

08.06.2017

Sprawdzanie obecności w laboratorium z wykorzystaniem legitymacji studenckiej

Przygotowali:
Patryk Smół 122097
Marcin Skorupiński 116906
Jakub Drapiewski 122093

07.06.2017

Spis treści

1	Opis projektu.....	4
2	Uzasadnienie wyboru tematu projektu	4
3	Zastosowane narzędzia oraz technologie	4
4	Uzasadnienie użytych technologii.....	4
4.1	MySQL.....	4
4.2	Java SE.....	4
4.3	Język Delphi	5
5	Podział prac.....	5
6	Funkcjonalność systemu	5
7	Implementacja strategicznych modułów.....	6
7.1	Schemat bazy danych	6
7.2	Ciąg komend umożliwiających operowanie na drzewie plików SmartCard.....	6
7.3	Model Studenta reprezentowany w aplikacji prowadzącego:	7
7.4	Model Zajęć reprezentowany w aplikacji prowadzącego:	8
7.5	Model reprezentujący podsystem obecności:	8
7.6	Model reprezentujący dane dotyczące prowadzącego.....	9
8	Informacje o elektronicznej legitymacji studenta	10
9	Wykorzystanie elektronicznej legitymacji studenckiej	11
10	Opis sposobu działania aplikacji klienckiej	11
11	Instrukcja użytkowania programu prowadzącego	11
11.1	Podgląd główny	12
11.2	Studenci	13
11.3	Sprawdź obecność	15
11.4	Raport obecności	16
11.5	Moje zajęcia.....	17
11.6	Modyfikacja obecności studenta.....	19
12	Wymagania sprzętowe	20
13	Wymagane oprogramowanie.....	20
14	Procedury przewidziane dla administratora systemu.....	20
14.1	Instalacja bazy danych	20
14.2	Aplikacja prowadzącego	20

14.3	Aplikacja studenta (hosta)	20
15	Możliwości dalszego rozwoju aplikacji.....	20
16	Podsumowanie.....	21
16.1	Zrealizowane cele	21
16.2	Niezrealizowane cele	21
17	Bibliografia	21

1 Opis projektu

System pozwala na sprawdzanie obecności studentów na zajęciach z wykorzystaniem legitymacji studenckiej. System składa się z trzech głównych modułów:

- Aplikacji prowadzącego
- Aplikacji działającej po stronie klienta
- Bazy danych

2 Uzasadnienie wyboru tematu projektu

Dany temat projektu został wybrany z uwagi na doświadczenie zespołu w projektowaniu podobnych rozwiązań. Projekt wydawał się również ciekawy ze względu na możliwość praktycznego zastosowania rozwiązania na zajęciach laboratoryjnych.

3 Zastosowane narzędzia oraz technologie

Technologie

- MySQL
- Java 8 SE
- Delphi Language

Narzędzia:

- Delphi RAD Studio XE 10.2 Tokio Edition
- PhpMyAdmin
- IntelliJ IDEA

4 Uzasadnienie użytych technologii

4.1 MySql

Jako serwer baz danych został użyty MySQL. Wydał się on najlepszym rozwiązaniem do tego celu. Posiadaliśmy wykupiony hosting, który jak większość posiadał obsługę baz danych MySQL. Dzięki temu baza jest scentralizowana i może być dostępna w każdej chwili z dowolnego miejsca.

4.2 Java SE

Język Java został wykorzystany do stworzenia demona mającego działać po stronie użytkowników (hostów) znajdujących się w salach laboratoryjnych. Język ten daje możliwość stworzenia lekkiej, przenośnej aplikacji działającej w tle. Java udostępnia również przyjazny interfejs do obsługi technologii SmartCard. Aplikacja

tak skonstruowana daje możliwość bezproblemowej pracy zarówno na systemach z rodziny Windows jak i Unix-pochodnych.

4.3 Język Delphi

Język Delphi umożliwił nam stworzenie nowoczesnej aplikacji przeznaczonej dla prowadzącego. Delphi jako jeden z niewielu języków zapewnia zaawansowany interfejs do obsługi baz danych jak i bogatą bibliotekę komponentów dających możliwość szybkiego tworzenia interfejsu programu na zasadzie *drag and drop*.

5 Podział prac

Tabela 1. Podział zadań na członków zespołu

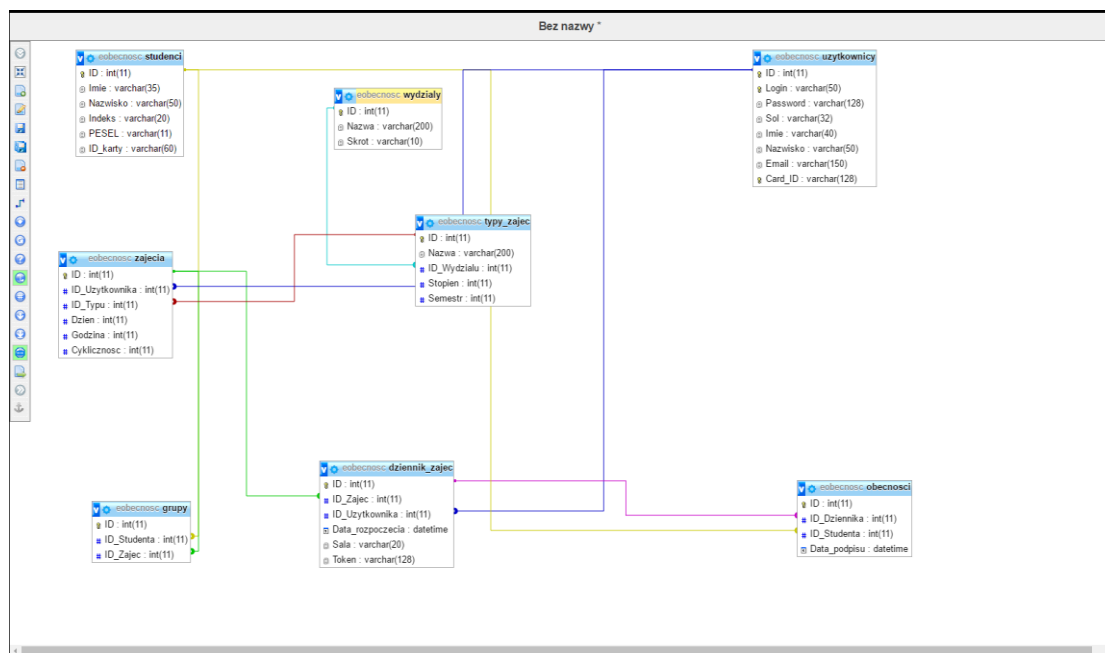
Członek zespołu	Zadania
Patryk Smół	Przygotowanie sprawozdania Implementacja aplikacji prowadzącego Przygotowanie bazy danych
Jakub Drapiewski	Przygotowanie sprawozdania Implementacja aplikacji prowadzącego Implementacja aplikacji działającej po stronie klienta
Marcin Skorupiński	Przygotowanie sprawozdania Implementacja aplikacji działającej po stronie klienta Przygotowanie bazy danych

6 Funkcjonalność systemu

- Automatyczna konfiguracja zainstalowanego czytnika *SmartCard*.
- Automatyczne wykrywanie momentu rozpoczęcia sprawdzania obecności.
- Praca w tle gwarantująca szybką i bezproblemową obsługę
- Obsługa kodowania UTF-8
- W pełni zautomatyzowany system sprawdzania obecności
- Możliwość ręcznego dodawania obecności
- Globalna baza studentów dostępna dla wszystkich prowadzących
- Podgląd obecności na zajęciach danego studenta
- Historia zajęć
- Możliwość usprawiedliwiania nieobecności na zajęciach

7 Implementacja strategicznych modułów

7.1 Schemat bazy danych



Rysunek 1. Schemat bazy danych

7.2 Ciąg komend umożliwiających operowanie na drzewie plików

SmartCard

Listing 1. Tablice bajtów umożliwiające odczytanie informacji z odpowiednich plików

```
private static final byte[] SELECT_MF = {(byte) 0x00, (byte) 0xA4, (byte) 0x00, (byte) 0x0C, (byte) 0x02, (byte) 0x3F, (byte) 0x00};
private static final byte[] SELECT_DF_T0 = {(byte) 0x00, (byte) 0xA4, (byte) 0x04, (byte) 0x00, (byte) 0x07, (byte) 0xD6, (byte) 0x16, (byte) 0x00, (byte) 0x00, (byte) 0x30, (byte) 0x01, (byte) 0x01};
private static final byte[] SELECT_DF_T1 = {(byte) 0x00, (byte) 0xA4, (byte) 0x04, (byte) 0x04, (byte) 0x07, (byte) 0xD6, (byte) 0x16, (byte) 0x00, (byte) 0x00, (byte) 0x30, (byte) 0x01, (byte) 0x01};
private static final byte[] SELECT_ELS_T0 = {(byte) 0x00, (byte) 0xA4, (byte) 0x00, (byte) 0x00, (byte) 0x02, (byte) 0x00, (byte) 0x02};
private static final byte[] SELECT_ELS_T1 = {(byte) 0x00, (byte) 0xA4, (byte) 0x02, (byte) 0x04, (byte) 0x02, (byte) 0x00, (byte) 0x02};
private static final byte[] READ_ELS = {(byte) 0x00, (byte) 0xB0, (byte) 0x00, (byte) 0x00, (byte) 0xFF};
```

7.3 Model Studenta reprezentowany w aplikacji prowadzącego:

Listing 2. Najważniejsze zmienne, funkcje i procedury modelu Student

```
type
TStudent = class
  ID: Integer;
  Imie: String;
  Nazwisko: String;
  Indeks: String;
end;
TStudentModel = class
public
  function GetStudentData(var Query: TFDQuery; ID: Integer) : TStudent;
  function ChangeStudentData(var Query: TFDQuery; Student: TStudent) : Boolean;
  function DeleteStudent(var Query: TFDQuery; ID: Integer) : Boolean;
  function AddStudent(var Query: TFDQuery; Student: TStudent) : Boolean;
  function IsPresence(var Query: TFDQuery; Student: TStudent; StudentID, RaportID: Integer)
: Boolean;
  function IsExist(var Query: TFDQuery; Indeks: String) : Boolean;
  procedure GetRaports(var Query: TFDQuery; LessonID: Integer);
  procedure GetLessons(var Query: TFDQuery; UserID: Integer; TeacherID: Integer);
  procedure Search(var Query: TFDQuery; Category: String; Value: String);
end;
```

Model ten umożliwia operowanie na zbiorze studentów. Pozwala na wykonanie operacji typu: Dodaj Studenta, Modyfikuj Studenta, Usuń Studenta oraz operacji z nim związanych typu: Sprawdzenie obecności, Wyszukiwanie itp. Interfejs reprezentuje również informacje o poszczególnym studencie w postaci rekordu.

7.4 Model Zajęć reprezentowany w aplikacji prowadzącego:

Listing 3. Najważniejsze zmienne, funkcje i procedury modelu Zajęć

```
type
TLesson = class
  ID: Integer;
  ID_Uzytkownika: Integer;
  Nazwa: String;
  Dzień: String;
  Godzina: String;
  Cyklicznosc: String;
  Wydział: String;
end;

TLessonModel = class
public
  procedure GetStudentsInLesson(var Query: TFDQuery; LessonID: Integer);
  procedure GetLessons(var Query: TFDQuery; UserID: Integer);
  function GetLessonData(var Query: TFDQuery; ID: Integer) : TLesson;
  function ChangeLessonData(var Query: TFDQuery; Lesson: TLesson) : Boolean;
  function DeleteLesson(var Query: TFDQuery; ID: Integer) : Boolean;
  function AddLesson(var Query: TFDQuery; Lesson: TLesson) : Boolean;
  function AddToLesson(var Query: TFDQuery; StudentID: Integer; LessonID: Integer) :
Boolean;
  function DeleteFromLesson(var Query: TFDQuery; StudentID: Integer; LessonID: Integer) :
Boolean;
end;
```

Model ten umożliwia operowanie na zbiorze zajęć. Pozwala na wykonanie operacji typu: Dodaj Zajęcia, Modyfikuj Zajęcia, Usuń Zajęcia. Dana implementacja pozwala na operacje związane ze studentami uczestniczącymi w danych zajęciach.

7.5 Model reprezentujący podsystem obecności:

Listing 4. Najważniejsze zmienne, funkcje i procedury podsystemu Obecności

```
type
TPresenceModel = class
private
  TempList: TStringList;
public
  function GetIPAddress : String;
  function AddToList(Indeks: String) : Boolean;
  function Accept(var Query: TFDQuery; DziennikID: Integer) : Boolean;
  function StartLesson(var Query: TFDQuery; LessonID: Integer; UserID: Integer; Sala:
String) : Integer;

  procedure ShowPresence(var Query: TFDQuery);
  procedure GetLog(var Query: TFDQuery; ID_Zajec: Integer);
  procedure Present(var Query: TFDQuery; ID_Dziennika: Integer);
  procedure unPresent(var Query: TFDQuery; ID_Dziennika: Integer);

  constructor Create(); overload;
end;
```


Dany model reprezentuje system obecności, Umożliwia na nim dokonywanie podstawowych operacji. Jest kluczowy system w celu sprawdzenia obecności studentów na określonych zajęciach.

7.6 Model reprezentujący dane dotyczące prowadzącego

Listing 5. Najważniejsze zmienne, funkcje i procedury dotyczące obsługi konta prowadzącego

```
type
  TUser = class
    ID: Integer;
    Login: String;
    Name: String;
    Surname: String;
    Email : String;
  end;

  TUserModel = class
  private
    FDQuery: TFDQuery;

    function GenSalt(Len: Integer) : String;
    function HashPassword(Passwd, Salt: String) : String;
  public
    constructor Create();

    function GetUserData(UserID: Integer) : TUser;
    function RegisterUser(User, Password, Name, Surname, Email: String) : Boolean;
    function ChangeUserData(UserID : Integer; Name, Surname, Email: String): Boolean;
    function ChangePassword(UserID: Integer; Password: String) : Boolean;
    function DeleteUser(UserID : Integer) : Boolean;

    function CheckUserName(Login: String) : Boolean;
    function CheckLogin(Login, Password: String) : Integer;
  end;
```

Klasa *TUserModel* umożliwia reprezentowanie ustawień oraz danych użytkownika (prowadzącego) przez model. Umożliwia on utworzenie konta, pobieranie danych użytkownika oraz sprawdzenie w przypadku rejestracji czy dany użytkownik już istnieje.

Przedstawione modele stanowią podstawę dalszego rozwoju aplikacji. Format kodu jest elastyczny umożliwiając dalszy rozwój. Każdy developer chcący rozbudować funkcjonalność danej aplikacji może bez problemu dokonać tego poprzez rozwój odpowiedniego z modeli reprezentujących dany segment systemu.

7.7 Funkcja ustawiająca protokół, który działa na karcie

Listing 6. Ustawianie odpowiedniego protokołu

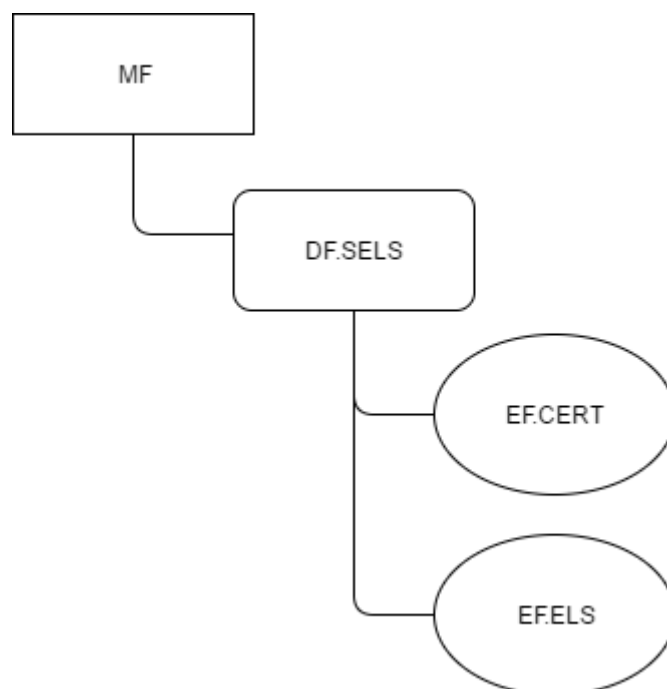
```
private void SetCardMode(String Protocol) {
    if (Protocol == null || Protocol.isEmpty()) {
        Protocol = "T=0"; //Set default protocol
    }

    try {
        this.StudentCard = this.Terminal.connect(Protocol);
        this.Channel = this.StudentCard.getBasicChannel();
    } catch (CardException e) {
        if (Protocol.equals("T=0")) {
            SetCardMode("T=1"); //If T="0" doesn't work, try T="1"
        } else {
            this.STATUS = false;
            System.err.println("Błąd połączenia z kartą");
        }
    }
}
```

Funkcja ta automatycznie zmienia protokół jeśli legitymacja posiada inny niż domyślnie ustawiony w programie.

8 Informacje o elektronicznej legitymacji studenta

Elektroniczna legitymacja studenta to dokument identyfikacyjny danego studenta wyposażony w interfejs stykowy oraz opcjonalnie w interfejs zbliżeniowy. Na karcie tej znajdują się informacje dotyczące imienia, nazwiska, numeru pesel, adresu zamieszkania oraz numeru indeksu oraz cyfrowe zdjęcie. Dane zgodnie ze standardem ISO 7816 ułożone są w drzewiastą strukturę katalogów DF (Directory File) oraz plików EF (Elementary File). Każdy plik zapisany na karcie posiada 2 bajtowy adres. Rolę korzenia pełni plik MF, do którego to muszą być przyłączone katalogi oraz pliki. W ramach projektu wykorzystaliśmy katalog DF.SELS zawierający dwa pliki: EF.CERT-kwalifikowany certyfikat emitenta oraz EF.ELS zawierający plik z danymi studenta. Strukturę drzewiastą plików legitymacji wykorzystanych w projekcie przedstawiono na rys 2.



Rysunek 2. Struktura dostępu do plików wykorzystywanych w projekcie

9 Wykorzystanie elektronicznej legitymacji studenckiej

Elektroniczną legitymację studenta wykorzystujemy do pobrania numeru indeksu przez Deamona zainstalowanego na komputerze studenta. Aplikacja ta uruchamiana jest automatycznie przy logowaniu się studenta do systemu. Numer indeksu następnie przesyłany jest do aplikacji prowadzącego przy pomocy protokołu UDP

10 Opis sposobu działania aplikacji klienckiej

Aplikacja kliencka działa w tle, po otrzymaniu informacji od aplikacji prowadzącego o rozpoczęciu sesji sprawdzania obecności na zajęciach, wyświetla stosowny komunikat. Aplikacja oczekuje na umieszczenie przez studenta legitymacji w czytniku kart. Po umieszczeniu legitymacji i odczytaniu z niej danych studenta aplikacja przesyła uzyskana dane do aplikacji prowadzącego.

11 Instrukcja użytkowania programu prowadzącego

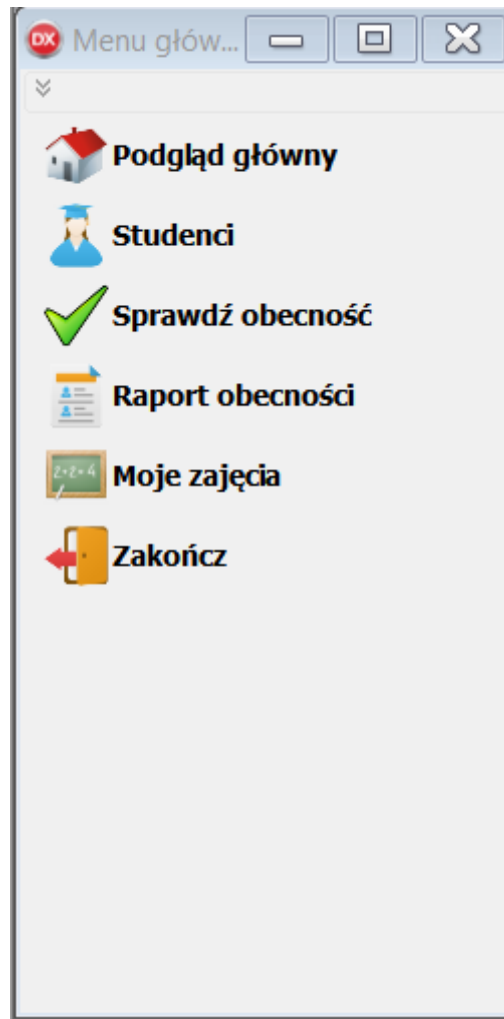
Funkcjonalność programu została podzielona na moduły, których wywołanie umieszczono w przyciskach: "Podgląd główny", "Studenci", "Sprawdź obecność", "Raport obecności", "Moje zajęcia", "Zakończ" panelu sterowania przedstawionego na rysunku 3. Każde wywołane okno wyposażone jest w pasek nazwy zawierający tytuł wybranej funkcjonalności oraz 3 przyciski: minimalizuj, przywróć okno, zamknij okno.

Opis przycisków paska nazwy:

- Minimalizuj - okno zostaje zmniejszone do formy przycisku na pasku zadań Systemu Windows
- Maksymalizuj - okno zostaje rozciągnięte na cały ekran
- Zamknij okno - zamyka okno

Maksymalizacja okna powoduje zmianę przycisku “Maksymalizuj” na przycisk “Przywróć”. Przycisk ten umożliwia przywrócenie oknu wcześniejszego rozmiaru.

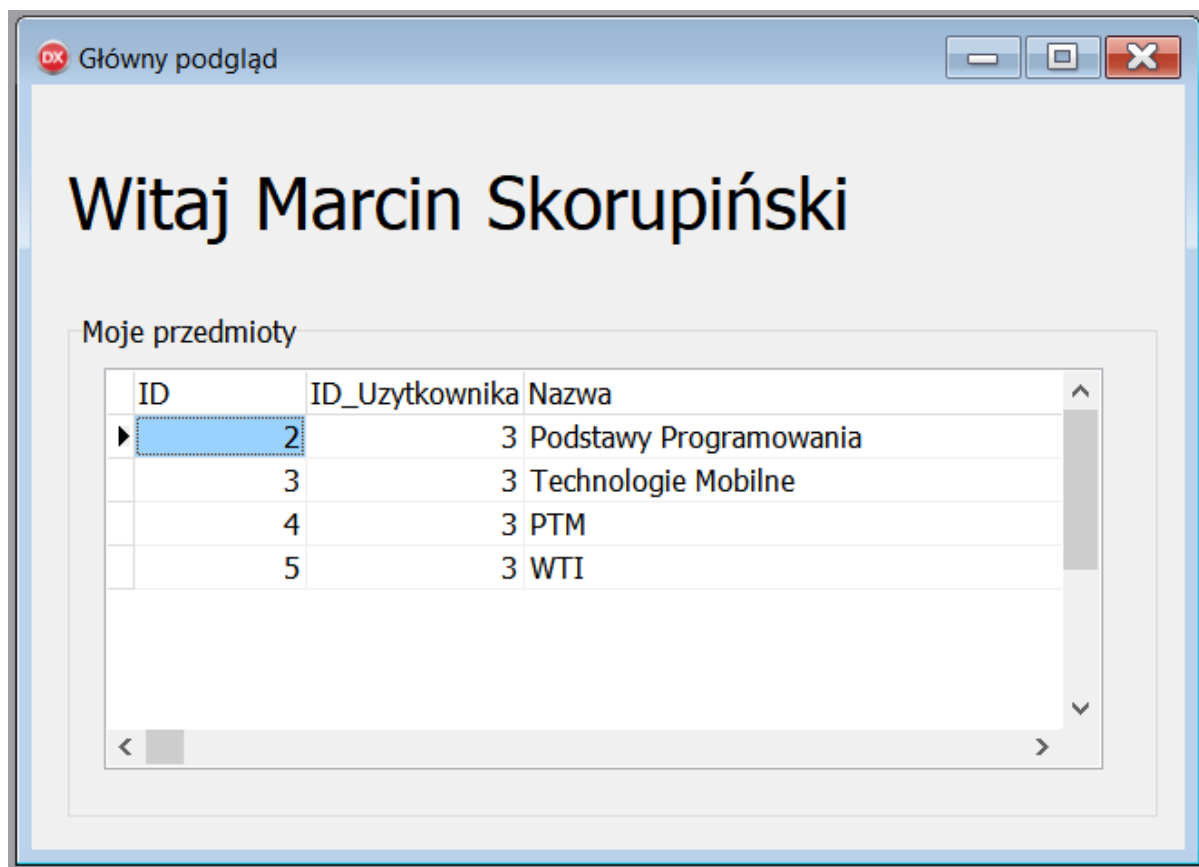
Strona główna pozwala wyświetlić kilka skalowalnych okien.



Rysunek 3. Panel sterowania programu prowadzącego

11.1 Podgląd główny

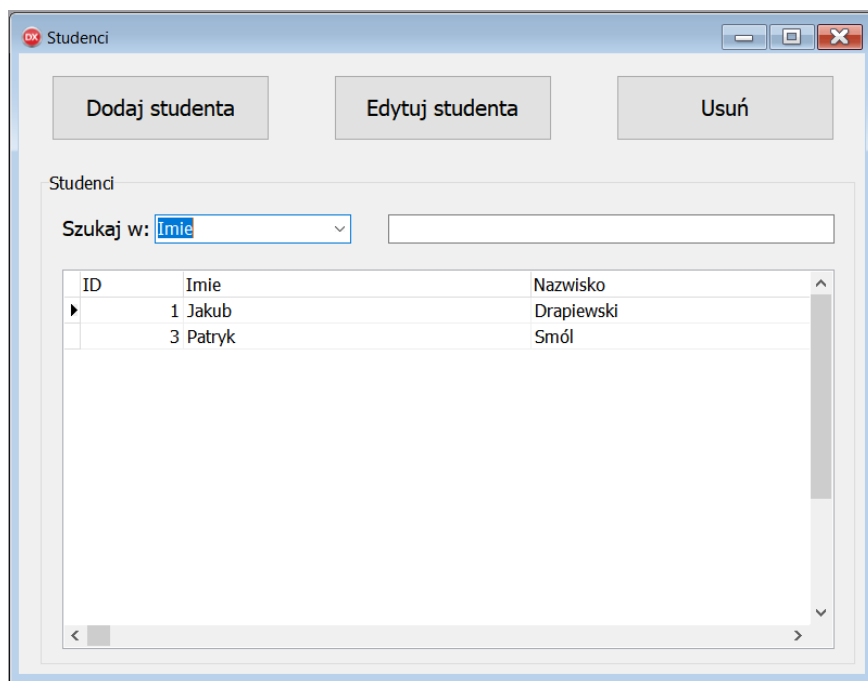
Zakładka “Podgląd główny” daje możliwość wyświetlenia przedmiotów prowadzonych przez zalogowanego użytkownika. Przedmioty pobierane są z bazy danych w czasie inicjalizowania okna i wpisywane do tabeli. Nad tabelą znajduje się etykieta zawierająca napis powitalny wraz z imieniem i nazwiskiem zalogowanej osoby. Przykładowe wypełnienie tabeli zostało przedstawione na rys 4.



Rysunek 4. Przykładowe wypełnienie tabeli przedmiotami

11.2 Studenci

Okno "Studenci" wyposażono w trzy przyciski umożliwiające: dodanie danych o nowym studencie, edytowanie danych i usunięcie danych. Dane o studentach pobierane są w czasie inicjalizacji okien, a następnie wyświetlane w tabeli. Okno zostało wyposażone w pasek inteligentnego wyszukiwania. Pasek ten wyszukuje i wyświetla sugerowanych studentów na podstawie wprowadzanych przez prowadzącego liter oraz wybranej kategorii.



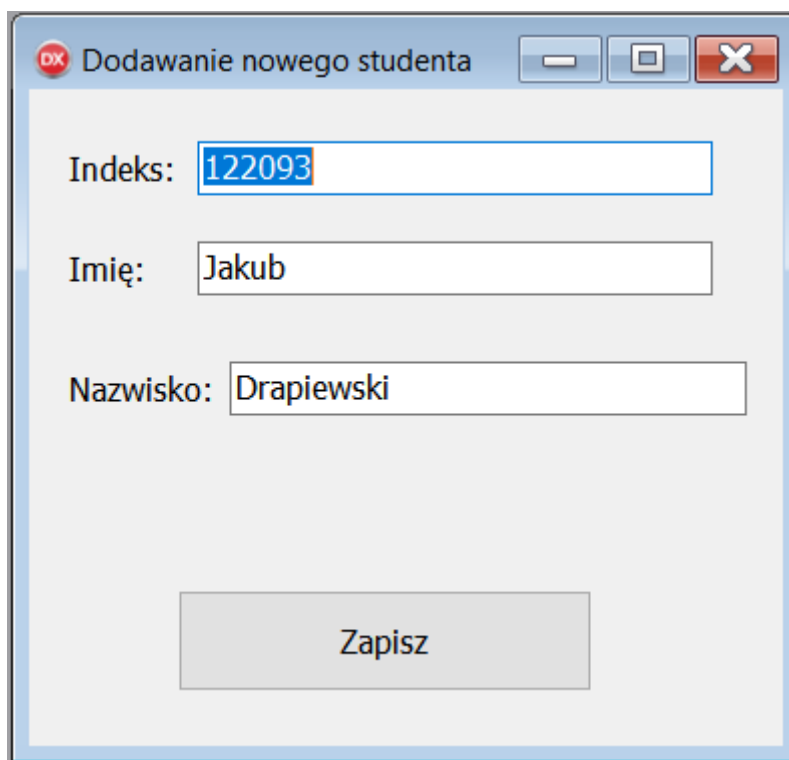
Rysunek 5. Okno zakładki Studenci

Dodawanie informacji o studencie odbywa się poprzez wybór przycisku Dodaj studenta interfejsu Studenci. Zostanie otwarte nowy widok, w którego odpowiednich polach należy wpisać imię, nazwisko oraz indeks. Zaakceptowanie wprowadzonych danych i dodanie ich do bazy danych zostanie wykonane po przyciśnięciu przycisku Zapisz. Wygląd okna Dodawanie nowego studenta pokazano na rysunku rys...

The screenshot shows a window titled 'Dodawanie nowego studenta'. It contains three input fields with labels: 'Indeks:', 'Imię:', and 'Nazwisko:'. Each label is followed by a text input box. At the bottom of the window, there is a large button labeled 'Zapisz'.

Rysunek 6. Okno dodawania nowego studenta do bazy danych

Edycję danych dotyczących określonego studenta można przeprowadzić poprzez zaznaczenia pozycji w tabeli i kliknięcie przycisku edytuj. W nowym oknie zostaną dodane do pól odpowiednie dane wybranego studenta. Dane te można edytować, a następnie nacisnąć przycisk Zapisz. Czynność ta spowoduje wprowadzenie odpowiednich zmian w bazie danych. Interfejs edycji studenta zademonstrowano na rysunku 7.



The image shows a software window titled "Dodawanie nowego studenta" (Adding a new student). It features three text input fields for data entry: "Indeks:" (Index) containing "122093", "Imię:" (First name) containing "Jakub", and "Nazwisko:" (Surname) containing "Drapiewski". Below these fields is a prominent button labeled "Zapisz" (Save). The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

Rysunek 7. Okno edycji studenta

11.3 Sprawdź obecność

Okno sprawdź obecność służy do rozpoczęcia sesji sprawdzającej obecność na zajęciach. Należy wybrać zajęcia z tabeli, można również wprowadzić salę w której odbywają się te zajęcia oraz zmienić domyślny czas (5 minut) trwania sesji sprawdzającej obecność. W celu rozpoczęcia sprawdzania obecności należy kliknąć w przycisk "Rozpocznij Sprawdzanie". Sesję można anulować w trakcie jej trwania, używając przycisku "Anuluj". Program zapyta czy zapisać anulowaną sesję.

Sprawdź obecność

Sprawdzanie obecności

Czas sprawdzania obecności: 5

Sala:

Zajęcia:

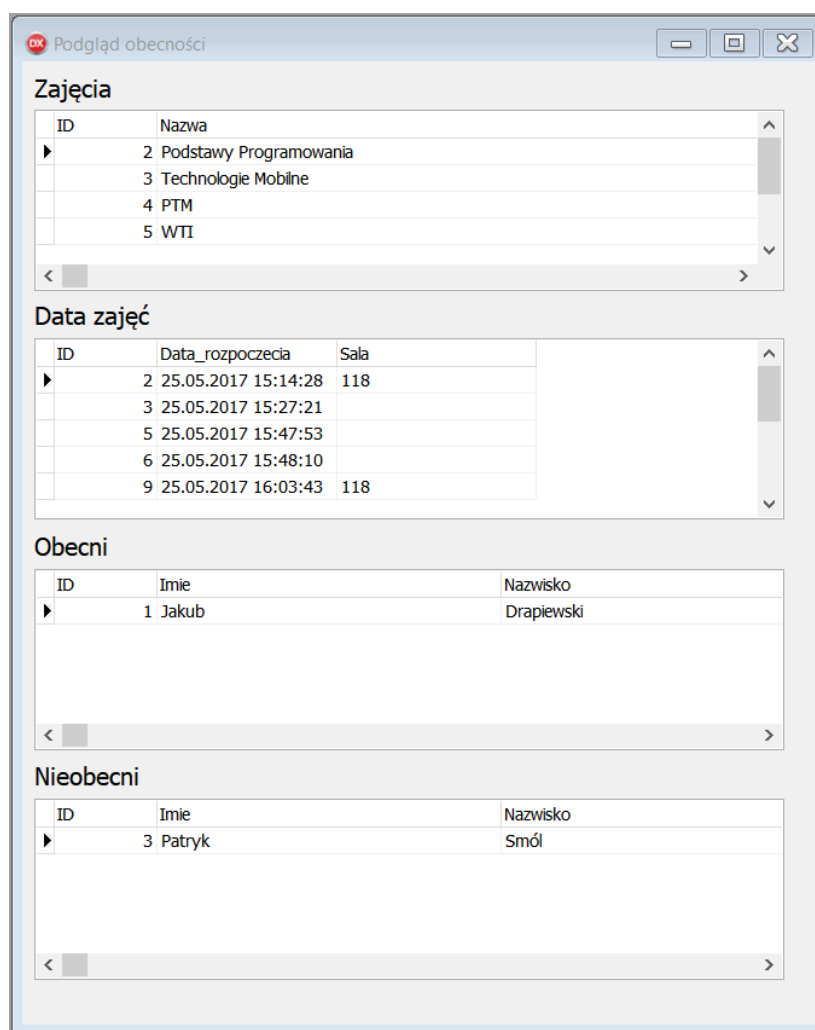
ID	Nazwa
2	Podstawy Programowania
3	Technologie Mobilne
4	PTM
5	WTI

Rozpocznij Sprawdzanie Anuluj

Rysunek 8. Okno sprawdzania obecności

11.4 Raport obecności

Panel “Podgląd obecności” pozwala na wyświetlenie listy obecnych i nieobecnych studentów na danych zajęciach. W tym celu należy zaznaczyć interesujące nas zajęcia z tabeli “Zajęcia”, następnie wybrać datę zajęć. W dwóch dolnych tabelach zostaną wyświetlone listy obecnych i nieobecnych studentów na danych zajęciach.



Rysunek 9. Okno podglądu obecności

11.5 Moje zajęcia

Panel "Zajęcia" pozwala na dodanie bądź usunięcie studenta z wybranych zajęć, dodanie/modyfikację/usunięcie zajęć.

W celu dodania studenta do zajęć należy wybrać zajęcia na które chcemy dodać studenta i użyć przycisku "Dodaj studenta", zostanie wyświetlony panel ze wszystkimi studentami dostępnymi w bazie danych. Należy wybrać studenta i kliknąć "Dodaj", student zostanie dodany do wcześniej wybranych zajęć.

Usunąć studenta z zajęć można poprzez wybranie go i użycie przycisku "Usuń", aplikacja poprosi o potwierdzenie usunięcia.

Chcąc dodać nowe zajęcia należy posłużyć się przyciskiem "Dodaj nowe zajęcia", zostanie wyświetlony panel przedstawiony na poniższym zdjęciu, należy go wypełnić i kliknąć zapisz, a zajęcia zostaną dodane.

Dodaj / Modyfikuj zajęcia

Nazwa:

Dzień:

Godzina:

Cykliczność:

Wydział:

Zapisz Zamknij

Rysunek 10. Okno dodawania zajęć

Modyfikacja zajęć odbywa się w tym samym panelu co dodawanie zajęć. W celu modyfikacji wybieramy zajęcia i klikamy "Modyfikuj zajęcia", edytujemy interesujące na pola i używamy przycisku "Zapisz".

Zajęcia

ID	Nazwa
2	Podstawy Programowania
3	Technologie Mobilne

Dodaj nowe zajęcia Modyfikuj zajęcia Usuń wybrane zajęcia

ID	Imię	Nazwisko	Indeks
1	Jakub	Drapiewski	122
3	Patryk	Smół	122

Dodaj studenta Usuń Studenta Odśwież

Rysunek 11. Okno obecności studentów na zajęciach

Dodaj / Modyfikuj zajęcia

Nazwa:

Dzień:

Godzina:

Cykliczność:

Wydział:

Zapisz Zamknij

Rysunek 12. Okno modyfikowania zajęć

11.6 Modyfikacja obecności studenta

W celu dokonania modyfikacji obecności studenta na zajęciach należy wejść w panel “Studenci”, następnie dwukrotnie kliknąć wiersz z danymi studenta dla którego chcemy dokonać modyfikacji obecności, zostanie otwarty panel indywidualny studenta w którym możemy dokonać modyfikacji obecności na wybranych zajęciach poprzez wybranie zajęć, a następnie dwukrotne kliknięcie na obecność którą chcemy zmienić.

122093 Jakub Drapiewski

Przedmiot:

ID	ID_Uzytkownika	Nazwa
3	3	Technologie Mobilne
4	3	PTM

Był

ID	Data_roz poczenia	Sala
12	25.05.2017 17:06:44	118
13	30.05.2017 15:55:48	

Nie był

ID	Data_roz poczenia	Sala

Rysunek 13. Okno obecności wybranego studenta na zajęciach

12 Wymagania sprzętowe

Do działania aplikacji klienckiej konieczny jest czytnik kart elektronicznych pozwalający na odczytanie danych studenta z legitymacji.

13 Wymagane oprogramowanie

Aplikacja odczytująca dane o studencie z elektronicznej legitymacji studenckiej wymaga zainstalowanego środowiska Java w wersji minimum 8. Ze względów bezpieczeństwa zaleca się używanie najnowszej wersji środowiska

14 Procedury przewidziane dla administratora systemu

14.1 Instalacja bazy danych

- System wymaga serwera MySQL w wersji co najmniej 5.5
- Baza powinna być dostępna na porcie 3306
- Baza danych powinna być dostępna z zewnątrz

14.2 Aplikacja prowadzącego

- Aplikacja powinna mieć dostęp do podsieci zawierającej wszystkie hosty laboratoryjne.
- Prowadzący korzystający z aplikacji powinien mieć bezproblemowy dostęp do sieci Internet nie blokowany żadnymi programami.

14.3 Aplikacja studenta (hosta)

- Aplikacja powinna mieć bezproblemowy dostęp do sieci Internet.

15 Możliwości dalszego rozwoju aplikacji

Wzbogacenie aplikacji o import i eksport danych.

16 Podsumowanie

16.1 Zrealizowane cele

- Możliwość sprawdzenia obecności
- Ręczna edycja obecności
- Dodawanie/usuwanie zajęć

16.2 Niezrealizowane cele

- Import/eksport studentów

17 Bibliografia

- Marek Goślawski, "Laboratorium Programowania Kart Elektronicznych",
http://www2.mcp.poznan.pl/?q=system/files/LabPKE_03+-+ELS.pdf
- Oracle, Java™ Smart Card I/O API
<https://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/>
- ISO/IEC7816-4
[http://www.embedx.com/pdfs/ISO STD 7816/info isoiec7816-4%7Bed21.0%7Den.pdf](http://www.embedx.com/pdfs/ISO_STD_7816/info_isoiec7816-4%7Bed21.0%7Den.pdf)
- Makdaam, "Czytanie ELS przez interfejs stykowy"
<http://www.makdaam.eu/2008/04/czytanie-els-przez-interfejs-stykowy/>