Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Filip Grotkowski
Nr albumu: 235984

USOS: System raportowania i analiz statystycznych

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr. Przemysława Biecka
Instytut Matematyki Uniwersytetu Warszawskiego

Wrzesień 2011

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

Ocenarium jest systemem analitycznym związanym z USOS-em. Jego celem jest pokazanie różnych aspektów funkcjonowania Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Oprócz tytułowych ocen są to ankiety, profile seminariów, rankingi prowadzących zajęcia czy *analiza przeżycia* studentów na studiach.

Ocenarium dostarcza środowisko do generowania *raportów*. Podstawą *raportu* są dane zaczerpnięte z bazy danych, zaś w jego skład wchodzą tabele, wykresy i wszystko to co ma pomóc w wizualizacji danych. W *Ocenarium raporty* generowane są za pomocą skryptów języka R, zaś produktem końcowym są dokumenty HTML.

Kluczowym elementem *Ocenarium* jest hurtownia danych zapożyczona z innego modułu USOS-a (*USOS Statystyki*). Pozwala ona na jednolity dostęp do danych pochodzących z wielu źródeł (USOS, IRK). W aktualnej wersji jedynym źródłem danych jest USOS.

Praca omawia system *Ocenarium* z różnych punktów widzenia. Poruszana jest integracja z hurtownią danych, architektura systemu, sporządzone *raporty* i w końcu – proces wdrażania.

Słowa kluczowe

inżynieria oprogramowania, statystyka, data mining, R, hurtownia danych, Oracle, USOS, wizualizacja danych, analiza współwystępowania

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

G. Mathematics of Computing
G.3 Probability and Statistics
Statistical software
H. Information Systems
H.4 Information Systems Applications

H.4.2 Types of Systems
Decision support (e.g. MIS)

Tytuł pracy w języku angielskim

USOS: Reporting and statistical analysis system

Spis treści

Podziękowania

Dziękuję mojemu promotorowi dr. Przemysławowi Bieckowi za udzieloną mi pomoc. Dziękuję nie mniej za zaproponowanie mi tematu tej pracy.

Osobne podziękowania dotyczą instalacji i wdrożenia *Ocenarium*. Proces ten nie byłby możliwy bez pomocy wielu osób. Opiekę i wsparcie już na samym początku okazała dr Janina Mincer-Daszkiewicz. W dostępie do źródłowej bazy danych USOS i migracji pomógł Artur Popławski. Odzyskanie z backupu hurtowni, pomoc w konfiguracji aplikacji Django oraz pomoc administracyjna dotycząca obu maszyn (`usoshurt.mimuw.edu.pl` i `usosphp.mimuw.edu.pl`) to zasługa Łukasza Szczęsnego z Laboratorium Komputerowego Wydziału MIM UW.

Pragnę wszystkim wymienionym osobom podziękować za udzieloną mi pomoc.

Rozdział 1

Wprowadzenie

1.1. Struktura pracy

Niniejszy rozdział wprowadza czytelnika w tematykę którą porusza praca. Definiujemy w nim słownik używanych pojęć (1.2) i krótko opisujemy

- USOS-a (1.3)
- problem który chcemy rozwiązać (1.4)
- poczynione założenia (1.5)
- i użytkowników końcowych systemu (1.6).

Rozdział 2 poświęcony jest sercu *Ocenarium* czyli hurtowni danych systemu *Statystyki*. Omawiana jest motywacja dlaczego w ogóle potrzebna jest specjalna, dedykowana baza danych dla naszego systemu.

W rozdziale 3 opisana jest architektura systemu.

Rozdział 4 zawiera szczegóły raportów wykonanych za pomocą *Ocenarium*. Na tych przykładach wskazane są istotniejsze trudności i problemy w komunikacji baza danych – moduł statystyczny – interfejs. Jednocześnie mamy okazję do przeglądu interfejsu *Ocenarium* i zrealizowanych zadań określonych w 1.6.

W rozdziale 5 pokazujemy w jaki sposób przebiega instalacja i konfiguracja *Ocenarium*. Poruszamy ten temat w odniesieniu do hurtowni danych (5.3) oraz części aplikacyjno-interfejsowej (5.2).

Podsumowanie pracy znajduje się w rozdziale 6. Wymieniamy w nim sukcesy i porażki projektu *Ocenarium*. Odnosimy się do założonych celów oraz podsumowujemy zrealizowane i nie zrealizowane zadania.

Epilogiem pracy są dodatki. Dodatek A pozwala w praktyce przyjrzeć się architekturze *Ocenarium*. Oprócz wartości dydaktycznej nakreśla na przykładzie w jaki sposób przebiega proces dodawania nowego *raportu* (por. 1.2).

Natomiast dodatek C opisuje zawartość załączonej do pracy płyty CD.

1.2. Słownik pojęć

- *Ocenarium*– nazwa projektu wykonanego w ramach niniejszej pracy magisterskiej.
- *raport* – główny produkt zapewniany przez *Ocenarium*. Jest generowany za pomocą skryptu pobierającego dane z bazy i prezentuje te dane w postaci tekstu, tabel czy obrazków osadzonych w dokumencie HTML.
- *ususobis* – SID (The Oracle System ID) bazy danych będącej kopią podstawowej bazy danych USOS. SID w ogólności ma jednoznacznie identyfikować bazę danych w systemie.
- *Statystyki* (*System Statystyki*, *System USOS Statystyki*) – system analityczny do informowania władz uczelni. Jako moduł USOS-a jest konkurencyjny w stosunku do naszego systemu. Dodatkowo *Ocenarium* korzysta z bazy danych *Statystyk*. Więcej informacji można znaleźć w rozdziale 2.3.
- *adminrol* – nazwa uprzywilejowanego użytkownika bazy danych *ususobis*.
- *hurtownia* – poza rozdziałami 2.1 i 2.2 – gdzie kontekst jest ogólniejszy – jest to baza danych (hurtownia danych) *Ocenarium* i *Systemu Statystyki*.
- *ususphp* – host interfejsu i części statystycznej *Ocenarium*.
- *usushurt* – host hurtowni danych *Ocenarium*.

1.3. USOS – wprowadzenie

Uniwersytecki System Obsługi Studiów powstał w 2000 r. na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. USOS to efekt współpracy największych polskich uczelni. System powstał w wyniku zapotrzebowania na kompleksowe narzędzie informatyczne służące do zarządzania sprawami studiów w szkole wyższej. Obecnie jest wdrożony na ponad 30 uczelniach w Polsce i obsługuję czwartą część wszystkich studiujących. Wśród głównych zastosowań USOS znajdują się między innymi:

- rekrutacja na studia i immatrykulacja,
- Elektroniczne Legitymacje Studenckie (drukowanie, przedłużanie ważności itp.),
- przygotowywanie oferty dydaktycznej (przedmioty, zajęcia, grupy, terminy, prowadzący),
- zarządzanie tokiem studiów (programy studiów wszystkich stopni i rodzajów, zapisy na zajęcia, protokoły z ocenami, zaliczenia itd.),
- podania studenckie,
- prace i egzaminy dyplomowe, elektroniczna archiwizacja prac dyplomowych,

- stypendia,
- akademiki,
- płatności za usługi edukacyjne,
- wsparcie dla Procesu Bolońskiego,
- praktyki zawodowe studentów,
- ankiety,
- sprawozdawczość,
- sprawy pracownicze (zatrudnienia etatowe i nieetatowe, rozliczanie pensum itp.),
- Biuro Karier,
- uniwersyteckie archiwum.

Dzięki obszerności zastosowań USOS pełni rolę centralnego punktu gromadzenia informacji z całej uczelni, co znacząco usprawnia zarządzanie studiami, umożliwia ujednolicenie procedur uczelnianych oraz pozwala na efektywne wprowadzanie inicjatyw ogólnouczelnianych, takich jak wspólna dla studentów wszystkich kierunków oferta przedmiotów nieobowiązkowych, lektoratów, zajęć z wychowania fizycznego, egzaminów certyfikacyjnych, a także centralna autoryzacja studentów i pracowników w serwisach internetowych uczelni, generowanie unikatowych w skali uczelni numerów indeksów i dyplomów. Przechowywanie danych w postaci cyfrowej istotnie redukuje liczbę generowanych tradycyjnych dokumentów, pozwala między innymi na wyeliminowanie papierowych protokołów z ocenami, kart egzaminacyjnych, podań studenckich, a nawet indeksów.

USOS to nie tylko scentralizowana baza danych obsługiwana przez pracowników administracji uczelnianej. Z systemem stowarzyszony jest szereg serwisów internetowych dla kandydatów na studia, studentów i nauczycieli akademickich.

Jak widać USOS jest wielkim źródłem informacji. Wiele z tych informacji z różnych względów (np. prawnych) nie może być upubliczniona, lub upubliczniona tylko wąskiej grupie osób. Jednak część z nich przy dobrych chęciach osób zarządzających Uczelnią może być zaprezentowana z korzyścią wszystkim pracownikom i studentom. To co jest warte podkreślenia to fakt, że dla statystyka USOS posiada wielką wartość jako źródło danych, a analizowane w niniejszej pracy aspekty USOS-a to tylko ułamek pracy która może być wykonana.

Więcej informacji znajduje się na stronie projektu USOS ([1]).

1.4. Sformułowanie problemu

Celem projektu *Ocenarium* jest dostarczenie systemu analitycznego generującego *raporty*. Priorytetową kwestią jest tutaj względna łatwość dodawania nowych *raportów*. Podstawą *raportu* są dane zaczerpnięte z bazy danych, zaś w jego skład wchodzą tabele,

wykresy i wszystko to co ma pomóc w wizualizacji danych. W *Ocenarium raporty* generowane są za pomocą skryptów języka R (3.2.1), zaś produktem końcowym są dokumenty HTML.

1.5. Założenia

Dane które analizujemy w *Ocenarium* pochodzą z bazy danych USOS. Dotyczą ocen uzyskiwanych przez studentów na poszczególnych przedmiotach, wyników ankiet czy skreśleń z listy studentów. Zatem specyfika tych danych – ich cykliczna i nieczęsta aktualizacji – pozwala na generowanie *raportów* offline. Dzięki temu nie musimy dbać o złożoność obliczeniową analiz.

Założeniem projektu jest to że większość jego kodu źródłowego napisana jest w R. Jednym z wyrazów tego faktu jest duży nacisk położony na prosty dostęp do bazy danych z poziomu kodu R. Staramy się tutaj przeprowadzić komunikację z bazą danych bez kroków pośrednich.

W *Ocenarium* zakres naszych zainteresowań zawężamy do danych związanych tylko z Wydziałem Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Nie oznacza to jednak, że rozszerzenie na inne Wydziały Uniwersytetu nie jest możliwe, wręcz przeciwnie, baza danych jest do tego gotowa.

1.6. Użytkownicy końcowi

Ocenarium nie ma jednej grupy docelowej użytkowników końcowych. Można wyróżnić kilka takich grup. Oto one wraz z raportami dla nich dedykowanymi. Gwiazdka przy raporcie oznacza że ta funkcjonalność została w *Ocenarium* zrealizowana.

1. Studenci zainteresowani są informacjami o tym
    - Jaki jest rozkład ocen z danego przedmiotu w kolejnych cyklach dydaktycznych? (*)
    - Jaka jest liczba osób która zalicza dany przedmiot w pierwszym terminie? (*)
    - Jaka jest liczba osób która nie zalicza przedmiotu w ogóle?
    - Jak wygląda lista najbardziej lubianych ćwiczeniowców lub takich których studenci osiągają najlepsze wyniki w nauce? (*)
2. Prowadzący zajęcia zainteresowani są informacjami o tym
    - Jakie wyniki osiągają ich studenci w nauce? (*)
    - Jak oceniani są przez studentów w ankietach? (*)
3. Komisja dydaktyczna zainteresowana jest informacjami o tym
    - Jak wygląda *analiza przeżycia* studentów na studiach? (*)
    - Na jakim etapie student kończy studia? (*)

- Jakie etapy powtarza? (*)
- Przez niezaliczenie których przedmiotów kończy studia?
- Jak długo studiuje?

Powyższa lista nie wyczerpuje oczywiście wszystkich możliwych grup beneficjentów działalności *Ocenarium*, nawet w aktualnej wersji.

Rozdział 2

Baza danych

W tym rozdziale omówimy bazę danych z której korzysta aplikacja *Ocenarium*. Po dokładniejsze opracowanie można sięgnąć do dokumentacji USOS stworzonej przez Grzegorza Kukawskiego ([5]), jego pracy magisterskiej ([6]), i dwóch wcześniejszych prac magisterskich poświęconych systemowi *Statystyki* ([7], [8]). Zaznaczmy, że sekcje tego rozdziału są opracowane głównie na podstawie pracy magisterskiej ([6]).

2.1. Hurtownie danych – wprowadzenie

Zaczniemy od scharakteryzowania ogólnych własności hurtowni danych. Po pierwsze, hurtownia danych jest rodzajem bazy danych. W funkcjonowaniu hurtowni standardowymi operacjami są odczyt i wyszukiwanie. Dużo rzadziej zdarza się zapis, czy modyfikacja danych. Z tego względu hurtownie są zoptymalizowane pod możliwie szybki odczyt, wyszukiwanie i szczegółową analizę zawartości na czym cierpią modyfikacja i usuwanie danych.

Architektura hurtowni danych składa się z trzech warstw: *stage*, *integration* i *access*. Warstwa *stage* jest użyta do przechowywania surowych danych i jest niewidoczna dla użytkownika. Warstwa *integration* służy do integracji danych i stwarza warstwę abstrakcji dla użytkownika. Ostatnia z warstw – warstwa *dostępu* – służy do udostępniania danych użytkownikowi.

Z definicji, głównym celem hurtowni jest gromadzenie danych. Dane te często pochodzą z wielu źródeł, które to mogą organizować je w niekompatybilny ze sobą sposób (np. bazy Oracle i MySQL jak to jest w przypadku USOS-a i IRK). Hurtownia integruje te dane. Do eksploracji danych hurtowni służą różne systemy wspierające. Wśród nich można wyróżnić

- ETL (ang. Extract, Transform, and Load). Są to narzędzia wspierające proces pozyskiwania danych z systemów zasilających hurtownie danych
- narzędzia do tworzenia raportów
- wielowymiarowe systemy kostek analitycznych (por. rozdz. 2.4.1)

Innym aspektem związanym z hurtowniami jest prezentacja trendów przedsiębiorstwa w czasie. Do tego potrzebne są dane historyczne, których zwykle systemy źródłowe nie przechowują, lub przechowują tylko w ograniczonym zakresie. Hurtownie danych niwelują ten problem.

Oprócz wspomnianych – analizy trendów i archiwizacji danych – głównymi polami zastosowań hurtowni są jeszcze wspieranie podejmowania decyzji, analiza efektywności czy analizy finansowe.

2.2. Business Intelligence

W tym rozdziale wprowadzimy pojęcie ściśle związane z *Ocenarium* oraz systemem *Statystyki*. Jest nim Business Intelligence.

Pokrywa ono szerokie spektrum zagadnień, takich jak: praktyki, metodyki, narzędzia, czy technologie informatyczne związane z analizą danych. Business Intelligence kojarzone jest z Hurtowniami Danych oraz systemami analityczno-raportującymi, jednak zakres BI jest dużo szerszy. W języku polskim nie mamy żadnego – zaakceptowanego przez całą branżę – odpowiednika dla tego pojęcia. W skrócie, Business Intelligence może być postrzegane, jako przekształcanie danych w informacje, a informacji w wiedzę w celu optymalizacji działania procesów biznesowych i całej organizacji.

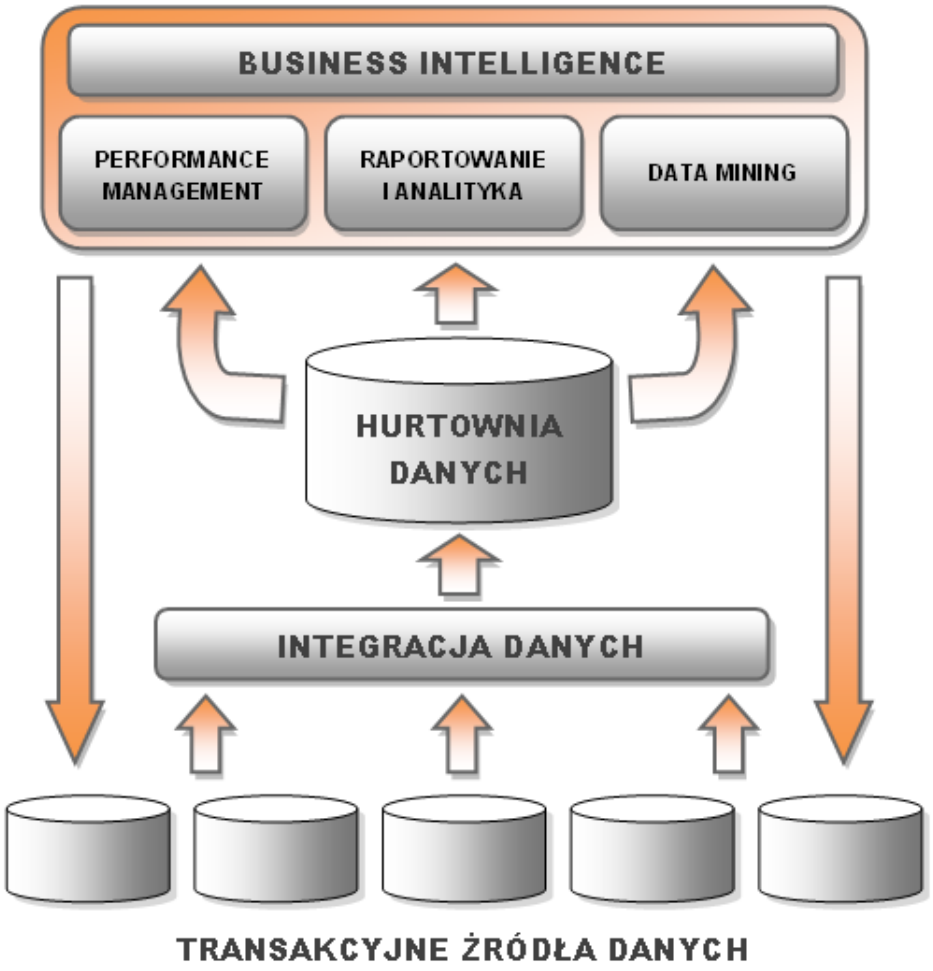
Na rysunku 2.1 prezentujemy schemat, który wiąże wprowadzone dotąd pojęcia. W następnym rozdziale zobaczymy (wraz z rysunkiem 2.5) konkretną realizację tego schematu. Dodatkowych informacji można zasięgnąć na portalu branżowym bi.pl ([23]).

2.3. System *Statystyki*

System *Statystyki* jest systemem analitycznym (Bussiness Intelligence, por. rozdz. 2.2) dedykowanym władzom uczelni. Zadaniem *Statystyk* jest dostarczenie użytkownikowi wiedzy uzyskanej na podstawie informacji zebranych w wolumenach danych. Dane mogą pochodzić z wielu źródeł, zaś wiedza może być prezentowana w różnych przekrojach. W ten sposób wspomagany jest proces podejmowania decyzji.

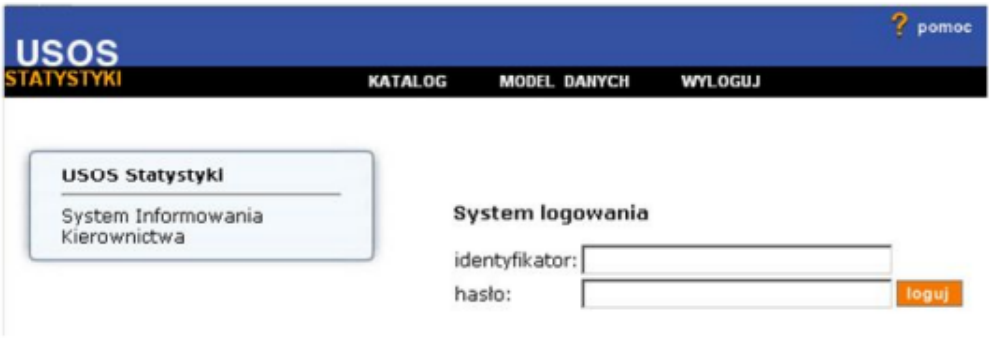
Statystyki są systemem niezależnym technicznie od innych (mają swoja własną bazę danych i interfejs użytkownika) i są przygotowane tak, aby mogły łatwo integrować się z innymi systemami w celu pozyskania danych do analizy. Na chwilę obecną hurtownia czerpie dane z systemu USOS oraz systemu Internetowej Rejestracji Kandydatów (IRK). Dotychczas powstały cztery wersje *Statystyk*. Trzy pierwsze były realizowane w ramach prac magisterskich: Jarosława Łukaszewicza ([8], UMK, 2005), Cezarego Bronnego ([7], MIM UW, 2006) i Grzegorza Kukawskiego ([6], MIM UW, 2008). Czwarta zaś jest rozwijana przez Uniwersytet Łódzki i była gotowa na początku 2011 roku (niestety nie wiemy nic o dokumentacji).

Aby dać czytelnikowi punkt zaczepienia pokazujemy dalej kilka obrazków z interfejsem użytkownika *Statystyk*. Na pierwszym (2.2) mamy ekran logowania wraz z menu głównym. Rysunek 2.3 to lista zdefiniowanych kategorii i statystyk. Z kolei rysunek 2.4



Rysunek 2.1: Architektura BI. Źródło: bi.pl ([23])

.



Rys. 1 Autentykacja

Rysunek 2.2: Ekran logowania *Statystyk*. Źródło: praca magisterska Grzegorza Kukawskiego ([6]).

przedstawia przykładową statystykę. Jest to struktura zatrudnienia na Wydziale MIM UW.

2.4. Hurtownia systemu *Statystyki*

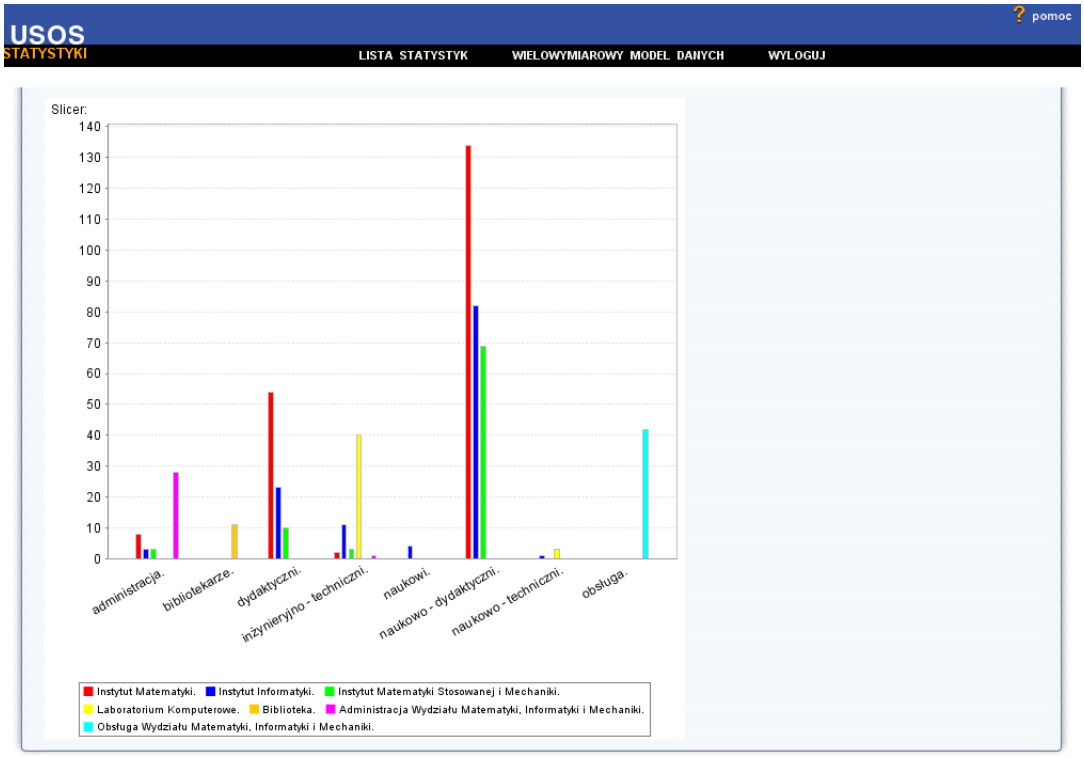
Najistotniejszym elementem *Statystyk* jest ich baza danych. W tym rozdziale chcielibyśmy omówić krótko architekturę hurtowni systemu *Statystyki*. Zanim do tego przejdziemy potrzebne jest wyjaśnienie jeszcze jednego pojęcia. Mianowicie chodzi o OLAP.

2.4.1. OLAP

Technologia OLAP jest angielskim skrótem rozwijanym jako Online Analytical Processing. Jest to zestaw narzędzi (oprogramowanie) wspierających podejmowania decyzji. Głównym zadaniem OLAP jest prezentacja i wyszukiwanie danych w hurtowni. Warto podkreślić, że OLAP został stworzony z myślą o przetwarzaniu dużej ilości danych w czasie rzeczywistym (najczęściej są to miliony rekordów w bazie). Systemy tego typu zazwyczaj są oparte na wielowymiarowym modelu danych, którego głównym pojęciem jest tak zwana **kostka/kostka analityczna** (ang. cube). **Kostka** jest tablicą o wielu **wymiarach**, na przecięciu których znajdują się wartości komórek, nazywane w terminologii fachowej **miarami**.

Aby rozjaśnić te pojęcia podamy przykłady związane z tematyką tej pracy. **Wymiarem** może być rok rozpoczęcia studiów, płeć, czy narodowość studenta. Przykładowymi **miarami** są liczba przedmiotów na które student się zapisał, liczba studiowanych przez niego kierunków czy średnia ocen na studiach. Upraszczając nieco definicje można napisać, że **wymiary** reprezentują dane opisowe (co?, kto?, kiedy?), **miary** zaś są wskaźnikami numerycznymi (ile?).

Rysunek 2.3: Lista zdefiniowanych kategorii i statystyk. Źródło: praca magisterska Grzegorza Kukawskiego ([6]).



Rysunek 2.4: Wykres przedstawiający strukturę zatrudnienia na Wydziale MIM UW. Źródło: praca magisterska Grzegorza Kukawskiego ([6]).

Ważną cechą **wymiarów** jest ich hierarchiczność – mogą być podzielone na poziomy, np. data ma **wymiary** dzień, kwartał, rok.

Implementacje OLAP

OLAP zasadniczo dzielimy na trzy typy architektur, które różnią się między sobą strukturą i organizacją danych. Oto one

- MOLAP (ang. multidimensional OLAP) – jest klasyczną architekturą OLAP. Opiera się na wielowymiarowych tablicach z zagregowanymi danymi. Przy dużym poziomie agregacji rozwiązanie to wymaga dużej przestrzeni dyskowej i dodatkowych nakładów na preprocessing. Jednak za tę cenę dostajemy szybkie operacje na strukturze wielowymiarowej (**kostkach**).

- ROLAP (ang. relational OLAP) – operuje na schematach relacyjnych. Tłumaczenie na strukturę wielowymiarową odbywa się poprzez serwer pośredniczący między użytkownikiem a serwerem relacyjnej bazy danych. Dodatkowo, w celu przyspieszenia analiz generowanych przez system tworzone są tabele agregujące, które w ramach struktury relacyjnej są gotowe do użycia przez serwer ROLAP.

  Zauważmy, że obok długiego preprocessingu, wadą wysokiego poziomu agregacji wstępnej dla obu wymienionych architektur jest miejsce zajmowane przez wyliczone dane, co uniemożliwia budowanie dużych baz analitycznych. Dlatego ważne jest zachowanie odpowiednich proporcji. Odbija się to w trzeciej z wymienianych przez nas architektur OLAP.

- HOLAP (ang. hybrid OLAP) – jest rozwiązaniem łączącym dwa poprzednie w takim sensie, że część danych jest utrzymywana na zasadach MOLAP, a część na zasadach ROLAP.

Więcej informacji o tym ciekawym zagadnieniu można znaleźć w [16] lub [17].

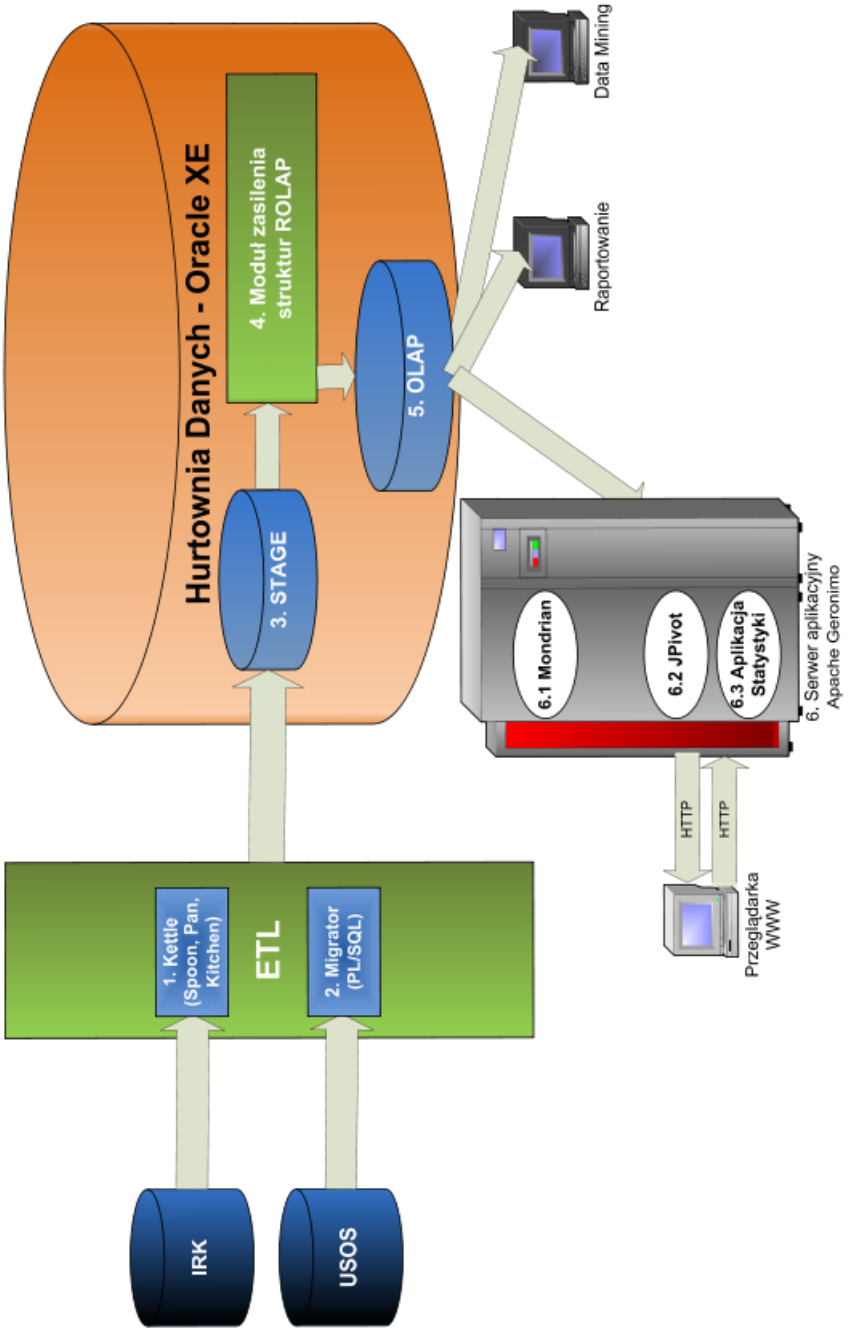
2.4.2. Architektura – schemat przetwarzania

Po teoretycznym wstępie przechodzimy do przedstawienia schematu architektury *Statystyk* za pomocą rysunku 2.5.

Zaczniemy od wyjaśnienia znaczenia i funkcji poszczególnych elementów architektury naszej hurtowni. Składniki które nas interesują mają numery od 1 do 5. Jest to część systemu związana z bazą danych.

- **Moduł ETL** – zestaw procesów, który jest odpowiedzialny za wczytanie danych z systemów źródłowych do schematu hurtowni danych STAGE.

- **Moduł STAGE** – element nr 3 na rysunku. Jest to Schemat przechowujący strukturę danych w postaci znormalizowanej, bez potrzebnych do analiz agregatów.



Rysunek 2.5: Architektura *Statystyk*. Źródło: praca magisterska Grzegorza Kukawskiego ([6]).

Opis tego punktu architektury warto rozwinąć. A to dlatego, że użycie dedykowanego serwera bazy danych miało największe znaczenie dla modułu zasilania hurtowni (aktualnie jest to STAGE). We wcześniejszych wersjach *Statystyk* analizy były oparte na zmaterializowanych widokach które stanowiły tabele faktów i były zasilane za pomocą standardowych mechanizmów dostarczonych przez firmę Oracle. Tabele **wymiarów** były najczęściej tabelami z relacyjnego modelu USOS. W miarę rozwoju *Statystyk* ujawniła się potrzeba wydzielenia specjalnej przestrzeni z ustrukturalizowanymi danymi z systemów źródłowych. Ponieważ bazą danych wykorzystaną na potrzeby hurtowni jest relacyjna baza Oracle, w naturalny sposób taką przestrzenią może być schemat bazy danych. W literaturze tego typu schemat nosi nazwę STAGE i tak też został nazwany w systemie *Statystyki*. Zatem podsumowując, schemat STAGE składa się z kopii tabel pochodzących ze źródłowych baz danych, ograniczonych ewentualnie do interesujących nas kolumn.

- **Moduł zasilania struktur ROLAP** – element nr 4 na rysunku. Jest to zestaw procesów zasilających strukturę ROLAP na podstawie danych ze schematu STAGE oraz wyliczających niezbędne agregaty.

- **Moduł OLAP** – element nr 5 na rysunku. Jest to schemat przechowujący struktury danych ROLAP, zoptymalizowane pod kątem kostek analitycznych.

- **Serwer bazy danych** – bazą danych na potrzeby hurtowni jest Oracle Database 10g Express Edition, która jest darmową wersją bazy danych Oracle.

2.4.3. Architektura – procesy

Zasilenie hurtowni danymi z bazy USOS

Na potrzeby wczytania danych z systemu USOS wykorzystano składowane procedury PL/SQL (element nr 2 na rysunku). Plikiem źródłowym jest `/home/fg235984/STATYSTYKI/SQL/05_STAGE.sql`. Takiego wyboru dokonano ze względu na użyty serwer zarządzania bazą danych systemu USOS oraz hurtowni danych. Mianowicie obie instancje są zainstalowane na bazie Oracle, i dlatego optymalnie było użyć mechanizmów dostarczanych przez tego producenta. Proces migracyjny danych z systemu USOS został przygotowany w ramach zaliczenia przedmiotu *Projekt dużej bazy danych* przez Macieja Hadysia ([18]).

Budowa pakietu obejmuje po jednej procedurze dla każdej z migrowanych tabel. W bazie zapisujemy datę ostatniej migracji, a w większości tabel każdy rekord posiada dodatkowo swoją datę utworzenia. Dzięki temu dodajemy, usuwamy i modyfikujemy odpowiednie rekordy. Zauważmy że nie musimy postępować tutaj naiwnie i czyścić tabelę schematu STAGE przy każdej migracji, a później na nowo dodawać całą zawartość tabeli źródłowej.

Oczywiście w przypadku mniejszych tabel możemy sobie pozwolić na usunięcie całej zawartości tabeli i przekopiowanie wszystkich rekordów ze źródła.

Zasilenie hurtowni danymi z bazy IRK

Bazą danych IRK jest MySQL. Z tego względu ponowne użycie procedur PL/SQL do migracji danych nie ma większego sensu. W hurtowni Grzegorza Kukawskiego proces importujący dane z IRK wykorzystuje darmowe narzędzie Pentaho Data Integration (Kettle) (por. [20]). Motywacją takiego wyboru są zarówno powody ekonomicznych, jak i bogata funkcjonalność tego narzędzia.

W najnowszej wersji *Statystyk* wykorzystano Migrator 2.0 (por. [19]). Migrator 2.0 (aktualnie 2.1.26) jest aplikacją synchronizującą bazy danych. W systemie USOS jest używany do synchronizacji danych pomiędzy centralną bazą Oracle oraz satelitarnymi bazami MySQL w poszczególnych aplikacjach WWW. Mimo, że Migrator 2.0 powstawał z myślą o systemie USOS, implementacja jest dość uniwersalna i może być użyta w zupełnie innych projektach (w szczególności synchronizacja wcale nie musi się odbywać pomiędzy silnikami baz danych Oracle i MySQL).

Praca jaką musi wykonać programista chcący przeprowadzić migrację polega na utworzeniu pliku XML, który później jest wejściem Migratora. Plik ten opisuje parametry połączenia między bazami i wymienia tabele które zamierzamy migrować.

Bardziej szczegółowe informacje znajdują się w dokumentacji [19].

Zasilenie struktur ROLAP

Proces zasilający strukturę kostek analitycznych jest zdefiniowany w pakiecie składowanym, napisanym w języku PL/SQL o nazwie `P_CUBES`. Plikami źródłowymi są `012p_cubes.pks` i `013p_cubes.pkb` w katalogu `/home/fg235984/STATYSTYKI/SQL/`. Pakiet zawiera definicje procedur. Wszystkie oprócz jednej z nich odpowiadają za wczytanie do struktur OLAP (element nr 5 rysunku) ze schematu STAGE (element nr 3) danych do poszczególnych **kostek** (por. rozdz. 2.4.1). Ostatnią procedurą, jest procedura sterująca `p_load_cubes`, która inicjuje zasilanie poszczególnych kostek. Na rysunku moduł ten jest reprezentowany przez element nr 4.

2.5. Wprowadzone zmiany w hurtowni

Wymienimy teraz czym różni się hurtownia systemu *Statystyki* od tej którą używa *Ocenarium*. Zmiany w głównej mierze wiążą się z dodaniem nowych tabel do schematu STAGE. Odpowiednie pliki źródłowe znajdują się w katalogu `/home/fg235984/STATYSTYKI/SQL/`. Są to

1. `01A_uprawnienia_do_tabel_ususa_dla_olap.sql` – nadanie uprawnień SELECT do tabel systemu USOS dla użytkownika bazodanowego OLAP. Skrypt wykonuje *adminrol* na bazie *ususbis*.

2. `01B_uprawnienia_dla_usera_stage.sql` – nadanie uprawnień do tworzenia widoków dla użytkownika bazodanowego STAGE na bazie *usoshurt*.

3. `02A_stage_tabele_sekwencje_synonimy.sql` – definicje nowych tabel i widoków (dla interfejsu). Skrypt wykonuje użytkownik STAGE na bazie *usoshurt*.

4. `04Adb_link_to_usos.sql` – link połączenia do źródłowej bazy danych *ususbis*.

5. `05A_STAGE.sql` – aktualizacja pakietu `usos_load` migrującego nowe tabele (zdefiniowane w pliku `02A_stage_tabele_sekwencje_synonimy.sql`)

Trzy ostatnie skrypty wykonuje użytkownik STAGE na bazie *usoshurt*.

Rozdział ten kończymy uwagą o tym, że *Ocenarium* korzysta z architektury hurtowni tylko do poziomu schematu STAGE. Dokładniej, nie wypełniamy struktur wielowymiarowych (element nr 5 rysunku). Związane jest to z tym, że naszym jedynym źródłem danych jest USOS i aktualnie nie są wykorzystywane właściwości integracyjne hurtowni. Z drugiej strony w implementacjach *raportów* wystarczające są zapytania odnoszące się do relacyjnej natury bazy danych USOS. Temat ten jest rozwinięty w rozdziale 2.6.

2.5.1. Nowe tabele w schemacie STAGE

Oprócz tabel z systemu *Statystyki Ocenarium* definiuje nowe tabele w schemacie STAGE. Ich lista znajduje się w dodatku B.

2.6. Motywacja użycia hurtowni danych w *Ocenarium*

W tym rozdziale uzasadnimy dlaczego mimo że wykorzystujemy wyłącznie dane z jednego źródła – z USOS-a – warto było skorzystać z hurtowni danych systemu *Statystyki*.

Pierwszym i zarazem oczywistym argumentem jest fakt, że nie powinniśmy powtarzać pracy już przez kogoś wykonanej. Tym bardziej że projekt *Statystyki* cały czas ewoluuje co wskazuje na to że ma duży potencjał i jest dobrze prowadzony.

Zauważmy że nie jest możliwe pozyskiwanie danych bezpośrednio z USOS-a. Wynika to z tego że skomplikowane i złożone operacje na bazie wykonywane przez *Ocenarium* mogłyby prowadzić do zagłodzenia użytkowników interakcyjnych – takich jak pracownicy dziekanatu, studenci czy pracownicy naukowi.

Ponadto, baza danych USOS jest migrowana co 24 godziny i korzystając z USOS-a nasza aplikacja byłaby uzależniona od tego czy jest on akurat zamknięty czy otwarty. Takich sytuacji chcemy uniknąć.

Alternatywnym rozwiązaniem mogłoby być "surowe" skopiowanie bazy USOS. To jednak ogranicza nas na dalszy rozwój, gdzie być może potrzebne będą analizy przekrojowe – korzystające z danych z kilku źródłowych baz danych. Wtedy – cechy hurtowni nie używane w aktualnej wersji *Ocenarium*, a ułatwiające integrację danych – mogą mieć duże znaczenie i decydować o efektywności całej aplikacji.

Reasumując, ważne jest zostawienie sobie możliwości rozwoju i otwarcia *Ocenarium*na nowe źródła danych, które nasza hurtownia wspiera. Mamy tu na myśli Internetową Rejestrację Kandydatów (IRK [14]).

2.7. *Ocenarium* vs *Statystyki*

W tym rozdziale pragniemy wyjaśnić skąd wziął się pomysł żeby nie rozwijać *Statystyk*, a rozpocząć własny, niezależny projekt korzystający jedynie z gotowej bazy danych. Chcielibyśmy wskazać kilka słabszych lub też niedostatecznie rozwiniętych części systemu *Statystyki*.

Piętą achillesową *Statystyk* jest ich część statystyczna i prezentacja danych. Prawdopodobnym powodem takiego stanu rzeczy są priorytety Twórców, którzy największy nacisk położyli na komunikację z bazą danych i interakcję z użytkownikiem poprzez bogaty interfejs. Być może bardziej zaawansowane prace statystyczne przewidziane są na późniejsze etapy. Niemniej jednak, bardziej zasadniczą przyczyną są zastosowane w projekcie biblioteki Javy, które nie dają tutaj dostatecznie dużych możliwości. W *Ocenarium* część statystyczna jest wykonana w R. W środowisku narzędzie to jest uważane za jednego z liderów. Więcej informacji o R w kontekście naszego projektu znajduje się w rozdziale 3.2.1.

Innym aspektem który ogranicza *Statystyki* jest brak możliwości wykonania bardziej złożonych analiz. Związane jest to z tym, że poza wcześniejszą agregacją na etapie wypełniania struktur hurtowni wszystkie zadania wykonywane są online. W *Ocenarium* ze względu na specyfikę *raportu* analizy przeprowadzane są offline i dzięki temu parametry wydajnościowe mają mniejszy priorytet.

Oczywiście można zapytać dlaczego nie warto było skupić się na poprawieniu wymienionych mankamentów w systemie pierwotnym. Otóż na liście priorytetów *Ocenarium* najwyżej znajdowało się przygotowanie różnorodnych *raportów* oraz środowiska do ich generowania. Z drugiej strony wprowadzanie poprawek do *Statystyk* mogłoby zająć zbyt dużo czasu i prowadzić do zaniedbania ważniejszych zadań. Dlatego też prościej było zrobić coś autonomicznego, od początku, i bez z góry narzuconych rozwiązań architektonicznych.

Rozdział 3

Architektura

W tym rozdziale przeanalizujemy budowę *Ocenarium*. Najpierw przyjrzymy się całej architekturze, a później przejdziemy do omówienia poszczególnych jej elementów. Diagram 3.1 przedstawia architekturę naszego systemu. Elementy o kolorach białym i niebieskim to komponenty *Ocenarium*. Natomiast zielone prostokąty to akcje które wiążą poszczególne komponenty.

3.1. Baza danych

Ze względu na znaczącą rolę i dużą objętość, sekcja poświęcona bazie danych *Ocenarium* została wyodrębniona w postaci osobnego rozdziału. Dlatego też po szczegóły odsyłamy czytelnika do rozdziału 2.

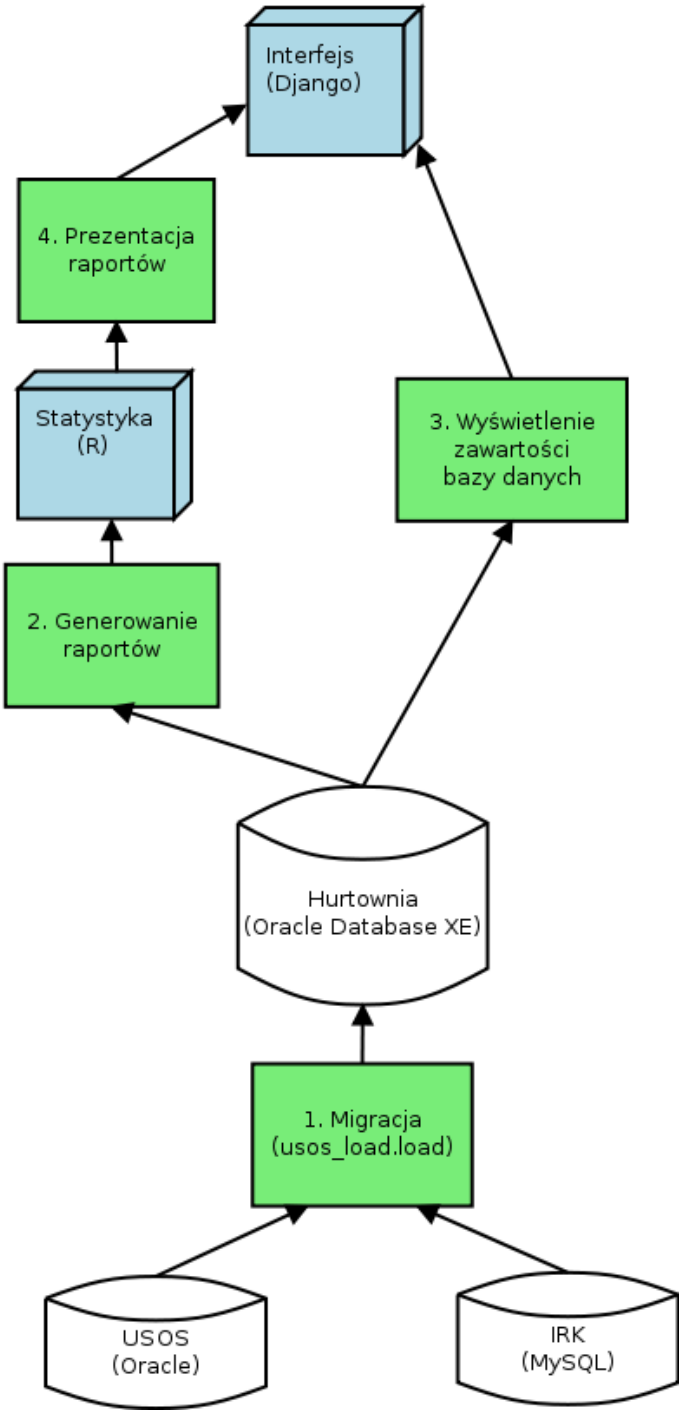
3.2. Część statystyczna – R

W tej sekcji omówimy ważne pakiety R, które zostały wykorzystane w pracy. Omówimy pokrótce ich funkcjonalności i sposób w jaki zostały w *Ocenarium* zastosowane. Najpierw jednak słowo wprowadzenia o języku R.

3.2.1. Język R

R jest darmowym środowiskiem programistycznym służącym do obliczeń statystycznych i wizualizacji danych. Kompiluje się i może być zainstalowane na wielu platformach UNIX-owych, a także systemach operacyjnych rodziny Windows i MacOS. Po więcej informacji można sięgnąć na stronę projektu [9] lub do przewodnika [11].

R posiada implementację wielu narzędzi statystycznych. Wśród nich są modele liniowe i nieliniowe, klasyczne testy statystyczne, analiza szeregów czasowych, czy klasteryzacja. Możliwe jest rozszerzanie R za pomocą dodatkowych pakietów oraz skryptów pisanych przez użytkownika.



Rysunek 3.1: Procesy i schemat przepływu w *Ocenarium*. Figury przestrzenne to komponenty systemu, zaś figury płaskie to akcje łączące poszczególne komponenty. Źródło: opracowanie własne.

Jednak największym atutem R jest jego silnik graficzny, uznawany w środowisku za najlepszy. Wykresy generowane są przez R w jakości nadającej się do publikacji. Wiele funkcji i metod rysujących wykresy posiada dobrze dobrane parametry, które często nie wymagają od użytkownika wielu modyfikacji. Z drugiej strony dostępne funkcje niskiego poziomu zapewniają całkowitą kontrolę nad grafiką.

3.2.2. R2HTML

Pakiet R2HTML pomaga eksportować obiekty R do plików HTML, dzięki czemu generowanie *raportów* staje się łatwiejsze i szybsze. Ciekawą cechą jest wykorzystanie funkcji `dev.print()`, która umożliwia zapisywanie obiektów (rysunków) *w locie*. Użytkownik może wtedy zapisać grafikę po wcześniejszym wyeksportowaniu jej do urządzenia X11. Takie rozwiązanie ma niewątpliwie duże walory dydaktyczny – student może zachować efekty swojej pracy w pliku. Dokumentacja i oficjalna strona pakietu są dostępne w [2]. Poniżej wymieniamy funkcje R2HTML stosowane w *Ocenarium*.

Funkcja	Opis
`HTML()`	Służy do zapisu obiektów tekstowych do pliku HTML. Stosujemy je do wstawiania tagów HTML, tekstów, a przede wszystkim tabel. Znanym pakietem który również eksportuje tabele do plików HTML jest pakiet `xtable` ([10]) – można go użyć jako alternatywę dla R2HTML.
`HTMLplot()`	Eksportuje grafikę z aktywnego urządzenia do formatu HTML. To tutaj użyta jest strategia zapisu *w locie*. Formatem pliku graficznego może być png, gif, bmp lub jpeg. Należy pamiętać, że aby móc wyeksportować grafikę trzeba uruchomić wcześniej X serwer na maszynie na której wykonujemy `HTMLplot()`.

3.2.3. *Ocenarium* – pakiet R

Powtarzający się w projekcie kod R został wydzielony w postaci pakietu o nazwie Ocenarium. W poniższej tabeli wymieniamy jego najważniejsze funkcje.

Funkcja	Opis
`readConfigFile()`	Wczytuje stałe z pliku konfiguracyjnego (ścieżka jest argumentem wejściowym). Są to parametry połączenia z bazą danych i inne współdzielone zmienne. Wynikiem zwracanym przez tę funkcję jest lista.
`executeSqlQuery()`	Wykonuje zapytanie na hurtowni danych. Jako parametry przyjmuje dane o połączeniu z bazą, zapytanie w postaci stringa i kilka bardziej technicznych argumentów. Zwraca w wyniku obiekt klasy `data.frame`. Podobną funkcjonalność zapewniają pakiety RODBC ([12]), czy ROracle ([13]).

Funkcja	Opis
`plotToHTML()`	Jest to modyfikacja metody `HTMLplot()` dostosowana do potrzeb *Ocenarium*. Działa podobnie do oryginału. Dodatkiem jest możliwość umieszczenia odnośnika związanego z pozycją w spisie treści raportu.

3.3. Interfejs – Django

Interfejs *Ocenarium*wykonany jest w technologii Django ([3]). Jest to wysokopoziomowe, opensource'owe (rozpowszechniane na licencji BSD) środowisko (framework) przeznaczony do tworzenia aplikacji internetowych. Django jest jednym z najbardziej rozpowszechnionych środowisk www, a swoją popularność zawdzięcza szybkości i łatwości z jaką można tworzyć nowe aplikacje. Napisany jest w Pythonie, zaś jego głównymi cechami są

1. Automatycznie generowany i bogaty panel administracyjny, z możliwością dalszego dostosowywania

2. Rozbudowany system autentykacji

3. Przyjazne adresy dokumentów z możliwością dowolnego ich kształtowania

4. Prosty lecz funkcjonalny system szablonów czytelny zarówno dla grafików jak i dla programistów

5. Oddzielenie logiki aplikacji (widok), logiki biznesowej (model), wyglądu (szablony) oraz baz danych. Django opiera się na wzorcu projektowym podobnym do MVC nazywanym MVT (Model-View-Template)

6. Bardzo duża skalowalność i wydajność

7. Możliwość uruchomienia w połączeniu z serwerem Apache za pomocą `mod_wsgi` (por. rozdz. 5.3.2) lub `mod_python`. Ma także możliwości współpracy z serwerami wspierającymi FastCGI

3.3.1. Budowa aplikacji Django

Aby uporządkować nieco pojęcia stosowane w dalszej części pracy przedstawimy ogólną budowę aplikacji Django. Po utworzeniu nowej aplikacji poleceniem `python manage.py startapp app` do korzenia naszego projektu dodany zostanie nowy katalog o nazwie `app/`. Będą się w nim znajdowały źródła aplikacji `app`. Dokładniej będą to pliki:

- `models.py` – plik z definicjami klas modelu aplikacji. Po zdefiniowaniu modelu Django udostępnia polecenie synchronizujące (`syncdb`), które analizuje modele wszystkich aplikacji i tworzy te tabele bazy danych, które jeszcze w niej nie istnieją. Powstaje w ten sposób tzw. model obiektowo-relacyjny (ORM).

- `views.py` – plik z definicjami widoków aplikacji. Każdy widok odpowiada za wykonanie jednej z dwóch rzeczy: zwrócenie obiektu HttpResponse zawierającego zawartość żądanej strony lub zgłoszenie wyjątku takiego jak Http404. To, co się będzie działo w jego wnętrzu jest decyzją programisty. Ogólnie rzecz biorąc widok pobiera dane zgodnie z przekazanymi parametrami, wczytuje szablon i renderuje go, wykorzystując pobrane dane

- `urls.py` – plik z adresami aplikacji. Stanowi on niejako spis zawartości aplikacji, zapewnia proste odwzorowanie między wzorcami URL i funkcjami Pythona (widokami)

Dodatkowo w korzeniu projektu znajdują się pliki

- `settings.py` – plik konfiguracyjny projektu. Zdefiniowane są w nim współdzielone stałe, informacje o bazie danych, zainstalowane aplikacje i inne globalne ustawienia

- `urls.py` – globalny plik z adresami aplikacji

3.3.2. Problemy z Django

Mimo wielu swoich szeroko opisanych zalet, Django posiada też dosyć dużą wadę o której chcielibyśmy wspomnieć. Jej istnienie ujawniło się dopiero w późniejszej fazie projektu. Chodzi mianowicie o prezentację danych w widoku. W wersji Django 1.2.1 możliwe jest w zasadzie tylko wyświetlanie rekordów pojedynczej tabeli. Ograniczając się do takich wariantów prezentacji danych mamy do dyspozycji bardzo wygodny system ORM (ang. object-relational mapper [22]). Zbędne w tym przypadku jest pisanie zapytań SQL (jak ma to miejsce np. w PHP). Jednak z drugiej strony chcąc wydrukować rekordy z tabeli która jest złączeniem innych pojawiają się kłopoty. Django nie daje takiej możliwości. Subiektywne propozycje wyjścia z tej sytuacji są dwie

1. Dodanie widoku w bazie danych który jest złączeniem tabel o które nam chodzi. Należy to zrobić w parze z dodaniem nowej klasy do modelu aplikacji Django.

2. Symulacja działania algorytmu złączania w SZBD i ręczne złączenie tabel na poziomie kodu pythona.

W *Ocenarium* zastosowane są oba wyżej wymienione podejścia. Pierwsze, w listingu prowadzących popularne przedmioty w zakładce *Prowadzący*. Drugie, w metodzie `lst(request, kierunek)` pliku `/home/users/fg235984/usosR/seminaria/views.py`.

Innym mankamentem Django jest brak obsługi tabel o kluczu złożonym. Jest to szczególnie dokuczliwe gdy działamy na zasadzie *reverse engineering*, tzn. chcemy połączyć nową aplikację Django z istniejącą już bazą danych i nie możemy dodać do tabeli nowej kolumny, która będzie kluczem (sztucznym kluczem). Tak też było w przypadku *Ocenarium*. Powodem takiego stanu rzeczy jest prawdopodobnie profil domyślnej aplikacji Django wg którego baza danych jest wypełniana danymi tylko poprzez aplikację – bez zewnętrznych źródeł danych. Takie założenie wymusza by baza danych była uruchamiana równocześnie z aplikacją.

3.4. Schemat przetwarzania

W tym rozdziale przyjrzymy się dokładniej w jaki sposób funkcjonuje *Ocenarium*. Skupiamy się na miejscach styku pomiędzy poszczególnymi komponentami systemu oraz sposobie w jaki przebiega komunikacja. Punktem odniesienia będzie rysunek 3.1 i jego zielone elementy (oznaczone numerami 1-4). Kolejność w jakiej omówimy rysunek to *od dołu do góry*.

3.4.1. Styk źródłowych baz danych z hurtownią danych. Migracja

Ponownie odsyłamy czytelnika do osobnej sekcji (patrz 2.4.3), w której temat zasilania hurtowni bazami źródłowymi został wyczerpany. Na rysunku 3.1 jest to element z numerem 1.

3.4.2. Styk hurtowni danych z modułem statystycznym. Generowanie *raportów*

Przedstawimy teraz ogólny algorytm generowania *raportu* przez *Ocenarium* (element nr 2 na rysunku 3.1). Schemat ten w praktyce jest zastosowany w rozdziale A.2 i może on służyć jako lektura uzupełniająca. Poniższe kroki dotyczą skryptu R.

1. Ustal katalog wyjściowy dla plików wynikowych (pliki HTML, grafiki)
2. Pobierz z pliku konfiguracyjnego dane połączenia do bazy danych
3. Za pomocą `executeSqlQuery()` pobierz z bazy danych dane prezentowane w *raporcie*
4. Na podstawie pobranych danych wygeneruj elementy *raportu* (grafiki, tekst, tabele) i zapisz je w wynikowym pliku HTML

Dwie kolejne sekcje dotyczą w gruncie rzeczy implementacji plików `views.py` w każdej z aplikacji Django wchodzących w skład *Ocenarium*.

3.4.3. Styk hurtowni danych z interfejsem. Wyświetlenie zawartości bazy danych

Omawiany element jest oznaczony numerem 3 na rysunku 3.1.

Jednym z dwóch typów widoków w *Ocenarium* jest wyświetlenie wycinka bazy danych. Może to być np. lista przedmiotów proponowanych na Wydziale MIM UW lub lista seminariów magisterskich. Nakreślimy pokrótce w jaki sposób jest to rozwiązane. Po pierwsze, posługujemy się w tym przypadku szablonami oferowanymi przez środowisko Django. Aby nie odsyłać czytelnika do zewnętrznej dokumentacji Django przybliżymy tutaj koncept szablonów.

Szablon to zwykły plik tekstowy zawierający zazwyczaj kod HTML oraz tagi – zmienne, za które zostanie wstawiona określona treść bądź też tagi kontrolujące logikę

szablonu. Szablon definiuje ogólny wygląd witryny, a także wskazuje *dziury* do wypełnienia przez szablony potomne. Takie rozwiązanie znacząco upraszcza zmianę wyglądu witryny, bo w wielu sytuacjach wystarczy zmienić tylko jeden plik – szablon bazowy. W razie wątpliwości dodatkowych informacji można zasięgnąć w dokumentacji Django w rozdziale poświęconym szablonom ([4]).

Możemy już wrócić do opisu kroków algorytmu wyświetlania danych z bazy danych w interfejsie. Pierwszą czynnością jest zdefiniowanie w modelu aplikacji Django (plik `models.py`) klasy odpowiadającej wyświetlanym rekordom tabel. Dzięki temu po zaimportowaniu tych klas w pliku `views.py` możemy za pomocą ORM (ang. Object-relational mapping) Django pobrać listę (kolekcję) obiektów. Pomocne są tutaj filtry i możliwość sortowania listy oferowane przez ORM. Następnie wysyłamy listę obiektów jako parametr do przygotowanego szablonu. Ten zaś ją formatuje i jako efekt końcowy otrzymujemy wyświetlone w interfejsie dane.

Po szczegółowe instrukcje odsyłamy do rozdziału A.3, gdzie wszystkie czynności są wykonane na przykładzie.

3.4.4. Styk modułu statystycznego z interfejsem

Na rysunku 3.1 jest to element numer 4.

Drugim wariantem widoku w *Ocenarium* jest widok prezentujący *raport* w interfejsie. Na tym etapie przetwarzania mamy już gotowe *raporty* w postaci dokumentów HTML. Zatem to co pozostaje zrobić to wczytać w widoku aplikacji (plik `views.py`) odpowiadającej *raportowi* (patrz C.1) zawartość pliku HTML. Jedyną korektą jakiej należy dokonać jest konwersja ścieżek na adresy bezwzględne. W tym celu korzystamy ze stałej `MEDIA_URL` zdefiniowanej w pliku konfiguracyjnym `settings.py`. Dla przykładu, przy konfiguracji z rozdziału 5 ścieżka `etapy1_inf.png` z raportu `/home/users/fg235984/usosR/static_files/student_html/students_inf.html` jest podmieniana na {{ `MEDIA_URL` }}`student_html/`).

Przyczyną dla której ścieżki nie są ustalane w ostatecznej formie już na poziomie kodu R jest sposób w jaki pakiet `R2HTML` wstawia grafiki do pliku HTML. Dokładniej, umieszcza on grafiki w katalogu bieżącym i do tej lokalizacji się odwołuje wstawiając tagi `<img src>`. Zamiana ścieżki na bezwzględną rozwiązuje ten problem.

Rozdział 4

Przygotowane *raporty*

W tym rozdziale przedstawimy wszystkie sporządzone w *Ocenarium raporty*. Ich kolejność wynika z kolejności pozycji menu w interfejsie użytkownika. Każdy rozdział dotyczy jednego *raportu* i z reguły składa się z dwóch podrozdziałów. Pierwszy jest wprowadzeniem, drugi zaś przedstawia szczegóły *raportu*.

4.1. Przedmioty

4.1.1. Omówienie *raportu*

Raport Przedmioty omawia oceny zdobywane przez studentów z poszczególnych przedmiotów oferowanych przez Wydział Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. W sumie było 1305 przedmiotów (stan na czerwiec 2011). W raporcie uwzględniane są informacje o liczności i średnich ocen w podziale na cykle dydaktyczne, terminy egzaminów i płeć. Podane są rozkłady ocen w kolejnych cyklach dydaktycznych. Analizujemy też średnie ocen ze względu na płeć i wpływ czynnika płci na ocenę (test Chi kwadrat). Jako ciekawostkę podajemy rozkład ocen w zależności od miesiąca urodzenia studenta. *Raport* może być użyteczny dla studentów, kadry dydaktycznej i komisji dydaktycznej. Wykresami wykorzystanymi w tej sekcji są `boxplot`, `barplot` i wykres interakcji (`interaction.plot`).

Implementacja tego *raportu* znajduje się w pliku `/home/users/fg235984/usosR/R/przedmiot/przedmiot_details.R`.

4.1.2. Szczegóły

Zacznijmy od przedstawienia wycinka bazy danych eksponowanego przez *raport Przedmioty*. Taki wstęp pomoże dodatkowo w wyjaśnieniu dlaczego zapytania do bazy danych są tak złożone.

Na rysunku 4.1 znajduje się fragment diagramu UML pod tytułem *Rozliczanie studentów: oceny i protokoły – ER diagram* (więcej w dokumentacji bazy danych USOS [15]). Pokazuje on ile i jakie tabele należy złączyć aby otrzymać listę ocen zdobywanych przez określonych studentów na ustalonych przedmiotach. W przypadku listy ocen są to

tabele `dz_oceny`, `dz_wartosci_ocen`, `dz_protokoly`, `dz_osoby`, `dz_przedmioty`. Zwróćmy jeszcze uwagę że w naszej hurtowni, która zawiera kopie tabel źródłowych sytuacja jest analogiczna z kosmetyczną zmianą dotyczącą nazw wszystkich tabel. Zaczynają się one bowiem prefiksem `tmp`, nie zaś `dz`.

Przechodzimy do szczegółowego omówienia elementów *raportu Przedmioty*. W tym celu jako przykład posłuży nam przedmiot *Bazy danych (1000-213aBAD)*. Patrząc okiem statystyka najważniejszym produktem i aspektem pozyskiwania danych jest ich ostateczny format jako tabeli/obiektu klasy `data.frame` w programie R. Jest to też punkt startowy pracy statystycznej. Dlatego poniżej podajemy 5 początkowych wierszy analizowanego zbioru danych. Dla zapewnienia anonimowości daty urodzenia zostały zastąpione przez symbol `YYYY-MM-DD`.

os_id	plec	data_ur	cdyd_kod	prz_kod	nazwa	jed_org_kod	tpro_kod	term_nr	wartosc
2316	M	YYYY-MM-DD	2004Z	1000-213aBAD	Bazy danych	10000000	EGZ	1	4.0
2382	M	YYYY-MM-DD	2004Z	1000-213aBAD	Bazy danych	10000000	EGZ	1	5.0
2426	M	YYYY-MM-DD	2004Z	1000-213aBAD	Bazy danych	10000000	EGZ	1	3.0
2472	M	YYYY-MM-DD	2004Z	1000-213aBAD	Bazy danych	10000000	EGZ	1	5.0
4169	M	YYYY-MM-DD	2004Z	1000-213aBAD	Bazy danych	10000000	EGZ	1	3.0

Mając taki zbiór danych możemy oprócz standardowych analiz rozkładu ocen przyjrzeć się również wpływowi efektu płci, czy daty urodzenia na ocenę z przedmiotu.

Na rysunku 4.2 widzimy początkową część tego *raportu*, łącznie z menu głównym *Ocenarium*. Aby przejść do *raportu* przedmiotu *Bazy danych* należy wejść w zakładkę *Przedmioty* i z listy przedmiotów wybrać ten którego szukamy. Jak widać na obrazku po spisie treści, w pierwszej kolejności wypunktowane są *statystyki opisowe*.

Dalej (rysunek 4.3), mamy rozkład ocen w ujęciu procentowym w poszczególnych cyklach dydaktycznych. Użytym wykresem jest w tym przypadku `barplot`.

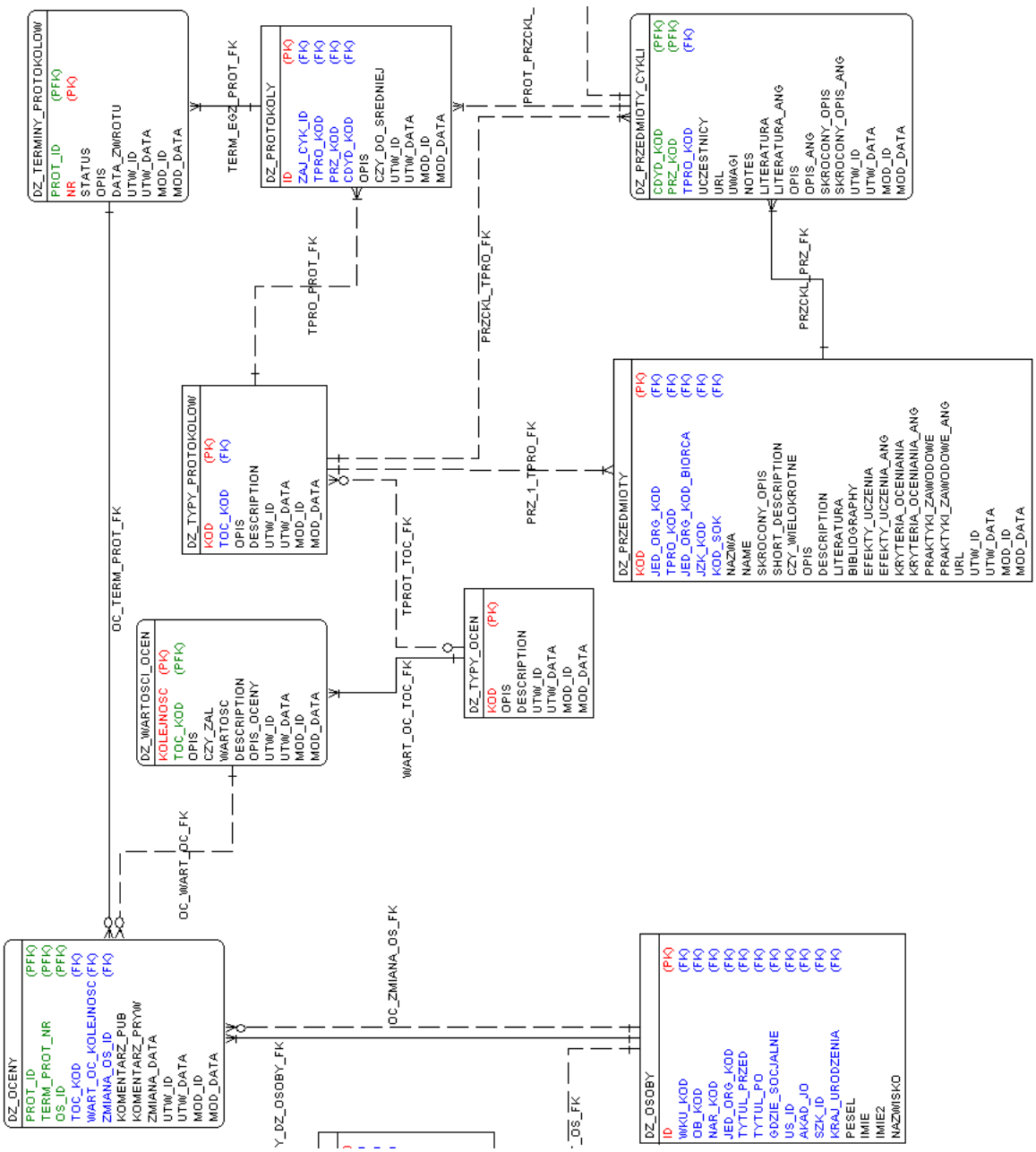
W kolejnej sekcji widzimy zestawienie średnich ocen w podziale na płeć (rysunek 4.4). Wykonano też test Chi kwadrat sprawdzający istotność efektu płci na ocenę z przedmiotu. Ta część zakończona jest tabelą na podstawie której został wygenerowany wykres interakcji (`interaction.plot` z pakietu `stats`).

Ostatnim prezentowanym wykresem (rysunek 4.5) jest `boxplot` z ocenami studentów urodzonych w poszczególnych miesiącach. Tak jak zaznaczyliśmy we wstępie należy traktować go jako ciekawostkę. Nie jest to wykres o wielkich walorach naukowych. Chodziło o sprawdzenie hipotezy o tym że studenci urodzeniu w letnich miesiącach radzą sobie lepiej od pozostałych.

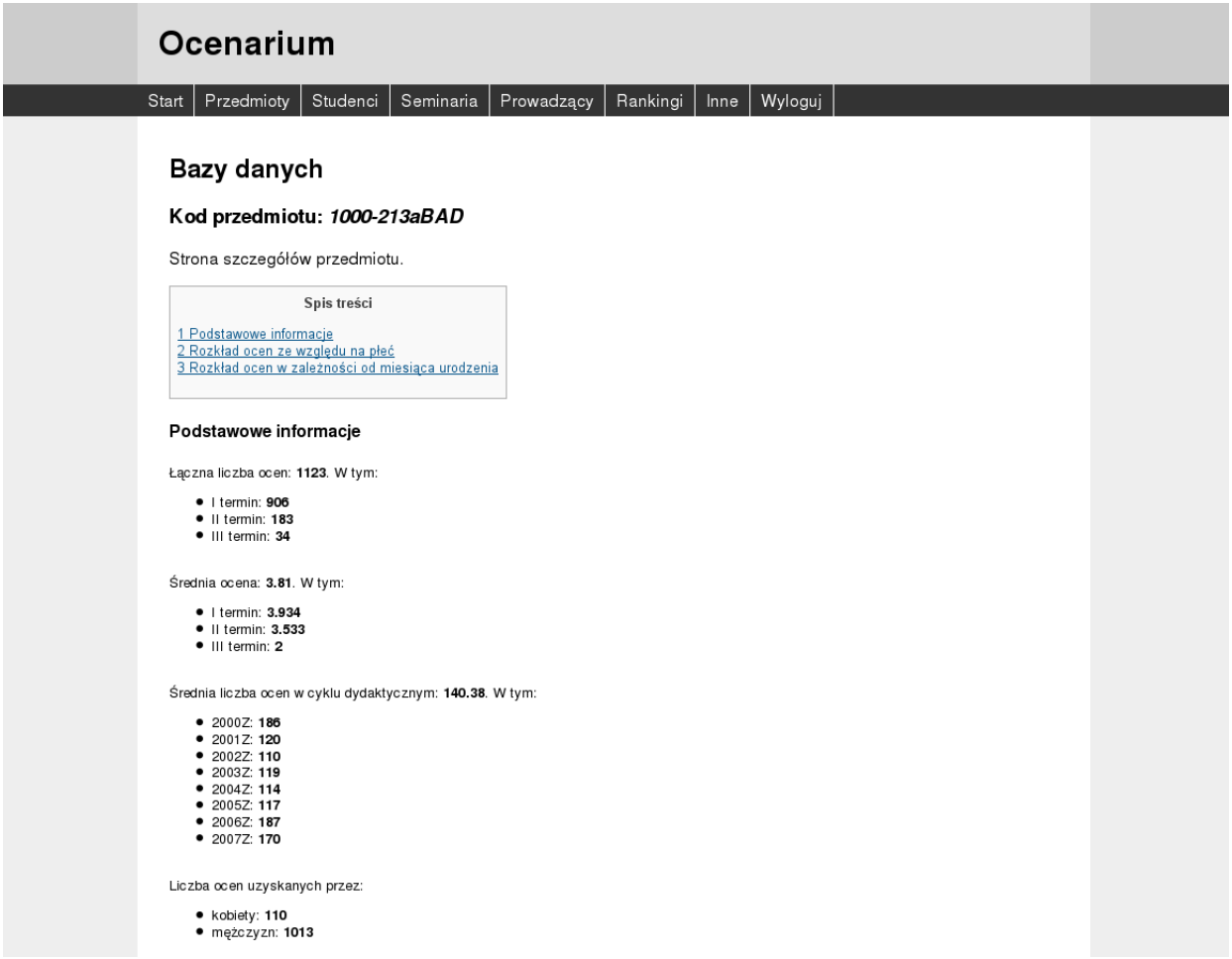
4.2. Studenci

4.2.1. Omówienie *raportu*

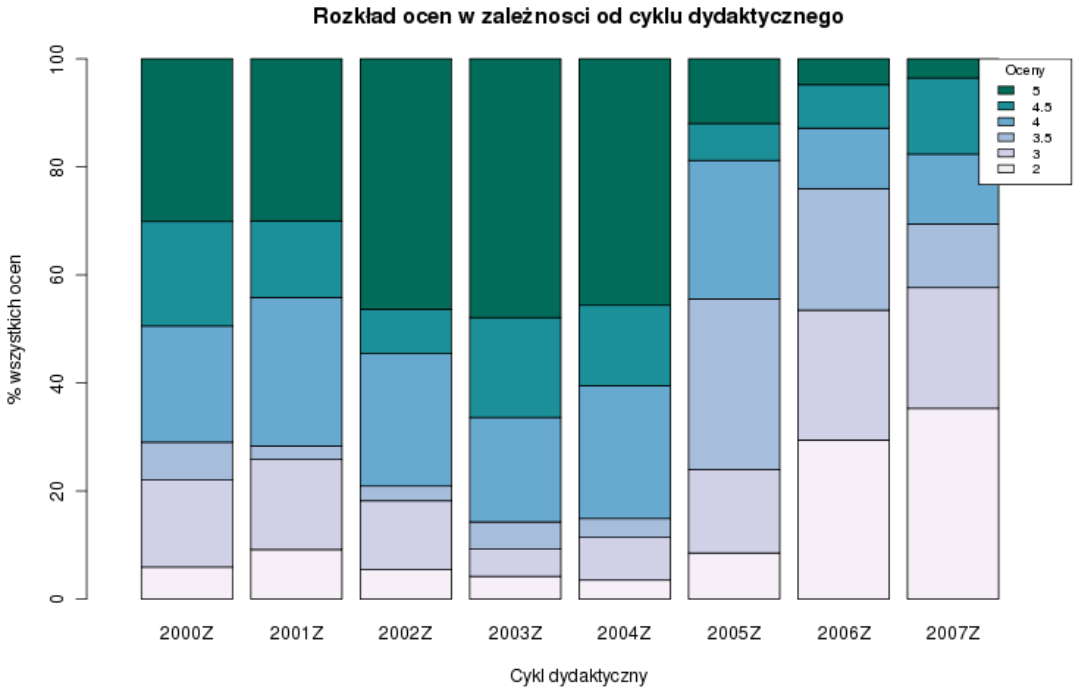
Raport Studenci został dodany z myślą o Komisji Dydaktycznej Wydziału Matematyki, Informatyki i Mechaniki. Grono to było pierwszym potencjalnym beneficjentem powstania *Ocenarium* i jedną z podstawowych motywacji do rozpoczęcia projektu. Warto też



Rysunek 4.1: Rozliczanie studentów: oceny i protokoły – ER diagram. Źródło: dokumentacja bazy danych USOS ([15]).

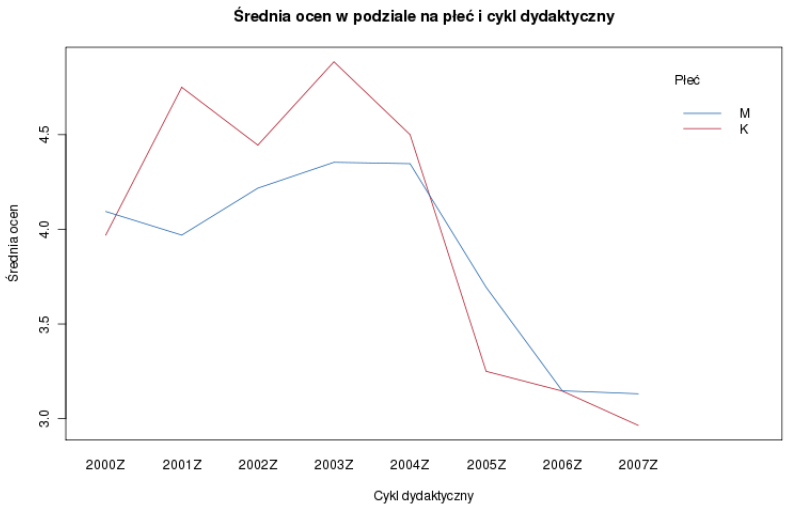


Rysunek 4.2: Raport *Przedmioty* dla przedmiotu *Bazy danych*. Źródło: opracowanie własne.



Rysunek 4.3: Zależność między rozkładem ocen a cyklem dydaktycznym na przedmiocie *Bazy danych*. Źródło: opracowanie własne.

Rozkład ocen ze względu na płeć

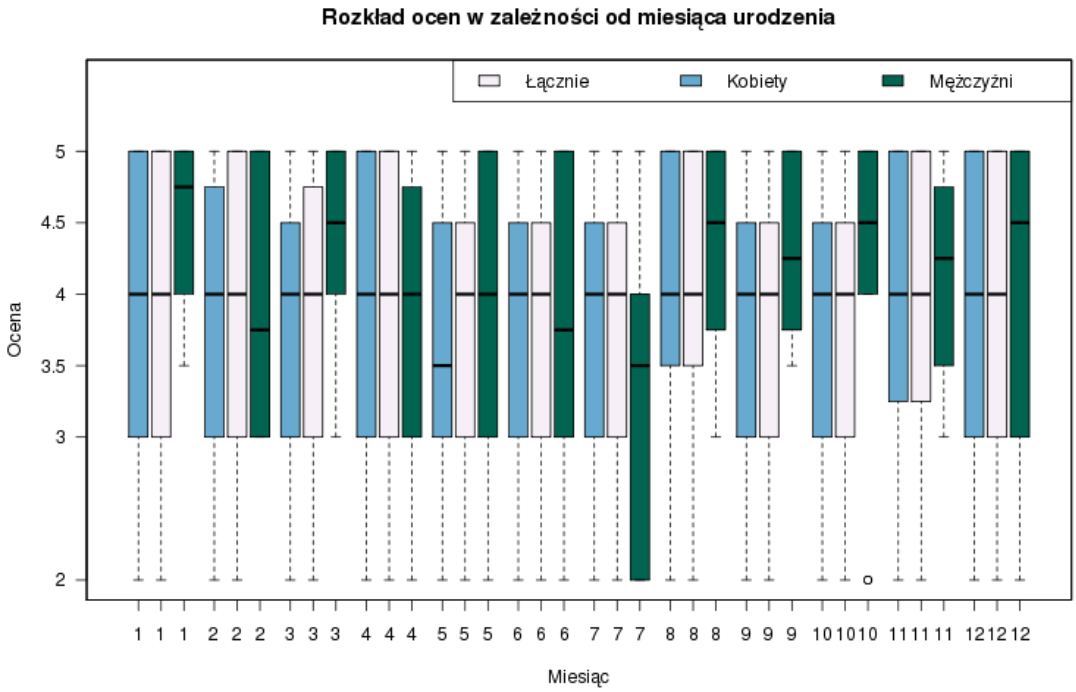


Płeć ma wpływ na ocenę z tego przedmiotu (Test Chi kwadrat):

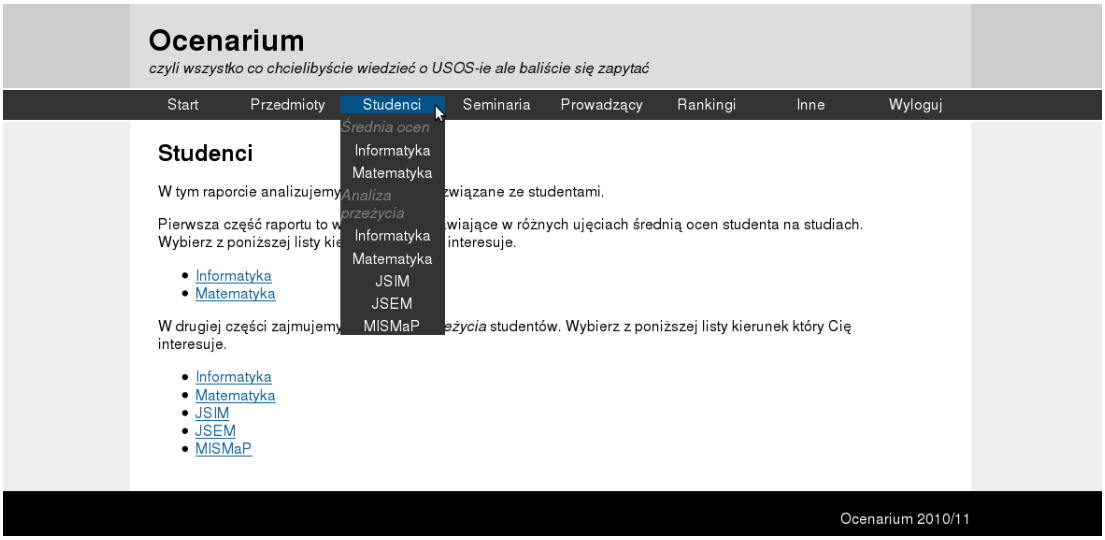
Średnie ocen w poszczególnych cyklach dydaktycznych.
NA oznacza brak danych (*not available*).

	Kobiety	Mężczyźni	Łącznie
2000Z	3.969	4.094	4.083
2001Z	4.75	3.969	4.008
2002Z	4.444	4.217	4.255
2003Z	4.885	4.354	4.412
2004Z	4.5	4.347	4.364
2005Z	3.25	3.694	3.671
2006Z	3.146	3.147	3.147
2007Z	2.964	3.131	3.118

Rysunek 4.4: Zależność między oceną średnią a płcią. Źródło: opracowanie własne.



Rysunek 4.5: Zależność między miesiącem urodzenia a oceną z przedmiotu. Źródło: opracowanie własne.



Rysunek 4.6: Strona główna *raportów* o studentach. Źródło: opracowanie własne.

wspomnieć, że w początkowej fazie *Ocenarium* – określanej potocznie wersją demo – obok dr. Przemysława Biecka, to Komisja Dydaktyczny była pierwszym recenzentem pracy. Ostatecznie *raport* o studentach nie został ukończony w takim wymiarze jakiego oczekiwała Komisja. W trakcie prac zmieniły się nieco priorytety i ustąpił on miejsca *raportowi Seminaria* (por. rozdz. 4.3).

Wizualnie, *raport* składa się z dwóch części. Pierwsza, podaje statystyki dotyczące średniej ocen na studiach. Druga to *analiza przeżycia* studentów, czyli przedstawienie zaliczeń poszczególnych etapów studiów w czasie. Obie części dzielą się ze względu na kierunek studiów.

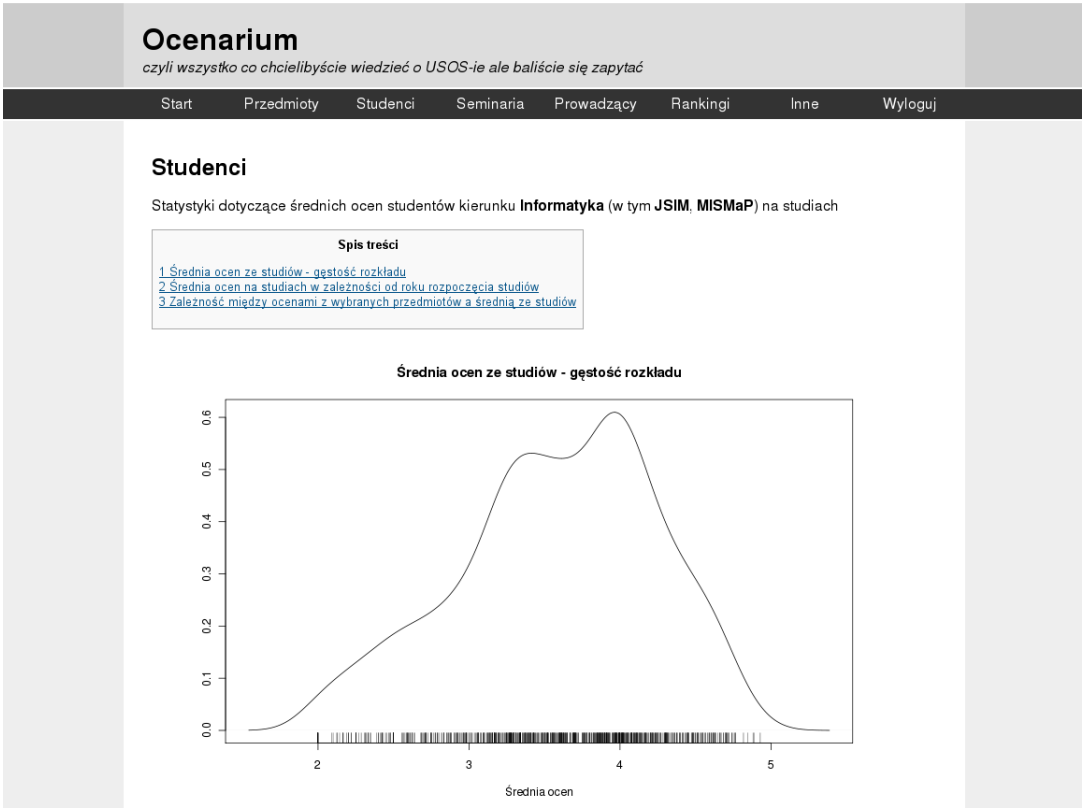
Plikami źródłowymi dla wymienionych *raportów* są odpowiednio `student_srednia.R` oraz `student_details.R` w katalogu `/home/users/fg235984/usosR/R/student/`.

4.2.2. Szczegóły

Na rysunku 4.6 widzimy stronę główną *raportu Studenci*. Jej zawartość została właściwie opisana w poprzedniej sekcji. Dlatego przejdziemy od razu do omówienia obu wspomnianych wyżej części.

Średnia ocen

Wybierając w menu *Studenci* → *Średnia ocen* → *Informatyka* przechodzimy do części poświęconej analizie średniej ocen studenta na studiach. Przyglądamy się tutaj kierunkom informatycznym (Informatyka, MISMaP, JSIM). Do wyboru są też kierunki Matematyczne (Matematyka, MISMaP, JSEM). Średnią ocen obliczamy jako średnią ze wszystkich



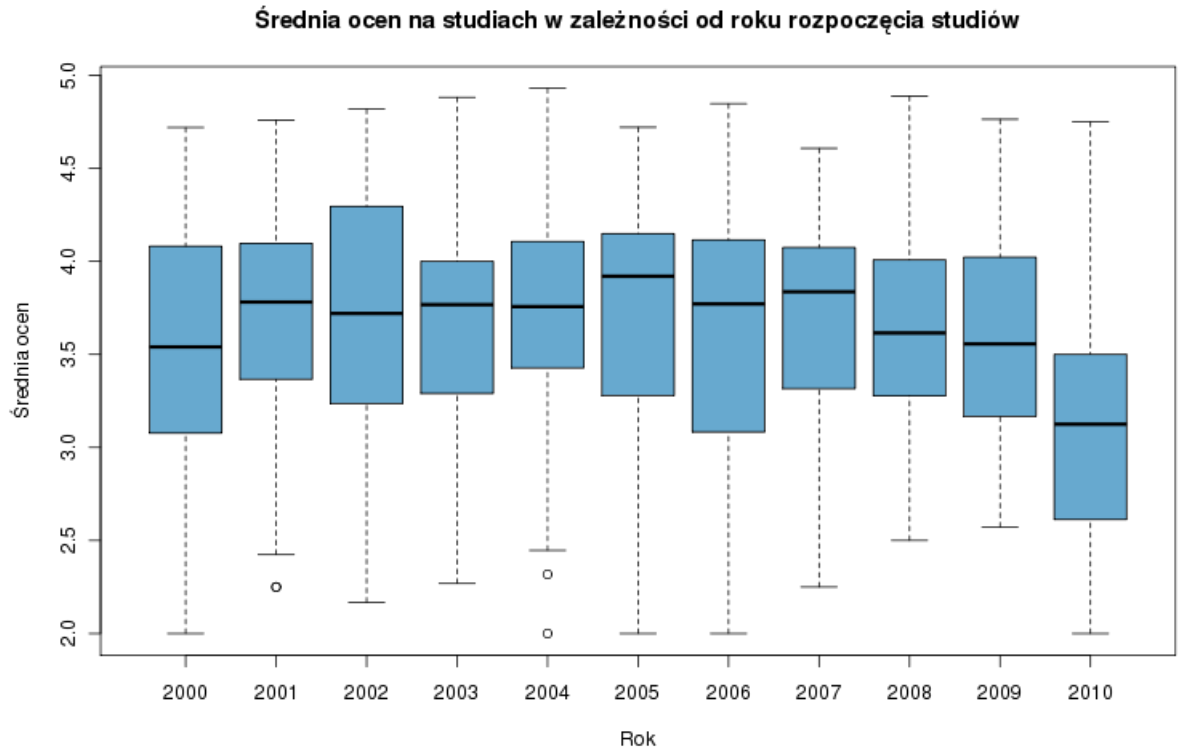
Rysunek 4.7: *Średnia ocen* w *raporcie* o studentach. Źródło: opracowanie własne.

ocen i wszystkich terminów egzaminów. Zaznaczmy, że nie ma tutaj żadnej wypracowanej umowy. Na Wydziale MIM UW również się to zmienia, a obliczanie średniej do dyplomu i do stypendium może się różnić.

Na początku (rysunek 4.7) widzimy rozkład średniej oceny studenta na studiach w postaci gęstości. Na osi X naniesione są wartości punktowe tej zmiennej (funkcja `rug()` w R).

W dalszej kolejności, na rysunku 4.8 przedstawiamy średnią ocen w podziale na rok rozpoczęcia studiów studenta. Dla młodszych roczników (2006-2010), z przyczyn naturalnych są to średnie tylko z dotychczasowych lat studiów. Rok rozpoczęcia studiów obliczamy na podstawie cyklu dydaktycznego pierwszego egzaminu do którego przystąpił student.

Na następnym wykresie interesujemy się informacją o tym jak przekładają się oceny z poszczególnych, pojedynczych przedmiotów na średnią ocenę na studiach. Wybraliśmy w tym celu cztery przedmioty: *Analizę matematyczną inf. I*, *Programowanie obiektowe*, *Wstęp do programowania* i *Algorytmy i struktury danych*. Z prezentowanych na rysunku 4.9 wyników widać dosyć dużą korelację pomiędzy dwiema analizowanymi zmiennymi.



Rysunek 4.8: Zależność między *Średnią ocen* a rokiem rozpoczęcia studiów. Źródło: opracowanie własne.

Celem takiego wykresu jest – oprócz ciekawości – znalezienie przedmiotu, który determinowałby jak z nauką w ogólności radzi sobie dany student. Zatem – wyrażając się obrazowo – naszym zadaniem jest znalezienie *papierka lakmusowego* oceniającego studentów.

Z technicznego punktu widzenia dwa ostatnie wykresy są narysowane za pomocą funkcji `boxplot()`, a kluczowa praca jest wykonana w zapytaniu SQL, które w pierwszym przypadku polega na złączeniu (po kolumnie `os_id` – identyfikator osoby) wyników zapytań *Średnia ocen* i *Początek studiów*, zaś w drugim *Średnia ocen* i *Ocena z pojedynczego przedmiotu*.

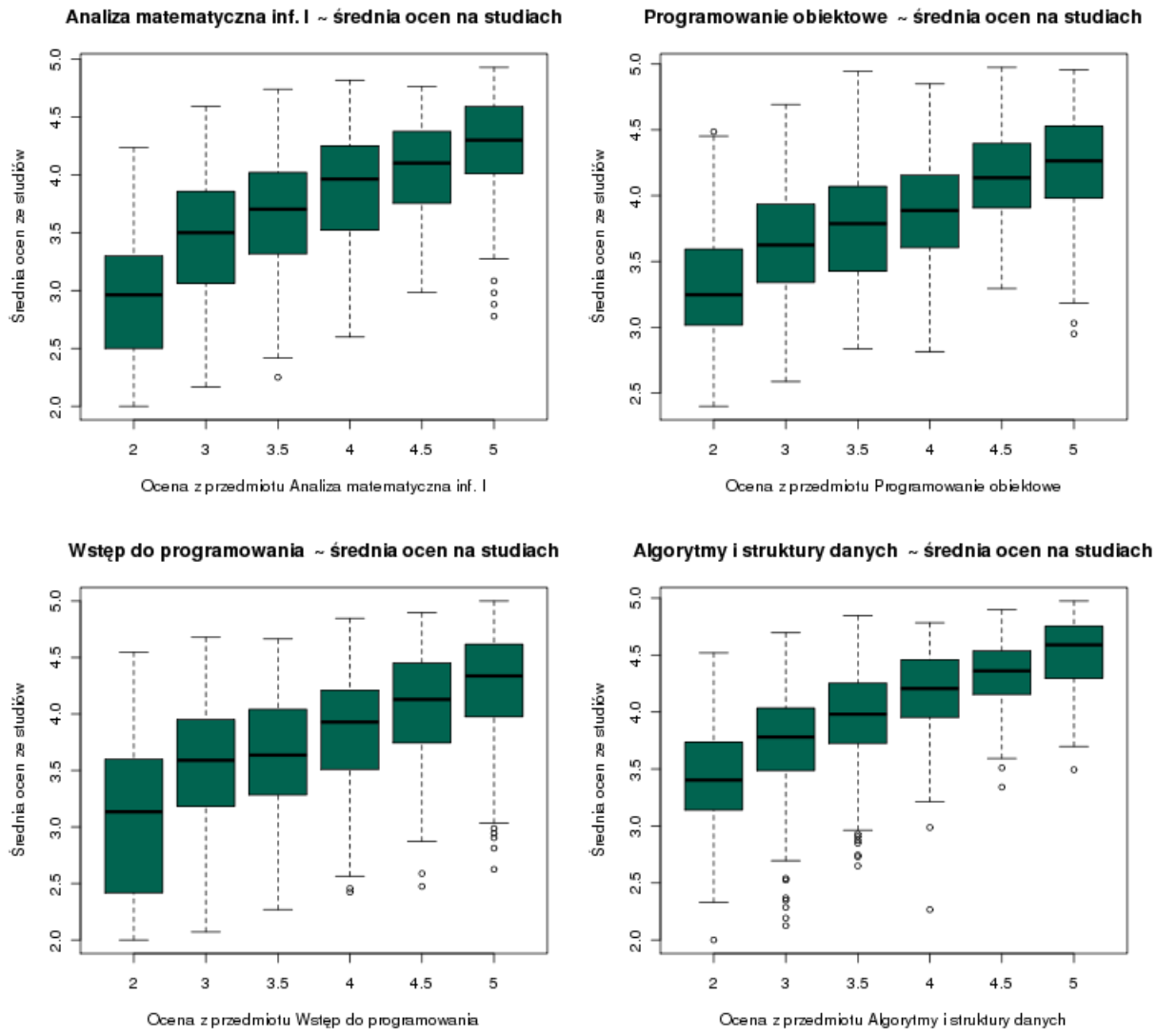
Analiza przeżycia

Zajmiemy się teraz *analizą przeżycia* studentów Wydziału MIM UW. Po kliknięci w menu głównym *raportu Studenci* zakładki *Studenci* → *Analiza przeżycia* → *Informatyka* widzimy to samo co na rysunku 4.10. Studenci którzy są przedmiotem badań pochodzą z rocznika 2007, tzn. zaczynali studia w semestrze zimowym 2007/08. Wybraliśmy akurat ten rocznik, ponieważ jest to pierwszy rocznik, który funkcjonuje wg nowego programu studiów. Oprócz tego jest nim bardzo zainteresowana Komisja Dydaktyczna. Zaznaczmy jeszcze, że interesują nas losy studentów na studiach pierwszego stopnia.

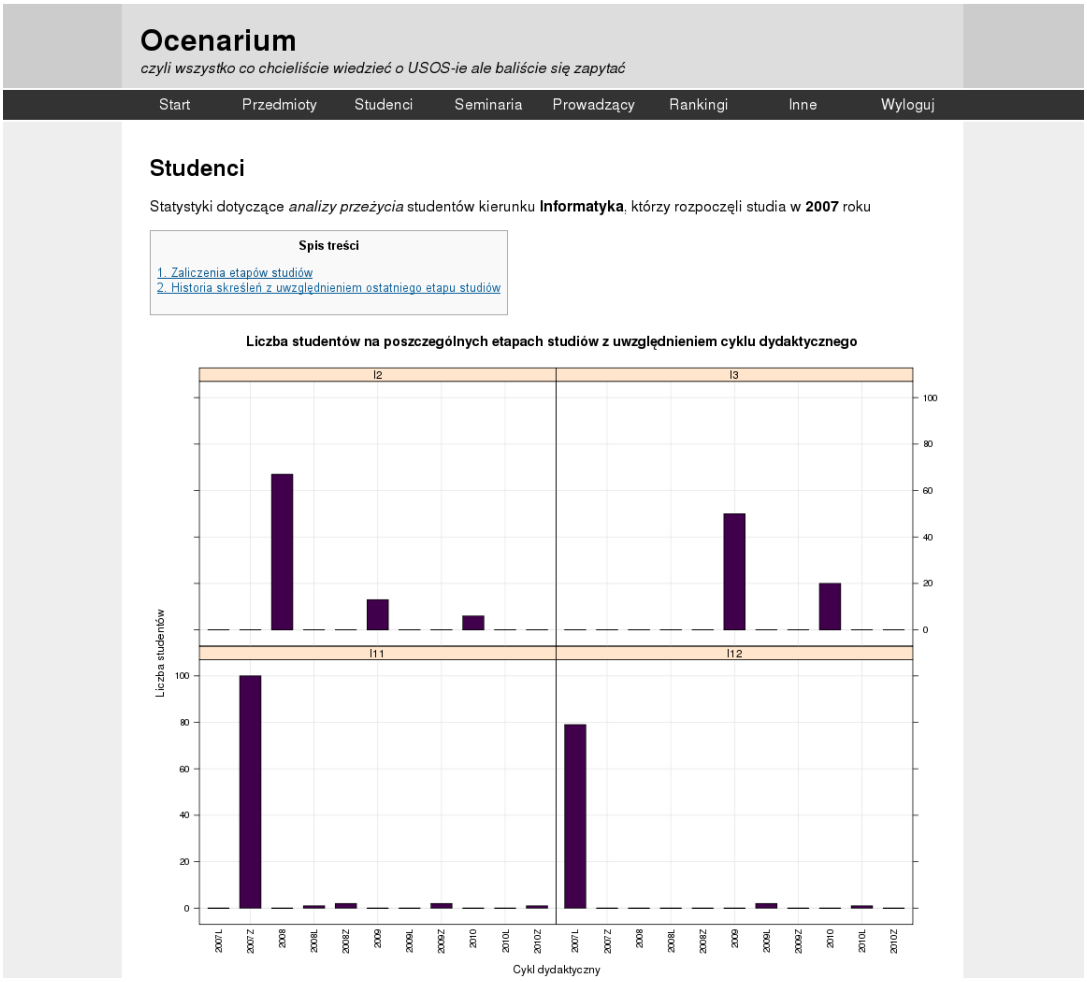
Dwa pierwsze wykresy (4.10 i 4.11) pokazują jak wyglądają zaliczenia poszczególnych etapów studiów w czasie. Widzimy na nich liczebności studentów na poszczególnych etapach studiów. Zauważmy (rysunek 4.11), że na pierwszym semestrze (etap `I11`) 21 studentów nie wywiązuje się ze swoich obowiązków. Na drugim semestrze (`I12`) jest ich 12, zaś na drugim roku (`I2`) 17. Można stąd wywnioskować, że biorąc pod uwagę studia pierwszego stopnia połowa studentów Informatyki z rocznika 2007 nie realizuje ich programu w planowym tempie. Ostatni z wykresów o zaliczeniach etapów (4.12) skupia się na pokazaniu liczności grup w ramach poszczególnych etapów. Tym razem bez uwzględniania czasu. Bierzemy pod uwagę wszystkie etapy jakie student przechodził w trakcie studiów, łącznie z ich powtarzaniem. Widać z niego, że po pierwszym zaliczonym semestrze studenci mają dużo chęci aby kontynuować studia. Inną obserwacją jest pewna anomalia, która polega na tym, że więcej osób było na etapie `I2`, niż `I12`. Wynika, to z tego że część studentów kierunku JSIM rezygnuje ze studiów matematycznych i od drugiego roku kontynuuje już tylko Informatykę. Innym wytłumaczeniem może być to że więcej osób powtarza etap `I2`, niż `I12`.

Jeśli chodzi o szczegóły implementacyjne, to użyty został tu pakiet *lattice* za pomocą którego narysowany jest pierwszy i trzeci wykres (`barchart()`). Drugi z wykresów to oczywiście `barplot`.

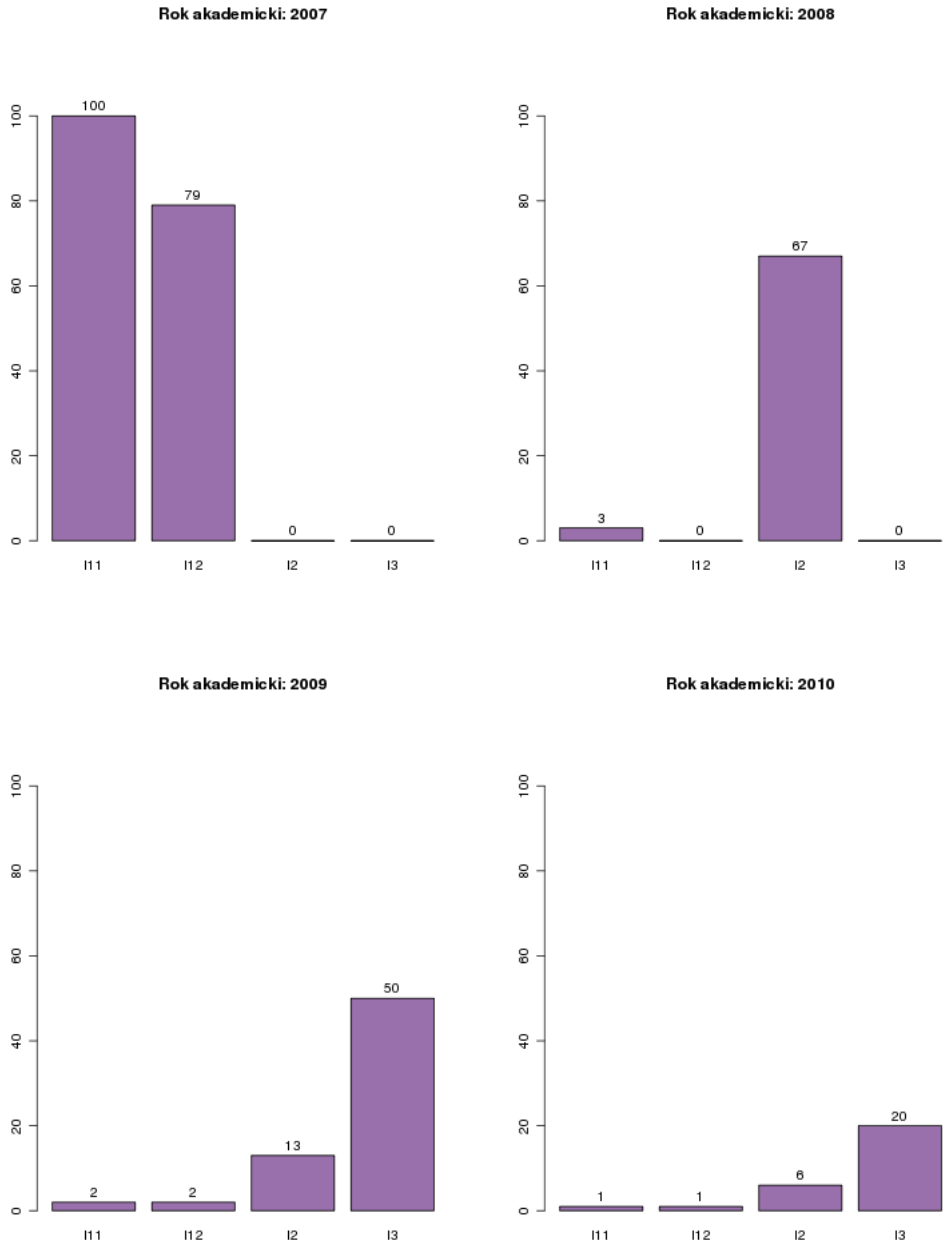
W bazie danych USOS istnieje tabela `historia_skr`, która jak nazwa wskazuje przechowuje informacje o skreśleniach z listy studentów. Tematem następnego wykresu są właśnie dane z tej tabeli. Jest on pokazany na rysunku 4.13, a informacje tam uwzględnione dotyczą daty skreślenia oraz etapu studiów do którego doszedł student. Widać wyraźnie że skreślenia kumulują się głownie w okresach następujących po sesjach egzaminacyjnych. Oprócz tego im dalszy etap studiów tym dynamika liczby skreśleń maleje. Funkcją użytą do narysowania tego wykresu jest `xyplot()` z pakietu *lattice*.



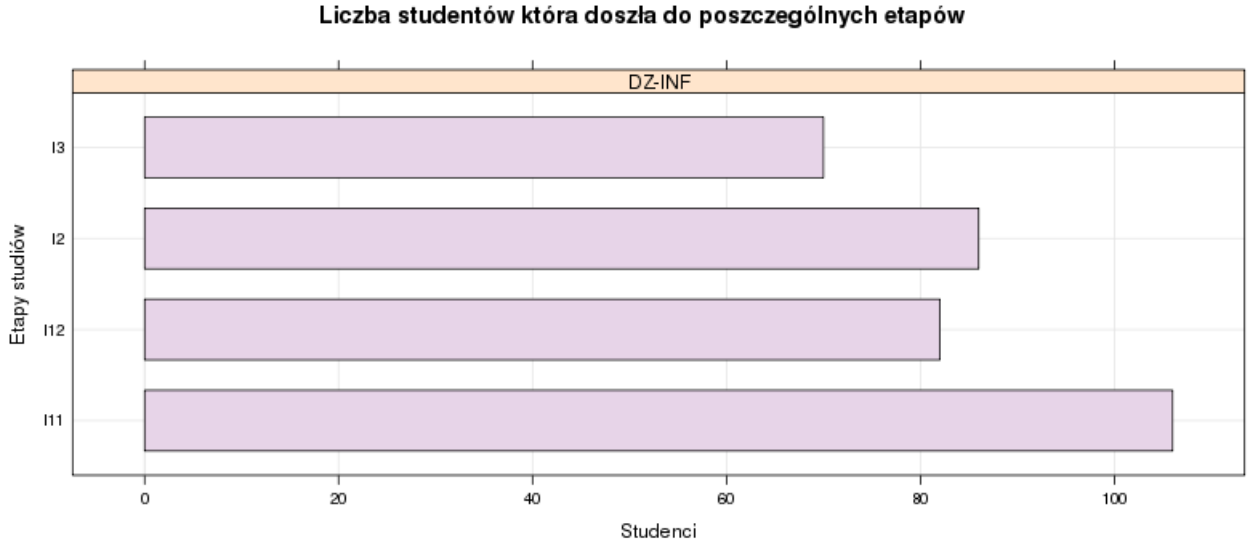
Rysunek 4.9: Zależność między oceną z pojedynczego przedmiotu, a *Średnią ocen* na studiach. Źródło: opracowanie własne.



Rysunek 4.10: Zaliczanie etapów studiów w czasie, cz. I. Źródło: opracowanie własne.



Rysunek 4.11: Zaliczanie etapów studiów w czasie, cz. II. Źródło: opracowanie własne.



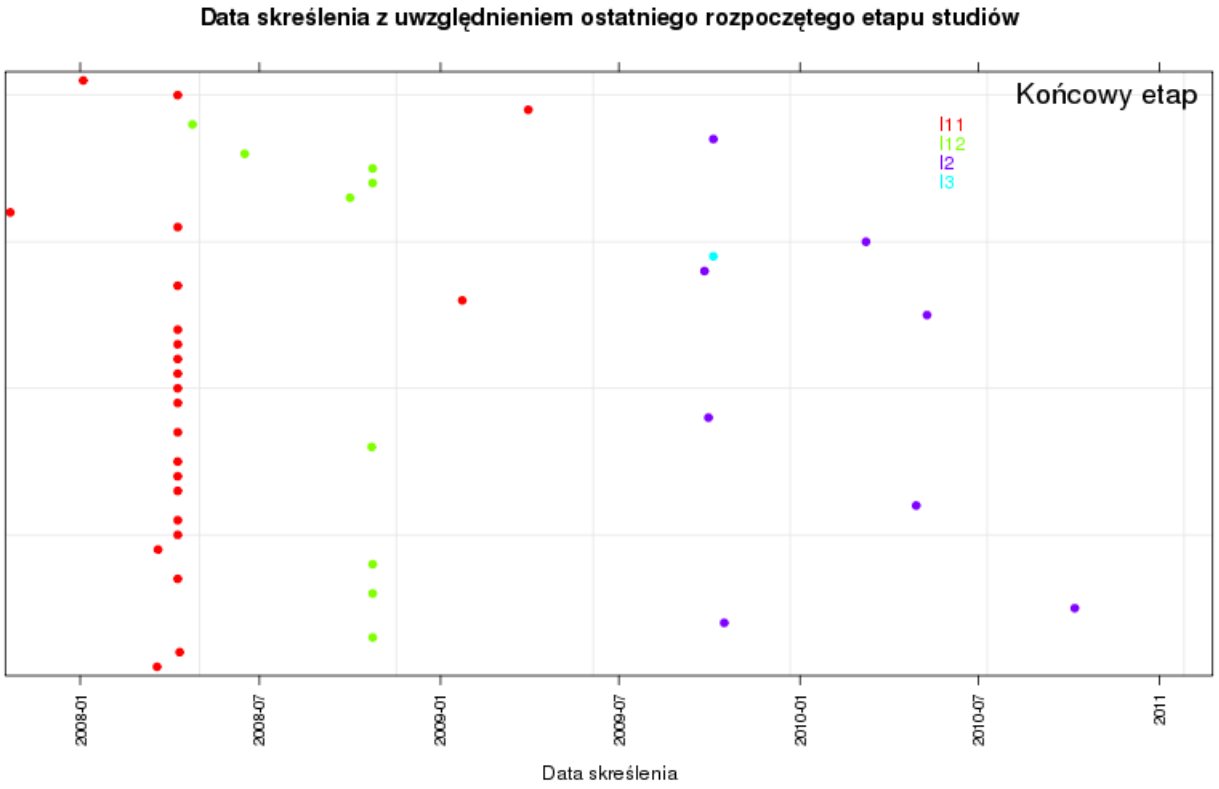
Rysunek 4.12: Zaliczanie etapów studiów, cz. III. Źródło: opracowanie własne.

4.3. Seminaria

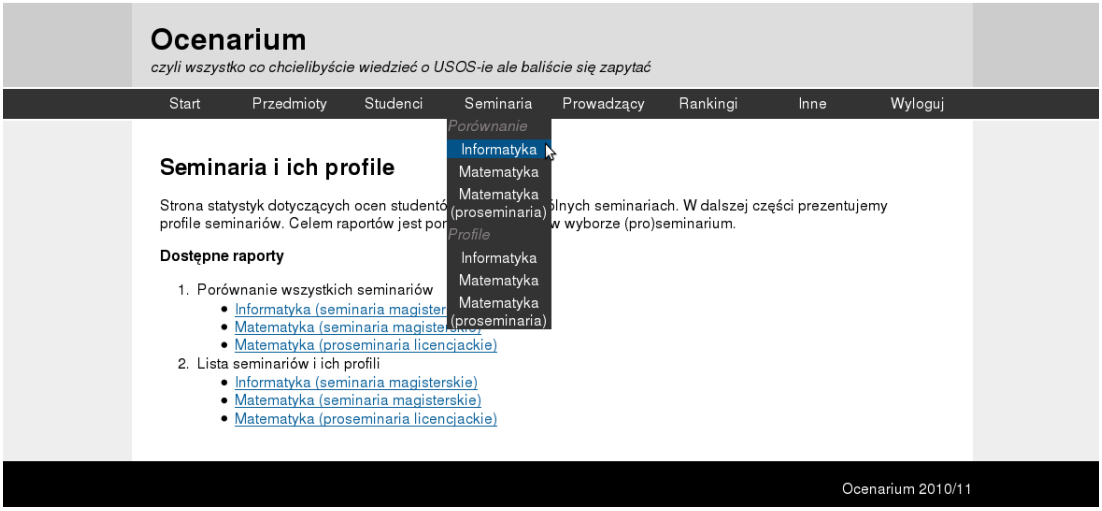
4.3.1. Omówienie *raportu*

Raport Seminaria jest najbardziej złożonym *raportem*. Jego przygotowanie zajęło też najwięcej czasu i pracy. Motywacją stojącą za tym aby właśnie w tym kierunku rozwijać *Ocenarium* była potencjalnie największa liczba odbiorców (użytkowników). Mianowicie zainteresowani informacjami o seminariach są wszyscy studenci. Często wybór odpowiedniego seminarium jest problematyczny, a studenci podejmując decyzje nie mają wielu informacji o tym czego można się spodziewać po poszczególnych seminariach. *Raport Seminaria* wychodzi im na przeciw i podaje dodatkowe informacje ułatwiające ten kluczowy na studiach wybór. Dodatkowym argumentem stojącym za tym by bardziej zaangażować się w analizę seminariów była *potrzeba chwili*. Bowiem w czasie jej opracowywania, na Wydziale MIM UW trwała rejestracja na seminaria magisterskie (proseminaria licencjackie).

Strukturalnie *raport* dzieli się na dwa segmenty. Pierwszy to *Porównanie* i polega on na zestawieniu ze sobą wszystkich seminariów danego kierunku. Przyglądamy się w nim jak studenci danego seminarium wypadają na egzaminach z przedmiotów obowiązkowych pierwszych lat studiów. Drugi natomiast to *Profile* seminariów i dotyczy on już poszczególnych seminariów. Pomysł polega na tym aby pokazać studentom jakie przedmioty wybierają uczestnicy danego seminarium, i na jakich przedmiotach sobie najlepiej radzą. Plikami źródłowymi dla tych segmentów są odpowiednio `seminaria_i_przedmioty.R` oraz `profil_seminarium.R` z katalogu `/home/users/fg235984/usosR/R/seminaria/`.



Rysunek 4.13: Historia skreśleń. Źródło: opracowanie własne.



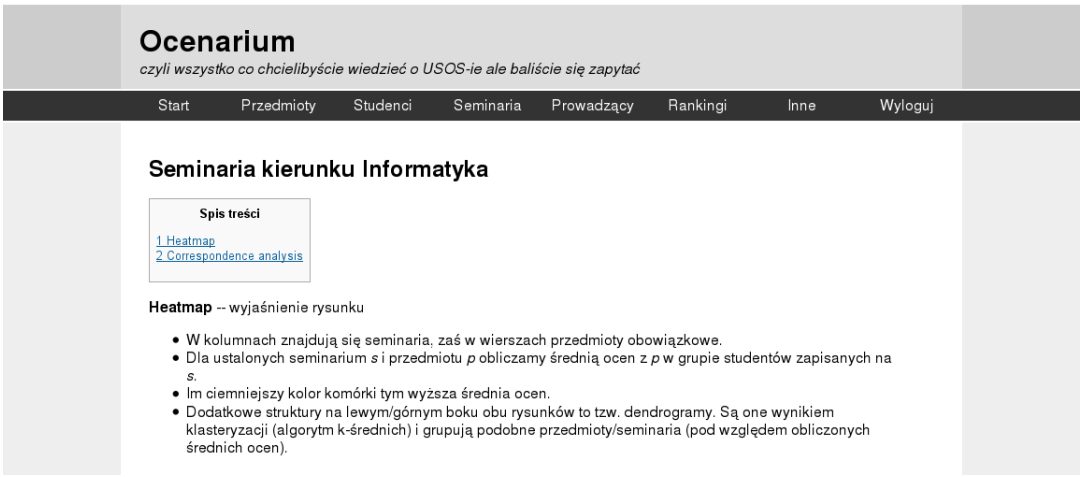
Rysunek 4.14: Strona główna *raportu Seminaria*. Źródło: opracowanie własne.

4.3.2. Scalanie pokrewnych przedmiotów

Zanim przejdziemy do omawiania *raportu* poruszymy jeden z problemów jaki pojawił się przy okazji analizy seminariów. Podstawowym aspektem jaki badamy w tym *raporcie* są oceny z przedmiotów uzyskiwane przez studentów danego seminarium. Innym aspektem są przedmioty najchętniej wybierane przez studentów danego seminarium. W ten sposób możemy porównywać seminaria między sobą i mówić w pewnym sensie o profilu seminarium.

Jednak aby poprawnie analizować oceny z przedmiotów musimy czasem traktować przedmioty o różnych kodach a nawet nazwach jako jeden przedmiot. Przykładem są tutaj przedmioty z nowego programu studiów pierwszego stopnia, np. *Algorytmy i struktury danych*. ASD prowadzone wg starego programu mają kod przedmiotu *1000-213aASD*, zaś wg nowego *1000-213bASD*. W tym przypadku jest to prawdopodobnie związane ze zmianą specyfiki przedmiotu, tj. wprowadzeniem laboratorium do jego programu. Podobnie jest z wieloma innymi przedmiotami i w tym przypadku traktujemy je jako jeden przedmiot (scalenie *po nazwie*). Innym przypadkiem są przedmioty mające więcej niż jeden potok. Przykładowo, *Analiza matematyczna I.1 (potok I) (1000-111bAM1a)* i *Analiza matematyczna I.1 (potok II) (1000-111bAM1b)*. Wtedy również scalamy te przedmioty traktując je jako jeden. Robimy to ucinając z nazwy sufiks rozpoczynający się nawiasem otwierającym '(', a później jeszcze ostatnią spację napisu który pozostanie. Takich manipulacji można dokonać już na poziomie zapytania SQL (funkcje `REGEXP_SUBSTR()` i `rtrim()`).

Nie scalamy natomiast potoków z gwiazdką, ponieważ naszym zdaniem jest to zatarcie ważnej informacji, która może zaburzać wyniki analiz.



Rysunek 4.15: Seminaria kierunku Informatyka – porównanie. Źródło: opracowanie własne.

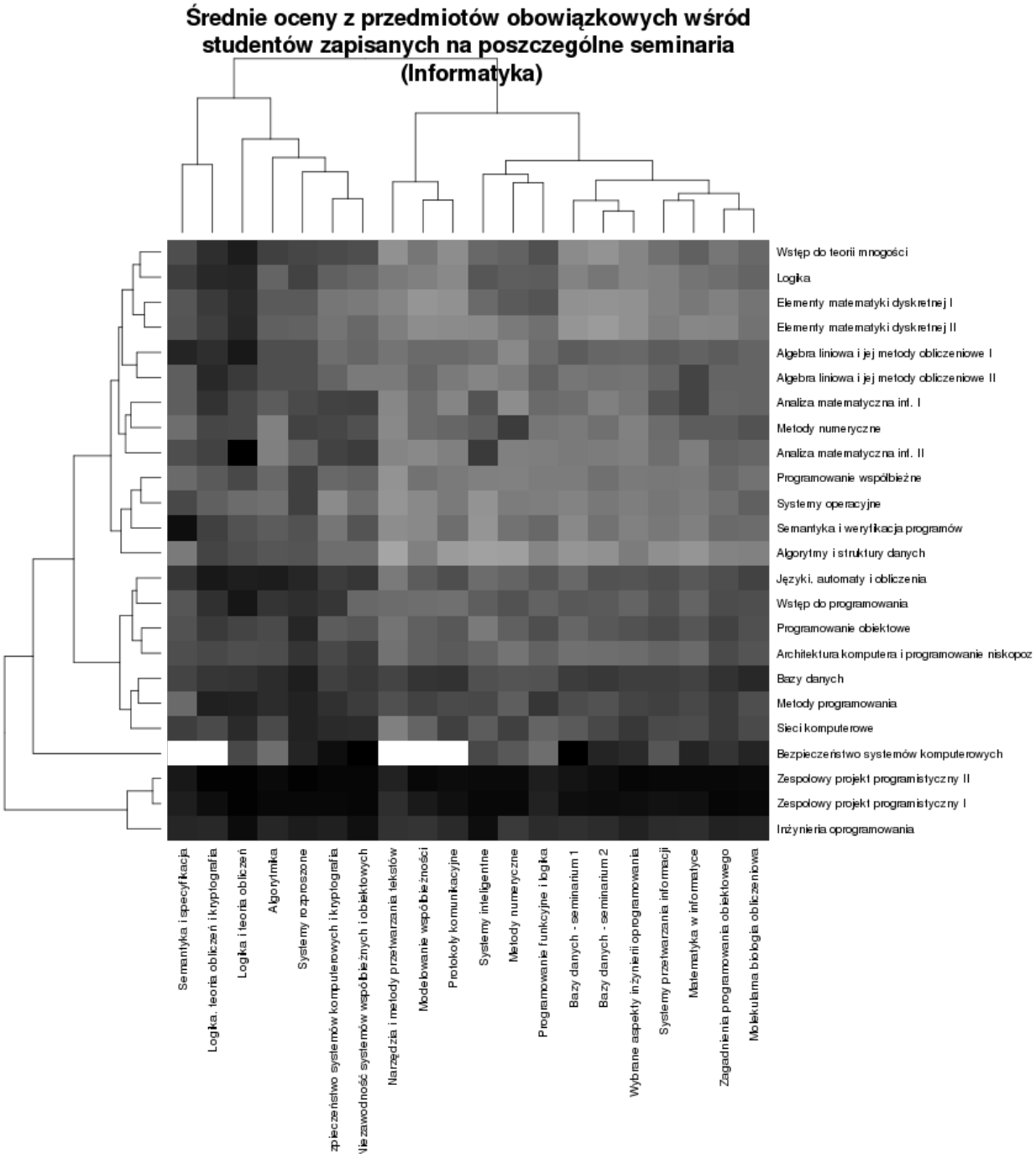
4.3.3. Szczegóły

Podobnie jak w opisie poprzedniego *raportu* zaczynamy od prezentacji jego strony tytułowej. Po wybraniu w menu *Ocenarium* zakładki *Seminaria* przechodzimy do strony ukazanej na rysunku 4.14. Widzimy, że zasadniczo mamy wybór pomiędzy dwiema opcjami: *Porównanie wszystkich seminariów* oraz *Lista seminariów i ich profili*. W dalszej części omówimy po je kolei.

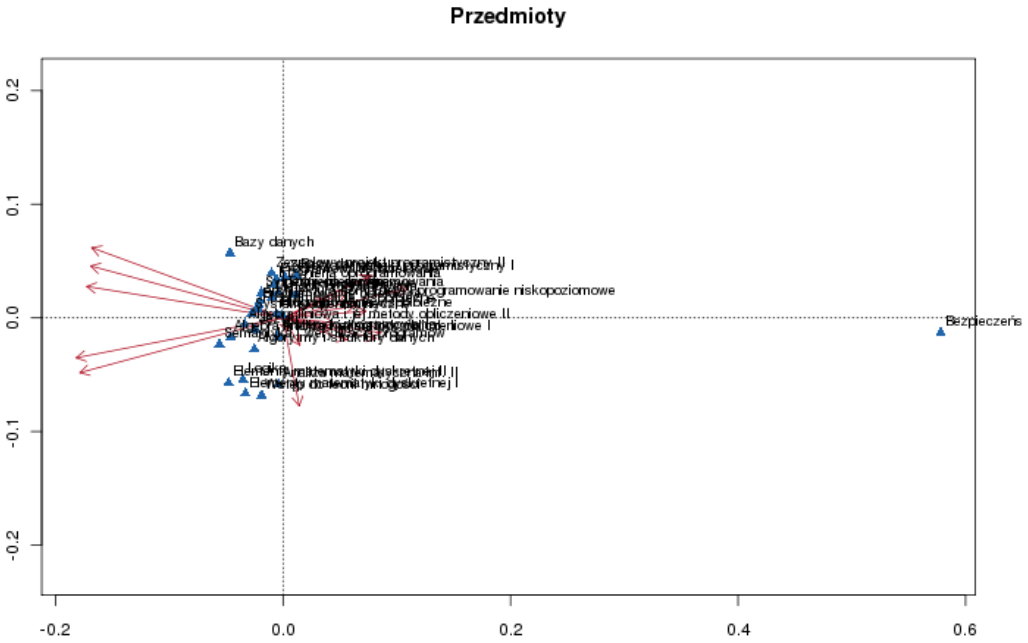
Porównanie seminariów

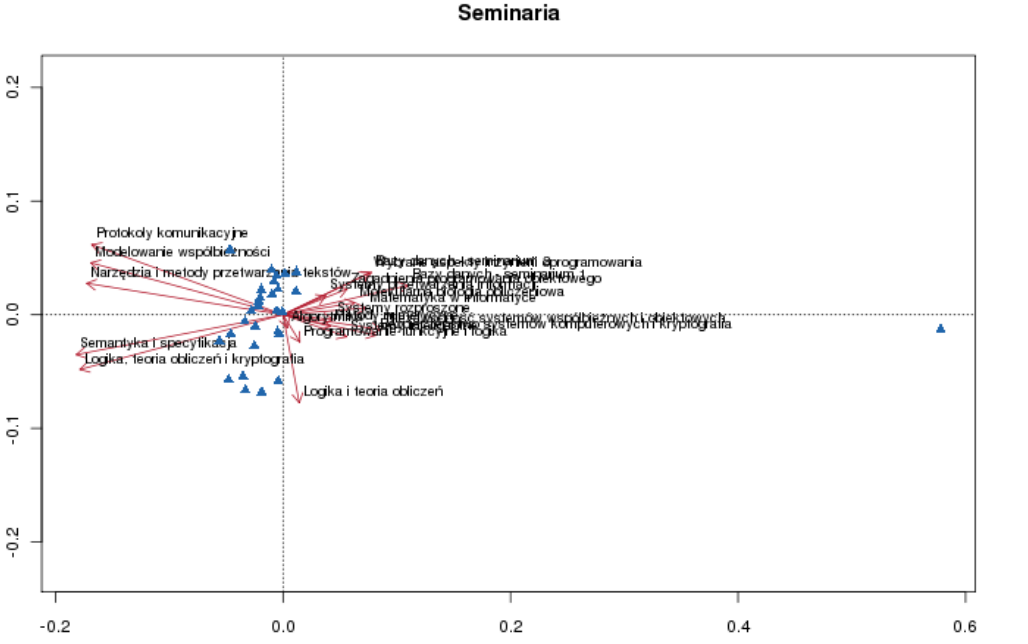
Wybierzmy zakładkę dotyczącą porównania seminariów magisterskich na Informatyce. Alternatywnie możemy w menu głównym *Ocenarium* wybrać kolejno *Seminaria* → *Porównanie* → *Informatyka* (tak jak na rysunku 4.14). Efekt będzie taki sam, tzn. strona z rysunku 4.15. Widać na niej wstęp do pierwszego wykresu, który z kolei pokazany jest na rysunku 4.16. Jest to wykres wygenerowany przez funkcję `heatmap()` (plik źródłowy to `/home/users/fg235984/usosR/R/seminaria/seminaria_i_przedmioty.R`). Jego podstawą – jak zwykle – jest odpowiednie zapytanie SQL. Polega ono na złączeniu wyników dwóch zapytań. Pierwsze dotyczy seminariów magisterskich na jakie chodzi dany student. Drugie zaś dotyczy ocen uzyskiwanych przez studentów na przedmiotach obowiązkowych pierwszych trzech lat studiów (studia pierwszego stopnia). Ostatnim krokiem jest agregacja złączenia po parach $(seminarium, przedmiot)$ w celu obliczenia średnich ocen w poszczególnych grupach.

Mając tak przygotowaną tabelę danych wystarczy w standardowy sposób wywołać funkcję `heatmap()`. Wynikiem jest tzw. *mapa ciepła*, czyli tablica o komórkach poko-



Rysunek 4.16: Wykres *Mapa ciepła*. Źródło: opracowanie własne.





Rysunek 4.17: Wykres *Analiza współwystępowania*. Źródło: opracowanie własne.

lorowanych różnymi odcieniami szarości. Im ciemniejszy (*cieplejszy*) kolor tym wyższa średnia ocen. Białe pola oznaczają brak odpowiednich obserwacji. Ciekawym uzupełnieniem tego wykresu są dendrogramy na lewym i górnym boku tablicy. Stosują one algorytm k-średnich do pogrupowania *podobnych* seminariów i przedmiotów.

Jeśli chodzi o wnioski z wykresu, to można zauważyć, że studenci seminarium *Systemy rozproszone* mają najlepsze oceny. Jest to zgodne z realiami rejestracji na seminaria, bowiem seminarium to jest jednym z najbardziej obleganych i trafiają tam dobrzy studenci. Widać też że oceny z przedmiotów *Zespołowy projekt programistyczny* i *Inżynieria oprogramowania* są zdecydowanie najlepsze.

Nie będziemy kontynuować już tego typu obserwacji, ponieważ jest ich zbyt dużo, zaś każdy czytelnik może je sam z łatwością poczynić.

Na kolejnym rysunku (4.17) widzimy wykres zatytułowany *Analiza współwystępowania*. Jest to technika statystyczna badania tablicy danych mająca na celu znalezienie związków pomiędzy kolumnami i wierszami. W naszym przypadku tablicą danych jest tablica omawiana przy okazji wykresu *Mapa ciepła*. Natomiast, to co chcemy ustalić to zależność między seminariami (kolumnami), a przedmiotami (wierszami). Niebieskie trójkąty oznaczają przedmioty obowiązkowe z pierwszych trzech lat studiów. Czerwone strzałki to poszczególne seminaria. Aby zwiększyć czytelność wykres składa się z dwóch części. Na pierwszej umieszczone są tylko etykiety przedmiotów, zaś na drugiej tylko etykiety seminariów. Poza tym niczym się one nie różnią.

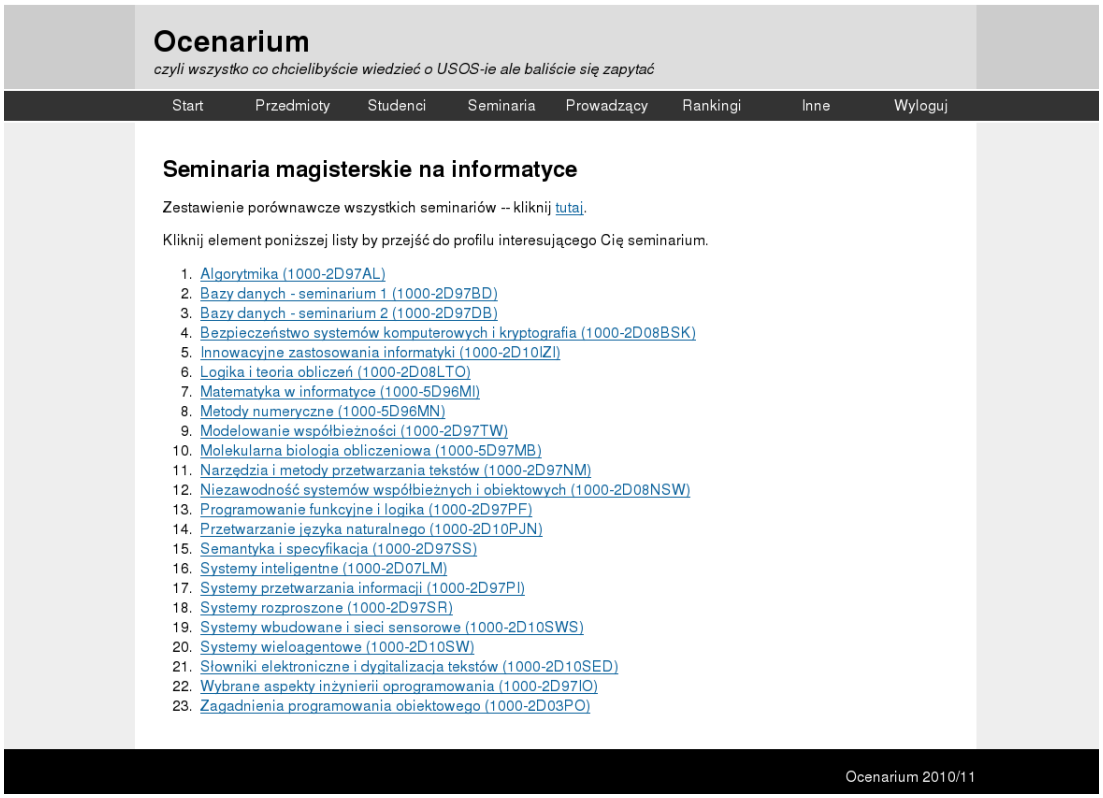
Profile seminariów

Jeśli klikniemy w menu głównym *Ocenarium* kolejno *Seminaria* → *Profile* → *Informatyka* (patrz rysunek 4.14), to przejdziemy do listy wygenerowanych *raportów* dotyczących profili poszczególnych seminariów kierunku Informatyka. W przeglądarce pojawi się wtedy to samo co na rysunku 4.18. Wybierzmy z listy pozycję *4. Bezpieczeństwo systemów komputerowych i kryptografia (1000-2D08BSK)*. W efekcie zobaczymy to co na rysunku 4.19.

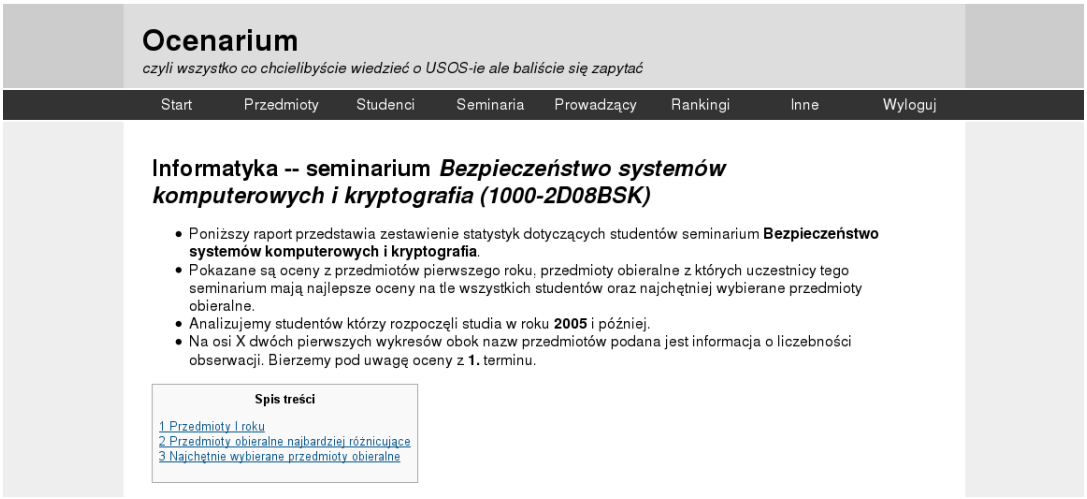
Omawianie poszczególnych elementów *raportu* zaczniemy od przytoczenia założeń. Badamy oceny studentów którzy rozpoczęli studia w roku 2005 i później. Oceny które bierzemy pod uwagę pochodzą z pierwszego terminu egzaminu. Oba te parametry można zmienić w skrypcie `profil_seminarium.R` (w katalogu `/home/users/fg235984/usosR/R/seminaria/`), który jest jedynym plikiem źródłowym *raportu Seminaria*.

Pierwszy z trójki wykresów (rysunek 4.20) pokazuje jak studenci wybranego przez nas seminarium radzą sobie na tle wszystkich studentów. Kryterium w tym wypadku są oceny z przedmiotów pierwszego roku studiów. Jest ich aż 17 ponieważ studenci z roczników 2005-06 studiowali jeszcze wg starego programu studiów. Oczywiście także tutaj stosujemy scalanie nazw przedmiotów opisane w rozdziale 4.3.2. Obok nazw przedmiotów podane są liczności grup obserwacji.

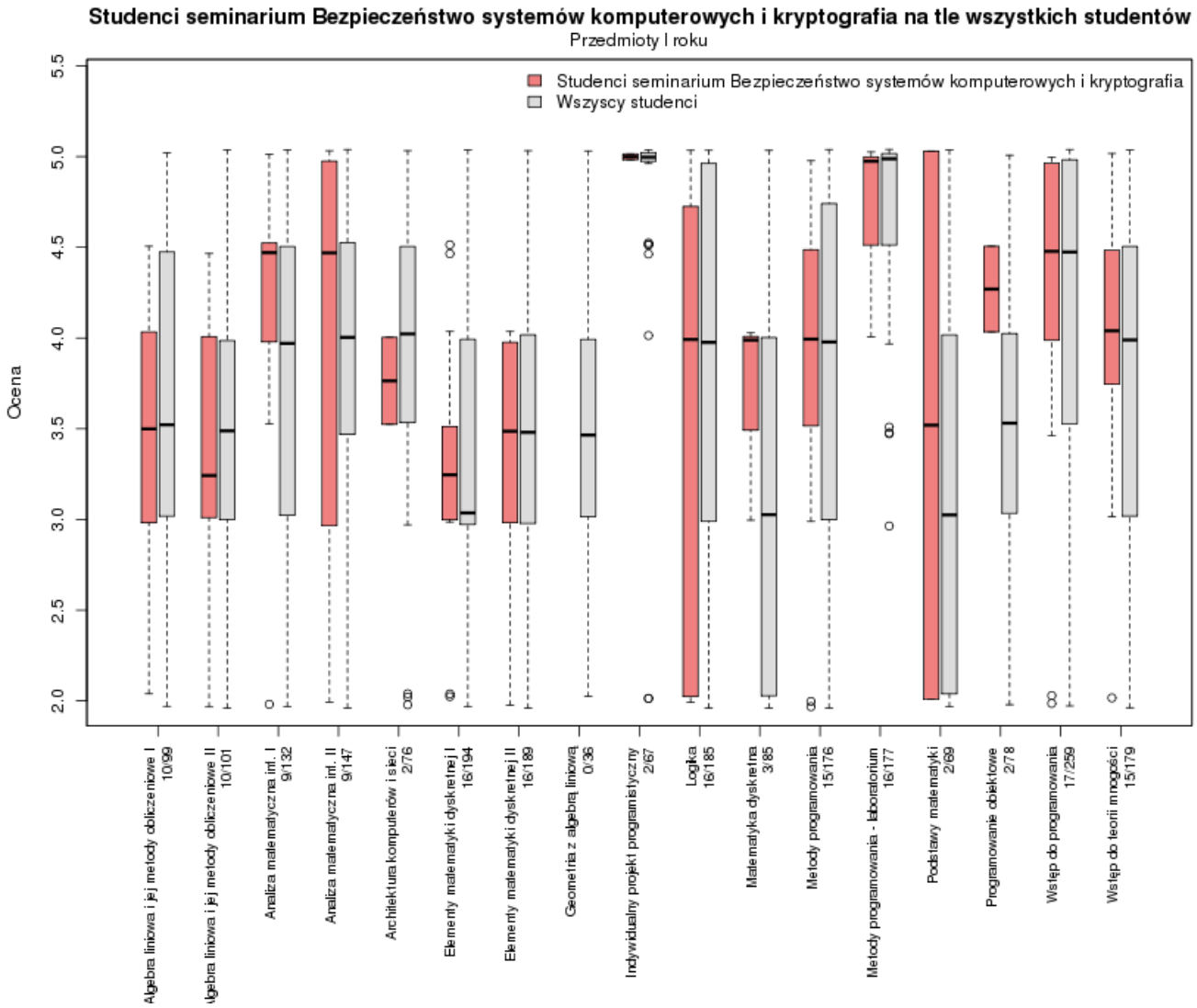
Drugi wykres obrazujący profil seminarium znajduje się na rysunku 4.21. Wybraliśmy maksymalnie 10 przedmiotów z puli przedmiotów obieralnych i monograficznych (*do wyboru*), na których studenci ustalonego seminarium radzą sobie najlepiej na tle



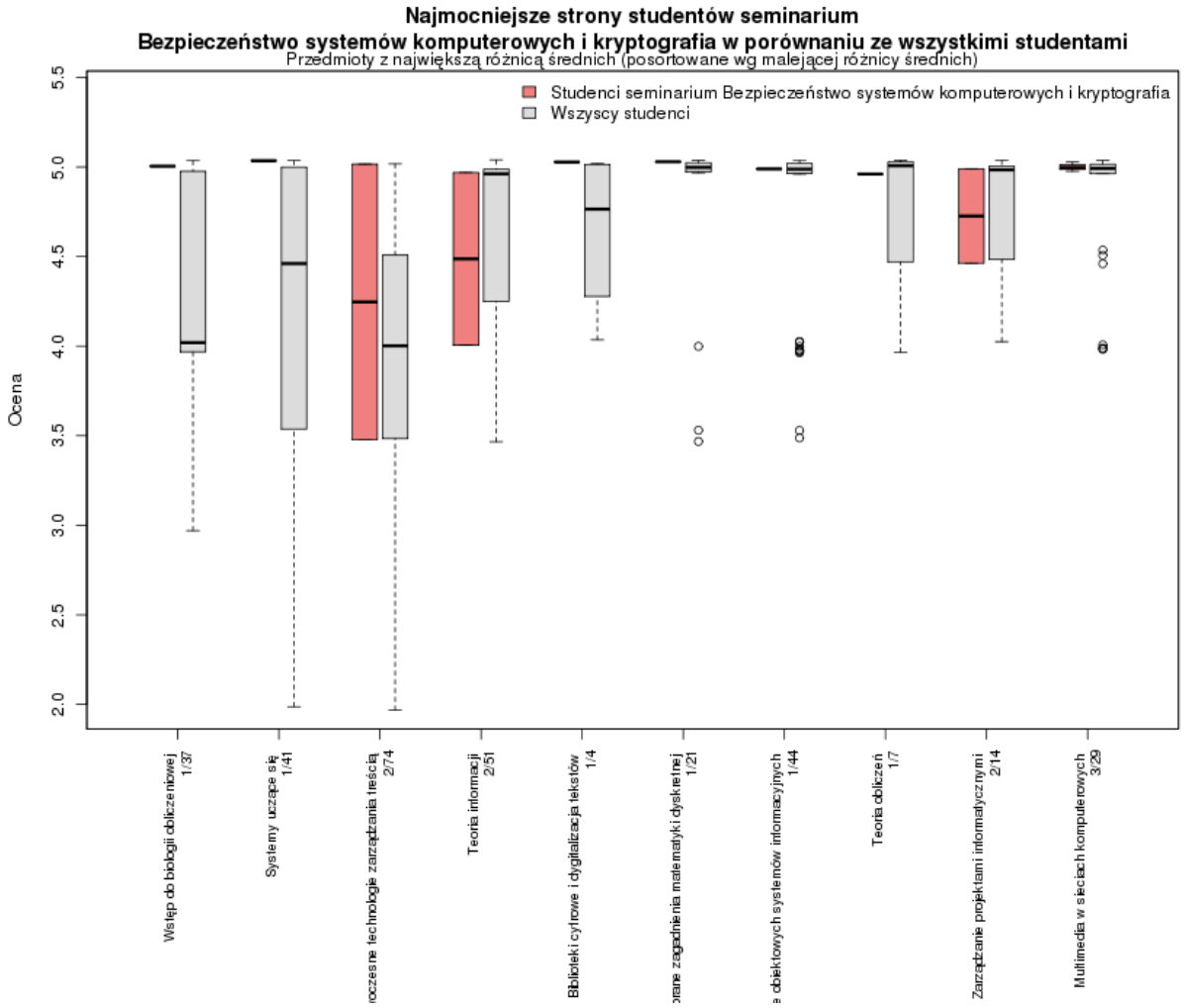
Rysunek 4.18: Lista *raportów* z profilami seminariów. Źródło: opracowanie własne.



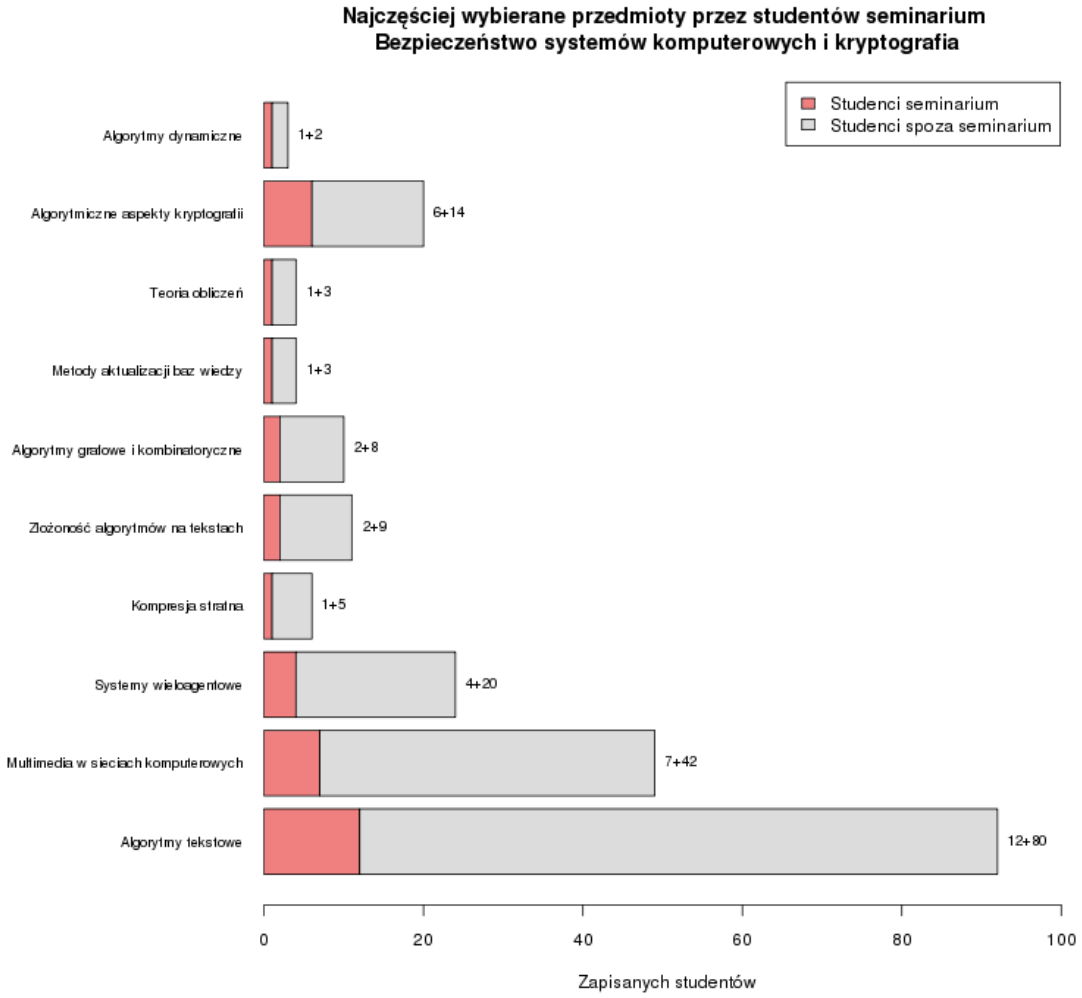
Rysunek 4.19: Profil seminarium *Bezpieczeństwo systemów komputerowych i kryptografia*. Źródło: opracowanie własne.



Rysunek 4.20: Oceny z przedmiotów pierwszego roku. Źródło: opracowanie własne.



Rysunek 4.21: Najmocniejsze strony studentów seminarium *Bezpieczeństwo systemów komputerowych i kryptografia*. Źródło: opracowanie własne.



Rysunek 4.22: Najchętniej wybierane przedmioty przez studentów seminarium *Bezpieczeństwo systemów komputerowych i kryptografia*. Źródło: opracowanie własne.

wszystkich studentów. Kryterium wyboru przedmiotów jest różnica średnich ocen wśród studentów seminarium i ogółu studentów. Kolejność przedmiotów na wykresie jest ustalona tak, że im bliżej lewej strony tym różnica średnich ocen jest korzystniejsza dla studentów seminarium. W naszym przypadku *Wstęp do biologii obliczeniowej* jest przedmiotem, gdzie różnica średnich na korzyść studentów seminarium jest największa. Podobnie jak na poprzednim wykresie podane są liczności grup obserwacji. W skrypcie `profil_seminarium.R` reguluje to parametr `min_liczba_ocen`.

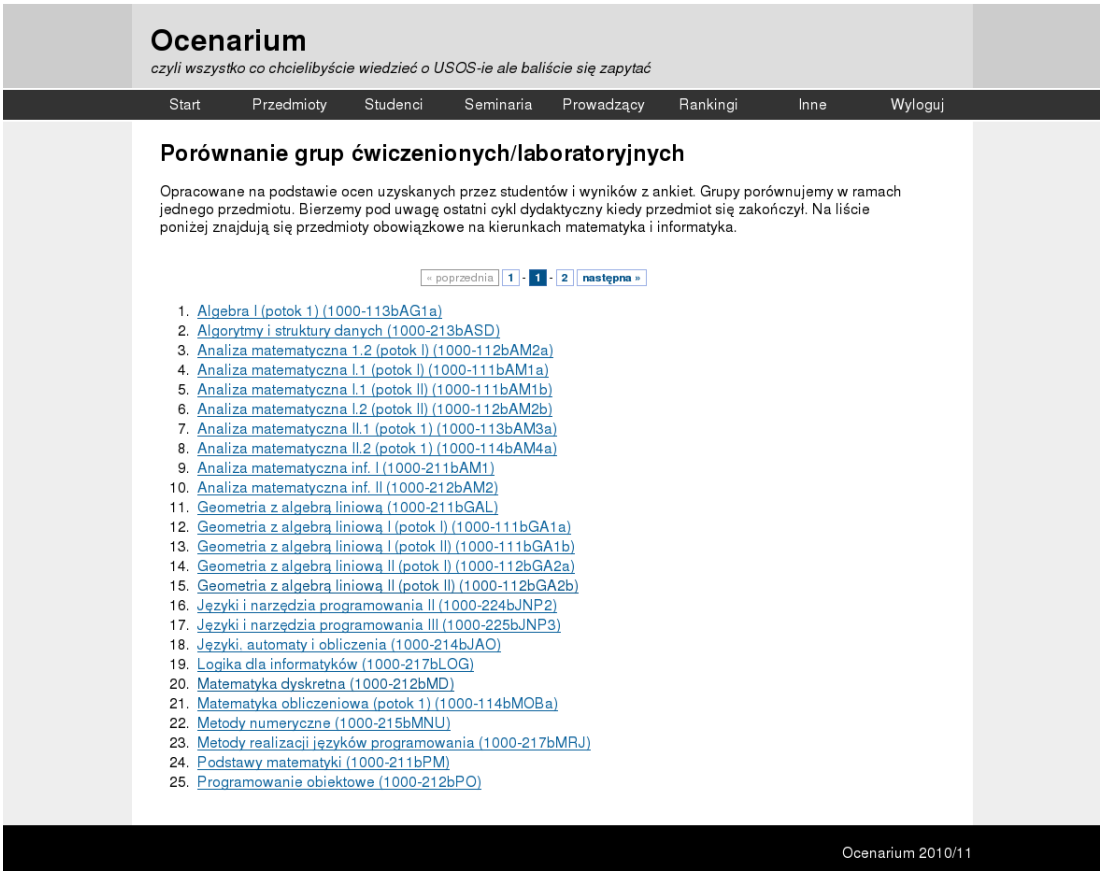
Wykres ten jest jednym z najbardziej złożonych, szczególnie w warstwie pozyskania danych do prezentacji. Nakreślimy zatem jak zostało to wykonane. Pierwszym krokiem było obliczenie różnic średnich ocen pomiędzy studentami seminarium, a wszystkimi studentami. Jest to rozwiązane w ten sposób, że za pomocą zapytania SQL obliczamy średnie ocen z przedmiotów obieralnych dla obu grup. Następnie z poziomu kodu R obliczamy różnice ocen między badanymi grupami. Scalamy po kolumnie `prz_nazwa` (nazwa przedmiotu) dwie ramki danych dołączając do nich kolumnę z różnicami średnich (`diffs`), sortujemy (malejąco) nową ramkę po kolumnie `diffs` i wybieramy listę początkowych dziesięciu przedmiotów. Mając taką listę – za pomocą odpowiedniego zapytania SQL – jesteśmy w stanie wybrać oceny studentów z dziesięciu przedmiotów na których najlepiej radzą sobie studenci seminarium. Wykonujemy to dwa razy – raz, tylko dla studentów seminarium i drugi, dla wszystkich studentów. Ostatnim krokiem jest narysowanie obok siebie dwóch wykresów (`boxplot`).

Trzeci z wykresów tego *raportu* (rysunek 4.21) pokazuje na jakie przedmioty najchętniej zapisują się studenci seminarium *Bezpieczeństwo systemów komputerowych i kryptografia*. Najchętniej oznacza tutaj największy stosunek liczby studentów wybranego seminarium do ogółu studentów. Widzimy z rysunku, że największy taki stosunek, wynoszący 1/2 mamy na *Algorytmach dynamicznych*. Podobnie jak przy poprzednim wykresie szukamy maksymalnie dziesięciu takich przedmiotów – parametr `top_licznosci`. W skrypcie `profil_seminarium.R` istnieje inny parametr (`min_liczba_osob`), który reguluje minimalną liczbę osób z seminarium którzy dany przedmiot wybrali (by wyeliminować przypadek, który tutaj mamy ze stosunkami 1/2, 1/3 itp.). Jeśli chodzi o sposób postępowania przy generowaniu tego wykresu, to jest on analogiczny do poprzedniego. Różnica polega na tym, że zamiast liczyć średnie oceny zliczamy osoby które mają ocenę z egzaminu (w którymkolwiek terminie) i zamiast liczyć różnicę średnich liczymy stosunek liczby studentów w obu grupach. Metoda mieszania kodu SQL i R do znalezienia wizualizowanych zbiorów danych pozostaje w mocy. Zmieniła się natomiast funkcja, którą rysujemy wykres. W tym przypadku jest nią `barplot()`.

4.4. Prowadzący

4.4.1. Omówienie *raportu*

W tym *raporcie* przyglądamy się prowadzącym zajęcia. Analizujemy grupy ćwiczeniowe (laboratoryjne) prowadzone przez poszczególnych ćwiczeniowców (laborantów) w ramach jednego przedmiotu. Obiektami naszego zainteresowania są oceny jakie studenci otrzy-



Rysunek 4.23: Lista przedmiotów do wyboru. Źródło: opracowanie własne.

mują na egzaminie oraz oceny wystawiane prowadzącym przez studentów w ramach ankiet podsumowujących semestr.

Głównym celem tego *raportu* było znalezienie trendów i korelacji między skutecznością prowadzącego (egzamin), a byciem lubianym przez studentów (ankiety). Inaczej można też na to patrzeć jako na ranking prowadzących.

Implementacja tego *raportu* znajduje się w pliku `/home/users/fg235984/usosR/R/prowadzacy/ranking_prowadzacych.R`.

4.4.2. Szczegóły

Przechodzimy do szczegółowego omówienia poszczególnych elementów *raportu*. Po kliknięciu zakładki *Prowadzący* w głównym menu przechodzimy do spisu przedmiotów dla których sporządzono *raporty*. Widać to na pierwszym rysunku (4.23). Lista składa się

Rysunek 4.24: Nagłówek *raportu Prowadzący*. Źródło: opracowanie własne.

z przedmiotów obowiązkowych, dla których jest co najmniej 3 prowadzących w zestawieniu. Po wybraniu przedmiotu *Analiza matematyczna inf. I (1000-211bAM1)* przechodzimy do *raportu* odpowiadającego temu przedmiotowi. W nagłówku wypisane są informacje ogólne dotyczące przedmiotu (rysunek 4.24) takie jak: cykl dydaktyczny, prowadzący wykład (koordynator) i termin egzaminu z którego oceny analizujemy. Pierwszy z wykresów pokazany jest na rysunku 4.25. Przedstawia on oceny jakie uzyskali studenci danego prowadzącego na egzaminie. Dodatkową informacja jest liczebność zbioru obserwacji zawarta w szerokości paska boxplotu.

Większość pracy implementacyjnej jest tutaj wykonana po stronie bazy danych. Mając gotowe zapytanie SQL narysowanie wykresu w tym przypadku jest standardowe.

Analogicznie przedstawia się rysunek 4.26, gdzie przyglądamy się ocenom jakie uzyskali prowadzący w ankietach.

Najciekawszym obiektem niniejszego *raportu* jest wykres łączący dwa wcześniejsze wykresy. Technicznie jest to *suma* dwóch boxplotów. *Sumami* w tym przypadku są *krzyże*. Jeden *krzyż* odpowiada jednemu nazwisku. Zatem jeśli jakiś pracownik prowadzi więcej niż jedną grupę, to na wykresie są one scalone w jedną. Podobnie jest zresztą w przypadku dwóch pierwszych wykresów. Poziome ramię *krzyża* odpowiada ocenom studen-

Rysunek 4.28: *Raport Rankingi*. Źródło: opracowanie własne.

4.5. Rankingi

4.5.1. Omówienie *raportu*

Raport Rankingi prezentuje zestawienie najbardziej lubianych pracowników dydaktycznych. Sympatię w stosunku do pracownika mierzymy na podstawie ankiet. Ankiety są wypełniane przez studentów pod koniec każdego cyklu dydaktycznego – zwykle na ostatnich zajęciach przed sesją. Na Wydziale MIM UW od semestru (2010Z) ankiety wypełniane są poprzez USOSweb. *Raport* może być użyteczny dla studentów i kadry dydaktycznej.

Implementacja tego *raportu* znajduje się w pliku `/home/users/fg235984/usosR/R/prowadzacy/zbiorcze_rankingi.R`.

4.5.2. Szczegóły

Zaczynamy od pokazania wstępu do *raportu* (rysunek 4.28), w którym krótko objaśniamy co znajduje się w dalszej jego części. Warto zwrócić uwagę na to, że tabele rankingowe zawierają po 12 pozycji, zaś analizowane dane pochodzą z cyklu dydaktycznego 2010Z (stałe `results_num_head` i `cykl_dyd`).

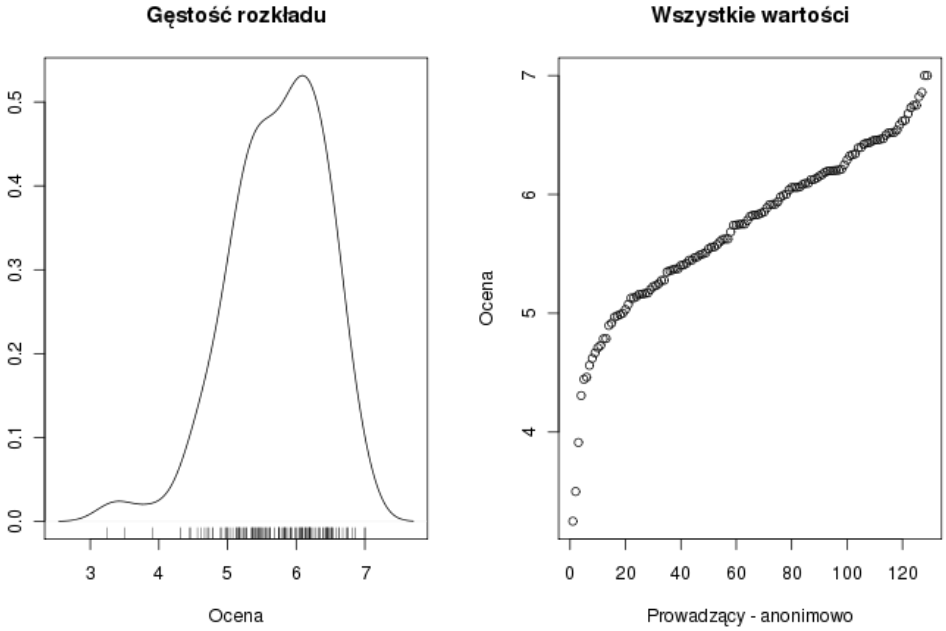
Na rysunku 4.29 znajduje się tabela z rankingiem wykładowców. Analogiczne tabele zostały wykonane dla ćwiczeniowców i prowadzących laboratoria. Jak widać tabela ma ograniczoną liczbę pozycji – wymienia tylko 12 najlepszych wyników. Jest tak ponieważ nie chcemy urazić osób znajdujących się na niższych pozycjach. Liczba na podstawie której ustalamy pozycje prowadzącego, to średnia arytmetyczna wszystkich ocen składowych ankiety. Dodatkowo podajemy liczbę wypełnionych ankiet, oraz odchylenie standardowe. Jeśli chodzi o techniczne aspekty tego raportu, to większość pracy została wykonana za pomocą odpowiedniego zapytania SQL (grupowanie po prowadzącym i przedmiocie), reszta to tylko wydrukowanie wyniku (funkcja `HTML()` z pakietu `R2HTML`).

Ranking wykładowców

	Kod	Nazwa	Prowadzący	Średnia ocena	Odchylenie standardowe	Liczba ankiet
1	1000-2M07MD	Wybrane zagadnienia matematyki dyskretnej	Adam Malinowski	7.00	0.000	5
2	1000-1M10ARR	Algorytmy rownolegle rozwiazywania rownan rozniczkowych	Maksymilian Dryja	7.00	0.000	1
3	1000-135PK1	Analiza portfelowa i rynki kapitalowe I	Mariusz Barylo	6.86	0.625	17
4	1000-2M08PMK	Programowanie mikrokontrolerow	Marcin Engel	6.82	0.435	12
5	1000-1M10RT	Rozmaitosci toryczne	Jaroslaw Wisniewski	6.75	0.577	2
6	1000-1MATB1	Matematyka B	Witold Szczechla	6.75	0.707	1
7	1000-135AG3	Algebra III	Jan Okninski	6.73	0.777	11
8	1000-2M10ATJ	Algebraiczna teoria jezykow	Mikolaj Bojanczyk	6.68	0.728	18
9	1000-2M08PMK	Programowanie mikrokontrolerow	Marcin Peczarski	6.62	0.567	12
10	1000-113bTP1*	Topologia I (potok *)	Roman Pol	6.62	0.745	19
11	1000-111bAM1a	Analiza matematyczna I.1 (potok I)	Pawel Strzelecki	6.59	0.797	75
12	1000-1M10MLM	Modele liniowe i mieszane na przykladzie analizy danych biologicznych i medycznych	Przemyslaw Biecek	6.54	0.779	3

Rysunek 4.29: Ranking wykładowców – tabela. Źródło: opracowanie własne.

Uzupełnieniem tabeli jest rysunek 4.30 z rozkładem średniej oceny wykładowcy. Składa się on z dwóch części. Pierwsza to gęstość rozkładu (`plot(density())` w R), druga zaś pokazuje w rosnącym porządku średnie wszystkich prowadzących.



Rysunek 4.30: Ranking wykładowców – rozkład ocen. Źródło: opracowanie własne.

Rozdział 5

Instalacja i konfiguracja

W tym rozdziale przyjrzymy się w jaki sposób poprawnie zainstalować i skonfigurować *Ocenarium*. Jako case-study posłuży nam instancja uruchomiona na hostach (vhostach) `usoshurt.mimuw.edu.pl` i `usosphp.mimuw.edu.pl`. Aplikacja jest wtedy dostępna w sieci pod adresem `http://usosphp.mimuw.edu.pl/usosR`.

5.1. Hosty – założenia

Wyjaśnijmy przede wszystkim, że *Ocenarium* jest wdrożone na dwóch hostach: `usoshurt.mimuw.edu.pl` i `usosphp.mimuw.edu.pl`. Pierwszy z nich hostuje część związaną z bazą danych (hurtownią danych), zaś drugi część statystyczną oraz interfejs www.

Oba hosty powinny dzielić kilka cech. Pierwszą z nich jest system operacyjny Linux (Debian 4.3.2-1.1). Drugą, skonfigurowany zdalny dostęp przez ssh z maszyny studenckiej `students.mimuw.edu.pl`.

Umawiamy się, że w dalszej części tego rozdziału podając katalogi i ścieżki do plików będziemy używać lokacji pochodzących z maszyn `usoshurt` i `usosphp`. Załączona do pracy płyta zawiera źródła zorganizowane w analogiczny sposób. W razie wątpliwości translacje ścieżek zostały wyjaśnione w dodatku C.

5.2. `usoshurt.mimuw.edu.pl`

Serwer `usoshurt.mimuw.edu.pl` hostuje hurtownię danych systemu *Statystyki*. Katalogiem gdzie umieszczone są źródła jest `/home/fg235984/STATYSTYKI/`.

W rozdziale 5.2.1 podajemy kolejno założenia odnośnie serwera hostującego hurtownię danych oraz listę zmiennych środowiskowych jakie należy ustawić. W rozdziale 5.2.2 pokazujemy jakie kroki należy wykonać by uruchomić hurtownię, a później zasilić ją danymi.

5.2.1. Wymagane oprogramowanie i założenia

Do instalacji hurtowni dla *Ocenarium* wykorzystana jest darmowa wersja systemu zarządzania bazą danych Oracle XE (Express Edition). Oprócz tego, potrzebne jest dowolne narzędzie pozwalające na połączenie się do bazy danych Oracle oraz na wykonanie skryptów SQL. Dobrym wyborem jest na przykład darmowe narzędzie SQL Plus.

Możemy się już zalogować na hosta `usoshurt.mimuw.edu.pl`. Używamy do tego protokołu ssh. Następnie zmieniamy katalog roboczy na `/home/fg235984/STATYSTYKI/SQL`. To w nim znajdują się skrypty SQL, które będziemy wykonywać.

Poniżej wypisujemy część pliku `/home/fg235984/.bash_profile` związaną z hurtownią. Są to definicje wymaganych zmiennych środowiskowych.

```
ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/product/10.2.0/server
ORACLE_OWNER=oracle
ORACLE_SID=XE
LSNR=$ORACLE_HOME/bin/lsnrctl
SQLPLUS=$ORACLE_HOME/bin/sqlplus
export ORACLE_HOME
export ORACLE_SID
export PATH=$ORACLE_HOME/bin:$PATH
LOG="$ORACLE_HOME_LISTNER/listener.log"
# Polskie znaki
export NLS_LANG=POLISH_POLAND.UTF8
```

5.2.2. Uruchomienie hurtowni

Zacznijmy od uwagi o tym, że instancja *Ocenarium* na której pracowaliśmy korzysta z instancji hurtowni wdrażanej przez Grzegorza Kukawskiego w związku z jego pracą magisterska ([6]). Dokładniej, została ona odzyskana z backup-u. W związku z tym instalacja i konfiguracja hurtowni opisana w dokumencie [5] pozostaje w mocy.

Dla czystości sumienia dodajmy jednak jakich czynności wymaga dotarcie do stanu odzyskanej z backup-u hurtowni. Przypomnijmy że wersja hurtowni z której korzysta *Ocenarium* nie wymaga pełnej instalacji i inicjalizacji struktur bazodanowych hurtowni. Wystarczy że ograniczymy się do kroków które kończą się zasileniem schematu STAGE (por. 2.4.2). Zatem instalacja i konfiguracja hurtowni danych systemu *Statystyki* wymaga wykonania czynności które są podzbiorem czynności opisanych w dokumencie [5]. Są to kroki 1-7 z rozdziału *1.1. Instalacja struktur bazodanowych.*

Część instalacji którą należy wykonać dodatkowo to skrypty wymieniane w rozdziale 2.5. Uruchamiamy je w kolejności w jakiej są tam wymienione (leksykograficznej ze względu na nazwę).

Ostatnią rzeczą jaką należy zrobić to zasilenie hurtowni danymi. Znów odwołujemy się do dokumentu [5]. Rozdział *1.2. Zasilenie systemu danymi z bazy USOS* dokładnie opisuje jak wywołać procedurę ładującą dane `stage.usos_load`. Alternatywnie, mówi też o tym dodatek A (zaraz po uwadze A.1.1).

5.3. usosphp.mimuw.edu.pl

Serwer `usosphp.mimuw.edu.pl` hostuje część związaną z interfejsem i obliczeniami statystycznymi. Poniżej wymieniamy czynności związane z poszczególnymi technologiami, które należy wykonać w celu uruchomienia aplikacji.

5.3.1. Django

Nasz interfejs wymaga zainstalowania Django. W tym celu, w systemie operacyjnym Debian wystarczy zainstalować pakiet `python-django`. Na hoście `usosphp` Django było zainstalowane przed powstaniem *Ocenarium*, dlatego krok ten mógł być pominięty. Szczegółowe informacje o instalacji są w dokumentacji Django [24].

5.3.2. WSGI i mod_wsgi

Zacznijmy od dwóch zdań wprowadzenia. Web Server Gateway Interface (WSGI) definiuje prosty i uniwersalny interfejs pomiędzy serwerami www, a aplikacjami www opartymi o język Python. Z kolei mod_wsgi jest modułem serwera HTTP Apache. Zapewnia on interfejs WSGI dla hostowania aplikacji www bazujących na Pythonie 2.3+ pod serwerem Apache. Zatem podsumowując, technologia prezentowana w tym punkcie służy uruchomieniu naszej aplikacji Django na serwerze Apache.

Mając do dyspozycji serwer Apache wzbogacony modułem mod_wsgi wystarczy tylko dodać do pliku konfiguracyjnego `/etc/apache2/apache2.conf` wpis

```
WSGIDaemonProcess usosR user=fg235984 group=users threads=20 display-name="fg235984-daemon"
WSGIProcessGroup usosR
WSGIScriptAlias /usosR /home/users/fg235984/www/django/src/usosR/usosR.wsgi
```

Dodatkowe informacje na temat integracji Django z Apache'm i mod_wsgi można znaleźć w [25].

5.3.3. R

Cały projekt nie ma prawa funkcjonować bez programu R. Na maszynie `usosphp` można go zainstalować wpisując

```
sudo apt-get install r-base
sudo apt-get install r-base-dev
```

Dodatkowo potrzebny jest pakiet `rgl`. Wykonujemy polecenia

```
sudo apt-get install r-cran-rgl
sudo apt-get build-dep r-cran-rgl
```

Dzięki `rgl` możemy w R korzystać z biblioteki do analizy korespondencji (`ca`).

Przygotowane *raporty* wymagają instalacji następujących pakietów R: `RColorBrewer`, `gplots`, `R2HTML`, `gregmisc`, `ca`, `rgl`.

5.3.4. tnsnames.ora

Na maszynie `usosphp` chcemy do naszej hurtowni odwoływać się poprzez alias `XE`. Aby to zapewnić w pliku `$ORACLE_HOME/network/admin/tnsnames.ora` (u nas `ORACLE_HOME=/usr/share/oracle/instantclient_11_1`) dodajemy wpis definiujący połączenie do hurtowni.

```
XE =
   (DESCRIPTION =
     (ADDRESS_LIST =
         (ADDRESS =
           (PROTOCOL = TCP)
           (Host = usoshurt.mimuw.edu.pl)
           (Port = 1521)
         )
     )
     (CONNECT_DATA = (SID = XE)
     )
)
```

5.3.5. X11

Generowanie w R wykresów w trybie batch wymaga wystartowania X serwera. Aby to zrobić podczas logowania na maszynę `usosphp` – za pośrednictwem skryptu `/home/users/fg235984/.bashrc` – wykonujemy kod uruchamiający X serwer. Przy okazji ustawiamy zmienną środowiskową `DISPLAY`, aby wywołanie funkcji `X11()` w R nie wymagało podawania tego argumentu (domyślnie `display` jest pobierane ze zmiennej środowiskowej).

```
# X11 (dla R)
export DISPLAY=":1"
if [ "`ps avx | grep Xv | wc -l`" = "1" ]; then
    Xvfb :1 &
    echo Start X serwera
else
    echo X server uruchomiony
fi
```

5.3.6. Zmienne środowiskowe

Oprócz wspomnianej wcześniej zmiennej `DISPLAY` należy nadać wartość zmiennej `ORACLE_HOME`. Odpowiedni fragment z pliku `/home/users/fg235984/.bashrc` to

```
export ORACLE_HOME=/usr/share/oracle/instantclient_11_1
```

5.4. Generowanie *raportów*

Mając uruchomioną i zasiloną danymi hurtownię oraz działający interfejs możemy przejść do wygenerowania *raportów*. Katalogiem gdzie umieszczone są źródła jest `/home/users/fg235984/usosR/`. Zaczynamy od zalogowania się na maszynę `usosphp`. Następnie zmieniamy katalog roboczy na `/home/users/fg235984/usosR/scripts` i wykonujemy skrypt `generate_all.sh`. Skrypt ten wykonuje wszystkie skrypty ze swojego

katalogu. Każdemu *raportowi* odpowiada jeden skrypt. Zatem chcąc wykonać, albo powtórzyć wykonanie jakiegoś *raportu* można uruchomić tylko wybrany skrypt.

Rozdział zakończmy uwagą, że sposób procedowania w przypadku dodawania nowego *raportu* został wyjaśniony w dodatku A. Może on być lekturą uzupełniającą dla niniejszego rozdziału.

5.5. Podsumowanie

Po wykonaniu czynności związanych z oboma hostami *Ocenarium* powinno być gotowe do użycia.

Adres aplikacji w sieci, przy podanej konfiguracji to `www.usosphp.mimuw.edu.pl/usosR`.

Rozdział 6

Podsumowanie

Rozdział ten jest rachunkiem sumienia tego co udało się zrobić, czego nie udało się zrobić lub na co nie starczyło czasu.

6.1. Wstęp

Zacznijmy od tego, że projekt ma duży potencjał. Wiąże się to z bogatym źródłem danych jakim jest USOS. Jak wiadomo nie był on do tej pory eksplorowany w kierunkach w jakich robi to *Ocenarium*. Sporym atutem projektu jest też docelowa grupa odbiorców (por. rozdz. 1.6). Właściwie każdy użytkownik USOS-a ma własne sugestie, przemyślenia o tym jakie informacje nie są upublicznione, a które z chęcią by sprawdził. Odzwierciedleniem tej tezy jest fakt, że podczas prezentacji na *Seminarium badawczym Zakładu Statystyki Matematycznej*, seminarium *Wybranych aspektów oprogramowania* i proseminarium *Zastosowania statystyki w biologii i medycynie* odzew, entuzjazm i mnogość uwag wśród słuchaczy były istotne. Słowem, zainteresowanie rozwojem projektu potencjalnych użytkowników potwierdza zasadność rozwijania *Ocenarium*. Prawdopodobnie stanie się tak ponieważ są osoby chętne pisać pracę magisterską na ten temat.

6.2. To czego nie udało się zrobić. Ścieżki rozwoju projektu

Pora przejść do meritum. Podamy teraz obszary, gdzie *Ocenarium* wymaga pracy. Podstawowym aspektem jest tutaj bardzo prosta architektura. Nie ma w niej miejsca na interakcję z użytkownikie, a wszystkie zasoby są statyczne. Poniekąd wiąże się to z tym że *raporty* generowane są offline. Jednak niektóre z nich np. *Rankingi* (4.5) z powodzeniem mogłyby być generowane na bieżąco z parametrami przekazywanymi przez użytkownika (liczba wyświetlanych w rankingu prowadzących, typ zajęć). Innymi słowy najlepsze rozwiązanie byłoby hybrydowe. W zależności od złożoności obliczeniowej, obciążenia systemu czy innych czynników zapytanie do *Ocenarium* mogłoby być realizowane online lub offline.

Krokiem który otworzyłby nowe możliwości przed *Ocenarium* niewątpliwie byłoby dodanie nowego źródła danych jakim jest IRK. Wymagałoby to jednak dużej pracy.

Wiąże się ona z faktem, iż IRK co roku jest startowane od nowa (z *czystą* bazą danych), a archiwalne edycje są składowane na nośnikach zewnętrznych (na płytach CD). Oprócz tego inny – w stosunku do USOS-a – jest sposób zasilania hurtowni danymi z IRK (migracja Migratorem 2.4.3). Te problemy mogą rzutować na czasochłonność tego rozszerzenia. Jednak w zamian dostajemy dodatkowy wymiar danych dotyczący studentów. Są to mianowicie wyniki matur które w zestawieniu z ocenami ze studiów dają podstawy do *raportu*, który znajduje się w obszarze zainteresowań Komisji dydaktycznej (korelacja pomiędzy wynikiem matury, a wynikami na studiach). Dodatkowo, to poprzez IRK prowadzona jest rejestracja kandydatów na studia drugiego stopnia. Te dane również warto uwzględnić.

Wymienione mankamenty w naturalny sposób wytyczają możliwe kierunki rozwoju *Ocenarium*.

Zastanówmy się jeszcze chwilę nad kolejnymi, potencjalnych funkcjonalnościami *Ocenarium*. Naturalnym rozszerzeniem jest dodanie personalizacji i integracja systemu z Centralnym Serwerem Uwierzytelniania, wykorzystującym mechanizm CAS (z ang. Centralny System Uwierzytelniania). Logowanie byłoby wtedy spójne, tzn. poprzez podanie nr. PESEL i hasła – takiego samego jak do innych uczelnianych serwisów. Dzięki temu każdy użytkownik zyskałby swoje konto. Mając zidentyfikowanego studenta moglibyśmy np. stwierdzić które z seminariów – wg naszych kryteriów – najbardziej mu pasuje. Taki ruch dawałby możliwość ewolucji *Ocenarium* w portal społecznościowy. Ciekawą informacją dla studenta mogą być wtedy nazwiska starszych studentów, którzy we wcześniejszych latach studiów rejestrowali się na podobne przedmioty co rozważany przez nas student, czy też otrzymywali podobne oceny.

Oczywiście nie zapominamy o zadaniach sformułowanych w rozdziale 1.6, gdzie wymieniamy funkcjonalności w podziale na użytkowników końcowych systemu. Nie zostały one w całości pokryte. W szczególności, w pełnym wymiarze nie został zrealizowany *raport Studenci*, będący na początku zadaniem o największym priorytecie.

6.3. To co udało się zrobić

Najważniejszym celem projektu *Ocenarium* było wykonanie wartościowych i zróżnicowanych tematycznie *raportów*. Zamierzenie to zostało zrealizowane, a *raporty* dotyczą większej liczby dziedzin niż tytułowe *Oceny* z *Ocenarium* (por. rozdz. 4).

Warto wspomnieć że efektem ubocznym *raportu seminaria* (4.3) są plakaty przygotowane przez dr. Przemysława Biecka zachęcające do świadomego wyboru seminarium (proseminarium). Zostały one wywieszone na Wydziale MIM UW w jednej z gablotek na korytarzu, zaś informacja o tej inicjatywie pojawiła się na stronie samorządu ([21]) i w emailu do wszystkich studentów.

Bibliografia

[1] Janina Mincer-Daszkiewicz i in.. *Strona domowa projektu USOS.* http://usos.edu.pl

[2] Eric Lecoutre. *R2HTML: HTML exportation for R objects* http://cran.r-project.org/web/packages/R2HTML/index.html 2010.

[3] *Django documentation* https://docs.djangoproject.com/en/dev/

[4] *The Django Book.* Chapter 4: Templates. http://www.djangobook.com/en/2.0/chapter04/

[5] Grzegorz Kukawski *Uniwersytecki System Obsługi Studiów. Hurtownia danych dla systemu Statystyki.* Międzyuniwersyteckie Centrum Informatyzacji, USOS 3.12, 17 grudnia 2008.

[6] Grzegorz Kukawski. *Uniwersytecki System Obsługi Studiów. Hurtownia danych dla systemu Statystyki.* Praca magisterska, 15 grudnia 2008.

[7] Cezary Bronny. *USOS – Statystyki 2006.* Praca magisterska. Wydział Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego, 2006.

[8] Jarosław Łukaszewicz. *USOS – Moduł Statystyki.* Praca magisterska. Wydział Matematyki i Informatyki UMK, Toruń, 2005.

[9] *The R Project for Statistical Computing* http://www.r-project.org/

[10] *xtable: Export tables to LaTeX or HTML.* http://cran.r-project.org/web/packages/xtable/index.html

[11] Przemysław Biecek. *Przewodnik po pakiecie R.* Oficyna Wydawnicza GiS. Wrocław 2008. ISBN 9788389020796 http://www.biecek.pl/R

[12] *RODBC: ODBC Database Access.* An ODBC database interface. http://cran.r-project.org/web/packages/RODBC/index.html

[13] *ROracle: Oracle database interface for R.* Oracle database interface (DBI) driver for R. http://cran.r-project.org/web/packages/ROracle/index.html

[14] *IRK.* Strona domowa systemu IRK (wymaga zalogowania). http://old.usos.edu.pl/IRK/

[15] Anna Olczak. *USOS – model info.* USOS 5.1, ver. 1.0, MUCI, 2011-01-18

[16] Erik Thomsen. *OLAP Solutions: Building Multidimensional Information Systems.* Wiley, 2002

[17] Robert Wrembel; Christian Koncilia. *Data Warehouses and OLAP: Concepts, Architectures and Solutions.* IRM Press, 2007

[18] Maciej Hadyś. *Jednokierunkowa replikacja danych między bazami Oracle.* 2008

[19] Wojciech Rygielski. *Migrator 2.0 w USOS. Wprowadzenie dla administratorów.* UW, 11.10.2010

[20] *Pentaho Data Integration (Kettle).* Pentaho Kettle Project. A powerful, meta-data-driven ETL tool designed to bridge the gap between business and IT. http://kettle.pentaho.com/

[21] *samorzad[at]mimuw.edu.pl.* Strona domowa samorządu studentów Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. http://samorzad.mimuw.edu.pl/2011/06/04/statystyki-seminaria/

[22] *Django's database-abstraction API.* Django documentation. https://docs.djangoproject.com/en/1.2/topics/db/aggregation/

[23] *BI.PL.* Business Intelligence Portal. http://bi.pl/

[24] *How to install Django.* Django documentation. https://docs.djangoproject.com/en/1.2/topics/install/

[25] *How to use Django with Apache and mod_wsgi.* Django documentation. https://docs.djangoproject.com/en/1.2/howto/deployment/modwsgi/

Dodatek A

Jak wykonać nowy raport?

W niniejszym dodatku prześledzimy na przykładzie jak uzupełnić bazę danych o nowe dane i utworzyć na ich podstawie raport wyświetlany w interfejsie. Raport dotyczył będzie analizy ocen z przedmiotów prowadzonych przez Wydział MIM UW.

Kolejne sekcje podają czynności jakie należy wykonać w poszczególnych modułach *Ocenarium* w celu dodania nowego *raportu*.

A.1. Baza danych

Logujemy się na maszynę *usoshurt*. Ustawiamy katalog roboczy na `/home/fg235984/STATYSTYKI/SQL`.

A.1.1. Utworzenie nowej tabeli

Zakładamy w tej części że użytkownik bazodanowy *OLAP* ma uprawnienia select do tabeli źródłowej `DZ_PRZEDMIOTY` bazy *usosbis*. Aby nadać takie uprawnienia odpowiednio uprzywilejowany użytkownik źródłowej bazy danych powinien wykonać np. takie polecenie

```
GRANT SELECT ON DZ_PRZEDMIOTY TO OLAP
```

Następnie używając klienta PL/SQL logujemy się do hurtowni jako użytkownik *STAGE*. Tworzymy nową tabelę `TMP_PRZEDMIOTY` która posiada kolumny analogiczne do źródłowej tabeli, ograniczone ewentualnie do interesującego nas podzbioru. Właściwy fragment pochodzący z pliku `02A_stage_tabele_sekwencje_synonimy.sql` to

```
CREATE TABLE TMP_PRZEDMIOTY
(
KOD   Varchar2 (20) NOT NULL,
JED_ORG_KOD Varchar2 (20) NOT NULL,
JED_ORG_KOD_BIORCA Varchar2 (20) NOT NULL,
NAZWA Varchar2 (200) NOT NULL,
UTW_ID Varchar2(30) DEFAULT USER NOT NULL,
UTW_DATA DATE DEFAULT SYSDATE NOT NULL,
MOD_ID Varchar2(30),
MOD_DATA DATE,
```

```
CONSTRAINT PK_TMP_PRZEDMIOTY PRIMARY KEY(KOD)
)
```

Wygodnie jest też zdefiniować synonim tak jak w pliku `05A_STAGE.sql`

```
CREATE SYNONYM SYN_PRZEDMIOTY FOR USOS_PROD_TAB.DZ_PRZEDMIOTY@usos_db_link
```

A.1.2. Migracja danych

Następnym krokiem jest zasilenie nowych struktur danymi z bazy źródłowej. W tym celu należy zaktualizować pakiet *usos_load* z pliku `05A_STAGE.sql`. Dodajemy nową procedurę o budowie bliźniaczej z pozostałymi.

```
procedure load_przedmioty (p_good_cnt in out integer, p_bad_cnt in out integer) is
   cursor cur_przedmioty is
      select kod, jed_org_kod, jed_org_kod_biorca, nazwa, utw_id, utw_data, mod_id, mod_data
      from syn_przedmioty sp
      where (sp.utw_data > v_last_date and sp.mod_data is not null
             and sp.mod_data > v_last_date)                    -- ins, nowy rekord modyfikowany
          or (sp.utw_data > v_last_date and sp.mod_data is null)-- ins, nowy rekord niemodyfikowany
          or (sp.utw_data < v_last_date and sp.mod_data is not null and sp.mod_data > v_last_date);
          -- upd, nowy rekord modyfikowany
   -- kursor szukajacy rekordow do usuniecia - sa po stronie hurtownii, nie ma po stronie ususa
   cursor cur_tmp_przedmioty_del is
      select kod
      from tmp_przedmioty o
      where not exists (select 1
                        from syn_przedmioty
                        where kod = o.kod and rownum = 1);

begin
   for r in cur_przedmioty loop
      begin
         -- Dodawanie nowych rekordow
         if (r.utw_data > v_last_date and nvl (r.mod_data, (sysdate + 1)) > v_last_date) then
            insert into tmp_przedmioty
                        (kod, jed_org_kod, jed_org_kod_biorca, nazwa,
                         utw_id, utw_data, mod_id, mod_data)
                 values (r.kod, r.jed_org_kod, r.jed_org_kod_biorca, r.nazwa,
                         r.utw_id, r.utw_data, r.mod_id, r.mod_data);
         else
         -- Aktualizaowanie nieaktualnych
            update tmp_przedmioty
               set kod = r.kod,
                   jed_org_kod = r.jed_org_kod,
                   jed_org_kod_biorca = r.jed_org_kod_biorca,
                   nazwa = r.nazwa,
                   utw_id = r.utw_id,
                   utw_data = r.utw_data,
                   mod_id = r.mod_id,
                   mod_data = r.mod_data
             where kod = r.kod;
         end if;

      good (p_good_cnt);
      exception
         when others then
            log_failed_rows (null, 'przedmioty', sqlerrm);
            bad (p_bad_cnt);
      end;
```

```
   end loop;
   begin
      -- Usuwanie starych rekordow
      for r in cur_tmp_przedmioty_del loop
         delete from tmp_przedmioty
                where kod = r.kod;
      end loop;

      good (p_good_cnt);
   exception
      when others then
         log_failed_rows (null, 'przedmioty', sqlerrm);
         bad (p_bad_cnt);
   end;

exception
   when others then
      log_failed_proc ('tmp_przedmioty', sqlerrm);
end load_przedmioty;
```

W przypadku mniejszych tabel można po prostu wyczyścić tabele poleceniem *truncate* i skopiować wszystkie rekordy.

```
procedure load_przedmioty is
begin
  EXECUTE IMMEDIATE 'truncate table TMP_PRZEDMIOTY';
  insert into TMP_PZEDMIOTY(KOD, JED_ORG_KOD, JED_ORG_KOD_BIORCA, NAZWA,
                            UTW_ID, UTW_DATA, MOD_ID, MOD_DATA)
  select KOD, JED_ORG_KOD, JED_ORG_KOD_BIORCA, NAZWA, UTW_ID, UTW_DATA, MOD_ID, MOD_DATA
  from SYN_PRZEDMIOTY;
  commit;
end;
```

Do funkcji *load* dodajemy wywołanie zdefiniowanej wcześniej funkcji migrującej.

```
load_przedmioty (v_good_cnt, v_bad_cnt);
```

Jeśli chcemy znacznie przyspieszyć migrację i ograniczyć się tylko do nowej tabeli możemy zakomentować wywołania pozostałych funkcji typu *load_**.

Uwaga A.1.1. *W tabeli vars znajduje się zmienna migr_data. Na jej podstawie migrowane są tylko wybrane rekordy z bazy źródłowej. Hurtownia jest zbudowana w ten sposób że dodawanie nowych tabel wymaga "zerowania" tej zmiennej. Innymi słowy definicje tabel i pakietu migrującego są określane jako zamknięta całość, na początku, przy inicjowaniu struktur hurtowni. Zatem chcąc je modyfikować jesteśmy zmuszeni do stosowania tego typu metod.*

```
truncate table vars;
INSERT INTO VARS (VAR, VALUE, DESCRIPTION)
            VALUES ('migr_data', '1950/01/01','Data ostatniej migracji');
```

Jeśli zmienna *migr_data* jest potrzebna można ją zapisać i przywrócić po zakończeniu migracji.

Aktualizujemy pakiet i uruchamiamy procedurę migrującą skryptem postaci:

```
@02A_stage_tabele_sekwencje_synonimy.sql
@05A_STAGE.sql
@run_usos_migr.sql
-- test czy ok
select * from vars;
select * from log_load;
```

gdzie *run_usos_migr.sql* jest postaci

```
declare
  v_wynik varchar2(2500);
begin
  v_wynik := USOS_LOAD.LOAD;
  dbms_output.put_line('blad migratora usos_load: '||v_wynik);
end;
```

Wynik migracji możemy sprawdzić wyświetlając zawartość tabel

```
SQL> select * from log_load;

        ID    NR_MIGR SRC_ROW_ID TABLE_NAME  TEXT                          UTW_ID UTW_DATA
---------- ---------- ---------- ----------- ----------------------------- ------ -------------
   5540979        100                        PODSUMOWANIE MIGRACJI:        STAGE  01-APR-11
                                             Wczytane: 49978 Bledne: 0

SQL> select * from log_load_critical;

no rows selected

SQL> select * from vars;

VAR         VALUE           DESCRIPTION
----------- --------------- ------------------------------
migr_data   2011/05/04      Data ostatniej migracji

SQL> select kod, nazwa from tmp_przedmioty;

KOD              NAZWA
---------------- ------------------------------------------------------
3600-3-IN1-JH2.2 Język hindi2.2
4023-Z-JOGA      Joga
3101-DW0030      Miasta antycznego Wschodu
3101-DW0032      Piśmiennictwo i literatura Starożytnego Wschodu
3101-DB221X      Historia archeologii Ameryki
1000-135GM2      Geometria II
...              ...
...              ...
```

Uwaga A.1.2. *Powyższe kroki dotyczyły tylko jednej tabeli TMP_PRZEDMIOTY. Jak się okaże w następnej sekcji potrzebne nam również będą tabele TMP_OCENY, TMP_WARTOSCI_OCEN, TMP_OSOBY, TMP_PROTOKOLY. Jednak schemat postępowania jest w ich przypadku analogiczny.*

A.2. Skrypt R

Przystępujemy do napisania skryptu generującego *raport*. Będzie to dokument HTML z wykresami.

Logujemy się na maszynę *usosphp*. Zmieniamy katalog na `/home/users/fg235984/usosR/R/przedmiot`. Na tym etapie mamy już dostęp – poprzez hurtownie – do nowych danych. Zatem możemy napisać korzystający z nich skrypt. Omawiany w tej sekcji schemat skryptu jest powielany w większości źródeł **.R* naszego projektu. Za przykład posłuży nam plik `usosR/R/przedmiot/przedmiot_details.R`, ale z uwagi na jego dużą objętość nie prześledzimy całej zawartości. Skupimy się na kluczowych fragmentach. Poniżej znajduje się lista pozycji które składają się na strukturę skryptu.

- Wczytujemy zmienne połączenia z bazą danych z pliku konfiguracyjnego

```
library(Ocenarium)
db.lst <- readConfigFile(config.file = "../../default.conf")
```

- Ustalamy katalog do którego trafiać będą pliki raportu (html, png)

```
output_dir <- path.expand(paste(getwd(), przedmiot_html_str, sep = ""))
```

- Uruchamiamy sterownik urządzenia graficznego dla X Window System

```
X11() # BATCH MODE
```

- Definiujemy zapytania do hurtowni

```
przedmiot.query <- function(przedmiot.kod.pattern = "1000-114aRP1*") {
  paste(
"select 'os_id'||'|'||'plec'||'|'||'data_ur'||'|'||'cdyd_kod'||'|'||'prz_kod'
||'|'||'nazwa'||'|'||'przed_jed_org_kod'||'|'||'tpro_kod'
||'|'||'term_prot_nr'||'|'||'toc_kod'||'|'||'wartosc'||'|'||'opis'
from DUAL;

select oceny.os_id||'|'||osoby.plec||'|'||TO_CHAR(osoby.data_ur, 'YYYY-MM-DD')
||'|'||prot.cdyd_kod||'|'||przed.kod||'|'||przed.nazwa||'|'||przed.jed_org_kod
||'|'||prot.tpro_kod||'|'||oceny.term_prot_nr||'|'||oceny.toc_kod
||'|'||wartosci.wartosc||'|'||wartosci.opis
from TMP_OCENY oceny, TMP_WARTOSCI_OCEN wartosci, TMP_OSOBY osoby,
     TMP_PROTOKOLY prot, TMP_PRZEDMIOTY przed
where  oceny.wart_oc_kolejnosc = wartosci.kolejnosc
and    oceny.toc_kod = wartosci.toc_kod
and    oceny.os_id = osoby.id
and    oceny.prot_id = prot.id
and    przed.kod = prot.prz_kod
and    prz_kod like '", przedmiot.kod.pattern, "';",
    sep = ''
  )
}

przedmiot.query <- przedmiot.query(przedmiot.kod.pattern = kod_przedmiotu)
oceny <- executeSqlQuery(db.lst$user, db.lst$pass, db.lst$db, przedmiot.query,
                         sep = '|', header = TRUE)
```

  Więcej informacji o składni funkcji `executeSqlQuery()`, jej argumentach i przykładach użycia można znaleźć w dokumentacji tej funkcji (wpisując `?executeSqlQuery` w konsolę R).

- Za pomocą funkcji `HTML()`, `HTMLplot()` z biblioteki `R2HTML` ([2]) zapisujemy do wynikowego HTML-a elementy raportu (wykresy, tabele, tekst), np.

```
interaction.plot(oceny_num[,"cdyd_kod"], oceny_num[,"plec"],
                 grade_to_int(oceny_num[, "opis"]),
                 main = "Średnia ocen w podziale na płeć i cykl dydaktyczny",
                 xlab = "Cykl dydaktyczny", ylab = "Średnia ocen",
                 trace.label = "Płeć",
                 col = c(interaction_plot_colors[1], interaction_plot_colors[7]),
                 cex = 1.5, lty = 1.8)
img_name <- paste("wykres2-", gsub("/", "$", kod_przedmiotu), sep = "")
HTMLplot(file = html_file, GraphDirectory = output_dir, GraphFileName = img_name,
         Caption = "", Width = IMG_W, Height = IMG_H)
```

Ostatnim krokiem jest oczywiście uruchomienie napisanego skryptu. Musimy wygenerować *raport* o przedmiotach dla wszystkich przedmiotów prowadzonych na wydziale MIM UW. Można to zrobić skryptem postaci (`scripts/generate_przedmiot_reports.sh`)

```
#!/bin/bash

TYLKO_MIM=1
Rscript=przedmiot_details.R
RscriptDir=przedmiot/
EXIT_CODE=0

TMPFILE=‘mktemp /tmp/$0.TMP.XXXXXX‘ || exit 1
LOGFILE=‘mktemp /tmp/$0.LOG.XXXXXX‘ || exit 1
FAILEDFILE=‘mktemp /tmp/$0.FAILED.XXXXXX‘ || exit 1
SPOOLFILE=‘mktemp /tmp/$0.SPOOL.XXXXXX‘ || exit 1
echo "Log is in $LOGFILE"
echo "Spool file in $SPOOLFILE"
echo "SQL script file in $TMPFILE"
echo "Raporty przedmiotów zakonczone niepowodzeniem:" > $FAILEDFILE

if [ "$TYLKO_MIM" = "1" ]
then
        KOD_PATTERN="1000%"
else
        KOD_PATTERN="%"
fi

echo connect stage/statystyki@XE > $TMPFILE
echo set echo off >> $TMPFILE
echo set feedback off >> $TMPFILE
echo set linesize 1000 >> $TMPFILE
echo set sqlprompt ’’ >> $TMPFILE
echo spool $SPOOLFILE >> $TMPFILE
echo set hea on >> $TMPFILE
echo set pagesize 0 >> $TMPFILE
echo set und off >> $TMPFILE
echo set trimspool on >> $TMPFILE
echo "select distinct kod from tmp_przedmioty where kod like ’$KOD_PATTERN’;" >> $TMPFILE
echo spool off >> $TMPFILE
echo "exit;" >> $TMPFILE
sqlplus /nolog @$TMPFILE >/dev/null

cd ../R/$RscriptDir

while read przedmiot; do
        bash -c "R CMD BATCH --vanilla ’--args kod_przedmiotu=\"$przedmiot\"’ $Rscript $LOGFILE"\
        || { let EXIT_CODE=1; echo $przedmiot >> $FAILEDFILE; }
done < $SPOOLFILE
```

```
rm -rf $TMPFILE $SPOOLFILE
cd -

exit $EXIT_CODE
```

A.3. Interfejs

Przechodzimy do prezentacji wygenerowanego raportu w interfejsie. W tym miejscu zakładamy, że mamy już wygenerowany *raport*. Pierwsza czynnością jest dodanie nowej aplikacji Django. Nazwijmy ją `stats`.

```
python manage.py startapp stats
```

Pliki które należy edytować są wypunktowane poniżej. Schemat postępowania – dla czytelników znających Django – powinien być oczywisty. Ponadto, zadaniem niniejszego dodatku nie jest szczegółowe omawianie sposobu budowania aplikacji we frameworku Django dlatego ograniczymy się poniżej tylko do rozwiązań, które nie pojawiają się w popularnych opracowaniach. Dokładniejszych wskazówek technicznych można zasięgnąć np. w [3].

- `/home/users/fg235984/usosR/urls.py` – dodajemy do definicji zmiennej `urlpatterns` nową linię

  ```
  urlpatterns = patterns('',
      ...
      (r'^stats/', include('usosR.stats.urls')),
      ...
  )
  ```

- `/home/users/fg235984/usosR/stats/urls.py` – ustawiamy zmienna `urlpatterns` na

  ```
  urlpatterns = patterns('usosR.stats.views',
      url(r'^$', 'index'),
      url(r'^(?P<przedmiot_kod>[a-zA-z0-9\-\*]*)/$', 'detail'),
  )
  ```

  Dwie powyższe akcje mają na celu wywołanie widoku `index` lub `detail` (z argumentem `przedmiot_kod`) w zależności od wpisanego do przeglądarki adresu (mapowanie linków na odpowiednie strony do wyświetlenia).

- `/home/users/fg235984/usosR/stats/models.py` – dodajemy nowa klasę reprezentującą przedmiot

  ```
  class TmpPrzedmioty(models.Model):
      kod = models.CharField(unique=True, max_length=20, primary_key=True)
      jed_org_kod = models.CharField(max_length=20)
      jed_org_kod_biorca = models.CharField(max_length=20)
      nazwa = models.CharField(max_length=200)
      utw_id = models.CharField(max_length=30)
      utw_data = models.DateField()
      mod_id = models.CharField(max_length=30, blank=True)
  ```

```
mod_data = models.DateField(null=True, blank=True)
class Meta:
    db_table = u'tmp_przedmioty'
```

Uzupełnienie modelu jest nam potrzebne by w następnym punkcie móc wyświetlić listę przedmiotów w interfejsie (widok `index` niżej).

- `/home/users/fg235984/usosR/stats/views.py` – w tym pliku dodajemy definicje dwóch widoków `index` i `detail`. Pierwszy z nich korzysta z template'u `/home/users/fg235984/usosR/templates/stats/index.html`, który wyświetla w postaci listy przedmioty prowadzone na Wydziale MIM UW.

  Implementacja widoku `detail` polega na wczytaniu *raportu* (statycznego pliku HTML) wygenerowanego dla przedmiotu określonego przez parametr `przedmiot_kod`. Następnie poprawiane są w nim wszystkie odnośniki. Tak zmodyfikowany *raport* jest wyświetlany w przeglądarce.

  Adresy do obu widoków to odpowiednio `usosphp.mimuw.edu.pl/usosR/stats/` i `usosphp.mimuw.edu.pl/usosR/stats/przedmiot_kod`, gdzie `przedmiot_kod` jest kodem dowolnego przedmiotu prowadzonego na Wydziale MIM UW.

- `/home/users/fg235984/usosR/templates/base.html` – dodajemy do menu interfejsu www pozycję

  ```
  <li><a href="/usosR/stats/">Przedmioty</a></li>
  ```

- `/home/users/fg235984/usosR/usosR.wsgi` – każda zmiana dotycząca interfejsu jest widoczna dopiero po zmianie czasu ostatniej modyfikacji tego pliku na aktualny czas. Można tego dokonać np. za pomocą polecenia `touch`.

  Wynik końcowy, tj. *raport Przedmioty* można obejrzeć na rysunkach w rozdziale 4.1.

Dodatek B

Nowe tabele w schemacie STAGE

W tym rozdziale pokazujemy listę tabel które są tworzone przez skrypt `02A_stage_tabele_sekwencje_synonimy.sql`. Ich nazwy i opisy kolumn są na tyle rozwinięte, że pominiemy dodatkowy komentarz. Zgodnie z konwencją z wcześniejszych wersji hurtowni nazwa każdej tabeli kopiowanej z USOS-a zaczyna się od prefiksu `TMP_`.

Nazwa kolumny	**Opis**
Nazwa tabeli: `TMP_PRZEDMIOTY`	
KOD	Kod przedmiotu
JED_ORG_KOD	Kod jednostki organizacyjnej oferującej przedmiot
JED_ORG_KOD_BIORCA	Kod jednostki organizacyjnej, dla której prowadzony jest przedmiot
NAZWA	Nazwa przedmiotu
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: `TMP_OCENY`	
PROT_ID	Identyfikator protokołu
TERM_PROT_NR	Numer terminu protokołu
OS_ID	Identyfikator osoby
TOC_KOD	Kod typu oceny
WART_OC_KOLEJNOSC	Kolejność wartości oceny
ZMIANA_DATA	Data ostatniej modyfikacji oceny (migrowane z USOSweb)
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu

Nazwa kolumny	Opis
Nazwa tabeli: TMP_WARTOSCI_OCEN	
KOLEJNOSC	Kolejność wartości oceny
TOC_KOD	Kod typu oceny
OPIS	Kolejność wartości oceny
CZY_ZAL	Data ostatniej modyfikacji oceny (migrowane z USOSweb)
WARTOSC	Wartość przy obliczaniu średnich
OPIS_OCENY	
Nazwa tabeli: TMP_PROTOKOLY	
ID	Identyfikator protokołu
TPRO_KOD	Typ protokołu
PRZ_KOD	Kod przedmiotu
CDYD_KOD	Kod cyklu dydaktycznego
CZY_DO_SREDNIEJ	
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: TMP_ODPOWIEDZI	
ID	Identyfikator odpowiedzi
INST_WWW_KOD	Kod instalacji WWW, gdzie padła odpowiedź
PYTSTU_ID	Identyfikator pytania
PRAC_ID	Identyfikator prowadzącego zajęcia
ZAJ_CYK_ID	Identyfikator zajęć cyklu
GR_NR	Numer grupy
WART_ODP_ID	Identyfikator wartości odpowiedzi
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu

Nazwa kolumny	**Opis**
Nazwa tabeli: TMP_PYTANIA_DO_STUDENTOW	
ID	
ANK_KOD	
ZAJ_CYK_ID	
TZAJ_KOD	
TRESC	
TYP_ODP_KOD	Kod typu odpowiedzi
PYTSTU_ID	ID pytania nadrzędnego
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: TMP_WART_ODP_ANKIET	
ID	
TYP_ODP_KOD	Kod typu odpowiedzi
WARTOSC	Wartość odpowiedzi
OPIS	Opis wyświetlany studentowi
Nazwa tabeli: TMP_PROWADZACY_GRUP	
PRAC_ID	
ZAJ_CYK_ID	Identyfikator zajęć cyklu dydaktycznego
GR_NR	Numer grupy
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: TMP_ZAJECIA_CYKLI	
ID	Identyfikator zajęć cyklu dydaktycznego
PRZ_KOD	Kod przedmiotu
CDYD_KOD	Kod cyklu dydaktycznego
TZAJ_KOD	Kod typu zajec
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: TMP_TYPY_ZAJEC	
KOD	Kod typu zajęć
OPIS	Opis typu zajęć

Nazwa kolumny	Opis
Nazwa tabeli: TMP_PRACOWNICY	
ID	Identyfikator pracownika
OS_ID	Identyfikator osoby
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: TMP_GRUPY_PRZEDMIOTOW	
KOD	Kod grupy przedmiotów
JED_ORG_KOD	
OPIS	Opis grupy przedmiotów
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: TMP_ELEM_GRUP_PRZEDMIOTOW	
GRPRZ_KOD	Kod grupy przedmiotów
PRZ_KOD	Kod przedmiotu
CDYD_POCZ	Kod cyklu dydaktycznego, od którego przedmiot jest w grupie
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
ID	
Nazwa tabeli: TMP_PROGRAMY	
KOD	Kod programu
OPIS	Opis programu
TCDYD_KOD	Jednostka w jakiej jest wyrażony czas trwania programu
TRYB_STUDIOW	
Nazwa tabeli: TMP_ETAPY	
KOD	Kod etapu
OPIS	Opis etapu
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu

Nazwa kolumny	**Opis**
Nazwa tabeli: `TMP_ETAPY_OSOB`	
ID	Identyfikator etapu studenta
PRGOS_ID	
PRG_KOD	Kod programu
CDYD_KOD	
ETP_KOD	
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: `TMP_PROGRAMY_OSOB`	
ID	
OS_ID	Identyfikator osoby
PRG_KOD	Kod programu
ST_ID	
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu
Nazwa tabeli: `TMP_OSOBY_GRUP`	
OS_ID	Identyfikator studenta
ZAJ_CYK_ID	Identyfikator zajęć cyklu dydaktycznego
GR_NR	Numer grupy
REJ_OS_ID	Identyfikator osoby, która dokonała rejestracji
REJ_DATA	Data dokonania rejestracji
UTW_ID	Identyfikator użytkownika, który utworzył rekord
UTW_DATA	Data utworzenia rekordu
MOD_ID	Identyfikator użytkownika, który zmodyfikował rekord
MOD_DATA	Data modyfikacji rekordu

Dodatek C

Zawartość załączonej do pracy płyty CD

Do pracy załączona została płyta CD. Zawiera ona kody źródłowe *Ocenarium* oraz część dokumentów, do których najczęściej się odwołujemy w pracy i do których dostęp może być utrudniony. Poniżej wymieniamy – wraz z opisem – katalogi najwyższego rzędu i kluczowe pliki zawarte na płycie. Podajemy też jak ścieżki z płyty tłumaczą się na ścieżki na maszynach na których *Ocenarium* zostało wdrożone (por. rozdz. 5).

- `usoshurt/` – źródła dla hurtowni danych. Na serwerze `usoshurt.mimuw.edu.pl` są one umieszczone w katalogu `/home/fg235984/`
- `usosphp/` – źródła dla części statystycznej oraz interfejsu www. Na serwerze `usosphp.mimuw.edu.pl` są one umieszczone w katalogu `/home/users/fg235984/www/django/src/usosR/`
- `pdf/` – źródła niniejszego dokumentu wraz z plikiem pdf
- `bbl/kukawski2008-12-15.pdf` – praca magisterska Grzegorza Kukawskiego ([6])
- `bbl/migrator2-admin-readme.pdf` – Migrator 2.0 w USOS. Wprowadzenie dla administratorów ([19])

C.1. Lokalizacje implementacji *raportów*

Podamy teraz w jakich katalogach hosta `usosphp.mimuw.edu.pl` znajdują się implementacje poszczególnych *raportów*. Źródła są umieszczone w dwóch katalogach, pierwszy odpowiada za część statystyczną, zaś drugi za część interfejsowa. Pomijamy przy tym prefiks każdej ścieżki, tzn. `/home/users/fg235984/www/django/src/usosR/`

- *Raport przedmioty* – `R/przedmiot/`, `stats/`
- *Raport studenci* – `R/student/`, `students/`

- *Raport seminaria* – `R/seminaria/`, `seminaria/`
- *Raport prowadzący* – `R/prowadzacy/`, `prowadzacy/`
- *Raport rankingi* – `R/seminaria/`, `rankingi/`

Dodatek D

Historia zmian

```
$Log: mgr.tex,v $
Revision 1.2 2011/08/18 22.40   Poprawki w stylistyce pracy cz. II i zmiany w architekturze
Revision 1.1 2011/07/5  13.20   Poprawki w stylistyce pracy cz. I.
Revision 1.0 2011/06/26 13.00   Uzupełnienie streszczenia pracy. Nowa sekcja w dodatku C
                                (Lokalizacje implementacji raportów).
Revision 0.2 2011/06/20 16.15   Wprowadzenie poprawek w rozdziałach 5. i 6.
                                Nowy dodatek o zawartości załączonej do pracy płyty CD.
Revision 0.1 2011/06/18 18.20   Podział na sekcje rozdziału Podsumowanie.
                                Dodanie ścieżek rozwoju projektu (rozdział 6.2. "To czego
                                nie udało się zrobić. Ścieżki rozwoju projektu").
```