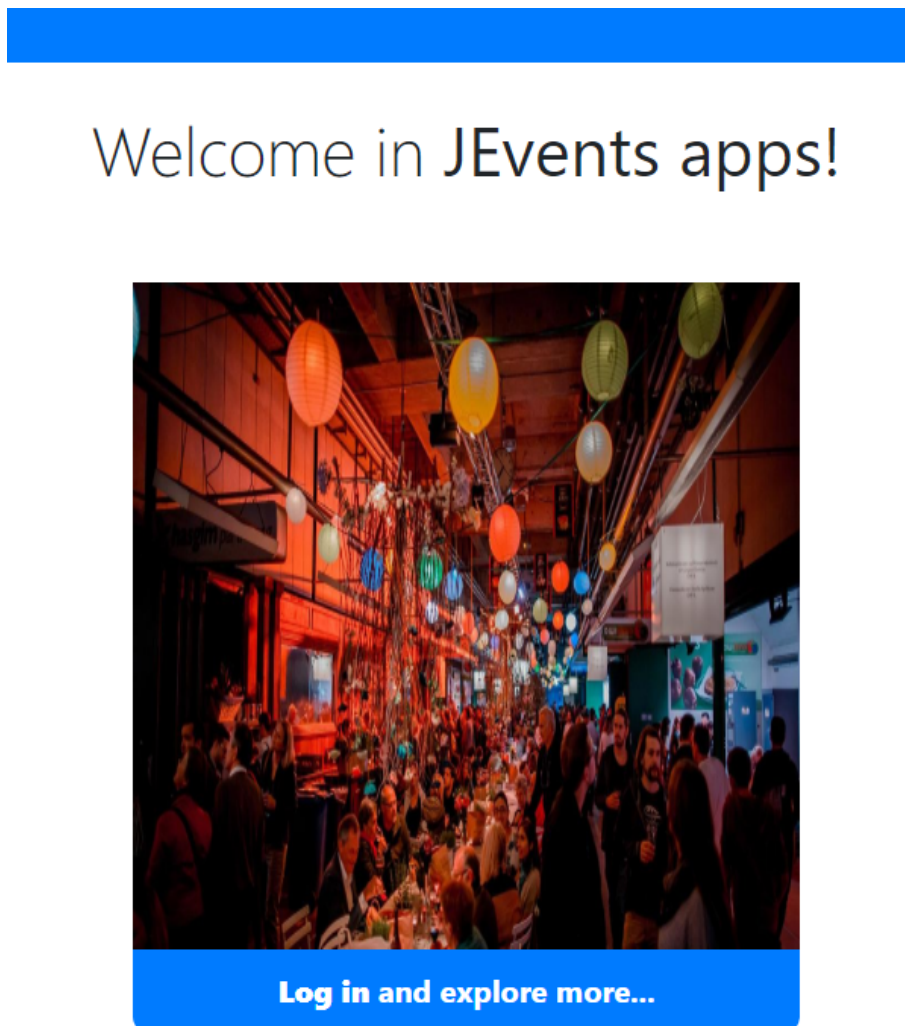
Rys. 3.3. Prezentacja komponentu events-item.

3.18. Prezentacja aplikacji

Przedstawiona aplikacja składa się z kilku części takich jak pasek nawigacyjny, widok startowy, reprezentację wszystkich wydarzeń oraz z możliwości ich dodania, edycji i usunięcia. Administrator posiada możliwość zarządzania użytkownikami, dodawania, edycji i usuwania wszystkich wydarzeń, natomiast użytkownik ma dostęp do modyfikacji jedynie własnych elementów oraz przeglądania wszystkich dostępnych.

3.18.1. Strona startowa

Stronę startową możemy podzielić na dwa elementy, przed i po zalogowaniu. Każda z nich udostępnia zbliżony widok, ale różną funkcjonalność.

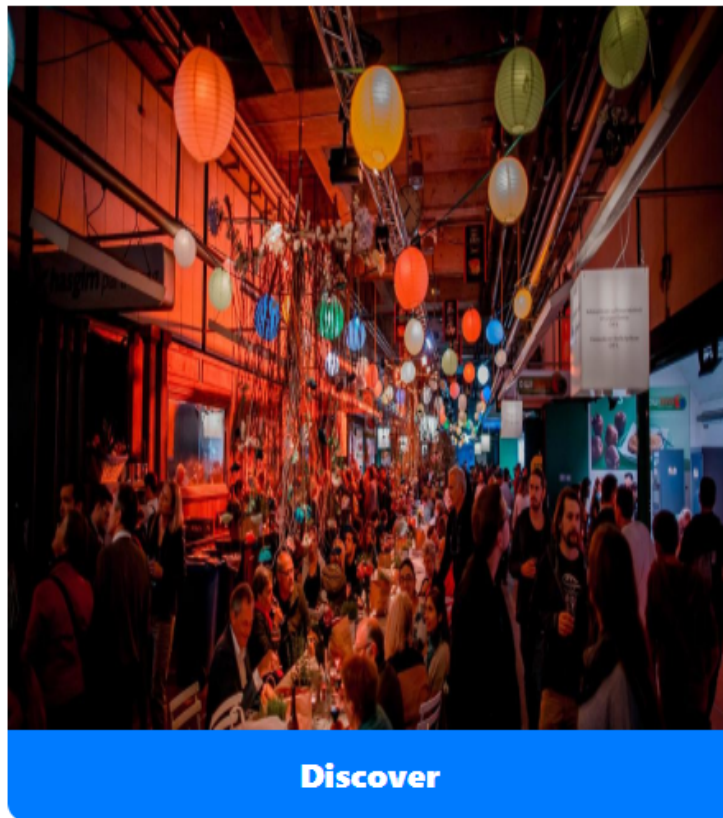


Rys. 3.4. Strona startowa przed zalogowaniem.

Za pomocą tego widoku, poprzez wciśnięcie przycisku znajdującego się pod zdjęciem, można przejść do widoku logowania.

W sytuacji, gdy użytkownik się zaloguje widok zmienia się wraz z funkcjonalnością, która przenosi na listę ofert.

Hi, konrad!



Rys. 3.5. Strona startowa po zalogowaniu.

3.18.2. Rejestracja i logowanie

Proces rejestracji jest bardzo prosty. Zainteresowany jest proszony o podanie loginu, e-maila oraz hasła. Następnie na jego pocztę zostaje wysłany link aktywacyjny dla konta.

Registration

Username

Your username is required.

Email

New password

Password strength:

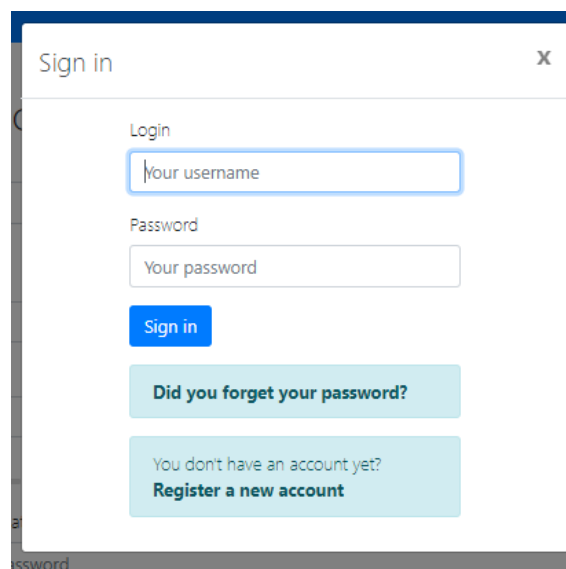
— — — — —

New password confirmation

Register

Rys. 3.6. Widok rejestracji.

Po udanej rejestracji, użytkownik ma możliwość zalogowania się do aplikacji.

A modal dialog box titled "Sign in" with a close button (X) in the top right corner. It contains two input fields: "Login" with the placeholder text "Your username" and "Password" with the placeholder text "Your password". Below the inputs is a blue "Sign in" button. Underneath the button are two light blue buttons: "Did you forget your password?" and "You don't have an account yet? Register a new account".

Rys. 3.7. Widok logowania.

Widok logowania jest minimalistyczny. Wymaga on podania loginu i hasła użytkownika. Po udanej operacji, zainteresowany otrzymuje dostęp do aplikacji.

3.18.3. Pasek nawigacji

Pasek nawigacji dostarcza dostępne opcje dla użytkowników i administratorów. Posiada on kilka sekcji, które nawigują do dostępnych komponentów.



Rys. 3.8. Prezentacja paska nawigacji.

Pasek gwarantuje dostęp do widoku głównego, listy wszystkich dostępnych wydarzeń, możliwości ich dodania oraz do zarządzania kontem. W przypadku administratora dodatkowo do zarządzania użytkownikami.

3.18.4. Widok dodawania wydarzenia

Widok ten jest podzielony na trzy sekcje: informacje, adres oraz zakończenie. Pierwsza z nich zawiera nazwę, opis, zdjęcie i czas wydarzenia. Wymienione wytyczne są niezbędne do zamieszczenia kompletnego zestawu danych dla potencjalnego zainteresowanego wydarzeniem. Użytkownik musi uzupełnić wszystkie pola wraz z poprawnym typem danych. W sytuacji, gdy w polu data pojawi się błędny typ i format, wyświetlony zostanie komunikat informujący o zaistniałej pomyłce. Kolejną sekcją jest adres, gdzie wymaganymi polami są: dokładna ulica pozwalająca określić położenie, miasto, kod pocztowy dla potwierdzenia zgodności z miastem oraz kraj. Ostatnia sekcja służy do zapisu, użytkownik ma dostępny przycisk, który dodaje wydarzenie oraz przenosi do zaktualizowanej listy ofert dostępnych dla wszystkich. Przejście pomiędzy sekcjami jest poprzedzone koniecznością wypełnienia wszystkich pól. Opisany widok jest również używany przy edycji wydarzenia. Po dodaniu wydarzenia, w dalszym ciągu istnieje możliwość edytowania danych, w celu poprawienia ewentualnych pomyłek lub aktualizacji przekazywanych informacji.

Create or edit a Events

1 Information ————— 2 Address ————— 3 Done

Event Name

Światowe Dni Gier

Description

W dniu tym przedstawione ...

Event Start

2019-07-06



Event End

2019-07-07



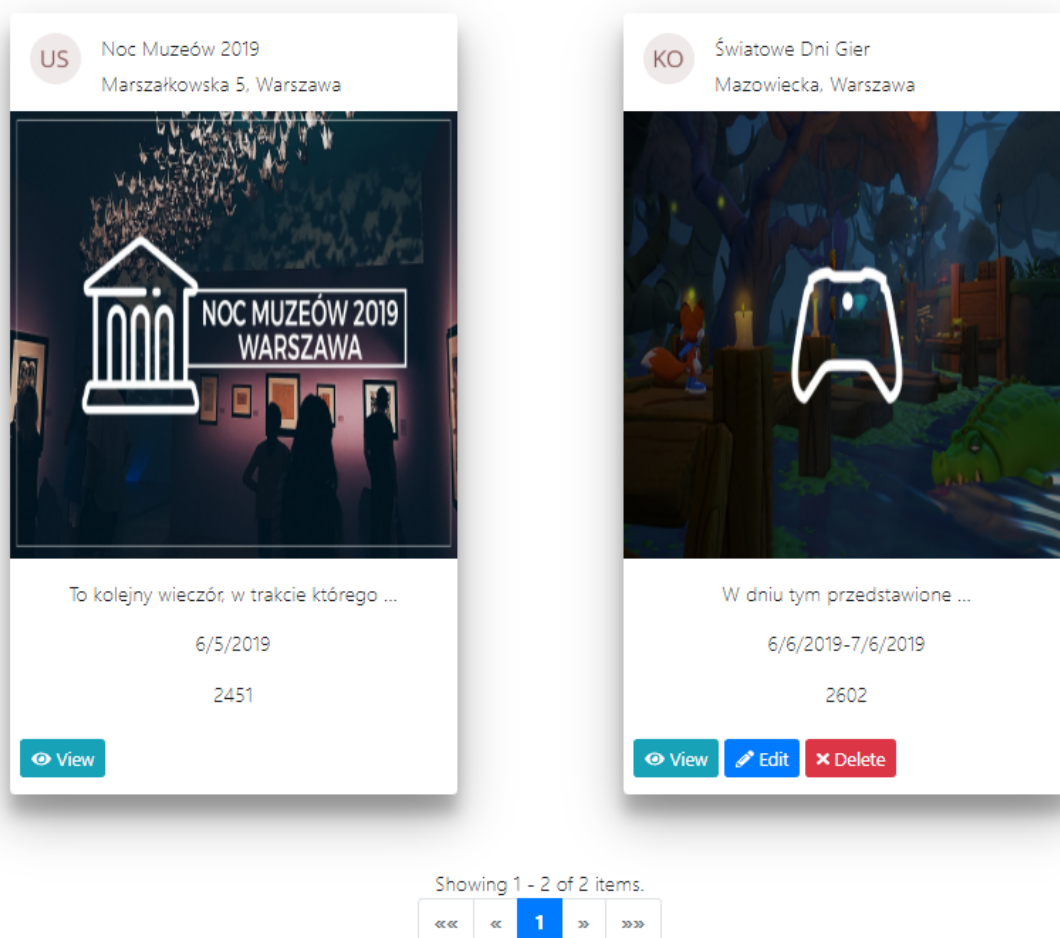
Choose File dnigier.jpg

Cancel

Rys. 3.9. Prezentacja dodawania wydarzenia.

3.18.5. Widok wszystkich wydarzeń

Widok ten przedstawia listę wszystkich dostępnych wydarzeń, dodanych przez użytkowników. Jest on zaprezentowany za pomocą widoku karty, stworzonego we wcześniejszej części pracy.

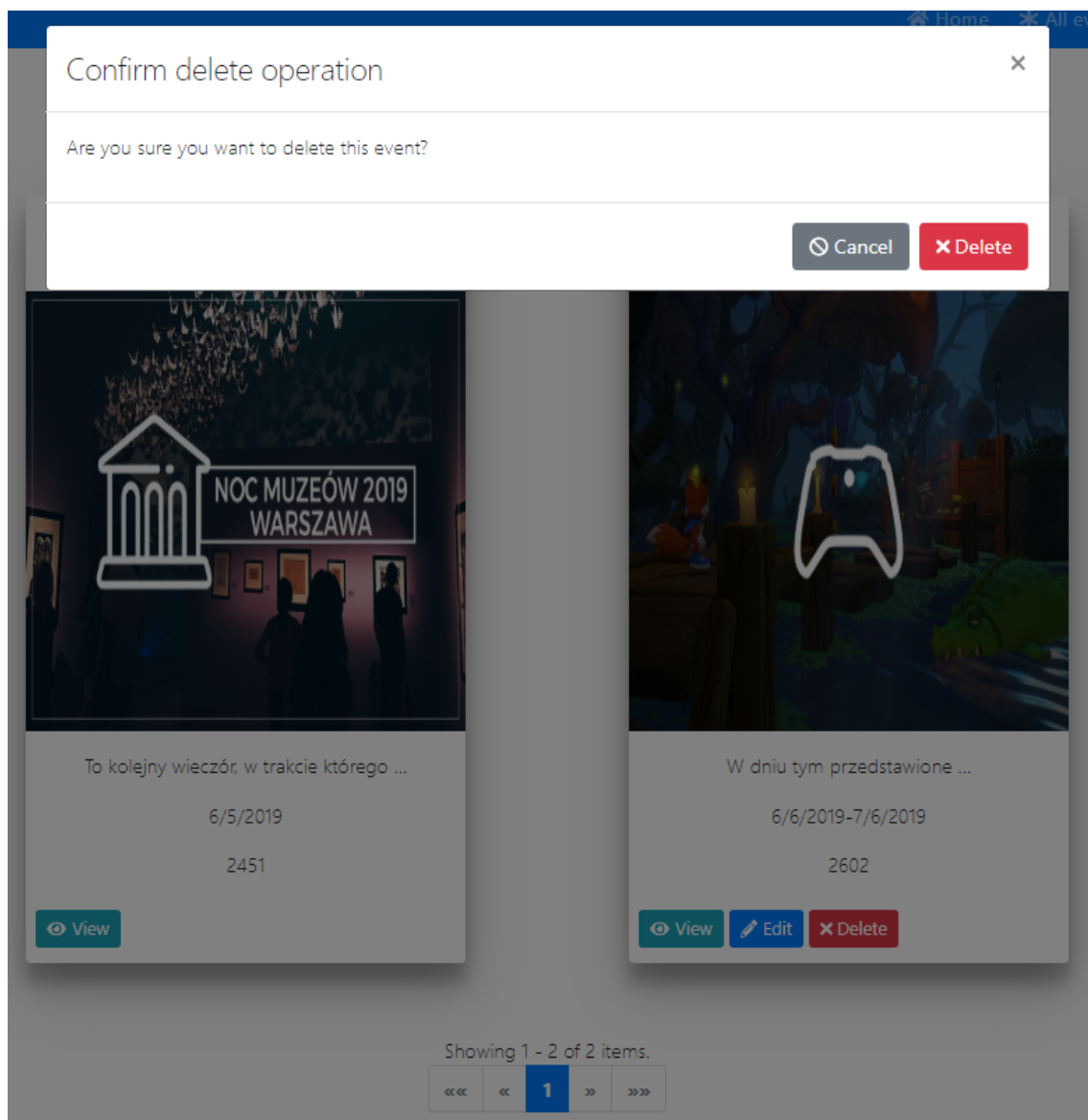


Rys. 3.10. Lista wydarzeń.

W przypadku przekroczenia na stronie maksymalnej ilości kart z wydarzeniami, kolejne zostają przerzucone na następne podstrony, do których można nawigować przy użyciu elementu znajdującego się pod listą.

3.18.6. Usuwanie wydarzenia

Autor oferty, w każdym momencie ma możliwość jej usunięcia. Po podjęciu próby skasowania wydarzenia, pojawi się komunikat, w którym należy potwierdzić lub anulować kontynuowanie procesu. Jeśli zostanie wybrana opcja kasowania element zniknie.



Rys. 3.11. Usuwanie wydarzenia.

3.18.7. Informacje szczegółowe dla wydarzenia

Sekcja ta zawiera dokładne informacje związane z wydarzeniem. W tym widoku, użytkownik może dowiedzieć się o wszystkich szczegółach związanych z daną ofertą. Między innymi ma możliwość: sprawdzenia jak nazywa się autor, zobaczenia dokładnego opisu oraz miejsce wydarzenia. Dodatkowo w tym widoku użytkownik ma prawo do edycji wydarzenia, jeśli jest ono stworzone przez niego.

ŚWIATOWE DNI GIER by Konrad

Author

[konrad]

Event Name

Światowe Dni Gier

Photo**Description**

W dniu tym przedstawione ...

Street/City/ZipCode/Country

Mazowiecka Mazowiecka 21-001 Polska

Date

Start Date: Sat Jul 06 2019 00:00:00 GMT+0200

End Date: Sun Jul 07 2019 00:00:00 GMT+0200

Rys. 3.12. Szczegóły wydarzenia.

Podsumowanie

Na rzecz pracy udało się stworzyć działającą aplikację, zgodną z założeniami. Utrudnieniem napotkanym w procesie tworzenia okazało się duże użycie pamięci RAM jak również mocy obliczeniowej procesora, przez co momentami praca była uniemożliwiona przez brak odpowiedzi od systemu operacyjnego.

Proces tworzenia aplikacji webowej jest czasochłonny. Łatwo można zauważyć, że składa się on z wielu czynników, zaczynając od warstwy serwerowej, a kończąc na wizualnej. Podczas tworzenia tego typu oprogramowania trzeba wykazać się dużą wiedzą w zakresie stosu technologicznego, w jakim aplikacja jest projektowana. Na rynku występuje podział, na programistów zajmujących się jednym lub drugim obszarem. Nieliczna grupa developerów, może pochwalić się bardzo dobrym zaznajomieniem z obiema technologiami i możliwością pracy w jednym projekcie nad oboma aspektami.

Mimo, że Angular jak i Spring Boot udostępniają szereg uproszczeń oraz pomocnych komponentów, często, aby zrozumieć działanie konkretnej składowej trzeba odwołać się do elementów z nią powiązanych i przestudiować je pod kątem działania. W Angular Framework elementem, który może przytłaczać jest liczba komponentów. Każdy z tych elementów po procesie wygenerowania posiada minimum trzy pliki z nim powiązane. Nawet przy kilku komponentach, złożoność projektu rośnie, jak również struktura traci na przejrzystości. Przy użyciu Spring Boot liczba klas powiązanych z każdą nową encją znacząco rośnie, przez co bez dobrej organizacji struktury katalogów, łatwo można się zagubić w projekcie.

Jednakowoż elementami, które przemawiają na korzyść każdego z tych rozwiązań jest fakt, że skupienie się na poszczególnych technologiach umożliwia jej opanowanie poprzez analizę szeroko rozbudowanej dokumentacji, napisanej w spo-

sób przystępny, jak również, poprzez wsparcie społeczności tworzącej to rozwiązanie.

Stworzona aplikacja jest wersją podstawową, która miała spełniać początkowe założenia. W niniejszej pracy skupiono się na przedstawieniu w najprostszy sposób procesu tworzenia przy użyciu Spring Boot i Angular. Efekt pracy można rozwinąć funkcjonalnie i wizualnie, aby poprawić jakość odbioru przez użytkownika, jednakże w opracowaniu nie było to elementem kluczowym. Pozwala to pokazać, że tworzenie nawet niewielkiej aplikacji bywa procesem bardzo rozbudowanym. Znajomość Springa ułatwiła w znacznym stopniu opracowanie i pozwoliła skupić się na zaznajomieniu z Angularem.

Bibliografia

- [1] Semmy Purewal. *Learning Web App Development*. O'Reilly Media, 2014.
- [2] Nicholas S Williams. *Professional Java for web applications*. John Wiley & Sons, 2014.
- [3] Jon Duckett. *HTML & CSS: design and build websites*. T. 15. Wiley Indianapolis, IN, 2011.
- [4] Tal Ater. *Building progressive web apps*. Ó'Reilly Media, Inc.", 2017.
- [5] Pete LePage. *Spring IoC Container*. URL: <https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/> (term. wiz. 05.06.2019).
- [6] Emmit Scott. *SPA design and architecture: understanding single page web applications*. Manning Publications Co., 2015.
- [7] Github. *The State of the Octoverse: top programming languages of 2018*. URL: <https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/> (term. wiz. 27.04.2019).
- [8] Jon Duckett. *Web Design with HTML, CSS, JavaScript and jQuery Set*. Wiley Publishing, 2014.
- [9] Mozilla. *CSS basics*. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics (term. wiz. 20.05.2019).
- [10] Eric A. Meyer i Estelle Weyl. *CSS: The Definitive Guide: Visual Presentation for the Web*. O'Reilly Media, 2017.
- [11] Axel Rauschmayer. *The Past, Present, and Future of JavaScript*. O'Reilly Media, 2012.

- [12] Elisabeth Robson i Eric T. Freeman. *Head First JavaScript Programming*. O'Reilly Media, 2012.
- [13] Nuimedia. *Javascript logo*. URL: <https://www.nuimedia.com/javascript-logo/> (term. wiz. 20.05.2019).
- [14] Steve Fenton. *Pro TypeScript: Application-Scale JavaScript Development*. Apress, 2017.
- [15] Github. *TypeScript logo*. URL: <https://github.com/remojansen/logo.ts> (term. wiz. 27.04.2019).
- [16] Miguel Grinberg. *Flask web development: developing web applications with python*. O'Reilly Media, Inc, 2018.
- [17] Herbert Schildt. *Java: The Complete Reference, Eleventh Edition*. McGraw-Hill Education Group, 2018.
- [18] Clinton Wong. *HTTP Pocket Reference: Hypertext Transfer Protocol*. O'Reilly Media, 2009.
- [19] Ludovic Demailly. *Building a RESTful web service with spring*. Packt Publishing Ltd, 2015.
- [20] Iuliana Cosmina i in. *Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools*. Apress, 2017.
- [21] Spring. *Spring Modules*. URL: <https://docs.spring.io/spring/docs/5.0.0.M1/spring-framework-reference/html/overview.html> (term. wiz. 22.05.2019).
- [22] Craig Walls. *Spring Boot in action*. Manning Publications, 2016.
- [23] Spring IO. *Spring Initializr Repository*. URL: <https://github.com/spring-io/initializr> (term. wiz. 14.06.2019).
- [24] Spring. *Spring Initializr*. URL: <https://start.spring.io/> (term. wiz. 04.06.2019).
- [25] JetBrains. *IntelliJ documentation*. URL: <https://www.jetbrains.com/idea/documentation/> (term. wiz. 20.06.2019).

- [26] Apache Maven. *Introduction to Maven*. URL: <https://maven.apache.org/guides/index.html> (term. wiz. 12.06.2019).
- [27] Apache. *Apache Maven Logo*. URL: <https://maven.apache.org/> (term. wiz. 09.06.2019).
- [28] Benjamin Muschko. *Gradle in action*. Manning, 2014.
- [29] Adam Freeman. *Pro Angular*. Springer, 2017.
- [30] Angular. *Full color Angular logo*. URL: <https://angular.io/presskit> (term. wiz. 09.06.2019).
- [31] Node.js. *Node.js documentation*. URL: <https://nodejs.org/en/docs/> (term. wiz. 19.06.2019).
- [32] Npmjs. *About npm*. URL: <https://docs.npmjs.com/about-npm/> (term. wiz. 10.06.2019).
- [33] Angular. *Angular CLI documentation*. URL: <https://angular.io/cli> (term. wiz. 20.06.2019).
- [34] JHipster. *JHipster documentation*. URL: <https://www.jhipster.tech/development/> (term. wiz. 14.06.2019).
- [35] JHipster. *JHipster Logo*. URL: <https://www.jhipster.tech/> (term. wiz. 10.06.2019).
- [36] Luca Ferrari. *PostgreSQL 11 Server Side Programming Quick Start Guide*. Packt Publishing, 2018.
- [37] Arango DB. *Performance benchmark*. URL: <https://www.arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/> (term. wiz. 18.06.2019).
- [38] Spring. *Boot features external config*. URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html> (term. wiz. 15.06.2019).
- [39] Spring. *Spring Data JPA*. URL: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/> (term. wiz. 15.06.2019).

- [40] Moises Macero, Macero i Anglin. *Learn Microservices with Spring Boot*. Springer, 2017.
- [41] Angular. *Introduction to ng modules*. URL: <https://angular.io/guide/ngmodules> (term. wiz. 17.06.2019).
- [42] Angular. *Services*. URL: <https://angular.io/tutorial/toh-pt4> (term. wiz. 20.06.2019).
- [43] Jeremy Wilken. *Angular in Action*. Manning, 2018.