Wydział Podstawowych Problemów Techniki
Politechnika Wrocławska

Framework e-commerce

Przemysław Magiera
Nr indeksu: 229773

Praca inżynierska napisana
pod kierunkiem
Wojciecha Macyny

Politechnika
Wrocławska
Wrocław 2017

Spis treści

Wstęp

Celem niniejszej pracy dyplomowej jest zaprojektowanie i zaimplementowanie frameworku służącego do usprawnienia implementacji systemów e-commerce. Istnieje wiele rozwiązań tego typu, jednak bardzo duża część z nich nie oferuje satysfakcjonujących parametrów wydajnościowych, przez co platformy oparte o takie frameworki są czesto bardzo powolne, do tego rozwijane od wielu lat wykorzystują stare rozwiązania i technologie. Prowadzi to często do niepotrzebnego skalowania pionowego aplikacji, czyli zwiększania mocy obliczeniowej. Proces ten wiąże się z bardzo dużymi kosztami, szczególnie w przypadku platform handlowych typu B2B. Zdecydowanie lepszym wyjściem okazuje się w takim przypadku jeden z dzisiejszych trendów budowania aplikacji, czyli skalowanie poziome, polegające na dzieleniu aplikacji według zastosowania poszczególnych komponentów, umieszczając niezależne jej części na serwerach dziedzinowych (architektura mikroserwisowa). Taka architektura pozwala na skalowanie tylko konkretnych, najbardziej narażonych na wzmożony ruch, elementów infrastruktury, co skutkuje bardzo dużą oszczędnością w stosunku do aplikacji monolitycznych. Budowa mikroserwisowa nie jest jednak cudownym środkiem na każdego rodzaju problemy dzisiejszych aplikacji internetowych, wiąże się z nim bowiem wiele problemów, jak chociażby integracja i synchronizacja między komponentami lub konieczność administracji bardzo złożonego środowiska. Właśnie ze względu na to ostatnie widzimy dziś tak wiele ofert pracy na stanowisko DevOps (development and operations).

Często dewelopment aplikacji idzie w parze z presją czasu, przez co zapomina się o jakości kodu i rozwiązaniach, które poprawiłyby wydajność i ograniczyły konieczność skalowania. Z zamkniętymi oczami podąża się za schematami i szablonami, aby dostarczyć rozwiązanie jak najszybciej, a nie jak najlepiej. Dlatego właśnie założeniem projektu w ramach pracy jest zaprojektowanie i implementacja frameworku e-commerce spełniającego następujące założenia funkcjonalne:

- wykorzystanie najnowszych technologii oraz IoC
- podłączony zewnętrzny serwer Apache Solr, służący do bardzo szybkiej obsługi zapytań związanych z katalogiem produktowym,
- prosty i efektywny system klasyfikacyjny dla produktów,
- reużywalny i rozszerzalny panel administracyjny
- zaawansowana obsługą uprawnień dla panelu administracyjnego,
- elastyczny model, pozwalający na rozszerzanie klas bez konieczności ingerowania w strukture systemu
- łatwy w skonfigurowaniu i wydajny mechanizm wyszukiwania,
- relacyjna i nierelacyjna baza danych,
- zaiplementowana obsługa procesu zamówienia,

- inicjalizer projektów, pozwalający szybko stworzyć przykładowe rozwiązanie e-commerce.
- mechanizm wersjonowania produktu

Praca składa się z czterech rozdziałów. W rozdziale pierwszym omówiono strukturę przedsiębiorstwa . . . , scharakteryzowano grupy użytkowników oraz przedstawiono procedury związane z obiegiem dokumentów. Szczegółowo opisano mechanizmy . . . . Przedstawiono uwarunkowania prawne udostępniania informacji . . . , w ramach . . . . W rozdziale drugim przedstawiono szczegółowy projekt systemy w notacji UML. Wykorzystano diagramy . . . . Zawarto w niej w pseudokod algorytmów generowania oraz omówiono jego właściwo- ści. . .

Słowniczek

indeksacja – proces synchronizacji pomiędzy relacyjną bazą danych, a szybką bazą noSQL, używany w systemie do encji najbardziej narzuconych na duże wykorzystanie. W skrócie:

> *By adding content to an **index**, we make it searchable by Solr.*[1]

facet – jest to atrybut danej encji, zazwyczaj wyszukiwalny. Używa się ich do implementacji filtrów używanych podczas wyszukiwania. Z dokumentacji Solra:

> *Searchers are presented with the indexed terms, along with numerical counts of how many matching documents were found were each term. Faceting makes it easy for users to explore search results, narrowing in on exactly the results they are looking for.* [1]

release – jest to atrybut danej encji, zazwyczaj wyszukiwalny. Używa się ich do implementacji filtrów używanych podczas wyszukiwania. Z dokumentacji Solra:

> *Searchers are presented with the indexed terms, along with numerical counts of how many matching documents were found were each term. Faceting makes it easy for users to explore search results, narrowing in on exactly the results they are looking for.* [1]

release – nieskopoziomowe udogodnienie w języku Java, pozwalające na operacje i wyświetlanie właściwości klasy Javowej.

> *Reflection is a feature in the Java programming language. It allows an executing Java program to examine or "introspect"upon itself, and manipulate internal properties of the program. For example, it's possible for a Java class to obtain the names of all its members and display them.* [4]

Analiza problemu

W tym rozdziale została przedstawiona analiza zagadnienia. Nakreślono problemy i omówiono proponowane przez system ich rozwiązania. Omówiono założenia funkcjonalne i niefunkcjonalne samego systemu jak i jego podsystemów, przedstawiono podobne rozwiązania informatyczne. Zawarty jest również słowniczek, potrzebny do pełnego zrozumienia zagadnienia.

Charakterystyka problemu

Aplikacje webowe, a w szczególności systemy e-commerce, są często wolne. Pisane przez niedoświadczonych deweloperów pod presją czasu nie zawsze wychodzą tak jak powinny. A jak powinny być napisane? W pierwszej kolejności muszą być dobrze przemyślane, a co za tym idzie ich architektura powinna być przygotowana na rozszerzenia i modyfikacje. Powiedzmy, że mamy sklep internetowy, na bazie którego chcielibyśmy stworzyć podobne rozwiązanie. Często jest to niemożliwe ze względu na budowę i obsługę komponentów. Dobrym przykładem na to jest indeksowanie produktów do mechanizmu wyszukującego.

Przykład 2.1 *W systemie istnieje klasa Product z zadeklarowanymi polami biznesowymi, powiedzmy że klient chce nowe pole myCustomField, oczywiście ma być ono wyszukiwalne. Rozszerzamy więc klasę Produkt do MyProdukt i dodajemy pole myCustomField. Mechanizm indeksacji nie ma możliwości wyciągnąć z Produktu pola dotyczącego klasy MyProdukt, ponieważ zaprzeczałoby to zasadom polimorfizmu.*

To tylko konkretny przykład, ale warto zwrócić uwage, że dotyczy on nie tylko produktów, a i również jakichkolwiek encji, którymi chcemy zarządzać w systemie. To pierwszy problem rozwiązań e-commerce, które nie są oparte o elastyczne frameworki.

Kolejną rzeczą idąca za słabą architekturą są tak zwane bottlenecki[1], które blokują szybkie działanie aplikacji. Przykład z życia:

Przykład 2.2 *Nasza platforma jest oparta o framework stworzony parę lat temu, do tego taki, który nie jest open-source, ma architekturę monolityczną. Nasz sklep zyskał na popularności i coraz częściej zdarzają nam się awarię związane z ograniczoną wydajnością, najszybszą i naturalną reakcją jest dokupienie nowego serwera do obsługi większej ilości klientów, jednak wiąże się to z bardzo dużymi kosztami.*

W tym przykładzie warto zwrócić uwagę na zamknięty charakter frameworku. Brak licencji open-source często skutkuje to tym, że projekt jest skazany na zamknięty cykl rozwoju (o ile w ogóle jest dalej rozwijany), ewentualne błędy nie mogą być naprawione *od ręki*. Raz napisana platforma komercyjna nie będzie po 5 latach wcale aktualna. Inaczej sprawy się mają w przypadku open-source'wych frameworków, gdzie możliwość zmiany technologi, a aktualizacje są dostępne praktycznie zawsze. To dzięki zasadzie *inversion of control*, kiedy to programista decyduje się na oddanie kontroli nad

[1] bottleneck - wspólny punkt dla aplikacji, przez muszą przejść użytkowicy korzystając z różnych funkcjonalności

częścią swojej funkcjonalności w ręce używanego przez siebie frameworku, aktualizacja do nowszych wersji używanych technologii jest zwykle bezbolesna i opiera się głównie na zmianie wersji w pliku konfiguracyjnym zależności projektowe.

Następny problem związany ze środowiskiem e-commerce to brak dobrych wyszukiwarek produktów na stronach. Często zdarza się, że wyszukiwarki przeszukują relacyjne bazy danych, zamiast korzystać z płaskich struktur jakie oferują nam rozwiązania typu noSQL. Do tego nie obsługują facetów [2], co sprawia trudności z wyszukaniem produktu i dopasowaniem go pod klienta, a przecież to jest główny biznes. Jasne i wiadome jest, że istnieją sklepy z dobrymi wyszukiwarkami, do tego mogące pochwalić się wysokim miernikiem TPS[3]. Jednakże są to rozwiązania bardzo drogie i niedostępne dla małych przedsiębiorstw, z drugiej strony implementowanie takich rzeczy na własną rękę jest również bardzo drogie, a do tego skomplikowane. W tym momencie z pomocą przychodzą właśnie frameworki, nie wszystkie jednak maja pełna obsługe mechanizmu indeksującego, a szczególnie nie w darmowych wersjach.

Istotną sprawą w opisywanych systemach jest archiwizacja produktu, bądź jej zupełny brak. Problem został opisany na poniższym przykładzie:

Przykład 2.3 *Klient zakupił produkt A o atrybutach (a, b, c, d) w sierpniu. W październiku manager sklepu zmienił atrybuty produktu A na (e, f, g), co wpłynęło również na cenę. W listopadzie klient zdecydował się na reklamację produktu. Obsługa sklepu otrzymała zgłoszenie, ma id produktu, jednak niewiele to pomoże, gdyż atrybuty i cena uległy zmianie.*

Ostatnią, nie mniej jednak ważną rzeczą, która warta jest opisania to sposób, w jaki charakteryzuje się produkty[4] w sklepach internetowych. Produkty należy opisywać w konsekwentny i najbardziej jednoznaczny sposób, jednak nie da się tego osiągnąć bazując tylko na polach ewentualnej klasy Produkt, jest to wybitnie nieelastyczne, każda zmiana wymaga trwałej ingerencji w kod i ponowny release.

Powyższe problemy niektórych systemów e-commerce można sprowadzić do następujących stwierdzeń:

- architektura nieprzygotowana pod nagłe modyfikacje i rozszerzenia
- najczęściej używane punkty aplikacji nie są odseparowane od reszty
- brak dobrych wyszukiwarek i mechanizmów filtrujących
- niekonkterny i mało abstrakcyjny sposób opisu encji biznesowych
- użycie zamkniętych technologi, nie korzystanie z open source, brak zastosowania *inversion of control*
- brak archiwizacji produktów

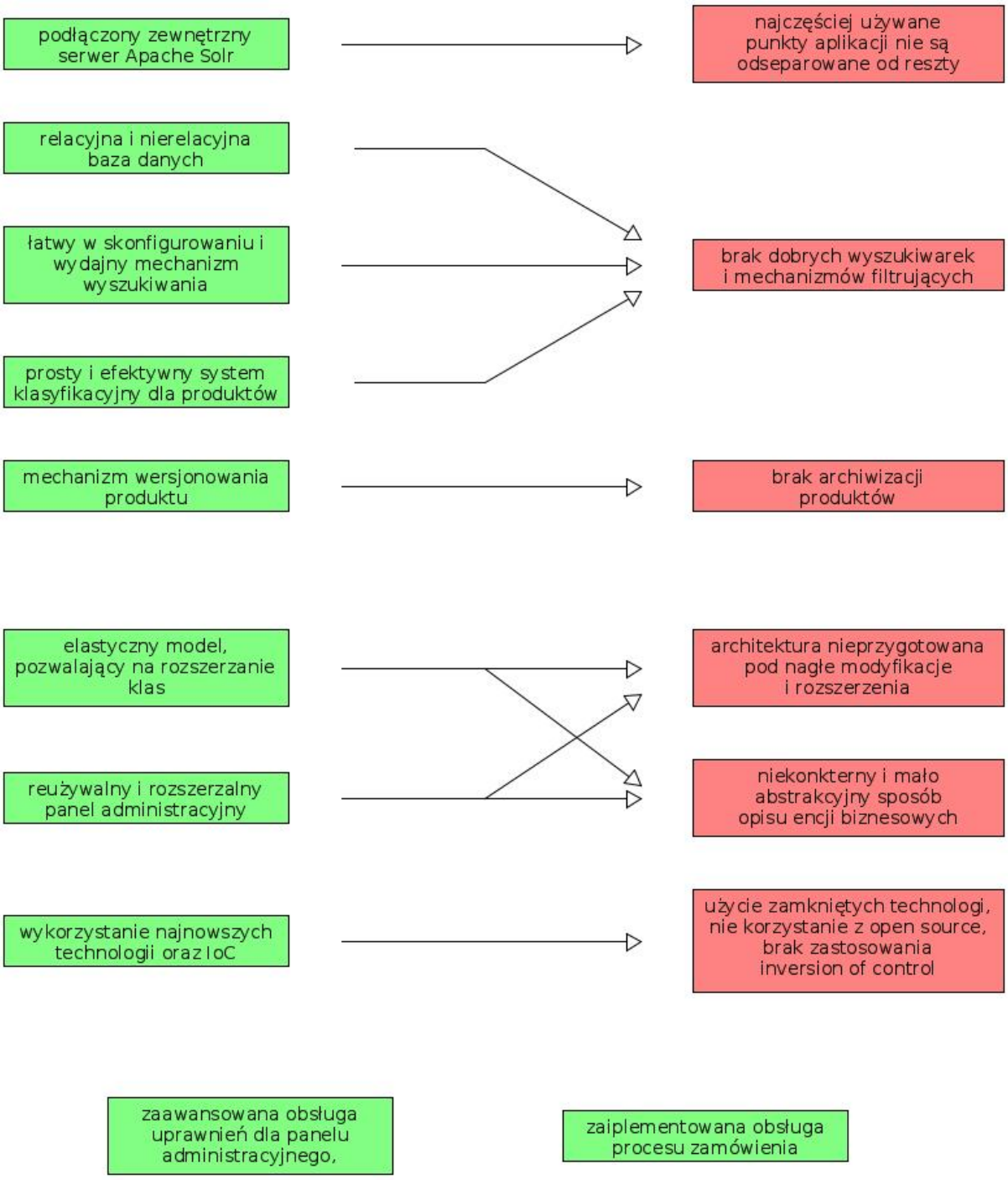
Proponowane rozwiązania

W wstępie określono listę wymagań funkcjonalnych. W podrozdziale **Charakterystyka problemu** nakreślono najważniejsze problemy sklepów internetowych i frameworków e-commerce. Rysunek 2.1 ilustruje pokrycie znalezionych problemów wymaganiami funkcjonalnymi opisywanego frameworka.

[2] wyjaśnienie terminu dostępne w sekcji **Słowniczek**

[3] *TPS* – transaction per second, ilość pełnych requestów wraz z odpowiedzią, jaką może obsłużyć serwer na sekundę.

[4] lub cokolwiek innego podlegającego klasyfikacji



Rysunek 2.1: Połączenie wymagań funkcjonalnych ze znalezionymi problemami.

Z rysunku 2.1 wynika, że *bez pokrycia* zostają dwa wymagania, nie oznacza to jednak, że są to niepotrzebne funkcjonalności. Są jednak powszechnie spotykane w frameworkach e-commerce, można nazwać je standardem. Jednak co mniej oczywiste system uprawnień został oparty na strukturze drzewiastej, uprawnienia są dziedziczone pomiędzy użytkownikami finalnego systemu opartego o framework.

Projekt systemu

W tym rozdziale przedstawiono szczegółowy projekt systemy w notacji UML uwzględniający wymagania funkcjonalne opisane w rozdziale 2. Do opisu relacji pomiędzy składowymi systemu wykorzystano diagramy .... Przedstawiono w pseudokodzie i omówiono algorytmy generowania ....

Grupy użytkowników i założenia

Przypadki użycia

Jak zostało zdefiniowane w poprzednim punkcie, w pracy przewidziano 3 grupy użytkowników. W zamyśle framework jest narzędziem dla programisty, jednak w systemie został zaimplementowany szereg rozwiązań gotowych do wykorzystania dla końcowych użytkowników, dlatego diagramy przypadków użycia zostały podzielone na trzy klasy:

- przypadki użycia Programisty
- przypadki użycia Użytkownika Administracyjnego potencjalnego serwisu e-commerce, opartego na opisywanym Frameworku
- przypadki użycia użytkownika końcowego, czyli Klienta

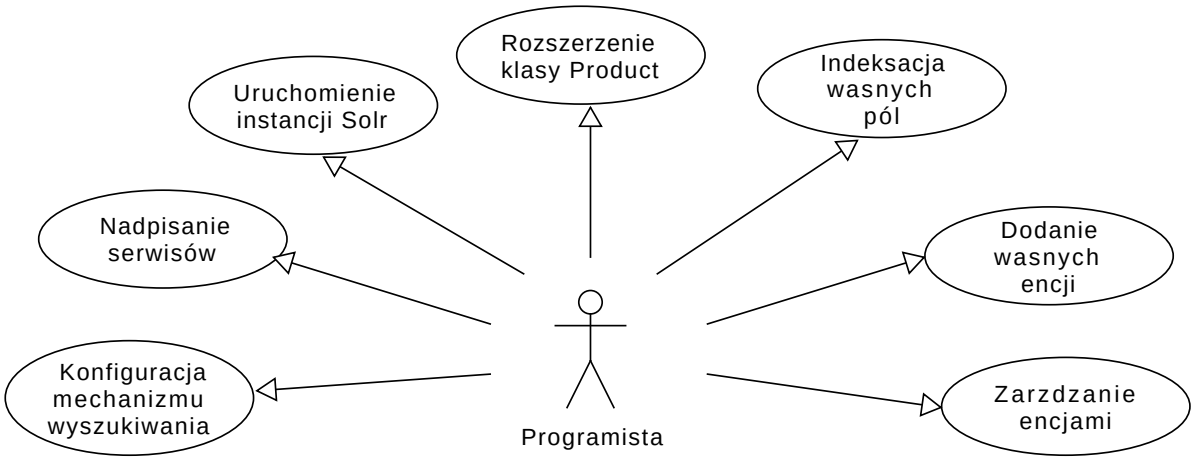
Na rysunku 3.1 zostały przedstawione najważniejsze przypadki użycia frameworku. Programista ma swobodny dostęp do rozszerzania encji, w szczególności klasy Produkt, która ma wyjatkowo strategiczne znaczenie w systemach e-commerce. Dodatkowo ma możliwość uczynienia niestandardowych pól wyszukiwalnymi przez klienta. Sytuacja została zobrazowana na poniższym przykładzie.

Przykład 3.1 *Załóżmy, że mamy niestandardowe pole proste (String) w encji klasyfikowanej przez twórców ewentualnego sklepu związanego z opisywanym frameworkiem jako finalna encja nadająca się do sprzedaży. Niech nazywa się* `MyProduct extends Product` *z polem* `myCustomField`*. Jedyne co w tej sytuacji musimy zrobić aby system mógł wyciągnąć wartość tego pola z encji (o której de facto nie wie) to wpisać do tabeli zawierającej indeksowane cechy produktu nazwę danego pola, system za pomocą refleksji*[1] *wyekstaktuje wartość pola z encji.*

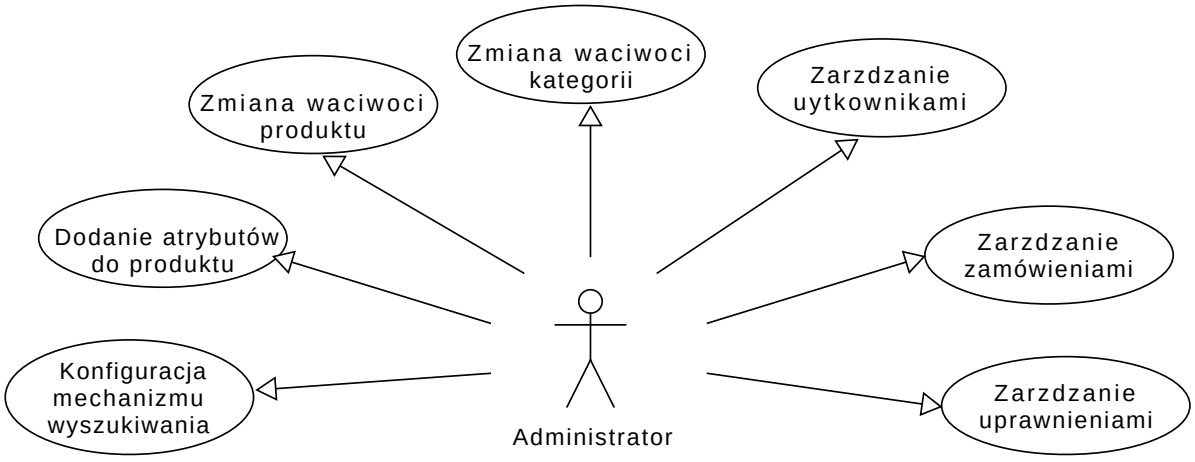
Dodane przez Programistę encje są obsługiwane przez framework, dodatkowo po dodaniu specjalnej adnotacji[2] nad nią, może być zarządzana w uniwersalnym panelu administracyjnym. Osoba zajmująca się implementacją sklepu opartego o opisywaną platformę może uruchomić dowolną *(skończoną)*

[1] *refleksja* (eng. reflection) – udogodnienie w języku Java, pozwalające na wyświetlenie i manipulacje właściwościami klasy. Więcej w sekcji **słowniczek**.

[2] adnotacja – używane w języku Java od wersji 1.7, najczęściej służą do określania dodatkowych właściwości pól bądź klas



Rysunek 3.1: Diagram przypadków użycia związany z Programistą.

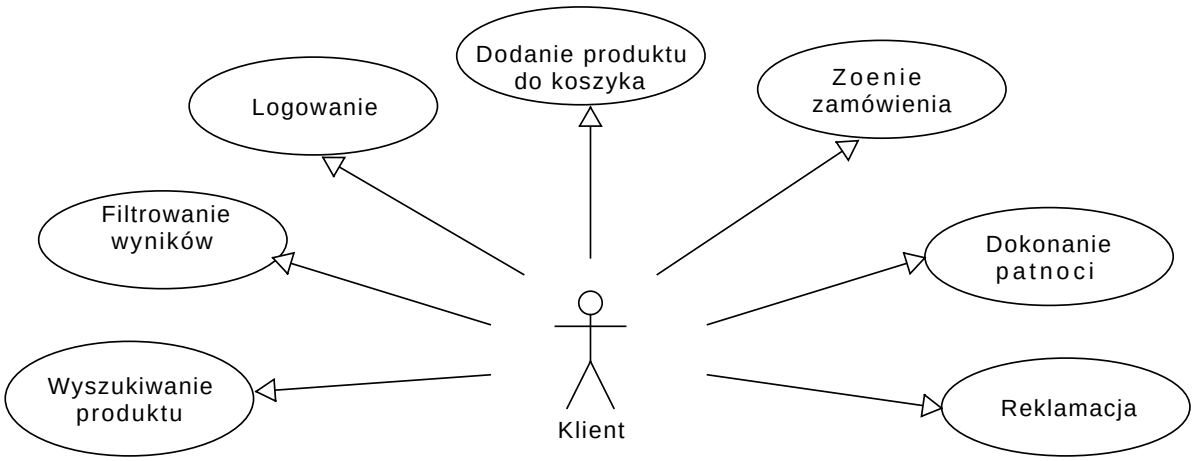


Rysunek 3.2: Diagram przypadków użycia związany z Administratorem ewentualniego systemu.

ilość instancji Apache Solr, czyli bazy danych noSQL, służącej do obsługiwania zapytań związanych z katalogiem produktowym (skalowalność pionowa tylko tej części aplikacji, która tego potrzebuje). W odniesieniu do przypadku użycia *Nadpisanie mechanizmu wyszukiwania* z rysunku 3.1 serwisy są oparte na interfejsach, zapewniając Programiście możliwość nadpisania jego logiki zgodnie z zasadami polimorfizmu.

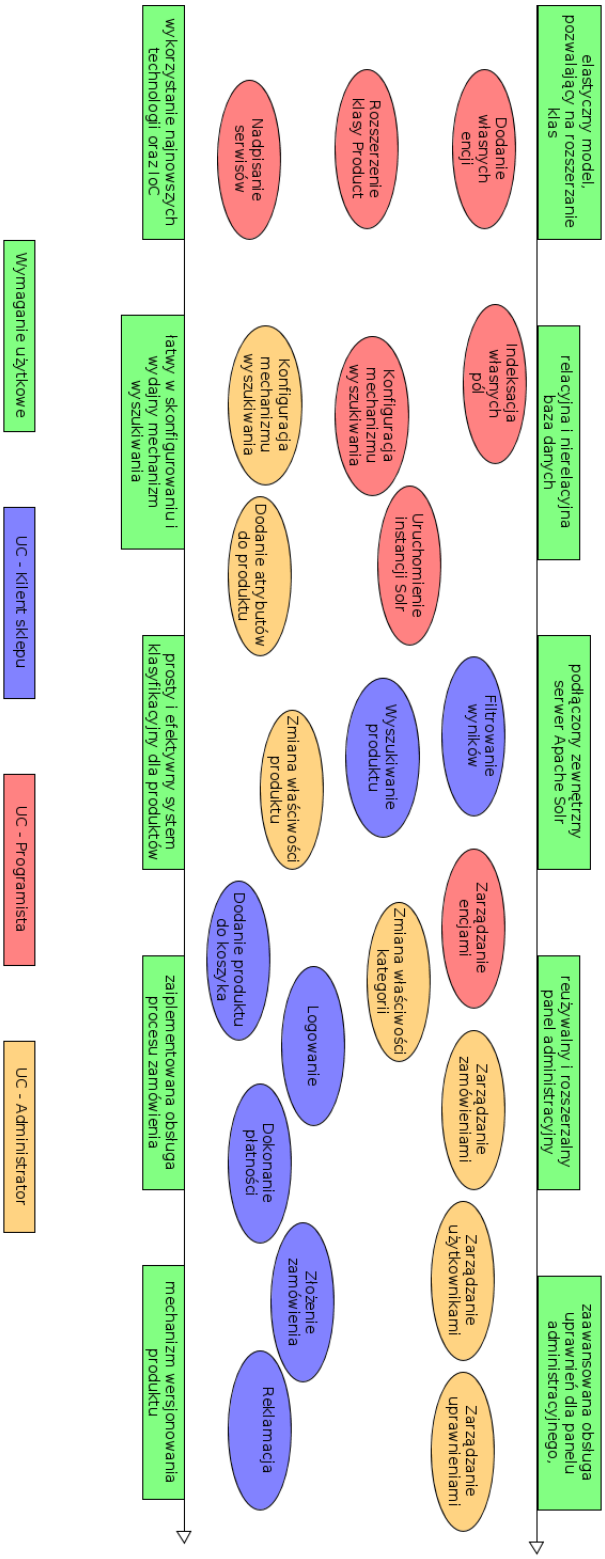
Rysunek 3.2 przedstawia przypadki użycia z punktu widzenia Administratora potencjalnego systemu. Z punktu widzenia platformy jest to również klient, gdyż framework zakłada, że nie ma on wiedzy technicznej i nie potrafi programować. Podobnie jak programista, może konfigurować mechanizm wyszukiwania, jednak bardziej wysokopoziomowo, np. deklaracja używanych facetów. Panel administracyjny zakłada zarządanie najważniejszymi encjami: produkt, kategoria, użytkownik, zamówienie, uprawnienie i parę innych, zdefiniowanych dokładniej w podrozdziale **Diagramy bazy danych**.

Diagram na rysunku 3.3 dotyczy przypadków użycia elementów frameworku przez końcowego użytkownika. Są to klasyczne funkcjonalności tradycyjnego sklepu internetowego. *Wyszukanie produktu* zostało zaprojektowane, tak aby możliwy był również do zaimplementowania mechanizm podpowiedzi i podświetlania. Apache Solr udostępnia taką funkcjonalność. *Reklamacja* dotyczy opisanego w rozdziale **Analiza problemu** kłopotu z archwizacją produktu, został on rozwiązany prostym mechanizmem wersjonowania.



Rysunek 3.3: Diagram przypadków użycia związany z Administratorem ewentualniego systemu.

Diagramy typu *use-case* ściśle wiążą się z wymaganiami funkcjonalnymi systemu. Jest wiadome, że można je również sklasyfikować pod względem aktorów występujących w systemie, dlatego też powiązania przypadków użycia z wymaganiami funkcjonalnymi zostały umieszczone na diagramie z rysunku 3.4. Na diagram należy patrzeć poziomo, po zapoznaniu się z legendą. Wymagania są w nieprzypadkowej kolejności, są ustawione od lewej do prawej. Im bardziej na prawo, tym wymaganie jest bardziej biznesowe, im bardziej na lewo – dotyczy rdzeniowych elementów platformy. Warto zauważyć zależność, że im dalej patrzymy na diagram, tym więcej niebieskich i żółtych *use case'ów* – tych zarezerowowanych dla Administracji i Klientów rozwiązania e-commerce. Natomiast im bardziej na lewo tym więcej czerwonego, czyli przypadków przemyślanych dla Programisty.



Rysunek 3.4: Diagram przypadków użycia związany z wymaganiami funkcjonalnymi

Diagramy aktywności

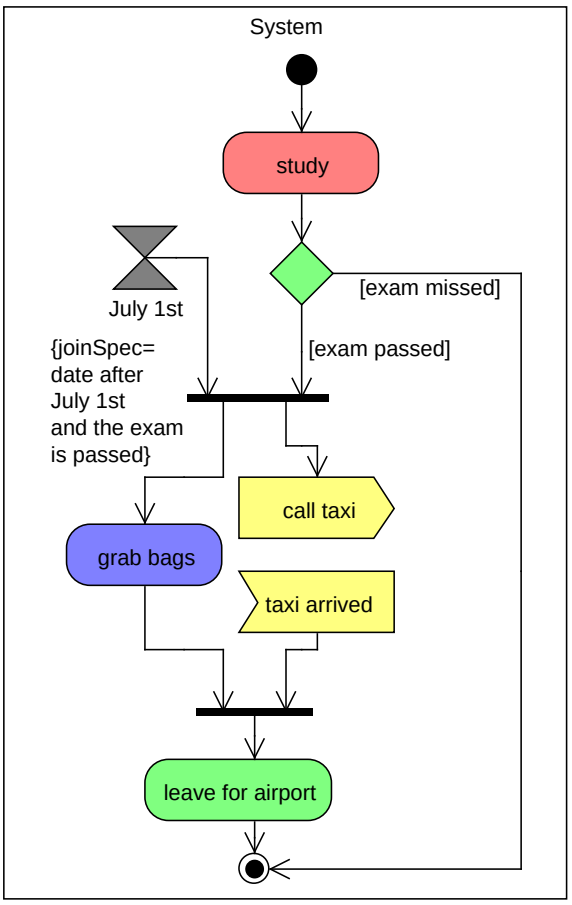
W tej sekcji zostały przedstawione diagramy aktywności dla elementów systemu. Aby zachować spójność w rozważaniach podzielono je na trzy grupy:

- diagramy dotyczące budowania rozwiązań e-commerce za pomocą platformy
- diagramy dotyczące zarządzania rozwiązaniami e-commerce
- diagramy dotyczące przebiegu funkcjinalności biznesowych związanych z kientem końcowym

Jak można zauważyć, te trzy grupy korespondują z aktorami opisywanego rozwiązania, zdefiniowanymi na początku rozdziału. Punkt pierwszy dotyczy Programisty, drugi Administratora, a ostatni Klienta końcowego.

Diagramy dotyczące dewelopmentu aplikacji

Niniejsze diagramy będą opisywały procesy związane z budowaniem rozwiązań opartych na podstawie opisywanego frameworku.



Rysunek 3.5: Diagram aktywności związany z procesem rejestracji dokumentu.

Diagramy sekwencji

W tej sekcji należy przedstawić diagramy sekwencji dla obiektów systemu zidentyfikowanych na podstawie wcześniejszych rozważań. Należy wykorzystać nazewnictwo wprowadzone w poprzednich rozdziałach, w szczególności odpowiadające definicjom wprowadzonych klas.

Diagramy stanów

W tej sekcji należy przedstawić diagramy stanów w których może znaleźć się system. Diagramy te są szczególnie istotne przy projektowaniu systemów czasu rzeczywistego.

Diagramy klas

W tej sekcji należy przedstawić diagramy klas dla odpowiednich elementów systemu zidentyfikowane na podstawie wcześniejszych rozważań

Projekt bazy danych

W tej sekcji należy przedstawić projekt bazy danych. Należy omówić wycinek rzeczywistości i odpowiadające mu zidentyfikowane elementy systemu, których wartości będą podlegać utrwalaniu. Należy przedyskutować wybór typów danych dla atrybutów poszczególnych obiektów. Należy uzasadnić wybór platformy DBMS. Dla relacyjnych baz danych należy przedyskutować jej normalizację.

Opis protokołów

W tej sekcji należy omówić protokoły wykorzystywane przez komponenty systemu. Omówić formaty komunikatów i zilustrować je przykładami.

Opis algorytmów

W tej sekcji należy wymienić i przedyskutować algorytmy wykorzystywane w systemie. Algorytmy należy przedstawić w pseudokodzie (wykorzystać pakiet `algorithm2e`). Omówienia poszczególnych kroków algorytmów powinny zawierać odwołania do odpowiednich linii pseudokodu. Dla zaproponowanych autorskich algorytmów należy przeprowadzić analizę ich złożoności czasowej i pamięciowej.

Algorytm bąblowania jest przedstawiony w Pseudokodzie 3.1.

Pseudokod 3.1: Wyporność przez bąblowanie

Input: Zbiór bąbli B
Output: Wyporność W

```
1  foreach b ∈ B do
2      Process(b);
3      for i ← 1 to |B| do
4          if Calculate(EW(i,b)) ⩽ 0 then
5              b ← 2 ∗ b;
6  while B ≠ ∅ do
7      for j ← 1 to |B| do
8          if Calculate(FT(j,b̂)) ⩽ 0 then
9              w ← 2 ∗ b̂;
10             W ← W ∪ {w};
11             B ← B \ {b};
```

Implementacja systemu

Opis technologii

Należy tutaj zamieścić krótki opis (z referencjami) do technologii użytych przy implementacji systemu.

Do implementacji systemu użyto języka JAVA w wersji ..., szczegółowy opis można znaleźć w [2]. Interfejs zaprojektowano w oparciu o HTML5 i CSS3 [3].

Omówienie kodów źródłowych

Kod źródłowy 4.1 przedstawia opisy poszczególnych metod interfejsu: `WSPodmiotRejestracjaIF`. Kompletne kody źródłowe znajdują się na płycie CD dołączonej do niniejszej pracy w katalogu `Kody` (patrz Dodatek A).

Kod źródłowy 4.1: Interfejs usługi Web Service: `WSPodmiotRejestracjaIF`.

```
package erejestracja.podmiot;
import java.rmi.RemoteException;
// Interfejs web serwisu dotyczącego obsługi podmiotów i rejestracji.
public interface WSPodmiotRejestracjaIF extends java.rmi.Remote{
// Pokazuje informacje o danym podmiocie.
// parametr: nrPeselRegon - numer PESEL podmiotu lub numer REGON firmy.
// return: Podmiot - obiekt transportowy: informacje o danym podmiocie.
public Podmiot pokazPodmiot(long nrPeselRegon) throws RemoteException;
// Dodaje nowy podmiot.
// parametr: nowyPodmiot - obiekt transportowy: informacje o nowym podmiocie.
// return: true - jeśli podmiot dodano, false - jeśli nie dodano.
public boolean dodajPodmiot(Podmiot nowyPodmiot) throws RemoteException;
// Usuwa dany podmiot.
// parametr: nrPeselRegon - numer PESEL osoby fizycznej lub numer REGON firmy.
// return: true - jeśli podmiot usunięto, false - jeśli nie usunięto.
public boolean usunPodmiot(long nrPeselRegon) throws RemoteException;
// Modyfikuje dany podmiot.
// parametr: podmiot - obiekt transportowy: informacje o modyfikowanym podmiocie.
// return: true - jeśli podmiot zmodyfikowano, false - jeśli nie zmodyfikowano.
public boolean modyfikujPodmiot(Podmiot podmiot) throws RemoteException;
// Pokazuje zarejestrowane podmioty na dany dowód rejestracyjny.
// parametr: nrDowoduRejestracyjnego - numer dowodu rejestracyjnego.
// return: PodmiotRejestracja[] - tablica obiektów transportowych: informacje o
// wszystkich zarejestrowanych podmiotach.
public PodmiotRejestracja[] pokazZarejestrowanePodmioty(
String nrDowoduRejestracyjnego) throws RemoteException;
// Nowa rejestracja podmiotu na dany dowód rejestracyjny.
```

```
// parametr: nrDowoduRejestracyjnego – numer dowodu rejestracyjnego.
// parametr: nrPeselRegon – numer PESEL podmiotu lub numer REGON firmy.
// parametr: czyWlasciciel – czy dany podmiot jest właścicielem pojazdu.
// return: true – jeśli zarejestrowano podmiot, false – jeśli nie zarejestrowano.
public boolean zarejestrujNowyPodmiot(String nrDowoduRejestracyjnego,
long nrPeselRegon, boolean czyWlasciciel) throws RemoteException;
// Usuwa wiązanie pomiędzy danym podmiotem, a dowodem rejestracyjnym.
// parametr: nrDowoduRejestracyjnego – numer dowodu rejestracyjnego.
// parametr: nrPeselRegon – numer PESEL podmiotu lub numer REGON firmy.
// return: true – jeśli podmiot wyrejestrowano, false – jeśli nie wyrejestrowano.
public boolean wyrejestrujPodmiot(String nrDowoduRejestracyjnego,
long nrPeselRegon) throws RemoteException;
```

Kod źródłowy 4.2 przedstawia procedurę przetwarzającą żądanie. Hasz utrwalany `%granulacja` wykorzystywany jest do komunikacji międzyprocesowej.

Kod źródłowy 4.2: Przetwarzanie żądania - procedura `process_req()`.

```
sub process_req(){
  my($r) = @_;
  $wyn = "";
  if ($r=~/get/i) {
        @reqest = split("_",$r);
        $zad = $reqest[0];
        $ts1 = $reqest[1];
        $ts2 = $reqest[2];
        @date1 = split(/\D/,$ts1);
        @date2 = split(/\D/,$ts2);
        print "odebralem:_$r";
        $wyn = $wyn."zadanie:_$zad\n";
        $wyn = $wyn."czas_od:_"."$date1[0]"."-"."$date1[1]"."-"."$date1[2]"."_"."$date1[3]"
        $wyn = $wyn."czas_do:_"."$date2[0]"."-"."$date2[1]"."-"."$date2[2]"."_"."$date2[3]"
        $wyn = $wyn.&sym_sens($ts1,$ts2);
        return $wyn;
  }
  if ($r=~/set gt/i) {
        @reqest = split("_",$r);
        $zad = $reqest[0];
        $ts1 = $reqest[1];
        $ts2 = $reqest[2];
        $gt = $reqest[2];
        dbmopen(%granulacja,"granulacja_baza",0644);
        $granulacja{"gt"}=$gt;
        dbmclose(%granulacja);
        $wyn = "\'GT\'_zmienione_na:_$gt";
  }
}
```

Instalacja i wdrożenie

W tym rozdziale należy omówić zawartość pakietu instalacyjnego oraz założenia co do środowiska, w którym realizowany system będzie instalowany. Należy przedstawić procedurę instalacji i wdrożenia systemu. Czynności instalacyjne powinny być szczegółowo rozpisane na kroki. Procedura wdrożenia powinna obejmować konfigurację platformy sprzętowej, OS (np. konfiguracje niezbędnych sterowników) oraz konfigurację wdrażanego systemu, m.in. tworzenia niezbędnych kont użytkowników. Procedura instalacji powinna prowadzić od stanu, w którym nie są zainstalowane żadne składniki systemu, do stanu w którym system jest gotowy do pracy i oczekuje na akcje typowego użytkownika.

Podsumowanie

W podsumowanie należy określić stan zakończonych prac projektowych i implementacyjnych. Zaznaczyć, które z zakładanych funkcjonalności systemu udało się zrealizować. Omówić aspekty pielęgnacji systemu w środowisku wdrożeniowym. Wskazać dalsze możliwe kierunki rozwoju systemu, np. dodawanie nowych komponentów realizujących nowe funkcje.

W podsumowaniu należy podkreślić nowatorskie rozwiązania zastosowane w projekcie i implementacji (niebanalne algorytmy, nowe technologie, itp.).

Bibliografia

[1] Apache solr technology. Web pages: https://lucene.apache.org/solr/guide/.

[2] Java technology. Web pages: http://www.oracle.com/technetwork/java/.

[3] B. Frain. *Responsive Web Design with HTML5 and CSS3*. Packt Publishing, 2012.

[4] G. McCluskey. Using java reflection. Web pages: https://www.oracle.com/technetwork/articles/java/javareflection-1536171.html.

Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.