



POLITECHNIKA ŚLĄSKA
WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Projekt inżynierski

Edytor i aplikacja udostępniająca plany studiów

Autorzy: Łukasz Kulig, Krzysztof Sałajczyk
Kierujący pracą: dr inż. Robert Tutajewicz

Gliwice, styczeń 2011.

Spis treści

1.	WSTĘP	8
2.	ANALIZA TEMATU	9
2.1	ZAŁOŻENIA	11
2.2	UŻYTE TECHNOLOGIE	12
2.2.1	Microsoft .NET 4 i język C#	12
2.2.2	LINQ to Entities.....	12
2.2.3	Wzorzec MVC	13
2.3	UŻYTE NARZĘDZIA.....	13
2.3.1	Microsoft Visual Studio 2010 Professional	13
2.3.2	Microsoft SQL Server 2005.....	14
2.3.3	Telerik WinForms Controls	14
2.3.4	TortoiseSVN	15
2.3.5	Framework NUnit	16
2.3.6	ReSharper.....	16
2.3.7	Google Code	16
3.	SPECYFIKACJA ZEWNĘTRZNA.....	16
3.1	PROCES INSTALACJI.....	17
3.2	LOGOWANIE	17
3.3	GŁÓWNE OKNO APLIKACJI.....	17
3.4	TWORZENIE PLANU.....	18
3.4.1	Tworzenie nowego planu.....	19

3.4.2	Wczytywanie planu.....	19
3.4.3	Dodawanie przedmiotów do planu	20
3.4.4	Edytowanie i usuwanie przedmiotów	21
3.4.5	Tworzenie reguł weryfikacji planu	22
3.4.6	Weryfikacja planu.....	23
3.5	ZARZĄDZANIE WYDZIAŁAMI	24
3.5.1	Dodawanie wydziału.....	24
3.5.2	Edycja wydziału.....	25
3.5.3	Usuwanie wydziału.....	25
3.6	ZARZĄDZANIE KIERUNKAMI	25
3.6.1	Dodawanie kierunku	25
3.6.2	Edycja kierunku	26
3.6.3	Usuwanie kierunku	26
3.7	ZARZĄDZANIE TYPAMI PRZEDMIOTÓW	26
3.7.1	Dodawanie typu przedmiotu.....	26
3.7.2	Edycja typu przedmiotu	27
3.7.3	Usuwanie typu przedmiotu	27
3.8	ZARZĄDZANIE TYPAMI STUDIÓW	27
3.8.1	Dodawanie typu studiów	27
3.8.2	Edycja typu studiów.....	28
3.8.3	Usuwanie typu studiów.....	28

3.9	ZARZĄDZANIE INSTYTUTAMI	28
3.9.1	Dodawanie instytutu	28
3.9.2	Edycja instytutu	29
3.9.3	Usuwanie instytutu	29
3.10	ZARZĄDZANIE SPECJALIZACJAMI.....	29
3.10.1	Dodawanie specjalizacji.....	30
3.10.2	Edycja specjalizacji.....	30
3.10.3	Usuwanie specjalizacji	30
3.11	ZARZĄDZANIE SEMESTRAMI	30
3.11.1	Dodawanie semestru	31
3.11.2	Edycja semestru.....	31
3.11.3	Usuwanie semestru.....	31
3.12	PRZEGLĄD PLANU	31
3.12.1	Wczytywanie planu	32
3.12.2	Zapis planu	32
3.12.3	Informacje o planie.....	32
3.13	ARCHIWUM	33
3.13.1	Wczytywanie planu	33
3.13.2	Kopiowanie planu	33
3.13.3	Informacje o planie.....	34
3.14	ZARZĄDZANIE UŻYTKOWNIKAMI.....	34

3.14.1	Dodawanie użytkownika	34
3.14.2	Edycja użytkownika	34
3.14.3	Usuwanie użytkownika	35
3.15	PRZEGLĄDANIE PLANU – WERSJA WEBOWA	35
3.15.1	Filtr	35
3.15.2	Przeglądanie planu	35
4.	SPECYFIKACJA WEWNĘTRZNA.....	37
4.1	APLIKACJA DESKTOPOWA	37
4.1.1	Prezentowanie idei	37
4.1.2	Diagram przypadków użycia	38
4.1.3	Konstrukcja aplikacji	38
4.1.4	Organizacja bazy danych	39
4.1.5	Przegląd ważniejszych klas i szczegóły implementacyjne	42
4.2	APLIKACJA WEBOWA.....	46
4.2.1	Prezentowanie idei	46
4.2.2	Konstrukcja aplikacji	47
4.2.3	Przegląd ważniejszych klas i szczegóły implementacyjne	47
5.	TESTOWANIE.....	49
5.1	TESTY JEDNOSTKOWE	49
5.2	TESTY FUNKCJONALNE.....	50
5.3	TESTY STRUKTURALNE.....	51
5.4	PODSUMOWANIE TESTÓW.....	51

6.	UWAGI KOŃCOWE	52
7.	BIBLIOGRAFIA	53
	SPIS RYSUNKÓW	54

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 część webową. Część desktopowa powinna umożliwiać zarządzanie planem (tworzenie i edycja), oraz zarządzaniem danymi potrzebnymi 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 zbiorem 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.

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.

Komentarz [R1]: Określić co kto robił

Komentarz [R2]: Zastanowić się czy nie wyróżniać wizualnie (inna czcionka, kursywa ?) takich elementów jak elementy interfejsu użytkownika (tytuły okien, teksty etykietek) oraz (inaczej) nazw będących identyfikatorami w programie.

Komentarz [R3]: Poszczególne moduły zostaną opisane w kolejnych punktach

Komentarz [R4]: oraz sugestii

2. ANALIZA TEMATU

Celem pracy było stworzenie aplikacji, która wspomogłaby 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, artystyczna). 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 wysnuło wniosek, iż aplikacja powinna w jak największym stopniu pozwalać definiować plan. Nie chodzi tu tylko o listę przedmiotów na danym semestrze. Aplikacja powinna pozwalać także na definiowanie nowych wydziałów, kierunków, typów przedmiotów, instytutów działających na danym wydziale, specjalizacji na danym kierunku, oraz semestrów. Takie możliwości edycji są spowodowane tym, iż co roku są wprowadzane zmiany w jednostkach uczelnianych, tworzone są nowe kierunki, a czasem całe wydziały. Zamierzeniem było stworzenie aplikacji, która pomimo zmian organizacyjnych nadal będzie mogła być w pełni funkcjonalna.

Analiza poszczególnych modułów zarządzających elementami składowymi planu:

Komentarz [R5]: Może jako podpunkt

- 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 „Konwersatorium” już może 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 aplikacji ma być również weryfikacja planu pod względem różnych ustalonych zasad. Dla konkretnego planu mogą być tworzone pewnego rodzaju reguły, a następnie program 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 określono główne założenia tworzonej aplikacji desktopowej:

- 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 / usuwanie przedmiotów,
 - recenzja planu – 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 plików o formacie pdf i xml.
- Zarządzanie użytkownikami:
 - tworzenie / edycja / usuwanie użytkowników,
 - tworzenie / edycja / usuwanie ról.

Natomiast aplikacja webowa ma następujące założenia:

- Udostępnianie planów do odczytu.
- Dostęp anonimowy.
- Filtracja planów.

2.2 UŻYTE TECHNOLOGIE

Do realizacji projektu zdecydowano 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 programowania¹, 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. Wiele 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 wykorzystano technologię Windows Forms (aplikacja desktopowa) oraz ASP.NET (aplikacja webowa), przy czym nie wykorzystano w aplikacji desktopowej standardowych kontrolki dostępnych w Visual Studio, ale kontrolki firmy Telerik.

Komentarz [R6]: odwołanie

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.

¹ Informacje zaczerpnięte z witryny <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

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

Komentarz [R7]: Solucja (z [ang. solution](#) – rozwiązanie) – tekst zawierający opis, jak ukończyć [grę komputerową](#) (za wikipedią); chyba lepsze inne słowo

2.2.3 Wzorzec MVC

W części webowej wykorzystany został wzorzec projektowy MVC (Model – View – Controller: Model – Widok – Kontroler). Wzorzec ten składa się z trzech elementów:

- Model – opisuje logikę aplikacji, warstwa danych.
- Widok – warstwa prezentacji, najczęściej jest to GUI.
- Kontroler – steruje widokiem na podstawie danych (np. wymusza odświeżenie widoku po zmianie danych).

Wzorzec ten poprzez rozdzielenie warstwy danych od warstwy prezentacji umożliwia łatwe zarządzanie każdą z nich i niezależne wykorzystywanie warstwy danych w różnych typach aplikacji. Dzięki temu zmiany wprowadzone w jednym miejscu nie wymuszają zmian w drugim. Odpowiednim przetworzeniem danych tak, aby widok mógł z nich skorzystać zajmuje się kontroler.

2.3 UŻYTE NARZĘDZIA

W trakcie pracy nad aplikacją wykorzystywaliśmy następujące narzędzia i środowiska:

Komentarz [R8]: Kolejnych wiszących liter nie będę już zaznaczał

- 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 korzystano przy realizacji pracy 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.

Wykorzystano wersję Professional gdyż jest ona (poprzez system MSDN AA) dla studentów darmowa, 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 wybrana została relacyjna baza danych Microsoft SQL Server 2005 w wersji Express. Jest to wersja darmowa do zastosowań niekomercyjnych. Zdecydowano 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 wykorzystano Microsoft SQL Server 2005 Management Studio. Pozwoliło to na bardzo łatwe stworzenie schematu bazy danych, na podstawie którego automatycznie zostały wygenerowane tabele.

Zrezygnowano 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 kontrolki do aplikacji desktopowych stworzonych przy użyciu technologii WinForms. Zdecydowano się na ich użycie, gdyż standardowe kontrolki dostępne w Visual Studio pod względem wizualnym nie prezentują się ciekawie.

Aplikacja poza spełnianiem swoich zadań powinna również być przyjazna dla użytkownika, co oznacza także przejrzystość interfejsu oraz atrakcyjność wizualną.

Skorzystano z wersji testowej tych kontroltek (pełne wersje są płatne, także do zastosowań niekomercyjnych). Jedynym znakiem, iż aplikacja wykorzystuje testową wersję kontroltek jest pojawiające się od czasu do czasu w trakcie działania programu okienko o korzystaniu z wersji testowej.



Rysunek 1 Okno informujące o wersji testowej kontroltek

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 do zastosowań komercyjnych.

TortoiseSVN jest jednym z najpopularniejszych programów zapewniających możliwość wersjonowania plików. Za jego pomocą można tworzyć lokalne repozytorium projektu. Mając projekt na zdalnym serwerze (w naszym wypadku wykorzystano serwis 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.

Zdecydowano 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 był 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 metod.

2.3.6 ReSharper

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

Jest to narzędzie komercyjne, płatne, jednak w projekcie wykorzystano 30 – dniową wersję testową.

2.3.7 Google Code

Jako repozytorium dla projektu użyto serwisu Google Code. Zastanawiano się również nad serwisami opartymi o system Jira, ale 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 przypadku tego projektu serwis był wykorzystywany przede wszystkim jako repozytorium, z którym łączono się przy pomocy programu TortoiseSVN. Korzystano również z systemu notek udostępnianych przez Google Code.

3. SPECYFIKACJA ZEWNĘTRZNA

W tym rozdziale przedstawiona zostanie instrukcja obsługi aplikacji desktopowej oraz webowej, a także proces instalacji.

3.1 PROCES INSTALACJI

W celu uruchomienia i poprawnego działania aplikacji desktopowej oraz webowej należy wykonać podane niżej czynności w takiej kolejności, jak są wypisane:

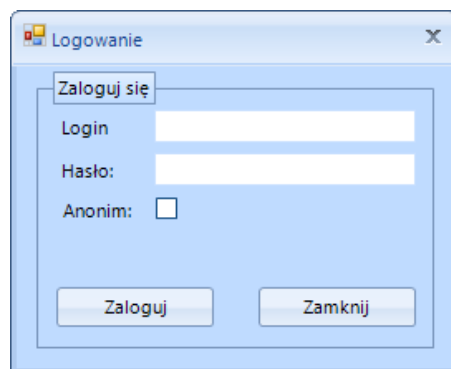
- Instalacja .NET Framework wersja 4.0.
- Instalacja Microsoft SQL Server 2005 Express Edition.
- Wykonanie skryptu tworzącego bazę danych.
- Wykonanie skryptu wypełniającego bazę przykładowymi danymi.
- Modyfikacja plików App.config oraz Web.config w celu ustawienia poprawnego Connection Stringa do połączenia z bazą danych.
- Dla aplikacji webowej: instalacja serwer IIS oraz utworzenie witryny WWW na serwerze wskazującej na katalog aplikacji webowej.

Komentarz [R9]: Czy jest gdzieś treść tego skryptu

Komentarz [R10]: Czy słowa w języku angielskim odmieniają się jak w polskim ?

3.2 LOGOWANIE

Po uruchomieniu programu użytkownikowi ukazuje się poniższe okno (Rysunek 2). Okno to służy do uwierzytelniania użytkownika. Użytkownik powinien wpisać swój login oraz hasło w odpowiednie pola a następnie kliknąć przycisk 'Zaloguj'. Jeżeli podane są błędne dane logowania użytkownik dostanie stosowną informację.



Rysunek 2 Forma logowania

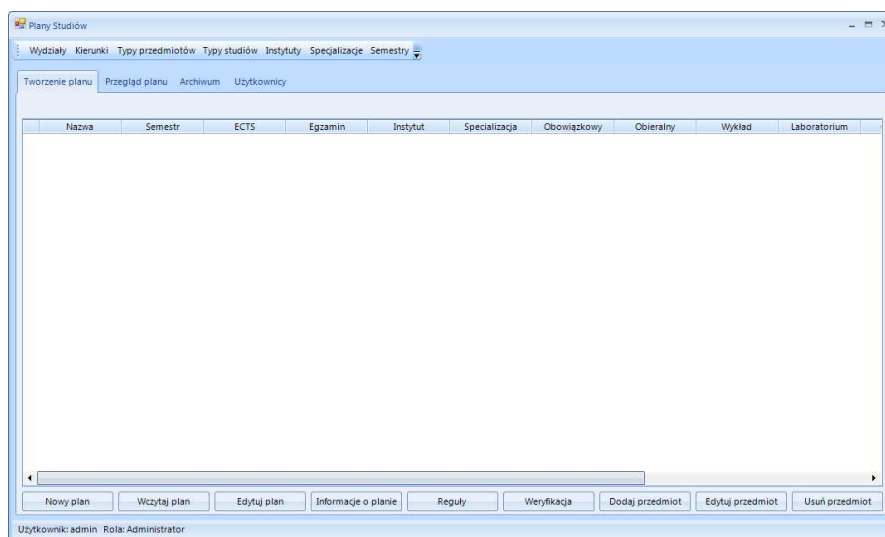
Komentarz [R11]: Raczej formularz (można być w dobrej formie)

3.3 GŁÓWNE OKNO APLIKACJI

Po zalogowaniu użytkownikowi ukazuje się główne okno aplikacji (Rysunek 3). Z tej formy użytkownik ma dostęp do wszystkich funkcji programu (określonych przez posiadaną przez niego rolę). Pokazany tutaj został przykład użytkownika o prawach administracyjnych – posiadającego – posiadającego dostęp do wszystkich funkcji programu.

Komentarz [R12]: Dalej uwag o formach nie piszę ale proszę o konsekwentne stosowanie słowa formularz

Użytkownik ma możliwość zarządzania: wydziałami, kierunkami, typami przedmiotów i studiów, instytutami, specjalizacjami, regułami, semestrami oraz może zweryfikować plan na podstawie utworzonych wcześniej reguł.



Rysunek 3 Główne okno aplikacji

Opcje związane z planem zostały podzielone na zakładki – ‘*Tworzenie planu*’ (udostępnia opcje bezpośrednio związane z tworzeniem nowego planu), ‘*Przegląd planu*’ (udostępnia opcje takie, jak przeglądanie już gotowych planów oraz ich eksport do plików w formacie *.pdf oraz *.xml; dodatkowo możliwe jest wyświetlenie informacji o danym planie). Zakładka ‘*Archiwum*’ służy do przeglądania planów archiwalnych (takich, które nie są już obowiązujące), oraz umożliwia także stworzenie nowego planu na podstawie planu archiwalnego. Ostatnia zakładka – ‘*Użytkownicy*’ służy do zarządzania użytkownikami.

Do zarządzania każdym z elementów służą osobne moduły, które zostały przedstawione w postaci osobnych okien, do których odnośniki znajdują się na pasku narzędziowym, bądź też są umiejscowione w odpowiednich zakładkach lub innych formach.

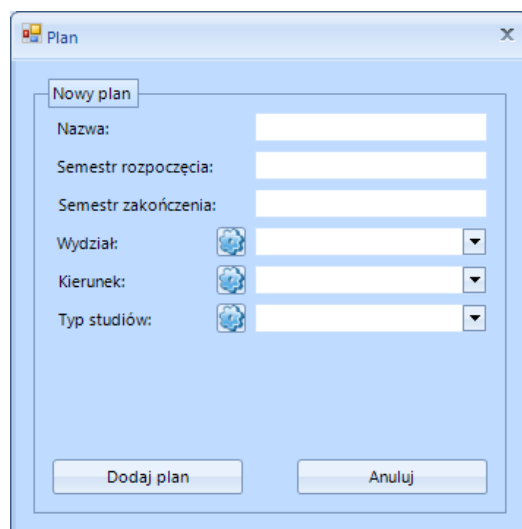
3.4 TWORZENIE PLANU

W tym rozdziale zostanie opisany proces tworzenia nowego oraz edycji istniejącego planu. Przedstawiona zostanie również możliwość skopiowania przedmiotów z planu zarchiwizowanego do tworzonego.

3.4.1 Tworzenie nowego planu

W celu stworzenia nowego planu użytkownik musi przejść na zakładkę *'Tworzenie planu'*, a następnie kliknąć przycisk *'Nowy plan'*, który wyświetli okno odpowiedzialne za tworzenie planu (Rysunek 4). W pole *'Nazwa'* należy wpisać nazwę tworzonego planu, w pola *'Semestr rozpoczęcia'* oraz *'Semestr zakończenia'* odpowiednio wartości liczbowe określające numery semestrów początkowego oraz końcowego. Należy także wybrać dostępny wydział, kierunek i typ studiów. Jeżeli nie ma do wyboru powyższych danych należy je ręcznie dodać korzystając z przycisków znajdujących się przy odpowiednich listach rozwijanych (dodawanie tych elementów zostało opisane w rozdziałach 3.5, 3.6 oraz 3.7.) Po uzupełnieniu danych należy kliknąć przycisk *'Dodaj plan'*. Po tej akcji użytkownik zostanie poinformowany o tym, iż plan został stworzony, następnie okno dodawania planu zostanie zamknięte.

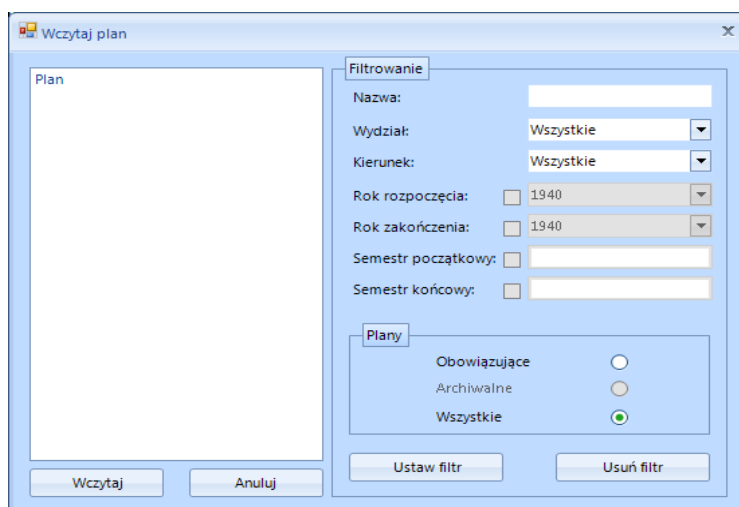
W przypadku podania błędnych danych, bądź nie podania ich wcale, użytkownik zostanie poinformowany o tym.



Rysunek 4 Tworzenie nowego planu

3.4.2 Wczytywanie planu

Aplikacja zapewnia możliwość wczytania planu. W celu wczytania już utworzonego wcześniej planu należy przejść na zakładkę *'Tworzenie planu'* i kliknąć przycisk *'Wczytaj plan'*. Po kliknięciu powinno pojawić się okno wczytywania planu (Rysunek 5). Po lewej stronie powinna pojawić się lista dostępnych planów.



Rysunek 5 Wczytywanie planu

Wygodne wczytywanie zapewnia filtr. Zapewnione zostało wyszukiwanie po nazwie, wydziałach, kierunkach, roku rozpoczęcia i zakończenia oraz semestrach początkowym i końcowym. Dodatkowo jest dostępna opcja wyszukiwania planów po ich statusie – obowiązujący, archiwalny bądź przeszukiwanie po obu (nie wszystkie opcje filtrowania po statusie są dostępne, np. gdy użytkownik chce załadować plan do edycji to opcja filtrowania po statusie archiwalnym nie jest dostępna, gdyż nie można edytować planu archiwalnego). Po sprecyzowaniu filtra należy kliknąć przycisk *‘Ustaw filtr’*. W celu wczytania konkretnego planu należy go wybrać z listy, a następnie kliknąć przycisk *‘Wczytaj’*.

3.4.3 Dodawanie przedmiotów do planu

Po utworzeniu nowego planu można rozpocząć dodawanie do niego przedmiotów. W tym celu należy kliknąć przycisk *‘Dodaj przedmiot’*. Pojawi się okno przedstawione na rysunku 6. Dla każdego przedmiotu można ustawić nazwę, semestr, na którym obowiązuje, ilość punktów ECTS, instytut (dodawanie instytutów i semestrów, w przypadku ich braku zostało opisane w rozdziałach 3.9 i 3.11). Można ustawić także to, czy przedmiot jest obowiązkowy lub obieralny oraz czy jest na specjalizacji jako przedmiot obieralny lub obowiązkowy. Należy także zdefiniować typ przedmiotu wraz z ilością godzin (dodawania typów przedmiotów oraz specjalizacji w przypadku ich braku zostało opisane w rozdziałach 3.7 i 3.10).

Zarządzanie przedmiotami - dodawanie przedmiotu

Dodaj przedmiot

Nazwa:

Semestr:

Punkty ECTS:

Egzamin: ☐

Kierunek: Informatyka

Wydział: AEII

Instytut:

Obowiązkowy: ☐

Obieralny: ☐

Specjalizacja	Obowiązkowy	Obieralny	Typ	Godziny
Click here to add a new row				

Rysunek 6 Dodawanie przedmiotu do planu

Po uzupełnieniu danych należy kliknąć przycisk *'Dodaj przedmiot'*. W przypadku braku wymaganych, bądź podaniu błędnych danych użytkownik zostanie powiadomiony o tym fakcie w stosowny sposób. Operację dodawania przedmiotów należy powtarzać, aż do wprowadzenia wszystkich przedmiotów przewidzianych dla konkretnego planu.

3.4.4 Edytowanie i usuwanie przedmiotów

Istnieje możliwość poprawy błędów popełnionych przy dodawaniu przedmiotów. W tym celu należy wczytać odpowiedni plan korzystając z zakładki *'Tworzenie planu'*, a następnie wybrać przedmiot, który chcemy edytować bądź usunąć z tabeli. Po wybraniu przedmiotu w zależności od akcji, którą chcemy wykonać, należy kliknąć przycisk *'Edytuj przedmiot'* bądź *'Usuń przedmiot'*.

W przypadku edycji zostanie wyświetlone okno edycji przedmiotu (Rysunek 7). Okno to daje możliwość zmiany danych, które można ustalić podczas dodawania przedmiotu (nazwę, semestr, na którym obowiązuje, ilość punktów ECTS, instytut, przedmiot jako obowiązkowy lub obieralny, dostępny na specjalizacji jako przedmiot obieralny lub obowiązkowy, typy przedmiotu wraz z ilością godzin).

Edytuj przedmiot

Nazwa: Podstawy Informatyki

Semestr: Pierwszy

Punkty ECTS: 5,0

Egzamin: ☐

Kierunek: Informatyka

Wydział: Automatyki Elektroniki i Informatyki

Instytut: Brak

Obowiązkowy: ☒

Obieralny: ☐

Specjalizacja	Główny	Obieralny

Typ	Godziny
Wykład	2
Laboratorium	0
Ćwiczenia	2
Projekt	0
Seminarium	0
WF	0

Wyczyść specjalizacje

Zapisz zmiany Zamknij

Rysunek 7 Edycja przedmiotu

W przypadku wybrania usuwania przedmiotu wybrany przez użytkownika przedmiot zostanie usunięty.

3.4.5 Tworzenie reguł weryfikacji planu

W celu dodania reguł weryfikacji planu należy po stworzeniu lub wczytaniu planu kliknąć przycisk 'Reguły'. Wyświetli się okno, które pozwala na dodawanie reguł do planu (Rysunek 8). Zdefiniowane wcześniej reguły zostaną przedstawione w formie tabelki.

Dostępnych jest sześć typów reguł (3 dotyczące punktów ECTS oraz 3 dotyczące liczby godzin), które można definiować dzięki dodatkowym opcjom – typy przedmiotu, przedmioty oraz zakres obowiązywania reguły (konkretny semestr bądź cały plan).

Rysunek 8 Dodawanie reguł

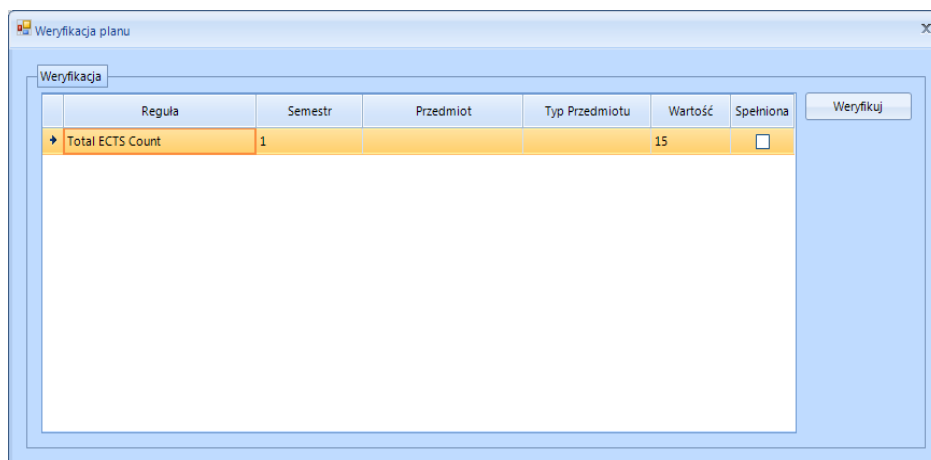
W celu dodania reguły do planu należy zaznaczyć typ reguły, a następnie wybrać dodatkowe opcje definiujące regułę. W przypadku reguł odnoszących się do typu przedmiotu należy wybrać typ przedmiotu, a w przypadku reguł odnoszących się do przedmiotu należy wybrać przedmiot. Jeżeli do planu nie został dodany żaden przedmiot to reguły dotyczące konkretnego przedmiotu nie będą dostępne. Dodatkowe opcje typu liczba punktów ECTS oraz liczba godzin należy wpisać w odpowiednie pola reguł. Należy także wybrać zakres obowiązywania reguł. Po zdefiniowaniu reguł dla planu należy kliknąć przycisk 'Dodaj regułę'. Nowe reguły zostaną dodane do tabeli. W przypadku podania błędnych lub niekompletnych danych użytkownik zostanie poinformowany o ewentualnych błędach w stosowny sposób.

3.4.6 Weryfikacja planu

W celu weryfikacji planu należy kliknąć przycisk 'Weryfikuj'. Po kliknięciu powinno pojawić się okno weryfikacji planu (Rysunek 9).

W przypadku, gdy nie zdefiniowano żadnych reguł, użytkownik zostanie o tym stosownie poinformowany (w celu dodania reguł weryfikacji planu patrz rozdział 3.4.5).

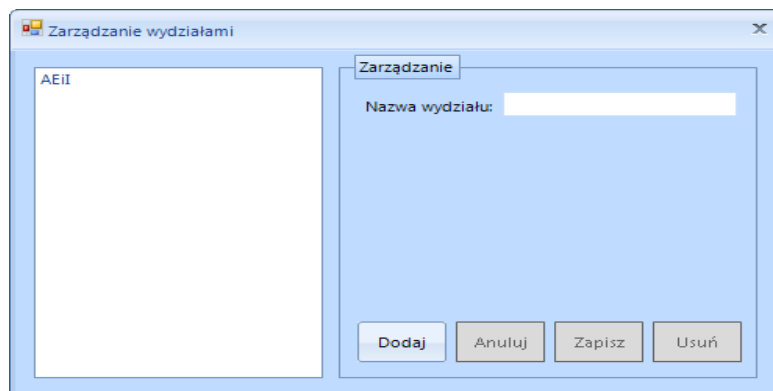
Po pojawieniu się okna w tabeli zostaną przedstawione reguły utworzone dla danego planu. W celu weryfikacji planu należy kliknąć przycisk 'Weryfikuj'. Użytkownik zostanie poinformowany o wynikach weryfikacji poprzez kolumnę 'Spełniona' znajdującą się w tabeli z regułami.



Rysunek 9 Weryfikacja planu

3.5 ZARZĄDZANIE WYDZIAŁAMI

W celu zarządzania wydziałami należy kliknąć przycisk 'Wydziały', znajdujący się na pasku narzędziowym w głównym oknie aplikacji. Po jego kliknięciu wyświetli się okno do zarządzania wydziałami (Rysunek 10).



Rysunek 10 Zarządzanie wydziałami

3.5.1 Dodawanie wydziału

W celu dodania nowego wydziału należy wpisać jego nazwę w pole 'Nazwa wydziału', a następnie kliknąć przycisk 'Dodaj'. Jeżeli użytkownik będzie chciał dodać wydział, który już istnieje, bądź nie poda nazwy wydziału, zostanie poinformowany o błędzie. Po dodaniu wydziału, pojawi się on na liście.

3.5.2 Edycja wydziału

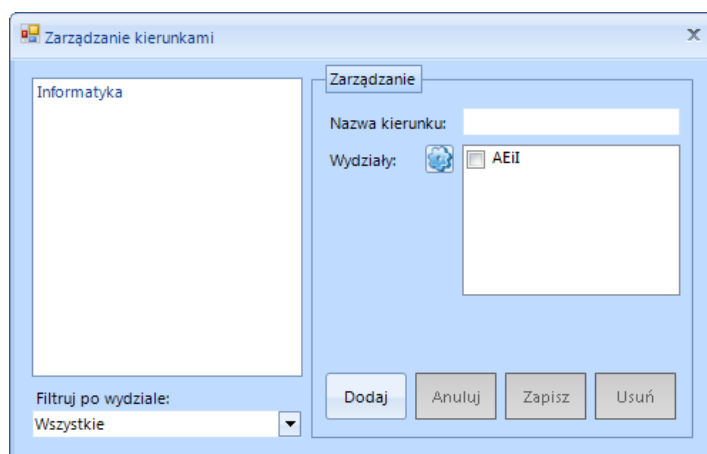
W celu edycji danego wydziału należy wybrać wydział dostępny na liście klikając w niego dwukrotnie. Następnie zmienić jego nazwę i zapisać zmiany klikając przycisk 'Zapisz'. Aby anulować zmiany należy kliknąć przycisk 'Anuluj'.

3.5.3 Usuwanie wydziału

Akcja usuwania wygląda podobnie do edycji z tym, że należy wybrać wydział i kliknąć przycisk 'Usuń'. Użytkownik zostanie powiadomiony, jeżeli nie uda się usunąć danego wydziału.

3.6 ZARZĄDZANIE KIERUNKAMI

W celu zarządzania kierunkami należy kliknąć przycisk 'Kierunki', znajdujący się na pasku narzędziowym w głównym oknie aplikacji. Po jego pojawieniu się okno widoczne na poniższym rysunku.



Rysunek 11 Zarządzanie kierunkami

3.6.1 Dodawanie kierunku

W celu dodania nowego wydziału należy wpisać jego nazwę w pole 'Nazwa kierunku', wybrać wydział, na którym kierunek będzie dostępny (można wybrać więcej niż jeden), a następnie kliknąć przycisk 'Dodaj'. Jeżeli użytkownik będzie chciał dodać kierunek, który już istnieje, bądź nie poda nazwy kierunku zostanie poinformowany o błędzie. Po dodaniu kierunku pojawi się na liście. W przypadku, gdy nie ma na liście żadnego wydziału użytkownik może dodać go poprzez formę zarządzania wydziałami. Opis dodawania wydziału został przedstawiony w rozdziale 3.5.

3.6.2 Edycja kierunku

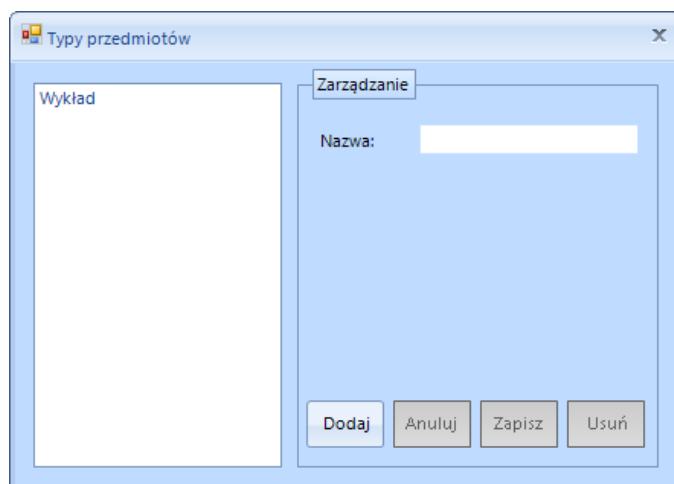
W celu edycji należy wybrać kierunek dostępny na liście klikając w niego dwukrotnie. Następnie zmienić jego nazwę lub wydział, na którym ma być dostępny i zapisać zmiany klikając przycisk 'Zapisz'. W celu anulowania edycji należy kliknąć przycisk 'Anuluj'. Został zastosowany tutaj filtr, który pozwala na szukanie kierunku na danym wydziale, bądź też na wszystkich wydziałach. Filtr uaktywnia się po wyborze wartości z listy rozwijanej 'Filtruj po wydziale'.

3.6.3 Usuwanie kierunku

Akcja usuwania wygląda podobnie do edycji z tym, że należy wybrać kierunek i kliknąć przycisk 'Usuń'. Użytkownik zostanie powiadomiony, jeżeli nie uda się usunąć danego kierunku.

3.7 ZARZĄDZANIE TYPMI PRZEDMIOTÓW

W celu zarządzania typami przedmiotów należy kliknąć przycisk 'Typy przedmiotów', znajdujący się na pasku narzędziowym w głównym oknie aplikacji. Po jego kliknięciu pojawi się okno zarządzania typami przedmiotów (Rysunek 12).



Rysunek 12 Zarządzanie typami przedmiotów

3.7.1 Dodawanie typu przedmiotu

W celu dodania nowego typu przedmiotu należy wpisać jego nazwę w pole 'Nazwa', a następnie kliknąć przycisk 'Dodaj'. Jeżeli użytkownik będzie chciał dodać typ,

który już istnieje, bądź nie poda jego nazwy, zostanie poinformowany o błędzie. Po dodaniu typu pojawi się on na liście.

3.7.2 Edycja typu przedmiotu

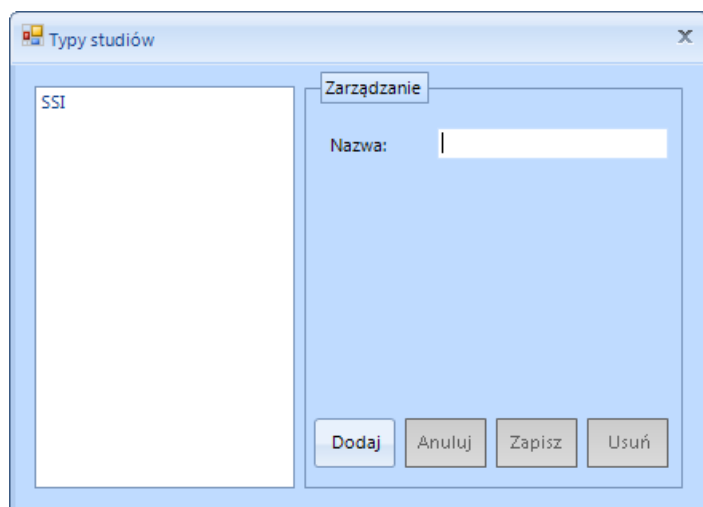
W celu edycji danego typu należy wybrać typ dostępny na liście, klikając w niego dwukrotnie. Następnie zmienić jego nazwę i zapisać zmiany klikając przycisk 'Zapisz'. W celu anulowania zmiany należy kliknąć przycisk 'Anuluj'.

3.7.3 Usuwanie typu przedmiotu

Akcja usuwania wygląda podobnie do edycji z tym, że należy wybrać typ przedmiotu i kliknąć przycisk 'Usuń'. Użytkownik zostanie powiadomiony, jeżeli nie uda się usunąć danego typu.

3.8 ZARZĄDZANIE TYPAMI STUDIÓW

W celu zarządzania typami studiów należy kliknąć przycisk 'Typy studiów', znajdujący się na pasku narzędziowym w głównym oknie aplikacji. Po jego kliknięciu pojawi się okno zarządzania typami studiów (Rysunek 13).



Rysunek 13 Zarządzanie typami studiów

3.8.1 Dodawanie typu studiów

W celu dodania nowego typu przedmiotu należy wpisać jego nazwę w pole 'Nazwa', a następnie kliknąć przycisk 'Dodaj'. Jeżeli użytkownik będzie chciał dodać typ,

który już istnieje, bądź nie poda jego nazwy, zostanie poinformowany o błędzie. Po dodaniu typu pojawi się on na liście.

3.8.2 Edycja typu studiów

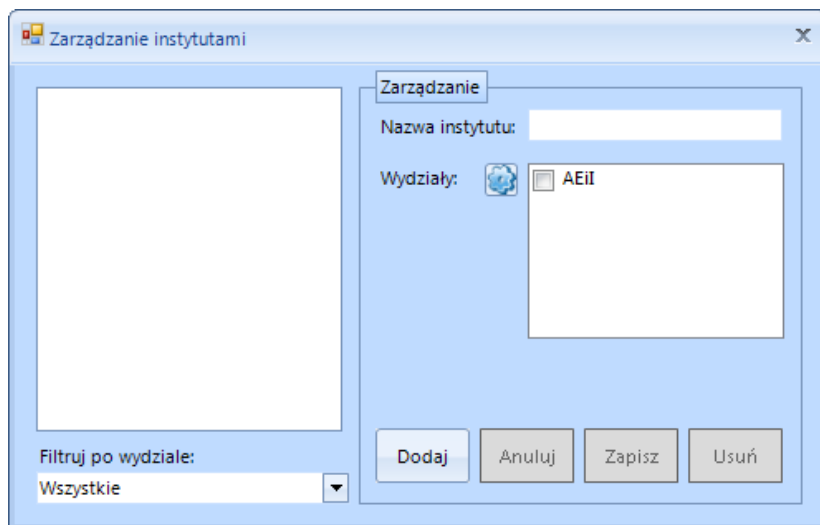
W celu edycji danego typu należy wybrać typ dostępny na liście, klikając w niego dwukrotnie. Następnie zmienić jego nazwę i zapisać zmiany klikając przycisk 'Zapisz'. W celu anulowania zmiany należy kliknąć przycisk 'Anuluj'.

3.8.3 Usuwanie typu studiów

Akcja usuwania wygląda podobnie do edycji z tym, że należy wybrać typ przedmiotu i kliknąć przycisk 'Usuń'. Użytkownik zostanie powiadomiony, jeżeli nie uda się usunąć danego typu.

3.9 ZARZĄDZANIE INSTYTUTAMI

W celu zarządzania instytutami należy kliknąć przycisk 'Instytuty', znajdujący się na pasku narzędziowym w głównym oknie aplikacji. Po jego kliknięciu pojawi się okno zarządzania instytutami (Rysunek 14).



Rysunek 14 Zarządzanie instytutami

3.9.1 Dodawanie instytutu

W celu dodania nowego instytutu należy wpisać jego nazwę w pole 'Nazwa instytutu', zaznaczyć odpowiedni wydział (jeżeli nie ma do wyboru żadnego wydziału należy go dodać – opis dodawania wydziału został przedstawiony w rozdziale 3.5), na

którym będzie dostępny instytut, a następnie kliknąć przycisk 'Dodaj'. Jeżeli użytkownik będzie chciał dodać instytut, który już istnieje na danym wydziale bądź nie poda jego nazwy, zostanie poinformowany o błędzie. Po dodaniu instytutu pojawi się on na liście.

Został tu zaimplementowany mechanizm filtracji, którego celem jest przedstawienie użytkownikowi wszystkich instytutów na danym wydziale, bądź wszystkich istniejących instytutów. W tym celu należy z listy rozwijanej 'Filtruj po wydziale' wybrać odpowiedni wydział. Filtracja następuje automatycznie po wyborze wartości z listy. Po skończeniu wyszukiwania wyniki zostaną przedstawione na liście.

3.9.2 Edycja instytutu

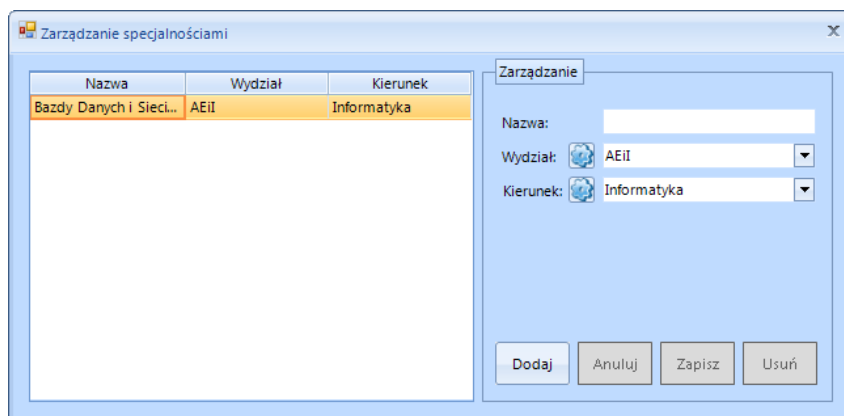
W celu edycji danego instytutu należy go wybrać z listy, klikając w niego dwukrotnie. Następnie zmienić jego nazwę lub wydział, na którym jest dostępny i zapisać zmiany klikając przycisk 'Zapisz'. W celu anulowania zmiany należy kliknąć przycisk 'Anuluj'.

3.9.3 Usuwanie instytutu

Akcja usuwania wygląda podobnie do edycji z tym, że należy wybrać instytut i kliknąć przycisk 'Usuń'. Użytkownik zostanie powiadomiony, jeżeli nie uda się usunąć danego instytutu.

3.10 ZARZĄDZANIE SPECJALIZACJAMI

W celu zarządzania specjalizacjami należy kliknąć przycisk 'Specjalizacje', znajdujący się na pasku narzędziowym w głównym oknie aplikacji. Po jego kliknięciu pojawi się okno zarządzania specjalizacjami (Rysunek 15).



Rysunek 15 Zarządzanie specjalizacjami

3.10.1 Dodawanie specjalizacji

W celu dodania nowej specjalizacji należy wpisać jej nazwę w pole 'Nazwa', zaznaczyć odpowiedni wydział (jeżeli nie ma do wyboru żadnego wydziału należy go dodać. Opis dodawania wydziału został przedstawiony w rozdziale 3.5), oraz kierunek (jeżeli nie ma do wyboru żadnego kierunku należy go dodać. Opis dodawania kierunku został przedstawiony w rozdziale 3.6), na którym będzie dostępna specjalizacja, a następnie kliknąć przycisk 'Dodaj'. Jeżeli użytkownik będzie chciał dodać specjalizację, która już istnieje na danym wydziale, na danym kierunku, bądź nie poda jego nazwy, zostanie poinformowany o błędzie. Po dodaniu specjalizacji pojawi się ona w tabeli.

3.10.2 Edycja specjalizacji

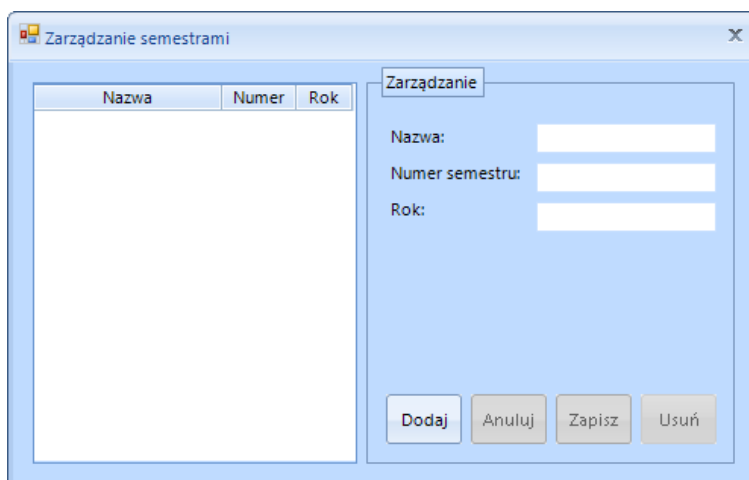
W celu edycji danej specjalizacji należy ją wybrać z tabeli, klikając w nią dwukrotnie. Następnie zmienić jej nazwę, wydział lub kierunek, na którym jest dostępna i zapisać zmiany klikając przycisk 'Zapisz'. W celu anulowania zmiany należy kliknąć przycisk 'Anuluj'.

3.10.3 Usuwanie specjalizacji

Akcja usuwania wygląda podobnie do edycji z tym, że należy wybrać specjalizację i kliknąć przycisk 'Usuń'. Użytkownik zostanie powiadomiony, jeżeli nie uda się usunąć danej specjalizacji.

3.11 ZARZĄDZANIE SEMESTRAMI

W celu zarządzania semestrami należy kliknąć przycisk 'Semestry', znajdujący się na pasku narzędziowym w głównym oknie aplikacji. Po jego kliknięciu pojawi się okno zarządzania specjalizacjami (Rysunek 16).



Rysunek 16 Zarządzanie semestrami

3.11.1 Dodawanie semestru

W celu dodania nowego semestru należy wpisać jego nazwę w pole 'Nazwa', wpisać numer semestru w pole 'Numer semestru', wpisać rok w pole 'Rok', a następnie kliknąć przycisk 'Dodaj'. Jeżeli użytkownik będzie chciał dodać semestr, który już istnieje, bądź nie poda jego nazwy, numeru semestru bądź roku zostanie poinformowany o błędzie. Po dodaniu semestru pojawi się on w tabeli.

3.11.2 Edycja semestru

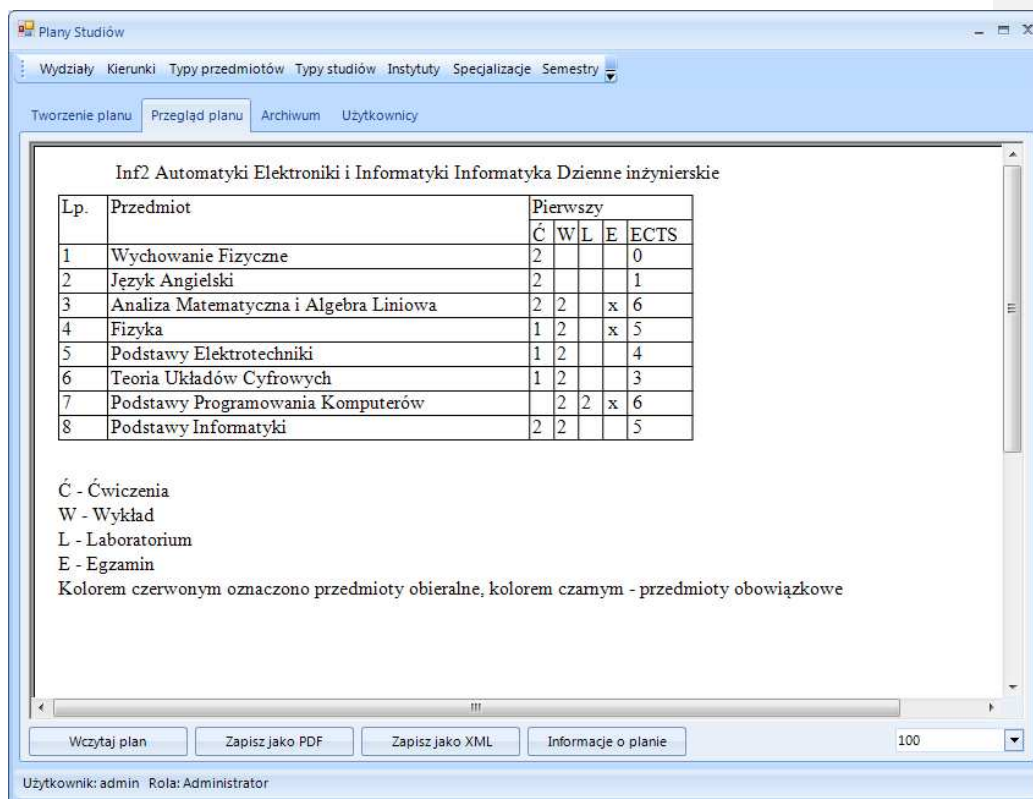
W celu edycji danego semestru należy go wybrać z tabeli, klikając w niego dwukrotnie. Następnie zmienić jego nazwę, numer semestru lub rok i zapisać zmiany klikając przycisk 'Zapisz'. W celu anulowania należy kliknąć przycisk 'Anuluj'.

3.11.3 Usuwanie semestru

Akcja usuwania wygląda podobnie do edycji z tym, że należy wybrać semestr i kliknąć przycisk 'Usuń'. Użytkownik zostanie powiadomiony, jeżeli nie uda się usunąć danego semestru.

3.12 PRZEGLĄD PLANU

Aplikacja umożliwia przeglądanie planu, oraz eksport do plików *.pdf oraz *.xml. W tym celu należy przejść na zakładkę 'Przegląd planu', znajdującą się w głównym oknie aplikacji (Rysunek 17).



Rysunek 17 Przegląd planu

3.12.1 Wczytywanie planu

Zakładka umożliwia wczytanie dowolnego planu stworzonego przez użytkownika. W tym celu użytkownik musi kliknąć przycisk 'Wczytaj plan', a następnie wybrać interesujący go plan z listy.

Komentarz [R13]: W jakim formacie ma być ten plan, bo chyba nie dowolnym? Co właściwie oznacza pojęcie wczytania planu (skąd jest wczytywany)

3.12.2 Zapis planu

Wczytany plan użytkownik może zapisać jako plik pdf albo xml. W tym celu użytkownik musi kliknąć odpowiednio przyciski 'Zapisz jako PDF' – aby zapisać plan jako plik pdf, lub 'Zapisz jako xml' – aby zapisać plan jako plik xml.

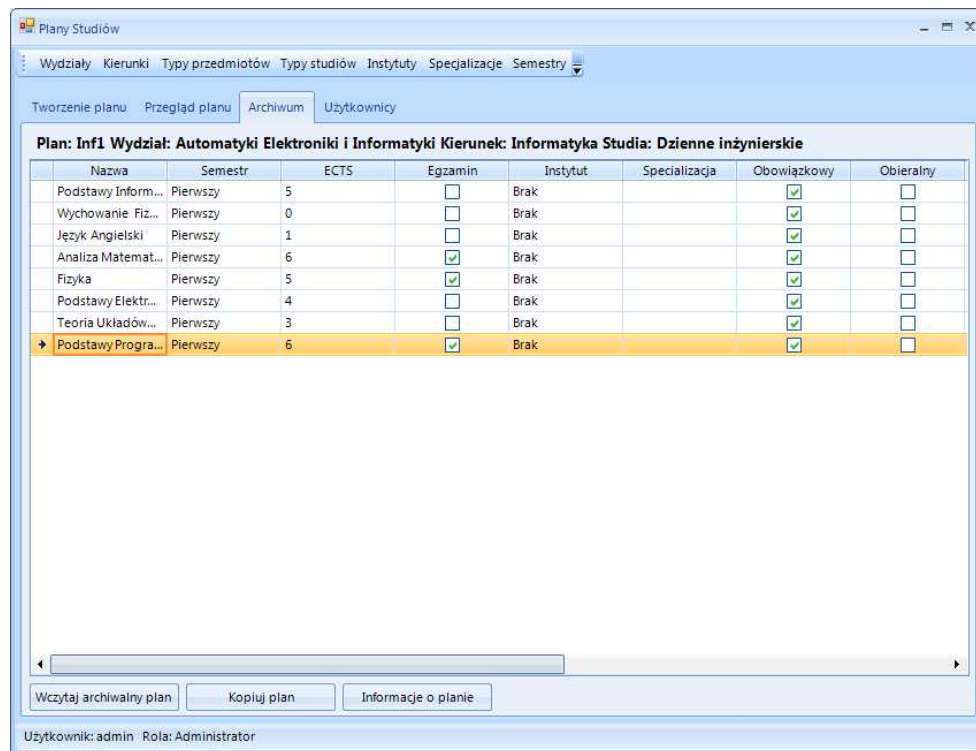
Komentarz [R14]: Gdzieś w pracy (niekoniecznie tu) musi pojawić się specyfikacja tego formatu planu (ufam że się gdzieś dalej pojawi)

3.12.3 Informacje o planie

Możliwe jest także wyświetlenie informacji o planie, należy w tym celu kliknąć przycisk 'Informacje o planie'. Po jego naciśnięciu pojawi się forma zawierająca informacje o planie.

3.13 ARCHIWUM

Aplikacja umożliwia przeglądanie zarchiwizowanego planu. W tym celu należy przejść na zakładkę 'Archiwum', znajdującą się w głównym oknie aplikacji (Rysunek 18).



Rysunek 18 Archiwum

3.13.1 Wczytywanie planu

Zakładka umożliwia wczytanie zarchiwizowanego planu. W tym celu użytkownik musi kliknąć przycisk 'Wczytaj archiwalny plan', a następnie wybrać interesujący go plan z listy.

3.13.2 Kopiowanie planu

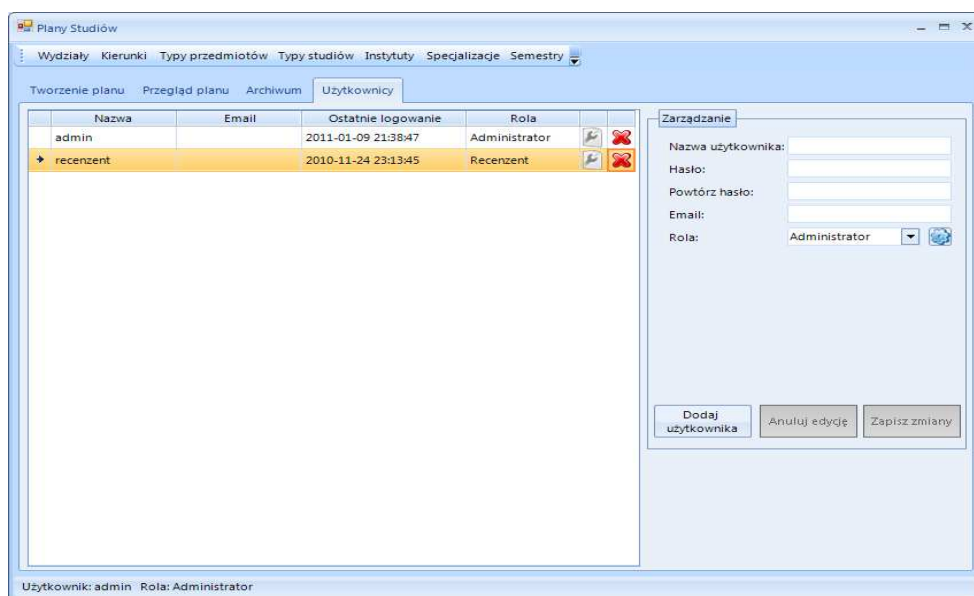
Możliwe jest kopiowanie planu zarchiwizowanego. W tym celu należy wczytać plan archiwalny, następnie wczytać lub utworzyć nowy plan na zakładce 'Tworzenie planu', a następnie kliknąć przycisk 'Kopiuj plan'.

3.13.3 Informacje o planie

Możliwe jest także wyświetlenie informacji o planie, należy w tym celu kliknąć przycisk 'Informacje o planie'. Po jego naciśnięciu pojawi się okno zawierające informacje o planie.

3.14 ZARZĄDZANIE UŻYTKOWNIKAMI

Aplikacja umożliwia zarządzanie użytkownikami. W tym celu należy przejść na zakładkę 'Użytkownicy', znajdującą się w głównym oknie aplikacji (Rysunek 19).



Rysunek 19 Użytkownicy

3.14.1 Dodawanie użytkownika

Możliwe jest dodanie nowego użytkownika, w tym celu należy wpisać nazwę użytkownika, podać hasło, e-mail oraz wybrać rolę dla tworzonego użytkownika. Jeżeli podane dane są nieprawidłowe, użytkownik zostanie o tym poinformowany. Po dodaniu nowego użytkownika pojawi się w tabeli.

3.14.2 Edycja użytkownika

W celu edycji użytkownika należy go wybrać z tabeli i kliknąć na przycisk klucza znajdujący się w tabeli, w wierszu użytkownika, którego chcemy edytować. Następnie można zmienić jego dane. W celu zatwierdzenia danych należy kliknąć przycisk 'Zapisz zmiany', w celu odrzucenia zmian przycisk 'Anuluj edycję'.

3.14.3 *Usuwanie użytkownika*

W celu usunięcia użytkownika należy go wybrać z tabeli i kliknąć na przycisk 'X' znajdujący się w tabeli, w wierszu użytkownika, którego chcemy edytować.

3.15 PRZEGLĄDANIE PLANU – WERSJA WEBOWA

Wersja webowa aplikacji pozwala na przeglądanie dowolnego planu (zarówno obowiązującego jak i zarchiwizowanego). Został zastosowany filtr, dzięki któremu wyszukiwanie konkretnego planu jest łatwiejsze.

3.15.1 *Filtr*

Filtr posiada następujące pola – 'Nazwa' (nazwa planu, wystarczy podać tylko część nazwy), 'Wydział' (wydział, po którym ma być szukany plan), 'Kierunek' (kierunek, po którym ma być szukany plan), 'Rok rozpoczęcia' (w celu wybrania tej opcji należy ją zaznaczyć; rok rozpoczęcia obowiązywania planu), 'Rok zakończenia' (w celu wybrania tej opcji należy ją zaznaczyć; rok zakończenia obowiązywania planu), 'Semestr początkowy' (w celu wybrania tej opcji należy ją zaznaczyć; semestr rozpoczęcia planu), 'Semestr końcowy' (w celu wybrania tej opcji należy ją zaznaczyć; semestr zakończenia planu), 'Plany' (wszystkie, archiwalne, obowiązujące).

Po stworzeniu filtru należy kliknąć przycisk 'Szukaj'. Po jego kliknięciu w liście rozwijanej 'Wybierz plan' pojawią się plany spełniające kryteria wyszukiwania.

3.15.2 *Przeglądanie planu*

W celu wyświetlenia planu należy wybrać jego nazwę z listy rozwijanej. Informacje o planie zostaną wyświetlone po prawej stronie w odpowiednich polach, a sam plan zostanie przedstawiony w postaci tabeli, znajdującej się pod obszarem 'Filtr'.

Przeglądanie planów:

Wybierz plan: Wybierz

Filtr

Nazwa:

Wydział: Wszystkie

Kierunek: Wszystkie

Rok rozpoczęcia: ☐ 1940

Rok zakończenia: ☐ 1940

Semestr początkowy: ☐

Semestr końcowy: ☐

Plany:

☒ Wszystkie

☐ Archiwalne

☐ Obowiązujące

Szukaj

Informacje o planie

Nazwa: Inf1
Kierunek: Informatyka
Wydział: Automatyki Elektroniki i Informatyki
Semestr początkowy: 1
Semestr końcowy: 7
Rok początkowy: 2007
Rok końcowy: 2011
Typ studiów: Dienne inżynierskie
Status: Zarchiwizowany

Nazwa	Semestr	ECTS	Egzamin	Instytut	Specjalizacja	Obowiązkowy	Obieralny	Wykład	Laboratorium	Ćwiczenia	Projekt	Seminarium	WF
Wychowanie Fizyczne	Pierwszy	0		Brak		Tak				2			
Język Angielski	Pierwszy	1		Brak		Tak				2			
Analiza Matematyczna i Algebra Liniowa	Pierwszy	6	Tak	Brak		Tak		2		2			
Fizyka	Pierwszy	5	Tak	Brak		Tak		2		1			
Podstawy Elektrotechniki	Pierwszy	4		Brak		Tak		2		1			

Rysunek 20 Przeglądanie planu - wersja webowa

4. SPECYFIKACJA WEWNĘTRZNA

W tym rozdziale zaprezentowana zostanie specyfikacja wewnętrzna aplikacji. W kolejnych podrozdziałach przedstawione zostaną schematy ideowe aplikacji desktopowej oraz webowej, schematy działania, diagramy przypadków użycia.

4.1 APLIKACJA DESKTOPOWA

4.1.1 Przedstawienie idei

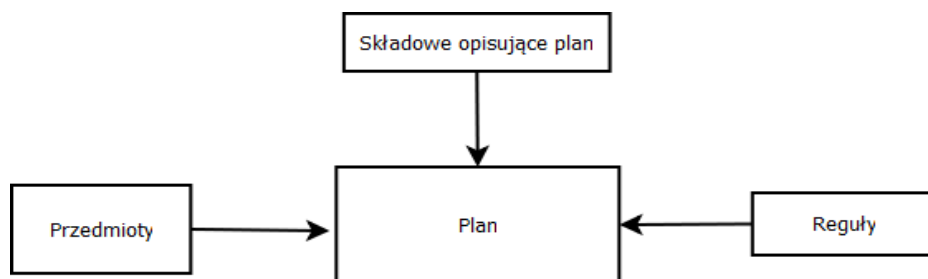
Główną ideą aplikacji jest tworzenie planu studiów. Plan studiów składa się z dwóch głównych elementów:

- składowe opisujące plan,
- przedmioty wchodzące w skład planu.

Składowe opisujące plan to: nazwa planu, semestr początkowy i końcowy, wydział, kierunek, typ studiów, rok rozpoczęcia i zakończenia obowiązywania planu, czy plan jest archiwalny oraz opinia.

Przedmiot także posiada składowe opisujące go. Są to: nazwa przedmiotu, semestr, punkty ECTS, występowanie egzaminu, instytut, obowiązkowość/obieralność przedmiotu, typy przedmiotu wraz z ilością godzin, specjalizacja.

Elementem opcjonalnym wchodzącym w skład planu są reguły, względem których plan będzie weryfikowany.



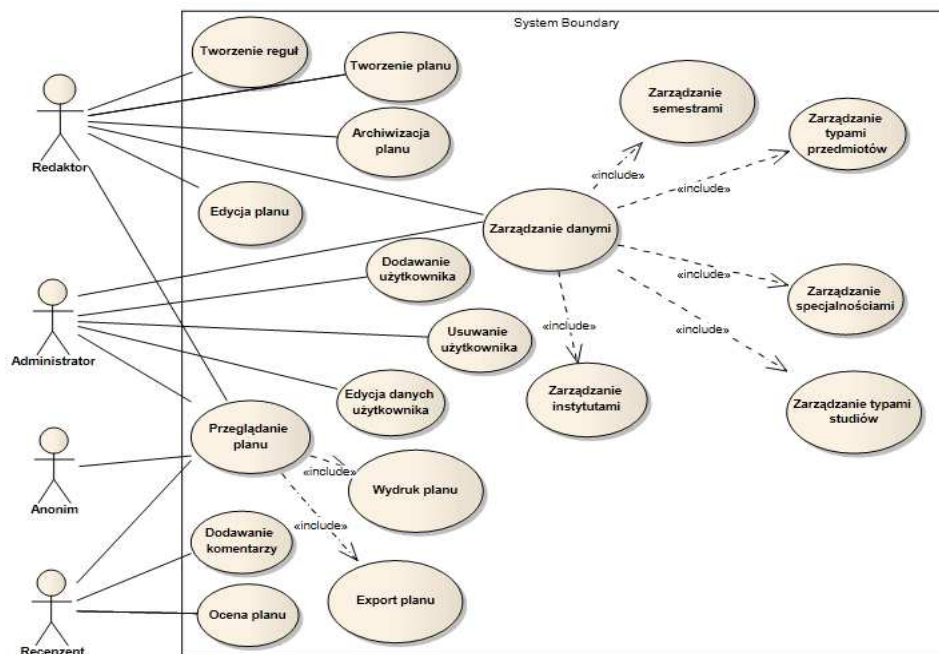
Rysunek 21 Elementy definiujące plan

Działanie aplikacji w głównej mierze opiera się o definiowanie poszczególnych elementów przedstawionych na powyższym rysunku (Rysunek 21). Wszystkie dane opisujące plan i przedmioty są przechowywane w bazie danych, której schemat zostanie przedstawiony w jednym z kolejnych podrozdziałów.

4.1.2 Diagram przypadków użycia

Na podstawie wcześniej opisanej analizy został stworzony diagram przypadków użycia dla aplikacji. Określeni zostali aktorzy, którymi są:

- Redaktor – tworzy plan.
- Recenzent – ma możliwość przeglądania planu i wystawiania opinii o nim.
- Administrator – posiada pełną kontrolę, może tworzyć plany, zarządzać elementami składowymi, tworzyć recenzje oraz zarządzać użytkownikami i rolami.
- Anonim – posiada jedynie możliwość przeglądania planów.



Rysunek 22 Diagram przypadków użycia

4.1.3 Konstrukcja aplikacji

Aplikacja została podzielona na trzy warstwy:

- warstwa danych,
- warstwa pośrednicząca,

- warstwa prezentacji.

Warstwa danych odpowiada za dane wykorzystywane w programie, które są przechowywane w zewnętrznej bazie. W tej warstwie odbywa się cała komunikacja z bazą danych oraz modyfikacja zawartych w niej rekordów.

Warstwa pośrednicząca ma na celu pośredniczenie w komunikacji pomiędzy warstwą prezentacji a warstwą danych. Odpowiada ona za walidację a później przekazywanie danych wprowadzonych przez użytkownika do bazy danych oraz przesyłanie danych żądanych przez użytkownika z warstwy danych do warstwy prezentacji.

Warstwa prezentacji służy do prezentowania danych użytkownikowi, pobierania od niego danych wejściowych i odpowiadania na wszystkie akcje przez niego wykonywane. Jest to graficzny interfejs użytkownika

Konstrukcję aplikacji przedstawiono na poniższym rysunku.



Rysunek 23 Konstrukcja aplikacji

4.1.4 Organizacja bazy danych

W bazie danych przechowywane są wszystkie dane dotyczące planu a także użytkowników. Strukturę bazy można podzielić na kilka części, mianowicie na:

- tabele przechowujące dane o składowych planów,
- tabele przechowujące dane o składowych przedmiotów,
- tabela łącząca przedmiot z planem,
- tabela z regułami,
- tabele przechowujące dane dotyczące użytkowników, praw i ról.

Dokładny schemat bazy danych został przedstawiony na rysunkach 22 i 23.

Do tabel przechowujących dane o składowych planu należą tabele:

- Departaments – przechowuje nazwy wydziałów,

- Institutes – dotyczy nazw instytutów,
- InstitutesDepartaments – łączy instytuty z wydziałami,
- Faculties – nazwy kierunków,
- FacultiesDepartaments – łączy kierunki z wydziałami,
- StudiesTypes – przechowuje typy studiów,
- Plans – tabela przechowująca zdefiniowane plany.

Tabele przechowujące dane o składowych przedmiotu to:

- Subjects – nazwy przedmiotów,
- SubjectTypes – nazwy typów przedmiotów ('Wykład', 'Ćwiczenia', itd.),
- SubjectsData – tabela przechowująca zdefiniowane przedmioty,
- SubjectTypesData – łączy przedmiot z typami przedmiotu i określa ilość godzin danego typu,
- Specializations – przechowuje nazwy specjalizacji i określa ich przynależność do wydziału i kierunku,
- SpecializationsData – określa czy przedmiot na specjalizacji jest obowiązkowy czy obieralny,
- Semesters – tabela przechowująca dane o semestrach.

Tabela łącząca przedmiot z planem:

- PlansData

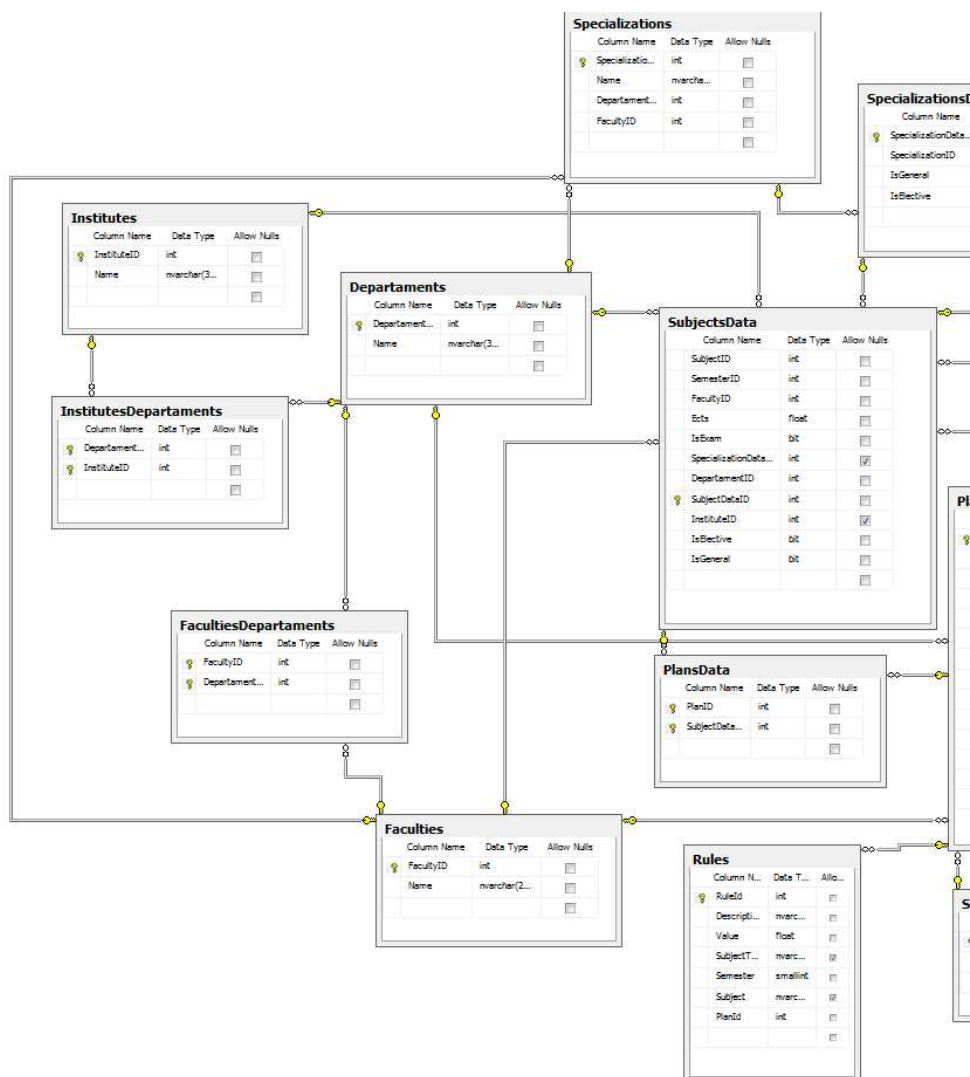
Tabela z regułami:

- Rules

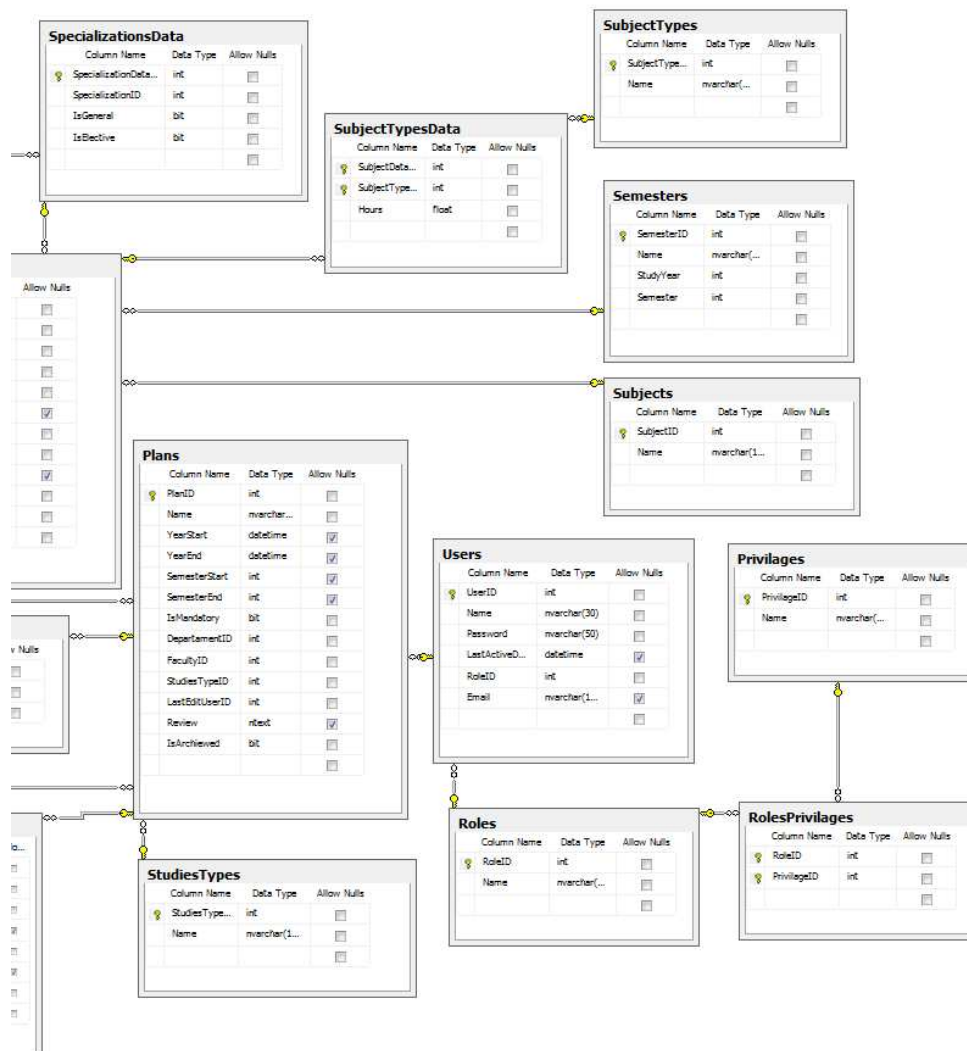
Tabele przechowujące dane dotyczące użytkowników, praw i ról:

- Users – przechowuje dane o użytkownikach,
- Privileges – przechowuje uprawnienia,
- Roles – zdefiniowane role użytkowników,
- RolesPrivileges – łączy role z uprawnieniami.

Komentarz [R15]: Czy i jeśli tak to jak jest to powiązane z systemem autoryzacji użytkowników i systemem uprawnień dostępnym w MS SQL Server ?



Rysunek 24 Schemat bazy danych cz. 1



Rysunek 25 Schemat bazy danych cz.2

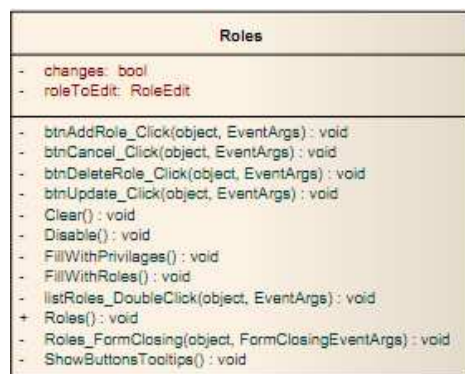
4.1.5 Przegląd ważniejszych klas i szczegóły implementacyjne

Aplikacja desktopowa składa się z dwóch projektów w **solucji**: projektu aplikacji, na który składają się warstwy prezentacji i pośrednicząca oraz projektu zawierającego warstwę danych (projekt ten jest wspólny dla aplikacji desktopowej i webowej).

Komentarz [R16]: Może inne określenie angielskiego solution

Warstwa prezentacji w aplikacji desktopowej to okienka użytkownika. Całość działa jak w standardowych aplikacjach okienkowych, co zostanie przedstawione na przykładzie klasy (okienka) *Roles*.

KLASA ROLES



Rysunek 26 Diagram UML klasy Roles

Opis poszczególnych pól i metod klasy:

- **changes** – pole oznaczające, czy użytkownik wprowadził jakieś zmiany w danych (edycja / dodanie / usunięcie danych). Przy zamknięciu okienka jest sprawdzana wartość tego pola i w zależności od wyniku jest zwracany odpowiedni rezultat do widoku wyżej, co spowoduje (lub nie) jego odświeżenie.
- **roleToEdit** – pole obiektu roli, która jest aktualnie w stanie edycji.
- **btn_AddRole_Click** – metoda obsługująca dodawanie nowej roli.
- **btnCancel_Click** – metoda anulująca tryb edycji.
- **btnDeleteRole_Click** – metoda obsługująca usunięcie roli.
- **btnUpdate_Click** – metoda obsługująca zapisanie zmian w edytowanej roli.
- **Clear** – metoda czyszcząca kontrolki po wykonaniu operacji.
- **Disable** – wyłącza przyciski w trybie edycji.
- **FillWithPrivileges** – metoda wypełniająca kontrolkę listą uprawnień z bazy danych
- **FillWithRoles** – wypełnia kontrolkę rolami pobranymi z bazy danych.
- **listRoles_DoubeClick** – metoda, która po podwójnym kliknięciu na roli przechodzi w tryb edycji tej roli.
- **Roles()** – publiczny konstruktor.

- Roles_FormClosing – metoda badająca pole changes przy zamykaniu okienka i ustawiająca odpowiedni rezultat.
- ShowButtonsTooltips – metoda, która ustawia dymki podpowiedzi dla przycisków (przyciski firmy Telerik nie mają atrybutu Tooltip, dlatego podpowiedzi trzeba ustawiać w kodzie).

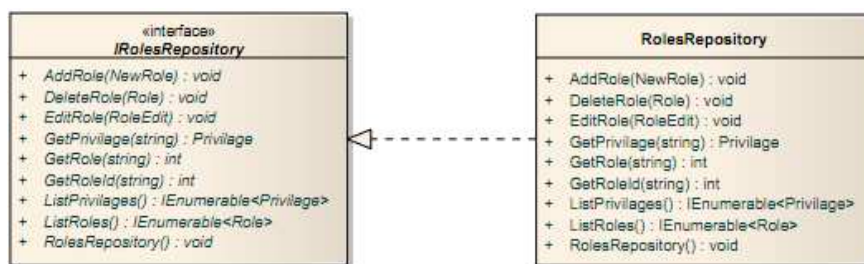
Klasa Roles odwołuje się do klasy RoleController w warstwie pośredniej, która odpowiada na wykonywane operacje przez użytkownika – sprawdza poprawność wprowadzonych przez niego danych i w przypadku danych poprawnych wykonuje żadaną operację, natomiast w przeciwnym razie zwraca błędy.



Rysunek 27 Diagram UML klasy RoleController

Metody klasy RoleController nie będą tutaj omawiane, gdyż ich nazwy dość jasno określają, do czego dana metoda służy. Jedynie należy tutaj wyjaśnić, iż wykorzystany został tutaj wzorzec Singleton, który pozwala na wystąpienie tylko jednego obiektu tej klasy w obrębie całego systemu. W momencie, gdy nastąpi odwołanie do pola statycznego Instance nastąpi sprawdzenie czy pole zostało zainicjowane, a jeżeli nie to zostanie stworzony po raz pierwszy obiekt. Natomiast, jeżeli obiekt tej klasy już istnieje w systemie to zostanie on zwrócony bez tworzenia nowego obiektu.

Jednym z pól klasy RoleController jest prywatne pole repository typu IRolesRepository. IRolesRepository to interfejs znajdujący się w warstwie danych. Klasa implementująca ten interfejs to RolesRepository, która odpowiada za operacje wykonywane na bazie danych. W tej klasie nie jest sprawdzana poprawność danych wprowadzanych przez użytkownika, gdyż tym zajmuje się klasa RoleController.



Rysunek 28 Diagram UML interfejsu IRolesRepository wraz z implementującą go klasą RolesRepository

Nazwy metod określają działanie danej metody, więc nie będą tutaj szczegółowo opisywane. Przedstawiona zostanie jedynie przykładowa metoda w celu zaprezentowania wykorzystania technologii LINQ.

```

public int GetRoleId(string rolename)
{
    return (from Role r in SPDatabase.DB.Roles
            where string.Compare(r.Name, rolename, true) == 0
            select r.RoleID).FirstOrDefault();
}
  
```

Rysunek 29 Wykorzystanie LINQ do operacji na bazie danych

Powyższa metoda ma zadanie pobrać numer id roli o zadanej nazwie. Metoda ta wybiera obiekt Role z tabeli Roles (SPDatabase.DB – singleton, zostanie omówiony później) i porównuje jego nazwę zadaną. Jeśli nazwy się zgadzają, obiekt jest zwracany, jeżeli nie – brany jest kolejny obiekt z bazy. Jeżeli nie znaleziono pasującego obiektu to zwracana jest wartość null.

W aplikacji całość zarządzania danymi słownikowymi odbywa się w przedstawiony powyżej sposób. Zarządzanie przedmiotami, wydziałami, kierunkami, planami itd. różni się tylko nieco odmienną implementacją z racji rodzaju danych, ale idea pozostaje ta sama.

POŁĄCZENIE Z BAZĄ DANYCH

Wszystkie operacje na bazie danych są wykonywane w ramach pewnego kontekstu. Kontekst ten jest tworzony z wykorzystaniem wzorca Singleton. W warstwie pośredniej stworzona została klasa częściowa SPDatabase, którą część Visual Studio wygenerował automatycznie (mapowanie tabel bazy na obiekty klas), a część – połączenie z bazą – została napisana ręcznie. Część klasy odpowiadająca za połączenie wygląda następująco:

```

public partial class SPDatabase
{
    private static SPDatabase db;
    public static SPDatabase DB
    {
        get
        {
            if (db == null)
                db = new SPDatabase(ConnectionString);

            return db;
        }

        set
        {
            db = value;
        }
    }

    private static string ConnectionString
    {
        get
        {
            return Helpers.Connection.GetDatabaseConnectionString();
        }
    }
}

```

Rysunek 30 Kod odpowiadający za połączenie z bazą danych

Jeżeli obiekt klasy SPDatabase nie istnieje jeszcze w systemie to jest tworzony z wykorzystaniem odpowiednich parametrów połączenia zapisanych w pliku konfiguracyjnym aplikacji (*.config); odczytem parametrów połączenia z tego pliku zajmuje się klasa Connections z przestrzeni Helpers).

4.2 APLIKACJA WEBOWA

4.2.1 Przedstawienie idei

Celem aplikacji webowej było udostępnianie już stworzonych planów anonimowym użytkownikom. Całość miała działać na serwerze WWW, co pozwalałoby każdemu użytkownikowi na przeglądanie planów bez instalowania jakiegokolwiek zewnętrznego oprogramowania poza przeglądarką internetową. Część webowa korzysta z tej samej bazy danych, co część desktopowa, co pozwala na natychmiastowe podglądanie zmian w planie wprowadzonych w aplikacji desktopowej za pomocą serwisu WWW.

4.2.2 Konstrukcja aplikacji

W aplikacji webowej wykorzystany został wzorzec MVC. Modelem w tym przypadku są klasy mapowane na obiekty baz danych oraz metody bezpośrednio operujące na bazie. Są to dokładnie te same klasy, które zostały wykorzystane w aplikacji desktopowej do przeglądania planu (klasy warstwy danych aplikacji desktopowej).

Kontroler przetwarza zapytanie otrzymane od użytkownika i zwraca odpowiednio zmodyfikowaną na podstawie otrzymanych danych część widoku.

Widokiem w tym przypadku jest strona serwisu WWW. Użytkownik wykonuje jakąś akcję, np. filtruje plany, widok przesyła zapytanie do kontrolera i w odpowiedzi otrzymuje zmodyfikowaną część widoku do wyświetlenia.

4.2.3 Przegląd ważniejszych klas i szczegóły implementacyjne

Jako widoki wykorzystywane są pliki *.aspx, które zawierają zwykły kod HTML oraz kod napisany w ASP.NET. Dodatkowo zostały wykorzystane widoki częściowe (PartialView) do wyświetlania tabeli z przedmiotami z planem - List.ascx, oraz informacji o planie – PlanInfo.ascx. Głównym i najważniejszym widokiem jest plik Plan.aspx. W nim wyświetlany jest filtr planów, sam plan i informacje o nim.

Obsługa kliknięcia na przycisk ‘Szukaj’ oraz wybrania planu do wczytania z rozwijanej listy jest wykonywana za pomocą skryptu JavaScript. Opisane to zostanie na przykładzie wybrania planu do wyświetlenia.

Po wybraniu dostępnego planu z listy rozwijanej pobierany jest jego identyfikator. Następuje przekierowanie do strony o adresie /Plans/Plan?PlanId=id. W tym momencie do konstruktora kontrolera (klasa PlanController) jest przekazywane id planu. Kontroler tworzy obiekt typu PlanList, w którym jednym z pól jest pole typu Plan. Do tego pola wczytywane są informacje o planie z bazy danych. Następnie do widoku zwracany jest stworzony obiekt PlanList. Na jego podstawie, widoki cząstkowe wyświetlają tabelę z przedmiotami oraz informacje o planie. Działanie filtra przebiega dokładnie tak samo, przy czym do kontrolera przesyłana jest większa liczba argumentów.

Poniżej przedstawiono konstruktor klasy PlanController.

```

public ActionResult Plan(int planId = 0, string name = "", int departamentId = 0, int facultyId = 0,
    string selectedPlan = "", int yearStart = 0, int yearEnd = 0, string semesterStart = "", string semesterEnd = "")
{
    PlanList data;
    filter.Name = name;
    if (departamentId != 0)
        filter.DepartmentName = _departmentsRepository.GetDepartament(departamentId).Name;
    if (facultyId != 0)
        filter.FacultyName = _facultiesRepository.GetFaculty(facultyId).Name;
    if (selectedPlan != null && !selectedPlan.Equals(""))
    {
        if (selectedPlan.Equals("all"))
            filter.All = true;
        else if (selectedPlan.Equals("arch"))
            filter.IsArchived = true;
        else if (selectedPlan.Equals("curr"))
            filter.IsMandatory = true;
    }

    if (selectedPlan != null && selectedPlan.Equals("") && planId == 0)
        filter.All = true;

    if (yearStart != 0)
        filter.YearStart = yearStart;
    if (yearEnd != 0)
        filter.YearEnd = yearEnd;

    int parsed;
    int.TryParse(semesterStart, out parsed);
    if (parsed != 0)
        filter.SemesterStart = parsed;
    int parsed2;
    int.TryParse(semesterEnd, out parsed2);
    if (parsed2 != 0)
        filter.SemesterEnd = parsed2;

    data = new PlanList(this.plansRepository.ListPlans(filter));

    data.PlanID = planId;
    List<SelectListItem> plans = new List<SelectListItem>(data.Plans);
    List<SelectListItem> faculties = new List<SelectListItem>(data.Faculties);
    data.FacultyID = int.Parse(faculties[0].Value);
    List<SelectListItem> departaments = new List<SelectListItem>(data.Departaments);
    data.DepartamentID = int.Parse(departaments[0].Value);
    List<SelectListItem> years = new List<SelectListItem>(data.Years);
    data.YearStartID = int.Parse(years[0].Value);
    data.YearEndID = data.YearStartID;
    data.SelectedPlan = this.plansRepository.GetPlan(data.PlanID);
    if (data.PlanID == 0)
    {
        if (plans.Count > 0)
        {
            data.PlanID = int.Parse(plans[0].Value);
            data.SelectedPlan = this.plansRepository.GetPlan(data.PlanID);
        }
        else
            return View(data);
    }
    return View(data);
}

```

Rysunek 31 Konstruktor kontrolera PlanController

5. TESTOWANIE

Aplikacja desktopowa została przetestowana testami funkcjonalnymi, testami strukturalnymi, oraz testami jednostkowymi z wykorzystaniem NUnita.

Testy funkcjonalne to testy, podczas których sprawdzana jest zgodność aplikacji z założeniami opisanymi w dokumentacji. Dane wejściowe w testach funkcjonalnych nie są opierane na podstawie budowy programu.

Testy strukturalne to testy, w których podawane są oczekiwane (poprawne) dane wejściowe, w celu sprawdzenia czy program (jego funkcje) działają zgodnie z oczekiwaniami.

Testy jednostkowe to testy, w których sprawdzane jest działanie pojedynczych metod w klasie. Na potrzeby testów jednostkowych tworzy się klasę testową, a następnie w niej metody testowe, w których sprawdzane jest działanie metod danej klasy. W tej aplikacji został wykorzystany framework NUnit, który bardzo dobrze wspiera pisanie testów jednostkowych. Dzięki testom jednostkowym możliwe jest szybkie wykrycie oraz usunięcie błędu. Ponadto testy jednostkowe są w pełni zautomatyzowane, można je uruchomić w każdej chwili i natychmiast otrzymać wyniki.

5.1 TESTY JEDNOSTKOWE

```
[Test]
public void ShouldCreateErrorMessageForNullDepartment()
{
    const string errorMessage = "Wybierz przynajmniej jeden wydział";

    _newFaculty.Departaments = null;
    bool result = _newFaculty.IsValid;

    Assert.IsNotNull(_newFaculty);
    Assert.IsFalse(result);
    Assert.IsTrue(_newFaculty.Errors.Count > 0);
    Assert.IsTrue(_newFaculty.Errors.Contains(errorMessage));
}
```

Komentarz [R17]: lepiej najpierw tekst a potem fragment kodu

Rysunek 32 Przykładowy test jednostkowy

Powyższy test sprawdza poprawność tworzenia obiektu typu *NewFaculty*, którego pole *Departaments* ma zaimplementowaną walidację – utworzenie nowego błędu, jeżeli pole *Departaments* będzie **nullem**.

Komentarz [R18]: mieć wartość null

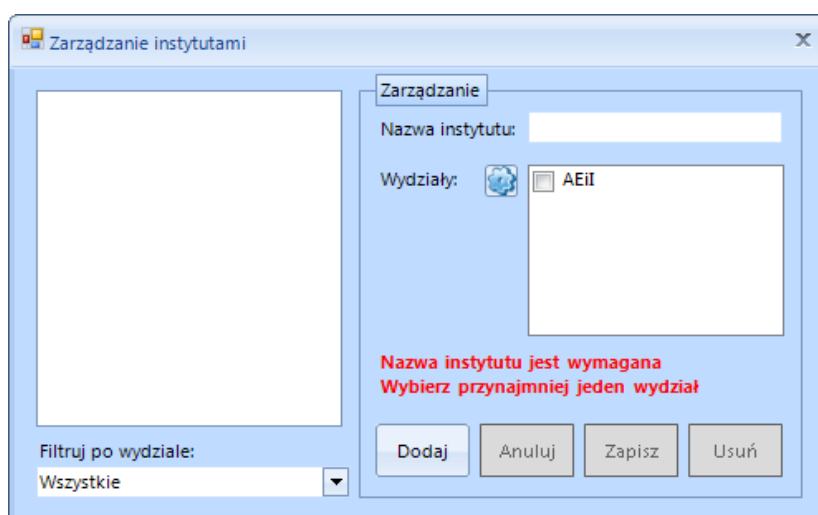
Test zaczyna się od ustawienia pola *Departaments* dla obiektu *_newFaculty*, który jest obiektem typu *NewFaculty*, jako null. Następnie wartość właściwości *IsValid*

(właściwość *IsValid* wywołuje metodę sprawdzającą warunki tworzenia obiektu *NewFaculty*) obiektu *_newFaculty*, jest przypisywana do zmiennej lokalnej. Następnie następuje sprawdzenie poprawności działania metody wywoływanej przez *IsValid* - sprawdzane jest:

- poprawne utworzenie obiektu *_newFaculty*,
- wartość właściwości *IsValid*
- czy został utworzony nowy błąd
- czy utworzony błąd zawiera poprawny komunikat

5.2 TESTY FUNKCJONALNE

Testy funkcjonalne zostały przeprowadzone manualnie, poprzez wprowadzanie różnego typu danych i sprawdzaniu czy funkcjonalność programu pokrywa się z założeniami.



Rysunek 33 Sprawdzenie funkcjonalności programu – walidacja danych

W powyższym teście sprawdzono, czy opisany w dokumentacji mechanizm walidacji danych podczas tworzenia nowego obiektu działa poprawnie. Podczas tworzenia nowego instytutu nie została podana jego nazwa, ani nie został wybrany żaden wydział, mimo to nastąpiła próba dodania takiego instytutu. W wyniku zastosowania mechanizmu walidacji, taki obiekt nie został stworzony i zostały wyświetlone komunikaty o błędach.

5.3 TESTY STRUKTURALNE

Przeprowadzenie tych testów polegało na sprawdzeniu, czy po wprowadzeniu poprawnych danych do programu, otrzymamy pożądane wyniki. Zostały w ten sposób przetestowane wszystkie funkcje programu.

5.4 PODSUMOWANIE TESTÓW

Zastosowanie licznych testów (jednostkowych, funkcjonalnych, strukturalnych) umożliwiło wykrycie krytycznych błędów w programie, które zostały szybko usunięte. Umożliwiły one także sprawdzenie, czy funkcjonalność opisana w dokumentacji projektu, pokrywa się z funkcjonalnością gotowej aplikacji.

Błędy, jakie znaleziono dzięki procesowi testowania dotyczyły przede wszystkim walidacji danych oraz braku inicjalizacji obiektów przed wykorzystaniem (błędy typu `NullReferenceException`).

6. UWAGI KOŃCOWE

Wszystkie postawione cele zostały zrealizowane. Zostały stworzone dwie aplikacje (desktopowa i webowa). Zarówno aplikacja desktopowa jak i webowa, która względem aplikacji desktopowej jest mocno ograniczona, spełnia wszystkie założenia funkcjonalne, które zostały ustalone.

W kolejnych wersjach można rozwinąć aplikację webową, aby funkcjonalnością nie odbiegała od aplikacji desktopowej.

Dodatkową funkcjonalnością, jaką można dodać w przyszłości może być informowanie użytkownika drogą poczty elektronicznej o wystawieniu recenzji dla edytowanego przez tego użytkownika planu. Również informacje o niespełnieniu reguł przez plan mogą być wysyłane pocztą elektroniczną

7. BIBLIOGRAFIA

1. B.Evjen, S.Hanselman, D.Rader, ASP .NET 3.5 z wykorzystaniem C# i VB, Helion, Gliwice 2010.
2. Troelsen A., Język C# 2008 i platforma .NET 3.5, PWN, Warszawa 2009.
3. LINQ To entities <http://msdn.microsoft.com/en-us/library/bb386964.aspx>
4. .NET Framework 4 <http://msdn.microsoft.com/en-us/library/w0x726c2.aspx>
5. NUnit <http://www.nunit.org/>
6. Reshaper: The Most Intelligent Extension for Visual Studio
<http://www.jetbrains.com/resharper/index.html>
7. Google Code www.code.google.com

SPIS RYSUNKÓW

Rysunek 1 Okno informujące o wersji testowej kontrolek.....	15
Rysunek 2 Forma logowania	17
Rysunek 3 Główne okno aplikacji.....	18
Rysunek 4 Tworzenie nowego planu	19
Rysunek 5 Wczytywanie planu	20
Rysunek 6 Dodawanie przedmiotu do planu	21
Rysunek 7 Edycja przedmiotu.....	22
Rysunek 8 Dodawanie reguł.....	23
Rysunek 9 Weryfikacja planu.....	24
Rysunek 10 Zarządzanie wydziałami	24
Rysunek 11 Zarządzanie kierunkami	25
Rysunek 12 Zarządzanie typami przedmiotów	26
Rysunek 13 Zarządzanie typami studiów	27
Rysunek 14 Zarządzanie instytutami.....	28
Rysunek 15 Zarządzanie specjalizacjami	29
Rysunek 16 Zarządzanie semestrami	31
Rysunek 17 Przegląd planu	32
Rysunek 18 Archiwum	33
Rysunek 19 Użytkownicy.....	34
Rysunek 20 Przeglądanie planu - wersja webowa.....	36
Rysunek 21 Elementy definiujące plan	37
Rysunek 22 Diagram przypadków użycia.....	38
Rysunek 23 Konstrukcja aplikacji.....	39

Rysunek 24 Schemat bazy danych cz. 1	41
Rysunek 25 Schemat bazy danych cz.2.....	42
Rysunek 26 Diagram UML klasy Roles.....	43
Rysunek 27 Diagram UML klasy RoleController.....	44
Rysunek 28 Diagram UML interfejsu IRolesRepository wraz z implementującą go klasą RolesRepository	45
Rysunek 29 Wykorzystanie LINQ do operacji na bazie danych.....	45
Rysunek 30 Kod odpowiadający za połączenie z bazą danych.....	46
Rysunek 31Konstruktor kontrolera PlanController	48
Rysunek 32 Przykładowy test jednostkowy	49
Rysunek 33 Sprawdzenie funkcjonalności programu – walidacja danych	50