JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 4 (2017) 137-141
Wysłane: 2017-05-29
Przyjęte: 2017-06-06

Aktualne narzędzia wytwarzania oprogramowania na platformie JEE

Paweł Ozdoba*, Beata Pańczyk

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł ma charakter przeglądowy i opisuje aktualnie stosowane narzędzia wspomagające proces wytwarzania oprogramowania na platformę Java Enterprise Edition. W artykule dokonano wyboru omawianych narzędzi w oparciu o statystyki wykorzystania udostępnione na stronach internetowych. Przeanalizowano zintegrowane środowiska programistyczne, narzędzia do kontroli wersji kodu, narzędzia automatyzujące budowę oprogramowania oraz analizujące poprawność kodu. W niniejszej pracy wskazano najbardziej optymalne (według autora) narzędzia, wykorzystując je do budowy przykładowej aplikacji testowej w środowisku JEE.

Słowa kluczowe: JEE; środowiska programistyczne; repozytorium binarne; wersjonowanie; statyczna analiza kodu

*Autor do korespondencji.

Adres e-mail: PawelOzdoba07@gmail.com

The current software tools for the JEE platform

Paweł Ozdoba*, Beata Pańczyk

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

Abstract. This article is an overview and describes the currently used tools supporting software development process for Java Enterprise Edition. The tools were selected based on the usage statistics provided by websites. Analysis includes integrated development environments, code revision tools, software automation tools, and code validation tools. This paper shows the most optimal (according to the author) tools, using them to build sample test application in JEE.

Keywords: JEE; integrated development environment; binary repository; versioning; static code analysis

*Corresponding author.

E-mail address: PawelOzdoba07@gmail.com

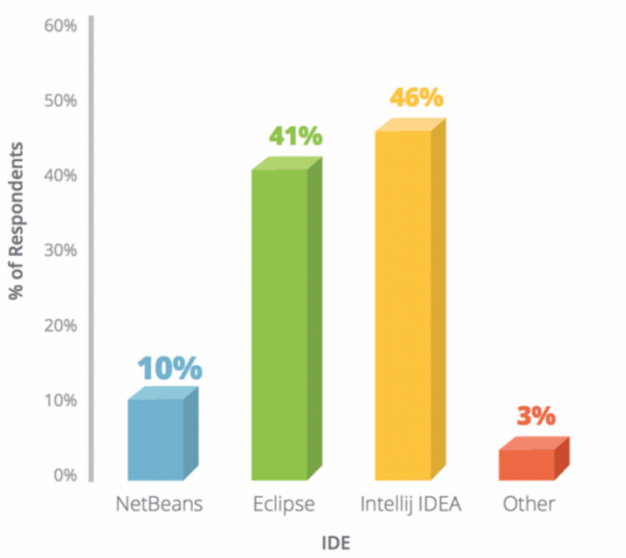
1. Wstęp

Szybki rozwój narzędzi programistycznych umożliwia efektywną pracę nawet z bardzo skomplikowanymi projektami. Wytwarzanie aplikacji Javy, z zachowaniem właściwego poziomu jakości, nie jest łatwym zadaniem. Jednak dzięki wykorzystaniu odpowiednich narzędzi programistycznych, można podnieść przewidywalność, niezawodność i efektywność procesu wytwarzania oprogramowania oraz sprawić, że projektowanie, kodowanie oraz wdrażanie wysokiej jakości aplikacji Java będzie maksymalnie uproszczone. Wdrażanie i stosowanie większości narzędzi programistycznych nie jest trudne, jednak wymaga wysiłku związanego z weryfikacją bieżących praktyk pod kątem zgodności z nowymi trendami. Dla każdego zadania związanego z kolejnym etapem wytwarzania aplikacji, istnieje przynajmniej kilka skutecznych narzędzi typu open source. Przybliżenie tych narzędzi oraz próba wskazania najbardziej optymalnego ich zestawu jest głównym celem niniejszego artykułu.

2. Narzędzia wspomagające proces tworzenia aplikacji JEE

2.1. Zintegrowane środowiska programistyczne

Najpopularniejsze środowiska programistyczne dla platformy Java Enterprise Edition to **Eclipse**, **NetBeans** i **IntelliJ.** Rysunki 1 i 2 przedstawiają popularność środowisk programistycznych dla Javy w roku 2016 na bazie portali *zeroturnaround.com* oraz *www.baeldung.com.* Rysunek 3 prezentuje udział poszczególnych IDE na przełomie lat 2012-2016.



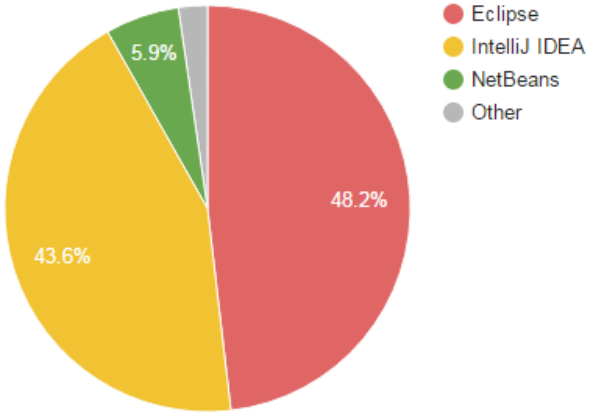
Rys.1. Popularność IDE dla Javy [1]

Porównując wartości liczbowe z rokiem 2012 (Rys. 3), widać wyraźny wzrost popularności IntelliJ, kosztem Eclipse..

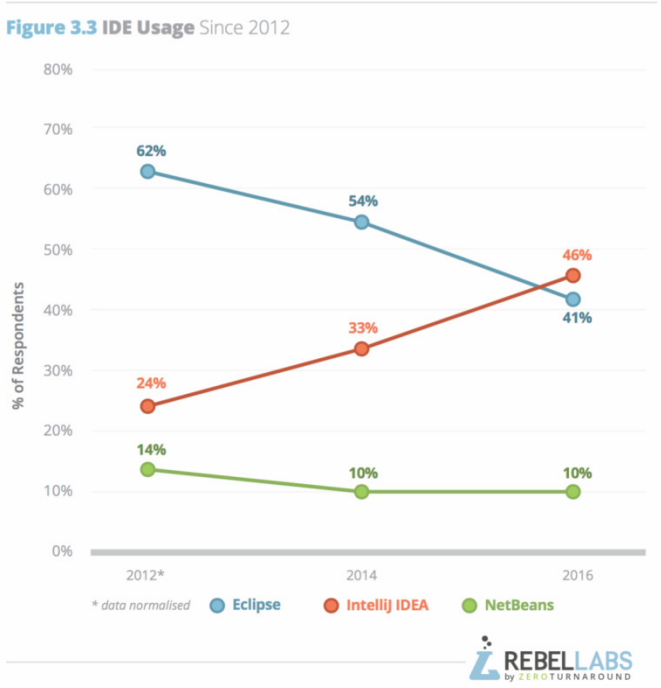
2.2. Kontrola wersji kodu

Systemy kontroli wersji (lub systemy zarządzania kodem źródłowym) pozwalają na przechowywanie wielu wersji plików, umożliwiając dostęp do ich poprzednich wersji.

Najpopularniejsze aktualne narzędzia wykorzystywane do kontroli wersji kodu to: **SVN** i **GIT** (Rys. 4).



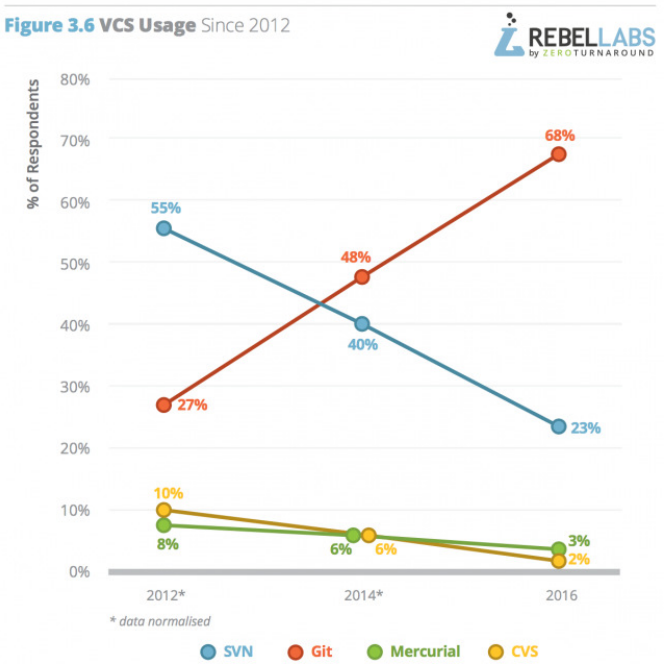
Rys.2. Popularność IDE dla Javy [2]



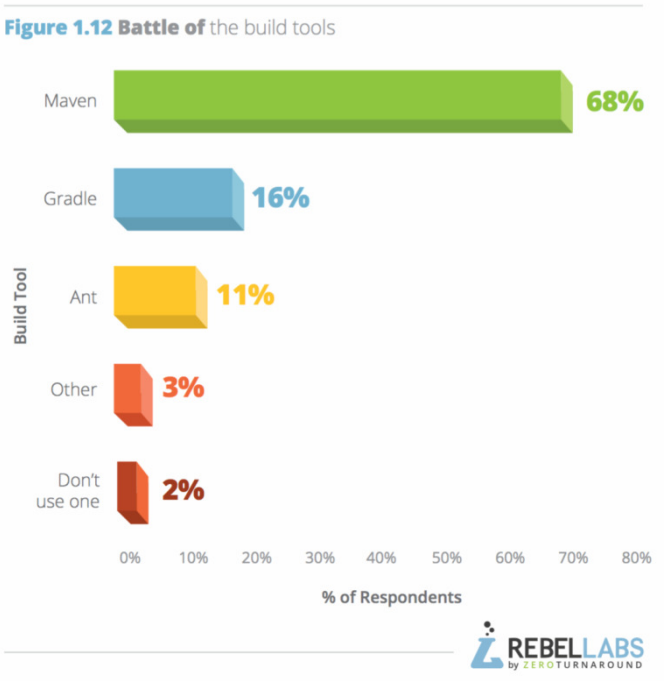
Rys.3. Popularność IDE w latach 2012-16 [3]

2.3. Automatyzacja budowy oprogramowania

W prostych projektach testowanie i kompilowanie oprogramowania można wykonać za pomocą narzędzi dostępnych w IDE. Duży projekt, który zawiera więcej niż jeden plik wykonywalny, wymaga większej kontroli. Konieczne staje się wtedy wykorzystanie narzędzia do automatycznej kompilacji oprogramowania, które uprości zarządzanie strukturą i zależnościami projektu.
Najpopularniejsze narzędzia automatyzujące budowę oprogramowania to: **ANT** , **MAVEN** oraz **GRADLE** (Rys. 5).



Rys.4. Popularność narzędzi kontroli wersji kodu [3]



Rys.5. Popularność narzędzi automatyzacji oprogramowania [1]

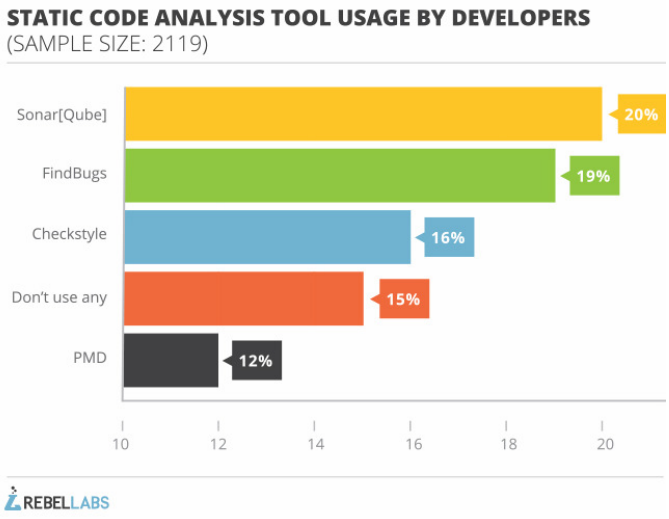
2.4. Analiza statyczna poprawności kodu

Napisanie elastycznego, łatwego w utrzymaniu i wysokiej jakości oprogramowania nie jest prostym zadaniem. Bardzo pomocne są wzorce programistyczne, zebrane w formie standardów, wskazujące dobre praktyki pisania kodu. Największe problemy występują, gdy programiści stosujący odmienne konwencje programowania, pracują nad tą samą aplikacją. W celu rozwiązania tych trudności powstało wiele narzędzi do analizy oprogramowania np. **SonarQube** czy **CheckStyle** (Rys. 6).

Analiza statyczna kodu obejmuje:

- analizę poprawności składni;

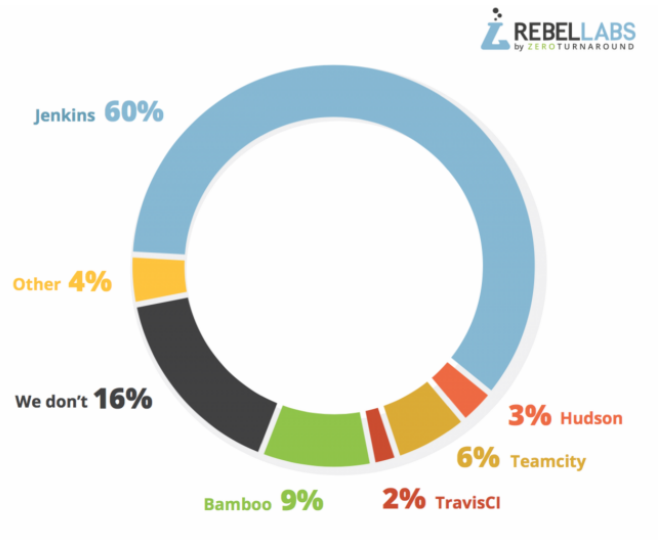
- błędy występujące przy danych wejściowych;
- sprawdzenie istnienia funkcji nieaktualnych (niebezpiecznych), przepełnienia bufora, wycieków pamięci;
- wykrycie stosowania zmiennych niezainicjalizowanych;
- szukanie nieużywanych fragmentów kodu, powtórzeń;
- podpowiedzi dotyczące poprawienia wydajności kodu;
- wskazówki co dostosowania standardów kodowania.



Rys.6. Popularność narzędzi do analizy statycznej kodu [1]

2.5. Serwery automatyzujące procesy wytwarzania kodu

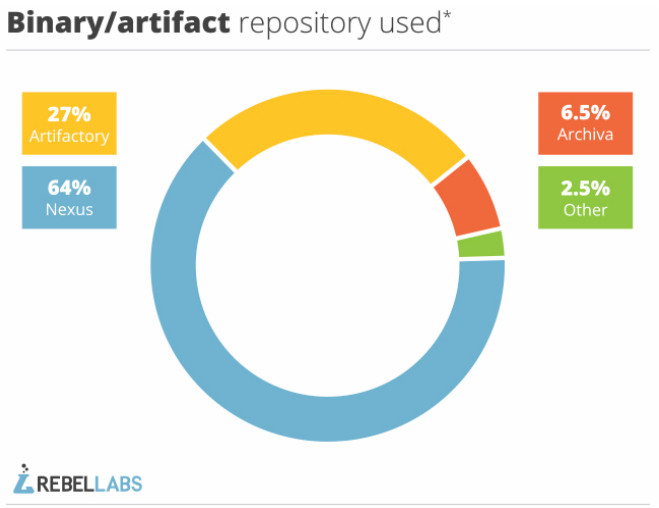
Ciągła integracja wymaga, aby za każdym razem, gdy programista wprowadzi jakąkolwiek zmianę, cały projekt został skompilowany i poddany różnokierunkowym testom zautomatyzowanym. Konieczna jest zatem stała dostępność działającego oprogramowania. Popularne narzędzia służące do ciągłej integracji to np.: **Jenkins, Bamboo, Hudson. Continuum**, **CruiseControl** (Rys 7).



Rys.7. Popularne narzędzia do ciągłej integracji kodu [1]

2.6. Serwery przechowujące dystrybucje aplikacji

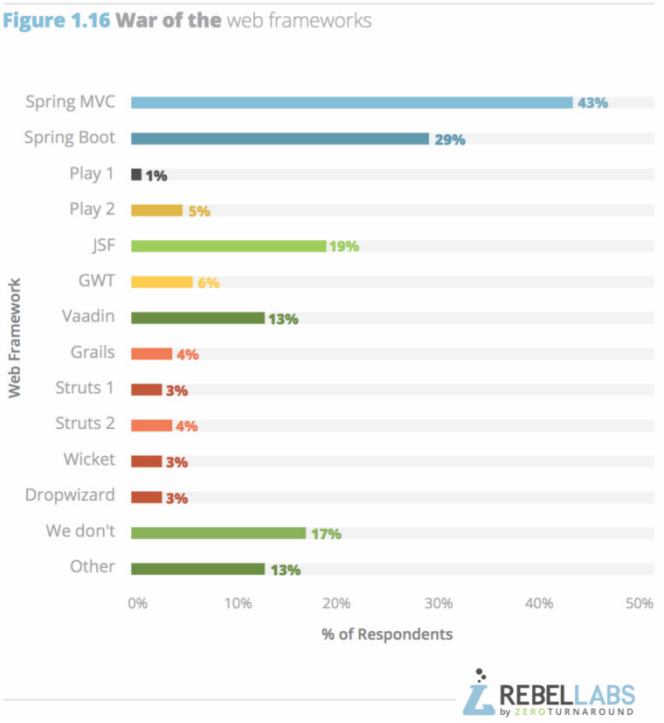
Repozytorium binarne jest to repozytorium oprogramowania dla pakietów, artefaktów i odpowiadających im metadanych. Może być używane do przechowywania plików binarnych. Serwer przechowujący binaria (dystrybucje aplikacji) służy do optymalizacji przechowywania i pobierania plików binarnych, które są stosowane podczas etapu wytwarzania oprogramowania. Po usunięciu całego repozytorium artefaktów, można przywrócić wszystkie wartościowe informacje. Na rysunku 8 przedstawiono zestawienie najpopularniejszych narzędzi do przechowywania dystrybucji kodu.



Rys.8. Popularność narzędzi do przechowywania dystrybucji kodu [1]

2.7. Frameworki do tworzenia aplikacji internetowych

Aplikacje internetowe w środowisku JEE można tworzyć z wykorzystaniem wielu dostępnych szkieletów programistycznych. Najpopularniejsze z nich zestawiono na rysunku 9.



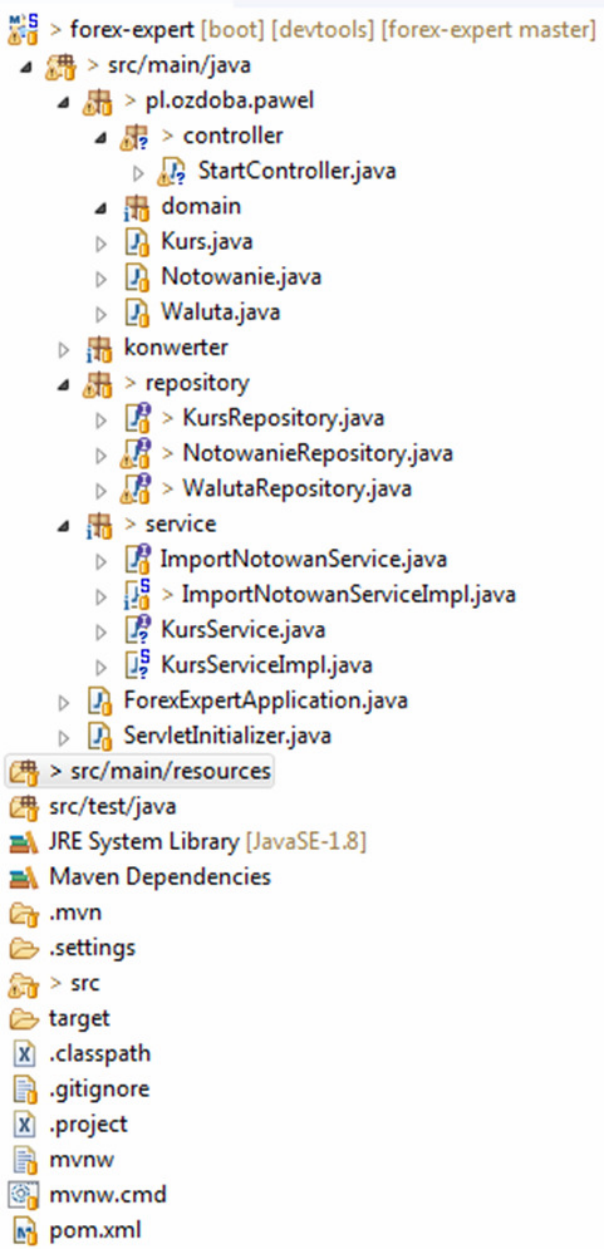
Rys.9. Popularność narzędzi do zarządzania zadaniami programistów [1]

3. Aplikacja testowa

Prosta aplikacja testowa pobiera ze strony [4] dane (wartość waluty, nazwę oraz datę notowania) za pomocą technologii REST i zapisuje je do bazy MySQL. Proces tworzenia przykładowej aplikacji internetowej „forex-expert" pokazano na przykładzie wykorzystania wybranego zestawu aktualnych i popularnych narzędzi:

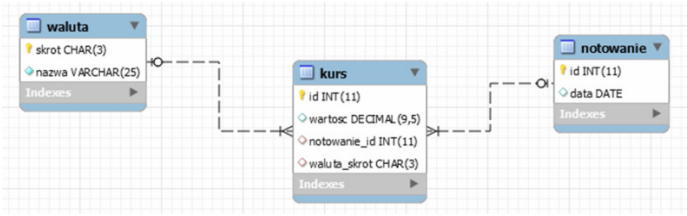
- Eclipse IDE (zintegrowane środowisko programistyczne),
- Maven (automatyzacja zarządzania projektem) [5],
- Spring Boot (szkielet aplikacji internetowej) [6],
- Git (system kontroli wersji kodu) i Bitbucket (zarządzanie repozytorium),
- SonarQube (statyczna analiza kodu),
- Jenkins (ciągła integracja),
- JFrog Artifactory (repozytorium binarne) [7].

Rysunek 10 przedstawia strukturę projektu Spring Boot [5] w Eclipse z podziałem na warstwy aplikacji.



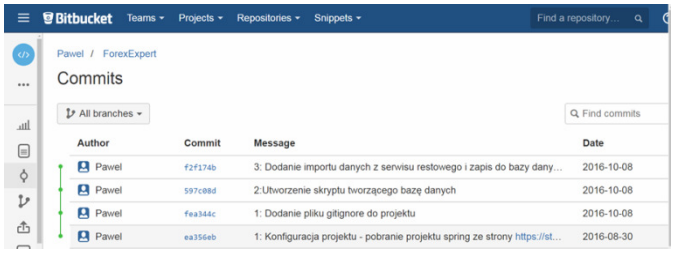
Rys.10. Struktura aplikacji testowej Spring Boot

W aplikacji zdefiniowano między innymi encje: *Kurs, Notowanie* i *Waluta*. Do operacji na bazie danych (Rys. 11) zastosowano technologię Hibernate.



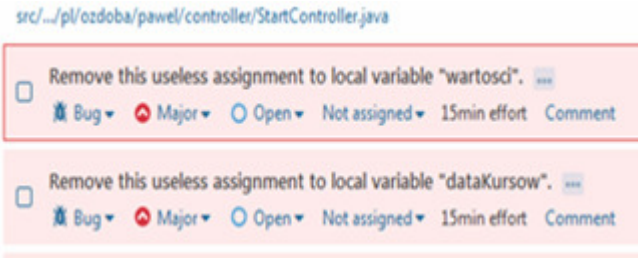
Rys. 11. Baza danych dla aplikacji testowej

BitBucket [8] został wykorzystany do zarządzania repozytorium GIT z wersjami projektów. Przykładowe zmiany repozytorium kodu GIT przedstawia rysunek 12.

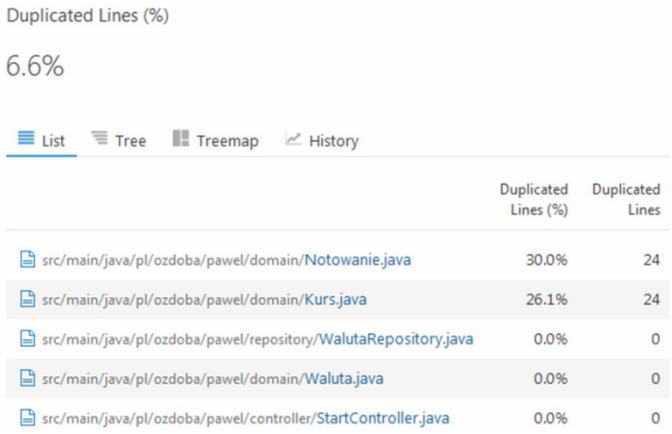


Rys. 12. Bitbucket i zmiany repozytorium GIT

Przykład analizy jakości kodu za pomocą SonarQube przedstawia rysunek 13 (uwagi o nie używaniu zmiennych lokalnych) oraz rysunek 14 (informuje o procencie powielenia linii kodu).

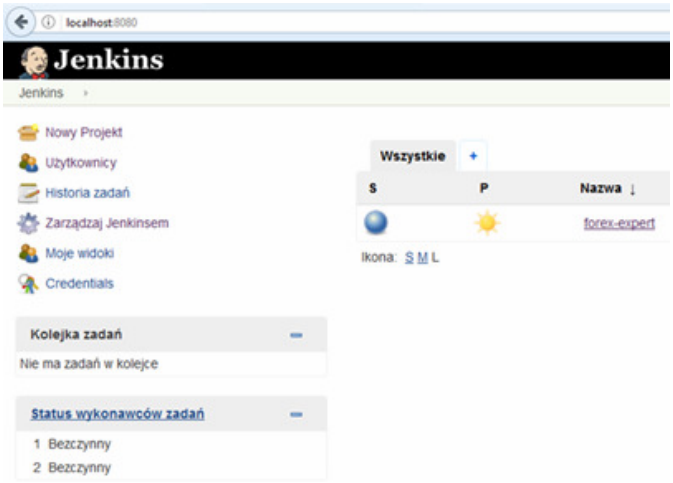


Rys. 13. Analiza jakości kodu za pomocą SonarQubej – uwaga dotycząca zmiennych lokalnych



Rys. 14. Analiza jakości kodu za pomocą SonarQubej – uwaga o duplikacji kodu

Do prawidłowej pracy narzędzia Jenkins (wybranego do procesu ciągłej integracji [9]), niezbędne jest zintegrowanie go za pomocą Maven z testową aplikacją (Rys. 15).



Rys. 15. Integracja na bazie serwera Jenkins

4. Wnioski

Po przeanalizowaniu aktualnych narzędzi wykorzystywanych do wytwarzania oprogramowania na platformę Java EE, wskazano te, które cieszą się największą popularnością wśród programistów na konkretnych etapach wytwarzania oprogramowania. Wiele darmowych narzędzi pozwala tworzyć komercyjne oprogramowanie. Wszystkie wybrane narzędzia mają rozbudowaną dokumentację oraz materiały dodatkowe pomagające w szybki sposób opanować podstawowe funkcjonalności. Dużą zaletą jest to, że większość z analizowanych narzędzi pozwala na integrację z innymi narzędziami.

Literatura

[1] https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/, [23.05.2017].

[2] http://www.baeldung.com/java-ides-2016, [24.05.2017].

[3] https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016-trends/, [23.05.2017].

[4] http://api.fixer.io/latest?base=PLN, [23.05.2017].

[5] https://maven.apache.org/ [25.04.2017].

[6] Gutierrez F., Pro Spring Boot , Apress, 2016.

[7] https://www.jfrog.com/binary-repository/, [24.05.2017].

[8] http://www.intenso.pl/atlassian/pl/atlassian-bitbucket, [23.05.2017].

[9] Humble J., Farley D.: Ciągłe dostarczanie oprogramowania. Automatyzacja kompilacji, testowania i wdrażania, Helion, 2015.