

POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

KIERUNEK: Elektronika i telekomunikacja (EIT)
SPECJALNOŚĆ: Zastosowania inżynierii komputerowej
w technice (EZI)

PRACA DYPLOMOWA MAGISTERSKA

Projektowanie systemów zdalnego nauczania z
wykorzystaniem usług sieciowych w środowisku
MS .NET

Designing distance learning systems using Web
Services in MS .NET environment

AUTOR:
Michał Franc

PROWADZĄCY PRACĘ:
dr inż. Robert Wójcik PWr, I-6

OCENA PRACY:

Mamie i tacie :D

Spis treści

Wstęp	2
1 Cel i zakres pracy	4
1.1 Cel pracy	4
1.2 Zakres pracy	4
2 Kształcenie na odległość	5
2.1 Definicja kształcenia na odległość	5
2.2 Korzyści płynące z używania systemów kształcenia na odległość	6
2.3 Ograniczenia kształcenia na odległość	6
2.4 Klasyfikacja systemów kształcenia na odległość	7
2.4.1 Systemy do zarządzania kursami(CMS)	7
2.4.2 Systemy wspomagające proces nauczania (LMS)	8
3 Wybrane technologie realizacji systemów webowych	9
3.1 Formaty danych JSON oraz XML	9
3.2 Protokoły SOAP oraz REST	12
3.3 Usługi sieciowe	16
3.4 Framework Asp.Net MVC	17
3.5 Mapowanie obiektowo relacyjne - NHibernate	19
3.6 Testy jednostkowe	22
4 Projekt systemu zdalnego nauczania	25
4.1 Wymagania projektowe	25
4.1.1 Wymagania funkcjonalne	25
4.1.2 Wymagania нефункционалне	28
4.2 Wybrane diagramy przypadków użycia	30
4.3 Projekt bazy danych	31
4.3.1 Opis encji	31
4.3.2 Uproszczony model conceptualny (CDM) - struktura tabel i relacje	33
4.3.3 Model fizyczny bazy danych PDM	34
4.4 Architektura Systemu	35
4.4.1 Wybrane diagramy sekwencji funkcjonalności	37
4.4.2 Mechanizmy zabezpieczeń	40
5 Implementacja systemu zdalnego nauczania	42
5.1 Zewnętrzny hosting	42
5.2 Realizacja bazy danych	43
5.3 Realizacja usług sieciowych	43
5.4 Wykorzystane narzędzia	44

5.4.1	Mechanizm logowania zdarzeń - NLog	44
5.4.2	Testy jednostkowe - NUnit	46
5.4.3	Mapowanie obiektowo relacyjne - NHibernate	51
5.5	Realizacja aplikacji	53
5.5.1	Realizacja mechanizmów dostępu do bazy danych	53
5.5.2	Realizacja mechanizmów przetwarzania danych	55
5.5.3	Realizacja protokołu komunikacji	56
5.5.4	Realizacja mechanizmów zabezpieczeń	57
5.5.5	Realizacja zewnętrznego API - przykładowe zastosowanie	58
5.5.6	Realizacja mechanizmu dynamicznej zmiany serwera	59
5.6	Interfejs użytkownika - Moduły	60
5.6.1	Testy	60
5.6.2	Kursy	61
5.6.3	Dziennik ocen	62
5.6.4	Listy	63
6	Testowanie i ocena efektywności	64
6.1	Wybrane Testy Funkcjonalne	64
6.2	Testy Mechanizmów Zabezpieczeń	65
6.3	Analiza wydajności NHibernate-a w porównaniu do zwykłych zapytań SQL	66
6.4	Testy wydajności mechanizmów przetwarzania danych	67
6.4.1	Analiza zapytań generowanych przez NHibernate za pomocą NHPro- filea	68
6.4.2	Przykładowa optymalizacja zapytań	69
6.4.3	Testy obciążeniowe	70
6.5	Testy wydajności mechanizmów komunikacji sieciowej	71
6.5.1	Porównanie formatu Json , Xml	71
6.5.2	Badanie czasu odpowiedzi usług sieciowych	72
6.5.3	Testy konfiguracji rozłożenia usług sieciowych	73
6.6	Wnioski z testów i badań	74
7	Podsumowanie	75
	Bibilografia	76

Wstep

WSTEP WSTEP WSTEP WSTEP WSTEP WSTEP WSTEP WSTEP

Rozdział 1

Cel i zakres pracy

1.1 Cel pracy

Celem niniejszej pracy jest zaprojektowanie , zaimplementowanie oraz przetestowanie systemu informatycznego wspomagającego proces zdalnego nauczania oparty o technologię firmy Microsoft z wykorzystaniem mechanizmu usług sieciowych do komunikacji z bazą danych.

1.2 Zakres pracy

Analiza wymagań systemu zdalnego nauczania. Projekt systemu w oparciu o mechanizmy usług sieciowych. Implementacja w oparciu o Framework Asp.Net Mvc 3 z wykorzystaniem bazy danych MSSql. Testy wydajnościowe , funkcjonalne. Analiza proponowanego rozwiązania pod względem wydajnościowych oraz praktycznym. Zastosowanie nowoczesnych trendów wytwarzania oprogramowania.

Rozdział 2

Kształcenie na odległość

2.1 Definicja kształcenia na odległość

Nie ma jednoznacznej definicji w pełni określającej charakter zdalnego nauczania. W literaturze można znaleźć wiele zwrotów próbujących jednoznacznie określić to zagadnienie.

Terminy często używane w kontekście zdalnego nauczania to e-nauczanie, nauczanie internetowe, nauczanie rozproszone, nauczanie sieciowe, tele-nauczanie, wirtualne nauczanie, nauczanie wspomagane komputerowo, nauczanie webowe, oraz nauczanie na odległość. [4]

W wielu opracowaniach naukowych termin ten sprowadza się do postaci **"Kształcenie na odległość"**, w niniejszej pracy będę posługiwał się tym terminem.

Wszystkie zwroty łączy jedną wspólną cechą. Zakładają one, że podstawą kształcenia na odległość jest niebezpośredni proces uczenia się. Tzn proces w którym osoba ucząca się nie przebywa w bezpośrednim kontakcie z nauczycielem. Interesant taki korzysta z dostępnych zasobów zdalnie. Najczęściej za pomocą komputera. Jest to nowoczesny sposób przekazywania wiedzy, który staje się bardzo popularny na świecie. Wiele uczelni edukacyjnych oraz firm konsultingowych wdraża takie systemy.

Alan Clarke w swojej książce słusznie zauważa że : *E_l - learning ... Stanowi swoistą rewolucję, której skutki są często porównywalne do wpływu, który wcześniej na kształcenie wywarły wynalezienie druku i masowa produkcja książek ...* [9]

Do kształcenia na odległość możemy zaliczyć zarówno proces samego nauczania jak i proces wspomagania bądź wzbogacania nauczania. Do działań wspomagających możemy zaliczyć przede wszystkim. Sposób prezentacji materiałów oraz sposób dostarczania kolejnych materiałów bazując na wynikach uczącego się. Założenia te nie są częścią tej pracy dyplomowej. W niniejszej pracy skupimy się na stworzeniu platformy posiadającej mechanizmy wspierające proces zdalnego nauczania zarówno od strony uczącego się jak i nauczyciela odpowiedzialnego za tworzenie i nadzorowanie procesu nauczania.

2.2 Korzyści płynące z używania systemów kształcenia na odległość

Proces kształcenia do 19 wieku odbywał się w charakterze bezpośredniego kontaktu , najczęściej w specjalnie wydzielonej do tego sali, pomiędzy nauczycielem oraz uczniami. Rewolucja informatyczna oraz coraz większa znajomość technologii informatycznych wśród społeczeństwa spowodowały , że pod koniec 20 wieku wzrosło zainteresowanie systemami kształcenia na odległość. Kształcenie na odległość posiada wiele cech pozytywnych w porównaniu do starego systemu i podejścia do nauczania

Alan Clark [9] wymienia wiele korzyści płynących z zastosowania e-learningu. Zwraca uwagę m.in na:

- dostępność materiałów. Przeważnie otrzymujemy całodobowy dostęp do interesujących nas materiałów.
- asynchroniczny charakter procesu kształcenia tzn. uczący sam decyduje kiedy chce korzystać z materiałów.
- swoboda wyboru miejsca , czasu oraz tempa uczenia się.
- ułatwiony proces dystrybucji materiałów oraz informacji
- ułatwienie komunikacji z osobami z całego świata

Józef Bednarek [5] zwraca również uwagę na zwiększoną efektywność procesu nauczania. Poprzez zwiększony przedział zrozumienia tematu oraz konieczność większego zaangażowania uczących się. Ważnym aspektem przemawiającym na korzyść kształcenia na odległość jest również znaczna oszczędność czasu i zasobów poprzez ułatwiony dostęp do nauczyciela oraz materiału dydaktycznego.

2.3 Ograniczenia kształcenia na odległość

Każde rozwiązanie posiada negatywne cechy . Podobnie jest w przypadku zagadnienia kształcenia na odległość .Do najważniejszych problemów związanych z wdrażaniem kształcenia na odległość możemy zaliczyć.

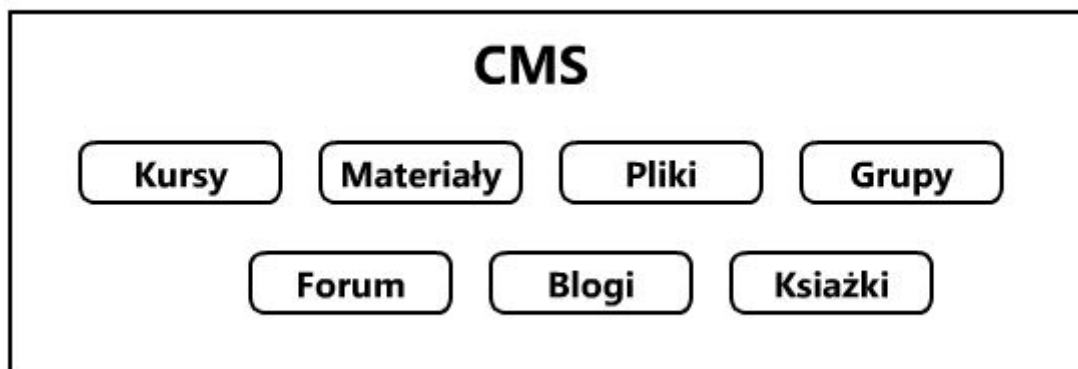
- konieczność przeszkolenia personelu dydaktycznego oraz uczniów.
- wdrożenie kosztownego systemu informatycznego obsługującego proces nauczania.
- wymagana większe zaangażowanie i samodzielna inicjatywa od uczącego się.
- konieczność dopasowania obecnego materiału dydaktycznego do nowych standardów.
- brak typowych dla bezpośredniego procesu nauczania mechanizmów interakcji społecznych mogących jedynie zajść w momencie gdy nauczyciel oraz uczniowie są w bezpośrednim kontakcie.
- anonimowość nauczyciela.

2.4 Klasyfikacja systemów kształcenia na odległość

A. Clark [9] klasyfikuje systemy E-learningowe w obrębie dwóch podstawowych klas.

- Systemy do zarządzania kursami(CMS)
- Systemy wspomagające proces nauczania (LMS)

2.4.1 Systemy do zarządzania kursami(CMS)



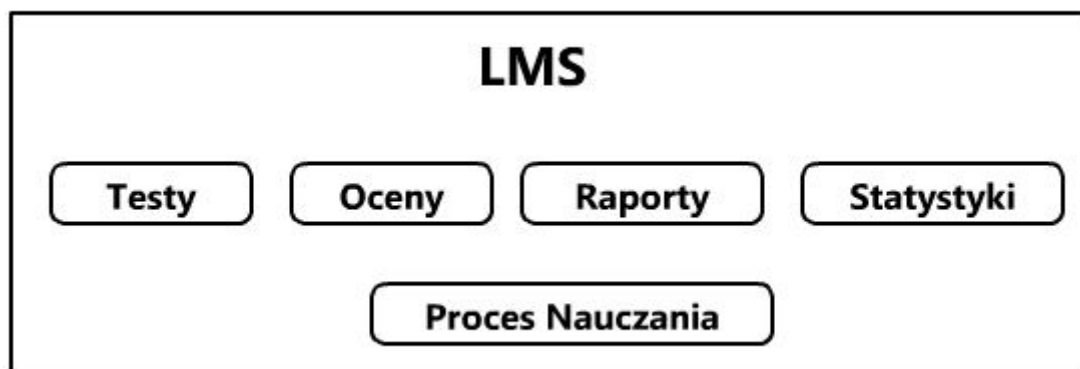
Rys.1 Podstawowe funkcje systemu klasy CMS

Systemy klasy CMS (Content Management System) są to aplikacjami dostarczającymi funkcje pozwalających zarządzać materiałami wspomagającymi proces nauczania , poprzez dostarczanie funkcji ułatwiających tworzenie oraz udostępnianie nowych materiałów. W systemach takich możemy wyszczególnić kilka podstawowych funkcji

- katalogowanie dostępnych materiałów dydaktycznych
- system administracji pozwalający zarządzać procesem udostępniania materiałów
- mechanizmy wspomagające proces tworzenia oraz publikowania nowych materiałów

Systemy takie zazwyczaj sprowadzają się do postaci katalogów udostępniających materiały.

2.4.2 Systemy wspomagające proces nauczania (LMS)



Rys.2 Podstawowe funkcje systemu klasy LMS

Systemy LMS są bardziej wyspecjalizowanymi systemami wspomagającymi proces nauczania. Poprzez dostarczanie niektórych funkcjonalności takich jak:

- moduł testów pozwalających zdalnie ewaluować umiejętności kursanta
- moduł ocen pozwalający dokumentować postęp oraz wyniki kursanta
- moduł wspomagający kolaborację oraz komunikację pomiędzy kursantami i nauczycielami
- moduł statystyk oraz raportów pozwalających ewaluować skuteczność procesu kształcenia
- moduł autonomicznego agenta podającego np sugestie kursantowi kierując oraz dostosowując jego tok nauczania

W niniejszej pracy stworzony zostanie system łączący cechy obu systemów.

Rozdział 3

Wybrane technologie realizacji systemów webowych

3.1 Formaty danych JSON oraz XML

JSON

JSON (JavaScript Object Notation) jest formatem danych wykorzystywanym przy transferze danych po sieci. Jest on alternatywą do standardu XML. Jest to nowy format danych opisany w 2006 roku w publikacji RFC4627[?] . Jest to lekki format z bardzo prostą semantyką. Zbiór danych składa się z pary ciągów znaków : nazwa , wartość.



Rys. Uproszczony Schemat formatu JSON

Semantyka formatu pozwala zdefiniować podstawowe typy danych takie jak : [?]

- Obiekt
- Tablicę obiektów
- Ciąg Znaków
- Liczbę
- Wartość bitową

Cechy formatu JSON

- ułatwione przetwarzanie formatu do obiektów języka Javascript
- lekki prosty format
- szybki proces przetwarzania
- łatwy w modyfikacji

XML

XML jest jednym z najbardziej rozpowszechnionych formatów danych. Wykorzystywany jest nie tylko jako nośnik danych ale również jest popularny w wielu jako format plików konfiguracyjnych. Niektóre skomplikowane silniki wykorzystują format XML do definiowania , scenariuszy testowych , interakcji w systemie.

Format ten powstał na bazie formatu SGML. Standard XML opisuje zarówno zestaw znaczników pozwalających opisać dokumenty jak i sposób parsowania formatu.[?]



Rys. Schemat formatu XML

XML składa się z ze znaczników zamykających oraz otwierających zwanych tagami , wewnątrz których znajduje się zawartość danego elementu. W formacie tym istnieje możliwość definiowania atrybutów opisujących dany tag. Tworzenie drzewiastej hierarchii elementów realizowane jest za pomocą prostego mechanizmu zamykania tagów w tagach.

Cechy formatu XML

- bardzo duże wsparcie narzędzi i zgodność z wieloma systemami na rynku
- duże bezpieczeństwo wiadomości
- możliwość definiowania dodatkowych atrybutów
- pochodne XML-a takie jak XML Schema , XSLT wzbogacające format o dodatkowe funkcjonalności

Porównanie JSON oraz XML

W momencie wejścia na rynek usług sieciowych , były one głównie wykorzystywane w świecie biznesowym. Rynek Enterprise zaadoptował na początku format XML i protokół oparty na tym formacie SOAP. XML był idealnym formatem ponieważ był już dobrze znany i miał spore wsparcie narzędzi oraz języków.

Wzrost zainteresowania formatem JSON zaczął się w 2006 roku wraz z powstaniem pierwszego oficjalnego opisu oraz wraz z adopcją tego formatu przez takie firmy jak Google czy Yahoo.

Format XML charakteryzuje większym poziomem bezpieczeństwa dzięki bardziej stabilnej i mocno typowanej semantyce. W przypadku formatu JSON istnieje pewne ryzyko pozwalające wstrzyknąć niepożądany kod ponieważ JSON wykorzystuje funkcje eval.

W porównaniu do XML-a format JSON charakteryzuje się mniejszą wagą przesyłanego ładunku , jest lżejszy i czytelniejszy dla odbiorcy.

<pre> { "ID":1, "Role":"Author", "name":"Franc", "first-name":"Michal", "age":25, "hobbies":{ "reading", "programming", { "sports":{ "volley-ball", "football" } } }, "address":{ "City":"Wroclaw", "Street":"Mosiezna", "NR":"19/22" } } </pre>	<pre> <?xml version="1.0" encoding="UTF-8" ?> - <json> <ID>1</ID> <Role>Author</Role> <name>Franc</name> <first-name>Michal</first-name> <age>25</age> <hobbies>reading</hobbies> <hobbies>programming</hobbies> - <hobbies> <sports>volley-ball</sports> <sports>football</sports> </hobbies> - <address> <City>Wroclaw</City> <Street>Mosiezna</Street> <NR>19/22</NR> </address> </json> </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Rys. Porównanie standardów (JSON , XML).

W pracy magisterskiej zastosowałem zarówno format JSON jak i XML. Dane używane przez serwis przesyłane są w formacie XML natomiast dane wystawione w API przesyłane są w formacie JSON.

3.2 Protokoły SOAP oraz REST

Współczesne aplikacje webowe udostępniające usługi sieciowe zbudowane są w większości na bazie protokołów REST bądź SOAP.

SOAP

Protokół SOAP jest protokołem transmisji danych wykorzystującym format XML. Może również wykorzystywać inne formaty jednak XML jest najpopularniejszym. Ramki wiadomości przesyłane są standardowo za pomocą protokołu HTTP przy użyciu komend GET, POST, DELETE. Istnieje też możliwość wykorzystania protokołu RPC.

Standard SOAP opisuje odpowiednią ustandaryzowaną strukturę wiadomości przesyłanej do serwera.

Wiadomość składa się z ramek.

- Envelope
- Header
- Body

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IProfileService/GetByName</a:Action>
    <a:MessageID>urn:uuid:675b3f2e-97a9-48a8-872b-e001be035a39</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://lam-pc:8888/ProfileService.svc</a:To>
  </s:Header>
  <s:Body>
    <GetByName xmlns="http://tempuri.org/">
      <userName>user12</userName>
    </GetByName>
  </s:Body>
</s:Envelope>
```

Rys. Przykładowa wiadomość SOAP

Element Header zawiera nagłówek informacji. Na wyżej zamieszczonym przykładzie widać m.in. adres do którego kierowana jest wiadomość oraz wartość uuid przydzielaną każdej wiadomości. Zapewnia to unikatowość wiadomości.

Element Body zawiera nazwę funkcji udostępnianej przez usługę sieciową oraz jej parametry w tym przypadku jest to jedna parametrowa funkcja GetByName z parametrem userName.


```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IProfileService/GetByName</a:Action>
    <a:MessageID>urn:uuid:675b3f2e-97a9-48a8-872b-e001be035a39</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://lam-pc:8888/ProfileService.svc</a:To>
  </s:Header>
  <s:Body>
    <GetByName xmlns="http://tempuri.org/">
      <userName>user12</userName>
    </GetByName>
  </s:Body>
</s:Envelope>
```

Rys. Przykładowa wiadomość zwrotna

Można zauważyć charakterystyczny uuid wiadomości który jest taki sam jak w przypadku wiadomości wysłanej do serwera. Na tej podstawie server usług sieciowych wie jak dopasować wiadomości. Jak widzimy w segmencie body został przesłany wynik wykonania usługi.

link do specyfikacji - <http://www.w3.org/TR/soap/>

Zalety SOAP

- stała składnia , mocne typowanie
- dużo narzędzi na rynku
- dojrzałość standardu

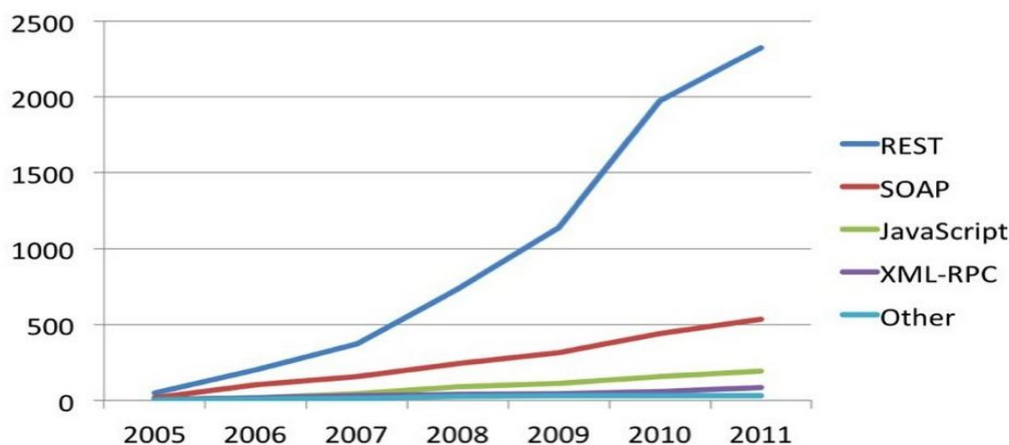
Wady SOAP

- przesyłanie nagłówek pożera dużo miejsca
- duży poziom komplikacji

REST

Soap jest skomplikowany i potrzebuje wielu targów oraz znaczników do przesyłania danych. W środowisku nowych małych firm powstał protokół REST jego popularność zaczęła znacząco rosnąć wraz pojawieniem się TWITTER API usłudze bazującej na formacie danych JSON.

Badania z Maja 2011 roku przeprowadzone przez serwis ProgrammableWeb pokazują, że prawie 73% usług sieciowych wystawionych w formie API obsługiwane jest na podstawie protokołu typu REST.



Distribution of API protocols and styles

Based on directory of 3,200 web APIs listed at ProgrammableWeb, May 2011

[?]

Architektura typu REST opiera się na jednym ważnym założeniu. Pobieranie oraz modyfikowanie obiektów jest ściśle powiązane z adresem URL. Dzięki zastosowaniu takiego rozwiązania można pobierać np zawartość danych z bazy za pomocą komend protokołu HTTP. Pobieranie można realizować np za pomocą komendy GET. Modyfikacje bądź usuwanie danych za pomocą komendy POST.

GET <http://codedashservices.mfranc.com/CourseService.svc/json/Get?id=16>
HTTP/1.1

Rys. Przykładowa wiadomość protokołu REST

Komenda wysyłana do serwera HTTP zawiera w sobie nazwę komendy GET oraz Adres. Funkcja oraz parametry zawarte są w treści adresu URL. W tym przypadku segment `Get?id=16` zawiera nazwę funkcji oraz parametr `id=16`

```
{
  {
    "CourseType": {
      "ID": 1,
      "TypeName": "csharp"
    },
    "CreationDate": "\/Date(1305319615000+0200)\/",
    "ID": 35,
    "Logo": "csharp",
    "Name": "C# Basic",
    "ShortDescription": "C# Basics Course"
  }
}
```

Rys. Przykładowa wiadomość zwrotna protokołu REST

W tym przypadku używamy formatu danych json do zwrócenia danych z bazy danych wystawionej za usługą sieciową.

Zalety REST

- prostota i lekkość ,nie ma potrzeby wysyłania dodatkowych znaczników
- czytelny
- prosty generacji i parsowaniu

Wady REST

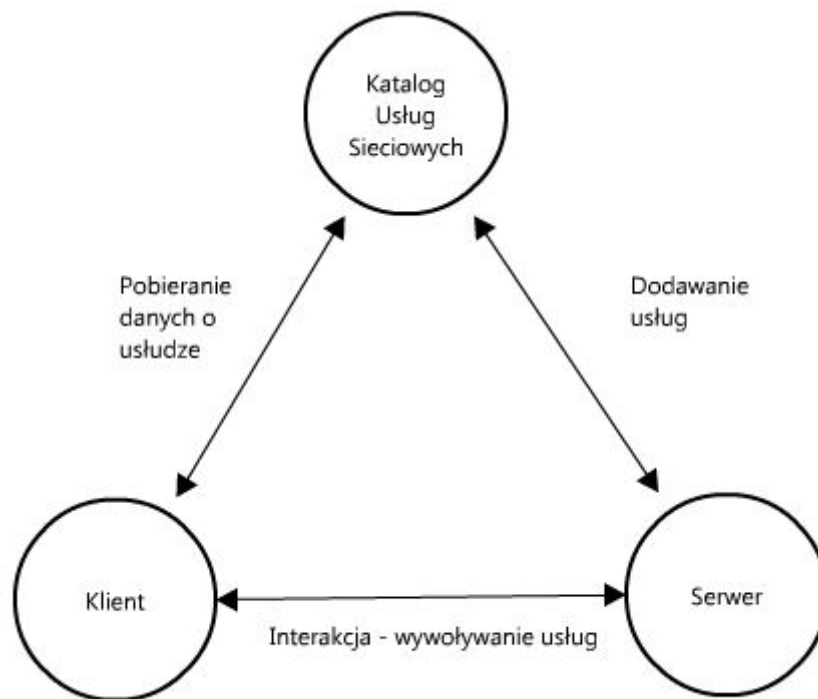
- prostota powoduje , że nie można tworzyć skompilowanych wiadomości

3.3 Usługi sieciowe

Usługi sieciowe są technologią pozwalającą inicjować komunikację z pomiędzy dwoma urządzeniami podłączonymi do sieci komputerowej.

Zgodnie z definicją z dokumentu <http://www.w3.org/TR/2002/WD-ws-arch-20021114/>

[Definition: A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols.]



Rys. Uproszczona architektura usług sieciowych

Klient rozpoczynając komunikację wysyła pierw zapytanie do katalogu usług sieciowych który zawiera informacje na temat dostępnych usług ich nazwy parametry wejściowe oraz opis. Klient wybiera z katalogu interesującą go usługę następnie tworzy odpowiednio sformatowaną wiadomość najczęściej w formacie xml i wysyła do serwera. Serwer interpretuje wiadomość wywołuje odpowiednią funkcję i zwraca wynik.

do opisu usług znajdujących się w katalogu używany jest format WSDL (Web Service Description Language) który jest rozszerzeniem XML-a

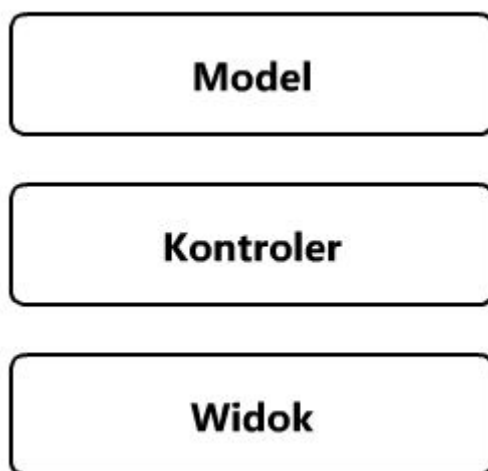
3.4 Framework Asp.Net MVC

Asp.Net Mvc 3 jest platformą stworzoną przez firmę Microsoft służącą do tworzenia aplikacji webowych. Działa jako nakładka na platformę Asp.Net. Jest odpowiedzią na nowe trendy zdobywające coraz to większą popularność w środowisku programistów webowych. Cały Framework oparty jest na wzorcu projektowym MVC.

Wraz z ewolucją aplikacji webowych i ich poziomu skomplikowania pojawiały się nowe podejścia oraz sposoby wytwarzania aplikacji webowych pozwalające tworzyć aplikacje. Jednym z takich podejść jest wykorzystanie wzorca Model View Controller. Pierwszy opis wzorca można znaleźć w dokumencie z 1979[?].

Od 2004 roku wzorzec ten zaczął zdobywać dużą popularność wraz z pojawieniem się nowych platform programistycznych : Ruby On Rails oraz Django.

Wprowadza on podział aplikacji na trzy oddzielne warstwy : Model , Widok , Kontroler.



Rys. Koncepcja MVC

Model

Reprezentuje warstwę bazy danych. Dostęp do modelu jest jedynie możliwy z poziomu kontrolera , który w wielu aplikacjach do komunikacji z modelem korzysta również z warstwy usług. Warstwa ta wystawia metody zdefiniowane w kontrakcie , z których korzysta kontroler.

Widok

Reprezentuje warstwę dostępną dla użytkownika systemu. Buduje się na podstawie modelu przekazanego przez kontroler.

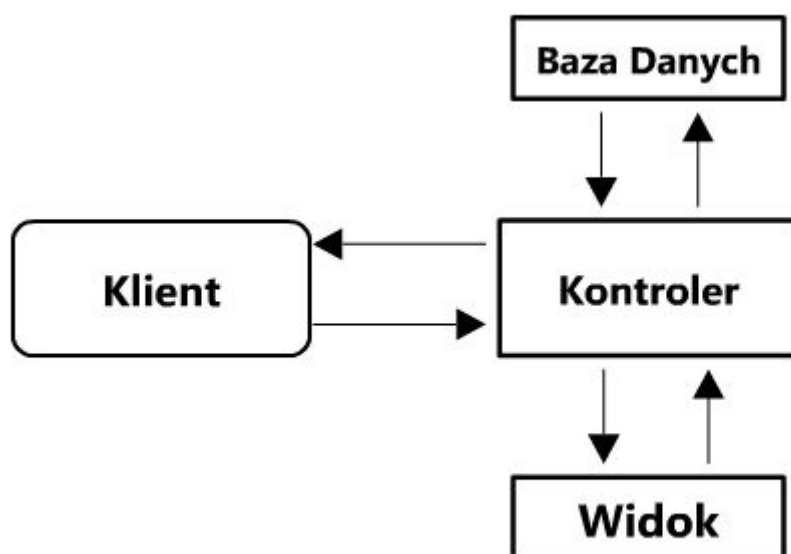
Kontroler

Warstwa odpowiedzialna za sterowanie przepływem danych i przetwarzaniem tych danych by mogły być wyświetlone w warstwie widoku.

Zastosowanie wzorca MVC przy projektowaniu aplikacji webowej wymaga większego nakładu pracy w początkowej fazie projektu. Wymierne korzyści ze stosowania tego

wzorca zaczynają być odczuwane dopiero w późniejszych etapach życia projektu. Przede wszystkim zastosowane konwencje i jawna separacja odpowiedzialności na trzy warstwy pozwala oddzielić od siebie logikę biznesową dostępną z poziomu klienta od logiki obsługującej dostęp do bazy danych. Jest to bardzo ważne ponieważ zmiany zachodzące w warstwie modelu tzn bazy danych nie powinny powodować zmian w warstwie widoku. Dzięki takiemu rozdziałowi powstaje lepszy kod , łatwiejszy w rozbudowanie oraz utrzymaniu. Dodatkowo projekt jest bardziej czytelny. Programista wiedzący że projekt został stworzony w oparciu o MVC automatycznie wie gdzie szukać poszczególnych implementacji systemu w celu przeprowadzenia modyfikacji.

Opis komunikacji



Rys. Schemat komunikacji Mvc.

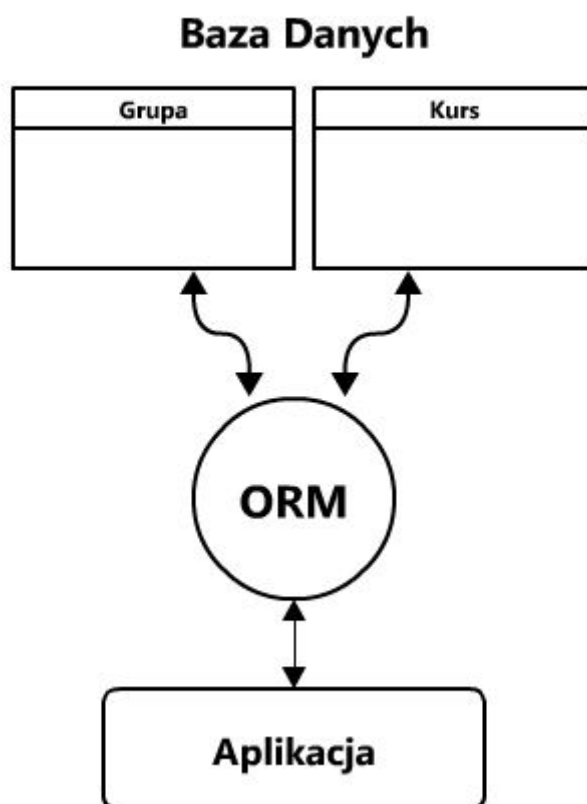
Klient realizuje zapytanie , które przechwytuje kontroler. w przypadku aplikacji webowej zapytanie będzie po prostu zwykłym odniesieniem się do określonego adresu url. Kontroler przejmuje żądanie. Jeżeli wygenerowanie odpowiedniego widoku nie wymaga pobrania danych z bazy danych. Kontroler pobiera dany widok i przekazuje go klientowi w swojej odpowiedzi na zadanie. Jeżeli widok wymaga pobrania danych z bazy danych , realizowane jest połączenie z modelem pobranie danych i wygenerowanie widoku z danymi i przekazanie go w wiadomości zwrotnej.

3.5 Mapowanie obiektowo relacyjne - NHibernate

Bazy danych są najważniejszą częścią systemu informatycznego. Stanowi większość implementacji aplikacji po stronie serwerowej. Przy tworzeniu systemów informatycznych logika dostępu do bazy danych pochłania bardzo dużo czasu. Dodatkowo jest podatna na błędy. Bezpośrednie tworzenie zapytań stało się zbyt kosztowne oraz trudne w utrzymaniu. Z rozwiązaniem takim wiąże się również inny problem mianowicie dochodzi do niekompatybilności zależności pomiędzy obiektami między systemami relacyjna bazodanowymi a systemami opartymi na dziedziczeniu i kompozycji klas znajdującymi się w języku programowania.

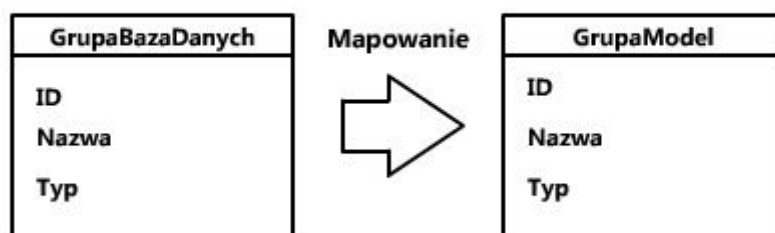
By ułatwić proces tworzenia kodu coraz więcej firm wykorzystuje specjalne biblioteki wspomagające proces tworzenia warstwy dostępu do danych. Nazywane one są Obiektowo relacyjnymi maopermai. Na rynku dostępnych jest wiele amperów najpopularniejszymi w środowisku jedna sa. Tworzony przez firmę Microsoft Entity Framework oraz NHibernate który jest implementacja frameworka Hibernate z technologii java na platformę .Net.

Dzięki zastosowaniu orma można wprowadzać bardzo szybko zmiany oraz w dużo łatwiejszy sposób wykonywać odpowiednie zapytania na bazie danych nie przejmując się tak naprawdę warstwa bazodanową. Dla programisty oporującego na tej warstwie cała komunikacja jest schowana pod interfejsami. Dzięki temu programista może skupić się na implementacji logiki oszczędzając czas na implementowaniu dostępu do bazy danych. Minusem takiego rozwiązania jest mniej wydajny proces pobierania danych. Problem ten można zniwelować poprzez odpowiednie sprofilowanie aplikacji i wyznaczanie najlepszych części systemu wymagających optymalizacji. W tym przypadku profilowane są zapytania SQL. Zapytania wymagające optymalizacji można zamienić na zwykłe zapytani sqlowe. Dzięki takiemu zabiegowi oszczędza się czas oraz fundusze przeznaczone na projekt.



Rysunek przedstawiający warstwę dostępu do danych
realizowaną za pomocą mappera obiektowo relacyjnego

Proces mapowania sprowadza się do określenia mapowań poprzez wskazanie narzędziu , które pole z bazy danych ma być połączone z obiektem wykorzystywanym w aplikacji. w ten sposób tworzone są specjalne klasy pośredniczące w komunikacji pomiędzy bazą danych a aplikacją.



Rysunek przedstawiający proces mapowania.

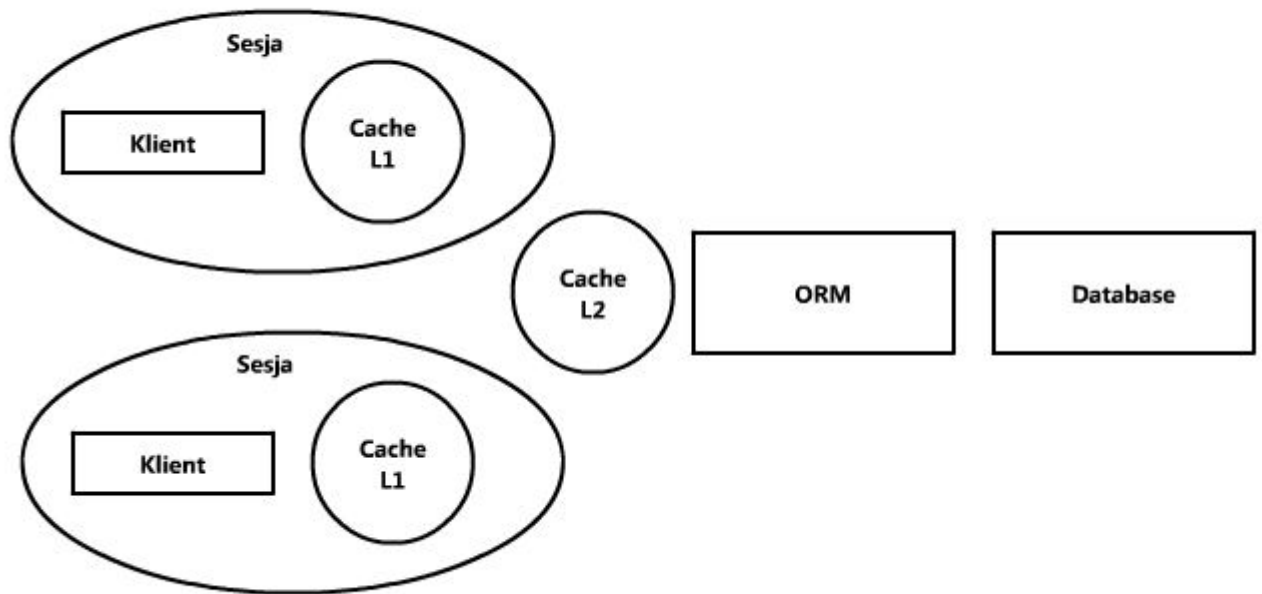
W NHibernate można mapować za pomocą plików konfiguracyjnych xml. Jest to wygodne podejście jednakże podatne na błędy i nieczytelne. Dlatego często stosuje się rozwiązanie typu FluentNHibernate bibliotekę , która pozwala generować pliki XML na podstawie kodu napisanego w języku platformy .Net. Mapowanie takie staje się bardziej czytelne.

NHibernate dodatkowo wspiera mechanizm cachowania pierwszego oraz drugiego poziomu dzięki którym proces pobierania danych jest bardziej wydajny.

Cache pierwszego poziomu trzyma wartości z tabel i w momencie gdy nastąpi kolejne odwołanie do danych sprawdzany jest cache po nazwie tabeli oraz numerze ID. Jeżeli obiekt znajduje się na poziomie cache pierwszego poziomu jest pobierany z tego cache bez konieczności wykonywania zapytania do bazy danych.

Cache pierwszego poziomu dostępny jest w obrębie tylko jednej sesji. Tzn dla każdego klienta zaczynającego pracę z systemem tworzona jest sesja każdy klient posiada swój własny cache z którego korzysta. By współ dzielić cache pomiędzy różnymi klientami i ich sesjami stosuje się cache drugiego poziomu.

Odpowiednie skonfigurowanie obu sposobów cachowania pozwala zoptymalizować proces zapytań do bazy danych.



Rysunek przedstawiający poziomy cachowania

3.6 Testy jednostkowe

Branża wytwarzania oprogramowania przechodzi przez okres ciągłych zmian szukając rozwiązań oraz metodyk najlepiej spełniających swoje zadanie. Metodyka Waterflass przejęta z innych dziedzin inżynierskich takich jak Automatyka Inżynieria budowała okazała się nie efektywna. Środowisko programistów zaczęło poszukiwać innych rozwiązań i w ten sposób powstała metodyka Extreme Programming. Która zakłada nie jeden okres planowania ale wiele okresów planowania następujących cyklicznie. By można było taką metodykę tworzyć oprogramowanie zaczęto używać wiele narzędzi wspomagających ten proces. Jednym z takich narzędzi stały się testy jednostkowe.

Testowanie od początku było dziedziną którą była przeprowadzana przez dział testów. Jednakże środowisko programistyczne odkryło że można przeprowadzać testy w kodzie podczas tworzenia oprogramowania tym samym polepszając jakość twórczego kodu. Testy jednostkowe są formą kodu który testuje inny kod. Jest to kod napisany w ten sposób że wykorzystuje stworzony kod i testuje pewne założenia.



Procedura przeprowadzania testu jednostkowego

Pozwalają ograniczyć ilość błędów. Powodują jednak że marnujemy trochę czasu a odpowiednio zdefiniowanie testów ale tak naprawdę programista i tak spędziłby podobną ilość czasu przeklikując np. aplikację.

```
[Test]
public void Can_get_all_courses_signatures()
{
    #region Arrange
    #endregion

    #region Act

    var courseSignatures = new CourseService().GetAllSignatures();

    #endregion

    #region Assert
    Assert.That(courseSignatures, Is.Not.Null);
    Assert.That(courseSignatures.Count, Is.EqualTo(3));
    Assert.That(courseSignatures.First(), Is.InstanceOf(typeof(CourseSignatureDto)));
    #endregion
}
```

Przykładowy test jednostkowy

Pisanie testów jednostkowych nie dość że daje nam możliwość przetestowania najszej aplikacji ale również wymusza się trzymanie lepszych wytycznych i wzorców projektowych ponieważ nie wszystkie dane da się łatwo zrobić.

Mockowanie oraz Stubowanie obiektów

Aplikacje biznesowe mają często bardzo skomplikowaną logikę wykorzystującą wiele modułów oraz zależności. Piszac testy jednostkowe powinno testować się tylko pojedyncze funkcjonalności. W momencie gdy np. nasza aplikacja do przeprowadzenia pewnej czynności wykorzystuje zewnętrzne zasoby np. dane z bazy danych w momencie gdy przeprowadzamy taki test musimy wykorzystać bazy danych. W tym momencie nie testujemy tylko jednej funkcjonalności ale również same połączenie bądź poprawną konfigurację połączenia.

By móc testować aplikację wraz z takimi zależnościami musimy bądź przeprowadzać testy operując na prawdziwej bazie danych bądź emulować bazy danych. Oba te podejścia wiążą się z pewnymi ograniczeniami i narzucają wiele czynności.

Innym podejściem zgoła innym jest tworzenie specjalnych klas udających połączenie z bazą danych. Klasy takie nazywamy Stubami bądź mockami w zależności od tego jak są konstruowane.

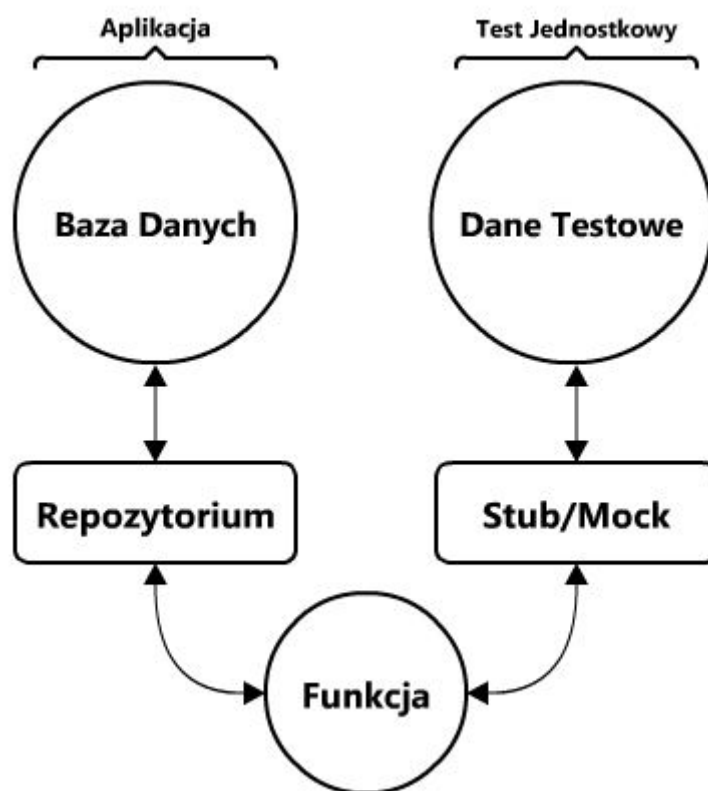
Stub

Stub jest specjalnym obiektem imitującym klasę. W ten sposób że posiada te same funkcje oraz testowe dane jak testowana klasa. W przypadku naszej komunikacji do bazy danych. Imitujący obiekt posiadałby w miejscu funkcji pobierającej dane funkcje zwracające stałą kolekcję obiektów.

Mock

Działa podobnie jak Stub. Różnica jednak polega na tym, że obiekty typu mock działają na prawdziwej klasie podczepiając się tylko pod wybrane funkcjonalności. Tzn. w naszym przypadku pobierania danych z bazy danych. Będziemy wykorzystywać naszą normalną klasę podniejając tylko jej funkcje dostępu do bazy danych.

Zastosowanie



Zastosowanie

Rozdział 4

Projekt systemu zdalnego nauczania

4.1 Wymagania projektowe

4.1.1 Wymagania funkcjonalne

Tworzony system łączy w sobie pewne wybrane cechy systemów klasy CMS oraz LMS. Do najważniejszych funkcji należy możliwość edytowania/publikowania kursów , testów , materiałów zdalnego nauczania oraz mechanizm dzienników ocen. Aplikacja wspiera proces tworzenia materiałów oraz testów poprzez prosty ale funkcjonalny system edycji.

Do poszczególnych funkcjonalności możemy zaliczyć.

Tworzenie użytkowników oraz przydzielanie uprawnień

System posiada mechanizm pozwalający zarządzać użytkownikami aplikacji wraz z możliwością przydzielania im odpowiednich poziomów uprawnień. Dostępny jest panel administracyjny zawierający listę użytkowników oraz udostępniający funkcje :

- Dodaj/Usuń użytkownika
- Aktywuj/Dezaktywuj użytkownika
- Modyfikacja uprawnień
- Dodaj/Usuń użytkownika z grupy.

Usunięcie użytkownika równoznaczne jest z usunięciem wszystkich danych powiązanych z tym użytkownikiem. Szczególnym przypadkiem usuwania użytkownika jest usuwanie użytkownika posiadającego uprawnienia umożliwiające tworzenie kursów oraz materiałów e-learningowych. W przypadku usunięcia takiego użytkownika kursy przechodzą na własność administratora systemu , który może przydzielić uprawnienia autorskie innemu użytkownikowi.

Zarządzanie kursami

Każdy użytkownik posiadający uprawnienia autorskie oraz administrator może stworzyć jeden lub więcej kursów które może edytować. Kurs opisany jest parametrami

- Nazwa
- Krótki opis

- Logo
- Rodzaj kursu

Każdy kurs posiada swoją odrębną listę materiałów nauczania , testów , mechanizm wymiany krótkich wiadomości (Shoutbox) oraz pojedynczą grupę użytkowników przynależących do danego kursu. Autor ma możliwość edycji tylko i wyłącznie kursów , którymi zarządza. Administrator może modyfikować każdy kurs znajdujący się w systemie.

Zarządzanie materiałami nauczania

Materiały nauczania są integralną częścią kursu. Każdy kurs może posiadać jeden lub więcej materiałów nauczania. W skład materiału nauczania wchodzi parametry opisowe pozwalające sklasyfikować dany materiał :

- Poziom materiału Początkujący , Średnio-Zaawansowany , Zaawansowany
- Opis
- Logo

Uprawnienia do edycji materiałów posiada administrator oraz autor kursu , do którego należy dany materiał.

Materiały nauczania - proces nauczania

Materiał nauczania prócz parametrów opisowych podzielony jest na trzy obszary.

Obszar informacyjny

Zawiera segmenty :

- Informacyjny - parametry kursu
- Opisowy - opisuje kontekst kursu
- Celów - opisuje cele jakie osiągnie kursant biorący udział w kursie

Obszar nauczania

W skład obszaru nauczania wchodzi sekcje zawierające materiały z zawartością merytoryczną , z której korzysta kursant. Materiał nauczania może posiadać jeden lub więcej sekcji. Każda sekcja opisana jest tytułem oraz polem zawartości , w którym umieszczamy wiedzę oraz materiały , które chcemy przekazać w procesie nauczania.

Obszar podsumowania

Zawiera segmenty

- Podsumowania - szybkie podsumowanie zdobytej wiedzy i najważniejszych rzeczy , które zapamiętują kursant.
- Więcej informacji - posiada zbiór dodatkowych materiałów oraz linków do nich

- Testów - posiada test stworzony na potrzeby materiału nauczania pozwalający sprawdzić wiedzę kursanta

Mechanizm nauczania stworzony jest w sposób liniowy. Kursant po kolei odkrywa kolejne segmenty oraz sekcje wchodzące w skład materiału nauczania. Na koniec kursant może sprawdzić swoją wiedzę wykonując test.

Zarządzanie testami

Aplikacja pozwala tworzyć testy będące istotnym elementem procesu nauczania. Testy mogą być powiązane z materiałem nauczania. Możliwość tworzenia oraz edycji testów posiada Administrator oraz Autor kursu posiadający odpowiednie uprawnienia.

Mechanizm rozwiązywania oraz sprawdzani testów

Kursant po skończeniu procesu nauczania może przystąpić do rozwiązywania testu. Test składa się z segmentu zawierającego treść pytania oraz segmentu odpowiedzi. Po rozwiązaniu testu wyświetlana jest ocena oraz wiadomość informująca o zaliczeniu testu. Następnie ocena zapisywana jest do dzienniczka ucznia

Zarządzanie ocenami

Mazdy kursant może przeglądać listę swoich ocen otrzymanych po rozwiązaniu testu. Kursant ma możliwość także automatycznego wyliczenia średniej ocen z danego kursu.

Filtrowanie kursów ora z materiałów

Kursant ma możliwość filtrowania kursów na podstawie typu kursu. Może również wyświetlić listę kursów do , których jest już zapisany. Kursant ma możliwość filtrowania materiałów nauczania na podstawie poziomu trudności oraz może sortować materiały na podstawie jego nazwy.

4.1.2 Wymagania niefunkcjonalne

Bardzo dobra jakość kodu i projektu

Platforma stworzona zostanie zgodnie z nowoczesnymi trendami oraz zasadami dobrego projektowania aplikacji tak by w przyszłości proces modyfikacji i utrzymania kodu był jak najmniej.

Bezpieczeństwo aplikacji

Dostęp do poszczególnych funkcjonalności systemu jest ograniczony w obrębie trzech ról przydzielanych do poszczególnych użytkowników platformy. Wyróżniamy trzy role:

Administrator

Administrator reprezentuje użytkownika odpowiedzialnego za zarządzanie całym systemem zdalnego nauczania.

Autor

Autor jest użytkownikiem posiadającym możliwość tworzenia nowych , edytowania kursów , materiałów nauczania oraz testów.

Kursant

Kursant jest podstawowym użytkownikiem posiadającym możliwość przeglądania zasobów aplikacji i interakcji z systemem bez możliwości wpływania na zawartość systemu. Kursanci posiadają możliwość umieszczania krótkich wiadomości tekstowych w module ShoutBox.

Mechanizm rejestracji , logowania zostanie zrealizowany za pomocą Frameworka Asp.Net Membership zapewniającego podstawowe funkcjonalności. W tym celu powstanie druga baza danych zawierająca dane logowania powiązana z główną bazą danych.

Bezpieczeństwo transmisji danych

Platforma komunikuje się z warstwą bazodanową za pomocą usług sieciowych. Transmisja danych przebiegała będzie na dwóch poziomach zabezpieczeń. Pierwszy poziom bez szyfrowania i bez zabezpieczeń będzie służył do przesyłania danych zawierających jedynie dane na temat kursów , testów oraz materiałów nauczania. Dane przesyłane na tym poziomie będą w przesyłane w niezaszyfrowanej postaci . Dane poufne takie jak dane logowania , hasła będą przesyłane bardziej bezpiecznym sposobem transmisji z użyciem rozszerzenia WS-Security protokołu SOAP.

Komunikacja za pomocą usług sieciowych

Dostęp do bazy danych wystawiony będzie za warstwą usług sieciowych zrealizowanych w technologii WCF. Poszczególne funkcjonalności bazy danych wystawione będą w formie sześciu usług sieciowych :

- Kursy
- Testy
- Profile

- Materiały nauczania
- Grupy
- Dziennik

Dodatkowo część funkcji zostanie udostępniona w formacie JSON używając protokołu REST. Dzięki temu inne aplikacje nie powiązane z platformą e-learningową mogą uzyskać dostęp do danych. By zaprezentować tę funkcjonalność został stworzony dodatek do popularnej platformy CMS WordPress pozwalający wyświetlić kursy danego autora na blogu.

Mechanizm dynamicznego odnajdywania serwera bazodanowego

By zwiększyć niezawodność systemu stworzony zostanie system, który w momencie problemów z komunikacją z podstawowym serwerem bazy danych przełączy system na zapasową bazę danych. Co jakiś czas generowane będzie zapytanie o dostępność serwera i w momencie braku odpowiedzi bądź błędu wysyłanego protokołem soap nastąpi automatyczne przekierowanie na endpointy serwera zapasowego.

Tworzenie kopii zapasowych bazy danych

Każdego dnia o określonej porze będzie tworzona kopia zapasowa bazy danych. Jednorazowo trzymany będzie siedem kopii.

Automatyczny mechanizm replikacji danych

Ponieważ rozwiązanie zakłada możliwość przekierowania na innych serwer usług posiadający kopię zapasową bazy danych. Musi zostać zapewniony mechanizm replikacji uaktualniający bazę danych na zapasowym serwerze. Replikacja taka będzie wykonywana raz dziennie.

Dostępność aplikacji

Aplikacja będzie dostępna z poziomu przeglądarki internetowej. Będzie dostosowana funkcjonalnie do przeglądarek :

- Chrome
- Firefox 4,5
- Internet Explorer 9
- Opera

Zostanie zagwarantowany dostęp do funkcjonalności z poziomu wymienionych wyżej przeglądarek. Nie zostanie zagwarantowany idealnie taki sam sposób prezentacji aplikacji.

Obciążalność

Aplikacja będzie w stanie obsłużyć jednorazowo 1000 żądań i będzie gwarantować czas odpowiedzi w granicach maksymalnie 1 sekundy

4.2 Wybrane diagramy przypadków użycia

4.3 Projekt bazy danych

4.3.1 Opis encji

Kurs

Zawiera dane dotyczące kursu : Id , Typ Kursu , Nazwę , Logo , Data Utworzenia , Opis , Krótki Opis , Pole Wiadomości

TypKursu

Słownik opisujący typ kursu.

UkonczonyTest

Encja reprezentująca ukończony test. Zawiera m.in otrzymaną ocenę oraz datę ukończenia testu.

Test

Reprezentuje Test wypełniany przez kursanta należący do materiału nauczania. Zawiera dane opisowe oraz sekcje zawierające merytoryczną treść kursu.

TypTestu

Słownik opisujący typ testu

PytanieTestowe

Reprezentuje pytanie należące do zbioru pytań wchodzących w skład testu. Zawiera listę odpowiedzi , tytuł oraz treść pytania.

PytanieTestoweOdpowiedz

Reprezentuje odpowiedzi będące częścią pytania. Zawiera treść odpowiedzi oraz wskaźnik czy jest to poprawna odpowiedź.

ShoutBox

Encja reprezentująca moduł krótkich wiadomości tekstowych. Zawiera listę wiadomości.

ShoutBoxWiadomosc

Reprezentuje wiadomość wyświetlaną w obrębie ShoutBoxa. Zawiera treść , login użytkownika oraz datę przesłania wiadomości.

Dziennik

Encja reprezentująca dziennik ocen. Każdy kursant posiada pojedynczy obiekt dziennika dla każdego kursu do którego dołączył. Posiada listę ocen przypisanych do danego użytkownika oraz kursu.

DziennikOcena

Encja ocena wchodząca w skład dziennika. Reprezentuje pojedynczą ocenę otrzymywaną po skończeniu testu.

MaterialNauczania

Opisuje pojedynczy materiał nauczania będący podstawowym środkiem przekazu merytorycznej zawartości dla kursanta.

Section

Sekcja jest integralną częścią materiału nauczania. Zawiera informacje dydaktyczne. Materiał nauczania może posiadać wiele sekcji.

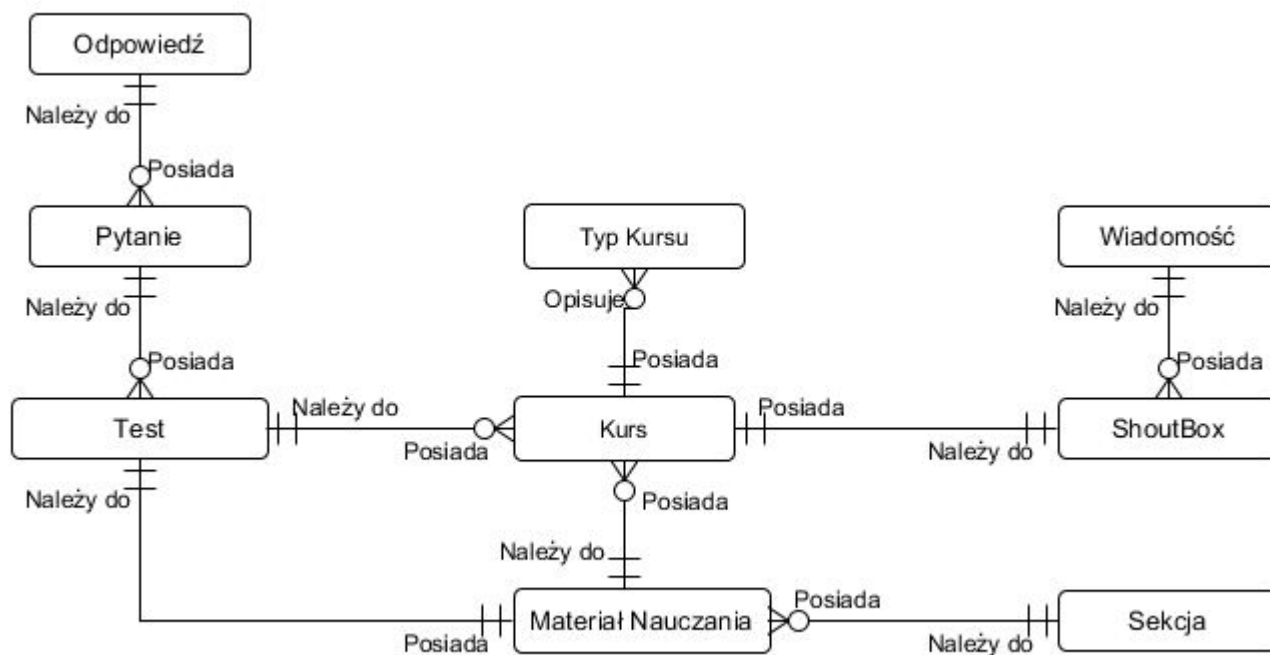
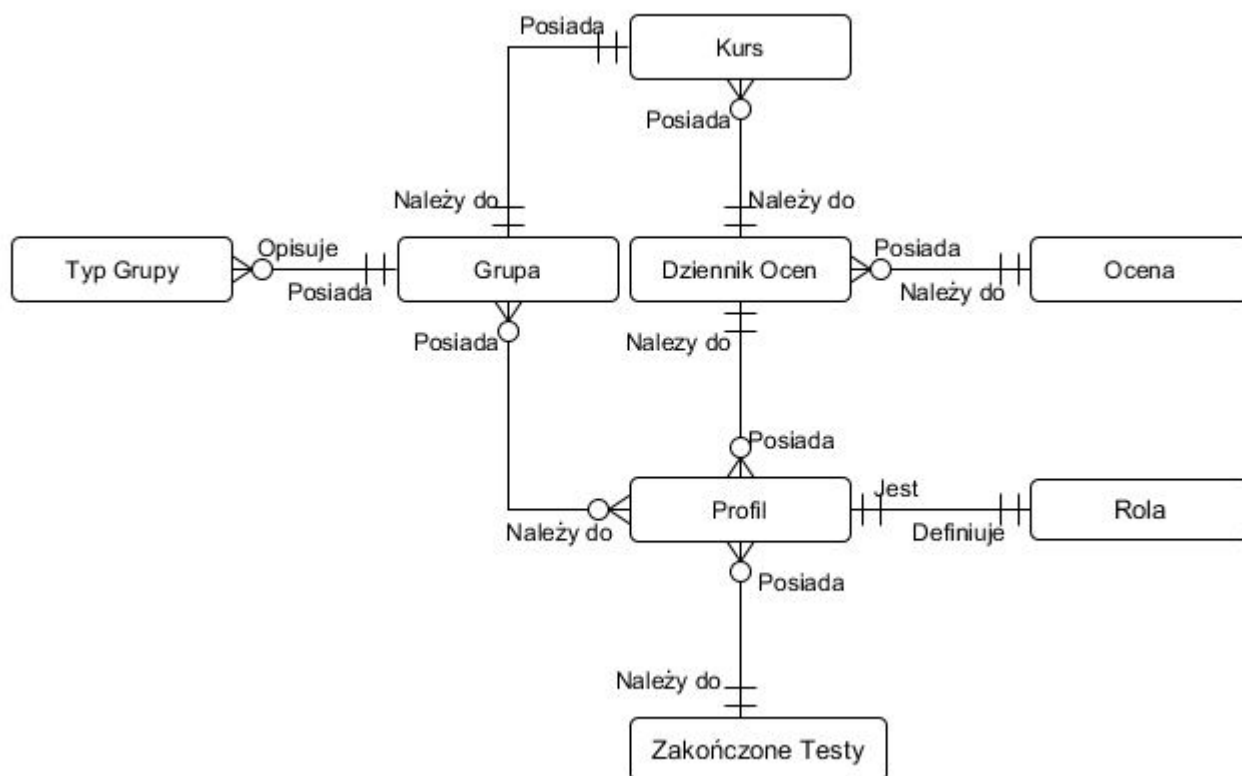
Profil

Reprezentuje użytkownika systemu.

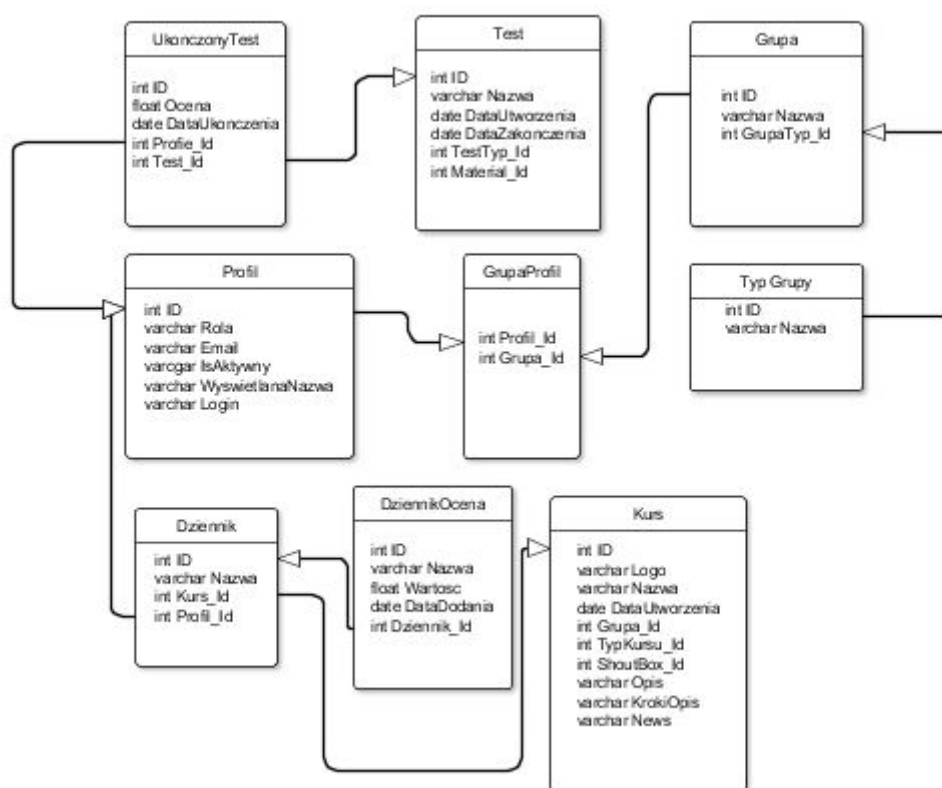
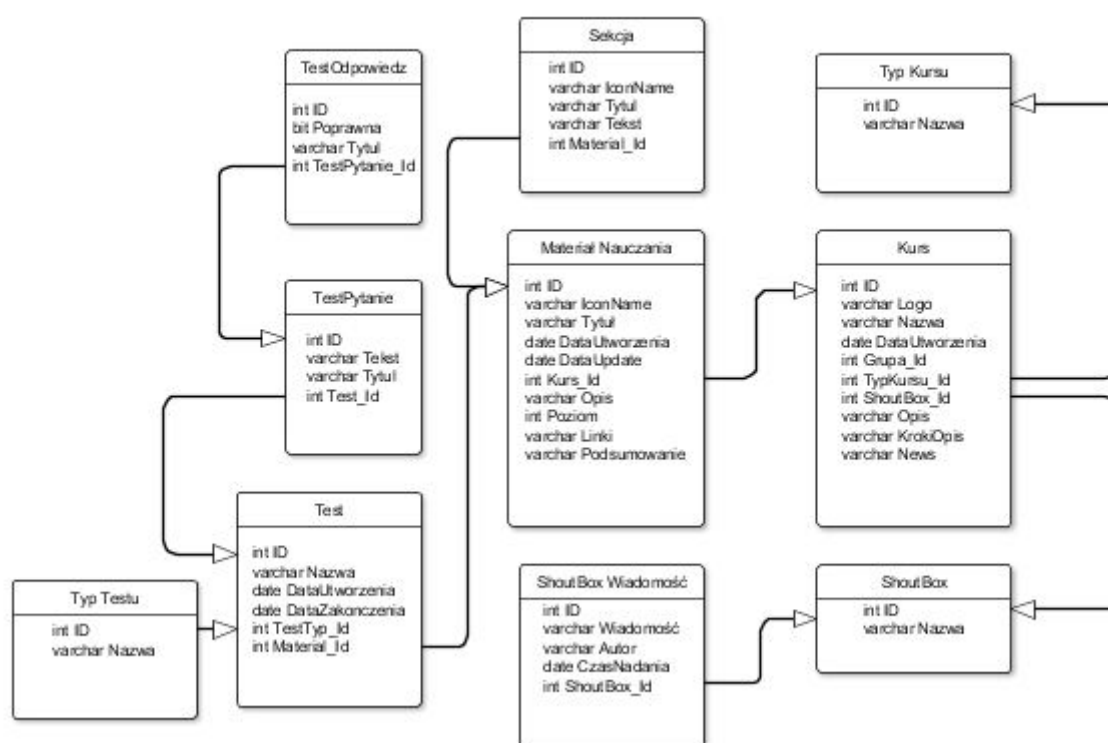
GrupaUzytkownikow

Encja opisująca grupę użytkowników. Każdy kurs posiada pojedynczą grupę użytkowników. Zawiera listę użytkowników.

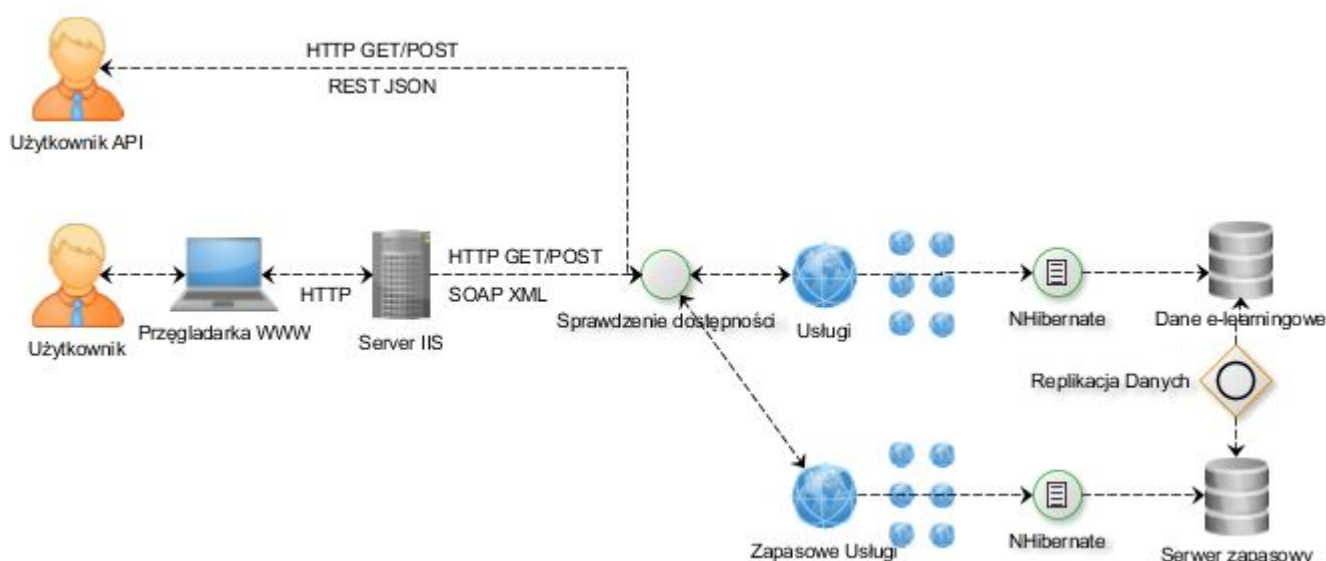
4.3.2 Uproszczony model konceptualny (CDM) - struktura tabel i relacje



4.3.3 Model fizyczny bazy danych PDM



4.4 Architektura Systemu



Rys. Schemat proponowanego rozwiązania

Proponowane rozwiązanie zakłada dwa sposoby dostępu do danych i aplikacji. Pierwszym z nich jest wykorzystanie aplikacji stworzonej w technologii Asp.Net Mvc postawionej na serwerze IIS. Dostęp taki realizowany jest za pomocą przeglądarki internetowej. Warstwa dostępu do bazy danych oraz usługi sieciowe są transparentne dla klienta korzystającego z przeglądarki internetowej. Po wykonaniu akcji przez klienta generowany jest odpowiednia strona www korzystająca z bazy danych. Nim taka strona zostanie wygenerowana serwer wysyła zapytanie do usługi sieciowej wykorzystując protokół HTTP jako warstwę transportową i protokół SOAP w formacie XML jako warstwę formatu wiadomości. Zapytanie jest przetwarzane przez serwer usług sieciowych, który wykorzystując framework NHibernate realizuje zapytanie do bazy danych. Dane następnie zwracane są zgodnie z protokołem SOAP do serwera i po przetworzeniu do klienta w formie odpowiedniej strony www.

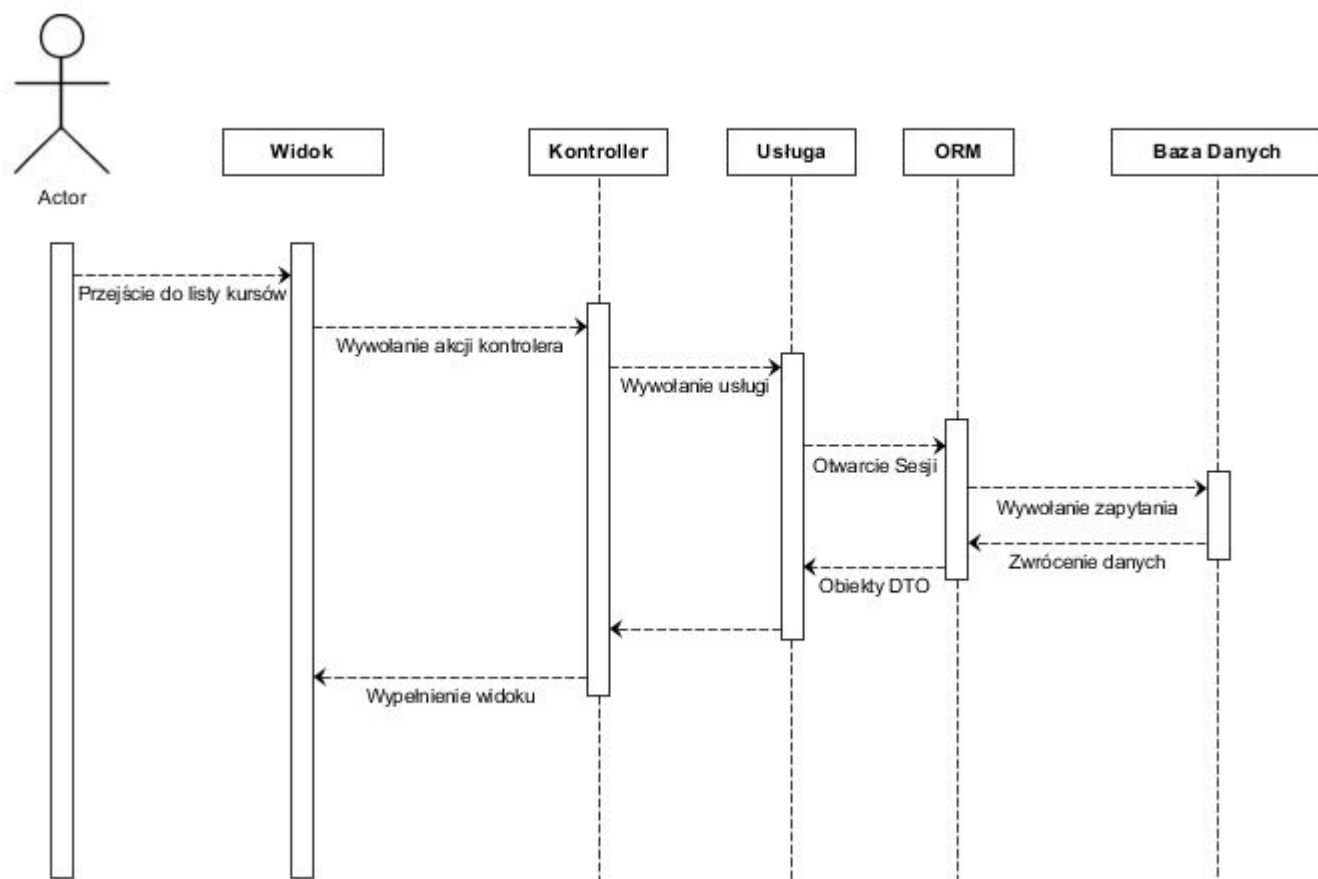
Drugim sposobem dostępu do danych jest skorzystanie z API wystawionego w formie protokołu REST. Zapytanie takie realizowane jest przy pomocy protokołu HTTP i zwykłej komendy np Get. Klient taki otrzymuje dane w postaci formatu JSON.

Komunikacja jest zrealizowana za pomocą frameworka WCF firmy Microsoft. Usługi sieciowe realizują dostęp do bazy danych. Są podzielone na sześć usług:

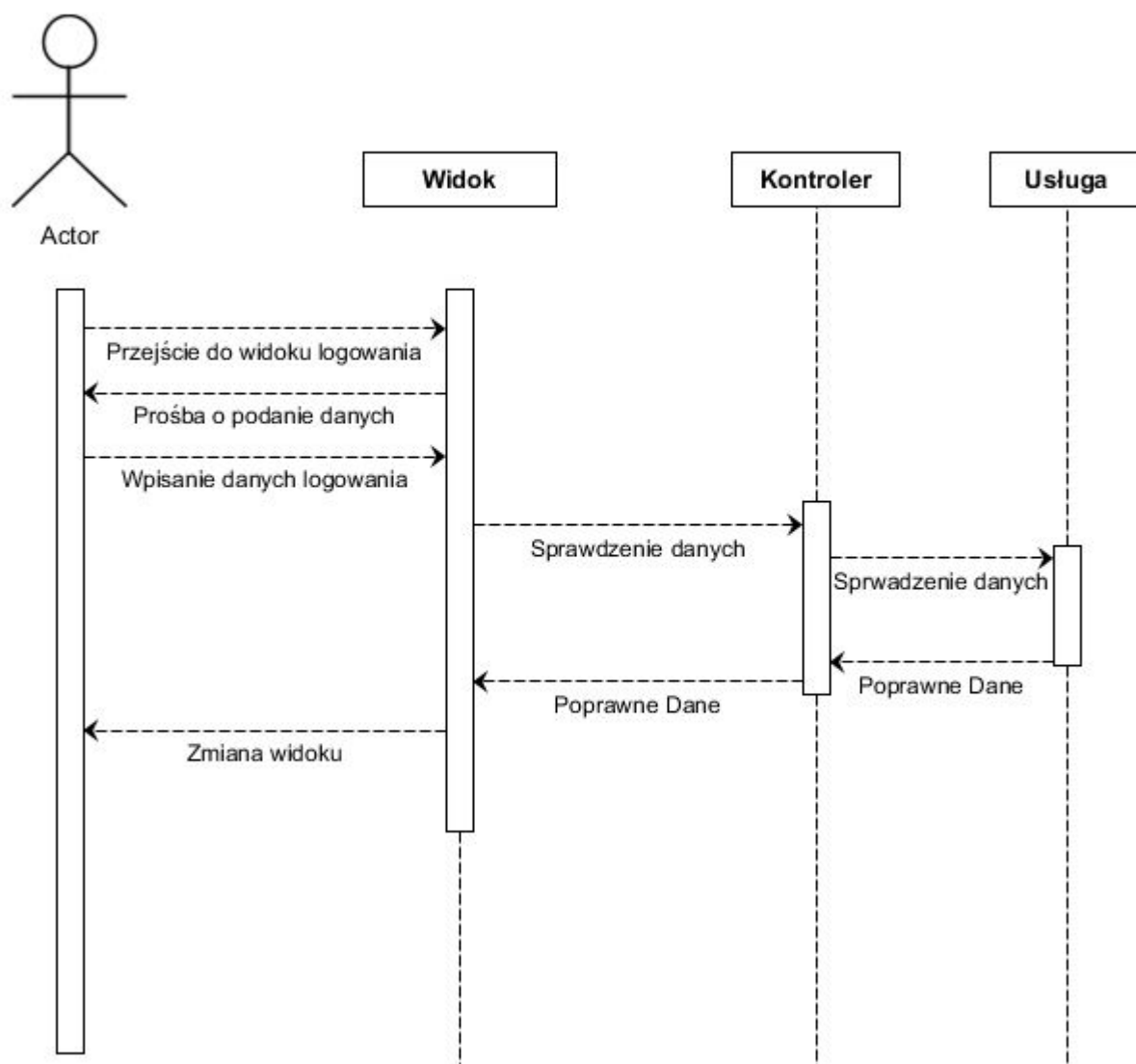
- Usługa kursów
- Usługa materiałów nauczania
- Usługa testów
- Usługa grup
- Usługa dzienników
- Usługa profili

Rozwiązanie zakłada postawienie zapasowego serwera usług sieciowych realizującego dostęp do zapasowej bazy danych. By zapewnić synchronizację danych pomiędzy tymi bazami projekt zakłada stworzenie prostego mechanizmu replikacji danych. W momencie odebrania zapytania od klienta serwer podejmuje próbę nawiązania połączenia z serwerem głównym gdy nie otrzyma odpowiedzi przełącza się na serwer zapasowy .

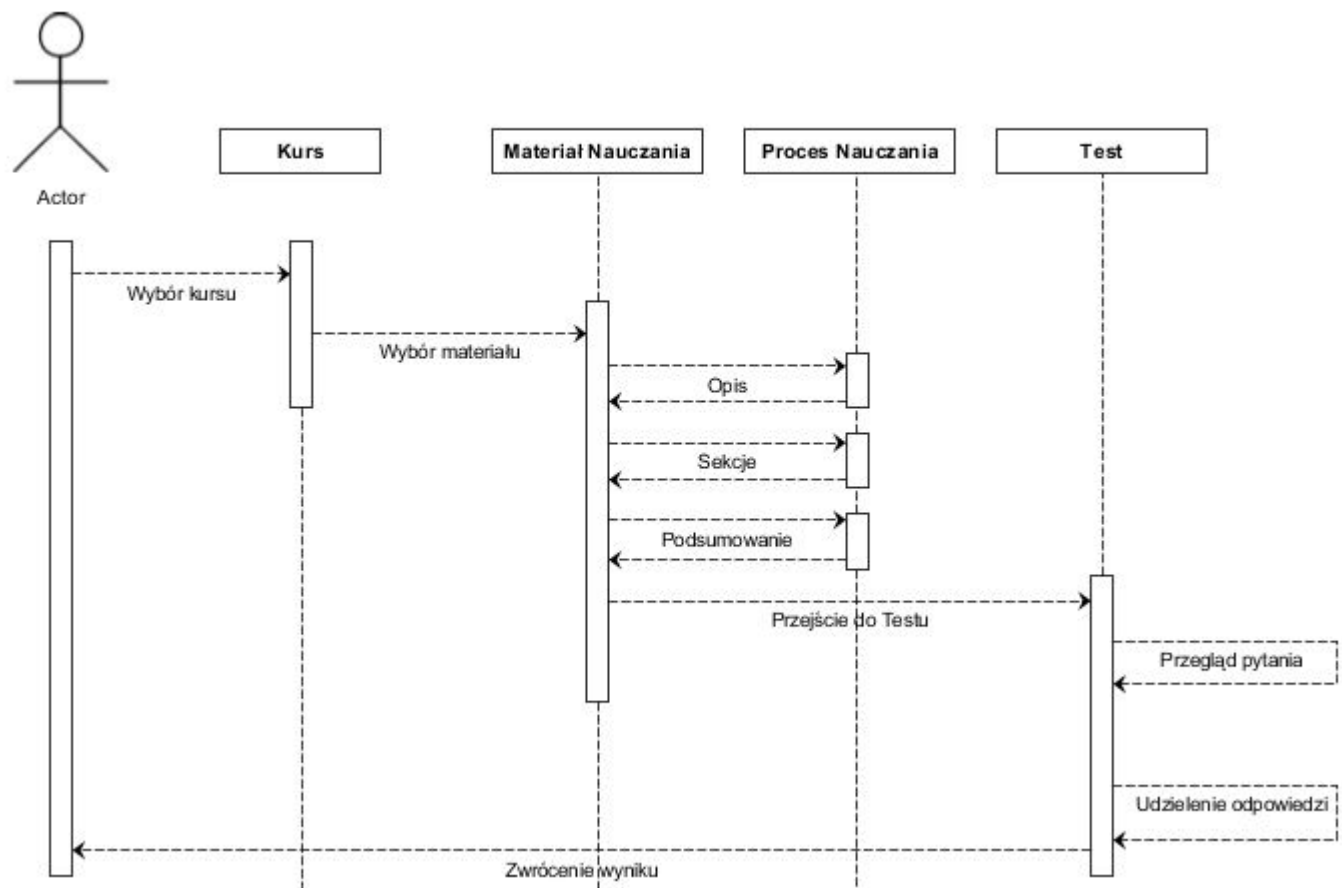
4.4.1 Wybrane diagramy sekwencji funkcjonalności



Rys. Diagram sekwencji generowania strony www



Rys. Diagram sekwencji logowania do aplikacji



Rys. Diagram sekwencji procesu nauczania

4.4.2 Mechanizmy zabezpieczeń

Wiadomości przesyłane w formacie JSON wystawione w formie protokołu REST do których ma dostęp użytkownik korzystający z API nie są w ogóle zabezpieczone. Są to dane ogólnie dostępne. Jedyną formą zabezpieczenia API będzie wystawienie funkcji i dostępu do bazy danych tylko dla wybranych danych. Ograniczenie dostępu zostanie zrealizowane poprzez odpowiednie parametry funkcji. Tzn pobierając np listę kursów klient nie będzie mógł jedynie podać ID użytkownika którego kursy ma wyświetlić.

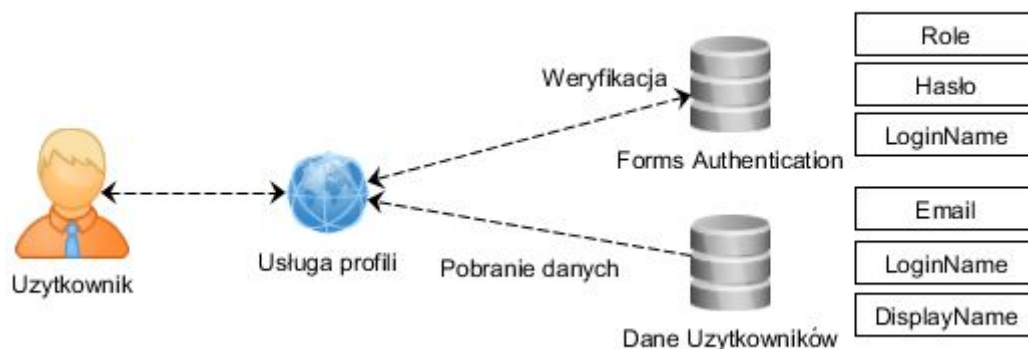
Wiadomości przesyłane protokołem SOAP z wykorzystaniem formatu danych XML będą już miały większy zakres zastosowanych zabezpieczeń. Wiadomości zawierające dane poufne jak dane logowania, hasła dane na temat użytkownika będą szyfrowane oraz przesyłane protokołem HTTPS po porcie 443 tzn pierw zostanie sprawdzona poprawność certyfikatom a dopiero po tym prawdziwa komunikacja po http, zapewni to dodatkowe bezpieczeństwo.

Dane zwykle bez poufnej zawartości jak dane kursów materiałów nauczania nie będą zabezpieczone.

Aplikacja odporna jest na ataki skryptowe. Nie ma możliwości wstrzyknięcia skryptu do aplikacji wywołując tym samym działania niepożądane. Każde pole tekstowe analizowane jest przez serwer pod punktem zawartości.

Dzięki zastosowaniu Mappera obiektowo relacyjnego warstwa dostępu do danych oddzielona jest od użytkownika mocno typowana warstwa która znacznie ogranicza możliwość wykonania ataku SQL Injection. Dodatkowo wprowadzane dane są sprawdzane pod względem możliwych ataków.

Ponieważ dostęp do aplikacji realizowany jest w sposób przypominający protokół REST tzn . Akcje dostępne są z poziomu adresu URL. Akcje opatrzone są dodatkowymi mechanizmami zabezpieczeń które przed sprawdzeniem sprawdzają poziom uprawnień użytkownika. Dzięki temu anonimowy użytkownik po próbie wywołania takiej akcji zostanie prze kierowany do panelu logowania natomiast zwykły użytkownik otrzyma informacje ze nie posiada odpowiedniego poziomu uprawnień.



Rys. Rozdzielenie mechanizmu logowania na dwie bazy danych

Mechanizm logowania oparty jest o framework Forms authentication zapewniający podstawowe funkcjonalności zarządzania użytkownikami. Forms authentication wymaga oddzielnej bazy danych. Usługa profili wiąże ze sobą profile w głównej bazie danych oraz profile z Forms Authentication.

Podział baz został wprowadzony po to by nie mieszać obu baz jedna baza jest typowo przystosowana do trzymywania danych użytkowników oraz spełnia wytyczne firmy Microsoft jest to ich produkt. Dodatkowo bazę profili można umieścić w pliku dodatkowo zwiększając bezpieczeństwo systemu bądź zmniejszając użycie zasobów. Przy hostingu na , którym wystawiłem aplikację mam określony limit ilości obsługiwanych baz danych MSSQL 2008 więc by zmniejszyć ilość baz mogę przenieść bez problemowo tę bazę do pliku.

Rozdział 5

Implementacja systemu zdalnego nauczania

5.1 Zewnętrzny hosting

Aplikacja została uruchomiona pod adresem <http://www.codedash.mfranc.com/> . Przestrzeni hostingowej dostarcza firma <http://www.webio.pl/> . Na serwerze postawione są: Dwie bazy danych:

- Dane profilowe
- Dane aplikacji

Oraz dwie aplikacje:

- Aplikacja główną stworzoną w oparciu o framework Aep.Net Mvc 3 będąca systemem e-learningowym
- Aplikacja z usługami sieciowymi

Aplikacja skonfigurowana jest tak by składować logi w odpowiednich katalogach. Dzięki temu istnieje możliwość szybkiego zdiagnozowania i poprawienia problemu z aplikacją.

Serwer hostingowy obsługuje Asp.Net Mvc 3 oraz .Net Framework 4.0

Moduł testujący usługę API udostępniającą dane po protokole REST został uruchomiony pod adresem <http://www.mfranc.com/codedash-test/> .

Do testów skonfigurowano również lokalny serwer usług sieciowych , będzie on wykorzystany przy testach mechanizmu dynamicznej zmiany serwera w momencie zerwania połączenia.

5.2 Realizacja bazy danych

Baza danych została zrealizowana na silniku bazodanowym MSSQL 2008 R2. Na serwerze wystawione są dwie bazy danych. Jedna odpowiedzialna za przechowywanie danych o użytkownikach, druga z zawartością danych aplikacji. Dostęp do obu baz realizowany jest za pomocą frameworka NHibernate, który jest mapperem relacyjno obiektowym, zapewniającym dostęp do bazy danych z poziomu klas i obiektów.

Wystawiono również na lokalnym serwerze zapasową bazę danych. Jest ona używana przy mechanizmie dynamicznej zmiany serwera usług sieciowych. Baza ta o ustalonej godzinie synchronizuje się z bazą główną.

5.3 Realizacja usług sieciowych

REST SOAP Aplikacja na mfranc.com

5.4 Wykorzystane narzędzia

5.4.1 Mechanizm logowania zdarzeń - NLog

W procesie wytwarzania oprogramowania bardzo ważnym aspektem zwiększającym znacząc powodzenie projektu jest zapewnienie odpowiedniego mechanizmu logowania zdarzeń oraz błędów wewnątrz aplikacji. Dzięki takiemu mechanizmowi przyspieszamy proces naprawy oraz znajdowania błędów. Dodatkowo możemy analizować poprawność działania aplikacji. Przyczynia się to znacząco do zmniejszenia kosztów utrzymania oraz wprowadzania zmian i poprawek w aplikacji.

W niniejszej pracy do stworzenia mechanizmu logowania zastosowałem popularną darmową bibliotekę NLog stworzoną przez Jarosława Kowalskiego. Jest to stosunkowo prosta a zarazem rozbudowana biblioteka.

Wykorzystuje ją się poprzez dołączenie odpowiedniej instancji klasy `Logger` do danej klasy `y`, z poziomu której chcemy logować zdarzenia.

```
private static NLog.Logger logger = NLog.LogManager.GetCurrentClassLogger();
```

Następnie w interesującym nas miejscu wywołujemy metody `Logger` określając rodzaj loga.

```
logger.Debug(Starting Add Mark [Get] with : journalId 0",id);
```

Do wyboru mamy m.in takie rodzaje jak:

- Informacja
- Debugowanie
- Błąd
- Błąd Krytyczny









Do `logger` możemy wprowadzić dowolnie sformatowany ciąg znaków.

Prócz wywołania `NLog` bardzo istotną rzeczą jest odpowiednie skonfigurowanie pliku konfiguracyjnego aplikacji. W pliku konfiguracyjnym możemy bowiem ustalić miejsce do którego chcemy zapisać logi. Dostępne są m.in :

- Plik
- Adres E-mail
- Wysyłanie danych po porcie
- Konsola
- Baza Danych

Dla każdego źródła istnieje możliwość skonfigurowania rodzaju informacji jakie ma zapisywać.

W aplikacji zastosowałem mechanizm zrzucania logów informacyjnych oraz związanych z procesem logowania do pliku oraz wysyłanie protokołem UDP na port 9999. Na tym porcie lokalnie nasłuchuje darmowa aplikacja Sentinel która pozwala szybko analizować wysyłane logi. Dzięki takiej konfiguracji przyspieszył się proces implementacji ponieważ miałem wgląd w proces wykonywania akcji wewnątrz aplikacji.

Log viewer			
Type	Date/Time	System	Description
	20:52:28	elearn.MvcApplication	Application Started
	20:52:38	elearn.Session.SessionState	Adding Current user data to session state
	20:52:44	NHiberanteDal.DTO.DtoMappings	DTO Mappings Initialized
	20:52:45	NHiberanteDal.DTO.DtoMappings	Problem initializing DTO mappings! - The following 2 properties on News LearningMaterials Add a custom mapping expression, ignore, or rename the property
	20:52:45	ELearnServices.ProfileService	Created ProfileService
	20:53:08	ELearnServices.ProfileService	Error : ProfileService.GetByName - Sequence contains no elements
	20:53:09	elearn.Session.SessionState	Error - Retrieving Current user data from session state Object reference not set to an instance of an object. at elearn.Session.CurrentProfileSession..ctor(ProfileModelDto profi at elearn.Session.SessionState.GetCurrentUserDataFromSession() i
	20:54:56	NHiberanteDal.DTO.DtoMappings	DTO Mappings Initialized

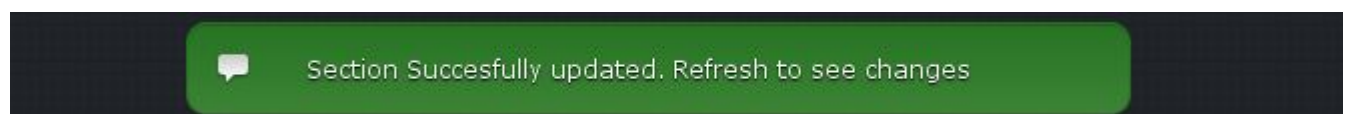
Rys. Przykładowy log z programu Sentinel

Dodatkowo by zapewnić szybszy czas reakcji na poprawki , logi oznaczone jako krytyczne skonfigurowane zostały tak by wysyłać wiadomość z logiem na adres email. Dzięki temu administrator otrzymuje szybko informację o tym , że dzieje się coś naprawdę ważnego powodującego błędne działanie aplikacji w stopniu którym aplikacja nie może działać stabilnie.

Logami objęte zostały :

- Operacje wykonywane na bazie danych
- Akcje wywoływane w aplikacji po stronie serwera

Prócz generowania logów po stronie serwerach w aplikacji zaimplementowany jest mechanizm wyświetlania alertów z informacjami po stronie użytkownika przeglądarki. Zadaniem alertów jest informowanie użytkownika o zachodzących akcjach takich jak np zapisanie danych do systemu bądź problem z połączeniem.



Rys. Przykładowy alert

5.4.2 Testy jednostkowe - NUnit

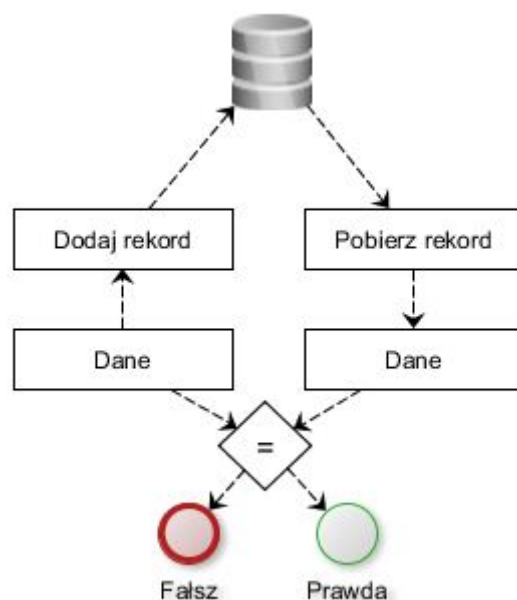
Testy jednostkowe są nowoczesnym narzędziem pozwalającym testować pojedyncze "jednostkowe" funkcjonalności aplikacji. Do przeprowadzenia testów w aplikacji użyłem frameworka NUnit.

Testami jednostkowymi została pokryta logika bazodanowa oraz część akcji wywoływanych przez kontrolery.

Testy dostępu do bazy danych

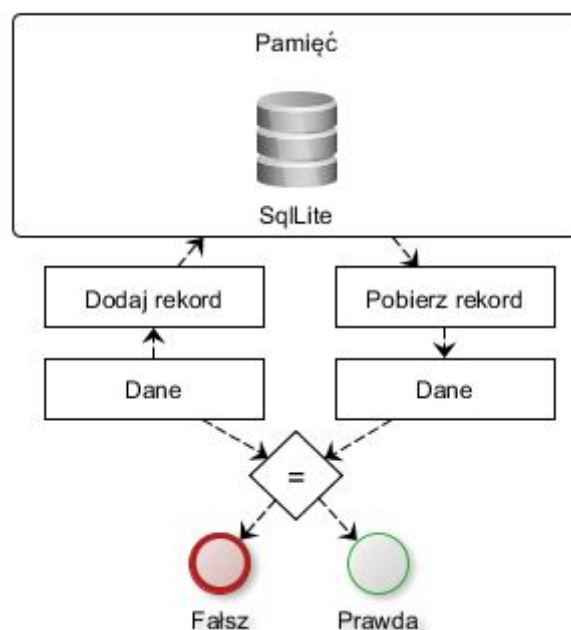
Testowanie jednostkowe bazy danych jest dość skomplikowanym zagadnieniem. Testowanie jednostkowe zakłada wykonywanie testów tylko pojedynczej funkcjonalności. W przypadku bazy danych byłaby to np. operacja dodawania rekordu do bazy. Procedura przeprowadzenia takiego testu polega na:

1. Wywołaniu metody dodawania rekordu do bazy danych. 2. Wywołania metody pobierania rekordu z bazy danych. 3. Operacja porównania rekordu pobranego z rekordem dodanym



Rys. Prosty test jednostkowy bazy danych

By przeprowadzić taki test potrzebujemy dostępu do bazy danych. Dostęp do bazy można zrealizować na kilka sposobów. Mianowicie można przeprowadzać testy na prawdziwej istniejącej bazie danych. W takim przypadku jednak należy pamiętać by rekord testowy po teście usunąć. Innym wyjściem jest stworzenie oddzielnej bazy danych i przeprowadzanie testów na niej. W tym przypadku wystarczy, że za każdym razem odtworzymy pustą bądź domyślną bazę danych. Testowanie na prawdziwej bazie danych jest czasochłonne. Lepszym rozwiązaniem jest przeprowadzanie testów na bazie danych generowanej w pamięci. W tym przypadku można skorzystać z bazy danych opartej na silniku SQLite. Open Sourcowego rozwiązania. Ten mechanizm został wykorzystany w mojej pracy.



Rys. Test jednostkowy z bazą danych w pamięci

Przed każdym rozpoczęciem testu generowana jest baza danych oraz wypełniana jest danymi potrzebnymi przy testach. Tzn generowane są dane testowe. W przypadku np. testowania elementu "Kurs" posiadającego listę elementów "Test". Generowany jest kurs w przykładową listą testów.

Ponieważ wykorzystuję mapper relacyjno obiektowy (NHibernate) , przed rozpoczęciem testów bazy danych przeprowadzany jest test konfiguracji NHibernatea. W przypadku błędnej konfiguracji od razu wiadomo , że problemem jest błąd tutaj a nie w testach przeprowadzanych na bazie danych.

Testy bazy danych obejmują :

- Testy generycznej klasy repozytorium
- Testy zapytań wykorzystujących język HQL

Łącznie na bazie danych przeprowadzanych jest 95 testów jednostkowych.

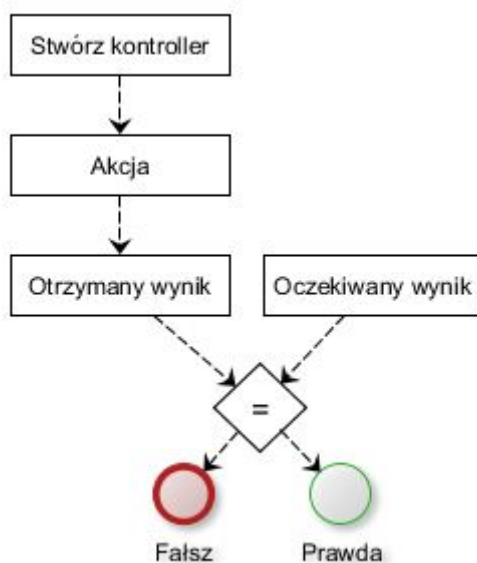
Testy kontrolerów - RhinoMocks

Testowanie jednostkowe kontrolerów jest bardziej złożone niż testy bazodanowe. W przypadku testów bazodanowych jedyną rzeczą od której one zależą jest dostęp do bazy danych. Ponieważ testy te wykonują operacje na bazie danych więc nie jest to problem.

W przypadku testowania kontrolerów sytuacja jest bardziej złożona. Kontrolery bardzo często wyciągają dane przed wygenerowaniem strony aplikacji. Baza danych jest ich zewnętrzną zależnością. Niestety testowanie dostępu do bazy danych nie jest sensem testów kontrolera. Test kontrolera musi testować jedynie swoje zachowanie na odpowiednie dane z bazy danych. Najlepszym rozwiązaniem byłoby w ogóle pominiecie bazy danych i wstrzyknięcie danych testowych na których chcemy opierać dalszy test. Na szczęście istnieje taka możliwość i do tego celu stosuje się specjalne obiekty zwane mockami bądź stubami. W niniejszej pracy wybrałem pierwszy rodzaj obiektów i posłużyłem się frameworkiem RhinoMocks do ich generacji.

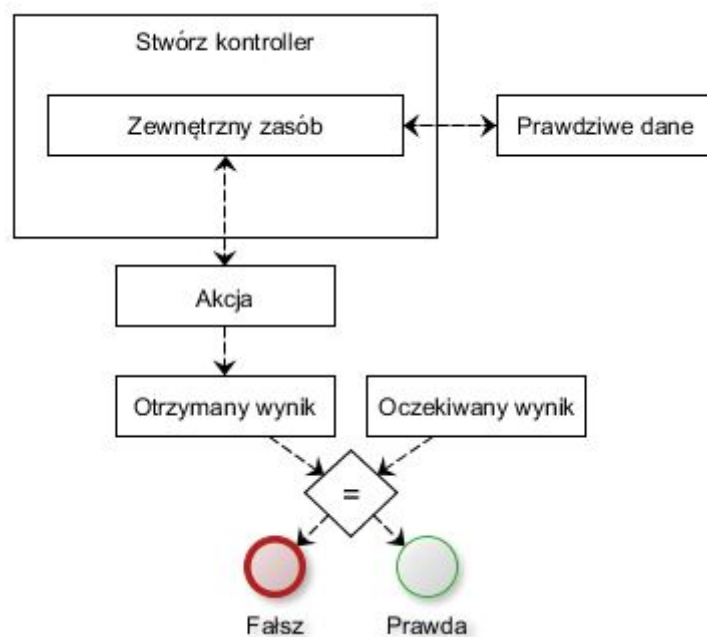
Dzięki zastosowaniu frameworka Asp.Net Mvc można testować kontroler, które można traktować jak odseparowane klasy od reszty systemu. Test kontrolera realizowany jest podobnie jak test bazodanowy.

1. Tworzymy dany kontroler wywołując jego konstruktor 2. Wywołujemy metodę (Akcję) na kontrolerze podając odpowiednie parametry. 3. weryfikujemy otrzymany wynik.

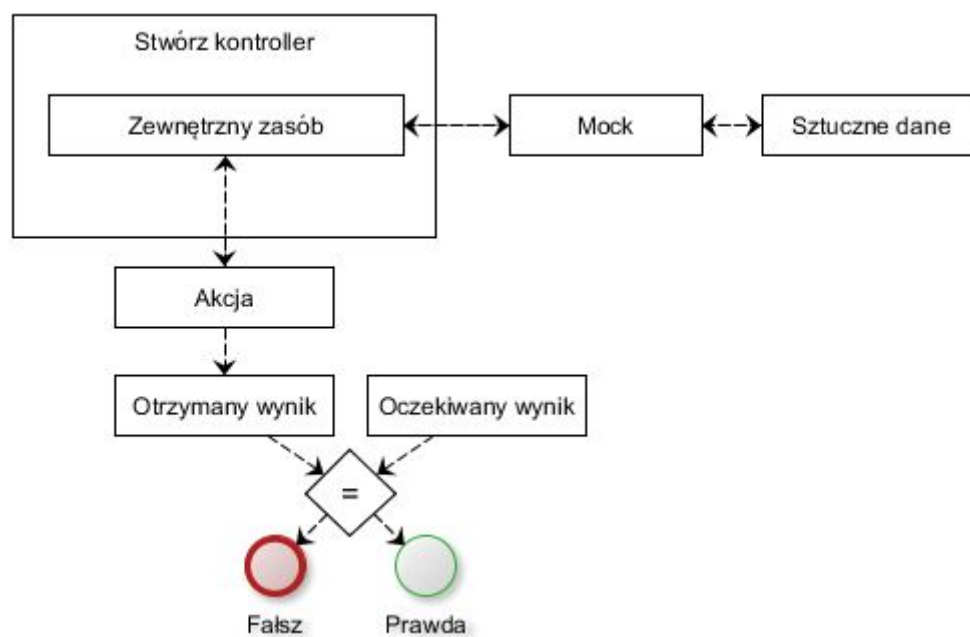


Rys. Test jednostkowy kontrolera bez zewnętrznych zasobów

W przypadku kontrolera korzystającego z zewnętrznych zasobów takich jak baza danych bądź globalna sesja, przed stworzeniem kontrolera należy zainicjować obiekt typu mock imitujący zasób i wstrzyknąć go do środka kontrolera ("Wstrzyknięcie zależności").



Rys. Test kontrolera z zewnętrznymi zasobami



Rys. Test z kontrolera przy wykorzystaniu mocka

```

[Test]
public void Post_if_create_in_db_course_failes_then_return_error_view()
{
    #region Arrange
    using (Mock.Record())
    {
        Expect.Call (CourseService.AddCourse (Course) ).IgnoreArguments () .Return (null) ;
    }
    #endregion

    #region Act
    ViewResult view;
    using (Mock.Playback())
    {
        view = (ViewResult)CourseController.Create(Course);
    }
    #endregion

    #region Assert
    Assert.That (view.ViewName, Is.EqualTo ("Error"));
    Assert.That (view.ViewBag.Error, Is.EqualTo (elearn.Common.ErrorMessages.Course.AddToDbError));
    #endregion
}

```

Rys. Przykładowy kod testu z mockiem

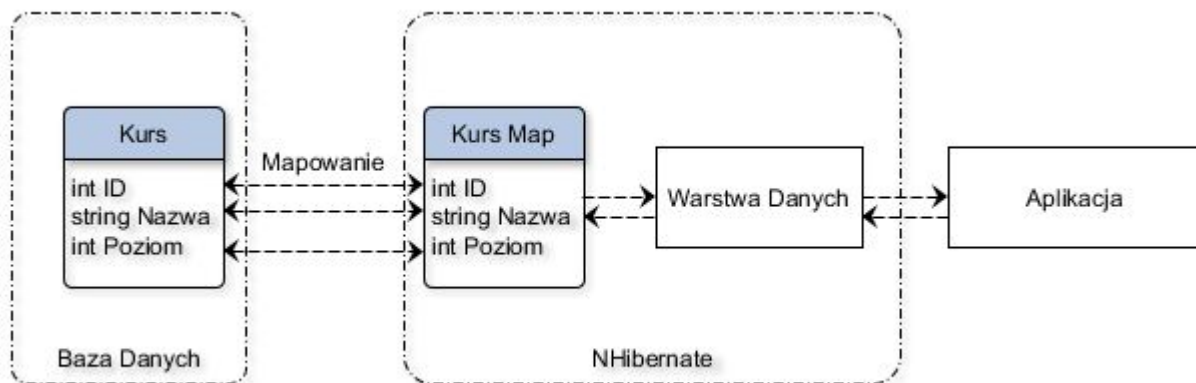
W powyższym kodzie testowany jest kontroler Kursów i jego metoda Create. Test testuje okoliczność gdy kontroler kursów pobierając dane z bazy danych otrzyma wartość null. W tym przypadku ma zwrócić widok błędu. Linijka tworzenia mocka po pierwsze definiuje, że oczekuje wywołania danej funkcji następnie określłam parametry wejściowe (w tym przypadku ignoruje je bo są nieistotne) i na koncu określłam jakie dane mają być zwrócone.

Po zdefiniowaniu mocka uruchamiam odtwarzanie jego zachowania i w momencie przeprowadzania testu i wywoływania metody Create aplikacja odpali test i po natrafieniu na zmockowaną metodę w tym przypadku AddCourse nie odwoła się do bazy danych ale do obiektu typu mock zwracając wartość null.

Łączn ilość testów 50.

5.4.3 Mapowanie obiektowo relacyjne - NHibernate

Dostęp do bazy danych realizowany zrealizowany jest za pomocą mappera obiektowo relacyjnego "NHibernate". Jest to open source'owa implementacja frameworka Hibernate popularnego na platformie Java.



Rys. Proces mapowania relacyjnego obiektowego

Tworzenie mapowań - FluentNHibernate

Standardowo mapowania w NHibernate definiuje się w oddzielnych plikach XML. Jest to dość problematyczna metoda podatna na błędy plus mapowania nie są dość czytelne. Innym sposobem generowania mapowań jest zastosowanie frameworka FluentNHibernate pozwalającego definiować mapowania wewnątrz kodu .Net

```
public class CourseModelMap : ClassMap<CourseModel>
{
    public CourseModelMap()
    {
        Id(x => x.ID);
        Map(x => x.Logo);
        Map(x => x.Name).Not.Nullable();
        Map(x => x.CreationDate).Not.Nullable();
        Map(x => x.Description);
        Map(x => x.ShortDescription);
        Map(x => x.News);
        Map(x=>x.Password).Nullable();

        //One
        References(x => x.Group).Not.Nullable().Cascade.All().Not.LazyLoad();
        References(x => x.CourseType).Not.Nullable().Not.LazyLoad();
        References(x => x.Forum).Not.Nullable().Cascade.All().Not.LazyLoad();
        References(x => x.ShoutBox).Not.Nullable().Cascade.All().Not.LazyLoad();

        //Many
        HasMany(x => x.Contents).KeyColumns.Add("CourseId").Cascade.SaveUpdate().LazyLoad();
        HasMany(x => x.Surveys).KeyColumns.Add("CourseId").Cascade.SaveUpdate().LazyLoad();
        HasMany(x => x.Tests).KeyColumns.Add("CourseId").Cascade.SaveUpdate().LazyLoad();
        HasMany(x => x.LearningMaterials).KeyColumns.Add("CourseId").Cascade.SaveUpdate().Not.LazyLoad();
    }
}
```

Mapowanie właściwości klasy

Definiowanie relacji 1 do 1

Definiowanie relacji 1 do wielu

Rys. Przykładowe mapowanie klasy Kurs

Mapowanie realizowane jest poprzez dziedziczenie generycznej klasy ClassMap. Wszystkie parametry mapowanej klasy muszą być publiczne oraz oznaczone słowem kluczowym virtual. Dlaczego virtual? FluentNHibernate wymaga tego słowa kluczowego by wstrzykiwać swoje specjalne metody.

Na początku należy zdefiniować które pole jest polem kluczem jest to wymagane pole bez którego nie można przeprowadzić procesu mapowania. Pole to musi być unikatowe. Następnie poprzez użycie Funkcji Map definiuje się mapowania parametrów do wierszy tabel. Kolejną istotną rzeczą do zdefiniowania są relacje. Reference używamy do zdefiniowania relacji 1 do 1 natomiast HasMany 1 do wielu. Przy każdej funkcji mapującej można ustawić dodatkowe opcje jak np wiersz przechowujący klucz do elementu referencyjnego bądź można zdefiniować czy dane powinny być późno wiązane czy nie.

Identyfikacja tabeli do której należy dana klasa realizowana jest poprzez nazwę klasy która jest mapowana. Istnieje oczywiście możliwość przeciążenia tej nazwy poprzez użycie metody Table()

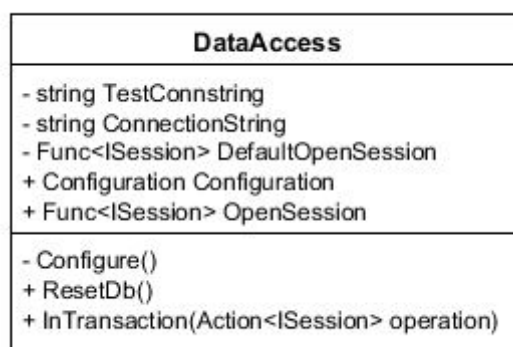
5.5 Realizacja aplikacji

5.5.1 Realizacja mechanizmów dostępu do bazy danych

Dostęp do bazy danych opiera się na frameworku NHibernate. Cała komunikacja oraz inicjalizacja opakowan

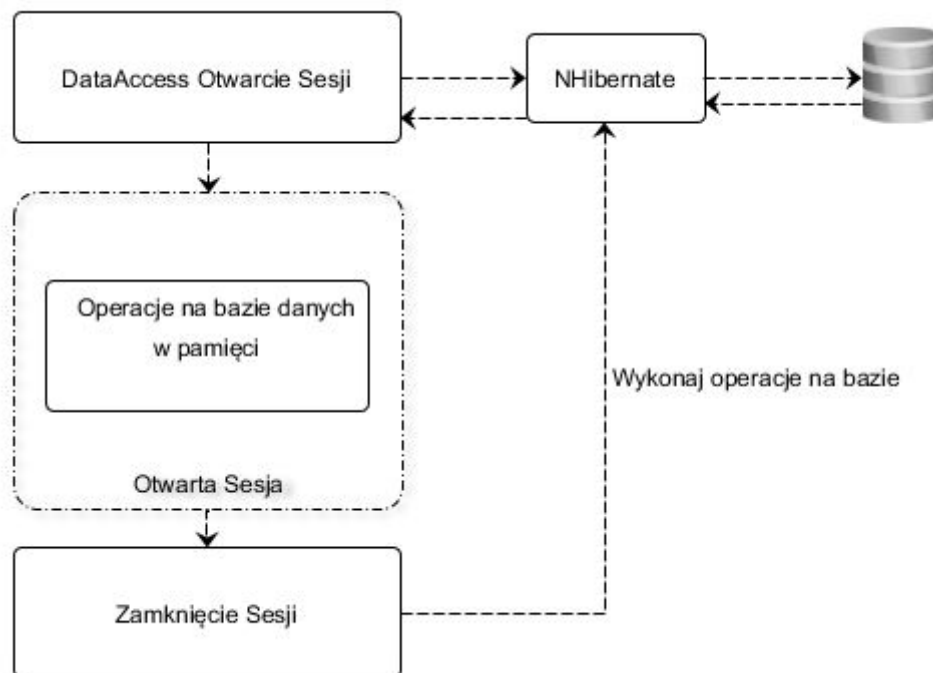
a jest wewnątrz klasy `DataAccess`, która upraszcza proces dostępu do danych.

`DataAccess` jest klasa obsługującym proces dostępu do bazy danych. Wykorzystywana jest na serwerach udostępniających usługi sieciowe. Jest odpowiedzialna przede wszystkim za konfigurowanie połączenia z bazą danych poprzez mapper obiektowo relacyjny NHibernate. Ponieważ testy jednostkowe pokrywające bazę danych wykorzystują bazę danych generowaną w posiada możliwość przekierowania dostępu na bazę danych z pamięci.



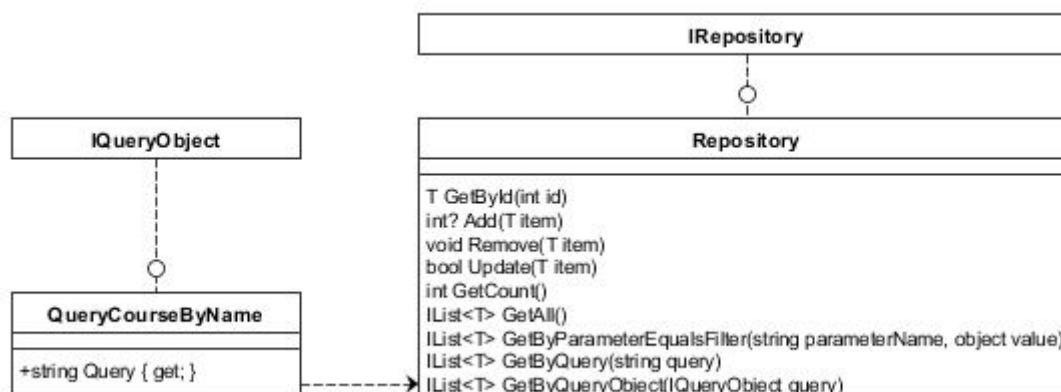
Rys. Diagram klasy `DataAccess`

Klasa ta pozwala zresetować bazę danych i wygenerować nową na podstawie mapowań. Dodatkowo dodana jest metoda `InTransaction` która pozwala przeprowadzić operację na bazie w obrębie transakcji. Jako parametr przyjmuje funkcję anonimową działającą na obiekcie sesji NHibernatea. By za inicjalizować komunikację należy przede wszystkim dostarczyć `connection string` będący ciągiem znaków określających adres serwera bazodanowego wraz z loginem i hasłem dostępu. Domyślnie wykorzystywany jest `connection string` testowy który powinien wskazywać na testową bazę danych.



Rys. Mechanizm dostępu do bazy przy użyciu klasy DataAccess

Każde rozpoczęcie operacji na bazie danych wymaga otwarcia sesji. Operację tę realizuje się poprzez wykonanie metody `OpenSession()`. Następnie w obrębie danej sesji należy wykonać operacje na bazie danych wykorzystując framework NHibernate.



Rys. Klasa Repository oraz QueryObject

Operacje wykonywane na bazie danych opakowane są w formie generycznej klasy `Repository<T>`, która opakowuje podstawowe operacje. Czasami istnieje potrzeba wykonania bardziej skomplikowanej operacji wtedy można wykorzystać klasę `Repository` wraz z obiektem zapytania `QueryObject`, który opakowuje ciąg znaków będący zapytaniem języka HQL.

```
public class QueryCourseByName : IQueryObject
{
    private readonly string _value;
    public QueryCourseByName(string value)
    {
        _value = value;
    }

    public string Query
    {
        get { return String.Format
            ("from CourseModel c where c.Name = '{0}'", _value);}
    }
}
```

Rys. Klasa QueryCourseByName opakowująca zapytanie języka HQL

Język HQL jest językiem bardzo podobnym do SQL-a. Wprowadza trochę uproszczeń ale zarazem usprawnia działanie zapytań ponieważ można operować na kolekcjach a nie tylko tabelach. Standardowe operacje wykonywane na SQL-u wymagające operacji JOIN mogą zostać zastąpione operacjami na kolekcjach. Przykład powyżej pozwala pobrać encje kursu na podstawie nazwy kurs , czyli tak naprawdę jest to standardowy where

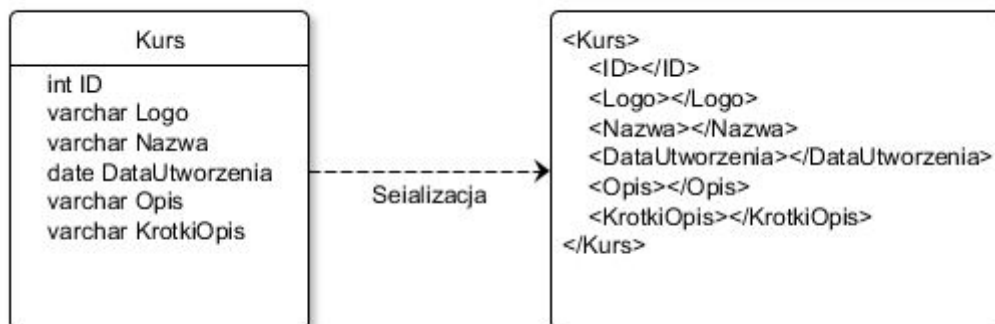
```
select lm.ID from LearningMaterialModel as lm ,
TestModel as test where test.ID = '{0}' and test
in elements(lm.Tests)
```

Rys. Skomplikowane zapytanie HQL

Można również tworzyć bardziej skomplikowane zapytania. Przypadek powyżej znajduje rodzica danego elementu. Jak widać użyta jest tutaj funkcja in elements która pozwala iterować po kolekcji wewnątrz pewnej encji pobranej z tabeli. W SQL-u w tym przypadku należałoby zrobić Join-a dwóch tabel. HQL jest szybki i bardzo sprawny.

5.5.2 Realizacja mechanizmów przetwarzania danych

Dane pobrane z bazy danych muszą być odpowiednio przetworzone. W przypadku usług sieciowych wystawionych na protokole SOAP w formacie XML, wykorzystywany jest serializowanie parametrów klasy do formatu XML. Każdy parametr oznaczony słowem kluczowym public ma tworzone element w dokumencie XML.



Rys. Serializacja klasa do XML

opisać proces parsowania xml, opisać proces parsowania json

Dane po pobraniu z bazy danych są w formie obiektów tzw proxy class udostępnianych przez NHibernate. Proxy klasa jest takim specjalnym wrapperem na konkretnej klasie zapewniającym takie funkcjonalności jak chociażby późna inicjalizacja elementów dzieci. Proxy klasy są spore.

opisać dlaczego stosujemy obiekty DTO, opisać dlaczego stosujemy obiekty signatury opisać problem z późną inicjalizacją w webservicach ze wymaga sesji i jest problem z przekazaniem sesji po web servicie. podjęte próby nie udało się.

Automapper z Entity do obiektu do DTO rozdział na różne modele i viewmodele. Parsowanie XML, JSON na wyjściu

5.5.3 Realizacja protokołu komunikacji

Opisanie konfiguracyjnych plików , endpointow

5.5.4 Realizacja mechanizmów zabezpieczeń

5.5.5 Realizacja zewnętrznego API - przykładowe zastosowanie

5.5.6 Realizacja mechanizmu dynamicznej zmiany serwera

5.6 Interfejs użytkownika - Moduły

5.6.1 Testy

5.6.2 Kursy

5.6.3 Dziennik ocen

5.6.4 Listy

Rozdział 6

Testowanie i ocena efektywności

6.1 Wybrane Testy Funkcjonalne

6.2 Testy Mechanizmów Zabezpieczeń

6.3 Analiza wydajności NHibernate-a w porównaniu do zwykłych zapytań SQL

proste zapytania skomplikowane zapytanie i porównanie czasów oraz ilości zapytań zgenerowanych

6.4 Testy wydajności mechanizmów przetwarzania danych

6.4.1 Analiza zapytań generowanych przez NHibernate za pomocą NHProfilea

6.4.2 Przykładowa optymalizacja zapytań

6.4.3 Testy obciążeniowe

6.5 Testy wydajności mechanizmów komunikacji sieciowej

6.5.1 Porównanie formatu Json , Xml

6.5.2 Badanie czasu odpowiedzi usług sieciowych

6.5.3 Testy konfiguracji rozłożenia usług sieciowych

6.6 Wnioski z testów i badań

Rozdział 7

Podsumowanie

Bibliografia

- [1] ADL. Scorm documentation. <http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/2004n/Documentation.aspx>.
- [2] A. Al-Ajlan, H. Zedan. E-learning (moodle) based on service oriented architecture. <http://www.cse.dmu.ac.uk/STRL/research/publications/pubs/2007/2007-23.pdf>.
- [3] A. Alkou, S. A.El-Seoud. Web services based authentication system for e-learning. *International Journal of Computing and Information Sciences*, 5(2):74–78, 2007.
- [4] T. Anderson, F. Elloumi. *Theory and Practice of Online Learning*. Athabasca University, 2004.
- [5] J. Bednarek, E. Lubina. *Kształcenie na odległość podstawy dydaktyki*. PWN, 2008.
- [6] E. Bertino, L. D. Martino, F. Paci, A. C. Squicciarini. *Security for Web Services and Service-Oriented Architectures*. Springer, 2010.
- [7] S. Carliner, P. Shank. *The E-Learning HandBook Past Promises , Present Challenges*. Pfeiffer, 2008.
- [8] E. Cerami. *Web Services Essentials - Distributed Applications with XML-RPC, SOAP, UDDI , WSDL*. O'Reilly, 2002.
- [9] A. Clarke. *E-learning : nauka na odległość*. WKŁ, 2007.
- [10] T. Earl. *Soa Design Patterns*. Prentice Hall, 2009.
- [11] B. Eyjen, S. Hanselman, D. Rader. *Asp.Net 3.5 z wykorzystaniem Csharp. Zaawansowane Programowanie*. Helion, 2010.
- [12] M. Fowler. *Architektura systemów zarządzania przedsiębiorstwem : wzorce projektowe*. Helion, 2005.
- [13] M. Fowler. *UML w kropelce*. LTP, 2005.
- [14] M. Fowler, K. Beck. *Refaktoryzacja : ulepszanie struktury istniejącego kodu*. WNT, 2006.
- [15] Z. Fryźlewicz, A. Salamon. *Podstawy architektury i technologii usług XML sieci Web*. PWN, 2008.
- [16] E. Gamma, R. Helm, R. Johnson, J. M. Vissides. *Wzorce Projektowe : elementy oprogramowania wielokrotnego użytku*. WNT, 2005.

- [17] A. Grewal, S. Rai, R. Phillips, C. C. Fung. The e-learning lifecycle and its services: The web services. *Proceedings of the Second International Conference on eLearning for Knowledge-Based Society*, 2005.
- [18] J. Górski. *Inżynieria Oprogramowania w projekcie informatycznym*. MIKOM, 2000.
- [19] M. A. Jab, H. K. Al-Omari. e-learning management system using service oriented architecture. *Journal of Computer Science*, 6(3):285–295, 2010.
- [20] C. McMurtry, M. Mercuri, N. Watling, M. Winkler. *Windows Communication Foundation Unleashed*. Sams, 2007.
- [21] X. Qi, A. Jooloor. Web service architecture for e-learning. *Systemics, Cybernetics and Informatics*, 3(5), 2006.
- [22] S. Resnick, R. Crane, C. Bowen. *Essential Windows Communication Foundation with .Net 3.5*. Addison Wesley, 2008.
- [23] M. Rosen, B. Lublinsky, K. T. Smith, M. J. Baler. *Service-Oriented Architecture and Design Strategies*. Wiley, 2008.
- [24] S. Schwinge. *Intelligent Information Technologies and Applications*, rozdział 1. Oakland University, 2008.
- [25] S. Weerawarana, F. Curbera, F. L. T. Storey, D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging*. Prentice Hall, 2005.