POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Projekt inżynierski

Edytor i aplikacja udostępniająca plany studiów

Autor: Łukasz Kulig, Krzysztof Sałajczyk

Kierujący pracą: dr inż. Robert Tutajewicz

Gliwice, styczeń 2011.

1. WSTĘP

Tematem pracy jest aplikacja, która ma za zadanie wspomagać proces tworzenia planów studiów, oraz umożliwiać przeglądanie już utworzonych planów.

Aplikacja powinna być podzielona na dwie części – część desktopową i cześć webową. Część desktopowa powinna umożliwiać zarządzanie planem – tworzenie i edycja, oraz zarządzaniem rzeczy potrzebnych do tworzenia planu – zarządzanie przedmiotami, semestrami, wydziałami, kierunkami, typami przedmiotów, typami studiów, instytutami oraz specjalizacjami. Za zarządzanie każdym elementem odpowiedzialny jest osobny moduł. Każdy moduł zostanie opisany w kolejnych punktach.

Podczas tworzenia planu możliwe jest dodawanie komentarzy, które zawierają informacje o błędach, sugestiach, które pomagają osobie tworzącej plan. Możliwe jest także tworzenie planu na podstawie planu już zarchiwizowanego, czyli takiego, który już nie obowiązuje. Zarchiwizowany plan można tylko przeglądać, nie można go przywrócić. Zapewniona jest możliwość tworzenia reguł, które następnie są weryfikowane. Wyróżnionych jest 6 typów reguł (zostaną opisane w kolejnych punktach). Weryfikacja planu polega na sprawdzeniu czy dany plan spełnia utworzone dla niego reguły, jeżeli reguły zostały spełnione użytkownik dostaje stosownie poinformowany.

Prócz możliwości tworzenia planu, aplikacja zapewnia przeglądanie planu, a także wydruk gotowego planu. Poprzez wydruk rozumiany jest tutaj eksport planu do plików o formacie *.pdf oraz *.xml.

Zapewniony jest tzw. moduł bezpieczeństwa, jego zadaniem jest ograniczenie funkcji programu dla użytkownika o określonej roli. Rola jest to najprościej mówiąc zbiór uprawnień, według których włączane są odpowiednie funkcje programu. Każdy użytkownik jest przypisany do konkretnej roli, możliwe jest tworzenie własnej roli, dla której można sprecyzować funkcje programu, które mają być dostępne. Dostęp do programu przed niepowołanymi osobami jest zapewniony poprzez mechanizm logowania. Mechanizm ten umożliwia pracę tylko osobom, które posiadają konto.

Jest dostępny także moduł umożliwiający zarządzanie użytkownikami. Zapewnia on podstawowe funkcje takie jak wyświetlanie listy użytkowników, dodawanie, usuwanie i

edycję użytkowników. Poprzez edycję rozumiane są tutaj akcje zmiany hasła, loginu, adresu e-mail oraz roli. Domyślnie dostęp do tego modułu ma tylko administrator systemu.

Część webowa spełnia tylko funkcję prezentacji planu. Nie są dla niej zastosowane mechanizmy ochronne, tak jak w przypadku części desktopowej. Każdy ma dostęp do listy planów i może wybrać interesujący go plan w celu jego obejrzenia.

2. ANALIZA TEMATU

Naszym celem było stworzenie aplikacji, która wspomoże proces tworzenia planów studiów, oraz umożliwi przeglądanie już stworzonych planów. Próby znalezienia jakichkolwiek rozwiązań udostępniających podobną funkcjonalność spełzły na niczym. Prawdopodobnie wynika to z tego, iż jednostki uczelniane posiadają swoje własne oprogramowanie służące do tworzenia planów studiów i jest ono wykorzystywane w zamkniętym gronie. Na uczelniach technicznych może być ono stworzone przez pracowników, natomiast pozostałe uczelnie mogły zlecić stworzenie takiego oprogramowania firmom zewnętrznym. Całkiem możliwym jest również fakt, iż do tworzenia planów studiów nie jest wykorzystywane żadne oprogramowanie poza standardowym edytorem tekstu.

W naszej opinii aplikacja wspomagająca tworzenie planu studiów powinna działać niezależnie od typu uczelni (uczelnia techniczna, humanistyczna). Wiadomo, iż różnice pomiędzy typami uczelni są dość spore. Na przykład na uczelniach technicznych jednym z typów zajęć może być laboratorium, natomiast na uczelniach humanistycznych taki typ zajęć raczej nie występuje.

Analizując zadany temat doszliśmy do wniosku, iż nasza aplikacja powinna w jak największym stopniu pozwalać definiować plan. Nie chodzi tu tylko o listę przedmiotów na danym semestrze, ale również aplikacja powinna pozwalać definiować nowe wydziały, kierunki, typy przedmiotów, instytuty działające na danym wydziale, specjalizacje na danym kierunku, oraz semestry. Takie możliwości edycji są spowodowane tym, iż co roku są wprowadzane zmiany w jednostkach uczelnianych, tworzone są nowe kierunki, a czasem całe wydziały. Naszym zamierzeniem było stworzenie aplikacji, która pomimo zmian organizacyjnych nadal będzie mogła być wykorzystywana.

Analiza poszczególnych modułów zarządzających elementami składowymi planu:

- Moduł „Wydziały” – umożliwia tworzenie, edycję oraz usuwanie wydziałów; jeżeli wydział posiada jakieś kierunki, instytuty, lub plany studiów to nie jest możliwe jego usunięcie.

- Moduł „Kierunki” – umożliwia tworzenie, edycję, usuwanie kierunków i przypisywanie kierunków do wydziałów; jeżeli dany kierunek posiada jakieś plany to nie można go usunąć. Należy tutaj nadmienić, iż dany kierunek może być prowadzony na kilku wydziałach, np. kierunek Informatyka na Politechnice Śląskiej jest prowadzony między innymi na wydziale Automatyki Elektroniki i Informatyki oraz na wydziale Elektrycznym.

- Moduł „Typy przedmiotów” – tworzenie, edycja usuwanie typów przedmiotów; jeżeli w jakimś planie istnieje przedmiot danego typu to nie można tego typu usunąć. Moduł ten ma na celu pozwolenie użytkownikowi na definiowanie dowolnych typów przedmiotów, np. na uczelniach humanistycznych typ „Laboratorium” nie wydaje się być potrzebnym, natomiast „Zajęcia językowe” już mogą wystąpić.

- Moduł „Typy studiów” – pozwala na tworzenie, edycję i usuwanie typów studiów. Moduł ten wprowadza możliwość tworzenia osobnych planów dla konkretnego kierunku na konkretnym wydziale, ale dla osobnych typów studiów, np. wieczorowych i dziennych.

- Moduł „Instytuty” – umożliwia edycję, tworzenie i usuwanie instytutów oraz przypisywanie instytutów do wydziałów. Podobnie jak w przypadku kierunków tak i tutaj instytut o danej nazwie może działać na więcej niż jednym wydziale.

- Moduł „Specjalizacje” – pozwala na tworzenie i edycję specjalizacji prowadzonych w ramach określonego kierunku na konkretnym wydziale, a także na usuwanie istniejących specjalizacji.

- Moduł „Semestry” – umożliwia tworzenie, edycję i usuwanie semestrów. Moduł ten ma na celu zapewnienie poprawności planu, gdy dany rok akademicki będzie składał się z niestandardowej liczby semestrów, np. poza

semestrami zimowym i letnim w danym roku akademickim będzie występował również jakiś semestr pośredni (np. wiosenny).

Jednym z dodatkowych zadań naszej aplikacji ma być również weryfikacja planu pod względem jakichś ustalonych zasad. Tzn. dla konkretnego planu mogą być tworzone pewnego rodzaju reguły, a później użytkownik może sprawdzić, czy plan spełnia wszystkie reguły. Funkcjonalność ta pozwala na sprawdzenie np. czy plan jest zgodny z regulaminem uczelni. Przykładem może być tutaj sprawdzenie, czy wszystkie zajęcia o nazwie „Wychowanie Fizyczne" są za 0 punktów ECTS (wymóg taki znajduje się w regulaminie studiów Politechniki Śląskiej).

Wszystkie dane wprowadzane przez użytkownika przechowywane będą w relacyjnej bazie danych, co pozwoli na utrzymywanie logicznej struktury warstwy danych aplikacji.

2.1 Założenia

Na podstawie powyższej analizy określiliśmy główne założenia tworzonej aplikacji:

- Definiowanie pojęć słownikowych:
  - wydziały,
  - kierunki,
  - instytuty,
  - specjalizacje,
  - typy przedmiotów,
  - typy studiów,
  - semestry.
- Tworzenie nowego planu dla danego kierunku na danym wydziale z uwzględnieniem typu studiów.

- Edycja planu:
    - dodawanie / edycja przedmiotów,
    - recenzja planu rozumiana – informacje dla edytora zapisane przez recenzenta,
    - publikowanie planu – zmiana stanu planu na „obowiązujący”,
    - archiwizacja planu – zmiana stanu planu na „archiwalny”.
- Weryfikacja planu na podstawie ustalonych reguł
- Wydruk planu:
    - podgląd pliku pdf,
    - wydruk do formatów pdf i xml.
- Zarządzanie użytkownikami:
    - tworzenie / edycja / usuwanie użytkowników,
    - tworzenie / edycja / usuwanie ról.

2.2 Użyte technologie

Do realizacji projektu zdecydowaliśmy się użyć języka C# i platformy Microsoft .NET 4. Do połączenia aplikacji z bazą danych i wszystkich operacji na danych wykorzystany został LINQ to Entities. Powodem takiego wyboru jest fakt, iż język C# jest aktualnie jednym z najbardziej rozpowszechnionych języków programowania, a także dzięki wbudowaniu wielu mechanizmów pozwala na szybkie tworzenie aplikacji. Poniżej znajdują się krótkie opisy wykorzystanych technologii.

2.2.1 Microsoft .NET 4 i język C#

Język C# jest aktualnie jednym z najbardziej popularnych języków programowania. Wielu pracodawców wymaga od przyszłych pracowników znajomości tego języka. Sam język – jako twór firmy Microsoft – jest bardzo dobrze przystosowany do pracy w środowisku Windows (które to jest najczęściej używanym systemem operacyjnym do

pracy biurowej). Sam język posiada wiele wbudowanych funkcji służących do interakcji z systemem i sprzętem. Dodatkowo dostępna jest bardzo bogata dokumentacja do języka.

Środowiskiem uruchomieniowym dla programów napisanych w języku C# jest platforma Microsoft .NET. W naszym wypadku skorzystaliśmy z wersji Microsoft .NET 4 jako standardowo dostępnej w środowisku Microsoft Visual Studio 2010 Professional. Sama platforma wspiera tworzenie aplikacji wykorzystujących różne technologie. Można tworzyć standardowe aplikacje okienkowe za pomocą Windows Forms lub WPF oraz serwisy internetowe wykorzystujące technologię ASP.NET lub Silverlight. W naszym przypadku do budowy aplikacji wykorzystujemy technologię Windows Forms, przy czym nie wykorzystujemy standardowych kontrolek dostępnych w Visual Studio, ale kontrolki firmy Telerik.

2.2.2 LINQ to Entities

LINQ to Entities to implementacja Linq pozwalająca na pisanie zapytań do modelu danych stworzonego w Entity Framework. Sama technologia Linq udostępnia wiele metod, które znacznie przyspieszają programowanie (np. konwersje różnych typów kolekcji na inne kolekcje). Język zapytań w LINQ to Entities jest bardzo podobny do języka zapytań SQL.

W naszym przypadku (mając już stworzoną bazę danych) wykorzystując Entity Framework stworzyliśmy w solucji plik mapowania bazy danych na obiekty klas. Ręcznie stworzona została klasa kontekstu, względem której wykonywane są wszystkie zapytania do bazy danych.

2.3 UŻYTE NARZĘDZIA

W trakcie pracy nad aplikacją wykorzystywaliśmy następujące narzędzia i środowiska:

- Microsoft Visual Studio 2010 Professional
- Microsoft SQL Server 2005
- Telerik WinForms Controls
- TortoiseSVN

- Framework NUnit
- ReSharper
- Portal www.code.google.com jako repozytorium

Poniżej znajduje się opis wymienionych narzędzi.

2.3.1 Microsoft Visual Studio 2010 Professional

Microsoft Visual Studio 2010 jest to zintegrowany pakiet narzędzi programistycznych dla platformy .NET. Pozwala on na pisanie aplikacji desktopowych, sieciowych oraz aplikacji webowych. Samo środowisko wspiera języki programowania takie jak:

- C#
- Visual Basic .NET
- F#
- C++
- J# (do wersji Microsoft Visual Studio 2005)

W wersji 2010 z jakiej my korzystamy dodano względem poprzednich wersji: .NET Framework 4, wsparcie dla SQL Server 2008 oraz nowe opcje do testowania oprogramowania.

W środowisko to wbudowany jest edytor kodu wspierający IntelliSense (inteligentne autouzupełnianie tworzonego kodu – znacznie przyspiesza pisanie aplikacji), debugger, kreatory do tworzenia interfejsu użytkownika, kreatory do tworzenia klas i schematów baz danych.

Skorzystaliśmy z wersji Professional gdyż jest ona dla nas darmowa jako dla studentów, oraz przede wszystkim udostępnia możliwość tworzenia testów jednostkowych z użyciem już wbudowanego narzędzia jakim jest MSTest.

Samo środowisko jest popularnym środowiskiem w firmach programistycznych i jego znajomość jest bardzo mile widziana przez pracodawców.

2.3.2 Microsoft SQL Server 2005

Do przechowywania danych wybraliśmy relacyjna bazę danych Microsoft SQL Server 2005 w wersji Express. Jest to wersja darmowa do zastosowań niekomercyjnych. Zdecydowaliśmy się na ten rodzaj bazy danych z racji tego, iż świetnie integruje się ona ze środowiskiem Microsoft Visual Studio 2010 Professional. Do zarządzania bazą danych korzystamy z Microsoft SQL Server 2005 Management Studio. Pozwoliło nam to na bardzo łatwe stworzenie schematu naszej bazy danych, na podstawie którego automatycznie zostały wygenerowane tabele.

Zrezygnowaliśmy z korzystania z baz danych innych firm (Oracle, MySql itd.), gdyż nie są one wspierane tak dobrze jak serwery bazodanowe firmy Microsoft przez Visual Studio.

2.3.3 Telerik WinForms Controls

Telerik WinForms Controls to zestaw kontrolek do aplikacji desktopowych tworzonych przy użyciu technologii WinForms. Zdecydowaliśmy się na ich użycie, gdyż standardowe kontrolki dostępne w Visual Studio pod względem wizualnym nie prezentują się ciekawie.

Jesteśmy zdania, że poza spełnianiem swoich zadań aplikacja powinna również być przyjazna dla użytkownika, co w naszym rozumieniu oznacza przejrzystość interfejsu oraz atrakcyjność wizualną.

W naszym przypadku skorzystaliśmy z wersji testowej tych kontrolek (pełne wersje są płatne, także do zastosowań niekomercyjnych). Jedynym znakiem, iż aplikacja wykorzystuje testową wersję kontrolek jest pojawiające się od czasu do czasu w trakcie korzystania aplikacji okienko o korzystaniu z wersji testowej.

TU BĘDZIE RYSUNEK

2.3.4 TortoiseSVN

TortoiseSVN jest programem do kontroli źródła, wersji, edycji pliku dla systemów Windows. Jest on oparty na programie Subversion, zapewnia miły dla oka i przyjemny w

użytkowaniu interfejs. Program ten jest programem całkowicie darmowym, również dla zastosowań komercyjnych.

TortoiseSVN jest jednym z najpopularniejszych programów zapewniających możliwość wersjonowania plików. Za jego pomocą możemy utworzyć lokalne repozytorium projektu. Mając projekt na zdalnym serwerze (w naszym wypadku korzystaliśmy z serwisu Google Code), wiele osób ma możliwość wprowadzania lokalnie zmian do poszczególnych plików i wysyłania tych plików na serwer jako nową wersję. Kolejna osoba może pobrać takie pliki i jeżeli występują jakieś konflikty (osoba pobierająca również wprowadziła zmiany do tych samych plików, które wykluczają zmiany w nowej wersji) – w łatwy sposób je rozwiązać.

Zdecydowaliśmy się na użycie tego programu gdyż jest on powszechnie wykorzystywany w firmach programistycznych i wielu pracodawców wymaga jego znajomości.

2.3.5 Framework NUnit

NUnit jest frameworkiem do tworzenia testów jednostkowych dla aplikacji tworzonych we wszystkich językach dla platformy .NET. Pierwotnie było on wzorowany na dostępnym w Javie frameworku JUnit.

W językach takich jak C# można dziedziczyć tylko z jednej klasy, co powoduje problemy przy tworzeniu testów jednostkowych. NUnit rozwiązuje te problemy poprzez wykorzystanie specyficznej cechy języka C# - atrybutów, do oznaczenia klas i metod testowych. Dzięki temu nadal możemy korzystać z dziedziczenia oraz jednocześnie tworzyć testy jednostkowe naszych metod.

2.3.6 ReSharper

ReSharper to dodatek to Visual Studio wspomagający pracę z kodem aplikacji. Jest to komercyjne narzędzie, które znacznie usprawnia proces refactoringu kodu, nawigowania pomiędzy powiązanymi klasami. Umożliwia również pilnowanie stylu kodowania, gdy projekt jest tworzony przez kilku programistów.

Jest to narzędzie komercyjne, płatne, jednak my korzystaliśmy z 30 – dniowej wersji testowej.

2.3.7 Google Code

Jako repozytorium dla projektu użyliśmy serwisu Google Code. Zastanawialiśmy się również nad serwisami opartymi o system Jira, ale doszliśmy do wniosku, iż tak rozbudowany system nie jest potrzebny dla tak małego projektu.

Serwis Google Code pozwala na tworzenie witryn połączonych z repozytorium dla projektów niekomercyjnych. Dostępny jest tam pewnego rodzaju moduł wiki oraz moduł notek.

Serwis udostępnia również możliwość podglądu plików źródłowych, porównywania kolejnych wersji plików, a także możliwość komentowania kodu.

W naszym przypadku serwis był wykorzystywany przede wszystkim jako repozytorium projektu, z którym łączyliśmy się przy pomocy programu TortoiseSVN. Korzystaliśmy również z systemu notek udostępnianych przez Google Code.

3. SPECYFIKACJA ZEWNĘTRZNA

4. SPECYFIKACJA WEWNĘTRZNA

5. URUCHAMIANIE I TESTOWANIE

6. WNIOSKI

7. BIBLIOGRAFIA

1. http://msdn.microsoft.com/en-us/library/bb386964.aspx
2. http://msdn.microsoft.com/en-us/library/w0x726c2.aspx
3. http://pl.wikipedia.org/wiki/.NET_Framework
4. http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
5. http://www.nunit.org/index.php?p=home
6. http://www.jetbrains.com/resharper/index.html

SPIS TREŚCI

1. WSTĘP 3

2. ANALIZA TEMATU 4

2.1 ZAŁOŻENIA 6

2.2 UŻYTE TECHNOLOGIE 7

2.2.1 Microsoft .NET 4 i język C# 7

2.2.2 LINQ to Entities 8

2.3 UŻYTE NARZĘDZIA 8

2.3.1 Microsoft Visual Studio 2010 Professional 9

2.3.2 Microsoft SQL Server 2005 10

2.3.3 Telerik WinForms Controls 10

2.3.4 TortoiseSVN 10

2.3.5 Framework NUnit 11

2.3.6 ReSharper 11

2.3.7 Google Code 12

3. SPECYFIKACJA ZEWNĘTRZNA 12

4. SPECYFIKACJA WEWNĘTRZNA 12

5. URUCHAMIANIE I TESTOWANIE 12

6. WNIOSKI 12

7. BIBLIOGRAFIA 12

SPIS TREŚCI 13