POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Paweł Rzeńca
Nr albumu s21852

Porównanie wytwarzania oprogramowania internetowego z wykorzystaniem różnych technologii

Praca magisterska

Dr. inż. Mariusz Trzaska

Warszawa, lipiec, 2022

Streszczenie

Niniejsza praca magisterska dotyczy kwestii porównania wytwarzania oprogramowania internetowego w oparciu o różne technologie. Istnieje wiele odmiennych języków programowania, które związane są z rozmaitymi platformami programistycznymi oraz rozwiązaniami. W pracy zmierzono się z problemem kompleksowego spojrzenia na cechy oraz różnice i podobieństwa, którymi się one wyróżniają. W tym celu między innymi zaimplementowane zostały dwie wersje systemu zarządzania przedsiębiorstwem. Jedna oparta jest o język Java oraz platformę programistyczną Spring, a druga o PHP i Laravel. Dostarczają one nie tyle gotowego rozwiązania, które mogłoby konkurować z oprogramowaniem znajdującym się na rynku komercyjnym, co dają bazę do porównania wybranych technologii. Praca może posłużyć nie tylko jako podstawa i źródło danych na temat tych technologii, które zostały wzięte do analizy porównawczej, ale również dostarcza informacji pod jakim względem można dokonać porównania innych na podstawie przedstawionych kryteriów jakimi charakteryzuje się oprogramowanie.

Słowa kluczowe: oprogramowanie internetowe, porównanie technologii webowych, platformy programistyczne, języki programowania, Java, PHP, Spring, Laravel.

1. Wstęp

Istnieje wiele różnego rodzaju języków programowania, technologii oraz podejść do tworzenia kodu. Języki programowania różnią się między sobą wieloma cechami, takimi jak sposób ich używania czy zastosowanie. Jedne są używane do wytwarzania oprogramowania na różne urządzenia o odmiennych zastosowaniach, a inne są nakierowane na implementowanie tylko jednego rodzaju oprogramowania. Różne bywa podejście do sposobu tworzenia kodu i relacje jakie zachodzą pomiędzy jego poszczególnymi elementami. Rozbieżności występują także w sposobie wytwarzania źródeł programu oraz ich przetwarzaniu. Istnieją języki, które określane są jako interpretowane, kompilowane oraz hybrydowe. Na początku niniejsza praca dyplomowa skupiona jest na wprowadzeniu w wcześniej wspomnianą tematykę, tak aby dostarczyć informacji, które pozwolą na zrozumienie i późniejsze przenalizowanie podobieństw i różnic, jakie mogą występować pomiędzy poszczególnymi rozwiązaniami. Jako iż jest ona skoncentrowana na procesie wytwarzania oprogramowania internetowego, znalazło się w niej także miejsce poświęcone temu tematowi. W związku z tym, że do porównania zostały wybrane języki Java oraz PHP to w następnych częściach zostały przedstawione kluczowe informacje dotyczące tych technologii oraz ich charakterystyka. Przyjrzano się również sposobowi, w którym tworzony jest dla nich kod źródłowy oraz temu jak jest on później obsługiwany. W ramach niniejszej pracy zbadano także kwestie wydajności oraz użycia zasobów.

We współczesnym świecie programowania powszechnie używanymi narzędziami, które wspierają proces tworzenia aplikacji są platformy programistyczne (ang. *frameworks*). Poszczególne rozwiązania różnią się między sobą możliwościami, sposobem działania, dostarczanymi komponentami, procedurą tworzenia nowego kodu, metodą dodawania nowych modułów czy popularnością. Inna może być również wydajność poszczególnych rozwiązań, podejście do bezpieczeństwa, w tym systemu autoryzacji istotnego z punktu widzenia oprogramowania internetowego. W celu dostarczenia dodatkowych danych do analizy porównawczej technologii używanych przy wytwarzaniu oprogramowania internetowego, w ramach niniejszej pracy zaimplementowane zostały dwie aplikacje internetowe. Aby oprogramowanie to odpowiadało nowoczesnym rozwiązaniom i współczesnym potrzebom, autor postanowił wykorzystać do ich stworzenia platformy programistyczne, w postaci Springa dla Javy i Laravela dla PHP. Przy wyborze kierowano się między innymi popularnością wybranych technologii, gdzie dane na ten temat również zostały przedstawione i uległy porównaniu.

Niniejsza praca dyplomowa przybliża wymienione wyżej kwestie i pokazuje czym mogą się różnić poszczególne technologie. Zawiera analizę, w której porównane zostały odmienne rozwiązania dla oprogramowania internetowego. Ukazane są podobieństwa oraz różnice występujące w poszczególnych technologiach i podejściach. Może ona również służyć jako cenne źródło informacji przy typowaniu odpowiedniego narzędzia dla wybranych wymagań i określonego celu.

2. Wstęp do programowania

Pierwszy podrozdział w tej części pracy ma na celu przedstawienie istoty programowania oraz krótkie omówienie różnych rodzajów oprogramowania. Ma to znaczenie w odniesieniu do późniejszych rozdziałów, gdzie zostały przedstawione konkretne możliwości w językach Java oraz PHP, które wpływają na ich zastosowanie. W kolejnym podrozdziale znajduje się charakterystyka języków programowania, gdzie występują rozbieżności w podejściu do tworzenia w nich kodu oraz relacji jakie zachodzą pomiędzy ich poszczególnymi elementami. Niniejsza praca skupia się na oprogramowaniu internetowym, zatem ostatni podrozdział poświęcony jest na przybliżenie początków internetu, kluczowych mechanizmów, które nim rządzą oraz przedstawieniu istotnych technologii z jego punktu widzenia.

2.1. Wprowadzenie do programowania

Programowanie w dzisiejszym świecie ze społecznego oraz gospodarczego punktu widzenia zajmuje i pełni bardzo istotną rolę. Większość ludzi posiada takie urządzenia jak komputery klasy PC, laptopy czy mniejsze urządzenia w postaci smartfonów i tabletów. Codzienne czynności takie jak zakupy, sprawdzanie ostatnich wiadomości ze świata czy prognozy pogody dokonujemy często, właśnie przy ich użyciu. Wiele instytucji naukowych oraz przedsiębiorstw funkcjonuje, opierając swoją działalność na komputerach. To co sprawia, że urządzenia te są interesujące z punktu widzenia użytkowników, to znajdujące się na nich aplikacje. Powstają one w wyniku procesu programowania.

Istnieje wiele rodzajów programów, uwzględniając ich przeznaczenie możemy wyróżnić między innymi:

- **Oprogramowanie okienkowe (ang. *desktop software*)** – Jest to oprogramowania działające na systemie operacyjnym w postaci odrębnych, prostokątnych okien. Przykładem takiego programu jest dobrze znany kalkulator, dołączony do systemu operacyjnego Windows.
- **Oprogramowanie wbudowane (ang. *embedded software*)** – Ta grupa programów, zawiera wszelkie oprogramowanie, które działa bezpośrednio na urządzeniu i jest z nim nierozerwalnie związana, sterując jego pracą. Przykładem takiego oprogramowania jest to, które znajduje się w pralce.
- **Oprogramowanie webowe (ang. *web software*)** – Przez ten rodzaj programów rozumiemy wszelkie oprogramowanie wykorzystujące możliwości Internetu. Przykładem są różnego typu strony internetowe.
- **Oprogramowanie na urządzenia mobilne (ang. *mobile software*)** – Oprogramowanie przeznaczone na urządzenia mobilne takie jak telefony i tablety z systemem Android czy iOS.
- **Programy wykorzystujące linie komend systemu operacyjnego (ang. *command-line programs*)** – Są to programy w postaci skryptów, wykorzystujące interfejs linii komend systemów operacyjnych takich jak Unix czy Windows. Są one często wykorzystywane do automatyzacji różnego typu zadań.

Jak podaje Alex Allain w swojej książce ‘C++ Przewodnik dla początkujących’ programowanie to [1]: ‘*Programowanie to proces pisania instrukcji w taki sposób, który umożliwi komputerowi ich zrozumienie i wykonanie. Same instrukcje nazywane są kodem źródłowym.*’.

Programiści zajmują się pisaniem kodu źródłowego, w językach przystępnych dla człowieka, który jednak nie jest jeszcze na tym etapie czytelny dla komputerów. Aby stał się on zrozumiały dla komputera, musi ulec procesowi kompilacji lub interpretacji, które zostaną opisane w niniejszej pracy magisterskiej.

2.2. Charakterystyka języków programowania

Rozpatrując języki programowania, możemy dokonać ich podziału na te, które określane są jako języki niskiego poziomu i te, które są określane jako języki wysokiego poziomu. Pierwsze, to takie które są blisko powiązane z językiem maszynowym. Przykładem jest język Assembler, którego instrukcje poddawane są translacji w dokładnie jedną instrukcję binarną języka maszynowego. Języki wysokiego poziomu ukierunkowane są bardziej na programistę niż bliskość w stosunku do maszyny. Używają one zarówno notacji matematycznej i naturalnej. Najczęściej instrukcja kodu napisanego w języku wysokopoziomowym jest tłumaczona na wiele instrukcji kodu maszynowego [2]. Przykładem takich języków są C# oraz JavaScript. Należy zaznaczyć, iż podział ten jest jednak płynny. Za język niskiego poziomu przez niektóre osoby może być uznawany dopiero Asembler, podczas gdy inni mogą uważać, że już języki C i C++ są wystarczająco blisko sprzętu, żeby zakwalifikować je jako niskopoziomowe.

W językach programowania kod może zostać napisany w pewnego rodzaju konwencji, stylu programowania, który nazywany jest paradygmatem [3]. Określa on przepływ informacji oraz sterowania w kodzie. Podczas jego pisania w danym języku, programista może często wybrać jeden z nich. Dobrą praktyką jest pisanie jednego programu, stosując pojedynczy paradygmat. Są języki takie jak PHP, które pozwalają pisać kod przykładowo w oparciu o paradygmat obiektowy lub funkcyjny. Istnieją również takie języki, jak C, gdzie język ten nie został zaprojektowany w celu pisania kodu w oparciu o obiektowy paradygmat, a dopiero język C++, który rozszerza język C między innymi o klasy, jest do tego przystosowany.

Możemy wyróżnić między innymi następujące paradygmaty programowania:

- **Programowanie funkcyjne** – Podchodzi do wszystkich podprogramów jak do funkcji, w sensie matematycznym, co oznacza, że funkcje takie przyjmują argumenty i zwracają pojedynczą wartość. Co istotne, moment w którym wywoływana jest funkcja nie ma tutaj znaczenia, ważne są za to dane wejściowe [4].

- **Programowanie imperatywne** – Jest to pierwsze podejście do programowania. Polega na takim podejściu do programu, w którym traktujemy go jako zestaw instrukcji dla maszyny. Wykonywane po kolei sekwencje takich instrukcji zmieniają stan komputera, przez co rozumiemy dane przetrzymywane w pamięci i związane z procesorem. Podejście to cechuje się bliskością w stosunku do komputera [5].

- **Programowanie obiektowe** – Wyznacznikiem programowania zorientowanego obiektowo są obiekty i klasy. Przez klasy rozumiemy pewnego rodzaju formy, na podstawie których powstają konkretne egzemplarze, nazywane obiektami. Obiekty cechują się atrybutami, których wartości mogą być różne dla poszczególnych egzemplarzy danej klasy oraz metodami, które określają funkcjonalność jaką mogą wykonywać. Zachodzą również pomiędzy nimi różnego rodzaju relacje. U podstaw paradygmatu obiektowego leżą cztery fundamenty:
    - hermetyzacja (ang. *encapsulation*) – hermetyzacja, nazywana jest również kapsułkowaniem. Polega ona na zamknięciu właściwości wewnątrz klasy przez uczynienie ich prywatnymi i wystawieniu interfejsów, które zapewniają do nich dostęp. Zastosowanie hermetyzacji pozwala na ochronę przed niezamierzonymi zmianami. Hermetyzacje osiąga się przykładowo dzięki zastosowaniu specjalnych metod nazywanych getterami i setterami, które służą do pobierania i ustawiania wartości właściwości (atrybutów) klasy.
    - dziedziczenie (ang. *inheritance*) – polega na tworzeniu klas w których zdefiniowane są atrybuty i funkcjonalność, które określają później tworzone obiekty. Takie właściwości mogą zostać przekazane przez dziedziczenie do innej klasy, co pozwala między innymi na pisanie mniejszej ilości kodu. Dzieje się tak ponieważ klasa, która dziedziczy przejmuje atrybuty i funkcjonalność klasy bazowej.
    - polimorfizm (ang. *polymorphism*) – zwany jest również wielopostaciowością. Dzięki zastosowaniu polimorfizmu kolekcje, mogą przyjmować obiekty różnych typów. Wielopostaciowość osiągana jest również przez dziedziczenie, gdzie klasy dziedziczące

posiadają własne implementacje metod z klasy bazowej i realizują ich funkcjonalność na własne potrzeby, w zależności od konieczności [6].
    - abstrakcja (ang. *abstraction*) – polega na ukrywaniu szczegółów implementacji i pokazywaniu jedynie ogólnych danych. Abstrakcja w programowaniu obiektowym realizowana jest przez zastosowanie klas abstrakcyjnych oraz interfejsów. Programista deklaruje w nich jedynie sygnatury metod, które implementowane są w klasach je implementujących [7].

- **Programowanie logiczne** – Program oparty o ten rodzaj programowania zawiera aksjomaty oraz cel. Te pierwsze wykorzystywane są do tego, aby udowodnić prawdziwość celu. Kod napisany podług paradygmatu logicznego odpowiada w pewien sposób mechanizmowi potwierdzania celu na podstawie przesłanek w postaci aksjomatów [8].

Istnieją również inne podejścia takie jak: strukturalne, proceduralne czy deklaratywne.

We współczesnych czasach jednym z wiodących paradygmatów programowania jest ten zorientowany obiektowo. W rozdziale poświęconym praktycznemu projektowi to właśnie w tej konwencji zostały zaimplementowane aplikacje napisane w języku Java oraz PHP, które są jednocześnie językami wysokiego poziomu.

2.3. Programowanie webowe

Początki Internetu sięgają późnych lat sześćdziesiątych dwudziestego wieku. Znajdują źródło w rządowej Agencji Zaawansowanych Projektów Badawczych w Departamencie Obrony Stanów Zjednoczonych. (ARPA), gdzie postawiono spróbować połączyć większą ilość komputerów znajdujących się w różnych miejscach, w celu uzyskania dostępu do zasobów w postaci danych oraz programów. Projekt ten nosił nazwę ARPANET od nazwy wspomnianej agencji oraz słowa *network*, oznaczającego z języka angielskiego sieć. Początkowo sieć ta łączyła maszyny znajdujące się w Cambridge, Massachusetts oraz w Kalifornii. Wraz z biegiem czasu, coraz większa ilość instytucji dołączała do tego projektu i tak w latach siedemdziesiątych dwudziestego wieku, sięgała już kilkaset dostępnych stron. Podczas ówczesnych prac ARPA powstał model protokołu komunikacyjnego TCP/IP, który wykorzystywany jest do dzisiaj. W latach osiemdziesiątych w agencji została podjęta decyzja o zaprzestaniu rozwijania projektu ARPANET, jako iż jego pierwotny cel został osiągnięty – połączono ze sobą maszyny, które były w stanie wymieniać między sobą informacje. W międzyczasie sieć zdążyła ewoluować do tysięcy komputerów na całym świecie. Jako jej użytkowników można było wymienić wojsko, firmy technologiczne oraz jednostki akademickie. Następnie amerykańska, rządowa agencja poświęcona nauce (National Science Foundation - NSF) przejęła istniejącą infrastrukturę w postaci linii telefonicznych wysokich szybkości, które zapewniły komunikację pomiędzy komputerami z wykorzystaniem protokołu TCP/IP. Wraz z upływem czasu prywatni dostawcy, rozpoczęli dodawanie do sieci domowych komputerów i w taki sposób sieć znana jako ARPANET przeobraziła się w Internet znany nam z dzisiejszej formy [9].

Jedną z najważniejszych gałęzi programowania jest właśnie programowanie webowe, czyli takie w wyniku którego powstają strony i aplikacje internetowe. Szacuje się, że światowy rynek usług internetowych w roku 2020 warty był 450,1 miliarda USD. Prognozowany jest dalszy wzrost wartości tego sektora do 632,4 miliarda USD w roku 2027 [10].

We wcześniejszej części niniejszej pracy magisterskiej został opisany proces formowania się Internetu, należałoby również wspomnieć co oznacza pojęcie ‘WWW’ i w jaki sposób różni się od Internetu [11]:

- **WWW (ang. *World Wide Web*)** – Przez WWW rozumiemy ogół wszystkich witryn internetowych, do których istnieje dostęp dzięki ich adresom oraz Internetowi. Sieć ta została opracowana w CERN (Europejskiej Organizacji Badań Jądrowych) przez Timothy Bernersa-Lee oraz Roberta Calliou w latach 1989-1990. Rok 1993, jest tym w którym efekt tej pracy

został udostępniony za darmo do użytku publicznego przez wszystkich zainteresowanych. W rezultacie możemy dzisiaj mówić o miliardach stron internetowych, na których znajdują się różnego rodzaju treści. Z pojęciem WWW wiążą się inne pojęcia HTTP, SMTP, FTP oraz URL, gdzie pierwsze trzy to przykłady warstwy aplikacji protokołu TCP/IP.

- **Internet** – Jako Internet rozumiemy sieć wzajemnie ze sobą połączonych urządzeń, czyli infrastrukturę. Sieć ta może być wykorzystywana do wyświetlania stron internetowych, ale może również być używana do takich czynności jak wymiana maili czy plików. Z pojęciem Internetu wiążą się pojęcie IP, które jest przykładem warstwy sieciowej protokołu TCP/IP oraz TCP i UDP, które są przykładami warstwy transportowej protokołu TCP/IP.

Poniżej znajduje się opis pojęć wspomnianych we wcześniejszym akapicie, a które są nierozerwalnie związane z koncepcją WWW oraz Internetu:
HTTP (ang. *HyperText transfer protocol*) – Jest to protokół leżący u podstaw sieci WWW, wykorzystywany na poziomie aplikacji do wymiany danych hipertekstowych, czyli takich, które zawierają hiperłącza. Należy do grupy protokołów bezstanowych, co oznacza, że następujące po sobie zapytania nie przetrzymują stanu - nie pamiętają poprzednich zapytań. Wykorzystywany jest w nim dwukierunkowy kanał komunikacji, gdzie można wyróżnić klienta wysyłającego zapytanie do serwera oraz serwer wysyłający odpowiedź na zapytanie klienta. Wiadomość HTTP składa się z dwóch części: z nagłówka, który jest obowiązkowy i zawiera informacje kontrolne oraz ciała wiadomości, która zawiera dodatkowe dane. Ciało wiadomości jest opcjonalne [12]. Protokół ten nie jest szyfrowany. Do różnych zadań wykorzystywane są inne metody HTTP, co zostało przedstawione w Tabeli 1 [13]:

Tabela 1. Spis metod HTTP wraz z ich opisem. **Źródło:** [13].

Metoda HTTP	**Opis**
Metoda GET	Służy do pobrania zasobu, określonego w żądaniu
Metoda POST	Przeznaczona jest do przesyłania obiektu
Metoda PUT	Wykorzystywana jest do zamiany obecnej encji, nową, która zdefiniowana jest w żądaniu, czyli aktualizacji
Metoda HEAD	Jest podobna do metody GET z tą różnicą, że nie zawiera ciała odpowiedzi
Metoda DELETE	Celem tej metody jest wykasowanie zasobu
Metoda TRACE	Służy do testowania kanału
Metoda OPTIONS	Odpowiada za informacje dotyczące opcji komunikacji kanału
Metoda CONNECT	Dzięki tej metodzie możliwe jest utworzenie tunelu z serwerem, rozpoznawanym przez zasób

Odpowiedzi od serwera zawierają numer, określający status odpowiedzi HTTP. W Tabeli 2 znajdują się podstawowe statusy wraz z ich znaczeniem [14]:

Tabela 2. Spis kodów odpowiedzi HTTP wraz z ich opisem. **Źródło**: [14].

Kod odpowiedzi HTTP	**Opis**
200	OK
400	Niepoprawny argument lub błędne żądanie
401	Brak autoryzacji dostępu do zasobu
402	Dostęp zabroniony
404	Nie odnaleziono zasobu
500	Wewnętrzny błąd serwera
503	Serwis jest obecnie nieosiągalny. Spróbuj ponownie później
504	Przekroczono czas bramy na odpowiedź na żądanie

Istnieje również wersja protokołu HTTP, która jest szyfrowana i nazywa się HTTPS (ang. *HyperText transfer protocol secure*), w którym wykorzystywany jest protokół SSL/TLS. Ta odmiana zapewnia większe bezpieczeństwo i jej zastosowanie przeciwdziała przechwytywaniu oraz modyfikacji danych, które są przesyłane.

SMTP (ang. *Simple mail transfer protocol*) – Jest to protokół komunikacyjny wykorzystywany do przesyłania i odbierania wiadomości e-mail przy pomocy Internetu.

FTP (ang. *File transfer protocol*) – Protokół ten używany jest do transferu plików i należy do grupy protokołów klient-serwer. Żeby wymiana plików była możliwa musi zostać ustanowione połączenie między komputerem lokalnym, a serwerem FTP. Po jego ustanowieniu możliwa jest dwukierunkowa wymiana plików oraz dokonywanie zmian na zdalnym serwerze. FTP wspiera transfer plików w ciągłości, co oznacza, że przy przerwach w połączeniu nie istnieje możliwość odzyskania danych. Domyślnie w komunikacji przy pomocy FTP wartości w postaci nazwy użytkownika i hasła są nie szyfrowane, co stanowić może problem związany z bezpieczeństwem. W związku z tym powstała odmiana FTP w postaci protokołu FTPS, czyli FTP-SSL lub FTP Secure. Wersja ta zabezpiecza wymianę informacji dotyczących autentykacji przy pomocy certyfikatu SSL. Wymaga ona, żeby serwer został skonfigurowany z użyciem certyfikatu SSL, tak aby była możliwość jego wspierania [15].

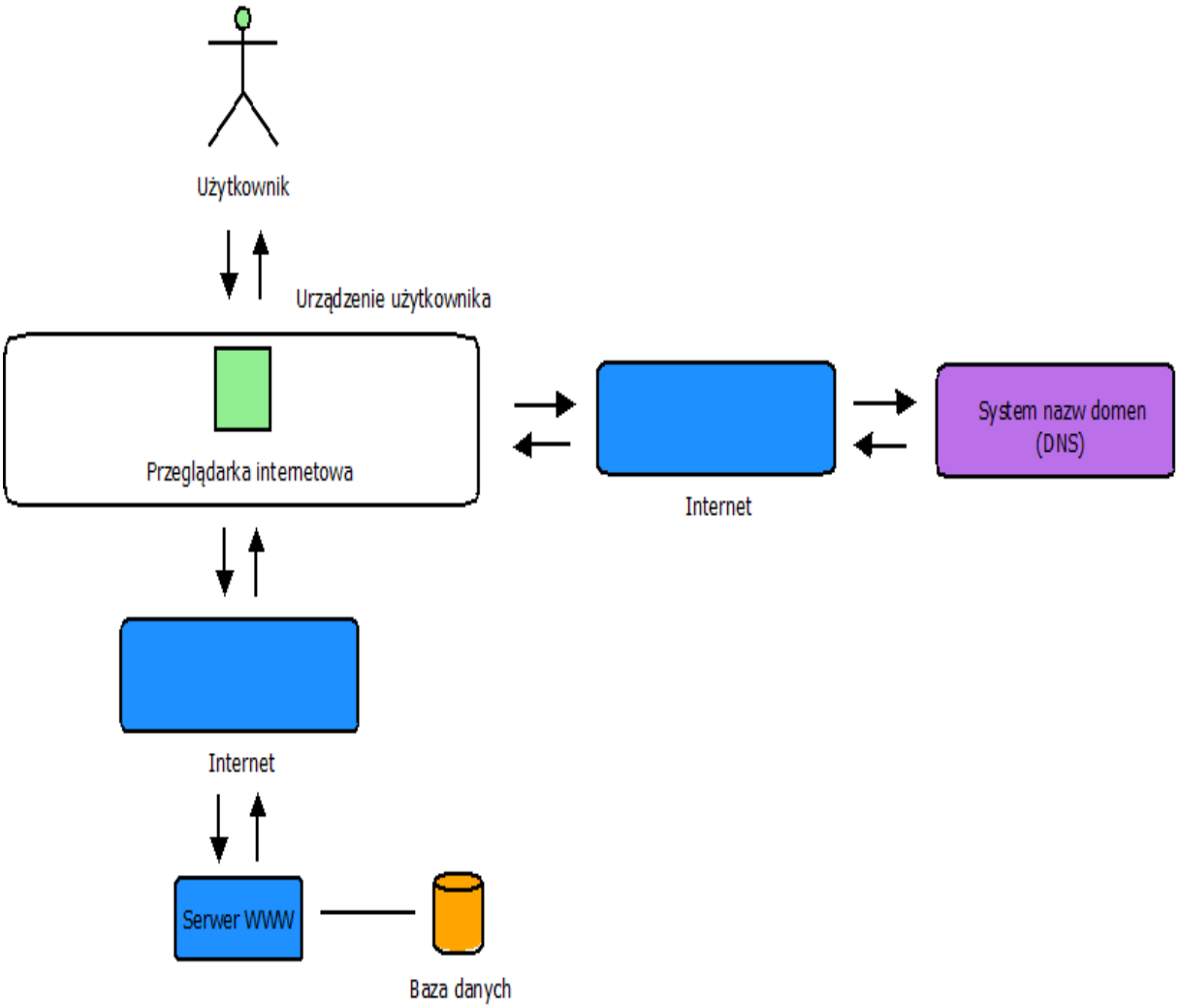
URL (ang. *Uniform resource locator*) – URL jest to jednolity standard lokalizowania zasobów w sieci Internet. Dzięki niemu możliwy jest dostęp do plików znajdujących się na serwerach. URL zapisywany jest w określonej konwencji:
schemat://nazwaHosta.domena[:port]/ścieżka/nazwaPliku
Schematem, czyli rodzajem zasobu może być przykładowo [16]:

- http lub https – dla pliku znajdującego się w sieci WWW,
- ftp – dla pliku znajdującego się na serwerze FTP,
- file – plik na lokalnym systemie,
- gopher – plik na serwerze Gopher.

UDP (ang. *User Datagram protocol*) [17] – Jest protokołem warstwy transportowej TCP/IP. Informacje do przesłania są w nim najpierw zapisywane do gniazda UDP, które następnie kapsułkowane są w ramach datagramów UDP, a te finalnie opakowywane są w datagramy IP i przesyłane w miejsce docelowe. To co charakteryzuje datagram jest jego długość, która przesyłana jest do celu, wspólnie z danymi. Protokół UDP nie gwarantuje, że przesłane informacje faktycznie dotrą do celu. Dodatkowo nie jest zapewnione zachowanie kolejności wysyłanych danych. Protokół UDP nie jest również niezawodny w przypadku wystąpienia błędu w sumie kontrolnej lub stabilności połączenia, ponieważ datagram w takim przypadku nie zostanie automatycznie wysłany ponownie. W przypadku protokołu UDP nie jest nawiązywane połączenie Klient-Serwer.

TCP (ang. *Transmission control protocol*) [17] – TCP tak samo jako UDP jest protokołem na poziomie warstwy transportowej. Występuje w nim połączenie, które nawiązywane jest pomiędzy klientem, a serwerem. Opiera się on na przesyłaniu danych w formie strumienia bajtów, które nie są niczym ograniczone. Protokół TCP dostarcza niezawodności, w przeciwieństwie do protokołu UDP, ponieważ po tym jak zostaną przesłane dane w kierunku serwera to oczekiwana jest odpowiedź zwrotna, zawierająca potwierdzenie. W przypadku, gdy takiego potwierdzenia nie będzie, w protokole TCP następuje przez pewien czas próba ponownego wysłania danych.

Przykładowy schemat działania oprogramowania webowego widoczny na Rysunku 1 przedstawia się następująco [18]:
- Użytkownik, korzysta z urządzenia posiadającego przeglądarkę internetową i wpisuje adres strony internetowej ‘https://www.pja.edu.pl/’.
↓
- Przeglądarka z wykorzystaniem Internetu łączy się z rozporoszonym systemem DNS. DNS (ang. *Domain Name System*) jest to system odpowiedzialny za to, żeby odpowiadać na zapytanie w formie nazwy domeny, adresem IP w postaci nie więcej niż 12 cyfr. Wszystkie urządzenia, które podłączone są do Internetu posiadają taki właśnie adres IP.
↓
- Przeglądarka po otrzymaniu odpowiedzi od DNS w postaci IP, wysyła zapytanie do serwera WWW.
↓
- Serwer WWW przetwarza żądanie. Wykonywany jest kod po stronie serwerowej, może również zachodzić na tym etapie proces łączenia się z innym oprogramowaniem, przykładowo w ramach obsługi zapytań do bazy danych. Następnie zwracana jest odpowiedź przez Internet do przeglądarki użytkownika.
↓
- Przeglądarka wyświetla użytkownikowi treść strony internetowej.



Rysunek 1. Schemat działania oprogramowania webowego z uwzględnieniem strony klienckiej oraz serwerowej. **Źródło**: Opracowanie własne.

Jeśli chodzi o obszary programowania webowego to możemy dokonać następującego podziału:
Strona kliencka (ang. *Frontend*) – Jest to ta część, którą użytkownik widzi i z którą wchodzi w bezpośrednią interakcję. Odpowiedzialna jest za wyświetlanie strony, za przesyłanie danych do strony serwerowej oraz odbieraniu od niej danych.

Do najistotniejszych technologii frontedowych zaliczamy:

- **HTML** – Rozwinięciem skrótu HTML jest *Hyper text markup language*, czyli hipertekstowy język znaczników. Jest on używany do tworzenia dokumentów, przeznaczonych do wyświetlania w przeglądarkach internetowych. Przy jego pomocy budowana jest struktura strony wraz z jej treścią. Tworzy szkielet witryny internetowej. Bazuje na różnego rodzaju tagach, wewnątrz których umieszcza się zawartość, która będzie później widoczna na stronie internetowej. Jedną z głównych zasad przy tworzeniu kodu HTML jest to, że każdy otwarty tag, musi również zostać zamknięty. Sposób zachowania tagów oraz jego cechy mogą zostać określone poprzez dodatkowe atrybuty.
- **CSS** – Jest to akronim od słów pochodzących z języka angielskiego *Cascading Style Sheets*, co oznacza kaskadowe arkusze stylów. CSS jest językiem, przy pomocy którego stylowane są strony internetowe. Dzięki jego zastosowaniu możliwe jest określenie poszczególnych cech elementów strony, takich jak: krój czcionki, jej wielkość oraz kolor, umiejscowienie poszczególnych komponentów znajdujących się w oknie przeglądarki oraz wiele innych właściwości.
- **JavaScript** – Język programowania, zaliczany do grona skryptowych. Został on wypuszczony w roku 1995 i był przeznaczony do oprogramowywania przeglądarki internetowej Netscape Navigator. Dopiero później dołączyły do niej inne przeglądarki. Za język JavaScript odpowiedzialne jest przedsiębiorstwo ECMA International, które zajmuje się standardem ECMA Script, czyli wzorcem opisującym JavaScript w takiej formie, aby był on obsługiwany przez różnego rodzaju przeglądarki w ten sam sposób [19]. Jest on językiem wysokiego poziomu, o dynamicznym typowaniu. Podlega interpretowaniu, jednak możliwe jest jego kompilowanie przy pomocy metody kompilacji '*Just-in-time*'. Wspiera on między innymi obiektowy paradygmat programowania. Język ten jest obecnie najpopularniejszym językiem do tworzenia kodu po stronie frontendowej witryn internetowych. Jego obecnie użycie po stronie klienckiej wynosi 97,8% wszystkich stron internetowych [20]. Dzięki zastosowaniu JavaScriptu możliwe jest dynamiczne wyświetlanie stron internetowych, wysyłanie informacji i żądań HTTP do strony serwerowej czy tworzenie interaktywnych elementów witryny. Przy jego pomocy możliwe jest również pisanie kodu strony serwerowej, przykładowo przy pomocy frameworka Node JS.

Strona serwerowa (ang. *Backend*) – To z kolei ta część, która odpowiedzialna jest za operacje po stronie serwerowej. Rozumiemy przez to między innymi akcje, określane jako CRUD, co jest akronimem od słów pochodzących z języka angielskiego *create, read, update, delete*, czyli stwórz, odczytaj, zaktualizuj oraz wykasuj. Odnosi się to do operacji na encjach, czyli rekordach najczęściej ulegających persystencji, to znaczy utrwaleniu w bazie danych. Po tej stronie zachodzą również wszelkie operacje w postaci obliczeń i różnego rodzaju logika biznesowa. Jeśli chodzi o stronę bazodanową to możemy dokonać podziału na dwie kategorie: takie bazy danych, które korzystają z relacyjnego modelu bazy danych i języka SQL oraz takie które z nich nie korzystają i są bazami określanymi jako NoSQL. Przykładowymi językami, które znajdują zastosowanie po stronie backendowej są Java oraz PHP. Zostały one wykorzystane do stworzenia kodu dla strony serwerowej w ramach niniejszej pracy magisterskiej i opisane zostały w późniejszych rozdziałach.

Podstawowy szkielet strony internetowej może wyglądać tak jak widoczne jest w Kodzie 1:

Kod 1. Przykładowy szkielet strony. **Źródło**: Opracowanie własne.

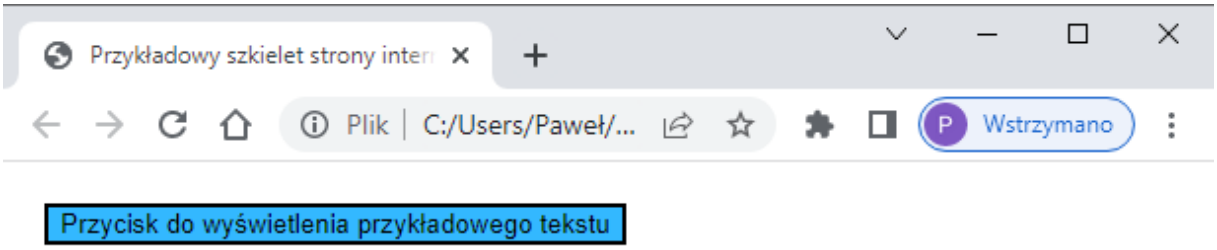
```
<!DOCTYPE html> <!-- 1) -->

<html> <!-- 2) -->
<head> <!-- 3) -->
    <meta name="author" content="Paweł Rzeńca">
    <title>Przykładowy szkielet strony internetowej</title>

    <style type="text/css"> <!-- 4) -->
        #displayTextButton {
            margin: 1em;
            background-color: #33b8ff;
        }

        #containerForText {
            margin: 1em;
            color: #ff0000;
            font-style: italic;
            font-weight: bold;
        }
    </style> <!-- 4) -->
</head> <!-- 3) -->

<body> <!-- 5) -->
    <button id="displayTextButton">
        Przycisk do wyświetlenia przykładowego tekstu
    </button>
    <div id="containerForText"> </div>

    <script> <!-- 6) -->
      var displayTextButton = document.getElementById("displayTextButton");
      var containerForText = document.getElementById("containerForText");

      displayTextButton.onclick = function () {
          containerForText.innerHTML = "Przykładowy tekst";
      };
    </script> <!-- 6) -->
</body> <!-- 5) -->

<html> <!-- 2) -->
```

Objaśnienie Kodu 1, zawierającego przykładowy szkielet strony:

1) Znacznik ‘`<!DOCTYPE html>`’ określa, że niniejszy plik to dokument HTML. Stanowi to informacje dla przeglądarki internetowej z jakim typem treści ma do czynienia.
2) Otwarcie i zamknięcie tagu ‘`<html>`’ mówi o początku oraz końcu dokumentu HTML.
3) Sekcja nagłówkowa ‘`<head>`’, w której zawarte są różnego rodzaju informacje na temat danej witryny internetowej.

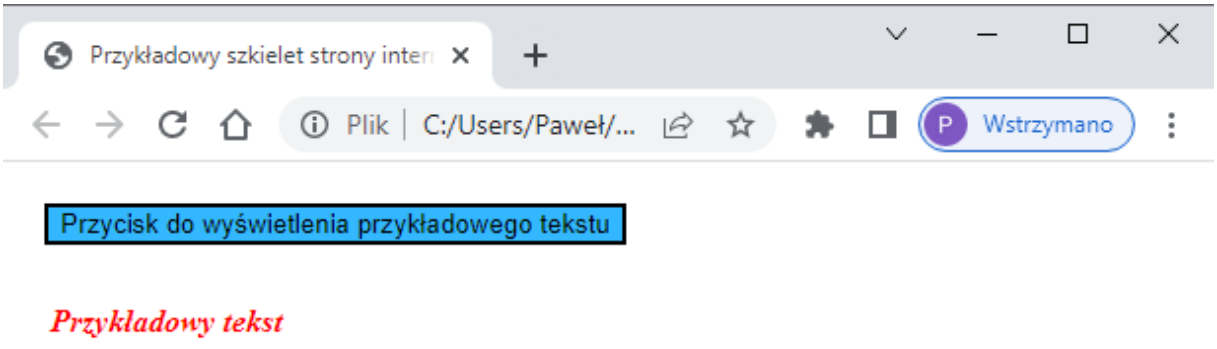
4) Blok kodu określony przez ‘<style type="text/css">’, w którym zdefiniowane zostały style CSS. Jest to jeden ze sposobów ich zamieszczania. Innymi są takie, gdzie dodawane są one wewnątrz znaczników HTML. Kolejnym jest wydzielenie ich do osobnego pliku, a następnie podlinkowanie w sekcji ‘<head>’ przy pomocy tagu ‘<link>’. Ten ostatni sposób pozwala wielu podstronom współdzielenie tych samych styli, co ogranicza ilość koniecznego kodu wymaganego do ostylowania witryny. W powyższym kodzie określony został kolor tła przycisku, marginesy oraz style określające wygląd tekstu.
5) Pomiędzy tagiem otwierającym i zamykającym ‘<body>’ zawarte jest ciało strony internetowej, czyli jej treść. W powyższym przykładzie treścią jest przycisk oraz kontener ‘ ’, przeznaczony w tym przypadku do wyświetlania tekstu.
6) Znacznik ‘<script>’ wewnątrz, którego zawarty został kod JavaScript. Podobnie jak w przypadku styli CSS istnieje możliwość wydzielenia kodu do innego pliku, a następnie umieszczenie do niego linku na wielu podstronach.

Rezultat po uruchomieniu Kodu 1 w oknie przeglądarki internetowej, prezentuje się tak jak na Rysunku 2:



Rysunek 2. Efekt po uruchomieniu kodu szkieletu strony internetowej. **Źródło**: Opracowanie własne.

Natomiast po kliknięciu na przycisk efekt widoczny jest na Rysunku 3:



Rysunek 3. Efekt po uruchomieniu kodu szkieletu strony internetowej oraz kliknięciu na przycisk. **Źródło**: Opracowanie własne.

3. Języki kompilowane, interpretowane oraz hybrydowe

Kod źródłowy napisany w języku programowania jest niezrozumiały dla komputera. Aby mógł on zostać wykonany musi ulec kompilacji albo interpretacji lub jednemu i drugiemu procesowi. Wyróżniamy więc podział na trzy rodzaje języków pod względem procesów, którym poddawany jest kod, aby został zrozumiany przez komputer:

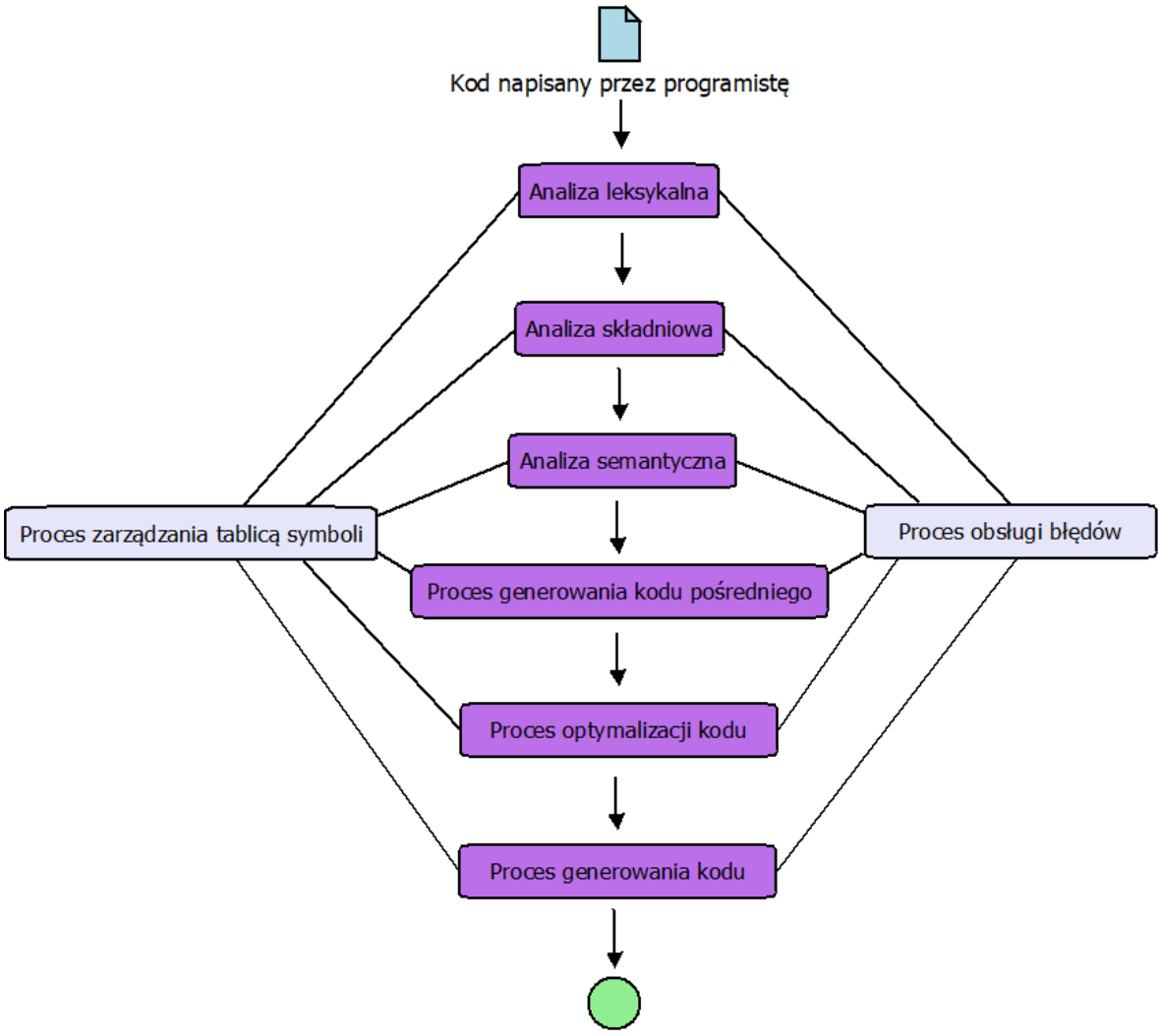
- języki wykorzystujące proces kompilacji,
- języki wykorzystujące proces interpretacji,
- języki hybrydowe, które wykorzystują zarówno proces kompilacji jak i interpretacji.

W następnych podrozdziałach znajduje się opis i charakterystyka, każdego z nich.

3.1. Języki kompilowane i proces kompilacji

Kompilacja jest to proces wykonywany przez kompilator, który polega na tłumaczeniu kodu napisanego w języku wysokiego poziomu na kod maszynowy [21], który rozumiany jest przez komputer. Efektem tego procesu może być również kod pośredni, co zostanie opisane szerzej w rozdziale 3.3 poświęconemu rozwiązaniom hybrydowym.

W procesie kompilacji wymienić możemy fazy ukazane na Rysunku 4:



Rysunek 4. Fazy kompilacji programu źródłowego. **Źródło**: [22].

Opis faz kompilacji wymienionych na Rysunku 1, przedstawia się następująco [22]:

Analiza leksykalna – Jest to analiza liniowa, która polega na załadowaniu danych w postaci strumienia znaków z kodu źródłowego programu, które czytane są od strony lewej do prawej, a następnie grupowane są w symbole leksykalne. Za symbole leksykalne w tym przypadku rozumiemy ciągi znaków, które wspólnie składają się na pewne znaczenie.

Analiza składniowa – Inaczej nazywana analizą hierarchiczną, na jej etapie zachodzi grupowanie ciągów znaków na podstawie kodu źródłowego programu, które powstały podczas analizy leksykalnej, w wyrażenia gramatyczne. Najczęściej reprezentacją takich wyrażeń jest drzewo wyprowadzenia. Wykorzystywane są one przez kompilator, które następnie je łączy w celu dalszego utworzenia kodu wynikowego.

Analiza semantyczna – Jest to analiza, w której zachodzi kontrola kodu źródłowego napisanego przez programistę pod względem semantycznym. Oznacza to sprawdzenie czy wyrażenia pod kątem ich znaczenia współgrają ze sobą w odpowiedni sposób. Tutaj również zachodzi sprawdzenie poprawności typów, czyli weryfikacji czy operatory mają poprawne argumenty względem specyfikacji danego języka. W tej fazie dane przygotowywane są do następnego etapu w postaci procesu generowania kodu pośredniego.

Proces generowania kodu pośredniego – W tej części, generowany jest kod pośredni odpowiadający programowi źródłowemu. Powinien on spełniać kryteria w postaci łatwości jego utworzenia oraz przetłumaczenia na kod wynikowy. Przykładem takiego kodu jest kod trójadresowy, który składa się z grup rozkazów posiadających maksymalnie trzy argumenty.

Proces optymalizacji kodu – Na tym etapie, kod pośredni ulega poprawkom i przekształceniom, których celem jest to, żeby kod maszynowy, generowany w następnym kroku był jak najszybszy.

Proces generowania kodu – Proces ten sprowadza się do generowania kodu maszynowego bądź kodu asemblera. Tutaj przypisywane są adresy w pamięci dla zmiennych, następnie rozkazy z kodu pośredniego przetłumaczone zostają na rozkazy opisane w kodzie maszynowym.

Proces zarządzania tablicą symboli – Zachodzi równolegle podczas wszystkich wyżej wymienionych faz. Polega na odnotowywaniu identyfikatorów, które zostały użyte w kodzie źródłowym, a także zbieraniu danych dotyczących ich atrybutów. Na atrybuty składają się takie informacje jak:

- typ,
- zasięg widoczności,
- ilości zajmowanego miejsca w pamięci,
- nazwy funkcji oraz liczba i typ przyjmowanych przez nie argumentów,
- typ zwracanej wartości przez funkcję,
- rodzaj przekazywania argumentów do funkcji (przez wartość czy przez referencje).

Zbiór, który zawiera w sobie powyższe informacje tj. identyfikatory i odpowiadające im atrybuty, nazywamy tablicą symboli. Dzięki niej możliwe jest zlokalizowanie rekordu dla poszczególnych identyfikatorów oraz zapis i odczyt danych.

Proces obsługi błędów – Podobnie jak przy zarządzaniu tablicą symboli, tak samo proces obsługi błędów funkcjonuje w każdej fazie kompilacji. Podczas ich działania mogą wystąpić błędy, które powinny zostać obsłużone, tak aby kompilacja nie zakończyła swojego działania po pierwszym napotkanym błędzie.

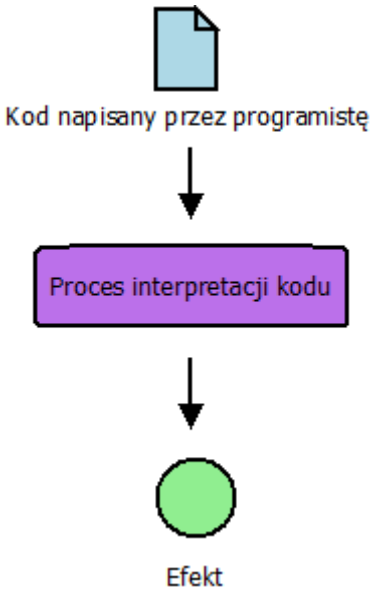
Pojęciem związanym z kompilacją, ale nie wchodzącym bezpośrednio do jej faz są preprocesory. Preprocesory są to programy, które przetwarzają kod przed skompilowaniem go.

Przykładem języka ulegającego procesowi kompilacji, jest język C++.

Procesem odwrotnym w stosunku do kompilacji jest dekompilacja.

3.2. Języki interpretowane i proces interpretacji

Interpretacja jest przeciwieństwem procesu kompilacji, bowiem nie zachodzi w niej proces translacji. W programach napisanych w językach interpretowanych kod interpretowany jest przez program zwany interpreterem [23]. W tym przypadku interpreter jest w stanie odczytać kod źródłowy i na jego podstawie wykonać określone w programie instrukcje. Kolejną cechą odróżniającą interpretację od kompilacji jest to, że w jej wyniku instrukcje wykonywane są natychmiastowo, w przeciwieństwie do tej drugiej, gdzie w wyniku jej działania powstaje kod wynikowy. Rysunek 5 przedstawia schemat czystej interpretacji.



Rysunek 5. Schemat interpretacji. **Źródło**: Opracowanie własne.

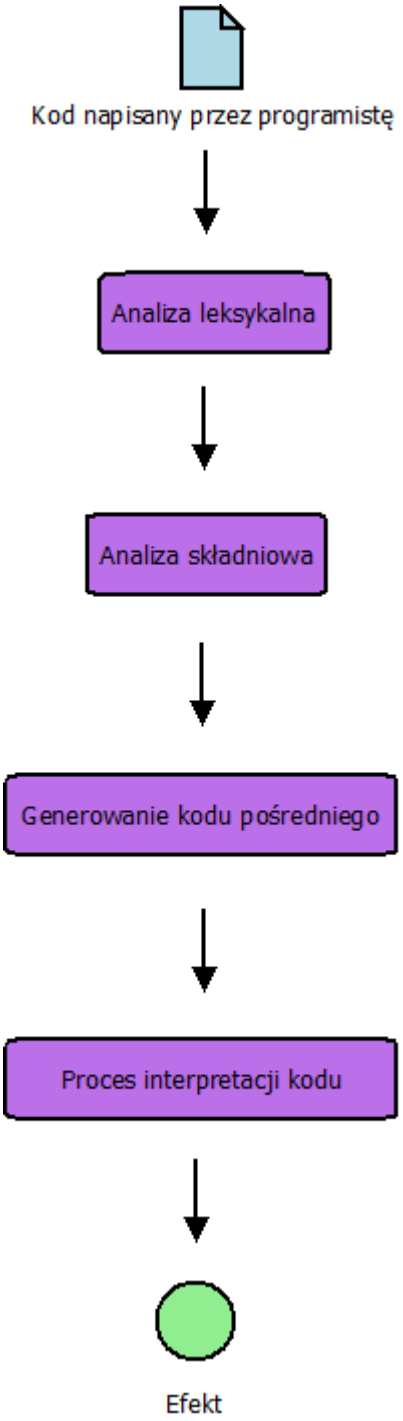
Proces interpretacji kodu polega na tym, że interpreter najpierw wczytuje kod napisany przez programistę, a następnie napotykając na poszczególne instrukcje w kodzie, natychmiast je wykonuje. Przykładem języka ulegającego procesowi interpretacji, jest język PHP.

Poniżej przykłady popularnych interpreterów [24]:

- Interpreter języka Lisp,
- Interpreter języka Basic,
- Interpreter języka linii komend Unix,
- Interpreter SQL.

3.3. Języki hybrydowe

Języki hybrydowe wykorzystują zarówno kompilacje jak i interpretacje. W takich językach następuje translacja na podstawie kodu źródłowego do kodu pośredniego. Nie powstaje tutaj gotowy program do wykonania, jak ma to miejsce w językach kompilowanych. Kod w takiej formie pozwala na jego szybsze procesowanie w dalszym etapie, niż miałoby to miejsce w przypadku czystej interpretacji. Dzieje się tak dlatego, ponieważ zdekodowanie instrukcji zawartych w kodzie źródłowym ma w tym przypadku miejsce tylko jeden raz [25]. Przykładem języka hybrydowego jest Java, w której kod kompilowany jest nie do kodu maszynowego, a do kodu bajtowego Javy, gdzie następnie ulega on obsłudze przez Wirtualną Maszynę Javy. Proces ten został opisany szerzej w rozdziale 4.2. Schemat przetwarzania kodu przez języki hybrydowe widoczny jest na Rysunku 6:



Rysunek 6. Schemat przetwarzania kodu w językach hybrydowych. **Źródło**: [25].

Analiza leksykalna oraz analiza składniowa zostały opisane w rozdziale 3.1, poświęconym językom kompilowanym. Generowanie kodu pośredniego polega na wytwarzaniu na podstawie kodu źródłowego napisanego przez programistę kodu pośredniego. Kod pośredni jest bliżej sprzętu niż kod źródłowy, natomiast jest on oczywiście dalej od sprzętu od kodu maszynowego. Przykładem kodu pośredniego jest Bajtowy kod Javy. Kod bajtowy wziął swoją nazwę od długości kodów operacyjnych w postaci jednego bajta. Kod pośredni jest często przeznaczony dla wirtualnych maszyn. W przypadku kodu pośredniego Javy, jest on przeznaczony dla Wirtualnej maszyny Javy (JVM). Takie rozwiązanie niesie za sobą szereg korzyści. Jedną z nich jest to, że do takiego kodu pośredniego mogą być kompilowane źródła napisane w różnych językach programowania. Do Kodu bajtowego Javy istnieje przykładowo możliwość skompilowania takich języków jak Clojure, Groovy czy Scala. Innym przykładem języka pośredniego jest Wspólny język pośredni, z angielskiego *Common intermediate language* (CLI) dla platformy .Net Microsoftu. Do takiej formy mogą być kompilowane takie języki jak C#, Visual Basic, F#, czy C++ zmodyfikowany przez Microsoft pod kątem CLI. Kolejną zaletą płynącą ze sposobu przetwarzania kodu w językach hybrydowych jest przenaszalność. Kod skompilowany do postaci kodu pośredniego może być odpalany w różnych miejscach, posiadających te samo środowisko uruchomieniowe w postaci wirtualnej maszyny. W przypadku CLI kod może być odpalany bez problemu na takich systemach operacyjnych jak Windows, Linux jak i macOS. Jeśli chodzi o platformę Microsoft to środowiskiem uruchomieniowym jest *Common Language Runtime* (CLR), gdzie istnieją jego odmiany dedykowane dla wcześniej wspomnianych systemów. Proces interpretacji kodu w przeciwieństwie do czystej interpretacji następuje w przypadku języków hybrydowych nie na podstawie plików z kodem źródłowym tylko na bazie kodu pośredniego. Kod pośredni, może być punktem wejściowym dla innego języka pośredniego. Przykładem takiej sytuacji jest system operacyjny Android, gdzie Kod bajtowy Javy, ulega transformacji do Kodu bajtowego Dalvik, który dopiero później jest kompilowany do kodu maszynowego [26].

4. Java – język wykorzystujący proces kompilacji

Następujący rozdział zawiera krótki wstęp do języka Java. Zawarte są w nim informacje dotyczące jego historii oraz zastosowań. Opisane są tutaj również cechy charakterystyczne czy też podział na platformy Javy. W drugim podrozdziale przedstawiony natomiast został sposób tworzenia i obsługi kodu napisanego w Javie oraz pojęcia z tym związane.

4.1. Wprowadzenie do języka Java i jego charakterystyka

Java jest językiem wysokopoziomowym, wspierającym obiektowy paradygmat programowania. Twórcą języka Java, jest James Gosling. A jego pierwsza wersja została wypuszczona podczas jego pracy w firmie Sun Microsystems w roku 1995. W roku 2010 firma Oracle Corporation zakończyła proces przejmowania Sun Microsystems i od tego czasu nabyła prawa do linii produktów w postaci oprogramowania Java. Język ten znany jest z tego, że raz napisany i skompilowany kod może zostać z łatwością odpalony na różnych środowiskach sprzętowych, z różnymi systemami operacyjnymi bez konieczności ponownej kompilacji. Jest on z sukcesem używany do wielu różnych zastosowań. Java stosowana jest do pisania kodu dla:

- **Programów okienkowych** – Z Javą związane są takie technologie jak JavaFX, czy wcześniej biblioteka Swing i jej poprzedniczka w postaci biblioteki AWT, które są powszechnie wykorzystywane do tworzenia aplikacji okienkowych.
- **Programów webowych** – Java jest również językiem, który jest bardzo często wykorzystywany do tworzenia oprogramowania internetowego. Zawiera on licznego rodzaju biblioteki, wspierające wytwarzanie tego typu aplikacji. Dodatkowo dla programów należących do tej grupy, wymienić można między innymi framework Spring oraz technologie EJB (ang. *Enterpise JavaBeans*). W przypadku tego pierwszego, posłużył on jako baza dla oprogramowania w wersji Java, które zostało stworzone w ramach niniejszej pracy. Projekt ten oraz wspominany framework zostały opisane szerzej w rozdziale szóstym.
- **Programów przeznaczonych na urządzenia mobilne** – Java jest jednym z głównych języków przeznaczonych do pisania aplikacji na system Android [27]. System ten jest obecnie najpopularniejszym systemem operacyjnym na urządzenia mobilne [28] jak i ma największy udział w rynku wszystkich systemów operacyjnych na świecie [29]. Oficjalnym zintegrowanym środowiskiem programistycznym do rozwoju aplikacji na niego jest Android Studio, które oczywiście wspiera język Java [30]. System Android jest wykorzystywany przez takie maszyny jak:
    - smartfony,
    - tablety,
    - telewizory,
    - systemy multimedialne znajdujące się w samochodach,
    - smartwatche,
    - czytniki książek w formie elektronicznej.

  Przy pomocy Javy programista ma możliwość tworzenia oprogramowania na wszystkie powyższe urządzenia.
- **Oprogramowania wbudowanego** – Tutaj należy wyróżnić platformę Java Card. Jest to technologia, która umożliwia takim narzędziom jak karty mikroprocesorowe (ang. *smartcards*) obsługiwać aplikacje Javy, nazywane apletami [31]. Ze względu na oczywiste ograniczenia w postaci miniaturyzacji, platforma Java Card dostarcza tylko niewielkiego zestawu możliwości Javy. Oprócz kart mikroprocesorowych Java stosowana jest jako oprogramowanie wbudowane na takich urządzeniach jak odtwarzacze Blu-ray, telefony wykorzystujące sieci komputerowe, różnego rodzaju mierniki pokazujące pobór wody czy energii elektrycznej oraz różnego rodzaju inne urządzenia [32].

Java jest podzielona na platformy. Platforma Javy jest to środowisko, w którym implementowane i zarządzane jest oprogramowanie. Składa się z trzech głównych komponentów: Wirtualnej maszyny Javy, Języka Java oraz Pakietów Javy [33]. Te ostatnie są jednostkami, w ramach których zorganizowane są powiązane ze sobą klasy, interfejsy czy typy enumeracyjne. Ich zastosowanie pozwala na uniknięcie problemów związanych z konfliktem w nazewnictwie i pozwala na logiczne pogrupowanie kodu. Są one podzielone na dwa podstawowe rodzaje [34]:

- Te, które są wbudowane w ramach Języka Java i pochodzą z Java API.
- Takie, które użytkownik/programista zdefiniował osobiście.

Można wyróżnić następujące, podstawowe edycje Javy [35]:

Java SE (Java Standard Edition) – Interfejs programistyczny tej platformy dostarcza rdzenia funkcjonalności języka Java. Zdefiniowane są tutaj wszystkie takie elementy jak obiekty Javy, typy podstawowe czy klasy wysokiego poziomu zawierające mechanizmy związane z bazami danych, sieciami, graficznym interfejsem użytkownika, bezpieczeństwem czy parsowaniem XML. Poza tym zawiera ona wirtualną maszynę, technologie do wdrażania, najrozmaitsze narzędzia do tworzenia kodu oraz wiele różnego rodzaju bibliotek, których używa się w ramach języka Java.

Java EE (Java Enterprise Edition) – Platforma *Java Enterprise Edition* stworzona jest na podstawie *Javy Standard Edition*. Zapewnia interfejs programowania aplikacji oraz środowisko uruchomieniowe dostosowane do rozwijania kodu oprogramowania sieciowego, wielowarstwowego, o dużej wielkości, cechującym się wysokim stopniem bezpieczeństwa, przystosowanym do skalowania.

Java ME (Java Micro Edition) – Edycja ta dostarcza API oraz małą wirtualną maszynę przeznaczoną do odpalania oprogramowania Javy na niewielkich urządzeniach, których przykładem mogą być telefony. Zapewnia ona interfejs programowania aplikacji, który jest podzbiorem API Standardowej Edycji Javy z dodatkowymi bibliotekami, które zawierają funkcjonalności będące użytecznymi przy tworzeniu aplikacji przeznaczonych na małe urządzenia.

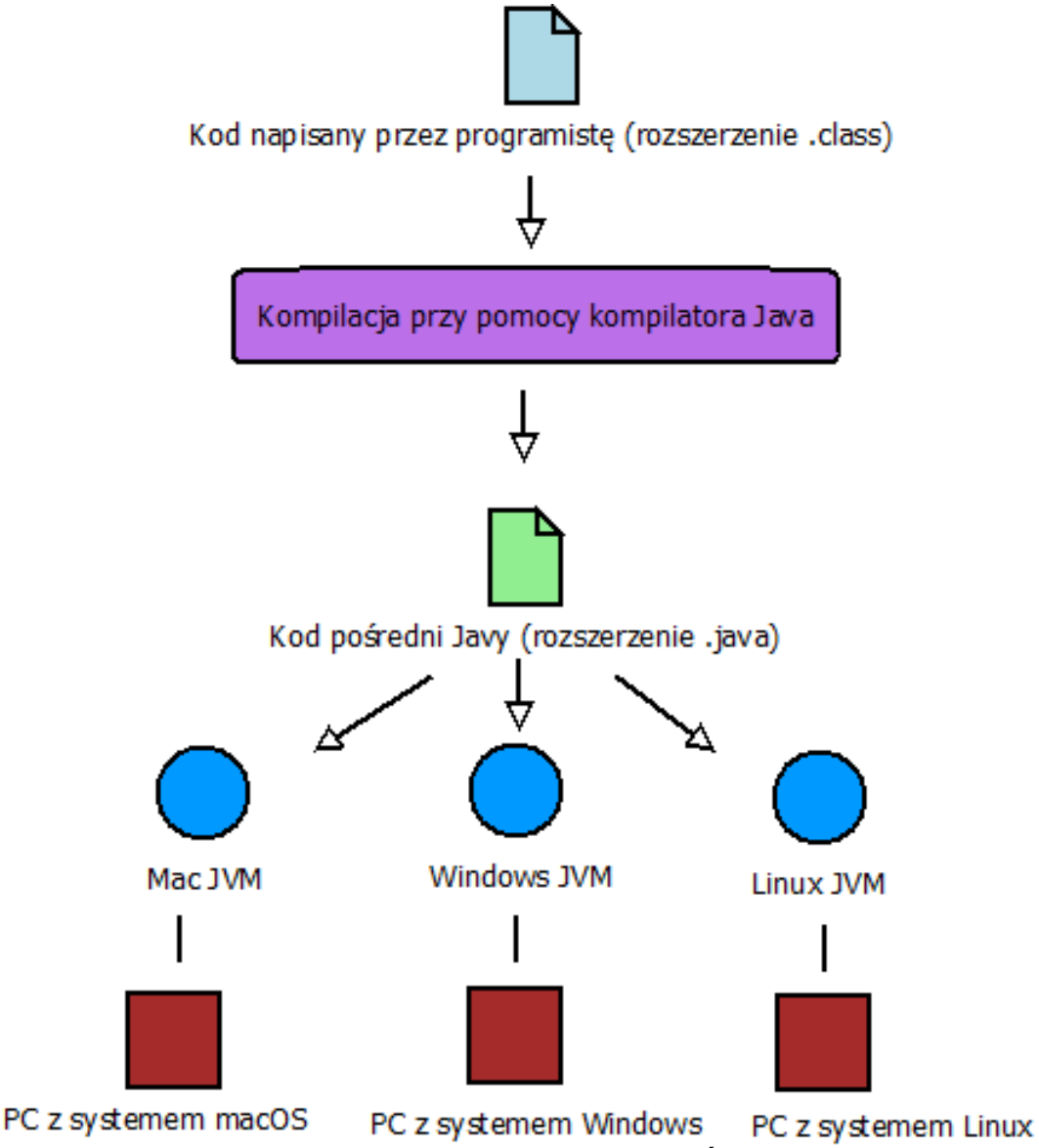
Java FX – Ta platforma przeznaczona jest do tworzenia lekkich internetowych aplikacji, dostarczających bogaty, dynamiczny interfejs użytkownika. Programy bazujące na JavaFX używają specjalnych silników dla grafiki i multimediów, dzięki czemu uzyskuje się większą wydajność po stronie klienckiej oraz nowoczesny wygląd. Dostarczany jest również wysokopoziomowy interfejs do łączenia się z sieciowymi źródłami danych. Aplikacje tworzone z użyciem tej platformy są często klientami serwisów opartych o platformę *Java Enterprise Edition*.

Zaliczana jest do języków silnie typowanych, co oznacza, że deklarując zmienną programista określa również jej typ. Podobnie jest z metodami, gdzie na etapie deklaracji podawany jest typ zwracany oraz rodzaj argumentów jakie dana metoda będzie przyjmować. Istnieją mechanizmy, które pozwalają ominąć taki sposób postępowania z kodem, jednak nie leży on u podstawowych założeń tego języka.

4.2. Tworzenie i obsługa kodu napisanego w języku Java

Java jest językiem, w którym zachodzi zarówno proces kompilacji jak i proces interpretacji. Programista pisząc kod zapisuje go w plikach o rozszerzeniu ‘class’. Pierwszym etapem w celu uruchomienia takiego kodu jest jego kompilacja. Oficjalnym kompilatorem języka Java jest ‘javac’. Kod kompilowany jest do kodu pośredniego w postaci Kodu bajtowego Javy. W wyniku kompilacji powstają pliki o rozszerzeniu ‘java’ – mogą one zostać do spakowane do paczki o rozszerzeniu ‘jar’, lub w przypadku aplikacji internetowej ‘war’. Do wykonania Kodu bajtowego Javy potrzebna jest Wirtualna Maszyna Javy, z ang. *Java Virtual Machine*, w skrócie JVM. Jest to pewien rodzaj maszyny liczącej w postaci maszyny wirtualnej, która posiada zestaw instrukcji i manipuluje pamięcią w trakcie swojego działania [36]. Istnieją jej różne implementacje, które powinny być zgodne z specyfikacją podaną przez firmę Oracle. Głównymi implementacjami JVM są 'Java HotSpot VM' oraz 'Oracle JRockit JVM' [37].

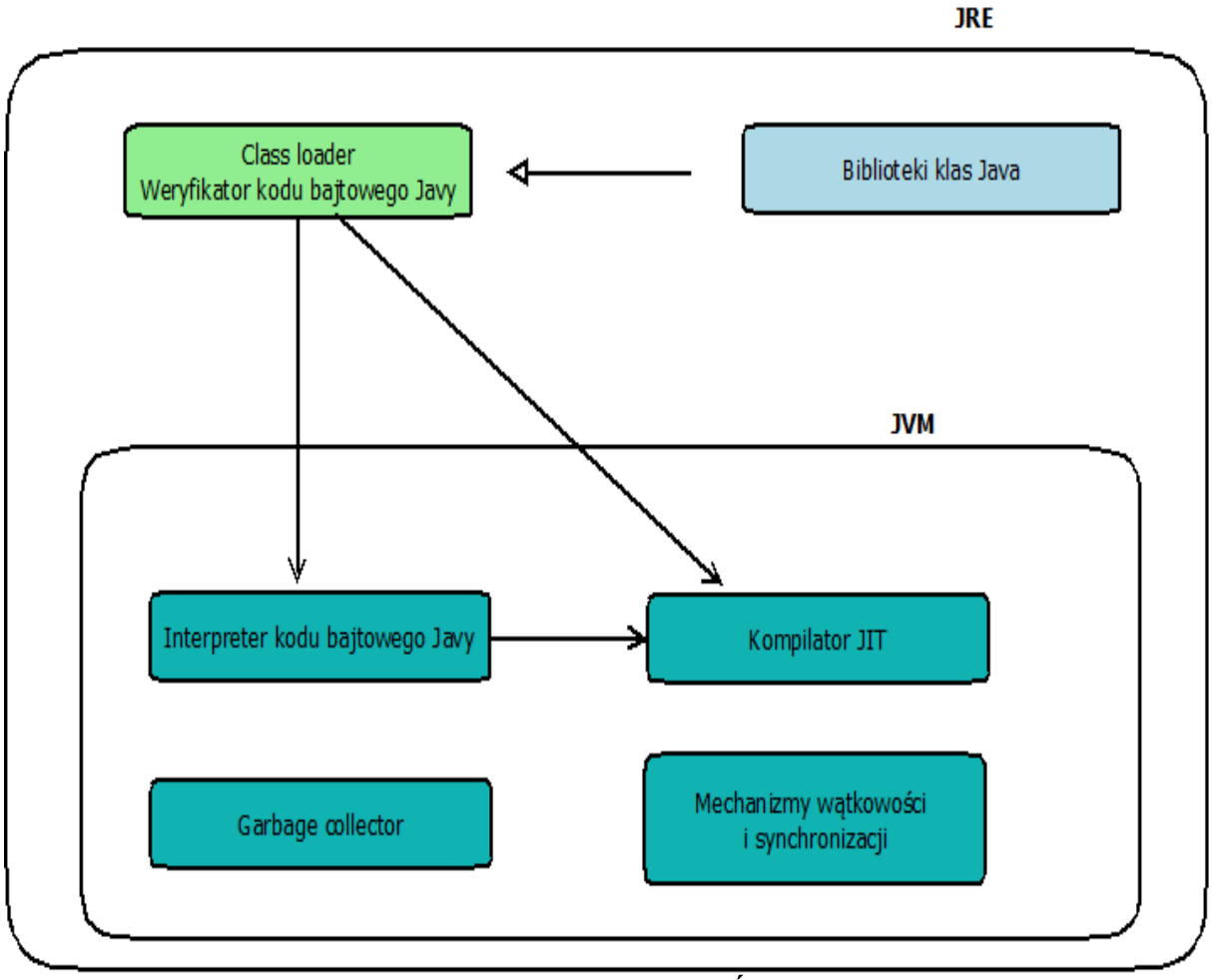
To implementacja HotSpot dołączana jest do oficjalnego środowiska uruchomieniowego Javy (JRE, z ang. *Java Runtime Environment*) oraz zestawu deweloperskiego dla Javy (JDK, z ang. *Java Development Kit*). Zarys sposobu wytwarzania i wykonywania kodu Javy przedstawiony został na Rysunku 7.



Rysunek 7. Schemat procesu tworzenia i wykonywania kodu Javy. **Źródło**: Opracowanie własne.

Zgodnie z Rysunkiem 7 wykorzystanie Wirtualnej maszyny Javy, pozwala na łatwą przenoszalność programów napisanych w Języku Java. Raz napisany i skompilowany kod może zostać uruchomiony zarówno na komputerach z systemem Windows, macOs, czy Linux, jeśli te posiadają stosowną wersję JVM.

Na Rysunku o numerze 8, znajduje się schemat środowiska uruchomieniowego Javy wraz z Wirtualną maszyną Javy.



Rysunek 8. Schemat JRE i JVM. **Źródło**: [38].

Poniżej znajduje się krótki opis komponentów wchodzących w skład JRE oraz JVM na podstawie książki *Beginning Java Programming. The Object-Oriented Approach* napisanej przez Profesora Barta Baesens'a oraz Doktorów Aimee Backiel'a i Seppe vanden Broucke'a [38]:

Biblioteki klas Java – Jest to zestaw wcześniej spakowanych elementów Javy w formie bibliotek. Dzięki nim programista ma możliwość wykorzystania wielu użytecznych, gotowych do użycia funkcjonalności, które zostały już dostarczone. Pozwala to na zaoszczędzenie czasu i skorzystanie z przetestowanych i zoptymalizowanych komponentów. Przykładem jest biblioteka 'Java.sql', która pozwala między innymi na uzyskanie dostępu do relacyjnej bazy danych SQL.

Class Loader – Odpowiedzialny jest za ładowanie, a następnie odczytywanie plików z rozszerzeniem 'class'. Posiada on trzy podstawowe składowe:

- Bootstrap class loader – Zajmuje się ładowaniem bibliotek rdzenia Javy.
- Class loader rozszerzeń – Ładuje klasy z katalogu zawierającego rozszerzenia, czyli następującej lokalizacji <JAVA_HOME>/jre/lib/ext.
- Systemowy class loader – Odpowiedzialny jest za załadowanie kodu z lokalizacji określonych przy pomocy zmiennej środowiskowej CLASSPATH. Pełni ona funkcję informującą aplikacje i narzędzia, w którym miejscu należy poszukiwać klas zdefiniowanych przez użytkownika [39].

Weryfikator kodu bajtowego Javy – Sprawdza czy Kod bajtowy Javy jest poprawny. Weryfikowane jest również czy nie została naruszona żadna z zasad bezpieczeństwa Javy. Na tym etapie sprawdzane są również zmienne i wyrażenia oraz czy nie następuje nieautoryzowany dostęp do pamięci. Możliwe jest wyłączenie weryfikatora Kodu Bajtowego Javy. Po tym jak weryfikator Kodu bajtowego Javy zakończy weryfikację, kod przekazywany jest do Wirtualnej Maszyny Javy.

Wirtualna Maszyna Javy – Główną składową JVM jest interpreter, którego rolą jest interpretowanie kodu bajtowego Javy. W jej skład wchodzi również *Garbage Collector*, który jest odpowiedzialny za czyszczenie pamięci oraz mechanizmy synchronizacji i wielowątkowości.
W przypadku wielu jej implementacji zawiera ona również kompilator. Kompilacja, która następuje wewnątrz JVM to kompilacja typu JIT, z angielskiego *Just In Time*. Schemat działania wygląda tak, że interpreter sprawdza jak często poszczególny kod jest wykonywany i te instrukcje, które wywoływane są z dużą częstotliwością przekazywane są do kompilatora JIT, który je skompiluje do kodu natywnego. Dzięki takiemu zabiegowi czas egzekucji kodu zostaje zmniejszony. Istnieje możliwość, żeby to użytkownik określił próg, który wyznaczy czy dany kod jest wykonywany z dużą częstotliwością, czy też nie.

5. PHP – język wykorzystujący proces interpretacji

Rozdział piąty poświęcony jest językowi PHP oraz procesowi, w którym jest on wytwarzany, a następnie różnym sposobom obsługi jego kodu źródłowego. Na wstępie w ramach wprowadzenia przedstawiona została zwięźle historia PHP wraz z przybliżeniem przebiegu kształtowania się tego języka z czasem i różnymi wersjami. Pokazany został udział procentowy różnych wydań PHP oraz zaprezentowana została jego charakterystyka, zastosowanie i pojęcia silnie z nim związane. W ostatnim podrozdziale ukazane zostały różne sposoby przetwarzania kodu napisanego w języku PHP w zależności od jego wersji.

5.1. Wprowadzenie do języka PHP i jego charakterystyka

PHP jest językiem nakierowanym na tworzenie stron internetowych. Został on stworzony w roku 1994 przez Rasmusa Lerdorfa. Początkowo służył on duńsko-kanadyjskiemu programiście jako narzędzie, które umożliwiało liczenie ilości wejść na jego internetowe CV, a skrót PHP pierwotnie oznaczał '*Personal Home Page Tools*', czyli narzędzia osobistej strony głównej. Przez długi okres czasu PHP rozwijany był głównie przez jedną osobę. W tym czasie projekt był rozwijany jako *Common Gateway Interface* (CGI), czyli interfejs pozwalający na współdziałanie z innymi komponentami znajdującymi się na serwerze WWW. PHP został w tym czasie wzbogacony o różne funkcjonalności, a jedną z nich była możliwość współdziałania z bazami danych. Od roku 1995 kod składający się na ten język został przez jego twórcę publicznie udostępniony.

Od wersji PHP 3.0 do projektu dołączyło dwóch izraelskich programistów Andi Gutmans oraz Zeev Suraski, którzy ocenili, że obecny stan języka nie jest wystarczający dla ich potrzeb w postaci uczelnianego projektu związanego z elektronicznym handlem. Język podczas ich prac został w znaczny sposób przebudowany. W wersji tej zostało wprowadzone wsparcie dla obiektowego programowania. W szczytowym momencie dla tego wydania języka, był on zainstalowany na około dziesięciu procentach wszystkich serwerów WWW w Internecie. Dopiero od wersji PHP 3.0 nabrał on formy, która przypomina ten język w aktualnym kształcie. Obecnie skrót PHP rozwijany jest do '*Hypertext PreProcsesor*'.

Jak się okazało nie był to ostatni znaczny wpływ wspomnianych programistów z Izraela, ponieważ w kolejnej wersji rozpoczęli oni pracę nad przepisaniem rdzenia PHP, co więcej w wyniku ich prac zadebiutował silnik Zend Engine, którego nazwa pochodzi od imion jego twórców [40]. Miał on duże znaczenie dla poprawy efektywności języka i został szerzej opisany w kolejnym rozdziale. Wraz z późniejszymi wydaniami następnych wersji języka, dodawane były nowe możliwości jakie oferuje on programistom oraz wprowadzane były kolejne jego usprawnienia w wydajności. Na dzień 19.01.2022 najnowsza wersja PHP to 8.1.1 i została wypuszczona 17 grudnia 2021 roku.

Użycie poszczególnych wersji PHP prezentuje Tabela 3, która została opracowane na podstawie informacji dostarczonych przez W3Techs [41]. Dane te są o tyle istotne, że w różnych wersjach kod PHP jest przetwarzany w różny sposób, co zostało opisane w rozdziale 5.2, który poświęcony jest tworzeniu i obsłudze kodu napisanego w języku PHP.

Tabela 3. Udział poszczególnych wersji PHP w ogóle stron wykorzystujących ten język. **Źródło**: [41].

Wersja PHP	**Wynik procentowy**
7	70,2%
5	27,7%
8	2,0%
4	0,2%
3	mniej niż 0.1%

Język PHP jest powszechnie używany do tworzenia systemów zarządzania treścią (ang. *Content Management System*, w skrócie CMS), które mają na celu ułatwienie tworzenie witryn internetowych oraz późniejsze zarządzanie nimi, nawet przez osoby nie będące programistami. Dostarczają one

stosunkowo prostego interfejsu dla użytkownika. Zdecydowanie najbardziej popularnym systemem CMS jest WordPress, który na chwilę obecną ma udział w rynku tego rodzaju systemów kształtujący się na poziomie 65,3%, a stosunek stron używających WordPressa do wszystkich stron internetowych wynosi aż 43,2% [42]. Został on napisany w języku PHP i to w tym języku piszę się również różnego rodzaju wtyczki do niego dedykowane [43]. Tabela 4 opracowana na podstawie danych, dostarczonych przez portal W3Techs [42] przedstawia udział w rynku poszczególnych rozwiązań w postaci systemów zarządzania treścią:

Tabela 4. Statystyki użycia systemów CMS. **Źródło**: [42].

	Udział w rynku wśród wszystkich systemów CMS	**Udział w rynku wśród wszystkich witryn WWW**
WordPress	65,3%	43,2%
Shopify	6,7%	4,4%
Wix	2,9%	1,9%
Squarespace	2,7%	1,8%
Joomla	2,6%	1,7%
Drupal	2,0%	1,3%
Adobe	1,6%	1,0%
Blogger	1,5%	1,0%
Bitrix	1,4%	0,9%
OpenCart	0,9%	0,6%
PrestaShop	0,7%	0,5%
Webflow	0,7%	0,4%
Weebly	0,5%	0,4%

Tabela, jest o tyle ciekawa z punktu widzenia języka PHP, że oprócz WordPressa został on wykorzystany do stworzenia następujących systemów:

- Joomla [44],
- Drupal [45],
- Bitrix [46],
- OpenCart [47]
- PrestaShop [48],
- Weebly [49].

Powyższe dane ilustrują jak duży udział ma PHP na rynku systemów CMS oraz stron internetowych. W jeszcze większym stopniu powiązanie Internetu z PHP pokazuje to, że jest on używany po stronie serwerowej w około 78,1% wszystkich stron internetowych [41].

Na głównej stronie oficjalnej witryny PHP (https://www.php.net/), można przeczytać, iż jest to język skryptowy. W tym miejscu należy wyjaśnić czym jest skryptowy język. Jak podaje Słownik języka polskiego skrypt jest to: „*prosty program komputerowy składający się z ciągu poleceń systemu operacyjnego lub instrukcji uruchamiających inne programy*” [50]. W ocenie autora niniejszej pracy nie jest to jednak definicja pozwalająca jednoznacznie określić jaki język może być uznany za skryptowy, a który już się do tej grupy nie zalicza. W Internecie można znaleźć wiele różnych definicji języka skryptowego i skryptu. Na stronie Uniwersytetu Loyola Marymunt zamieszczono opinię, według której nie istnieje jedna definicja języka skryptowego [51]. Podjęto jednak próby podania trzech definicji. Jedna z nich opisuje go jako język bardzo wysokiego poziomu, który cechuje się tym, że przy jego pomocy można w łatwy sposób wytworzyć kod, a programista nie musi się zajmować takimi kwestiami jak zarządzanie pamięcią, sprawdzanie granic i ma możliwość korzystania z prostych konstrukcji. W książce ‘Invitation to Computer Science’ autorstwa G. Michael Schneider’a oraz Judith Gersting [52] można natomiast znaleźć informację, iż język skryptowy to taki, który jest lekkim językiem, ulegającym procesowi interpretacji rozumianemu jako wykonywanie kodu instrukcja po instrukcji. Dodatkowo napisano w niej, że fragmenty skryptów mogą być umieszczane wewnątrz stron internetowych.

Język PHP z pewnością można uznać za spełniający wyżej wymienione kryteria skryptowego języka programowania. Jest on uznawany za język stosukowo przyjazny do nauki [53] [54]. Jest dynamicznie typowany, co oznacza przykładowo, że nie ma konieczności podawania typu zmiennej, gdyż istnieje mechanizm, który w czasie wykonywania kodu automatycznie rozpozna na postawie kontekstu jaki powinien być jej typ [55]. Dodatkowo cechuje się tym, że programista nie zajmuje się kwestią alokowania miejsca w pamięci i jego późniejszym zwalnianiem jak ma to przykładowo miejsce w języku C++. PHP również idealnie wpisuje się w to, że kod w nim napisany osadzany jest jako część będąca stroną internetową.

W PHP od wersji 5.3 wbudowany jest mechanizm służący do automatycznego odśmiecania pamięci (ang. *Garbage Collection*) [56].

PHP wspiera obiektowy paradygmat programowania, dostarczając mechanizmów pozwalających na definiowanie klas, określanie widoczności, abstrakcji czy dając możliwość tworzenia interfejsów [57].

Kod PHP może być pisany na różnych systemach operacyjnych takich jak Linux, Windows czy maxOS i na nich również może być postawiony serwer www, tworzący infrastrukturę dla oprogramowania webowego. Skrypty napisane w tym języku mogą być również odpalane bez udziału przeglądarki internetowej. PHP wyposażony jest w oddzielny, wbudowany interfejs linii komend przy którego pomocy można odpalać kod napisany w tym języku. Został on wprowadzony w wersji 4.3.0 [58].

Kluczowymi elementami wchodzącymi w skład systemu języka PHP są PEAR i PECL:

- **PEAR** – Jest to skrót od PHP Extension and Application Repository, czyli Repozytorium rozszerzeń i aplikacji PHP. PEAR to framework oraz system dystrybucji komponentów. Projekt ten został założony w roku 1999 rzez Stiga Bakken'a. Dostarcza ustrukturyzowaną bibliotekę z kodem open source, system dystrybucji kodu, rozszerzenie PECL, witrynę internetową oraz kopie zasobów. Kod dostarczany jest w ramach pakietów, gdzie każdy pakiet posiada swój własny zespół deweloperski, dokumentacje, numer określający wersję i może pozostawać w osobistej relacji względem innych paczek. Są one dystrybuowane w formie plików archiwum o formacie tar i zawierają w środku między innymi plik z opisem. Znajdują się w centralnym serwerze pod adresem pear.php.net, skąd mogą być pobierane [59].
- **PECL** – Rozwinięciem tego akronimu jest PHP Extension Community Library, co oznacza Społecznościową bibliotekę rozszerzeń PHP. Jest to repozytorium rozszerzeń dla języka PHP. Zajmuje się udostępnianiem katalogu, w którym znajdują się rozszerzenia PHP, skąd istnieje możliwość ich pobrania oraz dalszego rozwijania. Jest to siostrzany projekt w stosunku dla PEAR [60]. To co je wyróżnia jest to, że napisane są w języku C [59].

Interesującym faktem, że powstało rozszerzenie do PHP, które nazywa się PHP-GTK i na celu ma umożliwienie tworzenia okienkowych aplikacji z graficznym interfejsem użytkownika. Rozwiązanie to bazuje na bibliotece GTK+. Warto jednak odnotować, iż ostatnie wypuszczenie wersji miało miejsce w styczniu 2015 i dotyczyło wersji PHP 5.5 [61]. Innym rozwiązaniem, które pozwala na tworzenie programów okienkowych jest wxPHP, które opiera się na międzyplatformowej bibliotece wxWidgets [62]. Jednak i w tym przypadku ostanie wydanie wersji datowane jest na rok 2015 [63].

Język PHP jest dobrze udokumentowany, a oficjalna dokumentacja znajduje się pod adresem [64], gdzie dostępna jest zarówno w formie strony internetowej jak i istnieje możliwość jej ściągnięcia na lokalny komputer.

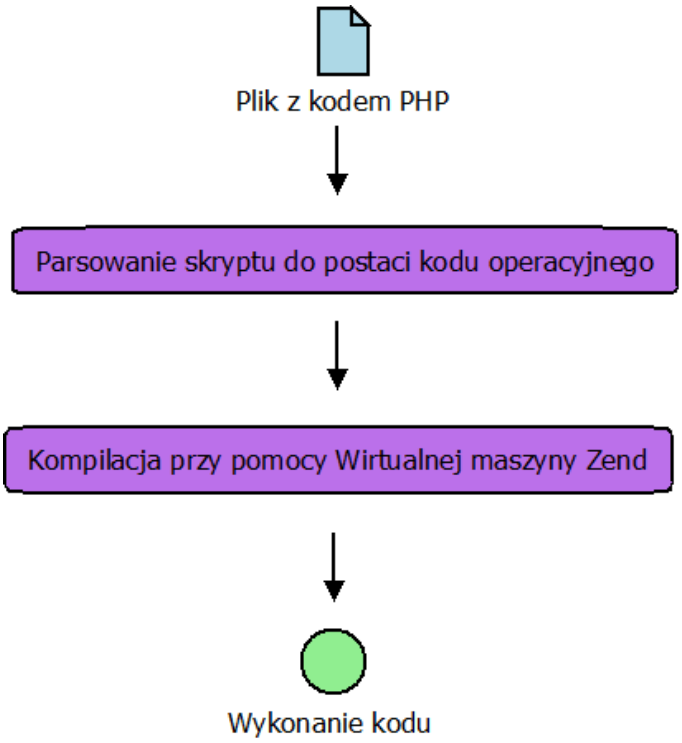
5.2. Tworzenie i obsługa kodu napisanego w języku PHP

Język PHP uznawany jest za interpretowany. Kod źródłowy zapisywany jest w plikach o rozszerzeniu ‘php’. Możliwe jest umieszczenie całej aplikacji PHP w archiwum PHP, o rozszerzeniu ‘phar’ [65]. Od wersji PHP 4 domyślnym silnikiem jest Zend Engine. Jest to interpreter kodu, który został napisany w języku C. Jego zadaniem jest parsowanie, interpretacja oraz wykonywanie kodu napisanego w języku PHP [66]. Działa on na zasadzie wirtualnej maszyny, która funkcjonuje jako interpreter. Źródła Silnika Zend są otwarte (ang. *open source*).

Portal ‘phpbuilder.com’ istniejący od roku 1999 podaje, że można wyróżnić następujące komponenty odpowiedzialne za obsługę kodu PHP przez Zend Engine [67]:

- **Analizator leksykalny (ang. *lexical analysis*)** – Kod PHP poddawany jest przez niego analizie leksykalnej. Ulega on również tutaj transformacji z pierwotnej formy kodu PHP do postaci tokenów, które są bezpośrednio zrozumiałe i akceptowane przez komputer. Po zakończeniu tych operacji cały kod przekazywany jest do następnego etapu, czyli parsowania.
- **Parser** – Kod otrzymany w wyniku analizy leksykalnej, w formie tokenów jest parsowany przez narzędzie nazywane parsem. Podczas tego stadium tworzone jest abstrakcyjne drzewo składniowe, na bazie którego może nastąpić optymalizacja przed przekazaniem go do generatora kodu. Efektem tego etapu, który określany jest jako kompilacja jest kod pośredni. Jest on niezależny od maszyny i przeznaczony jest dla Wirtualnej maszyny Zend (ang. *Zend virtual machine*). Zawiera tablicę instrukcji, dedykowanych dla maszyny wirtualnej, które nazywane są kodem operacyjnym i znane są również jako ‘opcodes’.
- **Egzekutor/wykonawca kodu (ang. *executor*)** – Skrypt, który w wyniku poprzednich etapów został sprowadzony do formy kodu pośredniego, jest przygotowany do wykonania, co następuje w ostatniej fazie przetwarzania kodu przez Silnik Zend. Oznacza to, że wykonawca (ang. *executor*), odczytuje po kolei każdą instrukcję z tablicy, a następnie wszystkie je wykonuje.

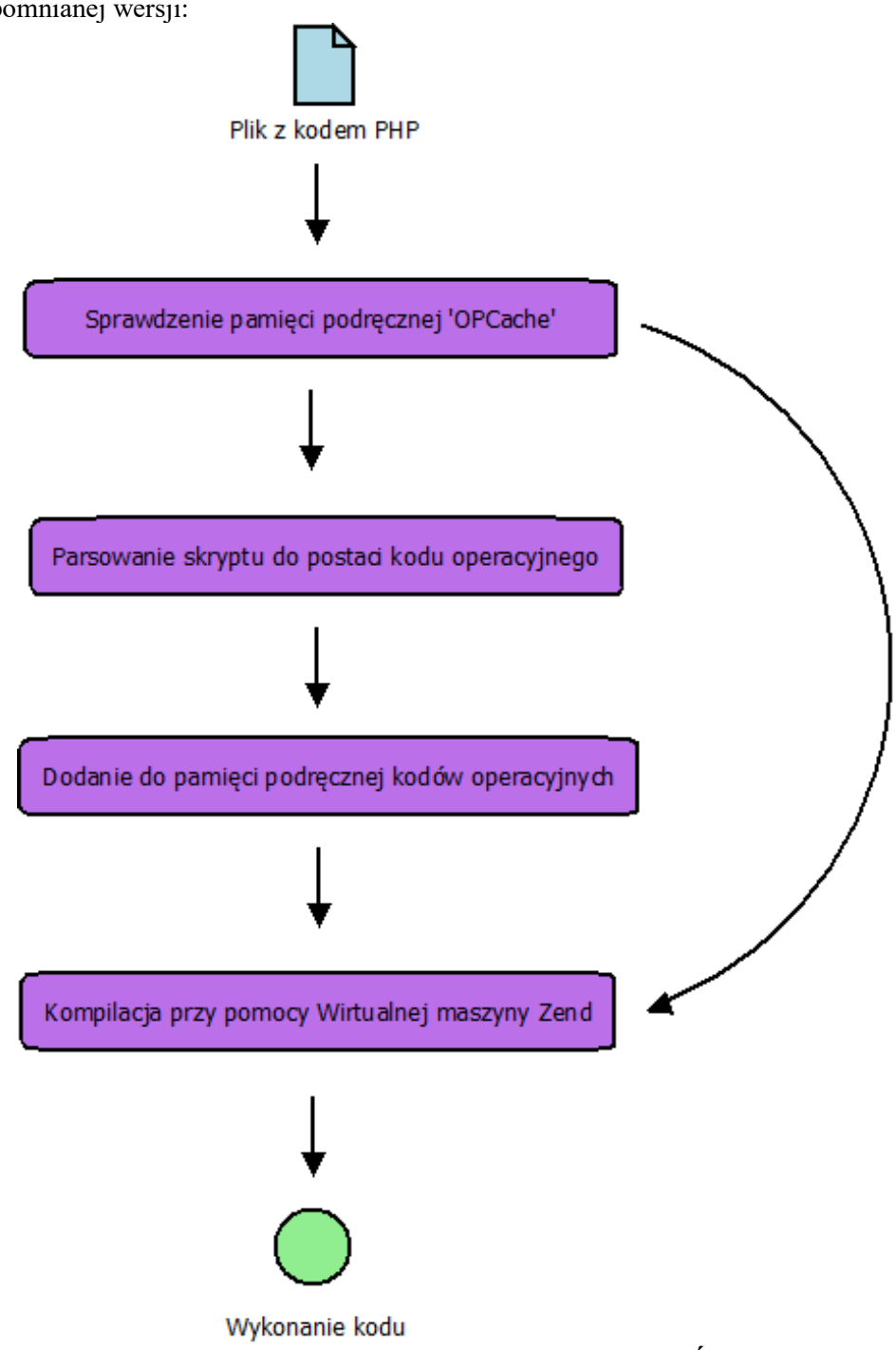
Poniżej opisane i przedstawione zostały schematy przetwarzania kodu źródłowego PHP w jego różnych wersjach na podstawie artykułu 'Exploring the New PHP JIT Compiler' z oficjalnej strony przedsiębiorstwa ‘Zend by Perforce’, ściśle związanego między innymi z silnikiem 'Zeng Engine' [68]. Na Rysunku 9, znajduje się pierwszy, najprostszy schemat przetwarzania kodu PHP we wczesnych wersjach tego języka:



Rysunek 9. Pierwszy schemat przetwarzania kodu PHP. **Źródło**: [68].

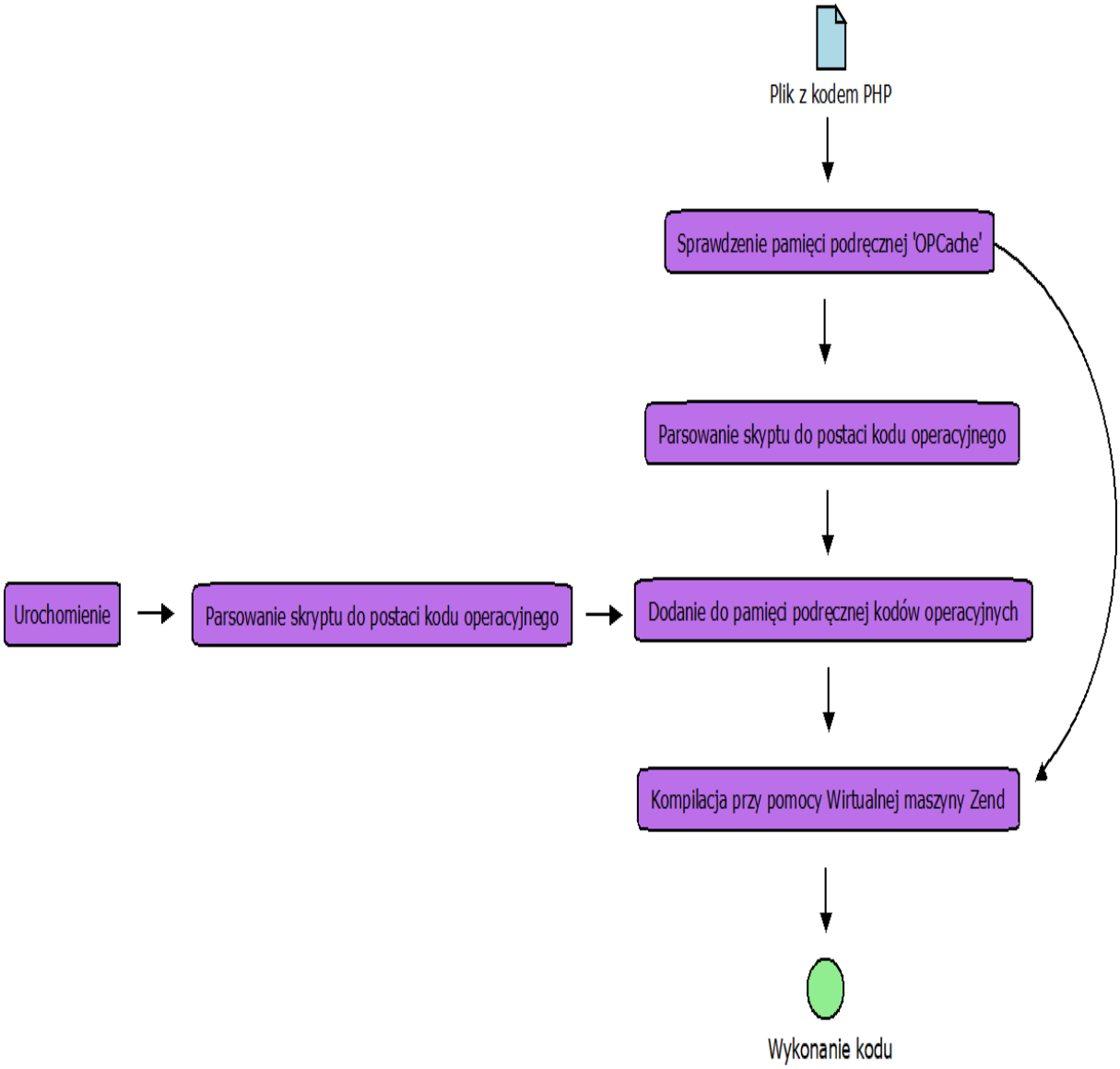
Z kodami operacyjnymi powstającymi na podstawie kodu PHP wiąże się pamięć podręczna, która znajduje się w silniku Zend - 'Cache Opcode'. Jej działanie wygląda następująco: Skrypt PHP zamieniany jest na kody operacyjne, następnie w przypadku, gdy nastąpi ponowne wykonanie kodu z danego pliku, a nie uległ on zmianie to zostaje wykonany kod operacyjny, który znajduje się już w pamięci podręcznej. Takie rozwiązanie przyśpiesza działanie oprogramowania zaimplementowanego w języku PHP. Zostało ono wprowadzone wraz z wersją PHP 5.5.0, w której dodano je do Silnika Zend jako rozszerzenie ‘Zend OPCache’ [69].

Na Rysunku 10 znajduje się przebieg wykonywania skryptów napisanych w języku PHP od wyżej wspomnianej wersji:



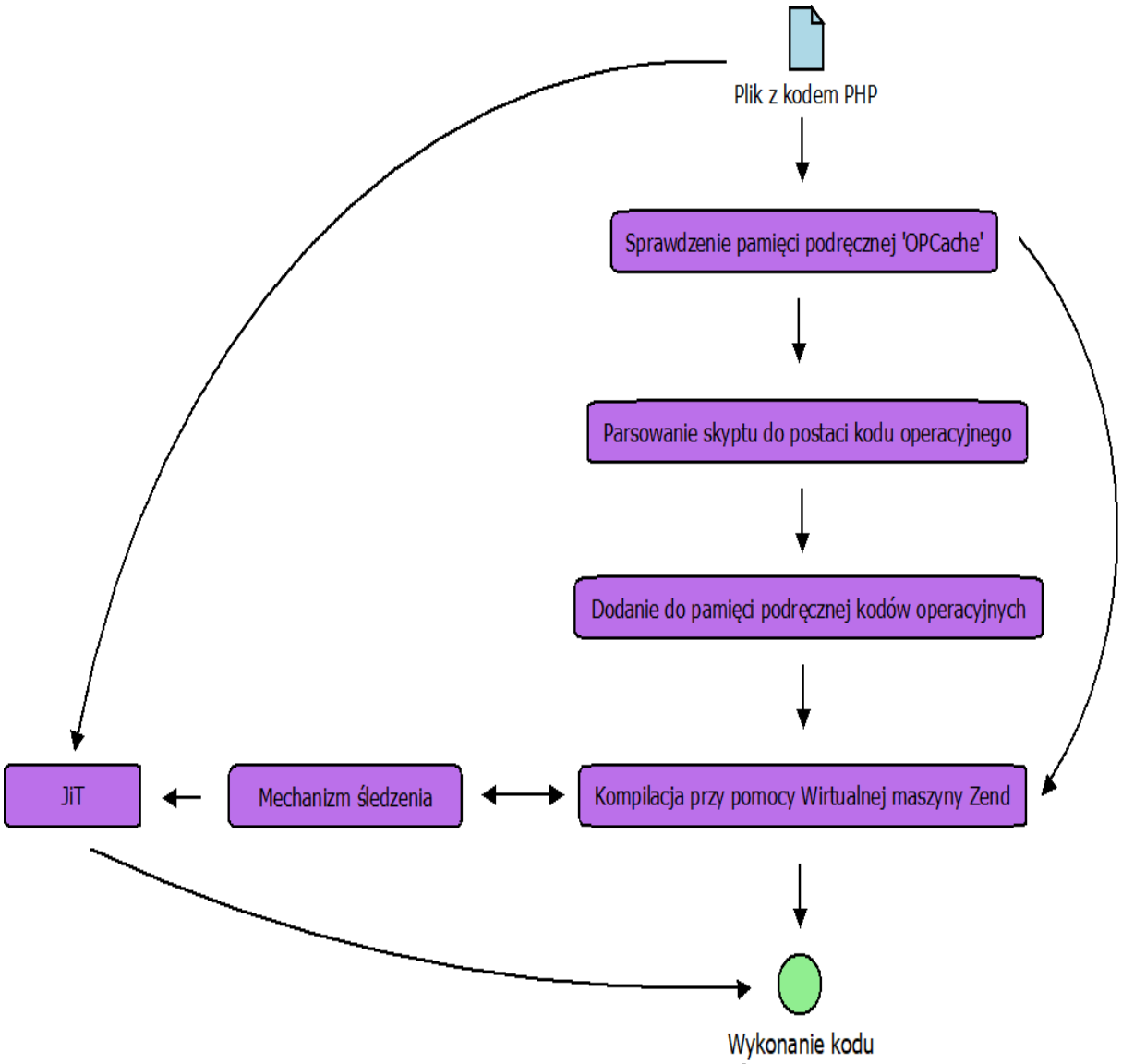
Rysunek 10. Drugi schemat przetwarzania kodu PHP. **Źródło**: [68].

Wersja PHP 5.5.0 pozostawiała jednak wciąż tak zwane wąskie gardło, czyli elementy, które znacząco spowalniały działanie kodu. Przykładowo współczesne frameworki PHP mają podatność na narzucanie obciążenia przez ładowanie wstępne takich mechanizmów jak: kontener wstrzykiwania zależności czy pobieranie konfiguracji. W celu poprawy tej sytuacji w wersji PHP 7.4 zostało wprowadzone ładowanie wstępnie pamięci podręcznej – ‘opcache preloading’. Rozwiązanie to polega na tym, że administrator ma możliwość wskazania plików z kodem PHP, które zostaną wcześniej wstępnie skompilowane do postaci kodów operacyjnych w pamięci podręcznej. Dzieje się to podczas uruchamiania kodu. Rysunek 11 przedstawia strukturę przetwarzania kodu z uwzględnieniem zmian, które doszły przy wypuszczeniu wersji PHP 7.4.



Rysunek 11. Trzeci schemat przetwarzania kodu PHP. **Źródło**: [68].

Ostatnie zmiany w procesie przetwarzania kodu PHP przyszły wraz z wersją 8.0.0. W wersji tej wprowadzono kompilacje JiT opierającą się na śledzeniu (ang. *JiT trace*) [70]. Opiera się ona na tworzeniu śladów, które zawierają aktualny stan instrukcji wirtualnej maszyny oraz takie informacje jak typy argumentów operatorów, spis klas znajdujących się w kontekście, czy wywołania metod, domknięć i funkcji. Mechanizm kompilacji JiT bada jak często i jak długo dana ścieżka jest wywoływana oraz jakiej funkcjonalności dostarcza. Na tej podstawie podejmuje decyzje co do tego czy kompilacja w tym przypadku przyniesie korzyści. W najnowszej wersji PHP na dzień pisania pracy schemat przetwarzania kodu wygląda tak jak na Rysunku 12.



Rysunek 12. Czwarty schemat przetwarzania kodu PHP. **Źródło**: [68].

Wyżej opisane schematy przetwarzania kodu pokazują, jak duży proces przebiegł podczas kształtowania się języka PHP. Nie wyglądał on od początku tak jak ma to miejsce teraz. Różne mechanizmy były kształtowane stopniowo, dzięki czemu efektywność PHP sukcesywnie rosła.

6. Praktyczny projekt oprogramowania wspomagającego zarządzanie przedsiębiorstwem

W niniejszym rozdziale przedstawiony zostanie cel dwóch projektów w postaci programów, które powstały jako część niniejszej pracy magisterskiej i dla których stworzenia zostały wykorzystane różne technologie, cechujące się podobieństwem w pewnych kwestiach, a w innych wykazujące się różnicami. Następnie zostaną przedstawione wymagania, które zostały przed nimi postawione oraz opisane zostaną technologie wykorzystane do stworzenia wspomnianych programów. W ostatnich rozdziałach znajduje się z kolei opis użytych platform programistycznych wraz z przedstawieniem praktycznej implementacji stworzonej w ich oparciu.

6.1. Cel projektu

W ramach projektu powstały dwie implementacje oprogramowania internetowego będące systemami, które wspierają zarządzanie przedsiębiorstwem. Korzystają one z relacyjnej bazy danych MySQL. Żeby dostarczyć wartościowego materiału do analizy, mają one taką samą funkcjonalność oraz zbliżony graficzny interfejs użytkownika. Współczesne podejście do wytwarzania oprogramowania internetowego wymaga do realizacji tego celu nowych sposobów i rozwiązań. Przykładem są tutaj platformy programistyczne (ang. *framework*) [71]. Pomagają one programiście w wytwarzaniu oprogramowania, dostarczając sprawdzonych, efektywnych i przetestowanych rozwiązań. Pozwala to na jego implementowanie w sposób szybszy i sprawniejszy, oparty na wzorcach, co z kolei ułatwia późniejsze utrzymywanie. Mając to na uwadze stworzone aplikacje w ramach projektu, zostały oparte na najpopularniejszych platformach programistycznych do wytwarzania oprogramowania internetowego dla języków, które ulegają analizie porównawczej w niniejszej pracy.

Stworzone projekty mają na celu ukazanie różnic i cech wspólnych pomiędzy implementacją internetowego oprogramowania w języku Java z użyciem frameworka Spring, a analogicznym oprogramowaniem napisanym w języku PHP i wykorzystującym framework Laravel. W obydwu przypadkach zostały one nazwane ‘PREMSS’, od inicjałów autora połączonych ze słowami pochodzącymi z języka angielskiego: ‘*Enterprise management support system*’, czyli Systemu wspomagającego zarządzanie przedsiębiorstwem.

6.2. Wymagania dla projektu

Jednym z najistotniejszych elementów procesu wytwarzania oprogramowania jest zebranie wymagań. Wykonanie tego kroku pozwala na jasne określenie oczekiwań co do oprogramowania, dzięki czemu z kolei można uniknąć efektu, gdzie wytworzony produkt rozmija się z tym czego potrzebuje użytkownik lub jego odbiorca.

W Tabeli 5 znajduje się lista wymagań dla oprogramowania realizowanego w ramach niniejszej pracy magisterskiej. Są one wspólne zarówno dla implementacji w języku Java jak i PHP.

Tabela 5. Spis wymagań dla implementacji projektu PREMSS w języku Java i PHP. **Źródło**: Opracowanie własne.

Id	Treść Wymagania	Rodzaj wymagania	Priorytet	Czy zrealizowane?
1	System powinien nazywać się 'PREMSS', co wiąże się z słowami 'Enterprise Management Support System'.	Wymaganie niefunkcjonalne	Wysoki	Tak
2	System powinien wspierać następujące przeglądarkę internetową: Google Chrome (od wersji 81.0.4044.129 64-bit).	Wymaganie funkcjonalne	Wysoki	Tak

3	Kolorystyka programu powinna być niebieska.	Wymaganie niefunkcjonalne	Średni	Tak
4	Dostęp do zasobów aplikacji powinien być ograniczony przez stronę do logowania, a autoryzacja powinna następować na podstawie nazwy użytkownika oraz hasła, gdzie przekierowanie do odpowiedniej części oprogramowania nastąpi na podstawie roli użytkownika.	Wymaganie niefunkcjonalne	Wysoki	Tak
4	Tło ekranu logowania powinno zawierać budynki biurowe, które kojarzyć się będą z pracą biurową.	Wymaganie niefunkcjonalne	Średni	Tak
5	Hasła użytkowników w bazie danych powinny być przetrzymywane w zakodowanej postaci.	Wymaganie funkcjonalne	Wysoki	Tak
6	System powinien umożliwiać po zalogowaniu użytkownikowi operację wylogowania się.	Wymaganie funkcjonalne	Wysoki	Tak
7	Wszystkie trwałe dane powinny być przechowywane w relacyjnej bazie danych - MySQL.	Wymaganie funkcjonalne	Wysoki	Tak
8	System powinien zostać zaimplementowany z użyciem jednych z najpopularniejszych technologii powiązanych z danym językiem.	Wymaganie niefunkcjonalne	Wysoki	Tak
9	W systemie powinni być rozróżnieni następujący użytkownicy: • Administrator, • Pracownik HR, • Pracownik standardowy.	Wymaganie funkcjonalne	Wysoki	Tak
10	Użytkownik o uprawnieniach Administrator powinien mieć dostęp do modułu: • Moje notatki, • Użytkownicy, • Lista pracowników.	Wymaganie funkcjonalne	Wysoki	Tak
11	Użytkownik o uprawnieniach Pracownik HR powinien mieć dostęp do modułu: • Moje notatki, • Lista Pracowników, • Lista wynagrodzeń.	Wymaganie funkcjonalne	Wysoki	Tak
12	Użytkownik o uprawnieniach Pracownik HR powinien mieć dostęp do modułu: • Moje notatki, • Produkty, • Magazyn, • Lista pracowników.	Wymaganie funkcjonalne	Wysoki	Tak
13	System powinien zawierać moduł 'Moje notatki'.	Wymaganie funkcjonalne	Wysoki	Tak
14	Użytkownicy o każdej roli dostępnej w systemie, powinni mieć możliwość definiowania i zmieniania notatek.	Wymaganie funkcjonalne	Wysoki	Tak

15	Do każdego użytkownika systemu powinna być przypisana tylko jedna notatka, gdzie podgląd i edycja tej notatki będzie możliwa tylko przez tego użytkownika.	Wymaganie funkcjonalne	Wysoki	Tak
16	System powinien zawierać moduł 'Lista pracowników'.	Wymaganie funkcjonalne	Wysoki	Tak
17	W module 'Lista pracowników' powinna znajdować się lista wszystkich zatrudnionych pracowników w organizacji.	Wymaganie funkcjonalne	Wysoki	Tak
18	W module 'Lista pracowników' powinny znajdować się dane pracownika w postaci: • imienia, • nazwiska, • numeru telefonu, • adresu e-mail, • działu (jednostki organizacyjnej).	Wymaganie funkcjonalne	Wysoki	Tak
19	Każdy z użytkowników zdefiniowanych w systemie powinien mieć możliwość podglądu zdefiniowanych pracowników.	Wymaganie funkcjonalne	Wysoki	Tak
20	Użytkownik o roli Pracownik HR powinien mieć możliwość dodawania nowych pracowników oraz edytowania i usuwania istniejących.	Wymaganie funkcjonalne	Wysoki	Tak
21	Dodawanie i edycja pracowników powinno odbywać się w oddzielnym widoku.	Wymaganie funkcjonalne	Wysoki	Tak
22	System powinien zawierać moduł 'Użytkownicy'.	Wymaganie funkcjonalne	Wysoki	Tak
23	W module 'Użytkownicy' powinna znajdować się lista wszystkich użytkowników zdefiniowanych w systemie.	Wymaganie funkcjonalne	Wysoki	Tak
24	W module 'Użytkownicy' powinny znajdować się dane użytkownika w postaci: • nazwy, • hasła, • roli, • imienia i nazwiska pracownika.	Wymaganie funkcjonalne	Wysoki	Tak
25	Użytkownik o roli Administrator powinien mieć możliwość podglądu zdefiniowanych użytkowników, dodawania nowych użytkowników oraz edytowania i usuwania istniejących.	Wymaganie funkcjonalne	Wysoki	Tak
26	Dodawanie i edycja użytkowników powinno odbywać się w oddzielnym oknie.	Wymaganie funkcjonalne	Wysoki	Tak
27	Okno do dodawania i edycji użytkowników powinno zawierać wyszukiwarkę pracowników.	Wymaganie funkcjonalne	Wysoki	Tak
28	System powinien zawierać moduł 'Produkty'.	Wymaganie funkcjonalne	Wysoki	Tak
29	W module 'Produkty' powinna znajdować się lista wszystkich produktów zdefiniowanych w systemie.	Wymaganie funkcjonalne	Wysoki	Tak
30	W module 'Produkty' powinny znajdować się dane produktów w postaci: • nazwa produktu,	Wymaganie funkcjonalne	Wysoki	Tak

	• dostępności, • ceny.			
31	Użytkownik o roli Pracownik standardowy powinien mieć możliwość podglądu zdefiniowanych produktów, dodawania nowych produktów oraz edytowania i usuwania istniejących	Wymaganie funkcjonalne	Wysoki	Tak
32	Dodawanie i edycja użytkowników powinno odbywać się w oddzielnym oknie.	Wymaganie funkcjonalne	Wysoki	Tak
33	System powinien zawierać moduł 'Magazyn'.	Wymaganie funkcjonalne	Wysoki	Tak
34	W module 'Magazyn' powinna znajdować się lista wszystkich pozycji magazynowych.	Wymaganie funkcjonalne	Wysoki	Tak
35	W module 'Magazyn' powinny znajdować się dane magazynowe w postaci: • nazwy produktu, • ilości dostępnych sztuk, • informacji czy produkt został zamówiony, • miejsca składowania.	Wymaganie funkcjonalne	Wysoki	Tak
36	Użytkownik o roli Pracownik standardowy powinien mieć możliwość podglądu zdefiniowanych pozycji magazynowych, dodawania nowych pozycji magazynowych oraz edytowania i usuwania istniejących.	Wymaganie funkcjonalne	Wysoki	Tak
37	Dodawanie i edycja pozycji magazynowych powinno odbywać się w oddzielnym oknie.	Wymaganie funkcjonalne	Wysoki	Tak
38	Okno do dodawania i edycji pozycji magazynowych powinno zawierać wyszukiwarkę produktów.	Wymaganie funkcjonalne	Wysoki	Tak
39	System powinien zawierać moduł 'Lista wynagrodzeń'.	Wymaganie funkcjonalne	Wysoki	Tak
40	W module 'Lista wynagrodzeń' powinna znajdować się lista wszystkich pozycji z wynagrodzeniami zdefiniowanych w systemie.	Wymaganie funkcjonalne	Wysoki	Tak
41	W module 'Lista wynagrodzeń' powinny znajdować się dane w postaci pozycji z wynagrodzeniami, zawierającymi poniższe informacje: • wynagrodzenie brutto, • wynagrodzenie netto, • numer konta, • imię i nazwisko pracownika.	Wymaganie funkcjonalne	Wysoki	Tak
42	Użytkownik o roli Pracownik HR powinien mieć możliwość podglądu zdefiniowanych pozycji z zarobkami, dodawania nowych oraz edytowania i usuwania istniejących.	Wymaganie funkcjonalne	Wysoki	Tak
43	Dodawanie i edycja pozycji z zarobkami powinno odbywać się w oddzielnym oknie.	Wymaganie funkcjonalne	Wysoki	Tak
44	Okno do dodawania i edycji pozycji z zarobkami powinno zawierać wyszukiwarkę pracowników.	Wymaganie funkcjonalne	Wysoki	Tak

45	Powinny istnieć kopie zapasowe źródeł systemu.	Wymaganie niefunkcjonalne	Wysoki	Tak

Wszystkie powyższe wymagania zostały pomyślnie zrealizowane, określają one cechy i funkcjonalności implementacji aplikacji w języku Java oraz frameworku Spring i drugiej implementacji w języku PHP oraz frameworku Laravel.

Niżej przedstawiony kod wraz z opisem mechanizmów oraz zrzuty ekranu przedstawiają według autora niniejszej pracy kluczowe fragmenty implementacji. Pozostały kod oraz widoki nie są według jego opinii kluczowe do przedstawienia schematów i technologii, ponieważ będą one opierać się na podobnych rozwiązaniach, jak te już zaprezentowane i opisane.

6.3. Zastosowane technologie

W obydwu wersjach implementacji po stronie klienckiej zostały zastosowane następujące technologie:

- HTML,
- CSS,
- Bootstrap,
- JavaScript,
- jQuery oraz jQuery UI.

W rozdziale 2.3 poświęconemu programowaniu internetowemu opisane zostały języki HTML, CSS i JavaScript. Bootstrap jest to platforma programistyczna oparta o wspomniane wcześniej języki, która dostarcza narzędzi do tworzenia responsywnych stron internetowych. Zawiera gotowe rozwiązania oraz style CSS, które ułatwiają tworzenie graficznego interfejsu użytkownika. jQuery to natomiast biblioteka programistyczna związana z językiem JavaScript. Pomaga ona w operacjach przetwarzania dokumentów HTML, obsłudze różnego rodzaju zdarzeń, czy też wysyłaniu zapytań HTTP i następnie obsłudze odpowiedzi. jQuery UI to rozwiązanie zbudowane na bazie jQuery, które zawiera mechanizmy do interakcji z graficznym interfejsem użytkownika, różnego rodzaju widżety oraz efekty [72]. Dostarcza również narzędzi, które wspierają tworzenie formularzy.

W przypadku strony serwerowej w jednej wersji implementacji wykorzystany został język Java, wraz z frameworkiem Spring, a w przypadku drugiej użyty został język PHP i framework Laravel. Do stworzenia kodu w tym pierwszym zostało użyte zintegrowane środowisko programistyczne IntelliJ Idea [73]. W przypadku drugiego wykorzystany został edytor Visual Studio Code [74], gdzie zainstalowana została między innymi wtyczka ‘PHP Debug’ [75] ułatwiająca proces debugowania kodu. Z użyciem wspomnianych wyżej platform programistycznych powstała również część strony klienckiej, jako iż są one także wyposażone w funkcjonalności ułatwiające tworzenie tej części oprogramowania.

Do przechowywania danych użyty został system do zarządzania relacyjnymi bazami danych MySQL. W oparciu o niego zostały stworzone dwie bazy o nazwie ‘premss_java_db’ dla implementacji w oparciu o język Java i framework Spring, oraz ‘premss_php_db’ przeznaczona dla programu napisanego w języku PHP z wykorzystaniem platformy programistycznej Laravel. Zastosowanie w projekcie znalazło również mapowanie relacyjnego obiektowe - dla Javy był to Hibernate a dla PHP Eloquent ORM. W jednym i drugim przypadku wykorzystany został silnik InnoDB.

6.4. Implementacja aplikacji webowej w języku Java z wykorzystaniem frameworka Spring

W ramach następującego rozdziału omówiony został framework Spring wraz z opisem jego najważniejszych modułów. Przedstawiona została również implementacja projektu, która posłużyła do ich zaprezentowania.

6.4.1 Wprowadzenie do frameworka Spring

Na początku należałoby omówić czym jest Spring. Spring jest to cała rodzina projektów związanych z projektem Spring, u których podstaw leży Platforma programistyczna Spring.

Platforma programistyczna Spring jest to framework dla Platformy Java, możliwie jest jednak implementowanie z jego udziałem programów również z wykorzystaniem takich języków jak Kotlin czy Groovy. Jego początki sięgają roku 2003. Ułatwia on tworzenie aplikacji i wspiera między innymi budowanie tych, których skala jest bardzo duża. Ma modułową budowę, gdzie to programista może określić, które z modułów są mu potrzebne. Jego kluczowymi komponentami są te związane z głównym kontenerem, pośród których można wyróżnić model konfiguracji oraz mechanizm związany z wstrzykiwaniem zależności [76].

Moduły zorganizowane są w następujące grupy [77]:

- **Core Container** – Kontener ten jest podstawowym modułem Springa, dostarcza on funkcjonalności związanych z wzorcem odwrócenia sterowania (IoC) oraz wstrzykiwania zależności (DI). Składa się również z implementacji wzorca fabryki dla Springa w postaci BeanFactor. Ten natomiast związany jest z komponentami Springa, które nazywane są Spring Beanami. Są to specjalne klasy Springowe. Te ostatnie przechowywane są przez Kontener Springowy. Core container obejmuje, również kontekst, dzięki któremu klasy źródłowe są przeszukiwane i odnalezione Spring Beany umieszczane są w kontenerze Springa.

- **Web** – Framework dostarcza licznych modułów ściśle związanych z wytwarzaniem oprogramowania internetowego. Zawiera mechanizmy, które zapewniają integracje sieciową, inicjalizację kontenera IoC z wykorzystaniem Servletów, kontekst zorientowany sieciowo czy funkcjonalności takie, które są przykładowo związane są z przesyłaniem plików czy klientem HTTP. Wspomniane Servlety są to klasy związane z stroną serwerową oprogramowania internetowego i oparte są na żądaniach i odpowiedziach HTTP – pojęcie to jest ściśle związane z językiem Java. Dodatkowo Spring wspiera wytwarzania oprogramowania webowego opartego o wzorzec MVC, przez dostarczenie modułu ‘spring-webmvc’.

- **Data Access/Integration** – Spring zapewnia wsparcie dla zarządzania danymi oraz integracji z różnymi bazami danych. Można tutaj wyróżnić moduł JDBC odpowiedzialny za łączenie się z bazą danych i kierowanie do niej zapytań przy pomocy języka SQL. Wspierany jest standard ORM JPA, wraz z takimi jego implementacjami jak przykładowo Hibernate. Dostarczony jest również moduł ‘spring-oxm’, który dostarcza warstwy abstrakcji dla mapowania pomiędzy dokumentami XML a obiektami [78]. Spring wspomaga również technologii Java Mesasaging Service, przez doostarczennie modułu ‘spring-jms’

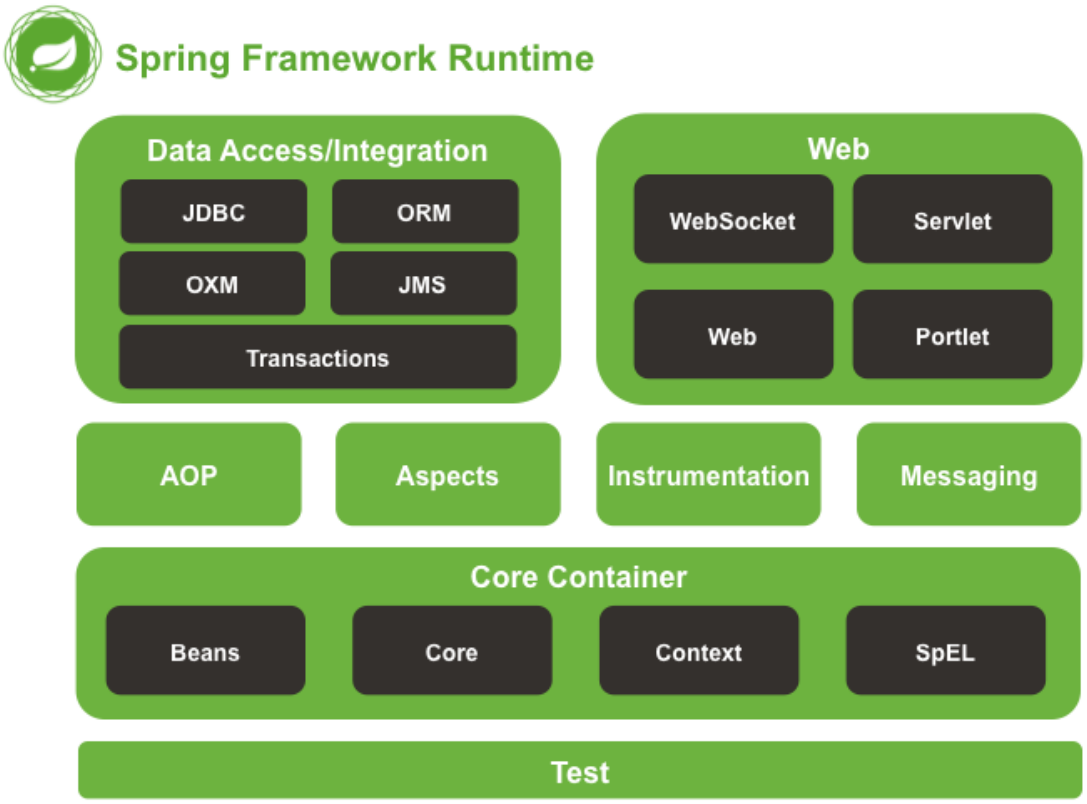
- **Messaging** – Moduł ten związany jest z dostarczeniem podstaw dla programów ukierunkowanych na pracę z wiadomościami oraz interfejsów programistycznych aplikacji oraz protokołów z nimi związanych.

- **AOP (Aspect Oriented Progamming)** – Spring wspiera programowanie ukierunkowane aspektowo. Jest to paradygmat programowania, który związany jest z specyficznym podejściem do struktury programu. Podczas gdy w przypadku programowania zorientowanego obiektowo, podstawową jednostką modułów są klasy, tak w tym przypadku są nią aspekty. Dzięki ich zastosowaniu podzieleniu na moduły mogą ulegać takie zagadnienia jak zarządzanie transakcjami. [79].

- **Instrumentation** – Moduł ‘spring-instrument’ związany jest między innymi z interfejsem Instrumentation [80], który dostarcza możliwości dodawania kodu bajtowego do metod, na których podstawie zostaną stworzone dane do późniejszej analizy przez różnego rodzaju narzędzia. Kluczowe jest to, że dodany kod nie modyfikuje zachowania ani stanów programu. Przykładami wspomnianych narzędzi są takie, które służą do profilowania kodu, logowania zdarzeń, badania pokrycia kodu czy agenci do monitorowania [81].

- **Test** – Spring wspiera tworzenie testów jednostkowych oraz testów integracyjnych dla Springowych komponentów. Używane są do tego takie narzędzia jak JUnit oraz TestNG. Dostarcza on również atrap obiektów (ang. *mock objects*), które są pomocne przy izolowaniu od testowanego kodu.

Umiejscowienie wyżej wymienionych i opisanych modułów w platformie Spring przedstawia Rysunek 13.



Rysunek 13. Komponenty Frameworka Spring. **Źródło**: [77].

Oficjalna strona platformy programistycznej Spring informuje, iż zalecane jest korzystanie z narzędzia do budowania oprogramowania, które wspiera zarządzanie zależnościami, znajdującymi się w repozytorium Mavena. Istnieje możliwość skorzystania z większej ilości systemów budowania, jednak w dokumentacji można znaleźć informację, iż zalecane jest do tego celu rozwiązanie Maven lub Gradle [82]. Ostatnim wydaniem frameworka Spring jest wersja 5.3.15 z datą 13.01.2022 [83].

Spring boot jest natomiast rozszerzeniem Springa [84], dzięki któremu tworzenie aplikacji Spring jest szybsze, a programista zwolniony jest z części pracy konfiguracyjnej. Bazuje on na frameworku Spring i jest rozwiązaniem opartym o strategię ‘konwencja ponad konfiguracje’ (ang. *convention over configuration*) [85]. Oznacza to, że proces konfigurowania aplikacji po stronie jej twórcy i co za tym idzie pisania kodu z tym związanego sprowadzany jest do minimum, co przyśpiesza

pracę nad tworzeniem oprogramowania. Obecnie najnowszą wersją Spring Boota jest 2.6.3, gdzie jej wypuszczenie datowane jest na dzień 20.01.2022 [86].

Z Spring Bootem wiąże się narzędzie Spring Initializr, które jest oficjalnym rozwiązaniem służącym do stworzenia szkieletu projektu Spring Boot [87]. Programista definiuje w nim takie elementy, jak:

- Narzędzie do budowania (czy jest to projekt oparty o Maven czy Gradle),
- Język programowania (do wyboru jest Java, Groovy oraz Kotlin),
- Wersję Spring Boota,
- Wersję Javy,
- Metadane dotyczące projektu (przykładowo jest to tytuł projektu, opis czy nazwa pakietu).

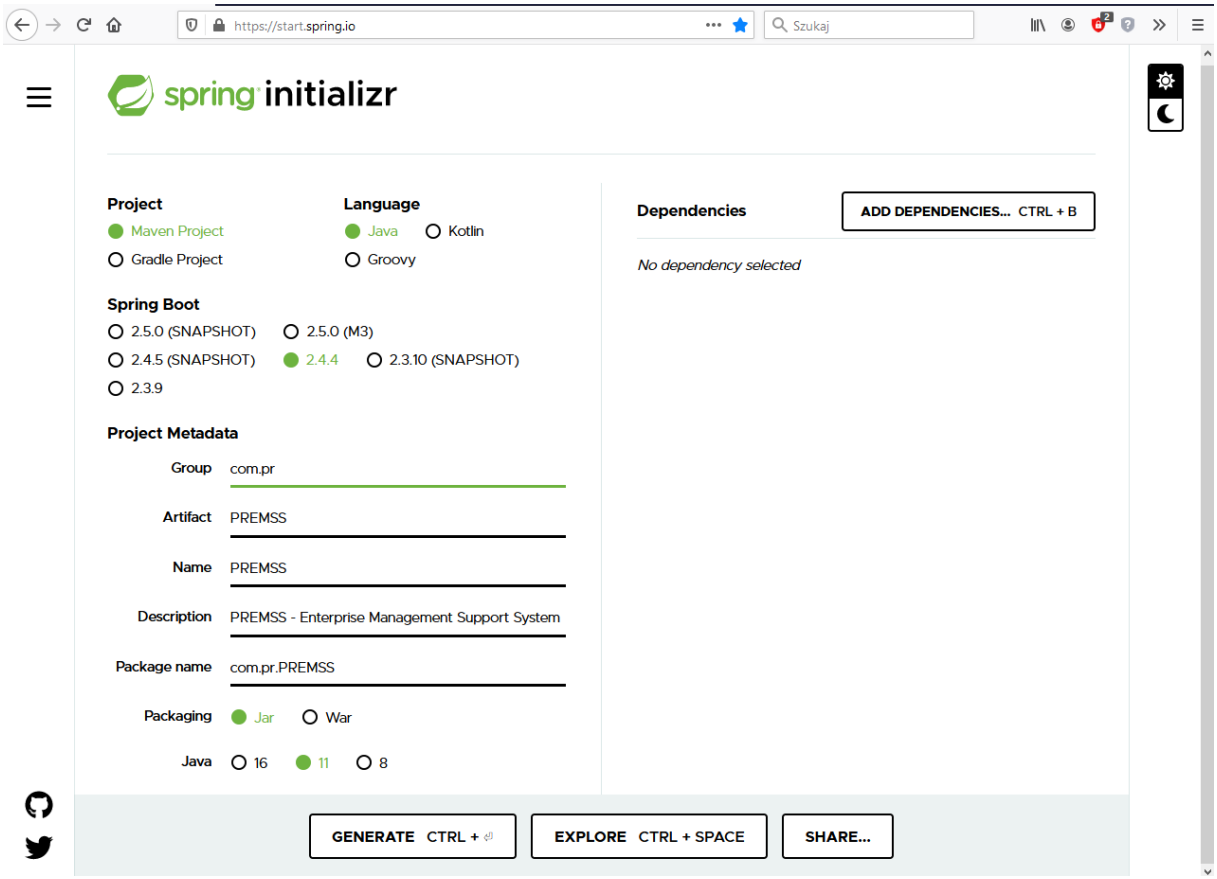
Dodatkowo, jeśli programista już na etapie tworzenia szkieletu projektu jest w stanie określić jakie dodatkowe zależności są mu potrzebne, to istnieje możliwość ich wyznaczenia i dodania przy pomocy rozwiązania Spring Initializr.

6.4.2 Praktyczna implementacja oprogramowania w oparciu o język Java oraz framework Spring

Następujące podrozdziały poświęcone są implementacji oprogramowania internetowego opartej o framework Spring, która powstała w ramach niniejszej pracy magisterskiej.

6.4.2.1. Inicjowanie projektu Java i Spring i zarządzanie zależnościami

Do stworzenia schematu projektu i jego zainicjalizowania wykorzystane zostało wcześniej opisane narzędzie Spring Initializr. Na tym etapie nie dodano żadnych dodatkowych zależności. Zrzut ekranu widoczny na Rysunku 14, prezentuje skonfigurowanie szkieletu projektu zaraz przed wygenerowaniem paczki. Pochodzi on z internetowej wersji narzędzia.



Rysunek 14. Tworzenie szkieletu projektu PREMSS – Java i Spring (Zrzut ekranu z oficjalnej strony Spring Initialzr). **Źródło**: [88].

Jak widać na Rysunku numer 14, w ramach niniejszego oprogramowania został wykorzystany Maven. Zarządzanie zależnościami odbywa się tutaj przez plik XML o nazwie ‘pom.xml’. Programista chcąc dodać nową zależność określa ją w znaczniku ‘<dependencies>’, przez umieszczenie w nim tagu ‘<dependency>’ w którym znajdują się odwołania do konkretnej zależności. W Kodzie 2, zaprezentowały został plik z końcowymi zależnościami dla projektu PREMSS.

Kod 2. Plik pom.xml dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
<?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">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.4.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.pr</groupId>
   <artifactId>PREMSS</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>PREMSS</name>
   <description>PREMSS - Enterprise Management Support System</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!-- Additional dependencies added manually
             after package generation by SpringInitializr -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>jquery</artifactId>
            <version>3.2.1</version>
        </dependency>
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>bootstrap</artifactId>
            <version>3.3.7-1</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
   <packaging>war</packaging>
</project>
```

6.4.2.2. Autentykacja i Logowanie - Java i Spring

Pierwszą stroną zaimplementowanej aplikacji w ramach niniejszej pracy i zarazem tą, którą widzi użytkownik na początku jest formularz logowania. Wygląda on tak jak na zrzucie ekranu w postaci Rysunku 15. Omawiany framework posiada wsparcie, pomagające w implementowaniu mechanizmów autentykacji i logowania. Związana jest z nim zależność ‘spring-boot-starter-security’, która widoczna jest w Kodzie 2. W oprogramowaniu PREMSS ze wspomnianymi funkcjonalnościami związane są między innymi następujące pliki:

- Plik ‘SecurityConfiguration.java’- Klasa Javy, w której określona została konfiguracja związana z bezpieczeństwem.

  W metodzie ‘`configure`’ zdefiniowane zostały dostępy do poszczególnych modułów przez użytkowników o rolach Administrator, Standardowy pracownik i Pracownik HR. Znajduje się tutaj również powiązanie z formularzem logowania oraz obsługą w przypadku pomyślnego logowania przez metodę ‘`PremssAuthenticationSuccessHandlerMethod`’. Wspomniana wcześniej metoda zwraca obiekt klasy ‘`PremssAuthenticationSuccessHanlder`’. Metoda ‘`securityUsers`’ określa użytkowników wraz z przypisanymi do nich rolami. Metoda ‘`paswordEncoder`’ jest natomiast związana ze sposobem haszowania haseł – zostało tutaj wykorzystane kodowanie ‘bcrypt’. Kod

3 zawiera źródła opisanego pliku. Znajdują się w nim komentarze ‘`Authentication based on DB`’ dla sekcji związanej z autentykacją na podstawie informacji z bazy danych. Dodatkowo w Kodzie 4 zawarty został alternatywny kod z tego samego pliku, który został zakomentowany w finalnej wersji programu. Widoczne są w nim komentarze ‘`Authentication in memory`’ dla wersji, z autentykacją w pamięci. Ta ostatnia opcja wiąże się z tym, iż Spring dostarcza głównie do celów deweloperskich możliwość stworzenia mechanizmów autentykacji, jeszcze przed stworzeniem bazy danych. Autor niniejszej pracy i implementacji, wykorzystał tę opcję i postanowił przedstawić zakomentowany kod źródłowy z nią związany, ponieważ ukazuje to możliwości Springa oraz procesu deweloperskiego z wykorzystaniem tej technologii.

Kod 3. Plik SecurityConfiguration.java dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
//(…)
@Configuration
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Autowired
    DataSource sourceOfData;

    @Override
    public void configure(HttpSecurity security) throws Exception {
        //Authentication based on DB:
        security.authorizeRequests()
                .antMatchers("/graphics/**" ,"/css/**").permitAll()
                .antMatchers("/HREmpMainPage").hasAnyAuthority(
                            "hr_employee","admin")
                .antMatchers("/StandardEmpMainPage").hasAnyAuthority(
                            "standard_employee", "admin")
                .antMatchers("/AdmMainPage").hasAnyAuthority("admin")
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login").permitAll()
                .loginProcessingUrl("/login")
                .failureUrl("/login?error=true")
               .successHandler(PremssAuthenticationSuccessHandlerMethod());
    }

    @Autowired
    public void securityUsers(AuthenticationManagerBuilder auth)
    throws Exception
    {
        //Authentication based on DB:
        auth.jdbcAuthentication()
                .dataSource(sourceOfData)
                .usersByUsernameQuery("SELECT name, password, enabled FROM
                                       users WHERE name = ? ")
                .authoritiesByUsernameQuery("SELECT name, role FROM users
                                            WHERE name = ?");
    }
```

```
    @Bean
    public AuthenticationSuccessHandler
    PremssAuthenticationSuccessHandlerMethod(){
        return new PremssAuthenticationSuccessHanlder();
    }

    //Authentication based on DB:
    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }

}
```

Kod 4. Plik SecurityConfiguration.java dla projektu PREMSS i alternatywny sposób autentykacji – Java i Spring. **Źródło**: Opracowanie własne.

```
//(…)
@Configuration
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Autowired
    DataSource sourceOfData;

    @Override
    public void configure(HttpSecurity security) throws Exception {
        //Authentication in memory:
       security.authorizeRequests()
               .antMatchers("/graphics/**" ,"/css/**").permitAll()
               .antMatchers("/HREmpMainPage").hasAnyRole("HrEmployee",
            "Administrator")
               .antMatchers("/StandardEmpMainPage").hasAnyRole(
            "StandardEmployee", "Administrator")
               .antMatchers("/AdmMainPage").hasAnyRole("Administrator")
               .anyRequest().authenticated()
               .and()
               .formLogin()
               .loginPage("/login").permitAll()
               .loginProcessingUrl("/login")
               .failureUrl("/login?error=true")
               .successHandler(PremssAuthenticationSuccessHandlerMethod());

    }

    @Autowired
    public void securityUsers(AuthenticationManagerBuilder auth)
    throws Exception
    {
        //Authentication in memory:
```

```
        auth.inMemoryAuthentication()
                .withUser("user1").password("{noop}user1")
                .roles("Administrator")
                .and()
                .withUser("user2").password("{noop}user2")
                .roles("StandardEmployee")
                .and()
                .withUser("user3").password("{noop}user3")
                .roles("HrEmployee");

    }

}
```

- Plik ‘PremsAuthenicationSuccessHandler.java’ – W klasie tej zostało określone zachowanie w reakcji na zdarzenie w postaci pomyślnego zalogowania się. Znajduje się tutaj między innymi kod odpowiedzialny za przekierowanie na odpowiednią ścieżkę, w zależności od roli użytkownika (Administrator, Pracownik standardowy oraz Pracownik HR). Instancja tej klasy znalazła swoje zastosowanie w wcześniej opisanej klasie konfiguracyjnej i podobnie jak w jej przypadku, znajdują się tutaj komentarze ‘`Authentication based on DB`’ w Kodzie 5 dla autentykacji na podstawie bazy danych oraz ‘`Authentication in memory`’ w Kodzie 6 związane z alternatywnym sposobem autentykacji opierającej się na informacjach z pamięci.

Kod 5. Plik PremsAuthenicationSuccessHandler.java dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
//(…)
public class PremssAuthenticationSuccessHanlder implements
AuthenticationSuccessHandler {

    private RedirectStrategy redirectStrategy =
    new DefaultRedirectStrategy();

    boolean hasAdministratorRole = false;
    boolean hasStandardEmployeeRole = false;
    boolean hasHrEmployeeRole = false;

    @Override
    public void onAuthenticationSuccess(HttpServletRequest arg0,
                                        HttpServletResponse arg1,
                                        Authentication authentication)
    throws IOException, ServletException
    {
        //Clear role values:
        hasAdministratorRole = false;
        hasStandardEmployeeRole = false;
        hasHrEmployeeRole = false;

        Collection<? extends GrantedAuthority> authorities =
        authentication.getAuthorities();
```

```
        for (GrantedAuthority grantedAuthority : authorities) {

            //Authentication based on DB:
            if (grantedAuthority.getAuthority().equals("admin")) {
                hasAdministratorRole = true;
                break;
            } else if (grantedAuthority.getAuthority().equals(
                        "standard_employee")) {
                hasStandardEmployeeRole = true;
                break;
            } else if (grantedAuthority.getAuthority().equals(
                        "hr_employee")) {
                hasHrEmployeeRole = true;
                break;
            }
        }

        //Authentication based on DB:
        if (hasAdministratorRole) {
            redirectStrategy.sendRedirect(arg0, arg1, "/AdmMainPage");
        } else if (hasStandardEmployeeRole) {
            redirectStrategy.sendRedirect(arg0, arg1,
                                          "/StandardEmpMainPage");
        } else if (hasHrEmployeeRole) {
            redirectStrategy.sendRedirect(arg0, arg1, "/HREmpMainPage");
        } else {
            throw new IllegalStateException();
        }

    }
}
```

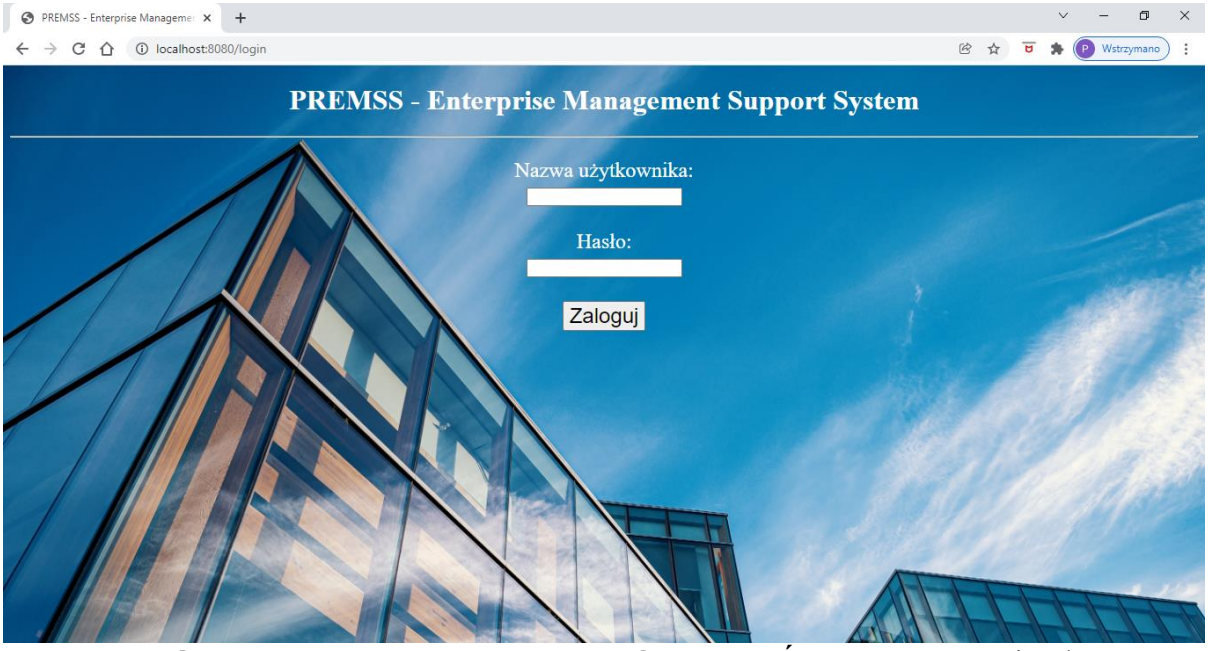
Kod 6. Plik PremsAuthenicationSuccessHandler.java dla projektu PREMSS i alternatywny sposób autentykacji – Java i Spring. **Źródło**: Opracowanie własne.

```
//(…)
public class PremssAuthenticationSuccessHanlder implements
AuthenticationSuccessHandler {

    private RedirectStrategy redirectStrategy =
    new DefaultRedirectStrategy();

    boolean hasAdministratorRole = false;
    boolean hasStandardEmployeeRole = false;
    boolean hasHrEmployeeRole = false;

    @Override
    public void onAuthenticationSuccess(HttpServletRequest arg0,
                                        HttpServletResponse arg1,
                                        Authentication authentication)
```

```
    throws IOException, ServletException
    {
        //Clear role values:
        hasAdministratorRole = false;
        hasStandardEmployeeRole = false;
        hasHrEmployeeRole = false;

        Collection<? extends GrantedAuthority> authorities =
        authentication.getAuthorities();

        for (GrantedAuthority grantedAuthority : authorities) {

            //Authentication in memory:
           if (grantedAuthority.getAuthority().equals(
           "ROLE_Administrator")) {
               hasAdministratorRole = true;
               break;
           } else if (grantedAuthority.getAuthority()
                       .equals("ROLE_StandardEmployee")) {
               hasStandardEmployeeRole = true;
               break;
           } else if (grantedAuthority.getAuthority()
                       .equals("ROLE_HrEmployee")) {
               hasHrEmployeeRole = true;
               break;
           }

        }

        //Authentication in memory:
       if (hasAdministratorRole) {
           redirectStrategy.sendRedirect(arg0, arg1, "/AdmMainPage");
       } else if (hasStandardEmployeeRole) {
           redirectStrategy.sendRedirect(arg0, arg1,
                                         "/StandardEmpMainPage");
       } else if (hasHrEmployeeRole) {
           redirectStrategy.sendRedirect(arg0, arg1, "/HREmpMainPage");
       } else {
           throw new IllegalStateException();
       }

    }
}
```

- Plik ‘LoginPage.html’ – Tutaj zdefiniowany jest widok formularza logowania.
- Plik ‘Main PremssController.java’ – W kontrolerze tym znajdują się metody związane z obsługą ścieżek dla trzech ról zdefiniowanych w oprogramowaniu oraz dla logowania i wylogowywania.

Widoki i kontrolery zostały szerzej omówione przy okazji opisywania wsparcia dla wzorca MVC w dalszej części rozdziału.

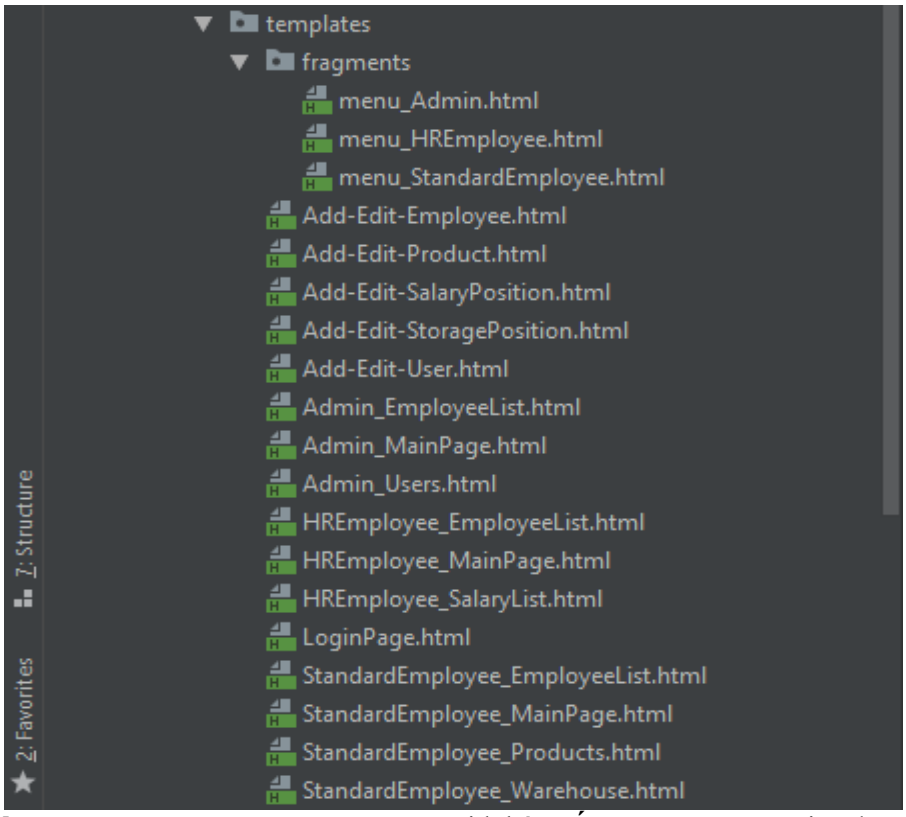


Rysunek 15. PREMSS, Java i Spring – Strona logowania. **Źródło**: Opracowanie własne.

6.4.2.3. MVC Java i Spring - Widok

Niniejsze oprogramowanie, powstało bazując na wzorcu projektowym MVC, który jak wiadomo ze wcześniejszej części pracy jest wspierany przez Springa. Poniżej opisana zostanie jego realizacja na podstawie implementacji dwóch modułów określonych w wymaganiach z rozdziału 6.2, czyli ‘modułu Produkty’ oraz ‘modułu Pozycje magazynowe’.

Widok (ang. *view*) – Lista wszystkich widoków, które powstały w ramach opisywanej tutaj implementacji widoczna jest na zrzucie ekranu w postaci Rysunku 16.



Rysunek 16. PREMSS, Java i Spring – Lista widoków. **Źródło**: Opracowanie własne.

W przypadku używania frameworka Spring, istnieje możliwość skorzystania z silnika szablonów Thymeleaf, który pozwala na użycie zarówno kodu HTML jak i tego związanego z Javą. Wykorzystanie takiego rozwiązania pomaga we współpracy z kodem napisanym w języku Java, między innymi z takimi komponentami jak modele czy kontrolery. Znalazł on zastosowanie w implementacji widoków dla oprogramowania PREMSS.

Po zalogowaniu się na konto Standardowego pracownika i przejściu do zakładki Produkty, użytkownik widzi listę produktów – efekt widoczny jest na Rysunku 17. Za widok ten odpowiedzialny jest plik ‘StandardEmployee_Products.html’. Źródła dla wspomnianego widoku przedstawione zostały jako Kod 7.

Kod 7. Plik StandardEmployee_Products.html dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta http-equiv="Content-Type" content="text/html"; charset="UTF-8" />
    <title>PREMSS - Enterprise Management Support System</title>

    <link rel="stylesheet" th:href="@{/webjars/bootstrap/3.3.7-1/
                                    css/bootstrap.min.css}" />
    <link rel="stylesheet" type="text/css" th:href="@{/css/shared.css}"/>

    <script th:src="@{/webjars/jquery/3.2.1/jquery.min.js}"></script>
    <script th:src="@{/webjars/bootstrap/3.3.7-1/
                    js/bootstrap.min.js}"></script>
</head>
<body>
```

```
<h1 style="text-align: center;" class="premssHeader">
PREMSS - Enterprise Management Support System</h1>
<hr>

<!-- Menu Standard Employee PREMSS - Thymeleaf Fragment -->
<div th:replace="~{fragments/menu_StandardEmployee :: menuFragment_Standard
Employee}"></div>
<!-- Menu Standard Employee PREMSS - Thymeleaf Fragment -->

<!-- Product list - Start -->
<div th:switch="${productList}" class="container my-5">

    <p class="my-5" align="right">
        <a href="@{/edit_product}" class="btn btn-primary">
            <i class="fas fa-user-plus ml-2"> Dodaj nowy produkt </i>
        </a>
    </p>

    <div class="col-md-10">
        <h2 th:case="null">Brak Produktów.</h2>
        <div th:case="*">
            <table class="table table-striped table-responsive-
                          md premssDataTable">
                <thead>
                <tr>
                    <th class="premssHeadlineText">Nazwa produktu</th>
                    <th class="premssHeadlineText">Dostępność</th>
                    <th class="premssHeadlineText">Cena</th>
                </tr>
                </thead>
                <tbody>
                <tr th:each="product : ${productList}">
                    <td th:text="${product.name}"></td>
                    <td th:text="${product.availability} ? 'Tak' : 'Nie'">
                    </td>
                    <td th:text="${product.price}"></td>
                    <td>
                       <a th:href="@{/edit_product/{id}(id=${product.id})}"
                          class="btn btn-primary">
                            <i>Edytuj</i>
                       </a>
                    </td>
                    <td>
                        <a th:href="@{/delete_product/{id}(id=${product.id}
                                    )}" class="btn btn-danger">
                            <i>Usuń</i>
                        </a>
                    </td>
                </tr>
```

```
                </tbody>
            </table>
        </div>

    </div>
</div>
<!-- Product list - End -->

</body>
</html>
```

Na przykładzie pokazanym jako Kod 5 widać również, że w nagłówku zostały dołączone style CSS i plik JavaScript związany z Bootstrapem oraz biblioteka jQuery. Wersja tych komponentów została dostarczona wraz z dodaniem zależności z grupy ‘org.webjars’ dla wszystkich widoków dostępnych z poziomu przycisków umieszczonych w paskach nawigacyjnych różnych ról. Zależności te wymienione zostały w pliku pom.xml (Kod 2). Przykład ten ukazuje również, użycie instrukcji warunkowej w miejscu ‘`text="${product.availability} ? 'Tak' : 'Nie''`, które pozwoliło na wyświetlenie informacji o dostępności w czytelny sposób dla użytkownika.

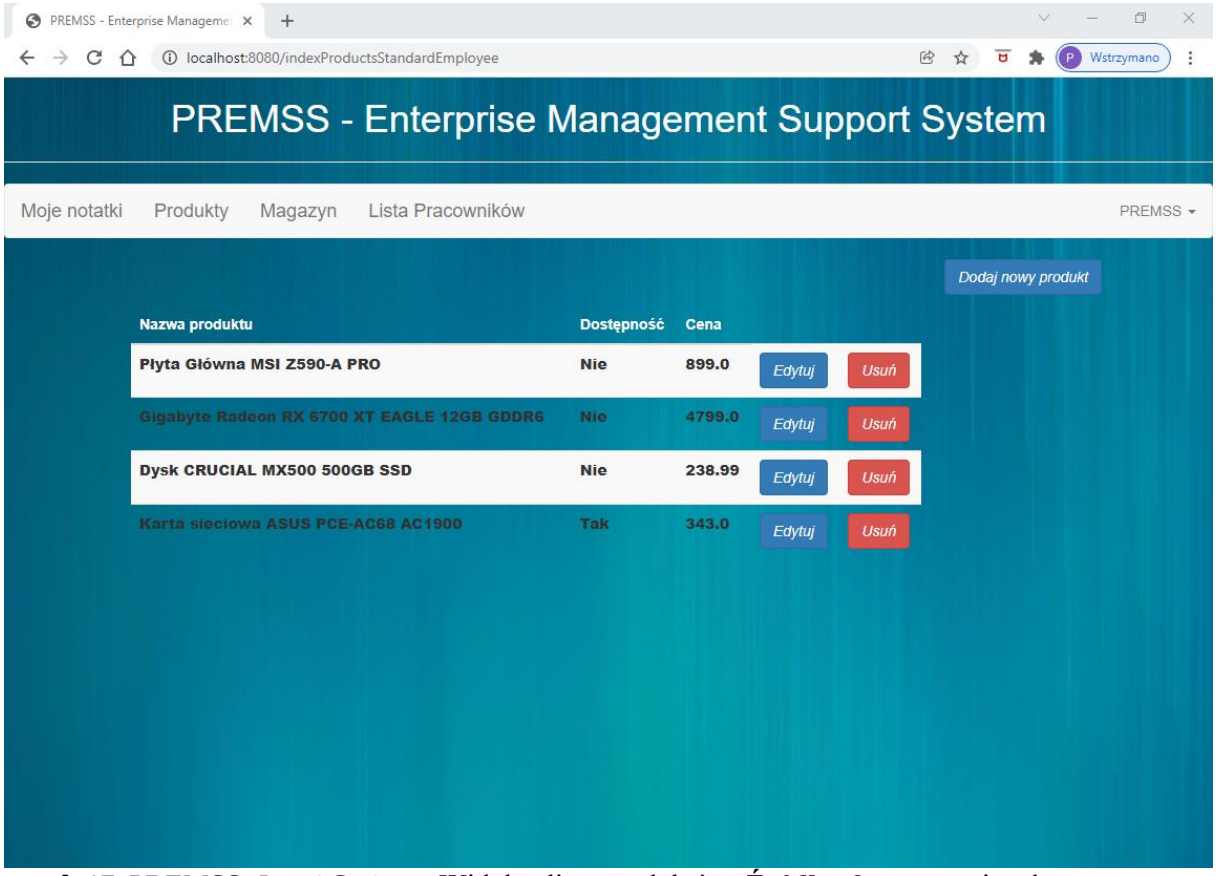
Thymeleaf posiada także rozwiązanie o nazwie ‘Thymeleaf fragments’. Pozwala ono na utworzenie jednego fragmentu kodu, który zostanie wielokrotnie wykorzystany w różnych miejscach. Mechanizm ten znalazł zastosowanie w niniejszej implementacji przy tworzenia menu dla każdej z ról. Kod 8 przedstawia źródła dla menu standardowego pracownika w postaci pliku ‘menu_StandardEmployee.html’.

Kod 8. Plik menu_StandardEmployee.html dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Menu - Standard Employee PREMSS - Thymeleaf Fragment</title>
</head>
<body>

<!-- Menu Standard employee - START -->
<div th:fragment="menuFragment_StandardEmployee">

    <nav class="navbar navbar-default" role="navigation">
        <div class="navbar-header">
            <a class="navbar-brand"
             th:href="@{/indexUserNotesStandardEmployee}">
                Moje notatki
            </a>
            <a class="navbar-brand"
             th:href="@{/indexProductsStandardEmployee}">
                Produkty
            </a>
            <a class="navbar-brand"
             th:href="@{/indexStoragePositionsStandardEmployee}">
```

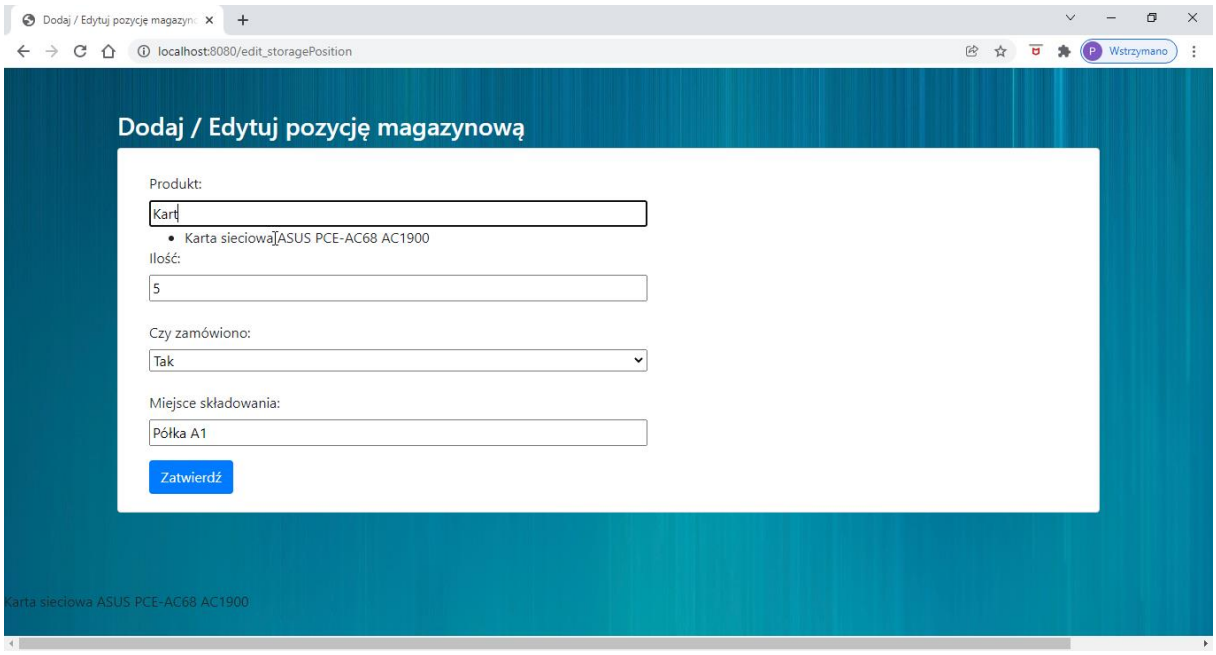
```
            Magazyn
         </a>
        <a class="navbar-brand"
         th:href="@{/indexEmployeesStandardEmployee}">
            Lista Pracowników
        </a>
    </div>

    <div class="collapse navbar-collapse navbar-ex1-collapse">

        <ul class="nav navbar-nav navbar-right">
            <li class="dropdown">
                <a href="#" class="dropdown-toggle"
                data-toggle="dropdown">PREMSS <b class="caret"></b></a>
                <ul class="dropdown-menu">
                    <li><a href="#" data-toggle="modal"
                data-target="#aboutPremssDialog">Informacje o systemie</a>
                    <li>
                        <a href="#" th:href="@{/logout}">Wyloguj</a>
                    </li>
                </ul>
            </li>
        </ul>
    </div>
</nav>

<!-- ### -->

<!-- Modal PREMSS -->
<div class="modal fade" id="aboutPremssDialog" role="dialog">
    <div class="modal-dialog">

        <!-- Modal content-->
        <div class="modal-content" id="aboutPremssDialogContent">

            <div class="modal-header aboutPremssDialogHeaderAndFotter">
                <button type="button" class="close"
                data-dismiss="modal">&times;</button>
                <h4 class="modal-title">
                  PREMSS - Informacje o systemie
                </h4>
            </div>

            <div class="modal-body">
                <p>Autor: Paweł Rzeńca</p>
                <p>Uczelnia: Polsko-Japońska Akademia
                 Technik Komputerowych (PJATK)</p>
                <br/>
                <p>Niniejsze oprogramowanie stanowi część pracy
```

```
                    magisterskiej autorstwa Pawła Rzeńcy.</p>
                <p>Wszelkie prawa zastrzeżone.</p>
            </div>

            <div class="modal-footer aboutPremssDialogHeaderAndFotter">
                <button type="button" class="btn btn-default"
                 data-dismiss="modal">Zamknij</button>
            </div>

        </div>

    </div>
</div>

</div>
<!-- Menu Standard employee - END -->

</body>
</html>
```



Rysunek 17. PREMSS, Java i Spring – Widok z listą produktów. **Źródło**: Opracowanie własne.

Ostatnim zaprezentowanym widokiem w ramach implementacji w Javie i Springu, będzie ten który związany jest z dodawaniem oraz edycją nowych pozycji magazynowych. Jego podgląd został przedstawiony na Rysunku 18.



Rysunek 18. PREMSS, Java i Spring – Widok do dodawania i edycji pozycji magazynowych. **Źródło**: Opracowanie własne

Za powyższy widok odpowiedzialny jest kod źródłowy (Kod 9), znajdujący się w pliku 'Add-Edit-StoragePosition.html'. Została w nim zaimplementowana wyszukiwarka produktów, a przykład jej działania widoczny jest na Rysunku 18, gdzie wyszukany został jeden z produktów znajdujący się w bazie danych. Implementacja wyszukiwarki opartej na technologii JavaScript wraz z użyciem wcześniej opisanych bibliotek jQuery oraz jQuery UI obrazuje zabezpieczenie Springa przed atakami CSRF (ang. *Cross-site request forgery*). Otóż, żeby tego typu żądanie kierowane do kontrolera zostało obsłużone, musi ono mieć dodany do nagłówka odpowiedni token. W Kodzie 9 odpowiedzialna jest za to linia '`headers: { "X-CSRF-TOKEN": token }`', wewnątrz metody '`ajax`'.

Kod 9. Plik Add-Edit-StoragePosition.html dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8">
    <meta th:name="${_csrf.parameterName}" th:content="${_csrf.token}"/>
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <title>Dodaj / Edytuj pozycję magazynową</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
                                 bootstrap/4.1.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/
                                 v5.4.1/css/all.css">
    <link rel="stylesheet" type="text/css" th:href="@{/css/shared.css}"/>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/
                 jquery.min.js"></script>
```

```
    <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/
                jquery-ui.min.js"></script>
</head>

<body>
<div class="container my-5">
    <h3 class="additionalWindowHeader"> Dodaj / Edytuj pozycję magazynową
    </h3>
    <div class="card">
        <div class="card-body">
            <div class="col-md-10">
                <form action="#" th:action="@{/create_storagePosition}"
                 th:object="${storagePosition}" method="post">
                    <div class="row">
                        <div class="form-group col-md-8">
                            <label for="name"
                             class="col-form-label">
                             Produkt: </label>
                            <input style="width: 100%" type="text"
                             th:field="*{product.name}"
                                   id="name" placeholder="Produkt" />
                            <input style="width: 100%" type="number"
                             th:field="*{product.id}"
                                   id="product_id" hidden />
                        </div>

                        <div class="form-group col-md-8">
                            <label for="count" class="col-form-label">
                             Ilość: </label>
                            <input style="width: 100%" type="number"
                              th:field="*{count}" id="count"
                              placeholder="Ilość" />
                        </div>

                        <div class="form-group col-md-8">
                            <label for="ordered" class="col-form-label">
                             Czy zamówiono:</label>
                            <select style="width: 100%" name="ordered"
                             id="ordered" th:field="*{ordered}">
                              <option th:value="1" th:text="Tak"></option>
                              <option th:value="0" th:text="Nie"></option>
                            </select>
                        </div>

                        <div class="form-group col-md-8">
                            <label for="storage_place"
                             class="col-form-label">
                              Miejsce składowania:</label>
                            <input style="width: 100%" type="text"
```

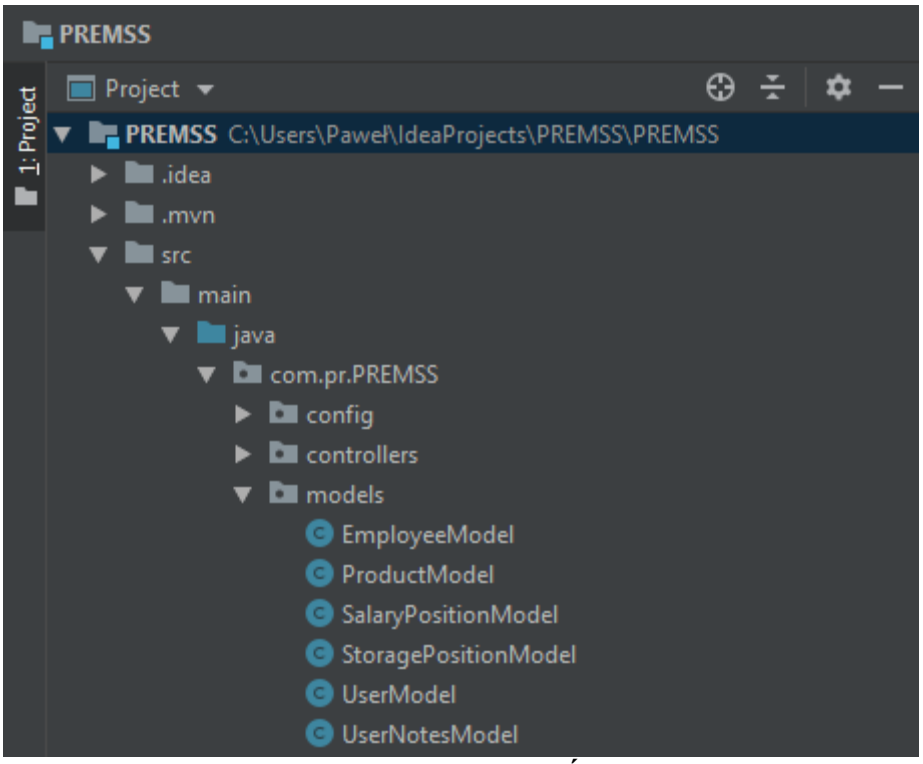
```
                        th:field="*{storage_place}" id="storage_place"
                        placeholder="Miejsce składowania" />
                </div>

                <div class="col-md-6">
                    <input type="submit" class="btn btn-primary"
                           value=" Zatwierdź ">
                </div>

                <input type="hidden" id="id" th:field="*{id}">

            </div>
        </form>
    </div>
</div>
</div>

<script th:inline="javascript">
$(document).ready(function () {
    var token = $("meta[name='_csrf']").attr("content");

    $("#name").autocomplete({
        source: function (request, response) {

            $.ajax({
                url: /*[[@{/getProducts}]]*/ '/getProducts',
                headers: { "X-CSRF-TOKEN": token },
                type: 'post',
                data: {
                    search: request.term
                },

                success: function (data) {

                    var responseArray = [];
                    for(var i = 0; i < data.length; i++) {

                        var item = {"label": data[i].name,
                                    "id": data[i].id} ;
                        responseArray.push(item);
                    }

                    response(responseArray);
                }
            });
        },
        select: function (event, ui) {
            $('#name').val(ui.item.label);
```

```
            $('#product_id').val(ui.item.id);
            return false;
        }
    });

});
</script>
</body>
</html>
```

6.4.2.4. MVC Java i Spring - Model

Model – Lista wszystkich zaimplementowanych modeli dla aplikacji w wersji Java i Spring znajduje się na rysunku 19. Modele zostały stworzone przy użyciu standardu JPA. W Kodzie 2, gdzie wymienione zostały wszystkie zależności, znajduje się pozycja 'spring-boot-starter-data-jpa'. Jest ona związana z silnikiem Hibernate, który jest implementacją JPA [89].

W Kodzie 10, można znaleźć źródła dla jednego ze stworzonych modeli – Pozycji magazynowych. Na jego podstawie przedstawiony zostanie schemat działania i tworzenia modelu. Żeby klasa stała się encją, która znajdzie odzwierciedlenie w bazie danych, dodano do niej adnotacje '`@Entity`'. Adnotacja '`@Table(name="Salary_positions")`', znajdująca się tuż nad deklaracją klasy określa pod jaką nazwą tabeli, dany model trafi do bazy. Pola klasy będącej encją, zostaną natomiast odwzorowane na kolumny tej tabeli. Do jednego z pól klasy dodana została adnotacja '`@Id`', która informuje, iż będzie ono identyfikatorem tworzonej tabeli. Również jako adnotacje nad polami, określane są relacje jakie zachodzą między poszczególnymi encjami – w Kodzie 10, dla pola '`product`' została określona relacja wiele do jednego, przez adnotacje '`@ManyToOne`'. Relacje znajdą odwzorowanie zarówno na poziomie kodu jak i bazy danych. To pierwsze ma taki wpływ, iż przykładowo podczas próby usunięcia rekordu, do którego referencje posiada inny obiekt, błąd wystąpi właśnie już na poziomie programu (jeżeli nie została określona kaskadowość, mówiąca przykładowo, iż ma on również zostać usunięty).



Rysunek 19. PREMSS, Java i Spring – Lista modeli. **Źródło**: Opracowanie własne.

Kod 10. Plik StoragePositionModel.java dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
//(…)
@Entity
@Table(name="Storage_positions")
public class StoragePositionModel {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private BigInteger count;

    private Boolean ordered;

    private String storage_place;

    @JsonIgnore
    @ManyToOne
    private ProductModel product;

    //Getters and Setters:
    public Long getId() { return id; }

    public void setId(Long id) { this.id = id; }

    public BigInteger getCount() { return count; }

    public void setCount(BigInteger count) { this.count = count; }

    public Boolean getOrdered() { return ordered; }

    public void setOrdered(Boolean ordered) { this.ordered = ordered; }

    public String getStorage_place() { return storage_place; }

    public void setStorage_place(String storage_place) {
       this.storage_place = storage_place; }

    public ProductModel getProduct() { return product; }

    public void setProduct(ProductModel product) {  this.product = product;
}

}
```

Przy okazji omawiania modeli, które wiążą się z bazami danych, należy wspomnieć iż konfiguracja kwestii z nią związanych takich jak dane dostępowe ma miejsce w pliku application.properties.

6.4.2.5. MVC Java i Spring - Kontroler

Kontroler (ang. *controller*) – Pozostałą częścią modelu MVC, którą należy opisać jest kontroler. Program zawiera cztery kontrolery: jeden główny kontroler aplikacji, któremu odpowiada plik MainPremssController.java oraz trzy, gdzie każdy z nich odpowiedzialny jest za inną rolę użytkownika zdefiniowanego w systemie, zgodnie z wcześniej opisanymi wymaganiami. Są to pliki AdminController.java, HrEmployeeController.java oraz StandardEmployeeController.java, dla kolejno roli Administratora, Pracownika HR oraz Standardowego pracownika.

W Kodzie 11 zostały zawarte źródła, dla kontrolera związanego z rolą Standardowego pracownika. Nad deklaracją klasy znajduje się adnotacja '`@Controller`', która oznacza, iż będzie ona pełnić funkcje kontrolera i zostanie umieszczona w kontenerze. Widać na niej deklarację pól, będących serwisami i repozytoriami, które oznaczone zostały adnotacją '`@Autowired`'. Dzięki jej zastosowaniu nie było konieczności ich inicjalizacji, ponieważ zostały on wstrzyknięte na podstawie kontenera Springa – używany jest w tym miejscu wzorzec wstrzykiwania zależności. W kontrolerze zostały również zdefiniowane metody, które na podstawie modelu dokonują takich operacji, jak pobranie danych i zwrócenie widoku, dodanie nowych danych czy ich usunięcie.

Warto tutaj zwrócić uwagę, iż nad takimi metodami znajdują się adnotacje '`@GetMapping`' lub '`@RequestMapping`', przy pomocy których dane metody zostały powiązane z konkretnymi ścieżkami do nich prowadzącymi.

Kod 11. Plik StandardEmployeeController.java dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne

```
//(…)
@Controller
public class StandardEmployeeController {

    @Autowired
    ProductService productService;

    @Autowired
    EmployeeService employeeService;

    @Autowired
    StoragePositionService storagePositionService;

    @Autowired
    UserNotesService userNotesService;

    @Autowired
    UserRepository userRepository;

    //<<< User Notes Controller Section - Standard Employee - START >>>
    @GetMapping("/indexUserNotesStandardEmployee")
    public String getIndexUserNotesStandardEmployee(Model model) {

        Object principal = SecurityContextHolder.getContext()
        .getAuthentication().getPrincipal();
        String username;

        if (principal instanceof UserDetails) {
            username = ((UserDetails) principal).getUsername();
        } else {
```

```
            username = principal.toString();
        }

        Long id = userRepository.findByName(username).getId();

        UserModel userModel = userRepository.findById(id).get();

        UserNotesModel userNotes = userModel.getUserNotes();
        model.addAttribute("userNotes", userNotes);

        return "StandardEmployee_MainPage.html";
    }

    @RequestMapping(path = "/update_usernotes_standardemployee",
                    method = RequestMethod.POST)
    public String updateUserNotes(UserNotesModel userNotes) {
        userNotesService.updateUserNotes(userNotes);

        return "redirect:/indexUserNotesStandardEmployee";
    }
    //<<< User Notes Controller Section - Standard Employee - END >>>

    //<<< Product Controller Section - START >>>
    @GetMapping("/indexProductsStandardEmployee")
    public String getIndexProductsStandardEmployee(Model model){
        List<ProductModel> productList = productService.getAllProducts();

        model.addAttribute("productList", productList);

        return "StandardEmployee_Products.html";
    }

    @RequestMapping(value = "/getProducts", method = RequestMethod.POST)
    @ResponseBody
    public List<ProductModel> getProducts(
    @RequestParam("search") String search){

        List<ProductModel> listOfStoragePositions =
        productService.getProducts(search);

        return listOfStoragePositions;
    }

    @RequestMapping(path = {"/edit_product", "/edit_product/{id}"})
    public String editProductById(Model model,
                                  @PathVariable("id") Optional<Long> id)
    {
        if (id.isPresent()) {
            ProductModel entity = productService.getProductById(id.get());
```

```
            model.addAttribute("product", entity);
        } else {
            model.addAttribute("product", new ProductModel());
        }
        return "Add-Edit-Product";
    }

    @RequestMapping(path = "/delete_product/{id}")
    public String deleteProductById(Model model, @PathVariable("id")
                                    Long id)
    {
        productService.deleteProductById(id);
        return "redirect:/indexProductsStandardEmployee";
    }

    @RequestMapping(path = "/create_product", method = RequestMethod.POST)
    public String createOrUpdateProduct(ProductModel product) {
        productService.createOrUpdateProduct(product);
        return "redirect:/indexProductsStandardEmployee";
    }
    //<<< Product Controller Section - END >>>

    //<<< StoragePosition Controller Section - START >>>
   @GetMapping("/indexStoragePositionsStandardEmployee")
    public String getIndexStoragePositionsStandardEmployee(Model model){
        List<StoragePositionModel> storagePositionList =
        storagePositionService.getAllStoragePositions();

        model.addAttribute("storagePositionList", storagePositionList);

        return "StandardEmployee_Warehouse.html";
    }

    @RequestMapping(path = {"/edit_storagePosition",
                            "/edit_storagePosition/{id}"})
    public String editStorageById(Model model,
                                  @PathVariable("id") Optional<Long> id)
    {
        if (id.isPresent()) {
            StoragePositionModel entity = storagePositionService
            .getStoragePositionById(id.get());
            model.addAttribute("storagePosition", entity);
        } else {
            model.addAttribute("storagePosition",
                               new StoragePositionModel());
        }
        return "Add-Edit-StoragePosition";
    }
```

```
    @RequestMapping(path = "/delete_storagePosition/{id}")
    public String deleteStoragePositionyId(StoragePositionModel model,
                                           @PathVariable("id") Long id)
    {
        storagePositionService.deleteStoragePositionById(id);
        return "redirect:/indexStoragePositionsStandardEmployee";
    }

    @RequestMapping(path = "/create_storagePosition",
                    method = RequestMethod.POST)
    public String createOrUpdateStoragPosition(
    StoragePositionModel storagePositionModel) {
        storagePositionService
        .createOrUpdateStoragePosition(storagePositionModel);
        return "redirect:/indexStoragePositionsStandardEmployee";
    }
    //<<< StoragePosition Controller Section - END >>>
    //<<< Employees Controller Section - Standard Employee - START >>>
    @GetMapping("/indexEmployeesStandardEmployee")
    public String getIndexEmployeesStandardEmployee(Model model){
      List<EmployeeModel> employeeList = employeeService.getAllEmployees();

      model.addAttribute("employeeList", employeeList);

      return "StandardEmployee_EmployeeList.html";
    }
    //<<< Employees Controller Section - Standard Employee - END >>>
}
```

6.4.2.6. Istotne elementy Springa - Repozytorium i Serwisy

Repozytorium (ang. *repository*) – W implementacji oprogramowania PREMSS w wersji, gdzie użyto języka Java oraz frameworka Spring, zastosowanie znalazły również repozytoria, które są komponentami dostarczanymi przez omawianą platformą programistyczną. Są one związane z operacjami na danych pochodzących z dowolnego źródła takimi jak: stwórz, odczytaj, zaktualizuj oraz usuń. Aby klasa stała się repozytorium dodawana jest do niej adnotacja '`@Repository`'. Klasy takie są singletonami i zarazem Spring Beanami, które trafiają do kontenera, skąd można je wstrzyknąć, co pokazane zostały przy okazji omawiania kontrolera. Dla każdego modelu zaimplementowane zostało odpowiadające mu repozytorium, tak więc ich lista dla programu przedstawia się następująco: EmployeeRepository, SalaryPositionRepostory, StoragePositionRepository, UserNotesRepository, UserRepository oraz ProductRepository. Kod 12 przedstawia interfejs w postaci tego ostatniego.

Kod 12. Plik ProductRepository.java dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
//(…)
@Repository
public interface ProductRepository extends CrudRepository<ProductModel,
Long> {
}
```

Serwisy (ang. *services*) – To kolejny typ klasy związany z frameworkiem Spring, który znalazł zastosowanie w implementacji niniejszego projektu oprogramowania. Serwisy związane są z obsługiwaniem operacji biznesowych. Do określenia tego, że dana klasa będzie pełnić rolę serwisu, dodawana jest nad nią adnotacja '`@Service`'. Tak samo jak w przypadku repozytorium, są one Spring Beanami umieszczanymi w kontenerze i są oparte o wzorzec singleton. Każdy z zaimplementowanych modeli w ramach programu PREMSS, posiada jeden z odpowiadających mu serwisów: EmployeeService, ProductService, SalaryPostionService, StoragePositonService, UserService. Dla drugiego z wymienionych repozytoriów, kod źródłowy został zamieszczony jako Kod 13.

Kod 13. Plik ProductService.java dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne

```
//(…)
@Service
public class ProductService {

    @PersistenceContext
    private EntityManager em;

    @Autowired
    ProductRepository productRepository;

    public List<ProductModel> getAllProducts(){
        List<ProductModel> listOfProducts =
        (List<ProductModel>) productRepository.findAll();

        if(listOfProducts.size() > 0){
            return  listOfProducts;
        } else{
            return new ArrayList<ProductModel>();
        }
    }

    public ProductModel getProductById(Long id){
        Optional<ProductModel> product = productRepository.findById(id);

        return product.get();
    }

    public ProductModel createOrUpdateProduct(ProductModel entity)
    {
        if(entity.getId()  == null)
        {
            entity = productRepository.save(entity);

            return entity;
        }
        else
        {
            Optional<ProductModel> product =
                    productRepository.findById(entity.getId());
```

```
            if(product.isPresent())
            {
                ProductModel newEntity = product.get();
                newEntity.setName(entity.getName());
                newEntity.setAvailability(entity.getAvailability());
                newEntity.setPrice(entity.getPrice());
                newEntity.setStorage_position(null);

                newEntity = productRepository.save(newEntity);

                return newEntity;
            } else {
                entity = productRepository.save(entity);

                return entity;
            }
        }
    }

    public void deleteProductById(Long id)
    {
        productRepository.deleteById(id);
    }

    public List<ProductModel> getProducts(String term){

        List<ProductModel> listOfStoragePositions =
        em.createQuery("from ProductModel p where " +"p.name LIKE :name",
                ProductModel.class)
                .setParameter("name", "%" + term + "%")
                .setFirstResult(0)
                .setMaxResults(5)
                .getResultList();

        return listOfStoragePositions;
    }
}
```

Przy okazji omawiania serwisów, warto wspomnieć, iż podczas implementacji serwisu związanego z użytkownikami, przy implementowaniu metody dla ich tworzenia i aktualizacji przez administratora, wykorzystano możliwość kodowania haseł. Jest ona dostarczana przez pakiet Springa: ‘org.springframework.security.crypto.password’. Kod odpowiedzialny za wspomnianą funkcjonalność po stronie oprogramowania PREMSS, pokazany jest w listingu oznaczonym jako Kod 14. Pochodzi on z pliku ‘UserService.java’:

Kod 14. Metoda createOrUpdateUser z pliku UserService.java dla projektu PREMSS – Java i Spring.
Źródło: Opracowanie własne

```
//(…)
    public UserModel createOrUpdateUser(UserModel entity)
    {
        if(entity.getId() == null)
        {
           String passwordEncrypted =
           passwordEncoder.encode(entity.getPassword().trim());
           entity.setPassword(passwordEncrypted);
           entity.setEnabled("1");
           entity = userRepository.save(entity);

           UserNotesModel relatedEntity = new UserNotesModel();
           relatedEntity.setUserNotes("Tu jest miejsce na Twoje notatki.");
           relatedEntity.setUser(entity);
           userNotesRepository.save(relatedEntity);


            return entity;
        }
        else
        {
            Optional<UserModel> user =
                    userRepository.findById(entity.getId());

            if(user.isPresent())
            {
                String passwordEncrypted =
                passwordEncoder.encode(entity.getPassword().trim());
                UserModel newEntity = user.get();
                newEntity.setName(entity.getName());
                newEntity.setPassword(passwordEncrypted);
                newEntity.setRole(entity.getRole());
                newEntity.setEmployee(entity.getEmployee());

                newEntity = userRepository.save(newEntity);
                return newEntity;
            } else {
                entity = userRepository.save(entity);
                return entity;
            }
        }
    }
//(…)
```

Wraz ze Spring Bootem dostarczony został serwer deweloperski odpalany w przypadku Mavena przy pomocy komendy ‘`mvn spring-boot:run`’ – znalazł on zastosowanie podczas implementacji wyżej opisanego programowania.

6.5. Implementacja aplikacji webowej w języku PHP z wykorzystaniem frameworka Laravel

Niniejszy rozdział zawiera opis frameworka Laravel wraz z omówieniem jego elementów oraz zaimplementowanego programu, który powstał w oparciu o tą platformę programistyczną.

6.5.1 Wprowadzenie do frameworka Laravel

Analogicznie jak w przypadku poprzedniego rozdziału na początek powinno odpowiedzieć się na pytanie czym jest Laravel? Laravel jest to framework przeznaczony do tworzenia aplikacji internetowych z użyciem języka PHP. Wspiera on tworzenie oprogramowania w oparciu o wzorzec Model-Widok-Kontroler (ang. *Model-View-Controler*), w skrócie MVC. Dostarcza narzędzi związanych z: testami jednostkowymi, wstrzykiwaniem zależności, zdarzeniami, warstwami abstrakcji na poziomie bazy danych oraz wielu innych. Posiada bogatą dokumentację, poradniki czy materiały szkoleniowe w formie video. Cechuje się łatwą skalowalnością do obsługi zapytań nawet rzędu setek milionów na miesiąc [90].

Wśród najważniejszych właściwości i funkcji Laravela możemy wymienić [91]:

- **Modułowość** – Laravel został stworzony na bazie ponad dwudziestu bibliotek. Zostały one pogrupowane w osobne moduły i są ściśle powiązane z narzędziem do zarządzania zależnościami – Composer przy pomocy, którego można nimi rozporządzać.

- **Trasowanie** – Trasowanie (ang. *routing*) w Laravelu, pozwala na wiązanie poszczególnych funkcji aplikacji z ścieżkami oraz metodami HTTP takimi jak POST, PUT, DELETE czy GET. Istnieje możliwość robienia tego manualnie co pozwala na pewną elastyczność i dostosowanie ścieżek do własnych potrzeb.

- **Testowalność** – Omawiany framework opiera się na koncepcji łatwego testowania. Dostarcza rozwiązań pozwalających na wiązanie ścieżek wspierających testy czy przykładowo funkcjonalności do potwierdzenia, że dana metoda została wywołana z odpowiedniej klasy.

- **Zarządzanie konfiguracją** – W związku z tym, że w procesie oprogramowania może być ono tworzono lokalnie, a następnie dystrybuowane na różne środowiska docelowe, w Laravelu umożliwiona jest łatwa konfiguracja przy pomocy pliku do tego przeznaczonego. W pliku tym definiowane są ustawienia takie jak dane dostępowe do bazy danych czy poświadczenia do serwera mailowego.

- **Silnik szablonów** – Niniejsza platforma programistyczna dostarcza również silnik szablonów – Blade. Pozwala on na tworzenie hierarchicznych układów stron, które mogą być wykorzystywane w wielu miejscach, w które będzie dynamicznie wstrzykiwana różna treść.

- **Narzędzie Query Build oraz ORM Eloquent** – Laravel wyposażony jest w rozwiązanie do mapowania obiektowo relacyjnego o nazwie Eloquent. Ułatwia to operowania na obiektowym kodzie, który później zostaje odwzorowany na bazę danych. Taka metoda podejścia do danych, bazuje na modelu. Posiada on również narzędzie Query Builder, które pozwala na elastyczne podejście do tworzenia zapytań do bazy danych, które tworzy się w języku PHP, a nie bezpośrednio w języku SQL.

- **Mechanizmy Schema builder, migrations oraz seedings** – Laravel dostarcza funkcjonalności pozwalających na definiowanie i określanie struktury bazy danych w PHP, a następnie jej śledzenia z użyciem mechanizmu migracji. Funkcjonalność seeding, pozwala natomiast na zdefiniowanie danych początkowych lub testowych, które zostaną utrwalone w bazie danych.

- **Funkcjonalności związanie z pocztą elektroniczną** – Laravel zawiera w sobie klasy i biblioteki, które pozwalają na wysyłanie maili zawierających załączniki. Wspiera on również popularne serwisy do wysyłki poczty elektronicznej.

- **Autentykacje** – Wraz z Laravelem dostarczone są mechanizmy związane z autentykacją, co jest zwłaszcza istotne z punktu widzenia oprogramowania internetowego. Wśród nich można wymienić funkcjonalności związane z: rejestracją użytkowników, ich autentykacją czy wysyłaniem przypomnień związanych z hasłami do użytkowników.

- **Redis** – Framework ten dostarcza możliwości do łączenia się i integracji z bazą danych Redis. Jest ona przykładem nierelacyjnej bazy danych, która opiera się na strukturze klucz-wartość.

- **Mechanizmy związane z obsługą wydarzeń i poleceń** – W Laravelu istnieją również rozwiązania, które pomagają w obsłudze wydarzeń (ang. *events*) oraz poleceń (ang. *commands*). Są one oparte o wzorce *Event bus* oraz *Command bus*.

Najbardziej aktualnym wydaniem Laravel'a jest wersja 8. Zaznaczyć należy, iż w niedługim czasie, bo w dniu 8 lutego 2022 planowane jest wypuszczenie wersji 9 [92].

Więcej praktycznych informacji na temat frameworka Laravel dostarcza następna część, w której została opisana implementacja projektu oprogramowania wspierającego zarządzanie przedsiębiorstwem, powstała w ramach niniejszej pracy magisterskiej.

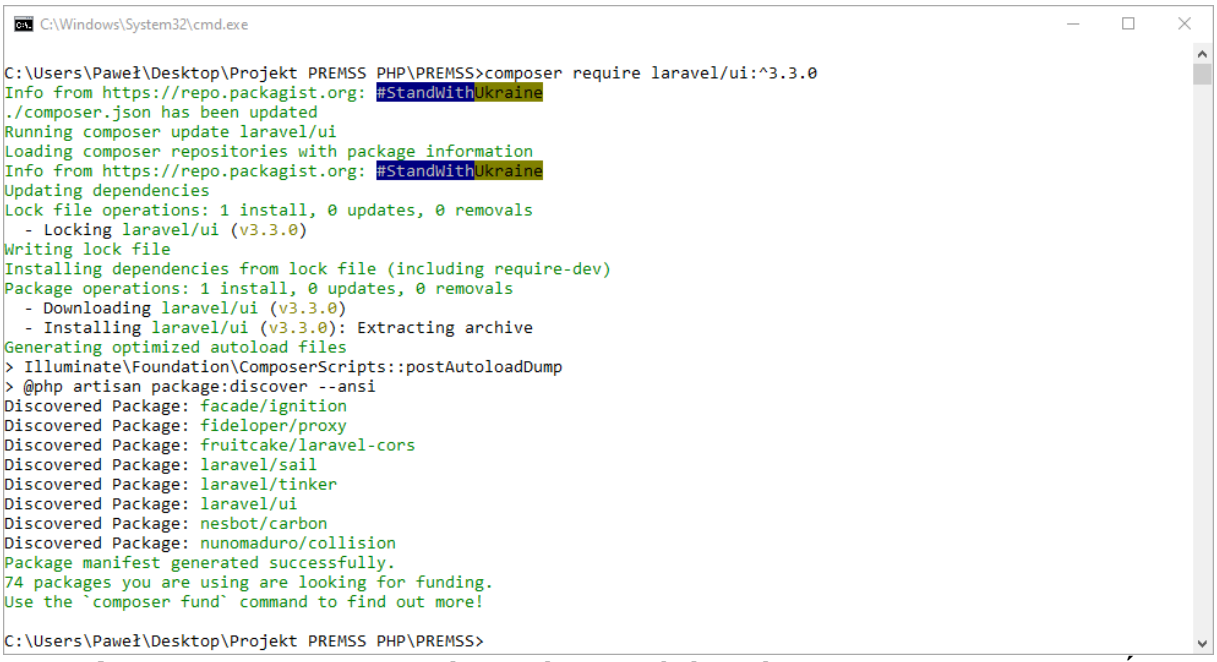
6.5.2 Praktyczna implementacja oprogramowania w oparciu o język PHP oraz framework Laravel

Celem poniższych podrozdziałów jest zaprezentowanie implementacji oprogramowania, która została stworzona na potrzeby niniejszej pracy dyplomowej i opiera się na frameworku Laravel.

6.5.2.1. Inicjowanie projektu PHP i Laravel i zarządzanie zależnościami

We wcześniejszym rozdziale wspomniane zostało narzędzie Composer służące do zarządzania paczkami oraz zależnościami. Przy jego pomocy możliwe jest również zainicjalizowanie projektu PREMSS, opartego o język PHP oraz framework Laravel. W tym celu z poziomu linii komend, zostało wpisane polecenie '`composer create-project --prefer-dist laravel/laravel PREMSS`', którego skutkiem było powstanie szkieletu aplikacji.

Podczas implementacji aplikacji opartej o Laravel, wystąpiła potrzeba pobrania dodatkowej paczki w postaci Laravel UI. Również do tego celu użyty został Composer. Schemat instalacji dodatkowej zależności z poziomu linii komend, przy pomocy wspomnianego narzędzia przedstawiony został na Rysunku 20.



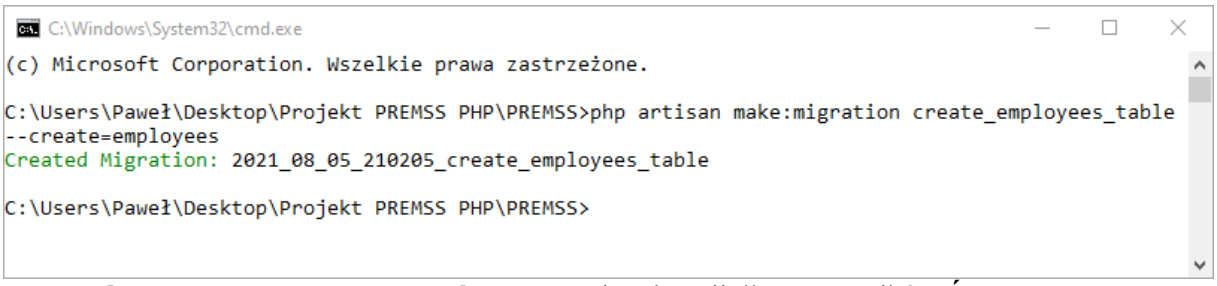
Rysunek 20. PREMSS, PHP i Laravel – Instalacja paczki laravel ui przy pomocy Composera. **Źródło**: Opracowanie własne.

Laravel zawiera w sobie interfejs linii komend Artisan, którego celem jest dostarczenie instrumentu posiadającego liczną liczbę poleceń, które używane są podczas tworzenia oprogramowania, przy użyciu tego frameworka. Instrument ten związany jest z komponentem o nazwie ‘Console component’ z innej platformy programistycznej PHP - Symfony [93].

Artisan CLI znalazł zastosowanie przy dodawaniu paczek i komponentów w ramach zaimplementowanego oprogramowania PREMSS przy użyciu języka PHP i frameworka Laravel.

6.5.2.2. Istotne elementy Laravela - Migracje i Ścieżki

Migracje (ang. *migrations*) – Laravel powiązany jest ściśle z pojęciem migracji. Migracje są to klasy, w których definiuje się schemat tabel w celu ich tworzenia oraz aktualizacji [94]. Do stworzenia klasy z migracją wykorzystywany jest Artisan. Przykładowa komenda do stworzenia migracji widoczna jest na Rysunku 21 – w tym przypadku stworzona została migracja dla pracowników (ang. *employees*).



Rysunek 21. PREMSS, PHP i Laravel –Stworzenie migracji dla pracowników **Źródło**: Opracowanie własne.

Po wykonaniu komendy widocznej na Rysunku 21, utworzona została klasa ‘CreateEmployeesTable’, która rozszerza klasę – ‘Migration’.

Inny przykład migracji zaimplementowanej w ramach oprogramowania PREMSS związany jest z pozycjami zawierającymi informacje dotyczące zarobków.

Przez wywołanie komendy ‘`php artisan make:migration create_salary_positions_table`’ został utworzony jej szkielet. Taka klasa po wygenerowaniu zawiera dwie metody ‘up’, gdzie zostanie ona wywołana podczas wykonywania migracji oraz ‘down’, która wykonana zostanie podczas jej cofania. Stan klasy z migracją po wygenerowaniu przez

wspomniane polecenie, prezentuje się tak jak w Kodzie 15. Widać na nim, iż dzięki zastosowaniu odpowiedniej konwencji nazewnictwa w tworzeniu komendy, zarówno klasa jak i nazwa tabeli (pierwszy argument metody '`create`') posiada od początku odpowiednią nazwę.

Kod 15. Plik 2021_12_17_214746_create_salary_positions_table.php (po wygenerowaniu) dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```php
<?php
//(…)
class CreateSalaryPositionsTable extends Migration
{
//(…)
    public function up()
    {
        Schema::create('salary_positions', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('salary_positions');
    }
}
```

Dodawanie kolejnych kolumn ma miejsce wewnątrz funkcji, przekazywanej jako drugi argument do metody '`create`'. W celu zdefiniowania dodatkowych kolumn tabeli, programista wywołuje na obiekcie '`$table`' metody, której nazwy określają typ danych i przyjmują jako argument nazwę kolumny. Dodatkowo istnieje tutaj również możliwość, określenia relacji, które mają zostać odzwierciedlone na poziomie bazy danych. Zaimplementowany kod dla migracji 'CreateSalaryPositionsTable' przedstawiony został jako Kod 16.

Kod 16. Plik 2021_12_17_214746_create_salary_positions_table.php (po zaimplementowaniu) dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```php
<?php
//(…)
class CreateSalaryPositionsTable extends Migration
{
    //(…)
    public function up()
    {
        Schema::create('salary_positions', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            $table->string('bank_account_number');
            $table->unsignedBigInteger('gross_salary');
            $table->unsignedBigInteger('net_salary');
            $table->unsignedBigInteger('employee_id');
```

```
            //Foreign key
            $table->foreign('employee_id')
            ->references('id')->on('employees')
            ->onUpdate('CASCADE')
            ->onDelete('CASCADE');
        });
    }

    public function down()
    {
        Schema::dropIfExists('salary_positions');
    }
}
```

Żeby migracja znalazła odwzorowanie w bazie danych, musi zostać uruchomiona. W celu wykonania migracji wykonywane jest polecenie '`php artisan migrate`' przy pomocy linii komend Artisan.

Ścieżki (ang. *routes*) – Laravel umożliwia elastyczne określenie ścieżek, tak aby z danym identyfikatorem w postaci adresu zasobu, wiązało się wywołanie konkretnego kodu [95]. Ta część ścieżek, która zdefiniowana została przez autora niniejszej pracy, określona została w pliku 'web.php', którego źródła znalazły odzwierciedlenie w Kodzie 17.

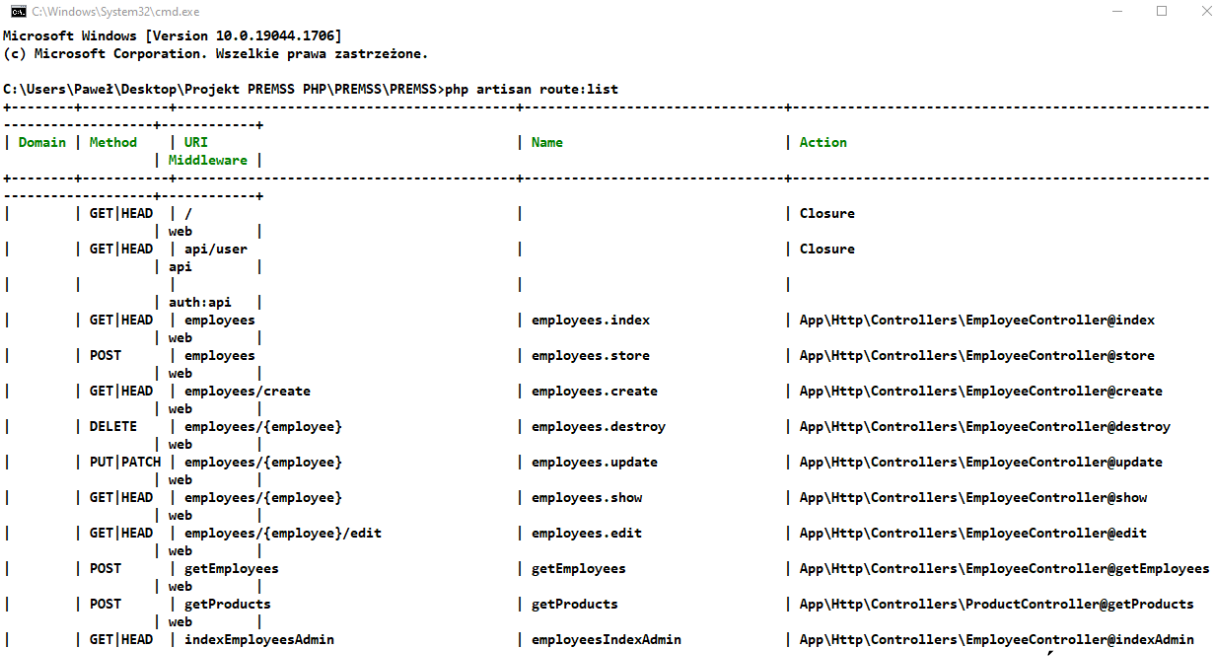
Kod 17. Plik web.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```
<?php
//(…)
//Route for not logged in users
Route::get('/', function () {
    return redirect('/login');
});

//Auth routes:
Auth::routes();

//Routes for UserNotes:
Route::resource('user_notes', UserNotesController::class);

//Routes for Employees:
Route::resource('employees', EmployeeController::class);

//Routes for Products:
Route::resource('products', ProductController::class);

//Routes for StoragePositions:
Route::resource('storage_positions', StoragePositionController::class);

//Routes for SalaryPositions:
Route::resource('salary_positions', SalaryPositionController::class);

//Routes for Users:
Route::resource('users', UserController::class);
```

```
// ### ### ###

//Route for Logout:
Route::get('logout', '\App\Http\Controllers\Auth\LoginController@logout');

//Route for Searching product by name:
Route::post('/getProducts', [ProductController::class, 'getProducts'])
->name('getProducts');

//Route for Searching employees:
Route::post('/getEmployees', [EmployeeController::class, 'getEmployees'])
->name('getEmployees');

// ### ### ###

//Route for Standard Employees view:
Route::get('/indexEmployeesStanardEmployee', [EmployeeController::class,
'indexStandardEmployee'])->name('employeesIndexStandardEmployee');

//Route for HR Employees view:
Route::get('/indexEmployeesHREmployee', [EmployeeController::class,
'indexHREmployee'])->name('employeesIndexHREmployee');

//Route for Admin Employees view:
Route::get('/indexEmployeesAdmin', [EmployeeController::class,
'indexAdmin'])->name('employeesIndexAdmin');

//Route for Standard employee User notes view:
Route::get('/indexUserNotesStandardEmployee', [UserNotesController::class,
'indexStandardEmployee'])->name('userNotesIndexStandardEmployee');

//Route for HR employee User notes view:
Route::get('/indexUserNotesHREmployee', [UserNotesController::class,
'indexHREmployee'])->name('userNotesIndexHREmployee');

//Route for Admin User notes view:
Route::get('/indexUserNotesAdmin', [UserNotesController::class,
'indexAdmin'])->name('userNotesIndexAdmin');

// ### ### ###

//Route for updating User notes by Standard employee:
Route::put('/updateUserNotesStandardEmployee', [UserNotesController::class,
'updateStandardEmployee'])->name('userNotesUpdateStandardEmployee');

//Route for updating User notes by HR employee:
Route::put('/updateUserNotesHREmployee', [UserNotesController::class,
'updateHREmployee'])->name('userNotesUpdateHREmployee');
```

```
//Route for updating User notes by admin:
Route::put('/updateUserNotesAdmin', [UserNotesController::class,
'updateAdmin'])->name('userNotesUpdateAdmin');
```

Z ścieżkami wiąże się narzędzie Artisan, które pozwala na sprawdzenie listy ścieżek określonych dla danego oprogramowania, zaimplementowanego w oparciu o framework Laravel. Służy do tego odpowiednie polecenie, gdzie efekt po jego wpisaniu został przedstawiony dla aplikacji PREMSS na Rysunku 22 (widoczna jest na nim tylko część ścieżek):



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Paweł\Desktop\Projekt PREMSS PHP\PREMSS\PREMSS>php artisan route:list
+--------+-----------+-----------------------------------------+-------------------------------+-------------------------------------------------
------------------+------------+
| Domain | Method    | URI                                     | Name                          | Action
                  | Middleware |
+--------+-----------+-----------------------------------------+-------------------------------+-------------------------------------------------
------------------+------------+
|        | GET|HEAD  | /                                       |                               | Closure
                  | web        |
|        | GET|HEAD  | api/user                                |                               | Closure
                  | api        |
|        |           |                                         |                               |
                  | auth:api   |
|        | GET|HEAD  | employees                               | employees.index               | App\Http\Controllers\EmployeeController@index
                  | web        |
|        | POST      | employees                               | employees.store               | App\Http\Controllers\EmployeeController@store
                  | web        |
|        | GET|HEAD  | employees/create                        | employees.create              | App\Http\Controllers\EmployeeController@create
                  | web        |
|        | DELETE    | employees/{employee}                    | employees.destroy             | App\Http\Controllers\EmployeeController@destroy
                  | web        |
|        | PUT|PATCH | employees/{employee}                    | employees.update              | App\Http\Controllers\EmployeeController@update
                  | web        |
|        | GET|HEAD  | employees/{employee}                    | employees.show                | App\Http\Controllers\EmployeeController@show
                  | web        |
|        | GET|HEAD  | employees/{employee}/edit               | employees.edit                | App\Http\Controllers\EmployeeController@edit
                  | web        |
|        | POST      | getEmployees                            | getEmployees                  | App\Http\Controllers\EmployeeController@getEmployees
                  | web        |
|        | POST      | getProducts                             | getProducts                   | App\Http\Controllers\ProductController@getProducts
                  | web        |
|        | GET|HEAD  | indexEmployeesAdmin                     | employeesIndexAdmin           | App\Http\Controllers\EmployeeController@indexAdmin
```

Rysunek 22. PREMSS, PHP i Laravel – Komenda do wyświetlenia listy ścieżek **Źródło**: Opracowanie własne.

6.5.2.3. MVC PHP i Laravel - Widok

Laravel posiada mechanizmy, które pozwalają skutecznie zaimplementować oprogramowanie oparte na wzorcu MVC. Stworzona aplikacja PREMSS została właśnie oparta na bazie tego wzorca.

Widok (ang. *view*) – Laravel został wyposażony w silnik szablonów Blade. Pozwala on na umieszczanie zarówno kodu w języku PHP jak i HTML w jednym pliku. Jego mechanizmy wspomagają pracę z kontrolerem oraz modelem, wspierając przez to wytwarzanie oprogramowania opartego o wzorzec MVC. Pliki oparte o Blade posiadają rozszerzenie '.blade.php' i kompilowane są do kodu PHP [96]. Przykładowy widok opracowany w ramach oprogramowania PREMSS, widoczny jest na Rysunku 23, a kod mu odpowiadający został wylistowany jako Kod 18. Dzięki użyciu wspomnianego silnika szablonów, usprawnione zostało operowanie na danych przesłanych w formie modelu, co obrazuje przykładowo miejsce, gdzie w celu rozpropagowania danych w tabeli, użyta na nich została pętla '`foreach`'.

Kod 18. Plik HREmployee_Employees.blade.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```
@extends('layouts.menu_HREmployee')

@section('title', 'PREMSS - Enterprise Management Support System')

@section('content')
<div class="container my-5">

    <p class="my-5" align="right">
        <a href="{{ route('employees.create') }}" class="btn btn-primary">
            <i class="fas fa-user-plus ml-2"> Dodaj nowego pracownika </i>
        </a>
    </p>

    <table class="table table-striped table-responsive-md premssDataTable">
        <thead>
            <tr>
                <th class="premssHeadlineText">Imię</th>
                <th class="premssHeadlineText">Nazwisko</th>
                <th class="premssHeadlineText">Numer telefonu</th>
                <th class="premssHeadlineText">Adres email</th>
                <th class="premssHeadlineText">Dział</th>
                <th></th>
                <th></th>
            </tr>
        </thead>
        <tbody>
            @foreach($employees as $employee)
            <tr>
                <td>{{ $employee->firstname }}</td>
                <td>{{ $employee->lastname }}</td>
                <td>{{ $employee->phoneNumber }}</td>
                <td>{{ $employee->emailAddress }}</td>
                <td>{{ $employee->department }}</td>
                <td>
                    <a href="{{ route('employees.edit',
                     $employee->id) }}" class="btn btn-primary">
                        Edytuj
                    </a>
                </td>
                <td>
                    <form action="{{ route('employees.destroy',
                     $employee->id)}}" method="post"
                     style="display: inline-block">
                        @csrf
                        @method('DELETE')
                        <button class="btn btn-danger" type="submit">
                          Usuń
```

```
                        </button>
                    </form>
                </td>
            </tr>
            @endforeach
        </tbody>
    </table>

</div>
@endsection
```

Blade umożliwia tworzenie layoutów. Zostało to wykorzystane do stworzenia trzech układów stron dla różnych ról, w których znajduje się odpowiednie dla nich menu, podpięte są pliki JavaScript i CSS oraz źródła związane z okienkiem dialogowym ‘Informacje o systemie’. Przykładowy layout dla roli pracownika HR, znajduje się w Kodzie 19. Znalazł on zastosowanie między innymi przy implementowaniu widoku z listą pracowników i przyciskami do akcji, gdzie natomiast jego kod źródłowy ukazany jest jako Kod 18. W układzie określone zostały sekcje, które przeznaczone są do rozszerzenia, a następnie w każdym z widoków związanym z pracownikiem HR, wstawiony został w ramach tych sekcji odpowiedni dla nich kod.

Kod 19. Plik menu_HREmployee.blade.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```
<!DOCTYPE html>
<html lang="{{app()->getLocale()}}">

<head>
    <meta charset="UTF-8">
    <meta name="viewport">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>@yield('title')</title>

    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}" defer></script>

    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">

    <!-- Custom css style for PREMSS: -->
    <link href="{{ asset('css/shared.css') }}" rel="stylesheet">
    <link href="{{ asset('css/premss_notepad.css') }}" rel="stylesheet">
</head>

<body>
    <h1 style="text-align: center;" class="premssHeader">
     PREMSS - Enterprise Management Support System</h1>
    <hr>
    <!-- Menu HR employee - Begining -->
    <nav class="navbar navbar-default" role="navigation"
     style="background-color: #fff;">
        <div class="navbar-header">
```

```
            <a class="navbar-brand"
             href="{{ route('userNotesIndexHREmployee') }}"
             style="color: #808080; font-weight: bold;">
             Moje notatki</a>
            <a class="navbar-brand"
             href="{{ route('employeesIndexHREmployee') }}"
             style="color: #808080; font-weight: bold;">
             Lista Pracowników</a>
            <a class="navbar-brand"
             href="{{ route('salary_positions.index') }}"
             style="color: #808080; font-weight: bold;">
             Lista wynagrodzeń</a>
        </div>

        <div class="navbar-ex1-collapse">

            <ul class="nav navbar-nav navbar-right">
                <li class="dropdown">
                    <a href="#" class="dropdown-toggle"
                     data-toggle="dropdown" style="color: #808080;">
                    PREMSS <b class="caret"></b></a>
                    <ul class="dropdown-menu">
                        <li><a href="#" style="color: #808080;"
                             data-toggle="modal"
                             data-target="#aboutPremssDialog">
                             Informacje o systemie</a>
                        <li>
                            <a href="{{ route('logout') }}"
                               style="color: #808080;"> Wyloguj</a>
                        </li>
                    </ul>
                </li>
            </ul>
        </div>

    </nav>

    <!-- HR standard employee - End -->

    <!-- Modal PREMSS -->
    <div class="modal fade" id="aboutPremssDialog" role="dialog">
        <div class="modal-dialog">

            <!-- Modal content-->
            <div class="modal-content" id="aboutPremssDialogContent">

                <div class="modal-header aboutPremssDialogHeaderAndFotter">
                    <button type="button" class="close"
                     data-dismiss="modal">&times;</button>
```

```
                    <h4 class="modal-title" style="position: absolute;">
                     PREMSS - Informacje o systemie</h4>
                </div>

                <div class="modal-body">
                    <p>Autor: Paweł Rzeńca</p>
                    <p>Uczelnia: Polsko-Japońska
                     Akademia Technik Komputerowych (PJATK)</p>
                    <br/>
                    <p>Niniejsze oprogramowanie stanowi część pracy
                     magisterskiej autorstwa Pawła Rzeńcy.</p>
                    <p>Wszelkie prawa zastrzeżone.</p>
                </div>

                <div class="modal-footer aboutPremssDialogHeaderAndFotter">
                    <button type="button" class="btn btn-default"
                     data-dismiss="modal">Zamknij</button>
                </div>

            </div>

        </div>
    </div>

    <main> @yield('content') </main>

</body>

</html>
```

Rysunek 23. PREMSS, PHP i Laravel – Widok z listą pracowników. **Źródło**: Opracowanie własne.

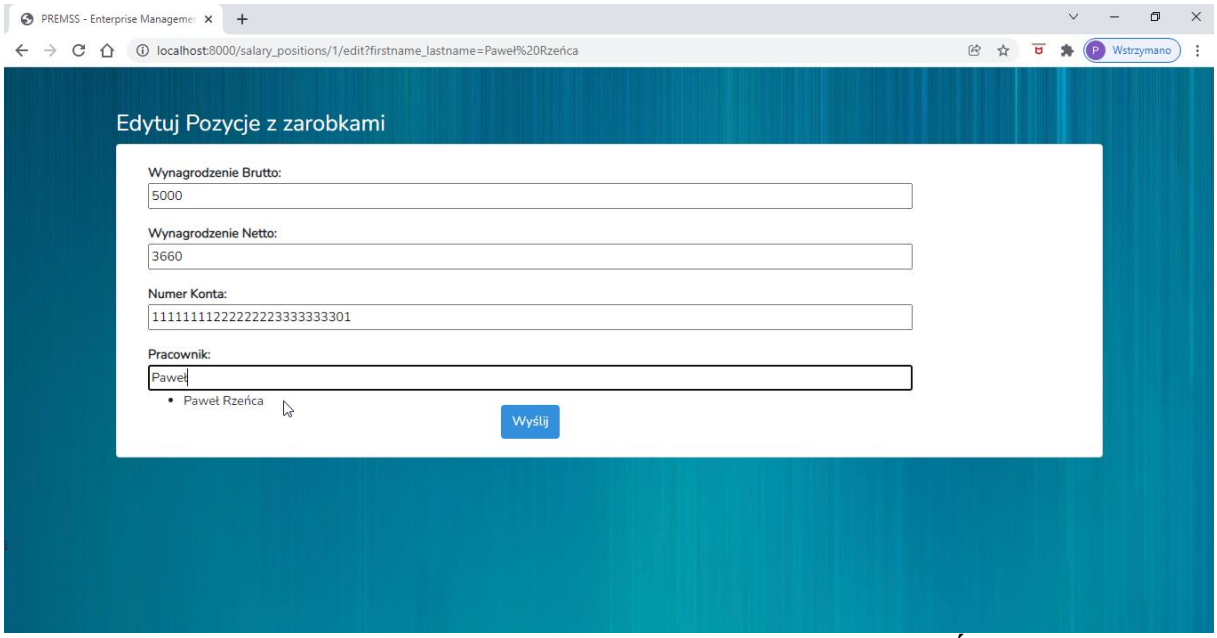
W przypadku pracownika HR, zaimplementowany został również między innymi widok w postaci listy z zarobkami oraz widoki do dodawania i edycji nowych pozycji. Schemat tego pierwszego nie odbiega znacząco od widoku, który obrazuje Rysunek 23. W przypadku tego ostatniego kod źródłowy widoczny jest jako Kod 20. natomiast widok obrazuje Rysunek 24. Kod ukazuje użycie silnika Blade, którego możliwości zostały wykorzystane między innymi do podpięcia akcji formularza czy podstawienia w odpowiednie jego pola aktualnych wartości podczas edycji. Zaimplementowana została tutaj również wyszukiwarka pracowników. Opiera się ona o język JavaScript wraz z biblioteką jQuery i jQuery UI. Działa ona w ten sposób, iż po wpisaniu frazy do pola wyszukiwania, zostaje wysłane żądanie do kontrolera, który w odpowiedzi zwraca listę wyników na podstawie wprowadzonej frazy. Żeby żądanie tego rodzaju zostało obsłużone przez kontroler, dodany został do niego token CSRF. Jest to zabezpieczenie Laravela związane z atakami CSRF.

Kod 20. Plik Edit-SalaryPosition.blade.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```
<!DOCTYPE html>
<html lang="{{app()->getLocale()}}">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta charset="utf-8">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>PREMSS - Enterprise Management Support System</title>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/
    jquery.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/
    jquery-ui.min.js"></script>

    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <!-- Custom css style for PREMSS: -->
    <link href="{{ asset('css/shared.css') }}" rel="stylesheet">
</head>
<body>
    <div class="container my-5">
        <h3 class="additionalWindowHeader"> Edytuj Pozycje z zarobkami</h3>
        <div class="card">
            <div class="card-body">
                <div class="col-md-10">
                    <form action="{{ route('salary_positions.update',
                     $salaryPosition->id) }}" method="POST">
                        @csrf
                        @method('PUT')

                        <div class="row">
                            <div class="col-xs-12 col-sm-12 col-md-12">
                                <div class="form-group">
                                    <strong>Wynagrodzenie Brutto:</strong>
                                    <input style="width: 100%" type="text"
                                name="gross_salary"
                                value="{{ $salaryPosition->gross_salary }}">
                                 </div>
```

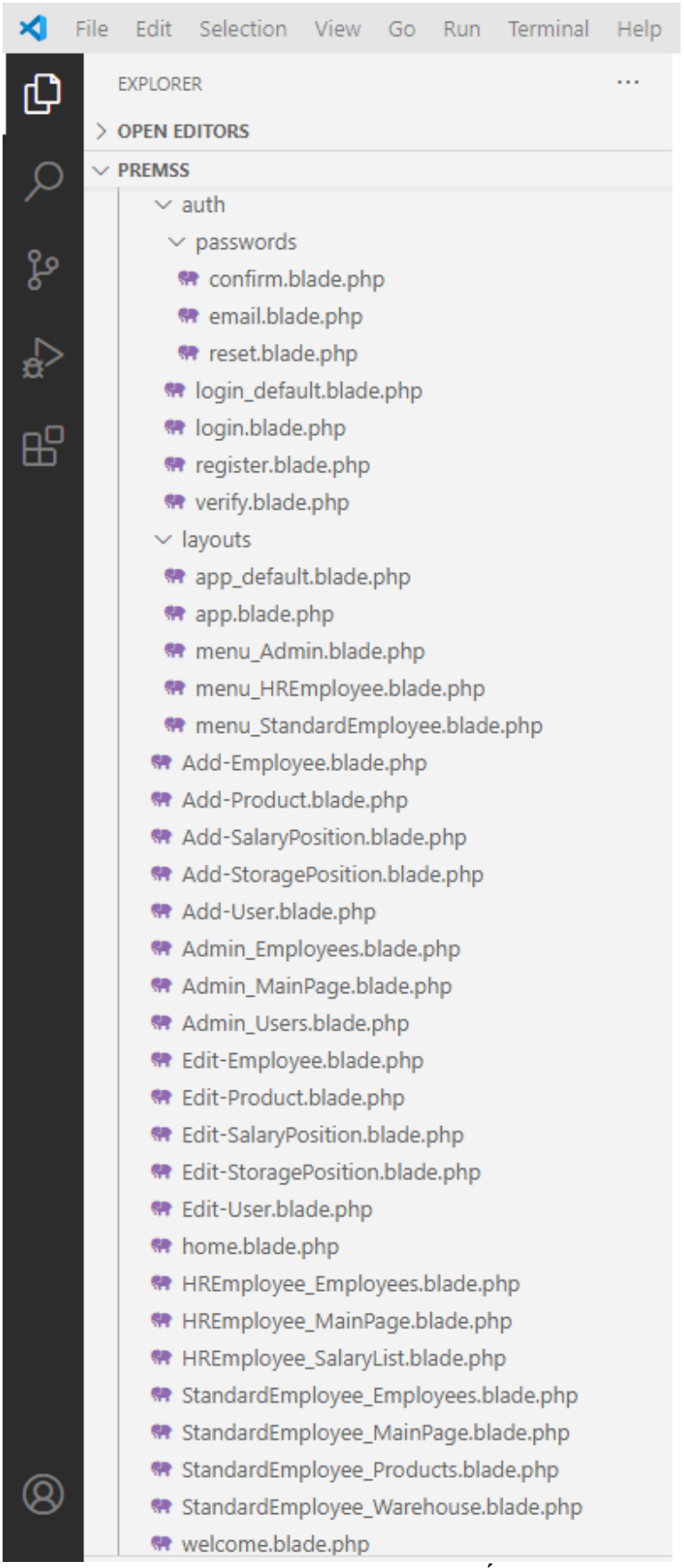
```
                        </div>

                        <div class="col-xs-12 col-sm-12 col-md-12">
                            <div class="form-group">
                                <strong>Wynagrodzenie Netto:</strong>
                                <input style="width: 100%" type="text"
                                 name="net_salary"
                                 value="{{ $salaryPosition->net_salary }}">
                            </div>
                        </div>

                        <div class="col-xs-12 col-sm-12 col-md-12">
                            <div class="form-group">
                                <strong>Numer Konta:</strong>
                                <input style="width: 100%" type="text"
                    name="bank_account_number"
                    value="{{ $salaryPosition->bank_account_number }}">
                            </div>
                        </div>

                        <div class="col-xs-12 col-sm-12 col-md-12">
                            <div class="form-group">
                                <strong>Pracownik:</strong>
                                <input style="width: 100%" type="text"
                      name="firstname_lastname" id="employee_serach"
                      value="{{ Request::get('firstname_lastname') }}">
                                <input style="width: 100%"
                     type="number" name="employee_id"
                     id="employee_id"
                     value="{{ $salaryPosition->employee_id }}" hidden>
                            </div>
                        </div>

                        <div class="col-xs-12 col-sm-12 col-md-12
                         text-center">
                            <button type="submit"
                             class="btn btn-primary">Wyślij</button>
                        </div>
                    </div>

                </form>
            </div>
        </div>
    </div>
</div>

<!-- Emmployee search -->
<script type="text/javascript">
    var CSRF_TOKEN = $('meta[name="csrf-token"]').attr('content');
```

```
        $(document).ready(function() {
            $("#employee_serach").autocomplete({
                source: function(request, response) {
                    $.ajax({
                        url: "{{route('getEmployees')}}",
                        type: 'post',
                        dataType: "json",
                        data: {
                            _token: CSRF_TOKEN,
                            search: request.term
                        },
                        success: function(data) {
                            response(data);
                        }
                    });
                },
                select: function(event, ui) {
                    $('#employee_serach').val(ui.item.label);
                    $('#employee_id').val(ui.item.value);
                    return false;
                }
            });
        });
    </script>
</body>
</html>
```



Rysunek 24. PREMSS, PHP i Laravel – Widok do edycji pozycji z zarobkami. **Źródło**: Opracowanie własne.

Lista wszystkich widoków aplikacji PREMSS widoczna jest na Rysunku 25.



Rysunek 25. PREMSS, PHP i Laravel – Lista widoków. **Źródło**: Opracowanie własne.

6.5.2.4. MVC PHP i Laravel - Model

Model – Laravel wyposażony jest w technologię pozwalającą na mapowanie obiektowo-relacyjne o nazwie Eloquent. Działa ona w taki sposób, że dla każdej tabeli w bazie danych istnieje odpowiadający mu model. Służy on do interakcji z tą tabelą, wspierając takie operacje jak dodawanie, usuwanie, aktualizacja czy pobieranie rekordów z bazy danych [97]. Laravel pozwala na tworzenie schematu modelu przy użyciu narzędzia Artisan. Możliwe jest również przy jego pomocy stworzenie zarówno kontrolera oraz korespondującego modelu przez użycie pojedynczej komendy. Przykładowy zaimplementowany model w ramach oprogramowania PREMSS widoczny jest jako Kod 21. Do jego stworzenia wykorzystane zostało polecenie: '`php artisan make:controller SalaryPositionController --resource --model=SalaryPosition`', za pomocą którego od razu utworzony został szkielet modelu 'SalaryPosition' jak i odpowiadającego mu kontrolera 'SalaryPositionController'.

Klasy będące modelami, dziedziczą po klasie abstrakcyjnej 'Model'. Dla modelu o nazwie 'SalaryPosition' domyślną nazwą tabeli jest 'salary_position'. Żeby to zmienić utworzone zostało pole '`$table`' z przypisaną wartością '`salary_positions`', co pozwoliło na powiązanie modelu właśnie z tą tabelą. Tablica '`$fillable`', pozwala na określenie tych kolumn i co za tym idzie właściwości w modelu, dla których będzie możliwość masowego przypisywania wartości – przykładowo podczas zapisywania do tabeli początkowych danych z poziomu kodu PHP. Przy pomocy modelu Laravela istnieje również możliwość zdefiniowania relacji, które będą zachodzić pomiędzy poszczególnymi modelami na poziomie kodu. W omawianym przykładzie zdefiniowane zostało przy pomocy metody '`belongsTo`', to że jedna pozycja z zarobkami należy do jednego pracownika.

Kod 21. Plik SalaryPosition.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```php
<?php
//(…)
class SalaryPosition extends Model
{
    protected $table = 'salary_positions';
    use HasFactory;

    protected $fillable = [ 'bank_account_number', 'gross_salary',
                            'net_salary', 'employee_id' ];

    public function employees(){
        return $this->belongsTo(Employee::class, 'employee_id', 'id');
    }
}
```

6.5.2.5. MVC PHP i Laravel - Kontroler

Kontroler (ang. *controller*) – W kontrolerze Laravel zdefiniowane są metody, który zawierają kod odpowiedzialny za obsługę poszczególnych ścieżek, związanych z jedną logiczną całością. Kod 22 pokazuje źródła kontrolera związanego z pracownikami - klasa EmployeeController.

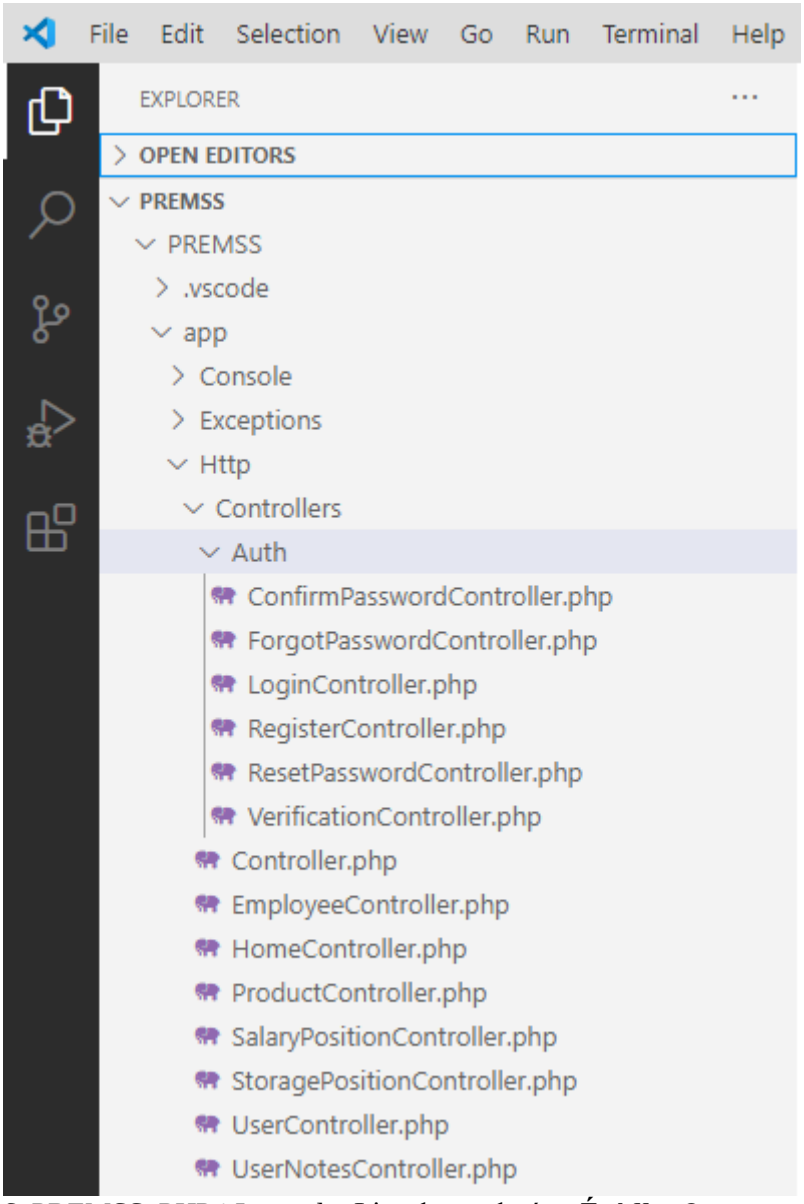
Kontrolery w Laravelu rozszerzają klasę 'Controller'. Na przedstawionym przykładzie widać między innymi:

- miejsca w których zwracany jest widok, przy pomocy pomocniczej funkcji '`view`', wraz z dołączonym modelem. Przykładem jest tutaj metoda '`indexStandardEmployee`'.
- metodę '`store`', która na podstawie dostarczonego żądania (obiekt '`$request`') przy pomocy mechanizmu wstrzykiwania zależności, najpierw waliduje dane, a następnie tworzy nowego pracownika i przekierowuje na ścieżkę o nazwie '`employeesIndexHREmployee`'.

Kod 22. Plik EmployeeController.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```php
<?php
//(…)
class EmployeeController extends Controller
{
//(…)
    public function indexStandardEmployee()
    {
        $employees = Employee::all();
        return view('StandardEmployee_Employees', compact('employees'));
    }

    public function indexAdmin()
    {
        $employees = Employee::all();
        return view('Admin_Employees', compact('employees'));
    }

    public function indexHREmployee()
    {
        $employees = Employee::all();
        return view('HREmployee_Employees', compact('employees'));
    }

    public function create()
    {
        return view('Add-Employee');
    }

    public function store(Request $request)
    {
        $request->validate(['firstname' => 'required',
                'lastname' => 'required', 'phoneNumber' => 'required',
                'emailAddress' => 'required', 'department' => 'required']);

        Employee::create($request->all());

        return redirect()->route('employeesIndexHREmployee');
    }

    public function edit(Employee $employee)
    {
        return view('Edit-Employee', compact('employee'));
    }

    public function update(Request $request, Employee $employee)
    {
        $request->validate(['firstname' => 'required',
```

```
                'lastname' => 'required', 'phoneNumber' => 'required',
                'emailAddress' => 'required', 'department' => 'required']);

        $employee->update($request->all());

        return redirect()->route('employeesIndexHREmployee');
    }

    public function destroy(Employee $employee)
    {
        $employee->delete();

        return redirect()->route('employeesIndexHREmployee');
    }

    public function getEmployees(Request $request)
    {
        $search = $request->search;

        if ($search == '') {
            $employees = Employee::orderby('lastname', 'asc')
            ->select('id','firstname', 'lastname')->limit(5)->get();
        } else {
            $employees = Employee::orderby('lastname', 'asc')
            ->select('id','firstname', 'lastname')
            ->where('lastname', 'like', '%' . $search . '%')
            ->orWhere('firstname', 'like', '%' . $search . '%')
            ->limit(5)->get();
        }

        $response = array();
        foreach ($employees as $employee) {
            $response[] = array("value" => $employee->id,
              "label" => $employee->firstname . " " . $employee->lastname);
        }

        return response()->json($response);

    }

}
```

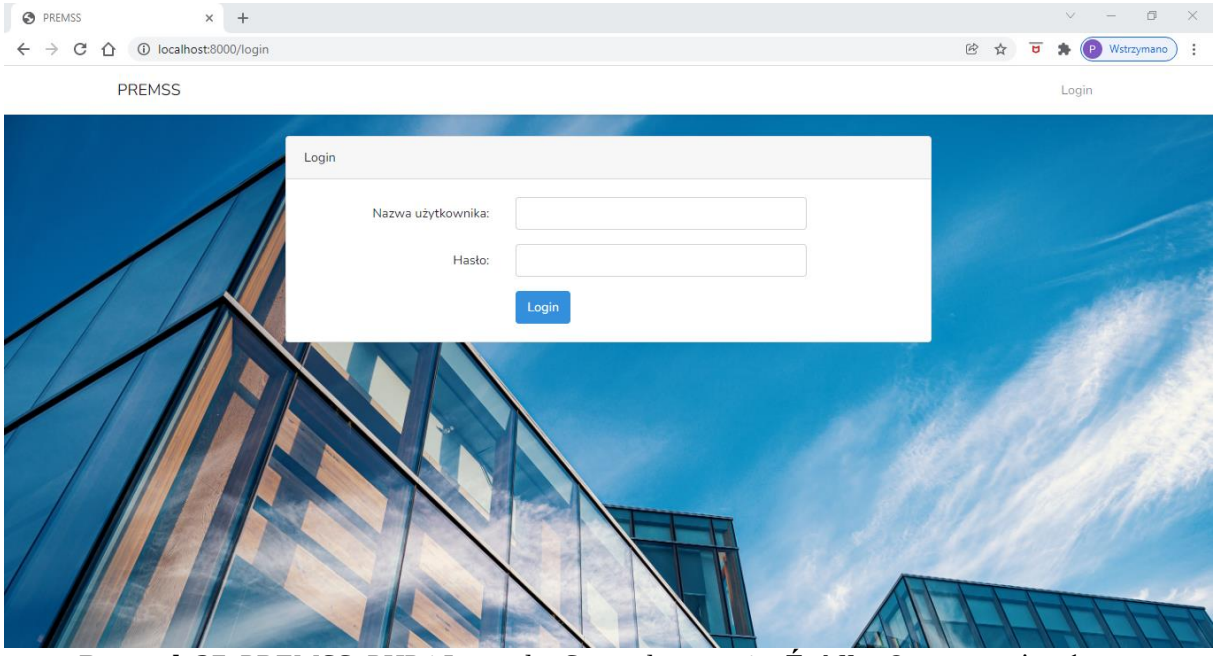


Rysunek 26. PREMSS, PHP i Laravel – Lista kontrolerów. **Źródło**: Opracowanie własne.

Lista wszystkich kontrolerów dla aplikacji PREMSS została zawarta na Rysunku 26.

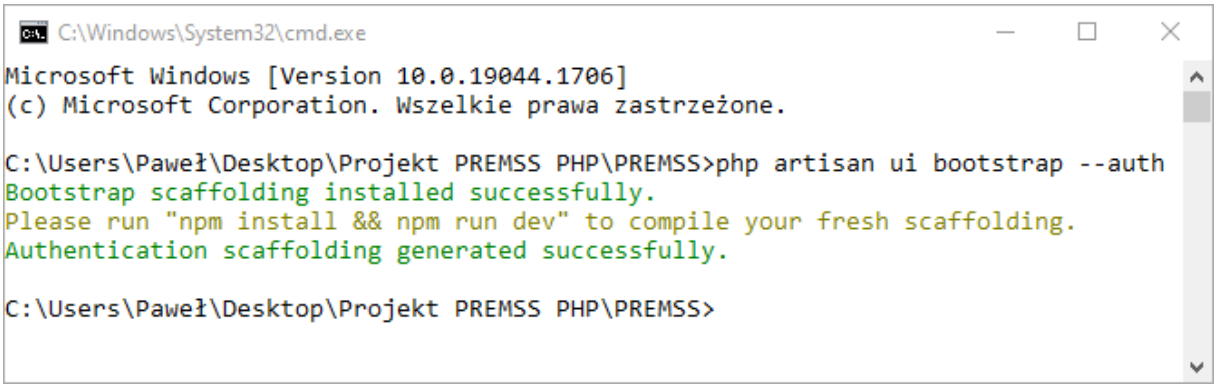
6.5.2.6. Autentykacja i Logowanie - PHP i Laravel

Laravel dostarcza również narzędzia, ułatwiające stworzenie systemu autentykacji i logowania. Jest to często istotna część oprogramowania internetowego i została ona zaimplementowana w omawianej tutaj aplikacji. Strona do logowania została przedstawiona na Rysunku 27.



Rysunek 27. PREMSS, PHP i Laravel – Strona logowania. **Źródło**: Opracowanie własne.

W celu dodania komponentów związanych z autoryzacją i logowaniem użyty został interfejs linii komend Artisan. Została w nim wpisana komenda, która służy do utworzenia szkieletu systemu autentykacji oraz interfejsu użytkownika z wykorzystaniem Bootstrapa. Rysunek 28 przedstawia ten proces oraz obrazuje sposób używania narzędzia Artisan.



Rysunek 28. PREMSS, PHP i Laravel – Komenda dla stworzenia szkieletu systemu autentykacji. **Źródło**: Opracowanie własne.

Na Rysunku 28 widać, iż po zainstalowaniu wcześniej wspomnianej paczki, otrzymana została informacja dotycząca komend '`npm install`' oraz '`npm run dev`'. Są one związane są menadżerem pakietów Npm dla języka JavaScript i zostały wykonane w następnej kolejności.

Z systemem autoryzacji związane są między innymi takie pliki jak:

- Plik auth.php – Zawiera konfiguracje autoryzacji Laravela.
- Kilka plików dla klas będących kontrolerami, związanymi z autoryzacją. Ich lista widoczna jest na Rysunku 26, gdzie te związany z autoryzacja, znajdują się w katalogu 'Auth'.
- W pliku 'web.php', którego źródła zostały przedstawione jako Kod 17, znajduje się następująca linia '`Auth::routes();`'. Jest ona odpowiedzialna za rejestrację ścieżek, związanych z autoryzacją, przy pomocy klasy Auth i jej statycznej metody '`routes`'.
- Pliki widoków związanych z autoryzacją. Są one widoczne na Rysunku 25, w podkatalogu 'auth', znajdującym się katalogu 'views'.
- Model 'User', gdzie zdefiniowany jest model dla użytkownika.

Domyślne autentykacja w Laravelu odbywa się przy użyciu adresu e-mail. W wyniku wykonania komendy związanej z dołączeniem komponentu związanego z autentykacją, dodany został widok związany z takim schematem, gdzie to użytkownik rejestruje swoje konto oraz określa swoje dane. Dostarczony model użytkownika czy widok związany z logowaniem był dostosowany do tego takiego podejścia. Nie było to zgodne z wymaganiami, jakie zostały postawione przed oprogramowaniem i gdzie określone zostało, że to administrator dodaje nowych użytkowników, a logowanie następuje na podstawie nazwy użytkownika oraz hasła. A to ostatnie powinno być przetrzymywane w bazie danych w zakodowanej formie. W celu dostosowania autentykacji do wymagań między innymi stworzony został nowy formularz logowania (na Rysunku 27 przedstawiony został efekt), zmodyfikowany model użytkownika oraz zaimplementowany został odpowiedni kod kontrolera logowania (LoginController). W tym ostatnim napisany został kod, który zgodnie z wymaganiami przekierowuje uwierzytelnionego użytkownika do odpowiedniego widoku na podstawie jego roli czy metoda odpowiedzialna za wylogowanie – fragmenty za to odpowiedzialne, wylistowane zostały jako Kod 23.

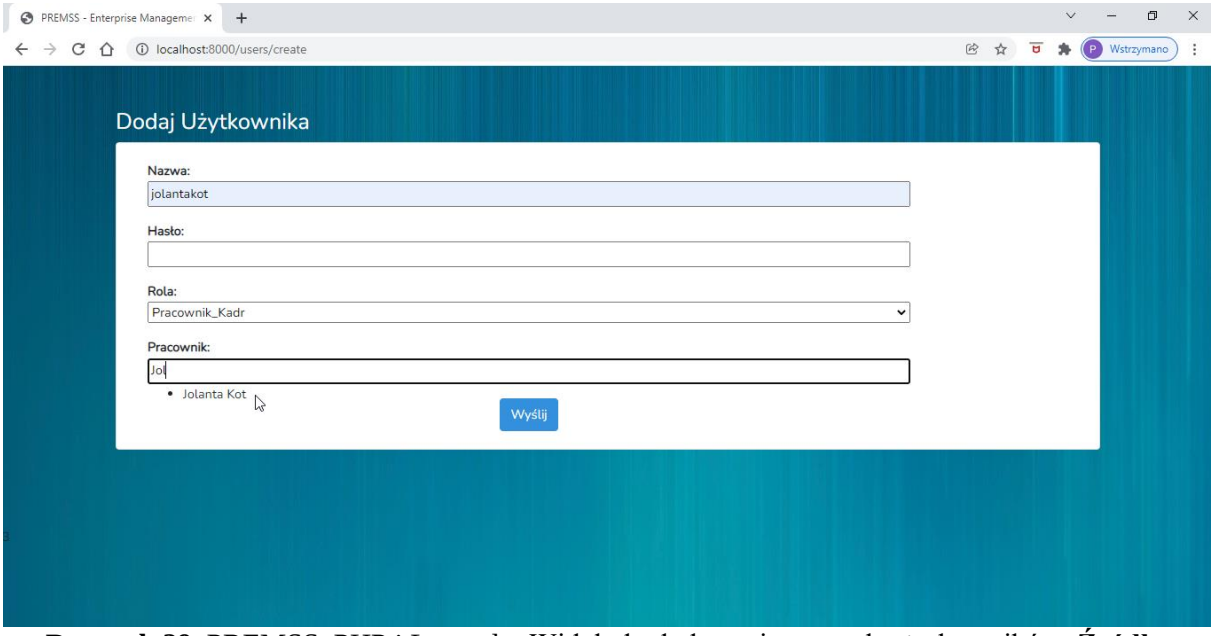
Kod 23. Plik LoginController.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```php
<?php
//(…)
class LoginController extends Controller
{
//(…)
    //Method for change default authorization based on email,
    //to authorization based on username:
    public function username()
    {
        return 'name';
    }

    //Function for handling redirection:
    protected function redirectTo()
    {
        if (Auth::user()->role == "admin") {
            return route('userNotesIndexAdmin');
        } else if (Auth::user()->role == "standard_employee") {
            return route('userNotesIndexStandardEmployee');
        } else if(Auth::user()->role == "hr_employee"){
            return route('userNotesIndexHREmployee');
        }
        else {
            dd('Wystąpił błąd dla roli: '
            . Auth::user()->role . " skontaktuj się z administratorem");
        }
    }

    //Logout handler:
    public function logout(Request $request){
        Auth::logout();
        return redirect('/login');
    }
}
```

Za obsługę operacji dotyczących użytkowników odpowiedzialny jest ‘UserController’. Z mechanizmami autoryzacji, wiąże się natomiast kodowanie haseł. W Laravelu istnieje możliwość wywołania statycznie na klasie Hash, metody ‘`make`’ która pomaga w łatwym kodowaniu haseł, które następnie trafiają do bazy danych. Zostało to wykorzystane we wcześniej wspomnianym kontrolerze. Dla zobrazowania tego, w Kodzie 24 przedstawiona została metoda, której celem jest zapisywanie nowych użytkowników do bazy danych, co wykonywane jest na podstawie informacji z formularza, służącego do dodawania nowych użytkowników. W metodzie tej tworzona jest również początkowa notatka, dla nowego użytkownika. Widok wspomnianego formularza znajduje się na Rysunku 29.

Kod 24. Plik UserController.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```php
<?php
//(…)
class UserController extends Controller
{
//(…)
    public function store(Request $request)
    {
        $request->validate(['name' => 'required',
                            'password' => 'required', 'role' => 'required',
                            'employee_id' => 'required']);

        $password = $request->input('password');
        $hashedPassword = Hash::make($password);

        $request['password'] = $hashedPassword;

        $createdUser = User::create($request->all());

        //Create related User Notes record:
        $userNotes = new UserNotes();
        $userNotes->user_notes = "Tu jest miejsce na Twoje notatki.";
        $userNotes->user_id = $createdUser->id;
        $userNotes->save();

        return redirect()->route('users.index');
    }

}
```



Rysunek 29. PREMSS, PHP i Laravel – Widok do dodawania nowych użytkowników. **Źródło**: Opracowanie własne.

Istnieje możliwość korzystania z serwera deweloperskiego, który dostępny jest po instalacji Laravela. Do jego odpalenia służy polecenie '`php artisan serve`'. Ta możliwość była wykorzystywana w trakcie implementowania oprogramowania PREMSS w wersji omówionej w niniejszym rozdziale.

7. Analiza porównawcza

Niniejszy rozdział skupia się na analizie poświęconej zbadaniu kwestii łączących język Java z językiem PHP oraz znalezieniu tych cech, które różnią te dwa języki programowania. Zebrane i porównane zostały w nim dane dotyczące zarówno ich popularności jak i faktycznego wykorzystania po stronnej serwerowej. Następnie na podstawie informacji zawartych w rozdziałach poświęconych wspominanym językom, przenalizowane zostało ich zastosowanie. Również rozdziały poświęcone sposobom ich przetwarzania, dostarczyły materiału do wskazania różnić i podobieństw z tego wynikających. W rozdziale tym można znaleźć także porównanie dotyczące tematu wspierania programowania współbieżnego przez oba języki, co wydaje się istotne chociażby ze względu na obecną architekturę procesorów. Porównaniu uległ również sposób podejścia do pisania kodu, znajdowania błędów w programach, procesu debugowania oraz inne cechy i mechanizmy obydwu języków.

Kolejny podrozdział skupia się natomiast na porównaniu w Springa oraz Laravela, z których użyciem zaimplementowane zostało programy opisane w rozdziale szóstym. Stanowią one bazę do przeprowadzenia analizy porównawczej w obrębie najbardziej popularnego frameworka Javy oraz najbardziej popularnego frameworka PHP, których przeznaczeniem jest wsparcie w wytwarzaniu aplikacji internetowych.

Podrozdział 7.3 zawiera spojrzenie na kwestię użycia zasobów sprzętowych i wydajności rozwiązań opartych o obydwie technologie.

Z uwagi na dynamiczną zmienność w zakresie technologii związanych z programowaniem, autor niniejszej pracy, pragnie odnotować, iż poniższe podrozdziały opisują aktualny znany mu stan na chwilę jej tworzenia, gdzie ostatnia ich aktualizacja przypada na przełom stycznia i lutego 2022 roku

7.1. Część I – Podobieństwa i różnice pomiędzy językiem Java oraz językiem PHP

Oba języki mają pewne cechy wspólne jak i występują pomiędzy nimi różnice. Kwestią, której należałoby się przyjrzeć na początku, jest popularność języka. Jednym z kryteriów oceny tego, może być indeks TIOBE. Jest to indeks pozwalający ocenić skalę popularności danego języka, na podstawie danych internetowych. Liczy się go między innymi na podstawie ilości przeszukiwań oraz ich rezultatów w najpopularniejszych silnikach wyszukiwania takich jak Google. Wyrażony jest on w punktach procentowych, gdzie suma wszystkich ocenianych języków wynosi sto procent [98]. Poniżej znajduje się tabela 6 obrazująca indeks TIOBE oraz zmiany jakie w nim zachodziły na przestrzeni stycznia 2021 do stycznia 2022 [99]:

Tabela 6. Indeks TIOBE w styczniu 2022. **Źródło**: [99].

Język programowania	**Miejsce w styczniu 2021**	**Miejsce w styczniu 2022**	**Wartość wskaźnika**	**Zmiana procentowa**
Python	3	1	13,58%	+1,86%
C	1	2	12,44%	-4,94%
Java	2	3	10,66%	-1,30%
C++	4	4	8,29%	+0,73%
C#	5	5	5,68%	+1,73%
Visual Basic	6	6	4,74%	+0,90%
JavaScript	7	7	2,09%	-0,11%
Assembler	11	8	1,85%	+0,21%
SQL	12	9	1,80%	+0,19%
Swift	13	10	1,41%	-0,02%
PHP	8	11	1,40%	-0,60%
R	9	12	1,25%	-0,65%

Obecnie najbardziej popularnym językiem według indeksu TIOBE jest Python. Powyższe dane pozwalają na uznanie, iż język Java jest bardziej popularny niż PHP. Zajmuje on trzecie miejsce z wynikiem 10,66%, a PHP jedenastą pozycję z wartością 1,40%. Warto również odnotować, iż jeden i drugi ma tendencje spadkową, jednak w przypadku PHP tendencja ta jest mniej dynamiczna.

Nieco inaczej kształtuje się sytuacja, jeśli spojrzymy na dane dostarczone przez portal Stackoverflow.com i ich raport z roku 2021. Opiera się on na ogólnoświatowej ankiecie przeprowadzonej przez wspomniany portal na próbie 83052 respondentów, która odbyła się pomiędzy 25 maja, a 15 czerwca 2021. Pytanie zawarte w ankiecie brzmiało „W których językach programowania, skryptowania i znaczników wykonywałeś intensywne prace w ciągu ostatniego roku i w których zamierzasz wykonywać pracę w roku przyszłym?". Tabela 7 przedstawia wyniki tego badania [100]:

Tabela 7. Wyniki ankiety dotyczącej najczęściej używanych języków programowania, skryptowych oraz znaczników. **Źródło**: Opracowanie własne na podstawie danych znajdujących się na stronie ankiety Stackoverflow z roku 2021 [100]

Język programowania	**Udział procentowy**
JavaScript	64,96%
HTML/CSS	56,07%
Python	48,24%
SQL	47,08%
Java	35,35%
Node.JS	33,91%
TypeScript	30,19%
C#	27,86%
Bash/Shell	27,13%
C++	24,31%
PHP	21,98%
C	21,01%

Dane te również potwierdzają, iż w ogólności to język Java jest bardziej popularny niż język PHP. Ten pierwszy otrzymał 29162 głosów co przełożyło się na 35,35%, a drugi otrzymał ich1 18310 co natomiast dało wynik w postaci 21,98%. Wskazują one jednak na mniejszą różnice niż obrazować może indeks TIOBE. Na podstawie przytoczonych danych z Tabeli 6 oraz 7 można uznać, iż obydwa języki cieszą się stosunkową dużą popularnością i znajdują się w jej szerokiej czołówce.

Niniejsza praca traktuje jednak w szczególności o oprogramowaniu internetowym. Danych na temat użycia języków programowania po stronie serwerowej dostarcza witryna W3Tech.com i znalazły one odzwierciedlenie w Tabeli 8 [101]:

Tabela 8. Udział poszczególnych języków programowania używany po stronie serwerowej dla stron internetowych. **Źródło**: [101].

Język programowania	**Udział procentowy w ogóle języków programowania używanych po stronie serwerowej**
PHP	78%
ASP.NET	7,9%
Ruby	6,0%
Java	3,7%
Scala	2,3%
JavaScript	1,8%

Język PHP znajduje się na pierwszym miejscu z wynikiem 78% i ma zdecydowanie największy udział w językach zastosowanych po stronie serwerowej. Java znajduje się na miejscu czwartym i jej rezultat na dzień 22.01.2022 to 3,7%. Różnica jest więc bardzo znacząca i PHP jest tutaj językiem, który można uznać za dominujący.

Z pewnością kwestią je łączącą jest to, że są powszechnie wykorzystywane do tworzenia oprogramowania internetowego. Jednak Java jest również często wykorzystywana do tworzenia innego rodzaju programów, takich jak aplikacje mobilne (technologia Android) czy aplikacje okienkowe (technologia JavaFX, SWING), czy nawet aplikacje osadzone (platforma JavaCard). Język PHP jest mniej wszechstronny, możliwym jest przy jego zastosowaniu stworzenie aplikacji okienkowej, jednak tak jak zostało opisane w rozdziale, który jest mu poświęcony, ostatnie aktualizacje projektów wspierających takie zastosowanie miały miejsce w roku 2015. Wykorzystywany jest on przede wszystkim do tworzenia stron internetowych.

Istotną różnicą jest proces obsługi kodu źródłowego – w przypadku Javy programista po jego napisaniu kompiluje go. W przypadku języka PHP nie zachodzi taki etap. Ma to wpływ na to, iż podczas kompilacji kodu w przypadku tego pierwszego dostarczone zostaną informacje o błędach. Poniżej znajdują się dwa przykłady prostego kodu stworzonego w języku PHP (Kod 25) oraz drugi analogiczny stworzony w języku Java (Kod 26). W jednym i drugim przykładzie znajduje się zamierzony błąd:

Kod 25. Kod PHP zawierający błąd. **Źródło**: Opracowanie własne.

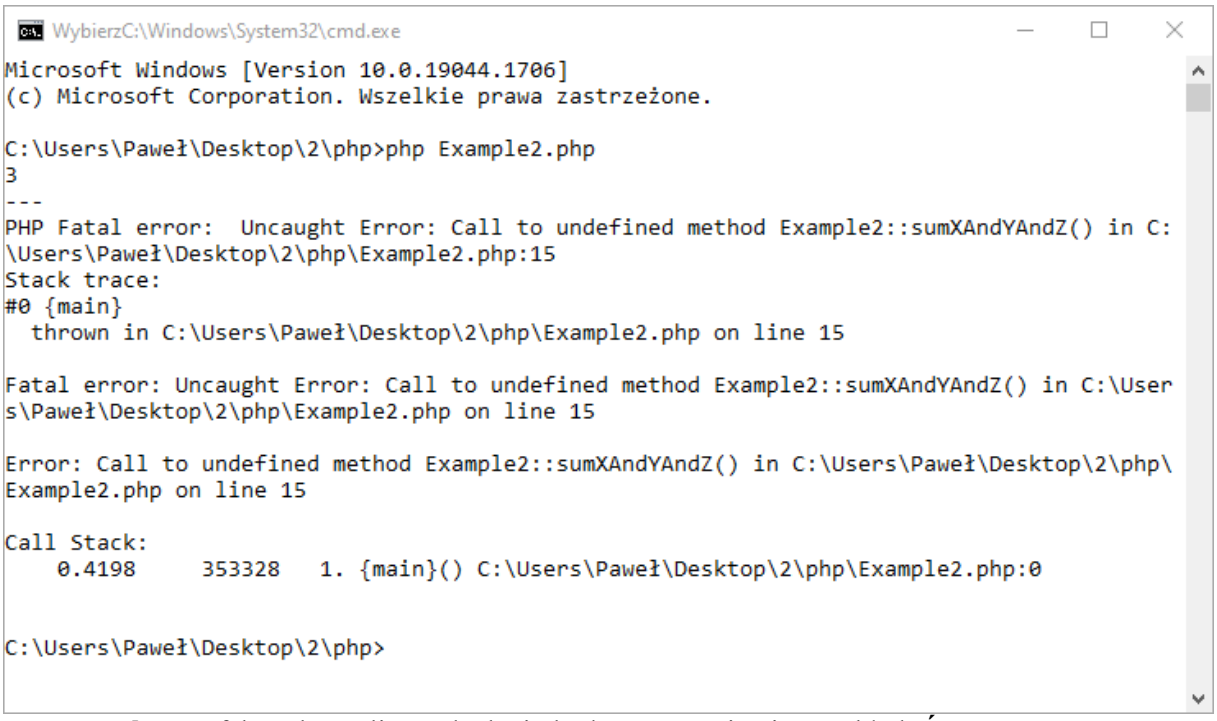
```php
<?php
class Example2 {
    private $x = 1;
    private $y = 2;

    public function sumXAndY(){
        return $this->x + $this->y;
    }
}

$example2TestObject = new Example2();
echo($example2TestObject->sumXAndY());
echo("\n---\n");
echo($example2TestObject->sumXAndYAndZ());
```

Kod 26. Kod Java zawierający błąd. **Źródło**: Opracowanie własne.

```java
public class Example2 {
    private Integer x = 1;
    private Integer y = 2;

    public Integer sumXAndY(){
        return this.x + this.y;
    }

    public static void main(String[] args){
        Example2 example2TestObject = new Example2();
        System.out.println(example2TestObject.sumXAndY());
        System.out.println("---");
        System.out.println(example2TestObject.sumXAndYAndZ());
    }
```

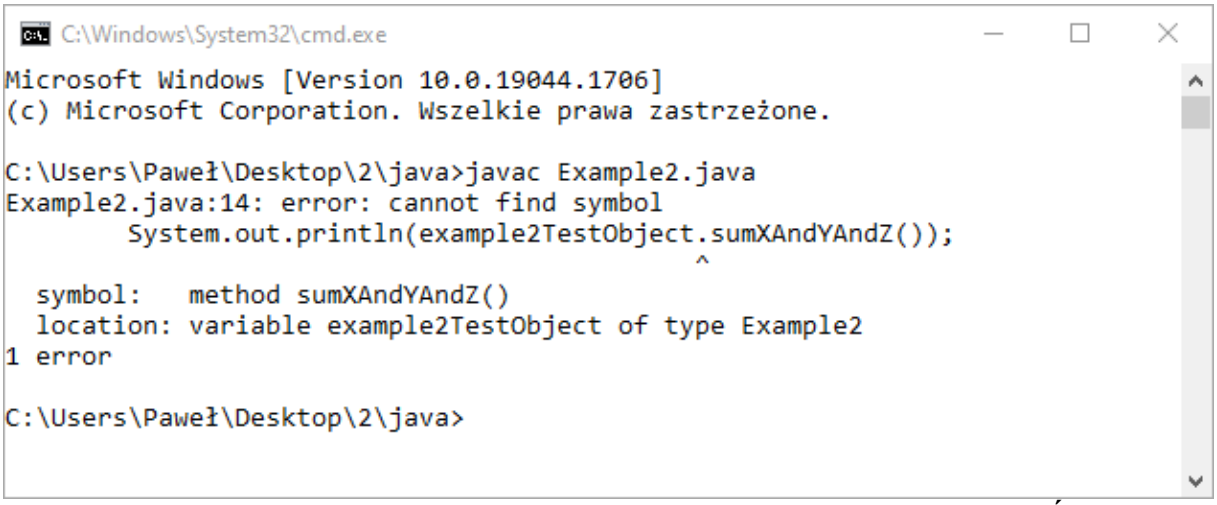
```
}
```

Po uruchomieniu pierwszego kodu efekt w konsoli obrazuje Rysunek 30:



Rysunek 30. Efekt w konsoli po odpaleniu kodu PHP zawierającego błąd. **Źródło**: Opracowanie własne.

Natomiast w przypadku Javy, przed uruchomieniem kodu musi on ulec wstępnej kompilacji. Efekt po wpisaniu polecenia służącego do skompilowania kodu Javy widać na zrzucie ekranu na Rysunku 31:



Rysunek 31. Efekt w konsoli po próbie kompilacji kodu Java zawierającego błąd. **Źródło**: Opracowanie własne.

Jak widać efekt w tych dwóch sytuacjach jest różny. Kod w przypadku języka PHP został wykonany do momentu wywołania metody, która nie została zdefiniowania – kod metody, która odpowiedzialna jest za dodanie dwóch liczb został zrealizowany. Jeśli chodzi o Javę to już na etapie kompilacji zostało wykryte, że występuje wywołanie metody, która nie istnieje. Doprowadziło to zwrócenia błędu i w efekcie nie powstał kod bajtowy Javy o rozszerzeniu ‘class’, który mógłby zostać następnie wykonany. Niesie to za sobą takie skutki, że w przypadku Javy już we wczesnej fazie,

programista ma możliwość uzyskania informacji o błędach, o których w przypadku PHP dowie się dopiero podczas wykonywania kodu. O ile powyższy przykład przedstawia prostą sytuację i programy, to jeśli chodzi o komercyjne oprogramowanie, może istnieć dużo bardziej skomplikowany i nieoczywisty przypadek. Możliwe, iż programista Java dowie się o problemie już na etapie kompilacji, a programista PHP w pesymistycznym przypadku otrzyma informacje zwrotną dopiero od użytkownika końcowego.

Biorąc pod uwagę proces przetwarzania kodu warto zwrócić również uwagę na różnicę występującą w kodzie pośrednim. W przypadku Javy ma on postać plików z rozszerzeniem '.class'. Natomiast jeśli chodzi o PHP bazują one tylko na mechanizmie pamięci podręcznej, gdzie przetrzymywane są kody operacyjne. Ma to wpływ na związanie z konkretnymi miejscami w pamięci na danej maszynie. To z kolei znajduje odzwierciedlenie w przenaszalności kodu, gdzie w przypadku Javy po zmianie kodu źródłowego na kod pośredni w wyniku kompilacji, istnieje możliwość przeniesienia plików z kodem bajtowym na inną maszynę, posiadającą analogiczną wirtualną maszynę Javy. W przypadku PHP nie istnieje podobna możliwość takiej migracji kodu pośredniego.

Jak widać na wcześniejszym przykładzie dotyczącym Javy (Kod 26), została stworzona przykładowa klasa, w której zawarta jest statyczna metoda '`main`'. Java została stworzona z myślą o obiektowym paradygmacie programowania i aby stworzyć program napisany w tym języku konieczne jest stworzenie przynajmniej jednej klasy z wymaganą metodą główną. Jeśli chodzi o język PHP to mechanizmy związane z obiektowością, były dopiero z czasem stopniowo do niego dodawane. Jest w nim możliwe stworzenie w pełni sprawnego kodu, który nie będzie posiadał ani jednej klasy i metody, zdefiniowanej przez użytkownika. Bazując na poprzednim przykładzie i eliminując już z niego wspomniany błąd, kod w przypadku PHP może wyglądać również w sposób zobrazowany w Kodzie 27:

Kod 27. Prosty skrypt PHP. **Źródło**: Opracowanie własne.

```php
<?php
$x = 1;
$y = 2;

function sumXAndY()
{
    global $x, $y;
    return $x + $y;
}

echo (sumXAndY());
```

Efektem po odpaleniu Kodu 27 przez interfejs linii komend jest zwrócenie wyniku: '3'. Jak widać skrypt nie posiada zdefiniowanej żadnej klasy ani metody (istnieje funkcja). W przypadku Javy napisanie kodu w taki sposób, jest niemożliwe.

Poprzednie przykłady pokazują inną, bardzo istotną różnicę pomiędzy omawianymi językami, a jest nią typowanie. Pisząc kod PHP programista nie jest zobligowany do podawania typów zmiennych ani do podawania typów zwracanych przez metody, czy też określania rodzaju przyjmowanych przez nie argumentów – nazywane jest to typowaniem dynamicznym. Java jest natomiast silnie typowana. Gdyby programista pominął podanie typu '`Integer`' w Kodzie numer 26, wylistowanym w niniejszym rozdziale to wyskoczyłby mu błąd. Sprawdzenie typów następuje w Javie już na etapie kompilacji. Odnotować jednak należy fakt, iż istnieją zarówno mechanizmy w PHP pozwalające programiście na określenie w sposób jawny typów, jak w i przypadku Javy na jawne wykorzystanie typowania dynamicznego, nie są to jednak rozwiązania leżące u podstawowych założeń tych języków.

Z możliwością wykonywania kodu bez jego wcześniejszej kompilacji wiąże się również szybkość procesu debugowania. W Javie przed każdym przystąpieniem do tej czynności, trzeba kod najpierw skompilować. Pisząc kod w języku PHP wystarczy, że jego twórca postawi *breakpoint*, a następnie wykona skrypt. Istnieją sytuację, gdzie występuje potrzeba sprawdzenia większej ilości rozwiązań programistycznych i w przypadku PHP będzie to zajmowało mniej czasu.

Pomiędzy językiem Java a PHP występują również różnice dotyczące przetwarzania współbieżnego. Twórcy języka Java stworzyli białą księgę (ang. *white book*), w której znajdują się kluczowe hasła, jakie przyświecały podczas jego tworzenia – jednym z jedenastu haseł jest wielowątkowość (ang. *multithreading*). Był to jeden z pierwszych powszechnych języków programowania, która wspierał współbieżność. Jednak ta cecha języka, w czasach jego powstawania nie była związania z wielordzeniowością procesorów, a z potrzebą wykorzystania korzyści z niej płynących do wykorzystania procesorów, gdy te nie były zajęte pracą, tak żeby dostarczyć użytkownikom nieprzerwanego działania interfejsów użytkownika [102]. Obecnie ma ona tym większe znaczenie, bowiem procesory, posiadające wiele rdzeni stały się powszechnie używane w wszelkiego rodzaju urządzeniach i korzystnym jest wykorzystanie możliwości z tego płynących. Należy uznać, iż u podstaw Javy znajduje się wsparcie dla wielowątkowości podczas gdy w przypadku PHP takie podejście z pewnością nie leży u podstaw tego języka.

W książce, w której współautorem jest Rasmus Lerdorf, czyli twórca języka PHP możemy znaleźć informację w rozdziale poświęconemu rozszerzeniom, iż wersja PHP na system operacyjny Windows nie wspiera wielowątkowych skryptów [103]. Znamienny i pozostający w kontraście do Javy jest już sam fakt, iż opis ten znalazł się w sekcji dotyczącej rozszerzeń, a nie integralnej części języka. Wymaga jednak odnotowania to, iż wydanie wspomnianej książki miało miejsce w roku 2013. Od tego czasu doszło rozszerzenie ‘pthreads’, które związane jest z wielowątkowością. Na oficjalnej stronie PHP w sekcji rozszerzeń widnieje jednak informacja, że jest ono uznawane za obecnie nieutrzymywane i martwe. Dodatkowo, można tam znaleźć ostrzeżenie, iż nie może być ono stosowane w środowisku serwera www, a przeznaczone jest jedynie dla aplikacji ograniczających się do interfejsu linii komend [104]. Jednym z innych rozwiązań, które również jest dodatkowym elementem i wymaga dodatkowej instalacji, jest rozszerzenie ‘parallel’ związane z wątkami interpretera PHP. Dostrzec należy jednak informację na oficjalnej stronie PHP w sekcji jemu poświęconej, iż jest on przeznaczony dla wersji od PHP 7.2 wzwyż [105], gdy jednak wejdziemy na stronę GitHub (adres tego repozytorium umieszczony jest na oficjalnej stronie rozszerzeń PHP – PECL [106]), gdzie znajdują się jego źródła to znajdziemy informacje, iż wsparcie dla wersji 7 zostało porzucone i obecnie jest ono dedykowane dla wersji PHP 8 [107]. Warto tutaj przytoczyć dane zaprezentowane w Tabeli 3, znajdującej się na stronie 23 mówiące, że udział PHP 8 wynosi 2,0%, a dla porównania samej wersji 5 - 27,7 %. Autor niniejszej pracy uważa, iż pomimo że istnieją wspomniane oraz inne rozwiązania to jednak prawidłowym jest uznanie, iż na chwilę obecną wsparcie dla przetwarzania współbieżnego w PHP, nie jest w przypadku tego języka równe językowi Java. W przypadku tego drugiego w Standardowej Platformie Javy (Java SE), możemy znaleźć przykładowo taką klasę jak ‘Thread’ czy inne rozwiązania, gdzie nie występuje potrzeba instalacji żadnych dodatkowych rozszerzeń dla wsparcia przetwarzania współbieżnego [108].

Zarówno w przypadku Javy tak jak i w przypadku PHP, jak zostało wspomniane w rozdziałach im poświęconych istnieje mechanizm Garbage Collector. Są one zatem podobne kwestii zarządzania pamięcią, ponieważ posiadają automatyczny sposób przeznaczony do jej zwalniania. Stoją one natomiast w opozycji do takich języków jak C czy C++, gdzie to programista odpowiedzialny jest za to, aby czyścić w odpowiednim momencie przydzielone wcześniej zasoby pamięciowe.

Jako wadę języka PHP można określić brak spójnej konwencji nazewnictwa funkcji bibliotecznych. Przykładowo część metod zapisana jest w konwencji, gdzie występuje podkreślenie ‘_’, przykładem może być tutaj funkcja ‘array_push’ [109], a inne podkreślenia nie posiadają i w języku znajduje się również funkcja ‘gettype’ [110]. W przypadku Javy konwencja nazewnictwa metod, znajdujących się w oficjalnych bibliotekach jest bardziej spójna.

Na koniec warto zaznaczyć, że zarówno Java jak i PHP są językami posiadającymi bogatą dokumentację, ilość materiałów jak i książek im poświęconym wraz z liczną społecznością wokół nich zbudowaną.

7.2. Część II – Podobieństwa i różnice pomiędzy frameworkiem Spring oraz frameworkiem Laravel

Rozpatrując podobieństwa i różnice, które występują pomiędzy frameworkami Spring oraz Laravel, pierwszą kwestią, którą można porównać jest popularność obydwu technologii. W Tabeli 9 zostały umieszczone wyniki badania przeprowadzonego przez portal Stackoverflow.com w roku 2021. Zostało ono wykonane na podstawie ankiety, w której zostało określone pytanie „W którym internetowym frameworku lub bibliotece wykonywałeś intensywne prace w ciągu ostatniego roku i w których zamierzasz wykonywać pracę w roku przyszłym?”. Wzięło w niej udział 67593 anektowanych [111].

Tabela 9. Wyniki ankiety dotyczącej najczęściej stosowanych internetowych platform programistycznych. **Źródło**: [111].

Framework internetowy	**Udział procentowy**
React.js	40,14%
jQuery	34,42%
Express	23,82%
Angular	22,96%
Vue.js	18,97%
ASP.NET Core	18,10%
Flask	16,14%
ASP.NET	15,74%
Django	14,99%
Spring	14,56%
Angular.js	11,49%
Laravel	10,12%
Ruby on Rails	7,04%
Gatsby	3,97%
FastAPI	3,88%
Symfony	3,85%
Svelte	2,75%
Drupal	2,39%

Platforma programistyczna Spring we wspomnianym badaniu uzyskała wynik w postaci 14,56%, a Laravel otrzymał 10,12% głosów, wyprzedzając przy tym inny framework PHP – Symfony. Ankieta pokazała, iż obie technologie cieszą się stosunkowo dużą popularnością i są zarazem najpopularniejszymi internetowymi platformami programistycznymi dla języków Java oraz PHP. Ukazuje ona jednak również, iż przykładowo konkurencyjne technologie Microsoftu w postaci ASP.NET Core oraz poprzedniczki ASP.NET są jeszcze bardziej popularnymi frameworkami.

W Springu i Laravelu występuje zróżnicowanie jeśli chodzi o sposób dodawania nowych komponentów takich jak modele, widoki czy kontrolery. W omawianym frameworku Javy standardowym podejściem jest po prostu utworzenie nowego pliku, który przykładowo będzie klasą, a następnie dodanie takich komponentów jak odpowiednie adnotacje, pola czy metody [112] [113]. Natomiast w przypadku frameworka PHP do utworzenia plików i szkieletów, wspomnianych oraz

innych komponentów takich jak przykładowo migracje, wykorzystywane jest narzędzie Artisan. Zostało to opisane w rozdziale poświęconemu praktycznej implementacji opartej o język PHP i platformę programistyczną Laravel.

Kolejną istotną różnicą jest podejście do obsługi ścieżek. Laravel posiada mechanizm ‘routes’, gdzie ścieżki definiowane są w oddzielnych plikach. W przypadku zaimplementowanego oprogramowania w ramach niniejszej pracy jako Kod 17 został przedstawiony plik ‘web.php’, gdzie określone zostały ścieżki. W przypadku Springa obsługa ścieżek, czyli ich powiązanie z odpowiednią częścią kodu dzieje się bezpośrednio w kontrolerze. To w nim przy użyciu adnotacji następuje powiązanie konkretnej ścieżki z daną metodą, co zostało opisane i przedstawione w ramach kontrolera pracownika – Kod 11.

Wykonana praktyczna implementacja oprogramowania internetowego wraz z opisami poszczególnych technologii pokazała natomiast, że zarówno Spring jak i Laravel dostarcza skutecznych narzędzi do implementacji opartej na wzorcu MVC. Obydwa frameworki posiadają mechanizmy, które dostarczają ułatwień związanych z takimi zagadnieniami jak aktualizacja widoku na podstawie modelu czy kontroler, który reaguje na zdarzenia i przykładowo odpowiedzialny jest za obsługę żądania, zapisu nowych danych.

Zarówno Spring jak i Laravel posiadają silniki szablonów. Jak zostało zaprezentowane w poprzednich częściach pracy, w przypadku tego pierwszego jest to Thymeleaf, a drugi związany jest z silnikiem Blade. Dzięki ich zastosowaniu prostsze staje się po stronie widoku przetworzenie danych dostarczonych w ramach modelu, usprawnione jest wiązanie obsługi żądań na podstawie akcji. Dostarczona jest również możliwość wygodnego zastosowania konstrukcji programistycznych związanych z danym językiem w kodzie określającym widok. Wszystko to usprawnia prace nad wytwarzaniem oprogramowania.

W tym miejscu należy wspomnieć, iż występuje istotna różnica pomiędzy rolą modelu w aplikacji opartej o framework Spring i Laravel. W przypadku tego pierwszego po uruchomieniu aplikacji, w bazie danych tworzone są tabele, na podstawie modelu wraz z kolumnami odpowiadającymi jego polom klasy. Jeśli chodzi o Laravela tabele tworzone są nie na bazie modelu, a na podstawie migracji. Ich uruchomienie następuje nie po odpaleniu programu, lecz po wywołaniu odpowiedniego polecenia z poziomu linii komend Artisana, tak jak zostało to opisane w jednym ze wcześniejszych rozdziałów. Konsekwencją tego jest to, że w przypadku Springa powiązania zarówno na poziomie bazy danych i kodu powstają na podstawie adnotacji w modelu. W przypadku Laravela relacje w bazie danych określamy przy pomocy migracji, a te na poziomie kodu określone są przed model.

Zarówno przy tworzeniu oprogramowania internetowego na bazie Springa jak i Laravela istnieje możliwość skorzystania z mapowania obiektowo relacyjnego. Zostało to wykorzystane przy tworzeniu projektów powstałych w ramach niniejszej pracy, co usprawniło implementowanie operacji dotyczących między innymi dodawania, odczytywania, aktualizacji oraz usuwania rekordów. W przypadku oprogramowania opartego o język Java wykorzystany został standard ORM JPA, a w przypadku języka PHP Eloquent ORM.

Kwestią wspólną dla obydwu platform programistycznych jest to, że silnie wykorzystują one wzorzec wstrzykiwania zależności. W rozdziałach poświęconych Springowi omówione zostało, że przykładowo w przypadku kontrolera istnieje możliwość pobrania Spring Beanów takich jak serwisy czy repozytoria z kontenera związanego z tym frameworkiem. Przez to zamiast tworzenia powiązania, można było je wstrzykiwać. W przypadku Laravela przedstawiony został przykład, gdzie zamiast tworzyć obiekt z żądaniem w kontrolerze, został on wstrzyknięty. Platforma programistyczna Laravel posiada również kontener związany z wstrzykiwaniem zależności, który nazywany jest kontenerem serwisów [114]. Mechanizm ten został wykorzystany przykładowo w kontrolerze użytkownika, gdzie na klasie Hash, statycznie wywołana została metoda '`make`' (Kod 24). W praktyce jednak, klasa ta nie posiada bezpośrednio implementacji wspomnianej metody, jest ona tylko fasadą, które dostarczają statycznego interfejsu do klas, znajdujących się kontenerze serwisów [115].

Również podczas tworzenia oprogramowania internetowego z pewnością istotna jest możliwość uruchamiania implementowanej aplikacji już na wczesnym etapie jej budowania. W przypadku programów działających po stronie serwerowej jest oczywiście do tego potrzebny serwer. Spring Boot

dostarcza wbudowany serwer Apache Tomcat [116], w przypadku Laravel'a istnieje możliwość odpalenia serwera deweloperskiego przy pomocy komendy '`php artisan serve`' [117]. W przypadku implementowania aplikacji opartej o język Java i platformę programistyczną Spring wykorzystywany był właśnie serwer Tomcat, w przypadku Laravela serwer deweloperski uruchamiany przy pomocy komendy Artisana.

Istotną różnicą jest sposób dodawania zależności w projekcie. W Laravelu domyślnym postępowaniem jest ich instalacja przy pomocy oddzielnego narzędzia w postaci Composera [118]. Jeśli chodzi o Springa wykonuje się to przez dedykowane temu pliki. Dokładny sposób zależny od wybranego narzędzia do budowania. Jednak zarówno, jeśli jest to Maven, Gradle [119] lub Ant [120] to zależności definiowane są w plikach do tego przeznaczonych. W implementacjach przedstawionych w poprzednich rozdziałach pokazane zostały przykłady, gdzie w celu dodawania zależności zostało użyte narzędzia Composer lub były one dodawane w postaci kodu wewnątrz pliku 'pom.xml' dla Mavena.

Ważną częścią procesu tworzenia oprogramowania są testy. Programiści często tworzą testy jednostkowe, które są między innymi wykorzystywane w podejściu TDD (z ang. *test driven development*), czyli budowania oprogramowania opartego o testy. Te ostatnie w takiej technice wytwarzania oprogramowania powstają jeszcze przed napisaniem tej funkcjonalności, której będą dotyczyć. Spring tak jak i Laravel dostarczają wsparcia dla testów jednostkowych, w przypadku tego pierwszego programista może wykorzystać JUnit o czym zostało wspomniane w rozdziale 6.4.1, Laravel natomiast wspiera popularne narzędzie do testów jednostkowych PHP w postaci PHPUnit [121].

Obie platformy programistyczne umożliwiają zbudowanie systemu autoryzacji, co zostało przedstawione w ramach rozdziałów poświęconych implementacji oprogramowania internetowego z ich wykorzystaniem. Dostarczają one narzędzi, które pozwalają stworzyć formularz logowania, mechanizmy autentykacji oparte na roli użytkownika i informacjach z bazy danych. Spring i Laravel zapewniają również funkcjonalności związane z kodowaniem zapisywanych haseł oraz pozwalające porównać te wprowadzane przez użytkownika z tymi, które już znajdują się w zakodowanej formie w bazie.

W tym miejscu należy wspomnieć, że obie platformy posiadają zabezpieczenia przed atakami CSRF i wymagają dołączania odpowiednich tokenów do żądań z formularzy, aby zostały one obsłużone. Zostało to omówione przy okazji przedstawiania zaimplementowanych wyszukiwarek produktów (Kod 9 dla Javy i Springa) oraz pracowników (Kod 20 dla PHP i Laravela).

W przypadku aplikacji internetowych, gdzie istnieje strona www, z którą użytkownik wchodzi w interakcje, kwestią mającą duże znaczenie jest to w jakim czasie odpowiedź od aplikacji będzie trafiać do użytkownika. Informacji dotyczących tego zagadnienia dostarcza artykuł pod tytułem 'Tworzenie aplikacji internetowych na platformie JEE i PHP – analiza porównawcza' opublikowany w Journal Computer Sciences Institute, gdzie pokazane zostały wyniki dla czasu odpowiedzi głównej strony jak i jednej wybranej strony.
Średnie czasy z tego pomiaru zawiera Tabela 10 [122].

Tabela 10. Spring i Laravel - Średnie czasy odpowiedzi stron. **Źródło**: [122].

	Średni czas odpowiedzi dla Javy i Springa wyrażony w milisekundach	**Średni czas odpowiedzi dla PHP i Laravela wyrażony w milisekundach**
Strona główna programu	150	714
Wybrana podstrona	95	503

Dane zawarte w Tabeli 10 obrazują, iż średni czas odpowiedzi zarówno w przypadku strony głównej i podstrony były dla Javy i Springa zdecydowanie szybsze, bo aż o 4,76 razy w przypadku tej

pierwszej i ponad 5 razy w przypadku tej drugiej, niż w przypadku PHP i Laravela. Może to mieć duże znaczenie chociażby dla czasu w jakim widok zostanie wyświetlony użytkownikowi aplikacji.

Laravel wiele funkcjonalności posiada jako wbudowaną część. Natomiast w przypadku Springa istnieje konieczność dodawania tych rzeczy osobno. Przykładem mogą być tutaj wykorzystane silniki szablonów – w przypadku Laravela Blade był już domyślnie zawarty w początkowej paczce, natomiast starter dla Thymeleaf został dodany przez zależność 'spring-boot-starter-thymeleaf', widoczną w Kodzie 2 (plik pom.xml). Z jednej strony można uznać to za korzyść, ponieważ programista ma już na starcie dostępną większą ilość narzędzi, których nie musi dodatkowo pobierać. Z drugiej strony aplikacja Springa nie jest obciążona komponentami, z których może się okazać, iż programista i tak nie skorzysta. Zajmują one w takim przypadku niepotrzebnie przestrzeń na dysku. W celu zobrazowania tej sytuacji w dniu 03.02.2022 ponownie zostały wygenerowane świeże wersje projektów, tak aby porównywany stan był aktualny i odnosił się do tej samej daty:

- Dla Laravela utworzono projekt PREMSS przy pomocy polecenia Composera: '`composer create-project --prefer-dist laravel/laravel PREMSS`'.
- Dla Springa wygenerowano paczkę przez narzędzie Spring Initializr*, a następnie po pobraniu rozpakowaną ją.
  *Różnicą w stosunku do sytuacji widocznej na Rysunku 14, jest to, że w dniu ponownego generowania projektu, aktualną, domyślną wersją Spring Boota jest 2.6.3 i dla takiej wersji został wygenerowany projekt.

Tabela 11. Spring i Laravel – Zajmowanie miejsce, ilość katalogów i plików dla świeżo wygenerowanego projektu. **Źródło**: Opracowanie własne.

	Projekt PREMSS po wygenerowaniu w wersji Spring i Java	**Projekt PREMSS po wygenerowaniu w wersji Laravel i PHP**
Rozmiar	76,7 KB	34,9 MB
Rozmiar na dysku	84,0 KB	46,6 MB
Liczba plików	10 plików	7344 plików
Liczba katalogów	15 folderów	1233 folderów

Dane dotyczące objętości katalogów dla tak stworzonych szkieletów projektów przedstawione zostały w Tabeli 11. Projekt początkowy PREMSS dla Laravela zajmuje zdecydowanie więcej miejsca na dysku: 46,6 MB w stosunku do 84 KB. Zawiera on również zdecydowanie większą liczbę plików i katalogów. Powyższe informacje, są potwierdzeniem, tego, że Laravel posiada bezsprzecznie większy narzut początkowy niż Spring.

Pomiędzy wytwarzaniem oprogramowania w oparciu o platformę Spring i Laravel występują oczywiście rozbieżności wynikające z samych różnić w języku Java oraz języku PHP, na nich skupił się jednak poprzedni rozdział.

7.3. Część III – Porównanie czasu wykonywania różnych operacji i zużycia zasobów sprzętowych

Ostatnią rzeczą, którą według autora należy wykonać w celu dopełnienia analizy porównawczej wybranych technologii w ramach niniejszej pracy dyplomowej, jest wykonanie badań związanych z czasem wykonywania różnego rodzaju funkcjonalności wraz z przetwarzaniem danych. Dodatkową, interesującą kwestią jest zużycie zasobów sprzętowych przez oprogramowanie internetowe zaimplementowane w oparciu o różne technologie. Zrealizowane testy dostarczają materiału do zbadania tych kwestii. Autor niniejszej pracy szukał takich danych w różnych źródłach, jednak obecny stan nie był zadowalający dla porównania wybranych rozwiązań w postaci języka Java wraz z Springiem oraz Laravela i języka PHP.

W celu zrobienia powyższego, do implementacji opisanej w rozdziałach 6.4.2 oraz 6.5.2 dodano funkcjonalność pozwalającą na przetestowanie i sprawdzenie czasów wykonywania takich operacji jak:

- Dodawanie nowego rekordu do bazy danych,
- Pobieranie danych z bazy dla dodanego rekordu,
- Usunięcie rekordu z bazy danych.

Są one związane z funkcjonalnością CRUD (ang. *create, read, update i delete*). W związku z tym, że operacje tworzenia i aktualizacji są zbliżone z punktu widzenia kodu źródłowego i jego przetwarzania, postanowiono z tego powodu zbadać jedynie to pierwsze, ponieważ operacja ta została uznana za bardziej kompletną – uzupełniane są wszystkie dane trafiające do bazy. Przy wspomnianych operacjach wykorzystane zostało mapowanie relacyjno-obiektowe, opisane przy omawianiu implementacji aplikacji PREMSS.

W celu dostarczenia dodatkowych danych, które przedstawiają całkiem inną operacje, zaimplementowano również metodę, odpowiedzialną za obliczanie n'tego wyrazu Ciągu Fibonacciego w sposób rekurencyjny.

Do przeprowadzenia badania wykorzystano platformę sprzętową w postaci komputera Dell Inspiron 3543, wyróżniającą się następującymi parametrami:

- Procesor Intel i5-5200U CPU (częstotliwość bazowa procesora 2.20 GHz, 2 rdzenie, 4 wątki),
- 8 GB RAM (DDR3L),
- Dysk twardy 500 GB (HDD),
- Karta graficzna Nvidia GeForce 920M oraz Intel HD Graphics 5500,
- Windows 10 Home (64 bitowy).

W testach wykorzystano wersję 11 Javy, zgodnie z plikiem 'pom.xml' przedstawionym jako Kod 2. W przypadku PHP jest to wersja 7.4.12, gdzie jak ukazane zostało w Tabeli 3 to wydanie 7 jest obecnie najbardziej popularne. Projekt dla Javy i Springa zgodnie z Rysunkiem 14, został zainicjalizowany wraz z Spring Bootem w wersji 2.4.4. Natomiast projekt PREMSS wykorzystujący Laravela został oparty o jego wersję 8.49.2.

Użyta wersja bazy danych to: „10.4.16-MariaDB for Win64 on AMD64".

Wszystkie testy dla oprogramowania PREMSS w wersji Laravel i PHP zostały wykonane na serwerze Apache HTTP Server 2.4.46.0, a dla odmiany w postaci implementacji opartej o Spring i Java zostały one wykonane na serwerze Apache Tomcat w wersji 9.0.4.4.

Schemat dla badań dotyczących stworzenia nowego rekordu, odczytu oraz jego usunięcia wyglądał następująco:

1) Doprowadzenie środowiska testowego do stanu początkowego przez takie czynności jak: wyłączenie zbędnego oprogramowania działającego w tle, uruchomienie bazy danych i jej wyczyszczenie, uruchomienie serwera czy w przypadku Laravela wyczyszczenie pamięci podręcznej przy pomocy polecenia '`php artisan optimize:clear`'.
2) Wysłanie przy pomocy biblioteki cURL zapytania HTTP mającego na celu dodanie nowego rekordu.
3) Wysłanie przy pomocy biblioteki cURL zapytania HTTP mającego na celu odczytanie rekordu stworzonego w poprzednim kroku.
4) Wysłanie przy pomocy biblioteki cURL zapytania HTTP mającego na celu wykasowanie rekordu.
5) Wyczyszczenie bazy danych.

Kroki od 2 do 5 zostały wykonane dziesięciokrotnie.

Dla Javy i Springa i przypadku tworzenia nowego rekordu, odczytu oraz kasowania rekordu wyniki badań zostały przedstawione w Tabeli 12. Natomiast dla PHP i Laravel analogiczne rezultaty zawarte są w Tabeli 14.

Dla ostatniego badania polegającego na znalezieniu 33 wyrazu Ciągu Fibonacciego rezultaty zostały przedstawione w Tabeli 13 dla Springa i Tabeli 15 dla Laravela. W tym przypadku schemat przedstawiał się następująco:

1) Doprowadzenie środowiska testowego do stanu początkowego w postaci takich czynności jak: wyłączenie zbędnego oprogramowania działającego w tle, uruchomienie bazy danych i jej wyczyszczenie, uruchomienie serwera.
2) Wysłanie przy pomocy biblioteki cURL zapytania HTTP mającego na celu wywołanie metody obliczającej n'ty wyraz Ciągu Fibonacciego.

Wszystkie wyniki przedstawione w wyżej wymienionych tabelach są przedstawione chronologicznie, co znaczy, że 'Pomiar 1' został wykonany jako pierwszy, a 'Pomiar 10' jako ostatni.

7.3.1 Badania wydajnościowe na podstawie wykonanej implementacji – Java i Spring

Dla wersji Java i Spring w celu testów dodany został kontroler 'PremssTestController' widoczny jako Kod 28.

Kod 28. Plik PremssTestController.java dla projektu PREMSS – Java i Spring. **Źródło**: Opracowanie własne.

```
//(…)
@Controller
public class PremssTestController {

    @Autowired
    ProductRepository productRepository;

    @RequestMapping(path = "/PremssTestApi/testCreateMethodPremssJava")
    @ResponseBody
    public String testCreateProductMethodPremssJava(){
      long timeOfExecutionStart = System.currentTimeMillis();
      //---

      ProductModel newProductRecord = new ProductModel();
      newProductRecord.setId(1L);
      newProductRecord.setName("Zasilacz Corsair CX450M 450W");
      newProductRecord.setAvailability(true);
      newProductRecord.setPrice(285.00);
      productRepository.save(newProductRecord);

      //---
      long timeOfExecutionEnd = System.currentTimeMillis();
      long durationOfExecution = timeOfExecutionEnd - timeOfExecutionStart;
      return "\nPremssJavaTestControllerResponse
      - testCreateProductMethodPremssJava czas wykonywania: " +
      durationOfExecution;
    }
```

```java
@RequestMapping(path = "/PremssTestApi/testReadMethodPremssJava")
@ResponseBody
public String testGetProductMethodPremssJava(){
  long timeOfExecutionStart = System.currentTimeMillis();
  //---

  Optional<ProductModel> product = productRepository.findById(1L);

  //---
  long timeOfExecutionEnd = System.currentTimeMillis();
  long durationOfExecution = timeOfExecutionEnd - timeOfExecutionStart;
  //System.out.println(product.get().getName());
  return "\nPremssJavaTestControllerResponse
  - testGetProductMethodPremssJava czas wykonywania: " +
  durationOfExecution;
}

@RequestMapping(path = "/PremssTestApi/testDeleteMethodJava")
@ResponseBody
public String testDeleteProductMethodPremssJava(){
  long timeOfExecutionStart = System.currentTimeMillis();
  //---

  productRepository.deleteById(1L);

  //---
  long timeOfExecutionEnd = System.currentTimeMillis();
  long durationOfExecution = timeOfExecutionEnd - timeOfExecutionStart;
  return "\nPremssJavaTestControllerResponse
  - testDeleteProductMethodPremssJava czas wykonywania: "
  + durationOfExecution;
}

@RequestMapping(path = "/PremssTestApi/
testFindNthFibonnaciNumberPremssJava")
@ResponseBody
public String testFindNthFibonnaciNumberPremssJava(){
  long timeOfExecutionStart = System.currentTimeMillis();
  //---

  long x = findNthFibonnaciNumberRecursivelyPremssJava(33);
  //---
  long timeOfExecutionEnd = System.currentTimeMillis();
  long durationOfExecution = timeOfExecutionEnd - timeOfExecutionStart;
  return "\nPremssJavaTestControllerResponse
  - testFindNthFibonnaciNumberPremssJava czas wykonywania: " +
  durationOfExecution +
  " ||| Obliczony 33 wyraz sekwencji Fibonacciego to: " + x;
```

```
    }

    public long findNthFibonnaciNumberRecursivelyPremssJava(int n){
        if (n == 0) {
            return 0;
        }else if( n == 1){
            return 1;
        }else {
           return (findNthFibonnaciNumberRecursivelyPremssJava(n -1) +
                   findNthFibonnaciNumberRecursivelyPremssJava(n - 2));
        }

    }
}
```

Tabele 12 oraz 13 pokazują wyniki badań opisanych w rozdziale 7.3.

Tabela 12. Pomiary wykonane dla operacji dodawania, odczytu i usuwania rekordów z bazy danych – PREMSS w wersji Java i Spring. **Źródło**: Opracowanie własne.

	Pomiary wykonane dla operacji dodawania nowego rekordu do bazy danych [ms]	**Pomiary wykonane dla operacji odczytu rekordu z bazy danych [ms]**	**Pomiary wykonane dla operacji usuwania rekordu z bazy danych [ms]**
Pomiar 1	232	96	141
Pomiar 2	110	3	211
Pomiar 3	98	4	103
Pomiar 4	83	4	178
Pomiar 5	99	3	80
Pomiar 6	163	5	101
Pomiar 7	154	4	97
Pomiar 8	79	3	112
Pomiar 9	160	4	131
Pomiar 10	58	4	112

Tabela 13. Pomiary wykonane dla operacji obliczenia 33 wyrazu ciągu Fibonacciego – PREMSS w wersji Java i Spring. **Źródło**: Opracowanie własne.

	Wynik [ms]
Pomiar 1	26
Pomiar 2	24
Pomiar 3	23
Pomiar 4	24
Pomiar 5	23
Pomiar 6	25
Pomiar 7	24
Pomiar 8	23
Pomiar 9	26
Pomiar 10	26

7.3.2 Badania wydajnościowe na podstawie wykonanej implementacji – PHP i Laravel

W ramach PHP i Laravela dla badań stworzono kontroler ‘PremssTestController’, którego kod źródłowy zobrazowany jest jako Kod 29.

Kod 29. Plik PremssTestController.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```php
<?php
//(…)
class PremssTestController extends Controller
{
    public function testCreateProductMethodPremssPHP()
    {
        $timeOfExecutionStart = microtime(true);
        //---

        $newProductRecord = new Product();
        $newProductRecord->id = 1;
        $newProductRecord->name = "Zasilacz Corsair CX450M 450W";
        $newProductRecord->availability = true;
        $newProductRecord->price = 285.00;
        $newProductRecord->save();

        //---
        $timeOfExecutionEnd = microtime(true);
        $durationOfExecutionInMilliSeconds =
        round(($timeOfExecutionEnd - $timeOfExecutionStart) * 1000);
        return "\nPremssPHPTestControllerResponse
        - testCreateProductMethodPremssPHP czas wykonywania: " .
        $durationOfExecutionInMilliSeconds;
    }


    public function testGetProductMethodPremssPHP()
    {
        $timeOfExecutionStart = microtime(true);
        //---

        $product = Product::find(1);

        //---
        $timeOfExecutionEnd = microtime(true);
        $durationOfExecutionInMilliSeconds =
        round(($timeOfExecutionEnd - $timeOfExecutionStart) * 1000);
        //echo $product->name;
        return "\nPremssPHPTestControllerResponse
        - testGetProductMethodPremssPHP czas wykonywania: " .
        $durationOfExecutionInMilliSeconds;
    }
```

```
    public function testDeleteProductMethodPremssPHP()
    {
        $timeOfExecutionStart = microtime(true);
        //---

        Product::destroy(1);

        //---
        $timeOfExecutionEnd = microtime(true);
        $durationOfExecutionInMilliSeconds =
        round(($timeOfExecutionEnd - $timeOfExecutionStart) * 1000);
        return "\nPremssPHPTestControllerResponse
        - testDeleteProductMethodPremssPHP czas wykonywania: " .
        $durationOfExecutionInMilliSeconds;
    }

    public function testFindNthFibonnaciNumberPremssPHP()
    {
        $timeOfExecutionStart = microtime(true);
        //---

        $x = $this->findNthFibonnaciNumberRecursivelyPremssPHP(33);

        //---
        $timeOfExecutionEnd = microtime(true);
        $durationOfExecutionInMilliSeconds =
        round(($timeOfExecutionEnd - $timeOfExecutionStart) * 1000);
        return "\nPremssPHPTestControllerResponse:
        testFindNthFibonnaciNumberPremssPHP czas wykonywania: " .
        $durationOfExecutionInMilliSeconds .
        " ||| Obliczony 33 wyraz sekwencji Fibonacciego to: "  . $x;
    }

    public function findNthFibonnaciNumberRecursivelyPremssPHP($n){
        if ($n == 0){
            return 0;
        }else if ($n == 1){
            return 1;
        }else{
        return ($this->findNthFibonnaciNumberRecursivelyPremssPHP($n-1)
              + $this->findNthFibonnaciNumberRecursivelyPremssPHP($n-2));
        }
    }

}
```

Dla obsługi ścieżek kontrolera przez API, zostały dodane linie kodu do pliku ‘api.php’ widoczne jako Kod 30.

Kod 30. Plik api.php dla projektu PREMSS – PHP i Laravel. **Źródło**: Opracowanie własne.

```
<?php
//(…)

//Routes for Premss Test API:
Route::get('/PremssTestApi/testCreateMethodPremssPHP',
[PremssTestController::class, 'testCreateProductMethodPremssPHP']);
Route::get('/PremssTestApi/testGetMethodPremssPHP',
[PremssTestController::class, 'testGetProductMethodPremssPHP']);
Route::get('/PremssTestApi/testDeleteMethodPremssPHP',
[PremssTestController::class, 'testDeleteProductMethodPremssPHP']);
Route::get('/PremssTestApi/testFindNthFibonnaciNumberPremssPHP',
[PremssTestController::class, 'testFindNthFibonnaciNumberPremssPHP']);
```

W przypadku niniejszej implementacji, autor zdecydował się na wprowadzenie dodatkowej konfiguracji dla PHP w pliku ‘php.ini’, w celu włączenia mechanizmu OPcache, omówionego w rozdziale poświęconemu przetwarzaniu kodu źródłowego napisanego w języku PHP. Kod związany z tym ustawieniem został przedstawiony jako Kod 31.

Kod 31. Ustawienia w pliku php.ini dla OPcache – PHP i Laravel. **Źródło**: Opracowanie własne.

```
; (…)
zend_extension=C:\xampp\php\ext\php_opcache.dll
opcache.enable=1
opcache.enable_cli=1
opcache.memory_consumption=256
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=4000
; (…)
```

Tabele 14 oraz 15 przedstawiają wyniki badań opisanych w rozdziale 7.3.

Tabela 14. Pomiary wykonane dla operacji dodawania, odczytu i usuwania rekordów z bazy danych – PREMSS w wersji PHP i Laravel. **Źródło**: Opracowanie własne.

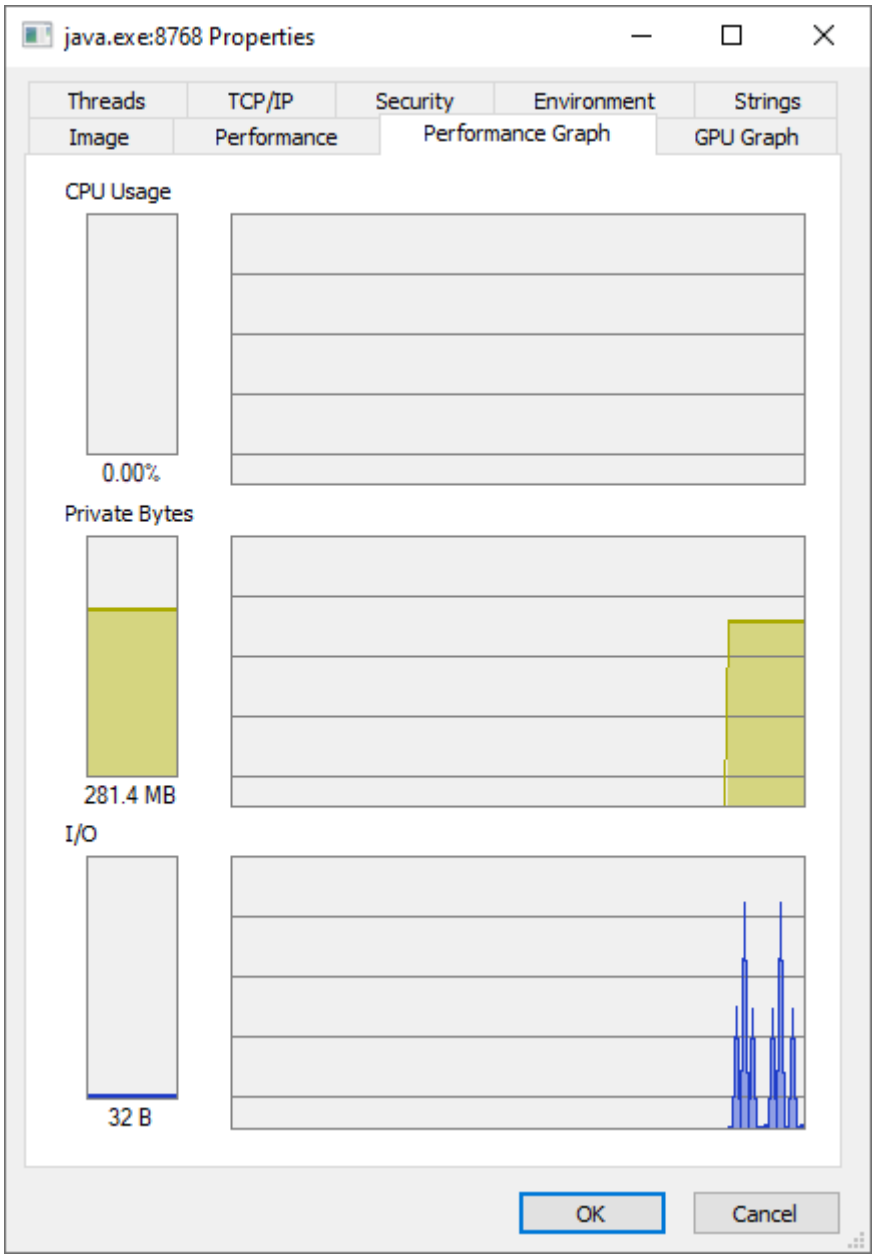
	Pomiary wykonane dla operacji dodawania nowego rekordu do bazy danych [ms]	**Pomiary wykonane dla operacji odczytu rekordu z bazy danych [ms]**	**Pomiary wykonane dla operacji usuwania rekordu z bazy danych [ms]**
Pomiar 1	201	75	113
Pomiar 2	105	10	70
Pomiar 3	164	10	117
Pomiar 4	163	10	93
Pomiar 5	163	10	78
Pomiar 6	57	10	147
Pomiar 7	168	10	126
Pomiar 8	116	12	135
Pomiar 9	169	11	130
Pomiar 10	211	13	117

Tabela 15. Pomiary wykonane dla operacji obliczenia 33 wyrazu ciągu Fibonacciego – PREMSS w wersji PHP i Laravel. **Źródło**: Opracowanie własne.

	Wynik [ms]
Pomiar 1	18358
Pomiar 2	17583
Pomiar 3	18608
Pomiar 4	17735
Pomiar 5	17964
Pomiar 6	18934
Pomiar 7	19002
Pomiar 8	18964
Pomiar 9	18329
Pomiar 10	17399

7.3.3 Badania zużycia zasobów sprzętowych oraz kolejne badania wydajnościowe

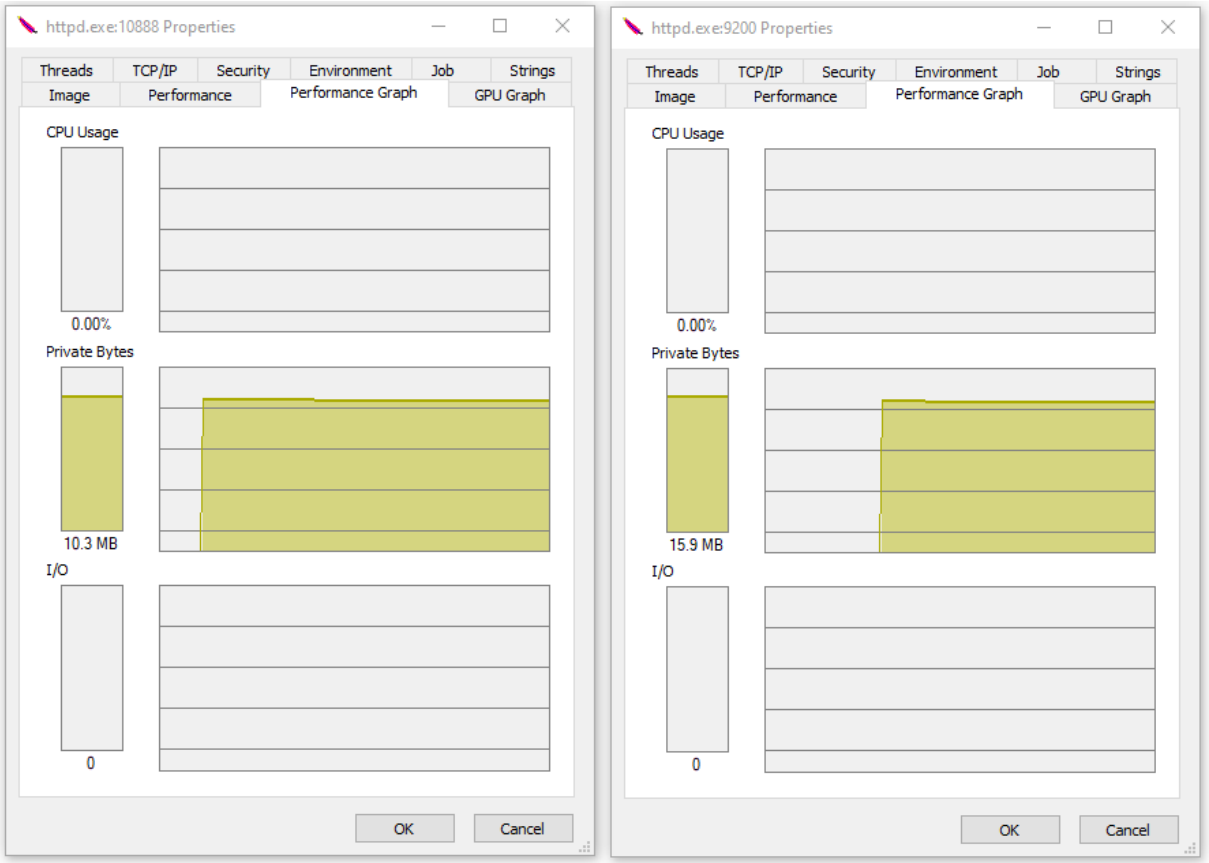
Kwestią, którą również postanowiono porównać jest stopień użycia zasobów sprzętowych podczas działania oprogramowania internetowego stworzonego w oparciu o technologie wybrane w ramach niniejszej pracy. Do zbadania zużycia pamięci wykorzystano narzędzie Process Explorer oraz Monitor Wydajności i wskaźnik ‘Private Bytes’. Jak podaje oficjalna strona Microsftu – Bajty Prywatne są to takie, które wyrażają całkowitą pamięć, która została alokowana przez dany proces, bez uwzględniania tych, które są współdzielone z innymi procesami [123]. W przypadku Javy po uruchomieniu serwera Apache Tomcat na którym znajduje się tylko aplikacja PREMSS uruchomiony został proces ‘java.exe’.



Rysunek 32. Zużycie zasobów przez Serwer Apache Tomcat z aplikacją PREMSS, Java i Spring.
Źródło: Opracowanie własne.

Rysunek 32 będący wycinkiem z narzędzia Process Explorer prezentuje, że wykorzystane zostało 281,4 MB. przez Server Apache Tomcat z aplikacją PREMSS (Java i Spring).

Natomiast po starcie serwera Apache HTTP Server, na którym znajduje się jedynie aplikacja PREMSS w wersji PHP, uruchomione zostały dwa procesy ‘httpd.exe’. Dla tych dwóch procesów przydzielono kolejno 10,3 MB oraz 15,9 MB co widoczne jest na Rysunku 33 i daje to w sumie 26,2 MB.



Rysunek 33. Zużycie zasobów przez Serwer Apache HTTP Server z aplikacją PREMSS, PHP i Laravel. **Źródło**: Opracowanie własne.

Aby zbadać zmianę zużycia pamięci w czasie przez dwie wersje aplikacji PREMSS postanowiono skorzystać z narzędzia Monitor wydajności oraz Apache Bench, które pozwala na wygenerowanie większej ilości żądań HTTP na wskazany adres. Przeprowadzono 5 testów dla każdej wersji aplikacji PREMS, gdzie każdy z nich składał się z 100 żądań HTTP, a odpytywanie następowało równolegle przez 10 klientów – wybranym adresem były strony do logowania. Wyniki dla Javy i Springa zaprezentowano w Tabeli 16, natomiast PHP i Laravela w Tabeli 17.

Tabela 16. Testy użycia pamięci – PREMSS w wersji Java i Spring. **Źródło**: Opracowanie własne.

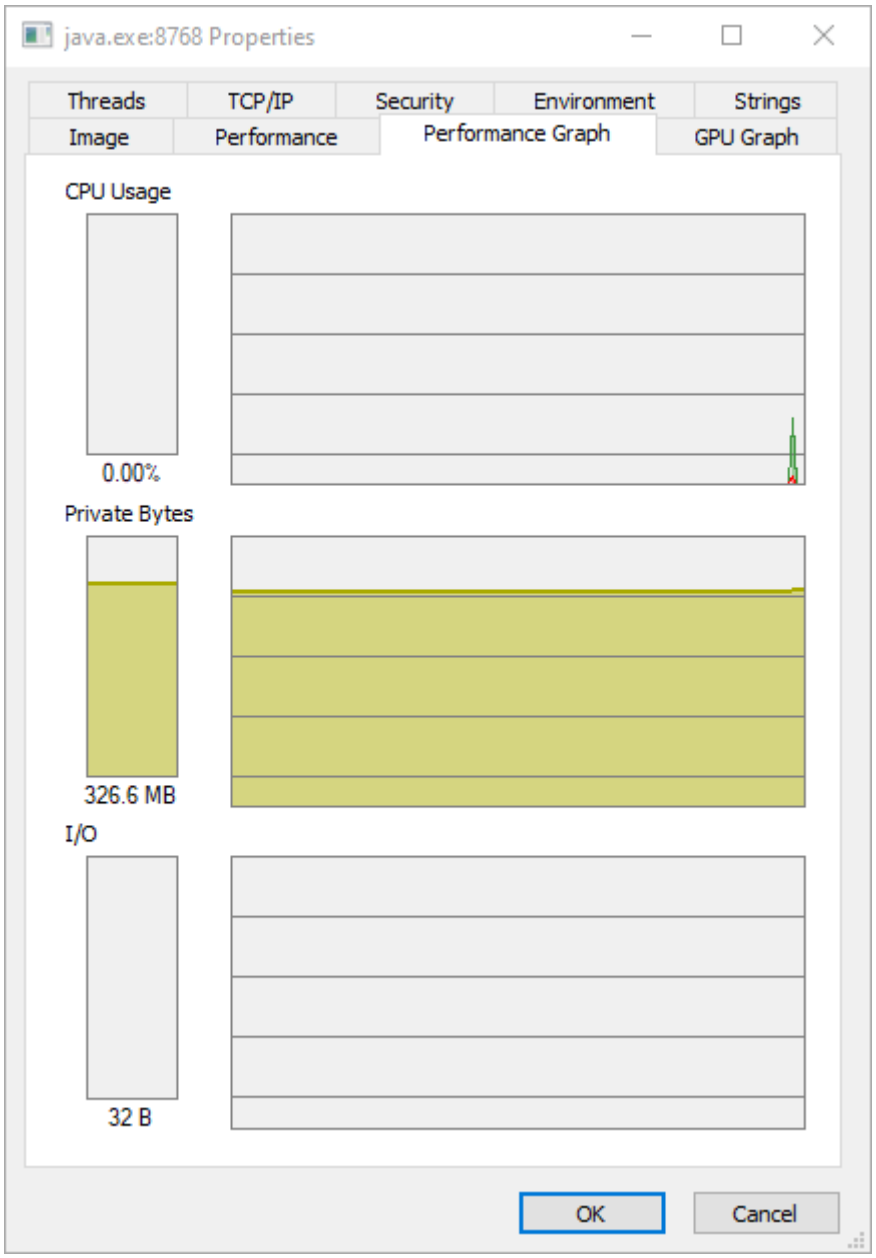
	Bajty Prywatne (Przed Testem) [MB]	**Bajty Prywatne (Po Teście) [MB]**	**Różnica [MB]**
Pomiar 1	281,76	289,01	7,25
Pomiar 2	309,07	310,48	1,41
Pomiar 3	310,17	314,89	4,72
Pomiar 4	314,85	315,04	0,19
Pomiar 5	323,66	326,62	2,96
Średnia	307,90	311,20	3,30

Tabela 17. Testy użycia pamięci – PREMSS w wersji PHP i Laravel. **Źródło**: Opracowanie własne.

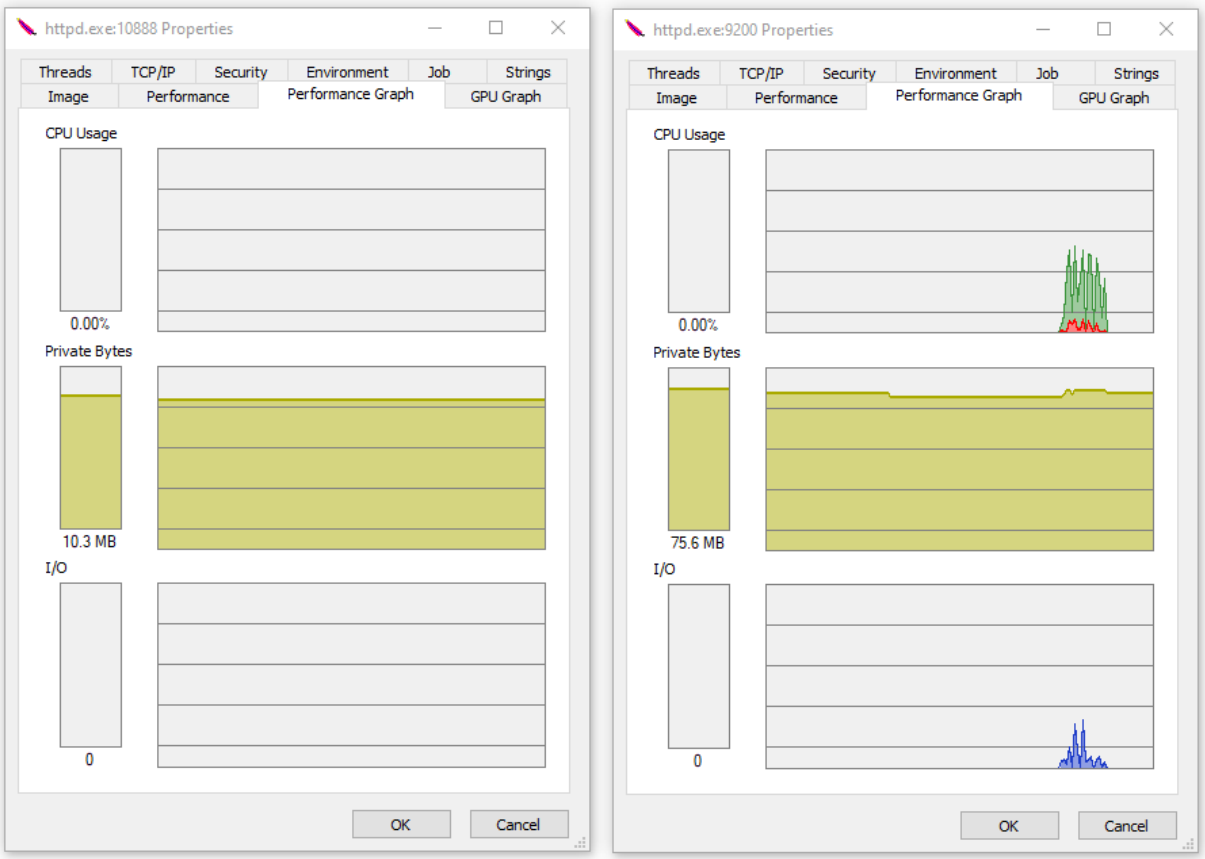
	Bajty Prywatne (Przed Testem) [MB]	**Bajty Prywatne (Po Teście) [MB]**	**Różnica [MB]**
Pomiar 1	15,87	75,18	59,31
Pomiar 2	71,82	75,93	4,11
Pomiar 3	74,44	75,93	1,49
Pomiar 4	74,66	76,53	1,87
Pomiar 5	73,46	76,60	3,14
Średnia	62,05	76,03	13,98

Autor niniejszej pracy chciałby zwrócić uwagę w odniesieniu do pomiaru 2 oraz pomiaru 3 zawartych w Tabeli 17 i rezultatu kolumnie „Bajty Prywatne (Po Teście)” na to, iż taka sama liczba bajtów nie wynika z błędnego zapisu, a faktycznie otrzymanych wartości.

Po przeprowadzaniu powyższych testów sytuacja w programie Process Explorer wyglądała w przypadku Javy i Springa tak jak na Rysunku 34 oraz jak na Rysunku 35 dla PHP i Laravela.



Rysunek 34. Zużycie zasobów przez Serwer Apache Tomcat z aplikacją PREMSS, Java i Spring – Po testach. **Źródło**: Opracowanie własne.



Rysunek 35. Zużycie zasobów przez Serwer Apache HTTP Server z aplikacją PREMSS, PHP i Laravel – Po testach. **Źródło**: Opracowanie własne.

Wspomniane narzędzie Apache Bench przy okazji wykonywania wcześniejszych testów, dostarczyło również dodatkowych, cennych informacji. W Tabeli 18 znajdują się informacje na temat żądań zakończonych sukcesem i porażką oraz transferu wyrażonego w bajtach, które przypadały na każde 100 żądań. W Tabeli 19 oraz 20 przedstawiono kolejno dla Javy i Springa orz PHP i Laravela ilość żądań na sekundę, średni czas jednego żądania oraz prędkość transferu. Rysunek 36 natomiast prezentuje przykładowe wykonanie testu wraz z jego wynikiem.

Tabela 18. Testy Apache Bench - Rezultat testów oraz transfer. **Źródło**: Opracowanie własne.

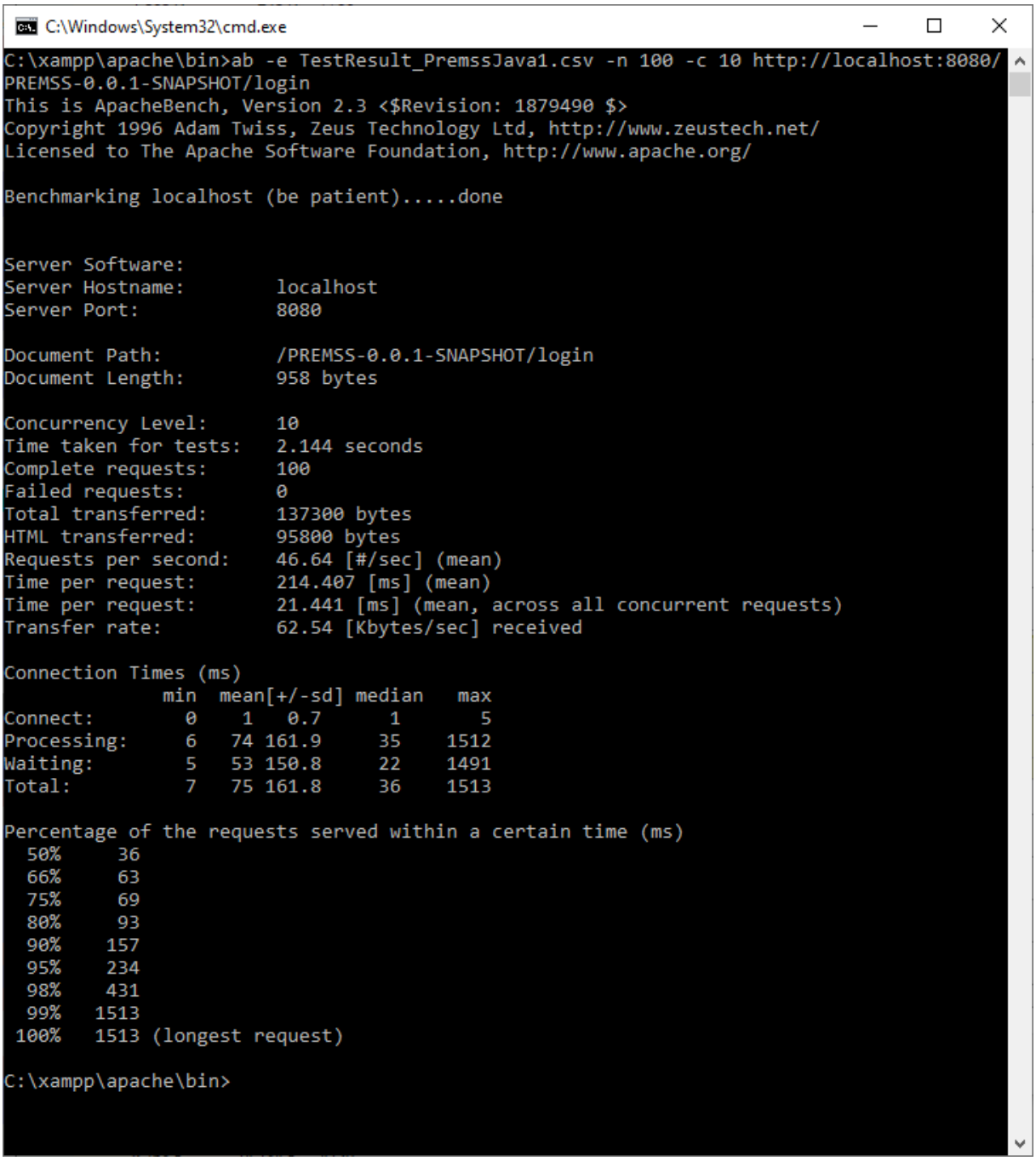
	Java i Spring	**PHP i Laravel**
Ilość żądań zakończonych sukcesem	1000	1000
Ilość żądań zakończonych porażką	0	0
Całkowity transfer na 100 żądań:	137300 Bajtów	634900 Bajtów

Tabela 19. Testy Apache Bench – Dane dot. żądań (Java i Spring). **Źródło**: Opracowanie własne.

	Ilość żądań na sekundę	**Średni czas jednego żądania [ms]**	**Prędkość transferu [KB/s]**
Test 1	46,64	214,407	62,54
Test 2	208,87	47,877	280,06
Test 3	226,94	44,065	304,29
Test 4	384,83	25,985	515,99
Test 5	342,05	29,236	458,62
Średnia:	241,87	72,314	324,30

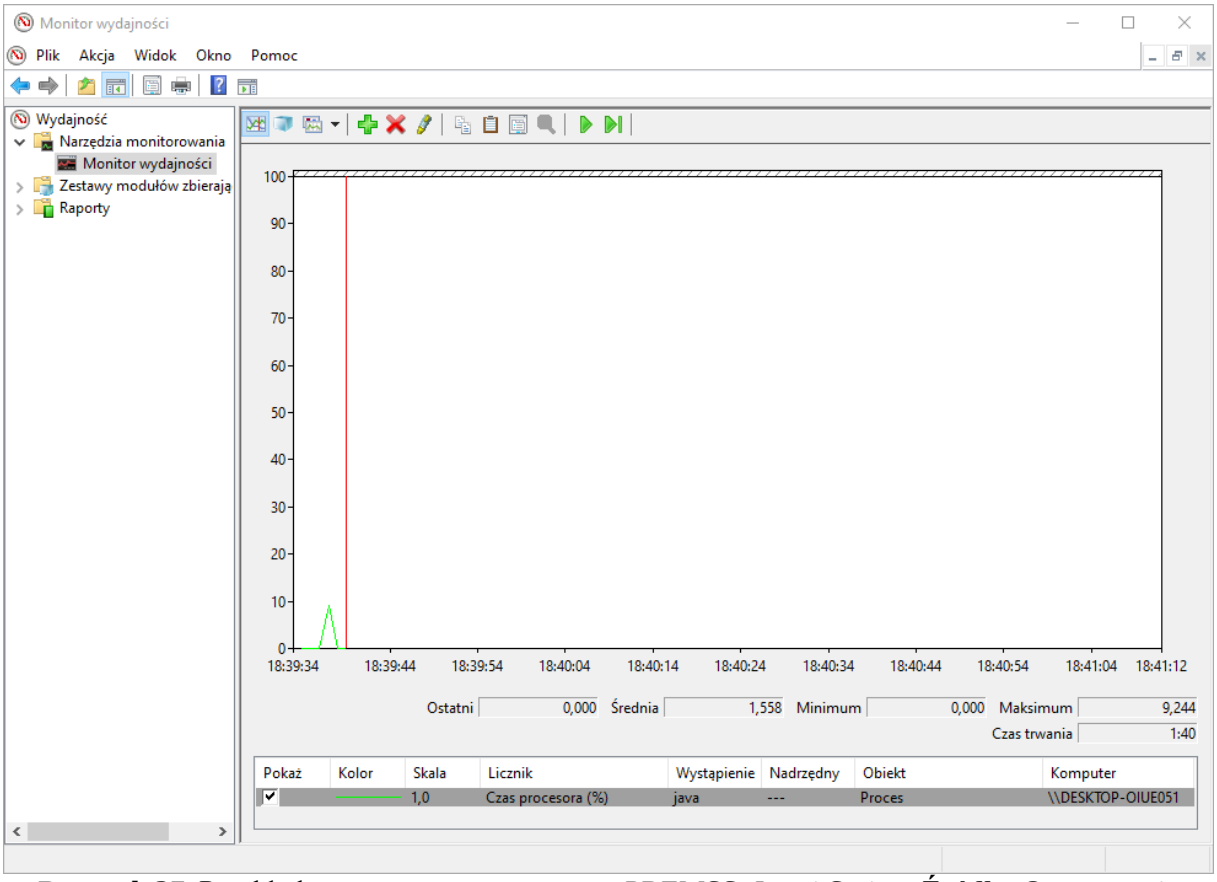
Tabela 20. Testy Apache Bench – Dane dot. żądań (PHP i Laravel). **Źródło**: Opracowanie własne.

	Ilość żądań na sekundę	**Średni czas jednego żądania [ms]**	**Prędkość transferu [KB/s]**
Test 1	3,48	2875,639	21,56
Test 2	5,48	1824,544	33,98
Test 3	5,52	1811,247	34,23
Test 4	5,48	1824,289	33,93
Test 5	5,50	1818,979	34,09
Średnia:	5,09	2030,940	31,56



Rysunek 36. Przykładowy test z wykorzystaniem narzędzia Apache Bench - Server Apache Tomcat z aplikacją PREMSS, Java i Spring. **Źródło**: Opracowanie własne.

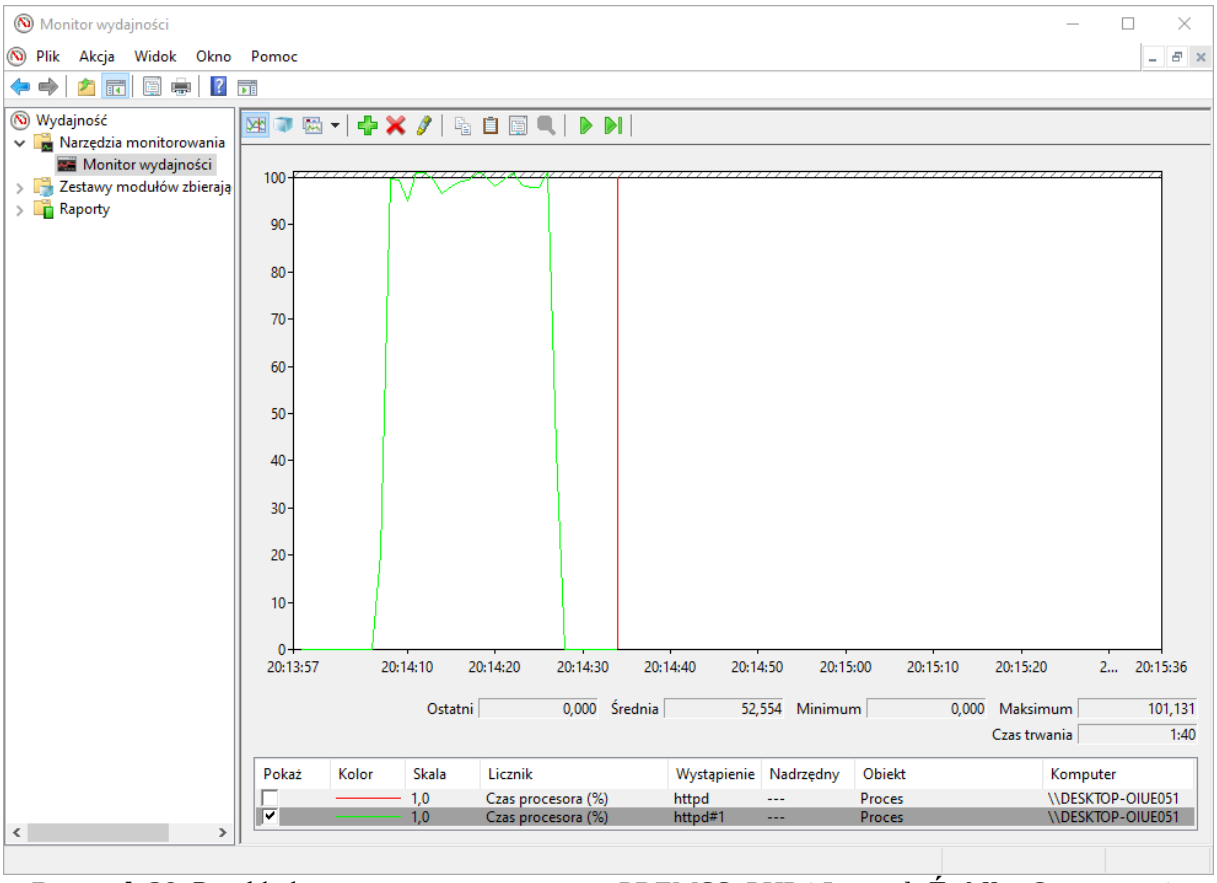
Autor pracy zdecydował również o porównaniu wykorzystania procesora, w tym celu wykorzystano Monitor Wydajności, który wbudowany jest w system Windows 10 oraz wskaźnik czasu procesora. Testy przeprowadzono na podstawie wcześniej zaprezentowanych metod służących do obliczenia trzydziestego trzeciego wyrazu Ciągu Fibonacciego poprzez obserwacje wspomnianego wskaźnika po wywołaniu tych metod. Dla Javy i Springa monitorowany był jeden proces 'java.exe', w przypadku PHP i Laravela monitorowane były dwa procesy 'httpd.exe' (przy czym tylko jeden z nich podczas testów wykorzystywał procesor i to wyniki dla niego zostały zebrane wyniki). Rysunek 37 i 38 przedstawiają przykładowe testy, które zostały przeprowadzone dla wybranych technologii. Tabele 21 i 22 prezentują natomiast po dziesięć wyników pomiarów, które zostały wykonane dla każdej wersji aplikacji PREMSS. Wyniki przedstawione są w kolejności odpowiadającej sekwencji wykonywania testów.



Rysunek 37. Przykładowy test czasu procesora – PREMSS, Java i Spring. **Źródło**: Opracowanie własne.

Tabela 21. Testy czasu procesora – PREMSS w wersji Java i Spring. **Źródło**: Opracowanie własne.

	Maksymalny Czas procesora [%]	**Czas trwania [s]**
Pomiar 1	61,101	3
Pomiar 2	9,244	2
Pomiar 3	6,330	2
Pomiar 4	6,304	2
Pomiar 5	6,336	3
Pomiar 6	6,308	2
Pomiar 7	3,153	2
Pomiar 8	3,133	2
Pomiar 9	3,140	2
Pomiar 10	7,712	3
Średnia	11,276	2,3



Rysunek 38. Przykładowy test czasu procesora – PREMSS, PHP i Laravel. **Źródło**: Opracowanie własne.

Tabela 22. Testy czasu procesora – PREMSS w wersji PHP i Laravel. **Źródło**: Opracowanie własne.

	Maksymalny Czas procesora [%]	**Czas trwania [s]**
Pomiar 1	101,212	24
Pomiar 2	101,131	22
Pomiar 3	100,714	19
Pomiar 4	101,211	20
Pomiar 5	101,260	20
Pomiar 6	101,230	20
Pomiar 7	101,120	21
Pomiar 8	101,116	20
Pomiar 9	101,017	21
Pomiar 10	101,216	20
Średnia	101,123	20,7

7.3.4 Podsumowanie badań wydajnościowych i zużycia zasobów sprzętowych

W Tabeli 23 zagregowane zostały dane ze wszystkich testów wydajnościowych, którym były poświęcone rozdziały 7.3.1 oraz 7.3.2.

Tabela 23. Zgrupowane dane poświęcone testom wydajnościowym PREMSS z rozdziałów 7.3.1 oraz 7.3.2. **Źródło**: Opracowanie własne.

	Java i Spring (czas wyrażony w milisekundach)	**PHP i Laravel (czas wyrażony w milisekundach)**
Średni czas dla operacji tworzenia nowego rekordu w bazie danych	123,6	151,7
Maksymalny czas dla operacji tworzenia nowego rekordu w bazie danych	232	211
Minimalny czas dla operacji tworzenia nowego rekordu w bazie danych	58	57
Średni czas dla operacji odczytywana informacji z bazy danych dotyczących wybranego rekordu	13	17,1
Maksymalny czas dla operacji odczytywana informacji z bazy danych dotyczących wybranego rekordu	96	75
Minimalny czas dla operacji odczytywana informacji z bazy danych dotyczących wybranego rekordu	3	10
Średni czas dla operacji usuwania wybranego rekordu z bazy danych	126,6	112,6
Maksymalny czas dla operacji usuwania wybranego rekordu z bazy danych	211	147
Minimalny czas dla operacji usuwania wybranego rekordu z bazy danych	80	70
Średni czas dla operacji obliczenia w sposób rekurencyjny 33 wyrazu Ciągu Fibonacciego	24,4	18287,6
Maksymalny czas dla operacji obliczenia w sposób rekurencyjny 33 wyrazu Ciągu Fibonacciego	26	19002
Minimalny czas dla operacji obliczenia w sposób rekurencyjny 33 wyrazu Ciągu Fibonacciego	23	17399

W przypadku aplikacji PREMSS zaimplementowanej w oparciu o język Java oraz Springa średni czas wykonywania operacji dodawania nowego rekordu do bazy po stronie kodu wyniósł 123,6 milisekund, a w przypadku PHP i Laravela 151,7 milisekund. Pierwszy zestaw technologii okazał się zatem szybszy dla tego typu operacji. Jeśli chodzi o różnicę w wartościach w przypadku Springa to pierwszy pomiar okazał się zarazem tym, w którym czas trwania operacji był najdłuższy, a ostatni najkrótszy. Nie należy jednak wyciągać z tego takiego wniosku, iż w późniejszych pomiarach czas ulega optymalizacji, ponieważ przykładowo przedostatni pomiar jest trzecim najdłuższym wynikiem w próbie. Dla Laravela natomiast maksymalny czas nastąpił przy pomiarze dziesiątym (ostatnim), a najkrótszy w pomiarze szóstym.

W sytuacji odczytywania pojedynczego rekordu z bazy danych przeciętny czas dla Laravela wyniósł 17,1 milisekund, a dla Springa 13 milisekund. W tym badaniu również przeciętny czas był niższy w przypadku Springa. Wyniki testów okazały się tym ciekawsze, iż ujawniły one optymalizacje zarówno w jednym i drugim przypadku. Otóż w przypadku tego pierwszego pierwszy pomiar wyniósł 75 milisekund, a w wypadku dziewięciu późniejszych testów wartość nie przekroczyła 13 milisekund. Podobnie sytuacja ukształtowała się również dla Springa, to przy pierwszym pomiarze odnotowano maksimum w postaci 96 milisekund, aby przy następnych zmierzonych wynikach wartość nie przekroczyła 5 milisekund. Jest to duży wzrost wydajności w jednej i drugiej technologii w stosunku do pierwszych pomiarów.

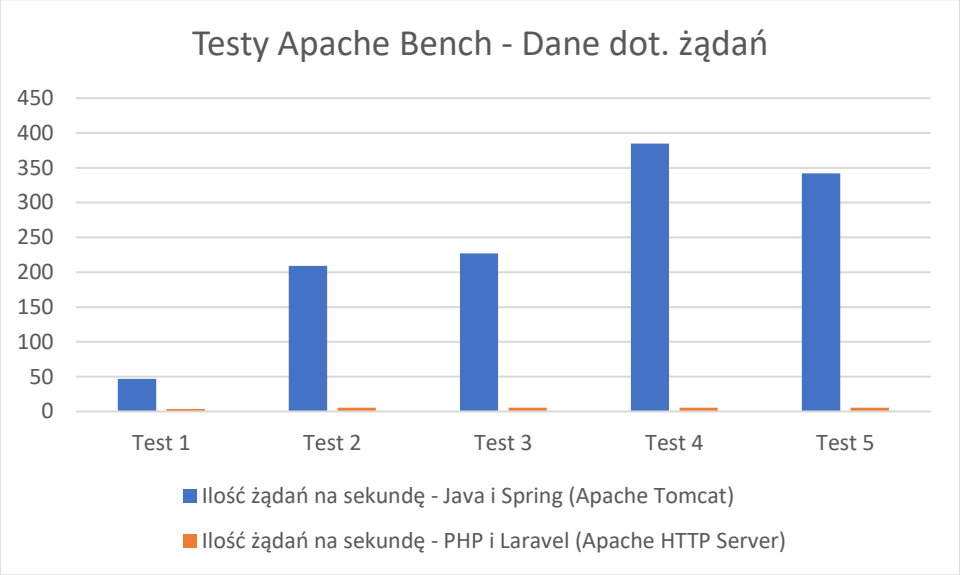
Dla wykonywania obsługi żądania usunięcia zasobu z bazy danych średni czas dla implementacji opartej o platformę programistyczną PHP wyniósł 112,6 milisekund. W przypadku tej bazującej na Javie przeciętny czas ukształtował się na poziomie 126,6 milisekundy. Jest to jedyne przeprowadzone badanie, w którym biorąc pod uwagę średnią arytmetyczną to oprogramowanie oparte na Laravelu wykazało lepszy rezultat. Autor niniejszej pracy uważa, że uwzględniając rozkład wyników w poszczególnych pomiarach, można uznać, iż czas nie ulegał poprawie w wyniku działania mechanizmów optymalizacji wraz z wykonywaniem kolejnych testów, zarówno w jednym i drugim przypadku. Przykładowo dla Springa najkrótszy rezultat w postaci 80 milisekund przypadł na pomiar piąty, a dla Laravela najniższa wartość 70 milisekund na pomiar drugi.

Ostatni test oparty na obliczeniu n'tego wyrazu Ciągu Fibonacciego wykazał bardzo duże różnice w czasie wykonywania operacji. Został on oparty na rekurencji. W przypadku Springa średni czas wyniósł 24,4 milisekundy, gdzie najgorszym odnotowanym rezultatem było 26 milisekund. Natomiast w przypadku Laravela średni czas wyniósł 18287,6 milisekund, a maksymalny odnotowany czas to 19002 milisekund. Najgorszy czas dla Laravela jest aż ponad 730 razy dłuższy niż najsłabszy czas dla Springa. Dla tego pierwszego przy żadnej próbie, nie został osiągnięty rezultat krótszy niż 17,399 sekund.

Opisane badania wydajnościowe pokazały różnice w czasach wykonywania operacji CRUD dla wybranych technologii w postaci języka Java wraz z platformą programistyczną Spring oraz języka PHP wraz z frameworkiem Laravel. W przypadku dodawania i odczytywania średnie pomiary okazały się korzystniejsze dla pierwszego zestawu. Natomiast dla drugiego, przeciętne czasy okazały się lepsze przy operacji usuwania rekordów. Uwzględniając wielkość różnic autor niniejszej pracy uważa, że zarówno jeden i drugi zbiór technologii może być z sukcesem wykorzystywany do implementowania małych i średnich aplikacji. W przypadku operacji wymagających bardziej złożonych obliczeń, co pokazało badanie dotyczące wyliczania trzydziestego trzeciego wyrazu Ciągu Fibonacciego metodą wykorzystującą rekurencję to implementacja oparta o Javę okazała się zdecydowanie bardziej wydajna. W związku z powyższym oraz możliwością wystąpienia innego rodzaju złożonych operacji obliczeniowych w dużych, skomplikowanych aplikacjach biznesowych, w tego typu oprogramowaniu autor wybrałby technologię bazującą na języku Java.

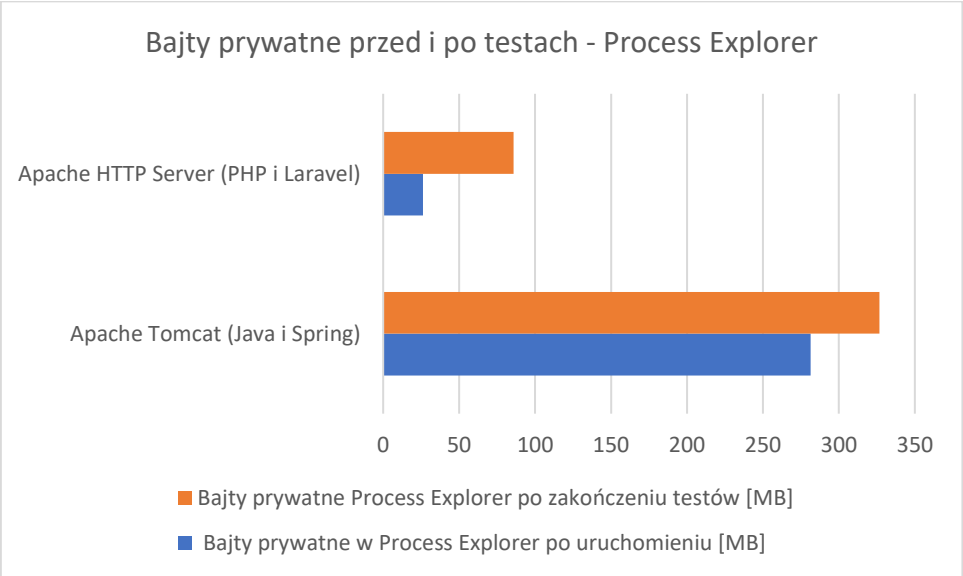
W rozdziale 7.3.3 zaprezentowane zostały dodatkowe badania wykonane przy pomocy narzędzia Apache Bench, które dostarczają informacji na temat sprawności w przetwarzaniu żądań (Tabela 19 oraz 20). Ukazują one, iż aplikacja napisana w Javie i Springu na serwerze Apache Tomcat była w stanie przetworzyć średnio 241,87 żądań na sekundę, podczas gdy aplikacja oparta o PHP i Laravela

znajdująca się na serwerze Apache HTTP Server tylko 5,09 żądań na sekundę. Różnica jest tutaj bardzo duża, ponieważ średnia obsłużonych żądań w ciągu sekundy jest ponad 47 razy większa w przypadku pierwszego zestawu. Wyniki wykazały, że w przypadku implementacji zarówno na jednej jak i drugiej technologii to w przypadku pierwszych testów obsłużonych zostało najmniej żądań na sekundę. Dla Javy było to 46,64 a dla PHP 3,48 – przy kolejnych testach nastąpił wzrost. Jak widać w Tabelach 19 oraz 20 wzrost ten był w korelacji ze wzrostem prędkości transferu. Przy założeniu utrzymania takiego związku i proporcjonalności to nawet jeśli transfer w przypadku PHP i Laravela osiągnąłby prędkość transferu Javy i Springa to wciąż ilość żądań na sekundę w przypadku tego pierwszego była by niższa, co wskazuje, że różnica nie jest wynikiem tylko tego parametru. Dla zobrazowania wyniku i różnicy w testach, rezultat został zaprezentowany na wykresie widocznym jako Rysunek 39.



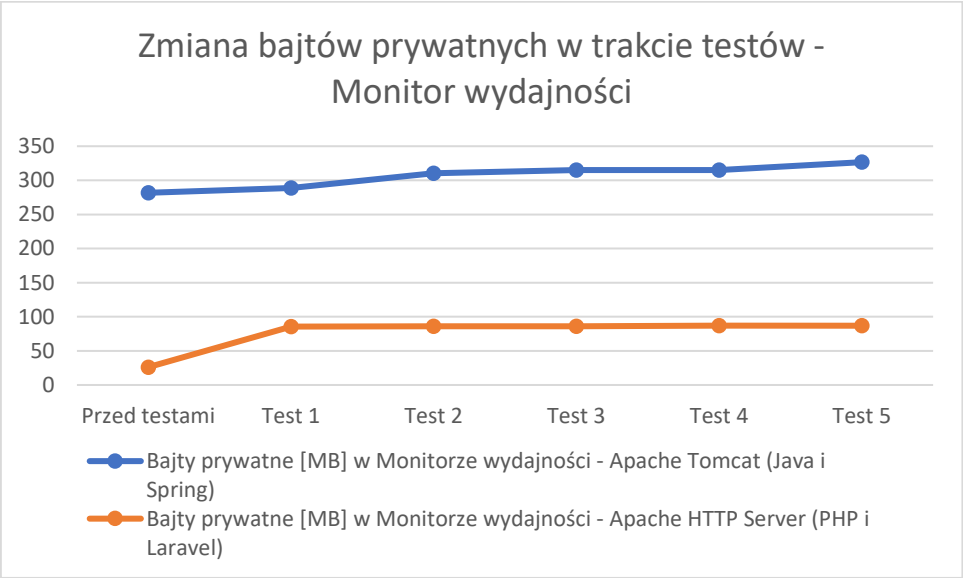
Rysunek 39. Wykres: Testy Apache Bench – Dane dot. żądań. **Źródło**: Opracowanie własne.

Testy zapotrzebowania na pamięć wykazały, że to oprogramowanie oparte na Javie i Springu wymagało więcej zasobów niż to napisane w PHP i Laravelu. Przed testami w przypadku pierwszego zestawu alokacji uległo 281,4 MB, a po wykonaniu testów Process Explorer wskazywał na 326,6 MB. Dla drugiego było to kolejno 26,2 MB i po testach 85,9 MB. Sytuacja została pokazana na wykresie w postaci Rysunki 40.



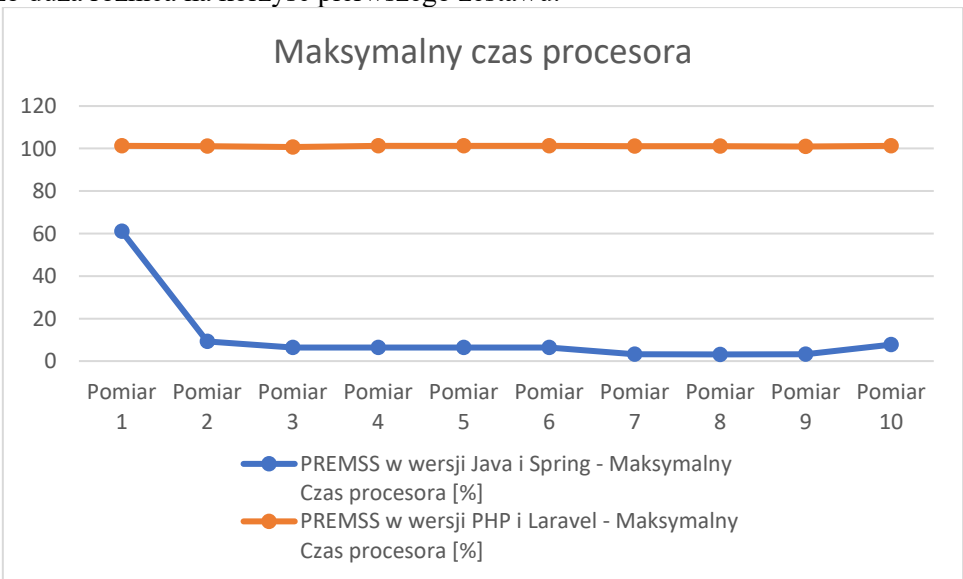
Rysunek 40. Wykres: Bajty prywatne przed i po testach – Process Explorer. **Źródło**: Opracowanie własne.

Wykres widoczny jako Rysunek 41 obrazuje zarejestrowaną w narzędziu Monitor wydajności zmianę w alokacji bajtów prywatnych podczas wykonywania poszczególnych testów (Tabela 16 oraz 17). Dla Apache Http Server z racji tego, że w ramach tego serwera działały dwa procesy, a jeden utrzymywał się na poziomie 10,3 MB (jest to zobrazowane na Rysunku 33 oraz 35) to dodatkowo do danych z wspomnianej tabeli właśnie taka liczba została dodana w przypadku każdego z testów. Otrzymane wyniki potwierdzają opisany na poprzedniej stronie rezultat, który został uzyskany z wykorzystaniem narzędzia Process Explorer oraz wniosek, że to PHP i Laravel zaalokował mniejszą ilość pamięci.



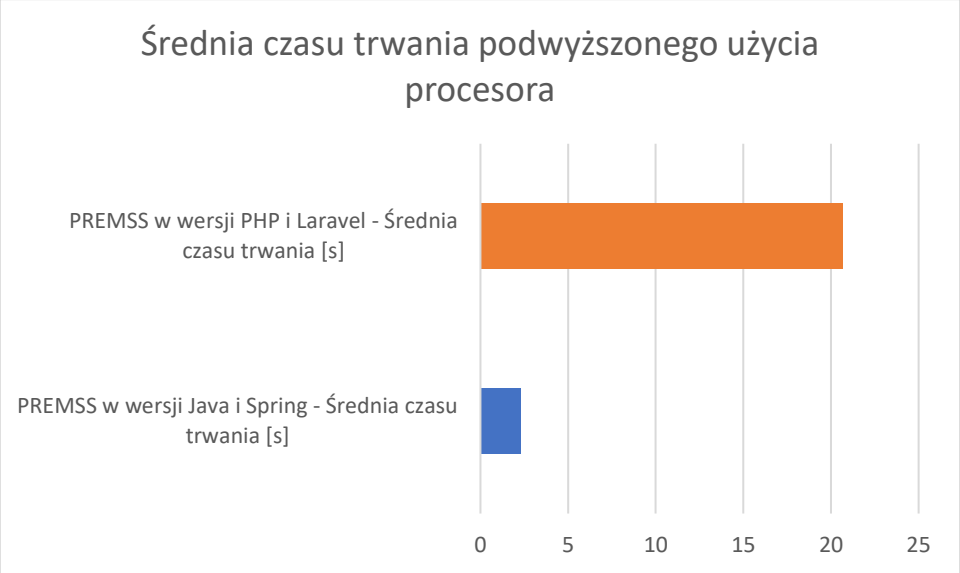
Rysunek 41. Wykres: Zmiana bajtów prywatnych w trakcie testów – Monitor wydajności. **Źródło**: Opracowanie własne.

Rysunek 42 na którym znajduje się wykres z maksymalnym czasem procesora, został stworzony na podstawie Tabeli 21 oraz 22. Ukazuje on, że w przypadku użycia tego zasobu, tym razem to aplikacja zaimplementowana w technologiach Java i Spring korzystała z procesora w mniejszym stopniu. Dodatkowo wykres pokazuje, że w przypadku wspomnianej technologii nastąpiła wyraźna poprawa, ponieważ przy pierwszym pomiarze maksymalny czas procesora wyniósł 61,101%, żeby w żadnym z następnych testów nie przekroczyć 9,244%. Średnia maksymalnego czasu procesora dla testów aplikacji PREMS w wersji Java i Spring wyniosła 11,276%, a w przypadku PHP i Laravela 101,123% - jest to więc bardzo duża różnica na korzyść pierwszego zestawu.



Rysunek 42. Wykres: Maksymalny czas procesora. **Źródło**: Opracowanie własne.

Dla dopełnienia obrazu użycia zasobu w postaci procesora, należało również wziąć pod uwagę jak długo jego trwało wzmożone wykorzystanie – w tym celu stworzony został wykres ze średnią czasu trwania podwyższonego użycia procesora w postaci Rysunku 43 (dane na podstawie Tabeli 21 oraz 22). W tym przypadku również to PHP i Laravel znajdujący się na serwerze Apache HTTP Server, wykazał gorszy rezultat. Średnia wyniosła bowiem tutaj 20,7 sekundy, podczas gdy dla aplikacji opartej o Javę i Springa na serwerze Apache Tomcat było to średnio jedynie 2,3 sekundy.



Rysunek 43. Wykres: Średnia czasu trwania podwyższonego użycia procesora. **Źródło**: Opracowanie własne.

Na końcu warto zaznaczyć, iż jedna i druga technologia w przeprowadzonych testach wykazała się taką samą niezawodnością – w każdym przypadku na 1000 żądań, nie było takich, które zakończyłyby się porażką (dane z Tabeli 18).

8. Podsumowanie

Niniejsza praca magisterska zgodnie z jej tytułem skupiona jest na analizie porównawczej oprogramowania internetowego z wykorzystaniem różnych technologii. Aby dostarczyć materiału do jej wykonania przedstawione zostały rozmaite aspekty, które charakteryzują różnego rodzaju oprogramowanie. Pokazano w niej, iż ich podziału można dokonać ze względu na trzy kategorie języków programowania, które dzielą je ze względu na sposób przetwarzania kodu źródłowego. Są to języki kompilowane, interpretowane oraz hybrydowe. Znalazło się w niej także miejsce na zaprezentowanie typów oprogramowania oraz paradygmatów, w oparciu o które wytwarzane są aplikacje. Do wykonania analizy porównawczej opierającej się na konkretnych technologiach wybrane zostały języki Java oraz PHP. Jako iż niniejsza praca skoncentrowana jest na oprogramowaniu internetowym, oprócz przedstawienia zagadnień związanych z tym rodzajem oprogramowania, zaprezentowane zostały także dwie platformy programistyczne dla wybranych języków: Spring oraz Laravel. To na ich podstawie zaimplementowana została aplikacja PREMSS, która dostarczyła materiału do opisania, a następnie porównania wytypowanych, konkretnych rozwiązań.

W ramach pracy przedstawione zostały cechy wspólne oraz różnice dla języków Java i PHP oraz platform programistycznych Spring i Laravel, które wykorzystywane są do tworzenia oprogramowania internetowego. Porównaniu uległa również wydajność implementacji opartych na poszczególnych rozwiązaniach oraz użycie zasobów. Autor ma nadzieję, iż praca dostarcza wartościowych informacji pozwalających ocenić zarówno te wybrane technologie jak i przedstawia aspekty oraz cechy języków i oprogramowania, pod kątem których można skonfrontować także inne zestawy wraz ze związanymi z nimi mechanizmami i narzędziami.

Bibliografia

[1]. Alex Allain. C++ Przewodnik dla początkujących, strona 16, 2014.

[2]. G. Michael Schneider, Judith L. Gersting. Invitation to Computer Science, edycja 8, strona 287, 2018

[3]. Strona Loyola Marymount University – Artykuł ‘Programming Paradigms’: https://cs.lmu.edu/~ray/notes/paradigms/, 01.08.2021 (data zaczerpnięcia informacji)

[4]. Strona University of Central Florida – Artykuł ‘Major Programming Paradigms’: http://www.cs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html, 01.08.2021 (data zaczerpnięcia informacji)

[5]. Strona projektu ‘Opracowanie programów nauczania na odległość na kierunku studiów wyższych – Informatyka’ powstała w wyniku współpracy czterech uczelni: Uniwersytetu Warszawskiego, Uniwersytetu Jagiellońskiego, Politechniki Poznańskiej i Politechniki Warszawskiej – Artykuł ‘Paradygmaty programowania/Wykład 1: Co to jest paradygmat programowania?’: http://wazniak.mimuw.edu.pl/index.php?title=Paradygmaty_programowania/Wyk%C5%82ad_1:_Co_to_jest_paradygmat_programowania%3F, 01.08.2021 (data zaczerpnięcia informacji)

[6]. Strona W3Schools - Artykuł ‘Java Polymorphism’: https://www.w3schools.com/java/java_polymorphism.asp, 30.12.2021 (data zaczerpnięcia informacji).

[7]. Strona W3Schools - Artykuł ‘Java Abstraction’: https://www.w3schools.com/java/java_abstract.asp, 30.12.2021 (data zaczerpnięcia informacji).

[8]. Anthony A. Aaby. Introduction to programming languages, strona 103, 2004.

[9]. Terrence W. Pratt, Marvin V. Zelkowitz. Programming Languages. Design and Implementation, edycja 4 strony 491-493, 2001.

[10]. Strona Yahoo Finance – Artykuł ‘Global Internet Services Industry (2020 to 2027) - Key Market Trends and Drivers - ResearchAndMarkets.com’: https://finance.yahoo.com/news/global-internet-services-industry-2020-123200726.html, 01.08.2021 (data zaczerpnięcia informacji).

[11]. Strona University of Alabama at Birmingham - Artykuł 'World Wide Web vs. Internet: What’s the Difference?': https://businessdegrees.uab.edu/blog/internet-vs-world-wide-web-whats-the-difference/, 04.01.2022 (data zaczerpnięcia informacji).

[12]. Strona University of Minnesota Duluth – Artykuł ‘The HyperText Transfer Protocol (HTTP) Protocol’: https://www.d.umn.edu/~gshute/net/http.html, 05.01.2022 (data zaczerpnięci informacji).

[13]. Strona Mozilla Developer Center – Artykuł ‘HTTP request methods’: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods, 05.01.2022 (data zaczerpnięcia informacji).

[14]. Strona Google Cloud – Artykuł ‘HTTP Response Codes’:

https://cloud.google.com/talent-solution/job-search/docs/http-response-codes,

05.01.2022 (data zaczerpnięcia informacji).

[15]. Strona Cornell University – Artykuł ‘Data Transfer: FTP - File Transfer Protocol’:

https://cvw.cac.cornell.edu/datatransfer/ftp, 05.01.2022 (data zaczerpnięcia informacji).

[16]. Strona Cornell University of Tulsa– Artykuł ‘Uniform Resource Locator’:

http://euler.mcs.utulsa.edu/~class_diaz/cs2503/Fall97/fall97/fall97_gen/node25.html, 05.01.2022 (data zaczerpnięcia informacji).

[17]. Strona Dartmouth College - Artykuł ‘Socket Programming’:

https://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html,

06.01.2022 (data zaczerpnięcia informacji).

[18]. Jon Duckett. HTML i CS. Zaprojektuj i zbuduj witrynę WWW, strona 9, 2014.

[19]. Marjin Haverbrake. Zrozumieć Javascript. Wprowadzenie do programowania, strona 24, 2015.

[20]. Strona W3Techs – Artykuł ‘Usage statistics of JavaScript as client-side programming language on websites’:

https://w3techs.com/technologies/details/cp-javascript/, 06.01.2022 (data zaczerpnięcia informacji).

[21]. Ian Sommerville. Software Engineering, edycja 9, strona 166, 2014.

[22]. Alfred V. Aho, Ravi Sethi, Jeffry D. Ullman. Kompilatory: Reguły, metody i narzędzia, strony 5-14, 2002 Wydawnictwa Naukowo-Techniczne

[23]. Robert W. Sebesta. Concepts of Programming Languages, edycja 11, strona 50, 2016.

[24]. David A. Watt (University of Glasgow, Scotland), Deryck F. Brown (The Robert Gordon University, Scotland), Programming Language Processors in Java. Compilers and interpreters, strona 35, 2000.

[25]. Robert W. Sebesta. Concepts of Programming Languages, edycja 11, strona 51, 2016.

[26]. Henrik Søndberg Karlsen, Erik Ramsgaard Wognsen, Mads Chr. Olesen, Ren´e Rydhof Hansen (Department of Computer Science, Aalborg University) – Artykuł ‘Study, Formalisation, and Analysis of Dalvik Bytecode’, strona 2.

[27]. Strona Android Authority – Artykuł ‘I want to develop Android apps — What languages should I learn?’:

https://www.androidauthority.com/develop-android-apps-languages-learn-391008/, 16.01.2022 (data zaczerpnięcia informacji).

[28]. Strona Statista - 'Global market share held by mobile operating systems': https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/, 16.01.2022 (data zaczerpnięcia informacji).

[29]. Strona Business of Apps – Artykuł ‘Android Statistics (2022)’:

https://www.businessofapps.com/data/android-statistics/, 16.01.2022 (data zaczerpnięcia informacji).

[30]. Oficjalna strona Androida – Artykuł ‘Meet Android Studio’:

https://developer.android.com/studio/intro, 16.01.2022 (data zaczerpnięcia informacji).

[31]. Prezentacja z oficjalnej strony Oracle ‘Java Card – The open application platform for secure elements’:

https://www.oracle.com/technetwork/java/javacard/overview/java-card-data-sheet-19-01-07-5250140.pdf, 17.01.2022 (data zaczerpnięcia informacji).

[32]. Oficjalna strona Java – Artykuł ‘Jak uzyskać oprogramowanie Java dla systemów osadzonych?’:

https://www.java.com/pl/download/help/java_embedded_pl.html, 17.01.2022 (data zaczerpnięcia informacji).

[33]. Oficjalna strona IBM – Artykuł ‘Java Platform’:

https://www.ibm.com/docs/en/i/7.4?topic=java-platform, 18.01.2022 (data zaczerpnięcia informacji).

[34]. Strona W3Schools – Artykuł ‘Java Packages’:

https://www.w3schools.com/java/java_packages.asp, 18.01.2022 (data zaczerpnięcia informacji).

[35]. Oficjalna strona Oracle – Artykuł ‘Differences between Java EE and Java SE’:

https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html, 18.01.2022 (data zaczerpnięcia informacji).

[36]. Lindholm Yellin, Bracha, Buckley. The Java Virtual Machine Specification. Java SE 8 Edition, strona 2, 2015

[37]. Oficjalna strona Oracle - Podstrona 'Java SE Core Technologies':

https://www.oracle.com/java/technologies/javase/javase-core-technologies-apis.html, 22.07.2021 (data zaczerpnięcia informacji).

[38]. Bart Baesens, Aimee Backiel, Seppe vanden Broucke. Beginning Java Programming. The Object-Oriented Approach, strony 15-19,2015

[39]. Oficjalna strona Oracle – Podstrona ‘PATH and CLASSPATH’:

https://docs.oracle.com/javase/tutorial/essential/environment/paths.html, 24.07.2021 (data zaczerpnięcia informacji).

[40]. Oficjalna strona PHP – Podstrona ‘History of PHP’:

https://www.php.net/manual/en/history.php.php,

19.01.2022 (data zaczerpnięcia informacji).

[41]. Oficjalna strona W3Techs – Podstrona ‘Usage statistics of PHP for websites’:

https://w3techs.com/technologies/details/pl-php,

19.01.2022 (data zaczerpnięcia informacji).

[42]. Oficjalna strona W3Techs – Podstrona ‘Usage statistics of content management systems’:

https://w3techs.com/technologies/overview/content_management,

19.01.2022 (data zaczerpnięcia informacji).

[43]. Oficjalna strona WordPress – Podstrona ‘Informacje’:

https://pl.wordpress.org/about/, 19.01.2022 (data zaczerpnięcia informacji).

[44]. Oficjalna strona Joomla – Podstrona dokumentacji ‘PHP essentials’:

https://docs.joomla.org/PHP_essentials, 19.01.2022 (data zaczerpnięcia informacji).

[45]. Oficjalna strona Drupal – Podstrona dotycząca licencjonowania i kodu źródłowego:

https://www.drupal.org/about/licensing#source-code, 19.01.2022 (data zaczerpnięcia informacji).

[46]. Oficjalna strona Bitrix – Podstrona ‘What is Bitrix Framework?’:

https://training.bitrix24.com/support/training/course/?COURSE_ID=68&CHAPTER_ID=05937&LESSON_PATH=5936.5937, 19.01.2022 (data zaczerpnięcia informacji).

[47]. Oficjalna strona OpenCart – Informacja o bazowaniu na podstawie PHP zawarta jest w nagłówku dokumentu HTML, znaczniku description:

https://www.opencart.com/, 19.01.2022 (data zaczerpnięcia informacji).

[48]. Oficjalna dokumentacja programistyczna Prestashop – Podstrona ‘Fundamentals of PrestaShop Development’: https://devdocs.prestashop.com/1.7/basics/introduction/, 19.01.2022 (data zaczerpnięcia informacji).

[49]. Strona Javatpoint – Artykuł ‘WordPress vs. Weebly’:

https://www.javatpoint.com/wordpress-vs-weebly, 19.01.2022 (data zaczerpnięcia informacji).

[50]. Słownik języka polskiego PWN – hasło ‘skrypt’:

https://sjp.pwn.pl/slowniki/skrypt.html, 19.01.2022 (data zaczerpnięcia informacji).

[51]. Strona Loyola Marymount University – Artykuł ‘Scripting Languages’:

https://cs.lmu.edu/~ray/notes/scriptinglangs/, 19.01.2022 (data zaczerpnięcia informacji).

[52]. G. Michael Schneider, Judith L. Gersting. Invitation to Computer Science, edycja 8,

strona 505 2018.

[53]. Strona Waukesha County Technical College - Podstrona ‘PHP Web Development - Course Description’: https://www.wctc.edu/academics/programs-courses/course-search/course-search-listing.php?code=152&num=129, 19.01.2022 (data zaczerpnięcia informacji).

[54]. Strona Northestern University - Artykuł ‘The 10 Most Popular Programming Languages to Learn in 2022’: https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/, 19.01.2022 (data zaczerpnięcia informacji).

[55]. Oficjalna strona PHP – Podstrona ‘Types’:

https://www.php.net/manual/en/language.types.intro.php, 19.01.2022 (data zaczerpnięcia informacji).

[56]. Oficjalna strona PHP – Podstrona ‘Garbage Collection’:

https://www.php.net/manual/en/features.gc.php, 19.01.2022 (data zaczerpnięcia informacji).

[57]. Oficjalna strona PHP – Podstrona ‘Classes and Objects - Introduction’:

https://www.php.net/manual/en/oop5.intro.php, 18.01.2022 (data zaczerpnięcia informacji).

[58]. Oficjalna strona PHP – Podstrona ‘PHP 4.3.0 Release Announcement’:

https://www.php.net/releases/4_3_0.php, 18.01.2022 (data zaczerpnięcia informacji).

[59]. Oficjalna strona Pear – Podstrona ‘What is PEAR?’:

https://pear.php.net/manual/en/about.pear.php, 18.01.2022 (data zaczerpnięcia informacji).

[60]. Oficjalna strona PECL: https://pecl.php.net/, 18.01.2022 (data zaczerpnięcia informacji).

[61]. Oficjalna Strona GTK-PHP – sekcja ‘Download’: http://gtk.php.net/download.php?language=en-US, 18.01.2022 (data zaczerpnięcia informacji).

[62]. Oficjalna strona wxPHP: https://wxphp.org/, 18.01.2022 (data zaczerpnięcia informacji).

[63]. Oficjalna strona wxPHP – Artykuł ‘New 3.0.2.0 Release!’: https://wxphp.org/news/new-3020-release, 18.01.2022 (data zaczerpnięcia informacji).

[64]. Oficjalna dokumentacja PHP - https://www.php.net/docs.php, 14.02.2022 (data zaczerpnięcia informacji).

[65]. Oficjalna strona PHP – Podstrona ‘Phar – Intoduction’: https://www.php.net/manual/en/intro.phar.php, 18.01.2022 (data zaczerpnięcia informacji).

[66]. Josh Lockhart, Modern PHP. New features and good practices. strona 3, 2015. O'Reilly.

[67]. Portal Phpbuilder poświęcony językowi PHP i istniejący od 1999 roku - Artykuł ‘PHP and Zend Engine Internals’: https://phpbuilder.com/php-and-zend-engine-internals/, 29.07.2021 (data zaczerpnięcia informacji).

[68]. Oficjalna strona Zend – Artykuł ‘Exploring the New PHP JIT Compiler’:

https://www.zend.com/blog/exploring-new-php-jit-compiler, 08.01.2022 (data zaczerpnięcia informacji).

[69]. Oficjalna strona PHP – Podstrona ‘PHP 5.5.0 Release Announcement’:

https://www.php.net/releases/5_5_0.php, 08.01.2022 (data zaczerpnięcia informacji).

[70]. Oficjalna strona PHP – Podstrona ‘PHP 8 Released’:

https://www.php.net/releases/8.0/en.php, 09.01.2022 (data zaczerpnięcia informacji).

[71]. Mariusz Trzaska (Polsko Japońska Akademia Technik Komputerowych) - Artykuł ‘WebODRA - A Web Framework for the Object-Oriented DBMS ODRA’, 2013 (publikacja dostępna jest pod adresem: http://thinkmind.org/index.php?view=article&articleid=web_2013_1_10_40010,

data odczytu 24.01.2022).

[72]. Oficjalna strona jQuery UI – Podstrona ‘About jQuery UI’:

https://jqueryui.com/about/, data 28.01.2022 (data zaczerpnięcia informacji).

[73]. Oficjalna strona zintegrowanego środowiska programistycznego ‘IntelliJ IDEA’: https://www.jetbrains.com/idea/, 14.02.2022 (data zaczerpnięcia informacji).

[74]. Oficjalna strona edytora kodu ‘Visual Studio Code’: https://code.visualstudio.com/, 14.02.202 (data zaczerpnięcia informacji).

[75]. Strona dotycząca rozszerzenia ‘PHP Debug’ dla Visual Studio Code: https://marketplace.visualstudio.com/items?itemName=xdebug.php-debug, 14.02.2022 (data zaczerpnięcia informacji).

[76]. Oficjalna dokumentacja Spring – Podstrona ‘Spring Framework Overview’:

https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html#overview,

data 26.01.2022 (data zaczerpnięcia informacji).

[77]. Oficjalna dokumentacja Spring – Podstrona ‘Introduction to the Spring Framework’:

https://docs.spring.io/spring-framework/docs/4.3.19.RELEASE/spring-framework-reference/html/overview.html, data 26.01.2022 (data zaczerpnięcia informacji).

[78]. Oficjalna dokumentacja Spring – Podstrona ‘Marshalling XML using O/X Mappers’:

https://docs.spring.io/spring-framework/docs/4.3.19.RELEASE/spring-framework-reference/html/oxm.html, data 26.01.2022 (data zaczerpnięcia informacji).

[79]. Oficjalna dokumentacja Spring – Podstrona ‘Aspect Oriented Programming with Spring’:

https://docs.spring.io/spring-framework/docs/4.3.19.RELEASE/spring-framework-reference/html/aop.html#aop-introduction, data 26.01.2022 (data zaczerpnięcia informacji).

[80]. Oficjalna strona Spring – Dokumentacja pakietu ‘org.springframework.instrument’: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/instrument/package-summary.html, 27.01.2022 (data zaczerpnięcia informacji).

[81]. Oficjalna strona Oracle – Dokumentacja interfejsu ‘Instrumentation’: https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/Instrumentation.html?is-external=true, 27.01.2022 (data zaczerpnięcia informacji).

[82]. Oficjalna strona Spring – Podstrona ‘Build Systems’: https://docs.spring.io/spring-boot/docs/2.1.17.BUILD-SNAPSHOT/reference/html/using-boot-build-systems.html, 27.01.2022 (data zaczerpnięcia informacji).

[83]. Oficjalne repozytorium Spring Framework na GitHub – Sekcja ‘releases’: https://github.com/spring-projects/spring-framework/releases,

27.01.2022 (data zaczerpnięcia informacji).

[84]. Craig Walls, Spring w akcji, wydanie 5 strona 26, 2015 Helion.

[85]. Oficjalna strona Spring – Podstrona ‘Spring Framework Overview’, sekcja ‘5. Getting Started’:

https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html#overview,

27.01.2022 (data zaczerpnięcia informacji).

[86]. Oficjalne repozytorium Spring Boot na GitHub – Sekcja ‘relaeses’: https://github.com/spring-projects/spring-boot/releases, 27.01.2022 (data zaczerpnięcia informacji).

[87]. Oficjalna strona Spring – Podstrona ‘Spring Initializr Reference Guide’: https://docs.spring.io/initializr/docs/current/reference/html/#introduction, 27.01.2022 (data zaczerpnięcia informacji).

[88]. Oficjalna strona Spring Initialzr: https://start.spring.io, 14.02.2022 (data zaczerpnięcia informacji).

[89]. Oficjalne repozytorium Maven – Podstrona dotycząca zależności ‘Spring Boot Starter Data JPA’:

https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa, 30.01.2022 (data zaczerpnięcia informacji).

[90]. Oficjalna strona Laravel: https://laravel.com/docs/8.x, 27.01.2022 (data zaczerpnięcia informacji).

[91]. Martin Bean. Laravel 5 Essentials, strona 6, 2015.

[92]. Oficjalna strona Laravel – podstrona ‘Release Notes’: https://laravel.com/docs/8.x/releases, 27.01.2022 (data zaczerpnięcia informacji).

[93]. Oficjalna strona Laravel – dokumentacja poświęcona ‘Artisan CLI’:

https://laravel.com/docs/5.0/artisan, 30.01.2022 (data zaczerpnięcia informacji).

[94]. Oficjalna strona Laravel – podstrona ‘Database: Migrations’:

https://laravel.com/docs/8.x/migrations, 30.01.2022 (data zaczerpnięcia informacji).

[95]. Paweł Kamiński. Laravel. Wstęp do programowania aplikacji internetowych, strona 80, 2020 Helion.

[96]. Oficjalna strona Laravel – podstrona ‘Blade Templates’:

https://laravel.com/docs/8.x/blade, 31.01.2022 (data zaczerpnięcia informacji).

[97]. Oficjalna strona Laravel – podstrona ‘Eloquent: Getting Started’:

https://laravel.com/docs/8.x/eloquent, 31.01.2022 (data zaczerpnięcia informacji).

[98]. Oficjalna strona Tiobe – podstrona ‘TIOBE Programming Community Index Definition’:

https://www.tiobe.com/tiobe-index/programming-languages-definition/,

21.01.2022 (data zaczerpnięcia informacji).

[99]. Oficjalna strona Tiobe – Artykuł ‘TIOBE Index for January 2022’:

https://www.tiobe.com/tiobe-index/, 21.01.2022 (data zaczerpnięcia informacji).

[100]. Strona Stackoverlow.com – Rezultat ankiety dotyczącej stosowania języków programowania, skryptowania oraz znaczników z roku 2021:

https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language, 22.01.2022 (data zaczerpnięcia informacji).

[101]. Oficjalna strona W3Techs – Podstrona ‘Usage statistics of server-side programming languages for websites’: https://w3techs.com/technologies/overview/programming_language,

22.01.2022 (data zaczerpnięcia informacji).

[102]. Cay S. Horstmann. Java Podstawy, wydanie 11 strona 27, 2020 Helion.

[103]. Kevin Tatore, Pater MacIntyre, Rasmus Lerdorf. Programming PHP. Creating Dynamic Web Pages, edycja 3 strona 332, 2013 O’Reilly.

[104]. Oficjalna strona PHP – Podstrona ‘pthreads - Introduction:

https://www.php.net/manual/en/intro.pthreads.php, 23.01.2022 (data zaczerpnięcia informacji).

[105]. Oficjalna strona PHP – Podstrona ‘parallel – Introduction’:

https://www.php.net/manual/en/intro.parallel.php, 23.01.2022 (data zaczerpnięcia informacji).

[106]. Oficjalna strona PECL – Podstrona paczki ‘parallel’:

https://pecl.php.net/package/parallel, 23.01.2022 (data zaczerpnięcia informacji).

[107]. Strona GitHub – Repozytorium rozszerzenia ‘parallel’ dla języka PHP:

https://github.com/krakjoe/parallel, 23.01.2022 (data zaczerpnięcia informacji).

[108]. Oficjalna dokumentacja Javy prowadzona przez Oracle – Podstrona ‘Class Thread’:

https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html, 23.01.2022 (data zaczerpnięcia informacji).

[109]. Oficjalna strona PHP – Podstrona poświęcona funkcji ‘array_push’:

https://www.php.net/manual/en/function.array-push.php, 23.01.2022 (data zaczerpnięcia informacji).

[110]. Oficjalna strona PHP – Podstrona poświęcona funkcji ‘gettype’:

https://www.php.net/manual/en/function.gettype.php, 23.01.2022 (data zaczerpnięcia informacji).

[111]. Strona Stackoverlow.com – Rezultat ankiety dotyczącej stosowania internetowych platform programistycznych: https://insights.stackoverflow.com/survey/2021#web-frameworks, 02.02.2022 (data zaczerpnięcia informacji).

[112]. Oficjalna strona Spring – Podstrona ‘Getting Started - Quick-start Spring CLI Example’: https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html#getting-started.installing.cli.quick-start, 02.02.2022 (data zaczerpnięcia informacji).

[113]. Oficjalna strona Spring – Podstrona ‘Building an Application with Spring Boot’:

https://spring.io/guides/gs/spring-boot/, 02.02.2022 (data zaczerpnięcia informacji).

[114]. Oficjalna strona Laravel – Podstrona ‘Service Container’:

https://laravel.com/docs/8.x/container, 02.02.2022 (data zaczerpnięcia informacji).

[115]. Oficjalna strona Laravel – Podstrona ‘Fascades’:

https://laravel.com/docs/8.x/facades, 02.02.2022 (data zaczerpnięcia informacji).

[116]. Oficjalny blog Spring – artykuł ‘Deploying Spring Boot Applications’:

https://spring.io/blog/2014/03/07/deploying-spring-boot-applications,

03.02.2022 (data zaczerpnięcia informacji).

[117]. Oficjalna strona Laravel – Dokumentacja:

https://laravel.com/docs/7.x, 03.02.2022 (data zaczerpnięcia informacji).

[118]. Oficjalna strona Laravel – Sekcja ‘Install Laravel’:

https://laravel.com/docs/4.2#install-laravel, 03.02.2022 (data zaczerpnięcia informacji).

[119]. Oficjalna strona Spring – Artykuł ‘Spring Boot Gradle Plugin Reference Guide’:

https://docs.spring.io/spring-boot/docs/2.1.17.BUILD-SNAPSHOT/gradle-plugin/reference/html/, 03.02.2022 (data zaczerpnięcia informacji).

[120]. Oficjalna strona Spring – Podstrona ‘Build Systems - Ant’:

https://docs.spring.io/spring-boot/docs/2.1.17.BUILD-SNAPSHOT/reference/html/using-boot-build-systems.html#using-boot-ant, 03.02.2022 (data zaczerpnięcia informacji).

[121]. Oficjalna strona Laravel – Podstrona ‘Testing’:

https://laravel.com/docs/8.x/testing, 03.02.2022 (data zaczerpnięcia informacji).

[122]. Sebastian Jędrych, Bartłomiej Jędruszak, Beata Pańczyk (Politechnika Lubelska) - Artykuł pod tytułem ‘Tworzenie aplikacji internetowych na platformie JEE i PHP – analiza porównawcza’, opublikowany na łamach ‘Journal Computer Sciences Institute’ - JCSI 11 (2019).

[123]. Strona Microsoft – Artykuł ‘Using Performance Monitor to Find a User-Mode Memory Leak’: https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/using-performance-monitor-to-find-a-user-mode-memory-leak, 16.02.2022 (data zaczerpnięcia informacji).

Spis rysunków

Spis tabel

Spis kodu