POLSKO-JAPOŃSKA WYŻSZA SZKOŁA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Marcin Cwalina
Nr albumu s6495

Generyczny system do analizy portali internetowych

Praca magisterska napisana
pod kierunkiem:

dr inż. Mariusz Trzaska

Warszawa, luty 2011

Streszczenie

Praca traktuje o problemie analizy zawartości portali internetowych. Skupia się na kwestii wykorzystania do tego celu narzędzi i technologii informatycznych. Autor kładzie nacisk na generyczne rozwiązanie pozwalające na szybkie i efektywne pobieranie informacji a także przedstawienie wyników analiz w zwartej i przejrzystej formie.

Praca nad problemem pozwoliła na stworzenie prototypu systemu, który odzwierciedla zaproponowane rozwiązania. Pozwala on na uruchomienie, dostarczonych wraz z nim, dwóch modułów zbierających dane oraz sześciu modułów do analizy danych. Prototyp jest aplikacją webową stworzoną przy wykorzystaniu technologii JavaEE 6, szablonów JSP, standardu JSTL i innych wolnych narzędzi przydatnych w implementacji.

SPIS TREŚCI

1 WSTĘP

Pojęcie analizy portali jest niezwykle szerokie. Niniejsza praca skupia się na analizie zawartości publicznie udostępnianych stron internetowych. Wyniki analiz są niezwykle przydatne pod względem wiedzy biznesowej. Na rynku istnieje niewiele narzędzi, które tego typu analizy przeprowadzają. W dalszej części pracy zostaną przedstawione niektóre z nich. Określone zostaną dane jakie są przez te narzędzia analizowane. Dodatkowo przedstawione zostaną wyniki analiz.

Istnieje wiele aplikacji, które utworzono w oparciu o zasady generyczności. Podstawowym ich założeniem jest współdziałanie głównej aplikacji oraz modułów rozszerzających jej funkcjonalność. Scharakteryzowane zostaną podstawowe zasady, jakie obowiązują przy tworzeniu tego typu oprogramowania. Ponadto autor zaprezentuje przykładowe narzędzia, które są aktualnie dostępne na rynku.

System stworzony zgodnie z zasadami modułowości umożliwia programistom lub użytkownikom rozszerzanie funkcjonalności aplikacji bez ingerencji w główny kod aplikacji. System powinien w łatwy sposób umożliwiać wykorzystanie stworzonych rozszerzeń. Zasada działania takiej aplikacji polega na wydzieleniu głównego programu, którego zadaniem jest wczytywanie zewnętrznych modułów, ich obsługa, instalacja, wyświetlanie dostępnych funkcji oraz obsługa oferowanej przez moduł funkcjonalności. Aplikacja może posiadać, albo system do zarządzania dodatkowymi wtyczkami, albo jasno określone zasady dodawania tych wtyczek, w ten sposób, aby można było je obsłużyć.

1.1 Cel pracy

Bezpośrednim celem pracy jest opracowanie koncepcji analizy portali internetowych przy wykorzystaniu bibliotek i technologii informatycznych pozwalających na jej implementację.

Pośrednim celem jej stworzenie prototypu, który obrazuje przedstawione rozwiązania. Stworzona aplikacja charakteryzować się będzie modułowością i możliwością rozszerzania swojej funkcjonalności poprzez instalowanie nowych wtyczek.

W ramach pracy stworzono wtyczki obrazujące działanie przyjętych rozwiązań:

- Pluginy pierwszego poziomu
    - Plugin obsługi portalu Allegro.pl
    - Plugin obsługi portalu Ceneo.pl
- Pluginy drugiego poziomu
    - Plugin obliczania średniej ceny produktów
    - Plugin obliczania mediany ceny produktów
    - Plugin wyświetlający najtańsze produkty
    - Plugin wyświetlający najdroższe produkty
    - Plugin rysujący wykres histogram rozkładu cen
    - Plugin rysujący wykres „Ramka-Wąsy” prezentujący medianę, kwartale i rozstęp

1.2 Rozwiązanie przyjęte w pracy

Stworzony prototyp jest aplikacją webową, która została stworzona przy wykorzystaniu narzędzi wchodzących w skład technologii JavaEE. Dodatkowo wykorzystano inne darmowe biblioteki, które ułatwiły proces implementacyjny. W skład tych bibliotek możemy wliczyć:

- Implementację standardu JSTL firmy Sun Microsystems Inc.
- Biblioteka JSPF udostępniona na licencji BSD przez German Research Center for Artificial Intelligence
- Biblioteka HttpClient firmy Apache

- Biblioteka JFreeCharts tworzonego w ramach jednego z projektów JFree.org
- Biblioteka CyberNeko HTML Parser rozprowadzania na licencji Apache 2.0

1.3 Rezultaty pracy

Bezpośrednim wynikiem pracy będzie opracowanie praktyk przy tworzeniu generycznych systemów analitycznych. Zaprezentowane zostaną technologie i biblioteki przydatne do implementacji przyjętych założeń. Przedstawione zostaną sposoby na optymalizację procesów analitycznych.

Pośrednim wynikiem pracy będzie działająca aplikacja webowa obrazująca działanie założonych rozwiązań. Do aplikacji dołączone zostaną dodatkowe pluginy pobierające dane z dwóch portali internetowych oraz pluginy, które dane będą analizować.

1.4 Organizacja pracy

Praca zaczyna się od przedstawienia problemu tworzenia generycznych systemów. W następnym rozdziale zostaną przedstawione istniejące na rynku rozwiązania ich wady i zalety.

Kolejny trzeci rozdział zawierał będzie informacje na temat narzędzi i technologii informatycznych, które zostały wykorzystane przy implementacji prototypu aplikacji.

W rozdziale czwartym zostaną przedstawione założenia jakie powinny zostać spełnione przy realizacji omawianego rozwiązania. Autor skupi się na scharakteryzowaniu funkcjonalności jakie należałoby uwzględnić podczas projektowania tego typu oprogramowania.

Rozdział piąty ma na celu zobrazowanie procesu implementacyjnego przyjętych założeń w prototypie aplikacji.

2 STAN SZTUKI

Analizy portali internetowych można dokonać na wiele sposobów. Zależy to od potrzeb użytkownika i jego spojrzenia na przedmiot zainteresowania. W niniejszym rozdziale autor przedstawi funkcjonujące na rynku narzędzia służące analizie portali internetowych. Zwrócić chce on szczególną uwagę na charakter analizy. Część istniejących produktów bada portale pod kątem kwestii technicznych. W rozdziale drugim autor poświęci szczególną uwagę analizatorom o charakterze biznesowym.

2.1 System analizy Fortuna1

System Fortuna1 stworzony i dystrybuowany jest przez firmę P8 z Krakowa. Adres strony, na której firma opisuje system, w chwili pisania pracy to *http:// fortuna1.pl.* Oferowany przez firmę system jest rozwiązaniem komercyjnym. W kolejnej części pracy zaprezentowane zostaną funkcje systemu, rozwiązania przyjęte przez twórców oraz przykładowe wyniki. Wszystkie wykorzystane tu dane zaczerpnięte zostały z publicznie dostępnych informacji zawartych na stronie firmy.

Omawiana aplikacja skupia się na analizie danych z jednego z najpopularniejszych serwisów aukcyjnych w Polsce Allegro.pl. Aplikacja zbiera informacje na temat zakończonych aukcji od roku 2008 włącznie. Informacje o aukcjach zawierają takie dane jak cena oferowanego produktu, datę wystawienia, datę zakończenia, ilość ofert i inne.

Serwis do głównych funkcji systemu zalicza następujące elementy:

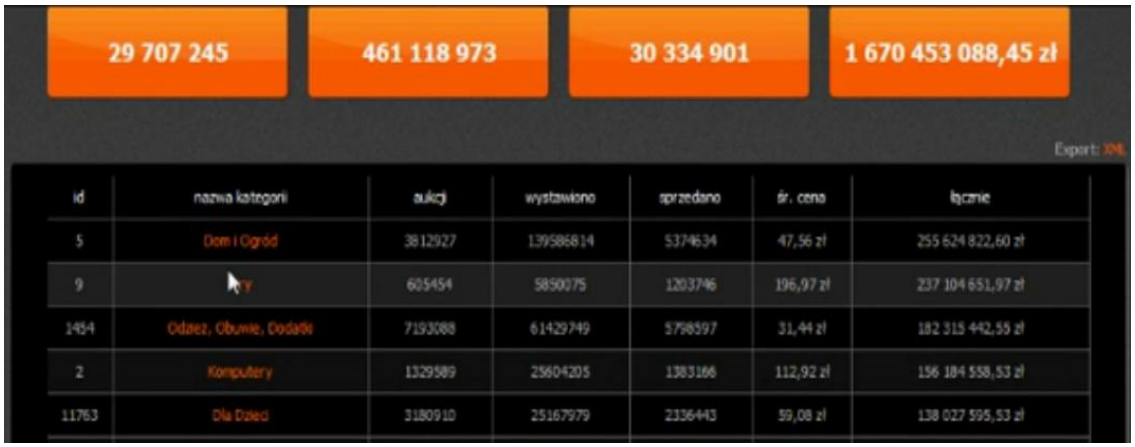
- Analiza sprzedaży
- Analiza kategorii
- Analiza kategorii w czasie
- Analiza sprzedawców

Każda z przedstawionych funkcji w różny sposób podchodzi do tematu analizy. W trakcie analizy sprzedaży, głównym elementem, nazywanym produktem, jest aukcja. Wyświetlane są informacje na temat ilości wystawionych produktów, ilości ofert kupna, średniej cenie, wartości sprzedaży.

Analizą kategorii nazywa się informacje uzyskane w drodze badania danej kategorii pod względem ilościowym, pojemnościowym, popytu, podarzy, dynamiki branży, wartości i ilości sprzedaży w określonym czasie, oraz atrakcyjności badanej kategorii.

Analiza kategorii w czasie ma za zadanie przedstawienie, w formie graficznej, wyników analizy dynamiki i wolumenu sprzedaży.

Analiza sprzedawców pozwala na wyszukiwanie użytkowników o największym współczynniku sprzedaży, produktów o największym popycie, czy też kategorii gdzie generowany jest najwyższy obrót.



Rysunek 1. Wyniki analizy kategorii

Źródło: fortuna1.pl

Wyniki analiz systemu oferowanego przez firmę mogą być przydatne do opracowywania różnego rodzaju strategii sprzedaży. Analizy wykorzystują dane historyczne zatem nie odzwierciedlają, aktualnie panującej sytuacji na rynku, ale dają obraz jak rynek zmieniał się w czasie. Taka informacja jest cenna i może być użyteczna w dziedzinie sprzedaży internetowej.

Do wad systemu możemy zaliczyć fakt, że dane zebrane przez system nie są dokładne. Firma podaje, że zebrane informacje o aukcjach, które odbyły się przed marcem 2009, nie były dokładne i mogą się różnić od faktycznych. Mowa tu głównie o liczbie sprzedanych produktów w ramach jednej aukcji. Założono, że każda złożona oferta kupna produktu dotyczy jednej sztuki. Jak się później okazało było to złe założenie, które wyeliminowano w marcu 2009 roku. Nie podjęto jednak decyzji o usunięciu błędnie zebranych danych, także wyniki analiz mogą wnosić pewne nieścisłości. Z informacji uzyskanej ze strony

producenta wynika, że dane o sprzedaży z wcześniejszych lat mogą być zaniżone o 10-15%.

2.2 System Manubia

System Manubia to komercyjne rozwiązanie do analizy portalu Allegro.pl. W chwili pisania pracy adres strony internetowej firmy, która oferuje oprogramowanie, to http://manubia.pl. W dalszej części podrozdziału autor skupi się na podstawowych możliwościach systemu i korzyściach dla klienta.

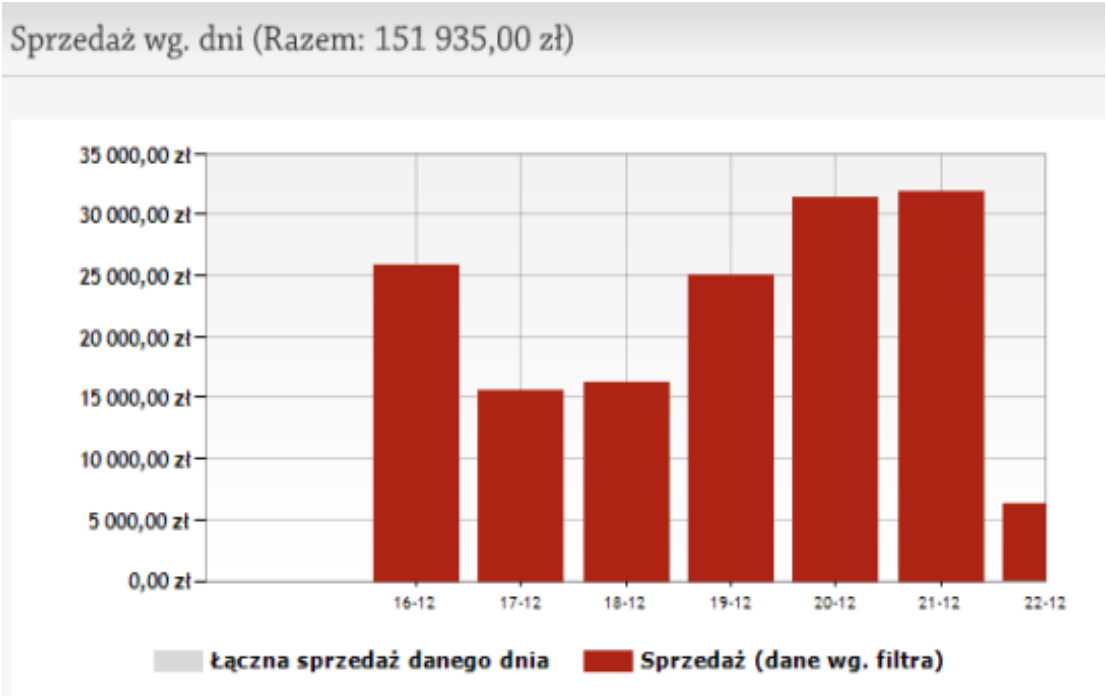
Rysunek 2. Raport sprzedaży systemu Manubia

Źródło: http://manubia.pl

Raporty oferowane przez system przedstawiają wyniki sprzedaży użytkowników portalu Allegro.pl. Po zarejestrowaniu i wykupieniu odpowiedniego abonamentu klient wybiera sprzedawcę, który będzie analizowany. Gdy tego dokona system zaczyna zbierać informacje na temat sprzedaży wybranego użytkownika. Dane są gromadzone i

magazynowane na serwerze. Klient po stwierdzeniu, że system zebrał odpowiednią porcję informacji ma możliwość wyświetlenia raportu.

Raport wygenerowany przez system zawiera informacje o ilości oraz wartości sprzedaży analizowanego użytkownika Allegro. W wynikach znajdziemy także informacje o produktach, które cieszą się największą popularnością oraz jak przedstawiają się wyniki sprzedaży w czasie. Do innych możliwości systemu można zaliczyć ukierunkowanie generowanych raportów. Manubia generuje raporty według czasu, kategorii produktów czy też słów kluczowych użytych przez sprzedawcę.



Rysunek 3. Raport sprzedaży systemu Manubia

Źródło: http://manubia.pl

Manubia jest rozwiązaniem płatnym. W czasie pisania pracy najtańsza wersja abonamentu to 59zł miesięcznie. Od wykupionego pakietu zależy czas archiwizowania danych, ilość sprzedawców jakich można dodać do systemu, możliwość generowania raportu zbiorczego oraz dziennych i tygodniowych raportów wysyłanych drogą elektroniczną.

Przedstawiony system może być źródłem cennych informacji. Raporty w jasny i prosty sposób dają obraz sprzedawców działających na portalu Allegro. Niejasna jest kwestia legalności zbierania danych osobowych przez omawiany system. Firma nie udostępnia informacji na temat legalności wygenerowanych raportów oraz celów do jakich możemy je

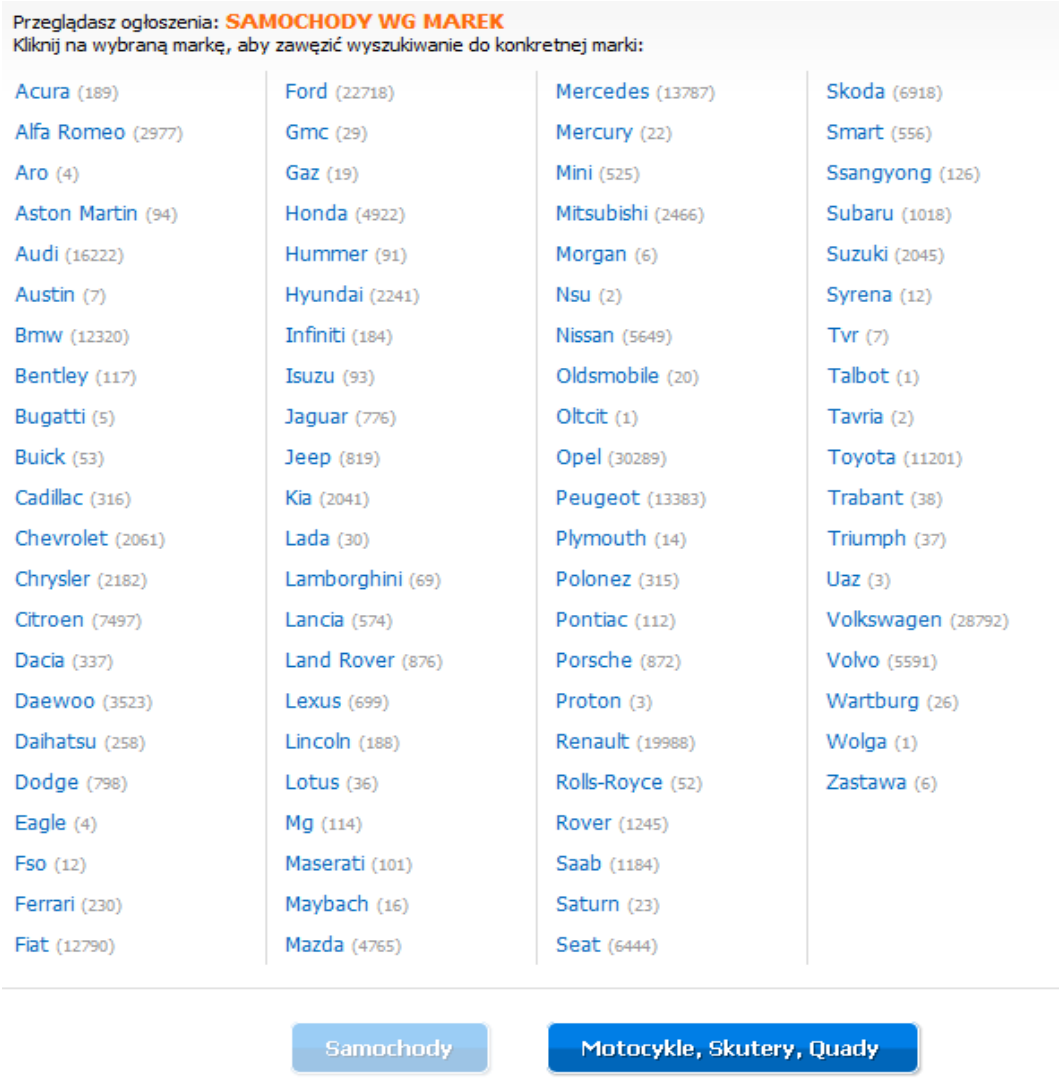
wykorzystać. Autorzy serwisu powinni w jasny sposób przedstawić sytuację prawną Manubia. Do tego czasu komercyjne wykorzystywanie wyników analizy systemu może być ryzykowne.

2.3 Serwis wroom.pl

Serwis określany przez media jako pierwsza na polskim rynku wyszukiwarka ogłoszeń motoryzacyjnych. Sposób działania serwisu porównywany jest do jednej z najbardziej popularnych wyszukiwarek internetowych Google.pl.

Firma, która jest właścicielem serwisu planuje rozszerzyć swoją działalność na cały świat. Systematycznie dodawane mają być nowe serwisy ogłoszeniowe z Polski, Europy, USA i Kanady.

Działanie systemu, polega na zbieraniu informacji o ogłoszeniach motoryzacyjnych za pomocą robota internetowego. Robot przeszukuje z góry określone portale internetowe i zapisuje wyniki na serwerach serwisu wroom.pl. Dzięki z góry określonej liście portali internetowych, możliwe jest zebranie szczegółowych informacji na temat każdego produktu.



Rysunek 4 Kategorie serwisu wroom.pl

Źródło: http://wroom.pl

Serwis daje możliwość przeglądania zebranych ogłoszeń. Do tego celu możemy użyć wyszukiwarki standardowej lub zaawansowanej. Możemy również przeglądać ogłoszenia klikając w kategorie wyświetlane na stronie.

Do zalet systemu można zaliczyć automatyczne tłumaczenie ogłoszeń zagranicznych. Przeliczane są również ceny produktów z obcych walut na polską. W trakcie wyszukiwania podawane są również średnie ceny odnalezionych produktów.

Serwis posiada rozbudowany moduł reklamy modułowej w wynikach wyszukiwania. Reklamodawcy mogą określić dla jakich użytkowników przeznaczone są ich reklamy i śledzić wyniki osiągane przez kampanię.

Usługi oferowane przez serwis wroom.pl są nowością na rynku. Pozwalają na uzyskanie jasnej informacji dotyczącej określonej grupy konsumenckiej. Do wad systemu można zaliczyć ograniczenie się do jednego segmentu produktów motoryzacyjnych. Mimo to istnieje duże prawdopodobieństwo, że system osiągnie sukces. Firma, w momencie pisania pracy dostała duży zastrzyk finansowy od zewnętrznego inwestora, co pozwoli jej na dalszy rozwój.

2.4 Strona websiteoptimization.com

Strona prowadzona jest przez firmę Website Optimalization, LLC. Jest to prywatna firma, która zajmuje się konsultingiem w sprawach optymalizacyjnych i marketingowych witryn internetowych. Udostępnia ona narzędzia do analizy stron internetowych.

Przestawienie rozwiązań oferowanych przez firmę ma na celu ukazanie technicznego podejścia do procesu analizy. Zebrane informacje dadzą ogólny zarys procesu analizy portali internetowych.

Do głównych narzędzi analitycznych jakie oferuje serwis zaliczamy:

- Analiza i optymalizacja stron internetowych
- Analiza szybkości
- Optymalizacja grafiki
- Optymalizacja szybkości
- Narzędzia marketingu w wyszukiwarkach internetowych
- Analiza współczynnika konwersji
- Analiza aktualnie prowadzonej kampanii marketingowej
- Optymalizacja pod względem różnych silników wyszukiwarek
- Rozwój, przeprojektowywanie, planowanie witryn internetowych
- Analiza zawartości pod względem językowym

Wymienione narzędzia pozwalają na szeroką analizę strony internetowej skupiając się w znacznej mierze na optymalizacji zawartości. Narzędzie Web Page Analyzer, analizujące szybkość wczytywania strony, zostało udostępnione w wersji darmowej.

Analiza szybkości rozpoczyna się od wprowadzenia adresu witryny. Po przesłaniu formularza i przepisaniu kodu zabezpieczającego captcha wyświetlane zostają wyniki analiz. Zawierają one wiele przydatnych informacji na temat strony.

Po wpisaniu adresu portalu Allegro.pl, wyświetlone zostają tabele zawierające informacje o wydajności omawianej strony internetowej. Tabela [Tabela 1] przedstawia podstawowe właściwości wczytywanej witryny. Określa rozmiar, w bajtach (bytes), poszczególnych części strony, a także czas, jaki potrzebny jest do wczytana tych części przy określonym łączu internetowym.

Object type	Size (bytes)	Download @ 56K (seconds)	Download @ T1 (seconds)
HTML:	9397	2.07	0.25
HTML Images:	106959	22.32	1.57
CSS Images:	92019	22.54	4.69
Total Images:	198978	44.86	6.26
Javascript:	40487	9.47	1.61
CSS:	4345	1.07	0.22
Multimedia:	0	0.00	0.00
Other:	0	0.00	0.00

Tabela 1. Object Size Totals

Źródło: websiteoptimization.com

Kolejna tabela [Tabela 2] przedstawia informacje na temat liczebności poszczególnych elementów witryny, takich jak liczba wczytanych obrazów, obrazów wczytanych za pomocą styli css, skryptów, importu plików styli css i innych.

External Object	QTY
Total HTML:	1
Total HTML Images:	5
Total CSS Images:	21
Total Images:	26

Total Scripts:	7
Total CSS imports:	1
Total Frames:	0
Total Iframes:	0

Tabela 2. External Objects

Źródło: websiteoptimization.com

Tabela [Tabela 3] zawiera informacje na temat czasu jaki potrzebny jest na wczytanie omawianej strony internetowej. Strona zawiera informacje o sposobie obliczania czasu podanego w tabeli. Podstawowe testy wykonywane są przy prędkościach ISDN i T1. Pozostałe wyniki są korygowane o współczynnik utraty pakietów. W tym przypadku o podaną wartość 0,7. Notatka wyjaśniająca w jaki sposób obliczany jest ten współczynnik podaje, że opóźnienie w obie strony dla każdego obiektu wynoszące 0,2 sekundy dla 35 obiektów daje łączny czas opóźnienia 7 sekund. Nie są brane pod uwagę opóźnienia wynikające z przetwarzania i wyświetlania XHML'a.

Connection Rate	Download Time
14.4K	203.25 seconds
28.8K	105.12 seconds
33.6K	91.11 seconds
56K	57.46 seconds
ISDN 128K	22.45 seconds
T1 1.44Mbps	8.34 seconds

Tabela 3. Download Times

Źródło: websiteoptimization.com

Tabela [Tabela 4] jest częścią tej wyświetlanej na stronie analizatora. Przedstawia ona informacje na temat obiektów jakie wykryte zostały podczas wczytywania witryny. Charakteryzuje każdy obiekt z osobna podając ilość wystąpień, rozmiar, typ ,oraz adres obiektu. W ostatniej kolumnie wyświetlane są komentarze, opisujące sugestie lub dodatkowe informacje na temat wczytanego obiektu.

QTY	SIZE#	TYPE	URL	COMMENTS
1	61320	IMG	http://staticpics.allegrostatic.pl/public/showcase/show_pics_21684	Header size = 275 bytes
1	24606	SCRIPT	http://allegro.pl/js/jquery-1.4.2.min.js	Header size = 308 bytes Congratulations! This file was compressed
1	23353	IMG	http://staticpics.allegrostatic.pl/public/showcase/show_pics_20603	Header size = 275 bytes
1	21368	IMG	http://staticpics.allegrostatic.pl/public/showcase/show_pics_21746	Header size = 275 bytes
1	16624	CSS IMG	http://static.allegrostatic.pl/site_images/1/0/layout/first-time-xmas.gif	Header size = 331 bytes
…				

Tabela 4. Page Objects

Źródło: websiteoptimization.com

Ostatnią częścią rezultatów są wyniki analizy i rekomendacje. W tej części analizy wyliczane są poszczególne elementy strony internetowej. Każdy wyświetlany element jest oznaczony odpowiednim kolorem odzwierciedlającym pozytywną bądź negatywną ocenę. Obok oceny, w formie kilku krótkich zdań, przestawione są rekomendacje bądź informacje o pozytywnej ocenie badanego elementu.

Analizie i ocenie poddane zostały elementy:

- TOTAL_HTML – liczba plików *.html wczytanych do dokumentu

- TOTAL_OBJECTS – liczba wszystkich obiektów w dokumencie
- TOTAL_IMAGES – liczba wczytanych obrazów
- TOTAL_CSS – liczba plików arkuszy styli wczytanych z zewnątrz
- TOTAL_SIZE – całkowity rozmiar dokumentu podany w bajtach
- TOTAL_SCRIPT – łączna liczba plików skryptów wczytanych z zewnątrz
- HTML_SIZE – rozmiar aktualnie wczytanego pliku html
- IMAGES_SIZE – łączny rozmiar obrazów wczytanych do dokumentu
- SCRIPT_SIZE – łączny rozmiar skryptów wczytanych z zewnątrz
- CSS_SIZE – łączny rozmiar arkuszy styli wczytanych z zewnątrz
- MULTIM_SIZE - łączny rozmiar zasobów multimedialnych

Analizy dokonywane przez narzędzie udostępnione przez stronę websiteoptimization.com skupiają się na wydajności , rozmiarach i czasie ładowania się analizowanej witryny. Wyniki pomogą podjąć decyzje o zmianach jakie warto wprowadzić, aby witryna była bardziej wydajna, wczytywała się szybciej i była lepiej oceniana przez użytkowników.

2.5 Podsumowanie

Istnieje wiele rozwiązań, które w różny sposób podchodzą do kwestii analizy portali internetowych. Autorowi pracy nie udało się znaleźć oprogramowania, które zajmowało się zarówno analizą portali jak i możliwością rozbudowywania funkcjonalności poprzez dodawanie nowych modułów.

Autor przedstawił istniejące rozwiązania analizujące portale internetowe. Omówione systemy w różny sposób podchodziły do kwestii analizy. Przedstawiono czego dotyczyły analizy i jakie otrzymywano wyniki. Pokazano zalety i wady działania tych systemów. Starano się określić czemu takie analizy mogłyby służyć i czy omawiane systemy spełniają przyjęte założenia.

3 NARZĘDZIA UŻYTE W PRACY

Rozdział ten zawiera informacje na temat narzędzi wykorzystywanych w trakcie projektowania i implementacji generycznego systemu do analizy portali internetowych. W głównej mierze skupiono się na takich narzędziach jak:

- Eclipse SDE
- Serwer Tomcat 7
- Technologia JavaEE 6
    - Enterprise JavaBeans
    - JavaServer Faces
    - Java Servlet
    - JavaServer Pages
    - Java Persistence API
- Standard JSTL
- Darmowe biblioteki, które przyśpieszyły proces implementacji systemu.

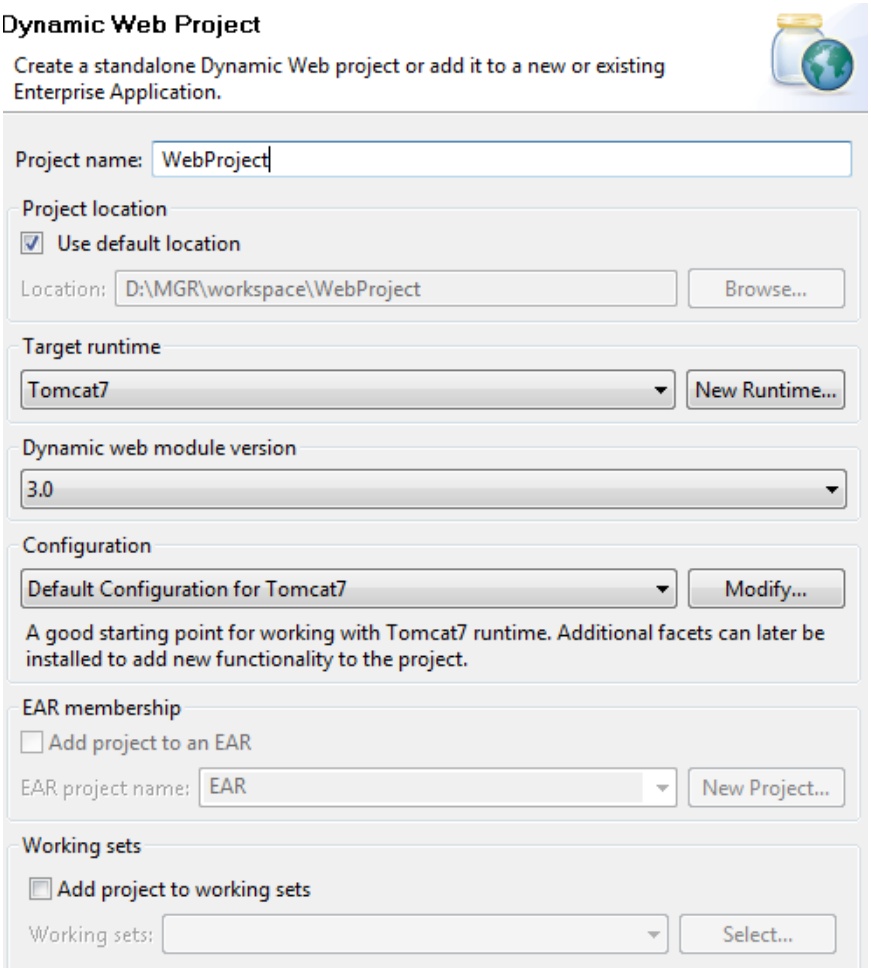
3.1 Eclipse SDE

Eclipse SDE (Software Development Environment), dalej nazywane Eclipse (patrz [3]), to środowisko, w skład którego wchodzi IDE (Integrated Development Environment), czyli zintegrowane środowisko programistyczne oraz system pluginów rozszerzający funkcjonalność aplikacji. Środowisko tworzone i udostępniane jest w ramach projektu Eclipse. Do głównych zadań aplikacji należy ułatwienie procesu tworzenia i rozwijania oprogramowania poprzez udostępnienie odpowiednich narzędzi.

W chwili pisania pracy Eclipse jest jednym z najbardziej popularnych środowisk do tworzenia aplikacji. Za jego pomocą tworzone są aplikacje konsolowe, graficzne, webowe, dedykowane na urządzenia mobilne i wiele innych. Pluginy rozszerzające funkcjonalność dodają możliwości takie jak zarządzanie serwerami, tworzenie dokumentacji, obsługa dodatkowych języków i inne. Eclipse w znacznej mierze został napisany w języku Java. Podstawowa instalacja umożliwia tworzenie oprogramowania w języku Java. Poprzez

dodatkowe pluginy możliwa jest praca z językami Ada, C, C++, COBOL, Perl, PHP, Ruby, Scala czy Scheme.

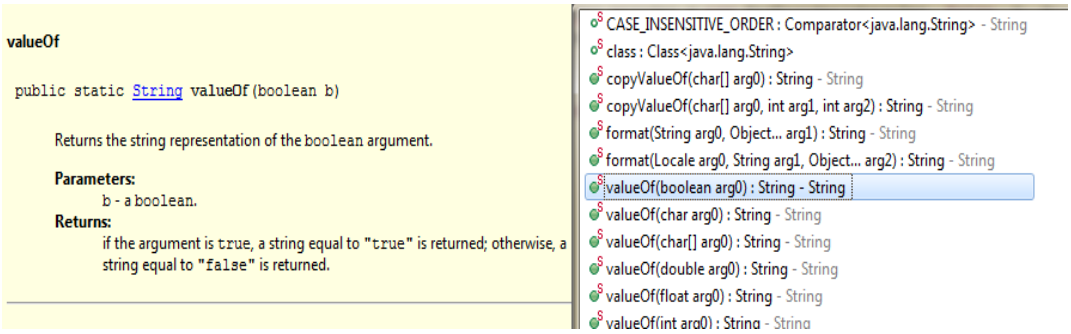
Do implementacji systemu, który jest tematem tej pracy, wykorzystano zestaw pluginów do tworzenia aplikacji webowych. Zestaw umożliwia dodawanie serwerów i zarządzanie nimi, tworzenie klas i serwletów w języku Java, tworzenie plików konfiguracyjnych i wdrożeniowych. Przy tworzeniu nowych projektów, jest możliwość użycia generatora, co znacznie ułatwia rozpoczęcie prac. Po uruchomieniu kreatora [Rysunek 5] mamy możliwość wyboru podstawowych parametrów projektu takich jak środowisko uruchomieniowe, wersja wykorzystanego standardu „Dynamic Web module", oraz szablonu pliku konfiguracyjnego. Możliwe jest też dokonanie decyzji czy nasz projekt ma być dodany do pliku EAR. Ponadto możemy umieścić swój projekt w tzw. Working Set, czyli zbiorze, który niesie dodatkowe korzyści w przypadku pracy nad rozbudowanymi projektami. Po wyborze nazwy projektu możemy zakończyć klikając Finish. Projekt zostanie wygenerowany, stworzona zostanie struktura katalogów. Pozwala to na rozpoczęcie pracy nad kolejnymi częściami projektu.



Rysunek 5. Kreator nowego dynamicznego projektu webowego

Źródło: Opracowanie własne

W Eclipse większość funkcji obsługiwanych jest za pomocą pluginów. Jednym z nich jest systemem intellisense ułatwiający pisanie aplikacji poprzez podpowiadanie składni. Intellisense dla języka Java nie ogranicza się jedynie do podpowiedzi, które z góry są określone, wręcz przeciwnie, dynamicznie wczytuje stworzone klasy i podpowiada składnie dla obiektów z całego projektu, czy też bibliotek które są dołączone do projektu z zewnątrz. Dzięki takiemu podejściu czas wytwarzania oprogramowania ulega znacznemu skróceniu. Programista nie musi przeglądać dokumentacji w celu przypomnienia sobie nazw metod, które mogą zostać wykorzystane. Dodatkową cechą systemu podpowiedzi jest wyświetlanie informacji na temat aktualnie przeglądanej metody.



Rysunek 6. Działanie systemu Intellisense dla języka Java

Źródło: Opracowanie własne

Edytor Eclipse umożliwia kolorowanie składni, co poprawia czytelność tworzonego kodu. Jest także możliwość grupowania tzw. folding części kodu w celu uzyskania większej przejrzystości.

Podsumowując, Eclipse to potężne narzędzie do profesjonalnego wytwarzania oprogramowania. Udostępnia szereg podstawowych funkcji, które sprawiają, że programowanie staje się łatwiejsze i wydajniejsze. W sytuacji gdy użytkownikowi zabraknie potrzebnego rozwiązania, może odnaleźć je wśród wielu rozszerzeń udostępnianych przez szereg repozytoriów

3.2 Tomcat 7

Tomcat jest kontenerem aplikacji webowych. Stworzony i udostępniany w ramach projektu Apache. Umożliwia uruchamianie oprogramowania tworzonego w technologii Java Servlets oraz JSP. Apache w wersji 7 implementuje Serwlety 3.0 oraz specyfikację Java Server Pages w wersji 2.2.

Siódma wersja serwera (patrz [5]) wykorzystuje kompilator Eclipse JDT do kompilowania stron JSP, w związku z tym do uruchomienia serwera nie jest niezbędnym pobieranie pakietów developerskich Javy. Za pomocą plików konfiguracyjnych można zmienić kompilator Javy na każdy, który obsługuje technologię Apache Ant.

Standardowa instalacja kontenera udostępnia interfejsy:

- annotations-api.jar
- catalina.jar
- catalina-ant.jar
- catalina-ha.jar
- catalina-tribes.jar
- el-api.jar
- jasper.jar
- jasper-el.jar
- ecj-3.6.jar
- jsp-api.jar
- servlet-api.jar
- tomcat-api.jar
- tomcat-coyote.jar
- tomcat-dbcp.jar

Aby wymienione interfejsy mogły być użyte, należy umieścić adres folderu, w którym znajdują się biblioteki w zmiennej classpath projektu. Zmienna określa miejsca, z których wczytywane są wszystkie klasy potrzebne do uruchomienia projektu.

W powyższej liście interfejsów wyróżniamy kilka podstawowych, które wymagane są do tworzenia aplikacji webowych. Biblioteka *jsp-api.jar* udostępnia szereg klas do obsługi Java Server Pages. Gdy znajdzie się w zasięgu klas projektu, możemy używać plików, które zawierają skrypty zgodne ze specyfikacją JSP.

Biblioteka *servlet-api.jar* zawiera klasy potrzebne to tworzenia serwletów. Serwlet jest obiektem klasy języka Java działającym po stronie serwera WWW na zasadzie żądanie-odpowiedź czym rozszerza jego funkcjonalność. Kwestia serwletów zostanie szczegółowo opisana w dalszej części pracy. Po tym jak klasy biblioteki znajdą się w zasięgu klas projektu, możliwe jest importowanie i wykorzystywanie udostępnionych interfejsów, klas abstrakcyjnych oraz zwykłych do tworzenia własnych implementacji serwletów.

Możliwość korzystania z adnotacji uzyskiwana jest po wczytaniu pakietu annotation-api.jar. Biblioteka udostępnia adnotacje, które zostały przygotowane przez deweloperów projektu Tomcat. Ułatwiają one pracę przez automatyczne generowanie często stosowanych części kodu.

Biblioteki jasper.jar, jasper-el.jar oraz el-api.jar zawierają implementację języka wyrażeń EL (ang. Expression Language). Język i jego właściwości zostaną przedstawione w dalszej części pracy w podrozdziale opisującym standard JSTL.

3.3 Technologia JavaEE 6

Java Platform, Enterprise Edition (patrz [6]) dalej zwana JavaEE jest zbiorem narzędzi i technologii do tworzenia oprogramowania zorientowanego na usługi biznesowe. Platforma oparta została na języku Java oraz technologii Java 2 Standard Edition. Zbiór technologii w zupełności wystarcza do pisania aplikacji, o których mowa była na początku podrozdziału. Mimo to wiele firm tworzy dodatkowe frameworki, takie jak Spring, które rozszerzają możliwości technologii.

Do głównych technologii wchodzących w skład platformy można zaliczyć:

- Enterprise JavaBeans
- JavaServer Faces
- Java Servlet
- JavaServer Pages

- Java Persistence API

3.3.1 Enterprise JavaBeans

Technologia EJB opiera się na idei tworzenia komponentów, zwanych bean'ami, które spełniają, określone przez specyfikację reguły. Bean'y charakteryzują się takimi cechami jak trwałość, bezpieczeństwo, transakcyjność, rozproszenie, wielodostęp (patrz [2]). Działanie komponentów polega na osadzaniu ich na serwerze aplikacji, gdzie udostępniają one swoją funkcjonalność lokalnie lub zdalnie za pomocą protokołu RMI.

Rozróżniamy 3 rodzaje komponentów:

- Komponenty encyjne
- Komponenty sesyjne
- Komponenty sterowane komunikatami

Każdy z rodzajów ziaren spełnia inną funkcję w aplikacji. Ziarna encyjne wykorzystywane są do obiektowego reprezentowania danych. Ziarna sesyjne tworzą logikę aplikacji. Ziarna sterowane komunikatami wykorzystuje się przy przetwarzaniu asynchronicznym lub przy modelu opartym o zasadę klient-dostawca.

3.3.2 JavaServer Faces

Technologia JavaServer Faces to specyfikacja tworzenia graficznych interfejsów użytkownika po stronie serwera. Dzięki implementacji interfejsów JSF, przez wiele narzędzi do tworzenia oprogramowania, ułatwiono tworzenie aplikacji internetowych. W skład JSF wchodzi zbiór interfejsów (ang. APIs), które reprezentują komponenty interfejsu użytkownika (ang. UI), umożliwiają zarządzanie ich stanem, obsługą zdarzeń (ang. events handling), walidacją oraz nawigacją. Zadaniem APIs jest też zapewnienie wsparcia dla wielojęzyczności i dostępności. JSF zawiera także implementację niestandardowych tagów, które można wykorzystać w skryptach JSP.

3.3.3 JavaServlet

Serwlet to klasa, która rozszerza możliwości serwera. Implementuje ona interfejs Java Servlet. Interfejs serwletu określa metody jakich należy użyć, aby klasa go implementująca mogła zostać serwletem. W JEE istnieją już klasy z wdrożonym interfejsem serwletu. Pierwszą z nich jest GenericServlet, która daje dostęp do parametrów i ustawień serwletu. Jest podstawą dla kolejnej klasy HttpServlet. Klasa HttpServlet poza metodami wdrożonymi wcześniej udostępnia metody do obsługi żądań protokołu http. Najważniejsze z nich to doGet oraz doPost. Poza nimi istnieją jeszcze metody doPut, doHead, doOptions, doTrace, doDelete, doConnect.

Nagłówek metody doGet() wygląda następująco:

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException
```

Metoda doGet uruchamiana jest w przypadku gdy użytkownik wywoła bezpośrednio serwlet wpisując jego adres w przeglądarce. Parametry tej metody to request i response, czyli obiekty klas reprezentujących wywołanie i odpowiedź przeglądarki. Przechowują opcje i parametry jakie przekazywane są pomiędzy serwerem a klientem. Nagłówek metody doPost jest taki sam jak metody opisanej wcześniej, różni się jedynie tym, że wywoływana jest po przesłaniu przez użytkownika żądania POST.

Podczas wdrażania napisanego serwletu i uruchamiania serwera serwlet nie wykonuje żadnych operacji. Dopiero w momencie wywołania go, przez użytkownika aplikacji, klasa ładowana jest przez serwer, tworzona jest instancja serwletu i wywoływana na nim jest metoda init() oraz service(). Dopiero wywołanie metody service() rozpoczyna obsługę żądań przez serwlet.

W trakcie wywołania jednej z metod zaczynających się od do*, mamy możliwość odczytania parametrów jakie zostały przesłane przez przeglądarkę. Do bardziej istotnych metod udostępnionych przez obiekt klasy HttpServletRequest, należą (patrz [1]):

- getParameter(String nazwa) – zwraca wartość parametru o podanej nazwie
- getParameterNames() – zwraca nazwy parametrów
- getCookies() – zwraca ciasteczka przechowywane na komputerze klienta
- getHeader(String nazwa) – zwraca nagłówek o podanej nazwie

- getQueryString() – zwraca parametry w formie łańcucha znaków
- getContextPath() – zwraca adres kontekstu aplikacji
- getServletPath() – zwraca adres dostępu do serwletu
- getRemoteUser() – zwraca dane zalogowanego usera bądź null w przypadku braku uwierzytelnienia

Po obsłużeniu zapytania serwlet przesyła do użytkownika wynik swojej pracy. Do tego celu wykorzystuje się obiekt klasy HttpServletResponse. Do podstawowych metod udostępnionych przez obiekt można zaliczyć:

- getOutputStream() – zwraca strumień zapisu binarnego
- getWriter() – zwraca obiekt zapisu tekstowego
- addCookie(Cookie cookie) – dodaje ciasteczko do ustawienia na komputerze klienta
- setStatus(int kod) – ustawia kod odpowiedzi
- sendError(int kod) – zwraca błąd o podanym w parametrze kodzie
- sendRedirect(String url) – przekierowuje na, podany jako parametr, adres url

3.3.4 JavaServer Pages

Technologia JSP umożliwia tworzenie dokumentów z wykorzystaniem HTML, XHTML, DHTML oraz XML. Dokumenty tego typu mogą zawierać zwykłe wyrażenia pisane w języku Java, ale także różnego rodzaju wyrażenia, obsługiwane przez serwer, takie jak EL (ang. Expression Language).

Pliki JSP można wykorzystać w różny sposób. Jedną z możliwości jest bezpośrednie odwołanie się do pliku przez podanie adresu pod jakim umieszczony jest on na serwerze. Jest to najprostszy sposób i nie wymaga żadnych dodatkowych czynności. Wywoływany plik zostanie automatycznie wczytany i przetworzony przez kontener, a wynik wysłany do przeglądarki.

Innym sposobem jest przekierowanie sterowania. Do tego celu wykorzystywany jest mechanizm o nazwie dyspozytor żądań. Za pomocą obiektu dyspozytora wywołujemy metodę *forward* i przekazujemy parametry *request* i *response.* Sterowanie przekazywane

jest do pliku JSP, gdzie na podstawie przesłanych parametrów generowany jest wynik zwracany do użytkownika.

Obiekt dyspozytora pozwala również na użycie metody *include*, która nie przekierowuje sterowania, ale wczytuje, przetwarza plik JSP i dołącza wynik do obiektu *response* zapytania.

Podsumowując, można powiedzieć technologia JSP została stworzona, aby oddzielić część prezentacji od części kodu, a jednocześnie ułatwić proces tworzenia interfejsów użytkownika. W późniejszych etapach działania aplikacji plik JSP jest przekształcany na serwlet, ale dzieje się to bez udziału programisty.

3.3.5 Java Persistence API

Java Persistence służy do mapowania obiektowo-relacyjnego. Standard stworzony i rozwijany przez firmę Sun Microsystems, która została wykupiona przez amerykańską firmę Oracle w roku 2010. Standard JPA to część standardu Enterprise JavaBeans w wersji 3.0.

Do głównych zadań standardu należy odzwierciedlanie danych zapisanych w relacyjnej bazie danych na język obiektowy. Obiekty utworzone w procesie translacji nazywa się encjami. Za pomocą adnotacji lub plików xml opisuje się jak dane relacyjne przekładają się na encje.

Do zapisu wyników operacji na danych używa się narzędzia o nazwie EntityManager. Za jego pomocą możemy dodawać i usuwać encje. Służy ono również do wyszukiwania danych przy użyciu klucza głównego (ang Primary key). Przy użyciu EntityManager'a możliwe jest tworzenie nowych zapytań, które przypominają składnią język zapytań SQL.

3.4 Standard JSTL

Standard JavaServer Pages Standard Tag Library (patrz [9]) rozszerza funkcjonalność standardu JSP, przez dodanie nowych wyrażeń nazywanych custom tags. Nowe wyrażenia sprawiają, że tworzenie skryptów widoku staje się łatwiejsze i efektywniejsze.

JSTL wnosi wiele pożytecznych narzędzi. Dodaje możliwość przetwarzania xml, stosowania wyrażeń warunkowych, pętli, obsługi międzynarodowości. Wszystko bez konieczności używania skrypletów w języku Java.

Aby rozpocząć pracę ze znacznikami JSTL należy dołączyć do projektu biblioteki z implementacją omawianego standardu. Biblioteki wykorzystane w prototypie omawianego systemu to:

- jstl-impl.jar
- jstl-api-1.2.jar

Do dyspozycji mamy pięć głównych bibliotek tagów:

- JSTL core
- JSTL fmt
- JSTL sql
- JSTL XML
- JSTL functions

JSTL core (patrz [1]) zawiera klasy do obsługi głównych znaczników JSTL. W skład core wchodzą takie znaczniki jak out, set, if, else, when, choose, forEach, url, import, param.

```
<c:forEach items=”${items}” var=”item”>
           <c:out var=”${item}” default=”NULL” />
</c:forEach>
```

Biblioteka fmt zawiera narzędzia do formatowania. W jej skład wchodzą takie funkcje jak setLocate, requestEncoding, timeZone, param, message, bundle, formatNumber, parseNumber, formatDate, parseDate.

```
<fmt:formatNumber type="number"
                  maxIntegerDigits="6"
                  maxFractionDigits="2
                  value="${num}" />
```

Kolejna biblioteka JSTL sql zawiera wyrażenia do obsługi zapytań do bazy danych. W jej skład wliczamy transaction, query, update, param, dateParam, setDataSource.

```
<sql:query var = "items" dataSource="${db}">
           select id, name, date, desc from items
</sql:query>
```

JSTL xml to narzędzia do przetwarzania xml. Możemy do nich zaliczyć taki funkcje jak choose, out, if, forEach, otherwise, param, parse, set, transform, when.

```
<x:parse var="idoc" xml="${items}" />
<x:out select=”$idoc/item/name/” />
```

Ostatnią biblioteką jest JSTL functions. Zawiera funkcje działające na Stringach. Wykorzystuje się do sprawdzania długości łańcucha znaków, usuwania zbędnych spacji, zamiany znaków i innych. Do ważniejszych funkcji można zaliczyć contains, endsWith, indexOf, join, length, replace, split, startsWith, substring, toLowerCase, toUpperCase, trim.

```
${fn:contains(“alalalala”,"ala")} <!-- true -->
${fn:toLowerCase(“AlaLalAla”)}    <!-- alalalala -->
```

Podsumowując standard JSTL w znacznym stopniu ułatwia i przyśpiesza proces tworzenia skryptu prezentacji. Pozwala na oddzielenie modelu aplikacji od jego widoku. Jego funkcjonalność w zupełności wystarcza do tego aby skrypt JSP nie zawierał żadnych wyrażeń z języka Java.

3.5 Java Simple Plugin Framework

Zewnętrzna biblioteka udostępniona na licencji BSD przez German Research Center for Artificial Intelligence. JSPF (patrz [4]) jest prostym frameworkiem do tworzenia i zarządzania pluginami. Redukuje czas tworzenia zewnętrznych modułów, całkowicie ukrywa implementację, udostępnia interfejsy do własnej implementacji. Aby przyśpieszyć proces implementacji udostępnia szeroką gamę adnotacji:

- @PluginImplementation
- @InjectPlugin
- @PluginLoaded
- @Timer
- @Thread

PluginImplementation jest podstawową adnotacją . Dodajemy ją przed nazwą klasy, która implementuje interfejs Plugin. Sprawia, że klasa będzie załadowana w trakcie

działania aplikacji (ang. Run time). W przeciwnym wypadku nie będzie można jej użyć przez menagera pluginów.

```
@PluginImplementation
public class Allegro_Analizer  implements Plugin  {
…
}
```

InjectPlugin określa, że zmienna lub stała nią oznaczona powinna znajdować się w implementacji pluginu. W przypadku jej braku dodawana jest zmienna i inicjalizowana wartością null. Zmienna oznaczona adnotacją musi być publiczna.

```
@InjectPlugin
public String name;
```

PluginLoaded służy do oznaczania metody, która ma być wykonana po tym jak Plugin zostanie wczytany i załadowany. Oznaczona metoda musi być publiczna w innym przypadku zostanie wyrzucony wyjątek informujący o tym, że metoda nie została znaleziona.

```
@PluginLoaded
public void checkPlugin(Plugin plugin) {
 ...
}
```

Adnotacja Timer użyta przy metodzie powoduje, że po zarejestrowaniu pluginu, jest ona wywoływana w cyklach. Każdy cykl wywoływany jest po czasie przekazanym w parametrze adnotacji *period.*

```
@Timer(period = 1000)
public void getRunTime() {
   second++;
   System.out.println("second="+second);
}
```

Metoda oznaczona adnotacją Thread, wywoływana będzie w momencie utworzenia nowego wątku Plugin (patrz [3]).

```
@Thread
public void run() {
```

```
  ...
}
```

W skład frameworka wchodzi menager pluginów. To za jego pomocą dodajemy pluginy z określonej lokalizacji. Tworzymy go za pomocą fabryki PluginManagerFactory. Mając do dyspozycji obiekt menagera wywołujemy metodę addPluginsFrom, a w parametrze podajemy źródło z którego mają być wczytane pluginy.

```
PluginManager pm = PluginManagerFactory.createPluginManager();
pm.addPluginsFrom(new File("plugins/").toURI());
pm.addPluginsFrom(new File("plugins/plugin.jar").toURI());
pm.addPluginsFrom(new URI("classpath://*"));
pm.addPluginsFrom(new
URI("http://jspf.googlecode.com/files/coolplugin.jar"));
pm.addPluginsFrom(new ClassURI(PluginImpl.class).toURI());
```

Po załadowaniu pluginów z określonego miejsca umieszczane są one w obiekcie PluginManager'a. Aby uzyskać dostęp do jednego z pluginów trzeba wywołać metodę zwracającą instancję pluginu na obiekcie managera.

```
Plugin plugin = pm.getPlugin(Plugin.class);
```

Można również, za pomocą obiektu klasy *PluginManagerUtil* zwrócić wszystkie pluginy jakie zostały dodane do menagera. W ten sposób otrzymamy kolekcję pluginów.

```
Collection<Plugin> pluginCollection;
PluginManagerUtil pmu = new PluginManagerUtil(pm);
pluginCollection = pmu.getPlugins(Plugin.class);
```

Framework pozwala na zdalne wczytywanie pluginów. Możliwe jest także cache'owanie, dzięki temu po pierwszym uruchomieniu menagera i wczytaniu pluginów każde kolejne wywoływane jest znacznie szybsze.

Podsumowując JSPF to wygodne narzędzie, które w prosty sposób pozwoli rozpocząć tworzenie aplikacji opartych na modułowości. Posiada przyjazny menager, który wczyta napisane pluginy z wielu lokalizacji bez konieczności tworzenia obszernych plików konfiguracyjnych.

3.6 Apache Commons

Projekt Apache Commons (patrz [7]) tworzony jest przez firmę Apache. Skupia się na dostarczeniu komponentów, które są często stosowane przy tworzeniu oprogramowania. Apache zapewnia, że udostępniane narzędzia mają minimalną zależność od innych bibliotek. Dzięki temu wdrażanie ich nie powinno sprawić żadnych kłopotów. Twórcy dbają również o to by ich moduły były na tyle kompatybilne wstecz na ile jest to możliwe. Także programiści korzystający z omawianych narzędzi nie muszą obawiać się, że po aktualizacji ich oprogramowanie przestanie działać.

W trakcie tworzenia prototypu wykorzystano biblioteki z pakietu Commons:

- apache.commons.logging
- apache.commons.httpclient
- apache.commons.codec

Pakiet logging zawiera narzędzia do obsługi i logowania błędów jakie mogą wystąpić w trakcie działania aplikacji. Autor nie wykorzystuje ich bezpośrednio, ale są one niezbędne do poprawnego działania innych bibliotek.

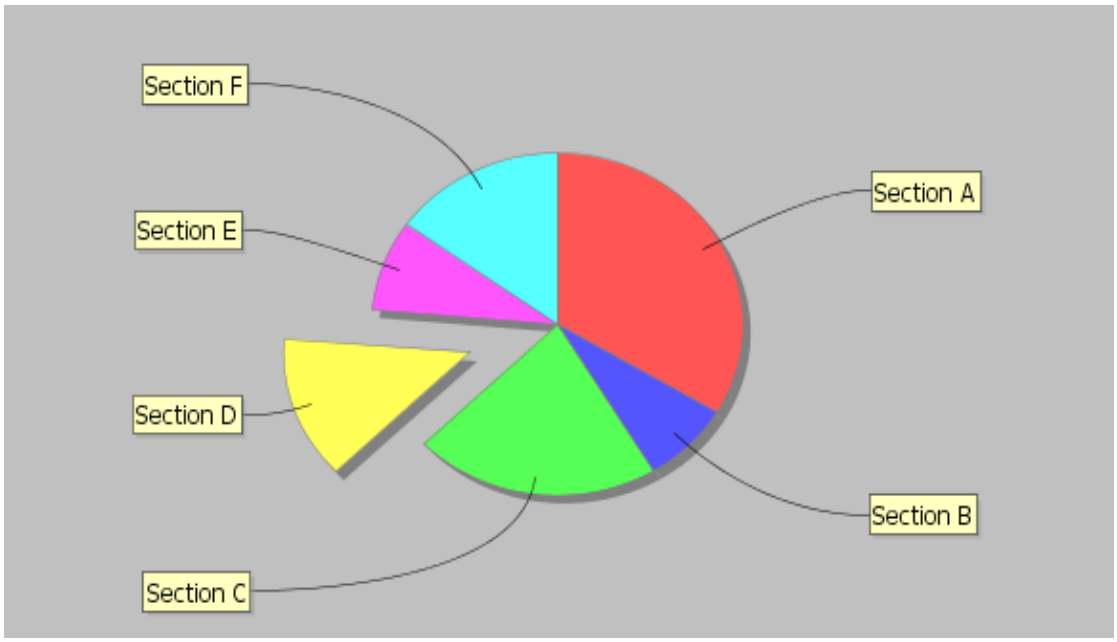
W skład pakietu httpclient wchodzą klasy klienta protokołu http. Do podstawowych zadań omawianego pakietu należy udostępnienie narzędzi do obsługi przepływu informacji przez protokół http. Podstawową klasą pakietu jest HttpClient. Po utworzeniu klienta tworzymy instancję klasy reprezentującej zapytanie. W tym wypadku będzie to klasa GetMethod. Do konstruktora przekazujemy adres zasobu do którego chcemy się odwołać. Nowo utworzony obiekt metody przekazujemy do metody executeMethod obiektu klienta. Ostatecznie pobieramy wynik działania zapytania przez wywołanie getResponseBody. Zwracany jest wynik w postaci tablicy bajtów. Ostatecznie zamykamy połączenie za pomocą releaseConnection.

```
HttpClient httpclient = new HttpClient()
GetMethod method = new GetMethod(href);
httpclient.executeMethod(method);
byte[] bytes = method.getResponseBody();
method.releaseConnection();
```

Narzędzia wchodzące w skład pakietu codec stanowią zbiór implementacji powszechnie stosowanych koderów i dekoderów takich jak base64, hex, phonetic, urls, binary. Wykorzystywane są one przez biblioteki użyte przy tworzeniu prototypu.

3.7 JFreeCharts

JFreeCharts (patrz [8]) jest darmową biblioteką udostępnianą na licencji LGPL w ramach projektu The JFreeChart. Umożliwia tworzenie wysokiej jakości wykresów. Może być wykorzystana zarówno w aplikacjach webowych jak i desktopowych.



Rysunek 7. Wykres kołowy biblioteki JFreeChart

Źródło: Opracowanie własne

Obsługuje wiele typów wyjściowych takich jak komponenty Swinga, pliki obrazów (PNG, JPEG) oraz formaty plików wektorowych (PDF, ESP, SVG).

Podstawowe wykresy możliwe do wygenerowania:

- Wykres kołowy [Rysunek 7]
- Wykres słupkowy
- Histogram
- Skumulowany wykres słupkowy
- Wykres liniowy

3.8 Cyber NekoHTML Parser

NecoHTML (patrz [10]) jest prostym parserem i stabilizatorem HTML, który wykorzystywany jest do przetwarzania dokumentów HTML. Dostęp do informacji z przetworzonych przez parser możliwy jest przez standardowe interfejsy XML.

Klasy parsujące oprócz przetwarzania źródła html dokonują naprawy struktury dokumentu i eliminują błędy powstałe w wyniku działania człowieka lub maszyny generującej html. Automatycznie zamykają znaczniki, wyłapują i dodają brakujące elementy.

NecoHTML został napisany w Javie z wykorzystaniem Xerces Native Interface (XNI), który jest podstawą implementacji Xerces2. Xerces2 jest implementacją standardu XML w projekcie Apache Xerces.

Parser html udostępniany jest na licencji Apache 2.0. Można go używać zarówno na potrzeby wolnego oprogramowania jak i do zamkniętych komercyjnych projektów.

4 PROJEKT SYSTEMU

Koncepcja pracy zakłada stworzenie generycznego systemu do analizy portali internetowych. Większość rozwiązań istniejących na rynku podchodzi do problemu analizy w sposób zbyt ograniczony. Skupia się na konkretnej dziedzinie lub portalu. Część z nich została opisana w rozdziale drugim. Po zapoznaniu się z informacjami tam zawartymi można stwierdzić, że brakuje aktualnie narzędzia, które podchodzi do problemu w szerszym wymiarze. Poniżej autor zaprezentuje nieistniejące na rynku rozwiązanie, które wyróżnia się uniwersalnym podejściem do analizy portali internetowych.

4.1 Przedmiot analizy

Dynamiczny rozwój globalnej sieci powoduje, że możemy znaleźć w niej informacje z każdej dziedziny naszego życia. Powstają portale zawierające treści na temat przepisów kulinarnych, ogłoszeń, wirtualnego handlu. Każdy z nich może być bogatym źródłem cennych informacji. Przeglądając portal kulinarny i analizując liczbę komentarzy, możemy dowiedzieć się jakie przepisy cieszą największą popularnością. Analizując portal aukcyjny możemy wywnioskować, czemu produkty z danej dziedziny sprzedają się lepiej od innych. Niestety duża ilość danych sprawia, że człowiek bez odpowiednich narzędzi informatycznych nie jest w stanie ich przeanalizować.

Rozwiązania istniejące na rynku dokonują analizy portali pod różnymi względami. Generalnie można podzielić je na dwie grupy. Pierwszą z nich stanowią narzędzia skupiające się na analizie technicznej. Zaś do drugiej zaliczyć można te, które koncentrują się na aspektach biznesowych.

Podejście proponowane w niniejszej pracy pozwoli na przełożenie informacji zawartych w sieci na konkretną wiedzę biznesową. Charakterystyka aplikacji przedstawionych w rozdziale drugim dostarczyła wielu cennych informacji na ten temat. Dowiedzieliśmy się jakiego typu portale poddawane są analizie a także jakie generują wyniki. Poznaliśmy również wady jakie te aplikacje posiadają. Podstawowym mankamentem jest ograniczenie się do jednego portalu lub dziedziny.

Zamysłem autora jest utworzenie uniwersalnego narzędzia, które pozwala na analizę danych pochodzących z różnych źródeł. W jednym miejscu, w krótkim czasie użytkownik będzie miał możliwość wyboru portalu, który go interesuje. Początkowa faza

funkcjonowania narzędzia przewiduje udostępnienie wyboru spośród kilku najbardziej popularnych. Ze względu na generyczne podejście do problemu możliwe będzie dodawanie kolejnych, wobec tego liczba portali nie będzie ograniczana w żaden sposób.

Autor zakłada, że nie będzie specjalnych wymagań jakie muszą spełniać portale. Niezależnie od strony dane muszą być pobrane i odpowiednio opakowane. Dzięki temu nie ograniczamy się do jednej kategorii witryn internetowych. Proces pobierania powinien być możliwy dla portali z każdym rodzajem informacjami.

4.2 Pobieranie danych

Omówione w poprzednim podrozdziale założenia możliwe są do zrealizowania poprzez stworzenie zbioru modułów odpowiedzialnych za proces pobierania danych. Zbiór ten będziemy nazywać pierwszym poziomem pluginów.

Każdy moduł, który wchodził będzie w skład tego poziomu będzie obsługiwał jeden portal internetowy. Jeśli będzie potrzeba analizy innego portalu wystarczy dopisać kolejny plugin, który go obsłuży. Dzięki takiemu podejściu nie ograniczamy się do jednego portalu.

Plugin, oprócz pobierania danych, musi wyodrębnić najważniejsze informacje i odpowiednio je opakować. Do tego celu powinien wykorzystać model danych oferowany przez system. Opakowywanie informacji jest ważną kwestią. Dzięki niej mamy możliwość przekazywania pobranych danych do innych części systemu. Po czym plugin nie jest już w żaden sposób odpytywany i może obsłużyć kolejne zapytania o pobranie informacji do analizy.

Aby zobrazować działanie pluginów pierwszego poziomu stworzone zostaną dwa moduły pobierania danych. Jednym z nich będzie moduł obsługujący portal Allegro.pl, drugi natomiast portal Ceneo.pl. Każdy z nich będzie pobierał i opakowywał dane w abstrakcyjny model danych. Przed procesem pobierania użytkownik będzie miał możliwość sprecyzowania jakie produkty go interesują. Do tego celu każdy z pluginów wyświetli formularz z parametrami precyzującymi żądanie.

4.3 Analiza danych

Modułowe podejście do procesu pobierania danych jest niezwykle uniwersalne. Możemy dodawać nowe portale i zbierać różnorodne informacje do analizy. W rozdziale drugim opisane narzędzia często po procesie pobierania danych wykonywały określone z góry analizy. Przeważnie odnosiły się tylko do jednego portalu. Jest to dobre rozwiązanie, ale mało uniwersalne. Przykładowo analizator obliczający średnią cenę produktów napisany dla portalu Allegro.pl w systemie Fortuna1, nie działałaby dla danych pobranych z systemu Manubia. Nie ma w tym nic dziwnego to są dwa różne systemy. Można sobie wyobrazić jak wiele by się wyróżniał system, który potrafiłby dokonać analizy danych, które pochodzą z różnych źródeł.

Omawiane podejście nie ogranicza możliwości analizy. Dane pochodzące z różnych źródeł będą mogły być analizowane przez jeden i ten sam analizator. Aby było to możliwe należy stworzyć, niezależny od pierwszego, drugi poziom pluginów.

Każdy moduł wchodzący w skład tego poziomu będzie pełnił inną funkcję analityczną. Dane, niezależnie skąd pochodzą, są przekazywane do modułu. Dzięki temu, że są opakowane w z góry znany model danych, plugin może je odczytać i dokonać analizy. Istnieje możliwość, że dane przekazane do analizatora nie są odpowiednie dla jego funkcji. Wystarczy w takim wypadku albo zwrócić wyjątek, który zostanie obsłużony przez program główny, albo wyświetlić informacje, że przekazane dane nie są odpowiednie.

Modułowość narzędzi analitycznych sprawi, że narzędzie go implementujące stanie się niezwykle uniwersalne. Nie będzie konieczne pisanie kolejnej funkcji do obliczania średniej ceny produktów nowego portalu internetowego. Będzie to już obsłużone przez wcześniej napisany moduł.

W ramach pracy przygotowanych zostanie sześć pluginów obrazujących działanie omawianego problemu. Każdy obsługiwać będzie dane pochodzące z obu pluginów pierwszego poziomu.

Pluginy drugiego poziomu:

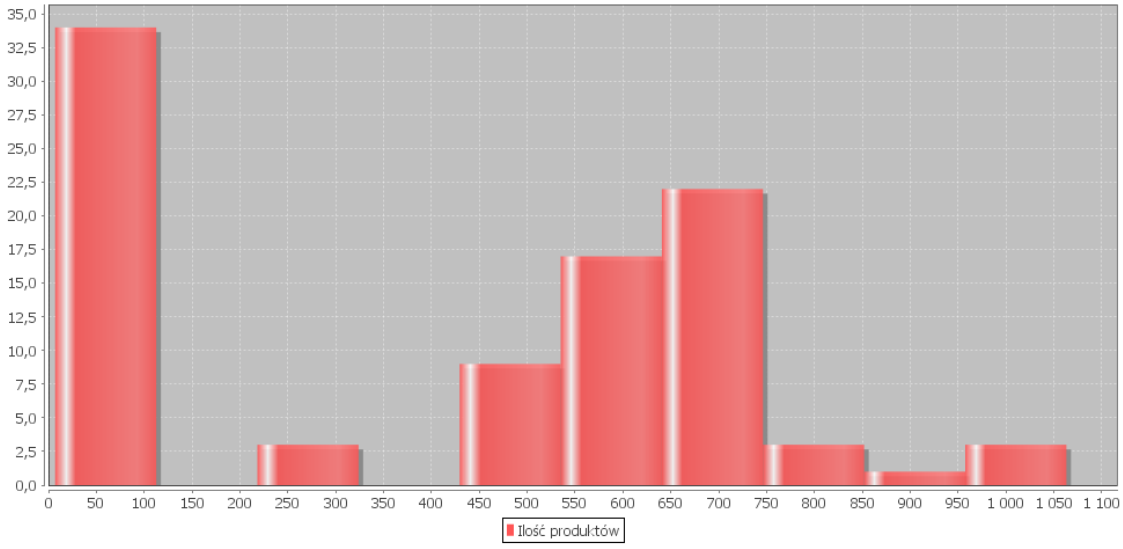
- Plugin do obliczania średniej ceny produktu
- Plugin do obliczania mediany cen produktu
- Plugin wyświetlający określoną liczbę najtańszych produktów

- Plugin wyświetlający określoną liczbę najdroższych produktów
- Plugin wyświetlający diagram histogram rozkładu cen
- Plugin wyświetlający diagram Ramka-Wąsy

Plugin obliczający średnią obliczał będzie średnią cenę produktów przekazanych do modułów. W przypadku mediany produkty zostaną posortowane od najtańszego do najdroższego i obliczona zostanie mediana cen.

Pluginy wyświetlające najdroższe i najtańsze produkty również będą musiały sortować produkty. W tym momencie autor napotkał problem. Operacja sortowania wykonywana jest aż trzy razy. Dzieje się tak gdyż każdy z modułów jest niezależny i działanie jednego nie wpływa na działanie drugiego. Sortowanie nie jest problemem gdyż przy zastosowaniu odpowiedniego algorytmu działa szybko. Lecz przy dużej liczbie produktów czas potrzebny do tej operacji w tym wypadku jest potrójny. Zakładając, że system będzie rozwijany w przyszłości należałoby zastanowić się jak rozwiązaniem tego problemu.

Innym rodzajem narzędzi analitycznych będą pluginy wyświetlające wykresy. Będą one przedstawiały rozkład cen według ilości produktów w przypadku histogramu [Rysunek 8], oraz tego jak się rozkładają ceny na przykładzie diagramu Ramka-Wąsy [Rysunek 9].



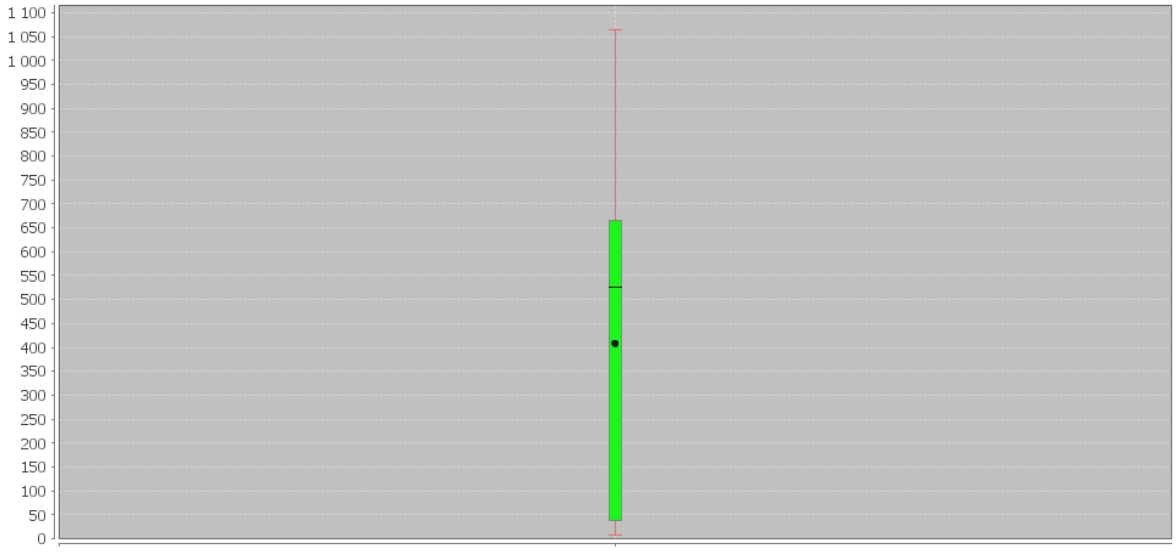
Rysunek 8 Histogram rozkładu cen produktów

Źródło: Opracowanie własne

Zadaniem histogramu jest ukazanie użytkownikowi ile produktów o określonej cenie zostało znalezionych na analizowanym portalu. Pozioma oś przedstawia cenę produktu a pionowa ilość ofert. W czytelny sposób obrazuje rozkład cen interesującego nas produktu.

Pozwala nam określić jakie wartości pojawiają się najczęściej (najwyższe słupki). Na podstawie histogramu możemy też dowiedzieć się czy średnia cena produktu nie jest zawyżona przez kilka wyjątkowo wysokich wartości. Informacja zawarta na tym wykresie dotyczy tez ilości produktów w danej cenie. Potencjalny sprzedawca może zorientować się jak wiele ofert, o podobnej wartości zamieszczonych jest w portalu.

Kolejny diagram Ramka-Wąsy [Rysunek 9] zawiera wiele ciekawych informacji na temat analizowanych danych. Użytkownik może odczytać minimalną i maksymalną cenie produktów. Wartości te zaznaczone są na rysunku za pomocą dwóch poziomych odcinków znajdujących się w górnej i dolnej części wykresu. Znajdują się one na końcach pionowych linii, elementy te nazywane są "wąsami". Zielony prostokąt, czyli "ramka" odzwierciedla typowe wartości ceny. Zakres ten, nazywany często zakresem międzykwartylowym ilustruje wartości cen jakie osiągnęło 50% produktów. Przedział ten otrzymujemy poprzez odrzucenie 25% najdroższych i najtańszych ofert. Wykres ramka-wąsy przedstawia w przejrzystej formie wiele parametrów opisowych, dzięki niemu możemy szybko zorientować się jak rozkładają się ceny interesującego nas produktu. Np. widoczna znaczna asymetria (jeden z wąsów jest znacznie krótszy, mediana odbiega istotnie od średniej) może zasygnalizować nam, że średnia cena została zawyżona przez małą liczbę ofert o wysokich wartościach. Zjawisko to widoczne jest na rysunku [Rysunek 9]



Rysunek 9. Diagram Ramka-Wąsy

Źródło: Opracowanie własne

Zaprezentowane moduły ukazują zasadę działania analizy przeprowadzanej przez system. Rozbudowa możliwości systemu będzie wiązała się z dodawaniem nowych

pluginów. Wraz z wzrostem ich liczby system stanie się niezwykle uniwersalnym narzędziem.

4.4 Wydajność

Projekt systemu zakłada, że stworzony prototyp będzie aplikacją webową. Działać będzie pod kontrolą kontenera aplikacji Tomcat 7. Wszystkie jego operacje udostępnione zostaną poprzez webowy interfejs użytkownika.

W związku z tym, że omawiane narzędzie udostępnione zostanie w sieci należy zadbać o jego dostęp do odpowiedniego łącza internetowego. Każde zapytanie skierowane do systemu będzie wiązało się z wykorzystaniem określonej przepustowości. Ponadto pluginy pierwszego poziomu pobierając odpowiednie dane z portali także obciążają łącze. Każda próba wykonania tej operacji to dodatkowe obciążenie. W pewnym momencie może się okazać, że nawet szybkie łącze serwerowe jest zbyt wolne. Wiązać się to będzie nie tylko ze znacznym wydłużeniem czasu oczekiwania na analizę, ale także może w znacznym stopniu spowolnić dostęp do samego systemu.

Aby zaradzić takiej sytuacji należy zoptymalizować system pod kontem korzystania z zewnętrznych źródeł. Autor niniejszej pracy uznał, że dobrym rozwiązaniem jest wdrożenie systemu cache'owania. Zadanie tego systemu wiązałoby się z pośrednictwem pomiędzy pluginami pierwszego poziomu a zewnętrznymi źródłami. Proces pobierania zaczynałby się od odpytania o dane menagera cache. Jeśli wcześniej zostały pobrane zwracane są z lokalnie przechowywanej kopii. Jeśli nie, manager pobiera je ze źródła zapisuje lokalną kopię i zwraca do pluginu. Kopia będzie mieć określony czas aktualności, po którym zostanie usunięta.

Omawiana optymalizacja w znacznym stopniu przyśpieszy proces zbierania danych. Szczególne ważne jest to w przypadku, gdy użytkownicy próbują dokonać analizy tych samych informacji. Jeśli zapytania o pobranie danych będą unikalne, system optymalizacyjny nie wpłynie na czas potrzebny do dokonania tej operacji. Może nawet powodować jego wydłużenie z powodu zapisywania lokalnej kopi. Istnieją jednak sytuacje, w których użytkownicy próbują analizować te same dane. Przykładem może być wielokrotne wysyłanie przez użytkownika tego samego formularza. Bez systemu cache plugin musiałby ponownie odpytywać witrynę o te same informacje. Optymalizacja powoduje, że dane raz pobrane mogą być wykorzystywane aż do momentu ich

dezaktualizacji. W przypadku prototypu autor uznał, że dane stają się nieaktualne po upłynięciu jednej godziny. Po tym czasie będą usuwane.

4.5 Prezentacja danych

Każda z powyższych funkcjonalności systemu, aby mogła być wykorzystana, musi być zaprezentowana w użyteczny sposób. Nawet najbardziej rozbudowane narzędzia z bogatą funkcjonalnością nie zostaną docenione przez użytkownika bez odpowiednio zaprojektowanego interfejsu.

Interfejs powinien spełniać kilka podstawowych funkcji. Jedną z nich jest intuicyjność obsługi. Większość podstawowych funkcji systemu powinna być łatwo dostępnych. Użytkownik w krótkim czasie zapoznaje się graficznym aplikacji. Jeśli nie będzie w stanie wyszukać podstawowej funkcji, która go interesuje, stwierdzi, że narzędzie jest bezużyteczne. Sytuacja taka może zniechęcić go do korzystania z danego programu.

Kolejną cechą dobrego interfejsu użytkownika jest spójność wyglądu. Człowiek przy ocenie oprogramowania w większości bada je pod względem estetycznym. Porównując aplikacje, które mają podobną funkcjonalność, najczęściej zwracamy uwagę w jaki sposób są one zaprezentowane. Nielogiczne rozmieszczenie przycisków, zbyt duża ich ilość, niespodziewana zmiany wyglądu interfejsu, wszystko to wpływa na negatywny odbiór systemu przez użytkownika.

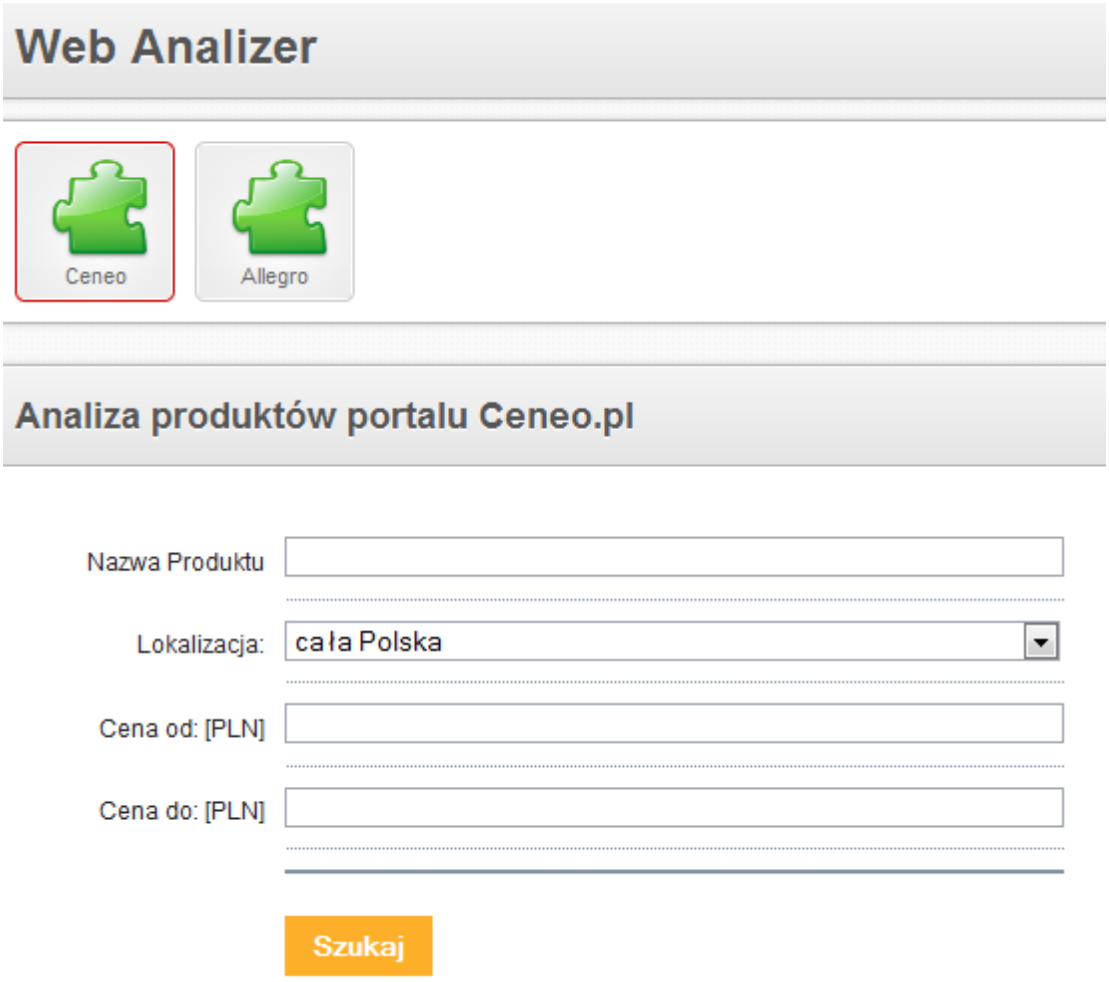
Stosując się do powyższych zaleceń autor stara się stworzyć prosty, estetyczny i łatwy w obsłudze interfejs dostępu do funkcji oferowanych przez system. Przed rozpoczęciem prac warto zastanowić się co powinien on oferować.

Do zadań interfejsu należeć będzie:

- Wyświetlenie informacji o systemie
- Wyświetlenie listy możliwych operacji do wykonania
- Umożliwienie wyboru portalu do analizy
- Wyświetlenie formularzy generowanych przez pluginy 1-szego poziomu
- Wyświetlenie wyników analizy pluginów 2 poziomu
- Umożliwienie zawężania wyników analizy

W związku z tym, że system będzie aplikacją webową należy zadbać o odpowiedni wygląd strony, na której wyświetlone zostaną jego funkcje. Zadanie to realizowane będzie przez skrypt layoutu. Umieszczony w nim kod HTML oraz style CSS będą podstawą wyglądu aplikacji. Kolejne elementy interfejsu dołączane będą do niego przez umieszczenie ich wewnątrz stworzonego szkieletu. Takie podejście ograniczy konieczność dopisywania kolejnych widoków, a wygląd systemu będzie spójny.

Projektowany prototyp będzie posiadał dwa poziomy modułów. Podstawowe funkcje odpowiedzialne za obsługę portali należą do pierwszego poziomu. Zadaniem interfejsu jest umożliwienie dostępu do nich. W tym celu wczytane pluginy pierwszego poziomu zostaną przedstawione w formie przycisków w górnej części strony. Takie rozmieszczenie powoduje, że z łatwością możemy wybrać portal, który nas interesuje. Po wybraniu witryny do wnętrza strony wczytany zostanie formularz. Wygenerowana strona będzie wyglądała tak jak to przedstawiono na rysunku [Rysunek 10].



Rysunek 10 Interfejs użytkownika - formularz

Źródło: Opracowanie własne

Po przesłaniu formularza pluginy pierwszego poziomu pobiorą dane i prześlą je do pluginów drugiego poziomu. Wyniki analiz zostaną wczytane do wnętrza strony. Nad wynikami wyświetli lista przycisków reprezentujących pluginy, które dokonywały analiz. Kliknięcie w jeden z nich spowoduje zawężenie ich do wybranego modułu. Funkcjonalność ta zilustrowana jest na rysunku [Rysunek 11].

Rysunek 11 Wyniki analizy portalu Ceneo.pl

Źródło: Opracowanie własne

Przedstawione rozwiązanie jest intuicyjne i łatwe w obsłudze. Nawet osoba, która ma po raz pierwszy styczność z systemem bez problemu dokona analizy poprzez wybór odpowiednich przycisków.

Poza omawianymi częściami interfejsu użytkownika istnieje dodatkowa, której zadaniem jest przekazanie informacji o systemie. W tym celu stworzono statyczne strony, które wczytywane są po kliknięciu w jeden z linków umieszczonych w nagłówku systemu.

Wszystkie założenia zostaną szczegółowo opisane w następnym rozdziale, gdzie poruszona zostanie kwestia implementacji.

5 IMPLEMENTACJA SYSTEMU

Rozdział ma za zadanie opisanie procesu implementacyjnego prototypu generycznego systemu analizy portali internetowych. Rozdział czwarty przedstawił ogólny projekt i założenia systemu, rozdział piąty będzie dotyczył implementacji przyjętych założeń.

5.1 Program główny

Do implementacji systemu wykorzystane zostaną narzędzia wchodzące w skład technologii Java Enterprise Edition. Prototyp będzie aplikacją webową działającą na serwerze, w tym wypadku będzie to Tomcat 7.

Do głównej części aplikacji webowej należeć będą serwlety i pliki JSP. Do ich zadań będzie należało wyświetlenie interfejsu użytkownika oraz zarządzanie, wyświetlanie i obsługa działania zewnętrznych modułów.

5.2 Serwlet główny

Główny serwlet *MainServlet* rozszerza funkcjonalność standardowej klasy *HttpServlet.* Metoda *initPlugins,* wchodząca w skład klasy, ma za zadanie obsługę zewnętrznych modułów. Po wywołaniu metody tworzona jest instancja menagera pluginów *net.xeoh.plugins.base.PluginManager*. Do tego celu wykorzystuje się fabrykę *PluginManagerFactory*. Obiekt menagera przekazywany jest jako parametr przy tworzeniu nowego obiektu *PluginManagerUtil.* Stworzony obiekt *PluginManagerUtil* pozwala na pobranie i obsługę pluginów z podanej lokalizacji. Aby dodać nową lokalizację do obiektu menagera wywołujemy metodę *addPluginsFrom,* a w jej parametrze podajemy źródło, z którego wczytywane będą pluginy.

Do pobierania pluginów wykorzystywana jest metoda *getPlugins*. Wywołujemy ją na obiekcie *PluginManagerUtil,* a w parametrze podajemy klasę pluginu. Wczytane zostaną tylko instancje danej klasy. Po wywołaniu metody pluginy trafiają do odpowiedniej kolekcji. Każda kolekcja odzwierciedla jeden poziom pluginów.

Stworzona klasa stanowi podstawę dla innych serwletów w prototypie systemu.

5.3 Pierwszy poziom pluginów

W skład pierwszego poziomu wchodzić będą pluginy, których zadaniem będzie pobieranie i wstępna obróbka danych z portali internetowych. Każdy z nich obsługiwać będzie jedną witrynę. Pluginy tego poziomu zwrócą też formularz do podania parametrów analizy.

Ogólny diagram klas pierwszego poziomu przedstawiony został na rysunku [Rysunek 12]. Klasa Allegro_Analizer implementuje interfejs Plugin. Interfejs Plugin rozszerza abstrakcyjny interfejs net.xeoh.plugins.base.Plugin z biblioteki Java Simple Plugin Framework.

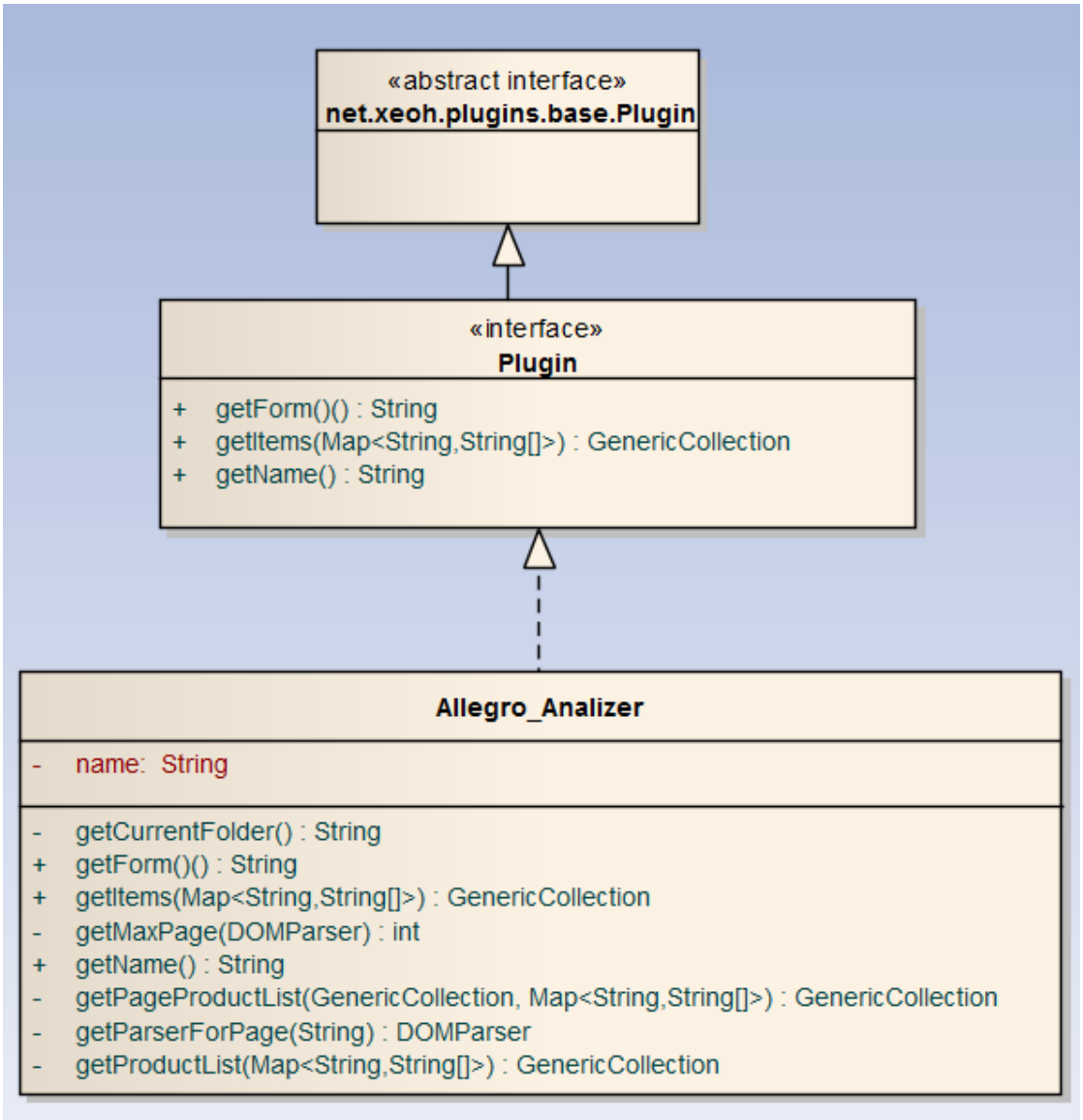
Metody wchodzące w skład interfejsu Plugin

- getForm
- getItems
- getName

Metoda *getName* zwracać będzie nazwę pluginu.

Metoda *getForm* zwróci łańcuch znaków zawierający kod HTML formularza. Powinien on zawierać pola, w których można będzie wskazać parametry potrzebne do przeprowadzenia analizy portalu internetowego.

Metoda *getItems* zwróci kolekcję danych pobranych z analizowanego portalu. Parametr przekazywany do metody powinien zawierać mapę atrybutów z wypełnionego formularza. Kolekcja GenericCollection rozszerzy klasę ArrayList ze standardowego pakietu Javy *java.util.ArrayList<E>*. Jej elementami będą obiekty GenericItem.



Rysunek 12. Diagram klas pluginu pierwszego poziomu Allegro_Analizer

Źródło: Opracowanie własne

Oprócz metod dostępnych publicznie plugin Allegro_Analizer implementuje prywatne metody użytku wewnętrznego. Nie można się do nich dostać z zewnątrz. Jedną z nich jest metoda *getParserForPage*. W parametrze metody przekazywany jest ciąg znaków z adresem strony internetowej. Do zadań metody należy zwrócenie treści analizowanej strony w formie obiektu klasy *DOMParser*. Kod potrzebny na realizację tego zadania został przedstawiony na poniższym listingu.

```
DOMParser parser;
try {
  parser = CacheManager.getInstance().getParserVObj(href);
  return parser;
```

```
}
catch (Exception e) { … }
return null;
```

Na początku metody deklarowana jest zmienna klasy DOMParser parser. Następnie przy użyciu statycznej metody CacheManager'a getInstance() zwracana jest instancja menagera cache. Przy użyciu tego obiektu wywołujemy metodę *getParserVObj* podając w parametrze adres strony. System optymalizacyjny, który zostanie opisany poniżej, zwraca obiekt parsera z zawartością strony o podanym adresie. Po udanej operacji zwróconą wartością inicjalizowana jest zmienna *parser*, po czym zostaje ona zwrócona do miejsca wywołania metody. W przypadku gdy operacja się nie powiedzie obsłużony zostanie wyjątek, a wartością zwróconą przez metodę będzie null.

Po tym jak zwrócony zostanie obiekt parsera będzie możliwe wykonanie kolejnej operacji - *getMaxPage*. Do zadań tej metody należy określenie na ilu stronach znajdują się wyniki. Najważniejsza jej część została przedstawiona na listingu.

```
…
NodeList nodes =
parser.getDocument().getElementsByTagName("a");
…
for(int i=0;i<nodes.getLength();i++) {
   node = nodes.item(i);
   NamedNodeMap nodeAtributes = node.getAttributes();
   for(int j=0;j<nodeAtributes.getLength();j++) {
      anode = nodeAtributes.item(j);
      try {
         int x = Integer.parseInt(node.getTextContent());
         if(x>0 && anode.getTextContent().equals("alleLink")
                && maxpage<x)
         {
             maxpage = x;
         }
      }
      catch(NumberFormatException nFE) {
           continue;
      }
   }
```

```
}
return maxpage;
```

Obiekt parsera zostaje wykorzystany do pobrania wszystkich linków jakie znajdują się w dokumencie. Następnie iterując po kolejnych elementach listy tworzymy mapę atrybutów linków. Kolejna pętla po atrybutach wyszukuje link, którego jeden z atrybutów ma wartość *alleLink*. Dodatkowo sprawdzana jest zawartość linka i porównywana z aktualną wartością zmiennej maxpage. W przypadku, gdy spełnione są warunki z klauzuli if, ustalana jest nowa maksymalna ilość stron wyników.

Mając ustaloną liczbę stron na której wyświetlane są produkty można przejść do następnego etapu pobierania danych. W tym celu iteracyjnie wywoływana jest metoda *getPageProductList* w jej parametrach podawany jest numer strony, parametry podane w formularzu oraz obiekt kolekcji produktów. Metoda rozpoczyna działanie przez skonstruowanie, na podstawie przesłanych argumentów, linku strony, do której będzie się odwoływać. Następnie za pomocą metody getParserForPage pobierany jest parser z kolekcją danych. Elementy strony są wyodrębniane i dodawane do obiektów GenericItem. W poniższym listingu umieszczono najważniejszą część kodu metody.

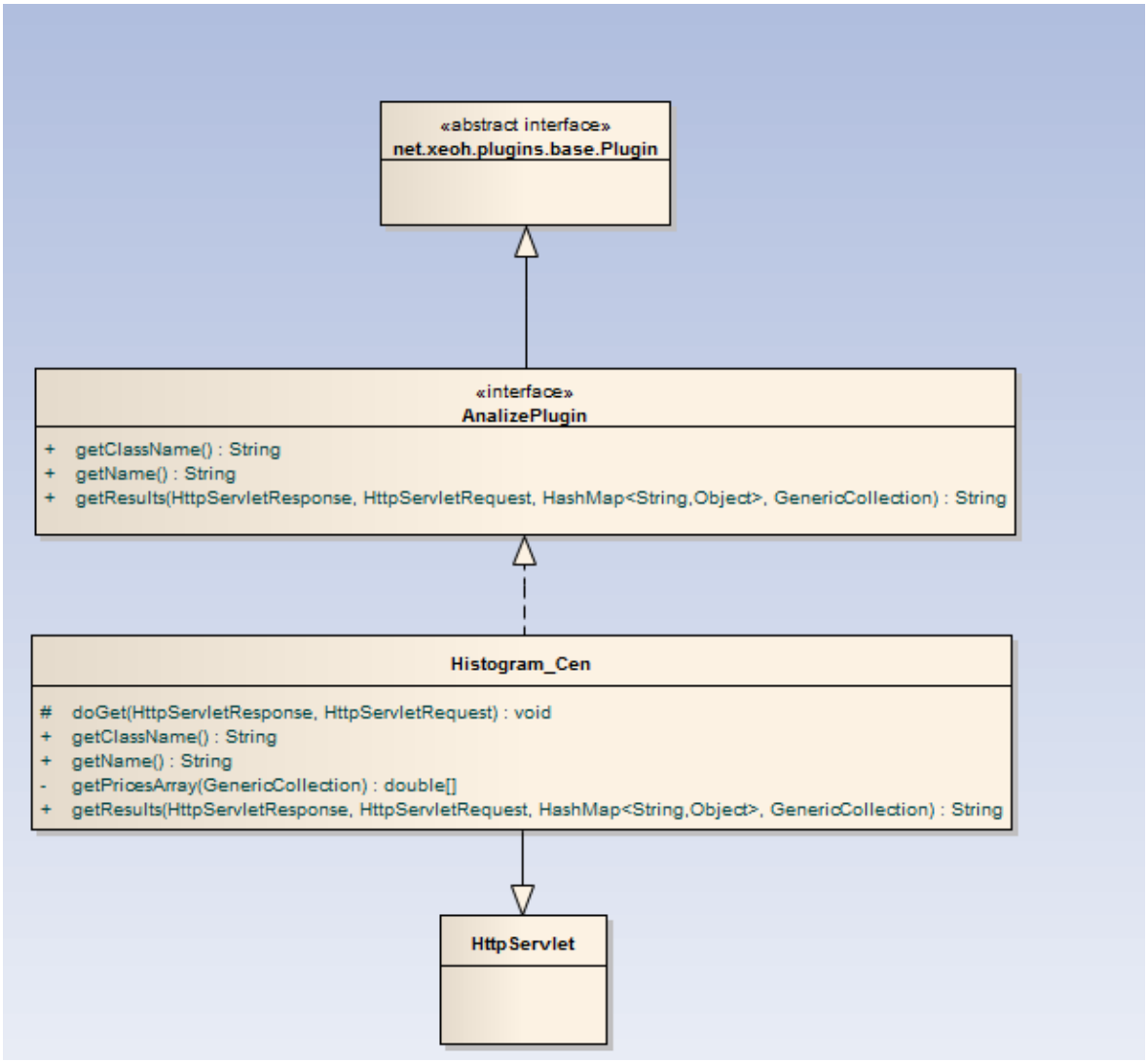
```
// wejście do boxu z opisem produktu
if(cnode.getTextContent().equals("cellName")) {
   //nazwa produktu
   Pattern p = Pattern.compile(".*");
   Matcher m = p.matcher(bnode.getTextContent());
   if(m.find()) {
     item.setValue("nazwa",
                   bnode.getTextContent().replaceAll("\\s{3,}|
                   \\n|\\r|start",""));
   }
   //link do produktu
   NodeList hnodes = bnode.getChildNodes();
   for(int h=0;h<hnodes.getLength();h++) {
      Node hhnode = hnodes.item(h);
      NamedNodeMap hnodeAtributes = hhnode.getAttributes();
      if(hnodeAtributes!=null) {
         Node Atrhhnode = hnodeAtributes.getNamedItem("href");
         if(Atrhhnode!=null) {
            item.setValue("href",Atrhhnode.getTextContent());
```

```
         }
      }
   }
}
// wejście do boxu z ceną produktu
if(cnode.getTextContent().equals("cellPrice")) {
   // wyrażenie wyszukujące wartość ceny produktu
   Pattern p = Pattern.compile("(\\d)*(\\s)?(\\d)*\\,(\\d)*");
   Matcher m = p.matcher(bnode.getTextContent());
   if(m.find()) {
      // dodawanie wartości do GenericItem
      item.setValue("cena", m.group().replace(",",
                    ".").replaceAll("\\s", ""));
   }
}
// wejście do boxu z ceną produktu wraz z dostawą
if(cnode.getTextContent().equals("cellTrans colOdd")) {
   // wyrażenie wyszukujące wartość ceny z dostawą
   Pattern p = Pattern.compile("(\\d)*(\\s)?(\\d)*\\,(\\d)*");
   Matcher m = p.matcher(bnode.getTextContent());
   if(m.find()) {
      item.setValue("cena_z_dostawa", m.group().replace(",",
                    "."));
   }
}
// wejście do boxu z ilością ofert
if(cnode.getTextContent().equals("cellOffer")) {
   // wyrażenie wyszukujące ilość
   Pattern pa = Pattern.compile("\\b[0-9]*\\b");
   // wyrażenie dla braku ofert
   Pattern pb = Pattern.compile("\\-");
   Matcher ma = pa.matcher(bnode.getTextContent());
   Matcher mb = pb.matcher(bnode.getTextContent());
   if(ma.find()) { // w przypadku gdy są oferty
      item.setValue("ofert",
                    ma.group().replaceAll("\\s|\\n|\\r|start",
                    ""));
```

```
    }
   if(mb.find()) { // w przypadku gdy brak ofert
       item.setValue("ofert", "0");
    }
}
…
products.add(item);
```

Przedstawiony listing w dużym skrócie ukazuje jakie operacje wykonywane są podczas obróbki pobranych danych. Po zakończeniu działania metody zwracana jest kolekcja opakowanych danych, którą można przekazać do pluginów drugiego poziomu.

5.4 Drugi poziom pluginów



Rysunek 13. Diagram klas pluginu drugiego poziomu Histogram_Cen

Źródło: Opracowanie własne

Elementami drugiego poziomu będą pluginy implementujące interfejs AnalizePlugin. Poziom ten odpowiedzialny będzie za analizę przekazywanej do niego kolekcji danych.

Metody interfejsu AnalizePlugin:

- getClassName
- getName
- getResults

Metoda getClass name powinna zwracać prostą nazwę klasy. Nazwa będzie wymagana przy generowaniu linków do konkretnych analizatorów.

Metoda getName zwróci ustaloną nazwę analizatora, która wyświetli się użytkownikowi. Służy to identyfikacji użytego analizatora.

Główną metodą w interfejsie będzie metoda GetResults. Parametrami metody są:

- HttpServletRequest request
- HttpServletResponse response
- HashMap<String,Object> params
- GenericCollection items

Parametry request i response, przekazywane do metody, mogą być przydatne przy próbie uzyskania dostępu do kontekstu serwletu bądź przy rozszerzaniu funkcjonalności serwletu przez plugin.

Mapa *params* zawierać będzie wszystkie parametry przekazane po przesłaniu formularza pluginu pierwszego poziomu. Metoda pobierać będzie także kolekcję danych stworzoną przez plugin pierwszego poziomu. Dane przekazywane będą w parametrze *items*. Mając do dyspozycji wszystkie przekazane parametry metoda dokona właściwych sobie analiz i przekaże wynik, w formie łańcucha znaków, do programu głównego.

Na przykładzie klasy Histogram_Cen omówione zostaną przykładowe metody dokonujące analizy przekazanych danych. Klasa oprócz implementacji interfejsu Plugin rozszerza funkcjonalność klasy *HttpServlet*. Dzięki temu możliwe będzie odwołanie się do pluginu z poziomu przeglądarki internetowej. Metoda *getResults* zwraca kod przedstawiony na poniższym listingu.

```
<img src=\""+request.getContextPath()+"/Histogram_Cen"
border="0" />
```

Na początku metody umieszczono w sesji dane, które będą wyświetlane na wykresie. Po wyświetleniu zwróconego kodu HTML przeglądarka starać się będzie wyświetlić obraz. W tym momencie odwoła się do adresu podanego w parametrze znacznika. Wskazuje on na serwlet omawianej klasy. W tym momencie wywoływana jest metoda doGet pluginu.

```
HistogramDataset jc = new HistogramDataset();
double[] data = (double[])
request.getSession().getAttribute("chartHistogramImgData");
jc.addSeries("Ilość produktów", data, 10);

   JFreeChart redChart = ChartFactory.createHistogram(
               "Histogram.",
               null,
               null,
               jc,
               PlotOrientation.VERTICAL,
               true,
               true,
               false);

   XYPlot redPlot = (XYPlot) redChart.getPlot();
   redPlot.setForegroundAlpha(0.85f);
   BufferedImage img = redChart.createBufferedImage(980, 500);
   response.setContentType("image/png");
   PngEncoder encoder = new PngEncoder(img, false, 0, 9);
   response.getOutputStream().write(encoder.pngEncode());
```

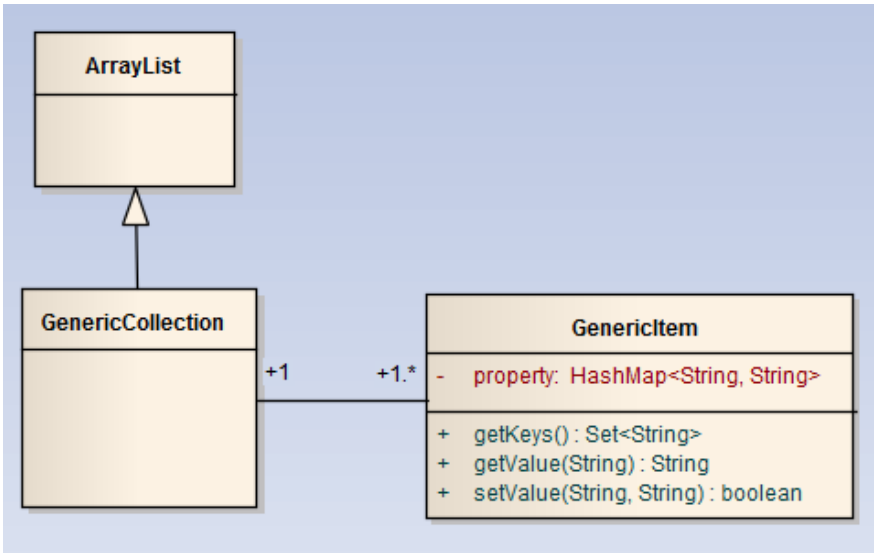
Kod metody umieszczony na powyższym listingu wyświetla wygenerowany histogram. Tworzony jest obiekt *HistogramDataset*. Następnie pobierane są dane z sesji użytkownika. Pobrane dane wraz z innymi parametrami dodawane są do obiektu diagramu. Na fabryce wykresów wywoływana jest statyczna metoda generująca histogram. Przekazywane są do niej parametry określające właściwości wykresu. Po dokonaniu kolejnych operacji z obiektu histogramu generowany jest obraz PNG. Ustawiane są

odpowiednie nagłówki odpowiedzi. Ostatecznie do wyjściowego strumienia obiektu responce przekazywany jest ciąg bajtów reprezentujący plik PNG.

Działanie powyżej opisanych operacji powoduje wyświetlenie się wygenerowanego histogramu na ekranie użytkownika.

5.5 Model danych

Informacje przekazywane pomiędzy poziomami pluginów będą opakowane w abstrakcyjny model danych. Założenia projektu zilustrowane zostały na diagramie klas [Rysunek 14].



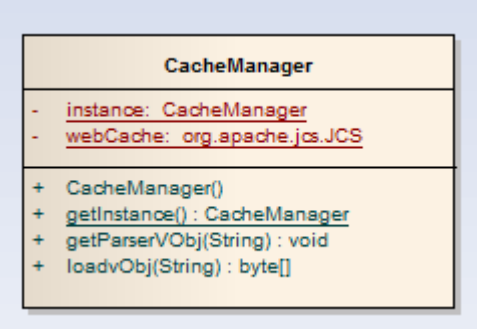
Rysunek 14. Diagram klas modelu danych

Źródło: Opracowanie własne

GenericCollection rozszerzy funkcjonalność standardowego typu ArrayList z pakietu *java.util*. W skład kolekcji wejdą obiekty klasy GenericItem. Klasa umożliwi stworzenie prostego modelu danych, który posiada klucz i wartość zapisaną pod tym kluczem. Dane przechowywane będą w haszowanej tablicy asocjacyjnej (ang HashMap). Przedstawiony model pozwoli na przekazanie niezbędnych informacji pomiędzy dwoma poziomami pluginów.

5.6 Optymalizacja pobierania danych

Każda próba pobrania porcji danych z portalu internetowego wiązać się będzie z wykorzystaniem pewnej przepustowości pasma dostępu do Internetu. Wraz ze wzrostem liczby użytkowników korzystających z omawianego systemu wzrastać będzie obciążenie łącza. W pewnym momencie może się okazać, że czas potrzebny do pobrania danych będzie tak długi, że system stanie się niefunkcjonalny. Aby zaradzić takiej sytuacji niezbędne jest wdrożenie systemu cache.



Rysunek 15 Diagram klasy CacheManager

Źródło: Opracowanie własne

Diagram przedstawiający implementację systemu optymalizacyjnego przedstawiony został na rysunku [Rysunek 15]. Zadaniem systemu cache będzie pośrednictwo pomiędzy pluginami pierwszego poziomu, a analizowanymi portalami. Zapytanie w formie adresu URL przesłane zostanie do systemu optymalizacyjnego. Cache sprawdzi czy dane już wcześniej nie zostały pobrane. Jeśli tak, to odczytane zostaną z nośnika cachu i zwrócone do pluginu. W przypadku gdy dane nie zostały wcześniej pobrane tworzony jest obiekt HttpClient. Klient pobierze niezbędne dane i zwróci ciąg bajtów. System cache zapisze dane pod adresem URL źródła danych. Po zapisaniu dane zostaną zwrócone do analizatora.

Opisana funkcjonalność została zrealizowana za pomocą kodu przedstawionego w listingu.

```
DOMParser parser = new DOMParser();
byte[] bytes = (byte[]) webCache.get(href);
if (bytes == null) {
   bytes = loadvObj(href);
}
InputSource is = new InputSource();
if(bytes!=null) {
```

```
   ByteArrayInputStream bs = new ByteArrayInputStream(bytes);
   is.setByteStream(bs);
}
parser.parse(is);
return parser;
```

Kod z powyższego listingu przedstawia kolejne instrukcje zwracające obiekt parsera. Na obiekcie menagera wywoływana jest metoda *get*, która zwraca dane umieszczone w cache pod podanym w parametrze kluczem. Jeśli zostanie zwrócona wartość null, dane pobierane są za pomocą metody *loadvObj(href)*. Następnie dane opakowywane są w klasę jaka może zostać przekazana do parsera. Po operacji parsowania obiekt zostaje zwrócony.

Metoda *loadvObj(href)* wywoływana w powyższym listingu pobiera dane z zewnętrznego źródła i zwraca je w postaci ciągu bajtów. Najważniejsza jej część została umieszczona poniżej.

```
HttpClient httpclient = new HttpClient();
GetMethod method = new GetMethod(href);
method.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,
  new DefaultHttpMethodRetryHandler(3, false));
httpclient.executeMethod(method);
byte[] bytes = method.getResponseBody();
method.releaseConnection();
…
webCache.put(href, bytes);
…
return bytes;
```

Pobieranie danych zaczyna się od stworzenia klienta http i obiektu metody requestu. Do metody przekazywane są parametry określające maksymalną liczbę prób uzyskania odpowiedzi od źródła. Następnie na obiekcie klienta wykonywana jest metoda odpowiedzialna za wysłanie zapytania. Jeśli nie wystąpił żaden błąd metoda zwraca ciąg bajtów pobranych danych. Zamykane jest połączenie, a kopia zwróconego ciągu umieszczana jest w systemie optymalizacyjnym. Ostatecznie zwracany on jest do miejsca wywołania metody.

Konfiguracja systemu cache powinna pozwalać na określenie miejsca przechowywania danych, a także czasu przechowywania danych. Omawiany system optymalizacyjny

korzysta z zewnętrznego pliku konfiguracyjnego. Plik o nazwie *cache.ccf* należy umieścić w głównym katalogu źródeł. Zostanie on automatycznie wczytany przez menagera. Poniżej przedstawiona jest konfiguracja jaka została stworzona na potrzeby prototypu.

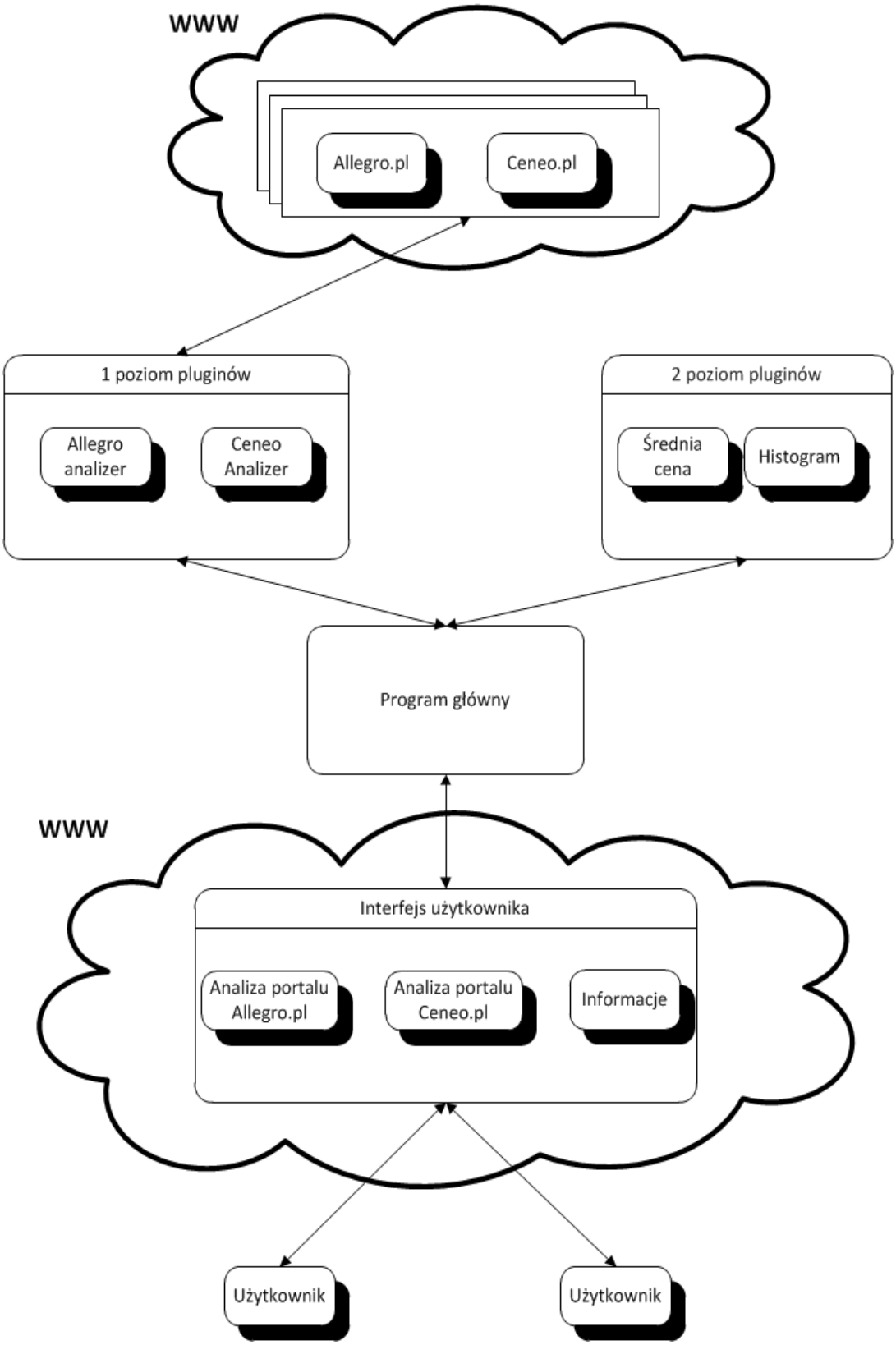
```
jcs.default=DC
jcs.default.cacheattributes=org.apache.jcs.engine.CompositeCacheAttributes
jcs.default.cacheattributes.MaxObjects=100000

jcs.region.webCache=DC
jcs.region.webCache.cacheattributes=org.apache.jcs.auxiliary.disk.indexed.Ind
exedDiskCacheAttributes
jcs.region.webCache.cacheattributes.MaxObjects=100000
jcs.region.webCache.cacheattributes.MemoryCacheName=org.apache.jcs.engine.mem
ory.lru.LRUMemoryCache
jcs.region.webCache.cacheattributes.UseMemoryShrinker=true
jcs.region.webCache.cacheattributes.MaxMemoryIdleTimeSeconds=360
jcs.region.webCache.cacheattributes.ShrinkerIntervalSeconds=360
jcs.region.webCache.elementattributes=org.apache.jcs.engine.ElementAttributes
jcs.region.webCache.elementattributes.IsEternal=false
jcs.region.webCache.elementattributes.MaxLifeSeconds=3600

jcs.auxiliary.DC=org.apache.jcs.auxiliary.disk.indexed.IndexedDiskCacheFactor
y
jcs.auxiliary.DC.attributes=org.apache.jcs.auxiliary.disk.indexed.IndexedDisk
CacheAttributes
jcs.auxiliary.DC.attributes.DiskPath=/tmp
jcs.auxiliary.DC.attributes.maxKeySize=100000
```

Najważniejsze parametry jakie można ustawić za pomocą pliku z powyższego listingu to miejsce przechowywania danych, maksymalna liczba obiektów jakie można dodać do cache, maksymalny czas ich przechowywania, czas po jakim wywoływane są metody usuwające nieaktualne obiekty oraz maksymalny rozmiar kluczy do obiektów.

5.7 Architektura systemu

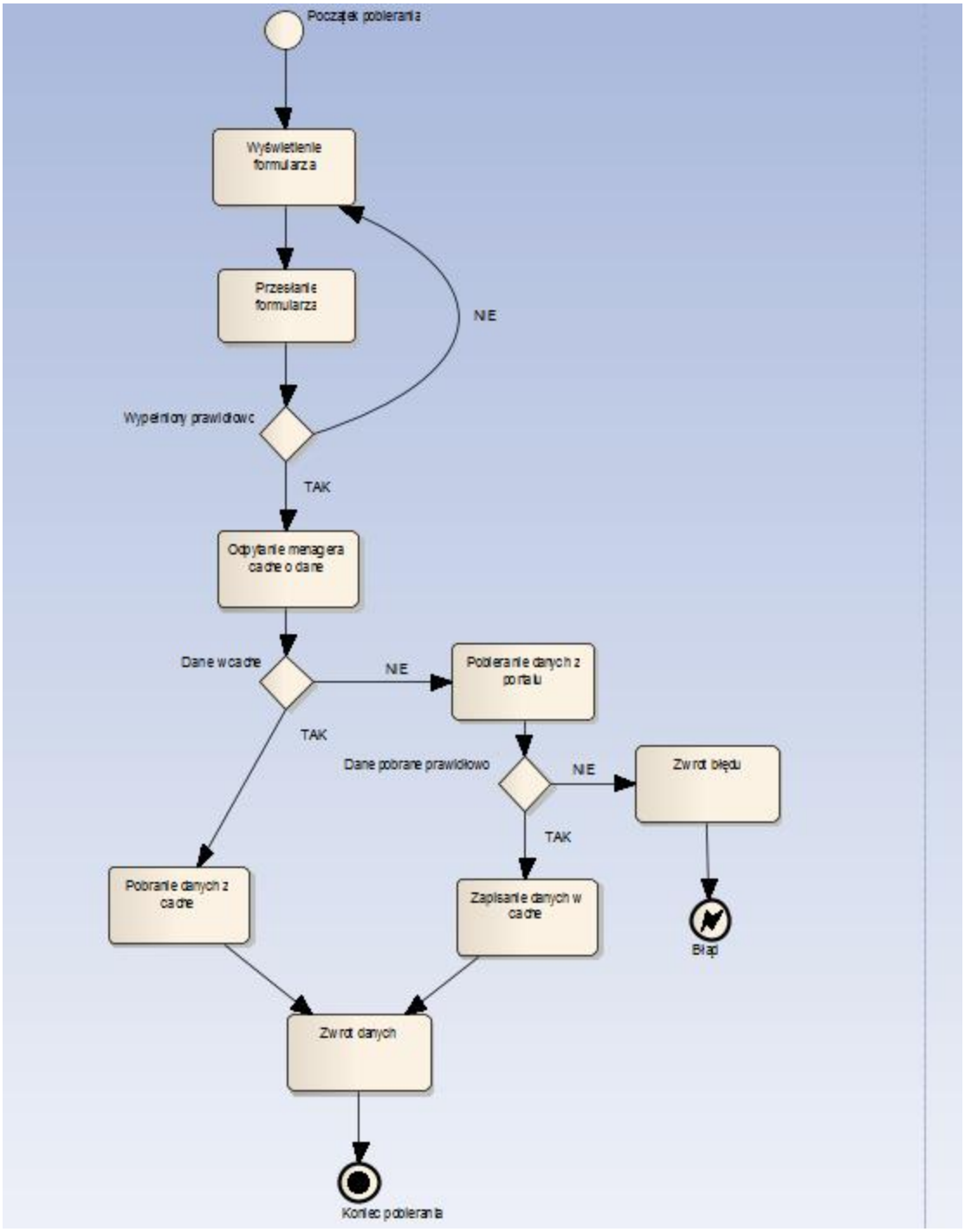


Rysunek 16. Architektura systemu analizy portali internetowych

Źródło: Opracowanie własne

Przedstawiony na rysunku [Rysunek 16] diagram obrazuje ogólną architekturę omówionego w poprzednich rozdziałach prototypu systemu. Zilustrowane zostały części składowe zaprezentowanego narzędzia oraz kierunki przepływu informacji.

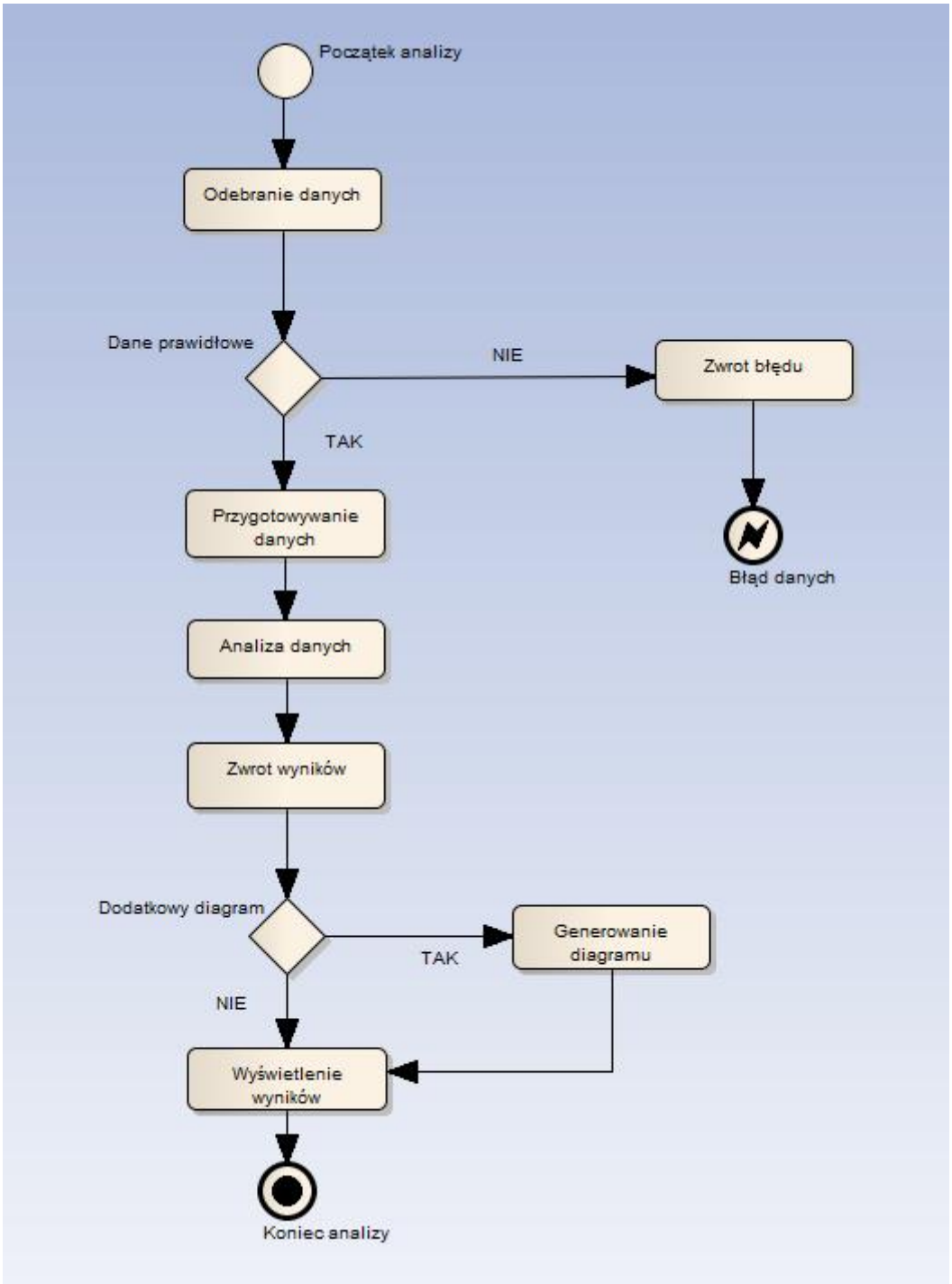
W kolejnej części podrozdziału przedstawione zostaną diagramy aktywności dla operacji pobierania i analizy danych.



Rysunek 17. Diagram aktywności procesu pobierania danych

Źródło: Opracowanie własne

Diagram na rysunku [Rysunek 17] przedstawia kolejne etapy przepływu sterowania w procesie pobierania danych z analizowanej strony internetowej. Kolejny rysunek [Rysunek 18] przedstawia diagram aktywności dla procesu analizy danych.



Rysunek 18. Diagram aktywności procesu analizy danych

Źródło: Opracowanie własne

6 PODSUMOWANIE

Analiza danych we współczesnym świecie staje się coraz ważniejsza. Wielkie korporacje tworzą własne hurtownie danych, aby opierając się na analizach zebranych informacji, móc przewidzieć kierunek rozwoju swojej firmy.

Dzięki dynamicznej ewolucji globalnej sieci dostęp do dużej ilości danych nie wymaga już gromadzenia danych ich wiele lat. Portale internetowe, często odwiedzane przez tysiące użytkowników, zawierają ich wystarczająco dużo. Można było je wykorzystać w podobny sposób jak te pochodzące z hurtowni. Serwisy aukcji internetowych zawierają informacje na temat milionów transakcji, które po przeanalizowaniu mogą dostarczyć wielu cennych informacji biznesowych.

Opierając się na powyższych przemyśleniach autor zaproponował wykorzystanie zaistniałych możliwości do realizacji omówionego podejścia do generycznego pobierania i analizy danych pochodzących z dowolnych portali internetowych. W trakcie realizacji powyższej pracy zaproponowane zostały założenia jakie powinny zostać spełnione przy próbie projektowania i implementacji tego typu oprogramowania.

Stworzony w ramach pracy prototyp systemu ukazuje działanie przyjętych założeń i może zostać wykorzystany do celów komercyjnych.

7 BIBLIOGRAFIA

[1]. **Rychlicki-Kicior, Krzysztof**. *Java EE 6. Programowanie aplikacji WWW*. Gliwice : Helion, 2010. 978-83-246-2659-5.

[2]. **Eckel, Bruce.** *Thinking in Java 3rd Edition*. New Jersey : Prentice Hall PRT, 2003. 0-13-100287-2

[3]. Dokumentacja Eclipse 3.6 Helios http://help.eclipse.org/helios/index.jsp

[4]. Dokumentacja Java Simple Plugin Framework http://code.google.com/p/jspf/

[5]. Dokumentacja Tomcat 7 http://tomcat.apache.org/tomcat-7.0-doc/index.html

[6]. Dokumentacja JavaEE 6 http://www.oracle.com/technetwork/java/javaee/documentation/apis-139520.html

[7]. Dokumetacja Apache Commons http://commons.apache.org/

[8]. Dokumentacja JFreeCharts http://www.jfree.org/jfreechart/api/javadoc/index.html

[9]. Dokumentacja standardu JSTL http://java.sun.com/products/jsp/jstl/reference/docs/index.html

[10]. Dokumentacja NecoHTML Parser http://nekohtml.sourceforge.net/

8 SPIS RYSUNKÓW

9 SPIS TABEL