AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Zarządzania

Katedra Informatyki Stosowanej

Praca licencjacka

Webowe aplikacje demonstrujące wybrane możliwości IBM Watson AI

Web applications demonstrating selected IBM Watson AI capabilities

Autor: *Anna Maria Urban*
Kierunek studiów: Informatyka i Ekonometria
Opiekun pracy: *dr Beata Basiura*

Kraków, 2020

Słownik

deep learning (pl. „głębokie uczenie się") – jest to jedna z działów sztucznej inteligencji, część obszernego pojęcia „uczenie maszynowe", która imituje działanie ludzkiego mózgu podczas przetwarzania danych i tworzenia wzorców używanych do podejmowania decyzji.[1]

framework – to pewnego rodzaju szkielet, za pomocą którego można budować aplikację, definiujący jej budowę i strukturę. Zawiera on zestaw gotowych bibliotek najczęściej do określonych zadań.[2]

język naturalny - język, którego ludzie używają do codziennego komunikowania się. Nazwa „język naturalny" jest głównie używana w dyscyplinach technicznych, aby odróżnić go na przykład od języków programowania.[3]

kod bajtowy – „lista instrukcji do wykonania przez wirtualną maszynę Javy (JVM)".[4]

maszyna wirtualna (ang. virtual machine, VM) – środowisko uruchomieniowe dla programów. Kontroluje ona odwołania uruchamianego programu prosto do sprzętu czy też systemu operacyjnego oraz je obsługuje.[5]

terabajt – bilion bajtów (10^{12})

współbieżność/Przetwarzanie współbieżne – współistnienie paru wątków/procesów, które operują na współdzielonych danych. System wykorzystując współbieżność właściwie, może skrócić czas czekania na usługę. Przetwarzanie współbieżne bardzo często wykorzystywane jest w serwisach (np. aplikacjach webowych), które to muszą obsługiwać żądania od klientów wpływające w dużej liczbie.

back-end – termin używany w kontekście tworzenia oprogramowania. Oznacza część aplikacji, która zajmuje się przetwarzaniem danych, wykonywaniem na nich pewnych operacji. Jest to wnętrze aplikacji. Back-end jest niewidoczny dla użytkownika.

front-end – termin używany w kontekście tworzenia oprogramowania. Oznacza część aplikacji, która jest widoczna dla odbiorcy: zajmuje się wyświetlaniem i pobieraniem danych od użytkownika oraz ich przekazywaniem do „wnętrza" aplikacji, czyli back-endu.

API – (ang. Application Programming Interface) Interfejs Programistyczny Aplikacji. Jest to dokładniej zbiór reguł, które określają w jaki sposób programy czy też podprogramy się ze sobą komunikują.[6]

[1] https://www.investopedia.com/terms/d/deep-learning.asp (12.01.2020)
[2] https://pl.wikipedia.org/wiki/Framework (12.01.2020)
[3] https://pl.wikipedia.org/wiki/J%C4%99zyk_naturalny (10.12.2019)
[4] https://pl.wikipedia.org/wiki/Kod_bajtowy_Javy (14.01.2020)
[5] https://pl.wikipedia.org/wiki/Maszyna_wirtualna (14.01.2020)
[6] https://pl.wikipedia.org/wiki/Interfejs_programowania_aplikacji (30.06.2020)

Spis treści

Wstęp

Wprowadzenie

Jedną z ostatnio powszechnie rozwijanych i niewątpliwie przełomowych dziedzin informatyki jest sztuczna inteligencja (ang. *artificial intelligence*, AI). Pojęcie to obejmuje logikę rozmytą, obliczenia ewolucyjne, sieci neuronowe, sztuczne życie i robotykę. Przede wszystkim, sztuczną inteligencję można wykorzystać do rozwiązywania problemów, które ciężko rozwiązać za pomocą dostępnych algorytmów. Za jej pomocą tworzone są m.in. programy komputerowe, które symulują inteligentne zachowania. Oprócz informatyki, sztuczna inteligencja również ma do czynienia z takimi dziedzinami nauki, jak neurologia, kognitywistyka, psychologia, a ostatnio nawet filozofia. [7] Gdy nie ma wszystkich danych, a istnieje potrzeba podjęcia pewnej decyzji, jest to rodzaj problemu z którym AI łatwo sobie poradzi. Inne rozwiązywalne przez nią problemy, to np. analiza języków naturalnych, maszynowe tłumaczenie tekstów, rozpoznawanie obrazów czy też mowy. Istnieje dział AI, który zajmuje się algorytmami potrafiącymi na własnych błędach uczyć się podejmować decyzje, a nawet pozyskiwać wiedzę.

International Business Machines Corporation (IBM) to przedsiębiorstwo amerykańskie posiadające swoje siedziby w Polsce i wielu innych krajach. Jest to jedna z najstarszych firm informatycznych na świecie, powstała w Stanach Zjednoczonych 16 czerwca 1911 roku. W Polsce, pierwszy oddział powstał w roku 1935 w Warszawie. Korporacja ta obejmuje wszelkie usługi informatyczne, w tym oprogramowanie oraz sprzęt. Odnosząc się do historii, to właśnie firmie IBM zawdzięcza się dysk twardy Winchester [8], dyskietkę, kursor i wiele innych. W 1981 roku miała miejsce premiera komputera osobistego, do którego to IBM wraz z Billem Gatesem – właścicielem niewielkiej wówczas firmy Microsoft, stworzyli architekturę uznawaną do dziś za wszechobecny standard. Niewątpliwie jest to największe przedsiębiorstwo dotyczące segmentu informatyki na całym świecie. [9]

Jednym z produktów firmy jest superkomputer nazwany Watson. Został on stworzony aby odpowiadać na pytania, które są zadawane w języku naturalnym. Komputer ten jest ściśle związany ze sztuczną inteligencją i oferuje szereg funkcji i rozwiązań, które mogą być wykorzystywane w wielu dziedzinach, przez nieograniczoną liczbę firm. Funkcje, jak i superkomputer zostaną szerzej opisane w rozdziale 1.

[7] https://pl.wikipedia.org/wiki/Sztuczna_inteligencja (10.12.2019)
[8] https://encyklopedia.interia.pl/komputery/news-dysk-winchester,nId,2017796 (10.12.2019)
[9] https://pl.wikipedia.org/wiki/IBM (10.12.2019)

Aktualność tematu pracy

To, jaki potencjał w kryje w sobie AI, jest największą możliwością gospodarczą naszej epoki. Do 2030 roku, według szacunków, przyczyni się ona do wzrostu PKB aż w wysokości 16 trylionów dolarów. W przedsiębiorstwach, w 2018 roku używano jej potencjału jedynie w 4%, natomiast w 2019 roku było to już 14%. [10] Jednak nadal jest to mało. Dlaczego więc, pomimo zadziwiających rozwiązań jakie oferuje, wykorzystywana jest w tak małym stopniu? Powodów jest wiele. Ludzie często nie posiadają wystarczających umiejętności, aby jej używać. Inni z kolei nie mają do tego narzędzi. Może się zdarzyć, że niektórzy są z góry sceptycznie nastawieni do tego typu rozwiązań. Natomiast wiele ludzi nie jest nawet świadomych, że istnieją pewne dostępne funkcje wykorzystujące AI, które mogłyby się im lub ich firmie przydać.

Dodatkowo, rynek informatyczny cały czas się rozwija, a ofert pracy na stanowisku programisty jest bardzo dużo. Istnieje zatem potrzeba uczenia młodszych pokoleń różnych technologii oraz języków programowania. Według informatycznego portalu „no fluff {jobs}", najczęściej występującymi technologiami w ogłoszeniach o pracę w dziedzinie IT w 2018 roku są: JavaScript (26,94%) oraz Java (24,31%).[11]

Cel pracy

W odpowiedzi do przedstawionych faktów, celem pracy jest stworzenie aplikacji, które będą implementowały wybrane możliwości wspomnianego superkomputera "IBM Watson" w szczególności dotyczących sztucznej inteligencji, jednocześnie pokazując ich przykładowe zastosowanie. Pojedyncza aplikacja będzie prezentować głównie jedną funkcję superkomputera.

Aplikacje posłużą zatem do zaprezentowania możliwości superkomputera w dziedzinie sztucznej inteligencji. Będą także pomocnym narzędziem w nauce języka Java oraz popularnego frameworku Spring, w którym zostaną napisane (część backend'owa), jak również języka TypeScript i frameworku Angular (część frontend'owa). Dzięki temu rynek IT będzie miał więcej wykwalifikowanych kandydatów na wciąż nie wypełnione miejsca pracy, a osoby mające styczność z napisanymi w ramach tej pracy aplikacjami będą wiedziały jak korzystać z nowych technologii, oraz rozwiązań sztucznej inteligencji.

[10] https://www.ibm.com/blogs/think/2019/10/watson-anywhere-the-future/ (10.12.2019)
[11] Raport „Rynek pracy IT w 2018 roku", no fluff {jobs}: M. Gawłowska-Bujok, T. Bujok, 2018

Struktura pracy

W rozdziale pierwszym znajduje się opis jak powstał superkomputer IBM Watson oraz analiza dwóch jego funkcji, które wybrano do przybliżenia przez aplikacje.

Rozdział drugi przedstawia najpopularniejsze dziś technologie, które wykorzystane zostały do napisania prezentowanych aplikacji. Ich wady, zalety, ścieżkę rozwoju która prowadzi do frameworków oraz zastosowanie.

Rozdział trzeci zawiera opis napisanych aplikacji wraz z wizualizacją efektów pracy. Dodatkowo zaprezentowano i opisano najważniejsze fragmenty strony serwera która zawiera logikę biznesową, oraz strony klienta, czyli: backend oraz frontend.

1 Opis i analiza wybranych funkcji IBM Watson AI

1.1 Jak powstał superkomputer Watson

Po zwycięstwie komputera szachowego IBM – Deep Blue z panującym wtedy mistrzem świata w turniejach szachowych z kontrolowanym czasem - Garrym Kasparowem w 1997 roku, IBM rozglądał się w poszukiwaniu kolejnego wyzwania. Początkiem 21 wieku popularna była gra w *„Jeopardy!”*. (j. polski: Niebezpieczeństwo!) Był to *„amerykański program telewizyjny stworzony przez Merv Griffin. Spektakl obejmuje konkurs quizowy, w którym uczestnicy otrzymują wskazówki dotyczące wiedzy ogólnej w formie odpowiedzi i muszą sformułować swoje odpowiedzi w formie pytań.”*[12] Charles Lickel – kierownik działu badań IBM był zaciekawiony tym teleturniejem i postrzegał go jako kolejną możliwość rozwoju dla firmy. Przedstawił pomysł wykonawcy IBM Research – Paulowi Hornowi, który to w 2005 roku również zauważył w teleturnieju potencjalne wyzwanie.

Problem polegał na skonstruowaniu maszyny, wykonującej znacznie bardziej skomplikowane rzeczy niż gra w szachy, która nie wymaga używania słów. Ostatni system występujący w teleturnieju: „Piquant”, zazwyczaj odpowiadał poprawnie jedynie na ok. 35% wskazówek, przy czym potrzebował do tego najczęściej kilku minut. W celu konkurowania z poprzednikiem, następny system musiałby odpowiadać na pytania w przeciągu paru sekund, przy czym problemy dotyczące teleturnieju zostały uznane za nierozwiązywalne.

Na początku nie było łatwo ze znalezieniem personelu badawczego, który chciałby się podjąć stworzenia tak skomplikowanej maszyny. Nie była to bowiem zwyczajna gra w szachy z kontrolowanym czasem, w której nie trzeba używać słów. W końcu David Ferrucci (kierownik działu analizy semantycznej i integracji IBM) podjął się wyzwania. Komputer nazwano „Watson” od Thomasa J. Watsona, który był założycielem oraz pierwszym prezesem IBM. Początkowe testy nie były zadowalające, aczkolwiek w 2008 roku udało mu się już konkurować z mistrzami teleturnieju. Swoich ludzkich przeciwników Watson regularnie pokonywał już do lutego 2010 roku.

[12] https://en.wikipedia.org/wiki/Jeopardy! (13.12.2019)

Wielkim wyczynem była wygrana komputera w 2011 roku z do tej pory najbardziej utytułowanymi, ludzkimi przeciwnikami, przy czym superkomputer nie był podłączony do Internetu. Miał za to dostęp do 200 milionów stron, które zużywały cztery terabajty miejsca na dysku, włącznie z całym tekstem z Wikipedii z 2011 roku. Zwycięstwo było krokiem milowym dla sztucznej inteligencji.[13]

Rysunek 1
IBM Watson kontra Jeopardy!, rok 2011 [14]

Mimo, że głównym autorem systemu jest IBM, nad jego rozwojem pracowało wielu doktorantów oraz wykładowców z *"Rensselaer Polytechnic Institute , Carnegie Mellon University , University of Massachusetts Amherst , na University of Southern California „s Information Sciences Institute , The University of Texas w Austin , w stanie Massachusetts Institute of Technology i University of Trento , a także studenci z New York Medical College."*[15]

Wraz z upływem czasu, Watson był rozwijany. Dziś posiada znacznie więcej niesamowitych funkcji i usprawnień, które zostaną wykorzystane w omówionych w rozdziale trzecim aplikacjach.

[13] https://www.computerweekly.com/photostory/450423802/AI-A-brief-history-of-man-versus-machine-intelliegnce/3/IBM-Watson-versus-Jeopardy (27.12.2019)
[14] https://www.computerweekly.com/photostory/450423802/AI-A-brief-history-of-man-versus-machine-intelliegnce/3/IBM-Watson-versus-Jeopardy (10.12.2019)
[15] https://pl.qwe.wiki/wiki/Watson_(computer) (27.12.2019)

Tutaj przybliżone zostaną tylko te wykorzystane w trzecim rozdziale funkcje.

1.2 Funkcja 1 – Personality Insights

Jedna z najbardziej intrygujących funkcji superkomputera IBM Watson to Personality Insights, polegająca na tworzeniu pełnego profilu osobowości na podstawie czyjejś wypowiedzi w formie tekstowej. Taka wypowiedź może być pobrana na przykład z mediów społecznościowych (np. Twitter, Facebook), danych korporacyjnych, czy też innej komunikacji cyfrowej.[16] Przykład: na podstawie listów motywacyjnych osób, które chcą zostać zatrudnione w danej firmie, można wyodrębnić określone cechy które są pożądane na danym stanowisku. Oczywiście, im więcej słów do analizy tym lepiej, o czym będzie wspomniane w dalszej części pracy. System oparty jest na algorytmach analizy danych oraz psychologii języka.

Język ludzki odzwierciedla bowiem styl myślenia, osobowość, powiązania społeczne, a także stany emocjonalne. Według wielu badaczy (których spis jak i publikacje możemy znaleźć tutaj: https://cloud.ibm.com/docs/services/personality-insights?topic=personality-insights-references#fast2008), istnieje możliwość przewidzenia aspektów osobowości na podstawie różnic w użyciu słów w przykładowych pismach, takich jak: eseje, tweety czy blogi. Nawet częstotliwość, z jaką są używane dane kategorie słów, może dostarczać wskazówek co do osobowości ich autora. Nawiązując do przeprowadzonych przez IBM badań, cechy osobowości wynikające z przeanalizowanych danych z mediów społecznościowych są w pewien sposób skorelowane z zachowaniem tych osób w sieci. Przykład: osoby, które poprzez publikowane w Internecie posty w dużej mierze zostały sklasyfikowane jako szukające wrażeń/dociekliwe, okazały się częściej reagować na internetowe bodźce. Natomiast ci sklasyfikowani bardziej jako ostrożni, robią to znacznie rzadziej. Kolejny przykład: sporo ostatnich badań z danymi sklepów detalicznych pokazało, że ludzie, którzy w analizie swoich tekstów w mediach społecznościowych zdobywali wysokie wyniki w cechach takich jak uporządkowanie, ostrożność czy też samodyscyplina, są aż o 40% bardziej skłonni, aby odpowiadać na oferty kuponów, niż reszta populacji. Osoby interesujące się transcendencją wykazywały większą chęć do czytania artykułów o środowisku, a ludzie z chęcią samodoskonalenia się znacznie częściej zaciekawieni byli artykułami na temat pracy.[17]

Jeśli chodzi o użycie tej wiedzy w praktyce: na pewno znajdą się firmy posiadające np. sklepy internetowe, które chciałyby posiadać takie informacje o swoich klientach. Z drugiej

[16] https://cloud.ibm.com/docs/services/personality-insights?topic=personality-insihts-gettingStarted (12.01.2020)
[17] https://cloud.ibm.com/docs/services/personality-insights?topic=personality-insights-science (13.01.2020)

strony można by pomyśleć, czy nie jest to już naruszanie ludzkiej prywatności. Trzeba mieć więc świadomość tego, że wszystko co robimy w Internecie zostawia za sobą ślad.

IBM w ramach usługi Personality Insights opracowało algorytmy służące do wnioskowania na temat cech osobowości według trzech modeli. Pierwszym z nich jest Wielka Piątka.

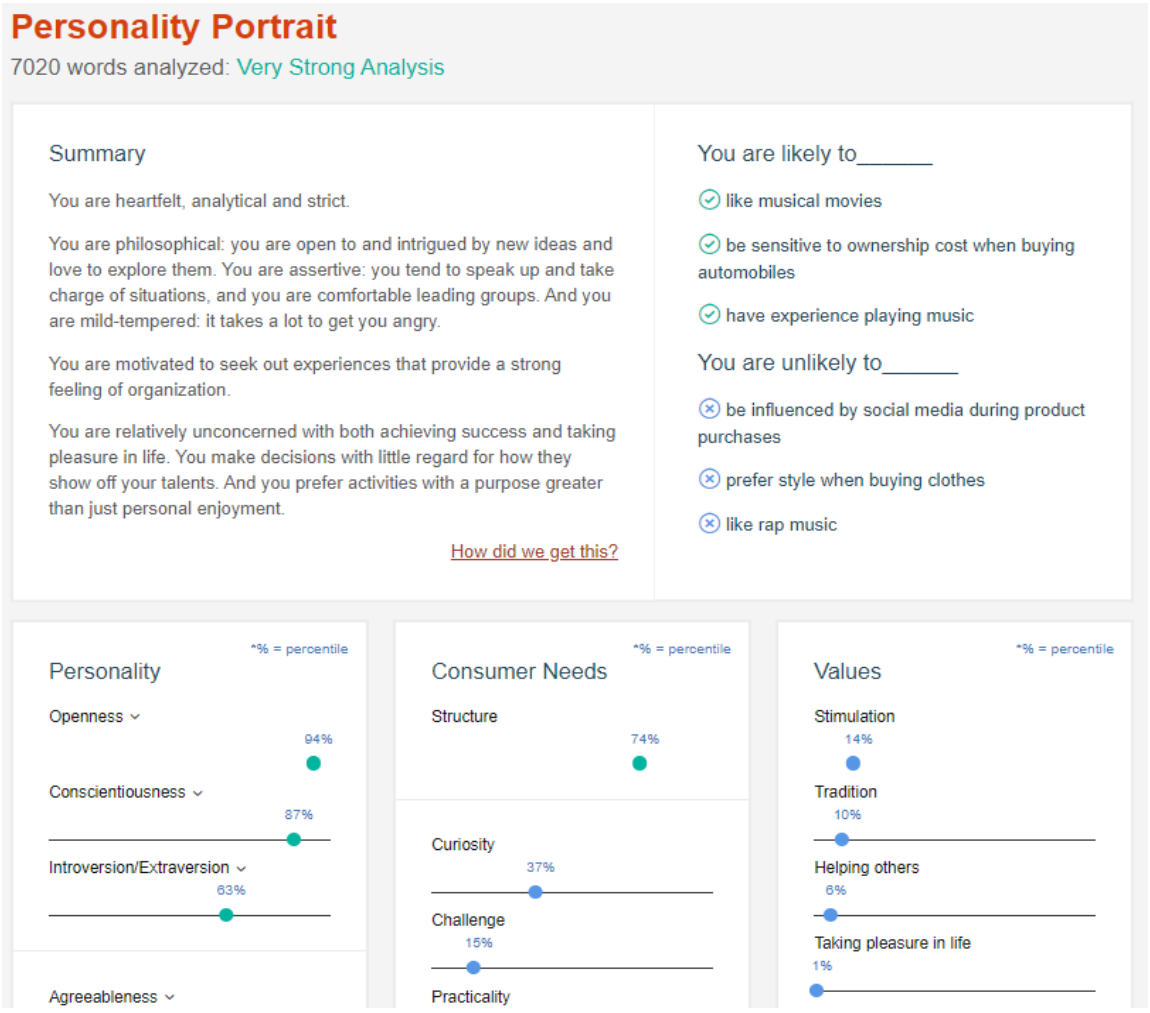
Wielka Piątka, nazywana inaczej Pięcioczynnikowym Modelem Osobowości (ang. *Big Five Model, Five Factor Model*), jest określeniem pięciu podstawowych wymiarów osobowości. Są to kolejno: ekstrawersja, ugodowość, sumienność, stabilność emocjonalna (w niektórych modelach jej odwrotność, czyli neurotyczność), oraz otwartość/intelekt (czasami cecha ta określana jest jako otwartość na doświadczenia)[18]. Aby na podstawie czyjejś wypowiedzi opisywać nie tylko język, ale też określać osobowość, użyto analizy czynnikowej. Dzięki niej zredukowano wiele określeń, które opisywały cechy do kilku podstawowych wymiarów. Na samym początku badania leksykalnie używano tylko terminologii w języku angielskim. W ich wyniku udało się cechy osobowości doprowadzić do opisanych pięciu wymiarów, które oznacza się rzymskimi liczbami: I, II, III, IV, V. [19] Dodatkowo, każdy z wyszczególnionych wymiarów ma sześć aspektów, które wspomagają jego charakterystykę. Jest to do tej pory najlepiej zbadany oraz najbardziej popularny model opisu osobowości, związany z nazwiskami Costy oraz McCrae.

Na podstawie badań w dziedzinach marketingu, psychologii oraz psycholingwistyki, IBM określił kolejny ważny aspekt, który został uwzględniony w usłudze, a mianowicie: „potrzeby". Wyszczególniono 12 kategorii uniwersalnych potrzeb ludzkich i określono je jako pragnienia, które społeczeństwo ma nadzieję zaspokoić, gdy rozważa nabycie pewnego produktu lub usługi.[20]

Trzecim i ostatnim aspektem, którym posiłkuje się model Personality Insights są „wartości". Można powiedzieć, że w pewnym sensie określają one człowieka: jego wybory, przekonania, motywację. Usługa wykorzystuje pięć podstawowych wartości, które zdefiniowane zostały przez Shaloma H. Schwartza i zatwierdziło je dla 20 krajów.[21]

[18] W. Strus, J. Cieciuch, *Poza Wielką Piątkę – Przegląd Nowych Modeli Struktury Osobowości*, Warszawa 2014, s.18
[19] J. Cieciuch, M. Łaguna, *Wielka Piątka i nie tylko. Cechy osobowości i ich pomiar*, Warszawa 2014, str. 240-241
[20], [18] https://cloud.ibm.com/docs/services/personality-insights?topic=personality-insights-science (13.01.2020)
[21] S. H. Schwartz, Universals in the content and structure of values: theoretical advances and empirical tests in 20 countries, 1992, str. 25-26.

Dostępne języki analizowanego tekstu to: arabski, angielski, japoński, koreański oraz hiszpański. Gotową analizę można natomiast przetłumaczyć na wyżej wymienione języki, oraz na: brazylijsko-portugalski, francuski, niemiecki, włoski, uproszczony chiński oraz tradycyjny chiński.[22] Usługę można wykorzystać na przykład do celów marketingowych. Logując się na stronę sklepu poprzez serwis społecznościowy, może on zebrać udostępnione przez użytkownika posty i na podstawie jego potrzeb, wartości oraz innych wspomnianych cech oferować spersonalizowane produkty. Obiekt analizowanego tekstu nie może przewyższyć 20 MB (w j. angielskim byłoby to aż około 50000 słów). Określono, że do 6000 słów jakość obserwacji ulega poprawie. Powyżej tej wartości, ilość użytych wyrazów nie ma już znaczenia. Poniżej - przeciwnie. Do statystycznie istotnych wniosków potrzebne jest 1200 słów, jak wskazują wytyczne według dokumentacji serwisu.[23]



Rysunek 2: Analiza wypowiedzi Baracka Obamy z debaty w 2012 roku, dzięki usłudze Personality Insights IBM Watson AI, opracowana przez IBM.[24]

[23] https://cloud.ibm.com/docs/personality-insights?topic=personality-insights-input (dostęp: 13.01.2020)
[24] https://personality-insights-demo.ng.bluemix.net/?_ga=2.265115469.337882622.1578833917-2031134999.1570907247 (dostęp 12.01.2020)

Na rysunku 2 znajduje się zrzut ekranu przykładu zaimplementowanego przez IBM, pokazującego jak może działać usługa. Jest to przerobiony dla użytkownika widok z uzyskanego w wyniku analizy formatu JSON. Jako tekstu użyto wypowiedzi Baracka Obamy z Debaty w 2012 roku.

1.3 Funkcja 2 – Tone Analyzer

Funkcja, która może wydawać się podobna, lecz tak naprawdę działa zupełnie inaczej, to Tone Analyzer. Personality Insights polega na utworzeniu całego profilu osobowości na podstawie tekstu, natomiast Tone Analyzer analizuje emocje skryte za danym zdaniem jedno po drugim, lub też, do wyboru: za całym kawałkiem tekstu. Wszystko zależy od wybranej opcji. Usługa ta przydaje się zawsze, gdy firma chciałaby poznać np. ton wypowiedzi swoich klientów, w celu udzielenia odpowiedniej odpowiedzi. Pozwala to uniknąć nieporozumień i może skutkować lepszą komunikacją klient – sklep, a co za tym idzie, lepszą opinią na temat samego sklepu, ponieważ potrzeby klientów są w ten sposób lepiej rozumiane. Podstawowym celem sprzedawcy jest między innymi, aby klient czuł się dobrze po zakupie oferowanego produktu, a w razie wątpliwości, aby odszedł z myślą, że został dobrze potraktowany. W zasadzie w każdym/każdej sytuacji, w której dana osoba byłaby zainteresowana tonem wypowiedzi swojego rozmówcy/osobie trzeciej, usługa ta miałaby zastosowanie. W zależności od rodzaju wykorzystania, możemy wybrać jedną z dwóch opcji:

1. Analiza ogólnego tonu

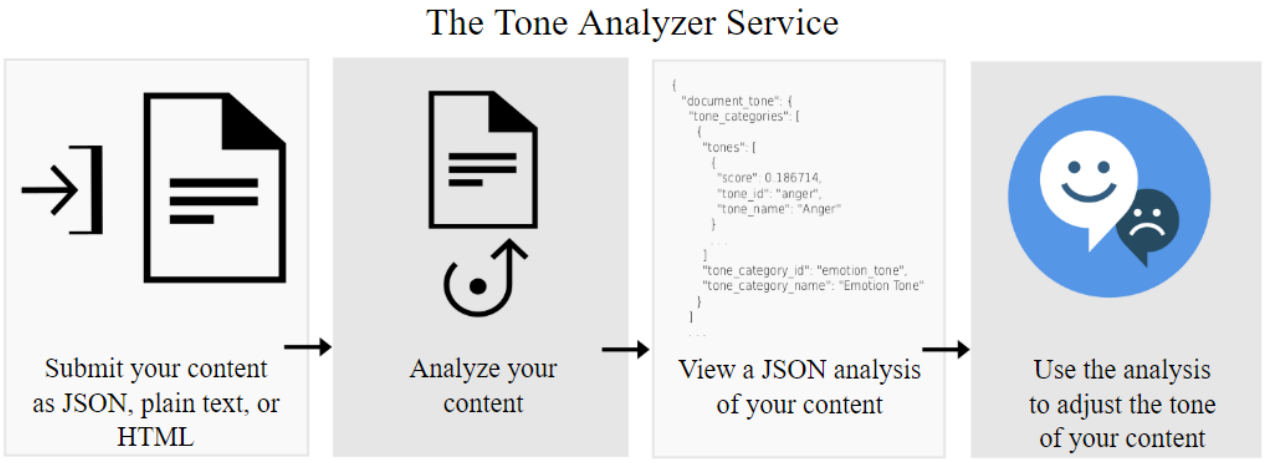
Gama emocji, które rozpoznaje Watson jako analiza ogólnego tonu wypowiedzi to: złość, smutek, strach, radość (emocjonalne), oraz z innej kategorii: ton analityczny, pewny siebie, niepewny. Razem jest ich siedem. Górny limit rozmiaru całego przesłanego pliku z tekstem do analizy to 128 KB, i jednocześnie maksymalnie 1000 zdań. (w różnych formatach: zwykłym tekście, HTML, czy też JSON). Odnośnie wspomnianych wcześniej opcji: omawiana usługa analizuje pierwsze 100 zdań pliku każde z osobna, natomiast do analizy całego dokumentu wykorzystywane już są wszystkie zdania, czyli w największej ich ilości 1000. Co jest jeszcze istotne, wspierane języki to angielski i francuski. Domyślną opcją jest angielski, dlatego jeżeli użytkownik chce inny język, trzeba to zaznaczyć w nagłówku zapytania. Obowiązkowe parametry do podania to wersja usługi, (obecna to 2017-09-21) oraz oczywiście,

tekst do analizy. Opcjonalnymi parametrami są: typ przesyłanego tekstu (application/json, text/html itp.), język, w którym na zostać zwrócona odpowiedź (znacznie więcej do wyboru, niż angielski i francuski), oraz czy potrzebna jest analiza każdego zdania z osobna – bowiem wykonana analiza całego dokumentu jest zwracana domyślnie.[25] Odpowiedź jest w formacie JSON. Oprócz wyniku pod postacią nazwy emocji, zwracana jest również kategoria, do której należy, oraz informacja w jakim natężeniu ona występuje. Np. „Smutek" może zostać wykryty zarówno na 90%, jak i na 50%.

2. Analiza tonu zaangażowania klienta

Ten rodzaj analizy sprawdzi się zwłaszcza w podanym wcześniej przykładzie klient-sklep. Gama analizowanych emocji różni się bowiem od tych wymienionych w poprzednim podpunkcie. Tutaj celuje się w aspekt marketingowy wypowiedzi, nastawiając się na zadowolenie klienta (lub jego brak). Ustalono więc, że istotnych będzie siedem następujących emocji/cech: sympatyczny, podekscytowany, zadowolony, smutny, sfrustrowany, uprzejmy oraz nieuprzejmy. Jeśli chodzi o wybór analizowanych oraz zwracanych języków, sprawa wygląda tak samo jak przy analizie ogólnego tonu. Obowiązkowymi parametrami zapytania także są wersja usługi (obecna: 2017-09-21) i wypowiedzi, które są obiektem zainteresowania.

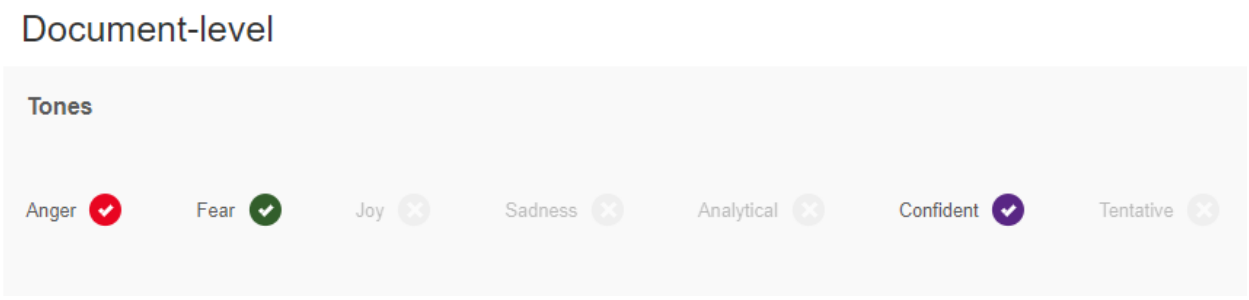
Bardziej szczegółowe informacje można znaleźć na stronie oficjalnej dokumentacji serwisu: https://cloud.ibm.com/apidocs/tone-analyzer. Na rysunku 2 zobrazowano uproszczony przepływ informacji podczas używania IBM Watson Tone Analyzer.



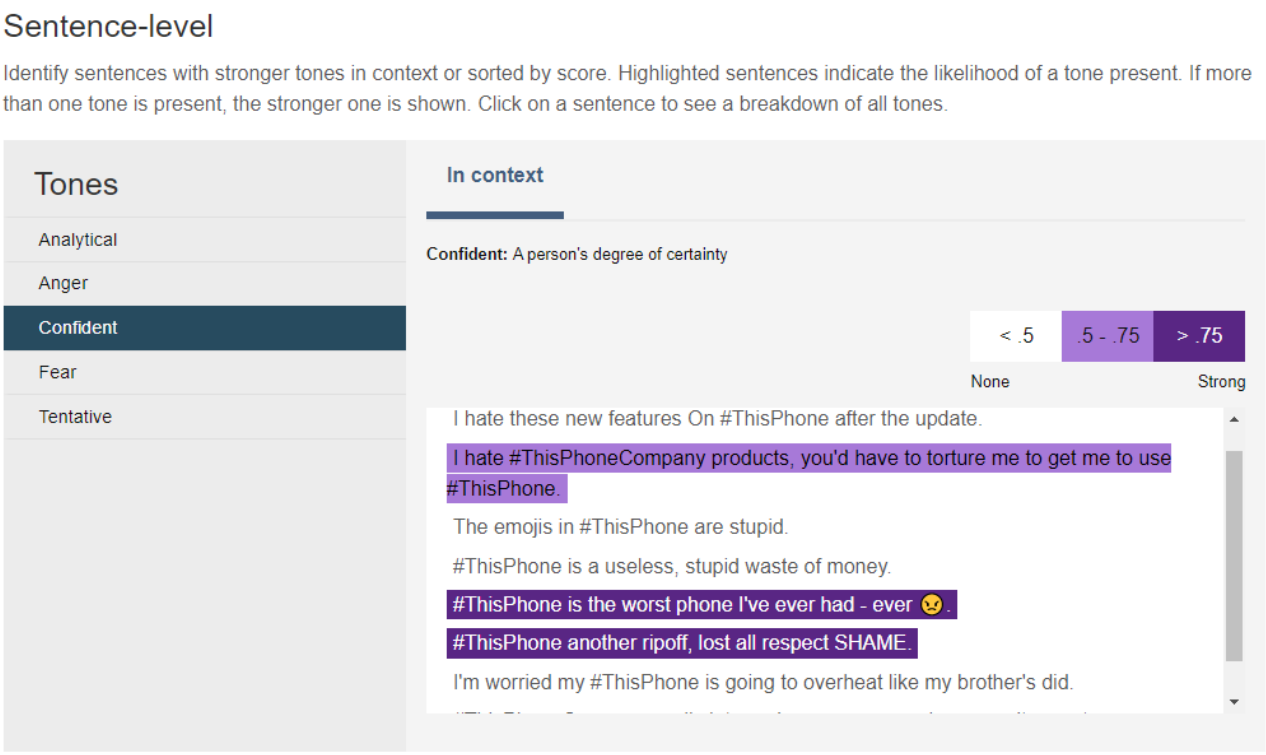
Rysunek 2
Przedstawienie ogólnego przebiegu przepływu informacji podczas używania usługi Tone Analyzer[26]

[25] https://cloud.ibm.com/apidocs/tone-analyzer#analyze-general-tone (14.03.2020)
[26] https://cloud.ibm.com/docs/tone-analyzer?topic=tone-analyzer-about (14.03.2020)

Otrzymany jako rezultat analizy plik o formacie JSON może być wykorzystany na wiele sposobów, wszystko zależy od chęci, wizji oraz potrzeby. W celu zobrazowania działania Tone Analyzer'a, efekt został bardzo przejrzyście przedstawiony na rysunkach 3 i 4. Odnoszą się one do tego samego kawałku tekstu. Dobrze widać 2 opcje analizy: tekstu jako całość, oraz tekstu zdanie po zdaniu.



Rysunek 3
Efekt analizy ogólnego tonu na poziomie całego dokumentu, nie zdań osobno.[27]



Rysunek 4
Efekt analizy ogólnego tonu pojedynczych zdań, nie dokumentu jako całość. Wyróżniona zakładka z emocją "pewność siebie".[28]

[27], [28] https://tone-analyzer-demo.ng.bluemix.net/?cm_mc_uid=10155467735015553357446&cm_mc_sid_50200000=26734711588354771949&cm_mc_sid_52640000=95640691588354844949&_ga=2.107633730.1971899266.1588354773-2031134999.1570907247 (14.03.2020)

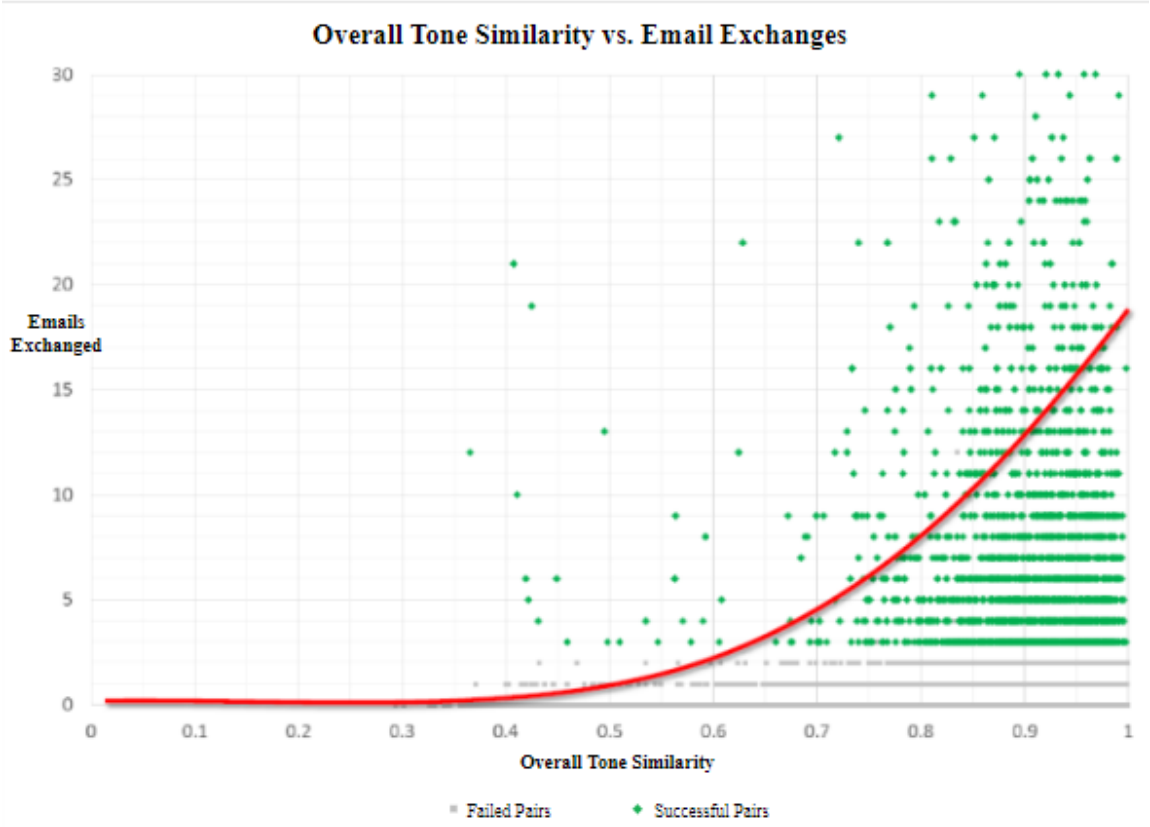
Na rysunku 4 postanowiono przedstawić procent pewności analizy danego zdania za pomocą intensywności koloru. Im ciemniejsze tło, tym pewniej zdanie zostało zaklasyfikowane pod dany ton. Jeżeli system był pewny na mniej niż 50%, zdania nie zostały wyszczególnione. Jasny fiolet przeznaczono dla wartości pomiędzy 50% a 75%, natomiast najciemniejszy fiolet na powyżej 75%, że zdanie emanuje pewnością siebie.

Jeśli chodzi o samą podstawę serwisu, czyli naukę, która pozwoliła określić w jaki sposób badać zdania – to psycholingwistyka. Zajmuje się ona odkrywaniem w jaki sposób ludzie przyswajają sobie język (w szczególności, jakie procesy psychiczne za tym stoją) oraz w jaki sposób go używają. W przypadku usługi Tone Analyzer, szczególnie skupiono się nad drugim zagadnieniem.[29] Okazuje się, że nawet częstotliwość wypowiadanych słów danego rodzaju może być wskazówką odnośnie osobowości, sposobu myślenia, stanu emocjonalnego. Człowiek sam często nawet nieświadomie określa ton emocjonalny osoby czytając napisaną przez nią wiadomość/email/komunikat, oraz odpowiada zgodnie z odczuwanymi emocjami. Z tekstu pisanego można przecież wyczuć radość, smutek, otwartość, pewność siebie, skrytość i wiele innych emocji. W ten sposób można zbudować swoją opinię na temat osoby, z którą ma się kontakt online, nawet nie znając jej osobiście. W komunikacji biznesowej wiele emocji trzeba odłożyć na dalszy plan i stosować się do pewnych zasad. Dlatego czytając na przykład emaile biznesowe ludzie częściej mogą odczuwać negatywne emocje. W mediach społecznościowych ludzie często specjalnie kreują swoją postać pod taką, jaką się od nich oczekuje. Można więc ukazywać odczuwane emocje poprzez tekst pisany, nie pisząc tak naprawdę o nich bezpośrednio, i tak samo odczuwać emocje, czytając tekst. Tak podświadomie robią ludzie. Pytanie jednak, czy komputer potrafiłby zrobić to samo – czyli odczytywać emocje z pisanego tekstu? Skąd można wiedzieć, czy zaufać analizie Watsona? Zarówno w usłudze Tone Analyzer jak i Personality Insights, IBM starał się odpowiedzieć na to pytanie.

Wiadomo, że serwis nie jest w stanie przewidzieć wszystkiego ze stuprocentową pewnością. Zawsze jest jakiś margines błędu. Obydwie opisywane usługi działają na podstawie uczenia maszynowego. Wykorzystują modele analizowania cech danej osoby na podstawie danych konstrukcji słów oraz uczą się na błędach. Przeprowadzono wiele badań naukowych aby serwis ten przetestować oraz ulepszać. Jedno z nich, najciekawsze to przewidywanie połączeń w pary na portalach randkowych. Jak można się domyślić, celem tego badania jest znalezienie podobieństwa (skorelowania) w tonach wypowiedzi dwóch osób mających konta

[29] J.B. Gleason, N.B. Ratnen, „*Psycholingwistyka*", Gdańskie Wydawnictwo Psychologiczne, Gdańsk 2005 (s. 17)

na danym portalu. W tym celu przeszukano mniej więcej 50 000 różnych profili i każdy z nich został poddany analizie Tone Analyzer'em. Następnie porównano wyniki, aby znaleźć korelacje. Opracowany został model statystyczny bazujący na podobieństwu tonów profili, w celu przewidzenia, czy jeden użytkownik z drugim się komunikuje. Następnie w modelu uwzględniono takie atrybuty, jak dane demograficzne. Rezultat porównano z faktycznym dopasowaniem, które zdefiniowano jako: użytkownicy, którzy ze sobą piszą, czyli: komunikują się za pośrednictwem owego portalu. Wyniki są bardzo ciekawe. Strony randkowe mają bowiem swoje predykatory co do przewidywania takich rzeczy. Okazuje się więc, że usługa ta jest w stanie poprawić ich działanie aż o 45%. IBM wykrył bowiem silną korelację pomiędzy liczbą wymienionych wiadomości, a podobieństwem tonów, co widać na rysunku 5 znajdującym się poniżej.[30]



Rysunek 5
Korelacja pomiędzy podobieństwem tonów a liczbą wymienionych wiadomości.[31]

[30] https://cloud.ibm.com/docs/services/tone-analyzer?topic=tone-analyzer-caseStudies (15.03.2020)
[31] https://cloud.ibm.com/docs/services/tone-analyzer?topic=tone-analyzer-caseStudies (15.03.2020)

2 Opis wykorzystywanych w aplikacjach technologii

Opisane w poprzednim rozdziale możliwości Watsona oraz ich przykładowe zastosowanie zostaną zaimplementowane za pomocą podanych poniżej technologii:

- Java (z frameworkiem Spring) - backend
- TypeScript (z frameworkiem Angular) - frontend

Jednak sama implementacja wybranych usług superkomputera to tak naprawdę niewielka część aplikacji. Istotną kwestię odgrywa zaprogramowanie całego systemu który ma zawierać przykładowe zastosowanie usługi, oraz interfejsu użytkownika. Aplikacje te nie potrzebują i nie używają bazy danych.

2.1 Aplikacje webowe – podstawy

Co to właściwie aplikacja webowa? Taką aplikację można nazwać również internetową. Otóż jest to „opatrzona w interfejs aplikacja uruchamiana na bazie przeglądarki stron www”[32]. Do korzystania z niej wystarczy jedynie dostęp do przeglądarki internetowej (oraz oczywiście, Internetu). Nie ma znaczenia, czy użytkownik używa telefonu, tabletu, komputera, laptopa. Można powiedzieć, że aplikacje takie „zainstalowane” są na serwerze, a użytkownik ma do nich dostęp poprzez Internet.[33] Przykłady takich serwerów to np. Apache HTTP Server lub nginx. Są to serwery HTTP (*Hypertext Transfer Protocol*, czyli protokół HTTP – w wersji szyfrowanej HTTPS), które to mają na celu serwowanie stron internetowych. Jak to dokładniej działa? Po wpisaniu adresu np. „www.facebook.com” do przeglądarki, zostaje wysłane zapytanie HTTP do serwera firmy u której została wykupiona ta usługa. Serwer z kolei odbiera to zapytanie, przetwarza je oraz odpowiada w formacie, który jest zrozumiały dla przeglądarki. Przeglądarka następnie wyświetla ową treść w formie strony.[34] Warto w tym całym systemie przytoczyć hasło takie jak protokół. W przypadku aplikacji webowej, która napisana została w języku X, protokół jest niezbędnym zbiorem reguł, aby dla każdego języka przeglądarka poradziła sobie z wyświetleniem oczekiwanej zawartości. Dla kontrastu od stron/aplikacji internetowych można przytoczyć aplikację desktopową - aby z niej korzystać, trzeba najpierw zainstalować ją na urządzeniu. Warto również rozumieć różnicę pomiędzy aplikacją internetową a stroną www. Aplikacja bowiem komunikuje się z serwerem, aby świadczyć konkretną usługę i reagować na działania użytkownika. W porównaniu do strony www, która

[32] https://www.empressia.pl/blog/72-aplikacje-webowe-czym-sa-jak-dzialaja-i-jakie-sa-ich-zalety (23.05.2020)
[33] https://www.samouczekprogramisty.pl/wprowadzenie-do-aplikacji-webowych/ (23.05.2020)
[34] https://www.samouczekprogramisty.pl/wprowadzenie-do-aplikacji-webowych/ (23.05.2020)

ma charakter czysto informacyjny, aplikacja jest interaktywna. Różnica jest oczywiście nie tylko w zastosowaniu strony, ale również w wykorzystywanych technologiach. Strony internetowe wykorzystują statyczne pliki (HTML, CSS). Aplikacje webowe natomiast zazwyczaj podzielone są na stronę serwera (w tym przypadku Java, ale też inne jak np. PHP, Node.js) oraz klienta (tutaj JavaScript i AngularJS). Technologii do wyboru jest naprawdę bardzo dużo.

Przykłady aplikacji webowych to sklep internetowy, portale społecznościowe, aplikacje wspomagające naukę online i procesy rekrutacji, a także systemy ERP (Enterprise Resource Planning). Dzięki takim systemom organizacja ma scentralizowane miejsce do zarządzania, planowania oraz przechowywania danych. Wszystko to ułatwia wspólna baza danych, która może zawierać dane z wielu sektorów firmy. Wchodzący na taką stronę mają dostępne informacje na bieżąco, co znacząco usprawnia komunikację w przedsiębiorstwie. Z pewnością zaletą takich aplikacji jest więc łatwa dostępność, jak i również bezpieczeństwo danych. W przypadku, gdy urządzenie lokalnie nagle wyłączy się, wszystkie pliki będą zapisane na zewnętrznym serwerze, a nie np. na pulpicie użytkownika. Dodatkowo, jednym z męczących aspektów, z którymi trzeba się mierzyć przy aplikacjach desktopowych jest regularne ich aktualizowanie. Jak można się spodziewać, w przypadku aplikacji internetowych ów problem znika.

2.2 REST API

REST, czyli z j. angielskiego „*Representational State Transfer*" to sposób komunikacji klient-serwer. Jest to styl architektury, czy też zbiór dobrych praktyk, który określa sposób w jaki dane są definiowane oraz umożliwia dostęp do nich. Jest on powszechnie dostępny i dziś szalenie często stosowany. „*RESTful Webservices* (inaczej RESTful web API) jest usługą sieciową zaimplementowaną na bazie protokołu HTTP i głównych zasad wzorca REST."[35] Autorem tego pomysłu jak i jego implementacji jest Roy Fielding. Napisał o nim w 2000 roku w swojej pracy doktorskiej. Chodziło między innymi o to, aby pomiędzy urządzeniem a urządzeniem korzystać z istniejących już protokołów, bez instalowania niepotrzebnych bibliotek. Przykładowo, aby używać istniejącego protokołu HTTP, nie trzeba pobierać do tego osobnych narzędzi na dane urządzenie. Kolejnym istotnym założeniem tego wzorca są zasoby – źródła danych oraz żądana akcja. Podczas komunikacji z serwerem najczęściej stosowanymi metodami są GET i POST, a zaraz po nich PUT, DELETE. Dzięki nim już programista jest w stanie napisać podstawowe API pozwalające na kolejno: odczyt danych, ich zapis, aktualizację,

[35] http://yarpo.pl/2012/07/29/rest-ciekawszy-sposob-na-komunikacje-client-server/ (24.05.2020)

oraz usuwanie. Mniej już znane metody to: PATCH, OPTIONS, HEAD. Pierwsza z nich jest podobna do PUT, jednak różni się tym, że polega jedynie na częściowym zmodyfikowaniu danych. Druga z nich zajmuje się zwracaniem informacji o tym, na czym polegają inne metody i operacje. Jest to wyjątkowo rzadko używane zapytanie. HEAD natomiast ma dużo wspólnego z GET. Różnicą jest to, iż HEAD nie zwraca danych w odpowiedzi. Jest to dobry sposób na sprawdzenie co może zwrócić żądanie GET, zanim zostanie wykonane, zwłaszcza przed pobraniem dużej ilości informacji. Jak wygląda cały ten proces? Jako pierwsze zostanie wyjaśnienie krótko pojęcie „endpoint". Endpoint to inaczej URL, który zwraca się do serwera. Jest to miejsce z którego API pobiera potrzebne informacje i następnie je zwraca. Przykład: istnieje strona pod adresem „www.test.pl". Endpoint to np. „/uzytkownik/info". Zatem po wejściu na URL „www.test.pl/uzytkownik/info", zostanie wysłane zapytanie do serwera o informacje dotyczące użytkownika oraz informacji o nim. A propos wzorca REST, zapytanie to będzie metodą GET, ponieważ będzie ono jedynie wyciągało informacje w celu zaprezentowania ich – nie jest to zapis, aktualizacja ani usuwanie. Proces działania API zatem wygląda następująco: Użytkownik wysyła zapytanie w postaci adresu URL/endpointa do interfejsu, który to przetwarza i zwraca odpowiedź. Odpowiedź jak i przesyłane dane mogą być w formie zarówno XML, jak i JSON, tekstowej, oraz binarnej. W większości przypadków jest to JSON lub XML.[36] Z zalet REST API można wymienić uniwersalność, ponieważ zarówno dla telefonu komórkowego aplikacja na telefon może korzystać z tego samego API co aplikacja webowa na komputer. Tworzenie endpointów jest intuicyjnym zabiegiem, pomagają one bowiem zorientować się w aplikacji. Format JSON (JavaScript Object Notation) jest nowszym oraz według wielu wygodniejszym sposobem na przekazywanie danych, i jest wspierany przez omawiany wzorzec. Ponadto zdecydowanym plusem jest możliwość oddzielenia warstwy klienta od serwera, oraz bardzo łatwe testowanie endpointów. Do tego drugiego zadania istnieje wspaniałe narzędzie Postman, lecz ponieważ jest to zagadnienie luźno związane z tematem pracy, nie będzie tutaj omawiane.

2.3 Java – wprowadzenie

Jest to wykorzystywana do tworzenia aplikacji technologia, będąca językiem programowania. Wielu myli ją z JavaScript'em, choć to zupełnie dwie różne rzeczy - to frontend'owy język i służy do pisania stron internetowych. Ten temat jednak zostanie poruszony później. Java powstała końcem 1995 roku i od razu zyskała uznanie programistów. Stworzona została z myślą o tym, aby być łącznikiem pomiędzy informacją a użytkownikami,

[36] MarkLogic Corporation, „MarkLogic Server – REST Application Developer's Guide", 2019, s. 11, wersja pdf

niezależnie od tego, skąd informacje pochodziły - czy to były bazy danych, serwis czy jakiekolwiek inne źródło.[37] Teraz jest w czołówce najbardziej popularnych języków programowania. W rankingu ze stycznia 2020 roku, znajduje się na 2 miejscu razem z Pythonem, ustępując miejsca jedynie językowi JavaScript.[38] Przede wszystkim jest to współbieżny (co oznacza w skrócie, że współpracuje z wieloma wątkami naraz, przez co aplikacje zyskują na szybkości) i obiektowy język programowania wysokiego poziomu, służący do ogólnego zastosowania. Obiektowy, to znaczy, że istnieje możliwość definiowania w kodzie różnych abstrakcyjnych elementów, które pochodzą ze świata realnego, oraz przypisywania im atrybutów. Przykładowo można utworzyć w kodzie abstrakcyjny „Dom", który dodatkowo przedstawiony zostanie jako zestaw takich atrybutów, jak liczba pokoi, adres, liczba domowników itp.

Warto znać zwięzłą historię Javy. Zaczęło się to tak, iż grupa firmy Sun Microsystems pod kierunkiem Jamesa Goslinga oraz Patricka Naughtona wpadła na pomysł stworzenia prostego języka, który to działał będzie na różnych platformach, niezależnie od ich parametrów. Był to rok 1991. Ów projekt nazwano Green. Wraz z jego rozwojem wymyślono sposób, w jaki główne założenie Javy będzie spełnione. Sposób ten to kompilacja do kodu pośredniego, aby dopiero ten był uruchamiany. Zostaną wyjaśnione teraz podstawowe pojęcia, w celu uniknięcia nieporozumień. Java to oczywiście język, w którym pisane są aplikacje. Maszyna wirtualna Javy (ang. JVM – Java Virtual Machine) je wykonuje. Natomiast środowisko Javy jest to nieocenione narzędzie które ułatwia, a wręcz umożliwia pisanie tych aplikacji. W czasach powstania Javy było to nowatorskie rozwiązanie. Zostało entuzjastycznie przyjęte i tak dziś, wraz z rozwojem techniki, metoda ta jest na porządku dziennym. Platforma .NET firmy Microsoft działa na podobnej zasadzie (powstała ona parę lat po Javie).[39] Tak więc program źródłowy kompiluje się do kodu bajtowego, a w tej postaci może już być wykonany przez maszynę wirtualną. Pierwsza wersja Javy (1.0) powstała w roku 1996 i nie osiągnęła wielkiej popularności. Jako, że przedsiębiorstwo Sun szybko to zauważyło, błędy zostały naprawione i stosunkowo niedługo powstała jej następna wersja (1.1), rozszerzona o nowe biblioteki i funkcjonalności. Po niej, wersja 1.2 przyniosła w 1998 roku między innymi nową nazwę: „Java 2 Standard Edition Development Kit version 1.2". W podobnym czasie, bo około roku 2000 powstały jeszcze dwie jej wersje: Micro Edition oraz Enterprise Edition. I tak do dzisiaj powstawały coraz to nowsze wersje Javy, różniące się od siebie głównie dodanymi funkcjonalnościami oraz wydajnością. Jedna z większych zmian zaszła w Javie 5, ponieważ

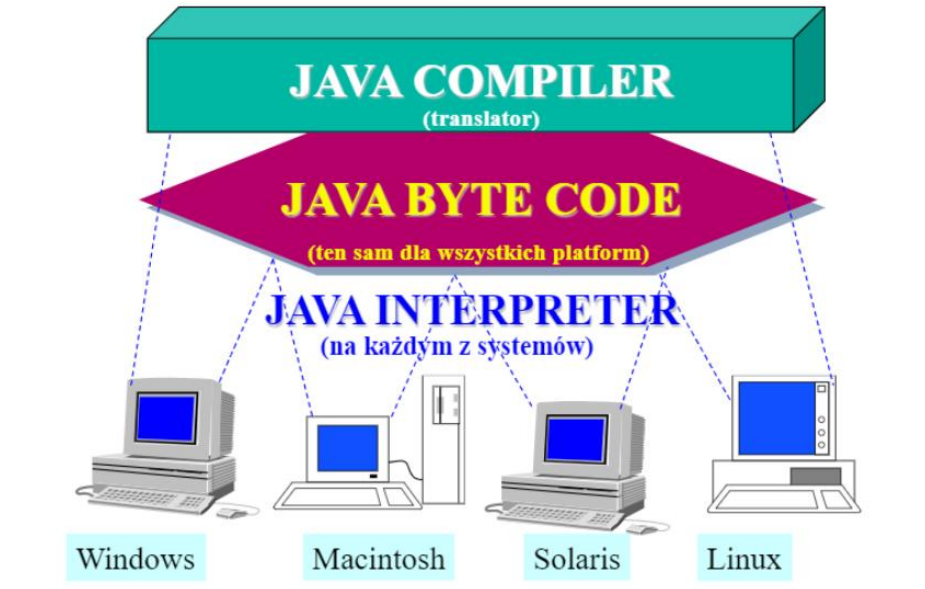
[37] C. S. Horstman, *Java Podstawy*, Gliwice, 2016
[38] https://itwiz.pl/jezyki-programowania-ranking-styczen-2020/ (15.05.2020)
[39] B. J. Evans, D. Flanagan „Java w pigułce" Wydanie VI, Helion S.A , Gliwice 2015

zostały wprowadzone klasy generyczne i statyczny import. Kolejna w Javie 8, gdzie wprowadzono elementy z programowania funkcyjnego.

Java oraz jej pierwotne koncepcje zostały zaczerpnięte głównie z języka C++ (m.in. słowa kluczowe, część składni) oraz SmallTalk, które na potrzeby nowego, bardziej uniwersalnego zastosowania zmodyfikowano.[40] W tworzenie standardowych implementacji Javy duży wkład mają takie przedsiębiorstwa, jak: IBM, Red Hat, HP, SAP, Apple i Fuitsu. Teraz steruje nią firma Oracle Corporation, która to kupiła twórcę Javy, czyli Sun Microsystems. Istnieją cztery rozgałęzienia tego języka programowania:

- Java ME (Mobile/Micro Edition),
- Java SE (Standard Edition)
- Java EE (Enterprise Edition)
- Java FX

Wszystkie z wymienionych platform składają się z wymienionej już wirtualnej maszyny Java oraz interfejsu API. Pozwala to napisanym aplikacjom działać na dowolnym systemie, zachowując wszystkie zalety tego języka. Główną z nich jest niezależność od platformy, co zostało zilustrowane na rysunku 6. Oprócz tego można wymienić: moc, bezpieczeństwo, stabilność, łatwość programowania (pomimo rozległości języka).[41]



Rysunek 6
Java - niezależność od platformy[42]

[40] https://pl.wikipedia.org/wiki/Java (14.01.2020)
[41] https://riptutorial.com/pl/java/example/27916/roznice-miedzy-java-ee--java-se--java-me-i-javafx (15.05.2020)
[42] https://docplayer.pl/38866069-Krotka-historia-java.html (15.05.2020)

Wymienione specyfikacje zostaną teraz szerzej omówione.

Java Mobile/Micro Edition (Java ME)

Platforma ta została stworzona do programowania aplikacji mobilnych, które działają na urządzeniach o znacznie bardziej ograniczonych zasobach niż standardowe komputery czy laptopy (np. na telefonach komórkowych czy palmtopach). Interfejs API to podzbiór JAVA SE API, jednak z wyłączeniem niepotrzebnych bibliotek oraz klas, z uwagi na to, iż zazwyczaj mniejsze urządzenie posiada wolniejszy procesor oraz mniej pamięci. Taką okrojoną zawartość Javy SE można nazwać konfiguracją, która jest uzupełniana przez profile. Profile dodają swoje klasy do istniejących już klas, dzięki czemu określone zadania zostaną wykonane zgodnie z ograniczeniami konkretnych urządzeń. Te z kolei mogą być rozszerzone jeszcze o pakiety opcjonalne. Taki sposób działania i różnorodność zapewniają dużą elastyczność projektantom oraz programistom. Przydaje się to przy rozmaitych konfiguracjach docelowego sprzętu, na którym aplikacje mają działać. Java ME wykorzystywana jest na przykład do tworzenia gier na telefony komórkowe.[43]

Java Standard Edition (Java SE)

Specyfikacja Java SE zawiera najbardziej podstawowe funkcjonalności tego języka. „Definiuje wszystko, od podstawowych typów i obiektów języka programowania Java do klas wysokiego poziomu używanych do pracy w sieci, bezpieczeństwa, dostępu do bazy danych, programowania graficznego interfejsu użytkownika (GUI) i analizy składni XML.” [44] W przeciwieństwie do Javy ME, umożliwia ona uruchomienie napisanych w Javie aplikacji na komputerach stacjonarnych czy serwerach. Jest podstawą Javy EE, ponieważ zawiera klasy dotyczące aplikacji webowych.

Java FX

Jeszcze niedawno do tworzenia graficznego interfejsu użytkownika w Javie używano Swing i AWT – były to rekomendowane technologie. Dlatego też jedną z większych zmian w Javie 8 jest zamiana tej rekomendacji na Javę FX, która tak naprawdę była dostępna już wcześniej, aczkolwiek dopiero w tej wersji weszła w skład podstawowych bibliotek. W jej skład wchodzi skryptowy język Java FX Script oraz omawiana już Java ME dla urządzeń mobilnych. Z zalet jej używania można wymienić większą czytelność kodu oraz bardziej nowocześnie wyglądające aplikacje, przy jednoczesnym zachowaniu wydajności. Co jeszcze

[43] https://pl.wikipedia.org/wiki/Java_Platform,_Micro_Edition (15.05.2020)
[44] https://riptutorial.com/pl/java/example/27916/roznice-miedzy-java-ee--java-se--java-me-i-javafx (15.05.2020)

się zmieniło? Otóż można zmieniać wygląd aplikacji w języku XML, Swing natomiast umożliwiał to jedynie za pośrednictwem kodu Javy. Edycja poprzez XML ma wiele zalet: jest szybka oraz umożliwia podzielenie zadań w projekcie: jedna osoba zajmuje się wyglądem, a inna zachowaniem programu. Java FX umożliwia też odtwarzanie mediów (np. filmów, plików MP3). Dodatkowo można zmieniać wygląd za pomocą stylów CSS (kolor/rozmiar czcionki, przycisków i innych elementów), co w porównaniu do wcześniejszych metod używanych w tym celu (setSize(), setColor()) jest ogromnym ułatwieniem. A co gdy nadal użytkownik tęskni za starymi metodami Swinga? Nic nie stoi na przeszkodzie aby dodać specjalną bibliotekę do aplikacji stworzonej w Javie FX o nazwie „SwingNode”. [45]

2.4 Java EE, czyli aplikacje webowe w Javie

Java Enterprise Edition (Java EE)

Jest to zbudowany na Javie SE standard do tworzenia zorientowanych na usługi aplikacji biznesowych. Te z kolei mają swojego brata, czyli aplikacje webowe. Tak naprawdę Javę EE tworzy zbiór wszelakich frameworków i technologii, ukierunkowanych w stronę tworzenia aplikacji biznesowych. Jest ich tak dużo, że nie sposób znać każdą. Ważne pojęcia, które należy wymienić w tym temacie to między innymi:

- JavaServer Pages (JSP, czyli HTML + Java),
- JavaServer Faces (JSF, czyli technologia prezentacji)
- JavaServer Pages Standard Tag Library (JSTL),
- Java Servlet (serwlety),
- Java Persistence API (JPA, czyli zarządzanie danymi, oraz framework Hibernate)
- Object-Relational Mapping (ORM)
- Enterprise JavaBeans (EJB).[46]

Jak wiadomo, aplikacje webowe działają na specjalnych serwerach. Najpopularniejszym serwerem obecnie jest Apache Tomcat. Do wyboru jest także między innymi serwer twórców Javy, firmy Sun Microsystems – GlassFish. Pojęć tych jest za dużo, aby je tutaj dobrze omówić. Jednak jako, że właściwie wszystkie aplikacje webowe działają na podstawie servletów, termin ten zostanie zwięźle wyjaśniony. Jest to można powiedzieć taka klasa w Javie, która może

[45] https://javastart.pl/baza-wiedzy/frameworki/javafx (15.05.2020)
[46] Krzysztof Rychlicki-Kicior „Java EE 6 – Programowanie aplikacji WWW”, Helion, Gliwice 2010

odbierać żądania oraz generować odpowiedzi. Jest niezależna od protokołów, ponieważ Java EE stawia na uniwersalność, jednak zdecydowana większość wykorzystuje protokół HTTP. Ciężkie byłoby tworzenie aplikacji za pomocą samych servletów, dlatego też opracowano wiele technologii, które mają na celu proces ten ułatwić, i jednocześnie rozszerzyć ich możliwości. Jedną z nich jest właśnie Spring.

2.4.1 Spring

Trudno nie zauważyć, iż Java EE ma tyle komponentów oraz interfejsów, że można się w tym łatwo pogubić. Nawet posiadając dobre narzędzia i posługując się wysokopoziomowym językiem programowania, tworzenie aplikacji webowych jest dość trudnym zadaniem. Zwłaszcza, gdy platformy obiecują o wiele więcej, niż mają do zaoferowania i okazuje się, że proces implementacji danego rozwiązania nie wygląda tak kolorowo jak można byłoby tego oczekiwać.[47] Spring to niesamowity framework, który posiada ogromny zestaw adnotacji i bibliotek wspomagających pisanie aplikacji w Javie (nie tylko webowych, a dowolnych). Zawiera on specjalne klasy, które znacznie skracają proces pisania programu, m.in. do komunikacji z bazą danych, otrzymywania oraz wysyłania zapytań http do serwera, czy też samego stawiania aplikacji. Dzięki temu programista może pominąć pewne aspekty, którymi zazwyczaj musiałby się przejmować. Wystarczy, że skupi się na opracowaniu logiki biznesowej, zostawiając w tyle niepotrzebne konfiguracje. Ma to swoje przełożenie na produktywność, oszczędność czasu oraz linijek kodu – z wykorzystaniem frameworku Spring można zauważyć jak kod staje się czytelniejszy i jednocześnie prostszy. Z zalet można wymienić również wydajność – szybkie uruchamianie aplikacji (Spring Boot), zoptymalizowane wykonanie. Spring został stworzony przez SpringSource i jest najpopularniejszym frameworkiem open source do Javy na świecie. Kwintesencją Springa jest kontener **IoC** (Inversion of Control, czyli odwrócenie sterowania) oraz **DI** (dependency injection, czyli wstrzykiwanie zależności).

Inversion of Control oraz **Dependency Injection**

W programowaniu obiektowym istnieją zależności pomiędzy obiektami. Zależność to gdy obiekt potrzebuje jakiegoś innego obiektu w celu zrealizowania danej funkcji. Na przykład jeśli obiekt A potrzebuje obiektu B, współpracuje z nim, to obiekt B jest zależnością obiektu

[47] R. Johnson, J. Hoeller, A. Arendsen, C. Sampaleanu, R. Harrop, T. Risberg, D. Davison, D. Kopylenko, M. Pollack, T. Templier, E. Vervaet, P. Tung, B. Hale, A. Colyer, J. Lewis, C. Leau, M. Fisher, S. Brannen, R. Laddad, A. Poutsma*, Spring. Java/j2ee Application Framework*. *The Spring Framework – Reference Documentation,* 2004-2008

A. Dependency Injection to wzorzec projektowy, który polega na tym, że zależności obiektu są podawane jako argumenty konstruktora lub do metod typu setter. Nazywa się to wstrzykiwaniem zależności. Co ważne, są one wstrzykiwane już w momencie tworzenia obiektu. Zajmuje się tym kontener Inversion of Control. Obiekty stworzone i zarządzane przez IoC nazywa się komponentami (ang. *bean*). Komponenty mogą być nawet POJO (*Plain Old Java Object*), czyli najzwyklejszymi klasami Javy, nie implementującymi interfejsów. Przy tworzeniu aplikacji nie trzeba implementować wzorców projektowych takich jak fabryka czy lokalizator usługi, ponieważ zadaniem kontenera IoC jest właśnie tworzenie i połączenie obiektów za pomocą wstrzykiwania zależności. Informacje o tym jakie obiekty z jakimi chcemy połączyć mogą być zdefiniowane na trzy sposoby:

- w pliku konfiguracyjnym o nazwie metadane konfiguracyjne (format XML),
- w kodzie
- poprzez adnotacje Javy

Kontener IoC odczytuje przekazane informacje i w ten sposób tworzy obiekty. Aplikacja posiadająca klasę MyController i MyService może przykładowo posiadać plik konfiguracyjny, jaki pokazano na rysunku 7:

```
<beans .....>
    <bean id="myController" class="sample.spring.controller.MyController">
        <constructor-arg index="0" ref="myService" />
    </bean>
    <bean id="myService" class="sample.spring.service.MyService"/>
</beans>
```

Rysunek 7
Fragment z pliku metadane konfiguracyjne (format XML) [48]

Z pliku łatwo wywnioskować, iż za pomocą <bean> można zadeklarować komponent, a w <constructor-arg> w parametrze „ref" jest możliwość podania obiektu, którego klasa MyController potrzebuje. Zatem obiekt MyService (również zadeklarowany jako <bean>) zostanie wstrzyknięty przez IoC jako argument konstruktora przy tworzeniu egzemplarza MyController. Ciekawostka: w tym celu kontener Springa wykorzystuje API Java Reflection. Mechanizm refleksji w Javie polega w skrócie na tym, że istnieje możliwość manipulowania interfejsami, klasami (np. tworzenie obiektów), metodami w trakcie działania programu, nie wiedząc jak się one nazywają. Działanie kontenera IoC jest pokazane na rysunku 8.

[48] J. Sharma, A. Sarin „Spring Framework – Wprowadzenie do tworzenia aplikacji" Wydanie II, Helion, Gliwice 2015

Projekty Spring

Spring to właściwie wiele projektów. Każdy z nich służy czemuś innemu: konfiguracji, pracy z bazą danych, bezpieczeństwu. Można wymienić spośród nich te najczęściej używane: Spring Boot, Spring Framework, Spring Data, Spring Cloud, Spring Cloud Data Flow, Spring

Rysunek 8
Sposób działania kontenera IoC [49]

Batch, Spring Security, Spring Integration i wiele innych. Na oficjalnej stronie (https://spring.io/) jest na ten temat dużo informacji. W ramach tej pracy zostanie przybliżone pojęcie Spring Boot, ponieważ jest on wykorzystywany w obydwóch aplikacjach demonstrujących usługi IBM Watson AI. Najprościej mówiąc jest to szkielet aplikacji. Opiera się o Spring Framework. Spring Boot znacznie skraca czas tworzenia gotowej aplikacji webowej. Posiada wiele wbudowanych modułów, które pozwalają na jej szybszą i mniej skomplikowaną implementację. Zawiera kontekst aplikacji, czyli połączenie jej wielu aspektów. Spring Boot jest już zintegrowany z serwerem o nazwie Tomcat. Jest on wbudowany i automatycznie konfigurowany. Aby uruchomić aplikację wystarczy wywołanie metody main() w klasie oznaczonej adnotacją @SpringBootApplication. Tylko tyle, ponieważ wszystko dzieje się pod spodem. To naprawdę magiczne narzędzie. W rozdziale trzecim zostanie to zilustrowane na przykładzie napisanej aplikacji. Spring Security jak nazwa wskazuje jest pomocne w kwestii bezpieczeństwa aplikacji, natomiast Spring Data wspomaga komunikację aplikacji z bazą danych. Aby aplikacja mogła używać danego projektu Springa, czy też innych funkcjonalności i bibliotek które nieraz są programiście potrzebne, dobrze jest użyć do tego takiego narzędzia, jak Maven lub Gradle.

[49] J. Sharma, A. Sarin „Spring Framework – Wprowadzenie do tworzenia aplikacji" Wydanie II, Helion, Gliwice 2015

2.4.2 Maven

W aplikacjach tworzonych w ramach tej pracy używany jest akurat Maven, natomiast równie często stosuje się narzędzie zwane Gradle. Wybór należy do programisty. Maven to narzędzie, które wspomaga zarządzać projektem, a dokładniej jego cyklem życia. Ułatwia on tworzenie raportów, kompilację, testowanie, kontrolę projektu, zarządzanie zależnościami i nie tylko. Jeśli aplikacja korzysta np. ze Springa czy Hibernate, będą potrzebne dodatkowe pliki JAR zawierające biblioteki tego frameworku. Trzeba je zaimportować. Można to zrobić pobierając owe pliki z oficjalnych stron a następnie ręcznie dodać je do docelowego projektu. Druga opcja to właśnie użycie Mavena. Za jego pomocą można dodać różne zasoby jako „dependencje" (inaczej zależności) w specjalnym pliku, w którym to charakteryzuje się mavenowy projekt, czyli pom.xml. Jest to bardzo proste w porównaniu do wcześniejszego podejścia. Owe narzędzie pobiera bowiem za programistę pliki JAR odpowiednich projektów, które zostały wskazane w pliku pom.xml i sprawia, że te pobrane biblioteki są już dostępne w kodzie. Zatem osoba zarządzająca aplikacją musi jedynie wyszukać odpowiednią „dependencję" z oficjalnej strony (https://mvnrepository.com/) oraz wkleić ją do wspomnianego pliku. Jest to nieoceniona pomoc również przy budowaniu projektu. Polecenie „mvn build" wykorzysta wszystkie informacje i zależności zadeklarowane w pom.xml i na ich podstawie zbuduje aplikację. Takich poleceń jest więcej, na przykład „mvn clean" , które jak sama nazwa wskazuje wyczyści folder „target", gdzie po poleceniu build będą znajdowały się pliki zbudowanego projektu. Początkowo Maven zaprojektowano specjalnie do projektu Jakarta Turbine, aby upraszczał probesy budowlane. Jednak tych projektów było więcej niż jeden, i tak w końcu firma Apache zaprojektowała takiego Mavena, który może budować wiele projektów naraz, wraz z innymi funkcjonalnościami które zostały wymienione na początku. Dodawanie dependencji do projektu zostanie przedstawione w praktyce w rozdziale trzecim.

2.5 TypeScript, czyli wzbogacony JavaScript

JavaScript to najprościej mówiąc obiektowy język programowania po stronie klienta, należący do technologii frontendowych. Wprowadził on nieco interaktywności do statycznych stron internetowych. Choć między innymi w języku tym napisany został Node.js, który to uznaje się językiem backendowym – po stronie serwera. Dzięki temu można uznać, że za pomocą jednego języka programista jest w stanie napisać pełną aplikację webową. Powstało zatem takie hasło, jak „JavaScript everywhere", co oznacza „JavaScript jest wszędzie".

Początek tego języka to 1995 rok pod nazwą LiveScript. Jego pierwszą wersję zawdzięcza się Brendonowi Eich. LiveScript pojawił się wtedy w Netscape 2.0. Nie minęło wiele czasu aż dwie te firmy: Netscape i Sun Microsystems (którą kojarzy się z Javą) porozumiały się i tym sposobem wraz z rokiem 1996 w przeglądarce tej zagościł JavaScript 1.0.[50] Z czasem osadził się on również w Internet Explorer 3 i innych przeglądarkach – wtedy określany jako JScript. Jednak była to już implementacja firmy Microsoft. Po jakimś czasie pomiędzy JScript a JavaScriptem pojawiło się za dużo różnic, a więc stwierdzono, iż powstanie wspólna specyfikacja. Podstawowa, międzynarodowa wersja jest w rezultacie zgodna ze standardem ECMA-262 (ECMA Script). Definiuje ona główne typy danych oraz semantykę języka. Dziś JavaScript jest najpopularniejszym językiem programowania. Jest lekki i interpretowany, zintegrowany z Javą i HTML. Jest również obiektowy – pojęcie to zostało już wyjaśnione przy Javie. Dodatkowo jest wieloplatformowy, co oznacza, że będzie dobrze działał na różnych systemach operacyjnych. Kod napisany w tym języku można osadzić w dokumencie HTML, oznaczony znacznikami <script>. Przykład:

```
<script type="text/javascript">

      kod JavaScript

</script>
```

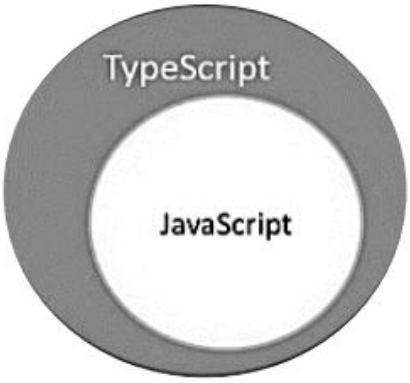
Można też użyć mniej bezpośredniego sposobu i w powyższych znacznikach podać ścieżkę do pliku JavaScript, jako atrybut „src”. Dzięki temu strona internetowa nie musi być statyczna, lecz może być dynamiczna czyli może pozwolić sobie na interakcję z użytkownikiem, na bieżąco kontrolując dokument HTML. Przykład takiego działania to walidacja, czy w formularzu poprawnie został wpisany adres e-mail danego użytkownika, a jeśli nie, to pole wypełnione niepoprawnie zostanie podświetlone na czerwono. Innym przykładem może być obsługa zdarzeń i przycisków. Językiem tym można posługiwać się nawet bezpośrednio z przeglądarki www, jest on bowiem w nią wbudowany.[51]

TypeScript

Niestety, gdy kodu JavaScript jest dużo, staje się on chaotyczny, a co za tym idzie: ciężko czytelny. Nie poprawia sytuacji również fakt, iż język ten znacznie kuleje jeśli chodzi o orientację obiektów, czy też sprawdzanie typów. Dodatkowo błędy sprawdzane są w trakcie kompilacji, co jednocześnie przekreśla działanie tego języka jako pełnoprawnej technologii backendowej (po stronie serwera).

[50] Andrzej Skowron „Technologie internetowe client-side na przykładzie języka JavaScript” w wersji .pdf
[51] TutorialsPoint www.tutorialspoint.pl „JavaScript language” w wersji .pdf s. 11

TypeScript został stworzony z myślą o naprawieniu największych błędów JavaScriptu, czyli nawiązując do powyższych cech jest to silnie typowany, obiektowy, kompilowany język open-source. Właściwie to zarówno język programowania, jak i zbiór narzędzi i bibliotek. Jego autorem jest Anders Hejlsberg z firmy Microsoft. Człowiek ten również zaprojektował C#. TypeScript ujrzał światło dzienne w 2012 roku (w wersji 0.8). Początkowo język ten był dostępny jedynie w Microsoft Visual Studio, jednak wraz z rosnącą popularnością pojawił się w takich IDE, jak najpopularniejsze: JetBrains WebStorm, Eclipse, Visual Studio Code i inne. W 2014 roku ogłoszono powstanie nowego kompilatora dla TypeScriptu, szybszego aż pięciokrotnie od poprzedniego.[52] Od tamtej pory język jest również dostępny na popularnej platformie GitHub, gdzie każdy może dodać swój komentarz czy sugestię odnośnie danego błędu i tym samym przyczynić się do rozwoju języka.



Rysunek 9
TypeScript jako nadzbiór JavaScriptu[53]

Jak to zostało zilustrowane na rysunku 9, TypeScript to nadzbiór JavaScriptu. W zasadzie wystarczy, że programista zna JavaScript i już jest w stanie pisać w TypeScripcie – czysto teoretycznie. Co ciekawe, kod TypeScript, aby zostać wykonanym konwertuje się w pierwszej kolejności do JavaScriptu. Może on korzystać z dowolnych bibliotek tego języka, frameworków i narzędzi. Z przedstawionych ciekawostek można również wywnioskować fakt, iż każdy plik o rozszerzeniu .js można zamienić na .ts, a plik nadal będzie mógł zostać wykonany, a nawet użyty w projekcie napisanym w języku TypeScript. Szybkim wnioskiem może być zatem, że TypeScript to tak naprawdę JavaScript, ale wzbogacony. Bazuje on bowiem na tej samej specyfikacji co JavaScript, czyli ECMA Script 5 oraz 6 (nowsze, istnieje bowiem 6 edycji standardu ECMA-262). Przejrzyście ilustruje to rysunek 10.

[52] https://pl.wikipedia.org/wiki/TypeScript (21.05.2020)
[53], [51] TutorialsPoint www.tutorialspoint.pl „TypeScript" w wersji .pdf s. 1

Rysunek 10
TypeScript i ECMAScript - zależności.[54]

Wśród zalet TypeScript należy wymienić:

- Możliwość kompilacji programu. JavaScript jest językiem interpretowanym, co oznacza, że dopiero w trakcie uruchomienia jest wczytywany, walidowany i wykonywany. Czyli programista o błędzie dowie się dość późno, a aby go znaleźć musi poświęcić dużo czasu. Znacznie wygodniejszym rozwiązaniem jest język kompilowany, jak TypeScript, który już podczas kompilacji jest w stanie znaleźć błędy i pokazać je programiście. Przydatne staje się podkreślanie na czerwono błędów zanim skrypt zostanie uruchomiony.
- Silne statyczne typowanie (opcjonalne), oraz wnioskowanie o typie dzięki TypeScript Language Service. Można podać typ zmiennej, lecz jeśli programista tego nie zrobi, wtedy dzięki TLS typ ten może zostać wywnioskowany patrząc na wartość zmiennej.
- Obsługa definicji typów dla bibliotek języka JavaScript, co już zostało wspomniane. Plik mający rozszerzenie „d.ts" posiada definicję zewnętrznych bibliotek tego języka. Oznacza to, iż kod TypeScript może korzystać z bibliotek JavaScript.
- Możliwość podejścia obiektowego w programowaniu. Kod napisany w TypeScript obsługuje koncepcje takie, jak interfejsy, dziedziczenie, klasy i inne.[55]

Wydaje się, że lepiej być nie może, a jednak. Mowa o frameworku Angular, który został napisany w tymże języku.

2.6 Angular

To niezwykle pomocny i ułatwiający programowanie framework JavaScriptu napisany w języku TypeScript do tworzenia aplikacji webowych. Na początku miał być wersją 2 powstałego wcześniej AngularJS, lecz Google postanowił, iż Angular będzie osobnym frameworkiem. Przy jego użyciu, programista może poświęcić swoją uwagę jedynie na

[55] TutorialsPoint www.tutorialspoint.pl „TypeScript" w wersji .pdf s. 2 (21.05.2020)

najważniejsze sprawy związane z aplikacją i kodem. Aplikacje napisane w Angularze nazywamy „Single Page Applications” czyli aplikacjami jednostronicowymi. Aplikacja taka dzieli się na komponenty. Pojedynczy komponent z kolei dzieli się na trzy części:

- Template (Szablon) - zawiera elementy HTML, które renderowane są na stronie
- Class (Klasa) – zawiera pola oraz metody, których używa się w częściach HTML i w ten oto sposób są wyświetlane na stronie,
- MetaData – składa się z Class i Template, tworzy komponent. Pole „selector” na przykład mówi jak w dokumencie HTML oznaczyć dany komponent w celu wyświetlenia go.[56]

Dzięki takiemu rozwiązaniu kod jest czytelny, łatwo znaleźć dany fragment i aż ciężko się pogubić. Jeśli programista zmieni wartość pola w elemencie Class, wtedy równocześnie aktualizuje się ono w Template i wyświetlaniu na stronie. Pola takie można umieścić w dokumencie HTML w sposób zaprezentowany na rysunku 11.

src/app/app.component.html

```
<h4>{{recommended}}</h4>
<img [src]="itemImageUrl2">
```

Rysunek 11
Osadzenie pola klasy w dokumencie HTML w obrębie tego samego komponentu na dwa sposoby[57]

Na rysunku 11 można zauważyć, że są dwa sposoby oznaczana pól danej klasy w dokumencie HTML. Wartość pola o nazwie „recommended” zostanie wyświetlona jako nagłówek (dzięki nawiasom {{…}}), natomiast jako zdjęcie będzie ukazany obrazek o nazwie „itemImageUrl2” (ponieważ element „src” został otoczony nawiasami […]). Jeśli pole „recommended” nie byłoby typu string lecz obiektem posiadającym swoje pola, do tychże można odnieść się po znaku kropki, np. „recommended.name”. Zabiegi te są bardzo intuicyjne i znacznie ułatwiają komunikacje widoku z danymi i logiką biznesową. W przypadku pola, które jest listą czy też tablicą elementów, istnieje łatwy sposób aby w dokumencie HTML się po nich przeiterować , pokazany na rysunku 12.

[56] https://www.youtube.com/watch?v=8sC55WKzJW4 (dostęp: 24.05.2020)
[57], [55] https://angular.io/guide/template-syntax - Oficjalna dokumentacja Angular (24.05.2020)

src/app/app.component.html (template input variable)

```
<ul>
  <li *ngFor="let customer of customers">{{customer.name}}</li>
</ul>
```

Rysunek 12
Iteracja po elementach kolekcji "customers" i wyświetlanie imienia każdego z nich[58]

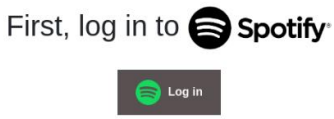
Angular to zgrabne połączenie wszystkich elementów aplikacji, szybkie poruszanie się po nich oraz intuicyjny kod. W tym oto frameworku zostały napisane dwie przedstawione w następnym rozdziale aplikacje, a szczególności ich części po stronie klienta.

3 Aplikacje i proces ich powstawania

Rozdział trzeci ma na celu zaprezentowanie tytułowych aplikacji webowych, które korzystają z serwisów superkomputera Watson, oraz po krótce sposobu ich powstawania.

3.1 Opis aplikacji demonstrującej funkcję Tone Analyzer

Zanim zostaną przedstawione szczegóły implementacji, należy wyjaśnić na czym polega budowana aplikacja. Otóż jak już zostało wspomniane, obydwa serwisy (Tone Analyzer jak i Personality Insights) potrzebują danych w formie tekstu, aby przeprowadzić jakąkolwiek analizę. Zatem pierwszym problemem, z którym można się zetknąć w celu przetestowania usługi czy też zademonstrowania jej działania w rzeczywistości jest to, skąd wziąć tekst do analizy. W przypadku usługi Tone Analyzer owe dane to generalizując, tekst polubionych przez użytkownika piosenek z popularnej aplikacji Spotify. Spotify jest to usługa, tzw. serwis streamingowy, który oferuje słuchanie głównie muzyki, ale również podcastów i filmów.[59] Według sprawozdania udostępnionego przez ów portal, w czwartym kwartale 2019 roku z tego serwisu korzystało już 271 milionów użytkowników, a liczba ta stale rośnie.[60] Aplikacja webowa demonstrująca wybraną funkcję Watsona działa następująco: przy wejściu na stronę można zalogować się poprzez Spotify, co zostało pokazane na rysunku 13.

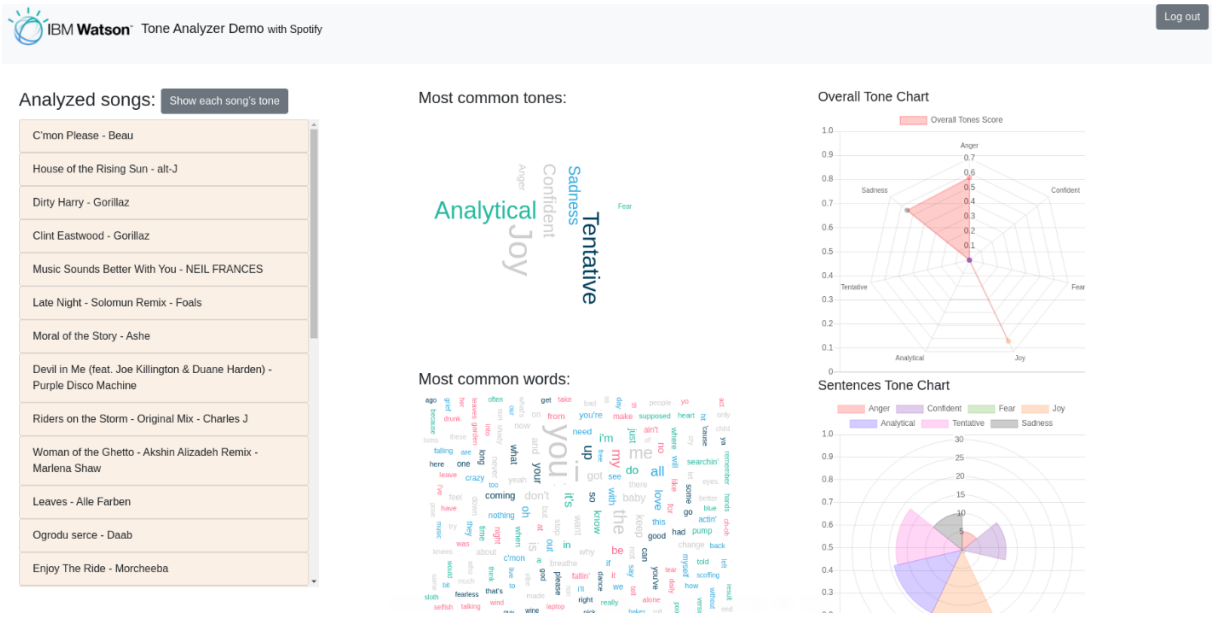


Rysunek 13
Ekran widoczny na stronie logowania w aplikacji demonstrującej działanie Tone Analyzer
Źródło: opracowanie własne

Aby aplikacja mogła pobrać ulubione piosenki użytkownika, musi on podczas logowania wyrazić zgodę na dostęp do tychże piosenek. Wymaga tego polityka Spotify. Gdy zgoda zostanie udzielona, program otrzymuje dostęp do najczęściej słuchanych utworów. Niestety nie do ich tekstu. Spotify bardzo często nie udostępnia słów piosenek, zatem w tym celu zostało użyte kolejne API: MusixMatch. Pozwala ono na sczytanie 30% tekstu danej

[59] https://support.spotify.com/pl/using_spotify/getting_started/what-is-spotify/ (16.06.2020)
[60] https://www.spidersweb.pl/2020/02/spotify-wyniki-q4-2019.html (16.06.2020)

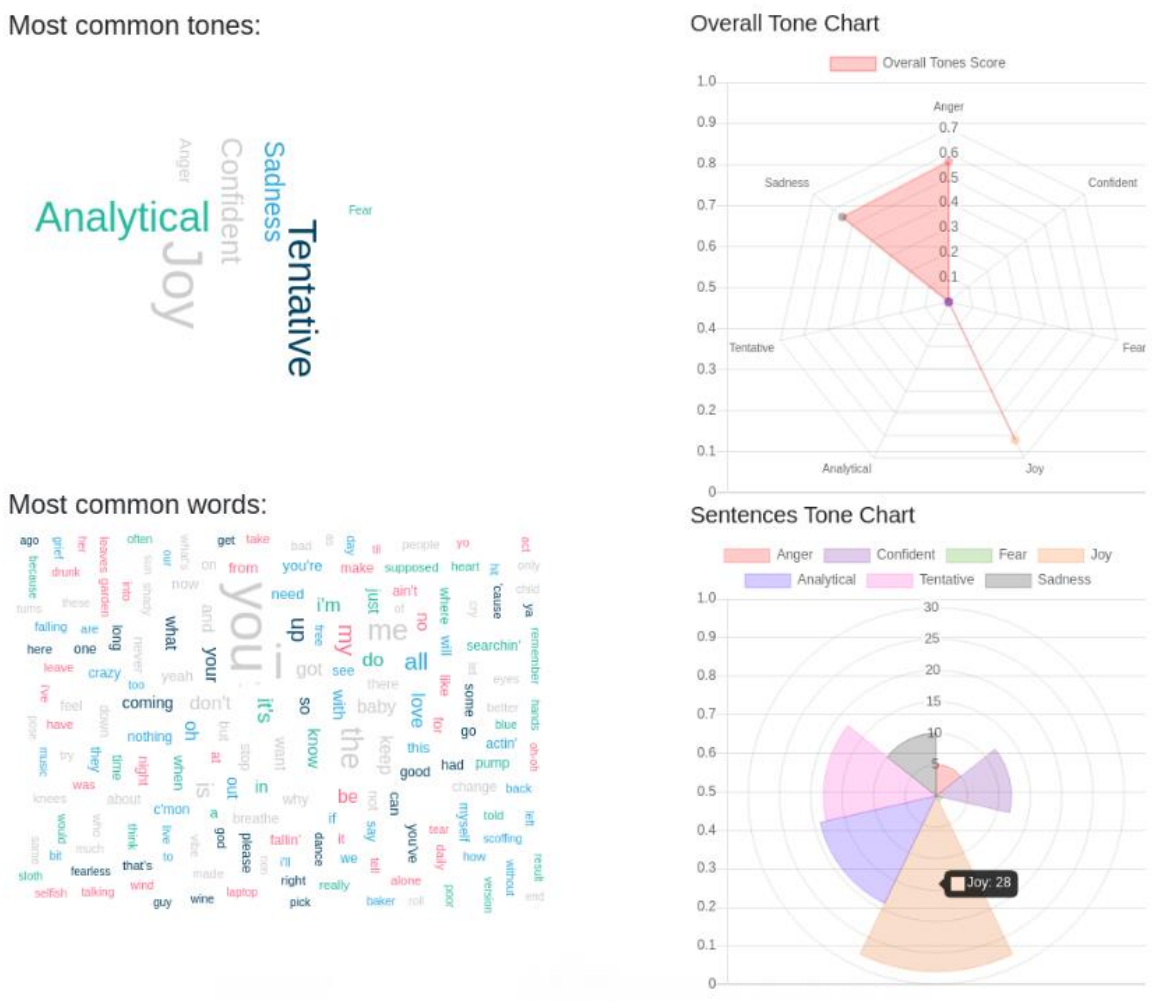
piosenki w wersji darmowej. Sczytane fragmenty tekstu zostały użyte w aplikacji. Można powiedzieć, że reszta tekstu to w dużej mierze powtarzający się refren, zatem nie jest to wielka strata. Zaletą API MusixMatch jest to, że posiada teksty do zdecydowanej większości piosenek. Jeśli znajdzie się wyjątek, piosenka taka musi zostać pominięta w analizie. W przypadku napisów w języku innym niż angielski, używany jest kolejny serwis superkomputera IBM Watson: Language Translator. Nie został on wcześniej opisany, ponieważ jego funkcją jest zwyczajnie tłumaczenie tekstu z danego języka na inny. Schemat bez szczegółów implementacyjnych wygląda więc następująco: Używając ogólnodostępnego API Spotify, sczytywane jest 50 ostatnio polubionych piosenek zalogowanego użytkownika. Jest to maksymalna możliwa ilość. Następnie piosenka po piosence wysyłane są zapytania do API MusixMatch, które to zwraca ich teksty. Jeśli są one w innym języku niż angielski, używany jest z kolei serwis IBM Watson Language Translator, który tłumaczy teksty piosenek na język angielski. W końcu tekst w języku angielskim poddawany jest usłudze Tone Analyzer. Wraz z końcem całego procesu, na ekranie aplikacji pokaże się efekt końcowy. Przykładowy wynik zaprezentowany zaostał na rysunku 14.



Rysunek 14
Strona główna aplikacji prezentującej usługę Tone Analyzer
Źródło: Opracowanie własne

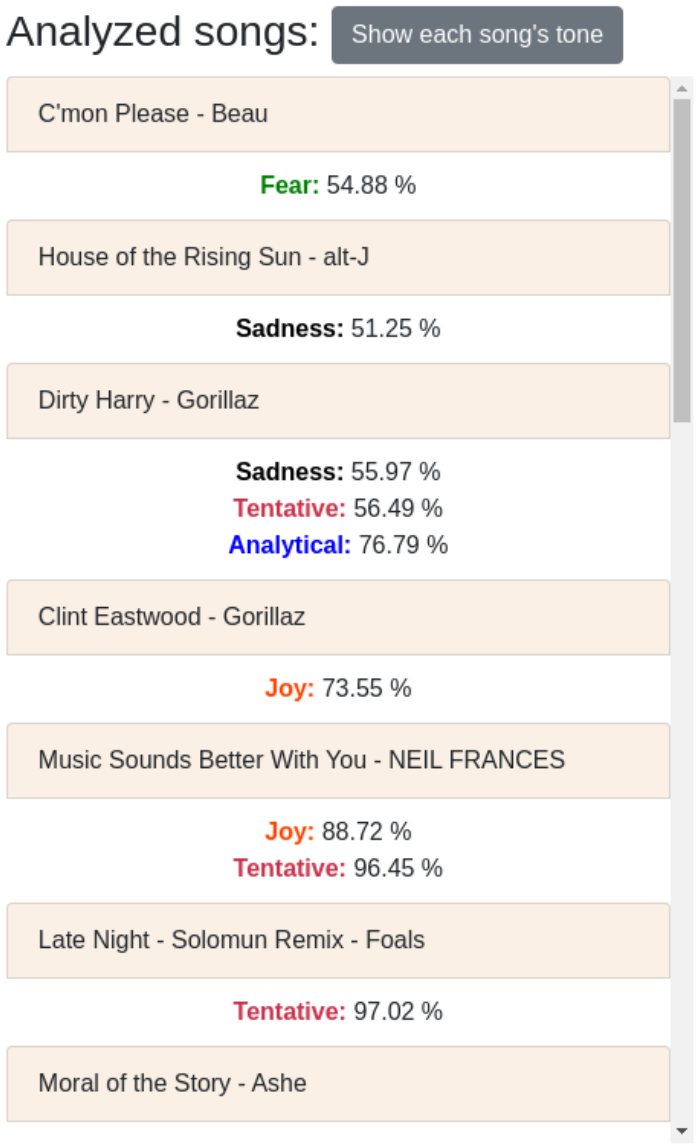
JSON zwracany przez usługę Tone Analyzer dzieli się na dwie części: „document_tone", która zawiera wykryte tony w całym tekście, oraz „sentences_tone". Ta druga część analizuje zdanie po zdaniu. Każde z nich jest poddawane analizie z osobna. Na stronie głównej aplikacji można zobaczyć, że oprócz listy piosenek, strona podzielona jest na cztery części. Obrazują one w inny sposób wyniki analizy. „Most common tones" przedstawia w formie chmury słów (ang. *wordcloud*) najczęściej występujące emocje w analizowanych

piosenkach. Korzysta się tu z drugiej części JSONa, czyli zliczana zostaje ilość zdań, w których wystąpiła dana emocja. Im słowo na wykresie jest większe, tym emocja ta wystąpiła większą ilość razy. „Sentences Tone Chart" to w zasadzie to samo, z tym, że dane przedstawione zostały na wykresie kołowym. Na rysunku 15 został umieszczony zrzut ekranu aplikacji, który utworzony został po najechaniu kursorem na pomarańczową część wykresu. Widać, że jest to emocja „Joy", czyli radość. Występuje ona w zdaniach aż 28 razy. Tak samo na chmurze słów wyraz Joy jest największy. Oznacza to, że ostatnie 50 polubionych przez użytkownika piosenek ma w sobie więcej radości niż innych emocji. „Most common words" to także chmura słów. Zliczane są tutaj nie najczęściej występujące tony, lecz najczęściej występujące wyrazy. „Overall Tone Chart" korzysta natomiast z pierwszej części JSONa, czyli ogólnej analizy całego tekstu. Wykryte zostały trzy główne emocje: Anger (złość), Sadness (smutek) oraz Joy (radość).



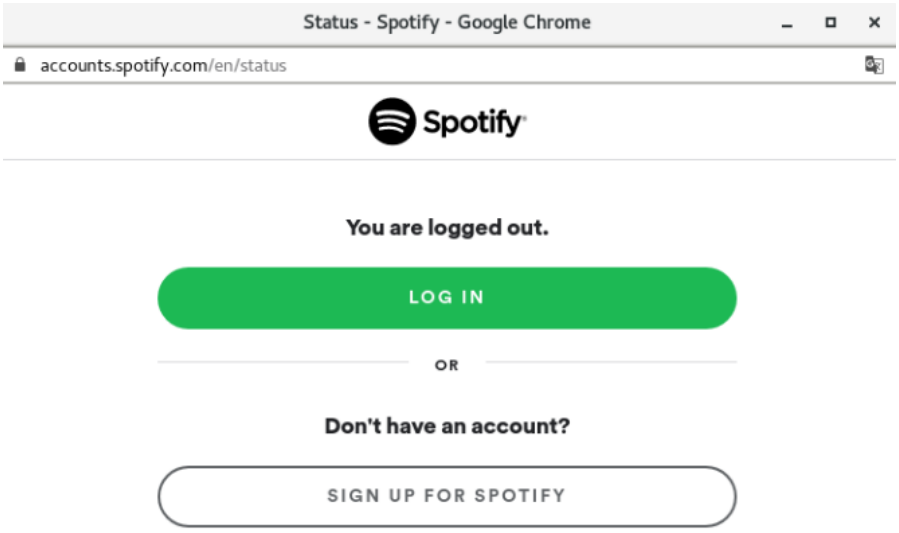
Rysunek 15
Część głównej strony która przedstawia efekty analizy Tone Analyzer na cztery różne sposoby
Źródło: Opracowanie własne

Na stronie głównej po lewej stronie znajduje się lista przeanalizowanych piosenek, z możliwością scrollowania. Piosenki na liście składają się z tytułu oraz nazwy wykonawcy. Analiza ogólna to jedno, natomiast dobrze by było, gdyby dało się zobaczyć nacechowanie emocjonalne każdej piosenki z osobna. Taką możliwość dostarcza przycisk „Show each song's tone". Dodatkowo, każda emocja ma swój kolor. Można to zobaczyć na rysunku 16.



Rysunek 16
Nacechowanie emocjonalne każdej z analizowanych piosenek z osobna w aplikacji demonstrującej działanie Tone Analyzer
Źródło: Opracowanie własne

Po zapoznaniu się ze wszystkim, gdy użytkownik kliknie przycisk znajdujący się w prawym górnym roku o nazwie „Log out", zostanie wylogowany. Wyświetli się komunikat wygenerowany przez Spotify API, pokazany na rysunku 17.



Rysunek 17
Komunikat o wylogowaniu wygenerowany przez Spotify API. Źródło: Opracowanie własne

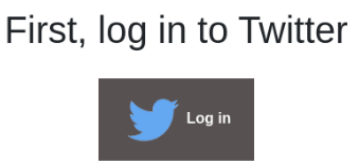
Najbardziej istotne fragmenty kodu z omawianej aplikacji zaprezentowane zostaną w podrozdziałach zaczynających się od napisu „Opis implementacji".

3.2 Opis aplikacji demonstrującej funkcję Personality Insights

Druga aplikacja demonstrująca usługę Personality Insights jest zrobiona w analogiczny sposób, z tą różnicą, iż używa innego serwisu. Do tego celu wykorzystane zostały dane pochodzące z Twittera a nie Spotify. Budowana aplikacja ma trochę inne zastosowanie. Dla nieobeznanych z tematem, Twitter to „serwis społecznościowy udostępniający usługę mikroblogowania założony 21 marca 2006 roku".[61] Jest to jeden z najpopularniejszych serwisów tego typu, popularny wśród elit oraz bardzo chętnie używany poprzez osoby publiczne, nawet polityków. Tweety są krótkimi (max. 280 znaków) wiadomościami, które widnieją na profilu ich autora, i to one są źródłem tekstu do analizy w przypadku aplikacji demonstrującej działanie usługi Personality Insights. Może je pisać każdy zarejestrowany użytkownik. Analiza więc nie będzie obejmowała piosenek, jak w poprzednim przypadku, lecz tym razem rozważane są konkretne osoby. Oczywiście tweety są publiczne, można się więc domyślić, że większość osób pisze tak, jak chce, aby to wyglądało. Niestety (choć jest to zasadna reguła) Twitter API, jak i inne najpopularniejsze serwisy tego typu, nie pozwalają korzystać z tekstu prywatnych wiadomości w zewnętrznych aplikacjach. Dlatego też do analizy

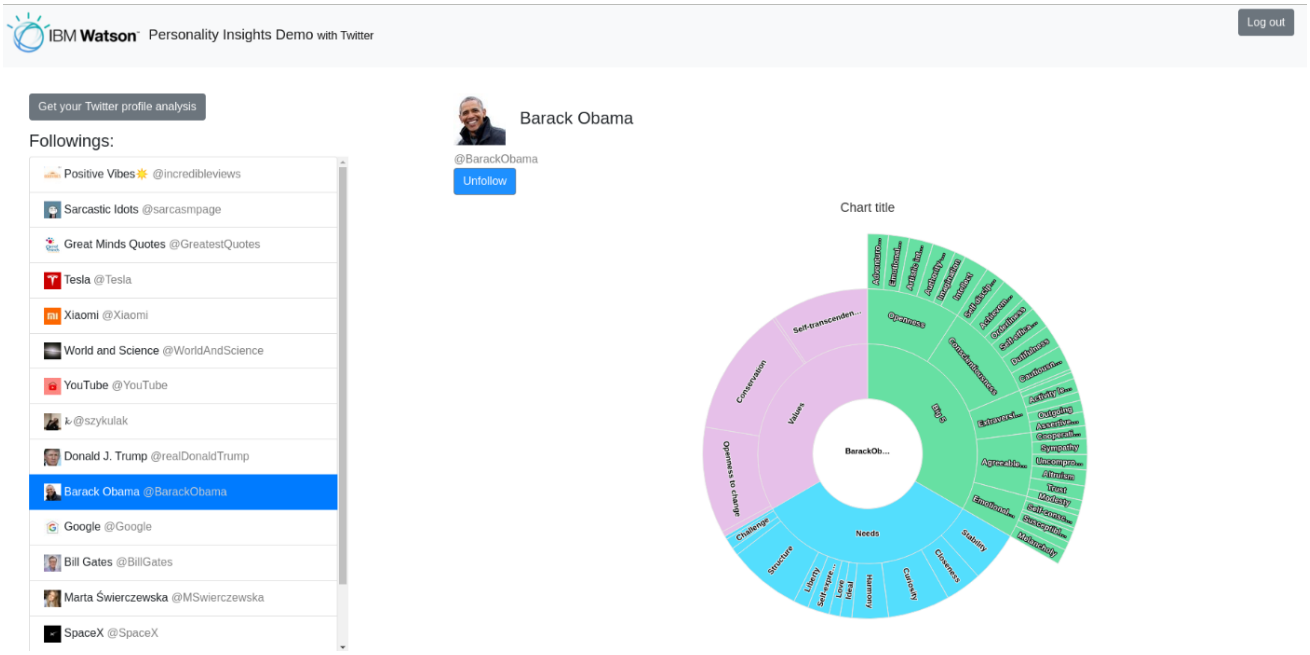
[61] https://pl.wikipedia.org/wiki/Twitter (21.06.2020)

podane zostaną posty publiczne znajomych zalogowanego użytkownika. Aby skorzystać ze stworzonej aplikacji, należy najpierw zalogować się do niej poprzez Twittera. Proces logowania wygląda podobnie jak w przypadku logowania się przez Spotify i został pokazany na rysunku 18.



Rysunek 18
Ekran widoczny na stronie logowania w aplikacji demonstrującej działanie Personality Insights
Źródło: Opracowanie własne

Po wyrażeniu zgody na dostęp do listy obserwowanych osób oraz postów publicznych, użytkownik zostaje przekierowany na stronę główną. Po lewej stronie znajduje się lista tych osób, a po prawej, zanim zostanie kliknięta konkretna osoba na liście, widnieje napis: *„Click on chosen person in the list and find out something about your friends' personality, or click on „Get your Twitter profile analysis" to check your personality based on your tweets!"* Ów tekst przedstawia co zrobić, aby skorzystać z aplikacji. Tłumacząc na j. polski: *Kliknij na przycisk „Get Profile" i dowiedz się czegoś o osobowości twoich znajomych, lub kliknij na „Get your Twitter profile analysis" aby sprawdzić Twoją osobowość na podstawie Twoich tweetów!* Zgodnie z zaleceniem, po wybraniu konkretnej osoby można zobaczyć w nietypowym wydaniu analizę postów publicznych w tym przypadku Baracka Obamy, a tym samym jego profilu osobowości. Napis, który był właśnie opisywany, wyświetlany jest w miejscu wykresu, zanim pierwszy raz użytkownik kliknie dowolną osobę z listy. Wybrana osoba na liście podświetla się na kolor niebieski. Zanim pokazany na rysunku 19 wykres się ukaże, należy chwilę poczekać, ponieważ w tym czasie pod spodem aplikacji dzieją się takie rzeczy, jak: pobieranie tweetów, sklejanie ich w jeden tekst, usuwanie niepotrzebnych kawałków tekstu oraz wiele innych podobnych procesów. Następnie gotowy tekst dostarczany jest serwisowi Personality Insights, który po chwili zwraca plik JSON z hierarchicznym układem cech osobowości wraz z procentem, który określa nasilenie danej cechy. Dane z pliku są parsowane aby wyświetlić je w przystępnej, widocznej na rysunku 19 formie.



Rysunek 19
Strona główna aplikacji po kliknięciu na wybraną osobę z listy
Źródło: Opracowanie własne

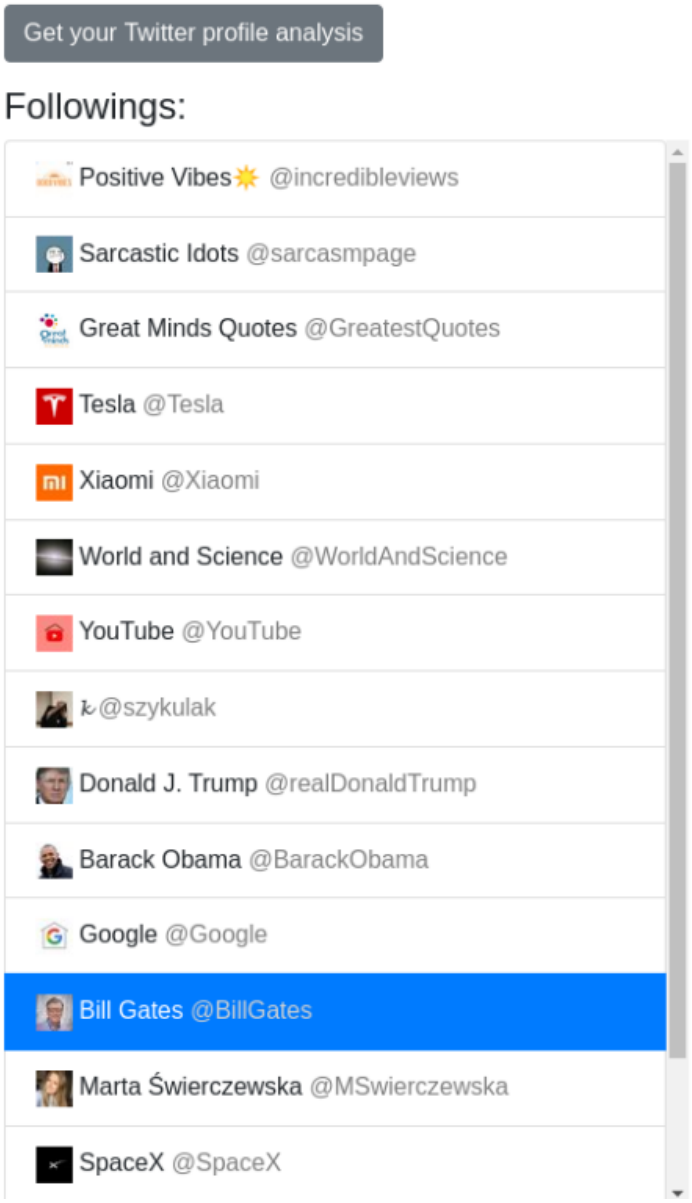
Ten niewątpliwie ciekawy typ wykresu nazywa się „Sunburst Chart". Jest on idealny w przypadku gdy zachodzi potrzeba wyświetlenia danych hierarchicznych. Framework Angular posiada gotowe rozwiązania, czyli mnóstwo interesujących typów wykresów do przedstawiania danych w różnej formie. Wystarczy przekazać je według odpowiednio wcześniej zdefiniowanego szablonu, a powstanie spektakularny efekt. Na samym środku, w białym kole wyświetlone zostaje imię i nazwisko analizowanej osoby. Naokoło, według schematu pliku, który zwraca serwis Personality Insights, w pierwszej kolejności można zauważyć podział na 3 główne kategorie. „Needs" czyli potrzeby, „Values" – wartości, oraz „Big 5" – Wielka Piątka. Te trzy podziały, a w szczególności Wielka Piątka, zostały opisane w rozdziale pierwszym przy opisie funkcji Personality Insights. Dodatkowo, pod zdjęciem profilowym analizowanej osoby widnieje przycisk „Unfollow". Jest to dodatkowa opcja odobserwowania użytkownika, gdy tylko zajdzie taka potrzeba. Na rysunku 20 zaprezentowana została analiza tweetów Billa Gatesa.

Rysunek 20
Wykres typu Sunburst powstały z wyników analizy serwisu Personality Insights danych które pochodzą z publicznych tweetów Billa Gatesa
Źródło: Opracowanie własne

Wartości i potrzeby mają tylko jeden poziom pod sobą, natomiast Wielka Piątka jest jak widać bardziej skomplikowana. Po najechaniu kursorem na interesującą użytkownika część wykresu, wyświetlona zostanie pełna nazwa cechy osobowości wraz z natężeniem, w którym ta występuje. Pokazane jest to na rysunku 20. Maksymalna wartość natężenia to jeden. Dodatkowo, im wartość natężenia jest większa, tym dany obszar na wykresie jest szerszy. Pozwala to wizualnie już na pierwszy rzut oka zorientować się w analizie, bez potrzeby krążenia kursorem po wartościach. Na przykład, już po pierwszych pięciu sekundach porównywania dwóch wykresów można stwierdzić, że według przeprowadzonej analizy Watsona Bill Gates jest bardziej ciekawy świata od Baracka Obamy, ponieważ pole „Curiosity" wyświetlane w kolorze niebieskim jako część potrzeb, jest znacznie szersze w przypadku Gates'a. W przypadku, gdy użytkownik Twittera nie publikuje zbyt wiele postów, serwis Watsona nie ma danych które mógłby analizować. Wtedy na miejscu wykresu pojawia się komunikat pokazany na rysunku 21.

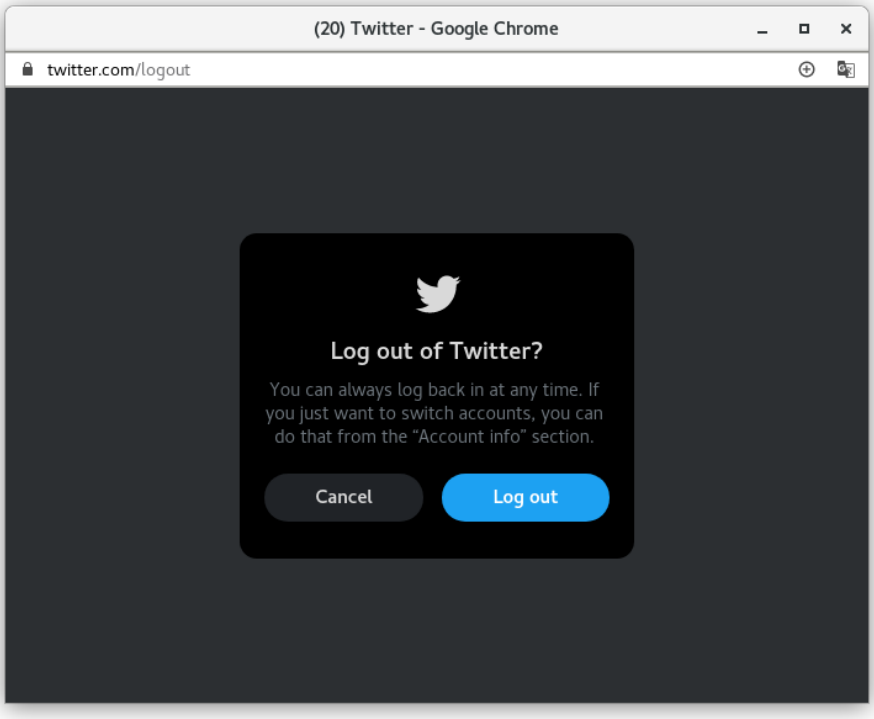
Rysunek 21
Komunikat występujący w przypadku gdy jest za mało danych do analizy
Źródło: Opracowanie własne

Przyglądając się lewej części ekranu można zauważyć listę osób, które obserwuje zalogowany użytkownik. Składa się ona ze zdjęcia profilowego, imienia i nazwiska, oraz unikalnej nazwy profilu po znaku „@", którą dany użytkownik sobie dobiera. Zostało to przedstawione na rysunku 22.



Rysunek 22
Lista osób obserwowanych na Twitterze przez zalogowanego w aplikacji użytkownika
Źródło: Opracowanie własne

Tak jak w aplikacji prezentującej działanie usługi Tone Analyzer, tak i w tym przypadku w prawym górnym rogu znajduje się przycisk „Log out”, który po naciśnięciu przekieruje użytkownika do wygenerowanego przez Twitter API okienka, aby upewnić się, czy aby na pewno ma wystąpić wylogowanie. Potwierdzenie wylogowania zostało zaprezentowane na rysunku 23.



Rysunek 23
Okno wyskakujące po naciśnięciu przycisku "Log out"
Źródło: Opracowanie własne

Po ponownym naciśnięciu „Log out” przez użytkownika, zostanie on przekierowany na pokazaną wcześniej stronę logowania aplikacji.

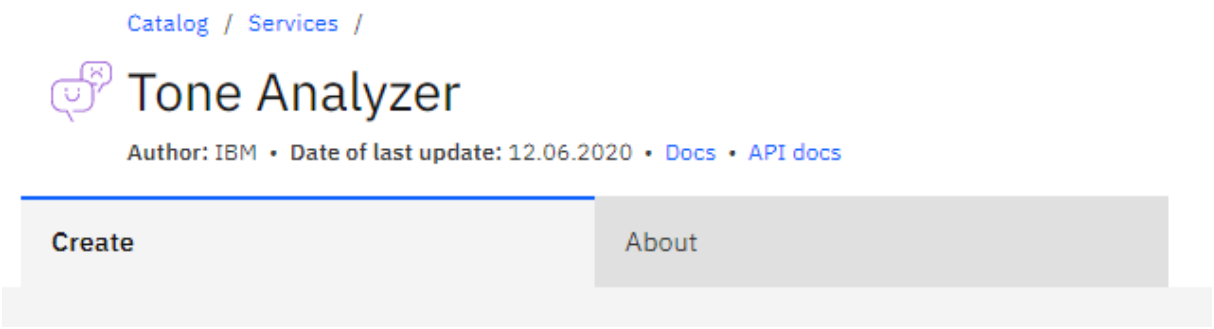
Najbardziej istotne fragmenty kodu z tej aplikacji zaprezentowane zostaną w podrozdziałach zaczynających się od napisu „Opis implementacji”.

3.3 Opis implementacji

3.3.1 Część wspólna

Obydwa opisane w poprzednim podrozdziale projekty używają do zadań po stronie serwera Javy 8 oraz frameworka Spring, natomiast część aplikacji po stronie klienta utworzono za pomocą języka TypeScript i frameworka Angular.

W celu rozpoczęcia pracy nad projektami zaczęto od stworzenia projektu mavenowego, korzystającego ze Spring Boot'a. Struktura plików w takim projekcie jest nieprzypadkowa i zawiera plik pom.xml, który został opisany w rozdziale drugim. Do tego pliku należy dołączyć odpowiednie fragmenty, które pozwalają aplikacji korzystać z zewnętrznych implementacji. Na samym początku między innymi warto dodać dependencję (zależność) wybranej usługi. Kluczową kwestią podczas implementacji połączenia z serwisem Watsona jest posiłkowanie się dokumentacją, którą można znaleźć po wejściu na stronę https://cloud.ibm.com/. Następnie należy kliknąć zakładkę „Catalog", a po niej „Services". Tam do wyboru znajdują się dziedziny, np. AI/Machine Learning, w której są opisywane usługi. Po kliknięciu w wybrany serwis, np. Tone Analyzer, należy utworzyć instancję owego serwisu posiłkując się wskazówkami. Te dostępne są pod linkami „Docs" oraz „API Docs", po prawej stronie od nazwy usługi. Po kliknięciu przycisku „Create" zostanie utworzony serwis Tone Analyzer, do którego wygeneruje się unikatowy klucz „API KEY" oraz adres URL. Są to dane, które konieczne są do utworzenia oraz późniejszego używania instancji serwisu w kodzie aplikacji. Można uzyskać do nich dostęp za każdym razem wchodząc na swoje konto na stronie cloud.ibm.com, klikając na utworzony serwis (porównaj rys.24).



Rysunek 24
Miejsce w które należy kliknąć aby uzyskać informacje dotyczące usługi i tego, jak zaimplementować połączenie z nią.[62]

Po kliknięciu w "API docs" po prawej stronie znajduje się najbardziej przydatna opcja, czyli kawałki kodu w wybranym języku, które pomagają zrozumieć użytkownikowi czytaną dokumentację w praktyce. To, jak mniej więcej powinno wyglądać zaimplementowanie wybranych fragmentów usługi. Stamtąd można skopiować zależność danego serwisu Watsona do pliku pom.xml, która pobierze automatycznie wszystkie potrzebne biblioteki. Na załączonym na rysunku 25 zrzucie ekranu widać potrzebną dependencję. Jak wskazuje komentarz oraz „artifactId", jest ona wspomnianą zależnością do klas IBM Watson.

[62] https://cloud.ibm.com/catalog/services/tone-analyzer (05.07.2020)

```
<!-- watson -->
<dependency>
    <groupId>com.ibm.watson</groupId>
    <artifactId>ibm-watson</artifactId>
    <version>8.1.0</version>
</dependency>
```

Rysunek 25
Dwie pierwsze zależności zamieszczone do pliku pom.xml, druga odpowiada za biblioteki IBM Watson
Źródło: Opracowanie własne

Taką samą zależność dodano w obydwóch aplikacjach. Teraz już pisząc kod można korzystać z klas jakie udostępniają funkcje opisywane w pierwszym rozdziale. Utworzenie instancji usługi Tone Analyzer wymaga wygenerowanych wcześniej API KEY oraz adresu URL serwisu, co prezentuje rysunek 26 zawierający stosowny zrzut ekranu.

```
@Component
public class ToneAnalyzerService {

    private static String KEY = TemporaryData.readExactKeyFromJson("/home/anna/Documents/credentials.json", "tone analyzer");
    private static String VERSION = "2017-09-21";
    private static String URL = "https://api.eu-gb.tone-analyzer.watson.cloud.ibm.com/instances/d10766b4-a61d-4b99-a759-97be4
    private static ToneAnalyzer toneAnalyzerInstance;

    private ToneAnalyzerService() {}

    public static ToneAnalyzer getInstance() { //singleton pattern
        if(toneAnalyzerInstance == null) {
            Authenticator authenticator = new IamAuthenticator(KEY);
            toneAnalyzerInstance = new ToneAnalyzer(VERSION, authenticator);
            toneAnalyzerInstance.setServiceUrl(URL);
        }
        return toneAnalyzerInstance;
    }

    public static ToneAnalysis analyzeToneFromText(ToneAnalyzer service, String text) {
        ToneOptions toneOptions = new ToneOptions.Builder()
                .text(text)
                .build();
        ToneAnalysis tone = service.tone(toneOptions).execute().getResult();
        return tone;
    }
```

Rysunek 26
Kod w środowisku Eclipse ukazujący połączenie z serwisem Tone Analyzer oraz metodę analyzeToneFromText() w języku Java
Źródło: Opracowanie własne

Jak można zauważyć, konstruktor klasy ToneAnalyzerService() jest prywatny, a utworzenie instancji serwisu odbywa się w metodzie getInstance(). Stworzy się ona tylko jeden raz, gdy na początku ma wartość null. Z każdym następnym wywołaniem tej metody, zwracany będzie już ten sam, utworzony wcześniej obiekt. Użyto tutaj bowiem wzorca Singleton, ponieważ nie ma potrzeby, aby obiektów ToneAnalyzer było więcej niż jeden. Tworząc obiekt IamAuthenticator należy podać jako parametr zmienną KEY, która odpowiada wspomnianemu kluczowi do autentykacji: API_KEY. Jest on pobierany z pliku, ponieważ nie powinno się udostępniać takich danych, zwłaszcza, gdy kod jest następnie udostępniany na zewnątrz, na przykład do

systemu kontroli wersji Git. Wtedy zmienną taką warto umieścić jako zmienną środowiskową. (ang. *environment variable*). W celu stworzenia już instancji ToneAnalyzer, podano ów obiekt autentykacji, wersję oraz adres url jako parametry.

Kolejną z ważniejszych części kodu jest metoda analyzeToneFromText(…). Podając jako parametr instancję serwisu oraz tekst, zostanie zwrócony obiekt ToneAnalysis, czyli gotowa już, wykonana za programistę analiza tekstu.

Analogicznie proces ten wygląda w przypadku usługi Personality Insights. Różnica jest jedynie w typie klas: Tworzona jest instancja klasy PersonalityInsights, zamiast ToneAnalyzer, oraz z analizy zwracany jest Profile, a nie ToneAnalysis, ponieważ w tym przypadku kluczowym rezultatem działania Watsona jest profil osobowości (porównaj rys. 27).

```
public class PersonalityInsightsService {

    private static String KEY = TemporaryDataService.readExactKeyFromJson("/home/anna/Documents/credentials.json", "personali
    private static String URL = "https://api.eu-gb.personality-insights.watson.cloud.ibm.com/instances/d6009cd8-87f5-49b8-b1
    private static String VERSION = "2018-05-01";
    private static PersonalityInsights personalityInsights;

    private PersonalityInsightsService() {}

    public static PersonalityInsights getInstance() {
        if(personalityInsights == null) {
            Authenticator authenticator = new IamAuthenticator(KEY);
            personalityInsights = new PersonalityInsights(VERSION, authenticator);
            personalityInsights.setServiceUrl(URL);
        }
        return personalityInsights;
    }

    public static Profile analizeFromText(PersonalityInsights service, String text){
        ProfileOptions options = new ProfileOptions.Builder()
                .text(text)
                .build();

        Profile resultAnalysis = service.profile(options).execute().getResult();
        return resultAnalysis;
    }
```

Rysunek 27
Kod w środowisku Eclipse ukazujący połączenie z serwisem Personality Insights oraz metodę analizeFromText() w języku Java
Źródło: Opracowanie własne

Ostatnią ważną cechą wspólną pomiędzy dwoma aplikacjami jest ich metoda main(..), której postać zawdzięcza się frameworkowi Spring Boot. Omawiana w rozdziale drugim adnotacja „@SpringBootApplication" jest widoczna nad nazwą klasy i za programistę zajmuje się konfiguracją projektu. Druga adnotacja „@ComponentScan" jak nazwa wskazuje, skanuje wybrane pakiety w poszukiwaniu komponentów Springa. Gdy zauważy klasę oznaczoną znakiem „@Component" (możliwe są też inne, np. Service), uznaje ją jako „ziarno" (ang. b*ean*), co sprawia, że użytkownik później nie musi przejmować się tworzeniem obiektów tych klas, a wystarczy, że używając ich w innej klasie doda nad nimi adnotację „@Autowired". Nazywa się to wstrzykiwaniem zależności (porównaj rys. 28).

```java
@SpringBootApplication
@ComponentScan(basePackages = {"controller", "model", "service"})
public class WatsonToneAnalyzerDemoApplication {

    public static void main(String[] args) throws SpotifyWebApiException, IOException {
        SpringApplication.run(WatsonToneAnalyzerDemoApplication.class, args);
    }
}
```

Rysunek 28
Metoda main(...) w konfiguracyjnej klasie projektu używającego frameworka Spring Boot
Źródło: Opracowanie własne

3.3.2 Personality Insights

Wynikowe obiekty klas Profile oraz ToneAnalyzer można używać w dowolny sposób, wyciągając z nich jedynie interesujące użytkownika informacje. Na przykładzie aplikacji demonstrującej Personality Insights, utworzono w tym celu specjalną klasę PIProfile (skrót od Personality Insights Profile), która posiada jedynie najważniejsze pola, takie jak zachowanie, potrzeby i wartości. Dostęp do nich jest prywatny. Dostęp do zmiennych odbywa się poprzez publiczne gettery oraz setery (porównaj rys. 29).

```java
public class PIProfile {

    public PIProfile(Map<String, Double> behavior, List<PersonalityTrait> personality, Map<String, Double> needs,
            List<ConsumptionPreferencesCategory> consumptionPreferences, String wordCountMessage,
            Map<String, Double> values) {

        this.behavior = behavior;
        this.personality = personality;
        this.needs = needs;
        this.consumptionPreferences = consumptionPreferences;
        this.wordCountMessage = wordCountMessage;
        this.values = values;
    }

    private Map<String, Double> behavior;
    private List<PersonalityTrait> personality;
    private Map<String, Double> needs;
    private List<ConsumptionPreferencesCategory> consumptionPreferences;
    private String wordCountMessage;
    private Map<String, Double> values;


    public Map<String, Double> getBehavior() {
        return behavior;
    }
    public void setBehavior(Map<String, Double> behavior) {
        this.behavior = behavior;
    }
```

Rysunek 29
Klasa PIProfile, reprezentująca najważniejsze cechy profilu osobowości, które następnie przesyłane są na stronę klienta aby być zaprezentowane jako wykres typu Sunburst Chart
Źródło: Opracowanie własne

Gdy użytkownik korzystający z aplikacji kliknie na konkretną osobę, która go interesuje, zostanie wysłane zapytanie GET do serwisu na adres „/profile/chosen”, z parametrem „screenName”, który mówi o tym jakiego użytkownika tweety wyszukać. W przypadku kliknięcia na przycisk „Get your profile analysis” również będzie to zapytanie GET, jednak pod adres „/profile/default”, w celu wyszukania tweetów zalogowanego użytkownika. Metoda getProfile(…), do której kontroler kieruje wykonie zapytania, w zależności od wartości parametru screenName (null lub konkretna wartość) wywoła odpowiednią akcję. Będzie nią pobranie tweetów odpowiedniej osoby. Zapytania te kontroluje kontroler „PersonalityInsightsController”. Zajmuje się on odbieraniem zapytań HTTP, a następnie przekierowywaniem ich do odpowiednich metod znajdujących się w serwisach. Po wykonaniu odpowiednich działań kontroler zwraca wynik z powrotem do części aplikacji po stronie klienta, owinięty w obiekt „ResponseEntity”. Pozwala on bowiem na zwrócenie również statusu HTTP. Zrzut ekranu prezentujący fragment kodu odpowiadający za omawiane funkcjonalności zawarty został na rysunku 30.

```
@RestController
@RequestMapping("/")
@CrossOrigin("*")
public class PersonalityInsightsController {

    @Autowired
    PIProfileService piProfileService;

    @GetMapping("/profile/chosen")
    public ResponseEntity<PIProfile> getProfile(@RequestParam String screenName) {
        return new ResponseEntity<PIProfile>(piProfileService.getProfile(screenName), HttpStatus.OK);
    }

    @GetMapping("/profile/default")
    public ResponseEntity<PIProfile> getProfile() {
        return new ResponseEntity<PIProfile>(piProfileService.getProfile(null), HttpStatus.OK);
    }
}
```

Rysunek 30
Kontroler Personality Insights odbierający zapytania HTTP i przekierowujący działanie do metody getProfile(…), który jako odpowiedź serwera zwraca obiekt klasy PIProfile
Źródło: Opracowanie własne

Po pobraniu wpisów konkretnego użytkownika (parametr screenName) z Twittera, są one przetwarzane przez metodę parsującą. Usuwa ona z tekstu wszystkie niepotrzebne znaki, cytaty innych użytkowników oraz emotikony. Idealnym narzędziem w tym celu są wyrażenia Regex, czyli specjalne ułożenia znaków, nadające im znaczenie. Metoda parsująca została pokazana na rysunku 31.

```
public String cutOffEveryNotWantedRegex(String tweets) {

    String markRegex = "@[/a-zA-Z0-9._-]+";
    String httpRegex = "(https:[/a-zA-Z0-9.-_]+)";

    String pureText = tweets
            .replaceAll(markRegex, " ")
            .replaceAll(httpRegex, " ");

    return pureText;
}
```

Rysunek 31
Metoda cutOffEveryNotWantedRegex(String tweets), używająca wyrażeń Regex, które pozwalają sparsować tekst tweetów do tekstu bez znaków specjalnych oraz cytatów innych użytkowników.
Źródło: Opracowanie własne

Istotną kwestią jest również ilość słów do analizy. Na poniższym zrzucie ekranu można zauważyć, że jeżeli ich ilość jest mniejsza niż 100, profil osobowościowy nie zostanie ustawiony. Dzieje się tak, ponieważ serwis Watsona potrzebuje minimum tyle słów, aby zadziałać poprawnie oraz z sensem. Gdy ilość słów jest odpowiednia, serwis Language Translator sprawdza, czy tekst jest w języku angielskim. Jeśli nie, przetłumaczy go (wywołanie funkcji translateTweetsIfNotEnglish(…)), a rezultat poda jako parametr do omawianej wcześniej metody analizeFromText(…). Ta po wykonaniu przypisze Watsonowy obiekt, czyli rezultat analizy do zmiennej „profile”. Zrzut ekranu zawierający ten fragment kodu pokazano na rysunku 32.

```
public boolean setProfile(String screenName) {
    String text = twitterModificationService.getChosenUserPureText(screenName);

    if(text.split(" ").length < 100) { //jesli tweety razem maja mniej niz 100 slow, to ni
        return false;
    }

    String translatedText = LanguageTranslatorService.translateTweetsIfNotEnglish(text);

    profile = PersonalityInsightsService.analizeFromText(
            PersonalityInsightsService.getInstance(), translatedText);
    return true;
}
```

Rysunek 32
Metoda setProfile(String screenName)
Źródło: Opracowanie własne

Jednak jak już wspomniano, rezultat analizy zapisany w zmiennej „profile” zawiera bardzo dużo informacji, które nie są potrzebne aplikacji. W związku z tym, następnym krokiem jest pobranie z obiektu jedynie niezbędnych wyników. Odbywa się to za pomocą metod pobierających odpowiednie cechy osobowości, np. getValues(), getPersonality(), getNeeds(). Pierwsza z nich została pokazana na rysunku 34. Na obiekcie wynikowym analizy, który ma

nazwę „profile” wywołana została funkcja getValues(), należąca już do usługi Personality Insights. Następnie do mapy przypisano jako klucz nazwę cechy, a jako wartość jej natężenie.

```
public Map<String, Double> getValues(){
    try {
        List<Trait> valuesList = profile.getValues();
        Map<String, Double> valuesMap = new HashMap<>();
        for(Trait trait : valuesList) {
            valuesMap.put(trait.getName(), trait.getPercentile());
        }
        return valuesMap;

    }catch(NullPointerException e) {
        System.err.println("Profile was never set");
        e.printStackTrace();
    }
    return null;
}
```

Rysunek 33
Metoda getValues(), pobierająca wartości z obiektu wynikowego analizy o nazwie "profile".
Źródło: Opracowanie własne

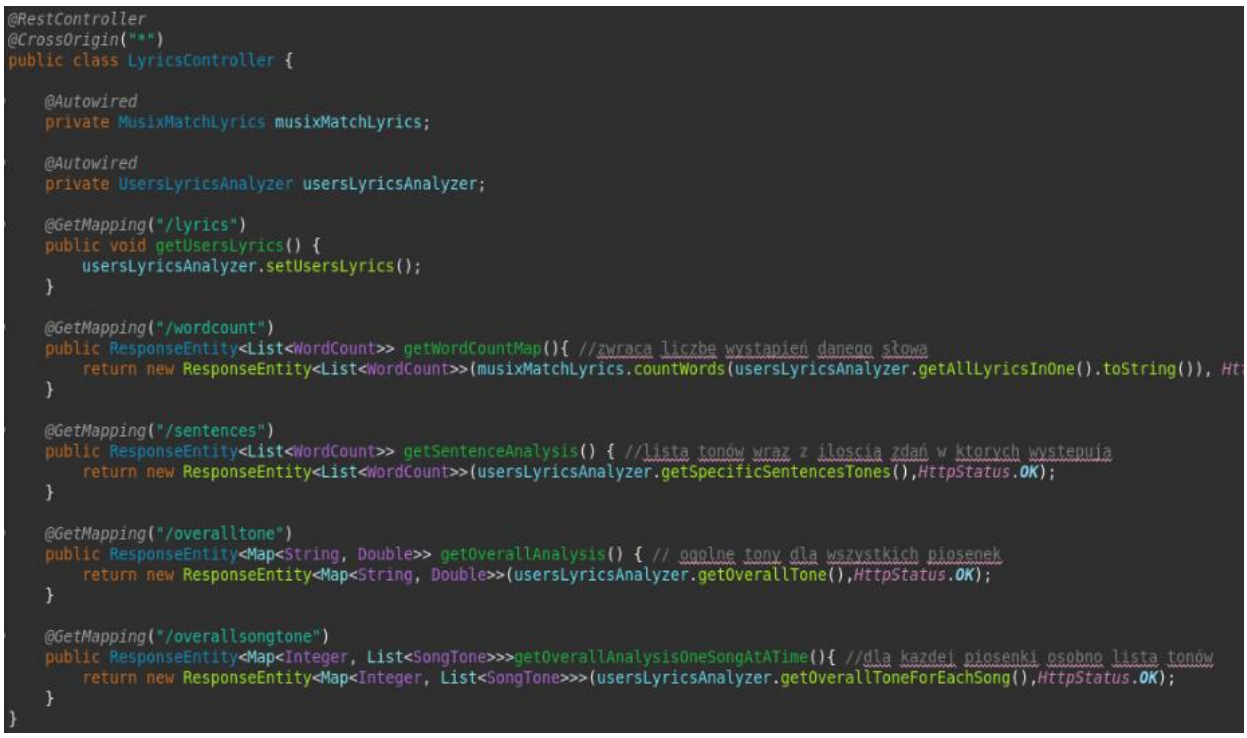
Wyniki te następnie przypisywane są do obiektu klasy PIProfile, który zwracany jest na frontendową część aplikacji, a następnie przedstawiany w formie wykresu Sunburst Chart. Instrukcję do jego stworzenia można znaleźć na stronie https://www.highcharts.com. Znajdują się tam również wskazówki dotyczące wykresów innego typu. W aplikacji po części klienta można zobaczyć metodę createChartData() (porównaj rys. 33), która zwraca obiekt zawierający dane potrzebne do prawidłowego wyświetlenia wykresu. Składa się on z kolejno: ustawień dotyczących nazw pól oraz ich kolorów (general), cech osobowości: potrzeby (needs), wartości (values), osobowości (personality), oraz z rozłożonej na czynniki osobowości (personalityChildren). Ostatnia część może wydawać się myląca, jednak gdy spojrzy się na gotowy wykres, można zauważyć w części przedstawiającej Wielką Piątkę (kolor zielony), że zawiera ona cechy podzielone na jeszcze bardziej szczegółowe pola.

```
createChartData(){
let general = [{
        id: '0.0',
        parent: '',
        name: this._chosenFriend.screenName,
        value:1,
        color: 'White'
  }, {
        id: '1.3',
        parent: '0.0',
        name: 'Big 5',
        value:1,
        color: '#67E0A3'

  }, {
        id: '1.1',
        parent: '0.0',
        name: 'Needs',
        value:1,
        color: '#54DEFD'
  }, {
        id: '1.2',
        parent: '0.0',
        name: 'Values',
        value:1,
        color: '#E6C0E9'
  }];
console.log("General: ");
console.log(general);
 let needs = this.getNeeds();
 let values = this.getValues();
 let personality=this.getPersonality();
 let personalityChildren = this.personalityChildren;

 let chartData: Trait[] = [];
 chartData=[ ...general,...needs,...values,...personality,...personalityChildren];

 return chartData;
}
```

Rysunek 34
Kod napisany we frameworku Angular (frontend), przedstawiający ustawienia wykresu typu Sunburst Chart.
Źródło: Opracowanie własne

3.3.3 Tone Analyzer

Aplikacja demonstrująca działanie serwisu Tone Analyzer, zgodnie z tym, co zostało przedstawione na początku rozdziału trzeciego, dzieli wyniki analizy na cztery części. Patrząc na zrzut ekranu zaprezentowany na rysunku 15 i porównując spostrzeżenia z rysunkiem 34, odpowiednim nagłówkom odpowiadają konkretne metody w kodzie aplikacji.



Rysunek 35
Kontroler LyricsController, zajmujący się odbieraniem zapytań HTTP i przekierowywaniem ich do odpowiednich metod w klasach serwisowych.
Źródło: Opracowanie własne

Aby zrozumieć kolejne części kodu, należy w pierwszej kolejności należy wyjaśnić jaka metoda odpowiada za dany wykres. Idąc z lewej strony, na stronie głównej (rys. 14 i 15) znajdują się nagłówki kolejno:

- **Most common tones**: Odpowiada mu metoda getSentenceAnalysis() z rysunku 34. Kontroler odbiera zapytanie HTTP wysłane z części aplikacji po stronie klienta na adres „/sentences", oraz przekierowuje wykonanie do metody getSpecificSentencesTones(). Jak wskazuje komentarz, zwraca ona liczbę zdań dla danego tonu, co później użytkownik widzi w formie wykresu typu WordCloud (chmura słów). Im większy ton, tym więcej razy wystąpił.
- **Most common words**: Metoda getWordCountMap() przekierowuje wykonanie odpowiednich działań do funkcji countWords(), która zlicza wystąpienia danego słowa we wszystkich piosenkach. Dane przesyłane są następnie na część aplikacji po stronie klienta przedstawione w ten sam sposób, co poprzednio: chmura słów.
- **Overall tone chart**: Na wykresie przypominającym pajęczynę (Radar Chart) przedstawiony jest ogólny wynik analizy, czyli wszystkich piosenek razem. Te dane nie musiały być specjalnie parsowane w celu uzyskania interesujących treści, tak jak w poprzednim przypadku, ponieważ zwrócony przez serwis Tone Analyzer obiekt

zawierał bezpośredni do nich dostęp. Zwraca go metoda getOverallTone(), za pośrednictwem funkcji kontrolera getOverallAnalysis().

- **Sentences tone chart**: Ostatni prezentowany na stronie głównej wykres przedstawia dokładnie te same dane, co „Most common tones", jednak w inny sposób (Polar Chart), wyświetlając konkretne wartości po najechaniu na nie kursorem myszki.

Ostatnia, nieomówiona jeszcze metoda znajdująca się na rysunku 34 to getOverallAnalysisOneSongAtATime(). Ta długa nazwa oznacza, że w ramach tej funkcji analizowany jest tekst każdej piosenki z osobna. Na stronie głównej aplikacji rezultat wyświetlony jest na liście piosenek po lewej stronie, po kliknięciu na przycisk „Show each song's tone". Pod każdym utworem wyświetlana jest wtedy emocja oraz procent, w jakim występuje w piosence. Zależności oznaczone adnotacją „@Autowired" (wzorzec projektowy Dependency Injection) to klasa MusixMatchLyrics zajmująca się wszystkim co związane z API MusixMatch, czyli pobieraniem napisów piosenek i obsługą błędów. Druga zależność, klasa UsersLyricsAnalyzer zajmuje się przetwarzaniem obiektu analizy na wszystkie wypunktowane wcześniej sposoby i zwracaniem listy lub mapy wartości, w zależności od potrzeb.

Warto zobrazować przykładowe metody zajmujące się parsowaniem danych od środka. Na rysunku 35 znajduje się funkcja setUsersLyrics() klasy UsersLyricsAnalyzer, która zajmuje się przygotowaniem napisów piosenek pod późniejsze ich wykorzystanie. W pętli for następuje iteracja po każdej z 50 sczytanych piosenek z API Spotify. Dla danej piosenki osobno wysyłane są zapytania do API MusixMatch, aby pobrać napisy utworu. Następnie, jeśli mają one wartość „null" lub są puste, pętla wykonuje kolejną iteracje. W przeciwnym przypadku, napisy przypisywane są do dwóch zmiennych: lyricsMap oraz allLyrics. Mapa napisów (lyricsMap), której kluczem jest ID piosenki a wartością jej treść, jest konieczna do wyświetlenia analizy każdej piosenki z osobna (efekt na rysunku 16). Natomiast do zmiennej allLyrics, za pomocą klasy StringBuilder dodawane są kolejno napisy każdej z piosenek, aby stworzyć z nich jeden tekst.

```
public synchronized void setUsersLyrics() {

    StringBuilder allLyrics = new StringBuilder(); // all lyrics in one
    Map<Integer, String> lyricsMap = new HashMap<>(); //lyrics separately for each  song
    List<Song> userSongs = spotifyApiClient.getUserLibrary(); //get songs

        for (Song s : userSongs) {
            String lyrics = musixMatchLyrics.getLyricsOfASong(s.getTitle(), s.getArtist());
            if(lyrics == null || lyrics == "") {
                continue;
            }
            lyricsMap.put(s.getId(), lyrics);
            allLyrics.append(lyrics);
        }
    this.lyricsOneAtATime = lyricsMap;
    this.allLyricsInOne = allLyrics;
}
```

Rysunek 36
Metoda setUsersLyrics(), która przupisuje napisy piosenek do dwóch zmiennych w różnych formach: tekstu jako całość, oraz napisy każdej piosenki z osobna jako mapy o wartościach <ID piosenki, napisy>
Źródło: Opracowanie własne

Na przykładzie napisów piosenek złożonych w całość, przedstawiona zostanie metoda analyzeLyricsTone(). Zrzut ekranu wyświetlony na rysunku 36 podaje złożone w całość napisy piosenek do usługi Tone Analyzer aby otrzymać ich analizę. Metoda analyzeToneFromText() została przedstawiona na rysunku 26.

```
public ToneAnalysis analyzeLyricsTone() {

    StringBuilder lyrics = this.allLyricsInOne;
    ToneAnalysis tone = ToneAnalyzerService.analyzeToneFromText(ToneAnalyzerService.getInstance(), lyrics.toString());
    return tone;
}
```

Rysunek 37
Metoda analyzeLyricsTone() która podaje tekst w całości do metody analyzeToneFromText(...) i zwraca obiekt ToneAnalysis
Źródło: Opracowanie własne

Następnie, aby dane wysłać na frontendową część aplikacji w przystępnej formie, można wyselekcjonować jedynie interesujące użytkownika części analizy. Na przykład funkcja „getSpecificSentencesTones()" za pomocą metody „getSentencesTone()" pobiera z obiektu analizy jedynie jego drugą część, czyli analizę każdego zdania z osobna. W następnej kolejności, aby znać najczęściej występujące tony, zlicza je za pomocą pętli for. Zaprezentowane zostało to jest to na rysunku 37. W tym celu wykorzystane zostały lambda wyrażenia, dostępne od Javy 8.

```
public List<WordCount> getSpecificSentencesTones() { // ile jest zdań o danym tonie

    //allLyrics = this.allLyricsInOne;
    Map<String, Integer> sentenceToneMap = new HashMap<>(); // k - tone_name, v - count
    List<SentenceAnalysis> sentenceToneList = this.analyzeLyricsTone().getSentencesTone();

    for (SentenceAnalysis sentence : sentenceToneList) {
        for (ToneScore score : sentence.getTones()) {
            sentenceToneMap.compute(score.getToneName(), (k, v) -> ((v == null) ? 1 : v + 1)); // zliczanie
        }
    }
    List<WordCount> toneData = new ArrayList<>();
    for (Map.Entry<String, Integer> entry : sentenceToneMap.entrySet()) {
        toneData.add(new WordCount(entry.getValue(), entry.getKey()));
    }
    return toneData;
}
```

Rysunek 38
Metoda getSpecificSentencesTones(), która zlicza zdania zaklasyfikowane przez serwis Tone Analyzer jako dany ton
Źródło: Opracowanie własne

Gotowe wyniki zostają następnie przesłane do aplikacji po stronie klienta, gdzie przedstawiono je w odpowiedniej formie. Na rysunku 38 zobaczyć można strukturę strony głównej w HTML.

```
<div class="main">
    <div class="row waiter" *ngIf="!wordData || !toneData || !songArray || !songsTonesMap || !TONESCOREMAP">
        <h4>We're generating your musics' analysis, please wait </h4>
        <span class="spinner-border spinner-border-sm spinner"></span>
    </div>
    <div class="row" *ngIf="wordData && toneData && songArray && songsTonesMap && TONESCOREMAP">
        <div class="col">
            <app-song-list [songArray]="songArray" [songsTonesMap]="songsTonesMap"></app-song-list>
        </div>
        <div class="col">
            <app-word-cloud [wordData]="wordData" [toneData]="toneData"></app-word-cloud>
        </div>
        <div class="col">
            <app-chart [TONESCOREMAP]="TONESCOREMAP" [toneData]="toneData"></app-chart>
        </div>
    </div>
</div>
```

Rysunek 39
Struktura strony głównej aplikacji demonstrującej działanie serwisu ToneAnalyzer, czyli plik main-page.component.html
Źródło: Opracowanie własne

Dzięki frameworkowi Angular można umieszczać w pliku HTML znaczniki zdefiniowane przez siebie. Ich nazwa pochodzi od nazwy komponentu z przedrostkiem „app”, na przykład <app-song-list>, czy też <app-word-cloud> (porównaj rys. 38). Strona główna jak widać w trakcie oczekiwania na analizę wyświetla napis „We’re generating your musics’ analysis, please wait”, czyli „Analiza twojej muzyki jest generowana, proszę poczekać”. Drugi znacznik podzielony jest natomiast na trzy części. Pierwsza z nich <app-song-list> zajmuje się wyświetlaniem listy piosenek, druga <app-word-cloud> wykresów typu WordCloud (chmura słów), natomiast trzecia <app-chart> zawiera pozostałe dwa wykresy.

Podsumowanie

Sztuczna inteligencja to temat cały czas aktualny i rozwijający się. Wielu ludzi, w tym przedsiębiorcy, nie zdaje sobie sprawy z istnienia intrygujących rozwiązań jakie dziedzina ta niesie ze sobą. Zaimplementowanie tych rozwiązań w swoich serwisach biznesowych może sprawiać problem. Celem pracy było stworzenie zestawu aplikacji webowych, które implementują wybrane usługi IBM Watson AI i w ciekawy sposób ich używają, pokazując przykładowe zastosowania. Budowane aplikacje wykorzystują narzędzia sztucznej inteligencji oparte o analizę tekstu. Podane w pracy przykładowe wykorzystanie tych narzędzi pokazuje możliwość zastosowania mocy IBM Watson AI w zadaniach dotyczących wszelakich opinii: produktu, firmy, pozycji i marki. Podany został dokładny opis efektu końcowego oraz najważniejsze fragmenty kodu. Omówiono obecnie najczęściej stosowane technologie do budowania aplikacji webowych, takie jak Java, Spring, Maven. Nie zabrakło również jednych z ważniejszych i często używanych wzorców projektowych, takich jak na przykład Dependency Injection lub wzorzec Singleton.

Napisane aplikacje nie tylko zachęcają do wykorzystania rozwiązań sztucznej inteligencji w praktyce, ale również pokazują w jaki sposób można je zaimplementować. Z wykorzystaniem serwisu IBM Watson Tone Analyzer, Spotify API oraz MusixMatch API, udało się zaimplementować aplikację, która analizuje emocje skryte za polubionymi przez użytkownika tekstami piosenek. Dzięki temu można zobaczyć jakie emocje przemawiają za słuchanymi ostatnio przez użytkownika utworami. Z kolei dzięki użyciu API popularnego portalu społecznościowego Twitter oraz serwisu IBM Watson Personality Insights, zaimplementowano aplikację wykonującą wnikliwą analizę profilu osobowości obserwowanych użytkowników lub siebie samego. Z tego wynika, że funkcje Watson'a można zastosować również w celach biznesowych, ponieważ jak pokazały badania przeprowadzone przez IBM, konkretne cechy osobowości mają wpływ na podejmowane decyzje.

Okazuje się, że wykorzystanie znanych i popularnych technologii internetowych sprawia, że budowanie takich aplikacji jest stosunkowo proste. Pokazane w pracy przykłady oswajają czytelnika kodu z najpopularniejszymi w dzisiejszych czasach językami programowania oraz frameworkami. Natomiast moc obliczeniowa superkomputera została wykorzystana w algorytmach do przetwarzania dużych ilości tekstu oraz wysuwania wniosków, co pozwoliło aby końcowym efektem obydwu aplikacji była szczegółowa analiza zarówno cech osobowości, jak i emocji skrytych za tekstem.

Bibliografia

Literatura i publikacje

1) M. Gawłowska-Bujok, T. Bujok, Raport „Rynek pracy IT w 2018 roku", no fluff {jobs}: 2018
2) W. Strus, J. Cieciuch, *Poza Wielką Piątkę – Przegląd Nowych Modeli Struktury Osobowości*, Warszawa 2014, s.18
3) J. Cieciuch, M. Łaguna, *Wielka Piątka i nie tylko. Cechy osobowości i ich pomiar*, Warszawa 2014, str. 240-241
4) S. H. Schwartz, Universals in the content and structure of values: theoretical advances and empirical tests in 20 countries, 1992, str. 25-26.
5) J.B. Gleason, N.B. Ratnen, „*Psycholingwistyka*", Gdańskie Wydawnictwo Psychologiczne, Gdańsk 2005 (s. 17)
6) MarkLogic Corporation, „MarkLogic Server – REST Application Developer's Guide", 2019, s. 11, wersja pdf
7) C. S. Horstman, *Java Podstawy*, Gliwice, 2016
8) B. J. Evans, D. Flanagan „Java w pigułce" Wydanie VI, Helion S.A , Gliwice 2015
9) Krzysztof Rychlicki-Kicior „Java EE 6 – Programowanie aplikacji WWW", Helion, Gliwice 2010
10) R. Johnson, J. Hoeller, A. Arendsen, C. Sampaleanu, R. Harrop, T. Risberg, D. Davison, D. Kopylenko, M. Pollack, T. Templier, E. Vervaet, P. Tung, B. Hale, A. Colyer, J. Lewis, C. Leau, M. Fisher, S. Brannen, R. Laddad, A. Poutsma*, Spring. Java/j2ee Application Framework*. *The Spring Framework – Reference Documentation,* 2004-2008
11) J. Sharma, A. Sarin „Spring Framework – Wprowadzenie do tworzenia aplikacji" Wydanie II, Helion, Gliwice 2015
12) Andrzej Skowron „Technologie internetowe client-side na przykładzie języka JavaScript" w wersji .pdf
13) TutorialsPoint www.tutorialspoint.pl „JavaScript language" w wersji .pdf s. 11 (21.05.2020)
14) TutorialsPoint www.tutorialspoint.pl „TypeScript" w wersji .pdf s. 1, 2 (21.05.2020)

Źródła internetowe:

- https://www.investopedia.com/terms/d/deep-learning.asp (12.01.2020)
- https://pl.wikipedia.org/wiki/Framework (12.01.2020)
- https://pl.wikipedia.org/wiki/J%C4%99zyk_naturalny (10.12.2019)
- https://pl.wikipedia.org/wiki/Kod_bajtowy_Javy (14.01.2020)
- https://pl.wikipedia.org/wiki/Maszyna_wirtualna (14.01.2020)
- https://pl.wikipedia.org/wiki/Interfejs_programowania_aplikacji (30.06.2020)

- https://pl.wikipedia.org/wiki/Sztuczna_inteligencja (10.12.2019)
- https://encyklopedia.interia.pl/komputery/news-dysk-winchester,nId,2017796 (10.12.2019)
- https://pl.wikipedia.org/wiki/IBM (10.12.2019)
- https://www.ibm.com/blogs/think/2019/10/watson-anywhere-the-future/ (10.12.2019)
- https://en.wikipedia.org/wiki/Jeopardy! (13.12.2019)
- https://en.wikipedia.org/wiki/Watson_(computer) (27.12.2019)
- https://www.computerweekly.com/photostory/450423802/AI-A-brief-history-of-man-versus-machine-intelliegnce/3/IBM-Watson-versus-Jeopardy (27.12.2019)
- https://www.computerweekly.com/photostory/450423802/AI-A-brief-history-of-man-versus-machine-intelliegnce/3/IBM-Watson-versus-Jeopardy (10.12.2019)
- https://cloud.ibm.com/docs/services/personality-insights?topic=personality-insihts-gettingStarted (12.01.2020)
- https://cloud.ibm.com/docs/services/personality-insights?topic=personality-insights-science (13.01.2020)
- https://cloud.ibm.com/docs/services/personality-insights?topic=personality-insights-input (13.01.2020)
- https://personality-insights-demo.ng.bluemix.net/?_ga=2.265115469.337882622.1578833917-2031134999.1570907247 (12.01.2020)
- https://cloud.ibm.com/apidocs/tone-analyzer#analyze-general-tone (14.03.2020)
- https://cloud.ibm.com/docs/tone-analyzer?topic=tone-analyzer-about (14.03.2020)
- https://tone-analyzer-demo.ng.bluemix.net/?cm_mc_uid=10155467735015553357446&cm_mc_sid_50200000=26734711588354771949&cm_mc_sid_52640000=95640691588354844949&_ga=2.107633730.1971899266.1588354773-2031134999.1570907247 (14.03.2020)
- https://cloud.ibm.com/docs/services/tone-analyzer?topic=tone-analyzer-caseStudies (15.03.2020)
- https://www.empressia.pl/blog/72-aplikacje-webowe-czym-sa-jak-dzialaja-i-jakie-sa-ich-zalety (23.05.2020)
- https://www.samouczekprogramisty.pl/wprowadzenie-do-aplikacji-webowych/ (23.05.2020)
- http://yarpo.pl/2012/07/29/rest-ciekawszy-sposob-na-komunikacje-client-server/ (24.05.2020)
- https://itwiz.pl/jezyki-programowania-ranking-styczen-2020/ (15.05.2020)
- https://pl.wikipedia.org/wiki/Java (14.01.2020)
- https://riptutorial.com/pl/java/example/27916/roznice-miedzy-java-ee--java-se--java-me-i-javafx (15.05.2020)
- https://docplayer.pl/38866069-Krotka-historia-java.html (15.05.2020)
- https://pl.wikipedia.org/wiki/Java_Platform,_Micro_Edition (15.05.2020)
- https://riptutorial.com/pl/java/example/27916/roznice-miedzy-java-ee--java-se--java-me-i-javafx (15.05.2020)
- https://javastart.pl/baza-wiedzy/frameworki/javafx (15.05.2020)
- https://pl.wikipedia.org/wiki/TypeScript (21.05.2020)
- https://www.youtube.com/watch?v=8sC55WKzJW4 (24.05.2020)
- https://angular.io/guide/template-syntax - Oficjalna dokumentacja Angular (24.05.2020)

- https://support.spotify.com/pl/using_spotify/getting_started/what-is-spotify/ (16.06.2020)
- https://www.spidersweb.pl/2020/02/spotify-wyniki-q4-2019.html (16.06.2020)
- https://pl.wikipedia.org/wiki/Twitter (21.06.2020)
- https://cloud.ibm.com/catalog/services/tone-analyzer (05.07.2020)

Spis rysunków: