

**Politechnika Wrocławskaw
Wydział Informatyki i Telekomunikacji**

Kierunek: **Informatyka techniczna (ITE)**
Specjalność: **Inżynieria systemów informatycznych (INS)**

**PRACA DYPLOMOWA
INŻYNIERSKA**

**Aplikacja internetowa do gromadzenia
i udostępniania informacji o
technologiach programistycznych**

**A web application for gathering and
sharing information about software
development**

Kamil Dywan

Opiekun pracy
dr inż, Paweł Rogaliński

Streszczenie

Praca dotyczy aplikacji webowej IT Tech umożliwiającej gromadzenie i udostępnianie informacji o technologiach programistycznych w postaci artykułów. Aplikacja zawiera również rozbudowane systemy zostawiania informacji zwrotnej oraz recenzji artykułów. Na początku tej pracy znajduje się krótki wstęp, w którym określono potrzebę realizacji takiego serwisu. Następnie dokonano analizy oraz oceny istniejących podobnych serwisów i zaproponowano mechanizmy, które rozwiązałyby największe problemy tych serwisów. Po wstępie znajduje się projekt systemu, w którym na początku określono wymagania funkcjonalne i niefunkcjonalne. Na podstawie wymagań oraz wstępu opracowano architekturę systemu. Realizowany system jest rozproszonym serwisem webowym, którego komunikacja odbywa się głównie poprzez REST. W architekturze systemu można wyróżnić aplikację kliencką, aplikację serwerową, warstwę bezpieczeństwa oraz bazy SQL i NoSQL. Następnie opracowano przypadki użycia wraz z ich szczegółowymi opisami oraz zaprojektowano odpowiadające im prototypy interfejsu użytkownika. Ostatnim etapem projektowania jest opracowanie diagramu ERD dla bazy SQL oraz schematów dla dokumentowej bazy NoSQL. Następnym etapem po projektowaniu jest implementacja systemu. Aplikacja serwerowa została zaimplementowana głównie z użyciem frameworka Spring, a do implementacji aplikacji klienckiej wykorzystano bibliotekę React. Ostatnim etapem implementacji systemu było wprowadzenie warstwy bezpieczeństwa w postaci zewnętrznego serwera do uwierzytelniania i autoryzacji o nazwie Keycloak. Działanie aplikacji serwerowej zostało zweryfikowane poprzez przeprowadzenie testów integracyjnych kontrolerów. Kolejnym etapem w pracy jest opisane wdrożenie całego systemu z wykorzystaniem platform Azure oraz MongoDB Cloud. Na końcu pracy znajduje się krótka instrukcja wdrożeniowa opisująca lokalne uruchomienie systemu.

Słowa kluczowe: Aplikacja webowa, Technologie programistyczne, System informacji zwrotnej, Java, Spring, JavaScript, TypeScript, React, MUI, Redux, PostgreSQL, MongoDB, Keycloak, Docker, Azure

Abstract

This thesis describes a web application IT Tech for gathering and sharing information in the form of articles about software development. The application also includes expanded systems of leaving feedback and article review. In the introduction of this thesis there is a brief about the need to implement the system. Then there are analysis and evaluation about existing similar services and later are proposed mechanisms that could solve the biggest problems of these services. After introduction there is project of the system in which are described functional and non-functional requirements. Based on requirements and introduction, was specified system architecture. IT Tech is distributed web application with REST as a main form of communication. In system architecture there are client app, server app, security layer and SQL and NoSQL databases. Then was specified use cases with their details and designed user interface prototypes according to them. Last part of the design was create ERD diagram for SQL database and schemas for NoSQL database. The next step is implementation of system. Server app was implemented mainly using Spring framework and React library was used for implementation of client app. Last step of system implementation was introduction of security layer

in form of an external server for authentication and authorization called Keycloak. Backend app was verified by using integration tests of controllers. The next step is the described deployment of the entire system using Azure and MongoDB Cloud platforms. At the end of the thesis, there is a deployment instruction which describing local start of the system.

Keywords: Web application, Programming technologies, Feedback system, Java, Spring, JavaScript, TypeScript, React, MUI, Redux, PostgreSQL, MongoDB, Keycloak, Docker, Azure

Spis treści

1. Wstęp	12
1.1. Wprowadzenie	12
1.2. Cel pracy	16
1.3. Zakres pracy	16
2. Projekt aplikacji IT Tech	17
2.1. Wymagania funkcjonalne	17
2.1.1. Słownik pojęć	17
2.1.2. Wymagania funkcjonalne dla systemu IT Tech	20
2.2. Wymagania niefunkcjonalne	23
2.3. Architektura systemu	23
2.3.1. Baza danych	25
2.3.2. Aplikacja serwerowa - Backend	25
2.3.3. Klient - Frontend	25
2.3.4. Serwer webowy	26
2.3.5. Warstwa bezpieczeństwa	26
2.4. Diagramy przypadków użycia	26
2.4.1. Identyfikacja aktorów	26
2.4.2. Diagramy przypadków użycia	27
2.4.3. Opisy przypadków użycia	32
2.5. Projekt baz danych	81
3. Implementacja systemu IT Tech	87
3.1. Technologie i narzędzia	87
3.1.1. Technologie	87
3.1.2. Narzędzia	89
3.2. Aplikacja serwerowa	89
3.3. Aplikacja kliencka	102
4. Testowanie systemu IT Tech	111
4.1. Testy kontrolera użytkowników	111
4.1.1. Test sprawdzający, czy można pobrać dane użytkownika na podstawie jego numeru konta	111
4.1.2. Test sprawdzający, czy nie można pobrać danych użytkownika na podstawie nieistniejącego numeru konta	112
4.1.3. Test sprawdzający, czy zostanie zwrócona prawda w przypadku istnienia użytkownika o podanym pseudonimie	112
4.1.4. Test sprawdzający, czy zostanie zwrócony fałsz w przypadku nieistnienia użytkownika o podanym pseudonimie	112
4.1.5. Test sprawdzający, czy można utworzyć użytkownika	113
4.1.6. Test sprawdzający, czy nie można utworzyć użytkownika z podanym istniejącym już numerem konta	113

4.1.7. Test sprawdzający, czy nie można utworzyć użytkownika z podanym istniejącym już pseudonimem	114
4.1.8. Test sprawdzający, czy nie można utworzyć użytkownika z pustymi danymi	114
4.1.9. Test sprawdzający, czy nie można utworzyć użytkownika z pustym id konta	114
4.1.10. Test sprawdzający, czy nie można utworzyć użytkownika z pustym pseudonimem	115
4.1.11. Test sprawdzający, czy można zaktualizować dane użytkownika o podanym id	115
4.1.12. Test sprawdzający, czy nie można zaktualizować danych użytkownika o podanym niepoprawnym id	116
4.1.13. Test sprawdzający, czy nie można zaktualizować danych użytkownika z podanymi pustymi danymi	117
4.1.14. Test sprawdzający, czy nie można zaktualizować danych użytkownika z podanym nieistniejącym id	117
4.1.15. Test sprawdzający, czy nie można zaktualizować danych użytkownika z podanym już istniejącym pseudonimem	117
4.1.16. Wyniki testów	118
4.2. Testy kontrolera kategorii technologii	118
4.2.1. Test sprawdzający, czy można pobrać poprawne drzewo kategorii technologii	118
4.2.2. Test sprawdzający, czy można pobrać dane kategorii technologii o podanym id	120
4.2.3. Test sprawdzający, czy nie można pobrać danych kategorii technologii o podanym niepoprawnym id	120
4.2.4. Test sprawdzający, czy nie można pobrać danych kategorii technologii o podanym nieistniejącym id	120
4.2.5. Wyniki testów	121
4.3. Testy kontrolera technologii	121
4.3.1. Test sprawdzający, czy można pobrać dane wszystkich technologii	121
4.3.2. Test sprawdzający, czy można pobrać listę technologii o podanym ich id kategorii	121
4.3.3. Test sprawdzający, czy nie można pobrać listy technologii o podanym niepoprawnym id kategorii	122
4.3.4. Test sprawdzający, czy nie można pobrać listy technologii o podanym nieistniejącym id kategorii	122
5. Wdrożenie systemu IT Tech	123
5.1. Baza danych PostgreSQL	123
5.2. Aplikacja kliencka oraz serwer webowy	126
5.3. Aplikacja serwerowa	131
5.4. Warstwa bezpieczeństwa	134
5.5. Baza danych MongoDB	137
5.6. Podsumowanie wdrożenia	138
6. Podsumowanie	139
Literatura	140
A. Instrukcja wdrożeniowa	142
B. Opis załączonej płyty CD/DVD	143

Spis rysunków

1.1. Fragment specyfikacji języka Java w wersji 17	13
1.2. Wikipedia	14
1.3. Medium	15
2.1. Cykl życia artykułu	19
2.2. Architektura systemu	24
2.3. Dziedziczenie między aktorami	27
2.4. Diagram przypadków użycia dla Użytkownika	28
2.5. Diagram przypadków użycia dla Niezalogowanego użytkownika	28
2.6. Diagram przypadków użycia dla Zalogowanego użytkownika	29
2.7. Diagram przypadków użycia dla Autora artykułu	29
2.8. Diagram przypadków użycia dla Autora komentarza o artykule	30
2.9. Diagram przypadków użycia dla Autora opinii o artykule	30
2.10. Diagram przypadków użycia dla Autora akceptacji opinii artykułu	30
2.11. Diagram przypadków użycia dla Recenzenta	31
2.12. Diagram przypadków użycia dla Administratora	31
2.13. Diagram przypadków użycia dla Systemu	32
2.14. Nagłówek strony niezalogowanego użytkownika	33
2.15. Formularz rejestracji	34
2.16. Formularz służący do akceptacji regulaminu	35
2.17. Formularz logowania	37
2.18. Nagłówek strony zalogowanego użytkownika	37
2.19. Formularz do ustawienia hasła po zresetowaniu hasła	37
2.20. Nagłówek strony zalogowanego użytkownika z otwartymi opcjami dotyczącymi konta	38
2.21. Panel z danymi użytkownika	39
2.22. Panel umożliwiający użytkownikowi edycję własnych danych	40
2.23. Formularz zmiany hasła	41
2.24. Nagłówek strony administratora	43
2.25. Panel do wyszukiwania użytkowników	43
2.26. Panel z danymi użytkownika widziany przez administratora	44
2.27. Panel umożliwiający administratorowi edycję danych użytkownika	45
2.28. Nagłówek strony dostępnego recenzenta z otwartymi opcjami dotyczącymi konta	48
2.29. Formularz z ustawieniem daty, do której ma być ustawniona nieobecność recenzenta razem z otwartym oknem wyboru daty	48
2.30. Nagłówek strony niedostępnego recenzenta z otwartymi opcjami dotyczącymi konta	48
2.31. Panel do wyszukiwania artykułów z podanymi kryteriami wyszukiwania oraz listą wyszukanych artykułów	50
2.32. Dostępne opcje sortowania artykułów	51
2.33. Panel z przykładową zawartością artykułu	52
2.34. Panel z formularzem dodania albo edytowania artykułu	54

2.35. Nagłówek strony recenzenta	56
2.36. Panel z listą artykułów przypisanych recenzentowi do weryfikacji	57
2.37. Panel służący do weryfikacji artykułu	58
2.38. Formularz akceptacji artykułu z podaną informacją zwrotną	59
2.39. Formularz odrzucenia artykułu	60
2.40. Panel z zawartością artykułu widziany z perspektywy administratora	63
2.41. Panel z listą komentarzy o artykule	64
2.42. Formularz dodania albo edytowania komentarza o artykule	65
2.43. Dostępne opcje zarządzania komentarzem	66
2.44. Lista komentarzy o artykule widziana z perspektywy administratora	67
2.45. Panel z listą opinii o artykule	68
2.46. Formularz dodania albo edytowania opinii o artykule	69
2.47. Panel z listą opinii o artykule widziany z perspektywy administratora	71
2.48. Panel do wyszukiwania technologii z podanymi kryteriami wyszukiwania oraz listą wyszukanych technologii	73
2.49. Dostępne opcje sortowania technologii	74
2.50. Panel z zawartością technologii	75
2.51. Panel do wyszukiwania technologii z podanymi kryteriami wyszukiwania oraz listą wyszukanych technologii widziany z perspektywy recenzenta	78
2.52. Panel z formularzem dodania albo modyfikacji technologii	79
2.53. Diagram ERD dla głównej bazy danych	81
 3.1. Użyte technologie	87
3.2. Architektura aplikacji serwerowej	90
3.3. Struktura projektu aplikacji serwerowej	91
3.4. Endpointy kontrolerów dla użytkowników, opinii oraz komentarzy	100
3.5. Endpointy kontrolerów dla artykułów oraz weryfikacji artykułów	101
3.6. Endpointy kontrolerów dla dostępności użytkowników, akceptacji opinii, technologii oraz kategorii technologii	102
3.7. Struktura projektu aplikacji klienckiej	103
3.8. Struktura folderu redux aplikacji klienckiej	106
3.9. Kod związany z serwerem Keycloak po stronie frontendu	107
 4.1. Wyniki testów integracyjnych backendu dla kontrolera użytkowników	118
4.2. Wyniki testów integracyjnych backendu dla kontrolera kategorii technologii	121
4.3. Wyniki testów integracyjnych backendu dla kontrolera technologii	122
 5.1. Podstawowa konfiguracja wdrożeniowa bazy danych PostgreSQL	124
5.2. Konfiguracja danych do logowania administratora bazy danych PostgreSQL	125
5.3. Wdrożona baza danych PostgreSQL	126
5.4. Podstawowa konfiguracja wdrożeniowa aplikacji klienckiej oraz serwera webowego	127
5.5. Konfiguracja Dockera po stronie Azure dla aplikacji klienckiej	128
5.6. Utworzona usługa App Service dla aplikacji klienckiej	129
5.7. Widok strony głównej aplikacji po jej wdrożeniu	131
5.8. Podsumowanie konfiguracji aplikacji wdrożeniowej	132
5.9. Przykładowe żądanie pobierające wszystkie technologie i uzyskana odpowiedź od wdrożonego backendu	134
5.10. Podsumowanie konfiguracji serwera uwierzytelniającego i autoryzującego	135
5.11. Konfiguracja serwera Keycloak	136
5.12. Interfejs użytkownika dla wdrożonego serwera Keycloak	137
5.13. Podstawowa konfiguracja wdrożeniowa bazy danych MongoDB	137

5.14. Dodatkowa konfiguracja wdrożeniowa bazy danych MongoDB	138
--	-----

Spis tabel

2.1. Opis przypadku użycia - Rejestracja	32
2.2. Opis przypadku użycia - Logowanie	35
2.3. Opis przypadku użycia - Wylogowanie	37
2.4. Opis przypadku użycia - Edycja własnych danych	38
2.5. Opis przypadku użycia - Zmiana hasła	40
2.6. Opis przypadku użycia - Edycja danych innych użytkowników	41
2.7. Opis przypadku użycia - Zarządzanie rolami użytkownika	45
2.8. Opis przypadku użycia - Zmiana własnej dostępności	46
2.9. Opis przypadku użycia - Wyszukiwanie artykułów	49
2.10. Opis przypadku użycia - Sortowanie artykułów	50
2.11. Opis przypadku użycia - Wyświetlenie zawartości artykułu	51
2.12. Opis przypadku użycia - Utworzenie artykułu	53
2.13. Opis przypadku użycia - Zarządzanie własnym artykułem	54
2.14. Opis przypadku użycia - Wyświetlenie listy przypisanych artykułów	56
2.15. Opis przypadku użycia - Weryfikacja artykułu	57
2.16. Opis przypadku użycia - Akceptacja artykułu	58
2.17. Opis przypadku użycia - Odrzucenie artykułu	59
2.18. Opis przypadku użycia - Zarządzanie artykułami	61
2.19. Opis przypadku użycia - Przeglądanie komentarzy o artykule	64
2.20. Opis przypadku użycia - Dodanie komentarza o artykule	64
2.21. Opis przypadku użycia - Zarządzanie własnym komentarzem o artykule	65
2.22. Opis przypadku użycia - Usuwanie komentarzy o artykule	66
2.23. Opis przypadku użycia - Przeglądanie opinii o artykule	67
2.24. Opis przypadku użycia - Dodanie opinii o artykule	68
2.25. Opis przypadku użycia - Zarządzanie własną opinią o artykule	69
2.26. Opis przypadku użycia - Usuwanie opinii o artykule	70
2.27. Opis przypadku użycia - Dodanie akceptacji opinii o artykule	71
2.28. Opis przypadku użycia - Usunięcie akceptacji opinii o artykule	71
2.29. Opis przypadku użycia - Wyszukiwanie technologii	72
2.30. Opis przypadku użycia - Sortowanie technologii	73
2.31. Opis przypadku użycia - Wyświetlenie opisu technologii	74
2.32. Opis przypadku użycia - Edycja klasyfikacji technologii	75
2.33. Opis przypadku użycia - Wyszukiwanie artykułów z przeterminowaną weryfikacją	79
2.34. Opis przypadku użycia - Przypisanie recenzentowi artykułu do weryfikacji	80
2.35. Opis przypadku użycia - Przypisanie artykułu do weryfikacji innemu recenzentowi	80

Spis listingów

2.1. Schemat kolekcji artykułów w postaci JSON dla bazy danych artykułów	82
2.2. Przykładowy dokument dla kolekcji artykułów zgodny ze zdefiniowanym schematem	83
2.3. Schemat kolekcji komentarzy w postaci JSON dla bazy danych artykułów	83
2.4. Przykładowy dokument dla kolekcji komentarzy zgodny ze zdefiniowanym schematem	84
2.5. Schemat kolekcji opinii w postaci JSON dla bazy danych artykułów	85
2.6. Przykładowy dokument dla kolekcji opinii zgodny ze zdefiniowanym schematem .	86
3.1. Repozytorium technologii	91
3.2. Profil lokalny	92
3.3. Encja technologii	93
3.4. Mapowanie jeden do wielu między technologiami i kategoriami technologii	93
3.5. Encja artykułu	93
3.6. Repozytorium technologii	94
3.7. Repozytorium artykułów	95
3.8. Serwis artykułów	95
3.9. Kontroler artykułów	96
3.10. Mapper artykułów	97
3.11. Zabezpieczenie endpointów	98
3.12. Konfiguracja JWT	99
3.13. Konfiguracja profilu lokalnego dla aplikacji klienckiej	104
3.14. Fragment zawartości pliku package.json używanego przez frontend	104
3.15. Kod komponentu awatar użytkownika	104
3.16. Model artykułu w aplikacji klienckiej	105
3.17. Fragment serwisu artykułów	105
3.18. Fragment kodu slice zalogowanego użytkownika	106
3.19. Konfiguracja komunikacji z serwerem Keycloak	107
3.20. Fragment kodu serwisu odpowiadającego za komunikację z serwerem Keycloak .	107
3.21. Komponent ProtectedRoute	109
3.22. Zabezpieczona nawigacja	109
4.1. Test integracyjny pobierania danych użytkownika po jego id konta	111
4.2. Test integracyjny pobierania danych użytkownika po nieistniejącym id konta	112
4.3. Test integracyjny zwracania prawdy w przypadku istnienia użytkownika o podanym pseudonimie	112
4.4. Test integracyjny zwracania fałszu w przypadku nie istnienia użytkownika o podanym pseudonimie	112
4.5. Test integracyjny tworzenia użytkownika	113
4.6. Test integracyjny tworzenia użytkownika z już istniejącym numerem konta	113
4.7. Test integracyjny tworzenia użytkownika z już istniejącym pseudonimem	114
4.8. Test integracyjny tworzenia użytkownika z pustymi danymi	114
4.9. Test integracyjny tworzenia użytkownika z pustym id konta	114
4.10. Test integracyjny tworzenia użytkownika z pustym pseudonimem	115

4.11. Test integracyjny aktualizowania danych użytkownika po id	115
4.12. Test integracyjny aktualizowania danych użytkownika po niepoprawnym id	116
4.13. Test integracyjny aktualizowania danych użytkownika z podanymi pustymi danymi	117
4.14. Test integracyjny aktualizowania danych użytkownika z podanym nieistniejącym id	117
4.15. Test integracyjny aktualizowania danych użytkownika z podanym już istniejącym pseudonimem	117
4.16. Test integracyjny pobierania drzewa kategorii technologii	118
4.17. Test integracyjny pobierania kategorii technologii o podanym id	120
4.18. Test integracyjny pobierania kategorii technologii o podanym id	120
4.19. Test integracyjny pobierania kategorii technologii o podanym nieistniejącym id	120
4.20. Test integracyjny pobierania wszystkich technologii	121
4.21. Test integracyjny pobierania listy technologii o podanym ich id kategorii	121
4.22. Test integracyjny pobierania listy technologii o podanym niepoprawnym id kategorii	122
4.23. Test integracyjny pobierania listy technologii o podanym nieistniejącym id kategorii	122
5.1. Konfiguracja Dockerfile dla aplikacji klienckiej i serwera webowego	129
5.2. Konfiguracja serwera webowego NGINX	130
5.3. Konfiguracja Dockerfile dla aplikacji serwerowej	132

Rozdział 1

Wstęp

1.1. Wprowadzenie

Z powodu szybkiego rozwoju informatyki potrzebne jest ciągłe bycie na bieżąco z technologiami programistycznymi oraz obserwowanie trendów. Ważne jest zatem aby publikowane treści były jak najbardziej aktualne. Nieaktualne treści najczęściej występują w książkach, gdyż pisanie książki trwa zazwyczaj bardzo długo np. rok. Często zdarza się sytuacja, że książka w dniu wydania już zawiera treści przestarzałe.

Programista, który chciałby się czegoś dowiedzieć o technologii, z którą nie jest zaznajomiony, najczęściej może skorzystać z dokumentacji tej technologii albo serwisu informacyjnego dostarczającego treści z wielu dziedzin (np. Wikipedia [1]). Z jednej strony dokumentacje technologii zawierają szczegółowe treści oraz rzadko kiedy wplatane są jakieś przykłady, a z drugiej strony encyklopedyczne serwisy informacyjne zawierają jedynie treści ogólne. Dokumentacja może być niezrozumiała dla tego użytkownika, a serwisy informacyjne obejmujące wiele dziedzin najczęściej nie dostarczą mu wystarczających informacji.

Dobrym przykładem typowej dokumentacji jest specyfikacja języka Java w wersji 17 [2]. Fragment tej specyfikacji jest przedstawiony na rysunku 1.1.

The screenshot shows a dark-themed PDF viewer interface. At the top, there are navigation controls: page number (86), total pages (848), zoom level (100%), and other icons. Below the header, the page content is displayed.

4.5 Parameterized Types

If c had been declared in a different package than τ , then the call to `mCPackage` would give rise to a compile-time error, as that member would not be accessible at the point where τ is declared.

4.5 Parameterized Types

A class or interface that is generic (§8.1.2, §9.1.2) defines a set of *parameterized types*.

A parameterized type is a class or interface type of the form $c<\tau_1, \dots, \tau_n>$, where c is the name of a generic class or interface, and $\langle\tau_1, \dots, \tau_n\rangle$ is a list of type arguments that denote a particular *parameterization* of the generic class or interface.

A generic class or interface has type parameters F_1, \dots, F_n with corresponding bounds B_1, \dots, B_n . Each type argument τ_i of a parameterized type ranges over all types that are subtypes of all types listed in the corresponding bound. That is, for each bound type s in B_i , τ_i is a subtype of $S[F_1 := T_1, \dots, F_n := T_n]$ (§4.10).

A parameterized type $c<\tau_1, \dots, \tau_n>$ is *well-formed* if all of the following are true:

- c is the name of a generic class or interface.
- The number of type arguments is the same as the number of type parameters in the generic declaration of c .
- When subjected to capture conversion (§5.1.10) resulting in the type $c<X_1, \dots, X_n>$, each type argument X_i is a subtype of $S[F_1 := X_1, \dots, F_n := X_n]$ for each bound type s in B_i .

It is a compile-time error if a parameterized type is not well-formed.

In this specification, whenever we speak of a class or interface type, we include parameterized types as well, unless explicitly excluded.

Two parameterized types are *provably distinct* if either of the following is true:

- They are parameterizations of distinct generic type declarations.
- Any of their type arguments are provably distinct.

Given the generic classes in the examples of §8.1.2, here are some well-formed parameterized types:

- `Seq<String>`
- `Seq<Seq<String>>`
- `Seq<String>.Zipper<Integer>`

70

Rys. 1.1: Fragment specyfikacji języka Java w wersji 17

źródło: <https://docs.oracle.com/javase/specs/jls/se17/jls17.pdf>

Przedstawiony fragment dotyczy typów sparametryzowanych. Specyfikacja ta jest bardzo obszerna, szczegółowa, rzadko kiedy wplatane są przykłady oraz brak jest kolorowych grafik, które mogłyby poprawić czytelność tekstu oraz jego zrozumienie. Dokumentacja ta jest zbyt trudna w odbiorze dla użytkowników, którzy nie są ekspertami w tej technologii.

Aktualnie nie ma popularnych serwisów informacyjnych zorientowanych jedynie na technologiach programistycznych, ale istnieje na rynku dużo serwisów, które są zorientowane na wiele dziedzin.

Jednym z takich serwisów jest niezwykle popularna Wikipedia, której przykładowy artykuł jest przedstawiony na rysunku 1.2. Wikipedia jest to serwis obejmujący bardzo wiele dziedzin, który umożliwia użytkownikom publikowanie treści w postaci artykułów. Treści publikowane w tym serwisie zazwyczaj są utrzymane w tonie encyklopedii. Serwis ten jest zorientowany na informacje i dlatego posiada słabo rozwiniętą formę społecznościową. Przy każdym z artykułów jest sekcja „Dyskusja”, w której można umieścić komentarz lub odpowiedzieć na komentarz innego użytkownika.

Rys. 1.2: Wikipedia

Źródło: <https://pl.wikipedia.org/wiki/Java>

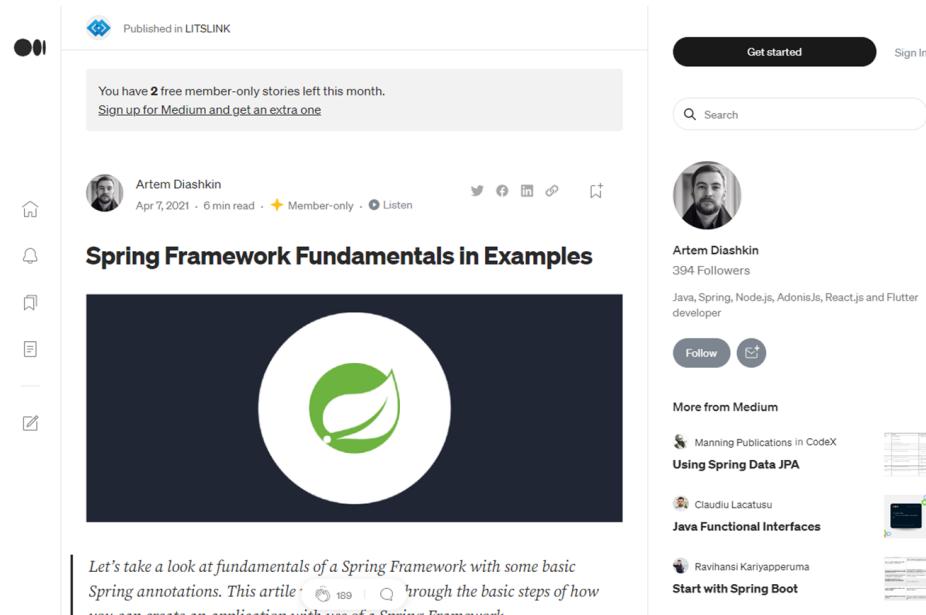
Można wyróżnić następujące zalety Wikipedii:

- czytelne i ładne artykuły,
- artykuły posiadają podobną budowę,
- darmowy dostęp dla każdego użytkownika,
- rozbudowany formularz tworzenia i edycji artykułu.

Do wad Wikipedii można zaliczyć:

- sekcja komentarzy jest nieczytelna,
- za mało kryteriów wyszukiwania,
- często artykuły są zbyt ogólne,
- brak systemu opinii.

Podobnym serwisem do Wikipedii jest Medium 1.3 [3]. Medium również obejmuje wiele dziedzin. Użytkownicy także publikują treści w tym serwisie poprzez artykuły, jednak treści te są najczęściej utrzymane w swobodniejszym tonie, niż encyklopedycznym. Serwis ten można zaliczyć do serwisów społecznościowych. Na początku każdego artykułu widnieje wyraźna informacja o autorze, można wejść na profil tego autora, zaobserwować go albo zobaczyć napisane przez niego artykuły. W serwisie Medium jest również system zagnieżdzonych komentarzy oraz oceniania artykułów. Każdy użytkownik może zostawić jednostopniową pochwałę do artykułu lub komentarza.



Rys. 1.3: Medium

Źródło: <https://medium.com/litslink/spring-framework-fundamentals-in-examples-b37101aa9250>

Można wyróżnić następujące zalety Medium:

- czytelne i ładne artykuły,
- rozbudowany system wyszukiwania artykułów (m.in. po kategorii, czy autorze),
- czytelna sekcja z komentarzami,
- rozbudowana forma społecznościowa.

Do wad Medium można zaliczyć:

- budowa artykułów może się mocno różnić,
- zbyt prosty formularz do tworzenia artykułów,
- zbyt mały zakres zostawianych ocen (jedynie jednostopniowa pochwała),
- aby mieć dostęp do wszystkich artykułów, trzeba posiadać płatne konto premium,
- brak obowiązkowego tekstowego uzasadnienia oceny artykułu.

Systemy umożliwiające dostarczanie treści każdemu użytkownikowi są szczególnie narażone na to, że publikowane treści będą nieodpowiednie np. spam, wirusy, czy informacje nieprawdziwe. W tego typu serwisach niezbędna jest moderacja. Często jednak niemożliwe jest zapewnienie odpowiedniej liczby moderatorów i potrzebny jest wtenczas mechanizm wspomagający ich pracę. Mechanizmem tym może być system komentarzy i opinii. Informacje zwrotne użytkowników na temat treści dostępnych w serwisie będą mogły być wtedy informacją o jakości tych treści zarówno dla moderatorów, jak i innych użytkowników. W przypadku wielu serwisów informacyjnych mechanizm zostawiania informacji zwrotnej (komentarze, oceny, opinie) jest słaby lub niewystarczająco rozbudowany. Najczęściej można się spotkać z następującymi niedociągnięciami:

- sekcja komentarzy jest nieczytelna oraz nie można odpowiedzieć na komentarz innego użytkownika,
- w przypadku opinii jest zbyt mały zakres wystawianej oceny (np. łapka w góre i w dół albo sama pochwała), jak i często nie ma wymogu zamieszczenia tekstu uzasadnienia do wystawionej oceny,
- brak możliwości ocenienia opinii innego użytkownika.

Podsumowując, problemem jest to, że nie ma popularnych serwisów informacyjnych, które zawierają treści będące pomiędzy dokumentacjami oraz ogólnymi serwisami informacyjnymi.

1.2. Cel pracy

Celem pracy jest zaprojektowanie i implementacja responsywnej aplikacji webowej IT Tech, która będzie umożliwiała użytkownikom tworzenie artykułów o technologiach programistycznych (np. dotyczących języków programowania), przeglądanie dostępnych artykułów oraz zostawienie informacji zwrotnej o artykule w postaci komentarza lub opinii. Artykuły będą mogły być wyszukiwane po wielu kryteriach, będzie rozbudowany system opinii, prosty i intuicyjny interfejs użytkownika oraz treści opublikowane w serwisie będą dostępne dla każdego użytkownika. Realizowana aplikacja będzie umożliwiała szybkie zamieszczanie treści każdemu użytkownikowi i dzięki temu możliwe będzie dostarczanie treści aktualnych oraz nadążanie za trendami. Aplikacja będzie rozwiązyaniem pośrednim między dokumentacjami oraz serwisami informacyjnymi obejmującymi wiele dziedzin. Nie będzie nacisku na dużą szczegółowość zamieszczanych treści oraz serwis będzie zorientowany jedynie na technologie programistyczne i przez to treści nie będą przez to zbyt ogólne.

1.3. Zakres pracy

Zakres pracy obejmuje: projekt, implementację, testowanie oraz wdrożenie i opracowanie dokumentacji. Projekt obejmuje: zdefiniowanie wymagań funkcjonalnych i niefunkcjonalnych, sporządzanie diagramów i opisów przypadków użycia, zrobienie widoków interfejsu użytkownika oraz wykonanie schematów bazy danych. Implementacja obejmuje: aplikacje serwerową (backend), aplikację kliencką (frontend) oraz uwierzytelnianie i autoryzację. Testowanie obejmuje testy integracyjne backendu.

Rozdział 2

Projekt aplikacji IT Tech

2.1. Wymagania funkcjonalne

Pierwszym etapem realizacji pracy jest określenie wymagań funkcjonalnych i niefunkcjonalnych. W tym celu konieczne jest wprowadzenie pojęć, które będą wykorzystywane w dalszej części pracy.

2.1.1. Słownik pojęć

W pracy będą wykorzystywane następujące pojęcia:

Technologia - pakiet oprogramowania lub narzędzia developerskie wykorzystywane przez informatyków, a w szczególności przez programistów. Technologia będzie opisywana następującymi atrybutami:

- Nazwa,
- Opis,
- Dostawca (opcjonalne),
- Ikona (opcjonalne),
- Kategoria,
- Data pierwszego wydania (opcjonalne),
- Data ostatniego wydania (opcjonalne),
- Daty edycji.

Podział technologii - hierarchiczny podział technologii na kategorie ze względu na ich architekturę. Klasyfikację tę można przedstawić w postaci następującego drzewa:

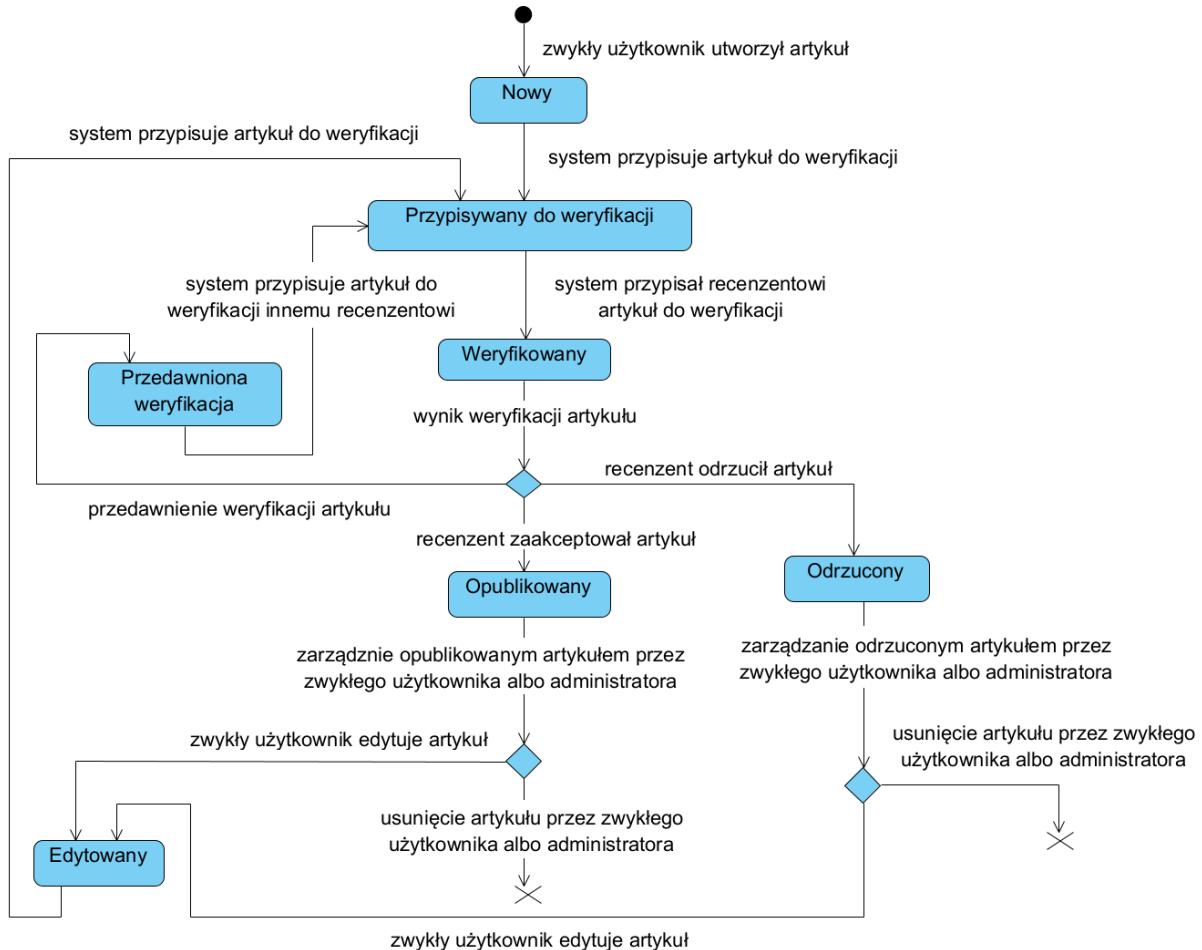
- Technologie informatyczne
 - Język programowania (np. Java, C++)
 - * Biblioteka (np. SFML, SDL, OpenGL),
 - * Framework (np. Spring, Spring Boot, React, Angular)
 - Język znaczników (np. TeX, HTML, XML)
 - Język bazy danych (np. SQL, GraphQL)
 - * Relacyjna (np. MySQL, Oracle Database, SQLite)
 - * Obiektowo-relacyjna (np. PostgreSQL)
 - * NoSQL (np. MongoDB, Cassandra)
- Środowiska uruchomieniowe
 - System operacyjny (np. Windows, Linux)
 - Środowiska wirtualizowane (JVM, .NET, node.js)

- Systemy wbudowane (np. Arduino, Raspberry Pi)
- Narzędzia developerskie
 - Środowisko programistyczne (np. Netbeans, Eclipse, Visual Studio Code, Visual Studio, ItelliJ)
 - System kontroli wersji (np. git)
 - * Serwisy hostujące gita (np. GitLab, GitHub)
 - CI/CD - ciągła integracja/ciągłe dostarczanie (np. Jenkins, GitLab CI)
 - Konteneryzacja (np. docker)
 - Orkiestracja (system do zarządzania, organizacji i planowania zasobów systemu – np. Docker Compose, Kubernetes)
- Inne (oprogramowanie, które nie zostało przypisane do żadnej z ww. kategorii)

Artykuł - opis danej technologii przygotowany przez użytkownika systemu. Artykuł będzie opisywany następującymi atrybutami:

- Status (możliwe wartości: nowy, przypisywany do weryfikacji, weryfikowany, przedawniona weryfikacja, odrzucony, opublikowany, edytowany),
- Tytuł,
- Autor,
- Data utworzenia,
- Data modyfikacji,
- Kategoria technologii wraz z całą hierarchiczną ścieżką prowadzącą do tej kategorii (np. dla Javy będzie to: Technologie Informatyczne / Język Programowania / Java),
- Dostawca technologii (opcjonalne),
- Ikona technologii (opcjonalne),
- Tekst artykułu,
- Lista opinii o artykule,
- Średnia ocena z co najmniej 5-ciu opinii o artykule,
- Data przypisania artykułu do weryfikacji,
- Recenzent przypisany do weryfikacji artykułu.

Artykuły podlegają cyklowi życia, który jest przedstawiony na rysunku 2.1:



Rys. 2.1: Cykl życia artykułu

Źródło: opracowanie własne

Weryfikacja artykułu - informacja o tym, czy artykuł powinien zostać opublikowany w serwisie (zaakceptowanie artykułu) albo wymaga jeszcze korekty (odrzucenie artykułu),

Dostępność recenzenta - informacja o tym, czy recenzent może weryfikować artykuły w danym czasie,

Komentarz do artykułu - tekstowa informacja zwrotna zalogowanego użytkownika o danym artykule,

Ocena artykułu - ocena wystawiona przez zalogowanego użytkownika na temat danego artykułu. Ocena ta jest w postaci gwiazdek będąca wartością od 1 do, przy czym wartość 1 oznacza ocenę negatywną i wskazuje, że artykuł posiada rażące błędy i powinien zostać edytowany albo usunięty, a wartość 5 oznacza ocenę pozytywną i wskazuje, że artykuł nie posiada rażących błędów oraz informacje w nim zawarte są niezwykle przydatne dla użytkowników,

Opinia o artykule - oceniająca informacja zwrotna zalogowanego użytkownika na temat danego artykułu, która zawiera tekstowy komentarz oraz ocenę tego artykułu,

Akceptacja opinii o artykule - ocena wystawiona przez zalogowanego użytkownika dotycząca opinii o artykule. Przyjmuje ona dwie wartości: ocena pozytywna (łapka w góre) oznaczająca, że użytkownik zgadza się z opinią i ocena negatywna (łapka w dół) oznaczająca, iż użytkownik nie zgadza się z opinią.

Rola - zestaw uprawnień użytkownika, które określają jego zakres dostępu do systemu oraz określonych funkcji tego systemu. Możliwe jest przypisanie danemu użytkownikowi ról, co jest równoznaczne z uzyskaniem przez tego użytkownika uprawnień zdefiniowanych w tych rolach. Wyróżniono w systemie następujące role:

- **Użytkownik** - każdy użytkownik korzystający z serwisu. Może on jedynie przeglądać zasoby serwisu,
- **Niezalogowany użytkownik** - niezalogowany użytkownik mający możliwość przeglądania zasobów serwisu oraz może się zarejestrować i zalogować,
- **Zalogowany użytkownik** - każdy zarejestrowany i zalogowany użytkownik. Może on przede wszystkim przeglądać i tworzyć artykuły oraz wystawiać komentarze i opinie o artykułach.
- **Autor artykułu** - zalogowany użytkownik mogący edytować oraz usuwać utworzone przez siebie artykuły,
- **Autor komentarza o artykule** - zalogowany użytkownik mogący edytować oraz usuwać utworzone przez siebie komentarze o artykule,
- **Autor opinii o artykule** - zalogowany użytkownik mogący edytować oraz usunąć utworzoną przez siebie opinię o artykule,
- **Autor akceptacji opinii o artykule** - zalogowany użytkownik mogący usunąć utworzoną przez siebie akceptację opinii o artykule,
- **Recenzent (ekspert od technologii)** - zalogowany użytkownik, który otrzymał od administratora rolę recenzenta. Jego głównym zadaniem jest weryfikacja artykułów przed ich opublikowaniem w serwisie. Innym uprawnieniem tego użytkownika jest m.in. zarządzanie klasifikacją technologii.
- **Administrator** (utrzymuje porządek w serwisie) - zalogowany użytkownik, który otrzymał od innego administratora rolę administratora. Odpowiada on przede wszystkim za przydzielanie ról użytkownikom oraz utrzymywanie porządku w serwisie poprzez edytowanie i usuwanie niewłaściwych treści (np. wirusy lub spam). Przy instalacji systemu powinno być tworzone co najmniej jedno konto użytkownika z rolą administratora, gdyż to właśnie administrator odpowiada za przydzielanie ról innym użytkownikom,
- **System** - wbudowane procedury, które umożliwiają automatyczne przypisywanie recenzenta do weryfikacji artykułów oraz automatyczne ponowne przypisanie artykułu do weryfikacji innemu recenzentowi, jeśli upłynął 1-tygodniowy termin weryfikacji artykułu przeprowadzanej przez aktualnego recenzenta.

2.1.2. Wymagania funkcjonalne dla systemu IT Tech

Po zdefiniowaniu pojęć, przystąpiono do określenia wymagań funkcjonalnych dla poszczególnych użytkowników.

Zdefiniowano następujące wymagania funkcjonalne:

- Użytkownik powinien mieć możliwość:
 - wyszukiwania artykułów po następujących polach:
 - * Tytuł,
 - * Autor,
 - * Kategoria technologii,
 - * Technologia,
 - * Dostawca technologii
 - * Zakres dat, w których mieści się data powstania artykułu,
 - * Zakres dat, w których mieści się data modyfikacji artykułu.
 - sortowania artykułów według następujących kryteriów:
 - * Nazwa,
 - * Nazwa technologii,
 - * Autor,

- * Data powstania,
 - * Data modyfikacji,
 - * Średnia ocena,
 - * Liczba opinii.
- wyświetlenia zawartości artykułu,
 - przeglądania komentarzy o artykule,
 - przeglądania opinii o artykule,
 - wyszukiwania technologii po następujących polach:
 - * Nazwa,
 - * Kategoria,
 - * Dostawca,
 - * Zakres dat, w których mieści się data pierwszego wydania,
 - * Zakres dat, w których mieści się data ostatniego wydania.
 - sortowania technologii według następujących kryteriów:
 - * Nazwa,
 - * Kategoria,
 - * Data pierwszego wydania,
 - * Data ostatniego wydania.
- wyświetlenia opisu technologii,
- **Niezalogowany użytkownik** powinien mieć możliwość:
 - zarejestrowania się
 - * Podczas rejestracji użytkownik będzie podawał następujące informacje:
 - Nazwa użytkownika,
 - Pseudonim (nazwa, pod którą użytkownik będzie mógł być rozpoznany w serwisie przez innych użytkowników)
 - Imię,
 - Nazwisko,
 - Adres e-mail,
 - (Opcjonalnie) Avatar,
 - Hasło.
 - * Po rejestracji i zalogowaniu, użytkownik uzyskuje uprawnienia zalogowanego użytkownika, aby uzyskać uprawnienia recenzenta lub administratora, należy w tym celu skontaktować się z administratorem, który będzie w stanie nadać taką rolę,
 - zalogowania się
 - * Podczas logowania użytkownik będzie wpisywał login albo e-mail oraz hasło.
- **Zalogowany użytkownik** powinien mieć możliwość:
 - edycji własnych danych,
 - zmiany hasła,
 - wylogowania się,
 - stworzenia artykułu, przy czym aby artykuł został opublikowany w serwisie, to artykuł ten musi przejść pozytywnie weryfikację przeprowadzaną przez recenzenta,
 - dodania komentarza o artykule,
 - dodania jednej opinii o artykule którego użytkownik nie jest autorem,
 - dodania jednej akceptacji opinii o artykule wystawionej przez innego użytkownika.
- **Autor artykułu** powinien mieć możliwość:

- zarządzania własnymi artykułami w następującym zakresie:
 - * edytowanie, przy czym aby zmieniony artykuł został opublikowany w serwisie, to artykuł ten musi przejść pozytywnie weryfikację przeprowadzaną przez recenzenta,
 - * usuwanie.
- **Autor komentarza o artykule** powinien mieć możliwość:
 - zarządzania własnymi komentarzami o artykule w następującym zakresie:
 - * edytowanie,
 - * usuwanie.
- **Autor opinii o artykule** powinien mieć możliwość:
 - zarządzania własną opinią o artykule w następującym zakresie:
 - * edytowanie,
 - * usuwanie.
- **Autor akceptacji opinii o artykule** powinien mieć możliwość:
 - usunięcia własnej akceptacji opinii o artykule.
- **Recenzent** powinien mieć możliwość:
 - edycji rodziny technologii,
 - ustawienia własnej dostępności,
 - weryfikacji artykułów według następujących założeń:
 - * po przypisaniu artykułu do weryfikacji danemu recenzentowi, recenzent ten będzie miał tydzień na akceptacje lub odrzucenie artykułu,
 - * przy akceptacji artykułu opcjonalne jest załączenie wiadomości (np. drobne uwagi), ale przy odrzuceniu artykułu załączenie wiadomości jest obowiązkowe i powinna ona zawierać informacje o tym, dlaczego artykuł został odrzucony
 - * jeśli recenzent nie zweryfikuje artykułu w ciągu tygodnia, wtenczas artykuł ten zostanie przypisany do weryfikacji innemu recenzentowi.
- **Administrator** powinien mieć możliwość:
 - edycji danych innych użytkowników,F
 - zarządzania artykułami w następującym zakresie:
 - * edytowanie,
 - * usuwanie.
 - usuwania komentarzy o artykule,
 - usuwania opinii o artykule,
 - przypisywania innym użytkownikom ról,
 - usuwania przypisanych ról innych użytkowników.
- **System** powinien mieć co 24 godziny możliwość automatycznego:
 - przypisywania artykułów do weryfikacji dostępnym recenzentom, którzy w ciągu ostatnich 30 dni zrecenzowali najmniejszą liczbę artykułów,
 - wyszukiwania artykułów z przedawnioną weryfikacją,
 - usunięcia przedawnionych weryfikacji artykułów oraz przypisania tych artykułów do ponownej weryfikacji.

2.2. Wymagania niefunkcjonalne

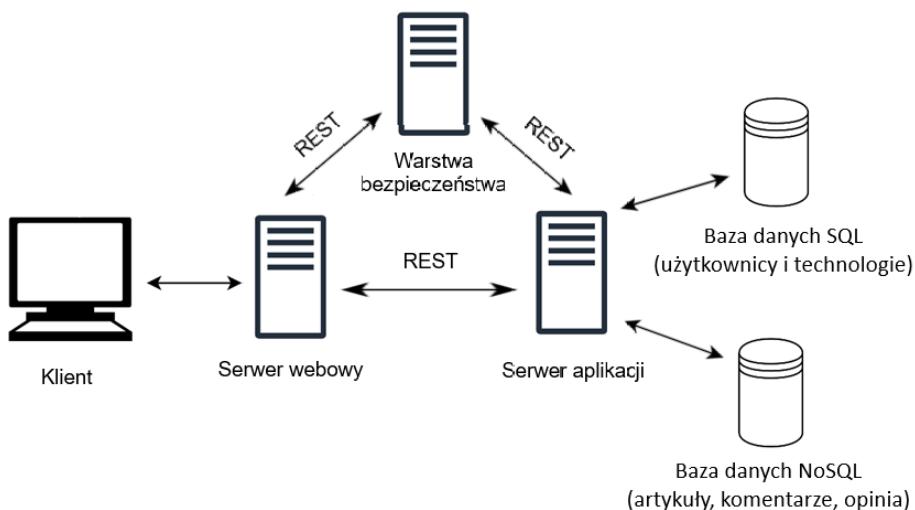
Na podstawie celu oraz zakresu pracy zdefiniowano następujące wymagania niefunkcjonalne:

- Wymagania dotyczące bezpieczeństwa systemu
 - Użytkownik o posiadanej przez siebie roli będzie miał dostęp do systemu jedynie w zakresie uprawnień przypisanych do tej roli,
 - Zewnętrzny serwer do uwierzytelniania i autoryzacji,
 - Dołączane tokenu na okaziciela (ang. *Bearer token*) przy wykonywaniu zapytań do Backendu,
 - Tworzone hasła powinny spełniać następujące kryteria:
 - * Minimalna długość - 8,
 - * Co najmniej jedna mała litera,
 - * Co najmniej jedna duża litera,
 - * Co najmniej jedna cyfra.
- Wykorzystywane technologie i narzędzia
 - Backend - Spring Boot,
 - Frontend - React (główny framework), Typescript (statyczne typowanie), MUI (biblioteka komponentów),
 - Główna baza danych - PostgreSQL,
 - Baza danych artykułów - MongoDB,
 - Warstwa bezpieczeństwa (serwer uwierzytelniania i autoryzacji) - Keycloak,
 - Dokumentacja - LaTeX.

2.3. Architektura systemu

Na podstawie analizy wymagań funkcjonalnych i niefunkcjonalnych, opracowano architekturę systemu. Realizowany system jest rozproszonym serwisem webowym, który w dużym uogólnieniu można opisać jako system typu klient-serwer. Klient wysyła żądanego do serwera aplikacji, a następnie serwer odpowiednio przetwarza otrzymane żądanego i zwraca klientowi odpowiedź, którą klient interpretuje i przedstawia użytkownikowi. Zdecydowano się na rozproszoną architekturę systemu, gdyż taka architektura zapewnia większą niezawodność i skalowalność niż w przypadku systemów scentralizowanych.

Szczegółową architekturę systemu przedstawiono na rysunku 2.2



Rys. 2.2: Architektura systemu

Źródło: opracowanie własne

Zdecydowano się na zaimplementowanie systemu w postaci serwisu webowego głównie dla tego, że taka forma aplikacji zapewnia dużą wygodę użytkowania z powodu braku konieczności jej zainstalowania przez użytkownika. Innym powodem jest to, że serwisy webowe są obecnie najpopularniejszym typem systemu i implementacja takiej aplikacji pozwoli na zdobycie wartościowego doświadczenia.

Komunikacja w systemie przebiega głównie poprzez protokół HTTP (ang. *Hypertext Transfer Protocol*). Protokół ten jest oparty na komunikacji typu klient-serwer, przy czym klientem jest najczęściej przeglądarka. Klientem wysyła żądanie do serwera, a serwer wysyła do klienta odpowiedź. Zawartość wiadomości HTTP można przede wszystkim podzielić na nagłówek (ang. *Header*) oraz ładunek (ang. *Body*). W nagłówku znajdują się pola typu klucz-wartość. Nagłówek żądania różni się od nagłówka odpowiedzi poprzez dostępne opcje, ale mają one również wspólne pola np. informacja o hoście docelowym, czy format wiadomości. W nagłówku żądania można wyróżnić jeszcze np. nagłówek **Authorization**, w którym umieszczane są poświadczenie uwierzytelnienia i autoryzacji. W nagłówku odpowiedzi można za to wyróżnić m.in. status odpowiedzi, który np. informuje o tym, czy żądanie zostało wykonane pomyślnie (np. status 200), czy też wystąpił błąd, a jeśli wystąpił, to z jakiego powodu (np. status 401 oznaczający nieudane uwierzytelnienie użytkownika). Ładunek jest opcjonalnym polem, które stanowi treść wiadomości HTTP i np. w przypadku odpowiedzi może zawierać listę wyszukanych artykułów.

Komunikacja w systemie jest zgodna ze stylem architektury oprogramowania REST (ang. *Representational state transfer*). REST jest to sposób i format w jaki komunikuje się klient z serwerem. Serwer udostępnia klientowi punkty końcowe (end-pointy), do których klient może wysłać żądania HTTP przesyłając przy tym jakieś dane np. tytuł wyszukiwanego artykułu. W skrócie komunikacja REST odznacza się następującymi cechami:

- bezstanowość (nie przechowuje informacji o poprzednich wymianach wiadomości),
- architektura klient-serwer,
- jednolity interfejs komunikacyjny – dzięki temu możliwe jest np. komunikowanie się systemów zaimplementowanych w różnych językach programowania, czy zainstalowanych na różnych platformach (np. mobilna),
- wykorzystywanie protokołu HTTP.

2.3.1. Baza danych

Baza danych to warstwa systemu odpowiedzialna za przechowywanie danych. Założono, że w systemie zostaną wykorzystane dwie bazy danych różnego typu: obiektowo-relacyjna (SQL) oraz nierelacyjna oparta na dokumentach (NoSQL).

Jako główną bazę danych wybrano bazę obiektowo-relacyjną, gdyż bazy relacyjne charakteryzują się tym, że zapewniają integralność danych poprzez tworzenie relacji np. poprzez powiązanie tabel kluczem obcym. W realizowanym systemie wykryto wiele przypadków, w których można utworzyć relacje np. właściciel artykułu jest powiązaniem między encjami użytkownika i artykułu oraz historia modyfikacji technologii jest dołączona do danej technologii. Zdecydowano się na obiektową warstwę bazy danych, gdyż upraszcza ona pracę z kilkoma wykorzystywanymi narzędziami (np. framework Hibernate języka Java), które są oparte na programowaniu obiektowym.

Artykuły mają złożoną zagnieźdzoną strukturę, zawierającą rozdziały, podrozdziały, komentarze oraz opinie. Nierelacyjna baza dokumentowa wydaje się być idealnym pomysłem do przechowywania takich informacji. Baza ta pozwala na przechowywanie struktur typu klucz-wartość, co zapewnia dużą elastyczność w przechowywaniu danych, gdyż pozwala na m.in. przechowywanie zagnieźdzonych struktur oraz umożliwia pobranie danych w postaci zagnieźdzonej struktury nawet poprzez jedno zapytanie. Dla porównania w przypadku relacyjnej bazy danych należałoby wykonać wiele klauzuli JOIN aby pobrać te same dane i przez co traciłoby się na efektywności.

2.3.2. Aplikacja serwerowa - Backend

Backend jest odpowiedzialny za przyjmowanie żądań od klienta, odpowiednie przetwarzanie tych żądań, wykonywanie pewnych operacji na danych przechowywanych w bazie danych na podstawie otrzymanych przez klienta danych i przekazywanie klientowi adekwatnej odpowiedzi. Backend udostępnia klientowi punkty końcowe (ang. *API endpoints*) do których klient może przesyłać żądanie. Punkty końcowe posiadają adresy URL np. <http://localhost:9000/user>. Adresy URL składają się nazwy protokołu, nazwy hosta docelowego, portu docelowego oraz reszty, która identyfikuje punkt końcowy oraz jego przeznaczenie. Ważnym elementem w przypadku backendu jest również wysłanie klientowi odpowiedniego statusu odpowiedzi oraz ewentualne dołączenie informacji zwrotnej. Dzięki temu klient będzie mógł odpowiednio zareagować na otrzymaną odpowiedź (np. w przypadku otrzymania negatywnej odpowiedzi, klient wyświetli użytkownikowi komunikat o błędzie, a w przypadku otrzymania odpowiedzi pozytywnej, klient wyświetli komunikat o sukcesie). Punkty końcowe backendu są chronione przez warstwę bezpieczeństwa. Klient będzie mógł wysłać żądanie na dany endpoint tylko wtedy, gdy użytkownik po stronie klienta będzie miał wymagane przez ten endpoint uprawnienia (role). Warstwa ta jest ściśle powiązana z warstwą bazy danych.

2.3.3. Klient - Frontend

Frontend jest odpowiedzialny za wysyłanie żądań do warstwy backendowej i następnie odpowiednie przetwarzanie oraz wyświetlanie danych otrzymanych w odpowiedzi od backendu. W systemie tym klientem jest strona internetowa renderowana po stronie użytkownika. Klient jest aplikacją typu SPA (ang. *Single Page Application*), co oznacza, że aplikacja składa się z jednej strony HTML, a jej zawartość jest zmieniana dynamicznie poprzez JavaScript. Zdecydowano się na aplikację typu SPA, a nie MPA (ang. *Multi Page Application*) głównie dlatego, że aplikacje SPA działają szybciej z powodu tylko jednej strony HTML. Kolejnymi kryteriami na korzyść SPA były łatwiejsze wdrożenie, gdyż wystarczą do tego jedynie 3 pliki (HTML, CSS i JavaScript), łatwiejsze wprowadzenie dynamicznej zawartości oraz ułatwione tworzenie aplikacji.

Frontend jest również chroniony przez warstwę bezpieczeństwa. Użytkownik będzie miał dostęp do podstron oraz funkcji systemu jedynie w zakresie posiadanych przez niego uprawnień (ról).

2.3.4. Serwer webowy

Warstwa odpowiedzialna za dostarczenie treści strony WWW w odpowiedzi na żądania HTTP użytkownika. Użytkownik jest połączony z tym serwerem poprzez swoją przeglądarkę. Innymi zadaniami tej warstwy są przekierowywanie żądań oraz zapewnienie bezpieczeństwa poprzez m.in. możliwość określenia dozwolonych nagłówków w wysyłanych do tego serwera żądaniach.

2.3.5. Warstwa bezpieczeństwa

Warstwa bezpieczeństwa jest osobnym serwerem, który odpowiada za uwierzytelnienie i autoryzację. Warstwa ta zabezpiecza zarówno warstwę kliencką, jak i serwer aplikacji. Warstwa kliencka jest chroniona poprzez blokowanie niektórych podstron, jak i funkcji systemu, które nie powinny być dostępne dla danego użytkownika. Mechanizmem bezpieczeństwa dla aplikacji serwerowej jest wymóg dostarczania prawidłowego tokenu dostępu (ang. *Access token*) oraz posiadania odpowiednich uprawnień przy wysyłaniu przez klienta żądania do zabezpieczonego punktu końcowego tego serwera.

Klient może uzyskać token dostępu poprzez zalogowanie się do serwera bezpieczeństwa. Logowanie może być wykonane poprzez wysłanie przez klienta żądania z danymi do logowania na odpowiedni endpoint serwera bezpieczeństwa. W przypadku pomyślnego logowania, klient otrzymuje od serwera bezpieczeństwa m.in. wygenerowane tokeny dostępu oraz odświeżania wraz z datami ważności tych tokenów. Token dostępu jest później dostarczany przez klienta w nagłówku wiadomości w polu przeznaczonym do autoryzacji: *Authorization: Bearer <token>*. Przed wysłaniem żądania do backendu klient sprawdza, czy token dostępu stracił swoją ważność (zwykle kilka minut) i jeśli token ten stracił swoją ważność, to klient wysyła żądanie z dołączonym tokenem odświeżania o wygenerowanie nowego tokenu dostępu, do którego dołączony będzie również aktualny token odświeżania. Token odświeżania może stracić swoją ważność jedynie w przypadku wylogowania albo nieaktywności użytkownika. Opisany proces uwierzytelnienia nazywa się uwierzytelnianiem na okaziciela (ang. *Bearer authentication*). Zastosowany serwer bezpieczeństwa przeprowadza autoryzację użytkowników w oparciu o ich role.

2.4. Diagramy przypadków użycia

Na podstawie zdefiniowanych wcześniej wymagań funkcjonalnych, wykonano identyfikację aktorów oraz sporządzono diagramy przypadków użycia. Do stworzenia diagramów wykorzystano program Visual Paradigm Standard.

2.4.1. Identyfikacja aktorów

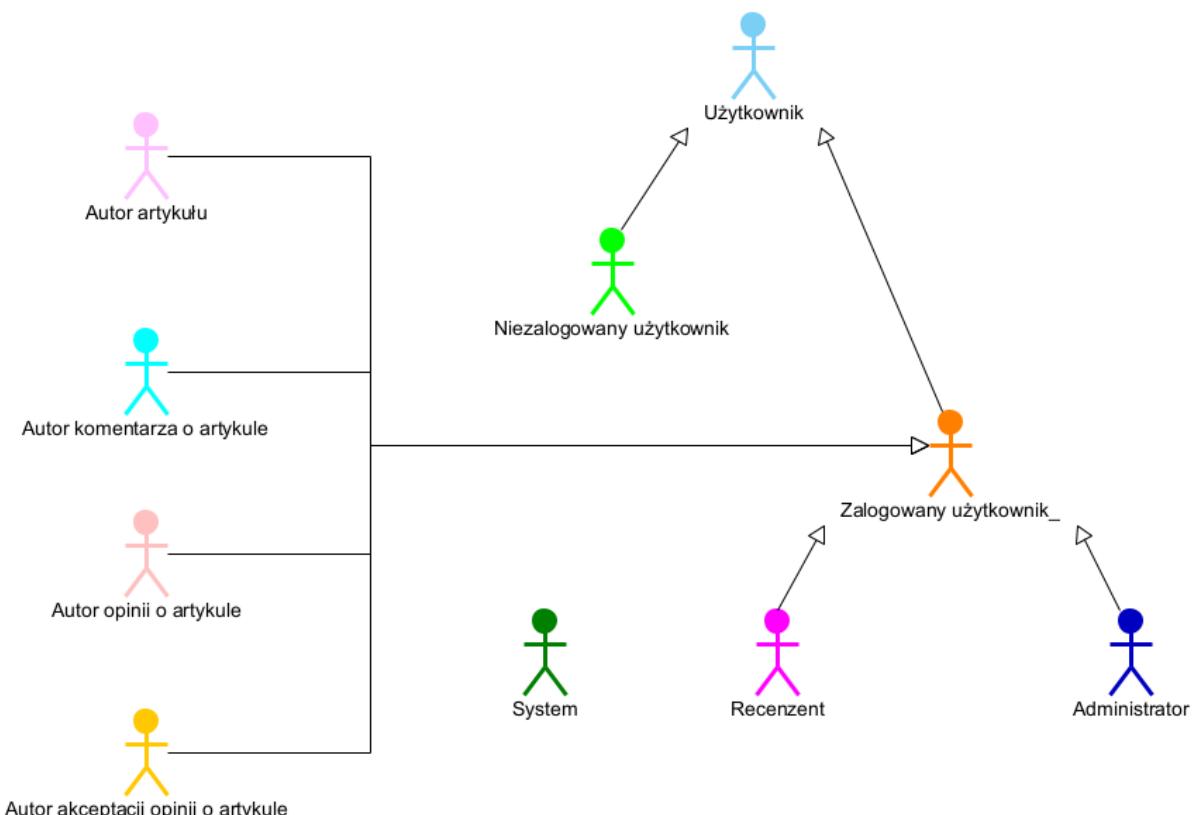
Na podstawie analizy wymagań funkcjonalnych zdefiniowano następujących aktorów:

- Użytkownik,
- Niezalogowany użytkownik,
- Zalogowany użytkownik,
- Autor artykułu,
- Autor komentarza o artykule,

- Autor opini o artykule,
- Autor akceptacji opinii o artykule,
- Recenzent,
- Administrator,
- System.

2.4.2. Diagramy przypadków użycia

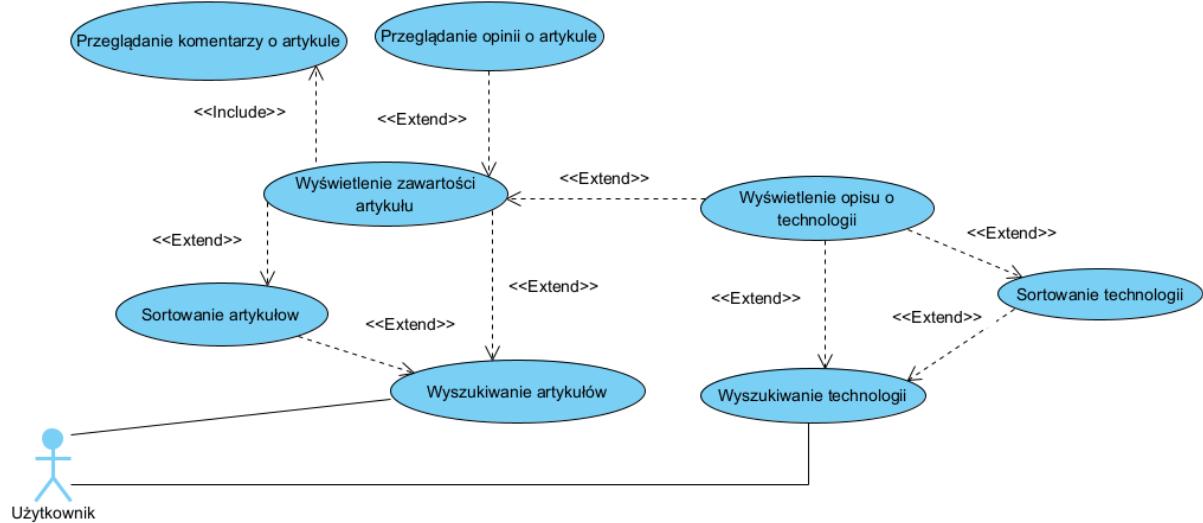
Na podstawie zdefiniowanych wcześniej wymagań funkcjonalnych oraz zidentyfikowanych aktorów, sporządzono diagramy przypadków użycia. Zdecydowano się na zastosowanie dziedziczenia między aktorami, gdyż opracowane role charakteryzują się tym, że posiadają wiele wspólnych uprawnień (przypadków użycia) i działanie to pozwala na zmniejszenie złożoności systemu. Dziedziczenie aktorów zostało przedstawione na rysunku 2.3.



Rys. 2.3: Dziedziczenie między aktorami

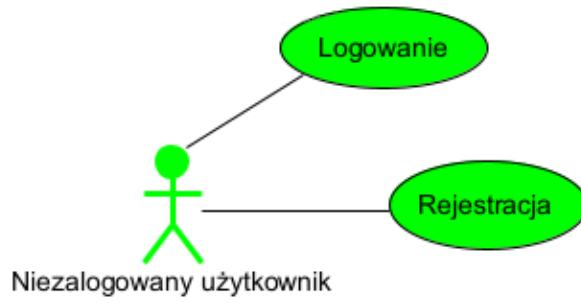
Źródło: opracowanie własne

Z powodu dużej złożoności systemu pod względem liczby wymagań funkcjonalnych i liczby ról, dla zachowania czytelności, stworzono osobne diagramy dla każdego aktora. Poniżej znajdują się diagramy przypadków użycia dla poszczególnych aktorów.



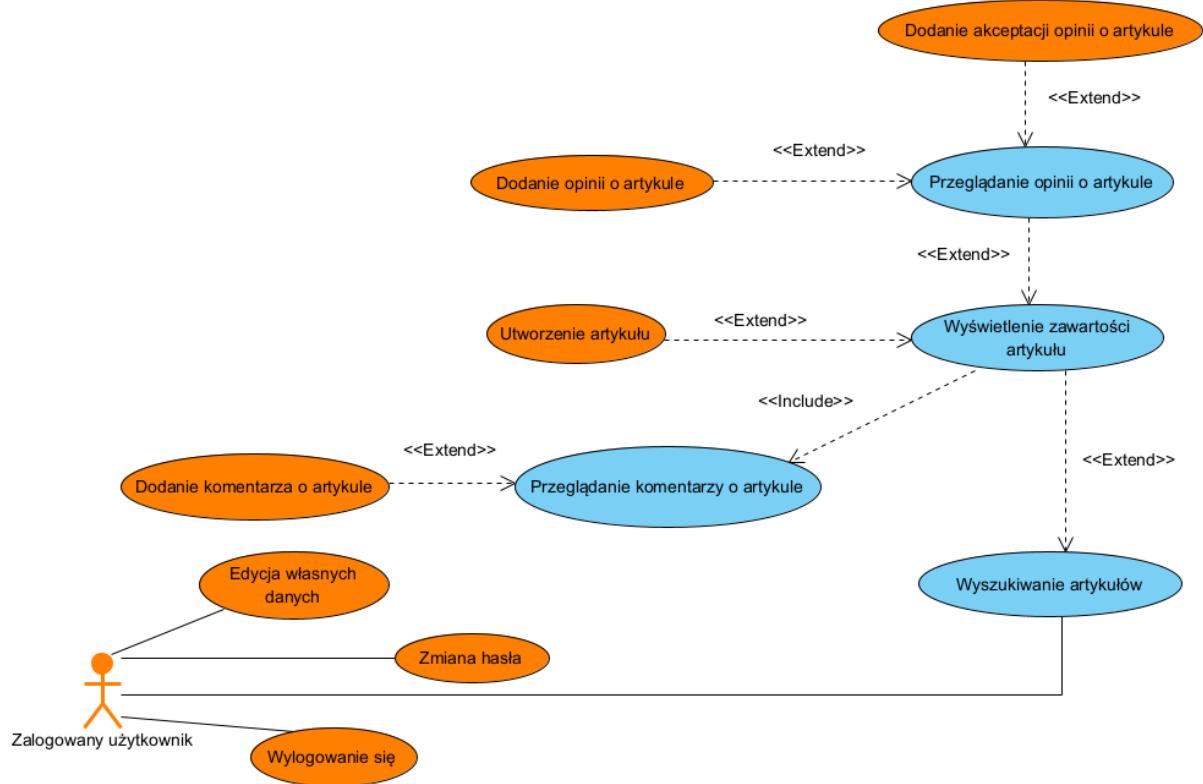
Rys. 2.4: Diagram przypadków użycia dla Użytkownika

Źródło: opracowanie własne



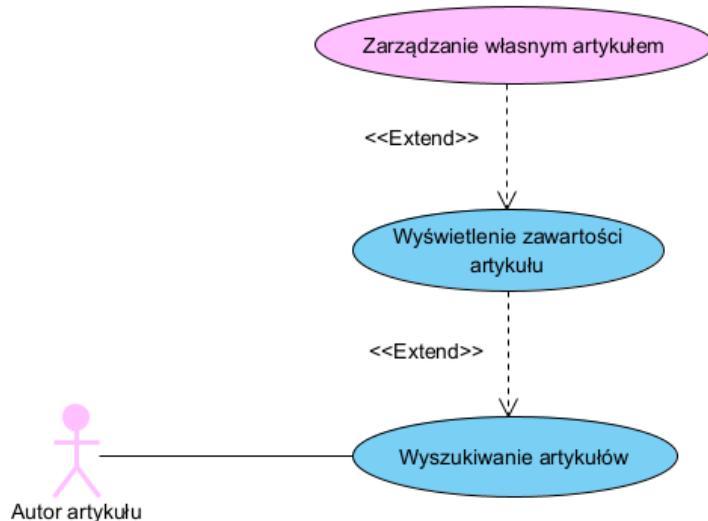
Rys. 2.5: Diagram przypadków użycia dla Niezalogowanego użytkownika

Źródło: opracowanie własne



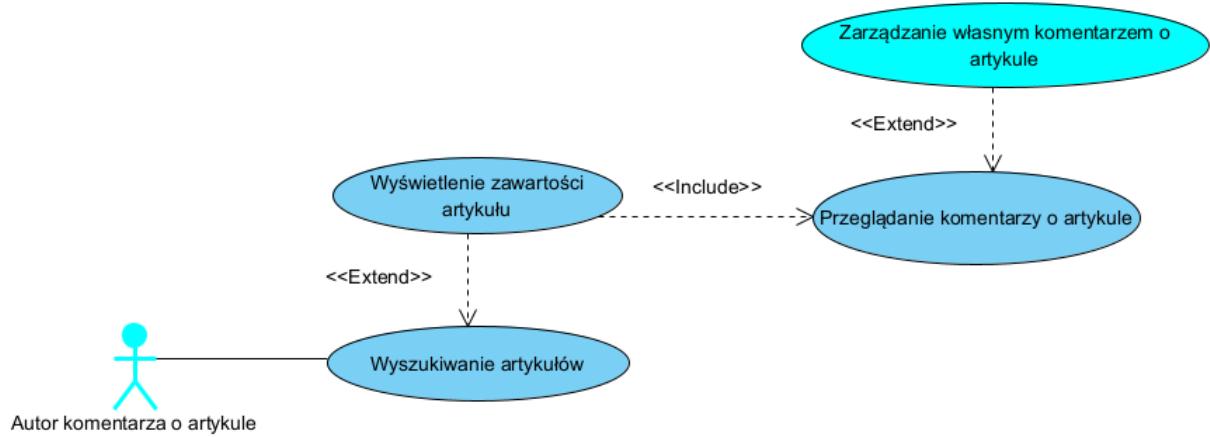
Rys. 2.6: Diagram przypadków użycia dla Zalogowanego użytkownika

Źródło: opracowanie własne



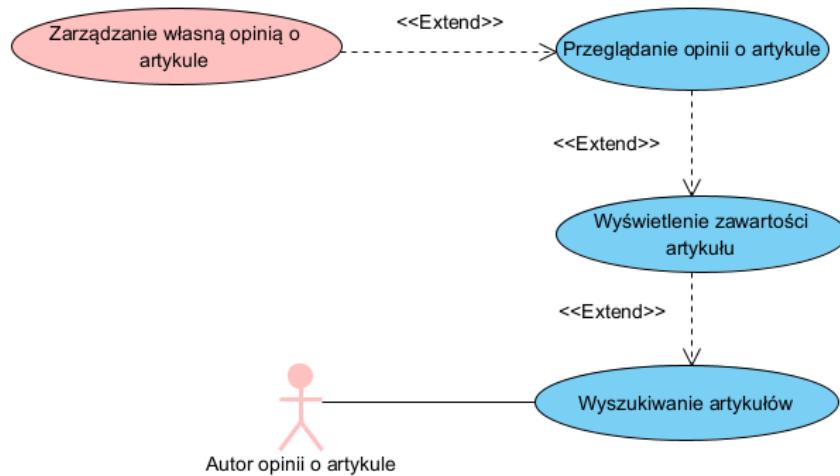
Rys. 2.7: Diagram przypadków użycia dla Autora artykułu

Źródło: opracowanie własne



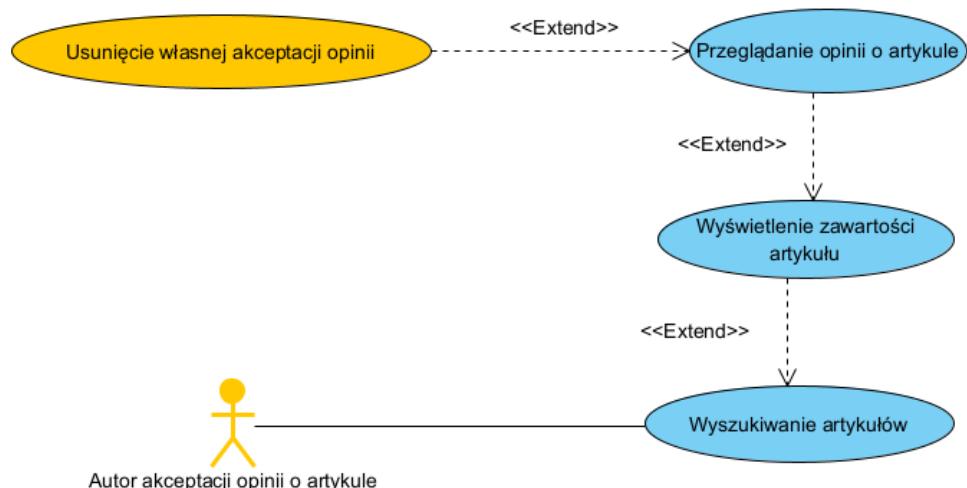
Rys. 2.8: Diagram przypadków użycia dla Autora komentarza o artykule

Źródło: opracowanie własne



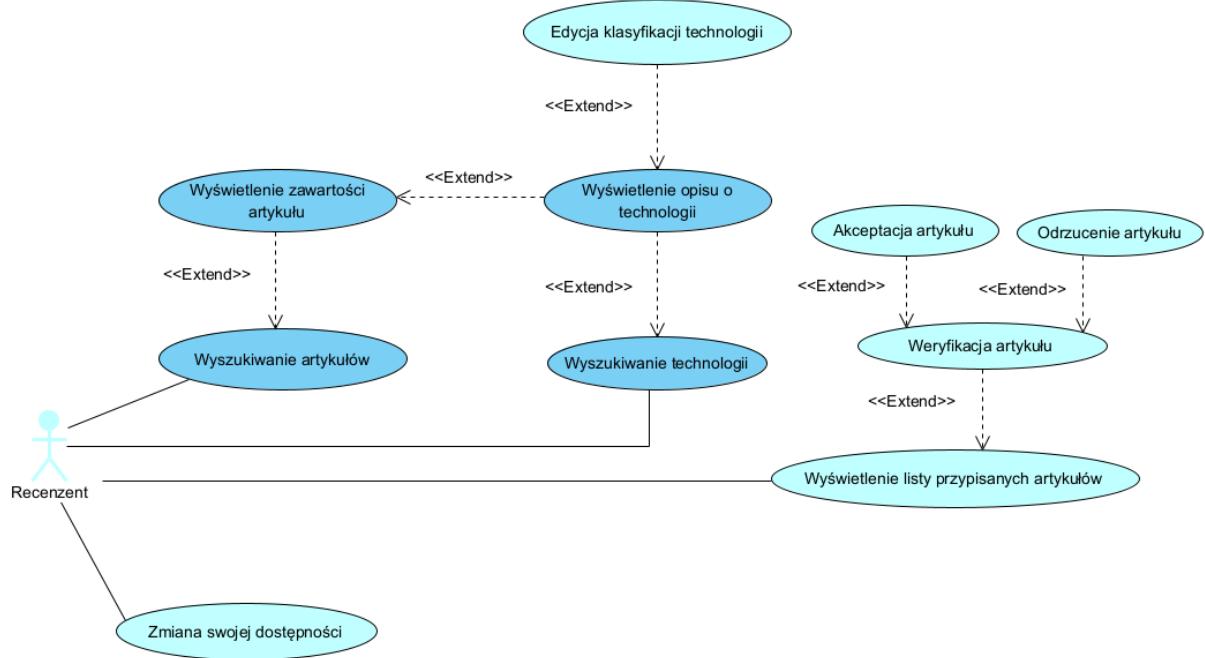
Rys. 2.9: Diagram przypadków użycia dla Autora opinii o artykule

Źródło: opracowanie własne



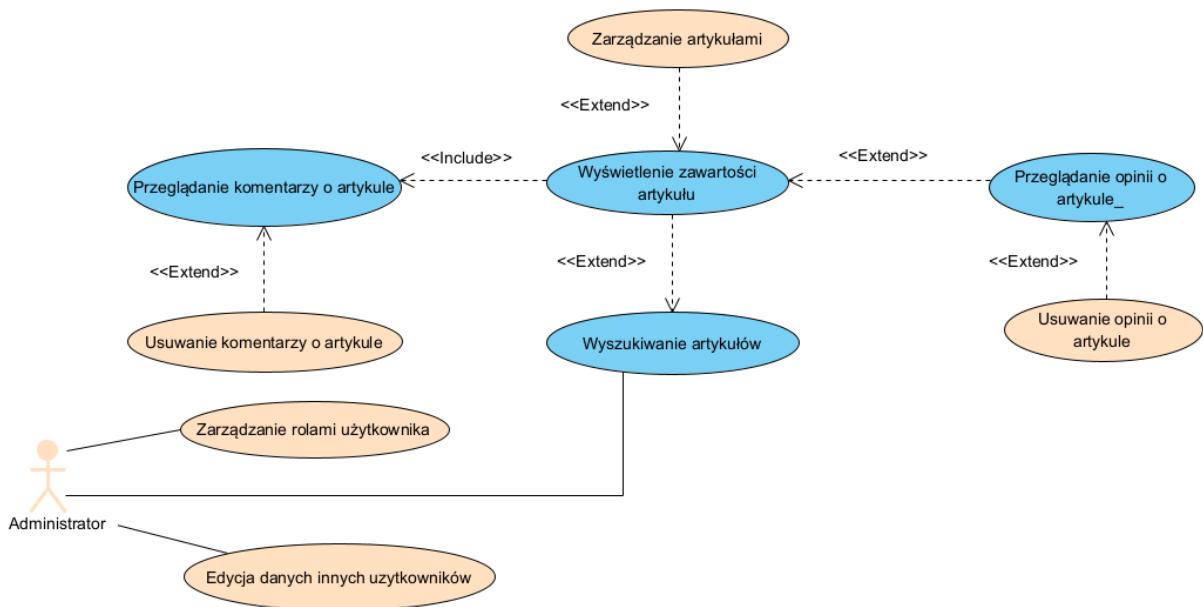
Rys. 2.10: Diagram przypadków użycia dla Autora akceptacji opinii artykule

Źródło: opracowanie własne



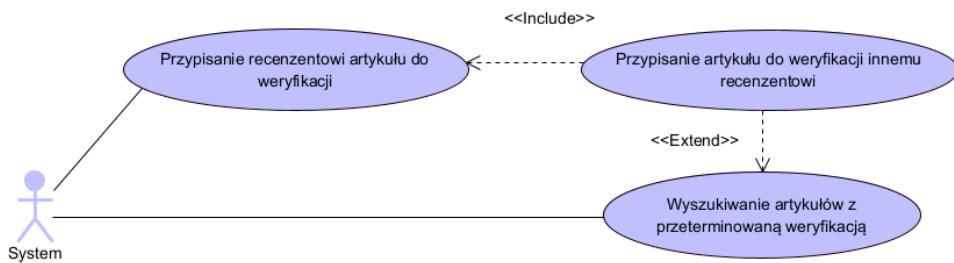
Rys. 2.11: Diagram przypadków użycia dla Recenzenta

Źródło: opracowanie własne



Rys. 2.12: Diagram przypadków użycia dla Administratora

Źródło: opracowanie własne



Rys. 2.13: Diagram przypadków użycia dla Systemu

Źródło: opracowanie własne

2.4.3. Opisy przypadków użycia

Dla każdego przypadku użycia zdefiniowanego w poprzednim punkcie, zrobiono szczegółowe opisy w postaci tabel, które składają się z następujących informacji:

- Nazwa,
- Cel,
- Występujący aktorzy,
- Warunki początkowe,
- Warunki końcowe,
- Przebieg,
- Alternatywne przebiegi.

Dla każdego przypadku użycia zrobiono dodatkowo prototypy ekranu użytkownika, aby zaplanować wygląd interfejsu użytkownika oraz aby opisy przypadków użycia były bardziej zrozumiałe.

Tab. 2.1: Opis przypadku użycia - Rejestracja

Nazwa:	Rejestracja
Cel:	Utworzenie konta zwykłego użytkownika
Aktorzy:	Niezalogowany użytkownik
Warunki początkowe:	Użytkownik jest niezalogowany
Warunki końcowe:	Założenie konta zwykłego użytkownika
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Rejestracja” na nagłówku strony (Rys. 2.14), 2. Aplikacja wyświetla formularz do rejestracji (Rys. 2.15), 3. Użytkownik wypełnia pola w formularzu, 4. Użytkownik kliką przycisk „Zarejestruj”, 5. Serwis przeprowadza walidację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 6. Po pozytywnej walidacji danych, aplikacja wyświetla regulamin serwisu, 7. Użytkownik przegląda cały regulamin i go akceptuje poprzez kliknięcie w pole wyboru „Akceptuję regulamin” (Rys. 2.16), 8. Serwis tworzy nowo konto.

Alternatywny przebieg:	<p>Użytkownik anuluje rejestrację</p> <p>3a. Użytkownik kliką przycisk „X” albo kliką w obszar poza formularzem,</p> <p>4a. Wyłączenie formularza.</p>
Alternatywny przebieg:	<p>Wprowadzone przez użytkownika dane nie przeszły walidacji</p> <p>6b. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie,</p> <p>7b. Powrót do punktu 3.</p>
Alternatywny przebieg:	<p>Użytkownik nie akceptuje regulaminu</p> <p>7c. Użytkownik nie akceptuje regulaminu poprzez kliknięcie w pole wyboru „Nie akceptuje regulaminu”, kliką przycisk „X” albo kliką w obszar poza formularzem,</p> <p>8c. Wyłączenie formularza.</p>

Logowanie

Rejestracja

Artykuły

Technologie

Rys. 2.14: Nagłówek strony niezalogowanego użytkownika

Źródło: opracowanie własne

Formularz rejestracyjny z polem do dodania avatara.

Formularz rejestracyjny (Rejestracja) składa się z następujących elementów:

- Pole imię (obowiązkowe, oznaczone gwiazdką *)
- Pole nazwisko (obowiązkowe, oznaczone gwiazdką *)
- Pole nazwa użytkownika (obowiązkowe, oznaczone gwiazdką *)
- Pole pseudonim (obowiązkowe, oznaczone gwiazdką *)
- Pole e-mail (obowiązkowe, oznaczone gwiazdką *)
- Pole hasło (obowiązkowe, oznaczone gwiazdką *)
- Pole powtórz hasło (obowiązkowe, oznaczone gwiazdką *)
- Pole do dodania avatara (zapisane jako "K")
- Pole "Dodaj awatar" z przyciskiem "+"
- Konferencyjna przycisk "Zarejestruj"

* - pole wymagane

Rys. 2.15: Formularz rejestracji

Źródło: opracowanie własne

X

Regulamin

1. Ogólne zasady

Użytkownicy nie powinni umieszczać w serwisie treści zawierających wulgaryzmy, będących spamem lub zawierających linki do niebezpiecznych stron (wirusy itp.). W pierwszym przypadku treści te będą usuwane przez administratorów, a w przypadku skrajnych lub powtarzających się takich działań, konto użytkownika odpowiadającego za tego typu treści może zostać zablokowane na jakiś czas (zwykle 1 dzień) lub na stałe.

2. Artykuły

Użytkownicy nie powinni publikować w artykułach informacji nieprawdziwych, nieaktualnych, tekst nie powinien zawierać błędów, zamieszczane rysunki powinny być czytelne, a kod powinien być wstawiony za pomocą pól specjalnie do tego przeznaczonych.

W przypadku niezastosowania się użytkownika do powyższych reguł, artykuł zostanie zedytorowany przez administratora w przypadku wymogu drobnych poprawek, a w przypadku dużych zmian, artykuł taki zostanie wycofany z serwisu, autor artykułu będzie musiał poprawić błędy i zmieniony artykuł przejdzie jeszcze raz weryfikację przez automatycznie wybranego recenzenta.

Akceptuję regulamin

Nie akceptuję regulaminu

Rys. 2.16: Formularz służący do akceptacji regulaminu

Źródło: opracowanie własne

Tab. 2.2: Opis przypadku użycia - Logowanie

Nazwa:	Logowanie
Cel:	Zalogowanie się użytkownika
Aktorzy:	Niezalogowany użytkownik
Warunki początkowe:	Użytkownik jest niezalogowany
Warunki końcowe:	Zalogowanie się użytkownika

Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Logowanie” na nagłówku strony (Rys. 2.14), 2. Aplikacja wyświetla formularz do logowania (Rys. 2.17), 3. Użytkownik wypełnia pola w formularzu, 4. Użytkownik kliką przycisk „Zaloguj”, 5. Serwis przeprowadza walidację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 6. Po pozytywnej walidacji danych, użytkownik zostaje zalogowany do systemu i będzie mógł korzystać z funkcji systemu w zakresie adekwatnym do jego uprawnień. Zmienia się wygląd nagłówka strony (Rys. 2.18),
Alternatywny przebieg:	<p>Użytkownik anuluje logowanie</p> <ol style="list-style-type: none"> 3a. Użytkownik kliką przycisk „X” lub kliką w obszar poza formularzem, 4a. Wyłączenie formularza.
Alternatywny przebieg:	<p>Wprowadzone przez użytkownika dane nie przeszły walidacji</p> <ol style="list-style-type: none"> 6b. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 7b. Powrót do punktu 3.
Alternatywny przebieg:	<p>Hasło użytkownika zostało wcześniej zresetowane (puste hasło)</p> <ol style="list-style-type: none"> 7c. Aplikacja wyświetla formularz do ustawienia hasła (Rys. 2.19) 8c. Użytkownik podaje hasło, 9c. Użytkownik kliką przycisk „Zapisz”, 10c. Serwis sprawdza, czy podano hasło oraz jeśli podano, to czy jest ono zgodne z narzuconym rozmiarem i formatem, 11c. Po pozytywnej walidacji hasła, hasło użytkownika zostaje zmienione.
Alternatywny przebieg:	<p>Wprowadzone hasło po resecie hasła nie przeszło walidacji</p> <ol style="list-style-type: none"> 11ca. Aplikacja zmienia kolor obramowania pola z haślem, na czerwono oraz wyświetla pod tym polem adekwatny komunikat o błędzie 12ca. Powrót do punktu 8c,

Formularz logowania z nagłówkiem "Logowanie". W środku znajdują się dwie pola do wpisania danych: "Nazwa użytkownika" i "Hasło", oraz niebieska przycisk "Zaloguj". W prawym górnym rogu jest ikona zamknięcia (X).

Rys. 2.17: Formularz logowania

Źródło: opracowanie własne

Rys. 2.18: Nagłówek strony zalogowanego użytkownika

Źródło: opracowanie własne

Formularz do ustawiania hasła z nagłówkiem "Ustawianie hasła". W środku znajdują się pole do wpisania "Hasło" i niebieska przycisk "Zapisz".

Rys. 2.19: Formularz do ustawienia hasła po zresetowaniu hasła

Źródło: opracowanie własne

Tab. 2.3: Opis przypadku użycia - Wylogowanie

Nazwa:	Wylogowanie
Cel:	Wylogowanie się użytkownika
Aktorzy:	Zalogowany użytkownik
Warunki początkowe:	Użytkownik jest zalogowany
Warunki końcowe:	Wylogowanie użytkownika z serwisu

Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką na swój awatar dostępny na nagłówku strony aby wyświetlić opcje dotyczące konta (Rys. 2.20), 2. Użytkownik kliką przycisk „Wyloguj się”, 3. Aplikacja wyświetla komunikat o wylogowaniu się, 4. Następuje wylogowanie użytkownika oraz zmienia się wygląd nagłówka strony (Rys. 2.14).
-----------	---



Rys. 2.20: Nagłówek strony zalogowanego użytkownika z otwartymi opcjami dotyczącymi konta

Źródło: opracowanie własne

Tab. 2.4: Opis przypadku użycia - Edycja własnych danych

Nazwa:	Edycja własnych danych
Cel:	Zapisanie danych użytkownika zgodnie z wprowadzonymi przez niego zmianami
Aktorzy:	Zalogowany użytkownik
Warunki początkowe:	Użytkownik jest zalogowany
Warunki końcowe:	Zapisanie zmienionych danych użytkownika
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką na swój awatar dostępny na nagłówku strony aby wyświetlić opcje dotyczące konta (Rys. 2.20), 2. Użytkownik kliką przycisk „Dane użytkownika”, 3. Aplikacja wyświetla panel z danymi użytkownika (Rys 2.21), 4. Użytkownik kliką przycisk „Edytuj dane”, 5. Serwis wyświetla formularz edycji danych (Rys 2.22), 6. Użytkownik zmienia dane w formularzu, 7. Użytkownik kliką przycisk „Zapisz”, 8. Serwis przeprowadza walidację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 9. Po pozytywnej walidacji danych, aplikacja zapisuje zmienione dane użytkownika.

Alternatywny przebieg:	<p>Użytkownik anuluje edycję danych</p> <p>6a. Użytkownik kliką przycisk „Anuluj”, 7a. Aplikacja wyświetla panel z danymi użytkownika (Rys 2.21).</p>
Alternatywny przebieg:	<p>Wprowadzone przez użytkownika dane nie przeszły walidacji</p> <p>9b. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 10b. Powrót do punktu 6.</p>

Dane użytkownika

Imię	Kamil	
Nazwisko	Dywan	
Nazwa użytkownika	kamil.dywan	Avatar 
Pseudonim	kamil_dywan	
E-mail	kamil.dywan@mail.pl	

[Edytuj dane](#)

Rys. 2.21: Panel z danymi użytkownika

Źródło: opracowanie własne

The screenshot shows a user profile editing interface. At the top, there's a header with 'IT Tech Technologie IT' and a user icon labeled 'kamil_dywan'. Below the header, there are two tabs: 'Artykuły' and 'Technologie'. The main area is titled 'Dane użytkownika' and contains fields for editing personal information:

- Imię: Kamil
- Nazwisko: Dywan
- Nazwa użytkownika: kamil.dywan
- Pseudonim: kamil_dywan
- E-mail: kamil.dywan@mail.com

There's also an 'Avatar' section showing a blue square placeholder with a white letter 'K' and a file input field 'avatar.png +'. At the bottom, there are two buttons: 'Zapisz' (Save) and 'Anuluj' (Cancel).

Rys. 2.22: Panel umożliwiający użytkownikowi edycję własnych danych

Źródło: opracowanie własne

Tab. 2.5: Opis przypadku użycia - Zmiana hasła

Nazwa:	Zmiana hasła
Cel:	Zmiana hasła zgodnie z wprowadzoną przez użytkownika wartością
Aktorzy:	Zalogowany użytkownik
Warunki początkowe:	Użytkownik jest zalogowany
Warunki końcowe:	Zapisanie zmienionego hasła użytkownika
Przebieg:	<ol style="list-style-type: none"> Użytkownik klikna na swój avatar dostępny na nagłówku strony aby wyświetlić opcje dotyczące konta (Rys. 2.20), Użytkownik klikna przycisk „Zmień hasło”, Aplikacja wyświetla formularz do zmiany hasła (Rys. 2.23) Użytkownik podaje aktualne oraz nowe hasło, Użytkownik klikna przycisk „Zapisz”, Serwis sprawdza, czy podano hasła, czy podane aktualne hasło jest prawidłowe oraz czy podane nowe hasło spełnia wymagania pod względem narzuconego rozmiaru i formatu danych, Po pozytywnej walidacji hasło użytkownika zostaje zmienione.
Alternatywny przebieg:	Użytkownik anuluje zmianę hasła 4a. Użytkownik klikna przycisk „Anuluj”, 5a. Aplikacja wyłącza formularz do zmiany hasła.

Alternatywny przebieg:	<p>Wprowadzone przez użytkownika dane nie przeszły walidacji</p> <p>7b. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie,</p> <p>8b. Powrót do punktu 4.</p>
------------------------	---

Formularz zmiany hasła zatytułowany "Zmiana hasła". W formularzu znajdują się dwie pola tekstowe: "Aktualne hasło" i "Nowe hasło", oba z niebieskimi ramkami. Poniżej nich znajdują się dwa niebieskie przyciski: "Zapisz" i "Anuluj".

Rys. 2.23: Formularz zmiany hasła

Źródło: opracowanie własne

Tab. 2.6: Opis przypadku użycia - Edycja danych innych użytkowników

Nazwa:	Edycja danych innych użytkowników
Opis:	Zapisanie danych innego użytkownika zgodnie z wprowadzonymi zmianami
Aktorzy:	Administrator
Warunki początkowe:	Użytkownik posiadający rolę administratora jest zalogowany
Warunki końcowe:	Zapisanie zmienionych danych innego użytkownika

Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Zarządzanie użytkownikami” znajdujący się w menu głównym strony (Rys. 2.24), 2. Aplikacja wyświetla podstronę z wyszukiwarką użytkowników, 3. Użytkownik wpisuje nazwę użytkownika (np. kamil), 4. Użytkownik kliką przycisk „Szukaj”, 5. Serwis wyszukuje użytkowników, 6. Aplikacja wyświetla listę wyszukanych użytkowników (Rys 2.25) 7. Użytkownik kliką na jednego z innych użytkowników, 8. Aplikacja wyświetla panel z danymi wybranego użytkownika (Rys 2.26), 9. Użytkownik kliką przycisk „Edytuj dane”, 10. Serwis wyświetla formularz edycji danych (Rys 2.27), 11. Użytkownik zmienia dane w formularzu, 12. Użytkownik kliką przycisk „Zapisz”, 13. Serwis przeprowadza validację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 14. Po pozytywnej walidacji danych, aplikacja zapisuje zmienione dane innego użytkownika.
Alternatywny przebieg:	<p>Użytkownik resetuje hasło innego użytkownika</p> <ol style="list-style-type: none"> 9a. Użytkownik kliką przycisk „Reset hasła”, 10a. Aplikacja zastępuje hasło wybranego użytkownika pustym ciągiem znaków, 11a. Powrót do punktu 8.
Alternatywny przebieg:	<p>Użytkownik anuluje edycję danych</p> <ol style="list-style-type: none"> 11b. Użytkownik kliką przycisk „Anuluj”, 12b. Aplikacja wyświetla panel z danymi użytkownika (Rys. 2.26).
Alternatywny przebieg:	<p>Wprowadzone przez użytkownika dane nie przeszły walidacji</p> <ol style="list-style-type: none"> 14c. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 15c. Powrót do punktu 11.



Rys. 2.24: Nagłówek strony administratora

Źródło: opracowanie własne

The screenshot shows a search interface titled "Wyszukiwanie użytkowników". At the top, there is a search input field containing "kamil", a blue "Szukaj" button, and a navigation bar with icons for <, 1, 2, ..., 8, >. Below this, a table lists search results:

Nazwa użytkownika	Pseudonim	Imię i nazwisko
kamil.dywan	kamil_dywan	Kamil Dywan
kamil.nowak	kamil_nowak	Kamil Nowak
kamil.kowalski	kamil_kowalski	Kamil Kowalski

Rys. 2.25: Panel do wyszukiwania użytkowników

Źródło: opracowanie własne



Dane użytkownika

Imię	Kamil	
Nazwisko	Dywan	
Nazwa użytkownika	kamil.dywan	Avatar
Pseudonim	kamil_dywan	
E-mail	kamil.dywan@mail.com	

[Reset hasła](#)[Edytuj dane](#)

Role użytkownika

Recenzent	<input checked="" type="checkbox"/>
Administrator	<input type="checkbox"/>

Rys. 2.26: Panel z danymi użytkownika widziany przez administratora

Źródło: opracowanie własne

Dane użytkownika

Imię	Kamil
Nazwisko	Dywan
Nazwa użytkownika	kamil.dywan
Pseudonim	kamil_dywan
E-mail	kamil.dywan@mail.com

Avatar

K

avatar.png +

Reset hasła

Zapisz Anuluj

Role użytkownika

Recenzent	<input checked="" type="checkbox"/>
Administrator	<input type="checkbox"/>

Rys. 2.27: Panel umożliwiający administratorowi edycję danych użytkownika

Źródło: opracowanie własne

Tab. 2.7: Opis przypadku użycia - Zarządzanie rolami użytkownika

Nazwa:	Zarządzanie rolami innego użytkownika
Cel:	Zmiana przypisanych ról innego użytkownika
Aktorzy:	Administrator
Warunki początkowe:	Wyświetlony panel z danymi innego użytkownika (Rys 2.26)
Warunki końcowe:	Zmiana przypisanych ról innego użytkownika

Przebieg:	<p>Inny użytkownik posiada wybraną rolę (np. Recenzent na Rys 2.26)</p> <ol style="list-style-type: none"> 1. Użytkownik kliką na pole wyboru przy wybranej roli (jeśli przy roli jest zaznaczone pole wyboru, to oznacza to, że inny użytkownik ma przypisaną taką rolę, a jeśli pole wyboru będzie puste, to oznacza, że inny użytkownik nie posiada takiej roli), 2. Pole wyboru staje się puste, 3. Aplikacja usuwa przypisanie roli innego użytkownika.
Alternatywny przebieg:	<p>Inny użytkownik nie posiada wybranej roli (np. Administrator na Rys 2.26)</p> <ol style="list-style-type: none"> 2a. Pole wyboru zostaje zaznaczone 3a. Aplikacja przypisuje wybraną rolę innemu użytkownikowi.

Tab. 2.8: Opis przypadku użycia - Zmiana własnej dostępności

Nazwa:	Zmiana własnej dostępności
Cel:	Zmiana własnej dostępności w przypadku przypisywania artykułów do weryfikacji
Aktorzy:	Recenzent
Warunki początkowe:	Użytkownik posiadający rolę recenzenta jest zalogowany
Warunki końcowe:	Zmiana dostępności użytkownika

Przebieg:	<p>Użytkownik ma własną dostępność ustawioną na „Dostępny”</p> <ol style="list-style-type: none"> 1. Użytkownik klikna na swój awatar dostępny na nagłówku strony aby wyświetlić opcje dotyczące konta, 2. Użytkownik klikna przycisk „Dostępny” (Rys. 2.28), 3. Aplikacja wyświetla formularz do ustawienia daty, do której użytkownik będzie miał ustawioną niedostępność, 4. Użytkownik klikna na pole do wypełnienia daty, 5. Serwis wyświetla formularz do wybrania daty (Rys. 2.29), 6. Użytkownik wybiera datę (np. 18.08.2014), 7. Aplikacja wyświetla ustawioną datę, 8. Użytkownik klikna przycisk „Zapisz”, 9. Serwis sprawdza, czy użytkownik podał datę, 10. Użytkownik podał datę, zatem aplikacja zmienia dostępność recenzenta na „Niedostępny” (Rys. 2.30), co będzie skutkowało tym, że takiemu recenzentowi nie będą przypisywane artykuły do weryfikacji. Po upłynięciu daty podanej przez użytkownika, status obecności tego użytkownika zostanie automatycznie zmieniony na „Dostępny”.
Alternatywny przebieg:	<p>Użytkownik anuluje zmianę dostępności</p> <ol style="list-style-type: none"> 4a. Użytkownik klikna w „X” w prawym górnym rogu lub kilka w obszar poza formularzem, 5a. Aplikacja wyłącza formularz do zmiany dostępności.
Alternatywny przebieg:	<p>Użytkownik nie podał daty</p> <ol style="list-style-type: none"> 10b. Aplikacja zmienia kolor obramowania pola z podaniem daty na czerwono oraz wyświetla pod tym polem komunikat „Pole wymagane”, 11b. Powrót do punktu 4.
Alternatywny przebieg:	<p>Użytkownik ma ustawioną własną dostępność na „Niedostępny”</p> <ol style="list-style-type: none"> 2c. Użytkownik klikna przycisk „Niedostępny” (Rys. 2.30), 3c. Aplikacja zmienia dostępność recenzenta na „Dostępny” (Rys. 2.28), czyli takiemu recenzentowi będą teraz przypisywane artykuły do weryfikacji.



Rys. 2.28: Nagłówek strony dostępnego recenzenta z otwartymi opcjami dotyczącymi konta

Źródło: opracowanie własne

The form is titled 'Ustawienie niedostępności' (Setting Unavailability). It includes a date input field labeled 'Do dnia' (Until) with the value '18/08/2014'. Below it is a calendar for August 2014, showing the 18th as the selected date. At the bottom of the form is a blue 'Zapisz' (Save) button.

Rys. 2.29: Formularz z ustawieniem daty, do której ma być ustwiona nieobecność recenzenta razem z otwartym oknem wyboru daty

Źródło: opracowanie własne



Rys. 2.30: Nagłówek strony niedostępnego recenzenta z otwartymi opcjami dotyczącymi konta

Źródło: opracowanie własne

Tab. 2.9: Opis przypadku użycia - Wyszukiwanie artykułów

Nazwa:	Wyszukiwanie artykułów
Cel:	Wyszukiwanie artykułów według podanych przez użytkownika kryteriów i wyświetlenie wyników tego wyszukiwania w postaci listy artykułów
Aktorzy:	Użytkownik
Warunki początkowe:	Brak
Warunki końcowe:	Wyświetlenie listy wyszukanych artykułów
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Artykuły” znajdujący się w panelu nawigacji strony (Rys. 2.14), 2. Aplikacja wyświetla widok związany z wyszukiwaniem artykułów, 3. Użytkownik uzupełnia kryteria wyszukiwania, 4. Użytkownik kliką przycisk „Szukaj”, 5. Serwis wyszukuje artykuły, 6. Aplikacja wyświetla listę wyszukanych artykułów (Rys. 2.31), przy czym własne artykuły są oznaczone niebieskim prostokątem.

Artykuły	Technologie																																																								
Wyszukiwanie artykułów Dodaj artykuł																																																									
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>Tytuł</td> <td><input type="text" value="Java"/></td> </tr> <tr> <td>Autor</td> <td><input type="text" value="kamil.dywan"/></td> </tr> <tr> <td>Kategoria technologii</td> <td><input type="text" value="Język programowania"/></td> </tr> <tr> <td>Technologia</td> <td><input type="text" value="Java"/></td> </tr> <tr> <td>Dostawca technologii</td> <td><input type="text" value="Oracle"/></td> </tr> <tr> <td>Data powstania</td> <td><input type="text" value="01.01.1995"/> <input type="text" value="01.07.2000"/></td> </tr> <tr> <td>Data modyfikacji</td> <td><input type="text" value="01.01.2022"/> <input type="text" value="01.12.2022"/></td> </tr> </table>		Tytuł	<input type="text" value="Java"/>	Autor	<input type="text" value="kamil.dywan"/>	Kategoria technologii	<input type="text" value="Język programowania"/>	Technologia	<input type="text" value="Java"/>	Dostawca technologii	<input type="text" value="Oracle"/>	Data powstania	<input type="text" value="01.01.1995"/> <input type="text" value="01.07.2000"/>	Data modyfikacji	<input type="text" value="01.01.2022"/> <input type="text" value="01.12.2022"/>																																										
Tytuł	<input type="text" value="Java"/>																																																								
Autor	<input type="text" value="kamil.dywan"/>																																																								
Kategoria technologii	<input type="text" value="Język programowania"/>																																																								
Technologia	<input type="text" value="Java"/>																																																								
Dostawca technologii	<input type="text" value="Oracle"/>																																																								
Data powstania	<input type="text" value="01.01.1995"/> <input type="text" value="01.07.2000"/>																																																								
Data modyfikacji	<input type="text" value="01.01.2022"/> <input type="text" value="01.12.2022"/>																																																								
<input type="button" value="Szukaj"/>																																																									
Sortowanie ▼ < 1 2 ... 8 >																																																									
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;">Java - poradnik</td> <td style="width: 15%;">Java</td> <td style="width: 15%;">Język progr.</td> <td style="width: 15%;"></td> <td style="width: 15%;">Adam Nowak</td> <td style="width: 15%;">21:00, 01.01.2022</td> <td style="width: 15%;"></td> <td style="width: 15%;">50%</td> </tr> <tr> <td colspan="8" style="border-top: none;"><hr/></td> </tr> <tr> <td colspan="8" style="border-bottom: none;">Podstawy Javy Java Język progr. Kamil Dywan 20:30, 04.03.2022 62%</td> </tr> <tr> <td colspan="8" style="border-top: none; border-bottom: none;"><hr/></td> </tr> <tr> <td colspan="8" style="border-bottom: none;">Wersje Javy Java Język progr. Kamil Dywan 22:00, 02.05.2022 80%</td> </tr> <tr> <td colspan="8" style="border-top: none; border-bottom: none;"><hr/></td> </tr> <tr> <td colspan="8" style="border-bottom: none;">Java - strumienie Java Język progr. Michał Nowak 19:30, 06.07.2022 98%</td> </tr> </table>		Java - poradnik	Java	Język progr.		Adam Nowak	21:00, 01.01.2022		50%	<hr/>								Podstawy Javy Java Język progr. Kamil Dywan 20:30, 04.03.2022 62%								<hr/>								Wersje Javy Java Język progr. Kamil Dywan 22:00, 02.05.2022 80%								<hr/>								Java - strumienie Java Język progr. Michał Nowak 19:30, 06.07.2022 98%							
Java - poradnik	Java	Język progr.		Adam Nowak	21:00, 01.01.2022		50%																																																		
<hr/>																																																									
Podstawy Javy Java Język progr. Kamil Dywan 20:30, 04.03.2022 62%																																																									
<hr/>																																																									
Wersje Javy Java Język progr. Kamil Dywan 22:00, 02.05.2022 80%																																																									
<hr/>																																																									
Java - strumienie Java Język progr. Michał Nowak 19:30, 06.07.2022 98%																																																									
Sortowanie ▼ < 1 2 ... 8 >																																																									

Rys. 2.31: Panel do wyszukiwania artykułów z podanymi kryteriami wyszukiwania oraz listą wyszukanych artykułów

Źródło: opracowanie własne

Tab. 2.10: Opis przypadku użycia - Sortowanie artykułów

Nazwa:	Sortowanie artykułów
Cel:	Posortowanie wyszukanych artykułów według opcji sortowania wybranej przez użytkownika oraz wyświetlenie listy tych posortowanych artykułów
Aktorzy:	Użytkownik
Warunki początkowe:	Wyświetlona lista wyszukanych artykułów (Rys. 2.31)
Warunki końcowe:	Wyświetlenie listy posortowanych artykułów
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką na rozsuwaną listę „Sortowanie”, 2. Użytkownik wybiera jedną z opcji sortowania (Rys. 2.32), 3. Aplikacja sortuje artykuły według wybranej opcji sortowania, 4. Aplikacja wyświetla listę posortowanych artykułów, przy czym własne artykuły są oznaczone niebieskim prostokątem.

Sortowanie ▼
Nazwa: malejąco
Nazwa: rosnąco
Technologia: malejąco
Technologia: rosnąco
Autor: malejąco
Autor: rosnąco
Data powstania: malejąco
Data powstania: rosnąco
Data modyfikacji: malejąco
Data modyfikacji: rosnąco
Średnia ocena: malejąco
Średnia ocena: rosnąco
Liczba opinii: malejąco
Liczba opinii: rosnąco

Rys. 2.32: Dostępne opcje sortowania artykułów

Źródło: opracowanie własne

Tab. 2.11: Opis przypadku użycia - Wyświetlenie zawartości artykułu

Nazwa:	Wyświetlenie zawartości artykułu
Cel:	Wyświetlenie zawartości artykułu
Aktorzy:	Użytkownik
Warunki początkowe:	Wyświetlona lista wyszukanych artykułów (Rys. 2.31)
Warunki końcowe:	Wyświetlenie zawartości artykułu
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik klikna na jeden z wyszukanych artykułów, 2. Aplikacja wyświetla zawartość artykułu oraz komentarze dotyczące tego artykułu (Rys. 2.33).

Artykuły	Technologie
<p>Kamil Dywan</p> <p>Data utworzenia: 14.07.2022, 00:32:45</p> <p>Data modyfikacji: 14.07.2022, 00:32:45</p> <p>Technologia → Język programowania → Java</p>  <p>Dostawca: Oracle</p> <p>Opis:</p> <p>Java jest to wspólnie dostępny, obiektowy język programowania. Język ten jest niezwykle popularny oraz uniwersalny i dlatego trudnym zadaniem byłoby znalezienie dziedziny na pograniczu informatyki, w której problemach nie można by było wykorzystać Javy. Jednym z powodów uniwersalności tego języka jest to, że programy napisane w Javie są uruchamiane na maszynie wirtualnej, co też powoduje, że aby uruchomić program napisany w Javie, wystarczy posiadać zaistalowane JRE (kod bajtowy klas standardowych razem z maszyną wirtualną). Obecnie Java jest wykorzystywana przede wszystkim jako backend w aplikacjach webowych i mobilnych.</p> <h3>Podstawy Javy</h3> <p>Java jest to wspólnie dostępny, oparty na klasach i mocno typowany obiektowy język programowania, który może być wykorzystywany w wielu dziedzinach. Został stworzony przez zespół pod kierownictwem Jamesa Goslinga z firmy Sun Microsystems, a pierwsze wydanie tego języka odbyło się w 1996 roku. Ważną cechą tego języka jest to, że przy komplikacji źródła kodu w nim napisanych, używana jest maszyna wirtualna nazywana JVM (ang. Java Virtual Machine).</p> <ol style="list-style-type: none"> 1. JVM Kod napisany w tym języku jest komplikowany do kodu bajtowego, który to następnie jest wykonywany przez maszynę wirtualną. Dodatkowo używana maszyna wirtualna sprawia, że język ten jest wieloplatformowy. ... 5. Zastosowania <p>Z powodu na używaną maszynę wirtualną, programy napisane w Javie mogą być uruchomione na praktycznie wszystkich urządzeniach, co prowadzi do tego, że zastosowania tego języka mogą być bardzo rozległe. Przede wszystkim język jest wykorzystywany w aplikacjach Webowych jako backend oraz aplikacjach mobilnych. W przypadku zastosowań biznesowych język jest wykorzystywany przede wszystkim w rozwiązańach wysokopoziomowych przez korporacje i duże firmy. Język ten z powodu używanej maszyny wirtualnej jest wolniejszy niż np. C++, dlatego też jest najczęściej wykorzystywany w aplikacjach wysokopoziomowych.</p> <p>Platforma Mobilne Web Wysokopoziomowe rozwiązania Korporacyjne aplikacje</p> <p>6. Wersje</p> <p>1.0 - 1996 2.0 - grudzień 1998 - 1999 ... 9.0 - wrzesień 2017 11.0 - wrzesień 2018 15.0 - wrzesień 2020 17.0 - wrzesień 2021 7. Linki</p> <p>[1] https://docs.oracle.com/en/java/ - oficjalna dokumentacja</p> <p> 75%</p> <p>Komentarze Opinie Opis technologii</p> <p>Dodaj komentarz</p> <p> Kamil Dywan 15.07.2022, 00:32:45 </p> <p>Dobry artykuł Dodaj komentarz</p>	<p>Status: Opublikowany</p> <p>Edytuj Usuń</p>

Rys. 2.33: Panel z przykładową zawartością artykułu

Źródło: opracowanie własne

Tab. 2.12: Opis przypadku użycia - Utworzenie artykułu

Nazwa:	Utworzenie artykułu
Cel:	Utworzenie artykułu i wyświetlenie zawartości tego artykułu
Aktorzy:	Zalogowany użytkownik
Warunki początkowe:	Wyświetlony panel do wyszukiwania artykułów (Rys. 2.31)
Warunki końcowe:	Wyświetlenie zawartości utworzonego artykułu
Przebieg:	<p>Dodawanie artykułu</p> <ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Dodaj artykuł”, 2. Aplikacja wyświetla formularz do dodania artykułu, 3. Użytkownik wypełnia wszystkie pola w formularzu (Rys. 2.34) 4. Użytkownik kliką przycisk „Zapisz”, 5. Serwis przeprowadza walidację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 6. Po pozytywnej walidacji danych, aplikacja zapisuje stan artykułu, 7. Użytkownik kliką przycisk „Wyślij do weryfikacji”, 8. Serwis przesyła artykuł do weryfikacji (zapisuje artykuł w kolejce artykułów, do których mają być przydzielone recenzencji), 9. Aplikacja wyświetla użytkownikowi wygląd utworzonego artykułu.
Alternatywny przebieg:	<p>Użytkownik anuluje dodanie artykułu</p> <ol style="list-style-type: none"> 3a. Użytkownik kliką przycisk „Anuluj”, 4a. Użytkownik zostaje przekierowany do strony z wyszukiwaniem artykułów (Rys. 2.31).
Alternatywny przebieg:	<p>Użytkownik nie uzupełnia wszystkich pól formularza podczas dodawania artykułu</p> <ol style="list-style-type: none"> 6b. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 7b. Przejście do punktu 3.

Tytuł Kategoria v

Java jest to współbieżny, oparty na klasach i mocno typowany obiektowy język programowania, który może być wykorzystywany w wielu dziedzinach. Został stworzony przez zespół pod kierownictwem Jamesa Goslinga z firmy Sun Microsystems, a pierwsze wydanie tego języka odbyło się w 1996 roku. Ważną cechą tego języka jest to, że przy komplikacji źródłek kodu w nim napisanych, używana jest maszyna wirtualna naazywana JVM (ang. Java Virtual Machine).

1. JVM

Kod napisany w tym języku jest kompilowany do kodu bajtowego, który to następnie jest wykonywany przez maszynę wirtualną. Dodatkowo używana maszyna wirtualna sprawia, że język ten jest wieloplatformowy.

...
6. Wersje

1.0 - 1996
 2.0 - grudzień 1998 - 1999
 ...
 9.0 - wrzesień 2017
 11.0 - wrzesień 2018
 15.0 - wrzesień 2020
 17.0 - wrzesień 2021

7. Linki

[1] <https://docs.oracle.com/en/java/> - oficjalna dokumentacja

Rys. 2.34: Panel z formularzem dodania albo edytowania artykułu

Źródło: opracowanie własne

Tab. 2.13: Opis przypadku użycia - Zarządzanie własnym artykułem

Nazwa:	Zarządzanie własnym artykułem
Cel:	Zmodyfikowanie lub usunięcie własnego artykułu
Aktorzy:	Autor artykułu
Warunki początkowe:	Wyświetlony panel do wyszukiwania artykułów (Rys. 2.31)
Warunki końcowe:	Zmodyfikowanie lub usunięcie artykułu

Przebieg:	<p>Edytowanie artykułu</p> <ol style="list-style-type: none"> 1. Użytkownik wyszukuje artykuły (Tab. 2.9), 2. Użytkownik kliką na jeden z wyszukanych artykułów, 3. Aplikacja wyświetla zawartość wybranego artykułu (Rys. 2.33), 4. Użytkownik kliką przycisk „Edytuj”, 5. Serwis wyświetla formularz do edycji artykułu (Rys. 2.34), 6. Użytkownik modyfikuje pola w formularzu, 7. Użytkownik kliką przycisk „Zapisz”, 8. Aplikacja przeprowadza validację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 9. Po pozytywnej validacji danych, aplikacja zapisuje stan edytowanego artykułu, 10. Użytkownik kliką przycisk „Wyślij do weryfikacji”, 11. Serwis przesyła artykuł do weryfikacji (zapisuje artykuł w kolejce artykułów, do których mają być przydzieleni recenzenci), 12. Aplikacja przedstawia użytkownikowi wygląd zmienionego artykułu.
Alternatywny przebieg:	<p>Użytkownik anuluje edytowanie artykułu</p> <ol style="list-style-type: none"> 6a. Użytkownik kliką przycisk „Anuluj”, 7a. Użytkownik zostaje przekierowany do strony z wyszukiwaniem artykułów (Rys. 2.31).
Alternatywny przebieg:	<p>Użytkownik nie uzupełnia wszystkich pól formularza podczas edytowania artykułu</p> <ol style="list-style-type: none"> 9b. Aplikacja zmienia kolor obramowania pól, które nie przeszły validacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 10b. Przejście do punktu 6.
Alternatywny przebieg:	<p>Usunięcie artykułu</p> <ol style="list-style-type: none"> 4c. Użytkownik kliką przycisk „Usuń”, 5c. Aplikacja wyświetla formularz z zapytaniem „Czy na pewno ten artykuł powinien zostać usunięty?”, 6c. Użytkownik wybiera opcję „Tak”, 7c. Serwis usuwa artykuł, 8c. Użytkownik zostaje przekierowany do strony z wyszukiwaniem artykułów (Rys. 2.31).

Alternatywny przebieg:	Anulowanie usunięcia artykułu 6ca. Użytkownik wybiera opcję „Nie” albo kliką w obszar poza formularzem, 7ca. Użytkownik zostaje przekierowany do strony z wyszukiwaniem artykułów (Rys. 2.31).
------------------------	--

Tab. 2.14: Opis przypadku użycia - Wyświetlenie listy przypisanych artykułów

Nazwa:	Wyświetlenie listy przypisanych artykułów
Cel:	Wyświetlenie listy artykułów przypisanych recenzentowi do weryfikacji
Aktorzy:	Recenzent
Warunki początkowe:	Użytkownik będący recenzentem jest zalogowany
Warunki końcowe:	Wyświetlenie listy przypisanych artykułów
Przebieg:	<ol style="list-style-type: none"> Użytkownik kliką przycisk „Weryfikacja artykułów” znajdujący się w menu głównym aplikacji (Rys. 2.35), Aplikacja wyświetla panel z listą artykułów przypisanych użytkownikowi do weryfikacji (Rys. 2.36),

Artykuły	Technologie	Weryfikacja artykułów
----------	-------------	-----------------------

Rys. 2.35: Nagłówek strony recenzenta

Źródło: opracowanie własne

Artykuły	Technologie	Weryfikacja artykułów
----------	-------------	-----------------------

Weryfikacja artykułów

Java - poradnik	Java	Język progr.		Adam Nowak	21:00, 01.01.2022
Podstawy Javy	Java	Język progr.		Kamil Dywan	20:30, 04.03.2022
React - podstawy	JS	Framework		Kamil Dywan	20:03, 07.03.2022
SQL - przykłady	JS	Język bazy danych		Kamil Dywan	20:03, 07.03.2022

<
1
2
...
8
>

Rys. 2.36: Panel z listą artykułów przypisanych recenzentowi do weryfikacji

Źródło: opracowanie własne

Tab. 2.15: Opis przypadku użycia - Weryfikacja artykułu

Nazwa:	Weryfikacja przypisanych artykułów
Cel:	Wyświetlenie panelu służącego do weryfikacji artykułu
Aktorzy:	Recenzent
Warunki początkowe:	Wyświetlona lista przypisanych użytkownikowi artykułów do weryfikacji (Rys. 2.36)
Warunki końcowe:	Wyświetlenie panelu służącego do weryfikacji artykułu
Przebieg:	1. Użytkownik klikna na jeden z artykułów, 2. Serwis wyświetla formularz do przeprowadzenia weryfikacji artykułu (Rys. 2.37).

Artykuły	Technologie	Weryfikacja artykułów
----------	-------------	-----------------------

Weryfikacja Artykułu „Podstawy Javy”

Kamil Dywan

Data utworzenia: 14.07.2022, 00:32:45

Data modyfikacji: 14.07.2022, 00:32:45

Technologia → Język programowania → Java



Dostawca: Oracle

Zastosowania:

- Backend,
- Aplikacje webowe.

Podstawy Javy

Java jest to wspólnie używany, oparty na klasach i mocno typowany obiektowy język programowania, który może być wykorzystywany w wielu dziedzinach. Został stworzony przez zespół pod kierownictwem Jamesa Goslinga z firmy Sun Microsystems, a pierwsze wydanie tego języka odbyło się w 1996 roku. Ważną cechą tego języka jest to, że przy komplikacji źródłek kodu w nim napisanych, używana jest maszyna wirtualna nazywana JVM (ang. Java Virtual Machine).

1. JVM

Kod napisany w tym języku jest komplikowany do kodu bajtowego, który następnie jest wykonywany przez maszynę wirtualną. Dodatkowo używana maszyna wirtualna sprawia, że język ten jest wieloplatformowy.

...

5. Zastosowania

Z powodu na używaną maszynę wirtualną, programy napisane w Javie mogą być uruchomione na praktycznie wszystkich urządzeniach, co prowadzi do tego, że zastosowania tego języka mogą być bardzo rozległe. Przed wszystkim język jest wykorzystywany w aplikacjach Webowych jako backend oraz aplikacjach mobilnych. W przypadku zastosowań biznesowych język jest wykorzystywany przede wszystkim w rozwiązaniach wysokopoziomowych przez korporacje i duże firmy. Język ten z powodu używanej maszyny wirtualnej jest wolniejszy niż np. C++, dlatego też jest najczęściej wykorzystywany w aplikacjach wysokopoziomowych.

Platforma

Mobilne

Web

Wysokopoziomowe rozwiązania

Korporacyjne aplikacje

6. Wersje

1.0 - 1996

2.0 - grudzień 1998 - 1999

...

9.0 - wrzesień 2017

11.0 - wrzesień 2018

15.0 - wrzesień 2020

17.0 - wrzesień 2021

7. Linki

[1] <https://docs.oracle.com/en/java/> - oficjalna dokumentacja

Akceptuj

Odrzuć

Rys. 2.37: Panel służący do weryfikacji artykułu

Źródło: opracowanie własne

Tab. 2.16: Opis przypadku użycia - Akceptacja artykułu

Nazwa:	Akceptacja artykułu
Cel:	Akceptacja artykułu, co będzie skutkowało tym, że artykuł zostanie opublikowany w serwisie

Aktorzy:	Recenzent
Warunki początkowe:	Wyświetlony formularz weryfikacji artykułu (Rys. 2.37)
Warunki końcowe:	Akceptacja artykułu
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik przegląda cały artykuł, 2. Użytkownik podaje wiadomość zwrotną dotyczącą treści artykułu: uzasadnienie akceptacji artykułu oraz ewentualne wskazówki, co można by było poprawić w artykule (Rys. 2.38), 3. Użytkownik kliką przycisk „Akceptuj”, 4. Artykuł zostaje zaakceptowany.
Alternatywny przebieg:	<p>Użytkownik anuluje akceptację artykułu</p> <ol style="list-style-type: none"> 2a. Użytkownik kliką przycisk „Anuluj”, 3a. Serwis wyłącza formularz do akceptacji artykułu.
Alternatywny przebieg:	<p>Użytkownik nie uzupełnia pola tekstowego</p> <ol style="list-style-type: none"> 2b. Przejście do punktu 3.

Wiadomość zwrotna dotycząca treści artykułu

Artykuł dobry, lecz przydałoby się dodać trochę informacji o twórcach tej technologii.

Akceptuj
|
Odrzuć

Anuluj

Rys. 2.38: Formularz akceptacji artykułu z podaną informacją zwrotną

Źródło: opracowanie własne

Tab. 2.17: Opis przypadku użycia - Odrzucenie artykułu

Nazwa:	Odrzucenie artykułu
Cel:	Odrzucenie artykułu, co będzie skutkowało tym, że artykuł nie zostanie opublikowany w serwisie, gdyż wymaga poprawek

Aktorzy:	Recenzent
Warunki początkowe:	Wyświetlony formularz służący do weryfikacji artykułu (Rys. 2.37)
Warunki końcowe:	Odrzucenie artykułu
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik przegląda cały artykuł, 2. Użytkownik uzupełnia uzasadnienie odrzucenia artykułu (Rys. 2.39), 3. Użytkownik kliką przycisk „Odrzuć”, 4. Serwis sprawdza, czy użytkownik uzupełnił pole tekstowe, 5. Po pozytywnej weryfikacji, artykuł zostaje odrzucony.
Alternatywny przebieg:	<p>Użytkownik anuluje akceptację artykułu</p> <ol style="list-style-type: none"> 2a. Użytkownik kliką przycisk „Anuluj”, 3a. Serwis wyłącza formularz do odrzucenia artykułu.
Alternatywny przebieg:	<p>Użytkownik nie uzupełnił pola tekstowego</p> <ol style="list-style-type: none"> 5b. Serwis ustawia obramowanie nieuzupełnionego pola tekstowego na czerwono oraz umieszcza pod tym polem komunikat o konieczności jego uzupełnienia, 6b. Powrót do punktu 2.

Wiadomość zwrotna dotycząca treści artykułu

Artykuł posiada informacje nieprawdziwe odnośnie wersji technologii oraz naniesionych zmian w ramach tych wersji

[Akceptuj](#) [Odrzuć](#)

[Anuluj](#)

Rys. 2.39: Formularz odrzucenia artykułu

Źródło: opracowanie własne

Tab. 2.18: Opis przypadku użycia - Zarządzanie artykułami

Nazwa:	Zarządzanie artykułami
Cel:	Modyfikacja lub usunięcie artykułu
Aktorzy:	Administrator
Warunki początkowe:	Wyświetlona lista z wyszukanymi artykułami (Rys. 2.31)
Warunki końcowe:	Modyfikacja lub usunięcie artykułu
Przebieg:	<p>Edytowanie artykułu</p> <ol style="list-style-type: none"> 1. Użytkownik kliką na jeden z wyszukanych artykułów, 2. Aplikacja wyświetla zawartość wybranego artykułu (Rys. 2.40), 3. Użytkownik kliką przycisk „Edytuj”, 4. Serwis wyświetla formularz do edycji artykułu (Rys. 2.34) 5. Użytkownik wypełnia wszystkie pola w formularzu, 6. Użytkownik kliką przycisk „Zapisz”, 7. Aplikacja przeprowadza walidację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 8. Po pozytywnej walidacji danych, aplikacja edytuje artykuł według zmian wprowadzonych przez użytkownika, 9. Aplikacja przedstawia użytkownikowi wygląd zmienionego artykułu.
Alternatywny przebieg:	<p>Użytkownik anuluje edytowanie artykułu</p> <ol style="list-style-type: none"> 5a. Użytkownik kliką przycisk „Anuluj”, 6a. Użytkownik zostaje przekierowany do strony z wyszukiwaniem artykułów (Rys. 2.31).
Alternatywny przebieg:	<p>Użytkownik nie uzupełnia wszystkich pól formularza podczas edytowania artykułu</p> <ol style="list-style-type: none"> 8b. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 9b. Przejście do punktu 5.
Alternatywny przebieg:	<p>Usunięcie artykułu</p> <ol style="list-style-type: none"> 3c. Użytkownik kliką przycisk „Usuń”, 4c. Aplikacja wyświetla formularz z zapytaniem „Czy na pewno ten artykuł powinien zostać usunięty?”, 5c. Użytkownik wybiera opcję „Tak”, 6c. Serwis usuwa artykuł, 7c. Użytkownik zostaje przekierowany do strony z wyszukiwaniem artykułów (Rys. 2.31).

Alternatywny przebieg:	Anulowanie usunięcia artykułu 5ca. Użytkownik wybiera opcję „Nie” albo kilka w obszarze poza formularzem, 6ca. Artykuł nie zostaje usunięty.
------------------------	--

IT Tech Technologie IT

Kamil Dywan

Artykuły	Technologie	Zarządzanie użytkownikami
----------	-------------	---------------------------

Kamil Dywan

Data utworzenia: 14.07.2022, 00:32:45

Data modyfikacji: 14.07.2022, 00:32:45

Technologia → Język programowania → Java



Dostawca: Oracle

Opis:

Java jest to współbieżny, obiektowy język programowania. Język ten jest niezwykle popularny oraz uniwersalny i dlatego trudnym zadaniem byłoby znalezienie dziedziny na pograniczu informatyki, w której problemach nie można by było wykorzystać Javy. Jednym z powodów uniewersalności tego języka jest to, że programy napisane w Javie są uruchamiane na maszynie wirtualnej, co też powoduje, że aby uruchomić program napisany w Javie, wystarczy posiadać zaistalowane JRE (kod bajtowy klas standardowych razem z maszyną wirtualną). Obecnie Java jest wykorzystywana przede wszystkim jako backend w aplikacjach webowych i mobilnych.

Podstawy Javy

Java jest to współbieżny, oparty na klasach i mocno typowany obiektowy język programowania, który może być wykorzystywany w wielu dziedzinach. Został stworzony przez zespół pod kierownictwem Jamesa Goslinga z firmy Sun Microsystems, a pierwsze wydanie tego języka odbyło się w 1996 roku. Ważną cechą tego języka jest to, że przy komplikacji źródeł kodu w nim napisanych, używana jest maszyna wirtualna nazywana JVM (ang. Java Virtual Machine).

1. JVM

Kod napisany w tym języku jest kompilowany do kodu bajtowego, który następnie jest wykonywany przez maszynę wirtualną. Dodatkowo używana maszyna wirtualna sprawia, że język ten jest wieloplatformowy.

...

5. Zastosowania

Z powodu na używaną maszynę wirtualną, programy napisane w Javie mogą być uruchomione na praktycznie wszystkich urządzeniach, co prowadzi do tego, że zastosowania tego języka mogą być bardzo rozległe. Przede wszystkim język jest wykorzystywany w aplikacjach Webowych jako backend oraz aplikacjach mobilnych. W przypadku zastosowań biznesowych język jest wykorzystywany przede wszystkim w rozwiązańach wysokopoziomowych przez korporacje i duże firmy. Język ten z powodu używanej maszyny wirtualnej jest wolniejszy niż np. C++, dlatego też jest najczęściej wykorzystywany w aplikacjach wysokopoziomowych.

Platforma
Mobilne
Web
Wysokopoziomowe rozwiązania
Korporacyjne aplikacje

6. Wersje

1.0 - 1996
2.0 - grudzień 1998 - 1999
...
9.0 - wrzesień 2017
11.0 - wrzesień 2018
15.0 - wrzesień 2020
17.0 - wrzesień 2021
7. Linki

[1] <https://docs.oracle.com/en/java/> - oficjalna dokumentacja

★★★★★ 75%

Komentarze	Opinie	Opis technologii
------------	--------	------------------

Dodaj komentarz

Kamil Dywan 15.07.2022, 00:32:45 :

Dobry artykuł

Dodaj komentarz

Rys. 2.40: Panel z zawartością artykułu widziany z perspektywy administratora

Źródło: opracowanie własne

Tab. 2.19: Opis przypadku użycia - Przeglądanie komentarzy o artykule

Nazwa:	Przeglądanie komentarzy o artykule
Cel:	Wyświetlenie komentarzy dotyczących danego artykułu
Aktorzy:	Użytkownik
Warunki początkowe:	Wyświetlona zawartość artykułu (Rys. 2.33)
Warunki końcowe:	Wyświetlenie komentarzy dotyczących danego artykułu
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik przewija artykuł na sam dół, 2. Użytkownik kliką przycisk „Komentarze”, 3. Wyświetlenie komentarzy o artykule (Rys. 2.41).

Komentarze Opinie Opis technologii

Dodaj komentarz

Kamil Dywan 15.07.2022, 00:32:45 :

Dobry artykuł

Dodaj komentarz

v Odpowiedzi

Adam Nowak 18.07.2022, 00:32:45

Zgadzam się

Dodaj komentarz

Kamil Dywan 20.07.2022, 00:32:45 :

Artykuł dobry, ale przydałoby się zamieścić więcej informacji o wersjach Javy oraz opis zmian w ramach tych wersji

Dodaj komentarz

Rys. 2.41: Panel z listą komentarzy o artykule

Źródło: opracowanie własne

Tab. 2.20: Opis przypadku użycia - Dodanie komentarza o artykule

Nazwa:	Dodanie komentarza o artykule
Cel:	Dodanie komentarza do artykułu
Aktorzy:	Zalogowany użytkownik
Warunki początkowe:	Wyświetlone komentarze o artykule (Rys. 2.41)
Warunki końcowe:	Dodanie komentarza do artykułu
Przebieg:	<p>Dodanie komentarza</p> <ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Dodaj komentarz”, 2. Aplikacja wyświetla formularz do utworzenia komentarza, 3. Użytkownik dodaje tekst komentarza (Rys. 2.42), 4. Użytkownik kliką przycisk „Zapisz”, 5. Aplikacja sprawdza, czy użytkownik podał tekst komentarza, 6. Po pozytywnej walidacji komentarza, aplikacja dodaje komentarz do artykułu.

Alternatywny przebieg:	Użytkownik anuluje tworzenie komentarza 3a. Użytkownik kliką przycisk „Anuluj” na formularzu do tworzenia komentarza, 4a. Wyłączenie formularza do dodania komentarza.
Alternatywny przebieg:	Użytkownik nie uzupełnia tekstu przy tworzeniu komentarza 6b. Aplikacja wyświetla użytkownikowi informacje o tym, że komentarz nie może być pusty, 7b. Przejście do punktu 3.

Podoba mi się ten artykuł. Dodałbyś może coś o Spring'u?

[Zapisz](#) [Anuluj](#)

Rys. 2.42: Formularz dodania albo edytowania komentarza o artykule

Źródło: opracowanie własne

Tab. 2.21: Opis przypadku użycia - Zarządzanie własnym komentarzem o artykule

Nazwa:	Zarządzanie własnym komentarzem o artykule
Cel:	Modyfikacja lub usunięcie własnego komentarza do artykułu
Aktorzy:	Autor komentarza o artykule
Warunki początkowe:	Wyświetlone komentarze o artykule (Rys. 2.41)
Warunki końcowe:	Modyfikacja lub usunięcie komentarza do artykułu
Przebieg:	<p>Edytowanie komentarza</p> <ol style="list-style-type: none"> 1. Użytkownik kliką trzy kropki przy jednym z komentarzy, 2. Aplikacja wyświetla listę działań, które mogą zostać wykonane w ramach wybranego komentarza (Rys. 2.43), 3. Użytkownik kliką przycisk „Edytuj”, 4. Aplikacja wyświetla formularz do edytowania komentarza (Rys. 2.42), 5. Użytkownik edytuje tekst komentarza, 6. Użytkownik kliką przycisk „Zapisz”, 7. Aplikacja sprawdza, czy użytkownik podał tekst komentarza, 8. Po pozytywnej walidacji komentarza, aplikacja edytuje komentarz do artykułu.

Alternatywny przebieg:	Użytkownik anuluje edytowanie komentarza 5a. Użytkownik kliką przycisk „Anuluj” na formularzu do edytowania komentarza, 6a. Wyłączenie formularza do edytowania komentarza.
Alternatywny przebieg:	Użytkownik nie uzupełnia tekstu przy edytowaniu komentarza 8b. Aplikacja wyświetla użytkownikowi informacje o tym, że komentarz nie może być pusty, 9b. Przejście do punktu 5.
Alternatywny Przebieg:	Usunięcie komentarza 3c. Użytkownik kliką przycisk „Usuń” przy jednym z komentarzy, 4c. Aplikacja wyświetla formularz z zapytaniem „Czy na pewno ten komentarz powinien zostać usunięty?”, 5c. Użytkownik wybiera opcję „Tak”, 6c. Serwis usuwa komentarz.
Alternatywny przebieg:	Użytkownik anuluje usuwanie komentarza 5ca. Użytkownik wybiera opcję „Nie”, 6ca. Komentarz nie zostaje usunięty.



Rys. 2.43: Dostępne opcje zarządzania komentarzem

Źródło: opracowanie własne

Tab. 2.22: Opis przypadku użycia - Usuwanie komentarzy o artykule

Nazwa:	Usuwanie komentarzy o artykule
Cel:	Usunięcie wybranego komentarza o artykule
Aktorzy:	Administrator
Warunki początkowe:	Wyświetlone komentarze o artykule (Rys. 2.44)
Warunki końcowe:	Usunięcie komentarza do artykułu

Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką trzy kropki przy jednym z komentarzy, 2. Aplikacja wyświetla listę działań, które mogą zostać wykonane w ramach wybranego komentarza (Rys. 2.43), 3. Użytkownik kliką przycisk „Usuń”, 4. Aplikacja wyświetla formularz z zapytaniem „Czy na pewno ten komentarz powinien zostać usunięty?”, 5. Użytkownik wybiera opcję „Tak”, 6. Serwis usuwa komentarz.
Alternatywny przebieg:	<p>Użytkownik anuluje usuwanie komentarza</p> <ol style="list-style-type: none"> 5a. Użytkownik wybiera opcję „Nie”, 6a. Komentarz nie zostaje usunięty.

Rys. 2.44: Lista komentarzy o artykule widziana z perspektywy administratora

Źródło: opracowanie własne

Tab. 2.23: Opis przypadku użycia - Przeglądanie opinii o artykule

Nazwa:	Przeglądanie opinii o artykule
Cel:	Wyświetlenie opinii dotyczących danego artykułu
Aktorzy:	Użytkownik
Warunki początkowe:	Wyświetlona zawartość artykułu (Rys. 2.33)
Warunki końcowe:	Wyświetlenie opinii dotyczących danego artykułu
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik przewija artykuł na sam dół, 2. Użytkownik kliką na przycisk „Opinie”, 3. Aplikacja wyświetla opinie o artykule (Rys. 2.45).

The screenshot shows a user interface for managing reviews. At the top, there are three tabs: "Komentarze", "Opinie" (which is selected), and "Opis technologii". Below the tabs is a button "Dodaj opinię". Two reviews are listed:

- Kamil Dywan** 15.07.2022, 00:32:45 :
Bardzo dobry artykuł!
Likes: 2 Dislikes: 0
- Adam Nowak** 20.07.2022, 00:32:45:
Artykuł dobry, ale przydałoby się zamieścić więcej informacji o wersjach Javy oraz opis zmian w ramach tych wersji
Likes: 10 Dislikes: 1

Rys. 2.45: Panel z listą opinii o artykule

Źródło: opracowanie własne

Tab. 2.24: Opis przypadku użycia - Dodanie opinii o artykule

Nazwa:	Dodanie opinii o artykule
Cel:	Dodanie opinii o artykule
Aktorzy:	Zalogowany użytkownik, który nie jest autorem artykułu
Warunki początkowe:	Wyświetlone opinie o artykule (Rys. 2.45)
Warunki końcowe:	Dodanie opinii o artykule
Przebieg:	<p>Dodanie opinii</p> <ol style="list-style-type: none"> Użytkownik klika przycisk „Dodaj opinię”, Aplikacja wyświetla formularz do utworzenia opinii, Użytkownik dodaje tekst opinii oraz wystawia ocenę (Rys. 2.46), Użytkownik klika przycisk „Zapisz”, Aplikacja sprawdza, czy użytkownik podał tekst opinii oraz ocenę, Po pozytywnej walidacji, aplikacja dodaje opinię.
Alternatywny przebieg:	<p>Użytkownik anuluje tworzenie opinii</p> <ol style="list-style-type: none"> a. Użytkownik klika przycisk „Anuluj” na formularzu do tworzenia opinii, a. Wyłączenie formularza do dodania opinii.

Alternatywny przebieg:	<p>Użytkownik nie uzupełnia wszystkich wymaganych pól formularza</p> <p>6b. Aplikacja zmienia kolor obramowania nieuzupełnionych pól na czerwono oraz wyświetla pod tymi polami komunikat „Pole wymagane”, 7b. Przejście do punktu 3.</p>
------------------------	--

Rys. 2.46: Formularz dodania albo edytowania opinii o artykule

Źródło: opracowanie własne

Tab. 2.25: Opis przypadku użycia - Zarządzanie własną opinią o artykule

Nazwa: Cel: Aktorzy: Warunki początkowe: Warunki końcowe: Przebieg: Alternatywny przebieg:	Zarządzanie własną opinią o artykule Modyfikacja lub usunięcie własnej opinii do artykułu Autor opinii o artykule Wyświetlone opinie o artykule (Rys. 2.45) Modyfikacja lub usunięcie własnej opinii do artykułu Edytowanie opinii 1. Użytkownik kliką trzy kropki przy jednej z opinii, 2. Aplikacja wyświetla listę działań, które mogą zostać wykonane w ramach wybranej opinii 3. Użytkownik kliką przycisk „Edytuj” 4. Aplikacja wyświetla formularz do edytowania opinii (Rys. 2.46), 5. Użytkownik edytuje opinię, 6. Użytkownik kliką przycisk „Zapisz”, 7. Aplikacja sprawdza, czy użytkownik podał tekst opinii oraz ocenę, 8. Po pozytywnej walidacji, aplikacja edytuje opinię. Użytkownik anuluje edytowanie opinii 5a. Użytkownik kliką przycisk „Anuluj” na formularzu do edytowania opinii, 6a. Wyłączenie formularza do edytowania opinii.
--	---

Alternatywny przebieg:	Użytkownik nie uzupełnia tekstu przy edytowaniu opinii 8b. Aplikacja zmienia kolor obramowania nieuzupełnionych pól na czerwono oraz wyświetla pod tymi polami komunikat „Pole wymagane”, 9b. Przejście do punktu 5.
Alternatywny Przebieg:	Usunięcie opinii 3c. Użytkownik kliką przycisk „Usuń”, 4c. Aplikacja wyświetla formularz z zapytaniem „Czy na pewno ta opinia powinna zostać usunięta?”, 5c. Użytkownik wybiera opcję „Tak”, 6c. Serwis usuwa opinię.
Alternatywny przebieg:	Użytkownik anuluje usuwanie opinii 5ca. Użytkownik wybiera opcję „Nie”, 6ca. Opinia nie zostaje usunięta.

Tab. 2.26: Opis przypadku użycia - Usuwanie opinii o artykule

Nazwa:	Usuwanie komentarzy o artykule
Cel:	Usunięcie wybranej opinii o artykule
Aktorzy:	Administrator
Warunki początkowe:	Wyświetlone opinie o artykule (Rys. 2.47)
Warunki końcowe:	Usunięcie opinii do artykułu
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką trzy kropki przy jednej z opinii, 2. Aplikacja wyświetla listę działań, które mogą zostać wykonane w ramach wybranej opinii 3. Użytkownik kliką przycisk „Usuń” 4. Aplikacja wyświetla formularz z zapytaniem „Czy na pewno ta opinia powinna zostać usunięta?”, 5. Użytkownik wybiera opcję „Tak”, 6. Serwis usuwa opinię.
Alternatywny przebieg:	Użytkownik anuluje usuwanie opinii 5a. Użytkownik wybiera opcję „Nie”, 6a. Opinia nie zostaje usunięta.

Komentarze Opinie Opis technologii

[Dodaj opinię](#)

★★★★★

 Kamil Dywan 15.07.2022, 00:32:45 :

Bardzo dobry artykuł

Like 2 Report X

★★★★★

 Adam Nowak 20.07.2022, 00:32:45 :

Artykuł dobry, ale przydałoby się zamieścić więcej informacji o wersjach Javy oraz opis zmian w ramach tych wersji

Like 10 Report 1

Rys. 2.47: Panel z listą opinii o artykule widziany z perspektywy administratora

Źródło: opracowanie własne

Tab. 2.27: Opis przypadku użycia - Dodanie akceptacji opinii o artykule

Nazwa:	Dodanie akceptacji opinii o artykule
Cel:	Dodanie akceptacji opinii o artykule
Aktorzy:	Zalogowany użytkownik, który nie jest autorem opinii o artykule oraz autorem akceptacji opinii o artykule
Warunki początkowe:	Wyświetlone opinie o artykule (Rys. 2.45)
Warunki końcowe:	Wyświetlenie opinii z zaktualizowanym stanem akceptacji pozytywnych i negatywnych
Przebieg:	<p>Dodanie pozytywnej akceptacji</p> <ol style="list-style-type: none"> Użytkownik kliką przy jednej z opinii kciuk w góre, Aplikacja zwiększa o 1 liczbę pozytywnych akceptacji wybranej opinii, Aplikacja zmienia kolor kciuka w góre na czarny, Aplikacja wyświetla opinię z zaktualizowanym stanem akceptacji pozytywnych i negatywnych.
Alternatywny przebieg:	<p>Dodanie negatywnej akceptacji</p> <ol style="list-style-type: none"> a. Użytkownik kliką przy jednej z opinii kciuk w dół, a. Aplikacja zwiększa o 1 liczbę negatywnych akceptacji wybranej opinii, a. Aplikacja zmienia kolor kciuka w dół na czarny, a. Aplikacja wyświetla opinię z zaktualizowanym stanem akceptacji pozytywnych i negatywnych.

Tab. 2.28: Opis przypadku użycia - Usunięcie akceptacji opinii o artykule

Nazwa:	Usunięcie akceptacji opinii o artykule
Cel:	Usunięcie akceptacji opinii o artykule

Aktorzy:	Autor akceptacji opinii o artykule
Warunki początkowe:	Wyświetlone opinie o artykule (Rys. 2.45)
Warunki końcowe:	Wyświetlenie opinii z aktualizowanym stanem akceptacji pozytywnych i negatywnych
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką w czerwony znak „X” przy akceptacjach wybranej opinii, 2. Aplikacja usuwa akceptację opinii, 3. Aplikacja zmienia kolor ustawionego kciuka na biały, 4. Aplikacja wyświetla opinię z aktualizowanym stanem akceptacji pozytywnych i negatywnych.

Tab. 2.29: Opis przypadku użycia - Wyszukiwanie technologii

Nazwa:	Wyszukiwanie technologii
Opis:	Wyszukiwanie technologii według podanych przez użytkownika kryteriów i wyświetlenie wyników tego wyszukiwania w postaci listy technologii
Aktorzy:	Użytkownik
Warunki początkowe:	Brak
Warunki końcowe:	Wyświetlenie listy wyszukanych technologii
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Technologie” znajdujący się w panelu nawigacji strony (Rys. 2.14), 2. Aplikacja wyświetla panel do wyszukiwania technologii, 3. Użytkownik uzupełnia kryteria wyszukiwania, 4. Użytkownik kliką przycisk „Szukaj”, 5. Serwis wyszukuje technologii, 6. Aplikacja wyświetla listę wyszukanych technologii (Rys. 2.48).

Artykuły	Technologie
----------	-------------

Wyszukiwanie technologii

Nazwa	<input type="text" value="Java"/>
Kategoria	<input type="text" value="Język programowania"/>
Zastosowania	<input type="text" value="Backend"/>
Dostawca	<input type="text" value="Oracle"/>
Data powstania	<input type="text" value="02.03.1950"/> <input type="text" value="03.05.2022"/>
Data modyfikacji	<input type="text" value="09.07.2020"/> <input type="text" value="28.08.2022"/>

Szukaj

Sortowanie	<input style="width: 150px;" type="text"/> <input type="button" value="<"/> <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="..."/> <input type="button" value="8"/> <input type="button" value=">"/>
------------	--

Java	Języki programowania	Oracle	21:00, 20.08.2022
C++	Języki programowania		20:00, 16.06.2020
Javascript	Języki programowania		22:00, 21.08.2022
React	Framework		23:00, 22.09.2022

Sortowanie	<input style="width: 150px;" type="text"/> <input type="button" value="<"/> <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="..."/> <input type="button" value="8"/> <input type="button" value=">"/>
------------	--

Rys. 2.48: Panel do wyszukiwania technologii z podanymi kryteriami wyszukiwania oraz listą wyszukanych technologii

Źródło: opracowanie własne

Tab. 2.30: Opis przypadku użycia - Sortowanie technologii

Nazwa:	Sortowanie technologii
Opis:	Posortowanie wyszukanych technologii według opcji sortowania wybranej przez użytkownika oraz wyświetlenie listy tych posortowanych technologii
Aktorzy:	Użytkownik
Warunki początkowe:	Wyświetlona lista wyszukanych technologii (Rys. 2.48)
Warunki końcowe:	Wyświetlenie listy posortowanych technologii
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik kliką na rozsuwaną listę „Sortowanie”, 2. Użytkownik wybiera jedną z opcji sortowania (Rys. 2.49), 3. Aplikacja sortuje listę technologii według wybranej opcji sortowania, 4. Aplikacja wyświetla listę posortowanych technologii.

Sortowanie	▼
Liczba opinii: malejąco	
Liczba opinii: rosnąco	
Data pierwszego wydania: malejąco	
Data pierwszego wydania: rosnąco	
Data ostatniego wydania: malejąco	
Data ostatniego wydania: rosnąco	
Średnia ocena: malejąco	
Średnia ocena: rosnąco	

Rys. 2.49: Dostępne opcje sortowania technologii

Źródło: opracowanie własne

Tab. 2.31: Opis przypadku użycia - Wyświetlenie opisu technologii

Nazwa:	Wyświetlenie opisu technologii
Cel:	Wyświetlenie opisu technologii
Aktorzy:	Użytkownik
Warunki początkowe:	Brak
Warunki końcowe:	Wyświetlenie opisu technologii
Przebieg:	<ol style="list-style-type: none"> 1. Użytkownik wyszukuje technologie (Tab. 2.29) 2. Użytkownik kliką na jedną z wyszukanych technologii, 3. Aplikacja wyświetla opis technologii (Rys. 2.50).
Alternatywny przebieg:	<p>Użytkownik wyświetla opis technologii poprzez artykuł</p> <ol style="list-style-type: none"> 1a. Użytkownik wyszukuje artykuły (Tab. 2.9), 2a. Użytkownik wyświetla zawartość artykułu (Tab. 2.11) zawierającą na początku opis technologii, której artykuł dotyczy.

Data pierwszego wydania: 01.01.1995, 12:25:32

Data ostatniego wydania: 22.03.2022, 21:32:45

Technologia → Język programowania → Java



Dostawca: Oracle

Opis:

Java jest to wspólnobieżny, obiektowy język programowania. Język ten jest niezwykle popularny oraz uniwersalny i dlatego trudnym zadaniem byłoby znalezienie dziedziny na pograniczu informatyki, w której problemach nie można by było wykorzystać Javy. Jednym z powodów uniwersalności tego języka jest to, że programy napisane w Javie są uruchamiane na maszynie wirtualnej, co też powoduje, że aby uruchomić program napisany w Javie, wystarczy posiadać zaistalowane JRE (kod bajtowy klas standardowych razem z maszyną wirtualną). Obecnie Java jest wykorzystywana przede wszystkim jako backend w aplikacjach webowych i mobilnych.

Rys. 2.50: Panel z zawartością technologii

Źródło: opracowanie własne

Tab. 2.32: Opis przypadku użycia - Edycja klasyfikacji technologii

Nazwa:	Edycja klasyfikacji technologii
Cel:	Dodanie nowej technologii, modyfikacja albo usunięcie istniejącej technologii
Aktorzy:	Recenzent
Warunki początkowe:	Wyświetlony panel do wyszukiwania technologii (Rys. 2.51)
Warunki końcowe:	Dodanie nowej technologii, modyfikacja albo usunięcie istniejącej technologii

Przebieg:	<p>Dodawanie technologii</p> <ol style="list-style-type: none"> 1. Użytkownik kliką „Dodaj technologię”, 2. Aplikacja wyświetla formularz do utworzenia technologii, 3. Użytkownik wypełnia wszystkie pola w formularzu (Rys. 2.52) 4. Użytkownik kliką przycisk „Zapisz”, 5. Serwis przeprowadza validację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 6. Po pozytywnej validacji danych, aplikacja tworzy technologię, 7. Aplikacja wyświetla użytkownikowi wygląd utworzonej technologii.
Alternatywny przebieg:	<p>Użytkownik anuluje dodanie technologii</p> <ol style="list-style-type: none"> 3a. Użytkownik kliką przycisk „Anuluj”, 4a. Użytkownik zostaje przekierowany do strony z wyszukiwaniem technologii (Rys. 2.51).
Alternatywny przebieg:	<p>Użytkownik nie uzupełnia wszystkich pól formularza podczas dodawania technologii</p> <ol style="list-style-type: none"> 6b. Aplikacja zmienia kolor obramowania pól, które nie przeszły validacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 7b. Przejście do punktu 3.
Alternatywny przebieg:	<p>Edytowanie technologii</p> <ol style="list-style-type: none"> 1c. Użytkownik wyszukuje technologie (Tab. 2.29), 2c. Użytkownik wyświetla opis technologii (Tab. 2.31), 3c. Użytkownik kliką przycisk „Edytuj”, 4c. Aplikacja wyświetla formularz do edytowania technologii, 5c. Użytkownik wypełnia wszystkie pola w formularzu (Rys. 2.52) 6c. Użytkownik kliką przycisk „Zapisz”, 7c. Serwis przeprowadza validację danych pod względem wymaganych pól oraz narzuconych rozmiarów i formatów danych, 8c. Po pozytywnej validacji danych, aplikacja modyfikuje technologię, 9c. Aplikacja wyświetla użytkownikowi wygląd zmodyfikowanej technologii.

Alternatywny przebieg:	Użytkownik anuluje edytowanie technologii 3ca. Użytkownik kliką przycisk „Anuluj”, 4ca. Wyłączenie formularza do edytowania technologii.
Alternatywny przebieg:	Użytkownik nie uzupełnia wszystkich pól formularza podczas edytowania technologii 8cb. Aplikacja zmienia kolor obramowania pól, które nie przeszły walidacji, na czerwono oraz wyświetla pod tymi polami adekwatny komunikat o błędzie, 9cb. Przejście do punktu 5c.
Alternatywny przebieg:	Usunięcie technologii 3cc. Użytkownik kliką przycisk „Usuń”, 4cc. Aplikacja wyświetla formularz z zapytaniem „Czy na pewno ta technologia powinna zostać usunięta?”, 5cc. Użytkownik wybiera opcję „Tak”, 6cc. Serwis usuwa wybraną technologię.
Alternatywny przebieg:	Użytkownik anuluje usunięcie technologii 5cca. Użytkownik wybiera opcję „Nie”, 6cca. Technologia nie zostaje usunięta.

Artykuły	Technologie																
Wyszukiwanie technologii Dodaj technologię																	
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;">Nazwa</td> <td style="width: 85%;"><input type="text" value="Java"/></td> </tr> <tr> <td>Kategoria</td> <td><input type="text" value="Język programowania"/></td> </tr> <tr> <td>Zastosowania</td> <td><input type="text" value="Backend"/></td> </tr> <tr> <td>Dostawca</td> <td><input type="text" value="Oracle"/></td> </tr> <tr> <td>Data powstania</td> <td><input type="text" value="02.03.1950"/> <input type="text" value="03.05.2022"/></td> </tr> <tr> <td>Data modyfikacji</td> <td><input type="text" value="09.07.2020"/> <input type="text" value="28.08.2022"/></td> </tr> </table>		Nazwa	<input type="text" value="Java"/>	Kategoria	<input type="text" value="Język programowania"/>	Zastosowania	<input type="text" value="Backend"/>	Dostawca	<input type="text" value="Oracle"/>	Data powstania	<input type="text" value="02.03.1950"/> <input type="text" value="03.05.2022"/>	Data modyfikacji	<input type="text" value="09.07.2020"/> <input type="text" value="28.08.2022"/>				
Nazwa	<input type="text" value="Java"/>																
Kategoria	<input type="text" value="Język programowania"/>																
Zastosowania	<input type="text" value="Backend"/>																
Dostawca	<input type="text" value="Oracle"/>																
Data powstania	<input type="text" value="02.03.1950"/> <input type="text" value="03.05.2022"/>																
Data modyfikacji	<input type="text" value="09.07.2020"/> <input type="text" value="28.08.2022"/>																
<input type="button" value="Szukaj"/>																	
Sortowanie <input style="width: 50px;" type="text"/> < 1 2 ... 8 >																	
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%;">Java</td> <td style="width: 25%;">Języki programowania</td> <td style="width: 25%;">Oracle</td> <td style="width: 25%;">21:00, 20.08.2022</td> </tr> <tr> <td>C++</td> <td>Języki programowania</td> <td></td> <td>20:00, 16.06.2020</td> </tr> <tr> <td>Javascript</td> <td>Języki programowania</td> <td></td> <td>22:00, 21.08.2022</td> </tr> <tr> <td>React</td> <td>Framework</td> <td></td> <td>23:00, 22.09.2022</td> </tr> </table>		Java	Języki programowania	Oracle	21:00, 20.08.2022	C++	Języki programowania		20:00, 16.06.2020	Javascript	Języki programowania		22:00, 21.08.2022	React	Framework		23:00, 22.09.2022
Java	Języki programowania	Oracle	21:00, 20.08.2022														
C++	Języki programowania		20:00, 16.06.2020														
Javascript	Języki programowania		22:00, 21.08.2022														
React	Framework		23:00, 22.09.2022														
Sortowanie <input style="width: 50px;" type="text"/> < 1 2 ... 8 >																	

Rys. 2.51: Panel do wyszukiwania technologii z podanymi kryteriami wyszukiwania oraz listą wyszukanych technologii widziany z perspektywy recenzenta

Źródło: opracowanie własne

Artykuły	Technologie	Weryfikacja artykułów
----------	-------------	-----------------------

Nazwa	Java *
Kategoria	Języki progr. ▼ *
Dostawca	Oracle
Data ostatniego wydania	01.01.1996
Data ostatniego wydania	22.03.2022
Ikona	<input type="file" value="ikona.png"/> +

Opis

Java jest to współbieżny, obiektowy język programowania. Język ten jest niezwykle popularny oraz uniwersalny i dlatego trudnym zadaniem byłoby znalezienie dziedziny na pograniczu informatyki, w której problemach nie można by było wykorzystać Javy. Jednym z powodów uniwersalności tego języka jest to, że programy napisane w Javie są uruchamiane na maszynie wirtualnej, co też powoduje, że aby uruchomić program napisany w Javie, wystarczy posiadać zaistalowane JRE (kod bajtowy klas standardowych razem z maszyną wirtualną). Obecnie Java jest wykorzystywana przede wszystkim jako backend w aplikacjach webowych i mobilnych.

[Zapisz](#) [Anuluj](#)

Rys. 2.52: Panel z formularzem dodania albo modyfikacji technologii

Źródło: opracowanie własne

Tab. 2.33: Opis przypadku użycia - Wyszukiwanie artykułów z przeterminowaną weryfikacją

Nazwa:	Wyszukiwanie artykułów z przeterminowaną weryfikacją
Cel:	Wyszukanie artykułów z przeterminowaną weryfikacją
Aktorzy:	System
Warunki początkowe:	Minęły 24 godziny od ostatniego wyszukania artykułów z przeterminowaną weryfikacją
Warunki końcowe:	Zwrócenie artykułów z przeterminowaną weryfikacją

Przebieg:	<ol style="list-style-type: none"> 1. Wyszukanie artykułów z przedawnioną weryfikacją, 2. Zapisanie artykułów z przedawnioną weryfikacją do kolejki artykułów do weryfikacji (jeśli nie są jeszcze w tej kolejce), 3. Zwrócenie artykułów znajdujących się w kolejce artykułów do weryfikacji.
-----------	---

Tab. 2.34: Opis przypadku użycia - Przypisanie recenzentowi artykułu do weryfikacji

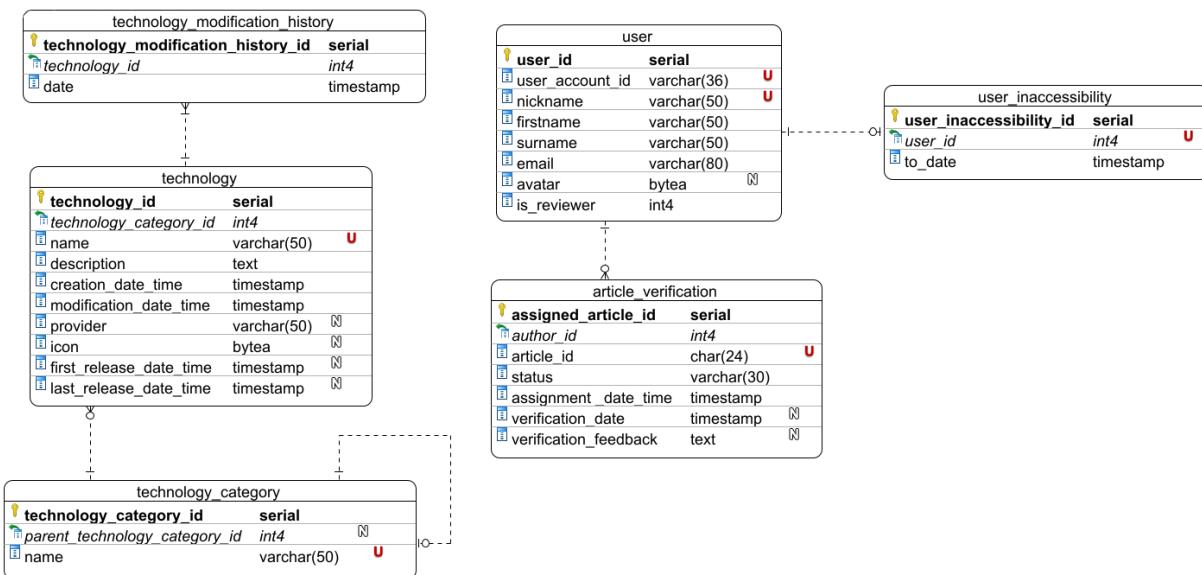
Nazwa:	Przypisanie recenzentowi artykułu do weryfikacji
Cel:	Przypisanie recenzentowi artykułu do weryfikacji
Aktorzy:	System
Warunki początkowe:	Zalogowany użytkownik utworzył artykuł albo autor artykułu dokonał modyfikacji istniejącego artykułu
Warunki końcowe:	Przypisanie wybranemu recenzentowi artykułu do weryfikacji albo oczekiwanie na dostępnego recenzenta
Przebieg:	<p>Jest przynajmniej jeden dostępny recenzent</p> <ol style="list-style-type: none"> 1. Przypisanie wybranemu dostępnemu recenzentowi artykułu do weryfikacji,
Alternatywny przebieg:	<p>Nie ma dostępnych recenzentów</p> <ol style="list-style-type: none"> 1a. System zapisuje artykuł do kolejki artykułów do weryfikacji, 2a. System sprawdza co 24 godziny, czy pojawił się dostępny recenzenta, 3a. Nie ma dalej dostępnych recenzentów.
Alternatywny przebieg:	<p>Pojawił się dostępny recenzent</p> <ol style="list-style-type: none"> 3aa. Powrót do punktu 1.

Tab. 2.35: Opis przypadku użycia - Przypisanie artykułu do weryfikacji innemu recenzentowi

Nazwa:	Przypisanie artykułu do weryfikacji innemu recenzentowi
Cel:	Przypisanie artykułu do weryfikacji innemu recenzentowi
Aktorzy:	System
Warunki początkowe:	Minął termin końcowy weryfikacji artykułu
Warunki końcowe:	Przypisanie innemu recenzentowi artykułu do weryfikacji albo oczekiwanie na pojawienie się dostępnego recenzenta
Przebieg:	<ol style="list-style-type: none"> 1. Przypisanie artykułu do weryfikacji innemu recenzentowi.

2.5. Projekt baz danych

Na podstawie zdefiniowanych wymagań funkcjonalnych oraz szczegółowych opisów przypadków użycia, zaprojektowano diagram związków encji (ERD) dla głównej bazy danych (obiektowo-relacyjna baza) oraz utworzono schematy danych dla bazy artykułów (nierelacyjna dokumentowa baza). Do sporządzenia diagramu ERD wykorzystano narzędzie Visual Paradigm Standard. Diagram ERD dla głównej bazy danych przedstawiono na rysunku 2.53.



Rys. 2.53: Diagram ERD dla głównej bazy danych

Źródło: opracowanie własne

Dla NoSQLowej bazy danych utworzono osobne schematy danych w postaci formatu JSON dla następujących kolekcji: artykuł, komentarz oraz opinia. Komentarze i opinie są przechowywane w oddzielnych kolekcjach, gdyż założono, że użytkownik będzie mógł w jednym momencie przeglądać komentarze albo opinie. Dla zdefiniowanych schematów danych bazy NoSQL utworzono również po jednym przykładowym dokumencie w formacie JSON.

Kod źródłowy schematu artykułów znajduje się na listingu 2.1.

Listing 2.1: Schemat kolekcji artykułów w postaci JSON dla bazy danych artykułów

```

1  {
2      "$jsonSchema": {
3          "bsonType": "object",
4          "title": "articles",
5          "required": [
6              "authorId",
7              "technologyId",
8              "title",
9              "content",
10             "status",
11             "creationDate",
12             "modificationDate"
13         ],
14         "properties": {
15             "_id": {
16                 "bsonType": "objectId"
17             },
18             "authorId": {
19                 "bsonType": "int"
20             },
21             "technologyId": {
22                 "bsonType": "int"
23             },
24             "title": {
25                 "bsonType": "string",
26                 "uniqueItems": true
27             },
28             "content": {
29                 "bsonType": "string"
30             },
31             "status": {
32                 "bsonType": "string",
33                 "enum": [
34                     "NEW",
35                     "ASSIGNING_TO_VERIFICATION",
36                     "VERIFICATION",
37                     "EXPIRED_VERIFICATION",
38                     "REFUSED",
39                     "PUBLISHED",
40                     "EDITED"
41                 ]
42             },
43             "creationDate": {
44                 "bsonType": "date"
45             },
46             "modificationDate": {
47                 "bsonType": "date"
48             },
49             "averageRating": {
50                 "bsonType": "double",
51                 "minimum": 0
52             }
53         }
54     }
55 }
```

Na podstawie schematu artykułów (listing 2.1) utworzono przykładowy dokument, który jest przedstawiony na listingu 2.2

Listing 2.2: Przykładowy dokument dla kolekcji artykułów zgodny ze zdefiniowanym schematem

```
1 {  
2     "_id":{  
3         "$oid":"634308ab8f35533467caa4f7"  
4     },  
5     "authorId":27,  
6     "technologyId":1,  
7     "title":"Podstawy Javy",  
8     "content":"<p style=\"text-align: justify;\">Java jest to wspołczesny, oparty na klasach i mocno typowany obiektowy język programowania...",  
9     "status":"NEW",  
10    "creationDate":{  
11        "$date":{  
12            "$numberLong":1665337515291  
13        }  
14    },  
15    "modificationDate":{  
16        "$date":{  
17            "$numberLong":1665337515294  
18        }  
19    }  
20 }
```

Kod źródłowy schematu komentarzy znajduje się na [listingu 2.3](#).

Listing 2.3: Schemat kolekcji komentarzy w postaci JSON dla bazy danych artykułów

```
1 {  
2     "$jsonSchema": {  
3         "bsonType": "object",  
4         "title": "comment",  
5         "required": [  
6             "articleId",  
7             "authorId",  
8             "content",  
9             "creationDate",  
10            "modificationDate"  
11        ],  
12        "properties": {  
13            "_id": {  
14                "bsonType": "objectId"  
15            },  
16            "articleId": {  
17                "bsonType": "objectId"  
18            },  
19            "parentCommentId": {  
20                "bsonType": "objectId"  
21            },  
22            "authorId": {  
23                "bsonType": "int"  
24            },  
25            "content": {  
26                "bsonType": "string"  
27            },  
28            "creationDate": {  
29                "bsonType": "date"  
30            },  
31            "modificationDate": {  
32                "bsonType": "date"  
33            }  
34        }  
35    }  
36}
```

```
34     }
35   }
36 }
```

Na podstawie schematu komentarzy (listing 2.3) utworzono przykładowy dokument, który jest przedstawiony na listingu 2.4

Listing 2.4: Przykładowy dokument dla kolekcji komentarzy zgodny ze zdefiniowanym schematem

```
1  {
2    "_id": {
3      "$oid": "63618aa9973c760cacac05af"
4    },
5    "articleId": {
6      "$oid": "634308ab8f35533467caa4f7"
7    },
8    "parentCommentId": {
9      "$oid": "63617ca71676af3917d483fd"
10   },
11   "authorId": 27,
12   "content": "Bardzo dobry artykuł. Napisałbyś może coś na temat Springa
13   → ?",
14   "creationDate": {
15     "$date": {
16       "$numberLong": "1667336873772"
17     }
18   },
19   "modificationDate": {
20     "$date": {
21       "$numberLong": "1667336873772"
22     }
23 }
```

Kod źródłowy schematu opinii znajduje się na listingu 2.5

Listing 2.5: Schemat kolekcji opinii w postaci JSON dla bazy danych artykułów

```

1  {
2      "$jsonSchema": {
3          "bsonType": "object",
4          "title": "opinion",
5          "required": [
6              "authorId",
7              "articleId",
8              "rating",
9              "content",
10             "creationDate",
11             "modificationDate"
12         ],
13         "properties": {
14             "_id": {
15                 "bsonType": "objectId"
16             },
17             "authorId": {
18                 "bsonType": "int"
19             },
20             "articleId": {
21                 "bsonType": "objectId"
22             },
23             "rating": {
24                 "bsonType": "int",
25                 "minimum": 1,
26                 "maximum": 5
27             },
28             "content": {
29                 "bsonType": "string"
30             },
31             "creationDate": {
32                 "bsonType": "date"
33             },
34             "modificationDate": {
35                 "bsonType": "date"
36             },
37             "acceptances": {
38                 "bsonType": "array",
39                 "items": {
40                     "bsonType": "object",
41                     "required": [
42                         "authorId",
43                         "value"
44                     ],
45                     "properties": {
46                         "_id": {
47                             "bsonType": "objectId"
48                         },
49                         "authorId": {
50                             "bsonType": "int"
51                         },
52                         "value": {
53                             "bsonType": "int",
54                             "enum": [
55                                 -1,
56                                 1
57                             ]
58                         }
59                     }
60                 }
61             }
62         }
63     }
64 }
```

```

60          }
61      },
62      "positiveAcceptancesCount": {
63          "bsonType": "int"
64      },
65      "negativeAcceptancesCount": {
66          "bsonType": "int"
67      }
68  }
69 }
70 }
```

Na podstawie schematu opinii (listing 2.5) utworzono przykładowy dokument, który jest przedstawiony na listingu 2.6

Listing 2.6: Przykładowy dokument dla kolekcji opinii zgodny ze zdefiniowanym schematem

```

1  {
2      "_id": {
3          "$oid": "639271358c5f5e6eaf655d54"
4      },
5      "author_id": 1,
6      "articleId": {
7          "$oid": "6391195fafa6bb0ab87cbda3"
8      },
9      "rating": 4,
10     "content": "Dobry artykuł, ale brakuje trochę informacji o wersjach
11        → tej technologii",
12     "creationDate": {
13         "$date": {
14             "$numberLong": "1670541621467"
15         }
16     },
17     "modificationDate": {
18         "$date": {
19             "$numberLong": "1670541630198"
20         }
21     },
22     "acceptances": [
23         {
24             "_id": {
25                 "$oid": "6333266e3a8c25217485239c"
26             },
27             "author_id": 2
28         },
29         {
30             "_id": {
31                 "$oid": "6333266e3a8c25217485239d"
32             },
33             "author_id": 3,
34             "value": -1
35         }
36     ],
37     "positiveAcceptancesCount": 0,
38     "negativeAcceptancesCount": 0
39 }
```

Rozdział 3

Implementacja systemu IT Tech

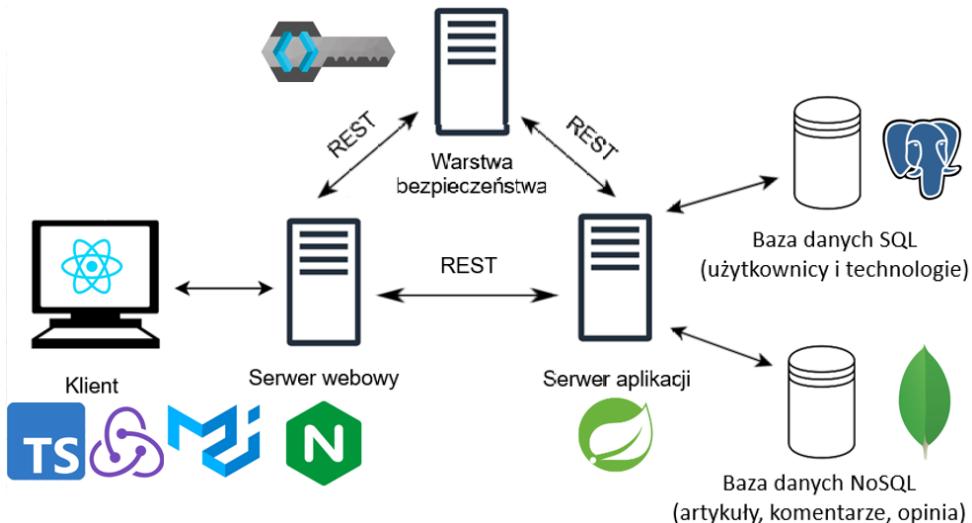
3.1. Technologie i narzędzia

W celu zaimplementowania zaprojektowanego systemu dokonano wyboru technologii i narzędzi. Przy ich wyborze kierowano się głównie ich odpowiedniością do realizowanego systemu, popularnością oraz własnym doświadczeniem uzyskanym w trakcie studiów. Poszukiwano przede wszystkim darmowych rozwiązań, gdyż realizowany w ramach tej pracy projekt jest niekomercyjny. Innymi ważnymi czynnikami wyboru były również rozbudowana i czytelna dokumentacja, niezbyt duży poziom skomplikowania oraz bogata i aktywna społeczność wokół danego oprogramowania.

3.1.1. Technologie

Użyto następujących technologii: Java [4], Spring [5], React [6], Typescript [7], Redux [8], MUI [9], PostgreSQL [10], MongoDB [11], NGINX [12], Keycloak [13].

Wybrane technologie mogą być przyporządkowane do poszczególnych warstw systemu (Rys. 3.1).



Rys. 3.1: Użyte technologie

Źródło: opracowanie własne

Baza danych

Jako system zarządzania obiektowo-relacyjną bazą danych wybrano PostgreSQL. PostgreSQL jest systemem darmowym, otwarto-źródłowym oraz niezwykle popularnym. PostgreSQL to najpopularniejszy RDBMS (ang. *Relational Database Management System*) tego typu, jak i jeden z najpopularniejszych otwartych RDBMS dostępnych na rynku. System ten ciągle się rozwija oraz jest niezawodny dzięki dosyć licznej społeczności zorientowanej wokół niego. Stanowi silną konkurencję dla innych baz danych pod względem bezpieczeństwa. Inną ważną zaletą tej technologii jest jej wysoka skalowalność.

Jako system zarządzania nierelacyjną bazą danych opartą na dokumentach wybrano MongoDB. MongoDB jest to otwarty i popularny system, który pozwala na wygodne przechowywanie danych w postaci klucz-wartość i nie wymaga użycia schematów, ale jest to zalecane. Dane są przechowywane jako dokumenty w postaci JSON. Dużymi zaletami tej technologii jest łatwość tworzenia danych oraz skalowalność. Z powodu wykorzystywanej formuły JSON, technologia ta jest wprost stworzona do użycia razem z językiem JavaScript, który jest wykorzystywany podczas realizacji tej pracy.

Aplikacja serwerowa - Backend

Do realizacji aplikacji serwerowej wybrano język Java. Java jest wieloplatformowym, obiektowym i opartym na maszynie wirtualnej niezwykle popularnym językiem programowania. Język wykorzystuje się w bardzo wielu dziedzinach, jednak najczęściej są to aplikacje webowe oraz mobilne. Z powodu popularności tej technologii oraz jej długiego czasu na rynku (od 1996 roku), zebrała się dosyć duża społeczność wokół niej. Java posiada dużą liczbę bibliotek oraz frameworków, które przyspieszają pracę programisty.

Do realizacji aplikacji wykorzystano popularny Javowy framework Spring. Spring pozwala głównie na tworzenie backendu aplikacji webowych i jest obecnie bezkonkurencyjny w przypadku aplikacji tego typu. Pisanie aplikacji w Springu jest proste. Spring posiada bogatą i czytelną dokumentację wraz z przykładami użycia. Wokół Springa zebrała się duża społeczność. Inną zaletą tego framework jest to, że tak samo jak w przypadku Javy, dostępna jest duża liczba różnych bibliotek. Wadą tej technologii jest dosyć skomplikowana konfiguracja startowa aplikacji, lecz na szczęście istnieją technologie, które dostarczają tę konfigurację za programistę. Jedną z takich technologii jest framework Spring Boot, który dostarcza całą konfigurację aplikacji łącznie serwerem i od razu możliwe jest uruchomienie prostego backendu.

Klient - Frontend

Do realizacji aplikacji klienckiej wybrano JavaScriptową bibliotekę React. React jest to popularna technologia pozwalająca tworzyć interfejsy graficzne aplikacji webowych. Głównym zastosowaniem tej technologii są aplikacje typu SPA. Tworzenie interfejsu graficznego w Reactcie jest proste i intuicyjne. Interfejs graficzny użytkownika tworzy się z tzw. komponentów. Komponenty pozwalają m.in. na ich ponowne użycie i dzięki temu można uniknąć niepotrzebnego powielania kodu.

JavaScript jest językiem słabo typowanym i przez to w trakcie komplikacji programu może być niewykryte wiele błędów. Mając na uwadze tę wadę, postanowiono wykorzystać technologię, która będzie umożliwiała wprowadzenie silnego typowania do aplikacji. Do tego celu wykorzystano Typescript, który jest nakładką na JavaScript. Typescript pozwala m.in. na tworzenie interfejsów, czy typowanie zmiennych.

W realizowanej aplikacji przewidziano system logowania oparty na tokenach na okaziciela i z tego powodu niezbędnym jest przechowywanie tych tokenów, aby możliwe było wysyłanie

żądań do zabezpieczonego backendu. Do tego celu idealnie pasuje Reactowa biblioteka Redux, która umożliwia przechowywanie odizolowanego od komponentów stanu aplikacji.

Do tworzenia ładnego interfejsu użytkownika wykorzystano popularną bibliotekę gotowych komponentów MUI. MUI zawiera bardzo dużą liczbę gotowych komponentów, których użycie jest proste. Biblioteka wspiera m.in. internacjonalizację, czy lokalizację.

Serwer webowy

Do realizacji serwera webowego klienta wybrano serwer NGINX. NGINX jest to darmowy i popularny serwer webowy, proxy oraz IMAP/POP3. Serwer ten oparty jest na zdarzeniach (ang. *event-driven approach*), co oznacza, że może obsłużyć wiele żądań na jednym wątku. Serwer ten jest często wykorzystywany przez duże serwisy, jak np. Netflix, Instagram, czy GitHub.

Warstwa bezpieczeństwa

Do realizacji warstwy bezpieczeństwa wybrano technologię Keycloak. Keycloak jest to otwarty system, który implementuje uwierzytelnianie i autoryzacje. Keycloak wspiera logowanie SSO (ang. *single sign-on*), które umożliwia zalogowanie się do wielu serwisów za pomocą jednego konta. Keycloak jest najczęściej wykorzystywany w formie osobnego serwera.

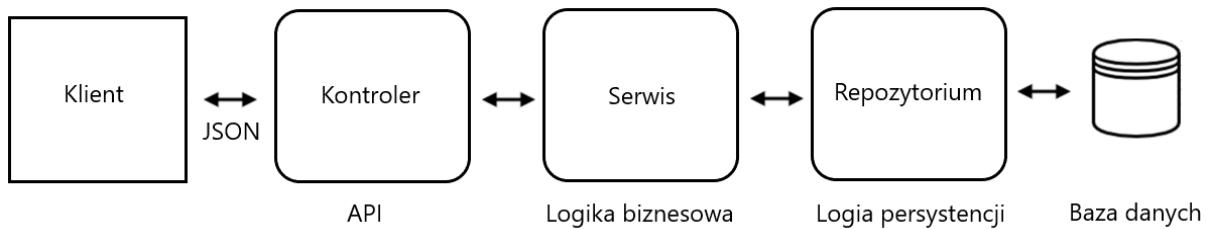
3.1.2. Narzędzia

Użyto następujących narzędzi:

1. LaTeX [14] - napisanie dokumentacji,
2. Visual Paradigm [15] - sporządzenie diagramów,
3. Figma [16] - zaprojektowanie interfejsu użytkownika,
4. pgAdmin [17] - zarządzanie bazą PostgreSQL,
5. MongoDBCompass [18] - zarządzanie bazą MongoDB,
6. Visual Studio Code [19] - środowisko programistyczne dla frontendu,
7. ItelliJ [20] - środowisko programistyczne dla backendu,
8. Postman [21] - testowanie wysyłania żądań do backendu,
9. REST-assured [22] - testy integracyjne backendu,
10. Swagger [23] - dokumentacja endpointów backendu,
11. Docker [24] - stworzenie i umieszczenie kontenerów aplikacji w rejestrze kontenerów,
12. MongoDB Cloud [25] - hostowanie bazy MongoDB,
13. Azure [26] - hostowanie całego systemu oprócz bazy MongoDB.

3.2. Aplikacja serwerowa

Aplikacja serwerowa jest dosyć złożonym systemem, który odpowiada za wiele różnych funkcji np. udostępnianie endpointów klientowi, czy zarządzanie bazą danych. Z powodu tej złożoności aplikacja serwerowa została podzielona na warstwy (Rys. 3.2). Każda warstwa odpowiada za odrębne funkcje. Komunikacja może się odbywać tylko pomiędzy sąsiednimi warstwami.



Rys. 3.2: Architektura aplikacji serwerowej

Źródło: opracowanie własne

Aplikacja serwerowa wykorzystuje mapowanie obiektowo-relacyjne (ang. *ORM*) poprzez framework **Hibernate** i dzięki temu możliwe jest wygodne operowanie na obiektach zamiast ręcznego wywoływania mniej czytelnych i mniej intuicyjnych zapytań do bazy danych.

Warstwa bazy danych odpowiada za zarządzanie bazą danych. Warstwa ta operuje na tzw. encjach (modelach), czyli odpowiednio oznaczonych klasach Javy, które odpowiadają tabelom albo dokumentom w bazie danych. W systemie można wyróżnić dwa rodzaje encji: encje przechowywane przez bazę PostgreSQL oraz encje przechowywane przez bazę MongoDB.

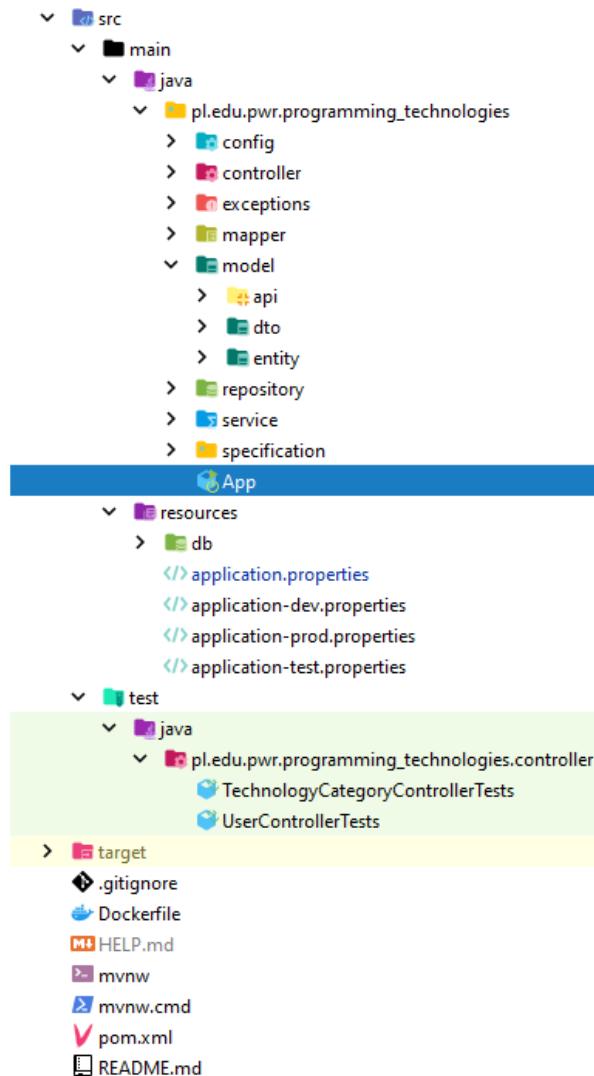
Repozytorium jest ściśle powiązane z warstwą bazy danych. Warstwa ta nakłada na bazę danych abstrakcję (logikę persystencji), która pozwala programistie na wywoływanie metod zamiast ręcznego pisania zapytań do bazy danych. Do repozytorium jest przypisana jedna encja wraz z jej kluczem podstawowym i repozytorium operuje jedynie na tej jednej encji. Tak samo jak w przypadku modeli, można wyróżnić dwa rodzaje repozytoriów: repozytoria operujące na bazie PostgreSQL oraz repozytoria operujące MongoDB.

Warstwa serwisu implementuje logikę biznesową systemu. Serwisy są to odpowiednio oznaczone klasy Javy, których metody implementują poszczególne funkcjonalności systemu od strony backendu. Serwisy korzystają z repozytoriów.

Warstwa kontrolera odpowiada za udostępnianie klientowi usług dostarczanych przez serwisy (np. utworzenie artykułu). Kontroler reprezentuje grupę endpointów i każdy endpoint dostarcza jedną z usług. Klient może skorzystać z danej usługi poprzez wysłanie żądania na endpoint powiązany z tą usługą.

Warstwa kliencka jest to aplikacja kliencka, która wysyła żądania do kontrolera, w celu skorzystania z udostępnionych przez niego usług.

Struktura aplikacji serwerowej jest przedstawiona na rysunku 3.3



Rys. 3.3: Struktura projektu aplikacji serwerowej

Źródło: opracowanie własne

Struktura aplikacji serwerowej jest zgodna z przedstawioną architekturą (Rys. 3.2). Backend jest oparty na narzędziu automatyzującym budowanie aplikacji Maven. Zależności oraz konfiguracja uruchomieniowa aplikacji są zapisane w pliku pom.xml. Fragment zawartości pliku pom.xml przedstawiony jest na listingu 3.1.

Listing 3.1: Repozytorium technologii

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3
    .org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.
        apache.org/xsd/maven-4.0.0.xsd">
    ...
    <groupId>pl.edu.pwr</groupId>
    <artifactId>technologie-programistyczne</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>technologie-programistyczne</name>
    <description>
        Praca inżynierska - aplikacja internetowa do gromadzenia i
        udostępniania informacji o technologiach programistycznych
    </description>
    <properties>
```

```

<java.version>17</java.version>
<springboot.version>${project.parent.version}</springboot.version>
<org.mapstruct.version>1.5.2.Final</org.mapstruct.version>
<org.projectlombok.version>1.18.24</org.projectlombok.version>
<maven-compiler-plugin.version>3.8.1</maven-compiler-plugin.version>
    ↵
</properties>
<dependencies>
    <!-- SPRING -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <version>${springboot.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <version>${springboot.version}</version>
        <scope>test</scope>
    </dependency>
    ...
<dependencies>
...
</project>

```

Na listingu 3.1 deklarowane są m.in. podstawowe parametry programu jak nazwa, czy wersja oraz ustalane są zależności wykorzystywane przez aplikację (znacznik dependency).

Konfiguracja aplikacji (np. link do bazy danych, czy numer portu na jakim działa aplikacja) jest zapisana w plikach o rozszerzeniu properties. Można wyróżnić 3 różne profile konfiguracyjne aplikacji: lokalny, produkcyjny oraz do testowania. Profil lokalny jest wykorzystywany w trakcie rozwoju aplikacji i w konfiguracji tej używane są lokalne bazy danych oraz jest możliwość modyfikowania struktury tabel (schematów dokumentów). Profil produkcyjny jest wykorzystywany przez wdrożoną aplikację i w konfiguracji tej używane są zdalne bazy danych oraz nie ma możliwości modyfikowania struktury tabel (schematów dokumentów). Profil do testowania jest wykorzystywany do testów aplikacji i w konfiguracji tej używana jest baza danych h2, która jest przechowywana w pamięci.

Przykładowy profil lokalny jest przedstawiony na listingu 3.2.

Listing 3.2: Profil lokalny

```

server.port=9000
spring.datasource.url = jdbc:postgresql://localhost:5432/
    ↵ programming_technologies
spring.datasource.username = programming_technologies
spring.datasource.password = postgres

spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=technologie-it
spring.data.mongodb.username=programming_technologies
spring.data.mongodb.password=mongodb

spring.jpa.hibernate.ddl-auto = update

spring.security.oauth2.resourceserver.jwt.jwk-set-uri=http://localhost
    ↵ :8080/realm/Programming_technologies/protocol/openid-connect/certs

frontend.url=http://localhost:3000

```

Fragment przykładowej encji technologii przechowywanej przez bazę PostgreSQL przedstawiono na listingu 3.3.

Listing 3.3: Encja technologii

```
@Builder
@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Table(name="TECHNOLOGIES")
public class TechnologyEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "technology_id")
    private Integer id;

    @Column(name = "name", nullable = false, unique = true)
    private String name;

    @Column(name = "description", nullable = false)
    private String description;

    @Column(name = "creation_date_time", nullable = false)
    private LocalDateTime creationDateTime;

    ...

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "technology_category_id")
    private TechnologyCategoryEntity technologyCategoryEntity;
}
```

Aby klasy były interpretowane jako encje i mogły być przechowywane w bazie PostgreSQL, należy nadać takiej klasie adnotację `Entity`. Wymagane jest również, aby encje posiadały zdefiniowane klucze podstawowe (Id). Klucze podstawowe są generowane automatycznie przez PostgreSQL i oznaczono to za pomocą wartości `Identity`. Założono, że encje będą oparte na zdefiniowanych wcześniej schematach bazy danych, dlatego każdy z atrybutów encji należą do dostosować do odpowiadających im pól na schemacie za pomocą odpowiedniej konfiguracji adnotacji `Column`. Przykładowo atrybut `creationDateTime` odpowiada polu o nazwie `creation_date_time` po stronie bazy danych. Ważnym elementem w przypadku stosowania ORM jest definiowanie relacji. Użyte technologie pozwalają na proste definiowanie relacji między encjami, które ogranicza się do zastosowania kilku adnotacji np. relacja wiele do jeden między technologiami i kategoriami technologii. W przypadku relacji można również m.in. zdefiniować kaskadowe aktualizowanie i usuwanie podrzędnej encji (np. relacja jeden do wielu technologii względem kategorii technologii 3.4), co okazało się być bardzo przydatne.

Listing 3.4: Mapowanie jeden do wielu między technologiami i kategoriami technologii

```
@Transient
@OneToMany(mappedBy="technologyCategoryEntity", fetch = FetchType.LAZY,
    ↪ cascade = CascadeType.ALL, orphanRemoval=true)
private List<TechnologyEntity> technologyEntityList;
```

Fragment przykładowej encji artykułu przechowywanej przez bazę MongoDB przedstawiono na listingu 3.5.

Listing 3.5: Encja artykułu

```

@Builder
@Data
@NoArgsConstructor
@AllArgsConstructor
@Document("articles")
public class ArticleEntity {

    ...

    @Id
    @MongoId(FieldType.OBJECT_ID)
    @Field(name = "_id")
    private ObjectId id;

    @NotNull
    @Field(name = "authorId")
    private Integer authorId;

    @NotNull
    @Field(name = "technologyId")
    private Integer technologyId;

    @NotNull
    @Indexed(unique = true)
    @Field(name = "title")
    private String title;

    @NotNull
    @Field(name = "content")
    private String content;

    @NotNull
    @Enumerated(EnumType.STRING)
    @Field(name = "status")
    private Status status;

    ...

    @Min(0)
    @Field(name = "averageRating")
    private Double averageRating;
}

```

Aby klasy były interpretowane jako encje i mogły być przechowywane w bazie MongoDB, należy nadać takiej klasie adnotację Document. Tak samo jak w przypadku PostgreSQL, wymagane jest aby encje posiadały zdefiniowane klucze podstawowe, ale klucze te powinny być typu ObjectId oraz muszą być one jeszcze oznaczone adnotacją MongoId. Klucze podstawowe są również generowane automatycznie przez MongoDB. Podobnie i w tym przypadku encje są oparte na zdefiniowanych wcześniej schematach bazy danych i każdy z atrybutów encji dostosowano do odpowiadających im pól na schematach, ale służyła do tego adnotacja Field, a nie Column. Niestety, ale przy użyciu frameworka Spring dla baz NoSQL, nie jest możliwe tworzenie relacji opartych na ORM między encjami. Rozwiązaniem tego problemu jest przechowywanie identyfikatorów powiązanych encji (np. authorId) oraz zapewnienie integralności danych po stronie backendu.

Fragment przykładowego repozytorium technologii operującego na bazie PostgreSQL przedstawiono na listingu 3.6.

Listing 3.6: Repozytorium technologii

```

@Repository
public interface TechnologyRepository extends JpaRepository<
    ↪ TechnologyEntity, Integer> {

    List<TechnologyEntity> findAllByTechnologyCategoryId(Integer
        ↪ technologyCategoryId);
    List<TechnologyEntity> findAllByProviderContainsIgnoreCase(String
        ↪ provider);

    default List<TechnologyEntity> findAllHavingTechnologyCategoryIdInTree(
        ↪ Integer technologyCategoryId){
        return findAll()
            .stream()
            .filter(t -> {
                Optional<TechnologyCategoryEntity> opt = Optional.
                    ↪ ofNullable(t.getTechnologyCategoryEntity());
                while(opt.isPresent()) {
                    if(opt.get().getId() == technologyCategoryId){
                        return true;
                    }
                    opt = Optional.ofNullable(opt.get().
                        ↪ getParentTechnologyCategoryEntity());
                }
            }
            .return false;
        })
        .collect(Collectors.toList());
    }
}

```

Aby interfejsy były interpretowane jako repozytoria, muszą być one oznaczone anotacją `Repository`. Dla baz relacyjnych dziedziczenie repozytoriów po interfejsie `JpaRepository`, pozwala na uzyskanie dostępu do kilku automatycznie wygenerowanych metod np. metodę do pobierania wszystkich encji. Inną funkcją tego dziedziczenia jest możliwość wygenerowania metod na podstawie ich samych deklaracji. Przykładowo deklaracja `findAllByTechnologyCategoryId` pozwoli na wygenerowanie definicji metody zwracającej listę technologii o danej kategorii.

Fragment przykładowego repozytorium artykułów operującego na bazie MongoDB przedstawiono na listingu 3.7.

Listing 3.7: Repozytorium artykułów

```

@Repository
public interface ArticleRepository extends MongoRepository <ArticleEntity ,
    ↪ ObjectId> {

    boolean existsByTitleIgnoreCase(String title);
    List<ArticleEntity> findAllByStatusOrStatus(ArticleEntity.Status
        ↪ status1, ArticleEntity.Status status2);
}

```

Podobnie jest w przypadku repozytoriów operujących na bazie MongoDB. Jedyną różnicą jest to, że repozytorium powinno dziedziczyć po interfejsie `MongoRepository`.

Fragment przykładowego serwisu artykułów przedstawiono na listingu 3.8.

Listing 3.8: Serwis artykułów

```

@Slf4j
@Service
@RequiredArgsConstructor
public class ArticleServiceImpl implements ArticleService {

```

```

private final ArticleRepository articleRepository;
private final CommentRepository commentRepository;
private final UserRepository userRepository;
private final TechnologyRepository technologyRepository;

...

@Override
public ArticleEntity getArticleById(ObjectId articleId) throws
    EntityNotFoundException{

    Optional<ArticleEntity> foundArticleOpt = articleRepository.
        ↪ findById(articleId);

    if(foundArticleOpt.isEmpty()){
        throw new EntityNotFoundException("Nie istnieje artykuł o takim
            ↪ id");
    }

    return foundArticleOpt.get();
}

...
}

```

Aby klasy były interpretowane jako serwisy, muszą być one oznaczone adnotacją `Service`. Serwis artykułów (listing 3.8) korzysta z repozytoriów operujących na artykułach, komentarzach, użytkownikach oraz technologiach. W serwisach zaimplementowana jest obsługa wyjątków i np. w przypadku nie podania id artykułu w metodzie `getArticleById`, zostanie rzucany wyjątek z komunikatem „Nie istnieje artykuł o takim id”. Wyjątki rzucane przez serwisy są później obsługiwane przez kontrolery.

Fragment przykładowego kontrolera artykułów przedstawiono na listingu 3.9.

Listing 3.9: Kontroler artykułów

```

@RestController
@CrossOrigin(origins = {"http://localhost:3000", "https://technologie-
    ↪ programistyczne.azurewebsites.net"})
@RequiredArgsConstructor
@RequestMapping(value = "/article")
public class ArticleController {

    private final ArticleService articleService;
    private final CommentService commentService;
    private final CommentMapper commentMapper = CommentMapper.INSTANCE;
    private final ArticleMapper articleMapper = ArticleMapper.INSTANCE;
    private final SearchCriteriaMapper searchCriteriaMapper =
        ↪ SearchCriteriaMapper.INSTANCE;
    private final UserRepository userRepository;
    private final TechnologyRepository technologyRepository;

    ...

    @GetMapping("/{articleId}")
    public ResponseEntity getArticleById(@PathVariable("articleId") String
        ↪ articleIdStr){

        if(!ObjectId.isValid(articleIdStr)){

```

```

    return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(""
        ↪ Podano niewłaściwe id artykułu");
}

ObjectId articleId = new ObjectId(articleIdStr);
ArticleEntity foundArticle;

try{
    foundArticle = articleService.getArticleById(articleId);
}
catch(EntityNotFoundException e){
    return ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.
        ↪ getMessage());
}

ArticleDTO foundArticleDTO = articleMapper.
    ↪ articleEntityToArticleDTO(
        foundArticle, userRepository, technologyRepository
);

return ResponseEntity.ok(foundArticleDTO);
}

...
}

```

Aby klasy były interpretowane jako kontrolery, muszą być one oznaczone adnotacją `Controller` albo `RestController`. Adnotacja `RestController` zawiera w sobie adnotację `Controller` oraz dodatkowo definiuje, że ciała żądań i odpowiedzi będą w formacie JSON. Kontroler artykułów (listing 3.8) korzysta przede wszystkim z serwisów operujących na repozytoriach artykułów i repozytoriach komentarzy. Endpointy przedstawionego kontrolera artykułów zaczynają się od członu `article` i mogą z nich korzystać jedynie klient lokalny i produkcyjny (adnotacja `CrossOrigin`). W warstwie kontrolerów została zaimplementowana obsługa błędów (wyjątków) rzucanych przez serwisy. W przypadku otrzymania błędu zwracany jest status http odpowiadający temu błędu wraz z wiadomością otrzymaną z serwisu. Przykładowo metoda `getArticleById` zwróci status 400 w przypadku podania niewłaściwego statusu artykułu. Kontrolery wysyłają użytkownikowi w odpowiedzi encje zmapowane do DTO (ang. *Data Transfer Object*), gdyż zazwyczaj obiekty pożądane przez klienta różnią się od encji, nie ma konieczności wysyłania wszystkich danych z encji oraz potrzebne są mapowania pomiędzy datą lokalną i datą ze strefą czasową.

Fragment przykładowego mappera artykułów przedstawiono na listingu 3.10.

Listing 3.10: Mapper artykułów

```

@Mapper(uses = {DateTimeMapper.class, UserMapper.class, TechnologyMapper.
    ↪ class, MongoObjectIDMapper.class})
public interface ArticleMapper {

    ArticleMapper INSTANCE = Mappers.getMapper(ArticleMapper.class);

    ...

    @Mapping(source = "id", target = "id", qualifiedByName = ""
        ↪ objectIdToHexString")
    @Mapping(
        source = "creationDate",
        target = "creationDate",
        qualifiedByName = "localDateTimeToOffsetDateTime"
)
}

```

```

)
@Mapping(
    source = "modificationDate",
    target = "modificationDate",
    qualifiedByName = "localDateTimeToOffsetDateTime"
)
@Mapping(source = "authorId", target = "authorDTO", qualifiedByName = "
    ↪ userIdToUserDTO")
@Mapping(
    source = "technologyId",
    target = "technologyDTO",
    qualifiedByName = "technologyIdToTechnologyDTO"
)
@Mapping(source = "status", target = "status")
ArticleDTO articleEntityToArticleDTO(
    ArticleEntity articleEntity,
    @Context UserRepository userRepository,
    @Context TechnologyRepository technologyRepository
);
...
}

```

Do mapowania obiektów wykorzystano technologię **Mapstruct**, która pozwala na m.in. automatyczne generowanie mapowania obiektów na podstawie jedynie deklaracji metod oraz annotacji tych tych metod. Przykładowo na listingu 3.10 przedstawiono mapowanie encji artykułu do DTO artykułu.

Endpointy kontrolerów są zabezpieczone przez Keycloak i przez to wysłanie żądania na dany endpoint będzie możliwe jedynie w przypadku posiadania przez użytkownika odpowiedniej roli. Fragment konfiguracji zabezpieczeń dla endpointów przedstawiono na listingu 3.11.

Listing 3.11: Zabezpieczenie endpointów

```

@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws
    ↪ Exception {
    CorsConfiguration configuration = new CorsConfiguration();
    configuration.setAllowedOrigins(Arrays.asList(
        "http://localhost:3000", "https://technologie-
            ↪ programistyczne.azurewebsites.net")
    );
    configuration.setAllowedMethods(Collections.singletonList("*"))
        ↪ ;
    configuration.setAllowedHeaders(Collections.singletonList("*"))
        ↪ ;

    http.csrf().disable().cors().configurationSource(request ->
        ↪ configuration);
    http.authorizeRequests(authz -> authz
        .antMatchers(HttpMethod.GET, "/user/**")
        .permitAll()
        .antMatchers(HttpMethod.POST, "/user")
        .permitAll()
        .antMatchers(HttpMethod.PUT, "/user/**")
        .hasRole("logged_user")
        ...
        .anyRequest()
        .authenticated())
        .oauth2ResourceServer()
        .jwt()

```

```

        .jwtAuthenticationConverter(
            ↪ jwtAuthenticationConverter());
    return http.build();
}

```

Keycloak przesyła role wewnątrz tokenu JWT (ang. *JSON Web Token*) w innym niż jest to zazwyczaj miejscu i przez to po stronie backendu należało wskazać, gdzie znajdują się role. Konfigurację tę przedstawiono na listingu 3.12.

Listing 3.12: Konfiguracja JWT

```

private JwtAuthenticationConverter jwtAuthenticationConverter() {
    Converter<Jwt, Collection<GrantedAuthority>>
        ↪ jwtGrantedAuthoritiesConverter = jwt -> {
        Map<String, Collection<String>> realmAccess = jwt.
            ↪ getClaim("realm_access");
        Collection<String> roles = realmAccess.get("roles");
        return roles.stream()
            .map(role -> new SimpleGrantedAuthority
                ↪ ("ROLE_" + role))
            .collect(Collectors.toList());
    };

    var jwtAuthenticationConverter = new JwtAuthenticationConverter
        ↪ ();
    jwtAuthenticationConverter.setJwtGrantedAuthoritiesConverter(
        ↪ jwtGrantedAuthoritiesConverter);

    return jwtAuthenticationConverter;
}

```

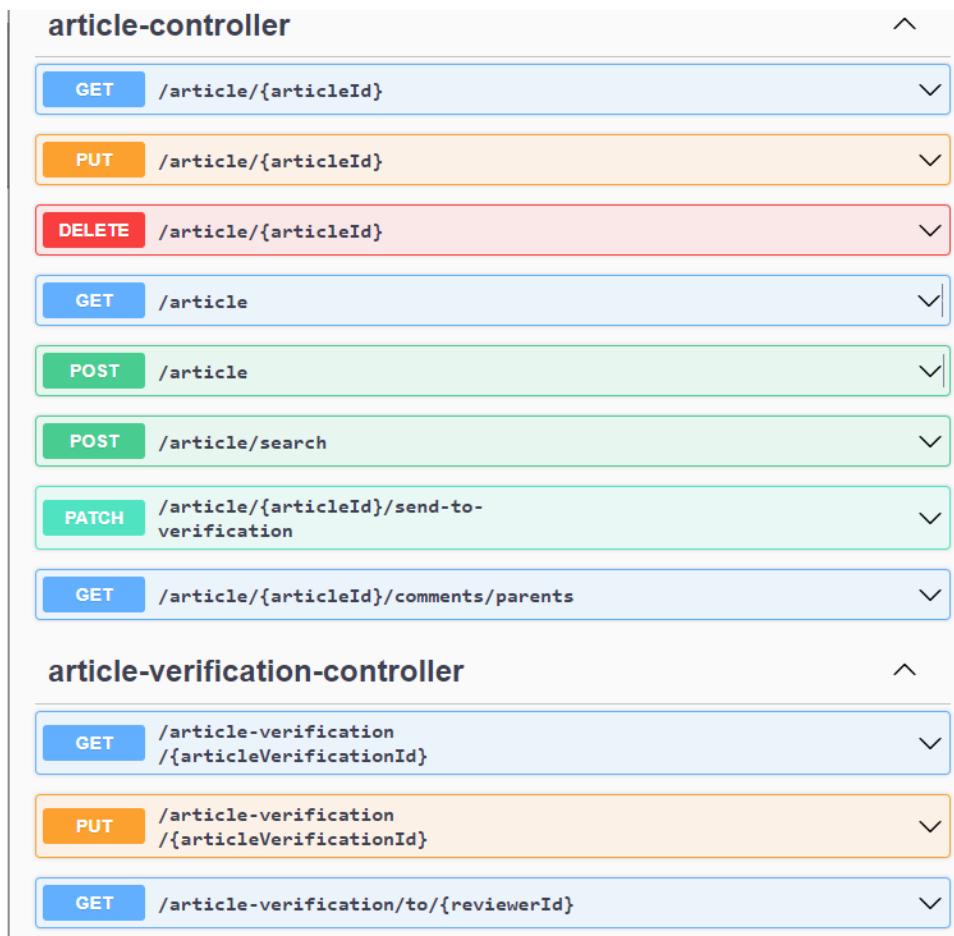
Dokumentacja wszystkich endpointów kontrolerów jest przedstawiona na rysunkach 3.4, 3.5 oraz 3.6.

The screenshot displays the Swagger UI interface for a RESTful API. It is organized into three main sections: **user-controller**, **opinion-controller**, and **comment-controller**.

- user-controller:** Contains four endpoints:
 - PUT /user/{userId}**
 - POST /user**
 - GET /user/user-account-id/{userAccountId}**
 - GET /user/nickname/{nickname}**
- opinion-controller:** Contains five endpoints:
 - PUT /opinion/{opinionId}**
 - DELETE /opinion/{opinionId}**
 - POST /opinion**
 - GET /opinions/article/{articleId}**
 - GET /opinion/author/{authorId}**
- comment-controller:** Contains four endpoints:
 - PUT /comment/{commentId}**
 - DELETE /comment/{commentId}**
 - POST /comment**
 - GET /comment/subs/{parentCommentId}**

Rys. 3.4: Endpointy kontrolerów dla użytkowników, opinii oraz komentarzy

Źródło: Opracowanie własne - zrzut ekranu z programu Swagger



Rys. 3.5: Endpointy kontrolerów dla artykułów oraz weryfikacji artykułów

Źródło: Opracowanie własne - zrzut ekranu z programu Swagger

user-inaccessibility-controller

- POST** /user-inaccessibility
- GET** /user-inaccessibility/user/{userId}
- DELETE** /user-inaccessibility/{userInaccessibilityId}

opinion-acceptation-controller

- POST** /acceptance
- DELETE** /acceptance

technology-controller

- GET** /technology
- GET** /technology/category/{technologyCategoryId}

technology-category-controller

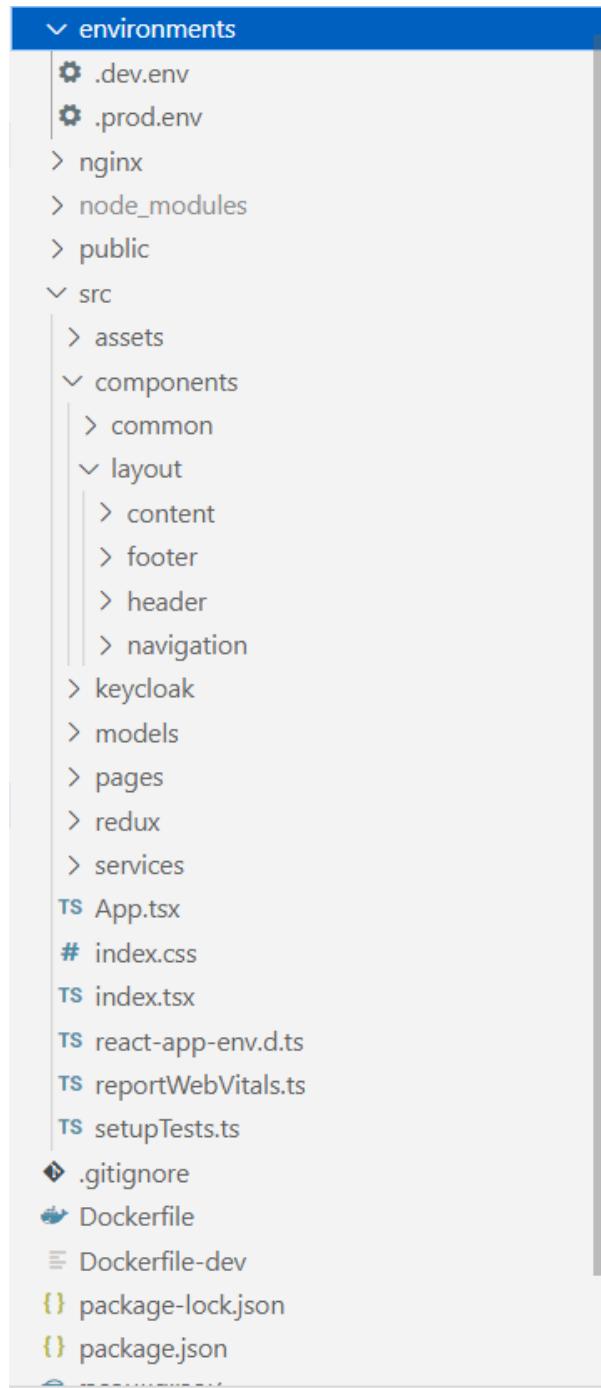
- GET** /technology-category/{technologyCategoryId}
- GET** /technology-category/tree

Rys. 3.6: Endpointy kontrolerów dla dostępności użytkowników, akceptacji opinii, technologii oraz kategorii technologii

Źródło: Opracowanie własne - zrzut ekranu z programu Swagger

3.3. Aplikacja kliencka

Aplikacji klienckiej ma mniej restrykcyjną architekturę niż backend, ale można wyróżnić wiele elementów odpowiedzialnych jedynie za swoją część funkcji. Zasada działania frontendu zostanie opisana na podstawie struktury projektu oraz przykładowych fragmentów kodu. Struktura aplikacji klienckiej jest przedstawiona na rysunku 3.7



Rys. 3.7: Struktura projektu aplikacji klienckiej

Źródło: opracowanie własne

Tak samo jak w przypadku backendu zastosowano różne profile konfiguracyjne aplikacji. Konfiguracja aplikacji (adresy backendu i serwera Keycloak) jest zapisana w osobnych plikach w folderze environments. Można wyróżnić 2 profile konfiguracyjne frontendu: lokalny oraz produkcyjny. Profil lokalny jest wykorzystywany w trakcie rozwijania aplikacji i w konfiguracji tej żądania są wysyłane do lokalnego backendu oraz lokalnego serwera Keycloak. W tej konfiguracji aplikacja kliencka jest uruchamiana na serwerze webowym dostarczonym przez create-react-app, który jest przeznaczony jedynie do rozwoju aplikacji. Profil produkcyjny jest wykorzystywany przez wdrożoną aplikację i w konfiguracji tej żądania są wysyłane do zdalnego backendu oraz zdalnego serwera Keycloak. W profilu tym nie jest zapewniony serwer we-

bowy, dlatego należy wdrożyć aplikację na odrębnym serwerze webowym. Przykładowy profil lokalny jest przedstawiony na listingu 3.13.

Listing 3.13: Konfiguracja profilu lokalnego dla aplikacji klienckiej

```
REACT_APP_API=http://localhost:9000
REACT_APP_KEYCLOAK_URL=http://localhost:8080
```

Odpowiednikiem pliku `pom.xml` używanego przez backend jest po stronie frontendu plik `package.json`, gdzie również są zapisane zależności oraz konfiguracja uruchomieniowa aplikacji. Fragment zawartości pliku `package.json` jest przedstawiony na listingu 3.14.

Listing 3.14: Fragment zawartości pliku `package.json` używanego przez frontend

```
1  {
2    "name": "my-app",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@emotion/react": "^11.10.4",
7      "@emotion/styled": "^11.10.4",
8      "@mui/icons-material": "^5.10.3",
9      "@mui/lab": "^5.0.0-alpha.102",
10     ...
11     "scripts": {
12       "start": "env-cmd -f environments/.dev.env react-scripts start",
13       "build": "env-cmd -f environments/.prod.env react-scripts build",
14       "test": "react-scripts test",
15       "eject": "react-scripts eject"
16     },
17     ...
18 }
```

Na listingu 3.14 ustalane są m.in. nazwa i wersja aplikacji, zależności projektu (`dependencies`) oraz komendy uruchomieniowe dla poszczególnych profili konfiguracyjnych (`scripts`).

Wygenerowane zależności na podstawie pliku `package.json` są przechowywane w folderze `node_modules`.

W folderze `nginx` oraz plikach `Dockerfile` zapisana jest konfiguracja wdrożeniowa aplikacji, która jest opisana w rozdziale o wdrożeniu aplikacji.

Zaimplementowana aplikacja jest typu SPA, co oznacza, że jest jedynie jedna strona HTML, której zawartość jest zmieniana dynamicznie. W folderze `public` zawarta jest m.in. ta jedna strona.

Folder `src` jest najważniejszy w tym projekcie, gdyż znajduje się w nim cały kod aplikacji i dlatego zawartość tego foldera zostanie opisana najdokładniej.

W folderze `assets` znajdują się mapy bitowe wykorzystywane przez projekt oraz regulamin serwisu.

W `components` są zamieszczone pojedyncze komponenty, które są powszechnie wykorzystywane w aplikacji (np. powiadomienia) lub stanowią układ strony (np. nagłówek, czy stopka strony). Zawartość przykładowego komponentu dostarczającego awatar użytkownika jest przedstawiona na listingu 3.15.

Listing 3.15: Kod komponentu awatar użytkownika

```
import { Avatar, SxProps, Theme } from "@mui/material";
import React, { useEffect } from "react";

const CustomAvatar = (props: { file: any, sx?: SxProps<Theme> }) => {
  const [img, setImg] = React.useState<any>('');
```

```

const loadImg = () => {

  if(!props.file)
    return

  setImg(props.file);
}

useEffect(() => {
  loadImg()
}, [props])

return (
  <Avatar
    alt="Avatar"
    src={'data:image/jpeg;base64,' + img}
    sx={{...props.sx}}
  />
);
}

export default CustomAvatar;

```

Wygląd komponentu awatar użytkownika jest aktualizowany za każdym razem, gdy zmieni się otrzymywany przez niego parametr `file`.

W folderze `models` znajdują się interfejsy obiektów używanych przez aplikacje. Przykładowy model artykułu jest przedstawiony na listingu 3.16.

Listing 3.16: Model artykułu w aplikacji klienckiej

```

import { ArticleStatus } from "./ArticleStatus"
import { Technology } from "./Technology"
import User from "./User"

export default interface Article {
  id: string,
  authorDTO: User,
  technologyDTO: Technology,
  title: string,
  content: string,
  status: ArticleStatus,
  creationDate: Date,
  modificationDate: Date,
  averageRating?: number
}

```

W interfejsach można m.in. zagnieździć obiekty powstałe na podstawie innych interfejsów.

W `Pages` znajdują się główne komponenty aplikacji, które pełnią rolę osobnych podstron.

W folderze `services` są zamieszczone serwisy, które są klasami udostępniającymi metody odpowiedzialne za wysyłanie żądań do backendu. Metody te zwracają odpowiedzi otrzymane od backendu, które będą mogły być później obsłużone adekwatnie od otrzymanego statusu. Fragment przykładowego serwisu artykułów jest przedstawiony na listingu 3.17.

Listing 3.17: Fragment serwisu artykułów

```

...
class ArticleAPIService {

  private apiUrl: string = `${process.env.REACT_APP_API as string}/
    ↳ article‘

```

```

search = (searchCriteria: ArticleSearchCriteria, pagination: Pagination
    ↵ , role: string, loggedUserId?: number) => {

    let params: any = {
        ...pagination,
        role: role,
    }

    if(loggedUserId){
        params.loggedUserId = loggedUserId
    }

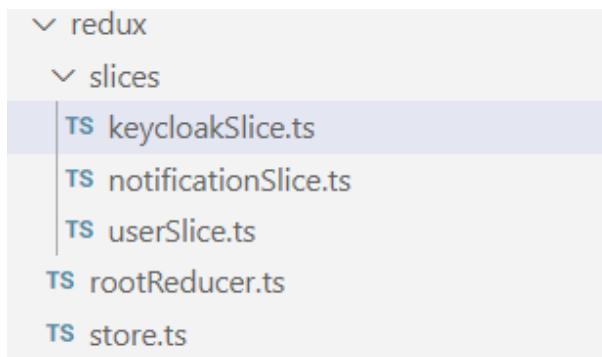
    return axios.post(`${this.apiUrl}/search`, searchCriteria, {
        params: params
    })
}

...
}

export default new ArticleAPIService();

```

W folderze `redux` znajduje się kod odpowiedzialny za przechowywanie i zarządzanie globalnym stanem aplikacji. Zawartość tego folderu jest przedstawiona na rysunku 3.8.



Rys. 3.8: Struktura folderu `redux` aplikacji klienckiej

Źródło: opracowanie własne

W `slice` definiowane są pojedyncze i zarządzalne magazyny danych, w `rootReducer` grupowane są `slice` oraz w `store` jest tworzony i konfigurowany jeden globalny magazyn stanu aplikacji. Fragment przykładowego kodu `slice` przechowującego dane o zalogowanym użytkowniku jest przedstawiony na listingu 3.18.

Listing 3.18: Fragment kodu slice zalogowanego użytkownika

```

...
export interface UserState {
    user: any,
}

const initialState: UserState = {
    user: {},
}

export const userSlice = createSlice({
    name: 'users',
    initialState,

```

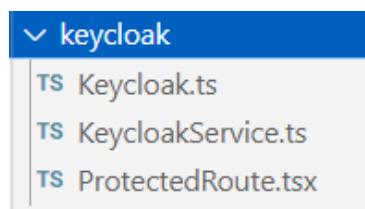
```

    reducers: {
      setUser(state, action: PayloadAction<User | {}>){
        state.user = action.payload
      }
    }
  ...
}

```

Na listingu 3.18 zdefiniowane są dane użytkownika (`initialState`) oraz metoda `setUser`, której wywołanie aktualizuje te dane.

Frontend jest zgodnie z początkowymi założeniami zabezpieczony poprzez zewnętrzny serwer do uwierzytelniania i autoryzacji o nazwie Keycloak. W folderze `keycloak` znajduje się kod odpowiedzialny za komunikację oraz bezpieczeństwo zapewniane przez serwer Keycloak. Zawartość tego folderu jest przedstawiona na rysunku 3.9



Rys. 3.9: Kod związany z serwerem Keycloak po stronie frontendu

Źródło: opracowanie własne

W pliku `Keycloak.ts` zdefiniowana jest komunikacja z serwerem Keycloak (3.19).

Listing 3.19: Konfiguracja komunikacji z serwerem Keycloak

```

const keycloak = {
  url: process.env.REACT_APP_KEYCLOAK_URL ,
  realm: "Programming_technologies",
  clientId: "frontend"
};

```

W pliku `KeycloakService.ts` znajduje się serwis, który odpowiada za wysyłanie żądań do serwera Keycloak oraz udostępnia metody pozwalające na m.in. zdekodowanie otrzymanego tokenu dostępu, czy ustawienie nagłówka autoryzacji. Na listingu 3.20 są przedstawione najważniejsze metody tego serwisu.

Listing 3.20: Fragment kodu serwisu odpowiadającego za komunikację z serwerem Keycloak

```

...
export const roles = {
  user: "user",
  logged_user: {
    id: "cec99090-10b9-4fc9-880c-9f72dca702eb",
    name: "logged_user"
  },
  administrator: {
    id: "dd8865ea-a3ee-4569-9298-5ea705f5c475",
    name: "administrator"
  },
  reviewer: {
    id: "c59b2c5e-c209-4e66-8c07-d4a5a0625552",
    name: "reviewer"
  }
}
class KeycloakService {

```

```

login(credentials: Credentials){

    const body = new URLSearchParams({
        client_id: keycloak.clientId as string,
        username: credentials.username,
        password: credentials.password,
        grant_type: 'password'
    });

    const header = {
        headers: {
            "Content-Type": "application/x-www-form-urlencoded"
        }
    }

    return axios.post(`${keycloak.url}/realms/${keycloak.realm}/
        ↪ protocol/openid-connect/token`, body, header)
}

getAccessTokenOnRefreshToken = (refreshToken: string) => {

    const body = new URLSearchParams({
        client_id: keycloak.clientId as string,
        grant_type: 'refresh_token',
        refresh_token: refreshToken
    });

    const header = {
        headers: {
            "Content-Type": "application/x-www-form-urlencoded"
        }
    }

    return axios.post(`${keycloak.url}/realms/${keycloak.realm}/
        ↪ protocol/openid-connect/token`, body, header)
}

logout = (refreshToken: string) => {

    const body = new URLSearchParams({
        client_id: keycloak.clientId as string,
        refresh_token: refreshToken
    });

    const header = {
        headers: {
            ...axios.defaults.headers.common,
            "Content-Type": "application/x-www-form-urlencoded"
        }
    }

    return axios.post(`${keycloak.url}/realms/${keycloak.realm}/
        ↪ protocol/openid-connect/logout`, body, header)
}

...
}
...

```

Na listingu 3.20 przedstawiono definicję ról i metody odpowiedzialne za logowanie, wygenerowanie nowego tokenu dostępu oraz wylogowanie. Przy logowaniu podawane są przede wszystkim login i hasło, a w odpowiedzi można otrzymać status 401 w przypadku podania nieprawidłowych danych logowania albo wygenerowane tokeny dostępu i odświeżania. W przypadku pomyślnego zalogowania się użytkownika, otrzymane role, tokeny dostępu oraz odświeżania są zapisywane w globalnym stanie aplikacji i dzięki temu dostęp do tych danych jest możliwy z poziomu każdego komponentu. Aby wygenerować nowy token dostępu, należy dodać do wiadomości m.in. aktualny token dostępu. Do wylogowania się jest przede wszystkim potrzebny aktualny token odświeżania.

W pliku `ProtectedRoute.ts` zdefiniowany jest komponent, który odpowiada za sprawdzenie i obsłużenie tego, czy dany użytkownik ma dostęp do żądanego przez niego komponentu. Fragment kodu komponentu `ProtectedRoute` jest przedstawiony na listingu 3.21.

Listing 3.21: Komponent `ProtectedRoute`

```
const PrivateRoute = (props: PrivateRouteProps) => {
  const keycloak = useSelector((state: RootState) => state.keycloak);

  if(!props.requiredLogin){
    return <React.Fragment>{props.children}</React.Fragment>
  }

  if(!props.requiredRole){
    const isLoggedIn = keycloak.authenticated
    return isLoggedIn ? <React.Fragment>{props.children}</React.
      ↪ Fragment> : <Forbidden/>
  }

  return keycloak.roles.includes(props.requiredRole) ? <React.Fragment>{
    ↪ props.children}</React.Fragment> : <Forbidden/>
}
```

W przypadku kiedy użytkownik będzie żądał dostępu do komponentu, do którego nie jest uprawniony (brak odpowiednich ról), to będzie on przekierowywany do strony z napisem „Brak dostępu do zasobu”. Użytkownikowi posiadającemu odpowiednie uprawnienia wyświetlony zostanie żądany przez niego komponent. Komponent `ProtectedRoute` jest używany w głównej nawigacji po stronach (listing 3.22).

Listing 3.22: Zabezpieczona nawigacja

```
<Route path="/manage-users" element={
  <ProtectedRoute requiredLogin={true} requiredRole={roles.administrator .
    ↪ name}>
    <Outlet />
  </ProtectedRoute>
}>
  <Route index element={<SearchUsers/>} />
  <Route path="user/:userId/:userAccountId" element={<ManageUser/>} />
</Route>
<Route path="/set-password" element={
  <ProtectedRoute requiredLogin={true}>
    <SetPassword/>
  </ProtectedRoute>
} />
```

Na listingu 3.22 przedstawiono przykładowe przekierowania do komponentów na podstawie ścieżki url. Założono, że dostęp do podstrony z zarządzaniem użytkownikami powinien mieć jedynie administrator oraz wylogować się może jedynie zalogowany użytkownik. Przekierowania

te można również zagnieździć i przykładowo dzięki temu wystarczyło jedno użycie komponentu **ProtectedRoute** do zabezpieczenia 2 komponentów **SearchUsers** oraz **ManageUser**.

Rozdział 4

Testowanie systemu IT Tech

Kolejnym etapem realizacji pracy jest wykonanie testów zaimplementowanego systemu. Prze prowadzone zostały testy integracyjne backendu.

Założeniem tych testów było dokładne przetestowanie jak największej liczby endpointów wszystkich kontrolerów. Testy te polegały na wysyłaniu odpowiednio spreparowanych żądań na poszczególne endpointy kontrolerów i następnie sprawdzaniu otrzymanych odpowiedzi. Prze prowadzono zarówno testy sprawdzające, czy dane funkcjonalności działają, jak i sprawdzające, czy zostanie zwrócony odpowiedni komunikat błędu w przypadku podania niepoprawnych danych. Do testów integracyjnych backendu wykorzystano przede wszystkim narzędzia REST-assured oraz JUnit. Testy były przeprowadzane w profilu testowym, który charakteryzuje się tym, że w tym trybie używana jest baza danych h2. Baza danych h2 jest zapisywana w pamięci i dzięki temu możliwe było przetestowanie funkcji backendu w środowisku bardzo podobnym do produkcyjnego. Inną zaletą stosowania bazy danych podczas testów jest to, że można w łatwy sposób korzystać z danych testowych poprzez zapisanie ich do bazy danych. W pierwszej kolejności przetestowano najprostsze i najważniejsze funkcje systemu.

4.1. Testy kontrolera użytkowników

4.1.1. Test sprawdzający, czy można pobrać dane użytkownika na podstawie jego numeru konta

Listing 4.1: Test integracyjny pobierania danych użytkownika po jego id konta

```
@Test
public void should GetUserByUserAccountId() {

    UserEntity user = users.get(1);

    Response response = given()
        .when()
        .get(url + "user-account-id/" + user.
            ↪ getUserAccountId());

    response.then().statusCode(HttpStatus.OK.value());
    UserDTO userDTO = response.as(UserDTO.class);
    UserEntity userEntity = userMapper.userDTOtoUserEntity(userDTO);

    assertEquals(userEntity.getUserAccountId(), user.getUserAccountId()
        ↪ );
}
```

4.1.2. Test sprawdzający, czy nie można pobrać danych użytkownika na podstawie nieistniejącego numeru konta

Listing 4.2: Test integracyjny pobierania danych użytkownika po nieistniejącym id konta

```
@Test
public void shouldNot GetUserByNotExistingUserId() {

    Response response = given()
        .when()
        .get(url + "user-account-id/-1");

    response.then().statusCode(HttpStatus.NOT_FOUND.value());
}
```

4.1.3. Test sprawdzający, czy zostanie zwrócona prawda w przypadku istnienia użytkownika o podanym pseudonimie

Listing 4.3: Test integracyjny zwracania prawdy w przypadku istnienia użytkownika o podanym pseudonimie

```
@Test
public void shouldReturnTrueWhenUserExistsByNickname() {

    Response response = given()
        .when()
        .get(url + "nickname/" + users.get(1).getNickname()
            );

    response.then().statusCode(HttpStatus.OK.value());

    boolean result = response.as(boolean.class);
    assertTrue(result);
}
```

4.1.4. Test sprawdzający, czy zostanie zwrócony fałsz w przypadku nieistnienia użytkownika o podanym pseudonimie

Listing 4.4: Test integracyjny zwracania fałszu w przypadku nie istnienia użytkownika o podanym pseudonimie

```
@Test
public void shouldReturnFalseWhenUserDoesntExistByNickname() {

    Response response = given()
        .when()
        .get(url + "nickname/" + "a");

    response.then().statusCode(HttpStatus.OK.value());

    boolean result = response.as(boolean.class);
    assertFalse(result);
}
```

4.1.5. Test sprawdzający, czy można utworzyć użytkownika

Listing 4.5: Test integracyjny tworzenia użytkownika

```

@Test
public void shouldCreateUser() {

    UserEntity userEntity = UserEntity.builder()
        .userAccountId("10")
        .nickname("10")
        .firstname("10")
        .surname("10")
        .email("10@mail.pl")
        .build();

    UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .body(userDTO)
        .post(url);

    response.then().statusCode(HttpStatus.CREATED.value());
    UserDTO createdUserDTO = response.as(UserDTO.class);
    UserEntity createdUserEntity = userMapper.userDTOToUserEntity(
        ↪ createdUserDTO);

    assertEquals(createdUserEntity.getUserAccountId(), userEntity.
        ↪ getUserAccountId());
    userRepository.deleteById(createdUserEntity.getId());
}

```

4.1.6. Test sprawdzający, czy nie można utworzyć użytkownika z podanym istniejącym już numerem konta

Listing 4.6: Test integracyjny tworzenia użytkownika z już istniejącym numerem konta

```

@Test
public void shouldNotCreateUserWithConflictUserAccountId() {

    UserEntity userEntity = UserEntity.builder()
        .userAccountId("1")
        .nickname("10")
        .firstname("10")
        .surname("10")
        .email("10@mail.pl")
        .build();

    UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .body(userDTO)
        .post(url);
}

```

```

        response.then().statusCode(HttpStatus.CONFLICT.value());
    }
}

```

4.1.7. Test sprawdzający, czy nie można utworzyć użytkownika z podanym istniejącym już pseudonimem

Listing 4.7: Test integracyjny tworzenia użytkownika z już istniejącym pseudonimem

```

@Test
public void shouldNotCreateUserWithConflictNickname() {

    UserEntity userEntity = UserEntity.builder()
        .userAccountId("10")
        .nickname("1")
        .firstname("10")
        .surname("10")
        .email("10@mail.pl")
        .build();

    UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .body(userDTO)
        .post(url);

    response.then().statusCode(HttpStatus.CONFLICT.value());
}

```

4.1.8. Test sprawdzający, czy nie można utworzyć użytkownika z pustymi danymi

Listing 4.8: Test integracyjny tworzenia użytkownika z pustymi danymi

```

@Test
public void shouldNotCreateUserWithEmptyData() {

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .post(url);

    response.then().statusCode(HttpStatus.BAD_REQUEST.value());
}

```

4.1.9. Test sprawdzający, czy nie można utworzyć użytkownika z pustym id konta

Listing 4.9: Test integracyjny tworzenia użytkownika z pustym id konta

```

@Test
public void shouldNotCreateUserWithNotGivenUserAccountId() {
}

```

```

        UserEntity userEntity = UserEntity.builder()
            .nickname("10")
            .firstname("10")
            .surname("10")
            .email("10@mail.pl")
            .build();

        UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

        Response response = given()
            .when()
            .body(userDTO)
            .contentType(MediaType.APPLICATION_JSON.toString
                ↪ ())
            .post(url);

        response.then().statusCode(HttpStatus.BAD_REQUEST.value());
    }
}

```

4.1.10. Test sprawdzający, czy nie można utworzyć użytkownika z pustym pseudonimem

Listing 4.10: Test integracyjny tworzenia użytkownika z pustym pseudonimem

```

@Test
public void shouldNotCreateUserWithNotGivenNickname() {

    UserEntity userEntity = UserEntity.builder()
        .userAccountId("10")
        .firstname("10")
        .surname("10")
        .email("10@mail.pl")
        .build();

    UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .body(userDTO)
        .post(url);

    response.then().statusCode(HttpStatus.BAD_REQUEST.value());
}

```

4.1.11. Test sprawdzający, czy można zaktualizować dane użytkownika o podanym id

Listing 4.11: Test integracyjny aktualizowania danych użytkownika po id

```

@Test
public void shouldUpdateUser() {

    UserEntity userEntity = UserEntity.builder()
        .nickname("11")

```

```

        .firstname("10")
        .surname("10")
        .email("10@mail.pl")
        .build();

UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

Response response = given()
    .when()
    .contentType(MediaType.APPLICATION_JSON.toString
        ↪ ())
    .body(userDTO)
    .put(url + users.get(0).getId());

response.then().statusCode(HttpStatus.OK.value());
UserDTO updatedUserDTO = response.as(UserDTO.class);
UserEntity updatedUserEntity = userMapper.userDTOToUserEntity(
    ↪ updatedUserDTO);

assertEquals(updatedUserEntity.getId(), users.get(0).getId());
assertEquals(updatedUserEntity.getNickname(), userEntity.
    ↪ getNickname());
assertEquals(updatedUserEntity.getFirstname(), userEntity.
    ↪ getFirstname());
assertEquals(updatedUserEntity.getSurname(), userEntity.
    ↪ getFirstname());
assertEquals(updatedUserEntity.getEmail(), userEntity.getEmail());
}

```

4.1.12. Test sprawdzający, czy nie można zaktualizować danych użytkownika o podanym niepoprawnym id

Listing 4.12: Test integracyjny aktualizowania danych użytkownika po niepoprawnym id

```

@Test
public void shouldNotUpdateUserWithInvalidId(){

    UserEntity userEntity = UserEntity.builder()
        .nickname("10")
        .firstname("10")
        .surname("10")
        .email("10@mail.pl")
        .build();

    UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .body(userDTO)
        .put(url + "a");

    response.then().statusCode(HttpStatus.BAD_REQUEST.value());
}

```

4.1.13. Test sprawdzający, czy nie można zaktualizować danych użytkownika z podanymi pustymi danymi

Listing 4.13: Test integracyjny aktualizowania danych użytkownika z podanymi pustymi danymi

```
@Test
public void shouldNotUpdateUserWithEmptyData() {

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .put(url + users.get(0).getId());

    response.then().statusCode(HttpStatus.BAD_REQUEST.value());
}
```

4.1.14. Test sprawdzający, czy nie można zaktualizować danych użytkownika z podanym nieistniejącym id

Listing 4.14: Test integracyjny aktualizowania danych użytkownika z podanym nieistniejącym id

```
@Test
public void shouldNotUpdateUserWhenIdDoesntExist() {

    UserEntity userEntity = UserEntity.builder()
        .nickname("10")
        .firstname("10")
        .surname("10")
        .email("10@mail.pl")
        .build();

    UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
            ↪ ())
        .body(userDTO)
        .put(url + "11");

    response.then().statusCode(HttpStatus.NOT_FOUND.value());
}
```

4.1.15. Test sprawdzający, czy nie można zaktualizować danych użytkownika z podanym już istniejącym pseudonimem

Listing 4.15: Test integracyjny aktualizowania danych użytkownika z podanym już istniejącym pseudonimem

```
@Test
public void shouldNotUpdateUserWithConflictNickname() {

    UserEntity userEntity = UserEntity.builder()
        .nickname("1")
        .firstname("10")
```

```

        .surname("10")
        .email("10@mail.pl")
        .build();

UserDTO userDTO = userMapper.userEntityToUserDTO(userEntity);

Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString
        ↵ ())
        .body(userDTO)
        .put(url + users.get(0).getId());

response.then().statusCode(HttpStatus.CONFLICT.value());
}

```

4.1.16. Wyniki testów

Wszystkie testy zakończyły się pomyślnie (rysunek 4.1).

Test Results		2 sec 154 ms
UserControllerTests		2 sec 154 ms
✓	shouldUpdateUser()	1 sec 766 ms
✓	shouldNotCreateUserWithConflictNickname()	41 ms
✓	shouldNotCreateUserWithConflictUserAccount()	22 ms
✓	shouldNotCreateUserWithNotGivenNickname()	17 ms
✓	shouldReturnFalseWhenUserDoesntExistByNickname()	26 ms
✓	shouldNotUpdateUserWithInvalidId()	20 ms
✓	shouldNotCreateUserWithEmptyData()	43 ms
✓	shouldCreateUser()	43 ms
✓	shouldNotUpdateUserWithEmptyData()	27 ms
✓	shouldNotCreateUserWithNotGivenUserAccount()	26 ms
✓	shouldNotUpdateUserWhenIdDoesntExist()	31 ms
✓	should GetUserByUserAccountId()	32 ms
✓	shouldNot GetUserByNotExistingUserAccountId()	18 ms
✓	shouldNotUpdateUserWithConflictNickname()	26 ms
✓	shouldReturnTrueWhenUserExistsByNickname()	16 ms

Rys. 4.1: Wyniki testów integracyjnych backendu dla kontrolera użytkowników

Źródło: opracowanie własne

4.2. Testy kontrolera kategorii technologii

4.2.1. Test sprawdzający, czy można pobrać poprawne drzewo kategorii technologii

Listing 4.16: Test integracyjny pobierania drzewa kategorii technologii

```

@Test
public void shouldGetTreeOfTechnologyCategories(){

    Response response = given()
        .when()
        .get(url + "tree");
}

```

```

response.then().statusCode(HttpStatus.OK.value());

List<ComplexTechnologyCategoryDTO>
    ↪ foundComplexTechnologyCategoryDTOs = Arrays.stream(
        response.as(ComplexTechnologyCategoryDTO[].class)
    ).toList();
List<TechnologyCategoryEntity> foundTechnologyCategories =
    technologyCategoryMapper.
        ↪ complexTechnologyCategoryDTOListToTechnologyCategoryEntityList
        ↪ (
            foundComplexTechnologyCategoryDTOs
        );

assertEquals(foundTechnologyCategories.get(0).getId(),
    ↪ technologyCategories.get(0).getId());
assertEquals(foundTechnologyCategories.get(1).getId(),
    ↪ technologyCategories.get(1).getId());
assertEquals(foundTechnologyCategories.get(2).getId(),
    ↪ technologyCategories.get(2).getId());

List<TechnologyCategoryEntity> childrenTechnologyCategories1 =
    foundTechnologyCategories.get(0).
        ↪ getChildrenTechnologyCategoryEntityList();

assertEquals(childrenTechnologyCategories1.get(0).getId(),
    ↪ technologyCategories.get(3).getId());

List<TechnologyCategoryEntity> childrenTechnologyCategories2 =
    foundTechnologyCategories.get(1).
        ↪ getChildrenTechnologyCategoryEntityList();

assertEquals(childrenTechnologyCategories2.get(0).getId(),
    ↪ technologyCategories.get(4).getId());
assertEquals(childrenTechnologyCategories2.get(1).getId(),
    ↪ technologyCategories.get(5).getId());

List<TechnologyCategoryEntity> childrenTechnologyCategories3 =
    foundTechnologyCategories.get(2).
        ↪ getChildrenTechnologyCategoryEntityList();

assertEquals(childrenTechnologyCategories3.get(0).getId(),
    ↪ technologyCategories.get(6).getId());

List<TechnologyCategoryEntity> childrenTechnologyCategories11 =
    childrenTechnologyCategories1.get(0).
        ↪ getChildrenTechnologyCategoryEntityList();

assertEquals(childrenTechnologyCategories11.get(0).getId(),
    ↪ technologyCategories.get(7).getId());
assertEquals(childrenTechnologyCategories11.get(1).getId(),
    ↪ technologyCategories.get(8).getId());
assertEquals(childrenTechnologyCategories11.get(2).getId(),
    ↪ technologyCategories.get(9).getId());

List<TechnologyCategoryEntity> childrenTechnologyCategories21 =
    childrenTechnologyCategories2.get(0).
        ↪ getChildrenTechnologyCategoryEntityList();

assertEquals(childrenTechnologyCategories21.get(0).getId(),
    ↪ technologyCategories.get(10).getId());

```

```

        assertEquals(childrenTechnologyCategories21.get(1).getId(),
                     ↪ technologyCategories.get(11).getId());
    }
}

```

4.2.2. Test sprawdzający, czy można pobrać dane kategorii technologii o podanym id

Listing 4.17: Test integracyjny pobierania kategorii technologii o podanym id

```

@Test
public void shouldGetTechnologyCategoryById() {

    Response response = given()
        .when()
        .get(url + "2");

    response.then().statusCode(HttpStatus.OK.value());
    TechnologyCategoryDTO technologyCategoryDTO = response.as(
        ↪ TechnologyCategoryDTO.class);

    assertEquals(technologyCategoryDTO.getId(), technologyCategories.
        ↪ get(1).getId());
}

```

4.2.3. Test sprawdzający, czy nie można pobrać danych kategorii technologii o podanym niepoprawnym id

Listing 4.18: Test integracyjny pobierania kategorii technologii o podanym id

```

@Test
public void shouldNotGetTechnologyCategoryByInvalidId() {

    Response response = given()
        .when()
        .get(url + "a");

    response.then().statusCode(HttpStatus.BAD_REQUEST.value());
}

```

4.2.4. Test sprawdzający, czy nie można pobrać danych kategorii technologii o podanym nieistniejącym id

Listing 4.19: Test integracyjny pobierania kategorii technologii o podanym nieistniejącym id

```

@Test
public void shouldNotGetTechnologyCategoryByNonExistingId() {

    Response response = given()
        .when()
        .get(url + "13");

    response.then().statusCode(HttpStatus.NOT_FOUND.value());
}

```

4.2.5. Wyniki testów

Wszystkie testy zakończyły się pomyślnie (rysunek 4.2).

Test Results		1 sec 811 ms
TechnologyCategoryControllerTests		1 sec 811 ms
✓	shouldGetTechnologyCategoryById()	1 sec 629 ms
✓	shouldGetTreeOfTechnologyCategories()	150 ms
✓	shouldNotGetTechnologyCategoryByNotExistin	16 ms
✓	shouldNotGetTechnologyCategoryByInvalidId()	16 ms

Rys. 4.2: Wyniki testów integracyjnych backendu dla kontrolera kategorii technologii

Źródło: opracowanie własne

4.3. Testy kontrolera technologii

4.3.1. Test sprawdzający, czy można pobrać dane wszystkich technologii

Listing 4.20: Test integracyjny pobierania wszystkich technologii

```
@Test
public void shouldGetAll(){

    List<Integer> technologiesIds = technologies.stream()
        .map(technologyEntity -> technologyEntity.getId())
        .collect(Collectors.toList());

    Response response = given()
        .when()
        .get(url);

    response.then().statusCode(HttpStatus.OK.value());
    TechnologyDTO[] gotTechnologies = response.as(TechnologyDTO[].class
        ↴ );

    for(int i=0; i < gotTechnologies.length; i++){
        assertTrue(technologiesIds.contains(gotTechnologies[i].
            ↴ getId()));
    }
}
```

4.3.2. Test sprawdzający, czy można pobrać listę technologii o podanym ich id kategorii

Listing 4.21: Test integracyjny pobierania listy technologii o podanym ich kategorii

```
@Test
public void shouldGetAllByTechnologyCategoryId(){

    Integer technologyCategoryId = 1;
    long numberofTechnologiesWithTechnologyCategory = technologies.
        ↴ stream()
        .filter(technologyEntity -> technologyEntity.
            ↴ getTechnologyCategoryEntity().getId() ==
            ↴ technologyCategoryId)
```

```

        .count();

Response response = given()
    .when()
    .get(url + "category/" + technologyCategoryId);

response.then().statusCode(HttpStatus.OK.value());
TechnologyDTO[] gotTechnologies = response.as(TechnologyDTO[].class
    ↵ );

assertEquals(gotTechnologies.length,
    ↵   numberofTechnologiesWithTechnologyCategory);
}

```

4.3.3. Test sprawdzający, czy nie można pobrać listy technologii o podanym niepoprawnym id kategorii

Listing 4.22: Test integracyjny pobierania listy technologii o podanym niepoprawnym id kategorii

```

@Test
public void shouldNotGetAllByInvalidTechnologyCategoryId(){

    Response response = given()
        .when()
        .get(url + "category/a");

    response.then().statusCode(HttpStatus.BAD_REQUEST.value());
}

```

4.3.4. Test sprawdzający, czy nie można pobrać listy technologii o podanym nieistniejącym id kategorii

Listing 4.23: Test integracyjny pobierania listy technologii o podanym nieistniejącym id kategorii

```

@Test
public void shouldNotGetAllByNotExistingTechnologyCategoryId(){

    Response response = given()
        .when()
        .get(url + "category/20");

    response.then().statusCode(HttpStatus.NOT_FOUND.value());
}

```

Wszystkie testy zakończyły się pomyślnie (rysunek 4.3).

Test Results		1 sec 912 ms
▼	✓ TechnologyControllerTests	1 sec 912 ms
▼	✓ shouldGetAll()	1 sec 788 ms
✓	shouldNotGetAllByInvalidTechnologyCategoryId()	35 ms
✓	shouldNotGetAllByNotExistingTechnologyCategoryId()	40 ms
✓	shouldGetAllByTechnologyCategoryId()	49 ms

Rys. 4.3: Wyniki testów integracyjnych backendu dla kontrolera technologii

Źródło: opracowanie własne

Rozdział 5

Wdrożenie systemu IT Tech

Ostatnim etapem prac tego projektu jest wdrożenie całego systemu. Przy wyborze platformy do wdrożenia systemu kierowano się przede wszystkim jak najniższą ceną usług, popularnością, stopniem wsparcia dla użytych w ramach tego projektu technologii, łatwością wdrożenia oraz własnym doświadczeniem. Początkowo zakładano wdrożenie całego systemu na Google Cloud Platform, gdyż jest to popularna platforma, możliwe jest darmowe wdrożenie systemu dzięki zarejestrowaniu się przez konto studenckie oraz mialem doświadczenie z tą platformą w ramach realizacji projektu zespołowego. Z powodu problemów z konfiguracją aplikacji, porzuceno ten pomysł i postanowiono wdrożyć system na innej platformie. Kolejnym wyborem była platforma Azure, która jest popularnym serwisem wdrożeniowym firmy Microsoft. Wybrano tę platformę, gdyż podobnie jak poprzednio możliwe jest darmowe wdrożenie systemu dzięki koncie, do którego przypisany jest e-mail studencki, zawiera prosty i intuicyjny interfejs, można w łatwy sposób wdrożyć aplikacje, jest popularniejszym rozwiązaniem niż Google Cloud Platform oraz mialem już doświadczenie z tą platformą. Do wdrożenia bazy danych PostgreSQL, aplikacji serwerowej, aplikacji klienckiej postawionej na serwerze NGINX oraz serwera do uwierzytelniania i autoryzacji użytkowników użyto platformy Azure. Azure nie posiada bogatego wsparcia dla bazy danych MongoDB, dlatego zdecydowano się na użycie innej platformy do wdrożenia tej bazy. Wybrano platformę MongoDB Cloud, gdyż jest to jeden z popularniejszych serwisów do wdrażania baz MongoDB oraz możliwe jest darmowe wdrożenie bazy danych.

5.1. Baza danych PostgreSQL

Do wdrożenia bazy danych PostgreSQL wybrano usługę **Azure Database for PostgreSQL**, która zapewnia w pełni zarządzaną oraz elastyczną bazę danych PostgreSQL. Podczas konfiguracji bazy danych wybrano przede wszystkim domyślne opcje.

Na początku przypisano usługę do subskrypcji studenckiej oraz grupy zasobów **inzynierka**, podano nazwę serwera bazy, lokalizację bazy danych, używaną wersję PostgreSQL oraz typ obciążenia. Konfiguracja ta jest przedstawiona na rysunku 5.1.

Serwer elastyczny

Podstawowe Sieć Zabezpieczenia (wersja zapoznawcza) Tagi Przeglądanie + tworzenie

Utwórz usługę Azure Database dla serwera elastycznego PostgreSQL. [Dowiedz się więcej](#)

Czy wiesz, że nowi użytkownicy platformy Azure mogą korzystać z PostgreSQL — serwera elastycznego bezpłatne do 750 godzin przy użyciu bezpłatnego konta platformy Azure? [Dowiedz się więcej](#)

Szczegóły projektu

Wybierz subskrypcję, aby zarządzać wdrożonymi zasobami i kosztami. Użyj grup zasobów jak folderów, aby organizować wszystkie Twoje zasoby i zarządzać nimi.

Subskrypcja * [Azure for Students](#)

Grupa zasobów * [inzynierka](#) [Utwórz nowy](#)

Szczegóły serwera

Wprowadź wymagane ustawienia dla tego serwera, w tym wybierania lokalizacji oraz konfigurowania zasobów obliczeniowych i magazynowych.

Nazwa serwera * [inzynierka-test](#)

Region * [North Europe](#)

Wersja PostgreSQL * [14](#)

Typ obciążenia **Produkcja (mały/średni rozmiar)** **Produkcja (duży rozmiar)** **Przetestowanie**

Szacowane koszty

Opis	Wartość
Obliczanie jednostki SKU	USD 144.25/miesiąc
Standard_D2ds_v4 (rdzenie wirtualne2, USD72.12 na rdzeń wirtualny)	2 x 72.12
Magazyn	USD 16.26/miesiąc
Wybrano magazyn 128 GiB (USD0.13 na GiB)	128 x 0.13
Przepustowość	
W przypadku wychodzącego transferu danych między usługami w różnych regionach będą naliczane dodatkowe opłaty. Każdy przychodzący transfer danych jest bezpłatny. Dowiedz się więcej	
Szacowana suma	USD 160.51/miesiąc
Ceny odzwierciedlają tylko oszacowania. Wyświetl kalkulator cen platformy Azure. Koronne opłaty będą wyświetlane w walucie lokalnej w widokach analizy kosztów i rozliczeń.	

[Przeglądanie + tworzenie](#) Następny: Sieć >

Rys. 5.1: Podstawowa konfiguracja wdrożeniowa bazy danych PostgreSQL

Źródło: <https://portal.azure.com/#create/Microsoft.PostgreSQLServer>

Następnie podano login oraz hasło administratora, a resztę opcji podstawowych pozostawiono z wartościami domyślnymi. Konfiguracja ta jest przedstawiona na rysunku 5.2.

Microsoft Azure

Strona główna > Usługi bezpłatne > Wybierz opcję wdrożenia usługi Azure Database for PostgreSQL >

Serwer elastyczny

Microsoft

Obliczenia i magazyn

Przeznaczenie ogólne, D2ds_v4
Rdzenie wirtualne: 2, 8 GiB pamięci RAM, magazyn: 128 GiB
Nadmiarowość obszaru geograficznego: Enabled
[Konfigurowanie serwera](#)

Strefa dostępności

Brak preferencji

Duża dostępność
Strefowa nadmiarowa wysoka dostępność powoduje wdrożenie rezerwowej repliki w innej strefie z włączoną funkcją automatycznego przełączania w tryb failover. Opcje wysokiej dostępności możesz również określić w bloku „Obliczenia i magazyn”.

Włącz wysoką dostępność

Uwierzytelnianie
Wybierz metody uwierzytelniania, które mają być obsługiwane na potrzeby uzyskiwania dostępu do tego serwera PostgreSQL. Uwierzytelnianie hasła w usłudze PostgreSQL umożliwia tworzenie i używanie RÓL (nazw użytkowników) oraz używanie hasła do uwierzytelniania.
Włączenie uwierzytelniania w usłudze Azure Active Directory umożliwia tworzenie RÓL na podstawie kont usługi Azure Active Directory i generowanie tokenu na potrzeby uwierzytelniania. [więcej](#)

Metoda uwierzytelniania

- tylko uwierzytelnianie PostgreSQL
- Tylko uwierzytelnianie usługi Microsoft Azure Active Directory
- Uwierzytelnianie PostgreSQL i usługi Azure Active Directory

Nazwa użytkownika administratora *

Hasło *

Potwierdź hasło *

Szacowane koszty

Obliczanie jednostki SKU	USD	144.25/miesiąc
Standard_D2ds_v4 (rdzenie wirtualne2, USD72.12 na rdzeń wirtualny)	2 x	72.12
Magazyn	USD 16.26/miesiąc	
Wybrano magazyn 128 GiB (USD0.13 na GiB)	128 x 0.13	
Przepustowość		
W przypadku wychodzącego transferu danych między usługami w różnych regionach będą naliczane dodatkowe opłaty. Każdy przychodzący transfer danych jest bezpłatny. Dowiedz się więcej		
Szacowana suma USD 160.51/miesiąc		
Ceny odzwierciedlają tylko oszacowania. Wyświetl kalkulator cen platformy Azure . Koranic opłaty będą wyświetlane w walucie lokalnej w widokach analizy kosztów i rozliczeń.		

Przeglądanie + tworzenie **Następny: Sieć >**

Rys. 5.2: Konfiguracja danych do logowania administratora bazy danych PostgreSQL

Źródło: <https://portal.azure.com/#create/Microsoft.PostgreSQLServer>

Kolejnym krokiem było ustawienie reguł zapory dla bazy danych i wybrano publiczny dostęp z dowolnej usługi platformy Azure. Pozostałe opcje pozostawiono z wartościami domyślnymi i zaakceptowano konfigurację. Po kilkunastu sekundach baza danych została wdrożona, co zostało przedstawione na rysunku 5.3.

The screenshot shows the Azure portal interface for managing a PostgreSQL database. The main page includes:

- Basic Information:** Shows the server name (inzynierka-test.postgres.database.azure.com), administrator (kamil), configuration (Azure for Students), and other details like version (14.5) and availability (2).
- Tags:** A section to add tags to the database.
- Wprowadzenie (Getting Started):** A guide to connecting to the database.
- Rozpoczynanie projektu (Starting a Project):** Steps to connect from various platforms.
- Konfiguruj bazę danych do użytku produkcyjnego (Configure Database for Production):** Options for backup, configuration, and monitoring.
- Monitoring:** Links to alerts and metrics.

Rys. 5.3: Wdrożona baza danych PostgreSQL

Źródło: <https://portal.azure.com/#create/Microsoft.PostgreSQLServer>

5.2. Aplikacja kliencka oraz serwer webowy

Do wdrożenia aplikacji klienckiej postawionej na serwerze webowym NGINX wykorzystano usługę App Service.

Na początku przypisano usługę do subskrypcji studenckiej oraz grupy zasobów **inzynierka**, ustalono nazwę serwera, wybrano wdrożenie aplikacji poprzez kontener Docker, lokalizację serwera, a pozostałe podstawowe opcje pozostawiono jako domyślne. Podstawowa konfiguracja jest przedstawiona na rysunku 5.4.

Microsoft Azure

Strona główna > App Services > Utwórz aplikację internetową ...

Podstawowe Docker Sieć Monitorowanie Tagi Przeglądanie + tworzenie

Funkcja App Service Web Apps umożliwia szybkie tworzenie, wdrażanie i skalowanie aplikacji internetowych, aplikacji mobilnych i aplikacji interfejsu API klasy korporacyjnej uruchamianych na dowolnej platformie. Spełniaj rygorystyczne wymagania dotyczące wydajności, skalowalności, zabezpieczeń i zgodności, używając w pełni zarządzanej platformy do konserwacji infrastruktury. [Dowiedz się więcej](#)

Szczegóły projektu

Wybierz subskrypcję, aby zarządzać wdrożonymi zasobami i kosztami. Użyj grup zasobów jak folderów, aby organizować wszystkie Twoje zasoby i zarządzać nimi.

Subskrypcja * Grupa zasobów * [Utwórz nowy](#)

Szczegóły wystąpienia

Potrzebujesz bazy danych? [Wypróbowaj nowe środowisko Internet i baza danych](#).

Nazwa * .azurewebsites.net

Publikuj * Kontener Docker Statyczna aplikacja internetowa

System operacyjny * Linux Windows

Region * [Nie możesz znaleźć planu usługi App Service? Spróbuj użyć innego regionu lub wybierz środowisko App Service Environment.](#)

Plany cenowe

Od warstwy cenowej planu usługi App Service zależy lokalizacja, funkcje, koszty i zasoby obliczeniowe skojarzone z aplikacją. [Dowiedz się więcej](#)

Plan systemu Linux (North Europe) * [Utwórz nową](#)

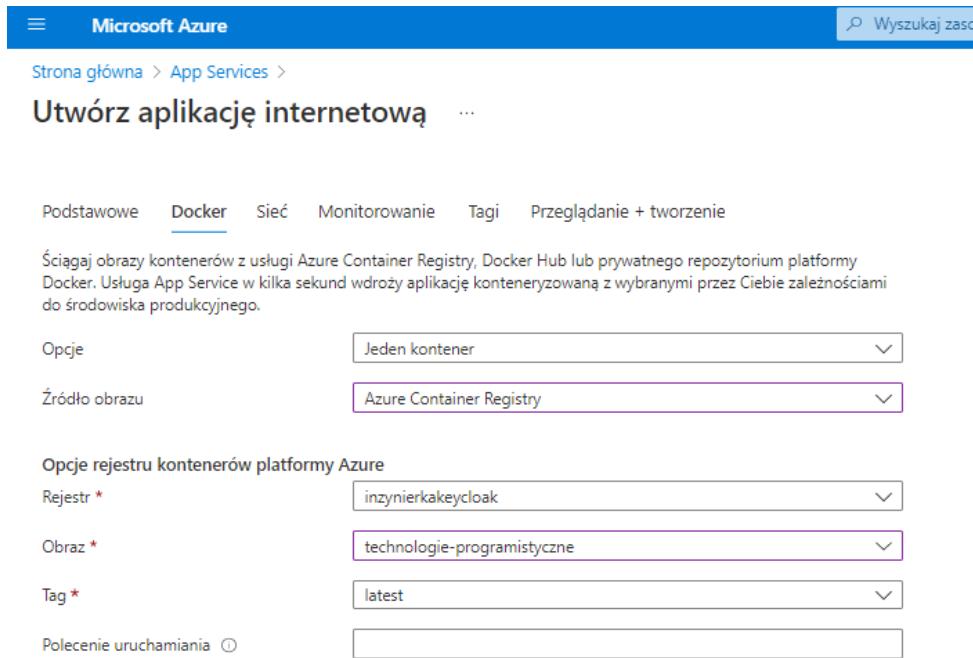
Plan cenowy * Łączna liczba jednostek ACU: 100, 1.75 GB pamięci

[Przeglądanie + tworzenie](#) [Poprzednia](#) [Następny: Docker >](#)

Rys. 5.4: Podstawowa konfiguracja wdrożeniowa aplikacji klienckiej oraz serwera webowego

Źródło: <https://portal.azure.com/#create/Microsoft.WebSite>

Kolejnym krokiem było podanie opcji dla Dockera. Wybrano jeden kontener, źródło obrazu ustawiono jako rejestr kontenerów Azure i na końcu podano nazwę rejestru oraz nazwę obrazu, na podstawie którego będzie tworzony kontener aplikacji. Konfiguracja Dockera po stronie Azure jest przedstawiona na rysunku 5.5.



Rys. 5.5: Konfiguracja Dockera po stronie Azure dla aplikacji klienckiej

Źródło: <https://portal.azure.com/#create/Microsoft.WebSite>

Resztę opcji pozostawiono z wartościami domyślnymi i zaakceptowano konfigurację. Po kilku sekundach usługa została utworzona i można było zauważyc widok przedstawiony na rysunku 5.6.

The screenshot shows the Microsoft Azure portal interface for managing an App Service. The left sidebar contains navigation links for 'Omówienie' (Overview), 'Wdrażanie' (Deployment), and 'Ustawienia' (Settings). The main panel displays basic app information: Grupa zasobów (Resource Group: inzynierka), Adres URL (https://inzynierka-test.azurewebsites.net), Stan (Running), Lokalizacja (Location: North Europe), Subskrypcja (Subscription: Azure for Students), and Identyfikator subskrypcji (Subscription ID: 632dcbeb-684b-4de7-a4ae-2cacb45609fe). It also shows 'Podstawowe elementy' (Basic elements) such as Plan usługi App Service (bazy3-front (B1: 0)), System operacyjny (Linux), and Sprawdzanie kondycji (Nieskonfigurowane). The 'Właściwości' (Properties) tab is selected, showing details for the application internet (Nazwa: inzynierka-test, Model publikowania: Kontener, Obraz kontenera: inzynierkakeycloak.azurecr.io/technologie-programisty...), Application Insights (Nazwa: Włącz funkcję Application Insights), Centrum wdrażania (Dzienniki wdrażania: Wyświetl dzienniki), Hosting (Typ planu: Plan usługi App Service, Nazwa: bazy3-front), and Skaluj w górę (plan usługi App).

Rys. 5.6: Utworzona usługa App Service dla aplikacji klienckiej

Źródło: <https://portal.azure.com/#@student.pwr.edu.pl/resource/subscriptions/632dcbeb-684b-4de7-a4ae-2cacb45609fe/resourceGroups/inzynierka/providers/Microsoft.Web/sites/inzynierka-test/appServices>

Po tych krokach usługa była skonfigurowana do automatycznego pobierania z rejestru kontenerów obrazu aplikacji.

Kolejnymi krokami było przygotowanie obrazu aplikacji wraz z serwerem webowym oraz wrzucenie tego obrazu na odpowiedni rejestr kontenerów.

Na początku za pomocą komendy `docker login inzynierkakeycloak.azurecr.io` zalogowano się do rejestru kontenerów Azure.

Następnie utworzono i odpowiednio skonfigurowano plik `Dockerfile`, tak aby na jego podstawie był tworzony obraz aplikacji klienckiej wystawionej na serwerze webowym. Konfiguracja obrazu dockerowego jest przedstawiona na listingu 5.1.

Listing 5.1: Konfiguracja Dockerfile dla aplikacji klienckiej i serwera webowego

```
FROM node:16.14.0 as build
WORKDIR /frontend
COPY .
RUN npm install
RUN npm run build

FROM nginx:stable-alpine
```

```
COPY --from=build /frontend/build /usr/share/nginx/html
COPY --from=build /frontend/nginx/nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE $PORT
CMD ["nginx", "-g", "daemon off;"]
```

Zdefiniowany plik Dockerfile jest typu **Multi-stage build**, gdyż zawiera wiele dyrektyw FROM i tym samym możliwe jest skonfigurowanie kilku technologii na raz. Na początku budowana jest aplikacja kliencka, a następnie aplikacja ta zostanie wystawiona na skonfigurowanym serwerze webowym NGINX.

Konfiguracja serwera NGINX jest podstawowa i znajduje się w pliku default.conf. Serwer NGINX został skonfigurowany za pomocą pliku default.conf. Konfiguracja ta jest bardzo podstawowa, gdyż ustawił jedynie niezbędne opcje do renderowania aplikacji SPA oraz reguły CORS (ang. *Cross-Origin Resource Sharing*) zezwalające na dostęp z każdego źródła. Konfiguracja ta jest przedstawiona na listingu 5.2.

Listing 5.2: Konfiguracja serwera webowego NGINX

```
1 server {
2
3     add_header 'Access-Control-Allow-Origin' '*';
4
5     location / {
6         root /usr/share/nginx/html;
7         index index.html index.htm;
8         try_files $uri $uri/ /index.html;
9         add_header 'Access-Control-Allow-Origin' '*';
10        add_header 'Access-Control-Allow-Credentials' 'true';
11        add_header 'Access-Control-Allow-Methods' 'GET, POST,
12                    → PUT, DELETE, OPTIONS';
13        add_header 'Access-Control-Allow-Headers' 'Access-
14                    → Control-Allow-Headers, AUTHORIZATION, CONTENT-TYPE
15                    → ';
```

Po skonfigurowaniu aplikacji klienckiej oraz serwera webowego, można było przystąpić do zbudowania obrazu Dockerowego. Za pomocą komendy docker build . -t frontend zbudowano obraz o nazwie „frontend”.

Następnym krokiem było powiązanie zbudowanego obrazu z nazwą jaką będzie miał ten obraz w rejestrze kontenerów. Do tego celu użyto komendy docker tag frontend inzynierkakeycloak.azurecr.io/technologie-programistyczne:latest.

Ostatnim krokiem było wrzucenie tak skonfigurowanego obrazu do rejestru kontenerów, gdzie na podstawie tego obrazu zostanie utworzony kontener aplikacji. Do tego celu użyto komendy docker push inzynierkakeycloak.azurecr.io/technologie-programistyczne:latest.

Po kilku minutach aplikacja została wdrożona i po przejściu do adresu strony <https://technologie-programistyczne.azurewebsites.net/>, można było zauważać widok strony głównej (rysunek 5.7).



Rys. 5.7: Widok strony głównej aplikacji po jej wdrożeniu

Źródło: <https://inzynierka-test.azurewebsites.net/>

5.3. Aplikacja serwerowa

Do wdrożenia aplikacji serwerowej wykorzystano usługę **App API** z zakładki „Marketplace”.

Konfiguracja wdrożeniowa backendu po stronie Azure jest prawie identyczna jak w przypadku aplikacji klienckiej i różni się jedynie nazwą aplikacji oraz nazwą obrazu.

Podsumowanie konfiguracji wdrożeniowej aplikacji serwerowej jest przedstawione na rysunku 5.8.

Szczegóły

Subskrypcja	632dcbbc-684b-4de7-a4ae-2cacb45609fe
Grupa zasobów	inzynierka
Nazwa	inzynierka-test-backend
Publikuj	Kontener Docker
Image: Tag	inzynierkakeycloak.azurecr.io/technologie-programistyczne-backend:latest
Adres URL serwera	https://inzynierkakeycloak.azurecr.io

Plan usługi App Service

Nazwa	bazy3-front
System operacyjny	Linux
Region	North Europe
Jednostka SKU	Podstawowa
Rozmiar	Mały
ACU	Łączna liczba jednostek ACU: 100
Pamięć	1.75 GB pamięci

Monitorowanie

Application Insights	Nie włączono
----------------------	--------------

Wdrożenie

Ciągłe wdrażanie	Nie włączone / Konfigurowanie po utworzeniu aplikacji
------------------	---

[Utwórz](#)
 [< Poprzednia](#)
 [Dalej >](#)
 [Pobierz szablon do automatyzacji](#)

<https://portal.azure.com/#home>

Rys. 5.8: Podsumowanie konfiguracji aplikacji wdrożeniowej

Źródło: <https://portal.azure.com/#create/Microsoft.ApiApp>

Po tych krokach usługa była skonfigurowana do automatycznego pobierania z rejestru kontenerów obrazu aplikacji.

Kolejnymi krokami było przygotowanie obrazu backendu oraz wrzucenie tego obrazu na odpowiedni rejestr kontenerów.

Na początku za pomocą komendy `docker login inzynierkakeycloak.azurecr.io` zalogowano się do rejestrów kontenerów Azure.

Następnie utworzono i odpowiednio skonfigurowano plik `Dockerfile`, tak aby na jego podstawie był tworzony obraz aplikacji serwerowej uruchomionej na profilu produkcyjnym. Konfiguracja obrazu dockerowego jest przedstawiona na listingu 5.3.

Listing 5.3: Konfiguracja Dockerfile dla aplikacji serwerowej

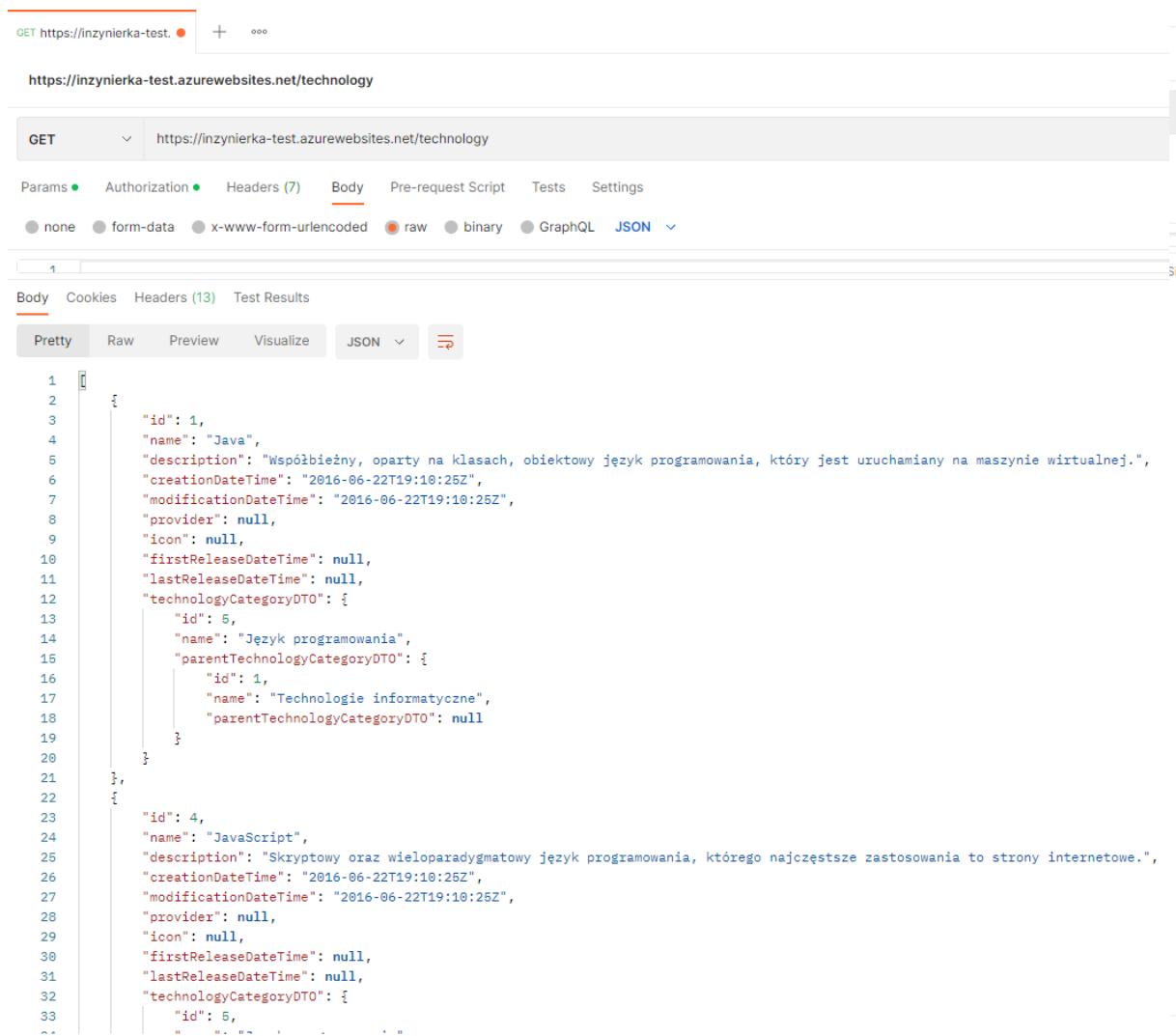
```
ARG PROFILE
FROM maven:3.8.4-openjdk-17
WORKDIR /backend
COPY .
EXPOSE $PORT
CMD mvn spring-boot:run -Dspring.profiles.active=${PROFILE}
```

Następnym krokiem było zbudowanie obrazu Dockerowego za pomocą komendy `docker build . -t backend`.

Następnie powiązano zbudowany obraz z nazwą jaką będzie miał ten obraz w rejestrze kontenerów. Do tego celu użyto komendy `docker tag backend inzynierkakeycloak.azurecr.io/technologie-programistyczne-backend:latest`.

Ostatnim krokiem było wrzucenie tak skonfigurowanego obrazu do rejestru kontenerów, gdzie na podstawie tego obrazu zostanie utworzony kontener aplikacji. Do tego celu użyto komendy `docker push inzynierkakeycloak.azurecr.io/technologie-programistyczne-backend:latest`.

Po kilkunastu minutach aplikacja została wdrożona i można już było wysłać do niej żądania. Przykładowe żądanie pobierające wszystkie technologie i uzyskana odpowiedź od wdrożonego backendu są przedstawione na rysunku 5.9.



Rys. 5.9: Przykładowe żądanie pobierające wszystkie technologie i uzyskana odpowiedź od wdrożonego backendu

Źródło: Opracowanie własne - zrzut ekranu z programu Postman

5.4. Warstwa bezpieczeństwa

Do wdrożenia warstwy bezpieczeństwa wykorzystano usługę App Service. Konfiguracja wdrożeniowa warstwy bezpieczeństwa po stronie Azure również jest prawie identyczna jak w przypadku aplikacji klienckiej i różni się jedynie nazwą aplikacji oraz źródłem i nazwą obrazu.

Podsumowanie konfiguracji wdrożeniowej serwera uwierzytelniającego i autoryzującego jest przedstawione na rysunku 5.10.

The screenshot shows the Microsoft Azure portal interface for creating a new app service. The top navigation bar includes 'Microsoft Azure', 'Strona główna > App Services >', 'Utwórz aplikację internetową ...', and tabs for 'Podstawowe', 'Docker', 'Sieć', 'Monitorowanie', 'Tagi', and 'Przeglądanie + tworzenie' (which is underlined). Below these tabs is a 'Podsumowanie' section containing a summary of the application's configuration.

Szczegóły	
Subskrypcja	632dc0cb-684b-4de7-a4ae-2cacb45609fe
Grupa zasobów	inzynierka
Nazwa	inzynierka-test-keycloak1
Publikuj	Kontener Docker
Image:Tag	quay.io/keycloak/keycloak:18.0.1
Adres URL serwera	https://index.docker.io

Plan usługi App Service	
Nazwa	bazy3-front
System operacyjny	Linux
Region	North Europe
Jednostka SKU	Podstawowa
Rozmiar	Mały
ACU	Łączna liczba jednostek ACU: 100
Pamięć	1.75 GB pamięci

Monitorowanie	
Application Insights	Nie włączono

Wdrożenie	
Ciągłe wdrażanie	Nie włączone / Konfigurowanie po utworzeniu aplikacji

[Utwórz](#)
 [< Poprzednia](#)
 [Dalej >](#)
 [Pobierz szablon do automatyzacji](#)

Rys. 5.10: Podsumowanie konfiguracji serwera uwierzytelniającego i autoryzującego

Źródło: <https://portal.azure.com/#create/Microsoft.WebSite>

Po tych krokach usługa była skonfigurowana do automatycznego pobierania z rejestru kontenerów obrazu aplikacji.

Użyty obraz `quay.io/keycloak/keycloak:18.0.1` zawiera gotową do uruchomienia aplikację Keycloak, którą wystarczy jeszcze jedynie skonfigurować do swoich potrzeb. Konfigurację aplikacji przeprowadzono w zakładce „Konfiguracja” w sekcji „Ustawienia”. Użyta konfiguracja jest przedstawiona na rysunku 5.11

5.4. Warstwa bezpieczeństwa

The screenshot shows the Azure portal interface for managing an App Service. The left sidebar has a tree view with 'inzynierka-test-keycloak1 | Konfiguracja' selected under 'App Service'. The main content area is titled 'Ustawienia aplikacji' (Application Settings). It displays a table of environment variables with their current values:

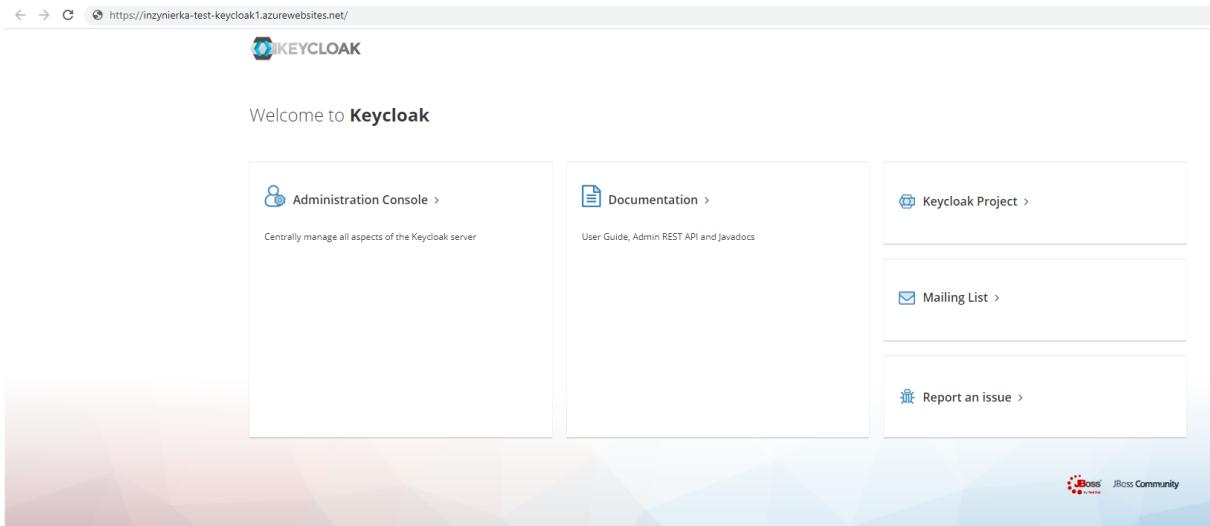
Nazwa	Wartość
DOCKER_REGISTRY_SERVER_PASSWORD	REDACTED
DOCKER_REGISTRY_SERVER_URL	https://index.docker.io
DOCKER_REGISTRY_SERVER_USERNAME	REDACTED
KC_DB	postgres
KC_DB_PASSWORD	postgreS%
KC_DB_URL	jdbcpostgresql://technologie-programistyczne.postgres.database.azure.com:5432/programming_technologies?sslfmode=require
KC_DB_USERNAME	programming_technologies
KC_PROXY	edge
KC_PROXY_ADDRESS_FORWARDING	true
KEYCLOAK_ADMIN	admin
KEYCLOAK_ADMIN_PASSWORD	REDACTED
WEBSITES_CONTAINER_START_TIME_LIMIT	1800
WEBSITES_ENABLE_APP_SERVICE_STORAGE	false

Below the table, there's a section for 'Parametry połączeń' (Connection parameters) with a note: 'Parametry połączenia są szyfrowane podczas magazynowania i przesyłane za pośrednictwem szyfrowanego kanału.'

Rys. 5.11: Konfiguracja serwera Keycloak

Źródło: <https://portal.azure.com/#@student.pwr.edu.pl/resource/subscriptions/632dcbeb-684b-4de7-a4ae-2cacb45609fe/resourcegroups/inzynierka/providers/Microsoft.Web/sites/inzynierka-test-keycloak1/appServices>

Po kilkudziesięciu sekundach aplikacja została wdrożona i po przejściu pod adres <https://inzynierka-test-keycloak1.azurewebsites.net/> można było zobaczyć interfejs użytkownika (rysunek 5.12).

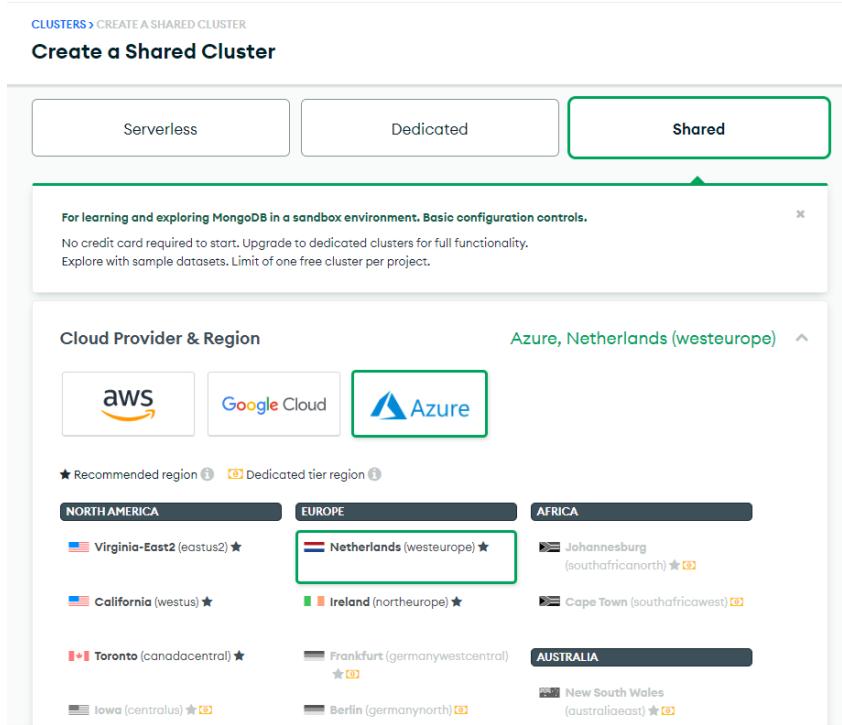


Rys. 5.12: Interfejs użytkownika dla wdrożonego serwera Keycloak

Źródło: <https://inzynierka-test-keycloak1.azurewebsites.net/>

5.5. Baza danych MongoDB

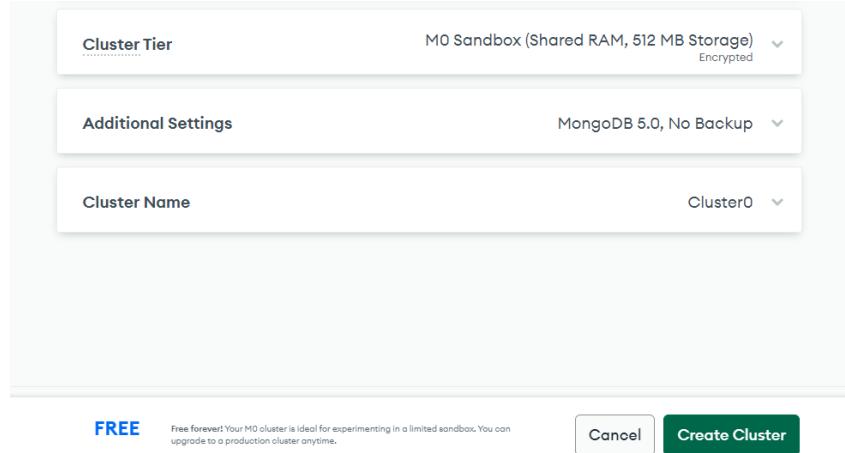
Pierwszym krokiem wdrożenia bazy danych MongoDB było skonfigurowanie i stworzenie klastra. Wybrano darmowy typ klastra, Azure jako dostawcę chmury oraz lokalizację bazy danych w Holandii (rysunek 5.13).



Rys. 5.13: Podstawowa konfiguracja wdrożeniowa bazy danych MongoDB

Źródło: <https://cloud.mongodb.com/v2/63322815dadae81bf1ca9139#clusters/edit?from=ctaClusterHeader>

Kolejnym krokiem było ustawienie zasobów bazy, wersji MongoDB oraz nazwy klastra (rysunek 5.14).



Rys. 5.14: Dodatkowa konfiguracja wdrożeniowa bazy danych MongoDB

Źródło: <https://cloud.mongodb.com/v2/63322815dadae81bf1ca9139#clusters/edit?from=ctaClusterHeader>

Następnie kliknięto przycisk „Create Cluster” i baza danych MongoDB została wdrożona.

5.6. Podsumowanie wdrożenia

Po wdrożeniu systemu konieczne jest jego przetestowanie, czy wszystkie zaimplementowane funkcjonalności działają w tak sam sposób jak w środowisku lokalnym. Dogłębnie przetestowano każdy z wdrożonych segmentów i wszystko działa zgodnie z oczekiwaniemi.

Rozdział 6

Podsumowanie

W ramach niniejszej pracy zaprojektowano i zaimplementowano aplikację webową IT Tech do gromadzenia i udostępniania informacji o technologiach programistycznych w postaci artykułów. Użytkownicy mają możliwość tworzenia, publikowania i przeglądania artykułów, jak i również mogą zostawiać informacje zwrotne do danych artykułów w postaci komentarza lub opinii.

Zaimplementowano system zabezpieczony w oparciu o role użytkowników. Jest system logowania i rejestracji. Przede wszystkim udało się zrealizować pełny cykl życia artykułu. Użytkownicy mogą tworzyć i edytować artykuły w przejrzystym formularzu. Został zaimplementowany system recenzji artykułów, w którym autor artykułu może m.in. przejrzeć uzasadnienie recenzenta co do wyniku weryfikacji. Użytkownicy mogą zarządzać swoimi danymi osobowymi. Zostały zaimplementowane mechanizmy sortowania oraz wyszukiwania artykułów według wielu kryteriów. Użytkownicy mogą prowadzić dyskusję na temat artykułu w sekcji komentarzy. Został zaimplementowany system opiniowania artykułów, który wymaga umieszczenia tekstowego uzasadnienia wystawionej oceny. Użytkownicy mogą wyrazić swoje zdanie o danej opinii w postaci oceny pozytywnej albo negatywnej. Stworzono rozbudowany panel administratora, w którym administratorzy mogą zarządzać danymi użytkowników, czy przypisywać im role. Na końcu udało się wdrożyć cały system, który jest dostępny cały czas pod adresem: <https://technologie-programistyczne.azurewebsites.net/>.

Aplikację udało się zrealizować prawie w całości. Z powodu niewystarczającego czasu nie udało się zrealizować wyszukiwania i sortowania technologii oraz zarządzania technologiami przez recenzenta oraz przetestowano wystarczająco backendu i nie wykonano testów aplikacji klienckiej. Na początku planowano wykonać testy integracyjne kontrolerów backendu w całości oraz dla dużej części funkcji frontendu. Przetestowano jedynie backend, ale i tak pokrycie tych testów było mniejsze niż 50%.

Podczas realizacji aplikacji najwięcej trudności mi przysporzyły użycie technologii Keycloak bez wymuszonego przekierowania do formularza logowania dostarczanego przez ten serwer oraz użycie dwóch różnego typu baz danych w jednym projekcie Spring. Problemy te wynikały głównie z braku doświadczenia.

Dzięki pracy nad tym projektem udało się zdobyć dużo doświadczenia z zakresu projektowania systemów informatycznych, implementacji aplikacji z wykorzystaniem technologii webowych, przeprowadzaniu testów jednostkowych oraz wdrożenia systemów na popularnych platformach chmurowych.

Literatura

- [1] Strona główna Wikipedia. <https://pl.wikipedia.org/>. [Dostęp dnia 13.12.2022].
- [2] Specyfikacja języka Java w wersji 17. <https://docs.oracle.com/javase/specs/jls/se17/jls17.pdf/>. [Dostęp dnia 13.12.2022].
- [3] Strona główna Medium. <https://medium.com/>. [Dostęp dnia 13.12.2022].
- [4] Dokumentacja języka Java w wersji 17. <https://docs.oracle.com/en/java/javase/17/>. [Dostęp dnia 13.12.2022].
- [5] Dokumentacja frameworka Spring. <https://docs.spring.io/spring-framework/docs/current/reference/html/>. [Dostęp dnia 13.12.2022].
- [6] Dokumentacja biblioteki React. <https://reactjs.org/docs/getting-started.html/>. [Dostęp dnia 13.12.2022].
- [7] Dokumentacja języka TypeScript. <https://redux.js.org/api/api-reference/>. [Dostęp dnia 13.12.2022].
- [8] Dokumentacja biblioteki Redux. <https://redux.js.org/api/api-reference/>. [Dostęp dnia 13.12.2022].
- [9] Dokumentacja biblioteki MUI. <https://mui.com/material-ui/getting-started/overview/>. [Dostęp dnia 13.12.2022].
- [10] Dokumentacja PostgreSQL. <https://www.postgresql.org/docs/>. [Dostęp dnia 13.12.2022].
- [11] Dokumentacja MongoDB. <https://www.mongodb.com/docs/>. [Dostęp dnia 13.12.2022].
- [12] Dokumentacja NGINX. <https://nginx.org/en/docs/>. [Dostęp dnia 13.12.2022].
- [13] Dokumentacja Keycloak. <https://www.keycloak.org/documentation/>. [Dostęp dnia 13.12.2022].
- [14] Dokumentacja LaTeX. <https://www.latex-project.org/help/documentation/>. [Dostęp dnia 13.12.2022].
- [15] Dokumentacja VisualParadigm. <https://www.visual-paradigm.com/support/documents/>. [Dostęp dnia 13.12.2022].
- [16] Dokumentacja Figma. <https://help.figma.com/hc/en-us/categories/360002051613-Get-started/>. [Dostęp dnia 13.12.2022].
- [17] Dokumentacja pgAdmin. <https://www.pgadmin.org/docs/>. [Dostęp dnia 13.12.2022].
- [18] Dokumentacja MongoDBCompass. <https://www.mongodb.com/docs/compass/current/>. [Dostęp dnia 13.12.2022].

- [19] Dokumentacja Visual Studio Code. <https://code.visualstudio.com/docs/>. [Dostęp dnia 13.12.2022].
- [20] Dokumentacja ItelliJ. <https://www.jetbrains.com/help/idea/getting-started.html/>. [Dostęp dnia 13.12.2022].
- [21] Dokumentacja Postman. <https://learning.postman.com/docs/publishing-your-api/documenting-your-api/>. [Dostęp dnia 13.12.2022].
- [22] Dokumentacja REST-assured. <https://github.com/rest-assured/rest-assured/wiki/GettingStarted>. [Dostęp dnia 13.12.2022].
- [23] Dokumentacja Swagger. <https://swagger.io/docs/>. [Dostęp dnia 13.12.2022].
- [24] Dokumentacja Docker. <https://docs.docker.com/>. [Dostęp dnia 13.12.2022].
- [25] Dokumentacja MongoDB Cloud. <https://www.mongodb.com/docs/atlas/>. [Dostęp dnia 13.12.2022].
- [26] Dokumentacja Azure. <https://learn.microsoft.com/en-us/azure/?product=popular>. [Dostęp dnia 13.12.2022].

Dodatek A

Instrukcja wdrożeniowa

W tej instrukcji zostanie pokazana procedura uruchomienia systemu w trybie lokalnym.

Do uruchomienia systemu są wymagane następujące narzędzia:

- Node.js (wersja v18.0.0) oraz npm (wersja 8.6.0) aby uruchomić frontend,
- Maven (wersja 3.6.3) oraz JDK (minimum wersja 17) aby uruchomić backend,
- PostgreSQL - postawiona baza danych o nazwie `programming_technologies`, loginie `programming_technologies` oraz haśla `postgres`,
- MongoDB - postawiona baza danych o nazwie `technologie-it`, loginie `programming_technologies` oraz haśla `postgres`.

W folderze `W04N_254049_2022_aplikacja` znajdują się następujące foldery:

- `frontend` - przechowujący kod aplikacji klienckiej,
- `backend` - zawierający kod aplikacji serwerowej,
- `keycloak` - zawierający serwer Keycloak.

Aby uruchomić aplikację kliencką, należy z folderu `frontend` uruchomić w konsoli komendy `npm install` do zainstalowania zależności oraz następnie `npm start` do uruchomienia aplikacji.

W celu uruchomienia aplikacji serwerowej, należy z folderu `backend` uruchomić w konsoli komendę `mvn spring-boot:run`.

Do uruchomienia serwera Keycloak, potrzebne będzie wywołanie ze ścieżki `keycloak/bin` komendy `kc.bat start-dev`.

Po tych czynnoścach aplikacja będzie dostępna pod adresem `localhost:3000`.

Dodatek B

Opis załączonej płyty CD/DVD

Na dołączonej do tej pracy płycie CD znajdują się wersja elektroniczna pracy o nazwie `W04N_254049_2022_praca_inżynierska.pdf` oraz kod aplikacji w folderze o nazwie `W04N_254049_2022_aplikacja`. Kod aplikacji podzielony jest na 3 foldery: `frontend`, `backend` oraz `keycloak`. W folderze `frontend` znajduje się kod aplikacji klienckiej. W folderze `backend` jest zawarty kod aplikacji serwerowej. W folderze `keycloak` przechowywany jest serwer Keycloak w wersji samodzielnej (ang. *standalone*).