

**PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA
W NOWYM SĄCZU**

INSTYTUT TECHNICZNY

PRACA DYPLOMOWA

**SYSTEM GENEROWANIA RAPORTÓW W PROCESIE
REKRUTACJI KANDYDATÓW NA STUDIA PROWADZONE W
PWSZ W NOWYM SĄCZU.**

Autor: Paweł Mysiński

Kierunek: Informatyka

Nr albumu: 20747

Promotor: dr inż. Antoni Ligęza

NOWY SĄCZ 2015

Spis treści

1. Wprowadzenie	4
1.1. Zagadnienie generowania raportów	4
1.2. Dotychczasowy proces generowania raportów	4
1.3. Cel i zakres pracy	4
2. Szablony raportów w systemie Latex	5
2.1. Środowisko kompilacji raportów	5
2.2. Idea działania szablonów	5
2.3. Tworzenie szablonów raportów do systemu rekrutacji	5
2.3.1. Wyświetlanie listy	6
3. Uruchomienie oraz testowanie Systemu	7
3.1. Generowanie przykładowych danych	7
3.1.1. Tworzenie bazy danych	7
3.1.2. Stworzenie struktury bazy	8
3.1.3. Generowanie testowych danych osobowych	9
3.1.4. Generowanie kandydatów na studentów	9
3.2. Dodanie zapytań SQL do szablonów	11
3.3. Przebieg generowania raportów	14
3.4. Wyniki testu	14
Bibliografia	15

1. Wprowadzenie

testtesttest

1.1. Zagadnienie generowania raportów

1.2. Dotychczasowy proces generowania raportów

1.3. Cel i zakres pracy

2. Szablony raportów w systemie Latex

2.1. Środowisko kompilacji raportów

blablabla nie mam pojęcia

2.2. Idea działania szablonów

Do wszystkich tych dokumentów potrzebny jest szablon w języku oprogramowania do zautomatyzowanego składu tekstu. W tej pracy został wybrany program LaTeX ze względu na jego możliwości automatyzacji procesu parsowania danych i uzupełniania nimi danych miejsc w tekście.

Stworzenie szablonów polega, więc na wcześniejszym przygotowaniu plików tex, zawierających wcześniej strukturę danego dokumentu z "pustymi" miejscami do uzupełnienia przez program. Do uzupełnienia tych miejsc można wykorzystać funkcję LaTeXu jaką jest tworzenie nowych środowisk z parametrami, gdzie odpowiednio parametry te będą wartościami, które zostaną wpisane w dane miejsce w danym dokumencie. Następnie wystarczy wywołać dane środowisko z odpowiednimi wartościami aby otrzymać uzupełniony dokument. Dane środowisko możemy wywoływać wielokrotnie od różnych wartości tworząc w ten sposób wiele dokumentów tego samego typu o różnych zmiennych wartościach takich jak np imię i nazwisko.

Do wytworzenia wywołań tych środowisk posłuży właśnie program stworzony w javie. Poprzez dodanie zapytania SQL w odpowiedniej formule do plików tex. Program DBRaportLatex wyszuka takie zapytanie i uzupełni szablon wywołaniami środowisk z wartościami parametrów, jakimi będą wartości pola z rekordów zapytania SQL.

2.3. Tworzenie szablonów raportów do systemu rekrutacji

W rekrutacji na uczelnie wykorzystuje się dokumenty, które należało dokładnie odwzorować w nowym systemie. Są to następujące dokumenty:

1. protokół przekazania
2. listy potwierdzenia podjęcia studiów
3. listy rankingowe
4. listy przyjętych
5. listy nieprzyjętych
6. decyzja o przyjęciu danego kandydata
7. decyzja o nieprzyjęciu danego kandydata

Przy tworzeniu szablonów wystąpiły powtarzające się problemy, które należało rozwiązać. Jako że rozwiązania tych problemów powtarzają się, to zamiast opisywania każdego

szablonu po kolei, przedstawione poniżej zostały najważniejsze problemy, wynikające z tworzenia tych szablonów.

2.3.1. Wyświetlanie listy

Problem występujący w protokole przekazania oraz we wszystkich listach. Przykładowo potrzebujemy wyświetlić poniższą listę, gdzie oczywiście wartości pochodzą z bazy danych:

```
Lp. Tok studiów Liczba kopert
1 Informatyka -- niestacjonarne STUDIA pierwszego stopnia 1455
2 Informatyka -- stacjonarne STUDIA pierwszego stopnia 729
3 Mechatronika -- niestacjonarne STUDIA pierwszego stopnia 1447
...
```

3. Uruchomienie oraz testowanie Systemu

Przed wdrożeniem programu do realnego systemu, program należy przetestować. Testy powinny być prowadzone na tymczasowej bazie danych, ponieważ idea testów jest taka, że po podmianie bazy danych na realną wszystko ma działać bez zmian. Zmienić się może tylko zapis połączenia z bazą danych w pliku konfiguracyjnym. Dzięki takiemu zabiegowi, będzie można być pewnym tego, że wszystko będzie działać na prawdziwej bazie danych. Do przeprowadzenia testów potrzebne będzie odtworzyć przyszłe środowisko, w którym będzie działać program, przygotować szablony dokumentów, które są wytwarzane w procesie rekrutacji oraz wygenerować dokumenty. Ostatnim już krokiem będzie sprawdzenie czy podczas tego procesu nie ma żadnych komplikacji oraz czy wygenerowane dokumenty nie zawierają błędów.

3.1. Generowanie przykładowych danych

W celu przetestowania systemu generowania raportów w procesie rekrutacji kandydatów na studia potrzebne będą testowe dane w dokładnie tej samej strukturze co w systemie rekrutacji, ponieważ w szablonach latexu znajdują się zapytania SQL do danych tabel w bazie danych. W celu otrzymania tych danych potrzebne będzie: 1. Utworzenie nowej bazy danych na silniku Firebird'a 2. Stworzenie wystarczającej struktury tabel odzwierciedlającą strukturę w systemie uczelnianym. 3. Wygenerowanie dużej ilości testowych danych osobowych. 4. Uzupełnienie tabel danymi, które zostały wygenerowane wcześniej oraz dodanie do nich dodatkowych, jednocześnie losowych, informacji na temat procesu rekrutacji. Po wykonaniu tych kroków, powinna powstać baza danych do której bez problem program połączy się i wyciągnie z niej potrzebne dane dokładnie jak z realnej bazy danych.

3.1.1. Tworzenie bazy danych

Do stworzenia pliku bazy danych na silniku firebird'a posłużyć się można narzędziem dostępnym w katalogu bin zainstalowanego serwera firebird. Narzędzie to pozwala z linii komend tworzyć i łączyć się z bazami danych. W tym przypadku użyta zostanie komenda „CREATE DATABASE”

```
C:\Program Files\Firebird\bin>isql
SQL>CREATE DATABASE 'D:\test_systemu\rekrutacja.fdb'
CON>user 'SYSDBA' password 'masterkey';
```

Po wykonaniu tego polecenia zostanie utworzona baza danych. Takie same dane należy teraz wpisać do pliku konfiguracyjnego programu czyli DBRaportLatex.bat aby program mógł się połączyć z tą bazą:

```
dbengine=firebirdsql
hostname=//localhost
port=3050
```

```
dbpath=D:\test systemu\rekrutacja.fdb
user=SYSDBA
password= masterkey
```

3.1.2. Stworzenie struktury bazy

Dotychczasowy system wykorzystywał tabelę (widok) która była generowana dynamicznie i która zawiera wszystkich studentów w rekrutacji. Zawiera ona wszystkie dane potrzebne do wytworzenia dokumentów. Jeden rekord to jeden student ze wszystkimi informacjami na jego temat. Dodatkowo jeszcze potrzebna jest tabela z informacjami na temat rekrutacji, takimi jak na przykład nazwisko i imię przewodniczącego komisji, czy data wydania decyzji przyjęcia studenta. Z tych tabel będą pobierane informacje, natomiast do wygenerowania danych potrzebne będą dwie dodatkowe tymczasowe tabele. Tabela główna z kandydatami(zapis SQL):

```
CREATE TABLE KANDYDAT_ALIGEZA (
    STUD_ID          INTEGER NOT NULL,
    OSOBA_ID         INTEGER NOT NULL,
    STUD_NRTECZKI    INTEGER,
    NAZWISKO         VARCHAR(100),
    IMIE             VARCHAR(100),
    NAZWISKOIMIONA   VARCHAR(200),
    ADR_ULICA_MIEJSCOWOSC_NR_DOMU VARCHAR(200),
    ADR_KOD_POCZTOWY_POCZTA VARCHAR(100),
    STUDIA_NAZWA     VARCHAR(100),
    STUD_ILPUNKTOW   INTEGER,
    STUD_ILPUNKTOWKREM INTEGER,
    TOKNAUKI_NAZWATOKU VARCHAR(200),
    KIERUNEK         VARCHAR(100),
    SPEC_ID          INTEGER,
    DATAPRZYJECIAPODANIA DATE,
    TOKNAUKI_ID      INTEGER,
    OSOBA_PESSEL     VARCHAR(50),
    KIERUNEK_MY      VARCHAR(200),
    FORMA_STUDIOW_MY VARCHAR(200),
    STOPIEN_STUDIOW_MY VARCHAR(100),
    KIERUNEK_FORMA_SKROT_MY VARCHAR(100),
    NR_DECYZJI       VARCHAR(100),
    CZY_PRZYJETY     INTEGER,
    DATA_DECYZJI    DATE,
    ILE_PUNKTOW      INTEGER,
    PANPANI          CHAR(1)
);
```

Tabela z dodatkowymi informacjami(wartości przypisane są do kluczy tekstowych, jest to tablica asocjacyjna):

```
CREATE TABLE SETUP_ALIGEZA (
    KLUCZ    VARCHAR(50) NOT NULL,
    WARTOSC  VARCHAR(100)
```


);

Tymczasowa tabela do zaimportowania listy imion i nazwisk oraz losowych peseli.

```
CREATE TABLE DANE (  
    IMIE_NAZ VARCHAR(200),  
    ADRES VARCHAR(200),  
    PESEL VARCHAR(50),  
    NAZWISKO VARCHAR(100),  
    IMIE VARCHAR(100)  
);
```

Tymczasowa tabela do procedury losowego uzupełniania informacji o rekrucie o kierunku jaki wybrał.

```
CREATE TABLE TOKNAUKI_ALIGEZA (  
    TOKNAUKI_ID INTEGER NOT NULL,  
    KIERUNEK_MY VARCHAR(50),  
    FORMA_STUDIOW_MY VARCHAR(50),  
    STOPIEN_STUDIOW_MY VARCHAR(50),  
    KIERUNEK_FORMA_SKROT_MY VARCHAR(10),  
    LICZBA_MIEJSC SMALLINT,  
    DATA_DECYZJI_OD TIMESTAMP,  
    DATA_DECYZJI_DO TIMESTAMP,  
    KOD_IKR VARCHAR(3)  
);
```

3.1.3. Generowanie testowych danych osobowych

Do wygenerowania kandydatów potrzeba imienia nazwiska oraz adresu. Takie dane dostępne są w książkach telefonicznych. Posługując się jedną z takich książek stworzony został plik csv o separatorze „;” zawierający po kolei imię z nazwiskiem, adres, pesel, nazwisko oraz imię. Pesel został dodany do każdej osoby jako losowy ciąg cyfr spełniający walidację peselu. Ze względu na fakt iż pesel został wygenerowany losowo, może zdarzyć się iż mężczyzna posiadać będzie kobiecy pesel, w następstwie czego, we wygenerowanych dokumentach wypisane zostanie „Pani” i na odwrót. Struktura pliku:

```
Abram Andrzej; Lwowska 116;88071640299;Abram;Andrzej  
Abram Halina; Ludwika Zamenhofa 2;86111210691;Abram;Halina  
...
```

Tak sformatowany plik CSV, łatwo zaimportować do bazy danych do tabeli „dane” ze względu na identyczną kolejność danych w kolumnach. Do importu wykorzystana została funkcja programu IBExpert „import data”. Jedna linijka w pliku zostaje zaimportowana jako jeden rekord, w którym każde pole po kolei odpowiada wartościom między średnikami. Zaimportowanych w ten sposób zostało 10001 rekordów (osób) do tabeli „dane” do dalszych manipulacji.

3.1.4. Generowanie kandydatów na studentów

Kolejnym krokiem jest uzupełnienie tabeli z tokami studiów. W testach dodanych zostało 8 przykładowych toków nauki.

1 Informatyka	niestacjonarne	pierwszego stopnia	INF–n
2 Mechatronika	niestacjonarne	pierwszego stopnia	MT–n
3 Mechatronika	stacjonarne	pierwszego stopnia	MT–s
...			

Uzupełnienia wymaga także tabela z dodatkowymi informacjami „SETUP_ALIGEZA” przykładowymi danymi:

```

dataWydaniaDecyzji      09.10.2015
miejsceWydaniaDecyzji   Nowy Sącz
przewodniczącyIKR       mgr inż. Sławomir Jurkowski
rokAkademicki           2015/2016
czyUwzględnicDateWydaniaDecyzji N
...
```

Mając już to wszystko potrzebna jest procedura, która utworzy listę kandydatów z tych wszystkich danych.

```

create procedure GENERUJ
returns (
    TESTCHAR varchar(50),
    TEST integer)
as
declare variable IMIE varchar(100);
declare variable NAZ varchar(100);
declare variable IMIENAZ varchar(200);
declare variable ADRES varchar(200);
declare variable PESEL varchar(50);
declare variable LICZNIK integer;
declare variable STOPIEN varchar(50);
declare variable KIERUNEK varchar(50);
declare variable FORMA varchar(50);
declare variable SKROT varchar(10);
declare variable RANDINT integer;
declare variable PUNKTY integer;
declare variable CZY_PRZYJETY integer;
declare variable DATA_DEC varchar(100);
begin
    licznik = 1;
    for select * from dane into
    :imienaz, :adres, :pesel, :naz, :imie
    do
    begin
        randint = CAST(round(rand()*7+1) as INTEGER);
        punkty = CAST(round(rand()*500) as INTEGER);
        if(punkty > 250) then czy_przyjety = 1;
        if(punkty <= 250) then czy_przyjety = 2;

        select kierunek_my, forma_studiow_my, stopien_studiow_my,
        kierunek_forma_skrot_my
```

```

FROM toknauki_aligeza where toknauki_id = :randint
INTO :kierunek,:forma,:stopien,:skrot;

select wartosc FROM setup_aligeza
WHERE klucz='dataWydaniaDecyzji'
INTO :data_dec;

INSERT INTO kandydat_aligeza
(stud_id,osoba_id,stud_nrteczki,nazwisko,imie,
nazwiskoimiona,adr_ulica_miejscowosc_nr_domu,
adr_kod_pocztowy_poczta,osoba_pesel,panpani,
studia_nazwa,toknauki_nazwatoku,kierunek,
kierunek_my,forma_studiow_my,stopien_studiow_my,
kierunek_forma_skrot_my, stud_ilpunktow,
stud_ilpunktowkrem,ile_punktow,czy_przyjety,
data_decyzji,nr_decyzji,dataprzyjeciapodania)
VALUES (:licznik,:licznik,cast(round(rand()*200+1) as integer),
:naz,:imie,:imienaz,:adres,
cast( 'Nowy Sacz 33-300' as varchar(100)),
:pesel,cast( 'M' as char(1)), :forma,:kierunek ||
' N inz. 3.50 2015/2016 zimowy',:kierunek,:kierunek,:forma,
:stopien,:skrot, :punkty,:punkty,:punkty,:czy_przyjety,
cast(:data_dec as DATE),'328/2015','2015-08-14');

licznik = :licznik + 1;
end
test = :licznik;
suspend;
end

```

Powyższa procedura z jednej osoby z tabeli dane tworzy jednego kandydata, losując mu tok nauki, ilość punktów oraz czy zostanie przyjęty lub nie. Dorzucane są także pewne stałe wartości, podobne do tych w oryginalnej bazie danych, które nie wymagają uzmiennienia. Procedura ta, po jednorazowym wywołaniu, wygenerowała 10001 rekordów w tabeli "kandydat_aligeza". Daje to wystarczającą ilość testowych kandydatów do przeprowadzenia testów. Baza danych z przeprowadzonych testów znajduje się w załączniku.

3.2. Dodanie zapytań SQL do szablonów

Stworzone szablony w poprzednim rozdziale nie zawierają zapytań, które odzwierciedlałyby wyselekcjonowanie danych z istniejącej struktury bazy. Zawierają tylko sztywne dane testujące środowiska latexowe. Aby szablony podczas procesu tworzenia dokumentów zostały uzupełnione poprawnymi danymi potrzeba je uzupełnić o odpowiednia zapytania SQL w odpowiedniej formie tak aby parser zawarty w programie był w stanie je znaleźć i wykonać.

W sumie potrzeba 9 zapytań, za pomocą których wyciągnięte zostaną dane do szablonów. Poniżej opisane zostały każde z zapytań oraz przykład jakie dane zostaną wyselekcjonowane

z bazy przez to zapytanie. W tej sekcji skupimy się tylko i wyłącznie na uzyskaniu potrzebnych danych do środowisk latexu, a nie jak one zostaną wykorzystane. Dokładny opis jak wygląda wykorzystanie takiego zapisu znajduje się w rozdziale 2.

Pierwsze zapytanie które zostanie omówione jest zapytanie o podstawowe parametry, informacje na temat rekrutacji. Znajduje się ono w pliku aaasetup.tex. Plik posiada 3 litery a w nazwie ponieważ program parsujący, otwiera pliki alfabetycznie a zapytanie to musi zostać wykonane jako pierwsze. Poniżej mamy przykład danych do zapisania:

```
\parametrRekrutacyjny{dataWydaniaDecyzji}{09.10.2015}
\parametrRekrutacyjny{miejsceWydaniaDecyzji}{Nowy Sącz}
\parametrRekrutacyjny{rokAkademicki}{2015/2016}
...
```

Aby uzyskać takie dane, potrzeba będzie pobrać wszystkie rekordy z tabeli SETUP_ALIGEZA (struktura tabeli znajduje się w poprzedniej sekcji) klucz oraz wartość. Zapytanie nie będzie więc zbyt skomplikowane:

```
SELECT klucz,wartosc FROM setup_aligeza
```

Po wywołaniu go faktycznie otrzymamy dane potrzebne nam dane. Jednak aby program parsujący był w stanie znaleźć takie zapytanie w szablonie potrzeba je obudować w odpowiednią konstrukcję, która będzie odpowiadać wyrażeniu regularnemu zapisanemu w parserze. Należy także wpisać nazwę środowiska w odpowiednie miejsce między znakami @@. Dodatkowo musimy jeszcze postarać się aby kompilator latex'a zignorował nasz zapis i go nie wyświetlał. Do tego celu zastosowana została komenda iffalse.

```
\iffalse@@parametrRekrutacyjny@@
SELECT klucz,wartosc FROM setup_aligeza
@END@\fi
```

Następnie zapytanie zawarte w pliku 'protokolprzekazania.tex'. Występuje tu nowy problem otworzenia środowiska na początku oraz zamknięcia na końcu. Pokazane jest to na przykładowych danych poniżej:

```
\protokolprzekazaniaA{1}
\protokolprzekazania{Informatyka ---
niestacjonarne STUDIA pierwszego stopnia}{1455}
\protokolprzekazania{Informatyka ---
stacjonarne STUDIA pierwszego stopnia}{729}
\protokolprzekazania{Mechatronika ---
niestacjonarne STUDIA pierwszego stopnia}{1447}
...
\endprotokolprzekazaniaA
```

Pomijając narazie otwarcie środowiska protokolprzekazaniaA1 oraz zamknięcie endprotokolprzekazaniaA, w zapytaniu potrzebujemy wyświetlić każdy tok nauczania oraz z agregowaną liczbę kandydatów na tym toku. Do agregacji użyjemy polecenia group by na polu tok, a aby otrzymać liczbę kandydatów funkcji count(*). Dodatkowo potrzeba także wybrać tylko kandydatów z danej daty decyzji. Do tego posłuży podzapytanie które pobierze date decyzji z aktualnej rekrutacji i porówna ją z datą decyzji dla danego kandydata.

```
SELECT k.kierunek_my||' --- '||k.forma_studiow_my||
' STUDIA '||k.stopien_studiow_my AS tok, count(*) AS ile
```

```
FROM kandydat_aligeza k
WHERE k.data_decyzji=(SELECT wartosc FROM
  setup_aligeza WHERE klucz='dataWydaniaDecyzji')
GROUP BY tok
```

Takie zapytanie wyświetli nam dane pomiędzy tymi pominiętymi dotychczas komendami. Rozwiązaniem wyświetlenia tych komend jest funkcja programu uzupełniającego szablon, a dokładnie funkcja grupowania. Dokładny opis tej funkcji znajduje się w rozdziale ref. Do zapytania dodamy jako pierwsze pole wartość 1, a następnie użyta zostanie funkcja grupowania po 1 polu wewnątrz szablonu:

```
\iffalse @@protokolprzekazania@1@@
SELECT 1 AS n,k.kierunek_my||' --- '||
k.forma_studiow_my||' STUDIA '||
k.stopien_studiow_my AS tok, count(*) AS ile
FROM kandydat_aligeza k
WHERE k.data_decyzji=(SELECT wartosc FROM
  setup_aligeza WHERE klucz='dataWydaniaDecyzji')
GROUP BY tok
@END@\fi
```

W kolejnym szablonie 'listasekretariat.tex' potrzebujemy utworzenia list kandydatów dla każdego toku nauki. Przez tok nauki rozumie się każdą kombinację kierunku, formy oraz stopnia studiów. W szablonie zostało to rozwiązane tak, że każda lista otwiera się poprzez wywołanie komendy listarsekretariatA z trzema parametrami, którymi są właśnie kierunek, forma oraz stopień studiów. Na przykładowych danych wygląda to następująco:

```
\listarsekretariatA{drugiego stopnia}{Informatyka}{stacjonarne}
\listarsekretariat{ Ablewicz Monika}{39}
\listarsekretariat{ Abram Halina}{69}
...
\endlistarsekretariatA
\listarsekretariatA{drugiego stopnia}{Mechatronika}{stacjonarne}
\listarsekretariat{ Adamek Danuta}{141}
\listarsekretariat{ Adamek Urszula}{35}
\listarsekretariat{ Adamik Zbigniew}{95}
...
\endlistarsekretariatA
...
```

W zapytaniu więc potrzebne będzie wybrać wszystkich kandydatów z datą decyzji aktualnej rekrutacji, a następnie wyświetlić kierunek, formę oraz stopień studiów, imię, nazwisko oraz numer teczek. W zapytaniu także potrzebne będzie posortowanie po polach które będą grupowane w programie uzupełniającym szablon. Każde na tych polach w zapytaniu SQL użyjemy polecenia ORDER BY. Ostatecznie poinformujemy program aby otrzymane dane grupował po 3 pierwszych polach:

```
\iffalse@@listarsekretariat@3@@
SELECT stopien_studiow_my,kierunek_my,forma_studiow_my,
      nazwiskoimiona,stud_nrteczki
FROM kandydat_aligeza
```

```
WHERE data_decyzji=(select wartosc from setup_aligeza where klucz='dataW
ORDER BY kierunek_forma_skrot_my, nazwiskoimiona
@END@\fi
```

W szablonach znajdują się jeszcze 3 kolejne podobne listy: przyjętych, nieprzyjętych oraz rankingowa. Zasada utworzenia zapytań do tych list jest identyczna jak w liście do sekretariatu. Różnią się one będą jedynie pewnymi warunkami wewnątrz zapytań SQL czy też polami po których będą sortowane dane.

Lista przyjętych:

```
\iffalse@@listaprzyjetych@3@@
SELECT stopien_studiow_my,kierunek_my,forma_studiow_my,
        nazwiskoimiona
FROM kandydat_aligeza
WHERE czy_przyjety=1 and
        data_decyzji=(SELECT wartosc FROM setup_aligeza WHERE klucz='dataW
ORDER BY kierunek_forma_skrot_my,
        nazwiskoimiona
@END@\fi
```

Lista nieprzyjętych:

```
\iffalse@@listaprzyjetych@3@@
SELECT stopien_studiow_my,kierunek_my,forma_studiow_my,
        nazwiskoimiona
FROM kandydat_aligeza
WHERE czy_przyjety=2 and
        data_decyzji=(SELECT wartosc FROM setup_aligeza WHERE klucz='dataW
ORDER BY kierunek_forma_skrot_my,
        nazwiskoimiona
@END@\fi
```

Lista rankingowa:

```
\iffalse@@listarankingowa@3@@
SELECT stopien_studiow_my,kierunek_my,forma_studiow_my,
        nazwiskoimiona,STUD_ILPUNKTOW
FROM kandydat_aligeza
WHERE data_decyzji=(select wartosc from setup_aligeza where klucz='dat
ORDER BY kierunek_forma_skrot_my, STUD_ILPUNKTOW desc
@END@\fi
```

3.3. Przebieg generowania raportów

Proces uruchamiania oraz przebiegu

3.4. Wyniki testu

Krótki opis wyników testu.

Bibliografia