Grzegorz Cylny, Bartłomiej Iwicki, Paweł Garbarz, Jacek Adamus

Baza Studentów

Specyfikacja techniczna projektu

1. Omówienie. Metoda SPIN.
2. Wymagania funkcjonalne i niefunkcjonalne.
3. Priorytetów MOSCOW.
4. Wymagania sprzętowe.
5. Struktura logiczna systemu.
6. Diagramy klas.
7. Projekt systemu bazodanowego.
8. Przykład interfejsu.
9. Plan Testów.
10. Szacowanie kosztów wdrożenia.
11. Pielęgnacja.

Załączniki:

1. Kod źródłowy.
2. Gotowe aplikacje wchodzące w skład systemu.
3. Schemat bazy danych w formacie pdf.

Ad1. Omówienie.

Situation – zbiorcze zestawienie podstawowych informacji

System tworzony jest z myślą o obsłudze dziekanatu, jednakowoż w dalszej perspektywie, po wykonaniu modyfikacji, może być używany w innych instytucjach. Idea systemu polega na skomputeryzowaniu prac biurowych, stąd możliwość jego szerokiego zastosowania.

Problem – omówienie problemu

Podstawowym problemem w pracach biurowych w instytucjach, bez odpowiedniego poziomu skomputeryzowania, jest długi czas obiegu papierowego. Aby odnaleźć informacje, musi być ona wyszukiwana w archiwum, następnie przeniesiona na stanowisko robocze, z którego po skończeniu prac musi być odniesiona na miejsce. Jest to kłopotliwe i czasochłonne. Ponadto brak spójności wprowadza możliwość generowania błędów, na przykład na różnych poziomach biurokratycznych informacja musi być wprowadzana wielokrotnie, co musi skończyć się przekłamaniem. Ponadto informacja nie jest dostępna dla wszystkich jednocześnie. Opóźnia to prace, opóźnienia powodują straty.

Implied Need – priorytet

Priorytetem aplikacji jest przeniesienie głównego obiegu informacji z warstwy papierowej na elektroniczną. Zwiększenie możliwości przepływu informacji. Skrócenie czasu dostępu do informacji. Osoby, w zależności od otrzymanych uprawnień, powinny mieć możliwość intuicyjnej, możliwie nie awaryjnej pracy.

Chcemy by nasza aplikacja odróżniała się od innych dostępnych na rynku swoją kompaktowością i mobilnością, tj. maksymalnie małym rozmiarem, przy zachowaniu wymaganej funkcjonalności.

Needs - potrzeby systemu

Przy tworzeniu wzorujemy się na najlepszych możliwych rozwiązaniach w tej dziedzinie. Moduły aplikacyjne będą tworzone w nowoczesnym języku C#. Korzystać będą z dynamicznych bibliotek dll. Baza danych natomiast oparta jest o technologię MySql. Nie wyklucza to jednak używania w przyszłości technologii komercyjnych jak Oracle, czy Microsoft SQL Server.

Ad2. Wymagania funkcjonalne i niefunkcjonalne.

Poniżej znajduje się wyspecyfikowana lista elementów funkcjonalnych i niefunkcjonalnych budowanego systemu.

a) Funkcjonalne

- Wyświetlanie listy studentów, wydziałów, przedmiotów, kierunków, roku studiów w dowolnej konfiguracji.
- Wyszukiwanie z możliwością sortowania wg różnych kryteriów wyboru.
- Panel dodawania, edycji, usuwania danych.
- Modularyzacja rozproszonego systemu z centralą bazodanową, pozwala na użycie multiplikowanego, dynamicznego mirroringu oraz metod zapewniania bezpieczeństwa i stabilności, jak rozbicie geolokalizacyjne.
- Możliwość generowania dokumentów i raportów w postaci zarówno elektronicznej jak i papierowej.
- Działanie w standardzie MRP I.
- Zarządzanie sferą edukacyjną, ale również biznesową.
- Wprowadzenie praw dostępu; pełna spójność i funkcjonalność systemu.
- Mechanizmy programowe do personalizacji interfejsu.

- Przejrzysty, intuicyjny, czytelny zarówno dla użytkownika, jak i późniejszych administratorów.
- System reagujący na możliwe sytuacje wyjątkowe, jak zerwane połączenie, wprowadzenie nieprawidłowych danych.

a) Niefunkcjonalne

- Bezpieczeństwo.
- Lepsze wykorzystanie posiadanej infrastruktury.
- Szybsze reagowanie na zmiany zachodzące w otoczeniu.
- Kontrola etapowa działania instytucji.
- Długofalowe obniżenie kosztów utrzymania środowiska. Microsoft wycenia serwery baz danych wg liczby gniazd procesorowych serwera, nie zaś liczby rdzeni i ich wydajności.

Ad3. Lista Priorytetów MOSCOW.

Must have – niezbędne składowe zawierające podstawowe funkcje/usługi

1. Wprowadzenie zróżnicowanych uprawnień.
2. Wyświetlanie szukanych informacji, zgodnie z uprawnieniami.
3. Tworzenie studenta z wszelkimi wymaganymi danymi.
4. Edycja danych studenta.
5. Usuwanie informacji zbędnych.
6. Zapewnienie właściwego poziomu bezpieczeństwa.

Should have – elementy, które powinny się znaleźć w systemie, ale nie są jego najistotniejszą składową

1. Eksport wybranych informacji do formatów drukowalnych.
2. Podpowiedzi dla użytkowników.

Could have – te elementy które mogą stanowić rozszerzenie funkcjonalne systemu, rozwiązania "przyszłościowe"

1. Stworzenie alternatywnych modułów Webowych.
2. Wykorzystanie alternatywnych systemów bazodanowych.

Won't have – elementy Systemu Informatycznego które nie będą przez nas realizowane

1. Synchronizacja z systemem całej uczelni.
2. Pobieranie informacji z zewnętrznych serwerów.

Ad4. Wymagania sprzętowe.

1. Server z obsługa ASP .NET , przynajmniej 3.0 , a także MySql 5.0.
2. Klienckie komputery PC z .NET przynajmniej 3.0 . RAM 64MB, Procesor 1Ghz, karta graficzna w standardzie SVGA. Dysk twardy, przynajmniej 10MB wolnego miejsca. Komunikacja za pomocą klawiatury i myszy.

Ocena architektury systemu, w kontekście sprzętu.

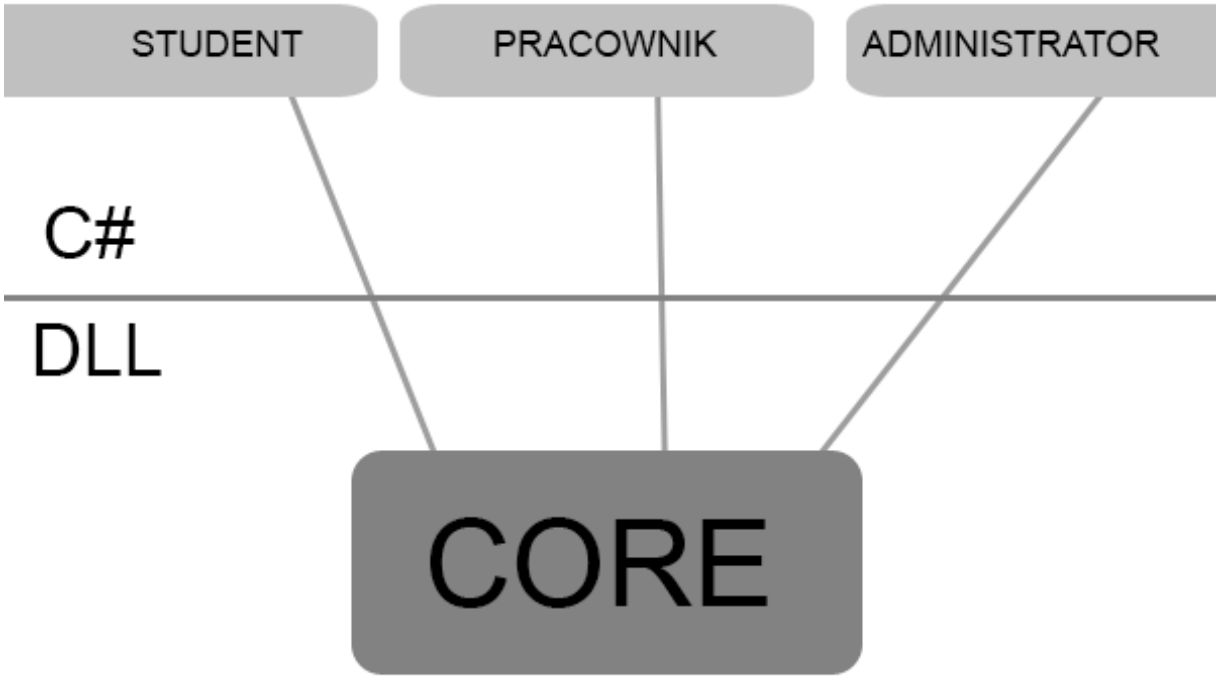
1. Niewielkie wymagania sprzętowe.
2. Dzięki językowi C# uzyskujemy małe, wydajne aplikacje.
3. Zewnętrzna baza danych nie obciąża klienckich maszyn.
4. Możliwość przenoszenia systemu na inne platformy.

Ocena architektury systemu, w kontekście wykorzystywanych technologii.

1. Możliwość użycia przy tworzeniu i rozwoju systemu darmowego środowiska do użytku komercyjnego - Microsoft Visual Studio Express 2008.
2. Centrala bazodanowa napisana w darmowej technologii MySql.

Ad5. Struktura logiczna systemu.

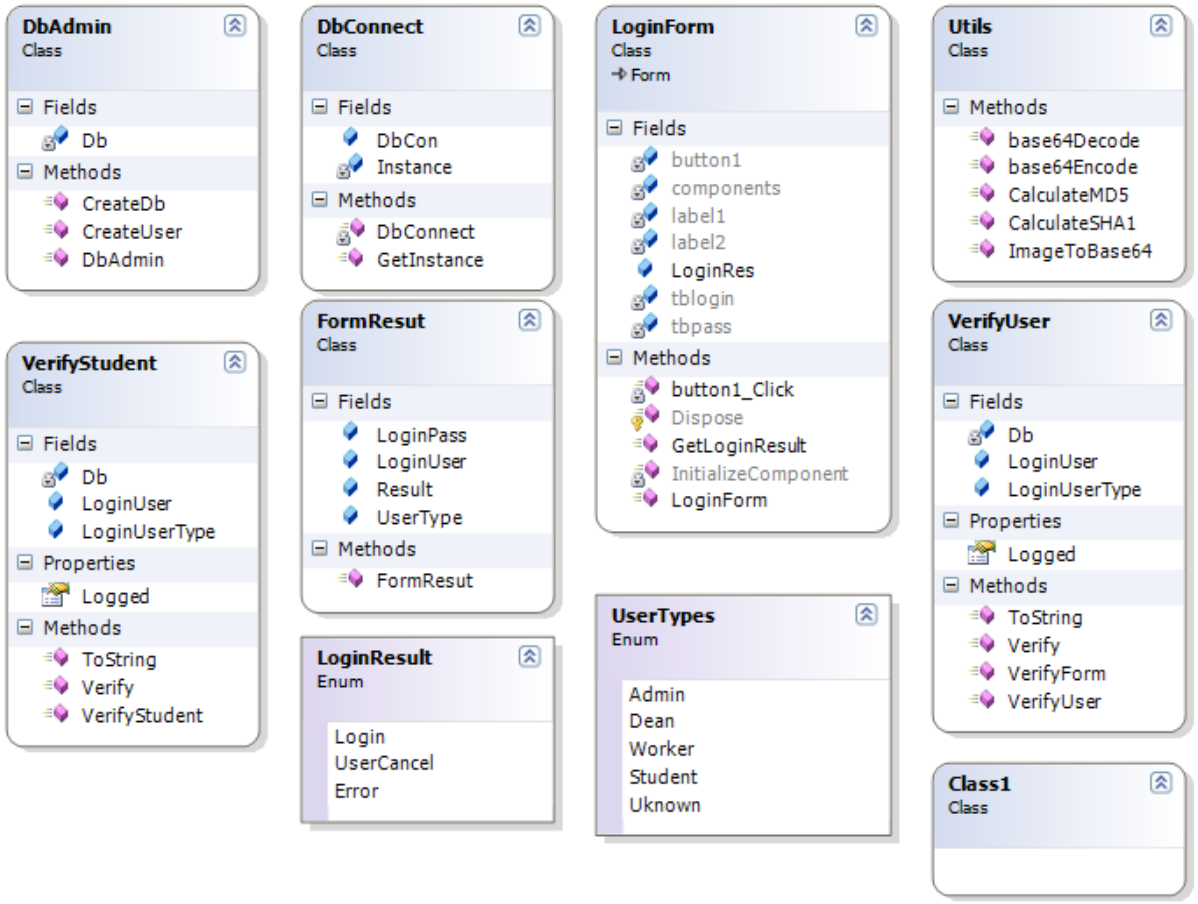
Nasz system opiera się na profesjonalnym rozwiązaniu, które polega na zaprojektowaniu centralnego rdzenia, wykorzystywanego w każdej aplikacji modułowej, na zasadzie biblioteki DLL. Jego konstrukcja i format pozwala na łatwe modyfikowanie bądź dodawanie nowych funkcjonalności. Moduły zaś ściśle współpracują z możliwościami, które udostępnia im rdzeń, w zależności od poziomu dostępu.



Ad6. Diagramy klas.

Poniżej zamieszczamy diagramy przedstawiające klasy występujące w naszym systemie, innymi słowy, jest to przedstawienie całej konstrukcji i wszystkich powiązań.

a) Rdzeń (*Core*)



W klasie DbAdmin znajdziemy metody do tworzenia bazy danych oraz użytkowników bazy.

Klasa DbConnect odpowiedzialna jest za łączenie się z serwerem bazodanowym. To w niej jest ustalony adres serwera bazodanowego.

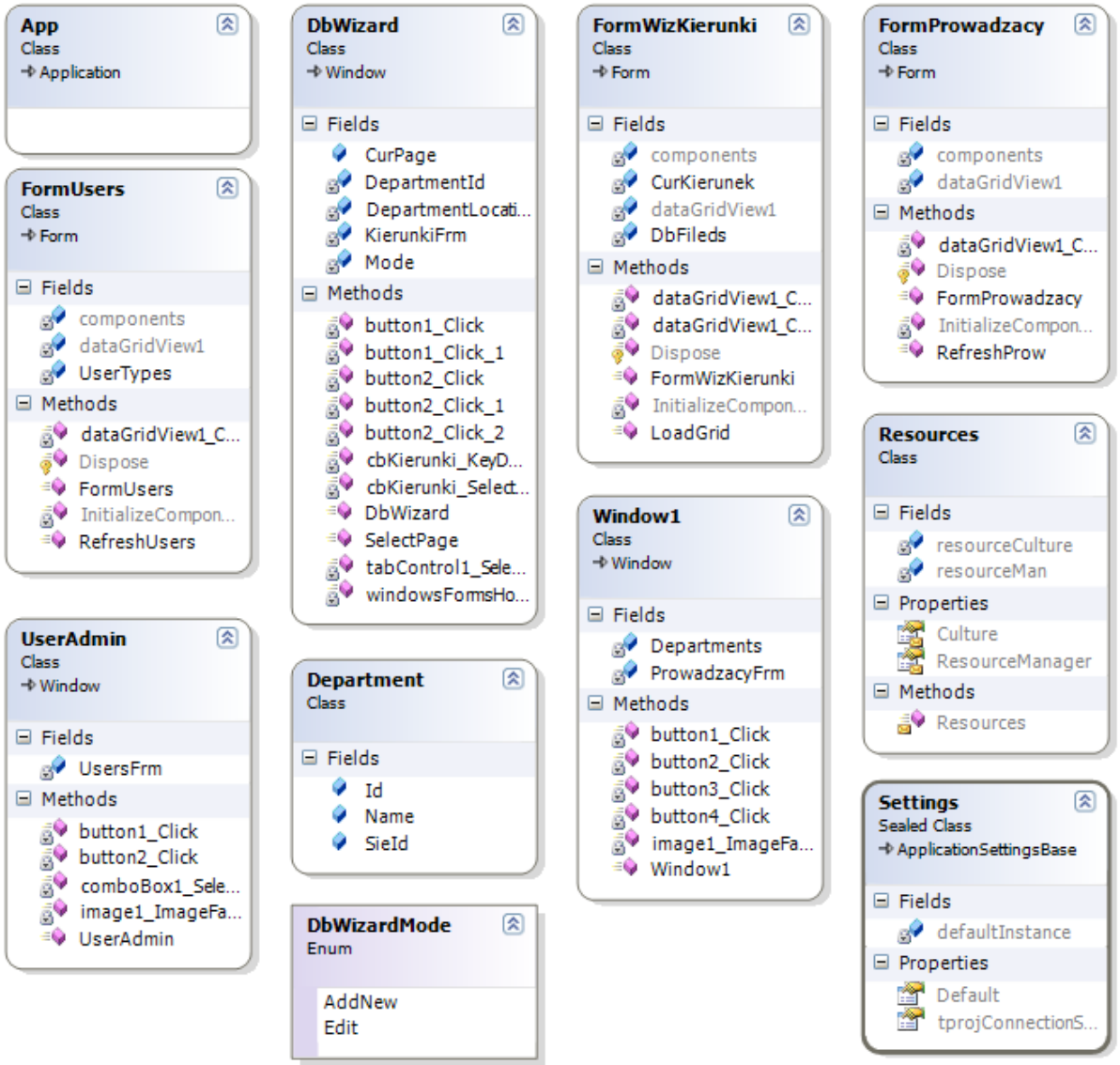
LoginForm jest formularzem wyświetlanym podczas uruchamiania programu, to na nim wyświetlane są informacje logowania.

Klasa Utils służy do kodowania i dekodowania haseł i adresów serwera.

Klasa VerifyStudent odpowiedzialna jest za walidacje studenta, który chce się połączyć z bazą. Po prawidłowej walidacji wykonuje metodę łączenia z bazą w przeciwnym wypadku wyrzuca błąd. VerifyUser działa

tak samo jak klasa VerifyStudent z tą różnicą, że przyznaje inne prawa dostępu.

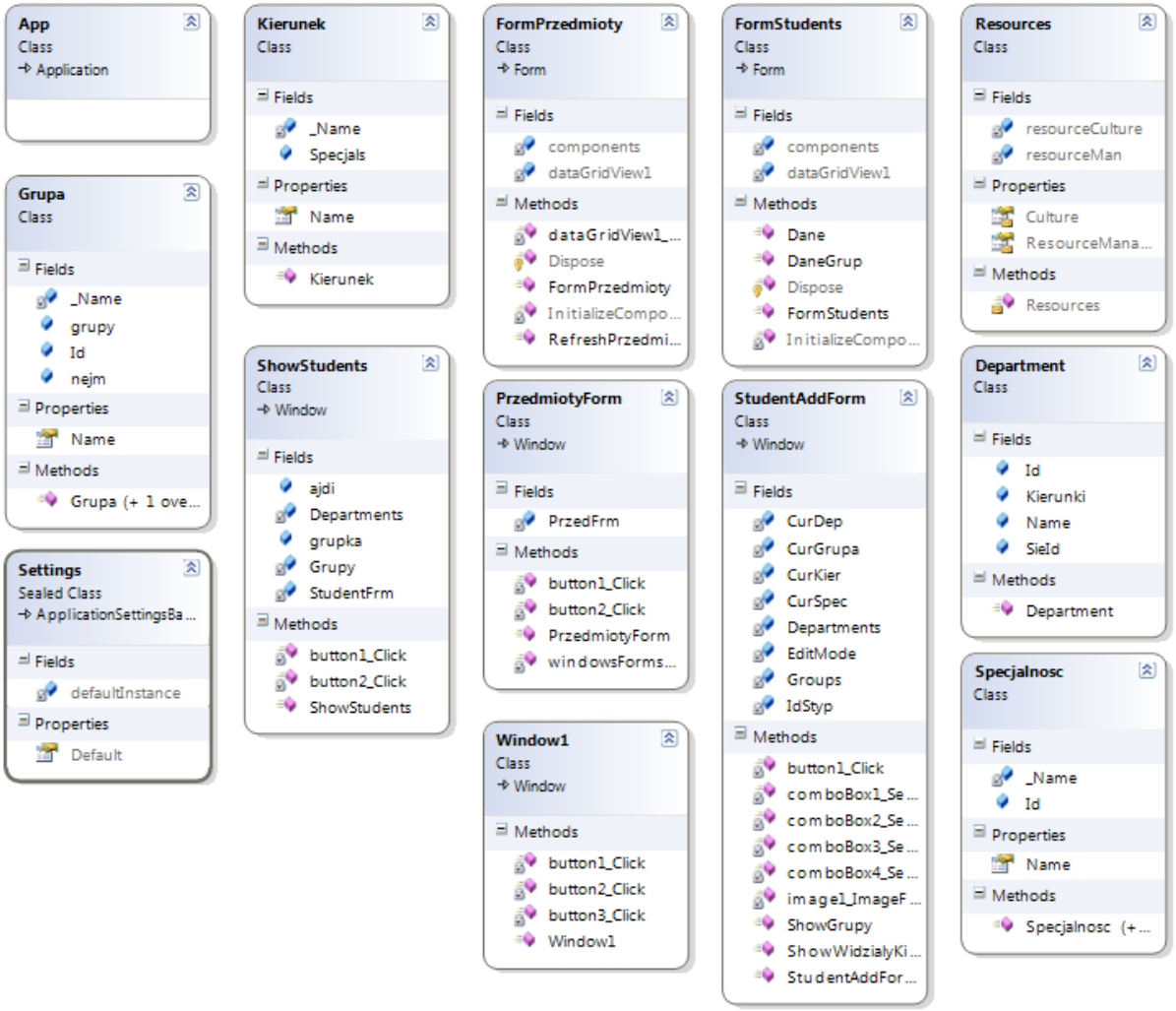
b) Moduł Administratora



Klasy FormUser, FormWizKierunki i FormProwadzacy są to formatki okien programu wyświetlające odpowiednio użytkowników, kierunki oraz prowadzących.

Klasa DbWizard to formatka panelu administratora, na niej wyświetlane są wszystkie informacje zawarte w w/w klasach. Również w tej klasie zdefiniowane są metody odpowiedzialne za zapytanie SQL z bazą.

c) Moduł Pracownika



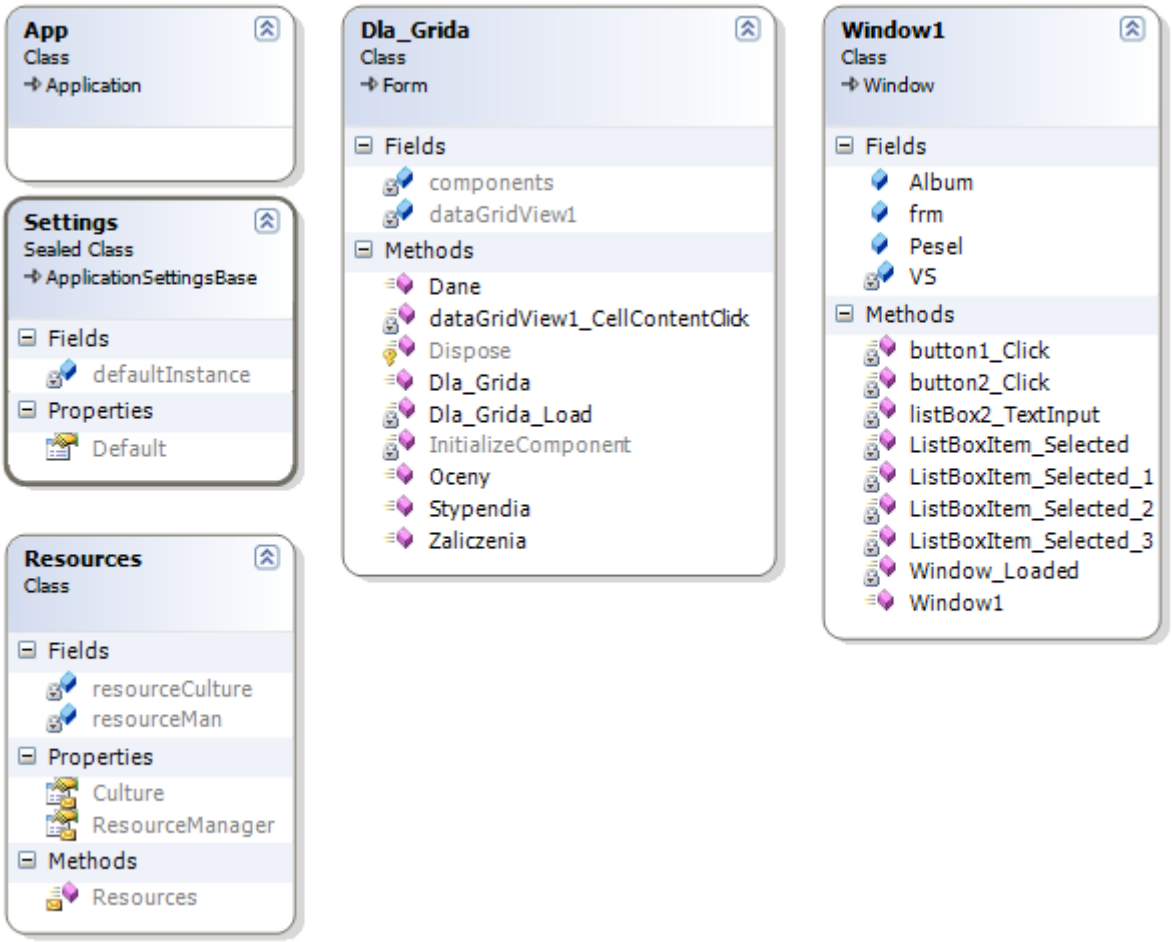
Klasy FormPrzedmioty i FormStudents służą do wyświetlania odpowiednio przedmiotów i studentów.

W klasie Grupa znajdują się informacje o grupie.

Analogicznie jak poprzednio w klasach Kierunek, Department i Specjalność znajdziemy informacje zawierające poszczególne rekordy w bazie danych.

Klasy ShowStudents, StudentAddForm oraz Window1 odpowiadają odpowiednio za wyświetlanie studentów, dodawanie studentów do bazy oraz okno główne pracownika.

d) Moduł Studenta



W tym module klasa Dla_Grida odpowiada za wyświetlanie informacji o zalogowanym studencie.
Window1 jest to okienko aplikacji.

Ad7. Projekt systemu bazodanowego.

Baza danych w przypadku naszego projektu opiera się na technologii *MySQL*, co bynajmniej nie wyklucza użycia w przyszłości innego systemu relacyjnego, jak np. *Oracle*, *IBM DB2*, itd.

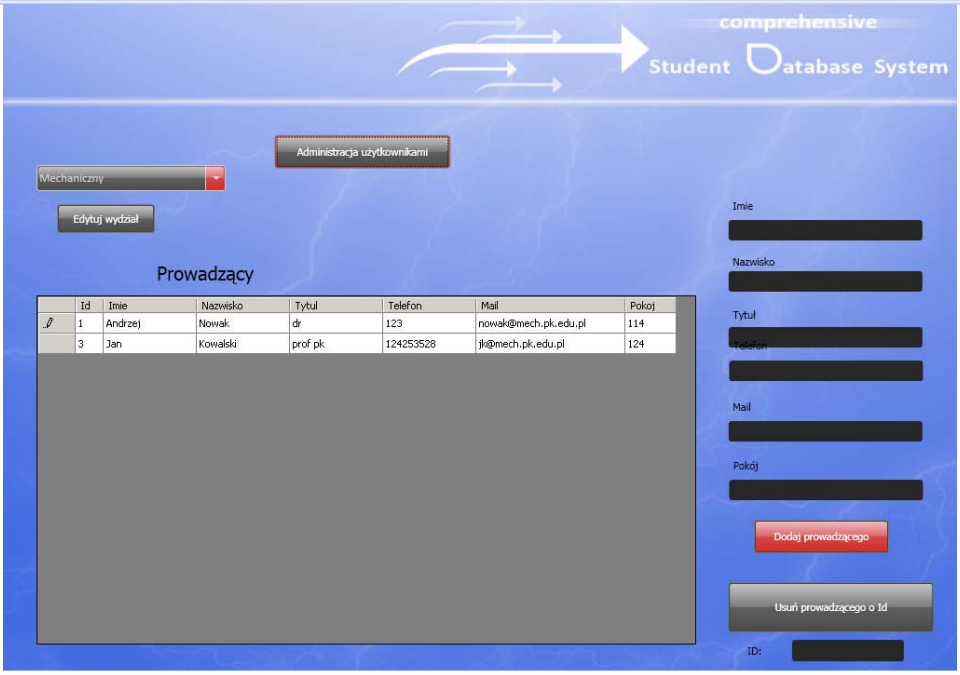
Zapewnia to elastyczność systemu i odwzorowanie struktury bazy w kodzie programu.

Struktura bazy została zaprojektowana wg wszystkich obowiązujących norm. Jest czytelna i przejrzysta.

Ze względu na duży rozmiar obrazu, schemat dołączamy w osobnym pliku (*patrz Załącznik nr3*).

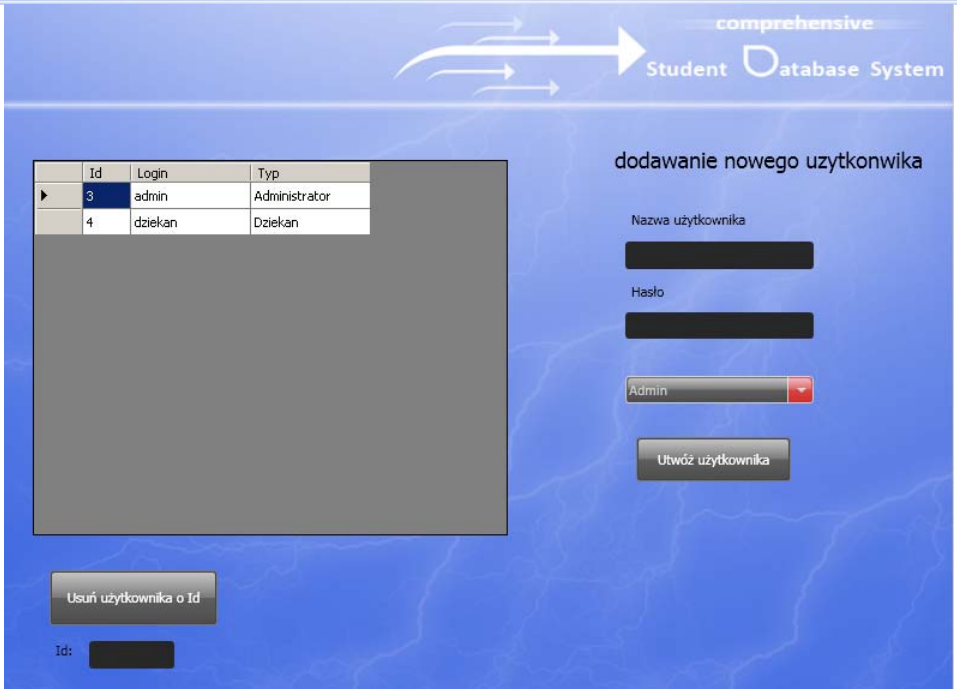
Ad8. Przykład interfejsu.

Wygląd i filozofię działania interfejsu użytkownika przedstawimy w oparciu o *Moduł Administratora.*



Interfejs1

Interfejs2



Interfejs3

Interfejs użytkownika w każdym module opiera się o takie same zasady. Po uprzednim zalogowaniu, otwiera się główne okno modułu – *Interfejs1.* Umożliwia ono przeprowadzanie podstawowych operacji, zatwierdzanych klikaniem na buttony.

Jeżeli użytkownik chce uzyskać bardziej wyrafinowane możliwości, po wybraniu odpowiedniej opcji umieszczonej w listach lub buttonach, zostaje przeniesiony do nowego okna, które przeprowadza go krok po kroku, przez cały proces edycyjny – *Interfejs2.* Drugą możliwością, w przypadku funkcji, nie wymagających wypełniania wielu pól, użytkownik przenosi się do zwykłego nowego formularza – *Interfejs3.*

Budowa całego interfejsu została wykonana przy użyciu WPF – silniku bazującym na .NET3, WPF integruje interfejs użytkownika i grafikę aplikacji.

Ad9. Plan Testów.

Zaplanowane testy mają na celu weryfikację osiągniętych efektów, względem postawionych przez klienta wymagań. Pragniemy również ustalić zgodność naszego systemu z założeniami wyspecyfikowanymi w priorytetach *Moscow.*

Testy komponentów

W tym miejscu sprawdzimy wszelkie funkcjonalności systemu, dostępne potem w aplikacjach klienckich.

Dane testowe podzielimy zgodnie z przyjętymi standardami na:

1. nietypowe poprawne
2. typowe poprawne i błędne
3. nietypowe błędne

Testy te odbywać się będą po zaimplementowaniu każdej nowej funkcjonalności aby sprawdzić jej działanie.

Testy aplikacji klienckich, integracyjne

Ten rodzaj testów ma na celu zbadanie powiązań pomiędzy funkcjami, a w szerszym ujęciu badanie wzajemnych oddziaływań danych funkcjonalności, aplikacji klienckich na siebie wzajemnie. Badanie to pozwoli odkryć ewentualne braki spójności w systemie.

Testy te obejmować będą następujące strategie:

1. wątkowa THREAD , np. działanie kilku procesów jednocześnie
2. obciążeniowa STRESS , np. zachowanie bazy danych pod dużym obciążeniem
3. wydajnościowa PERFORMANCE , np. sortowanie wielu danych

Ponadto przetestowane zostanie spełnienie wymagań sprzętowych.

Testy te odbywać się będą po przejściu testów komponentów oraz po wyprodukowaniu systemu w wersji *Release Candidate.* Zakończenie tych testów warunkuje przejście do kolejnego etapu testów.

Testy środowiska bazodanowego

Testowana tutaj będzie poprawność bazy danych, zarówno pod względem technicznym, jak i zgodności z postawionymi przed nią zadaniami.

Testy te właściwie trwać będą przez cały okres produkcji oprogramowania.

Testy akceptacji

Test ten przeprowadza klient. Pomyślne przejście tego testu przez oprogramowanie oznacza akceptację klienta i oznacza formalne zakończenie prac nad pierwszą pełną wersją oprogramowania.

Ad10. Szacowanie kosztów wdrożenia.

Obliczenie kosztu sprzętu sprowadza się do prostej zależności.

koszt sprzętu = koszt narzędzi + koszty licencji

Do wykonania systemu używamy darmowego środowiska Visual Studio Express 2008 oraz również bezpłatnego oprogramowania bazodanowego MySql. Reasumując koszt licencji programowych:

koszty licencji = 0zł;

Koszt sprzętu zawiera się w stanowisku komputerowym typu PC dla każdego z projektantów. Dodatkowy koszt stanowi serwer testowy z wykupionym stałym IP.

1. koszt jednego stanowiska 2000zł
2. koszt serwera testowego 1000zł

Powyższe wiadomości implikują:

koszty sprzętu = 2000x4 + 1000 = 9000zł.

Następną kwestią jest koszt opracowania systemu.

Tu sprawa nie jest tak trywialna.

koszt opracowania = koszt podstawowy x niezawodność x czas x zasoby x narzędzia x ekspertyza x pensja

koszt podstawowy = Złożoność Systemu x KDSI x Multiplikator wyrażone w osobomiesiącach

(*Złożoność Systemu*- od 2.4 dla s.jednowarstwowych do 3.6 dla wielowarstwowych,
KDSI Kilo Delivered Source Instructions -tysiące dostarczonych linii kodu, *Multiplikator*- złożoność danych od 0.7 dla alfanumerycznych do 1.7 dla obiektowych -medialnych)

niezawodność -współczynnik niezawodności i bezpieczeństwa systemu od 0.5 dla nieistotnego poziomu do 2.0 dla w pełni zabezpieczonego systemu

czas -wydłużenie lub skrócenie czasu realizacji od 0.5 dla długiego cyklu do 2.0 dla wykonania przed terminem

zasoby - wielkość zasobów danych/sieci od 0.5 dla MB do 2.0 dla TB

narzędzia - współczynnik dostępności narzędzi, od 0.5 dla w pełni dostępnych CASE do 2.0 dla unikatowych rozwiązań bez wspomagania

ekspertyza - współczynnik zależny od zewnętrznych audytów i ekspertyzy, od 0.7 dla braku ekspertyzy do 2.0 dla wymagających certyfikatów

pensja - pensja programisty

koszt podstawowy = 3.2 x 6 x 1.2 = 23.04

koszt opracowania= 23.04 x 1.5 x 2 x 1 x 0.5 x 0.7 x 2800 = 67737.60zł.

Koszt całościowy

Koszt = koszt sprzętu + koszt opracowania = 9000 + 67737.60

76737.60zł

Ad11. Pielęgnacja.

Zagadnienie pielęgnacji będzie obejmować pomoc techniczną sytemu, konserwację serwerów rdzeniowych jak i lustrzanych, udoskonalenia, poprawki i łatki aplikacji finalnej.

Przez pierwsze dwa lata od momentu publikacji pierwszej finalnej wersji aplikacji, objęta ona będzie e-mailową pomocą techniczną. Pomoc ta będzie obejmowała rozwiązywanie problemów jak i służyła na zbieraniu

listy najczęstszych usterek i błędów systemu, co skutkować będzie wypuszczaniem poprawek do systemu.

Głównym i najistotniejszym elementem pielęgnacji będzie konserwacja serwerów bazodanowych. Standardowo, z założenia w ostatni dzień miesiąca wykonywana będzie konserwacja serwerów, sprawdzanie spójności bazy i stworzenie fizycznej kopii zapasowej bazy danych. W/w konserwacja będzie wykonywana na serwerze głównym, natomiast na serwerach lustrzanych w kolejnych dniach, aby podczas konserwacji serwera głównego wszelkie obliczenia przejmował serwer lustrzany.

Problemy z systemem nadsyłane za pośrednictwem pomocy technicznej jak i nowo powstałe wynikające ze zmian oprogramowania i sprzętu będą systematycznie rozwiązywane w postaci wypuszczania łatek systemu.

Koszt pielęgnacji:

Konserwacja serwerów bazodanowych – rdzeniowego + 2 serwery lustrzane (czas trwania około 4 godziny, standardowa stawka serwisanta bazodanowego 50zł/h).

3 serwery * 4 godziny * 50zł = **600zł**

Koszt obsługi pomocy:

Do opracowywania poprawek wystarczy 3 osoby. Koszt obsługi pomocy to koszt pensji pracowników.

3 osoby * 2000zł = **6000zł**