

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	1
1 Cel i zakres pracy	3
1.1 Cel pracy	3
1.2 Zakres pracy	3
2 Kształcenie na odległość	4
2.1 Definicja kształcenia na odległość	4
2.2 Korzyści płynące z używania systemów kształcenia na odległość	5
2.3 Ograniczenia kształcenia na odległość	5
2.4 Klasyfikacja systemów kształcenia na odległość	6
2.4.1 Systemy do zarządzania kursami(CMS)	6
2.4.2 Systemy wspomagające proces nauczania (LMS)	7
3 Wybrane technologie realizacji systemów webowych	8
3.1 Formaty danych JSON oraz XML	8
3.2 Protokoły SOAP oraz REST	11
3.3 Usługi sieciowe	15
3.4 Framework Asp.Net MVC	16
3.5 Mapowanie obiektowo relacyjne - NHibernate	18
4 Projekt systemu zdalnego nauczania	20
4.1 Wymagania projektowe	20
4.1.1 Wymagania funkcjonalne	20
4.1.2 Wymagania нефункционалне	23
4.2 Projekt bazy danych	25
4.2.1 Opis encji	25
4.2.2 Uproszczony model konceptualny (CDM) - struktura tabel i relacje	27
4.2.3 Model fizyczny bazy danych PDM	28
4.3 Architektura Systemu	29
4.3.1 Wybrane diagramy sekwencji funkcjonalności	30
4.3.2 Mechanizmy zabezpieczeń	33
5 Implementacja systemu zdalnego nauczania	35
5.1 Zewnętrzny hosting	35
5.2 Realizacja bazy danych	35
5.3 Realizacja aplikacji	36
5.3.1 Realizacja mechanizmów dostępu do bazy danych	36
5.3.2 Realizacja mechanizmów przetwarzania danych	39
5.3.3 Realizacja protokołu komunikacji	41

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	50
5.5	Interfejs użytkownika	52
6	Testowanie i ocena efektywności	55
6.1	Wybrane Testy Mechanizmów Zabezpieczeń	55
6.1.1	Sprawdzenie mechanizmów szyfrowania wiadomości	55
6.1.2	Sprawdzenie mechanizmu logowania oraz poziomów uprawnień	55
6.1.3	Zabezpieczenie przed atakami typu wstrzykiwanie skryptów	55
6.2	Testy wydajności mechanizmów przetwarzania danych	57
6.2.1	Wybrane testy funkcjonalne	58
6.2.2	Analiza oraz optymalizacja zapytań generowanych przez NHibernate za pomocą narzędzia NHProf	60
6.2.3	Testy obciążeniowe	62
6.2.4	Badanie czasu odpowiedzi usług sieciowych	66
6.2.5	Testy konfiguracji rozłożenia usług sieciowych	67
6.3	Wnioski z testów i badań	68
7	Podsumowanie	69
	Bibilografia	70

Wstęp

1.Dlaczego realizuje te prace 2.Aspekt badawczy 3.Aspekt naukowy

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 problem 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.

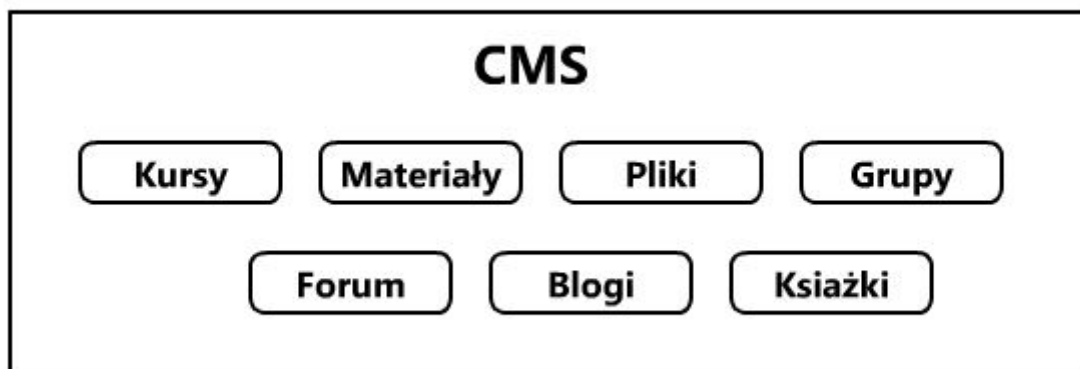
- konieczność przeszkolenia personelu dydaktycznego oraz uczniów.
- wdrożenie kosztownego systemu informatycznego obsługującego proces nauczania.
- wymagana większe zaangażowanie i samodzielna inicjatywa od uczącego się.
- konieczność dopasowania obecnego materiału dydaktycznego do nowych standardów.
- 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)



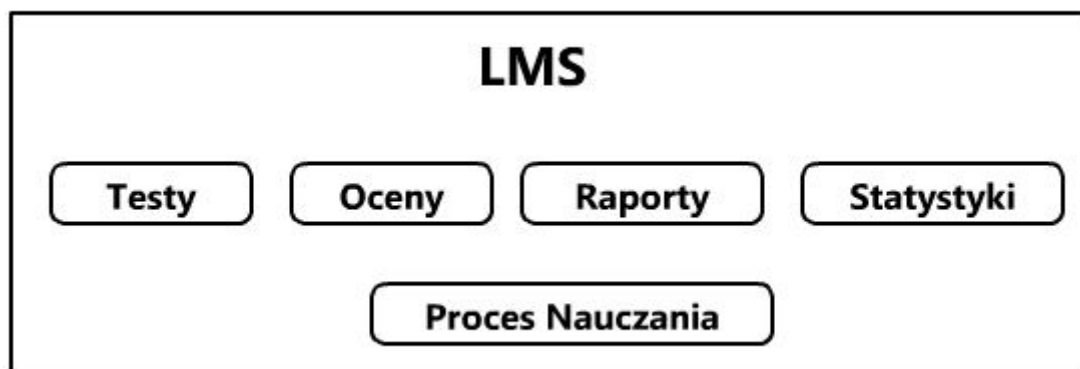
Rys.1 Podstawowe funkcje systemu klasy CMS

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.

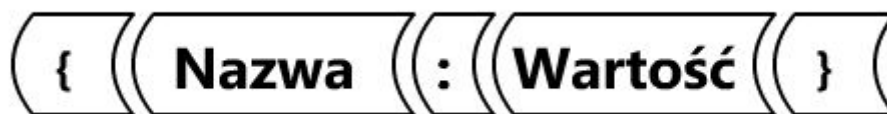
Rozdział 3

Wybrane technologie realizacji systemów webowych

3.1 Formaty danych JSON oraz XML

JSON

JSON (JavaScript Object Notation) jest lekkim formatem danych wykorzystywanym przy transferze danych po sieci. Jest to stosunkowo młody format danych opisany w 2006 roku w publikacji RFC4627[?] . Charakteryzuje się bardzo prostą ale zarazem czytelną reprezentacją danych. Zbiór danych składa się z pary ciągów znaków : nazwa , wartość.



Rys. Uproszczony Schemat formatu JSON

Najważniejszą cechą przemawiającą za używaniem formatu JSON jest jego prostota , czytelność oraz co najważniejsze ułatwione przetwarzanie danych w formacie JSON na obiekty języka Javascript za pomocą funkcji `eval()`". Cecha ta pozwala usprawnić proces tworzenia aplikacji webowych po stronie klienta tzn. z przeglądarki.

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 jako popularny format m.in plików konfiguracyjnych. Wiele skomplikowanych aplikacji wykorzystują format XML do definiowania m.in: scenariuszy testowych , interakcji w systemie ,przepływów danych. Na bazie tego formatu powstał również format XHTML , który łączy cechy formatu HTML oraz XML.

Format XML powstał na bazie formatu SGML , jest jego rozwinięciem. Dokument standaryzujący opisuje nie tylko sposób formatowania danych ale również opis u-standardyzowanego sposobu przetwarzania dokumentu.[?]



Rys. Uproszczony Schemat formatu XML

Cechy 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.

- bardzo duże wsparcie narzędzi i zgodność z wieloma systemami na rynku
- duże bezpieczeństwo przesyłanej 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 kandydatem , był już dobrze znany i miał spore wsparcie narzędzi oraz środowisk programistycznych.

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 w swoich zewnętrznych API.

Format XML charakteryzuje się większym poziomem bezpieczeństwa dzięki bardziej stabilnej i mocno typowanej semantyce. W przypadku formatu JSON nie ma możliwości określenia typu danych wartości ponieważ w jego specyfikacji brakuje możliwości definiowania atrybutów. Problem ten można pominąć dzięki zastosowaniu JSON schema określającego typy danych w oddzielnym dokumencie.

Format JSON może zostać bardzo łatwo przetworzony za pomocą funkcji eval dostępnej w języku programowania Javascript. W związku z tym przypadku aplikacji nadmiernie wykorzystujących ten sposób przetwarzania istnieje zwiększone ryzyko wstrzyknięcia niepożądanych skryptów.

Najważniejszą różnicą pomiędzy oboma formatami jest jednak ilość bajtów potrzebna do przesłania danych.

<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 oraz SOAP.

SOAP

Protokół SOAP jest protokołem transmisji danych wykorzystującym głównie format XML (Może również wykorzystywać inne formaty jednak XML jest tym najpopularniejszym). 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 segmentów.

- 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

Segment Header zawiera nagłówek informacji. Na wyżej zamieszczonym przykładzie widać m.in. adres do którego kierowana jest wiadomość oraz wartość uuid przydzielaną każdej wiadomości, która unikatowym identyfikatorem wiadomości. Segment Body zawiera przesyłane dane. W tym przypadku m.in. nazwę funkcji udostępnianej przez usługę sieciową wraz z parametrami.


```
<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 dopasowuje obie wiadomości i wie, że server odpowiedział na żądanie. Jak widzimy w tym przykładzie w segmencie body został przesłany wynik wykonania usługi.

Zalety SOAP

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

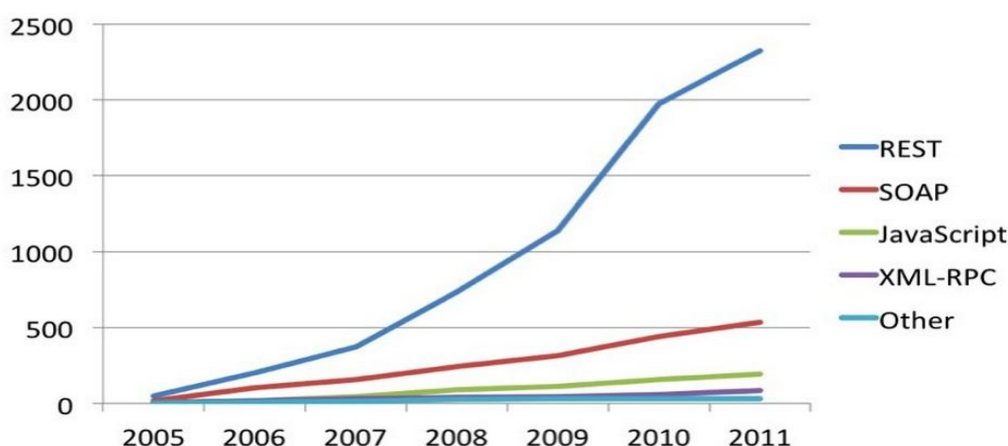
Wady SOAP

- przesyłanie nagłówek zwiększa ilość przesyłanych danych
- duży poziom komplikacji

REST

SOAP i jego rozszerzenia powodują, że staje się on dość skomplikowany poza tym wysyła wraz z zapytaniem do serwera dodatkowe dane w formie segmentów. W środowisku nowych małych firm zaistniała potrzeba na bardziej lekki protokół. Odpowiedzą na oczekiwania rynku stał się protokół REST. Popularność tego protokołu zaczęła znacząco rosnąć wraz pojawieniem się usługi mikro blogów Twitter oraz wystawienia przez tę usługę rozbudowanego API. Api Twittera używa Resta oraz formatu JSON do wystawiania wielu danych.

Badania z Maja 2011 roku przeprowadzone przez serwis ProgrammableWeb pokazują, że prawie 73% usług sieciowych wystawionych w formie API obsługiwane jest na bazie protokołu 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`. Jak widać jest to zwykłe zapytanie bez dodatkowych informacji w przeciwieństwie do protokołu SOAP, który potrzebuje wysłać całą wiadomość w formacie XML. W przypadku protokołu REST wszystkie istotne dane jak nazwa funkcji oraz parametr zawarte są w adresie URL.

```
{
  {
    "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

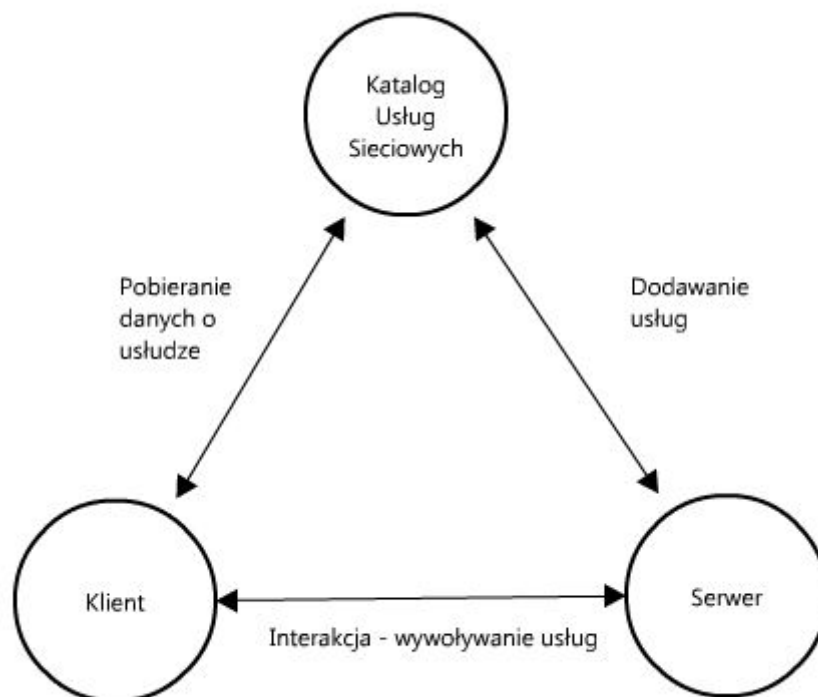
- prostota i lekkość ,nie ma potrzeby wysyłania dodatkowych danych
- czytelny

Wady REST

- prostota powoduje , że nie można tworzyć skompilowanych wiadomości
- dostępnych mniej opcji
- zmniejszone bezpieczeństwo

3.3 Usługi sieciowe

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



Rys. Uproszczona architektura usług sieciowych

W systemach rozproszonych możemy wyróżnić dwa typy urządzeń : klient oraz serwer. Klient rozpoczynając komunikację wysyła 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 i wysyła do serwera. Serwer interpretuje wiadomość wywołuje odpowiednią funkcję i zwraca wynik.

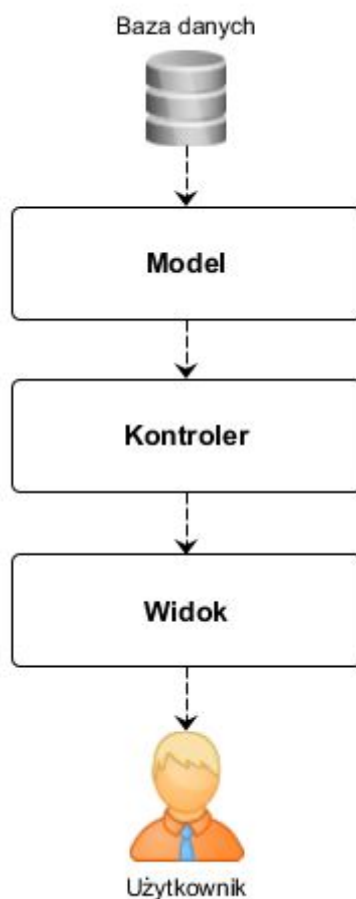
Dostępne usługi w katalogu opisane są w formacie WSDL (Web Service Description Language) , który bazuje na formacie XML.

3.4 Framework Asp.Net MVC

Asp.Net Mvc 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 oprogramowania webowego. Główną zaletą tego rozwiązania jest fakt że jest ono oparte na wzorcu projektowym MVC.

Wraz z ewolucją aplikacji webowych i ich poziomem 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[?]. Prawdziwa rewolucja zaczęła się w 2004 roku wraz z pojawieniem się nowych platform developerskich takich jak : Ruby on Rails na języku Ruby oraz Django związanego z językiem Python

Wzorzec ten wprowadza on podział aplikacji na trzy oddzielne warstwy : Model , Widok , Kontroler.



Rys. Uproszczona koncepcja wzorca MVC

Model

Reprezentuje warstwę danych , które mogą być zarówno w formie m.in. bazy danych , pliku. Warstwa ta jako jedyna ma dostęp do źródła danych. Dostęp do modelu jest jedynie możliwy z poziomu kontrolera , który korzysta z metod zdefiniowanych w warstwie modelu. W wielu aplikacjach pośrednio do komunikacji używa się dodatkowo warstwy usług.

Kontroler

Warstwa odpowiedzialna za sterowanie przepływem danych i przetwarzaniem tych danych do postaci wyświetlanej w warstwie widoku. Kontroler decyduje , który widok zostanie wyświetlony.

Widok

Reprezentuje warstwę bezpośrednio dostępną dla użytkownika systemu. Tworzony jest na podstawie modelu przekazanemu przez kontroler.

Zastosowanie wzorca MVC przy projektowaniu aplikacji webowej wymaga większego nakładu pracy w początkowej fazie projektu. Wymierne korzyści ze stosowania tego wzorca zaczynają być zauważane dopiero w późniejszych etapach życia projektu. Przede wszystkim zastosowane konwencje i 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.

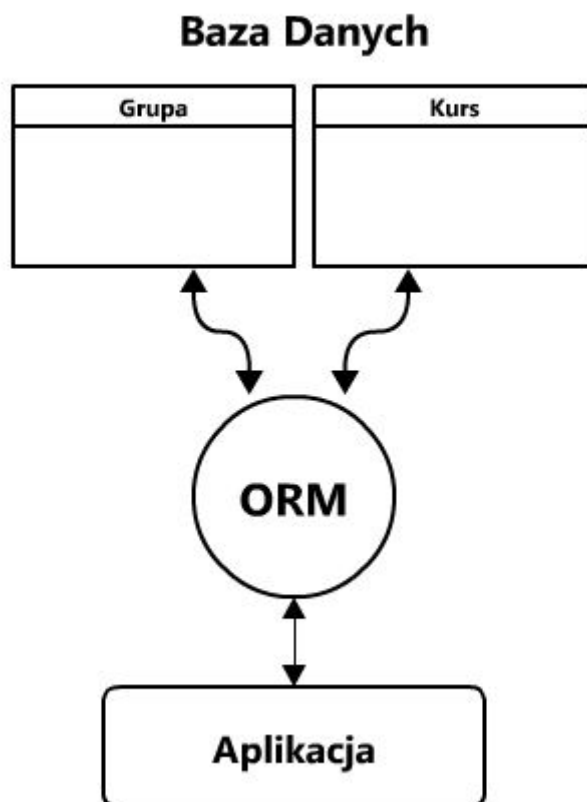
Opis komunikacji

Komunikacja w architekturze MVC rozpoczyna się od klienta zgłaszającego żądanie (np. o wyświetlenie danej strony www). Żądanie to jest przechwytywane przez kontroler. Jeżeli wygenerowanie strony www nie wymaga pobrania danych z źródła danych. Kontroler pobiera dany widok i przekazuje go klientowi. Jeżeli natomiast widok wymaga pobrania danych z bazy danych , realizowane jest połączenie z modelem za pośrednictwem , którego pobrane są dane i przekazany jest odpowiedni widok w formie wiadomości zwrotnej.

3.5 Mapowanie obiektowo relacyjne - NHibernate

Bazy danych są najważniejszą częścią systemu informatycznego. Implementacja dostępu do bazy jest jednym z bardziej czasochłonnych elementów realizacji projektu po stronie serwera. Dodatkowo logika ta jest szczególnie podatna na błędy. Bezpośrednie tworzenie zapytań stało się zbyt kosztowne oraz trudne w utrzymaniu. Z rozwiązaniami takimi wiąże się również inny problem, mianowicie dochodzi do niekompatybilności pomiędzy sposobem przedstawiania zależności pomiędzy obiektami między systemami relacyjnymi bazodanowymi a systemami opartymi na dziedziczeniu i kompozycji klas znajdującymi się w współczesnych platformach programistycznych.

By ułatwić proces tworzenia kodu coraz więcej firm wykorzystuje specjalne biblioteki wspomagające proces tworzenia warstwy dostępu do danych. Nazywane są one Obiektowo relacyjnymi mapperami (dalej zwanymi ORM-ami). Na rynku dostępnych jest wiele rozwiązań tego typu. Do najpopularniejszych w środowisku .Net należą m.in.: Entity Framework oraz NHibernate.

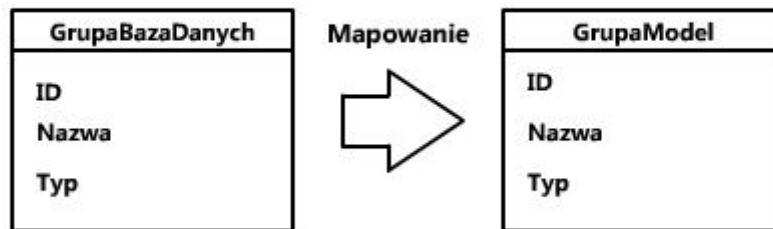


ORM obsługujący bazę danych

Dzięki zastosowaniu ORM-a można wprowadzać bardzo szybko zmiany oraz w dużo łatwiejszy sposób wykonywać odpowiednie zapytania na bazie danych nie przejmując się tak naprawdę warstwą bazodanową. Z punktu widzenia programisty posługującego się warstwą danych opartą na mapperze cała komunikacja jest schowana pod interfejsami. Dzięki temu można skupić się na implementacji konkretnej logiki związanej z funkcjonalnością aplikacji 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 części systemu wymagających optymalizacji. W tym przypadku profilowane są zapytania SQL. Zapytania wymagające optymalizacji można zamienić na zwykłe zapytania języka SQL. Dzięki takiemu rozwiązaniu oszczędza się czas oraz fundusze przeznaczone na projekt.

Proces mapowania sprowadza się do określenia, które pole z bazy danych ma być powiązane z pola obiektu wykorzystywanego w aplikacji. W ten sposób tworzone są specjalne klasy pośredniczące w komunikacji pomiędzy bazą danych a aplikacją.



Rysunek przedstawiający proces mapowania.

W pracy dyplomowej zdecydowałem się zastosować framework NHibernate. Jest to rozwiązanie będące implementacją frameworka Hibernate dostępnego w środowisku Javy na potrzeby środowiska .Net. 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.

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 oraz 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 zarządzane przez usuwanego użytkownika 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. 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 jedną lub więcej sekcji. Każda sekcja opisana jest tytułem oraz polem zawartości , w którym umieszczamy treść oraz materiały.

Obszar podsumowania

Zawiera segmenty

- Podsumowania - szybkie podsumowanie zdobytej wiedzy i najważniejszych rzeczy
- Więcej informacji - posiada zbiór dodatkowych materiałów poszerzającym zawartość kursu

- 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. Po skończeniu procesu nauczania 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 sprawdzania testów

Kursant po skończeniu procesu nauczania może przystąpić do rozwiązywania testu. Test składa się z dwóch obszarów : treści pytania oraz przydzielonych odpowiedzi. Po rozwiązaniu testu wyświetlana jest ocena oraz wiadomość informująca o zaliczeniu/nie zaliczeniu testu. Następnie ocena zapisywana jest do dziennika ocen danego ucznia.

Zarządzanie ocenami

Mazdy kursant może przeglądać listę swoich ocen otrzymanych w procesie nauczania 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 nakład pracy włożony w proces modyfikacji i utrzymania kodu był jak najmniejszy.

Bezpieczeństwo aplikacji

Dostęp do poszczególnych funkcjonalności systemu jest ograniczony w obrębie 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 wchodzącym w skład każdego kursu.

Mechanizm rejestracji oraz logowania zostanie zrealizowany za pomocą frameworka Asp.Net Membership zapewniającego rozbudowane i bezpieczne funkcjonalności. W tym celu powstanie druga baza danych , zawierająca dane wymagane w procesie 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 rozszerzeń WS-Security.

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 na protokole REST zapewniając dostęp do wybranych danych aplikacją nie powiązaną z platformą. By zaprezentować tę funkcjonalność zostanie stworzony dodatek do popularnej platformy "WordPress" pozwalający wyświetlić kursy danego autora na blogu.

Mechanizm dynamicznego przekierowania na serwer zapasowy

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 usługi sieciowe serwera zapasowego.

Tworzenie kopii zapasowych bazy danych

Każdego dnia o określonej porze będzie tworzona kopia zapasowa bazy danych. Jednocześnie 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+
- 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 Projekt bazy danych

4.2.1 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.

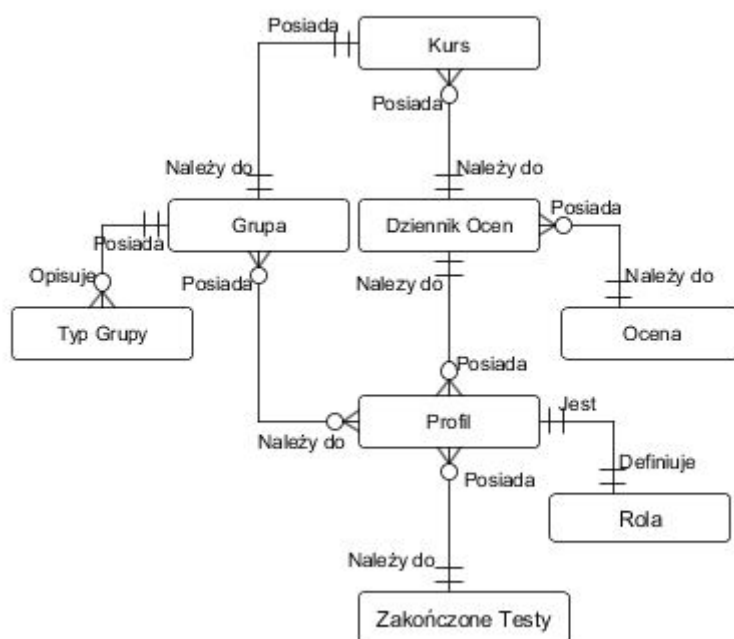
Profil

Reprezentuje użytkownika systemu.

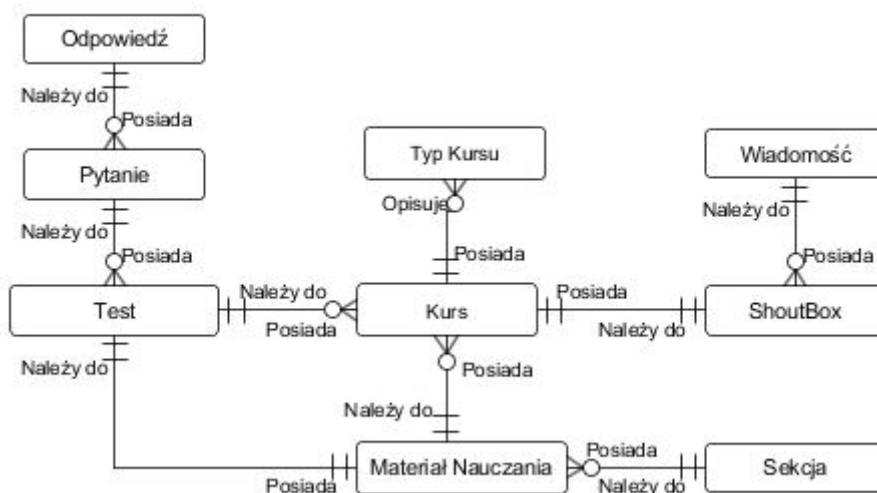
GrupaUzytkownikow

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

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

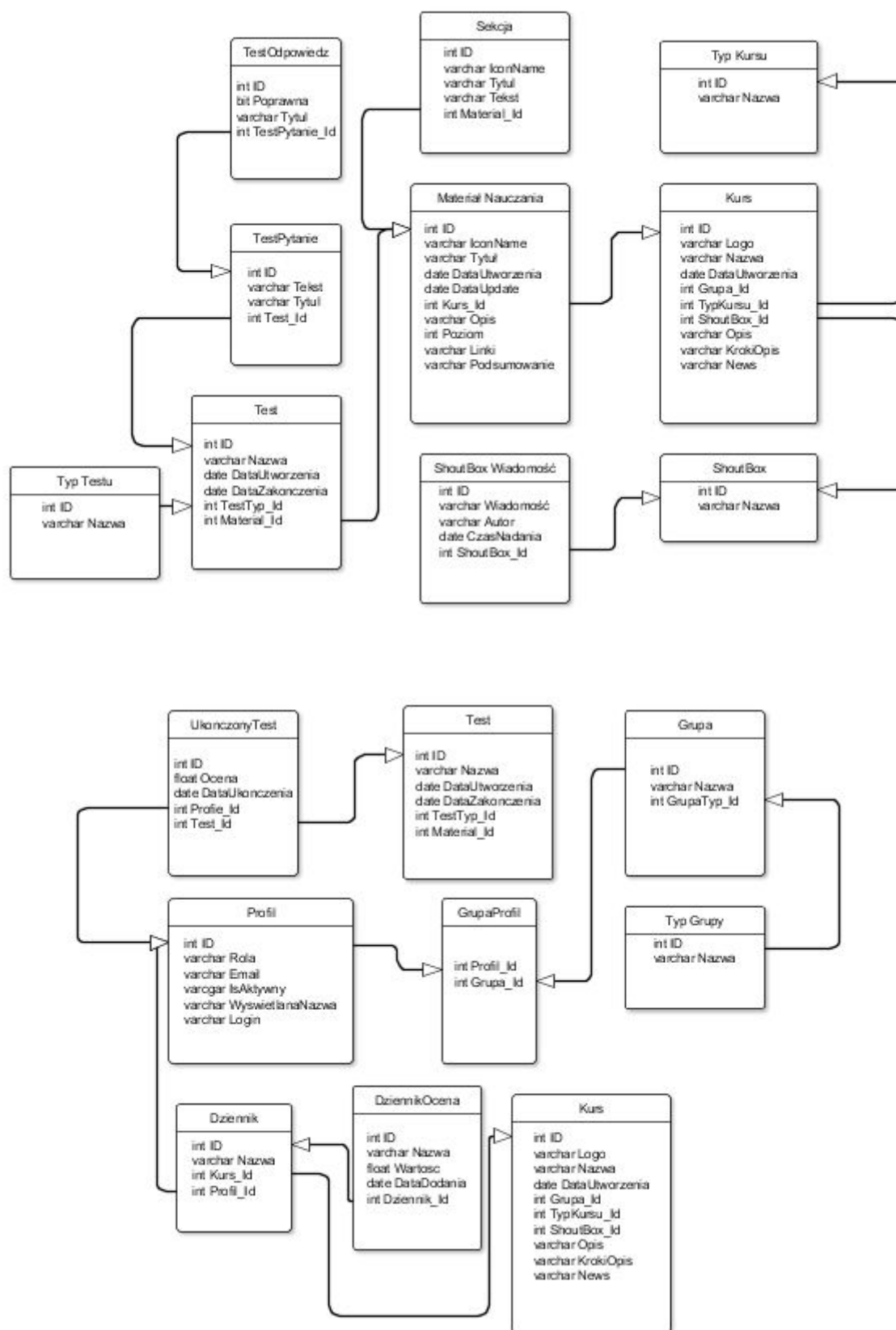


Rys. Relacje pomiędzy tabelami opisującymi profil

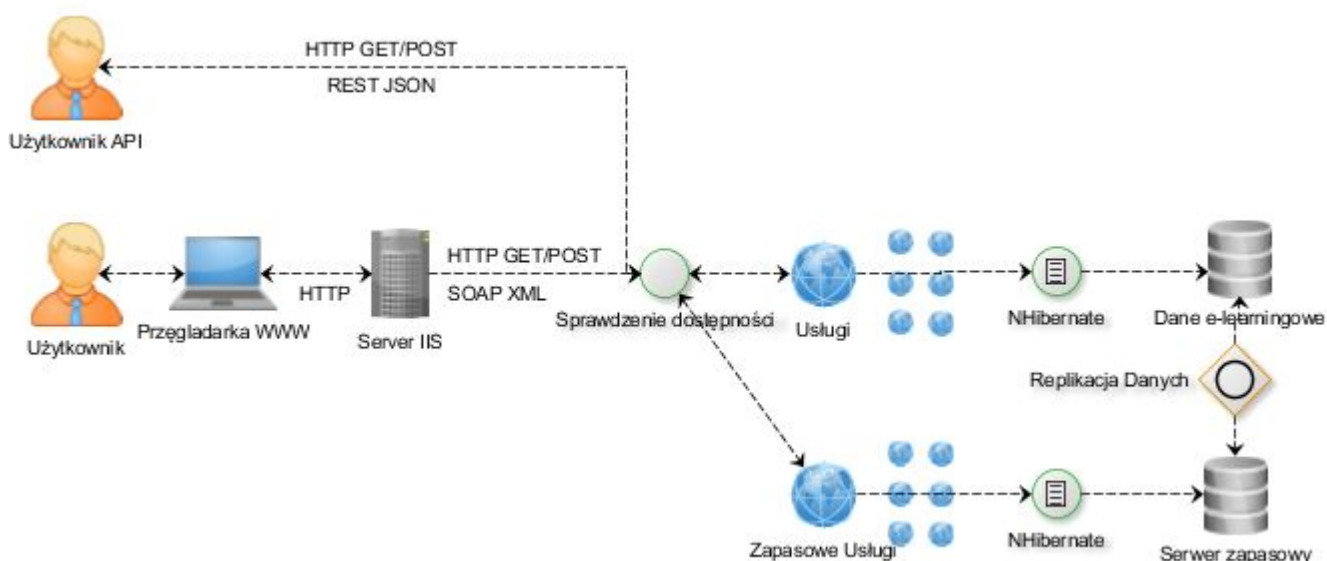


Rys. Relacje pomiędzy tabelami opisującymi kurs

4.2.3 Model fizyczny bazy danych PDM



4.3 Architektura Systemu



Rys. Schemat proponowanego rozwiązania

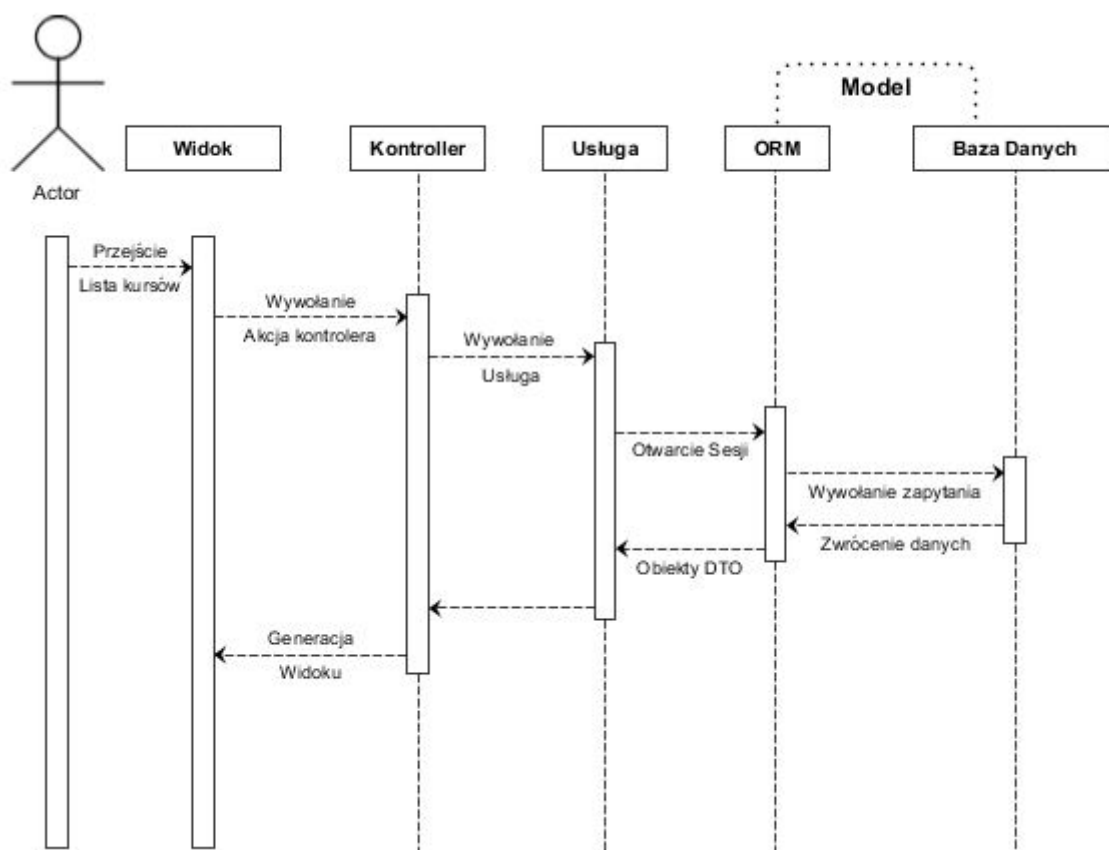
Proponowane rozwiązanie zakłada dwa sposoby dostępu do danych i aplikacji. Pierwszym z nich jest wykorzystanie aplikacji stworzonej w technologii Asp.Net Mvc postawionej na serwerze IIS. Dostęp taki realizowany jest za pomocą przeglądarki internetowej. Warstwa dostępu do bazy danych oraz usługi sieciowe są transparentne dla klienta korzystającego z przeglądarki internetowej. Po wykonaniu akcji przez klienta generowana 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.

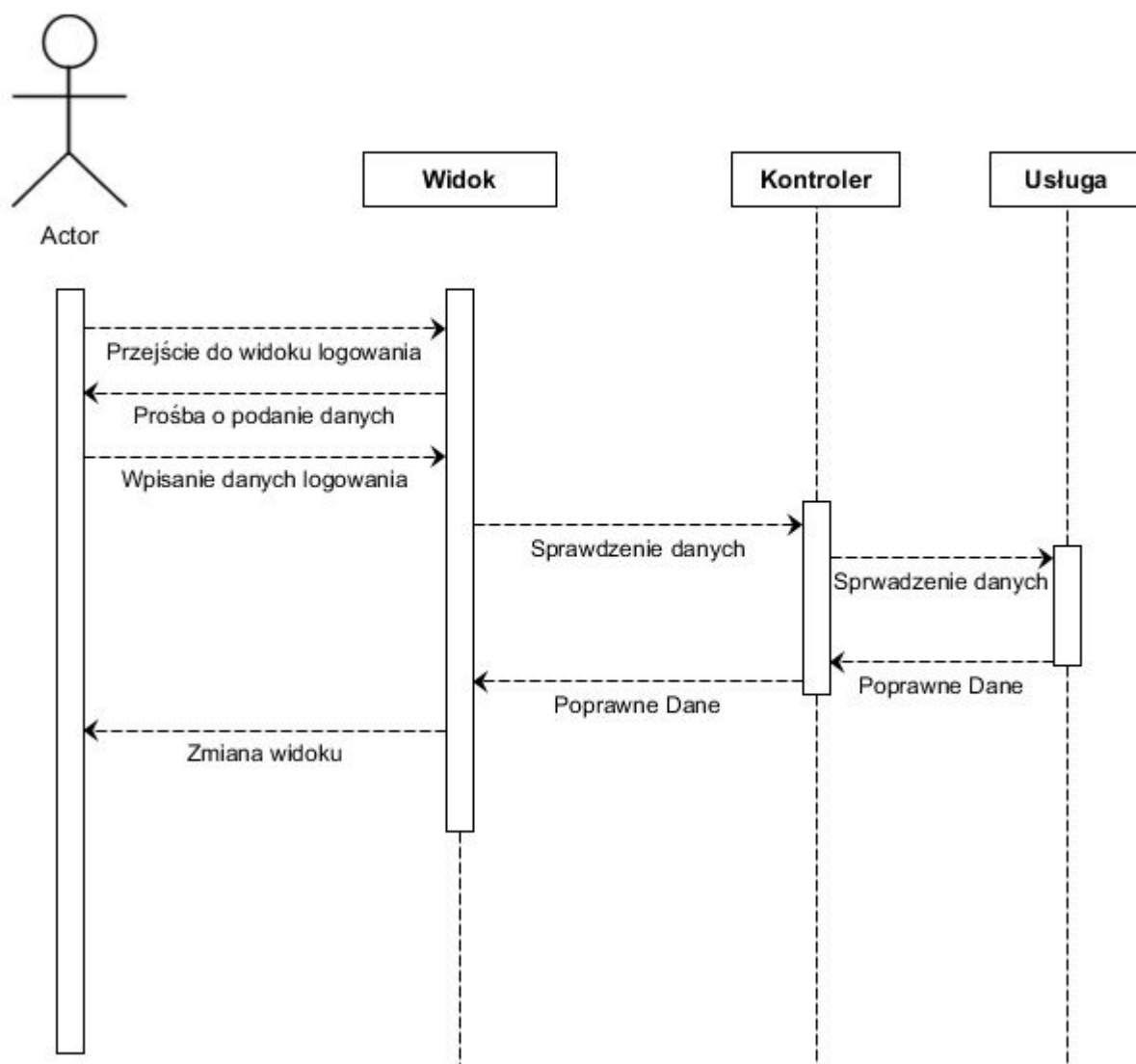
Komunikacja jest zrealizowana za pomocą frameworka WCF firmy Microsoft. Usługi sieciowe realizujące dostęp do bazy danych są podzielone na sześć usług dostępnych pod różnymi adresami URL

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.

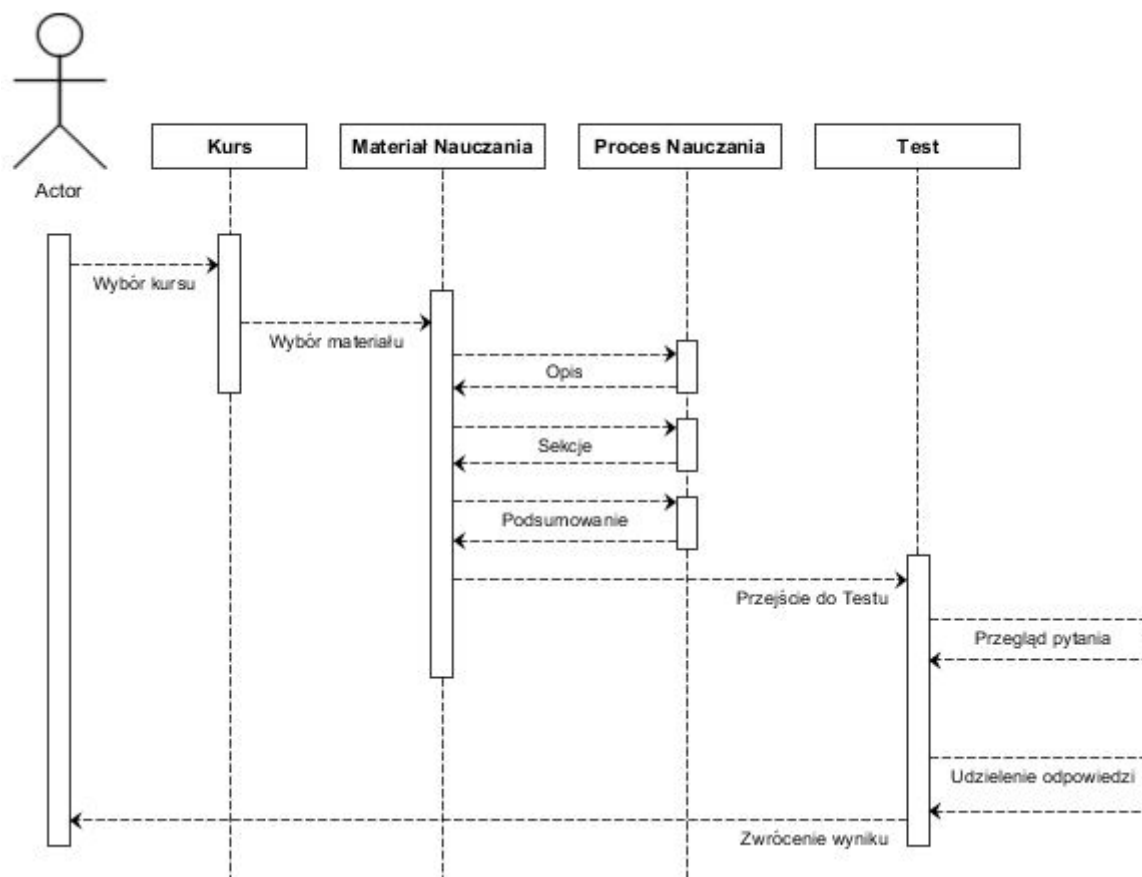
4.3.1 Wybrane diagramy sekwencji funkcjonalności



Rys. Diagram sekwencji generowania strony www



Rys. Diagram sekwencji logowania do aplikacji



Rys. Diagram sekwencji procesu nauczania

4.3.2 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.

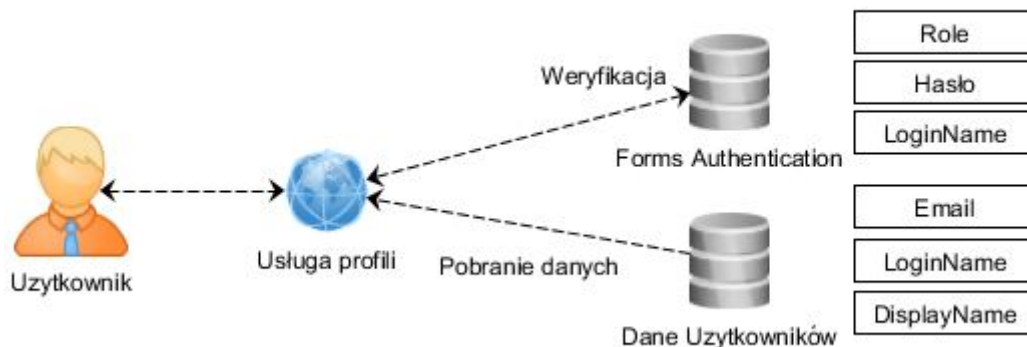
Wiadomości przesyłane protokołem SOAP z wykorzystaniem formatu danych XML będą już miały większy zakres zastosowanych zabezpieczeń. Wiadomości zawierające dane poufne jak dane logowania, hasła dane na temat użytkownika będą szyfrowane oraz przesyłane protokołem HTTPS po porcie 443 tzn pierw zostanie 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 wystawione są:

Dwie bazy danych jedna zawierająca dane logowania wspierająca framework Asp.Net Membership provider oraz druga baza danych zawierająca dane aplikacji czyli m.in. dane o kursach , materiałach nauczania.

Oraz dwie aplikacje:

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

Platforma została skonfigurowana by składować logi ,wydarzeń oraz błędów aplikacji, w odpowiednich katalogach. Dzięki temu istnieje możliwość szybkiego zdiagnozowania i poprawienia problemów aplikacji.

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.

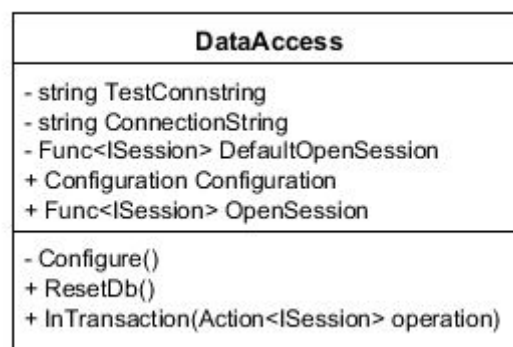
5.2 Realizacja bazy danych

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 aplikacji

5.3.1 Realizacja mechanizmów dostępu do bazy danych

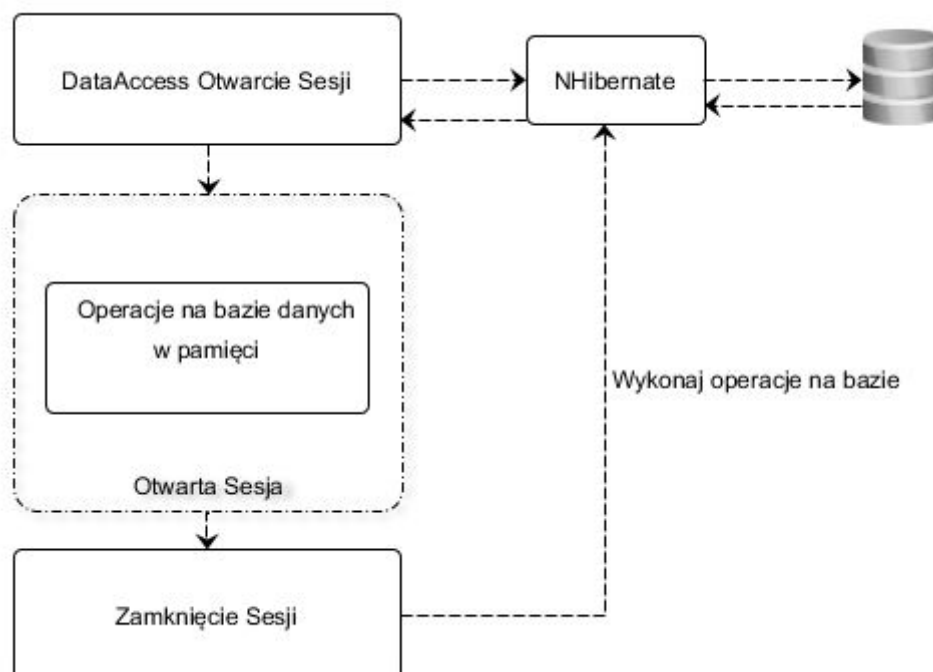
Dostęp do bazy danych opiera się na frameworku NHibernate. Cała komunikacja oraz inicjalizacja opakowana jest wewnątrz klasy `DataAccess`. Klasa ta wykorzystywana jest na serwerach udostępniających usługi sieciowe. Jest odpowiedzialna przede wszystkim za konfigurowanie połączenia z bazą danych poprzez mapper obiektowo relacyjny NHibernate. Ponieważ testy jednostkowe pokrywające bazę danych wykorzystują bazę danych generowaną w pamięci istnieje możliwość wstrzyknięcia odpowiedniej konfiguracji zmieniającej źródło danych na pamięć serwera testowego.



Rys. Diagram klasy `DataAccess`

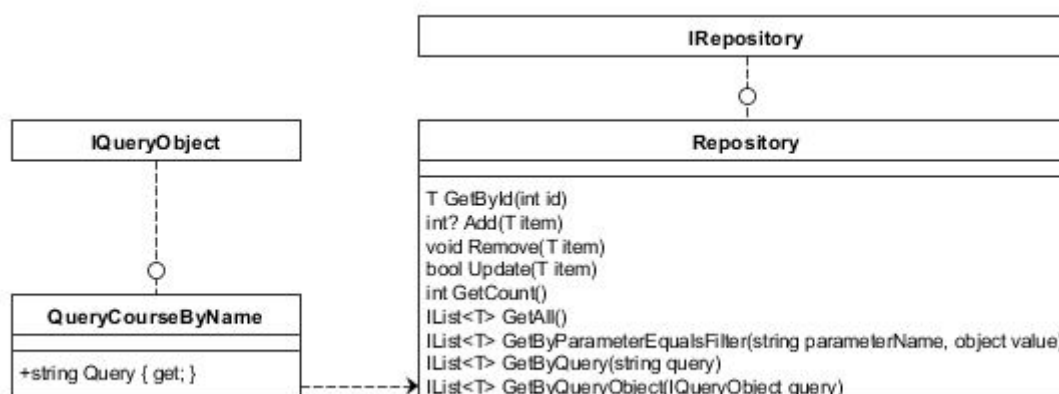
Klasa ta zawiera metody pozwalające zresetować bazę danych tzn. usunąć wszystkie tabele oraz dane i wygenerować nową strukturę bazy wraz z danymi testowymi. Posiada ona również metodę `InTransaction()` która pozwala przeprowadzić operację na bazie w obrębie transakcji. Metoda ta jako parametr przyjmuje wskaźnik do funkcji opakowany w specjalnej klasie dostępnej na platformie .Net. Język Csharp nie posiada jawnego wsparcia dla wskaźników (istnieje możliwość uruchomienia wsparcia kosztem wyłączenia pewnego segmentu kodu z automatycznego zarządzania stertą poprzez mechanizm "garbage collector") dlatego stosuje się klasy opakowywujące. Istnieje również możliwość przesłania funkcji anonimowej podobnie jak w języku programowania Javascript. Podobny mechanizm został zastosowany w parametrze `OpenSession` który również zwraca opakowany wskaźnik do funkcji. Dzięki takiemu rozwiązaniu istnieje możliwość podmienienia logiki otwarcia sesji. Szczególnie jest to przydatne w przypadku implementacji testów jednostkowych operujących na bazie danych w pamięci.

Do otwarcia połączenia z serwerem wymagany jest ciąg znaków zwany "connection stringiem". Ciąg ten zawiera adres serwera bazodanowego, nazwę bazy danych oraz informacje wymagane w procesie logowania do serwera. Domyślnie zdefiniowany jest ciąg przekierowujący na testową bazę danych. By dostarczyć inny ciąg należy zmodyfikować odpowiednio plik konfiguracyjny aplikacji webowej ("web.config").



Rys. Mechanizm dostępu do bazy przy użyciu klasy DataAccess

Rozpoczęcie operacji na bazie danych wymaga otwarcia sesji. Operację tę realizuje się poprzez wykonanie metody **OpenSession()**. Funkcja ta zwraca wskaźnik do funkcji, która zwraca klasę sesji frameworka **NHibernate**. Obiekt sesji wymagany jest do przeprowadzania wszystkich operacji na bazie danych. Klasa ta jest implementacją wzorca projektowego **Unit Of Work**, który jest bardzo wygodnym sposobem zarządzania takim zasobem jak sesja.



Rys. Klasa Repository oraz QueryObject

Korzystanie bezpośrednio z obiektem sesji jest bardzo wygodnym rozwiązaniem. Jednakże w przypadku bazy danych wystawionej za warstwą usług sieciowych zarządzanie sesją dostępu do bazy danych jest bardzo skomplikowanym zagadnieniem. Po próbach przesyłania obiektu sesji m.in. przy wykorzystaniu rozwiązania **NHibernate Remoting** zdecydowałem się opakować wykorzystywanie sesji w klasach będących implementacją

wzorca projektowego **Repozytorium**.

Operacje wykonywane na bazie danych opakowane są w formie generycznej klasy `Repository<T>`. Dostarcza ona podstawowych metod pozwalających realizować operację z zakresu **CRUD** (Tworzenie , Czytanie , Modyfikacja , Usuwanie). Wykonywanie bardziej skomplikowanych zadań można zrealizować za pomocą metody **GetByQuery** bądź **GetByQueryObject** . Pierwsza metoda przyjmuje jako parametr ciąg znaków będący zapytaniem języka HQL. Druga metoda przyjmuje klasę implementującą interfejs **IQueryObject**. Jest to specjalny interfejs pozwalający zdefiniować obiekty opakowujące zapytania języka HQL w ściśle typowanych klasach. Dzięki takiemu zabiegowi programista nie operuje bezpośrednio na znakach ale na klasach. Rozwiązanie to pozwala stworzyć kod lepszej jakości.

```
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

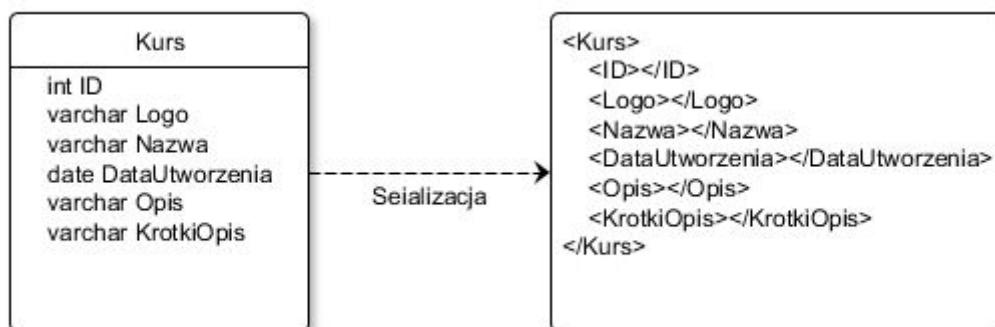
Język HQL jest językiem przypominającym składnią standardowy język zapytań SQL. Wprowadza trochę uproszczeń ale zarazem usprawnia działanie zapytań poprzez mechanizmy pozwalające operować na obiektach i ich kolekcjach. SQL operuje na tabelach i modelu relacyjnym natomiast HQL opiera swoje operacje na obiektach i ich kolekcjach zorientowanych obiektowo. Przykład powyżej pozwala pobrać encje kursu na podstawie jego nazwy. Jest to zapytanie podobne do prostego zapytania typu **SELECT** z klauzulą **WHERE** w przypadku języka SQL.

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

Rys. Skomplikowane zapytanie HQL

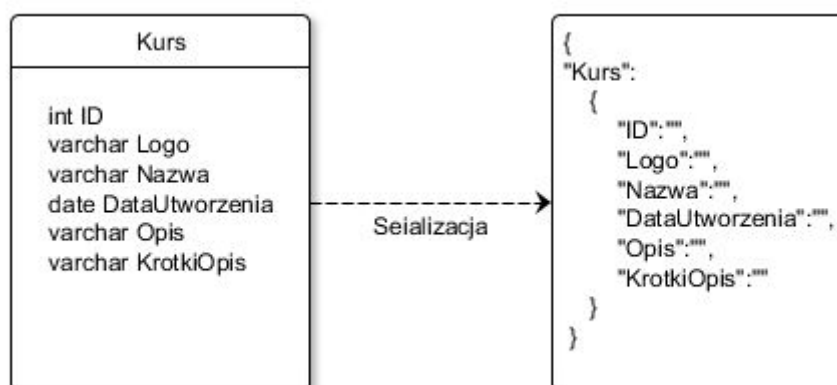
HQL pozwala również przeprowadzać bardziej skomplikowane operacje. Kod zaprezentowany na rysunku powyżej pozwala znaleźć numer ID rodzica podanego obiektu. W tym przypadku wykorzystywana jest funkcja "in elements" która pozwala przeszukiwać kolekcję danego elementu.

5.3.2 Realizacja mechanizmów przetwarzania danych



Rys. Przykład przekształcenia do formatu XML

Przed wysłaniem danych pobranych z bazy danych do aplikacji muszą być one przetworzone do odpowiedniego formatu. W przypadku usług sieciowych wystawionych na protokole SOAP korzystających z formatu XML, wykorzystywany jest mechanizm serializowania wszystkich publicznych parametrów klasy. Element główny tworzony jest na podstawie nazwy klasy. Elementy należące do elementu głównego zwane jego dziećmi zawierają odpowiednio nazwę parametru oraz jego wartość. W projekcie użyto standardowego serializera dostępnego wraz z platformą .Net. Konfiguracja serializacji sprowadza się do oznaczenia specjalnymi atrybutami parametrów danej klasy.

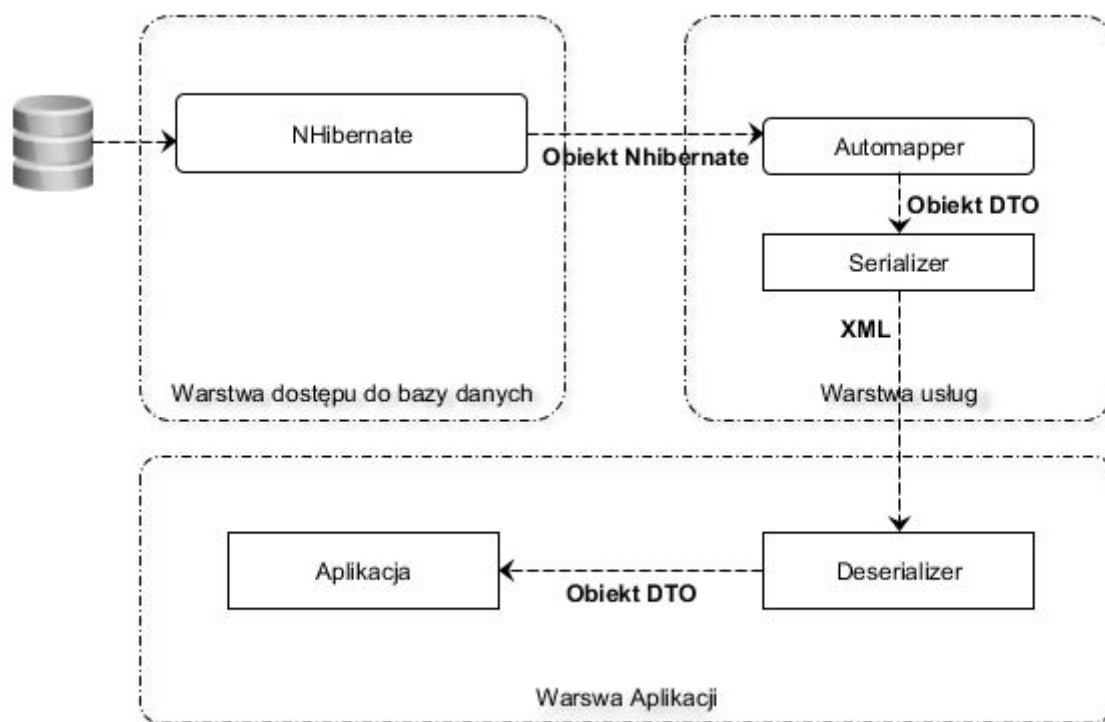


Rys. Przykład przekształcenia do formatu JSON

Dane wystawione na zewnątrz aplikacji przy użyciu protokołu **REST**, przetwarzanie są do formatu danych **JSON**. Jest to lekki i bardzo czytelny format.

Przed przetworzeniem danych do odpowiedniego formatu, wykonywany jest proces ódchudzania"klas tzn. przetwarzania klasy ogólnej na klasę szczególną zawierającą tylko wymagane parametry oraz dane. Proces ten jest bardzo ważny w przypadku dostępu do bazy danych realizowanego za pomocą mappera obiektowo relacyjnego takiego jak **NHibernate**. Framework ten pobierane dane opakowuje w specjalne klasy implementującej wzorzec projektowy **"Klasa proxy"**. Wzorzec ten rozszerza możliwości klasy dodaje dodatkowe funkcjonalności takie jak możliwość przeprowadzenia późnej inicjalizacji. Klasa

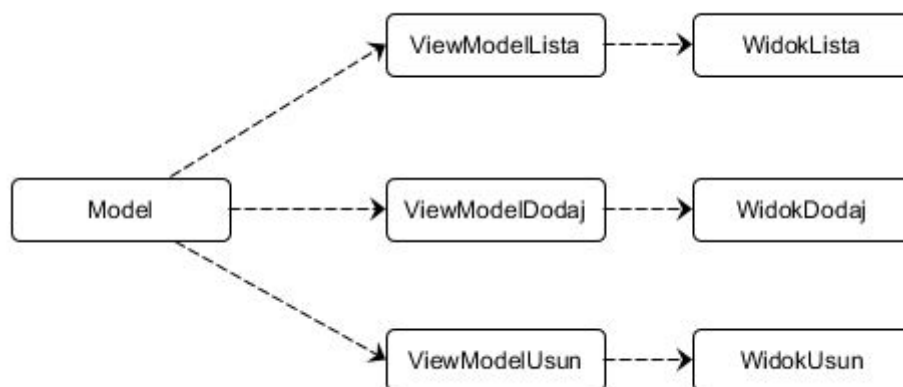
generowana przez NHibernate zawiera wiele dodatkowych metod oraz pól. Dodatkowo wszelkie kolekcje .Netowe przekształcane są w specjalne kontenery dostępne jedynie z poziomu aplikacji mającej dostęp do biblioteki NHibernate. Zgodnie z założeniami jedynie serwer ma mieć powiązanie z tym frameworkiem. Zadaniem klienta jest jedynie wywoływanie metod wykorzystujących podstawowe obiekty frameworka .Net Klient nie może zależeć od implementacji warstwy bazodanowej.



Rys. Graficzne przedstawienie procesu przetwarzania danych.

Proces przetwarzania obiektów generowanych przez framework NHibernate na obiekty przesyłane po sieci tzw. DTO ("Data Transport Object") realizowany jest przy pomocy biblioteki **AutoMapper**. Dostarcza ona funkcjonalności usprawniających żmudny proces przetwarzania jednego typu danych w drugi. By móc korzystać z **AutomMapper** na początku konfiguruje się mapowania (podobnie jak w przypadku mapowań powiązanych z frameworkiem NHibernate).

W aplikacji każdy model reprezentujący encje bazodanową posiada klasę odpowiednią klasę **DTO**. Klasy **DTO** dziedziczą po jednej wspólnej generycznej klasie bazowej dostarczającej metody pozwalające w łatwy sposób wywoływać metody dostarczane przez framework **AutoMapper** przekształcające jedną klasę w drugą.



Rys. Przekształcanie obiektów typu ViewModel.

Niemniej istotnym czynnikiem przemawiającym za ódcchudzaniem"klas jest ilość danych przesyłana po sieci. Przesyłanie całych obiektów z danymi, z których nie będziemy korzystać byłoby bardzo niewydajnym i kosztownym rozwiązaniem. Rozważmy przypadek wyświetlania listy kursów. Kurs jest rozbudowanym obiektem posiadającym wiele parametrów nie tylko opisujących kurs ale również zawierających kolekcje Materiałów Nauczania, Grupę itd. Do wyświetlenia listy kursów potrzebujemy jedynie jego parametrów opisowych. W takim przypadku przesyłanie całej zawartości kursu byłoby mało wydajne. By ograniczyć ilość przesyłanych danych stosuje się tzw ViewModele (Modele Widoku). Są to klasy specjalnie dopasowane do widoku, w którym zostaną wyświetlone. View Modele mogą dodatkowo łączyć wiele różnych encji.

5.3.3 Realizacja protokołu komunikacji

Komunikacja pomiędzy aplikacją a bazą danych oparta jest na rozwiązaniu **WCF (Windows Communication Foundation)**. Jest to najnowszy framework firmy Microsoft pozwalający tworzyć aplikacje wykorzystujące usługi sieciowe. Dzięki zastosowaniu tej technologii można tworzyć bardzo łatwo konfigurowalne mechanizmy przetwarzania danych przy wykorzystaniu usług sieciowych. **WCF** nie tylko jest nowym frameworkiem ale również pozwala korzystać z wcześniejszych rozwiązań dostępnych na platformie .Net: ASMX, COM+, .Net Remoting.

Konfiguracja frameworka **WCF** odbywa się poprzez zdefiniowanie bindigów oraz endpointów.

```

<service behaviorConfiguration="DefaultELearnServices" name="ELearnServices.CourseService">
  <endpoint name="jsonEP" address="json" binding="webHttpBinding" bindingConfiguration="Non
  <endpoint address="" binding="basicHttpBinding" contract="ELearnServices.ICourseService"
    <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
  <host>
  <host>
  <baseAddresses>
    <add baseAddress="http://codedashservices.mfranc.com/" />
  </baseAddresses>
</host>
</service>

```

Rys. Ustawienia usługi po stronie serwera.

Usługa po stronie serwera zdefiniowana jest za pomocą endpointów. Są to węzły pozwalające konfigurować sposób ,adres oraz protokół pod jakim będą dostępne usługi sieciowe. W zaprezentowanym przykładzie zdefiniowane są trzy endpointy.

Pierwszy nazwany JsonP jest ednpiointem wystawiającym daną usługę po protokole REST w formacie JSON. Konfiguracja tego endpointa ustawia brak zabezpieczeń tzn dane nie są szyfrowane. Jako binding używany jest **webHttpBinding** jest to binding wystawiający usługę sieciową dostępną z poziomu zwykłych zapytań protokołu HTTP bez opakowywania żądań w ramki SOAP.

Drugi endpoint definiuje dostęp z poziomu protokołu SOAP. Mechanizm ten realizowany jest poprzez binding **basicHttpBindig**

Trzecim endpointem jest wystawienie katalogu usług w postaci wiadomości WS-MEX (WS-MetadataExchange). Jest to endpoint do , którego zwraca się klient w momencie gdy chce zbadać jakie usługi są dostępne na danym serwerze

Po stronie klienta definiujemy własny binding i ustawiamy odpowiedni endpoint.

```
<binding name="WSHttpBinding_ICourseService" closeTimeout="00:01:00"
  openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
  allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
  maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
  messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
  useDefaultWebProxy="true">
  <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
    maxBytesPerRead="4096" maxNameTableCharCount="16384" />
  <security mode="None">
    <transport clientCredentialType="None" proxyCredentialType="None"
      realm="" />
    <message clientCredentialType="UserName" algorithmSuite="Default" />
  </security>
</binding>
```

Rys. Ustawienia bindingu po stronie klienta.

Binding po stronie klienta pozwala zdefiniować szereg opcji , konfigurującymi wygląd żądań słanych do serwera z usługami sieciowymi.

```
<endpoint address="http://codedashservices.mfranc.com/CourseService.svc" binding="basicHttpBinding"
  bindingConfiguration="WSHttpBinding_ICourseService" contract="CourseService.ICourseService"
  name="WSHttpBinding_ICourseService" />
```

Rys. Ustawienia endpoint po stronie klienta.

Najważniejszymi częściami definicji endpointa po stronie klienta jest wskazanie adresu oraz bindingu , który określa w jakiej postaci oczekujemy danych.

Przy komunikacji SOAP , która w tym przypadku jest ściśle powiązana typami. Tzn serializer po stronie serwera i deserializer po stornie klienta operują na tych samych typach

i muszą być zgodne w wersji. Klasami tymi są obiekty DTO. Takie rozwiązanie wprowadza pewne ograniczenia ale za to usprawnia proces tworzenia systemu. Obiekty DTO są prostymi klasami kontenerowymi służącymi jedynie do transferu danych więc uznałem, że takie rozwiązanie będzie przynosiło więcej plusów niż minusów.

```
$.ajax({
  type: "GET",
  url: "http://codedashservices.mfranc.com/CourseService.svc/json/Get",
  data : "id=16",
  datatype: 'json',
  success: function(data) {
    $.each(data , function(index,item) {
      //Przetwarzanie danych
    });
  },
  error : function(data) {
    alert('error');
  },
  statusCode: {
    404: function() {
      alert('page not found');
    }
  }
})
```

Rys. Ustawienia endpoint po stronie klienta korzystającego z API.

W przypadku danych wystawionych przez protokół REST dostępnych dla innych klientów sytuacja jest trochę inna. Klient nie posiada informacji do jakich obiektów deserializować dane. Nie musi posiadać biblioteki z obiektami DTO. Klient nie potrzebuje specjalnej konfiguracji i nie potrzebuje być nawet powiązany z technologią .Net. W tym przypadku dane otrzymywane przez klienta przekazywane są w czystej nie przetworzonej postaci. Jedynie co jest potrzebne do konfiguracji to adres serwera z usługami sieciowymi. W zaprezentowanym kodzie wykorzystuje funkcję popularnej biblioteki **JQuery** wspomagającej funkcjonalności języka **JavaScript**.

5.4 Wykorzystane narzędzia

5.4.1 Mechanizm logowania zdarzeń - NLog

W procesie wytwarzania oprogramowania bardzo ważnym aspektem zwiększającym znaczącą powodzenie projektu jest zapewnienie odpowiedniego mechanizmu logowania zdarzeń oraz błędów występujących wewnątrz aplikacji. Dzięki zaimplementowaniu takiego mechanizmu, przyspieszamy proces naprawy oraz identyfikacji błędów. Dodatkowo możemy analizować poprawność działania aplikacji poprzez analizę wiadomości zwracanych przez system. 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 pozwalająca tworzyć mechanizmy logowania.

By podczepić mechanizm logowania do danej klasy wystarczy stworzyć wewnątrz niej element statyczny będący instancją klasy **NLog.Logger**. Dla każdej klasy, która chcemy objąć mechanizmem logowania tworzymy oddzielną instancję loggera.

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

Po zainicjalizowaniu klasy Loggera wystarczy, że w interesującym nas miejscu wywołamy określoną metodę reprezentującą rodzaj logowanej wiadomości.

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

Nlog udostępnia m.in rodzaje wiadomości typu:









- Informacja
- Debugowanie
- Błąd
- Błąd Krytyczny

Prócz wywołania NLoga bardzo istotną rzeczą jest odpowiednie skonfigurowanie pliku konfiguracyjnego aplikacji. W pliku konfiguracyjnym możemy bowiem ustalić miejsce do którego chcemy zapisywać logi. Do wyboru mamy m.in:

- Plik
- Adres E-mail
- Wysłanie 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 analizować wysyłane logi. Dzięki takiej konfiguracji przyspieszył się proces implementacji ponieważ miałem wgląd w proces wykonywania akcji wewnątrz aplikacji.

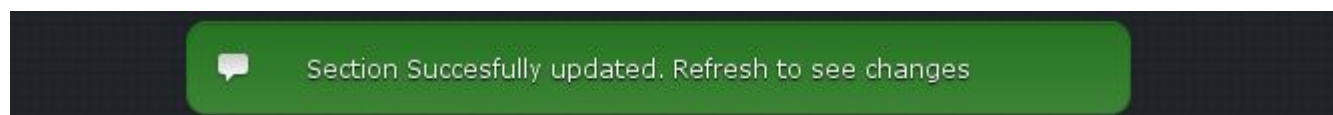
Log viewer			
Type	Date/Time	System	Description
	20:52:28	elearn.MvcApplication	Application Started
	20:52:38	elearn.Session.SessionState	Adding Current user data to session state
	20:52:44	NHiberanteDal.DTO.DtoMappings	DTO Mappings Initialized
	20:52:45	NHiberanteDal.DTO.DtoMappings	Problem initializing 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 były wysyłane w formie wiadomości email z logiem na określony adres 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



Rys. Przykładowy alert

Prócz generowania logów po stronie serwera 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 wystąpienie problemu z połączeniem.

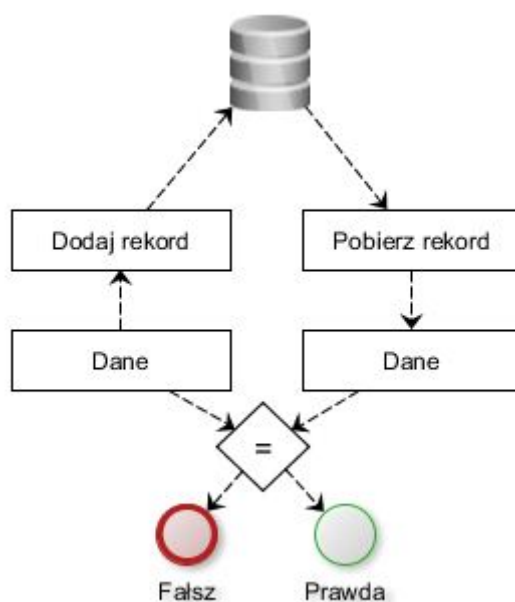
5.4.2 Testy jednostkowe - NUnit

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

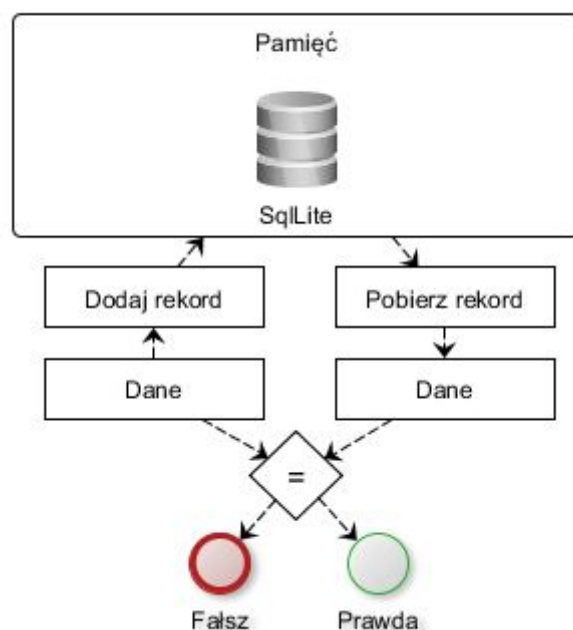
Testowanie jednostkowe bazy danych jest dość skomplikowanym zagadnieniem. Testowanie jednostkowe zakłada wykonywanie testów tylko pojedynczej funkcjonalności. W przypadku bazy danych byłaby to np. operacja dodawania rekordu do bazy. Procedura przeprowadzenia takiego testu polega na:

- Wywołanie metody dodawania rekordu do bazy danych.
- Wywołanie metody pobierania rekordu z bazy danych.
- 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**, która jest bardzo popularnym darmowym rozwiązaniem.



Rys. Test jednostkowy z bazą danych w pamięci

Przed każdym rozpoczęciem testu generowana jest baza danych oraz wypełniana jest danymi potrzebnymi przy testach. W przypadku np. testowania elementu "Kurs" posiadającego listę elementów "Test". Generowany jest kurs w przykładową listą testów. Ponieważ wykorzystuję mapper relacyjno obiektowy "Hibernate", przed rozpoczęciem testów bazy danych przeprowadzany jest test konfiguracji frameworka.

Testy bazy danych obejmują :

- Testy generycznej klasy repozytorium
- Testy zapytań wykorzystujących język HQL

Baza danych pokryta jest łącznie ponad 90 testami jednostkowymi.

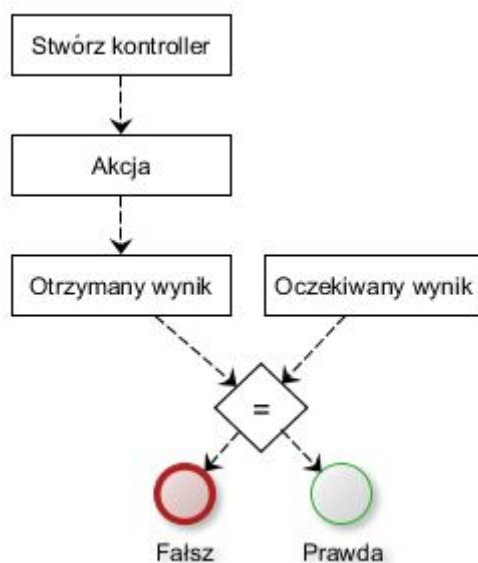
Testy kontrolerów - RhinoMocks

Testy jednostkowe kontrolerów są bardziej złożone niż testy wykonywane na warstwie dostępu do bazy danych. Kontroler bardzo często komunikuje się z warstwą bazodanową wyciągając dane potrzebne do wygenerowania widoku. Baza danych jest ich zewnętrzną zależnością. Testowanie dostępu do bazy danych nie jest sensem testów kontrolera. Test kontrolera musi testować jedynie proces przetwarzania danych pozyskiwanych z bazy danych. W tym przypadku najlepszym rozwiązaniem byłoby w ogóle pominięcie bazy danych i wstrzyknięcie danych testowych na których przetestujemy zachowanie kontrolera. Na szczęście istnieje taka możliwość i do tego celu stosuje się specjalne obiekty zwane **Moc-kami/Stubami**. W niniejszej pracy wybrałem pierwszy rodzaj obiektów i posłużyłem się frameworkiem RhinoMocks do ich generacji.

Test kontrolera realizowany jest podobnie jak test bazodanowy.

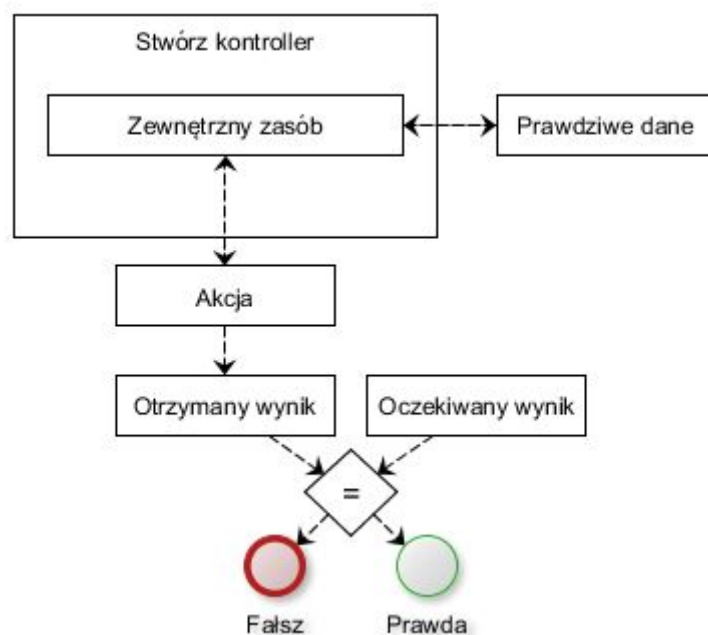
- Tworzymy dany kontroler wywołując jego konstruktor.

- Wywołujemy metodę (Akcję) na kontrolerze podając odpowiednie parametry.
- Weryfikujemy otrzymany wynik.

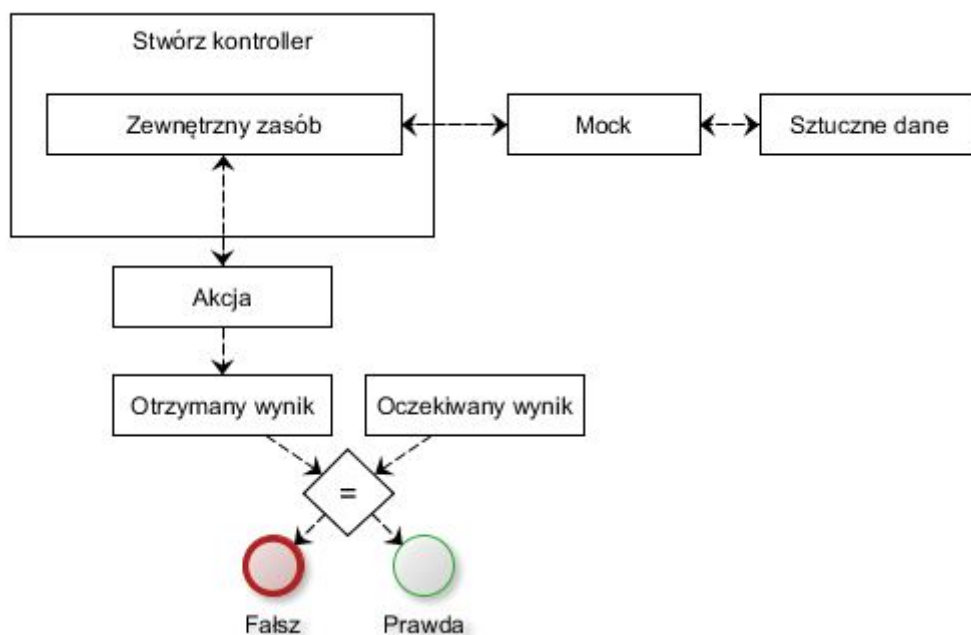


Rys. Test jednostkowy kontrolera bez zewnętrznych zasobów

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.



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

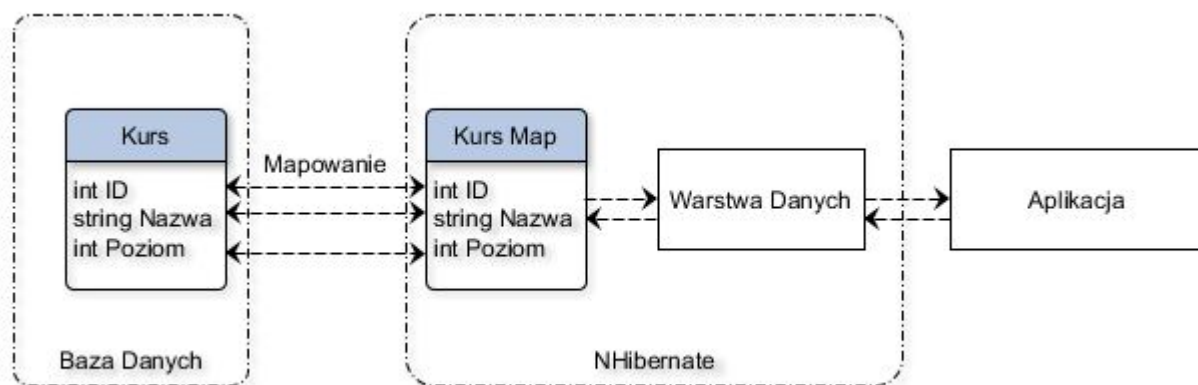
W powyższym kodzie testowany jest kontroler Kursów i jego metoda **Create**. Test testuje okoliczność gdy kontroler kursów pobierając dane z bazy danych otrzyma wartość **null**. W tym przypadku ma zwrócić widok błędu. Linijka tworzenia **Mocka** po pierwsze definiuje, że oczekuje wywołania danej funkcji następnie określa jakich oczekujemy parametrów i na końcu określa jakie dane mają być zwrócone.

Po zdefiniowaniu **MOCK-a** uruchamiane jest odtwarzanie jego zachowania. W momencie przeprowadzania testu podczas wywoływania metody **Create** po natrafieniu na podmienioną metodę (w przypadku przedstawionym w zamieszczonym kodzie źródłowym **AddCourse**), test nie skorzysta z bazy danych ale z udawanego obiektu otrzymując wartość **null**.

Logika kontrolerów pokryta jest łącznie ponad 50 testami jednostkowymi.

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 plikach konfiguracyjnych **XML**. Jest to dość problematyczna metoda podatna na błędy oraz tworzony kod nie jest do końca czytelny. Innym sposobem generowania mapowań jest zastosowanie frameworka **FluentNHibernate** pozwalającego definiować mapowania wykorzystując mocno typowany i kompilowany kod jednego z języków **CLR**.

```

public class CourseModelMap : ClassMap<CourseModel>
{
    public CourseModelMap()
    {
        Id(x => x.ID);
        Map(x => x.Logo);
        Map(x => x.Name).NotNullable();
        Map(x => x.CreationDate).NotNullable();
        Map(x => x.Description);
        Map(x => x.ShortDescription);
        Map(x => x.News);
        Map(x=>x.Password).Nullable();

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

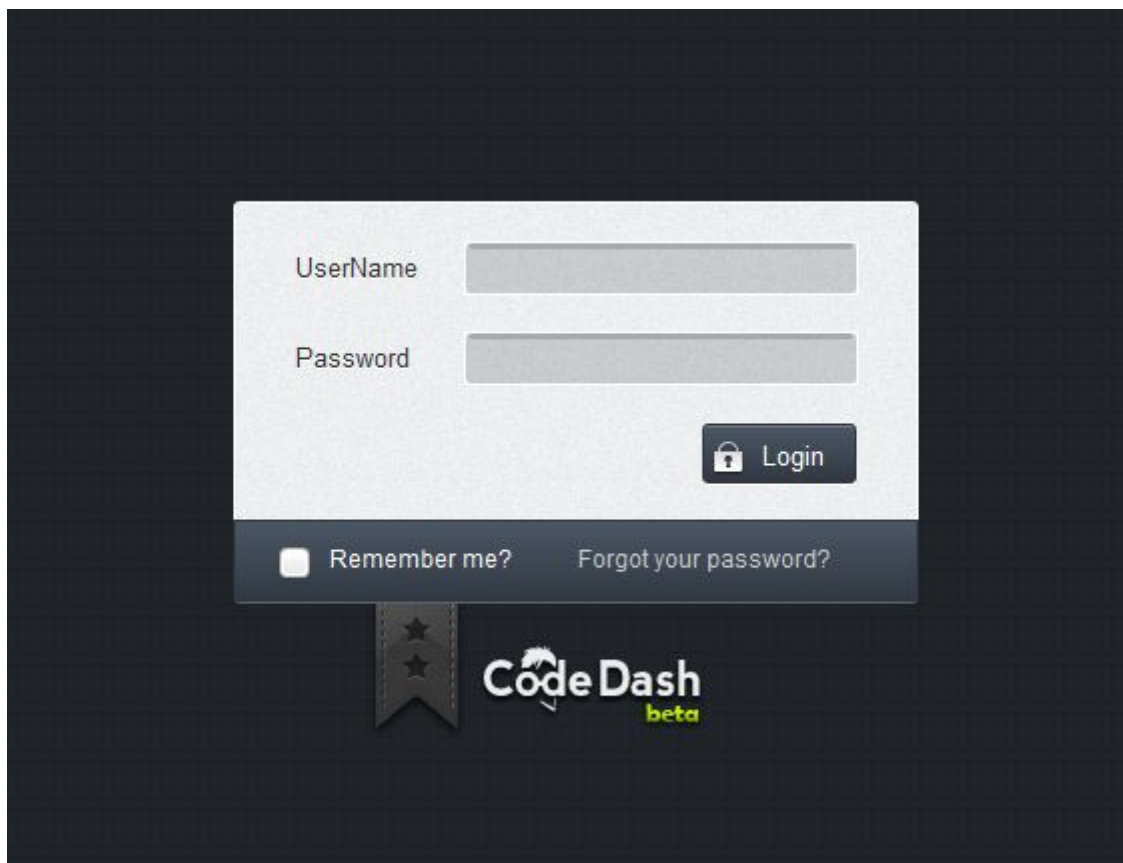
```

Rys. Przykładowe mapowanie klasy Kurs

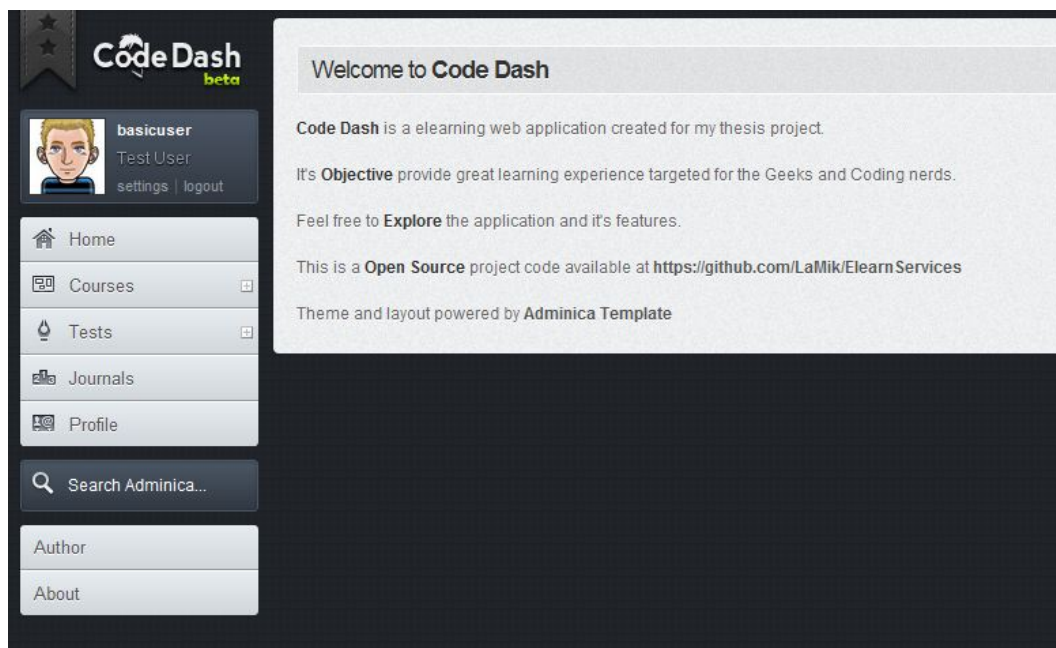
Mapowanie realizowane jest poprzez dziedziczenie generycznej klasy **ClassMap** dostępnej w bibliotece **FluentNHibernate**. Wszystkie parametry mapowanej klasy muszą być publiczne oraz oznaczone słowem kluczowym **virtual**. Na początku należy zdefiniować parametr które będzie zmapowany na pole będące kluczem bazy danych. Jest to wymagane parametr bez , którego nie można przeprowadzić procesu mapowania. Następnie poprzez użycie Funkcji **Map()** definiuje się mapowania parametrów do określonych wierszy tabel. Kolejną istotną rzeczą do zdefiniowania są relacje. Metoda **Reference()** mapuje relacje jeden do jednego natomiast **HasMany()** pozwala zdefiniować relację jeden 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()** podając jako parametr ciąg znaków określający nazwę tabeli.

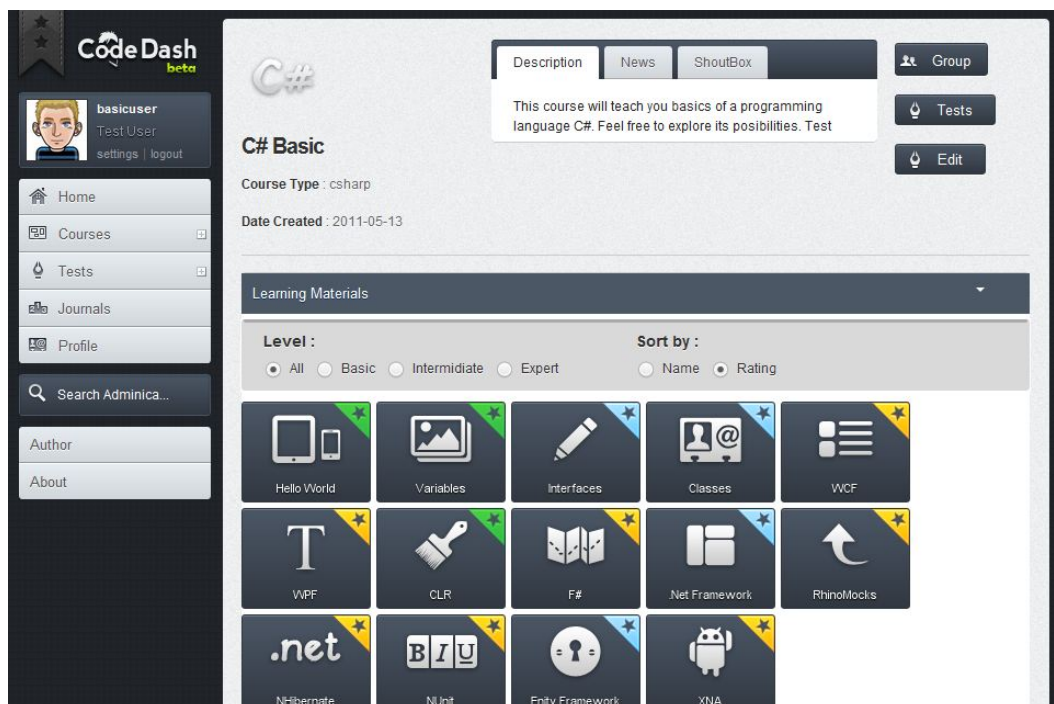
5.5 Interfejs użytkownika



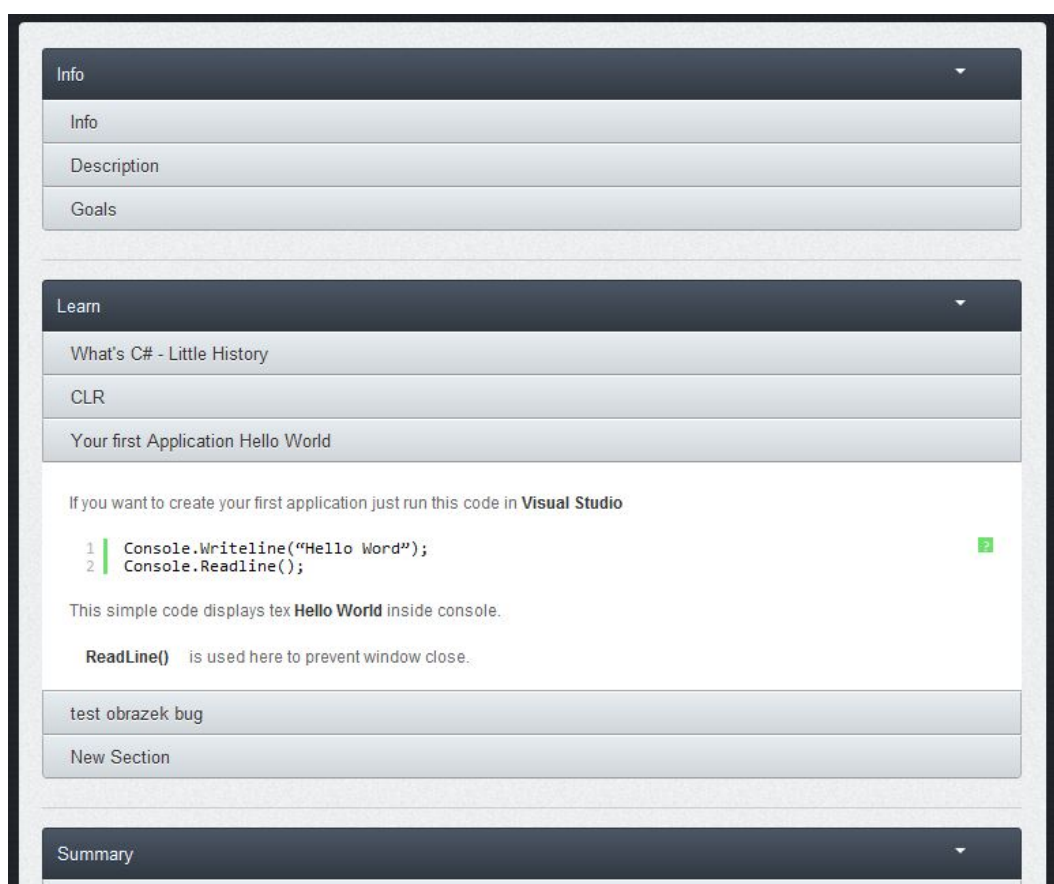
Rys. Widok logowania



Rys. Widok strony głównej



Rys. Widok kursu



Rys. Widok materiału nauczania

The screenshot shows a web application for a test. At the top, there's a dark header with a 'Questions' dropdown menu. Below it, a list of four questions is displayed: 'What's CLR?', 'Is C# Object Oriented?', 'What will this function do?', and 'C# is an evolution of?'. The third question, 'What will this function do?', is selected. Below the list, there's a detailed view of the selected question. It has a 'Question' section with the text 'C# is an evolution of? *More than one answer*' and an 'Answers' section with four radio button options: 'C++', 'Java', 'Pythtrue', and 'C'. At the bottom of the interface, there's a 'Finish' button with a circular arrow icon.

Rys. Widok testu

The screenshot shows the 'Edit Question' interface. At the top, there's a dark header with a 'Questions' dropdown menu. Below it, a list of two questions is displayed: 'What's CLR?' and 'Is C# Object Oriented?'. The second question, 'Is C# Object Oriented?', is selected. Below the list, there's a detailed view of the selected question. It has a 'QuestionLabel' section with the text 'What will this function do'. Below that, there's a rich text editor with the text 'What will this function do?', 'Console.ReadLine();', and '*!More than one correct Answer!*'. To the right of the editor, there's a list of answers with checkboxes: 'Closes Window', 'Wait's for input', 'Prevents window close', and 'Pdf Reader Functittrue'. At the bottom left, there's a 'Create' button with a checkmark icon.

Rys. Widok edycji testu

Rozdział 6

Testowanie i ocena efektywności

6.1 Wybrane Testy Mechanizmów Zabezpieczeń

6.1.1 Sprawdzenie mechanizmów szyfrowania wiadomości

Do sprawdzenia zawartości przesyłanych wiadomości użyję testowej aplikacji (**WcfTestClient**) pozwalającej wysyłać zapytania do usługi sieciowej. Aplikacja ta jest udostępniana wraz z pakietem **Visual Studio**.

6.1.2 Sprawdzenie mechanizmu logowania oraz poziomów uprawnień

Sprawdzenie działania logowania i blokad dostępu do aplikacji

6.1.3 Zabezpieczenie przed atakami typu wstrzykiwanie skryptów

W tym teście przetestujemy odporność aplikacji na próby wstrzyknięcia skryptów do pól tekstowych. Po wpisaniu skryptu aplikacja powinna zwrócić błąd informujący o próbie przeprowadzenia takiego ataku.

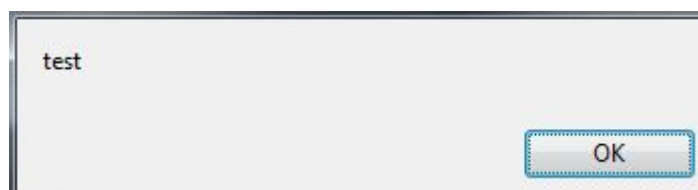


Rys. Atak poprzez wstrzykiwanie skryptów.

W tym przypadku serwer zwrócił odpowiedni błąd informujący o próbie wstrzyknięcia skryptu. Zabezpieczenie to jest domyślnym ustawieniem w frameworku **Asp.Net Mvc**. Dane zwracane z widoku przed przetworzeniem i wysłaniem do odpowiedniego kontrolera odpowiedzialnego za uaktualnienie danych w bazie są sprawdzane w celu wykrycia potencjalnie niebezpiecznych skryptów. Istnieje możliwość jawnego wyłączenia tego zabezpieczenia. Opcja ta została użyta w przypadku edycji materiałów nauczania oraz testów. Pola tekstowe w tym przypadku możliwość umieszczania fragmentów kodu, które zostałyby zinterpretowane jako potencjalne zagrożenie.



Rys. Atak poprzez wstrzykiwanie skryptów wyłączony mechanizm.



Rys. Udany atak typu script inject.

W trakcie testów odkryłem , że istnieje możliwość strzyknięcia skryptu , który odpalał się po przeładowaniu strony. W zaprezentowanym przykładzie wyświetlany jest jedynie prosty alert z testową informacją , jednakże taka luka pozwala wrzucać najrozmaitsze skrypty pozwalające wyrządzić spore szkody. Dobrym przykładem może być chociażby wstrzyknięcie keyloggera przechwytyującego wszystkie znaki wprowadzane przez użytkownika.

Do oczyszczenia wprowadzanych danych stworzyłem prosty mechanizm oparty na wyrażeniach regularnych. Standardowe rozwiązanie kodujące odpowiednie znaki było dość problematyczne ponieważ część wyświetlanych wartości była dodatkowo przetwarzana po stronie klienta. Po wprowadzeniu poprawek serwer pozwala zapisać takie dane jednakże zabezpieczenie odpowiednio oczyszcza zapisywany tekst.

6.2 Testy wydajności mechanizmów przetwarzania danych

6.2.1 Wybrane testy funkcjonalne

Tworzenie materiałów nauczania

Scenariusz testowy obejmuje:

- stworzenie nowego materiału nauczania
- stworzenie parametrów opisowych
- stworzenie sekcji nauczania

Akcja	Oczekiwany rezultat	Wynik
Przejsie do panelu edycji	Wygenerowanie odpowiedniego widoku	OK
Dodanie materiału nauczania	Wygenerowanie nowego materiału w bazie danych Przejsie do widoku edycji	OK OK
Edycja pól opisowych	Uaktualnienie danych w bazie Wyświetlenie alerta informującego o powodzeniu akcji	OK OK
Dodawanie Sekcji nauczania	Dodanie sekcji w bazie Wyświetlenie alerta informującego o powodzeniu akcji	OK OK
Edytowanie Sekcji nauczania	Uaktualnienie sekcji w bazie e Wyświetlenie alerta informującego o powodzeniu akcji	OK OK

Tabela. 6.1 Testy funkcjonalne tworzenie materiałów nauczania

Tworzenie testów

Scenariusz testowy obejmuje:

- stworzenie nowego testu
- stworzenie nowego pytania z odpowiedziami

Akcja	Oczekiwany rezultat	Wynik
Przejsie do panelu tworzenia testu	Wygenerowanie odpowiedniego widoku	OK
Dodanie Testu	Wygenerowanie nowego testu Dodanie testu do bazy danych Przejsie do widoku edycji	OK OK OK
Dodanie nowego pytania	Wyświetlenie okna modalnego z formularzem edycji pytania Dodanie opisu oraz dodanie odpowiedzi	OK OK
Zatwierdzenie pytania	Dodanie pytania do bazy danych	OK

Tabela. 6.2 Testy funkcjonalne tworzenie testu

Rozwiązywanie testów

Scenariusz testowy obejmuje:

- uruchomienie testu
- rozwiązanie testu
- sprawdzenie wyniku

Akcja	Oczekiwany rezultat	Wynik
Przejsie do panelu rozwiązywania testu	Pobranie danych z bazy danych	OK
	Wygenerowanie widoku	OK
Zatwierdzenie testu	Wyliczenie wyniku	OK
	Dodanie rekordu do bazy danych	OK
	Wygenerowanie widoku z wynikiem	OK

Tabela. 6.3 Testy funkcjonalne rozwiązywania testu

Tworzenie kursu

Scenariusz testowy obejmuje:

- Stworzenie kursu

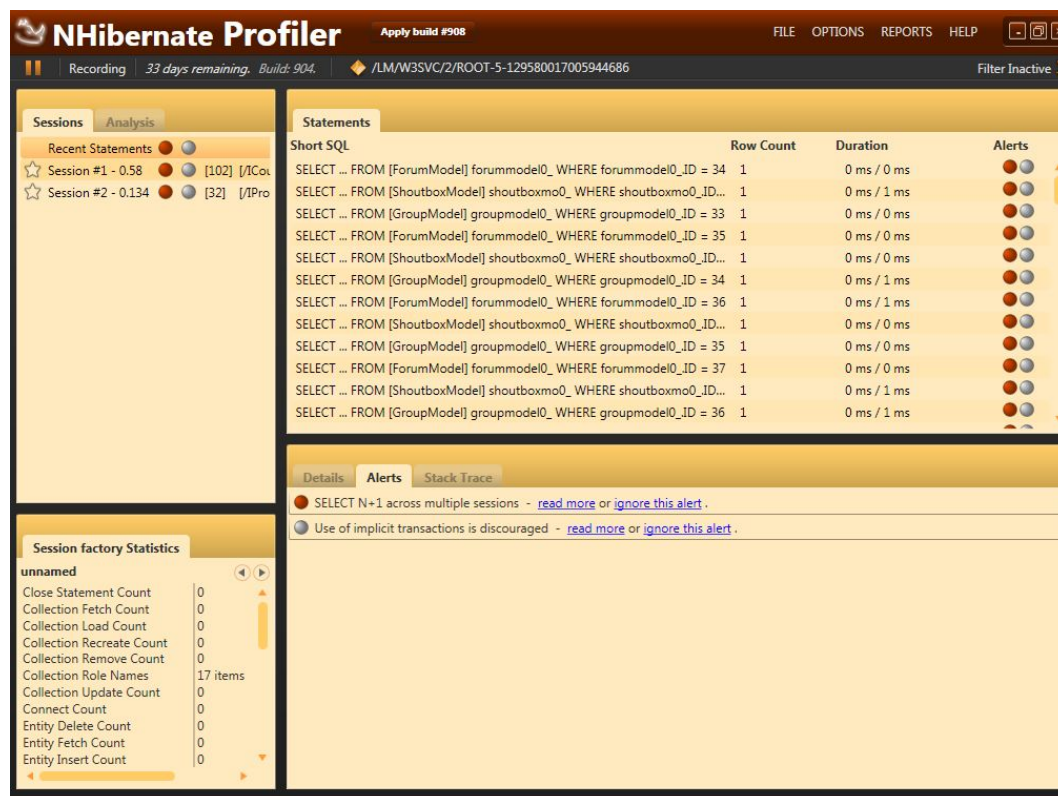
Akcja	Oczekiwany rezultat	Wynik
Przejsie widoku tworzenia kurs	Wygenerowanie widoku z formularzem	OK
Zatwierdzenie danych	Stworzenie kursu w bazie danych	OK
	Stworzenie grupy w bazie danych	OK
	Stworzenie shoutboxa w bazie danych	OK
	Dodanie aktualnego użytkownika jako autora kursu	OK

Tabela. 6.4 Testy funkcjonalne Tworzenie kursu

6.2.2 Analiza oraz optymalizacja zapytań generowanych przez NHibernate za pomocą narzędzia NHProf

NHibernate jest bardzo dobrym i popularnym mapperem obiektowo relacyjnym. Prócz samej biblioteki dostępne są różne narzędzia pozwalające analizować działanie frameworka. Jednym z takich narzędzi jest **NHProf**. Produkt **Ayende Rahien-a** znanego bloggera oraz jednej z najbardziej aktywnych osób w zespole odpowiedzialnym za przeniesienie Hibernate-a z Javy na platformę .Net.

Aplikacja ta jest niestety płatna ale istnieje możliwość przetestowania , w żaden sposób nieograniczonej wersji aplikacji , w terminie do 30 dni.



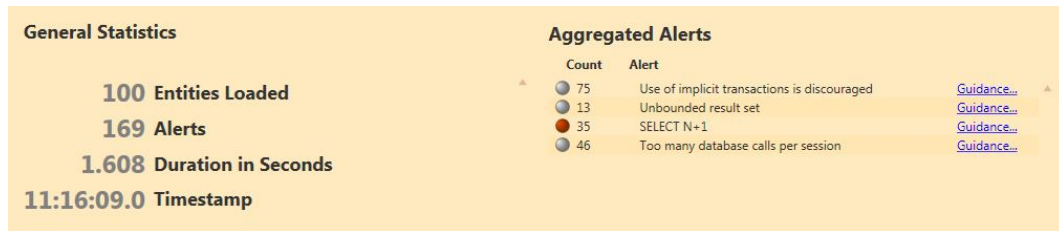
Rys. Główny widok aplikacji NHProf.

NHprof jest aplikacją pozwalającą analizować zapytania generowane poprzez framework **NHibernate**. Prócz mechanizmu przechwytywania zapytań posiada świetne narzędzia analizujące działanie **ORM-a**. Potrafi wykryć najczęstsze problemy wynikające z błędnej konfiguracji , błędnych mapowań bądź błędnych mechanizmów przetwarzania.

Korzystanie z tego narzędzia jest stosunkowo proste. Wystarczy podłączyć odpowiednią bibliotekę i za inicjalizować określoną statyczną klasę. **NHprof** korzysta z biblioteki **log4net** (alternatywna biblioteka upraszczająca mechanizmy logowania) i również należy ją dołączyć do projektu. W przypadku omawianych testów. Inicjalizacja mechanizmu profilowania jest uruchamiana w momencie startu usługi sieciowej.

NHProf pozwala wyznaczyć trzy istotne parametry , na których oprzemy całą analizę i proces optymalizacji.

- Czas zapytania - określa całkowity czas jaki jest potrzebny na wykonanie danej operacji.
- Ilość generowanych zapytań - określa liczbę zapytań potrzebnych do wykonania danej operacji.
- Ilość pobieranych encji - określa ilość zwróconych obiektów



Rys. Przykładowy widok prezentujący wynik profilowania.

Pobieranie danych

Przeanalizujmy proces pobierania pojedynczego obiektu typu Kurs.

Czas zapytania	1.608s
Ilość zapytań	75
Ilość encji	100

Tabela. 6.5 Pierwsza próba

Pierwsza próba nie wygląda najlepiej. Po pierwsze pobierane jest łącznie 100 encji. Jest to za duża liczba oznaczająca, że podczas pobierania danych odwołujemy się do wielu niepotrzebnych danych. Najprawdopodobniej jest to spowodowane nieoptymalną konfiguracją mapowań oraz mechanizmów późnej inicjalizacji. Kolejnym parametrem rzucającym się w oczy jest ilość zapytań. Idealnie powinno być generowane jedno zapytanie. W tym przypadku jednak mamy ich naprawdę sporo. Ilość generowanych zapytań jest szczególnie ważna w przypadku skalowalności aplikacji. Zbyt duża ilość zapytań może bardzo szybko wyczerpać pulę dostępnych połączeń. W tym przypadku tak duża liczba zapytań najprawdopodobniej spowodowana jest niepotmyalną implementacją mechanizmu zarządzania sesją. Czas potrzebny an wygenerowanie zapytań jest wprost proporcjonalny do ilości encji i zapytań wobec tego również jest zbyt duży.

Czas zapytania	0.308s
Ilość zapytań	8
Ilość encji	25

Tabela. 6.6 Po wprowadzeniu poprawek

Proces wprowadzania poprawek warto zacząć od analizy danych wymaganych do generacji tego konkretnego widoku. Widok Kursu prezentuje dane na temat kursu oraz listę dostępnych materiałów nauczania. Dodatkowo wymaga pobrania encji grupy przynależącej do danego kursu by określić czy dany użytkownik należy do grupy. Po przeanalizowaniu generowanych zapytań okazało się, że pobieram dużo niepotrzebnych danych. Poprawki objęły m.in. . Modyfikacje mapowań oraz odpowiednie skonfigurowanie procesu późnej inicjalizacji.

Zapisywanie danych

Analizę zapisu do bazy danych przeprowadzimy na obiekcie materiał nauczania. Encja ta składa się z wielu parametrów opisowych. Scenariusz testowy przewiduje modyfikację jednej sekcji nauczania i zapis zmian do bazy danych. W tym przypadku nie otrzymamy parametru ilości pobranych encji ponieważ jest to zapytanie, które nic nie zwraca z bazy danych. Podobnie z parametrem ilości zapytań w tym przypadku będziemy analizować ilość generowanych zapisów do bazy danych.

Czas operacji	3.668s
Ilość zapisów	60

Tabela. 6.7 Pierwsza próba

Próba wykazała, że zarówno czas operacji jak i ilość zapytań są zbyt duże i należy je poprawić tym bardziej, że scenariusz testowy zakładał modyfikację tak naprawdę jednego wiersza tabeli. Jak widać coś tu jest nie tak skoro wykonujemy aż 60 operacji zapisu. Po analizie logów generowanych przez aplikację **NHProf** okazało się, że podczas uaktualniania danych materiału nauczania automatycznie wykonywane były operacje uaktualniania elementów z nim powiązanych. Główny problem był z testem należącym do materiału. Test posiada wiele pytań i odpowiedzi. W pierwszej próbie dochodziło do aktualizacji każdego pytania i odpowiedzi mimo tego, że te dane nie były zmieniane.

Czas operacji	0.571s
Ilość zapisów	13

Tabela. 6.8 Po wprowadzeniu poprawek

Po poprawie mapowań oraz mechanizmu uaktualniania danych można zauważyć wyraźną poprawę zarówno w czasie potrzebnym na wykonanie operacji jak i ilości generowanych zapytań. Największą zmianą było wyłączenie kaskadowego uaktualniania danych w przypadku testu powiązanego z uaktualnianym materiałem nauczania.

6.2.3 Testy obciążeniowe

Zgodnie z projektem. Aplikacja powinna obsługiwać co najmniej 100 użytkowników jednocześnie w czasie nie większym niż 1 sekunda. Testy obciążeniowe zostaną wykonane wraz z użyciem darmowej aplikacji **WCAT** dostępnej w pakiecie narzędzi do serwera IIS. Aplikacja ta w dość łatwy sposób pozwala zasymulować operacje wielu użytkowników wykonując wiele zapytań na raz do serwera. Konfiguracja przypadków testowych odbywa się poprzez zdefiniowanie testów w trzech plikach tekstowych.

script.txt

Pozwala definiować konkretne zapytania tzn. gdzie i jak wykonać zapytanie

distribution.txt

Pozwala definiować niesymetryczne rozłożenie testów na różne strony np. jedna strona będzie ładowana z częstotścią 70

config.txt

Pozwala ustalić opcje testów takie jak : Czas testu , czas oczekiwania na test , ilość symulowanych klientów , ilość symulowanych wątków itd.

Ustawienia

Warmuptime	5s
Duration	60s
CooldownTime	5s
NumClientMachines	1
NumClientThreads	100

Tabela. 6.9 Konfiguracja symulowanych użytkowników

Aplikacja została skonfigurowana tak by symulować działanie 100 użytkowników na raz. Czas działania aplikacji ustawiony na 60s. Dodatkowa analiza działania aplikacji zostanie przeprowadzona przez profiler **ANTS Performance Profiler** firmy RedGate. Narzędzie to pozwala monitorować wykorzystywane zasoby oraz szczegółowo przeanalizować jakie metody zostały wywołane.

Scenariusz testowy

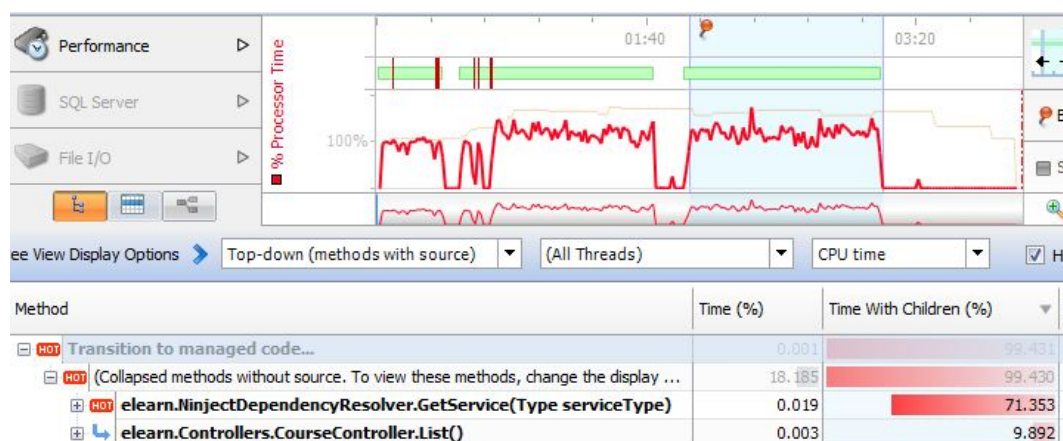
Scenariusz testowy zakłada wysłanie zapytania w formie komendy **GET** do serwera **IIS**. Zapytanie będzie odwoływało się do widoku wyświetlającego wszystkie kursy. By móc wykonać ten test zostały wyłączone wszelkie zabezpieczenia aplikacji by dać dostęp symulowanym użytkownikom testowym.

Przeprowadzenie testu

Minimalny	3.120s
Średni	3.773s
Maksymalny	5.600s

Tabela. 6.10 Średnie Czasy odpowiedzi

Wyniki nie spełniają założeń. Czasy odpowiedzi są zbyt duże. Należy podjąć próbę poprawy tego stanu.

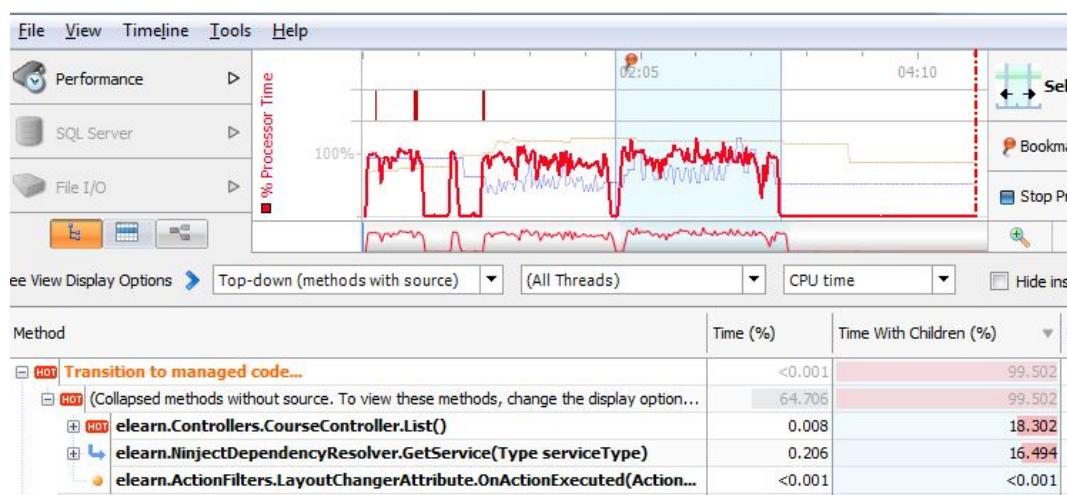


Rys. Wynik programu profilującego

Po analizie logów programu profilującego możemy zauważyć, że większość czasu użytkownik poświęca oczekując na dane z bazy danych.

Bardzo dużo czasu marnowane jest na wykorzystywanie mechanizmów biblioteki **Ninject** do wstrzykiwania zależności. Biblioteka ta pozwala odseparować inicjalizację odpowiednich klas, w tym przypadku klas obsługujących usługi, do innej warstwy. Dzięki temu nie trzeba w wielu miejscach kodu inicjalizować obiektów.

Ninject pozwala definiować zasięg tworzenia obiektów. W powyższym przypadku gdzie widać jak dużo czasu poświęcamy na tworzenie obiektów była ustawiona opcja tworzenia nowej instancji klasy dla każdego żądania tzn. każdy zapytanie do serwera generowane przez symulowanego klienta powodowało tworzenie nowej instancji klasy usług kursów. **Ninject** opiera inicjalizację obiektów na kosztownych mechanizmach. Po zastosowaniu opcji tworzenia jednej globalnej instancji klasy poziom wykorzystania biblioteki znacznie zmalał.



Rys. Wynik programu profilującego

Minimalny	3.120s
Średni	3.773s
Maksymalny	5.600s

Tabela. 6.11 Średnie Czasy odpowiedzi

Analiza czasów odpowiedzi pokazała, że nie tu tkwi problem i należy zoptymalizować działanie aplikacji po stronie serwera z dostępem do bazy danych wystawionym za usługami sieciowymi.

Analiza narzędziem **NHProf** wykazała, że nieoptymalizowane operacje na bazie danych wymagają 200 milisekund czasu do pobrania danych. NHProf wykazał, że wykonujemy za dużo zapytań do bazy pobierając niepotrzebne dane. Dodatkowo uruchomimy cache drugiego poziomu.

Po wprowadzeniu poprawek w mapowań widać poprawę w czasie dostępu do bazy danych/

	Przed	Poprawki mapowań	Cache
Ilość encji	36	20	0
Ilość zapytań	21	5	0
Czas zapytania	200ms	18ms	13ms

Tabela. 6.12 Operacje na bazie danych

Minimalny	2.300s
Średni	3.007s
Maksymalny	4.600s

Tabela. 6.13 Średnie Czasy odpowiedzi

Nadal jednak nie jest to zadowalający wynik. Po przeprowadzeniu analizy tego co się dzieje po stronie serwera okazało się , że ponad 50 procent czasu wywoływania usług jest poświęcane na generowanie raportów dla profilera NHibernate po wyłączeniu profilera.

Widać teraz , że Profile NHibernate-a dodawał praktycznie 2 sekundy czasu odpowiedzi. Bardzo ważne jest by go wyłączyć na serwerze produkcyjnym.

Profil aplikacji wykazał , że po stronie klienta 90 procent czasu poświęcamy na generowanie zapytania i oczekiwanie odpowiedzi od usługi sieciowej. W związku z tym pozostała nam jeszcze jedna opcja , która możliwe że poprawi czasy odpowiedzi. Mianowicie cachowanie zapytań do usług sieciowych. Zapytanie ponieważ jest zwykłą wiadomością protokołu HTTP może być cachowane normalnymi mechanizmami jakimi cachujemy np generowanie stron.

Ustawimy cachowanie usługi pobierającej listę kursów tak by wygasło po 60 sekundach.

Włączenie cachowania zapytań WCF-owych przyniosło całkiem spory zysk czasowy.

Procedura profilowania oraz poprawiania generacji tego konkretnego widoku była bardzo owocna. Z początkowego czasu generowania równego 3 s zrobiło się nie całe 0.5 s.

Minimalny	1.014s
Średni	1.198s
Maksymalny	1.701s

Tabela. 6.14 Średnie Czasy odpowiedzi

Minimalny	405ms
Średni	455ms
Maksymalny	656ms

Tabela. 6.15 Średnie Czasy odpowiedzi

6.2.4 Badanie czasu odpowiedzi usług sieciowych

6.2.5 Testy konfiguracji rozłożenia usług sieciowych

6.3 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. Evjen, 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.