JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 40–45
Received: 24 October 2021
Accepted: 27 October 2021

Comparison of ASP.NET Core and Spring Boot ecosystems

Porównawnanie ekosystemów ASP.NET Core i Spring Boot

Petro Kopyl*, Teofil Rozaliuk* , Jakub Smołka

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The article describes a comparative analysis of the ASP.NET Core and Spring Boot framework ecosystems. The research was carried out on the basis of implemented two applications with identical functionality, which use the PostgreSQL database engine. In the implementation of the application, appropriate ORM (Object-Relational Mapping) tools were used to perform database operations, ie Spring Data and Entity Framework Core, technologies enabling the implementation of authentication and authorization (Spring Security and ASP.NET Core Identity) and several additional libraries that simplify the entire process of building the application. The criteria of comparison were the ease and intuitiveness of a given tool in the implementation of the application, the offered possibilities of the tools implementing authentication and authorization mechanisms and the efficiency of database operations. Based on the research, it was found that Spring Data technology is a faster tool than Entity Framework Core, while Spring Security, unlike Asp.Net Core Identity, is less integrated into the framework ecosystem, because it has a smaller set of ready-made solutions for database integration.

Keywords: Spring Boot; ASP.NET Core; comparative analysis.

Streszczenie

Artykuł opisuje analizę porównawczą ekosystemów szkieletów aplikacji ASP.NET Core oraz Spring Boot. Badania przeprowadzono w oparciu o dwie autorskie aplikacje o identycznej funkcjonalności, wykorzystujące silnik bazodanowy PostgreSQL. W implementacji aplikacji wykorzystano odpowiednie narzędzia ORM (ang. Object-Relational Mapping) do wykonywania operacji bazodanowych tj. Spring Data i Entity Framework Core, technologie umożliwiające implementację uwierzytelniania i autoryzacji (Spring Security i ASP.NETCore Identity) oraz kilka dodatkowych bibliotek, które upraszczają cały proces budowania aplikacji. Kryteriami porównania były łatwość i intuicyjność danych technologii w implementacji aplikacji, oferowane możliwości narzędzi implementujących mechanizmy uwierzytelniania i autoryzacji oraz wydajność operacji bazodanowych. Na podstawie wyników przeprowadzonych badań stwierdzono, że technologia Spring Data jest szybszym narzędziem od Entity Framework Core, natomiast Spring Security w odróżnieniu od Asp.Net Core Identity jest mniej zintegrowany w ekosystem frameworku, ponieważ posiada mniejszy zestaw gotowych rozwiązań do integracji z bazą danych.

Słowa kluczowe: Spring Boot; ASP.NET Core; analiza porównawcza.

*Corresponding author

Email address: **petro.kopyl@pollub.edu.pl** (P. Kopyl), **teofil.rozaliuk@pollub.edu.pl** (T. Rozaliuk)

1. Wstęp

W dzisiejszych czasach istnieje wiele różnych frameworków do budowy aplikacji serwerowych. Każdy z nich posiada swoje wady oraz zalety. W zależności od wymagań projektowanych systemów oraz innych czynników, takich jak doświadczenie programisty, koszty wsparcia projektu itd., wybór właściwego frameworku odgrywa istotną rolę przy implementacji systemu i pozwoli na stworzenie wydajnych i niezawodnych aplikacji webowych. Dodatkowo różne szkielety aplikacji zawierają swoje rozwiązania do implementacji poszczególnych funkcjonalności, wykorzystując gotowe biblioteki, które składają się na ekosystem tego frameworku. W danej pracy do badania wzięto dwa najbardziej popularne frameworki: Spring Boot [1] oraz ASP.NET Core [2].

W 2004 roku została wydana pierwsza wersja Spring Framework - framework z otwartym kodem źródłowym dla platformy Java. Dane narzędzie powstało jako alternatywa Enterprise JavaBeans. Spring nadaje programistom Java większą swobodę projektowania, ponadto zapewnia dobrze udokumentowane i łatwe w użyciu narzędzia do rozwiązywania problemów pojawiających się podczas tworzenia aplikacji. Spring można traktować jako zbiór mniejszych frameworków. Większość z nich może działać niezależnie od siebie, jednak razem zapewniają większą funkcjonalność. W 2016 roku dużą popularność osiągnął Spring Boot, ponieważ wprowadził automatyczną konfigurację projektów opartych o szkielet programistyczny Spring Framework, co znacznie przyspieszyło i uprościło pracę programistom.

ASP.NET Core to technologia tworzenia aplikacji webowych o otwartym kodzie wspierana przez Microsoft. Założona w 2016 roku jako następca technologii ASP.NET. Jest to modularna technologia wieloplatformowa (działa pod Windows, Linux oraz MacOS). Razem z językiem C# jest rozbudowanym, ciągle rozwijającym narzędziem dla programistów, które umożliwia łatwe wsparcie oraz migrację projektów do najnowszych wersji. Jedną

z głównych zalet jest duży zbiór narzędzi, ułatwiających implementację dla często się powtarzających scenariuszy biznesowych dla typowych aplikacji internetowych, takich jak np. uwierzytelnienie oraz praca z systemem bazodanowym.

2. Cel, teza i metody badań

Celem pracy jest przeprowadzenie analizy porównawczej ekosystemów dwóch szkieletów aplikacji Spring Boot oraz ASP.NET Core pod kątem wydajności wykonywania operacji bazodanowych, oraz wygody użycia narzędzi do implementacji systemów uwierzytelnienia i autoryzacji na podstawie prostej aplikacji internetowej zaimplementowanej w obu technologiach.

W niniejszym artykule zdefiniowano trzy tezy:

T1: Spring Data jest bardziej wydajnym narzędziem od Entity Framework Core przy współpracy z bazą danych PostgreSql.

T2: ASP.NET Core Identity jest bardziej elastycznym i wygodnym w użyciu narzędziem w porównywaniu do Spring Security.

Dla potwierdzenia wyżej wymienionych tez wykorzystano metodę badań opartą na analizie porównawczej obu frameworków. W tym celu stworzono dwie, funkcjonalnie identyczne, testowe aplikacje internetowe wykorzystując następujące narzędzia i technologie:

- szkielety programistyczne Spring Boot oraz ASP.NET Core;
- środowisko programistyczne IntelliJ IDEA oraz Visual Studio;
- serwer bazodanowy PostgreSQL [3];
- Maven - narzędzie do automatyzacji budowy projektów Java;
- Spring Security oraz ASP.NET Core Identity – narzędzia, które zapewnia mechanizmy budowania systemów uwierzytelniania i autoryzacji;
  - Spring Data [4] oraz Entity Framework Core [5] – technologie służące do realizacji warstwy dostępu do danych.

3. Przegląd literatury

Istnieje wiele prac, w których przeprowadzono analizę porównawczą różnych narzędzi do tworzenia aplikacji internetowych. W niniejszym artykule wybrano do porównania dwa najbardziej znaczące w dzisiejszych czasach frameworki do budowania aplikacji webowych Spring Boot oraz ASP.NET Core. Technologie często można spotkać w pracach badawczych w zestawie z innymi frameworkami, natomiast w literaturze nie znaleziono bezpośredniego ich porównania.

Przy budowaniu każdej nowoczesnej aplikacji internetowej bardzo ważnym elementem jest bezpieczeństwo. Ciekawym artykułem jest [6], w którym autor opisuje, jak ważne jest bezpieczeństwo, prezentuje przykłady przeprowadzonych różnego rodzaju ataków oraz porównuje rozwiązania, aby chronić swoje własne aplikacje przed nimi. Dodatkowo na podstawie prostej utworzonej przez autora aplikacji zostały przeprowadzone badania, w których wykonano ataki różnego rodzaju. Celem badań było wykluczenie lub potwierdzenie podatności aplikacji na ataki. W rezultacie okazało się, że nawet w prostej aplikacji internetowej, mogą być wykryte podatności na znane ataki oraz typowe błędy bezpieczeństwa tj. niewłaściwy sposób przechowywania hasła, lub błędy przy zarządzaniu sesją. Dla każdego z omówionych przypadków zostało przedstawione rozwiązanie mające na celu poprawę bezpieczeństwa.

Nie mniej ważnym czynnikiem podczas napisania aplikacji webowych jest poprawne wybranie i implementacje wzorcu architektonicznego. W większości aplikacji webowych używano wzorca MVC lub jego odmian. W artykule [7] autorzy porównują realizację tego wzorca w technologiach PHP oraz .Net Framework. Testowanie przeprowadzono na podstawie dwóch aplikacji o identycznych funkcjonalnościach, realizujących operacje dodawania, edycji oraz usuwania. Do napisania aplikacji w .Net Framework wykorzystano szkielet aplikacji webowych ASP.NET MVC oraz lokalną bazę danych SQL Server. Dla aplikacji napisanej w PHP użyto CakePHP oraz MySQL. Podczas testów okazało się, że .Net Framework w większości wskaźników zachowuje się lepiej od PHP, chociaż strony internetowe napisane w PHP są szybsze przy odbieraniu żądań i wysyłaniu odpowiedzi. W następnej pracy [8] wykonano porównanie kolejnych dwóch frameworków Spring Boot oraz JavaServer Faces na podstawie utworzonych w obu technologiach dwóch identycznych aplikacji internetowych. Jako kryteria dla analizy porównawczej brano pod uwagę ogólną strukturę aplikacji, podstawowe metryki kodu, efektywność wykonania operacji z danymi oraz efektywność ładowania zasobów przez przeglądarkę internetową. Struktura projektów aplikacji w obu technologiach jest podobna, natomiast bardziej wyraźne różnice widać przy porównaniu metryk kodu. Na podstawie pomiarów zauważono, że dla Spring Boot liczba linii kodu jest mniejsza niż w przypadku JSF, dlatego że Spring Boot zawiera wiele wbudowanych narzędzi, co z drugiej strony powoduje to, że rozmiar projektu jest znacznie większy.

W celu badania efektywności wykonywania operacji bazodanowych w obu aplikacjach zaimplementowano identyczne funkcjonalności: dodawanie, edycja oraz usuwanie rekordów. Do wykonywania pomiarów przygotowano kilka scenariuszy, gdzie każdy powtarzano 10 razy, a jako wynik pomiaru brano ich średni czas. Okazało się, że w przypadku pobierania, wyszukiwania oraz dodawania małej liczby rekordów Spring Boot jest znacznie szybszy od JSF, natomiast w przypadku dodawania większej liczby rekordów JSF jest nieznacznie bardziej efektywny. Jeśli chodzi o efektywność ładowania wszystkich niezbędnych zasobów aplikacji w przeglądarce internetowej, to tutaj też znacząco wygrywa Spring Boot. W rezultacie, na podstawie przeprowadzonych badań autorzy pracy, sformułowali wniosek, że

Spring Boot jest jednak korzystniejszym wyborem niż JavaServer Faces przy tworzeniu aplikacji internetowych JEE. Bardzo ciekawą pracą też jest praca [9] w której porównano od razu 4 różne szkielety programistyczne: Laravel, Rails, Django oraz Spring. Tutaj autorzy skupiają się na ważnych dla programisty kryteriach takich jak skalowalność aplikacji napisanych w wybranych technologiach, wsparcie wielojęzyczności, wsparcie urządzeń mobilnych oraz trendy biznesowe. W rezultacie okazało się, że wszystkie frameworki, które brały udział w porównaniu, spełniają wybrane kryteria, a różnice między nimi są nieznaczące. Liderem z nieznaczną przewagą został wybrany Spring Framework. Aplikacje napisane przy użyciu tego narzędzia są bardziej skalowalne. Spring ma też lepszą i bardziej rozbudowaną dokumentację, która wynika z dojrzałości technologii. Aby lepiej ocenić wybrane frameworki, autorzy również wprowadzili dodatkowe kryteria porównawcze, które głównie skupiają się na ocenie popularności tych frameworków wśród społeczności deweloperów oraz łatwość napisania stron internetowych w wybranych technologiach. Sporą przewagę uzyskał tutaj framework Django. Ma on niski próg wejścia i doskonale nadaje się do pisania dużych programów w małych ramach czasowych. Ważnym czynnikiem przemawiającym za tą technologią jest aktywne wsparcie przez społeczność programistów na platformach reddit, GitHub oraz Stack Overflow.

Oczywiście, należy pamiętać, że w epoce internetu, kiedy ciągle powstają nowe technologie, rozwijają się stare albo odwrotnie przestają być wpierane i używane, wszystko bardzo szybko się zmienia. Dlatego niektóre wyniki testów przeprowadzonych w poszczególnych wyżej opisanych pracach są już niestety nieaktualne.

4. Aplikacje testowe

Do porównania ekosystemów obu szkieletów aplikacji webowych została stworzona aplikacja testowa ─ messanger webowy. Do realizacji aplikacji wykorzystano wszystkie wymienione w punkcie 2. narzędzia. Dzięki nim zaimplementowano takie funkcjonalności jak typowe operacje CRUD (ang. Create, Read, Update, Delete) [10], uwierzytelnianie oraz autoryzacja użytkowników.

W obu rozwiązaniach wykorzystano serwer bazodanowy PostgreSQL. Schemat samej bazy danych jest przedstawiony na Rysunku 1.

5. Analiza porównawcza

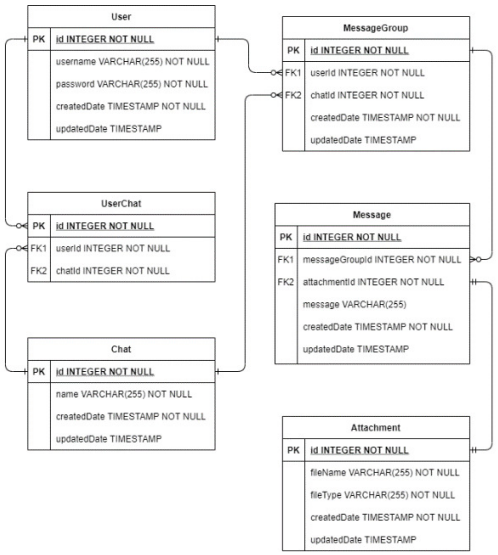
Aplikacje testowe porównano według następujących kryteriów:

- wydajność operacji bazodanowych,
- łatwość implementacji uwierzytelnienia oraz autoryzacji.

Wszystkie testy zostały przeprowadzone na komputerze o parametrach przedstawionych w Tabeli 1.

Tabela 1: Parametry sprzętowe komputera

Parametr	Wartość
System operacyjny	Windows 10
Procesor	Intel Core i7
Pamięć RAM	8 GB
Wersja PostgreSQL	13
Wersja SpringBoot	2.5.2
Wersja ASP.NET Core	5.0.7



Rysunek 1: Schemat bazy danych dla testowej aplikacji.

5.1. Wydajność operacji bazodanowych

W celu porównania wydajności wykonania operacji bazodanowych dla obu aplikacji zmierzono ich czasy. W badaniach brano pod uwagę trzy podstawowe operacje: dodawanie rekordów do bazy, ich pobieranie z bazy oraz usuwanie. Aby obiektywnie ocenić wydajność, dla każdej z nich przygotowano po 6 scenariuszy, które przedstawione są w Tabeli 2. Pomiary dla każdego scenariusza powtarzano 1000 razy i jako wynik przyjęto wartość średnią.

Operacje bazodanowe są dość szybkie, dla tego ich czasy wykonywania mierzono w milisekundach. Samo wywoływanie tych operacji zostało zrealizowane przy pomocy bibliotek ORM: w przypadku aplikacji w Spring Boot jest to Spring Data oraz Entity Framework Core dla aplikacji w ASP.NET Core. Czas mierzono bezpośrednio w kodzie. Brano pod uwagę wyłącznie czas wykonywania operacji bazodanowej, żadna logika biznesowa nie była uwzględniana. Przykład wykonywania pomiaru czasu dla operacji zapisu grupy powiadomień w aplikacji w Spring Boot przedstawiono na Listingu 1.

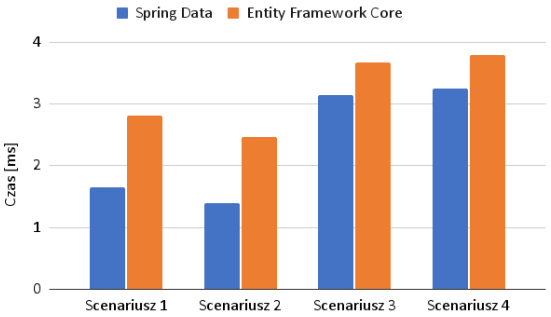
Listing 1: Przykład mierzenie czasu operacji bazodanowej

```
Long start = System.currentTimeMillis();
MessageGroup savedMessageGroup = messageGroupRepository.save(newMessageGroup);
Long end = System.currentTimeMillis();
operationService.create( timespan: end - start, operationType);
```

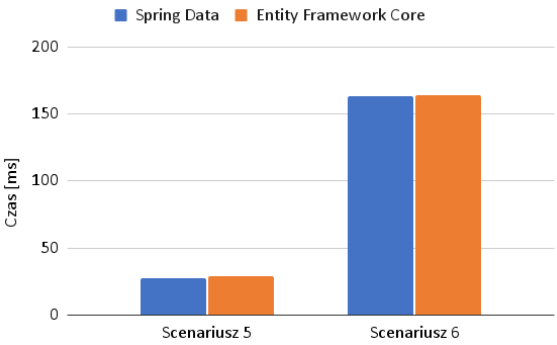
Tabela 2: Scenariusze testowe

Scenariusz	Opis
1	Zapis grupy powiadomień z jedną krótką (<10 znaków) wiadomością tekstową bez załącznika
2	Zapis grupy powiadomień z jedną długą (>200 znaków) wiadomością tekstową bez załącznika
3	Zapis grupy powiadomień z jednym załącznikiem bez wiadomości tekstowej
4	Zapis grupy powiadomień z pięcioma krótkimi wiadomościami tekstowymi bez załącznika
5	Zapis grupy powiadomień z pięcioma długimi wiadomościami tekstowymi bez załącznika
6	Zapis grupy powiadomień z pięcioma załącznikami bez wiadomości tekstowych
7	Pobranie grupy powiadomień z jedną krótką (<10 znaków) wiadomością tekstową bez załącznika
8	Pobranie grupy powiadomień z jedną długą (>200 znaków) wiadomością tekstową bez załącznika
9	Pobranie grupy powiadomień z jednym załącznikiem bez wiadomości tekstowej
10	Pobranie grupy powiadomień z pięcioma krótkimi wiadomościami tekstowymi bez załącznika
11	Pobranie grupy powiadomień z pięcioma długimi wiadomościami tekstowymi bez załącznika
12	Pobranie grupy powiadomień z pięcioma załącznikami bez wiadomości tekstowych
13	Usunięcie grupy powiadomień z jedną krótką (<10 znaków) wiadomością tekstową bez załącznika
14	Usunięcie grupy powiadomień z jedną długą (>200 znaków) wiadomością tekstową bez załącznika
15	Usunięcie grupy powiadomień z jednym załącznikiem bez wiadomości tekstowej
16	Usunięcie grupy powiadomień z pięcioma krótkimi wiadomościami tekstowymi bez załącznika
17	Usunięcie grupy powiadomień z pięcioma długimi wiadomościami tekstowymi bez załącznika
18	Usunięcie grupy powiadomień z pięcioma załącznikami bez wiadomości tekstowych

Na Rysunkach 2 oraz 3 przedstawiono wyniki dla obu aplikacji w postaci diagramów dla pierwszych sześciu scenariuszy, które dotyczyły operacji zapisu rekordów do bazy danych. Tutaj wyraźnie widać, jak w zależności od liczby przesyłanych wiadomości zmienia się czas wykonania operacji, im więcej przesyłanych danych, tym dłuższy czas ich zapisu. Ciekawe wyniki, na które należy zwrócić uwagę, są w przypadku scenariuszy o numerach 3 i 6, w których mierzono czas wykonania zapisu rekordów w postaci plików. W przypadku przesyłania pojedynczego pliku średni czas jego zapisu wynosi mniej niż 4 milisekundy, natomiast w przypadku przesyłania pięciu plików na raz ten czas wielokrotnie się zwiększa (ponad 150 milisekund). Natomiast ważny jest ten fakt, że dla wszystkich zdefiniowanych w artykule scenariuszy dotyczące operacji zapisu lepsze wyniki uzyskuje aplikacja zrealizowana w Spring Boot, w której użyto technologii Spring Data.



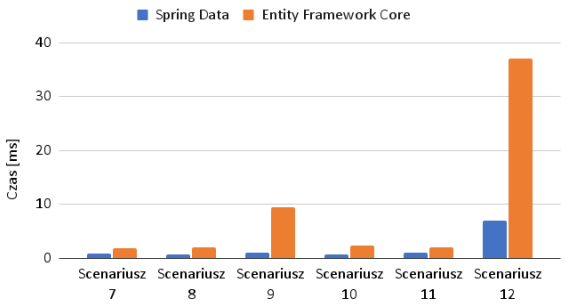
Rysunek 2: Średnie czasy pomiarów dla scenariuszy 1-4.



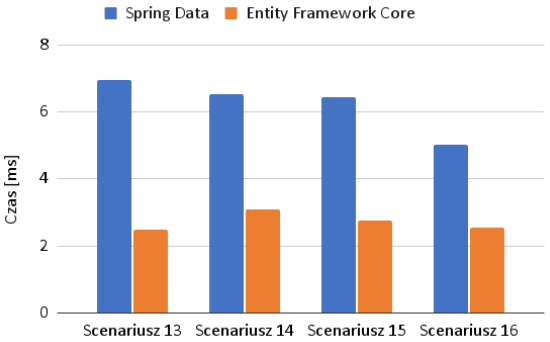
Rysunek 3: Średnie czasy pomiarów dla scenariuszy 5-6.

Kolejne 6 scenariuszy dotyczy operacji pobierania danych (Rysunek 4). Tutaj, tak samo, jak i w przypadku operacji zapisu, czas ich wykonania zależy od rozmiaru pobierających danych. W scenariuszach o numerach 9 oraz 12 dane są pobierane w postaci pliku. Widać, że czasy ich wykonania są znacznie dłuższe niż w przypadku innych scenariuszy, gdzie pobierane są dane w postaci wiadomości tekstowej. Jest to potwierdzeniem tego, że pliki są znacznie większe od samej treści tekstowej.

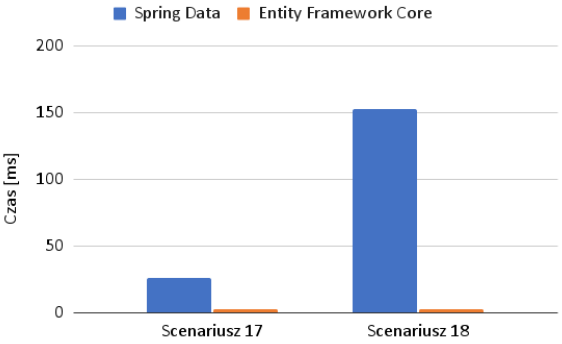
Ponadto, w przypadku technologii Entity Framework Core, operacje pobierania plików są wykonywane znacznie dłużej w porównaniu do technologii Spring Data. We wszystkich pozostałych scenariuszach, dotyczących pobierania danych, Spring Data też pokazuje lepsze rezultaty.



Rysunek 4: Średnie czasy pomiarów dla scenariuszy 7-12.



Rysunek 5: Średnie czasy pomiarów dla scenariuszy 13-16.



Rysunek 6: Średnie czasy pomiarów dla scenariuszy 17-18.

Ciekawe wyniki przedstawione są na dwóch ostatnich diagramach (Rysunek 5 oraz Rysunek 6). Prezentują one wyniki pomiarów dla scenariuszy, w których wykonywano operacje usuwania rekordów z bazy danych. W przypadku technologii Spring Data można zauważyć identyczną do poprzednich diagramów zależność czasu wykonania operacji od rozmiaru danych. Natomiast w przypadku technologii Entity Framework Core okazuje się, że przy zwiększeniu rozmiaru danych czas ten znacząco nie zmienia się, dla poszczególnych scenariuszy czas jest podobny, przy czym jest krótszy niż w przypadku technologii Spring Data. W tej części badań technologia Entity Framework Core uzyskuje lepsze wyniki pod kontem wydajnościowym.

5.2. Uwierzytelnianie oraz autoryzacja

Porównanie systemów autoryzacji, uwierzytelnienia oraz zarządzania użytkownikami zostało ocenione według następujących kryteriów:

- rozszerzalność narzędzia,
- wygoda pracy z tokenem JWT,
- współpraca z innymi narzędziami w obrębie wybranej technologii.

Do oceny rozszerzalności narzędzi Spring Security oraz Entity Framework Core wprowadzono 5 warunków:

- Czy dane narzędzie oferuje możliwość implementacji uwierzytelnienia i autoryzacji za pośrednictwem firm trzecich (Google lub Facebook)?
- Czy dane narzędzie pozwala na własną implementację mechanizmu hashowania haseł?
- Czy dane narzędzie oferuje możliwość przechowywania informacji o zalogowanym użytkowniku na różne sposoby (Cookies, JWT Token, baza danych)?
- Czy dane narzędzie pozwala na łatwy sposób rozbudowywania polityki autoryzacji, takie jak roli użytkownika?
- Czy pozwala na łatwą rozbudowę modelu użytkownika?

Za każdy spełniający warunek technologia uzyskiwała jeden punkt. W wyniku oceny wybranych technologii okazało się, że zarówno ASP.NET Core Identity, jak i Spring Security spełniają każdy z wyżej wymienionych warunków i dostają maksymalną liczbę punktów - 5. Wygodę pracy z tokenem JWT oceniono według skali od 1 do 5, gdzie:

- 1 - nie nadaje się do implementacji JWT,
- 2 - wymaga własnej implementacji JWT,
- 3 - pozwala na implementację JWT za pośrednictwem firm trzecich,
- 4 - oferuje własne dodatkowe narzędzia do implementacji oraz walidacji JWT,
- 5 - oferuje własne narzędzia do implementacji JWT oraz bierze na siebie realizację autoryzacji użytkownika za pomocą JWT.

Każdy z ocenianych frameworków oferuje własne narzędzia do pracy z JWT, ale biorąc pod uwagę to, że ASP.NET Core Identity posiada dodatkowe narzędzia, ułatwiające implementację uwierzytelnienia i autoryzacji za pomocą JWT, oceny tych frameworków są następujące: ASP.NET Core Identity uzyskuje 5 punktów, natomiast Spring Security – 4 punkty.

Ostatnie kryterium dotyczyło współpracy testowanych systemów do implementacji uwierzytelnienia oraz autoryzacji z innymi narzędziami w obrębie wybranej technologii. To kryterium zostało ocenione według skali od 1 do 5, gdzie:

- 1 - wymaga własnej implementacji integracji pomiędzy systemem uwierzytelnienia i autoryzacji oraz szkieletem aplikacji webowej,
- 2 - współpracuje ze szkieletem aplikacji webowej przy pomocy rozwiązań firm trzecich,
- 3 - współpracuje ze szkieletem aplikacji webowej za pomocą gotowych narzędzi,
- 4 - współpracuje ze szkieletem aplikacji webowej za pomocą gotowych narzędzi oraz z rozwiązaniem ORM za pośrednictwem firm trzecich lub za pomocą własnej implementacji takiej integracji,

- 5 - współpracuje ze szkieletem aplikacji webowej oraz z rozwiązaniem ORM za pomocą gotowych narzędzi.

Według tej skali Spring Security uzyskał 4 punkty, natomiast ASP.NET Core Identity uzyskał 5 punktów. W Tabeli 3 przedstawiono wyniki oceniania wybranych narzędzi.

Tabela 3: Wyniki oceniania framewroków uwierzytelnienia i autoryzacji Spring Security oraz ASP.NET Core Identity

Kryterium	Spring Security	ASP.NET Core Identity
Rozszerzalność narzędzia	5	5
Wygodność pracy z tokenem JWT	4	5
Współpraca z innymi narzędziami w obrębie wybranej technologii	4	5

6. Wnioski

W artykule przeprowadzono analizę porównawczą ekosystemów dwóch najbardziej popularnych szkieletów aplikacji ASP.NET Core oraz Spring Boot. Do przeprowadzenia badań stworzono dwie aplikacje testowe typu REST o identycznej funkcjonalności messengera webowego. Aplikacje testowe zostały stworzone w różnych stosach technologicznych. Pierwsza aplikacja napisana została w języku Java. Jako szkielet wybrano Spring Boot. Do komunikacji z bazą danych użyto narzędzia Spring Data, do realizacji uwierzytelnienia oraz autoryzacji użytkownika — Spring Security. Drugą aplikację testową napisano w języku C# za pomocą frameworku ASP.NET Core. Tutaj do komunikacji z bazą danych służy Entity Framework Core, natomiast za realizację mechanizmów uwierzytelnienia oraz autoryzacji odpowiada ASP.NET Core Identity.

Podczas analizy badawczej narzędzi do komunikacji z bazą danych brano pod uwagę czas wykonywania operacji CRUD. W celu obiektywnej oceny poszczególnych operacji zdefiniowano kilka scenariuszy. Po przetestowaniu każdego ustalono, że Spring Data w operacjach zapisu oraz odczytu danych jest szybszy od Enitity Framework Core. Natomiast Entity Framework Core jest szybszy przy usuwaniu rekordów. Uwzględniając to, że Spring Data w większości scenariuszach uzyskał lepsze rezultaty, można stwierdzić, że jest bardziej wydajnym narzędziem od Entity Framework. Teza *T1* została potwierdzona.

Systemy uwierzytelnienia oraz autoryzacji subiektywnie oceniono według następujących kryteriów: rozszerzalność narzędzia, wygoda pracy z tokenem JWT oraz współpraca z innymi narzędziami w obrębie wybranej technologii. Każdy ze szkieletów aplikacji dostał wysokie oceny i w całości spełnia wybrane kryteria. Warto zaznaczyć, że ASP.NET Core Identity jest bardziej zintegrowany w ekosystem szkieletu, w obrębie którego pracuje. ASP.NET Core Identity posiada dodatkowe mechanizmy do integracji z Entity Framework Core, a także dodatkowe już gotowe klasy do pracy z tokenem JWT. Z tego powodu ASP.NET Core Identity dostał wyższą ocenę w takich kryteriach jak wygoda pracy z tokenem JWT oraz współpraca z innymi narzędziami w obrębie wybranej technologii. Na podstawie tego można stwierdzić, że teza *T2* została potwierdzona ─ Asp.Net Core Identity jest bardziej elastycznym i wygodnym w użyciu narzędziem w porównywaniu do Spring Security.

Literatura

[1] Szkielet programistyczny Spring Boot, https://spring.io/projects/spring-boot , [26.01.2021].

[2] J. Ciliberti, ASP. NET Core Recipes: A Problem-Solution Approach, Apress, New York, 2017.

[3] Serwer bazodanowy PostgreSQL, https://www.postgresql.org/ , [22.10.2021].

[4] Narzędzie Spring Data, https://spring.io/projects/spring-boot , [22.10.2021].

[5] J. Lerman, R. Miller, Programming Entity Framework: Code First, O'Reilly Media, Sebastopol, 2011.

[6] M. Furtak, Bezpieczeństwo aplikacji internetowych, JCSI 3 (2017) 1-5.

[7] M. Jailia, A. Kumar, M. Agarwal and I. Sinha, Behavior of MVC (Model View Controller) based Web Application developed in PHP and .NET framework, 2016 International Conference on ICT in Business Industry & Government (ICTBIG) (2016) 1-5.

[8] M. M. Kizeweter, B. Pańczyk, Porównanie technologii tworzenia aplikacji internetowych JEE na przykładzie JavaServer Faces i Spring Boot, JCSI 3 (2017) 28-32.

[9] M. Kaluža, M. Kalanj, A comparison of back-end frameworks for web application development, Computer Science 7 (2019) 317-332.

[10] Operacje bazodanowe typu CRUD, https://www.codecademy.com/articles/what-is-crud , [19.10.2021].